Politecnico di Torino

Corso di Laurea in Ingegneria Gestionale L8

Monografia di Laurea

La Baionetta Reader

Relatore:
Prof. Fulvio Corno

Candidato:
Fabio Molinaris

Novembre 2017

Indice

1. Capitolo 1

1.1. Proposta di progetto

Il progetto fa riferimento, nello specifico, ad un blog (labaionetta.blogspot.it).

Si vuole arrivare ad ottenere un applicativo desktop, costantemente aggiornato che renda facile e comoda la lettura e l'esplorazione degli articoli pubblicati.

1.1.1. Descrizione del problema proposto

Ci si pone il problema di ricevere sull'applicativo desktop tutti gli aggiornamenti e organizzare i dati in modo che vi si possa lavorare, facilitarne la ricerca e la visualizzazione.

1.1.2. Descrizione della rilevanza gestionale del problema

La creazione di applicativi che rendano user-friendly l'interazione con database e criteri di ricerca è parte integrante del lavoro di un gestionale dell'informazione.

La necessità di rendere trasparente la creazione e gestione di banche dati è un problema non solo più aziendale ma anche di piccoli gruppi di lavoro come un giornale o un blog.

1.1.3. Descrizione dei data-set per la valutazione

I dati che si andranno a realizzare sono: Articoli caratterizzati da Titolo, Tag, Autore, link al quale è pubblicato e data di pubblicazione.

Gli autori saranno caratterizzati da un campo di testo che ne riporta nome e cognome oppure un alias.

I tag saranno semplici stringhe che ne riportano il valore.

Le parole chiave dell'articolo saranno legate all'articolo tramite il link di pubblicazione dello stesso e saranno caratterizzate dalla parola di riferimento e da un valore numerico che ne riporta il peso calcolato in funzione del numero di volte in cui è ripetuta.

1.1.4. Descrizione preliminare degli algoritmi coinvolti

Il database di tutte le informazioni sarà in SQL, richiederà delle query per recuperarne il valore e salvarne di nuove.

L'applicativo desktop sarà sviluppati in Java e l'interfaccia grafica userà JavaFX.

Sarà necessario sviluppare due applicazioni differenti, una dedicata all'aggiornamento del database, che si collegherà periodicamente via RSS al blog. La seconda applicazione sarà il programma vero e proprio utilizzato dagli utenti e comunicherà esclusivamente con il database, utilizzerà il link dell'articolo per consentirne la lettura.

1.1.5. Descrizione preliminare delle funzionalità previste per l'applicazione software

L'applicativo server side, quello cioè predisposto all'aggiornamento del database, dovrà verificare la presenza di nuovi articoli e trovarne le parole chiave. Questa app prenderà il nome di "La Baionetta Updater".

Per il lato client l'utente potrà ricercare, anche in modo combinato, secondo parole chiave, autore, tag e data di pubblicazione; al momento della lettura gli verranno inoltre fornite maggiori informazioni su quanto sta leggendo e gli sarà data la possibilità di leggere altri articoli tra i cinque maggiormente correlati con quello scelto. L'app inoltre ricorderà gli articoli già letti ed evidenzierà quelli di cui la lettura non è ancora avvenuta. Questo secondo applicativo verrà denominato "La Baionetta Reader".

2. Capitolo 2

2.1. Descrizione dettagliata del problema

Ci si pone il problema di ricevere tutti gli aggiornamenti avvenuti all'interno del blog "La Baionetta" e organizzare i dati in modo che vi si possa lavorare, facilitarne la ricerca e la visualizzazione.

Si è quindi scelto di creare un sistema di applicazioni Java, l'immagine ideale di funzionamento posiziona le due applicazioni in luoghi diversi.

Vi è una prima applicazione che risiede sul server in cui vi è il database che si preoccupa di aggiornare in modo utile e coerente il database stesso e una secondo applicativo per permettere ad ogni client di connettersi al database e ricevere le informazioni che desidera.

Questo permette di tenere traccia in modo sicuro di quanto avviene sul blog, ricevendo in input i link dei nuovi articoli pubblicati sfruttando il servizio RSS; e offre all'utente le funzionalità richieste, restituendo le informazioni in modo organizzato.

Le criticità maggiori sono la non completa indipendenza dalla piattaforma di pubblicazione (in questo caso Blogger.com) e la sola possibilità di leggere gli articoli e non quella di pubblicarne di nuovi.

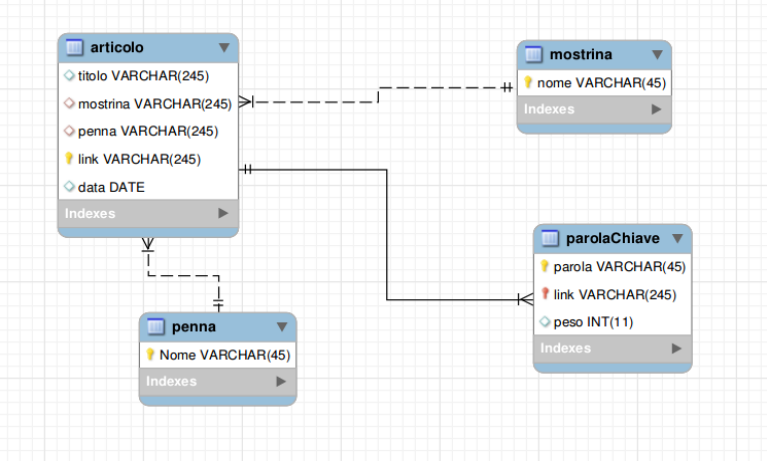
Specularmente, le più grandi potenzialità sono proprio la possibilità di evolvere facilmente quest'applicativo ad una piattaforma indipendente con possibilità di creare spazi appositi di pubblicazione e recuperare in modo pulito e veloce i testi.

3. Capitolo 3

3.1. Descrizione del data-set

Il database è stato costruito andando a leggere tutte le informazioni necessarie presenti pubblicamente sulla piattaforma Blogger all'indirizzo www.labaionetta.blogspot.it

Come si vede in figura il database ha una struttura formata da quattro tabelle.



La tabella articolo è composta da: titolo, mostrina, penna, link e data.

La tabella mostrina è caratterizzata dal nome.

La tabella penna è caratterizzata dal campo nome che rappresenta nome e cognome dell'autore o uno pseudonimo.

La tabella parolaChiave è formata dai campi parola, link e peso.

Per ogni articolo vi è una mostrina, una penna e una o più parole chiave. Per ogni mostrina e per ogni penna vi sono più articoli.

4. Capitolo 4

4.1. Descrizione ad alto livello delle strutture dati e degli algoritmi utilizzati

Tutte le informazioni necessarie sono salvate in liste, set e in un grafo.

All'avvio del programma viene creato il grafo, sfruttando la libreria jGraphT, pesato e non orientato; i vertici di questo grafo sono gli articoli e il peso dell'arco che collega tra di loro i vertici è dato dalla somma dei pesi delle parole simili tra i due articoli presi in considerazione.

Due parole sono ritenute simili solo in due casi: se sono di lunghezza diversa occorre che una sia contenuta nell'altra; se sono di lunghezza uguale è necessario che il numero di lettere che differiscono sia al più uguale a uno.

Una parola è ritenuta "parola chiave" nel momento in cui è ripetuta più di quattro volte all'interno del testo.

Le parole chiave vengono salvate nel database dall'app "La Baionetta Updater". Quest'ultimo, ogni ora controlla la presenza di nuovi articoli e ne recupera tutti i dati necessari; per facilitare la lettura del testo viene utilizzata la libreria Jsoup che consente di selezionare il testo contenuto all'interno del tag "post-body entry-content" della pagina html in cui è pubblicato l'articolo e di sanitizzarlo in modo trasparente. A questo elenco di parole così ottenute vengono tolti tutti i simboli, i caratteri speciali e la punteggiatura e mantenute solo quelle con una lunghezza superiore a tre caratteri.

Qualsiasi informazione venga richiesta al grafo questo la restituisce come lista di articoli che vengo ordinati per data, fa eccezione il caso di richiesta di articoli correlati che vengono ordinati in base al peso dell'arco che li collega con l'articolo di partenza.

5. Capitolo 5

5.1. Diagramma delle classi

5.1.1. La Baionetta Reader

L'applicazione chiamate "Reader" è la più "vicina" all'utente finale. In quanto è quella che utilizzerà in atto pratico.

È dotata di interfaccia grafica e quindi di una classe Controller (che qui non è riportata).

La parte logica del codice è tutta interna alla classe Model che ha quindi al suo interno dei Set dedicati alla memorizzazione di tutti gli oggetti Articolo, Penna e Mostrina che riceve dal DAO al momento dell'inizializzazione.

Il Model si occupa della creazione del grafo e le parole chiave degli articoli vengono chieste ai singoli articoli durante questa operazione.

Una ulteriore funzionalità che viene resa possibile dal Model è la registrazione dell'avvenuta lettura di un articolo, attraverso la classe LocalBackup viene creato un file (se questo non esiste), oppure aggiornato all'occorrenza, in cui viene progressivamente aggiunto l'attributo "link" dell'oggetto Articolo che viene letto, questo elenco viene interrogato ad ogni avvio del programma in modo da evidenziare, attraverso l'interfaccia grafica, quegli articoli di cui non è ancora avvenuta la lettura.

Per consentire la lettura dell'articolo si è optato per la creazione di un apposito oggetto Browser (di cui viene riportato il Controller nel diagramma delle classi).

L'oggetto BrowserController riceve dal Model l'oggetto Articolo che l'utente ha deciso di leggere attraverso l'interfaccia grafica, crea una nuova finestra contenente una WebView in cui viene caricato l'articolo utilizzando il suo link, e fornisce alcune informazioni sull'articolo stesso e i cinque articoli più fortemente correlati, questa operazione viene fatta dal Model ordinando in base al peso gli archi che collegano l'articolo in lettura con gli articoli vicini.

5.1.2. La Baionetta Updater

La Baionetta Updater è l'app server side, che opera sulla stessa macchina in cui è installato il database e si occupa di aggiornarlo.

Non vi è interfaccia grafica e quindi non vi è Controller.

I beans sono esattamente gli stessi già visti in precedenza.

All'interno del Main è presente un oggetto TimerTask che, ogni ora, chiede al Model di verificare la presenza di aggiornamenti.

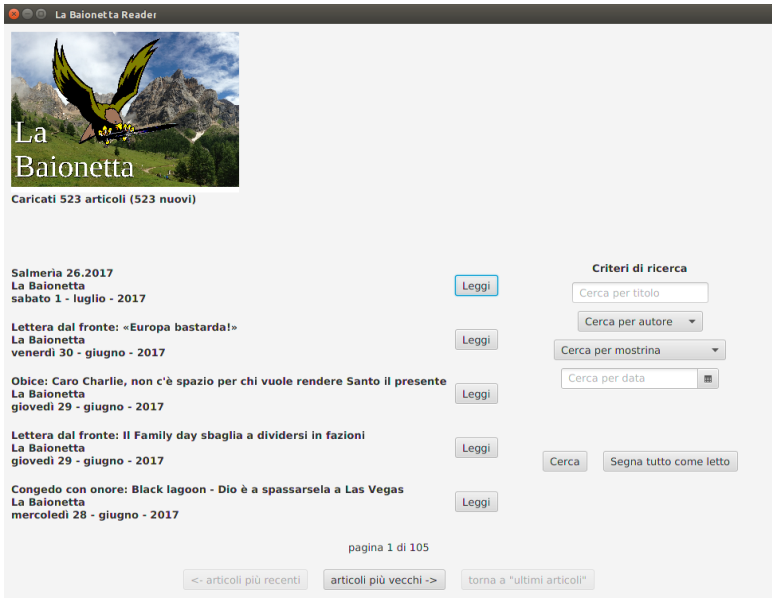
Il Model chiederà a sua volta alla classe ArticoloFeed, la quale si collega all'url in cui risiedono i feed RSS, la presenza di articoli e verifica se gli articoli creabili con le informazioni ottenute sono già presenti nel database o meno.

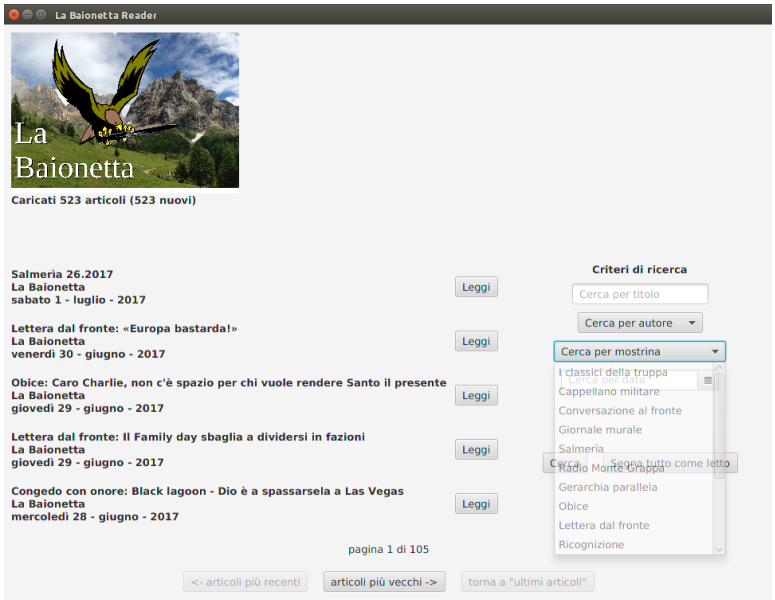
Se l'articolo presente all'interno del feed RSS non è presente nel database viene chiesto al DAO di aggiungerlo, questo esegue una query SQL di tipo "INSERT" e da questo momento in poi questo nuovo articolo è recuperabile dall'applicazione dell'utente.

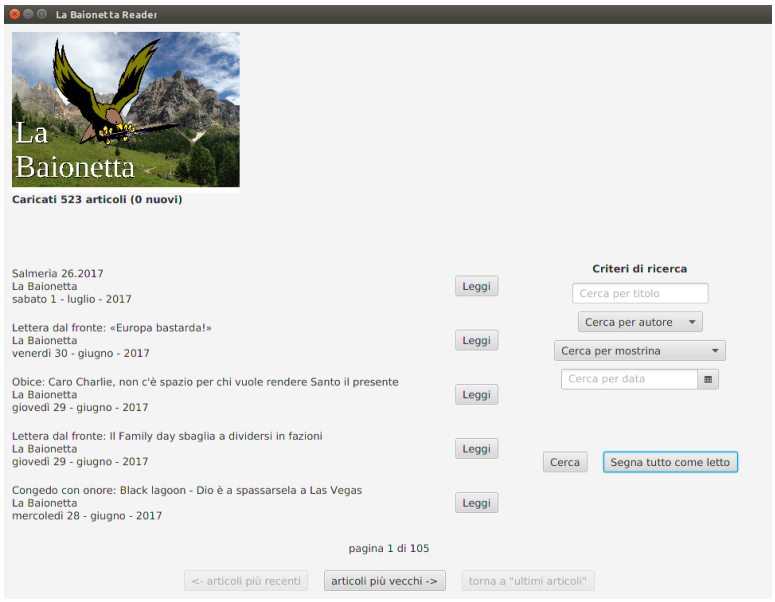
6. Capitolo 6

6.1. Videate dell'applicazione

6.1.1. La Baionetta Reader







Link al video dimostrativo: https://youtu.be/_3XGCIhe2JI

7. Capitolo 7

7.1. Valutazioni sui risultati ottenuti

In questo momento l'applicazione si trova a gestire 523 articoli, caratterizzati da 16 mostrine differenti, e 4 penne.

La parte più pesante è sicuramente quella dedicata alla creazione del grafo in quanto vi sono un totale di 8378 parole chiave da tenere in considerazione.

La parte di richiesta delle informazioni al database è stata, durante lo sviluppo, sempre migliorata in modo da ottenere la più grande quantità di informazioni possibili con il minimo numero di query. Questo lo si può apprezzare nella misurazione del tempo di avvio dell'applicazione che, come si vede nel video, è di circa 4 secondi.

Sicuramente con l'aumentare della quantità di articoli e di parole chiave presenti nel database si riscontreranno progressivi rallentamenti, per questo il proseguire dello sviluppo del software dovrà rendere sempre più precisa la scelta delle parole chiave e ottimizzata la creazione del grafo.

Quindi, relativamente ai problemi che ci si era prefigurati di risolvere al capitolo 2, si può affermare che l'applicativo risulta utilizzabile senza difficoltà anche in un uso normale e giornaliero, gli aggiornamenti vengono ricevuti e notificati in modo puntuale e gli articoli correlati sono presentati in modo chiaro e funzionale.