ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Δυναμική απεικόνιση γεωσυσχετισμένων μετρικών χρήσης, επιρροής και εμβέλειας γλωσσικού περιεχομένου από το live open API του Twitter

Μάριος Σιγανός

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
Υπεύθυνος
Ιωάννης Αναγνωστόπουλος
Επίκουρος Καθηγητής

Λαμία, Σεπτέμβριος 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ
ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Δυναμική απεικόνιση γεωσυσχετισμένων μετρικών χρήσης, επιρροής και εμβέλειας γλωσσικού περιεχομένου από το live open API του Twitter

Μάριος Σιγανός

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων
Ιωάννης Αναγνωστόπουλος
Επίκουρος Καθηγητής

Λαμία, Σεπτέμβριος 2015

Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. *Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί* ***χωρίς να τα περικλείω σε εισαγωγικά*** *και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

2. *Δέχομαι ότι η αυτολεξεί* ***παράθεση χωρίς εισαγωγικά****, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. *Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια*

4. *Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.*

Ημερομηνία: ……/…../20……

Ο – Η Δηλ.

(Υπογραφή)

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.

Δυναμική απεικόνιση γεωσυσχετισμένων μετρικών χρήσης, επιρροής και εμβέλειας γλωσσικού περιεχομένου από το live open API του Twitter

Μάριος Σιγανός

Τριμελής Επιτροπή:

Ιωάννης Αναγνωστόπουλος, Επίκουρος Καθηγητής (επιβλέπων)

Βασίλειος Πλαγιανάκος, Αναπληρωτής Καθηγητής

Αθανάσιος Λουκόπουλος, Λέκτορας

Ευχαριστήριο Σημείωμα

Στο σημείο αυτό θα ήθελα να εκφράσω τις ειλικρινείς ευχαριστίες μου στον επιβλέποντα καθηγητή κ. Ιωάννη Αναγνωστόπουλο για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα καθώς και για την πολύτιμη καθοδήγηση και υποστήριξη του κατά την διάρκεια εκπόνησης της εργασίας.

Τέλος, ένα μεγάλο ευχαριστώ στην οικογένεια μου και στους φίλους μου για τη συνεχή υποστήριξη καθ' όλα τα χρόνια των προπτυχιακών μου σπουδών.

ΠΕΡΙΛΗΨΗ

Οι ιστοσελίδες κοινωνικής δικτύωσης με την πάροδο των χρόνων έχουν μετατραπεί σε πολύτιμες πηγές για συλλογή πληροφοριών. Η εκμετάλευση του μεγάλου όγκου δεδομένων από αυτές και η εξαγωγή χρήσιμων συμπερασμάτων αποτελούν μεγάλες προκλήσεις. Στα πλαίσια της παρούσας διπλωματικής εργασίας αναπτύχθηκε ένα σύστημα αυτόματης συλλογής γεωσυσχετισμένων δεδομένων από το Twitter και μέσω μιας διαδικτυακής εφαρμογής δίνεται η δυνατότητα στο χρήστη να δει σε πραγματικό χρόνο την κατανομή των tweets στο χάρτη δίνοντας ταυτόχρονα πληροφορίες σχετικά με τη χρήση, την επιρροή αλλά και την εμβέλεια του γλωσσικού περιεχομένου.

Λέξεις-κλειδιά: Twitter, tweets, κοινωνικά δίκτυα, ιστοσελίδες κοινωνικής δικτύωσης, εξώρυξη κοινωνικού ιστού, επιρροή, γλωσσικό περιεχόμενο, μεταδεδομένα τοποθεσίας, γεωγραφική κατανωμή, πραγματικού-χρόνου

ABSTRACT

Social networking sites have become over the years valuable sources for information gathering. The exploitation of large amounts of data from these andthe extraction of useful conclusions represent great challenges. In the context of this thesis, an automated system for geospatial data collection from Twitter has been developed and through a web application the user is able to view in real time the distribution of tweets on the map while giving information on the use, influence and the range of the linguistic content.

Keywords: Twitter, tweets, social networks, social networking sites, social web mining, influence, linguistic content, geospatial data, geographical distribution, real-time

Περιεχόμενα

1. Εισαγωγή

1.1 Γενικό πλαίσιο

Η καθολική εξάπλωση του Διαδικτύου έχει επιφέρει σημαντικές αλλαγές τόσο στις μεθόδους και τα μέσα με τα οποία οι άνθρωποι επικοινωνούν και αλληλεπιδρούν μεταξύ τους όσο και στους τρόπους με τους οποίους ανταλλάσουν πληροφορίες. Κεντρικό ρόλο στην καθιέρωση των αλλαγών αυτών έχουν διαδραματίσει τα λεγόμενα διαδικτυακά κοινωνικά μέσα και οι ιστότοποι κοινωνικής δικτύωσης και διαμοίρασης περιεχομένου. Οι ι στότοποι αυτοί εμφανίζουν ολοένα και μεγαλύτερη δημοτικότητα, όπως αποδεικνύεται από τον συνεχώς αυξανόμενο αριθμό νέων μελών και επισκέψεων που δέχονται καθημερινά. Μια από τις δημοφιλέστερες πλατφόρμες αυτού του τύπου είναι η ιστοσελίδα κοινωνικής δικτύωσης Twitter. Τα δεδομένα που υπάρχουν στο Twitter αποτελούν σημαντική πηγή πληροφοριών, που για να αξιοποιηθούν όμως πρέπει πρώτα να οργανωθούν και να αναλυθούν με κάποιο αποτελεσματικό τρόπο. Ο όγκος των δεδομένων που υπάρχουν στο Twitter, καθώς και ο ρυθμός που αυτά παράγονται καθιστά την διαχείριση τους μια σημαντική πρόκληση. Περισσότερα από το 3% όλων των tweets βρέθηκαν να έχουν διαθέσιμη εγγενή πληροφορία τοποθεσίας, ενώ χάρη στη τοποθεσία που παρέχεται από το χρήστη και από τα πεδία στο προφίλ περισσότερο από το ένα τρίτο όλων των tweets έχει αναγνωρισμένη γεωγραφική θέση με υψηλή ακρίβεια. [22] Η ακριβής γεωγραφική θέση δημιουργίας ενός tweet μπορεί να προσδιοριστεί αυτόματα με την βοήθεια τεχνολογιών όπως WiFi ή GPS που τα σύγχρονα κινητά τηλέφωνα παρέχουν.

Κύριος στόχος της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας διαδικτυακής εφαρμογής που θα διαχειρίζεται τα δεδομένα του Twitter σε πραγματικό χρόνο, δίνοντας έμφαση τόσο στην γεωγραφική πληροφορία που συνοδεύει αυτά τα δεδομένα, όσο και σε μετρικές που βασίζονται στο γλωσσικό περιεχόμενο.

1.2 Σκοπός και Περιγραφή

Σκοπός της παρούσας διπλωματικής εργασίας είναι να εξαχθούν και να αξιοποιηθούν χρήσιμες πληροφορίες από το Twitter. Πιο συγκεκριμένα το ενδιαφέρον επικεντρώνεται στη συλλογή tweets σε πραγματικό χρόνο, που συνοδεύονται από γεωγραφική πληροφορία, με σκοπό την ανάπτυξη ενός συστήματος που παρέχει στο χρήστη μετρικές χρήσης, επιρροής και εμβέλειας γλωσσικού περιεχομένου. Για την συλλογή των δεδομένων σε πραγματικό χρόνο αξιοποιήθηκαν οι δυνατότητες της οργανωμένης και λειτουργικής διεπαφής προγραμματισμού εφαρμογών (API) του Twitter. Αφού συλλεχθούν τα δεδομένα, εξάγονται οι πληροφορίες που χρειάζονται και στην συνέχεια αποθηκεύονται σε μια βάση δεδομένων. Ως βάση δεδομένων χρησιμοποιήθηκε η μη-σχεσιακή (NoSQL) βάση δεδομένων MongoDB. Μετά ακολουθεί η ανάλυση των δεδομένων και διάφοροι υπολογισμοί όπως η επιρροή του κάθε χρήστη που δημοσίευσε ένα tweet. Η γλώσσα προγραμματισμού που επιλέχθηκε για την ανάπτυξη του συστήματος είναι η python κυρίως λόγω της ευκολίας χρήσης και της γρήγορης ανάπτυξης κώδικα. Τέλος δημιουργήθηκε ένα απλό και εύχρηστο διαδικτυακό γραφικό περιβάλλον,το οποίο δίνει στον χρήστη την δυνατότητα να λάβει τη ζητούμενη πληροφορία. Για αυτόν τον σκοπό χρησιμοποιήθηκαν διάφορα μέσα για την οπτικοποίηση

της πληροφορίας όπως χάρτες, γραφήματα και χάρτες θερμότητας. Η εφαρμογή είναι δυναμική καθώς ο χρήστης μπορεί να αλληλεπιδράσει μαζί της μέσω των διαφόρων επιλογών εμφάνισης και περιορισμού των αποτελεσμάτων. Στην π λευρά του πελάτη χρησιμοποιήθηκε ένα σύνολο γνωστών διαδικτυακών τεχνολογιών όπως HTML5, CSS3, Javascript και JQuery αλλά και βιβλιοθηκών, προκειμένου να παρουσιαστούν τα αποτελέσματα στον χρήστη.

1.3 Οργάνωση κειμένου

Το παρόν κείμενο επιχειρεί να παρουσιάσει με σαφή και περιεκτικό τρόπο κάθε πτυχή της ανάπτυξης του συστήματος και των δυνατοτήτων του.

Στο δεύτερο κεφάλαιο γίνεται μια επισκόπηση στο χώρο κάνοντας αναφορά τόσο στα κοινωνικά δίκτυα γενικά αλλά και στο Twitter πιο συγκεκριμένα. Στο ίδιο κεφάλαιο περιγράφονται και οι μετρικές που χρησιμοποιούνται.

Το τρίτο κεφάλαιο επικεντρώνεται στην αρχιτεκτονική του συστήματος που υλοποιήθηκε στα πλαίσια της παρούσας πτυχιακής εργασίας και γίνεται αναλυτική περιγραφή όλων των σταδίων που υπάρχουν. Επιπρόσθετα, αναφέρονται οι τεχνολογίες που χρησιμοποιήθηκαν κατά την υλοποίηση της εφαρμογής.

Στο τέταρτο κεφάλαιο γίνεται η παρουσίαση της εφαρμογής και η περιγραφή των βασικών λειτουργιών της. Περιλαμβάνεται επίσης ένα παράδειγμα χρήσης.

Στο πέμπτο κεφάλαιο γίνεται η σύνοψη της διπλωματικής εργασίας, παρουσιάζονται τα συμπεράσματα που εξήχθησαν από την εκπόνηση της εργασίας και αναφέρονται προτάσεις για μελλοντική επέκταση του συστήματος.

2. Επισκόπηση στο χώρο

2.1 Κοινωνικά δίκτυα

Τα κοινωνικά δίκτυα έχουν γίνει ένα πολύ δημοφιλές εργαλείο επικοινωνίας μεταξύ των χρηστών του Διαδικτύου. Εκατομμύρια χρήστες μοιράζονται τις απόψεις τους πάνω σε θέματα της καθημερινότητας και συνεπώς οι ιστοσελίδες αυτές κατακλύζονται από πληθώρα δεδομένων. Ένα κοινωνικό δίκτυο είναι μία κοινωνική δομή αποτελούμενη από "κόμβους" (συνήθως άτομα ή επιχειρήσεις) οι οποίοι συνδέονται μεταξύ τους με έναν ή περισσότερους καθορισμένους τύπους αλληλεξάρτησης, όπως φιλία, συγγένεια, κοινά ενδιαφέροντα, οικονομικές συναλλαγές, αντιπάθεια, σεξουαλικές επαφές ή σχέσεις πεποιθήσεων, γνώσεων, οραμάτων ή αξιών. [12] Τα δίκτυα που δημιουργούνται μέσω του Παγκόσμιου Ιστού ή αλλιώς ιστοσελίδες κοινωνικής δικτύωσης είναι διαδικτυακές υπηρεσίες που επιτρέπουν στα άτομα να δημιουργήσουν ένα δημόσιο προφίλ, να δημιουργήσουν μια λίστα των χρηστών με τους οποίους μπορούν να μοιράζονται πληροφορίες, και να προβάλουν τις συνδέσεις εντός του συστήματος. Έχουν οποιοδήποτε αριθμό μελών, από μερικές δεκάδες μέχρι εκατομμύρια και προσφέρουν στους ανθρώπους τη δυνατότητα να έρθουν σε επαφή μεταξύ τους με βάση κοινά τους ενδιαφέροντα. [13] Οι χρήστες του διαδικτύου παρουσιάζουν μια ραγδαία αύξηση στην οποία σημαντικό ρόλο παίζουν οι υπηρεσίες κοινωνικής δικτύωσης που κερδίζουν συνεχώς περισσότερους χρήστες. Οι ιστοσελίδες αυτές κατακλύζονται από πληθώρα δεδομένων πολλά από τα οποία συνοδεύονται από μεταδεδομένα, όπως η γεωγραφική θέση (geotagging) ή κάποια ετικέτα. Λόγω της μαζικής παραγωγής προσιτών σε κόστος κινητών τηλεφώνων με ενσωματωμένη μονάδα παγκοσμίου συστήματος εντοπισμού θέσης (Global Positioning System ή GPS), τα μεταδεδομένα τοποθεσίας, όπως το γεωγραφικό μήκος και πλάτος συνδέονται αυτόματα με τα υπόλοιπα δεδομένα που οι χρήστες παράγουν. Είναι λοιπόν εμφανής η ανάγκη και η πρόκληση να μπορέσουν να συλλεχθούν αυτά τα δεδομένα και να διαχειριστούν κατάλληλα ώστε να εξαχθούν χρήσιμες πληροφορίες. Χαρακτηριστικά παραδείγματα κοινωνικών δικτύων είναι το Facebook και το Twitter.

2.2 Twitter

2.2.1 Περιγραφή

Το Twitter είναι μία ιστοσελίδα κοινωνικής δικτύωσης που επιτρέπει στους χρήστες του να στέλνουν και να διαβάζουν σύντομα μηνύματα, μέχρι 140 χαρακτήρες, τα οποία ονομάζονται tweets. Τα μηνύματα μπορούν να αναγνωστούν και από μη συνδεδεμένους χρήστες, αλλά μόνο οι συνδεδεμένοι μπορούν να δημοσιεύσουν κείμενα. Τα tweets είναι δημόσια ορατά από προεπιλογή, αλλά οι αποστολείς μπορούν να περιορίσουν την παράδοση των μηνυμάτων μόνο στους ακολούθους τους. Οι χρήστες μπορούν να κάνουν ένα τιτίβισμα μέσω της ιστοσελίδας του Twitter, συμβατών εξωτερικών εφαρμογών, ή μέσω σύντομου μηνύματος (SMS) που διατίθενται σε ορισμένες χώρες. Το retweet είναι όταν ένα τιτίβισμα προωθείται μέσω του Twitter από τους χρήστες. Οι χρήστες μπορούν να εγγραφούν στα tweets άλλων χρηστών. Αυτό είναι γνωστό ως "ακολουθώ" (following) και οι συνδρομητές

είναι γνωστοί ως "ακόλουθοι" (followers). Οι χρήστες μπορούν να ο μαδοποιήσουν τις δημοσιεύσεις τους σύμφωνα με το θέμα ή τον τύπο με τη χρήση των hashtags που είναι λέξεις ή φράσεις με πρόθεμα ένα "#" σημάδι. Ομοίως, το σύμβολο "@" ακολουθούμενο από ένα όνομα χρήστη χρησιμοποιείται από έναν χρήστη για να αναφερθεί ή να απαντήσει σε άλλους. Στο Twitter υπάρχει επίσης η δυνατότητα οι χρήστες να ακολουθήσουν λίστες χρηστών και όχι απλά μεμονωμένους χρήστες με το "Twitter Lists". Τα tweets τέθηκαν σε ένα αυστηρό όριο 140 χαρακτήρων για συμβατότητα με τα μηνύματα SMS. Το όριο των 140 χαρακτήρων αύξησε επίσης τη χρήση των υπηρεσιών συντόμευσης URL και υπηρεσιών φιλοξενίας περιεχομένου. Μια λέξη, φράση ή το θέμα που αναφέρεται σε μεγαλύτερο ποσοστό από άλλα λέγεται ότι είναι ένα "trending topic". Τα θέματα αυτά βοηθούν το Twitter και τους χρήστες του να καταλάβουν τι συμβαίνει στον κόσμο και ποιες είναι οι απόψεις των ανθρώπων σχετικά με αυτό. [18]

Το Twitter είναι μια πλούσια πηγή κοινωνικών δεδομένων λόγω του άνοιγματος του για δημόσια κατανάλωση, καθαρό και καλά τεκμηριωμένο API, πλούσιο εξοπλισμό για προγραμματιστές, καθώς και ευρεία απήχηση στους χρήστες.

Υπάρχουν περίπου 316 εκατομμύρια ενεργοί χρήστες κάθε μήνα και γίνονται 500 εκατομμύρια tweets κάθε μέρα. [16]

2.2.2 Twitter APIs

Το Twitter αποτελεί αντικείμενο έρευνας αλλά και την βάση για την ανάπτυξη πολλών εφαρμογών. Συνεπώς η αξιοποίηση του στα πλαίσια τέτοιων εφαρμογών μπορεί να φανεί ιδιαίτερα χρήσιμη.

Για να μπορέσει να αξιοποιηθεί η πληροφορία από το Twitter πρέπει οι εφαρμογές να αποκτήσουν πρόσβαση σε αυτά τα δεδομένα. Για αυτό το λόγο το Twitter παρέχει στους προγραμματιστές κάποιες διεπαφές προγραμματισμού εφαρμογών (APIs), ώστε οι εφαρμογές να μπορούν να επικοινωνούν με το Twitter. Οι εφαρμογές αυτές μπορούν να αλληλεπιδρούν με το Twitter έχοντας δικαιώματα είτε ανάγνωσης και εγγραφής είτε μόνο ανάγνωσης. Για τη χρήση των APIs χρειάζεται η δημιουργία ενός λογαριασμού στο Twitter. Στη συνέχεια είναι απαραίτητη η απόκτηση διαπιστευτηρίων που γίνεται με τη δημιουργία νέας εφαρμογής από τον λογαριασμό. Για την απλότητα της ανάπτυξης, τα βασικά κομμάτια των πληροφοριών που θα πρέπει να πάρει κάποιος από τις ρυθμίσεις της καινούριας εφαρμογής είναι τα ακόλουθα: *consumer key*, *consumer secret*, *access token*, and *access token secret*.

Στη συνέχεια θα γίνει περιγραφή των βασικών χαρακτηριστικών του Rest API και του Streaming API που είναι δύο από τις διεπαφές του Twitter.

REST APIs

Τα REST APIs παρέχουν πρόσβαση μέσω προγραμματισμού για να μπορεί κάποιος να διαβάσει και να γράψει δεδομένα στο Twitter. Τα REST APIs ταυτοποιούν εφαρμογές και

χρήστες του Twitter με τη χρήση του OAuth. Οι απαντήσεις είναι διαθέσιμες σε μορφή JSON.

Streaming APIs

Τα Streaming APIs παρέχουν συνεχώς νέες απαντήσεις σε ερωτήματα στο REST API μέσω μιας μακράς διάρκειας ζωής HTTP σύνδεσης. Δίνουν στους προγραμματιστές πρόσβαση χαμηλής λανθάνουσας κατάστασης στην παγκόσμια ροή δεδομένων του Twitter. Τα μεμονωμένα μηνύματα που λαμβάνονται από αυτό το API είναι κωδικοποιημένα σε JSON.

Το Twitter προσφέρει τρία καταληκτικά σημεία (endpoints) ροής, κάθε ένα από τα οποία είναι προσαρμοσμένο σε διαφορετικές περιπτώσεις χρήσης.

Public streams
Ροές των δημόσιων δεδομένων που ρέουν μέσω του Twitter. Κατάλληλο για να ακουθήσει κανείς συγκεκριμένους χρήστες ή θέματα αλλά και την εξόρυξη δεδομένων.

User streams
Ροές για ένα χρήστη που περιέχουν δεδομένα που αντιστοιχούν σε ένα χρήστη του Twitter.

Site streams
Η έκδοση πολλών χρηστών για τις ροές ενός χρήστη. Προορίζονται για διακομιστές που πρέπει να συνδεθούν στο Twitter για λογαριασμό πολλών χρηστών.

Πιο αναλυτικά για τις δημόσιες ροές που ήταν και αυτές που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής.

Μόλις οι εφαρμογές δημιουργήσουν μια σύνδεση σε ένα καταληκτικό σημείο ροής, αρχίζουν να λαμβάνουν μια ροή των tweets, χωρίς να χρειάζεται να ασχοληθεί κανείς με το polling ή τα όρια που υπάρχουν στο REST API.

Υπάρχουν τα ακόλουθα τρία καταληκτικά σημεία:

- ✓ **POST statuses / filter**
  Επιστρέφει δημόσιες ενημερώσεις καταστάσης που αντιστοιχούν στην αναζήτηση με βάση κάποια φίλτρα. Μπορούν να προσδιορίζονται πολλαπλές παράμετροι γεγονός που επιτρέπει πλέον στις εφαρμογές να χρησιμοποιούν μια ενιαία σύνδεση με το streaming API. Στα φίλτρα επιτρέπεται η χρήση λέξεων-κλειδιών, αναγνωριστικών χρηστών και τοποθεσιών.

- ✓ **GET statuses / sample**
  Επιστρέφει ένα μικρό τυχαίο δείγμα όλων των δημόσιων ενημερώσεων κατάστασης.

- ✓ **GET statuses / firehose**
  Επιστρέφει όλες τις δημόσιες καταστάσεις. Αυτό το καταληκτικό σημείο απαιτεί ειδική άδεια για την πρόσβαση.

Κάθε λογαριασμός μπορεί να δημιουργήσει μόνο μία σύνδεση στα δημόσια καταληκτικά σημεία. Μία πιθανή απόπειρα δημιουργίας σύνδεσης περισσότερο από μία φορά με τα ίδια διαπιστευτήρια του λογαριασμού θα οδηγήσει σε αποσύνδεση της παλαιότερης. Στις περιπτώσεις που γίνονται υπερβολικές προσπάθειες σύνδεσης είτε επιτυχείς είτε ανεπιτυχείς υπάρχει ο κίνδυνος αυτόματης απαγόρευσης της IP.

Για την υλοποίηση της εφαρμογής έπρεπε να επιλεχθεί μία διεπαφή εφαρμογών που να είναι ικανή να την εφοδιάσει με μεγάλο όγκο δεδομένων πραγματικού χρόνου που να περιέχουν γεωγραφική πληροφορία. Για αυτό το λόγω επιλέχθηκε το Streaming API. Με ένα απλό αίτημα HTTP στο API η εφαρμογή αποκτά πρόσβαση στη δημόσια ροή δεδομένων. Πιο συγκεκριμένα με τη χρήση του καταληκτικού σημείου sample γίνεται λήψη ενός δείγματος της ροής πραγματικού χρόνου που αντιστοιχεί περίπου στο 1% όλων των δημόσιων tweets. [24] Αυτό γίνεται μέσω μιας τεχνικής τυχαίας δειγματοληψίας που αντιπροσωπεύει το μεγαλύτερο πληθυσμό των tweets. Αν και μπορεί να σκεφτεί κάποιος ότι το 1% φαίνεται ασήμαντο, στη πραγματικότητα όμως η ταχύτητα των tweets μπορεί σε στιγμές αιχμής να είναι δεκάδες χιλιάδες tweets ανά δευτερόλεπτο. Η πρόσβαση σε έως και 1% του συνόλου των δημόσιων tweets είναι πολύ σημαντική. Στη συνέχεια επιλέγονται μόνο τα tweets που περιέχουν τις συντεταγμένες της τοποθεσίας που δημοσιεύτηκε το καθε ένα από αυτά.

Σε ένα tweet υπάρχουν πολλά περισσότερα απ'όσα μπορεί να δει κανείς με το μάτι.

Εικόνα 1. Παράδειγμα ενός tweet όπως εμφανίζεται στη σελίδα του Twitter.

Στη συνέχεια υπάρχει ένα παράδειγμα ενός tweet σε μορφή json όπου μπορεί πολύ εύκολα να αντιληφτεί κάποιος ότι το μέγεθος του είναι πολύ μεγαλύτερο από τους 140 χαρακτήρες κειμένου που θεωρείται ως ένα tweet.


```
{
 "created_at": "Wed Aug 26 19:47:31 +0000 2015",
 "id": 636626044473208800,
 "id_str": "636626044473208833",
 "text": "This is an example of a #tweet! #social http://t.co/8TtBk1U1CB",
 "source": "<a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>",
 "truncated": false,
```

```
  "in_reply_to_status_id": null,
  "in_reply_to_status_id_str": null,
  "in_reply_to_user_id": null,
  "in_reply_to_user_id_str": null,
  "in_reply_to_screen_name": null,
  "user": {
    "id": 244977111,
    "id_str": "244977111",
    "name": "Marios Siganos",
    "screen_name": "msiganos",
    "location": "San Francisco, CA",
    "description": "Software_Engineer Biomedical_Informatics",
    "url": "http://t.co/67zfVt8lQj",
    "entities": {
      "url": {
        "urls": [
          {
            "url": "http://t.co/67zfVt8lQj",
            "expanded_url": "http://msiganos.csbigeeks.com",
            "display_url": "msiganos.csbigeeks.com",
            "indices": [
              0,
              22
            ]
          }
        ]
      },
      "description": {
        "urls": []
      }
    },
    "protected": false,
    "followers_count": 225,
    "friends_count": 711,
    "listed_count": 26,
    "created_at": "Sun Jan 30 15:46:33 +0000 2011",
    "favourites_count": 8,
    "utc_offset": -25200,
    "time_zone": "Pacific Time (US & Canada)",
    "geo_enabled": true,
    "verified": false,
    "statuses_count": 1139,
    "lang": "en",
    "contributors_enabled": false,
    "is_translator": false,
    "is_translation_enabled": false,
    "profile_background_color": "131516",
    "profile_background_image_url": "http://abs.twimg.com/images/themes/theme14/bg.gif",
    "profile_background_image_url_https": "https://abs.twimg.com/images/themes/theme14/bg.gif",
    "profile_background_tile": true,
    "profile_image_url": "http://pbs.twimg.com/profile_images/570572368403406848/ojsI0v-D_normal.jpeg",
    "profile_image_url_https": "https://pbs.twimg.com/profile_images/570572368403406848/ojsI0v-
D_normal.jpeg",
    "profile_banner_url": "https://pbs.twimg.com/profile_banners/244977111/1404220570",
    "profile_link_color": "05BC24",
    "profile_sidebar_border_color": "EEEEEE",
    "profile_sidebar_fill_color": "EFEFEF",
    "profile_text_color": "333333",
```

```
  "profile_use_background_image": true,
  "has_extended_profile": false,
  "default_profile": false,
  "default_profile_image": false,
  "following": true,
  "follow_request_sent": false,
  "notifications": false
},
"geo": null,
"coordinates": null,
"place":  {
  "id": "0052e89a59f7eb6b",
  "url": "https://api.twitter.com/1.1/geo/id/0052e89a59f7eb6b.json",
  "place_type": "city",
  "name": "Kallikrateia",
  "full_name": "Kallikrateia, Kentriki Makedonia",
  "country_code": "GR",
  "country": "Hellas",
  "contained_within":  [],
  "bounding_box":  {
    "type": "Polygon",
    "coordinates":  [
      [
        [
          23.0048484,
          40.272597
        ],
        [
          23.1606688,
          40.272597
        ],
        [
          23.1606688,
          40.4312588
        ],
        [
          23.0048484,
          40.4312588
        ]
      ]
    ]
  },
  "attributes":  {}
},
"contributors": null,
"is_quote_status": false,
"retweet_count": 0,
"favorite_count": 0,
"entities":  {
  "hashtags":  [
    {
      "text": "tweet",
      "indices":  [
        24,
        30
      ]
    },
    {
```

```
      "text": "social",
      "indices": [
       32,
       39
      ]
     }
    ],
    "symbols": [],
    "user_mentions": [],
    "urls": [
     {
      "url": "http://t.co/8TtBk1U1CB",
      "expanded_url": "http://dib.uth.gr",
      "display_url": "dib.uth.gr",
      "indices": [
       40,
       62
      ]
     }
    ]
   },
   "favorited": false,
   "retweeted": false,
   "possibly_sensitive": false,
   "possibly_sensitive_appealable": false,
   "lang": "en"
}
```


Αντικείμενα του API
Υπάρχουν τέσσερα κύρια αντικείμενα στο API και είναι τα ακόλουθα: "Tweets", "Users", "Entities" και "Places".

Στη συνέχεια θα αναφερθούν κάποια πεδία των αντικειμένων που λαμβάνονται από τα API και τα οποία χρησιμοποιήθηκαν στην εφαρμογή.

Αντικείμενο Tweets
Τα Tweets ή αλλιώς οι ενημερώσεις κατάστασης είναι το βασικό δομικό στοιχείο στο Twitter.

Στο αντικείμενο Tweets υπάρχουν τα ακόλουθα πεδία:

- *id_str*
  Είναι η αναπαράσταση συμβολοσειράς του μοναδικού αναγνωριστικού για κάθε tweet.

- *text*
  Είναι το πραγματικό UTF-8 κείμενο της ενημέρωσης κατάστασης.

- *created_at*
  Είναι η ώρα UTC που δημιουργήθηκε το tweet.

- *coordinates*

Αντιπροσωπεύει τη γεωγραφική θέση αυτού του tweet όπως αναφέρθηκε από το χρήστη ή κάποια άλλη εφαρμογή.

- *coordinates*
  Είναι το γεωγραφικό μήκος και πλάτος της θέσης του Tweet με τη μορφή [γεωγραφικό μήκος, γεωγραφικό πλάτος].

- *entities*
  Παρέχουν μεταδεδομένα και πρόσθετες συναφείς πληροφορίες σχετικά με περιεχόμενο που δημοσιεύτηκε στο Twitter.
  - *hashtags*
    Αντιπροσωπεύει τα hashtags που έχουν εξαχθεί από το κείμενο του Tweet.

- *lang*
  Υποδηλώνει ένα αναγνωριστικό γλώσσας BCP 47 που αντιστοιχεί στη γλώσσα του κειμένου του tweet που ανιχνεύεται από το Twitter.

Αντικείμενο Users
Στο αντικείμενο Users υπάρχουν τα ακόλουθα πεδία:

- *id_str*
  Είναι η αναπαράσταση συμβολοσειράς του μοναδικού αναγνωριστικού για κάθε χρήστη.

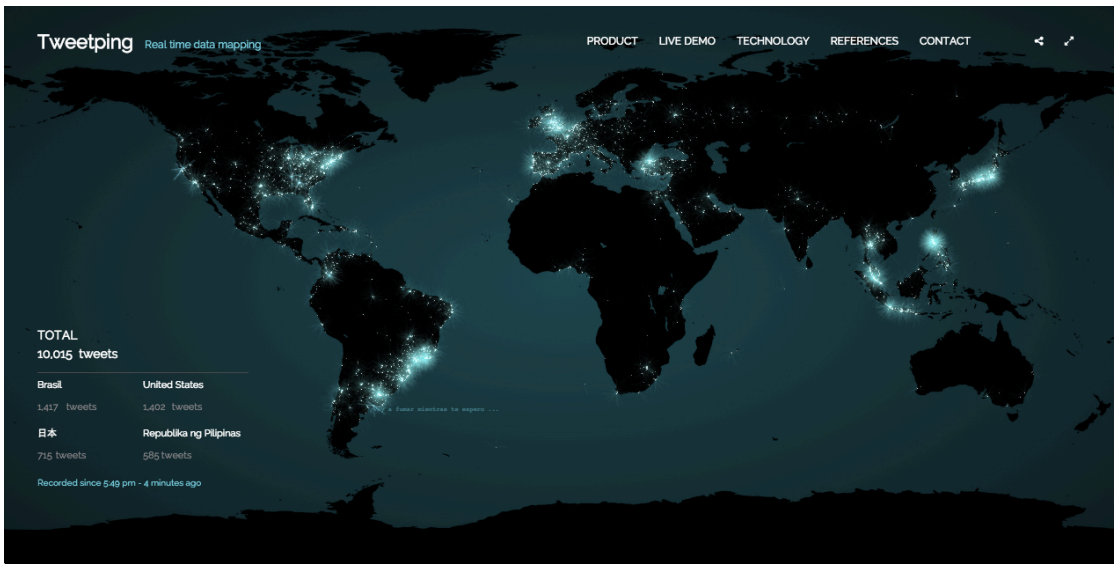
- *screen_name*
  Είναι το όνομα ή ψευδώνυμο που χρησιμοποιεί ο χρήστης για να ταυτοποιήσει τον εαυτό του.

2.2.3 Υπάρχουσες εφαρμογές με γεωγραφική συσχέτιση πληροφορίας

Σε αυτό το σημείο θα γίνει αναφορά σε κάποιες εφαρμογές του διαδικτύου που έχουν παρόμοια χαρακτηριστικά με αυτή που υλοποιήθηκε στα πλαίσια της διπλωματικής.

http://tweetping.net

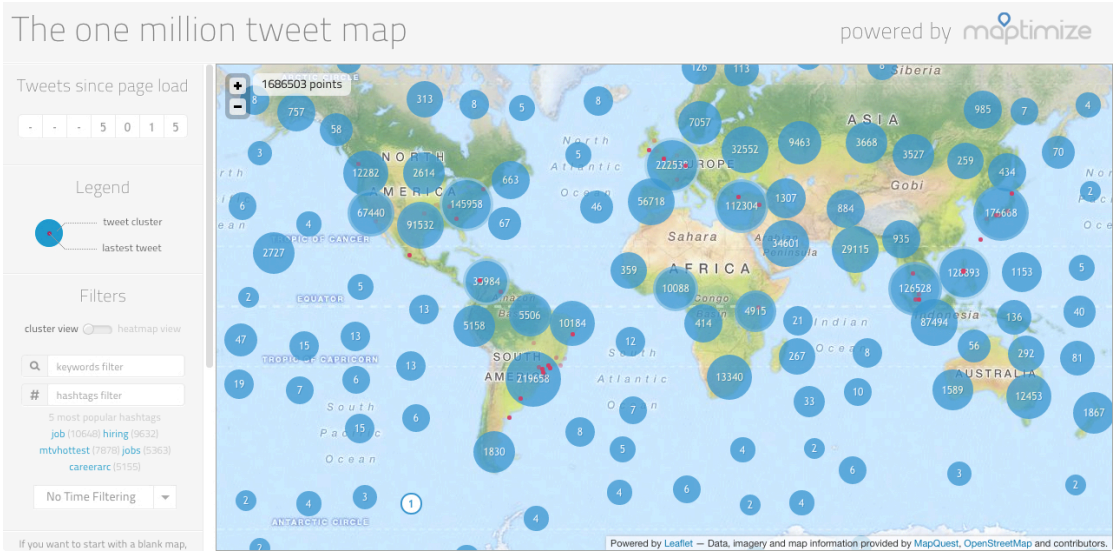
Αυτή η σελίδα είναι μια διαδικτυακή εφαρμογή που αφορά τη χαρτογράφηση της ροής του Twitter σε πραγματικό χρόνο με βάση τη γεωγραφική θέση. Ο χρήστης έχει τη δυνατότητα να δει τα γεωσυσχετισμένα tweets να απεικονίζονται στον χάρτη. Σε αυτόν εμφανίζονται ο αριθμός των συνολικών tweets από την έναρξη της εφαρμογής, οι χώρες με τα περισσότερα tweets και ο αριθμός τους. Επιπλέον ο χρήστης μπορεί να δει τα hashtags που έχουν χρησιμοποιηθεί τις περισσότερες φορές και το περιεχόμενο των tweets.



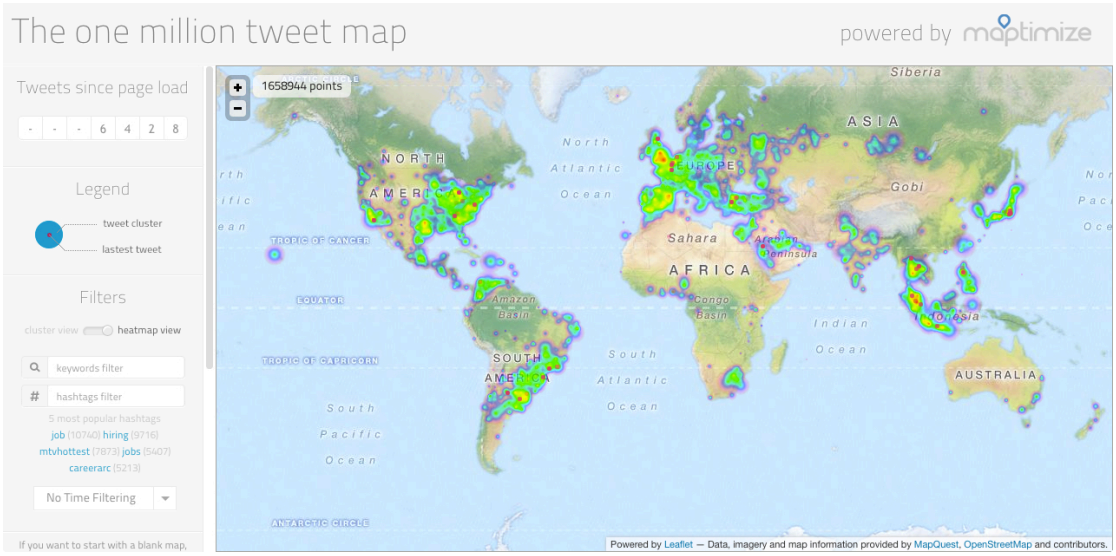
Εικόνα 2. Κεντρική σελίδα της εφαρμογής Tweetping.

http://onemilliontweetmap.com

Αυτή η σελίδα αντιστοιχίζει τα τελευταία γεωσυσχετισμένα tweets από το Twitter streaming API. Τα δεδομένα στον χάρτη ενημερώνονται σε πραγματικό χρόνο και διατηρούνται μόνο το ένα τελευταίο εκατομμύριο tweets. Κάθε δευτερόλεπτο, προστίθενται περίπου πενήντα νέα tweets. Υπάρχουν δύο διαφορετικές επιλογές εμφάνισης. Η πρώτη είναι σε μορφή ομαδοποιημένων tweets και η δεύτερη ένας χάρτης θερμότητας. Εμφανίζονται τα πιο διάσημα tweets και ο αριθμός που έχουν χτησιμοποιηθεί. Υπάρχουν δύο φίλτρα τα οποία μπορεί να χρησιμοποιήσει ο χρηστης για να περιορίσει τα αποτελέσματα. Το ένα είναι η επιλογή συγκεκριμένης λέξης ή hashtag και το δεύτερο είναι ο χρονικός περιορισμός των αποτελεσμάτων.



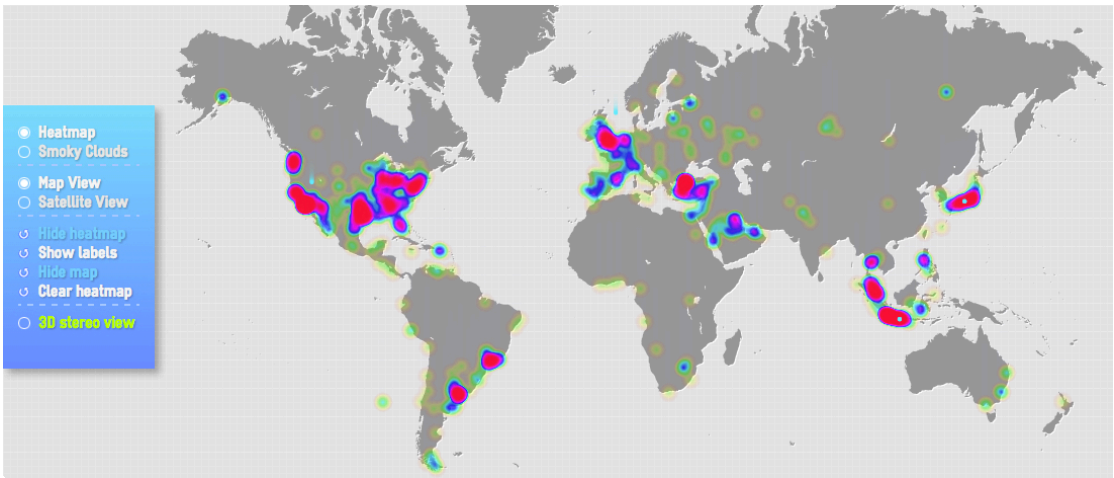
Εικόνα 3. Κεντρική σελίδα της εφαρμογής One million tweet map. Απεικόνιση με clusters.



Εικόνα 4. Κεντρική σελίδα της εφαρμογής «One million tweet map». Απεικόνιση με heatmap.

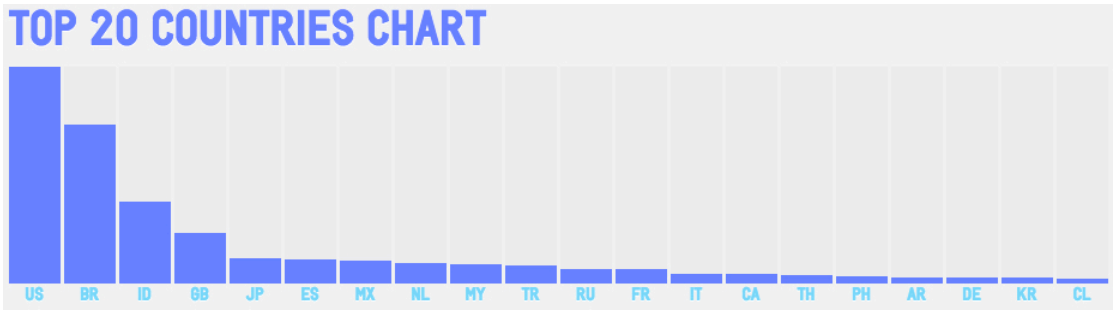
http://aworldoftweets.frogdesign.com

Η σελίδα αυτή απεικονίζει τα τελευταία γεωσυσχετισμένα tweets σε ένα χάρτη. Υπάρχει η δυνατότητα εμφάνισης των δεδομένων ως χάρτη θερμότητας ή σύνεφα καπνού. Ο χάρτης μπορεί να είναι σε κανονική μορφή (Map view) ή όψη δορυφόρου (Satellite view). Επιπρόσθετα η εφαρμογή εμφανίζει τα ποσοστά των πρώτων πέντε (5) χωρών προέλευσης των tweets για τη χρονική περίοδο από την έναρξη της εφαρμογής, για τη τρέχουσα μέρα αλλά και συνολικά από το Νοέμβριο του 2010. Ένα διάγραμμα απεικονίζει τις είκοσι (20) πρώτες χώρες προέλευσης και υπάρχουν στατιστικά στοιχεία για κάθε ήπειρο. Για τη συλλογή των δεδομένων έγινε χρήση του Twitter streaming API.

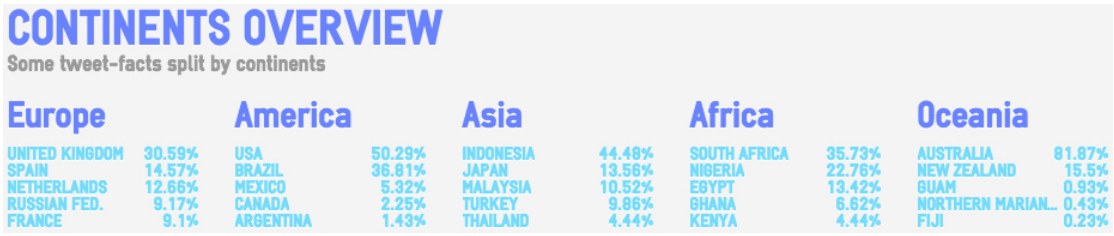


Εικόνα 5. Κεντρική σελίδα της εφαρμογής «A world of Tweets». Απεικόνιση με heatmap.

Εικόνα 6. Στατιστικά από την εφαρμογή «A world of Tweets».



Εικόνα 7. Στατιστικά από την εφαρμογή «A world of Tweets».

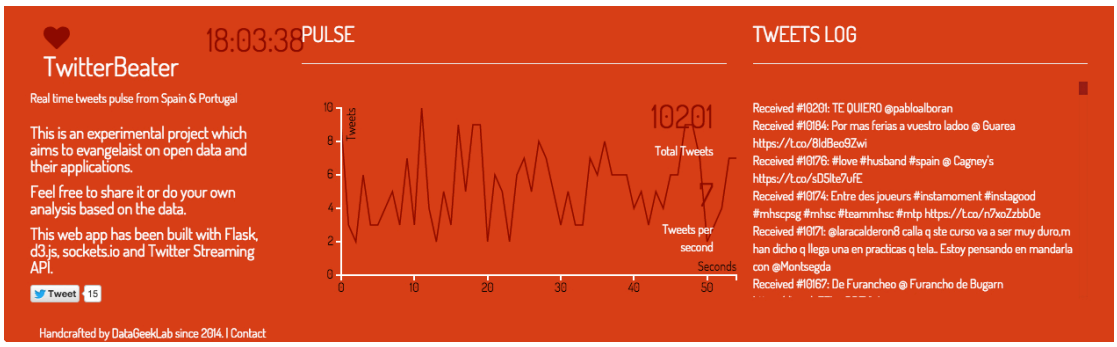


Εικόνα 8. Στατιστικά από την εφαρμογή «A world of Tweets».

http://twitterbeater.datageeklab.com

Η εφαρμογή αυτή συλλέγει tweets που γίνονται στη περιοχή της Ισπανίας και της Πορτογαλίας και τα εμφανίζει σε χάρτη. Υπάρχει διάγραμμα που δείχνει το πλήθος των tweets κάθε δευτερόλεπτο κ αθώς επίσης και εμφάνιση του κειμένου των tweets. Για τη συλλογή των δεδομένων χρησιμοποιήθηκε το Twitter streaming API, για τη δημιουργία του εξυπηρετητή το πλαίσιο διαδικτυακών εφαρμογών της Python που ονομάζεται Flask και τέλος για την ώθηση των δεδομένων από τον διακομιστή στον πελάτη χρήσιμοποιήθηκε το πρωτόκολλο WebSockets.

Εικόνα 9. Χάρτης που απεικονίζει τα tweets που γίνονται από την εφαρμογή «TwitterBeater».



Εικόνα 10. Διάγραμμα του ρυθμού των tweets και ιστορικό με το κείμενο της κάθε δημοσίευσης από την εφαρμογή «TwitterBeater».

2.4 Μετρικές

Παρακάτω γίνεται περιγραφή των μετρικών που έχουν χρησιμοποιηθεί στην εφαρμογή που υλοποιήθηκε στα πλαίσια της παρούσας πτυχιακής εργασίας.

2.4.1 Επιρροή χρήστη

Η κοινωνική επιρροή παρουσιάζεται όταν τα συναισθήματα, οι απόψεις, ή οι συμπεριφορές κάποιου επηρεάζονται από άλλους. Η κοινωνική επιρροή παίρνει πολλές μορφές και παίζει σημαντικό ρόλο σε τομείς όπως η κοινωνικοποίηση, η συμμόρφωση, η ηγεσία, η πειθώ και η αγοραλογία. Σύμφωνα με τον ψυχολόγο από το Χάρβαρντ, Χέρμπερτ Κέλμαν υπάρχουν τρεις μεγάλες κατηγορίες κοινωνικής επιρροής η συμμόρφωση, η αναγνώριση και η εσωτερίκευση. [11] Όταν ένας χρήστης μοιράζεται κάτι σε ένα κοινωνικό μέσο ή στην πραγματική ζωή και οι άνθρωποι ανταποκρίνονται, αυτό είναι επιρροή. Όσον αφορά το Twitter πρόκειται για μια μετρική που δείχνει την αποτελεσματικότητα και σημαντικότητα του γλωσσικού περιεχομένου ενός tweet. Η κανονικότητα, τα retweets και οι αναφορές είναι τα τρία ζωτικά σημεία της επιρροής ενός λογαριασμού Twitter. Η κανονικότητα σχετίζεται με το πόσο συχνά δημοσιεύονται tweets από ένα λογαριασμό και είναι ίσως η πιο βασική ένδειξη για την ισχύ του. Εάν δε δημιουργείται νέο περιεχόμενο τακτικά, οι ακόλουθοι μπορεί να αποχωρήσουν. Όσον αφορά τα retweets τα περισσότερα από αυτά συμβαίνουν μέσα σε λίγα λεπτά από τη στιγμή που δημοσιεύεται ένα tweet. Αλλά τα retweets είναι λιγοστά και σε κάποιο βαθμό αντανακλούν τον αριθμό των ακολούθων. Τέλος ένας

λογαριασμός Twitter μπορεί να κληθεί άμεσα χρησιμοποιώντας την αναφορά. Οι ωριαίες αναφορές συσχετίζονται έντονα με τα ωριαία retweets. Στη συγκεκριμένη εφαρμογή η επιρροή υπολογίζεται χρησιμοποιώντας τη διεπαφή προγραμματισμού εφαρμογών της Moz / Followerwonk και ονομάζεται social authority. Αυτή η βαθμολογία είναι μια κλίμακα από ένα έως εκατό και βασίζεται κυρίως στα retweets των πιο πρόσφατων tweets ενός χρήστη.

Ακολουθεί περιγραφή δύο μετρικών για την επιρροή ενός χρήστη στο Twitter.

Social authority

Μετρική από Moz / Followerwonk.

Είναι ένα μέτρο που δείχνει την επιρροή του περιεχομένου ενός χρήστη στο Twitter. Οι τιμές του είναι από ένα (1) έως εκατό (100). [10]

Το Social Authority εστιάζεται στο περιεχόμενο που δημιουργεί ο χρήστης, και πώς αντηχεί στο κοινό του. Αυτό είναι που ξεχωρίζει το Social Authority ως μετρική.

Το Social Authority υπολογίζεται με διαφανή τρόπο και περιλαμβάνει τρεις συνιστώσες [9]:

- ✓ Το ρυθμό των retweets των τελευταίων μερικών εκατοντάδων tweets που δεν έχουν αναφορά (@mention)
- ✓ Μια φθορά χρόνου για να ευνοείται η πιο πρόσφατη δραστηριότητα
- ✓ Άλλα δεδομένα του χρήστη όπως ο αριθμός των ακολούθων και των φίλων του τα οποία έχουν βελτιστοποιηθεί μέσω ενός μοντέλου παλινδρόμησης που έχει εκπαιδευτεί στο ρυθμό των retweets.

Ως βάση επιλέχθηκαν τα retweets καθώς το να μεταφέρει κάποιος το περιεχόμενο κάποιου άλλου στο δίκτυό του είναι μια αξιόλογη δραστηριότητα. Το γεγονός αυτό δείχνει μια σημαντική δέσμευση για τη προέλευση του περιεχομένου. [9]

Επιπλέον, τα retweets είναι ένας μεγάλος μεσολαβητής για άλλα σημαντικά στοιχεία. Για παράδειγμα, ο αριθμός των retweets που παίρνει ένας χρήστης συσχετίζεται έντονα με τον αριθμό των mentions που λαμβάνει. Τα retweets συσχετίζονται έντονα με τα κλικ. Ένας υψηλότερος ρυθμός retweet συνδέεται με μεγαλύτερη επισκεψιμότητα στις διευθύνσεις URL που υπάρχουν στα tweets. [9]

Influence Metric

Μετρική από Influence Tracker (influencetracker.com).

Το Influence Metric, όπως ονομάζεται η συγκεκριμένη μετρική που προσφέρεται από την εφαρμογή, είναι μια μέτρηση που έχει ως στόχο να περιγράψει τόσο τη σημασία όσο και τον αντίκτυπο ενός λογαριασμού σε ένα κοινωνικό δίκτυο. Χαμηλότερη τιμή του είναι μηδέν (0), ενώ δεν υπάρχει ανώτατο όριο . Όσο υψηλότερη είναι η τιμή, τόσο μεγαλύτερο αντίκτυπο έχει ένας λογαριασμός στο κοινωνικό δίκτυο.

Ένας παράγοντας για τον υπολογισμό της επιρροής είναι η αναλογία των ακολούθων προς τους φίλους (FtF). Ένας άλλος σημαντικός παράγοντας είναι ο ρυθμός δημιουργίας tweets (TCR). Για τον υπολογισμό αυτού του ρυθμού, γίνεται ανάλυση των τελευταίων 100 tweets

του χρήστη. Με αυτόν το τρόπο η μετρική είναι δυναμική καθώς εξαρτάται από την πιο πρόσφατη δραστηριότητα του κάθε χρήστη στο Twitter. [23] Πιο συγκεκριμένα γίνεται χρήση της παρακάτω εξίσωσης:

$$\text{Influence Metric} = \frac{tweets_k}{Days_{since\ k_{th}\ tweet}} * OOM(Followers) * \left(\frac{Followers}{Following} + 1\right),$$
$$όπου\ OOM = Order\ of\ Magnitude$$

2.4.2 Εμβέλεια γλωσσικού περιεχομένου

Ο όρος εμβέλεια μίας γλώσσας αναφέρεται στην μέση απόσταση της τοποθεσίας που γίνεται το κάθε tweet από τη πρωτεύουσα της χώρας που αντιστοιχεί στη γλώσσα του συγκεκριμένου tweet. Με αυτή τη μετρική μπορεί κάποιος να δει πόσο μακρυά από την κάθε πρωτεύουσα χρησιμοποιείται η κάθε γλώσσα για τη δημοσίευση των tweets.

3. Περιγραφή Συστήματος

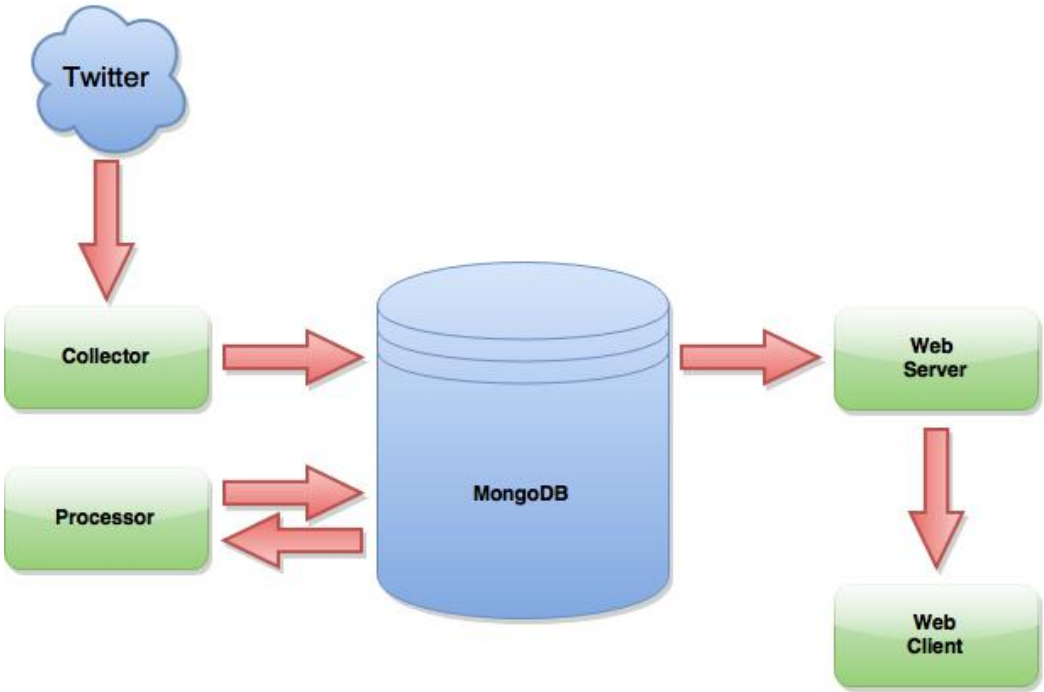
3.1 Εισαγωγή

Κύριος στόχος της παρούσας διπλωματικής είναι ο σχεδιασμός ενός συστήματος που θα συλλέγει γεωσυσχετισμένα δεδομένα από το Twitter σε πραγματικό χρόνο και θα παρέχει στο χρήστη μετρικές χρήσης, επιρροής και εμβέλειας γλωσσικού περιεχομένου. Αυτό το κεφάλαιο επικεντρώνεται στην αρχιτεκτονική του συστήματος αυτού καθώς και την περιγραφή όλης της διαδικασίας που ακολουθήθηκε. Τέλος θα γίνει αναφορά στα εργαλεία και στις τεχνολογίες που χρησιμοποιήθηκαν στην ανάπτυξη του συστήματος

3.2 Αρχιτεκτονική

Στα πλαίσια της αρχιτεκτονικής της εφαρμογής μας αποφασίστηκε η συλλογή των δεδομένων, η επεξεργασία και η παρουσίαση τους να είναι λειτουργίες διαχωρισμένες μεταξύ τους.

Η εφαρμογή αποτελείται από τέσσερις επιμέρους εφαρμογές. Η πρώτη είναι αυτή που συλλέγει τα δεδομένα από το Twitter. Η δεύτερη επεξεργάζεται τα δεδομένα που έχουν συλλεχθεί και κάνει κάποιους επιπλέον υπολογισμούς. Η τρίτη είναι ο εξυπηρετητής της διαδικτυακής εφαρμογής και τέλος, η τέταρτη αποτελεί τον πελάτη της εφαρμογής που παρουσιάζονται τα δεδομένα στο χρήστη.



Εικόνα 11. Αρχιτεκτονική του συστήματος

Στη συνέχεια θα ακολουθήσει πλήρη περιγραφή της αρχιτεκτονικής και των διαφόρων φάσεων της εφαρμογής.

3.2.1 Συλλογή δεδομένων

Το συγκεκριμένο πρόγραμμα έχει υλοποιηθεί με τη χρήση της γλώσσας προγραμματισμού python. Για τη συλλογή των δεδομένων χρησιμοποιήθηκε το Twitter Streaming API το οποίο δίνει πρόσβαση χαμηλής λανθάνουσας κατάστασης (low latency) στην παγκόσμια ροή δεδομένων του Twitter. Για την πρόσβαση στο API χρησιμοποιήθηκε η βιβλιοθήκη της python tweepy. Αρχικά το πρόγραμμα συνδέεται στον εξυπηρετητή του Twitter. Στη συνέχεια αποκτά πρόσβαση στη ροή δεδομένων και μέσω του καταληκτικού σημείου sample λαμβάνει ένα δείγμα των tweets που αντιστοιχεί περίπου στο 1% των δημόσιων tweets που γίνονται κάθε δευτερόλεπτο. Η εφαρμογή αναλύει τα δεδομένα από τη σύνδεση τα οποία είναι σε μορφή json και αποθηκεύει αυτά που χρειάζονται σε μία συλλογή της βάσης δεδομένων MongoDB. Για κάθε εισερχόμενο tweet αποθηκεύονται τα ακόλουθα στοιχεία: το αναγνωριστικό του tweet, η ώρα δημιουργίας του, το κείμενο, οι συντεταγμένες της τοποθεσίας που έγινε, τα hashtags που περιέχονται, η γλώσσα του κειμένου, το αναγνωριστικό του χρήστη που δημοσίευσε το συγκεκριμένο tweet και το όνομα που χρησιμοποιεί για την ταυτοποίηση του.

3.2.2 Επεξεργασία των δεδομένων

Η υλοποίηση του συγκεκριμένου προγραάμματος έγινε με τη χρήση της γλώσσας προγραμματισμού python. Αρχικά το πρόγραμμα διαβάζει τα δεδομένα που έχουν αποθηκευτεί στη βάση δεδομένων και εξάγει αυτά που χρειάζονται. Στη συνέχεια υπολογίζει τη γεωδαιτική απόσταση μεταξύ του σημείου που πραγματοποιήθηκε το κάθε tweet και του σημείου που βρίσκεται η πρωτεύουσα της χώρας που αντιστοιχεί στην γλώσσα που είναι γραμμένο το συγκεκριμένο tweet. Για τον υπολογισμό αυτό γίνεται χρήση της βιβλιοθήκης geopy και του τύπου Great-circle. Τα δεδομένα, έπειτα, προωθούνται σε μία ουρά όπου θα πραγματοποιηθεί ο υπολογισμός της επιρροής του χρήστη που δημοσίευσε το κάθε tweet ή αλλιώς "Social authority". Αυτό γίνεται με ερωτήματα που στέλνονται στο Social Authority API των Followerwonk - moz. Τέλος τα δεδομένα αποθηκεύονται εκ νέου στη MongoDB.

Η βάση δεδομένων που χρησιμοποιήθηκε για την αποθήκευση των δεδομένων είναι η MongoDB. Βάσεις δεδομένων όπως η MongoDB είναι ιδανικές για την αποθήκευση των απαντήσων από το Twitter API, καθώς έχουν σχεδιαστεί για την αποτελεσματική αποθήκευση δεδομένων JSON. Η MongoDB είναι μια ισχυρή και καλά τεκμηριωμένη βάση δεδομένων που λειτουργεί καλά για μικρές ή μεγάλες ποσότητες δεδομένων. Για τις διάφορες αλληλεπιδράσεις με τη βάση χρησιμοποιήθηκε η βιβλιοθήκη της python PyMongo.

Για την επικοινωνία του προγράμματος που συλλέγει τα δεδομένα και αυτού που τα επεξεργάζεται δημιουργήθηκε ένα σύστημα δημοσίευσης / εγγραφής. Αν και υπάρχουν πολλά συστήματα ανταλλαγής μηνυμάτων όπως το Redis, το AMQP και το ØMQ, επιλέχθηκε η MongoDB καθώς είναι μια πολύ ελκυστική εναλλακτική λύση. Χρησιμοποιώντας τις συλλογές ανώτατου ορίου της MongoDB και τους tailable cursors δημιουργήθηκε ένα απλό pub/sub σύστημα για να μπορούν να επικοινωνούν τα δεδομένα

μεταξύ των δύο προγραμμάτων. Οι συλλογές ανώτατου ορίου (Capped Collections) είναι ένα ωραίο χαρακτηριστικό της MongoDB που επιτρέπει τη δημιουργία μιας υψηλής απόδοσης κυκλικής ουράς. Για αρχή δημιουργήθηκε μια συλλογή ανώτατου ορίου στην οποία αποθηκεύονται τα δεδομένα. Το πρόγραμμα συλλογής δεδομένων αποθηκεύει τα δεδομένα στη συλλογή δηλαδή κάνει δημοσίευση ενός μηνύματος και το πρόγραμμα επεξεργασίας κάνει εγγραφή στη συλλογή ώστε να μπορεί να λαμβάνει κάθε νέο έγγραφο που δημοσιεύεται.

Το παρακάτω αποτελεί ένα παράδειγμα των δεδομένων που αποθηκεύονται στη βάση δεδομένων.


```
{
   "_id" : ObjectId("55e0fb735ef6db0c5d938da2"),
   "t_id" : "637420202632871937",
   "t_uname" : "kianajayleen",
   "t_geo" : [ -79.875221, 36.033444 ],
   "c_distance" : 6299.011955759781,
   "t_hashtags" : [ ],
   "t_lang" : "en",
   "t_date" : "Sat Aug 29 00:23:13 +0000 2015",
   "c_fwsa" : 18.74,
   "t_uid" : "3069326656",
   "t_text" : "I always go whenever I am bored too."
}
```


Το *_id είναι το αναγνωριστικό της συγκεκριμένης εγγραφής που δημιουργείται από μόνο του, τα κλειδιά που έχουν το πρόθεμα t_ είναι τα δεδομένα που λαμβάνονται από το Twitter και τέλος αυτά που έχουν το πρόθεμα c_ είναι οι επιπλέον υπολογισμοί που γίνονται.*

3.3.3 Εξυπηρετητής

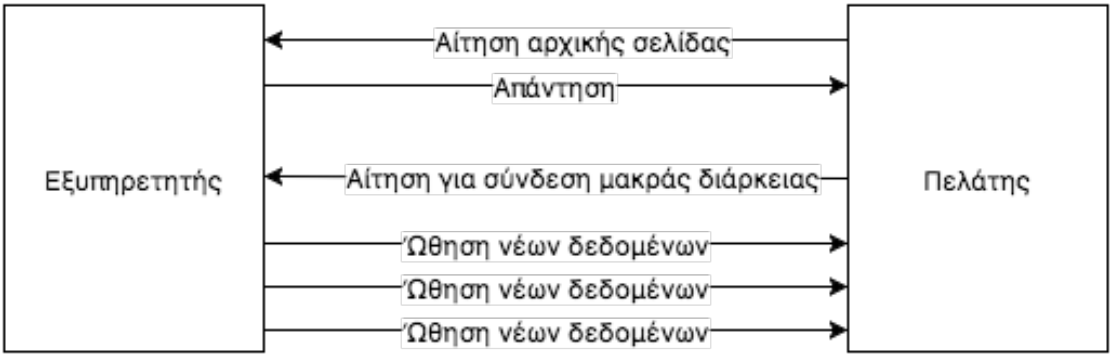
Ο εξυπηρετητής(web server) για την εφαρμογή έχει υλοποιηθεί με τη χρήση της γλώσσας προγραμματισμού python και το πλαίσιο διαδικτυακών εφαρμογών Tornado. Η βιβλιοθήκη Motor χρησιμοποιήθηκε για την αλληλεπίδραση με τη βάση καθώς δίνει τη δυνατότητα για ασύγχρονες λειτουργίες. Υπάρχει ένα αντικείμενο Application που με τον πίνακα δρομολόγησης κατευθύνει τις εισερχόμενες αιτήσεις στους χειριστές αιτήσεων. Στην εφαρμογή δημιουργήθηκαν δύο χειριστές. Ο ένας αφορά την ιστοσελίδα της διαδικτυακής εφαρμογής και ο δεύτερος τα δεδομένα τα οποία στέλνονται στην ιστοσελίδα. Για τον πρώτο χειριστή έχει δημιουργηθεί μία μέθοδος που φορτώνει το πρότυπο της ιστοσελίδας που έχει δημιουργηθεί. Μόλις η μέθοδος επιστρέψει η αίτηση θεωρείται τελειωμένη και αποστέλλεται η α πάντηση στον πελάτη. Ο δεύτερος χειριστής είναι πιο περίπλοκος. Σε αντίθεση με τον προηγούμενο χειριστή αυτός πρέπει να είναι ασύγχρονος ώστε να μη προκαλεί διακοπές στο υπόλοιπο σύστημα όταν εκτελεί τις αργές λειτουργίες του. Η εφαρμογή είναι πραγματικού χρόνου και επομένως απαιτεί μια μακράς διάρκειας και ως επί το πλείστον αδρανής σύνδεση ανά χρήστη. Γίνεται χρήση λοιπόν λειτουργιών Εισόδου/Εξόδου χωρίς αναμονή (non blocking I/O) ή αλλιώς ασύγχρονης Ε/Ε (asynchronous I/O) με σκοπό να αυξηθεί η απόδοση και η δυνατότητα κλιμάκωσης της εφαρμογής. Όταν ο

πελάτης κάνει ένα ερώτημα, δημιουργείται μία σύνδεση που παραμένει ενεργή μέχρι να τερματιστεί η εφαρμογή του πελάτη. Το πρόγραμμα εγγράφεται σε μια συλλογή ανωτάτου ορίου της MongoDB και γίνεται χρήση ένος pub/sub συστήματος που περιγράφηκε παραπάνω. Κάθε φορά που δημοσιεύονται νέα δεδομένα στη συλλογή, αυτά με τη σειρά τους διαβάζονται και αποστέλλονται στον πελάτη.

Λίγα λόγια για τον τρόπο επικοινωνίας του εξυπηρετητή με τον εξυπηρετούμενο. Το HTTP πρωτόκολλο δεν είναι ιδανικό για εφαρμογές πραγματικού χρόνου επειδή είναι ένα πρωτόκολλο χωρίς πληροφορίες κατάστασης. Αυτό σημαίνει ότι ενώ ο εξυπηρετούμενος μπορεί να στέλνει HTTP αιτήματα στον εξυπηρετητή και ο εξυπηρετητής να απαντάει σε αυτά, το ανάποδο δεν γίνεται. Αν ένα γεγονός συμβεί στον εξυπηρετητή, αυτός δεν έχει κάποιο τρόπο να ενημερώσει τον εξυπηρετούμενο για την αλλαγή της κατάστασης του σε πραγματικό χρόνο. Πρέπει να περιμένει να λάβει ένα αίτημα από τον εξυπηρετούμενο ώστε να ενημερώσει για την αλλαγή. Για να λυθεί αυτό το πρόβλημα ο εξυπηρετούμενος πρέπει να χρησιμοποιεί περιοδικές ερωτήσεις (polling). Πρέπει δηλαδή να ρωτάει συνεχώς τον εξυπηρετητή για την κατάστασή του. Με αυτόν τον τρόπο μπορούμε να εξομοιώσουμε την επικοινωνία πραγματικού χρόνου. Αυτή η λύση όμως δεν είναι καθόλου αποδοτική καθώς ο εξυπηρετητής πρέπει να χειριστεί πολλά εισερχόμενα αιτήματα, ακόμα και αν δεν έχει συμβεί κάποια αλλαγή στην κατάσταση του. Μια πιο αποδοτική λύση είναι η τεχνική μακράς περιόδευσης (long polling). Σε αυτήν την τεχνική ο εξυπηρετούμενος στέλνει μόνο ένα αίτημα στον εξυπηρετητή και ο τελευταίος διατηρεί την σύνδεση ανοικτή μέχρι να παρουσιαστεί κάποια αλλαγή στην κατάσταση του. Αυτοί οι περιορισμοί παρακίνησαν την ανάπτυξη πρωτοκόλλων όπως τα Server-Sent Events και τα WebSockets. Στην παρούσα εφαρμογή έγινε χρήση του πρώτου λόγω της απλότητας του.

3.3.4 Εξυπηρετούμενος

Ο πελάτης(client) της διαδικτυακής εφαρμογής έχει υλοποιηθεί με τη χρήση της HTML5 καθώς επίσης και τεχνολογιών όπως CSS3, javascipt και jQuery. Η εφαρμογή αρχικά κάνει αίτημα στον εξυπηρετητή για την αρχική σελίδα. Μόλις ολοκληρωθεί η φόρτωση της σελίδας γίνεται αίτημα στον εξυπηρετητή για τη δημιουργία μίας σύνδεσης μακράς διάρκειας και αρχίζει να δέχεται δεδομένα σε πραγματικό χρόνο με τη χρήση των Server-Sent Events. Καθώς λαμβάνονται τα δεδομένα από τον εξυπηρετητή αποθηκεύονται σε αντικείμενα της Javascript. Τα δεδομένα απεικονίζονται αυτόματα σε χάρτη χρησιμοποιώντας το Google Maps API και μόνο για την αλλαγή των προτιμήσεων είναι απαραίτητη η αλληλεπίδραση του χρήστη.



Εικόνα 12. Επικοινωνία πελάτη-εξυπηρετητή

3.3 Τεχνολογίες

Μεριά Εξυπηρετητή

~okeanos

Το ~okeanos (https://okeanos.grnet.gr) είναι μία υπηρεσία IaaS (Infrastracture as a Service). Έχει σχεδιαστεί και αναπτυχθεί από το Ελληνικό Δίκτυο Έρευνας και Τεχνολογίας (ΕΔΕΤ), προς όφελος της ελληνικής ακαδημαϊκής κοινότητας. Δίνει τη δυνατότητα σε κάποιον να φτιάξει τον δικό του υπολογιστή, που θα είναι πάντα συνδεδεμένος στο διαδίκτυο, χωρίς να ανησυχεί για πιθανόν αποτυχίες του υλικού, τη συνδεσιμότητα και προβλήματα λογισμικού. Στην ουσία με το ~okeanos η πιο συγκεκριμένα με το Cyclades μπορεί κάποιος να αποκτήσει εικονικές μηχανές (Virtual Machines) και εικονικά δίκτυα.

Linux

Το Linux είναι ένα λειτουργικό σύστημα υπολογιστών παρόμοιο με το Unix και συμβατό με το POSIX που έχει συναρμολογηθεί σύμφωνα με το μοντέλο της ανάπτυξης ελεύθερου και ανοιχτού κώδικα λογισμικού και διανομής. [4]

Ubuntu

Το Ubuntu είναι ένα ανοικτού κώδικα, ελεύθερο και δωρεάν λειτουργικό σύστημα βασισμένο στον πυρήνα Linux. [18]

Το λειτουργικό σύστημα Ubuntu είναι αυτό που επιλέχθηκε για τον υπολογιστή που χρησιμοποιήθηκε για την εγκατάσταση του συστήματος.

Python

Η Python είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού.

Κύριο χαρακτηριστικό της είναι η ευκολία χρήσης της καθώς το συντακτικό της επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα απ'ότι θα ήταν δυνατόν σε γλώσσες όπως η C++ ή η Java. [8]

Η Python υποστηρίζει πολλαπλά πρότυπα προγραμματισμού, συμπεριλαμβανομένων των αντικειμενοστραφή, προστακτικό και λειτουργικό προγραμματισμό ή διαδικαστικά στυλ. Τέλος διαθέτει ένα δυναμικό σύστημα τύπων, αυτόματη διαχείριση μνήμης και έχει μία μεγάλη και ολοκληρωμένη πρότυπη βιβλιοθήκη. [8]

Η python χρησιμοποιήθηκε για την υλοποίηση των μερών του εξυπηρετητή του συστήματος.

Μη-σχεσιακές (NoSQL) βάσεις δεδομένων

Μια NoSQL βάση δεδομένων παρέχει ένα μηχανισμό για την αποθήκευση και την ανάκτηση δεδομένων που διαμορφώνονται με τρόπο διαφορετικό από τις σχέσεις πινάκων που χρησιμοποιούνται σε σχεσιακές βάσεις δεδομένων.

Στα κίνητρα για την προσέγγιση αυτή περιλαμβάνονται η απλότητα του σχεδιασμού, η οριζόντια κλιμάκωση, και ο καλύτερος έλεγχος της διαθεσιμότητας. Τα NoSQL συστήματα έχουν φτιαχτεί έτσι ώστε να μπορούν να διαχειριστούν μεγάλες ποσότητες δεδομένων χωρίς κατ'ανάγκη να διατηρούν μία συγκεκριμένη δομή. Οι δομές δεδομένων που χρησιμοποιούνται από NoSQL βάσεις δεδομένων διαφέρουν από εκείνες που χρησιμοποιούνται σε σχεσιακές β άσεις δεδομένων. Μερικοί τύποι τέτοιων βάσεων δεδομένων είναι στήλη, έγγραφο, κλειδί-τιμή και γράφημα. [7]

Οι NoSQL βάσεις δεδομένων χρησιμοποιούνται όλο και περισσότερο σε διαδικτυακές εφαρμογές μεγάλων δεδομένων σε πραγματικό χρόνο. Τα συστήματα NoSQL ονομάζονται επίσης "Όχι μόνο SQL" (Not Only SQL) για να τονιστεί ότι μπορούν επίσης να υποστηρίξουν γλώσσες επερωτήσεων παρόμοιες με SQL. [7]

MongoDB

Η MongoDB είναι μία ανεξαρτήτου πλατφόρμας βάση δεδομένων προσανατολισμένη σε έγγραφα. Είναι ένας ισχυρός, γρήγορος, ευέλικτος, επεκτάσιμος και κλιμακούμενος χώρος αποθήκευσης δεδομένων. [20] Καθώς ανήκει στις NoSQL βάσεις δεδομένων, αποφεύγει την παραδοσιακή δομή σχεσιακών βάσεων δεδομένων που βασίζονται σε πίνακες. Αντιθέτως, βασίζεται σε έγγραφα μορφής JSON με δυναμικά σχήματα, καθιστώντας την ενσωμάτωση των δεδομένων σε ορισμένους τύπους εφαρμογών ευκολότερη και ταχύτερη. Η MongoDB είναι ελεύθερο λογισμικό και ανοικτού κώδικα. Είναι απίστευτα πλούσια σε χαρακτηρικά, μερικά από τα οποία είναι τα ακόλουθα [5]:

- ✓ Προσανατολισμένη σε έγγραφα
  Μια εγγραφή της MongoDB είναι ένα έγγραφο, το οποίο είναι μια δομή δεδομένων που αποτελείται από ζεύγη πεδίου και τιμής. Τα έγγραφα της MongoDB είναι παρόμοια με τα αντικείμενα JSON. Οι τιμές των πεδίων μπορούν να περιλαμβάνουν και άλλα έγγραφα, πίνακες, και πίνακες εγγράφων. Η προσέγγιση του προσανατολισμού σε έγγραφα καθιστά δυνατή την αντιπροσώπευση πολύπλοκων ιεραρχικών σχέσεων με μία ενιαία εγγραφή. Αυτό ταιριάζει πολύ φυσικά με τρόπο που οι προγραμματιστές στις σύγχρονες αντικειμενοστρεφείς γλώσσες σκέφτονται για τα δεδομένα τους.

- ✓ Ad hoc ερωτήματα
  Η MongoDB υποστηρίζει αναζήτηση σύμφωνα με το πεδίο, ερωτήματα εύρους και αναζητήσεις κανονικών εκφράσεων. Τα ερωτήματα μπορούν να επιστρέψουν συγκεκριμένα τμήματα των εγγράφων και επίσης περιλαμβάνουν λειτουργίες JavaScript που ορίζονται από το χρήστη.

- ✓ Δημιουργία ευρετηρίου
  Οποιοδήποτε πεδίο σε ένα έγγραφο MongoDB μπορεί να καταλογοποιηθεί. Δευτερεύοντα ευρετήρια είναι επίσης διαθέσιμα.

- ✓ Δημιουργία αντιγράφων
  Η MongoDB παρέχει υψηλή διαθεσιμότητα με σύνολα αντιγράφων. Υπάρχουν τα κύρια και τα δευτερεύοντα αντίγραφα. Το κύριο αντίγραφο εκτελεί όλες τις εγγραφές και τις αναγνώσεις. Τα δευτερεύοντα αντίγραφα χρησιμοποιούνται όταν το κύριο αντίγραφο αποτύχει.

- ✓ Εξισορρόπηση φόρτου εργασίας
  Η MongoDB κλιμακώνεται οριζόντια χρησιμοποιώντας sharding. Ο χρήστης επιλέγει ένα κλειδί, το οποίο καθορίζει το πώς θα κατανεμηθούν τα δεδομένα σε μια συλλογή. Τα δεδομένα χωρίζονται σε εύρη (με βάση το κλειδί) και διανέμεται σε πολλαπλά shards. Η MongoDB μπορεί να τρέξει πάνω σε πολλούς εξυπηρετητές, εξισορροπώντας το φόρτο εργασίας και/ή να αντιγράφει δεδομένα για να κρατήσει το σύστημα σε λειτουργία σε περίπτωση αποτυχίας του υλικού. Νέοι εξυπηρετητές μπορούν να προστεθούν σε μια βάση δεδομένων που είναι ήδη σε λειτουργία.

- ✓ Αποθήκευση αρχείων
  Η MongoDB μπορεί να χρησιμοποιηθεί ως ένα σύστημα αρχείων, εκμεταλλευόμενη την εξισορρόπηση φόρτου εργασίας και των χαρακτηριστικών της αντιγραφής δεδομένων σε πολλαπλούς υπολογιστές για την αποθήκευση αρχείων. Αυτή η λειτουργία ονομάζεται GridFS. Αντί για την αποθήκευση ενός αρχείου σε ένα ενιαίο έγγραφο, το GridFS χωρίζει ένα αρχείο σε τμήματα, ή κομμάτια, και αποθηκεύει κάθε ένα από αυτά τα κομμάτια ως ξεχωριστό έγγραφο. Σε ένα σύστημα MongoDB με πολλά μηχανήματα, τα αρχεία μπορούν να διανεμηθούν και να αντιγραφούν πολλές φορές μεταξύ των υπολογιστών με διαφάνεια, καταλήγοντας αποτελεσματικά στη δημιουργία ενός ισορροπημένου φόρτου εργασίας και ανεκτικό σε σφάλματα συστήματος.

- ✓ Συσσωμάτωση
  Η MapReduce μπορεί να χρησιμοποιηθεί για ομαδοποιημένη επεξεργασία δεδομένων και συσσωμάτωση λειτουργιών. Το πλαίσιο συσσωμάτωσης επιτρέπει στους χρήστες να αποκτήσουν το είδος των αποτελεσμάτων για το οποίο χρησιμοποιείται η SQL συνθήκη GROUP BY.

- ✓ Εκτέλεση Javascript στη μεριά του εξυπηρετητή
  Η JavaScript μπορεί να χρησιμοποιηθεί σε ερωτήματα, συναρτήσεις συσσωμάτωσης, και να σταλθεί απευθείας στη βάση δεδομένων για να εκτελεστεί.

- ✓ Συλλογές ανωτάτου ορίου
  Η MongoDB υποστηρίζει συλλογές σταθερού μεγέθους που ονομάζονται capped collections. Αυτό το είδος της συλλογής διατηρεί τη σειρά εισαγωγής και μόλις επιτευχθεί το καθορισμένο μέγεθος συμπεριφέρεται σαν μια κυκλική ουρά.

Η MongoDB χρησιμοποιήθηκε για τη δημιουργία ενός συστήματος δημοσίευσης / εγγραφής κάνοντας χρήση των συλλογών ανωτάτου ορίου.

Tornado

Το Tornado είναι ένα πλαίσιο διαδικτυακών εφαρμογών της Python και μία ασύγχρονη βιβλιοθήκη δικτύωσης. Με τη χρήση λειτουργιών εισόδου / εξόδου του δικτύου χωρίς

διακοπές, το Tornado μπορεί να κλιμακωθεί σε δεκάδες χ ιλιάδες ενεργές συνδέσεις, καθιστώντας το ιδανικό για long polling, WebSockets, και άλλες εφαρμογές που απαιτούν μια μακράς διάρκειας ζωής σύνδεση σε κάθε χρήστη. Ο εξυπηρετητής υψηλών επιδόσεων είναι γνωστός για την ταχύτητά του, την απλότητα και την επεκτασιμότητα. Είναι αρκετά ισχυρός ώστε να μπορεί να χειριστεί σοβαρή κυκλοφορία διαδικτύου, αλλά είναι ελαφρύς για να ρυθμιστεί και να γραφτεί και μπορεί να χρησιμοποιηθεί για πολλές και διαφορετικές εφαρμογές και βοηθητικά προγράμματα. [14]

Το Tornado μπορεί να διαιρεθεί σε τέσσερα βασικά μέρη [14]:

- ✓ Ένα πλαίσιο διαδικτύου για τη δημιουργία διαδικτυακών εφαρμογών.
- ✓ Υλοποιήσεις του HTTP πρωτοκόλλου τόσο από τη μεριά του Πελάτη αλλά και του Εξυπηρετητή.
- ✓ Μια ασύγχρονη βιβλιοθήκη δικτύωσης που περιλαμβάνει τα δομικά στοιχεία για τα μέρη του HTTP και μπορεί επίσης να χρησιμοποιηθεί για την υλοποίηση άλλων πρωτοκόλλων.
- ✓ Μια βιβλιοθήκη coroutine η οποία επιτρέπει το γράψιμο ασύγχρονου κώδικα με πιο άμεσο τρόπο από ό,τι με αλυσιδωτή σύνδεση με callbacks.

Το Tornado χρησιμοποιήθηκε για την υλοποίηση του διακομιστή.

Tweepy

Μία εύκολη στη χρήση βιβλιοθήκη της Python για την πρόσβαση στο Twitter API. Επιτρέπει στη python να επικοινωνήσει με τη πλατφόρμα του Twitter και να χρησιμοποιήσει το API του.

Motor

(MOngoDB + TORnado)

Είναι ένας ασύγχρονος οδηγός της python για το Tornado και τη MongoDB. Χρησιμοποιείται για την ασύγχρονη λειτουργία του διακομιστή.

OAuth

Το Twitter χρησιμοποιεί το OAuth για να παρέχει εξουσιοδοτημένη πρόσβαση στο API του. Πρόκειται για ένα ανοιχτό πρωτόκολλο που επιτρέπει την ασφαλή χορήγηση άδειας από εφαρμογές με μια απλή και τυποποιημένη μέθοδο.

Μοντέλο ελέγχου ταυτότητας (API v1.1)
Υπάρχουν δύο μορφές ελέγχου ταυτότητας που κάνουν χρήση του OAuth 1.0A.

- ✓ **Έλεγχος ταυτότητας εφαρμογής-χρήστη**
  Αυτή είναι η πιο κοινή μορφή ελέγχου ταυτότητας. Η υπογεγραμμένη αίτηση αναγνωρίζει την ταυτότητά της εφαρμογής και την ταυτότητα του χρήστη. Έτσι

χορηγούνται τα δικαιώματα στην εφαρμογή να κάνει κλήσεις στο API για λογαριασμό του χρήστη, εκπροσωπούμενη από το διακριτικό πρόσβασης του.

- ✓ **Έλεγχος ταυτότητας μόνο για την εφαρμογή**
  Ο έλεγχος ταυτότητας μόνο για την εφαρμογή είναι μια μορφή ελέγχου ταυτότητας όπου η εφαρμογή κάνει αιτήματα στο API για δικό της λογαριασμό, χωρίς την εμπλοκή κάποιου χρήστη. Οι κλήσεις στο API εξακολουθούν να έχουν περιορισμούς, αλλά ισχύουν για ολόκληρη την εφαρμογή, και όχι για κάθε χρήστη.

Η εφαρμογή που υλοποιήθηκε κάνει χρήση της δεύτερης μορφής ελέγχου ταυτότητας.

geopy

Το geopy είναι μία βιβλιοθήκη της Python για διάφορες δημοφιλείς διαδικτυακές υπηρεσίες γεωκωδικοποίησης. Δίνει τη δυνατότητα υπολογισμού γεωδαιτικής απόστασης μεταξύ δύο σημείων με τη χρήση του τύπου Vincenty ή Great-circle. Η απόσταση Great-circle χρησιμοποιεί ένα σφαιρικό μοντέλο της γης, χρησιμοποιώντας τη μέση ακτίνα των 6.372,795 χιλιόμετρων, με αποτέλεσμα ένα σφάλμα έως περίπου 0,5%. Η απόσταση Vincenty χρησιμοποιεί ένα πιο ακριβές ελλειψοειδές μοντέλο της γης. [1]

Υπάρχουν πολλά δημοφιλή ελλειψοειδή μοντέλα, και το ποιο θα είναι το πιο ακριβές εξαρτάται από το που είναι τα σημεία που βρίσκονται στη γη. Ο τύπος του Great-circle είναι οριακά λιγότερο ακριβής, αλλά παράγει πάντα ένα αποτέλεσμα και για αυτό το λόγο επιλέχθηκε να χρησιμοποιηθεί στην εφαρμογή.

Μεριά Εξυπηρετούμενου

HTML5 Server-Sent Events (SSEs)

Τα SSEs είναι μια HTML5 τεχνολογία που επιτρέπει στον εξυπηρετητή να ωθεί νέα δεδομένα στους πελάτες. Είναι μια καλύτερη λύση από το να κάνει ο πελάτης έλξη (polling) νέων δεδομένων κάθε λίγα δευτερόλεπτα ή να απαιτείται αλληλεπίδραση με τον χρήστη. Η λειτουργικότητα της έλξης δεδομένων και της ώθησης δεδομένων είναι η ίδια αλλά η ώθηση δεδομένων έχει κάποια πλεονεκτήματα. Το μεγαλύτερο πλεονέκτημα είναι η χαμηλότερη λανθάνουσα κατάσταση. Τα SSEs χρησιμοποιούνται όταν η εφαρμογή απαιτεί χαμηλή λανθάνουσα κατάσταση και εξοικονόμηση εύρους ζώνης. Ένα άλλο βασικό χαρακτηριστικό των SSEs είναι η απλότητα χρήσης. Ο κώδικας για το κομμάτι του εξυπηρετητή μπορεί να είναι μόλις λίγες γραμμές. Τέλος, επειδή τα SSEs λειτουργούν μέσω των υπάρχοντων HTTP / HTTPS πρωτοκόλλων, λειτουργούν με τους υπάρχοντες διακομιστές μεσολάβησης και τις υπάρχουσες τεχνικές ελέγχου ταυτότητας. Είναι σημαντικό να γίνει αναφορά και σε μια άλλη τεχνολογία που είναι παρόμοια και είναι τα WebSockets. Είναι μια πιο περίπλοκη τεχνολογία για να υλοποιηθεί στην πλευρά του διακομιστή και συνήθως χρειάζεται μια βοηθητική βιβλιοθήκη. Σε αντίθεση με τα SSEs υπάρχει αμφίδρομη επικοινωνία, δηλαδή ο διακομιστής μπορεί να ωθήσει τα δεδομένα στον πελάτη και ο πελάτης μπορεί να ωθήσει τα δεδομένα πίσω στο διακομιστή. Ένα πλεονέκτημα για τα WebSockets σε σχέση με τα SSEs είναι ένα ότι είναι ένα δυαδικό πρωτόκολλο, ενώ τα SSEs χρησιμοποιούν UTF-8. Τα δυαδικά δεδομένα πρόκειται να είναι μεγαλύτερα όταν αποστέλλονται μέσω SSEs. Αν

απαιτείται αποστολή μεγάλων ποσοτήτων δυαδικών δεδομένων από το διακομιστή στον πελάτη, τότε τα WebSockets είναι η καλύτερη επιλογή. Τέλος, όπως και τα SSEs χρησιμοποιεί μια υποδοχή TCP/IP και είναι ελαφρύ πρωτόκολλο. [21]

Τα Server-Sent Events χρησιμοποιήθηκαν για την αποστολή των δεδομένων από τον εξυπηρετητή στον πελάτη.

HTML

Η HTML (HyperText Markup Language) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Χρησιμοποιείται για την περιγραφή της δομής των ιστοσελίδων.

CSS

Η CSS (Cascading Style Sheets) είναι μια γλώσσα που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης μέσω της στιλιστικής του ανάπτυξης.

Javascript

Η Javascript είναι μια δυναμική γ λώσσα σεναρίων που υποστηρίζει αντικειμενοστραφή, προστακτικό κ αι συναρτησιακό προγραμματισμό. Χρησιμοποιείται κυρίως για προγραμματισμό στην πλευρά του εξυπηρετούμενου καθώς τρέχει πάνω σε φυλλομετρητές (browsers).

jQuery

Η jQuery είναι μια βιβλιοθήκη της JavaScript σχεδιασμένη να απλοποιήσει την υλοποίηση σεναρίων στη πλευρά του πελάτη.

Bootstrap

Το Bootstrap είναι μια συλλογή εργαλείων ανοιχτού κώδικα για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών.

Google Maps API

Οι χάρτες Google (Google Maps) είναι μία υπηρεσία χαρτογράφησης στο Διαδίκτυο. Χρησιμοποιείται για να δει κάποιος την τοποθεσία διαφόρων πραγμάτων, την αναζήτηση της θέσης μιας διεύθυνσης, τη λήψη οδηγιών κατεύθυνσης, και πολλά άλλα πράγματα. Προσφέρεται ένα API που επιτρέπει στους χάρτες να ενσωματωθούν σε ιστοσελίδες τρίτων και να προβάλλουν δεδομένα με αποτελεσματικό και χρήσιμο τρόπο.

Το Google Maps API αποτελείται από κώδικα HTML, CSS και JavaScript. Τα πλακίδια του χάρτη είναι εικόνες που έχουν φορτωθεί στο παρασκήνιο με κλήσεις Ajax και στη συνέχεια εισάγεται στη σελίδα HTML. Το API αποτελείται από αρχεία JavaScript που περιέχουν κλάσεις με μεθόδους και ιδιότητες που μπορούν να χρησιμοποιήθούν από τους προγραμματιστές. [25]

Οι συντεταγμένες χρησιμοποιούνται για να εκφραστούν οι τοποθεσίες στον κόσμο. Υπάρχουν πολλά διαφορετικά συστήματα συντεταγμένων. Αυτό που χρησιμοποιείται στο Google Maps είναι το Παγκόσμιο Γεωδαιτικό Σύστημα 84 (World Geodetic System ή WGS 84), το οποίο είναι το ίδιο σύστημα που χρησιμοποιεί το Παγκόσμιο Σύστημα Στιγματοθέτησης (Global Positioning System ή GPS). Οι συντεταγμένες εκφράζονται χρησιμοποιώντας γεωγραφικό πλάτος (latitude) και μήκος (longitude). [25]

Το γεωγραφικό πλάτος μετρά από το νότο στο βορρά, και το γεωγραφικό μήκος από τα δυτικά προς τα ανατολικά. Στον ισημερινό, το γεωγραφικό πλάτος είναι 0. Αυτό σημαίνει ότι οτιδήποτε κάτω από τον ισημερινό (νότιο ημισφαίριο) έχει αρνητικό αριθμό, και οτιδήποτε πάνω από αυτό (βόρειο ημισφαίριο) έχει θετικό αριθμό. Ομοίως, υπάρχει μία μηδενική γραμμή για το γεωγραφικό μήκος. Ονομάζεται πρώτος μεσημβρινός (prime meridian), και διατρέχει από το Γκρήνουιτς της Αγγλίας. Κάθε θέση που βρίσκεται ανατολικά της γραμμής αυτής έχει έναν θετικό αριθμό, και οτιδήποτε βρίσκεται στα δυτικά έχει έναν αρνητικό αριθμό. [25]

Υπάρχει η δυνατότητα να προστεθούν αντικείμενα στο χάρτη για να οριστούν σημεία, γραμμές, περιοχές, ή συλλογές αντικειμένων. Το Google Maps Javascript API ονομάζει αυτά τα αντικείμενα επικαλύψεις (overlays). Οι επικαλύψεις σχετίζονται με συγκεκριμένες συντεταγμένες.

Το Google Maps API παρέχει διάφορους τύπους επικαλύψεων που μπορούν να προστεθούν μέσω προγραμματισμού:

- ✓ Τα σημεία στο χάρτη που εμφανίζονται χρησιμοποιώντας δείκτες (markers).
- ✓ Ένα παράθυρο πληροφοριών (info window) είναι ένα ιδιαίτερο είδος επικάλυψης για την εμφάνιση περιεχομένου που μπορεί να περιέχει κείμενο ή εικόνες σε ένα αναδυόμενο παράθυρο σε μία συγκεκριμένη τοποθεσία στο χάρτη.
- ✓ Γραμμές στο χάρτη που εμφανίζονται χρησιμοποιώντας polylines και είναι μια διατεταγμένη ακολουθία τοποθεσιών.
- ✓ Περιοχές ενός αυθαίρετου σχήματος στο χάρτη εμφανίζονται με πολύγωνα (polygons).
- ✓ Μπορούν να οριστούν κύκλοι (circles) και τετράγωνα (rectangles) στο χάρτη.
- ✓ Μπορεί να τοποθετηθεί μια εικόνα σε ένα χάρτη χρησιμοποιώντας μια επικάλυψη εδάφους (ground overlay).
- ✓ Μπορούν να εφαρμοστούν προσαρμοσμένες επικαλύψεις (custom overlays).

Λόγω του μεγάλου όγκου των δεδομένων δημιουργούνται πάρα πολλοί δείκτες στο χάρτη και αυτό μπορεί να προκαλέσει οπτική υπερφόρτωση και η αλληλεπίδραση με το χάρτη να γίνει αργή. Για να ξεπεραστεί αυτή η κακή απόδοση, οι πληροφορίες που εμφανίζονται στο χάρτη πρέπει να απλοποιηθούν. Η ομαδοποίηση απλοποιεί την οπτικοποίηση των δεδομένων με την ενοποίηση αυτών που βρίσκονται σε κοντινή απόσταση στο χάρτη σε μία συγκεντρωτική μορφή.

Στην εφαρμογή γίνεται χρήση του Marker Clusterer που είναι μια βιβλιοθήκη εφαρμογών στην πλευρά του πελάτη η οποία δημιουργεί και διαχειρίζεται συμπλέγματα ανά επίπεδο ζουμ για πολλούς δείκτες. Εφαρμόζει ο μαδοποίηση με βάση το πλέγμα σε μια συλλογή δεικτών. Η ομαδοποίηση με βάση το πλέγμα γίνεται με τη διαίρεση του χάρτη σε τετράγωνα ενός συγκεκριμένου μεγέθους και στη συνέχεια, την ο μαδοποίηση των δεικτών σε κάθε τετραγωνικό πλέγμα. Αυτή η τεχνική μπορεί να είναι α ρκετά γρήγορη, διότι απαιτεί μόνο τον έλεγχο των δεικτών για το αν είναι ανάμεσα σε ένα σύνολο συντεταγμένων. Δεν είναι απαραίτητος κανένας περίπλοκος υπολογισμός απόστασης. Κάθε σύμπλεγμα που έχει περισσότερους από έναν δείκτες δείχνει τον αριθμό των δεικτών που περιλαμβάνει στο εσωτερικό του αλλιώς εμφανίζεται ένας μόνο δείκτης. Σε κάθε επίπεδο ζουμ εφαρμόζεται ο ίδιος αλγόριθμος.

JSON

Το JSON (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και γράψουν. Είναι εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate). [3]

Τα δεδομένα που συλλέγονται από το Twitter είναι σε μορφή JSON.

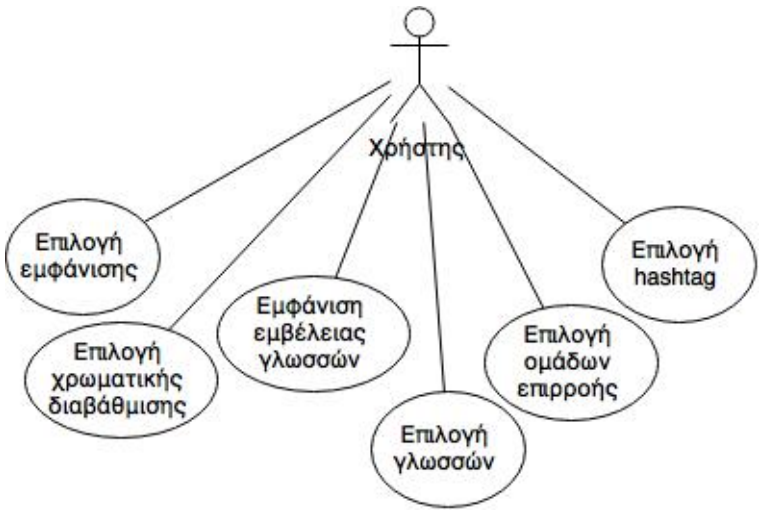
4. Η υλοποιημένη εφαρμογή

Στην ενότητα αυτή θα γίνει παρουσίαση της εφαρμογής που δημιουργήθηκε στα πλαίσια της παρούσας εργασίας και περιγραφή των βασικών λειτουργιών που μπορεί να εκτελέσει ένας χρήστης μέσω αυτής. Η κεντρική σελίδα της εφαρμογής (Εικόνα 13) χωρίζεται σε δύο μέρη, στο χάρτη που απεικονίζονται τα δεδομένα και στο μενού των επιλογών.

Εικόνα 13. Κεντρική σελίδα της εφαρμογής.

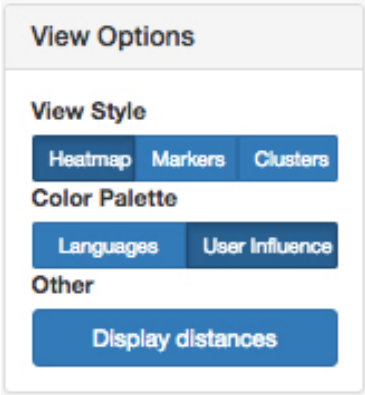
Η εφαρμογή ξεκινάει να λαμβάνει tweets σε πραγματικό χρόνο μόλις πραγματοποιηθεί η σύνδεση με τον εξυπηρετητή και εμφανίζει αυτόματα την τοποθεσία του καθενός στο χάρτη εφόσον βέβαια έχει επιλεχθεί τουλάχιστον μία γλώσσα και μία ο μάδας επιρροής από το μενού.

Στη σελίδα αναγράφεται το πλήθος των δεδομένων που έχει λάβει η εφαρμογή από τον εξυπηρετητή αλλά και το πλήθος των δεδομένων που εμφανίζονται στο χάρτη.



Εικόνα 14. Διάγραμμα περιπτώσεων χρήσης.

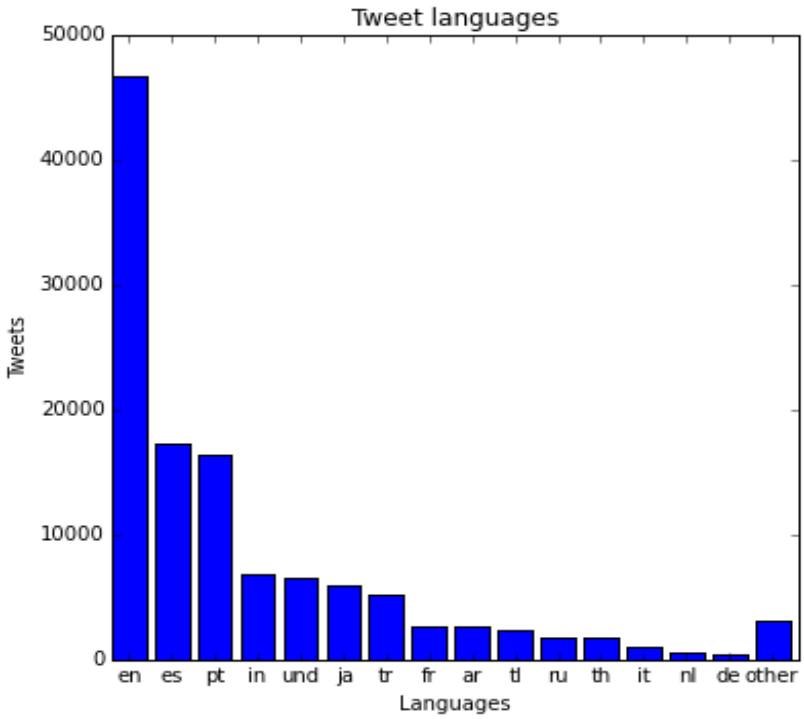
Ο χρήστης έχει τη δυνατότητα να αλληλεπιδρά με την εφαρμογή και έχει μια σειρά από επιλογές (Εικόνα 14) που περιλαμβάνουν μεταξύ άλλων τον τρόπο με τον οποίο θα απεικονίζονται τα δεδομένα στο χάρτη όπως επίσης την επιλογή τόσο των γλωσσών αλλά και των ομάδων επιρροής που επιθυμεί.



Εικόνα 15. Επιλογές για την εμφάνιση των δεδομένων

Πιο αναλυτικά ο χρήστης έχει τρεις (3) επιλογές για τον τρόπο που θα οπτικοποιούνται να δεδομένα. Η πρώτη είναι η εμφάνιση των συντεταγμένων των tweets ως μεμονωμένα σημεία στον χάρτη (markers). Η δεύτερη είναι η απεικόνιση των σημείων ως χάρτης θερμότητας (heatmap) όπου η ένταση του κάθε χρώματος αναπαριστά το πλήθος των tweets σε κάθε περιοχή. Δίνει δηλαδή πληροφορία για τη διασπορά των tweets στο χώρο. Όσο πιο σκούρο είναι το κάθε χρώμα τόσο μεγαλύτερο είναι και το πλήθος των tweets. Τέλος υπάρχει η επιλογή να εμφανίζονται τα κοντινά σημεία ομαδοποιημένα ως ένα όπου αναγράφεται ο αριθμός των σημείων που περιέχει (clusters). Ο χρήστης μπορεί επίσης να πατήσει πάνω σε κάθε σημείο που υπάρχει tweet και να δει περισσότερα στοιχεία για αυτό όπως για παράδειγμα το όνομα του χρήστη που έκανε το συγκεκριμένο tweet, να ακολουθήσει ένα σύνδεσμο όπου μπορεί να δει το tweet μέσα από τη ιστοσελίδα του Twitter, τη γλώσσα στην οποία είναι γραμμένο το tweet και την επιρροή του χρήστη. Επιπρόσθετα μπορεί να γίνει η επιλογή να εμφανίζονται κύκλοι με κέντρο την πρωτεύουσα της κάθε επιλεγμένης χώρας και ακτίνα έναν αριθμό που προσωμοιώνει την εμβέλεια της κάθε γλώσσας. Ο χρήστης έχει επίσης τη δυνατότητα να επιλέξει διάφορα φίλτρα για να περιορίσει τα δεδομένα που θα εμφανίζονται. Μπορεί να επιλέξει μέχρι δέκα γλώσσες και μέχρι τέσσερις ομάδες επιρροής χρηστών. Κάθε γλώσσα ή ομάδα επιρροής χρηστών απεικονίζεται στο χάρτη με διαφορετικό χρώμα και ο χρήστης είναι αυτός που επιλέγει ποιο από τα δύο θέλει να βλέπει με διαφορετικά χρώματα.

Η επιλογή των γλωσσών που είναι διαθέσιμες στο χρήστη έγινε με βάση μία ανάλυση (Εικόνα 16) της συχνότητας εμφάνισης κάθε γλώσσας από μία συλλογή περίπου 120.000 tweets.



Εικόνα 16. Συχνότητα εμφάνισης γλωσσών σε ένα δείγμα από 120.000 tweets

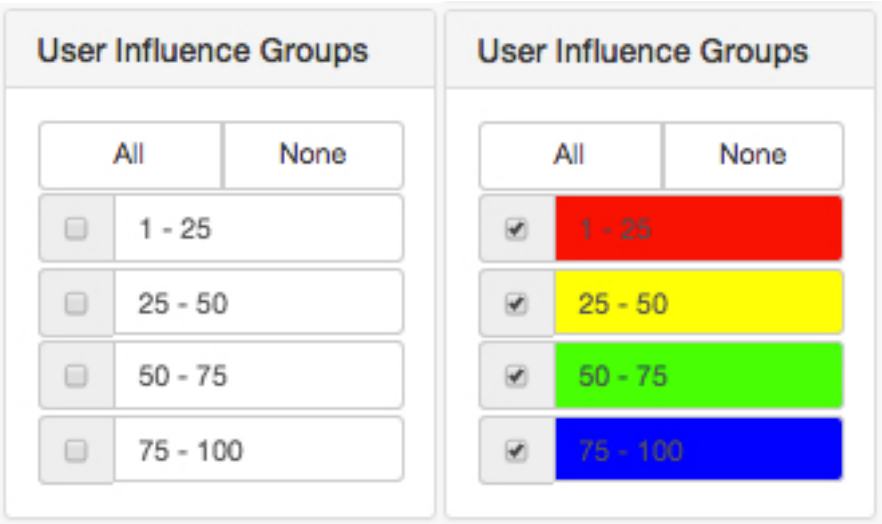
Οι γλώσσες που επιλέχθηκαν είναι δεκατέσσερις (14) και είναι οι ακόλουθες:

- ✓ English
- ✓ Spanish
- ✓ Indonesian
- ✓ Japanese
- ✓ Portuguese
- ✓ Turkish
- ✓ Tagalog
- ✓ Thai
- ✓ Russian
- ✓ Arabic
- ✓ French
- ✓ Italian
- ✓ Dutch
- ✓ German.

Στη παρακάτω εικόνα (Εικόνα 17) φαίνεται το μενού της επιλογής γλωσσών πριν και μετά την επιλογή κάποιας γλώσσας. Υπάρχει επίσης η δυνατότητα να επιλεχθούν οι πρώτες γλώσσες μέχρι το σύνολο των γλωσσών να είναι δέκα (10) καθώς επίσης και να αφαιρεθούν όλες οι γλώσσες. Κάθε γλώσσα που επιλέγεται αποκτά και διαφορετικό χρώμα που εξαρτάται από τη σειρά επιλογής.

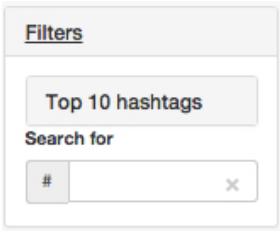
Εικόνα 17. Επιλογές γλώσσας (πριν και μετά την επιλογή)

Οι ομάδες επιρροής που είναι διαθέσιμες στο χρήστη αποφασίστηκε να είναι τέσσερις (4): 1-25, 26-50, 51-75 και 76-100. Στην παρακάτω εικόνα (Εικόνα 18) φαίνεται το μενού για την επιλογή των ομάδων επιρροής που επιθυμεί ο χρήστης να εμφανίζονται στο χάρτη. Τα χρώματα για κάθε ομάδα είναι είναι συγκεκριμένα.



Εικόνα 18. Επιλογές ομάδας επιρροής (πριν και μετά την επιλογή)

Στα φίλτρα της εφαρμογής περιλαβάνεται και το hashtag. Ο χρήστης έχει τη δυνατότητα να δει τα πρώτα δέκα σε αριθμό εμφανίσεων hashtag και να επιλέξει ένα από αυτά ώστε να περιορίσει ακόμα περισσότερο τα δεδομένα που θα εμφανίζονται στο χάρτη. Με αυτό το τρόπο θα εμφανίζονται στο χάρτη μόνο τα tweets που περιέχουν το συγκεκριμένο hashtag.



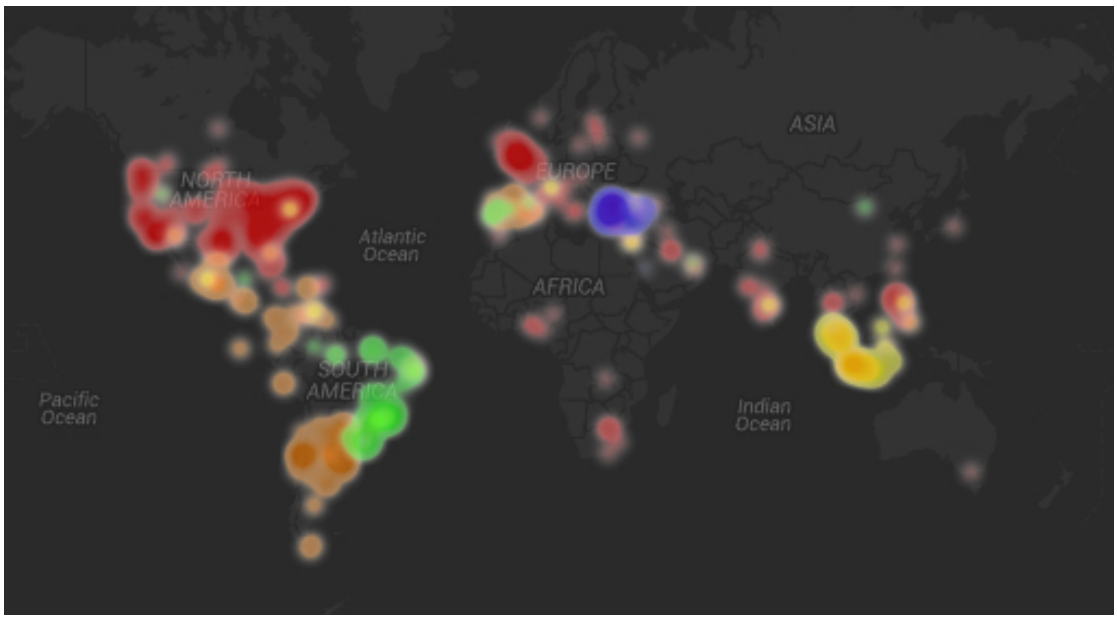
Εικόνα 19. Φίλτρο για την επιλογή hashtag

Τέλος, η εφαρμογή δίνει τη δυνατότητα στο χρήστη να εμφανίσει κάποια διαγράμματα. Αυτά είναι ένα διάγραμμα πίτας που δείχνει το ποσοστό εμφάνισης της κάθε επιλεγμένης γλώσσας και ένα ραβδόγραμμα που δείχνει το πλήθος των tweets που έγιναν σε κάθε επιλεγμένη γλώσσα.

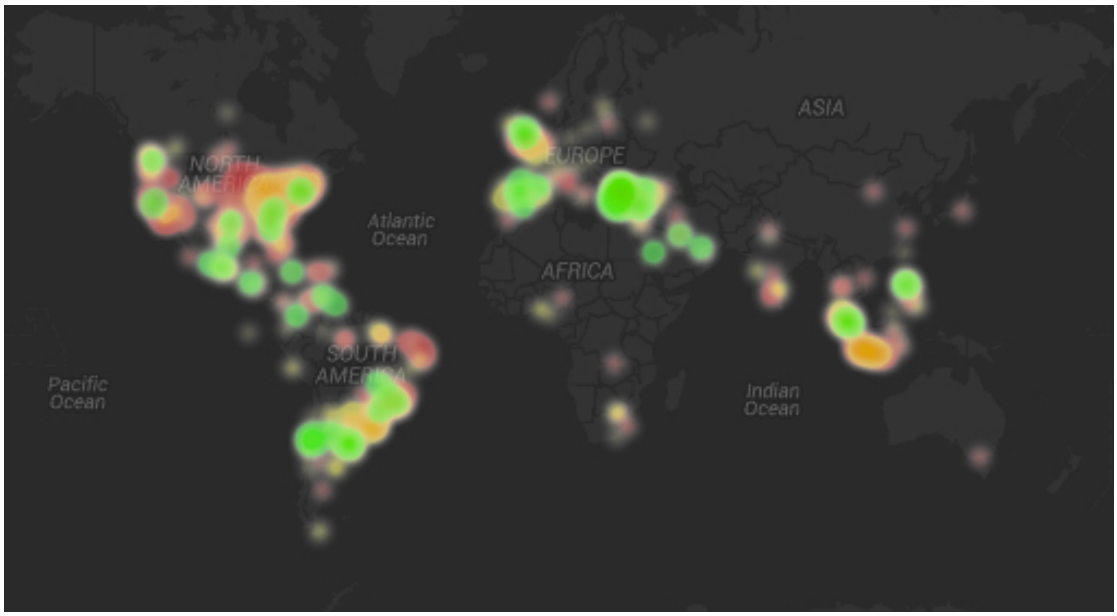
Εικόνα 20. Επιλογή για την εμφάνιση των γραφημάτων.

Για την εμφάνιση των νέων δεδομένων στο χάρτη δε χρειάζεται αλληλεπίδραση του χρήστη καθώς γίνεται αυτόματα κάθε φορά που η ιστοσελίδα δέχεται νέα δεδομένα από τον εξυπηρετητή. Ωστόσο, μετά από κάθε αλλαγή στις επιλογές, ο χρήστης πρέπει να πατήσει ένα κουμπί για να ανανεωθεί ο χάρτης.

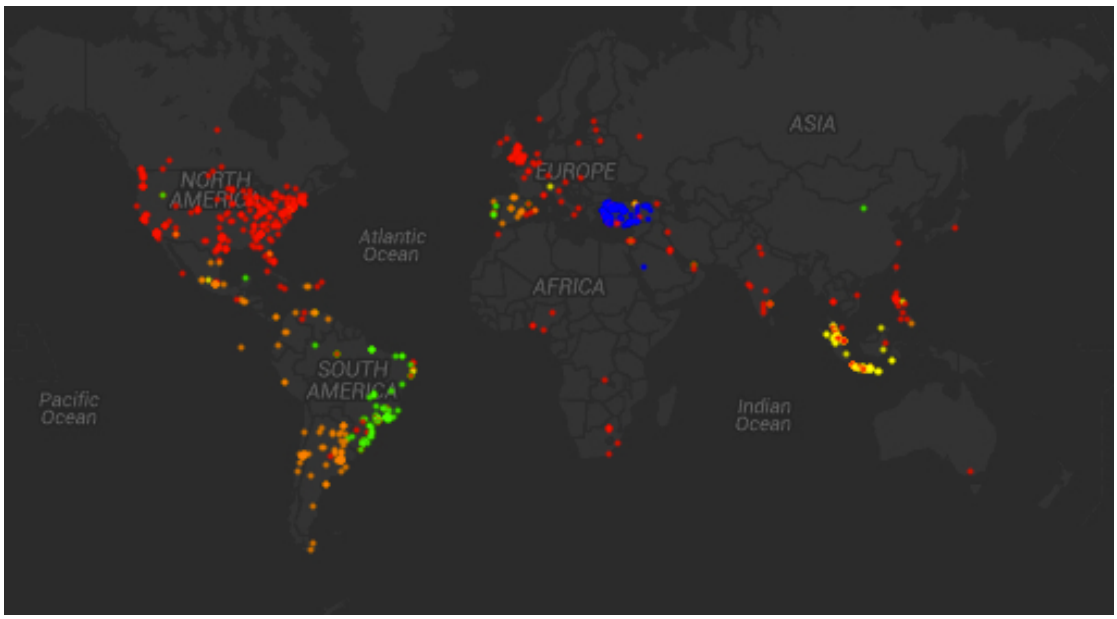
Για την καλύτερη παρουσίαση της εφαρμογής πραγματοποιήθηκε η εκτέλεση της μέχρι να συλλεχθούν περίπου 1000 tweets. Οι γλώσσες που επιλέχθηκαν είναι οι ακόλουθες πέντε (5): English, Spanish, Indonesian, Portuguese και Turkish. Επίσης επιλέχθηκαν όλες οι ομάδες επιρροής. Παρακάτω φαίνονται τα στιγμιότυπα οθόνης που δείχνουν τον τρόπο που απεικονίζονται τα δεδομένα στο χάρτη.



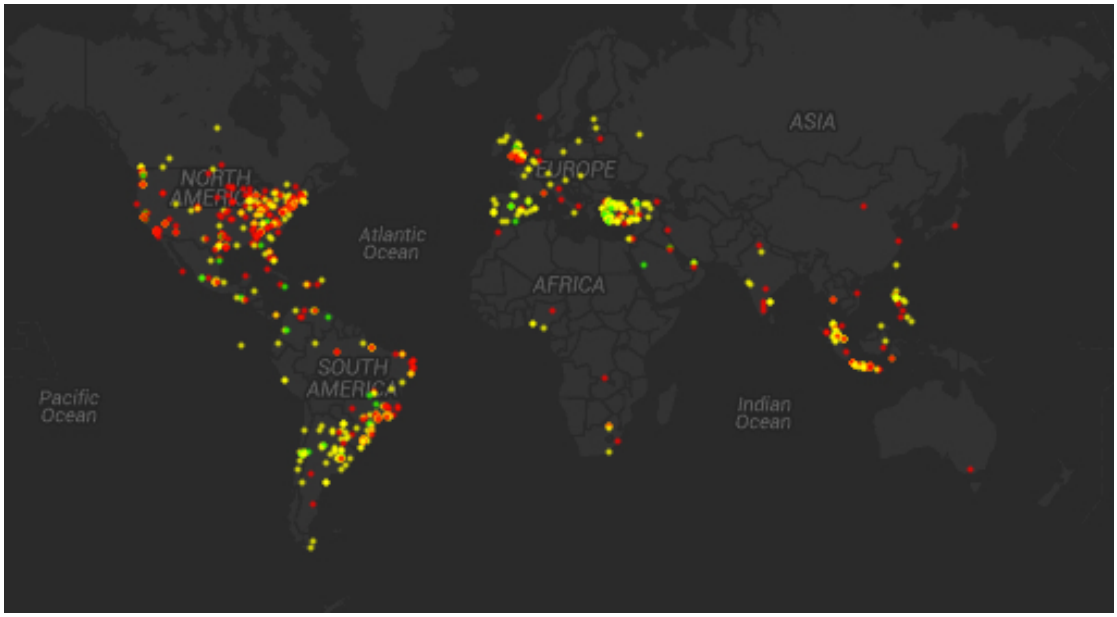
Εικόνα 21. Αναπαράσταση τη γεωγραφικής κατανομής των γλωσσών με χάρτη θερμότητας.



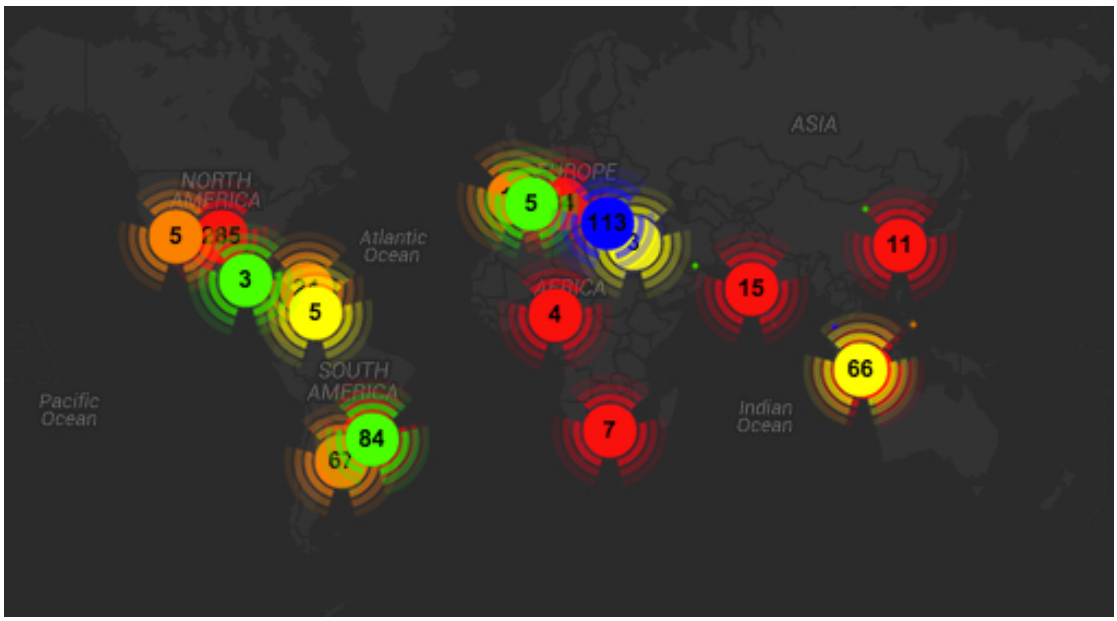
Εικόνα 22. Αναπαράσταση της γεωγραφικής κατανομής των ομάδων επιρροής με χάρτη θερμότητας.



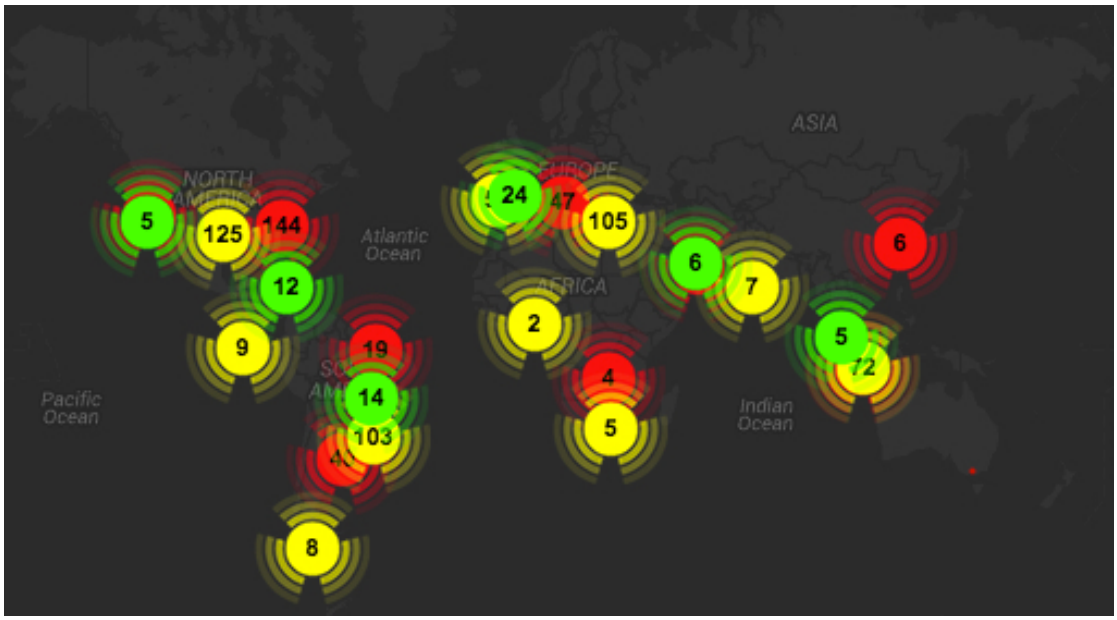
Εικόνα 23. Αναπαράσταση τη γεωγραφικής κατανομής των γλωσσών με δείκτες.



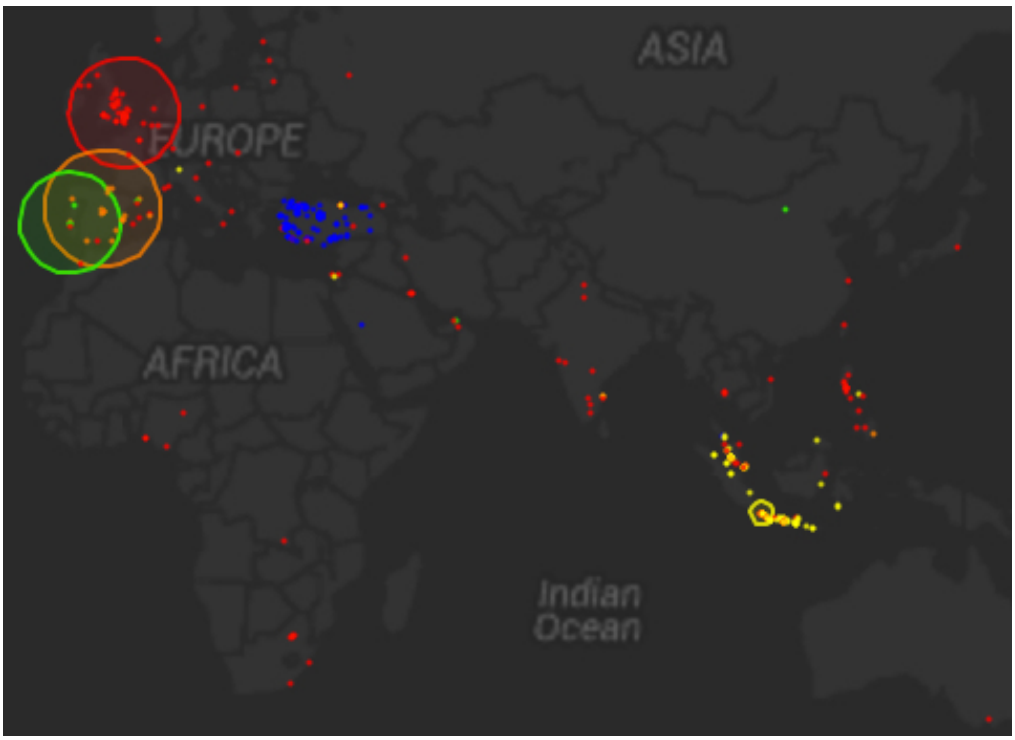
Εικόνα 24. Αναπαράσταση τη γεωγραφικής κατανομής των ομάδων επιρροής με δείκτες.



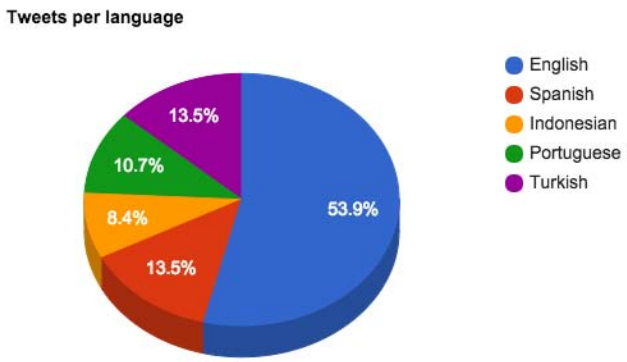
Εικόνα 25. Αναπαράσταση τη γεωγραφικής κατανομής των γλωσσών με συμπλέγματα δεικτών.



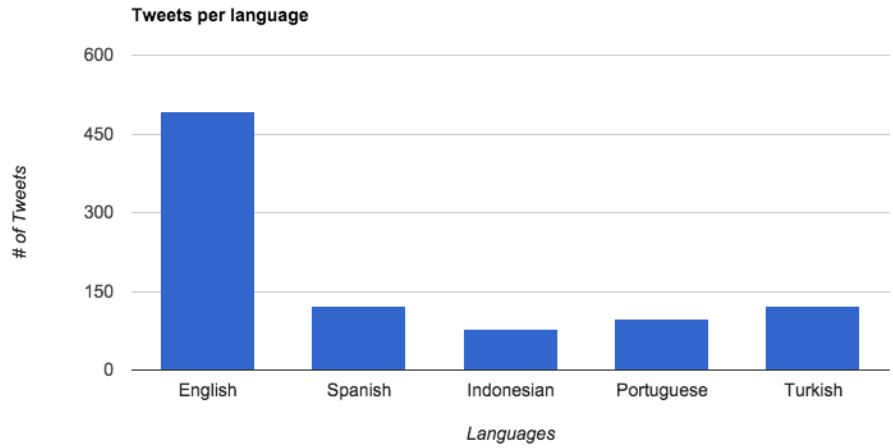
Εικόνα 26. Αναπαράσταση τη γεωγραφικής κατανομής των ομάδων επιρροής με συμπλέγματα δεικτών.



Εικόνα 27. Εμβέλεια χρήσης της κάθε επιλεγμένης γλώσσας.



Εικόνα 28. Αναπαράσταση του ποσοστού εμφάνισης της κάθε επιλεγμένης γλώσσας με διάγραμμα πίτας.



Εικόνα 29. Αναπαράσταση του πλήθους των tweets που έγιναν σε κάθε επιλεγμένη γλώσσα με ραβδόγραμμα.

5. Επίλογος

5.1 Σύνοψη - Συμπεράσματα

Συνοψίζοντας, αναπτύχθηκε ένα σύστημα αυτόματης συλλογής δεδομένων πραγματικού χρόνου από την ιστοσελίδα κοινωνικής δικτύωσης Twitter. Τα δεδομένα που συλλέγονται αναλύονται και μέσω της διαδικτυακής εφαρμογής δίνεται η δυνατότητα στους χρήστες της να δουν τη γεωγραφική κατανομή τόσο των ίδιων των tweets αλλά και της κάθε γλώσσας που είναι γραμμένα. Επίσης δίνεται η δυνατότητα να δει κάποιος την επιρροή των χρηστών σε συνδυασμό με τη κάθε γλώσσα δημοσίευσης.

Η μελέτη διαδικτυακών κοινωνικών δικτύων είναι ένας χώρος που παρουσιάζει έντονο ενδιαφέρον καθώς υπάρχει ένας μεγάλος όγκος δεδομένων από τον οποίο μπορούν να εξαχθούν πολλά και χρήσιμα συμπεράσματα. Το Twitter, με το ανοικτό API του, έχει γίνει μία από τις πιο δημοφιλείς πηγές δεδομένων για κοινωνική έρευνα. Οι προσπάθειες όμως που εστιάζονται στη γεωγραφία του Twitter είναι λίγες. Στην παρούσα διπλωματική έγινε προσπάθεια να δωθεί έμφαση στη γλωσσική γεωγραφία του Twitter αλλά και στην επιρροή των χρηστών.

Στα πλαίσια της παρούσας εργασίας μελετήθηκαν οι ιστοσελίδες κοινωνικής δικτύωσης και πιο συγκεκριμένα το Twitter. Έγινε γνωστή η διαδικασία σύνδεσης με το Twitter και ο τρόπος χρήσης των διεπαφών προγραμματισμού εφαρμογών του. Σημαντική ήταν η εξοικείωση με τη γλώσσα προγραμματισμού python και των διαφόρων πλαισίων διαδικτυακών εφαρμογών και βιβλιοθηκών καθώς επίσης και η χρήση γνωστών διαδικτυακών γλωσσών και τεχνολογιών. Μεγάλο κέρδος από τη πτυχιακή αυτή είναι επίσης και η επαφή με τις μη-σχεσιακές βάσεις δεδομένων αλλά και τα συστήματα ανταλλαγής μηνυμάτων. Κατά τη διάρκεια ανάπτυξης της εφαρμογής έγινε προσπάθεια να υπάρχει όσο το δυνατόν ορθότερη αρχιτεκτονική και να ακολουθηθούν βασικές αρχές εγγραφής κώδικα.

Το αποτέλεσμα κρίνεται αρκετά ικανοποιητικό αλλά η εφαρμογή έχει αρκετά περιθώρια βελτίωσης. Υπάρχουν οι περιορισμοί που θέτει η διεπαφή προγραμματισμού εφαρμογών του Twitter αλλά και το Social Authority API. Το πλήθος των tweets του δείγματος που λαμβάνεται και περιέχει γεωγραφική πληροφορία είναι σχετικά μικρό. Τα παραπάνω οδηγούν στο να γεμίζει ο χάρτης με δεδομένα με πολύ αργό ρυθμό και ο χρήστης να χρειάζεται να περιμένει για κάποιο σημαντικό χρονικό διάστημα ώστε η εικόνα που θα έχει μπροστά του να είναι ικανή για να τον βοηθήσει να εξάγει τα δικά του χρήσιμα συμπεράσματα. Από τα παραπάνω γίνεται εύκολα αντιληπτή η ανάγκη εύρεσης ενός τρόπου ώστε να τροφοδοτείται η εφαρμογή με περισσότερα δεδομένα σε πιο σύντομο χρονικό διάστημα. Όσο η εφαρμογή είναι ενεργή, λαμβάνει συνεχώς δεδομένα από τον εξυπηρετητή και τα διατηρεί μέχρι τον τερματισμό της. Αυτό όπως είναι λογικό έχει ως αποτέλεσμα την στνεχόμενη αύξηση της μνήμης που χρησιμοποιείται. Η αναζήτηση μίας άλλης τεχνικής για την αποθήκευση μεγάλου πλήθους δεδομένων είναι απαραίτητη. Επίσης, όταν η εφαρμογή έχει έναν αρκετά μεγάλο όγκο δεδομένων μπορεί να οδηγήσει σε αργή ανανέωση του χάρτη.

5.2 Μελλοντικές επεκτάσεις

Στο υπάρχον σύστημα θα μπορούσαν να προστεθούν πολλές και διάφορες επιπλέον λειτουργίες μερικές από τις οποίες είναι οι ακόλουθες:

- ✓ Υπολογισμός της επηρροής του κάθε χρήστη και με τη βοήθεια άλλων υπηρεσιών όπως για παράδειγμα το Influence Tracker (influencetracker.com).

- ✓ Συσχέτιση της πυκνότητας των tweets με πραγματικά γεγονότα όπως συναυλίες, αγώνες, εκλογές, φυσικές καταστροφές, κ.α..

- ✓ Ενσωμάτωση επιπλέον γραφημάτων σχετικών με την επιρροή των χρηστών για κάθε γλώσσα. Θα μπορούσαν έτσι να βγαίνουν συμπεράσματα για την επιρροή της κάθε γλώσσας σε πραγματικό χρόνο.

Βιβλιογραφία

1. *Geopy*. Ανάκτηση από https://github.com/geopy/geopy
2. *Influence Tracker*. Ανάκτηση από http://influencetracker.com
3. *Json*. Ανάκτηση από http://json.org
4. *Linux*. Ανάκτηση από https://el.wikipedia.org/wiki/Linux
5. *MongoDB*. Ανάκτηση από https://en.wikipedia.org/wiki/MongoDB
6. *MongoDB*. Ανάκτηση από https://www.mongodb.org
7. *NoSQL*. Ανάκτηση από https://en.wikipedia.org/wiki/NoSQL
8. *Python*. Ανάκτηση από https://el.wikipedia.org/wiki/Python
9. *Social authority* Ανάκτηση από https://moz.com/blog/social-authority
10. *Social authority*. Ανάκτηση από https://followerwonk.com/social-authority
11. *Social influence*. Ανάκτηση από https://en.wikipedia.org/wiki/Social_influence
12. *Social network*. Ανάκτηση από https://en.wikipedia.org/wiki/Social_network
13. *Social networking service*. Ανάκτηση από https://en.wikipedia.org/wiki/Social_networking_service
14. *Tornado*. Ανάκτηση από http://www.tornadoweb.org
15. *Tornado*. Ανάκτηση από https://en.wikipedia.org/wiki/Tornado_(web_server)
16. *Twitter - About company*. Ανάκτηση από https://about.twitter.com/company
17. *Twitter Developers*. Ανάκτηση από https://dev.twitter.com
18. *Twitter*. Ανάκτηση από https://en.wikipedia.org/wiki/Twitter
19. *Ubuntu*. Ανάκτηση από https://el.wikipedia.org/wiki/Ubuntu
20. Chodorow, K & Dirolf , M. (2010). *MongoDB: The Definitive Guide*. USA: O'Reilly Media.
21. Cook, D. (2014). *Data Push Apps with HTML5 SSE*. USA: O'Reilly Media.
22. Leetaru, K., Wang, S., Cao, G., Padmanabhan, A. & Shook, E. *Mapping the global Twitter heartbeat: The geography of Twitter*. First Monday, Volume 18, Number 5 - 6 May 2013
23. Razis, G. & Anagnostopoulos, I. *InfluenceTracker: Rating the impact of a Twitter account*. AIAI 2014 Workshops, IFIP AICT 437, 2014, pp. 184–195
24. Russell, M. (2014). *Mining the Social Web, Second Edition*. USA: O'Reilly Media.
25. Svennerberg, G. (2010). *Beginning Google Maps API 3*. USA: Apress Media.
26. Αντωνίου, Η. (2015). *Ανάλυση και Οπτικοποίηση Δεδομένων από Μέσα Κοινωνικής Δικτύωσης*.
27. Γεμενετζή, Κ. (2010). *Σύστημα Συλλογής και Ανάλυσης Δεδομένων Διαδικτυακών Κοινωνικών Μέσων*.
28. Καρακόλης Ε. (2015). *Αναζήτηση και Ανάλυση Influential Χρηστών Twitter*.

Παράρτημα Ι

	Γλώσσα	Κωδικός Γλώσσας (Twitter)	Πρωτεύουσα	Συντεταγμένες
1	English	en	London, UK	[51.5073509, -0.1277583]
2	Spanish	es	Madrid, Spain	[40.4167754, -3.7037902]
3	Indonesian	in	Jakarta, Indonesia	[-6.2087634, 106.845599]
4	Japanese	ja	Tokyo, Japan	[35.7090259, 139.7319925]
5	Portuguese	pt	Lisbon, Portugal	[38.7222524, -9.1393366]
6	Turkish	tr	Ankara, Turkey	[39.92077, 32.85411]
7	Tagalog	tl	Manila, Philippines	[14.5995124, 120.9842195]
8	Thai	th	Bangkok, Thailand	[13.7563309, 100.5017651]
9	Russian	ru	Moscow, Russia	[55.755826, 37.6173]
10	Arabic	ar	Riyadh, Saudi Arabia	[24.633333, 46.716667]
11	French	fr	Paris, France	[48.856614, 2.3522219]
12	Italian	it	Rome, Italy	[41.9027835, 12.4963655]
13	Dutch	nl	Amsterdam, The Netherlands	[52.3702157, 4.895167900000001]
14	German	de	Berlin, Germany	[52.52000659999999, 13.404954]

Πίνακας 1. Λίστα διαθέσιμων γλωσσών και σχετικές πληροφορίες.

Παράρτημα II

Κώδικας Εφαρμογής

Πρόγραμμα συλλογής δεδομένων

```
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import time
import sys
import datetime
import signal
import json
from pymongo import MongoClient
import config
from TweetObject import TweetObject

def saveToMongoDB(post):
    client = MongoClient(config.mongoDB_host, config.mongoDB_port)
    db = client[config.mongoDB_database]
    collection = db[config.mongoDB_collection]
    collection.insert(post)

class listener(StreamListener):

    def on_connect(self):
        # Called once connected to streaming server.
        # Perform some work prior to entering the read loop.

        print "Connected to streaming server"

    def parse_data(self,tweet_json):
        tweet = TweetObject(tweet_json)
        data = {}
        data['t_text'] = tweet.getTweetText()
        data['t_date'] = tweet.getTweetDate()
        data['t_id'] = tweet.getTweetId()
        data['t_lang'] = tweet.getTweetLang()
        data['t_geo'] = tweet.getTweetCoordinates()
        data['t_uid'] = tweet.getUser().getUserId()
        data['t_uname'] = tweet.getUser().getUserScreenName()
        data['t_hashtags'] = tweet.getTweetHashtags()
        data['c_distance'] = 'NA'
```

```
        return data

    def on_data(self, raw_data):
        # Called when raw data is received from connection.
        # Handle the stream data

        try:

            data = json.loads(raw_data)

            if 'in_reply_to_status_id' in data:
                collected_data = self.parse_data(data)
                saveToMongoDB(collected_data)

            elif 'limit' in data:
                if self.on_limit(data['limit']['track']) is False:
                    return False
            elif 'disconnect' in data:
                if self.on_disconnect(data['disconnect']) is False:
                    return False
            elif 'warning' in data:
                if self.on_warning(data['warning']) is False:
                    return False

        except (KeyboardInterrupt, SystemExit):
            sys.exit(0)

        except BaseException as e:
            print 'Failed on data: ' + str(e)

    def on_exception(self, exception):
        """Called when an unhandled exception occurs."""
        print 'Unhandled Exception: ' + str(exception)
        return

    def on_limit(self, track):
        """Called when a limitation notice arrives"""
        print 'Limitation notice: ' + str(track)
        return

    def on_error(self, status_code):
        """Called when a non-200 status code is returned"""
        print 'Error (non-200 status code): ' + str(status_code)
        return False

    def on_timeout(self):
        """Called when stream connection times out"""
        print 'Stream connection timeout'
```

```
        return

    def on_disconnect(self, notice):
        """Called when twitter sends a disconnect notice
        Disconnect codes are listed here:
        https://dev.twitter.com/docs/streaming-apis/messages#Disconnect_messages_disconnect
        """
        print 'Disconnect notice: ' + str(notice)
        return

    def on_warning(self, notice):
        """Called when a disconnection warning message arrives"""
        print 'Disconnection warning message: ' + str(notice)
        return

def signal_handler(signal, frame):
    raise SystemExit

signal.signal(signal.SIGTERM, signal_handler)

auth = OAuthHandler(config.ckey, config.csecret)
auth.set_access_token(config.atoken, config.asecret)
while True:
    try:
        l = listener()
        twitterStream = Stream(auth, l)
        twitterStream.sample()
    except (KeyboardInterrupt, SystemExit):
        print 'Tweet collector was terminated.'
        sys.exit(0)
    except BaseException as e:
        print 'Failed on sample: ' + str(e)
        print 'Restarting...'
        time.sleep(5)
```

Πρόγραμμα επεξεργασίας δεδομένων

```
from pymongo import MongoClient
import time
import datetime
import sys
import signal
import logging
from GeoService import GeoService
from FollowerWonk import FollowerWonk
import config as tc_conf
import tp_conf
import Queue
import threading
```

```
class FWSA (threading.Thread):

    processing_delay = 1
    max_ids_per_request = 25
    max_requests_per_second = 5

    def __init__(self, threadID, name, q_in, q_out):
        super(FWSA,self).__init__()
        self.threadID = threadID
        self.name = name
        self.q_in = q_in
        self.q_out = q_out

    def run(self):
        while True:

            qs = self.q_in.qsize()

            if qs == 0:
                time.sleep(self.processing_delay)
                continue

            if qs <= self.max_ids_per_request:
                processing_items_num = qs
            else:
                processing_items_num = self.max_ids_per_request
            processing_items = []

            for i in range(processing_items_num):
                q_item = self.q_in.get()
                self.q_in.task_done()
                processing_items.append(q_item)

            fwsa_req = ",".join([ item['t_uname'] for item in processing_items])
            fwsa_res = FollowerWonk.social_authority(fwsa_req)

            for item in processing_items:
                if item['t_uname'] in fwsa_res:
                    item["c_fwsa"] = round( float(fwsa_res[item['t_uname']]) ,2 )
                else:
                    item["c_fwsa"] = -1
                self.q_out.put(item)

            time.sleep(self.processing_delay)


class MongodbReader(object):

    def __init__(self,config):
```

```
        self.config = config
        self.client = self.get_connection()
        self.database = self.get_database(self.client)
        self.geoService = GeoService()

    def get_database(self,client):
        return client[self.config['database']]

    def get_collection(self,database):
        return database[self.config['collection']]

    def get_connection(self):
        return MongoClient(self.config['host'], self.config['port'])

    def on_start(self):
        pass

    def start(self,rtype='rt'):
        self.on_start()
        collection = self.get_collection(self.database)
        count = 0
        latest = collection.find().sort( [('$natural', -1)] ).limit(1);
        latest_id = latest[0]["_id"]
        query = {"_id": {"$gt": latest_id}}
        cursor = collection.find(query, tailable=True, await_data=True)
        while cursor.alive:
            try:
                post = cursor.next()
                count += 1
                latest_id = post["_id"]
                self.process_data(count, post)
            except StopIteration:
                if cursor.alive == False:
                    print "Cursor died."
                    query = {"_id": {"$gt": latest_id}}
                    cursor = collection.find(query, tailable=True, await_data=True)
                print "StopIteration Error!"
                time.sleep(1)

    def process_data(self,id,data):
        print id,"::",data

class MongodbWriter(object):

    def __init__(self,config):
        self.config = config
        self.client = self.get_connection()
        self.database = self.get_database(self.client)
        self.collection = self.get_collection(self.database)
```

```
    def get_database(self,client):
        return client[self.config['database']]

    def get_collection(self,database):
        return database[self.config['collection']]

    def get_connection(self):
        return MongoClient(self.config['host'], self.config['port'])

    def insert_document(self,doc):
        self.collection.insert(doc)

class MongodbWriterThreaded(MongodbWriter, threading.Thread):

    def __init__(self,threadID,name,config,queue):
        MongodbWriter.__init__(self,config)
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.q = queue

    def run(self):
        while True:
            q_item = self.q.get()
            self.q.task_done()
            self.insert_document(q_item)

class TweetProcessor(MongodbReader):

    user_ids = []
    threads = []
    def __init__(self,config):
        super(TweetProcessor, self).__init__(config['db_in'])
        self.q_fwsa_in = Queue.Queue()
        self.q_fwsa_out = Queue.Queue()
        mdb_writer                                                    =
MongodbWriterThreaded(1,"mdb_writer_thread",config['db_out'],self.q_fwsa_out)
        mdb_writer.daemon = True
        mdb_writer.start()
        self.threads.append(mdb_writer)
        fwsa_thread = FWSA(1,"fwsa_thread",self.q_fwsa_in, self.q_fwsa_out)
        fwsa_thread.daemon = True
        fwsa_thread.start()
        self.threads.append(fwsa_thread)

    def on_start(self):
        pass
```

```
    def process_data(self,id,data):
        try:
            utcTimeNow = datetime.datetime.utcnow().replace(microsecond=0)

            if self.check_language(data['c_lang']) == False:
                return

            if data['t_geo'] == []:
                return

            data['c_distance']                                                    =
self.calculate_distance(self.geoService,data['c_lang'],data['t_geo'])
            del data['_id']
            self.q_fwsa_in.put(data)
        except BaseException as e:
            print 'Failed on process_data(): ' + str(e)

    def process_per_second(self,id,data):
        pass

    def save_data(self,mongodb_writer,data):
        mongodb_writer.insert_document(data)

    def calculate_distance(self,geoService,lang,geo):
        return geoService.getGeoDistance(lang, geo[::-1]) or 'NA'

    def check_language(self,lang):
        if lang in tc_conf.languages.keys():
            return True
        else:
            return False

def signal_handler(signal, frame):
        raise SystemExit

signal.signal(signal.SIGTERM, signal_handler)

config = {
    'db_in' : tp_conf.mdb_conf_in,
    'db_out' : tp_conf.mdb_conf_out,
}

while True:
    try:
        tp = TweetProcessor(config)
        tp.start()
    except (KeyboardInterrupt, SystemExit):
        print 'Tweet processor was terminated.'
        sys.exit(0)
```

```
    except BaseException as e:
        print 'Failed: ' + str(e)
        print 'Restarting...'
        time.sleep(5)
```

Πρόγραμμα υπολογισμού γεωδαιτικής απόστασης

```
from geopy.geocoders import GoogleV3
from geopy import distance
import config

class GeoService(object):

    origins = {}

    def __init__(self, default=False):
        if default:
            geolocator = GoogleV3()
            for key in config.languages:
                address, (lat, lon) = geolocator.geocode(config.languages[key]['capital'])
                self.setOrigin(key,[lat,lon])
        else:
            for key in config.languages:
                self.setOrigin(key,config.languages[key]['capital']['geo'])

    def setOrigin(self, language, coordinates):
                    self.origins[language] = coordinates

    def getOrigins(self):
        return self.origins

    def getGeoDistance(self, language, coordinates, formula="great_circle"):
        if language is not None and language in self.origins and coordinates is not None and
len(coordinates) > 0:
            if formula == "vincenty":
                return distance.vincenty(self.origins[language], coordinates).kilometers
            elif formula == "great_circle":
                return distance.great_circle(self.origins[language], coordinates).kilometers
            else:
                return None
        else:
            return None
```

Πρόγραμμα υπολογισμού του Social Authority

```
from hashlib import sha1
from time import time
import base64
import urllib2, urllib
import json
import hmac
import json
import socket

follower_wonk_access_id = ''
follower_wonk_secret_key = ''

class FollowerWonk(object):
    @staticmethod
    def social_authority(users):

        # Every request to the Social Authority API follows the same basic format:
        #                                                    http://api.followerwonk.com/social-
authority?{screen_name};{user_id};{AccessID};{Timestamp};{Signature}

        uri = 'https://api.followerwonk.com/social-authority'

        while True:
            try:
                # The Timestamp parameter is a Unix timestamp (UTC) that indicates for how long
this request is valid.
                # This should be a time in the future, usually no more than several minutes later
than the moment your client code initiates the transfer.
                datime = int(time() + 500)

                # The Signature parameter is an HMAC-SHA1 hash of your Access ID, followed by
a new line, followed by the Timestamp parameter, using your Secret Key.
                s          =          hmac.new(follower_wonk_secret_key,          "%s\n%s"          %
(follower_wonk_access_id, datime), sha1).digest()

                # This hash must be base64 and URL-encoded before being placed in the request
query string.
                b64  = base64.b64encode(s)
                signature = urllib.quote_plus(b64)

                auth          =          "AccessID=%s;Timestamp=%s;Signature=%s;"          %
(follower_wonk_access_id, datime, signature)
                req = urllib2.Request("%s?screen_name=%s;%s" % (uri, users, auth))
                r = urllib2.urlopen(req, None, 3)
                response = json.loads( r.read() )
                r.close()
            except Exception,e:
                print "FWSA Request Error!",e
                continue
```

```
            except BaseException,e:
                print "FWSA Request Error! baseexception:",e
                continue
            break

        if '_embedded' not in response:
            return {}

        res = {user['screen_name'] : user['social_authority'] for user in response['_embedded']}
        return res
```

Πρόγραμμα για την εξαγωγή δεδομένων από το Twitter

```
class TweetObject(object):

    tweet_types = {
        'status' : 'in_reply_to_status_id',
        'delete' : 'delete',
        'event' : 'event',
        'direct_message' : 'direct_message',
        'friends' : 'friends',
        'limit' : 'limit',
        'disconnect' : 'disconnect',
        'warning' : 'warning',
        'status_withheld' : 'status_withheld',
        'user_withheld' : 'user_withheld',
        'scrub_geo' : 'scrub_geo',
        'errors' : 'errors',
    }

    def __init__(self,json):
        self.json = json

    def getTweetType(self):
        for key in self.tweet_types:
            if self.tweet_types[key] in self.json:
                return key
        return None

    def getTweetId(self):
        return self.json['id_str']

    def getTweetDate(self):
        return self.json['created_at']

    def getTweetSource(self):
        return self.json['source']
```

```
    def getTweetCoordinates(self):
        if self.json['coordinates'] is not None:
            return self.json['coordinates']['coordinates']
        else:
            return []

    def getTweetHashtags(self):
        hashtags_array = self.json['entities']['hashtags']
        hashtags_text = []
        for hashtag in hashtags_array:
            hashtags_text.append(hashtag['text'])
        return hashtags_text

    def getTweetLang(self):
        return self.json['lang']

    def getTweetText(self):
        return self.json['text']

    def getUser(self):
        return UserObject(self.json['user'])


class UserObject(object):

    def __init__(self, json):
        self.json = json

    def getUserId(self):
        return self.json['id_str']

    def getUserScreenName(self):
        return self.json['screen_name']
```

Εξυπηρετητής

```
import tornado.ioloop
import tornado.web
import tornado.gen
from tornado.options import define, options
from motor import MotorClient
from pymongo import MongoClient
import os
import json
import datetime
```

```
# MongoDB
mongoDB_host = ''
mongoDB_port = None
mongoDB_database = ''
mongoDB_collection = ''
mongoDB_collection_inf = ''

define('port', default=80, help='Run on the given port', type=int)
define('address', default='0.0.0.0', help='Bind to given address', type=str)

class TweetmapHandler(tornado.web.RequestHandler):
    def get(self):
        print "GET /tweetmap request from", self.request.remote_ip
        self.render("tweetmap.html")

class DataHandler(tornado.web.RequestHandler):

    def initialize(self):
                    self.set_header('Content-Type','text/event-stream; charset=utf-8')
                    self.set_header('Cache-Control','no-cache')
                    self.set_header('Connection','keep-alive')

    def on_connection_close(self):
        print "Client closed the connection."
        self.finish()

    def on_finish(self):
        print "End of a request."

    @tornado.web.asynchronous
    @tornado.gen.coroutine
    def get(self):
        print "GET /data request from", self.request.remote_ip
        col = self.settings['db_col']
        latest = col.find().sort( [('$natural', -1)] ).limit(1)[0];
        yield latest.fetch_next
        doc = latest.next_object()
        latest_id = doc["_id"]
        query = {
            "_id": {"$gt": latest_id}
        }
        params = {
            "tailable": True,
            "await_data": True
        }
        cursor = col.find(query, **params)

        while True:
            if not cursor.alive:
```

```
            # While collection is empty, tailable cursor dies immediately
            print "cursor died..."
            yield tornado.gen.Task(tornado.ioloop.IOLoop.instance().add_timeout,
datetime.timedelta(seconds=1))
            print "requering..."
            cursor = col.find(query, **params)

        while (yield cursor.fetch_next):
            message = cursor.next_object()
            data = {}
            data['t_lang'] = str(message["t_lang"])
            data['t_geo'] = message["t_geo"]
            data['t_id']  = message["t_id"]
            data['t_uname']  = message["t_uname"]
            data['t_hashtags'] = message["t_hashtags"]
            data['c_distance'] = message["c_distance"]
            data['c_fwsa'] = message["c_fwsa"]
            self.write( 'data:' + json.dumps(data) + '\n\n')
            self.flush()

if __name__ == "__main__":

    try:
        tornado.options.parse_command_line()
        client = MotorClient(mongoDB_host, mongoDB_port)
        db = client[mongoDB_database]
        db_col = db[mongoDB_collection]
        db_col_inf = db[mongoDB_collection_inf]

        # Tornado Application Settings
        settings = {
            "template_path" : os.path.join(os.path.dirname(__file__), "templates"),
            "static_path" : os.path.join(os.path.dirname(__file__), "static"),
            "debug" : True,
            "db_col" :  db_col,
            "db_col_inf" :  db_col_inf,
        }

        # Tornado Request Handlers
        handlers = [
            (r"/data", DataHandler),
            (r"/tweetmap", TweetmapHandler),
            (r"/files/(.*)", tornado.web.StaticFileHandler, {"path":
os.path.join(os.path.dirname(__file__), "files")}),
        ]

        # Create Tornado Application
        application = tornado.web.Application(handlers, **settings)
        application.listen(options.port, options.address)
```

```
        tornado.ioloop.IOLoop.instance().start()

    except KeyboardInterrupt:
        pass
    finally:
        print "Tornado server terminated"
```

Κώδικας HTML για τον εξυπηρετούμενο

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="author" content="Marios Siganos">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Tweetmap</title>
        <link rel="shortcut icon" href="{{ static_url("images/lemur.png") }}">
                    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">
                    <link rel="stylesheet" type="text/css" href="{{
static_url("styles/tweetmap.css") }}">
                    <script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
        <script src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
                    <script
src="https://maps.googleapis.com/maps/api/js?v=3.exp&libraries=visualization"></script>
        <script type="text/javascript" src="https://www.google.com/jsapi"></script>
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script>
        <script src="{{ static_url("scripts/markerclusterer_packed.js") }}"></script>
        <script src="{{ static_url("scripts/map-styles.js") }}"></script>
        <script src="{{ static_url("scripts/languages.js") }}"></script>
                    <script src="{{ static_url("scripts/colors.js") }}"></script>
                    <script src="{{ static_url("scripts/tweetmap.js") }}"></script>
    </head>
    <body>
        <header>
            <div id="header">
                <div id="btnHome"><a href="http://snf-49525.vm.okeanos.grnet.gr"><span
class="glyphicon glyphicon-home" aria-hidden="true"></span></a></div>
                Tweetmap
                <div id="btnMenu"><a href="#"><span class="glyphicon glyphicon-
cog"></span></a></div>
            </div>
        </header>
        <div id="main">
            <div id="sidebar">
                <div class="well well-sm">Total Tweets:<div id="tweetCounter">-</div></div>
```

```
            <button type="button" id="btnUpdate" data-updating-text="Updating..." class="btn
btn-danger btn-block" autocomplete="off">Update</button>
            <div class="panel-group" id="accordion" role="tablist" aria-
multiselectable="true">
              <div class="panel panel-default">
                <div class="panel-heading" role="tab" id="heading1">
                  <h4 class="panel-title">
                    <a data-toggle="collapse" data-parent="#accordion" href="#collapse1" aria-
expanded="true" aria-controls="collapse1">
                      View Options
                    </a>
                  </h4>
                </div>
                <div id="collapse1" class="panel-collapse collapse" role="tabpanel" aria-
labelledby="heading1">
                  <div class="panel-body">
                    <label for="opt_grp_view">View Style</label>
                    <div class="btn-group btn-group-sm btn-group-justified" id="opt_grp_view"
data-toggle="buttons">
                      <label class="btn btn-primary active">
                        <input type="radio" name="opt_view" value="heatmap"
autocomplete="off" checked>Heatmap
                      </label>
                      <label class="btn btn-primary">
                        <input type="radio" name="opt_view" value="markers"
autocomplete="off">Markers
                      </label>
                      <label class="btn btn-primary">
                        <input type="radio" name="opt_view" value="clusters"
autocomplete="off">Clusters
                      </label>
                    </div>
                    <label for="opt_grp_color">Color Palette</label>
                    <div class="btn-group btn-group-sm btn-group-justified" id="opt_grp_color"
data-toggle="buttons">
                      <label class="btn btn-primary active">
                        <input type="radio" name="opt_color" value="languages"
autocomplete="off" checked>Languages
                      </label>
                      <label class="btn btn-primary ">
                        <input type="radio" name="opt_color" value="influence"
autocomplete="off" >User Influence
                      </label>
                    </div>
                    <label for="opt_distances">Other</label>
                    <button type="button" class="btn btn-primary btn-block" data-
toggle="button" id="opt_distances" aria-pressed="false" autocomplete="off">Display
distances
                    </button>
```

```
                </div>
              </div>
            </div>

            <div class="panel panel-default">
              <div class="panel-heading" role="tab" id="heading2">
                <h4 class="panel-title">
                  <a class="collapsed" data-toggle="collapse" data-parent="#accordion"
href="#collapse2" aria-expanded="false" aria-controls="collapse2">
                    Tweet Languages
                  </a>
                </h4>
              </div>
              <div id="collapse2" class="panel-collapse collapse" role="tabpanel" aria-
labelledby="heading2">
                <div class="panel-body">
                  <div class="btn-group btn-group-justified" role="group">
                    <div class="btn-group" role="group">
                      <button type="button" class="btn btn-default"
id="btn_lang_all">All</button>
                    </div>
                    <div class="btn-group" role="group">
                      <button type="button" class="btn btn-default"
id="btn_lang_none">None</button>
                    </div>
                  </div>
                  <div id="opt_langs_list">
                  </div>
                </div>
              </div>
            </div>
            <div class="panel panel-default">
              <div class="panel-heading" role="tab" id="heading3">
                <h4 class="panel-title">
                  <a class="collapsed" data-toggle="collapse" data-parent="#accordion"
href="#collapse3" aria-expanded="false" aria-controls="collapse3">
                    User Influence Groups
                  </a>
                </h4>
              </div>
              <div id="collapse3" class="panel-collapse collapse" role="tabpanel" aria-
labelledby="heading3">
                <div class="panel-body">
                  <div class="btn-group btn-group-justified" role="group">
                    <div class="btn-group" role="group">
                      <button type="button" class="btn btn-default"
id="btn_inf_all">All</button>
                    </div>
                    <div class="btn-group" role="group">
```

```
                        <button type="button" class="btn btn-default"
id="btn_inf_none">None</button>
                      </div>
                    </div>
                    <div class="input-group">
                      <span class="input-group-addon">
                        <input type="checkbox" name="opt_influence" value="0"
autocomplete="off" >
                      </span>
                      <div class="form-control" id="inf_box_0">1 - 25</div>
                    </div>
                    <div class="input-group">
                      <span class="input-group-addon">
                        <input type="checkbox" name="opt_influence" value="1"
autocomplete="off" >
                      </span>
                      <div class="form-control" id="inf_box_1">25 - 50</div>
                    </div>
                    <div class="input-group">
                      <span class="input-group-addon">
                        <input type="checkbox" name="opt_influence" value="2"
autocomplete="off" >
                      </span>
                      <div class="form-control" id="inf_box_2">50 - 75</div>
                    </div>
                    <div class="input-group">
                      <span class="input-group-addon">
                        <input type="checkbox" name="opt_influence" value="3"
autocomplete="off" >
                      </span>
                      <div class="form-control" id="inf_box_3">75 - 100</div>
                    </div>
                  </div>
                </div>
              </div>
              <div class="panel panel-default">
                <div class="panel-heading" role="tab" id="heading4">
                  <h4 class="panel-title">
                    <a class="collapsed" data-toggle="collapse" data-parent="#accordion"
href="#collapse4" aria-expanded="false" aria-controls="collapse4">
                    Filters
                    </a>
                  </h4>
                </div>
                <div id="collapse4" class="panel-collapse collapse" role="tabpanel" aria-
labelledby="heading4">
                  <div class="panel-body">
                    <div class="panel panel-default">
                      <div class="panel-heading" role="tab" id="heading5">
```

```
                        <h4 class="panel-title">
                          <a class="collapsed" data-toggle="collapse" href="#collapse5" aria-
expanded="false" aria-controls="collapse5">
                            Top 10 hashtags
                          </a>
                        </h4>
                      </div>
                      <div id="collapse5" class="panel-collapse collapse" role="tabpanel"
aria-labelledby="heading5">
                        <div class="list-group" id="hashtags_list"></div>
                      </div>
                    </div>
                    <label for="opt_grp_color">Search for</label>
                    <div class="input-group" id="opt_grp_hashtag">
                      <span class="input-group-addon">#</span>
                      <div class="form-control"><div id="opt_hashtag"></div><button
id="delete_hashtag" type="button" class="close" aria-label="Close"><span aria-
hidden="true">&times;</span></button></div>
                    </div>
                  </div>
                </div>
              </div>
            </div>
            <div class="panel panel-default">
              <div class="panel-heading">
                <h3 class="panel-title">Charts</h3>
              </div>
              <div class="panel-body">
                <label for="chart_grp_tpl">Tweets per language</label>
                <div class="btn-group btn-group-justified" id="chart_grp_tpl" role="group">
                  <div class="btn-group" role="group">
                    <button type="button" id="btnPieChart" class="btn btn-primary"
>Pie</button>
                  </div>
                  <div class="btn-group" role="group">
                    <button type="button" id="btnColumnChart" class="btn btn-primary"
>Column</button>
                  </div>
                </div>
              </div>
            </div>
          </div>
          <div id="counter"><div id="points">0</div>points</div>
          <div id="map"></div>
          <div class="modal fade" id="modalChart" tabindex="-1" role="dialog" aria-
hidden="true">
            <div class="modal-dialog modal-lg">
              <div class="modal-content">
                <div class="modal-header">
```

```
                <button type="button" class="close" data-dismiss="modal" aria-
label="Close"><span aria-hidden="true">&times;</span></button>
                <h4 class="modal-title">Statistics</h4>
              </div>
              <div class="modal-body">
                <div id="chart"></div>
              </div>
              <div class="modal-footer">
                <a id="btnDownloadChart" class="btn btn-default" download>Download</a>
              </div>
            </div>
          </div>
        </div>
      </div>
      <footer>
        <div id="footer" > &copy; 2015 <a target="_blank"
href="https://www.linkedin.com/in/msiganos">Marios Siganos</a>. All rights
reserved.</div>
      </footer>
  </body>
</html>
```

Κώδικας Javascript για τον εξυπηρετούμενο

```
var marker_colors = {};
var serverData = [];
var markers = [];
var appOptions = {};
var influence_groups = 4;
var clusters = {};
clusters["languages"] = {};
clusters["influence"] = {};
var heatmap = {};
heatmap["languages"] = {};
heatmap["influence"] = {};
var heatmapData = {};
heatmapData["languages"] = {};
heatmapData["influence"] = {};
var capitalCircles = {};
var distancesAverages = {};
var tweetStats = {};
var chartData;
var chart;
var chartRefresh;

function setChartData(options) {
  var arrayToDataTable = [];
  arrayToDataTable.push(['Language', 'Tweets']);
```

```
   for (var i in options["languages"]){
      arrayToDataTable.push([languages[options["languages"][i]]["name"],  tweetStats[
options["languages"][i] ] ]);
   }
   chartData = google.visualization.arrayToDataTable(arrayToDataTable);
}

function initChart(type) {

         setChartData(appOptions);
   var options;
   switch (type){
      case "pie":
         chart = new google.visualization.PieChart( $('#chart').get(0) );
         options = {
            title : 'Tweets per language',
                  is3D: true,
               };
         break;
      case "column":
         chart = new google.visualization.ColumnChart( $('#chart').get(0) );
         options = {
                  title : 'Tweets per language',
                  hAxis : { "title" : "Languages"},
                  vAxis : { "title" : "# of Tweets"},
                  legend : { "position" : "none"}
         };
         break;
   }
         refreshChart(options);
}

function refreshChart(chartOptions) {
   setChartData(appOptions);
         chart.draw(chartData, chartOptions);
         $("#btnDownloadChart").attr("href", chart.getImageURI());
         chartRefresh = setTimeout( function(){
      refreshChart(chartOptions)
         } , 1000);
}


google.maps.event.addDomListener(window, 'load', initialize);

$(document).ready(function() {

   google.load('visualization', '1', {packages:['corechart'], callback:function() {}});

   for (var i in chosenLanguages){
```

```
        tweetStats[chosenLanguages[i]] = 0;
    }

    var max_langs = 10;

    for (var i in chosenLanguages){
        var lang = chosenLanguages[i];
        $('#opt_langs_list').append("<div class=\"input-group\"><span class=\"input-group-
addon\"><input type=\"checkbox\" name=\"opt_languages\" value=\"" + lang + "\"
id=\"opt_lang_" + lang + "\"></span><div class=\"form-control\" data-color-id=\"-1\"
id=\"lang_box_" + lang + "\" ><img src=\"../static/images/flags/" + lang + ".png\"
class=\"flag\">" + languages[lang]['name'] + "</div>" +  "</div>");
    }

    $('#btnPieChart').click(function(){
        $('#chart').empty();
        $("#btnDownloadChart").attr("href","");
        if(appOptions["languages"].length!=0){
            $('#modalChart').modal('show');
            setTimeout( function(){
                initChart("pie");
            },500);
        }
    });

    $('#btnColumnChart').click(function(){
        $('#chart').empty();
        if(appOptions["languages"].length!=0){
            $('#modalChart').modal('show');
            setTimeout( function(){
                initChart("column");
            },500);
        }
    });

    $('#modalChart').on('hidden.bs.modal', function (e) {
        clearTimeout(chartRefresh);
    });


    // Listeners
    $('#btnUpdate').on('click', function () {
        var $btn = $(this).button('updating');
        setTimeout(function () {
        updateMap();
        $btn.button('reset');
        },100);

    });
```

```
    $('#btnMenu').click(function(){
        $("#sidebar").toggle("slow");
    });

    $("#delete_hashtag").click(function(){
        $("#opt_hashtag").empty();
    });

    $("input:checkbox[name=opt_languages]").change(function(data) {

        if (this.checked) {
            var count = $("input:checkbox[name=opt_languages]:checked").length;

            var color_id;
            var found;
            var langColors = getColorIds();
            for(var i in colorGradients){
                color_id = i;
                found = false;
                for(var lang in langColors){
                    if (i == langColors[lang]){
                        found = true;
                        break;
                    }
                }
                if (found == false){
                    break;
                }
            }
            $("#lang_box_" + this.value).css('background-color',
colorGradients[color_id][colorGradients[color_id].length-1]);//.css('color','#fff');
            $("#lang_box_" + this.value).data("color-id",color_id);

            if(count==max_langs){
                disableCheckboxes("opt_languages",true);
            }
        }else{
            $("#lang_box_" + this.value).data("color-id",-1);
            $("#lang_box_" + this.value).css('background-color', "#fff");
            var count = $("input:checkbox[name=opt_languages]:checked").length;
            disableCheckboxes("opt_languages",false);
        }
            });

    $("input:checkbox[name=opt_influence]").change(function(data) {
        if (this.checked) {
            $("#inf_box_" + this.value).css('background-color',
colorGradientsInf[this.value][colorGradientsInf[this.value].length-1]);
```

```
        }else{
            $("#inf_box_" + this.value).css('background-color', "#fff");
        }
            });

    $('#btn_lang_all').click(function(){
             changeCheckboxesState("opt_languages", true, true);
            });

            $('#btn_lang_none').click(function(){
             changeCheckboxesState("opt_languages", false, true);
            });

    $('#btn_inf_all').click(function(){
             changeCheckboxesState("opt_influence",true, true);
            });

    $('#btn_inf_none').click(function(){
        changeCheckboxesState("opt_influence", false, true);
            });

    changeCheckboxesState("opt_influence", true, true);

    appOptions = getOptions();

});

function disableCheckboxes(name,state){
    $("input:checkbox[name=" + name + "]").each(function() {
                if (this.checked == false) {
                $(this).prop("disabled", state);
            }
    });
}

function changeCheckboxesState(name, state, change){
    if (change === undefined) {
         change = true;
    }

    $("input:checkbox[name=" + name + "]").each(function() {
                if (this.checked != state && this.disabled == false) {
                $(this).prop("checked", state);
                if(change){
                    $(this).change();
                }
            }
    });
}
```

```
function getOptions(){
    var options = {};
    options["view"] = getOptionsView();
    options["color"] = getOptionsColor();
    options["distances"] = getOptionsDistances();
    options["languages"] = getOptionsLanguages();
    options["influence"] = getOptionsInfluence();
    options["tweets"] = getOptionsMaxTweets();
    options["hashtag"] = getOptionsHashtag();
    options["color_ids"] = getColorIds();
    return options;
}

function getOptionsView(){
    return $("input:radio[name=opt_view]:checked").val();
}

function getOptionsColor(){
    return $("input:radio[name=opt_color]:checked").val();
}

function getOptionsDistances(){
    return $("#opt_distances").attr("aria-pressed");
}

function getOptionsLanguages(){
    var data = [];
    $("input:checkbox[name=opt_languages]").each(function() {
            if (this.checked) {
            data.push( this.value );
        }
    });
    return data;
}

function getColorIds(){
    var data = {};
    $("input:checkbox[name=opt_languages]").each(function() {
            if (this.checked) {
            data[this.value] = $("#lang_box_" + this.value).data("color-id");
        }
    });
    return data;
}

function getOptionsInfluence(){
    var data = [];
    $("input:checkbox[name=opt_influence]").each(function() {
```

```
            if (this.checked) {
            data.push( parseInt(this.value) );
        }
    });
    return data;
}

function getOptionsMaxTweets(){
    return $('#opt_max_tweets').val();
}

function getOptionsHashtag(){
    return $('#opt_hashtag').text();
}

function getInfluenceGroup(influence){
    if (influence <= 25 ){
        return 0;
    }else if (influence <= 50){
        return 1;
    }else if (influence <= 75){
        return 2;
    }else{
        return 3;
    }
}

function getColorGradient(options, lang, infGroup){
    var gradient = [];
    if (options["color"] == "languages"){
        var id = options["color_ids"][lang];
        gradient = colorGradients[id];
    }else{
        gradient = colorGradientsInf[infGroup];
    }
    return gradient;
}

function getMarkerColor(options,lang,influence){
    var infGroup = getInfluenceGroup(influence);
    var gradient = getColorGradient(options, lang, infGroup);
    return gradient[gradient.length-1];
}

function updateCounter(options){
    if (options["view"] == "heatmap"){
        var points_sum = 0;
        if (options["color"] == "languages"){
            for (var i in options["languages"]){
```

```
            points_sum += heatmapData["languages"][options["languages"][i]].getLength();
        }
    }else{
        for (var i=0;i<options["influence"].length;i++){
            points_sum += heatmapData["influence"][options["influence"][i]].getLength();
        }
    }
    $("#points").text(points_sum);
}else{
    $("#points").text(markers.length);
}
}

function updateHashtags(opts){
    var hashtags = {};
    for (var i in serverData){
        var data = serverData[i];
        var influenceGroup = getInfluenceGroup(data["influence"]);
        if (opts["influence"].indexOf(influenceGroup) > -1 &&
opts["languages"].indexOf(data["lang"]) > -1){

            if ( opts["hashtag"] != "" && data["hashtags"].indexOf( opts["hashtag"]) == -1){
                continue;
            }

            var tweet_hashtags = data["hashtags"];

            for (var j in tweet_hashtags ){
                var hashtag = tweet_hashtags[j];
                if (hashtag in hashtags){
                    hashtags[hashtag] += 1;
                }
                else{
                    hashtags[hashtag] = 1;
                }
            }
        }
    }

    var hashtags_sorted = [];

    for (var hashtag in hashtags)
        hashtags_sorted.push([hashtag, hashtags[hashtag]]);

    hashtags_sorted.sort(function(a, b) {return b[1] - a[1]});
    top_hashtags = hashtags_sorted.slice(0,10)
    var list_items = "";

    for (var i in top_hashtags ){
```

```
        list_items += "<a href=\"#\" class=\"list-group-item hashtag-item\" data-hashtag=\"" +
top_hashtags[i][0] + "\"><span class=\"badge\">" + top_hashtags[i][1] + "</span>" +
top_hashtags[i][0] + "</a>";
    }

    $('#hashtags_list').html(list_items);
    $( ".hashtag-item" ).click(function() {
        $("#opt_hashtag").text($(this).data('hashtag'));
    });

}

function initialize() {
            initMap();
            initSSEs();
}

function initMap(){

    var mapOptions = {
                    zoom : 2,
                    center : new google.maps.LatLng(0, 0),
        mapTypeControl: false,
                    panControl : false,
                    streetViewControl : false,
        styles : ShadesOfGrey,
            };

            map = new google.maps.Map( $('#map').get(0), mapOptions);
            map.controls[google.maps.ControlPosition.RIGHT_TOP].push( $('#counter').get(0) );
}

function addMarker(data,options){

    var influenceGroup = getInfluenceGroup(data["influence"]);

    if ( options["hashtag"] != "" && data["hashtags"].indexOf( options["hashtag"]) == -1){
        return;
    }

    if (options["influence"].indexOf(influenceGroup) == -1 ||
options["languages"].indexOf(data["lang"]) == -1){
        return;
    }

    var myLatlng = new google.maps.LatLng(data['geo'][1], data['geo'][0]);

    if (options["view"] == "markers" || options["view"] == "clusters"){
        var color = getMarkerColor(options,data["lang"],data["influence"]);
```

```
        var marker = new google.maps.Marker({
            position: myLatlng,
            icon: {
                path: google.maps.SymbolPath.CIRCLE,
                fillColor: color,
                strokeColor: color,
            },
        });
        var user_link = 'https://twitter.com/' + data['uname'];
        var tweet_link = 'https://twitter.com/' + data['uname'] + '/status/' + data['id'];
        var row1 = "<tr><th colspan=\"2\"><a target=\"_blank\" href=\"" + tweet_link +
"\">Tweet</a></th></tr>";
        var row2 = "<tr><th>User</th><td><a target=\"_blank\" href=\"" + user_link + "\">" +
data['uname'] + "</a></td></tr>";
        var row3 = "<tr><th>Language</th><td>" + languages[data["lang"]]['name'] +
"</td></tr>";
        var row4 = "<tr><th>Influence</th><td>" + data["influence"] + "</td></tr>";
        var rows = row1 + row2 + row3 + row4;
        var table = "<table class=\"table table-bordered\">" + rows + "</table>";
        contentString = table;
        var infowindow = new google.maps.InfoWindow({
            content: contentString
        });
        google.maps.event.addListener(marker, 'click', function() {
            infowindow.open(map,marker);
        });
        markers.push(marker);
    }

    var key = null;
    if (options["color"] == "languages"){
        key = data["lang"];
    }else{
        key = influenceGroup;
    }

    switch (options["view"]){
        case "heatmap":
            heatmapData[options["color"]][key].push(myLatlng);
            heatmap[options["color"]][key].setData(heatmapData[options["color"]][key]);
            break;
        case "markers":
            marker.setMap(map);
            break;
        case "clusters":
            clusters[options["color"]][key].addMarker(marker);
            break;
    }
```

```
    if(options["distances"] == "true"){
        distancesAverages[data["lang"]]["sum"] += data["distance"];
        distancesAverages[data["lang"]]["count"] += 1;
        capitalCircles[data["lang"]].setRadius(distancesAverages[data["lang"]]["sum"] /
distancesAverages[data["lang"]]["count"] * 100 );
    }

    updateCounter(options);
    updateHashtags(options);
}

function clearMarkers(){
    for (var i in markers){
        markers[i].setMap(null);
    }
    markers = [];
}

function initClusters(options){
    if (options["color"] == "languages"){
        for (var i in options["languages"]){
            initCluster(options["languages"][i],
getMarkerColor(options,options["languages"][i],null), options);
        }
    }else{
        for (var i=0;i<options["influence"].length;i++){
            var grad = getColorGradient(options, null, options["influence"][i]);
            initCluster(options["influence"][i], grad[grad.length-1], options);
        }
    }
}

function initCluster(key,color,options){
    var mcOptions = {
        styles : [
        {
            width: 53,
            height: 53,
            averageCenter : true,
            url: "static/images/cluster-icons/m1/" + color.substring(1) + ".png"
        }],
        gridSize: 50,
        maxZoom: 15
    };
    var mc = new MarkerClusterer(map, [], mcOptions);
    clusters[options["color"]][key] = mc;
}

function clearClusters(){
```

```
    for (var lang in clusters["languages"]){
        clusters["languages"][lang].clearMarkers();
    }

    for (var i in clusters["influence"]){
        clusters["influence"][i].clearMarkers();
    }

    clusters["languages"] = {};
    clusters["influence"] = {};

}

function initHeatmap(options){

    if (options["color"] == "languages"){
        for (var i in options["languages"]){
            initHeatmapLayer(options["languages"][i],options);
        }
    }else{
        for (var i=0;i<options["influence"].length;i++){
            initHeatmapLayer(options["influence"][i],options);
        }
    }

}

function initHeatmapLayer(key,options){
    heatmapData[options["color"]][key] = new google.maps.MVCArray();
    heatmap[options["color"]][key] = new google.maps.visualization.HeatmapLayer({
                    data : heatmapData[options["color"]][key],
                    gradient : ["rgba(255, 255, 255, 0)"].concat( getColorGradient(options, key,
key).slice(3) ),
                    map: map,
    });
}

function clearHeatmap(){
    for (var i in heatmapData["languages"]){
        heatmapData["languages"][i].clear();
        heatmap["languages"][i].setMap(null);
    }

    for (var i in heatmapData["influence"]){
        heatmapData["influence"][i].clear();
        heatmap["influence"][i].setMap(null);
    }
```

```
    heatmapData["languages"] = {};
    heatmap["languages"] = {};
    heatmapData["influence"] = {};
    heatmap["influence"] = {};
}

function calculateDistancesAverage(lang,selectedData){

        var sum = 0;
        var length = 0;
        for (var i in selectedData){
            var data = selectedData[i];
            var data_distance = data["distance"];
            var data_lang = data["lang"];
            if (data_lang == lang){
                sum += parseFloat(data_distance);
                length +=1;
            }
        }
        return (sum / length);
}

function initCircles(options){
    for(var i in options["languages"]){
        var lang = options["languages"][i];
        var cap_lat = languages[lang]['capital']['geo'][0];
        var cap_lon = languages[lang]['capital']['geo'][1];
        var populationOptions = {
          strokeColor: colorGradients[ options["color_ids"][lang] ][ colorGradients[
options["color_ids"][lang] ].length-1 ],
          strokeOpacity: 0.8,
          strokeWeight: 2,
          fillColor : colorGradients[ options["color_ids"][lang] ][ colorGradients[
options["color_ids"][lang] ].length-1 ],
          fillOpacity: 0.1,
          clickable: false,
          center: new google.maps.LatLng(cap_lat, cap_lon),
        };
        distancesAverages[lang] = {};
        distancesAverages[lang]["sum"] = 0;
        distancesAverages[lang]["count"] = 0;
        capitalCircles[lang] = new google.maps.Circle(populationOptions);
        capitalCircles[lang].setMap(map);
    }
}

function clearCircles(){
    for (var i in capitalCircles){
        capitalCircles[i].setMap(null);
```

```
    }

    distancesAverages = {};
    capitalCircles = {};
}

function updateMap(){
    $("#points").text(0);
    $('#hashtags_list').empty();
    appOptions = getOptions();
    clearHeatmap();
    clearMarkers();
    clearClusters();
    clearCircles();
    switch (appOptions["view"]){
        case "heatmap":
            initHeatmap(appOptions);
            break;
        case "markers":
            break;
        case "clusters":
            initClusters(appOptions);
            break;
    }

    if(appOptions["distances"] == "true"){
        initCircles(appOptions);
    }

    for (var i in serverData){
        var data = serverData[i];
        addMarker(data,appOptions);
    }
}

function processData(data){

    var parsedData = {};
    parsedData["lang"] = data['c_lang'];
    parsedData["geo"] = data['t_geo'];
    parsedData["influence"] = data['c_fwsa'];
    parsedData["uname"] = data['t_uname'];
    parsedData["id"] = data['t_id'];
    parsedData["hashtags"] = [];
    parsedData["distance"] = data["c_distance"];

    for (var i in data["t_hashtags"]){
        parsedData["hashtags"].push( data["t_hashtags"][i].toLowerCase() );
    }
```

```
        if (parsedData["geo"].length > 0 && chosenLanguages.indexOf( parsedData["lang"] )
> -1 && parsedData["influence"] != -1) {
      serverData.push(parsedData);
      addMarker(parsedData, appOptions);
      tweetStats[parsedData["lang"]] += 1;
      $('#tweetCounter').text(serverData.length);
        }
}

function initSSEs() {

        if (!!window.EventSource) {
                var source = new EventSource('/data');
        } else {
        return;
        }

        source.addEventListener('message', function(e) {

      // Check origin of messages
      if (e.origin != '  && e.origin != '') {
        console.log('Origin unknown');
        return;
      }

                var data = JSON.parse(e.data);
      processData(data);


        }, false);

        source.addEventListener('open', function(e) {

                console.log("Connection was opened");
        }, false);

        source.addEventListener('error', function(e) {

                if (e.eventPhase == EventSource.CLOSED) {
                        console.log("Connection was closed");
                }

                switch( e.target.readyState ){
        case EventSource.CONNECTING:
          console.log("Reconnecting...");
          break;
        case EventSource.CLOSED:
          console.log("Connection failed. Will not retry.");
```

```
            break;
        }
        }, false);
}
```

Κώδικας CSS για τον εξυπηρετούμενο

```
html,body {
    height: 100%;
    margin: 0px;
}

body{
    background-color: #f8f8f8;
}

header {
    height: 30px;
    width: 100%;
    text-align: center;
}

#header {
    font-size: 20px;
}

#btnHome{
    float: left;
    margin-left: 5px;
}

#btnMenu{
    float: right;
    margin-left: 5px;
}

footer {
    height: 20px;
    width: 100%;
}

#main {
    width: 100%;
    height: calc(100% - 30px - 20px);
    border-bottom: 2px solid;
    border-top: 2px solid;
}
```

```
#map {
   height: 100%;
}

#sidebar {
   height: 100%;
   width: 250px;
   float: right;
   border-left: 5px solid;
   overflow: scroll;
   padding: 5px;
}

a {
   color: inherit;
   text-decoration: none;
}

a:hover {
   color: maroon;
}

#footer {
   text-align: center;
}

.menu {
   text-align: center;
   border-bottom: 1px solid;
}

.flag{
   width: 15px;
   height: 15px;
   border: 1px solid #555;
   border-radius: 50%;
   margin-left: 2px;
   margin-right: 5px;
   vertical-align: middle;
}

.opt-group {
   display: block;
}

#counter {
   font-size: medium;
   background-color: rgba(255,255,255,0.7);
   border-radius: 7px;
```

```
    padding: 2px 5px 2px 5px;
    margin: 5px;
    display: inline;
}

#points {
    float: left;
    margin-right: 2px;
}

.input-group{
    margin-top: 2px;
    margin-bottom: 2px;
}

.btn:focus, .btn:active:focus, .btn.active:focus {
        outline: 0 none;
}

#delete_hashtag {
    float: right;
}

#opt_hashtag {
    float: left;
}

#chart {
    height: 400px;
    display: block;
    margin: 0 auto;
}

.table{
    text-align: center;
}

#btnUpdate {
    margin-bottom: 20px;
}

#tweetCounter {
    margin-left: 5px;
    float: right;
}
```