Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη διαδικτυακής εφαρμογής για την αναζήτηση κειμενικών δεδομένων με τη συνεργασία NoSQL βάσης δεδομένων και του ElasticSearch**
Ονοματεπώνυμο Φοιτητή	**Δημήτρης Τσαρούχας**
Πατρώνυμο	**Νικόλαος**
Αριθμός Μητρώου	**ΜΠΠΛ/ 10030**
Επιβλέποντες	**Χρήστος Δουληγέρης, Καθηγητής (Συνεπίβλεψη: Βασίλης Μενεκλής, Δρ)**

Ημερομηνία Παράδοσης **Νοέμβριος 2013**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή) (υπογραφή) (υπογραφή)

Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

ΠΕΡΙΛΗΨΗ

Με την εμφάνιση του Web 2.0 στο χώρο του διαδικτύου και κυρίως με την τεράστια εξάπλωση των μέσων κοινωνικής δικτύωσης, η παραγωγή κειμενικών δεδομένων από τους εκατομμύρια χρήστες που τα χρησιμοποιούν καθημερινά αυξάνει συνεχώς. Η αποθήκευση των δεδομένων αυτών, καθώς και η δομή δεδομένων που θα χρησιμοποιείται, αποτελούν σημαντικά προβλήματα που επιζητούν λύσεις. Ένα άλλο σημαντικό θέμα είναι η δημιουργία μηχανών αναζήτησης που θα διευκολύνουν την εύρεση συγκεκριμένων πληροφοριών μέσα από αυτά.

Για την αντιμετώπιση της αποθήκευσης έχουν δημιουργηθεί οι βάσεις δεδομένων NoSQL που με τις ιδιαίτερες ιδιοτήτες τους και τη δομή δεδομένων JSON που χρησιμοποιούν αντιμετωπίζουν απόλυτα το πρόβλημα αυτό. Για τη διευκόλυνση της ανάπτυξης κατάλληλων μηχανών αναζήτησης έχουν δημιουργηθεί ειδικοί εξυπηρετητές αναζητήσεων, ανοιχτού κώδικα, όπως το ElasticSearch, που με την ευρεία γκάμα εργαλείων που διαθέτουν προσφέρουν λύσεις προς αυτήν την κατεύθυνση.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας διαδικτυακής εφαρμογής που, χρησιμοποιώντας τις παραπάνω τεχνολογίες, θα λειτουργεί ως μηχανή αναζήτησης για κειμενικά δεδομένα που αποθηκεύονται σε μια NoSQL βάση δεδομένων. Για τους σκοπούς της αναζήτησης χρησιμοποιούνται τόσο οι δυνατότητες της MongoDB, που αποτελεί την πιο γνωστή NoSQL βάση δεδομένων, αλλά και το ElasticSearch που διαθέτει περισσότερο εξειδικεύμενες λειτουργίες για το σκοπό αυτό.

ABSTRACT

The advent of Web 2.0 and the wide spread of social media are driving the daily increase of online textual data. Storage of such data and also the data structure that will be used to hold them are important problems, which seek solutions. Another important issue is the creation of search engines which will facilitate finding specific information within that big dataset.

NoSQL databases are a good way to deal with storage issues due to their particular properties and the JSON data structure they provide. ElasticSearch as a distributed, open source, search engine server and equipped with a broad range of tools, facilitates the development of search engines.

The purpose of this thesis is the development of a web application, which will use the above technologies and will serve as a search engine for textual data stored in a NoSQL database. MongoDB, as the most known NoSQL database, along with ElasticSearch co-operate, offering their specialized functions to achieve this goal.

Πίνακας Περιεχομένων

Κεφάλαιο 1 - Εισαγωγή

1.1 Περιγραφή του προβλήματος

Ο αρχικός σκοπός της παρούσας διπλωματικής εργασίας ήταν η ανάπτυξη μιας διαδικτυακής εφαρμογής που θα είχε ως βασική λειτουργικότητα την αναζήτηση πληροφορίας με δομημένο τρόπο, δηλαδή ορισμένα κριτήρια αναζήτησης με τη χρήση του ElasticSearch (Elasticsearch: Open Source Distributed Real Time Search & Analytics 2013). Συνοπτικά, το ElasticSearch αποτελεί έναν ανοιχτού κώδικα, κατανεμημένο εξυπηρετητή αναζητήσεων σε πραγματικό χρόνο (distributed, real time search server) που βοηθά στη δημιουργία μηχανών αναζήτησης και περιλαμβάνει μια ευρεία γκάμα κατάλληλων βοηθητικών εργαλείων που θα αναλυθεί επί μακρόν στη συνέχεια της εργασίας.
Στην πορεία της έρευνας που ξεκίνησε με το έναυσμα που δόθηκε από αυτήν την πρώτη ιδέα και μετά από μια σύντομη έρευνα στο χώρο των τεχνολογιών γύρω από τις σύγχρονες μηχανές αναζήτησης, θεωρήθηκε περισσότερο ενδιαφέρον και χρήσιμο αυτή η τεχνολογία να συνεργαστεί με κάτι άλλο επίσης πρωτοεμφανιζόμενο στο χώρο και εξίσου ενδιαφέρον. Επιλέχθηκε λοιπόν να μελετηθεί παράλληλα με το ElasticSearch η δυνατότητα μιας NoSQL (Vaish 2013) βάσης δεδομένων στο πεδίο της αναζήτησης πληροφορίας. Σκοπός της χρησιμοποίησης και των δύο τεχνολογιών δεν είναι η σύγκριση των δυνατοτήτων τους, αλλά η συνεργασία τους σε μια εφαρμογή που δουλεύει ως απλή μηχανή αναζήτησης. Η επιλογή αυτή προκρίθηκε, γιατί και οι δύο τεχνολογίες αποθηκεύουν τα δεδομένα τους βάσει μιας συγκεκριμένης και σύγχρονης δομής δεδομένων που ονομάζεται JSON (JavaScript Object Notation), η οποία στην ουσία περιγράφεται ως μια απλή απεικόνιση κειμενικών δεδομένων και συσχετισμένων πινάκων με τη μορφή αντικειμένων, αρκετά εύκολων στην αναγνώριση και στην κατανόησή τους από το ανθρώπινο μάτι. Η βάση δεδομένων που επιλέχθηκε ως αντιπροσωπευτική της NoSQL τεχνολογίας είναι η MongoDB (Banker 2012).
Ακολουθώντας λοιπόν αυτήν την πορεία έρευνας πάνω στο χώρο των τεχνολογιών των μηχανών αναζήτησης, έπρεπε να βρεθούν και τα κατάλληλα δεδομένα που θα αποτελούσαν το πεδίο των αναζητήσεων. Έχοντας ως κριτήρια για την επιλογή των δεδομένων την ποσότητα, την ευκολία πρόσβασης και τη συμβατότητα της δομής τους με τις υπό έρευνα τεχνολογίες, το καταλληλότερο μέσο παροχής τους κρίθηκε το Twitter (Russell 2011). Το Twitter ως ένα από τα πιο δημοφιλή κοινωνικά δίκτυα παρέχει ελεύθερα ένα μεγάλο μέρος των δεδομένων του που παράγονται από τους χρήστες του με μεγάλη ροή κειμενικών δεδομένων ανά δευτερόλεπτο και το κυριότερο, η δομή αυτών των δεδομένων προσφέρεται μέσω του API του Twitter με τη μορφή JSON.

1.2 Περιγραφή της εφαρμογής

Επομένως, έχοντας ως βασικό γνώμονα όλη την παραπάνω μελέτη του χώρου των μηχανών αναζήτησης και στόχο να δημιουργηθεί μια διαδικτυακή εφαρμογή, στην οποία ο χρήστης θα μπορεί να εκτελεί αναζητήσεις σε κειμενικά δεδομένα, χρησιμοποιώντας την ίδια στιγμή και τις

δύο προαναφερόμενες τεχνολογίες. Παράλληλα, έγινε προσπάθεια να μπορεί να κατανοεί ο χρήστης μέσω της εφαρμογής τα πλεονεκτήματα και τα μειονεκτήματα που έχει η καθεμιά από αυτές τις τεχνολογίες υλοποίησης στην ποιότητα των αποτελεσμάτων αφενός και αφετέρου στο χρόνο προσπέλασής τους.
Πρωταρχικό στάδιο της δομής της διαδικτυακής εφαρμογής είναι η συλλογή των κειμενικών δεδομένων, πάνω στα οποία θα γίνονται οι αναζητήσεις. Αυτό επιτυγχάνεται μέσω ενός σεναρίου εντολών (script), γραμμένο στη γλώσσα Python, το οποίο αφενός δημιουργεί τη σύνδεση με τη διεπαφή προγραμματισμού εφαρμογών (API) του Twitter και αφετέρου αποθηκεύει τη ροή των κειμενικών δεδομένων σε μια βάση NoSQL, της MongoDB. Πριν τα αποθηκεύσει διενεργεί δύο φιλτραρίσματα. Το πρώτο έχει ως σκοπό να επιτρέπει να αποθηκεύονται στη βάση, μόνο εκείνα τα tweets που είναι δηλωμένα ότι είναι γραμμένα στην αγγλική γλώσσα και το δεύτερο είναι υπεύθυνο να διατηρεί μόνο μερικά από τα υπάρχοντα πεδία JSON από τα πολλά που δίνει το API του Twitter. Συγκεκριμένα, η κάθε εγγραφή στη βάση περιέχει 10 πεδία (id, κείμενο, χρήστης, ημερομηνία, αριθμός followers, αριθμός followings, αριθμός tweets, αριθμός retweets και τοποθεσία).
Το δεύτερο στάδιο αποτελείται από την κατεξοχήν διαδικτυακή εφαρμογή, η οποία επικοινωνεί με τη βάση. Η εφαρμογή είναι κατασκευασμένη με το CodeIgniter που αποτελεί μια ανοιχτού κώδικα πλατφόρμα κατασκευής διαδικτυακών εφαρμογών. Βασικός σκοπός της εφαρμογής είναι η αναζήτηση πληροφορίας μέσω δύο τεχνολογιών, του ElasticSearch και της MongoDB. Συγκεκριμένα, δίνεται η δυνατότητα να επιλέξει ο χρήστης ανάμεσα σε δύο επιμέρους ιστοσελίδες ανάλογα με την τεχνολογία που θέλει να χρησιμοποιήσει. Οι αναζητήσεις που μπορεί να διενεργήσει σε καθεμιά από τις δύο ιστοσελίδες είναι δύο ειδών. Έχει τη δυνατότητα να αναζητήσει μια μόνο λέξη-κλειδί μέσα από το κείμενα των tweets και μπορεί να εκτελέσει και σύνθετη αναζήτηση με τρία κριτήρια, λέξη-κλειδί, τοποθεσία και εύρος ημερομηνιών. Η εμφάνιση των αποτελεσμάτων των αναζητήσεων γίνεται στην ίδια ιστοσελίδα ανά δεκάδες με ταυτόχρονη εμφάνιση του χρόνου προσπέλασης για την πρώτη δεκάδα αποτελεσμάτων, ώστε ο χρήστης να μπορεί να συγκρίνει και την απόδοση των δύο τεχνολογιών, πέραν από την ποιότητα, η οποία γίνεται εμφανής από τα ίδια τα αποτελέσματα.

1.3 Web 2.0 και Κοινωνικά Δίκτυα

Πριν προχωρήσει η περιγραφή των τεχνολογικών θεμάτων και η περαιτέρω μελέτη της εφαρμογής, κρίνεται χρήσιμο να αποσαφηνιστούν δύο όροι που συνδέονται άμεσα με την εφαρμογή, καθώς από εκεί πηγάζει η δεξαμενή των δεδομένων που διαχειρίζεται. Πρόκειται για τους όρους Web 2.0 και Κοινωνικά Δίκτυα που η εμφάνισή τους άλλαξε άρδην τον παγκόσμιο χάρτη του διαδικτύου.
Πριν την εμφάνιση του Web 2.0, περίπου στα μισά της προηγούμενης δεκαετίας, δύο ήταν οι λέξεις που μπορούσαν να περιγράψουν πιο πιστά το διαδίκτυο, στατικό και μονοδιάστατο. Το περιεχόμενο των ιστοσελίδων είχε τα προαναφερθέντα χαρακτηριστικά, οι ίδιες οι ιστοσελίδες ήταν κατασκευασμένες μόνο με χρήση HTML (Hypertext Markup Language) και CSS (Cascade Style Sheets) και οι χρήστες δεν μπορούσαν να έχουν καμία αλληλεπίδραση με τις ιστοσελίδες και το περιεχόμενό τους. Αυτή η πρώιμη μορφή του διαδικτύου προέβαλε μια πραγματικότητα, δοσμένη μόνο από την πλευρά του εκδότη της κάθε πληροφορίας – περιεχομένου και κανένας άλλος δεν μπορούσε ούτε να την αλλάξει ούτε να τη σχολιάσει. Φυσικά και ήταν ένα τεράστιο βήμα για την περαιτέρω εξέλιξη του διαδικτύου, αλλά με πολύ περιορισμένες δυνατότητες.

Το Web 2.0

Ο όρος Web 2.0 εισήχθη, για να περιγράψει την αλλαγή που επιτελέστηκε από τις πρώιμες στατικές ιστοσελίδες στις σύγχρονες δυναμικές. Εμπνευστής αυτού του όρου ήταν ο Darcy DiNuzzi (1999), αλλά το Web 2.0 έγινε γνωστό στην παγκόσμια κοινότητα μετά το 2005 από τον Dale Dougherty και από τον εκδοτικό οίκο O'Reilly Media (O'Reilly 2005). Σε γενικές γραμμές, το Web 2.0 δεν αναφέρεται σε κάποια νέα ανακάλυψη στο χώρο του διαδικτύου, αλλά σε μια ριζική αλλαγή στον τρόπο, με τον οποίο κατασκευάζονται οι ιστοσελίδες και στην αλληλεπίδραση που έχουν οι χρήστες με αυτές.
Πιο συγκεκριμένα, μια ιστοσελίδα Web 2.0 τεχνολογίας είναι μια ιστοσελίδα που επιτρέπει στους χρήστες να αλληλεπιδρούν και να συνεργάζονται στα πλαίσια μιας ψηφιακής κοινότητας, όπου υπάρχει η δυνατότητα δημιουργίας περιεχομένου από τον κάθε χρήστη ξεχωριστά, σε αντίθεση με τις ιστοσελίδες παλαιού τύπου. Μερικά χαρακτηριστικά παραδείγματα Web 2.0 ιστοσελίδων είναι:

- οι ιστοσελίδες κοινωνικής δικτύωσης (πχ Facebook)
- τα προσωπικά ιστολόγια (Blogs)
- οι ιστοσελίδες διαμοιρασμού videos (Youtube)
- τα Wikis, ιστοσελίδες δηλαδή, όπου συμμετοχικά οι χρήστες συντάσσουν κείμενα
- τα Mash-ups, δηλαδή ιστοσελίδες, που συνδυάζουν τη χρήση δεδομένων και εφαρμογών από διαφορετικές πηγές.

Συγκεκριμένα χαρακτηριστικά και λειτουργίες που διαθέτουν αυτές οι Web 2.0 ιστοσελίδες και στην ουσία αποτελούν την ειδοποιό διαφορά με τις παλιότερες τεχνολογίες είναι:

- Η δυνατότητα αναζήτησης μέσα στο περιεχόμενό τους, που επιτυγχάνεται μέσω λέξεων-κλειδιών, δοσμένες από το χρήστη.
- Οι υπερσύνδεσμοι (links) που συνδέουν με τέτοιο τρόπο το περιεχόμενο, ώστε να δημιουργείται ένας ιστός από συσχετιζόμενες πληροφορίες.
- Η δυνατότητα της συγγραφής και της ανανέωσης του περιεχομένου ως απόρροια συνεργατικής προσπάθειας πολλών χρηστών έναντι της απλής συγγραφής κειμένων που απηχούν τις ιδέες και τις αντιλήψεις ενός μόνο ατόμου. Παραδείγματος χάρη, σ' ένα Wiki ένα άρθρο μπορεί να συγγραφεί συνεργατικά από μια ομάδα ειδικών, ενώ ακόμα και σ' ένα ιστολόγιο η οπτική γωνία ενός συγγραφέα μπορεί να σχολιαστεί ποικιλοτρόπως από πολλαπλά σχόλια.
- Τα Tags, δηλαδή η χρήση μεμονωμένων λέξεων, όπου συγκεντρώνουν ευρύτερες έννοιες με σκοπό τη διευκόλυνση των αναζητήσεων. Συλλογές από Tags που έχουν δημιουργηθεί από πολλούς χρήστες και αφορούν συγκεκριμένο θέμα ονομάζονται ταξινομίες.
- Επεκτάσεις τεχνολογιών, οι οποίες μπορούν να ενσωματώσουν στην ιστοσελίδα πολυμεσικές πληροφορίες, όπως εικόνα και ήχο.
- Ειδοποιήσεις νέου περιεχομένου, όπως είναι το RSS (Real Simple Syndicate) feeds, ώστε οι χρήστες να παραμένουν συνεχώς ενήμεροι για τυχόν αλλαγές στο περιεχόμενο.

Τα Κοινωνικά Δίκτυα (Social Media)

Τα κοινωνικά δίκτυα μπορούν να χαρακτηριστούν ως μια από τις πιο σημαντικές εξελίξεις στο χώρο του διαδικτύου. Η ανάπτυξή τους οφείλεται κυρίως στην εξάπλωση των Web 2.0 τεχνολογιών αφενός και αφετέρου στον αντίκτυπο που συνεχίζουν να προκαλούν τα τελευταία χρόνια στην κοινωνική ζωή των ανθρώπων ανά τον κόσμο.
Ως κοινωνικά δίκτυα ορίζονται οι τρόποι, με τους οποίους διάφορα άτομα αλληλεπιδρούν μεταξύ τους, δημιουργώντας, διαμοιράζοντας και ανταλλάσσοντας πληροφορίες και ιδέες μέσα

στα πλαίσια εικονικών κοινοτήτων και δικτύων. Οι Andreas Kaplan και Michael Haenlein (2010) ορίζουν τα κοινωνικά δίκτυα ως μια ομάδα από διαδικτυακές εφαρμογές, οι οποίες δημιουργήθηκαν μετά την ιδεολογική και τεχνολογική εδραίωση του Web 2.0 και οι οποίες βοηθούν τη δημιουργία και την ανταλλαγή περιεχομένου μεταξύ των χρηστών. Η διείσδυσή τους στην καθημερινότητα είναι τόσο μεγάλη που έχει επιφέρει τεράστιες αλλαγές στον τρόπο επικοινωνίας, όχι μόνο των μεμονωμένων ανθρώπων, αλλά ακόμα και στον τρόπο που επικοινωνούν και αλληλεπιδρούν με το κοινό μεγάλοι παγκόσμιοι οργανισμοί και διάσημες εταιρείες.

Τα κοινωνικά δίκτυα διαφέρουν σε σχέση με τα παραδοσιακά μέσα σε πολλούς παράγοντες. Οι πιο σημαντικοί από αυτούς είναι η αμεσότητα, η οποία είναι πολλαπλάσια, καθώς ο χρήστης στην περίπτωση των μέσων κοινωνικής δικτύωσης έχει σε μεγάλο ποσοστό ενεργητικό ρόλο στην αλληλεπίδρασή του με το μέσο που χρησιμοποιεί, αλλά και με τους υπόλοιπους χρήστες που επικοινωνεί και η συχνότητα της χρήσης τους, μια και η πρόσβαση στα σύγχρονα αυτά μέσα διευκολύνεται ιδιαίτερα από την εξάπλωση των ευρυζωνικών δικτύων και των σύγχρονων έξυπνων κινητών τηλεφώνων. Σύμφωνα με στοιχεία της εταιρείας μετρήσεων Nielsen, οι χρήστες επισκέπτονται πολύ συχνότερα τα κοινωνικά δίκτυα σε σχέση με οποιεσδήποτε άλλες ιστοσελίδες. Επίσης, με την εμφάνισή τους, σημειώθηκε σημαντική αύξηση στο χρόνο που ξοδεύεται στη χρήση είτε ηλεκτρονικού υπολογιστή είτε κινητού τηλεφώνου. Στις ΗΠΑ η αύξηση αυτή περιήλθε στο 37% από 88 εκατομμύρια λεπτά τον Ιούλιο του 2011 σε 121 εκατομμύρια λεπτά τον Ιούλιο του 2012.

Οι τεχνολογίες που σχετίζονται με τα κοινωνικά δίκτυα είναι πολλών ειδών. Μερικά από αυτά ήδη προαναφέρθηκαν παραπάνω και γίνονται ακόμα πιο πολλά, καθώς υπάρχουν και συγκεκριμένες τεχνολογίες (aggregators) που συγκεντρώνουν σε μια ιστοσελίδα το περιεχόμενο από διάφορα άλλα μέσα. Οι Kaplan και Haenlein (2010), εφαρμόζοντας μια σειρά από θεωρίες που πηγάζουν τόσο από το χώρο έρευνας των παραδοσιακών μέσων (πχ. κοινωνική παρουσία) και των κοινωνικών διαδικασιών (αυτό-παρουσίαση, αυτό-αποκάλυψη), χώρισαν τα κοινωνικά δίκτυα σε έξι διαφορετικές κατηγορίες.

- Πλατφόρμες συνεργασίας (πχ. Wikipedia),
- Ιστολόγια και μικροϊστολόγια (πχ Twitter),
- Κοινότητες ανταλλαγής πολυμεσικού περιεχομένου (πχ Youtube, DailyMotion),
- Μέσα κοινωνικής δικτύωσης (πχ Facebook)
- Εικονικά παιχνίδια (πχ World of Warcraft) και
- Εικονικές κοινωνίες (πχ Second Life).

Πλεονεκτήματα και μειονεκτήματα κοινωνικών δικτύων

Όπως όλες οι μεγάλες εξελίξεις στο χώρο των επιστημών και των τεχνολογιών, έτσι και η εξάπλωση των κοινωνικών δικτύων στο χώρο του διαδικτύου ακολουθείται από συγκεκριμένα πλεονεκτήματα και μειονεκτήματα. Τα πλεονεκτήματα είναι εμφανή σε διάφορα πεδία τόσο της δημόσιας όσο και της προσωπικής ζωής. Βασική είναι η συνδρομή των κοινωνικών δικτύων στη διαφήμιση και στο ηλεκτρονικό εμπόριο, καθώς τα κοινωνικά δίκτυα αποτελούν ένα μέρος, όπου άμεσα και πολύ γρήγορα καταναλωτικά αγαθά και υπηρεσίες μπορούν να παρουσιαστούν σε ένα τεράστιο καταναλωτικό κοινό που έχει πρόσβαση σ' αυτά τα νέα μέσα επικοινωνίας. Επινοούνται συνεχώς νέοι τρόποι προώθησης προϊόντων και υπηρεσιών, χρησιμοποιώντας στο έπακρο τις δυνατότητες που προσφέρονται από τα κοινωνικά δίκτυα. Επίσης, βασικές αλλαγές παρατηρούνται και στον τρόπο που οι άνθρωποι επικοινωνούν μέσω των κοινωνικών δικτύων, καθώς η αμεσότητα και η ευκολία που προσφέρουν τείνουν να αλλάξουν σε μεγάλο βαθμό το χάρτη των προσωπικών σχέσεων των ανθρώπων, ιδιαίτερα στις νεαρές ηλικίες.

Από την άλλη πλευρά, μερικές πτυχές των κοινωνικών δικτύων έχουν αρνητικό αντίκτυπο τόσο σε κοινωνικά θέματα όσο και στην εξέλιξη του διαδικτύου. Τα κοινωνικά δίκτυα εξαιτίας της ανωνυμίας που προσφέρουν μπορούν πολύ εύκολα να αποτελέσουν πρόσφορο έδαφος για την ανάπτυξη αντικοινωνικών συμπεριφορών μεταξύ παιδιών (πχ cyber-bullying) όσο και κοινών εγκληματικών ενεργειών (πχ ηλεκτρονικές απάτες). Επίσης, από το πρίσμα της τεχνολογίας και της εξέλιξης του διαδικτύου ο ίδιος ο εμπνευστής του παγκόσμιου ιστού Tim Berners-Lee (2011) επισημαίνει πως τα μεγάλα κοινωνικά δίκτυα μετατρέπονται σε τεράστια μονοπώλια και αποτελούν τροχοπέδη τόσο στη δημιουργία νέων ιδεών όσο και τη μεταφορά δεδομένων από τη μια ιστοσελίδα στην άλλη, πράγμα πολύ χρήσιμο για την ελεύθερη διακίνηση περιεχομένου μέσα στον παγκόσμιο ιστό.

1.4 Το Twitter

Οι ανάγκες της παρούσας διπλωματικής εργασίας για κειμενικά δεδομένα, τα οποία θα βρίσκονταν ελεύθερα σε μεγάλες ποσότητες και σε μια εύκολα διαχειρίσιμη δομή, όπως το JSON , κατέστησαν τη μελέτη του Twitter μονόδρομο. Για να γίνουν περισσότερο κατανοητά τα τεχνικά θέματα που θα αναλυθούν παρακάτω, κρίνεται χρήσιμο να παρατεθούν μερικά εισαγωγικά σημεία για το συγκεκριμένο κοινωνικό δίκτυο που γνωρίζει τεράστια ακμή τα τελευταία χρόνια.

Δομή και λειτουργία του Twitter

Το Twitter είναι από τα πιο γνωστά και δημοφιλή κοινωνικά δίκτυα, έχει τη μορφή μικροϊστολογίου, όπου οι χρήστες έχουν τη δυνατότητα να συντάσσουν και να διαβάζουν “tweets”, δηλαδή περιορισμένα μέχρι 140 χαρακτήρες κείμενα. Πιο συγκεκριμένα, γραμμένοι στην υπηρεσία του δικτύου χρήστες μπορούν να διαβάζουν και να αναρτούν tweets, σε αντίθεση με τους αδήλωτους χρήστες, οι οποίοι μπορούν μόνο να τα διαβάζουν. Ο κάθε χρήστης διαθέτει έναν αριθμό χρηστών που μπορούν να διαβάζουν τις δικές του αναρτήσεις, γνωστοί και ως “followers”, αλλά ταυτόχρονα διαθέτει και ο ίδιος έναν αριθμό χρηστών που έχει επιλέξει να διαβάζει τις δικές του αναρτήσεις, γνωστοί και ως “followings”. Ο κάθε χρήστης, επίσης, μπορεί να δηλώσει την τοποθεσία, στην οποία βρίσκεται. Το Twitter είναι προσβάσιμο από διάφορες πηγές, όπως τη δική του ιστοσελίδα, από SMS και από έναν αρκετά μεγάλο αριθμό εφαρμογών τόσο του διαδικτύου όσο και κινητών τηλεφώνων.

Όσον αφορά τη σύνταξη ενός tweet, οι χρήστες μπορούν να ομαδοποιούν τα tweets ανά θεματική ενότητα με τη χρήση των λεγόμενων hashtags, δηλαδή συγκεκριμένων λέξεων ή φράσεων με το πρόθεμα “#”, όπου στην ουσία αποτελούν υπερσυνδέσμους σε όλα τα άλλα tweets που περιέχουν το ίδιο hashtag κάθε φορά. Ομοίως, το σύμβολο “@”, ακολουθούμενο από το προσωνύμιο κάποιου χρήστη, χρησιμοποιείται, για να γίνει αναφορά ή να δοθεί απάντηση σε άλλους χρήστες. Επίσης, για να επαναχρησιμοποιηθεί ένα ήδη αναρτημένο μήνυμα από κάποιον άλλον χρήστη και να διαμοιραστεί στους δικούς του followers, γνωστό και ως “retweet”, πρέπει να χρησιμοποιηθεί η λέξη “RT” μπροστά από το tweet που αναρτάται εκ νέου.

Πλεονεκτήματα του Twitter

Σε γενικές γραμμές, το Twitter έχει μετεξελιχθεί σε μια επανάσταση στο χώρο της επικοινωνίας, καθώς λόγω της αμεσότητάς του και του τηλεγραφικού του λόγου χρησιμοποιείται ευρέως για το σχολιασμό της επικαιρότητας, την αναμετάδοση γεγονότων που συμβαίνουν σε πραγματικό χρόνο, όπως κοινωνικών αναταραχών πχ τα γεγονότα της αραβικής άνοιξης, ως μέσο επαφής των εταιρειών με τους πελάτες τους ή των διασημοτήτων με τους θαυμαστές τους και ως μέσο

για την προώθηση προϊόντων και υπηρεσιών από τη μια, αλλά και ως δημόσιο βήμα για την έκφραση παραπόνων και κρίσεων για τα παραπάνω από την άλλη.

Τεχνολογικά, ολόκληρη η δομή του κάθε tweet συνοδεύεται από μια τεράστια συλλογή πληροφοριών, για την οποία θα γίνει αναλυτική αναφορά στη συνέχεια. Αυτό που πρέπει να σημειωθεί εδώ είναι ότι μέσω του API που προσφέρεται και της JSON μορφής που χρησιμοποιείται, γίνεται ιδιαίτερα εύκολη η διαχείριση όλων των κειμενικών δεδομένων που προσφέρονται.

1.5 Η δομή της διπλωματικής εργασίας

Η δομή της εργασίας έχει την ακόλουθη μορφή. Το πρώτο κεφάλαιο αφιερώθηκε στην περιγραφή του προβλήματος που η παρούσα εργασία κλήθηκε να μελετήσει, αλλά και της ίδιας της εφαρμογής που αναπτύχθηκε στα πλαίσια αυτής της μελέτης. Επίσης, έγινε αναφορά σε εισαγωγικές έννοιες που είναι άμεσα συνδεδεμένες με βασικά συστατικά της εργασίας.
Το δεύτερο κεφάλαιο θα αναλωθεί στη ανάλυση συγκεκριμένων εργασιών που καταπιάνονται με αναζήτηση πληροφορίας με δομημένο τρόπο και χρησιμοποιούν τεχνολογίες, όπως οι NoSQL βάσεις δεδομένων και το ElasticSearch. Επίσης, θα διενεργηθεί σύγκριση των εργασιών μεταξύ τους και θα επισημανθούν όλα τα συγκριτικά πλεονεκτήματα και μειονεκτήματά τους.
Το τρίτο κεφάλαιο θα έχει ως πυρήνα την περιγραφή των πιο σημαντικών τεχνολογιών που χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής. Η καθεμία ξεχωριστά θα αναλυθεί, ώστε να γίνουν κατανοητά τα βασικά τους χαρακτηριστικά και εν συνεχεία θα περιγραφεί ο ρόλος και η χρήση της καθεμιάς στη λειτουργικότητα της εφαρμογής.
Στο τέταρτο κεφάλαιο θα αναλυθεί επισταμένως η αρχιτεκτονική της εφαρμογής. Θα παρατεθούν UML διαγράμματα που θα αποσαφηνίζουν όλες τις πτυχές της, αλλά και περιγραφική ανάλυση όλων των λειτουργιών ξεχωριστά.
Τέλος, στο τελευταίο κεφάλαιο θα παρατεθούν όλα τα συμπεράσματα που εξήχθησαν κατά τη διάρκεια ανάπτυξης της εφαρμογής, όπως δυσκολίες που αντιμετωπίστηκαν, γνώση, η οποία κατακτήθηκε από την προσπέλαση και μη αυτών των δυσκολιών, αλλά και διάφοροι περιορισμοί που ανακαλύφθηκαν από την εφαρμογή των συγκεκριμένων τεχνολογιών. Εν κατακλείδι, θα γίνει λόγος για μελλοντικές επεκτάσεις, οι οποίες μπορούν να κάνουν την παρούσα εφαρμογή περισσότερο εύχρηστη και ελκυστική ως προς το χρήστη, αλλά και για περαιτέρω τεχνολογίες που μπορούν να προστεθούν στη λειτουργικότητά της.

Κεφάλαιο 2 - Βιβλιογραφία και παρουσίαση παρόμοιων εφαρμογών

2.1 Εισαγωγή

Το κεφάλαιο που ακολουθεί ασχολείται με δύο θέματα. Το πρώτο είναι η παράθεση της βιβλιογραφίας που χρησιμοποιήθηκε για την κατασκευή της εφαρμογής και την τεκμηρίωση του θεωρητικού της υπόβαθρου και το δεύτερο είναι η σύγκριση της λειτουργικότητας και της ιδέας της εφαρμογής με άλλες παρόμοιες. Κυρίως πρόκειται για εγχειρήματα, όπου κειμενικά δεδομένα που προέρχονται από κοινωνικά δίκτυα, αλλά και από άλλες πηγές και κύριο χαρακτηριστικό τους είναι ο μεγάλος όγκος τους αποθηκεύονται σε NoSQL βάσεις δεδομένων και ευρετηριάζονται με την βοήθεια του ElasticSearch, ώστε να διευκολύνεται η αναζήτηση μέσα σ' αυτά.

2.2 Βιβλιογραφία

Για τις ανάγκες της ανάπτυξης της εφαρμογής και της συγγραφής της παρούσας διπλωματικής εργασίας χρησιμοποιήθηκαν αρκετά είδη πηγών. Αρχικά χρησιμοποιήθηκε συγκεκριμένη βιβλιογραφία, ιδιαίτερα για τα περισσότερο θεωρητικά ζητήματα, όσο και ιστοσελίδες που περιλαμβάνουν εγχειρίδια για τη χρήση συγκεκριμένων συστατικών της εφαρμογής, ιστολόγια που περιέχουν άρθρα για μια σειρά θεμάτων που αφορούσαν την υλοποίηση συγκεκριμένων τμημάτων του κώδικα και λύσεις σε δυσκολίες τόσο σε υλοποίηση κώδικα όσο και εγκατάσταση τεχνολογιών που έπρεπε να δουλεύουν σε απόλυτη αντιστοιχία μεταξύ τους και ήταν αναγκαίο να ακολουθηθούν συγκεκριμένα βήματα κατά τη διαδικασία εγκατάστασής τους. Ιδιαίτερα σημαντική βοήθεια ήταν επίσης ιστοσελίδες, όπου αποτελούν κοινωνικά δίκτυα μεταξύ προγραμματιστών, όπως το GitHub, από το οποίο αντλήθηκαν βιβλιοθήκες που αποτέλεσαν στήριγμα για την ανάπτυξη ορισμένων λειτουργιών της εφαρμογής. Αλλά και ιστοσελίδες διαδικτυακών συζητήσεων (forums), όπως το stackoverflow.com, όπου και βρέθηκαν λύσεις σε δυσκολίες κατά την ανάπτυξη του κώδικα. Ειδικότερα, η παράθεση της βιβλιογραφίας και όλων των συναφών πηγών που προαναφέρθηκαν θα χωριστεί ανάλογα με τα διάφορα τμήματα της εφαρμογής, όπως θα αναλυθούν και παρακάτω.

Python - Twitter

Αρχικά λοιπόν για τη δημιουργία του αρχείου σεναρίου εντολών (script file), το οποίο φιλτράρει και αποθηκεύει τα tweets σε μια MongoDB βάση δεδομένων βοήθησαν οι ιδέες του Russell (2011) για τη διαχείριση και την επεξεργασία των κειμενικών δεδομένων που προσφέρει το Twitter μέσω του API του. Συγκεκριμένα, ο τρόπος φιλτραρίσματος και αποθήκευσης των κειμενικών δεδομένων του Twitter παρουσιάζεται στο How to Capture Tweets (2011), όπου και αναλύονται όλες οι επιμέρους παράμετροι, όπως η χρήση των βιβλιοθηκών Tweepy και Pymongo και του Twitter API. Ο κώδικας αυτός παραμετροποιήθηκε κατάλληλα σύμφωνα με τις ανάγκες της εφαρμογής. Για την περαιτέρω χρήση της Python και την ανάπτυξη του κώδικα ως σημείο αναφοράς χρησιμοποιήθηκαν το διαδικτυακό εγχειρίδιο της γλώσσας (python.org 2013) και για την κατανόηση των παραδειγμάτων για τη διαχείριση του Twitter API συνέβαλε ο Beazley (2006).

NoSQL - MongoDB

Συνεχίζοντας με την υποδομή του πίσω μέρους (backend) της εφαρμογής, για την κατανόηση του τρόπου λειτουργίας της MongoDB βάσης δεδομένων σημαντική ήταν η εξήγηση για τη γενικότερη φιλοσοφία, δομή και λειτουργία των NoSQL βάσεων δεδομένων και τις διαφορές που έχουν αυτές οι βάσεις με τις παραδοσιακές σχεσιακές βάσεις (SQL), όπως αυτή αναπτύσσεται από τον Vaish (2013). Ιδιαίτερα το γεγονός ότι οι NoSQL βάσεις δεδομένων είναι ιδανικές για παρουσίαση δεδομένων χωρίς την προϋπόθεση συγκεκριμένου σχήματος εξαιτίας της αποθήκευσης των δεδομένων σε μορφή JSON (Vaish 2013, p. 11) οδήγησε στην επιλογή ενός τέτοιου είδους βάσης ως βασικής επιλογής για τις ανάγκες της εφαρμογής. Για τη διαχείριση και τη λειτουργία της ίδιας της MongoDB, αλλά και για τα θετικά της χαρακτηριστικά που την ξεχώρισαν για τις συγκεκριμένες ανάγκες ανάμεσα από τις υπόλοιπες βάσεις του είδους της συνέβαλε ο Banker (2012). Βασικός πυλώνας για την κατανόηση της λειτουργίας της MongoDB υπήρξε και το διαδικτυακό εγχειρίδιο της MongoDB (mongodb.org 2013) και περισσότερο οι οδηγίες για την ενεργοποίηση των replica set oplog (operations log) (Replication 2013).

ElasticSearch

Όσον αφορά το ElasticSearch εξαιτίας του γεγονότος ότι πρόκειται για πολύ καινούργια τεχνολογία η βιβλιογραφία είναι περιορισμένη και οι υποστηρικτικές πηγές που υπάρχουν είναι το εγχειρίδιο της επίσημης ιστοσελίδας (elasticsearch.org), που χρησιμοποιήθηκε κατά κόρον, διάφορα άρθρα από ιστολόγια και ιστοσελίδες γραμμένα από χρήστες και το GitHub, όπως και άλλες ιστοσελίδες διαδικτυακών συζητήσεων με προεξάρχουσα το stackoverflow.com. Από τις λίγες πηγές που προχωράνε ένα βήμα πιο πέρα είναι το εγχειρίδιο των (Rafał Kuć, Marek Rogoziński 2013) και που βοηθάει ιδιαίτερα στην κατανόηση τόσο της λειτουργίας όσο και της ίδιας της δομής του ElasticSearch και σε θεωρητικό και σε πρακτικό επίπεδο. Βασικό επίσης για την κατανόηση του θεωρητικού υπόβαθρου του ElasticSearch στάθηκε και η μελέτη του Lucene και των υποδομών που κληροδότησε για την ανάπτυξη του ElasticSearch, όπως περιγράφονται με άρτιο τρόπο από τους (Hatcher, Gospodnetić & McCandless 2009).

Ανάπτυξη εφαρμογής

Πέρα όμως από τα παραπάνω που κατά κύριο λόγο κάλυψαν τις θεωρητικές ανάγκες κατανόησης, υπάρχουν πηγές που συνετέλεσαν επικουρικά στην επίλυση πρακτικών δυσκολιών. Συγκεκριμένα, για την δημιουργία της ανάλυσης και της χαρτογράφησης βοήθησαν οι αναλυτικές οδηγίες του Cholakian (2013), τα άρθρα Constructing more complicated mapping (2012) και ElasticSearch 101 (2013). Στις δύο τελευταίες πήγες υπάρχουν αναλυτικές οδηγίες βήμα προς βήμα για τη δημιουργία αναλύσεων και χαρτογραφήσεων που εκτείνονται από απλές μέχρι και αρκετά σύνθετες περιπτώσεις. Επίσης, πολύ χρήσιμη πηγή για τη συνεργασία της MongoDB και του ElasticSearch μέσω της χρήσης του άρθρώματος MongoDB River αποτελεί το εγχειρίδιο του Willy (2012) και το άρθρο A complete guide to Integrating MongoDB (2012).

Για το εμπρός μέρος (frontend) μέρος της εφαρμογής βασική πηγή για την ανάπτυξή του υπήρξε το διαδικτυακό εγχειρίδιο του CodeIgniter (ellislab.com/codeigniter 2013), όπου περιέχει αναλυτικές οδηγίες και παραδείγματα για όλες τις βιβλιοθήκες και τις κλάσεις που παρέχει αυτό το πλαίσιο λογισμικού PHP, καθώς και θεωρητική ανάλυση της αρχιτεκτονικής Μοντέλου – Άποψης – Ελεγκτή, πάνω στην οποία βασίζεται. Συμπληρωματικά, χρησιμοποιήθηκε και το εγχειρίδιο της PHP στο διαδίκτυο (php.net 2013). Κύριες πηγές για την σύνδεση του CodeIgniter με την MongoDB, καθώς η πρώτη δεν διαθέτει εγγενώς τέτοια βιβλιοθήκη, όπως για την SQL παραδείγματος χάρη, υπήρξε το εγχειρίδιο της βιβλιοθήκης που συνδέει τις δύο τεχνολογίες στο GitHub του Bilbie (2013) και το άρθρο Using MongoDB and CodeIgniter On Windows (2011), όπου περιγραφεί τη σύνδεση πρώτα της PHP με την MongoDB.

Για τη συνεργασία μεταξύ του CodeIgniter και του ElasticSearch κύριος αρωγός υπήρξε το εγχειρίδιο της βιβλιοθήκης Elastica (elastica.io 2013), όπου αναλύονται επισταμένως όλες οι επιμέρους κλάσεις, το αυτόματο φόρτωμα της βιβλιοθήκης από το CodeIgniter και το στήσιμο μιας ανάλυσης και μιας χαρτογράφησης με τη χρήση της Elastica. Βασική βοήθεια για το τελευταίο προσέφερε και το άρθρο Using Elastica to query ElasticSearch (2012).
Κατά τα λοιπά, για τη δημιουργία της γραφικής απεικόνισης της ιστοσελίδας και των υπολοίπων λειτουργιών της χρησιμοποιήθηκαν κυρίως τα εγχειρίδια του Twitter Bootstrap (getbootstrap.com 2013) και της βιβλιοθήκης της Javascript, JQuery (jquery.com 2013). Για την ένωση του CodeIgniter και του Twitter Bootstrap χρησιμοποιήθηκε το εγχειρίδιο της βιβλιοθήκης CodeIgniter-Bootstrap από το GitHub.

Κλείνοντας το κεφάλαιο αυτό, για τη δημιουργία των διαγραμμάτων UML στο κεφάλαιο 4 ως πηγή χρησιμοποιήθηκε το άρθρο UML basics: An introduction (2003) και η θεωρητική ανάλυση του Schmuller (2004).

2.2 Σύγκριση με παρόμοιες εργασίες

Η εμφάνιση τόσο των NoSQL βάσεων δεδομένων όσο και του ElasticSearch ήταν αρκετά πρόσφατη, γι' αυτό ακόμα δεν έχουν αφήσει το στίγμα τους στο χώρο της πληροφορικής πολλές εφαρμογές που να έχουν παρόμοιο πεδίο εργασίας και προβληματισμού. Η παρούσα εφαρμογή εξάλλου δεν φιλοδοξεί να σταθεί ως μια αυτόνομη εφαρμογή, αλλά κυρίως θα είναι ιδιαίτερα χρήσιμη, αν αποτελεί ένα συγκεκριμένο τμήμα κάτι μεγαλύτερου. Αυτές οι σκέψεις θα παρουσιαστούν εκτενέστερα στο τελευταίο κεφάλαιο της παρούσας εργασίας. Για τους παραπάνω λόγους βρέθηκαν κάποιες εργασίες, που βέβαια δεν μπορούν να συγκριθούν απόλυτα με την παρούσα εφαρμογή, γιατί το πεδίο τους είναι αρκετά μεγαλύτερο, αλλά ένα κομμάτι τους, κυρίως αυτό που σχετίζεται με τη συγκέντρωση και αποθήκευση κειμενικών δεδομένων και η ευρετηρίαση τους, ώστε σε επόμενο στάδιο να είναι περισσότερο εύκολη η ανάκτησή τους, βρίσκεται σε πλήρη αντιστοίχιση.
Ο κοινός παρονομαστής όλων των εφαρμογών είναι αυτό που περιγράφηκε παραπάνω, δηλαδή από κάπου εισρέουν δεδομένα που πρέπει να αποθηκευτούν, με τέτοιο τρόπο, ώστε μετέπειτα τα δεδομένα αυτά να μπορούν να ανακτηθούν μέσω κάποιας υπηρεσίας αναζήτησης. Τα αποτελέσματα αυτά των αναζητήσεων θα μπορούν να αποτελούν πεδίο για περαιτέρω έρευνες που ξεπερνούν το παρόν αντικείμενο, όπως είναι η εξόρυξη πληροφορίας και η μηχανική μάθηση.

ThemeStreams

Ξεκινώντας λοιπόν την παρουσίαση των σχετικών εργασιών και εφαρμογών, η πρώτη που βρίσκεται σε αρκετά μεγάλη συνάφεια με τη λογική της παρούσας εργασίας είναι το ThemeStreams (Rooij, Odijk & Rijke 2013). Αυτή η εφαρμογή αντλεί δεδομένα από το Twitter API και αποθηκεύει κειμενικά δεδομένα που δημοσιεύονται από συγκεκριμένες κατηγορίες χρηστών που όλοι σχετίζονται με την πολιτική σκηνή της Ολλανδίας. Αυτοί οι χρήστες είναι χωρισμένοι σε 4 κατηγορίες, πολιτικοί, δημοσιογράφοι, στελέχη πολιτικών κομμάτων ή άλλων οργανισμών που σχετίζονται ευθέως με κάποιο κόμμα ή πολιτικό και τέλος ηθοποιοί και διασημότητες που εκφέρουν στα tweets τους έντονο πολιτικό λόγο. Μέχρι τη στιγμή που γραφόταν η εν λόγω αναφορά τους είχαν καταφέρει να συλλέξουν περίπου 3,5 εκατομμύρια tweets από όλα τα είδη των παραπάνω χρηστών που προσεγγιστικά φθάνουν τον αριθμό των 245 χιλιάδων. Στη συνέχεια από ένα γραφικό περιβάλλον μιας ιστοσελίδας οι χρήστες μπορούν να ανακτήσουν τα tweets, έχοντας δύο κύριες επιλογές είτε να διαλέξουν ένα από τα

προκαθορισμένα θέματα και να δουν πως τα tweets σχετικά με αυτά αυξομειώνονται σε όγκο στη μονάδα του χρόνου, παράλληλα και με τις πιο συχνές λέξεις που διαθέτουν είτε να διαλέξουν κάποια λέξη-κλειδί και επίσης να δουν γραφικά τη συχνότητά της μέσα στο χρόνο μαζί με τις πιο συχνές λέξεις γύρω από αυτήν.
Αυτό όμως που είναι ιδιαίτερα ενδιαφέρον είναι ότι η συγκέντρωση όλων των δεδομένων γίνεται απευθείας στο ElasticSearch που στην ουσία το χρησιμοποιούν ως την κύρια βάση δεδομένων τους. Αυτή η επιλογή, όπως θα αναλυθεί και στα συμπεράσματα στο πέμπτο κεφάλαιο, έχει μια σειρά από προβλήματα. Μερικά από αυτά είναι η έλλειψη ασφάλειας, διάρκειας (durability) και διαθεσιμότητας (availability) των δεδομένων και η μη ύπαρξη αρκετά εξελιγμένων βιβλιοθηκών και εργαλείων για περαιτέρω διευκόλυνση της διαχείρισης των δεδομένων. Όλα αυτά συντείνουν πως η ύπαρξη ενός ενδιάμεσου επιπέδου αποθήκευσης των δεδομένων σε μια NoSQL βάση δεδομένων λύνει τα παραπάνω προβλήματα και δεν αποτελεί και ιδιαίτερα δύσκολο πρόβλημα, καθώς υπάρχει η κατάλληλη υποδομή (Rivers), ώστε τα δεδομένα να οδηγούνται από τη NoSQL βάση προς το ElasticSearch.

AVResearcher

Την ίδια ακριβώς αρχιτεκτονική στην εφαρμογή τους ακολούθησαν και οι Huurnink et al (2013). Η εφαρμογή τους ονομάζεται AVResearcher και σκοπό έχει την αναζήτηση μεταδεδομένων συνδεδεμένων με έναν μεγάλο αριθμό από οπτικοακουστικές εκπομπές. Οι εκπομπές αυτές μπορούν να αναζητηθούν από τους χρήστες όχι μόνο βάσει των παραδοσιακών καταλόγων που περιγράφουν το εκάστοτε πρόγραμμα, αλλά μέσω των υποτίτλων τους και των αναφορών τους στο Twitter. Τα αποτελέσματα επιστρέφονται ανάλογα με το θέμα και το χρόνο δημοσίευσής τους.
Η αρχιτεκτονική του εγχειρήματος είναι αρκετά απλή. Τα δεδομένα που είναι προς αποθήκευση αντλούνται από ξεχωριστές πηγές, δηλαδή οι κατάλογοι περιγραφής των εκπομπών από το ολλανδικό Ινστιτούτο Ήχου και Εικόνας, οι υπότιτλοι από τους υπεύθυνους των ίδιων των ολλανδικών εκπομπών και τα tweets από το Twitter API. Συγκεκριμένα για το Twitter, αποθηκεύονται 25 επίσημες θεματικές ενότητες (hashtags) που αφορούν ισάριθμες τηλεοπτικές εκπομπές. Το σημαντικό είναι ότι όλα αυτά τα δεδομένα, παρόλη την ανομοιογένεια που ενδεχομένως έχουν, ευρετηριάζονται και αποθηκεύονται με τη βοήθεια του ElasticSearch και χωρίς τη μεσολάβηση κάποιου ενδιάμεσου επιπέδου. Οι συντάκτες της εργασίας δεν αναφέρουν λεπτομέρειες για τις διευκολύνσεις που τους παρέχει αύτη τους η επιλογή, αλλά σίγουρα θα αντιμετωπίζουν τις δυσκολίες της χρήσης μόνο ενός επιπέδου αποθήκευσης, όπως αυτές εξηγήθηκαν και στην παραπάνω εφαρμογή.

Παράλληλα, οι χρήστες μπορούν να έχουν πρόσβαση στα δεδομένα μέσω ενός γραφικού περιβάλλοντος που προσφέρει μια ιστοσελίδα και για κάθε αναζήτησή τους έχουν τη δυνατότητα να βλέπουν τον αριθμό των εκπομπών που περιέχουν στην περιγραφή τους τη λέξη-κλειδί, τη χρονική κατανομή των αποτελεσμάτων και τη συχνότητά των λέξεων στα αποτελέσματα που βρέθηκαν. Υπάρχει επίσης η δυνατότητα για ταυτόχρονη αναζήτηση δύο λέξεων-κλειδιών και εμφάνιση στην ίδια οθόνη των αντίστοιχων αποτελεσμάτων.

Kongress

Στο ίδιο μήκος κύματος κινείται και η εφαρμογή Kongress (Grosvenor, Kendall & Sanders 2012). Η εφαρμογή έχει ως κύριο πεδίο δράσης την εξόρυξη πληροφορίας από δεδομένα που προέρχονται από μέσα κοινωνικής δικτύωσης, στην προκειμένη φάση της μόνο από το Twitter. Η εφαρμογή προορίζεται να λειτουργεί μόνο ως υπηρεσία σε έξυπνα τηλέφωνα (smart phones) και ο κύριος σκοπός της είναι η συλλογή tweets με τη χρήση του Twitter API από μέλη του αμερικανικού Κογκρέσου, τους χρήστες που αυτά τα μέλη ακολουθούν (followers), αλλά και χρήστες που ακολουθούν τα ίδια τα μέλη (followings), από τα όποια θα βγαίνει το συμπέρασμα

ποιος χρήστης – μέλος του Κογκρέσου επηρεάζει περισσότερο τους συναδέλφους του μέσα στο Κογκρέσο και ποιοι χρήστες followers ή followings ασκούν μεγαλύτερη επιρροή στα μέλη του Κογκρέσου. Η εφαρμογή αυτή πηγαίνει το όλο εγχείρημα ακόμα ένα βήμα πιο πέρα, καθώς θα αποπειράται να οπτικοποιεί αυτές τις σχέσεις αλληλοεπίδρασης μέσω ενός οπτικού δικτύου ανάλυσης, το οποίο μάλιστα θα έχει και γεωγραφικές διαστάσεις, αφού η θέση όλων των χρηστών θα απεικονίζεται πάνω σε γεωγραφικό χάρτη.
Όλα αυτά είναι ιδιαιτέρως ενδιαφέροντα και αποδεικνύουν πως το φιλτράρισμα και η αποθήκευση δεδομένων από κοινωνικά δίκτυα με συντεταγμένο τρόπο είναι το πρώτο βήμα, για να εφαρμοστούν πάνω σ' αυτά πολύ πιο σύνθετες και εξειδικευμένες αναλύσεις που μπορούν να οδηγήσουν σε άκρως ενδιαφέροντα συμπεράσματα και διαπιστώσεις. Γι' αυτό το πρώτο βήμα λοιπόν η παρούσα εφαρμογή χρησιμοποιεί το ElasticSearch ως κύρια μονάδα ευρετηρίασης και αποθήκευσης των δεδομένων.
Από αυτήν την εφαρμογή μπορεί εύκολα να αναγνωριστεί η υπεροχή του ElasticSeach στο χώρο των μηχανών αναζήτησης σε πραγματικό χρόνο και ταυτόχρονα αποδεικνύεται πως αυτές οι δυνατότητες δεν θα μπορούσαν να εξασφαλιστούν από καμία βάση δεδομένων είτε σχεσιακή είτε NoSQL. Στο Kongress λοιπόν ο χρήστης αναζητώντας μια λέξη-κλειδί μπορεί να δει αποτελέσματα από τρεις διαφορετικές πηγές, δηλαδή από τις ψηφοφορίες των μελών του Κογκρέσου, από τα tweets που έχουν δημοσιεύσει τα ίδια, αλλά και οι followers και οι followings τους. Για να μπορούν αυτές οι αναζητήσεις να επιστρέφουν σωστά αποτελέσματα, οι δημιουργοί της εφαρμογής έχουν κατασκευάσει 5 διαφορετικά ευρετήρια αντί για ένα. Διαθέτουν ξεχωριστά ευρετήρια για τα νομοσχέδια που ψηφίστηκαν, για τα μητρώα ψήφων των μελών του Κογκρέσου, για στοιχεία του κάθε χρήστη στο Twitter (id, url χρήστη κτλ), για τα ίδια τα tweets και επιπλέον ένα που αποθηκεύει τις συνδέσεις μεταξύ των μελών και των followers/followings τους. Όλα αυτά τα ευρετήρια και κυρίως το τελευταίο έχουν ως σκοπό την επιτάχυνση των αναζητήσεων και την επίτευξη μεγαλύτερης ακρίβειας στην ανάκτηση των αποτελεσμάτων.

Γενικότερα, η υιοθέτηση πολλαπλών ευρετηρίων είναι μια ιδέα που μπορεί να υιοθετηθεί και για την παρούσα εφαρμογή της διπλωματικής εργασίας σε μετέπειτα στάδια, όπου το πεδίο των αναζητήσεων δεν θα είναι μόνο τα tweets, αλλά και άλλα δεδομένα από αυτά που δίνει το Twitter API. Μ' αυτόν τον τρόπο μπορούν οι αναζητήσεις να γίνουν περισσότερο σύνθετες και ο χρόνος επιστροφής αποτελεσμάτων μικρότερος. Επίσης, όπως φάνηκε και από τις προηγούμενες εργασίες, το ElasticSearch και η ευρετηρίαση των δεδομένων συντελούν, ώστε ανομοιόμορφα δεδομένα από διαφορετικές πηγές να μπορούν να συγχωνευτούν σε μια κοινή βάση δεδομένων, απ' όπου επιστρέφονται ομοιογενή αποτελέσματα. Αυτή η συγχώνευση διάφορων δεδομένων εξάλλου μπορεί να διευκολύνει και πιο σύνθετες αναλύσεις που οδηγούν σε πολύ πλούσια και πολύπλοκα συμπεράσματα.

Αναζήτηση στο Liber Usualis

Το παρόν κεφάλαιο κλείνει με μια εφαρμογή που παρόλο που βασίζεται σε διαφορετικό γνωστικό πεδίο απ' όσα μελετήθηκαν μέχρι τώρα, ακολουθεί παρόμοια τακτική στην ευρετηρίαση και αποθήκευση δεδομένων. Πρόκειται για μια εφαρμογή (Thompson, Hankinson & Fujinaga 2012), της οποίας τα δεδομένα προέρχονται από μια έντυπη έκδοση του 1961, 2340 σελίδων, ύμνων της καθολικής εκκλησίας με το όνομα Liber Usualis. Αυτή, αφού ψηφιοποιήθηκε μέσω λογισμικού οπτικής αναγνώρισης χαρακτήρων (OCR), τα ηλεκτρονικά έγγραφα που προέκυψαν αποθηκεύτηκαν σε μια NoSQL βάση δεδομένων και συγκεκριμένα στην CouchDB. Τα ηλεκτρονικά αυτά έγγραφα δεν αποτελούνται μόνο από κείμενα, αλλά και από μουσικές νότες. Για την ευρετηρίαση και την αποθήκευση των εγγράφων χρησιμοποιούνται συνδυαστικά τόσο η CouchDB όσο και το ElasticSearch.

Το σημαντικό που πρέπει να σημειωθεί από αυτήν την εφαρμογή είναι πως η ευρετηρίαση των εγγράφων δεν έγινε στο επίπεδο του ElasticSearch, όπως στην εφαρμογή της παρούσας διπλωματικής, αλλά κατά το πρώτο επίπεδο της αποθήκευσης στη βάση δεδομένων, κάνοντας χρήση των δυνατοτήτων ευρετηρίασης των NoSQL βάσεων δεδομένων. Κυρίως επειδή πρόκειται κατά κύριο λόγο για μη εξ ολοκλήρου κειμενικά δεδομένα, δημιουργήθηκαν πολλαπλές συλλογές μέσα στην βάση δεδομένων, όπου για κάθε συλλογή υπήρχε διαφορετικό ευρετήριο με τη χρήση της μεθόδου n-gram. Δηλαδή μια συλλογή είχε χωρισμένα τα δεδομένα με 2-gram, άλλη με 3-gram κοκ. Σύμφωνα με τους δημιουργούς της εφαρμογής αυτή η επιλογή, παρόλο που δημιουργεί πληθώρα εγγραφών, βοηθά στην ταχύτερη ανάκτηση των δεδομένων που είναι μη κειμενικά δεδομένα και η πληροφορία βρίσκεται σε επίπεδο νότας, δηλαδή ενός χαρακτήρα και όχι σε επίπεδο λέξης, όπου οι χαρακτήρες είναι περισσότεροι του ενός. Σε σύγκριση με την εφαρμογή της διπλωματικής, η τακτική αυτή ευρετηρίασης πρώτα στο επίπεδο της NoSQL βάσης δεδομένων θα ήταν εφικτή, όχι τόσο για λόγους ταχύτερης ανάκτησης όσο περισσότερο για λόγους σύγκρισης αυτών των ευρετηρίων με των αντίστοιχων του ElasticSearch, αλλά η απουσία ολοκληρωμένης βιβλιοθήκης συνεργασίας της MongoDB με το CodeIgniter αποτέλεσε εμπόδιο αυτής της προσπάθειας.

Συμπεράσματα

Σε γενικές γραμμές από τις παραπάνω εφαρμογές και εργασίες που παρουσιάστηκαν φαίνεται πως η επιλογή του ElasticSearch ως κύριου τρόπου ευρετηρίασης και αποθήκευσης δεδομένων που προέρχονται από το Twitter είναι ιδιαίτερα χρήσιμη και αυτό οφείλεται κυρίως στο γεγονός ότι τόσο τα δεδομένα του Twitter API δίνονται σε μορφή JSON όσο και ο τρόπος αποθήκευσης στο ElasticSearch βασίζεται σ' αυτήν την μορφή. Η επιλογή αυτή γίνεται, παρόλες τις αδυναμίες του ElasticSearch που υπογραμμίστηκαν παραπάνω. Τέλος, φάνηκε πως και η συνεργασία βάσεων δεδομένων NoSQL και ElasticSearch αποτελεί μια λύση που οδηγεί σε θετικά αποτελέσματα εξαιτίας της ομοιότητας της χρησιμοποιούμενης δομής δεδομένων και της απουσίας σχήματος.

Κεφάλαιο 3 - Παρουσίαση τεχνολογιών

3.1 Εισαγωγή

Το παρόν κεφάλαιο έχει ως σκοπό πρώτα να παραθέσει και κατόπιν να περιγράψει τις πιο σημαντικές τεχνολογίες που χρησιμοποιήθηκαν για την κατασκευή της εφαρμογής. Επίσης, θα γίνει προσπάθεια να καταγραφούν τα θετικά και τα τυχόν αρνητικά σημεία των τεχνολογιών, όπως αυτά έγιναν κατανοητά κατά τη διάρκεια της ανάπτυξης της εφαρμογής. Παράλληλα, με τα

παραπάνω θα καταδειχθεί και ο ρόλος των τεχνολογιών με την περιγραφή της χρήσης τους στα διάφορα συστατικά που συνθέτουν τις λειτουργίες της εφαρμογής.

3.2 NoSQL Βάσεις Δεδομένων

Ξεκινώντας την περιγραφή από τη βάση δεδομένων, η οποία αποτελεί ένα από τα βασικά συστατικά στοιχεία της εφαρμογής, καθώς εκεί εδρεύουν όλα τα δεδομένα που διαχειρίζεται η εφαρμογή, θα γίνει αναφορά αρχικά στον τύπο της βάσης και στη συνέχεια θα περιγραφεί η ίδια η βάση δεδομένων. Πρόκειται για μια βάση δεδομένων τύπου NoSQL και η βάση που χρησιμοποιήθηκε για την υλοποίηση είναι η MongoDB.

Από τις σχεσιακές στις NoSQL βάσεις δεδομένων

Ο πιο γνωστός τύπος βάσεων δεδομένων είναι οι σχεσιακές βάσεις δεδομένων, οι οποίες πρωτοεμφανίστηκαν τη δεκαετία του 70. Αυτές οι βάσεις δεδομένων, πασίγνωστες σήμερα, δημιουργήθηκαν, έτσι ώστε να αποθηκεύουν δεδομένα βάσει ενός προκαθορισμένου μοντέλου και να προσπελαύνονται βάσει μιας συγκεκριμένης γλώσσας δημιουργίας ερωτημάτων, τη Δομημένη Γλώσσα Ερωτημάτων (SQL). Εκείνη την εποχή, ο χώρος αποθήκευσης δεδομένων ήταν ακριβός και τα σχήματα αποθήκευσης (data schemas) αρκετά απλά. Όμως, ιδιαίτερα μετά την εξάπλωση του Web 2.0, η ποσότητα των δεδομένων προς αποθήκευση αυξήθηκε σε δυσθεώρητα επίπεδα. Εκτός αυτού, η πρόσβαση στα δεδομένα γίνεται συχνότερα και ο βαθμός επεξεργασίας τους είναι πολύ μεγαλύτερος. Σ' αυτό έχουν συντελέσει καθοριστικά οι τεράστιες ποσότητες δεδομένων που παράγονται από τις κινήσεις εκατομμυρίων χρηστών ανά τον κόσμο μέσα στα κοινωνικά δίκτυα. Δεδομένα, από τα οποία μπορούν να συναχθούν πληροφορίες με τεράστια οικονομική, εμπορική και όχι μόνο χρησιμότητα.

Παράλληλα με τα παραπάνω, σήμερα τόσο οι υπολογιστικές δομές όσο και οι στρατηγικές ανάπτυξης υπολογιστικών συστημάτων έχουν αλλάξει ριζικά. Τα φθηνά σε κόστος συστήματα νεφοϋπολογιστικής (cloud computing) έχουν αντικαταστήσει τις δαπανηρές και πολύπλοκες δομές που βασίζονται σε έναν και μόνο εξυπηρετητή (server). Έτσι, οι νέες εφαρμογές που κάνουν χρήση αυτών των νέων εξελίξεων γίνονται αποδέκτες τεράστιων όγκων δεδομένων που ζητούν χώρους αποθήκευσης, ώστε να επεξεργαστούν κατάλληλα.

Οι σχεσιακές βάσεις δεδομένων δεν σχεδιάστηκαν με σκοπό να διαχειρίζονται δεδομένα σε τόσο μεγάλη ποσότητα και σε τόσο μεγάλη ταχύτητα που απαιτούν οι νέες εφαρμογές ούτε κατέχουν εκείνα τα χαρακτηριστικά που μπορούν να τις καταστήσουν συμβατές με τεράστιες αποθήκες δεδομένων και με συστήματα με τεράστια υπολογιστική δεινότητα, όπως αυτά που προσφέρει σήμερα η νεφοϋπολογιστική.

Χαρακτηριστικά μιας NoSQL βάσης δεδομένων

Οι βάσεις δεδομένων NoSQL διαθέτουν μια σειρά από χαρακτηριστικά που τις διαφοροποιούν από τις παραδοσιακές σχεσιακές βάσεις δεδομένων. Βασικό χαρακτηριστικό τους είναι ότι επιτρέπουν την εισροή δεδομένων χωρίς να χρειάζεται από πριν να υπάρχει κάποιο προκαθορισμένο μοντέλο, όπως το μοντέλο οντοτήτων – συσχετίσεων (E-R Model) των σχεσιακών βάσεων. Αυτό δίνει τη δυνατότητα να γίνονται αλλαγές στη βάση δεδομένων σε πραγματικό χρόνο, χωρίς να υπάρχει η ανησυχία για τυχόν κατάρρευση του συστήματος, με το οποίο είναι συνδεδεμένη η βάση. Επίσης, η έλλειψη προκαθορισμένου μοντέλου διευκολύνει τη δημιουργία οποιασδήποτε βάσης σε πολύ σύντομο χρόνο, ελαχιστοποιεί την ποσότητα του κώδικα που πρέπει να συνταχθεί και μειώνει το χρόνο που χρειάζεται για τη μετέπειτα διαχείριση και συντήρηση της βάσης.

Επόμενο χαρακτηριστικό των NoSQL βάσεων είναι η δυνατότητα του αυτόματου κατακερματισμού (auto-sharding). Αυτό σημαίνει πως οι βάσεις αυτές έχουν την εγγενή δυνατότητα να διανέμουν αυτόματα τα δεδομένα σ' έναν αριθμό από εξυπηρετητές, χωρίς να υπάρχει η προηγούμενη απαίτηση από την εφαρμογή να έχει ορίσει τη διάταξη αυτών των εξυπηρετητών. Τόσο τα δεδομένα όσο και τα ερωτήματα αυτόματα κατανέμονται ισορροπημένα στους εξυπηρετητές. Σε περίπτωση που ένας από τους εξυπηρετητές καταρρεύσει, υπάρχει η δυνατότητα γρήγορης αντικατάστασής του, χωρίς να διαταραχθεί η λειτουργία της εφαρμογής.
Εξίσου σημαντικό χαρακτηριστικό τους είναι η αντικατάσταση (replication). Αυτό σημαίνει πως δίνεται από την ίδια τη βάση η δυνατότητα αποκατάστασης μετά από κατάρρευση (disaster recovery), χωρίς να απαιτείται η μεσολάβηση της εφαρμογής στη διαχείριση των διαδικασιών αυτών. Ο ίδιος ο κατασκευαστής της βάσης μπορεί να δομήσει εικονικά το αποθηκευτικό περιβάλλον της βάσης σύμφωνα με τις προδιαγραφές που επιθυμεί.
Τέλος, ακόμα ένα σημαντικό πλεονέκτημα των βάσεων αυτού του είδους είναι οι ενσωματωμένες δυνατότητες προσωρινής αποθήκευσης (caching) που διαθέτουν. Μπορούν να διατηρούν στη μνήμη του συστήματος τα δεδομένα που χρησιμοποιούνται με μεγαλύτερη συχνότητα από το χρήστη. Αυτό αφαιρεί την ανάγκη διατήρησης ενός ξεχωριστού επιπρόσθετου επιπέδου προσωρινής αποθήκευσης.

Ορισμός και είδη NoSQL βάσεων δεδομένων

NoSQL ουσιαστικά σημαίνει "όχι σχεσιακή βάση" και "όχι SQL", επομένως από τον ίδιο τον ορισμό γίνεται αντιληπτό πως υπάρχουν πολλοί διαφορετικοί τρόποι, για να εφαρμοστεί η NoSQL τεχνολογία. Γενικότερα, οι NoSQL βάσεις δεδομένων περιλαμβάνουν τις παρακάτω υλοποιήσεις:

- Βάσεις δεδομένων εγγράφων (Document database): Κάθε κλειδί ταιριάζει με μια ομάδα εγγράφων. Τα έγγραφα μπορούν να διαθέτουν πολλά διαφορετικά ζευγάρια κλειδιού-τιμής ή κλειδιού-πίνακα ή ακόμα και εμφωλευμένα έγγραφα. Η βασικότερη εκπρόσωπος αυτής κατηγορίας, αλλά και μια βάση από τις πιο γνωστές στον χώρο των NoSQL βάσεων είναι η MongoDB.
- Βάσεις γράφων (Graph stores): Αυτού του είδους η υλοποίηση χρησιμοποιείται, για να αποθηκεύσει πληροφορίες σχετικές με δίκτυα, όπως παραδείγματος χάρη τα κοινωνικά δίκτυα. Τέτοιες βάσεις δεδομένων είναι η Neo4J και η HyperGraphDB.
- Βάσεις κλειδιού-τιμής (Key-Value stores): Αυτές οι βάσεις είναι οι πιο απλές του τύπου NoSQL. Κάθε διαφορετικό στοιχείο μέσα στη βάση είναι αποθηκευμένο ως χαρακτηριστικό όνομα ιδιότητας, ή κλειδί, μαζί με την τιμή του. Τέτοιες βάσεις είναι η Riak and η Voldemort. Η Redis επιτρέπει, επίσης, στην κάθε τιμή να έχει και ένα τύπο, όπως παραδείγματος χάρη "ακέραιος", πράγμα που προσθέτει μεγαλύτερη λειτουργικότητα στη βάση.
- Βάσεις ευρείας στήλης (Wide-column stores): Εδώ εκπρόσωποι αυτού του είδους είναι η Cassandra και HBase, οι οποίες έχουν βελτιστοποιηθεί, έτσι ώστε να υπάρχει η δυνατότητα να εφαρμοστούν ερωτήματα σε βάσεις με τεράστιες ποσότητες δεδομένων.

3.3 Η βάση δεδομένων MongoDB

Στα πλαίσια της παρούσας εφαρμογής επιλέχθηκε να χρησιμοποιηθεί η βάση δεδομένων MongoDB. Ακολουθεί η συνοπτική περιγραφή τα εν λόγω βάσης δεδομένων.

Χαρακτηριστικά της MongoDB

Η MongoDB αποτελεί την αιχμή της τεχνολογίας NoSQL και συγκαταλέγεται στην υποκατηγορία των βάσεων δεδομένων εγγράφων. Το όνομα της προέρχεται από το "humongous" που σημαίνει τεράστιος και αυτό ακριβός υποδηλώνει, δηλαδή την ικανότητά της να αντεπεξέρχεται σε τεράστιες ποσότητες δεδομένων και κυρίως σε μορφή εγγράφων. Ως NoSQL βάση δεν βασίζεται στο παραδοσιακό σχεσιακό μοντέλο, αλλά στην ουσία αποθηκεύει τα έγγραφα σε μορφή JSON, χρησιμοποιώντας δυναμικά σχήματα. Αυτή τη μορφή, με την οποία τα έγγραφα αποθηκεύονται στη MongoDB, οι δημιουργοί της την ονομάζουν BSON, όπου σημαίνει "Binary JSON". Πρόκειται δηλαδή για μιας δυαδικής μορφής απεικόνιση κειμενικής δομής δεδομένων και σχεσιακών πινάκων. Στη MongoDB αυτή η απεικόνιση αποτελεί ένα έγγραφο ή ένα αντικείμενο. Αυτή η τεχνική είναι από τα κυριότερα γνωρίσματα της συγκεκριμένης βάσης που την κάνουν ταχύτατη και ιδιαίτερη εύκολη στη διαχείριση κειμενικών δεδομένων.
Αναλύοντας παρακάτω τα χαρακτηριστικά της MongoDB, διαφαίνονται και τα πλεονεκτήματα που μπορεί να προσφέρει στην οποιαδήποτε εφαρμογή την χρησιμοποιεί.

Απουσία σχήματος

Από τα πιο σημαντικά στοιχεία της MongoDB είναι η απουσία σχήματος που υιοθετεί ως NoSQL βάση δεδομένων. Μια ομάδα από έγγραφα της MongoDB ονομάζεται συλλογή. Η συλλογή είναι το ακριβώς αντίστοιχο με τους πίνακες στις σχεσιακές βάσεις. Μια συλλογή ενυπάρχει σε μια απλή βάση δεδομένων και γενικότερα οι συλλογές δεν προαπαιτούν κανενός είδους σχήματος. Αυτό σημαίνει πως τα έγγραφα μέσα σε μια ορισμένη συλλογή δεν απαιτείται να έχουν ούτε τα ίδια πεδία ούτε την ίδια δομή και επίσης, τα πεδία που βρίσκονται μέσα στα έγγραφα μιας συλλογής μπορούν να διαθέτουν διαφορετικού είδους δεδομένα. Κάθε έγγραφο οφείλει μόνο να διαθέτει οικεία πεδία ως προς την οντότητα ή το αντικείμενο που το κάθε έγγραφο αντιπροσωπεύει. Στην πραγματικότητα πάντως, η πλειονότητα των εγγράφων σε μια συλλογή διαθέτει ακριβώς την ίδια δομή. Η ευελιξία σχήματος σημαίνει ότι υπάρχει η δυνατότητα μοντελοποίησης των εγγράφων στη MongoDB, έτσι ώστε να βρίσκονται σε συνάφεια με τα αντικείμενα που σχετίζονται με την εφαρμογή. Γενικότερα στη μοντελοποίηση δεδομένων, πρέπει να λαμβάνεται υπόψη οι εγγενείς ιδιότητες και απαιτήσεις των αντικειμένων της εφαρμογής και οι μεταξύ τους σχέσεις. Άρα είναι σημαντικό, πριν από τον καθορισμό οποιουδήποτε μοντέλου στη MongoDB να λαμβάνονται σοβαρά υπόψη η ποσότητα των δεδομένων που θα αποθηκευθούν, η αύξησή τους στη μονάδα του χρόνου και τα είδη των ερωτημάτων που η εφαρμογή θα διενεργεί στη βάση.

Αντικατάσταση

Άλλο χαρακτηριστικό της MongoDB και όπως προαναφέρθηκε όλων των NoSQL βάσεων δεδομένων είναι η αντικατάσταση (replication). Η αντικατάσταση στην ουσία είναι η διαδικασία συγχρονισμού των δεδομένων μέσω πολλαπλών εξυπηρετητών. Η διαδικασία αυτή προσφέρει επάρκεια και αυξάνει τη διαθεσιμότητα των δεδομένων. Έχοντας εξασφαλισμένα πολλαπλά αντίγραφα των δεδομένων σε μια σειρά από διαφορετικούς εξυπηρετητές, η ιδιότητα της αντικατάστασης προφυλάσσει τη βάση δεδομένων από την κατάρρευση του ενός και μοναδικού εξυπηρετητή, αστοχία του υλικού του υπολογιστή και διακοπή της ίδιας της υπηρεσίας, διευκολύνοντας ταυτόχρονα την ανάκτηση τυχών χαμένων δεδομένων και τη διατήρηση αντιγράφων. Σε ορισμένες περιπτώσεις η αντικατάσταση μπορεί να χρησιμοποιηθεί, ώστε να αυξηθεί η χωρητικότητα της μνήμης που αφιερώνεται στην ανάγνωση δεδομένων. Έτσι, οι πελάτες (clients) αποκτούν την ικανότητα να στέλνουν διαδικασίες εγγραφής και ανάγνωσης σε διάφορους εξυπηρετητές. Παράλληλα, η διατήρηση αντιγράφων σε διαφορετικά κέντρα

δεδομένων αυξάνει την εντοπιότητα (locality) και τη διαθεσιμότητα των δεδομένων κυρίως σε κατανεμημένες εφαρμογές.
Στη συνέχεια, θα περιγραφεί πως υλοποιείται αυτή η ιδιότητα στη MongoDB. Όλα τα δεδομένα φιλοξενούνται σ' ένα σύνολο από αντίγραφα (replica set). Από αυτά το πρωτεύον (primary) αντίγραφο δέχεται όλες τις διαδικασίες εγγραφής, ενώ τα υπόλοιπα, δηλαδή τα δευτερεύοντα (secondaries), αντιγράφουν τις διαδικασίες από το πρωτεύον, ώστε στο τέλος να διαθέτουν ακριβώς τα ίδια δεδομένα. Το πρωτεύον σύνολο δέχεται όλες τις διαδικασίες εγγραφής από τους πελάτες. Κάθε σύνολο αντιγράφων μπορεί να διαθέτει ένα μόνο πρωτεύον σύνολο. Για την υποστήριξη της αντικατάστασης όλες οι κινήσεις (logs) του πρωτεύοντος αντιγράφου αποθηκεύονται σε μια συγκεκριμένη εγγενή συλλογή της MongoDB που ονομάζεται oplog (operations logs). Σ' αυτήν την συλλογή αποθηκεύονται όλες οι μετατροπές των δεδομένων όλων των βάσεων δεδομένων και εν συνεχεία τα δευτερεύοντα σύνολα αντιγράφουν αυτές τις αλλαγές και τις εφαρμόζουν στα αποθηκευμένα δεδομένα. Επομένως, μ' αυτήν τη διαδικασία τα δεδομένα ενημερώνονται συνεχώς μετά από οποιαδήποτε αλλαγή. Πρόκειται για μια ασύγχρονή διαδικασία, που το τελικό της αποτέλεσμα είναι πλήρως ενημερωμένα αρχεία τόσο στο πρωτεύον σύνολο όσο και στα δευτερεύοντα, ώστε οι ιδιότητες της αντικατάστασης να είναι έτοιμες να τεθούν σε εφαρμογή ανά πάσα στιγμή.

Κατακερματισμός

Επόμενο σημαντικό χαρακτηριστικό είναι ο κατακερματισμός. Πρόκειται για μια διαδικασία, κατά την οποία αποθηκεύονται δεδομένα από πολλαπλούς πελάτες και ο έλεγχος της αύξησης των δεδομένων καθορίζεται από την ίδια τη MongoDB. Καθώς ο όγκος των δεδομένων αυξάνεται, ένας μεμονωμένος εξυπηρετητής ενδέχεται να μην είναι επαρκής ούτε για την αποθήκευση των δεδομένων ούτε για την εξυπηρέτηση των διαδικασιών εγγραφής και ανάγνωσης. Ο κατακερματισμός επιλύει αυτά τα προβλήματα με την οριζόντια κλιμάκωση (horizontal scaling), προσθέτοντας περισσότερους εξυπηρετητές για την υποστήριξη της αύξησης των δεδομένων και των διαδικασιών εγγραφής και ανάγνωσης.

Πιο συγκεκριμένα, ο κατακερματισμός διαιρεί τα δεδομένα και τα διανέμει σε πολλαπλούς εξυπηρετητές, γνωστούς και ως θραύσματα (shards). Το κάθε θραύσμα αποτελεί μια ανεξάρτητη βάση δεδομένων και συλλογικά, όλα αυτά τα θραύσματα δημιουργούν μια μοναδική και νοητή βάση δεδομένων. Τους σκοπούς της διεκπεραίωσης εγγραφών και αναγνώσεων και της διαχείρισης δεδομένων σε μεγάλες ποσότητες, ο κατακερματισμός τους εξυπηρετεί με το να διαμοιράζει τις απαιτούμενες διαδικασίες και τον όγκο των δεδομένων ξεχωριστά σε όλα τα θραύσματα. Με την αύξηση των δεδομένων το κάθε θραύσμα παροδικά έχει να διαχειριστεί λιγότερες διαδικασίες. Μ' αυτόν τον τρόπο, μια ομάδα από θραύσματα μοιράζεται το φόρτο των διαδικασιών οριζόντια. Για παράδειγμα, αν μια βάση δεδομένων διαθέτει δεδομένα της τάξεως του 1 TB και υπάρχουν 4 θραύσματα, τότε το κάθε θραύσμα θα πρέπει να αποθηκεύσει 250 GB, ενώ αν υπάρχουν 40 θραύσματα, τότε το καθένα θα αποθηκεύσει 25 GB.

Ευρετήρια

Τέλος, ένα από τα πολύ χρήσιμα εργαλεία που προσφέρει η MongoDB είναι η δυνατότητα χρήσης ευρετηρίων (index). Μέσω αυτού του χαρακτηριστικού αυξάνεται η απόδοση στις διαδικασίες ανάγνωσης, όταν πρόκειται για συχνά επαναλαμβανόμενα ερωτήματα προς τη βάση. Μάλιστα τα ευρετήρια είναι ιδιαίτερα χρήσιμα, όταν το συνολικό μέγεθος των δεδομένων ξεπερνά την ποσότητα της διαθέσιμης μνήμης τυχαίας προσπέλασης (RAM).
Ένα ευρετήριο είναι μια δομή δεδομένων, όπου διευκολύνει το γρήγορο εντοπισμό εγγράφων βάσει των αποθηκευμένων τιμών σε συγκεκριμένα πεδία. Ουσιαστικά, τα ευρετήρια της MongoDB δεν διαφέρουν σε κάτι από τα ευρετήρια στις υπόλοιπες βάσεις δεδομένων. Η

MongoDB υποστηρίζει ευρετήρια σε κάθε πεδίο και υποπεδίο που εμπεριέχεται μέσα σ' έγγραφα συλλογών της MongoDB.
Συνοπτικά, τα ευρετήρια της MongoDB έχουν τα παρακάτω χαρακτηριστικά.

- Τα ευρετήρια μπορούν να οριστούν ακόμα και ένα επίπεδο πριν τις συλλογές.
- Υπάρχει η δυνατότητα ορισμού ευρετηρίων σ' ένα μόνο πεδίο ή σε πολλαπλά πεδία με τη χρήση σύνθετων ευρετηρίων.
- Τα ευρετήρια ενισχύουν τις επιδόσεις των ερωτημάτων σε μεγάλο βαθμό.
- Τέλος, τα ευρετήρια της MongoDB είναι τύπου B-δέντρων. Δηλαδή μιας δεντρικής δομής δεδομένων, όπου όλα τα δεδομένα είναι διατεταγμένα και επιτρέπεται να διενεργηθούν ερωτήματα, προσθήκες και διαγραφές σε λογαριθμικό χρόνο. Τα B-δέντρα αποτελούν μια γενίκευση των δυαδικών δέντρων αναζητήσεων, όπου ο ένας κόμβος μπορεί να έχει παραπάνω από δύο παιδιά. Τα B-δέντρα είναι κατάλληλα για αναζητήσεις, αναγνώσεις και εγγραφές σε μεγάλα σώματα δεδομένων, όπως ακριβώς αυτά που μπορεί να φιλοξενήσει η MongoDB.

3.3 Το ElasticSearch

Το ElasticSearch χρησιμοποιείται ως εργαλείο για αναζήτηση λέξεων μέσα από κειμενικά δεδομένα. Το εύρος των δυνατοτήτων του είναι αρκετά μεγάλο και μερικές από αυτές τις δυνατότητες θα αναλυθούν στη συνέχεια, αλλά κατά κόρον είναι φτιαγμένο, για να αποτελεί την αναγκαία υποδομή, πάνω στην οποία δημιουργούνται μηχανές αναζήτησης, αλλά και για να παρέχει στατιστικές αναλύσεις πάνω σε σώματα κειμένων. Στην παρούσα εφαρμογή λειτουργεί, παράλληλα με τη MongoDB ως ένας εναλλακτικός τρόπος αποθήκευσης και αναζήτησης κειμενικών δεδομένων.

Χαρακτηριστικά και δομή του ElasticSearch

Πιο συγκεκριμένα, το ElasticSearch είναι ένας ανεξάρτητος εξυπηρετητής βάσης δεδομένων (standalone database server), γραμμένο σε Java, που δέχεται δεδομένα και τα αποθηκεύει με τέτοιο τρόπο, ώστε να διευκολύνει αναζητήσεις πάνω σ' αυτά. Είναι ιδιαίτερα εύχρηστο προγραμματιστικά, καθώς το κυρίως πρωτόκολλο που χρησιμοποιεί βασίζεται σε HTTP/JSON. Ο ίδιος ο μηχανισμός του HTTP είναι εξ ολοκλήρου ασύγχρονος από μόνος του, επομένως κανένα νήμα πληροφορίας που αποστέλλεται δεν χρειάζεται να περιμένει απόκριση. Το ElasticSearch προσφέρει μεγάλες δυνατότητες επεκτασιμότητας, αφήνοντας περιθώρια ποικίλων ομαδοποιήσεων, πράγμα το οποίο αντικατοπτρίζει και το ίδιο το όνομά του. Επίσης, η χρήση του μπορεί να εφαρμοστεί σε διάφορα πεδία αναζητήσεων, όπως παραδείγματος χάρη, στα προϊόντα που διαθέτει μια βάση δεδομένων ή στα άρθρα που διαθέτει ένα ιστολόγιο. Το ElasticSearch διαθέτει τις δυνατότητες να ξεπερνά τα εμπόδια που δημιουργεί στις αναζητήσεις η φυσική γλώσσα.

Apache Lucene

Ο πυρήνας του ElasticSearch στην ουσία είναι ένα άλλο λογισμικό, ιδιαίτερα γνωστό και με ισχυρές δυνατότητες, το Apache Lucene. Το ElasticSearch μπορεί να μελετηθεί περισσότερο σε βάθος, αν μελετηθεί παράλληλα με το Lucene και τις δικές του υποδομές. Οι αλγόριθμοι του ElasticSearch που σχετίζονται είτε με ταίριασμα κειμένου (text matching) είτε με βελτιστοποιημένα ευρετήρια όρων προς αναζήτηση προέρχονται από τις υποδομές του Lucene. Το νέο που κομίζει το ElasticSearch στην ήδη υπάρχουσα τεχνολογία του Lucene είναι το πιο εύχρηστο και ακριβές API, η επεκτασιμότητα, η διαλειτουργικότητα με γλώσσες

προγραμματισμού πέραν της Java, η ευχρηστία προγραμματιστικής διαχείρισής του, η ομαδοποίηση (clustering) και η αντικατάσταση (replication), γνωρίσματα που ήδη αναφέρθηκαν στις NoSQL τεχνολογίες και καινούργιες, πιο εύχρηστες εφαρμογές, οι οποίες αποτελούν επεκτάσεις των Java κλάσεων του Lucene.

Ευρετήριο

Βασικό συστατικό του ElasticSearch είναι το ευρετήριο (index), όπου και αποθηκεύονται τα δεδομένα. Όπως είχε προαναφερθεί και στην περιγραφή των NoSQL βάσεων δεδομένων, τα ευρετήρια για το ElasticSearch έχουν ακριβώς την ίδια λειτουργία με τους πίνακες των σχεσιακών βάσεων. Αλλά αντίθετα από αυτές, οι τιμές που είναι αποθηκευμένες σ' ένα ευρετήριο προορίζονται, ώστε να συνδράμουν στην επιτάχυνση μιας πιθανής αναζήτησης σε κειμενικά δεδομένα. Για την ακρίβεια το ανάλογο ενός ευρετηρίου στο ElasticSearch είναι μια συλλογή της MongoDB.

Δομή

Η κυρίως αποθηκευτική οντότητα του ElasticSearch είναι το έγγραφο. Σε αναλογία προς τις σχεσιακές βάσεις, ένα έγγραφο είναι μια γραμμή από δεδομένα σ' έναν πίνακα. Συγκρίνοντας ένα έγγραφο του ElasticSearch με ένα έγγραφο της MongoDB, και τα δύο μπορούν να έχουν διαφορετικές δομές, αλλά αυτό του ElasticSearch πρέπει να διαθέτει τους ίδιους τύπους για κοινά πεδία. Τα έγγραφα αποτελούνται από πεδία (γραμμές), αλλά κάθε πεδίο μπορεί να εμφανίζεται περισσότερες από μια φορές, τότε σ' αυτήν την περίπτωση ονομάζεται πεδίο πολλαπλών τιμών (multivalued). Κάθε πεδίο διαθέτει ένα τύπο (κείμενο, νούμερο, ημερομηνία κοκ), όπως επίσης μπορεί να εμπεριέχει ένα μέρος από κάποιο έγγραφο (subdocument) ή και πίνακες. Οι τύποι των πεδίων μπορούν επίσης να είναι σύνθετοι. Ο τύπος των πεδίων έχει ιδιαίτερη σημασία στο ElasticSearch, γιατί παρέχει στην εκάστοτε μηχανή αναζήτησης πληροφορίες για το πόσο συχνές διαδικασίες, παραδείγματος χάρη σύγκρισης ή ταξινόμησης, πρέπει να λάβουν χώρα. Στην περίπτωση του ElasticSearch βέβαια ο καθορισμός αυτών των διαδικασιών επιτυγχάνεται με αυτόματο τρόπο. Από την άλλη, στις σχεσιακές βάσεις τα έγγραφα δεν απαιτείται να έχουν προκαθορισμένη δομή. Το κάθε έγγραφο μπορεί να διαθέτει διαφορετικό σύνολο πεδίων και επίσης τα πεδία δεν είναι απαραίτητο να είναι γνωστά πριν από την ανάπτυξη της εφαρμογής, παρά μόνο αν από πριν καθοριστεί κάποιου είδους σχήματος.

Εξίσου σημαντικό στο ElasticSearch είναι και ο τύπος του εγγράφου, όπου σ' ένα ευρετήριο μπορούν να αποθηκευτούν πολλές οντότητες με διαφορετικούς σκοπούς. Για παράδειγμα, σ' ένα ιστολόγιο που διαθέτει ταυτόχρονα άρθρα και σχόλια, ο τύπος του εγγράφου διαφοροποιεί εύκολα αυτά τα δύο διαφορετικά είδη κειμενικών δεδομένων. Είναι επίσης πολύ σημαντικό να επισημανθεί πως το κάθε έγγραφο μπορεί να έχει τη δική του διαφορετική δομή και πως αυτός ο διαχωρισμός βοηθάει σημαντικά στη διαχείριση των δεδομένων. Όμως δεν πρέπει να αγνοούνται και οι περιορισμοί που υφίστανται, καθώς οι διαφορετικοί τύποι ενός εγγράφου δεν μπορούν να θέσουν διαφορετικούς τύπους για την ίδια οντότητα.

Ανεξάρτητός εξυπηρετήτης

Επίσης, το ElasticSearch έχει την ιδιότητα να λειτουργεί και ως ανεξάρτητος εξυπηρετητής, αλλά και να λειτουργεί μέσω πολλαπλών συνεργαζόμενων εξυπηρετητών, ώστε να μπορεί να επεξεργάζεται ογκώδη σύνολα δεδομένων και να διαθέτει ανοχή σε τυχών σφάλματα. Το σύνολο αυτών των συνεργαζόμενων εξυπηρετητών ονομάζεται σύμπλεγμα (cluster) και ο καθένας από αυτούς ξεχωριστά κόμβος (node).

Κατακερματισμός

Σημαντικό χαρακτηριστικό του ElasticSearch αποτελεί και ο κατακερματισμός, ο οποίος είναι χρήσιμος στην περίπτωση, όπου ένας κόμβος υπερφορτωθεί από δεδομένα είτε εξαιτίας έλλειψης μνήμης τυχαίας προσπέλασης (RAM) είτε έλλειψης χώρου στο σκληρό δίσκο είτε οποιαδήποτε άλλης αδυναμίας λογισμικού. Σ' αυτές τις περιπτώσεις, υπάρχει η ευελιξία τα δεδομένα να κατακερματιστούν σε μικρότερα κομμάτια, τα λεγόμενα θραύσματα (shards), όπου το καθένα από αυτά στην ουσία αποτελεί ένα ανεξάρτητο ευρετήριο με τις ιδιότητες που κληρονομεί από το Lucene. Κάθε θραύσμα μπορεί να τοποθετηθεί σε διαφορετικό εξυπηρετητή και μ' αυτόν τον τρόπο τα δεδομένα διαχέονται σε ολόκληρο το σύμπλεγμα των εξυπηρετητών. Όταν τίθενται ερωτήματα σε ένα ευρετήριο, τα αποτελέσματα που επιστρέφονται ουσιαστικά συντίθενται από πολλαπλά θραύσματα. Το ElasticSearch αποστέλλει το εκάστοτε ερώτημα στα σχετικά θραύσματα και συνενώνει τα αποτελέσματα με τέτοιον τρόπο που δεν επιβαρύνει τη λειτουργία της εφαρμογής περαιτέρω, καθώς η εύρεση των θραυσμάτων είναι λειτουργία που διενεργείται αποκλειστικά από το ίδιο.

Αντικατάσταση

Τέλος, σημαντικό ρόλο στο ElasticSearch παίζουν και τα αντίγραφα που μπορεί να έχει ένα θραύσμα (shard replicas), τα όποια προσδίδουν στην εφαρμογή μεγάλο βαθμό διαθεσιμότητας όσον αφορά τα δεδομένα. Το πρωτεύον θραύσμα χρησιμοποιείται ως το μέρος, όπου κατευθύνονται οι λειτουργίες εκείνες που προκαλούν αλλαγές στο ευρετήριο. Ένα αντίγραφο (replica) είναι μια πιστή αναπαραγωγή του πρωτεύοντος θραύσματος και κάθε θραύσμα μπορεί να έχει από κανένα ως έναν αριθμό από αντίγραφα. Στην περίπτωση εκείνη, όπου το πρωτεύον θραύσμα δεν ανταποκρίνεται λόγω μη διαθεσιμότητας που οφείλεται σε αδυναμία του υλικού, τότε ένα σύμπλεγμα από εξυπηρετητές μπορεί να προαγάγει ένα αντίγραφο ως το νέο πρωτεύον θραύσμα.

Διαδικασία ευρετηρίασης

Στη συνέχεια, μετά την περιγραφή της δομής και των χαρακτηριστικών του ElasticSeach, θα ήταν χρήσιμο να γίνει αναφορά για τη διαδικασία που ακολουθείται από την στιγμή που ένα έγγραφο αποθηκεύεται εντός του ElasticSearch μέχρι τη στιγμή που ανακτάται μέσω κάποιου ερωτήματος.

Η πρόσθεση δεδομένων μέσα σ' ένα ευρετήριο μπορεί να γίνει είτε απευθείας, απλώς με την αποθήκευση μερικών εγγράφων χωρίς προηγούμενο καθορισμό κάποιου ειδικού ευρετηρίου είτε μπορεί ο χρήστης να επιλέξει τον ευρετηριασμό των δεδομένων με συγκεκριμένο τρόπο. Το ElasticSearch διαθέτει αρκετές επιλογές για τον τρόπο, με τον οποίο μπορεί να γίνει ευρετηρίαση δεδομένων. Οι δύο βασικοί μέθοδοι που ακολουθούνται για τον εξειδικευμένο καθορισμό ενός ευρετηρίου είναι η ανάλυση (analysis) και η χαρτογράφηση (mapping), οι οποίες και θα αναλυθούν παρακάτω.

Ανάλυση

Στο ElasticSearch, με τη δημιουργία ενός ευρετηρίου ταυτόχρονα καθορίζεται και ο αριθμός των θραυσμάτων και των αντιγράφων που θα χρησιμοποιηθούν. Όπως προαναφέρθηκε, ένα θραύσμα είναι ένα κομμάτι από τα δεδομένα και το αντίγραφο είναι μια επανάληψη των ίδιων δεδομένων με τη μορφή ενός αντιγράφου ασφαλείας (backup). Όταν λοιπόν υπάρχει ένας κόμβος, τότε όλα τα θραύσματα και όλα τα αντίγραφα βρίσκονται μέσα σε αυτόν τον κόμβο. Ενώ στην περίπτωση που οι κόμβοι είναι περισσότεροι, τότε τα δεδομένα κατανέμονται ανάμεσα σ' αυτούς τους κόμβους.

Τα δεδομένα στο ElasticSearch αναλύονται σε δύο φάσεις. Η πρώτη φορά είναι όταν ένα έγγραφο αποθηκεύεται και η πληροφορία της ανάλυσής του καταγράφεται και αυτή στο ευρετήριο και η δεύτερη φορά είναι όταν διενεργείται κάποιο ερώτημα. Το ElasticSearch αναλύει το ερώτημα της αναζήτησης και ψάχνει στις αποθηκευμένες πληροφορίες του ευρετηρίου.

Η διαδικασία της ανάλυσης ευρετηρίου (index analysis) λειτουργεί ως ένα ρυθμιζόμενο μητρώο αναλυτών (analyzers), οι οποίοι μπορούν τόσο να αναλύσουν σε πεδία ένα έγγραφο κατά την είσοδό του όσο και ένα ερώτημα με τη μορφή συμβολοακολουθίας (string). Οι αναλυτές γενικότερα αποτελούνται από έναν και μόνο διαχωριστή λεκτικών μονάδων (tokenizer) και από κανένα μέχρι περισσότερα φίλτρα λεκτικών μονάδων. Ως διαχωριστής λεκτικών μονάδων ορίζεται το εργαλείο εκείνο που λαμβάνει ως είσοδο μια συμβολοακολουθία (string) και επιστρέφει το σύνολο των λεκτικών μονάδων που εμπεριέχονται στην ακολουθία αυτή. Παραδείγματος χάρη, ο προκαθορισμένος αναλυτής (standard analyzer) αποτελείται από τον προκαθορισμένο διαχωριστή λεκτικών μονάδων, ο οποίος εφαρμόζει το προκαθορισμένο φίλτρο λεκτικών μονάδων, όπου κανονικοποιεί όλες τις λεκτικές μονάδες, το φίλτρο μικρογράμματων λέξεων, όπου μετατρέπει όλες τις λεκτικές μονάδες με μικρά γράμματα και το φίλτρο των stop-words λέξεων, όπου απομακρύνει όλες τις λέξεις που γραμματικά είναι άρθρα, αντωνυμίες, σύνδεσμοι κτλ, δηλαδή μικρές σε μέγεθος λέξεις με περισσότερο λειτουργικό χαρακτήρα και λιγότερο σημασιολογικό.

Χαρτογράφηση

Στη συνέχεια ακολουθεί η χαρτογράφηση, κατά την οποία καθορίζεται το είδος των δεδομένων που θα τοποθετηθεί σε κάθε πεδίο. Αν δεν υπάρξει καθορισμός χαρτογράφησης, τότε το ElasticSearch θα προσπαθήσει να μαντέψει το είδος το δεδομένων και θα τα χαρτογραφήσει αυτόματα, χωρίς αυτό να συνιστά αλάνθαστη διαδικασία. Στη χαρτογράφηση, πέρα από το είδος των δεδομένων σε κάθε πεδίο, δίνεται η δυνατότητα να προσδοθεί βαρύτητα (boost) σε ορισμένα πεδία, ώστε σ' αυτά να δίνεται προτεραιότητα κατά τη διάρκεια των αναζητήσεων.

Αναζήτηση

Το επόμενο βήμα που ακολουθεί, εφόσον έχουν ολοκληρωθεί η παραπάνω διαδικασίες και έχουν προστεθεί κάποια δεδομένα στο ευρετήριο, είναι η αναζήτηση. Μια βασική αναζήτηση εκτελεί ένα ερώτημα διαμέσου του ElasticSearch. Το επιστρεφόμενο αποτέλεσμα του ερωτήματος μπορεί και αυτό να φιλτραριστεί είτε με τον καθορισμό συγκεκριμένων φίλτρων, όπως παραδείγματος χάρη λέξεων κλειδιών ή εύρος ημερομηνιών είτε με τον καθορισμό συγκεκριμένων όψεων (facets), ώστε να επιστρέφεται μόνο ένα συγκεκριμένο κομμάτι των δεδομένων, όπως για παράδειγμα, να επιστρέφονται αποτελέσματα μόνο από ένα συγκεκριμένο πεδίο, μέχρι ένα ορισμένο μέγεθος και σε φθίνουσα σειρά.

Πλεονεκτήματα ElasticSearch

Συνοψίζοντας για το ElasticSearch, θα αναφερθούν μερικές από τις περιπτώσεις, όπου η χρήση του στην κατασκευή μηχανών αναζήτησης είναι ιδιαίτερη χρήσιμη. Τα παρακάτω παραδείγματα είναι χαρακτηριστικές περιπτώσεις των προβλημάτων που μπορεί να επιλύσει.

- Έχει την ικανότητα να επιστρέφει τα καλύτερα αποτελέσματα μέσα από έναν μεγάλο αριθμό περιγραφών προϊόντων, ιδιαίτερα όταν αναζητούνται φράσεις, όπως "chef's knife" για παράδειγμα. Αυτό καθίσταται δυνατό με την χρήση των διαχωριστών λεκτικών μονάδων, που περιγράφηκαν παραπάνω.
- Δοθέντος του προηγούμενου παραδείγματος, υπάρχει η δυνατότητα να αναζητηθούν σε ποια συγκεκριμένα υποκαταστήματα υπάρχει το συγκεκριμένο προϊόν και από ποιες ποσότητες και πάνω. Σ' αυτό μπορεί να βοηθήσει ο καθορισμός συγκεκριμένων όψεων στα επιστρεφόμενα αποτελέσματα.
- Υπάρχει και η δυνατότητα αυτοσυμπλήρωσης (auto-completing) στο κουτί αναζήτησης σε μερικώς γραμμένες λέξεις βασιζόμενη σε προηγούμενες αναζητήσεις.

Σε γενικές γραμμές, το ElasticSearch αποτελεί την τέλεια λύση στο να επιστρέφει αποτελέσματα κατά προσέγγιση από δεδομένα, καθώς βαθμολογεί τα αποτελέσματα βάσει της ποιότητας. Ενώ

το ElasticSearch μπορεί να φέρει εις πέρας ακριβές ταίριασμα σε κειμενικές αναζητήσεις και στατιστικούς υπολογισμούς, στην πραγματικότητα ο τρόπος που το πετυχαίνει είναι μια εγγενώς προσεγγιστική διαδικασία. Αυτή του η ιδιότητα το ξεχωρίζει και από τις περισσότερες παραδοσιακές βάσεις δεδομένων.

3.4 Το πρότυπο Μοντέλο – Άποψη – Ελεγκτής (Model – View – Controller)

Με τις προηγούμενες αναφορές στη MongoDB και στο ElasticSearch ολοκληρώθηκε η περιγραφή των υποδομών, όπου αποθηκεύονται τα κειμενικά δεδομένα και της επεξεργασίας, στην οποία υποβάλλονται, προκειμένου να διευκολύνεται τόσο η αποθήκευση όσο και η αναζήτηση μέσα σ' αυτά. Στην ουσία αυτές οι τεχνολογίες αποτελούν και τη βάση της εφαρμογής – το back-end δηλαδή – και στη συνέχεια θα περιγραφούν οι τεχνολογίες που χρησιμοποιήθηκαν για την κατασκευή της ιστοσελίδας που επιτρέπει στο χρήστη να αλληλεπιδράσει με την εφαρμογή – δηλαδή το front-end. Η βασική αρχιτεκτονική της διεπαφής στηρίζεται πάνω στο πρότυπο Μοντέλο – Άποψη – Ελεγκτής (MVC) και η εφαρμογή διαδικτυακού πλαισίου λογισμικού (Web Application Framework) που χρησιμοποιήθηκε είναι το CodeIgniter.

Ορισμός

Το πρότυπο Μοντέλο – Άποψη – Ελεγκτής (MVC) αποτελεί μια αρχιτεκτονική ανάπτυξης λογισμικού, κατά την οποία η παρουσίαση της πληροφορίας διαχωρίζεται από την αλληλεπίδραση του χρήστη μ' αυτήν. Αν και αρχικά αυτή η αρχιτεκτονική κατασκευάστηκε για ανάπτυξη λογισμικού που αφορούσε τους προσωπικούς υπολογιστές, σήμερα έχει γίνει γνωστή σε ευρεία κλίμακα ως αρχιτεκτονική που υιοθετείται στην ανάπτυξη εφαρμογών στον παγκόσμιο ιστό (World Wide Web) και είναι συμβατή με όλες τις γνωστές γλώσσες προγραμματισμού. Υπάρχουν πολλές εφαρμογές διαδικτυακού πλαισίου λογισμικού, εμπορικές και μη που βασίζονται πάνω σ' αυτήν την αρχιτεκτονική, όπως το CodeIgniter που προαναφέρθηκε, το Django, το Ruby on Rails κά.

Πλεονεκτήματα MVC προτύπου

Στη συνέχεια, πριν περιγραφεί η δομή και τα χαρακτηριστικά του MVC προτύπου, είναι χρήσιμο να αποσαφηνιστούν τα προτερήματα της χρήσης του σε μια διαδικτυακή εφαρμογή. Οποιαδήποτε εφαρμογή, λοιπόν, έχει δομηθεί πάνω στο MVC πρότυπο, έχει κατανεμημένες τις βασικές της λειτουργικότητες σε τρεις τύπους αρχείων, δηλαδή τα μοντέλα, τις απόψεις και τους ελεγκτές. Αυτή η τεχνική επιτρέπει στο κάθε κομμάτι της να σχεδιάζεται, να διαχειρίζεται και να ελέγχεται ανεξάρτητα από τα υπόλοιπα. Επίσης, ενισχύεται μ' αυτόν τον τρόπο η δυνατότητα να παραμένει ο κώδικας τακτοποιημένος, πράγμα το οποίο διευκολύνει την επεκτασιμότητα και την επαναχρησιμοποίηση του κώδικα, την εύρεση λαθών και διόρθωσή τους, αλλά και αυξάνει τις επιδόσεις της εφαρμογής σε ταχύτητα απόκρισης. Τέλος, ο διαχωρισμός αυτός σε πολλαπλά αρχεία ευνοεί τη συνεργασία πολλών ανθρώπων πάνω στην ίδια εφαρμογή, καθώς μπορεί ο καθένας να κατέχει διακριτούς ρόλους στην ανάπτυξη και τη συντήρησή της.

Ελεγκτής

Εφόσον έγινε κατανοητή η σημασία χρήσης αυτής της αρχιτεκτονικής, ακολουθεί η ανάλυση των επιμέρους συστατικών της. Αρχικά, ο ελεγκτής μπορεί να χαρακτηριστεί ως ένας διαμεσολαβητής μεταξύ του μοντέλου και της άποψης. Οι βασικές εργασίες που εκτελεί ο ελεγκτής είναι:

- Λαμβάνει δεδομένα από τον χρήστη μέσω των αρχείων που αποτελούν το κομμάτι της άποψης.
- Διατηρεί τα δεδομένα, όπως κάθε άλλος τύπος διαδικτυακής εφαρμογής. Έχοντας υπ' όψιν τη λειτουργικότητα της PHP, τότε μπορεί να ειπωθεί πως τα δεδομένα διατηρούνται σε $_GET και $_POST μεταβλητές.
- Ο ελεγκτής είναι υπεύθυνος να καθορίζει ποιες λειτουργίες χρειάζεται να λάβουν χώρα και γι' αυτό χρησιμοποιεί το κατάλληλο μοντέλο, ώστε να τις θέσει σε εφαρμογή.
- Εφόσον δημιουργήσει ένα αντικείμενο του μοντέλου, τότε θέτει σε εφαρμογή τη λειτουργία και διατηρεί ό, τι επιστραφεί από αυτήν.
- Τέλος, ο ελεγκτής φορτώνει την κατάλληλη άποψη, ανάλογα με τους σκοπούς του χρήστη και αποστέλλει τις πληροφορίες που δέχθηκε από το μοντέλο.

Μοντέλο

Το μοντέλο αναπαριστά τη δομή των δεδομένων που διαχειρίζεται η εφαρμογή. Είναι υπεύθυνο να διατηρεί συναρτήσεις και μεταβλητές που εμπλέκονται με τα δεδομένα που αυτό αναπαριστά. Είναι πολύ βοηθητικό στην κατανόηση της λογικής του μοντέλου, αν θεωρηθεί ως ο πυρήνας ενός αντικειμένου στον αντικειμενοστραφή προγραμματισμό. Επομένως, το μοντέλο διεκπεραιώνει τις εξής λειτουργίες:

- Λαμβάνει οδηγίες διαμέσου του ελεγκτή και επεξεργάζεται δεδομένα που συνήθως λαμβάνει από κάποια βάση δεδομένων.
- Όταν οι αναγκαίες πληροφορίες επιστραφούν από τις συναρτήσεις ή τα δεδομένα τοποθετηθούν μέσα στις κατάλληλες μεταβλητές, τότε ο ελεγκτής τις χρησιμοποιεί ως αντικείμενα.

Άρα συνοπτικά το μοντέλο προσθέτει, ενημερώνει και ανακτά δεδομένα από τη βάση δεδομένων, ευρισκόμενο σε μια συνεχή επικοινωνία με τον ελεγκτή, κατά την οποία ανταλλάσσονται δεδομένα ανάλογα με τις απαιτήσεις του χρήστη.

Άποψη

Τέλος, τις πληροφορίες που έχει αιτηθεί ο ελεγκτής από το μοντέλο, τις αποστέλλει στην άποψη, ώστε να εμφανιστούν προς το χρήστη με συγκεκριμένο τρόπο. Στην ουσία η άποψη αποτελεί ένα σύστημα εμφάνισης της πληροφορίας που δομείται με συγκεκριμένο τρόπο, έτσι ώστε να μπορεί να εμφανιστεί είτε σε μια ιστοσελίδα είτε σε μια εφαρμογή κινητού τηλεφώνου. Η άποψη μπορεί να αποτελείται από μια ή περισσότερες ιστοσελίδες που να είναι γραμμένες με τη συνεργασία της γλώσσας χαρακτηρισμού υπερκειμένου (Hyper Text Markup Language - HTML), της γλώσσας κατασκευής ιστοσελίδων δυναμικού περιεχομένου (PHP) και της γλώσσας φύλλων στυλ (Cascading Style Sheets - CSS). Σ' αυτές τις ιστοσελίδες, επίσης, μπορούν να συνδράμουν και όλες οι υπόλοιπες γλώσσες που συντελούν στην κατασκευή ιστοσελίδων, όπως παραδείγματος χάρη η JavaScript.

Πρέπει να σημειωθεί πως μια διαδικτυακή εφαρμογή αποτελείται από σύνολα ελεγκτών, μοντέλων και απόψεων, δηλαδή από διάφορα αρχεία που επιτελούν τις λειτουργίες που αναλύθηκαν. Ένας όμως ελεγκτής παίζει το ρόλο του βασικού ελεγκτή, ο οποίος δέχεται όλες τις αιτήσεις και καλεί τους συγκεκριμένους ελεγκτές που χειρίζονται συγκεκριμένες ενέργειες σε κάθε περίπτωση.

3.5 Τα Πλαίσια Λογισμικού PHP και το CodeIgniter

Η παρούσα εφαρμογή, προκειμένου να εκμεταλλευτεί στο έπακρο τα οφέλη του προτύπου Μοντέλο – Άποψη – Ελεγκτής (MVC), έχει αναπτυχθεί βασισμένη πάνω σε ένα πλαίσιο λογισμικού PHP (PHP Framework) που στηρίζεται σ' αυτήν την αρχιτεκτονική και ονομάζεται CodeIgniter. Στη συνέχεια, θα παρουσιαστούν μερικά από τα προτερήματα της ανάπτυξης διαδικτυακών εφαρμογών με τη βοήθεια πλαισίων λογισμικού PHP και ειδικά του CodeIgniter. Μερικά από τα πιο γνωστά, εκτός από το CodeIgniter είναι τα Zend Framework, CakePHP, Symphony κά.

Πλεονεκτήματα πλαισίων λογισμικού PHP

Υπάρχουν πολλά κριτήρια, τα οποία μπορούν να χαρακτηρίσουν ένα πλαίσιο λογισμικού PHP, τα ουσιαστικότερα όμως από αυτά είναι:

- Προσφέρουν υψηλού επιπέδου ασφάλεια. Τα διαδικτυακά αυτά πλαίσια παρέχουν στην πλειοψηφία τους συνήθως λειτουργίες αναγνώρισης, πιστοποίησης και τελικώς εξουσιοδότησης στους χρήστες της εφαρμογής, περιορίζοντας την πρόσβαση σε λειτουργίες με βάση ορισμένα θεμιτά κριτήρια.
- Συνδέονται με βάσεις δεδομένων υπό συγκεκριμένες συνθήκες. Τα διαδικτυακά πλαίσια δημιουργούν έναν ενοποιημένο τρόπο διαχείρισης βάσεων δεδομένων, ο οποίος αποσκοπεί στη σύνδεση και τη χρήση των βάσεων δεδομένων σε ένα υψηλότερο επίπεδο εννοιών.
- Έχει προβλεφθεί, ώστε να παρέχουν χαρτογράφηση των Ενιαίων Εντοπιστών Πόρων (Uniform Resource Locators - URLs), δηλαδή ο σχηματισμός τους διαθέτει σαφή λογική, αλλά ταυτόχρονα είναι και συμβατός με τον τρόπο λειτουργίας των σύγχρονων μηχανών αναζήτησης.
- Ορισμένα πλαίσια επιτρέπουν μέσω καταλλήλων αυτοματοποιημένων διεπαφών την εύκολη και γρήγορη εγκατάσταση και παραμετροποίησή τους στις εκάστοτε ανάγκες μίας διαδικτυακής εφαρμογής.
- Τα πλαίσια αυτά διαθέτουν μια πλήρη και πλούσια τεκμηρίωση από έγγραφα, τα οποία μπορούν να σταθούν αρωγός ακόμα και σε κάποιον που δεν είναι επαγγελματίας στο χώρο του διαδικτυακού προγραμματισμού. Επίσης, συνήθως γύρω από αυτά δραστηριοποιούνται κοινότητες προγραμματιστών που αποτελούν σημαντική βοήθεια.
- Τέλος, διαθέτουν εγγενείς συλλογές βιβλιοθηκών, οι οποίες συντελούν στο να υπάρχει ήδη ένα υπόβαθρο για την ανάπτυξη οποιασδήποτε λειτουργίας της εφαρμογής. Αν, παραδείγματος χάρη, πρέπει να δημιουργηθεί μια φόρμα που θα λαμβάνει δεδομένα και χρειάζεται να γίνεται έλεγχος στα εισερχόμενα δεδομένα, τα πλαίσια λογισμικού PHP διαθέτουν ήδη έτοιμες κλάσεις γι' αυτόν τον σκοπό. Οπότε καμία σχεδόν ενέργεια δεν ξεκινάει από το μηδέν.

Γενικότερα, τα πλαίσια λογισμικού PHP συμβάλλουν στην εξάλειψη επαναλαμβανόμενου κώδικα και στη συστηματοποίηση της διαδικασίας ολοκλήρωσης μιας διαδικτυακής εφαρμογής. Αποτελούν ισχυρά εργαλεία που είναι σε θέση να συνδράμουν τη συγγραφή κώδικα με τακτοποιημένο και εύκολο τρόπο, παρέχοντας μια σειρά από βοηθήματα.

CodeIgniter και MVC πρότυπο

Αφού έγινε αναφορά στα γενικά χαρακτηριστικά και τα προτερήματα των πλαισίων λογισμικού PHP, στο σημείο αυτό θα παρουσιαστούν πιο εξειδικευμένα πάνω στο CodeIgniter που χρησιμοποιήθηκε για τις ανάγκες της κατασκευής της παρούσας εφαρμογής. Αρχικά, λοιπόν, το CodeIgniter αποτελεί ένα από τα πιο γνωστά και δημοφιλή πλαίσια λογισμικού PHP. Ένα από τα χαρακτηριστικά που το διακρίνει είναι ότι υιοθετεί το πρότυπο αρχιτεκτονικής Μοντέλου – Άποψης – Ελεγκτή που περιγράφηκε σε προηγούμενο κεφάλαιο. Ο διαχωρισμός της ανάπτυξης

μιας εφαρμογής στα πλαίσια αυτού του προτύπου σημαίνει πως το κάθε τμήμα ολοκληρώνει τη δουλειά του με εξαιρετική επάρκεια, διατηρώντας τις σχέσεις του με τα υπόλοιπα σε συνεχή συνεργασία, αλλά και με σαφέστατα όρια για το που τελειώνουν και που αρχίζουν οι αρμοδιότητες του καθενός. Οι αλλαγές σε καθένα από τα τρία διαφορετικά μέρη που απαρτίζουν αυτό το πρότυπο δεν επηρεάζουν τα υπόλοιπα και τη γενικότερη λειτουργία της εφαρμογής, καθώς και στα πλαίσια αυτού του προτύπου παρέχεται το έδαφος για την εφαρμογή όλων των κανόνων του αντικειμενοστραφούς προγραμματισμού.

Επιπλέον διευκολύνσεις

Πέρα από το πρότυπο αρχιτεκτονικής, το CodeIgniter είναι ιδιαίτερα εύκολο στην εγκατάστασή του και επιπλέον παρέχει μια μεγάλη συλλογή από βιβλιοθήκες, βοηθούς (helpers) και πρόσθετα (plug-ins) που διευκολύνουν σε μεγάλο βαθμό την ανάπτυξη της εφαρμογής, γιατί δεν πρέπει σε κάθε βήμα της κατασκευής να ανακαλύπτεται από την αρχή ο τροχός. Παραδείγματος χάρη, υπάρχει ειδική βιβλιοθήκη που ολοκληρώνει τη σύνδεση με μια MySQL βάση δεδομένων, όπως επίσης διατίθεται και ξεχωριστή κλάση, πάνω στην οποία μπορεί να δημιουργηθεί ο μηχανισμός για την επικύρωση δεδομένων που δίνονται μέσα από μια φόρμα.

Επιπλέον, ο τρόπος δομής των φακέλων που περιλαμβάνουν την κάθε λειτουργική μονάδα του πλαισίου είναι γραμμικός, κάτι που διευκολύνει ιδιαίτερα την τακτοποίηση των εκάστοτε αρχείων, αλλά και την κατανόηση του τρόπου, με τον οποίο τα διάφορα μέρη επικοινωνούν μεταξύ τους. Το γεγονός, επίσης, ότι πρόκειται για λογισμικό ανοιχτού κώδικα το κάνει ιδιαίτερα επεκτάσιμο, αλλά και εύκολα παραμετροποιήσιμο ανάλογα με τις ανάγκες τις εκάστοτε εφαρμογής.

Ένα από τα πλεονεκτήματα του CodeIgniter είναι ότι δημιουργεί καθαρά και εύκολα αναγνωρίσιμα από τις μηχανές αναζήτησης Ενιαίους Εντοπιστές Πόρων (Uniform Resource Locators - URLs). Το CodeIgniter αντί να χρησιμοποιεί το πρότυπο “query-string” στη δημιουργία URLs που σχετίζεται με όλα τα δυναμικά συστήματα, παρέχει μια μέθοδο που βασίζεται σε διαχωρισμό τμημάτων (segment based). Αυτή η μέθοδος, άκρως συμβατή με το πρότυπο MVC, έχει την παρακάτω μορφή.

example.com/class/function/ID

1. Το πρώτο τμήμα αναπαριστά τον ελεγκτή που πρέπει να κληθεί.
2. Το δεύτερο αναπαριστά τη συνάρτηση ή τη μέθοδο που πρέπει να κληθεί
3. Το τρίτο, και οποιοδήποτε άλλο τμήμα μπορεί να υπάρξει, αναπαριστά το ID και οποιαδήποτε άλλη μεταβλητή, πρόκειται να αποσταλεί στον ελεγκτή.

Επίσης, η κλάση URI και ο βοηθός URL Helper περιέχουν συναρτήσεις που διευκολύνουν ολόκληρη τη διαχείριση των URI δεδομένων. Επίσης, το URI Routing που διαθέτει το CodeIgniter δίνει τη δυνατότητα επαναπροσδιορισμού των URLs για ακόμη περισσότερη ευελιξία.

Τέλος, ένας ακόμα λόγος που συντείνει στην επιλογή του CodeIgniter ως ιδανικού πλαισίου λογισμικού PHP για την ανάπτυξη εφαρμογών, είναι ότι διαθέτει πλήρη τεκμηρίωση για όλα σχεδόν τα θέματα τόσο του MVC προτύπου όσο και διάφορων άλλων θεμάτων που αφορούν την ανάπτυξη εφαρμογών, που μπορούν να σταθούν αρωγός στην αντιμετώπιση οποιαδήποτε προβλήματος ή δυσκολίας.

3.6 To Twitter Bootstrap

Το Twitter Bootstrap αποτελεί μια συλλογή από εργαλεία δημιουργίας ιστοσελίδων και διαδικτυακών εφαρμογών. Αποτελείται από πρότυπα σχεδίασης (design templates) για φόρμες, κουμπιά, τυπογραφίες, πλοήγηση και άλλα συστατικά δημιουργίας διεπαφών που βασίζονται σε HTML και CSS, καθώς επίσης διαθέτει και επεκτάσεις για JavaScript. Η συγκεκριμένη συλλογή εργαλείων είναι συμβατή με όλους τους σύγχρονους φυλλομετρητές (browsers).

Δομή

Η δομή της συγκεκριμένης συλλογής εργαλείων είναι αρθρωτή και κυρίως αποτελείται από την τεχνολογία LESS Stylesheets, μιας δυναμικής γλώσσας δηλαδή για την κατασκευή φύλλων στυλ, η οποία υλοποιεί τα διάφορα μέρη του Twitter Bootstrap. Το αρχείο με το όνομα bootstrap.less περιλαμβάνει όλα τα βασικά στοιχεία των υπολοίπων φύλλων στυλ. Οι χρήστες μπορούν να χρησιμοποιήσουν αυτό το αρχείο και από εκεί εκ των υστέρων να επιλέξουν τα στοιχεία που επιθυμούν να χρησιμοποιήσουν.

Χαρακτηριστικά και πλεονεκτήματα

Τα βασικά χαρακτηριστικά του Twitter Bootstrap που διευκολύνουν τη δημιουργία της εμφάνισης μιας εφαρμογής ή μιας ιστοσελίδας είναι τα παρακάτω:

- Σύστημα πλέγματος και ευέλικτη σχεδίαση. Το Twitter Bootstrap στην αρχική του μορφή διαθέτει ένα πλέγμα δώδεκα στηλών, όπου μπορεί να παραμετροποιηθεί ανάλογα με τις ανάγκες της εκάστοτε εργασίας. Επίσης, διαθέτει μια ποικιλία τεσσάρων διαφορετικών αναλύσεων που μπορούν να χρησιμοποιηθούν ανάλογα με τη συσκευή, για την όποια προορίζεται η εργασία (κινητό, ταμπλέτα, προσωπικός υπολογιστής).
- Κατανοητά φύλλα στυλ (CSS). Διαθέτει ομάδες από φύλλα στυλ, όπου παρέχει ορισμούς για όλα τα κομβικά συστατικά που είναι γραμμένα σε HTML, όπως φόρμες, πίνακες και τυπογραφίες.
- Επαναχρησιμοποιήσιμα συστατικά. Πέρα από τα βασικά συστατικά, προσφέρει και άλλα συστατικά εξίσου σημαντικά στη σχεδίαση διεπαφών, όπως κουμπιά, ετικέτες, εικονίδια, μηνύματα προειδοποιήσεων και μπάρες εξέλιξης μιας διαδικασίας. Αυτά τα συστατικά στην ουσία επεκτείνουν σε λειτουργικότητα τα βασικά.
- Συστατικά JavaScript. Παρέχει επίσης και συστατικά υλοποιημένα σε JavaScript και μάλιστα στην πιο εξελιγμένη μορφή του πρόσθετου JQuery. Οπότε δίνεται η δυνατότητα για υλοποίηση πρόσθετων στοιχείων στη δημιουργία μιας διεπαφής, όπως κουτιά διαλόγου ή καρουζέλ. Επίσης, τα συστατικά αυτά επεκτείνουν τη λειτουργικότητα κάποιων ήδη υπαρχόντων στοιχείων των διεπαφών, συμπεριλαμβανομένης και της λειτουργίας της αυτόματης συμπλήρωσης (auto-complete) των πεδίων εισαγωγής δεδομένων από το χρήστη.

Τέλος, να συμπληρωθεί πως η χρήση του Twitter Bootstrap είναι ιδιαίτερα εύκολη, καθώς το μόνο που απαιτείται είναι το κατέβασμά του και η συμπερίληψή του στην εκάστοτε εργασία ως ένα απλό φύλλο στυλ.

3.7 Η γλώσσα προγραμματισμού Python

Για τις ανάγκες της παρούσας εφαρμογής και πιο συγκεκριμένα για τον τρόπο διαχείρισης του API του Twitter και την αποθήκευση των δεδομένων προς τη MongoDB χρησιμοποιήθηκε η γλώσσα Python. Έτσι, δημιουργήθηκε ένα σενάριο εντολών, που με τη βοήθεια των κατάλληλων βιβλιοθηκών, που θα αναλυθούν σε επόμενο κεφάλαιο, υλοποιήθηκε αυτή η

διεργασία επεξεργασίας της πληροφορίας που δίνεται σε μορφή JSON από το Twitter και αποθήκευσης στη βάση δεδομένων.

Χαρακτηριστικά και πλεονεκτήματα

Η Python είναι μια διερμηνευόμενη (interpreted) γλώσσα προγραμματισμού υψηλού επιπέδου, που σχεδιάστηκε με έμφαση στην εύκολη ανάγνωση του κώδικά της. Η ανάπτυξή της ξεκίνησε το Δεκέμβριο του 1989 από τον Guido van Rossum στο Εθνικό Ινστιτούτο Έρευνας Πληροφορικής και Μαθηματικών της Ολλανδίας (CWI). Ο δημιουργός της εμπνεύστηκε το όνομα της από τους Monty Pythons, το γνωστό γκρουπ κωμικών από τη Μεγάλη Βρετανία.

Υποστηρίζει πολλά είδη προγραμματισμού, όπως αντικειμενοστραφή, διαδικαστικό, συναρτησιακό και άλλα, και παρέχει ένα πλήρως δυναμικό σύστημα τύπων και αυτόματη διαχείριση μνήμης. Είναι μια γλώσσα προγραμματισμού γενικής χρήσης, ιδανική για κατασκευή εφαρμογών όλων των ειδών ακόμα και δυναμικών ιστοσελίδων. Η βασική της βιβλιοθήκη είναι αρκετά μεγάλη και διαθέτει εκτενή τεκμηρίωση.
Μερικά από τα βασικά της χαρακτηριστικά που την κάνουν ιδιαίτερα ελκυστική και εύκολη στη χρήση είναι τα παρακάτω.

- Η Python είναι απλή και μινιμαλιστική γλώσσα σε τέτοιο σημείο που ο κώδικας της μοιάζει με ψευδοκώδικα. Ένα ιδιαίτερο χαρακτηριστικό της γλώσσας είναι η χρήση κενών διαστημάτων (whitespace) για το διαχωρισμό των συντακτικών δομών του προγράμματος, σε αντίθεση με την πρακτική άλλων γλωσσών, όπου για τον ίδιο σκοπό χρησιμοποιούνται ειδικά σύμβολα (πχ αγκύλες). Αυτό, σε συνδυασμό με το ότι χρησιμοποιεί πλήρεις αγγλικές λέξεις στη θέση συμβόλων, καθιστούν τον κώδικα της Python ευανάγνωστο.
- Είναι γλώσσα υψηλού επιπέδου και όπως όλες οι γλώσσες αυτού του είδους αποκρύπτουν όλες τις διεργασίες που λαμβάνουν χώρα σε χαμηλό επίπεδο, όπως παραδείγματος χάρη τη διαχείριση της μνήμης.
- Είναι γλώσσα ελεύθερου και ανοιχτού κώδικα, πράγμα που σημαίνει ότι ο πηγαίος κώδικάς της μπορεί να διαχειρίζεται και να επεκτείνεται ελεύθερα και πλαισιώνεται από μια κοινότητα προγραμματιστών που αφενός διασπείρει την αποκτημένη γνώση γύρω από θέματα που την αφορούν και αφετέρου τη βελτιώνει και την ισχυροποιεί συνεχώς.
- Είναι γλώσσα ερμηνευόμενη που σημαίνει πως δεν χρειάζεται μεταγλώττιση σε δυαδικό κώδικα. Απλά το εκάστοτε πρόγραμμα εκτελείται άμεσα από τον πηγαίο κώδικα. Εσωτερικά, η Python μετατρέπει τον πηγαίο κώδικα σε μια ενδιάμεση μορφή, η οποία ονομάζεται bytecode, έπειτα μεταφράζει το bytecode στην εγγενή γλώσσα του συστήματος και τελικά τρέχει τον κώδικα. Όλη αυτή η διαδικασία διευκολύνει τη χρήση της Python, αφού δεν υπάρχει η ανάγκη μεταγλώττισης ή φόρτωσης συγκεκριμένων βιβλιοθηκών.
- Η Python υποστηρίζει τόσο διαδικασιοστραφή όσο και αντικειμενοστραφή προγραμματισμό. Στις διαδικασιοστραφείς γλώσσες, το πρόγραμμα χτίζεται πάνω σε διαδικασίες ή συναρτήσεις, οι οποίες δεν είναι τίποτε άλλο παρά επαναχρησιμοποιήσιμα κομμάτια κώδικα. Στις αντικειμενοστραφείς γλώσσες το πρόγραμμα χτίζεται πάνω σε αντικείμενα, τα οποία συνδυάζουν δεδομένα και λειτουργικότητα. Η Python έχει έναν πολύ ισχυρό, αλλά απλό τρόπο υποστήριξης του αντικειμενοστραφούς προγραμματισμού, ειδικά όταν συγκρίνεται με γλώσσες, όπως η C++ ή η Java.
- Είναι επεκτάσιμη και έτσι υπάρχει η δυνατότητα ενσωμάτωσης της Python μέσα σε προγράμματα γραμμένα σε C/C++, για να τους επιτρέπεται η εκτέλεση σεναρίων εντολών με πιο εύκολο τρόπο.

- Τέλος, εγγενώς η Python διαθέτει μια τεράστια συλλογή από βιβλιοθήκες, που προσφέρουν μεγάλη βοήθεια, ώστε η συγγραφή του κώδικα να μην ξεκινάει από την αρχή χωρίς κανένα υπόβαθρο. Αυτές βιβλιοθήκες αφορούν κανονικές εκφράσεις, ενσωμάτωση βάσεων δεδομένων, νημάτωση (threading), γραφικά περιβάλλοντα δημιουργίας διεπαφών (GUI) κά.

Κεφάλαιο 4 - Αρχιτεκτονική της εφαρμογής

4.1 Εισαγωγή

Στο κεφάλαιο που ακολουθεί θα περιγραφούν όλα τα επιμέρους τμήματα που αποτελούν την εφαρμογή που αναπτύχθηκε και όλες οι διαδικασίες που επιτελούνται μέχρι το τελικό αποτέλεσμα που εμφανίζεται στο χρήστη. Η περιγραφή θα ξεκινήσει από τον τρόπο συλλογής των κειμενικών δεδομένων από το Twitter, θα ακολουθήσει η αποθήκευσή τους στη MongoDB και εν συνεχεία το πέρασμα των δεδομένων στο ElasticSearch, το οποίο αποτελεί έναν εναλλακτικό τρόπο αποθήκευσης και ευρετηρίασης. Επίσης, θα παρουσιαστεί διεξοδικά η διεπαφή της εφαρμογής, όπως αυτή διαρθρώνεται, ακολουθώντας την αρχιτεκτονική Μοντέλου – Άποψης – Ελεγκτή, αλλά και όλα τα υπόλοιπα πρόσθετα τμήματά της. Τέλος, με τη βοήθεια της Ενοποιημένης Γλώσσας Μοντελοποίησης (UML) θα παρουσιαστούν με οπτικό τρόπο όλες οι λειτουργίες της εφαρμογής και ο τρόπος αλληλεπίδρασής της με τους χρήστες.

4.2 Η διεπαφή προγραμματισμού εφαρμογών (API) του Twitter

Πριν την περιγραφή της διαχείρισης των κειμενικών δεδομένων, είναι απαραίτητο να περιγραφεί η δομή των κειμενικών δεδομένων και ο τρόπος διάθεσής τους από το Twitter. Αυτό θα γίνει περισσότερο εφικτό, περιγράφοντας τη διεπαφή προγραμματισμού εφαρμογών API που προσφέρεται από το Twitter, ώστε να διευκολύνεται η διαχείριση και η επεξεργασία των δεδομένων του.

Δομή Twitter API

Το συγκεκριμένο API και συγκεκριμένα το REST API v1.1 διαθέτει μια μεγάλη ποικιλία από επιλογές και διευκολύνσεις για τη διαχείριση των δεδομένων. Για παράδειγμα, δίνεται η δυνατότητα να συλλεχθούν συγκεκριμένες χρονοσειρές (timelines) από tweets, δηλαδή συλλογές από τα πιο πρόσφατα tweets κάποιου χρήστη είτε κάποιας συγκεκριμένης θεματικής ενότητας (hashtag), όπως επίσης και άλλα στοιχεία που σχετίζονται με στοιχεία των χρηστών, όπως η γεωγραφική θέση που έχουν δηλώσει ή με αναζητήσεις σε λέξεις-κλειδιά που έχουν εμφανιστεί μέσα σ' ένα συγκεκριμένο χρονικό διάστημα.

Αυτό που χρησιμοποιείται στα πλαίσια της παρούσας εφαρμογής είναι το API που σχετίζεται με τη ροή των δεδομένων (streaming API). Το Twitter προσφέρει τρεις διαφορετικές παραλλαγές αυτών των ροών που δίνουν πρόσβαση στα δεδομένα που δημιουργούνται από τους χρήστες του κοινωνικού δικτύου παγκοσμίως. Αυτά είναι τα παρακάτω

- οι δημόσιες ροές (public streams) που δίνουν πρόσβαση σ' όλα τα δεδομένα που δεν έχουν κάποιους ιδιωτικούς περιορισμούς. Η περίπτωση αυτή είναι ιδιαίτερα χρήσιμη για συγκέντρωση δεδομένων με γνώμονα κάποιο συγκεκριμένο τομέα έρευνας και για εξόρυξη πληροφορίας (data mining),
- οι ροές χρήστη, που πρόκειται για ροές δεδομένων μεμονωμένων χρηστών και περιλαμβάνουν το σύνολο των στοιχείων τόσο των κειμενικών όσο και οποιωνδήποτε άλλων μεταδεδομένων συνοδεύουν το κάθε tweet ξεχωριστά
- και ροές ιστοσελίδων, που προορίζονται για εξυπηρετητές, προκειμένου να συνδέονται στο Twitter εκ μέρους πολλαπλών χρηστών.
-

Για τις ανάγκες της εφαρμογής που στην ουσία είναι το γέμισμα μιας βάσης δεδομένων με κειμενικά δεδομένα, χωρίς να ανταποκρίνονται σε κάποιο συγκεκριμένο περιορισμό πέρα από την γλώσσα, χρησιμοποιήθηκαν τα tweets που επιστρέφει η δημόσια ροή δεδομένων.

Δομή ενός tweet

Η δομή ενός tweet είναι ιδιαίτερα δαιδαλώδης, καθώς το απλό κείμενο του tweet συνοδεύεται από ένα πλήθος άλλων πληροφοριών και ολόκληρη αυτή η δομή είναι δοσμένη σε μορφή JSON (JavaScript Object Notation). Παρακάτω παρατίθεται η μορφή ενός tweet, όπως αυτή επιστρέφεται από το API.


```
{
    "favorited":false,
    "text":"Testing link https:\/\/t.co\/jvivDycK",
    "possibly_sensitive_editable":true,
    "truncated":false,
    "created_at":"Fri Feb 10 15:51:21 +0000 2012",
    "retweeted":false,
    "retweet_count":0,
    "coordinates":null,
    "source":"\u003Ca href=\"http:\/\/realitytechnicians.com\"
rel=\"nofollow\"\u003EOAuth Dancer Reborn\u003C\/a\u003E",
    "in_reply_to_status_id_str":null,
```

```
    "possibly_sensitive":false,
    "entities":
    {
        "user_mentions":
        [
        ],
        "urls":
        [
            {
                "indices":
                [
                    13,
                    34
                ],
                "url":"https:\/\/t.co\/jvivDycK",

"display_url":"dev.twitter.com\/discussions\/56\u2026",

"expanded_url":"https:\/\/dev.twitter.com\/discussions\/5653"
            }
        ],
        "hashtags":
        [
        ]
    },
    "geo":null,
    "in_reply_to_user_id_str":null,
    "place":null,
    "in_reply_to_user_id":null,
    "in_reply_to_screen_name":null,
    "id_str":"16799910160932044 8",
    "user":
    {
        "default_profile":false,
        "notifications":null,
        "profile_use_background_image":true,
        "time_zone":null,
        "created_at":"Wed Mar 03 19:37:35 +0000 2010",
        "verified":false,
        "geo_enabled":true,
        "profile_text_color":"333333",

"profile_background_image_url":"http:\/\/a0.twimg.com\/profile_backgro
und_images\/80151733\/oauth-dance.png",
        "description":"",
        "default_profile_image":false,
        "profile_link_color":"0084B4",
        "url":"http:\/\/bit.ly\/oauth-dancer",
        "follow_request_sent":null,
```

```
        "favourites_count":6,
        "lang":"en",

"profile_background_image_url_https":"https:\/\/si0.twimg.com\/profile
_background_images\/80151733\/oauth-dance.png",
        "profile_background_color":"C0DEED",
        "followers_count":23,

"profile_image_url":"http:\/\/a2.twimg.com\/profile_images\/730275945\
/oauth-dancer_normal.jpg",
        "contributors_enabled":false,
        "profile_background_tile":true,
        "protected":false,

"profile_image_url_https":"https:\/\/si0.twimg.com\/profile_images\/73
0275945\/oauth-dancer_normal.jpg",
        "location":"San Francisco,
         CA",
        "profile_sidebar_fill_color":"DDEEF6",
        "name":"OAuth Dancer",
        "id_str":"119476949",
        "listed_count":0,
        "following":null,
        "screen_name":"oauth_dancer",
        "id":119476949,
        "is_translator":false,
        "show_all_inline_media":false,
        "statuses_count":153,
        "utc_offset":null,
        "friends_count":14,
        "profile_sidebar_border_color":"C0DEED"
    },
    "id":167999101609320448,
    "contributors":null,
    "in_reply_to_status_id":null
}
```


Πλεονεκτήματα JSON

Η παραπάνω δομή δεδομένων είναι χαρακτηριστικό παράδειγμα της JavaScript Object Notation (JSON). Κυρίως η JSON εξαιτίας του ότι χρειάζεται μικρή υπολογιστική δύναμη, για να επεξεργασθεί, εφόσον πρόκειται για απλή συμβολοσειρά, χρησιμοποιείται ευρέως για την αναπαράσταση κειμενικών δεδομένων. Επίσης, ένας επιπλέον λόγος που την καθιστούν ιδιαίτερα ελκυστική προς χρήση είναι το γεγονός πως ο τρόπος που δομεί τα δεδομένα γίνεται αντιληπτός με ευκολία από το ανθρώπινο μάτι. Ταυτόχρονα όμως είναι ακριβώς το ίδιο εύκολο και από τις μηχανές να διαπερνούν τα δεδομένα, αλλά και να δημιουργούν δομές JSON.

Χαρακτηριστικά JSON

Η JSON περιλαμβάνει δύο διαφορετικές δομές. Η πρώτη είναι μια συλλογή από ζευγάρια ονόματος-τιμής. Ανάλογα με την γλώσσα προγραμματισμού αυτή η συλλογή μπορεί να θεωρηθεί ως αντικείμενο, εγγραφή, δομή λεξικό, λίστα κά. Η δεύτερη δομή είναι μια ταξινομημένη λίστα τιμών. Στις πιο πολλές γλώσσες προγραμματισμού, αυτό μεταφράζεται ως πίνακας ή λίστα.
Παραδείγματος χάρη, ένα αντικείμενο JSON, όπως είναι η παραπάνω δομή ενός tweet, θεωρείται ως ένα μη ταξινομημένο σύνολο από ζευγάρια ονόματος και τιμής. Ένα αντικείμενο ξεκινά με μια { (αριστερή αγκύλη) και τελειώνει με μια } (δεξιά αγκύλη). Κάθε όνομα ακολουθείται από : (άνω και κάτω τελεία) και τα ζεύγη όνομα/τιμή διαχωρίζονται από ένα , (κόμμα). Όπως, παραδείγματος χάρη, τα δεδομένα που τελικά αποθηκεύονται στην βάση δεδομένων που έχουν την παρακάτω μορφή.

```
{  "_id" : ObjectId("522a027cbae211291ca1260b"),
  "date" : "09/06/2013",
  "followers" : 264,
  "user" : "Ally :))",
  "tweets" : 13545,
  "text" : "RT @WWATourUpdate: TO BE ON THE FIRST PAGE OF NAMES
WRITTEN WIN 5 ONE DIRECTION TICKETS RT THIS (the most can be 200)",
  "followings" : 603,
  "retweet_count" : 0,
  "nickname" : "_idknialler",
  "location" : "under my rock" }
```

Για τις ανάγκες της εφαρμογής η δομή JSON που δίνει το Twitter API απλοποιήθηκε στην παραπάνω μορφή, έτσι ώστε να διατηρούνται στη βάση δεδομένων μόνο εκείνα τα δεδομένα που είναι χρήσιμα για την εφαρμογή. Τα δεδομένα αυτά είναι το id του tweet, η ημερομηνία που δημοσιεύτηκε, οι ακόλουθοι (followings) του χρήστη, το όνομα του χρήστη, το ίδιο το κείμενο του tweet, ο αριθμός των χρηστών (followers) που ακολουθεί ο χρήστης, ο αριθμός των αναδημοσιεύσεων (retweet) του tweet από άλλους χρήστες, το υποκοριστικό όνομα του χρήστη (username) και η τοποθεσία του, όπως αυτός την έχει δηλώσει. Ο τρόπος της επεξεργασίας της αρχικής δομής και της αποθήκευσης των δεδομένων θα αναλυθεί αμέσως μετά.

4.3 Διαδικασία αποθήκευσης δεδομένων στη MongoDB

Το γεγονός ότι τα δεδομένα προσφέρονται από το Twitter σε μορφή JSON οδήγησε στη σκέψη η αποθήκευση τους να γίνει σε NoSQL βάση δεδομένων, εφόσον αυτό το είδος των βάσεων χρησιμοποιεί ως κύρια μορφή αποθήκευσης επίσης την JSON. Ακόμα, το ζήτημα της σύνδεσης του Twitter API με τη βάση και η αποθήκευση των tweets λύθηκε με τη δημιουργία ενός αρχείου σεναρίου εντολών (script file) στη γλώσσα Python. Η υλοποίηση του αρχείου έγινε στην Python αφενός εξαιτίας της ευκολίας ανάπτυξης κώδικα, καθώς πρόκειται για γλώσσα υψηλού επιπέδου και αφετέρου την ύπαρξη βιβλιοθηκών που διευκόλυναν τόσο τη διαχείριση του API όσο και την άμεση αποθήκευση.

Δήλωση εφαρμογής στο Twitter API

Το πρώτο βήμα πριν τη δημιουργία του αρχείου σεναρίου εντολών ήταν να δηλωθεί η παρούσα εφαρμογή ως μια έγκυρη εφαρμογή του Twitter. Το γνωστό κοινωνικό δίκτυο διαθέτει ένα πρωτόκολλο ελέγχου ταυτότητας (authentication protocol) που ονομάζεται OAuth και στην ουσία επιτρέπει στις διάφορες εφαρμογές να κάνουν χρήση των υπηρεσιών του, εφόσον πρώτα έχουν δηλωθεί και έχουν εξασφαλίσει τους απαραίτητους κωδικούς πιστοποίησης. Συγκεκριμένα, μετά το τέλος της εγγραφής δημιουργείται ένας συνδυασμός από τέσσερις κωδικούς (CONSUMER KEY, CONSUMER SECRET, ACCESS TOKEN, ACCESS TOKEN SECRET), ο οποίος εξασφαλίζει την πρόσβαση στις παρεχόμενες υπηρεσίες του API.

Δημιουργία αρχείου σεναρίου εντολών

Στη συνέχεια, οι δύο βιβλιοθήκες της Python που συμβάλλουν στις δύο καίριες εργασίες που επιτελεί το αρχείο σεναρίου εντολών είναι η Tweepy και η Pymongo. Η Tweepy δίνει πρόσβαση στο Twitter API και συγκεκριμένα η μέθοδός της tweepy.streaming.Stream() επιστρέφει τη δημόσια ροή (public stream) του Twitter, δεχόμενη ως ορίσματα τους κωδικούς ελέγχου ταυτότητας της εφαρμογής, την κλάση CustomStreamListener(), η οποία δημιουργήθηκε ώστε να επιλεχθούν συγκεκριμένα πεδία που πρέπει να εμφανίζονται και όχι ολόκληρη η δομή JSON που παραχωρεί το Twitter και η χρονική καθυστέρηση, με την οποία θα γίνεται η επικοινωνία με το API.

Η Pymongo έχει ως ρόλο να επικοινωνεί με την MongoDB και να αποθηκεύει τα τροποποιημένα δεδομένα από το Tweepy. Πιο συγκεκριμένα, η Pymongo εδραιώνει τη σύνδεση με τη MongoDB στη διεύθυνση localhost/27017 και στη συνέχεια δημιουργεί μια κενή βάση δεδομένων και μια συλλογή, όπου θα τοποθετηθούν τα δεδομένα. Επίσης, δημιουργεί τα πεδία, όπου θα μπει το κάθε είδος των δεδομένων (χρήστης, tweet, τοποθεσία κοκ) και τροποποιεί, όταν χρειάζεται τα δεδομένα, ώστε να αποθηκευθούν με συγκεκριμένο τρόπο. Παραδείγματος χάρη, φροντίζει τα κειμενικά δεδομένα να έχουν κωδικοποίηση UTF-8 και η ημερομηνία να είναι της μορφής (mm/dd/yyyy), ώστε να εξασφαλιστεί η ομοιομορφία των δεδομένων, αναγκαία για το στάδιο της αναζήτησης των δεδομένων. Επίσης, στα πλαίσια της αποθήκευσης διενεργείται και ένας έλεγχος στο πεδίο, όπου δηλώνεται η γλώσσα που χρησιμοποιεί ο χρήστης, προκειμένου να γίνονται αποδεκτά μόνο τα tweets χρηστών που έχουν δηλώσει την αγγλική γλώσσα. Αυτό βοηθάει, ώστε να μη γεμίζει η βάση με άχρηστα για την αναζήτηση tweets που είναι γραμμένα σε αλφάβητα που δεν χρησιμοποιούν λατινικούς χαρακτήρες (κινέζικα, αραβικά κοκ).

Το παραπάνω αρχείο σεναρίου εντολών δεν έχει ενταχθεί ως ένα κομμάτι της διαδικτυακής εφαρμογής, αλλά εκτελείται ανεξάρτητα, για να γεμίζει τη βάση με δεδομένα, τα οποία μετέπειτα χρησιμοποιούνται για τις αναζητήσεις που εκτελεί η εφαρμογή.

4.4 Διαδικασία σύνδεσης MongoDB - ElasticSearch

Όπως προαναφέρθηκε, σκοπός της εφαρμογής είναι να διενεργεί αναζητήσεις στα δεδομένα τόσο μέσω της MongoDB όσο και του ElasticSearch. Ο καλύτερος τρόπος, για να περάσουν και ταυτόχρονα να ευρετηριαστούν τα δεδομένα από τη MongoDB στο ElasticSearch αποδείχθηκε ένα άρθρωμα του ElasticSearch που ονομάζεται MongoDB River. Γενικότερα, τα αρθρώματα (plugins) του ElasticSearch συμβάλλουν στην επέκταση της βασικής του λειτουργικότητας με πρόσθετες δυνατότητες. Μερικά από τα πιο γνωστά αρθρώματα που έχουν αναπτυχθεί επίσημα από τους κατασκευαστές του ElasticSearch εκτός από το MongoDB River είναι το CouchDB River, το Twitter River και το Wikipedia River. Επίσης, παρόμοια αρθρώματα έχουν αναπτυχθεί και από την κοινότητα που ασχολείται με το ElasticaSearch, όπως το Dropbox River και το RSS River. Όπως γίνεται κατανοητό και από τα

ονόματά τους, σκοπός όλων αυτών των αρθρωμάτων είναι η μεταφορά δεδομένων προς το ElasticSearch από διάφορες πηγές.

Σύνδεση MongoDB River – MongoDB - ElasticSearch

Η όλη διαδικασία σύνδεσης της MongoDB με το ElasticSearch επιβάλλει ορισμένες προϋποθέσεις. Αρχικά είναι απαραίτητο τα τρία συστατικά που φιλοξενούν και διαχειρίζονται τα δεδομένα, δηλαδή η MongoDB, το ElasticSearch και το MongoDB River, να είναι εναρμονισμένα βάσει των εκδόσεων. Υπάρχει συγκεκριμένος τρόπος που πρέπει να είναι εγκαταστημένα, ώστε η λειτουργία τους να είναι απρόσκοπτη. Στην περίπτωση της εφαρμογής, οι εγκατεστημένες εκδόσεις των συστατικών είναι το MongoDB River Plugin - 1.6.11, το ElasticSearch - 0.90.2 και η MongoDB 2.4.5. Επίσης, ακόμα και αν έχει ήδη εγκατασταθεί και τρέχει ένα στιγμιότυπο της MongoDB, αυτό δεν είναι αρκετό, ώστε να διεκπεραιωθεί η μεταφορά των δεδομένων προς το ElasticSearch. Αυτό οφείλεται στο γεγονός ότι το άρθρωμα MongoDB River, προκειμένου να τεθεί σε λειτουργία, χρειάζεται η MongoDB να έχει σε λειτουργία ένα σύνολο από εξυπηρετητές αντικατάστασης (replica set), όπως επίσης και το replica set oplog (operations log), δηλαδή μια συλλογή σταθερού μεγέθους (capped collection), όπου αποθηκεύονται συνεχώς όλες οι διεργασίες που μεταβάλλουν τα δεδομένα που είναι αποθηκευμένα στις διάφορες φιλοξενούμενες βάσεις δεδομένων. Όταν αυτή η συλλογή γεμίσει, τότε τα δεδομένα γράφονται πάνω στα ήδη υπάρχοντα. Το MongoDB River εξαρτάται από αυτήν την λειτουργία, γιατί μέσω αυτού του τρόπου προωθεί τις αλλαγές στην εκάστοτε βάση δεδομένων προς το ElasticSearch που έχει δεχθεί τα αρχικά δεδομένα από τη MongoDB και περιμένει να ενημερωθεί για τις διάφορες αλλαγές που συντελούνται σ' αυτήν. Εξάλλου, η MongoDB δεν διαθέτει κάποιου άλλου είδους εγγενές σύστημα επισήμανσης για αλλαγές στα δεδομένα των βάσεων που φιλοξενεί, επομένως το άρθρωμα χρησιμοποιεί την oplog συλλογή για τη δουλειά αυτή.

Δημιουργία ανάλυσης

Στη συνέχεια και εφόσον η MongoDB βρίσκεται στην παραπάνω λειτουργία, τα δεδομένα θα πρέπει να περάσουν μέσω του MongoDB River στο ElasticSearch. Πριν γίνει αυτό όμως θα πρέπει να έχει προετοιμαστεί κατάλληλα και το ElasticSearch, για να δεχθεί τα δεδομένα από την βάση. Η κατασκευή του ευρετηρίου που θα αποδεχθεί τα δεδομένα που θα εισρεύσουν απαιτεί την δημιουργία τόσο της ανάλυσης (analysis) όσο και της χαρτογράφησης (mapping).

Η ανάλυση ταυτίζεται με την δήλωση εκείνων των μηχανισμών που θα εφαρμοστούν πάνω στα δεδομένα, ώστε αυτά να ευρετηριαστούν με τον επιθυμητό τρόπο. Οπότε στην ανάλυση δηλώνονται και επιλέγονται οι κατάλληλοι αναλυτές (analyzers) και τα κατάλληλα φίλτρα. Για τις ανάγκες της παρούσας εφαρμογής η μορφή της ανάλυσης που χρησιμοποιείται είναι η εξής.

```
"analysis":{
            "filter":{
                "trip_ngram": {
                    "type" : "edgeNgram",
                    "max_gram" : 15,
                    "min_gram" : 3 }
            },
           "analyzer":{
              "index_ngram_analyzer" : {
                  "type" : "custom",
                  "tokenizer" : "standard",
                  "filter" : ["standard", "lowercase", "trip_ngram"]
              },
```

```
            "search_ngram_analyzer" : {
              "type" : "custom",
              "tokenizer" : "standard",
              "filter" : ["standard", "lowercase"]
            }
        }
    }
```

Συγκεκριμένα, στην παρούσα ανάλυση κατασκευάζονται δύο διαφορετικοί αναλυτές. Ο index_ngram_analyzer, ο οποίος αφορά τα δεδομένα που θα δέχεται το ElasticSearch από τη βάση και ο search_ngram_analyzer που θα εφαρμόζει ακριβώς τους ίδιους κανόνες και στα ερωτήματα που θα τίθενται από το χρήστη. Τελικός σκοπός είναι τα ερωτήματα που δίνονται να έχουν ακριβή ταύτιση με τα δεδομένα που θα γίνεται προσπάθεια να ανακτηθούν από τη βάση. Οι παραπάνω δύο αναλυτές, όπου είναι της μορφής custom, γιατί δεν υπάρχουν προκατασκευασμένοι από το ElasticSearch περιέχουν τον προκαθορισμένο διαχωριστή λεκτικών μονάδων (tokenizer) που διαχωρίζει την εκάστοτε λέξη μιας πρότασης σε μεμονωμένες λεκτικές μονάδες και δύο επίσης προκαθορισμένα φίλτρα. Το βασικό φίλτρο, όπου προσπαθεί να εξομαλύνει τις λεκτικές μονάδες που παράχθηκαν από το διαχωριστή λεκτικών μονάδων και το φίλτρο μικρογράμματων λέξεων (lowercase), όπου όλα τα κειμενικά δεδομένα μετατρέπονται σε μικρογράμματη γραφή. Το τρίτο φίλτρο που εφαρμόζεται μόνο στον πρώτο αναλυτή που είναι υπεύθυνος μόνο για τα δεδομένα είναι και αυτός τύπου custom και εμπεριέχει έναν διαχωριστή λεκτικών μονάδων της κατηγορίας edgeNgram. Αυτός ο διαχωριστής είναι μια παραλλάγη του Ngram διαχωριστή, ο οποίος δημιουργεί υποσύνολα της εκάστοτε λεκτικής μονάδας, όπου το μήκος τους είναι ίσο με το εύρος του N. Στην περίπτωση του edgeNgram αυτά τα υποσύνολα ξεκινούν μόνο από την αρχή της λέξης και στην περίπτωση της προκειμένης ανάλυσης έχουν εύρος από 3 μέχρι 15 γράμματα. Συνεπώς, η λεκτική μονάδα “players” αναλύεται ως “pla”, “play”, “playe”, “player”, “players”, επομένως αν αναζητηθεί η λέξη “play” από το χρήστη, τότε επιστρέφεται και η λέξη “players”, αλλά επίσης και οι λέξεις “playing”, “played” κοκ.

Δημιουργία χαρτογράφησης

Για να ολοκληρωθεί το στάδιο αυτό είναι απαραίτητος και ο ορισμός της χαρτογράφησης που εν προκειμένω έχει την ακόλουθη μορφή.

```
"mappings":{
      "new2":{
         "properties":{
            "date" : {"type" : "date", "format" : "MM/dd/yy"},
            "followers" : {"type" : "long", "index" : "not_analyzed"},
            "user" : {"type" : "string"},
            "tweets" : {"type" : "long", "index" : "not_analyzed"},
            "text":{
              "type" : "multi_field",
                "fields" : {
                    "text.autocomplete" : {
                        "search_analyzer" : "search_ngram_analyzer",
                        "index_analyzer" : "index_ngram_analyzer",
                        "type" : "string"
            }
        }
    }
```

```
    },
            "followings" : {"type" : "long", "index" :
"not_analyzed"},
            "retweet_count" : {"type" : "long", "index" :
"not_analyzed"},
            "nickname" : {"type" : "string"},
            "location" : {"type" : "string"}
        }
      }
    }
```

Στη χαρτογράφηση ορίζονται όλα τα πεδία των δεδομένων που θα αποθηκευτούν στο ElasticSearch. Επίσης, ορίζεται ο τύπος των δεδομένων, η μορφή τους, αλλά και οι αναλυτές που θα εφαρμοστούν. Στην προκειμένη περίπτωση, οι δύο αναλυτές που ορίστηκαν στην ανάλυση θα εφαρμοστούν στο πεδίο text.autocomplete, όπου είναι παιδί του πεδίου text. Το πεδίο text, επειδή είναι τύπου multi_field, έχει τη δυνατότητα να έχει παιδιά. Εδώ στο πεδίο αυτό θα φιλοξενηθούν όλα τα νέα token που θα δημιουργηθούν από τον index_ngram_analyzer και θα προσπαθήσουν να ταυτιστούν με τα token που θα προέρχονται από τον search_ngram_analyzer. Η ένδειξη not_analyzed σημαίνει πως το περιεχόμενο των συγκεκριμένων πεδίων είναι διαθέσιμο για αναζήτηση, αλλά δεν έχει υποστεί οποιαδήποτε επεξεργασία από κάποιον διαχωριστή λεκτικών μονάδων.

Εισροή δεδομένων στο ElasticSearch

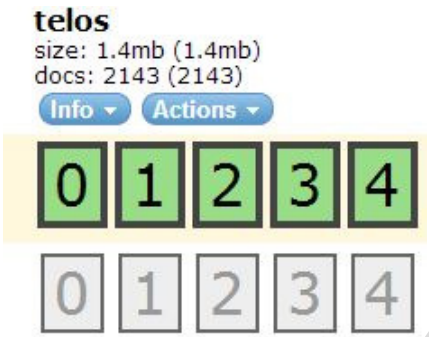
Εφόσον, λοιπόν, με την κατασκευή της ανάλυσης και της χαρτογράφησης ολοκληρώθηκε η προετοιμασία του ElasticSearch, το μόνο που απομένει είναι η εισροή των δεδομένων στο ευρετήριο. Αυτό θα γίνει μένω του River με τον παρακάτω τρόπο.

```
curl -XPUT "localhost:9200/_river/telos/_meta" -d '
 { "type": "mongodb",
        "mongodb": { "db": "new2", "collection": "new2" },
        "index": { "name": "telos", "type": "new2"}
        }'
```

Μέσω της εντολής –curl και της γραμμής εντολών ορίζεται πως στο ευρετήριο με το όνομα telos και τον τύπο new2 θα εισρεύσουν τα δεδομένα που βρίσκονται στη MongoDB και πιο συγκεκριμένα στη βάση δεδομένων new2 που διαθέτει τη συλλογή new2. Έπειτα από αυτό το βήμα τα δεδομένα περνούν από την επεξεργασία που αναλύθηκε και τοποθετούνται στο συγκεκριμένο ευρετήριο.

Επίβλεψη μέσω του ElasticSearch Head Master

Η επίβλεψη του ElasticSearch και των ευρετηρίων του μπορεί να γίνει μέσω μια εφαρμογής που ονομάζεται ElasticSearch Head Master. Μέσω αυτής της εφαρμογής φαίνεται το μέγεθος του ευρετηρίου, ο αριθμός των εγγράφων που εμπεριέχει, η κατάστασή του όσων αφορά τα θραύσματα που χρησιμοποιεί και την κατανομή τους και προσφέρεται η δυνατότητα για μια επίβλεψη των δεδομένων που αποθηκεύονται. Το ευρετήριο τέλος έχει την παρακάτω μορφή.



Εικ.1 Το ευρετήριο telos του ES, όπως φαίνεται από το ES Head Master

Σύνταξη αναζήτησης απευθείας προς το ElasticSearch

Επίσης, ιδιαίτερα χρήσιμο τόσο για την δημιουργία του ευρετηρίου όσο και για την εποπτεία των ερωτημάτων και κατά συνέπεια ότι όλα λειτουργούν ομαλά είναι το άρθρωμα του φυλλομετρητή Chrome με το όνομα Sense. Το Sense είναι μια εφαρμογή που στην ουσία ο χρήστης μπορεί απευθείας να συντάξει τόσο την ανάλυση όσο και τη χαρτογράφηση σε μορφή JSON. Επίσης, μπορεί να συντάξει ερωτήματα προς το ευρετήριο του ElasticSearch πάλι στην ίδια μορφή, ώστε να επαληθεύσει την ορθότητα του ευρετηρίου και να περάσει στην επόμενη φάση, που είναι η ένταξη αυτού του ευρετηρίου σε κάποια εφαρμογή. Ένα ερώτημα σε μορφή JSON έχει την ακόλουθη μορφή.

```
"query": {
        "filtered": {
            "query": {
                "query_string": {
                    "query": "play",
                        "fields" : [ "text.autocomplete"
                            ]
                }
            },
            "filter": {
            "and": [
                {
                    "range": {
                        "date": {
                            "from": "07/20/2013",
                            "to": "10/20/2013"
                        }
                    }
                },
                {
                    "term": {
                        "location": "usa"
                    }
                }
            ]
        }
    }
```

```
 }
```

και η απόκριση από το ElasticSearch, όπως φαίνεται μέσω του Sense είναι η εξής

```
"{
   "took": 7,
   "timed_out": false,
   "_shards": {
      "total": 5,
      "successful": 5,
      "failed": 0
   },
   "hits": {
      "total": 1,
      "max_score": 1.6022295,
      "hits": [
         {
            "_index": "telos",
            "_type": "new2",
            "_id": "52619ebdbae21101a0be1a57",
            "_score": 1.6022295,
            "_source": {
               "_id": "52619ebdbae21101a0be1a57",
               "date": "10/18/2013",
               "followers": 0,
               "user": "Marian Gardner",
               "tweets": 5,
               "text": "@LindsayEvelyn hi will you play with me??
http://t.co/RukvV4ihwq",
               "followings": 0,
               "retweet_count": 0,
               "nickname": "marian2135",
               "location": "USA"
            }
         }
      ]
   }
}
```

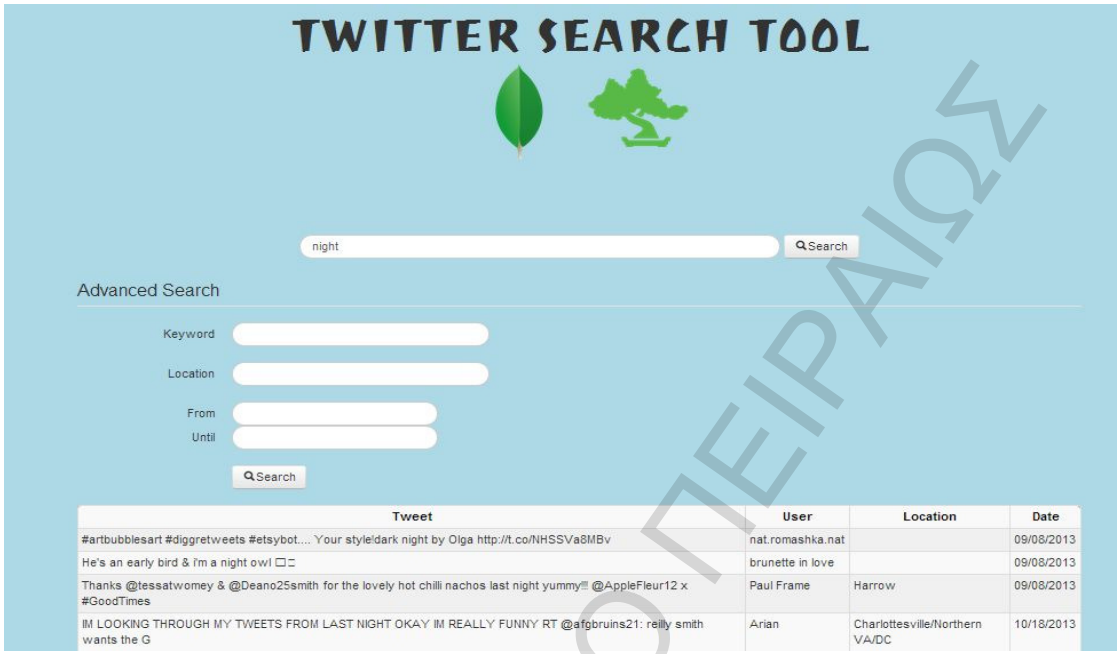
Επομένως, επαληθεύεται εκ των πραγμάτων πως η λειτουργία είναι κανονική, εφόσον επιστράφηκε ένα αποτέλεσμα, όπου το κείμενο περιέχει τη λέξη-κλειδί που δόθηκε μέσα στο συγκεκριμένο εύρος ημερομηνιών και από τη δοσμένη τοποθεσία. Παρακάτω θα αναλυθεί πως αυτή η τυπολογία των ερωτημάτων θα δομηθεί μέσω της PHP με τη βοήθεια της Elastica, μιας βιβλιοθήκης PHP για τη διαχείριση του ElasticSearch μέσω της PHP.
Οπότε στο σημείο αυτό πλέον τόσο η MongoDB όσο και το ElasticSearch διαθέτουν ακριβώς τα ίδια δεδομένα, αποθηκευμένα με το σωστό τρόπο σύμφωνα με τις προδιαγραφές τους. Στην ουσία η βάση της εφαρμογής, το back-end δηλαδή, είναι έτοιμο, ώστε να συνδεθεί με τη διεπαφή χρήστη που αποτελεί το front-end της εφαρμογής.

4.5 Η διεπαφή χρήστη (user interface) της εφαρμογής

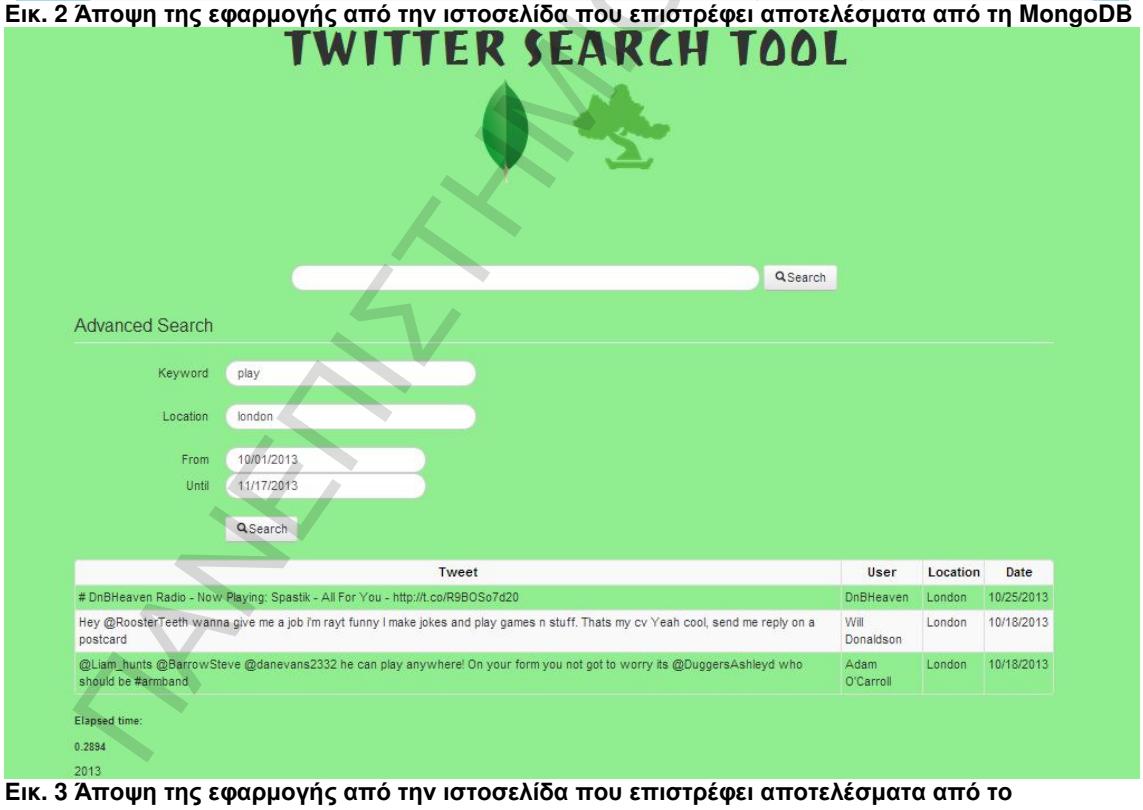
Η διεπαφή βασίζεται από πλευρά αρχιτεκτονικής στο πρότυπο Μοντέλο – Άποψη – Ελεγκτής (MVC), το οποίο υλοποιήθηκε με τη βοήθεια της εφαρμογής διαδικτυακού πλαισίου λογισμικού (Web Application Framework) CodeIgniter. Όπως προαναφέρθηκε και στην περιγραφή της αρχιτεκτονικής MVC, η υλοποίηση μιας διεπαφής βάσει των αρχών αυτού του προτύπου διαχωρίζει τα αρχεία σε τρεις διακριτές κατηγορίες όσα δηλαδή και τα τμήματα που αποτελούν το MVC πρότυπο.

Περιγραφή διεπαφής

Πριν η περιγραφή της διεπαφής προχωρήσει στον τρόπο υλοποίησής της και στις επιλογές που υιοθετήθηκαν, θα γίνει μια ανάλυση της σχεδίασής της. Η διεπαφή αποτελείται από δύο ιστοσελίδες, όπου από την καθεμία ο χρήστης μπορεί να επικοινωνεί με τα δεδομένα που είναι αποθηκευμένα στην MongoDB και στο ElasticSearch αντίστοιχα. Στην κάθε σελίδα δίνεται η δυνατότητα να διενεργηθούν δύο διαφορετικού είδους αναζητήσεις. Η πρώτη επιλογή είναι μια απλή αναζήτηση μόνο με λέξη-κλειδί και η δεύτερη είναι μια σύνθετη αναζήτηση, όπου ο χρήστης μπορεί να αναζητήσει λέξη-κλειδί μέσα σ' ένα ορισμένο χρονικό περιθώριο και βάσει συγκεκριμένης τοποθεσίας. Αν επιστραφούν αποτελέσματα από τις αναζητήσεις που διενεργήθηκαν, τότε τα αποτελέσματα εμφανίζονται στο κάτω μέρος της διεπαφής σε δεκάδες, αν υπερβαίνουν τα δέκα αποτελέσματα. Αν δεν επιστραφούν αποτελέσματα ή παραβιαστούν οι περιορισμοί των αναζητήσεων που θα περιγραφούν παρακάτω, τότε εμφανίζονται ανάλογα μηνύματα. Στο πάνω μέρος της διεπαφής υπάρχουν σύνδεσμοι με το εικονίδιο της κάθε πηγής δεδομένων, απ' όπου ο χρήστης μπορεί να επιλέξει, ώστε τα αποτελέσματα να του επιστραφούν από αυτή. Επίσης, για την κάθε αναζήτηση επιστρέφεται και ο χρόνος απόκρισης και επιστροφής των πρώτων δέκα ή λιγότερων αποτελεσμάτων, ώστε ο χρήστης να μπορεί να συγκρίνει τις επιδόσεις των δύο διαθέσιμων επιλογών σε ίδιες αναζητήσεις.



Εικ. 2 Άποψη της εφαρμογής από την ιστοσελίδα που επιστρέφει αποτελέσματα από τη MongoDB



Εικ. 3 Άποψη της εφαρμογής από την ιστοσελίδα που επιστρέφει αποτελέσματα από το ElasticSeach

Διαδικασία εμφάνισης αποτελεσμάτων από τη MongoDB με απλή αναζήτηση

Ξεκινώντας την περιγραφή της δομής της κάθε ιστοσελίδας βάσει του προτύπου MVC, τα βασικά αρχεία κατατάσσονται σε 3 κατηγορίες όσα και τα μέρη του προτύπου. Η διεπαφή διαθέτει δύο διαφορετικές ιστοσελίδες και για κάθε μια από αυτές υπάρχει ακριβώς ο ίδιος διαχωρισμός στα αρχεία, αφού για την κάθε μια υπάρχει ξεχωριστός ελεγκτής που ενώνει διαφορετικά μοντέλα και απόψεις.
Στα πλαίσια του μοντέλου MVC, η διεπαφή, όπως τη βλέπει ο χρήστης, κατασκευάζεται από τα αρχεία της άποψης. Η πρώτη σελίδα που εμφανίζεται στο χρήστη είναι αυτή που του δίνει πρόσβαση στα αρχεία που είναι αποθηκευμένα στη MongoDB. Η εφαρμογή δίνει τη δυνατότητα αναζήτησης μεμονωμένων λέξεων και όχι φράσεων. Σ' αυτό το σημείο ο χρήστης έχει δύο επιλογές είτε να κάνει μια απλή αναζήτηση με μια λέξη-κλειδί είτε να κάνει μια σύνθετη αναζήτηση, που θα αποτελείται από τη λέξη-κλειδί, μια τοποθεσία και ένα εύρος ημερομηνιών.

Έστω ότι ο χρήστης επιλέγει την απλή αναζήτηση, δίνοντας μια απλή λέξη στη φόρμα αναζήτησης. Αμέσως μόλις πατήσει το κουμπί της αναζήτησης, τότε η λέξη αυτή αποθηκεύεται σε μια μεταβλητή με το όνομα search και αποστέλλεται με τη μέθοδο post στον ελεγκτή με το όνομα test και συγκεκριμένα στη συνάρτησή του mongoSearchSubmit. Στο σημείο αυτό, αρχικά διενεργείται έλεγχος για το κατά πόσο η λέξη που δόθηκε πληροί τις προϋποθέσεις που έχουν οριστεί μέσω της συνάρτησης του CodeIgniter form_validation. Οι προϋποθέσεις αυτές είναι, εκτός του ότι πρέπει να υπάρχει μια λέξη, είναι ότι αυτή η λέξη πρέπει να μην είναι μικρότερη από 3 γράμματα και όχι μεγαλύτερη από 20. Αν στο σημείο αυτό ο έλεγχος αποτύχει, τότε πάλι μέσω της ίδιας συνάρτησης στέλνονται κατάλληλα μηνύματα προς την άποψη mongodbView που ενημερώνουν το χρήστη πως η αποστολή της συγκεκριμένης λέξης προς τη βάση δεδομένων απέτυχε.
Στο σημείο αυτό να διασαφηνιστεί πως η επικοινωνία των ελεγκτών στην εφαρμογή προς τις απόψεις γίνεται με τη βοήθεια αρχείων Javascript. Μάλιστα, χρησιμοποιήθηκε για τη δημιουργία αυτών των αρχείων η βιβλιοθήκη JQuery, όπου είναι σχεδιασμένη με τέτοιο τρόπο, ώστε να διευκολύνει και να απλοποιεί την εκτέλεση εντολών από την πλευρά του πελάτη (client side scripting) και όχι από τον εξυπηρετήτη. Τα δεδομένα που στέλνονται από τον ελεγκτή test προς το αρχείο Javascript js_json μορφοποιούνται βάσει της δομής JSON, ώστε να είναι εύκολα προσπελάσιμα. Στη συνέχεια, το αρχείο js_json είτε σε περίπτωση αποτυχίας είτε σε περίπτωση επιτυχίας, καθορίζει σε ποιο τμήμα (div) της HTML της άποψης θα εμφανίσει συγκεκριμένες επιλογές.

Περίπτωση λάθος αναζήτησης

Παραδείγματος χάρη, σε περίπτωση που χρήστης δώσει λέξη με 2 γράμματα, τότε η form_validation δεν θα περάσει τη λέξη προς το μοντέλο, ώστε να αναζητηθεί στη βάση δεδομένων και θα στείλει προς το js_json αρχείο δύο δεδομένα σε μορφή JSON το μήνυμα πως η λέξη έχει λιγότερους από δύο χαρακτήρες και ένα πρόσθετο πως η επιτυχία του ερωτήματος είναι ψευδής. Αυτό το τελευταίο χρησιμεύει ως ένδειξη (flag), ώστε η μέθοδος success του js_json αρχείου να διαχωρίσει τις ενέργειες που πρέπει να εκτελέσει σε περίπτωση που αυτό είναι αληθές ή ψευδές. Στην προκειμένη περίπτωση είναι ψευδές, οπότε η ενέργεια που θα κάνει είναι να τοποθετήσει το μήνυμα που στέλνεται από τον ελεγκτή στο div errors της άποψης και ο χρήστης να ενημερωθεί πως στη φόρμα αναζήτησης πρέπει να ορίσει λέξη μεγαλύτερη των 3 γραμμάτων.

Περίπτωση σωστής αναζήτησης

Αν ο χρήστης όριζε μια λέξη μεγαλύτερη των 3 και μικρότερη των 20 γραμμάτων, τότε η form_validation, θα επέτρεπε τη συνέχεια των διαδικασιών και η μεταβλητή search που εμπεριέχει τη λέξη προς αναζήτηση θα κατέληγε ως όρισμα της μεθόδου getTweets του

μοντέλου test_model. Οπότε από τον ελεγκτή καλείται η συγκεκριμένη μέθοδος, έχοντας δύο ορίσματα, τη λέξη προς αναζήτηση και τον αριθμό 11, που είναι το όριο των αποτελεσμάτων που θα ζητήσει από τη βάση δεδομένων, προκειμένου η εμφάνιση αργότερα να γίνει σε δεκάδες και να μην εμφανιστούν όλα τα αποτελέσματα με την μια, κάνοντας δύσκολη την ανάγνωσή τους από το χρήστη. Στο προσκήνιο τώρα έρχεται το μοντέλο, όπου μέσω της getTweets στέλνει στη MongoDB τη λέξη μέσω της search, το όριο μέσω της offset και επιλέγει τη βάση, μέσα στην οποία θα γίνει η αναζήτηση (new2). Η βάση θα επιστρέψει τα αποτελέσματα και θα τα τοποθετήσει μέσα στη μεταβλητή query και θα τα επιστρέψει στον ελεγκτή. Ο ελεγκτής με την σειρά του θα τοποθετήσει σε μια δομή JSON, τα αποτελέσματα, την ένδειξη ότι η διαδικασία ήταν επιτυχής και το χρόνο απόκρισης της διαδικασίας και θα τα αποστείλει στο js_json αρχείο, για να τα προωθήσει προς την άποψη. Στη συνέχεια, το js_json, αφού αναγνωρίσει ότι η διαδικασία ήταν επιτυχημένη, θα προχωρήσει στην προώθηση των αποτελεσμάτων προς την άποψη. Πριν κάνει αυτό όμως θα υπολογίσει των αριθμό των αποτελεσμάτων, ώστε αν είναι περισσότερα από 10, να εμφανίσει την πρώτη δεκάδα και στη συνέχεια να εμφανίσει κατάλληλο κουμπί, όπου ο χρήστης θα επιλέγει με το πάτημα του να εμφανίσει την επόμενη δεκάδα, αν είναι λιγότερα από 10 να τα εμφανίσει όλα και να μην εμφανίσει το κουμπί επιλογής και αν δεν υπάρχει κανένα αποτέλεσμα, να εμφανίσει μήνυμα που να ενημερώνει κατάλληλα το χρήστη.
Να σημειωθεί εδώ, πως για κάθε πάτημα του κουμπιού Show more, καλείται η μέθοδος του ελεγκτή handleShowMoreTweets, όπου ξανακαλεί την getTweets με τα ίδια ορίσματα, ώστε να ζητήσει από τη βάση τα επόμενα 10 αποτελέσματα, αν αυτά υπάρχουν. Όλες αυτές οι επιλογές, δρομολογούνται προς το div results της άποψης για εμφάνιση. Επίσης, μαζί με τα πρώτα αποτελέσματα ή την ένδειξη ότι κανένα αποτέλεσμα δεν βρέθηκε, εμφανίζεται και ο χρόνος απόκρισης. Αυτή είναι και όλη η διαδικασία που ακολουθείται από την στιγμή που ο χρήστης δώσει μια λέξη, μέχρις ότου τα tweets που εμπεριέχουν αυτήν την λέξη εμφανιστούν στο χρήστη μαζί με τον χρήστη που το δημοσίευσε, την τοποθεσία που δήλωσε και την ημερομηνία δημοσίευσης.

Διαδικασία εμφάνισης αποτελεσμάτων από τη MongoDB με σύνθετη αναζήτηση

Στην περίπτωση που ο χρήστης επιλέξει τη σύνθετη αναζήτηση, δεν αλλάζουν και πολλά πράγματα στη διαδικασία που θα ακολουθηθεί. Η διαφορά θα είναι πως από την φόρμα επιλογής δεν θα αποσταλεί μόνο η λέξη, αλλά μαζί μ' αυτήν άλλη μια που θα πρέπει να αναζητηθεί στο πεδίο της τοποθεσίας και το εύρος ημερομηνιών που δίνεται μέσα από ειδική εφαρμογή που εμφανίζει ημερολόγιο. Επίσης, η μέθοδος του ελεγκτή που θα αποστείλει όλα τα παραπάνω δεδομένα στο μοντέλο είναι η mongoAdvancedSearchSubmit, η οποία με τη σειρά της καλεί τη μέθοδο του μοντέλου getadvancedTweets και της περνάει ως ορίσματα όλα τα παραπάνω μαζί με το όριο αποτελεσμάτων. Μια διαφορά που υπάρχει μεταξύ της αναζήτησης τοποθεσίας και της λέξης-κλειδί είναι πως για τη λέξη-κλειδί ζητείται αυστηρά η λέξη ως αυτοτελή οντότητα, ενώ για την τοποθεσία δίνεται η δυνατότητα να είναι τμήμα μιας μεγαλύτερης λέξης. Αυτό επιτυγχάνεται μέσω της μεθόδου της PHP MongoRegex και υιοθετήθηκε ως επιλογή, γιατί οι τοποθεσίες πολλές φορές δίνονται από τους χρήστες όχι ως μεμονωμένες λέξεις, αλλά ως κομμάτι μιας μεγαλύτερης λέξης, όπως για παράδειγμα (Austin/Texas). Στη συνέχεια όλη η διαδικασία είναι ακριβώς ίδια, όπως περιγράφηκε στην απλή αναζήτηση.

Επιλογή αναζήτησης μέσω ElasticSearch

Στην πρώτη σελίδα της διεπαφής υπάρχει επίσης η επιλογή που θα φέρει τον χρήστη στην ιστοσελίδα, όπου θα μπορεί να κάνει αναζητήσεις, χρησιμοποιώντας αυτήν την φορά το ElasticSearch ως πηγή. Από τα δύο εικονίδια της MongoDB και του ElasticSearch, ο χρήστης μπορεί να πλοηγείται πότε στη μια και πότε στην άλλη ιστοσελίδα ανάλογα ποιο από τα δύο μέσα επιθυμεί για τις αναζητήσεις του.

Elastica

Η διαδικασία που ακολουθείται, ώστε η λέξη-κλειδί να οδηγηθεί από τη φόρμα αναζήτησης μέχρι το μοντέλο που θα επικοινωνήσει μέχρι τη βάση δεδομένων και τα επιστρεφόμενα αποτελέσματα να εμφανιστούν στο χρήστη, είναι σε γενικές γραμμές παρόμοια. Η προσοχή σ' αυτό το σημείο πρέπει να εστιαστεί στο μοντέλο testEs_model και στον τρόπο, με τον οποίο γίνεται η επικοινωνία μεταξύ της διεπαφής και του ElasticSearch.

Προηγουμένως στη MongoDB η επικοινωνία μεταξύ μοντέλου γινόταν μέσω μιας συγκεκριμένης βιβλιοθήκης, προκειμένου το CodeIgniter να επικοινωνεί με την MongoDB. Το ίδιο ακριβώς χρειάζεται και το ElasticSearch, ώστε να δεχθεί εντολές διαμέσου της PHP. Τον ρόλο αυτό στην προκειμένη περίπτωση παίζει η Elastica, μια βιβλιοθήκη που υλοποιεί της εντολές προς το ElasticSearch, χρησιμοποιώντας PHP. Η Elastica φορτώνεται στο CodeIgniter ως μια οποιαδήποτε εξωτερική βιβλιοθήκη.

Διαδικασία εμφάνισης αποτελεσμάτων από το ES με απλή αναζήτηση

Για την απλή αναζήτηση η μέθοδος getTweets του μοντέλου testEs_model υλοποιείται με την χρήση της Elastica. Αρχικά λοιπόν δημιουργείται ένα αντικείμενο Elastica_Query_Builder, το οποίο στη μεταβλητή query θα φιλοξενεί την εκάστοτε αναζήτηση και το όριο αποτελεσμάτων που πρέπει να επιστρέφεται. Πρέπει να τονιστεί πως η δομή του ερωτήματος πρέπει να δομείται κατά αντιστοιχία προς τη δομή της ανάλυσης. Η χρήση του αντικείμενος Elastica_Query_Builder προτιμήθηκε έναντι των άλλων αντικειμένων της Elastica, γιατί η δομή του είναι παρεμφερή με το ερώτημα, όπως θα δίνονται απευθείας στο ElasticSearch χωρίς τη μεσολάβηση της Elastica. Δηλαδή στην περίπτωση αυτή το ερώτημα θα είχε την παρακάτω μορφή,

```
{
    "query": {
        "query_string": {
            "query": "good",
            "fields": [
               "text.autocomplete"
               ]
        }
    }
}
```

ενώ μέσω της Elastica έχει αυτήν τη μορφή

```
$query = new Elastica_Query_Builder();
        $query
            ->query()
                ->queryString()
                    ->field('query', $search)
                    ->fields(array("text.autocomplete"))
                ->queryStringClose()
            ->queryClose()
            ->from(0)
            ->size($offset);
```

Η αναλογία προς την ανάλυση είναι φανερή, καθώς η μεταβλητή search, όπου διαθέτει τη λέξη που έδωσε ο χρήστης και πρόκειται να περάσει από τον search_ngram_analyzer προσπαθεί να ταιριάξει με το πεδίο text.autocomplete, όπου είναι αποθηκευμένα όλα τα παράγωγα για κάθε

λέξη που δημιούργησε ο index_ngram_analyzer. Επομένως, αν δοθεί η λέξη πχ PLAY, τότε ο search_ngram_analyzer θα την μετατρέψει σε play και θα προσπαθήσει να την ταιριάξει, τόσο με το play όσο και με το players και playing, γιατί και τα δύο εμπεριέχουν αυτό το τμήμα λέξης. Από τη στιγμή που η αναζήτηση επιτύχει και επιστρέψει αποτελέσματα, η διαδικασία είναι ακριβώς ίδια, όπως περιγράφηκε προηγούμενα.

Διαδικασία εμφάνισης αποτελεσμάτων από το ES με σύνθετη αναζήτηση

Αντίστοιχη με τα παραπάνω είναι και η διαδικασία στη σύνθετη αναζήτηση. Η μέθοδος getAdvancedTweetsEs του μοντέλου testEs_model υλοποιείται μέσω της Elastica, ώστε το ερώτημα να δομηθεί με τον ίδιο τρόπο, όπως και στην απευθείας ανάθεση του ερωτήματος. Επομένως, το απευθείας ερώτημα που έχει την παρακάτω μορφή,

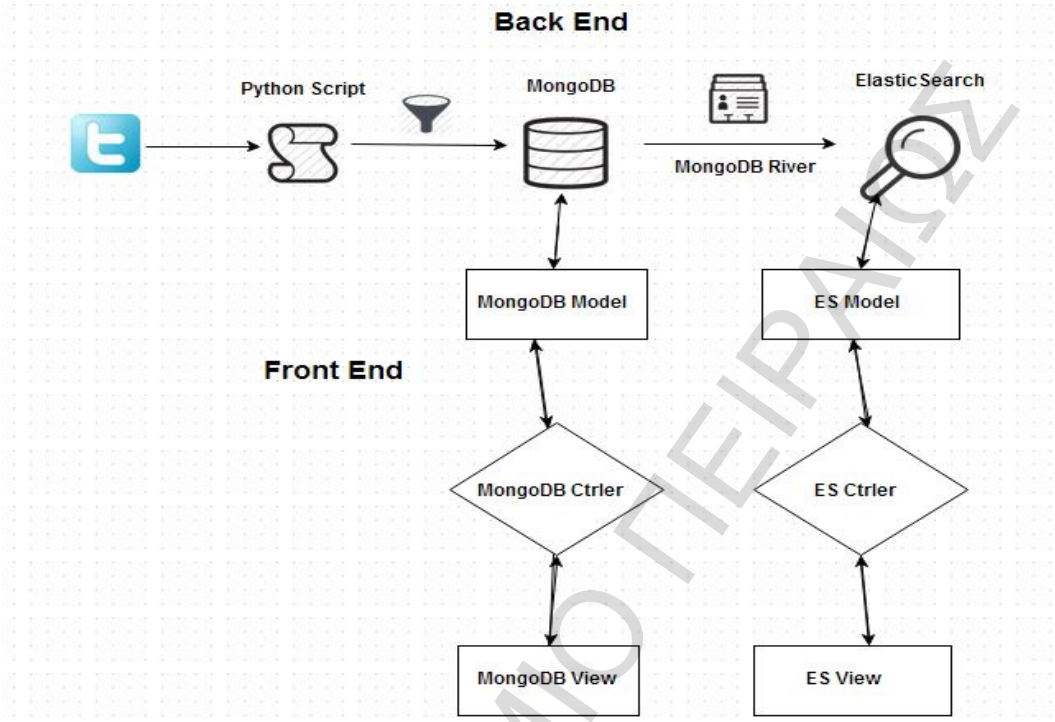
```
{
    "query": {
        "filtered": {
            "query": {
                "query_string": {
                    "query": "play",
                        "fields" : [ "text.autocomplete"
                            ]
                }
            },
            "filter": {
            "and": [
                {
                    "range": {
                        "date": {
                            "from": "07/20/2013",
                            "to": "10/20/2013"
                        }
                    }
                },
                {
                    "term": {
                        "location": "usa"
                    }
                }
            ]
        }
    }
    }
}
```

τροποποιείται μέσω του αντικειμένου Elastica_Query_Builder στην εξής μορφή για κάθε ερώτημα που ορίζεται από τον χρήστη

```
$query = new Elastica_Query_Builder();
        $query
            ->query()
                ->filteredQuery()
                    ->query()
```

```
                ->queryString()
                    ->field("query", $keyword)
                        ->fields(array("text.autocomplete"))
                ->queryStringClose()
            ->queryClose()
            ->filter()
                ->bool()
                    ->should()
                        ->range()
                            ->fieldOpen('date')
                                ->field('from', $datepicker)
                                ->field('to', $datepicker1)
                            ->fieldClose()
                        ->rangeClose()
                    ->shouldClose()
                    ->should()
                        ->term()
                            ->field ("location", $location)
                        ->termClose()
                    ->shouldClose()
                ->boolClose()
            ->filterClose()
        ->filteredQueryClose()
    ->queryClose()
    ->from(0)
    ->size($offset);
```

Εδώ το σημείο που πρέπει να παρατηρηθεί είναι η προσθήκη των φίλτρων σε αντιστοιχία με τα πρόσθετα πεδία της τοποθεσίας και του εύρους ημερομηνιών. Η συνθήκη and του ερωτήματος υλοποιείται μέσω της Elastica με τη μέθοδο bool, όπου συνδυάζει και τα δύο φίλτρα με την προσθήκη της μεθόδου should, όπου με τη σειρά της καθορίζει η ύπαρξη δεδομένων και στα δύο φίλτρα να είναι προαιρετική. Το ίδιο επιτρέπεται και στον έλεγχο που γίνεται στο επίπεδο του ελεγκτή, καθώς μόνο η λέξη-κλειδί είναι υποχρεωτική, προκειμένου να προχωρήσει η αναζήτηση στο επόμενο επίπεδο. Στη συνέχεια και αφού επιστραφούν αποτελέσματα, η διαδικασία είναι ίδια, όπως περιγράφηκε παραπάνω.
Ολόκληρη η αρχιτεκτονική της εφαρμογής, όπως περιγράφηκε μπορεί να αναπαρασταθεί από το παρακάτω σχεδιάγραμμα.



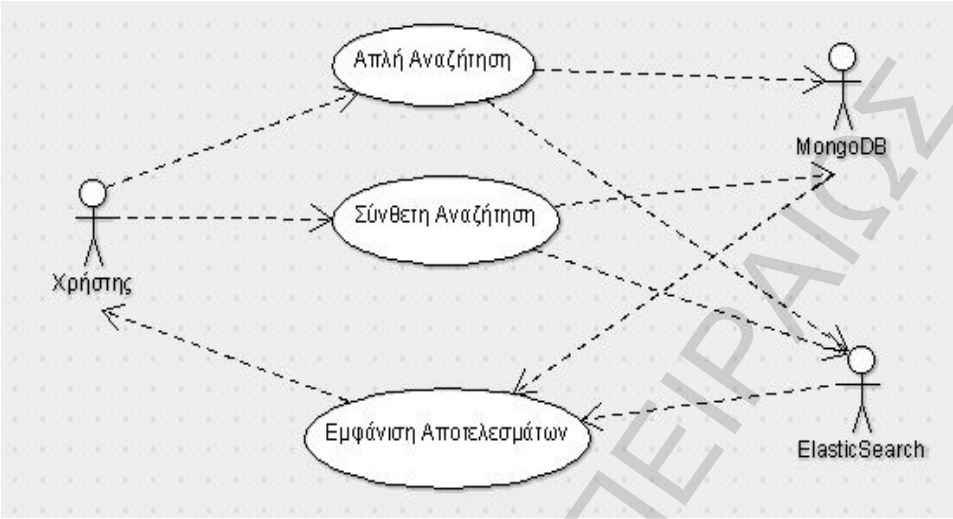
Εικ. 4 Αρχιτεκτονική εφαρμογής

4.6 Διαγράμματα Ενοποιημένης Γλώσσας Σχεδιασμού (UML)

Η ενοποιημένη γλώσσα σχεδιασμού (unified modeling language - UML) είναι μια γραφική γλώσσα για την οπτική παράσταση, τη διαμόρφωση προδιαγραφών και την τεκμηρίωση συστημάτων που βασίζονται σε λογισμικό. Σκοπός της είναι ο σχεδιασμός αντικειμενοστραφών συστημάτων που μέσω αυτών δίνεται μια απλοποιημένη παράσταση της εφαρμογής. Η σχεδίαση των διαγραμμάτων βοηθάει στην οπτική παράσταση του εκάστοτε συστήματος, στον προσδιορισμό της δομής και της συμπεριφοράς του, στη δημιουργία ενός προτύπου, πάνω στο οποίο θα κατασκευαστεί το σύστημα και στην τεκμηρίωση των διαφόρων επιλόγων που έγιναν κατά της ανάπτυξη του συστήματος.

Διάγραμμα περιπτώσεων χρήσης

Η UML είναι μια πλήρης και πλούσια γλώσσα με εξαιρετικά ευρύ πεδίο εφαρμογής και διαθέτει 9 διαφορετικούς τύπους διαγραμμάτων, μέσω των οποίων δίνεται η δυνατότητα να σχεδιαστούν τα επιμέρους τμήματα κάποιου συστήματος. Ο πρώτος τύπος που θα παρουσιαστεί είναι το διάγραμμα περιπτώσεων χρήσης (use case diagram). Τα διαγράμματα αυτά αποτελούνται από τις περιπτώσεις χρήσης, τους δρώντες (actors) που βρίσκονται έξω από το σύστημα, τις σχέσεις εξάρτησης, γενίκευσης και σύνδεσης και τα όρια του συστήματος. Στην περίπτωση της εφαρμογής το διάγραμμα περιπτώσεων χρήσης είναι το παρακάτω.

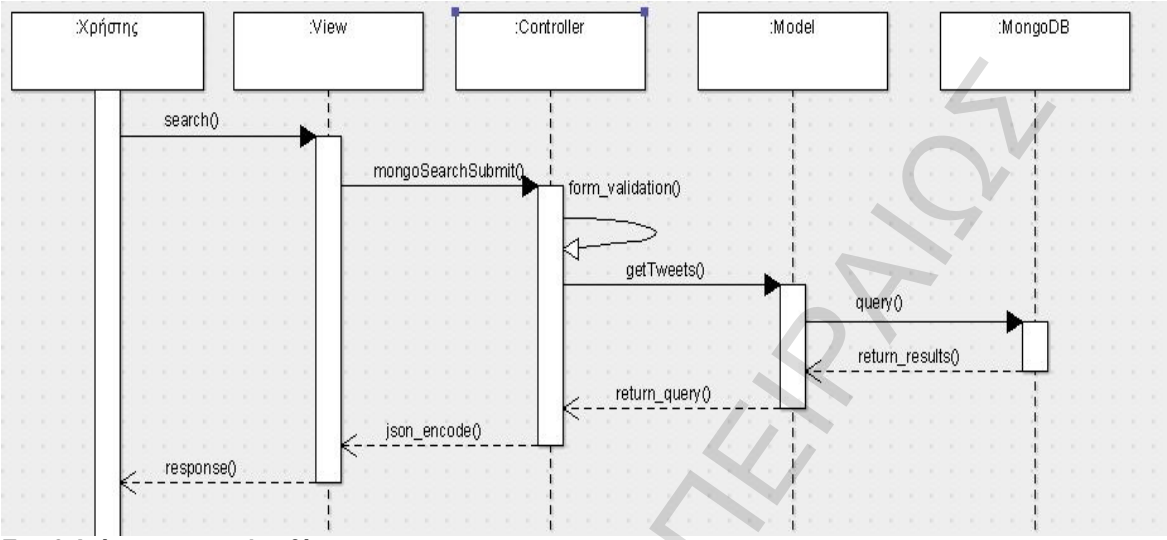


Εικ. 5 Διάγραμμα περιπτώσεων χρήσης

Όπως φαίνεται και από το ίδιο, το διάγραμμα είναι αρκετά απλό, γιατί δεν υπάρχει διαδικασία εισόδου στο σύστημα που να απευθύνεται σε διαφορετικά επίπεδα χρηστών. Ο ρόλος του χρήστη είναι μονοδιάστατος και έχει μόνο δύο επιλογές. Η πρώτη είναι να αποστείλει ένα απλό ερώτημα αναζήτησης προς τους άλλους δύο χρήστες που είναι εκτός συστήματος και είναι οι βάσεις δεδομένων και η δεύτερη να αποστείλει ένα σύνθετο ερώτημα προς αυτές. Με τη σειρά τους οι βάσεις δεδομένων διοχετεύουν κατάλληλα την απόκρισή τους στις αναζητήσεις μέσω του συστήματος. Το τελευταίο είναι υπεύθυνο να εμφανίσει τα αποτελέσματα των αναζητήσεών στο χρήστη. Επομένως, οι 3 δρώντες χρησιμοποιούν το σύστημα, ώστε να ανταλλάσσουν ερωτήματα αναζητήσεων και αποτελέσματα, που εμφανίζει τελικά το σύστημα στονχρήστη.

Διάγραμμα ακολουθίας

Το δεύτερο διάγραμμα που θα βοηθήσει στην κατανόηση της εφαρμογής είναι το διάγραμμα ακολουθίας (sequence diagram). Πρόκειται για διάγραμμα αλληλεπίδρασης (συμπεριφοράς) που παρουσιάζει τον τρόπο που διαφορετικά αντικείμενα συνεργάζονται μεταξύ τους σε μια χρονική ακολουθία. Διαθέτει αντικείμενα, σχέσεις μεταξύ αντικειμένων, μηνύματα, τη διάρκεια της ζωής κάθε αντικειμένου και την περιοχή ελέγχου για κάθε αντικείμενο. Επίσης, το διάγραμμα ακολουθίας διαθέτει δύο διαστάσεις. Η κάθετη διάσταση δείχνει την ακολουθία των μηνυμάτων/κλήσεων βάσει του χρόνου που λαμβάνουν χώρα, ενώ η οριζόντια διάσταση απεικονίζει τα στιγμιότυπα των αντικειμένων, στα οποία στέλνονται τα μηνύματα. Η ανάγνωση ενός τέτοιου διαγράμματος είναι πολύ απλή. Από την αριστερή πλευρά φαίνεται εκείνο το αντικείμενο, από το οποίο ξεκινάει η όλη ακολουθία και προχωρώντας προς τα δεξιά φανερώνεται η αλληλουχία των μηνυμάτων που στέλνονται από τα αντικείμενα, καθώς και οι αποκρίσεις τους σ' αυτά. Παρακάτω παρουσιάζεται το διάγραμμα ακολουθίας που απεικονίζει την αλληλουχία των διαδικασιών για μια απλή αναζήτηση.

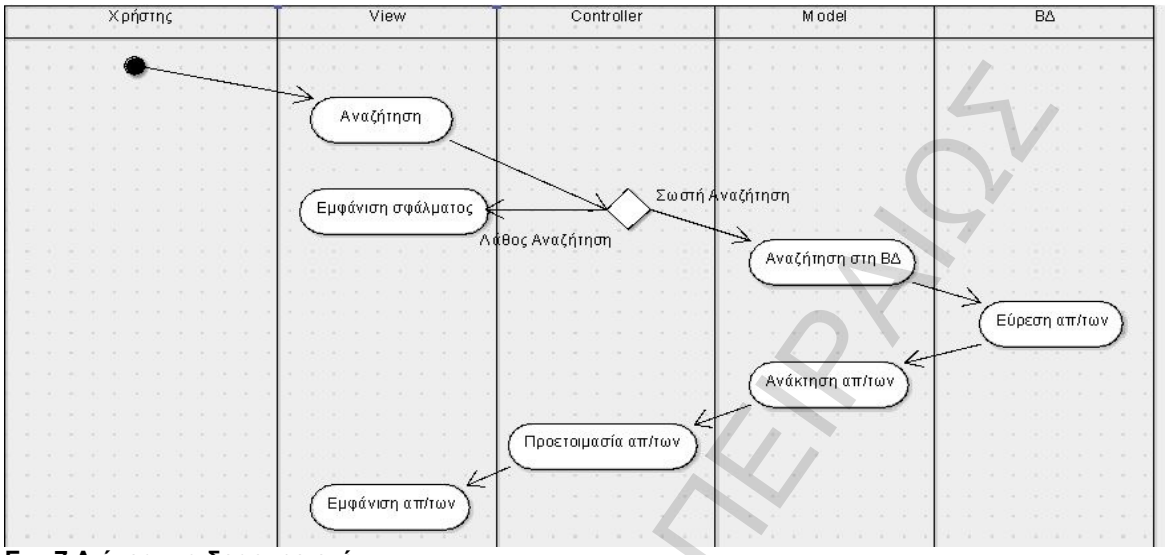


Εικ. 6 Διάγραμμα ακολουθίας

Με σκοπό την καλύτερη κατανόηση της λειτουργίας της εφαρμογής αναλύθηκαν ως ξεχωριστά αντικείμενα και τα τρία μέρη της αρχιτεκτονικής MVC. Οπότε αρχικά ο χρήστης αναζητεί μια λέξη μέσω της φόρμας που υπάρχει στην άποψη, αυτή η λέξη κλειδί περνάει στον ελεγκτή και αφού πρώτα ελεγχθεί μέσω της form_validation() ότι πληροί τις προϋποθέσεις αποστέλλεται στο μοντέλο που είναι υπεύθυνο να επικοινωνήσει με τη βάση δεδομένων. Αν η αναζήτηση επιστρέψει αποτελέσματα, τότε αυτά αποστέλλονται στο μοντέλο, το οποίο και θα τα προωθήσει στον ελεγκτή, ώστε με τη σειρά του θα τα ενοποιήσει μαζί με το χρόνο απόκρισης και την ένδειξη ότι η αναζήτηση ήταν επιτυχής σε μια δομή JSON και θα τα περάσει στην άποψη και πάλι, ώστε με τη βοήθεια της Javascript και της HTML να εμφανιστούν στο χρήστη.

Διάγραμμα δραστηριοτήτων

Το τρίτο διάγραμμα που παρουσιάζεται παρακάτω είναι το διάγραμμα δραστηριοτήτων (activity diagram) που απεικονίζει την αλληλουχία των δραστηριοτήτων (workflow) σε μια διεργασία. Σ' αυτά τα διαγράμματα υπάρχουν οι διαδρομές (swimlanes), όπου εκεί εμφανίζονται οι δραστηριότητες που αφορούν ξεχωριστά το κάθε αντικείμενο και οι διακλαδώσεις υπό συνθήκη (branches), όπου απεικονίζονται οι αποφάσεις που λαμβάνονται από κάποια αντικείμενα, ενώ η πορεία των δραστηριοτήτων βρίσκεται σε εξέλιξη. Το διάγραμμα είναι το εξής.



Εικ. 7 Διάγραμμα δραστηριοτήτων

Όπως και στο διάγραμμα ακολουθίας, έτσι και σ' αυτό το διάγραμμα το κάθε τμήμα της αρχιτεκτονικής MVC παρουσιάζεται ως ξεχωριστό αντικείμενο, έτσι ώστε να εμφανίζεται καλύτερα η συμμετοχή του καθενός μέρους στη λειτουργία της εφαρμογής. Το διάγραμμα αυτό ισχύει και για τους δύο τρόπους αναζήτησης που προσφέρονται, εφόσον οι διαδικασίες που ακολουθούνται όσον αφορά την αναζήτηση και την εμφάνιση των αποτελεσμάτων είναι κοινές. Αρχικά ο χρήστης αναζητεί μια λέξη, η οποία μέσω της άποψης περνά στον ελεγκτή. Στο σημείο αυτό ελέγχεται η ορθότητά της και αν αποτύχει, τότε εμφανίζεται μήνυμα λάθους στην άποψη, αλλιώς η αναζήτηση μέσω του μοντέλου τίθεται στη βάση δεδομένων που αποστέλλει τα αποτελέσματα μέσω πάλι του μοντέλου στον ελεγκτή. Ο ελεγκτής δομεί κατάλληλα τα αποτελέσματα και τα εμφανίζει στο χρήστη μέσω της άποψης.

Κεφάλαιο 5 - Συμπεράσματα και μελλοντικές επεκτάσεις της εφαρμογής

5.1 Εισαγωγή

Σ' αυτό το τελευταίο κεφάλαιο θα γίνει προσπάθεια να παρουσιαστούν γενικά συμπεράσματα που συνήχθησαν από όλη τη διαδικασία δημιουργίας της εφαρμογής και από τις λειτουργικές της δυνατότητες, όπως επίσης και συνοπτικά θα εκφραστούν οι νέες γνώσεις που προσέφερε η ενασχόληση με όλα τα αντικείμενα που συνθέτουν την παρούσα διπλωματική εργασία. Στο τελευταίο τμήμα αυτού του κεφαλαίου θα γίνει λόγος για μελλοντικές επεκτάσεις που μπορεί να έχει η εφαρμογή.

5.2 Συμπεράσματα

Γνώσεις που αποκτήθηκαν

Αρχικά, πριν γίνει αναφορά στα διάφορα συμπεράσματα, ας παρουσιαστούν οι ποικίλες νέες γνώσεις που αποκομίστηκαν με την ενασχόληση με την παρούσα διπλωματική. Οι διαφορετικοί τομείς που εμπλέκονται είναι αρκετοί και έτσι οι γνώσεις είναι ποικίλες, ξεκινώντας από τον τρόπο που γίνεται η διαχείριση ενός API μέσω μιας αντικειμενοστραφούς γλώσσας προγραμματισμού, με σκοπό τη συλλογή συγκεκριμένων δεδομένων και το κατάλληλο φιλτράρισμά τους. Σημαντικές επίσης ήταν οι γνώσεις που αποκτήθηκαν από την ενασχόληση με μια κατηγορία βάσεων δεδομένων διαφορετικών από τις μέχρι τώρα γνωστές σχεσιακές βάσεις. Οι NoSQL βάσεις δεδομένων έχουν εντελώς διαφορετική φιλοσοφία και εξαιτίας των διευκολύνσεων που παρέχουν, ιδιαίτερα στον τομέα της διαχείρισης μεγάλων συλλογών δεδομένων (big data) είναι σίγουρο πως τα επόμενα χρόνια θα κυριαρχήσουν στον τομέα αυτό. Συνάμα ήταν αρκετά χρήσιμη η κατανόηση του τρόπου λειτουργίας ενός εξυπηρετητή αναζήτησης (search server), όπως το ElasticSearch, καθώς και η δημιουργία μέσω αυτού μιας μηχανής αναζήτησης που λειτουργεί σε πραγματικό χρόνο.

Πέρα από την κατανόηση της λειτουργίας των υποδομών αυτών καθ' αυτών σημαντικός είναι και ο τρόπος που όλα αυτά μαζί συνδέονται υπό το πρίσμα κατασκευής μιας διαδικτυακής εφαρμογής. Μάλιστα αυτό έγινε ακόμα περισσότερο ενδιαφέρον από το γεγονός ότι η βάση δημιουργίας της ιστοσελίδας ήταν ένα συγκεκριμένο πλαίσιο λογισμικού PHP, όπως το CodeIgniter. Η χρησιμοποίησή του έδωσε την ευκαιρία να κατανοηθούν σε βάθος και μερικά άλλα γνωστικά πεδία, όπως είναι η χρήση του αντικειμενοστραφούς προγραμματισμού για την κατασκευή μιας διαδικτυακής εφαρμογής, αλλά και της αρχιτεκτονικής Μοντέλου – Άποψης – Ελεγκτή, όπου πλέον μια εφαρμογή σταμάτησε να φαίνεται ως μια ενιαία οντότητα, αλλά ως μια συλλογή από διαφορετικά μεταξύ τους αρχεία χωρισμένα σε ομάδες, που η καθεμιά ομάδα από αυτές έχει να επιτελέσει ξεχωριστό έργο στη λειτουργία της εφαρμογής. Επίσης, σημαντική γνώση υπήρξε και η κατανόηση πως για την εμφάνιση των δεδομένων σε μια διαδικτυακή εφαρμογή που φιλοξενείται σε μια ιστοσελίδα παίζουν ρόλο και άλλες τεχνολογίες που βοηθούν τόσο στον τρόπο εμφάνισης, όπως το Twitter Bootstrap, όσο και στη διαχείριση των

ανακτώμενων δεδομένων και στον καθορισμό του τρόπου παρουσίασής τους στο χρήστη, όπως όλα αυτά πραγματοποιούνται με τη βοήθεια της βιβλιοθήκης Javascript, JQuery.
Τέλος, ίσως το πιο χρήσιμο από όλα είναι ο τρόπος συνεργασίας όλων αυτών των φαινομενικά ανεξάρτητων τεχνολογιών και για περισσότερη σαφήνεια, ας αναφερθούν ορισμένα παραδείγματα. Αρχικά ο τρόπος, με τον όποιο μέσω δύο βιβλιοθηκών της Python, Tweepy και Pymongo, τα κειμενικά δεδομένα του Twitter αποθηκεύονται με ορισμένο τρόπο στη MongoDB. Η χρήση του αρθρώματος MongoDB River, με τη βοήθεια του οποίου τα αποθηκευμένα δεδομένα στη NoSQL βάση δεδομένων περνούν μέσα σ' ένα προκαθορισμένο ευρετήριο του ElasticSearch. Η χρήση της βιβλιοθήκης Elastica, απ' όπου γίνεται η διαχείριση του ευρετηρίου του ElasticSearch μέσω ενός αρχείου PHP που αποτελεί τμήμα του Μοντέλου του CodeIginiter. Και αυτά είναι μόνο τα πιο σημαντικά από τα γεφυρώματα που έπρεπε να γίνουν, ώστε στο τέλος ο χρήστης για κάθε λέξη-κλειδί που αναζητεί, να εμφανίζονται τα σωστά αποτελέσματα.

Βιβλιοθήκες για συνεργασία MongoDB - CodeIgniter

Στο σημείο αυτό θα αναφερθούν τα πιο σημαντικά συμπεράσματα, τα οποία κυρίως σχετίζονται με την υποδομή του πίσω μέρους (backend) της εφαρμογής. Αρχικά λοιπόν η MongoDB, παρόλο που είναι ιδανική και για λόγους ασφάλειας και διαθεσιμότητας των δεδομένων, αλλά παράλληλα διαθέτει ολοκληρωμένο σύστημα ευρετηρίασης κειμενικών δεδομένων (full text indexing), δεν υποστηρίζεται από κάποια ολοκληρωμένη βιβλιοθήκη που να συνεργάζεται πλήρως με το CodeIgniter. Η μόνη βιβλιοθήκη που βρέθηκε (Bilbie 2013) υποστηρίζει μόνο τις βασικές λειτουργίες (CRUD). Ο μόνος τρόπος ευρετηρίασης θα έπρεπε να γίνεται απευθείας από τη γραμμή εντολών της MongoDB είτε από τη γραμμή εντολών είτε μέσω κάποιου αρχείου σεναρίου εντολών, αλλά και πάλι θα υπήρχε δυσκολία στην ανάκτηση των αποτελεσμάτων, αφού η συγκεκριμένη βιβλιοθήκη χρησιμοποιεί μόνο κανονικές εκφράσεις, για να ανακτήσει αποτελέσματα μέσω της μεθόδου MongoRegex της PHP.

ElasticSearch ως κύρια βάση δεδομένων

Στην αντίπερα όχθη, και το ElasticSearch δεν ενδείκνυται να λειτουργεί ως μοναδικός τρόπος ευρετηρίασης και αποθήκευσης δεδομένων, παρόλο που διαθέτει άρτιο, ολοκληρωμένο και πολύ σύγχρονο τρόπο δημιουργίας ευρετηρίου μέσω της σχετικής βιβλιοθήκης (Elastica) που το καθιστά προσβάσιμο από την PHP και κατ' επέκταση το CodeIgniter, αλλά και εγγενές άρθρωμα (Twitter River), που δίνει τη δυνατότητα τα κειμενικά δεδομένα να ευρετηριάζονται και να αποθηκεύονται μέσα σε ευρετήριο του ElasticSearch απευθείας από το Twitter API χωρίς τη μεσολάβηση καμίας άλλης τεχνολογίας. Όμως το γεγονός ότι δεν προσφέρει κάποιο σύστημα ασφάλειας για τα αποθηκευμένα δεδομένα, διαθεσιμότητα των δεδομένων στο ίδιο επίπεδο, όπως η MongoDB και εργαλεία αρκετά εύρωστα εξαιτίας του ότι είναι πολύ νέα ως τεχνολογία, οδηγούν στο συμπέρασμα πως μια συνεργασία των δύο τεχνολογιών αποτελεί την καλύτερη λύση. Εξάλλου, η χρήση του MongoDB River διευκολύνει το πέρασμα δεδομένων από τη βάση προς το ευρετήριο του ElasticSearch. Επίσης, η ύπαρξη ενός προηγούμενου επιπέδου αποθήκευσης σε βάση NoSQL εξασφαλίζει πως τα δεδομένα θα βρίσκονται πάντα σε JSON μορφή, γεγονός που κάνει πολύ εύκολη τη μετάβασή τους στο ElasticSearch.

Χρόνος απόκρισης MongoDB - ElasticSearch

Στην εφαρμογή που αναπτύχθηκε για τις ανάγκες της παρούσας διπλωματικής εργασίας ο χρήστης μπορεί να έχει πρόσβαση στα δεδομένα που είναι αποθηκευμένα και στις δύο τεχνολογίες που χρησιμοποιούνται στο πίσω μέρος της εφαρμογής. Η διάφορα είναι ότι στα αποτελέσματα που προέρχονται από τη MongoDB δεν έχει εφαρμοστεί κάποιο συγκεκριμένο ευρετήριο, ενώ σ' αυτά που εμφανίζονται μέσω του ElasticSearch τα ευρετήρια που παρουσιάστηκαν στα προηγούμενα κεφάλαια. Για να μπορέσει να γίνει σύγκριση στο χρόνο που χρειάζονται οι δύο τεχνολογίες, ώστε να επιστρέψουν αποτελέσματα σε μια συγκεκριμένη ερώτηση χρησιμοποιήθηκε η μέθοδος benchmark του CodeIgniter. Η μέθοδος αυτή

εφαρμόζεται μέσα στις συναρτήσεις mongoSearchSubmit και esSearchSubmit των δύο Ελεγκτών και υπολογίζει το χρόνο που κάνει μια λέξη-κλειδί να περάσει από τον Ελεγκτή προς το Μοντέλο και την επιστροφή των αποτελεσμάτων από το Μοντέλο προς τον Ελεγκτή. Παίρνοντας ως παράδειγμα λέξης-κλειδιού τη λέξη “football”, η MongoDB επιστρέφει τα πρώτα 10 αποτελέσματα σε 0,09 sec, ενώ το ElasticSearch σε 0,1 sec. Γενικά όσες αναζητήσεις και αν διενεργήθηκαν πάντα το ElasticSearch ήταν λίγο πιο αργό σε σχέση με τη MongoDB, αλλά δεν μπορούν να συναχθούν ασφαλή συμπεράσματα για την ταχύτητα απόκρισης των δύο τεχνολογιών, καθώς και οι δύο επικοινωνούν με το CodeIgniter μέσω βιβλιοθηκών που σαφώς αλλοιώνουν τους χρόνους πραγματικής απόκρισης.

Διαδικασίες αναζήτησης MongoDB - ElasticSearch

Γενικότερα, ο τρόπος λειτουργίας των δύο τεχνολογιών είναι πολύ διαφορετικός και εξυπηρετεί διαφορετικές ανάγκες κάθε φορά. Παραδείγματος χάρη, το ElasticSearch ως εξέλιξη του Lucene χρησιμοποιεί το μοντέλο χώρου διανυσμάτων (vector space model) και αντεστραμμένα ευρετήρια (inverted indexes) για εξόρυξη πληροφορίας, που είναι επαρκέστατοι τρόποι, για να συγκριθούν ομοιότητες εγγραφών προς ένα ερώτημα. Στην ουσία όταν ένα ερώτημα στέλνεται στο ElasticSearch, η απάντηση είναι ήδη γνωστή, καθώς το μεγαλύτερο κομμάτι της εργασίας του στηρίζεται στην κατάταξη των αποτελεσμάτων, ξεκινώντας από εκείνα που ταιριάζουν περισσότερο στο κατατεθέν ερώτημα. Αυτό είναι και ένα βασικό σημείο, οι μηχανές αναζήτησης σε αντίθεση με τις πραγματικές βάσεις δεδομένων δεν εγγυώνται ακριβή αποτελέσματα, αλλά βαθμολογούν αποτελέσματα ανάλογα με τη συνάφεια που έχουν προς το ερώτημα.

Από την άλλη πλευρά, η προσέγγιση της MongoDB ταιριάζει περισσότερα με τις διαδικασίες αναζήτησης των NoSQL βάσεων δεδομένων, δηλαδή προσπαθεί να συγκρίνει δύο έγγραφα JSON μεταξύ τους. Αυτή η διαδικασία σίγουρα προσφέρει πολύ καλές επιδόσεις, αλλά χρειάζεται τα ευρετήρια να δημιουργούνται με μεγάλη προσοχή, προκειμένου να επιστραφούν σωστά αποτελέσματα. Στην προκειμένη περίπτωση της παρούσας εφαρμογής δεν χρησιμοποιείται κανενός είδους ευρετήριο για τη MongoDB, καθώς αυτή η εργασία ανατίθεται στο ElasticSearch.

5.3 Μελλοντικές επεκτάσεις

Η παρούσα εφαρμογή, παρόλο που μπορεί να σταθεί και ως μια αυτόνομη εφαρμογή, καθώς έχει τη δυνατότητα επιστροφής αποτελεσμάτων σε συγκεκριμένες αναζητήσεις από χρήστες, έχει πολλές επεκτάσεις που μπορούν να γίνουν, καθώς με τη συγκεκριμένη της μορφή μοιάζει περισσότερο ως ένα κομμάτι που μπορεί να προστεθεί σε μια εφαρμογή με μεγαλύτερες δυνατότητες.

Θεματική αποθήκευση δεδομένων

Συγκεκριμένα λοιπόν όσον αφορά την υποδομή της, στην τωρινή μορφή της ο χρήστης δεν μπορεί να έχει πρόσβαση στη βάση δεδομένων. Η βάση γεμίζει μέσω ενός αρχείου σεναρίου εντολών από το Twitter API με μόνο φιλτράρισμα να γίνεται στη γλώσσα και να επιτρέπει μόνο τα tweets με δηλωμένη την αγγλική γλώσσα να αποθηκεύονται. Σε μια μελλοντική επέκταση ο χρήστης θα μπορεί να έχει τη δυνατότητα δυναμικά να επιλέγει ποιες συγκεκριμένες λέξεις-κλειδιά θα περιέχουν τα tweets που θα αποθηκεύονται στη βάση δεδομένων και έτσι θα μπορεί να φτιάχνει συλλογές από tweets βάσει προκαθορισμένων επιλογών. Αυτό θα του δίνει τη δυνατότητα να εκτελεί αναζητήσεις μέσα σε tweets με συγκεκριμένη θεματολογία. Παραδείγματος χάρη, αν έχει αποθηκεύσει tweets με τη λέξη-κλειδί αυτοκίνητο, αναζητώντας τη

λέξη φθηνό, θα μπορέσει να δει αμέσως ποια αυτοκίνητα θεωρούν οι χρήστες πως είναι πιο οικονομικά για αγορά.

Δεδομένα από διαφορετικές πηγές

Ακόμα, όσον αφορά τις υποδομές αποθήκευσης δεδομένων, μπορεί να υπάρξει η δυνατότητα να μην εισρέουν δεδομένα μόνο από το Twitter, αλλά να συμπληρωθεί και το Facebook, καθώς και αυτό διαθέτει δικό του API που μάλιστα δομεί τα δεδομένα σε μορφή JSON.

Τοπική αποθήκευση δεδομένων

Επίσης, μπορεί να ενταχθεί μια επιπλέον λειτουργία σε επίπεδο λειτουργίας της ιστοσελίδας, ώστε τα αποτελέσματα να υπάρχει η δυνατότητα να αποθηκευτούν στον υπολογιστή του χρήστη ως ένα αρχείο csv, για παράδειγμα, και ο χρήστης να έχει τη δυνατότητα για περαιτέρω επεξεργασία.

Περισσότερα πεδία αναζητήσεων

Παράλληλα, η σύνθετη αναζήτηση μπορεί να αποκτήσει περισσότερα πεδία, όπως παραδείγματος χάρη αυτό του χρήστη, ώστε να δίνεται η δυνατότητα να αναζητούνται tweets από συγκεκριμένους χρήστες.

Γραφική και ποσοτική απεικόνιση δεδομένων

Ακόμα, άλλες δυνατότητες που μπορούν να ενταχθούν και υπερβαίνουν τα αντικείμενα που αναλύθηκαν στη συγκεκριμένη διπλωματική είναι η ένταξη γραφικών παραστάσεων που θα αντλούνται από τα αποτελέσματα που επιστρέφονται. Παραδείγματος χάρη, σύννεφα λέξεων (word clouds) που θα δείχνουν την συχνότητα των λέξεων που απαντώνται στα αποτελέσματα ή κατανομή των αποτελεσμάτων μέσα στον χρόνο. Ακόμα, μπορεί να υπάρχει και ειδικό γράφημα που να δείχνει για παράδειγμα ποιοι ήταν οι πιο ενεργοί χρήστες σε συγκεκριμένα θέματα, ώστε έτσι να αναπαριστάται γραφικά ποιοι από αυτούς έχουν μεγαλύτερη επιρροή στα θέματα αυτά.

Ποιοτική απεικόνιση δεδομένων

Επιπλέον και κλείνοντας υπάρχει ακόμα και η δυνατότητα, εφαρμόζοντας τεχνικές μηχανικής μάθησης και ανάλυσης συναισθήματος, για το κάθε tweet που εμφανίζεται να υπάρχει και πρόσημο για το αν έχει θετικό ή αρνητικό σημασιολογικό πρόσημο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΒΙΒΛΙΑ

Banker, K, 2012, *MongoDB in Action*, Manning Publications, New York

Beazley, MD, 2006, *Python Essential Reference*, 3rd edn, Sams Publishing, Indianapolis

Cholakian, A, 2013, Exploring ElasticSearch, Available from: <http://exploringelasticsearch.com/>

Hatcher, E, Gospodnetić, O & McCandless M, 2009, *Lucene in Action*, 2nd edn, Manning Publications, New York

Kuć, R & Rogoziński, M, 2013, *ElasticSearch Server,* Packt Publishing, Birmingham

Russell, AM, 2011, *21 Recipes for Mining Twitter,* O'Reilly, Sebastopol

Schmuller, J, 2004, Sams *Teach Yourself UML in 24 Hours*, 3rd edn, Sams Publishing, Indianapolis

Vaish, G, 2013, *Getting Started with NoSQL,* Packt Publishing, Birmingham

ΙΣΤΟΣΕΛΙΔΕΣ & ΙΣΤΟΛΟΓΙΑ

Abrahamsson, J, *ElasticSearch 101 – a getting started tutorial,* 2013. Available from: <http://joelabrahamsson.com/elasticsearch-101/> [2 July 2013]
Bell, D, *UML basics: An introduction to the Unified Modeling Language*, 2003. Available from: < http://www.ibm.com/developerworks/rational/library/769.html/> [15 June 2003]
Bilbie, A, *CodeIgniter MongoDB Library*, 2013. Available from: < https://github.com/alexbilbie/codeigniter-mongodb-library>
Brian, K, *Using MongoDB and CodeIgniter On Windows*, 2011. Available from: <http http://bhbrayeun.blogspot.gr/2011/03/using-mongodb-and-codeigniter-on.html> [27 March 2011]

Edvartsen, C, *Using Elastica to query ElasticSearch*, 2012. Available from: <http://tech.vg.no/2012/07/03/using-elastica-to-query-elasticsearch/> [3 July 2012]
Gandham, S, *A complete guide to Integrating MongoDB with Elastic Search*, 2012. Available from: <http://satishgandham.com/2012/09/a-complete-guide-to-integrating-mongodb-with-elastic-search/> [3 September 2012]
O'Reilly, T, *What Is Web 2.0*, 2005. Available from: < http://oreilly.com/web2/archive/what-is-web-20.html> [30 September 2005]
Tong, Z, *Constructing more complicated mapping in ElasticSearch*, 2012. Available from: <http://euphonious-intuition.com/2012/08/more-complicated-mapping-in-elasticsearch/> [8 January 2012]
Willie, R, *MongoDB River Plugin for ElasticSearch*, 2013. Available from: < https://github.com/richardwilly98/elasticsearch-river-mongodb>
Σπινέλλης, Δ, *Ανάλυση και σχεδίαση με UML*. Available from: < http://www.dmst.aueb.gr/dds/ism/oo/indexw.htm/>
CodeIgniter User Guide, 2013. Available from: < http://ellislab.com/codeigniter/user-guide/>
CodeIgniter Bootstrap, 2013. Available from: < https://github.com/sjlu/CodeIgniter-Bootstrap>
Elasticsearch: Open Source Distributed Real Time Search & Analytics, 2013. Available from: < http://www.elasticsearch.org/>
JQuery API Documentation, 2013. Available from: < http://api.jquery.com/>
How to Capture Tweets in Real-time with Twitter's Streaming API, 2011. Available from: <http://answers.oreilly.com/topic/2605-how-to-capture-tweets-in-real-time-with-twitters-streaming-api/> [6 April 2011]
PHP Documentation, 2013. Available from: < http://php.net/docs.php/>
Python Documentation, 2013. Available from: <http://www.python.org/doc/>
Replication, 2013. Available from: < http://docs.mongodb.org/manual/replication/>
Storing and indexing documents, 2012. Available from: <http://elastica.io/getting-started/storing-and-indexing-documents.html/>
what is elasticsearch?, 2013. Available from: <http://www.elasticsearch.org/overview/>

ΑΡΘΡΑ

Berners, Tim , 2011, 'Long Live the Web: A Call for Continued Open Standards and Neutrality', *Scientific American*, 24 April 2012
Grosvenor, D, Kendall, J & Sanders, A, 2012, *Data Analytics Course Project Checkpoint 2*
DiNuzzi, D, 1999, 'Fragmented Future', *Print magazine* April 1999, p.32
Huurnink B et al, 2013, *AVResearcher: Exploring Audiovisual Metadata*
Kaplan Andreas M., Haenlein Michael ,2010, 'Users of the world, unite! The challenges and opportunities of social media'. *Business Horizons 53 (1),* p. 61
Rooij, OD, Odijk, D & Rijke, DM, 2013, *ThemeStreams: Visualizing the Stream of Themes Discussed in Politics*

Thompson, J, Hankinson, A & Fujinaga, I, 2012, *SEARCHING THE LIBER USUALIS: USING COUCHDB AND ELASTICSEARCH TO QUERY GRAPHICAL MUSIC DOCUMENTS*

ΠΑΡΑΡΤΗΜΑ

test.py - Αρχείο σεναρίου εντολών (script) που αποθηκεύει τα tweets μέσω του Twitter API σε μια MongoDB βάση δεδομένων.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys, tweepy, pymongo
import dateutil.parser as parser

# Query terms

# Q = sys.argv[1:]

connection = pymongo.Connection("localhost", 27017)
db = connection.new2

CONSUMER_KEY = 'b6l9C7FvZEHJa9ZUjUfWmQ'
CONSUMER_SECRET = 'ywMFNXQgw7IEK7iSFm5bUprhpGxfB6kqs0X9aS55iLk'

ACCESS_TOKEN = '328717381-zZqz8cx32H1uNZPhIT5OrfEgLDGae1CxBCzI49sO'
ACCESS_TOKEN_SECRET = 'ifmNPD41bk9xl7vQPHUN4g08lZrLgyttVkLXrn9WMM'

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)

class CustomStreamListener(tweepy.StreamListener):

    def on_status(self, status):
        try:
        if (status.lang == 'en'):
                date = status.created_at
                db.new2.save({'text' : status.text.encode('utf-8'),
                              'user' : status.author.name.encode('utf-
8'),
                              'nickname' :
status.author.screen_name.encode('utf-8'),
                              'location' :
status.author.location.encode('utf-8'),
                              'followers' :
status.author.followers_count,
                              'followings' :
status.author.friends_count,
                              'tweets' : status.author.statuses_count,
                              'text' : status.text.encode('utf-8'),
                              'retweet_count' : status.retweet_count,
```

```
                                'date' : date.strftime("%m/%d/%Y")[:10]
                            })

                print (#status.author.name.encode('utf-8'),
                       #status.author.screen_name.encode('utf-8'),
                       status.author.location.encode('utf-8'),
                       #status.author.followers_count,
                       #status.author.friends_count,
                       #status.author.statuses_count,
                       status.text.encode('utf-8'),
                       #status.retweet_count,
                       date.strftime("%m/%d/%Y")[:10]
                       )

        except Exception, e:
            print >> sys.stderr, 'Encountered Exception:', e
            pass

    def on_error(self, status_code):
        print >> sys.stderr, 'Encountered error with status code:',
status_code
        return True # Don't kill the stream

    def on_timeout(self):
        print >> sys.stderr, 'Timeout...'
        return True # Don't kill the stream

# Create a streaming API and set a timeout value of 60 seconds.

streaming_api = tweepy.streaming.Stream(auth, CustomStreamListener(),
timeout=60)

# Optionally filter the statuses you want to track by providing a list
# of users to "follow".

#print >> sys.stderr, 'Filtering the public timeline for "%s"' % ('
'.join(sys.argv[1:]),)

#streaming_api.filter(follow=None, track=Q)
streaming_api.sample()
```

Ακολουθούν οι κώδικες που υλοποιούν τη σελίδα, όπου εμφανίζονται τα αποτελέσματα μέσω της MongoDB. Το PHP Framework που χρησιμοποιείται είναι το CodeIgniter.
test.php – Ο ελεγκτής της ιστοσελίδας που εμφανίζει αποτελέσματα από τη MongoDB.

```
<?php
class Test extends CI_Controller
{
    protected $data = array();
```

```
    function  __construct()
    {
        parent::__construct();
        $this->load->helper(array('form', 'url'));
        $this->load->library('form_validation');
        $this->load->model('test_model');

        // initialization of custom config items
        $this->data['stylesheet_url'] = $this->config-
>item('stylesheet_url');

    }

    public function index()
    {
        $this->load->view('header', $this->data);
        $this->load->view('mongodbView', $this->data);
        $this->load->view('footer', $this->data);
    }

    public function mongoSearchSubmit()
    {
        $this->benchmark->mark('code_start');

        $this->form_validation->set_rules('search', 'Search',
'trim|required|min_length[3]|max_length[20]');
        $this->form_validation->set_message('required', '%s is
required.');
        $this->form_validation->set_error_delimiters('<span
class="text-error">', '</span>');

        if ($this->form_validation->run() == FALSE)
        {
            echo json_encode(array(
                                "search" => form_error('search'),
                                "success" => false
                            ));
        }
        else
        {
            $search = $this->input->post('search');

            $query = $this->test_model->getTweets($search, 11);
            $this->benchmark->mark('code_end');

            echo json_encode(array(
                                "result" => $query,
                                "success" => true,
```

```
                                "benchmark" => $this->benchmark-
>elapsed_time('code_start', 'code_end')
                            ));
        }
    }

    public function handleShowMoreTweets()
    {
        $search = $this->input->post('search');
        $offset = $this->input->post('offset');

        $query = $this->test_model->getTweets($search, $offset);

        echo json_encode(array(
                            "result" => $query,
                            "offset" => $offset,
                            "success" => true
                        ));
    }

    public function mongoAdvancedSearchSubmit()
    {
        $this->benchmark->mark('code_start');

        $this->form_validation->set_rules('keyword', 'Keyword',
'trim|required|min_length[3]|max_length[20]');
        $this->form_validation->set_rules('location', 'Location',
'trim|required|min_length[3]|max_length[20]');
        $this->form_validation->set_message('required', '%s is
required.');
        $this->form_validation->set_error_delimiters('<span
class="text-error">', '</span>');

        if ($this->form_validation->run() == FALSE)
        {
            echo json_encode(array(
                                "keyword" => form_error('keyword'),
                                "location" => form_error('location'),
                                "success" => false
                            ));
        }
        else
        {
            $keyword = $this->input->post('keyword');
            $location = $this->input->post('location');
            $datepicker = $this->input->post('datepicker');
            $datepicker1 = $this->input->post('datepicker1');
```

```
            $query = $this->test_model->getAdvancedTweets($keyword,
$location, $datepicker, $datepicker1, 11);
            $this->benchmark->mark('code_end');

            echo json_encode(array(
                                "result" => $query,
                                "success" => true,
                                "benchmark" => $this->benchmark-
>elapsed_time('code_start', 'code_end')
                               ));
        }
    }

    public function handleShowMoreTweetsAdvanced()
    {
        $keyword = $this->input->post('keyword');
        $location = $this->input->post('location');
        $datepicker = $this->input->post('datepicker');
        $datepicker1 = $this->input->post('datepicker1');
        $offset = $this->input->post('offset');

        $query = $this->test_model->getAdvancedTweets($keyword,
$location, $datepicker, $datepicker1, $offset);

        echo json_encode(array(
                                "result" => $query,
                                "offset" => $offset,
                                "success" => true
                           ));
    }
}
```

test_model.php - Το μοντέλο της ιστοσελίδας που εμφανίζει αποτελέσματα από τη MongoDB.

```
<?php
class Test_model extends CI_Model
{
    public function __construct()
    {
        parent::__construct();
        //$this->load->database();
        $this->load->library('mongo_db');
    }

    public function getTweets($search, $offset)
    {
        $query = $this->mongo_db->where(array('text' => new
MongoRegex('/\W'.$search.'\W/i')))
                                ->limit($offset)
```

```
                                ->get('new2');
        return $query;
    }

    public function getadvancedTweets($keyword, $location,
$datepicker, $datepicker1, $offset)
    {
        $query = $this->mongo_db->where(array('text' => new
MongoRegex('/\W'.$keyword.'\W/i')))
                                ->where(array('location' => new
MongoRegex('/\b'.$location.'\b/i')))
                                ->where_gte('date', $datepicker)
                                ->where_lte('date', $datepicker1)
                                ->limit($offset)
                                ->get('new2');
        return $query;
    }
}
```

mongodbView.php - Η άποψη της ιστοσελίδας που εμφανίζει αποτελέσματα από τη MongoDB.

```
<head>
<link rel="stylesheet" type="text/css" href="<?php echo
$stylesheet_url; ?>css/style.css" />
<script type="text/javascript" src="<?php echo $stylesheet_url;
?>js/js_json.js"></script>
</head>
<body>
    <div class="container">
     <!-- Main Search -->
     <div class="row-fluid">
        <div id="errors" style="text-align:center;"></div>
        <form class="form-horizontal" id="searchForm" action="<?php
echo base_url(); ?>test/mongoSearchSubmit" method="post" style="text-
align:center" >
            <input type="text" id="search" name="search" value=""
class="input-xxlarge search-query">
            <button id="submit" class="btn" type="submit"
value="submit" name="submit"><i class="icon-
search"></i>Search</button>
        </form>
    </div>
    <!-- Advanced Search -->
    <div class="row-fluid">
    <form class="form-horizontal" id="searchAdvancedForm"
action="<?php echo base_url(); ?>test/mongoAdvancedSearchSubmit"
method="post" >
        <fieldset>
        <div id="legend">
```

```
            <legend class="">Advanced Search</legend>
        </div>
    <!-- Keyword -->
        <div id="advancedErrors" style="text-align:center;"></div>
        <div id="advancedErrors1" style="text-align:center;"></div>
        <div class="control-group">
            <label class="control-label" for="keyword">Keyword</label>
            <div class="controls">
                <input type="text" id="keyword" name="keyword"
placeholder="" class="input-xlarge search-query">
            </div>
        </div>
    <!-- Location -->
        <div class="control-group">
            <label class="control-label"
for="location">Location</label>
            <div class="controls">
                <input type="text" id="location" name="location"
placeholder="" class="input-xlarge search-query">
            </div>
        </div>
    <!-- Date -->
        <div class="control-group">
            <label class="control-label" for="date-from">From</label>
            <div class="controls">
                <input type="text" id="datepicker" name="datepicker"
class="input-large search-query" />
            </div>
            <label class="control-label" for="date-
until">Until</label>
            <div class="controls">
                <input type="text" id="datepicker1" name="datepicker1"
class="input-large search-query" />
            </div>
        </div>

    <!-- Button -->
        <div class="control-group">
            <div class="controls">
                <button class="btn" type="submit" value="submit1"
name="submit1"><i class="icon-search"></i>Search</button>
            </div>
        </div>
    </fieldset>
    </form>
    </div>
    <!-- showResults -->
```

```
    <table class="table table-striped table-bordered table-condensed"
id="results">
    </table>
    <button id="showMoreButton" type="submit" class="btn
displayNone"><i class="icon-plus-sign"></i>Show more</button>

    <button id="showMoreButtonAdvanced" type="submit" class="btn
displayNone"><i class="icon-plus-sign"></i>Show more advanced</button>

    <div class="displayNone">
        <div id="baseUrl" title="<?php echo base_url(); ?>"></div>

    </div>
```

js_json - Ελέγχει την εμφάνιση των αποτελεσμάτων που προέρχονται από τη MongoDB στην άποψη.

```
var offsetValue = 10;

$(function()
    {
        var ajaxFormOptions =
        {
            dataType : 'json',
            success : function(response)
            {
                if (response.success == true)
                {
                    $('#benchmark').html('<h6>Elapsed time:<h6>' +
response.benchmark);
                    if (response.result.length > 10)
                    {

$('#showMoreButton').removeClass('displayNone');
                        response.result.pop();
                        $('#results').html('');
                        $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');
                        for (var i in response.result)
                        {
$('#results').append('<tr><td>' + response.result[i].text +
'</td><td>' + response.result[i].user +  '</td><td>'
                            + response.result[i].location +
'</td><td>' + response.result[i].date + '</td></tr>');
                        }
                    }
                    else
                    {
                        $('#results').html('');
```

```
                        if (response.result.length > 0)
                        {
                        $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');

                        }
                        else
                        {
                        $('#results').html('<h3>Sorry, there is no
result!</h3>');
                        }
                        for (var i in response.result)
                        {
                            $('#results').append('<tr><td>' +
response.result[i].text + '</td><td>' + response.result[i].user +
'</td><td>' + response.result[i].location + '</td><td>' +
response.result[i].date + '</td></tr>');
                        }
                    }

                }
                else
                {
                    $('#errors').html(response.search);
                }
            }

        };
    $('#searchForm').ajaxForm(ajaxFormOptions);

        var ajaxFormOptionsAdvanced =
        {
            dataType : 'json',
            success : function(response)
            {
                if (response.success == true)
                {
                    $('#benchmark').html('<h6>Elapsed time:<h6>' +
response.benchmark);
                    if (response.result.length > 10)
                    {

$('#showMoreButtonAdvanced').removeClass('displayNone');
                        response.result.pop();
                        $('#results').html('');
                        $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');
```

```
                        for (var i in response.result)
                        {
                            $('#results').append('<tr><td>' +
response.result[i].text + '</td><td>' + response.result[i].user +
'</td><td>'
                            + response.result[i].location +
'</td><td>' + response.result[i].date + '</td></tr>');
                        }
                    }
                    else
                    {
                        $('#results').html('');
                        if (response.result.length > 0)
                        {
                        $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');

                        }
                        else
                        {
                        $('#results').html('<h3>Sorry, there is no
result!</h3>');
                        }
                        for (var i in response.result)
                        {
                            $('#results').append('<tr><td>' +
response.result[i].text + '</td><td>' + response.result[i].user +
'</td><td>' + response.result[i].location + '</td><td>' +
response.result[i].date + '</td></tr>');
                        }
                    }

                }
                else
                {
                    $('#advancedErrors').html(response.keyword);
                    $('#advancedErrors1').html(response.location);
                }
            }

        };
    $('#searchAdvancedForm').ajaxForm(ajaxFormOptionsAdvanced);

    $('#showMoreButton').click(function(event) {
        event.preventDefault();
        offsetValue += 10;
        var path = $('#baseUrl').attr('title') +
"test/handleShowMoreTweets";
```

```
        $.ajax({
              type: "POST",
              url: path,
              dataType : 'json',
              data: { search: $('#search').val(), offset: offsetValue
}
            }).success(function(response) {
              if (response.result.length < response.offset)
              {
                  $('#showMoreButton').addClass('displayNone');
              }
              else
              {
                $('#results').html('');
                $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');
                for (var i in response.result)
                {
                    $('#results').append('<tr><td>' +
response.result[i].text + '</td><td>' + response.result[i].user +
'</td><td>'
                    + response.result[i].location + '</td><td>' +
response.result[i].date + '</td></tr>');
                }
              }
            });
    });

    $('#showMoreButtonAdvanced').click(function(event) {
        event.preventDefault();
        offsetValue += 10;
        var path = $('#baseUrl').attr('title') +
"test/handleShowMoreTweetsAdvanced";
        $.ajax({
              type: "POST",
              url: path,
              dataType : 'json',
              data: { keyword: $('#keyword').val(), location:
$('#location').val(), datepicker: $('#datepicker').val(), datepicker1:
$('#datepicker1').val(), offset: offsetValue }
            }).success(function(response) {
              if (response.result.length < response.offset)
              {

$('#showMoreButtonAdvanced').addClass('displayNone');
              }
              else
              {
```

```
                $('#results').html('');
                $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');
                for (var i in response.result)
                {
                    $('#results').append('<tr><td>' +
response.result[i].text + '</td><td>' + response.result[i].user +
'</td><td>' + response.result[i].location + '</td><td>' +
response.result[i].date + '</td></tr>');
                }
            }
        });
    });
});
```

testEs.php - Ο ελεγκτής της ιστοσελίδας που εμφανίζει αποτελέσματα από το ElasticSearch.

```
<?php
include(BASEPATH.'libraries/elasticLib/elastica1.php');

class TestEs extends CI_Controller
{
    protected $data = array();

    function  __construct()
    {
        parent::__construct();
        $this->load->helper(array('form', 'url'));
        $this->load->library('form_validation');
        $this->load->library('elasticLib/elasticaInit');
        $this->load->model('testEs_model');

        // initialization of custom config items
        $this->data['stylesheet_url'] = $this->config-
>item('stylesheet_url');
    }

    public function index()
    {
        $this->load->view('header', $this->data);
        $this->load->view('esView', $this->data);
        $this->load->view('footer', $this->data);

    }

    public function esSearchSubmit()
    {
        $this->benchmark->mark('code_start');
```

```
        $this->form_validation->set_rules('search', 'Search',
'trim|required|min_length[3]|max_length[20]');
        $this->form_validation->set_message('required', '%s is
required.');
        $this->form_validation->set_error_delimiters('<span
class="text-error">', '</span>');

        if ($this->form_validation->run() == FALSE)
        {
            echo json_encode(array(
                                "search" => form_error('search'),
                                "success" => false
                            ));
        }
        else
        {
            $search = $this->input->post('search');
            $query = $this->testEs_model->getTweetsEs($search, 11);
            $this->benchmark->mark('code_end');

            echo json_encode(array(
                                "result" => $query,
                                "success" => true,
                                "benchmark" => $this->benchmark-
>elapsed_time('code_start', 'code_end')
                            ));
        }
    }

    public function handleShowMoreTweets()
    {
        $search = $this->input->post('search');
        $offset = $this->input->post('offset');

        $query = $this->testEs_model->getTweetsEs($search, $offset);

        echo json_encode(array(
                            "result" => $query,
                            "offset" => $offset,
                            "success" => true
                        ));
    }

    public function esAdvancedSearchSubmit()
    {
        $this->benchmark->mark('code_start');
```

```
        $this->form_validation->set_rules('keyword', 'Keyword',
'trim|required|min_length[3]|max_length[12]');
        $this->form_validation->set_rules('location', 'Location',
'trim|required|min_length[3]|max_length[12]');
        $this->form_validation->set_message('required', '%s is
required.');
        $this->form_validation->set_error_delimiters('<span
class="text-error">', '</span>');

        if ($this->form_validation->run() == FALSE)
        {
            echo json_encode(array(
                                "keyword" => form_error('keyword'),
                                "location" => form_error('location'),
                                "success" => false
                            ));
        }
        else
        {
            $keyword = $this->input->post('keyword');
            $location = $this->input->post('location');
            $datepicker = $this->input->post('datepicker');
            $datepicker1 = $this->input->post('datepicker1');

            $query = $this->testEs_model-
>getAdvancedTweetsEs($keyword, $location, $datepicker, $datepicker1,
11);
            $this->benchmark->mark('code_end');

            echo json_encode(array(
                                "result" => $query,
                                "success" => true,
                                "benchmark" => $this->benchmark-
>elapsed_time('code_start', 'code_end')
                            ));
        }
    }

    public function handleShowMoreTweetsAdvanced()
    {
        $keyword = $this->input->post('keyword');
        $location = $this->input->post('location');
        $datepicker = $this->input->post('datepicker');
        $datepicker1 = $this->input->post('datepicker1');
        $offset = $this->input->post('offset');

        $query = $this->testEs_model->getAdvancedTweetsEs($keyword,
$location, $datepicker, $datepicker1, $offset);
```

```
        echo json_encode(array(
                                "result" => $query,
                                "offset" => $offset,
                                "success" => true
                            ));
    }
}
```

testEs_model.php - Το μοντέλο της ιστοσελίδας που εμφανίζει αποτελέσματα από το ElasticSearch.

```
<?php
class TestEs_model extends CI_Model
{

    public function __construct()
    {
        parent::__construct();
        $this->load->library('elasticLib/elasticaInit');
    }

    public function getTweetsEs($search, $offset)
    {
        $query = new Elastica_Query_Builder();
        $query
            ->query()
                ->queryString()
                    ->field('query', $search)
                    ->fields(array("text.autocomplete"))
                ->queryStringClose()
            ->queryClose()
            ->from(0)
            ->size($offset);

        $query = new Elastica_Query($query->toArray());
        $search = new Elastica_Search(new Elastica_Client());
        $resultSet = $search->addIndex('telos')
                            ->addType('new2')
                            ->search($query);
        $results = array();
        foreach ($resultSet as $hit)
        {
            $results[] = $hit->getData();
        }
        return $results;
    }

    public function getAdvancedTweetsEs($keyword, $location,
$datepicker, $datepicker1, $offset)
```

```
    {
        $query = new Elastica_Query_Builder();
        $query
            ->query()
                ->filteredQuery()
                    ->query()
                        ->queryString()
                            ->field("query", $keyword)
                                ->fields(array("text.autocomplete"))
                        ->queryStringClose()
                    ->queryClose()
                    ->filter()
                        ->bool()
                            ->should()
                                ->range()
                                    ->fieldOpen('date')
                                        ->field('from', $datepicker)
                                        ->field('to', $datepicker1)
                                    ->fieldClose()
                                ->rangeClose()
                            ->shouldClose()
                            ->should()
                                ->term()
                                    ->field ("location", $location)
                                ->termClose()
                            ->shouldClose()
                        ->boolClose()
                    ->filterClose()
                ->filteredQueryClose()
            ->queryClose()
            ->from(0)
            ->size($offset);

        $query = new Elastica_Query($query->toArray());

        $search = new Elastica_Search(new Elastica_Client());
        $resultSet = $search->addIndex('telos')
                            ->addType('new2')
                            ->search($query);
        $results = array();
        foreach ($resultSet as $hit)
        {
            $results[] = $hit->getData();
        }
        return $results;
    }
}
```

eView.php - Η άποψη της ιστοσελίδας που εμφανίζει αποτελέσματα από το ElasticSearch.

```
<head>
<link rel="stylesheet" type="text/css" href="<?php echo
$stylesheet_url; ?>css/styleEs.css" />
<script type="text/javascript" src="<?php echo $stylesheet_url;
?>js/js_jsonEs.js"></script>
<link href='http://fonts.googleapis.com/css?family=Joti+One'
rel='stylesheet' type='text/css'>
</head>
<body>
    <div class="container">
     <!-- Main Search -->
     <div class="row-fluid">
        <div id="errors" style="text-align:center;"></div>
        <form class="form-horizontal" id="EsSearchForm" action="<?php
echo base_url(); ?>testEs/esSearchSubmit" method="post" style="text-
align:center" >
            <input type="text" id="search" name="search" value=""
class="input-xxlarge search-query">
            <button id="submit" class="btn" type="submit"
value="submit" name="submit"><i class="icon-
search"></i>Search</button>
        </form>
    </div>
    <!-- Advanced Search -->
    <div class="row-fluid">
    <form class="form-horizontal" id="EsSearchAdvancedForm"
action="<?php echo base_url(); ?>testEs/esAdvancedSearchSubmit"
method="post" >
        <fieldset>
        <div id="legend">
            <legend class="">Advanced Search</legend>
        </div>
    <!-- Keyword -->
        <div id="advancedErrors" style="text-align:center;"></div>
        <div id="advancedErrors1" style="text-align:center;"></div>
        <div class="control-group">
            <label class="control-label" for="keyword">Keyword</label>
            <div class="controls">
                <input type="text" id="keyword" name="keyword"
placeholder="" class="input-xlarge search-query">
            </div>
        </div>
    <!-- Location -->
        <div class="control-group">
            <label class="control-label"
for="location">Location</label>
            <div class="controls">
```

```
                <input type="text" id="location" name="location"
placeholder="" class="input-xlarge search-query">
            </div>
        </div>
    <!-- Date -->
        <div class="control-group">
            <label class="control-label" for="date-from">From</label>
            <div class="controls">
                <input type="text" id="datepicker" name="datepicker"
class="input-large search-query" />
            </div>
            <label class="control-label" for="date-
until">Until</label>
            <div class="controls">
                <input type="text" id="datepicker1" name="datepicker1"
class="input-large search-query" />
            </div>
        </div>

    <!-- Button -->
        <div class="control-group">
            <div class="controls">
                <button class="btn" type="submit" value="submit1"
name="submit1"><i class="icon-search"></i>Search</button>
            </div>
        </div>
    </fieldset>
    </form>
    </div>

   <!-- showResults -->
    <table class="table table-striped table-hover table-bordered
table-condensed" id="results">
    </table>
    <button id="showMoreButton" type="submit" class="btn
displayNone"><i class="icon-plus-sign"></i>Show more</button>

    <button id="showMoreButtonAdvanced" type="submit" class="btn
displayNone"><i class="icon-plus-sign"></i>Show more advanced</button>

    <div class="displayNone">
        <div id="baseUrl" title="<?php echo base_url(); ?>"></div>
    </div>
```

js_jsonEs - Ελέγχει την εμφάνιση των αποτελεσμάτων που προέρχονται από το ElasticSearch στην άποψη.

```
var offsetValue = 10;
```

```
$(function()
    {
        var ajaxFormOptions =
        {
            dataType : 'json',
            success : function(response)
            {
                if (response.success == true)
                {
                    $('#benchmark').html('<h6>Elapsed time:<h6>' +
response.benchmark);
                    if (response.result.length > 10)
                    {

$('#showMoreButton').removeClass('displayNone');
                        response.result.pop();
                        $('#results').html('');
                        $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');
                        for (var i in response.result)
                        {
$('#results').append('<tr><td>' + response.result[i].text +
'</td><td>' + response.result[i].user +  '</td><td>'
                            + response.result[i].location +
'</td><td>' + response.result[i].date + '</td></tr>');
                        }
                    }
                    else
                    {
                        $('#results').html('');
                        if (response.result.length > 0)
                        {
                        $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');

                        }
                        else
                        {
                        $('#results').html('<h3>Sorry, there is no
result!</h3>');
                        }
                        for (var i in response.result)
                        {
                            $('#results').append('<tr><td>' +
response.result[i].text + '</td><td>' + response.result[i].user +
'</td><td>' + response.result[i].location + '</td><td>' +
response.result[i].date + '</td></tr>');
```

```
                        }
                    }

                }
                else
                {
                    $('#errors').html(response.search);
                }
            }

        };
    $('#EsSearchForm').ajaxForm(ajaxFormOptions);

        var ajaxFormOptionsAdvanced =
        {
            dataType : 'json',
            success : function(response)
            {
                if (response.success == true)
                {
                    $('#benchmark').html('<h6>Elapsed time:<h6>' +
response.benchmark);
                    if (response.result.length > 10)
                    {

$('#showMoreButtonAdvanced').removeClass('displayNone');
                        response.result.pop();
                        $('#results').html('');
                        $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');
                        for (var i in response.result)
                        {
                            $('#results').append('<tr><td>' +
response.result[i].text + '</td><td>' + response.result[i].user +
'</td><td>'
                            + response.result[i].location +
'</td><td>' + response.result[i].date + '</td></tr>');
                        }
                    }
                    else
                    {
                        $('#results').html('');
                        if (response.result.length > 0)
                        {
                        $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');
```

```
                        }
                        else
                        {
                        $('#results').html('<h3>Sorry, there is no
result!</h3>');
                        }
                        for (var i in response.result)
                        {
                            $('#results').append('<tr><td>' +
response.result[i].text + '</td><td>' + response.result[i].user +
'</td><td>' + response.result[i].location + '</td><td>' +
response.result[i].date + '</td></tr>');
                        }
                    }

                }
                else
                {
                    $('#advancedErrors').html(response.keyword);
                    $('#advancedErrors1').html(response.location);
                }
            }
        };
    $('#EsSearchAdvancedForm').ajaxForm(ajaxFormOptionsAdvanced);

    $('#showMoreButton').click(function(event) {
        event.preventDefault();
        offsetValue += 10;
        var path = $('#baseUrl').attr('title') +
"testEs/handleShowMoreTweets";
        $.ajax({
              type: "POST",
              url: path,
              dataType : 'json',
              data: { search: $('#search').val(), offset: offsetValue
}
            }).success(function(response) {
              if (response.result.length < response.offset)
              {
                $('#showMoreButton').addClass('btn displayNone');
              }
              else
              {
                $('#results').html('');
                $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');
                for (var i in response.result)
                {
```

```
                    $('#results').append('<tr><td>' +
response.result[i].text + '</td><td>' + response.result[i].user +
'</td><td>'
                    + response.result[i].location + '</td><td>' +
response.result[i].date + '</td></tr>');
                }
            }
        });
    });

    $('#showMoreButtonAdvanced').click(function(event) {
        event.preventDefault();
        offsetValue += 10;
        var path = $('#baseUrl').attr('title') +
"testEs/handleShowMoreTweetsAdvanced";
        $.ajax({
            type: "POST",
            url: path,
            dataType : 'json',
            data: { keyword: $('#keyword').val(), location:
$('#location').val(), datepicker: $('#datepicker').val(), datepicker1:
$('#datepicker1').val(), offset: offsetValue }
        }).success(function(response) {
            if (response.result.length < response.offset)
            {
                $('#showMoreButtonAdvanced').addClass('btn
displayNone');
            }
            else
            {
              $('#results').html('');
              $('#results').html('<tr id="tableTitle"><td
id="tableTitle">Tweet</td><td id="tableTitle">User</td><td
id="tableTitle">Location</td><td id="tableTitle">Date</td></tr>');
                for (var i in response.result)
                {
                    $('#results').append('<tr><td>' +
response.result[i].text + '</td><td>' + response.result[i].user +
'</td><td>'
                    + response.result[i].location + '</td><td>' +
response.result[i].date + '</td></tr>');
                }
            }
        });
    });
});
```

elasticaInit.php - Αρχείο που φορτώνει τη βιβλιοθήκη Elastica στο CodeIgniter.

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access
allowed');
class ElasticaInit{
    public function __construct(){
        function autoload_elastica ($class) {
            $file = str_replace('_', '/', $class) . '.php';
            if (substr($class, 0, 9) == 'Elastica_' &&
file_exists(BASEPATH . "libraries/elasticLib/" . $file)) {
                require_once BASEPATH . "libraries/elasticLib/" .
$file;
            }
        }
        spl_autoload_register('autoload_elastica');
    }
}
new self();
```

Ακολουθούν τα header και footer που είναι κοινά και στις δύο ιστοσελίδες.
header.php

```
<!DOCTYPE html>
<head>
    <title>TWITTER SEARCH TOOL</title>
    <meta charset="utf-8"/>
    <script type="text/javascript" src="http://code.jquery.com/jquery-
1.9.1.js"></script>
    <script type="text/javascript"
src="http://code.jquery.com/ui/1.10.2/jquery-ui.js"></script>
    <script type="text/javascript" src="<?php echo $stylesheet_url;
?>js/form_plugin.js"></script>
    <script type="text/javascript" src="<?php echo $stylesheet_url;
?>js/my_js.js"></script>
    <link rel="stylesheet" type="text/css"
href="http://code.jquery.com/ui/1.10.2/themes/smoothness/jquery-
ui.css" />
    <link rel="stylesheet" type="text/css" href="<?php echo
$stylesheet_url; ?>css/bootstrap/css/bootstrap.css" />
    <link rel="stylesheet" type="text/css" href="<?php echo
$stylesheet_url; ?>css/bootstrap/css/bootstrap-responsive.css" />
    <link href='http://fonts.googleapis.com/css?family=Joti+One'
rel='stylesheet' type='text/css'>
    <link rel="stylesheet" type="text/css" href="<?php echo
$stylesheet_url; ?>css/datepicker.css" />
</head>
<body>
    <div id="navbar" class="navbar">
        <div class="hero-unit" align="center">
            <h1>TWITTER SEARCH TOOL</h1>
            <a href="http://localhost/CodeIgniter-
Bootstrap/public_html/" style="text-decoration:none;">
```

```
                <img src="<?php echo $stylesheet_url;
?>images/mongodb-leaf.png" title='MongoDB' style="height:8%;
width:8%;" />
            </a>
            <a href="http://localhost/CodeIgniter-
Bootstrap/public_html/testEs" style="text-decoration:none;">
                <img src="<?php echo $stylesheet_url;
?>images/bonsai1.png" title='ElasticSearch' style="height:8%;
width:8%;" />
            </a>
        </div>
    </div>
```

footer.php

```
<div id="benchmark" style="text-align:left;"></div>
<div><?php echo date("Y"); ?></div>
</body>
</html>
```

Η ανάλυση και η χαρτογράφηση που χρησιμοποιούνται στο index «telos» του ElasticSearch.

```
{
  "settings":{
        "analysis":{
            "filter":{
                "trip_ngram": {
                    "type" : "edgeNgram",
                    "max_gram" : 15,
                    "min_gram" : 3 }
            },
           "analyzer":{
              "index_ngram_analyzer" : {
                  "type" : "custom",
                  "tokenizer" : "standard",
                  "filter" : ["standard", "lowercase", "trip_ngram"]
              },
              "search_ngram_analyzer" : {
                "type" : "custom",
                "tokenizer" : "standard",
                "filter" : ["standard", "lowercase"]
              }
           }
        }
    },
  "mappings":{
      "new2":{
         "properties":{
            "date" : {"type" : "date", "format" : "MM/dd/yy"},
```

```
            "followers" : {"type" : "long", "index" : "not_analyzed"},
            "user" : {"type" : "string"},
            "tweets" : {"type" : "long", "index" : "not_analyzed"},
            "text":{
              "type" : "multi_field",
                "fields" : {
                    "text.autocomplete" : {
                        "search_analyzer" : "search_ngram_analyzer",
                        "index_analyzer" : "index_ngram_analyzer",
                        "type" : "string"
            }
        }
    },
            "followings" : {"type" : "long", "index" :
"not_analyzed"},
            "retweet_count" : {"type" : "long", "index" :
"not_analyzed"},
            "nickname" : {"type" : "string"},
            "location" : {"type" : "string"}
         }
      }
    }
}
```