



Progetto di Large-Scale and Multi-Structured Databases

# **Discovery Companies Application**

Realizzato da:

Matteo Castrignano  
Francesco Marabotto  
Leo Maltese

# 1 SOMMARIO

---

2	Introduzione .....	3
3	Principali attori dell'applicazione .....	4
3.1	Utente Anonimo .....	4
3.2	Utente .....	4
3.3	Utente Professionale .....	4
3.4	Amministratore .....	4
4	Requisiti funzionali .....	5
4.1	Utente Anonimo .....	5
4.2	Utente .....	5
4.3	Utente Professionale .....	5
4.4	Amministratore .....	5
5	Requisiti non funzionali .....	7
6	Definizione delle classi .....	7
6.1	Definizione degli attributi .....	9
7	Architettura del software .....	13
8	Design dell'applicazione .....	15
8.1	MongoDB Design .....	15
8.1.1	Organizzazione dei documenti e delle collezioni .....	15
8.1.2	Operazioni CRUD – MongoDB .....	18
8.2	Neo4j Design .....	20
8.1.3	Nodi e Relazioni .....	21
8.1.4	Operazioni CRUD – Neo4j .....	23
9	Implementazione .....	25
9.1	Implementazione delle operazioni crud - MongoDB .....	25
9.2	Analytics Functions - MongoDB .....	27
9.3	Implementazione delle operazioni crud – Neo4j .....	30
9.4	Analytics Functions – Neo4j .....	30
9.5	Struttura del repository .....	35
10	Generazione del database .....	36
11	Test eseguiti e analisi statistiche .....	37
11.1	Indici su MongoDB .....	37
12	Manuale d'uso .....	39

## 2 INTRODUZIONE

---

L'applicazione sviluppata fornisce un servizio per utenti che vogliono investire e nello stesso tempo scoprire nuove realtà nel campo dei mercati azionari. Tale applicazione prende il nome di "Discovery Companies Application".

In questa applicazione sono presenti informazioni e dati relativi ai mercati azionari, compresi quelli delle principali aziende che appartengono agli indici "Nasdaq" e "Dow Jones".

L'applicazione è pensata per due categorie principali di utenti: gli utenti professionali e gli utenti normali.

Gli utenti professionali sono utenti i quali hanno particolari competenze e conoscenze nell'ambito dei mercati finanziari. Dopo essersi registrati e aver effettuato il login, possono scrivere dei report riguardo alle aziende presenti nell'applicazione.

Un utente normale è un utente che, una volta che si è registrato e ha eseguito il login, può esplorare le informazioni riguardo alle aziende che sono presenti e leggere i report che gli utenti professionali hanno scritto.

Un utente, una volta che ha scoperto un'azienda che può interessargli, può aggiungerla a una propria watchlist, per poterne seguire l'andamento ed eventualmente leggere i report ad essa associati.

Per scoprire nuove aziende, un utente può iniziare a seguire altri utenti in modo da vedere quali sono le aziende che gli altri utenti hanno nella watchlist.

Lo scopo principale dell'applicazione è proprio questo: consentire agli utenti di potersi seguire a vicenda e di poter scoprire altre aziende.

Un utente professionale può essere valutato dagli utenti normali in base ai report che ha scritto; un utente normale può infatti dare una valutazione all'utente professionale tramite un voto da 1 a 5.

I dati relativi all'applicazione sono interamente gestiti dall'amministratore, ad esempio può eliminare alcuni utenti ed aggiornare i dati delle aziende.

## **3 PRINCIPALI ATTORI DELL'APPLICAZIONE**

---

In questa sezione sono descritti i principali attori/utenti che utilizzano l'applicazione.

### **3.1 UTENTE ANONIMO**

È un utente che non si è registrato o che non ha ancora effettuato il login.

### **3.2 UTENTE**

È un utente che ha effettuato il login all'applicazione. Può iniziare a seguire gli altri utenti, può prendere visione delle informazioni delle aziende presenti nell'applicazione, può leggere i report scritti riguardo alle aziende e può valutare gli utenti professionali.

### **3.3 UTENTE PROFESSIONALE**

È un utente con particolari competenze nell'ambito dei mercati finanziari: può essere ad esempio un analista, un operatore di borsa, un esperto di finanza. Una volta che ha effettuato il login, può scrivere dei report riguardo alle aziende presenti nell'applicazione in modo da farli leggere agli utenti. Inoltre, può eseguire tutte le operazioni consentite dagli utenti normali, con un solo limite: può seguire solamente le aziende e non altri utenti.

### **3.4 AMMINISTRATORE**

È il gestore dell'applicazione: si preoccupa di mantenere i dati aggiornati, gestisce l'aggiunta e la rimozione di aziende e la rimozione degli utenti dall'applicazione.

## 4 REQUISITI FUNZIONALI

---

In questa sezione sono descritti i requisiti funzionali dell'applicazione:

### 4.1 UTENTE ANONIMO

- a. Se l'utente non è registrato, allora può registrarsi sull'applicazione se non è già in possesso di un account.
- b. Se l'utente è registrato ma non ha ancora effettuato il login, può farlo inserendo username e password.

### 4.2 UTENTE

- a. Un utente può eseguire il login, se si è già registrato in precedenza, oppure può effettuare il logout se è già loggato.
- b. Un utente può cercare un'azienda.
- c. Un utente può prendere visione delle informazioni dell'azienda sulla quale ha effettuato una ricerca.
- d. Un utente può assegnare un indice di gradimento ad un utente professionale (da 1 a 5).
- e. Un utente può iniziare a seguire un'azienda, che verrà quindi aggiunta alla sua watchlist.
- f. Un utente può iniziare a seguire un altro utente.
- g. Un utente può osservare le watchlist di altri utenti.
- h. Un utente può vedere le proprie informazioni e aggiornarle.
- i. Un utente può modificare la propria valutazione riguardo a un utente professionale.
- j. Un utente può visionare i report scritti riguardo una particolare azienda oppure i report scritti da un particolare utente professionale.

### 4.3 UTENTE PROFESSIONALE

Un utente professionale può eseguire le stesse operazioni degli utenti normali, col solo limite di non poter seguire altri utenti di qualsiasi tipo, in aggiunta:

- a. Un utente professionale può scrivere dei report riguardo una o più aziende presenti nell'applicazione.

### 4.4 AMMINISTRATORE

L'utente amministratore dispone di tutte le funzionalità dell'applicazione, in aggiunta:

- a. Un amministratore può aggiungere o rimuovere aziende dall'applicazione.
- b. Un amministratore può rimuovere utenti (normali o professionali) dall'applicazione.
- c. Un amministratore può aggiungere, aggiornare e rimuovere dati relativi alle aziende presenti nell'applicazione.

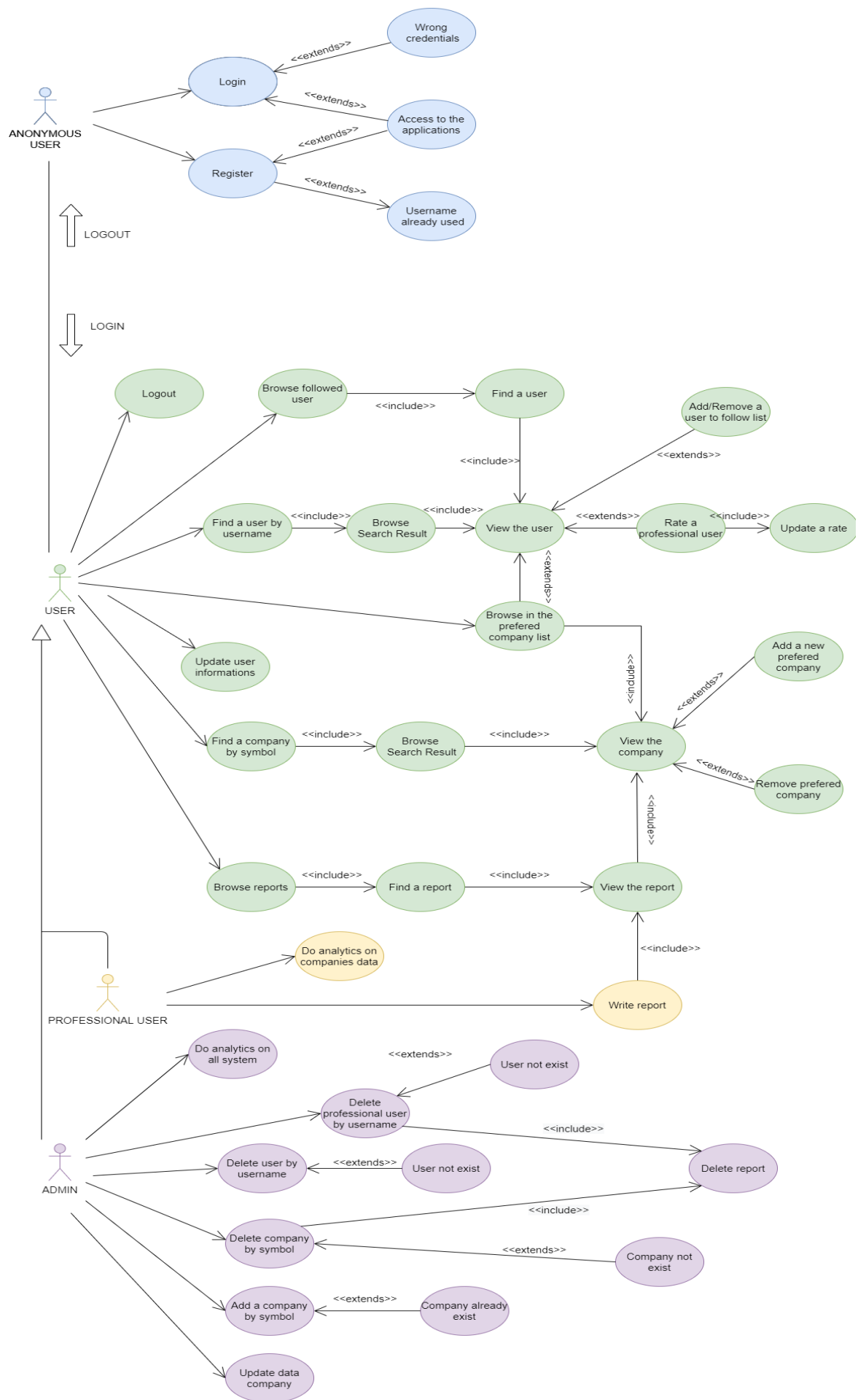


Figura 1 Diagramma Use-Case

## 5 REQUISITI NON FUNZIONALI

---

- *Availability*: il sistema deve garantire una buona disponibilità dei dati.
- *Consistency*: l'applicazione deve avere i dati sempre in uno stato consistente, almeno per quanto riguarda i dati finanziari. Essendo infatti un'applicazione che tratta di finanza, avere i dati sempre disponibili e consistenti è un requisito molto importante per garantire un buon servizio all'utenza.
- *Response speed*: l'applicazione deve garantire che gli utenti e gli utenti professionali possano prendere visione delle informazioni di un'azienda in un tempo breve.
- *Security*: l'applicazione garantisce un livello minimo di sicurezza, dato da una login tramite username e password.
- *Usability*: l'applicazione deve essere facilmente utilizzabile dai vari utenti.

## 6 DEFINIZIONE DELLE CLASSI

---

CLASSE	DESCRIZIONE
<b>User</b>	È il principale attore dell'applicazione. Può seguire altri utenti, può seguire utenti professionali, può seguire aziende, può leggere i report e può dare una valutazione agli utenti professionali
<b>ProfessionalUser</b>	È un utente con competenze nell'ambito dei mercati finanziari: può seguire aziende e può scrivere dei report che verranno poi letti dagli utenti normali
<b>Admin</b>	È il gestore dell'applicazione. Può aggiungere e rimuovere utenti normali e utenti professionali e può aggiungere, aggiornare e modificare i dati relativi alle aziende
<b>Company</b>	È un'azienda con varie informazioni reperibili da parte degli utenti normali e degli utenti professionali.
<b>Summary</b>	Dati e informazioni riguardo l'azienda, relativi all'ultima settimana in borsa
<b>History</b>	Dati storici, con frequenza settimanale, riguardo a un'azienda.
<b>Report</b>	Informazioni e analisi riguardo un'azienda. Sono scritti dagli utenti professionali.

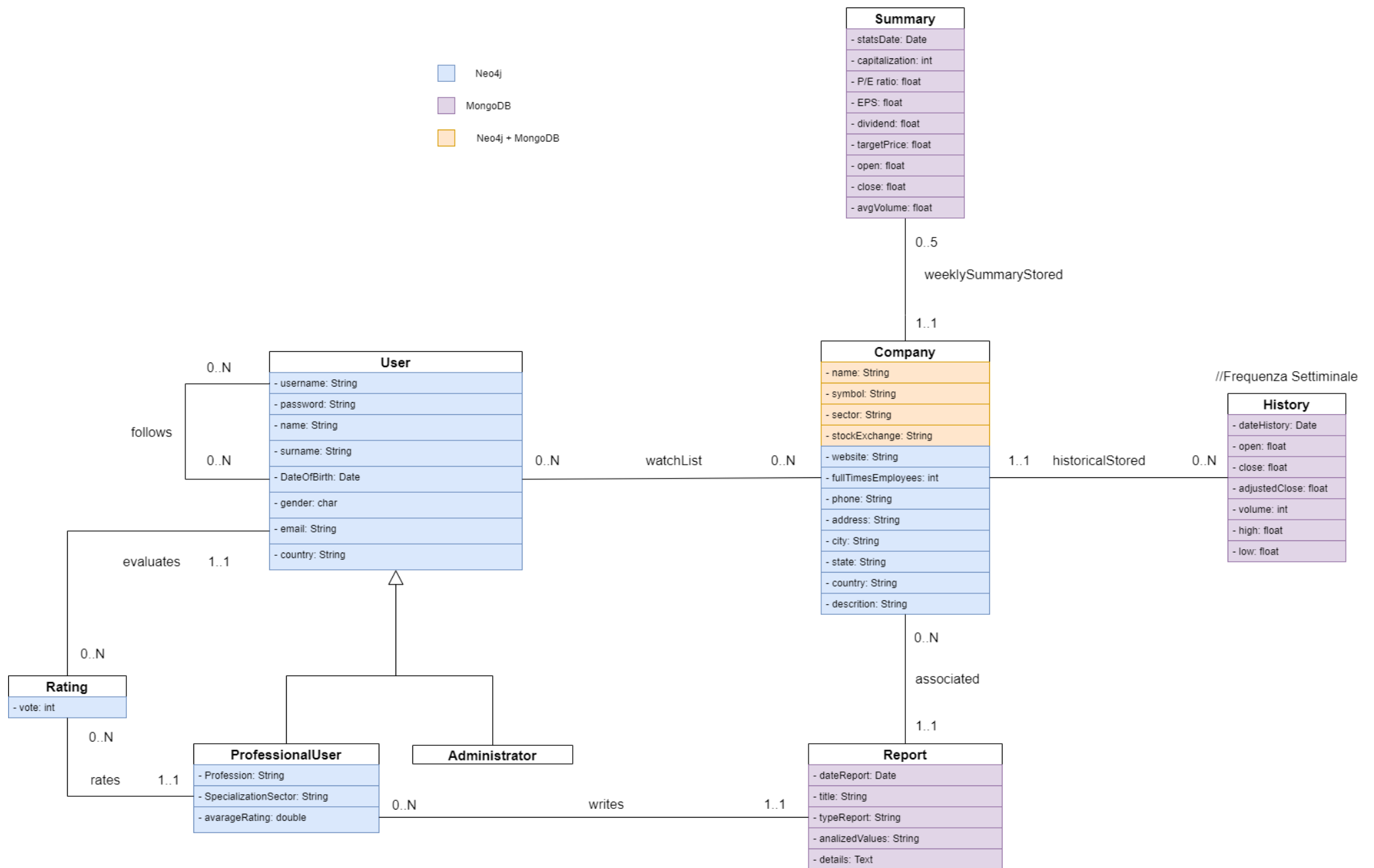


Figura 2 Diagramma delle classi



## 6.1 DEFINIZIONE DEGLI ATTRIBUTI

**Classe: User**

ATTRIBUTO	TIPO	DESCRIZIONE
<b>username</b>	String	Nickname scelto dall'utente durante la registrazione. Identifica univocamente l'utente
<b>password</b>	String	Password scelta dall'utente durante la registrazione. Insieme all'username, consente all'utente di effettuare il login all'applicazione
<b>name</b>	String	Nome dell'utente
<b>surname</b>	String	Cognome dell'utente
<b>dateOfBirth</b>	String	Data di nascita dell'utente
<b>gender</b>	char	Sesso dell'utente
<b>email</b>	String	Contatto email dell'utente
<b>country</b>	String	Paese di origine dell'utente
<b>typeUser</b>	int	Tipologia dell'utente, serve per distinguere facilmente un utente da un utente professionale e dall'amministratore. Per l'utente il valore di questo campo è 2, per il professional user è 1, mentre per l'admin è uguale a 0

**Classe: ProfessionalUser**

(struttura come User, con in più i seguenti attributi)

ATTRIBUTO	TIPO	DESCRIZIONE
<b>profession</b>	String	Professione svolta dall'utente professionale
<b>specializationSector</b>	String	Settore nel quale l'utente professionale è specializzato

<b>averageRating</b>	double	Media dei voti che gli utenti normali hanno attribuito all'utente professionale
----------------------	--------	---

### Classe: Administrator

(stessa struttura di User)

### Classe Company

ATTRIBUTO	TIPO	DESCRIZIONE
<b>symbol</b>	String	Identifica univocamente l'azienda all'interno dell'applicazione
<b>name</b>	String	Nome dell'azienda
<b>exchange</b>	String	E' l'indice di borsa nel quale il titolo è scambiato
<b>sector</b>	String	Settore nel quale l'azienda si inserisce (es. tecnologico, sanitario etc)
<b>fullTimeEmployees</b>	int	Numero di dipendenti dell'azienda
<b>description</b>	String	Breve descrizione dell'azienda
<b>city</b>	String	Città dove è situata la sede principale dell'azienda
<b>phone</b>	String	Numero di telefono dell'azienda
<b>state</b>	String	Stato (americano) dove è situata la sede principale dell'azienda
<b>country</b>	String	Nazione dove è situata la sede principale dell'azienda
<b>address</b>	String	Indirizzo della sede principale dell'azienda
<b>mostProfitablePeriod</b>	String	Periodo di tempo nel quale l'azienda ha avuto un maggior incremento

### Classe: Summary

ATTRIBUTO	TIPO	DESCRIZIONE
<b>symbol</b>	String	Identifica univocamente l'azienda all'interno dell'applicazione
<b>date</b>	String	Giorno a cui fanno riferimento gli altri attributi
<b>marketCap</b>	float	Capitalizzazione di mercato dell'azienda
<b>peRatio</b>	float	È il rapporto tra il prezzo corrente di un'azione e l'utile atteso per ogni azione
<b>EPS</b>	float	Gli utili per azione (earnings per share o EPS) sono gli utili che un'azienda ha generato parametrati al numero di azioni emesse dall'azienda stessa
<b>dividend</b>	String	È la parte di utile che viene consegnato dall'azienda ai suoi azionisti
<b>targetPrice</b>	float	È il livello di prezzo che gli analisti ritengono che il valore dell'azione raggiungerà entro il prossimo anno
<b>open</b>	float	È il valore di apertura in borsa dell'azienda, nel giorno 'date'
<b>close</b>	float	È il valore di chiusura in borsa dell'azienda, nel giorno 'date'
<b>avgVolume</b>	float	È il numero medio di azioni trattate negli ultimi 3 mesi
<b>volume</b>	float	È il numero di azioni scambiate nel giorno 'date'

### Classe: History

ATTRIBUTO	TIPO	DESCRIZIONE
<b>symbol</b>	String	Identifica univocamente l'azienda all'interno dell'applicazione
<b>dateHistory</b>	String	Giorno a cui fanno riferimento gli altri attributi

<b>open</b>	float	È il valore di apertura in borsa dell'azienda, nel giorno 'date'
<b>close</b>	float	È il valore di chiusura in borsa dell'azienda, nel giorno 'date'
<b>adjustedClose</b>	float	È il valore di chiusura aggiustata in borsa dell'azienda, nel giorno 'date'
<b>Volume</b>	int	È il numero di azioni scambiate nel giorno 'date'
<b>high</b>	float	È il valore massimo raggiunto dal titolo nel giorno 'date'
<b>low</b>	float	È il valore minimo raggiunto dal titolo nel giorno 'date'

### Classe: Report

<b>ATTRIBUTO</b>	<b>TIPO</b>	<b>DESCRIZIONE</b>
<b>dateReport</b>	String	È il giorno nel quale il report è stato scritto
<b>typeReport</b>	String	Indica la tipologia di report (descrittivo, analitico etc)
<b>analizedValues</b>	String	Indica una lista di valori (es. marketCap, peRatio, EPS etc) che sono stati analizzati e commentati dall'utente professionale che ha scritto il report
<b>details</b>	String	Altre informazioni e dettagli, scritti dall'utente professionale
<b>symbol</b>	String	Simbolo che identifica univocamente l'azienda oggetto del report
<b>username</b>	String	Nickname dell'utente professionale che ha scritto il report
<b>title</b>	String	Titolo del report

## 7 ARCHITETTURA DEL SOFTWARE

L'applicazione sviluppata è Client-Server.

L'Università di Pisa ci ha fornito tre macchine virtuali aventi i seguenti indirizzi IP:

1. 172.16.3.150
2. 172.16.3.121
3. 172.16.3.119

L'applicazione è stata sviluppata tramite l'utilizzo di un database a documento, MongoDB, e di un database a grafo, Neo4j.

MongoDB è stato implementato sulle tre macchine virtuali sopra riportate, creando un cluster nel quale la prima macchina è stata configurata come Primary e le altre due come Secondary.

Neo4j è invece stato implementato in locale, creando un Causal Cluster composto da 3 core servers e 3 replica servers.

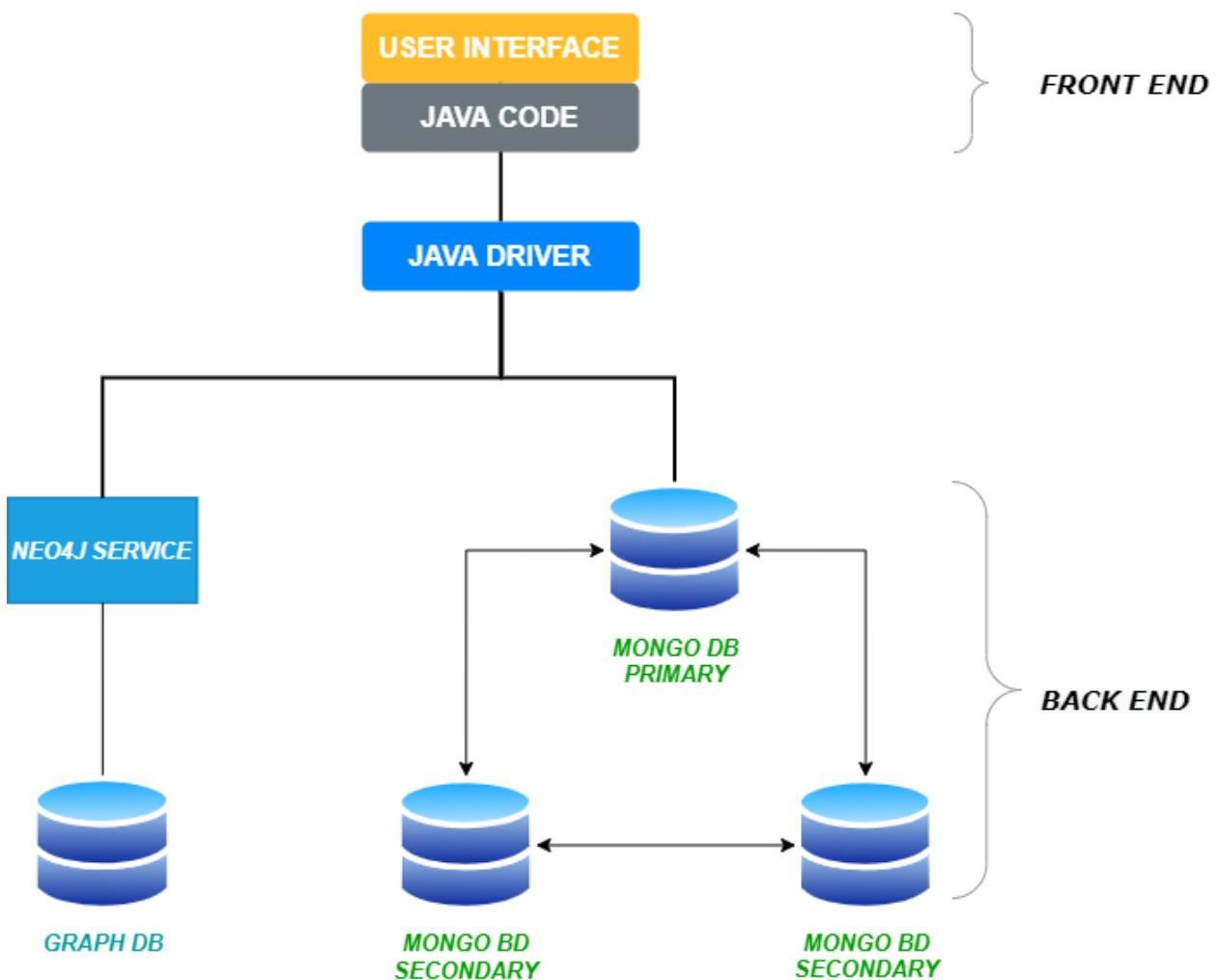


Figura 3 Diagramma architettura software

I due database, MongoDB e Neo4j, sono stati gestiti nel seguente modo:

- Su MongoDB sono stati memorizzati tutti i dati finanziari delle aziende al fine di averli sempre in uno stato consistente e disponibili anche in caso di network partitions.  
MongoDB si adatta bene perché ci permette di aver delle buone performance nonostante la notevole quantità di dati che deve essere gestita; per questo motivo è stato deciso di effettuare le operazioni di lettura con una politica che dà la precedenza ai replica servers, leggendo eventualmente sul primario qualora questi non fossero disponibili. Per le operazioni di scrittura viene invece adottata una politica di majority: le scritture vengono effettuate su un numero di server pari alla metà più uno, quindi due nel nostro caso. In questo modo riusciamo a garantire l'*eventual consistency*. Occorre sottolineare che, per quanto riguarda i dati giornalieri delle aziende, si è deciso di rendere tali dati sempre consistenti al momento dell'aggiornamento in modo tale che tutti gli utenti possano avere la stessa visione delle aziende alla chiusura della borsa, pertanto per tali dati è stata scelta una politica di scrittura su tutti i server.
- Su Neo4j sono stati memorizzati i dati riguardanti gli utenti, le informazioni generali delle aziende e le varie relazioni tra di essi, al fine di avere un servizio veloce e che possa gestire una quantità di dati e relazioni crescente nel tempo.  
Neo4j, tramite il suo meccanismo di *causal consistency*, è particolarmente adatto in quanto i dati che vogliamo memorizzare necessitano di esser sempre disponibili e consistenti. Tuttavia, per il CAP Theorem si va a sacrificare la tolleranza alle network partitions e, nel caso in cui si verifichi un partizione, presupponiamo che la nostra applicazione non sia momentaneamente fruibile dai vari utenti.

Infine, si vuole sottolineare che sono state inserite delle linee di codice in modo tale che entrambi i database mantengano uno stato consistente anche in caso di errori in fase di aggiornamento.

## 8 DESIGN DELL'APPLICAZIONE

---

### 8.1 MONGODB DESIGN

#### 8.1.1 Organizzazione dei documenti e delle collezioni

Le informazioni relative alle aziende sono state organizzate in documenti che si trovano in due collezioni, "companies" e "history"; inoltre, per memorizzare le informazioni relative ai report scritti dagli utenti professionali, è stata creata un'omonima collezione.

Osserviamo la struttura dei documenti che fanno parte di queste collezioni, partendo dai documenti che compongono la collezione "companies":

```
{
  "Exchange": "NMS",
  "Symbol": "AAPL",
  "Sector": "Technology",
  "Summary": [
    {
      "Market_cap": 2.40399994E12,
      "Volume": 1.57611712E8,
      "PE_Ratio": 43.57,
      "EPS": 3.28,
      "Close": 142.92,
      "Date": "2021-01-18",
      "Avg_Volume": 1.09086248E8,
      "Open": 143.07,
      "Dividend": "0.82 (0.59%)",
      "Target_price": 136.09
    },
    {
      "Market_cap": 2.408E12,
      "Volume": 9.8390456E7,
      "PE_Ratio": 43.65,
      "EPS": 3.28,
      "Close": 143.16,
      "Date": "2021-01-19",
      "Avg_Volume": 1.09868912E8,
      "Open": 143.6,
      "Dividend": "0.82 (0.57%)",
      "Target_price": 136.09
    },
    {
      "Market_cap": 2.2210001E12,
      "Volume": 9.876996E7,
      "PE_Ratio": 40.25,
```

```

      "EPS": 3.28,
      "Close": 132.03,
      "Date": "2021-01-20",
      "Avg_Volume": 1.07957416E8,
      "Open": 128.66,
      "Dividend": "0.82 (0.62%)",
      "Target_price": 134.51
    },
    {
      "Market_cap": 2.30300005E12,
      "Volume": 1.1969072E8,
      "PE_Ratio": 41.73,
      "EPS": 3.28,
      "Close": 136.87,
      "Date": "2021-01-21",
      "Avg_Volume": 1.08451344E8,
      "Open": 133.8,
      "Dividend": "0.82 (0.62%)",
      "Target_price": 134.51
    },
    {
      "Market_cap": 2.3400001E12,
      "Volume": 1.1445936E8,
      "PE_Ratio": 42.4,
      "EPS": 3.28,
      "Close": 139.07,
      "Date": "2021-01-22",
      "Avg_Volume": 1.08457456E8,
      "Open": 136.28,
      "Dividend": "0.82 (0.60%)",
      "Target_price": 134.51
    }
  ],
  "Name": "Apple Inc."
}

```

In questo documento, vengono memorizzati alcuni campi necessari per identificare un'azienda, ovvero **"name"**, **"symbol"**, **"sector"**, **"stockExchange"**; per inquadrare correttamente un'azienda nel suo complesso, questi quattro campi sono importanti.

Conoscendo questi quattro campi, è possibile sapere il nome dell'azienda, il simbolo che la identifica univocamente, il settore all'interno del quale l'azienda si colloca e la borsa nella quale il titolo è scambiato.

Vengono inoltre memorizzate le informazioni relative ai dati storici della settimana lavorativa, che va dal lunedì al venerdì, alla voce "summary". Le voci di summary sono quindi cinque, una per ogni giorno della settimana che va dal lunedì al venerdì.



I dati di ogni "summary" vengono memorizzati tramite documenti innestati, nell'entità 'company', in quanto ogni volta che è necessario recuperare i dati giornalieri della settimana in borsa di un'azienda, questo avviene effettuando una sola operazione di lettura al database.

Complessivamente, quindi, per ogni azienda vengono memorizzati i quattro campi che servono per identificarla e in più alcuni dati di interesse relativi all'ultima settimana in borsa dell'azienda. Questo sarà utile per dare l'opportunità ad utenti normali e utenti professionali di eseguire funzionalità avanzate e scoprire nuove e importanti caratteristiche riguardo l'azienda. Ad esempio, può essere molto interessante confrontare i dati dell'ultima settimana in borsa con i dati storici degli ultimi mesi, ed è per questo che, oltre al "summary", è presente la collezione che verrà ora analizzata: history.

Osserviamo la struttura dei documenti che formano la collezione "history":

```
{
  "Date": "2016-01-11",
  "Open": 24.7425,
  "Close": 24.282499,
  "High": 25.297501,
  "Low": 23.84,
  "AdjClose": 22.49704,
  "Volume": 1217348800,
  "Symbol": "AAPL"
}
```

Per quanto riguarda questa collezione, essa memorizza i dati storici dell'azienda identificata dal campo "symbol", con frequenza settimanale.

Questi dati storici sono stati scelti con frequenza settimanale dato che per poter studiare l'andamento di un titolo negli ultimi anni, infatti, è sufficiente avere dati storici con granularità settimanale e confrontarli con i dati della settimana corrente.

Osserviamo infine la struttura dei documenti che compongono la collezione "report":

```
{
  "Date": "2021-01-07",
  "Username": "fra.rossi",
  "Symbol": "AZN",
  "Title": "Is AstraZeneca Stock a Buy?",
  "Type": "Description",
  "AnalyzedValue": "PE_Ratio",
  "Text": "AstraZeneca (NASDAQ:AZN) has arguably been one of the more disappointing
    coronavirus vaccine stocks. Over the past year ..."
}
```

I report sono una delle parti più importanti dell'applicazione. Sono scritti dagli utenti professionali e possono essere letti da ogni utente.

L'obiettivo dei report è quello di tenere sempre aggiornata l'utenza riguardo le aziende, ovvero riguardo sia l'andamento del titolo sia quello del funzionamento dell'azienda stessa. Gli utenti professionali possono infatti sia scrivere report tecnici, ovvero riguardanti l'andamento del titolo, sia report più descrittivi che possono comprendere analisi sul business dell'azienda, sul settore al quale l'azienda appartiene o sulle trimestrali che le aziende pubblicano. Gli utenti che leggono questi report possono quindi rimanere sempre aggiornati, scoprire nuove aziende che prima non conoscevano e cogliere nuove opportunità di investimento.

In base alla qualità del report, all'accuratezza delle informazioni riportate dagli utenti professionali, e in base al gusto personale, ogni utente può dare una propria valutazione ad ogni utente professionale. La valutazione che un utente professionale si vede dare può ad esempio spingerlo a scrivere report sempre migliori, nel caso in cui la valutazione non lo soddisfi.

### 8.1.2 Operazioni CRUD – MongoDB

Sono ora mostrate le operazioni CRUD che possono essere eseguite sul database MongoDB. Di seguito, verrà riportata anche una stima della frequenza di utilizzo delle operazioni offerte agli utenti:

- *low*: meno 50 volte al giorno
- *medium*: meno di 150 volte e più di 50 volte al giorno
- *high*: più di 150 volte al giorno

#### Collezione: Companies

OPERAZIONE	FREQUENZA ATTESA	DESCRIZIONE
createCompany	low	Un'azienda viene aggiunta al database
readCompany_bySymbol	high	Un utente (o un utente professionale) legge le informazioni relative a un'azienda
updateCompany_Summary	low	Le informazioni relative alle 5 voci "summary" dell'azienda vengono aggiornate
deleteCompany_bySymbol	low	Un'azienda viene tolta dal database

## Collezione: History

OPERAZIONE	FREQUENZA ATTESA	DESCRIZIONE
createHistory	low	I dati storici relativi a una settimana (campo date = lunedì della settimana) vengono aggiunti al database
readHistory_bySymbol	high	Un utente (o un utente professionale) legge le informazioni relative ai dati storici di un'azienda
readHistory_byPeriod	high	Un utente (o un utente professionale) legge i dati storici di un'azienda relativi a un particolare periodo di tempo
deleteHistory_bySymbol	low	I dati storici di un'azienda vengono rimossi dal database
deleteHistory_byPeriod	low	I dati storici di un'azienda, relativi a un dato periodo di tempo, sono rimossi dal database

## Collezione: Report

OPERAZIONE	FREQUENZA ATTESA	DESCRIZIONE
createReport	medium	Un report viene aggiunto al database
readReports_BySymbol	high	Un utente o un utente professionale prendono visione delle informazioni relative ad alcuni report. I report sono relativi a una particolare azienda
readReports_ByProfessional User_Username	high	Un utente o un utente professionale prendono visione delle informazioni relative ad alcuni report. I report sono relativi a un utente professionale
updateReport_Text_byTitle	low	Il testo di un report, trovato tramite il suo titolo e l'username dell'utente professionale che l'ha scritto, viene aggiornato
deleteReport_bySymbol	low	Un report, trovato tramite il simbolo dell'azienda a cui il report fa riferimento, viene rimosso dal database
deleteReport_byUsername	low	Un report, trovato tramite l'username dell'utente professionale che l'ha scritto, viene rimosso dal database

## Routine di aggiornamento

OPERAZIONE	FREQUENZA ATTESA	DESCRIZIONE
updateSummary	daily	Aggiorna i dati giornalieri delle summary di ogni azienda
updateHistory	weekly	Settimanalmente svuota le summary e aggiorna i dati delle history

## 8.2 NEO4J DESIGN

Verrà ora presentata la struttura del Graph Database, partendo dai nodi e dalle relazioni che la compongono.

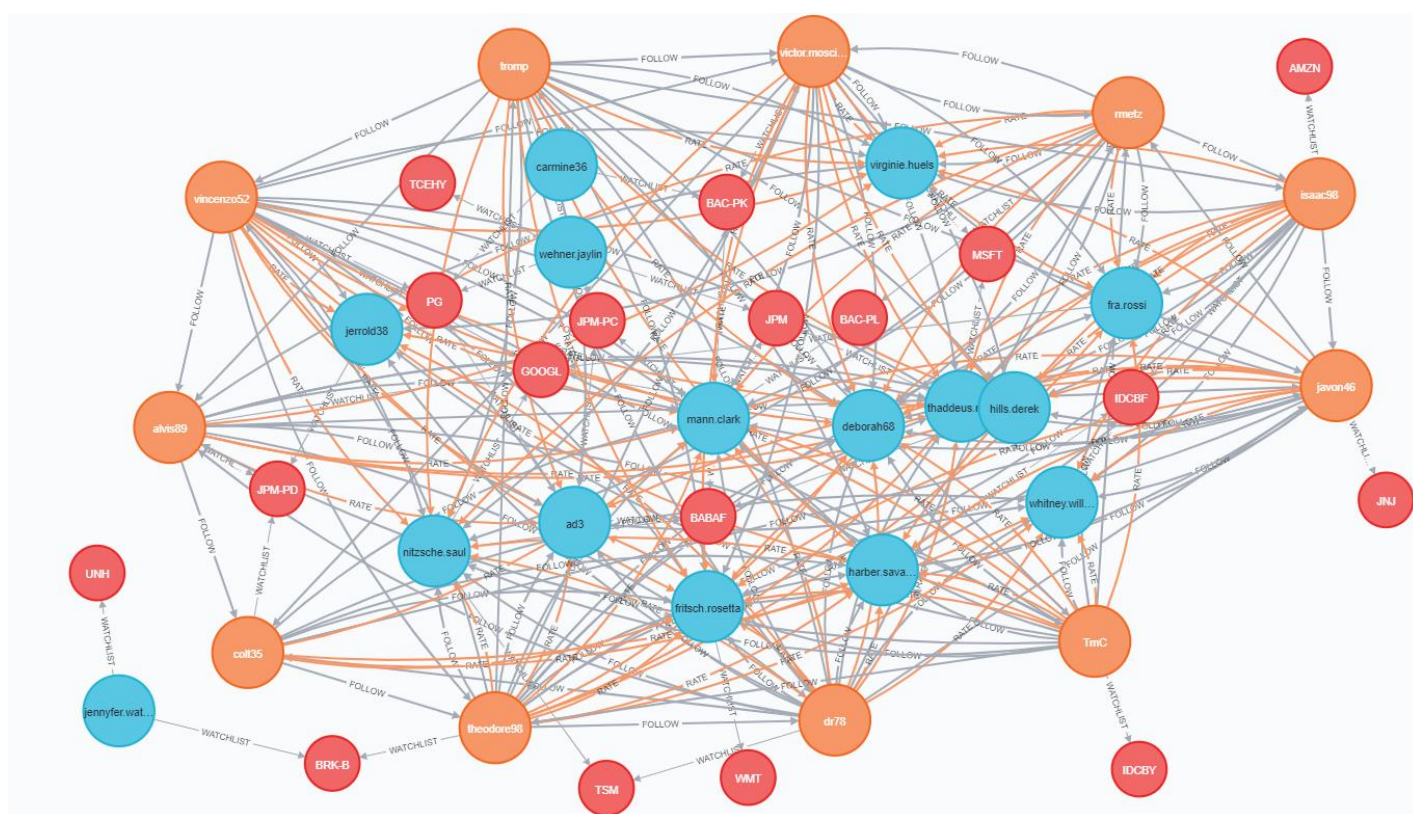


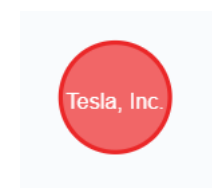
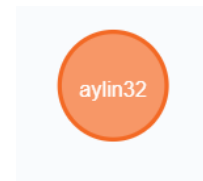
Figura 4  
Grafo completo

### 8.1.3 Nodi e Relazioni

#### NODI

I nodi che compongono il grafo sono tre:

- Il nodo "**User**", che memorizza le informazioni degli utenti che si iscrivono all'applicazione; è in questa categoria che rientra l'admin.
- Il nodo "**Professional\_User**", che memorizza le informazioni degli utenti professionali che si iscrivono all'applicazione per scrivere i report.
- Il nodo "**Company**", che memorizza le informazioni generali delle aziende (telefono, indirizzo, ...) che sono presenti nell'applicazione.



#### RELAZIONI

Le relazioni che legano tra loro i nodi sono anch'esse tre:

- La relazione "**FOLLOW**", che rappresenta il legame tra due utenti dei quali almeno uno segue l'altro. Le due possibilità sono quindi:
  - **User -> FOLLOW -> User**: in questo caso, un utente segue un altro utente.
  - **User -> FOLLOW -> Professional\_User**: in questo caso, un utente segue un utente professionale.

Questa relazione non ha attributi.

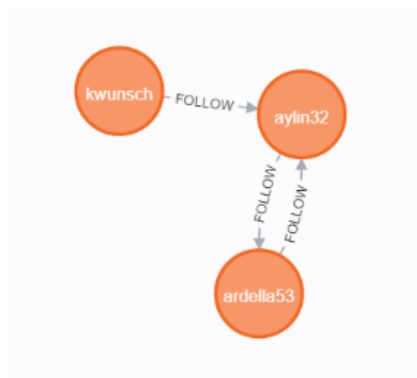


Figura 5  
Tre utenti che si seguono

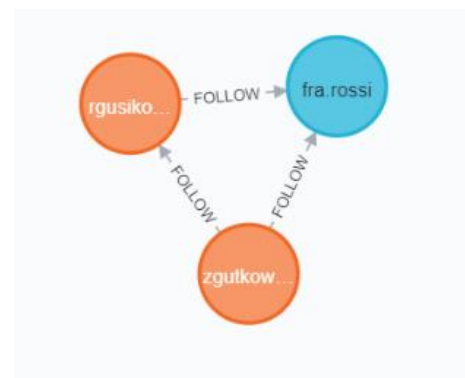


Figura 6  
Due utenti che seguono un utente professionale

- La relazione "**WATCHLIST**", che rappresenta il collegamento tra un utente (o un utente professionale) e un'azienda da lui seguita. Anche in questo caso, come per la relazione 'FOLLOW' ci sono due possibilità:
  - User -> WATCHLIST -> Company**: in questo caso, un utente aggiunge un'azienda alla sua watchlist.
  - Professional\_User -> WATCHLIST -> Company**: in questo secondo caso, invece, è un utente professionale ad aggiungere un'azienda alla sua watchlist.

Questa relazione non ha attributi.

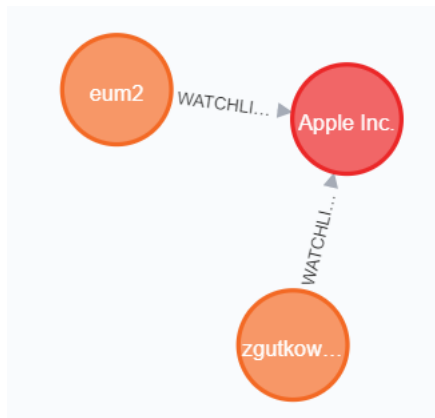


Figura 8  
Due utenti che seguono un'azienda

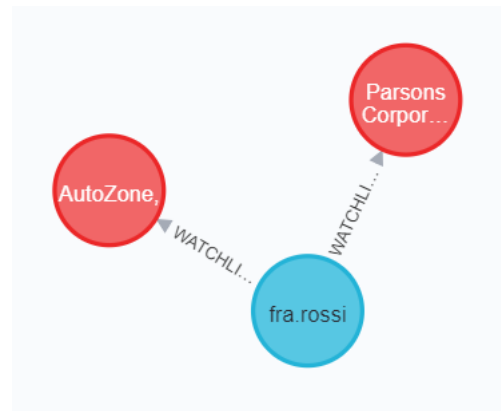


Figura 7  
Un utente professionale che segue due aziende

- La relazione "**RATE**", che rappresenta la valutazione di un utente ad un utente professionale. Un utente può infatti dare un voto ad un utente professionale, per esempio in base alla qualità e all'accuratezza dei report scritti da quest'ultimo. La valutazione espressa dall'utente viene memorizzata nell'attributo "vote" della relazione. Questo attributo è composto da un intero compreso tra 1 e 5, estremi inclusi.

Questa relazione ha la seguente struttura:

- User -> RATE -> Professional\_User**

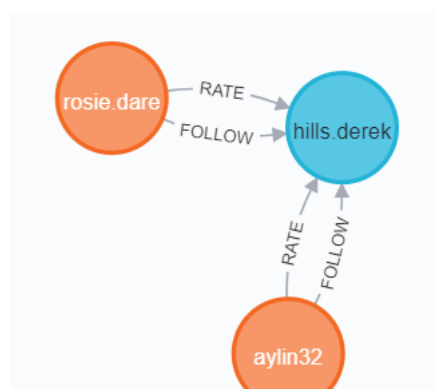


Figura 9  
Due utenti che danno un voto a un utente professionale

### 8.1.4 Operazioni CRUD – Neo4j

Sono ora mostrate le operazioni CRUD che possono essere eseguite sul database Neo4j, per ogni nodo. Per le frequenze valgono le stesse considerazioni riportate nel paragrafo di MongoDB.

#### Nodo: 'User'

OPERAZIONE	FREQUENZA ATTESA	DESCRIZIONE
addUser	low	Un utente viene aggiunto al database
readUser_ByUsername	high	Un utente legge le informazioni relative a un altro utente
addUser_toFollow	medium	Un utente inizia a seguire un altro utente
unfollowUser	low	Un utente smette di seguire un altro utente
followCompany_ByUser	high	Un utente inizia a seguire un'azienda
unfollowCompany_ByUser	low	Un utente smette di seguire un'azienda
followProfessionalUser_ByUser	medium	Un utente inizia a seguire un utente professionale
unfollowProfessionalUser_ByUser	low	Un utente smette di seguire un utente professionale
rateProfessionalUser	high	Un utente valuta un utente professionale
deleteUser_ByUsername	low	un utente viene rimosso dal database

#### Nodo: 'Professional\_User'

OPERAZIONE	FREQUENZA ATTESA	DESCRIZIONE
addProfessionalUser	low	Un utente professionale viene aggiunto al database
readProfessionalUser_ByUsername	high	Un utente (o un utente professionale) legge le informazioni relative a un altro utente professionale
followCompany_ByProfessionalUser	high	Un utente professionale inizia a seguire un'azienda
unfollowCompany_ByProfessionalUser	medium	Un utente professionale inizia a seguire un'azienda
deleteProfessionalUser	low	Un utente professionale viene rimosso dal database

**Nodo: 'Company'**

OPERAZIONE	FREQUENZA ATTESA	DESCRIZIONE
addCompany	low	Un'azienda viene aggiunta al database
readCompany_BySymbol	high	Un utente o un utente professionale prendono visione delle informazioni riguardo un'azienda
deleteCompany_BySymbol	low	Un'azienda viene rimossa dal database

**Nodo: 'Admin'**

OPERAZIONE	FREQUENZA ATTESA	DESCRIZIONE
addAdmin	low	Un admin promuove un utente normale a nuovo amministratore



## 9 IMPLEMENTAZIONE

---

### 9.1 IMPLEMENTAZIONE DELLE OPERAZIONI CRUD - MONGODB

Sono qui riportate le implementazioni relative a quattro tra le più rilevanti operazioni CRUD mostrate appena sopra.

#### Operazione "readCompany\_bySymbol":

```
public static Companies readCompany_bySymbol(String symbol)
{
    FindIterable<Document> iterable = collectionCompanies.find(eq("Symbol", symbol));
    MongoCursor<Document> cursor = iterable.iterator();

    String profit;
    Companies c = new Companies();

    try {
        while (cursor.hasNext()) {
            JSONObject json = new JSONObject(cursor.next());

            List<Summary> summaries = new ArrayList<>();

            if (!json.getJSONArray("Summary").isEmpty())
            {
                JSONArray a = new JSONArray(json.getJSONArray("Summary"));

                Summary s = new Summary();

                for (int j = 0; j < a.length(); j++)
                {
                    s.setDate(a.getJSONObject(j).getString("Date"));
                    s.setClose(a.getJSONObject(j).getFloat("Close"));
                    s.setOpen(a.getJSONObject(j).getFloat("Open"));
                    s.setVolume(a.getJSONObject(j).getFloat("Volume"));
                    s.setAvgVolume(a.getJSONObject(j).getFloat("Avg_Volume"));
                    s.setMarketCap(a.getJSONObject(j).getFloat("Market_cap"));
                    s.setPeRatio(a.getJSONObject(j).getFloat("PE_Ratio"));
                    s.setEPS(a.getJSONObject(j).getFloat("EPS"));
                    s.setDividend(a.getJSONObject(j).getString("Dividend"));
                    s.setTargetPrice(a.getJSONObject(j).getFloat("Target_prize"));

                    summaries.add(s);
                }
            }
            if(json.isNull("mostProfitablePeriod"))
                profit = null;
            else
                profit = json.getString("mostProfitablePeriod");

            c = new Companies(json.getString("Symbol"), json.getString("Name"),
                json.getString("Exchange"),
                json.getString("Sector"), profit, summaries );
        }

    } finally {
        cursor.close();
    }
    return c;
}
```

## Operazione "readHistory\_byPeriod"

```
public static List<History> readHistory_byPeriod(String start_date, String end_date, String symbol) //OK
{
    List<History> historyList = new ArrayList<>();
    History h;
    FindIterable<Document> iterable = collectionHistory.find(and(eq("Symbol", symbol),gt("Date",
start_date), lt("Date", end_date)));
    MongoCursor<Document> cursor = iterable.iterator();

    try {
        while (cursor.hasNext()) {
            JSONObject json = new JSONObject(cursor.next());
            //System.out.println(json.toString());
            h = new History(json.getString("Symbol"), json.getString("Date"), json.getFloat("Open"),
json.getFloat("Close"),
                json.getFloat("AdjClose"), json.getInt("Volume"), json.getFloat("High"),
json.getFloat("Low"));
            historyList.add(h);
        }

    } finally {
        cursor.close();
    }

    return historyList;
}
```

## Operazione "readReports\_byUsername"

```
public static List<Report> readReports_byUsername(String username)//OK
{
    FindIterable<Document> iterable = collectionHistory.find(eq("Username", username));
    MongoCursor<Document> cursor = iterable.iterator();
    List<Report> reportsList = new ArrayList<>();
    Report r;

    try {
        while (cursor.hasNext()) {
            JSONObject json = new JSONObject(cursor.next());
            r = new
Report(json.getString("Title"),json.getString("Date"),json.getString("Type"),json.getString("AnalizedValu
e"),
                json.getString("Text"),json.getString("Symbol"),json.getString("Username"));
            reportsList.add(r);
        }

    } finally {
        cursor.close();
    }

    return reportsList;
}
```

## 9.2 ANALYTICS FUNCTIONS - MONGODB

Sono ora mostrate le funzionalità analytics che possono essere eseguite sul database MongoDB.

### Analytic 1

Tramite questa analytic, che può esser eseguita solo da un admin, quest'ultimo scopre qual è stato il mese e l'anno più redditizio di ogni azienda presente nell'applicazione. Data la complessità dell'operazione una volta eseguita, il dato ottenuto viene memorizzato all'interno del documento di ogni compagnia in modo tale che poi possa esser fruibile da tutti gli utenti dell'applicazione per poter cogliere un'opportunità di investimento.

```
db.history.aggregate([
  {
    $group:
    {
      _id:
      {
        symbol: '$Symbol',
        year: { $year: { $dateFromString: { dateString: '$Date', format: "%Y-%m-%d" } } },
        month: { $month: { $dateFromString: { dateString: '$Date', format: "%Y-%m-%d" } } },
        day: { $dayOfMonth: { $dateFromString: { dateString: '$Date', format: "%Y-%m-%d" } } },
        close: '$Close'
      }
    }
  },
  { $sort: { _id: 1 } },
  {
    $group:
    {
      _id: { symbol: "$_id.symbol", year: "$_id.year", month: "$_id.month"},
      lastCloseMonth : { $last: "$_id.close"},
      firstCloseMonth : { $first: "$_id.close"}
    }
  },
  { $sort: { _id: 1 } },
  {
    $addFields: { differenceMonth : { $subtract: ["$lastCloseMonth", "$firstCloseMonth"] } }
  },
  { $sort: { differenceMonth: -1 } },
  {
    $group:
    {
      _id: { symbol: "$_id.symbol"},
      maxMonth : { $max: "$differenceMonth"},
      Year: { $first: "$_id.year"},
      Month: { $first: "$_id.month"}
    }
  },
],
```

```

{
    $project:{
        _id:0,
        Symbol:"$_id.symbol",
        mostProfitablePeriod: { $concat: [ { $toString: "$Year" }, "-", { $toString: "$Month" },
        "-", { $toString: "$maxValue"} ] }
    },
    { $merge : { into: "companies", on: "Symbol", whenMatched: "merge", whenNotMatched: "insert" } }
], {allowDiskUse : true}
)

```

## Analytic 2

Mediante questa seconda analytic, un utente professionale o un admin può verificare se in un azienda il volume di affari scambiati durante la settimana corrente è, per ogni giorno della settimana, superiore al volume medio di affari scambiati negli ultimi 3 mesi. Se questa condizione è verificata, allora si può stabilire che l'interesse sul titolo stia crescendo. Maggiore è infatti l'interesse su un titolo e più è probabile che il titolo salga.

```

db.companies. aggregate(
{
    $match:{ "Summary.Avg_Volume": { $lt: 100 } }
},
{
    $unwind : "$Summary"
},
{
    $project:{
        _id: {symbol: '$Symbol' },
        count: { $cond:[{ $gt: [ "$Summary.Volume" , "$Summary.Avg_Volume" ] }, 1, 0] }
    }
},
{
    $group:{
        _id:{ symbol:"$_id.symbol" },
        tot: { $sum: "$count" }
    }
},

```

```
{
  $project: {
    _id:{ symbol:"$_id.symbol" },
    interesse: { $cond:[ { $gt: [ "$tot" , 2 ] }, 'yes' , 'no' ] } //INSERIRE 5
  }
},
{ $sort: { interesse:-1 } }
)
```

### Analytic 3

Questa analytic è utilizzata per trovare, per ogni settore, l'azienda più sottovalutata. Per quanto riguarda aziende appartenenti a uno stesso settore, infatti, il valore del PE Ratio è un indicatore di quanto un'azienda sia sovrastimata o sottostimata dai mercati. Minore è il PE Ratio, maggiore è la sottovalutazione del titolo e, quindi, maggiore potrebbe essere il rendimento futuro del titolo stesso.

Lo scopo è quello di offrire all'utenza la possibilità di scovare, tra tutte le aziende appartenenti a uno specifico settore, quella più sottovalutata e quindi una di quelle più interessanti.

```
db.companies.aggregate(
{
  $match:{ "Summary.Market_cap":{ $lt: 100000000000 } }
},
{
  $unwind: { path: "$Summary" } },
{ $sort: { Date:-1 } },
{
  $group:{
    _id: {sector: "$Sector", symbol:"$Symbol"},
    data: { $last: "$Summary.Date"},
    PE: { $last: "$Summary.PE_Ratio"}
  }
},
{
  $match:{ PE: { $ne:0 } } },
{
  $match:{ "_id.sector": { $ne:"" } } },
{ $sort: { "_id.sector":-1, PE:-1 } },
{
  $group:{
    _id: {sector:"$_id.sector"},
    symbol: { $last: "$_id.symbol"},
    PE: { $last:"$PE"}
  }
}
)
```

### 9.3 IMPLEMENTAZIONE DELLE OPERAZIONI CRUD – Neo4j

OPERAZIONE	IMPLEMENTAZIONE CYPHER
Un utente inizia a seguire un utente professionale (USER)	<pre> MATCH (u1:User) WHERE u1.username = \$username MATCH (p:Professional_User) WHERE p.username = \$username_pf CREATE (u1)-[FOLLOW]-&gt;(p) </pre>
Un utente esprime una valutazione riguardo un utente professionale (USER)	<pre> MATCH (u:User {username:\$username}) MATCH (p:Professional_User{username:\$username_pf}) MERGE (u)-[:FOLLOW]-&gt;(p) MERGE (u)-[r:RATE]-&gt;(p) ON CREATE SET r.vote = \$vote ON MATCH SET r.vote = \$vote </pre>
Un utente o un utente professionale prende visione delle informazioni riguardo un altro utente professionale (PROFESSIONAL_USER)	<pre> MATCH (p:Professional_User) WHERE p.username = \$username RETURN p </pre>
Un'azienda viene aggiunta al database (ADMIN)	<pre> CREATE (c:Company{ symbol: \$symbol, name: \$name, exchange: \$exchange, fullTimeEmployees: \$ftse, description: \$description, city: \$city, phone: \$phone, state: \$state, country: \$country, address: \$address, website: \$website, sector: \$sector }) </pre>

### 9.4 ANALYTICS FUNCTIONS – Neo4j

Sono ora mostrate le funzionalità analytics che possono essere eseguite sul database Neo4j, osservandone l'implementazione mediante l'uso del linguaggio Cypher.

#### Analytic 1: 'User Following'

domain-specific	graph-centric
Dato l'username di un utente, si vuole sapere quanti utenti egli segue e quante aziende fanno parte della sua watchlist	Dato un nodo <b>User</b> , si vuole sapere quanti sono gli archi in uscita verso altri nodi <b>User</b> e verso nodi <b>Company</b>

## IMPLEMENTAZIONE CYPHER

```
MATCH (u1:User)-[:FOLLOW]->(u2:User),(u1:User)[:WATCHLIST]-(c1:Company)
WHERE u1.username = $username
RETURN count(DISTINCT u2) as NumberFollowerUser, count(DISTINCT c1) as
NumberFollowerCompany
```

### Analytic 2: 'ProfessionalUser Follow'

domain-specific	graph-centric
Dato l'username di un utente professionale, si vuole sapere il numero di aziende da lui seguite e il numero di utenti da cui è seguito	Dato un nodo <b>ProfessionalUser</b> , si vuole sapere quanti sono gli archi in uscita verso nodi <b>Company</b> e il numero di archi entranti da nodi <b>User</b>

## IMPLEMENTAZIONE CYPHER

```
MATCH (u1:User)-[:FOLLOW]->(u2:Professional_User),(u2:Professional_User)-[:WATCHLIST]-(c1:Company)
WHERE u2.username = $username
RETURN count(DISTINCT u2) as NumberFollower, count(DISTINCT c1) as
NumberFollowedCompany
```

### Analytic 3: 'Suggested Company'

domain-specific	graph-centric
Dato l'username di un utente, si vuole sapere quali sono le aziende che non sono presenti nella sua watchlist e che sono seguite da altri utenti	Dato un nodo <b>User</b> , si vuole trovare quali sono i nodi <b>Company</b> che distano 1 o 2 hop dal nodo stesso

## IMPLEMENTAZIONE CYPHER

```
MATCH (u1:User)-[*1..2]-(u2:User)-[w:WATCHLIST]-(c1:Company)
WHERE u1.username = $username
AND NOT (u1)-[WATCHLIST]-(c1)
AND NOT u1.username = u2.username
RETURN DISTINCT c1.symbol as Company LIMIT 100
```

#### Analytic 4: 'Followed Company List (By User)'

domain-specific	graph-centric
Dato l'username di un utente, si vuole sapere la lista delle aziende da lui seguite	Dato un nodo <b>User</b> , si vuole trovare il numero di archi in uscita verso nodi <b>Company</b>

#### IMPLEMENTAZIONE CYPHER

**MATCH** (u1:User)-[:WATCHLIST]-(c1:Company)

**WHERE** u1.username = \$username

**RETURN** c1

#### Analytic 5: 'Followed Company List (By ProfessionalUser)'

domain-specific	graph-centric
Dato l'username di un utente professionale, si vuole sapere la lista delle aziende da lui seguite	Dato un nodo <b>ProfessionalUser</b> , si vuole trovare il numero di archi in uscita verso nodi <b>Company</b>

#### IMPLEMENTAZIONE CYPHER

**MATCH** (u1:Professional\_User)-[:WATCHLIST]-(c1:Company)

**WHERE** u1.username = \$username

**RETURN** c1

#### Analytic 6: 'Followed User List'

domain-specific	graph-centric
Dato l'username di un utente, restituisce la lista degli utenti da lui seguiti	Dato un nodo <b>User</b> , si vuole trovare i nodi <b>User</b> raggiungibili attraverso un arco in uscita

#### IMPLEMENTAZIONE CYPHER

**MATCH** (u1:User)-[:FOLLOW]->(u2:User)

**WHERE** u1.username = \$username

**RETURN** u2.username **AS** user



### Analytic 7: 'Followed Professional User List'

domain-specific	graph-centric
Dato l'username di un utente professionale, restituisce la lista degli utenti da lui seguiti	Dato un nodo <b>ProfessionalUser</b> , si vuole trovare i nodi <b>User</b> che hanno un arco in uscita verso il nodo stesso

#### IMPLEMENTAZIONE CYPHER

```
MATCH (u1:User)-[:FOLLOW]->(u2:Professional_User)
```

```
WHERE u1.username = $username
```

```
RETURN u2.username AS user
```

### Analytic 8: 'All Users List'

domain-specific	graph-centric
Restituisce la lista di tutti gli utenti iscritti all'applicazione	Restituisce tutti i nodi <b>User</b> che fanno parte del grafo

#### IMPLEMENTAZIONE CYPHER

```
MATCH (n:User)
```

```
RETURN n ORDER BY n.username
```

### Analytic 9: 'All Professional Users List'

domain-specific	graph-centric
Restituisce la lista di tutti gli utenti professionali iscritti all'applicazione	Restituisce tutti i nodi <b>ProfessionalUser</b> che fanno parte del grafo

#### IMPLEMENTAZIONE CYPHER

```
MATCH (n:Professional_User)
```

```
RETURN n ORDER BY n.username
```

## Analytic 10: 'All Companies List'

domain-specific	graph-centric
Data una lettera dell'alfabeto, restituisce una lista di tutte le aziende che iniziano con quella lettera	Restituisce tutti i nodi <b>Company</b> aventi l'attributo "Symbol" che inizia con un particolare carattere

## IMPLEMENTAZIONE CYPHER

**MATCH** (n:Company)

**WHERE** n.symbol STARTS WITH \$car

**RETURN** n **ORDER BY** n.symbol

## 9.5 STRUTTURA DEL REPOSITORY

La struttura del repository del progetto CompaniesDiscoveryApplication è riportata in questa sezione:

**./:** contiene il file POM utilizzato per generare le dipendenze del maven e per costruire il progetto.

**Scraper** : questa cartella contiene tutti gli scraper che sono stati realizzati per poter reperire correttamente i dati

**src/main/java:** contiene:

- **API:** contiene 3 classi:
  - **NestedDocuments:** per creare i documenti innestati
- **Entities:** contiene le classi dell'applicazione, ovvero:
  - **Admin:** admin dell'applicazione
  - **User:** utente che si iscrive all'applicazione
  - **ProfessionalUser:** utente professionale
  - **Companies:** informazioni relative all'azienda
  - **Summary:** documento innestato in "companies"
  - **History:** dati storici dell'azienda
  - **Report:** report scritti dagli utenti professionali
- **MongoDB:** contiene 3 classi:
  - **Analytics:** contiene le funzionalità analytics che possono essere eseguite su MongoDB
  - **CrudOperations:** contiene le operazioni CRUD che possono essere eseguite su MongoDB
  - **MongoDatabaseAccess:** contiene le informazioni riguardo alla connessione a MongoDB
  - **UpdateEntities:** contiene le routine per l'aggiornamento del database
- **Neo4j:** contiene 3 classi:
  - **Analytics:** contiene le funzionalità analytics che possono essere eseguite su Neo4j
  - **CrudOperations:** contiene le operazioni CRUD che possono essere eseguite su Neo4j
  - **Neo4jDatabaseAccess:** contiene le informazioni riguardo alla connessione a Neo4j

La repository è disponibile al seguente link:

<https://github.com/Matteo-Castrignano/CompaniesDiscoveryApplications>

## 10 GENERAZIONE DEL DATABASE

---

I dati utilizzati in questa applicazione sono stati reperiti dai seguenti link:

- Link 1: <https://it.finance.yahoo.com>
- Link 2: <https://financialmodelingprep.com/developer/docs>

Una volta scaricata la lista delle aziende dal link 2, si è proceduti a reperire gli altri dati nel seguente modo:

- **History:** sono i dati storici delle aziende, reperiti dal link 2 tramite uno scraper.
- **Companies:** sono i dati con le principali informazioni delle aziende, reperiti anch'essi dal link 2 tramite uno scraper dedicato. Sono dati che non cambiano frequentemente nel tempo; fanno parte di questi dati, ad esempio, la sede e il numero di telefono dell'azienda.
- **Summary:** sono i dati giornalieri delle aziende, reperiti dal link 2 tramite uno scraper che viene eseguito ogni giorno alla chiusura della borsa americana (ore 16:00 UTC-5). Questi dati cambiano giornalmente.
- **Report:** sono inseriti nel database ogni volta che sono scritti da un utente professionale. Il loro numero cresce nel tempo.
- **User e ProfessionalUser:** sono inseriti nel database ogni volta che un utente normale o un utente professionale si iscrive all'applicazione. Il loro numero può crescere o diminuire nel tempo.

## 11 TEST ESEGUITI E ANALISI STATISTICHE

Al fine di poter eseguire dei test sulle performance del database nell'eseguire le query, sono stati inseriti degli indici.

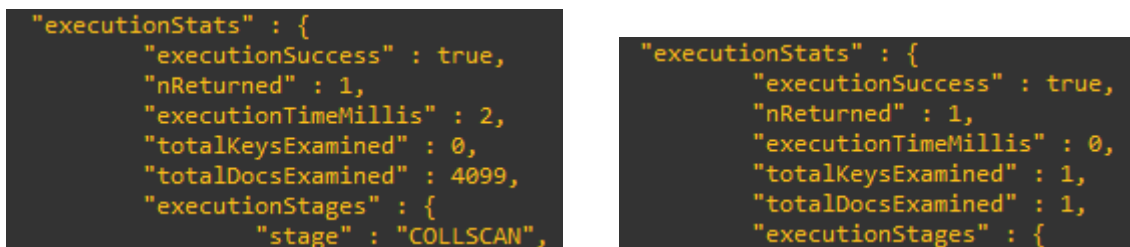
Relativamente alla parte di Neo4j, l'inserimento di indici non ha portato a miglioramenti riguardo il tempo di esecuzione delle query.

Lo stesso non si può dire per MongoDB; l'inserimento di indici su tale database a documento ha portato a una sensibile diminuzione del tempo di esecuzione delle query che hanno una frequenza di esecuzione maggiore, ottenendo così dei miglioramenti .

### 11.1 INDICI SU MONGODB

#### Indice 1: velocizzare le letture delle informazioni di un'azienda

Il primo indice è stato inserito, sul simbolo di una azienda (campo "Symbol"), per velocizzare la lettura delle informazioni delle aziende, dato che questa è un'operazione molto frequente nell'applicazione. Osserviamo per prima cosa le prestazioni di tale lettura prima dell'inserimento dell'indice:



```
"executionStats" : {
  "executionSuccess" : true,
  "nReturned" : 1,
  "executionTimeMillis" : 2,
  "totalKeysExamined" : 0,
  "totalDocsExamined" : 4099,
  "executionStages" : {
    "stage" : "COLLSCAN",
```

```
"executionStats" : {
  "executionSuccess" : true,
  "nReturned" : 1,
  "executionTimeMillis" : 0,
  "totalKeysExamined" : 1,
  "totalDocsExamined" : 1,
  "executionStages" : {
```

Figura 10 Performance prima (a sinistra) e dopo (a destra) l'inserimento dell'indice

Possiamo notare che il tempo di esecuzione della query passa da 2 millisecondi a circa zero, e che il numero di documenti esaminati si riduce drasticamente, passando da 4099 a 1.

#### Indice 2: velocizzare l'analytic che mostra l'interesse crescente o decrescente verso un'azienda (analytic numero 2)

Il secondo indice è stato inserito, sul volume medio di un'azienda (campo "avgVolume"), al fine di migliorare le prestazioni della analytic numero 2.

Osserviamo le prestazioni dell'esecuzione di questa analytic prima e dopo l'inserimento dell'indice:

```

"executionStats" : {
  "executionSuccess" : true,
  "nReturned" : 174,
  "executionTimeMillis" : 11,
  "totalKeysExamined" : 0,
  "totalDocsExamined" : 4099,
  "executionStages" : {
    "stage" : "PROJECTION_SIMPLE",

```

```

"executionStats" : {
  "executionSuccess" : true,
  "nReturned" : 174,
  "executionTimeMillis" : 5,
  "totalKeysExamined" : 298,
  "totalDocsExamined" : 174,
  "executionStages" : {

```

Figura 11 Performance della analytic prima (a sinistra) e dopo (a destra) l'inserimento dell'indice

In questo caso, notiamo che il tempo di esecuzione della analytic risulta più che dimezzato, passando da 11 millisecondi a 5 millisecondi; inoltre, il numero di documenti esaminati cala vistosamente, passando da 4099 prima dell'inserimento dell'indice a 174 dopo aver inserito l'indice.

### Indice 3: velocizzare le letture dei dati storici di un'azienda

Il terzo indice è stato inserito sull'operazione di lettura delle informazioni della "history" di un'azienda, operazione molto frequente nell'applicazione. È risultato conveniente inserire un indice, sul simbolo di un'azienda (campo "Symbol"), al fine di ottenere migliori prestazioni nell'esecuzione di questa query. Come per i casi precedenti, andiamo ad osservare le prestazioni del database prima e dopo l'inserimento dell'indice:

```

"executionStats" : {
  "executionSuccess" : true,
  "nReturned" : 110,
  "executionTimeMillis" : 586,
  "totalKeysExamined" : 0,
  "totalDocsExamined" : 927389,
  "executionStages" : {
    "stage" : "COLLSCAN",

```

```

"executionStats" : {
  "executionSuccess" : true,
  "nReturned" : 110,
  "executionTimeMillis" : 1,
  "totalKeysExamined" : 122,
  "totalDocsExamined" : 122,
  "executionStages" : {

```

Figura 12 Performance della query prima (a sinistra) e dopo (a destra) l'inserimento dell'indice

Eseguendo un confronto tra le due prestazioni, notiamo che il tempo di esecuzione della query passa da 586 millisecondi a 1 millisecondo, e che il totale dei documenti esaminati passa da 927389 documenti a 122 documenti. Anche in questo caso l'inserimento dell'indice risulta particolarmente rilevante.

## 12 MANUALE D'USO

---

All'avvio dell'applicazione l'utente si trova davanti una schermata per scegliere se loggarsi oppure registrarsi all'applicazione qualora vi fosse ancora iscritto.

```
Choose operation:
1. Login
2. Sing In
3. Shutdown
1
Insert Username: ugo.falleni
Insert Password: root

Choose operation:
-1. Show Comands
0. Logout
-1
```

Figura 13

Una volta che l'utente si è loggato può visionare la lista dei suoi comandi a disposizione in base al suo ruolo ed effettuare le varie operazioni disponibili.

```
Choose operation:
----- OPERATION ON USER -----
1. User list
2. Find a user
3. Follow a new user
4. Unfollow a user
5. Delete user
----- OPERATION ON PROFESSIONAL USER -----
6. Professional user list
7. Find a professional user
8. Delete professional user
9. Follow a new professional user
10. Rate a professional user
11. Unfollow a professional user
----- OPERATION ON COMPANY -----
12. Company list
13. Read a company data
14. Add a new company
15. Delete a company
16. Read all company history
17. Read a company history by period
18. Follow a new company
19. Unfollow a company
----- OPERATION ON REPORT -----
20. Read all report of a company
21. Read all report of a professional user
----- ANALYTICS & UPDATE DATA -----
22. Most profitable period
23. Verify the interest of companies
24. Most underrated companies
25. Update history
26. Update summary
27. Personal information
28. Get suggests companies
29. Add a new admin
-----
-1. Show Comands
0. Logout
```

Figura 14