Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Υλοποίηση web εφαρμογής, για την επισήμανση δημοφιλών θεμάτων σε μέσα κοινωνικής δικτύωσης βάσει τοποθεσίας»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Υλοποίηση web εφαρμογής, για την επισήμανση δημοφιλών θεμάτων σε μέσα κοινωνικής δικτύωσης βάσει τοποθεσίας.**
Title	**Implementation of Web Application about Signalization of the most popular topics in Social Media according Location.**
Ονοματεπώνυμο Φοιτητή	**Χατζηγεωργίου Ηλίας**
Πατρώνυμο	**Παναγιώτης**
Αριθμός Μητρώου	**ΜΠΣΠ 14095**
Επιβλέπων	**Δουληγέρης Χρήστος, Καθηγητής**

Ημερομηνία Παράδοσης: 21/09/2017

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Χρήστος Δουληγέρης Καθηγητής	Μιχάλης Ψαράκης Επίκουρος Καθηγητής	Παναγιώτης Κοτζανικολάου Επίκουρος Καθηγητής

Αφιέρωση

Αφιερώνω την συγκεκριμένη εργασία στην Οικογένεια μου.

Περιεχόμενα

Abstract

The potential user of the application, has the opportunity to see real-time highlighting of popular themes within social networks as they were presented on the basis of the criteria he set himself. Such criteria may be location, time of day, type of subject, etc.

Ευχαριστίες

Πρώτα και πριν από όλους θα ήθελα να ευχαριστήσω θερμά τον κ. Δουληγέρη Χρήστο που όντας καθηγητής μου στα πλαίσια των μεταπτυχιακών μου σπουδών, με βοήθησε στην απόφαση λήψης της συγκεκριμένης διπλωματικής εργασίας. Ως επιβλέπων καθηγητής δήλωσε παρών στις όποιες δυσκολίες αντιμετώπισα κατά τη διάρκεια εκπόνησης της εργασίας. Επίσης θα ήθελα να ευχαριστήσω τον κ. Μαλατρά Απόστολο, επιβλέποντα καθηγητή.

Για το τέλος έχω αφήσει τους φίλους μου που πραγματικά δεν ξέρω με ποια λόγια να τους ευχαριστήσω. Με στήριξαν ψυχολογικά, κατά την δύσκολη χρονική περίοδο εκπόνησης της παρούσας μελέτης. Χωρίς την βοήθεια τους δεν θα τα είχα καταφέρει. Πέτρο, Άννα, Θανάση, Άρη, Βαγγέλη, Αλίκη, Στέφη, Γεωργία, Ηλία, Κώστα, Παντελή σας ευχαριστώ…

Την εργασία αυτή την αφιερώνω στον Πατέρα μου, Παναγιώτη, τη Μητέρα μου, Χρυσάνθη και τον αδερφό μου Γιάννη.

Κεφάλαιο 1ο

Εισαγωγή

Στα πλαίσια της διπλωματικής εργασίας με τίτλο: «Υλοποίηση web εφαρμογής, για την επισήμανση δημοφιλών θεμάτων σε μέσα κοινωνικής δικτύωσης βάσει τοποθεσίας», θα πραγματοποιηθεί ανάλυση παρόμοιων εφαρμογών που εδρεύουν στο διαδίκτυο, περιγραφή της αρχιτεκτονικής και των υπηρεσιών τους, καθώς και της λίστας με τα πλεονεκτήματα ή μειονεκτήματα που ενδέχεται να συγκεντρώνουν.

Πριν όμως παραθέσουμε τις εφαρμογές αυτές, είναι επιτακτική ανάγκη να δώσουμε έναν ορισμό, καθώς και μια σύντομη περιγραφή της φύσης αυτών των web εφαρμογών. Καθημερινά, εκατομμύρια χρήστες κοινωνικών δικτύων ανά τον κόσμο, ξοδεύουν αρκετές ώρες παραθέτοντας τους προβληματισμούς τους, τη γνώμη τους, ενημερώνονται για τις τελευταίες εξελίξεις που λαμβάνουν χώρα κτλ. Πρόκειται για τεράστιες ποσότητες ανεπεξέργαστων πληροφοριών, οι οποίες ωστόσο, με την κατάλληλη επεξεργασία και ειδική τροποποίηση, εξάγουν εκπληκτικά αποτελέσματα και οδηγούν σε χρήσιμα συμπεράσματα. Συμπεράσματα κάθε λογής, όπως για παράδειγμα, ανάλυση πληθυσμού, πολιτικών ή θρησκευτικών πεποιθήσεων, ανάλυση συναισθήματος κτλ. Η παραπάνω εξόρυξη γνώσης, χαρακτηρίζεται ως «θησαυρός» για διάφορες εταιρίες, οργανισμούς, ακόμη και κυβερνήσεις κρατών.

Ο σκοπός λοιπόν των εφαρμογών αυτών, είναι να δέχονται τα δεδομένα ακατέργαστα ως εισροή, να τα επεξεργάζονται, να τα φιλτράρουν και να τα ομαδοποιούν, δίνοντας ως εκροή από το Application , την οπτικοποίηση των αποτελεσμάτων πάνω σε χάρτη. Με τον τρόπο αυτό, ο δυνητικός χρήστης της εφαρμογής, έχει την ευκαιρία να δει σε πραγματικό χρόνο, την επισήμανση δημοφιλών θεμάτων μέσα στα κοινωνικά δίκτυα, όπως αυτά παρουσιάστηκαν με βάση τα κριτήρια που έθεσε ο ίδιος. Τέτοια κριτήρια μπορεί να είναι η τοποθεσία, το χρονικό διάστημα της ημέρας, το είδος του θέματος κτλ.

Κεφάλαιο 2ο

Παρόμοιες Εφαρμογές

Μετά από έρευνα που έγινε στο διαδίκτυο, συγκεντρώθηκαν οι παρακάτω εφαρμογές και παρουσιάζονται με τους αντίστοιχους τίτλους τους. Στην συνέχεια θα πραγματοποιηθεί λεπτομερής ανάλυση κάθε εφαρμογής. Η περιγραφή της αρχιτεκτονικής έγινε με την συνδρομή του **Built With App**.

- **Trendsmap**
- **Tweography**
- **TwitterMap.tv**
- **One Million TweetMap**
- **A World Of Tweets**

Trendsmap

Το **Trendsmap**, είναι μια web εφαρμογή, η οποία παρουσιάζει τις τελευταίες τάσεις από το κοινωνικό δίκτυο **Twitter**, για οπουδήποτε πάνω στον κόσμο. Πρόκειται για μια ολοκληρωμένη πρόταση οπτικοποίησης με χρήση χάρτη. Παρέχει κλιμακωτές υπηρεσίες εγγραφής τριών κατηγοριών: Free, Basic, Plus, φαίνεται στον παρακάτω πίνακα:

	Free	**Basic 9$/month 99$/year**	**Plus 19$/month 190$/year**
Latest Global and Local Trends	✓	✓	5x More Trends
Ad Free	✗	✓	✓
Top Users, Videos, Images and Links	✗	✓	✓
Detailed Zoom	✗	✗	4 More Levels
7 Day History	✗	✗	✓
Filter By Words, Users and Hashtags	✗	✗	✓
Top videos, Images and Links For Each Local Trend	✗	✗	✓
Enhanced UI Functionality	✗	✗	✓

Αρχιτεκτονική Συστήματος:

Web Server: nginx
SSL Certificate: Trustwave SSL, Comodo SSL, Comodo Positive SSL, Comodo Positive SSL Wildcard
Nameserver Provider: DNS Made Easy DNS
Email Services: Google Apps For Business, Amazon SES, Fastmail, SPF, Mandrill
Advertising: Google Remarking, DoubleClick.Net, Google Publisher Tag
Analytics and Tracking: Google Analytics with Ad Tracking, Google Conversion Tracking, Google Analytics, Google Analytics Classic
JavaScript Libraries: Google API, Twitter Platform, jQuery, D3JS, Google Hosted Libraries, Google Hosted jQuery
Mapping: Leaflet
Content Delivery Network: AJAX Libraries API, Bootstrap CDN
Widgets: Google Plus One Platform, Rickshaw, Vine Embed, Font Awesome
CSS Media Queries: Min Width

Tweography

Το **Tweography**, παρουσιάζει τα πιο πρόσφατα **Tweets** βάσει τοποθεσίας, απεικονίζοντας τα στο χάρτη της Google. Απαραίτητη προϋπόθεση για πλοήγηση στην εφαρμογή, είναι η ύπαρξη ή η δημιουργία λογαριασμού στο κοινωνικό δίκτυο **Twitter**. Αυτό ίσως αποτρέψει ένα χρήστη ο οποίος επιθυμεί μια απλή περιήγηση στην εφαρμογή και όχι την περαιτέρω διαδικασία. Επίσης, εφόσον υφίσταται ο λογαριασμός στο **Twitter**, απαιτείται είσοδος στην εφαρμογή με τα στοιχεία του λογαριασμού αυτού(**Sign In**).

Αρχιτεκτονική Συστήματος:

Web Server: Apache
Hosting Providers: Wisconsin CyberLynk Network
SSL Certificate: GeoTrust SSL
JavaScript Libraries: jQuery

TwitterMap.tv

Το **TwitterMap.tv**, είναι μια εφαρμογή εφάμιλλη των παραπάνω, ωστόσο βρίσκεται υπό καθεστώς πώλησης από τον ιδιοκτήτη της. Για το λόγο αυτό, δεν παρουσιάζει κάποιο ιδιαίτερο ενδιαφέρον, αφήνοντας να εννοηθεί ότι η ανάπτυξή της εξαρτάται αποκλειστικά και μόνο από τον μελλοντικό αγοραστή του **Domain**.

Αρχιτεκτονική Συστήματος:

Web Server: nginx
Advertising: Adblock Acceptable Ads, Google AdSense, Google Adsense For Domains
Analytics and Tracking: Google Analytics, Google Analytics Classic, Google Analytics Anonymize ID
JavaScript Libraries: JSON 3
Content Delivery Network: CloudFront
Widgets: Google Font API
Document Information: XHTML Transitional, Javascript, CSS
Encoding: UTF-8
Park Domain Providers: ParkingCrew

One Million TweetMap

Το **web app One Million TweetMap**, χαρτογραφεί τις τελευταίες αναρτήσεις (**Tweets**) του κοινωνικού δικτύου Twitter, τα οποία δίνονται ως εκροή από το **Twitter Stream API**. Τα χαρτογραφημένα δεδομένα ανανεώνονται **σε πραγματικό χρόνο**. Φιλοσοφία της εφαρμογής είναι να αποθηκεύονται και να οπτικοποιούνται οι τελευταίες ένα εκατομμύριο αναρτήσεις από όλο τον κόσμο. Κάθε δευτερόλεπτο, περίπου πενήντα **Tweets** προστίθενται. Αυτό συμβαίνει διότι υπάρχει περιορισμός από το **Twitter Stream**.

Αρχιτεκτονική Συστήματος:

Web Server: Rack Cache, nginx, nginx 1.1
Nameserver Providers: OVH DNS
Hosting Providers: Hetzner
Email Services: OVH Mail
Analytics and Tracking: Google Analytics, Google Analytics Classic
JavaScript Libraries: jQuery, jQuery 1.8.2, Twitter Platform, Modernizr yepnope, Moment JS, HTMLShiv, Google Hosted Libraries, Google Hosted jQuery
Content Delivery Network: GStatic Google Static Content, AJAX Libraries API
Widgets: Twitter Tweet Button
Mapping: Google Maps, Google Maps API
Mobile: Mobile Non Scaleable Content (Apple), Viewport Meta
CSS Media Queries: Device Width, Orientation, Min Width, Device Height, Max Width

A World Of Tweets

Η εφαρμογή A World Of Tweets, συνδυάζει τα bits πληροφοριών με το γεωγραφικό σημείο από το οποίο προέρχονται. Εκθέτει, την τοποθεσία των ανθρώπων που πραγματοποιούν tweeting την τελευταία ώρα. Όσο περισσότερα tweets υπάρχουν από μια συγκεκριμένη περιοχή, τόσο «θερμότερη» ή ερυθρότερη γίνεται.

Αρχιτεκτονική Συστήματος:

Web Server: Apache
Analytics and Tracking: Google Analytics, Google Analytics Classic
JavaScript Libraries: Google Hosted Libraries, Google Hosted jQuery, jQuery, jQuery 1.4.4, SWFObject, Twitter Platform
Content Delivery Network: AJAX Libraries API
Widgets: Facebook Like Button, Twitter Tweet Button, Pinterest
Document Information: HTML5 DocType, Meta Description, CSS, Conditional Comments, Javascript, Iframe, HTML5 Specific Tags
Encoding: UTF-8

Κεφάλαιο 3ο

Τεχνολογίες

Στο παρόν κεφάλαιο γίνεται αναφορά στις τεχνολογίες που επιλέχθηκαν και χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής ***GreekTrend***. Με τον όρο τεχνολογίες εννοούνται όλα εκείνα τα εργαλεία/υπηρεσίες client-side ή server-side τα οποία παρέχονται από τον παγκόσμιο ιστό ή είναι συμβατά με αυτόν και καθιστούν εφικτή την αποτελεσματική σχεδίαση & ανάπτυξη διαδικτυακών εφαρμογών με χαρακτηριστικά την διαδραστικότητα και τη χρηστικότητα.

PHP

Η PHP (***PHP: Hypertext Preprocessor***) είναι μια γλώσσα προγραμματισμού για τη δημιουργία σελίδων web με δυναμικό περιεχόμενο. Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό ***server*** του Παγκόσμιου Ιστού (π.χ. Apache), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, που είτε θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα ***HTML*** ή θα επεξεργασθεί τις εισόδους δίχως να προβάλλει την έξοδο στο χρήστη, αλλά θα τις μεταβιβάσει σε κάποιο άλλο ***PHP script***.

Αποτελεί μια από τις πιο διαδεδομένες τεχνολογίες στο Παγκόσμιο Ιστό, καθώς χρησιμοποιείται από πληθώρα εφαρμογών και ιστότοπων. Η ευρύτητα στη χρήση της είναι απόρροια της ευκολίας που παρουσιάζει ο προγραμματισμός με αυτή αλλά και στο γεγονός πως είναι μια γλώσσα η οποία βρίσκεται σχεδόν σε κάθε διακομιστή. Διάσημες εφαρμογές που κάνουν εκτενή χρήση της ***PHP*** είναι το γνωστό Σύστημα Διαχείρισης Περιεχομένου (***Content Management System, WordPress και το Drupal***). Ένα αρχείο με κώδικα PHP θα πρέπει να έχει την κατάλληλη επέκταση (***π.χ. *.php, *.php4, *.phtml κ.ά.***). Η ενσωμάτωση κώδικα σε ένα αρχείο επέκτασης ***.html*** δεν θα λειτουργήσει και θα εμφανίσει στον ***browser*** τον κώδικα χωρίς καμία επεξεργασία, εκτός αν έχει γίνει η κατάλληλη ρύθμιση στα ***MIME types*** του ***server***. Επίσης ακόμη κι όταν ένα αρχείο έχει την επέκταση ***.php***, θα πρέπει ο *server* να είναι ρυθμισμένος για να επεξεργάζεται και να μεταγλωττίζει τον κώδικα ***PHP*** σε ***HTML*** που καταλαβαίνει το πρόγραμμα πελάτη. Ο διακομιστής ***Apache***, που χρησιμοποιείται σήμερα από τα λειτουργικά συστήματα ***GNU/Linux, Microsoft Windows, Mac OS X*** υποστηρίζει εξ ορισμού την εκτέλεση κώδικα ***PHP***, είτε με την χρήση ενός πρόσθετου (mod_php) ή με την αποστολή του κώδικα προς εκτέλεση σε εξωτερική διεργασία ***CGI*** ή ***FCGI*** ή με την έλευση της ***php5.4*** , όπου υποστηρίζεται η εκτέλεση σε πολυάσχολους ιστοχώρους, ***FastCGI Process Manager (FPM)***.

Javascript

Η JavaScript (***JS***) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά, αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (***client-side scripts***) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται. Είναι μια γλώσσα σεναρίων, που βασίζεται στα πρωτότυπα (***prototype-based***), είναι

δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη ***C***. Η ***JavaScript*** αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη ***Java***, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της ***JavaScript*** προέρχονται από τις γλώσσες προγραμματισμού ***Self*** και ***Scheme***. Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (***multi-paradigm***), υποστηρίζοντας αντικειμενοστρεφές, προστακτικό και συναρτησιακό στυλ προγραμματισμού.

Η J***avaScript***, τέλος, χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων — τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (***site-specific browsers***) και οι μικρές εφαρμογές της επιφάνειας εργασίας (***desktop widgets***). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για ***JavaScript*** (όπως το ***Node.js***) έχουν επίσης κάνει τη συγκεκριμένη γλώσσα πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού από την πλευρά του διακομιστή (***server-side***).

phpMyAdmin

Το **phpMyAdmin** είναι ένα δωρεάν εργαλείο διαχείρισης για βάσεις δεδομένων, όπως είναι η ευρέως γνωστή στο διαδίκτυο ***MySQL*** και ***MariaDB***. Ως φορητή εφαρμογή ***web,*** όπως λογίζεται και γράφεται κυρίως στην PHP, έχει γίνει ένα από τα πιο δημοφιλή εργαλεία διαχείρισης της MySQL, ειδικά για υπηρεσίες ***web hosting***.

Τα χαρακτηριστικά που παρέχει το πρόγραμμα περιλαμβάνουν:

- Web Interface
- Διαχείριση βάσεων δεδομένων ***MySQL*** και ***MariaDB***.
- Εισαγωγή δεδομένων από ***CSV*** και ***SQL***.
- Εξαγωγή δεδομένων σε διάφορες μορφές, όπως: ***CSV***, ***SQL***, ***XML***, ***PDF*** (μέσω της βιβλιοθήκης ***TCPDF***), ***ISO/IEC 26300*** - ***OpenDocument Text and Spreadsheet***, ***Word, Excel, LaTeX***.
- Διαχείριση πολλαπλών διακομιστών.
- Δημιουργία ***PDF graphics*** των layout των ΒΔ.
- Δημιουργία περίπλοκων ερωτημάτων χρησιμοποιώντας ***Query-by-Example (QBE)***.
- Αναζήτηση σε παγκόσμιο επίπεδο σε μια βάση δεδομένων ή σε ένα υποσύνολο της.
- Μετασχηματισμός αποθηκευμένων δεδομένων σε οποιαδήποτε μορφή χρησιμοποιώντας ένα σύνολο προκαθορισμένων λειτουργιών, όπως προβολή δεδομένων ***BLOB*** ως εικόνας ή συνδέσμου λήψης.
- Ζωντανά διαγράμματα για την παρακολούθηση της δραστηριότητας διακομιστή ***MySQL***, όπως συνδέσεις, διαδικασίες, χρήση CPU / μνήμης κ.λπ.
- Δυνατότητα εργασίας με διαφορετικά λειτουργικά συστήματα που κυκλοφορούν ανά τον κόσμο.

CSS

Η ***CSS*** (***Cascading Style Sheets-Διαδοχικά Φύλλα Στυλ***) ή (***αλληλουχία φύλλων στυλ***) είναι μια γλώσσα υπολογιστή που ανήκει στην κατηγορία των γλωσσών φύλλων στυλ που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης. Χρησιμοποιείται λοιπόν για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις

γλώσσες ***HTML*** και ***XHTML***, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστότοπου.

Η ***CSS*** είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα, δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και να δίνει περισσότερες δυνατότητες σε σχέση με την ***html***. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της ***CSS*** κρίνεται ως απαραίτητη.

Bootstrap

Το ***Bootstrap*** είναι μια συλλογή εργαλείων ανοιχτού κώδικα (Ελεύθερο λογισμικό) για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει ***HTML*** και ***CSS*** για τις μορφές τυπογραφίας, κουμπιά πλοήγησης και άλλα στοιχεία του περιβάλλοντος, καθώς και προαιρετικές επεκτάσεις ***JavaScript***. Είναι το πιο δημοφιλές πρόγραμμα στο ***GitHub*** και έχει χρησιμοποιηθεί μεταξύ άλλων, από τη ***NASA*** και το ***MSNBC***.

Έχει σχετικά ελλιπή υποστήριξη για ***HTML5*** και ***CSS***, αλλά είναι συμβατό με όλους τους φυλλομετρητές (***browsers***). Βασικές πληροφορίες συμβατότητας των ιστοσελίδων ή εφαρμογών είναι διαθέσιμες για όλες τις συσκευές και τα προγράμματα περιήγησης. Υπάρχει μια έννοια της μερικής συμβατότητας που κάνει τα βασικά στοιχεία μιας ιστοσελίδας που διατίθενται για όλες τις συσκευές και τα προγράμματα περιήγησης. Για παράδειγμα, οι ιδιότητες πάνω στις οποίες θεσπίστηκε το ***CSS3*** για στρογγυλεμένες γωνίες, κλίσεις και σκιές, χρησιμοποιούνται από το ***Bootstrap*** παρά την έλλειψη υποστήριξης από μεγάλα προγράμματα περιήγησης στο ***Web***. Αυτά επεκτείνουν τη λειτουργικότητα του πακέτου εργαλείων, αλλά δεν απαιτούνται για τη χρήση του.

Από την έκδοση 2.0 υποστηρίζεται επίσης ανταποκρίσιμος σχεδιασμός (***responsive design***). Αυτό σημαίνει ότι η διάταξη των ιστοσελίδων προσαρμόζεται δυναμικά, λαμβάνοντας υπόψη τα χαρακτηριστικά της συσκευής που χρησιμοποιείται (***PC, tablet, κινητό τηλέφωνο***).

Το ***Bootstrap*** είναι ανοικτού κώδικα και είναι διαθέσιμο στο ***GitHub***. Οι προγραμματιστές ενθαρρύνονται να συμμετέχουν στο έργο και να κάνουν τη δική τους συνεισφορά στην πλατφόρμα.

Από πλευράς δομής και λειτουργίας, είναι σπονδυλωτό και αποτελείται ουσιαστικά από μια σειρά ***stylesheets*** που εφαρμόζουν τα διάφορα συστατικά του πακέτου εργαλείων. Ένα στυλ που ονομάζεται ***bootstrap.less*** περιλαμβάνει τα συστατικά ***stylesheets***. Οι προγραμματιστές μπορούν να προσαρμόσουν το αρχείο ***Bootstrap***, επιλέγοντας τα στοιχεία που θέλουν να χρησιμοποιήσουν στο έργο τους.

Προσαρμογές είναι δυνατές σε περιορισμένη έκταση μέσω ενός κεντρικού στυλ διαμόρφωσης. Η χρήση γλώσσας στυλ επιτρέπει τη χρήση για μεταβλητές, λειτουργίες και φορείς (operators), ένθετους επιλογείς, γνωστά και ως μείγματα ***mixin***.

Τέλος, από την έκδοση 2.0, η διαμόρφωση του ***Bootstrap*** περιέχει επιπλέον μία ειδική επιλογή "Προσαρμογή" στην τεκμηρίωση (***documentation***). Επιπλέον, ο σχεδιαστής του έργου επιλέγει σε μια φόρμα τα επιθυμητά συστατικά και τα προσαρμόζει, εάν είναι αναγκαίο, σε τιμές διαφόρων εναλλακτικών λύσεων για τις ανάγκες του. Στη συνέχεια δημιουργείται ένα πακέτο που περιλαμβάνει ήδη το προ-χτισμένο ***CSS*** στυλ.

HTML

Η ***HTML*** (αρχικοποίηση του αγγλικού ***HyperText Markup Language***, ελλ. Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα σήμανσης και τα στοιχεία της αποτελούν θεμέλιο για όλες των ιστοσελίδες.

Η ***HTML*** γράφεται υπό μορφή στοιχείων, τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες τέτοιου τύπου συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

Ο σκοπός ενός ***web browser*** είναι να διαβάζει τα έγγραφα ***HTML*** και να τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο ***browser*** δεν εμφανίζει τις ετικέτες ***HTML***, αλλά τις χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello world</p>
  </body>
</html>
```

Εικόνα 1. Το πρόγραμμα Hello World, σε γλώσσα προγραμματισμού HTML

Τα στοιχεία της ***HTML*** χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η ***HTML*** επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η ***JavaScript***, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων ***HTML***.

Οι ***Web browsers*** μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης ***CSS*** για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. Ο οργανισμός ***W3C***, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την ***HTML*** και τα ***CSS***, ενθαρρύνει τη χρήση των δεύτερων αντί διαφόρων στοιχείων της ***HTML*** για σκοπούς παρουσίασης του περιεχομένου.

Κεφάλαιο 4ο

4.1 Μεταφόρτωση των αρχείων του app GreekTrend σε Web Server

Η υλοποίηση της web εφαρμογής ***GreekTrend***, χωρίζεται σε δύο σκέλη. Το front-end και το back-end. Χρησιμοποιήθηκε το ***140dev Streaming API Framework***. Πρόκειται για ένα ***module,*** το οποίο είναι προϊόν δουλειάς του ***Adam Green***. Ο ***Adam Green*** είναι ο ***author*** μιας βιβλιοθήκης πηγαίου κώδικα με το όνομα ***140dev streaming API framework*** , την οποία διανέμει ελεύθερα και δωρεάν στο site του με άδεια δημόσιας χρήσης (***General Puplic License***). Κατόπιν αναβάθμισής του, ο κώδικας είναι πλέον συμβατός με τη νέα απαίτηση για χρήση του ***OAuth***, προκειμένου να δημιουργηθεί σύνδεση ***API Streaming*** του ***Twitter***. Ο στόχος αυτού του κώδικα είναι να παρέχει μια πολύ απλοποιημένη διεπαφή (***interface***) στο ***API Streaming*** του ***Twitter***. Η τρέχουσα έκδοση που παρέχεται από τον ***Author***, δίνει την δυνατότητα για μια βάση δεδομένων, με σκοπό την συνάθροιση των ***tweets***. Η βιβλιοθήκη ***140dev*** είναι γραμμένη σε γλώσσα ***PHP*** και ***Javascript***. Τέλος, χρησιμοποιείται τη βάση δεδομένων ***MySQL***, για αποθήκευση.

4.2 Δημιουργία MySQL Βάσης Δεδομένων

Στα πλαίσια της ανάπτυξης και λειτουργίας της εφαρμογής ***GreekTrend*** , απαιτείται η δημιουργία ΒΔ (Βάσης Δεδομένων), για την αποθήκευση όλης της ροής δεδομένων από το ***Streaming API*** του ***Twitter***. Με τη βάση δεδομένων να βρίσκεται στη «θέση» της, γίνεται η εισαγωγή των πινάκων. Υπάρχουν διάφοροι τρόποι εκτέλεσης των εντολών ***CREATE TABLES*** . Η μέθοδος που χρησιμοποιείται, είναι το «τρέξιμο» της ***phpMyAdmin*** σε τοπικό διακομιστή ***(local server)***, η τοποθέτηση της Βάσης και η εισαγωγή του ***SQL*** αρχείου ***mysql_database_schema.sql*** , έτσι ώστε να εισάγονται οι πίνακες. Το αρχείο περιέχει τις εξής εντολές με τις οποίες δημιουργούνται οι πίνακες στη Βάση Δεδομένων:

```
CREATE TABLE IF NOT EXISTS `json_cache` (

 `tweet_id` bigint(20) unsigned NOT NULL,

 `cache_id` int(10) unsigned NOT NULL AUTO_INCREMENT,

 `cache_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,

 `raw_tweet` text CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,

 PRIMARY KEY (`cache_id`),
```

```
  KEY `tweet_id` (`tweet_id`),

  KEY `cache_date` (`cache_date`)

) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;



CREATE TABLE IF NOT EXISTS `tweets` (

  `tweet_id` bigint(20) unsigned NOT NULL,

  `tweet_text` varchar(160) NOT NULL,

  `created_at` datetime NOT NULL,

  `geo_lat` decimal(10,5) DEFAULT NULL,

  `geo_long` decimal(10,5) DEFAULT NULL,

  `user_id` bigint(20) unsigned NOT NULL,

  `screen_name` char(20) NOT NULL,

  `name` varchar(20) DEFAULT NULL,

  `profile_image_url` varchar(200) DEFAULT NULL,

  `is_rt` tinyint(1) NOT NULL,

  PRIMARY KEY (`tweet_id`),

  KEY `created_at` (`created_at`),
```

```
  KEY `user_id` (`user_id`),
  KEY `screen_name` (`screen_name`),
  KEY `name` (`name`),
  FULLTEXT KEY `tweet_text` (`tweet_text`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
CREATE TABLE IF NOT EXISTS `tweet_mentions` (
  `tweet_id` bigint(20) unsigned NOT NULL,
  `source_user_id` bigint(20) unsigned NOT NULL,
  `target_user_id` bigint(20) unsigned NOT NULL,
  KEY `tweet_id` (`tweet_id`),
  KEY `source` (`source_user_id`),
  KEY `target` (`target_user_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
CREATE TABLE IF NOT EXISTS `tweet_tags` (
  `tweet_id` bigint(20) unsigned NOT NULL,
  `tag` varchar(100) NOT NULL,
  KEY `tweet_id` (`tweet_id`),
```

```
  KEY `tag` (`tag`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

CREATE TABLE IF NOT EXISTS `tweet_urls` (

  `tweet_id` bigint(20) unsigned NOT NULL,

  `url` varchar(140) NOT NULL,

  KEY `tweet_id` (`tweet_id`),

  KEY `url` (`url`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

CREATE TABLE IF NOT EXISTS `users` (

  `user_id` bigint(20) unsigned NOT NULL,

  `screen_name` varchar(20) NOT NULL,

  `name` varchar(20) DEFAULT NULL,

  `profile_image_url` varchar(200) DEFAULT NULL,

  `location` varchar(30) DEFAULT NULL,

  `url` varchar(200) DEFAULT NULL,

  `description` varchar(200) DEFAULT NULL,

  `created_at` datetime NOT NULL,
```

```
  `followers_count` int(10) unsigned DEFAULT NULL,

  `friends_count` int(10) unsigned DEFAULT NULL,

  `statuses_count` int(10) unsigned DEFAULT NULL,

  `time_zone` varchar(40) DEFAULT NULL,

  `last_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP

    ON UPDATE CURRENT_TIMESTAMP,

  PRIMARY KEY (`user_id`),

  KEY `user_name` (`name`),

  KEY `last_update` (`last_update`),

  KEY `screen_name` (`screen_name`),

  FULLTEXT KEY `description` (`description`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

4.3 Εγγραφή της εφαρμογής στο Twitter Apps

Ξεκινώντας με την έκδοση V1.1 του ***Streaming API***, κάθε σύνδεση με το ***API*** απαιτεί τα δικά του μοναδικά ***OAuth tokens***. Για λόγους απλότητας, μπορεί να χρησιμοποιηθεί ένα σύνολο ***Tokens*** για έναν μόνο χρήστη. Εφόσον πραγματοποιηθεί η εγγραφή της εφαρμογής, στη δική μας περίπτωση της ***GreekTrend***, θα υπάρχουν τέσσερα ***OAuth Tokens***. Τα ονόματα των ***tokens*** ενδείκνυται να είναι

διαφορετικά στη σελίδα ***dev.twitter.com*** και στον κώδικα βιβλιοθήκης ***Phirehose***. Στην παρακάτω λίστα φαίνονται τα ονόματα όπως παρουσιάζονται στη σελίδα του Twitter:

Consumer_key = TWITTER_CONSUMER_KEY
Consumer_secret = TWITTER_CONSUMER_SECRET
Access_token = OAUTH_TOKEN
Access_token_secret = OAUTH_SECRET

4.4 Τροποποίηση των config αρχείων

Πριν το στάδιο της εκτέλεσης του κώδικα, θα πρέπει να εισαχθούν τιμές στα ***config*** αρχεία. Αυτό μπορεί να υλοποιηθεί μέσω επεξεργασίας σε τοπικό επίπεδο, είτε με κάποιον editor στο διαδίκτυο, ώστε να γίνουν τα παρακάτω βήματα:

1. Άνοιγμα του αρχείου ***db_config.php*** και συμπλήρωση των πεδίων που περιλαμβάνουν το όνομα χρήστη, τον κωδικό πρόσβασης και το όνομα της Βάσης Δεδομένων MySQL που δημιουργήθηκε για το σκοπό αυτό.
2. Άνοιγμα του αρχείου ***140dev_config.php*** και συμπλήρωση της διεύθυνσης του ηλεκτρονικού ταχυδρομείου (πχ του δυνητικού χρήστης της εφαρμογής ***GreekTrend***) για το αρχείο ***TWEET_ERROR_ADDRESS***. Μ' αυτή τη διεύθυνση θα υπάρχει η δυνατότητα στο χρήστης να δέχεται ροή μηνυμάτων σφάλματος μέσω ηλεκτρονικού ταχυδρομείου, σε περίπτωση που υπάρξουν.
3. Πρέπει επίσης να συμπληρωθεί το σύνολο των ***OAuth tokens*** για τη σύνδεση με το ***API Streaming***. Αυτές οι τιμές πρέπει να τοποθετηθούν στις ακόλουθες δηλώσεις ορισμού στο αρχείο ***140dev_config.php***:

```
Define ('TWITTER_CONSUMER_KEY', '******');
Define ('TWITTER_CONSUMER_SECRET', '******');
Define ('OAUTH_TOKEN', '******');
Define ('OAUTH_SECRET', '******');
```

4.5 Τελική διαγνωστική δοκιμή της εγκατάστασης

Για να εξαλειφθεί οποιαδήποτε πιθανότητα λάθους ή δυσλειτουργίας του κώδικα, το αρχείο ***db_test.php*** είναι αυτό το οποίο όταν εκτελεστεί σωστά σε έναν browser, θα καθιστά σαφές ότι η βάση δεδομένων δημιουργήθηκε επιτυχώς και ότι οι επιλογές διαμόρφωσης της είναι εξίσου σωστές και αυτές.

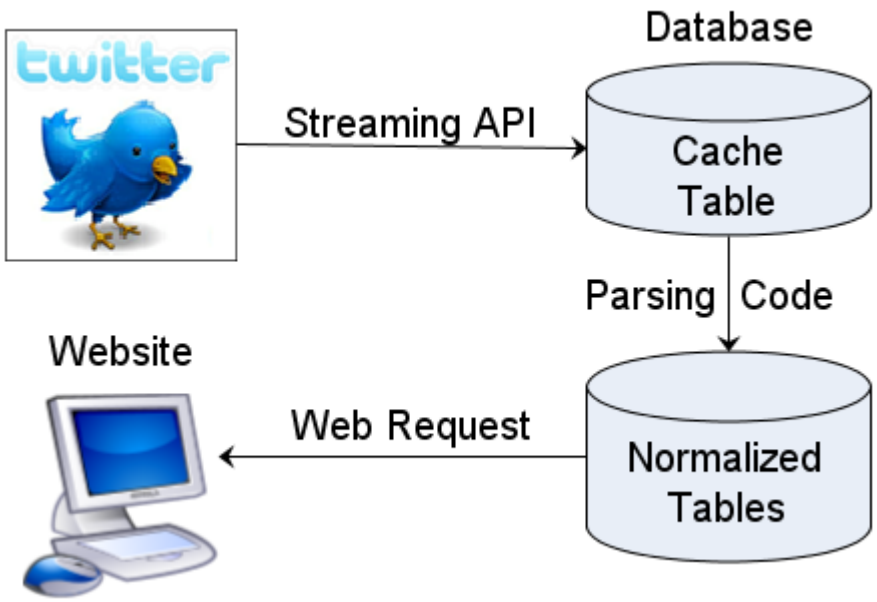
4.6 Καταχώριση των λέξεων-κλειδιών για τη συλλογή των tweets

Ο κώδικας συλλέγει ***tweets*** από το ***API streaming Twitter*** και δίνει τη δυνατότητα στο χρήστη να επιλέξει τις λέξεις-κλειδιά για συλλογή. Κάθε ***tweet*** που περιέχει λέξεις, παραδίδεται μέσα από το **API** σε πραγματικό χρόνο. Στο ***Documentation*** του ***Twitter***, αναφέρεται ότι μπορεί να παρακολουθηθούν έως και 400 λέξεις-κλειδιά από προεπιλογή. Κάθε λέξη-κλειδί μπορεί στην πραγματικότητα να είναι ολόκληρη φράση, αν και στο ***documentation*** δεν αναφέρεται μέγιστο μήκος φράσης.

4.7 Πλεονεκτήματα μνήμης cache στη MySQL ΒΔ με χρήση του Streaming API Twitter

Η θεμελίωση αυτής της προσέγγισης, είναι να γίνει ξεκάθαρος ο διαχωρισμός μεταξύ της διαδικασίας συλλογής νέων tweets από το front-end μέρος της εφαρμογής ***GreekTrend***, έτσι ώστε να εμφανίζονται τα κατάλληλα δεδομένα στο ***User Interface*** το συντομότερο δυνατό.

Αυτή η αρχιτεκτονική δημιουργεί ένα «τοίχος» μεταξύ του ***Streaming API Twitter*** και του front-end της εφαρμογής. Στη μία πλευρά στο back-end υπάρχει το API και ο κώδικας που συλλέγει δεδομένα από αυτό. Από την άλλη πλευρά, στο front-end του application, βρίσκεται ο κώδικας που παρέχει τα επιθυμητά αποτελέσματα στους χρήστες. Ενδιάμεσα, βρίσκονται οι τεχνικές σχεδιασμού της MySQL βάσης δεδομένων και προγραμματισμού που απαιτούνται για την αποθήκευση των δεδομένων από το σύνολο των tweets που εισέρχονται σε ένα κανονικοποιημένο σχήμα βάσης δεδομένων.



Εικόνα 2. Επισκόπηση του application σε σχεδιάγραμμα

Υπάρχουν πολλά πλεονεκτήματα σε αυτήν την αρχιτεκτονική:

- **Συγκεκριμένο rate limit.** Οποιαδήποτε χρήση του ***Streaming API Twitter*** πρέπει να έχει ενσωματωμένη στην αρχιτεκτονική ένα επιτρεπτό rate. Είναι αδύνατο να προβλέψουμε ποια θα είναι τα όρια των ροών μέσω του stream ανά πάσα στιγμή, αφού υπόκεινται σε συχνές αλλαγές που επιτρέπουν στο Twitter να προσαρμόζεται στις απαιτήσεις. Πρέπει να καταστεί σαφές ότι δεν είναι εφικτό, για λόγους λειτουργικότητας του application ***GreekTrend,*** να γίνει απευθείας σύνδεση με το API, επειδή όταν το API φτάσει τα μέγιστα όριά του, η εφαρμογή θα αρχίσει να αποτυγχάνει σε θέματα παραγωγικότητας. Η λύση δίνεται με τον περιορισμό του αριθμού των κλήσεων στο ελάχιστο, και στη συνέχεια με την αποθήκευση των αποτελεσμάτων στην cache μνήμη της βάσης δεδομένων. Με αυτό τον τρόπο, χωρίς πιθανότητα αποτυχίας, θα προκύψουν οπτικοποιημένα αποτελέσματα στο χρήστη.

- **Βελτιστοποίηση απόδοσης.** Η απόδοση του ***API Twitter*** ποικίλλει σε διάφορες χρονικές στιγμές. Για να πετύχει η βελτιστοποίηση της απόδοσης, αρκεί να περιοριστεί η εφαρμογή μόνο στην κλήση του server της βάσης δεδομένων.

- **Ανοχή σε σφάλματα**. Ένα από τα μεγαλύτερα οφέλη του διαχωρισμού της εφαρμογής από το ***Twitter API***, είναι η δυνατότητα να παραμένει σε λειτουργία ακόμη και όταν το ***Twitter*** είναι εκτός λειτουργίας. Με το να γίνεται κλήση με εντολές SQL στη βάση δεδομένων, μειώνεται στο ελάχιστο η πιθανότητα αποτυχίας από το Twitter να αποτρέψει την εφαρμογή από το να εμφανίζει έγκυρα οπτικοποιημένα αποτελέσματα. Το χειρότερο που μπορεί να συμβεί είναι να πρέπει να εναρμονιστεί ο χρήστης με την ιδέα ότι θα συνεχίσει να βλέπει τα παλαιά ***tweets***.

- **Προστασία από αλλαγές του Twitter API**. Το ***API*** "εξελίσσεται" με αρκετά γρήγορο ρυθμό και η συμβατότητα δεν διατηρείται πάντα, με κίνδυνο να εμφανιστούν θέματα λειτουργικότητας. Συνεπώς, πρέπει να υπάρξει μέριμνα για την αποτυχία των κλήσεων προς το ***API***. Όπως αναφέρθηκε παραπάνω, ο διαχωρισμός της εφαρμογής από το ***API*** συνεπάγεται ότι απαιτούνται μόνο αλλαγές στις κλήσεις προς αυτό.

- **Tweet data mining.** Μέσω της ανάπτυξης του συγκεκριμένου application, δίνονται αρκετές ευκαιρίες για περαιτέρω εξέλιξη. Μια από αυτές, είναι η εξόρυξη δεδομένων **(*data mining*)**. Μέσω της εξόρυξης, δίνεται η ευκαιρία για συλλογή πληροφοριών από τη χρήση λέξεων, καθώς οι συμπεριφορές των χρηστών, επιτρέπουν να δημιουργηθεί ένα ευρύ φάσμα λειτουργιών. Τέτοιες λειτουργίες χαρακτηρίζονται οι εμπορικές συναλλαγές, οι δημοσκοπήσεις, τα «πιστεύω» και οι άποψη του κοινού σε συγκεκριμένες χρονικές περιόδους.

Αφού έγινε ανάλυση παρόμοιων συστημάτων, με χρήση του διαδικτύου, όπου παρουσιάστηκε η δομή τους, η αρχιτεκτονική τους, τα πλεονεκτήματα και τα μειονεκτήματά τους, σειρά παίρνει η παρουσίαση της εν λόγω εφαρμογής. Εν προκειμένω της ***GreekTrend***, η οποία στηρίζεται στην ίδια φιλοσοφία με τις προαναφερθείσες εφαρμογές που ήδη κυκλοφορούν. Όπως αναφέρεται παραπάνω, το μεγαλύτερο μέρος της εφαρμογής στηρίχθηκε στη βιβλιοθήκη του ***Adam Green***.

4.8 Αρχιτεκτονική Κώδικα

Η βασική λειτουργία του ***Twitter database server*** είναι η συλλογή των ***tweets*** από το ***API Streaming Twitter*** μετά από το request το δυνητικού χρήστη της εφαρμογής ***Greektrend*** και η διανομή των δεδομένων που θα παραχθούν σε μια σειρά πινάκων ***MySQL*** που υποστηρίζουν τον υπόλοιπο κώδικα.

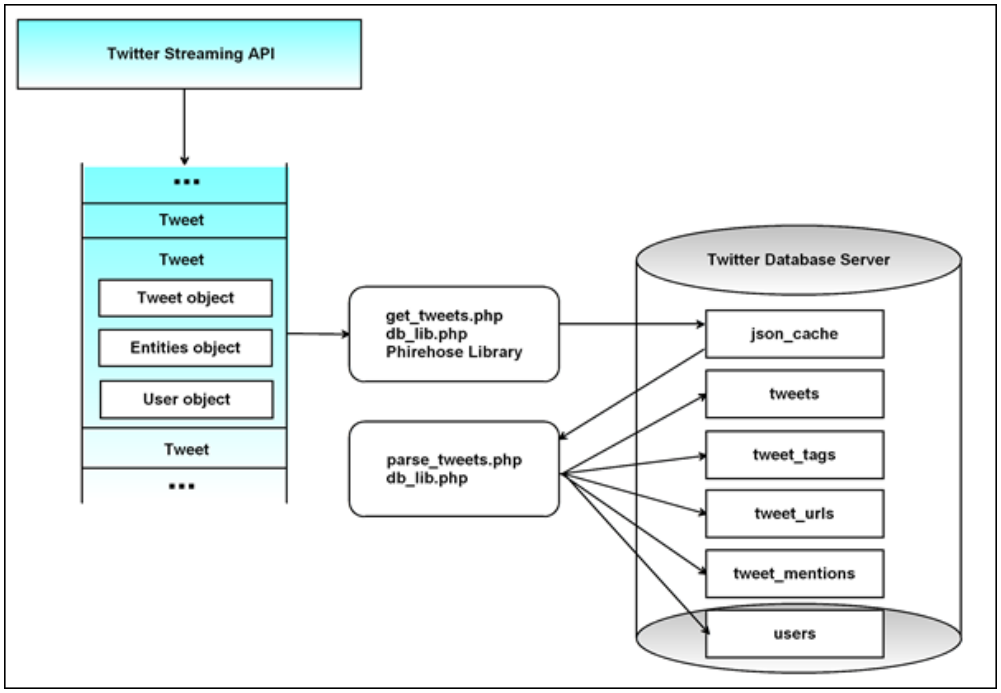
Η βάση δεδομένων MySQL «γεμίζει» με τα παραχθέντα δεδομένα σε δύο βήματα: α) τη λήψη των tweets και β) την ανάλυσή τους σε πολλούς πίνακες. Είναι σημαντικό να διαχωρίζονται οι λειτουργίες αυτές, επειδή τα tweets ενδέχεται να αποστέλλονται μέσω του API Twitter με πολύ γρήγορο ρυθμό (fast rate). Για παράδειγμα, εάν κάθε «τιτίβισμα» αναλύεται και εισάγεται σε πολλούς πίνακες, όπως έχει ληφθεί ακατέργαστο, τόσο ο κώδικας όσο και η βάση δεδομένων είναι πιθανόν να μην είναι σε θέση να συμβαδίζουν με τη ροή δεδομένων, με αποτέλεσμα να χάνονται tweets. Για να επιλυθεί αυτό το πρόβλημα, πραγματοποιείται πρώτα η αποθήκευση των tweets, όπως αυτά παραλαμβάνονται σε έναν απλό πίνακα ***cache*** χωρίς κάποια περαιτέρω ανάλυση. Μέσω μιας ξεχωριστής διαδικασίας, γίνεται η ανάλυση και η αποθήκευση σε ξεχωριστούς πίνακες στη βάση δεδομένων MySQL.

Όσο αφορά την συλλογή tweets, το πρώτο βήμα γίνεται από το από το αρχείο ***get_tweets.php***, το οποίο εκτελείται ως διαδικασία συνεχούς υπόβαθρου (**continuous background process**). Όταν λαμβάνεται ένα νέο, το ***get_tweets.php*** χρησιμοποιεί το αρχείο ***db_lib.php***, με σκοπό την εισαγωγή των στοιχείων στον πίνακα ***json_cache***. Ενώ όλα αυτά είναι σε εξέλιξη, η σύνδεση με το API Streaming Twitter διατηρείται μέσω της βιβλιοθήκης ***Phirehose***.

Το Twitter Streaming API επιστρέφει δεδομένα σε μορφή JSON. Για να γίνει η διαδικασία συλλογής όσο το δυνατόν γρηγορότερα, ολόκληρο το payload του αρχείου JSON για ένα και μόνο tweet, αποθηκεύεται στη βάση δεδομένων ως μία μονή συμβολοσειρά (**string**), χωρίς ανάλυση.

Μια ξεχωριστή διαδικασία παρασκηνίου (**background process**), εκτελείται για το αρχείο ***parse_tweets.php***, το οποίο λαμβάνει τα δεδομένα JSON για κάθε εισερχόμενο «τιτίβισμα» από τον πίνακα ***json_cache***, αναλύει τα components του και τα εισάγει σε ξεχωριστό πίνακα για χρήση. Και σε αυτή την περίπτωση, το αρχείο ***db_lib.php*** χρησιμοποιείται για τη διαχείριση του MySQL κώδικα.

Οι υπόλοιπες μονάδες του κώδικα της εφαρμογής **GreekTrend**, είναι σε θέση να βασίζονται στην συγκεκριμένη βάση δεδομένων, υποστηρίζοντας δεδομένα χωρίς να υπάρχει υποψία άμεσης επαφής με το API.

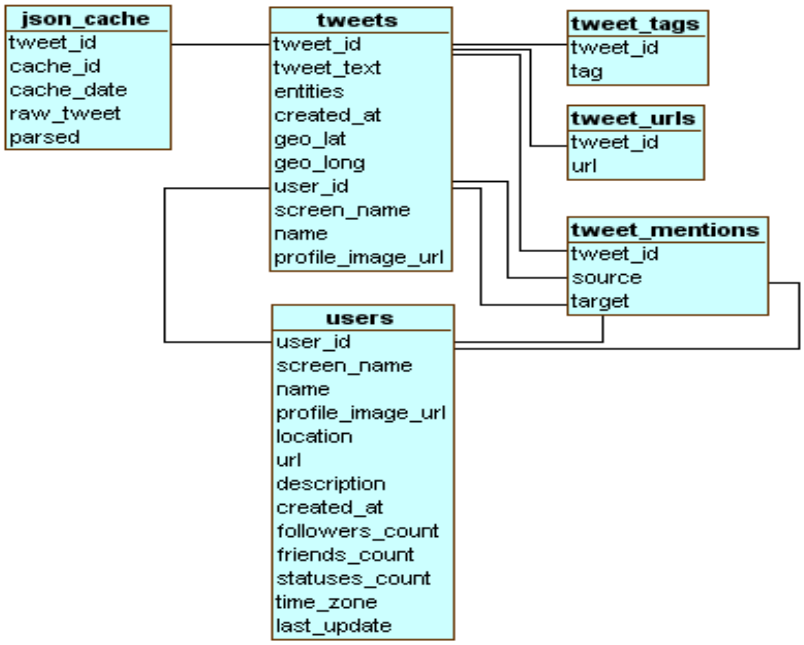


Εικόνα 3. Απεικόνιση της Αρχιτεκτονικής Κώδικα

4.9 Το σχήμα της MySQL βάσης δεδομένων

Όπως περιγράφεται στην αρχιτεκτονική κώδικα, τα tweets λαμβάνονται από το API Streaming Twitter και αποθηκεύονται στη βάση δεδομένων σε δύο βήματα. Πρώτα το όλο το ***payload*** για κάθε "τιτίβισμα" αποθηκεύεται στον πίνακα ***json_cache*** , χάρη στο *αρχείο* ***get_tweets.php***. Σε αυτό το σημείο, δεν πραγματοποιείται ανάλυση. Αντιθέτως, όλα τα δεδομένα εισάγονται στο πεδίο ***raw_tweet***. Έτσι εξασφαλίζεται ότι τα tweets μπορούν να υποστούν επεξεργασία το ταχύτερο δυνατό. Στη συνέχεια, το ***parse_tweets.php*** αναλύει κάθε νέο tweet και διανέμει τα δεδομένα που έχουν παραχθεί σε μια σειρά από πίνακες, όπως είναι οι παρακάτω: **tweets, tweet_mentions, tweet_tags, tweet_urls και users**.

Θεωρητικά, σε ένα πλήρως κανονικοποιημένο σχήμα, ο πίνακας ***tweets*** στη MySQL βάση, θα περιείχε μόνο το ***user_id*** ως σύνδεσμο προς τα υπόλοιπα δεδομένα χρήστη στον πίνακα ***users***. Η φιλοσοφία του σχεδιασμού της συγκεκριμένης βάσης δεδομένων, στα πλαίσια της ανάπτυξης της εφαρμογής, είναι η ομαλοποίηση αρχικά των δεδομένων σε ξεχωριστούς πίνακες και στη συνέχεια η διάσπασή τους με αυτόν τον κανόνα, για λόγους επιδόσεων. Σε αυτήν την περίπτωση, το πιο συνηθισμένο ***query*** θα αξιοποιήσει όλα όσα απαιτούνται για να οπτικοποιηθεί ένα τιτίβισμα, δηλαδή τόσο το "τιτίβισμα" όσο και τα δεδομένα του συντάκτη. Προκειμένου να γίνει η εξαγωγή του ***query*** από έναν μόνο πίνακα, οι τιμές των ***screen_name***, ***name*** και ***profile_image_url*** του συντάκτη του ***tweet*** περιλαμβάνονται σε κάθε ***row*** των ***tweets***.



Εικόνα 4. Απεικόνιση Σχεσιακού μοντέλου ΒΔ της εφαρμογής GreekTrend

4.10 Βιβλιοθήκη Phirohose

Το μεγαλύτερο μέρος του έργου, για τη λήψη των tweets από το Twitter Streaming API γίνεται από τη βιβλιοθήκη ***Phirehose***, η οποία εγκαθίσταται μαζί με το ***framework 140dev Twitter***. Το ***Twitter Streaming API*** απαιτεί μια μόνιμη δικτυακή σύνδεση. Για το λόγο αυτό, υιοθετήθηκε η φιλοσοφία χρήσης της βιβλιοθήκης ***Phirehose***. Στα πλαίσια της εφαρμογής ***GreekTrend***, αναπτύχθηκε κώδικας, ο οποίος συνδράμει στη μόνιμη και συνεχή σύνδεση με τον server του ***API Twitter***. Σκοπός είναι η λήψη μηνυμάτων tweets σε πραγματικό χρόνο (***real-time)***, κυρίως για θέματα αποδοτικότητας. Συμπεραίνουμε λοιπόν, ότι η βιβλιοθήκη κάνει όλη τη δουλειά, όπως: τη δημιουργία της σύνδεσης, τη λήψη των tweets και την αποκατάσταση της σύνδεσης σε περίπτωση που αποτύχει. Επιπλέον, ενημερώνει αυτόματα τις λέξεις-κλειδιά της αναζήτησης που παράχθηκε από τον δυνητικό χρήστη της εφαρμογής ή τα αναγνωριστικά χρήστη που χρειάζονται για τη συλλογή των tweets.

4.11 Αρχείο db_lib.php

Στα πλαίσια της εφαρμογής και της λειτουργικότητάς της, το σύνολο του κώδικα στηρίζεται στο συγκεκριμένο αυτό ***script***, με σκοπό να «διαβάσει» αλλά και να «γράψει» στη βάση δεδομένων MySQL. Καταγράφει τυχόν σφάλματα στο αρχείο ***error_log.txt*** στο ***directory*** της βάσης. Θα πρέπει να τονισθεί εδώ, ότι ο έλεγχος του αρχείου καταγραφής θα πρέπει να συμβαίνει τακτικά, αφού το ***Twitter*** αλλάζει συχνά τις δομές δεδομένων του, προκαλώντας αποτυχίες σε εισροές της βάσης δεδομένων. Τέλος, το αρχείο καταγραφής αποτελεί ένα μεγάλο βοήθημα εντοπισμού σφαλμάτων στην σύνταξη/γραφή νέου κώδικα.

4.12 Αρχείο db_test.php

Όπως προδίδει η ονομασία του, ***db_test.php***, το αρχείο αυτό συμβάλλει στη σωστή λειτουργία της ΒΔ MySQL. Μόλις εγκατασταθεί στο διακομιστή της, μπορεί να εκτελεστεί το παραπάνω ***script*** από οποιοδήποτε πρόγραμμα περιήγησης ιστού για να εξασφαλισθεί ότι όλα λειτουργούν ορθώς.

4.13 Αρχείο get_tweets.php

Η βιβλιοθήκη ***Phirehose*** κάνει τη συλλογή ***tweets*** από το API Streaming Twitter πολύ απλή, όπως φαίνεται σε αυτό το αρχείο. Όπως προαναφέρθηκε, όλες οι εργασίες δημιουργίας και διατήρησης μιας συνεχούς σύνδεσης στο ***API***, γίνονται από τη ***Phirehose***. Επειδή τα δεδομένα των ***tweet*** αποθηκεύονται σε μια βάση δεδομένων MySQL, στον κώδικα συμπεριλαμβάνονται διεργασίες για τη δημιουργία μια συνεχούς σύνδεσης με τη βάση δεδομένων. Αυτό είναι ταχύτερο από το να επανασυνδέεται στη ΒΔ κάθε φορά που εισάγεται ένα νέο ***tweet***.

Η εκκίνηση της διαδικασίας συλλογής απαιτεί μερικά απλά βήματα:

- Το ***API Streaming*** του ***Twitter*** πραγματοποιεί έλεγχο ταυτότητας ***OAuth***, οπότε πρέπει να παρέχονται στη ***Phirehose*** τα διακριτικά του χρήστη. Τα διαπιστευτήρια αυτά δίνονται έπειτα από την εγγραφή στο ***Twitter Developing***, και με την προϋπόθεση ότι υπάρχει έγκυρος και ενεργός λογαριασμός στο ***Twitter*** (γραμμή 45). Αυτές οι τιμές αποθηκεύονται στο ***140dev_config.php***.
- Οι λέξεις-κλειδιά που χρησιμοποιούνται για την επιλογή tweets από το ***API*** μεταβιβάζονται ως πίνακας στη ***Phirehose*** με τη βοήθεια της συνάρτησης ***setTrack ()*** (γραμμή 53).
- Τέλος, η συνάρτηση ***consume ()*** καλείται για να ξεκινήσει τη συλλογή (γραμμή 57).

4.14 Αρχείο monitor_tweets.php

Το ***monitor_tweets.php*** ελέγχει τον πίνακα ***tweets*** της ΒΔ για νέα «τιτιβίσματα» και στέλνει μήνυμα σφάλματος στο ηλεκτρονικό ταχυδρομείο αν δεν εντοπίσει κάποιο καινούριο. Ο έλεγχος και η αποστολή του μηνύματος σφάλματος στη διεύθυνση του ηλεκτρονικού ταχυδρομείου, αποθηκεύονται στο ***140dev_config.php***.

4.15 Αρχείο parse_tweets.php

Το ***parse_tweets.php*** επεξεργάζεται κάθε νέο ***tweet*** που προστίθεται στον πίνακα ***json_cache***, με τη βοήθεια του ***get_tweets.php***. Το περιεχόμενο αυτών των ***tweets*** εξάγεται από το πεδίο ***raw_tweet*** στον πίνακα ***json_cache*** και εισάγεται στους εξής πίνακες: ***tweets***, ***tweet_tags***, ***tweet_urls***, ***tweet_mentions*** και ***users***. Εάν ένας χρήστης βρίσκεται ήδη στον πίνακα ***users***, τα δεδομένα του ενημερώνονται με τις τιμές από το πιο πρόσφατο «τιτίβισμα».

Οι οντότητες σε κάθε ***tweet payload*** περιέχουν δεδομένα για κάθε **@*mention, tag*** και διεύθυνση ***URL*** στο ***tweet***. Το αντικείμενο έχει αναλυθεί και οι ατομικές **@mentions**, ***tags*** και διευθύνσεις ***URL*** αποθηκεύονται στους πίνακες ***tweet_mentions, tweet_tags*** και ***tweet_urls*** αντίστοιχα. Αυτοί οι πίνακες, μπορούν δυνητικά να εξάγουν δεδομένα για τη συλλογή στατιστικών στοιχείων, όπως για παράδειγμα τα πιο συχνά χρησιμοποιούμενα ***tags*** και ***urls***.

Κεφάλαιο 5ο

Επισκόπηση του GreekTrend app

Στα κεφάλαια που προηγήθηκαν, έγινε εκτενής αναφορά και ανάλυση τόσο των εφαρμογών που ανήκουν στο πεδίο του ***GreekTrend app***, των τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής όσο και της αρχιτεκτονικής του κώδικα. Βήμα-βήμα παρουσιάστηκαν οι λειτουργίες των αρχείων ***.php***, που συγκροτούν στο σύνολό τους τον πυρήνα του app. Πρόκειται δηλαδή για το ***back-end*** κομμάτι της εφαρμογής.

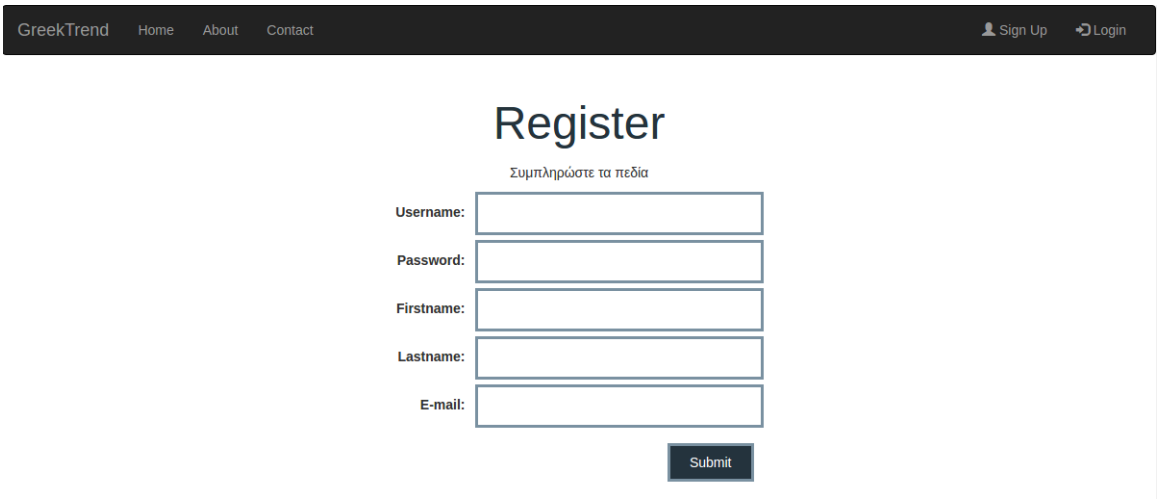
Σε αυτό το κεφάλαιο, θα γίνει παρουσίαση του ***front-end*** του ***app***. Συγκεκριμένα θα γίνει η αναφορά όλων των πτυχών της λειτουργίας της. Αυτές είναι:

- Η εγγραφή του χρήστη στην εφαρμογή ***GreekTrend (Register)***
- Η είσοδος του εγγεγραμμένου χρήστη ***(Login)***
- Το ***main page*** της εφαρμογής
- Η λειτουργία ***Home***
- Η λειτουργία ***About***
- Η λειτουργία **Contact**
- Η έξοδος του εγγεγραμμένου χρήστη **(*Logout)***
- Η λειτουργία επικοινωνίας μέσω ***email***
- Η λειτουργία αναζήτησης λέξης-κλειδιού **(*search*)**
- Η λειτουργία ***Trends***

5.1 Η εγγραφή του χρήστη στην εφαρμογή *GreekTrend* (*Register*)

Στο καλωσόρισμα του δυνητικού χρήστη στο app, γίνεται σε αυτόν ξεκάθαρο ότι αποτελεί απαραίτητη προϋπόθεση η εγγραφή του στην εφαρμογή, προκειμένου να χρησιμοποιήσει τις υπηρεσίες του ***GreekTrend***. Σκοπός της εγγραφής των χρηστών, είναι η συγκέντρωση ενός σταθερού αριθμού ανθρώπων που θα τη χρησιμοποιούν, στοχεύοντας μελλοντικά στην αύξηση του αριθμού αυτού και αποφεύγοντας όσο είναι δυνατό την περιστασιακή χρήση ή την απλή επίσκεψη.

Το σύνολο των επιλογών και των λειτουργιών εντός της εφαρμογής είναι αρκετά απλουστευμένο έτσι ώστε να μπορεί ο καθένας να τη χρησιμοποιεί. Δεν απαιτείται δηλαδή, κάποιο επίπεδο δεξιοτήτων ή γνώσεων.



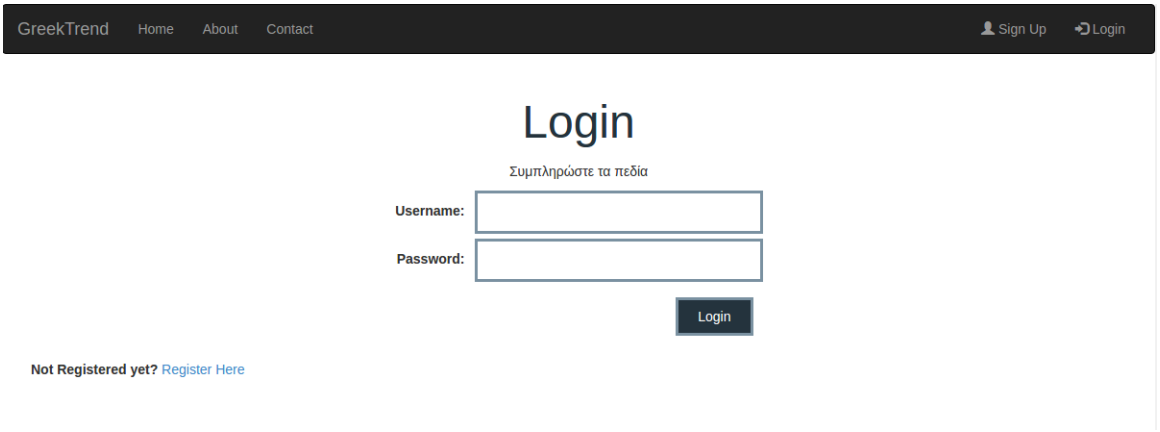
Εικόνα 5. Απεικόνιση της Register Page του GreekTrend App

Όπως φαίνεται στην παραπάνω εικόνα, η ***Register Page*** του ***app***, αποτελείται από πεδία συμπλήρωσης των βασικών στοιχείων που θα πρέπει να συμπληρωθούν από το δυνητικό χρήστη υποχρεωτικά, προκειμένου να ολοκληρωθεί η διαδικασία της εγγραφής του στην εφαρμογή. Πρόκειται για τα πεδία: ***username, password, firstname, lastname, e-mail.***

Η διαδικασία ολοκληρώνεται με την επιλογή ***submit.*** Για να θεωρηθεί ορθή η εγγραφή, ο χρήστης οφείλει υποχρεωτικά να συμπληρώσει όλα τα πεδία, ειδάλλως θα ενημερώνεται με σχετικό μήνυμα ότι έχει ξεχάσει κάποιο κενό. Όλα τα παραπάνω προσωπικά στοιχεία αποθηκεύονται στη ***MySQL*** ΒΔ, με το όνομα ***cms***.

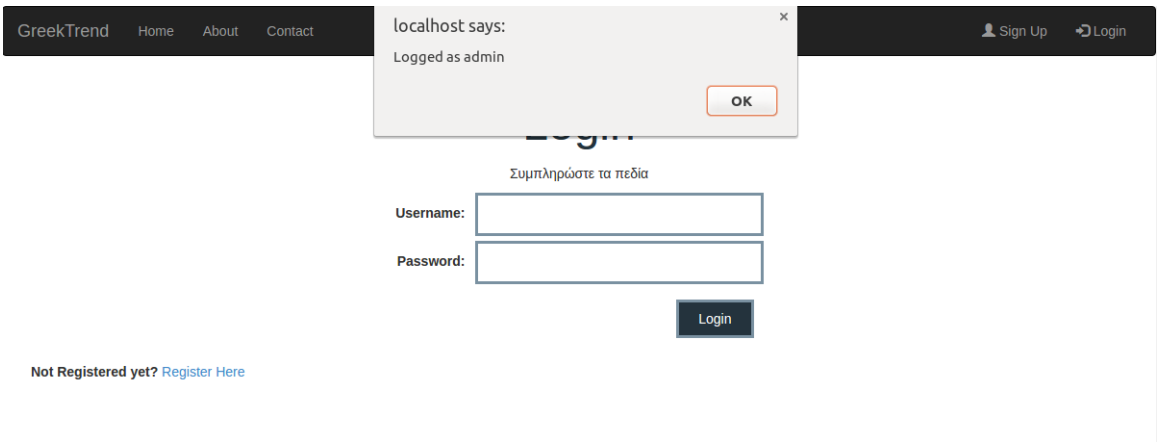
5.2 Η είσοδος του εγγεγραμμένου χρήστη (*Login*)

Εφόσον έχει ολοκληρωθεί η εγγραφή του χρήστη στην εφαρμογή, σειρά παίρνει η είσοδός του σε αυτή. Συμπληρώνοντας ορθά τα πεδία ***Username*** και ***Password***, που έχει ο ίδιος στην κατοχή του από την εγγραφή, του παρέχεται η είσοδος στο ***Main Page*** του ***GreekTrend App***. Να σημειωθεί ότι όπως ίσχυε στην εγγραφή, έτσι και κατά την είσοδο του, ο χρήστης δεν πρέπει να αφήνει κενό πεδίο, διαφορετικά δεν θα μπορέσει να έχει πρόσβαση, όσες προσπάθειες και αν γίνουν από μέρους του. Σχετικό μήνυμα διαλόγου (***dialog box)*** θα τον ειδοποιεί να συμπληρώσει το κενό πεδίο.



Εικόνα 6. Απεικόνιση της Login Page του GreekTrend App

Να τονισθεί ότι σε περίπτωση που ο χρήστης έχει αυξημένα δικαιώματα στην εφαρμογή, χαρακτηρίζεται για παράδειγμα ως ***Admin***, τότε ένα ***dialog box***, όπως φαίνεται στην παρακάτω εικόνα θα εμφανιστεί, καλωσορίζοντας τον κατά την είσοδό του. Σε περίπτωση που, ένας επισκέπτης επιθυμεί να εισέλθει, μπορεί με την επιλογή που παρουσιάζεται κάτω αριστερά του App, να πραγματοποιήσει εγγραφή.



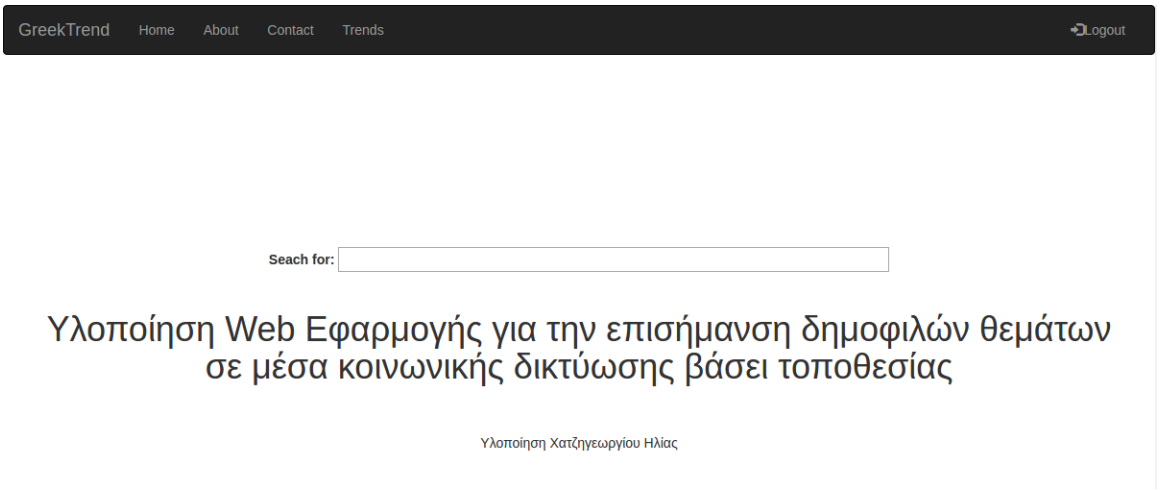
Εικόνα 7. Απεικόνιση Εισόδου Χρήστη ως Admin

5.3 Το Main Page της εφαρμογής

Έχοντας ολοκληρώσει επιτυχώς τις διαδικασίες εγγραφής και εισόδου στην εφαρμογή, ο χρήστης έχει πλέον πρόσβαση στην κεντρική σελίδα (***main page)*** του app. Το κύριο σενάριο της εφαρμογής είναι προφανές. Ο δυνητικός χρήστης, έχει μπροστά του το ***searchbar***, με το οποίο πληκτρολογώντας τη λέξη κλειδί της επιθυμίας του, μπορεί να δει τα αποτελέσματα της αναζήτησής του, οπτικοποιημένα σε χάρτη της ***Google***, με τη μορφή πινέζας (***pins***). Τα δεδομένα προέρχονται

από το ***Twitter*** , χάρη στη μόνιμη σύνδεση που “γεφυρώνει” την εφαρμογή με το ***Streaming API Twitter.*** Τα αποτελέσματα ενδέχεται να παρουσιάζονται με την μορφή ***hashtag*** ή να βρίσκεται η λέξη-κλειδί μέσα σε πρόταση. Αξίζει να σημειωθεί ότι δεν υπάρχει γλωσσικός περιορισμός όσο αφορά την αναζήτηση που θα επιλέξει να πραγματοποιήσει ο χρήστης του app.

Πέραν της αναζήτησης, υπάρχουν οι λειτουργίες ***Home***, ***About***, ***Contact***, ***Trends***, ***Sign up*** και ***Logout***. Συγκεκριμένα, μέσα από τις επιλογές ***About*** και ***Contact***, του δίνεται η επιλογή να μάθει λίγα λόγια για την εφαρμογή και τον ***Author*** της, καθώς και να επικοινωνήσει μαζί του μέσω ηλεκτρονικού ταχυδρομείου σε περίπτωση απορριών που ενδεχομένως να προκύψουν στην πορεία χρήσης του app. Η λειτουργία ***Trends***, θα αναλυθεί εκτενέστερα παρακάτω.

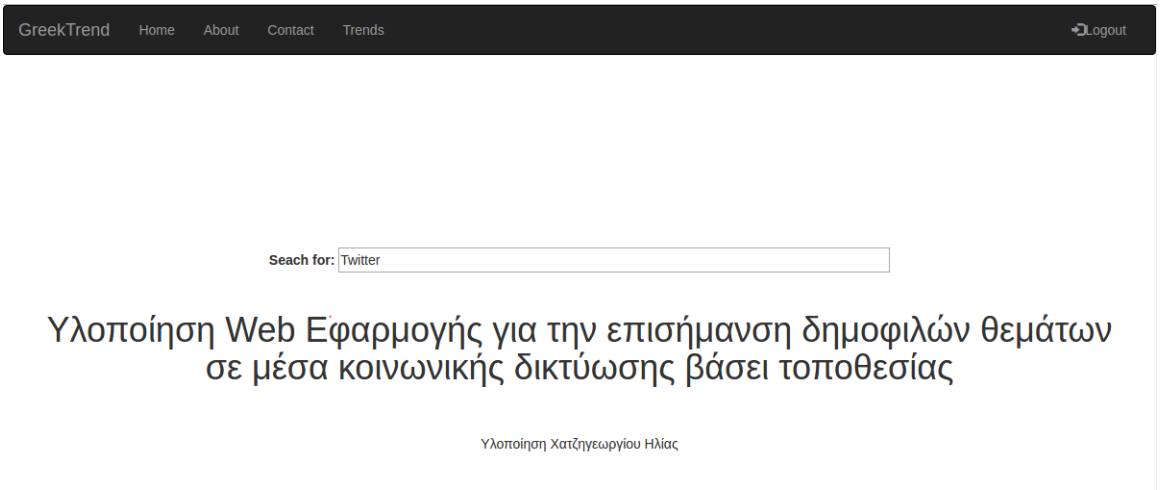


Εικόνα 8. Απεικόνισης της Main Page

5.4 Η λειτουργία αναζήτησης λέξης-κλειδιού (*search*)

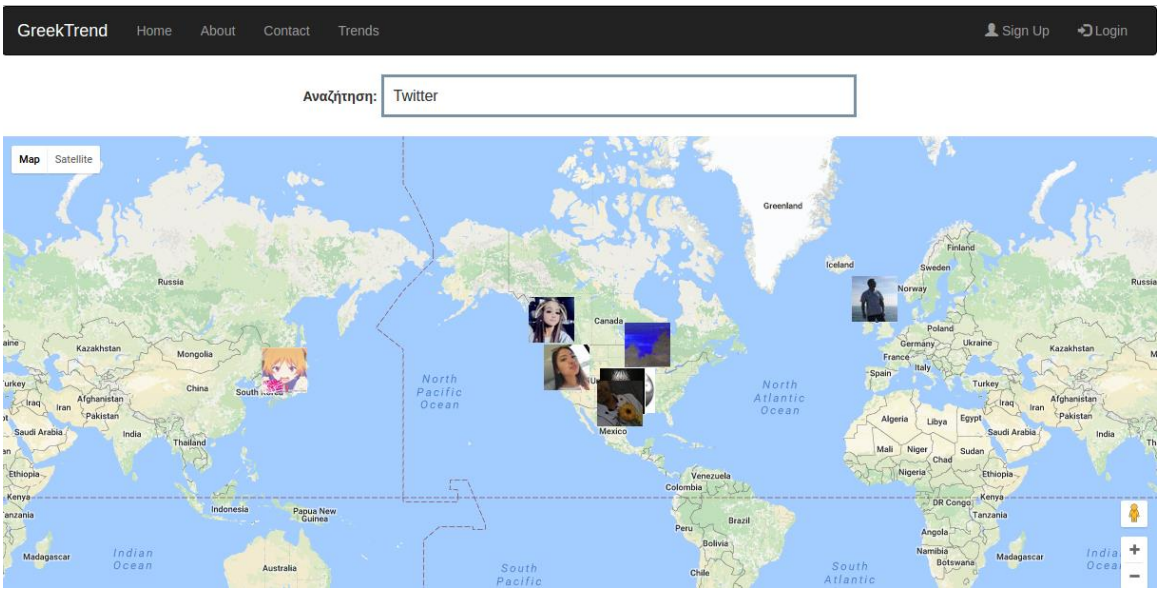
Ουσιαστικά πρόκειται για τον πυρήνα της εφαρμογής. Με τη λειτουργία αυτή, ο χρήστης έχει πρόσβαση σε θέματα που τον ενδιαφέρουν ανά την Ελλάδα και τον κόσμο. Η διαδικασία είναι απλή. Ο χρήστης εισάγει τη λέξη-κλειδί της επιθυμίας του, σε όποια γλώσσα επιλέξει και “χτυπώντας” το πλήκτρο ***enter***, θέτει σε λειτουργία όλο τον μηχανισμό που θα παράγει τα αποτελέσματα. Δίνεται ένας ελάχιστος χρόνος κάποιων δευτερολέπτων, έτσι ώστε να συγκεντρωθούν όσα περισσότερα δεδομένα γίνεται.

Χάριν παραδείγματος, θα εισαχθεί η λέξη-κλειδί: ***Twitter***. Θα πραγματοποιηθεί όλη η διαδικασία από την στιγμή εισαγωγής του λήμματος μέχρι την οπτικοποίηση των αποτελεσμάτων στο χάρτη.



Εικόνα 9. Εισαγωγή λέξης-κλειδί στην αναζήτηση

Κατά την εκτέλεση της αναζήτησης, όπως προαναφέρθηκε μπαίνει σε λειτουργία η επικοινωνία της εφαρμογής με το ***Streaming API Twitter***, με σκοπό την εξαγωγή αποτελεσμάτων. Με το πέρας την αναζήτησης, γίνεται μετάβαση στην σελίδα με την παρουσίαση των αποτελεσμάτων “καρφιτσωμένα” στο χάρτη.

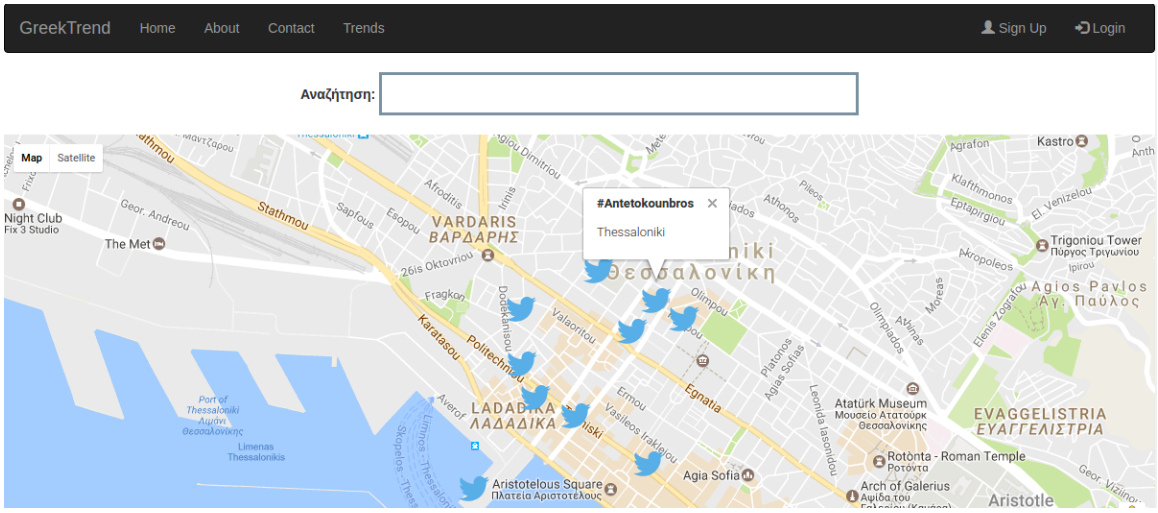


Εικόνα 10. Απεικόνιση οπτικοποίησης αποτελεσμάτων

5.5 Η λειτουργία Trends

Η λειτουργία αυτή, δίνει τη δυνατότητα παροχής πληροφοριών, σχετικά με τις τάσεις που υπάρχουν. Συγκεκριμένα, γίνεται αναφορά στην ευρύτερη περιοχή της πόλης της Θεσσαλονίκης (από επιλογή του σχεδιαστή). Ο δυνητικός χρήστης επιλέγοντας τη λειτουργία ***Trends***, έχει πρόσβαση σε ένα σύνολο από τάσεις, στις οποίες γίνεται επισκόπηση, των θεμάτων ενδεχομένως που απασχολούν τους κατοίκους της Θεσσαλονίκης τις τελευταίες μέρες. Πρόκειται για ένα χρήσιμο εργαλείο, καθώς

ενδέχεται να εξαχθούν πολλά και χρήσιμα δεδομένα προς φιλτράρισμα και επεξεργασία, ανάλογα με τις ανάγκες του εκάστοτε χρήστη. Η μόνη διαφορά που παρουσιάζει με τη λειτουργία της αναζήτησης, είναι στην οπτικοποίηση των αποτελεσμάτων. Στη λειτουργία ***Trends***, η οπτικοποίηση αναπτύσσεται με ***pins,*** τα οποία έχουν τη μορφή του σπουργιτιού, το ***Twitter Logo***.



Εικόνα 11. Απεικόνιση λειτουργίας Trends

Στην παραπάνω εικόνα, αναπαριστάται παράδειγμα της λειτουργίας ***Trends***. Όπως φαίνεται μία από τις τάσεις που υπάρχουν στην πόλη της Θεσσαλονίκης, αναφέρεται στα αδέρφια και Έλληνες καλαθοσφαιριστές ***Γιάννη και Θανάση Antetokounmpo***, οι οποίοι το τελευταίο διάστημα με τα επιτεύγματά τους γίνονται αντικείμενο σχολιασμού παγκοσμίως. Πρέπει να τονισθεί ότι το παράδειγμα αυτό, εκτελέστηκε στις 19 Ιουνίου 2017, μια χρονική στιγμή καθόλου τυχαία αν λάβουμε υπόψη ότι ξεκινούσε μία σειρά από εκδηλώσεις και φιλανθρωπικούς αγώνες προς τιμήν τους.

Κεφάλαιο 6ο

Συμπεράσματα

Αυτό το κεφάλαιο παρουσιάζει τα συμπεράσματα που προέκυψαν κατά την εκπόνηση της διπλωματικής εργασίας. Πιο αναλυτικά, θα γίνει αναφορά στις γνώσεις που λάβαμε κατά την διάρκεια της εργασίας, στις προοπτικές που έχει η εφαρμογή, στις όποιες σκοπέλους αναπτύχθηκαν και τέλος, στις βελτιώσεις που θα μπορούσαν να γίνουν μελλοντικά.

Ξεκινώντας από τον τίτλο της διπλωματικής εργασίας, προκύπτει η ανάγκη να μιλήσουμε για την ανάπτυξη ***web application***, το οποίο τάσσεται με τις σύγχρονες ανάγκες της εποχής μας, ανάγκες που έχουν να κάνουν με την συγκομιδή, αποθήκευση, επεξεργασία, ανάλυση δεδομένων/πληροφοριών στο διαδίκτυο και τέλος την εξαγωγή χρήσιμων συμπερασμάτων που ενδεχομένως προκύπτουν. Δεν μιλάμε όμως για μια απλή εισροή και εκροή δεδομένων στην εξίσωσή μας μέσα στο σύστημα. Έχουμε να κάνουμε με την εκμετάλλευση μεταβλητών/αρωγών, οι οποίες τα τελευταία χρόνια ήρθαν στα πράγματα και μπορούν να παίξουν καταλυτικό ρόλο στην συγκέντρωση δεδομένων. Πρόκειται για τα μέσα κοινωνικής δικτύωσης, τα οποία χαίρουν μεγάλης αναγνώρισης ανά τον κόσμο και παρατηρείται διαρκής αύξηση του αριθμού τους με κύριο λόγο, την κάλυψη των

αναγκών που δυνητικά εμφανίζονται στον ανθρώπινο πληθυσμό. Στη δική μας περίπτωση, το μέσο κοινωνικής δικτύωσης που χρησιμοποιήσαμε ήταν το ευρέως γνωστό, ***Twitter***.

Από τα πρώτα μας βήματα στο ξεκίνημα, έγινε μια μελέτη για το πώς γίνεται το ***project planning*** εφαρμογής. Ουσιαστικά την ανάλυση των απαιτήσεων μιας εφαρμογής, την επιλογή των εργαλείων – τεχνολογιών, το καταμερισμό των προς υλοποίηση θεμάτων και την σειρά της υλοποίησης. Έτσι στην περίπτωση μας, πρώτα έγινε η συγκέντρωση και η ανάλυση των απαιτήσεων. Εκεί ουσιαστικά καταγράψαμε το τι θέλουμε να κάνει η εφαρμογή μας, επιλέξαμε το τι εργαλεία θα χρησιμοποιήσουμε, και ξεκίνησε η υλοποίηση από την δημιουργία της βάσης δεδομένων ένα από το πιο σημαντικά κομμάτια του ***GreekTrend App***. Είναι πολύ σημαντικό να αναφέρουμε ότι προηγήθηκε λεπτομερής αναφορά αναζήτησης και παρουσίασης παρόμοιων εφαρμογών που κυκλοφορούν για εμπορική χρήση, με την προς υλοποίηση δική μας. Συγκεκριμένα, παρουσιάστηκαν οι χρήσεις, οι δομές αλλά και οι αρχιτεκτονικές της εκάστοτε εφαρμογής που βρέθηκε στη μελέτη αυτή. Κατά τη διάρκεια της υλοποίησης της εφαρμογής, τα πρακτικά θέματα που προέκυψαν δεν ήταν λίγα, με αποτέλεσμα να υπάρχει συνεχής προσαρμογή στα νέα δεδομένα. Προς αντιμετώπιση τυχόν προβλημάτων επιλέξαμε να αλλάξουμε την αρχική μας μελέτη ώστε να μπορέσουμε να συνεχίσουμε ομαλά. Αυτό δείχνει ότι σε μια τόσο πολύπλοκη και πολυεπίπεδη εφαρμογή ποτέ δεν μπορείς να προβλέψεις τα πάντα.

Στην συνέχεια, αφήνοντας πίσω το θεωρητικό κομμάτι, στα πλαίσια της υλοποίησης, μεγάλο "στοίχημα" για την εφαρμογή μας, ήταν η σύνδεση του **front-end** με το **back-end** τμήματός της, καθώς και η φιλοσοφία έναρξης και διατήρησης μιας μόνιμης επικοινωνίας του ***application*** με το ***Streaming API Twitter***. Οι μεγαλύτερες δυσκολίες παρουσιάστηκαν με το **API**, διότι κατά την αρχική μας επιλογή, προτιμήσαμε το ***REST API Twitter.*** Ωστόσο διαπιστώθηκε ότι η εισροή και η ποιότητα των δεδομένων δεν κάλυπταν ούτε στο ελάχιστο τις βασικές απαιτήσεις για την εφαρμογή. Επομένως, εγκαταλείψαμε την προοπτική του ***REST*** και στραφήκαμε στις επιλογές που προσφέρει το δεύτερο ***API Twitter***, το ***Streaming***. Ωστόσο υπήρχαν περιπτώσεις που εμφανίστηκαν αρκετά αδιέξοδα κατά τη διάρκεια ανάπτυξης. Για παράδειγμα, το ***Streaming API Twitter***, εμφανιζόταν ουκ ολίγες φορές εκτός λειτουργίας ***(offline)***, ή υπήρχαν περιπτώσεις όπου δεχθήκαμε απαγόρευση εισόδου ***(ban)***, με την αιτιολογία ότι προσπαθούσαμε πολλαπλές εισόδους για επιβλαβείς σκοπούς. Τα προβλήματα αντιμετωπίστηκαν επιτυχώς, με την εξολοκλήρου αλλαγή και επανεγκατάσταση με τις κατάλληλες αλλαγές του διαύλου επικοινωνίας μεταξύ εφαρμογής και ***API***.

Κομβικό σημείο για την περάτωση της διπλωματικής εργασίας, ήταν η λύση στο πρόβλημα που παρουσιάστηκε κατά την οπτικοποίηση των αποτελεσμάτων στον χάρτη. Κατά την εκτέλεση της αναζήτησης, η βάση δεδομένων γέμιζε με ***tweets***, ωστόσο δεν εμφανίζονταν. Το πρόβλημα βρισκόταν στο γεγονός ότι κατά την σύνταξη ενός ***tweet***, ο συντάκτης του κειμένου δεν είχε ενεργοποιημένη την τοποθεσία της συσκευής του, με αποτέλεσμα να μην υπάρχει "αποτυπωμένη" η γεωγραφική πληροφορία. Ωστόσο, εξετάστηκε η εναλλακτική επιλογή, στη οποία είχαμε ***location*** αλλά όχι ακριβείς συντεταγμένες. Η μόνη διαφορά σε αυτά τα ***tweets***, είναι ότι ενδεχομένως να παρουσιάζουν κάποια χιλιομετρική απόκλιση με την ακριβή τοποθεσία του ***tweet*** σε σχέση με αυτά που έχουν ***X,Y.***

Η προς υλοποίηση web εφαρμογή, χαρακτηρίζεται ως ένα χρήσιμο εργαλείο το οποίο ακολουθεί τις τάσεις των τεχνολογιών, δομημένη και ανεπτυγμένη με εργαλεία και γλώσσες προγραμματισμού αρκετά δημοφιλή προς το ευρύ κοινό. Η εφαρμογή αυτή θα λέγαμε ότι εξυπηρετεί τις βασικές ανάγκες του χρήστη, στα πλαίσια αναζήτησης και οπτικοποίησης των λέξεων που επιθυμεί, έχοντας αφήσει πολλές προοπτικές παραμετροποίησης και βελτιστοποίησης ανάλογα με τις ανάγκες που μελλοντικά θα προκύψουν.

Βιβλιογραφία

Gelernter, Judith, and Gang Wu**. "*High performance mining of social media data*."** Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the extreme to the campus and beyond. ACM, [2012].

Zhang, Wei, and Judith Gelernter. ***"Geocoding location expressions in Twitter messages: A preference learning method."*** *Journal of Spatial Information Science* 2014.9 (2014): 37-70.

How to get tweets from API Twitter, retrieved from https://dev.twitter.com/rest/reference/get/search/tweets

How to find trends from API Twitter, retrieved from https://dev.twitter.com/rest/reference/get/trends/place

REST API Twitter Documentation, retrieved from https://dev.twitter.com/rest/public

Streaming API Twitter Documentation, retrieved from https://dev.twitter.com/streaming/overview

How to build a web application with twitter api, available from https://code.tutsplus.com/tutorials/building-with-the-twitter-api-using-real-time-streams--cms-22194

Adam Green **"140dev Streaming API Framework" available from http://140dev.com/free-twitter-api-source-code-library/**

Lerdorf, Rasmus (2012-07-20). **«*I wonder why people keep writing that PHP was ever written in Perl. It never was. #php*»**

"Introduction: What can PHP do?". PHP Manual. Retrieved 2017.

Jump up to**: *"Embedding PHP in HTML".*** O'Reilly. 2001-05-03. Retrieved 2017**.**

«*First mention of HTML Tags on the www-talk mailing list*». World Wide Web Consortium. 20 Οκτωβρίου 1991. Ανακτήθηκε το 2017**.**

«*Index of elements in HTML 4*». World Wide Web Consortium. 24 Δεκεμβρίου 1999. Ανακτήθηκε το 2017.

Connolly, Daniel (24 November 1992). ***«Document Type Definition for the HyperText Markup Language as used by the World Wide Web application»***. CERN. Ανακτήθηκε το 2017. Ενότητα ***«Revision History».***

Spurlock, Jake (2013). Bootstrap (1. ed. έκδοση). Sebastopol, CA: O'Reilly Media, σελ. 1. ISBN 978-1-449-34391-0.

Cochran, David (November 12, 2012). Twitter Bootstrap Web Development (1st έκδοση). Packt Publishing, σελ. 100. ISBN 978-1849518826.

Nixon, Robin. ***Learning PHP, MySQL, JavaScript, and CSS: A step-by-step guide to creating dynamic websites. " O'Reilly Media, Inc."***.

Hamilton, Naomi (2008-06-31). ***«The A-Z of Programming Languages: JavaScript»***. computerworld.com.au.

«JavaScript: The World's Most Misunderstood Programming Language». Crockford.com. Ανακτήθηκε το 2017.

"phpMyAdmin 4.7.2 is released". phpMyAdmin. phpMyAdmin contributors. 2017-06-29. Retrieved 2017-06-29.

"phpMyAdmin - About". phpMyAdmin. Retrieved 2017.

Delisle, Marc (2010). Mastering phpMyAdmin 3.3.x for Effective MySQL Management. Packt Publishing. p. 359. ISBN 978-1-84951-354-8.

Πίνακας Εικόνων

Παράρτημα κώδικα

Σε αυτό το παράστημα, παραθέτω τον κώδικα που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής GreekTrend.

About.php

```
<!doctype html>

<?php
  session_start();
  if (!$_SESSION['Username']){
    header ("location:Login.php");
  }
  ?>

<html>
<head>
<meta charset="utf-8">
<title>About</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="css/bootstrap.min.css" rel="stylesheet">
<!--<link href="css/style.css" rel="stylesheet">-->

<!-- Include jQuery and bootstrap JS plugins -->
<script src="js/jquery-2.1.0.min.js"></script>
<script src="js/bootstrap.min.js"></script>
</head>

<body>
<!--header-->
<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="IndexSite1.php">GreekTrend</a>
    </div>
    <div>
    <button class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
          <span class="glyphicon glyphicon-align-justify"></span>
      </button>
      <nav class="navbar-collapse collapse" role="navigation">
     <ul class="nav navbar-nav">
      <li class="active">
            <li><a href="IndexSite1.php">Home</a></li>
            <li><a href="about.php">About</a></li>
            <li><a href="contact.php">Contact</a></li>
            <li><a href="trends.php">Trends</a></li>
            </li>
     </ul>

     <ul class="nav navbar-nav navbar-right">

```

```
        <li><a href="Logout.php"><span class="glyphicon glyphicon-log-
in"></span> Logout</a></li>
      </ul>
      </nav>
    </div>
  </div>
</nav>
<!-- Site banner -->
<div class="banner">
    <div class="containerfirst">
        <h1>Διπλωματική Εργασία</h1>
        <p>ΠΜΣ,Προηγμένα Συστήματα Πληροφορικής</p>
    </div>
</div>

<!-- Middle content section -->
<div class="banner">
    <div class="containerseccond">

            <h1>Υλοποίηση Web Εφαρμογής για την επισήμανση δημοφιλών θεμάτων σε μέσα κο
ινωνικής δικτύωσης βάσει τοποθεσίας </h1>
        <p>Χατζηγεωργίου Ηλίας</p>


    </div>
</div>
<!--footer-->

<footer class="footer">
      <div class="container">
        <p class="text-muted">Πειραιάς 2016</p>
      </div>
    </footer>

</body>
</html>
```

Contact.php

```
<!doctype html>
<?php
   session_start();
   if (!$_SESSION['Username']){
      header ("location:Login.php");
   }
   ?>

<html>
<head>
<meta charset="utf-8">
<title>Contact</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="css/bootstrap.min.css" rel="stylesheet">
```

```
15. <!--<link href="css/style.css" rel="stylesheet">-->
16. <link href="css/contact_form.css" rel="stylesheet">
17.
18.
19.
20. <!-- Include jQuery and bootstrap JS plugins -->
21. <script type="text/javascript" src="js/jquery-2.1.0.min.js"></script>
22. <script type="text/javascript" src="js/bootstrap.min.js"></script>
23. <script type="text/javascript" src="js/jquery.form.js"></script>
24. <script type="text/javascript" src="js/jquery.validate.min.js"></script>
25. <script type="text/javascript" src="js/contact.js"></script>
26. </head>
27.
28. <body>
29. <!--header-->
30. <nav class="navbar navbar-inverse">
31.   <div class="container-fluid">
32.     <div class="navbar-header">
33.       <a class="navbar-brand" href="IndexSite1.php">GreekTrend</a>
34.     </div>
35.     <div>
36.     <button class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
37.           <span class="glyphicon glyphicon-align-justify"></span>
38.       </button>
39.       <nav class="navbar-collapse collapse" role="navigation">
40.     <ul class="nav navbar-nav">
41.       <li class="active">
42.       <li><a href="IndexSite1.php">Home</a></li>
43.             <li><a href="about.php">About</a></li>
44.             <li><a href="contact.php">Contact</a></li>
45.             <li><a href="trends.php">Trends</a></li>
46.             </li>
47.     </ul>
48.
49.     <ul class="nav navbar-nav navbar-right">
50.
51.       <li><a href="Logout.php"><span class="glyphicon glyphicon-log-
    in"></span> Logout</a></li>
52.     </ul>
53.     </nav>
54.   </div>
55.  </div>
56. </nav>
57. <!-- Site banner -->
58. <div class="banner">
59.   <div class="container">
60.       <h1 class="align-center">  Επικοινωνία </h1>
61.       <p class="text-center">Συμπληρώστε τα πεδία</p>
62.   </div>
63. </div>
64.
65. <!-- Middle content section -->
66. <div class="middle">
67.   <div class="container">
68.
69.     <!-- Form itself -->
70.       <form id="contactform" action="php/processForm.php" method="post">
71.             <table>
```

```
72.                 <tr>
73.                   <td><label for="name">Όνομα:</label></td>
74.                   <td><input type="text" id="name" name="name" /></td>
75.                 </tr>
76.                 <tr>
77.                   <td><label for="surname">Επώνυμο:</label></td>
78.                   <td><input type="text" id="surname" name="surname" /></td>
79.                 </tr>
80.                  <tr>
81.                   <td><label for="telephone">Τηλέφωνο:</label></td>
82.                   <td><input type="text" id="phone" name="phone" /></td>
83.                 </tr>
84.                 <tr>
85.                   <td><label for="email">Email:</label></td>
86.                   <td><input type="email" id="email" name="email" /></td>
87.                 </tr>
88.                 <tr>
89.                   <td><label for="message">Message:</label></td>
90.                   <td><textarea id="message" name="message" rows="5" cols="20">
```

IndexSite1.php

```
1.
    <!doctype html>
2.
3.  <?php
4.    session_start();
5.    if (!$_SESSION['Username']){
6.      header ("location:Login.php");
7.    }
8.    ?>
9.  <html>
10. <head>
11. <meta charset="utf-8">
12. <title>Home</title>
13. <meta name="viewport" content="width=device-width, initial-scale=1"<!--
    <link href="css/style.css" rel="stylesheet">
14.
15. <!-- Include jQuery and bootstrap JS plugins -->
16. <link href="css/bootstrap.min.css" rel="stylesheet">
17. <script src="js/jquery-2.1.0.min.js"></script>
18. <script src="js/bootstrap.min.js"></script>
19. </head>
20. <body>
21. <!--header-->
22. <nav class="navbar navbar-inverse">
23.   <div class="container-fluid">
24.     <div class="navbar-header"> <a class="navbar-
    brand" href="IndexSite1.php">GreekTrend</a> </div>
25.     <div>
26.       <button class="navbar-toggle" data-toggle="collapse" data-target=".navbar-
    collapse"> <span class="glyphicon glyphicon-align-justify"></span> </button>
27.       <nav class="navbar-collapse collapse" role="navigation">
```

```
28.         <ul class="nav navbar-nav">
29.           <li class="active">
30.           <li><a href="IndexSite1.php">Home</a></li>
31.           <li><a href="about.php">About</a></li>
32.           <li><a href="contact.php">Contact</a></li>
33.           <li><a href="trends.php">Trends</a></li>
34.           </li>
35.         </ul>
36.         <ul class="nav navbar-nav navbar-right">
37.           <li><a href="Logout.php"><span class="glyphicon glyphicon-log-
    in"></span>Logout</a></li>
38.         </ul>
39.       </nav>
40.     </div>
41.   </div>
42. </nav>
43. <!-- Site banner -->
44. <div class="banner">
45.   <div class="container" >
46.     <form name="search" method="post" action="show_test.php">
47.       <b>Seach for:</b>
48.       <input type="text" name="keyword" size="70"/>
49.     </form>
50.     <br>
51.   </div>
52. </div>
53.
54. <!-- Middle content section -->
55. <div class="middle">
56.   <div class="container">
57.     <h1 align="center">Υλοποίηση Web Εφαρμογής για την επισήμανση δημοφιλών θεμάτων σ
    ε μέσα κοινωνικής δικτύωσης βάσει τοποθεσίας </h1>
58.     <br>
59.     <br>
60.     <p align="center">Υλοποίηση  Χατζηγεωργίου Ηλίας </p>
61.   </div>
62. </div>
63. <!--footer-->
64.
65. <footer class="footer">
66.   <div class="container">
67.     <p class="text-muted">Πειραιάς 2016</p>
68.   </div>
69. </footer>
70. </body>
71. </html>
```

Login.php

```
1.
    <!doctype html>
2.  <?php
3.    session_start();
4.  ?>
5.  <html>
6.  <head>
7.  <meta charset="utf-8">
8.  <title>Login Page</title>
9.  <meta name="viewport" content="width=device-width, initial-scale=1">
10. <link href="css/bootstrap.min.css" rel="stylesheet">
11. <!--<link href="css/style.css" rel="stylesheet">-->
12. <link href="css/contact_form.css" rel="stylesheet">
13.
14. <!-- Include jQuery and bootstrap JS plugins
15. <script type="text/javascript" src="js/bootstrap.min.js"></script>
16. <script type="text/javascript" src="js/jquery.form.js"></script>-->
17. <script type="text/javascript" src="js/jquery-2.1.0.min.js"></script>
18. <script type="text/javascript" src="js/jquery.validate.min.js"></script>
19. <script type="text/javascript" src="js/contact.js"></script>
20. </head>
21.
22. <body>
23. <!--header-->
24. <nav class="navbar navbar-inverse">
25.   <div class="container-fluid">
26.     <div class="navbar-header"> <a class="navbar-
    brand" href="IndexSite1.php">GreekTrend</a> </div>
27.     <div>
28.       <button class="navbar-toggle" data-toggle="collapse" data-target=".navbar-
    collapse"> <span class="glyphicon glyphicon-align-justify"></span> </button>
29.       <nav class="navbar-collapse collapse" role="navigation">
30.         <ul class="nav navbar-nav">
31.           <li class="active">
32.           <li><a href="IndexSite1.php">Home</a></li>
33.           <li><a href="about.php">About</a></li>
34.           <li><a href="contact.php">Contact</a></li>
35.           </li>
36.         </ul>
37.         <ul class="nav navbar-nav navbar-right">
38.           <li><a href="Register.php"><span class="glyphicon glyphicon-
    user"></span> Sign Up</a></li>
39.           <li><a href="Login.php"><span class="glyphicon glyphicon-log-
    in"></span> Login</a></li>
40.         </ul>
41.       </nav>
42.     </div>
43.   </div>
44. </nav>
45. <!-- Site banner -->
46. <div class="banner">
47.   <div class="container" style=" padding:0px;" >
48.     <h1>Login</h1>
49.     <p class="text-center">Συμπληρώστε τα πεδία</p>
```

```
  </div>
</div>

<!-- Middle content section -->
<div class="middle">
  <div class="container">

    <!-- Form itself -->
    <form action="Login.php" method="post">
      <table>
        <tr>
          <td><label for="Username">Username:</label></td>
          <td><input type="text" id="Username" name="Username" /></td>
        </tr>
        <tr>
          <td><label for="Password">Password:</label></td>
          <td><input type="password" id="Password" name="Password" /></td>
        </tr>
        <tr>
          <td colspan="2" align="center"><input type="submit" name="login" value='Login' /></td
>
        </tr>
      </table>
    </form>
    <b>Not Registered yet?</b> <a href="Register.php"> Register Here</a> </div>
</div>
<!--footer-->
<footer class="footer">
  <div class="container">
    <p class="text-muted">Πειραιάς 2016</p>
  </div>
</footer>
</body>
</html>
<?php

  $user_db_con = mysqli_connect('localhost', 'root', 'root', 'cms');
     if (!$user_db_con) {
        die('Could not connect: ' . mysqli_connect_error());
 }
     if(isset($_POST['login'])){
        $username = $_POST['Username'];
        $password = $_POST['Password'];
        $query = "SELECT * FROM users WHERE Username = '$username' AND Password = '$
password'";
        $run = mysqli_query($user_db_con,$query);


     if(!$run || mysqli_num_rows($run)>0){

         $_SESSION['Username']= $username;

                    $adminCheckquery = "SELECT * FROM users WHERE Username = '" + $usern
ame + "' AND admin = 1";
                    $adminCheckqueryrun = mysqli_query($user_db_con,$adminCheckquery);

                    if(!$adminCheckqueryrun || mysqli_num_rows($adminCheckqueryrun) == 1){
                       echo "<script> alert('Logged as admin');</script>";
```

```
                    echo "<script>window.open('IndexSite1.php','_self')</script>";


                }

                else{


                    echo "<script>alert('Login Successfully Completed!')</script>";
                    echo "<script>window.open('IndexSite1.php','_self')</script>";
                }


            }

        }



    ?>
```

Logout.php

```
<?php

    session_start();
    session_destroy ();

    header ("location:Login.php");


    ?>
```

OathPhirehose.php

```
<?php
require_once('Phirehose.php');
/**
*
*
* @internal At time of writing thise overrides getAuthorizationHeader() from the parent class;
* all other functions are helper functions for that.
*/
```

```
9.  abstract class OauthPhirehose extends Phirehose
10. {
11.     protected $auth_method;
12.     /**
13.     * The Twitter consumer key. Get it from the application's page on Twitter.
14.     * If not set then the global define TWITTER_CONSUMER_KEY is used instead.
15.     */
16.     public $consumerKey=null;
17.     /**
18.     * The Twitter consumer secret. Get it from the application's page on Twitter.
19.     * If not set then the global define TWITTER_CONSUMER_SECRET is used instead.
20.     */
21.     public $consumerSecret=null;
22.     /**
23.     */
24.     protected function prepareParameters($method = null, $url = null,
25.         array $params)
26.     {
27.         if (emptyempty($method) || emptyempty($url))
28.             return false;
29.         $oauth['oauth_consumer_key'] = $this->consumerKey?$this-
    >consumerKey:TWITTER_CONSUMER_KEY;
30.         $oauth['oauth_nonce'] = md5(uniqid(rand(), true));
31.         $oauth['oauth_signature_method'] = 'HMAC-SHA1';
32.         $oauth['oauth_timestamp'] = time();
33.         $oauth['oauth_version'] = '1.0A';
34.         $oauth['oauth_token'] = $this->username;
35.         if (isset($params['oauth_verifier']))
36.         {
37.             $oauth['oauth_verifier'] = $params['oauth_verifier'];
38.             unset($params['oauth_verifier']);
39.         }
40.         // encode all oauth values
41.         foreach ($oauth as $k => $v)
42.             $oauth[$k] = $this->encode_rfc3986($v);
43.         // encode all non '@' params
44.         // keep sigParams for signature generation (exclude '@' params)
45.         // rename '@key' to 'key'
46.         $sigParams = array();
47.         $hasFile = false;
48.         if (is_array($params))
49.         {
50.             foreach ($params as $k => $v)
51.             {
52.                 if (strncmp('@', $k, 1) !== 0)
53.                 {
54.                     $sigParams[$k] = $this->encode_rfc3986($v);
55.                     $params[$k] = $this->encode_rfc3986($v);
56.                 }
57.                 else
58.                 {
59.                     $params[substr($k, 1)] = $v;
60.                     unset($params[$k]);
61.                     $hasFile = true;
62.                 }
63.             }
64.             if ($hasFile === true)
65.                 $sigParams = array();
```

```
66.         }
67.         $sigParams = array_merge($oauth, (array) $sigParams);
68.         // sorting
69.         ksort($sigParams);
70.         // signing
71.         $oauth['oauth_signature'] = $this->encode_rfc3986($this-
    >generateSignature($method, $url, $sigParams));
72.         return array('request' => $params, 'oauth' => $oauth);
73.     }
74.     protected function encode_rfc3986($string)
75.     {
76.         return str_replace('+', ' ', str_replace('%7E', '~', rawurlencode(($string))));
77.     }
78.     protected function generateSignature($method = null, $url = null,
79.         $params = null)
80.     {
81.         if (emptyempty($method) || emptyempty($url))
82.             return false;
83.         // concatenating and encode
84.         $concat = '';
85.         foreach ((array) $params as $key => $value)
86.             $concat .= "{$key}={$value}&";
87.         $concat = substr($concat, 0, -1);
88.         $concatenatedParams = $this->encode_rfc3986($concat);
89.         // normalize url
90.         $urlParts = parse_url($url);
91.         $scheme = strtolower($urlParts['scheme']);
92.         $host = strtolower($urlParts['host']);
93.         $port = isset($urlParts['port']) ? intval($urlParts['port']) : 0;
94.         $retval = strtolower($scheme) . '://' . strtolower($host);
95.         if (!emptyempty($port) && (($scheme === 'http' && $port != 80) || ($scheme === 'https' &
    & $port != 443)))
96.             $retval .= ":{$port}";
97.         $retval .= $urlParts['path'];
98.         if (!emptyempty($urlParts['query']))
99.             $retval .= "?{$urlParts['query']}";
100.                 $normalizedUrl = $this->encode_rfc3986($retval);
101.                 $method = $this->encode_rfc3986($method); // don't need this but why not?
102.                 $signatureBaseString = "{$method}&{$normalizedUrl}&{$concatenatedParams}";

103.                 # sign the signature string
104.                 $key = $this->encode_rfc3986($this->consumerSecret?$this-
    >consumerSecret:TWITTER_CONSUMER_SECRET) . '&' . $this->encode_rfc3986($this-
    >password);
105.                 return base64_encode(hash_hmac('sha1', $signatureBaseString, $key, true));
106.             }
107.             protected function getOAuthHeader($method, $url, $params = array())
108.             {
109.                 $params = $this->prepareParameters($method, $url, $params);
110.                 $oauthHeaders = $params['oauth'];
111.                 $urlParts = parse_url($url);
112.                 $oauth = 'OAuth realm="",';
113.                 foreach ($oauthHeaders as $name => $value)
114.                 {
115.                     $oauth .= "{$name}=\"{$value}\",";
116.                 }
117.                 $oauth = substr($oauth, 0, -1);
118.                 return $oauth;
```

```
119.            }
120.            /** Overrides base class function */
121.            protected function getAuthorizationHeader($url,$requestParams)
122.            {
123.                return $this->getOAuthHeader('POST', $url, $requestParams);
124.            }
125.        }
```

Phirehose.php

```
1.  <?php
2.  /**
3.   * A class that makes it easy to connect to and consume the Twitter stream via the Streaming
    API.
4.   *
5.   * Note: This is beta software - Please read the following carefully before using:
6.   *  - http://code.google.com/p/phirehose/wiki/Introduction
7.   *  - http://dev.twitter.com/pages/streaming_api
8.   * @author  Fenn Bailey <fenn.bailey@gmail.com>
9.   * @version 1.0RC
10.  */
11. abstract class Phirehose
12. {
13.
14.   /**
15.    * Class constants
16.    */
17.   const FORMAT_JSON      = 'json';
18.   const FORMAT_XML       = 'xml';
19.   const METHOD_FILTER    = 'filter';
20.   const METHOD_SAMPLE    = 'sample';
21.   const METHOD_RETWEET   = 'retweet';
22.   const METHOD_FIREHOSE  = 'firehose';
23.   const METHOD_LINKS     = 'links';
24.   const METHOD_USER      = 'user';  //See UserstreamPhirehose.php
25.   const METHOD_SITE      = 'site';  //See UserstreamPhirehose.php
26.   const EARTH_RADIUS_KM  = 6371;
27.   /**
28.   * @internal Moved from being a const to a variable, because some methods (user and site) n
    eed to change it.
29.   */
30.   protected $URL_BASE         = 'https://stream.twitter.com/1.1/statuses/';
31.
32.
33.   /**
34.    * Member Attribs
35.    */
36.   protected $username;
```

```
37.   protected $password;
38.   protected $method;
39.   protected $format;
40.   protected $count; //Can be -
      150,000 to 150,000. @see http://dev.twitter.com/pages/streaming_api_methods#count
41.   protected $followIds;
42.   protected $trackWords;
43.   protected $locationBoxes;
44.   protected $conn;
45.   protected $fdrPool;
46.   protected $buff;
47.   // State vars
48.   protected $filterChanged;
49.   protected $reconnect;
50.   /**
51.   * The number of tweets received per second in previous minute; calculated fresh
52.   * just before each call to statusUpdate()
53.   * I.e. if fewer than 30 tweets in last minute then this will be zero; if 30 to 90 then it
54.   * will be 1, if 90 to 150 then 2, etc.
55.   *
56.   * @var integer
57.   */
58.   protected $statusRate;
59.   protected $lastErrorNo;
60.   protected $lastErrorMsg;
61.   /**
62.   * Number of tweets received.
63.   *
64.   * Note: by default this is the sum for last 60 seconds, and is therefore
65.   * reset every 60 seconds.
66.   * To change this behaviour write a custom statusUpdate() function.
67.   *
68.   * @var integer
69.   */
70.   protected $statusCount=0;
71.   /**
72.   * The number of calls to $this->checkFilterPredicates().
73.   *
74.   * By default it is called every 5 seconds, so if doing statusUpdates every
75.   * 60 seconds and then resetting it, this will usually be 12.
76.   *
77.   * @var integer
78.   */
79.   protected $filterCheckCount=0;
80.   /**
81.   * Total number of seconds (fractional) spent in the enqueueStatus() calls (i.e. the customized

82.   * function that handles each received tweet).
83.   *
84.   * @var float
85.   */
86.   protected $enqueueSpent=0;
87.   /**
88.   * Total number of seconds (fractional) spent in the checkFilterPredicates() calls
89.   *
90.   * @var float
91.   */
92.   protected $filterCheckSpent=0;
```

```
93.  /**
94.  * Number of seconds since the last tweet arrived (or the keep-alive newline)
95.  *
96.  * @var integer
97.  */
98.  protected $idlePeriod=0;
99.  /**
100.          * The maximum value $this->idlePeriod has reached.
101.          *
102.          * @var integer
103.          */
104.          protected $maxIdlePeriod=0;
105.          /**
106.          * Time spent on each call to enqueueStatus() (i.e. average time spent, in millisecond
      s,
107.          * spent processing received tweet).
108.          *
109.          * Simply: enqueueSpent divided by statusCount
110.          * Note: by default, calculated fresh for past 60 seconds, every 60 seconds.
111.          *
112.          * @var float
113.          */
114.          protected $enqueueTimeMS=0;
115.          /**
116.          * Like $enqueueTimeMS but for the checkFilterPredicates() function.
117.          * @var float
118.          */
119.          protected $filterCheckTimeMS=0;
120.          /**
121.          * Seconds since the last call to statusUpdate()
122.          *
123.          * Reset to zero after each call to statusUpdate()
124.          * Highest value it should ever reach is $this->avgPeriod
125.          *
126.          * @var integer
127.          */
128.          protected $avgElapsed=0;
129.          // Config type vars - override in subclass if desired
130.          protected $connectFailuresMax = 20;
131.          protected $connectTimeout = 5;
132.          protected $readTimeout = 5;
133.          protected $idleReconnectTimeout = 90;
134.          protected $avgPeriod = 60;
135.          protected $status_length_base = 10;
136.          protected $userAgent      = 'Phirehose/1.0RC +https://github.com/fennb/phirehose';

137.          protected $filterCheckMin = 5;
138.          protected $filterUpdMin   = 120;
139.          protected $tcpBackoff     = 1;
140.          protected $tcpBackoffMax  = 16;
141.          protected $httpBackoff  = 10;
142.          protected $httpBackoffMax  = 240;
143.          protected $hostPort = 80;
144.          protected $secureHostPort = 443;
145.
146.          /**
147.           * Create a new Phirehose object attached to the appropriate twitter stream method.
```

```
148.             * Methods are: METHOD_FIREHOSE, METHOD_RETWEET, METHOD_SAMPLE,
     METHOD_FILTER, METHOD_LINKS, METHOD_USER, METHOD_SITE. Note: the method
     might cause the use of a different endpoint URL.
149.             * Formats are: FORMAT_JSON, FORMAT_XML
150.             * @see Phirehose::METHOD_SAMPLE
151.             * @see Phirehose::FORMAT_JSON
152.             *
153.             * @param string $username Any twitter username. When using oAuth, this is the 'oa
     uth_token'.
154.             * @param string $password Any twitter password. When using oAuth this is you oA
     uth secret.
155.             * @param string $method
156.             * @param string $format
157.             *
158.             * @todo I've kept the "/2/" at the end of the URL for user streams, as that is what
159.             *    was there before AND it works for me! But the official docs say to use /1.1/
160.             *    so that is what I have used for site.
161.             *     https://dev.twitter.com/docs/api/1.1/get/user
162.             *
163.             * @todo Shouldn't really hard-code URL strings in this function.
164.             */
165.             public function __construct($username, $password, $method = Phirehose::METH
     OD_SAMPLE, $format = self::FORMAT_JSON, $lang = FALSE)
166.             {
167.               $this->username = $username;
168.               $this->password = $password;
169.               $this->method = $method;
170.               $this->format = $format;
171.               $this->lang = $lang;
172.              switch($method){
173.                  case self::METHOD_USER:$this-
     >URL_BASE = 'https://userstream.twitter.com/1.1/';break;
174.                  case self::METHOD_SITE:$this-
     >URL_BASE = 'https://sitestream.twitter.com/1.1/';break;
175.                  default:break;  //Stick to the default
176.                  }
177.             }
178.
179.             /**
180.              * Returns public statuses from or in reply to a set of users. Mentions ("Hello @user!"
     ) and implicit replies
181.              * ("@user Hello!" created without pressing the reply button) are not matched. It is up
      to you to find the integer
182.              * IDs of each twitter user.
183.              * Applies to: METHOD_FILTER
184.              *
185.              * @param array $userIds Array of Twitter integer userIDs
186.              */
187.             public function setFollow($userIds)
188.             {
189.               $userIds = ($userIds === NULL) ? array() : $userIds;
190.               sort($userIds); // Non-optimal but necessary
191.               if ($this->followIds != $userIds) {
192.                 $this->filterChanged = TRUE;
193.               }
194.               $this->followIds = $userIds;
195.             }
196.
```

```
197.            /**
198.             * Returns an array of followed Twitter userIds (integers)
199.             *
200.             * @return array
201.             */
202.            public function getFollow()
203.            {
204.              return $this->followIds;
205.            }
206.
207.            /**
208.             * Specifies keywords to track. Track keywords are case-
    insensitive logical ORs. Terms are exact-matched, ignoring
209.             * punctuation. Phrases, keywords with spaces, are not supported. Queries are subje
    ct to Track Limitations.
210.             * Applies to: METHOD_FILTER
211.             *
212.             * See: http://apiwiki.twitter.com/Streaming-API-Documentation#TrackLimiting
213.             *
214.             * @param array $trackWords
215.             */
216.            public function setTrack(array $trackWords)
217.            {
218.              $trackWords = ($trackWords === NULL) ? array() : $trackWords;
219.              sort($trackWords); // Non-optimal, but necessary
220.              if ($this->trackWords != $trackWords) {
221.                $this->filterChanged = TRUE;
222.              }
223.              $this->trackWords = $trackWords;
224.            }
225.
226.            /**
227.             * Returns an array of keywords being tracked
228.             *
229.             * @return array
230.             */
231.            public function getTrack()
232.            {
233.              return $this->trackWords;
234.            }
235.
236.            /**
237.             * Specifies a set of bounding boxes to track as an array of 4 element lon/lat pairs de
    noting <south-west point>,
238.             * <north-
    east point>. Only tweets that are both created using the Geotagging API and are placed from
    within a tracked
239.             * bounding box will be included in the stream. The user's location field is not used to
     filter tweets. Bounding boxes
240.             * are logical ORs and must be less than or equal to 1 degree per side. A locations p
    arameter may be combined with
241.             * track parameters, but note that all terms are logically ORd.
242.             *
243.             * NOTE: The argument order is Longitude/Latitude (to match the Twitter API and Ge
    oJSON specifications).
244.             *
245.             * Applies to: METHOD_FILTER
246.             *
```

```
247.              * See: http://apiwiki.twitter.com/Streaming-API-Documentation#locations
248.              *
249.              * Eg:
250.              *  setLocations(array(
251.              *      array(-122.75, 36.8, -121.75, 37.8), // San Francisco
252.              *      array(-74, 40, -73, 41),             // New York
253.              *  ));
254.              *
255.              * @param array $boundingBoxes
256.              */
257.             public function setLocations($boundingBoxes)
258.             {
259.               $boundingBoxes = ($boundingBoxes === NULL) ? array() : $boundingBoxes;
260.               sort($boundingBoxes); // Non-optimal, but necessary
261.               // Flatten to single dimensional array
262.               $locationBoxes = array();
263.               foreach ($boundingBoxes as $boundingBox) {
264.                 // Sanity check
265.                 if (count($boundingBox) != 4) {
266.                   // Invalid - Not much we can do here but log error
267.                   $this-
    >log('Invalid location bounding box: [' . implode(', ', $boundingBox) . ']','error');
268.                   return FALSE;
269.                 }
270.                 // Append this lat/lon pairs to flattened array
271.                 $locationBoxes = array_merge($locationBoxes, $boundingBox);
272.               }
273.               // If it's changed, make note
274.               if ($this->locationBoxes != $locationBoxes) {
275.                 $this->filterChanged = TRUE;
276.               }
277.               // Set flattened value
278.               $this->locationBoxes = $locationBoxes;
279.             }
280.
281.             /**
282.              * Returns an array of 4 element arrays that denote the monitored location bounding
    boxes for tweets using the
283.              * Geotagging API.
284.              *
285.              * @see setLocations()
286.              * @return array
287.              */
288.             public function getLocations() {
289.               if ($this->locationBoxes == NULL) {
290.                 return NULL;
291.               }
292.               $locationBoxes = $this->locationBoxes; // Copy array
293.               $ret = array();
294.               while (count($locationBoxes) >= 4) {
295.                 $ret[] = array_splice($locationBoxes, 0, 4); // Append to ret array in blocks of 4
296.               }
297.               return $ret;
298.             }
299.
300.             /**
301.              * Convenience method that sets location bounding boxes by an array of lon/lat/radiu
    s sets, rather than manually
```

```
302.              * specified bounding boxes. Each array element should contain 3 element subarray
    containing a latitude, longitude and
303.              * radius. Radius is specified in kilometers and is approximate (as boxes are square).

304.              *
305.              * NOTE: The argument order is Longitude/Latitude (to match the Twitter API and Ge
    oJSON specifications).
306.              *
307.              * Eg:
308.              *  setLocationsByCircle(array(
309.              *      array(144.9631, -37.8142, 30), // Melbourne, 3km radius
310.              *      array(-0.1262, 51.5001, 25),   // London 10km radius
311.              *  ));
312.              *
313.              *
314.              * @see setLocations()
315.              * @param array
316.              */
317.             public function setLocationsByCircle($locations) {
318.               $boundingBoxes = array();
319.               foreach ($locations as $locTriplet) {
320.                 // Sanity check
321.                 if (count($locTriplet) != 3) {
322.                   // Invalid - Not much we can do here but log error
323.                   $this-
    >log('Invalid location triplet for ' . __METHOD__ . ': [' . implode(', ', $locTriplet) . ']','error');
324.                   return FALSE;
325.                 }
326.                 list($lon, $lat, $radius) = $locTriplet;
327.                 // Calc bounding boxes
328.                 $maxLat = round($lat + rad2deg($radius / self::EARTH_RADIUS_KM), 2);
329.                 $minLat = round($lat - rad2deg($radius / self::EARTH_RADIUS_KM), 2);
330.                 // Compensate for degrees longitude getting smaller with increasing latitude
331.                 $maxLon = round($lon + rad2deg($radius / self::EARTH_RADIUS_KM / cos(deg2r
    ad($lat))), 2);
332.                 $minLon = round($lon -
     rad2deg($radius / self::EARTH_RADIUS_KM / cos(deg2rad($lat))), 2);
333.                 // Add to bounding box array
334.                 $boundingBoxes[] = array($minLon, $minLat, $maxLon, $maxLat);
335.                 // Debugging is handy
336.                 $this->log('Resolved location circle [' . $lon . ', ' . $lat . ', r: ' . $radius . '] -
    > bbox: [' . $minLon .
337.                   ', ' . $minLat . ', ' . $maxLon . ', ' . $maxLat . ']');
338.               }
339.               // Set by bounding boxes
340.               $this->setLocations($boundingBoxes);
341.             }
342.
343.             /**
344.              * Sets the number of previous statuses to stream before transitioning to the live stre
    am. Applies only to firehose
345.              * and filter + track methods. This is generally used internally and should not be need
    ed by client applications.
346.              * Applies to: METHOD_FILTER, METHOD_FIREHOSE, METHOD_LINKS
347.              *
348.              * @param integer $count
349.              */
350.             public function setCount($count)
```

```
351.            {
352.              $this->count = $count;
353.            }
354.            /**
355.             * Restricts tweets to the given language, given by an ISO 639-
    1 code (http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes).
356.             *
357.             * @param string $lang
358.             */
359.            public function setLang($lang)
360.            {
361.              $this->lang = $lang;
362.            }
363.            /**
364.             * Returns the ISO 639-
    1 code formatted language string of the current setting. (http://en.wikipedia.org/wiki/List_of_IS
    O_639-1_codes).
365.             *
366.             * @param string $lang
367.             */
368.            public function getLang()
369.            {
370.              return $this->lang;
371.            }
372.
373.            /**
374.             * Connects to the stream API and consumes the stream. Each status update in the s
    tream will cause a call to the
375.             * handleStatus() method.
376.             *
377.             * Note: in normal use this function does not return.
378.             * If you pass $reconnect as false, it will still not return in normal use: it will only retur
    n
379.             *   if the remote side (Twitter) close the socket. (Or the socket dies for some other ex
    ternal reason.)
380.             *
381.             * @see handleStatus()
382.             * @param boolean $reconnect Reconnects as per recommended
383.             * @throws ErrorException
384.             */
385.            public function consume($reconnect = TRUE)
386.            {
387.              // Persist connection?
388.              $this->reconnect = $reconnect;
389.
390.              // Loop indefinitely based on reconnect
391.              do {
392.
393.                // (Re)connect
394.                $this->reconnect();
395.
396.                // Init state
397.                $lastAverage = $lastFilterCheck = $lastFilterUpd = $lastStreamActivity = time();
398.                $fdw = $fde = NULL; // Placeholder write/error file descriptors for stream_select
399.
400.                // We use a blocking-
    select with timeout, to allow us to continue processing on idle streams
```

```
401.                //TODO: there is a bug lurking here. If $this->conn is fine, but $numChanged returns zero, because readTimeout was
402.                //    reached, then we should consider we still need to call statusUpdate() every 60 seconds, etc.
403.                //    ($this->readTimeout is 5 seconds.) This can be quite annoying. E.g. Been getting data regularly for 5 5 seconds,
404.                //    then it goes quiet for just 10 or so seconds. It is now 65 seconds since last call to statusUpdate() has been
405.                //    called, which might mean a monitoring system kills the script assuming it has died.
406.                while ($this->conn !== NULL && !feof($this->conn) &&
407.                  ($numChanged = stream_select($this->fdrPool, $fdw, $fde, $this->readTimeout)) !== FALSE) {
408.                  /* Unfortunately, we need to do a safety check for dead twitter streams - This seems to be able to happen where
409.                   * you end up with a valid connection, but NO tweets coming along the wire (or keep alives). The below guards
410.                   * against this.
411.                   */
412.                  if ((time() - $lastStreamActivity) > $this->idleReconnectTimeout) {
413.                    $this->log('Idle timeout: No stream activity for > ' . $this->idleReconnectTimeout . ' seconds. ' .
414.                      ' Reconnecting.','info');
415.                    $this->reconnect();
416.                    $lastStreamActivity = time();
417.                    continue;
418.                  }
419.                  // Process stream/buffer
420.                  $this->fdrPool = array($this->conn); // Must reassign for stream_select()
421.                  //Get a full HTTP chunk.
422.                  //NB. This is a tight loop, not using stream_select.
423.                  //NB. If that causes problems, then perhaps put something to give up after say trying for 10 seconds? (but
424.                  //   the stream will be all messed up, so will need to do a reconnect).
425.                  $chunk_info=trim(fgets($this->conn)); //First line is hex digits giving us the length
426.                  if($chunk_info=='')continue;    //Usually indicates a time-out. If we wanted to be sure,
427.                    //then stream_get_meta_data($this->conn)['timed_out']==1.  (We could instead
428.                    //   look at the 'eof' member, which appears to be boolean false if just a time-out.)
429.                    //TODO: need to consider calling statusUpdate() every 60 seconds, etc.
430.                  // Track maximum idle period
431.                  // (We got start of an HTTP chunk, this is stream activity)
432.                  $this->idlePeriod = (time() - $lastStreamActivity);
433.                  $this->maxIdlePeriod = ($this->idlePeriod > $this->maxIdlePeriod) ? $this->idlePeriod : $this->maxIdlePeriod;
434.                  $lastStreamActivity = time();
435.                  //Append one HTTP chunk to $this->buff
436.                  $len=hexdec($chunk_info);   //$len includes the \r\n at the end of the chunk (despite what wikipedia says)
437.                  //TODO: could do a check for data corruption here. E.g. if($len>100000){...}
438.                  $s='';
439.                  $len+=2;    //For the \r\n at the end of the chunk
440.                  while(!feof($this->conn)){
441.                    $s.=fread($this->conn,$len-strlen($s));
```

```
442.                 if(strlen($s)>=$len)break;  //TODO: Can never be >$len, only ==$len??
443.                 }
444.               $this->buff.=substr($s,0,-2);   //This is our HTTP chunk
445.               //Process each full tweet inside $this->buff
446.               while(1){
447.                 $eol = strpos($this->buff,"\r\n");  //Find next line ending
448.                 if($eol===0) {  // if 0, then buffer starts with "\r\n", so trim it and loop again
449.                   $this->buff = substr($this->buff,$eol+2);  // remove the "\r\n" from line start
450.                   continue; // loop again
451.                 }
452.                 if($eol===false)break;   //Time to get more data
453.                 $enqueueStart = microtime(TRUE);
454.                 $this->enqueueStatus(substr($this->buff,0,$eol));
455.                 $this->enqueueSpent += (microtime(TRUE) - $enqueueStart);
456.                 $this->statusCount++;
457.                 $this->buff = substr($this->buff,$eol+2);    //+2 to allow for the \r\n
458.               }
459.               //NOTE: if $this->buff is not empty, it is tempting to go round and get the next HTTP chunk, as
460.               //  we know there is data on the incoming stream. However, this could mean the below functions (heartbeat
461.               //  and statusUpdate) *never* get called, which would be bad.
462.               // Calc counter averages
463.               $this->avgElapsed = time() - $lastAverage;
464.               if ($this->avgElapsed >= $this->avgPeriod) {
465.                 $this->statusRate = round($this->statusCount / $this->avgElapsed, 0);          // Calc tweets-per-second
466.                 // Calc time spent per enqueue in ms
467.                 $this->enqueueTimeMS = ($this->statusCount > 0) ?
468.                   round($this->enqueueSpent / $this->statusCount * 1000, 2) : 0;
469.                 // Calc time spent total in filter predicate checking
470.                 $this->filterCheckTimeMS = ($this->filterCheckCount > 0) ?
471.                   round($this->filterCheckSpent / $this->filterCheckCount * 1000, 2) : 0;
472.                 $this->heartbeat();
473.                 $this->statusUpdate();
474.                 $lastAverage = time();
475.               }
476.               // Check if we're ready to check filter predicates
477.               if ($this->method == self::METHOD_FILTER && (time() - $lastFilterCheck) >= $this->filterCheckMin) {
478.                 $this->filterCheckCount++;
479.                 $lastFilterCheck = time();
480.                 $filterCheckStart = microtime(TRUE);
481.                 $this->checkFilterPredicates(); // This should be implemented in subclass if required
482.                 $this->filterCheckSpent +=  (microtime(TRUE) - $filterCheckStart);
483.               }
484.               // Check if filter is ready + allowed to be updated (reconnect)
485.               if ($this->filterChanged == TRUE && (time() - $lastFilterUpd) >= $this->filterUpdMin) {
486.                 $this->log('Reconnecting due to changed filter predicates.','info');
487.                 $this->reconnect();
488.                 $lastFilterUpd = time();
489.               }
490.
491.             } // End while-stream-activity
492.             if (function_exists('pcntl_signal_dispatch')) {
493.               pcntl_signal_dispatch();
```

```
494.                 }
495.                 // Some sort of socket error has occured
496.                 $this->lastErrorNo = is_resource($this->conn) ? @socket_last_error($this-
    >conn) : NULL;
497.                 $this->lastErrorMsg = ($this->lastErrorNo > 0) ? @socket_strerror($this-
    >lastErrorNo) : 'Socket disconnected';
498.                 $this->log('Phirehose connection error occured: ' . $this->lastErrorMsg,'error');
499.                 // Reconnect
500.               } while ($this->reconnect);
501.               // Exit
502.               $this->log('Exiting.');
503.
504.             }
505.             /**
506.              * Called every $this-
    >avgPeriod (default=60) seconds, and this default implementation
507.              * calculates some rates, logs them, and resets the counters.
508.              */
509.             protected function statusUpdate()
510.             {
511.                 $this->log('Consume rate: ' . $this->statusRate . ' status/sec (' . $this-
    >statusCount . ' total), avg ' .
512.                   'enqueueStatus(): ' . $this-
    >enqueueTimeMS . 'ms, avg checkFilterPredicates(): ' . $this->filterCheckTimeMS . 'ms (' .
513.                   $this->filterCheckCount . ' total) over ' . $this-
    >avgElapsed . ' seconds, max stream idle period: ' .
514.                    $this->maxIdlePeriod . ' seconds.');
515.                 // Reset
516.                 $this->statusCount = $this->filterCheckCount = $this->enqueueSpent = 0;
517.                 $this->filterCheckSpent = $this->idlePeriod = $this->maxIdlePeriod = 0;
518.             }
519.
520.             /**
521.              * Returns the last error message (TCP or HTTP) that occured with the streaming AP
    I or client. State is cleared upon
522.              * successful reconnect
523.              * @return string
524.              */
525.             public function getLastErrorMsg()
526.             {
527.               return $this->lastErrorMsg;
528.             }
529.
530.             /**
531.              * Returns the last error number that occured with the streaming API or client. Numb
    ers correspond to either the
532.              * fsockopen() error states (in the case of TCP errors) or HTTP error codes from Twit
    ter (in the case of HTTP errors).
533.              *
534.              * State is cleared upon successful reconnect.
535.              *
536.              * @return string
537.              */
538.             public function getLastErrorNo()
539.             {
540.               return $this->lastErrorNo;
541.             }
542.
```

```
543.
544.            /**
545.             * Connects to the stream URL using the configured method.
546.             * @throws ErrorException
547.             */
548.            protected function connect()
549.            {
550.              // Init state
551.              $connectFailures = 0;
552.              $tcpRetry = $this->tcpBackoff / 2;
553.              $httpRetry = $this->httpBackoff / 2;
554.              // Keep trying until connected (or max connect failures exceeded)
555.              do {
556.                // Check filter predicates for every connect (for filter method)
557.                if ($this->method == self::METHOD_FILTER) {
558.                  $this->checkFilterPredicates();
559.                }
560.
561.                // Construct URL/HTTP bits
562.                $url = $this->URL_BASE . $this->method . '.' . $this->format;
563.                $urlParts = parse_url($url);
564.
565.                // Setup params appropriately
566.                $requestParams=array();
567.
568.                //$requestParams['delimited'] = 'length';    //No, we don't want this any more
569.                // Setup the language of the stream
570.                if($this->lang) {
571.                  $requestParams['language'] = $this->lang;
572.                }
573.
574.                // Filter takes additional parameters
575.                if (($this->method == self::METHOD_FILTER || $this-
    >method == self::METHOD_USER) && count($this->trackWords) > 0) {
576.                  $requestParams['track'] = implode(',', $this->trackWords);
577.                }
578.                if ( ($this->method == self::METHOD_FILTER || $this-
    >method == self::METHOD_SITE)
579.                    && count($this->followIds) > 0) {
580.                  $requestParams['follow'] = implode(',', $this->followIds);
581.                }
582.                if ($this->method == self::METHOD_FILTER && count($this-
    >locationBoxes) > 0) {
583.                  $requestParams['locations'] = implode(',', $this->locationBoxes);
584.                }
585.                if ($this->count <> 0) {
586.                  $requestParams['count'] = $this->count;
587.                }
588.
589.                // Debugging is useful
590.                $this-
    >log('Connecting to twitter stream: ' . $url . ' with params: ' . str_replace("\n", '',
591.                  var_export($requestParams, TRUE)));
592.
593.                /**
594.                 * Open socket connection to make POST request. It'd be nice to use stream_cont
    ext_create with the native
```

```
595.                 * HTTP transport but it hides/abstracts too many required bits (like HTTP error res
     ponses).
596.                 */
597.                $errNo = $errStr = NULL;
598.                $scheme = ($urlParts['scheme'] == 'https') ? 'ssl://' : 'tcp://';
599.                $port = ($urlParts['scheme'] == 'https') ? $this->secureHostPort : $this->hostPort;
600.
601.                $this-
     >log("Connecting to {$scheme}{$urlParts['host']}, port={$port}, connectTimeout={$this-
     >connectTimeout}");
602.
603.                @$this-
     >conn = fsockopen($scheme . $urlParts['host'], $port, $errNo, $errStr, $this-
     >connectTimeout);
604.
605.                // No go - handle errors/backoff
606.                if (!$this->conn || !is_resource($this->conn)) {
607.                  $this->lastErrorMsg = $errStr;
608.                  $this->lastErrorNo = $errNo;
609.                  $connectFailures++;
610.                  if ($connectFailures > $this->connectFailuresMax) {
611.                    $msg = 'TCP failure limit exceeded with ' . $connectFailures . ' failures. Last err
     or: ' . $errStr;
612.                    $this->log($msg,'error');
613.                    throw new PhirehoseConnectLimitExceeded($msg, $errNo); // Throw an excep
     tion for other code to handle
614.                  }
615.                  // Increase retry/backoff up to max
616.                  $tcpRetry = ($tcpRetry < $this->tcpBackoffMax) ? $tcpRetry * 2 : $this-
     >tcpBackoffMax;
617.                  $this->log('TCP failure ' . $connectFailures . ' of ' . $this-
     >connectFailuresMax . ' connecting to stream: ' .
618.                    $errStr . ' (' . $errNo . '). Sleeping for ' . $tcpRetry . ' seconds.','info');
619.                  sleep($tcpRetry);
620.                  continue;
621.                }
622.
623.                // TCP connect OK, clear last error (if present)
624.                $this->log('Connection established to ' . $urlParts['host']);
625.                $this->lastErrorMsg = NULL;
626.                $this->lastErrorNo = NULL;
627.
628.                // If we have a socket connection, we can attempt a HTTP request -
       Ensure blocking read for the moment
629.                stream_set_blocking($this->conn, 1);
630.
631.                // Encode request data
632.                $postData = http_build_query($requestParams, NULL, '&');
633.                $postData = str_replace('+','%20',$postData); //Change it from RFC1738 to RFC3
     986 (see
634.                    //enc_type parameter in http://php.net/http_build_query and note that enc_type
     is
635.                    //not available as of php 5.3)
636.                $authCredentials = $this->getAuthorizationHeader($url,$requestParams);
637.
638.                // Do it
639.                $s = "POST " . $urlParts['path'] . " HTTP/1.1\r\n";
640.                $s.= "Host: " . $urlParts['host'] . ':' . $port . "\r\n";
```

```
641.                $s .= "Connection: Close\r\n";
642.                $s.= "Content-type: application/x-www-form-urlencoded\r\n";
643.                $s.= "Content-length: " . strlen($postData) . "\r\n";
644.                $s.= "Accept: */*\r\n";
645.                $s.= 'Authorization: ' . $authCredentials . "\r\n";
646.                $s.= 'User-Agent: ' . $this->userAgent . "\r\n";
647.                $s.= "\r\n";
648.                $s.= $postData . "\r\n";
649.                $s.= "\r\n";
650.
651.                fwrite($this->conn, $s);
652.                $this->log($s);
653.
654.                // First line is response
655.                list($httpVer, $httpCode, $httpMessage) = preg_split('/\s+/', trim(fgets($this-
    >conn, 1024)), 3);
656.
657.                // Response buffers
658.                $respHeaders = $respBody = '';
659.                $isChunking = false;
660.                // Consume each header response line until we get to body
661.                while ($hLine = trim(fgets($this->conn, 4096))) {
662.                  $respHeaders .= $hLine."\n";
663.                  if(strtolower($hLine) == 'transfer-encoding: chunked') $isChunking = true;
664.                }
665.
666.                // If we got a non-200 response, we need to backoff and retry
667.                if ($httpCode != 200) {
668.                  $connectFailures++;
669.
670.                  // Twitter will disconnect on error, but we want to consume the rest of the respons
    e body (which is useful)
671.                  //TODO: this might be chunked too? In which case this contains some bad chara
    cters??
672.                  while ($bLine = trim(fgets($this->conn, 4096))) {
673.                    $respBody .= $bLine;
674.                  }
675.
676.                  // Construct error
677.                  $errStr = 'HTTP ERROR ' . $httpCode . ': ' . $httpMessage . ' (' . $respBody . ')';
678.
679.                  // Set last error state
680.                  $this->lastErrorMsg = $errStr;
681.                  $this->lastErrorNo = $httpCode;
682.
683.                  // Have we exceeded maximum failures?
684.                  if ($connectFailures > $this->connectFailuresMax) {
685.                    $msg = 'Connection failure limit exceeded with ' . $connectFailures . ' failures. L
    ast error: ' . $errStr;
686.                    $this->log($msg,'error');
687.                    throw new PhirehoseConnectLimitExceeded($msg, $httpCode); // We eventual
    ly throw an exception for other code to handle
688.                  }
689.                  // Increase retry/backoff up to max
690.                  $httpRetry = ($httpRetry < $this->httpBackoffMax) ? $httpRetry * 2 : $this-
    >httpBackoffMax;
691.                  $this->log('HTTP failure ' . $connectFailures . ' of ' . $this-
    >connectFailuresMax . ' connecting to stream: ' .
```

```
692.                     $errStr . '. Sleeping for ' . $httpRetry . ' seconds.','info');
693.                   sleep($httpRetry);
694.                   continue;
695.
696.                 } // End if not http 200
697.                 else{
698.                   if(!$isChunking)throw new Exception("Twitter did not send a chunking header. Is t
      his really HTTP/1.1? Here are headers:\n$respHeaders");   //TODO: rather crude!
699.                 }
700.                 // Loop until connected OK
701.               } while (!is_resource($this->conn) || $httpCode != 200);
702.
703.               // Connected OK, reset connect failures
704.               $connectFailures = 0;
705.               $this->lastErrorMsg = NULL;
706.               $this->lastErrorNo = NULL;
707.
708.               // Switch to non-blocking to consume the stream (important)
709.               stream_set_blocking($this->conn, 0);
710.
711.               // Connect always causes the filterChanged status to be cleared
712.               $this->filterChanged = FALSE;
713.
714.               // Flush stream buffer & (re)assign fdrPool (for reconnect)
715.               $this->fdrPool = array($this->conn);
716.               $this->buff = '';
717.
718.             }
719.               protected function getAuthorizationHeader($url,$requestParams)
720.               {
721.                   throw new Exception("Basic auth no longer works with Twitter. You must derive
      from OauthPhirehose, not directly from the Phirehose class.");
722.                   $authCredentials = base64_encode($this->username . ':' . $this->password);
723.                   return "Basic: ".$authCredentials;
724.               }
725.
726.             /**
727.               * Method called as frequently as practical (every 5+ seconds) that is responsible for
       checking if filter predicates
728.               * (ie: track words or follow IDs) have changed. If they have, they should be set using
       the setTrack() and setFollow()
729.               * methods respectively within the overridden implementation.
730.               *
731.               * Note that even if predicates are changed every 5 seconds, an actual reconnect will
       not happen more frequently than
732.               * every 2 minutes (as per Twitter Streaming API documentation).
733.               *
734.               * Note also that this method is called upon every connect attempt, so if your predica
      tes are causing connection
735.               * errors, they should be checked here and corrected.
736.               *
737.               * This should be implemented/overridden in any subclass implementing the FILTER
       method.
738.               *
739.               * @see setTrack()
740.               * @see setFollow()
741.               * @see Phirehose::METHOD_FILTER
742.               */
```

```
743.            protected function checkFilterPredicates()
744.            {
745.              // Override in subclass
746.            }
747.
748.            /**
749.             * Basic log function that outputs logging to the standard error_log() handler. This sh
    ould generally be overridden
750.             * to suit the application environment.
751.             *
752.             * @see error_log()
753.             * @param string $messages
754.             * @param String $level 'error', 'info', 'notice'. Defaults to 'notice', so you should set t
    his
755.             *     parameter on the more important error messages.
756.             *     'info' is used for problems that the class should be able to recover from automati
    cally.
757.             *     'error' is for exceptional conditions that may need human intervention. (For insta
    nce, emailing
758.             *         them to a system administrator may make sense.)
759.             */
760.            protected function log($message,$level='notice')
761.            {
762.              @error_log('Phirehose: ' . $message, 0);
763.            }
764.            /**
765.             * Performs forcible disconnect from stream (if connected) and cleanup.
766.             */
767.            protected function disconnect()
768.            {
769.              if (is_resource($this->conn)) {
770.                $this->log('Closing Phirehose connection.');
771.                fclose($this->conn);
772.              }
773.              $this->conn = NULL;
774.              $this->reconnect = FALSE;
775.            }
776.
777.            /**
778.             * Reconnects as quickly as possible. Should be called whenever a reconnect is requ
    ired rather that connect/disconnect
779.             * to preserve streams reconnect state
780.             */
781.            private function reconnect()
782.            {
783.              $reconnect = $this->reconnect;
784.              $this->disconnect(); // Implicitly sets reconnect to FALSE
785.              $this->reconnect = $reconnect; // Restore state to prev
786.              $this->connect();
787.            }
788.
789.            /**
790.             * This is the one and only method that must be implemented additionally. As per the
    streaming API documentation,
791.             * statuses should NOT be processed within the same process that is performing coll
    ection
792.             *
793.             * @param string $status
```

```
794.          */
795.         abstract public function enqueueStatus($status);
796.         /**
797.          * Reports a periodic heartbeat. Keep execution time minimal.
798.          *
799.          * @return NULL
800.          */
801.         public function heartbeat() {}
802.
803.         /**
804.          * Set host port
805.          *
806.          * @param string $host
807.          * @return void
808.          */
809.         public function setHostPort($port)
810.         {
811.           $this->hostPort = $port;
812.         }
813.
814.         /**
815.          * Set secure host port
816.          *
817.          * @param int $port
818.          * @return void
819.          */
820.         public function setSecureHostPort($port)
821.         {
822.           $this->secureHostPort = $port;
823.         }
824.       } // End of class
825.       class PhirehoseException extends Exception {}
826.       class PhirehoseNetworkException extends PhirehoseException {}
827.       class PhirehoseConnectLimitExceeded extends PhirehoseException {}
```

Register.php

```
1.  <!doctype html>
2.
3.  <html>
4.  <head>
5.  <meta charset="utf-8">
6.  <title>Register</title>
7.  <meta name="viewport" content="width=device-width, initial-scale=1">
8.  <link href="css/bootstrap.min.css" rel="stylesheet">
9.  <!--<link href="css/style.css" rel="stylesheet">-->
10. <link href="css/contact_form.css" rel="stylesheet">
11.
12.
13.
14. <!-- Include jQuery and bootstrap JS plugins
15. <script type="text/javascript" src="js/bootstrap.min.js"></script>
```

```
16. <script type="text/javascript" src="js/jquery.form.js"></script>-->
17. <script type="text/javascript" src="js/jquery-2.1.0.min.js"></script>
18.
19. <script type="text/javascript" src="js/jquery.validate.min.js"></script>
20. <script type="text/javascript" src="js/contact.js"></script>
21. </head>
22.
23. <body>
24. <!--header-->
25. <nav class="navbar navbar-inverse">
26.   <div class="container-fluid">
27.     <div class="navbar-header">
28.       <a class="navbar-brand" href="IndexSite1.php">GreekTrend</a>
29.     </div>
30.     <div>
31.     <button class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
32.           <span class="glyphicon glyphicon-align-justify"></span>
33.       </button>
34.       <nav class="navbar-collapse collapse" role="navigation">
35.       <ul class="nav navbar-nav">
36.       <li class="active">
37.       <li><a href="IndexSite1.php">Home</a></li>
38.             <li><a href="about.php">About</a></li>
39.             <li><a href="contact.php">Contact</a></li>
40.             </li>
41.       </ul>
42.
43.       <ul class="nav navbar-nav navbar-right">
44.       <li><a href="Register.php"><span class="glyphicon glyphicon-
    user"></span> Sign Up</a></li>
45.       <li><a href="Login.php"><span class="glyphicon glyphicon-log-
    in"></span> Login</a></li>
46.       </ul>
47.       </nav>
48.     </div>
49.   </div>
50. </nav>
51. <!-- Site banner -->
52. <div class="banner">
53.     <div class="container" style=" padding:0px;" >
54.       <h1>Register</h1>
55.       <p class="text-center">Συμπληρώστε τα πεδία</p>
56.     </div>
57. </div>
58.
59. <!-- Middle content section -->
60. <div class="middle">
61.     <div class="container">
62.
63.       <!-- Form itself -->
64.         <form action="Register.php" method="post">
65.             <table>
66.               <tr>
67.                 <td><label for="Username">Username:</label></td>
68.                 <td><input type="text" id="Username" name="Username" /></td>
69.               </tr>
70.                <tr>
71.                 <td><label for="Password">Password:</label></td>
```

```
                    <td><input type="password" id="Password" name="Password" /></td>
                </tr>
                <tr>
                    <td><label for="Firstname">Firstname:</label></td>
                    <td><input type="text" id="Firstname" name="Firstname" /></td>
                </tr>
                <tr>
                    <td><label for="Lastname">Lastname:</label></td>
                    <td><input type="text" id="Lastname" name="Lastname" /></td>
                </tr>
                <tr>
                    <td><label for="E-mail">E-mail:</label></td>
                    <td><input type="email" id="E-mail" name="E-mail" /></td>
                </tr>
                <tr>
                    <td colspan="2" align="center">
                    <input type="submit" name="submit" value='Submit' /></td>
                </tr>
            </table>
        </form>

    </div>
</div>
<center><b>Already Registered</b><br><a href='Login.php'>Login Here </a></center>

<!--footer-->
<footer class="footer">
    <div class="container">
                <p class="text-muted">Πειραιάς 2015</p>
            </div>
        </footer>

    </body>
    </html>

    <?php
      $user_db_con = mysqli_connect('localhost', 'root', '', 'cms');
     if (!$user_db_con) {
        die('Could not connect: ' . mysqli_connect_error());
     }
     if(isset($_POST['submit'])){
       $username = $_POST['Username'];
       $password = $_POST['Password'];
       $email = $_POST['E-mail'];
       $Firstname = $_POST['Firstname'];
       $Lastname = $_POST['Lastname'];
       $check_email = "SELECT * FROM users WHERE Email='$email'";
       $check_user = "SELECT * FROM users WHERE Username='$username'";
       $run = mysqli_query($user_db_con,$check_email);
       $run1 = mysqli_query($user_db_con,$check_user);
       $query = "INSERT INTO users (Username, Password, Firstname, Lastname, Email)
VALUES ('$username', '$password', '$Firstname', '$Lastname', '$email')";
       if($username==''){
        echo "<script>alert('Username field cannot be empty!')</script>";
              echo "<script>window.open('register.php','_self')</script>";
       }
       else if($password==''){
         echo "<script>alert('Password field cannot be empty!')</script>";
```

```
129.              echo "<script>window.open('register.php','_self')</script>";
130.            }
131.            else if(strlen($password)<=6){
132.              echo "<script>alert('Password must have more than 6 characters!')</script>";
133.              echo "<script>window.open('register.php','_self')</script>";
134.            }
135.            else if($email==''){
136.              echo "<script>alert('E-mail field cannot be empty!')</script>";
137.              echo "<script>window.open('register.php','_self')</script>";
138.            }
139.            else if($Firstname==''){
140.              echo "<script>alert('Firstname field cannot be empty!')</script>";
141.              echo "<script>window.open('register.php','_self')</script>";
142.            }
143.            else if($Lastname==''){
144.              echo "<script>alert('Lastname field cannot be empty!')</script>";
145.              echo "<script>window.open('register.php','_self')</script>";
146.            }
147.            else
148.            {
149.             if(!$run1 || mysqli_num_rows($run1)>0){
150.              echo "<script>alert('Username: $username already exists in database, give us an
    other username!')</script>";
151.              echo "<script>window.open('register.php','_self')</script>";
152.             }
153.             else if(!$run || mysqli_num_rows($run)>0){
154.              echo "<script>alert('E-mail: $email already exists in database, give us another e-
    mail!')</script>";
155.              echo "<script>window.open('register.php','_self')</script>";
156.             }
157.             else
158.             {
159.              if(mysqli_query($user_db_con,$query)){
160.               echo "<script>alert('Registration Successfully Completed!')</script>";
161.               echo "<script>window.open('Login.php','_self')</script>";
162.              }
163.
164.             }
165.            }
166.
167.          }
168.         ?>
```

Show_test.php

```
1.  <!doctype html>
2.
3.  <html>
4.  <head>
5.  <meta charset="utf-8">
6.  <title>Register</title>
7.  <meta name="viewport" content="width=device-width, initial-scale=1">
8.  <link href="css/bootstrap.min.css" rel="stylesheet">
9.  <!--<link href="css/style.css" rel="stylesheet">-->
```

```
<link href="css/contact_form.css" rel="stylesheet">


<!-- Include jQuery and bootstrap JS plugins
<script type="text/javascript" src="js/bootstrap.min.js"></script>
<script type="text/javascript" src="js/jquery.form.js"></script>-->
<script type="text/javascript" src="js/jquery-2.1.0.min.js"></script>

<script type="text/javascript" src="js/jquery.validate.min.js"></script>
<script type="text/javascript" src="js/contact.js"></script>
</head>

<body>
<!--header-->
<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="IndexSite1.php">GreekTrend</a>
    </div>
    <div>
    <button class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
          <span class="glyphicon glyphicon-align-justify"></span>
       </button>
       <nav class="navbar-collapse collapse" role="navigation">
      <ul class="nav navbar-nav">
       <li class="active">
       <li><a href="IndexSite1.php">Home</a></li>
            <li><a href="about.php">About</a></li>
            <li><a href="contact.php">Contact</a></li>
            </li>
      </ul>

      <ul class="nav navbar-nav navbar-right">
       <li><a href="Register.php"><span class="glyphicon glyphicon-
user"></span> Sign Up</a></li>
       <li><a href="Login.php"><span class="glyphicon glyphicon-log-
in"></span> Login</a></li>
      </ul>
      </nav>
    </div>
  </div>
</nav>
<!-- Site banner -->
<div class="banner">
    <div class="container" style=" padding:0px;" >
       <h1>Register</h1>
       <p class="text-center">Συμπληρώστε τα πεδία</p>
    </div>
</div>

<!-- Middle content section -->
<div class="middle">
    <div class="container">

       <!-- Form itself -->
         <form action="Register.php" method="post">
              <table>
```

```
66.                 <tr>
67.                   <td><label for="Username">Username:</label></td>
68.                   <td><input type="text" id="Username" name="Username" /></td>
69.                 </tr>
70.                  <tr>
71.                   <td><label for="Password">Password:</label></td>
72.                   <td><input type="password" id="Password" name="Password" /></td>
73.                 </tr>
74.                 <tr>
75.                   <td><label for="Firstname">Firstname:</label></td>
76.                   <td><input type="text" id="Firstname" name="Firstname" /></td>
77.                 </tr>
78.                 <tr>
79.                   <td><label for="Lastname">Lastname:</label></td>
80.                   <td><input type="text" id="Lastname" name="Lastname" /></td>
81.                 </tr>
82.                 <tr>
83.                   <td><label for="E-mail">E-mail:</label></td>
84.                   <td><input type="email" id="E-mail" name="E-mail" /></td>
85.                 </tr>
86.                 <tr>
87.                   <td colspan="2" align="center">
88.                   <input type="submit" name="submit" value='Submit' /></td>
89.                 </tr>
90.               </table>
91.             </form>
92.
93.     </div>
94. </div>
95. <center><b>Already Registered</b><br><a href='Login.php'>Login Here </a></center>
96.
97. <!--footer-->
98. <footer class="footer">
99.       <div class="container">
100.                 <p class="text-muted">Πειραιάς 2015</p>
101.               </div>
102.             </footer>
103.
104.           </body>
105.           </html>
106.
107.           <?php
108.             $user_db_con = mysqli_connect('localhost', 'root', '', 'cms');
109.           if (!$user_db_con) {
110.               die('Could not connect: ' . mysqli_connect_error());
111.           }
112.           if(isset($_POST['submit'])){
113.             $username = $_POST['Username'];
114.             $password = $_POST['Password'];
115.             $email = $_POST['E-mail'];
116.             $Firstname = $_POST['Firstname'];
117.             $Lastname = $_POST['Lastname'];
118.             $check_email = "SELECT * FROM users WHERE Email='$email'";
119.             $check_user = "SELECT * FROM users WHERE Username='$username'";
120.             $run = mysqli_query($user_db_con,$check_email);
121.             $run1 = mysqli_query($user_db_con,$check_user);
122.             $query = "INSERT INTO users (Username, Password, Firstname, Lastname, Email)
      VALUES ('$username', '$password', '$Firstname', '$Lastname', '$email')";
```

```
123.            if($username==''){
124.             echo "<script>alert('Username field cannot be empty!')</script>";
125.                    echo "<script>window.open('register.php','_self')</script>";
126.            }
127.            else if($password==''){
128.              echo "<script>alert('Password field cannot be empty!')</script>";
129.              echo "<script>window.open('register.php','_self')</script>";
130.            }
131.            else if(strlen($password)<=6){
132.              echo "<script>alert('Password must have more than 6 characters!')</script>";
133.              echo "<script>window.open('register.php','_self')</script>";
134.            }
135.            else if($email==''){
136.              echo "<script>alert('E-mail field cannot be empty!')</script>";
137.              echo "<script>window.open('register.php','_self')</script>";
138.            }
139.            else if($Firstname==''){
140.              echo "<script>alert('Firstname field cannot be empty!')</script>";
141.              echo "<script>window.open('register.php','_self')</script>";
142.            }
143.            else if($Lastname==''){
144.              echo "<script>alert('Lastname field cannot be empty!')</script>";
145.              echo "<script>window.open('register.php','_self')</script>";
146.            }
147.            else
148.            {
149.             if(!$run1 || mysqli_num_rows($run1)>0){
150.              echo "<script>alert('Username: $username already exists in database, give us an
    other username!')</script>";
151.              echo "<script>window.open('register.php','_self')</script>";
152.             }
153.             else if(!$run || mysqli_num_rows($run)>0){
154.              echo "<script>alert('E-mail: $email already exists in database, give us another e-
    mail!')</script>";
155.              echo "<script>window.open('register.php','_self')</script>";
156.             }
157.             else
158.             {
159.              if(mysqli_query($user_db_con,$query)){
160.               echo "<script>alert('Registration Successfully Completed!')</script>";
161.               echo "<script>window.open('Login.php','_self')</script>";
162.              }
163.
164.             }
165.            }
166.
167.          }
168.         ?>
```

Trends.php

```
<!doctype html>
<?php


include "twitteroauth/twitteroauth.php";

$consumer_key = "K4Txq65bYg59Z1eqCDvFtYf8h";
$consumer_secret = "g21jdS18HmRyfu0jU7x47mW6xOMnNUBVRrCuDqw2jjfxnv65cp";
$access_token = "4922855261-8xsru98lephQjpXIt4LIoH38QWthkXRlxw9uhwx";
$access_token_secret = "otYqYz8hrd67BiK5z71bCJ0RHf8vYTFqFfuDvXHlTVFmq";
$twitter = new TwitterOAuth($consumer_key,$consumer_secret,$access_token,$access_token_secret);
//$tweets = $twitter->get('https://api.twitter.com/1.1/trends/place.json?id=2442047');
$tweets = $twitter->get("trends/place", [ "id" => 963291]);

?>

<html>
<head>
<meta charset="utf-8">
<title>trends on map</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="css/bootstrap.min.css" rel="stylesheet">
<!--<link href="css/style.css" rel="stylesheet">-->
<link href="css/contact_form.css" rel="stylesheet">

<!-- Include jQuery and bootstrap JS plugins
<script type="text/javascript" src="js/bootstrap.min.js"></script>
<script type="text/javascript" src="js/jquery.form.js"></script>-->
<script type="text/javascript" src="js/jquery-2.1.0.min.js"></script>
<script type="text/javascript" src="js/jquery.validate.min.js"></script>
<script type="text/javascript" src="js/contact.js"></script>
</head>

<body>
<!--header-->
<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header"> <a class="navbar-brand" href="IndexSite1.php">GreekTrend</a> </div>
    <div>
      <button class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse"> <span class="glyphicon glyphicon-align-justify"></span> </button>
      <nav class="navbar-collapse collapse" role="navigation">
        <ul class="nav navbar-nav">
          <li class="active">
          <li><a href="../IndexSite1.php">Home</a></li>
          <li><a href="../about.php">About</a></li>
          <li><a href="../contact.php">Contact</a></li>
      <li><a href="../trends.php">Trends</a></li>
          </li>
        </ul>
        <ul class="nav navbar-nav navbar-right">
          <li><a href="../Register.php"><span class="glyphicon glyphicon-user"></span> Sign Up</a></li>
```

```
52.           <li><a href="../Login.php"><span class="glyphicon glyphicon-log-
    in"></span> Login</a></li>
53.         </ul>
54.       </nav>
55.     </div>
56.   </div>
57. </nav>
58.
59.
60. <script>
61.
62.
63.
64.     var thessaloniki = <?php echo json_encode($tweets);?>;
65. //    <?php
66. //    $tweets = $twitter->get("trends/place", [ "id" => 24543541]); ?>
67.
68. //    var athens = <?php echo json_encode($tweets);?>;
69.
70.
71. //console.log(athens)
72. //console.log(thessaloniki)
73.
74. if(!String.linkify) {
75.     String.prototype.linkify = function() {
76.
77.         // http://, https://, ftp://
78.         var urlPattern = /\b(?:https?|ftp):\/\/[a-z0-9-+&@#\/%?=~_|!:,.;]*[a-z0-9-
    +&@#\/%=~_|]/gim;
79.
80.         // www. sans http:// or https://
81.         var pseudoUrlPattern = /(^|[^\/])(www\.[\S]+(\b|$))/gim;
82.
83.         // Email addresses
84.         var emailAddressPattern = /[\w.]+@[a-zA-Z_-]+?(?:\.[a-zA-Z]{2,6})+/gim;
85.
86.         return this
87.             .replace(urlPattern, '<a href="$&">$&</a>')
88.             .replace(pseudoUrlPattern, '$1<a href="http://$2">$2</a>')
89.             .replace(emailAddressPattern, '<a href="mailto:$&">$&</a>');
90.     };
91. }
92.     var loc = thessaloniki// athens//.concat(thessaloniki);
93.
94.
95.     var geocoder;
96.     var map;
97.     var bounds;
98.     var infoWindows = [];
99.
100.                 function initMap() {
101.
102.                   var markers = [];
103.                   var xartis = {lat: 38.091506, lng: 23.9061081};
104.
105.                   map = new google.maps.Map(document.getElementById('map'), {
106.                     zoom: 5,
107.                     center: xartis
```

```
108.                });
109.                bounds = new google.maps.LatLngBounds();
110.                geocoder = new google.maps.Geocoder();
111.                for(var i = 0; i < loc.length; i++){
112.                    for(var j in loc[i].trends){
113.                        console.log(loc[i]);
114.                         geocodeAddress(loc[i],j, loc[i].locations[i].name);
115.
116.                    }
117.                }
118.
119.            }
120.
121.
122.        function getRandomArbitrary(min, max) {
123.            return Math.random() * (max - min) + min;
124.        }
125.
126.        function coordinakias(pos){
127.
128.          var con_lng = pos.lng() - getRandomArbitrary(-0.0001,0.01);
129.          var con_lat= pos.lat() - getRandomArbitrary(-0.0001,0.01);
130.
131.
132.          return {lat: con_lat, lng: con_lng};
133.
134.
135.        }
136.
137.        function geocodeAddress(locations, i, addr) {
138.          var title = locations.trends[i].name;
139.          var addraddress = addr;
140.          geocoder.geocode({
141.              'address': addr
142.            },
143.
144.            function(results, status) {
145.              if (status == google.maps.GeocoderStatus.OK) {
146.                var marker = new google.maps.Marker({
147.                  map: map,
148.                  position: coordinakias(results[0].geometry.location),
149.                  title: title,
150.                  animation: google.maps.Animation.DROP,
151.                  address: address,
152.                  icon: "tw.png"
153.                })
154.                infoWindow(marker, map, title, address);
155.                bounds.extend(marker.getPosition());
156.                map.fitBounds(bounds);
157.              } else {
158.                //alert("geocode of " + address + " failed:" + status);
159.              }
160.            });
161.        }
162.
163.        function infoWindow(marker, map, title, address) {
164.          google.maps.event.addListener(marker, 'click', function() {
```

```
165.              var html = "<div><b>" + title.linkify() + "</b></br></br><p>" + address + "<br></div
    </p></div>";
166.
167.              iw = new google.maps.InfoWindow({
168.                content: html,
169.                maxWidth: 350
170.              });
171.              closeAllInfoWindows();
172.              infoWindows.push(iw);
173.
174.              iw.open(map, marker);
175.            });
176.          }
177.
178.          function createMarker(results) {
179.            var marker = new google.maps.Marker({
180.              map: map,
181.              position: results[0].geometry.location,
182.              title: title,
183.              animation: google.maps.Animation.DROP,
184.              address: address
185.            })
186.            bounds.extend(marker.getPosition());
187.            map.fitBounds(bounds);
188.            infoWindow(marker, map, title, address);
189.            return marker;
190.          }
191.
192.          function closeAllInfoWindows() {
193.            for (var i=0;i<infoWindows.length;i++) {
194.              infoWindows[i].close();
195.            }
196.          }
197.
198.          </script>
199.
200.          <!DOCTYPE html>
201.          <html lang="en">
202.          <head>
203.          <meta charset="UTF-8">
204.          <title>GreekTrend API SEARCH</title>
205.          </head>
206.          <body>
207.          <form action="show_test.php" method="post">
208.            <div align="center">
209.              <label> Αναζήτηση: </label>
210.              <input class= "searchBar" type="text" name="keyword"/>
211.            </div>
212.          </form>
213.          <div id="map" style="height:500px!important"></div>
214.          <?php print_r( $tweets) ?>
215.          <?php
216.          $reg_exUrl = "/(http|https|ftp|ftps)\:\/\/[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(\/\S*)?/";
217.          foreach ($tweets->statuses as $key => $tweet) { ?>
218.          Tweet : <img src="<?=$tweet->user->profile_image_url;?>" />
219.          <?php    $text = preg_replace('@(https?://([-\w\.]+)+(:\d+)?(/([-
    \w/_\.]*(\?\S+)?)?)?)@', '<a href="$1" target="_blank">$1</a>', $tweet->text);?>
220.
```

```
221.        <?=$text?>
222.
223.        <br>
224.        <?php } ?>
225.        <script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCz9K74m2hztviLm0QUI1hLeEo1F1rM8_M&callback=initMap"
            async defer></script>
226.
227.        </body>
228.        </html>
```