UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea Magistrale in Informatica

Basi di Dati 2

WTFunko - Un'e-commerce di Funko Pop!

WTFunko - An e-commerce of Funko Pop!

Candidati:

Antonio Garofalo

Marco Palmisciano

Luciano Bercini

Professori:

Prof. Genoveffa Tortora

Dr. Luigi Di Biasi

ANNO ACCADEMICO 2023/2024

Indice

Capitolo 1

Introduzione

1.1 Panoramica del progetto

Il progetto "**WTFunko**" ha come scopo lo sviluppo di un e-commerce dedicato alla vendita di "Funko Pop!". I Funko Pop! sono figure collezionabili in vinile prodotte dall'azienda americana Funko. Queste figure sono facilmente riconoscibili grazie al loro stile unico, caratterizzato da teste sovradimensionate e corpi piccoli. I Funko Pop! rappresentano una vasta gamma di personaggi provenienti da vari ambiti della cultura pop, tra cui film, serie TV, fumetti, videogiochi, sport e celebrità.

1.2 Obiettivi del progetto

L'obiettivo di questo progetto è lo sviluppo di un e-commerce (WTFunko) dedicato all'acquisto di Funko Pop!, utilizzando un database NoSQL per gestire i dati. I Funko Pop! sono figure collezionabili molto popolari, e creare una piattaforma dedicata alla loro vendita richiede un database flessibile e scalabile.

Capitolo 2

Descrizione del progetto

2.1 Requisiti funzionali

WTFunko è una piattaforma online che permette agli amministratori di gestire i prodotti, gli ordini, gli utenti e l'inventario mentre gli utenti possono effettuare tutte le operazioni tipiche di un e-commerce.

2.1.1 Funzionalità per gli utenti

- **Filtraggio prodotti**: gli utenti possono filtrare i prodotti in base ai diversi brand (Marvel, Disney, etc...).
- **Ordinamento prodotti**: è possibile visualizzare i prodotti ordinati per prezzo (crescente o decrescente), titolo (crescente o decrescente) e infine in base agli ultimi arrivi (ordine di inserimento).
- **Ricerca prodotti**: è possibile cercare i prodotti disponibili tramite termini di ricerca.
- **Gestione Account Utente**: gli utenti possono effettuare la registrazione di un account ed effettuare il login nel caso in cui sono già registrati. Possibilità di visionare il proprio storico ordini.
- **Visualizzazione dettagliata dei prodotti**: gli utenti possono visualizzare in dettaglio i prodotti e navigare prodotti simili.

2.1.2 Funzionalità di gestione

È possibile per i gestori effetuare operazioni di aggiunta, rimozione o aggiornamento delle informazioni inerenti a prodotti, ordini, utenti e inventario, garantendo la corretta operatività dell'e-commerce.

2.2 Requisiti non funzionali

WTFunko, essendo una piattaforma online adibita alla vendita dei Funko Pop!, deve soddisfare in particolare i seguenti requisiti non funzionali:

- **Flessibilità**: Data la dinamicità dei prodotti Funko Pop!, deve essere possibile per i gestori poter modificare la struttura dei dati in maniera semplice-Un esempio di tale modifica può essere l'inserimento di una nuova categoria di prodotti.
- **Scalabilità**: Dato il gran numero di appassionati dei Funko Pop!, il sistema deve poter gestire una grande mole di utenti e di transazioni, tenendo in considerazioni periodi di picco come festività e periodi di sconto.
- **Prestazioni**: Bisogna garantire tempi di risposta rapidi per tutte le operazioni effettuate dagli utenti.

2.3 Scelta del Database MongoDB

Per soddisfare i requisiti sopra descritti, è stata scelta una soluzione NoSQL. In particolare, MongoDB è stato selezionato per i seguenti motivi:

- **Schema flessibile**: MongoDB è schema-less, permettendo di aggiungere, rimuovere o modificare campi nei documenti in maniera semplice. Questa flessibilità è utile per gestire modifiche dei requisiti o aggiunte di funzionalità.
- **Scalabilità orizzontale**: MongoDB è progettato per scalare orizzontalmente, il che significa che può gestire un aumento del carico distribuendo

i dati su più server. Questo è particolarmente utile per gestire la mole di dati e transazioni presente in periodi di grande volume di traffico come ad esempio le festività e i periodi di sconto.

- **Performance elevata**: I database NoSQL sono ottimizzati per operazioni di lettura e scrittura ad alta velocità. Questo significa che possono gestire efficacemente un gran numero di query simultanee, garantendo tempi di risposta rapidi per gli utenti del sito e migliorando l'esperienza complessiva.

2.4 Struttura del Database

Il database è strutturato in modo da riflettere le principali entità del sistema:

- **Users**: Documenti che contengono informazioni sugli utenti, come username, mail e password (password sottoposta a funzione hash bcrypt).
- **Funko Pop! (Products)**: Documenti che descrivono i Funko Pop!, con dettagli come nome, categoria, prezzo, quantità disponibile, ecc.
- **Orders**: Documenti che registrano gli ordini effettuati dagli utenti, inclusi i prodotti acquistati e lo stato dell'ordine.

2.5 Modello dei Dati

Ecco come sono strutturati i documenti in MongoDB:

2.5.1 Funkos (Products)

```
funko: {
    _id (Integer): L'identificatore univoco del prodotto,
    title (String): Il nome del prodotto,
    product_type (String): La tipologia del prodotto,
    price (Real): Il prezzo del prodotto,
    quantity (Integer): La quantità nell'inventario del prodotto,
    interest (String List): Le categorie applicabili al prodotto,
    license (String List): Le licenze del prodotto,
    tags (String List): Le label applicate al prodotto,
    vendor (String): Il venditore,
    form_factor (String List): Il fattore di forma,
    feature (String List): Peculiarità del prodotto,
    related (Integer List): I prodotti più correlati,
    description (String): La descrizione del prodotto,
    img (String): L'URL dell'immagine del prodotto
}
```

2.5.2 Orders

```
orders: {
    _id (String): L'identificatore univoco dell'ordine,
    user (Object): {
        username (String): Il nome utente dell'amministratore,
        email (String): L'indirizzo email dell'amministratore
    },
    products (List): [
    {
        _id (Integer): L'identificatore univoco del prodotto,
        title (String): Il nome del prodotto,
        product_type (String): La tipologia del prodotto,
        price (Real): Il prezzo del prodotto,
        amount (Integer): La quantità ordinata del prodotto,
        interest (String List): Le categorie applicabili al prodotto,
        img (String): L'URL dell'immagine del prodotto
    }, ...],
    total (Real): Il totale dell'ordine,
    date (String): La data e l'ora dell'ordine,
    status (String): Lo stato dell'ordine
    }
}
```

2.5.3 Users

```
user: {
    "_id": L'identificatore univoco dell'utente,
    "username": Lo username dell'utente,
    "email": L'email dell'utente,
    "password": La password (hash) dell'utente
}
```

Capitolo 3

Dataset e fonte di dati

Il processo **ETL (Extract, Transform, Load)** è fondamentale per la gestione e l'integrazione dei dati all'interno di sistemi complessi come quelli utilizzati in un e-commerce. In questa sezione, approfondiremo ciascuna delle tre fasi del processo ETL e il loro ruolo nella preparazione del nostro dataset per la WebApp e-commerce di WTFunko.

3.1 Cos'è un Processo ETL?

ETL è l'acronimo di Extract, Transform, Load, che rappresenta le tre fasi principali di un processo di integrazione dei dati:

- **Estrazione (Extract)**: I dati vengono estratti da una o più fonti esterne. Nel nostro caso, la fonte è il dataset di Kaggle.
- **Trasformazione (Transform)**: I dati estratti vengono trasformati in un formato adatto all'analisi o all'inserimento nel database. Questo include operazioni di pulizia, normalizzazione, aggregazione, e arricchimento dei dati.
- **Caricamento (Load)**: I dati trasformati vengono caricati nel sistema di destinazione, che nel nostro caso è un database MongoDB.

3.1.1 Estrazione (Extract)

L'estrazione è la prima fase del processo ETL e consiste nel recuperare i dati da una o più fonti. Nel nostro caso, la fonte principale è stata il dataset di Kaggle. Durante questa fase, sono stati eseguiti i seguenti passaggi:

- **Identificazione delle fonti**: Il dataset specifico sui FunkoPop! è stato individuato su Kaggle, una piattaforma rinomata per la sua vasta gamma di dataset di alta qualità.
- **Download dei dati**: I dati sono stati scaricati in formato grezzo, precisamente il dataset era offerto in formato .csv.

3.1.2 Trasformazione (Transform)

La trasformazione è la fase più critica e complessa del processo ETL. Consiste nel convertire i dati estratti in un formato idoneo per l'analisi e l'utilizzo nel sistema di destinazione. Le operazioni eseguite durante questa fase includono:

- **Pulizia dei dati**: Sono stati identificati e corretti errori nei dati, come valori mancanti, duplicati ed errori tipografici. Ad esempio, abbiamo rimosso i record con dati mancanti critici e standardizzato i formati dei dati per coerenza (come l'eliminazione di codice HTML nelle descrizioni).
- **Aggregazione e arricchimento**: I dati sono stati aggregati e arricchiti con ulteriori informazioni, come per esempio la quantità in stock generata in maniera random per simulare un vero magazzino.
- **Conversione del formato**: I dati trasformati sono stati convertiti in formato JSON, ideale per l'integrazione con il nostro database MongoDB (ed eventualmente usabile come formato raw per API).

3.1.3 Caricamento (Load)

Il caricamento è la fase finale del processo ETL, in cui i dati trasformati vengono inseriti nel sistema di destinazione. Per il nostro progetto, il sistema di destinazione è un database MongoDB. Durante questa fase, sono state eseguite le seguenti operazioni:

- **Connessione al database**: È stata stabilita una connessione sicura al nostro database MongoDB.
- **Inserimento dei dati**: I dati in formato JSON sono stati caricati nel database, garantendo l'integrità e la coerenza dei dati.
- **Verifica e validazione**: Dopo il caricamento, i dati sono stati verificati e validati per assicurarsi che fossero correttamente inseriti e pronti per l'uso. Questo include la verifica della struttura del JSON e la coerenza dei dati con i nostri modelli dati nel backend.

Il processo ETL è stato essenziale per garantire che i dati utilizzati dalla nostra WebApp fossero accurati, aggiornati e facilmente accessibili. Grazie a questo processo, siamo in grado di offrire un'esperienza utente ottimale, con funzionalità avanzate di ricerca e gestione dei prodotti.

3.2 Dettagli Dataset

Il dataset utilizzato è Funk Pop Dataset, qui sono elencate alcune delle caratteristiche che ci hanno convinto a sceglierlo:

- Entry number: 1380 valori unici.
- Attribute number: 13 attributi utili.
- Attributi semanticamente chiari.
- Tool di scraping d'aggiornamento fornito dallo sviluppatore.
- Open-Source del dataset e del pacchetto di utils.

3.3 Esempio di dati incorretti

3.3.1 Codice HTML ridondante

```
<p>"It's Crunch Time" with the Count Chocula Pop! Tee!
   Count Chocula dashes under the light of the full moon;
   perhaps to get some delicious spooky cereal? This black
   crewneck tee will become a go-to tee in no time and you'
   ll love pairing it with a variety of bottoms for
   versatile casualwear options. Tee shirt is in youth
   unisex sizes: XS, SM, MD, LG and XL. 100% Cotton.</p>
<p><i>Please note: We are not able to exchange sizes or
   offer replacements for apparel that was sent in the size
    ordered.</i></p>
<p><a href="https://www.funko.com/sizingchart" title="
   Apparel Size Chart" target="_blank">Apparel Size Chart</
   a></p>
```

3.3.2 Attributi inutili

Alcuni attributi sono stati scartati, fondamentalmente legati a quando il dataset è stato generato:

- Created_at - Data di creazione della tupla.
- Published_at - Data di pubblicazione della tupla.
- Updated_at - Data di ultima modifica della tupla.
- Grid - Grid shopify (img).
- Handle - Alcuni metadati.

Capitolo 4

Tecnologie Adoperate

In questo capitolo, verranno descritte le tecnologie utilizzate nello sviluppo della webapp e-Commerce WTFunko.

4.1 Frontend

4.1.1 React

React[1] è una libreria JavaScript[2] open-source sviluppata da Facebook, utilizzata per la creazione di interfacce utente (UI). È particolarmente apprezzata per la sua capacità di rendere più semplice il processo di sviluppo di componenti UI riutilizzabili. Le caratteristiche principali di React includono:

- **Component-based**: React utilizza un approccio basato su componenti, dove ogni parte dell'interfaccia utente è rappresentata come un componente isolato. Questo favorisce una gestione più modulare, facilitando la manutenzione e il riutilizzo del codice.
- **Virtual DOM**: React utilizza un Virtual DOM (Document Object Model) per migliorare le performance delle applicazioni. Il Virtual DOM è una rappresentazione leggera della struttura dell'interfaccia utente che React mantiene in memoria e confronta con il DOM effettivo. Questo approccio consente a React di aggiornare solo le parti del DOM che sono cambiate, rendendo l'applicazione più veloce ed efficiente.

- **JSX: React introduce JSX (JavaScript XML)**, una sintassi che permette di scrivere codice HTML all'interno di JavaScript. Questo permette agli sviluppatori di definire facilmente la struttura dell'interfaccia utente direttamente nel codice JavaScript, migliorando la leggibilità e facilitando la manipolazione dinamica dei componenti.

4.1.2 Vite

Vite[3] è un framework di sviluppo front-end moderno e leggero, progettato per facilitare la creazione di applicazioni web veloci. Alcune caratteristiche salienti di Vite includono:

- **Dev server veloce**: Vite è noto per il suo dev server veloce che supporta il caricamento rapido dei moduli. Utilizza ES Modules nativi del browser durante lo sviluppo, evitando la necessità di compilazioni e bundling pesanti durante il processo di sviluppo.

- **Supporto per ES Modules**: Vite supporta nativamente ES Modules, consentendo agli sviluppatori di importare moduli JavaScript direttamente all'interno del browser. Questo approccio è più efficiente rispetto agli strumenti tradizionali che richiedono la creazione di bundle statici.

- **Plugin System**: Vite offre un sistema di plugin estensibile che permette agli sviluppatori di personalizzare e estendere facilmente il funzionamento del framework secondo le proprie esigenze. Questo include supporto per plugin che migliorano il supporto di vari tipi di file (come TypeScript, React, CSS, ecc.) e ottimizzazioni per le prestazioni.

4.1.3 Integrazione di React con Vite

React e Vite sono complementari e possono essere integrati insieme per sfruttare al meglio le loro rispettive caratteristiche. Per utilizzare React con Vite, è possibile configurare Vite per supportare React utilizzando il plugin ufficiale

@vitejs/plugin-react. Questo plugin abilita il supporto per JSX e facilita l'integrazione di React all'interno di un progetto Vite senza la necessità di configurazioni complesse.

In sintesi, React è una libreria per la creazione di interfacce utente component-based, nota per il suo Virtual DOM e l'uso di JSX. Vite, d'altra parte, è un framework di sviluppo front-end moderno, veloce e leggero, ideale per il caricamento rapido dei moduli durante lo sviluppo. Insieme, React e Vite consentono di creare applicazioni web moderne e performanti, migliorando la produttività degli sviluppatori e l'esperienza dell'utente finale.

4.2 Backend

4.2.1 FastAPI

Il backend del nostro WTFunko Store è stato sviluppato utilizzando FastAPI[4], un moderno framework web per Python[5] che permette di creare API RESTful in modo rapido ed efficiente. FastAPI è stato scelto per le sue eccellenti performance, la facilità d'uso e la capacità di generare automaticamente documentazione interattiva.

Caratteristiche principali di FastAPI

- **Performance elevate**: FastAPI è basato su Starlette per il web e Pydantic per la gestione dei dati. Questo consente al framework di offrire performance molto elevate, paragonabili a quelle di Node.js e Go, rendendolo adatto per applicazioni web ad alta intensità di traffico.
- **Facilità di utilizzo**: FastAPI permette di definire le API utilizzando le annotazioni di tipo di Python. Questo rende il codice più leggibile e aiuta a prevenire errori, migliorando la produttività degli sviluppatori.
- **Validazione automatica dei dati**: Grazie a Pydantic, FastAPI esegue automaticamente la validazione dei dati di input e di output, assicurando che i dati rispettino le specifiche definite dagli sviluppatori.
- **Documentazione automatica**: FastAPI genera automaticamente documentazione interattiva delle API utilizzando OpenAPI e Swagger UI. Questo facilita la comprensione e l'uso delle API da parte degli sviluppatori e dei team di integrazione.
- **Supporto per l'asincronia**: FastAPI supporta nativamente le operazioni asincrone utilizzando async e await, permettendo di gestire un numero elevato di richieste concorrenti in modo efficiente.

Struttura del progetto

Il progetto backend è strutturato in modo modulare per facilitare la manutenzione e l'espansione futura. Di seguito una panoramica delle principali componenti:

- **App principale**: Il punto di ingresso dell'applicazione, dove viene creata l'istanza di FastAPI e vengono registrate le varie route.
- **Router**: I vari endpoint dell'API sono organizzati in router separati per diverse funzionalità (ad esempio, gestione dei prodotti, gestione degli ordini, autenticazione degli utenti). Questo consente di mantenere il codice organizzato e facilmente navigabile.
- **Modelli**: I modelli Pydantic sono utilizzati per definire la struttura dei dati di input e output, garantendo la validazione automatica dei dati.
- **Database**: La connessione al database MongoDB è gestita utilizzando un driver MongoDB per Python (come Motor o PyMongo). Le operazioni di database sono incapsulate in repository dedicati per mantenere il codice pulito e separare la logica di business dalle operazioni di persistenza dei dati.
- **Middleware**: FastAPI permette di aggiungere middleware per gestire funzionalità cross-cutting come l'autenticazione, il logging e la gestione degli errori.

Vantaggi dell'uso di FastAPI

- **Velocità di sviluppo**: La sintassi semplice e le funzionalità integrate di FastAPI accelerano il processo di sviluppo, riducendo il tempo necessario per implementare nuove funzionalità.
- **Performance**: La gestione asincrona delle richieste e l'efficienza del framework garantiscono alte performance, rendendo FastAPI ideale per applicazioni scalabili e ad alta intensità di traffico.

- **Documentazione**: La generazione automatica di documentazione facilita la collaborazione tra i team e migliora la comunicazione con gli sviluppatori che integrano le API.

4.2.2 Sicurezza

La sicurezza delle API è essenziale per proteggere i dati sensibili degli utenti e garantire che solo gli utenti autorizzati possano accedere a determinate risorse. Nel nostro backend FastAPI, abbiamo implementato diverse misure di sicurezza, tra cui il middleware CORS, la suddivisione delle API in base ai livelli di accesso e l'uso di token per l'autenticazione.

Middleware CORS (Cross-Origin Resource Sharing)

Il middleware CORS è utilizzato per controllare quali domini possono accedere alle API del server. Questo è particolarmente importante per le applicazioni web che fanno richieste AJAX a un dominio diverso da quello da cui è stata caricata l'applicazione.

4.2.3 Documentazione API

Nella creazione delle route per le nostre REST API abbiamo pensato di avere un accesso completo sul database tramite lo sviluppo di tutte le operazioni **CRUD (Create, Read, Update, Delete)**. Inoltre è stato implementata una route di default del server "**/docs**" che permette di avere una visione d'insieme di tutti gli endpoint REST API e la possibilità di utilizzarli (secondo un criterio di accesso).

Di seguito sono mostrati tutti gli endpoint:

Users

Endpoint	**Metodo**	**Sommario**	**Accesso**
/login	POST	User Login	Users
/signup	POST	User Signup	Users
/deleteAccount/{username}	DELETE	User Account Delete	Users
/getAllUsers	GET	Get All Users	Admin
/getUser	GET	Get User Information	Admin
/insertUser	POST	Insert New User	Admin
/deleteUser/{username}	DELETE	Delete User	Admin
/clearUsers	DELETE	Delete All Users	Admin
/updateUser	PUT	Update User Information	Admin

Tabella 4.1: API del Server FastAPI per gli utenti

Orders

Endpoint	Metodo	Sommario	Accesso
/getAllOrders	GET	Get All Orders	Admin
/getUserOrders	GET	Get User Orders	Users
/getOrderInfo	GET	Get Order Information	Users
/insertOrder	POST	Insert New Order	Users
/deleteOrder/{order_id}	DELETE	Delete Order by id	Admin
/deleteOrder/{username}	DELETE	Delete Order by username	Admin
/clearOrders	DELETE	Delete All Orders	Admin
/updateOrder	PUT	Update Order Information	Admin

Tabella 4.2: API del Server FastAPI per gli ordini

Products

Endpoint	Metodo	Sommario	Accesso
/getProducts	GET	Get Products	Products
/getAllProducts	GET	Get All Products	Products
/getUniqueProductsCount	GET	Get Unique Products Count	Products
/getByID/{product_id}	GET	Get Product by ID	Products
/getByCategory/{category}	GET	Get Products by Category	Products
/getByProductType/{product_type}	GET	Get Products by Product Type	Products
/getBySearch/{search_string}	GET	Get Products by Search String	Products
/sortingByName	GET	Sort Products by Name	Products
/sortingByPrice	GET	Sort Products by Price	Products
/insertProduct	POST	Insert Product	Admin
/deleteProduct/{product_id}	DELETE	Delete Product	Admin
/clearProducts	DELETE	Delete All Products	Admin
/updateProduct/{product_id}	PUT	Update Product	Admin

Tabella 4.3: API del Server FastAPI per i prodotti

4.2.4 Ottimizzazioni

Essendo che abbiamo voluto sfruttare tutta la potenza di calcolo necessaria e ottimizzare i vantaggi di un backend strutturato per un e-commerce tramite tecnologia NoSQL, abbiamo implementato diverse ottimizzazioni lungo il percorso di sviluppo. Queste ottimizzazioni ci hanno permesso di ridurre significativamente i tempi di attesa per le risposte alle query e di ottimizzare l'utilizzo della memoria.

Qui elencati alcune delle ottimizzazioni fatte:

- **Loading parziale** - Il database è caricato dal frontend parzialmente (circa 20 oggetti alla volta e salvati in cache) ciò ci permette di avere una latenza decisamente minore invece di ottenere lunghi tempi di attesa per caricare l'intero catalogo. Inoltre vengono calcolati in anticipo quantità di prodotti e altre informazioni del database utilizzabili in locale dal frontend.

- **Indici su MongoDB** - Sono stati utilizzati indici sui campi più utilizzati dalle query che riguardano i prodotti. Questo ha permesso l'ottimizzazione dei tempi di esecuzione delle query.

- **Generalizzazione Fetch** - La funzione di fetch è stata sviluppata in maniera generica e adattata ad ogni metodo possibile (GET, POST, etc...) seguendo la logica della libreria Axios ma sfruttando i vantaggi della funzione built-in fetch di Javascript.

- **Cache della memoria browser** - Utilizzando le chiamate built-in del framework React (i.e localStorage) siamo riusciti ad ottimizzare il salvataggio di informazioni (come quelle dell'utente loggato) in modo tale da diminuire le chiamate inutili al backend e l'esecuzione di query futili.

- **Caricamento dinamico degli elementi** - Gli elementi vengono reindirizzati in maniera dinamica e non statica (proprio grazie al framework di React) rendendo l'applicazione decisamente più leggera.

- **Adattamento delle funzioni in modalità async** - Le funzioni sono state sviluppate in maniera concorrente cercando di avere uno schedule di esecuzione sincronizzato in un ambiente asincrono.

Molte altre ottimizzazioni sono state svolte per portare il sistema in uno stato prestazionale elevato e, come detto prima, sfruttare al massimo i vantaggi offerti da un architettura client-server basata su tecnologie NoSQL.

4.3 MongoDB

Nel progetto WTFunko, abbiamo adottato MongoDB[6] come database principale per svariati vantaggi che offre agli e-commerce, tra cui la scalabilità, la flessibilità dello schema e le prestazioni nelle operazioni di lettura e scrittura.

4.3.1 Struttura delle Collezioni

Abbiamo organizzato i dati in MongoDB utilizzando tre collezioni principali:

- **Users**: Questa collezione contiene i dati relativi agli utenti registrati sul nostro sito.
- **Orders**: Qui vengono memorizzati tutti gli ordini effettuati dai clienti. Ogni ordine è rappresentato da un documento che contiene dettagli come prodotti acquistati, prezzo totale, stato dell'ordine, e altre informazioni correlate.
- **Products**: Questa collezione contiene i dati sui prodotti disponibili nel nostro catalogo. Ogni documento prodotto include informazioni come nome, descrizione, prezzo, categorie, e altre caratteristiche specifiche del prodotto.

4.3.2 Benefici ottenuti da MongoDB

MongoDB si è dimostrato particolarmente adatto per WTFunko per diverse ragioni:

- **Flessibilità dello schema**: Con MongoDB, abbiamo la libertà di aggiungere nuovi campi ai documenti esistenti senza dover modificare uno schema rigido. Questo si è rivelato utile nel gestire variazioni nei dati dei prodotti o nelle preferenze degli utenti nel tempo.
- **Query veloci**: Grazie alla sua architettura di database NoSQL, MongoDB consente di eseguire query molto efficienti anche su grandi volumi di dati, supportando così una risposta rapida alle richieste dei clienti.

- **Integrazione con Python**: Utilizziamo la libreria MongoDB per Python (PyMongo) per interfacciare il nostro backend FatAPI con il database in maniera facile e veloce. Questo ci consente di scrivere codice più pulito e leggibile per gestire l'accesso ai dati e le operazioni di manipolazione delle collezioni direttamente da Python ed elaborare le response delle REST API per gli utenti.

Le query in MongoDB sono state scritte in maniera intellegibile tramite Python con funzioni di aggregazione (filtri) fatte ad-hoc per il nostro sistema di vendita. All'accensione del server il sistema riconosce se il database è già popolato dal nostro dataset, in caso affermativo procede con il runtime in caso negativo viene fatta una routine di popolamento secondo i nostri schemi e modelli.

I modelli sono stati costruiti rispecchiando la struttura JSON proposta nella descrizione del progetto nel capitolo 2 di questa documentazione.

Capitolo 5

Conclusione

Nel corso di questo progetto è stata presentata e sviluppata l'idea di un e-commerce di Funko Pop! utilizzando un database NoSQL. La scelta di MongoDB permette di soddisfare i requisiti di scalabilità, flessibilità e performance necessari per un sistema di questo tipo.

Grazie alla sua capacità di scalare orizzontalmente, MongoDB è in grado di gestire un elevato volume di traffico e di transazioni, garantendo prestazioni elevate anche durante i picchi di utilizzo. La sua struttura flessibile consente di adattare facilmente il database ai cambiamenti nei requisiti dei dati, permettendo una rapida evoluzione del sistema senza la necessità di complesse migrazioni di schema.

Inoltre, le potenti funzionalità di query di MongoDB offrono un accesso efficiente e rapido ai dati, migliorando l'esperienza utente e ottimizzando le operazioni quotidiane del negozio online. Complessivamente, l'adozione di MongoDB fornisce una solida base per sviluppare un e-commerce di successo, capace di crescere e adattarsi alle esigenze del mercato.

5.1 Sviluppi Futuri

1. **Integrazione di Sistemi di Pagamento**
    - Descrizione: Integrare vari sistemi di pagamento come PayPal, Apple Pay, Google Pay e altri.
    - Obiettivi: Aumentare la flessibilità e le opzioni di pagamento per i clienti.
    - Vantaggi: Miglioramento dell'esperienza utente, riduzione dell'abbandono del carrello.
2. **Recensioni e Valutazioni dei Prodotti**
    - Descrizione: Consentire agli utenti di lasciare recensioni e valutazioni sui prodotti acquistati.
    - Obiettivi: Costruire fiducia e trasparenza con i clienti.
    - Vantaggi: Aumento delle conversioni, feedback utile per migliorare i prodotti.
3. **Wishlist**
    - Descrizione: Consentire agli utenti di aggiungere dei prodotti nella loro lista desideri.
    - Obiettivi: Permettere all'utente di accedere ai prodotti desiderati più semplicemente.
    - Vantaggi: Aumento delle conversioni.
4. **Programma di Fedeltà**
    - Descrizione: Creare un programma di fedeltà che premi i clienti ricorrenti con sconti, punti e offerte esclusive.
    - Obiettivi: Fidelizzare i clienti e aumentare la retention.
    - Vantaggi: Incremento delle vendite ripetute, creazione di una base di clienti fedeli.

5. **App Mobile**

   - Descrizione: Sviluppare un'app mobile per iOS e Android che permetta agli utenti di acquistare Funko Pop, gestire gli ordini e ricevere notifiche su offerte e nuovi prodotti.
   - Obiettivi: Offrire un'esperienza di acquisto ottimizzata per dispositivi mobili.
   - Vantaggi: Aumento delle vendite da mobile, miglioramento dell'engagement degli utenti.

6. **Funzionalità di Realtà Aumentata (AR)**

   - Descrizione: Integrare la realtà aumentata per permettere agli utenti di visualizzare i Funko Pop nel loro ambiente prima di acquistarli.
   - Obiettivi: Migliorare l'esperienza di acquisto online.
   - Vantaggi: Riduzione dei resi, aumento della soddisfazione del cliente.

7. **Supporto Multilingua e Multivaluta**

   - Descrizione: Aggiungere supporto per più lingue e valute per espandere il mercato a livello globale.
   - Obiettivi: Aumentare la base di clienti internazionali.
   - Vantaggi: Incremento delle vendite globali, miglioramento dell'accessibilità.

Queste sono le principali idee per gli sviluppi futuri per ampliare l'insieme di funzionalità offerte da WTFunko.

Riferimenti

[1] Facebook, Inc., *React Documentation*, Created by Jordan Walke, 2013.

[2] ECMAScript Standards Committee, *JavaScript Documentation*. Ecma International, 1995.

[3] Evan You, *Vite Documentation*, Creator of Vue.js, 2020.

[4] Sebastián Ramírez, *FastAPI Documentation*, Born on 7th July 1985, 2018.

[5] Guido van Rossum, *Python Documentation*. Python Software Foundation, 1991.

[6] MongoDB, Inc., *MongoDB Documentation*, Founded by Dwight Merriman, Eliot Horowitz, and Kevin P. Ryan, 2009.