Progetto di Introduzione alla programmazione per il Web

Matteo Battilana - 180209 Massimo Girondi - 178114 Daniele Isoni - 181839

Versione del 4 ottobre 2017

Indice

1 Tecnologie utilizzate

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed,

volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

2 Database

3 Librarie incluse

4 Autocompletamento

L'autocompletamento presente nella barra di ricerca è stato realizzato mediante la libreria JavaScript **bootstrap-ajax-typeahead**, la quale recupera i suggerimenti per l'autocompletamento dalla servlet AutoCompleteServlet mediante richieste Ajax, a cui la servlet rispone in formato JSON.

La servlet effettua una ricerca per similarità (secondo l'algoritmo di Jaro-Winkler) all'interno di una lista, la quale viene caricata all'avvio dell'applicazione analizzando i nomi dei prodotti presenti nel database e aggiornata a intervalli prestabiliti (ogni 30 minuti) mediante le classi Executors e ScheduledExecutorService presenti all'interno di Java stesso.

Abbiamo scelto questa implementazione per evitare di sovraccaricare il database eseguendo una query a ogni richiesta di autocompletamento (e quindi a ogni tasto premuto). Un'altra opzione, poi scartata, era aggiornare la lista dell'autocompletamento a ogni modifica della tabella dei prodotti, ma questo avrebbe rallentato ogni operazione sulla tabella stessa.

L'algoritmo di Jaro-Winkler è utilizzato anche durante la scansione del database, per evitare di inserire nella lista dei suggerimenti titoli doppi o troppo simili (ad esempio "frigo" e "frigorifero") dato che questi verrebbero comunque selezionati al momento della ricerca, dove è applicata ancora una volta la similarità.

5 Risultati della ricerca

la ricerca tra i prodotti è realizzata mediante una servlet che elabora la richiesta e restituisce la lista dei prodotti sottoforma di JSON. L'unico parametro obbligatorio è `q` che rappresenta la query di ricerca. I parametri opzionali sono:

- `p`: la pagina desiderata (nel caso di una ricerca multipagina)
- `s`: il metodo di ordinamento dei risultati, di default alfabetico (0), alfagetico inverso(1), prezzo crescente (2), prezzo decrescente(3), valutazione decrescente(4), numero recensioni decrescente(5).
- `c`: la categoria
- `min` e `max`: gli estremi dei prezzi.
- `minRev`: il minimo del valore delle recensioni.

Il controllo sul nome del prodotto e sul range di prezzo è effettuato all'interno del DAO del prodotto (mediante SQL per il prezzo e algoritmo di Jaro-Winkler per il nome), mentre il controllo sulla categoria e sul valore minimo della media delle recensioni è effettuato mediante un predicato invocato dal metodo `removeIf` della lista prodotti. Oltre alla lista prodotti nel risultato sono restituiti anche il numero di pagine, il tipo di ordinamento usato e la pagina corrente.

6 Logging

Il logging, fondamentale per un'applicazione di questo tipo, è stato implementato mediante la libreria Apache Log4j, che consente di gestire in modo molto semplice i vari livelli di log e il salvataggio automatico sul disco.

Abbiamo poi creato un wrapper attorno alla libreria per limitare il più possibile il numero di istruzioni richieste per scrivere un messaggio nel log.

I log vengono salvati nella cartella `$CATALINA_HOME/logs` in file testuali che iniziano con il nome `BuyHub`. La cartella `$CATALINA_HOME` è in genere la cartella di installazione di TomCat o GlassFish, ma potrebbe variare in base alla configurazione del sistema.

7 Dettagli implementativi

7.1 Orari di apertura e negozi solo online

Per semplificare la struttura del DB abbiamo deciso di implementare i vari punti vendita dei negozi come una relazione uno a molti tra `shop` e `coordinate`. All'interno della tabella `coordinate` abbiamo inserito il campo `opening_hours` che contiene gli orari del punto vendita, nel caso in cui per il cliente sia disponibile il ritiro in negozio. Nel caso di negozi che effettuano soltanto vendita online questo campo resta vuoto e viene sostituito all'interno della pagina web con un messaggio che specifica le modalità di vendita solo online del negozio.

8 Upload/Download

L'upload dei file è stato gestito mediante la libreria `com.oreilly.servlet`, mentre il salvataggio dei file è gestito da alcuni metodi all'interno della classe `Utility`.

In particolare, il metodo `saveJPEG` si occupa di salvare un immagine, dandogli un nome casuale basato su un UUID, convertendola a JPEG, mediante la libreria ImageIO, integrata in Java. Purtroppo questa libreria non gestisce correttamente le trasparenze nelle immagini PNG, e al momento, non è ancora stata trovata una soluzione per ovviare a questo problema, in quanto non è possibile dedurre se questo errore di codifica è avvenuto o no.

I file vengono poi salvati in una cartella definita all'interno del file `config.properties`, nel quale è possibile inserire sia un path relativo (in base alla propria cartella `catalina.base`), oppure un percorso assoluto, che verrà automaticamente interpretato e utilizzato per il salvataggio.

Il dowload avviene mediante la servlet `UploadedContentServlet`, che risponde alle richieste che arrivano alle url del tipo `/UploadedContent/*`, in modo tale da simulare a tutti gli effeti una cartella. Il file richiesto viene inviato in modalità "inline", e con il relativo MIME impostato. Nel caso il file non sia presente nel disco, viene restituita un'immagine che indica la non presenza del file.

La scelta di utilizzare questo sistema è stata piuttosto ardua, in quanto, non volendo usare path assoluti, l'unica opzione per caricare un file direttamente con Java, era caricarlo nella cartella di esecuzione (la `contextPath`). Nel momento in cui però sarebbe avvenuto un redeploy tutti i file sarebbero stati cancellati, oltre al fatto che alcuni server Java non permettono la scrittura nella `contextPath`. Altri server, invece, mantengono direttamente tutto il `.war` dell'applicazione in RAM, senza creare una cartella nel disco rigido, e creare cartelle locali alla `contextPath` in questo contesto è abbastanza sconsigliato (aumenterebbe a dismisura l'uso di RAM).

Un'altra soluzione, volendo realizzare un prodotto per il mondo reale, sono i CDN, in particolare quelli specifici per le immagini, come cloudinary.com o imgix.com. Questi servizi, però, sono in genere a pagamento e non permettono il controllo completo delle proprie immagini. Per questi motivi la scelta è ricaduta nel caricare le immagini in un path locale alla cartella di esecuzione, ma esterno al `contextPath`, in modo da preservare i file tra le diverse esecuzioni e deploy dell'applicazione.

9 Altri capitoli

10 Conclusioni