
**ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ**

**ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ**
**ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ**

---

# ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**ΘΕΜΑ: «Ανάπτυξη εργαλείου αυτόματης ενημέρωσης λογαριασμού στο Twitter»**

Ιωαννίδης Σωκράτης (08/3397)
sioannid@it.teithe.gr
Κιπριτσής Αλέξανδρος (08/3378)
akiprits@it.teithe.gr

Επιβλέπων καθηγητής: Δρ. Ηλιούδης Χρήστος
iliou@it.teithe.gr

**Θεσσαλονίκη 2014**

# ΠΡΟΛΟΓΟΣ

Η μεταφορά πληροφορίας μεταξύ μη ομοειδών συστημάτων με χρήση διεπαφών προγραμματισμού εφαρμογών, αποτελεί το ερευνητικό τμήμα της παρούσας πτυχιακής εργασίας. Ο τρόπος λειτουργίας ενός διαδικτυακού συστήματος αυτόματων ενημερώσεων, που μεταφέρει πληροφορία έμμεσα σε στοχευμένο κοινό, αποτελεί μια ενδιαφέρουσα θεματική με πολλές εφαρμογές στη διαφήμιση προϊόντων, την ειδησεογραφία, την οικονομία και άλλους τομείς.

Πιο συγκεκριμένα, η δυνατότητα εξωτερικής διαχείρισης περιεχομένου που αποκτά κανείς χρησιμοποιώντας μια διεπαφή κοινωνικού δικτύου, επιτρέπει την ύπαρξη και λειτουργία εφαρμογών, οι οποίες δημιουργούν αυτόματα αναρτήσεις, δημοσιοποιώντας με αυτόν τον τρόπο περιεχόμενο, που αντλείται από κάποια τρίτη πηγή. Ωστόσο, μια τέτοια διεπαφή μπορεί να χρησιμοποιηθεί και για άλλους σκοπούς, όπως για παράδειγμα η απόκτηση περιεχομένου, η συλλογή στατιστικών στοιχείων και η αλλαγή ρυθμίσεων κάποιου λογαριασμού. Μια τέτοια εφαρμογή, η οποία αναρτά αυτόματα ενημερώσεις σε ένα λογαριασμό στο Twitter, αποπειραθήκαμε να δημιουργήσουμε και εμείς.

# ΠΕΡΙΛΗΨΗ

Η δημιουργία μιας εφαρμογής, που επιτρέπει την ανταλλαγή δεδομένων μεταξύ δύο συστημάτων στο διαδίκτυο, αποτελεί δύσκολο έργο, ειδικά όταν τα συστήματα αυτά δε μοιάζουν μεταξύ τους. Ωστόσο ο βαθμός δυσκολίας μειώνεται δραστικά, όταν κάποιο από τα συστήματα αυτά παρέχει διεπαφή προγραμματισμού εφαρμογών (Application Programming Interface ή API) και εκμηδενίζεται σχεδόν, αν τέτοιες διεπαφές παρέχονται αμφότερα. Η δυσκολία πηγάζει κυρίως από την ανάγκη μετατροπής της μορφής της πληροφορίας, που αντλείται ή αποστέλλεται από την πηγή των δεδομένων, έτσι ώστε να γίνεται κατανοητή από το δέκτη.

Οι προγραμματιστές διεπαφών προγραμματισμού εφαρμογών φροντίζουν για τη διευκόλυνση αυτής της επικοινωνίας, δημιουργώντας ένα απλό προκαθορισμένο λεξιλόγιο και συντακτικό, τα οποία μπορεί κανείς να μελετήσει και να χρησιμοποιεί. Τέτοια λεξιλόγια και συντακτικά παρέχει και το Twitter, που είναι ένα από τα πιο διαδεδομένα κοινωνικά δίκτυα. Η παρούσα πτυχιακή εργασία, στόχο έχει τη μελέτη αυτών των διεπαφών και των τεχνολογιών που χρησιμοποιούνται, με απώτερο σκοπό τη δημιουργία μιας εφαρμογής, η οποία θα επικοινωνεί με το συγκεκριμένο κοινωνικό δίκτυο και θα δημιουργεί αναρτήσεις σε κάποιον προκαθορισμένο λογαριασμό, αντλώντας δεδομένα από κάποια ιστοσελίδα. Ως προς το περιεχόμενο της, κάθε ανάρτηση πρέπει να παρέχει ενημέρωση σε σχέση με δημοσιεύσεις, που έχουν γίνει στην πηγή της πληροφορίας. Σημαντικό τμήμα της έρευνας αποτέλεσε η διαδικασία επιλογής των δύο πιθανών μοντέλων απόκτησης πληροφορίας pull και push, αλλά και συνδυασμών τους.

Η εφαρμογή που δημιουργήσαμε τελικά, αντλεί πληροφορία από την ιστοσελίδα δημόσιων ανακοινώσεων του Τμήματος Μηχανικών Πληροφορικής και, αφού την επεξεργαστεί κατάλληλα, δημιουργεί περιεκτικές αναρτήσεις σε ένα λογαριασμό Twitter, παρέχοντας και έναν υπερσύνδεσμο ως δείκτη στην πηγή. Ωστόσο με κατάλληλες επεκτάσεις του κώδικα, είναι δυνατό να γίνονται και άλλες εργασίες, όπως ανάρτηση αυτόματων ενημερώσεων σε άλλα κοινωνικά δίκτυα ή η αντιστροφή της πορείας των δεδομένων, δηλαδή από το Twitter προς κάποιο άλλο σύστημα.

Οι τεχνολογίες, που μελετήθηκαν και χρησιμοποιήθηκαν για τη δημιουργία της εφαρμογής, είναι το πρωτόκολο HTTP και η γλώσσα CURL για την απόκτηση και την μετβίβαση των δεδομένων, οι γλώσσες σήμανσης HTML και XML και η γλώσσα επερωτήσεων XPATH για την ανάγνωση και το διαχωρισμό της χρήσιμης πληροφορίας και η ανοικτή τυποποιημένη μορφοποίηση JSON ως μοντέλο ανταλλαγής μηνυμάτων με το Twitter. Ως σημείο αναφοράς χρησιμοποιήθηκε η RESTful διεπαφή προγραμματισμού εφαρμογών του Twitter, ενώ για την ταυτοποίηση και αυθεντικοποίηση της εφαρμογής χρησιμοποιήθηκε το ανοικτό πρότυπο OAuth, που είναι προαπαιτούμενο για οποιαδήποτε μετέπειτα επικοινωνία με το κοινωνικό δίκτυο. Ο πηγαίος κώδικας γράφτηκε σε PHP και όλα αυτά γίναν δυνατά με τη χρήση ενός περιβάλλοντος LAMP.

# ABSTRACT

The development of an application, which enables multiple Information Systems to communicate with each other, is a challenging task. However, when one or more of those Systems provide Application Programming Interfaces, the process becomes simpler. The difficulty usually arises from the need to transform the data which is being traded, in a way that the target system can understand messages containing information sent or acquired from the source.

Application Programming Interface programmers try to confront this issue, by providing a simple syntax and vocabulary, when creating these interfaces. The social network called Twitter also provides its own syntaxes and vocabularies, which can be used to trade information with third party applications. One of the goals of this thesis is the development of such an application, which will be able to aquire information from a website, filter it and tweet it. A great part of this research included the selection of a model to aquire data from the source, in particular the pull and push models and their combinations.

The application is pulling data from the source, which in our case is the public news page of the Department of Informatics, and creates new tweets, providing a title and a link, which points to the corresponding announcement at the source. However, it is possible to extend or configure this application in order to create posts on other social networks for example, or even turn the flow of the data, while using Twitter as the source instead. Of course in this case, the target has to provide some means of changing its content.

The technologies studied and used for the development of this application include the HTTP protocol and CURL language for acquiring and transmitting the data, the HTML and XML markup languages, as well as the query language XPATH for the parsing and filtering of the information. The open standard format JSON was used to trade data with Twitter, while the RESTful API it provides was used as an end point in order to transmit this data. OAuth is the only means of authentication supported by Twitter. The source code is written in the programming language PHP and the host is an Apache server on Linux.

# ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία είναι αφιερωμένη στους γονείς μας, που είναι στο πλευρό μας μονίμως και μας στέκονται σε ό,τι εμπόδιο ή πρόκληση συναντούμε στη ζωή μας.

Θέλουμε επίσης να ευχαριστήσουμε τον κ. Ηλιούδη, για την καθοδήγηση που μας παρείχε καθόλη τη διάρκεια των σπουδών μας.

Και, τέλος, θέλουμε να ευχαριστήσουμε τους φίλους μας, οι οποίοι μας στήριξαν και μας δίνανε κουράγιο, για να ολοκληρώσουμε αυτήν την εργασία.

# ΠΕΡΙΕΧΟΜΕΝΑ

# Ευρετήριο εικόνων

# 1 Εισαγωγή

## 1.1 Στόχος της εργασίας

Στην παρούσα πτυχιακή εργασία, θα μελετήσουμε τον τρόπο ανάπτυξης ενός εργαλείου αυτόματης ενημέρωσης του Twitter. Με τη χρήση αυτού του εργαλείου θα αυτοματοποιείται η διαδικασία της ενημέρωσης του προσωπικού, των φοιτητών, αλλά και τρίτων, που μπορεί να ενδιαφέρονται (εφόσον όλοι χρησιμοποιούν το Twitter).

Πιο ειδικά, στόχος είναι η δημιουργία μιας εφαρμογής, η οποία θα ελέγχει για νέες δημόσιες ανακοινώσεις του Τμήματος Μηχανικών Πληροφορικής και, εφόσον υπάρχουν, θα δημιουργεί tweets για καθεμία από αυτές σε κάποιον προκαθορισμένο λογαριασμό στο Twitter.

Η επιλογή του ιστότοπου της ΄Υδρας, βέβαια, είναι αυθαίρετη. Το εργαλείο αυτό μπορεί με μερικές τροποποιήσεις να αντλεί δεδομένα και από άλλες πηγές. Επίσης, με κάποιες προσθήκες είναι δυνατό να επικοινωνεί με άλλες διεπαφές κοινωνικών δικτύων.

## 1.2 Υπηρεσίες αυτόματης ενημέρωσης και κοινωνικά δίκτυα

Οι εφαρμογές αυτόματης ενημέρωσης άρχισαν να χρησιμοποιούνται κατά κόρον μετά την εξάπλωση των πρώτων κοινωνικών δικτύων, κυρίως λόγω της μεγάλης απήχησης που απέκτησαν αυτά. Όσο πιο πολύ χρησιμοποιείται ένα κοινωνικό δίκτυο, τόσο πιο εύκολη γίνεται η μαζική μετάδοση πληροφορίας, είτε αυτή η πληροφορία είναι κάποια είδηση, είτε διαφήμιση ενός προϊόντος, κτλ.

Τα κοινωνικά δίκτυα είναι πλέον μέρος της καθημερινότητας μιας μεγάλης μερίδας ανθρώπων, η οποία ολοένα και αυξάνεται. Η διαφήμιση (με τη γενική έννοια του όρου) σε ένα τέτοιο χώρο αποτελεί πλέον όχι μόνο κάτι το ελκυστικό, αλλά είναι και αναγκαία. Δεν νοείται ειδησεογραφικός οργανισμός ή οποιαδήποτε άλλη επιχείρηση με κύρος, να μην χρησιμοποιεί τα γνωστά κοινωνικά δίκτυα για προβολή των προϊόντων της. Κάθε τέτοια επιχείρηση, χρησιμοποιεί (ή πρέπει να χρησιμοποιεί) κάποιο εργαλείο αυτόματων ενημερώσεων, για την σίγουρη και άμεση ανάρτηση πληροφορίας σε κοινωνικά δίκτυα. Ωστόσο, η χρήση μιας τέτοιας εφαρμογής δεν περιορίζεται μόνο σε επιχειρήσεις, αλλά μπορεί να γίνεται και από ιδιώτες ή για παράδειγμα ένα Τμήμα Πληροφορικής ενός δημόσιου τριτοβάθμιου εκπαιδευτικού ιδρύματος. Για τον λόγο αυτό, ενώ υπάρχουν αρκετά επαγγελματικά πακέτα που παρέχουν τέτοιες και συναφείς υπηρεσίες (όπως στατιστικές μετρήσεις για παράδειγμα), υπάρχουν και πολλές εφαρμογές που μπορεί να χρησιμοποιηθούν εντελώς δωρεάν.

## 1.3 Σπουδαιότητα και προστιθέμενη αξία της εργασίας

Με τη χρήση αυτής της εφαρμογής, δε θα είναι απαραίτητη πλέον η σύνδεση στον ιστότοπο Ύδρα (hydra.it.teithe.gr) για να ελέγξει κανείς αν υπάρχουν νέες δημόσιες ανακοινώσεις. Έτσι, η διαδικασία αυτή γίνεται πιο εύκολη. Θα μπορεί κάποιος να χρησιμοποιήσει ένα Smartphone, στο οποίο θα εκτελείται η εφαρμογή Twitter, για να δέχεται αυτόματα ενημερώσεις, χωρίς να είναι απαραίτητο να ελέγξει ένα λογαριασμό e-mail ή να διαβάσει την αντίστοιχη σελίδα της Ύδρας. Η εργασία με αυτόν τον τρόπο προσθέτει αξία στον ιστότοπο της Ύδρας, διευκολύνοντας την ενημέρωση των χρηστών της.

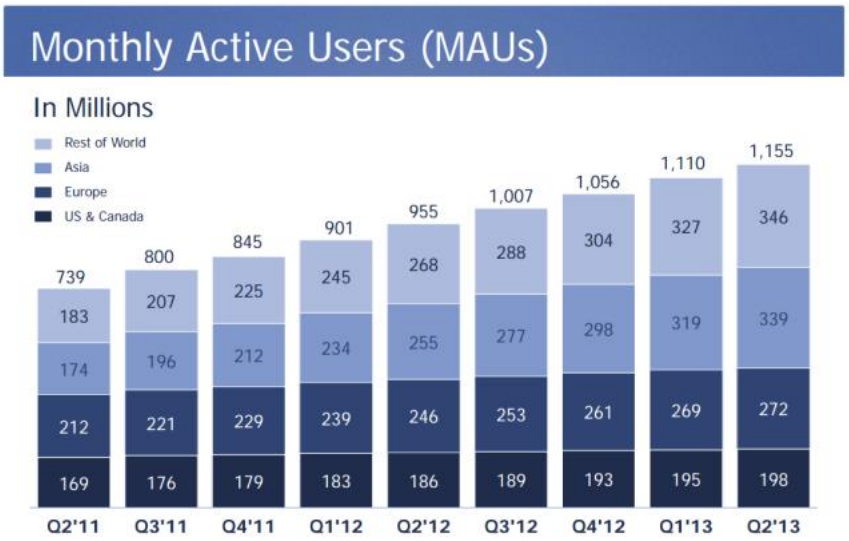
Η εφαρμογή που τελικά δημιουργήσαμε αποτελεί λογισμικό ανοιχτού κώδικα και ως εκ τούτου μπορεί κανείς να επέμβει στον κώδικα για να κάνει ό,τι αλλαγή ή διόρθωση επιθυμεί. Φυσικά μπορεί να επεκταθεί επίσης, ώστε να δημιουργεί αναρτήσεις και σε άλλα κοινωνικά δίκτυα για παράδειγμα ή να δέχεται είσοδο από περισσότερες πηγές.

Κάτι καινούριο που προσφέρει, το οποίο δε μπορέσαμε να βρούμε ως υφιστάμενη λύση, είναι η δυνατότητα εξαγωγής πληροφορίας από τον πηγαίο κώδικα μιας ιστοσελίδας, τον οποίο διατρέχει, επεξεργάζεται κατάλληλα και δημιουργεί ένα περιεκτικό μήνυμα προς ανάρτηση.

Μια τέτοια έρευνα από την άλλη μεριά, δίνει τη δυνατότητα εξοικείωσης με το API του Twitter αλλά και τις διεπαφές προγραμματισμού εφαρμογών γενικότερα, όπως και με τον τρόπο επικοινωνίας μεταξύ μη ομοειδών συστημάτων.

## 1.4 Twitter

Συγκεκριμένα το Twitter[1] επελέγη, διότι είναι η πιο διαδεδομένη πλατφόρμα κοινωνικής δικτύωσης στην οποία μεταφέρονται σημαντικές πληροφορίες και ειδήσεις, όπως για παράδειγμα σε περιπτώσεις κρίσεων, πολέμων και άλλα σημαντικά γεγονότα ανά την υφήλιο. Είναι ίσως το πιο σοβαρό κοινωνικό δίκτυο με τόσο ευρεία απήχηση. Τον τελευταίο καιρό μάλιστα υπάρχει άνοδος της χρήσης του, πράγμα το οποίο ίσως οφείλεται εν μέρει στο γεγονός ότι τα τελευταία χρόνια οι κρίσεις παγκόσμια αυξάνονται και κάθε χρόνο αυξάνεται και ο αριθμός των ατόμων με πρόσβαση στο διαδίκτυο. Η παρακάτω εικόνα είναι ενδεικτική της ραγδαίας αύξησης χρήσης, ειδικά εκτός Αμερικής, Καναδά και Ευρώπης (όπου οι περισσότεροι ήδη έχουν πρόσβαση στο διαδίκτυο).




**Εικόνα 1.1 Διάγραμμα αριθμού ενεργών χρηστών Twitter**

Το Twitter είναι μια υπηρεσία online social networking και microblogging, που επιτρέπει στους χρήστες της να στέλνουν μηνύματα κειμένου ως 140 χαρακτήρες, τα ονομαζόμενα tweets.

Η υπηρεσία από τη στιγμή που δημιουργήθηκε απέλαβε παγκόσμια δημοσιότητα πολύ γρήγορα, έχοντας πάνω από μισό εκατομμύριο χρήστες ήδη το 2012, οι οποίοι παρήγαγαν πάνω από 340 εκατομμύρια νέα tweets καθημερινά. Επίσης, το Twitter εκείνον τον καιρό διαχειριζόταν περισσότερα από 1.6 δισεκατομμύρια queries αναζητήσεων την ημέρα.

Το κοινωνικό αυτό δίκτυο είναι μία από τις δέκα σελίδες με τις περισσότερες επισκέψεις τη μέρα και έχει περιγραφεί ως “το SMS του Internet”. Οι χρήστες χωρίς λογαριασμό μπορούν να διαβάσουν τα δημόσια tweets ενεργών χρηστών, ενώ οι ενεργοί χρήστες μπορούν να δημιουργήσουν νέα tweets μέσω του web interface, μέσω SMS και μέσω πληθώρας εφαρμογών κινητών συσκευών.

### *1.4.1 Τεχνολογίες*

Από τις 31 Αυγούστου 2010, εφαρμογές Twitter τρίτων υποχρεούνται να χρησιμοποιούν την OAuth, μια μέθοδο αυθεντικοποίησης που δεν απαιτεί από τους χρήστες να εισάγουν τον κωδικό τους στην εφαρμογή που κάνει την αυθεντικοποίηση. Έτσι, σύμφωνα με την ίδια την Twitter, οδηγούμαστε σε βελτιωμένη εμπειρία χρήστη και μεγαλύτερη ασφάλεια. Το Application Programming Interface (API) της υπηρεσίας επιτρέπει σε άλλες υπηρεσίες του διαδικτύου να επικοινωνούν με αυτήν, ενώ η λειτουργία της ίδιας βασίζεται κατά κύριο λόγο σε λογισμικό ανοιχτού κώδικα.

Το Twitter[23] χρησιμοποιεί μια σειρά από τεχνολογίες, από τις οποίες ένα μεγάλο ποσοστό είναι ελεύθερο λογισμικό ανοιχτού κώδικα. Η διεπαφή ιστού χρησιμοποιεί το γνωστό και πολυχρησιμοποιημένο Ruby on Rails framework (πάνω σε μια ειδικά διαμορφωμένη πλατφόρμα για όσο το δυνατόν καλύτερη απόδοση επιδόσεων).

Στην αρχή χρησιμοποιούνταν βάσεις δεδομένων MySQL και sharding για την αποθήκευση των tweets, ωστόσο δημιουργούνταν προβλήματα με την εγγραφή και την ανάγνωση στο Twitter, οπότε και η εταιρεία αποφάσισε να επανασχεδιάσει την εφαρμογή.

Ταυτόχρονα και σε άλλα επίπεδα εκτελούνταν επανασχεδιασμοί, από το Ruby on Rails search stack σε έναν διακομιστή Java με την ονομασία “Blender” και από έναν Ruby persistent διακομιστή ουράς (ονόματι Starling) σε λογισμικό γραμμένο σε Scala[1].

Όλες οι παραπάνω αλλαγές οδήγησαν σταδιακά στη δυνατότητα του κάθε εξυπηρέτη να επεξεργάζεται δέκα με είκοσι χιλιάδες αιτήματα το δευτερόλεπτο, από τριακόσια που μπορούσε ως τότε!

Ταυτόχρονα οι μηχανικοί του ανέπτυξαν διεπαφές για την επικοινωνία με τρίτες εφαρμογές και υπηρεσίες ιστού (APIs)[7].

Κάθε tweet που αναρτάται, εισέρχεται σε ένα μητρώο αποκτώντας ένα μοναδικό αναγνωριστικό με χρήση του λογισμικού Snowflake και προστίθεται πληροφορία geolocation σε αυτό, με χρήση του “Rockdove”. Έπειτα η υπηρεσία σμίκρυνσης t.co (του Twitter) ελέγχει για συνδέσμους spam και μικραίνει το αντίστοιχο URL. Τα tweets αποθηκεύονται σε μια βάση MySQL (μέσω ενός λογισμικού ονόματι Gizzard) και αποστέλλεται ανάδραση στους χρήστες για την επιτυχία της ανάρτησης. Τέλος, αποστέλλονται σε μηχανές αναζήτησης μέσω του Firehose API.

Ολόκληρη η διαδικασία διαρκεί 350ms και την διαχειρίζεται το λογισμικό FlockDB[24].

### *1.4.2 Η ορολογία του Twitter*

Λόγω της ραγδαίας εξάπλωσης, το Twitter έχει δημιουργήσει το δικό του μικρό λεξικό. Οι πιο σημαντικοί όροι επεξηγούνται παρακάτω.

Tweet: Το κοινό μήνυμα του Twitter το οποίο αποτελείται από έναν έως 140 χαρακτήρες.

Retweet: Ένα tweet το οποίο έχει αναμεταδοθεί από έναν χρήστη προς αυτούς που τον παρακολουθούν.

Hashtag (#): Το σύμβολο της δίεσης χρησιμοποιείται μέσα στα tweets για λόγους αναγνωρισιμότητας ενός θέματος συνήθως, και γρήγορης αναζήτησης. Βάση της δίεσης, οι χρήστες μπορούν εύκολα να βρούνε τα θέματα που τους ενδιαφέρουν και να τα παρακολουθούν.

Mention: Η αναφορά σε έναν χρήστη. Πραγματοποιείται χρησιμοποιώντας το “@” ακολουθούμενο από το ψευδώνυμο του χρήστη.

Feed: Η αρχική σελίδα κάθε χρήστη. Εδώ εμφανίζονται τα tweets των χρηστών που παρακολουθεί κάποιος.

Lists: Το Twitter παρέχει των μηχανισμό των λιστών προκειμένου να μπορεί κάθε χρήστης να οργανώνει καλύτερα τις αναρτήσεις χρηστών τους οποίους παρακολουθεί.

Direct Message: Ο μηχανισμός που χρησιμοποιείται όταν θέλουμε να στείλουμε ένα μήνυμα σε κάποιον άλλο χρήστη χωρίς να είναι ορατό από τρίτους.

## 1.5 Σύντομη περιγραφή κεφαλαίων

Στο επόμενο κεφάλαιο περιγράφονται οι υπηρεσίες αυτόματης ενημέρωσης και τα μοντέλα τους. Γίνεται σύγκριση των μοντέλων pull και push και αναφορά σε κάποιους δυνατούς συνδυασμούς τους. Στη συνέχεια, γίνεται αξιολόγηση των δύο μοντέλων και επιλογή του πιο βολικού για τις ανάγκες υλοποίησης της εφαρμογής.

Στο τρίτο κεφάλαιο γίνεται μια σύντομη παρουσίαση κάποιων υφιστάμενων εφαρμογών, οι οποίες έχουν παρόμοιο στόχο, δηλαδή την αυτόματη ανάρτηση νέων tweets στο Twitter, τα οποία η κάθε μια μπορεί να λαμβάνει από διαφορετικού είδους πηγή ή πηγές. Κάνουμε σύγκριση των υπηρεσιών αυτών, μελετώντας τον τρόπο λειτουργίας τους και τις δυνατότητές τους.

Στο τέταρτο κεφάλαιο αναλύονται οι απαιτήσεις της υλοποίησης μιας τέτοιας εφαρμογής και στη συνέχεια περιγράφεται ο τελικός σχεδιασμός της υπηρεσίας. Γίνεται σύντομη αναφορά στα εργαλεία που χρησιμοποιήθηκαν για την υλοποίησή της και περιγράφονται κάποια βασικά συστατικά της. Επίσης, παρουσιάζονται τα API που παρέχει το Twitter και γίνεται η επιλογή αυτού που θα χρησιμοποιηθεί.

Στο κεφάλαιο που ακολουθεί γίνεται μια συνοπτική περιγραφή του τεχνολογικού περιβάλλοντος που χρησιμοποιήθηκε και ειδικότερα των σημείων που έχουν άμεση σχέση με την δημιουργία μιας τέτοιας εφαρμογής. Εξηγείται γιατί έχουν επιλεγεί οι συγκεκριμένες τεχνολογίες (όπου αυτό έχει νόημα) και τι δυνατότητες δίνουν. Εδώ γίνεται και μια εκτενής περιγραφή των API του Twitter.

Στο έκτο κεφάλαιο γίνεται αναλυτική περιγραφή της υπηρεσίας, σημαντικών τμημάτων του κώδικα της εφαρμογής και της μοντελοποίησης της πληροφορίας.

Στο κεφάλαιο επτά γίνεται λόγος για την καινοτομία της εφαρμογής και την προστιθέμενη αξία που προσφέρει.

Το όγδοο κεφάλαιο, τέλος, περιέχει πιθανές μελλοντικές επεκτάσεις της εφαρμογής, τρόπους βελτίωσής της ή πρόσθεσης αξίας.

# 2 Χαρακτηριστικά ανάλογων πλατφορμών και αναλυτική περιγραφή του μοντέλου που έχει υιοθετηθεί από την υπηρεσία

Στο διαδίκτυο υπάρχουν πολλές δυνατότητες για αναμετάδοση πληροφορίας και ενημέρωση χρηστών. Έχουν στο παρελθόν δημιουργηθεί τεχνολογίες αυτόματης αναμετάδοσης, όπως τα RSS feeds, οι mailing lists και τα subscriptions. Με την έξαρση των κοινωνικών δικτύων, όμως, άρχισε πολύς κόσμος να χρησιμοποιεί επιπλέον και διάφορες πλατφόρμες κοινωνικής δικτύωσης για να ενημερώνεται. Μπορεί στο Facebook να κάνει “Like”, στο Twitter “Follow” και στο Google+ “Add”, οπότε αυτόματα λαμβάνει ενημερώσεις από συγκεκριμένα πρόσωπα ή πάνω σε συγκεκριμένα θέματα.

Αυτό, βέβαια, δεν είναι απαραίτητα πάντα καλό, γιατί κάποιοι χρήστες χρησιμοποιούν μόνο τα συγκεκριμένα για να ενημερώνονται, και πολλές φορές, μάλιστα, δεν τα χρησιμοποιούν καν για ενημέρωση. Επίσης πολλές φορές σε τέτοια δίκτυα (και όχι μόνο) κυκλοφορούν «ειδήσεις», οι οποίες δεν είναι αληθείς. Μάλιστα έχουν υπάρξει περιπτώσεις κυκλοφορίας ειδήσεων για πρώτη φορά σε κοινωνικά δίκτυα, τις οποίες αναμετέδωσαν «έγκυρα» υποτίθεται μέσα ενημέρωσης (τηλεόραση, ραδιόφωνο, κτλ.). Σε κάποιες περιπτώσεις, δε, η εγκυρότητα μιας είδησης δεν είναι δυνατό να εξακριβωθεί.

Ωστόσο, η σημαντικότητα των κοινωνικών δικτύων έγκειται στο γεγονός ότι χρησιμοποιούνται μαζικά και οι ειδήσεις ρέουν με ταχύτατους ρυθμούς. Μπορεί κανείς γρήγορα και εύκολα να μάθει για πράγματα τα οποία συμβαίνουν σε άλλες ηπείρους ή στη γειτονιά του, χωρίς να πρέπει να κοιτάξει αλλού για κάθε τι. Το Twitter, συγκεκριμένα, έχει χρησιμοποιηθεί κατά κόρον σε περιπτώσεις κρίσεων (σεισμοί, πυρκαγιές, πλημμύρες κτλ.) και σε περιπτώσεις πολέμων, καταστολών και άλλες, με στόχο την ενημέρωση αλλά και για τον συντονισμό διάφορων δράσεων.

Κάθε εφαρμογή που ανακτά πληροφορία από οποιαδήποτε πηγή και δημιουργεί νέες κοινοποιήσεις στο Twitter, αποτελείται από τρία βασικά σκέλη. Την ανάκτηση της πληροφορίας, την επεξεργασία της και την ανάρτηση της κοινοποίησης. Η πρώτη φάση της ανάκτησης των δεδομένων είναι σημαντικό να μελετηθεί, διότι είναι αυτή στην οποία μπορούμε να έχουμε σημαντικές διαφοροποιήσεις. Η δεύτερη φάση έχει νόημα μόνον όταν είναι απαίτηση ή ανάγκη να γίνει μετατροπή της πληροφορίας και, στην περίπτωσή μας, ο περιορισμός είναι η δημιουργία του tweet, το οποίο μπορεί να έχει μέχρι 140 χαρακτήρες το πολύ. Η τρίτη φάση δεν προσφέρεται για έρευνα, εφόσον ο μόνος τρόπος επικοινωνίας με το Twitter είναι μέσω κάποιου API που το ίδιο παρέχει, οπότε αρκεί να μελετήσουμε αυτό.

## 2.1 Ανάκτηση πληροφορίας

Σαν μοντέλα επικοινωνίας και ροής δεδομένων σε κατανεμημένες αρχιτεκτονικές, τα μοντέλα push και pull είναι γνωστά και διαφοροποιούνται

στο αν η ροή των δεδομένων γίνεται με γνώμονα νέα δεδομένα ή γνώμονα κάποιο αίτημα. Και στα δύο αυτά μοντέλα υπάρχει μία δρώσα και παθητική οντότητα, οι οποίες διακρίνονται με βάση το ποιος ξεκινά πρώτα την αλληλεπίδραση, δηλαδή στη μία περίπτωση την αποστολή των δεδομένων ή στην άλλη την αίτηση τους[5].

### *2.1.1 Το μοντέλο push*

Το μοντέλο push έχει τα δεδομένα ως γνώμονα. Μόλις ο κόμβος εξυπηρέτησης αναγνωρίσει πως υπάρχει περιεχόμενο για δημοσίευση ή αποστολή σε άλλους κόμβους, θα στείλει το περιεχόμενο στους κόμβους, χωρίς να έχει ζητηθεί από αυτούς εκ των προτέρων. Δεδομένου ότι η αποστολή της πληροφορίας πραγματοποιείται πρώτη, το συγκεκριμένο μοντέλο προσδιορίζει τον αποστολέα ως τη δρώσα οντότητα.

#### *2.1.1.1 Πλεονεκτήματα και μειονεκτήματα*

Στο μοντέλο push, σε αντίθεση με το μοντέλο pull, οι δέκτες του περιεχομένου δεν ρωτούν τον κόμβο εξυπηρέτησης αν υπάρχει περιεχόμενο για αυτούς. Ως εκ τούτου, ο κόμβος εξυπηρέτησης δεν διακόπτεται συχνά από αιτήματα στα οποία δεν μπορεί να απαντήσει με χρήσιμη πληροφορία. Αυτό αποτρέπει την υπερφόρτωση του διακομιστή, η οποία μπορεί να προκληθεί από τα συνεχόμενα αιτήματα (flood), που μπορεί να καταλήξει ακόμη και σε αδυναμία του server να αποκριθεί ως και σε κρέμασμά του. Επίσης εξαλείφεται αποτελεσματικά το θέμα της ασφάλειας που αφορά την ευπάθεια σε επιθέσεις άρνησης υπηρεσιών.

Ως ένα μειονέκτημα, μπορούμε να αναφέρουμε το γεγονός πως ο παραλήπτης μπορεί να μην ενδιαφέρεται για την διάδοση των συγκεκριμένων δεδομένων σε αυτόν, και να μην είναι ικανοποιημένος, γιατί σε αυτήν την περίπτωση χρησιμοποιεί πόρους χωρίς λόγο.

### *2.1.2 Το μοντέλο pull*

Στο μοντέλο αυτό, ο κόμβος - αποστολέας περιμένει μέχρι ένας παραλήπτης να στείλει μια διακοπή, ζητώντας περιεχόμενο. Καθώς το αίτημα για πληροφορία πραγματοποιείται πρώτο, το συγκεκριμένο μοντέλο προσδιορίζει τον παραλήπτη ως τη δρώσα οντότητα. Ο κόμβος εξυπηρέτησης πρέπει να διαχειρίζεται το περιεχόμενο επιτυχώς, μέχρι κάποιος παραλήπτης να ζητήσει την αποστολή μέρους του. Ο παραλήπτης έχει την ελευθερία να αποφασίζει, σύμφωνα με το δικό του συμφέρον, το περιεχόμενο που θα ζητηθεί και επίσης, έχει τη δυνατότητα να εξετάζει την εγκυρότητα του αποστολέα πριν τη διαδικασία αίτησης. Ως εκ τούτου ο παραλήπτης έχει τον έλεγχο πάνω στο τι λαμβάνει.

#### *2.1.2.1 Πλεονεκτήματα και μειονεκτήματα*

Στο θέμα της ασφάλειας, ο παραλήπτης έχει ένα μεγάλο διάστημα χρόνου για την επαλήθευση της ταυτότητας του αποστολέα, στο χρονικό διάστημα που μεσολαβεί μεταξύ του αιτήματος και της παραλαβής του

περιεχομένου. Εκτός από το να καθίσταται επιρρεπής σε επιθέσεις άρνησης υπηρεσιών όμως, ο κόμβος εξυπηρέτησης επιβαρύνεται με την πολυπλοκότητα διαχείρισης του περιεχομένου, με την αποθήκευση του περιεχομένου μέχρι την απαίτηση του παραλήπτη, τον έλεγχο εάν ο αιτών είναι ο αρχικός προβλεπόμενος παραλήπτης του περιεχομένου και την παρακολούθηση εάν κάποια πληροφορία δε ζητηθεί ποτέ από κάποιον κόμβο, προκαλώντας τη συσσώρευση του περιεχομένου στην προσωρινή μνήμη, με αποτέλεσμα να γεμίζει και να μένει δεσμευμένη χωρίς λόγο.

### *2.1.3 Συνδυασμοί των δύο μοντέλων (αφαιρετικά)*

Υπάρχουν τέσσερις δυνατότητες, όσον αφορά τον τρόπο με τον οποίο μπορεί κανείς να αποκτήσει την πληροφορία, που θα αποσταλεί στο Twitter.

Η μία επιλογή είναι η εφαρμογή να αναζητά τα δεδομένα από την πηγή (pull), κάθε φορά που θέλουμε να γίνεται ανανέωση των κοινοποιήσεων. Αυτό μπορεί να γίνει δημιουργώντας ένα cronjob στο crontab του Linux για παράδειγμα, οπότε η εφαρμογή μας θα εκτελείται, θα αναζητά πληροφορία και θα κάνει νέες κοινοποιήσεις (εφόσον υπάρχουν νέες ανακοινώσεις) ανά τακτά χρονικά διαστήματα.

Η δεύτερη δυνατότητα είναι η εφαρμογή να δέχεται πληροφορίες από την πηγή (push), όταν υπάρχει κάτι καινούριο. Για να είναι αυτό δυνατό, πρέπει να αλλάξει ο κώδικας της πηγής, ώστε να αποστέλλεται η πληροφορία αυτόματα στην εφαρμογή, που θα είναι σε αναμονή μονίμως, περιμένοντας κάποια νέα ανακοίνωση. Για να επιτευχθεί κάτι τέτοιο, πρέπει η εφαρμογή να διατηρεί ένα κανάλι επικοινωνίας ανοικτό ανά πάσα στιγμή, ειδάλλως ενδέχεται να υπάρξει απώλεια πληροφορίας.

Αν η εφαρμογή ανακτά τα δεδομένα μέσω κάποιου ενδιάμεσου ανά τακτά χρονικά διαστήματα, μπορεί να έχουμε συνδυασμό push και pull. Η πηγή αποστέλλει μεν τα δεδομένα, σε έναν ενδιάμεσο δε, όπως έναν λογαριασμό ηλεκτρονικού ταχυδρομείου, άρα ακολουθείται το μοντέλο push. Στη συνέχεια η εφαρμογή που είναι υπεύθυνη για την επεξεργασία και μεταβίβαση της πληροφορίας θα την αντλήσει από το ενδιάμεσο σύστημα και όχι την πηγή, ωστόσο ακολουθείται το μοντέλο pull σε αυτή τη φάση. Ένας τρόπος με τον οποίο θα μπορούσε να υλοποιηθεί ένα τέτοιο σύστημα είναι με χρήση Message Queues (ουρές μηνυμάτων). [1]

Η τέταρτη επιλογή, είναι η ίδια η πηγή να κοινοποιεί μια ανακοίνωση με το που δημιουργείται κατευθείαν στο Twitter.

Εφόσον δεν είναι πάντα δυνατό να έχουμε πρόσβαση στον κώδικα της πηγής, δε μπορούμε να θεωρήσουμε δόκιμη καμία επιλογή, η οποία περιλαμβάνει push. Θεωρητικά, πρέπει να μπορούμε να χρησιμοποιούμε την εφαρμογή για διάφορες πηγές, με όσο το δυνατόν ελάχιστες προσαρμογές του κώδικά μας.

[1] *Σύμφωνα με έρευνα[22] των Πανεπιστημίων της Βομβάης και της Μασσαχουσέτης, πολλές φορές όταν είναι κρίσιμη η γρήγορη επικαιροποίηση των δεδομένων, πρέπει να χρησιμοποιείται κάποιου είδους συνδυασμός pull και push.*

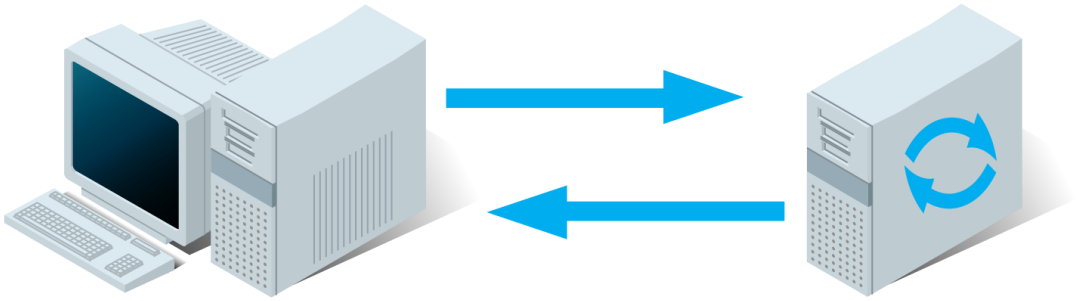
Ωστόσο θεωρείται σκόπιμο να αναλύσουμε κάποιες πιθανές εναλλακτικές λύσεις και τεχνικές παρακάτω.

## 2.1.4 Πιθανές υλοποιήσεις ανάκτησης πληροφορίας

### 2.1.4.1 Χρήση του πρωτοκόλλου http (και μόνον)

Σε αυτό το πρωτόκολλο η εφαρμογή αιτείται πληροφορία από τον διακομιστή (pull). Είναι η πιο απλή υλοποίηση, την οποία και επιλέξαμε, διότι η εφαρμογή μας δεν απαιτεί άμεση ανανέωση της πληροφορίας (δεν είναι κρίσιμη η ανανέωση σε πραγματικό χρόνο) και οποιαδήποτε άλλη περίπτωση θα επιβάρυνε τη διαδικασία χωρίς λόγο και θα δέσμευε περιττούς πόρους.

Μετά την σύνδεση μέσω TCP (δηλαδή την ολοκλήρωση του “3-way handshake”), η εφαρμογή ζητά κάποια πληροφορία από τον διακομιστή. Ο διακομιστής επεξεργάζεται το αίτημα, υπολογίζει την απάντηση και την αποστέλλει στην εφαρμογή. Με αυτήν την αποστολή ολοκληρώνεται όλη η διαδικασία απόκτησης της πληροφορίας.



**2.1 Πρωτόκολλο HTTP**

Το πρωτόκολλο αυτό χρησιμοποιείται σε όλες τις παρακάτω πιθανές υλοποιήσεις και αποτελεί τη βάση τους (πρώτο βήμα).

### 2.1.4.2 JSONP

Αλλιώς και “JSON with padding”, είναι μια τεχνική επικοινωνίας σε προγράμματα JavaScript, που εκτελείται ζητώντας δεδομένα από έναν διακομιστή. Για να μπορεί να λειτουργήσει, φυσικά πρέπει ο διακομιστής να γνωρίζει πως να απαντήσει στέλνοντας JSONP-formatted αποτελέσματα. Η τεχνική δεν πρέπει να συγχέεται με το JSON και δεν είναι δυνατή η υλοποίηση με αποστολή JSON-formatted απαντήσεων. Οι παράμετροι JSONP περνάνε ως παράμετροι σε ένα script και καθορίζονται από τον διακομιστή.

Για παράδειγμα:

```
<script type="application/javascript"
    src="http://server2.example.com/Users/1234?jsonp=parseResponse">
</script>
```

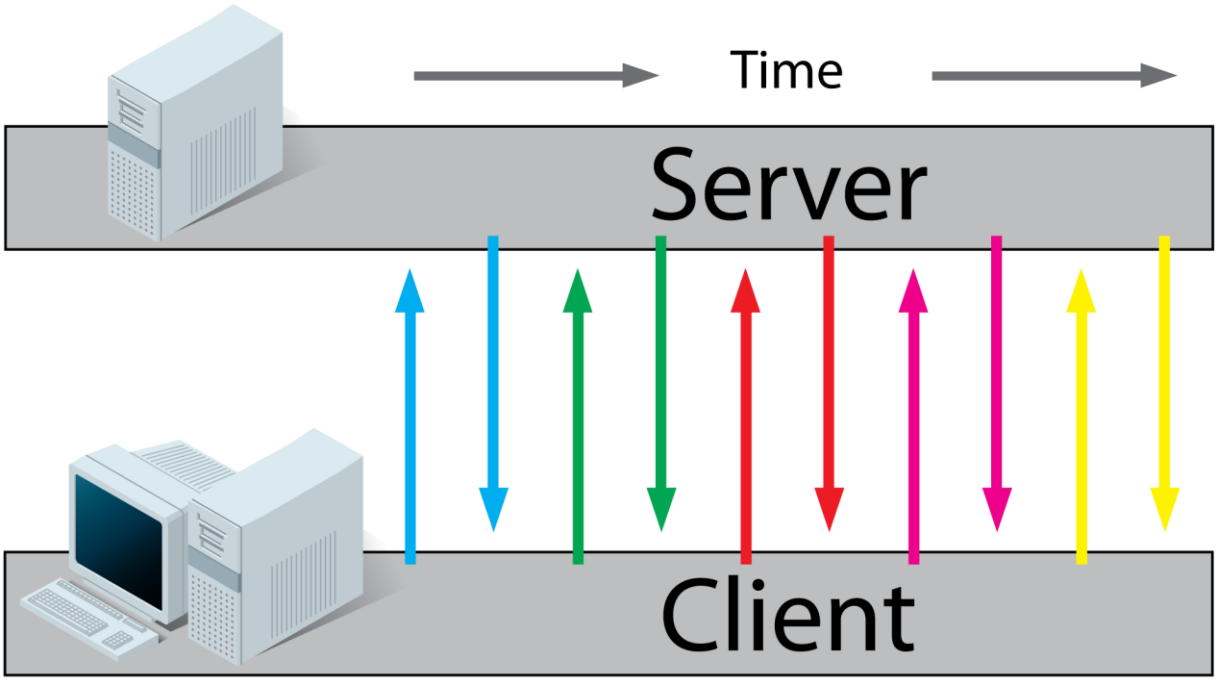
Η εφαρμογή στην πραγματικότητα θα κατεβάσει το συγκεκριμένο script και θα το εκτελέσει τοπικά, ώστε να λάβει την απάντηση στο αίτημα που έκανε.

### *2.1.4.3 AJAX (AJAJ & AHAH)*

Πρόκειται για ένα σύνολο τεχνικών Javascript, που χρησιμοποιούνται στην πλευρά του πελάτη για τη δημιουργία ασύγχρονων εφαρμογών. Μπορεί να χρησιμοποιηθεί για την αποστολή και παραλαβή δεδομένων από το διακομιστή ασύγχρονα, κάνοντας χρήση του XMLHttpRequest object. Αντίθετα με την ονομασία του αντικειμένου, η χρήση XML δεν είναι απαραίτητη[2]. Μπορεί να χρησιμοποιηθεί και JSON (AJAJ) και τα αιτήματα δεν είναι απαραίτητα ασύγχρονα. Επίσης, στο AHAH έχουμε την επιστροφή τμημάτων (X)HTML αντί XML και είναι ακόμη δυνατό να επιστρέφεται και απλό κείμενο.

### *2.1.4.4 AJAX Polling*

Σε αυτήν την περίπτωση, η εφαρμογή πάλι αιτείται πληροφορία από το διακομιστή και ισχύουν όλα όσα ισχύουν και στην χρήση του απλού http. Η διαφορά είναι, ότι μετά την αποστολή της πληροφορίας η εφαρμογή εκτελεί κώδικα javascript ο οποίος κάθε τόσο ζητά από το διακομιστή ένα αρχείο. Ο διακομιστής κάνει τους υπολογισμούς και αποστέλλει το αρχείο, σαν να είχαμε απλά http. Η διαδικασία αυτή επαναλαμβάνεται επ' αόριστον.



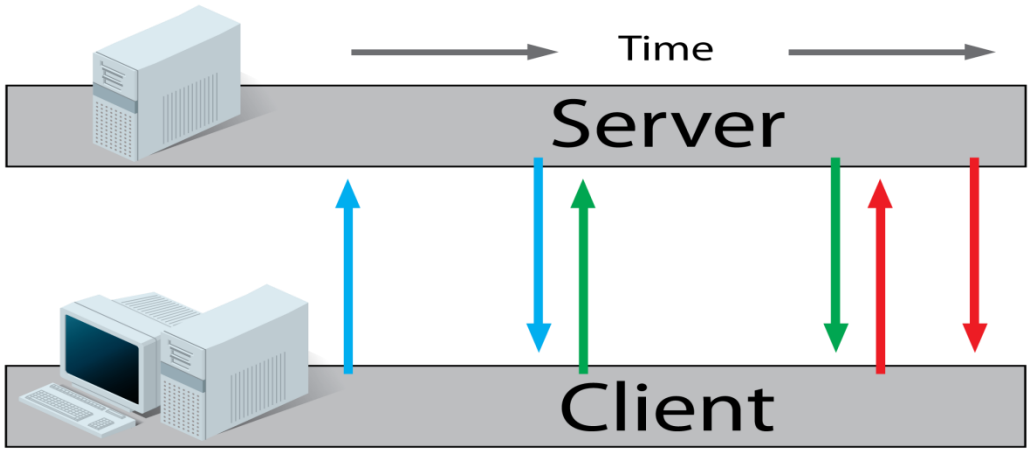

**2.2 Ajax Polling**

*2 Στις πρώτες μορφές ήταν απαραίτητη η χρήση XML και έτσι προέκυψε η ονομασία.*

Το τεράστιο μειονέκτημα είναι ότι κάθε πελάτης - εφαρμογή μπορεί να αιτείται νέα πληροφορία κάθε λίγο, ενώ αυτή δεν υπάρχει. Και όλο αυτό οδηγεί σε μια απίστευτη σπατάλη πόρων και στις δύο πλευρές.

### *2.1.4.5 AJAX Long-Polling*

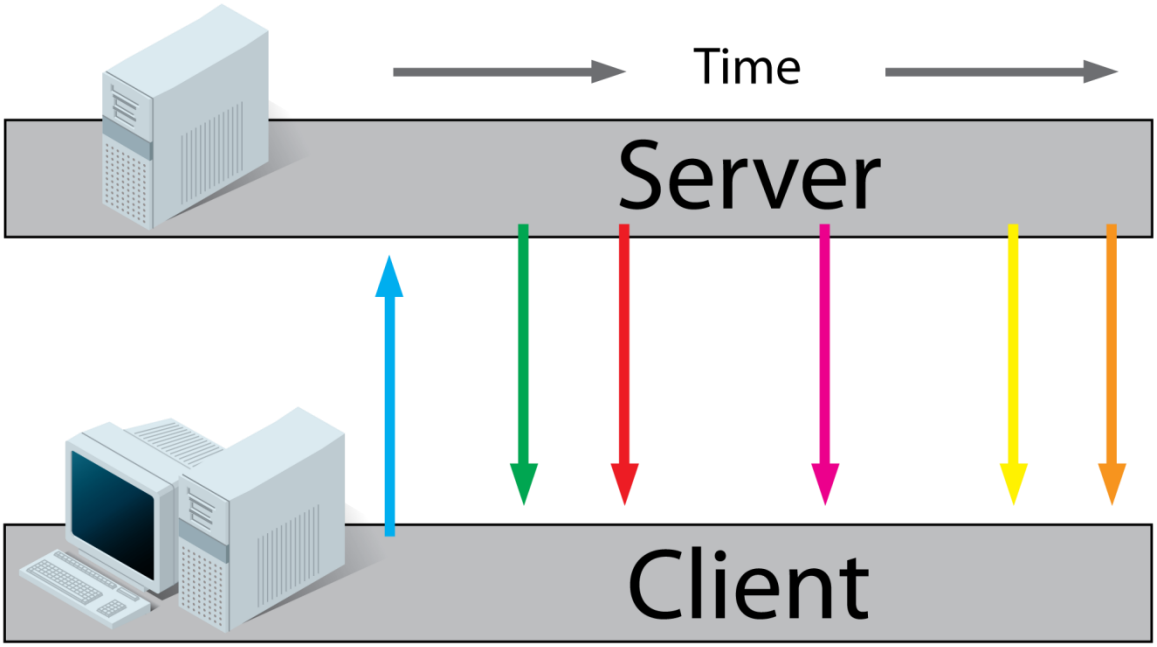
Η διαφορά του AJAX Long-Polling από το AJAX Polling έγκειται στο γεγονός, πως όταν η εφαρμογή ζητήσει την ανανέωση των δεδομένων, ο διακομιστής δε στέλνει άμεσα απάντηση, αλλά περιμένει έως ότου να υπάρχει ανανεωμένη πληροφορία. Όταν υπάρχει νέα πληροφορία, τότε την αποστέλλει στην εφαρμογή. Μετά την παραλαβή της νέας πληροφορίας, η εφαρμογή στέλνει αμέσως ένα νέο αίτημα στο διακομιστή για ανανέωση της πληροφορίας. Η διαδικασία αυτή επαναλαμβάνεται επ' αόριστον. Η συγκεκριμένη διαδικασία είναι σαφώς πιο οικονομική από θέμα πόρων, σε σχέση με το AJAX Polling.




**2.3 Ajax Long-Polling**

### *2.1.4.6 HTML5 Server Sent Events (SSE) / EventSource*

Στο πρώτο βήμα και πάλι η εφαρμογή ζητά πληροφορία στέλνοντας ένα http request στο διακομιστή. Στη συνέχεια εκτελείται κώδικας στην εφαρμογή, που ανοίγει μια σύνδεση με τον διακομιστή. Κάθε φορά που υπάρχει νέα πληροφορία, ο διακομιστής την αποστέλλει στην εφαρμογή σαν event. Η συγκεκριμένη μέθοδος χρησιμοποιείται όταν χρειαζόμαστε μονόδρομη κίνηση πληροφοριών από τον διακομιστή προς την εφαρμογή και ο διακομιστής πρέπει να εκτελεί ένα event loop.

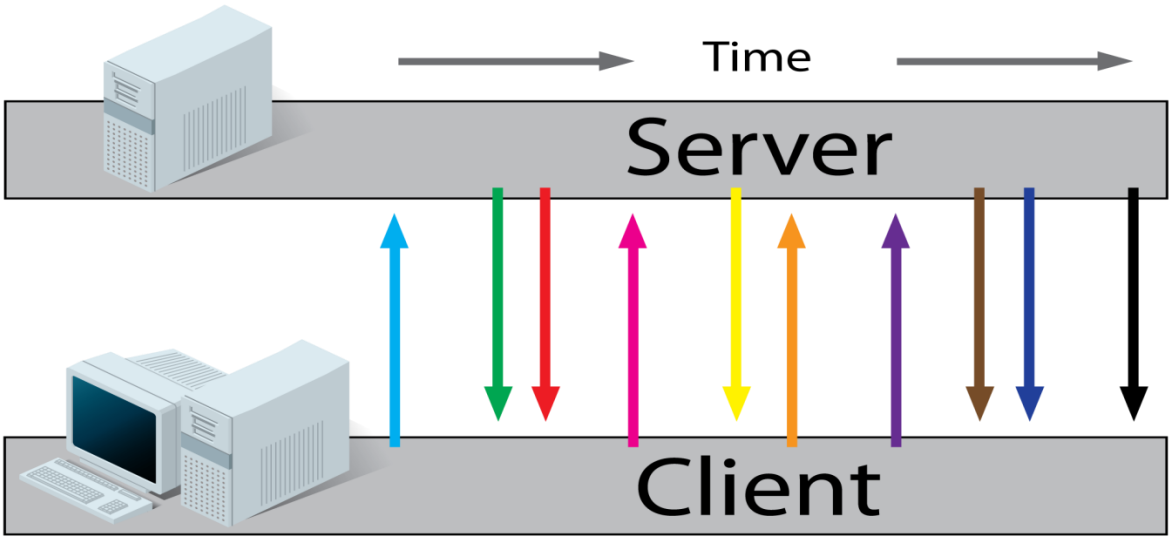



**2.4 HTML 5 Sever Sent Events (SSE)**

### *2.1.4.7 HTML5 Websockets*

Τα Websockets είναι μια νέα σχετικά τεχνολογία και πολλά υποσχόμενη. Χρησιμοποιείται για αμφίδρομη επικοινωνία σε σχεδόν πραγματικό χρόνο μεταξύ διακομιστών και εφαρμογών. Έχει την ίδια λειτουργία με τα Server-Side Events, με τη διαφορά ότι το κανάλι επικοινωνίας που ανοίγει χρησιμοποιείται όχι μόνο για την αποστολή δεδομένων από το διακομιστή στην εφαρμογή, αλλά και αντίστροφα. Βρίσκει εφαρμογές σε διαδικτυακά παιχνίδια, στα Google Documents και άλλες υπηρεσίες που χρειάζονται αμφίδρομη επικοινωνία για ανταλλαγή δεδομένων σε (σχεδόν) πραγματικό χρόνο. Και σε αυτήν την περίπτωση, ο διακομιστής πρέπει να εκτελεί ένα event loop.

Μπορεί κανείς να πει, ότι συνδυάζοντας SSE και AJAX (για αποστολή νέων αιτημάτων από την εφαρμογή) αποκτάμε μια λειτουργικότητα παρόμοια με τα Websockets. Βέβαια, τα τελευταία έχουν και άλλες χρήσιμες εφαρμογές, οι οποίες όμως δε θεωρείται σκόπιμο να αναλυθούν εδώ.




**2.5 HTML 5 Websockets**

### *2.1.4.8 Comet*

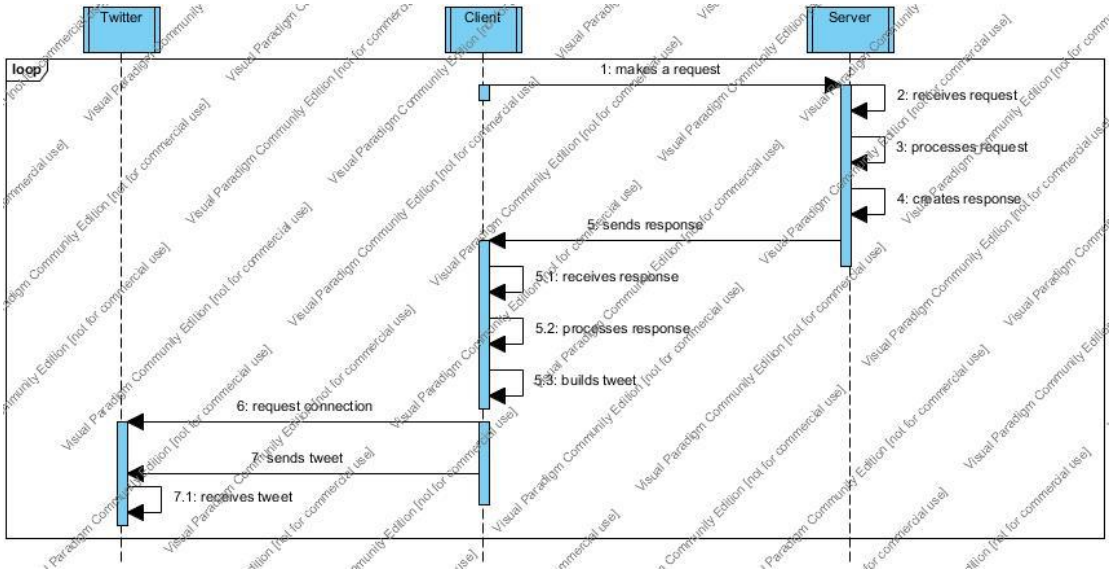
Πρόκειται για μια υλοποίηση, η οποία ανοίγει ένα κανάλι επικοινωνίας, κάνοντας χρήση απλής http. Όταν όμως αποσταλλεί το αίτημα από την εφαρμογή, ο διακομιστής στέλνει απάντηση χωρίς να δηλώσει ότι έχει ολοκληρωθεί. Έτσι, η σύνδεση παραμένει ανοιχτή για όσο χρόνο θεωρείται σκόπιμο! Για να γίνονται κατανοητά τα μηνύματα από την εφαρμογή, αντί για τέλος μηνύματος ο διακομιστής στέλνει ένα μήνυμα του τύπου "EndOfCometMessage\n" στο τέλος κάθε νέου κομματιού πληροφορίας.

## *2.1.5 Ροές στα δύο βασικά μοντέλα*

Στη συνέχεια περιγράφονται οι ροές και παρουσιάζονται τα αντίστοιχα διαγράμματα ακολουθίας για τα δύο πιθανά μοντέλα. Γίνεται αξιολόγηση των δύο μοντέλων, η οποία είναι μείζονος σημασίας και θα οδηγήσει τελικά στην επιλογή του καταλληλότερου για την ανάπτυξη της συγκεκριμένης υπηρεσίας και την επίτευξη του αντίστοιχου στόχου.

| Ροή στο μοντέλο pull | | |
|---|---|---|
| TWITTER | CLIENT | SERVER |
| ...stands by for requests... | | ...stands by for requests... |
| ... | makes a request -> | -> receives request |
| ... | waits for response | processes request |
| ... | ... | builds response |
| ... | receives response <- | <- sends response |
| ... | processes response | ...stands by for requests... |
| ... | builds tweet | ... |
| ... | <- requests connection | ... |
| receives tweet <- | <- sends tweet | ... |
| ...stands by for requests... | | ... |

**2.6 Ροή στο μοντέλο pull**

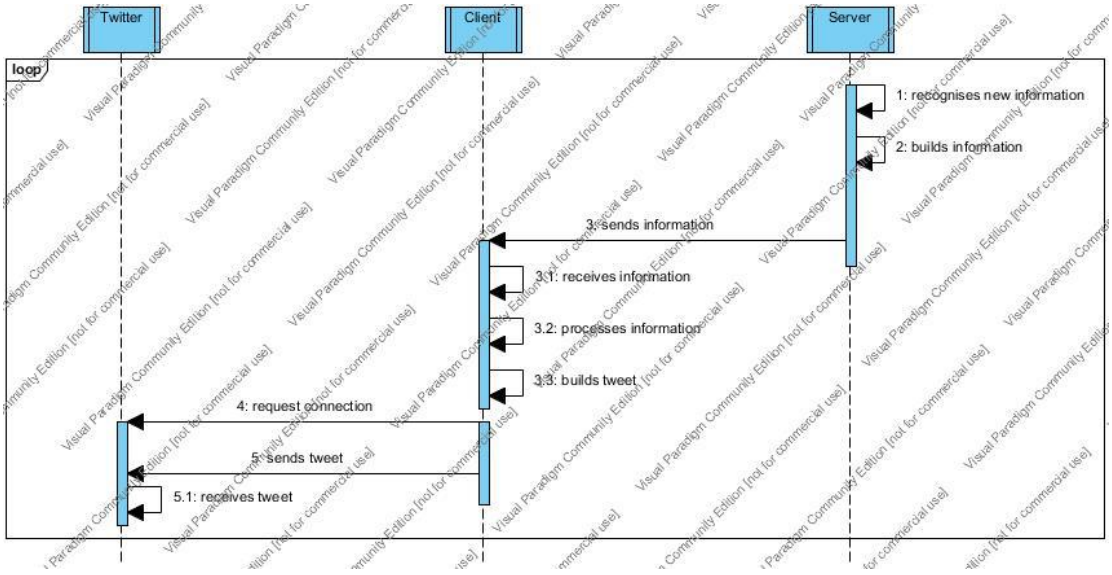



**2.7 Διάγραμμα ακολουθίας στο μοντέλο pull**

Είναι προφανές ότι στο μοντέλο pull η διαδικασία εκκινείται από την ενδιάμεση εφαρμογή, σε αντίθεση με τα παρακάτω σχήματα, όπου φαίνεται ότι η διαδικασία εκκινείται από την πηγή των δεδομένων. Η διαφοροποίηση αυτή είναι σημαντική, γιατί στην περίπτωση του push το ενδιάμεσο σύστημα πρέπει να είναι συνεχώς σε αναμονή, μέχρι να δεχτεί δεδομένα. Αυτό στην περίπτωση μας για παράδειγμα, μπορεί να σημαίνει, πως αν δεν υπάρχει κάποια ανακοίνωση για μερικούς μήνες, στο συνολικό διάστημα αυτό η εφαρμογή θα είναι συνεχώς ενεργή, ενώ δε θα παράγει ουσιαστικά κανένα έργο.

| Ροή στο μοντέλο push | | |
|---|---|---|
| TWITTER | CLIENT | SERVER |
| ...stands by for requests... | ...stands by for data... | |
| ... | ... | recognizes new information |
| ... | ... | builds information |
| ... | <- receives information | <- sends information |
| ... | processes information | |
| ... | builds tweet | |
| ... | <- requests connection | |
| <- receives tweet | <- sends tweet | |
| ...stands by for requests... | ...stands by for data... | |

**2.8 Ροή στο μοντέλο push**




**2.9 Διάγραμμα ακολουθίας στο μοντέλο push**

Όπως φαίνεται στα παραπάνω σχήματα, στην πρώτη περίπτωση (pull) ο διακομιστής περιμένει κάποιο σήμα από τον παραλήπτη. Στη δεύτερη (push), ο παραλήπτης περιμένει για κάποιο σήμα από την πηγή. Οπότε ανάλογα με την περίπτωση, μπορεί να συμφέρει είτε η μία λύση είτε η άλλη. Εύκολα όμως γίνεται φανερό, ότι στην δεύτερη περίπτωση πρέπει να υπάρχουν τρία συστήματα μονίμως ενεργά, για να ολοκληρωθεί η διαδικασία. Στην πρώτη, αυτό ισχύει μόνο για δύο συστήματα, τα οποία ούτως ή άλλως παραμένουν (κατά κανόνα) ενεργά συνεχώς. Η ενδιάμεση εφαρμογή δεν είναι απαραίτητο να είνα μόνιμα ενεργή. Ενεργοποιείται μόνο για λίγο ανά τακτά χρονιά διαστήματα. Αυτό προφανώς οδηγεί στην εξοικονόμηση πόρων, οι οποίοι μπορούν να διατεθούν σε άλλες διεργασίες.

### *2.1.6 Αξιολόγηση μοντέλων*

Οι δύο βασικοί τρόποι αναμετάδοσης της πληροφορίας δεν έχουν τρομερές διαφορές ποσοτικά. Ποιοτικά μπορεί να έχουν κάποιες διαφορές, τις οποίες πρέπει να λάβουμε υπόψη, ανάλογα με τις απαιτήσεις.

Όταν ανακαλούμε τα δεδομένα από την πηγή (pull), τότε μπορούμε να το κάνουμε όποτε μας βολεύει. Ενώ μιλάμε όμως για ανακοινώσεις, οι οποίες πολλές φορές πρέπει να γνωστοποιούνται άμεσα, αυτό σημαίνει πως δεν είναι λογικό να εκτελούμε την εφαρμογή μια φορά τη μέρα για παράδειγμα.

Επίσης, στην περίπτωση που ανακαλούμε δεδομένα, η εφαρμογή εκτελείται όποτε έχει προγραμματιστεί, το οποίο σημαίνει ότι δεσμεύει πόρους από το σύστημα κάποια δεδομένα χρονικά διαστήματα (συγκριτικά πολύ μικρά). Αν υπάρχει πρόβλημα πόρων, αυτό είναι θετικό. Αντίθετα, στην περίπτωση που η πηγή μας αποστέλλει τα δεδομένα και αυτά αναμεταδίδονται άμεσα, τότε πρέπει μονίμως να υπάρχει ένα πρόγραμμα που θα περιμένει να δεχτεί κάποιο νέο σήμα και όποτε δεν ισχύει αυτό, οδηγούμαστε σε απώλεια δεδομένων. Αυτά τα δεδομένα, ανάλογα με την

περίπτωση, μπορεί να μην είναι δυνατό να αποκτηθούν στη συνέχεια με κανέναν τρόπο.

Μια άλλη διαφοροποίηση είναι πως, αν κάνουμε pull κάθε τέταρτο της ώρας για παράδειγμα, τότε σε περίπτωση που δε μπορέσουμε να συνδεθούμε στην πηγή για οποιονδήποτε λόγο κάποια φορά, την επόμενη φορά (ή τελικά όταν είναι δυνατή αυτή η σύνδεση) θα γίνουν σίγουρα νέες αναρτήσεις (εφόσον υπάρχουν νέες ανακοινώσεις) και μάλιστα όλες οι νέες ανακοινώσεις θα αναμεταδοθούν ως νέες κοινοποιήσεις. Στην περίπτωση του push από την άλλη μεριά, δεν είναι τόσο απλό, όπως παραπάνω το ενδεχόμενο, ώστε να λύνεται με το να προσπαθήσουμε ξανά να ανακτήσουμε πληροφορία. Πρέπει να υπάρχει κάποιος μηχανισμός ασφαλείας (στην πηγή) για να μην χάνονται πιθανές αναρτήσεις.

Το άμεσα ορατό πλεονέκτημα της δεύτερης λύσης, απέναντι στην πρώτη, είναι ότι η ανάρτηση μιας νέας πληροφορίας γίνεται πιο γρήγορα. Εφόσον όμως μπορούμε να ελέγχουμε συνέχεια για νέες ανακοινώσεις και δεν είναι τραγικό αν αργήσει η ανάρτηση μερικά λεπτά, δε θεωρείται σοβαρό αυτό το πλεονέκτημα.

Ένα πιο κρυφό πλεονέκτημα της δεύτερης λύσης είναι η δυνατότητα της πηγής να παρέχει στην εφαρμογή τα δεδομένα που θέλει αυτή μόνο. Σε ορισμένες περιπτώσεις αυτό μπορεί να είναι σημαντικό. Στη δική μας περίπτωση όμως, ούτε αυτό αποτελεί πλεονέκτημα, εφόσον έχουμε να κάνουμε με δημόσιες ανακοινώσεις, δηλαδή ορατές σε όλους, άρα δεν υπάρχει λόγος να αποκρύπτεται πληροφορία.

Το πιο σημαντικό μειονέκτημα του push όμως, είναι πως πρέπει να έχουμε πρόσβαση στον κώδικα της πηγής. Αυτό δεν κάνει την εφαρμογή μας ευέλικτη, μιας και κάποια στιγμή μπορεί να χρειαστεί να δημιουργήσουμε αναρτήσεις από περιεχόμενο μιας ιστοσελίδας ενός εξυπηρέτη που δε μας ανήκει για παράδειγμα. Με την άλλη μέθοδο αυτό δεν είναι πρόβλημα, γιατί πρέπει και αρκεί να μπορούμε να διαβάσουμε τη νέα ιστοσελίδα - πηγή.

### *2.1.7 Συμπέρασμα*

Για όλους τους παραπάνω λόγους, επιλέξαμε να χρησιμοποιήσουμε τη μέθοδο του pull, αναζητώντας πληροφορία από την πηγή, αυτόματα βέβαια, σε δικό μας ορισμένο χρόνο. Η άλλη λύση θα ήταν ίσως μονόδρομος, αν η υπηρεσία είχε να κάνει με δεδομένα των οποίων η άμεση μετάδοση ήταν ζωτικής σημασίας.

## 2.2 Επεξεργασία πληροφορίας

Ο τρόπος με τον οποίο επεξεργαζόμαστε την ανακτημένη πληροφορία εξαρτάται από την ποιότητά της (και την ποσότητά της, εφόσον έχουμε τον περιορισμό των 140 χαρακτήρων). Υπάρχουν πολλές δυνατότητες για την επεξεργασία της και για τη δημιουργία νέων κοινοποιήσεων, και προφανώς εξαρτώνται από το είδος της πηγής. Αν η πηγή είναι μια ιστοσελίδα (ένα blog για παράδειγμα), μπορούμε να χρησιμοποιήσουμε HTML parsers ή XML

parsers. Αν η πηγή είναι ένα μήνυμα ηλεκτρονικού ταχυδρομείου, μπορούμε να χρησιμοποιήσουμε mail parsers. Η πηγή μπορεί να είναι ακόμη μια βάση δεδομένων, η οποία μπορεί να ελέγχεται τακτικά για νέο περιεχόμενο, οπότε, αν για παράδειγμα αυτά που έχουμε ανακτήσει είναι απλά strings, χρησιμοποιούμε τους αντίστοιχους μηχανισμούς για να παράγουμε την προς ανάρτηση κοινοποίηση. Επίσης μπορεί να έχουμε ως πηγή κάποιο RSS feed ή ο,τιδήποτε άλλο, οπότε προσαρμόζουμε ανάλογα την εφαρμογή.

## 2.3 Ανάρτηση κοινοποιήσεων

Υπάρχει μόνο ένας τρόπος να δημιουργήσουμε νέες κοινοποιήσεις στο Twitter μέσα από μια εφαρμογή. Υποχρεωνόμαστε να χρησιμοποιήσουμε κάποιο από τα API που παρέχεται από το ίδιο το Twitter. Όλες οι υφιστάμενες εφαρμογές κάνουν το ίδιο φυσικά. Κάποιες από αυτές θα αναλυθούν όσο το δυνατόν περισσότερο στο επόμενο κεφάλαιο.

# 3 Υφιστάμενες υπηρεσίες

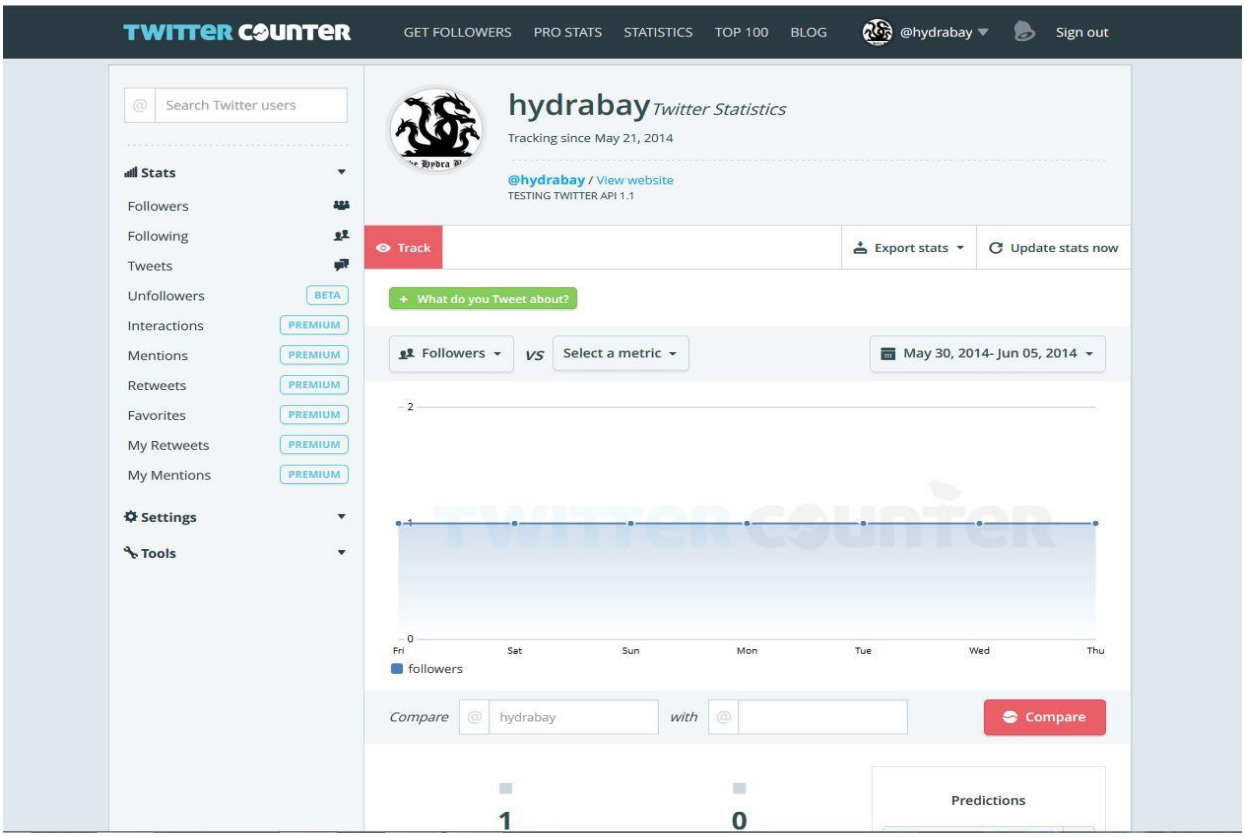
Έχουν αναπτυχθεί αρκετές υπηρεσίες, οι οποίες με τον ένα ή τον άλλο τρόπο δημιουργούν νέες αναρτήσεις στο Twitter. Παρακάτω θα επεξηγήσουμε τον τρόπο λειτουργίας τους και θα τις αξιολογήσουμε, με βάση τη δυνατότητα παραμετροποίησής τους από το χρήστη, τους περιορισμούς λειτουργίας ως προς το περιβάλλον στο οποίο εκτελούνται, την επεκτασιμότητά τους, το αν αποτελούν ελεύθερο λογισμικό ανοιχτού κώδικα και αν παρέχονται και μπορούν να χρησιμοποιούνται δωρεάν.

## 3.1 TwitterMail

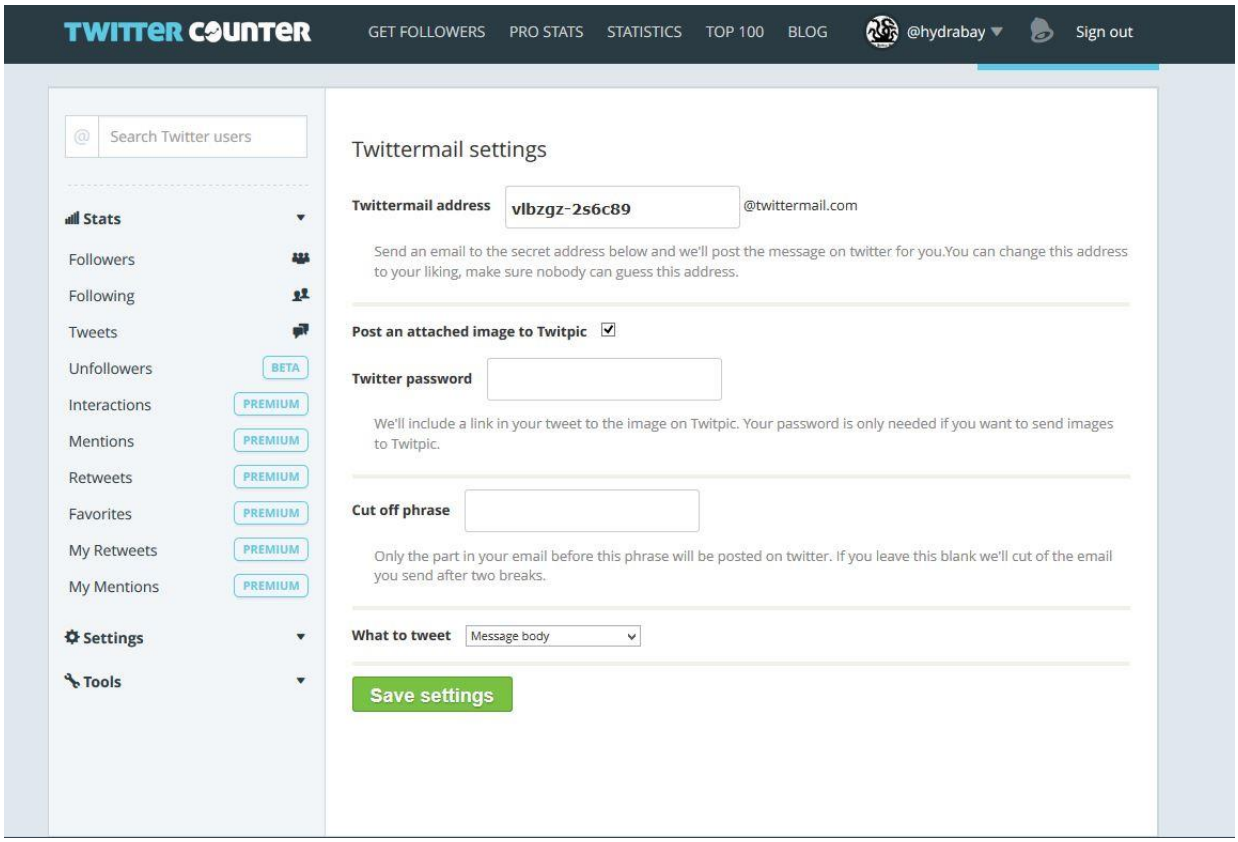
Ο χρήστης του TwitterMail[25] μπορεί να συνδέσει ένα λογαριασμό email με το λογαριασμό του στο Twitter. Κάθε μήνυμα που αποστέλλεται στο συγκεκριμένο λογαριασμό αναπαράγεται στο Twitter. Η εφαρμογή προσφέρει επίσης την δυνατότητα αποστολής email σε προσωπικό λογαριασμό όταν κάποιος follower στείλει στο χρήστη προσωπικό μήνυμα (δηλαδή λειτουργεί και αντίστροφα).

Το TwitterMail προφανώς χρησιμοποιεί pull για να δημιουργήσει νέες αναρτήσεις, εφόσον δε γίνεται κάποια επέμβαση στην υπηρεσία ηλεκτρονικού ταχυδρομείου. Ένα πιθανό πρόβλημα μπορεί να είναι το γεγονός, πως ένα μήνυμα ηλεκτρονικού ταχυδρομείου μπορεί να έχει περισσότερους από 140 χαρακτήρες, οπότε και θα χάνεται πληροφορία αν ισχύει κάτι τέτοιο. Άρα είναι ευθύνη του ίδιου του χρήστη να φροντίζει τα μηνύματα που θέλει να αναρτηθούν να έχουν μέγιστο μήκος 140 χαρακτήρων. Αυτή η μετάθεση ευθύνης στο χρήστη αποτελεί προφανές μειονέκτημα.

Όσον αφορά τα κριτήρια αξιολόγησης που θέσαμε, η συγκεκριμένη εφαρμογή είναι κλειστού κώδικα, δεν επιβάλλει περιορισμούς ως προς το περιβάλλον του χρήστη, διότι είναι εφαρμογή ιστού, επιτρέπει περιορισμένη παραμετροποίηση και δεν είναι δυνατό να επεκταθεί από τρίτους.




**Εικόνα 3.1 Κεντρική οθόνη διαχείρισης του TwitterMail**




**Εικόνα 3.2 Παραμετροποίηση λογαριασμού στο TwitterMail**

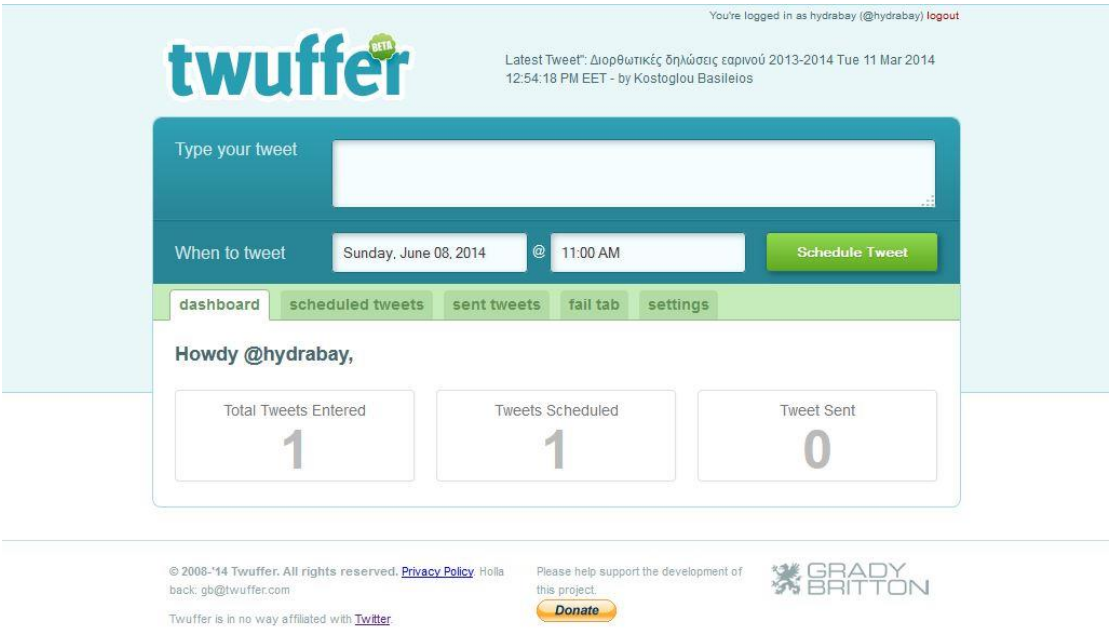
## 3.2 Twuffer TweetLater

Ο χρήστης της εφαρμογής[26] αυτής μπορεί να δημιουργεί νέες κοινοποιήσεις, οι οποίες αναρτώνται σε δεδομένο μελλοντικό χρόνο. Πιθανότατα χρησιμοποιείται μια βάση δεδομένων για να αποθηκεύσει τα προς ανάρτηση tweets.

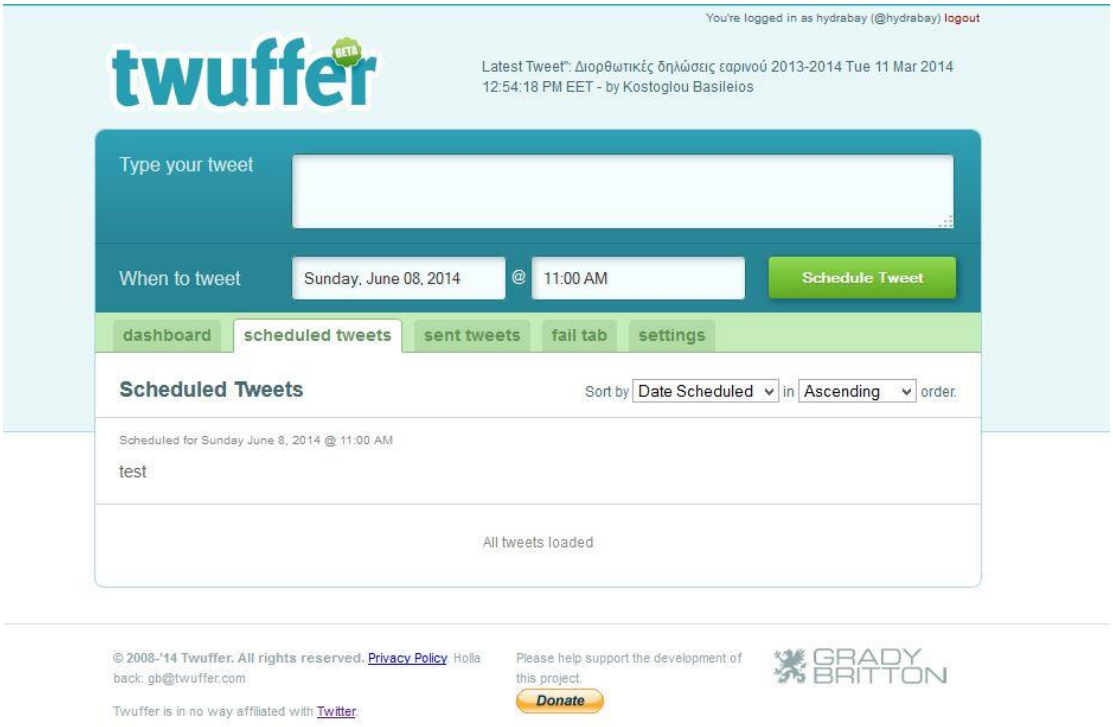
Αποτελεί ένα πολύ ενδιαφέρον εργαλείο, στο οποίο μπορεί να ανατρέξει ο χρήστης ανά πάσα στιγμή και να ελέγξει για πιθανές αποτυχίες. Μπορεί επίσης να δει τις προηγούμενες αναρτήσεις που έχουν γίνει από αυτήν την πηγή και να ορίσει να γίνονται αναρτήσεις με βάση χρονοδιάγραμμα! Είναι υποχρεωτικό να οριστεί η χρονική ζώνη του χρήστη φυσικά, για να γίνονται οι αναρτήσεις τη σωστή ώρα.

Στην περίπτωση αυτής της υπηρεσίας, ενδεχομένως γίνεται ανά τακτά χρονικά διαστήματα σάρωση της βάσης για μηνύματα που πρέπει να ανακοινωθούν (pull), αλλά το πιο πιθανό είναι να γίνεται push από τη βάση όταν φτάσει η στιγμή ανάρτησης ενός μηνύματος.

Η υπηρεσία αυτή, αν και είναι εφαρμογή κλειστού κώδικα, παρέχεται δωρεάν και δεν παρέχει περαιτέρω παραμετροποιήσεις πέραν της χρονικής στιγμής που θα γίνονται οι αναρτήσεις, ούτε είναι δυνατό να επεκταθεί.



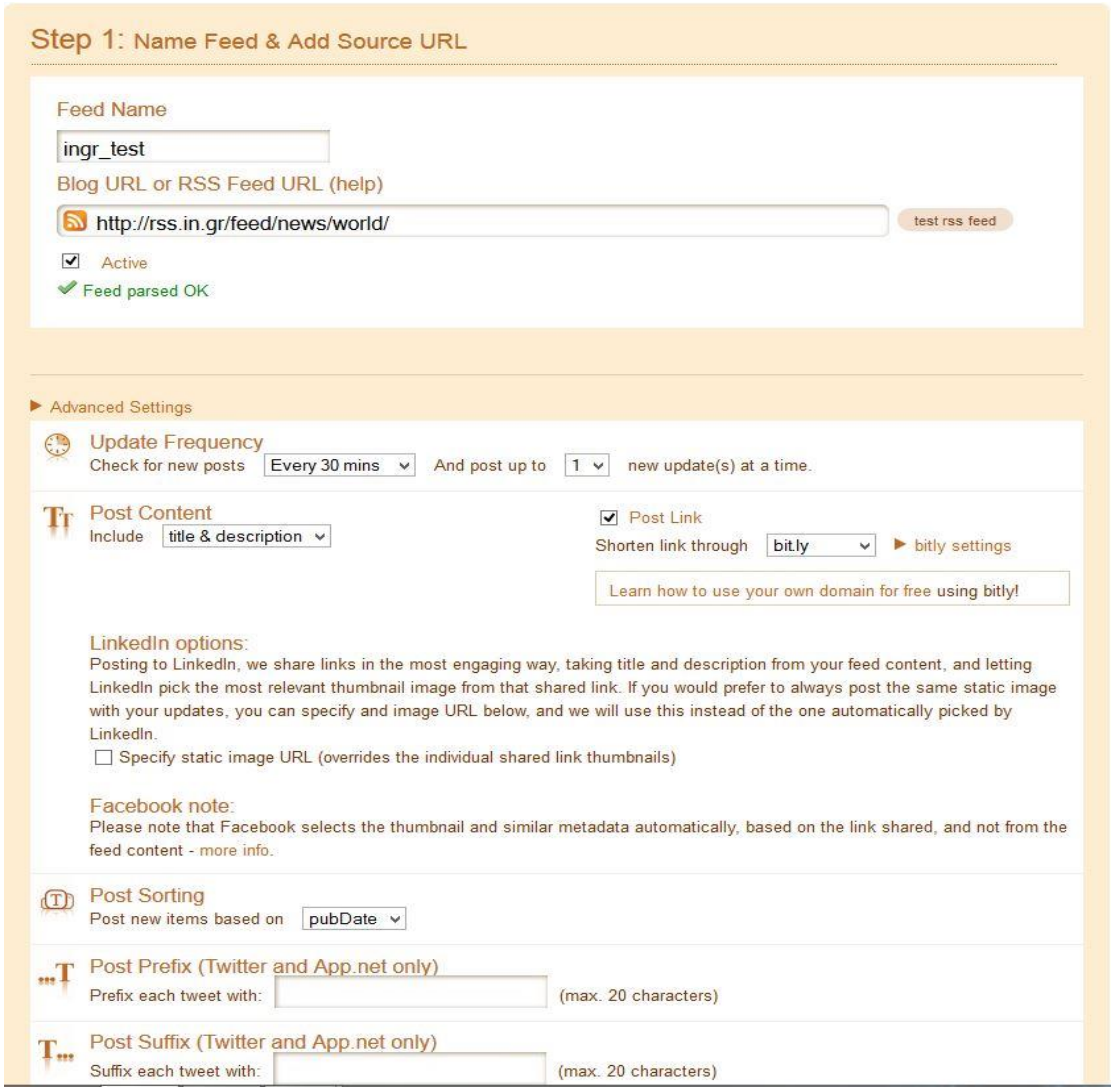

**Εικόνα 3.3 Κεντρική οθόνη διαχείρισης του twuffer**




**Εικόνα 3.4 Οθόνη διαχείρισης προγραμματισμένων αναρτήσεων στο twuffer**

## 3.3 Twitterfeed

Είναι μια δωρεάν υπηρεσία[27], που συνδέει ένα RSS feed ή ακόμη και ένα blog με ένα λογαριασμό στο Twitter, οπότε και αναμεταδίδονται αυτόματα και αναρτώνται στο Twitter τα νέα μηνύματα. Επιπλέον, είναι δυνατή η σύνδεση με περισσότερα κοινωνικά δίκτυα, όπως το Facebook και το LinkedIn - το όνομα της εφαρμογής είναι ίσως κάπως παραπλανητικό. Η εφαρμογή παρέχει κάποιου είδους στατιστικές σε σχέση με το ίδιο το feed (ή τα feeds, εφόσον μπορούν να συνδεθούν και περισσότερες από μία πηγές δεδομένων) και τη δυνατότητα καθορισμού του ρυθμού ελέγχου για νέα προς ανάρτηση (οπότε γίνεται pull) και κάποιες κατά περίπτωση ρυθμίσεις για κάθε λογαριασμό κοινωνικής δικτύωσης. Πέρα από αυτό δεν είναι δυνατό να γίνουν άλλες ρυθμίσεις ή επεκτάσεις.




**Εικόνα 3.5 Παραμετροποίηση λογαριασμού στο TwitterFeed**


**Εικόνα 3.6 Σύνδεση με τον λογαριασμό του Twitter για την ανάρτηση του feed**

## 3.4 Buffer

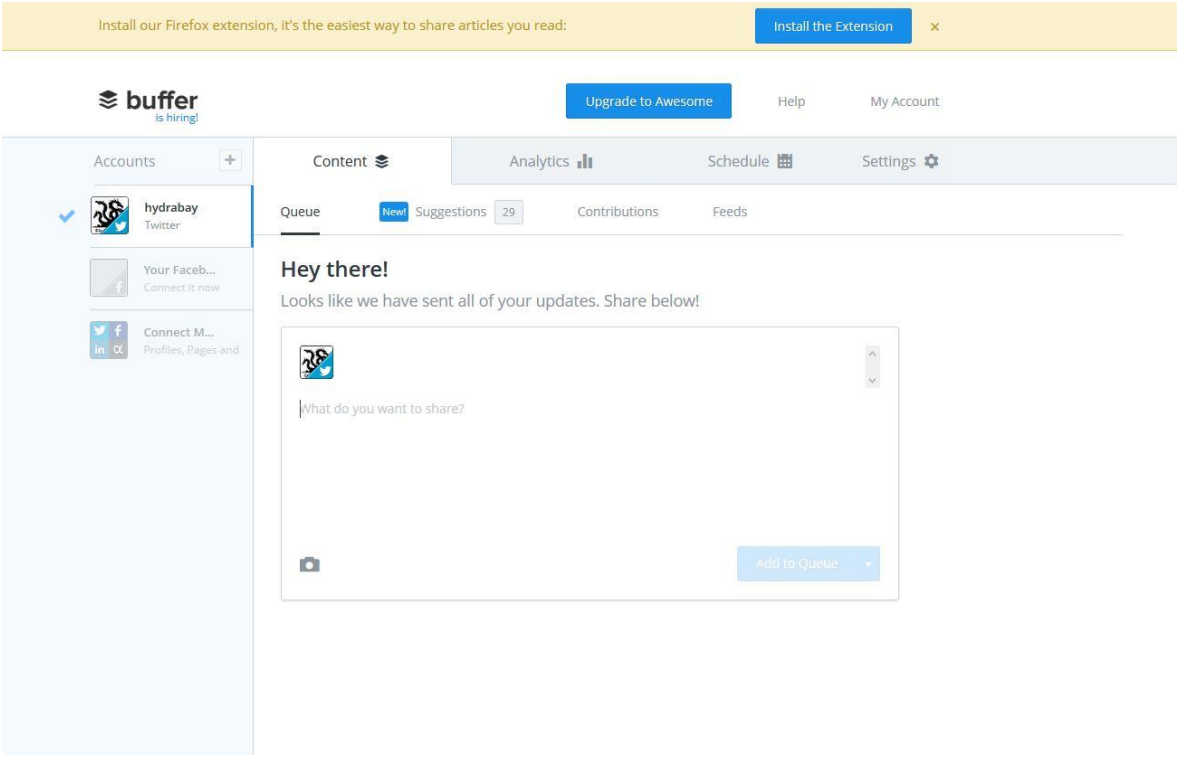
Το Buffer[28] είναι μια υπηρεσία, που προσφέρεται σε δύο βασικές εκδόσεις, την Individual και την Corporate. Η πρώτη, την οποία και δοκιμάσαμε, απευθύνεται στους κοινούς χρήστες ενώ η δεύτερη στις επιχειρήσεις. Η Individual έκδοση είναι δωρεάν ενώ υπάρχει και δυνατότητα αναβάθμισης στην έκδοση Awesome, στην οποία, μεταξύ άλλων, προσφέρονται πολλών ειδών στατιστικά και δυνατότητα ανάρτησης με χρονοδιάγραμμα. Στην δωρεάν έκδοση ο χρήστης έχει τη δυνατότητα να κάνει δημοσίευση αναρτήσεων βάση των trends[3], τα οποία το Buffer του προβάλλει, με αρκετά λεπτομερή τρόπο. Ο χρήστης μπορεί επίσης να συνδέσει ένα RSS feed οπότε τα νέα μηνύματα αναμεταδίδονται αυτόματα και αναρτώνται στο Twitter (push).

Επιπλέον είναι δυνατή η σύνδεση με περισσότερα κοινωνικά δίκτυα, όπως το Facebook, το LinkedIn και το Google+.

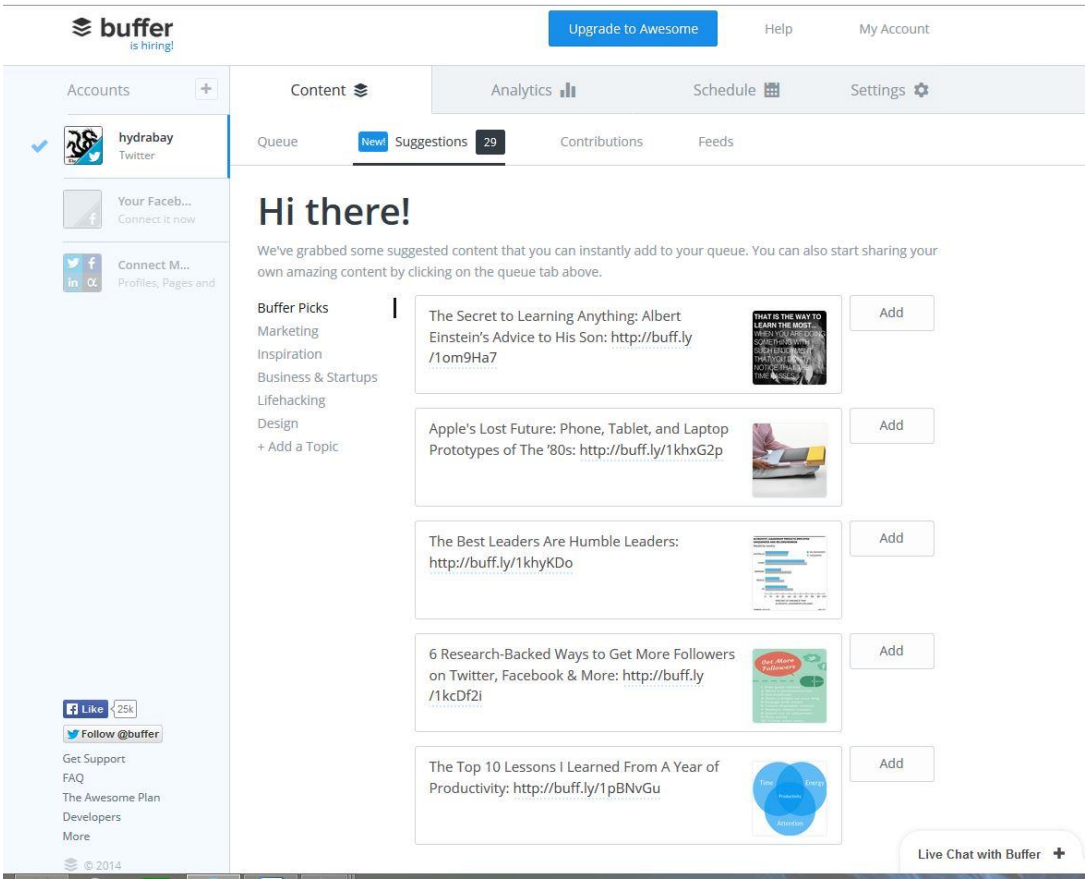
Πέρα από αυτές τις ρυθμίσεις δεν είναι δυνατό να γίνουν άλλες, ούτε γίνονται επεκτάσεις.

Να σημειωθεί πως το περιβάλλον εργασίας του Buffer είναι αρκετά φιλικό, τόσο στην δωρεάν όσο και στις επί πληρωμήν εκδόσεις, καθώς είναι πολύ κατατοπιστικό, όπως μπορεί να παρατηρήσει κανείς και στις εικόνες παρακάτω. Παρέχεται επίσης επέκταση για το Google Chrome και το Firefox η οποία συνεργάζεται με το πρόγραμμα για απλοποίηση της διαδικασίας ανάρτησης.

[3] Trends: θεματολογίες που έχουν αποκτήσει προσωρινά μεγάλη δημοσιότητα




**Εικόνα 3.7 Κεντρική οθόνη διαχείρισης του Buffer**

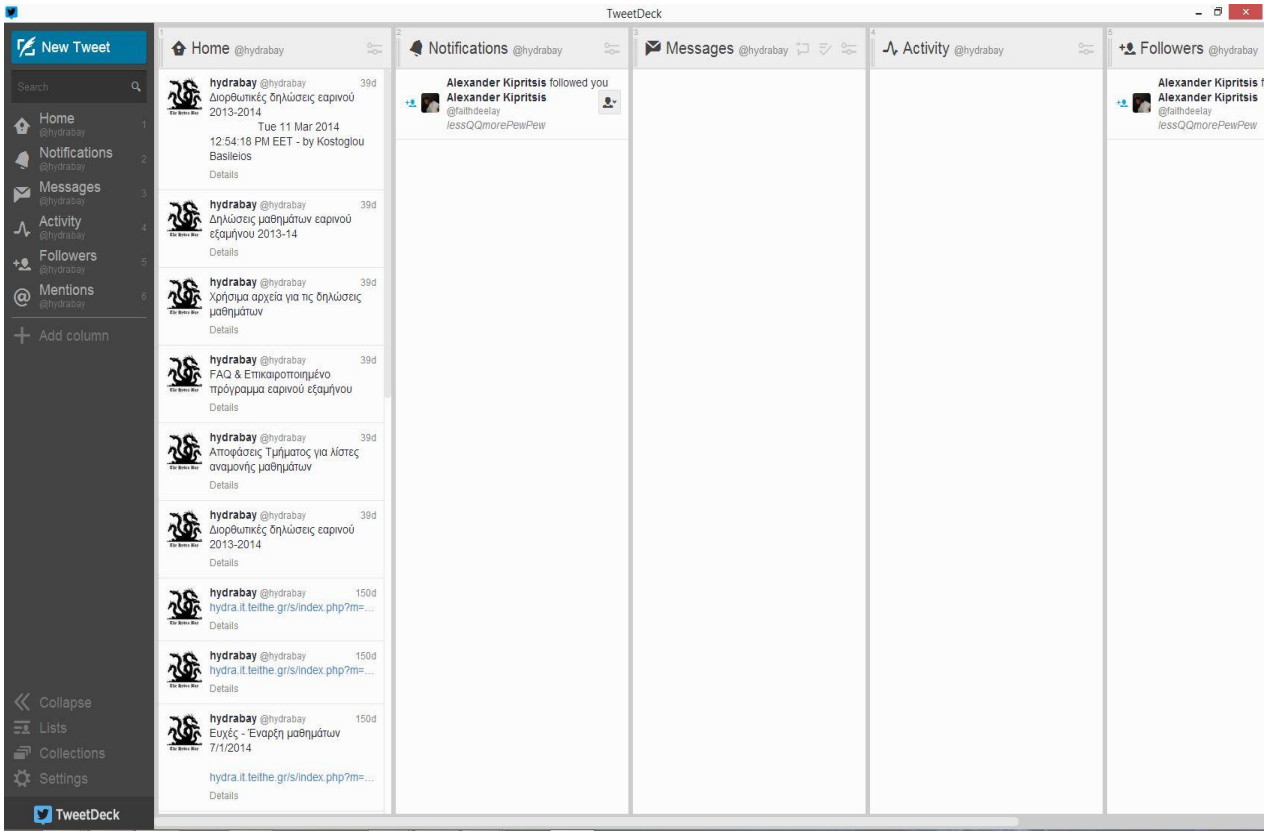



**Εικόνα 3.8 Προτάσεις αναρτήσεων του Buffer βάσει δημοφιλών trend.**

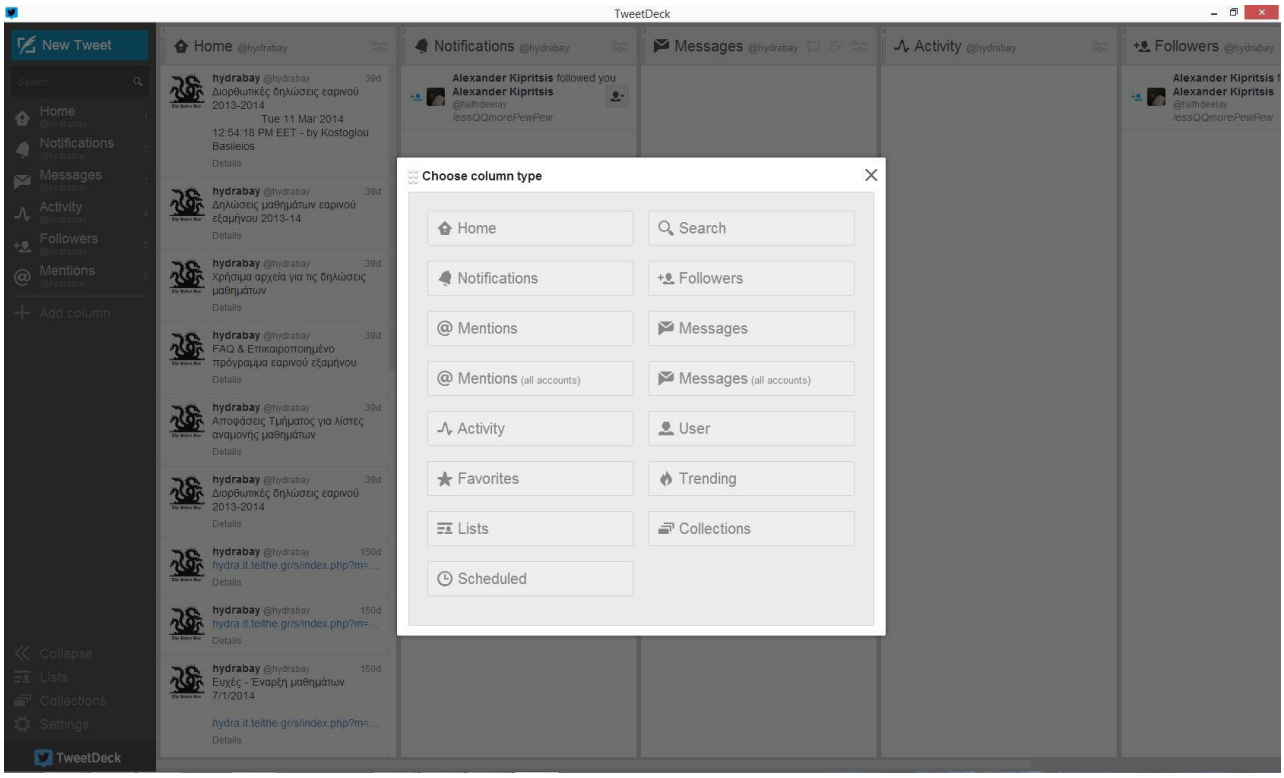
## 3.5 TweetDeck

Το TweetDeck[29] προσφέρεται δωρεάν, ως ένα Web Interface αλλά και ως desktop εφαρμογή. Η εφαρμογή δίνει μεγαλύτερη βάση στην παρακολούθηση ενός λογαριασμού Twitter, ενώ παρέχει εργαλεία στατιστικών μετρήσεων. Μέσω της εφαρμογής, ο χρήστης έχει στην οθόνη του όλες τις βασικές λειτουργίες του Twitter, διαμορφωμένες με έναν πιο φιλικό τρόπο προς αυτόν. Οι λειτουργίες αυτές αποκτούν αυξημένη αξία με την προσθήκη της χρονοπρογραμματισμένης ανάρτησης και του Link Shortening.

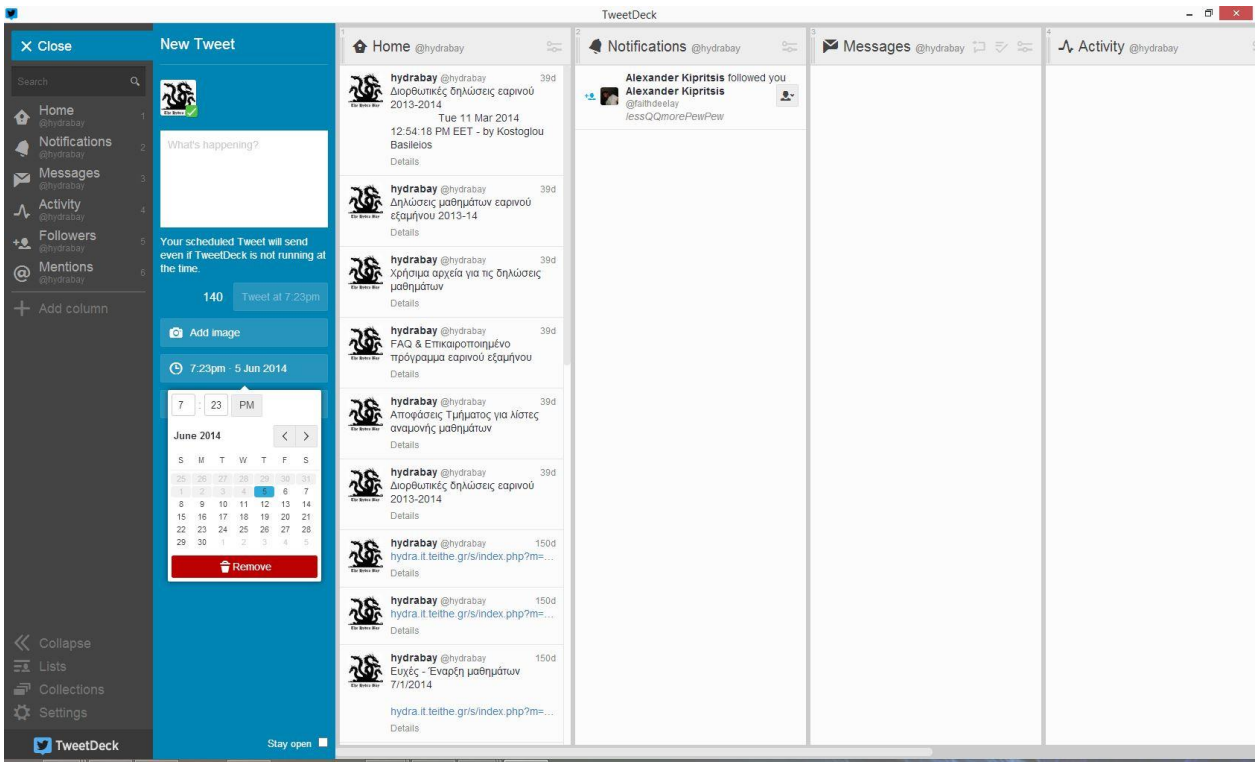
Η εφαρμογή δεν είναι επεκτάσιμη και είναι διαθέσιμη σε Windows, Macintosh καθώς και σε Linux. Η παραμετροποίηση που παρέχει δεν αφορά τόσο τη δημιουργία νέων αναρτήσεων, όσο το κομμάτι της παρακολούθησης του λογαριασμού στο Twitter, οπότε θεωρούμε πως διαθέτει ένα μέτριο βαθμό παραμετροποίησης.




**Εικόνα 3.9 Κεντρική οθόνη διαχείρισης του TweetDeck**




**Εικόνα 3.10 Επιλογή πεδίων παρακολούθησης στο TweetDeck**



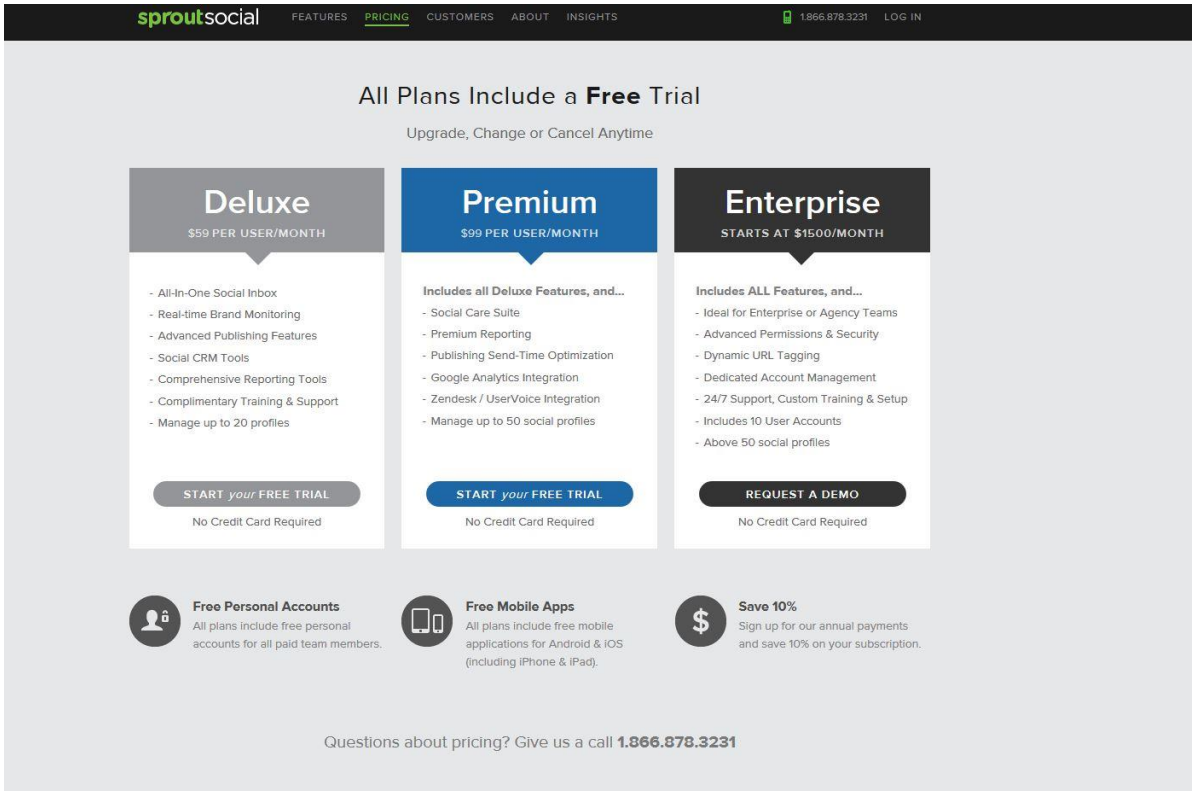

**Εικόνα 3.11 Δημοσίευση μιας νέας ανάρτησης**

## 3.6 Sprout Social

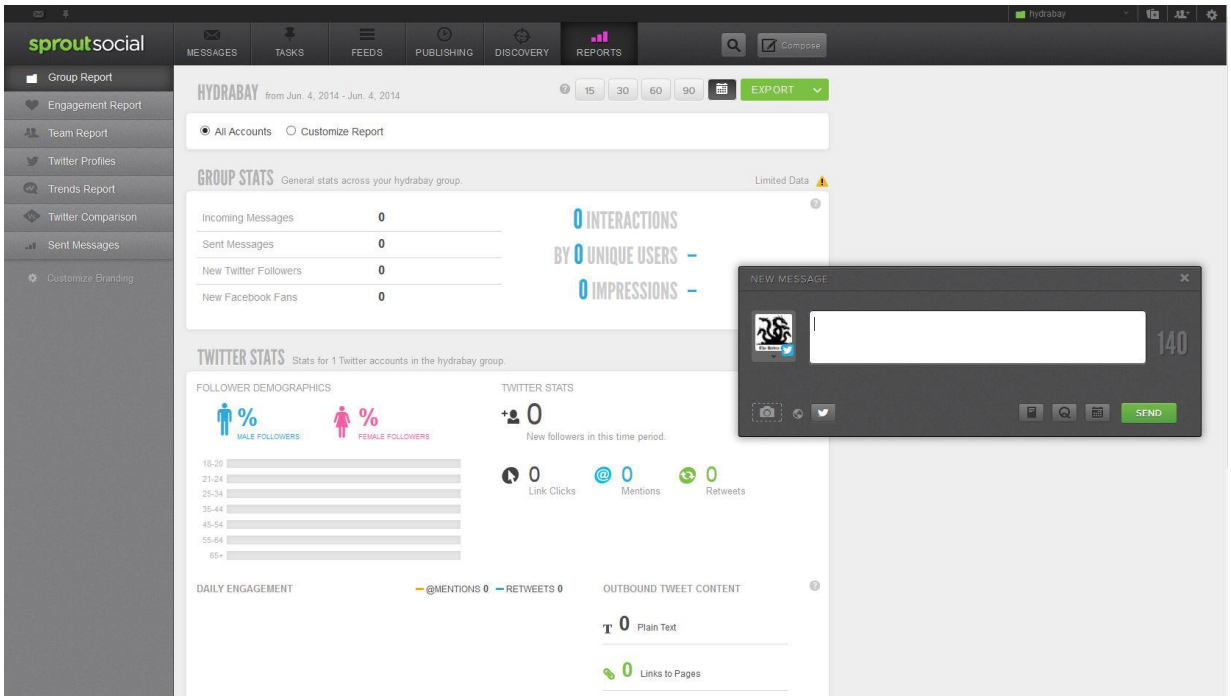
Το Sprout Social[30] είναι ένα εργαλείο το οποίο απευθύνεται σε επιχειρήσεις, γεγονός που γίνεται εμφανές από τις άδειες χρήσης του. Υπάρχουν τριών ειδών άδειες χρήσης, που μπορεί κανείς να αποκτήσει επί πληρωμή, οι οποίες φαίνονται στην παρακάτω εικόνα. Ωστόσο καθεμιά από αυτές προσφέρει δωρεάν δοκιμή διάρκειας 30 ημερών.

Μέσω του Web Interface προσφέρονται λεπτομερέστατα στατιστικά στοιχεία που αφορούν τα σημαντικά θέματα στο Twitter και τα διάφορα trends. Μεταξύ άλλων, παρέχονται οι υπηρεσίες της σύνδεσης με RSS Feed, ώστε να γίνονται αυτόματες αναρτήσεις, χρονοπρογραμματισμός ανάρτησης στο Twitter και σύνδεση με άλλα δίκτυα κοινωνικής δικτύωσης για να διευκολύνεται η διαδικασία της ανάρτησης σε πολλούς λογαριασμούς.

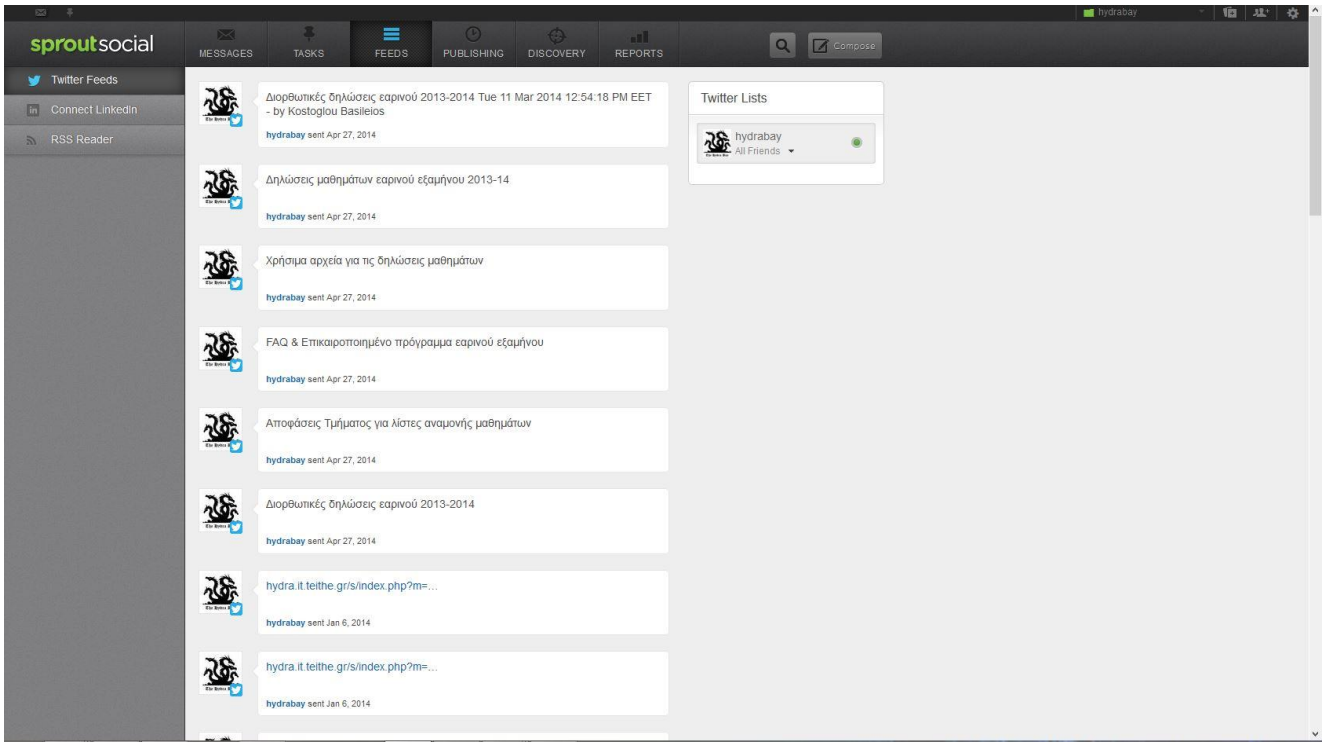
Πέραν των βασικών ρυθμίσεων, η εφαρμογή δεν δέχεται επιπλέον παραμετροποίηση ή επέκταση.




**Εικόνα 3.12 Επιλογή άδειας χρήσης στο Sprout Social.**




**Εικόνα 3.13 Υπηρεσίας στατιστικών και δημιουργία μιας νέας ανάρτησης.**



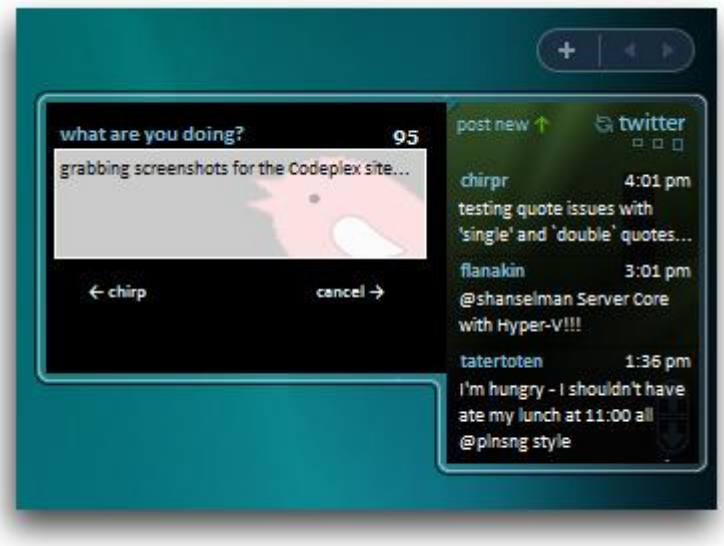

**Εικόνα 3.14 Κεντρική οθόνη του Sprout Social.**

## 3.7 Chirpr

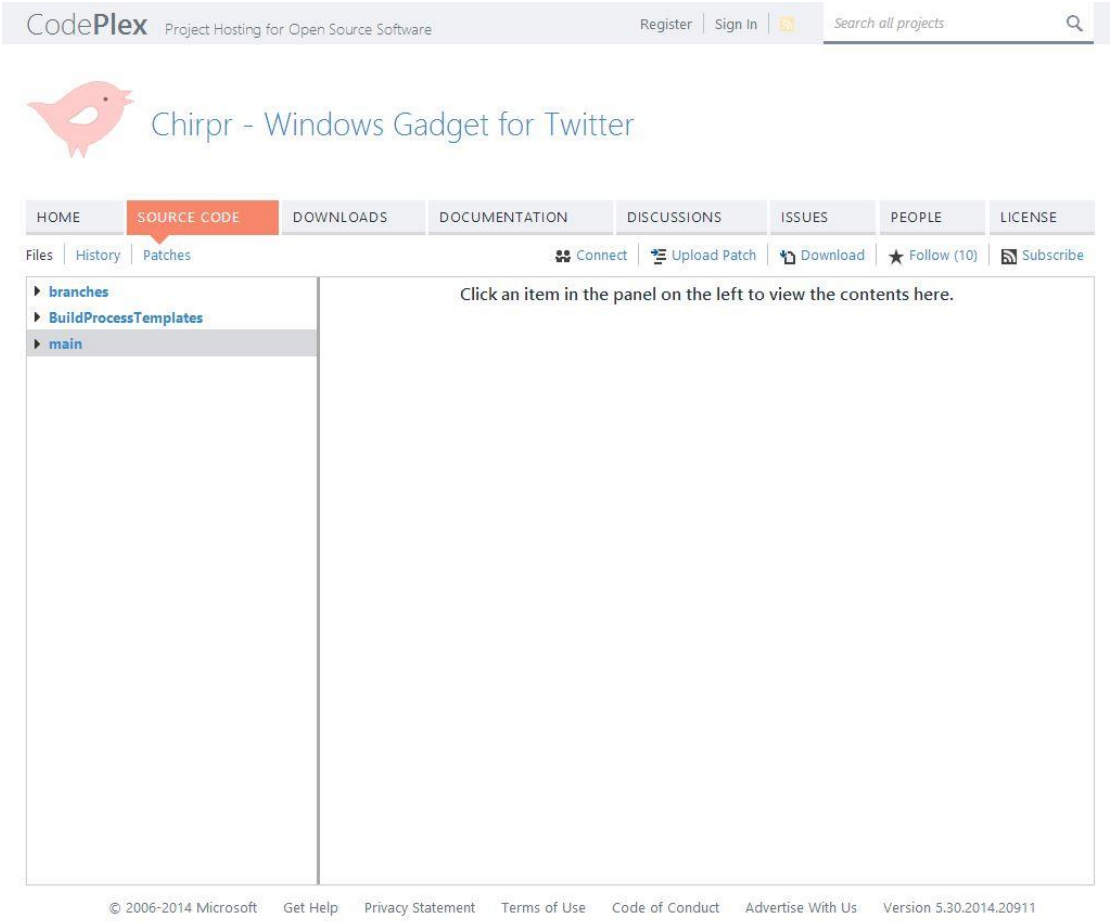
Το Chirpr[43] αποτελεί ένα Windows Widget, σχεδιασμένο ώστε να φέρνει τις τελευταίες ενημερώσεις από το Twitter, κατευθείαν στην επιφάνεια εργασίας του χρήστη.

Η εφαρμογή παρέχει αύτοματη ενημέρωση, με ειδοποίηση ήχου, και προβολή των νέων αναρτήσεων ανά 60 δευτερόλεπτα και προσφέρει στον χρήστη τη δυνατότητα να κάνει αναρτήσεις, ή να απαντάει σε υπάρχουσες, από την επιφάνεια εργασίας του.

Η υπηρεσία παρέχεται δωρεάν και ο κώδικας είναι διαθέσιμος στην επίσημη σελίδα. Συνεπώς η εφαρμογή είναι επεκτάσιμη, γεγονός που δεν ισχύει σε καμία από τις προηγούμενες υφιστάμενες λύσεις.




**Εικόνα 3.15 Σύνθεση νέου μηνύματος μέσω του Widget στην επιφάνεια εργασίας**




**Εικόνα 3.16 Η επίσημη σελίδα του Chirpr παρέχει τον πηγαίο κώδικα της εφαρμογής**

## 3.8 Σύγκριση υπηρεσιών

Ο παρακάτω πίνακας παρουσιάζει συγκριτικά τις δυνατότητες και τα μειονεκτήματα ή πλεονεκτήματα της κάθε μιας. Δυστυχώς δε μπορέσαμε να εντοπίσουμε καμία εφαρμογή ανοικτού κώδικα, ούτε κάποια εφαρμογή που να παρέχει κάποιο άλλο τρόπο επέκτασης. Εξαίρεση αποτελεί το Chirpr, το οποίο όμως ως Windows widget μπορεί να χρησιμοποιηθεί μόνο σε μερικές εκδόσεις Windows. Ωστόσο, είναι θετικό ότι υπάρχουν αρκετές εφαρμογές που παρέχονται δωρεάν.

**Πίνακας 3-1 Συγκριτικός πίνακας αξιολόγησης**

| | Παραμετρ. | Περιβάλλον | Επεκτάσιμη | ΕΛ/ΛΑΚ | Δωρεάν |
|---|---|---|---|---|---|
| TwitterMail | ✓ | ✓ | x | x | ✓ |
| Twuffer TL | x | ✓ | x | x | ✓ |
| TwitterFeed | ~ | ✓ | x | x | ✓ |
| Buffer | ✓ | ✓ | x | x | ~ |
| TweetDeck | ~ | ✓ | x | x | ✓ |
| SproutSocial | ~ | ✓ | x | x | x |
| Chirpr | ~ | x | ✓ | ✓ | ✓ |

Φαίνεται να είναι σημαντικός παράγοντας στην υλοποίηση των περισσότερων εφαρμογών ο χρόνος στον οποίο θα γίνονται αναρτήσεις. Το TweetLater είναι μια υπηρεσία που έχει δημιουργηθεί με μοναδικό σκοπό αυτόν, τη μελλοντική ανάρτηση μηνυμάτων. Ωστόσο, εφόσον αποτελεί αυτοσκοπό, δε θεωρούμε ότι αυτό είναι στοιχείο παραμετροποίησης της υπηρεσίας, ενώ στις υπόλοιπες εφαρμογές, που δίνουν τέτοια δυνατότητα, το δεχόμαστε σαν πλεονέκτημά τους.

Επίσης, αρκετές από τις εφαρμογές που εντοπίσαμε, δίνουν τη δυνατότητα να στέλνουν ενημερώσεις σε περισσότερα κοινωνικά δίκτυα, πράγμα που φυσικά έχει νόημα, εφόσον γλυτώνει κανείς έτσι εύκολα πόρους και μπορεί να έχει συγκεντρωμένες όλες τις ρυθμίσεις σε μία τοποθεσία.

## 3.9 Επικοινωνία με το Twitter

Όλες οι παραπάνω υφιστάμενες υπηρεσίες χρησιμοποιούν κάποιο Twitter API για να δημιουργήσουν νέες αναρτήσεις στο Twitter. Δυστυχώς δεν είναι δυνατό να καταλήξουμε στο συμπέρασμα για το ποια διεπαφή χρησιμοποιείται στην κάθε περίπτωση, εφόσον δεν έχουμε πρόσβαση στον κώδικα και τα αποτελέσματα (οι αναρτήσεις στο Twitter δηλαδή) δεν αποκαλύπτουν κάτι για να βγάλουμε ασφαλές συμπέρασμα.

Τα δύο API που παρέχει το κοινωνικό δίκτυο πάντως, είναι το REST API, το οποίο είναι χρήσιμο για διεσπαρμένα μοτίβα επικοινωνίας και με τη χρήση του οποίου κάθε σύνδεση είναι και ένα request, και το Streaming API, το οποίο μπορεί να χρησιμοποιείται για τακτική επικοινωνία και διατηρεί μια ανοικτή σύνδεση για να πραγματοποιηθούν πολλαπλά requests. Οι δύο αυτές διεπαφές θα αναλυθούν εκτενώς στο πέμπτο κεφάλαιο, ωστόσο με την έρευνα, που έχει γίνει μέχρι εδώ, μπορούμε να ξεκινήσουμε το σχεδιασμό της υπηρεσίας. Το επόμενο κεφάλαιο θα δημιουργήσει μια θεωρητική θεμελίωση, πάνω στην οποία θα στηριχτούμε για να δημιουργήσουμε την εφαρμογή.

# 4 Ανάλυση απαιτήσεων και περιγραφή σχεδιασμού της υπηρεσίας

Στο κεφάλαιο αυτό κάνουμε περιγραφή του σχεδιασμού της υπηρεσίας, σύμφωνα με το μοντέλο που τελικά επιλέχθηκε, δηλαδή το pull. Περιγράφουμε τα εργαλεία που χρησιμοποιήσαμε, κάνουμε ανάλυση απαιτήσεων και παρουσιάζουμε τη ροή εκτέλεσης της εφαρμογής.

## 4.1 Εργαλεία που χρησιμοποιήθηκαν

Τα εργαλεία που χρησιμοποιήθηκαν τελικά για την υλοποίηση της εφαρμογής, είναι τα εξής:

- Το ολοκληρωμένο περιβάλλον ανάπτυξης Netbeans για τον προγραμματισμό της εφαρμογής.
- Μια διανομή *AMP (LAMP, XAMPP…), η οποία είναι ένα πακέτο προγραμμάτων ελεύθερου λογισμικού, είναι ανεξάρτητη πλατφόρμας και περιέχει:
    - Τον εξυπηρέτη ιστοσελίδων http Apache
    - Την βάση δεδομένων MySQL
    - Ένα διερμηνέα για κώδικα γραμμένο σε γλώσσες προγραμματισμού PHP ή Perl.
- Τη Διεπαφή Προγραμματισμού Εφαρμογών του Twitter.
- Το κατανεμημένο σύστημα διαχείρισης εκδόσεων λογισμικού (Distributed Version Control System, DVCS) Git, που διευκολύνει τη σύγκριση εκδόσεων προγραμμάτων που γράφονται ατομικά ή ομαδικά.

## 4.2 Ανάλυση απαιτήσεων

Η εφαρμογή εκτελείται σε έναν αποκλειστικό διακομιστή, ο οποίος πρέπει να έχει πρόσβαση στο διαδίκτυο και στον οποίο εκτελείται Apache και PHP.

### *4.2.1 Λειτουργικές απαιτήσεις*

Από πλευράς επεξεργαστικής ισχύος, μεγέθους μνήμης και δίσκου οι απαιτήσεις είναι εξαιρετικά χαμηλές ουσιαστικά, γιατί οι περισσότεροι διακομιστές τρέχουν ούτως ή άλλως Apache και PHP, που χρησιμοποιούνται και για άλλες υπηρεσίες. Η ίδια η εφαρμογή χρησιμοποιεί ελάχιστους πόρους (σύμφωνα με τα σημερινά δεδομένα). Αξιοσημείωτο είναι ίσως, ότι είναι δυνατό να ρυθμίσουμε το ρυθμό επανεκτέλεσης της εφαρμογής. Έτσι, ουσιαστικά μπορούμε να ελέγξουμε πόσους πόρους θα χρησιμοποιεί κατά

μέσο όρο τη μέρα, αν και με τις σημερινές δυνατότητες των υπολογιστών, το κέρδος σε πόρους είναι μηδαμινό, όσο μειώνουμε τον ρυθμό επανεκτέλεσης της συγκεκριμένης εφαρμογής.

Ως λειτουργικό σύστημα για την εκτέλεση μπορεί να επιλεγεί οποιοδήποτε από τα συνηθισμένα, εφόσον όλα τα εργαλεία που χρησιμοποιήθηκαν υπάρχουν σε εκδόσεις για όλα τα λειτουργικά συστήματα.

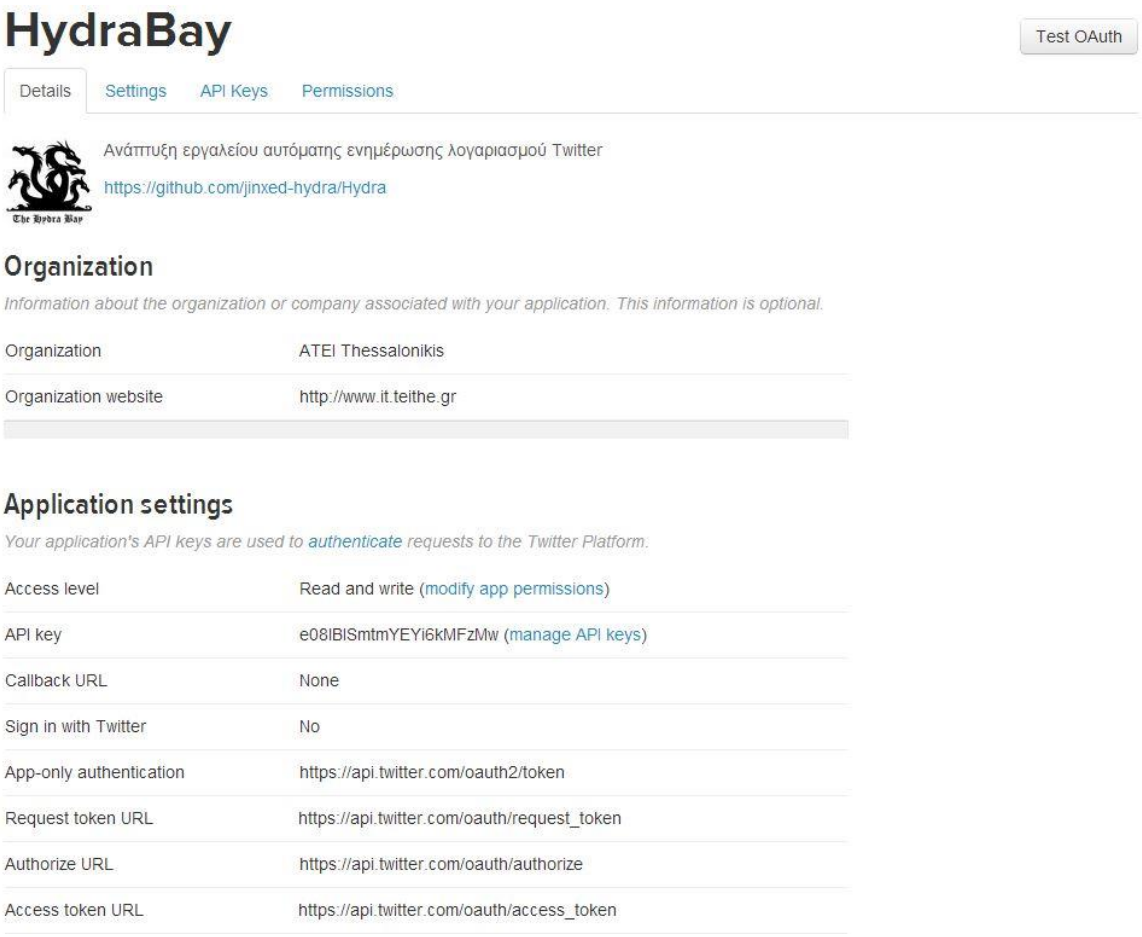
### *4.2.2 Τεχνολογικές απαιτήσεις*

Για την εκτέλεση της εφαρμογής, απαιτείται δυνατότητα σύνδεσης στο Twitter και στην πηγή των δεδομένων (hydra). Για την τελευταία αρκεί η πρόσβαση της εφαρμογής στο διαδίκτυο, δεδομένου ότι η hydra εξυπηρετεί. Τα δεδομένα είναι διαθέσιμα δημόσια, οπότε δε χρειάζεται κάποια αυθεντικοποίηση.

Ωστόσο, στην περίπτωση της σύνδεσης στο Twitter, πρέπει να γίνει αυθεντικοποίηση. Για αυτόν το λόγο, παρέχεται από το Twitter, όταν ζητηθεί, ένα ζευγάρι αναγνωριστικών. Επίσης, το Twitter παρέχει δύο επιπλέον αναγνωριστικά, τα οποία ορίζουν το λογαριασμό, στου οποίου τη σελίδα θα γίνουν οι νέες αναρτήσεις. Κατά τη δημιουργία της εφαρμογής, πρέπει αυτή να δηλωθεί στο Twitter για να αποκτήσουμε αυτά τα δύο ζευγάρια αναγνωριστικών.

Δηλώνουμε επίσης μία δεξαμενή πηγαίου κώδικα, όπου μπορούμε να προσθέσουμε και την τεκμηρίωση του κώδικά μας, ώστε να είναι δυνατή η επαναχρησιμοποίηση και η επέκταση της εφαρμογής ακόμη και από τρίτους.

Η παρακάτω εικόνα περιέχει κάποιες βασικές ρυθμίσεις της εφαρμογής, όπως κυρίως το επίπεδο πρόσβασης της εφαρμογής στο Twitter.




**Εικόνα 4.1 Ρυθμίσεις ασφαλείας εφαρμογής**

Φυσικά για να είναι δυνατό να κάνουμε νέες αναρτήσεις, απαιτείται δυνατότητα εγγραφής από το επίπεδο πρόσβασης.

Η συγκεκριμένη εφαρμογή, όπως υλοποιήθηκε, απαιτεί η πηγή των δεδομένων να είναι μια προσβάσιμη ιστοσελίδα του διαδικτύου σε καλά ορισμένη HTML.

## 4.3 Περιγραφή ροής προγράμματος

Η εφαρμογή εκτελεί συνοπτικά τα παρακάτω βήματα με τη δεδομένη σειρά:

1. Ανάγνωση constants (πηγή, αναγνωριστικά Twitter, κτλ)
2. Εκκίνηση session και ανάγνωση δεδομένων τελευταίας εκτέλεσης
3. Αρχικοποίηση αντικειμένου επικοινωνίας με τη διεπαφή
4. Ανάκτηση περιεχομένων Ύδρας υπό μορφή HTML
5. Αποθήκευση υπό μορφή XML σε αρχείο
6. Φόρτωση των δεδομένων σε SimpleXMLElement
7. Διαχωρισμός των ανακοινώσεων από στοιχεία που δεν ενδιαφέρουν
8. Μετατροπή της φιλτραρισμένης πληροφορίας σε associative PHP array για εύκολη και γρήγορη πρόσβαση
9. Εκτίμηση του τίτλου της ανακοίνωσης και του σχετικού υπερσυνδέσμου με ταυτόχρονη μείωση μήκους του τίτλου αν χρειάζεται, ώστε να ικανοποιεί τον περιορισμό των 140 χαρακτήρων
10. Σύνδεση στο Twitter με τα δεδομένα ζεύγη αναγνωριστικών και αίτημα για δημιουργία της νέας ανάρτησης
11. Σημείωση στο αρχείο καταγραφής της επιτυχίας ή της αποτυχίας εκτέλεσης του αιτήματος
12. Αποθήκευση μεταβλητών στο session για μετέπειτα χρήση, ενημέρωση αρχείου καταγραφής με στατιστικές μετρήσεις της εκτέλεσης και επιστροφή ροής στο λειτουργικό σύστημα

Τα βήματα 9-11 επαναλαμβάνονται μέχρι να συναντήσουμε κάποια ανακοίνωση που ήδη έχουμε κάνει στο παρελθόν, οπότε διακόπτουμε τη ροή προκαταβολικά και την επιστρέφουμε στο λειτουργικό σύστημα.

### *4.3.1 Ανάκτηση περιεχομένων*

Φορτώνεται το περιεχόμενο της ιστοσελίδας σε μία μεταβλητή τύπου DOM Document.

Η μέθοδος που χρησιμοποιείται για την ανάκτηση των δεδομένων είναι η file_get_contents(), ενώ για την αποθήκευση στη μεταβλητή χρησιμοποιείται η loadHTML().

### 4.3.2 Δημιουργία αρχείου XML

Δημιουργούμε ένα νέο αρχείο XML έκδοσης 1.0 για να αποθηκεύσουμε την πληροφορία[4] υπό μορφή XML. Αυτό θα μας βοηθήσει στο να διαχωρίσουμε τα περιττά δεδομένα από αυτά που πραγματικά χρειαζόμαστε και να μετατρέψουμε την πληροφορία σε μια βολική μορφή.

### 4.3.3 Φιλτράρισμα και μετατροπή μορφής πληροφορίας

Χρησιμοποιώντας XPath στο SimpleXMLElement, ξεχωρίζουμε τις πληροφορίες που εν δυνάμει μας χρειάζονται. Για παράδειγμα δεν μας χρειάζεται το <header> της ιστοσελίδας, ούτε το κομμάτι του <body> που έχει να κάνει με τη σύνδεση και αποσύνδεση του χρήστη. Μετά το φιλτράρισμα, μπορούμε πια να μετατρέψουμε το SimpleXMLElement σε associative PHP array, το οποίο μας διευκολύνει στην πρόσβαση της πληροφορίας από εκεί και πέρα, αλλά μειώνει και το συνολικό χρόνο εκτέλεσης του script δραματικά. Ο πίνακας αυτός περιέχει μόνο ανακοινώσεις.

### 4.3.4 Έλεγχος ανακοινώσεων και ανάρτηση tweets

Τα βήματα 9 ως 11 αποτελούν το σώμα ενός βρόχου, ο οποίος αποτελεί τον κορμό του προγράμματος, με την έννοια ότι αυτός είναι υπεύθυνος για την ανάγνωση όλων των μηνυμάτων, την εκμαίευση κειμένου προς μεταβολή και την παραγωγή και αποστολή tweets. Όλες οι λεπτομέρειες αυτής της διαδικασίας παρουσιάζονται αναλυτικά στο κεφάλαιο 6.

### 4.3.5 Σύνδεση με το Twitter

Σε αυτό το βήμα δίνονται οι παράμετροι που χρειάζονται, ώστε να γίνει δυνατή η σύνδεση με το Twitter. Είναι απαραίτητη η ύπαρξη τους, με όποιο τρόπο και να υλοποιηθεί η εφαρμογή.

Η παράμετρος ‘access_token’, που εμπεριέχει ένα ζεύγος παραμέτρων (‘token’ και ‘secret’), χρειάζεται για να δώσουμε στο Twitter να καταλάβει σε ποιο λογαριασμό θα γίνονται τα tweets, ενώ η παράμετρος ‘oauth_options’, που επίσης εμπεριέχει ζεύγος παραμέτρων (‘consumerKey’ και ‘consumerSecret’), είναι απαραίτητη για τη σύνδεση στο Twitter ως ο συγκεκριμένος χρήστης (αυθεντικοποίηση). Η αυθεντικοποίηση δεν ήταν απαραίτητη στην έκδοση 1.0 του Twitter API για όλες τις ενέργειες ενός χρήστη, ωστόσο στην έκδοση 1.1 είναι απαραίτητη για λόγους ασφάλειας και στατιστικής[2].

### 4.3.6 Αρχείο καταγραφής

Κατά τη διάρκεια της ροής του προγράμματος παράγονται αναφορές (όταν αυτό είναι σκόπιμο). Αυτές παρουσιάζονται στο χρήστη κατά την εκτέλεση και αποθηκεύονται στο αρχείο καταγραφής που έχουμε ορίσει.

[4] *Η έκδοση 1.1 δεν υποστηρίζεται από την SimpleXmlElement της PHP, την οποία χρησιμοποιούμε για να διαβάσουμε την πληροφορία[4]*

## 4.4 Βασικά συστατικά

Τρεις κλάσεις[5] είναι απαραίτητες για την επιτυχή εκτέλεση της εφαρμογής, αυτή που περιέχει τον κύριο κορμό της εφαρμογής και η βιβλιοθήκη επικοινωνίας με το Twitter:

- Η βιβλιοθήκη, η οποία φροντίζει για την αυθεντικοποίηση, επικοινωνία και ανταλλαγή δεδομένων με το Twitter.
- Η βοηθητική κλάση, που περιέχει στατικές μεθόδους που βοηθούν στην μετατροπή της πληροφορίας σε διαφορετικές μορφές και στην επεξεργασία της.
- Η κλάση εργάτης είναι η κύρια κλάση και περιέχει τον κορμό της εφαρμογής. Αντλεί πληροφορία από την πηγή, την επεξεργάζεται με τη βοήθεια της βοηθητικής κλάσης και στη συνέχεια χρησιμοποιεί τη βιβλιοθήκη για να παράγει νέες αναρτήσεις.

Κάνοντας χρήση των κλάσεων αυτών, μπορούμε να εκτελούμε πολλά instances της εφαρμογής ταυτόχρονα (π.χ. για παραγωγή αναρτήσεων από διαφορετικές πηγές) με ελάχιστες αλλαγές στον κώδικα. Πιθανότερο είναι να μη χρειάζεται καμία αλλαγή, αλλά μόνο να θέσουμε σωστές τιμές στις σταθερές μεταβλητές που ορίζονται στην αρχή της κύριας κλάσης. Μπορούμε επίσης με την προσθήκη επιπλέον βιβλιοθηκών και κάνοντας τις αντίστοιχες κλήσεις στην κύρια κλάση να φροντίσουμε να παράγει η εφαρμογή αναρτήσεις σε περισσότερα κοινωνικά δίκτυα.

Θεωρείται σκόπιμο να αναφέρουμε ως βασικό, το γεγονός ότι η εφαρμογή εκτελείται ανά τακτά χρονικά διαστήματα από το λειτουργικό σύστημα. Η αυτοματοποίηση αυτή καθιστά την εφαρμογή απολύτως αυτόνομη. Δεν χρειάζεται καμία ενέργεια από το χρήστη, ώστε να μεταδίδονται οι νέες ανακοινώσεις στο Twitter. Το επόμενο κεφάλαιο περιγράφει τις τεχνολογίες που καθιστούν δυνατή την υλοποίηση μιας τέτοιας εφαρμογής.

[5] *Βιβλιοθήκη: TwitterAPIExchange, Βοηθητική: Helper, Εργάτης: Hydrabay*

# 5 Συνοπτική περιγραφή του τεχνολογικού περιβάλλοντος που χρησιμοποιήθηκε

Για την υλοποίηση και λειτουργία της εφαρμογής γίνεται χρήση διαφόρων τεχνολογιών ή εργαλείων, μερικές από τις οποίες έχουν επιλεγεί (και θα μπορούσαν να αντικατασταθούν με άλλες), ενώ άλλες επιβάλλονται από τις υπηρεσίες με τις οποίες επικοινωνεί η εφαρμογή (π.χ. από το API του Twitter).

Αρχικά θα αναλυθούν οι τεχνολογίες και τα εργαλεία που έχουμε επιλέξει (Apache, PHP, XML, XPath) και θα εξηγήσουμε τους λόγους για τους οποίους έγιναν οι συγκεκριμένες επιλογές.

Στη συνέχεια ακολουθεί μια σύντομη περιγραφή των χαρακτηριστικών των τεχνολογιών που επιβάλλεται να χρησιμοποιηθούν από μια τέτοιου είδους εφαρμογή, άσχετα με τον τρόπο υλοποίησής της και ειδικά οι διεπαφές προγραμματισμού που παρέχει το Twitter (Rest & Streaming APIs).

## 5.1 PHP & Apache


**Εικόνα 5.1 Λογότυπο της PHP**

Η PHP[7][8][9] είναι μια γλώσσα προγραμματισμού για τη δημιουργία σελίδων web με δυναμικό περιεχόμενο. Μια σελίδα PHP περνά από επεξεργασία από ένα συμβατό διακομιστή του Παγκόσμιου Ιστού (π.χ. Apache), ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, που θα σταλεί στο πρόγραμμα περιήγησης των επισκεπτών σε μορφή κώδικα HTML.

### *5.1.1 Επεκτάσεις αρχείων και διακομιστές*

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 ευρέως σε συστήματα με 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 υποστηρίζονται η εκτέλεση σε πολυάσχολους ιστοχώρους, FastCGI Process Manager (FPM).

### *5.1.2 Περιβάλλοντα *AMP*

Ο συνδυασμός Linux/Apache/PHP/MySQL[10], που είναι η πιο δημοφιλής πλατφόρμα εκτέλεσης ιστοσελίδων είναι γνωστός και με το ακρωνύμιο LAMP. Παρόμοια, ο συνδυασμός */Apache/PHP/MySQL ονομάζεται *AMP[12], όπου το πρώτο αρχικό αντιστοιχεί στην πλατφόρμα, στην οποία εγκαθίστανται ο Apache, η PHP και η MySQL (π.χ. Windows, Mac OS X).

Ο LAMP[11] συνήθως εγκαθίσταται και ρυθμίζεται στο Linux με τη βοήθεια του διαχειριστή πακέτων της εκάστοτε διανομής. Στην περίπτωση άλλων λειτουργικών συστημάτων, επειδή το κατέβασμα και η ρύθμιση των ξεχωριστών προγραμμάτων μπορεί να είναι πολύπλοκη, υπάρχουν έτοιμα πακέτα προς εγκατάσταση, όπως το XAMPP και το WAMP για τα Windows και το MAMP για το Mac OS X.

### *5.1.3 Συμπέρασμα:*

*Η PHP είναι μια γλώσσα προγραμματισμού πολύ διαδεδομένη, η οποία υποστηρίζεται από όλα τα ευρέως διαδεδομένα λειτουργικά συστήματα που χρησιμοποιούνται σαν εξυπηρέτες και, κυρίως, από το Linux. Τα *nix συστήματα είναι τα πιο ασφαλή και σταθερά και ως εκ τούτου επιλέγουμε να τα χρησιμοποιήσουμε ως εξυπηρέτες, συμπεριλαμβάνοντας τα πακέτα που υποστηρίζουν δωρεάν. Ο Apache είναι επίσης ένας πολύ διαδεδομένος διακομιστής σελίδων διαδικτύου, που παρέχεται ως πακέτο από κάθε γνωστή διανομή Linux.*

*Σημειωτέον: όλα τα προγράμματα - συστατικά που αποφασίσαμε να χρησιμοποιήσουμε είναι ελεύθερα και δωρεάν και χρησιμοποιούνται ευρέως από διάφορα υπολογιστικά συστήματα.*

## 5.2 XML


**Εικόνα 5.2 Λογότυπο της XML**

### *5.2.1 Γενικά*

Η XML[13][14] σχεδιάστηκε να ικανοποιήσει πολλές ανάγκες δίνοντας στα έγγραφα ένα μεγαλύτερο επίπεδο προσαρμοστικότητας στο στυλ και τη δομή από αυτό που υπήρχε παλαιότερα στην HTML. Η XML προσφέρει στους σχεδιαστές της HTML τη δυνατότητα να προσθέτουν περισσότερα στοιχεία στη γλώσσα.

Είναι κάτι περισσότερο από markup language είναι metalanguage, δηλαδή μια γλώσσα που χρησιμοποιείται για να καθορίσει νέες markup γλώσσες. Η XML συμπληρώνει και δεν αντικαθιστά την HTML. Ενώ η HTML χρησιμοποιείται στη διατύπωση και την εμφάνιση των δεδομένων, η XML αναπαριστά τη συναφή έννοια των δεδομένων.

Η XML περιγράφει μια κατηγορία πληροφοριών (data objects) που καλούνται XML έγγραφα (documents). Τα έγγραφα αυτά αποτελούνται από μονάδες αποθήκευσης που καλούνται entities (οντότητες), οι οποίες περιέχουν πληροφορίες αναλυμένες ή μη. Οι αναλυμένες πληροφορίες αποτελούνται από χαρακτήρες (characters) οι οποίοι συνθέτουν character data και άλλοι οι οποίοι συνθέτουν markup. Η μορφή markup κωδικοποιεί την περιγραφή της τελικής αποθήκευσης του εγγράφου καθώς και τη λογική δομή. Τα XML έγγραφα δεν είναι πολύπλοκα αλλά απλά και πολύ αποτελεσματικά.

### *5.2.2 Στόχοι*

1. Κάποιοι ενδιαφέροντες προσχεδιασμένοι στόχοι της XML είναι:
2. Η XML πρέπει να είναι εύχρηστη στο Internet.
3. Η XML πρέπει να υποστηρίζει μεγάλη ποικιλία από εφαρμογές.
4. Θα είναι εύκολο να γράφονται προγράμματα που επεξεργάζονται XML έγγραφα.
5. Τα XML έγγραφα θα πρέπει να είναι ευανάγνωστα.
6. Ο σχεδιασμός XML θα πρέπει να προετοιμάζεται γρήγορα.
7. Ο σχεδιασμός XML θα πρέπει να είναι τυπικός και περιεκτικός.
8. Τα XML έγγραφα θα πρέπει να δημιουργούνται εύκολα.

*Όλοι οι παραπάνω στόχοι της γλώσσας, καθιστούν τη χρήση της μια πολύ καλή πιθανή λύση στο πρόβλημά μας.*

### *5.2.3 Χρήση*

Υπάρχουν πολλοί λόγοι για τους οποίους μπορεί κανείς σήμερα να χρησιμοποιήσει την XML. Στην συγκεκριμένη παράγραφο όμως, δε θα σταθούμε στα πλεονεκτήματα που μπορεί να μας αποφέρει η χρήση της XML σε δεδομένες εφαρμογές, όπως είναι για παράδειγμα το εμπόριο στο διαδίκτυο, τα μαθηματικά, η βιολογία, η χημεία κ.α. – αλλά θα εστιάσουμε σε συγκεκριμένες ιδιότητες της XML οι οποίες είναι ιδιαίτερα χρήσιμες για όλες τις εφαρμογές και καθιστούν την XML ως ένα από τους πιο δημοφιλείς τρόπους για την αναπαράσταση και την περιγραφή δεδομένων.

- *Η XML είναι εύκολα αναγνώσιμη από ανθρώπους και μηχανές*: Οι περισσότερες μορφές αποθήκευσης δεδομένων ήταν είτε κατάλληλες για μετάφραση από προγράμματα λογισμικού (π.χ. dBase, GIF), είτε αναγνώσιμα από ανθρώπους (text ή CSV αρχεία). Η XML ορίζει ένα σύνολο από κανόνες που κάνουν την μετάφραση από υπολογιστή πολύ απλή. Έτσι ικανοποιούνται και οι δύο πλευρές, αφού τα XML έγγραφα διατηρούν ως βάση τους το κείμενο κι έτσι μπορεί εύκολα να τα χειριστεί ένας άνθρωπος.
- *Η XML είναι φιλική στα αντικείμενα (object-friendly):* Ενώ το σχεσιακό μοντέλο δεδομένων εμφανίζει μεγάλη επιτυχία για την επεξεργασία μεγάλων ποσοτήτων δεδομένων αποθηκευμένων σε πίνακες, ο χειρισμός άλλων τύπων δεδομένων - όπως είναι το υπερκείμενο, τα πολυμέσα, τα γραφικά, οι μαθηματικές ή χημικές φόρμουλες, η ιεραρχική πληροφορία κ.ά. – δεν είναι τόσο απλός. Η XML από την άλλη πλευρά είναι φιλική στα αντικείμενα, υπό την έννοια ότι είναι κατάλληλη για την περιγραφή αντικειμένων του πραγματικού κόσμου ή οποιουδήποτε αφαιρετικού προβλήματος μοντελοποιώντας τις ιδιότητες όπως ακριβώς είναι, αντί να χρειάζεται μια κανονικοποιημένη διάσπαση σε διάφορους πίνακες, με τους οποίους συνδέονται διάφορες σχέσεις. Αυτό κάνει τα XML έγγραφα περισσότερο κατανοητά κι έτσι μειώνεται ο χρόνος που απαιτείται για την σχεδίαση και υλοποίηση υπολογιστικών συστημάτων που βασίζονται στην XML.
- *Η XML είναι ένας ξεκάθαρος και καθορισμένος τρόπος να δομήσουμε, να περιγράψουμε και να ανταλλάξουμε δεδομένα.*

*Ένα XML έγγραφο (όπως και ένα HTML) μπορεί να αναπαρασταθεί σαν ένα δέντρο,* του οποίου οι κόμβοι περιέχουν τα δομικά συστατικά του εγγράφου, δηλαδή τα elements, το κείμενο, τα attributes, τα σχόλια και τις processing instruction. Η δενδρική αναπαράσταση ενός XML εγγράφου θυμίζει την δομή των αρχείων και των φακέλων στον σκληρό δίσκο ενός

υπολογιστή. Οι κόμβοι είναι πανομοιότυποι με τα αρχεία και τους φακέλους. Έτσι όπως οι φάκελοι μπορεί να περιέχουν άλλα αρχείακαι φακέλους, έτσι και ένας κόμβος μπορεί να περιέχει άλλους κόμβους.

Το δέντρο έχει έναν κόμβο αφετηρίας (root node) που περιέχει όλους τους υπόλοιπους κόμβους του δέντρου. Ο κόμβος της αφετηρίας περιέχει το element έγγραφο. Ο κόμβος που περιέχει το έγγραφο και οι κόμβοι που περιέχουν elements περιέχουν λίστες με τους κόμβους – παιδιά. Κάθε κόμβος μέσα στο δέντρο, εκτός από τον κόμβο αφετηρίας, έχει ένα κόμβο – γονέα και οι κόμβοι – γονείς μπορεί να έχουν ένα αριθμό από κόμβους - παιδιά ή απογόνους. Οι κόμβοι που περιέχουν κείμενο, attributes και σχόλια, δεν έχουν απογόνους.

### *5.2.4 Document Object Model (DOM)*

Στην προηγούμενη παράγραφο, είδαμε πώς τα XML έγγραφα μπορούν να αναπαρασταθούν ως δέντρα. Εκμεταλλευόμενοι την δενδρική αυτή δομή των εγγράφων, μπορούμε να χρησιμοποιήσουμε τους κατάλληλους αλγορίθμους, προκειμένου να μπορέσουμε να επεξεργαστούμε τα δεδομένα που αναπαρίστανται σε αυτά.

Για τον σκοπό αυτό χρειαζόμαστε ένα XML parser, ο οποίος ουσιαστικά διαβάζει το έγγραφο και δημιουργεί μία ιεραρχική δενδρική δομή του εγγράφου στην μνήμη. Η δενδρική αυτή δομή περιέχει όλα τα συστατικά ενός XML εγγράφου, τα οποία είδαμε αναλυτικά νωρίτερα. Προκειμένου να λάβουμε ή να αναζητήσουμε την πληροφορία που ο parser ανακτά από το XML έγγραφο, χρησιμοποιούμε τις μεθόδους που είναι ορισμένες στο API (application programming interface) του parser.

Μία σημαντική απόφαση που θα χρειαστεί επομένως να πάρει κανείς όταν βρίσκεται στην αρχή ενός XML project, είναι ποιο API θα χρησιμοποιήσει. Τα πιο σημαντικά APIs για την επεξεργασία των XML εγγράφων με PHP, είναι το SimpleXMLElement και το DOM (Document Object Model).

Το Document Object Model[15], είναι ένα πολύπλοκο API που μοντελοποιεί ένα XML έγγραφο ως δέντρο. Με το DOM μπορούμε να αναλύσουμε έγγραφα, καθώς και να δημιουργήσουμε καινούρια. Το κάθε XML έγγραφο αναπαριστάνεται σαν ένα document object. Στα έγγραφα μπορούμε να κάνουμε αναζητήσεις και ενημερώσεις καλώντας τις μεθόδους και τα objects που περιέχει το Document object. Parsers που είναι βασισμένοι στο DOM υπάρχουν διαθέσιμοι σε μία ποικιλία από προγραμματιστικές γλώσσες και συνήθως είναι διαθέσιμοι χωρίς χρέωση. Μία από αυτές τις γλώσσες είναι και η PHP, που υποστηρίζει και την XPath, η οποία αναλύεται λίγο περισσότερο στη συνέχεια.

### *5.2.5 XML Path Language (XPath)*

Έχοντας πλέον δει την δομή ενός XML εγγράφου αναλυτικά, εξετάζουμε τώρα πώς μπορούμε να χρησιμοποιήσουμε τη δομή αυτή, με τη βοήθεια της XPath, για να εντοπίσουμε συγκεκριμένα τμήματα του εγγράφου.

Για την υλοποίηση των εκφράσεων της XPath εκμεταλλευόμαστε τη δενδρική δομή των XML εγγράφων, καθώς στην XPath αντιμετωπίζουμε τα XML έγγραφα σαν να είναι δέντρα στα οποία το κάθε τμήμα του εγγράφου παριστάνεται ως κόμβος. Η XPath υποστηρίζει τους τύπους κόμβων που παρουσιάσαμε νωρίτερα στο DOM. Επομένως υποστηρίζει τους εξής τύπους : root, element, attribute, text, comment και processing instruction.

#### *5.2.5.1 Location Paths*

Ένα location path [16] (μονοπάτι τοποθεσίας) είναι μία έκφραση, με την οποία ορίζουμε τον τρόπο με τον οποίο θα κινηθούμε μέσα στο XML έγγραφο, από ένα κόμβο σε έναν άλλο. Το location path αποτελείται από τα location steps (βήματα τοποθεσίας), καθένα από τα οποία αποτελείται από ένα axis (άξονα), ένα node test (επιλογέα κόμβου) και ένα προαιρετικό predicate (φίλτρο).

Για να μπορέσουμε να εντοπίσουμε ένα συγκεκριμένο κόμβο μέσα στο έγγραφο, χρησιμοποιούμε πολλαπλά location steps, καθένα από τα οποία κάνει την ερευνά μας περισσότερο συγκεκριμένη.

#### *5.2.5.2 Axis*

Η αναζήτηση σε ένα XML έγγραφο ξεκινάει από ένα context node (κόμβο αναφοράς). Όλες οι διασχίσεις γίνονται με αφετηρία τον context node. Ένας άξονας καθορίζει ποιοι κόμβοι, που σχετίζονται με τον context node, θα συμπεριληφθούν στην έρευνά μας. Ο άξονας που επιλέγουμε για την έρευνά μας, μας δείχνει ακόμα την σειρά των κόμβων στο έγγραφο. Έτσι οι άξονες που επιλέγουν κόμβους που ακολουθούν τον context node σε document order καλούνται forward axes. Οι άξονες που επιλέγουν κόμβους που προηγούνται σε σχέση με τον context node σε document order καλούνται reverse axes.

Η Xpath[17] υποστηρίζει συνολικά 13 άξονες, τους οποίους παραθέτουμε στον πίνακα 5.1, καθώς και μια σύντομη περιγραφή του καθενός.

Ο κάθε άξονας έχει ένα κύριο τύπο κόμβου που ανταποκρίνεται στον τύπο κόμβου που ο συγκεκριμένος άξονας μπορεί να επιλέξει.

Έτσι για τον attribute άξονα ο κύριος τύπος κόμβου είναι attribute. Για τον namespace άξονα ο κύριος τύπος κόμβου είναι namespace. Όλοι οι υπόλοιποι άξονες έχουν τον element κύριο τύπο κόμβου.

**Πίνακας 5-1 XPath Axis**

| Axis Name | Ordering | Description |
|---|---|---|
| self | none | Ο ίδιος ο context node. |

| | | |
|---|---|---|
| parent | reverse | Ο γονιός του context node, αν υπάρχει. |
| child | forward | Τα παιδιά του context node, αν υπάρχουν. |
| ancestor | reverse | Οι πρόγονοι του context node, αν υπάρχουν. |
| ancestor-or-self | reverse | Οι πρόγονοι του context node, καθώς και ο ίδιος. |
| descendant | forward | Οι απόγονοι του context node. |
| descendant-or-self | forward | Οι απόγονοι του context node, καθώς και ο ίδιος. |
| following | forward | Οι κόμβοι στο XML έγγραφο που ακολουθούν τον context node, χωρίς να συμπεριλαμβάνουμε τους απογόνους. |
| following-sibling | forward | Οι γειτονικοί κόμβοι που ακολουθούν τον context node. |
| preceding | reverse | Οι κόμβοι στο XML έγγραφο που προηγούνται του context node, χωρίς να συμπεριλαμβάνουμε τους προγόνους. |
| preceding-sibling | reverse | Οι γειτονικοί κόμβοι που προηγούνται του context node. |
| attribute | forward | Οι attribute κόμβοι του context node. |
| namespace | forward | Οι namespace κόμβοι του context node. |

### *5.2.6 Συμπέρασμα*

*Ακριβώς λόγω της ομοιότητας των δύο γλωσσών, (XML και HTML) είναι βολικό να κάνουμε χρήση XML για να διαβάσουμε δεδομένα από ένα HTML έγγραφο (όπως είναι η πηγή μας).*

*Η δομή της XML είναι αυστηρή γλώσσα και ως εκ τούτου, μόνο και μόνο κάνοντας χρήση της, μπορεί κανείς να αποφύγει λάθη τα οποία θα προκύψουν από τυχόν προβλήματα δομής της πηγής, από την οποία αντλείται πληροφορία. Στην περίπτωσή μας, προτιμούμε να μη γίνει ανάρτηση νέας ανακοίνωσης, από το να γίνει λάθος ανάρτηση.*

*Η XML και η XPath υποστηρίζονται από την PHP (SimpleXMLElement, xpath(), DOMDocument...), άρα είναι ιδανικές για χρήση.*

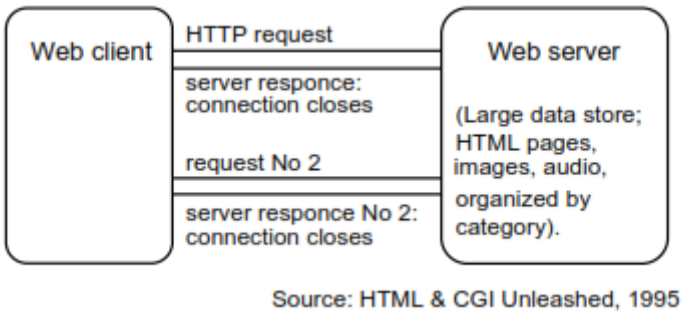
*Σημειωτέον: όλες οι τεχνολογίες αυτές είναι επίσης ελεύθερες και δωρεάν και χρησιμοποιούνται ευρέως από πάρα πολλά υπολογιστικά συστήματα.*

## 5.3 Πρωτόκολλο HTTP

### *5.3.1 Εισαγωγή*

Το πρωτόκολλο HTTP[33] είναι το πιο συνηθισμένο στον ηλεκτρονικό χώρο του World Wide Web και χρησιμοποιείται από τη συγκεκριμένη υπηρεσία του δικτύου Internet από το 1990. Το HTTP αποτελεί ένα πρωτόκολλο του επιπέδου εφαρμογών στα δίκτυα υπολογιστών και χρησιμοποιείται κυρίως σε διανεμημένα πληροφορικά συστήματα υπερμέσων. Είναι ένα γενικό, αντικειμενοστρεφές πρωτόκολλο που μπορεί να χρησιμοποιηθεί σε ένα πλήθος εφαρμογών, για παράδειγμα σε εξυπηρέτες - διανομείς (servers) και διανεμημένα συστήματα διαχείρισης αντικειμένων[33].

Τα μηνύματα του HTTP[34] μοιάζουν σημαντικά με αυτά των πρωτοκόλλων FTP (File Transfer) και NNTP (Network News). Η βασική τους διαφορά είναι ο stateless χαρακτήρας του HTTP που δεν εντοπίζεται στα υπόλοιπα. Η απουσία μνήμης κρίνεται αποδοτική (efficient) για το πρωτόκολλο όταν ένας σύνδεσμος (link) από ένα αντικείμενο οδηγεί σε ένα αντικείμενο που βρίσκεται αποθηκευμένο σε άλλο server. Επίσης[35] η ιδιότητα αυτή κρίνεται κατάλληλη εφόσον ο client επιστρέφει πληροφορία στον χρήστη με βάση URIs και όχι παλαιότερες ενέργειες του.




**Εικόνα 5.3 Stateless HTTP**

### *5.3.2 Λειτουργία*

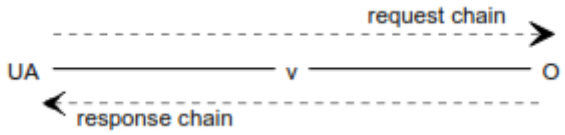
Το HTTP[34] ακολουθεί το μοντέλο request/response. Ο client εγκαθιδρύει μία σύνδεση με τον server (κάνοντας χρήση του πρωτοκόλλου TCP) και αποστέλλει μία αίτηση προς αυτόν η οποία περιέχει:

- Την μέθοδο που πρόκειται να εφαρμοστεί σαν αποτέλεσμα της αίτησης (request method). Η χρήση του όρου μέθοδος οφείλεται στον αντικειμενοστρεφή προσανατολισμό του πρωτοκόλλου.
- Ενα Universal Resource Identifier (URI). Ο πόρος στον οποίο πρόκειται να εφαρμοστεί η παραπάνω μέθοδος.
- Την έκδοση του χρησιμοποιούμενου πρωτοκόλλου.
- Ενα μήνυμα που ακολουθεί την μορφή MIME (Multipurpose Internet Mail Extensions) και περιέχει πληροφορία σχετικά με τον client, πιθανά το σώμα του μηνύματος κα.

Ο server απαντάει με ένα μήνυμα που περιέχει:

- Μία γραμμή κατάστασης (Status line) που περιέχει την έκδοση του πρωτοκόλλου και κωδικό επιτυχίας/αποτυχίας (success/error code).
- Ενα μήνυμα που ακολουθεί την μορφή MIME και περιέχει πληροφορία σχετικά με τον server, μεταπληροφορία σχετικά με το μεταφερόμενο αντικείμενο και πιθανά το σώμα του μηνύματος.

Η επικοινωνία HTTP συνήθως πραγματοποιείται μεταξύ ενός user agent (UA) και ενός origin server (O). Στην απλούστερη της μορφή αυτή η επικοινωνία πραγματοποιείται με μία μόνο σύνδεση (v) και έχει ως εξής:




**Εικόνα 5.4 Απλή μορφή HTTP επικοινωνίας.**

Η επικοινωνία HTTP γίνεται περισσότερο σύνθετη όταν μεταξύ του UA και του O (request/response chain) παρεμβάλλονται ενδιάμεσοι (intermediaries). Αυτοί εμφανίζονται σε τρεις μορφές: proxy, gateway και tunnel.

Ενας proxy ενδιάμεσος αποτελεί έναν πράκτορα προώθησης (forwarding agent) ο οποίος δέχεται αιτήσεις για κάποιο URI σε απόλυτη μορφή (absolute form), ανασκευάζει τα σχετικά μηνύματα μεταβάλλοντας όλα τα συστατικά τμήματα τους και τα προωθεί στον server ο οποίος προσδιορίζεται από το URI.

Ενας gateway ενδιάμεσος αποτελεί ένα πράκτορα παραλαβής (receiving agent) ο οποίος τοποθετείται στο αμέσως υψηλότερο επίπεδο από ορισμένους servers και μεταφράζει τις αιτήσεις στο πρωτόκολλο που οι servers αυτοί αντιλαμβάνονται και μπορούν να ερμηνεύσουν.

Ενας tunnel ενδιάμεσος λειτουργεί ως σημείο μεταγωγής (relay point) μεταξύ δύο συνδέσεων χωρίς ναπαρεμβαίνει στο περιεχόμενο των μηνυμάτων. Tunnels χρησιμοποιούνται όταν η επικοινωνία HTTP θα πρέπει να διέλθει από ενδιάμεσους όπως π.χ. firewalls.

Οπως επισημάνθηκε παραπάνω η HTTP επικοινωνία βασίζεται σε συνδέσεις του πρωτοκόλλου TCP/IP. Η εξ' ορισμού (default) TCP θύρα είναι η 80 αλλά δεν αποκλείεται η χρήση και άλλων.

Επίσης δεν αποκλείεται η πραγματοποίηση της HTTP επικοινωνίας πάνω από άλλα πρωτόκολλα μεταφοράς στο Internet ή σε άλλα δίκτυα. Η μόνη προϋπόθεση που τίθεται από το HTTP για το πρωτόκολλο του δικτυακού υποστρώματος είναι η αξιόπιστη μεταφορά (reliable transport).

Το πρωτόκολλο HTTP παρέχει πρόσβαση και σε άλλα πρωτόκολλα που χρησιμοποιούνται στο Internet, μεταξύ των οποίων τα ακόλουθα:

- File Transfer Protocol (FTP)
- Simple Mail Transfer Protocol (SMTP)
- Network News Transfer Protocol (NNTP)
- WAIS
- gopher
- Telnet and TN3270

Γενικά, οι συνδέσεις εκκινούνται από τον client πριν από την αποστολή της αίτησης και τερματίζονται από τον server μετά την αποστολή της απάντησης.

### *5.3.3 Μέθοδοι Αιτήσεων*

Στην 1.0 έκδοση του HTTP[35] υποστηρίζονται οι μέθοδοι GET, HEAD και PUT με τα εξής χαρακτηριστικά:

**GET**: Η μέθοδος GET αφορά στην ανάκτηση της οποιασδήποτε πληροφορίας (αντικειμένου) καθορίζεται από το URI της αίτησης (Request URI). Εάν το URI της αίτησης υποδεικνύει μία διαδικασία επεξεργασίας δεδομένων θα πρέπει να επιστραφούν, ως απάντηση, τα δεδομένα όπως αυτά προέκυψαν από την σχετική διαδικασία.

Μία αίτηση GET μπορεί να υποβληθεί υπό συγκεκριμένη συνθήκη (conditional GET). Στην περίπτωση αυτή, στην επικεφαλίδα της σχετικής αίτησης συμπεριλαμβάνεται το πεδίο If-modifiedsince.

Το προσδιοριζόμενο αντικείμενο ανακτάται μόνο στην περίπτωση που η ημερομηνία της τελευταίας ενημέρωσης/ μεταβολής του είναι πιο πρόσφατη από την ημερομηνία που καθορίζεται από το πεδίο If-modified-since. Η δυνατότητα conditional GET στοχεύει στην ελαχιστοποίηση του δικτυακού φόρτου επιτρέποντας την χρήση των cached αντιγράφων στους clients. Με τον μηχανισμό αυτό αποφεύγεται η ανταλλαγή περιττών δεδομένων στις περιπτώσεις αντικειμένων που δεν διακρίνονται για τις συχνές μεταβολές τους.

**HEAD**: Η μέθοδος αυτή είναι τελείως ανάλογη με την GET. Χρησιμοποιείται για τον έλεγχο συνδέσμων υπερκειμένου (hypertext links) σχετικά με την δυνατότητα πρόσβασης τους, την εγκυρότητα καθώς και ενδεχόμενες πρόσφατες μεταβολές τους. Δεν προβλέπεται η δυνατότητα conditional HEAD.

Στην μέθοδο HEAD ο server δεν επιστρέφει, στην απάντηση του, το σώμα της προσδιοριζόμενης πληροφοριακής οντότητας (πεδίο Entity-body) παρά μόνο μεταπληροφορία για αυτήν. Η επιστρεφόμενη μεταπληροφορία είναι η ίδια με την περίπτωση της μεθόδου GET. Οπως επισημάνθηκε παραπάνω χρησιμοποιείται κατά κύριο λόγο από τους browsers που εφαρμόζουν caching για την ανάκτηση αντικειμένων με βάση το πεδίο επικεφαλίδας Last-modified-since. Εάν η ημερομηνία αυτή είναι νεότερη από αυτή του αντικειμένου της cache ζητείται το περισσότερο πρόσφατο αντικείμενο. Οι μέθοδοι GET και HEAD έχει επικρατήσει να χρησιμοποιούνται μόνο για την ανάκτηση πληροφορίας (retrieval) και όχι για άλλες λειτουργίες.

**POST**: Η μέθοδος αυτή υποδεικνύει στον server να δεχτεί την οντότητα που μεταφέρεται στην αίτηση σαν ένα νέο στιγμιότυπο (εγγραφή, καταχώρηση) του πόρου που προσδιορίζεται από το URI. Η μέθοδος POST σχεδιάστηκε για την αντιμετώπιση αναγκών όπως:

- Υποβολή ενός μηνύματος σε μία bulletin board, newsgroup, mailing list ή παρόμοια συλλογή άρθρων.
- Πέρασμα παραμέτρων σε μία διαδικασία επεξεργασίας δεδομένων σαν αποτέλεσμα υποβολής φόρμας (form submission).
- Επέκταση μίας βάσης δεδομένων με την προσθήκη εγγραφών κα.

Η πραγματική λειτουργία η οποία εκτελείται σαν αποτέλεσμα της POST αίτησης προσδιορίζεται από τον server και συχνά εξαρτάται από το URI της αίτησης. Η οντότητα που περιέχεται στην αίτηση καθίσταται για τον πόρο που προσδιορίζεται στο URI ως ένα αρχείο για τον υπερκείμενο κατάλογο που το περιλαμβάνει ή μία εγγραφή για την βάση δεδομένων στην οποία ανήκει. Οι απαντήσεις σε POST αιτήσεις δεν επιδέχονται caching.

Οι αιτήσεις POST μπορεί να μην έχουν σαν αποτέλεσμα πόρους που είναι προσπελάσιμοι σε μελλοντική αναφορά (αντιπροσωπεύονται από

κάποιο URI). Το αποτέλεσμα των POST αιτήσεων (αναφορικά με τους πόρους που ενδεχομένως διαμορφώθηκαν) περιγράφεται στα success/error codes τα οποία επιστρέφονται από τον server.

### *5.3.4 Secure Sockets Layer (SSL)*

Το πρωτόκολλο SSL[36],[37] στοχεύει στην διασφάλιση της μυστικότητας (privacy) και της αξιοπιστίας (reliability) στην επικοινωνία μεταξύ δύο εφαρμογών. Αναπτύχθηκε από την Netscape Communications, υποβλήθηκε στο W3C ως πρόταση της εταιρίας για την υιοθέτηση του ως προτύπου, και είναι διαθέσιμο σε μορφή Internet Draft. Σχεδιάστηκε για την υποστήριξη πρωτοκόλλων επιπέδου εφαρμογής όπως τα HTTP, NNTP, FTP και Telnet. Αποτελείται από δύο επίπεδα (layers).

Στο κατώτερο επίπεδο του SSL τοποθετείται το SSL Record Protocol. Το SSL Record Protocol προϋποθέτει για την λειτουργία του ένα αξιόπιστο πρωτόκολλο μεταφοράς όπως το TCP και χρησιμοποιείται για την ενθυλάκωση (encapsulation) πρωτοκόλλων υψηλότερου επιπέδου. Ενα απόαυτά τα επίπεδα είναι το SSL Handshake Protocol. Το τελευταίο πρωτόκολλο επιτρέπει στον server και στον client να πιστοποιήσουν (authenticate) ο ένας τον άλλο (κάνοντας χρήση digital signature και certificate) και να διαπραγματευτούν αλγόριθμο και κλειδιά κρυπτογράφησης. Η διαπραγμάτευση αυτή γίνεται πριν την ανταλλαγή δεδομένων μεταξύ πρωτοκόλλων επιπέδου εφαρμογής. Το βασικό πλεονέκτημα του SSL είναι η ανεξαρτησία του από τα πρωτόκολλα αυτά.

Το πρωτόκολλο SSL παρέχει ασφάλεια σύνδεσης (connection security) η οποία έχει τρεις βασικές ιδιότητες:

- Η σύνδεση είναι ιδιωτική. Κρυπτογράφηση χρησιμοποιείται μετά από την αρχική χειραψία (handshake) για τον καθορισμό ενός μυστικού κλειδιού. Για την κρυπτογράφηση των δεδομένων χρησιμοποιούνται συμμετρικοί αλγόριθμοι όπως DES, RC4 κλπ.
- Η σύνδεση μπορεί να πιστοποιηθεί χρησιμοποιώντας μη συμμετρικό αλγόριθμο κρυπτογράφησης (ή δημόσιου κλειδιού) όπως RSA, DSS κλπ..
- Η σύνδεση είναι αξιόπιστη Στην μεταφορά των μηνυμάτων περιλαμβάνεται έλεγχος εγκυρότητας (integrity check) με την χρήση αλγορίθμου MAC (Message Authenticity Check) βάσει κλειδιών (keyed MAC).

Μία SSL σύνοδος διαθέτει μνήμη (stateful session). Ο συντονισμός των καταστάσεων του client και του server αποτελεί ευθύνη του SSL Handshake Protocol. Μία σύνοδος μπορεί να περιέχει πολλαπλές ασφαλείς συνδέσεις.

Το επίπεδο SSL Record δέχεται δεδομένα (μηνύματα) από τα ανώτερα στρώματα σε μη-κενά blocks που δεν έχουν κάποιο συγκεκριμένο μήκος. Τα blocks αυτά κερματίζονται (fragmentation) από το επίπεδο σε εγγραφές

(SSLPlaintext records) μέγιστου μήκους 214 bytes. Πολλαπλά μηνύματα ανωτέρων πρωτοκόλλων μπορούν να συμπεριληφθούν σε μία τέτοια εγγραφή. Οι εγγραφές συμπιέζονται (προαιρετικά) κάνοντας χρήση του αλγορίθμου που έχει οριστεί στην τρέχουσα κατάσταση της συνόδου. Με την εφαρμογή της συμπίεσης οι SSLPlaintext εγγραφές μετασχηματίζονται σε SSLCompressed δομές. Η συμπίεση είναι lossless.


**Εικόνα 5.5 Μηχανισμός ανταλλαγής μηνυμάτων μέσω SSL**

## 5.4 Curl

Η Curl [21], που χρησιμοποιείται για την αποστολή των δεδομένων στο Twitter είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού για διαδραστικές web εφαρμογές, στόχος της οποίας είναι να παρέχει μια ομαλότερη μετάβαση από τη μορφοποίηση στον προγραμματισμό. Καθιστά δυνατή την ενσωμάτωση πολύπλοκων αντικειμένων σε απλά έγγραφα χωρίς να χρειάζεται η εναλλαγή μεταξύ γλωσσών προγραμματισμού και πλατφόρμων ανάπτυξης.

Η γλώσσα προσπαθεί να αντιμετωπίσει ένα χρόνιο πρόβλημα: τα διάφορα δομικά στοιχεία που συνθέτουν κάθε σύγχρονο έγγραφο web συχνά απαιτούν διαφορετικές μεθόδους υλοποίησης: διαφορετικές γλώσσες, διαφορετικά εργαλεία, διαφορετικά πλαίσια, συχνά εντελώς διαφορετικές ομάδες. Το τελευταίο - και δυσκολότερο - εμπόδιο είναι η συνεργασία όλων αυτών των στοιχείων με αξιόπιστο και συνεπή τρόπο. Η Curl προσπαθεί να παρακάμψει αυτά τα προβλήματα με την παροχή ενός ακριβούς συντακτικού

και ενός σημασιολογικού περιβάλλοντος σε όλα τα επίπεδα της δημιουργίας web περιεχομένου : από την απλή HTML σε πολύπλοκες αντικειμενοστρεφείς εφαρμογές.

Η Curl συνδυάζει text markup ( όπως η HTML ) , scripting ( όπως η JavaScript ) , και heavy-duty computing ( όπως η Java , C # , C + + ), μέσα σε ένα ενιαίο πλαίσιο. Χρησιμοποιείται σε μια μεγάλη πληθώρα Business2Business και Business2Consumer εφαρμογών.

Αποτελεί μια γλώσσα σήμανσης, όπως η HTML, ενώ ταυτόχρονα περιλαμβάνει μια αντικειμενοστρεφή γλώσσα προγραμματισμού που υποστηρίζει πολλαπλή κληρονομικότητα. Οι Curl εφαρμογές δεν είναι υποχρεωμένες να τηρούν τον διαχωρισμό των πληροφοριών, το στυλ, τη συμπεριφορά που έχουν επιβάλλει η HTML και η JavaScript, ωστόσο αυτό το στυλ προγραμματισμού μπορεί να χρησιμοποιηθεί στην Curl.

Ενώ η γλώσσα Curl μπορεί να χρησιμοποιηθεί ως αντικατάσταση της HTML για την παρουσίαση μορφοποιημένου κειμένου, οι δυνατότητες της εκτείνονται και πέρα από αυτό. Τόσο η συγγραφή ( HTML - επίπεδο) όσο και οι προγραμματιστικές δομές της Curl μπορούν να επεκταθούν σε επίπεδο χρήστη. Η γλώσσα είναι σχεδιασμένη έτσι ώστε οι Curl εφαρμογές να μπορούν να γίνονται compile στο μηχάνημα του χρήστη και να τρέχουν σε μεγάλες ταχύτητες.

## 5.5 OAuth

### *5.5.1 Εισαγωγή*

Το OAuth[39] είναι ένα ανοιχτό πρότυπο πρωτόκολλου για έλεγχο ταυτότητας που επιτρέπει σε ένα χρήστη να χρησιμοποιήσει τις λειτουργίες των υπηρεσιών του Διαδικτύου, όπως αυτές που παρέχονται από το Facebook ή το Twitter, μέσα σε άλλες εφαρμογές (desktop, web, mobile, κλπ).

Σύμφωνα με την επίσημη ιστοσελίδα του[41] :

*Το OAuth είναι ένα πλαίσιο εξουδιοδότησης που επιτρέπει σε μια εφαρμογή τρίτου να αποκτήσει περιορισμένη πρόσβαση σε μια υπηρεσία HTTP.*

Πριν δημιουργηθεί το OAuth, υπήρχαν και άλλες μέθοδοι ελέγχου ταυτότητας που βοήθησαν στην προστασία των πιστοποιητικών των χρηστών από άλλες εφαρμογές, ενώ ταυτόχρονα επέτρεπαν την πρόσβαση στο API. Η Google, η Yahoo!, η AOL και η Amazon έχουν δημιουργήσει και χρησιμοποιούν τις δικές τους μεθόδους ελέγχου ταυτότητας.

Το OAuth 1.0 κυκλοφόρησε το 2007 και το OAuth 1.0 revision A, μια αναθεωρημένη έκδοση για καλύτερη ασφάλεια, κυκλοφόρησε το 2008.

Σήμερα, υπάρχει πλέον το OAuth 2.0 το οποίο δημιουργήθηκε από την ομάδα εργασίας IETF OAuth. Η διαδικασία εξακρίβωσής του είναι πολύ απλή, οπότε μια πληθώρα από υπηρεσίες επέλεξε να χρησιμοποιήσει μία από τις τελευταίες εκδόσεις του OAuth 2.0.

### *5.5.2 Βασικές Έννοιες*

- Πάροχος υπηρεσιών (Service Provider) - Ο Πάροχος Υπηρεσιών ελέγχει όλες τις πτυχές της εφαρμογής OAuth. Ο Πάροχος Υπηρεσιών είναι ο όρος που χρησιμοποιείται για να περιγράψει την ιστοσελίδα ή την web-υπηρεσία όπου βρίσκονται τα δεδομένα. Μπορεί να είναι μια ιστοσελίδα κοινής χρήσης φωτογραφιών, όπου οι χρήστες κρατούν άλμπουμ, μια ηλεκτρονική τραπεζική υπηρεσία, ή οποιαδήποτε άλλη υπηρεσία όπου αποθηκεύονται προσωπικά δεδομένα. Το OAuth δεν δίνει αναφέρει ότι ο πάροχος υπηρεσιών θα είναι επίσης ο πάροχος ταυτότητας που σημαίνει ότι ο Πάροχος Υπηρεσιών μπορεί να χρησιμοποιήσει τα δικά του ονόματα χρηστών και κωδικούς πρόσβασης για να εξουσιοδοτήσει τους χρήστες.

- Χρήστης (User) - Ο χρήστης είναι ο λόγος που το OAuth υπάρχει καθώς χωρίς τους χρήστες, δεν θα υπήρχε καμία ανάγκη για το OAuth. Οι χρήστες έχουν δεδομένα που δεν θέλουν να μοιραστούν με τον Πάροχο Υπηρεσιών, αλλά με ένα άλλο site. Στο OAuth, το πρωτόκολλο σταματά τουλάχιστον μία φορά προκειμένου να δοθεί άδεια από τον χρήστη και να επιτρέψει την πρόσβαση.

- Καταναλωτής (Consumer) – Μια εφαρμογή που προσπαθεί να έχει πρόσβαση στα δεδομένα του χρήστη. Αυτό μπορεί να είναι μια ιστοσελίδα, μια desktop εφαρμογή, μια φορητή συσκευή, ή οτιδήποτε άλλο συνδέεται με το διαδίκτυο. Ο καταναλωτής είναι αυτός που θέλει να πάρει την άδεια του χρήστη για να αποκτήσετε πρόσβαση στα δεδομένα.

- Tokens - Χρησιμοποιούνται αντί των διαπιστευτηρίων χρήστη για την πρόσβαση σε δεδομένα. Ένα token είναι γενικά μια τυχαία σειρά από γράμματα και αριθμούς που είναι μοναδικό, δύσκολο να το μαντέψει κανείς, και ζευγάρι με ένα μυστικό για την προστασία του Token από την κατάχρηση. Το OAuth ορίζει δύο διαφορετικούς τύπους token: Αίτησης και την πρόσβασης.

### *5.5.3 Διαδικασία εξακρίβωσης στοιχείων*

Η διαδικασία εξουσιοδότησης του OAuth περιλαμβάνει μια σειρά από αλληλεπιδράσεις μεταξύ της εφαρμογής web, των διακομιστών εξουσιοδότησης του Twitter και του τελικού χρήστη.

Σε βασικό επίπεδο, η διαδικασία έχει ως εξής[42]:

1. Η εφαρμογή ζητάει πρόσβαση και παίρνει ένα μη εγκεκριμένο token αίτησης από τον διακομιστή εξουσιοδότησης του Twitter.
2. Το Twitter ζητά από τον χρήστη να σας χορηγήσει την πρόσβαση των απαιτούμενων στοιχείων.
3. Η αίτησή σας παίρνει ένα εγκεκριμένο token αίτησης από το διακομιστή.
4. Ανταλλαγή του εγκεκριμένου token αίτησης με ένα token πρόσβασης.
5. Χρήση του token πρόσβασης προκειμένου να υπάρξει ανταλλαγή δεδομένων με τους διακομιστές του Twitter.

Όταν η εφαρμογή ζητήσει αρχικά την πρόσβαση στα δεδομένα ενός χρήστη, το Twitter εκδίδει ένα μη εξουσιοδοτημένο token αίτησης για την εφαρμογή.

Εάν ο χρήστης δεν είναι ήδη συνδεδεμένος , το Twitter ζητάει από το χρήστη να συνδεθεί. Έπειτα εμφανίζει μια σελίδα που αναφέρει στο χρήστη σε ποια δεδομένα ζητάει πρόσβαση η εφαρμογή.

Εάν ο χρήστης εγκρίνει το αίτημα πρόσβασης της εφαρμογής, το Twitter εκδίδει ένα εξουσιοδοτημένο token αίτησης. Κάθε token αίτησης ισχύει μόνο για μία ώρα. Μόνο εξουσιοδοτημένο token αίτησης μπορεί να ανταλλαγεί με ένα token πρόσβασης, και αυτή η ανταλλαγή μπορεί να γίνει μόνο μία φορά ανά εξουσιοδοτημένο token αίτησης.

Τα token πρόσβασης έχουν μεγάλη διάρκεια ζωής. Κάθε ένα είναι μοναδικό για το λογαριασμό χρήστη που καθορίζεται στην αρχική αίτηση εξουσιοδότησης και παρέχουν πρόσβαση μόνον στις υπηρεσίες που ορίζονται στο εν λόγω αίτημα. Τα πιστοποιητικά θα πρέπει να αποθηκεύονται σε ασφαλές μέρος, καθώς μέσω αυτών παρέχεται πρόσβαση στα δεδομένα του χρήστη.

### *5.5.4 OAuth 2*

Είναι δύσκολο να χρησιμοποιηθεί το OAuth 1.0 για μια εφαρμογή που δεν είναι διαδικτυακή εφαρμογή. Επιπλέον, η διαδικασία αυθεντικοποίησης και εξουσιοδότησης είναι πολύ περίπλοκη προκαλώντας μια επιβάρυνση για τον πάροχο υπηρεσιών.

Το OAuth 2.0[40] βελτιώνει αυτές τις αδυναμίες. Δεν είναι συμβατό με το OAuth 1.0 και ακόμα δεν υπάρχει τελική σταθερή έκδοση. Ωστόσο, πολλές

εταιρείες υπηρεσιών του Διαδικτύου χρησιμοποιούν ήδη OAuth 2.0. Τα παρακάτω είναι τα κύρια χαρακτηριστικά του OAuth 2.0[38]:

- Ενισχυμένη υποστήριξη για εφαρμογές, όχι για εφαρμογές web
- Δεν χρειάζεται κρυπτογράφηση
- Χρησιμοποιεί HTTPS, όχι HMAC
- Απλοποιημένη υπογραφή
- Δεν χρειάζεται ταξινόμηση και κωδικοποίηση URL
- Βελτίωση του Token πρόσβασης

Όταν ιδρύθηκε το token πρόσβασης στο OAuth 1.0, είχε μεγάλη διάρκεια ζωής. Στο Twitter, το token πρόσβασης δεν λήγει. Για μεγαλύτερη ασφάλεια, το OAuth 2.0 παρέχει τη δυνατότητα να καθοριστεί η διάρκεια ζωής του token πρόσβασης.

Η ορολογία που χρησιμοποιείται από το OAuth 2.0 είναι εντελώς διαφορετική από εκείνη του OAuth 1.0. Τα δύο πρωτόκολλα μπορεί να εξυπηρετούν τον ίδιο σκοπό, αλλά είναι εντελώς διαφορετικά.

## 5.6 JSON

Το JSON[18] χρησιμοποιείται από το API του Twitter για την ανταλλαγή πληροφοριών και δεν είναι κάτι που έχουμε επιλέξει οι ίδιοι, αλλά πολύ πιθανόν θα το επιλέγαμε. Ωστόσο, εάν ήταν δυνατή η επιλογή να επικοινωνήσουμε μέσω XML, θα επιλέγαμε αυτήν τη μέθοδο, εφόσον ήδη έχουμε φέρει την πληροφορία στη μορφή αυτή. Θεωρείται σκόπιμο να αναλυθούν κάποια τμήματα του προτύπου αυτού.

Το JSON[19] (JavaScript Object Notation) είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και να το γράψουν. Είναι εύκολο για τις μηχανές να το αναλύσουν (parse) και να το παράγουν (generate). Είναι βασισμένο πάνω σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript. Το JSON είναι ένα πρότυπο κειμένου το οποίο είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού, αλλά χρησιμοποιεί πρακτικές (conventions) οι οποίες είναι γνωστές στους προγραμματιστές της οικογένειας προγραμματισμού C, συμπεριλαμβανομένων των C, C++, C#, Java, JavaScript, Perl, Python, και πολλών άλλων. Αυτές οι ιδιότητες κάνουν το JSON μια ιδανική γλώσσα προγραμματισμού ανταλλαγής δεδομένων.

Το JSON είναι χτισμένο σε δύο δομές[20]:

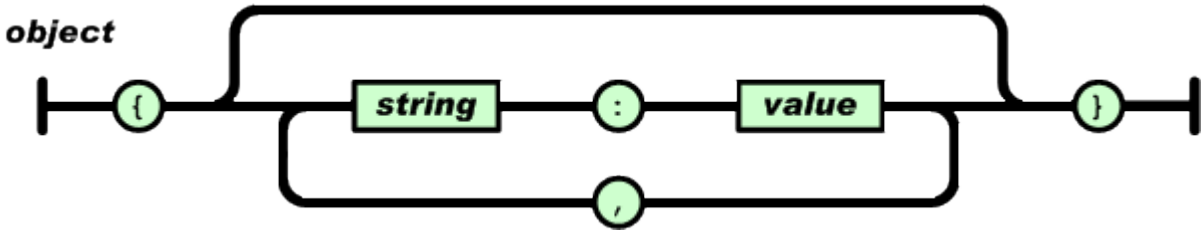
- Μια συλλογή από ζευγάρια ονομάτων/τιμών. Σε διάφορες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένα object, καταχώρηση, δομή, λεξικό, πίνακα hash (hash table), λίστα κλειδιών, ή associative πίνακα.

- Μία ταξινομημένη λίστα τιμών. Στις περισσότερες γλώσσες προγραμματισμού, αυτό αντιλαμβάνεται ως ένας πίνακας (array), διάνυσμα, λίστα, ή ακολουθία.

Αυτά είναι τα universal data structures. Ουσιαστικά όλες οι μοντέρνες γλώσσες προγραμματισμού τα υποστηρίζουν με τον έναν ή τον άλλον τρόπο. Λογικό είναι πως ένα πρότυπο δεδομένων το οποίο είναι εύκολα μεταβαλλόμενο με γλώσσες προγραμματισμού οι οποίες επίσης είναι βασισμένες σε αυτές τις δομές.

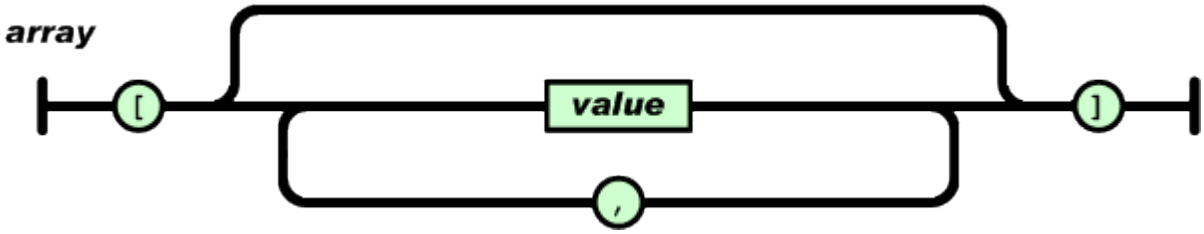
Στο JSON, παίρνουν αυτές τις μορφές:

Ένα αντικείμενο (object) είναι ένα άτακτο σύνολο από ζευγάρια ονόματων/τιμών. Ένα αντικείμενο (object) ξεκινάει με { (αριστερό άγκιστρο) και τελειώνει με } (δεξιό άγκιστρο). Κάθε όνομα ακολουθείται από : (άνω-κάτω τελεία) και τα ζευγάρια ονόματος/τιμής χωρίζονται από , (κόμμα).



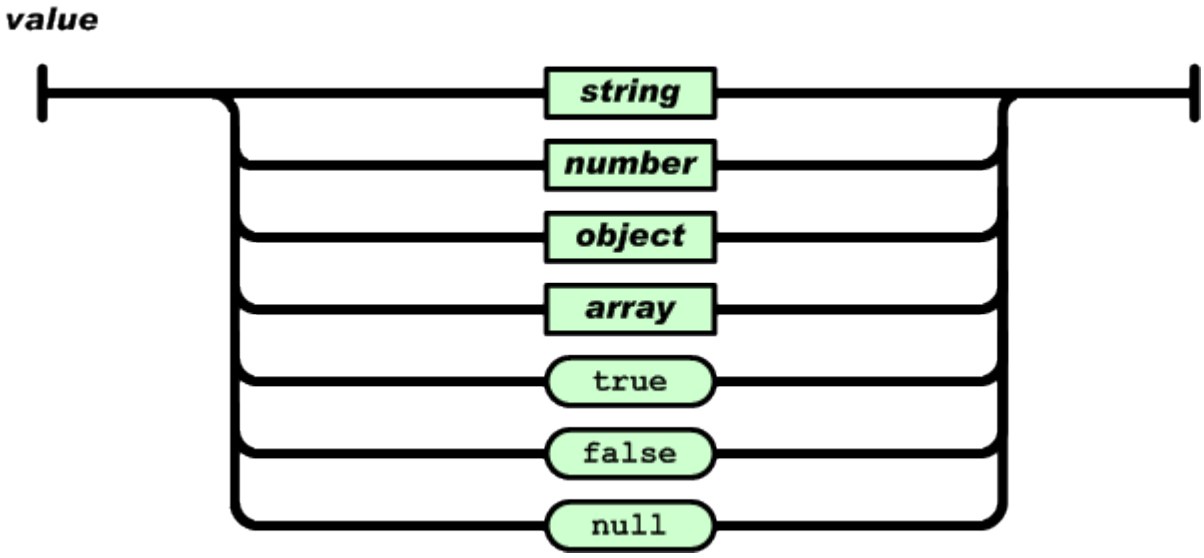

**Εικόνα 5.6 Διάταξη αντικειμένου**

Ένας πίνακας (array) είναι μια συλλογή από τιμές σε σειρά. Ένας πίνακας (array) ξεκινάει με [ (αριστερή αγγύλη) και τελειώνει με ] (δεξιά αγγύλη). Οι τιμές χωρίζονται με , (κόμμα).



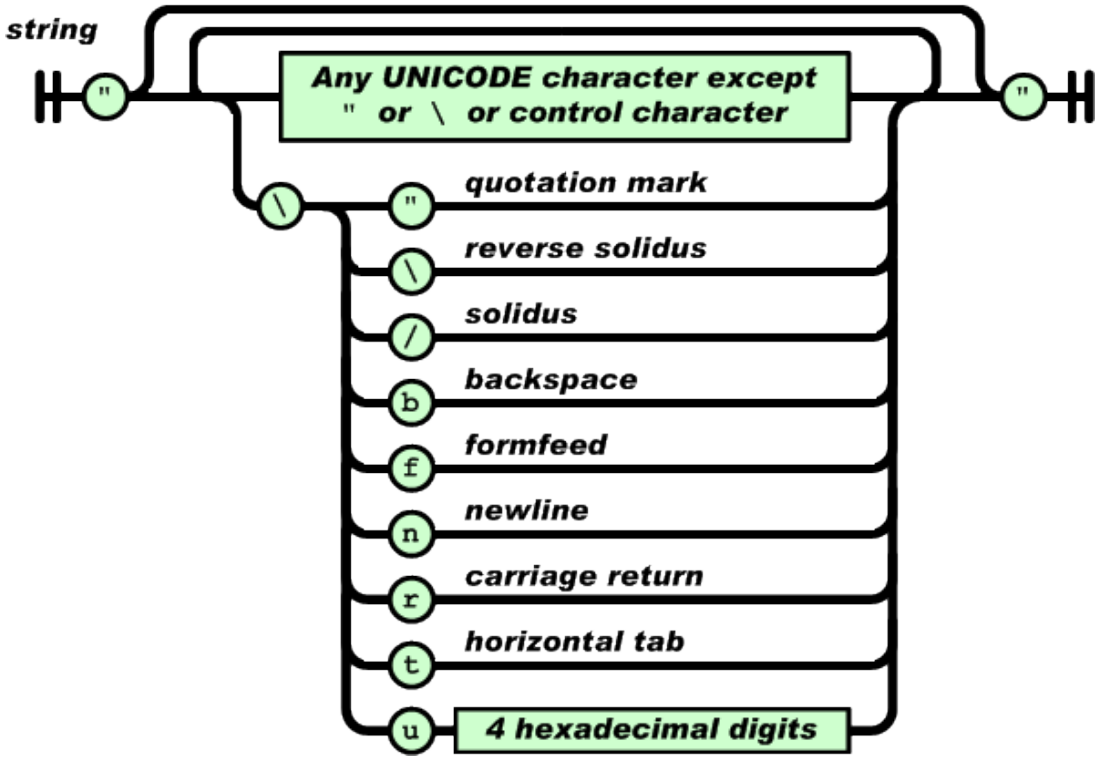

**Εικόνα 5.7 Διάταξη πίνακα**

Μία τιμή μπορεί να είναι string μέσα σε διπλά quotes, ή αριθμός (number), ή true ή false ή null, ή αντικείμενο (object) ή πίνακας (array). Αυτές οι τιμές μπορεί να είναι και ανακατεμμένες.



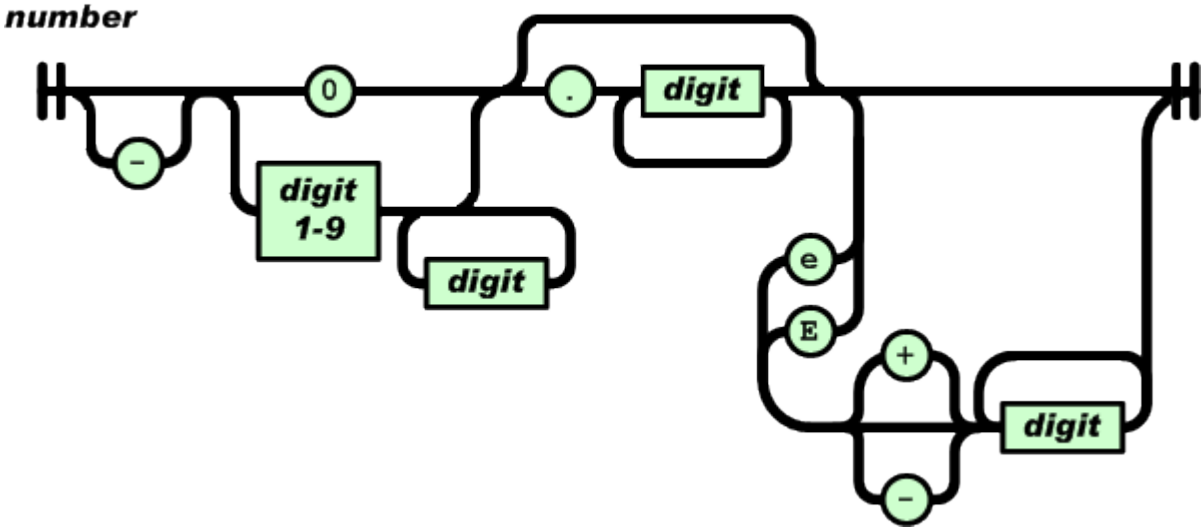

**Εικόνα 5.8 Διάταξη τιμής**

Ένα string είναι μια συλλογή από μηδέν ή περισσότερους Unicode χαρακτήρες, μέσα σε διπλά quotes, χρησιμοποιώντας αντίστροφους κάθετους \ (backslash) για escapes. Ένας χαρακτήρας αντιπροσωπεύεται ως ένας μονός χαρακτήρας string. Ένα string μοιάζει πολύ σαν ένα C ή Java string.




**Εικόνα 5.9 Διάταξη String**

Ένας αριθμός (number) μοιάζει πάρα πολύ με ένα C ή Java αριθμό (number), με την διαφορά πως τα οκταδικά και δεκαεξαδικά συστήματα δεν χρησιμοποιούνται.




**Εικόνα 5.10 Διάταξη αριθμού**

Τα κενά (whitespace) μπορούν να εισαχθούν ανάμεσα σε οποιοδήποτε ζευγάρι tokens. Με εξαίρεση μερικών λεπτομερειών κωδικοποίησης (encoding), αυτό περιγράφει γενικότερα την γλώσσα (προγραμματισμού).

## 5.7 Twitter API

Το Twitter[6] παρέχει δύο ειδών API, τα οποία υποστηρίζουν διάφορες δυνατοτήτες: το REST API και το Streaming API. Η διαφορά των δύο έγκειται στο γεγονός ότι στο δεύτερο πρέπει να υπάρχει μια μόνιμα ανοιχτή σύνδεση με το Twitter, ενώ στην πρώτη γίνεται απλά ένα αίτημα στο Twitter και στέλνεται μια απάντηση, που μπορεί να περιέχει έναν κωδικό επιτυχίας ή αποτυχίας ή/και κάποια άλλα δεδομένα (όπως το περιεχόμενο ενός tweet, τους followers ενός λογαριασμού κτλ.).

Εκτός από αυτά τα δύο βασικά APIs, παρέχεται και ένα επιπλέον API ονόματι Search API. Αυτό μπορεί να χρησιμοποιηθεί για την μαζική ανάκτηση αναρτήσεων με ταυτόχρονο φιλτράρισμα, σύμφωνα με διάφορες παραμέτρους. Το Twitter υποστηρίζει πως οι αναρτήσεις που επιστρέφονται με χρήση της συγκεκριμένης διεπαφής δεν είναι όλες οι δυνατές (γιατί δεν προστίθενται σε ευρετήρια όλες τις αναρτήσεις). Δεν καταφέραμε να εντοπίσουμε με ποια κριτήρια γίνεται η επιλογή αν κάποια ανάρτηση θα προστεθεί στο index ή όχι. Στην ουσία το Search API αποτελεί υποκατηγορία του REST API, ωστόσο το ίδιο το Twitter τα διαχωρίζει, οπότε θεωρούμε σημαντικό να αναφερθεί το παρόν, έστω περιληπτικά.

Ο τρόπος λειτουργίας του REST API θα περιγραφεί αναλυτικότερα παρακάτω, εφόσον είναι αυτό που χρησιμοποιήσαμε, ενώ για τη λειτουργία και τις δυνατότητες του Streaming API θα κάνουμε μια συντομότερη περιγραφή.

### *5.7.1 Streaming API*

Η διεπαφή αυτή επιτρέπει σε προγραμματιστές εφαρμογών να δημιουργούν μια σύνδεση χαμηλής καθυστέρησης στο παγκόσμιο ρεύμα δεδομένων των Tweets[31]. Μια καθώς πρέπει υλοποίηση επιτρέπει σε μια εφαρμογή να δέχεται (με push από το διακομιστή) ενημερώσεις συμβάντων, αποφεύγοντας την καθυστερήση που δημιουργείται λόγω αιτήσεων - ερωτημάτων που γίνονται στο REST API.

Το Twitter παρέχει διάφορα σημεία στα οποία μπορεί να υπάρχει μια τέτοια σύνδεση, το κάθε ένα από τα οποία έχει παραμετροποιηθεί για συγκεκριμένες περιπτώσεις χρήσης.

- Public streams: ρεύματα δημόσιων δεδομένων που κινούνται στο Twitter. Είναι χρήσιμα για την παρακολούθηση συγκεκριμένων χρηστών ή θεμάτων και για εξόρυξη δεδομένων.
- User streams: Ρεύματα ενός μοναδικού χρήστη, που περιέχουν (σχεδόν) όλα τα δεδομένα που έχουν να κάνουν με το τι βλέπει κάποιος χρήστης στο Twitter. Αυτό το ρεύμα μπορεί να χρησιμοποιηθεί για την ανάπτυξη μιας εφαρμογής για κάποιο κινητό τηλέφωνο για παράδειγμα, με την οποία ο χρήστης θα μπορεί να παρακολουθεί το Twitter.
- Site streams: Η έκδοση πολλών χρηστών που χρησιμοποιούν user streams. Τα συγκεκριμένα υπάρχουν για χάρη διακομιστών οι οποίοι πρέπει να συνδέονται στο Twitter για λογαριασμό πολλαπλών χρηστών.

#### *5.7.1.1 Public streams*

Τα παρακάτω καταληκτικά σημεία (endpoints) παρέχουν δείγματα δημόσιων δεδομένων που ρέουν στο Twitter. Μόλις μια εφαρμογή δημιουργήσει μια σύνδεση σε ένα endpoint, αποστέλλονται σε αυτό tweets, χωρίς την ανάγκη αίτησης.

Τα παρακάτω συνδετικά σημεία παρέχονται από το Streaming API και αφορούν public streams:

POST statuses/filter

GET statuses/sample

GET statuses/firehose

#### *Συνδέσεις*

Κάθε λογαριασμός Twitter μπορεί να δημιουργήσει συνδέσεις παντού (σε κάθε endpoint), ωστόσο μπροεί να διατηρήσει ανοιχτή μια μόνον σύνδεση.

Αν επιχειρηθεί η δημιουργία νέας σύνδεσης σε ένα άλλο endpoint, ενώ υπάρχει ήδη μια ενεργή, τότε η παλιότερη σύνδεση θα αποσυνδεθεί αυτόματα.

Πελάτες οι οποίοι κάνουν κατάχρηση αιτημάτων συνδέσεων, είτε επιτυχών είτε ανεπιτυχών, κινδυνεύουν να γίνει αυτόματα αποκλεισμός της διεύθυνσης IP τους.

#### *5.7.1.2 User streams*

Αυτού του είδους οι ροές παρέχουν δεδομένα και συμβάντα που είναι αποκλειστικά συνδεδεμένα με το λογαρασμό ενός αυθεντικοποιημένου χρήστη. Δεν είναι δυνατή η χρήση τους για συνδέσεις μεταξύ δύο διακομιστών. Σε αυτήν την περίπτωση πρέπει να γίνει χρήστη του site stream endpoint.

Καταληκτικό σημείο: GET user

### *Συνδέσεις*

Αν θέλουμε να χρησιμοποιήσουμε αυτού του είδους endpoint, πρέπει να λαμβάνουμε υπόψιν ότι μπορεί για κάθε λογαριασμό Twitter να υπάρχουν μόνο κάποιος περιορισμένος αριθμός συνδέσεων τέτοιου είδους για κάθε εφαρμογή OAuth (βλ. κεφ. 5), άσχετα από τις διευθύνσεις IP. Αν υπερβούμε τον αριθμό αυτόν, η παλαιότερη σύνδεση τερματίζεται. Ένας λογαριασμός στον οποίο συνδέονται πολλές διαφορετικές εκτελέσεις (instances) μιας εφαρμογής Oauth, θα προκαλεί συνέχεια αποσύνδεση της παλιότερης κάθε φορά, οπότε καταλήγουμε σε μόνιμες αποσυνδέσεις και επανασυνδέσεις.

Σε τέτοια περίπτωση πρέπει η εφαρμογή να μπορεί να χειριστεί το HTTP error code 420, που σημαίνει ότι γίνονται αλλεπάλληλες επανασυνδέσεις στο λογαριασμό. Αν λάβει ο χρήστης αυτόν τον κωδικό (ή τη μετάφραση του από την εφαρμογή σε κάποιο μήνυμα), πρέπει να κλείσει όσο το δυνατόν περισσότερες εκτελέσεις της εφαρμογής (π.χ. σε άλλους υπολογιστές, κινητές συσκευές κτλ.) για να γίνει πάλι δυνατή η σύνδεση στο λογαριασμό.

Βεβαίως αυτός ο περιορισμός ισχύει για κάθε εφαρμογή χωριστά (δηλαδή για κάθε ξεχωριστό αναγνωριστικό εφαρμογής OAuth). Επίσης υπάρχει περιορισμός στον αριθμό συνδέσεων ανά IP διεύθυνση, άσχετα με το ποιες εφαρμογές εκτελούνται. Εφαρμογές οι οποίες επιτρέπουν την ταυτόχρονη θέαση λογαριασμών Twitter μπορούν να διατηρήσουν μια μόνο ενεργή σύνδεση ανά λογαριασμό.

#### *5.7.1.3 Site streams*

Τα endpoints αυτά επιτρέπουν σε υπηρεσίες (όπως ιστοσελίδες) να δέχονται ενημερώσεις σε πραγματικό χρόνο για λογαριασμό πολλών χρηστών. Μπορούν να αποσταλλούν συμβάντα για κάθε χρήστη που έχει χορηγήσει OAuth πρόσβαση στην εφαρμογή. Εφαρμογές με λίγους χρήστες ή εφαρμογές επιφάνειας εργασίας πρέπει κατά κανόνα να χρησιμοποιούν τα User streams. Προς το παρόν αυτές οι ροές και τα αντίστοιχα endpoints

μπορούν να χρησιμοποιηθούν μόνο από συγκεκριμένους λογαριασμούς στους οποίους έχει επιτραπεί κατ' εξαίρεση, μια και βρίσκεται ακόμη σε beta στάδιο.

Καταληκτικό σημείο: GET site

### *Προστατευμένα δεδομένα*

Οι εφαρμογές που κάνουν χρήση του συγκεκριμένου είδους ροής, επιβάλλεται να κάνουν φιλτράρισμα των δεδομένων που θα παρουσιαστούν στον τελικό χρήστη, μια και αυτή η ροή μεταφέρει και δημόσια και ιδιωτικά δεδομένα:

Τα δεδομένα πρέπει να είναι μη προσβάσιμα εξ' ορισμού και να μπορούν να γίνουν προσβάσιμα μόνο ανώνυμα, αφού αποδειχθεί πως είναι δημόσια. Τα δεδομένα που είναι ιδιωτικά πρέπει να είναι ορατά μόνο στους χρήστες που μπορούν να αποδείξουν την ταυτότητά τους.

### *Σύνδεση*

Τα Site streams δέχονται και απαντούν μόνο αιτήματα από τον αυθεντικοποιημένο ιδιοκτήτη/δημιουργό της εφαρμογής. Κατά τη σύνδεση γίνεται και ο καθορισμός των λογαριασμών χρηστών, οι οποίοι έχουν αποδεχθεί και επικυρώσει την πρόσβαση μέσω OAuth στην εφαρμογή κάποια στιγμή στο παρελθόν.

Ροές για μεγάλο αριθμό χρηστών χρησιμοποιούν σημαντικό μέγεθος εύρους ζώνης και πρέπει να ληφθεί υπόψη, πως εφαρμογές, οι οποίες δεν δύνανται να διαβάσουν τα δεδομένα με την ταχύτητα που στέλνονται, αποσυνδέονται αυτόματα από το σύστημα.

## *5.7.2 REST API*

### *5.7.2.1 Έκδοση*

Το Rest API του Twitter, το οποίο και χρησιμοποιήσαμε, πρόσφατα άλλαξε έκδοση (1.1 από 1.0). Αυτό οδήγησε στη διακοπή της ορθής λειτουργίας όλων των εφαρμογών (μέσα σε αυτές και της δικής μας), οι οποίες πρέπει να ενημερωθούν κατάλληλα, ώστε να χρησιμοποιούν το νέο τρόπο επικοινωνίας. Οι αλλαγές που έγιναν, στοχεύουν κυρίως στην παροχή αυξημένης ασφάλειας.

### *5.7.2.2 Χαρακτηριστικά των RESTful διεπαφών*

Το REST (Representational State Transfer) API του Twitter λειτουργεί, φυσικά, όπως όλες οι RESTful[6] διεπαφές. Για αυτόν το λόγο, κρίνεται απαραίτητο να επισημανθούν κάποια βασικά χαρακτηριστικά των RESTful API. Ωστόσο, δε θα γίνει ανάλυση της ίδιας της αρχιτεκτονικής λογισμικού REST, πέραν των περιορισμών που θέτει.

*6 APIs τα οποία διατηρούν τους περιορισμούς της αρχιτεκτονικής REST αποκαλούνται RESTful*

Τα RESTful APIs είναι μια πολύ δημοφιλής επιλογή, όταν πρόκειται να δημιουργηθεί μια διεπαφή προγραμματισμού εφαρμογών για διαδικτυακές εφαρμογές, λόγω της απλής υλοποίησης τους και της εύκολα κατανοητής χρήσης τους. Είναι διεπαφές που έχουν ως κύριο προσόν τις γρήγορες επιδόσεις, τη δυνατότητα κλιμάκωσης και την μεταφερσιμότητα ή επαναχρησιμοποίηση. Παρέχουν επίσης αυξημένη αξιοπιστία και ευκολία επέκτασης και τροποποίησης.

Όλα αυτά τα πλεονεκτήματα όμως προϋποθέτουν τη συνεχή συμμόρφωση με τους περιορισμούς που θέτει το REST. Αν μια υπηρεσία παραβεί κάποιον περιορισμό, δε μπορεί να θεωρείται πλέον RESTful. Αυτοί οι περιορισμοί είναι οι εξείς:

1. Uniform interface: Πρέπει να χρησιμοποιείται μια ενιαία διεπαφή, μέσω της οποίας επικοινωνεί κάθε πελάτης με τους διακομιστές.
2. Μοντέλο Client–Server: Μια καθολική διεπαφή διαχωρίζει τους διακομιστές από τους πελάτες, πράγμα που σημαίνει πως οι πελάτες και οι διακομιστές λειτουργούν τελείως αυτόνομα. Οι μεν διακομιστές διατηρούν τα δεδομένα (οπότε μπορεί να γίνει κλιμάκωση εύκολα), ενώ οι πελάτες έχουν τη δυνατότητα να διατηρούν το portability του κώδικά τους. Εφόσον δεν αλλάξει η διεπαφή, τα δύο μέρη μπορούν να διατηρούνται ή να συντηρούνται, να αλλάζουν και να επεκτείνονται παράλληλα και ανεξάρτητα.
3. Stateless protocol: Οι διακομιστές πρέπει να μπορούν να απαντήσουν σε κάθε αίτηση πελάτη μέσω της διεπαφής, χωρίς να αποθηκεύουν δεδομένα σε σχέση με το κάθε αίτημα. Η κατάσταση του session διατηρείται από τον πελάτη και αποτελεί ευθύνη του ίδιου να δημιουργήσει νέα αιτήματα, αλλάζοντας έτσι κατάσταση. Ενόσω εκκρεμούν αιτήσεις, η κατάσταση θεωρείται μεταβατική. Συνοψίζοντας, στα stateless πρωτόκολλα το κάθε αίτημα θεωρείται άσχετο από τα προηγούμενα και η επικοινωνία αποτελείται από ένα ζευγάρι αιτήματος - απάντησης.
4. Cachable responses: Οι πελάτες πρέπει να γνωρίζουν αν μπορούν να διατηρούν απαντήσεις σε αιτήματα που έχουν θέσει. Αυτό δηλώνει πως οι απαντήσεις των διακομιστών πρέπει να είναι ορισμένες ως cachable ή μη. Αν οι διακομιστές μπορούν να δώσουν cachable απαντήσεις (έχει σημασία εδώ πόσο καλός είναι και ο σχεδιασμός εκτός των άλλων), υπάρχει δυνατότητα ακόμη καλύτερης διαχείρισης της κλιμάκωσης και αύξησης επιδόσεων του συστήματος συνολικά.
5. Layered systems: Ο πελάτης δεν μπορεί να γνωρίζει συνήθως, αν τελικά πήρε απάντηση από τον κεντρικό διακομιστή ή από την cache κάποιου ενδιάμεσου, που χρησιμοποιείται για αύξηση των επιδόσεων και της κλιμάκωσης του συστήματος.

Όλες οι RESTful διεπαφές, πρέπει να καθορίζονται με βάση τα παρακάτω:

1. Διαθέτουν ένα base URI (Universal Resource Identifier)
2. Υποστηρίζουν ένα Internet Media Type (JSON, XML, Atom, κ.ά.)
3. Υποστηρίζουν τουλάχιστον κάποιες από τις standard HTTP μεθόδους (GET, PUT, POST…)
4. Χρησιμοποιούν hypertext υπερσυνδέσμους για να αλλάξει η κατάσταση ενός session (state) και για να παραθέσουν συσχετιζόμενους πόρους

Το REST API του Twitter υποστηρίζει μόνο τις μεθόδους GET και POST και η επικοινωνία επιτυγχάνεται με την ανταλλαγή JSON αντικειμένων. Το αντικείμενο που αποστέλλεται για παράδειγμα κατά την αίτηση ανάρτησης ενός νέου tweet, αναγνωρίζεται με τη χρήση του υπερσυνδέσμου https://api.twitter.com/1.1/statuses/update.json, ενώ για να λάβει κανείς τις τελευταίες είκοσι αναρτήσεις ενός λογαριασμού πρέπει να θέσει το αίτημα με χρήση του https://api.twitter.com/1.1/statuses/mentions_timeline.json.

#### *5.7.2.3 Δυνατότητες του REST API*

Το REST API του Twitter παρέχει μεγάλο αριθμό από ενέργειες που μπορεί να κάνει κάποια εφαρμογή – πελάτης. Κάποιες βασικές ενέργειες παρατίθενται εδώ, χωρισμένες σε κατήγορίες:

*Timeline manipulation*

GET statuses/user_timeline

GET statuses/home_timeline

GET statuses/mentions_timeline

*Tweets manipulation*

POST statuses/update

POST statuses/retweet/:id

*Search*

GET search/tweets

*Direct Messages*

GET direct_messages

GET direct_messages/new

*Friends & Followers*

GET friends/ids

GET followers/ids

*Users*

GET account/verify_credentials

GET users/search

*OAuth*

GET oauth/authenticate

Αυτές είναι μόνο μερικές σημαντικές ενέργειες αυτών των κατηγοριών και πιθανότατα οι πιο πολυχρησιμοποιημένες. Υπάρχουν μια σειρά από διαφορετικές κατηγορίες ενεργειών, όπως Lists, Favorites, Places/Geo, Trends, Help και άλλες. Οι ενέργειες που καλούνται κατά την εκτέλεση της εφαρμογής μας αναλύονται στο κεφάλαιο 6.

#### *5.7.2.4 Περιορισμοί*

Η REST διεπαφή του Twitter θέτει κάποιους περιορισμούς στον αριθμό αιτημάτων ανά χρήστη και ανά εφαρμογή. Αυτοί οι περιορισμοί μπορεί να ποικίλλουν, ανάλογα με την ενέργεια. Υπάρχουν τρόποι να απαγγιστρωθεί κανείς από τέτοιους περιορισμούς, αν αυτό είναι σκόπιμο. Ειδικά στην περίπτωση των εφαρμογών, κάτι τέτοιο μπορεί να είναι εξαιρετικά χρήσιμο, γιατί μια εφαρμογή μπορεί να χρησιμοποιείται από μεγάλο αριθμό χρηστών. Κάποιες τέτοιες λύσεις είναι η χρήση caching (τοπικά) ή η παροχή προτεραιότητας σε ενεργούς χρήστες (αν για παράδειγμα πρόκειται για απόκτηση πληροφοριών των timelines χρηστών).

Ακόμη, είναι δυνατό να χρησιμοποιήσει κανείς το Streaming API αντί του REST, στο οποίο οι περιορισμοί αριθμών αιτημάτων είναι κατάλληλα προσαρμοσμένοι για συνδέσεις μακράς διάρκειας, εφόσον όμως το API αυτό είναι ακόμη σε πειραματικό στάδιο, η δυνατότητα αυτή δίνεται μόνο σε όσους έχουν πρόσβαση σε αυτό.

Σε περίπτωση που μια εφαρμογή ή ένας χρήστης υπερβαίνει τα όρια αυτά συχνά, ενδέχεται να θεωρηθεί πως γίνεται για κακόβουλα αίτια και να γίνει blacklisted, οπότε και δε θα λαμβάνει απαντήσεις από το Twitter API.

# 6 Ενδεικτικά συστατικά της υπηρεσίας

Στη συνέχεια θα παραθέσουμε και θα αναλύσουμε τα βασικότερα σημεία της υπηρεσίας. Θα περιγράψουμε τις μεθόδους που καλούνται καθ' όλη τη διάρκεια εκτέλεσης και έχουν να κάνουν με την απόκτηση, επεξεργασία και αποστολή της πληροφορίας. Επίσης, θα γίνει αναφορά και επεξήγηση του τρόπου με τον οποίο πρέπει να οριστούν κάποιες παράμετροι, έτσι ώστε να γίνει ένα επιτυχμένο tweet, για παράδειγμα οι μεταβλητές που χρειάζονται για τη σύνδεση στο Twitter.

## 6.1 Ορισμός παραμέτρων - constants

Οι μεταβλητές που χρειαζόμαστε για την επιτυχή εκτέλεση της υπηρεσίας, είναι παράμετροι που ορίζουν ποια είναι η πηγή μας, η σχετική θέση των συνδέσμων (κάθε ανακοίνωσης) στον ιστότοπο, η σχετική τοποθεσία μέσα στο έγγραφο (ιστοσελίδα HTML) του κάθε τίτλου και υπερσυνδέσμου, όπως και τα δεδομένα αναγνωριστικά που πρέπει να χρησιμοποιήσουμε για την επικοινωνία με το Twitter. Αυτές οι παράμετροι ορίζονται ως constants στην κύρια κλάση. Η εφαρμογή είναι έτσι φτιαγμένη, ώστε να μη χρειάζεται αλλαγή του κώδικα που εκτελείται, πέραν των παραμέτρων αυτών, αν θέλουμε να κάνουμε κάποια αλλαγή. Σε περίπτωση που θέλουμε να αλλάξουμε πηγή για παράδειγμα, αρκεί να δώσουμε στις αντίστοιχες παραμέτρους τις ορθές τιμές.

Ορίζουμε λοιπόν ως πηγή την ιστοσελίδα των δημόσεων ανακοινώσεων της Ύδρας και ως βάση των εσωτερικών υπερσυνδέσμων το ίδιο το domain.

```
define("SOURCE",
"https://hydra.it.teithe.gr/s/index.php?nocache=1402345472&m=itdep-bbstud");

define("BASE", "http://hydra.it.teithe.gr");
```

Έτσι έχουμε ότι χρειαζόμαστε και έχει να κάνει με την απόκτηση της πληροφορίας, αλλά έχουμε αποκτήσει και το αρχικό κομμάτι των υπερσυνδέσμων που δείχνουν σε κάθε ανακοίνωση χωριστά, οπότε μπορούμε να συνεχίσουμε με τον ορισμό των μεταβλητών που βοηθούν στην επεξεργασία της πληροφορίας.

Οι πληροφορίες μας βρίσκονται μέσα σε ένα πίνακα κλάσης “vehi-list” και κάθε μια από αυτές περιέχονται μέσα σε μια ξεχωριστή σειρά-εγγραφή του πίνακα αυτού, κλάσης “data”. Έτσι, για το διαχωρισμό αυτών των δεδομένων από τα υπόλοιπα του εγγράφου που θα φορτώσουμε (π.χ. τα headers, το login button, άλλα tabs πέραν των δημόσιων ανακοινώσεων που δε μας ενδιαφέρουν) χρησιμοποιούμε σαν διαδρομή στην XPATH την παρακάτω.

```
define("XPATH",
```

```
'//table[@class="vehi-list"]/tr[@class="data"]');'
```

Ακόμη όμως δεν έχει τελειώσει ο τελικός διαχωρισμός της πληροφρίας, αφού πρέπει να έχουμε στα χέρια μας έναν τίτλο-κείμενο (τουλάχιστον) και κάποιον υπερσύνδεσμο προς μια συγκεκριμένη ανακοίνωση. Καθορίζουμε λοιπόν, ότι κάθε ανακοίνωση (χρήσιμη πληροφορία) βρίσκεται στην σειρά - εγγραφή μέσα σε κόμβους παιδιά με το tag “td”, όπως δηλαδή σε κάθε πίνακα HTML.

```
define("NEWS", serialize(array("children", "td")));
```

Και εσωτερικά για κάθε ανακοίνωση, όπως φαίνεται παρακάτω, το κείμενο του τρίτου απογόνου τύπου “td” (δηλαδή το νούμερο δύο) αποτελεί τον τίτλο, ενώ το πέμπτο (νούμερο τέσσερα) έχει κάποιο δικό του απόγονο τύπου anchor, του οποίου το attribute “href” μας δίνει το υπόλοιπο κομμάτι του υπερσυνδέσμου (το πρώτο το έχουμε ήδη σε μια μεταβλητή που ορίστηκε παραπάνω ως βάση). Το μηδέν που εμφανίζεται, δηλώνει το πρώτο anchor που θα βρεθεί, εδώ προφανώς έχουμε μόνο έναν τέτοιο κόμβο για κάθε ανακοίνωση.

```
define("TITLE", serialize(array("2", "text")));

define("LINK", serialize(array("4", "children", "a", "0", "attributes",
"href")));
```

Είναι φανερό πως εδώ πλέον έχουμε να κάνουμε με πίνακες της PHP[7]. Γενικά η πληροφορία μας περιέχεται μέσα σε έναν συσχετιστικό πίνακα, ο οποίος περιέχει συσχετιστικούς πίνακες. Αυτό που αποθηκεύουμε στις παραπάνω μεταβλητές είναι ουσιαστικά η θέση μέσα στον πίνακα του κάθε τίτλου και κάθε συνδέσμου. Ορίζοντας τα αντίστοιχα indexes μέσα σε αυτό το array μπορούμε να αποκτήσουμε την ημερομηνία που έγινε μια ανακοίνωση και τον δημιουργό της, ή ακόμη και συνδέσμους αρχείων που έχουν να κάνουν με αυτή.

```
define("ACCESS_TOKEN", "----------");

define("ACCESS_TOKEN_SECRET", "----------");

define("CONSUMER_KEY", "----------");

define("CONSUMER_SECRET","----------");
```

Τέλος, καθορίζουμε τις μεταβλητές που χρησιμοποιούνται για την αναγνώριση της εφαρμογής από το Twitter και το λογαριασμό στον οποίο θέλουμε να αναπαράγονται οι ανακοινώσεις.

*7 Θα εξηγηθεί παρακάτω, πως μετατρέπουμε την πληροφορία που έχουμε λάβει τελικά σε PHP associative array. Αυτό το κάνουμε γιατί είναι σαφώς πιο εύκολη η χρήση ενός array (και ταχύτερη) για την πρόσβαση σε δεδομένα.*

## 6.2 Βασικά τμήματα της κεντρικής / κύριας κλάσης - εργάτη

Ο κύριος στόχος της εφαρμογής είναι η επικοινωνία με το API του Twitter και η δημιουργία νέων αναρτήσεων. Για να επιτύχουμε κάτι τέτοιο, χρειαζόμαστε καταρχήν έναν τρόπο να επικοινωνήσουμε μαζί του, οπότε δημιουργούμε ένα αντικείμενο τύπου TwitterAPIExchange, δηλαδή της βιβλιοθήκης μας, υπεύθυνο να κάνει αυτές τις ενέργειες.

```
$twitter=Helper::initTwitter(ACCESS_TOKEN,
ACCESS_TOKEN_SECRET, CONSUMER_KEY,
CONSUMER_SECRET);
```

Εκτελούμε τη μέθοδο initTwitter() της βοηθητικής κλάσης, δίνοντάς της ως παραμέτρους τα απαραίτητα αναγνωριστικά και εκχωρούμε το αποτέλεσμά της σε μια μεταβλητή, την οποία θα χρησιμοποιούμε από εδώ και πέρα κάθε φορά που θέλουμε να επικοινωνήσουμε με το Twitter.

Ο επόμενος στόχος που πρέπει να εκπληρωθεί για την αποστολή ενός νέου Tweet είναι η απόκτηση της πληροφορίας που θέλουμε να μεταδώσουμε. Αυτήν την πληροφορία αντλούμε και αποθηκεύουμε σε μεταβλητές της PHP για να την φιλτράρουμε και να την επεξεργαστούμε, πριν επιχειρήσουμε να την αποστείλουμε ως tweet.

Αρχικά λοιπόν δημιουργούμε ένα αντικείμενο DOMDocument. Η χρήση ενός τέτοιου τύπου αντικειμένου είναι απαραίτητη, όταν θέλουμε να αποθηκεύσουμε τα περιεχόμενα μιας ιστοσελίδας. Τα έγγραφα αυτά έχουν μεταβλητές οι οποίες μπορούν να περιέχουν το σύνολο των κόμβων ενός εγγράφου DOM και παρέχουν μεθόδους για την πρόσβαση στους κόμβους και την επεξεργασία τους.

```
$contents = new DOMDocument;

$contents->loadHTML(file_get_contents(SOURCE));

file_put_contents('new.xml', $contents->saveXml());
```

Την απόκτηση των δεδομένων επιτυγχάνουμε με τη χρήση της php μεθόδου file_get_contents, που δέχεται ως παράμετρο τη διαδρομή σε ένα αρχείο, που μπορεί να βρίσκεται στο διαδίκτυο ή και τοπικά σε κάποιο αποθηκευτικό μέσο. Η πληροφορία αυτή όμως στην περίπτωσή μας θέλουμε να φορτωθεί σε ένα έγγραφο DOM, άρα την έξοδο της μεθόδου αυτής χρησιμοποιούμε ως παράμετρο στη μέθοδο loadHTML του DOM εγγράφου. Με αυτόν τον τρόπο, φορτώνονται τελικά στο έγγραφο - μεταβλητή οι κόμβοι του αρχικού εγγράφου (ιστοσελίδα).

Έπειτα, και εφόσον θέλουμε να χρησιμοποιήσουμε XML για την επεξεργασία των δεδομένων, φροντίζουμε να μετατρέψουμε το έγγραφο από HTML σε XML. Αυτό είναι δυνατό μόνον όταν η HTML του εγγράφου είναι

valid. Στην περίπτωση της Ύδρας αυτό ισχύει[8]. Ουσιαστικά, δεν αλλάζει τίποτα πέραν της προσθήκης πληροφορίας στην αρχή του εγγράφου, η οποία δηλώνει πως το έγγραφο είναι τύπου XML.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

Όταν δεν παρέχονται συγκεκριμένες παράμετροι στην εντολή που κάνει τη μετατροπή, όπως στην περίπτωσή μας, η έκδοση ορίζεται πως είναι η 1.0, η κωδικοποίηση UTF-8 και το έγγραφο θεωρείται πως δεν ακολουθεί κάποιο σχήμα.

Έπειτα φορτώνουμε σε μια μεταβλητή τα περιεχόμενα του αρχείου που δημιουργήσαμε, υπό την μορφή SimpleXMLElements και τα CDATA φορτώνονται ως strings. Αυτό επιτυγχάνεται με τη χρήση της τελευταίας παραμέτρου.

```
$simplexml=simplexml_load_file('new.xml',

'SimpleXMLElement', LIBXML_NOCDATA);
```

Τώρα μπορούμε να χρησιμοποιήσουμε XPATH πάνω σε αυτά τα SimpleXMLElements για να ξεχωρίσουμε τα κομμάτια της πληροφορίας που μας είναι χρήσιμα.

```
$xpath = $simplexml->xpath(XPATH);
```

Έτσι πια, έχουμε στη νέα μεταβλητή μόνο τους κόμβους οι οποίοι είναι χρήσιμοι και περιέχουν ανακοινώσεις. Όλοι αυτοί οι κόμβοι είναι απόγονοι του κόμβου που ορίζει τον πίνακα των ανακοινώσεων (δηλαδή του αρχικού HTML table που είναι και ο αρχικός κόμβος γονέας). Ακόμη βέβαια δεν έχουμε αποκτήσει την καθαρή πληροφορία.

Για να καταφέρουμε κάτι τέτοιο, θα χρησιμοποιήσουμε PHP arrays. Συγκεκριμένα, θα μετατρέψουμε το SimpleXMLElement σε συσχετιστικό πίνακα. Αυτό θα μας βοηθήσει ιδιαίτερα στην ταχύτατη και εύκολη πρόσβαση στα δεδομένα (τίτλο, ημερομηνία, κτλ). Η μετατροπή γίνεται με την κλήση της μεθόδου transform2Array() της βοηθητικής κλάσης.

```
$newsArray=array_reverse( Helper::transform2Array($xpath));
```

Αυτή η μέθοδος δέχεται ως παράμετερο ένα αντικείμενο τύπου SimpleXMLElement και επιστρέφει έναν συσχετιστικό πίνακα, χωρίς να χαθεί καθόλου πληροφορία[9]. Η μέθοδος array_reverse() της PHP χρησιμοποιείται για να αντιστραφεί η σειρά των δεδομένων, μιας και διαβάζουμε πρώτα την πιο πρόσφατη ανακοίνωση (άρα είναι η πρώτη εισαγωγή στον πίνακα), αλλά θέλουμε να την αναρτήσουμε τελευταία.

*8 Την εγκυρότητα ενός εγγράφου μπορεί κανείς να ελέγξει εύκολα με τη χρήση ενός εργαλείου που παρέχει το W3C και βρίσκεται στην παρακάτω διεύθυνση: http://validator.w3.org/*

*9 Άλλες μέθοδοι, όπως η χρήση του συνδυασμού json_decode και json_encode για παράδειγμα, οδηγούν δυστυχώς σε απώλεια πληροφορίας, λόγω κακής συνεργασίας με τα SimpleXMLElements της PHP.*

Ως εδώ, αυτό που έχουμε καταφέρει είναι η απόκτηση ενός PHP array με περιεχόμενα όπως το παρακάτω σε κάθε index:

## Παράδειγμα στοιχείου του πίνακα γονέα

```
Array
( [0] => Array
        ( [name] => td [text] => Kostoglou B.
            [attributes] => Array
                ( [class] => left)
            [children] => Array ()
        )
    [1] => Array
        ( [name] => td [text] => 20 Sep 2013
            [attributes] => Array
                ( [class] => center )
            [children] => Array()
        )
    [2] => Array
        ( [name] => td
           [text] => Πρόγραμμα 2ης εβδομάδας εξετάσεων, 23/9 - 27/9
            [attributes] => Array
                ( [class] => center)
            [children] => Array()
        )
    [3] => Array (…)
    [4] => Array
        ( [name] => td
            [text] =>
            [attributes] => Array
                (
                    [class] => center
                )

            [children] => Array
                (
                    [a] => Array
                        ( [0] => Array
                                (
                                    [name] => a
                                    [text] => View
                                    [attributes] => Array
                                        ( [href] =>
/s/index.php?nocache=1402880580&m=itdep-
bbstud&s=&bid=13&bbviewid=4933
                                            [class] => veh-link
)
                                    [children] => Array()
)))))
```

Εδώ πλέον γίνεται φανερό, πως χρησιμοποιούνται και τα indexes που έχουν οριστεί ως πίνακες (στα constants). Υπενθυμίζουμε, για παράδειγμα, ότι το “text” του τρίτου index (δηλαδή το νούμερο δύο) περιέχει τον τίτλο της ανακοίνωσης.

Στη συνέχεια ακολουθεί ένας βρόχος επανάληψης τύπου foreach, κάνοντας χρήση του οποίου μας δίνεται η δυνατότητα να διαβάσουμε τον τίτλο της κάθε ανακοίνωσης και τον σύνδεσμο, και στη συνέχεια μπορούμε να επιχειρήσουμε να δημιουργήσουμε ένα νέο tweet για κάθε ανακοίνωση. Η προσπάθεια ανάρτησης γίνεται μέσα στο βρόχο.

```
$title = trim(Helper::getValue(unserialize(TITLE), $newDiv));

$subLink = Helper::getValue(unserialize(LINK), $newDiv);
```

Για να καταφέρουμε να λάβουμε τον τίτλο για παράδειγμα, καλούμε την μέθοδο getValue() της βοηθητικής κλάσης. Αυτή η μέθοδος δέχεται ως παράμετρο έναν πίνακα που περιέχει indexes (όπως ορίστηκε στην αρχή) και έναν συσχετιστικό πίνακα, τον οποίο καλείται να διατρέξει. Η πληροφορία που επιστρέφεται είναι ένα string, που στην περίπτωσή μας είναι ο τίτλος ή ο υπερσύνδεσμος. Τέτοιες κλήσεις γίνονται για να αποκτήσουμε και τα υπόλοιπα δεδομένα, αν αυτό θεωρείται απαραίτητο. Το αποτέλεσμα της κλήσης, δηλαδή το string, το εκχωρούμε σε μια μεταβλητή, αφού όμως πρώτα αφαιρέσουμε πιθανούς κενούς χαρακτήρες από την αρχή και το τέλος του με τη μέθοδο trim() της PHP. Εφόσον έχουμε εκχωρήσει τις τιμές του συνδέσμου και του τίτλου, μπορούμε πλέον να ετοιμάσουμε το tweet.

Το Twitter θέτει περιορισμό 140 χαρακτήρων για κάθε ανάρτηση. Αυτός ο περιορισμός μας αναγκάζει να ελέγχουμε το μήκος της πιθανής ανάρτησης και να επεμβαίνουμε, αποκόπτοντας τμήματά της, όταν αυτή ξεπερνά το επιτρεπόμενο μήκος.

```
$titleTemp = Helper::reduceLength($title);
```

Ο στόχος αυτός επιτυγχάνεται με την κλήση μιας νέας μεθόδου της βοηθητικής κλάσης, ονόματι reducelength(). Αυτή δέχεται τον τίτλο ως παράμετρο και φροντίζει να τον φέρει στο επιθυμητό μήκος. Προφανώς δε μπορούμε να επέμβουμε στο άλλο στοιχείο που θα αναρτηθεί, δηλαδή τον υπερσύνδεσμο, γιατί μετά δε θα λειτουργεί.

Στο ενδιάμεσο έχουμε φροντίσει να κάνουμε κατάλληλους ελέγχους για το αν έχουμε κάνει παρόμοιο tweet, έτσι ώστε να αποφύγουμε περιττές κλήσεις στο Twitter API. Αν φτάσουμε σε τέτοιο σημείο, η εκτέλεση του προγράμματος διακόπτεται και επιστρέφεται η ροή στο λειτουργικό σύστημα.

Ενώ όμως αυτή η συνθήκη δεν ισχύει, μπορούμε να επιχειρήσουμε να δημιουργήσουμε ένα νέο tweet. Η κλήση προς το API γίνεται με τη βοήθεια φυσικά του αντικειμένου TwitterApiExchange που έχουμε δημιουργήσει.

```
$twitter->buildOauth($apiURL, $requestMethod)

            ->setPostfields($postfields)

            ->performRequest(true);
```

Χρησιμοποιείται method chaining, ώστε να συνδεθούμε στο Twitter API στο κατάλληλο URL (statuses/update), κάνοντας την κατάλληλη κλήση και αποστέλλοντας την απαραίτητη πληροφορία. Αν η κλήση γίνει επιτυχώς, το API μας επιστρέφει το αντικείμενο που στείλαμε υπό μορφή JSON, ενώ αν αποτύχει μας επιστρέφεται πάλι ένα αντικείμενο JSON, που περιέχει κάποιον κωδικό λάθους και το αντίστοιχο μήνυμα. Για τη δομή του αντικειμένου αυτού θα γίνει λόγος στο τέλος του κεφαλαίου, αλλά αναλυτικότερα για τις μεθόδους που γίνονται αλυσιδωτά κατά την κλήση του API μπορεί να διαβάσει κανείς αμέσως παρακάτω.

## 6.3 Βασικά τμήματα της βιβλιοθήκης

Η βιβλιοθήκη TwitterAPIExchange περιέχει όλες τις απαραίτητες μεθόδους για την παραγωγή strings, που αναπαριστούν URLs, με τη χρήση των οποίων επιτυγχάνεται η επικοινωνία με το Twitter. Μπορεί να χρησιμοποιηθεί η μέθοδος GET της HTTTP αλλά και η POST. Θεωρείται περιττό να αναλυθεί το πως κτίζονται αυτά τα strings. Ωστόσο ιδιαίτερο ενδιαφέρον παρουσιάζει η μέθοδος, που τελικά εκτελεί το κάθε αίτημα προς το API. Αυτή η μέθοδος δεν είναι άλλη από την performRequest(). Χρησιμοποιεί curl για την εκτέλεση των αιτημάτων, χρησιμοποιώντας έναν πίνακα με παραμέτρους.

```
$options = array(
    CURLOPT_HTTPHEADER => $header,
    CURLOPT_HEADER => false,
    CURLOPT_URL => $this->url,
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_TIMEOUT => 10,
    CURLINFO_HEADER_OUT => true,
    CURLOPT_SSL_VERIFYPEER => true,
    CURLOPT_SSL_VERIFYHOST => 2,
    CURLOPT_CAINFO => getcwd() . "/CAcert.crt"
);
```

Αυτές οι παράμετροι ορίζουν τον τρόπο με τον οποίο θα πραγματοποιηθεί το αίτημα. Σημαντικό είναι να αναφέρουμε, ότι από την έκδοση 1.1 του API είναι υποχρεωτική η χρήση ενός Certificate, που φροντίζει για την ασφαλή επικοινωνία. Αυτό το πιστοποιητικό παρέχει το ίδιο το Twitter.

Σημειώνει βέβαια πως μπορεί να αλλάξει ανά πάσα στιγμή, οπότε η εφαρμογή μπορεί να σταματήσει να λειτουργεί.

Αφού καθοριστούν λοιπόν οι απαραίτητες παράμετροι της curl, πρέπει να οριστεί το περιεχόμενο του αιτήματος:

```
if (!is_null($postfields)) {

    $options[CURLOPT_POSTFIELDS] = $postfields;

} else {

    if ($getfield !== '') {

        $options[CURLOPT_URL] .= $getfield;

    }

}
```

Προσθέτουμε ως επιπλέον παράμετρο λοιπόν το POST ή το GET, που έχουμε ήδη ορίσει, και είμαστε έτοιμοι να αποστείλουμε το αίτημα.

```
$feed = curl_init();

curl_setopt_array($feed, $options);

$json = curl_exec($feed);

curl_close($feed);
```

Αποθηκεύουμε σε μια μεταβλητή ονόματι $json την επιστροφή της curl_exec(), για την περίπτωση που μας είναι χρήσιμο αυτό το δεδομένο. Αν είναι, το επιστρέφουμε στην καλούσα κλάση:

```
if ($return) {

    return $json;

}
```

Το αντικείμενο τύπου JSON που επιστρέφεται, μπορεί να περιέχει κάποιο μήνυμα λάθους μαζί με τον αντίστοιχο κωδικό σε περίπτωση αποτυχίας ή τον κωδικό 200, που σημαίνει πως το αίτημα πέτυχε. Σε περίπτωση επιτυχίας (ανάλογα με τον τύπο του αιτήματος) το API μπορεί να επιστρέφει περιεχόμενο, που μπορεί να περιγράφει ένα tweet, ένα χρήστη, μία λίστα από επαφές και άλλα.

Στην περίπτωση της statuses/update, που εκτελούμε εμείς, και εφόσον όλα πάνε καλά, επιστρέφεται το περιεχόμενο του POST αντικειμένου σε μορφή JSON από τη διεπαφή.

## 6.4 Βασικά τμήματα της βοηθητικής κλάσης

```
public static function getValue($indexes, $arrayToAccess) {
    if (count($indexes) > 1) {
        return Helper::getValue(array_slice($indexes, 1),
$arrayToAccess[$indexes[0]]);
    } else {
        return $arrayToAccess[$indexes[0]];
    }
}
```

Η μέθοδος getValue() δέχεται ως παραμέτρους έναν πίνακα από indexes και έναν πίνακα από τον οποίο θέλουμε να αντλήσουμε πληροφορία. Λειτουργεί αναδρομικά και, για κάθε στοιχείο του πίνακα indexes επιστρέφει κάθε φορά στην προηγούμενη κλήση έναν υποπίνακα που περιέχει το ζητούμενο, ώσπου να καταλήξει σε αυτό, οπότε σταματάει η αναδρομή και επιστρέφεται μια τιμή στην καλούσα κλάση.

```
public static function transform2Array($simplexml) {
    $i = 0;
    $array = array();
    foreach ($simplexml as $new) {
        $array[$i++] = Helper::xmlObjToArr($new);
    }
    return $array;
}
```

Η μέθοδος transform2Array() δέχεται ως παράμετρο ένα αντικείμενο τύπου SimpleXMLElement και, με τη βοήθεια της xmlObjToArr(), το μετατρέπει σε PHP array. Η xmlObjToArr() διαβάζει όλα τα δεδομένα τα οποία περιέχει το αντικείμενο SimpleXMLElement που της παραδίδεται και δημιουργεί συσχετιστικούς πίνακες για κάθε XML tag, attribute, value του αρχείου προέλευσης του αντικειμένου αυτού. Καλεί αναδρομικά τον εαυτό της και ως εκ τούτου, κάθε παραγόμενος συσχετιστικός πίνακας περιέχεται μέσα σε έναν γονέα, μέχρι να φτάσει στην ρίζα. Το σώμα αυτής της μεθόδου μπορεί να βρει κανείς στο παράρτημα, όπου γίνεται παράθεση όλου του κώδικα.

Στο τέλος φυσικά επιστρέφεται ένας πίνακας της PHP στην προηγούμενη μέθοδο transform2Array(), η οποία με τη σειρά της τοποθετεί όλες τις ανακοινώσεις στην περίπτωση μας σε έναν δικό της συσχετιστικό πίνακα και τον επιστρέφει στην κύρια κλάση.

Μια ακόμη μέθοδος της βοηθητικής κλάσης, στην οποία έχει γίνει αναφορά, είναι η reduceLength(). Αυτή δέχεται ως παράμετερο ένα string και κάποιον αριθμό, και φροντίζει να μειώσει το μήκος του string, ώστε να είναι μικρότερο από τον αριθμό αυτόν. Έτσι επιτυγχάνουμε να μη γίνονται αιτήματα αναρτήσεων στο API του Twitter με απαγορευμένο μήκος, οπότε και μειώνουμε τις περιττές κλήσεις που θα αποτύγχαναν χωρίς αποτέλεσμα.

```
public static function reduceLength($str, $length = 100) {

    echo strlen($str) . ": " . $str . "<br>";

    if (strlen(html_entity_decode($str)) > $length) {

        $lastSpacePosition = strrpos($str, ' ');

        $str = substr($str, 0, $lastSpacePosition);

        if (strlen($str) > $length) {

            Helper::reduceLength($str);

        }

    }

    return $str;

}
```

Και αυτή η μέθοδος είναι αναδρομική. Καλεί τον εαυτό της μειώνοντας κάθε φορά το string κατά μία λέξη, εντοπίζοντας τον τελευταίο κενό χαρακτήρα και αποκόπτοντας ότι βρίσκεται μετά από αυτόν. Στο τέλος, βέβαια, επιστρέφεται στην καλούσα κλάση ένα string με αποδεκτό μήκος.

## 6.5 Περιγραφή αντικειμένου JSON

Εφόσον είναι ξεκάθαρος πλέον ο τρόπος επικοινωνίας με το REST API του Twitter, μπορούμε πλέον να δώσουμε ένα παράδειγμα αντικειμένου που αποστέλλεται μεταξύ διακομιστή και εφαρμογής.

Η εφαρμογή μας δημιουργεί ένα αντικείμενο POST το οποίο αποστέλλεται στο Twitter μέσω της διεπαφής, περικλεισμένο από τα κατάλληλα HTTP headers. Το API στη συνέχεια επιστρέφει αυτό το

αντικείμενο, συμπληρωμένο με όλες τις μεταβλητές που το περιγράφουν, ακόμη και αν εμείς δεν τις αποστείλαμε:


```
{
  "created_at":"Mon Jun 16 22:49:22 +0000 2014",
  "id":478670698442129409,
  "id_str":"478670698442129409",
  "text":"\u03a0\u03c1\u03cc\u03b3\u03c1\u03b1\u03bc\u03bc\u03b1 2\u03b7\u03c2 \u03b5\u03b2\u03b4\u03bf\u03bc\u03ac\u03b4\u03b1\u03c2 \u03b5\u03be\u03b5\u03c4\u03ac\u03c3\u03b5\u03c9\u03bd, 23\/9 - 27\/9 ... http:\/\/t.co\/AmBm7VvlkP",
  "source":"\u003ca href=\"https:\/\/github.com\/jinxed-hydra\/Hydra\" rel=\"nofollow\"\u003eHydraBay\u003c\/a\u003e",
  "truncated":false,
  "in_reply_to_status_id":null,
  "in_reply_to_status_id_str":null,
  "in_reply_to_user_id":null,
  "in_reply_to_user_id_str":null,
  "in_reply_to_screen_name":null,
  "user":{
    "id":1648421870,
    "id_str":"1648421870",
    "name":"hydrabay",
    "screen_name":"hydrabay",
    "location":"Thessaloniki",
    "description":"TESTING TWITTER API 1.1",
    "url":"http:\/\/t.co\/RZbJ5773ma",
    "entities":{
      "url":{
        "urls":[
          {
            "url":"http:\/\/t.co\/RZbJ5773ma",
            "expanded_url":"http:\/\/hydra.it.teithe.gr\/s\/",
            "display_url":"hydra.it.teithe.gr\/s\/",
            "indices":[
              0,
              22
            ]
          }
        ]
      },
      "description":{
        "urls":[

        ]
      }
    },
    "protected":false,
    "followers_count":1,
    "friends_count":0,
    "listed_count":0,
    "created_at":"Mon Aug 05 18:41:19 +0000 2013",
    "favourites_count":0,
    "utc_offset":10800,
    "time_zone":"Athens",
    "geo_enabled":false,
    "verified":false,
    "statuses_count":883,
    "lang":"en",
    "contributors_enabled":false,
    "is_translator":false,
    "is_translation_enabled":false,
    "profile_background_color":"C0DEED",
    "profile_background_image_url":"http:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png",
    "profile_background_image_url_https":"https:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.pn
```

```
g",
    "profile_background_tile":false,
    "profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/378800000256967114\/f7f85291d8b1
01ab313d885169948894_normal.jpeg",
    "profile_image_url_https":"https:\/\/pbs.twimg.com\/profile_images\/378800000256967114\/f7f852
91d8b101ab313d885169948894_normal.jpeg",
    "profile_link_color":"0084B4",
    "profile_sidebar_border_color":"C0DEED",
    "profile_sidebar_fill_color":"DDEEF6",
    "profile_text_color":"333333",
    "profile_use_background_image":true,
    "default_profile":true,
    "default_profile_image":false,
    "following":false,
    "follow_request_sent":false,
    "notifications":false
  },
  "geo":null,
  "coordinates":null,
  "place":null,
  "contributors":null,
  "retweet_count":0,
  "favorite_count":0,
  "entities":{
    "hashtags":[

    ],
    "symbols":[

    ],
    "urls":[
      {
        "url":"http:\/\/t.co\/AmBm7VvlkP",
        "expanded_url":"http:\/\/tinyurl.com\/obmyxpv",
        "display_url":"tinyurl.com\/obmyxpv",
        "indices":[
          51,
          73
        ]
      }
    ],
    "user_mentions":[

    ]
  },
  "favorited":false,
  "retweeted":false,
  "possibly_sensitive":false,
  "lang":"el"
}
```


Από όλα τα παραπάνω, εμείς αποστείλαμε μόνο το “text” και ένα “url” στο API. Ωστόσο, η διεπαφή μας επιστρέφει ένα ολόκληρο αντικείμενο, τηρώντας τους κανόνες του REST. Αυτό σημαίνει πως οφείλει να μας επιστρέψει το αντικείμενο ως όλον και όχι κάποια αφαίρεσή του. Στην αντίθετη περίπτωση, το API δεν θα ήταν πια RESTful. Οπότε επιστρέφεται πληροφορία όπως η σελίδα που έγινε η ανάρτηση, ο χρόνος ανάρτησης, το μοναδικό αναγνωριστικό που ορίζει πλέον το tweet που κάναμε (unique ID) και άλλα.

## 6.6 Output εκτέλεσης εφαρμογής

Η εφαρμογή κατά την εκτέλεσή της παράγει έξοδο στην οθόνη (όταν εκτελείται μέσω κάποιου φυλλομετρητή) και εξάγει δεδομένα επιτυχίας ή αποτυχίας και σε ένα αρχείο καταγραφής (logfile). Τα δεδομένα αυτά μπορούν να χρησιμοποιηθούν για λόγους debugging ή στατιστικής. Συγκεκριμένο παράδειγμα από το αρχείο καταγραφής:

*1402884953: success - Μικρή διόρθωση στο πρόγραμμα εξετάσεων εργαστηρίων*

*#######*

*Summary:*

*S: 1*

*F: 0*

*Script Execution Time: 22 seconds*

*#######*

Μπορεί κανείς να δει πότε πέτυχε ή απέτυχε η δημιουργία μιας ανάρτησης (UNIX timestamp), όπως και το σύνολο των επιτυχών (S) ή ανεπιτυχών (F) αναρτήσεων και τη συνολική διάρκεια μιας εκτέλεσης σε δευτερόλεπτα. Σε περίπτωση αποτυχίας ανάρτησης, εκτός από τον τίτλο της ανακοίνωσης στο logfile καταγράφεται και ο κωδικός λάθους μαζί με την περιγραφή, έτσι ώστε να μπορεί κανείς να καταλήξει εύκολα σε συμπεράσματα, για το τι μπορεί να οδήγησε σε συνεχείς αποτυχίες για παράδειγμα.

# 7 Προστιθέμενη αξία και καινοτομία

Η υπηρεσία αυτή προσθέτει αξία στον ιστότοπο της Ύδρας, διότι παρέχει τη δυνατότητα στους χρήστες της να ενημερώνονται αυτόματα, μέσω του Twitter. Κάθε έξυπνο τηλέφωνο προσφέρει εφαρμογή Twitter, οπότε αν εκτελείται αυτή και ο χρήστης είναι συνδεδεμένος στο διαδίκτυο, λαμβάνει άμεσα ειδοποίηση πως έχει ανακοινωθεί κάτι καινούριο. Το ίδιο ισχύει και για κάποιον χρήστη υπολογιστή με λειτουργικό σύστημα Windows και εγκατεστημένο το chirpr. Έτσι ο επισκέπτης της Ύδρας φτάνει σε αυτήν με ένα κλικ ή ένα tap της οθόνης και, μάλιστα, μόνο όταν υπάρχει κάτι καινούριο να ελέγξει και μόνο αν αυτό είναι κάτι που όντως τον ενδιαφέρει.

Η καινοτομία της εφαρμογής διαφαίνεται στον σκοπό που επιτυγχάνει, δηλαδή την εκμαίευση πληροφορίας από μια ιστοσελίδα και την μεταφορά τμήματος αυτής σε κοινωνικό δίκτυο. Ενώ υπάρχουν πολλές εφαρμογές που αντλούν πληροφορία από RSS, blogs και μηνύματα ηλεκτρονικής αλληλογραφίας για παράδειγμα, δεν έχουμε εντοπίσει κάποια που να αντλεί από τα περιεχόμενα ιστοσελίδας. Βεβαίως αυτό μπορεί να ισχύει, διότι μια τέτοια προσπάθεια μειώνει τη δυνατότητα επαναχρησιμοποίησης του κώδικα. Ωστόσο ο τρόπος με τον οποίο υλοποιήσαμε την εφαρμογή, ακυρώνει σχεδόν το μειονέκτημα αυτό. Το μόνο που χρειάζεται για να αντληθεί πληροφορία από κάποια τρίτη ιστοσελίδα[10], είναι να εντοπίσει κανείς σε ποιο τμήμα της βρίσκεται αυτή και να ρυθμίσει ανάλογα τις αντίστοιχες παραμέτρους.

Επιπρόσθετα, η εφαρμογή αποτελεί λογισμικό ανοικτού κώδικα, που σημαίνει πως μπορεί κανείς να την επεκτείνει εφόσον το επιθυμεί. Η πιο σημαντική επέκτασή της ίσως είναι η προσθήκη νέων βιβλιοθηκών και αντίστοιχων κλήσεων για την αποστολή αυτόματων ενημερώσεων και σε άλλα κοινωνικά δίκτυα. Το επόμενο κεφάλαιο αφορά κάποιες από τις πιθανές επεκτάσεις που μπορεί να γίνουν στη συγκεκριμένη υπηρεσία.

[10] Επιβάλλεται βεβαίως η ιστοσελίδα να παρέχει valid HTML

# 8 Μελλοντικές επεκτάσεις της εφαρμογής και της υπηρεσίας

## 8.1 Επεκτάσεις ως προς την απόκτηση και επεξεργασία των δεδομένων

Μία από τις πιθανές επεκτάσεις θα μπορούσε να είναι η είσοδος δεδομένων από διαφορρετικές πηγές. Αυτή τη στιγμή η εφαρμογή αντλεί δεδομένα απο μία HTML σελίδα και λειτουργεί για κάθε σελίδα με τη συγκεκριμένη διαμόρφωση. Η εφαρμογή ήδη επεξεργάζεται αρχεία XML καθώς τα χρησιμοποιεί, έτσι λοιπόν θα μπορούσε να γίνει μια μικρή προσθήκη ώστε να διαβάζει δεδομένα κατευθείαν από XML αρχεία.

Θα μπορούσαμε επίσης να επεκτείνουμε την εφαρμογή έτσι ώστε να αντλεί δεδομένα από μια βάση δεδομένων, που θα περιλάμβανε βάσεις SQL ή NoSQL, και αρχεία Microsoft Access. Ανάλογα με τις απαιτήσεις, θα μπορούσαν να χρησιμοποιηθούν και αρχεία Microsoft Excel ή ακόμα και ένα απλό αρχείο κειμένου σαν είσοδος. Αυτή η υπηρεσία δε, ήταν αρχικά προγραμματισμένη να διαβάζει από έναν λογαριασμό email τα μηνύματα και λειτουργούσε κανονικά, αλλά αργότερα λόγω αλλαγής κατεύθυνσης η ιδέα δεν προχώρησε. Ένας ακόμη τρόπος εισόδου είναι η χρήση JSON αντικειμένων που αποστέλλονται από την πηγή. Αυτά μπορούμε στη συνέχεια να τα επεξεργαστούμε χρησιμοποιώντας JSONPath.

Ανάλογα με την είσοδο δεδομένων που καθορίζουμε, αλλάζει πιθανότατα και ο τρόπος επεξεργασίας των δεδομένων. Σε κάποιες περιπτώσεις μπορεί και να μην είναι απαραίτητο να γίνει επεξεργασία.

Στο παράδειγμα της εισόδου από ένα λογαριασμό ηλεκτρονικής αλληλογραφίας, υπάρχουν δύο τρόποι να διαβάσει κανείς τα περιεχόμενα ενός μηνύματος. Ο πρώτος είναι κάνοντας απλά ένα parse το κείμενο του μηνύματος, ενώ ο δεύτερος είναι η αντιμετώπιση του ως αντικειμένου. Αυτό εξαρτάται από το αν το μήνυμα είναι MIME (Multipurpose Internet Mail Extensions) ή non-MIME (απλό μήνυμα). Η MIME μορφή είναι ένα Internet standard το οποίο δημιουργήθηκε για να υποστηρίζει χαρακτήρες πέραν του ASCII, συνημμένα αρχεία που δεν είναι απλό κείμενο και σώματα μηνύματος με πολλαπλά μέρη (multipart). Άρα ακόμη και όταν η είσοδος είναι ίδιου τύπου, η επεξεργασία μπορεί να γίνεται διαφορετικά, ανάλογα με την περίπτωση.

Εάν υπάρχει η δυνατότητα επέμβασης στον κώδικα της πηγής (ή των πηγών), είναι δυνατή η χρήση διαφορετικών μοντέλων (π.χ. push ή κάποιον συνδυασμό push-pull). Σε τέτοια περίπτωση μπορεί για παράδειγμα να υλοποιηθούν ουρές στην πηγή, υπεύθυνες για την μεταφορά της πληροφορίας στις εφαρμογές πελάτες (όπως τη δική μας). Επίσης μια ουρά μπορεί να αποτελεί ενδιάμεση εφαρμογή που εκτελείται σε ανεξάρτητο σύστημα, δηλαδή να μην αποτελούν τμήμα ούτε της πηγής, ούτε της εφαρμογής.

### *8.1.1 Διάσπαση εφαρμογής και χρήση Queues*

Η υπηρεσία μπορεί να διαχωριστεί σε δύο ξεχωριστές εφαρμογές, από τις οποίες η μια κάνει pull ανά τακτά χρονικά διαστήματα από την Ύδρα και χρησιμοποιώντας ένα message queue αποστέλει (push) πληροφορία στη δεύτερη, η οποία είναι επιφορτισμένη με τη δημιουργία αναρτήσεων στο Twitter. Αυτό έχει το πλεονέκτημα ότι η μεν πρώτη μπορεί να αντλεί δεδομένα από περισσότερες πηγές, ενώ η δεύτερη μπορεί να δέχεται δεδομένα από τρίτα message queues (ουρές). Επίσης πολλές εφαρμογές της δεύτερης μορφής μπορούν να παρακολουθούν την ουρά. Άρα μπορούμε σε ένα λογαριασμό του Twitter να αναρτούμε tweets που προέρχονται από διάφορες ουρές ή από μια ουρά η οποία αντλεί δεδομένα από περισσότερες πηγές, που μπορεί να είναι και διαφορετικών τύπων.

Οι ουρές χρησιμοποιούνται για να επικοινωνούν αξιόπιστα μεταξύ τους δύο διανεμημένες διαδικασίες. Γενικά, μια ουρά δέχεται δεδομένα από κάποια πηγή (ή τα αίτειται) και τα προωθεί στη συνέχεια σε όσους είναι συνδεδεμένοι στην άλλη «άκρη».

Οι εφαρμογές Message Oriented Middleware (MOM), όπως οι ουρές, είναι χτισμένες για να χειριστούν αυτές τις περιπτώσεις χρήσης. Υποθέτουν πως τα μηνύματα σε ένα φυσιολογικό σύστημα θα διαγραφούν πολύ γρήγορα μετά την λήψη τους και έτσι μπορούν να κάνουν βελτιστοποιήσεις για να αποφευχθεί η επιβάρυνση. Η εφαρμογή μπορεί επίσης να ωθήσει τα μηνύματα προς τους καταναλωτές αντί να ζητάν οι καταναλωτές τα μηνύματα από την εφαρμογή. Αυτό μειώνει περαιτέρω την καθυστέρηση που περιέχεται στην επεξεργασία των νέων μηνυμάτων που στέλνονται στο σύστημα.

Ειδικότερα, κάποιες ουρές[32] μπορεί να χρησιμοποιούνε μόνο τη μνήμη ενός υπολογιστή για την προσωρινή αποθήκευση δεδομένων (caching), ενώ άλλες μπορεί να χρησιμοποιούν και αρχεία στο δίσκο ή μια βάση.

Μια άλλη σημαντική διαφοροποίηση μπορεί να είναι η αντίδραση μιας ουράς, σε περίπτωση που κάποιος παραλήπτης δεν είναι συνδεδεμένος. Μερικές ουρές καταστρέφουν τα μηνύματα που πρέπει να μεταφερθούν, αν δε καταφέρουν να τα παραδώσουν άμεσα. Σε τέτοια περίπτωση, μπορεί να υπάρξει απώλεια πληροφορίας. Σε εφαρμογές που μεταφέρουν κρίσιμες πληροφορίες, αυτό είναι σίγουρα ανασταλτικός παράγοντας για τη χρήση μιας συγκεκριμένης υλοποίησης.

Επίσης, τα μοντέλα που χρησιμοποιούν οι ουρές μπορεί να διαφέρουν. Κάποιες ουρές είναι event-driven, άλλες κάνουν χρήση του publish-subscribe pattern. Παρακάτω παρουσιάζουμε συνοπτικά τις ιδιότητες τριών διαδεδομένων υλοποιήσεων, την Beanstalkd, την ActiveMQ και την RabbitMQ.

#### *8.1.1.1 Beanstalkd*

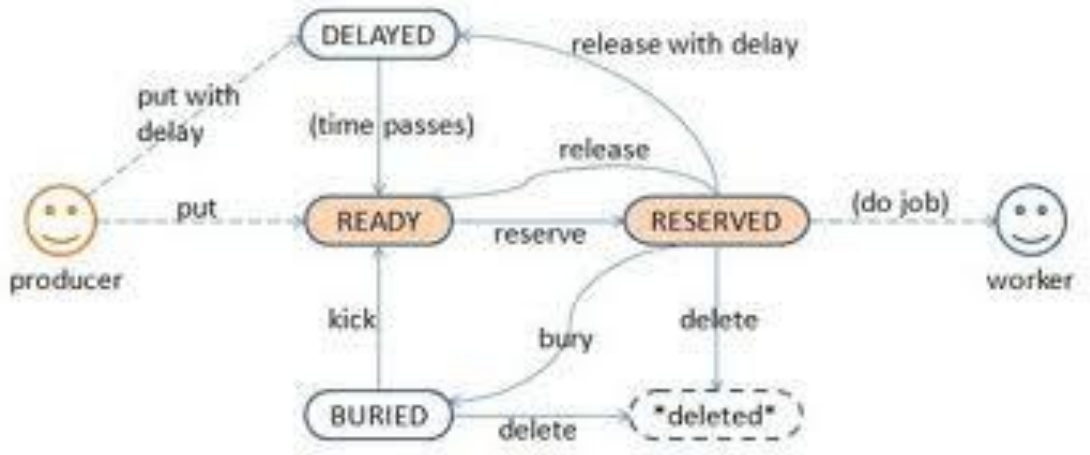
Η Beanstalkd είναι μια ασύγχρονη ουρά που υποστηρίζει τη διανομή εργασιών σε ένα κλειστό δίκτυο, προκειμένου να τους δοθεί προτεραιότητα

και να υλοποιηθούν. Προσφέρει τη δυνατότητα οργάνωσης των εργασιών σε “tubes” (όπου κάθε “tube” αντιστοιχεί σε έναν τύπο εργασίας), τα οποία δημιουργούνται από τον παραγωγό (producer). Ένας καταναλωτής (consumer) μπορεί να συνδέεται σε όσα tubes επιθυμεί, καθώς δεν υπάρχει περιορισμός στον αριθμό συνδέσεων στην ουρά.

Όταν μια εργασία ολοκληρωθεί επιτυχώς, ο καταναλωτής μπορεί να διαγράψει την εργασία από το “tube”. Σε περίπτωση αποτυχίας, ο καταναλωτής μπορεί να “θάψει την εργασία” (bury). Αυτό σημαίνει πως η εργασία δεν θα διαγραφεί αλλά θα είναι διαθέσιμη αργότερα για περαιτέρω έλεγχο.

Μερικά απο τα πλεονεκτήματα της Beanstalkd είναι τα εξής:

- Σχεδιασμένη από το μηδέν για να είναι μια ουρά εργασίας
- Πολύ γρήγορη
- Καταναλώνει πολύ λίγη CPU
- Αλλάζει εύκολα ρυθμίσεις
- Διαθέτει εξελιγμένη ροή προγράμματος ικανή να καλύψει όλες τις ανάγκες




**Εικόνα 8.1 Τρόπος λειτουργίας Beanstalkd**

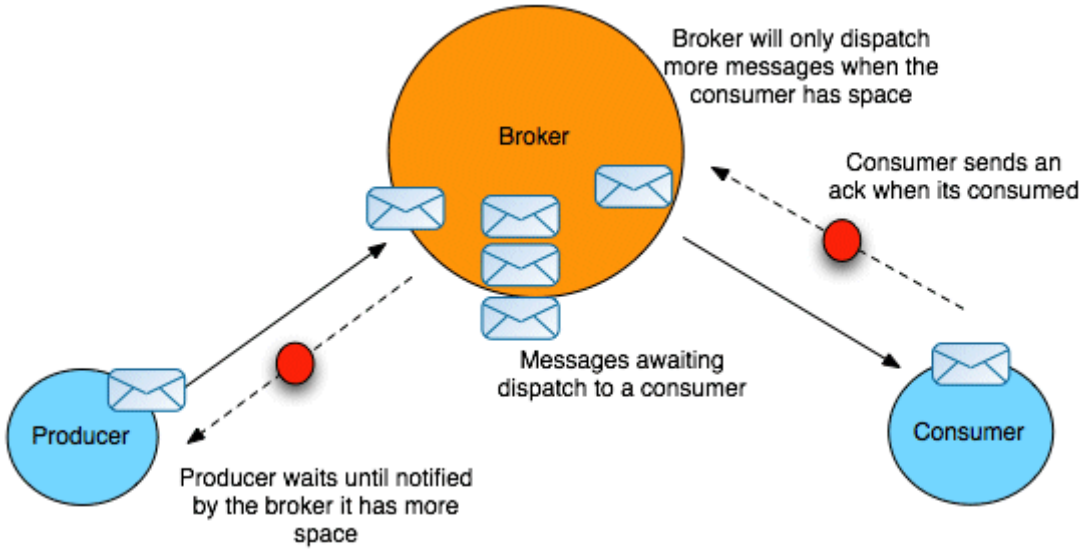
### *8.1.1.2 ActiveMQ*

Η ActiveMQ είναι ένα open-source λογισμικό μηνυμάτων που μπορεί να χρησιμοποιηθεί ως βάση για κατανεμημένες εφαρμογές που βασίζονται στα μηνύματα.

Κάποια από τα πλεονεκτήματα της ActiveMQ είναι τα εξής:

- Επιτρέπει τη συνεργασία εφαρμογών, που έχουν δημιουργηθεί σε διαφορετικές γλώσσες και σε διαφορετικά λειτουργικά συστήματα
- Παρέχει αξιόπιστη επικοινωνία. Οι παραγωγοί/καταναλωτές των μηνυμάτων δε χρειάζεται να είναι διαθέσιμοι την ίδια στιγμή

- Κλιμάκωση - Μπορεί να επεκταθεί οριζόντια με την προσθήκη περισσότερων υπηρεσιών που μπορούν να χειριστούν όλο και περισσότερα μηνύματα
- Υποστηρίζει ασύγχρονη επικοινωνία
- Μειωμένη ζεύξη - Μια υπηρεσία μπορεί να αλλάξει τα χαρακτηριστικά της, συμπεριλαμβανομένης της θέσης της, του πρωτοκόλλου της και της διαθεσιμότητας της, χωρίς να επηρεάζει τον πελάτη.




**Εικόνα 8.2 Τρόπος λειτουργίας ActiveMQ**

### *8.1.1.3 RabbitMQ*

Η RabbitMQ είναι ένας μεσάζων για την ανταλλαγή μηνυμάτων και επιτρέπει στις εφαρμογές τη σύνδεση σε μια κοινή πλατφόρμα. Οι εφαρμογές μπορούν να συνδεθούν μεταξύ τους, ως τμήματα μιας μεγαλύτερης εφαρμογής, ή να συνδεθούν σε συσκευές και δεδομένα χρηστών. Η διαδικασία είναι ασύγχρονη και η αποσύνδεση των εφαρμογών είναι ανεξάρτητη από την αποστολή και την λήψη των δεδομένων.

Κάποια από τα πλεονεκτήματα είναι τα εξής:

- Αξιοπιστία – Η RabbitMQ προσφέρει μια πληθώρα χαρακτηριστικών συμπεριλαμβανομένης της εμμονής, της αναφοράς παράδοσης, της επιβεβαίωσης έκδοσης και της υψηλής διαθεσιμότητας.
- Clustering - Αρκετοί servers RabbitMQ σε ένα τοπικό δίκτυο μπορούν να ομαδοποιηθούν μαζί, λειτουργώντας ως ένας ενιαίος μεσίτης.
- Διαθεσιμότητα ουρών - Διασφαλίζεται ότι, ακόμη και σε περίπτωση αστοχίας υλικού, τα μηνύματά είναι ασφαλη.

- Πολλαπλά πρωτόκολλα – Η RabbitMQ υποστηρίζει την ανταλλαγή μηνυμάτων σε μια ποικιλία από πρωτόκολλα ανταλλαγής μηνυμάτων.
- UI διαχείρισης – Ένα εύκολο στη χρήση περιβάλλον διαχείρισης επιτρέπει την παρακολούθηση και των έλεγχο κάθε μηνύματος.
- Tracing - Αν το σύστημα ανταλλαγής μηνυμάτων αντιμετώπισε κάποιο σφάλμα, προσφέρεται εντοπισμός μηνυμάτων προκειμένου να διορθωθεί το λάθος.
- Επεκτασιμότητα-Υποστηρίζονται plugin τα οποία μπορούν να δημιουργήσουν οι χρήστες.

## 8.2 Επεκτάσεις ως προς την έξοδο των δεδομένων

Συγκεκριμένα για το Twitter δεν μπορεί να γίνει κάποια επέκταση σε σχέση με το τι δεδομένα εξάγει η εφαρμογή, πέραν ίσως της αποστολής σε περισσότερους λογαριασμούς. Δημοσίευση των δεδομένων θα μπορούσαμε ωστόσο εύκολα να κάνουμε σε μια σειρά social media, όπως το Facebook, το Google Plus και το LinkedIn, καθώς και αυτά παρέχουν διεπαφές προγραμματισμού, που δε διαφέρουν κατά πολύ από αυτές του Twitter. Επίσης μπορεί να δημιουργηθεί μια προσθήκη για την ανάρτηση των δεδομένων σε blog, καθώς όλο και περισσότερος κόσμος έχει αρχίσει να τα χρησιμοποιεί σε καθημερινή βάση.

# 9 Ενδεικτική βιβλιογραφία


- Αλεξόπουλος, Αριστείδης, Λαγογιάννης, Γεώργιος (2011), Τηλεπικοινωνίες και δίκτυα υπολογιστών ; Ασύρματα συστήματα, τηλεφωνία , δίκτυα, Παπάγου, Ελλάδα, Εκδόσεις Γιαλός.
- Παύλος, Παπαπαύλου (2013), Μάθετε το twitter, Κριονέρι, Ελλάδα, Εκδόσεις Digerati A.E.
- Στυλιανός, Ανέστης (2013), Η τεχνολογία XML, Καλλιθέα, Ελλάδα.
- Σιδέρη, Μελπομένη (2011), Το Βιβλίο του Twitter ; Ένας οδηγός για "αθώους" χρήστες, Αθήνα, Ελλάδα, Εκδόσεις Κλειδάριθμος
- Handlock, Kris (2007), Ajax ανάπτυξη web εφαρμογών = Ajax for web application developers, Μετάφραση: Χρυσούλα Κουτρούμπα, Αθήνα, Ελλάδα, Εκδόσεις Μ. Γκιούρδας.
- Holzner, Steven (2009), Οδηγός της Ajax = Ajax: A beginner's guide, Μετάφραση: Γιάννης Σαμαράς, Αθήνα, Ελλάδα, Εκδόσεις Μ. Γκιούρδας.
- Holzner, Steven (2009), Οδηγός της XML = XML: A beginner's guide : Go beyond the basics with Ajax, XHTML, XPath 2.0, XSLR 2.0 and XQuery ; Προχωρήστε πέρα από τα βασικά με Ajax, XHTML, XPath 2.0, XSLT 2.0 & XQuery, Μετάφραση: Χρυσούλα Κουτρούμπα, Αθήνα, Ελλάδα, Εκδόσεις Μ. Γκιούρδας.
- McMahon, Richard Alan (2004), Εισαγωγή στα δίκτυα υπολογιστών = Introduction to networking, Μετάφραση: Γιάννης Σαμαράς, Αθήνα, Ελλάδα, Εκδόσεις Μ. Γκιούρδας.
- Meloni, Julie (2004), Μάθετε PHP, MySQL και Apache = Teach yourself PHP, MySQL and Apache All In One : All in one ; Ολα σε ένα, Μετάφραση: Ελένη Γκαγκάτσιου, Αθήνα, Ελλάδα, Εκδόσεις Μ. Γκιούρδας.
- Mercer, Dave (2004), Οδηγός της XML = XML a Beginner's' Guide, Μετάφραση: Μαίρη Γκλαβά – Αθήνα, Ελλάδα, Εκδόσεις Μ. Γκιούρδας.
- Sterling, Hughes (2002), PHP Οδηγός Προγραμματισμού = PHP Developer's Cookbook, Μετάφραση: Γιάννης Σαμαράς, Αθήνα, Ελλάδα, Εκδόσεις Μ. Γκιούρδας.
- Thomson, Laura, Welling, Luke (2011), Ανάπτυξη Web Εφαρμογών με PHP και MySQL, 4η Έκδοση = PHP and MySQL Web Development (4th Edition), Μετάφραση: Μαίρη Γκλαβά, Αθήνα, Ελλάδα, Εκδόσεις Μ. Γκιούρδας.

# 10 Πηγές


1. Twitter – Wikipedia - http://en.wikipedia.org/wiki/Twitter
2. Changes coming in Version 1.1 of the Twitter API - https://dev.twitter.com/blog/changes-coming-to-twitter-api
3. Well-formed element – Wikipedia - http://en.wikipedia.org/wiki/Well-formed_element
4. SimpleXMLElement::asXML - http://www.php.net/manual/en/simplexmlelement.asxml.php
5. Push Vs. Pull model : Communication & Data flow models - http://nadeeshanihewage.blogspot.gr/2012/04/push-vs-pull-model-communication-data.html
6. Twitter Developer - https://dev.twitter.com/
7. PHP 5 Introduction - http://www.php.net/manual/en/intro-whatis.php
8. What is PHP? - http://www.w3schools.com/php/php_intro.asp
9. PHP – Wikipdia - http://en.wikipedia.org/wiki/PHP
10. XAMPP Apache + MySQL + PHP + Perl - https://www.apachefriends.org/index.html
11. XAMPP – Wikipedia - http://en.wikipedia.org/wiki/XAMPP
12. XML – Wikipedia - http://en.wikipedia.org/wiki/XML
13. Extensible Markup Language (XML) 1.0 - http://www.it.uom.gr/project/xml/Home%20Page.htm
14. Extensible Markup Language (XML) – W3C - http://www.w3.org/XML/
15. Document_Object_Model – Wikipedia - http://en.wikipedia.org/wiki/Document_Object_Model
16. Επέκταση Της Γλώσσας Ερωτήσεων Xquery Για Υποστήριξη Ερωτήσεων - http://artemis-new.cslab.ece.ntua.gr:8080/jspui/handle/123456789/4135?mode=full

17. XQuery - http://dlib.ionio.gr/ctheses/0506tab575a/Louvari_%20Xquery.doc
18. JSON – Wikipedia - http://en.wikipedia.org/wiki/JSON
19. JSON: What It Is, How It Works, & How to Use It - http://www.copterlabs.com/blog/json-what-it-is-how-it-works-how-to-use-it/
20. Εισαγωγή στο JSON - http://www.json.org/json-el.html
21. Curl – Wikipedia - http://en.wikipedia.org/wiki/Curl_(programming_language)
22. Adaptive Push-Pull: Disseminating Dynamic Web Data - http://www-ccs.cs.umass.edu/~krithi/web/WWW10/www10/
23. Twitter Migration - http://blog.evanweaver.com/2009/09/24/ree/
24. How Twitter tweets your tweets with open source - http://www.zdnet.com/how-twitter-tweets-your-tweets-with-open-source-7000003526/
25. Twitter Counter Official Page - http://twittercounter.com/pages/twittermail
26. Twuffer Official Page - http://twuffer.com/
27. TwitterFeed Official Page - http://twitterfeed.com/
28. Buffer Official Page - https://bufferapp.com/
29. TweetDeck Official Page - https://about.twitter.com/products/tweetdeck
30. Sprout Social Official Page - http://sproutsocial.com/
31. The Streaming APIs – Twitter - https://dev.twitter.com/docs/api/streaming
32. Message queue – Standars and Protocols – Wikipedia - http://en.wikipedia.org/wiki/Message_queue#Standards_and_protocols
33. Πρωτόκολλο HTTP - http://netlab.teiath.gr/JSPWiki/attach/NetLabEx/Exercise6.pdf

34. HYPERTEXT TRANSFER PROTOCOL - http://alexandra.di.uoa.gr/mmtech/msTech/1_HTML/PDFs/chp2.pdf
35. Θέματα Ασφάλειας Δικτύων Για Μηχανικούς Πληροφορικής - http://bit.ly/1sk8teH
36. Hypertext Transfer Protocol – Wikipedia - http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
37. Πρωτόκολλο SSL (Secure Sockets Layer) - Βικιπεδία - http://el.wikipedia.org/wiki/SSL
38. How does OAuth 2 work? - http://stackoverflow.com/questions/4727226/on-a-high-level-how-does-oauth-2-work
39. OAuth for Dummies - http://marktrapp.com/blog/2009/09/17/oauth-dummies/
40. OAuth 2 Simplified - http://aaronparecki.com/articles/2012/07/29/1/oauth2-simplified
41. Introduction to OAuth (in Plain English) - http://blog.varonis.com/introduction-to-oauth/
42. How Does Twitter's OAuth System Work? - http://stackoverflow.com/questions/1390881/how-does-twitters-oauth-system-work
43. Chirpr - Windows Gadget for Twitter - http://chirpr.codeplex.com/

# 11 Παράθεση κώδικα εφαρμογής

## 11.1 Η κλάση εργάτης

```
<?php

/*
 * Constants start
 */

//SOURCE DEFINITION
define("SOURCE",
"https://hydra.it.teithe.gr/s/index.php?nocache=1402345472&m=itdep-bbstud");
define("BASE", "http://hydra.it.teithe.gr");

//define("SOURCE", "http://www.thepressproject.gr/");
//define("BASE", "http://www.thepressproject.gr/");
//NEWS DEFINITION
define("XPATH", '//table[@class="vehi-list"]/tr[@class="data"]');
define("NEWS", serialize(array("children", "td")));
define("AUTHOR", serialize(array("0", "text")));
define("DATE", serialize(array("1", "text")));
define("TITLE", serialize(array("2", "text")));
define("LINK", serialize(array("4", "children", "a", "0", "attributes", "href")));

//define("XPATH", '//div[@class="article"]');
//define("NEWS", serialize(array("children", "title","children","h3","children")));
//define("AUTHOR", serialize(array("0", "text")));
//define("DATE", serialize(array("1", "text")));
//define("TITLE", serialize(array("0", "text")));
//define("LINK", serialize(array("4", "children", "a", "0", "attributes", "href")));
//TWITTER OAUTH DEFINITION
define("ACCESS_TOKEN", 'xxxxxxxxxxxxx');
define("ACCESS_TOKEN_SECRET", 'xxxxxxxxxxxxx');
define("CONSUMER_KEY", 'xxxxxxxxxxxxx');
define("CONSUMER_SECRET", 'xxxxxxxxxxxxx');

/*
 * Constants end
 */

//add required helper class
require_once 'Helper.php';

//save current timestamp, in order to calculate script run time
$startTime = time();
//save contents of logfile
$log = file_get_contents('hydrabay.log');
//start the session and read last tweeted message if it exists
session_start();
if (isset($_SESSION['lastTweet'])) {
```

```
    $lastTweet = $_SESSION['lastTweet'];
} else {
    $lastTweet = "";
}

/*
 * This variable is used only for development / staging
 * Comment it out if script is running on production
 */
$lastTweet = "";

//define charset UTF-8 for use with greek characters
header('Content-Type: text/html;charset=utf-8');
mb_internal_encoding('UTF-8');

//initialize the object used for communicating with Twitter
$twitter = Helper::initTwitter(ACCESS_TOKEN, ACCESS_TOKEN_SECRET,
CONSUMER_KEY, CONSUMER_SECRET);
//define end point of Twitter API calls
$apiURL = 'https://api.twitter.com/1.1/statuses/update.json';
//define API request method
$requestMethod = 'POST';

//get contents from source as HTML and save in XML
//-- this will not work as intended if website HTML is not valid
$contents = new DOMDocument;
$contents->loadHTML(file_get_contents(SOURCE));
file_put_contents('new.xml', $contents->saveXml());

//load contents of XML file in simpleXMLElement PHP Object.
$simplexml = simplexml_load_file('new.xml', 'SimpleXMLElement',
LIBXML_NOCDATA);

//filter information aquired using xpath
$xpath = $simplexml->xpath(XPATH);

//transform information to PHP array and reverse it
$newsArray = array_reverse(Helper::transform2Array($xpath));

//init counters of successful and unsuccessful attempts to tweet
$success = 0;
$fail = 0;

foreach ($newsArray as $newPost) {

    //get current announcement
    $newDiv = Helper::getValue(unserialize(NEWS), $newPost);

    //get author
    if (null != Helper::getValue(unserialize(AUTHOR), $newDiv)) {
        $author = trim(Helper::getValue(unserialize(AUTHOR), $newDiv));
    }
```

```
//get date
if (null != Helper::getValue(unserialize(DATE), $newDiv)) {
    $date = trim(Helper::getValue(unserialize(DATE), $newDiv));
}

//get title
if (null != Helper::getValue(unserialize(TITLE), $newDiv)) {
    $title = trim(Helper::getValue(unserialize(TITLE), $newDiv));
}

//update variable value used for checking and abort if
//last tweet is current announcement
$tweetCheck = $title . $date . $author;
if ($lastTweet == $tweetCheck) {
    echo "<hr>Reached last tweeted announcement while parsing news.<br>";
    die("Exiting.");
}

//if sublink exists, save it
if (null != Helper::getValue(unserialize(LINK), $newDiv)) {
    $subLink = Helper::getValue(unserialize(LINK), $newDiv);
}

//prepend baseURL to sublink
$link = Helper::get_tiny_url(BASE . $subLink);

//echo current announcement
echo "<hr>" . $title . " " . $date . " " . $author . " " . $link . "<br><br>";

//prepare title to be tweeted
//if length is more than "x" characters, reduce the title's length
//check the Helper class if changing this value is required
$titleTemp = Helper::reduceLength(trim($title));
if ($titleTemp != $title) {
    echo "Title to be tweeted: " . $titleTemp . "<br>";
}
$titleTweet = html_entity_decode($titleTemp);

//define status to be tweeted
$postfields = array('status' => $titleTweet . ": " . $link);

echo "<br><strong>POST STATUS: " . $postfields['status'] . "</strong><br><hr>";

//attempt to create new tweet and report / log result
try {
    echo "Tweeting <br>";
    /* echo */ $twitter->buildOauth($apiURL, $requestMethod)
            ->setPostfields($postfields)
            ->performRequest(true);
} catch (Exception $e) {
    echo "<br><strong>Failed tweet: " . $e . "</strong><br>";
```

```
        $fail++;
        $log .= time() . ": FAIL - " . $title . "\n-------" . $e . "\n";
        continue;
    }
    $success++;
    $log .= time() . ": success - " . $title . "\n";
    $_SESSION['lastTweet'] = $tweetCheck;
}
//final report
if ($success > 0) {
    echo "<hr>Succeeded posting " . $success . " tweets.";
}
if ($fail > 0) {
    echo "<hr>Failed tweeting " . $fail . " tweets.";
}
//save extended report to logfile
$execTime = time() - $startTime;
$log .= "########\nSummary:\nS: " . $success . "\nF: " . $fail . "\nScript Execution
Time: " . $execTime . " seconds\n########\n";
file_put_contents('hydrabay.log', $log);

//exit
die("<hr>Exiting.");
```

## 11.2 Η βοηθητική κλάση

```
<?php

require_once 'TwitterApiExchange.php';

class Helper {
    /*
     * This function receives two arrays as parameters
     * The first parameter is an array containing index paths,
     * used to return a value from a specific subindex, while the
     * second is the array which contains the value
     */

    public static function getValue($indexes, $arrayToAccess) {
        if (count($indexes) > 1) {
            return Helper::getValue(array_slice($indexes, 1),
$arrayToAccess[$indexes[0]]);
        } else {
            return $arrayToAccess[$indexes[0]];
        }
    }

    /*
     * URL shortener API call to tinyurl.com's API
     * parameter is URL to be shortened
     * returns shortened URL
     */

    public static function get_tiny_url($url) {
        $ch = curl_init();
        $timeout = 5;
        curl_setopt($ch, CURLOPT_URL, 'http://tinyurl.com/api-create.php?url=' .
$url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
        curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, $timeout);
        $data = curl_exec($ch);
        curl_close($ch);
        return $data;
    }

    /*
     * Creates Object to be used for communication with Twitter API
     */

    public static function initTwitter($aTok, $aTokSec, $cKey, $cSec) {
        $twitterConnectSettings = array(
            'oauth_access_token' => $aTok,
            'oauth_access_token_secret' => $aTokSec,
            'consumer_key' => $cKey,
            'consumer_secret' => $cSec
        );
```

```
        return new TwitterAPIExchange($twitterConnectSettings);
    }

    /*
     * Function receives two parameters, a string and an integer (length)
     * returns new substring of string, removing 1 word from the end until
     * length is less that supplied integer
     */

    public static function reduceLength($str, $length = 100) {
        echo strlen($str) . ": " . $str . "<br>";
        if (strlen(html_entity_decode($str)) > $length) {
            $lastSpacePosition = strrpos($str, ' ');
            $str = substr($str, 0, $lastSpacePosition);
            if (strlen($str) > $length) {
                Helper::reduceLength($str);
            }
        }
        return $str;
    }

    /*
     * SimpleXMLElement to Array functions
     */

    public static function transform2Array($simplexml) {
        $i = 0;
        $array = array();
        foreach ($simplexml as $new) {
            $array[$i++] = Helper::xmlObjToArr($new);
        }
        return $array;
    }

    public static function xmlObjToArr($obj) {
        $namespace = $obj->getDocNamespaces(true);
        $namespace[NULL] = NULL;

        $children = array();
        $attributes = array();
        $name = strtolower((string) $obj->getName());

        $text = trim((string) $obj);
        if (strlen($text) <= 0) {
            $text = NULL;
        }

        // get info for all namespaces
        if (is_object($obj)) {
            foreach ($namespace as $ns => $nsUrl) {
                // atributes
                $objAttributes = $obj->attributes($ns, true);
```

```
            foreach ($objAttributes as $attributeName => $attributeValue) {
                $attribName = strtolower(trim((string) $attributeName));
                $attribVal = trim((string) $attributeValue);
                if (!empty($ns)) {
                    $attribName = $ns . ':' . $attribName;
                }
                $attributes[$attribName] = $attribVal;
            }

            // children
            $objChildren = $obj->children($ns, true);
            foreach ($objChildren as $childName => $child) {
                $childName = strtolower((string) $childName);
                if (!empty($ns)) {
                    $childName = $ns . ':' . $childName;
                }
                $children[$childName][] = Helper::xmlObjToArr($child);
            }
        }
    }

    return array(
        'name' => $name,
        'text' => $text,
        'attributes' => $attributes,
        'children' => $children
    );
  }

}
```

## 11.3 Η βιβλιοθήκη επικοινωνίας με το Twitter

```
<?php

/**
 * Twitter-API-PHP : Simple PHP wrapper for the v1.1 API
 *
 * PHP version 5.3.10
 *
 * @category Awesomeness
 * @package Twitter-API-PHP
 * @author James Mallison <me@j7mbo.co.uk>
 * @license MIT License
 * @link http://github.com/j7mbo/twitter-api-php
 */
class TwitterAPIExchange {

    private $oauth_access_token;
    private $oauth_access_token_secret;
    private $consumer_key;
    private $consumer_secret;
    private $postfields;
    private $getfield;
    protected $oauth;
    public $url;

    /**
     * Create the API access object. Requires an array of settings::
     * oauth access token, oauth access token secret, consumer key, consumer secret
     * These are all available by creating your own application on dev.twitter.com
     * Requires the cURL library
     *
     * @param array $settings
     */
    public function __construct(array $settings) {
        if (!in_array('curl', get_loaded_extensions())) {
            throw new Exception('You need to install cURL, see:
http://curl.haxx.se/docs/install.html');
        }

        if (!isset($settings['oauth_access_token']) ||
!isset($settings['oauth_access_token_secret']) || !isset($settings['consumer_key']) ||
!isset($settings['consumer_secret'])) {
            throw new Exception('Make sure you are passing in the correct parameters');
        }

        $this->oauth_access_token = $settings['oauth_access_token'];
        $this->oauth_access_token_secret = $settings['oauth_access_token_secret'];
        $this->consumer_key = $settings['consumer_key'];
        $this->consumer_secret = $settings['consumer_secret'];
    }
```

```
/**
 * Set postfields array, example: array('screen_name' => 'J7mbo')
 *
 * @param array $array Array of parameters to send to API
 *
 * @return TwitterAPIExchange Instance of self for method chaining
 */
public function setPostfields(array $array) {
    if (!is_null($this->getGetfield())) {
        throw new Exception('You can only choose get OR post fields.');
    }

    if (isset($array['status']) && substr($array['status'], 0, 1) === '@') {
        $array['status'] = sprintf("\0%s", $array['status']);
    }

    $this->postfields = $array;

    return $this;
}

/**
 * Set getfield string, example: '?screen_name=J7mbo'
 *
 * @param string $string Get key and value pairs as string
 *
 * @return \TwitterAPIExchange Instance of self for method chaining
 */
public function setGetfield($string) {
    if (!is_null($this->getPostfields())) {
        throw new Exception('You can only choose get OR post fields.');
    }

    $search = array('#', ',', '+', ':');
    $replace = array('%23', '%2C', '%2B', '%3A');
    $string = str_replace($search, $replace, $string);

    $this->getfield = $string;

    return $this;
}

/**
 * Get getfield string (simple getter)
 *
 * @return string $this->getfields
 */
public function getGetfield() {
    return $this->getfield;
}

/**
```

```
    * Get postfields array (simple getter)
    *
    * @return array $this->postfields
    */
   public function getPostfields() {
      return $this->postfields;
   }

   /**
    * Build the Oauth object using params set in construct and additionals
    * passed to this method. For v1.1, see: https://dev.twitter.com/docs/api/1.1
    *
    * @param string $url The API url to use. Example:
https://api.twitter.com/1.1/search/tweets.json
    * @param string $requestMethod Either POST or GET
    * @return \TwitterAPIExchange Instance of self for method chaining
    */
   public function buildOauth($url, $requestMethod) {
      if (!in_array(strtolower($requestMethod), array('post', 'get'))) {
         throw new Exception('Request method must be either POST or GET');
      }

      $consumer_key = $this->consumer_key;
      $consumer_secret = $this->consumer_secret;
      $oauth_access_token = $this->oauth_access_token;
      $oauth_access_token_secret = $this->oauth_access_token_secret;

      $oauth = array(
         'oauth_consumer_key' => $consumer_key,
         'oauth_nonce' => time(),
         'oauth_signature_method' => 'HMAC-SHA1',
         'oauth_token' => $oauth_access_token,
         'oauth_timestamp' => time(),
         'oauth_version' => '1.0'
      );

      $getfield = $this->getGetfield();

      if (!is_null($getfield)) {
         $getfields = str_replace('?', '', explode('&', $getfield));
         foreach ($getfields as $g) {
            $split = explode('=', $g);
            $oauth[$split[0]] = $split[1];
         }
      }

      $base_info = $this->buildBaseString($url, $requestMethod, $oauth);
      $composite_key = rawurlencode($consumer_secret) . '&' .
rawurlencode($oauth_access_token_secret);
      $oauth_signature = base64_encode(hash_hmac('sha1', $base_info,
$composite_key, true));
      $oauth['oauth_signature'] = $oauth_signature;
```

```
        $this->url = $url;
        $this->oauth = $oauth;

        return $this;
    }

    /**
     * Perform the actual data retrieval from the API
     *
     * @param boolean $return If true, returns data.
     *
     * @return string json If $return param is true, returns json data.
     */
    public function performRequest($return = true) {
        if (!is_bool($return)) {
            throw new Exception('performRequest parameter must be true or false');
        }

        $header = array($this->buildAuthorizationHeader($this->oauth), 'Expect:');

        $getfield = $this->getGetfield();
        $postfields = $this->getPostfields();

        $options = array(
            CURLOPT_HTTPHEADER => $header,
            CURLOPT_HEADER => false,
            CURLOPT_URL => $this->url,
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_TIMEOUT => 10,
            CURLINFO_HEADER_OUT => true,
            CURLOPT_SSL_VERIFYPEER => true,
            CURLOPT_SSL_VERIFYHOST => 2,
            CURLOPT_CAINFO => getcwd() . "/CAcert.crt"
        );

        if (!is_null($postfields)) {
            $options[CURLOPT_POSTFIELDS] = $postfields;
        } else {
            if ($getfield !== '') {
                $options[CURLOPT_URL] .= $getfield;
            }
        }

        $feed = curl_init();
        curl_setopt_array($feed, $options);
        $json = curl_exec($feed);
        curl_close($feed);

        if ($return) {
            return $json;
        }
```

```
    }

    /**
     * Private method to generate the base string used by cURL
     *
     * @param string $baseURI
     * @param string $method
     * @param array $params
     *
     * @return string Built base string
     */
    private function buildBaseString($baseURI, $method, $params) {
        $return = array();
        ksort($params);

        foreach ($params as $key => $value) {
            $return[] = "$key=" . $value;
        }

        return $method . "&" . rawurlencode($baseURI) . '&' .
rawurlencode(implode('&', $return));
    }

    /**
     * Private method to generate authorization header used by cURL
     *
     * @param array $oauth Array of oauth data generated by buildOauth()
     *
     * @return string $return Header used by cURL for request
     */
    private function buildAuthorizationHeader($oauth) {
        $return = 'Authorization: OAuth ';
        $values = array();

        foreach ($oauth as $key => $value) {
            $values[] = "$key=\"" . rawurlencode($value) . "\"";
        }

        $return .= implode(', ', $values);
        return $return;
    }

}
```