Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

« Προηγμένα Συστήματα Πληροφορικής »

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Δημιουργία web εφαρμογής στο κοινωνικό δίκτυο Twitter με χρήση Javascript και διασύνδεση με το REST και Streaming API** **Creating a web application in Twitter social network using Javascript and integration with REST and Streaming API**
Ονοματεπώνυμο Φοιτητή	Φατούρος Ιωάννης
Πατρώνυμο	Αγγελής
Αριθμός Μητρώου	ΜΠΣΠ 13118
Επιβλέπων	Αλέπης Ευθύμιος

1. ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω την οικογένειά μου, τη μητέρα μου Εριφύλη, τον πατέρα μου Αγγελή και τον αδελφό μου Βασίλη, για την αμέτρητη υποστήριξη που μου προσέφεραν σε όλη τη διάρκεια της φοίτησής μου. Η στήριξή τους τόσο ψυχολογικά όσο και οικονομικά με βοήθησε να ολοκληρώσω της φοιτητικές μου υποχρεώσεις και την μεταπτυχιακή διατριβή με επιτυχία. Ιδιαίτερες ευχαριστίες στον επιβλέποντα καθηγητή κύριο Ευθύμιο Αλέπη για την δυνατότητα που μου έδωσε να ασχοληθώ με το τομέα των web εφαρμογών και τον χρόνο που μου αφιέρωσε, καθώς και σε όλους τους καθηγητές του τμήματος Πληροφορικής.

2. ΠΕΡΙΛΗΨΗ

Σκοπός της μεταπτυχιακής διατριβής είναι να ερευνήσει πως δημιουργείται μια εφαρμογή στο κοινωνικό δίκτυο Twitter, να πραγματοποιήσει μια ανασκόπηση και να βρει άλλες εφαρμογές που έχουν αναπτυχθεί για το σκοπό αυτό και τέλος να δημιουργήσει μια εφαρμογή διαδικτύου που θα εκμεταλλεύεται το συγκεκριμένο κοινωνικό δίκτυο για να λάβει δεδομένα, και να εξάγει συμπεράσματα. Στο πλαίσιο αυτό, η εφαρμογή που έχει υλοποιηθεί μπορεί να χρησιμοποιηθεί από άτομα για να μπορούν να χρησιμοποιούν μαζικά ένα λογαριασμό και να μπορούν να πραγματοποιούν δημοσιεύσεις και οποιαδήποτε άλλη ενέργεια εκ μέρους του χρήστη – ιδιοκτήτη του λογαριασμού του κοινωνικού δικτύου. Τα στατιστικά που παρέχονται δίνουν τη δυνατότητα στους χρήστες να παρακολουθήσουν την κίνηση – αριθμό δημοσιεύσεων που πραγματοποιείται με βάση το hashtag που έχουν δημιουργήσει καθώς και τις συσκευές που χρησιμοποιούν οι χρήστες του κοινωνικού δικτύου για το σκοπό αυτό.

3. ABSTRACT

The purpose of the dissertation is to investigate how to create an application on the Twitter social network, to conduct a review and find other similar applications developed for this purpose and finally to create an internet web application that will receive data and draw conclusions. In this context, the implemented application can be used by individuals to be able to use an account in bulk and be able to make publications and any action on behalf of the user – owner of the Twitter account. The statistics provided enable users to track the traffic – number of publications made on the basis of the hashtag they have created and the devices used by social network users for that purpose.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

4. ΕΙΣΑΓΩΓΗ

Η εφαρμογή δίνει τη δυνατότητα σε χρήστες να μπορούν να χρησιμοποιούν ταυτόχρονα ένα λογαριασμό στο κοινωνικό δίκτυο του Twitter. Οι χρήστες είναι σε θέση να ενεργούν εκ μέρους του λογαριασμού αυτού και να πραγματοποιήσουν οποιαδήποτε ενέργεια, π.χ να δημοσιεύσουν κάποιο tweet, να προγραμματίσουν τη δημοσίευση οποιουδήποτε tweet στο μέλλον. Επίσης παρέχεται η δυνατότητα να παρακολουθήσουν σε πραγματικό χρόνο tweets με βάση κάποια λέξη που επιθυμούν ή με βάση κάποιο hashtag. Τέλος οι χρήστες μπορούν να δουν σε μορφή διαγραμμάτων την κίνηση που έχει δημιουργηθεί γύρω από κάποιο hashtag – αριθμός δημοσιευμένων tweets / χρόνο, καθώς και το μέσο που έχουν χρησιμοποιήσει για το σκοπό αυτό.

5. ΑΝΑΣΚΟΠΗΣΗ ΠΕΔΙΟΥ

Μετά από έρευνα για εργαλεία και εφαρμογές που χρησιμοποιούν το Twitter καταλήγουμε στο συμπέρασμα ότι υπάρχουν αρκετά τέτοια εργαλεία που έχουν σχεδιαστεί για να παρουσιάσουν ένα διαφορετικό τρόπο οπτικοποίησης ή ανάλυσης των tweets των χρηστών, των ανθρώπων στο δίκτυό και των tweets από τα άτομά του δικτύου.

Πολλά πέτυχαν το σκοπό αυτό και αρκετά έχουν αποτύχει. Τα παρακάτω εργαλεία, ωστόσο είναι ευρέως γνωστά και αρκετά επιτυχημένα και αξίζει να τα σημειώσουμε.

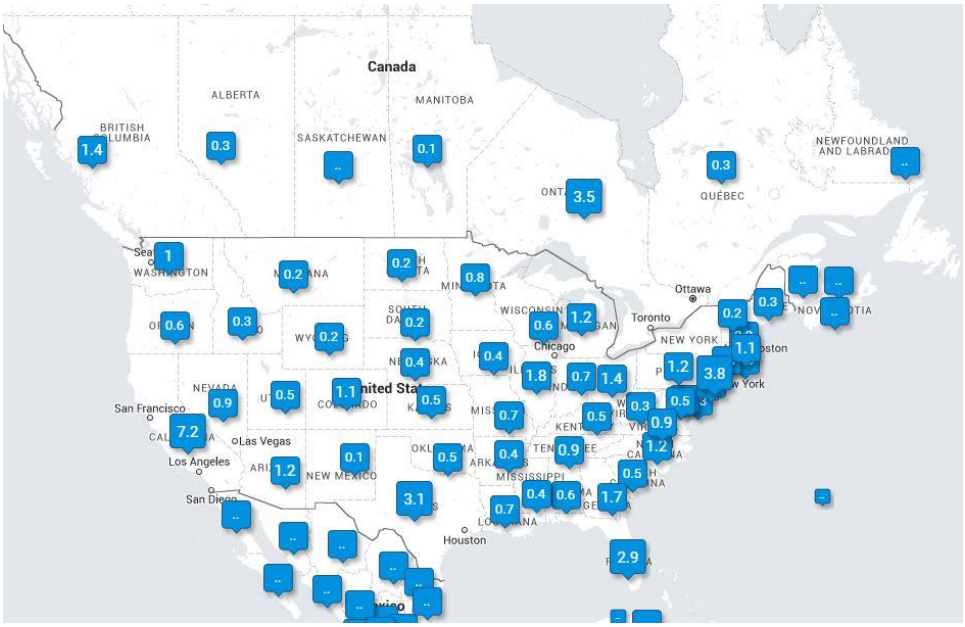
1. TweetsMap

Το TweetsMap είναι ένα χρήσιμο εργαλείο Twitter για την ανάλυση και την απεικόνιση του δικτύου Twitter ενός χρήστη.

Όπως δηλώνει και το όνομά του, παρουσιάζει πως κατανέμονται οι ακόλουθοι ενός χρήστη στο χάρτη, σε ποσοστά.

Εικόνα 1 User interface από την εφαρμογή.

Τα δεδομένα αυτά μπορούν να προβληθούν και σε επίπεδο επαρχίας ή πολιτείας ή ακόμα και σε επίπεδο πόλης.



Εικόνα 2 Κατανομή των tweets ανά επαρχία.

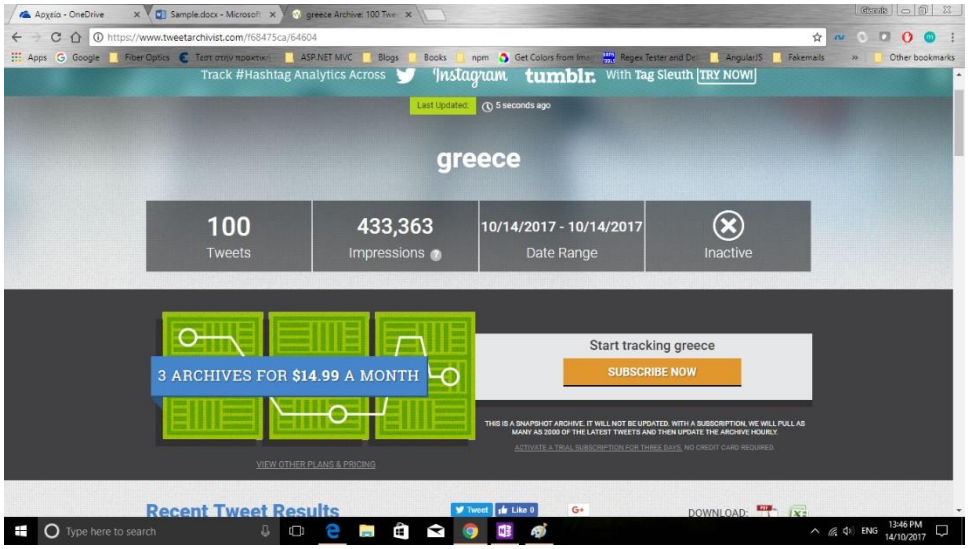
Επιπρόσθετα, η συγκεκριμένη εφαρμογή έχει τη δυνατότητα:

- Απλής δημοσίευση ενός tweet μέσω της πλατφόρμας.
- Χρονοπρογραμματισμός Tweet, να ορίσει δηλαδή ο χρήστης ένα tweet και την ώρα που θέλει και να γίνει αυτόματα, χωρίς να χρειάζεται να είναι συνδεδεμένος στην εφαρμογή εκείνη τη στιγμή.
- Διαγράμματα που παρουσιάζουν την αύξηση ή πτώση ανά περιοχή, στον αριθμό των ακολούθων, tweets κτλ.
- Πλήρης ανάλυση οποιουδήποτε λογαριασμού χρήστη.

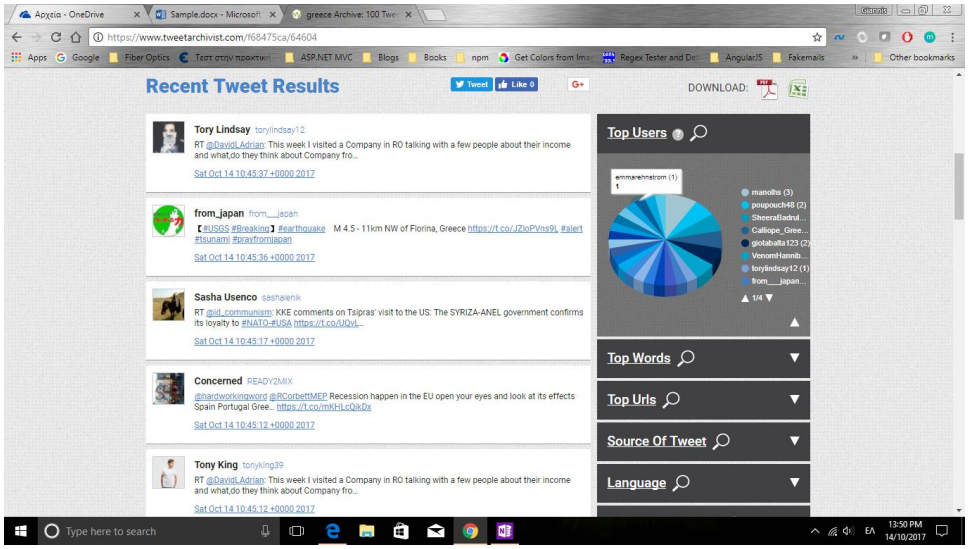
2. Tweet Archivist

Ορισμένα από τα χαρακτηριστικά της συγκεκριμένης εφαρμογής είναι:

- Ψάξιμο στο Twitter, είτε μιας λέξης, είτε ενός hashtag ή ακόμα και μιας ερώτησης Twitter.
- Παρουσίαση αποτελεσμάτων.
- Γραφική απεικόνιση με ποσοστό δημοσίευσης ανά χρήστη, συσκευή που χρησιμοποίησε (android, iphone, web, ή μέσω κάποιας πλατφόρμας), γλώσσα, hashtags, mentions κτλ.
- Καταχώρηση μιας συγκεκριμένης λέξης και αποθήκευση όλων των tweets που σχετίζονται με αυτή.
- Ελεύθερη χρήση της εφαρμογής με περιορισμένες λειτουργίες.



Εικόνα 3 Αρχική οθόνη της εφαρμογής Tweet Archivist



Εικόνα 4 Αναζήτηση και προβολή αποτελεσμάτων από την εφαρμογή Tweet Archivist.

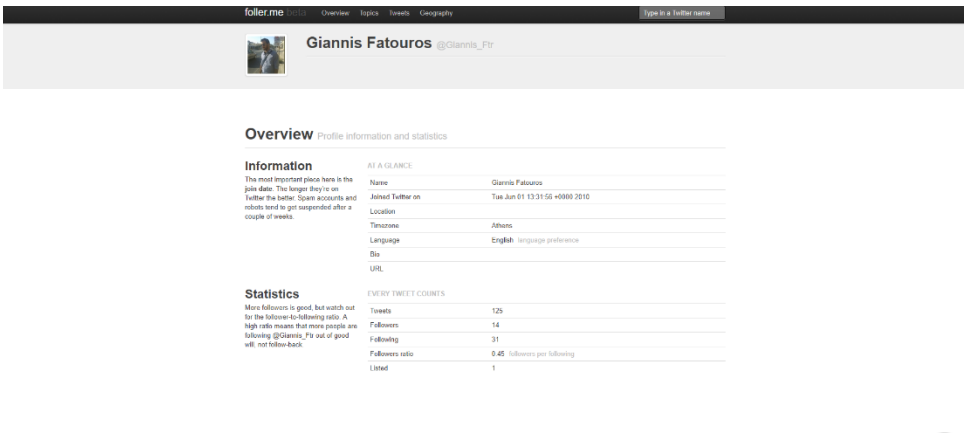
3. Follor.me

Ιστοσελίδα: https://foller.me

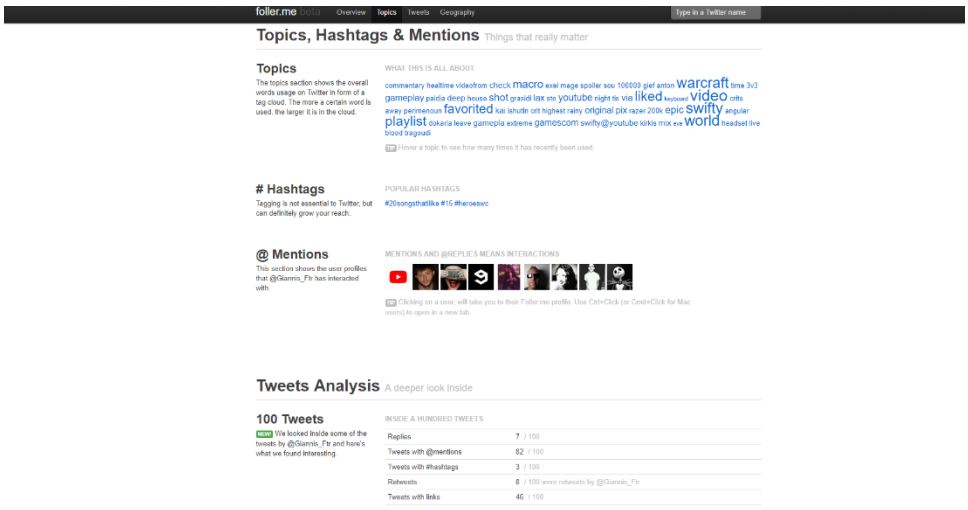
Είναι μια δωρεάν εφαρμογή που επιτρέπει γρήγορη ανάλυση ενός χρήστη του Twitter, με παρουσίαση στατιστικών που αφορούν:

- Κατηγορίες θεμάτων με βάση τις λέξεις που χρησιμοποιεί όταν δημοσιεύει.
- Hashtags που χρησιμοποιεί.
- Mentions άλλων χρηστών.

- Περιλαμβάνει τις ώρες της ημέρας που συνήθως επιλέγει ένας χρήστης για να δημοσιεύση κάποιο tweet.
- Κατανομή των τελευταίων 100 tweet.



Εικόνα 5 Αρχική οθόνη από την εφαρμογή Foller.me.



Εικόνα 6 Προβολή αποτελεσμάτων.(α)

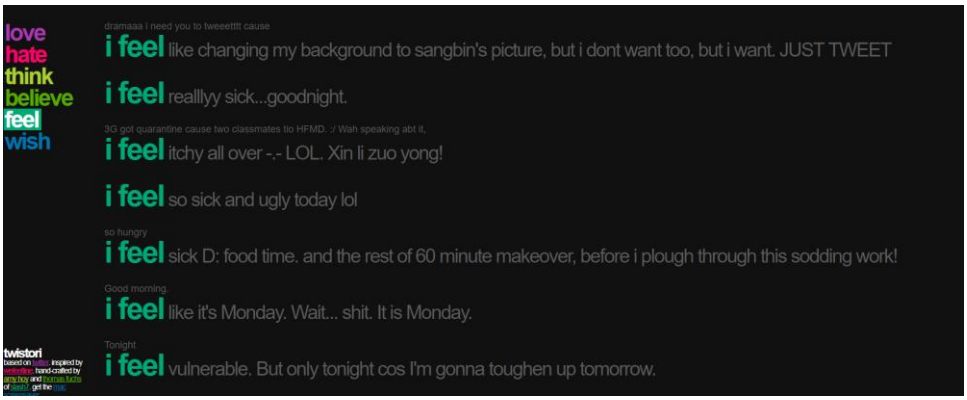
Εικόνα 7 Προβολή αποτελεσμάτων. (β) συνέχεια.

4. Twistori

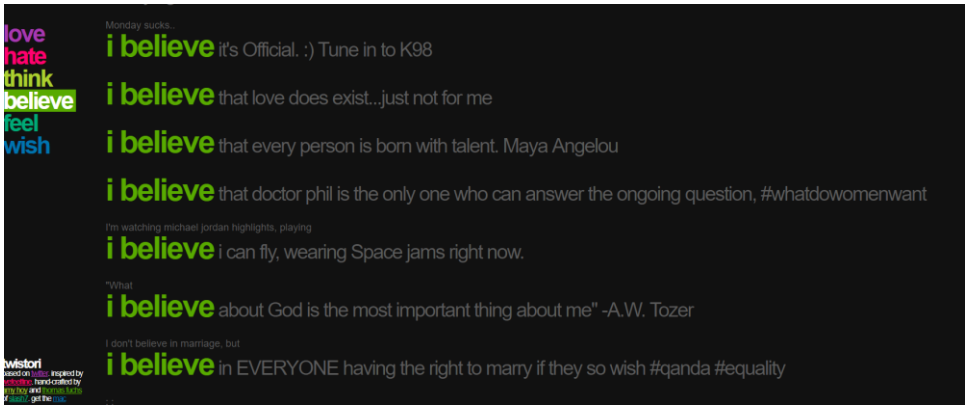
Ιστοσελίδα: twistori.com

Άλλη μια εφαρμογή που χρησιμοποιεί το Twitter API, αυτή τη φορά για να παρουσιάσει σε πραγματικό χρόνο tweets που περιέχουν συγκεκριμένες συναισθηματικές λέξεις:

Love, hate, think, believe, feel, wish



Εικόνα 8 Προβολή tweets που περιέχουν τη λέξη feel.

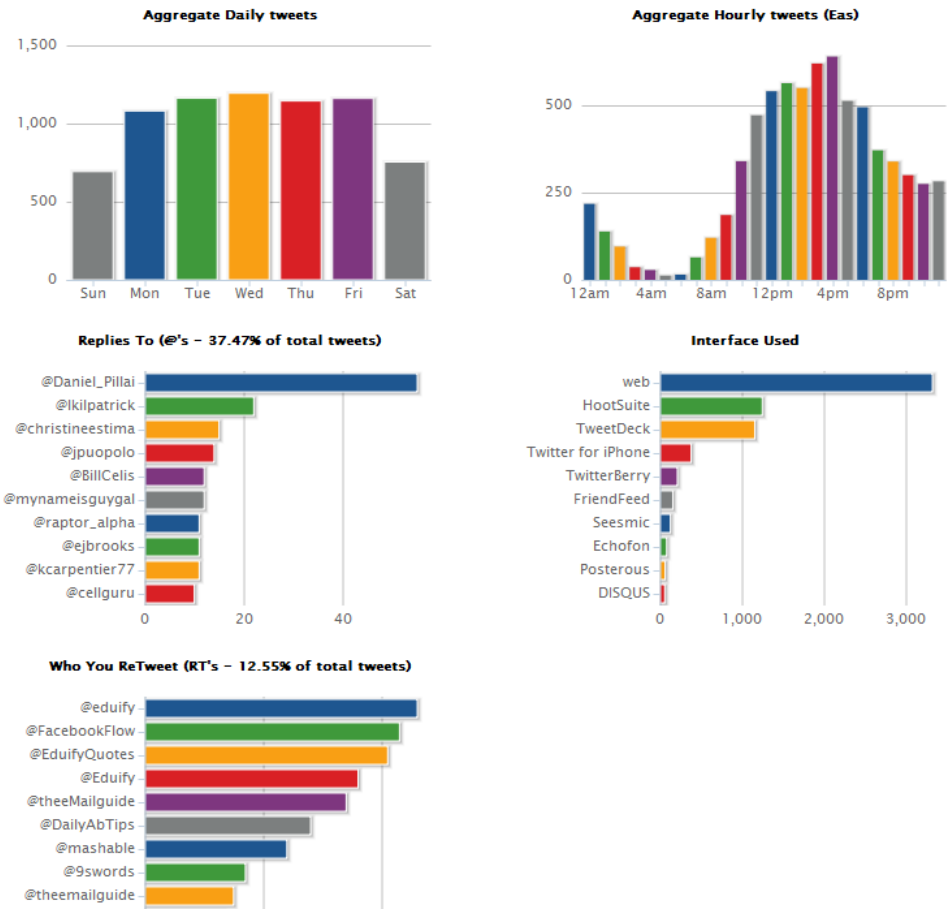


Εικόνα 9 Προβολή tweets που περιέχουν τη λέξη believe.

5. Tweetstats

Ιστοσελίδα: www.tweetstats.com

Το Tweetstats, παρουσιάζει στατιστικά στοιχεία για το λογαριασμό ενός χρήστη, με βάση τα tweets ανά ώρα, ανά μήνα κτλ, στατιστικά που έχουν να κάνουν με το timeline ενός χρήστη και στατιστικά retweet. Η εφαρμογή παρέχεται δωρεάν και έχει τη δυνατότητα να κάνεις donate αν τη βρίσκεις χρήσιμη.



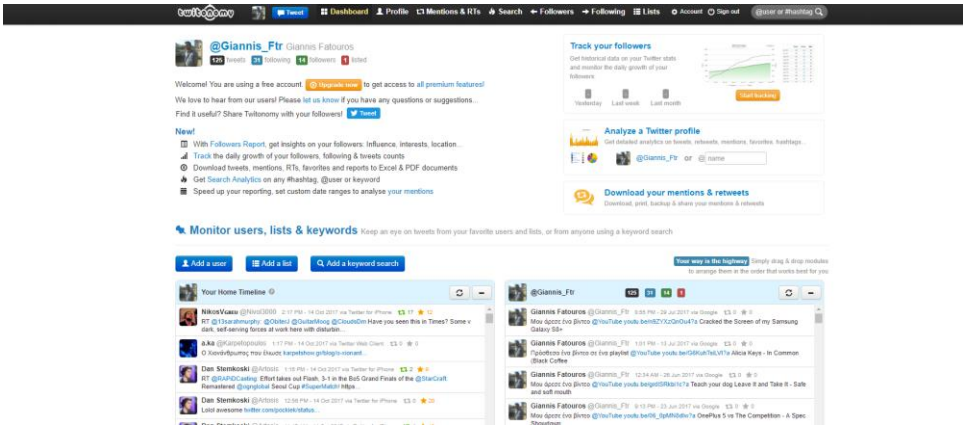
Εικόνα 10 Διαγράμματα μέσα απο την εφαρμογή Tweetstats.

6. Twitonomy

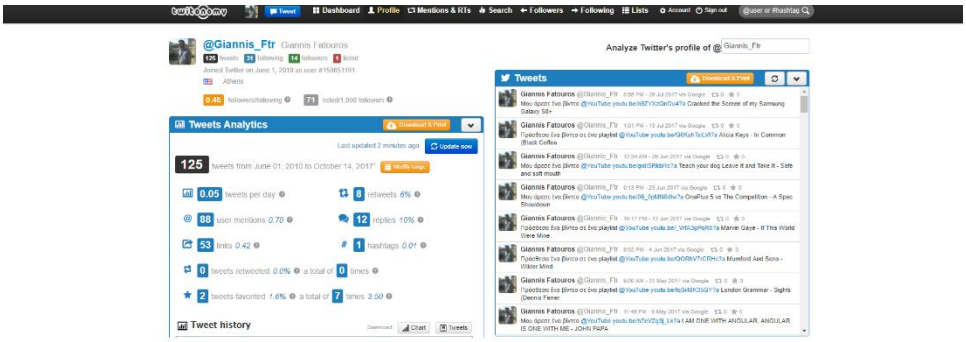
Ιστοσελίδα: www.twitonomy.com

Η εφαρμογή αυτή παρέχεται δωρεάν, αλλά έχει και premium έκδοση με περισσότερες δυνατότητες.

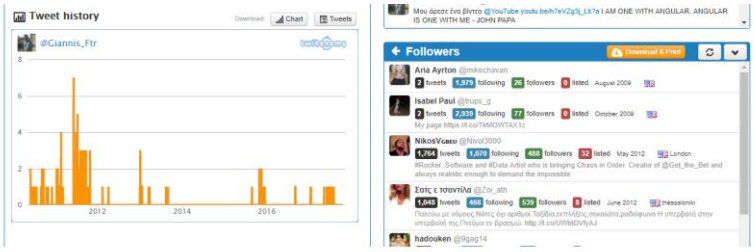
- Επιτρέπει στο χρήστη να δει το Home Timeline του, καθώς και τα δημοσιευμένα του tweets.
- Ανάλυση ενός Twitter προφίλ, λογαριασμού.
- Καταγραφή σε ημερήσια βάση του αριθμού των ακολούθων, φίλων και αριθμό tweets. (premium έκδοση)
- Search Analytics με βάση κάποιο hashtag, @user ή κάποιο συγκεκριμένο keyword.



Εικόνα 11 Αρχική οθόνη από την εφαρμογή Twitonomy .



Εικόνα 12 Χρήσιμα analytics του λογαριασμού ενός χρήστη.

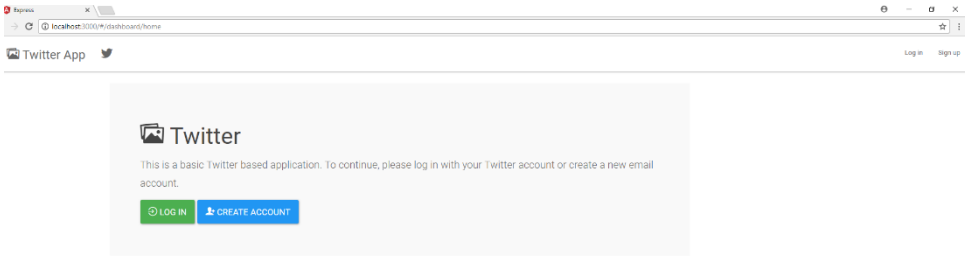


Εικόνα 13 Διαγράμαμτα που δείχνουν την κατανομή των δημοσιεύσεων ενός χρήστη στο χρόνο.

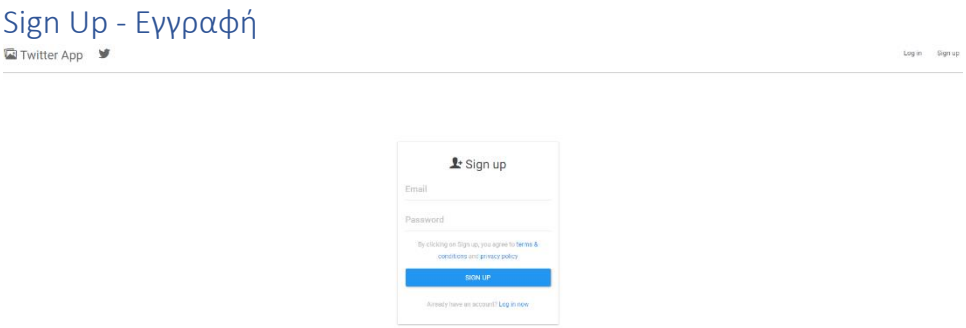
Πρόσφατα το Twitter εξέλιξε τη δική του πλατφόρμα analytics για όλους τους χρήστες, με τη δυνατότητα να μπορούν όλοι να παίρνουν δεδομένα σχετικά με τα tweets τους απευθείας από το Twitter.

6. ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ

Η πρώτη οθόνη που εμφανίζεται όταν πληκτρολογούμε τη διεύθυνση της εφαρμογής, μας δίνει τη δυνατότητα να κάνουμε σύνδεση ή να μεταβούμε στην οθόνη της εγγραφής. Χρειάζεται ένας έγκυρος λογαριασμός είτε μέσω της εφαρμογής του Twitter, είτε χρησιμοποιώντας και δημιουργώντας ένα λογαριασμό. Ένας χρήστης μπορεί να χρησιμοποιήσει την εφαρμογή αν και μόνο έχει λογαριασμό. Υπάρχουν δύο κατηγορίες χρηστών - ρόλοι, ένας απλός χρήστης "user" και ένας "admin" χρήστης. Δεν έχουν διαφορετικό user interface ανάλογα με το ρόλο τους, ορισμένα components - μέρη της εφαρμογής εμφανίζονται και εξαφανίζονται ανάλογα με το ρόλο του χρήστη.



Εικόνα 14 Αρχική οθόνη της εφαρμογής.

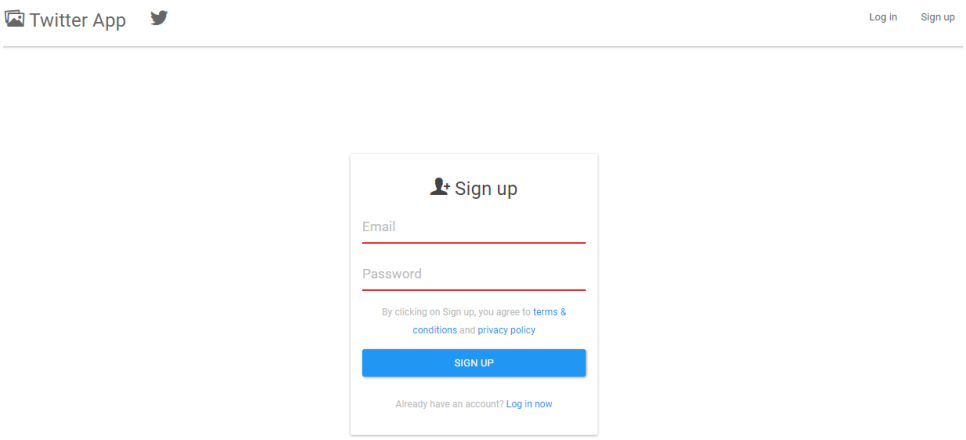


Εικόνα 15 Οθόνη εγγραφής νέου χρήστη.

Περιγραφή

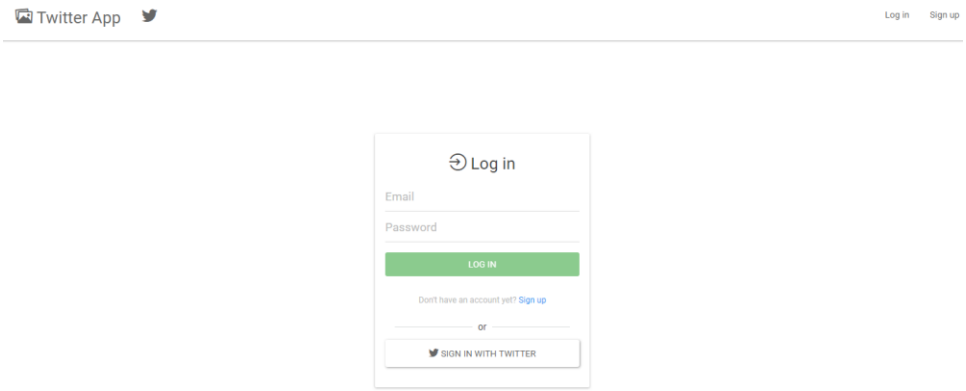
Ο χρήστης μπορεί να δημιουργήσει ένα καινούριο λογαριασμό στην εφαρμογή. Πρέπει να συμπληρώσει τη διεύθυνση ηλεκτρονικού του ταχυδρομείου (email) καθώς και ένα κωδικό πρόσβασης (password). Αν δεν συμπληρωθεί σωστά κάποιο από τα inputs της φόρμας τότε εμφανίζεται περιγραφικό μήνυμα που αναδεικνύει το λάθος του χρήστη.

Σε περίπτωση που δεν συμπληρωθούν τα απαραίτητα πεδία από το χρήστη τότε τα inputs κοκκινίζουν, αναδεικνύοντας το λάθος.



Εικόνα 16 Ένδειξη μηνυμάτων λάθους όταν δεν έχει συμπληρωθεί σωστά η φόρμα.

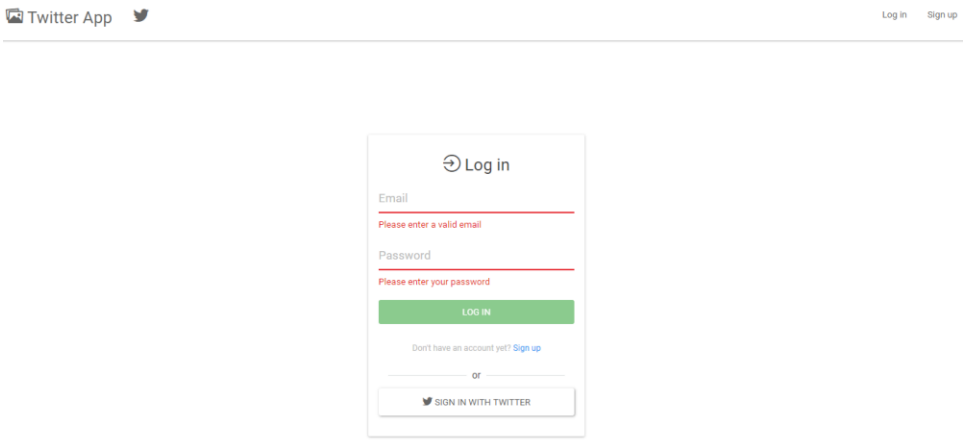
Log In - Είσοδος



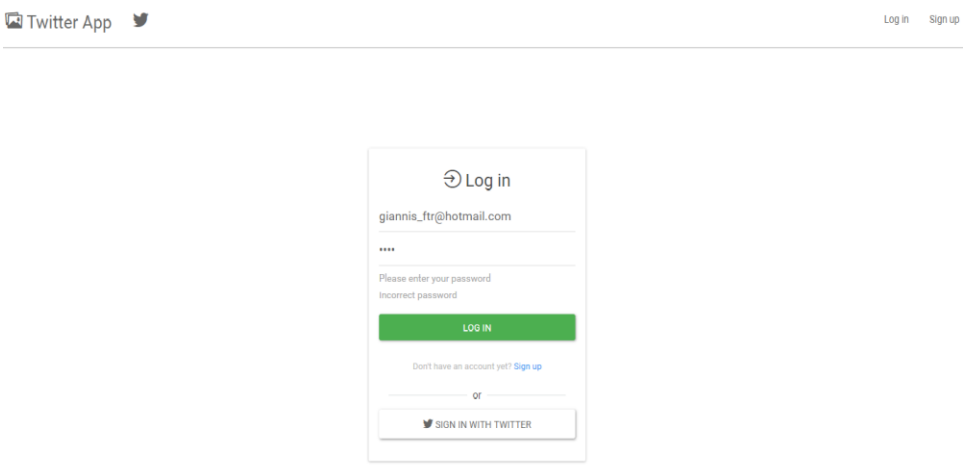
Εικόνα 17 Οθόνη εισόδου εγγεγραμμένου χρήστη.

Περιγραφή

Στην παραπάνω οθόνη, ένας εγγεγραμμένος χρήστης μπορεί να κάνει είσοδο στην εφαρμογή, αρκεί να συμπληρώσει τη διεύθυνση ηλεκτρονικού ταχυδρομείου και τον κωδικό πρόσβασης, ώστε να μπορέσει να αυθεντικοποιηθεί από το σύστημα. Ένας χρήστης μπορεί επίσης να συνδεθεί, πραγματοποιώντας είσοδο με το λογαριασμό Twitter που διαθέτει. Αν η φόρμα δεν έχει συμπληρωθεί σωστά εμφανίζονται περιγραφικά μηνύματα λάθους, καθώς και μηνύματα που μας επιστρέφει ο server όταν πραγματοποιήσουμε ένα αποτυχημένο log in.

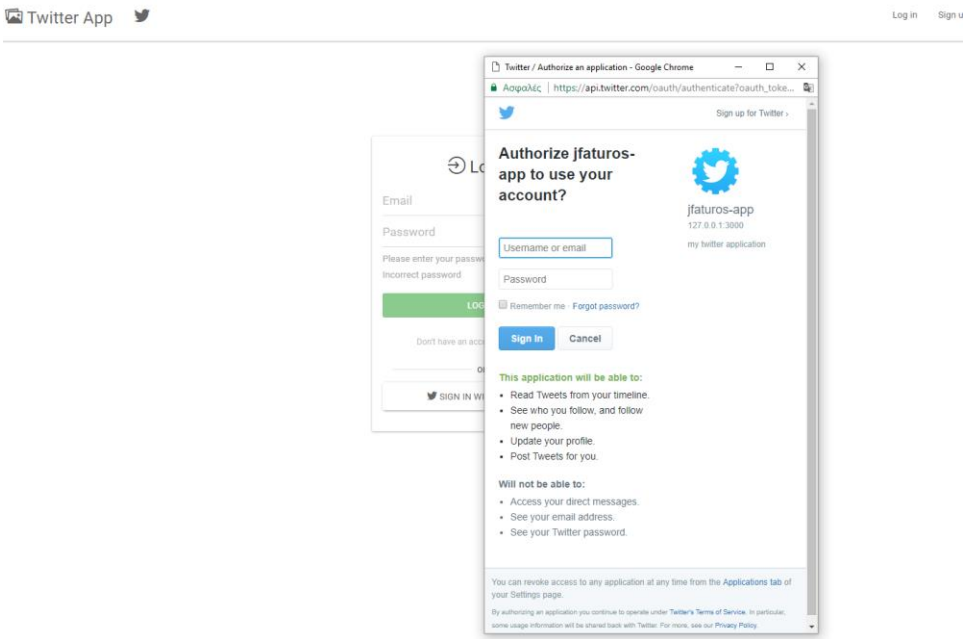


Εικόνα 18 Μηνύματα λάθους στη φόρμα εισόδου.



Εικόνα 19 Μηνύματα λάθους από τον server.

Είσοδος με τη χρήση του λογαριασμού Twitter.



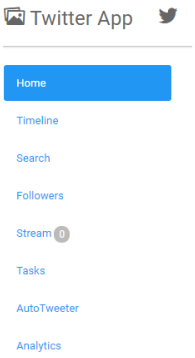
Εικόνα 20 Είσοδος χρήστη με χρήση του twitter λογαριασμού του.

Στην παραπάνω εικόνα ο χρήστης καλείται να συμπληρώσει τα στοιχεία του twitter λογαριασμού του, δίνοντας έγκριση στην εφαρμογή που έχει δημιουργηθεί να αποκτήσει πρόσβαση σε δεδομένα του λογαριασμού του. Έπειτα το twitter ανακατευθύνει το χρήστη πίσω στην εφαρμογή.

Σε όλες τις επιτυχημένες εισόδου ο χρήστης θα μεταφερθεί αυτόματα στην αρχική οθόνη της εφαρμογής, που είναι η οθόνη Home.

Οθόνη Home

Η οθόνη αυτή αποτελεί την πρώτη οθόνη που θα δει ένας χρήστης όταν πραγματοποιεί επιτυχή είσοδο στην εφαρμογή. Αριστερά, βρίσκεται μια navigation bar, που μπορεί να δώσει τη δυνατότητα στο χρήστη να μεταβεί σε άλλες λειτουργίες της εφαρμογής.

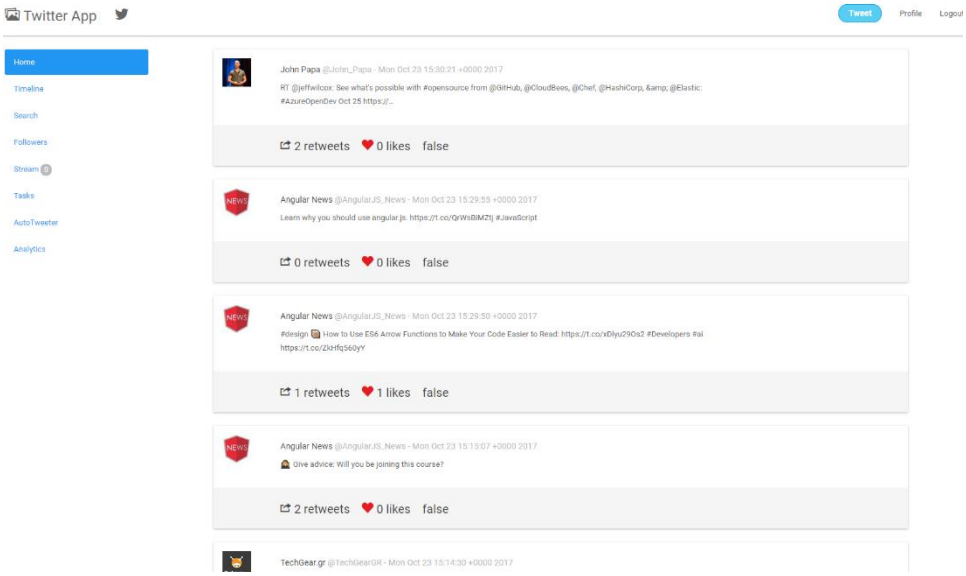


Εικόνα 21 Navigation Bar. Μενού πλοήγησης.

Στο πάνω μέρος και δεξιά, εμφανίζονται τα κουμπιά Tweet και links που οδηγούν το χρήστη στο προφίλ του λογαριασμού του Twitter του και Logout (link) για την αποσύνδεση του χρήστη και έξοδο από την εφαρμογή.

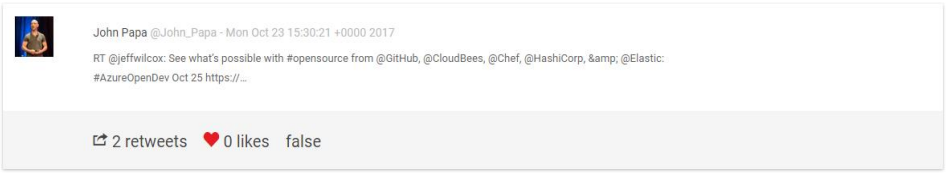
Εικόνα 22 Navigation Bar Top.

Στο κύριο μέρος της οθόνης Home εμφανίζονται όλα τα tweets που εμφανίζονται στην timeline του χρήστη που έχουμε συνδέσει την εφαρμογή και διαχειριζόμαστε το λογαριασμό του.



Εικόνα 23 Οθόνη Home.

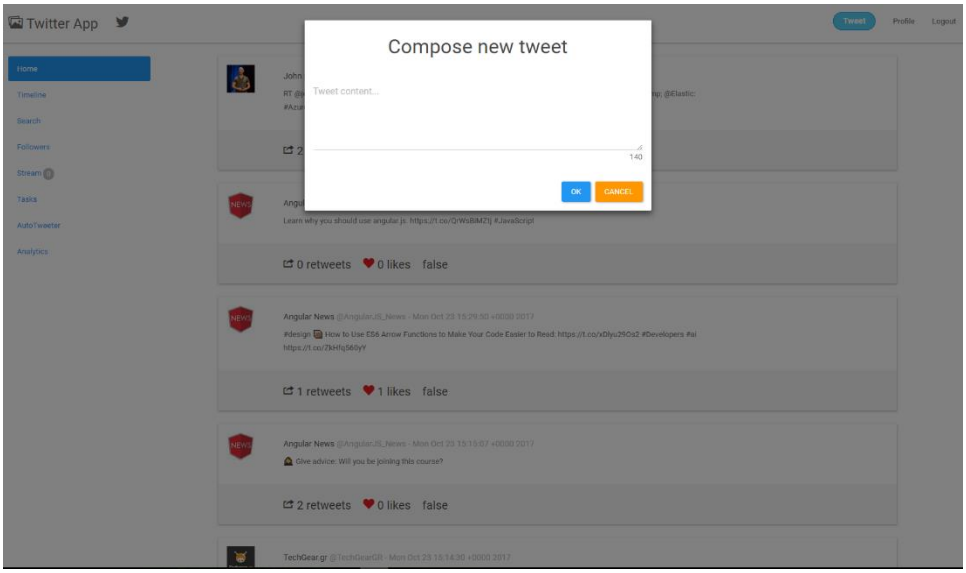
Στην οθόνη αυτή μπορούμε να δούμε πληροφορία που αφορά τα retweets, likes καθώς και το περιεχόμενο του Tweet, μια εικόνα του χρήστη που δημοσίευσε το tweet και το screen_name και ημερομηνία δημοσίευσης. Η πληροφορία αυτή εμφανίζεται στο χρήστη σαν κάρτα.



Εικόνα 24 Προβολή ενός tweet.

Tweet

Χρησιμοποιώντας το σύνδεσμο Tweet, που εμφανίζεται στην οθόνη Home, πάνω δεξιά, ένας χρήστης μπορεί να δημοσιεύσει ένα νέο Tweet. Στον χρήστη εμφανίζεται ένα modal που εμφανίζει μια φόρμα προς συμπλήρωση, ουσιαστικά εμφανίζεται ένα textarea HTML tag, οπου ο χρήστης μπορεί να χρησιμοποιήσει για να γράψει το περιεχόμενο του tweet που επιθυμεί να δημοσιεύσει. Το textarea περιλαμβάνει το μέγιστο αποδεκτό όριο χαρακτήρων που έχει ορίσει το Twitter, και είναι οι 140 χαρακτήρες. Καθώς ο χρήστης πληκτρολογεί, ενημερώνεται για το υπόλοιπο αριθμό των χαρακτήρων που μπορεί να χρησιμοποιήσει.

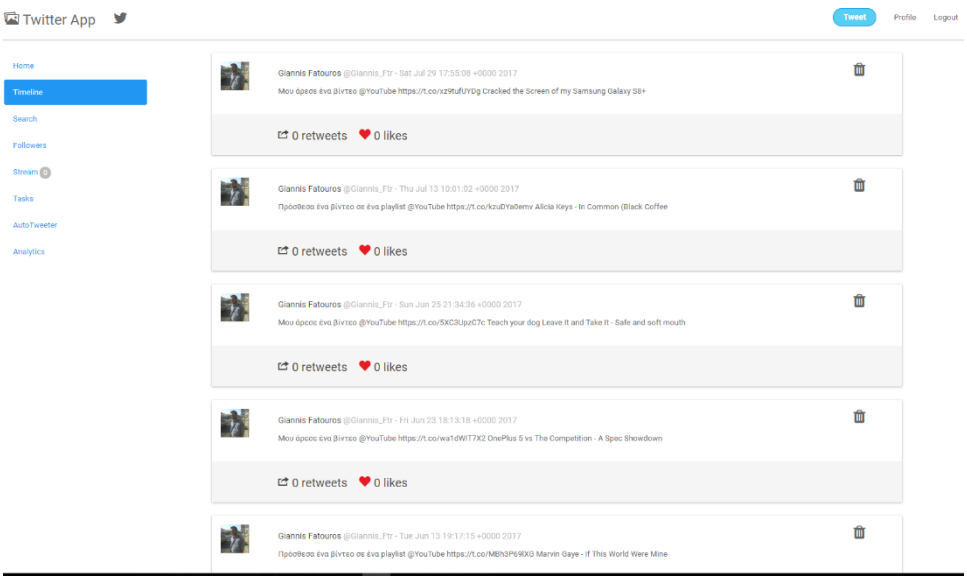


Εικόνα 25 Δημιουργία νέου tweet.

Πατώντας το κουμπί Cancel, ακυρώνεται ότι έχει πληκτρολογήσει ο χρήστης και κλείνει το αναδυόμενο αυτό παράθυρο, ενώ όταν ο χρήστης πατήσει το κουμπί OK, τότε θα γίνει η δημοσίευση του Tweet και θα κλείσει το modal.

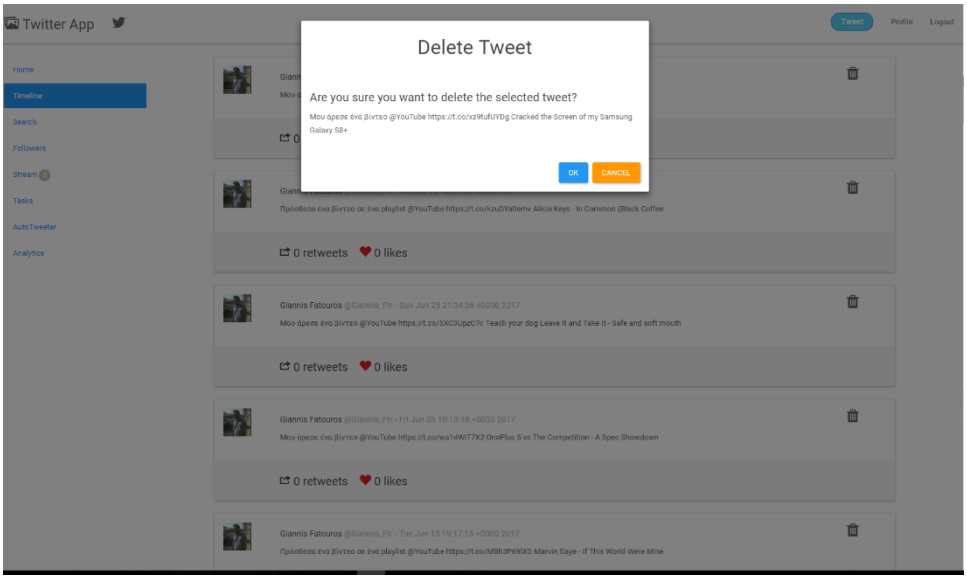
Timeline

Στην οθόνη αυτή εμφανίζονται τα τελευταία δημοσιευμένα tweets του χρήστη. Και σε αυτή την οθόνη η παρουσίαση του κάθε tweet, παρουσιάζεται ως κάρτα, περιλαμβάνει πληροφορία όπως, η φωτογραφία που έχει επιλέξει ο χρήστης να εμφανίζεται στο προφίλ, το όνομα χρήστη και το screen_name που έχει στο Twitter, η ημερομηνία και ώρα δημοσίευσης, καθώς και πόσα retweets και πόσα likes έχει το κάθε tweet post ξεχωριστά. Ταυτόχρονα υπάρχει η δυνατότητα να διαγραφεί ένα Tweet.



Εικόνα 26 Οθόνη Timeline. Περιέχει τα δημοσιευμένα tweet.

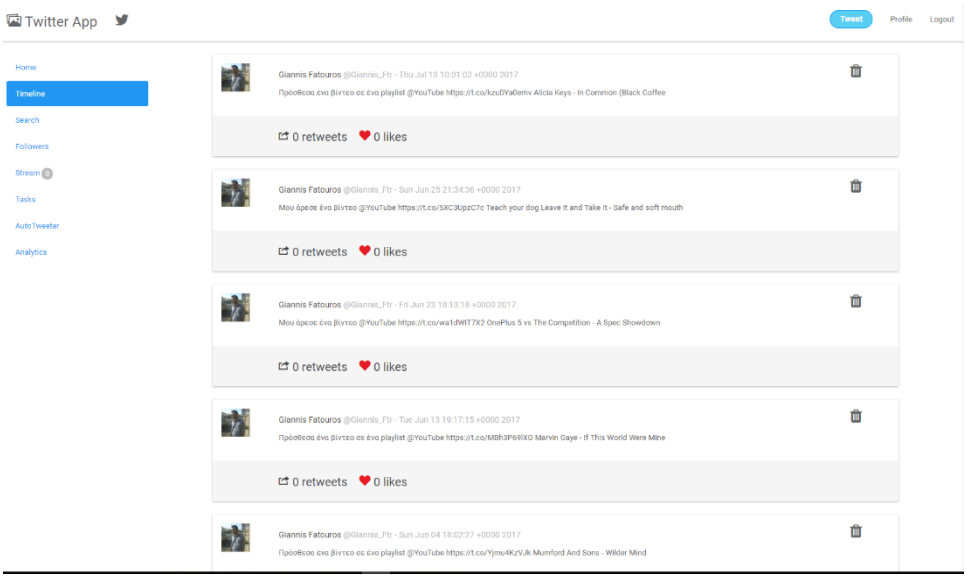
Ο χρήστης μπορεί να διαγράψει ένα tweet, πατώντας το κάδο ανακύκλωσης που εμφανίζεται στο δεξιό μέρος της κάρτας. Τότε, θα εμφανιστεί ένα προειδοποιητικό μήνυμα που θα πληροφορήσει το χρήστη για την ενέργεια που προτίθεται να κάνει.



Εικόνα 27 Προειδοποίηση διαγραφής ενός tweet.

Εδώ ο χρήστης έχει δύο επιλογές:

- Cancel, κλείνει το αναδυόμενο παράθυρο και επιστρέφει στην οθόνη που βρισκόταν χωρίς να γίνει καμία διαγραφή.
- OK, τότε στέλνεται ένα request για διαγραφή του επιλεγμένου tweet, και ταυτόχρονα ενημερώνεται και η οθόνη Timeline και διαγράφεται το επιλεγμένο tweet.

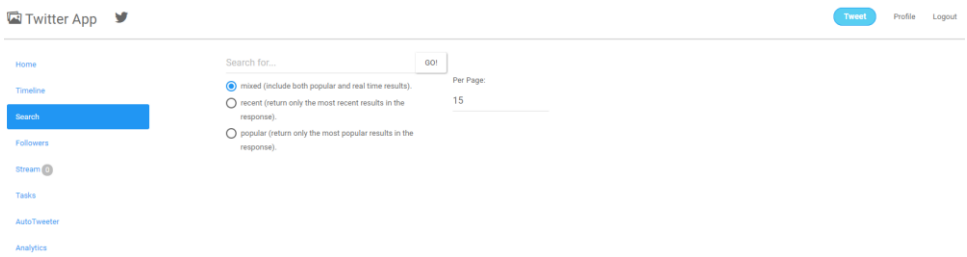


Εικόνα 28 Ανανεωμένη λίστα των tweets μετά τη διαγραφή.

Η παραπάνω οθόνη αποτελεί την οθόνη που θα δει ο χρήστης μετά από μια επιτυχημένη διαγραφή, η οθόνη έχει ενημερωθεί και δεν περιέχει το Post που διέγραψε ο χρήστης, χωρίς να χρειάζεται να γίνει refresh στη σελίδα.

Search

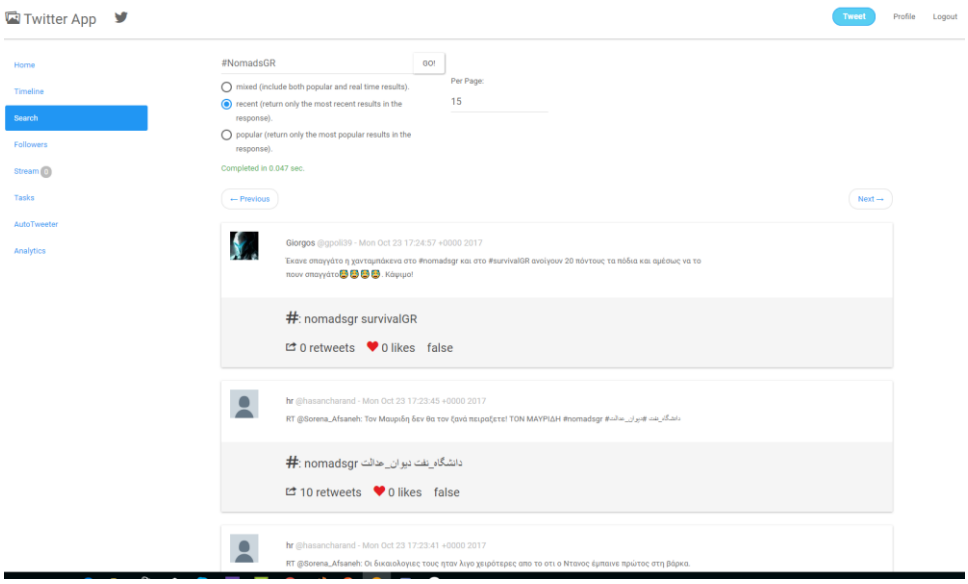
Η οθόνη search επιτρέπει στο χρήστη να χρησιμοποιήσει το Search API του Twitter, για να αναζητήσει tweets με βάση κάποιο keyword ή κάποιο query όπως ορίζει το documentation του Twitter, Search Tweets(docs). Για την αναζήτηση, μπορεί να επιλέξει να ψάξει με βάση τα ποιο πρόσφατα είτε τα ποιο δημοφιλή είτε και τα δύο. Επίσης, μπορεί να επιλέξει τον αριθμό των αποτελεσμάτων που θέλει να δει ανά σελίδα.



Εικόνα 29 Οθόνη Search - Αναζήτησης.

Στο πεδίο Search for... , μπορεί να εισάγει οποιοδήποτε κείμενο θέλει να ψάξει για περιεχόμενο, είτε να ψάξει για κάποιο #hashtag, είτε για @user_mention, η εφαρμογή θα στείλει το query στο Search API του Twitter χρησιμοποιώντας την συνάρτηση javascript encodeURIComponent, η οποία αναλαμβάνει να κάνει encode όλους τους ειδικούς χαρακτήρες που έχουν χρησιμοποιηθεί.

Πατώντας το κουμπί GO!, στέλνεται το request στο server μας, ο οποίος έπειτα το προωθεί στο Twitter Search API, για να πάρει απάντηση με τα αποτελέσματα.



Εικόνα 30 Χρήση της αναζήτησης και προβολή αποτελεσμάτων.

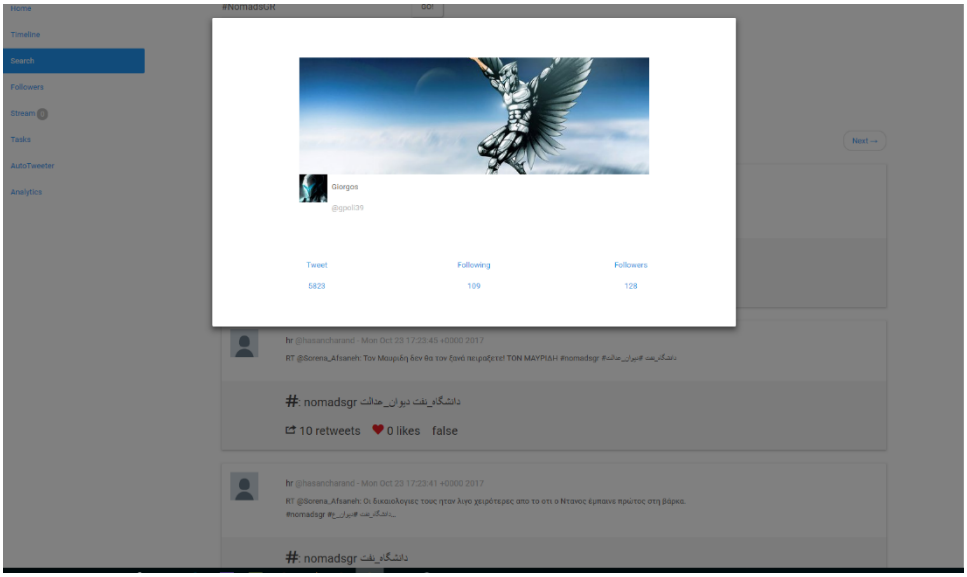
Και σε αυτή την οθόνη, όλα τα αποτελέσματα - tweets, εμφανίζονται με τη μορφή κάρτας, περιέχοντας πληροφορία για το όνομα του χρήστη, το screen_name του χρήστη, ημερομηνία και ώρα δημοσίευσης, το κείμενο του tweet, τα hashtags που έχουν χρησιμοποιηθεί καθώς και τον αριθμό των retweets, των likes, και τη φωτογραφία που έχει ο χρήστης να εμφανίζεται.

Στο σημείο αυτό, ο χρήστης μπορεί να κάνει τις εξής δύο ενέργειες:

- Next, πατώντας το κουμπί next, εμφανίζονται τα επόμενα (#per page) tweets.

Εικόνα 31 Κουμπί Next, για περισσότερα αποτελέσματα.

- Click στην εικόνα ενός χρήστη, όπου θα ξεκινήσει ένα νέο request που θα μας φέρει και θα εμφανίσει σε ένα αναδυόμενο παράθυρο, πληροφορίες που αφορούν το χρήστη, που έχει δημοσιεύσει το συγκεκριμένο tweet. Τον αριθμό των tweets που έχει δημοσιεύσει, τον αριθμό των followers και τον αριθμό των χρηστών που ακολουθεί (Following), καθώς και το όνομα χρήστη, screen_name, φωτογραφία και background φωτογραφία που έχει στο λογαριασμό του Twitter του. Πατώντας σε οποιοδήποτε σημείο εκτός του παραθύρου αυτού κάνει το modal να κλείσει, και ο χρήστης επιστρέφει στην προηγούμενη οθόνη, αυτή του search.



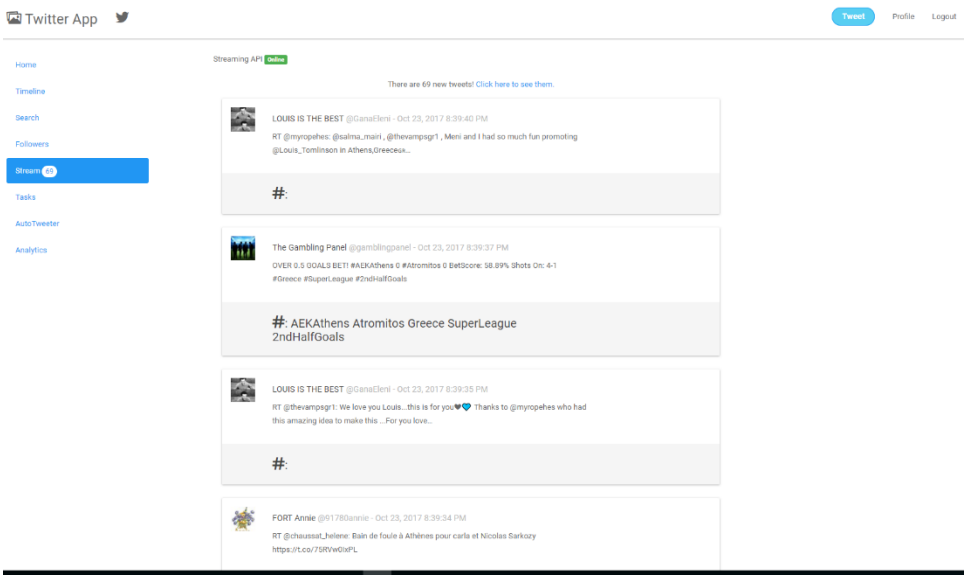
Εικόνα 32 Προβολή ενός χρήστη.

Stream

Η οθόνη αυτή παρουσιάζει στους χρήστες real time tweets που καταγράφονται στο σύστημα και που σχετίζονται με τις λέξεις (hashtags) που έχει ορίσει ο χρήστης με ρόλο "admin". Η οθόνη αυτή διαφοροποιείται ανάλογα με το ρόλο του χρήστη.

User

Ένας απλός χρήστης μπορεί να δει στην εφαρμογή την παρακάτω οθόνη,



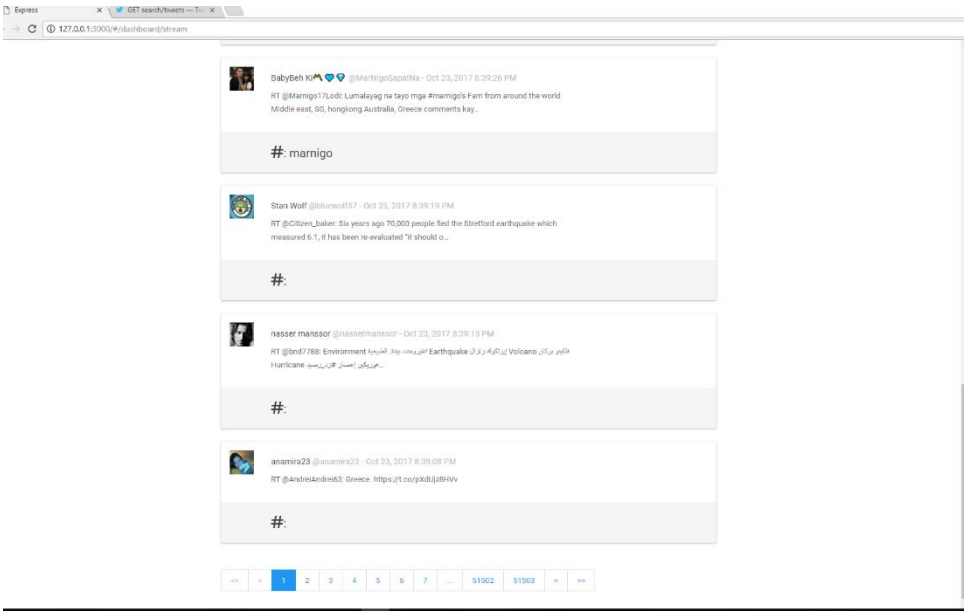
Εικόνα 33 Οθόνη stream, όπως φαίνεται από τον χρήστη user.

Που περιλαμβάνει real time περιεχόμενο των tweets, με βάση τις λέξεις που έχουν οριστεί από το "admin".

Ένδειξη με την πληροφορία της κατάστασης του μηχανισμού που καταναλώνει το Streaming API του twitter.

Ένδειξη με τον αριθμό των νέων tweets, που έχουν καταχωρηθεί στη βάση δεδομένων.

Τα αποτελέσματα που επιστρέφονται περιορίζονται στον αριθμό 10, και χρειάζεται ο χρήστης να χρησιμοποιήσει το μηχανισμό paging που έχει κατασκευαστεί για να δει επόμενα ή προηγούμενα tweets.

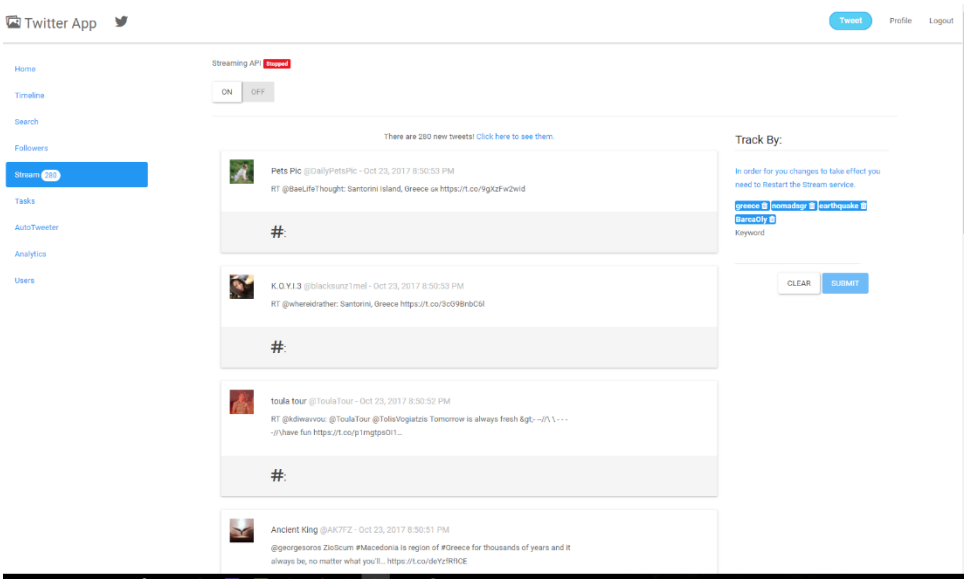


Εικόνα 34 Μηχανισμός paging για προβολή αποτελεσμάτων.

Admin

Όπως περιγράφτηκε παραπάνω, ο χρήστης με ρόλο "admin", έχει μερικές επιπλέον λειτουργίες στην συγκεκριμένη οθόνη.

Μπορεί να κλείσει / ανοίξει, την καταγραφή των tweets στο σύστημα, με τη χρησιμοποίηση του ON / OFF διακόπτη που έχει κατασκευαστεί.



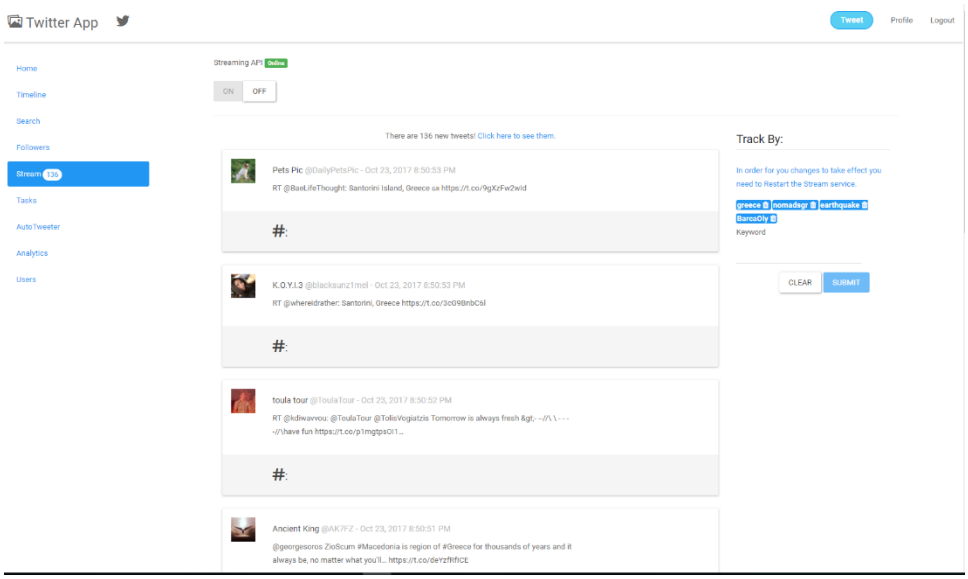
Εικόνα 35 Οθόνη stream, όπως φαίνεται από έναν χρήστη με ρόλο admin. Η ένδειξη του streaming είναι off.

Καθώς και να προσθέσει/αφαιρέσει κάποιο keyword για καταγραφή.

Για την προσθήκη ενός νέου keyword αρκει να πληκτρολογήσει το keyword ου επιθυμεί και να πατήσει το κουμπί SUBMIT.

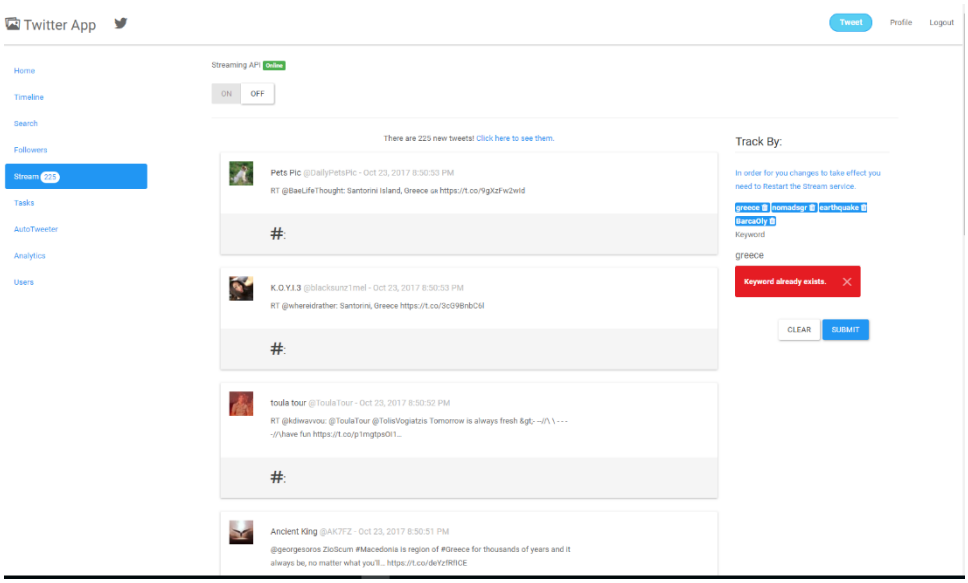
Αν θέλει να ακυρώσει και να διαγράψει το περιεχόμενο που έχει εισάγει μπορεί να πατήσει το πλήκτρο CLEAR.

Αν θέλει να διαγράψει κάποιο keyword, τότε μπορεί να το κάνει πατώντας τον κάδο ανακύκλωσης που εμφανίζεται δίπλα σε κάθε keyword.



Εικόνα 36 Οθόνη stream, όπως φαίνεται από έναν χρήστη με ρόλο admin. Η ένδειξη του streaming είναι on.

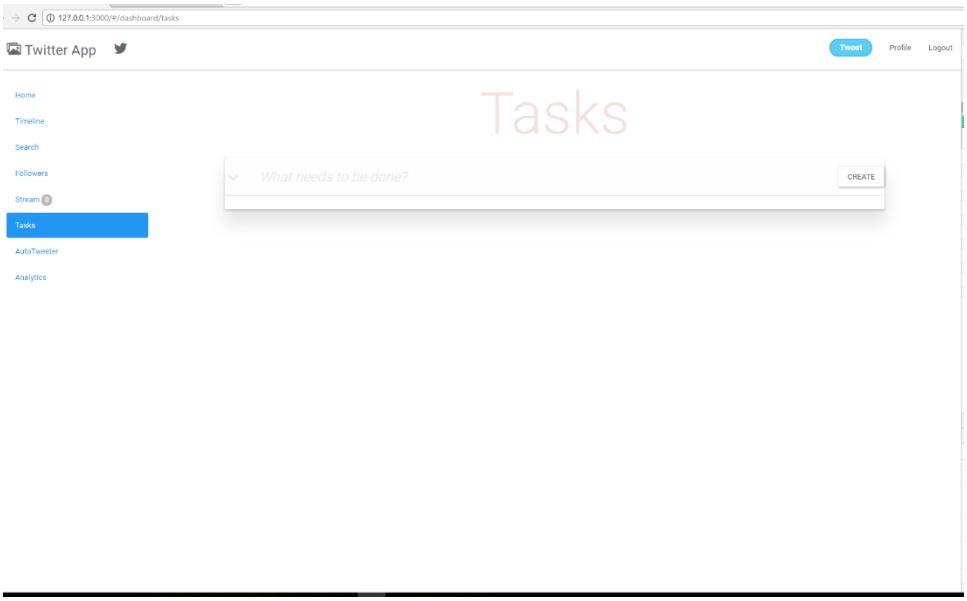
Σε περίπτωση που ο χρήστης προσπαθήσει να καταχωρήσει το ίδιο keyword, τότε το σύστημα θα τον πληροφορήσει με ένα ειδικό μήνυμα λάθους.



Εικόνα 37 Μήνυμα λάθους κατά την εισαγωγή νέας λέξης αναζήτησης.

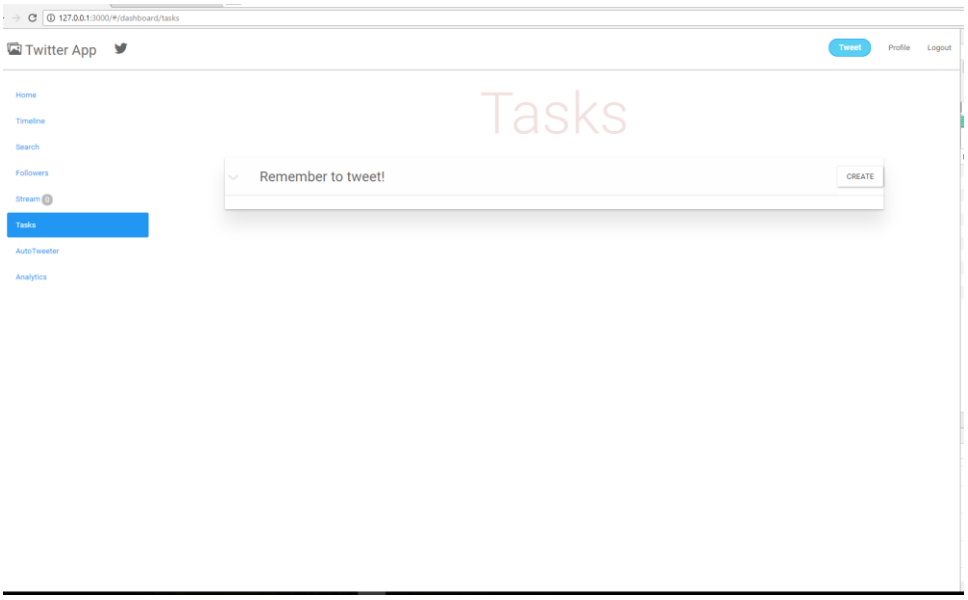
Tasks

Στην οθόνη αυτή ένας χρήστης μπορεί να δει όλες τις εργασίες - tasks που έχει καταχωρήσει.

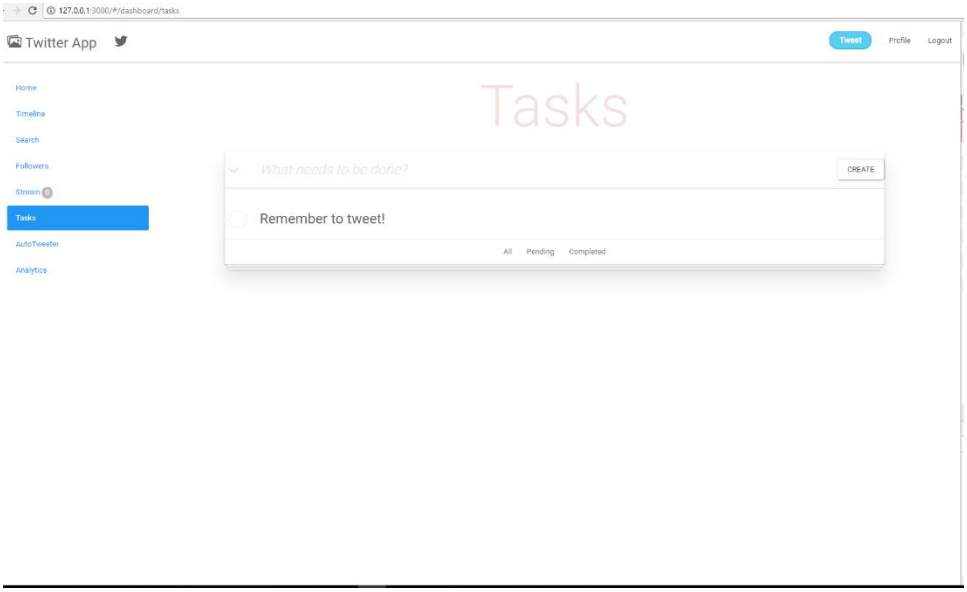


Εικόνα 38 Οθόνη εργασιών - Tasks.

Για δημιουργία μιας νέας εργασίας, ο χρήστης καλείται να συμπληρώσει το πεδίο, What needs to be done? Και έπειτα να πατήσει το κουμπί CREATE. Τότε αν η εργασία καταχωρηθεί επιτυχώς θα ενημερωθεί η λίστα με τις εργασίες.

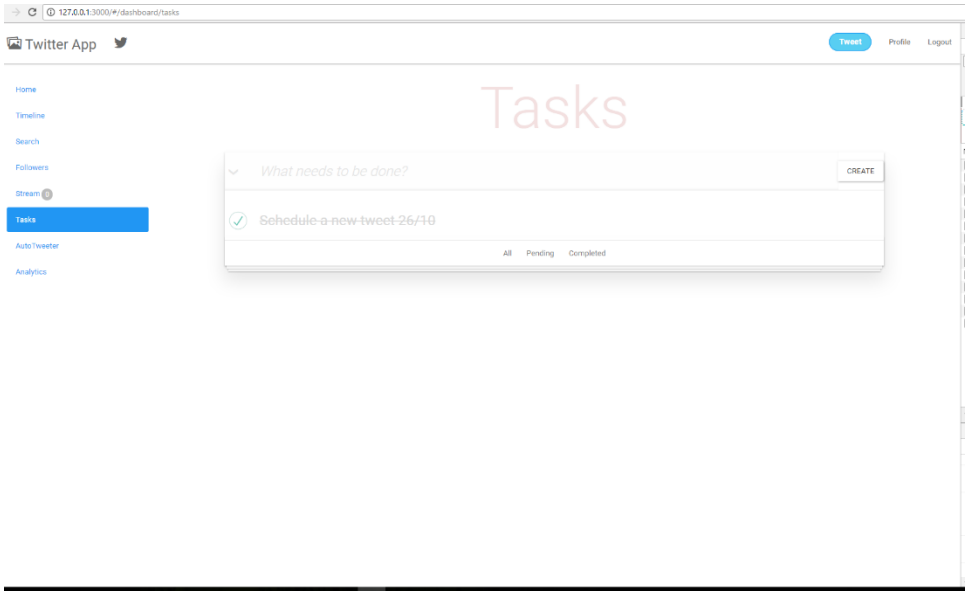


Εικόνα 39 Δημιουργία μιας νέας εργασίας.



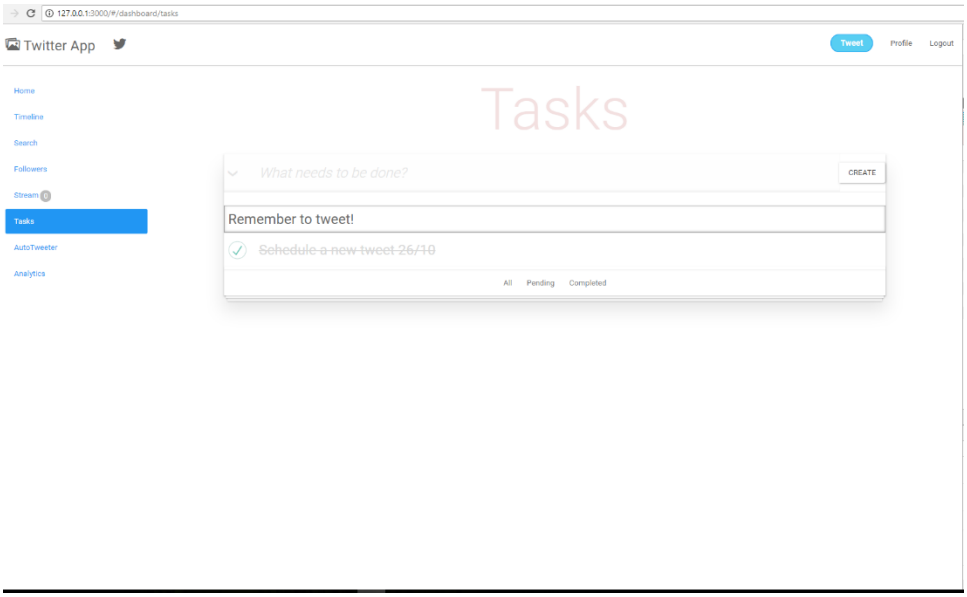
Εικόνα 40 Καταχώρηση μιας νέας εργασίας.

Μπορεί να φιλτράρει τις εργασίες με βάση την κατάστασή τους, All, Pending, Completed. All, αφορά όλες τις εργασίες. Pending, φιλτράρει τις εργασίες με βάση ποιες είναι μη ολοκληρωμένες και τέλος Completed, φιλτράρει και παρουσιάζει μόνο όσες εργασίες είναι ολοκληρωμένες.



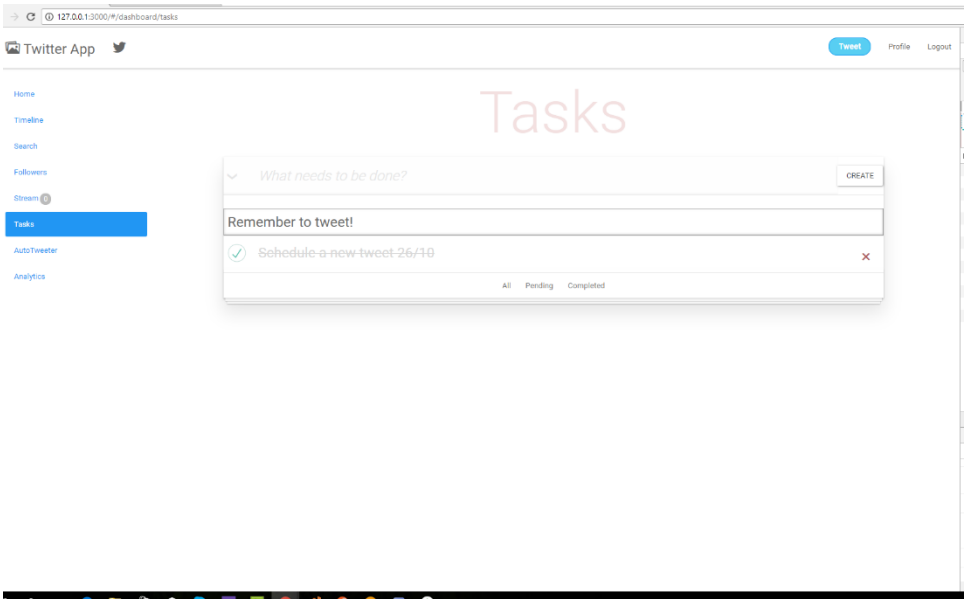
Εικόνα 41 Αλλαγή μιας εργασίας ώς completed.

Ένας χρήστης μπορεί να τροποποιήσει το περιεχόμενο μιας εργασίας, πατώντας διπλό κλικ στο κείμενο, τότε θα εμφανιστεί στη θέση του κειμένου, ένα editable input. Ο χρήστης μπορεί να αλλάξει το κείμενο και είτε να πατήσει enter, έτσι ώστε να σωθούν οι αλλαγές του είτε να πατήσει το escape και να ακυρωθούν οι όποιες αλλαγές. Τροποποίηση όμως μπορεί και να γίνει κάνοντας κλικ στο κύκλο μπροστά από το κείμενο της εργασίας, μαρκάροντάς την ως completed.



Εικόνα 42 Τροποποίηση του ονόματος μια εργασίας.

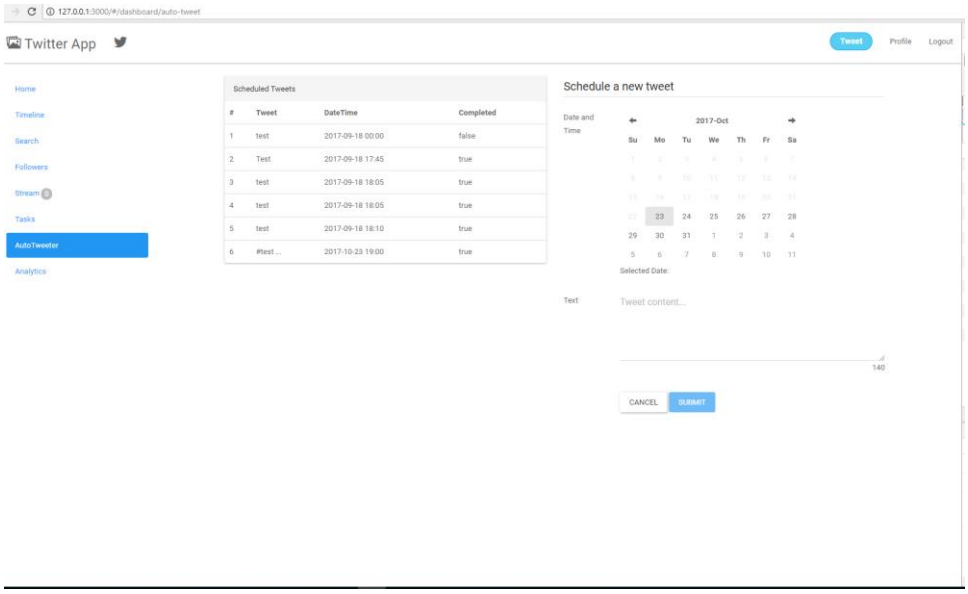
Τέλος, ένας χρήστης μπορεί να διαγράψει όποια εργασία επιθυμεί, κάνοντας κλικ στο σύμβολο X, δεξιά από το όνομά της. Η διεγραμμένη εργασία θα αφαιρεθεί από τη λίστα.



Εικόνα 43 Ο χρήστης μπορεί να διαγράψει μια εργασία πατώντας το Χ.

AutoTweeter

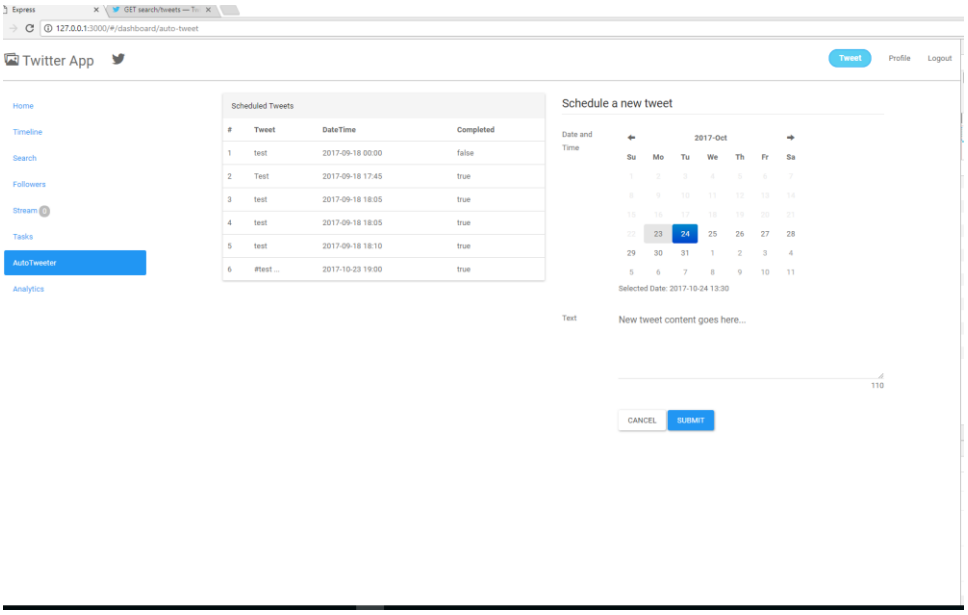
Στην οθόνη αυτή ένας χρήστης μπορεί να καταχωρήσει ένα καινούριο tweet και να το προγραμματίσει για μια μελλοντική ημερομηνία και ώρα.



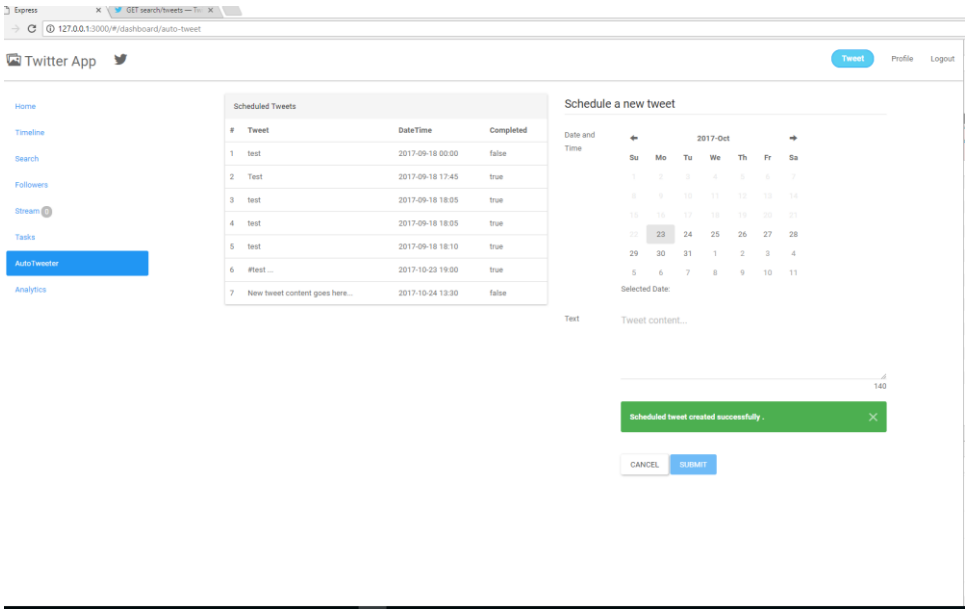
Εικόνα 44 Οθόνη AutoTweeter.

Στον πίνακα προβάλλονται όλα τα προγραμματισμένα tweets (Tweet), με την ημερομηνία και ώρα (DateTime) καθώς και την ένδειξη αν είναι δημοσιευμένο ή όχι (Completed)

Επίσης, όπως είναι φυσικό, υπάρχει η φόρμα με την οποία ο χρήστης μπορεί να προγραμματίσει ένα tweet. Οπότε, περιλαμβάνεται ένα ημερολόγιο (Date and Time) με τις μελλοντικές ημερομηνίες διαθέσιμες. Αφού ο χρήστης επιλέξει μια ημέρα τότε θα εμφανιστεί η επιλογή για την ώρα και τα λεπτά. Ακριβώς από κάτω (Text), ο χρήστης μπορεί να πληκτρολογήσει το κείμενο προς δημοσίευση, έως 140 χαρακτήρες. Αν και μόνο αν ο χρήστης έχει επιλέξει μια ημερομηνία και ώρα καθώς και αν έχει πληκτρολογήσει κάποιο περιεχόμενο στο textarea Text, θα ενεργοποιηθεί το κουμπί SUBMIT, για την καταχώρηση, διαφορετικά το κουμπί παραμένει disabled, όπως παραπάνω. Πατώντας το SUBMIT, και εάν η καταχώρηση γίνει επιτυχώς, ο πίνακας θα ενημερωθεί με την καινούρια καταχώρηση και η φόρμα θα μείνει κενή. Τέλος θα πληροφορηθεί ο χρήστης για την καταχώρησή του!



Εικόνα 45 Προγραμματισμός ενός νέου tweet. Επιλογή ημερομηνίας.

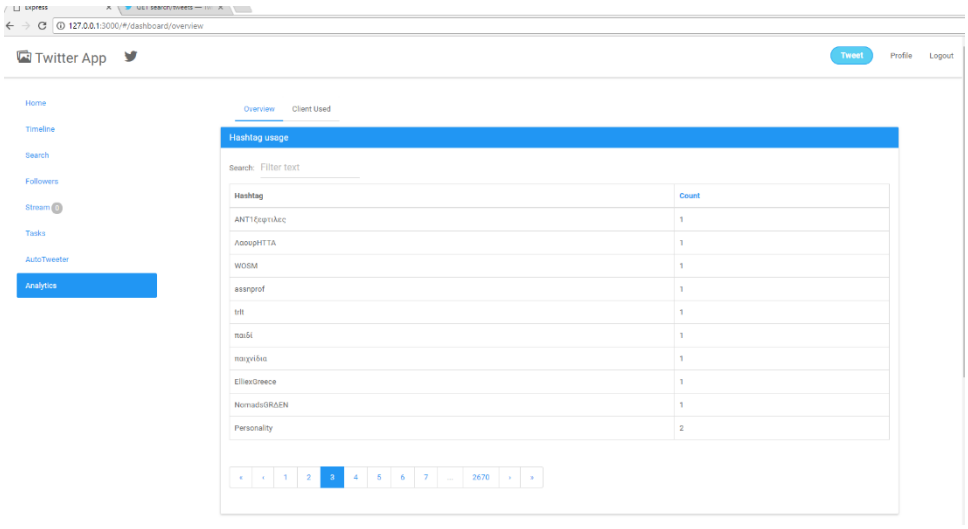


Εικόνα 46 Δημιουργία ενός προγραμματισμένου tweet.

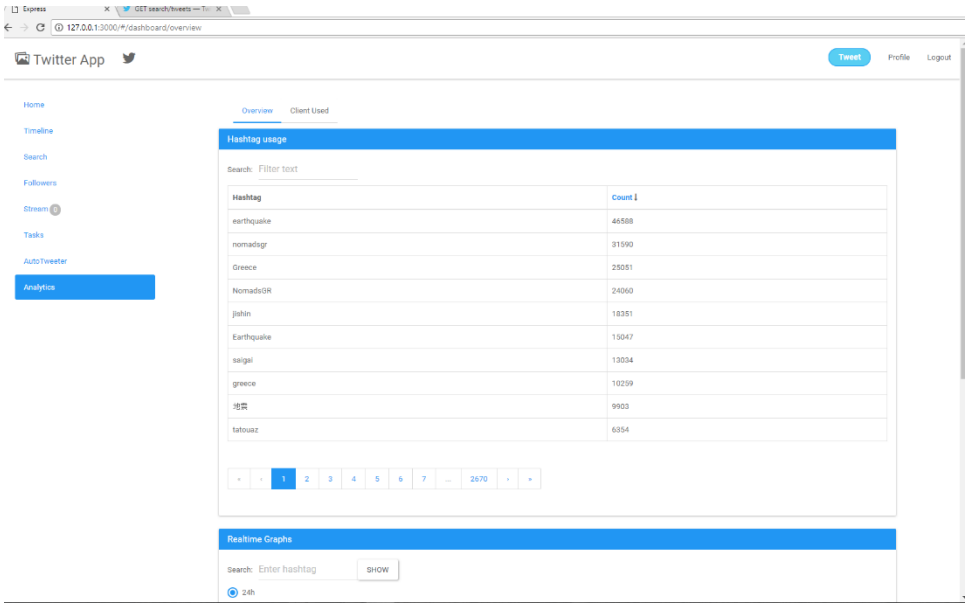
Analytics

Στην οθόνη αυτή παρουσιάζονται διάφορα χρήσιμα διαγράμματα που πληροφορούν το χρήστη για την αποθηκευμένη πληροφορία. Η οθόνη αυτή έχει δυο επιλογές: Overview, Client Used.

Στην πρώτη, Overview, εμφανίζονται πόσα tweets έχουν καταγραφεί με κάποιο συγκεκριμένο hashtag. Ο χρήστης μπορεί να φιλτράρει το περιεχόμενο, να χρησιμοποιήσει το μηχανισμό σελίδων για να δει επόμενα αποτελέσματα και τέλος να σορτάρει τα αποτελέσματα πατώντας στο Count είτε κατά αύξων είτε φθίνων αριθμό.

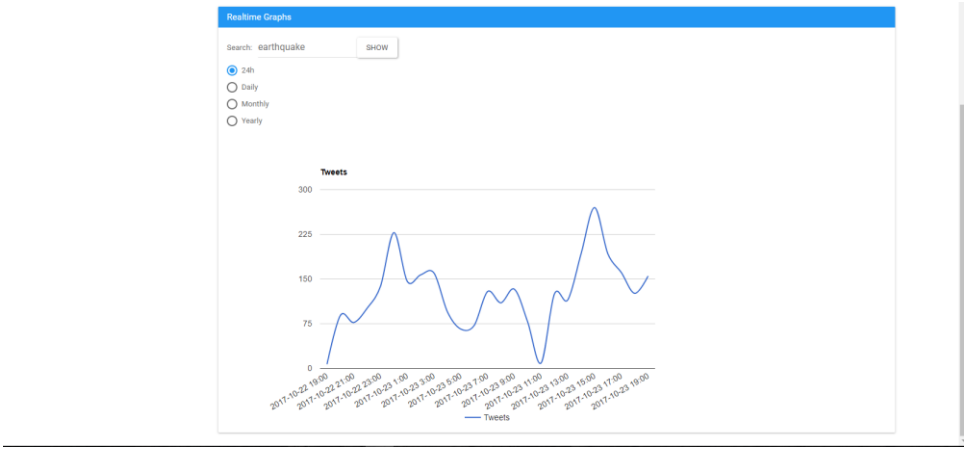


Εικόνα 47 Οθόνη Analytics.

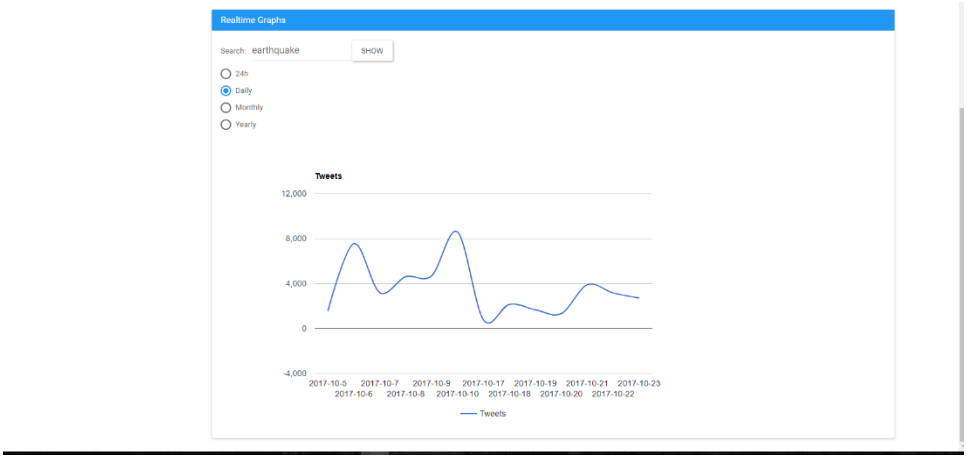


Εικόνα 48 Σορτάρισμα με βάση το count (φθίνων)

Ακριβώς από κάτω στην οθόνη, εμφανίζεται πληροφορία και διαγράμματα που δείχνουν τις ώρες που οι χρήστες επιλέγουν να δημοσιεύσουν τα tweets τους με βάση το τελευταίο 24ωρο. Διάγραμμα που δείχνει την κατανομή των tweets καθημερινά, μηνιαία και ετήσια.

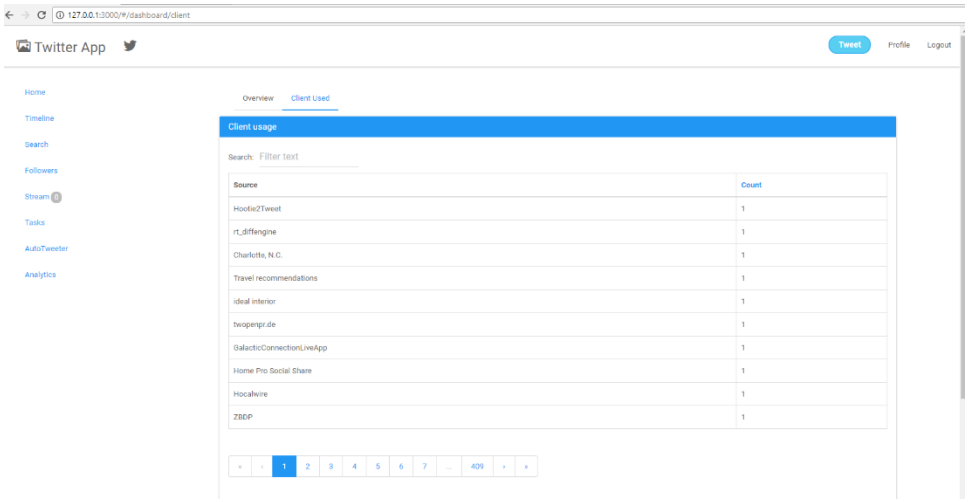


Εικόνα 49 Διαγράμματα κίνησης ενός hashtag μέσα στη μέρα (24 ώρες).

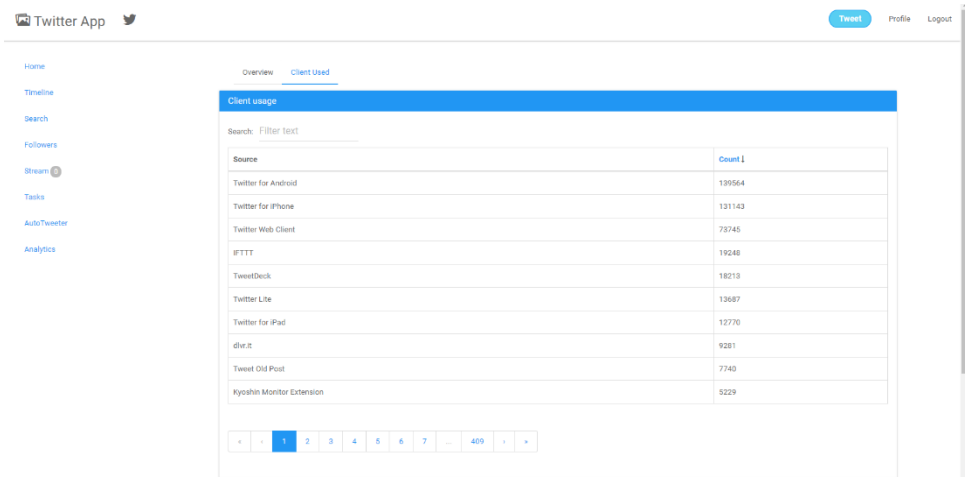


Εικόνα 50 Διαγράμματα κίνησης ενός hashtag ημερήσια.

Στην δεύτερη, Client Used, παρουσιάζεται η πληροφορία που αφορά την συσκευή ή τον client, που έχουν χρησιμοποιήσει οι χρήστες όταν δημοσιεύουν κάποιο tweet. Στην πρώτη οθόνη, εμφανίζονται όλοι τα διαθέσιμα sources που έχουν καταχωρηθεί με τον αριθμό των tweets που έχουν καταγραφεί. Ο χρήστης μπορεί να φιλτράρει τα αποτελέσματα καθώς και να χρησιμοποιήσει το μηχανισμό σελίδων για να δει επόμενα αποτελέσματα. Τέλος, μπορεί να επιλέξει να σορτάρει τα αποτελέσματα με βάση τον αριθμό count.



Εικόνα 51 Οθόνη παρουσίασης συσκευών χρησιμοποίησης.



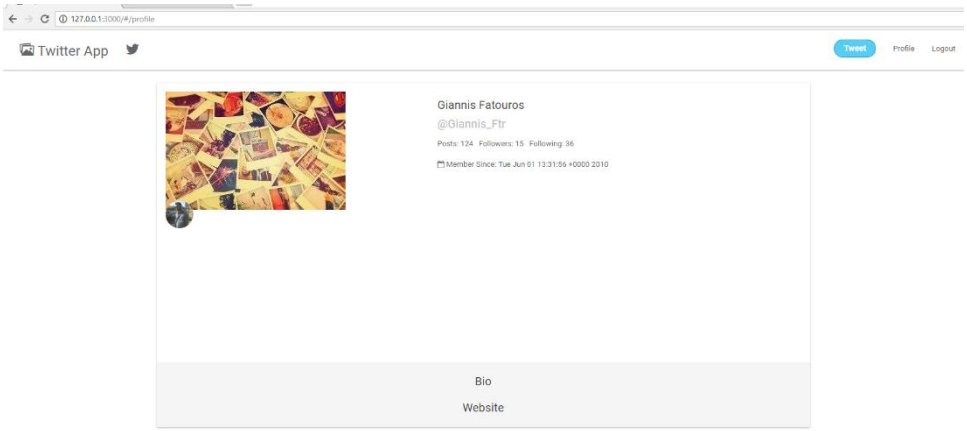
Εικόνα 52 Σορτάρισμα αποτελεσμάτων με βάση το count.

Ακριβώς από κάτω εμφανίζεται διάγραμμα πίτας που παρουσιάζει σε ποσοστά τον αριθμό των tweets που έχουν δημοσιευθεί και με ποια συσκευή ανά hashtag.

Εικόνα 53 Παρουσίαση αποτελεσμάτων με μορφή πίτας.

Profile

Στη συγκεκριμένη οθόνη, παρουσιάζεται το προφίλ του χρήστη, μόνο όταν είναι συνδεδεμένος με το twitter λογαριασμό του.



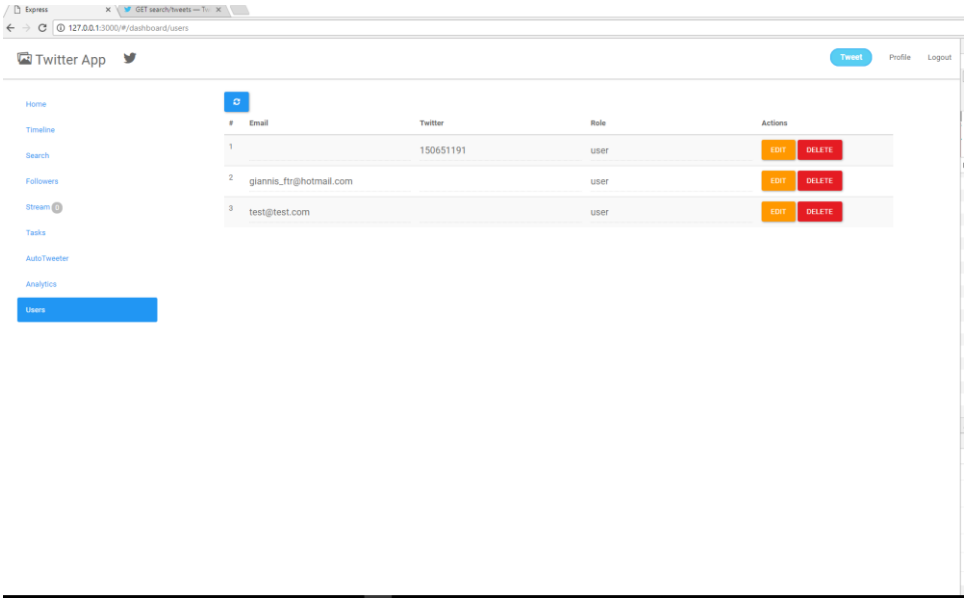
Εικόνα 54 Οθόνη προφίλ του χρήστη.

Logout

Επιλέγοντας το Logout, ο χρήστης αποσυνδέεται από το λογαριασμό του, και ανακατευθύνεται στην αρχική οθόνη της εφαρμογής.

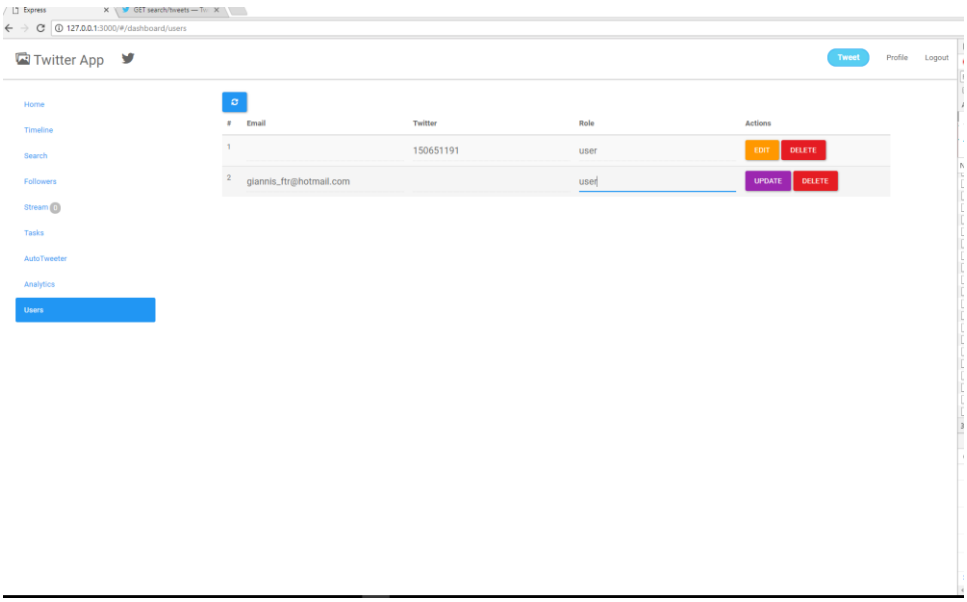
Εικόνα 55 Navigation Bar Top - Logout.

Users (Λειτουργία Admin only)



Εικόνα 56 Οθόνη παρουσίασης των χρηστών της εφαρμογής.

Ένας χρήστης με ρόλο "admin" μπορεί να χρησιμοποιήσει τη συγκεκριμένη λειτουργία για να παρακολουθήσει τους χρήστες της εφαρμογής. Μπορεί να διαγράψει έναν συγκεκριμένο χρήστη καθώς και να του τροποποιήσει το ρόλο του.



Εικόνα 57 Τροποποίηση ενός χρήστη.

Η τροποποίηση γίνεται πατώντας το κουμπί EDIT, μπορεί να αλλαχθεί ο ρόλος του χρήστη και έπειτα να ενημερωθεί πατώντας το κουμπί UPDATE που εμφανίστηκε στη θέση του κουμπιού EDIT.

Η διαγραφή γίνεται απλά πατώντας το κουμπί DELETE, τότε ο χρήστης διαγράφεται από τη βάση δεδομένων μας και αφαιρείται και από τη λίστα με τους χρήστες.

Ο admin μπορεί ανά πάσα στιγμή να κάνει ανανέωση των χρηστών, πατώντας το μπλε κουμπί με το σύμβολο refresh, για να εμφανιστούν νέοι χρήστες που έχουν εγγραφεί.

7. ΤΕΧΝΟΛΟΓΙΕΣ

Με τον όρο Full-stack development, μπορούμε να πούμε ότι είναι η υλοποίηση όλων των μερών ενός προγράμματος ή μιας εφαρμογής / Web εφαρμογής. Η έννοια full stack ξεκινάει με τη βάση δεδομένων και τον web server στο back end, εμπεριέχει το application logic στο ενδιάμεσο και καταλήγει μέχρι το user interface στο λεγόμενο front end. Συγκεκριμένα το MEAN stack που έχει χρησιμοποιηθεί απαρτίζεται από τέσσερις κύριες τεχνολογίες:

1. MongoDB, η βάση δεδομένων.
2. Express, το web framework.
3. AngularJS, το front-end framework.
4. Node.js, ο web server

Η MongoDB υπάρχει από το 2007, και διατηρείται από την MongoDB Inc, προηγουμένως γνωστή ως 10gen.
Το Express, web framework κυκλοφόρησε για πρώτη φορά το 2009 από τον T. J. Holowaychuk και από τότε έχει γίνει το πιο δημοφιλές framework για το Node.js. Είναι ανοιχτού κώδικα, με περισσότερους από 100 συνεργάτες και αναπτύσσεται και υποστηρίζεται ενεργά.
Η AngularJS είναι ανοιχτού κώδικα και υποστηρίζεται από την Google. Υπάρχει από το 2010 και συνεχώς αναπτύσσεται και επεκτείνεται.
Το Node.js δημιουργήθηκε το 2009 και η ανάπτυξή του και η συντήρησή του χρηματοδοτούνται από τη Joyent. Χρησιμοποιεί τον πυρήνα της μηχανής JavaScript V8 της Google κατά κύριο λόγο.
Η επιλογή των παραπάνω τεχνολογιών - stack έγινε γιατί χρησιμοποιώντας αποκλειστικά και μόνο μία γλώσσα προγραμματισμού την JavaScript είναι εφικτό να υλοποιηθεί μια web εφαρμογή από το back end μέχρι το front end.

7.1 Node.js: ο Web server

Με το Node.js μπορούμε να δημιουργήσουμε τον δικό μας Web Server και να υλοποιήσουμε Web εφαρμογές. Δεν είναι αποκλειστικά ένας Web Server ούτε αποτελεί γλώσσα προγραμματισμού. Περιέχει μια ενσωματωμένη βιβλιοθήκη HTTP, πράγμα που σημαίνει ότι δεν χρειάζεται να χρησιμοποιήσουμε ένα ξεχωριστό Web Server, όπως είναι ο Apache, Nginx κτλ. Επιπροσθέτως, η V8 JavaScript engine που χρησιμοποιεί είναι αρκετά γρήγορη και ενθαρρύνει την ανάπτυξη σε ασύγχρονο κώδικα. Αλλά το μεγαλύτερο προτέρημα είναι η δυνατότητα να γράφει ένας developer σε ίδια γλώσσα προγραμματισμού και στον Server και στον Client (Browser).

7.2 Express: Web Framework

Είναι ένα ευέλικτο Web Framework για το Node.js που περιέχει ένα ισχυρό σύνολο λειτουργιών για την υλοποίηση Web εφαρμογών και κινητών εφαρμογών.

7.3 MongoDB: Database

Η MongoDB είναι document – type database. Αντίστοιχα με το σχεσιακό μοντέλο μια γραμμή (εγγραφή) σε ένα πίνακα είναι ένα document (έγγραφο) και αυτό περιγράφει τι δεδομένα εμπεριέχει για αποθύκευση. Πιο ειδικά, η MongoDB αποθηκεύει documents ως BSON, το οποίο είναι binary JSON (JavaScript Serialized Object Notation). Σε γενικές γραμμές, JSON είναι μια δομή για να κρατάμε δεδομένα στην JavaScript. Ένα παράδειγμα τέτοιου document είναι το παρακάτω:

```
{
   "firstName" : "John",
   "lastName" : "Doe",
    _id : ObjectId("52a79effc2aaab0c1000107")
}
```

Όπως παρατηρούμε ένα document εμπεριέχει key – value pairs (ζεύγη). Υποστηρίζει τη χρησιμοποίηση δευτερεύων δεικτών (indexes), όχι μόνο σε πεδία τα οποία είναι μοναδικά, πράγμα που της δίνει μεγάλη ταχύτητα.

7.3.1 Mongoose

Το Data Modeling, γίνεται με τη χρησιμοποίηση του Mongoose, με το οποίο περιγράφουμε τί ακριβώς δεδομένα μπορούν να βρίσκονται σε ένα document. Κατά κύριο λόγο, το mongoose έχει φτιαχτεί από την ίδια την εταιρεία που δημιούργησε και τη MongoDB για να προσπαθήσει να δώσει ένα schema στη βάση δεδομένων, η οποία είναι schema-less, δεν έχει καθορισμένο schema, είναι ευέλικτη, και η μορφή με την οποία αποθηκεύει τα δεδομένα δεν είναι σταθερή. Αυτός είναι και άλλος ένα λόγος για τον οποίο επιλέχθηκε η συγκεκριμένη βάση, καθώς μπορούμε να αλλάζουμε τα μοντέλα που αποθηκεύουμε στα documents συνεχώς κατά την φάση της ανάπτυξης της εφαρμογής. Το Mongoose, μας επιτρέπει να εισάγουμε validations στα μοντέλα που θα χρησιμοποιήσουμε και διαχειρίζεται τη διασύνδεση με τη βάση δεδομένων, παρέχοντάς μας ένα εύκολο API γι' αυτό. Θα μπορούσε να χαρακτηριστεί και σαν ενδιάμεσος ανάμεσα στη βάση μας (MongoDB) και στην Server Side γλώσσα προγραμματισμού (Node.js).

7.4 AngularJS: To Front End Framework

Η AngularJS είναι ένα JavaScript Framework για να μας διευκολύνει να δουλεύουμε με δεδομένα στο front end. Με τον παραδοσιακό τρόπο όλη η επεξεργασία των δεδομένων και η λογική της εφαρμογής βρισκόταν στον Server, ο οποίος μας τα έφερνε στον browser μας μέσω της HTML. Τώρα μπορούμε να μεταφέρουμε λογική στον Browser έυκολα αποφορτίζοντας τον Server μας, μόνο στο να μας παρέχει δεδομένα και τίποτα άλλο. Κάτι το οποίο κάνει καλά η Angular, είναι ότι είναι σχεδιασμένη για single-page application λειτουργικότητα. Με την τεχνολογία αυτή όλα γίνονται μέσα στον Browser και η σελίδα δεν κάνει ποτέ full reload. Πράγμα που σημαίνει ότι όλη η λογική της εφαρμογής, η επεξεργασία των δεδομένων, τα templates διαχειρίζονται από τον Browser. Συνεπώς, μειώνεται και ο αριθμός των πόρων που χρειαζόμαστε από τον Server, καθώς ο Browser του χρήστη αναλαμβάνει να κάνει τη δύσκολη δουλειά και ο Server μένει μόνο να μας σερβίρει στατικά αρχεία HTML και δεδομένα σε κάθε request.

7.5 Twitter Bootstrap

Για την υλοποίηση του user interface έχει χρησιμοποιηθεί το Bootstrap, το οποίο είναι ένα front-end framework όπου παρέχει ένα σύνολο από CSS Classes, styles και HTML components έτσι ώστε να διευκολύνει και να επιταχύνει τη δημιουργία των HTML σελίδων της εφαρμογής.

ΔΙΑΣΥΝΔΕΣΗ ΜΕ ΕΞΩΤΕΡΙΚΑ ΣΥΣΤΗΜΑΤΑ

Η διασύνδεση και η επικοινωνία με το Twitter για ανταλλαγή και την αποθήκευση πληροφορίας στην εφαρμογή πραγματοποιείται με τη χρησιμοποίηση του REST API και του Streaming API που προσφέρει η πλατφόρμα του Twitter. Τα βασικά endpoints είναι διαθέσιμα δωρεάν.

ΔΗΜΙΟΥΡΓΙΑ ΕΦΑΡΜΟΓΗΣ ΣΤΟ TWITTER

Το βασικό REST και το Streaming API προυποθέτουν την δημιουργία μιας εφαρμογής στο Twitter έτσι ώστε να μπορέσουμε να χρησιμοποιήσουμε το OAuth-based authorization σύστημα της πλατφόρμας. Για να το πραγματοποιήσουμε αυτό, επισκεπτόμαστε την σελίδα apps.twitter.com, για τη δημιουργία μιας.

Έπειτα επιλέγουμε Create New App, έτσι ώστε να δημιουργήσουμε τη δικιά μας εφαρμογή.

Σο πεδίο Name - όνομα, αρκεί να δώσουμε το όνομα της εφαρμογής μας. Αυτό το όνομα θα παρουσιάζετε στους χρήστες όταν τους ζητείτε να δώσουν πρόσβαση στην εφαρμογή να χρησιμοποιήσει τα στοιχεία τους στο Twitter.
Στο πεδίο Description - περιγραφή, μπορούμε να δώσουμε μια σύντομη περιγραφή της εφαρμογής μας, και πάλι αυτό το θα εμφανιστεί στους χρήστες κατά το βήμα αυθεντικοποίησης.
Στο πεδίο Website - Ιστοσελίδα, πληκτρολογούμε το URL που δείχνει στην εφαρμογή μας.
Στο πεδίο Callback URL, πληκτρολογούμε το URL όπου το twitter θα ανακατευθύνει το χρήστη έπειτα από μια επιτυχημένη αυθεντικοποίηση.
Τέλος, για να προχωρήσουμε στην δημιουργία αρκεί να τσεκάρουμε το Yes,
I have read and agree to the Twitter Developer Agreement.
Πρέπει να συμπληρώσουμε όλα τα απαραίτητα, υποχρεωτικά πεδία που μας ζητούνται και όταν είμαστε έτοιμοι, δημιουργούμε την εφαρμογή πατώντας το κουμπί "Create your Twitter application".
Μόλις ολοκληρώσουμε με επιτυχία το βήμα αυτό, τότε μεταβαίνουμε στην οθόνη application settings, όπου μπορούμε να επεξεργαστούμε στοιχεία της εφαρμογής μας, και περιέχει και τα απαραίτητα συστατικά για να μπορούμε να χρησιμοποιήσουμε το REST API του Twitter.

ΑΥΘΕΝΤΙΚΟΠΟΙΗΣΗ

Το Twitter χρησιμοποιεί το OAuth πρωτόκολλο για την παροχή εξουσιοδοτημένης πρόσβασης.
Υπάρχουν δύο επιλογές για να αποκτήσουμε πρόσβαση:

1. User authentication - έλεγχος ταυτότητας χρήστη

Αυτή είναι η πιο κοινή μορφή αυθεντικοποίησης που χρησιμοποιεί το OAuth 1.0a πρωτόκολλο. Υπάρχουν αρκετές βιβλιοθήκες που υλοποιούν το πρωτόκολλο αυτό και το Twitter προτρέπει να χρησιμοποιούνται αυτές. Για κάθε γλώσσα προγραμματισμού υπάρχουν αρκετές υλοποιήσεις. Για την εφαρμογή θα χρησιμοποιήσουμε την βιβλιοθήκη "twit" η οποία είναι για το Node.js και υποστηρίζει και το REST και το Steaming API.
https://github.com/ttezel/twit

2. Application-only authentication - έλεγχος ταυτότητας μόνο για εφαρμογή

Μια εφαρμογή κάνει μια κλήση σε κάποιο από τα endpoints του REST API για λογαριασμό της, χωρίς να χρειάζεται να γνωρίζει ποιος χρήστης το πραγματοποιεί. Οι κλήσεις στο API εξακολουθούν να είναι

περιορισμένες βάσει της μεθόδου. Πιο συγκεκριμένα αυτός ο περιορισμός αναφέρεται σε πόσα requests - αιτήματα για κάποια μέθοδο μπορεί να γίνει σε διάστημα 15 λεπτών χρόνου.

ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Βάση δεδομένων της εφαρμογής, όπως παρουσιάστηκε παραπάνω, έχει επιλεχθεί η MongoDB. Οποιαδήποτε εγγραφή ή προσπέλαση στη βάση γίνεται με τη χρήση του Mongoose framework (ORM), και με τη βοήθεια του WEB API που έχει υλοποιηθεί με το Express.js framework.
Πιο συγκεκριμένα, για τις ανάγκες της εφαρμογής χρησιμοποιήθηκαν τα παρακάτω collections για την αποθήκευση των δεδομένων.

1. Users
   Εδώ κρατάμε την πληροφορία για τους χρήστες που έχουν εγγραφεί στην εφαρμογή.
   - _id, το μοναδικό αναγνωριστικό που χρησιμοποιεί η MongoDB για indexing.
   - Twitter (String), το twitter id που έχουμε αν κάνουμε εγγραφή στην εφαρμογή με τη χρήση του twitter λογαριασμού μας.
   - Email (String), το email του χρήστη σε περίπτωση που κάνουμε εγγραφή στην εφαρμογή χωρίς τη χρήση του Twitter.
   - Password (String), το συνθηματικό του χρήστη σε περίπτωση που κάνουμε εγγραφή στην εφαρμογή χωρίς τη χρήση του Twitter, κρυπτογραφημένο με τη βιβλιοθήκη bcrypt(1).
   - Picture (String), το url της φωτογραφίας του χρήστη από το Twitter.
   - AccessToken (String), το OAuth token που έχουμε πάρει από το Twitter κατά τη σύνδεση του λογαριασμού μας.
   - Role (String), ο ρόλος του χρήστη, μπορεί να είναι είτε "user" είτε "admin".
   - Tasks (Reference id), είναι Array από _id που δείχνουν στο collection Tasks.
2. Tweets
   Εδώ αποθηκεύουμε όλα τα tweets που επιθυμεί ο χρήστης να κάνει track.
   - _id, το μοναδικό αναγνωριστικό που χρησιμοποιεί η MongoDB για indexing.
   - Twitter_id (string), το id του tweet που έχει αποδοθεί από το Twitter.
   - Author (String), το όνομα του χρήστη που έχει δημοσιεύσει το tweet.
   - Created_at (Date), η ημερομηνία που δημιουργήθηκε το tweet στη βάση.
   - Updated_at (Date), η ημερομηνία που έγινε update το tweet στη βάση.
   - Screenname (String), το όνομα που έχει επιλέξει ο χρήστης να εμφανίζεται.
   - Source (String), η συσκευή που χρησιμοποιήθηκε για να δημοσιευθεί το tweet.
   - Lang (String), η γλώσσα που γράφτηκε το tweet.
   - Hashtags (String Array), τα hashtags που ακολουθούν το συγκεκριμένο tweet.
   - Date (Date), η ημερομηνία δημοσίευσης του tweet.
   - Avatar (String), η εικόνα που έχει επιλέξει ο χρήστης ώς εικόνα προφίλ ή avatar.
   - Body (String), το κείμενο που περιέχετε στο tweet.
3. Tasks
   Εδώ αποθηκεύουμε όλες τις εργασίες - Tasks που θέλει να κρατήσει ο χρήστης για υπενθύμιση.
   - _id, το μοναδικό αναγνωριστικό που χρησιμοποιεί η MongoDB για indexing.
   - User (Reference id), πρόκειται για το _id του user που έχει καταχωρήσει την εργασία.

- Subject (String), το κείμενο που έχει επιλέξει να εμφανίζεται στη συγκεκριμένη εργασία.
- Completed (Boolean), δείχνει αν η συγκεκριμένη εργασία είναι ολοκληρωμένη ή όχι.

4. Hashtags

Εδώ αποθηκεύουμε όλα τα διαφορετικά hashtags που κάνει track η εφαρμογή μας. Κάθε hashtag, αποτελεί ένα ξεχωριστό document (έγγραφο) στη βάση δεδομένων μας.

- _id, το μοναδικό αναγνωριστικό που χρησιμοποιεί η MongoDB για indexing.
- Created_at (Date), η ημερομηνία που δημιουργήθηκε το hashtag στη βάση.
- Updated_at (Date), η ημερομηνία που έγινε update το hashtag στη βάση.
- Text (String), το κείμενο που δηλώνει ποιο είναι το περιεχόμενο του hashtag.
- Tweets (Array από Reference ids), πρόκειται για δείκτη προς το Tweet Collection, και αφορά τα tweet ids που σχετίζονται με το συγκεκριμένο hashtag.

5. Tweetscheduleds

Σε αυτό το Collection αποθηκεύονται όλα τα tweets που επιθυμούν οι χρήστες να καταχωρήσουν και να προγραμματίσουν τη δημοσίευσή τους αυτόματα από το σύστημα σε μελλοντικό χρόνο, χωρίς να γίνει η δημοσίευση από τους ίδιους.

- _id, το μοναδικό αναγνωριστικό που χρησιμοποιεί η MongoDB για indexing.
- Text (String), το κείμενο που περιέχετε στο tweet.
- Datetime (Date), η ημερομηνία και ώρα που έχει προγραμματίσει ο χρήστης να γίνει η δημοσίευση του συγκεκριμένου tweet.
- Completed (Boolean), δηλώνει αν ένα tweet έχει δημοσιευθεί ή όχι.

6. Trackbies

Σε αυτό το Collection αποθηκεύονται όλα τα keywords που θέλουν οι χρήστες. Με βάση αυτά ενεργοποιείται το Streaming API του Twitter και δίνει αποτελέσματα - tweets. Στη συγκεκριμένη περίπτωση αφορά όλα τα hashtags που θέλει να κάνει καταγραφή ο χρήστης.

- _id, το μοναδικό αναγνωριστικό που χρησιμοποιεί η MongoDB για indexing.
- Keyword (String), το κείμενο - λέξη που καταχωρεί ο χρήστης.

7. Analytics

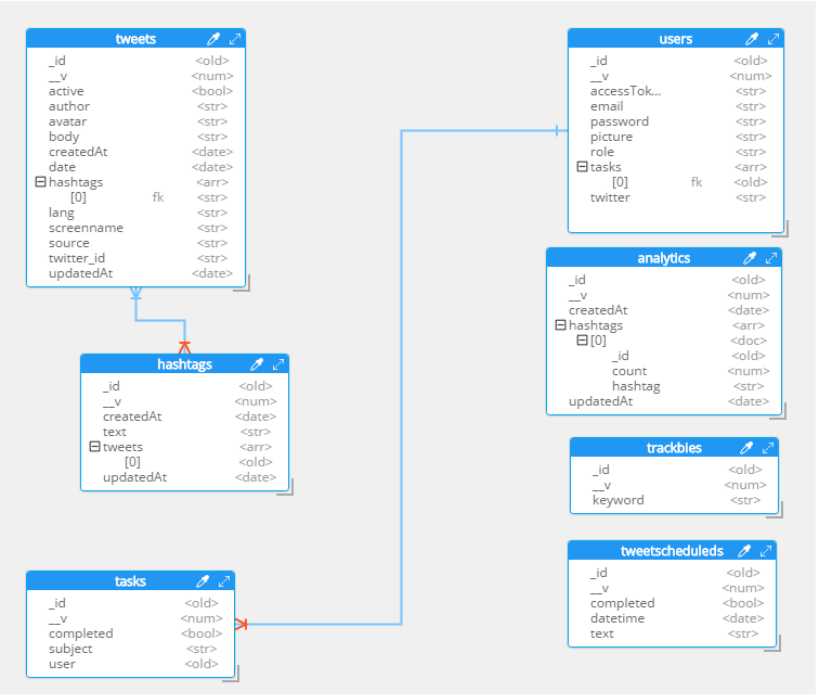
Σε αυτό το Collection αποθηκεύονται όλα τα analytics, δηλαδή οι τιμές
- counters των tweets στο τέλος της ημέρας ανά hashtag που έχει καταχωρήσει ο χρήστης στο Collection (TrackBies)

- _id, το μοναδικό αναγνωριστικό που χρησιμοποιεί η MongoDB για indexing.
- Created_at (Date), η ημερομηνία που δημιουργήθηκε στη βάση.
- Updated_at (Date), η ημερομηνία που έγινε update στη βάση.
- Hashtags (Array)
  - Object
    - Hashtag (String), το κείμενο του hashtag
    - Count (Integer), ο αριθμός των tweets που έχουν δημοσιευθεί και σχετίζονται με το συγκεκριμένο hashtag.
    - _id (Reference id), πρόκειται για δείκτη στο Hashtag Collection και σχετίζεται με το Hashtag που έχει καταχωρηθεί.

Παρουσίαση ER διαγράμματος.

Παρακάτω παρουσιάζετε το διάγραμμα ER της βάσης.

1. Ένας User - χρήστης μπορεί να έχει πολλά tasks, ένα task ανήκει σε ένα χρήστη
2. Ένα Tweet μπορεί να έχει πολλά hashtags, ένα hashtag μπορεί να ανήκει σε πολλά tweets.



Εικόνα 58 ER διάγραμμα.

Web Services

Γενικά

Το REST (συντομία - ακρωνύμιο για τον όρο REpresentational State Transfer) είναι ένα αρχιτεκτονικό στυλ που έχει καθοριστεί για να βοηθήσει στη δημιουργία και οργάνωση κατανεμημένων συστημάτων. Η λέξη - κλειδί από τον ορισμό αυτό πρέπει να είναι το στυλ, επειδή μια συμαντική πτυχή του REST, είναι ότι είναι ένα αρχιτεκτονικό στυλ - όχι κατευθυντήρια γραμμή, όχι πρότυπο, αυτό θα σήμαινε ότι υπάρχει μια σειρά αυστηρών κανόνων που πρέπει να ακολουθούμε για να καταλήξουμε να έχουμε μια RESTful αρχιτεκτονική. Η βασική ιδέα πίσω από το REST είναι ότι ένα κατανεμημένο σύστημα, οργανωμένο RESTfully, θα βελτιωθεί στους ακόλουθους τομείς:

- Απόδοση: Ο τρόπος επικοινωνίας που προτείνεται από το REST πρέπει να είναι αποτελεσματικός και απλός, επιτρέποντας την ενίσχυση των επιδόσεων στα συστήματα που το υιοθετούν.
- Επεκτασιμότητα: Κάθε κατανεμημένο σύστημα θα πρέπει να είναι σε θέση να χειριστεί αυτή την πτυχή αρκετά και η απλή αλληλεπίδραση που προτείνεται από το REST επιτρέπει σε μεγάλο βαθμό αυτό.

- Απλότητα διεπαφής (interface): Μια απλή διεπαφή επιτρέπει την απλούστερη αλληλεπίδραση μεταξύ συστημάτων, η οποία με τη σειρά της μπορεί να προσφέρει οφέλη όπως αυτά που αναφέρθηκαν προηγουμένως.
- Τροποποίηση: Ο κατανεμημένος χαρακτήρας του συστήματος και ο διαχωρισμός που προτείνει το REST, επιτρέπει την τροποποίηση ανεξάρτητων στοιχείων μεταξύ τους με ελάχιστο κόστος και κίνδυνο.
- Φορητότητα: Το REST δεν εξαρτάται από τη γλώσσα προγραμματισμού και την τεχνολογία που θα επιλέξουμε να χρησιμοποιήσουμε και μπορεί να "καταναλωθεί" από οποιοδήποτε είδος τεχνολογίας.
- Αξιοπιστία: Ο περιορισμός stateless που προτείνεται από το REST, επιτρέπει την ευκολότερη ανάκτηση ενός συστήματος μετά από αποτυχία.
- Ορατότητα: Κάθε σύστημα δεν χρειάζεται να κοιτάξει πέρα από ένα μήνυμα αίτησης (request) για να καθορίσει την πλήρη κατάσταση του αιτήματος.

Περιορισμοί του REST

Έξι κύριοι περιορισμοί ορίζουν ένα σύστημα ως RESTful. Αυτοί οι περιορισμοί ορίζουν τους τρόπους με τους οποίους ο διακομιστής (server) μπορεί να επεξεργάζεται και να ανταποκρίνεται στις αιτήσεις του πελάτη (client), έτσι ώστε το σύστημα να αποκτά τις παραπάνω ιδιότητες. Εάν μια υπηρεσία - σύστημα παραβιάζει κάποιον από τους απαιτούμενους περιορισμούς, δεν μπορεί να θεωρηθεί RESTful.
Οι τυπικοί περιορισμοί REST έχουν ως εξής:

Client-server architecture

Οι πρώτοι περιορισμοί που προστίθενται είναι εκείνοι του αρχιτεκτονικού στυλ πελάτη - εξυπηρετητή. Η βασική αρχή πίσω από τους περιορισμούς πελάτη - διακομιστή είναι ο διαχωρισμός των αρμοδιοτήτων. Ο διαχωρισμός αυτός, της διεπαφής χρήστη από τις αρμοδιότητες αποθήκευσης δεδομένων βελτιώνει τη φορητότητα του περιβάλλοντος του χρήστη σε πολλαπλές πλατφόρμες. Επίσης, βελτιώνει την επεκτασιμότητα και τέλος επιτρέπει στα διάφορα μέρη ενός συστήματος να εξελιχθούν - ανεξάρτητα.

Stateless

Η επικοινωνία πελάτη - διακομιστή περιορίζεται από το γεγονός ότι δεν αποθηκεύεται στον διακομιστή καμία κατάσταση από τον πελάτη μεταξύ των αιτημάτων. Κάθε αίτημα από οποιονδήποτε πελάτη περιέχει όλες τις πληροφορίες που είναι απαραίτητες για την εξυπηρέτηση του αιτήματος και η κατάσταση της περιόδου σύνδεσης διατηρείται στον πελάτη. Η κατάσταση της περιόδου σύνδεσης μπορεί να μεταφερθεί από το διακομιστή σε μια άλλη υπηρεσία, όπως μια βάση δεδομένων, για να διατηρηθεί και να επιτραπεί η επαλήθευση ταυτότητας. Ο πελάτης αρχίζει να στέλνει αιτήματα όταν είναι έτοιμος να κάνει τη μετάβαση σε μια νέα κατάσταση. Η αναπαράσταση κάθε κατάστασης περιέχει συνδέσμους που μπορούν να χρησιμοποιηθούν την επόμενη φορά που ο πελάτης επιλέγει να ξεκινήσει μια νέα κατάσταση μετάβασης.

Cacheability

Οι πελάτες και οι ενδιάμεσοι μπορούν να αποθηκεύσουν απαντήσεις (responses) σε cache. Η σωστή διαχείριση της προσωρινής αυτής αποθήκευσης εξαλείφει μερικώς ή εντελώς κάποιες αλληλεπιδράσεις πελάτη - διακομιστή, βελτιώνοντας περαιτέρω την επεκτασιμότητα και την απόδοση.

Layered system

Ένας πελάτης δεν μπορεί συνήθως να πει αν είναι συνδεδεμένος απευθείας με τον τελικό διακομιστή ή με έναν ενδιάμεσο. Οι ενδιάμεσοι διακομιστές ενδέχεται να βελτιώσουν τη δυνατότητα επέκτασης του συστήματος, επιτρέποντας την εξισορρόπηση φορτίου (load balancing) και την παροχή κοινής μνήμης (caches).

Code on demand (προαιρετικό)

Οι διακομιστές μπορούν προσωρινά να επεκτείνουν ή να προσαρμόσουν τη λειτουργικότητα ενός πελάτη μεταφέροντας εκτελέσιμο κώδικα.

Uniform interface

Ο περιορισμός αυτός είναι θεμελιώδης για την αρχιτεκτονική οποιαδήποτε REST service. Απλοποιεί και χωρίζει την αρχιτεκτονική, επιτρέποντας σε κάθε μέρος να εξελιχθεί ανεξάρτητα. Υπάρχουν τέσσερις περιορισμοί:

Resource identification in requests

Ο διακομιστής μπορεί να στείλει δεδομένα από τη βάση δεδομένων τους ως HTML, XML ή JSON, κανένα από τα οποία δεν είναι η εσωτερική αναπαράσταση του διακομιστή.

Resource manipulation through representations

Όταν ένας υπολογιστής - πελάτης κατέχει μια αναπαράσταση ενός
πόρου, συμπεριλαμβανομένων οποιουδήποτε metadata που το ακολουθεί, διαθέτει αρκετές πληροφορίες για την τροποποίηση ή τη διαγραφή ου πόρου.

Self-descriptive messages

Κάθε μήνυμα περιλαμβάνει αρκετές πληροφορίες για να περιγράψει τον τρόπο επεξεργασίας του μηνύματος.

Hypermedia as the engine of application state (HATEOAS)

Έχοντας πρόσβαση σε ένα αρχικό URI για την εφαρμογή REST, ένας πελάτης θα πρέπει στη συνέχεια να μπορεί να χρησιμοποιεί συνδέσμους που παρέχονται από το διακομιστή δυναμικά για να ανακαλύψει όλες τις διαθέσιμες ενέργειες και πόρους που χρειάζεται.

Υλοποίηση

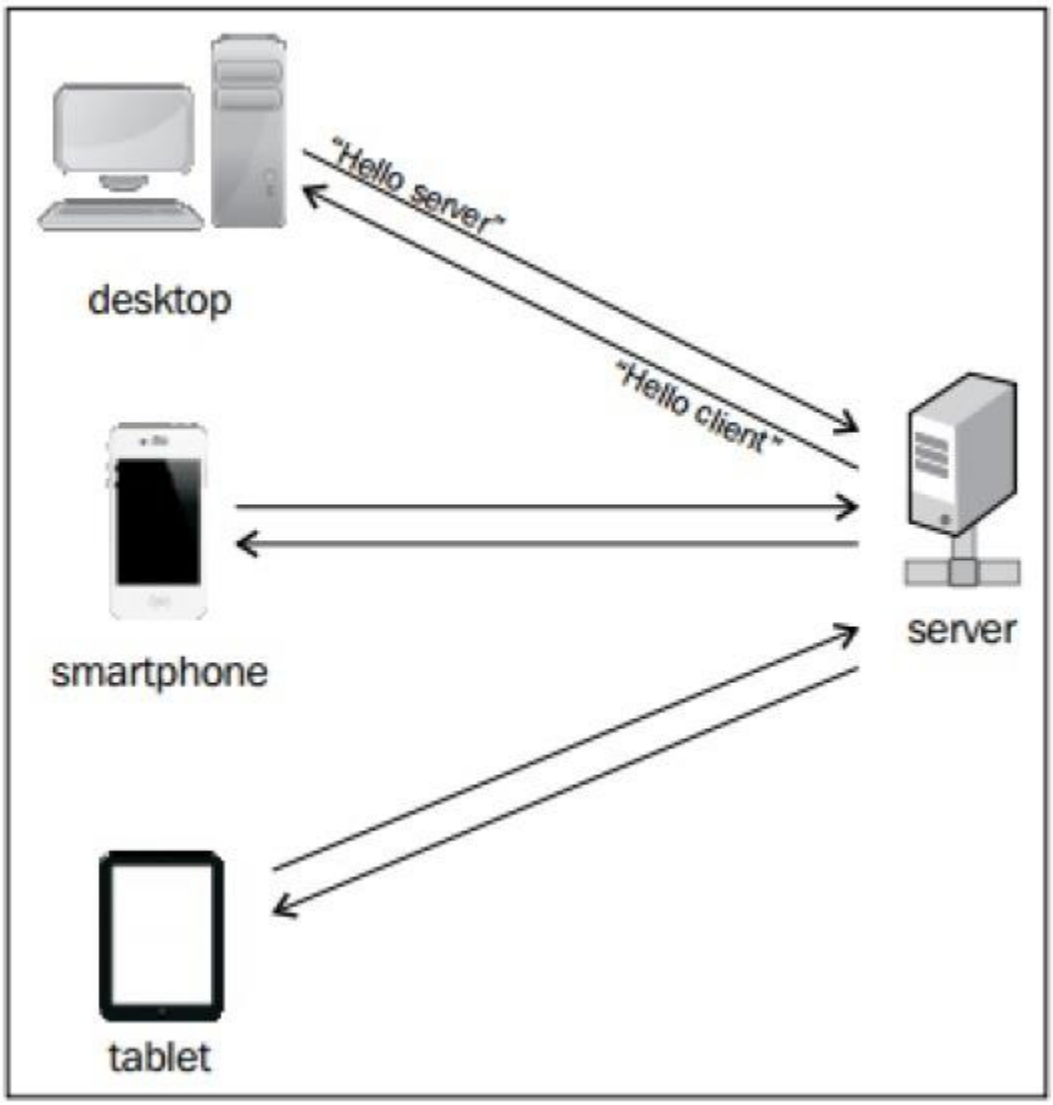
Για την υλοποίηση των REST services χρησιμοποιήθηκε το Node.js και πιο συγκεκριμένα το Express.js με τη βοήθεια του οποίου φτιάχτηκαν τα διαφορετικά endpoints για να μπορούν να χρησιμοποιούν οι πελάτες - clients του συστήματος.
Η εφαρμογή - πελάτης από τη μεριά της καταναλώνει τα services αυτά χρησιμοποιώντας τη γλώσσα προγραμματισμού Javascript, σχεδιασμένη και οργανωμένη με το Angular.js (framework).

SOCKET

Λίγα λόγια

Ως Web Sockets ορίζεται η αμφίδρομη επικοινωνία μεταξύ servers και clients, πράγμα που σημαίνει ότι και τα δύο αυτά μέρη επικοινωνούν και ανταλλάσσουν δεδομένα ταυτόχρονα.

Το μεγαλύτερο πλεονέκτημα των web sockets είναι ότι παρέχει αμφίδρομη επικοινωνία σε μία μόνο σύνδεση TCP, επικοινωνία σε πραγματικό χρόνο μεταξύ server και client, cross-platform πρότυπο.



Εικόνα 59 Web socket protocol example.

Socket.io

Το socket.io είναι μια βιβλιοθήκη για το Node.js που επιτρέπει την αμφίδρομη επικοινωνία βασισμένη σε γεγονότα (events) και σε πραγματικό χρόνο.

Υλοποίηση

Για την ενσωμάτωση του socket.io απαιτούνται τα παρακάτω δύο μέρη:

- Ένας server - διακομιστής με το Node.JS HTTP server: socket.io, σε διαφορετική port από την εφαρμογή.

```
var server = require('http').createServer(app);
 var io = require('socket.io')(server);
 server.listen(4000);
```

- Μια βιβλιοθήκη για τον client - πελάτη, που φορτώνεται στον browser του χρήστη: socket.io-client.
- Event, που γίνεται emit από τον server, `io.emit('tweet', tweet);`

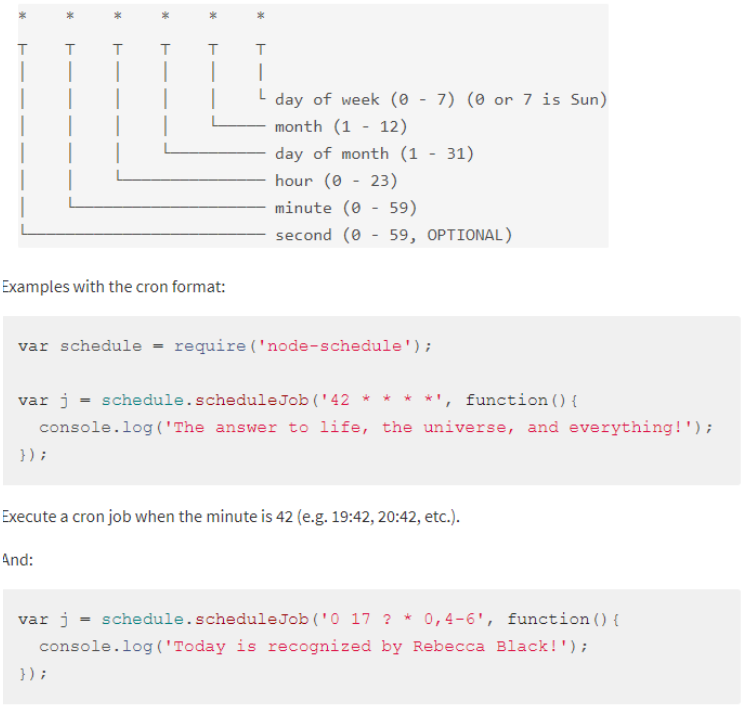
Με τη βιβλιοθήκη αυτή μπορούμε να στέλνουμε και να λαμβάνουμε οποιοδήποτε event επιθυμούμε, με οποιοδήποτε δεδομένα, σε μορφή JSON.
Συγκεκριμένα ο client έχει οριστεί να ακούει για το event "tweet". Με αυτό τον τρόπο μπορούμε να περνάμε πληροφορία από το server σε οποιοδήποτε χρήστη είναι συνδεδεμένος στο socket. Τελικά ενημερώνεται για το πόσα νέα tweets έχουν καταγραφεί στην βάση δεδομένων.

CRON JOBS

Είναι ένα βοηθητικό πρόγραμμα - λειτουργία, που προγραμματίζει μια εντολή ή μια σειρά ενεργειών σε έναν διακομιστή, για αυτόματη εκτέλεση σε συγκεκριμένη ώρα και ημερομηνία. Ένα cron job είναι η ίδια η προγραμματισμένη εργασία. Είναι πολύ χρήσιμες για την αυτοματοποίηση επαναλαμβανόμενων εργασιών. Για παράδειγμα, μπορούμε να ορίσουμε μια εργασία cron για να διαγράφει προσωρινά αρχεία κάθε εβδομάδα για να εξοικονομείτε χώρος στο δίσκο είτε για την αποστολή ειδοποιήσεων μέσω ηλεκτρονικού ταχυδρομείου.

Node-schedule

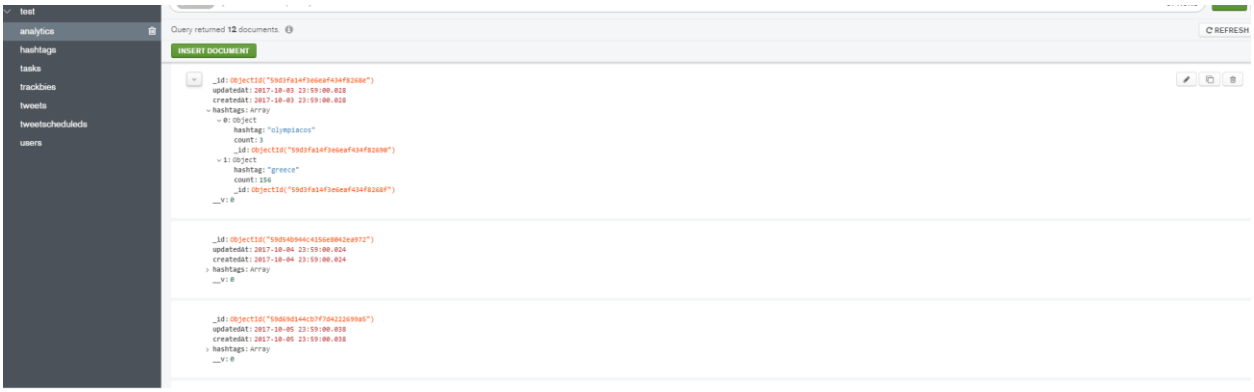
Το Node schedule είναι μια βιβλιοθήκη που μας επιτρέπει να προγραμματίζουμε εργασίες για εκτέλεση σε συγκεκριμένες ημερομηνίες, με προαιρετικούς κανόνες επανάληψης. Χρησιμοποιεί ένα μόνο χρονοδιακόπτη ανά πάσα χρονική στιγμή αντί να επανεκτιμά τις εργασίες που είναι προς εκτέλεση κάθε δευτερόλεπτο / λεπτό.



Εικόνα 60 Node schedule. Προγραμματισμός ενός νέου cron job.

Λειτουργικότητα

- Προγραμματισμός ενός νέου tweet σε μια συγκεκριμένη ημερομηνία και ώρα. Τα προγραμματισμένα αυτά tweets καταχωρούνται σε δικό τους Collection, που ονομάζεται tweetscheduled. Το cron job έχει οριστεί ανά λεπτό, και εξετάζουμε από τα δεδομένα του συγκεκριμένου Collection, πόσα πληρούν τις προυποθέσεις τις ώρας και είναι completed: false. Όταν ένα προγραμματισμένο tweet δημοσιευθεί, τότε γίνεται update αυτόματα το property completed σε true.
- Συγκέντρωση αναλυτικών κάθε μέρα στις 23:59, για καταγραφή ιστορικού. Με την συγκεκριμένη εργασία, καταγράφεται η πληροφορία που αφορά, πόσα tweets έχουν καταγραφεί με τα hashtags που έχουμε επιλέξει. Η πληροφορία αυτή, καταγράφεται στη βάση δεδομένων της εφαρμογής, στο Collection που ονομάζεται analytics με τη μορφή:



Εικόνα 61 Schema του μοντέλου analytics.

API

Σε αυτό το τμήμα θα παρατεθούν όλα τα web services που έχουν υλοποιηθεί για την εφαρμογή. Μπορούμε να στείλουμε παραμέτρους μέσω του body και σε μορφή JSON (application/json), ενώ όλα τα URLs ξεκινούν με τη μορφή:

http://127.0.0.1:3000/api/

REGISTER

Title	***REGISTER***

	Πραγματοποιείται η εγγραφή του χρήστη στο σύστημα ***Στέλνει ο χρήστης τα δεδομένα που είναι απαραίτητα για την πραγματοποίηση της εγγραφής του στο σύστημα. Σαν απάντηση δέχεται επιβεβαίωση της εγγραφής του εφόσον είναι επιτυχής, αλλιώς δέχεται ένα μήνυμα λάθους που επιδεικνύει τι δεν πήγε σωστά.***
URL	`/auth/signup`
Method	*POST*

Data Params	{
	"email": [String],
	"password": [String]
	}
	
	Example:
	{
	"email": "test@test.com",
	"password": "test"
	}

Success Response	**Success:** Successful registration
	
	Status Code: 200 OK
	Content:
	{
	"token": [JWT],
	"user": [User Object]: {
	"email": "test@test.com",
	"role": "user",
	"tasks": [],
	"_id": "59ed9c526cd53d982397a6fe"
	},

	}

Error Response	**Error:** Email is already taken
	
	Status Code: 409 CONFLICT
	
	Content:
	{
	
	"message": "Email is already taken."
	
	}

LOGIN

Title	***LOGIN*** ***Πραγματοποιείται η σύνδεση του χρήστη στην εφαρμογή*** ***Ο χρήστης στέλνει τα δεδομένα που είναι απαραίτητα για την σύνδεση με την εφαρμογή. Αυτά είναι το email του χρήστη και ο κωδικός του.***
URL	`/auth/login`
Method	*POST*

Data Params	{
	
	"email": [String],
	
	"password": [String]
	
	}
	
	Example:
	
	{
	
	"email": "test@test.com",
	
	"password": "test"
	
	}

Success Response	**Success1:** Successful login
	
	Status Code: 200 OK
	Content:
	{
	"token": [JWT],
	"user": [User Object]: {
	"email": "test@test.com",
	"role": "user",
	"tasks": [],
	"_id": "59ed9c526cd53d982397a6fe"
	},
	}
	
	Error2: Incorrect email
	Content:
	{
	"message": {
	"password": "Incorrect email"

Error Response	**Error1:** Incorrect password
	
	Status Code: 401 Unauthorized
	Content:
	{
	"message": {
	"password": "Incorrect password"
	}
	}

	} }

SIGN IN WITH TWITTER

(a) Βήμα 1.

Title	***SIGN IN WITH TWITTER (a)*** ***Πραγματοποιείται η εγγραφή του χρήστη στο σύστημα μέσω του Twitter. Στέλνοντας ένα request στο https://api.twitter.com/oauth/authenticate πραγματοποιείται η αυθεντικοποίηση του χρήστη μέσω του twitter και έπειτα γυρνώντας στην εφαρμογή, αν ο χρήστης δεν είναι ήδη εγγεγραμμένος καταχωρείται.***
URL	**/auth/twitter**
Method	***POST***

Data Params	{
	
	"authorizationEndpoint": "https://api.twitter.com/oauth/authenticate",
	
	"defaultUrlParams": null,
	
	"name": "twitter",
	
	"oauthType": "1.0",
	
	"popupOptions": { height: "645", width: "495" },
	
	"redirectUri": "http://localhost:3000",
	
	"redirectUrlParams": null,
	
	"scope": null,
	
	"scopeDelimiter": null,
	
	"scopePrefix": null,
	
	"url": "/auth/twitter"
	
	}

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	"oauth_callback_confirmed": "true",
	"oauth_token": [String],
	"oauth_token_secret": [String]
	}

Error Response	

(b) Βήμα 2

Title	***SIGN IN WITH TWITTER (b)*** ***Πραγματοποιείται η εγγραφή του χρήστη στο σύστημα μέσω του Twitter. Στέλνοντας ένα request στο https://api.twitter.com/oauth/authenticate πραγματοποιείται η αυθεντικοποίηση του χρήστη μέσω του twitter και έπειτα γυρνώντας στην εφαρμογή, αν ο χρήστης δεν είναι ήδη εγγεγραμμένος καταχωρείται.***
URL	`/auth/twitter`
Method	***POST***

Data Params
 	
(τα δεδομένα που πήραμε ώς απάντηση απο το βήμα 1)
	{
	"oauth_token": [String],
	"oauth_token_secret": [String]
	}

Success Response	**Success:** Successful request **Status Code: 200 OK** Content: {

	<pre>"token": [JWT],
	"user": [User Object]: {
	"accessToken": [JWT],
	"picture": [String],
	"role": [String],
	"_id": "59ed9c526cd53d982397a6fe",
	"twitter": [String]
	},
	}</pre>

Error Response	

HOME TIMELINE

Title	***HOME TIMELINE*** ***Παρουσιάζεται όλο το timeline ενός χρήστη όπως αυτό θα εμφανιζόταν εάν ήταν συνδεδεμένος στον Twitter λογαριασμό του.***
URL	`/api/home_timeline`
Method	***GET***
Data Params	

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	[Array Tweet Object],
	}

Error Response	

USER TIMELINE

Title	***USER TIMELINE*** ***Παρουσιάζεται όλο το timeline ενός χρήστη όπως αυτό θα εμφανιζόταν εάν ήταν συνδεδεμένος στον Twitter λογαριασμό του. Συγκεκριμένα αφορά όλα τα tweets που ο χρήστης έχει δημοσιεύσει από το λογαριασμό του.***
URL	`/api/user_timeline`
Method	***GET***
Data Params	

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	
	`[Array Tweet Object],`
	
	}

Error Response	

DELETE TWEET FROM TIMELINE

Title	***DELETE TWEET FROM TIMELINE*** ***Παρουσιάζεται η δυνατότητα διαγραφής μιας δημοσίευσης-Tweet, από το User Timeline.***
URL	`/api/statuses/destroy`
Method	***POST***

Data Params	{
	
	`"tweet_id": [String]`
	
	}

Success Response	**Success:** Successful request **Status Code: 200 OK** Content: {

	[Tweet Object] }
Error Response	

SEARCH

Title	***SEARCH*** ***Παρουσιάζεται η δυνατότητα αναζήτησης περιεχομένου από tweets με βάση κάποιο keyword ή κάποιο query. Η αναζήτηση αυτή ψάχνει tweets που έχουν δημοσιευθεί τις τελευταίες 7 ημέρες.***
URL	`/api/search`
Method	***POST***

Data Params	{
	
	"q": [String],
	
	"count": [Integer],
	
	"result_type": [String], - "mixed", "recent", "popular"
	
	}

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	
	"statuses": [Array Tweet Object],
	
	"search_metadata": {
	
	"completed_in": [Number],
	
	"count": [Number],
	
	"next_results": [String Url],
	
	"query": [String],
	
	"refresh_url": [String Url]

	} }
Error Response	

ADD TASK

Title	***ADD TASK*** ***Παρουσιάζεται η δυνατότητα δημιουργίας μιας εργασίας από έναν χρήστη.***
URL	`/api/task`
Method	***POST***

Data Params	{
	
	"subject": [String]
	
	}

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	
	"info": [String],
	
	"data": {
	
	"completed": [Boolean],
	
	"subject": [String],
	
	"user": [String-id of user reference],
	
	"_id": [String]
	
	}
	
	}

Error Response	**Status Code: 400 Bad Request** Content: { "info": "Error during task create",

	<pre>"error": [Error]
	
	}</pre>

VIEW/READ TASKS

Title	*VIEW/READ TASKS* *Επιστρέφει όλα τα διαθέσιμα tasks - εργασίες για έναν χρήστη.*
URL	/api/task
Method	*GET*
Data Params	

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	<pre>{
	
	"info": [String],
	
	"data": {
	
	"completed": [Boolean],
	
	"subject": [String],
	
	"user": [String-id of user reference],
	
	"_id": [String]
	
	}
	
	}</pre>

Error Response	

FIND A TASK BY ID

Title	*FIND A TASK BY ID* *Παρουσιάζεται η δυνατότητα αναζήτησης μιας εργασίας από έναν χρήστη.*
URL	/api/task/:id
Method	*GET*
Data Params	
Success Response	**Success:** Successful request

	Status Code: 200 OK
	Content:
	{
	"info": [String],
	"data": {
	"completed": [Boolean],
	"subject": [String],
	"user": [String-id of user reference],
	"_id": [String]
	}
	}

Error Response	**Status Code: 400 Bad Request**
	
	Content:
	{
	"info": "Error during find task",
	"error": [Error]
	}

UPDATE A GIVEN TASK

Title	***UPDATE A GIVEN TASK*** ***Παρουσιάζεται η δυνατότητα τροποποίησης μιας εργασίας από έναν χρήστη. Ο χρήστης μπορεί να μετονομάσει την εργασία είτε να αλλάξει το status completed.***
URL	`/api/task/:id`
Method	***PUT***

Data Params	{
	"completed": [Boolean],
	"subject": [String]
	}

Success Response	**Success:** Successful request

	Status Code: 200 OK
	Content:
	{
	"info": [String]
	}

Error Response	**Status Code: 400 Bad Request**
	
	Content:
	{
	"info": "Error during task update",
	"error": [Error]
	}
	
	Content:
	{
	"info": "task not found"
	}

DELETE A GIVEN TASK

Title	***DELETE A GIVEN TASK*** ***Παρουσιάζεται η δυνατότητα διαγραφής μιας εργασίας από έναν χρήστη.***
URL	`/api/task/:id`
Method	***DELETE***
Data Params	

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	"info": [String]
	}

Error Response	**Status Code: 400 Bad Request**

	Content:
	{
	"info": "Error during remove task",
	"error": [Error]
	}
	
	Content:
	{
	"info": "task not found"
	}

CREATE A NEW TRACK BY

Title	***CREATE A NEW TRACK BY*** ***Εδώ δημιουργείται μια καινούργια καταχώρηση λέξης, η οποία θα χρησιμοποιηθεί για να κάνει το Streaming API του Twitter να μας φέρει σχετικά αποτελέσματα.***
URL	`/api/trackBy`
Method	***POST***

Data Params	{
	"keyword": [String]
	}

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	"info": [String],
	"data": [TrackBy Object]: {
	"keyword": [String],
	"_id": [String]

	} }

Error Response	**Keyword already exists**
	
	Status Code: 400 Bad Request
	
	Content:
	{
	
	"info": "Keyword already exists"
	
	}
	
	Any other error
	
	Status Code: 400 Bad Request
	
	Content:
	{
	
	"error": "[Error]"
	
	}

VIEW/READ ALL TRACKS

Title	***VIEW/READ ALL TRACKS*** ***Προβολή όλων των keywords που έχουν καταχωρηθεί από έναν admin του συστήματος.***
URL	**/api/trackBy**
Method	***GET***
Data Params	

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	
	"info": [String],
	
	"data": [TrackBy Object]: {

	<pre>"keyword": [String],
	"_id": [String]
	}
	}</pre>

Error Response	**Status Code: 400 Bad Request**
	
	Content:
	{
	<pre>"info": "Error during find tasks"</pre>
	}

DELETE A GIVEN TRACK

Title	***DELETE A GIVEN TRACK*** ***Διαγραφή ενός συγκεκριμένου track – keyword από το σύστημα.***
URL	`/api/trackBy/:id`
Method	***DELETE***
Data Params	

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	<pre>"info": [String]</pre>
	}

Error Response	**Status Code: 400 Bad Request**
	
	Content:
	{
	<pre>"info": "Error during track deletion"</pre>
	}
	
	Cannot Find Task
	Status Code: 404 NOT FOUND

	Content:
	{
	
	"info": "Cannot find track"
	
	}

GET ALL SCHEDULED TWEETS

Title	***GET ALL SCHEDULED TWEETS*** ***Προβολή όλων των προγραμματισμένων tweets του συστήματος.***
URL	**/api/scheduled-tweet**
Method	***GET***
Data Params	

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	
	"info": [String],
	
	"data": [Array Scheduled Tweet]: {
	
	"completed": [Boolean],
	
	"datetime": [DateTime],
	
	"text": [String]
	
	}
	
	}

Error Response	**Status Code: 400 Bad Request**
	
	Content:
	{
	
	"info": "Error during find tweets"
	
	}

SCHEDULE A NEW TWEET

Title	***SCHEDULE A NEW TWEET*** ***Δημιουργία ενός προγραμματισμένου tweet.***
URL	`/api/scheduled-tweet`
Method	***POST***

Data Params	{
	"text": [String],
	"datetime": [DateTime] - "2017-10-23T16:00:00.000Z"
	}

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	"info": [String],
	"data": [Scheduled Tweet]: {
	"completed": [Boolean],
	"datetime": [DateTime],
	"text": [String]
	}
	}

Error Response	**Status Code: 400 Bad Request**
	
	Content:
	{
	"info": "Error during tweet create"
	}

POST A TWEET

Title	***POST A TWEET*** ***Δημοσίευση ενός νέου tweet στο Twitter, μέσα από την εφαρμογή.***
URL	`/api/tweet`

Method	***POST***

Data Params	{
	"status": [String]
	}

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	[Tweet Object]
	}

Error Response	

SHOW USER

Title	***SHOW USER*** ***Προβολή ενός χρήστη εγγεγραμμένου στο Twitter. Η αναζήτηση του χρήστη γίνεται είτε μέσω του screen_name του χρήστη είτε μέσω του user_id του χρήστη.***
URL	`/api/users/show`
Method	***POST***

Data Params	{
	"screen_name": [String],
	"user_id": [String]
	}

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	[Twitter User Object]
	}

Error Response	

Analytics API

Overview

Title	***Overview*** ***Παρουσιάζονται αποτελέσματα που αφορούν πόσα καταχωρημένα tweets υπάρχουν ανά hashtag.***
URL	`/api/analytics/overview`
Method	***GET***
Data Params	

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	"data": [
	{
	"count": [Number],
	"keyword": [String]
	}
	]
	}

Error Response	

HASHTAG ANALYTICS

Title	***HASHTAG ANALYTICS*** ***Ο χρήστης μπορεί να πάρει τιμές που αφορούν, πόσα tweets δημοσιεύονται για μια συγκεκριμένη χρονική περίοδο. Η αναζήτηση μπορεί να γίνει για τις ακόλουθες τιμές περιόδου: "24h", "daily", "monthly", "yearly" και το hashtag που θέλει ο χρήστης.***
URL	`/api/analytics/tweet`
Method	***GET***
Data Params	{ "period": [String], "tweet": [String]

	}

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	
	"data": [
	
	{
	
	"count": [Number],
	
	"_id": {
	
	"dayOfMonth": [Number],
	
	"hour": [Number],
	
	"month": [Number],
	
	"year": [Number]
	
	}
	
	}
	
	]
	
	}

Error Response	

CLIENT ANALYTICS

Title	***CLIENT ANALYTICS*** ***Ο χρήστης μπορεί να πάρει τιμές που αφορούν, πόσα tweets δημοσιεύονται ανά client που χρησιμοποιείται για να γίνει μια δημοσίευση. Η δημοσίευση ενός tweet μπορεί να γίνει με χρήση πολλών και διαφορετικών clients. Την πληροφορία αυτή μπορούμε να την πάρουμε από το Twitter Object, χρησιμοποιώντας το propery source: π.χ "Twitter for Android", "Twitter for iPhone", "Twitter Web Client" είναι ορισμένοι απο τους συνηθέστερους clients που μπορεί να χρησιμοποιήσει ένας χρήστης.***
URL	`/api/analytics/client`
Method	***GET***
Data Params	

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	"data": [
	{
	"count": [Number],
	"_id": [String] "client used"
	}
	]
	}

Error Response	

CLIENT ANALYTICS PER HASHTAG

Title	***CLIENT ANALYTICS PER HASHTAG*** ***Ο χρήστης μπορεί να πάρει τιμές που αφορούν, πόσα tweets δημοσιεύονται ανά client που χρησιμοποιείται για να γίνει μια δημοσίευση. Η δημοσίευση ενός tweet μπορεί να γίνει με χρήση πολλών και διαφορετικών clients. Την πληροφορία αυτή μπορούμε να την πάρουμε από το Twitter Object, χρησιμοποιώντας το propery source: π.χ "Twitter for Android", "Twitter for iPhone", "Twitter Web Client" είναι ορισμένοι απο τους συνηθέστερους clients που μπορεί να χρησιμοποιήσει ένας χρήστης. Με αυτό το endpoint μπορούμε να πάρουμε την πληροφορία που αφορά, πόσο είναι το ποσοστό χρήσης της κάθε συσκευής για ένα συγκεκριμένο hashtag.***
URL	`/api/analytics/client`
Method	***POST***

Data Params	{
	"hashtag": [String]
	}

Success Response	**Success:** Successful request

	Status Code: 200 OK
	Content:
	{
	"data": [
	{
	"count": [Number],
	"_id": [String] "client used"
	}
	]
	}

Error Response	

STREAM STATUS

Title	***STREAM STATUS*** ***Προβολή της κατάστασης που βρίσκεται το service που έχει κατασκευαστεί, το οποίο καταναλώνει το Streaming API του Twitter.***
URL	`/api/stream-status`
Method	*GET*
Data Params	

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	"status": [Boolean]
	}

Error Response	

GET STORED TWEETS

Title	***GET STORED TWEETS*** ***Προβολή των 10 πρώτων καταχωρημένων tweet. Μαζί με την πληροφορία των tweets, επιστρέφεται και ο αριθμός των συνολικών tweets καθώς και το σύνολο των σελίδων που χρειάζεται για να δείξει η εφαρμογή τα αποτελέσματα.***
URL	`/api/tweets`
Method	***GET***
Data Params	

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	
	"pages": [Number],
	
	"total": [Number],
	
	"data": [{Twitter Object}]
	
	}

Error Response	

GET STORED TWEETS PAGED

Title	***GET STORED TWEETS PAGED*** ***Προβολή των 10 επόμενων καταχωρημένων tweet. Μαζί με την πληροφορία των tweets, επιστρέφεται και ο αριθμός των συνολικών tweets καθώς και το σύνολο των σελίδων που χρειάζεται για να δείξει η εφαρμογή τα αποτελέσματα.***
URL	`/api/tweets`
Method	***POST***
Data Params	{ "page": [Number], "skip": [Number], ο αριθμός των καινούριων αποτελεσμάτων που έχουν έρθει με το Streaming API.

	}

Success Response	**Success:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	
	"total": [Number],
	
	"data": [{Twitter Object}]
	
	}

Error Response	

Admin API

Παρακάτω παρατίθενται τα endpoints που έχει πρόσβαση μόνο ένας χρήστης με ρόλο "admin".

Είναι το user management, δηλαδή η προβολή όλων των χρηστών της εφαρμογής, η δυνατότητα τροποποίησης και διαγραφής.

Έπειτα είναι η ενεργοποίηση ή το σταμάτημα της καταγραφής μέσω του Streaming API (twitter).

View Users

Title	***View users*** ***Προβολή όλων των εγγεγραμμένων χρηστών της εφαρμογής.***
URL	`/api/users`
Method	***GET***
Data Params	

Success Response	**Success1:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	
	"info": "Users retrieved successfully",
	
	"data": [User Object]: {
	
	"email": "test@test.com",

	<pre>"role": "user",
	"twitter": "if exists",
	"picture": [String],
	"tasks": [],
	"_id": "59ed9c526cd53d982397a6fe"
	},
	}</pre>

Error Response	**Error:** Error while retrieving users
	
	Status Code: 400
	
	Content:
	<pre>{
	"info": "Error while retrieving users",
	"error": [Error]
	}</pre>

Edit user

Title	***Edit user*** ***Δυνατότητα τροποποίησης ενός χρήστη.***
URL	**/api/user**
Method	***PUT***

Data Params	<pre>{
	"user": [User object]
	}</pre>

Success Response	**Success1:** Successful login

	Status Code: 200 OK
	Content:
	{
	"info": "User updated.",
	"data": [User Object]
	}

Error Response	**Error:** Error while updating user
	
	Status Code: 400
	
	Content:
	{
	"info": "Error while updating user.",
	"error": [Error]
	}

Delete user

Title	***Delete user*** ***Δυνατότητα διαγραφής ενός χρήστη.***
URL	`/api/user/:id`
Method	***DELETE***
Data Params	

Success Response	**Success1:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	"info": "User deleted successfully.",
	"data": [User Object]
	}

Error Response	**Error:** Error while deleting user

	Status Code: 400
	
	Content:
	{
	
	"info": "Error while deleting user.",
	
	"error": [Error]
	
	}

Start/Stop stream

Title	***Start/Stop stream*** ***Δυνατότητα κλεισίματος ή εκκίνησης προγραμματιστικά του Streaming API.***
URL	`/api/stream`
Method	***POST***

Data Params	{
	
	"status": [Boolean]
	
	}

Success Response	**Success1:** Successful request
	
	Status Code: 200 OK
	Content:
	{
	
	"info": "Status changed",
	
	"status": [Boolean]
	
	}

Error Response	

8. ΣΥΜΠΕΡΑΣΜΑΤΑ

Η εφαρμογή βοηθάει τους χρήστες να χειρίζονται μαζικά έναν λογαριασμό στο κοινωνικό δίκτυο του Twitter, παραδείγματος χάρη μιας εταιρείας. Μπορούν εύκολα να εντοπίζουν χρήσιμο υλικό και να προγραμματίζουν δημοσιεύσεις καθώς και να παρακολουθούν την κίνηση που δημιουργούν hashtags που δημιουργούν. Μελλοντικά βήματα θα μπορούσε να γίνουν έτσι ώστε να ενσωματωθούν περισσότερες λειτουργίες από το Twitter (REST API). Επιπλέον η εφαρμογή αυτή θα μπορούσε να παρέχεται as a service σε εταιρείες ή άτομα που θέλουν να τη χρησιμοποιήσουν για το σκοπό αυτό, ακολουθώντας τις οδηγίες για τη δημιουργία μια εφαρμογής στο Twitter και έπειτα βάζοντας τα απαραίτητα κλειδιά στο σύστημα. Τέλος, θα μπορούσε να υλοποιηθεί και αντίστοιχη εφαρμογή για κινητές συσκευές, χωρίς καν να χρησιμοποιηθεί native γλώσσα για την ανάπτυξη της, παρά χρησιμοποιώντας ένα framework για υλοποίηση υβριδικών εφαρμογών σε κινητά, π.χ Ionic Framework.

9. ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΠΗΓΕΣ

1. https://developer.twitter.com/en/docs
2. https://expressjs.com/en/4x/api.html
3. https://docs.mongodb.com/
4. http://mongoosejs.com/docs/guide.html
5. https://docs.angularjs.org/api
6. https://en.wikipedia.org/wiki