Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica per il Management

Big Data, NoSQL e Machine Learning: un'applicazione di prediction e recommendation basata sulle API di Amazon

Relatore:
Chiar.mo Prof.
MARCO DI FELICE

Presentata da:
MATTIA BALLO

Sessione II
Anno Accademico 2014/2015

Introduzione

A causa dell'enorme quantità di dati, che viene generata in tempi sempre più brevi, è stata sviluppata una nuova tipologia di database definiti non relazionali. Essi nascono al fine di rappresentare dei dati, definiti *big data*, proprio a causa delle caratteristiche prima citate. Essi, come dice il nome, possiedono uno schema logico, differente dagli approcci canonici, atto alla rappresentazione e all'analisi di questi particolari dati. Oltre a nuovi sistemi di memorizzazione, sono nati anche sistemi per l'analisi automatica dei dati. Questi sistemi, definiti sistemi di *machine learning*, attuano delle tecniche di apprendimento automatico per imparare ad analizzare i dati in maniera autonoma attraverso il modello che meglio vi si adatta.

Lo scopo di questa tesi è approfondire uno specifico ambiente per quanto riguarda il mondo *NoSQL* e uno per quanto riguarda il mondo del *Machine Learning* e successivamente utilizzare i sistemi descritti, insieme ad altre tecnologie innovative, per sviluppare, in tempi contenuti, un'applicazione di *prediction* e *recommendation*. Dimostrando così che queste tecnologie di gestione dei dati, oltre che essere molto potenti e performanti, sono anche molto intuitive e di semplice utilizzo.

Durante la prima parte verranno analizzate le esigenze che hanno portato allo sviluppo di questi sistemi, affrontando le caratteristiche principali e le principali fonti che generano questo tipo di dati, prendendo anche in considerazione gli attuali utilizzi da parte di aziende di forte rilevanza in ambito informatico. Mostrando i vari approcci presenti oggi, le loro caratteristiche

e la loro struttura.
Verranno poi analizzati gli ambienti di MongoDB e PredictionIO, approfondendo la struttura e le modalità di utilizzo di questa categoria di sistemi.

Nella seconda parte verrà invece descritta *Item Price Watcher*, un'applicazione di *prediction* e *recommendation* sviluppata utilizzando le più moderne tecnologie web e di virtualizzazione unite all'utilizzo dei sistemi approfonditi nella prima parte. In questo modo si potrà constatare i vari passi da seguire per l'implementazione di un sistema reale utilizzato per la registrazione di prodotti ricavati da varie sorgenti in maniera automatica(attraverso le API Amazon) e manuale. Questi prodotti vengono poi analizzati dal sistema di *Machine Learning*, per il suggerimento di prodotti simili acquistati da altri utenti, e attraverso una regressione polinomiale viene effettuata una previsione sul prezzo futuro in base allo storico.
Al fine di analizzare i risultati prodotti verrà poi analizzato l'errore commesso dall'interpolazione, per verificare l'adattamento del modello ad uno scenario di questo tipo.

Indice

Elenco delle figure

Elenco delle tabelle

Parte I

Stato dell'Arte e Metodologie

Capitolo 1

Big Data, NoSQL e Machine Learning

In questo capitolo verranno introdotti i *Big Data*, i database *NoSQL*, in particolare le loro caratteristiche e la loro storia nel mondo delle basi di dati, e i sistemi di *Machine Learning*, utilizzati in concomitanza con questi sistemi per effettuare calcoli avanzati.

1.1 Introduzione ai Big Data

L'evoluzione tecnologica, nel corso degli ultimi anni, ha portato ad un notevole incremento nella mole di dati generati dai singoli dispositivi, siano essi industriali o facenti parte dell'elettronica di consumo. La varietà della natura e della struttura di questi dati, insieme alla mole e alla velocità di produzione, hanno introdotto un problema fondamentale per quanto riguarda la memorizzazione e l'analisi di questi.
Anche il *Web 2.0* da diversi anni è una fonte sempre crescente di dati. I più importanti sono i cosiddetti *user-generated content*, ovvero i dati prodotti principalmente dagli utenti, come foto, video, post, blog e dati di altra natura. Con la diffusione dei social network, la mole di contenuti generati dagli utenti ha visto una crescita esponenziale. per questo negli ultimi anni ha visto

prendere piede il termine *big data* che rappresenta dati prodotti in grandi quantità e in tempi rapidi, la cui elaborazione richiede tecnologie e risorse differenti da quelle utilizzate dai sistemi di memorizzazione convenzionali.

1.1.1 Caratteristiche distintive

La raccolta e l'analisi di questi dati, per scopi di marketing e di management, è oggi una delle attività su cui le aziende del settore investono maggiormente. L'insieme dei processi aziendali, delle tecnologie volte al raggiungimento di tale scopo e delle informazioni ricavate da esso è definito con la locuzione *Business Inteligence* [1]. Tuttavia, l'utilizzo di sistemi convenzionali non è adatto alla rappresentazione e all'analisi dei big data a causa delle loro caratteristiche:

Volume Come detto in precedenza, e come si evince dal nome, uno delle caratteristiche dei big data, è la mole di dati che immagazzina. Dati generati dall'utente attraverso l'utilizzo di piattaforme del Web 2.0 oppure dati generati automaticamente da macchine industriali (sensori, *Distributed Control System*[1], strumenti scentifici), transazioni bancarie o sui mercati finanziari possono raggiungere volumi mastodontici. La *Aureus Analitycs* stima che nel 2020 l'insieme di tutti i dati in formato digitale sarà pari a 40 zettabyte, circa 5,2 exabyte per ogni uomo, donna o bambino presente sulla terra [2].

Variety La diversità dei formati, o la mancanza di una struttura che possa essere rappresentata attraverso una tabella in un database relazionale, è la seconda caratteristica dei big data. Esistono infatti vari dati non strutturati o semistrutturati come documenti di testo(TXT, CSV, PDF, Word, Excel, etc.), post dei blog, commenti su social network o sulle piattaforme di microblogging. Oltre alla varietà di strutture e for-

[1]Un *Distributed Control System* è un sistema di controllo automatico costituito da diversi sottosistemi, tra cui quello di acquisizione e di elaborazione dei dati. `https://it.wikipedia.org/wiki/Sistema_di_controllo_distribuito`

mati, è presente anche la varietà delle fonti: alcuni dati possono essere generati in maniera automatica (sensori, log di server, etc.) mentre altri da utenti (user-generated content).
Per il salvataggio di questi dati spesso si ricorre a soluzioni *NoSQL*, che verranno introdotti nella sezione successiva, in quanto non impongono una rigidità dello schema logico a differenza delle soluzioni relazionali.

Velocity Oltre alla mole di dati, anche la velocità con cui vengono generati è un'altra caratteristica fondamentale dei big data. Pensando ai già citati sensori, occorre utilizzare strumenti che siano in grado di tenere il passo con queste fonti di dati. Per le aziende un'altra sfida posta dai dati generati ad alta velocità è la capacità di analizzarli in tempi altrettanto rapidi. L'esigenza di ottenere tempi rapidi di risposta ha portato allo sviluppo di database non relazionali come i database *column-oriented* o i *key/value store*, che verranno approfonditi in seguito. [3]

1.1.2 Rischi e criticità

I big data, come ogni cosa, non presentano solo aspetti positivi. Infatti possiamo analizzare due aspetti critici fondamentali, che potrebbero, in qualche modo, vanificare i vantaggi sopra descritti. Questi aspetti sono la qualità dei dati e la proprietà di essi.

Qualità

La qualità dei dati è determinata da un insieme di caratteristiche:

- *Accuratezza*: conformità dei dati ai valori reali.
- *Completezza*: presenza di tutti i dati necessari a descrivere un'entità, una transazione o un evento.
- *Consistenza*: assenza di contraddizioni nei dati.
- *Assenza di duplicazione*: campi, record o tabelle devono essere presenti solo una volta, evitando duplicazioni.

- *Integrità*: Questo termine è di solito utilizzato in riferimento ai database relazionali, dove attraverso alcuni strumenti come tipi di dato, *check constraint*[2], chiavi primarie e chiavi esterne si è in grado di far rispettare determinati vincoli ai dati.

La qualità dei dati deve essere controllata e verificata e il processo di *data quality* deve indicare quali sono i dati con livelli di integrità, completezza, consistenza e accuratezza ritenuti accettabili e quali devono essere migliorati. Per quanto riguarda i big data si possono distinguere tre tipi di dati, in base ai concetti descritti prima:

- **Dati provenienti dai sistemi operazionali**: si tratta di dati generati da sistemi operazionali che producono una grande quantità di dati (finanza, grande distribuzione, etc.).

- **Dati provenienti da sensori, RFID[3] e strumenti scientifici**: essendo generati da macchina non producono errori di immissione, ma tuttavia possono presentare degli errori dovuti ai difetti dei sensori o degli strumenti di misura.

- **Dati provenienti dal Web**: questi dati si presentano in formato semistrutturato: i metadati[4], che costituiscono la parte strutturata, sono più affidabili e completi, mentre il testo è spesso soggetto a errori ed imprecisioni.

Un'altra questione importante è quella della categorizzazione dell'informazione. Non è, infatti, sempre possibile distinguere il significato o il contesto di una parola. Inoltre le fonti Web potrebbero presentare problemi di veridicità.

[2]Un *check constraint* rappresenta un vincolo di integrità in SQL che deve essere rispettato da tutte le righe di una tabella. `https://en.wikipedia.org/wiki/Check_constraint`

[3]Radio-frequency identification. `https://en.wikipedia.org/wiki/Radio-frequency_identification`

[4]Un metadato è un'informazione che descrive un insieme di dati. `https://it.wikipedia.org/wiki/Metadato`

Questi problemi, tuttavia, non sono di importanza fondamentale per analisi che non richiedono l'esattezza e la precisione dei dati come la *sentiment analysis* (conosciuta anche come *opinion mining*, essa si riferisce all'utilizzo di sistemi di analisi del testo, di computazione del linguaggio naturale e di computazione linguistica per identificare il senso soggettivo di un determinato dato).

1.1.3 Immagazzinamento dei dati

Come detto in precedenza, l'immagazzinamento dei big data pone due problemi difficilmente affrontabili con le tecnologie tradizionali: mole di dati e presenza di dati semistrutturati o non strutturati.
Per far fronte a queste problematiche molte aziende hanno puntato alla creazione di sistemi di immagazzinamento dati di varia natura, ma che hanno in comune la corretta gestione dei big data. Questa categoria di basi di dati è chiamata *NoSQL*(*Not only Structured Query Language*).
Essi attraverso l'utilizzo di algoritmi avanzati e sistemi di *Cloud Computing*, sono in grado di processare grandi quantità di dati in tempi molto minori rispetto ai sistemi di database relazionali. Nella prossima sezione verrà introdotto questo ambiente e verranno elencati i vari approcci che sono presenti oggi.

1.2 Introduzione ai database NoSQL

I database NoSQL si possono definire come una classe di database che non aderiscono al modello relazionale delle basi di dati. Essi infatti non sono costruiti utilizzando uno schema (*schemaless*) e per questo motivo, generalmente, non utilizzano il linguaggio SQL per la manipolazione dei dati [3].
Nonostante inizialmente il termine fosse utilizzato per indicare una netta scissione dal modello relazionale, oggi questo termine viene utilizzato per indicare il fatto che esistono diversi casi d'uso per i quali il modello relazionale rappresenta una forzatura, ma tanti altri per i quali tale modello è ancora la

soluzione migliore [4].
La nascita di questa tipologia di database è da riscontrarsi negli anni '60, con alcuni software come *MultiValue*(TRW, 1965) e *IBM IMS*(sviluppato per il programma spaziale *Apollo*, 1966). Tuttavia il nome NoSQL, con l'attuale siginificato, è comparso solo nel 2009 quando *Eric Evans* lo utilizzò per definire la branchia dei database non relazionali [5].
Come si evince dalla figura 1.1, oggi questo tipo di database rappresenta una grande fetta di mercato del mondo delle basi di dati.

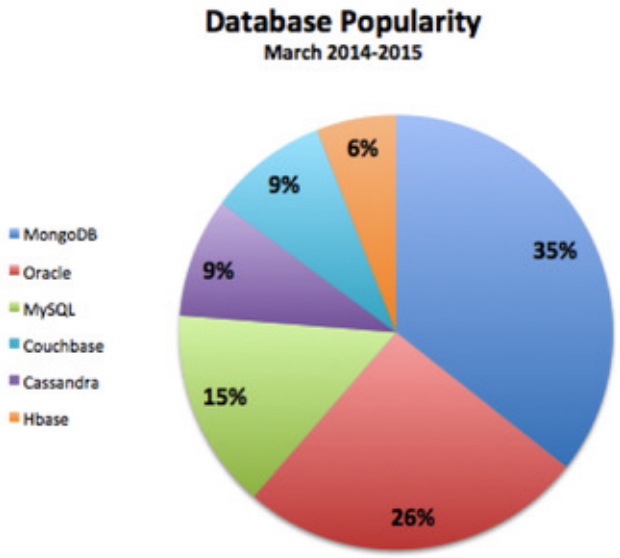


Figura 1.1: NoSQL databases eat into the relational database market [6]

1.2.1 Caratteristiche principali

A differenza deigli *RDBMS*(Relational DataBase Management System), che seguono i principi delle transazioni dette *ACID*(Atomicity, Consistency, Isolation Durability), i database NoSQL seguono un principio detto *BASE*, definito come segue:

Basic Availability ad ogni richiesta vi è la garanzia di una risposta, sia che l'esecuzione sia andata a buon fine, sia in caso contrario.

Soft state lo stato del sistema può cambiare nel tempo, anche senza la presenza di input, per raggiungere la consistenza dei dati.

Eventual consistency il database si può trovare momentaneamente in uno stato di inconsistenza, tuttavia alla fine manterrà la consistenza.

Queste proprietà sono dovute principalmente al fatto che questa tipologia di database è nata per essere veloce, flessibile e distribuita. Infatti, secondo il *Teorema di CAP* [5], è impossibile per un sistema informatico distribuito garantire contemporaneamente *Consistency* (tutti i nodi vedono gli stessi dati nello stesso momento), *Availability* (la garanzia che ogni richiesta riceva una risposta su ciò che sia riuscito o fallito) e *Partition tolerance* (il sistema continua a funzionare nonostante arbitrarie perdite di messaggi).
Grazie a questo principio essi godono di alcune caratteristiche che li differenziano dai RDBMS [3], quali:

- **Assenza di schema**: quasi tutte le implementazioni di NoSQL permettono una rappresentazione dei dati senza uno schema, semplificando l'evoluzione dello schema logico tramite l'aggiunta di campi o l'inserimento di documenti annidati.
- **Tempo di sviluppo**: la velocità è favorita dal fatto di non dover creare query complesse in SQL.
- **Velocità**: anche con una bassa mole di dati, i tempi di risposta di questi sistemi sono nell'ordine dei millisecondi a differenza delle centinaia di millisecondi relativi ai sistemi canonici.
- **Scalabilità**: l'aggiunta di nuove risorse mostra un miglioramento delle prestazioni in proporzione alla mole di dati.

[5] Eric Brewer, Principles of Distributed Computing (PODC), University of California, Berkley, 2000

1.2.2 Tipologie di salvataggio

I database NoSQL sono suddivisi in base allo schema utilizzato per la rappresentazione. Vista la necessità di gestione di grandi moli di dati, spesso in tempo reale, essi sono principalmente sviluppati attraverso una struttura orizzontale, ottimizzando quindi inserimento e recupero dei dati in grandi realtà.
Le tipologie più diffuse sono quattro:

Column-oriented Questo tipo di database salva i dati su colonne, all'opposto degli RDBMS che salva su righe come mostra la Figura 1.2.

I sistemi di **Online Analytical Processing(OLAP)** che necessita-

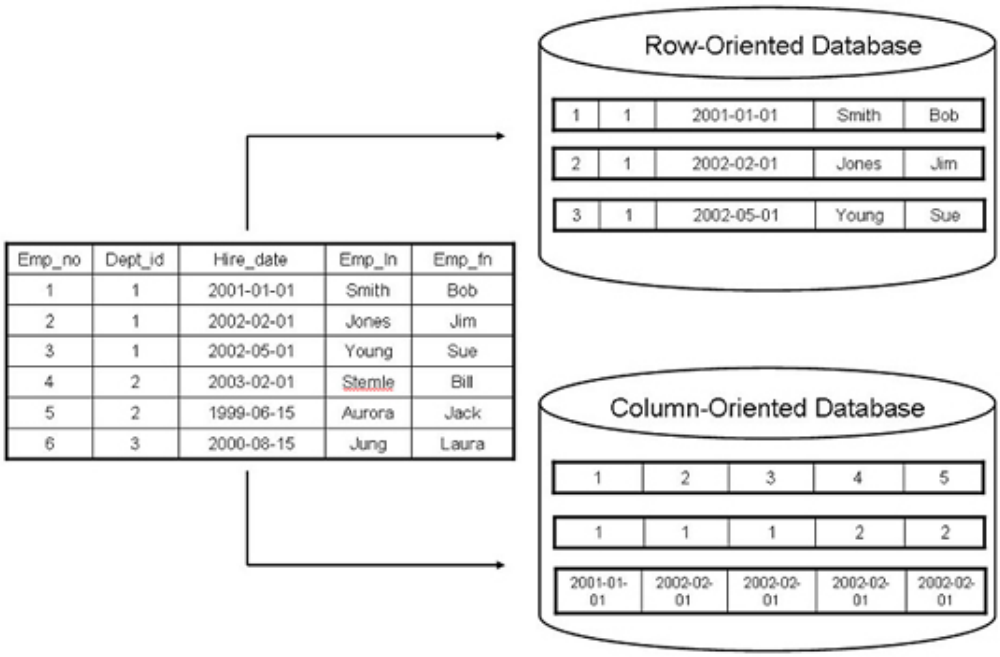


Figura 1.2: Column and Row Based Database Storage [7]

no di processare i dati, hanno bisogno di un accesso column-oriented. Normalmente l'accesso a questi database viene effettuato attraverso protocolli proprietari, per quanto riguarda soluzioni commerciali, op-

pure attraverso standard liberi(es. *Remote Method Invocation*[6]).
La Tabella 1.1 mostra alcuni database che utilizzano questa struttura. Molte soluzioni con questo approccio consentono l'aggiunta di una

Column-oriented
Oracle RDBMS Columnar Expression
Microsft SQL Server 2012 Enterprise Edition
Apache Cassandra
HBase
Google BigTable

Tabella 1.1: column-oriented database

nuova colonna, senza preoccuparsi di riempire i valori di default per le righe già presenti. Questo da una grande elasticità nella modellazione, in quanto cambiamenti futuri non risultano difficoltosi da implementare.
Ci sono vantaggi, inoltre, per le operazioni di aggregazione come massimo, minimo, media e somma, soprattutto nei dataset di grandi dimensioni. Questo avviene poichè il database consente un accesso parziale ai dati senza toccare le colonne che non servono ai fini dell'operazione. Siccome esse sono tutte uniformi ed ogni riga ha la stessa lunghezza, eccetto in casi particolari, si può introdurre un'efficiente gestione della memoria secondaria. Per esempio comprimendo due valori uguali adiacenti. [3]

Document Store Conosciuti anche come *document-oriented*, questi database consento l'inserimento, il recupero e la manipolazione di dati semistrutturati. Molti database di questa categoria utilizzano formati di

[6]La *Remote Method Invocation* (invocazione remota di metodi) o *RMI* è una tecnologia che consente a processi Java distribuiti di comunicare attraverso una rete. `https://it.wikipedia.org/wiki/Remote_Method_Invocation`

rappresentazione dei dati quali XML, JSON, BSON e YAML con l'accesso ai dati effettuato tramite *API RESTful*[7] che utilizzano il protocollo *Apache Thrift*[8] per mantenere l'interoperabilità cross-language.
Come si nota dalla Figura 1.3, ogni documento, chiamato record, rappresenta una riga del database. Ogni record può avere una struttura completamente differente con un insieme di campi o colonne variabile. Per questo motivo il database potrebbe non supportare uno schema, o comunque validare un documento in base ad uno schema nella sua totalità. Ciò nonostante è possibile la creazione e l'interrogazione di indici. La Tabella 1.2 mostra alcuni database che utilizzano questa

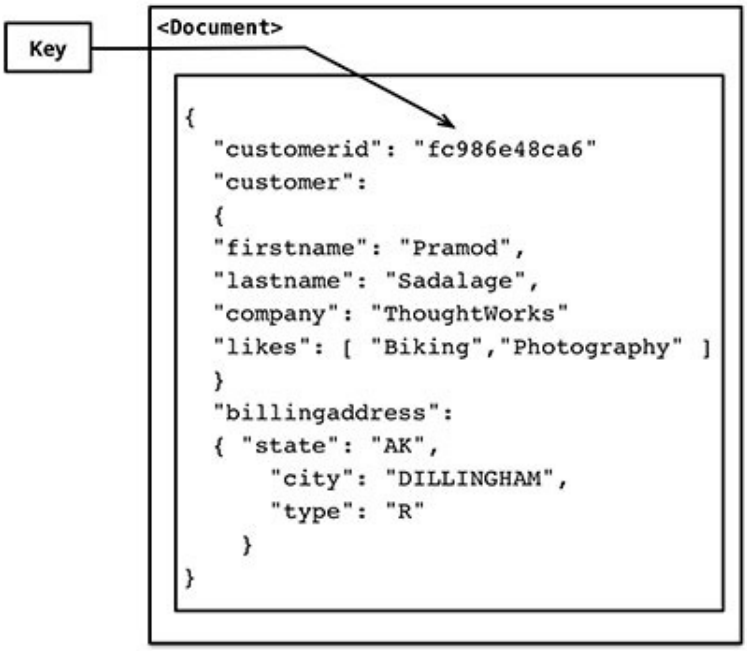


Figura 1.3: Document-Oriented database [8]

[7]Le API RESTful sono basate sull'architettura REST e utilizzano il protocollo HTTP per recuperare, creare, cancellare e aggiornare le risorse di un sistema. https://it.wikipedia.org/wiki/Representational_State_Transfer

[8]Apache Thrift è un protocollo di comunicazione binario utilizzato come framework RPC(*Remote Procedure Call*). https://it.wikipedia.org/wiki/Apache_Thrift

struttura, divisi per formato di rappresentazione utilizzato.

Il vantaggio più grande di questo tipo di approccio è che il conte-

Document store	
JSON-based	**XML-based**
MongoDB	BaseX
CouchDB	
Lotus Notes	
Apache Cassandra	
Redis	

Tabella 1.2: document store database

nuto è salvato senza uno schema. Per questo possono essere salvati più dati con formati differenti e la struttura può cambiare nel tempo. Potrebbe inoltre essere possibile recuperare e aggiornare parzialmente un documento, in particolare per quanto riguarda i formati basati su XML[9] che possono integrare *XQuery 3.0*[10]. La ricerca, in questo tipo di database, può essere meno efficiente di quella nei database di tipo *column-oriented*, tuttavia essi permettono la creazione di indici, utilizzando un parametro presente in tutti i record.

Key-Value Store L'approccio *key-value* è molto simile al document store, con la differenza che consente la memorizzazione di un documento in relazione ad una chiave, che quindi sarà esterna al documento stesso, come mostrato dalla Figura 1.4. Anche in questo caso non vi è la necessità di uno schema per la memorizzazione dei dati, tuttavia sono

[9]XML (*eXtensible Markup Language*) è un linguaggio marcatore basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento. `https://it.wikipedia.org/wiki/XML`

[10]XQuery è un linguaggio d'interrogazione e di programmazzione funzionale utilizzato per interrogare e trasformare delle collezioni di dati strutturati e non strutturati rappresentati in formato XML. `https://it.wikipedia.org/wiki/XQuery`

presenti alcuni vincoli che lo differenziano dall'approccio descritto in precedenza:

- La chiave deve essere specificata a priori. Non può quindi essere creata dal sistema in maniera automatica in fase di inserimento.
- Il valore del documento è "opaco", quindi non possono essere creati e interrogati indici su di esso. Per recuperare un documento è necessario conoscere il valore della chiave.

7b976c48...	name: Bill Watterson	state: DC	birth_date: 1953
7c8f33e2...	name: Howard Tayler	state: UT	birth_date: 1968
7d2a3630...	name: Randall Monroe	state: PA	
7da30d76...	name: Dave Kellett	state: CA	

Figura 1.4: Key-Value store [9]

Un esempio di memorizzazione key-value sono le mappe, gli array associativi[11] o le tabelle hash[12]. L'utilizzo principale di queste strutture è quello degli *in-memory database*[13], tuttavia esso funziona anche con database persistenti.
La Tabella 1.3 mostra alcuni database che utilizzano questo approccio: Questo tipo di approccio è ottimizzato per le query sulle chiavi. Siccome sono stati pensati per funzionare in memoria primaria, è stato implementato un meccanismo di scadenza delle chiavi, a scorrimento

[11]L'array associativo è un array i cui elementi sono accessibili mediante nomi anziché indici puramente numerici. `https://it.wikipedia.org/wiki/Array_associativo`

[12]In informatica una hash table è una struttura dati usata per mettere in corrispondenza una data chiave con un dato valore. `https://it.wikipedia.org/wiki/Hash_table`

[13]Per *in-memory database*(IMDB) si intende un DBMS che gestisce i dati nella memoria centrale. `https://it.wikipedia.org/wiki/In-memory_database`

Key-Value store
Redis
Memcached
MemcacheDB
Berkley DB
Voldemort

Tabella 1.3: key-value store database

o assoluto, dopo il quale il documento viene rimosso. Le chiavi possono essere generate in maniera inteligente, per esempio seguendo un pattern, e successivamente è possibile recuperare un sottoinsieme delle chiavi che corrisponde al pattern fornito, come nel caso di Redis.
Occorre tuttavia notare che la complessità della scansione ($O(N)$) è molto superiore rispetto alla complessità del recupero data la chiave ($O(1)$). Facendo un esempio, Redis, in funzione su un laptop di fascia entry-level, è in grado di scansionare un database contenente 1 milione di chiavi in circa 40 millisecondi. [10]
Nonostante non sia possibile effettuare query sui valori dei documenti, in questo approccio si possono referenziare i tipi associati ad ogni chiave. In base al valore è possibile fornire funzionalità avanzate come incremento atomico, aggiornamento/inserimento di campi multipli, intersezioni, unioni e differenze a livello insiemistico. [3]

Graph I database Graph rappresentano una categoria particolare della famiglia NoSQL dove le relazioni sono rappresentate da un grafo[14]. Vi possono essere collegamenti multipli fra due elementi del grafo, a indicare la relazione multipla fra essi.
Le relazioni rappresentate possono essere relazioni sociali, collegamenti fra due zone geografiche o topologia della rete fra due sistemi fra loro

[14]Un grafo è un insieme di elementi detti nodi o vertici che possono essere collegati fra loro da linee chiamate archi o lati o spigoli. `https://it.wikipedia.org/wiki/Grafo`

connessi. Un esempio di grafo è quello rappresentato dalla Figura 1.5. Essendo una tecnologia nuova in relazione al mondo NoSQL, non sono

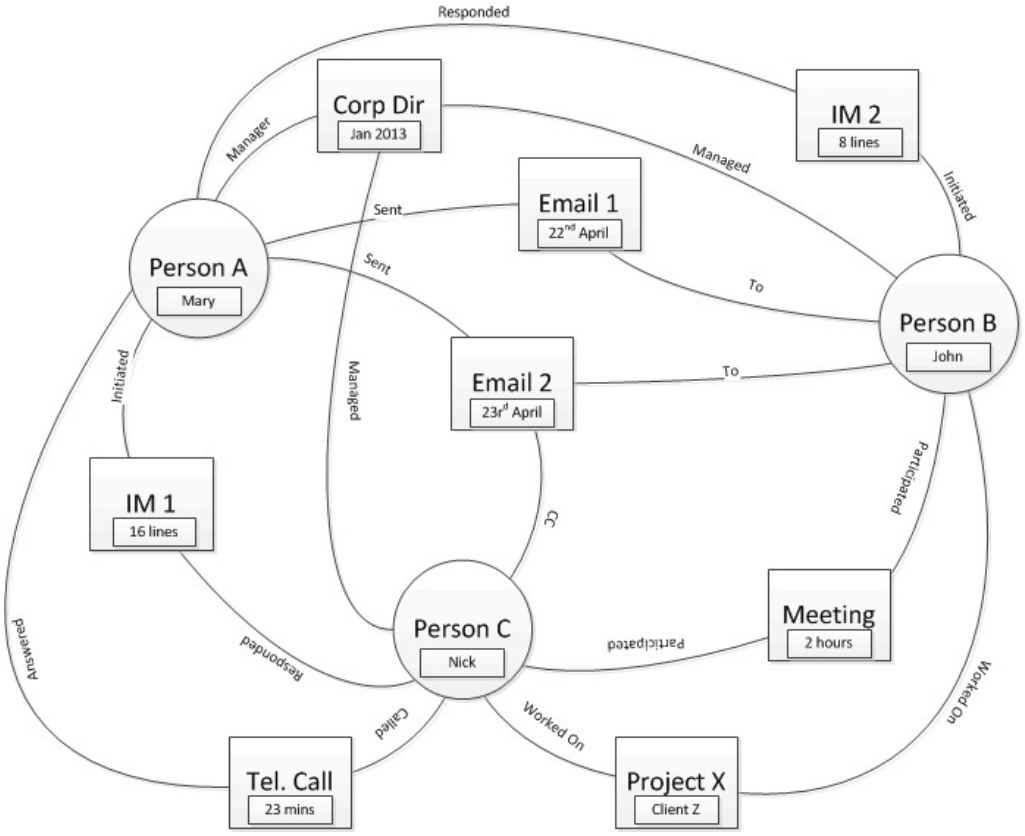


Figura 1.5: Graph store [11]

molti i progetti che utilizzano questa tecnologia. Nella Tabella 1.4 si trovano i più conosciuti.

I vantaggi di queste rappresentazioni si vedono in diversi ambiti: dal calcolo del percorso alle indicazioni geografiche, dal calcolo del *PageRank*[15] di un sito web al calcolo delle conoscenze in un social network e molti altri. Queste strutture sono infatti ottimizzate per la rappresen-

[15]Il PageRank è un algoritmo di analisi che, dato un insieme di elementi, assegna un punteggio ad ognuno di essi per quantificare la sua importanza all'interno dell'insieme. `https://it.wikipedia.org/wiki/PageRank`

Graph store
Neo4j
HyperGraphDB
InfoGrid
OrientDB
DEX

Tabella 1.4: graph store database

tazione di dati con una forte presenza di relazioni. Tuttavia l'assenza di relazioni, in un caso di studio, porta all'automatica esclusione di questo approccio. Infatti esso è stato sviluppato per garantire semplicità nelle operazioni di rappresentazione, manipolazione e recupero delle relazioni fra le entità del sistema.
Molto spesso si utilizza un ibrido di più approcci, utilizzando un approccio document store per il salvataggio dei dati e un approccio graph store per il salvataggio delle relazioni fra essi, come nel caso di Amazon. [3]

1.3 Introduzione ai sistemi di Machine Learning

Molto spesso, nel campo dell'analisi dei dati, si sente parlare di *Machine Learning* e di *Data Mining*. Anche se molto spesso questi termini vengono usati come sinonimi, in quanto possiedono caratteristiche comuni e utilizzano gli stessi algoritmi, essi si riferiscono a due analisi differenti.
Per *Data Mining* si intendono quelle tecniche, utilizzate su di un set di dati, che servono per scoprire proprietà nascoste nei dati ma già presenti. Per *Machine Learning* si intende un ramo dell'intelligenza artificiale che si occupa di creare sistemi in grado di imparare dai dati. Esso si basa sul principio della generalizzazione, il quale rappresenta la capacità di un algoritmo di lavorare

su situazioni nuove, dopo essere stato "addestrato" su un primo insieme di dati preso come esempio. Le tecniche di Machine Learning posso essere utilizzate in vari settori. Alcuni esempi sono:

- *Churn analysis.* Consiste nell'analisi della clientela, individuando i clienti che potrebbero passare alla concorrenza, riuscendo quindi ad agire in anticipo per evitare questa situazione.

- *Campagne pubblicitarie mirate.* Consiste nell'individuazione, tra i prospect, quelli con maggiore probabilità di acquistare prodotti dell'azienda, così da creare campagne mirate.

- *Market basket analysis.* Consiste nel suggerire ad un cliente altri prodotti, che potrebbero interessargli, in base al suo comportamento di acquisto.

- *Adaptive website.* Consiste nell'adattare la struttura e la rappresentazione delle informazioni, in un sito web, in base all'interazione dell'utente col sito, in modo da migliorarne l'interazione futura [12] .

- *Game playing.* Consiste nell'apprendere le regole e le tecniche di un gioco strategico, giocando attraverso altri avversari. Questo permette di migliorare l'esperienza finale per l'utente.

- *Recommender system.* Consiste nel calcolo del 'rating' o della preferenza che un utente potrebbe dare ad un determinato prodotto, al fine di aumentare l'esperienza di acquisto online [13].

Questi sono solo alcuni degli esempi di utilizzo di sistemi di Machine Learning. Le tecniche e gli algoritmi utilizzati da questi sistemi possono differenziarsi in maniera sostanziale, tuttavia esistono due macrogruppi di tecniche e algoritmi nel Machine Learning [14]:

Supervised Learning In questo caso l'algoritmo apprende utilizzando dati già categorizzati, al fine di predire il valore della categoria per i nuovi

elementi. In questo caso, quindi, il dataset di apprendimento contiene sia i dati in input che i risultati.

Unsupervised Learning In questo caso il processo di apprendimento avviene senza sapere qual'è il risultato corretto. Il dataset di apprendimento, infatti, non contiene alcuna categorizzazione.

Alcuni esempi di software che offrono questi servizi sono *Apache Mahout*, *Prediction.io*, R, *Apache Spark*, *Microsoft Azure Machine* e *MATLAB*.

Capitolo 2

NoSQL: MongoDB

In questo capitolo viene approffondito l'approccio document store prendendo come esempio *MongoDB* e valutando le sue caratteristiche e il suo utilizzo a livello di sviluppo di basi di dati. [15]

2.1 Introduzione

MongoDB è un database document-oriented a scopo generico potente, flessibile e scalabile. Combina la scalabilità del database ad alcune caratteristiche come indici secondari, query mirate, ordinamenti, aggregazioni e indici geospaziali. Come detto in precedenza uno degli aspetti fondamentali è la scalabilità. Questa necessità è richiesta, soprattutto ai giorni nostri, a causa dell'enorme quantità di dati da salvare e processare. Esistono due tipi di scalabilità: *scaling up*, ovvero l'aumento delle prestazioni della macchina, e lo *scaling out*, ovvero la distribuzione dei dati su più macchine. MongoDB è stato progettato per effettuare uno *scaling out*. Infatti modello document-oriented rende facile la divisione dei dati su più server.
Un'altro degli obbiettivi che MongoDB si è posto è quello delle performance. Esso, infatti, effettua un'allocazione dinamica dei documenti, così da incrementare le performance a scapito della memoria.

Nelle sezioni successive verranno approfondite le tecniche di memorizzazione e le procedure per la manipolazione dei dati.

2.2 Struttura

2.2.1 I Documenti

I documenti sono l'unità base per MongoDB e sono equivalenti alle righe nei RDBMS. Esso è rappresentato da un insieme ordinato di chiavi con associato un valore. La rappresentazione di un documento varia da linguaggio a linguaggio. Nel formato JSON il documento è rappresentato come segue:

```
{
  "arguments" : { "number" : 10 },
  "url" : "http://localhost:8080/restty-tester/collection",
  "method" : "POST",
  "header" : {
    "Content-Type" : "application/json"
  },
  "body" : [
    {
      "id" : 0,
      "name" : "name 0",
      "description" : "description 0"
    },
    {
      "id" : 1,
      "name" : "name 1",
      "description" : "description 1"
    }
  ],
  "output" : "json"
}
```

Figura 2.1: JSON example

Occorre ricordare che MongoDB è case sensitive[1] e sensibile ai tipi. Anche l'ordine è importante, tuttavia MongoDB si occupa di riordinarli autonomamente.

[1]Si dice *case sensitive* ogni operazione di analisi del testo che distingua due parole uguali in base all'uso di lettere maiuscole o minuscole. `https://it.wikipedia.org/wiki/Sensibile_alle_maiuscole`

2.2.2 Le Collezioni

Una collezione è un insieme di documenti. Si possono definire come le tabelle in un RDBMS. Le collezioni hanno uno schema dinamico, ciò significa che all'interno di una collezione ci possono essere più documenti con forme differenti. Non vi è quindi la necessità di utilizzare più collezioni per rappresentare documenti differenti, tuttavia mantenere i documenti separati per natura può semplicare lo sviluppo e può impedire di incorrere in errori in fase di esecuzione delle query.
Ogni collezione è identificata da un nome in codifica UTF-8[2].

2.2.3 I Tipi di Dato

I documenti in MongoDB sono rappresentati in `BSON` (Binary JSON). Le caratteristiche di questo formato sono la *leggerezza*, quindi risulta più rapido nella trasmissione sul network, la *traversabilità*, quindi risulta più rapido l'accesso ai documenti e ai campi, e l'*efficienza*, per quanto riguarda la codifica e la decodifica dei dati. L'insieme dei tipi supportati da MongoDB è il seguente:

null Questo tipo può essere utilizzato sia per rappresentare un campo vuoto, sia per rappresentare un campo non presente.

boolean Questo dato può assumere i valori di `true` e `false`.

number Questo campo è rappresentato in memoria da un numero *floating point*[3] lungo 64 bit. Esso può essere utilizzato sia per rappresentare numeri reali, che per rappresentare numeri interi.

string Questo tipo rappresenta una qualunque stringa avente la codifica UTF-8.

[2]UTF-8 (*Unicode Transformation Format 8 bit*) è una codifica dei caratteri Unicode in sequenze di lunghezza variabile di byte. `https://it.wikipedia.org/wiki/UTF-8`

[3]*Floating point* indica il metodo di rappresentazione approssimata dei numeri reali e di elaborazione dei dati usati dai processori per compiere operazioni matematiche. `https://it.wikipedia.org/wiki/Numero_in_virgola_mobile`

date Le date vengono salvate in millisecondi a partire dal 1 gennaio 1970. Il fuso orario non viene salvato.

regular expression Le query possono essere effettuate utilizzando espressioni regolari di JavaScript per il recupero dei dati.

array Un'insieme o una lista di valori può essere rappresentata attraverso un array.

embedded document Un documento può contenere un altro documento come valore di uno dei campi del documento stesso.

object id Questo tipo, che occupa 12 byte di memoria, viene utilizzato per rappresentare gli identificatori dei documenti.

binary data Questo tipo rappresenta una stringa di byte arbitrari. Questo è l'unico modo di rappresentare le stringhe con codifica differente da UTF-8.

code Rappresenta del codice JavaScript, come ad esempio uno script.

2.3 Inserimento, Aggiornamento e Cancellazione

Un esempio dell'utilizzo degli argomenti trattati in questa sezione è presente nell'Appendice A.

2.3.1 Inserimento e Creazione

L'inserimento è il metodo base per l'aggiunta di dati a MongoDB. L'operazione di inserimento inserisce nel documento un campo aggiuntivo (`_id`) se non specificato e salva il documento in memoria. Vi possono essere delle situazioni in cui si devono inserire, contemporaneamente, più documenti in una collezione. In questo caso si può utilizzare il cosiddetto *batch insert*. Esso

consente di inviare un array di documenti al database, velocizzando significativamente le prestazioni dell'inserimento. Tuttavia esso si può utilizzare solo per l'inserimento di più documenti all'interno della stessa collezione. Non è quindi possibile inserire più documenti in collezioni diverse.
Attualmente MongoDB non accetta messaggi più lunghi di 48MB, che quindi risulta l'unica limitazione per quanto riguarda l'inserimento.
Un'altra limitazione di memoria riguarda la dimensione massima di ogni documento che risulta pari a 16MB. Essendo arbitrario questo limite, potrebbe essere elevato in implementazioni future. Occorre notare che attualmente il limite inserito risulta essere comunque molto elevato, basti pensare che l'intero testo del libro *Guerra e Pace* occupa circa 3.14MB.
In fase di inserimento MongoDB effettua una validazione minima dei dati inseriti. Verifica infatti la struttura base del documento e inserisce un iddentificativo come detto in precedenza. Questa validazione minima mostra quanto possa essere facile inserire dei dati non validi. Una buona norma, infatti, è quella di effettuare una sorta di prevalidazione dei dati prima dell'inserimento nel database.

2.3.2 Aggiornamento

Una volta che il documento è salvato nel database può essere aggiornato utilizzando il metodo di aggiornamento. Esso si utilizza inserendo la query che referenzia il documento da aggiornare e un modificatore che descrive le modifiche da effettuare sul documento. L'aggiornamento è un'operazione atomica, perciò non possono avvenire due aggiornamenti in maniera simultanea. Nel caso di richieste contemporanee, esse verranno eseguite in sequenza e l'utima modifica utile sarà quella che arriverà al servizio per ultima.
L'aggiornamento più semplice è il rimpiazzo di un documento con un altro. Questo può avvenire nel caso di uno stravolgimento dello schema. Può succedere che la query di referenziazione recuperi più di un documento. In quel caso l'operazione non viene effettuata e MondoDB risponde con un messaggio d'errore.

Tuttavia questo è un caso particolare in quanto, generalmente, vengono modificate solo certe porzioni di un documento. MongoDB fornisce una serie di modificatori per semplificare operazioni d'aggiornamento complesse. Di seguito è presente una lista dei principali modificatori.:

`$set` Imposta il valore di un campo di un documento. Se il campo non è presente lo crea all'interno del documento selezionato. Questo modificatore può essere utilizzato per alterare lo schema.

`$unset` Rimuove il campo dal documento selezionato, se è presente.

Occorre sempre utilizzare uno di questi due modificatori per aggiungere, rimuovere o modificare un campo.

`$inc` Incrementa il valore di un campo o lo crea nel caso in cui non sia presente. Questo modificatore è simile a `$set` ma è pensato per la modifica di un dato di tipo *number*. Se viene passato un valore negativo al modificatore esso può essere usato anche per il decremento.

`$push` Inserisce un elemento all'interno di un campo di tipo *array* se esiste, altrimenti crea un nuovo campo di tipo *array*. Possono essere utilizzati dei modificatori aggiuntivi, in combinazione con questo, per effettuare operazioni più complesse:

- `$each` Viene utilizzato per inserire più elementi in un array attraverso una singola operazione.
- `$slice` Mantiene nell'array un numero di elementi pari al valore (negativo) inserito come parametro. Nel caso in cui il valore sia maggiore del numero degli elementi presenti, mantiene tutti gli elementi.
- `$sort` Ordina gli elementi dell'array in ordine crescente o decrescente in base al valore del parametro.

`$ne` Rappresenta l'acronimo di "not equal" e serve per effettuare l'aggiornamento condizionale di un documento nel caso in cui venga rispettata la condizione.

`$addToSet` Serve per aggiungere un elemento ad un array senza duplicazione, quindi solo se non è già presente.

`$pop` Serve per rimuovere il primo o l'ultimo elemento di un array in base al parametro fornito.

`$pull` Simile a `$pop` ma consente di specificare l'elemento dell'array da rimuovere.

`$` In combinazione con un altro modificatore serve per modificare il valore di un elemento dell'array che rispecchia la query utilizzata per il recupero.

Alcuni modificatori sono più rapidi di altri, in base al tipo di operazione che svolgono e in base alla modifica. Per esempio un incremento di valore di un numero non modifica lo spazio occupato da un documento mentre la modifica di una stringa o la rimozione di un elemento da un array si. Ciò comporta quindi un numero di operazioni maggiori da effettuare.
MongoDB consente anche l'inserimento di un documento nel database nel caso in cui non venga trovato durante un'operazione di query. Questo metodo è chiamato *upsert*. Oltre a questo è possibile anche modificare più documenti contemporaneamente. Ciò risulta utile nel caso di una migrazione dello schema.

2.3.3 Cancellazione

La cancellazione può avvenire su di un intera collezione oppure effettuando una query per differenziare i documenti da eliminare. Una volta che i dati vengo cancellati non c'è la possibilità ne di recuperarli in alcun modo ne di annullare l'operazione.
Eliminare un documento è, generalmente, un'operazione discretamente veloce. Tuttavia quanto si vuole svuotare una collezione, conviene cancellare la collezione stessa e ricreare gli indici su di una vuota. Si è dimostrato che su di un computer di fascia media, la cancellazzione di 1 milione di documenti

impiega circa 9000ms. Questo tempo scende a 1ms nel caso in cui si cancelli la collezione stessa, compresa di indici e metadati.

2.4 Query sui Dati

Per effettuare delle operazioni di query viene utilizzato il metodo `find`. Esso restituisce un sottoinsieme di elementi di una collezione in base alla query inserita. Il parametro di questo metodo indica le condizioni da rispettare per definire l'insieme dei documenti da restituire. Non sempre è necessario recuperare l'intero documento. Come secondo paramatro si può definire quali campi di un documento restituire. In questo modo la quantità di dati restituiti risulta minore e la query più performante.
Una limitazione delle query è il fatto che i valori delle chiavi della condizione devono essere delle costanti. Per esempio non è possibile specificare che un campo di un documento sia uguale ad un'altro dello stesso. Un esempio dell'utilizzo degli argomenti trattati in questa sezione è presente nell'Appendice A.

2.4.1 Operatori Condizionali

Esistono alcuni operatori di confronto in MongoDB: `$lt`, `$lte`, `$gt`, `$gte`, `$ne` che corrispondono rispettivamente a $<$, $\leq$, $>$, $\geq$, $\neq$. Combinare questi valori può essere utile per indicare un range di valori nella condizione.

2.4.2 Operatori Logici

Esistono molti modi di effettuare dei confronti logici in MongoDB:

- `$in` Attraverso questo operatore è possibile verificare più valori per un singolo campo del documento (OR), anche con tipi di dato differenti. Per verificare l'opposto di questa condizione si utilizza l'operatore `$nin`.

- `$or` A differenza di `$in`, questo operatore può essere utilizzato per verificare più valori per più campi del documento. Per verificare l'opposto di questa condizione si utilizza l'operatore `$nor`.
- `$not` Questo operatore viene utilizzato per negare una condizione.
- `$and` Questo operatore viene utilizzato per verificare la veridicità di due o più condizioni.
- `$exist` Questo operatore viene utilizzato per verificare se un campo è presente oppure no nel documento.

2.4.3 Operatori sugli Array

In MongoDB esistono alcuni operatori che consento di verificare una condizione riferita agli elementi di un campo del documento di tipo *array*:

- `$all` Attraverso questo operatore è possibile verificare che il campo contenga tutti gli elementi passati come parametro.
- `$size` Serve per verificare il numero di elementi presenti nell'array.
- `$slice` Restituisce il numero di elementi indicato presenti nell'array.

Questi sono solo alcuni degli operatori presenti in MongoDB. Combinandoli assieme è possibile effettuare query molto complesse che permettono quindi di adattarsi ad ogni caso.

2.5 Sharding

Lo *sharding*, noto anche come *partitioning*, si riferisce alla tecnica di divisione dei dati su più macchine. Così facendo è possibile aumentare la mole di dati e la velocità di computazione senza dover utilizzare macchine più performanti. Normalmente è sempre possibile effettuare un cosiddetto *manual sharding*, che consiste nella creazione di più connessioni a database differenti

a livello di applicazione. Questa soluzione può essere funzionale ma risulta di difficile manutenzione nel caso in cui si voglia aggiungere o rimuovere un nodo dal *cluster*[4].

2.5.1 Lo Sharding in MongoDB

MongoDB supporta una tecnica chiamata *autosharding*, che nasconde l'architettura dall'applicazione che lo utilizza simulando, attraverso varie macchine, un singolo sistema. Questa tecnica consente la creazione di un cluster di più macchine, chiamato *shards*, e di dividere le collezioni su di esse, creando dei sottoinsiemi di dati. Per poter simulare un solo sistema esso utilizza un processo di indirizzamento chiamato `mongos`. Esso mantiene una tabella di contenuti che indica quale dato si trova su ogni *shard*, indirizzando le richieste verso quello pertinente (Figura 2.2).

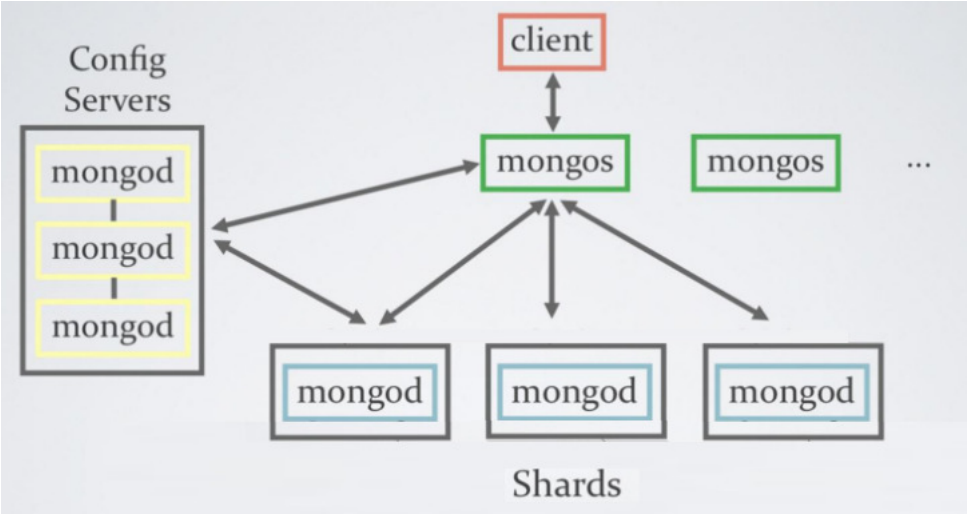


Figura 2.2: MongoDB Sharding [16]

Decidere quando attuare questa tecnica è sempre una scelta complicata. Questo perchè essere precipitosi e attuarla troppo presto provoca un incre-

[4]In informatica un *cluster* (dall'inglese grappolo), è un insieme di computer connessi tra loro tramite una rete telematica. `https://it.wikipedia.org/wiki/Computer_cluster`

mento della complessità del sistema e delle scelte progettuali, difficilmente modificabile in futuro. D'altro canto, effettuare lo sharding di un sistema sovraccarico provocherebbe, inevitabilmente, il blocco del sistema stesso per tutto il periodo di transizione. Generalmente lo sharding viene utilizzato per:

- Incrementare la RAM disponibile.
- Incrementare lo spazio su disco.
- Ridurre il carico del server.
- Leggere e scrivere dati con un throughput[5] maggiore rispetto a quello gestito da un singolo sistema.

Generalmente la scelta ottimale è quella di passare da un sistema *non-sharded* ad un sistema con tre o più shards direttamente. Questo perchè, con un numero minore, si ha una maggiore latenza e un throughput inferiore a causa della gestione dei metadata, dell'indirizzamento e dello spostamento dei dati.

2.6 Strumenti di Amministrazione

Esistono alcuni strumenti di amministrazione di MongoDB. Essi possono essere visuali, come *Robomongo*, oppure a linea di comando, come la *MongoDB Shell*.

2.6.1 MongoDB Shell

La MongoDB Shell è un terminale a linea di comando basato su JavaScript, che consente l'interazione con un'istanza di MongoDB da linea di comando. Essa è utile sia per amministrare il servizio, sia per effettuare operazioni sui dati. Essa è un interprete di JavaScript, per cui ogni operazione ammessa in JavaScript può essere effettuata sulla shell allo stesso modo. Con

[5]Per *throughput* di un canale di comunicazione si intende la sua capacità di trasmissione "effettivamente utilizzata". `https://it.wikipedia.org/wiki/Throughput`

essa è possibile anche creare script multi-riga, che vengono validati dalla shell stessa.
La potenza della shell, oltre a quello detto in precedenza, è il fatto di essere essa stessa un client MongoDB. All'avvio, infatti, essa si connette ad un database di prova e assiocia alla variabile globale `db` la connessione. Attraverso quella è possibile effettuare tutte le operazioni sul database e sui dati.

2.7 Diffusione

Rank			DBMS	Database Model	Score		
Nov 2015	Oct 2015	Nov 2014			Nov 2015	Oct 2015	Nov 2014
1.	1.	1.	Oracle	Relational DBMS	1480.95	+13.99	+28.82
2.	2.	2.	MySQL	Relational DBMS	1286.84	+7.88	+7.77
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1122.33	-0.90	-97.87
4.	4.	↑ 5.	MongoDB +	Document store	304.61	+11.34	+59.87
5.	5.	↓ 4.	PostgreSQL	Relational DBMS	285.69	+3.56	+28.33
6.	6.	6.	DB2	Relational DBMS	202.52	-4.28	-3.71
7.	7.	7.	Microsoft Access	Relational DBMS	140.96	-0.87	+2.12
8.	8.	↑ 9.	Cassandra +	Wide column store	132.92	+3.91	+40.93
9.	9.	↓ 8.	SQLite	Relational DBMS	103.45	+0.78	+8.17
10.	10.	↑ 11.	Redis +	Key-value store	102.41	+3.61	+20.06
11.	11.	↓ 10.	SAP Adaptive Server	Relational DBMS	83.71	-1.93	-0.91
12.	12.	12.	Solr	Search engine	79.77	+0.71	+2.96
13.	13.	13.	Teradata	Relational DBMS	77.08	+3.64	+9.85
14.	14.	↑ 16.	Elasticsearch	Search engine	74.77	+4.55	+31.71
15.	15.	15.	HBase	Wide column store	56.46	-0.78	+9.49
16.	16.	↑ 17.	Hive	Relational DBMS	54.91	+1.35	+18.30
17.	17.	↓ 14.	FileMaker	Relational DBMS	51.73	+1.95	+0.39
18.	18.	↑ 20.	Splunk	Search engine	44.61	+1.11	+12.44
19.	19.	↑ 21.	SAP HANA	Relational DBMS	39.62	+0.52	+11.26

Figura 2.3: database rank table, November 2015 [17]

Come si nota dalla Figura 2.3, la diffusione e l'utilizzo di MongoDB come DBMS è in continua crescita, anche grazie ai continui sviluppo che lo stanno rendendo molto potente e facile da usare. Alcuni esempi di aziende famose che utilizzano questo sistema sono i seguenti [18]:

- Craigslist per la memorizzazione di oltre 2 miliardi di documenti.
- Forbes per la memorizzazione di articoli e dati societari.
- The New York Times nella sua applicazione di caricamento di fotografie.
- The Guardian per il suo sistema di identificazione.
- CERN come back-end del Data Aggregation System nel Large Hadron Collider.
- Foursquare implementa MongoDB su Amazon AWS per memorizzare le località e le registrazioni degli utenti nelle località.
- eBay lo utilizza per i suggerimenti della ricerca e per State Hub, il Cloud Manager interno.

Capitolo 3

Machine Learning: PredictionIO

In questo capitolo viene introdotto il servizio *PredictionIO* e le sue caratteristiche principali, oltre che alla sua struttura e al metodo di utilizzo. [19]

3.1 Introduzione

PredictionIO è un server open source[1] di Machine Learning utilizzato per sviluppare e rilasciare applicazioni di previsione. Esso possiede un motore di calcolo chiamato *engine*. L'architettura dell'engine è basata su *MVC* [2] *per Machine Learning*. La parte principale di questo server è la *engine deployment platform* basata su Apache Spark[3]. Essa consente di rilasciare engine di previsione come servizi distribuiti sul web. Oltre a queste caratteristiche è

[1] *Open source*, in informatica, indica un software di cui gli autori rendono pubblico il codice sorgente. `https://it.wikipedia.org/wiki/Open_source`

[2] Il *Model-View-Controller* (MVC) è un pattern di progettazione molto diffuso nello sviluppo di sistemi software. `https://it.wikipedia.org/wiki/Model-View-Controller`

[3] Apache Spark è un framework open source di cluster computing sviluppato nei laboratori AMPLab della University of California, Berkeley. `https://en.wikipedia.org/wiki/Apache_Spark`

presente anche un *Event Server*, ovvero un sistema, scalabile, di acquisizione e analisi dei dati basato su Apache HBase[4].

3.2 Componenti Principali

PredictionIO è composto principalmente dai seguenti componenti:

- **PredictionIO platform** Il componente di Machine Learning utilizzato per creare, utilizzare e rilasciare dei engine con algoritmi di Machine Learning.

- **Event Server** Il sistema di analisi di Machine Learning utilizzato per unificare gli eventi provenienti da più piattaforme.

- **Template Gallery** Una collezione di engine di Machine Learning che possono essere scaricati e modificati liberamente.

3.2.1 Event Server

In uno scenario comune, l'event server di PredictionIO acquisisce dati in continuazione dalle applicazioni. Fatto questo l'engine di PredictionIO costruisce dei modelli predittivi, con uno o più algoritmi, utilizzando i dati. Una volta rilasciato come servizio web, esso viene utilizzato principalmente per due scopi:

1. Fornire i dati all'engine, per la valutazione e l'apprendimento del modello.

2. Offrire una visione unificata per l'analisi dei dati.

[4]HBase è un database non relazionale distribuito, successore di Google's BigTable, scritto in linguaggio Java. `https://en.wikipedia.org/wiki/Apache_HBase`

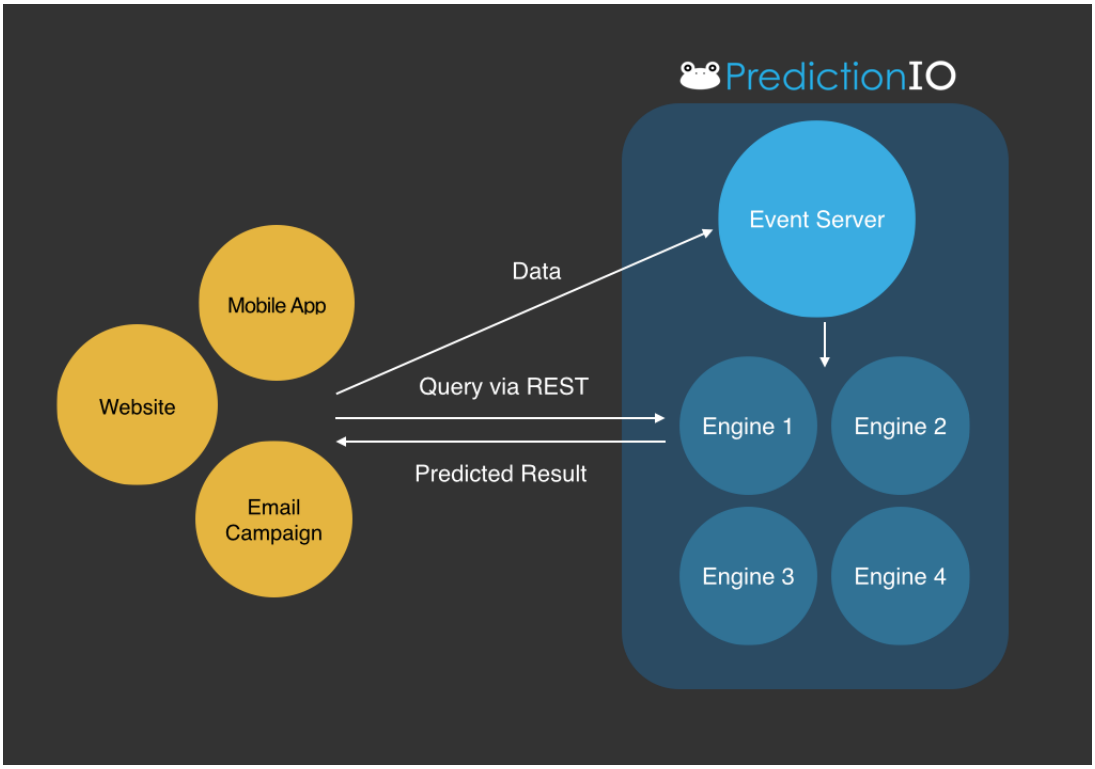


Figura 3.1: PredictionIO Components

3.2.2 Engine

Questo compenete è il responsabile delle previsioni. Esso contiene uno o più algoritmi di Machine Learning che utilizza per leggere i dati per l'apprendimento e costruire modelli di previsione. Una volta che viene rilasciato come servizio web, comunica con l'applicazione attraverso l'utilizzo API REST in tempo reale.

Le funzioni principali di un engine sono l'apprendimento del modello di previsione, partendo dai dati di apprendimento, e la comunicazione real-time con le applicazioni. Ogni engine computa i dati e construisce il modello di previsione in maniera indipendente. Per questo è possibile rilasciare più engine, che assolvono a funzioni diverse, sulla stessa piattaforma.

La Template gallery offre una moltitudine di schemi di engine per i compiti

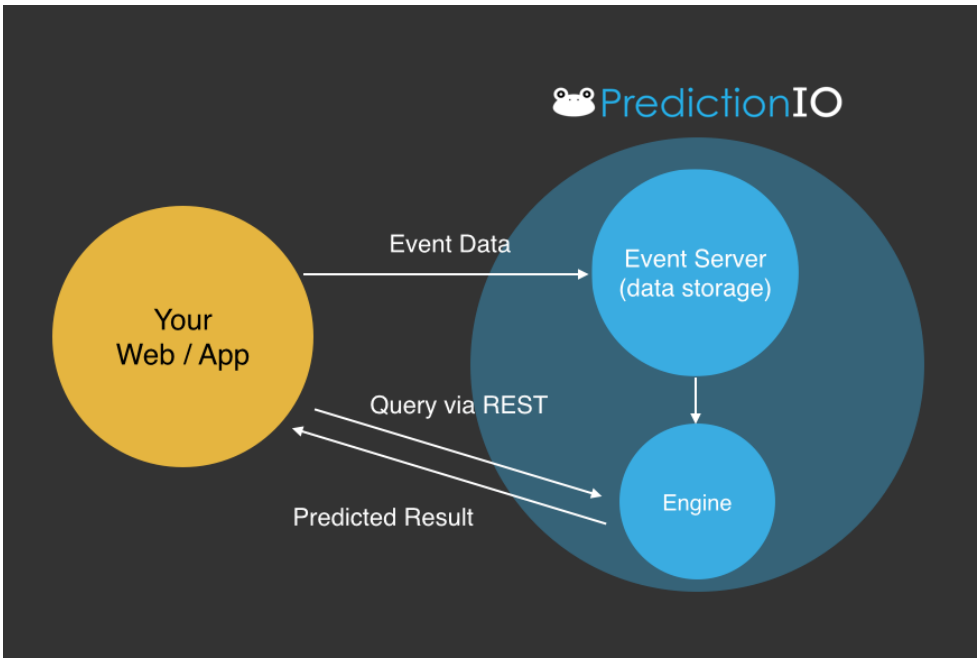


Figura 3.2: PredictionIO Event Server

di Machine Learning più disparati. Essendo open source, è possibile partire da quest'ultimi e modificarli per creare un engine che si adatti ad ogni situazione.

3.2.3 Template Gallery

La Template Gallery fornisce molti engine che possono svolgere varie funzioni. Segue una lista di alcuni engine ufficiali:

- **Universal Recommender** Recommender system[5] generico, potenzialmente utilizzabile in qualunque ambito.
- **E-Commerce Recommendation** Recommender system sviluppato per applicazioni di E-Commerce[6], con alcune caratteristiche quali l'esclusio-

[5]Sistema di filtraggio delle informazioni utilizzato per predirre le preferenze dell'utente. `https://www.wikidata.org/wiki/Q554950`

[6]Commercio Elettronico. `https://it.wikipedia.org/wiki/Commercio_elettronico`

ne dai risultati dei prodotti terminati, e la raccomandazione di prodotti ad utenti appena registrati.

- **Product Ranking** Questo engine stila una lista dei prodotti ordinata sencondo le preferenze di un utente.

- **Lead Scoring** Questo engine calcola la probabilità di un utente, nella sessione corrente, di convertire il suo ruolo(es. registrazione ad un sito durante la sua navigazione).

- **Complementary Purchase** Recommender system utilizzato per i prodotti complementari, ovvero quei prodotti che spesso un utente acquista insieme ad altri.

- **Similar Product** Recommender system utilizzato per trovare prodotti "simili" a quelli forniti. La similarità non è definita nè dall'utente, nè dai prodotti, ma dalle azioni precedenti dell'utente stesso.

- **Classification** Engine utilizzato per classificare gli utenti sulla base di tre caratteristiche fondamentali. Esso ha integrato l'algoritmo di classificazione bayesiana[7] di Apache Spark MLib.

3.3 Componenti DASE

Come mostrato dalla Figura 3.3, le componenti di un engine vengono definite DASE:

Data Source e Data Preparator Il Data Source si occupa di leggere i dati da una sorgente e di trasformarli nel formato desiderato. Il Data Preparator si occupa di esaminare i dati e inviarli all' Algorithm per il modello d'apprendimento.

[7]Un classificatore bayesiano è un classificatore basato sull'applicazione del teorema di Bayes. `https://it.wikipedia.org/wiki/Classificatore_bayesiano`

Algorithm Questo componente comprende l'algoritmo di Machine Learning e il settaggio dei suoi paramateri, determinando come viene costruito un modello di previsione.

Serving Questo componente si occupa di ricevere le query e inviare i risultati delle previsioni. Nel caso in cui l'engine possieda più algoritmi, esso si occupa anche di combinare i risultati di tutti gli algoritmi in uno solo.

Evaluation Metrics Questo componente si occupa di valutare l'accuratezza di una previsioni assegnandoli un valore numerico. Esso può essere utilizzato per confrontare gli algoritmi o i parametri di configurazione di essi.

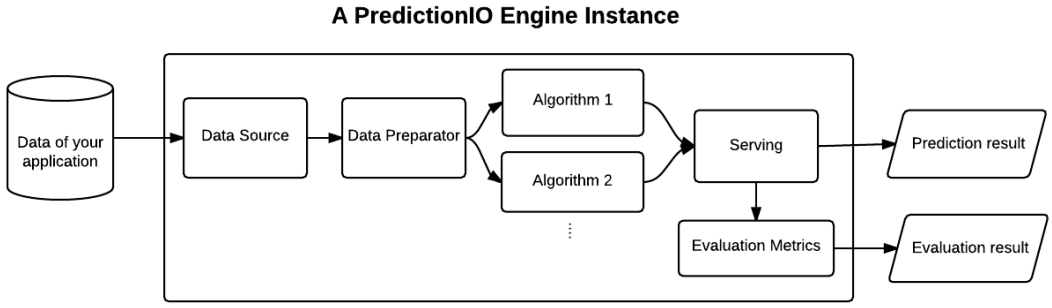


Figura 3.3: PredictionIO Engine DASE Components

3.4 Integrazione con le Applicazioni

Il metodo principale per l'interazione delle applicazioni con PredictionIO è attraverso le API REST, ovvero richieste Http[8] che restituiscono un risultato in formato `JSON`. Esistono tuttavia delle API anche per altri linguaggi di programmazione tra cui Java(compreso Android), PHP, Python e Ruby.

[8]Hypertext Transfer Protocol. `https://it.wikipedia.org/wiki/Hypertext_Transfer_Protocol`

Inoltre, essendo un sistema open source, sono presenti anche altre API sviluppate dalla comunità.
Questo perchè, grazie alla sua struttura, PredictionIO risulta di facile installazione e utilizzo. Un approfondimento è presente nel capitolo 4.

Parte II

Progettazione e Implementazione

Capitolo 4

Caso di Studio: Item Price Watcher

Item Price Watcher è un applicativo desktop sviluppato per monitorare il prezzo di un prodotto relativo ad un sito di e-commerce, effettuando una previsione sul suo andamento e consigliando all'utente altri prodotti simili presenti nel sistema. Questo applicativo consente la gestione delle sorgenti da cui provengono i prodotti, la gestione dei prodotti e delle loro caratteristiche, incluso il prezzo, manualmente o attraverso servizi web, e la visualizzazione di statistiche sui vari prodotti. Risulta possibile segnare i prodotti che vengono acquistati, in modo tale che, attraverso un sistema di raccomandazioni, sia possibile consigliare dei prodotti simili, sulla base delle scelte fatte dagli altri utenti. Questo applicativo è sviluppato per piattaforme Linux, Mac OS e Windows sia a 32bit che a 64bit.

4.1 Progettazione

L'applicattivo è stato implementato utilizzando le ultime tecnologie web, virtualizzate attraverso l'utilizzo di `nw.js`, un componente open source di `node.js`. Il database è stato implementato utilizzando MongoDB, il sistema

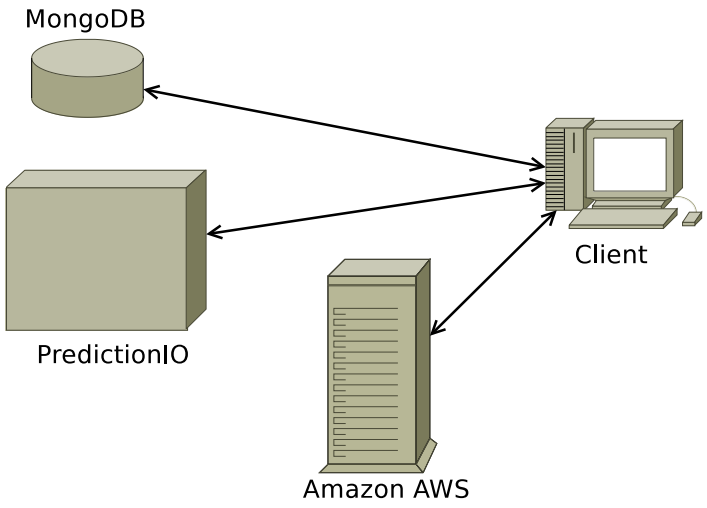


Figura 4.1: Struttura Item Price Watcher

di recommendation è stato implementato utilizzando PredictionIO e per il web service per il recupero di prodotti è stato utilizzato Amazon AWS.

4.2 Struttura e Modalità di Utilizzo

All'avvio del programma appare un menù sulla destra da dove è possibile scegliere la sezione che più interessa. La pagina principale è la `dashboard`, dove sono presenti delle statistiche sul sistema.

4.2.1 Dashboard Page

La Figura 4.2 mostra la pagina principale che si incontra all'avvio del programma. In questa pagina sono presenti delle statistiche numeriche che rappresentano i dati presenti nel sistema. In particolare vengono indicati il numero di sorgenti, il numero di prodotti presenti, il numero di prodotti acquistati e un grafico rappresentante il numero di prezzi per sorgente presenti nel sistema.

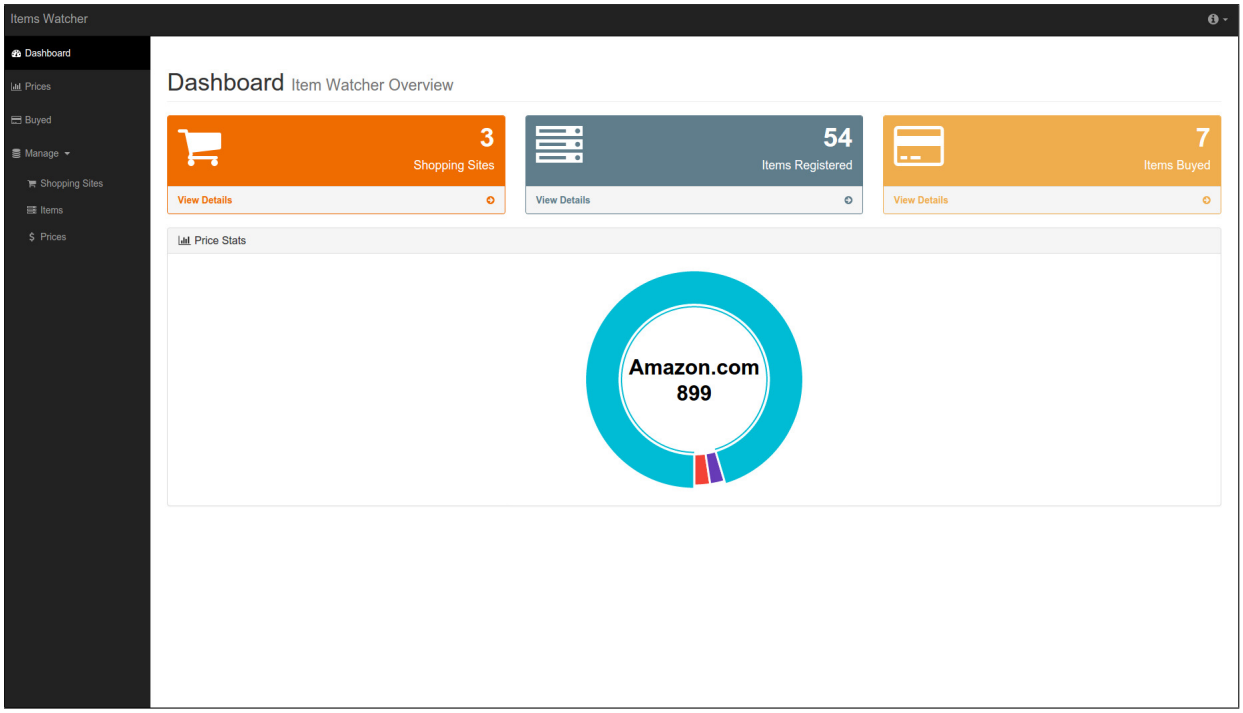


Figura 4.2: Dashboard Page

4.2.2 Prices Page

La Figura 4.3 mostra la pagina in cui vengono rilevate le statistiche sui prezzi dei prodotti. Cliccando su di un prodotto presente nella tabella sulla sinistra è possibile vedere le statistiche sulla destra. Le statistiche sono composte da un grafico che mostra l'andamento dei prezzi registrati, e da tre pannelli che indicano rispettivamente: il prezzo massimo raggiunto, la previsione del prezzo nel futuro prossimo e il prezzo minore raggiunto. In alto a destra sono presenti due pulstanti: uno verde con un più che serve per l'aggiunta manuale di un prezzo, specificando sorgente e data, e uno bianco con la scritta *Update*, utilizzato per l'aggiornamento automatico dei prezzi utilizzando il servizio *Amazon AWS*. L'aggiornamento automatico può essere effettuato esclusivamente per i prodotti inseriti attraverso lo stesso servizio. Il sistema è in grado autonomamente di capire se è possibile aggiornare il

prezzo di un prodotto oppure no.

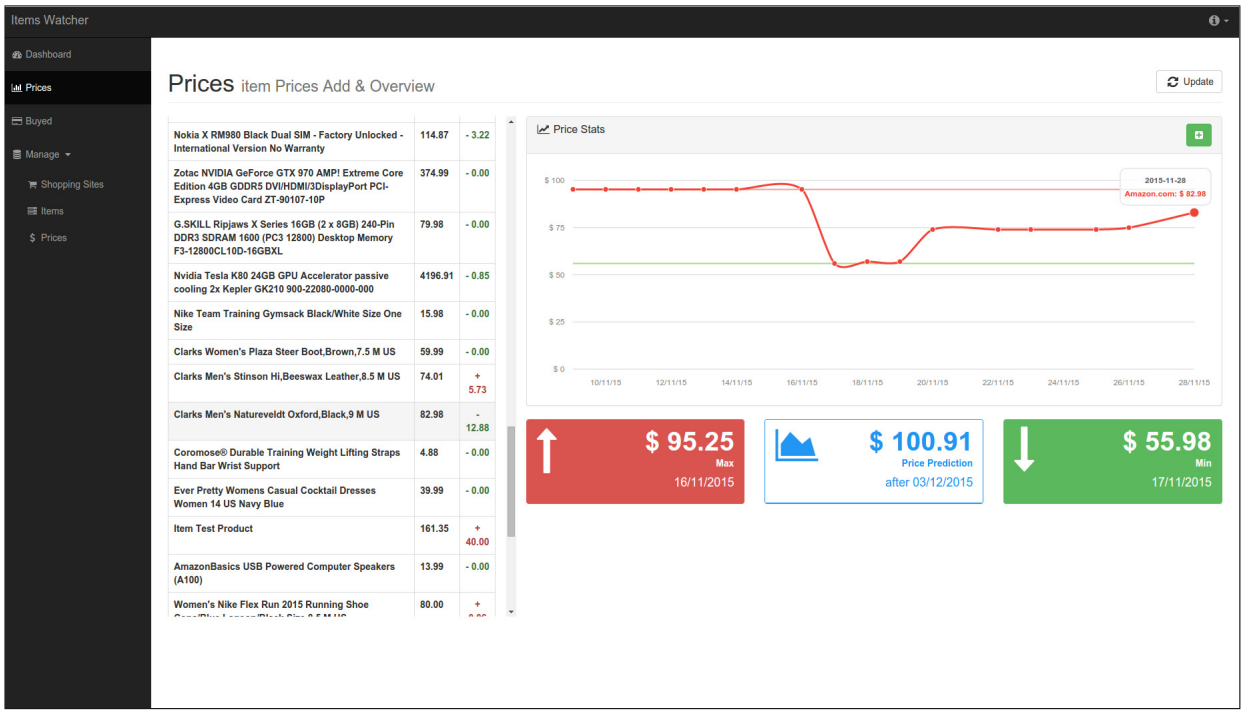


Figura 4.3: Prices Page

4.2.3 Buyed Page

La Figura 4.4 mostra la pagina in cui vengono visualizzati i prodotti che sono stati comprati dall'utente. Sulla parte destra sono presenti, inoltre, dei suggerimenti di prodotti che potrebbero interessare, sulla base degli acquisti degli altri utenti. In questa pagina è possibile anche rimuovere uno dei prodotti da quegli acquistati.

4.2.4 Manage Page

Questo gruppo di pagine viene utilizzato per la gestione dei dati a livello di anagrafico. Sono presenti tre pagine principali che servono per gestire le varie tipologie di dato.

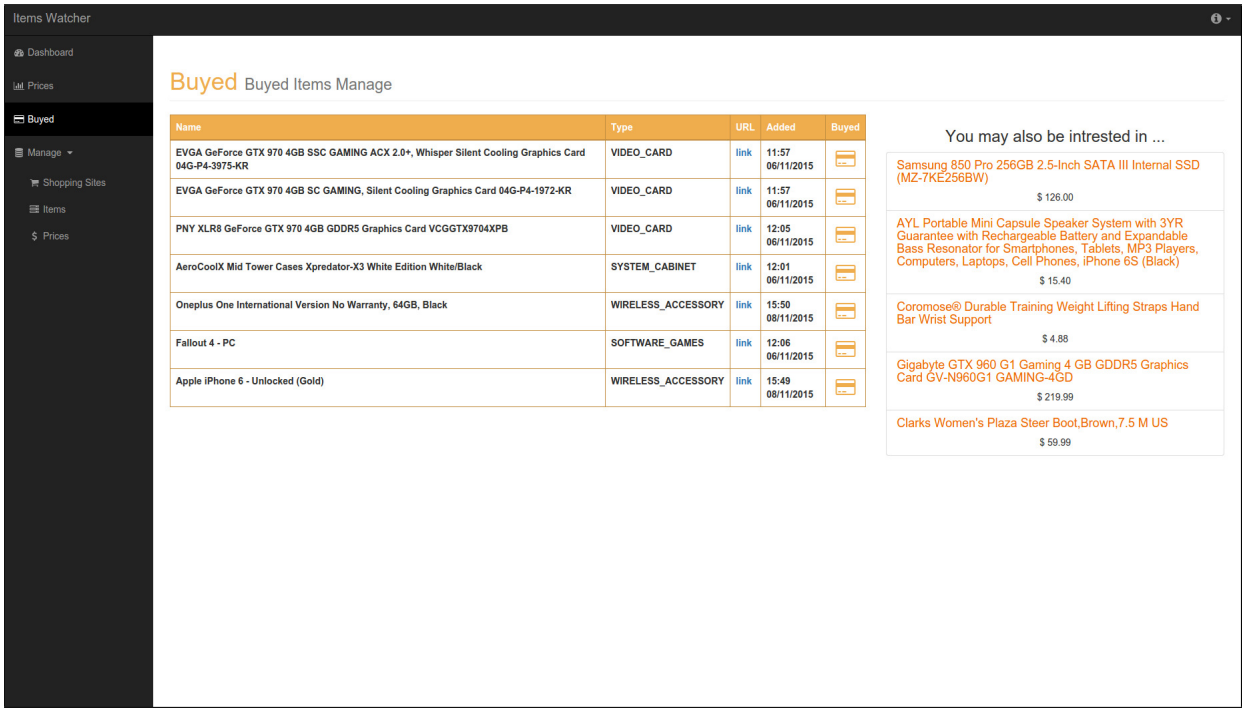


Figura 4.4: Buyed Page

Manage Prices

La Figura 4.5 rappresenta la pagina di gestione dei prezzi. In questa pagina è possibile gestire i prezzi inseriti relativi ad ogni prodotto. Da qui è possibile cancellare o modificare i singoli prezzi. Essi appaiono nella lista a destra una volta che si clicca su di un prodotto.

Manage Items

La Figura 4.6 rappresenta la pagina di gestione dei prodotti. Da qui è possibile aggiungere, modificare o cancellare i prodotti. L'aggiunta può essere sia manuale che automatica, effettuata attraverso una ricerca in una finestra modale apposito che, attraverso le API Amazon, recupera i prodotti da *amazon.com* e tutte le loro caratteristiche e permette di aggiungerli al sistema, in base a quelli selezionati.

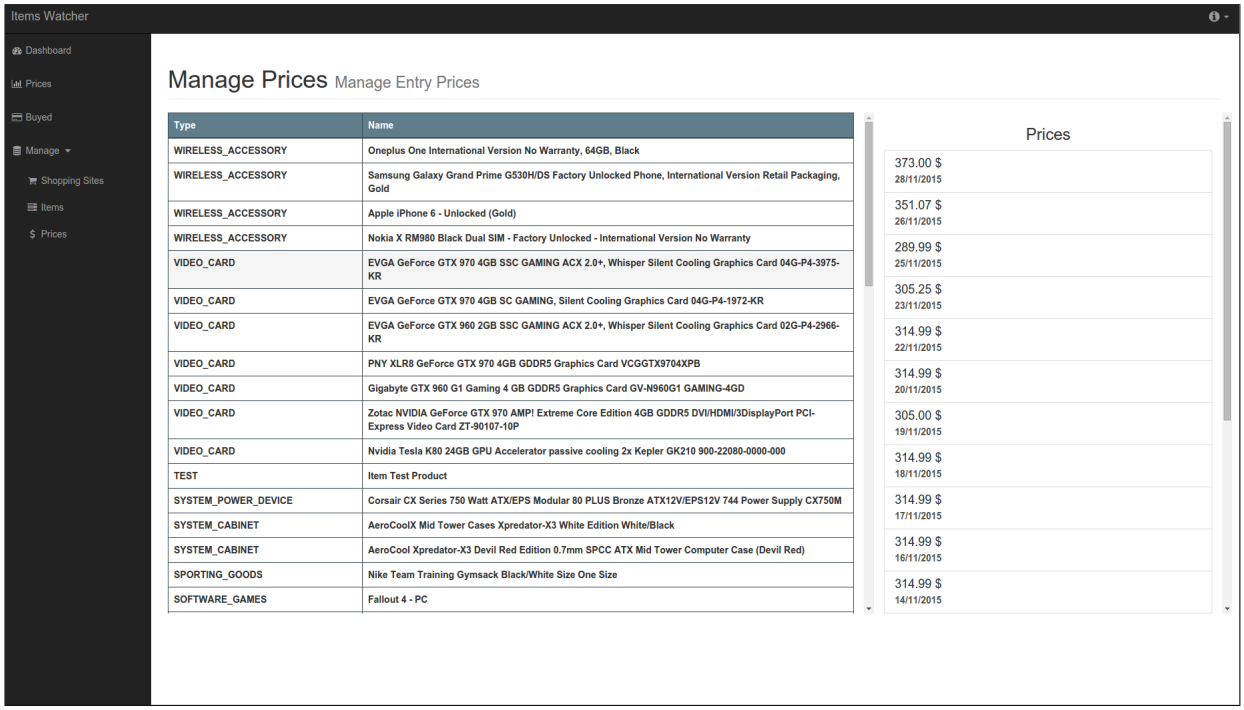


Figura 4.5: Manage Prices Page

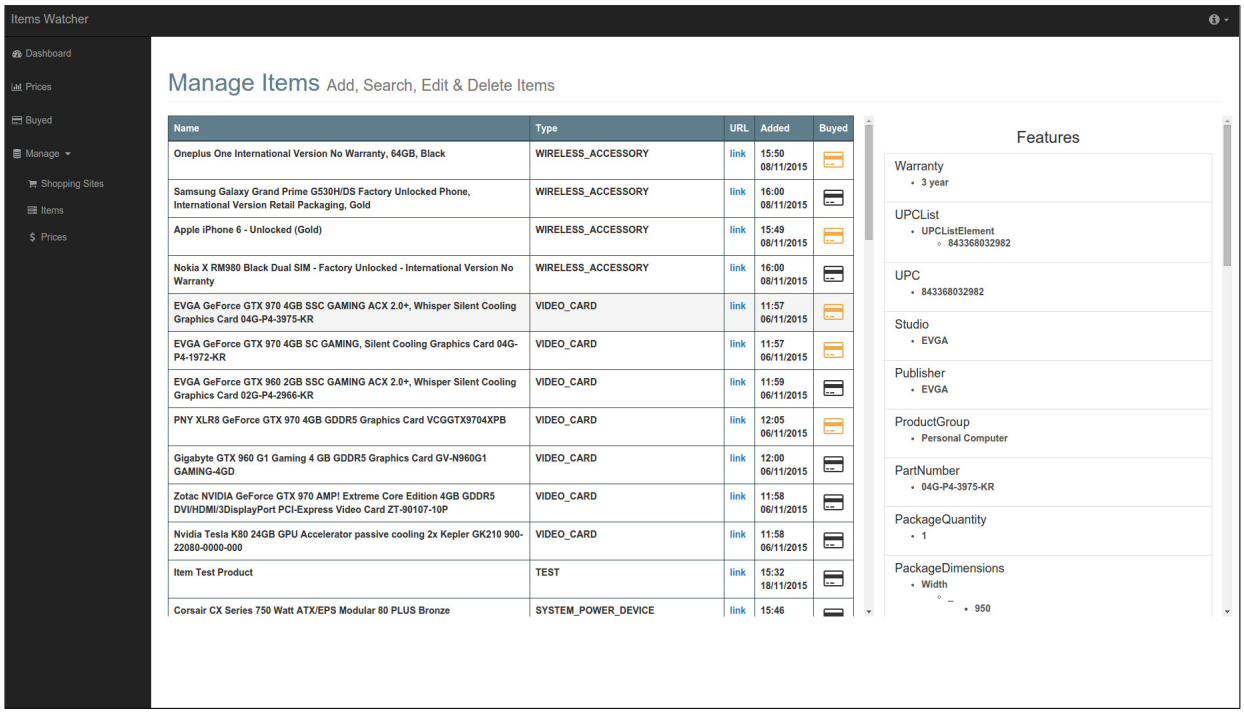


Figura 4.6: Manage Items Page

Manage Shopping Sites

La Figura 4.7 rappresenta la pagina per la gestione delle sorgenti dei prezzi. In questa pagina è possibile aggiungere, modificare o rimuovere una sorgente specifica.

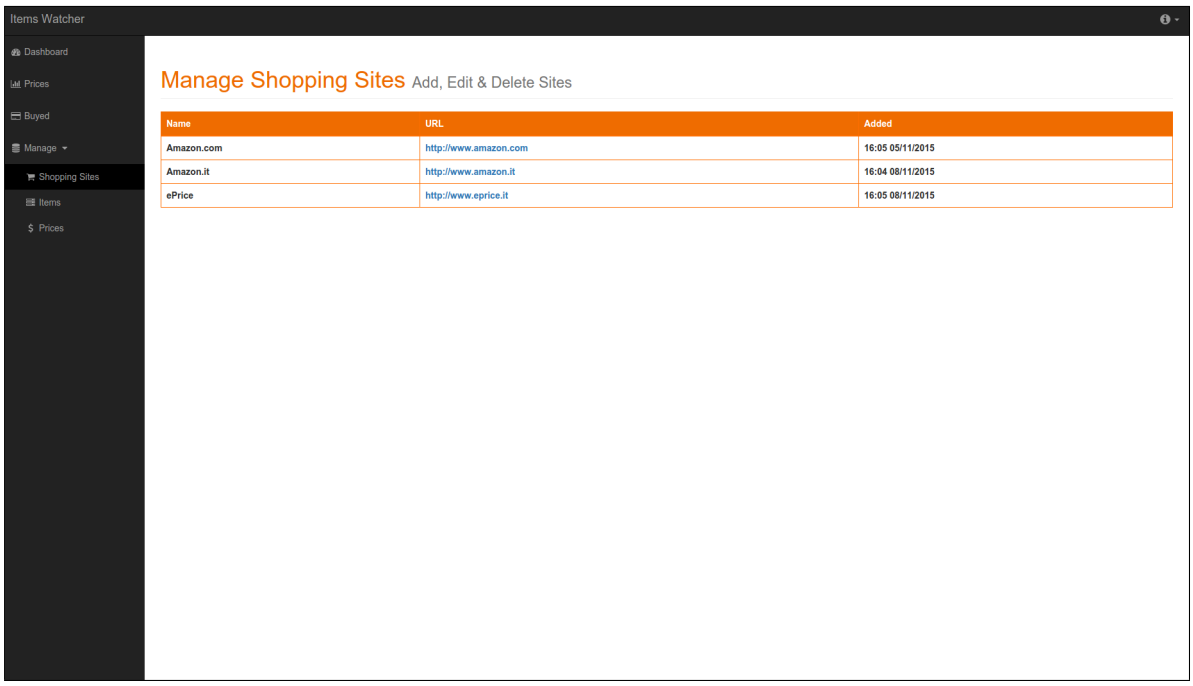


Figura 4.7: Manage Sites Page

4.3 Client

Come detto in precedenza il client è stato creato utilizzando le ultime tecnologie web, in particolare utilizzando una combinazione di JavaScript, Html5, Css3 e Nodejs e derivati framework e librerie. Attraverso l'utilizzo di nw.js è stata, successivamente, creata un applicazione desktop, virtualizzando il sito creato, senza bisogno di scrivere codice aggiuntivo.
Per il recupero automatico dei prodotti sono state utilizzate le API AMAZON

AWS. In questo modo è stato possibile recuperare prodotti, caratteristiche e prezzi direttamente da *amazon.com*.

4.3.1 Framework e Librerie

Come detto in precendenza sono stati utilizzati framework web e librerie di JavaScript per lo sviluppo dell'applicazione. Segue una lista dei più rilevanti.

Bootstrap

Sviluppato inizialmente da un designer e un developer di Twitter, Bootstrap è diventato uno dei front-end framework[1] più popolari su internet, anche grazie al fatto di essere open source. Il suo obbiettivo è quello di facilitare lo sviluppo di siti web responsive[2] e applicazioni web. Una sua peculiarità è quella di essere compatibile con la maggior parte dei browser presenti oggi. [20]
Attualmente Bootstrap è utilizzato da una buona parte dei siti presenti sul web. Un esempio sono:

- Spotify - The Drop (`http://www.spotify-thedrop.com/#/`).
- HBO NOW (`https://order.hbonow.com/`).
- MongoDB (`https://www.MongoDB.com/cloud`).
- NASA (`https://www.nasa.gov/`).

[1] Un front-end framework rappresenta un'interfaccia per l'utente finale. `https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)`

[2] Il *responsive web design* (RWD), indica una tecnica di web design per la realizzazione di siti in grado di adattarsi graficamente in modo automatico al dispositivo coi quali vengono visualizzati. `https://it.wikipedia.org/wiki/Design_responsivo`

jQuery

jQuery è una librearia di JavaScript leggera, veloce e piena di funzionalità. Essa consente di effettuare con facilità la manipolazione degli elementi del DOM[3], la cattura degli eventi, le animazione e le chiamate AJAX[4]. Essendo versatile e funzionante su tutti i browser, oggi è una libreria molto diffusa nel web ed è supportata da aziende come WordPress o Neobux. [21]

regression.js

Questa libreria JavaScript contiene al suo interno un'insieme di metodo per la stima dell'andamento di una serie storica [22]. Attualmente supporta delle funzioni per effettuare alcune regressioni quali:

- Regressione Lineare
- Regressione Esponenziale
- Regressione Logaritmica
- Distribuzione di Potenza
- Regressione Polinomiale

In questa applicazione è stata utilizzata la regressione polinomiale, vista la maggior similarità in relazione all'andamento dei prezzi rispetto alle altre. Un approfondimento verrà affrontato nel Capitolo 5.

morris.js

Questa libreria JavaScript consente la creazione di vari tipi di grafici per la rappresentazione dei dati. Essa consente una totale personalizzazione

[3]Il *Document Object Model* (DOM) è una forma di rappresentazione dei documenti strutturati come modello orientato agli oggetti. `https://it.wikipedia.org/wiki/Document_Object_Model`

[4]In informatica AJAX(*Asynchronous JavaScript and XML*) è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive. `https://it.wikipedia.org/wiki/AJAX`

dei grafici, oltre che l'opportunità di abilitare e disabilitare la funzionalità responsive. [23]

4.3.2 Node.js

Node.js è un framework asincrono orientato agli eventi ed è stato sviluppato per costruire applicazioni web scalabili. Esso consente più connessioni contemporanee, comportandosi come un server[5].
Grazie al componente `child_process` è possibile avviare dei sotto processi che condividono gli stessi socket[6], in modo da simulare un'architettura a cluster e quindi suddividere il carico su più risorse. [24]
In questo progetto è stata utilizzata la versione *4.4.2 LTS*[7] oltre che a suoi moduli elencati di seguito.

Mongoose

Mongoose è un modulo di Node utilizzato per creare un modello di dati in MongoDB. Questo modulo consente infatti di aggiungere uno strato di validazione a MongoDB, consentendo di poter generare dei valori di default o di effettuare dei controlli sui valori dei vari campi. Al tempo stesso, tuttavia, consente la modifica dello schema creato. Esso si occupa, autonomamente, di generare una collezione per ogni schema che si vuole rappresentare e di effettuare le query nella collezione corretta, semplificando anche l'utilizzo dei modificatori o degli operatori di aggregazione. Al fine di definire lo schema esso introduce i seguenti tipi di dato:

String utilizzato per stringhe e caratteri.

Number utilizzato per tutti i tipi numerici.

[5]Un server in informatica è un sottosistema informatico di elaborazione e gestione del traffico di informazioni. `https://it.wikipedia.org/wiki/Server`

[6]Un socket è il punto in cui il codice applicativo di un processo accede al canale di comunicazione per mezzo di una porta. `https://it.wikipedia.org/wiki/Socket_(reti)`

[7]Long Term Support. `https://it.wikipedia.org/wiki/Ciclo_di_vita_del_software`

Date utilizzato per rappresentare le date, contiene anche alcuni metodi per il recupero e la modifica.

Buffer utilizzato per rappresentare un insieme di byte, come ad esempio immagini o file. Mongoose stesso si occupa della conversione nel formato appropriato durante l'esecuzione della query.

Boolean utilizzato per rappresentare un valore *true* o *false*.

Mixed utilizzato per non definire un tipo fisso ad un campo. Esso infatti potrà assumere qualunque valore senza che venga effettuata una validazione.

Objectid utilizzato per rappresentare un campo di tipo id. Questo è il tipo che rappresenta l'id del documento in MongoDB.

Array utilizzato per rappresentare un array, contiene al suo interno alcuni metodi per la gestione.

Oltre a questi tipi è possibile utilizzare dei tipi definiti dall'utente creando uno schema e assegnandolo come tipo di un campo di un documento, creando così un sottodocumento. Nella Sezione 4.5 verrà presentato lo schema utilizzato per rappresentare i dati. [25]

Node-apac

Node-apac(Node Amazon Product Advertising Client) è un modulo di Node utilizzato come client per le *Amazon Product Advertising API*. Esso costruisce in automatico la struttura ed effettua delle chiamate Http alle API, convertendo i risultati che esse restituiscono da XML a JSON. [26]
Le *APA API* fanno parte del pacchetto di Amazon AWS e consentono l'accesso ai prodotti di Amazon e alle loro caratteristiche, oltre che al prezzo. Esse consentono di monetizzare i contenuti del proprio sito web, guadagnando delle commissioni sui prodotti comprati da Amazon attraverso lo stesso servizio.

Nella Sezione 4.4 verrà effettuato un approfondimento su Amazon AWS e sulla struttura delle richieste.

4.3.3 NW.js

NW.js, una volta chiamata *node-webkit*, è un app open source basata su Chromium e NodeJS che consente la virtualizzazione cross-platform[8] di un'applicazione web. Grazie alla sua struttura è, inoltre, possibile richiamare direttamente i moduli di Node direttamente, senza dover passare dal server. Essa consente anche le richieste ajax cross-domain senza alcun bisogno di modificare la struttura. [27]
Inizialmente nata nel *Intel Open Source Technology Center*, oggi è finanziata da *Intel* stessa e viene utilizzata per creare una moltitudine di applicazioni desktop. [28]

4.4 Amazon AWS

Gli *Amazon Web Services* sono una serie di servizi web presenti una piattaforma di cloud-computing di Amazon. Questi servizi sono definiti per 11 regioni geografiche in tutto il mondo. Amazon fornisce questi servizi per garantire una potenza di calcolo in cloud ad un prezzo economico. Questi servizi sono nati nel 2006 e consentono il loro utilizzo attraverso una serie di API REST interrogabili attraverso il protocollo *http*. [29]
Come già citato in precedenza, in questa applicazione è stato utilizzato il servizio di Product Advertising attraverso il modulo di node-apac.
L'inizializzazzione del componente avviene nel seguente modo:

```
1 OperationHelper = require('apac').OperationHelper;
2 var opHelper = new OperationHelper({
3     awsId: ID,
4     awsSecret: KEY,
5     assocId: TAG,
```

[8] Multipiattaforma. https://it.wikipedia.org/wiki/Multipiattaforma

```
6       xml2jsOptions: {
7           'explicitArray': false
8       },
9       version: '2013-08-01'});
```

ID, KEY e TAG rappresentano i parametri di configurazione per poter accedere al servizio. Seguono le richieste effettuate all'interno del programma.

4.4.1 Ricerca di un Prodotto

```
1 opHelper.execute('ItemSearch', {
2         'SearchIndex': 'All',
3         'Keywords': 'nvidia vga',
4         'ResponseGroup': 'ItemAttributes,Offers',
5         'ItemPage': 0
6     }, callback);
```

Attraverso questa chiamata è possibile ottenere tutti i prodotti, comprese le caratteristiche, che contengono le parole chiave nel titolo. Specificando il numero della pagina è possibile ottenere varie pagine. Questo perchè la richiesta restituisce massimo 10 elementi per pagina.

4.4.2 Recupero del Prezzo di un Prodotto

```
1 opHelper.execute('ItemLookup', {
2     'ItemId': ASIN,
3     'ResponseGroup': 'Offers'
4 }, callback);
```

Attraverso questa chiamata è possibile recuperare tutte le offerte presenti per un prodotto, identificato dall'ASIN[9]. In questo modo, effettuando questa richiesta in maniera continuata nel tempo, è stato possibile recuperare i prezzi di un prodotto, successivamente salvati nel database.

[9] *Amazon Standard Identification Number*. https://it.wikipedia.org/wiki/ASIN

4.5 MongoDB

Come detto in precedenza attraverso *mongoose* si è costruito lo schema dei dati.

Site

```
var SiteSchema = new Schema({
    name: String,
    url: String,
    date: {
        type: Date,
        default: Date.now
    }
});
```

Price

```
var PriceSchema = new Schema({
    site: {
        type: Schema.Types.ObjectId,
        ref: 'Site'
    },
    amount: {
        type: Number,
        min: 0
    },
    date: {
        type: Date,
        default: Date.now
    }
})
```

Item

```
var ItemSchema = new Schema({
    name: String,
    type: String,
    url: String,
```

```
    ASIN: String,
    date: {
        type: Date,
        default: Date.now
    },
    buyed: {
        type: Boolean,
        default: false
    },
    features: Schema.Types.Mixed,
    prices: [PriceSchema]
});
```

Da notare che `SiteSchema` non verrà utilizzato come modello, ma semplicemente come tipo definito dall'utente. Per rendere questi schemi dei modelli effettivi è stato eseguito il seguente codice:

```
var Site = mongoose.model('Site', SiteSchema);
var Price = mongoose.model('Price', PriceSchema);
var Item = mongoose.model('Item', ItemSchema);
```

Così facendo si sono introdotte le validazioni sui singoli modelli. Senza contare che utilizzando le operazioni di default si è semplificato l'inserimento, potendo omettere alcuni campi.
Utilizzando un campo di tipo `Objectid` è stato, inoltre, possibile generare delle relazioni, seppur deboli, fra i documenti. Queste relazioni, nonostante non presentino vincoli di integrità referenziale, vengono utilizzate da mongoose, durante le query, per recuperare il documento a cui il campo si riferesce. Tutto questo senza dover effettuare delle operazioni aggiuntive. Segue un esempio di questo funzionamento:

```
Item.find({}).populate('prices.site').exec(callback);
```

In questo modo è stato possibile recuperare i siti a cui ogni prezzo faceva riferimento, oltre che al prodotto stesso.
Un altro esempio delle semplificazioni offerte da mongoose è quello delle operazioni di aggregazione. Per esempio per contare il numero di prodotti presenti è bastato fare

```
Item.count({}, callback);
```

Inserimento, cancellazione e aggiornamento sono stati effettuati utilizzando i metodi descritti nell'Appendice A.

4.6 PredictionIO

Come detto in precedenza, per effettuare le raccomandazioni dei prodotti è stato utilizzato PredictioIO. In particolare si è partiti dal template di **Universal Recommender** e si è sviluppato un sistema per consigliare i prodotti in base ai propri acquisti e a quelli degli altri utenti. Siccome attualmente l'applicazione funziona solo con un utente sono stati inseriti dei dati pseudo casuali di sei utenti che hanno acquistato i prodotti, così da consentire al sistema di apprendimento di poter generare un modello di recommendation. Maggiore sarà la mole di dati che il sistema possiede, più accurato sarà il modello.
Di seguito sono riportate le richieste che sono state utilizzate per effettuare le operazioni all'interno dell'applicazione.

4.6.1 Aggiunta di un Utente

```
var req = http.request({
    host: "127.0.0.1",
    hostname: "localhost",
    port: "7070",
    method: "POST",
    path: "/events.json?accessKey=" + PIO_ACCESS_KEY,
    headers: {
        'Content-Type': 'application/json'
    }
}, callback);

req.write(JSON.stringify({
        "event": "$set",
        "entityType": "user",
        "entityId": user
```

```
16      }));
17  req.end();//$
```

In questo caso `user` rappresenta il nome dell'utente da inserire. In caso di nomi duplicati il sistema restituisce un errore.

4.6.2 Aggiunta di un Prodotto

```
1  var req = http.request({
2      host: "127.0.0.1",
3      hostname: "localhost",
4      port: "7070",
5      method: "POST",
6      path: "/events.json?accessKey=" + PIO_ACCESS_KEY,
7      headers: {
8          'Content-Type': 'application/json'
9      }
10 }, callback);
11
12 req.write(JSON.stringify({
13     "event": "$set",
14     "entityType": "item",
15     "entityId": item.id,
16     "properties": {
17         "categories": [item.type]
18     }));
19 req.end();//$
```

In questo caso `item.id` e `item.type` rappresentano rispettivamente l'id del documento relativo al prodotto e la categoria di appartenenza.

4.6.3 Acquisto di un Prodotto da Parte dell'Utente

```
1  var req = http.request({
2      host: "127.0.0.1",
3      hostname: "localhost",
4      port: "7070",
5      method: "POST",
6      path: "/events.json?accessKey=" + PIO_ACCESS_KEY,
7      headers: {
```

```
8           'Content-Type': 'application/json'
9       }
10  }, callback);
11
12  req.write(JSON.stringify({
13          "event": "view",
14          "entityType": "user",
15          "entityId": user,
16          "targetEntityType": "item",
17          "targetEntityId": item.id
18      }));
19  req.end();
```

Come per il caso precedente `item.id` rappresenta l'id del documento riferito a quel prodotto.

4.6.4 Recupero dei Prodotti Raccomandati

```
1   var req = http.request({
2       host: "127.0.0.1",
3       hostname: "localhost",
4       port: "8000",
5       method: "POST",
6       path: "/queries.json",
7       headers: {
8           'Content-Type': 'application/json'
9       }
10  }, callback);
11
12  req.write(JSON.stringify({
13          "items": items,
14          "num": number
15      }));
16  req.end();
```

In questo caso `items` e `number` si riferiscono rispettivamente all'insieme dei prodotti di cui trovare la raccomandazione e il numero di prodotti che il sistema deve restituire.

Capitolo 5

Analisi dei Risultati

In questo capitolo viene analizzato il modello utilizzato per la previsione dei prezzi. Successivamente viene rappresentato l'errore relativo al prezzo previsto e quello reale.

5.1 Interpolazione Polinomiale

L'interpolazione polinomiale è l'interpolazione di una serie di valori attraverso una funzione polinomiale passante per i punti dati. Si è scelta questa interpolazione, rispetto alle altre, vista la natura non lineare della variazione dei prezzi. Questa è un caso particolare di approssimazione attraverso i polinomi che tende a trovare un polinomio che approssima i punti dati con l'errore più piccolo possibile. Per un eventuale approfondimento si può visionare l'approfondimento di *Diego Alberto* "Regressione Polinomiale". [30]

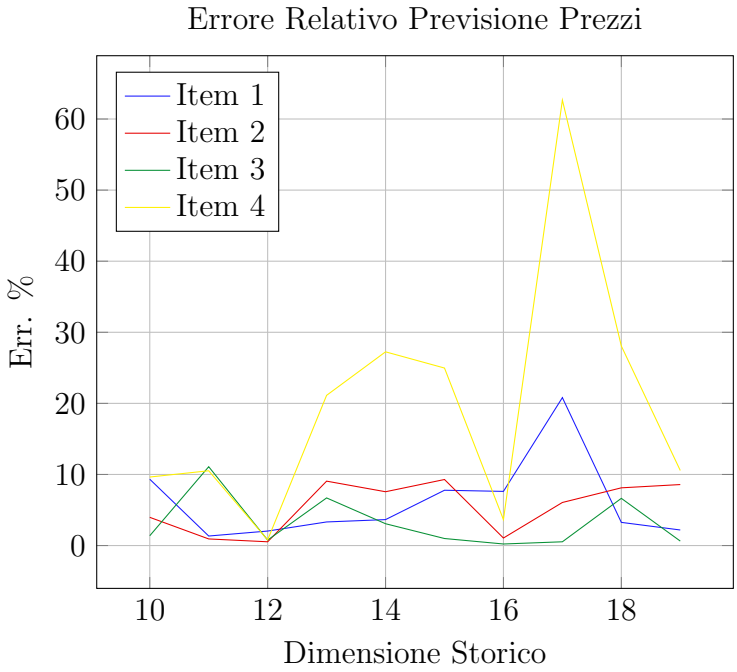
5.2 Errore sulla Previsione dei Prezzi

Per analizzare l'errore dell'interpolazione polinomiale si è studiato il prezzo di quattro prodotti di categorie diverse presenti su *amazon.com*. Su ogni prezzo è stato calcolato l'errore relativo percentuale. La Tabella 5.1 che mo-

stra l'errore sulla previsione dei prezzi in base al numero di prezzi presenti nello storico.

Err. %	**Errore Relativo per Prodotto**									
	10	11	12	13	14	15	16	17	18	19
Item 1	9.33	1.34	2.04	3.32	3.65	7.78	7.62	20.81	3.26	2.19
Item 2	3.97	0.93	0.53	9.06	7.57	9.29	1.06	6.06	8.12	8.58
Item 3	1.68	11.09	0.77	6.70	3.07	0.99	0.22	0.53	6.64	0.63
Item 4	9.64	10.52	0.80	21.13	27.25	24.97	3.65	62.66	28.07	10.56

Tabella 5.1: Errore relativo sulla previsione dei prezzi



Come si può notare dalla tabella gli errori variano molto da prodotto a prodotto. Le celle evindeziate in `rosso` rappresentano delle variazioni del prezzo dell'oggetto di oltre il 15%, che in alcuni casi, come nel caso del Item 4, arrivano a toccare il 60%. Come si può notare dalla tabella l'errore si alza improvvisamente e tende ad abbassarsi costantemente nelle rilevazioni successive, nel caso in cui non avvenga un'altra variazione rilevante oppure

non inizia una serie costante.
Le celle evidenziate in blu rappresentano una serie di prezzi costanti che quindi non subiscono variazioni. In questi periodi l'errore della previsione varia in due modi: nel caso in cui ci sia stata una variazione significativa esso tende ad aumentare per poi ridursi nuovamente mentre, nel caso opposto, esso tende subito a ridursi.

5.2.1 Errore Medio per Fascia di Prezzo

Da notare il fatto che gli Item hanno avuto delle fluttuazioni dei prezzi sempre all'interno di un intervallo di prezzi limitato, disposto nel modo seguente:

Item 1 : $100\$ - 200\$$ con un errore medio di 6.13%.

Item 2 : $300\$ - 400\$$ con un errore medio di 5.51%.

Item 3 : $400\$ - 500\$$ con un errore medio di 3.23%.

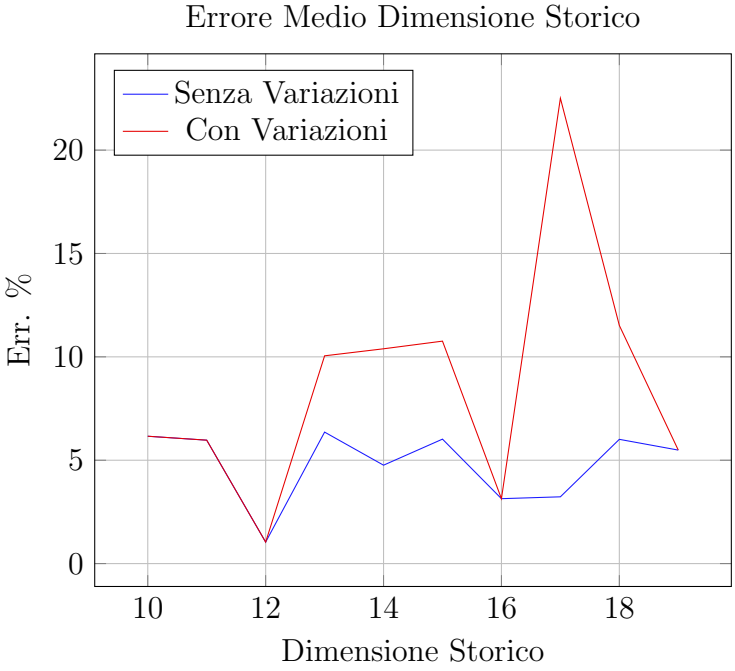
Item 4 : $1\$ - 100\$$ con un errore medio di 19.93%.

Si noti come nelle fasce di prezzo più alte l'errore tende ad essere, mediamente, minore rispetto alle fasce di prezzo più basse. Questo è spiegato dal fatto che la variazione relativa di prezzo risulta maggiore nelle fasce più basse. Basti pensare che una variazione di 50$ su 100$ si traduce in una variazione del 50%, mentre la stessa variazione su 1000$ si traduce in una variazione del 5%.

Err. %	**Errore Medio Storico**									
	10	11	12	13	14	15	16	17	18	19
Con Variazioni	6.16	5.97	1.04	10.05	10.39	10.76	3.14	22.51	11.52	5.49
Senza Variazioni	6.16	5.97	1.04	6.36	4.76	6.02	3.14	3.23	6.01	5.49

Tabella 5.2: Errore relativo medio in base alla dimensione dello storico

5.2.2 Errore Medio per Dimensione dello Storico



Come si vede dalla Tabella 5.2, se non si tiene conto delle variazioni significative di prezzo, l'errore medio risulta attorno al 5%. Mentre nel caso in cui si tengano in considerazione le variazioni esso tende a crescere a oltre l'8%.

Non esiste un vero e proprio modello di previsione che si possa utilizzare per prevedere il prezzo di un prodotto. Se si pensa ai periodi di sconto, che impongono variazioni consistenti, o ai rialzi, dovuti alla domanda crescente, si comprende come queste componenti, che influiscono notevolmente sul prezzo, siano di difficile previsione.

Nonostante questo, si è dimostrato che utilizzando questo tipo di interpolazione è possibile individuare il prezzo del prodotto con uno scarto di circa il 5%. Errore che risulta rilevante ma che, tuttavia, consente di identificare l'andamento dei prezzi nel breve periodo.

Conclusioni

In questa tesi si è introdotto l'ambiente dei Big Data, con i database NoSQL per la loro memorizzazione e i sistemi di Machine Learning per la loro analisi. Si è effettuata una rassegna su due servizi che rappresentano i due ambienti, i quali sono stati utilizzati per la creazione di un programma di prediction e recommendation al fine di verificare la loro effettiva usabilità e di analizzare i dati sull'accuratezza.

Nella prima parte si è parlato di come l'aumento della mole di dati in circolazione abbia portato allo sviluppo di nuove tecnologie per la loro memorizzazione e alla nascita del termine *big data* per riferirsi ad essi. Sono state introdotte le caratteristiche principali, insieme alle motivazioni per cui non è possibile utilizzare i sistemi di memorizzazione canonici per la loro elaborazione. Si è introdotto poi l'insieme dei database non relazionali, in particolare le loro caratteristiche e le motivazioni dietro alla loro nascita, effettuando una rassegna su MongoDB. Infine si è affrontato il mondo del Machine Learning, spiegando le caratteristiche e le necessità economiche che risiedono dietro all'utilizzo di questi sistemi. Si è poi effettuata una rassegna su PredictionIO.

Nella seconda parte sono state utilizzate le tecnologie approfondite in precedenza per lo sviluppo di un'applicazione di previsione dei prezzi e suggerimento di prodotti simili chiamata *Item Price Watcher*. Quest'applicazione, sviluppata attraverso le ultime tecnologie web e di virtualizzazione, ha messo in evidenza le metodologie da applicare al fine di integrare un'applicazione con database NoSQL e sistemi di Machine Learning. Infine è stata effetuata un'analisi dell'errore prodotto dall'interpolazione polinomiale sullo storico

presente, prendendo in esame il prezzo di quattro prodotti di categorie e fasce di prezzo differenti.

Nonostante le API di Amazon siano state uno strumento di totale supporto per la raccolta dei dati, esse presentavano alcune limitazioni, volute e non, che hanno rallentato molto la creazione dello storico. Prima fra tutte, l'impossibilità di rilevare i prezzi di un prodotto nell'immediato passato. A causa di questa limitazione è stato implementato un sistema per la raccolta giornaliera automatica dei prezzi dei beni relativi alla sorgente di *amazon.com*. Per questo, la dimensione dello storico dei prezzi è risultata molto limitata, soprattutto in fase di analisi dell'errore sulla previsione. Questa dimensione limitata ha portato a comportamenti anomali che sono stati evidenziati nella Tabella 5.1 che riportava l'errore relativo della previsione in base al prezzo.
Un'altra problematica riscontrata è stata quella dell'apprendimento di PredictionIO per quanto riguardava il modello di suggerimento dei prodotti simili. Siccome esso è basato sull'attività di più utenti, e l'applicazione risulta invece ad utente singolo nello stato attuale, sono stati inseriti degli eventi pseudo-casuali di acquisto da parte di utenti fittizzi. Questi eventi, tuttavia, hanno portato a raccomandazioni talvolta banali. Attraverso la raccolta di dati derivanti dall'utilizzo del sistema da più utenti, sarebbe possibile effettuare un'apprendimento più preciso al fine di ricevere dei risultati più significativi.

Considerando la natura dell'applicazione, ossia un'applicazione web virtualizzata, sarebbe interessante generare la piattaforma web insieme all'applicativo, gestendo il caso della multiutenza. In questo scenario la quantità di dati potrebbe crescere al punto tale da poter apprezzare le caratteristiche di MongoDB e PredictionIO. Così facendo la conseguente analisi dell'errore sulle previsioni sarebbe più precisa e quindi potrebbe essere possibile verificare l'accuratezza del modello di regressione polinomiale.
Un'altra idea di sviluppo futuro è quella di rendere l'applicazione social, gestendo oltre alla multiutenza, anche le relazioni fra gli utenti. In questo modo un utente, se desidera, può condividere con i suoi amici la lista dei

prodotti che ha acquistato, basando il sistema di sugerimento dei prodotti principalmente sulle scelte fatte dagli amici, poi su quelle degli utenti generici. Questo, insieme all'implementazione di nuovi servizi per l'aggiunta automatica dei prodotti, sono gli sviluppi fondamentali per il futuro di quest'applicazione.

Da questo studio si può quindi affermare che lo sviluppo di applicativi attraverso le ultime tecnologie per l'analisi e la memorizzazione dei dati, non implica un aumento eccessivo della complessità in fase di sviluppo, anzi certe volte lo rende più fluido. Tuttavia ciò non significa che i database relazionali siano stati soppiantati da questo nuovo approccio. Esistono ancora alcune realtà, soprattutto piccole, in cui la complessità è tale da preferire gli approcci classici. Alcune volte viene usato anche un ibrido delle due tecnologie, questo perchè i database NoSQL non sono un sostituto, bensì un'alternativa.

Appendice A

MongoDB

Le operazioni elencate di seguito sono state effettuate attraverso la MongoDB Shell.

Inserimento

```
1 //Inserimento semplice
2 db.foo.insert({"bar" : "baz"})
3 //Inserimento batch
4 db.foo.batchInsert([{"_id" : 0}, {"_id" : 1}, {"_id" : 2}])
```

Cancellazione

```
1 //Cancellazione di tutti gli elementi nella collection
2 db.foo.remove()
3 //Cancellazione degli elementi che rispettano la query
4 db.foo.remove({"opt-out" : true})
```

Aggiornamento

```
1 //Documento d'esempio
2 {
3     "_id" : ObjectId("4b2b9f67a1f631733d917a7a"),
4     "name" : "joe",
```

```
5       "friends" : 32,
6       "enemies" : 2
7   }
8
9   //Sostituzione di un documento
10  var joe = db.users.findOne({"name" : "joe"});
11  joe.relationships = {"friends" : joe.friends, "enemies" : joe
        .enemies};
12  joe.username = joe.name;
13  delete joe.friends;
14  delete joe.enemies;
15  delete joe.name;
16  db.users.update({"name" : "joe"}, joe);
17  //Documento Aggiornato
18  {
19      "_id" : ObjectId("4b2b9f67a1f631733d917a7a"),
20      "username" : "joe",
21      "relationships" : {
22          "friends" : 32,
23          "enemies" : 2
24      }
25  }
```

Modificatori Aggiornamento

```
1   db.users.update({"_id" : ObjectId("4b253b067525f35f94b60a31")
        }, {"$set" : {"favorite book" : "War and Peace"}})
2   db.users.update({"name" : "joe"}, {"$unset" : {"favorite book
        " : 1}})
3   db.games.update({"game" : "pinball", "user" : "joe"}, {"$inc"
        : {"score" : 50}})
4   db.games.update({"game" : "pinball", "user" : "joe"}, {"$inc"
        : {"score" : -10}})
5   db.movies.find({"genre" : "horror"}, {"$push" : {
6       "top10" : {
7           "$each" : [{
8               "name" : "Nightmare on Elm Street",
9               "rating" : 6.6
```

```
            }, {
            "name" : "Saw",
            "rating" : 4.3
            }],
        "$slice" : -10,
        "$sort" : {"rating" : -1}
        }
    }
})
db.users.update({"_id" : ObjectId("4b2d75476cc613d5ee930164")
    },{"$addToSet" : {
     "emails" : "joe@hotmail.com"
     }
})
db.lists.update({}, {"$pull" : {"todo" : "laundry"}})
```

Query

```
//Recupero dei documenti che rispecchiano la query
db.users.find({"age" : 27})
//Recupero dei campi dei documenti che rispecchiano la query
db.users.find({}, {"username" : 1, "email" : 1})
{
    "_id" : ObjectId("4ba0f0dfd22aa494fd523620"),
    "username" : "joe",
    "email" : "joe@example.com"
}
//Espressioni regolari
db.users.find({"name" : /joe/i})
```

Modificatori Query

```
db.raffle.find({"ticket_no" : {"$in" : [725, 542, 390]}})
db.raffle.find({"ticket_no" : {"$nin" : [725, 542, 390]}})
db.raffle.find({"$or" : [{"ticket_no" : 725}, {"winner" :
    true}]})
db.users.find({"id_num" : {"$not" : {"$mod" : [5, 1]}}})
db.users.find({"age" : {"$lt" : 30, "$gt" : 20}})
```

```
6 db.food.find({fruit : {$all : ["apple", "banana"]}})
7 db.food.find({"fruit" : {"$size" : 3}})
8 db.blog.posts.findOne(criteria, {"comments" : {"$slice" :
    10}})
```

Bibliografia

[1] Wikipedia. Business intelligence — wikipedia, l'enciclopedia libera, 2015. URL https://it.wikipedia.org/w/index.php?title=Business_intelligence&oldid=75542021. [Online; accessed 12-novembre-2015].

[2] Michael de Waal-Montgomery. World's data volume to grow 40% per year & 50 times by 2020: Aureus, Gennaio 2015. URL http://e27.co/worlds-data-volume-to-grow-40-per-year-50-times-by-2020-aureus-20150115-2/. [Online; accessed 11-november-2015].

[3] Gaurav Vaish. *Getting Started with NoSQL*. Packt Publishing, Marzo 2013. ISBN 1849694982.

[4] Wikipedia. Nosql — wikipedia, l'enciclopedia libera, 2015. URL https://it.wikipedia.org/w/index.php?title=NoSQL&oldid=74661602. [Online; accessed 20-novembre-2015].

[5] Knut Haugen. A brief history of nosql, 2010. URL https://blog.knuthaugen.no/2010/03/a-brief-history-of-nosql.html. [Online; accessed 20-november-2015].

[6] Matt Asay. Nosql databases eat into the relational database market, 2015. URL http://www.techrepublic.com/article/nosql-databases-eat-into-the-relational-database-market/. [Online; accessed 23-november-2015].

[7] bi-insider.com. Column and row based database storage, 2014. URL `http://bi-insider.com/business-intelligence/column-and-row-based-database-storage/`. [Online; in data 23-novembre-2015].

[8] John Ellis. noSQL - Document-Oriented, 2015. URL `http://www.johnmellis.com/blogs/nosql-document-oriented`. [Online; accessed 26-November-2015].

[9] Kim Eastlake. Data Analytics- NoSQL, 2015. URL `http://community.mis.temple.edu/kimeastlake/data-analytics-nosql/`. [Online; accessed 26-November-2015].

[10] Redislabs. Keys pattern, 2011. URL `http://redis.io/commands/keys`. [Online; accessed 26-November-2015].

[11] Roberto Gimeno. Your Data and NOSQL: Graph Databases, 2013. URL `http://www.gimeno.eu/2013/10/23/your-data-and-nosql-graph-databases/`. [Online; accessed 26-November-2015].

[12] Wikipedia. Adaptive website — wikipedia, the free encyclopedia, 2015. URL `https://en.wikipedia.org/w/index.php?title=Adaptive_website&oldid=677766311`. [Online; accessed 25-November-2015].

[13] Wikipedia. Recommender system — wikipedia, the free encyclopedia, 2015. URL `https://en.wikipedia.org/w/index.php?title=Recommender_system&oldid=692150590`. [Online; accessed 25-November-2015].

[14] Alessandro Rezzani. *Big Data. Architettura, tecnologie e metodi per l'utilizzo di grandi basi di dati.* PerCorsi di Studio. Apogeo Education, Febbraio 2014. ISBN 8838789894.

[15] Kristina Chodorow. *MongoDB: The Definitive Guide*. O'Reilly Media, 2013. ISBN 1449344682.

[16] Conan Zhang. Mongodb autosharding, 2013. URL `http://blog.fens.me/mongodb-shard/`. [Online; accessed 28-november-2015].

[17] Solid IT. Db-engines ranking, 2015. URL `http://db-engines.com/en/ranking`. [Online; accessed 28-november-2015].

[18] Wikipedia. Mongodb — wikipedia, l'enciclopedia libera, 2015. URL `http://it.wikipedia.org/w/index.php?title=MongoDB&oldid=76063224`. [Online; accessed 28-november-2015].

[19] PredictionIO. URL `https://prediction.io/`. [Online; accessed 28-november-2015].

[20] Bootstrap. URL `http://getbootstrap.com/`. [Online; accessed 2-december-2015].

[21] jQuery Foundation. jquery. URL `https://jquery.org/`. [Online; accessed 2-december-2015].

[22] Tom Alexander. regression.js. URL `https://github.com/Tom-Alexander/regression-js`. [Online; accessed 2-december-2015].

[23] Olly Smith. morris.js. URL `http://morrisjs.github.io/morris.js/index.html`. [Online; accessed 2-december-2015].

[24] Node.js Foundation. nodejs. URL `https://nodejs.org/en/`. [Online; accessed 30-november-2015].

[25] LearnBoost. mongoose. URL `http://mongoosejs.com/index.html`. [Online; accessed 2-december-2015].

[26] Dustin McQuay. Node-apac. URL `https://github.com/dmcquay/node-apac`. [Online; accessed 2-december-2015].

[27] inc Amazon. Amazon product advertising api. URL `https://affiliate-program.amazon.com/gp/advertising/api/detail/main.html`. [Online; accessed 2-december-2015].

[28] Intel. Nw.js. URL `https://github.com/nwjs/nw.js`. [Online; accessed 2-december-2015].

[29] inc Amazon. Amazon web services. URL `https://aws.amazon.com/it/`. [Online; accessed 2-december-2015].

[30] Diego Alberto. *Regressione Polinomiale*. matematicamente.it, Aprile 2005.

[31] David Hows, Peter Membrey, and Eelco Plugge. *MongoDB Basics*. Apress, 2014. ISBN 1484208953.

[32] Wikipedia. Machine learning — wikipedia, the free encyclopedia, 2015. URL `https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=691986240`. [Online; accessed 25-November-2015].

Ringraziamenti

Potrei scrivere intere pagine sulle persone che vorrei ringraziare e che mi hanno permesso trovarmi qui oggi. La prima persona da ringraziare è sicuramente il Dott. Marco Di Felice, che ha subito visto le potenzialità del mio progetto e mi ha spronato a svilupparlo, rendendolo l'argomento principale di questa tesi, nonostante io non lo avessi neanche preso in considerazione.
Subito dopo voglio ringraziare i miei genitori che mi hanno permesso di intraprendere questo percorso, nonostante le avversità riscontrate, soprattutto della mia attitudine a non ascoltarli. Vi voglio bene perchè con me non vi siete mai arresi e mi avete insegnato a mettermi in gioco per ottenere ciò che voglio. Grazie.
Un grazie a mio fratello Simone che mi vede come un esempio da seguire ma che molto spesso non considero come dovrei. Spero ora di passare del tempo con te. Grazie.
Un grazie speciale va anche a tutti i miei familiari, soprattutto i miei cari nonni che mi sopravvalutano come ogni buon nonno dovrebbe fare con il proprio nipote e per questo non smetterò mai di volergli bene.
Un altro grande ringraziamento va a Bianca, la mia ragazza, che mi ha sempre supportato e sopportato. Mi ha dato la determinazione per riuscire a terminare il mio percorso. Mi ha fatto capire ciò che voglio. Mi ha fatto rinascere. Per te ci sarà sempre un posto speciale nella mia vita.
Un grazie va anche a tre persone diventate parte integrante della mia vita. Grazie Marco(*Fusk*), perchè fin da piccoli siamo stati insieme e siamo cresciuti insieme. Aiutandoci l'un l'altro.

Grazie Mattia(*Rava*), che mi hai accompagnato nel mio percorso universitario e non; e grazie delle serate passate in appartamento a Bologna, non le dimenticherò mai.
Grazie Marco(*Bacca*), perchè in pochi anni mi hai dimostrato cosa significhi la fratellanza e mi hai sempre protetto in tutte le situazioni che accadevano, spero un giorno di poter ricambiare il favore.
Detto questo ringrazio tutte le persone che sono venute a vedermi e quelle che non sono venute le ringrazio in ugual modo. Grazie per le belle parole, i complimenti, gli insulti, le liti, le discussioni, le risate, i discorsi profondi e meno profondi. Se oggi sono come sono è grazie a tutti voi.