PROGETTAZIONE DI BASI DI DATI CON MODELLI DI NUOVA GENERAZIONE

Prof. Fabio Fumarola A.A. 2014/2015

**Lilliurlian**

**Documentazione**

Gruppo No14:

Arvizzigno Valerio

Russo Mirko 609392 ITPS russomirko28@gmail.com

Sommario

[**1. Storia e motivazione del contesto** 3](#_Toc434833983)

[**2. Analisi delle funzionalità** 4](#_Toc434833984)

[**3. Analisi dei costi delle operazioni** 7](#_Toc434833985)

[**4. Modello dei dati** 8](#_Toc434833986)

[**5. Scelta del database NoSQL** 10](#_Toc434833987)

[**6. Casi di test** 12](#_Toc434833988)

**1. Storia e motivazione del contesto**

L’URL Shortening è una tecnica utilizzata per abbreviare lunghi indirizzi web (URL) in indirizzi web aventi un numero molto contenuto di caratteri.

Infatti molti URL, tra domini e sotto-domini, sono eccessivamente lunghi ed è utile avere la possibilità di accorciarli per utilizzarli in servizi che limitano il numero di caratteri impiegabili (e.g. Twitter e alcuni servizi di Instant-Messaging).

Il primo servizio di URL Shortening è stato creato nel gennaio 2002 da Kevin Gilbertson.

Da allora la tecnica dell’URL Shortening si è evoluta fino ad avere la possibilità di implementare una serie di funzionalità molto interessanti, tra cui:

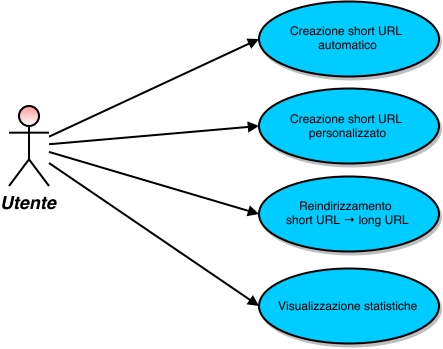
* personalizzazione dello short URL da creare
* tracciamento del numero di click effettuati su un dato short URL
* tracciamento di dati degli utenti, tra cui geolocalizzazione dell’indirizzo IP e browser utilizzato.

Tuttavia le caratteristiche degli short URL possono creare una serie di svantaggi agli utenti, tra cui:

* possibilità di essere reindirizzati a siti malevoli a causa dell’oscuramento dell’URL originale
* perdita della privacy.

Vi sono numerosi servizi web che consentono di abbreviare gli URL, tra cui annoveriamo bit.ly, goo.gl, ern.is, adf.ly e tinyurl.com.

**2. Analisi delle funzionalità**



**Scenari procedurali:**

**Creazione short URL automatico**

|  |  |
| --- | --- |
| **Descrizione** | L’utente vuole creare uno short URL per un long URL. |
| **Post-condizioni per successo** | Il sistema genera lo short URL e viene salvato nel database associato ai relativi long URL e statistiche. |
| **Post-condizioni per fallimento** | Il sistema non genera lo short URL e il database non viene modificato |
| **Evento innescante** | Generazione short URL |
| **Attore primario** | Utente |
| **Scenario di base** | 1. L’utente inserisce il long URL. 2. L’utente preme il pulsante “Generate!”. 3. Se il long URL è valido, il sistema genera lo short URL e lo visualizza sullo schermo. |
| **Scenario alternativo: long URL non valido** | 1. L’utente inserisce il long URL. 2. L’utente preme il pulsante “Generate!”. 3. Se il long URL non è valido, il sistema notifica all’utente la non validità del long URL immesso. |

**Creazione short URL personalizzato**

|  |  |
| --- | --- |
| **Descrizione** | L’utente vuole associare uno short URL personalizzato ad un long URL. |
| **Post-condizioni per successo** | Lo short URL personalizzato viene salvato nel database associato ai relativi long URL e statistiche. |
| **Post-condizioni per fallimento** | Il database non viene modificato |
| **Evento innescante** | Generazione short URL |
| **Attore primario** | Utente |
| **Scenario di base** | 1. L’utente inserisce il long URL e lo short URL personalizzato. 2. L’utente preme il pulsante “Generate!”. 3. Se il long URL e lo short URL personalizzato sono validi, il sistema genera lo short URL e lo visualizza sullo schermo. |
| **Scenario alternativo: long URL o short URL personalizzato non validi** | 1. L’utente inserisce il long URL e lo short URL personalizzato. 2. L’utente preme il pulsante “Generate!”. 3. Se il long URL o lo short URL personalizzato non sono validi il sistema notifica all’utente la non validità del long URL o dello short URL personalizzato immessi. |

**Reindirizzamento short URL → long URL**

|  |  |
| --- | --- |
| **Breve descrizione** | L’utente vuole aprire la pagina web avente il long URL associato ad uno short URL |
| **Post-condizioni per successo** | L’utente viene reindirizzato correttamente alla pagina web |
| **Post-condizioni per fallimento** | L’utente viene reindirizzato ad una pagina di errore |
| **Evento innescante** | Apertura della pagina corrispondente ad uno short URL. |
| **Attore primario** | Utente |
| **Scenario base** | 1. L’utente inserisce lo short URL nella barra degli indirizzi. 2. L’utente preme il pulsante “vai all’URL” del browser in uso. |

**Visualizzazione statistiche**

|  |  |
| --- | --- |
| **Breve descrizione** | L’utente vuole vedere le statistiche relative allo short URL immesso. |
| **Post-condizioni per successo** | L’utente visualizza le statistiche dello short URL immesso |
| **Post-condizioni per fallimento** | Viene generato un messaggio di errore che indica che lo short URL immesso non esiste |
| **Evento innescante** | Visualizzazione delle statistiche relative allo short URL |
| **Attore primario** | Utente |
| **Scenario di base** | 1. L’utente preme il tasto “Statistic” nella barra di navigazione. 2. L’utente inserisce lo short URL nell’apposita sezione. 3. L’utente preme il pulsante “Get Statistics!”. 4. Il sistema mostra all’utente le statistiche dello short URL. |

**3. Analisi dei costi delle operazioni**

Di seguito sono elencate le operazioni effettuate sul database e i relativi costi:

**Creazione short URL automatico:**

Il sistema controlla nel database se esiste già uno short URL associato al long URL immesso (1 lettura). In caso affermativo restituisce all’utente lo short URL già presente, mentre in caso negativo viene generato un nuovo short URL e, insieme alle relative informazioni, viene salvato nel database (1 scrittura).

**Creazione short URL personalizzato:**

Richiede gli stessi costi dello short URL automatico.

**Reindirizzamento short URL → long URL:**

Il sistema controlla nel database se esiste il long URL associato allo short URL immesso (1 lettura). In caso affermativo reindirizza l’utente alla pagina web associata al long URL e aggiorna le informazioni salvate nel database relative allo short URL (1 scrittura).

**Visualizzazione statistiche:**

Il sistema controlla nel database se esiste lo short URL immesso (1 lettura) in caso affermativo restituisce le statistiche dello short URL.

**4. Modello dei dati**

Dato che ad ogni short URL esistente sono associati dati eterogenei (URL originario, il numero di click ricevuti dallo short URL, gli indirizzi IP degli utenti che hanno cliccato sullo short URL, il nome del browser utilizzato), si è scelto di aggregarli sotto forma di documento secondo il modello Document-Oriented.

La struttura del modello dei dati è la seguente:

document{

“shortUrl”: String,

“longUrl”: String,

“isCustom”: Boolean,

“createdOn”: Date,

“totalClicks”: Integer,

“browser”:{

“Chrome”: Integer,

“Firefox”: Integer,

“Opera”: Integer,

…

},

“OS”:{

“Windows”: Integer,

“CentOS”: Integer,

“OS X”: Integer,

…

},

“country”:{

“Italy”: Integer,

“Germany”: Integer,

“France”: Integer,

…

},

}

Il campo “shortUrl” rappresenta l’URL che reindirizza all’URL originario (rappresentato da “longUrl”) e che viene generato dal sistema o scelto dall’utente.

Il campo “isCustom” determina se lo short URL è stato scelto dall’utente o meno.

Il campo “createOn” rappresenta la data di creazione dello short URL.

Il campo “totalClicks” rappresenta il numero dei click totali ricevuti dallo short URL.

Il documento “browser” include il numero dei click effettuati sullo short URL dagli utenti che utilizzano un determinato Web browser.

Il documento “OS” include il numero dei click effettuati sullo short URL dagli utenti che utilizzano un determinato Sistema Operativo

Il documento “country” include il numero dei effettuati sullo short URL dagli utenti aventi un indirizzo IP associato ad una certa nazione.

**5. Scelta del database NoSQL**

Il gruppo ha scelto il database NoSQL Document-Oriented MongoDB.

MongoDB (da "humongous", enorme) è un DBMS non relazionale, orientato ai documenti.

MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di documenti in stile JSON con schema dinamico (MongoDB chiama il formato BSON), rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce.

Lo sviluppo di MongoDB è iniziato nel 2007 dall'azienda (allora chiamata) 10gen. Nel 2009, MongoDB è stato reso open source come prodotto indipendente con una licenza AGPL.

Rilasciato sotto una combinazione della GNU Affero General Public License e dell'Apache License, MongoDB è un software libero e open source

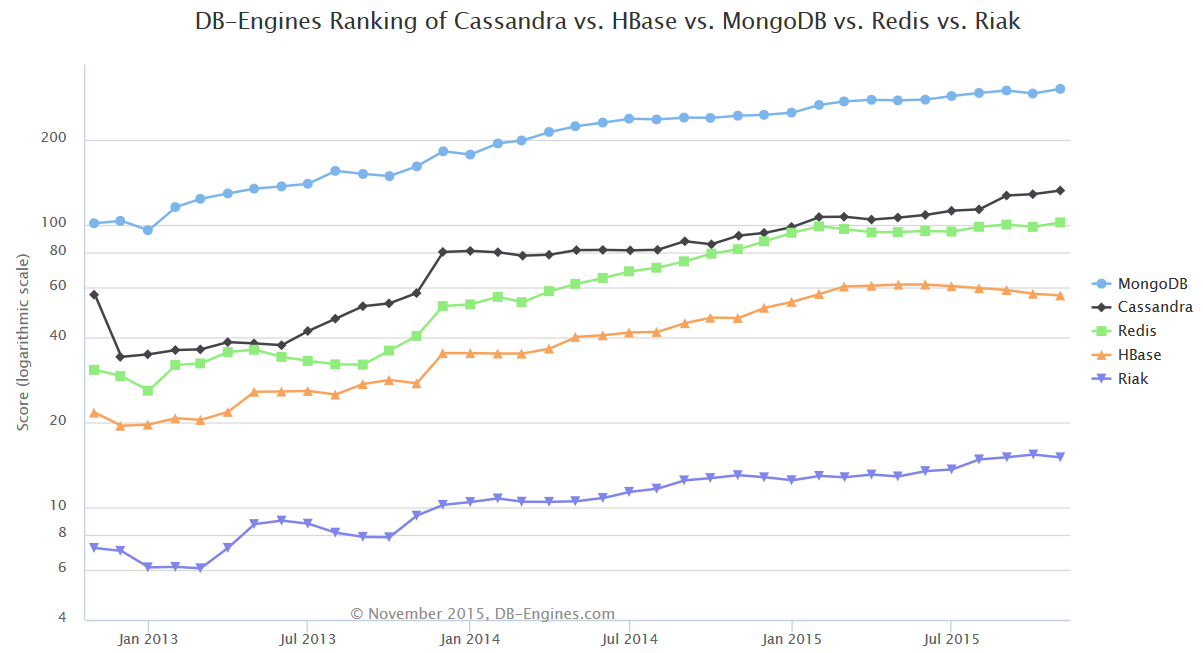
(*fonte: “Wikipedia”*).

MongoDB possiede numerose caratteristiche, e il gruppo lo ha scelto per il progetto Lilliurlian principalmente per le seguenti:

* MongoDB supporta ricerche per campi, intervalli e regular expression.

Le query possono restituire interi documenti o campi specifici dei singoli documenti.

* MongoDB scala orizzontalmente usando l’auto-sharding, ovvero distribuisce i dati su più server in maniera automatizzata, lasciando che sia il motore di MongoDB a gestire l’interrogazione e la scrittura dei dati su più macchine.
* MongoDB fornisce alta disponibilità e aumento del carico gestito attraverso i replica set (un server master e vari server slave) e attraverso la gestione automatica dei fail-over (elezione automatica di un nuovo server master).

**DB-Engines Ranking: MongoDB vs. Cassandra vs. Redis vs. HBase vs. Riak**

**6. Test dei metodi**

I seguenti metodi delle rispettive classi sono stati testati usando il framework JUnit:

**Package** lilliurlian.utility

**Classe** BadContentChecker

**Metodo** isBlank () : Se nella stringa in input è inclusa una sequenza di caratteri proibita restituisce **true**, altrimenti **false**.

**Classe** EmptyStringChecker

**Metodo** check() : Se la stringa in input equivale ad una stringa non inizializzata, o di lunghezza nulla o piena esclusivamente di spazi bianchi restituisce **true**, altrimenti **false**.

**Classe** ShortStringGenerator

**Metodo** nextString() : Restituisce una stringa generata automaticamente dal sistema.

**Classe** SpecialCharFinder

**Metodo** isFound() : Se nella stringa in input è incluso un carattere speciale restituisce **true**, altrimenti **false**.

**Classe** IPGeoloc

**Metodo** getCountryIso () : Restituisce una stringa contenente il codice ISO della nazione a cui è associato l’indirizzo IP immesso come parametro in input.

**Metodo** getCountryName () : Restituisce una stringa contenente il nome della nazione a cui è associato l’indirizzo IP immesso come parametro in input.