POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Informatica

Tesi di Laurea Magistrale

Sviluppo di un sistema di raccomandazione non personalizzato per prodotti

Relatori
prof. Alessandro Fiori

Candidato
Andrea Gron

Anno Accademico 2021-2022

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

Con l'avvento dei servizi di streaming e degli acquisti online, la possibilità di poter dare suggerimenti e raccomandazioni all'utente finale è diventato un fattore basilare per conseguire migliori risultati e avere utenti più soddisfatti.
Questa Tesi Magistrale, si è concentrata sullo sviluppo di un sistema raccomandazione non personalizzato di prodotti per la piattaforma *www.technobattles.com*.
In questa piattaforma non è presente un metodo per conferire un punteggio ai singoli prodotti. Per questo è stato sviluppato un sistema di calcolo per definire uno score per i prodotti, basato sul valore delle specifiche che essi hanno, come ad esempio: il peso di una fotocamera, lo zoom massimo che ha o il massimo dei megapixel che ha.
All'inserimento del sistema di raccomandazione degli oggetti attribuendo un punteggio, è stato anche inserito nella piattaforma un sistema di valutazione basato sulla popolarità che i prodotti hanno per gli utenti, ovvero basato sul numero di volte in cui l'oggetto viene visionato.

1.1 Panoramica sulla tesi

In questo documento verranno presentati tutti gli argomenti fondamentali e le scelte progettuali fatte per raggiungere gli obiettivi preposti. In particolare nel Capitolo 2 si definisce un contesto teorico riferito agli argomenti introdotti per l'implementazione della piattaforma. Verranno spiegati in modo approfondito i flussi ETL e i sistemi di raccomandazione.
Per quanto riguarda le ETL verrà fornita una spiegazione dei vari passaggi che compongono il processo ETL, ovvero estrazione, trasformazione e caricamento. Verranno inoltre esaminate altre possibili alternative, anche più recenti, al classico modello ETL.
Per quanto riguarda i sistemi di raccomandazione, verranno introdotte le diverse tipologie esistenti nel mondo dell'IT e verranno mostrati dei casi di utilizzo in alcune delle più grandi compagnie tech del mondo.
Il Capitolo 3, si concentra sull'ambiente di sviluppo della piattaforma, ovvero sulle tecnologie utilizzate in questa Tesi a riguardo della piattaforma. Insieme alle diverse spiegazioni dei framework e applicazioni utilizzate saranno presenti sempre delle alternative alle tecnologie utilizzate.
Nel Capitolo 4 si indaga sul database utilizzato dalla piattaforma, mostrando in particolare le differenti collezioni modificate ed utilizzate in questa Tesi.
Il Capitolo 5, si concentra sulle implementazioni sviluppate per arrivare al completamento degli obiettivi preposti, ovvero il sistema di raccomandazione. Il primo passaggio è stato sviluppare il sistema di duplicazione e trasformazione delle specifiche utile al calcolo dello score per il sistema di raccomandazione. Successivamente sono state sviluppate le pagine per le gestione e la creazione degli score, la pagina di ricerca degli oggetti ed infine il processo di calcolo della popolarità dei singoli oggetti.
La Tesi si conclude con il Capitolo 6 dove vengono rappresentati i casi di utilizzo

all'interno della piattaforma. In pratica, vengono mostrate le varie pagine sviluppate e come l'utente può interagire, facendo riferimento alle parti implementate nel capitolo precedente.

Capitolo 2

Tematiche trattate

In questo capitolo verranno spiegate in maniera dettagliata le fondamenta teoriche su cui si basa il lavoro effettuato sulla piattaforma.
Per entrare nel merito dei sistemi introdotti, si parte dalla struttura teorica su cui si basa la piattaforma technobattles, in particolare indagando sui sistemi che la costituiscono: ETL e sistemi di raccomandazione.
Per quanto riguarda le ETL verrà fornita una spiegazione dei vari passaggi che compongono il processo ETL, ovvero estrazione, trasformazione e caricamento. Verranno inoltre fornite delle alternative, anche più recenti, al classico modello ETL.
Per quanto riguarda i sistemi di raccomandazione, verranno spiegate le diverse tipologie esistenti nel mondo dell'IT e verranno mostrati dei casi di utilizzo in alcune delle più grandi compagnie tech del mondo.

2.1 ETL

Una semplice definizione di Extract, Transform e Load (ETL[14]) potrebbe essere "l'insieme dei processi che consentono di trasferire i dati dai sistemi OLTP (On-Line Transaction Processing) a un data warehouse". Il processo ETL fa parte della Data Staging Area, come prima fase di ricezione dei dati derivati da fonti eterogenee, assicurando la qualità e la coerenza dei dati. In un data warehouse (DW), il processo ETL è la parte più dispendiosa in termini di tempo, tanto che si parla addirittura dell'80% del tempo totale di sviluppo del progetto. In quanto componente importante di un sistema di supporto decisionale orientato ai database, il DW memorizza i dati relativi alle attività e agli eventi correlati a un processo aziendale distinto. Il DW stesso, viene definito come un sistema che estrae, pulisce, conforma e fornisce i dati di origine in un archivio di dati dimensionali e quindi supporta e implementa l'interrogazione e l'analisi per il processo decisionale.
Nella costruzione del data warehouse, si può implementare l'approccio bottom-up di Kimball[7] nello sviluppo di un modello multidimensionale. Sulla base del processo di progettazione dimensionale in quattro fasi di Kimball, le attività di progettazione del modello si articolano nei seguenti passaggi:

- Selezionare i processi aziendali: da modellare, tenendo conto dei requisiti aziendali e delle fonti di dati disponibili.

- Dichiarare la grana[1]: definire una rappresentazione individuale della tabella dei fatti che si riferisce alle misure dei processi aziendali.

- Scegliere le dimensioni: determinare gli insiemi di attributi che descrivono le misure della tabella dei fatti. Le dimensioni tipiche sono tempo e data.

[1]Livello di dettaglio disponibile nello schema

- Identificare i fatti numerici: definire quali misure includere nelle tabelle dei fatti. I fatti devono essere coerenti con la grana dichiarata

L'importante vantaggio di questo approccio è costituito dai data mart[2], come rappresentazione dell'unità di business. Essi hanno una configurazione iniziale molto rapida e sono facili da integrare tra altri data mart.
Per quanto riguarda i DW su scala aziendale, questo metodo permette di estendere il DW esistente e di fornire anche una capacità di reporting.
Naturalmente questo processo può essere anche utilizzato per un normale db come nel caso di questa piattaforma.

2.1.1 Processo ETL

I processi ETL stanno diventando parte integrante di quasi tutte le applicazioni in uso, siccome i dati sono di importanza fondamentale per tutti i processi aziendali. Alcuni esempi di utilizzo dei processi ETL sono:

- Migrare i dati da un'applicazione ad un'altra senza ricorrere ad errori.
- Replicare insiemi di dati in modo da effettuare backup o analisi su di essi.
- Inserire i dati in DW, così da renderli disponibili per i vari processi aziendali.
- Migrare le applicazioni da locale ad infrastrutture Cloud.

Secondo Kimball[6] il flusso di dati in un processo ETL può essere diviso in 4 parti: Extract, Clean, Conform[3] e Deliver, ovvero estrarre i dati, pulirli, uniformarli e erogarli.

[2]Database strutturati in base all'argomento che rappresentano una parte del DW

[3]Solitamente le fasi di Clean e Conform vengono unite in un'unica sezione comunemente chiamata Transform

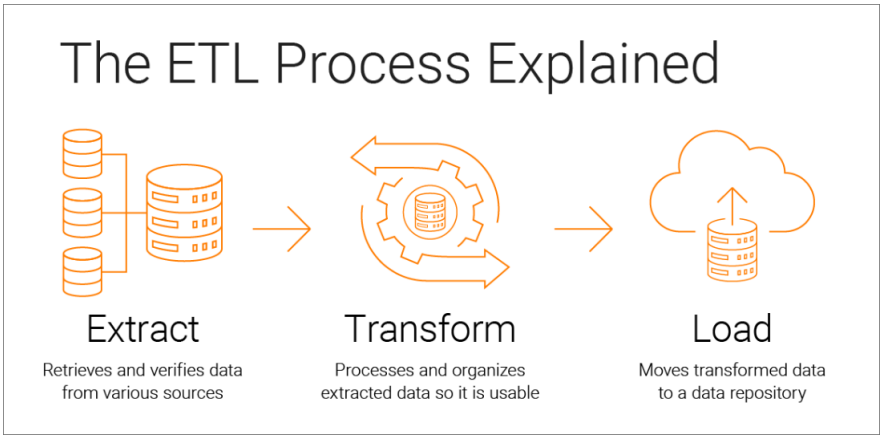

Figura 2.1. Processo ETL

Extract

La fase di estrazione è una delle più importanti, definisce quali e quanti dati vanno estratti dalle diverse fonti in modo da poterli successivamente pulire e infine utilizzare nel DW.
L'estrazione può essere effettuata da diverse fonti tra cui: altri database esistenti, siti web tramite web-scraper o API dedicate, report delle applicazioni.
I dati estratti da queste fonti sono solitamente in formato xml oppure direttamente estratti dalle risposte http, quindi sono ancora in formato grezzo e di fatto vengono poi trasformati nelle fasi successive di pulizia e conformità in modo da poter essere utilizzati.

Clean & Conform

La pulizia e la conformità sono le fasi principali in cui il sistema ETL aggiunge valore ai dati. Le altre fasi di estrazione e consegna sono ovviamente necessarie, ma si limitano a spostare e riformattare i dati. La pulizia e la conformità modificano effettivamente i dati e forniscono indicazioni sulla possibilità di utilizzarli per gli

scopi previsti.
Le fasi di pulizia e conformità generano metadati potenti. Guardando indietro verso le fonti originali, questi metadati sono una diagnosi di ciò che non funziona nei sistemi di origine. In definitiva, i dati "sporchi" possono essere risolti solo cambiando il modo in cui i sistemi di origine raccolgono i dati.
Una cosa fondamentale in questo passaggio è la qualità dei dati, in particolare l'accuratezza. Avere dei dati accurati significa che siano:

- **Corretti**: I valori e le descrizioni dei dati descrivono in modo veritiero e fedele gli oggetti ad essi associati.
- **Non ambiguo**: I valori e le descrizioni dei dati possono avere un solo significato.
- **Consistenti**: I valori e le descrizioni dei dati utilizzano una costante convenzione notazionale per trasmettere il loro significato.
- **Completi**: Le descrizioni e i valori individuali dei dati non sono nulli e inoltre il numero di record sia completo o che non si siano persi del tutto dei record in qualche punto del flusso di informazioni.

La trasformazione avviene tramite l'applicazione di una serie di regole definite a livello aziendale. Gli standard che assicurano l'accessibilità e la qualità dei dati durante questa fase dovrebbero includere:

- **Standardizzazione**: Quali dati che saranno presi in considerazione, come verranno formattati e memorizzati, così come di altri fattori chiave che andranno a definire le fasi successive del processo.
- **Deduplicazione**: Segnalazione di duplicazioni e successiva esclusione e/o eliminazione dei dati ridondanti.

- **Verifica**: Esecuzione di verifiche automatiche per mettere a confronto informazioni simili. Le attività di verifica consentono di pulire ulteriormente i dati inutilizzabili e di contrassegnare eventuali anomalie.
- **Ordinamento**: Ottimizzazione dell'efficienza all'interno dei DW tramite raggruppamento e ordinamento in categorie di elementi come dati grezzi, audio, multimediali e altri oggetti. Le regole di trasformazione determinano come ogni singolo dato viene classificato e dove sarà collocato nella fase successiva.

Deliver

Nell'ultima fase del processo ETL, si prevede il caricamento dei dati appena estratti e trasformati in una nuova destinazione, che può essere un DW o un db.
É possibile eseguire il caricamento sia tramite l'utilizzo di codice da riga di comando, sia tramite interfaccia grafica. Tuttavia, ci sono alcuni aspetti da tenere sotto controllo. Gestire le eccezioni, ad esempio, può essere estremamente impegnativo siccome spesso le estrazioni dei dati possono non andare a buon fine.

2.1.2 Alternative alle ETL

Le alternative al processo ETL classico comprendono principalmente: L'ELT e lo streaming ETL. Questi diversi sistemi si analizzano per valutare come possono impattare il sistema rispetto al modello classico di ETL.

ELT

ELT è l'acronimo di "Extract, Load, and Transform" e descrive le tre fasi della moderna pipeline di dati. Il processo ELT è più conveniente dell'ETL, è adatto a set di dati strutturati e non strutturati di grandi dimensioni e anche quando la velocità è importante.

Come si può notare in Fig. 2.2 tutti i dati vengono estratti dall'origine. In seguito vengono immediatamente caricati nel sistema di destinazione (data warehouse, data mart o data lake). Possono essere dati grezzi, non strutturati, semi-strutturati e strutturati. In fine i dati vengono trasformati nel sistema di destinazione e sono pronti per essere analizzati dagli strumenti di analisi dei dati.
I benefici delle ELT rispetto alle ETL sono:

- **L'analisi di dati flessibile e real-time**: non bisogna aspettare l'estrazione e la trasformazione dei dati siccome il dataset è già completo.

- **Manutenzione minore e costi minori**: utilizzando un ecosistema basato sul cloud i costi proposti sono molto più bassi e inoltre si ha a disposizione una maggiore varietà di opzioni di piano per archiviare ed elaborare i dati. Inoltre, il processo di ELT richiede in genere una manutenzione ridotta, dato che tutti i dati sono sempre disponibili e il processo di trasformazione è solitamente automatizzato e basato sul cloud.

Streaming ETL

Streaming ETL (Extract, Transform, Load) è l'elaborazione e lo spostamento di dati in tempo reale da un luogo all'altro.
Negli ambienti di dati tradizionali, il processo ETL estraeva dati da un sistema di origine, di solito in base a una pianificazione, trasformava i dati e li caricava in un repository come un data warehouse o un database. Questo è il modello "batch ETL" (cfr. Capitolo 3, Sezione 2.1). Poiché molti ambienti aziendali moderni non possono aspettare ore o giorni affinché le applicazioni gestiscano i diversi dati l'ETL deve rispondere ai nuovi dati in tempo reale, non appena questi vengono generati.

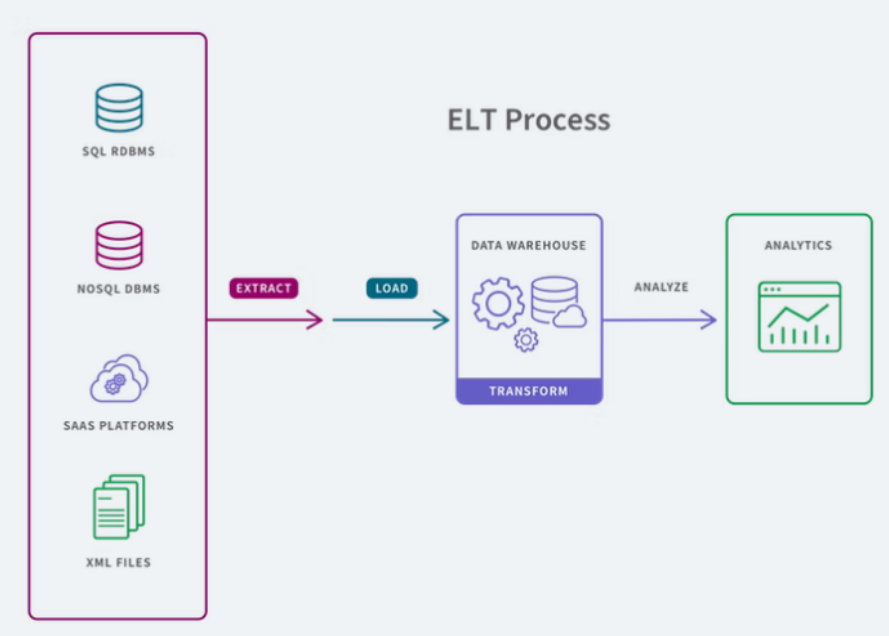

Figura 2.2. Processo ELT

Concettualmente, un'architettura ETL in streaming (o in tempo reale) è fondamentalmente uguale a un'architettura ETL tradizionale. I dati di origine alimentano un sistema che elabora e trasforma i dati da quell'origine, mentre l'output viene consegnato a una destinazione. Quando viene visualizzato sotto forma di diagramma, l'architettura presenta spesso l'origine a sinistra, il motore ETL al centro e la destinazione a destra come descritto in Fig. 2.3.
Le fonti alimentano i dati a una piattaforma di streaming. Questa piattaforma funge da spina dorsale per le applicazioni ETL di streaming, ma anche per molti altri tipi di applicazioni e processi di streaming. L'applicazione ETL in streaming può estrarre i dati dalla sorgente, oppure la sorgente può pubblicare i dati direttamente all'applicazione ETL in streaming. Quando un processo ETL in streaming viene completato, può passare i dati a destra verso una destinazione (potenzialmente un DW), oppure può inviare un risultato all'origine a sinistra.

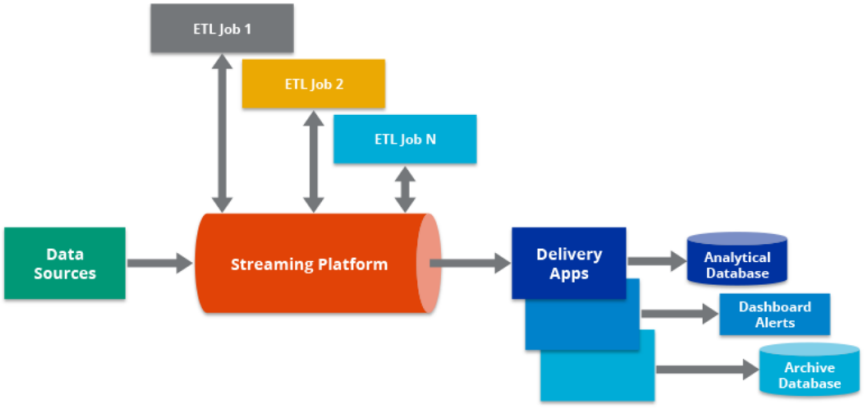

Figura 2.3. Processo Streaming ETL

2.2 Sistemi di raccomandazione

Le tecniche di raccomandazione hanno sempre svolto un ruolo cruciale nelle attività di marketing. I sistemi di raccomandazione sono disponibili da molto tempo, ma solo di recente hanno iniziato a essere identificati con le attività di e-commerce. Un esempio di raccomandazione fisica che i rivenditori adottano è l'uso della vetrina, l'esposizione di un manichino con il proprio marchio di abbigliamento fuori dai loro negozi. Tali raccomandazioni diventano una sfida in uno scenario di shopping online.
L'obiettivo di qualsiasi sistema di raccomandazione è aiutare gli utenti a trovare i prodotti o gli articoli che desiderano. Nell'e-commerce, il sistema presenta all'utente l'elenco degli articoli che più probabilmente può acquistare. In questi casi, il sistema deve disporre di tutte le informazioni sugli utenti (come i loro dati anagrafici, gli articoli acquistati in precedenza, i canali di acquisto, ecc.) e sui prodotti (come gli attributi dei prodotti, i prezzi, gli sconti disponibili, le descrizioni, ecc.).

Esistono 4 categorie di sistemi di raccomandazione:

- **Non personalizzati**

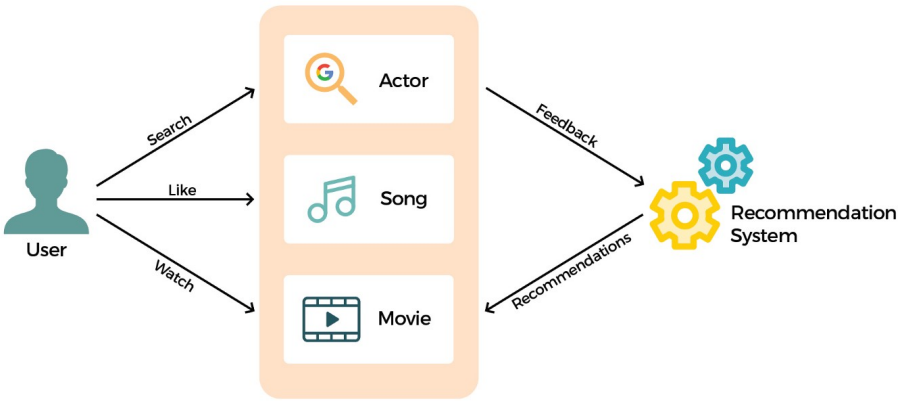

Figura 2.4. Sistema di raccomandazione

- **Basati sul contenuto**
- **Filtraggio collaborativo**
- **Sistemi ibridi**

2.2.1 Raccomandazione non personalizzata

Mentre un sistema di raccomandazione personalizzato suggerisce prodotti a un utente in base alla sua storia di acquisti precedenti, un sistema di raccomandazione non personalizzato mostra alle persone in modo generico quali siano i prodotti più diffusi nel breve periodo.
In questo modo non c'è differenziazione, ovvero vengono raccomandati a tutti gli utenti gli stessi oggetti, siccome non avviene una profilazione dell'utente basata su ciò che il singolo cliente seleziona dal catalogo online e salva. Da questa semplice operazione il sistema calcola per ogni oggetto selezionato la popolarità direzionando così in modo generico le preferenze degli utenti.
Esistono vari tipi di sistemi non personalizzati, a seconda dell'utilizzo da parte del cliente.

Product trending

Si basa sull'utilizzo di prodotti di tendenza nei siti di e-commerce, principalmente nella loro home page o in un angolo delle pagine di visualizzazione dei prodotti. Lo scopo principale della visualizzazione di questi prodotti di tendenza è quello di mantenere i visitatori impegnati sino ad acquisirli come clienti veri e propri, aggiornando costantemente i prodotti in tempo reale o quasi.

Product ranking

Il ranking dei prodotti, aiuta a decidere l'ordine in cui i prodotti devono essere visualizzati sulla pagina web di ogni singolo utente. Che si tratti di una home page, di una pagina di categoria o di una pagina di risultati di una ricerca, i prodotti presentati devono avere un ordine logico. Intuitivamente, questo ordine dovrebbe essere basato sulla probabilità che un prodotto venga cliccato o acquistato dai clienti.

Prediction Based Ranking

Con questa tecnica, le metriche vengono utilizzate per prevedere la classifica dei prodotti per il giorno successivo con l'aiuto dei dati storici. Il vantaggio principale di questa tecnica è che un modello addestrato può aiutare a classificare i prodotti in relazione alla variazione dei loro prezzi su base giornaliera.
Vengono utilizzati vari algoritmi di apprendimento automatico, come la regressione lineare, la regressione a foresta casuale, ecc. che possono essere utilizzati per addestrare modelli nel prevedere le vendite del giorno successivo utilizzando caratteristiche di input, come le vendite storiche in dollari, le visualizzazioni, le unità vendute, le scorte disponibili, il prezzo giornaliero, il giorno della settimana, il mese (per modellare la stagionalità), lo sconto disponibile, il rating, il numero di recensioni, il numero di clic, ecc.

2.2.2 Raccomandazione basata sul contenuto

Il filtraggio basato sui contenuti utilizza le specifiche degli articoli per raccomandare altri articoli simili a quelli che piacciono all'utente, in base alle sue azioni precedenti o a un feedback esplicito.
Questo modello utilizza spesso il dot product per calcolare una metrica di somiglianza tra due oggetti in base alle preferenze espresse dagli utenti. Si consideri il caso in cui l'embedding dell'utente x e l'embedding dell'applicazione y siano entrambi vettori binari. Poiché il prodotto è definito come:

$$(x \cdot y) = \sum_{i=1}^{d} x_i y_i \tag{2.1}$$

una caratteristica che compare in entrambi i vettori x e y contribuisce con un 1 alla somma. In altre parole, il prodotto scalare, o dot product, è il numero di caratteristiche attive in entrambi i vettori contemporaneamente. Un prodotto di punti elevato indica quindi un maggior numero di caratteristiche comuni e quindi una maggiore somiglianza.
Ricapitolando, l'obiettivo del filtraggio basato sui contenuti è quello di classificare i prodotti con parole chiave specifiche, imparare ciò che piace al cliente, cercare quei prodotti a lui collegati nel database e quindi raccomandare altri prodotti similari invogliandolo ad acquistare.
Questo tipo di sistema di raccomandazione dipende in larga misura dagli input forniti dagli utenti; alcuni esempi comuni sono Google, Wikipedia, ecc. Ad esempio, quando un utente cerca un gruppo di parole chiave, Google visualizza tutti gli articoli composti da quelle parole chiave e le utilizza per inviare all'utente informazioni in merito.

2.2.3 Raccomandazione con filtraggio collaborativo

I metodi per i sistemi di raccomandazione che si basano principalmente sulle interazioni precedenti tra gli utenti e gli oggetti target sono noti come metodi di filtraggio collaborativo.
Di conseguenza, tutti i dati passati sulle interazioni degli utenti con gli oggetti di destinazione saranno inseriti in un sistema di filtraggio collaborativo. Queste informazioni sono solitamente registrate come una matrice, con le righe che rappresentano gli utenti e le colonne che rappresentano gli oggetti.
La premessa di base di questi sistemi è che i dati precedentemente raccolti dagli utenti siano sufficienti per generare una previsione. In altre parole, non si ha bisogno di altro che di dati storici, di altri input dell'utente, di dati di tendenza attuali e così via.
Inoltre, i metodi di filtraggio collaborativo si dividono in due sottogruppi: metodi basati sulla memoria e metodi basati sul modello.

Basati sulla memoria

I metodi basati sulla memoria sono i più semplici perché non utilizzano alcun modello. Partono dal presupposto che le previsioni possono essere fatte solo sulla base della "memoria" dei dati passati e in genere utilizzano un semplice approccio di misurazione della distanza, come ad esempio il nearest neighbor.

Basati sul modello

Gli approcci basati sul modello, invece, di solito presuppongono e fanno riferimento a una qualche forma di modello sottostante e cercando di garantire che le previsioni fatte si adattino correttamente al modello.

2.2.4 Raccomandazione con sistemi ibridi

Un sistema di raccomandazione ibrido è un tipo speciale di sistema di raccomandazione che può essere considerato come la combinazione del metodo di filtraggio collaborativo e del contenuto. Combinare insieme il filtraggio collaborativo e quello basato sul contenuto può aiutare a superare le carenze che si riscontrano nell'uso separato e, in alcuni casi, può essere più efficace. Gli approcci ibridi ai sistemi di raccomandazione possono essere implementati in vari modi, ad esempio utilizzando metodi basati sul contenuto e metodi collaborativi per generare previsioni separatamente e combinando poi le previsioni o semplicemente aggiungendo le funzionalità dei metodi collaborativi a un approccio basato sul contenuto (e viceversa).

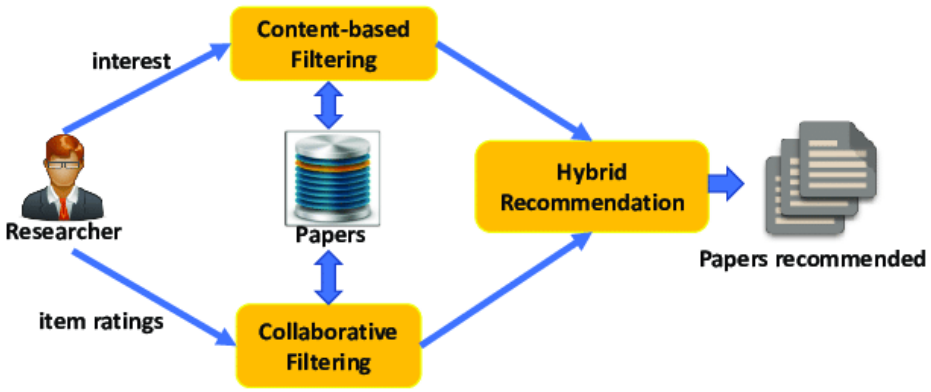

Figura 2.5. Sistema di raccomandazione ibrido

2.2.5 Use Cases

Nel descrivere i differenti sistemi di raccomandazione di alcune delle più grosse compagnie tech al mondo, tra cui Netflix, TikTok, Youtube (Google), non si entrerà

nello specifico per quanto riguarda i dettagli tecnici dei diversi algoritmi utilizzati, preferendo, per l'obiettivo della Tesi, definire nel dettaglio il loro funzionamento.

Netflix

Netflix[10] è un servizio di streaming in abbonamento che consente di guardare serie TV e film senza pubblicità su un dispositivo connesso a Internet.
Come spiegato in un loro stesso blog[9], una delle cose più utili per gli abbonati è incorporare le raccomandazioni per personalizzarle il più possibile. La personalizzazione inizia dalla homepage, che consiste in gruppi di video disposti in file orizzontali. Ogni riga ha un titolo che trasmette la connessione significativa prevista tra i video di quel gruppo. La maggior parte della personalizzazione si basa sul modo in cui vengono selezionate le righe, sulla scelta degli elementi da includere in esse e in quale ordine collocare tali elementi.
Un esempio è la riga della Top 10, cioè la migliore ipotesi sui dieci titoli che probabilmente possono piacere di più all'utente. Nello specifico l'utente viene inteso come l'intera famiglia che accede al servizio, per questo nella Top 10 è possibile trovare cose che possono piacere a tutti i componenti del nucleo. In molte parti del sistema, infatti, non viene ottimizzata solo l'accuratezza, ma anche la diversità.
L'obiettivo dei sistemi di raccomandazione è quello di presentare a una persona una serie di articoli interessanti tra cui scegliere. Questo obiettivo viene solitamente raggiunto selezionando alcuni elementi e catalogandoli in base all'ordine di gradimento (o utilità) atteso. Poiché il modo più comune di presentare gli articoli consigliati è in una forma di elenco, come le varie righe di Netflix, c'è bisogno di un modello di classificazione appropriato che possa utilizzare un'ampia varietà di informazioni per ottenere una classificazione ottimale degli articoli finalizzata ai gusti di ciascuno dei membri dell'account.
Se si cerca una funzione di classificazione che ottimizzi il consumo, una base ovvia

è la popolarità degli articoli. Il motivo è chiaro: in media, un utente è più propenso a guardare ciò che la maggior parte degli altri guarda. Tuttavia, la popolarità è l'opposto della personalizzazione: produrrà lo stesso ordine di articoli per ogni membro. Pertanto, l'obiettivo diventa quello di trovare una funzione di classificazione personalizzata che sia migliore della popolarità degli articoli, in modo da poter soddisfare meglio i membri con gusti diversi.

Il modo migliore per consigliare ad un utente un nuovo prodotto può essere l'uso delle valutazioni previste come funzione di classificazione che può portare a raccomandare articoli troppo di nicchia o poco conosciuti, e può escludere articoli che il singolo componente della famiglia vorrebbe guardare anche se non li valuta molto. Per compensare questo problema, piuttosto che usare la popolarità o la valutazione singolarmente, si può pensare di produrre classifiche che bilancino entrambi questi aspetti. Per farlo, bisogna costruire un modello di previsione delle classifiche utilizzando queste due caratteristiche.

Ci sono molti modi per costruire una funzione di classifica, che vanno dai semplici metodi di punteggio alle preferenze a coppie, fino all'ottimizzazione dell'intera classifica. L'approccio di punteggio risulta più elementare scegliendo che la funzione di ranking sia una combinazione lineare di popolarità e valutazione prevista. Si ottiene così un'equazione della forma F_rank(u,v) = w1 p(v) + w2 r(u,v) + b, dove u=utente, v=oggetto video, p=popolarità e r=valutazione prevista. Una volta che si ottiene una funzione di questo tipo, si può inserire una serie di video attraverso la funzione e ordinarli in ordine decrescente in base al punteggio.

Questo insieme di problemi di apprendimento automatico è noto come "Learning to rank" ed è centrale per prefigurare scenari applicativi per i motori di ricerca o il targeting degli annunci.

TikTok

Su TikTok[12], il feed For You riflette le preferenze uniche di ogni utente. Il sistema consiglia i contenuti classificando i video in base a una combinazione di fattori, a partire dagli interessi espressi dall'utente come nuovo utente e regolando anche gli argomenti a cui l'utente indica di non essere interessato, per formare il feed For You personalizzato.
Le raccomandazioni si basano su una serie di fattori, tra cui:

- **Interazioni dell'utente**, come i video a cui viene messo "mi piace" o condivisi, gli account che un utente inizia a seguire, i commenti che vengono pubblicati e i contenuti creati.
- **Informazioni sui video**, che possono includere dettagli come didascalie, suoni e hashtag.
- **Impostazioni del dispositivo e dell'account**, come la lingua preferita, l'impostazione del Paese e il tipo di dispositivo.

Questi fattori sono inclusi per garantire che il sistema sia ottimizzato per le prestazioni, ma hanno un peso minore nel sistema di raccomandazione rispetto ad altri dati misurati, poiché gli utenti non li esprimono attivamente come preferenze.
Tutti questi fattori vengono elaborati dal sistema di raccomandazione e ponderati in base al loro valore per uno specifico utente. Un forte indicatore di interesse, come il fatto che un utente finisca di guardare interamente un video più lungo, riceverà un peso maggiore rispetto, per esempio, ad un indicatore debole, come il fatto che lo spettatore e il creatore del video si trovino entrambi nello stesso Paese.
I video vengono quindi classificati per determinare la probabilità che un utente sia interessato a un contenuto e poi consegnati a ogni singolo feed di For You.
Generalmente un video riceve più visualizzazioni se pubblicato da un account che ha molti follower, in virtù del fatto che l'account ha costruito una base di follower

più ampia, in questo caso però né il numero di follower né il fatto che l'account abbia avuto in precedenza video ad alto rendimento sono fattori diretti nel sistema di raccomandazione in quanto questo elemento potrebbe costituire una forma di vizio per la capacità selettiva.

YouTube

Il sistema di raccomandazioni di YouTube[13] si basa sul semplice principio di aiutare le persone a trovare i video che desiderano guardare e che offrono loro un valore aggiunto. Le raccomandazioni sono presenti in due punti principali: la homepage e il pannello "Avanti". La homepage è ciò che si vede la prima volta che si apre YouTube e mostra un mix di raccomandazioni personalizzate, abbonamenti e le ultime notizie e informazioni. Il pannello Up Next appare quando si sta guardando un video e suggerisce contenuti aggiuntivi basati su ciò che si sta guardando, oltre ad altri video che si ritiene possano interessare l'utente.
Il sistema di raccomandazioni si evolve costantemente, imparando ogni giorno da oltre 80 miliardi di informazioni che vengono chiamate segnali, ma inizialmente non era così.
Quando il primo sistema di raccomandazione di YouTube venne lanciato, nel 2008, era semplicemente basato sulla popolarità dei video, ovvero i video erano classificati in base a quanti click ricevevano e questo indicava la popolarità.
Nel 2011 a questo sistema venne aggiunto il watchtime dei video. In pratica si notò che il semplice click non bastava più, ma era importante anche sapere per quanto tempo un utente aveva visto quel video. Questo forniva al sistema dei segnali personalizzati su ciò che l'utente desiderava guardare. Quindi, se un utente appassionato di tennis ha guardato 20 minuti di clip di highlights di Wimbledon e solo pochi secondi di video di analisi della partita, si può supporre che abbia trovato più utile guardare gli highlights.

L'inserimento di tale specifica portò ad una caduta del 20% nella visualizzazioni dei video, ma alla consapevolezza che avrebbe portato maggiore qualità e valore all'esperienza dell'utente.
A seguito di ciò si iniziò a misurare il "tempo di visione apprezzato" mediante l'utilizzo di risposte a sondaggi. Per essere certi che gli spettatori fossero soddisfatti dei contenuti che guardavano, venne misurato il "tempo di visione apprezzato", ovvero il tempo trascorso a guardare un video che l'utente considerava prezioso. Questo tempo viene valutato attraverso sondaggi che chiedono agli utenti di dare un voto da una a cinque stelle al video che hanno guardato, fornendo così un parametro per determinare quanto il contenuto sia stato soddisfacente. Se l'utente assegna a un video una o due stelle, gli si chiede perché ha dato un voto così basso. Allo stesso modo, se il video viene valutato da quattro a cinque stelle, viene chiesto il perché è stato fonte di ispirazione o significativo. Solo i video valutati con quattro o cinque stelle vengono conteggiati come tempo di visione apprezzato.
Naturalmente, non tutti compilano un sondaggio per ogni video che guardano. Sulla base delle risposte ricevute, viene addestrato un modello di apprendimento automatico, tramite machine learning, per prevedere le potenziali risposte al sondaggio per tutti. Per testare l'accuratezza di queste previsioni, vengono trattenute di proposito alcune risposte al sondaggio dall'addestramento. In questo modo si può sempre controllare quanto il nostro sistema sia in linea con le risposte reali.
Come ultima aggiunta a questo sistema di raccomandazione, viene dato un peso più importante a "mi piace", condivisioni e "non mi piace". Questo perchè, in media, le persone hanno maggiori probabilità di essere soddisfatte dai video che condividono o che piacciono. Il sistema utilizza queste informazioni per cercare di prevedere la probabilità che l'utente condivida o apprezzi altri video. Se un video consigliato non piace, è il segnale di una analisi non completamente corretta.

Ecco perché le raccomandazioni svolgono un ruolo così importante nel mantenimento di una piattaforma responsabile. Collegano gli spettatori a informazioni di alta qualità e riducono al minimo le possibilità di vedere contenuti problematici.

Capitolo 3

Ambiente di sviluppo

L'ambiente di sviluppo di questa piattaforma comprende diversi framework e linguaggi di programmazione.
Tutta l'applicazione web è gestita tramite i container di Docker in cui vengono eseguiti diversi componenti che mantengono il sito attivo e funzionante.
La parte di back-end del sito è sviluppata tramite Python usando il framework Django che permette uno sviluppo più semplice e veloce del sito rispetto alle comuni implementazioni.
Tutta la parte del db è gestita con il database non relazionale MongoDb orientato ai documenti, il quale viene utilizzato tramite la libreria di Python pymongo che permette di utilizzare una sintassi simile a quella usata nella shell di MongoDb per poter eseguire le varie operazioni.
Tutta la parte di frontend, invece, è sviluppata con HTML, CSS e JavaScript, che insieme permettono la creazione e la gestione dell'esperienza dell'utente.
In seguito verranno spiegate le diverse tecnologie precedentemente citate, con le ulteriori alternative a disposizione.

3.1 Tecnologie

3.1.1 Docker

Docker[4] è una piattaforma open-source per sviluppare ed eseguire applicazioni all'interno di container software utilizzando sistemi operativi Linux. Docker permette di far eseguire tutte le applicazioni su un unico sistema operativo. Utilizza le funzionalità del kernel e sfrutta l'isolamento delle risorse e i namespace separati per isolare ciò che l'applicazione può vedere del sistema operativo.
Queste applicazioni sono impacchettate in immagini di Docker: un pacchetto di software leggero, autonomo ed eseguibile che combina il codice sorgente dell'applicazione con tutte le librerie del sistema operativo e le dipendenze necessarie per eseguire il codice in qualsiasi ambiente. Quando una immagine diventa eseguibile essa diventa un container, che può essere avviato, fermato o spostato tramite l'API Docker o la CLI (command line interface).
Docker utilizza un'architettura client-server come si può notare in Figura 3.1. Il client parla con il daemon di Docker, che fa il lavoro pesante di costruire, eseguire e distribuire i containers. Il client e il daemon possono essere eseguiti sullo stesso sistema, oppure è possibile collegare un client Docker a un daemon remoto. Il client Docker e il daemon comunicano usando un'API REST, su socket UNIX o un'interfaccia di rete.

Docker Compose

É un tipo di client Docker, che permette di lavorare con applicazioni costituite da un insieme di containers. Con Compose, si utilizza un file YAML per configurare i servizi dell'applicazione. Poi, con un singolo comando, si creano e si avviano tutti i servizi dalla configurazione.

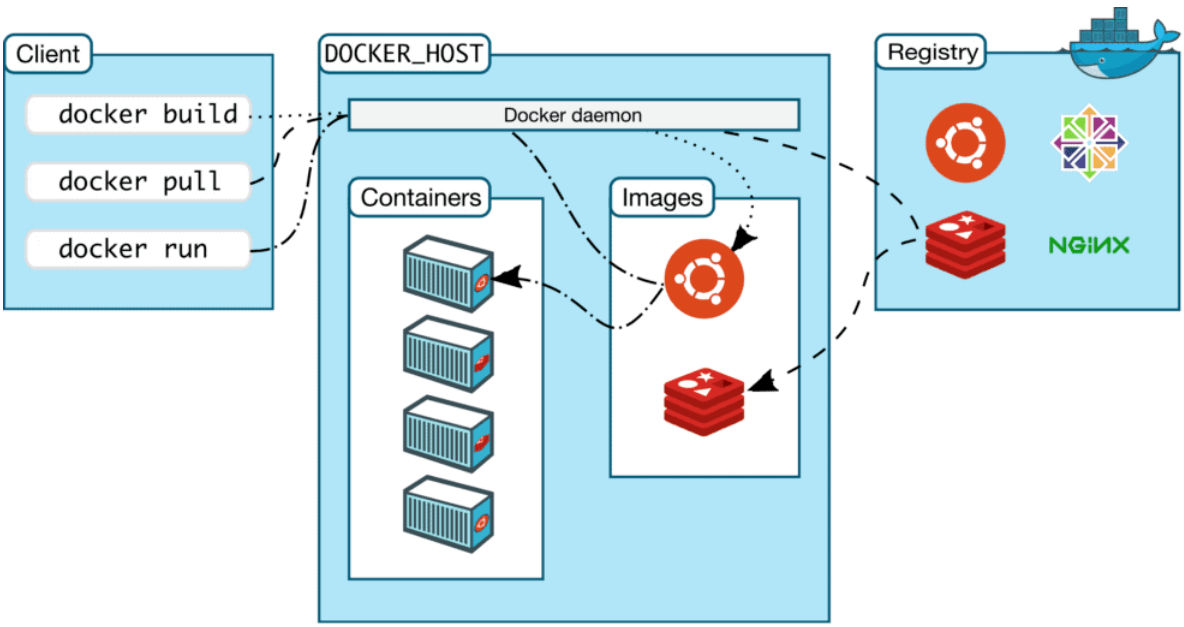

Figura 3.1. Architettura Docker

Docker nella piattaforma

Docker Compose viene utilizzato per creare tutti i container che permettono il funzionamento della piattaforma. Praticamente vengono creati 7 containers diversi, ognuno dei quali con un compito specifico ed isolato dagli altri:

1. **techmongodb**: Sono tre container destinati alla gestione del db non relazionale tramite mongoDb, ma solo uno di questi viene usato (techmongodb1) siccome gli altri fungono da replica sets in caso di guasti. É il container in cui sono memorizzati tutti i dati e le specifiche degli item presenti nel sito.

2. **techredis**: Redis è un datastore in-memory open-source usato come db, cache e anche come message broker. Redis in questo caso viene usato con docker Compose per lo sviluppo locale come cache dell'applicazione web.

3. **celerytech**: Celery permette di eseguire task o lavori in maniera asincrona. Per quanto riguarda questa piattaforma viene usato per eseguire task lunghe

che, se eseguite in maniera sincrona, bloccherebbero l'utilizzo del sito. Inoltre celery viene spesso utilizzato per schedulare delle task nel tempo avendo così la possibilità di automatizzare alcuni lavori che sarebbero da svolgere quotidianamente, in modo tale da avere un sito sempre aggiornato e funzionante. Ne verranno spiegati alcuni esempi nei capitoli successivi.

4. **tech**: É il container più importante siccome garantisce l'esecuzione del sito in se, con la parte pubblica e privata. Gestisce tutte le richieste https effettuate e lo scambio di dati tra front-end e back-end.

5. **techmongosetup**: viene utilizzato solo per inizializzare automaticamente il replica set di MongoDB la prima volta. Mentre le altre volte controlla solamente che sia inizializzato uscendo senza far nulla.

3.1.2 Django

Django[3] è un framework web Python di alto livello che facilita con uno sviluppo rapido un design pulito e pragmatico. Si prende cura di gran parte dello sviluppo web, in modo da potersi concentrare sulla scrittura dell'applicazione senza bisogno di partire da zero. È gratuito, open source e facilmente scalabile. Inoltre si occupa automaticamente dell'autenticazione degli utenti, dell'amministrazione dei contenuti e delle mappe del sito.

Architettura

Per mantenere il codice semplice e pulito Django propone un'architettura con tre elementi base:

- **Model**: è responsabile della manutenzione dei dati dell'applicazione e funge da mediatore tra l'interfaccia del sito web e il database.

- **Template**: è un file che descrive come il contenuto di uno o più model deve essere inserito.

- **View**: contiene la logica della nostra applicazione, gestisce l'interazione con l'utente ed interagisce sia con i model che con i template.

La peculiarità di questa architettura è che le tre componenti sono separate tra loro a livello di organizzazione di codice, ma sono molto collegate. Ad esempio, in una interazione classica con una piattaforma, sviluppata mediante django, avviene prima una risoluzione dell'url tramite l'URLConf di django che in base all'url digitato richiama la view corrispondente insieme a dati aggiuntivi, se necessari. Successivamente dentro la view, vengono raccolti dei dati nel db, i quali vengono mandati al template corrispondente in modo tale da poter generare un front-end con dati dinamici.

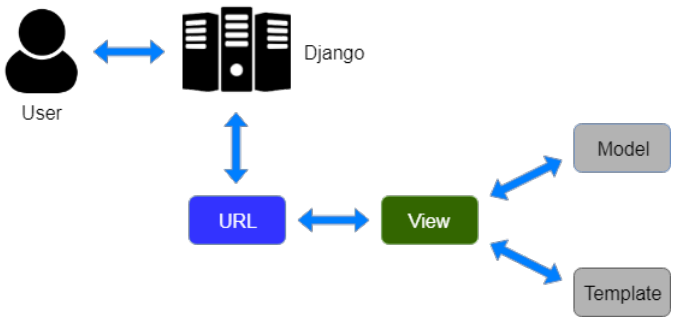

Figura 3.2. Architettura Django

Alternative

Esistono diverse alternative al molto utilizzato framework di python Django, ovviamente sono molto meno diffuse siccome viene usato da tante compagnie nel mondo tech. In seguito alcune alternative molto popolari.

Pyramid

Pyramid è un framework full stack per Python, sviluppato nell'ambito del Pylons project. Open source e completamente gratuito, è un framework non opinionated: non impone prescrizioni, ma offre allo sviluppatore molta scelta su come gestire diversi aspetti come sessioni, autenticazione, autorizzazione e altro. Tra le funzionalità principali di Pyramid ci sono la generazione di URL, lo sviluppo di API, le estensioni di configurazione, i test e la documentazione dei dati. Estremamente versatile, si adatta tanto a progetti piccoli quanto ad applicazioni grandi e complesse.

TurboGears

Questo framework presenta alcune peculiarità che lo rendono particolarmente interessante: un sistema di dispatch delle richieste che impone una struttura ordinata all'applicazione, non richiedendo alcuna configurazione di regole di routing, un template engine totalmente XHTML-compliant, con identificazione degli errori, un potente sistema a Widget per la creazione di form che rende intuitivo creare interazioni con validazione dei dati, supporto integrato a SQLAlchemy e MongoDB ad ogni livello del framework, un Admin auto generato per gestire tutti i dati della vostra applicazione web, una struttura a componenti che consente di abilitare, disabilitare o sostituire qualsiasi comportamento del framework, la possibilità di essere usato sia come framework full-stack che come micro-framework.
La differenza più grande con Django è la scalabilità, in questo TurboGears è migliore.

Flask

È classificato come microframework perché non richiede strumenti o librerie particolari. Non ha un livello di astrazione del database, la validazione dei moduli o

altri componenti per i quali librerie di terze parti preesistenti forniscono funzioni comuni. Tuttavia, Flask supporta estensioni che possono aggiungere funzionalità alle applicazioni come se fossero implementate in Flask stesso. Esistono estensioni per i mappatori oggetti-relazionali, la validazione dei moduli, la gestione degli upload, varie tecnologie di autenticazione aperta e diversi strumenti comuni legati al framework.
Rispetto a Django è un framework molto più leggero, da utilizzare preferibilmente in progetti piccoli, anche single-page.

web2py

É un framework facile da usare che si concentra sulla sicurezza e sulla velocità di sviluppo. Copre tutte le funzionalità di Django, poiché offre molte caratteristiche già pronte, come un server web, un pannello di amministrazione, un database, una griglia di widget o un wiki, ed entrambi possono essere utilizzati per svolgere gli stessi compiti. L'unica differenza è che Web2Py potrebbe non essere intuitivamente facile e ancora nell'uso poco sviluppato, poiché è molto recente e ha una comunità più piccola rispetto a Django.

3.1.3 MongoDB

MongoDB[8] è un sistema di gestione di db non relazionale orientato ai documenti. Essendo classificato come db NoSQL si discosta dalla normale concezione di db, in cui i dati sono salvati in tabelle, ma utilizza invece documenti flessibili JSON-like (in particolare MongoDB usa BSON), nei quali i campi possono variare da un documento all'altro pur facendo parte della stessa collezione, cosa non possibile con i normali db.
Ovviamente mantiene le peculiarità più importanti di un db, ovvero eseguire query sui dati presenti nei documenti, eseguire update, delete ed insert di dati anche su

più documenti contemporaneamente.
La caratteristica principale di mongoDB è la possibilità di eseguire operazioni di aggregazione sui dati. Tramite queste operazioni è possibile elaborare più documenti alla volta ed eseguire diverse operazioni su di essi, come può essere una group, per poi utilizzare i risultati calcolati.

Pymongo

Pymongo[11] è una distribuzione Python contenente strumenti per lavorare con MongoDB ed è il modo consigliato per lavorare con MongoDB da Python.
Permette di utilizzare la sintassi della shell di mongoDb in python, così da poter eseguire le operazioni CRUD[1] o di aggregazione nel codice.

Alternative

In questa sezione verranno discusse alcune alternative, sia relazionali che non relazionali, a MongoDB.

Cassandra

Apache Cassandra è un database NoSQL gratuito e open source. Implementa un'architettura di archiviazione a colonne e può gestire grandi volumi di dati distribuiti su più nodi Apache Cassandra.
In termini di scalabilità, siccome MongoDB ha un singolo nodo Master, può eseguire solo un'operazione di scrittura alla volta e quindi può essere considerato limitato in termini di scalabilità della scrittura. D'altra parte, Apache Cassandra, avendo più nodi master, può coordinare più operazioni di scrittura allo stesso tempo.
La differenza sostanziale con MongoDB è il framework delle aggregazioni, di fatto

[1]Le 4 operazioni basilari che si possono eseguire sulle tabelle: create, read, update, delete

Cassandra non ne possiede uno incorporato, come ha MongoDB, e quindi deve appoggiarsi a tool esterni.

DynamoDB

DynamoDB è un database NoSQL proprietario di Amazon che supporta dati chiave-valore e documenti tramite gli Amazon Web Services. Questa soluzione di AWS fornisce una piattaforma di database scalabile, altamente disponibile e sicura per qualsiasi applicazione.
Supporta l'auto sharding e il bilanciamento del carico ed è ideale per le applicazioni che memorizzano una grande quantità di dati con requisiti di latenza severi, ma è obbligatorio utilizzarlo sulla piattaforma AWS poiché è un loro prodotto esclusivo.

MySQL

MySQL è un relational database management system (RDBMS) composto da un client a riga di comando e un server. Entrambi sono multi piattaforma e sono disponibili ufficialmente su praticamente tutte le distribuzioni conosciute di UNIX e Linux.
Essendo un database relazionale la struttura dei dati è a tabelle e non più a documenti, questo porta ad avere un sistema più rigido rispetto alle collezioni.
Le peculiarità principali sono la possibilità di avere a disposizione più di 50 milioni di righe per salvare i dati e l'insieme di tool di gestione e di analisi per le tabelle.
Un'ottima scelta per i dati strutturati con la priorità di un'elevata sicurezza delle informazioni.

3.1.4 HTML, CSS

HTML è un linguaggio di programmazione volto allo sviluppo web, è principalmente usato per il disaccoppiamento della struttura logica di una pagina web e la

sua rappresentazione, gestita tramite gli stili CSS.
Il CSS[2] è un linguaggio usato per definire la formattazione di documenti HTML, XHTML e XML, ad esempio i siti web e relative pagine web.
L'introduzione del CSS si è resa necessaria per separare i contenuti delle pagine HTML dalla loro formattazione o layout e permettere una programmazione più chiara e facile da utilizzare, sia per gli autori delle pagine stesse sia per gli utenti, garantendo contemporaneamente anche il riutilizzo di codice ed una sua più facile manutenzione.

Bootstrap

HTML permette di utilizzare diversi framework, uno dei più usati è Bootstrap ovvero un framework di sviluppo front-end gratuito e open source per la creazione di siti. Il framework Bootstrap è costruito su HTML, CSS e JavaScript per facilitare lo sviluppo di siti e applicazioni responsive.
Il design responsive rende possibile per una pagina web di rilevare le dimensioni e l'orientamento dello schermo del visitatore e di adattare automaticamente la visualizzazione di essa.
Bootstrap include componenti dell'interfaccia utente, layout e strumenti JavaScript insieme al framework per l'implementazione.

3.1.5 JavaScript

JavaScript[5] (JS) è un linguaggio di programmazione leggero, interpretato o compilato just-in-time con funzioni di prima classe. Sebbene sia noto soprattutto come linguaggio di scripting per le pagine Web, viene utilizzato anche in molti ambienti non-browser, come Node.js, Apache CouchDB e Adobe Acrobat. JavaScript è un linguaggio dinamico, multi-paradigma, single-thread, basato su prototipi, che

supporta stili orientati agli oggetti, imperativi e dichiarativi.
Infatti allo sviluppo statico del front-end fatto in HTML viene affiancato lo sviluppo su JavaScript che permette di rendere la pagina più dinamica con la possibilità di gestire l'interazione dell'utente con il sito, cosa non possibile solamente con HTML.

Node.js

Node.js è un runtime system orientato all'esecuzione di codice JavaScript. Consente di utilizzare JavaScript anche per scrivere il codice da eseguire lato server, ad esempio per la produzione del contenuto delle pagine web dinamiche prima che la pagina venga inviata al browser dell'utente, cosa che prima non era possibile siccome si potevano incorporare gli script JavaScript solo all'interno degli HTML . Inoltre è basato sulla gestione di eventi, cosa che permette di gestire I/O asincroni ottimizzando così la velocità dell'applicazione web.

3.1.6 Nginx

Nginx è un server web HTTP che può essere utilizzato anche come reverse proxy, server di posta, bilanciamento del carico e server TCP/UDP generico. Nginx utilizza un'architettura che consente al server di reagire ai segnali hardware generati dalle operazioni di input e output. Questo lo rende estremamente efficiente nella gestione di grandi quantità di richieste concorrenti.

3.1.7 Celery

Celery[1] è un sistema distribuito semplice, flessibile e affidabile per elaborare grandi quantità di messaggi, fornendo al contempo alle operazioni gli strumenti

necessari per la manutenzione di tale sistema. È una coda di task che si concentra sull'elaborazione in tempo reale e supporta anche la programmazione dei task.
Il framework Django supporta Celery, infatti viene utilizzato spesso per la gestione dei task asincroni, ovvero dei task che sarebbero troppo lunghi da svolgere in maniera sincrona e che quindi bloccherebbero l'utilizzo del sito web.

Capitolo 4

Database

Il db viene gestito tramite MongoDB, la sua struttura è di tipo non relazionale basata sui documenti. Si parte come sono organizzate le collezioni specificando poi come i differenti documenti vengono utilizzati nella sezione pubblica e privata. I dati presenti dal database hanno una duplice provenienza, inizialmente i dati venivano estratti da un sito web di nome Kagoo e quindi tutte le specifiche e gli oggetti provenivano da quel sito. Successivamente il sito è stato chiuso ed è ciò che ha comportato l'utilizzo di un diverso sito da cui prelevare gli oggetti e le relative specifiche: *Icecat*. Questo cambiamento ha definito la presenza di tipologie diverse di specifiche degli oggetti individuate da uno stesso nome ma con differente path all'interno del db, queste stesse specifiche erano usate da due tipi di oggetti diversi: quelli di Icecat e quelli di Kagoo. Sono stati quindi sviluppati dei metodi che utilizzassero entrambi i tipi di specifiche e oggetti, facendo attenzione a non generare errori nel db.

4.1 Collezioni

Siccome non si definiscono delle classi in python, per gestire le informazioni si utilizzano direttamente i dizionari in JSON per comporre le collezioni. In questo modo i documenti all'interno delle collezioni possono avere una struttura differente gli uni dagli altri.
Nelle tabelle successive l'asterisco implica l'opzionalità del campo, ovvero quando non è presente per tutti i documenti della collezione.

4.1.1 Features

La collezione Features descrive le differenti specifiche presenti all'interno degli oggetti. Questa collezione può essere suddivisa in 3 diversi tipologie di documenti che rappresentano i tipi di features:

- *input*
- *transformed*
- *output*

Questa distinzione è nota grazie al campo *@type* presente in Tabella 4.1.
La collezione rappresenta il flusso del processo ETL all'interno della piattaforma per quanto riguarda le specifiche degli oggetti.
Le features di tipo **input** rappresentano l'inizio della fase di trasformazione dei dati del processo ETL, infatti costituiscono il valore iniziale della specifica con le successive trasformazioni da dover eseguire.
Come si può notare in Tabella 4.1, sono presenti: il *nome* della specifica, il *path*, un vettore *transform* rappresentato in Tabella 4.2 e come campo opzionale lo *score* (rappresentato come boolean).
Il campo *transform* definisce il modo in cui la specifica di un oggetto deve essere

trasformata, questa inizialmente presenta un valore di testo ed è quindi necessario individuare il corretto metodo di trasformazione del dato di partenza. La piattaforma permette di attuare direttamente questi tipi di trasformazione come successivamente spiegato nel Capitolo 5, Sezione 5.1.1.
Mentre il campo *score* definisce se una specifica può essere utilizzata per il calcolo dello score, anche questo sarà spiegato successivamente nel Capitolo 5, Sezione 5.1.2.
Le features di tipo **transformed** rappresentano un punto di collegamento tra le features di input e quelle di output, in particolare, come si può notare in tabella 4.3, presentano un campo *input* che indica la feature di input a cui fa riferimento e un campo *score* come in quella appena descritta.
Le features di tipo **output**, rappresentano come le specifiche vengono viste all'interno degli oggetti nella piattaforma, in pratica rappresentano la fase finale del processo di trasformazione del processo ETL.
Nel documento sono presenti diversi campi come si può notare in Tabella 4.4, alcuni uguali alle features spiegate in precedenza tra cui lo schema, il path e il nome e altri che invece definiscono alcune proprietà che la specifica avrà all'interno della piattaforma come: *typevar* tramite il quale è possibile capire il tipo di variabile (se testuale, numerica o booleana), *visible* che indica se la specifica è visibile tra quelle dell'oggetto sulla piattaforma, *mapping* che permette di capire la feature di tipo transformed a cui fa riferimento e di conseguenza quella di input, *score* indica se la specifica può essere usata per calcolare lo score, *filter* indica se la feature viene usata nella pagina di ricerca per filtrare gli oggetti. Il campo *output* viene utilizzato solo sulle features che vengono duplicate e trasformate per il calcolo dello score. Siccome le specifiche presenti sono tutte di tipo testuale, sono state eseguite delle trasformazioni, che verranno spiegate in seguito (cfr. Capitolo 5),

per rendere alcune specifiche numeriche o booleane. Per fare ciò le specifiche vanno duplicate e di conseguenza vengono create delle copie delle features di input, transformed e output all'interno della collezione. Essendo i duplicati collegati alla specifica testuale iniziale, nella collezione e nello specifico nelle features di tipo output duplicate viene inserito il campo output che rappresenta l'id della features output a cui fa riferimento, in modo da tracciare da quale specifica viene eseguita la trasformazione.

Campo	**Descrizione**	**Tipo**
name	Nome della feature	String
@type	Tipo di feature	String
schema	Schema a cui appartiene la feature	String
path	Path della feature	String
transform	Array con le trasformazioni da eseguire	Array
score*	Indica se usata per gli score	Boolean

Tabella 4.1. Descrizione documento Features input

Campo	**Descrizione**	**Tipo**
f	Tipo di trasformazione da applicare	String
params	Parametri da modificare in base al tipo di trasformazione	Object

Tabella 4.2. Descrizione array Features input transform

4.1.2 Items

La collezione Items rappresenta gli oggetti presenti nella piattaforma nella loro interezza, ovvero con tutte le specifiche e le informazioni necessarie.
Tra i campi che compongono i documenti si ha: il *title* che indica il nome dell'oggetto e che viene utilizzato nelle varie pagine della piattaforma, *price* indica il prezzo dell'oggetto (tramite un Double), *slug* è un identificativo univoco composto dal

Campo	Descrizione	Tipo
name	Nome della feature	String
@type	Tipo di feature	String
schema	Schema a cui appartiene la feature	String
path	Path della feature	String
input	Id della features di input a cui fa riferimento	ObjectId
score*	Indica se usata per gli score	Boolean

Tabella 4.3. Descrizione documento Features transformed

Campo	Descrizione	Tipo
name	Nome della feature	String
@type	Tipo di feature	String
schema	Schema a cui appartiene la feature	String
path	Path della feature	String
category	Tipo di categoria della feature	String
categoryN	Numero di categoria	Int32
No	Numero feature	Int32
typevar	Tipo della categoria (number,bool)	String
required	Indica se richiesta	Boolean
visible	Indica se visibile	Boolean
mapping	Array che indica da quale feature transformed arriva	Array
score*	Indica se usata per gli score	Boolean
filter*	Indica se usata per gli score	Boolean
output*	Id della features di output a cui fa riferimento	ObjectId

Tabella 4.4. Descrizione documento Features output

brand dell'oggetto più il suo nome senza spazi tramite il quale è possibile eseguire query sul db, siccome è univoco per ogni oggetto, *brand* indica il brand dell'oggetto, *published* indica se è già stato pubblicato sulla piattaforma, *specs* rappresenta la lista delle specifiche, ognuna delle quali definita nella collezione features spiegata precedentemente, con il loro valore di un tipo definito nel documento corrispondente delle features di output.

Il campo *score* è un oggetto che contiene il valore dello score calcolato. Per la

procedura di calcolo dello score si rimanda al Capitolo 5, Sezione 5.1.2.

Il campo *popularity* contiene il valore della popolarità dell'oggetto di quel mese calcolato tenendo conto dei valori dei mesi precedenti usando una funzione di decadimento esponenziale, descritta nel Capitolo 5, Sezione 5.1.3.

Campo	**Descrizione**	**Tipo**
title	Nome dell'oggetto	String
price	Prezzo dell'oggetto	Double
schema	Schema a cui appartiene l'oggetto	String
slug	Brand+nome dell'oggetto usato per gli url	String
releaseDate	Data di rilascio	Date
brand	Brand dell'oggetto	String
sku	Identificativo dell'oggetto	String
published	Indica se è pubblicato sul sito	Boolean
specs	Contiene la lista delle specifiche dell'oggetto con i relativi valori	Object
images	Stringhe rappresentanti il path delle immagini dell'oggetto	Array
pubinfo	Indica quando è stato inserito e modificato	Object
model	Modello dell'oggetto	String
aff_hist	Informazioni sul link di affiliazione	Object
score	Score dell'oggetto	Object
popularity	Popolarità dell'oggetto	Double

Tabella 4.5. Descrizione documento Items

4.1.3 Score

Nella collezione Scores vengono salvate tutte le specifiche che compongono i differenti tipi di score specificando i loro pesi.

Esistono due tipi di documenti in questa collezione definiti dal campo *@type*: il tipo score e il tipo feature.

Il tipo score come si può notare in Tabella 4.6 ha come campi: il *rank* dello score, lo *schema* di riferimento, il *name* e gli *aggregates* ovvero la lista degli score aggregati

per quello schema specifico. Questa lista, rappresentata in Tabella 4.7, presenta al suo interno l'insieme degli score aggregati per quello schema e all'interno di ognuno di essi è presente il peso che ha l'aggregato, campo *w*, e la lista delle specifiche di cui è composto, campo *f*, con i relativi pesi.
Il tipo feature rappresenta le features che vengono utilizzate per calcolare i vari scores. Come si può notare in Tabella 4.8 ha come campi: lo *schema* di riferimento, il *path* della specifica, il *label* ovvero il nome della feature, il *tag* cioè un array in cui sono segnati gli aggregati di cui la specifica fa parte e il *reverse* che indica se nello score vada calcolata all'inverso (1-val).

Campo	**Descrizione**	**Tipo**
@type	Tipo di score	String
schema	Schema di riferimento	String
rank	Rank dello score	Int32
name	Nome dello score	String
aggregates	Lista di aggregati per questo score	Object

Tabella 4.6. Descrizione documento Score

Campo	**Descrizione**	**Tipo**
w	Peso dell'aggregato	Int32
f	Insieme di features che compongono l'aggregato con i relativi pesi	Object

Tabella 4.7. Descrizione documento Score aggregate

4.1.4 Popularity

La collezione Popularity contiene i documenti utili a calcolare la popolarità degli oggetti. Ogni documento che compone la collezione è un oggetto di cui viene calcolata la popolarità mensile. I campi che compongono questi documenti sono:

Campo	Descrizione	Tipo
@type	Tipo di score	String
schema	Schema di riferimento	String
path	Path della feature	String
label	Nome della feature	String
tag	Indica a quali aggregates fa parte	Array
reverse	Indica se va calcolata all'opposto	Boolean

Tabella 4.8. Descrizione documento Score feature

name ovvero il nome dell'oggetto, lo *slug* cioè l'identificativo spiegato nella collezione Items, lo *schema* di riferimento dell'oggetto e l'array di *values*. Questo array ha una entry per ogni mese e ogni entry è un oggetto che contiene diverse informazioni, come descritto nella Tabella 4.10: il campo *date* che contiene la data all'inizio del mese di cui si vuole calcolare la popolarità, il *value* cioè il numero di volte in cui l'oggetto è stato visualizzato (sia in pagina singola sia in comparazione), *totalValue* rappresenta il numero tatale di click su tutti gli oggetti di un singolo schema e *normValue* ovvero il valore normalizzato tra 0 e 1 di *value* in base a *totalValue*. Questo valore viene successivamente utilizzato per calcolare la funzione di decadimento sui mesi precedenti.

Campo	Descrizione	Tipo
name	Nome dell'oggetto o della feature	String
slug	Identificatore dell'oggetto o della feature	String
@type	Tipologia (item/features)	String
schema	Schema di riferimento	String
values	Array di valori per mese	Array

Tabella 4.9. Descrizione documento Popularity

Campo	**Descrizione**	**Tipo**
date	Data in cui inizia il calcolo	Date
value	Numero di click	Int32
totalValue	Numero totale di click per schema	Int32
normValue	Valore di totalValue normalizzato a 1	Double

Tabella 4.10. Descrizione documento Popularity values

Capitolo 5

Implementazioni

La piattaforma su cui si è svolto il lavoro è divisa in due parti principali: una parte pubblica e una parte privata.
La parte pubblica rappresenta tutto ciò che un utilizzatore può fare e vedere, quindi le pagine di comparazioni tra prodotti, di ricerca dei prodotti e di ricerca per brand. Tutta questa parte di front-end è sviluppata con html, css e javascript con cui viene gestita l'interazione dell'utente con la pagina. La parte di back-end, invece, è sviluppata in python tramite l'ausilio della libreria pymongo per la gestione del db e di django per lo sviluppo dell'architettura del sito, di fatto le View di django presenti nel progetto richiamano i rispettivi template html.
La parte privata, invece, permette agli amministratori di gestire tutta l'architettura del sito: è possibile gestire le specifiche degli oggetti, duplicarle, creare features per la generazione di scores oppure inizializzare alcune parti del sito. Come per la parte precedente anche questa è sviluppata in html, css e javascript per il front-end mentre in python per il back-end esattamente come la parte pubblica.
Siccome le due parti sono divise l'una dall'altra, tutta la struttura peculiare a django è duplicata. Ogni sezione, privata e pubblica, contiene le diverse cartelle create da django in fase di inizializzazione di progetto. Le due parti hanno una cartella

differente per le Views (private e public in Figura 5.1), mentre per quanto riguarda i template html esiste una cartella sola con due sottocartelle una per la parte pubblica e una per la parte privata (_templates/private e _templates/public).

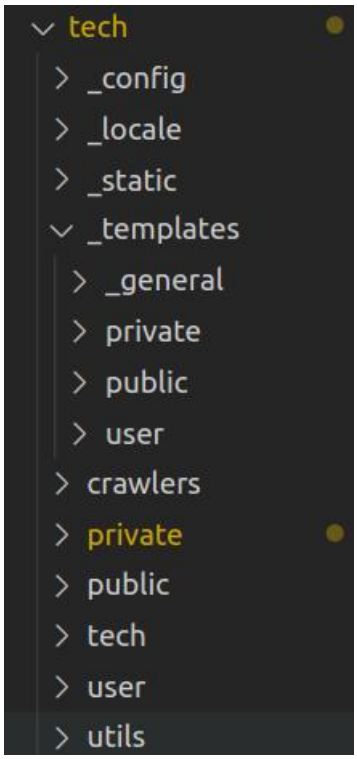

Figura 5.1. Organizzazione directory

5.1 Private

La sezione privata permette all'amministratore del sito di gestire il sito web con la possibilità di modificare diversi elementi del sito e degli oggetti utili alla comparazione.
Questa parte è divisa in 3 sotto categorie:

- **Items**

- **Site**
- **Configuration**

Items

In questa sotto categoria è possibile gestire gli oggetti presenti nel db, è quindi possibile aggiungere o modificare le features degli oggetti o gli oggetti stessi. É inoltre possibile inizializzare la sezione di ricerca implementata in questo progetto. In questo modo si creano le features su cui si possono filtrare gli oggetti nelle corrispettive pagine di ricerca.
Come si può notare in Figura 5.2 è possibile gestire la mappatura dei dati, calcolare gli scores o inizializzare le features prese da Icecat.

Il lavoro principale svolto in questo progetto è stato elaborato nella sezione di data model in cui è possibile duplicare le features da utilizzare per il calcolo degli scores e per la pagina di ricerca. Nella sezioni seguenti verranno spiegati i meccanismi e le implementazioni utilizzati per sviluppare queste due parti.

Site

In questa sotto categoria è possibile modificare gli url del sito, i brand dei prodotti presenti e modificare gli oggetti.
In manage Meta è possibile modificare gli url del sito nella sezione privata e pubblica. In manage Brand sono presenti i brand dei prodotti del sito ed è possibile inserire il link al sito originale di quel brand, ed è inoltre possibile modificare il nome. In manage Items è possibile andare ad inserire una review dell'oggetto oppure andarlo a modificare.

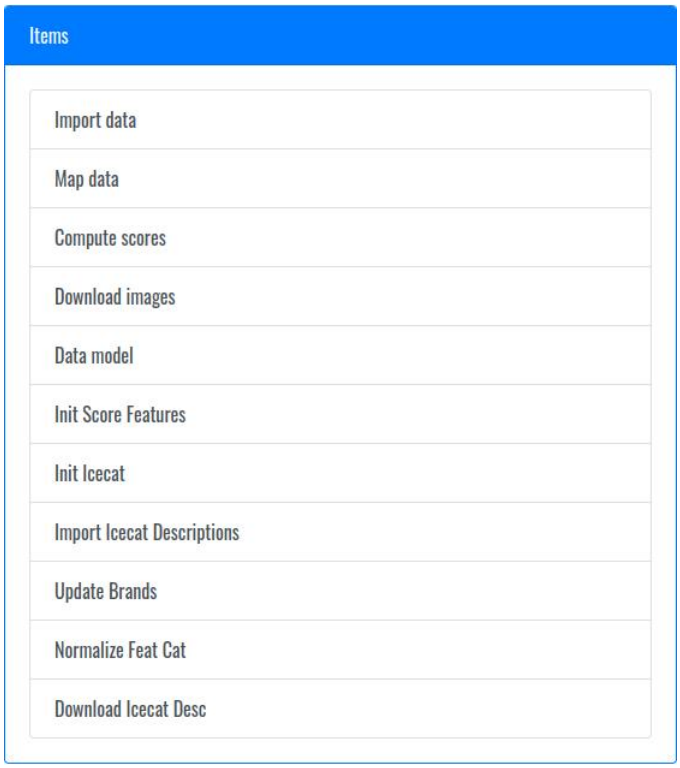

Figura 5.2. Menu Items sezione privata

Figura 5.3. Menu Site sezione privata

Configuration

In questa sotto categoria è possibile controllare alcune configurazioni del sito.
Si possono gestire le affiliazioni ai siti come Amazon o Ebay presenti nelle pagine

di spiegazione di un oggetto. Si possono verificare le traduzioni siccome il sito è tradotto in 6 differenti lingue. Si può inoltre definire lo stile del sito, tra cui logo, file css e le immagini. Infine è possibile gestire Icecat, ovvero il sito da cui vengono prelevati buona parte degli oggetti trattati poi nella piattaforma.

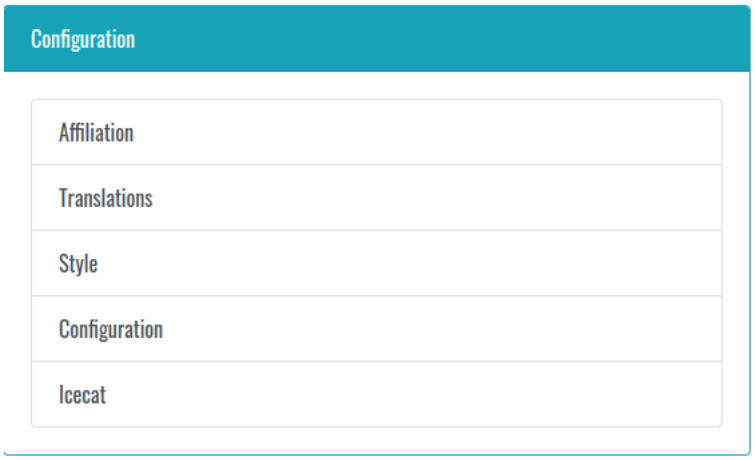

Figura 5.4. Menu Configuration sezione privata

5.1.1 Trattamento delle features

Il lavoro di trasformazione delle features è stato eseguito in modo tale da poter fornire dati al sistema di raccomandazione.
Le specifiche degli oggetti all'interno della collezione Items e le specifiche all'interno della collezione Features erano rappresentate in modo tale per cui non era possibile estrarre dei valori numerici per calcolare lo score dell'oggetto. É stato quindi necessario sviluppare un sistema di duplicazione e trasformazione delle specifiche degli oggetti.

Duplicazione

Tutte le specifiche che si volevano usare per il calcolo dello score sono state duplicate, generando una pagina all'interno della sezione private. Bisogna notare che è stato necessario gestire le due tipologie di specifiche provenienti dai due siti[1], questo perchè le specifiche di Icecat presentavano già la versione numerica duplicata mentre quelle di Kagoo presentavano solo quella testuale. Siccome le due tipologie di specifiche andavano separate e riconosciute è stato modificato il campo *path*, in particolare nelle specifiche numeriche è stato semplicemente aggiunto il suffisso '_val' al *path* di quelle testuali.

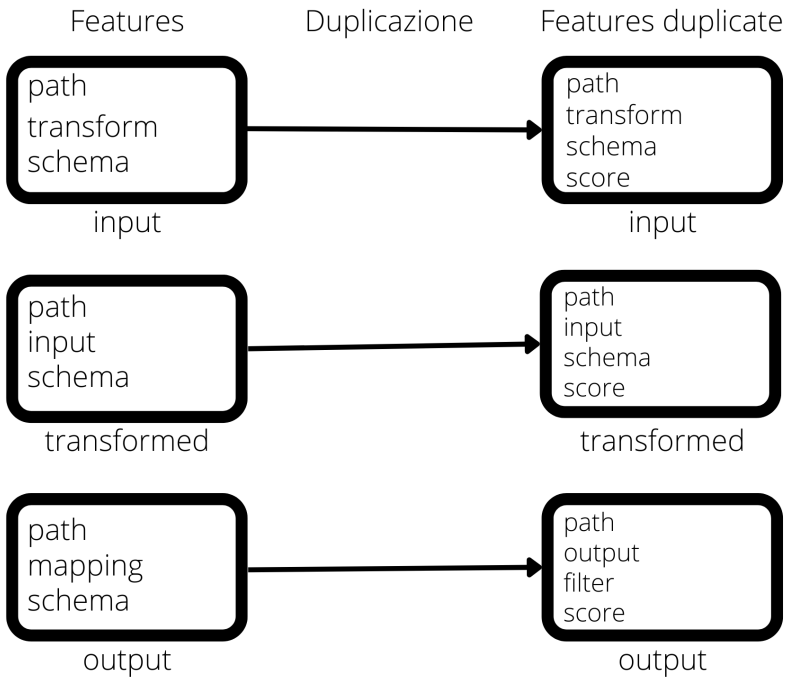

Figura 5.5. Duplicazione delle Features

[1]Si fa riferimento a quanto illustrato nell'introduzione del Capitolo 4.

Duplicazione specifiche di Icecat

Essendo le specifiche numeriche già duplicate e presenti nel db è stato necessario solamente modificare i documenti delle features.
Per quanto riguarda quelle di tipo output è stato necessario collegare le specifiche originali testuali a quelle numeriche. Ciò è stato realizzato tramite l'inserimento del campo *output* in quelle numeriche e l'inserimento dei campi *score* e *filter* rispettivamente a true e false in quelle testuali.
Per quanto riguarda le features di tipo input e transformed è stato aggiunto il campo *score* impostato a true in quelle numeriche.
Per le specifiche booleane, non essendo presente la versione duplicata con il suffisso '_val', è stato necessario duplicare i tre tipi di features in modo analogo a quanto spiegato precedentemente, settando in maniera appropriata i vari campi opzionali e impostando il *path* adeguatamente.

Duplicazione specifiche di Kagoo

Le specifiche di Kagoo sono presenti solamente in forma testuale all'interno degli oggetti, è stato quindi necessario sviluppare un sistema di duplicazione e successiva trasformazione delle specifiche.
Inizialmente dalla piattaforma si sceglie quale feature duplicare e si scelgono le diverse trasformazioni che bisogna eseguire sulla specifica testuale. Ad esempio, è possibile eseguire una replace per rimuovere una parte del testo o una number per trasformare la parte testuale in numero oppure concatenarle. Una volta operate queste scelte si possono salvare, tenendo presente però che la specifica risultante dovrà essere di tipo numerico o booleano.
A questo punto avviene la creazione delle nuove features duplicate, che si presenteranno in questo modo:

- **input**: presenta il *path* con il suffisso '_val' aggiunto e il campo *score* impostato a true, inoltre viene definito, nel vettore *transform*, la serie di trasformazioni che vanno effettuate sulla specifica, definite in precedenza.

- **transformed**: anch'essa presenta *path* con il suffisso '_val' aggiunto e il campo *score* impostato a true. Inoltre il campo *input* viene impostato con l'id della feature di input a cui fa riferimento, ovvero quella appena creata.

- **output**: come per le specifiche di Icecat, vengono aggiunti i campi opzionali necessari, cambiato il tipo di variabile della specifica, impostato *visible* a false ed impostato il campo *mapping* con il riferimento alla feature di tipo transformed appena creata.

Tutti gli altri campi dei documenti rimangono invariati.
In questo modo vengono create 3 nuove features che fanno riferimento alle specifiche testuali con le informazioni necessarie alla trasformazione, ma che non vengono viste sulla parte pubblica della piattaforma.

Questa fase di duplicazione è fondamentale allo scopo di calcolare lo score degli oggetti, che rappresenta una parte significativa del processo ETL utile alla creazione delle fondamenta del sistema di raccomandazione.

Trasformazione

Trasformazione specifiche di Icecat

Le specifiche di Icecat non necessitano di trasformazione, siccome presentano già i valori numerici e booleani all'interno degli oggetti.

Trasformazione specifiche di Kagoo

All'interno degli oggetti presi da Kagoo, come già indicato, le specifiche sono rappresentate da campi testuali e come spiegato precedentemente viene effettuata una duplicazione delle features in cui viene specificato come esse vadano trasformate.
La trasformazione in se avviene sempre all'interno della pagina in cui vengono duplicate le specifiche mediante una mappatura dei dati. In particolare, vengono estratte le specifiche utilizzate per lo score (campo *score* uguale a true) e si controlla per ogni oggetto dello schema selezionato se queste features ne fanno parte.
A questo punto vengono estratte le specifiche degli oggetti, applicate le trasformazioni specificate nella feature di tipo input e inserite come una nuova specifica nell'oggetto con il suffisso '_val'.
Il processo di trasformazione è basato su alcune funzioni implementate all'interno della piattaforma, generalizzate, in modo tale che possano funzionare per ogni tipo di oggetto.

5.1.2 Score

Questa sezione è dedicata al metodo di calcolo dello score degli oggetti e al motivo per cui viene elaborato in questo modo.
Il calcolo dello score parte da una specifica pagina predisposta nella sezione privata della piattaforma. In questa pagina è possibile definire gli score con i relativi aggregati in base allo schema selezionato. Esiste la possibilità di definire più aggregati in modo da creare più score per una stessa categoria.
É possibile scegliere il nome dello score, il nome dell'aggregato e selezionare le specifiche dell'oggetto che vanno a contribuire alla definizione dello score. Dopo aver definito i nomi verrà aggiunto nel db un documento di tipo score (come spiegato nella Sezione 4.1.3), in cui sarà presente oltre il nome dello score e dell'aggregato

anche il suo peso, senza ancora alcun riferimento alle features che saranno inserite successivamente.
Le specifiche disponibili sono tutte quelle di tipo testuale presenti nella collezione Featurs (di tipo output), con lo *score* uguale a true. Tali specifiche diventano disponibili per il calcolo dello score, solamente dopo aver effettuato le varie operazioni di duplicazione e trasformazione, qualora necessarie, siccome le specifiche di Icecat sono già a disposizione dello score.
A questo punto è possibile scegliere le specifiche da inserire nell'aggregato, definendone i relativi pesi che andranno ad influire sull'aggregato stesso.

Calcolo dello score

Il calcolo dello score viene eseguito solo dopo aver definito tutte le specifiche e i relativi pesi all'interno di un aggregato.
Il calcolo viene eseguito, per una categoria di oggetti, considerando per ogni specifica il valore massimo e minimo presenti nella collezione Items. In seguito per ogni oggetto si prende il valore della specifica e lo si divide per la differenza tra massimo e minimo, il risultato viene poi moltiplicato per il peso che ha la specifica all'interno dell'aggregato. Questo procedimento viene effettuato per tutte le feature avendo così tutti i valori degli score di ogni specifica. In seguito si prendono questi valori, si sommano, e si effettua una media sul numero di specifiche presenti nello score.
Di seguito una rappresentazione matematica della formula dello score:

$$score\_oggetto = \frac{1}{n} * \sum_{i=1}^{n} (\frac{val}{max_i - min_i} * peso_i) \tag{5.1}$$

dove n rappresenta il numero di specifiche, tra quelle selezionate, presenti nell'oggetto.

Score nel sistema di raccomandazione

Lo score presente negli oggetti è parte fondamentale del sistema di raccomandazione della piattaforma, insieme alla popolarità.
La raccomandazione della piattaforma può essere collocata nel genere non personalizzato, come spiegato nella Sezione 2.2.1, siccome non avviene uno studio del singolo utente, ma della situazione generica. In particolare si può considerare una raccomandazione non personalizzata basata sullo score, nel senso che i calcoli effettuati vengono eseguiti direttamente sugli oggetti da raccomandare e non sull'interazione che l'utente ha con essi.
Lo score viene appunto calcolato partendo dalle specifiche dei singoli oggetti e non valutando come l'utente interagisce con la piattaforma nella ricerca dei prodotti. Questo perchè nella piattaforma non esiste la possibilità di creare un profilo utente e quindi la raccomandazione deve rimanere la più generica possibile.

5.1.3 Popularity

In questa sezione verrà spiegato come la popolarità viene calcolata e come essa influisce nel sistema di raccomandazione.
La gestione della popolarità viene sviluppata nella piattaforma nel lato di backend, ovvero nel sistema della API del sito. Tutte le volte che viene visualizzato un oggetto, che sia su pagina singola tramite la pagina di ricerca o su pagina di comparazione viene incrementato il numero di click che l'oggetto stesso ha ricevuto in quel mese.
La popolarità viene gestita mensilmente, nel senso che all'inizio di ogni mese vengono calcolate le popolarità del mese precedente in base al numero di click ricevuti, ma tenendo in considerazione le popolarità dei mesi precedenti.

Calcolo della popolarità

Il calcolo della popolarità viene eseguito mensilmente, all'inizio di ogni mese vengono conteggiati il numero di click sull'oggetto e il numero totale di click per quella categoria di oggetti. In seguito viene normalizzato il valore di click di ogni oggetto, dividendolo per il numero totale di oggetti in modo tale da avere il risultato normalizzato tra 0 ed 1.
A questo punto avviene il calcolo vero e proprio della popolarità dell'oggetto: vengono presi i valori della popolarità dei mesi precedenti e viene calcolata una funzione di decadimento esponenziale, in cui più il mese è lontano da quello corrente meno influirà nella popolarità. Questi valori vengono in seguito sommati in modo tale da avere la popolarità generale dell'oggetto.
Di seguito una rappresentazione matematica della formula della popolarità:

$$popolarità\_oggetto = \sum_{i=0}^{t}(\frac{click_i}{click\_totali_i} * e^{-i}) \tag{5.2}$$

dove t rappresenta il numero di mesi su cui effettuare il calcolo.

Popolarità nel sistema di raccomandazione

La piattaforma non permette ad un utente di registrarsi e creare il proprio profilo, quindi è necessario che i vari oggetti vengano raccomandati in modo più generico. Come per lo score, la popolarità appartiene al gruppo della raccomandazione non personalizzata, ovvero non basata sulle scelte del singolo utente, ma più generica. La popolarità, nel modo in cui viene intesa e calcolata nella piattaforma può essere ricondotta alle categorie di raccomandazione di tipo product trending e product ranking. Questo perchè il product trending si basa sull'utilizzo di prodotti di tendenza nella home page, cosa che succede con l'utilizzo di score e popolarità, mentre per quanto riguarda il product ranking si basa sull'ordinare i prodotti in relazione alla probabilità che un prodotto venga cliccato.

La popolarità serve all'utente per capire quanto gli altri utenti apprezzino l'oggetto, differendo dallo score che offre un punteggio all'oggetto stesso, permettendo così ad ogni utente di avere la possibilità di capire i prodotti di tendenza del momento.

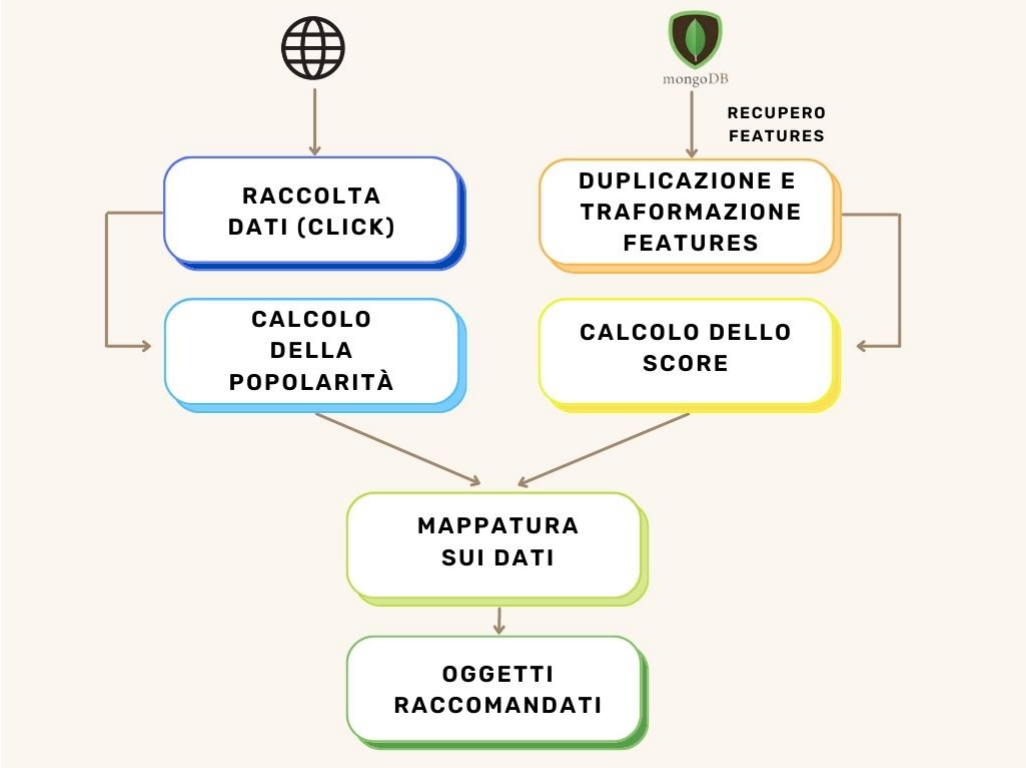

Figura 5.6. Flusso di processo score e popolarità

5.2 Public

La parte pubblica della piattaforma comprende tutto quello che l'utente può vedere e con cui può interagire.

Questa parte è divisa in varie sezione: homepage, pagina di ricerca, pagina di comparazione, pagina singola e pagina dei brand.

L'*homepage* presenta una barra di ricerca in cui cercare l'oggetto desiderato, la lista delle ultime uscite divise per categoria e i due menù per raggiungere la pagina di ricerca e di comparazione.
La *pagina di comparazione* permette di scegliere due oggetti appartenenti alla stessa categoria e li mette a confronto, mostrando tutte le loro specifiche con le varie differenze.
La *pagina di ricerca* mostra, in base alla categoria di oggetti selezionata, la lista di tutti gli oggetti appartenenti a quella categoria con la possibilità di utilizzare dei filtri per selezionare ciò che l'utente desidera.
La *pagina dei brand* permette di visualizzare, in base alla categoria di oggetti selezionata, la lista di tutti gli oggetti appartenenti a quel brand.
La *pagina singola* permette di visualizzare un singolo prodotto mostrando tutte le sue specifiche. Questa pagina è raggiungibile dalla pagina di ricerca o da quella dei brand.

5.2.1 Pagina di ricerca

La pagina di ricerca permetta all'utente di vedere tutti gli oggetti disponibili, con la possibilità di applicare dei filtri in modo da cercare l'oggetto desiderato.
I filtri possono essere diversi e dipendono dalla categoria dell'oggetto. In tutte le categorie sono presenti il prezzo e la lista dei brand. Mentre le altre specifiche vengono scelte quando vengono duplicate per lo score (cfr. Capitolo 5, Sezione 5.1.1, paragrafo duplicazione specifiche di Kagoo). Infatti nella pagina di duplicazione delle features è possibile impostare se la specifica deve essere utilizzata come filtro per la pagina di ricerca.
Le specifiche utilizzate per filtrare possono essere di due tipi: numeriche o booleane. Per filtrare sulle specifiche numeriche è stato necessario sviluppare uno slider

con range da 0 al valore massimo che la specifica ha nel db. Mentre per le specifiche booleane è possibile selezionare una checkbox che indica se si vuole che quella specifica sia presente nell'oggetto.
I prodotti che escono in questa pagina sono ordinati in base allo score che hanno, ed inoltre presentano anche la loro popolarità espressa tramite delle stelle (tra 0 e 5).
La parte di sviluppo più complicata è stata quella di gestire, a livello di back-end, tutti gli oggetti, questo perchè alcune categorie contengono più di 9000 oggetti. Quindi tutte le volte che veniva modificato un filtro questi oggetti andavano ricaricati, essendo la piattaforma stateless[2].
Il problema è stato risolto grazie allo sviluppo di una funzione di paginazione che utilizza le funzioni di aggregazione di MongoDB e all'utilizzo delle DataTables[3] per rappresentare la tabella con gli oggetti. In particolare la funzione di aggregazione carica solo i primi dieci risultati nella tabella, tenendo conto dei filtri desiderati. Quando si vuole cambiare pagina viene effettuata un'altra richiesta ajax[4] in cui viene specificato da che pagina bisogna partire per avere nella tabella gli oggetti corretti, successivi a quelli appena visti.
Inoltre per avere sempre a disposizione la lista dei filtri attivi sono presenti dei badge che indicano il filtro impostato, con i relativi valori.

[2]Piattaforma senza stato, quando si ricarica la pagina si perde tutto quello salvato dentro.

[3]DataTables è un plug-in per la libreria Javascript jQuery. Aggiunge diverse funzionalità avanzate a qualsiasi tabella HTML.

[4]AJAX permette di scambiare dati con un server e aggiornare parti di una pagina web senza ricaricare l'intera pagina.

5.3 Task asincrone

I dati utilizzati dalla piattaforma vengono scaricati ed aggiornati settimanalmente, per questo sono stati definiti dei task asincroni che permettono di aggiornare correttamente il db nel momento in cui vengono scaricati dei nuovi dati.
Le task asincrone vengono gestite con Celery (cfr Capitolo 3, Sezione 3.1.7), tramite il quale è possibile sottomettere dei task in modo asincrono. In questo modo, i task che risultano più pesanti del punto di vista esecutivo vengono gestiti in modo asincrono evitando al sistema di fermarsi.
Sono stati sviluppati tre task asincroni nella piattaforma per la gestione dello score e della popolarità: uno di gestione delle features appena scaricate, uno di gestione delle funzione di decadimento della popolarità e uno per inizializzare il db per la corretta gestione degli score.

5.3.1 Sincronizzazione features

Dal momento in cui Kagoo, il sito da cui venivano prelevati gli oggetti e le relative specifiche, ha chiuso, i dati vengono scaricati da Icecat, un altro sito che fornisce gli oggetti con le relative specifiche. La comodità dell'utilizzo di icecat sta nel fatto che fornisce, dove possibile, le specifiche sia in forma testuale che numerica. Questo rende il processo di duplicazione delle features spiegato precedentemente inutile, ma c'è ancora la necessità di collegare le specifiche testuali a quelle numeriche e renderle disponibili per lo score e per la pagina di ricerca.
Questa funzione, che viene eseguita una volta a settimana, permette appunto di collegare queste specifiche appena inserite nel db. In particolare vengono considerate tutte le specifiche che non presentano i campi *score*, *filter* e *output*, ovvero quelle appena aggiunte, e vengono effettuate le modifiche necessarie. Per le specifiche di tipo output avviene l'inserimento del campo *output* in quelle numeriche e

l'inserimento dei campi *score* e *filter* rispettivamente a true e false in quelle testuali. Alle specifiche di tipo input e transformed, per quanto riguarda le specifiche numeriche, viene aggiunto il campo *score* impostato a true. In questo modo le specifiche numeriche risultano disponibili per il calcolo dello score ed inoltre appaiono collegate alla loro versione testuale.

5.3.2 Funzione di decadimento

Questa funzione permette di calcolare il valore normalizzato della popolarità degli oggetti.
Mensilmente questa funzione viene eseguita per effettuare il calcolo della popolarità del mese appena trascorso, applicando la funzione di decadimento. In particolare, per ogni oggetto che è stato visitato almeno una volta viene calcolato il valore del numero di click, normalizzato ad 1, rispetto il numero totale di click che quella categoria ha ricevuto. Una volta calcolato questo numero viene applicata una funzione di decadimento esponenziale per i mesi precedenti. Il calcolo appena citato viene spiegato nel Capitolo 5, Sezione 5.1.3 *Calcolo della popolarità*.
Dopo aver calcolato la popolarità, basandosi sui valori del mese corrente e dei mesi passati, vengono aggiornati tutti i valori della popolarità che gli oggetti hanno. Di conseguenza nella pagina di ricerca verrà mostrato il nuovo valore della popolarità.

5.3.3 Inizializzazione del db

Questa funzione permette di inizializzare il db in modo tale da avere: tutte le specifiche scelte collegate, duplicate e trasformate, le specifiche numeriche mappate sugli oggetti, gli score già generati e calcolati.
In particolare, sono state scelte una decina di specifiche per ogni categoria, le più

importanti, utilizzate per inizializzare il sistema degli score e per avere delle specifiche su cui filtrare nella pagina di ricerca.
Nella funzione le specifiche scelte vengono duplicate (se necessario) e collegate tra di loro. Successivamente vengono mappate sugli oggetti, ovvero vengono inseriti i valori numerici tra le specifiche. Ed infine vengono aggiunte alla collezione di score così da poter calcolare gli score di ogni oggetto.
In pratica questa funzione riassume in parte il lavoro che la piattaforma può svolgere.

Capitolo 6

Use cases

In questo capitolo vengono trattati i diversi casi d'uso delle parti elaborate nella piattaforma.
In particolare si mostra come le parti implementate e spiegate nel capitolo precedente, possono essere usate nella piattaforma. Soffermandosi particolarmente sulla gestione delle features, la creazione degli score e la pagina di ricerca.

6.1 Gestione delle features

Nella sezione private viene gestita la duplicazione e la trasformazione delle features. Come si può notare in Figura 6.1, è possibile scegliere la categoria di oggetti e la percentuale di copertura della specifica, ovvero in quanti oggetti è presente la features. In seguito viene caricata la tabella in cui sono presenti: le diverse specifiche, i bottoni che permettono di sincronizzare le specifiche di Icecat e di mappare i dati. La tabella è ordinata secondo la copertura delle specifiche, dato presente nella tabella stessa.

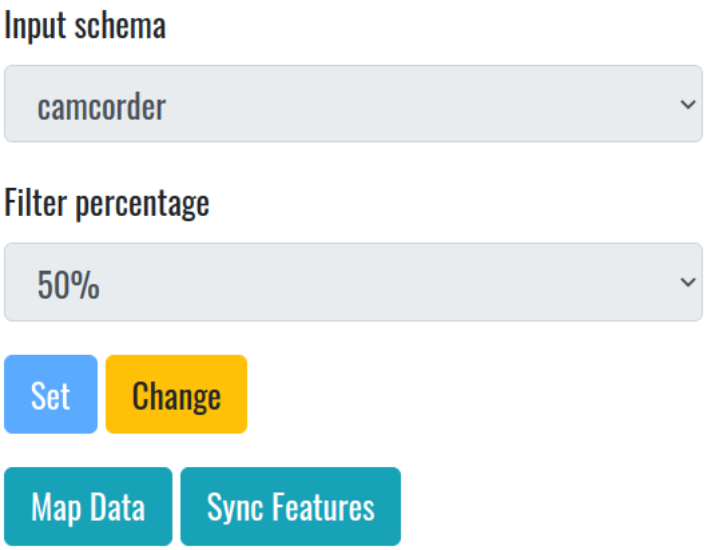

Figura 6.1. Selezione categoria e percentuale di filtraggio

La tabella, come si può notare in Figura 6.2, contiene il nome, il path, il tipo, la copertura e in quanti oggetti è presente. Inoltre sono presenti due bottoni: il bottone copia per la duplicazione e selezione delle trasformazioni da effettuare sulla feature e lo slider per impostare la specifica tra i filtri presenti nella pagina di ricerca.

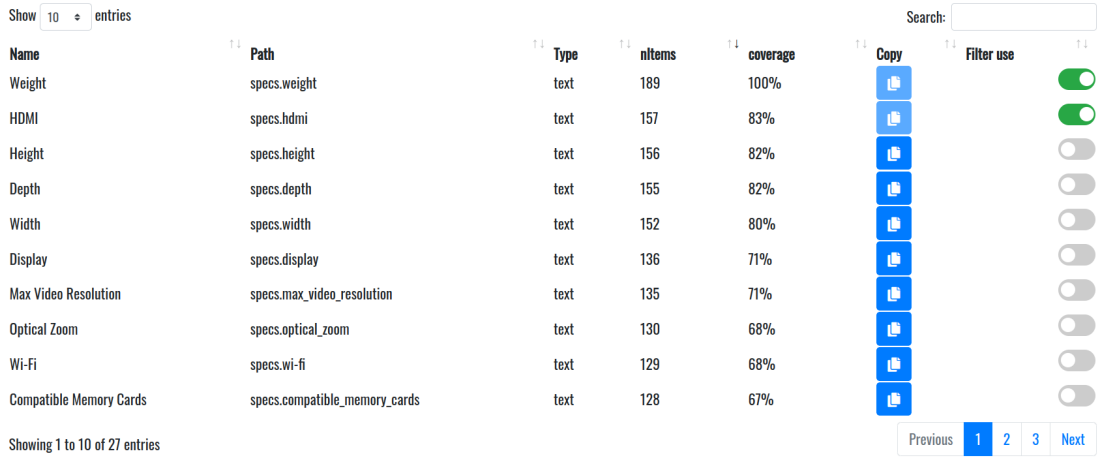

Figura 6.2. Tabella delle features

Premendo il bottone di duplicazione degli oggetti esce un form sotto forma di pop-up, mostrato in Figura 6.3.
In questo form è possibile impostare: il nome della nuova specifica (solitamente lo stesso di prima con l'aggiunta di Number o Boolean), se definire la specifica come filtro per la pagina di ricerca. Inoltre sono presenti due bottoni: un bottone per vedere i valori che ha la specifica negli oggetti (bottone con la i) e un bottone per aggiungere le trasformazioni da attuare sugli oggetti.
In Figura 6.3 si può notare che sono state aggiunte due trasformazioni:

- **Replace**: Sostituisce una parte della specifica testuale con quello che si desidera. Nell'esempio viene sostituita la stringa 'mm' con una stringa vuota così da rimuovere l'unità di misura dalla specifica presente negli oggetti. Inoltre la piattaforma gestisce i cambi di unità di misura nel caso in cui in alcuni oggetti siano presenti diverse unità di grandezza (es. peso espresso in chilogrammi o grammi).

- **Number**: Converte la specifica testuale in numero, intero o floating point. Infatti permette di scegliere la tipologia tramite un menù a tendina.

Va fatto notare che la scelte delle trasformazioni deve obbligatoriamente terminare con una Number o una Boolean (permette di convertire stringhe in valori true o false) così da poter creare delle specifiche o numeriche o booleane da poter usare per il calcolo dello score. Nel caso in cui non terminino con una di esse non è possibile salvare la nuova specifica.
Una volta salvato vengono create le varie entry nel db e la tabella si aggiorna disattivando il bottone di duplicazione e attivando lo slider del filtro.
Nel momento in cui vengono duplicate le specifiche desiderate è possibile mappare i dati sugli oggetti cliccando il bottone Map Data.

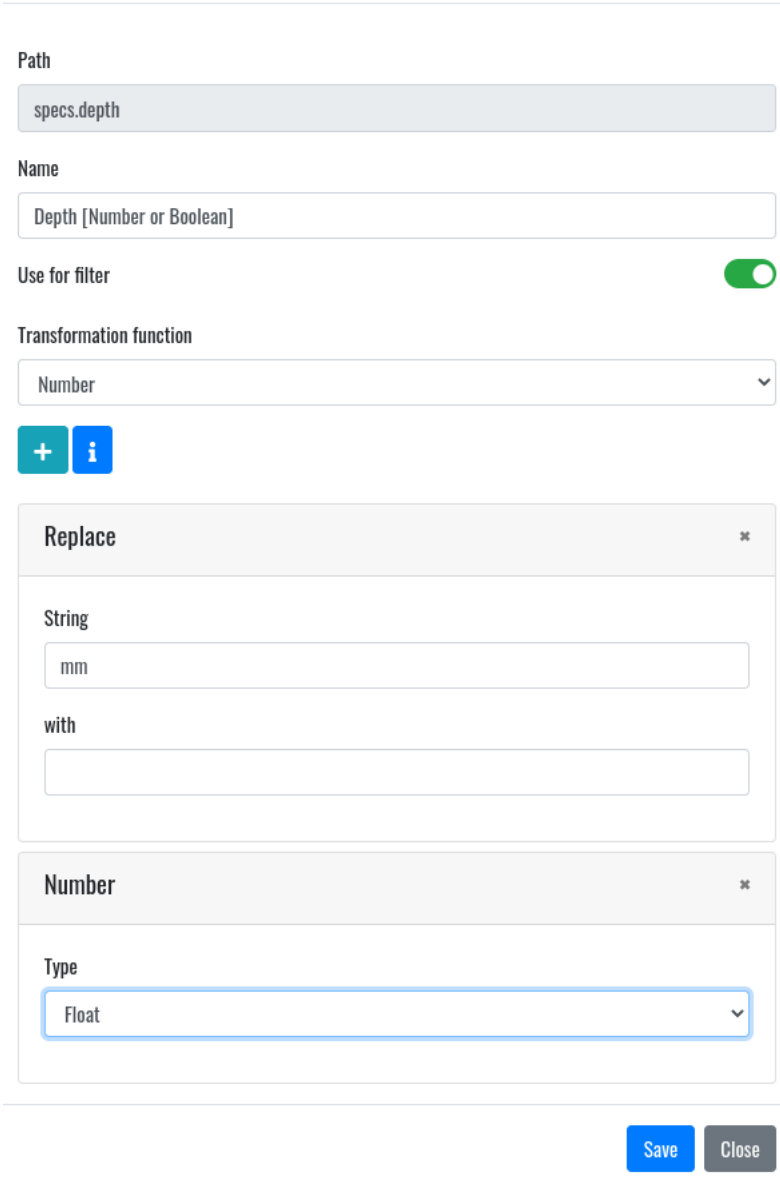

Figura 6.3. Form duplicazione e trasformazione features

6.2 Gestione dello score

Nella sezione private viene gestita la creazione degli score e la relativa selezione delle specifiche che li compongono.
Come si può notare in Figura 6.4, è possibile scegliere la categoria di oggetti, il nome che lo score ha e il nome dell'aggregato che si vuole creare per quello score. Una volta scelto il nome dello score si può premere sul bottone di Add Score e in automatico verrà generata la prima riga della tabella che conterrà solamente il nome.
Una volta scelto il nome dello score si può selezionare il nome dell'aggregato, il quale una volta scelto e premuto il bottone Add Aggregated feature sarà aggiunto alla tabella come seconda riga.

La tabella, come spiegato precedentemente e come si può notare in Figura 6.5, ha come titolo il nome dello score e come prima riga il nome dell'aggregato (riga in giallo) con il suo peso modificabile. Le righe sotto l'aggregato rappresentano l'insieme di specifiche che lo compongono con i relativi pesi modificabili.
Ogni riga può essere eliminata premendo il corrispettivo bottone (il cestino) ed è inoltre possibile modificare il peso che la specifica o l'aggregato può avere nello score.
Inoltre è possibile creare più aggregati per ogni score con specifiche diverse.

Per aggiungere una specifica all'interno dell'aggregato bisogna premere il bottone con il più. A quel punto uscirà un form, che si può vedere in Figura 6.6, nel quale è possibile scegliere le specifiche.
Le specifiche che si possono scegliere sono solo quelle che hanno subito il processo di duplicazione e trasformazione spiegato nel Capitolo 5, Sezione 5.1.1, ovvero quelle numeriche e booleane. Nel form è inoltre possibile scegliere se calcolare lo score di quella specifica all'opposto o no, ovvero anzichè calcolare il valore normalmente lo si sottrae a 1 calcolandone l'opposto (1-val).

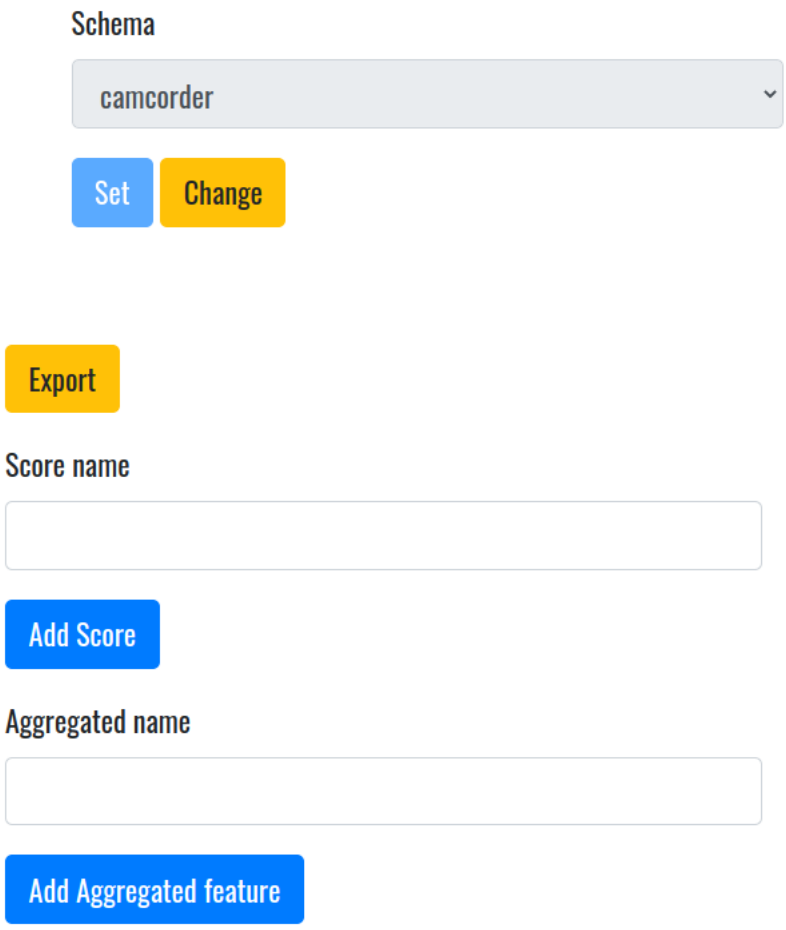

Figura 6.4. Selezione nomi score e aggregato per una categoria

Una volta scelte le specifiche che si vogliono utilizzare per calcolare lo score, presenti come badge a fondo del form, si può premere il bottone per salvare le features e in automatico usciranno nella tabella con relativo peso uguale 1.

6.2.1 Calcolo dello score

Dopo aver selezionato le specifiche che compongono gli score bisogna effettuare il calcolo dello score da inserire all'interno degli oggetti.

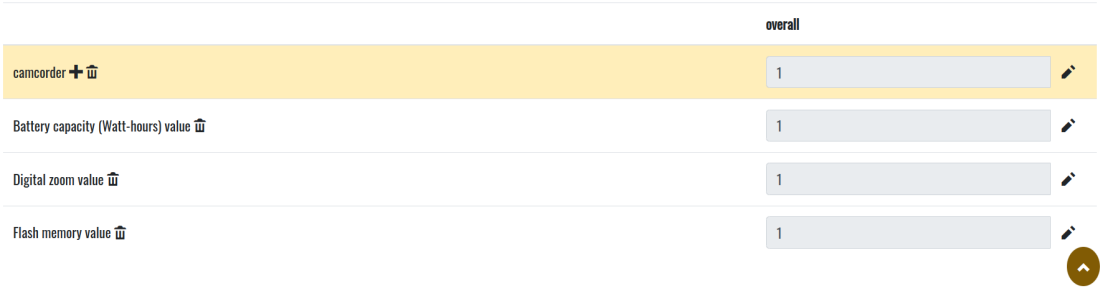

Figura 6.5. Tabella specifiche che compongono lo score

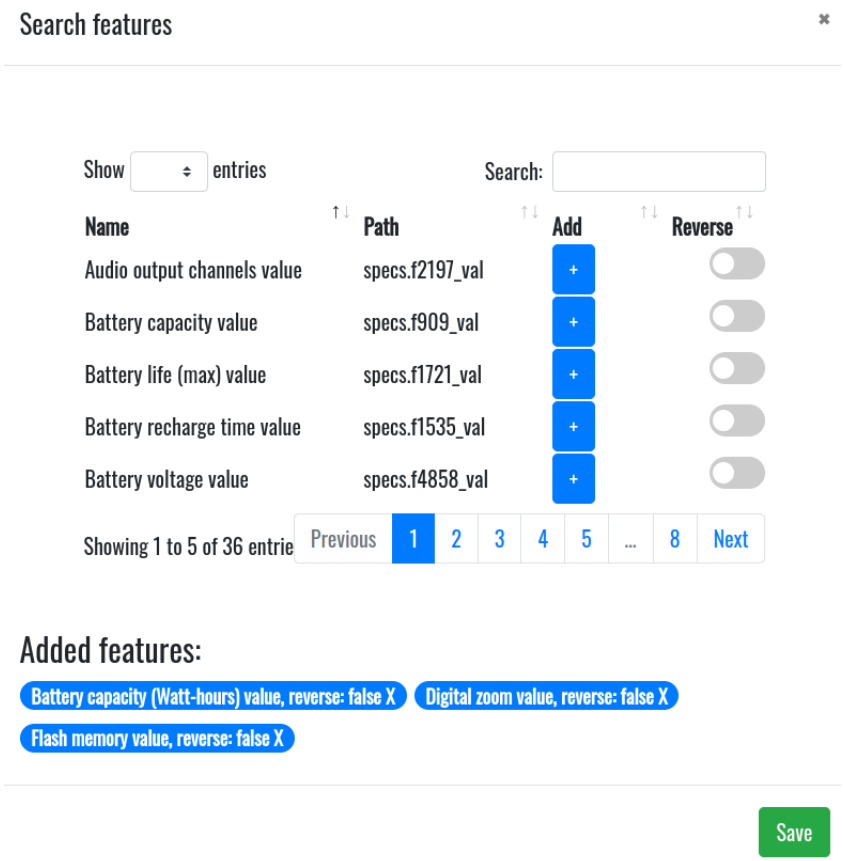

Figura 6.6. Form selezione specifiche per score

Nel menu Items della sezione private, Figura 5.2, esiste una voce *Compute scores* che fa partire un task asincrono che esegue i calcoli degli score e aggiorna tutti gli

oggetti con i rispettivi valori.

6.3 Pagina di ricerca

Nella pagina di ricerca sono presenti tutti i prodotti di una categoria scelta, con i relativi filtri.
Nella parte a sinistra si trovano i filtri da poter applicare ai prodotti, come si può vedere in Figura 6.7. É possibile scegliere i filtri mostrati in questa pagina andando nella sezione di gestione delle features mostrata nel Capitolo 6, Sezione 6.1 e selezionando lo slider della specifica che si vuole usare come filtro.
I filtri possono essere di tre tipi: gli *slider* che rappresentano le specifiche numeriche comprese tra 0 ed il massimo della specifica, le *checkbox dei brand* e le *checkbox presenti in other features* che rappresentano le specifiche booleane. Per i filtri numerici c'è la possibilità di usare lo slider o le input box in cui definire direttamente il valore, come si può vedere in Figura 6.8. I filtri dei brand si possono invece selezionare attraverso delle checkbox contenenti i brand da filtrare, in cui è specificato il numero di oggetti di quel brand specifico, come si può notare in Figura 6.9. Per quanto riguarda, invece, le specifiche booleane si trova una lista di checkbox sotto la categoria Other features in cui selezionandone una si filtrano gli oggetti che hanno quella specifica impostata a true, come si può notare sempre in Figura 6.9.
Una volta scelti i filtri questi appariranno come badge sopra la tabella dei prodotti, sotto la scritta Applied filters. In base al tipo di filtro selezionato, nel badge compariranno il valore massimo ed il minimo oppure se la determinata specifica è presente negli oggetti.
Quando vengono scelti i filtri la tabella si aggiorna automaticamente, quindi appena vengono modificati saranno subito indicati i prodotti desiderati e rispondenti alle caratteristiche scelte.

Nella tabella sono inoltre presenti score e popolarità, già descritti nei capitoli precedenti, ed è possibile ordinare la tabella in base ai loro valori, ma bisogna notare che la tabella è di default ordinata per score.

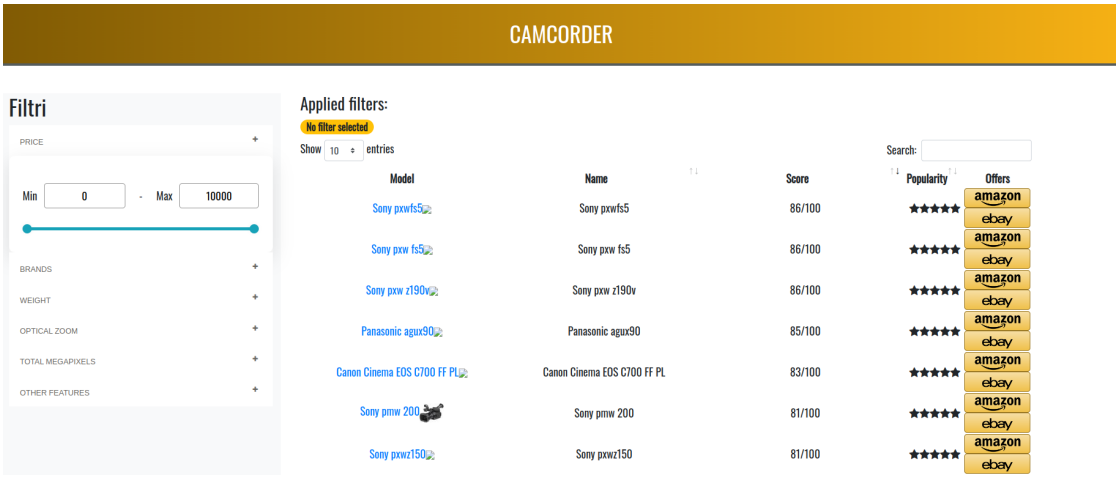

Figura 6.7. Pagina di ricerca senza filtri applicati

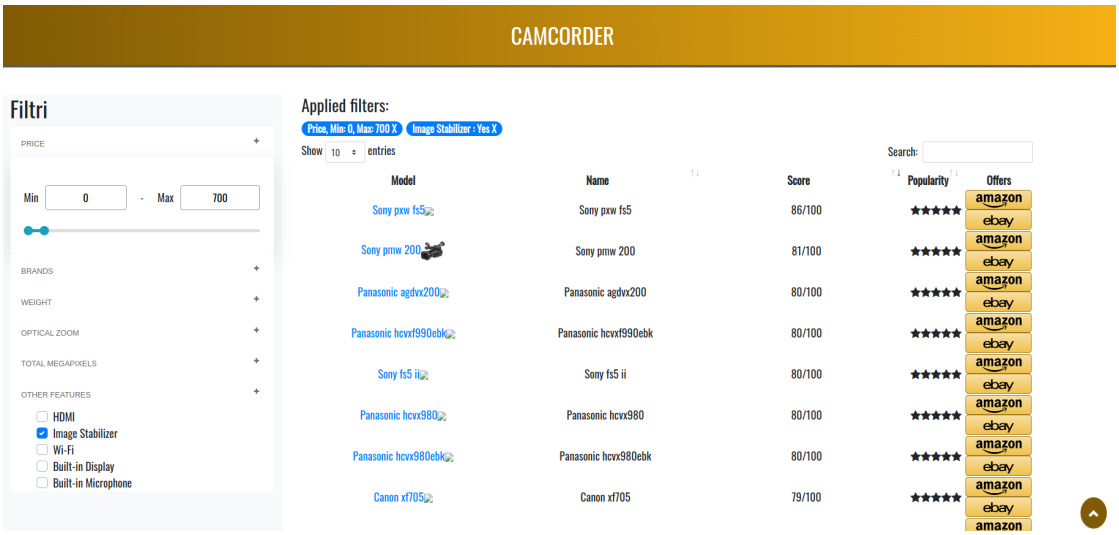

Figura 6.8. Pagina di ricerca con due filtri applicati

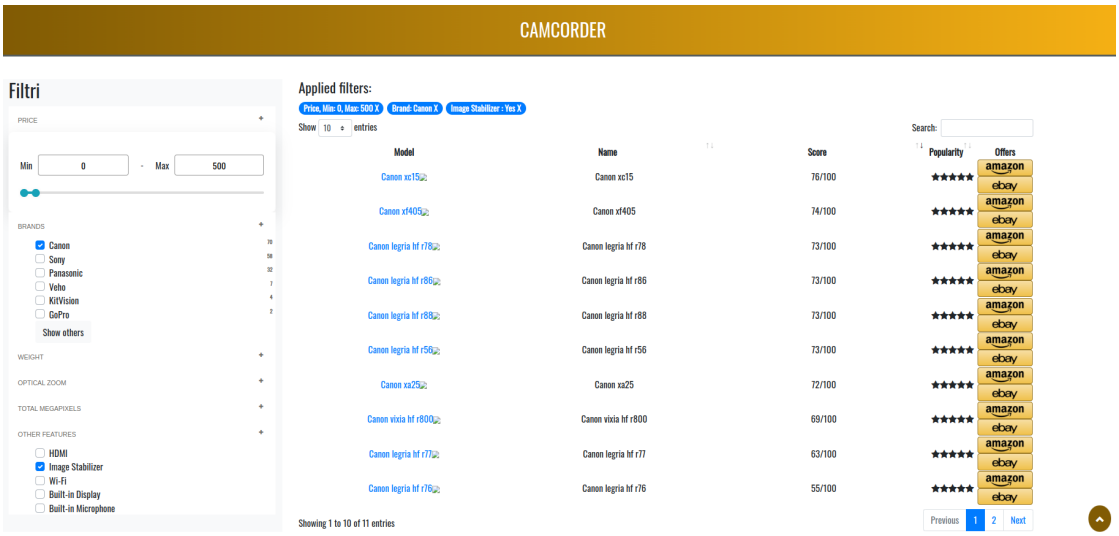

Figura 6.9. Pagina di ricerca con tre filtri applicati

Capitolo 7

Conclusioni e lavori futuri

L'obbiettivo principale della Tesi è quello di creare un sistema di raccomandazione non personalizzato capace di suggerire i prodotti agli utenti.
A seguito della progettazione, si può affermare di aver raggiunto i requisiti principali per avere un sistema di raccomandazione non personalizzato. In particolare è stato sviluppato un primo sistema basato sul valore che le specifiche hanno all'interno degli oggetti ed un secondo sistema basato sulla popolarità dell'oggetto, ovvero quante volte è stato visionato il prodotto dagli utenti.
In particolare, i risultati ottenuti possono essere visibili nella pagina di ricerca appositamente sviluppata, in cui si possono visionare i differenti punteggi degli oggetti con la rispettiva popolarità. Questi dati sono consultabili anche nelle pagine singole degli oggetti o in quelle di comparazione.
Per quanto riguarda il lavoro futuro, sarà possibile sviluppare altri metodi per raccomandare gli oggetti. Per ora lo score è calcolato solo sulle specifiche numeriche o booleane, quindi un possibile miglioramento potrebbe consistere nell'includere nel calcolo anche altre specifiche testuali definendo così un sistema di punteggi su misura.
Un altro miglioramento potrebbe essere quello di sviluppare la possibilità per gli

utenti di registrarsi. In questo modo si potrebbe passare da un sistema di raccomandazione non personalizzato ad uno personalizzato basato sullo studio del comportamento del singolo utente.

Bibliografia

[1] Celery. Celery - distributed task queue, 2020. URL https://docs.celeryq.dev/en/stable/.

[2] CSS. Css, 2022. URL https://it.wikipedia.org/wiki/CSS.

[3] Django. Why django?, 2020. URL https://www.djangoproject.com/start/overview/.

[4] Docker. Docker overview, 2020. URL https://docs.docker.com/get-started/overview/.

[5] Javascript. Javascript, 2022. URL https://developer.mozilla.org/en-US/docs/Web/JavaScript?retiredLocale=it.

[6] Joe Kimball, Ralph. Caserta. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data*. Wiley, 2004.

[7] Margy Kimball, Ralph. Ross. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley, 2013.

[8] MongoDB. What is mongodb?, 2021. URL https://www.mongodb.com/docs/manual/?_ga=2.217079122.557250938.1653226659-993624357.1632819934.

[9] Netflix. Netflix recommendations: Beyond the 5 stars, 2012. URL `https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429`.

[10] Netflix. Che cos'è netflix?, 2022. URL `https://help.netflix.com/it/node/412`.

[11] Pymongo. Pymongo 4.1.1 documentation, 2021. URL `https://pymongo.readthedocs.io/en/stable/`.

[12] TikTok. How tiktok recommends videos foryou, 2020. URL `https://newsroom.tiktok.com/en-us/how-tiktok-recommends-videos-for-you`.

[13] YouTube. On youtube's recommendation system, 2021. URL `https://blog.youtube/inside-youtube/on-youtubes-recommendation-system/`.

[14] Ardhian Agung Yulianto. Extract transform load (etl) process in distributed database academic data warehouse. *APTIKOM Journal on Computer Science and Information Technologies*, 4(2):64–71, 2019.