Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάλυση, σχεδιασμός και υλοποίηση συνεργατικής εφαρμογής για διαδίκτυο πραγματικού χρόνου (Analysis, Design and implementation of collaborative application for real time web)**
Ονοματεπώνυμο Φοιτητή	**Παπαθεοδώρου Αντώνιος - Νικόλαος**
Πατρώνυμο	**Χρυσόστομος**
Αριθμός Μητρώου	**ΜΠΠΛ/ 10034**
Επιβλέπων	**Χρήστος Δουληγέρης, Καθηγητής**
Συνεπιβλέπων	**ΔΡ. Βασίλειος Μενεκλής**

Ημερομηνία Παράδοσης **23/07/2013**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Δουληγέρης Χρήστος Καθηγητής	Κωνσταντόπουλος Χαράλαμπος Επίκουρος Καθηγητής	Πικράκης Άγγελος Λέκτορας

Περίληψη

Σήμερα, το διαδίκτυο, είναι ένας δυναμικός ιστός από χρήστες, εφαρμογές και συσκευές - μία συνεχώς εξελισσόμενη πλατφόρμα, στην οποία οι χρήστες διεξάγουν συνεχώς ένα αυξανόμενο μερίδιο της καθημερινής τους ζωής.

Το WebSocket της HTML5, αντιπροσωπεύει την πιο σημαντική αναβάθμιση στην ιστορία του παγκόσμιου ιστού. Εκμεταλλευόμενοι αυτό το νέο πρότυπο μαζί με ένα σύνολο από συστήματα και πρότυπα σχεδίασης (design patterns) λογισμικού, καλά προσαρμοσμένα στην δυναμική ροή δεδομένων, στην παρούσα διπλωματική εργασία επιχειρείται να παρουσιαστεί πως ο τρόπος με τον οποίο μπορούν να υλοποιηθούν εφαρμογές διαδικτύου που εντυπωσιάζουν τους χρήστες με τις δυνατότητες πραγματικού χρόνου που παρέχουν.

Για αυτό το σκοπό αναπτύχθηκε μία πρότυπη διαδικτυακή εφαρμογή (G-nnect), παρόμοια με την γνωστή εφαρμογή κοινωνικής δικτύωσης Twitter, αποτελούμενη από χρήστες, προφίλ χρηστών, μηνύματα, σχέσεις μεταξύ των χρηστών. Η εφαρμογή υλοποιήθηκε με βάση τις αρχές που διέπουν την σχεδιαστική αρχιτεκτονική MVC και αξιοποιώντας σύγχρονες προγραμματιστικές τεχνικές (Ajax, Websockets, Real-Time Data Flows).

Abstract

Today's internet, is a dynamic web from users, applications and devices - a constantly evolving platform, where users continuously carries an increasing share of their daily life.

The WebSocket of HTML5, represents the most major upgrade in the history of web. By leveraging this new standard along with a set of software designs patterns well-suited to dynamic data flows, this thesis try to presents how Living Web applications that astound users with the capabilities of real time interactions between them, can be built.

In this thesis, a prototype web application, named "G-nnect", similar to the famous social networking application Twitter, was developed, consisting of users, user profiles, messages, relationships between users. The application was implemented using the principles governing the design and MVC architecture using modern programming techniques

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον Δρ.Μενεκλή Βασίλειο για την συμβολή και την υποστήριξή που μου παρείχε για την ολοκλήρωση αυτής της εργασίας.

Ακόμη θα ήθελα να ευχαριστήσω τους κοντινούς μου ανθρώπους για την υπομονή που επέδειξαν καθ'όλη την διάρκεια των μεταπτυχιακών σπουδών μου.

Περιεχόμενα

1 Εισαγωγή

Η «Αραβική Άνοιξη» (Wikipedia, 2013a) πυροδοτήθηκε και τροφοδοτήθηκε μέσω των ιστοσελίδων κοινωνικής δικτύωσης όπως το Facebook (Wikipedia, 2013b) και το Twitter (Wikipedia, 2013c). Τις επόμενες προσεχείς ημέρες, τα μέσα κοινωνικής δικτύωσης, από απλό μέσο αλληλεπίδρασης και επικοινωνίας μεταξύ φίλων, έγιναν «όπλο» που παρείχε δύναμη στους ανθρώπους και επέφερε σημαντικές αλλαγές στον κόσμο. Ο κάθε ένας παρατήρησε τη δύναμη των ανθρώπων και οι άνθρωποι παρατήρησαν τις δυνατότητες των μέσων κοινωνικής δικτύωσης και για το τι ήταν ικανό να συμβεί μέσω αυτών. Η καρδιά όλου αυτού ήταν η τεχνολογία, που έκανε όλα αυ-τά δυνατά να συμβούν. Η τεχνολογία προσπέρασε όλα τα εμπόδια στην επικοινωνία και εξαπλώθηκε στον κόσμο γρηγορότερα ακόμα και από μία παγκόσμια πυρκαγιά. Αυτή είναι η δύναμη του διαδικτύου πραγματικού χρόνου (Real Time Web).

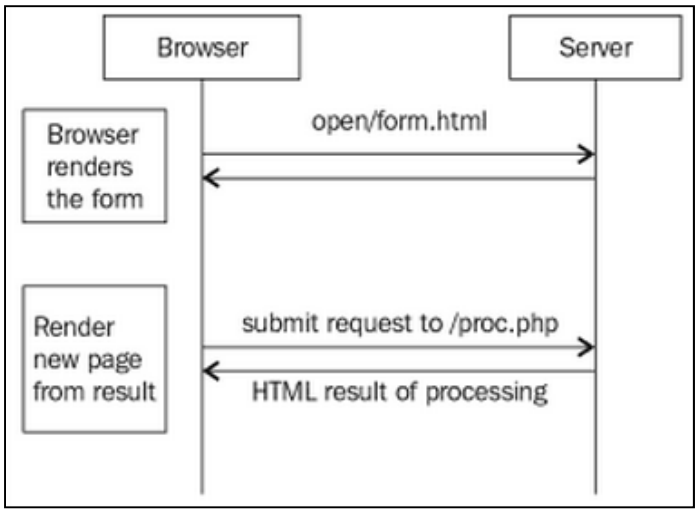
1.1 Τι είναι το διαδίκτυο πραγματικού χρόνου

Οι χρήστες του διαδικτύου είναι συνηθισμένοι σε ιστοσελίδες και εφαρμογές, όπου επιλέγουν κάποιον σύνδεσμο ή κάνουν κλικ σε κάποιο κουμπί ή αλλάζουν την τιμή σε κάποιο πεδίο εισαγωγής κειμένου και εκτελείται κάποια ενέργεια η οποία προκαλεί τυχούσες αλλαγές στην ιστοσελίδα που προβάλλεται. Ωστόσο αν κάποιος αφήσει ανοιχτή τη σελίδα του στο twitter για λίγη ώρα, μην έχοντας αλληλεπίδραση με αυτή, ο χρήστης θα ειδοποιηθεί όταν λάβει νέα μηνύματα. Αυτό εννοούμε γενικά όταν αναφερόμαστε στο διαδίκτυο πραγματικού χρόνου, με άλλα λόγια την ανανέωση του περιεχομένου μίας σελίδας ή μίας εφαρμογής χωρίς την παρεμβολή κάποιας ενέργειας του χρήστη-επισκέπτη. Όταν υπάρχει νέο περιεχόμενο να προβληθεί από την εφαρμογή, αυτή αναλαμβάνει να το προβάλλει.

1.2 Αναδρομή στο παρελθόν

Το διαδίκτυο πραγματικού χρόνου δεν είναι κάτι καινούργιο. Μία από τις πρώτες προσπάθειες υλοποίησης του ήταν με την χρήση των Java Applets (Wikipedia., 2013d). Πολλοί θα θυμούνται τις συνομιλίες στο Yahoo!, τα δωμάτια συνομιλιών (chat rooms), τα δικτυακά παιχνίδια σκακιού, στα τέλη της δεκαετίας του ‘90. Στη συνέχεια ήρθε η τεχνολογία Flash και τα πρόσθετα ActiveX (ActiveX Plug in). Αυτή η τάση δεν αφορούσε μόνο το χώρο της ψυχαγωγίας, αλλά απευθυνόταν και σε μεγάλες επιχειρήσεις και οργανισμούς (enterprise market). Γιατί όμως το διαδίκτυο πραγματικού χρόνου είναι τόσο σημαντικό αυτήν την εποχή; Επειδή, ο τρόπος που υλοποιείται η λειτουργικότητα του πραγματικού χρόνου στο διαδίκτυο, καθώς και το κόστος που επιφέρει κάτι τέτοιο έχουν πλέον αλλάξει. Από εκεί που θεωρούταν εξεζητημένο χαρακτηριστικό μιας εφαρμογής, πλέον αποτελεί αναγκαιότητα - απαίτηση χρηστών. Από εκεί που ήταν ένα δύσκολο από τεχνικής άποψης κομμάτι, τείνει να γίνει σιγά - σιγά ένα επικαιροποιημένο πρότυπο με την μορφή των WebSockets (Wikipedia, 2013e) και των Server Sent Events (SSE) (Wikipedia, 2013f). Πως όμως φτάσαμε από το στατικό διαδίκτυό σε αυτό το σημείο;

Το διαδίκτυο και οι διαδικτυακές εφαρμογές, όπως τις γνωρίζουν οι χρήστες, είναι φτιαγμένα πάνω από το πρωτόκολλο HTTP. Το πρωτόκολλο HTTP, αποτελεί ένα σύστημα αίτησης - απόκρισης, όπου ο «πελάτης» ένα αίτημα για πληροφορίες, στον «εξυπηρετητή», ο οποίος αποκρίνεται παρέχοντας τις ζητούμενες πληροφορίες στον «πελάτη». Στις περισσότερες περιπτώσεις, οι ζητούμενες πληροφορίες μπορεί να είναι είτε σελίδες HTML, είτε πληροφορίες της μορφής XML ή JSON. Στο σχήμα 1-1 απεικονίζεται η αλληλεπίδραση «πελάτη ή φυλλόμετρητη/client» - «εξυπηρετητή/server»

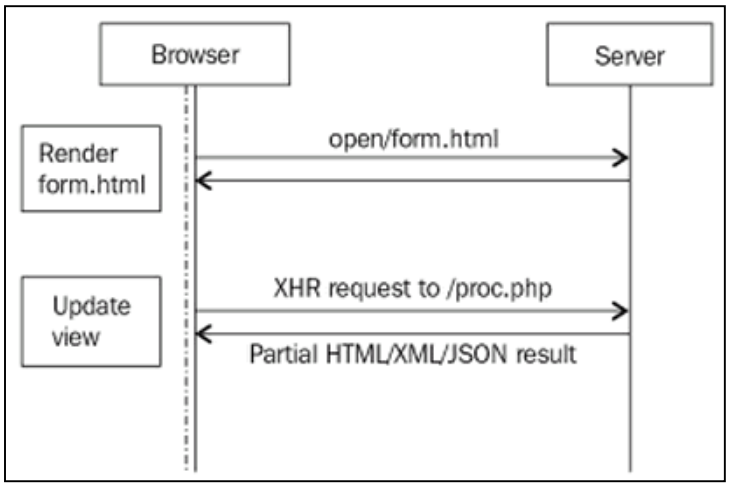


Σχήμα 1-1: Μοντέλο πελάτη/ εξυπηρετητή

Το 1995, οι εταιρίες Sun και Netscape, ανακοίνωσαν τη συνεργασία τους, με αποτέλεσμα την ενσωμάτωση του περιβάλλοντος εκτέλεσης της Java (Java Runtime) στο φυλλομετρητή της Netscape. Αυτή ήταν η αρχή του διαδραστικού διαδικτύου. Αν και τα Java Applets είχαν πολύ κακή φήμη, αποτέλεσαν κάτι πρωτοποριακό στο πεδίο του διαδικτύου πραγματικού χρόνου. Κατά τις πρώτες ημέρες του διαδικτύου πραγματικού χρόνου, τα java applets χρησιμοποιοντουσαν σχεδόν παντού, στις δικτυακές συνομιλίες, στα δικτυακά παιχνίδια ακόμα και στις δικτυακές διαφημίσεις.

Την ίδια χρονιά, η Netscape λάνσαρε τη γλώσσα προγραμματισμού σεναρίων, JavaScript (αρχικά LiveScript), και μια μικρή εταιρία ονόματι FutureWave ξεκίνησε να δουλεύει πάνω σε ένα λογισμικό animation, το FutureSplash Animator. Και τα δύο, αποτέλεσαν την αιτία, ώστε αργότερα τα Java Applets, σχεδόν να εξαφανισθούν από το διαδίκτυο.

Το 1999, η Microsoft χρησιμοποίησε τη δική της τεχνολογία πλαισίου (iframe) και τη γλώσσα JavaScript, προκειμένου να ανανεώνει ειδήσεις και τιμές μετοχών που προβάλλονταν στην προεπιλεγμένη κεντρική σελίδα (http://home.microsoft.com) του φυλλομετρητή Internet Explorer. Την ίδια χρονιά η Microsoft ανακοίνωσε μια επέκταση για τον Internet Explorer, το συστατικό XMLHTTP, το οποίο αρχικά προοριζόταν για να φορτώνει XML δεδομένα ασύγχρονα χρησιμοποιώντας JavaScript. Αυτή ήταν η εποχή, όπου η XML ήταν μόδα, και καθένας ήθελε να τη χρησιμοποιεί για οτιδήποτε υλοποιούσε. Πολύ γρήγορα, αυτή η επέκταση υιοθετήθηκε και από άλλους φυλλομετρητές, όπως Mozilla, Safari και Opera, με το συστατικό XmlHttpRequest (ή XHR για συντομία). Ωστόσο η εμφάνιση του Gmail από την Google ήταν που έκανε το AJAX (Asynchronous JavaScript and Xml) ευρέως χρησιμοποιούμενο όρο στην ανάπτυξη ιστοσελίδων. Το σχήμα 1-2 δείχνει ένα αίτημα με χρήση AJAX.

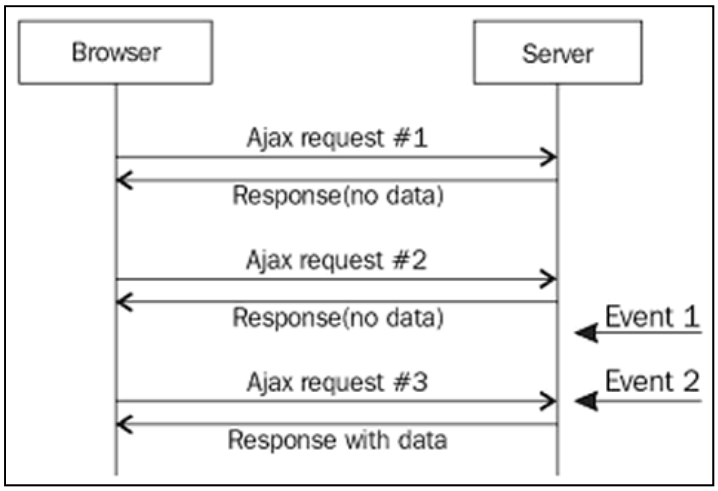


Σχήμα 1-2: Αίτημα με Javascript (Ajax)

Επίσης το Gmail έριξε φως στα πλεονεκτήματα των αυτόματων ενημερώσεων του περιεχομένου των ιστοσελίδων και άνοιξε την πόρτα σε διάφορες αμυχές[;] κτισμένες πάνω στο AJAX προκειμένου να ωθήσει δεδομένα από τον εξυπηρετητή (ή τουλάχιστον να δώσει την ψευδαίσθηση κάτι τέτοιου).

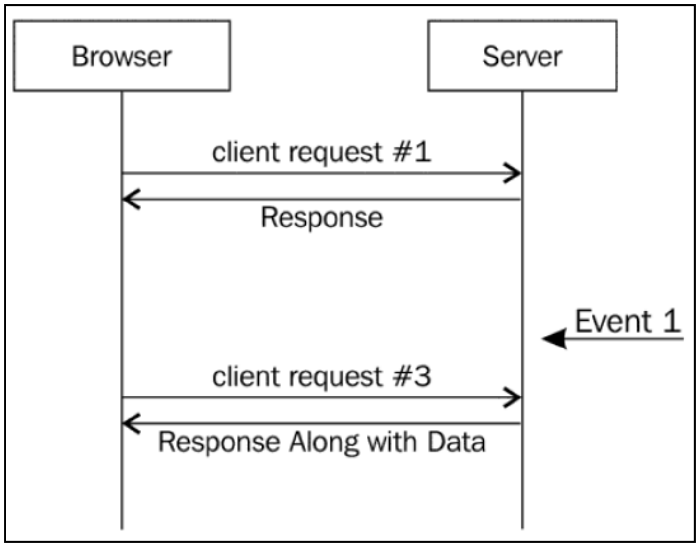
Συλλογικά, αυτές οι τεχνολογίες αναφέρονται με τον όρο Comet - ένας όρος που εισήγαγε ο Alex Russel το 2006 στο προσωπικό του ιστολόγιο (Russell, 2006). Η προσέγγιση Comet δεν αντιμετώπιζε το πρόβλημα των αυτόματων ενημερώσεων μονοδιάστατα. Αντιθέτως εισήγαγε πολλαπλούς μηχανισμούς για να δώσει την αίσθηση ότι τα δεδομένα ρέουν από τον εξυπηρετητή προς τον πελάτη. Αυτοί οι μηχανισμοί περιλάμβαναν κρυφά πλαίσια, περιοδικούς ελέγχους και λήψη δεδομένων με AJAX (AJAX Polling), μακροπρόσθεσμούς περιοδικούς ελέγχους και λήψη δεδομένων με AJAX (AJAX Long Polling), και την ετικέτα script για λήψη δεδομένων από διαφορετικά ονόματα χώρου (domains).

Ο πρώτος και ο πιο εύκολος στην υλοποίηση μηχανισμός είναι αυτός του AJAX Polling, όπου ο φυλλομετρητής (ή πελάτης) ελέγχει περιοδικά για δεδομένα, αποστέλλοντας ένα HTTP αίτημα, ενώ ο εξυπηρετητής απαντά με αποκρίσεις που δεν εμπεριέχουν δεδομένα, εκτός και αν έχει δεδομένα να στείλει πίσω στο φυλλομετρητη. Μετά από ένα γεγονός, όπως η λήψη ενός email ή η δημιουργία/ενημέρωση μίας εγγραφής στην βάση δεδομένων, ο εξυπηρετητής απαντά, στέλνοντας τα νέα δεδομένα ως απάντησης στο επόμενο αίτημα που θα λάβει. Στο Σχήμα 1-3 απεικονίζεται αυτός ο μηχανισμός.



Σχήμα 1-3: Λήψη δεδομένων με περιοδικά αιτήματα AJAX (Polling)

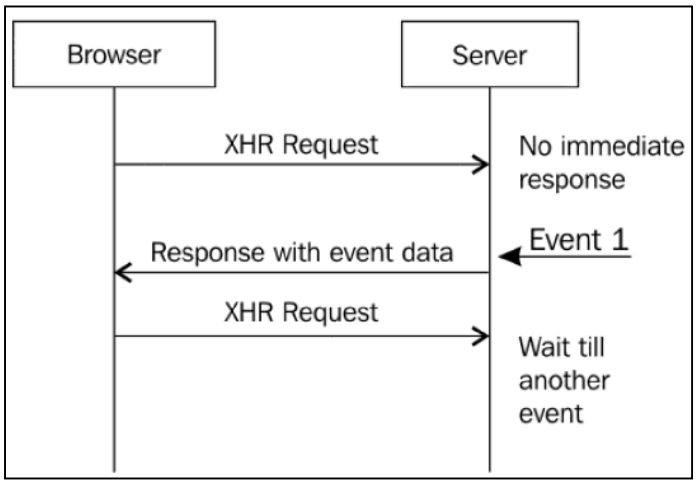
Ωστόσο, παρατηρεί κανείς ότι υπάρχει ένα πρόβλημα με αυτόν τον μηχανισμό. Ο φυλλομετρητής θα πρέπει να στέλνει αιτήματα προς τον εξυπηρετητή ακόμα και όταν δεν υπάρχουν δεδομένα για να λάβει πίσω. Αυτό υποχρεώνει τον εξυπηρετητή να λαμβάνει και να επεξεργάζεται δεδομένα από το αίτημα που δέχεται ακόμα και αν δεν έχει να παραδώσει πίσω τίποτα ως απάντηση. Μια λύση στο παραπάνω πρόβλημα είναι η εφαρμογή της τεχνικής όπου τα δεδομένα στέλνονται πίσω στον πελάτη μέσω άλλων μηνυμάτων (piggybacking). Με αυτό τον τρόπο ο εξυπηρετητής δε στέλνει πίσω στο φυλλομετρητή μόνο τα δεδομένα που αφορούν το εκάστοτε αίτημα που επεξεργάζεται, αλλά και επιπλέον πληροφορίες που τυχόν είναι διαθέσιμες. Προϋπόθεση ωστόσο, είναι ο πελάτης να είναι σε θέση να καταλαβαίνει και συνεπώς να ενεργεί ανάλογα, πάνω στα επιπλέον δεδομένα που λαμβάνει. Στο σχήμα 1-4 απεικονίζεται ο μηχανισμός piggybacking.



Σχήμα 1-4: Piggybacking

Καθώς τα νέα δεδομένα αποστέλλονται μόνο όταν υπάρχει μια ενέργεια από την πλευρά του τελικού χρήστη, αυτό προκαλεί καθυστερήσεις στην λήψη δεδομένων από τον φυλλομετρητή. Η λύση για άμεση λήψη γεγονότων, αποφεύγοντας τις συχνές ερωτήσεις στον εξυπηρετητή είναι η τεχνική Long Polling.

Στην τεχνική Long Polling, ο φυλλομετρητής στέλνει ένα αίτημα στον εξυπηρετητή, ο οποίος όμως δεν απαντάει αμέσως , αν δεν έχει δεδομένα να στείλει και έτσι αναστέλλει το αίτημα χωρίς να το διακόψει. Όταν ένα συμβάν, λάβει μέρος, τότε ο εξυπηρετητής κλείνει το υπό αναστολή αίτημα, απαντώντας ταυτόχρονα σε αυτό, στέλνοντας τα δεδομένα που έχει. Αμέσως μόλις ο πελάτης λάβει την απάντηση, αποστέλλει ένα νέο αίτημα.



Σχήμα 1-5: Long Polling

Αν και οι προαναφερθείσες τεχνικές δουλεύουν έως ένα βαθμό, ωστόσο βασίζονται στην μη ορθολογική χρήση του πρωτοκόλλου HTTP και του συστατικού XHR, καθώς αυτά δεν προορίζονταν για την υλοποίηση διαδικτυακών εφαρμογών πραγματικού χρόνου.

1.3 Εισαγωγή στις σύγχρονες τεχνολογίες διαδικτυακής επικοινωνίας

Με την ραγδαία εξέλιξη των φυλλομετρητών, οδηγούμενη αρχικά από τον Firefox και μετέπειτα από τον Chrome, η πολύ αναμενόμενη αναβάθμιση της HTML, η HTML5, υιοθετήθηκε ευρέως. Στην HTML5, υπάρχουν δύο νέες μέθοδοι για να σταλούν δεδομένα από τον εξυπηρετητή στον πελάτη. Στην πρώτη μέθοδο συναντάμε τον όρο Server Sent Events (SSE) και στην δεύτερη μέθοδο συναντάμε το πρωτόκολλο WebSocket.

Η μέθοδος SSE είναι πιο απλή από τις δύο και η επικοινωνία σε αυτήν είναι μονόδρομη (simplex), δηλαδή απλά στέλνονται δεδομένα από τον εξυπηρετητή προς τον πελάτη. Σε αυτή την προσέγγιση, υπάρχει μια διεπαφή προγραμματισμού (API) σε JavaScript για να δημιουργήσει μια πηγή γεγονότων (event source), δηλαδή ένα ρεύμα συνεχούς ροής δεδομένων (stream), στο οποίο ο εξυπηρετητής στέλνει συμβάντα. Εδώ συνεχίζεται να γίνεται αξιοποίηση του XHR αιτήματος.

Με το πρωτόκολλο WebSocket, το οποίο επίσης αποτελεί μια διεπαφή προγραμματισμού σε JavaScript, η επικοινωνία είναι ταυτόχρονα αμφίδρομη (full duplex). Ο φυλλομετρητής, αρχικοποιεί μια σύνδεση socket με τον εξυπηρετητή, ο οποίος, επίσης υποστηρίζει το πρωτόκολλο WebSocket. Πάνω σε αυτή την σύνδεση socket που δημιουργήθηκε, ο πελάτης και ο εξυπηρετητής ανταλλάσουν μηνύματα, στέλνοντας και λαμβάνοντας δεδομένα.

1.4 Σύνοψη

Σε αυτό το κεφάλαιο παρουσιάστηκε η πορεία και την εξέλιξη του συνεργατικού διαδικτύου με αναφορές σε τεχνικές που χρησιμοποιούνταν μέχρι και σήμερα.

Στο δεύτερο κεφάλαιο γίνεται μια συνοπτική παρουσίαση συγγραμμάτων και άλλων διπλωματικών εργασιών, τα οποία αποτέλεσαν πηγή έμπνευσης της παρούσας εργασίας.

Στο τρίτο κεφάλαιο γίνεται παρουσίαση του πρωτόκολλου WebSocket και της σχετικής προγραμματιστικής διεπαφής που το διέπει καθώς επίσης περιγράφονται οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής της παρούσας διπλωματικής εργασίας. Οι τεχνολογίες που παρουσιάζονται είναι, το πλαίσιο εργασίας Zend Framework, η βάση δεδομένων MongoDB, η βιβλιοθήκη ODM Doctrine, το σύστημα λογισμικού Node.js με την υποστηρικτική βιβλιοθήκη Socket.IO, ο μηχανισμός publish/ subscribe της in-memory βάσης δεδομένων Redis.

Στο τέταρτο κεφάλαιο παρουσιάζεται η διαδικασία σχεδίασης και ανάπτυξης της εφαρμογής με βάση τη μέθοδο ICONIX. Απαιτήσεις υψηλού επιπέδου, περιπτώσεις χρήσης, διαγράμματα ευρωστίας, ακολουθίας και κλάσεων είναι τα θέματα που παρουσιάζονται.

Στο πέμπτο κεφάλαιο αναφέρονται μελλοντικές επεκτάσεις και βελτιώσεις που επιδέχεται η εφαρμογή καθώς και τα τελικά συμπεράσματα που βγήκαν από την εκπόνηση της παρούσας διπλωματικής εργασίας.

2 Σχετική βιβλιογραφία

Σε αυτό το κεφάλαιο παρουσιάζονται περιληπτικά σχετικά συγγράμματα και ορισμένες εργασίες που μελετήθηκαν, οι οποίες ασχολήθηκαν με αντικείμενα, παρόμοια με αυτό της παρούσας διπλωματικής εργασίας. Είναι άξιες αναφοράς μιας και έδωσαν ιδέες για την υλοποίηση της παρούσας διπλωματικής εργασίας.

2.1 Βιβλιογραφία

Comet and Reverse Ajax - The Next-Generation 2.0

Phil McCarthy, Dave Crane

Σε αυτό το σύγγραμμα (Phil McCarthy, Dave Crane, 2008) αρχικά γίνονται εκτενείς αναφορές σε διάφορες τεχνικές (Polling, Long Polling, Piggybacking, iframes, κ.α.) που χρησιμοποιούνται για την όχι και τόσο ορθολογική προσομοίωση εφαρμογών πραγματικού χρόνου. Στη συνέχεια παρουσιάζεται το πρωτόκολλο Bayeux, όπου συναντάται η πρώτη αναφορά στο μοντέλο Δημοσίευση/ Συνδρομή (Publish - Subscribe), όπου τα δεδομένα διαχέονται σε εικονικά κανάλια μετάδοσης και τα δεδομένα λαμβάνονται από πελάτες - συνδρομητές των συγκεκριμένων καναλιών.

Αυτές οι αναφορές βοήθησαν ιδιαίτερα να γίνει κατανοητό και να προσδιοριστεί το πεδίο του προβλήματος της παρούσας εργασίας και αποτέλεσαν το ερέθισμα για περαιτέρω έρευνα.

The Definitive Guide to HTML5 WebSocket

Vanessa Wang, Frank Salim, Peter Moskovits

Σε αυτό το σύγγραμα (Wang, et al., 2013) γίνεται μια μεγάλη, εις βάθος συζήτηση για το τι είναι τα WebSockets, τα προβλήματα που μας βοηθούν να λύσουμε, καθώς και μια σειρά από πρακτικά παραδείγματα. Μας δείχνει πως μπορούν να υλοποιηθούν εφαρμογές διαδικτύου (web apps), πραγματικού χρόνου (real time), χρησιμοποιώντας μια νέα, επαναστατική, ευρέως διαδεδομένη, ανοιχτού προτύπου (open industry standard) τεχνολογία, ονόματι WebSocket, η οποία υποστηρίζει ταυτόχρονη αμφίδρομή (full-duplex, bidirectional) επικοινωνία μεταξύ της εφαρμογής πελάτη και απομακρυσμένων εξυπηρετητών μέσω του διαδικτύου, χωρίς τη χρήση πρόσθετων (plugins). Τέλος, εξηγεί τι πρέπει να γνωρίζει κανείς σχετικά με την τεχνολογία WebSocket και για πιο λόγο από εδώ και στο εξής θα πρέπει να σκεφτόμαστε με βάση αυτή τη νέα ανερχόμενη τεχνολογία καθώς εκτιμάται ότι θα τροποποιήσει όχι μόνο τον τρόπο που αναπτύσσονται οι εφαρμογές διαδικτύου, αλλά και τον τρόπο με τον οποίο αλληλεπιδρούν οι χρήστες με πληροφορίες και συσκευές μέσω του παγκόσμιου ιστού (Web).

Professional Node.js: Building Javascript Based Scalable Software

Pedro Teixeira

Η ανάπτυξη εφαρμογών διαδικτύου πραγματικού χρόνου είναι μία από τις κύριες περιπτώσεις χρήσης της τεχνολογίας Node και αυτό το σύγγραμα (Teixeira, 2012) εξηγεί πως αυτό είναι δυνατόν με τη χρήση επιπλέον βιβλιοθηκών όπως Connect, Express.js, and Socket.IO.

Ένα από τα συμπεράσματα που εξάγεται είναι ότι η βιβλιοθήκη Socket.IO αποτελεί ιδανική λύση, ανεξαρτήτου προγράμματος πλοήγησης για την παροχή αμφίδρομης και πραγματικού χρόνου επικοινωνία μεταξύ πελάτη και εξυπηρετητή. Πολλοί πελάτες μπορούν να συνδεθούν ο ένας με τον άλλον, μεταδίδοντας προς όλες τις συνδεδεμένες υποδοχές (sockets), ενώ επίσης υπάρχουν δυνατότητες σύνδεσης υποδοχών (socket join), αποχώρηση από συγκεκριμένα δωμάτια μετάδοσης (rooms) και επιλογή μετάδοσης μόνο σε συγκεκριμένα δωμάτια.

Επίσης γίνεται αναφορά στη δυνατότητα να κατανεμηθεί μια εφαρμογή Socket.IO σε πολλές διεργασίες Node.js χρησιμοποιώντας την βάση δεδομένων Redis, η οποία αποτελεί το κεντρικό σημείο επικοινωνίας

2.2 Σχετικές εργασίες

Enabling real-time collaborative data-intensive Web Applications - A case study using server-side JavaScript

Tobias Höfler, München, 2013

Σε αυτή την εργασία (Höfler, 2013) μελετάται ο τρόπος με τον οποίο εφαρμογές διαδικτύου πραγματικού χρόνου, συνεργατικές, κυριαρχούμενες από τα δεδομένα (data-intensive) μπορούν να υλοποιηθούν χρησιμοποιώντας JavaScript εκτελούμενη στον εξυπηρετητή με τη χρήση της πλατφόρμας NodeJS. Επιπλέον, ένα συνεργατικό πρόγραμμα επεξεργασίας κειμένου έχει σχεδιαστεί, που επιτρέπει τη συνεργατική επεξεργασία του κειμένου και παρέχει δυνατότητες συνεργασίας όπως συνομιλία και αποστολή πρόσκλησης σε άλλους χρήστες για να εργαστούν από κοινού στο ίδιο έγγραφο. Για την υλοποίηση του προγράμματος χρησιμοποιήθηκαν τεχνολογίες, ίδιες με την παρούσα διπλωματική (Node.js, Socket.IO).

Κατόπιν έρευνας που διεξήχθη σε οργανισμούς, τα τελικά αποτελέσματα φανερώνουν την γενικότερη ικανοποίηση που υπάρχει γύρω από την χρήση της JavaScript και νεοεισερχόμενης πλατφόρμας Node.js.

Αποδεικνύεται επίσης ότι η Node.js χρησιμοποιείται τυπικά για την υλοποίηση εφαρμογών διαδικτύου. Οι συμμετέχοντες επιπλέον επιβεβαιώνουν, ότι η Node.js είναι ήδη επιχειρησιακά έτοιμη (enterprise ready) και ότι η σημασία της θα αυξηθεί στο μέλλον.

Real-Time Voice Chat Realized in JavaScript

Afshin Aresh, Sweden, 2012

Σε αυτή την εργασία (Aresh, 2012) παρουσιάζεται η δυνατότητα επικοινωνίας σε πραγματικό χρόνο μεταξύ των προγραμμάτων πλοήγησης (Browsers) υλοποιημένη σε JavaScript. Ο στόχος είναι να συνδυάσει την HTML5 με προγραμματιστική διεπαφή, Mozilla Audio Data API, για την επικοινωνία μεταξύ των προγραμμάτων πλοήγησης (browsers). Για τη μετάδοση των δεδομένων στον πελάτη προορισμού αλλά και για τη λήψη δεδομένων από τον πελάτη, απαιτείται ένα αμφίδρομο κανάλι επικοινωνίας. Η προγραμματιστική διεπαφή της HTML5, WebSocket API παρέχει πλήρη αμφίδρομη επικοινωνία πάνω σε μια TCP σύνδεση. Το περιβάλλον του εξυπηρετητή που επιλέχθηκε σε αυτή την εργασία είναι το Node.js. Όλα τα παραπάνω συνδυάζονται μεταξύ τους προκείμενου να δοθεί η δυνατότητα αποστολής των δεδομένων μεταξύ των browsers μέσω ενός εξυπηρετητή.

Scalable web application using node.JS and CouchDB

Umesh Paudyal, Sweden , 2011

Ο στόχος αυτής της εργασίας (Paudyal, 2011) είναι η ανάπτυξη μιας πρότυπης εφαρμογής και η αξιολόγηση της επεκτασιμότητας (scalability) και της απόδοση της μαζί με μια βοηθητική μη-σχεσιακή (NoSql) βάση δεδομένων. Για την εφαρμογή χρησιμοποιήθηκε η τεχνολογία JavaScript (nodeJS) και ως βάση δεδομένων χρησιμοποιήθηκε η couchDB.

Το τελικό συμπέρασμα δείχνει ότι η τεχνολογία node.js παρέχει ένα πλαίσιο εργασίας το οποίο μπορεί να χρησιμοποιηθεί για τη δημιουργία επεκτάσιμων και υψηλής απόδοσης εξυπηρετητών, δεδομένου και της ασύγχρονης προσέγγισης προγραμματισμού το διέπει. Τέλος εξηγείται πως η CouchDB βάση δεδομένων μπορεί να γίνει επεκτάσιμη και κατανεμημένη με κάποια τροποποιήσεις όπως αντιγραφή (replication), ομαδοποίηση (clustering) και την εξισορρόπηση φορτίου (load balancing) της βάσης δεδομένων.

Real-time annotation of video streams using staged processing

Holmstad Øyvind, Norwegian, 2011

Σε αυτή την εργασία (Holmstad, 2011) άξιο αναφοράς αποτελεί η χρησιμοποίηση του προτύπου μηνύματος, δημοσίευσης/συνδρομής (Publish/Subscribe) που υλοποιείτε από την in-memory βάση δεδομένων Redis, ούτος ώστε να υπάρχει ροή γεγονότων και ενημερώσεων τον όποιον συστατικών του συστήματος σε πραγματικό χρόνο

Investigating NoSQL from a SQL Perspective

PETER SÖDERGREN, BJÖRN ENGLUND, Stockholm, 2011

Σε αυτή την εργασία (PETER SÖDERGREN, BJÖRN ENGLUND, 2011) εκτελούνται κάποιες δοκιμές επιδόσεων σε τρεις, μη - σχεσιακές βάσεις δεδομένων, ονόματι couchDB, MongoDB και Neo4j. Για αυτές τις δοκιμές χρησιμοποιήθηκαν τύποι και δομές δεδομένων που μπορούν να βρεθούν περισσότερο σε ένα κοινωνικό δίκτυο, που περιλαμβάνει ανθρώπους, μηνύματα, γεγονότα, σχόλια, κ.α.

Τα αποτελέσματα από τις δοκιμές επιδόσεων έδειξαν ότι η MongoDB είναι η γρηγορότερη και αυτό διότι είναι η καταλληλότερη για τον τύπο των δεδομένων μας και των ερωτημάτων τις συγκεκριμένης εργασίας.

Average time (ms)	MongoDB	Neo4j	CouchDB
Query nr 1 / BIG	10,1	95,6	295,05
Query nr 2 / BIG	44,25	103,05	1867,6
Query nr 1 / SMALL	6,25	103,85	204,65
Query nr 2 / SMALL	22,45	85,9	956,3

Πίνακας 2-1: Αποτελέσματα μετρήσεων απόδοσης

2.3 Σύνοψη

Σε αυτό το κεφάλαιο έγινε παρουσίαση συγγραμμάτων και άλλων εργασιών που πραγματεύονται θέμα, παρόμοιο με αυτό της παρούσας διπλωματικής εργασίας, παρουσιάζοντας περιπτώσεις χρήσης αλλά και προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν. Με αυτό το κεφάλαιο, ολοκληρώνεται το θεωρητικό μέρος της εργασίας.

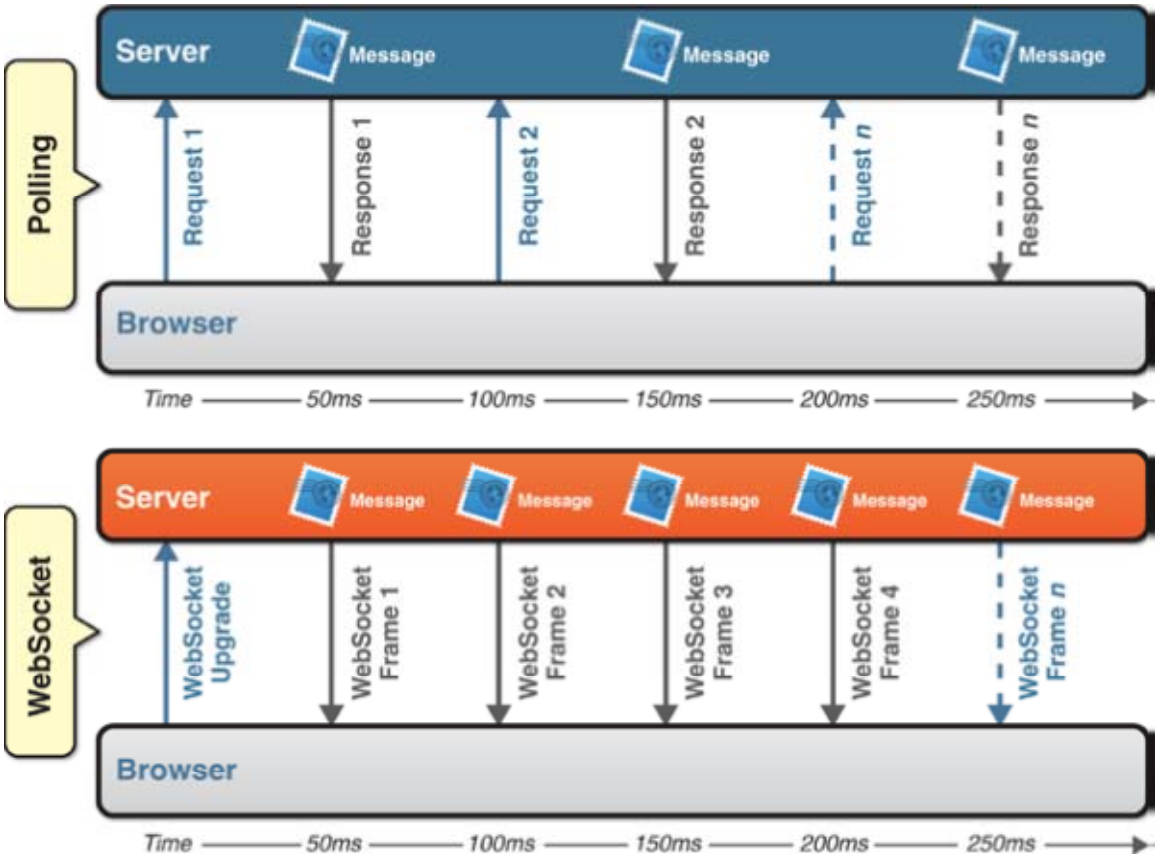
Στο επόμενο κεφάλαιο θα εμβαθύνουμε σε ποιο τεχνικές λεπτομέρειες με την εισαγωγή στο πρωτόκολλο WebSocket, στη προγραμματιστική διεπαφή (API) που το διέπει και στις τεχνολογίες που χρησιμοποιούνται για την υλοποίηση της παρούσας διπλωματικής εργασίας.

3 Τεχνολογίες

3.1 Πρωτόκολλο WebSocket

Το WebSocket είναι μία μοναδική σύνδεση υποδοχής (socket connection) ή τερματική σύνδεση, που επιτρέπει ταυτόχρονη και αμφίδρομη (full-duplex, bi-directional) επικοινωνία. Με το πρωτόκολλο WebSocket, το HTTP αίτημα, μετατρέπεται σε ένα μοναδικό αίτημα, για το άνοιγμα μια σύνδεσης υποδοχής, χρησιμοποιόντας την ίδια σύνδεση τόσο από τον πελάτη προς τον εξυπηρετητή, όσο και από τον εξυπηρετητή προς τον πελάτη.

Με την τεχνολογία WebSocket, μειώνεται σημαντικά η καθυστέρηση επικοινωνίας και ανταλλαγής δεδομένων, διότι από την στιγμή που εγκατασταθεί μια σύνδεση WebSocket, ο εξυπηρετητής μπορεί και στέλνει πίσω στον πελάτη δεδομένα, όσο αυτά είναι διαθέσιμα. Σε αντίθεση με την τεχνική Polling, με το WebSocket γίνεται ένα μοναδικό αίτημα, ενώ στην συνέχεια, ο εξυπηρετητής δεν χρειάζεται να περιμένει αίτημα από τον πελάτη. Παρόμοια, ο πελάτης μπορεί να στέλνει μηνύματα στον εξυπηρετητή ανά πάσα στιγμή. Αυτό το μοντέλο αποστολής ενός μοναδικού αιτήματος μειώνει σημαντικά την καθυστέρηση, αντί της τεχνικής Polling, με την οποία στέλνεται ένα αίτημα ανά τακτά χρονικά διαστήματα, ανεξάρτητα των δεδομένων που είναι διαθέσιμα.



Σχήμα 3-1: Polling vs WebSocket

3.1.1 Εκκίνηση Εγκαθίδρυσης Σύνδεσης (Opening handshake)

Κάθε σύνδεση WebSocket, ξεκινάει με ένα αίτημα HTTP. Αυτό το αίτημα, είναι όπως κάθε άλλο αίτημα HTTP, με την διαφορά ότι περιλαμβάνει μια επιπλέον ειδική κεφαλίδα - «Upgrade». Η κεφαλίδα «Upgrade» υποδεικνύει, ότι ο πελάτης επιθυμεί να αναβαθμίσει τη σύνδεση χρησιμοποιώντας διαφορετικό πρωτόκολλο. Σε αυτή την περίπτωση, αυτό το διαφορετικό πρωτόκολλο είναι το WebSocket.

Σε αυτό σημείο, παρουσιάζεται ένα παράδειγμα εκκίνησης σύνδεσης (ή χειραψίας) WebSocket που καταγράφεται κατά την διάρκεια σύνδεσης στην τοποθεσία ws://echo.websocket.org. Μέχρις ότου ολοκληρωθεί η εγκαθίδρυση σύνδεσης, η συνεδρία WebSocket (WebSocket Session), συμμορφώνονται σύμφωνα με το πρωτόκολλο HTTP/1.1 .

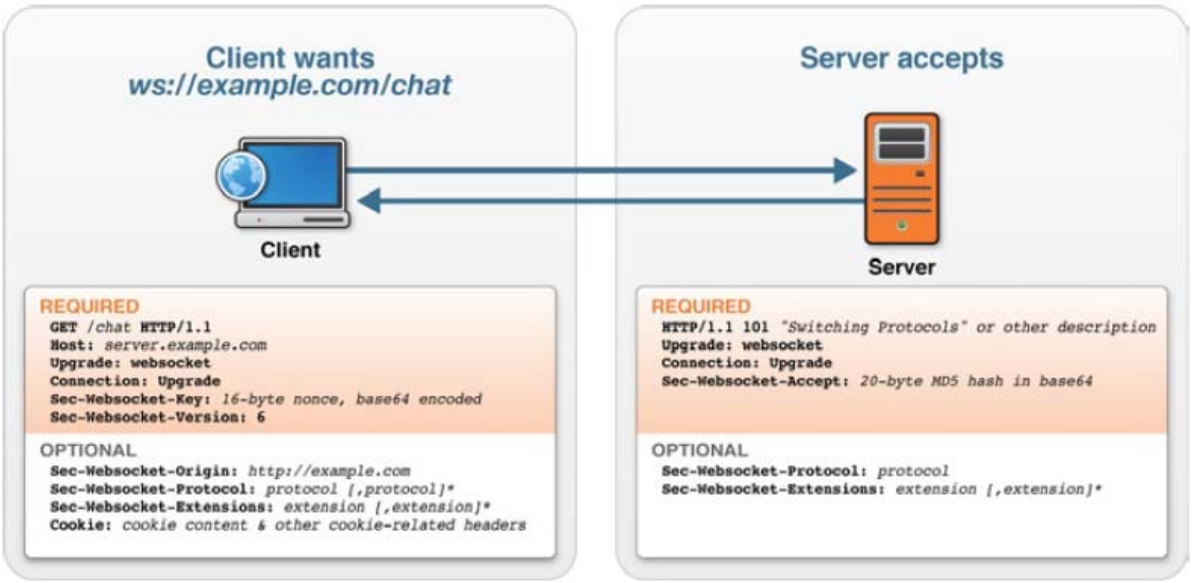
Παρακάτω φαίνεται ένα αίτημα HTTP, όπως τα στέλνει ο πελάτης:

GET /echo HTTP/1.1
Host: echo.websocket.org
Origin: http://www.websocket.org
Sec-WebSocket-Key: 7+C600xYybOv2zmJ69RQsw==
Sec-WebSocket-Version: 13
Upgrade: websocket

Παρακάτω φαίνεται η απόκριση του εξυπηρετητή στο παραπάνω αίτημα:

101 Switching Protocols
Connection: Upgrade
Date: Wed, 20 Jun 2012 03:39:49 GMT
Sec-WebSocket-Accept: fYoqiH14DgI+5ylEMwM2sOLzOi0=
Server: Kaazing Gateway
Upgrade: WebSocket

Στο σχήμα 3-2 φαίνεται η αναβάθμιση ενός αιτήματος από τον πελάτη προς τον εξυπηρετητή, σε websocket. Διαδικασία γνωστή και ως «WebSocket σύναψη χειραψίας».



Σχήμα 3-2: Παράδειγμα εγκαθίδρυσης σύνδεσης WebSocket

Στο σχήμα 3-2 φαίνονται οι απαιτούμενες και οι προαιρετικές κεφαλίδες. Κάποιες κεφαλίδες είναι απολύτως αναγκαίες και πρέπει να είναι παρούσες και ακριβείς για να πετύχει μια WebSocket σύνδεση. Άλλες κεφαλίδες σε αυτή τη χειραψία είναι προαιρετικές, αλλά επιτρέπονται, διότι η χειραψία είναι μια αίτηση και απάντηση HTTP. Μετά από μια επιτυχή αναβάθμιση, η σύνταξη της σύνδεσης μεταβαίνει σε μια μορφή πλαισίωσης δεδομένων (data-framing) που χρησιμοποιείται για την αναπαράσταση των μηνυμάτων WebSocket. Η σύνδεση δε θα πετύχει αν ο εξυπηρετητής δεν απαντήσει με κωδικό απόκρισης 101, την κεφαλίδα «Upgrade» και την κεφαλίδα «Sec-WebSocket-Accept» Η τιμή της κεφαλίδας απόκρισης «Sec-WebSocket-Accept», προέρχεται από την τιμή της κεφαλίδας αιτήματος «Sec-WebSocket-Key» και περιέχει ένα ειδικό «κλειδί» απόκρισης που πρέπει να ταιριάζει ακριβώς με ότι αναμένει ο πελάτης.

3.1.2 Υπολογισμός του κλειδιού απόκρισης

Για να ολοκληρωθεί επιτυχώς μια WebSocket χειραψία ώστε να εγκαθιδρυθεί η σύνδεση, θα πρέπει ο εξυπηρετητής να απαντήσει με ένα υπολογισμένο κλειδί. Μια τέτοια απόκριση θα σημαίνει ότι ο εξυπηρετητής καταλαβαίνει πλήρως το WebSocket πρωτόκολλο. Χωρίς την ακριβή απόκριση, μπορεί να καταστεί δυνατόν να ξεγελαστεί ένας ανυποψίαστος HTTP εξυπηρετητής, ώστε να αναβαθμίσει μια αίτηση σύνδεσης κατά λάθος.

Η λειτουργία απόκρισης στην πλευρά του εξυπηρετητή, παίρνει την τιμή της κεφαλίδας «Sec-WebSocket-Key» που έστειλε ο πελάτης με το αίτημα του και επιστρέφει την υπολογιζόμενη τιμή που ο πελάτης περιμένει στην κεφαλίδα απόκρισης «Sec-WebSocket-Accept».

Στη διαδικασία για εγκαθίδρυση σύνδεσης WebSocket και στον υπολογισμό του κλειδιού απόκρισης, το πρωτόκολλο WebSocket, βασίζεται στις κεφαλίδες με το πρόθεμα sec, οι οποίες ορίζονται στο πρωτόκολλο μορφοποίησης RFC 6455 (Fette & Melnikov, 2011). Στο πίνακα 1 περιγράφονται οι κεφαλίδες -sec- του πρωτοκόλλου WebSocket.

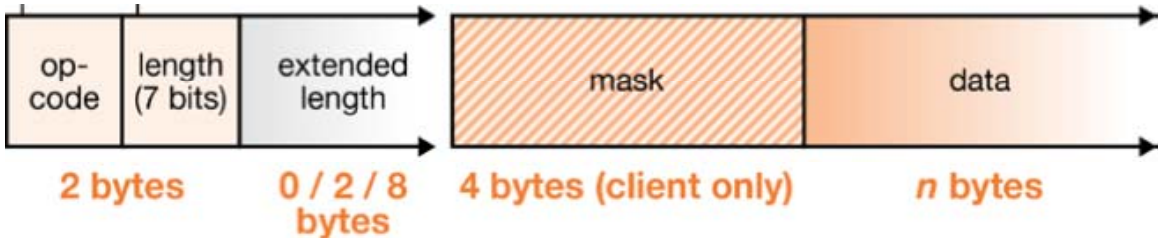
Κεφαλίδα	**Περιγραφή**
Sec-WebSocket-Key	Μπορεί να εμφανίζεται μόνο μία φορά σε ένα αίτημα HTTP. Χρησιμοποιείται στην διαδικασία για εγκαθίδρυση σύνδεσης WebSocket από τον πελάτη προς τον εξυπηρετητή για να αποφθεχθούν επιθέσεις από άλλα πρωτόκολλα.
Sec-WebSocket-Accept	Μπορεί να εμφανίζεται μόνο μία φορά σε μια απόκριση HTTP. Χρησιμοποιείται στην διαδικασία για εγκαθίδρυση σύνδεσης WebSocket από το εξυπηρετητή στον πελάτη, για να επιβεβαιώσει ότι ο εξυπηρετητής καταλαβαίνει το πρωτόκολλο WebSocket.
Sec-WebSocket-Extensions	Μπορεί να εμφανιστεί πολλές φορές σε ένα αίτημα HTTP, αλλά μπορεί να εμφανίζεται μόνο μία φορά σε μια απόκριση HTTP. Χρησιμοποιείται στην διαδικασία για εγκαθίδρυση σύνδεσης WebSocket από το εξυπηρετητή στον πελάτη και από τον πελάτη στον εξυπηρετητή. Αυτή η κεφαλίδα βοηθά τον πελάτη και εξυπηρετητή να συμφωνήσουν σε ένα σύνολο επεκτάσεων επιπέδου πρωτοκόλλων που θα χρησιμοποιηθούν κατά την διάρκεια της σύνδεσης
Sec-WebSocket-Protocol	Χρησιμοποιείται στην διαδικασία για εγκαθίδρυση σύνδεσης WebSocket από το πελάτη στον εξυπηρετητή, στη συνέχεια, από τον εξυπηρετητή για να διαπραγματευτεί ένα υπό-πρωτόκολλο. Αυτή η κεφαλίδα γνωστοποιεί τα πρωτόκολλα που η εφαρμογή-πελάτη μπορεί να χρησιμοποιήσει. Ο εξυπηρετητής χρησιμοποιεί την ίδια κεφαλίδα για να επιλέξει το πολύ ένα από αυτά τα πρωτόκολλα.
Sec-Websocket-Version	Χρησιμοποιείται στην διαδικασία για εγκαθίδρυση σύνδεσης WebSocket από το πελάτη στον εξυπηρετητή για την συμβατότητα έκδοσης. Η έκδοση για RFC 6455 είναι πάντα 13. Ο εξυπηρετητής απαντά με αυτή την επικεφαλίδα, αν δεν υποστηρίζει την έκδοση του πρωτοκόλλου που έχει ζητηθεί από τον πελάτη. Στην περίπτωση αυτή, η κεφαλίδα που αποστέλλεται από το εξυπηρετητή παραθέτει τις εκδόσεις που υποστηρίζει.

Πίνακας 1: Επικεφαλίδες μηνύματος WebSocket

3.1.3 Μορφή Μηνύματος

Όσο μια σύνδεση WebSocket είναι ανοικτή, ο πελάτης και ο εξυπηρετητής μπορούν να στέλνουν μηνύματα ο ένας στον άλλο, ανά πάσα στιγμή. Αυτά τα μηνύματα αναπαριστούνται στο δίκτυο με μια δυαδική σύνταξη που σηματοδοτεί τα όρια μεταξύ των μηνυμάτων και περιλαμβάνει περιεκτικές πληροφορίες για τον τύπο τους. Πιο συγκεκριμένα, αυτές οι δυαδικές κεφαλίδες σηματοδοτούν τα όρια ανάμεσα στα επονομαζόμενα πλαίσια (frames). Τα πλαίσια είναι δεδομένα που από μόνα τους δεν έχουν ιδιαίτερη σημασία, άλλα μπορούν να συνδυαστούν μεταξύ τους, για να σχηματίσουν μηνύματα. Οι όροι «Πλαίσιο» και «Μήνυμα» χρησιμοποιούνται με την ίδια σημασία όταν μιλάμε για WebSocket. Αυτό γίνεται, γιατί είναι σπάνιο (τουλάχιστον σήμερα) να χρησιμοποιείται περισσότερο από ένα πλαίσιο ανά μήνυμα. Επίσης, στα πρώτα προσχέδια του πρωτοκόλλου, τα πλαίσια ήταν μηνύματα, και η αναπαράσταση του μηνύματος στο σύρμα ονομάζονταν πλαισίωση.

Υπάρχει συνήθως μόνο ένα πλαίσιο σε ένα μήνυμα, αλλά ένα μήνυμα που μπορεί να αποτελείται από οποιοδήποτε αριθμό πλαισίων. Το σχήμα 3-3 απεικονίζει την κεφαλίδα ενός πλαισίου WebSocket.



Σχήμα 3-3: Κεφαλίδα WebSocket

3.1.4 Κωδικός εντολής (op-code)

Κάθε μήνυμα WebSocket, έχει έναν κωδικό εντολής, διευκρινίζοντας τον τύπο του ωφέλιμου φορτίου ή πληροφορίας (payload) του μηνύματος. Ο κωδικός εντολής αποτελείται από τα τέσσερα τελευταία bits του πρώτο byte της κεφαλίδας του πλαισίου. Οι κωδικοί εντολών έχουν μια αριθμητική τιμή, όπως περιγράφονται στο πίνακα 2.

Κωδικός εντολής	**Τύπος ωφέλιμου φορτίου του μηνύματος**	**Περιγραφή**
1	Text	Ο τύπος δεδομένων του μηνύματος είναι κείμενο.
2	Binary	Ο τύπος δεδομένων του μηνύματος είναι δυαδικός (δεδομένα σε δυαδική αναπαράσταση)
8	Close	Ο πελάτης ή ο εξυπηρετητής στέλνει μια χειραψία κλεισίματος στον εξυπηρετητή ή στον πελάτη.
9	Ping	Ο πελάτης ή ο εξυπηρετητής στέλνει ένα αίτημα ping στον εξυπηρετητή ή στον πελάτη.
10	Pong	Ο πελάτης ή ο εξυπηρετητής στέλνει ένα αίτημα pong στον εξυπηρετητή ή στον πελάτη.

Πίνακας 2: Κωδικοί εντολών

Με τέσσερα δυαδικά ψηφία να χρησιμοποιούνται για κωδικούς εντολών, μπορεί να υπάρχουν έως και 16 διαφορετικές τιμές. Το πρωτόκολλο WebSocket ορίζει μόνο πέντε κωδικούς εντολών, και οι εναπομείναντες προορίζονται για μελλοντική χρήση σε επεκτάσεις.

3.1.5 Μήκος πλαισίου (length)

Το πρωτόκολλο WebSocket κωδικοποιεί το μήκος του πλαισίου χρησιμοποιώντας ένα μεταβλητό αριθμό bits, που επιτρέπει μικρά μηνύματα να χρησιμοποιούν μια συμπαγή κωδικοποίηση, ενώ εξακολουθεί να επιτρέπει στο πρωτόκολλο να μεταφέρει μεσαίου ή ακόμα και μεγάλου μεγέθους μηνύματα. Για μηνύματα μεγέθους κάτω από 126 Bytes, το μήκος εμπεριέχεται σε ένα από τα πρώτα δύο Bytes της κεφαλίδας. Για μηνύματα μεγέθους μεταξύ 126 και 216 Bytes, χρησιμοποιούνται δύο επιπλέον Bytes.

3.1.6 Αποκωδικοποίηση Κείμενο (decoding text)

Το WebSocket μήνυμα κειμένου, κωδικοποιείται με βάση το πρότυπο κωδικοποίησης UTF-8. Το UTF-8 πρόκειται για μια κωδικοποίηση μεταβλητού μήκους, συμβατή και με την κωδικοποίηση ASCII. Η κωδικοποίηση UTF-8 είναι η μόνη που επιτρέπεται στα WebSocket μηνύματα κειμένου.

Διατηρώντας την κωδικοποίηση UTF-8, αποτρέπεται μια Βαβέλ διαφορετικών κωδικοποιήσεων που υπάρχουν σε μυριάδες μορφοποιήσεις απλού κειμένου (plain text) και πρωτόκολλα, από την παρεμπόδιση της διαλειτουργικότητας.

3.1.7 Απόκρυψη (masking)

Τα WebSocket πλαίσια (frames) που αποστέλλονται από το πρόγραμμα πλοήγησης (browser) προς τον εξυπηρετητή, είναι «επικαλυπτόμενα» (masked) προκειμένου να επιτευχθεί η απόκρυξη του περιεχομένου τους. Σκοπός της διαδικασίας επικάλυψής (masking) δεν είναι η αποτροπή υποκλοπών, αλλά προορίζεται για ασυνήθιστους λόγους ασφαλείας και να βελτιώσει την συμβατότητα με ήδη υπάρχοντες εξυπηρετητές διαμεσολάβησης (Proxy Servers).

Το πρώτο bit, του δεύτερου byte της κεφαλίδας του πλαισίου, δείχνει αν το πλαίσιο είναι επικαλυπτόμενο (masked). Αν υπάρχει μια μάσκα, θα είναι τέσσερα bytes, μετά το διευρυμένο τμήμα μήκους της κεφαλίδας πλαισίου.

Κάθε ωφέλιμο φορτίο που λαμβάνεται από έναν WebSocket εξυπηρετητή, πρέπει να υποστεί αφαίρεση μάσκας (unmasking), πριν την επεξεργασία του. Μετά την αφαίρεση μάσκας, ο εξυπηρετητής θα έχει το αρχικό περιεχόμενο του μηνύματος; Τα μηνύματα σε δυαδική μορφή μπορούν να παραδοθούν άμεσα ενώ τα μηνύματα κειμένου θα υποστούν UTF-8 αποκωδικοποίηση και θα είναι διαθέσιμα σαν συμβολοσειρές από την προγραμματιστική διεπαφή WebSocket στην πλευρά του εξυπηρετητή.

3.1.8 Πολλαπλά πλαίσια μηνυμάτων

Το fin bit στη μορφή πλαίσιο WebSocket, επιτρέπει μηνύματα πολλαπλών πλαισίων ή την ροή μερικώς διαθέσιμών μηνυμάτων, τα οποία είναι κατακερματισμένα ή μη ολοκληρωμένα. Για την μετάδοση ενός μη ολοκληρωμένου μηνύματος, αποστέλλεται ένα πλαίσιο, έχοντας το fin bit ίσο με το μηδέν. Το τελευταίο πλαίσιο έχει fin bit ίσο με ένα, δείχνοντας με αυτό τον τρόπο ότι το μήνυμα τελειώνει με το ωφέλιμο φορτίο (payload) του πλαισίου.

3.1.9 Κλείσιμο σύνδεσης (Closing handshake)

Οι WebSocket συνδέσεις αρχίζουν πάντα με μια χειραψία ανοίγματος, καθώς αυτός είναι ο μόνος τρόπος για να προετοιμαστεί η μετέπειτα συνομιλία. Στο Διαδίκτυο και άλλα αναξιόπιστα δίκτυα, οι συνδέσεις μπορεί να κλείσουν ανά πάσα στιγμή, γι 'αυτό δεν είναι δυνατόν να ειπωθεί ότι οι συνδέσεις καταλήγουν πάντα με μια χειραψία κλεισίματος. Μερικές φορές η βασική υποδοχή(socket) TCP κλείνει απότομα. Η χειραψία κλεισίματος, κλείνει με ευγενικό τρόπο τις συνδέσεις, επιτρέποντας στις εφαρμογές να καταλάβουν τη διαφορά μεταξύ συνδέσεων του τερματίστηκαν από πρόθεση και συνδέσεων που τερματίστηκαν κατά λάθος.

Όταν μια WebSocket σύνδεση τερματίζει, το τελικό σημείο(εξυπηρετητής ή πελάτης) που τερματίζει την σύνδεση, στέλνει ένα αριθμητικό κώδικα και ένα λεκτικό για να εξηγήσει για ποιο λόγο επιλέγει να κλείσει την socket σύνδεση. Ο κωδικός και το λεκτικό κωδικοποιούνται μέσα στην ωφέλιμη πληροφορία του πλαισίου με κωδικό εντολής 8 (close).

3.2 Διεπαφή Προγραμματισμού (API) WebSocket

Η διεπαφή προγραμματισμού WebSocket, είναι μια διεπαφή που επιτρέπει στις εφαρμογές να χρησιμοποιούν το πρωτόκολλο WebSocket. Με την χρήση του API, οι εφαρμογές μπορούν και ελέγχουν ένα ταυτόχρονα αμφίδρομο κανάλι επικοινωνίας, μέσω του οποίου μπορούν να στέλνουν και να λαμβάνουν μηνύματα.

Όπως έχει ήδη αναφερθεί, μια WebSocket σύνδεση εδραιώνεται με την αναβάθμιση του πρωτόκολλο HTTP σε πρωτόκολλο WebSocket, κατά τη διάρκεια της αρχικής χειραψίας μεταξύ πελάτη και εξυπηρετητή, υπό την ίδια υποκείμενη TCP σύνδεση. Μετά την εδραίωση της σύνδεσης, μηνύματα μπορούν να σταλούν και προς τις δύο κατευθύνσεις με την βοήθεια των μεθόδων που ορίζονται στην διεπαφή WebSocket. Στον κώδικα της εφαρμογής, χρησιμοποιούνται ασύγχρονοι χειριστές γεγονότων (event listeners) για να χειριστούν κάθε φάση του κύκλου ζωής μιας σύνδεσης.

Το WebSocket API είναι μια καθοδηγούμενη από συμβάντα (event driven) διεπαφή προγραμματισμού. Μετά την εδραίωση μιας ταυτόχρονης αμφίδρομης σύνδεση, όταν ο εξυπηρετητής έχει δεδομένα να στείλει στον πελάτη, ή αν οι πόροι που ενδιαφέρουν τον πελάτη αλλάξουν την κατάστασή τους, τότε ο εξυπηρετητής στέλνει αυτόματα δεδομένα ή ειδοποιήσεις.

Με την καθοδηγούμενη από συμβάντα διεπαφή, ο πελάτης δεν χρειάζεται να εφαρμόσει την τεχνική polling για να λάβει την πιο ενημερωμένη κατάσταση του στοιχειοθετημένου πόρου, αντίθετα ο πελάτης απλά ακούει για ενημερώσεις και αλλαγές.

3.2.1 Κατασκευαστής (Constructor) WebSocket

Για τη δημιουργία μια σύνδεσης WebSocket σε έναν εξυπηρετητή, αρχικοποιείται ένα αντικείμενο WebSocket, δείχνοντας σε μια διεύθυνση URL, που αντιπροσωπεύει το τελικό σημείο που δύναται να γίνει η σύνδεση. Το πρωτόκολλο WebSocket ορίζει δύο σχήματα URI, το ws και το wss για μη κρυπτογραφημένη και κρυπτογραφημένη κίνηση μεταξύ του πελάτη και του εξυπηρετητή αντίστοιχα. Το WebSocket σχήμα ws, είναι ανάλογο με αυτό του HTTP URI. Το wss (WebSocket Secure) URI σχήμα, αντιπροσωπεύει μια ασφαλή WebSocket σύνδεση πάνω από το ασφαλές πρωτόκολλο TLS και χρησιμοποιεί τον ίδιο μηχανισμό ασφάλειας με αυτό που χρησιμοποιεί το πρωτόκολλο HTTPS για ασφαλής συνδέσεις.

Ο κατασκευαστής WebSocket παίρνει ένα απαιτούμενο όρισμα, το URL (η διεύθυνση URL του εξυπηρετητή που δύναται να πραγματοποιηθεί η σύνδεση) και ένα προαιρετικό όρισμα, τα πρωτόκολλα (είτε ως ένα ενιαίο όνομα του πρωτοκόλλου ή ένας πίνακας από ονόματα πρωτόκολλων που ο εξυπηρετητής πρέπει να περιλαμβάνει στην απάντησή του προς εγκαθίδρυση της σύνδεσης.

Το παρακάτω απόσπασμα κώδικα δείχνει την χρήση του κατασκευαστή WebScocket, με ένα όρισμα, το οποίο πρέπει να είναι μια πλήρης χαρακτηρισμένη διεύθυνση URL ξεκινώντας με ws:// ή wss:// σχήμα. Εάν υπάρχει συντακτικό λάθος στο URL, ο κατασκευαστής θα εγείρει μια εξαίρεση (throw exception).

```
var ws = new WebSocket("ws://www.websocket.org");
```

Κατά τη σύνδεση σε ένα εξυπηρετητή WebSocket, μπορεί να χρησιμοποιηθεί προαιρετικά το δεύτερο επιχείρημα για να γνωρίσει η εφαρμογή στον εξυπηρετητή την λίστα με τα πρωτόκολλα που υποστηρίζει, δηλαδή υπάρχει μια διαπραγμάτευση του πρωτοκόλλου που θα χρησιμοποιηθεί.

Για να εξασφαλιστεί ότι ο πελάτης και ο εξυπηρετητής αποστέλλουν και λαμβάνουν μηνύματα που καταλαβαίνουν και οι δύο πλευρές, θα πρέπει να χρησιμοποιούν το ίδιο πρωτόκολλο. Ο κατασκευαστής WebSocket παρέχει την δυνατότητα να οριστεί το πρωτόκολλο ή τα πρωτόκολλα που ο πελάτης μπορεί να χρησιμοποιήσει για την επικοινωνία με τον WebSocket εξυπηρετητή. Ο

εξυπηρετητής με την σειρά του διαλέγει το πρωτόκολλο που θα χρησιμοποιηθεί. Μόνο ένα πρωτόκολλο μπορεί να χρησιμοποιηθεί. Αυτά τα πρωτόκολλα χρησιμοποιούνται πάνω από το πρωτόκολλο WebSocket. Ένα από τα μεγάλα οφέλη του WebSocket, είναι η ικανότητα της διαστρωμάτωσης πρωτόκολλων που χρησιμοποιούνται ευρέως, πάνω από το WebSocket πρωτόκολλο, κάτι το οποίο επιτρέπει να γίνουν σπουδαία πράγματα, όπως να λάβουν χώρα παραδοσιακές εφαρμογές desktop στο διαδίκτυο.

Κατά την αρχική χειραψία σύνδεσης WebSocket, ο πελάτης αποστέλλει την κεφαλίδα Sec-WebSocket-Protocol με το όνομα του πρωτοκόλλου. Ο εξυπηρετητής επιλέγει μηδέν ή ένα πρωτόκολλο και απαντά με την κεφαλίδα Sec-WebSocket-Protocol να περιέχει το ίδιο όνομα με αυτό που ζήτησε ο πελάτης, διαφορετικά κλείνει την σύνδεση. Αυτή η διαδικασία καλείται διαπραγμάτευση πρωτοκόλλου.

Η διαπραγμάτευση πρωτοκόλλου είναι χρήσιμη, προκειμένου να προσδιοριστεί, ποιο πρωτόκολλο ή έκδοση ενός πρωτοκόλλου υποστηρίζεται από έναν δοσμένο WebSocket εξυπηρετητή. Μια εφαρμογή μπορεί να υποστηρίξει πολλαπλά πρωτόκολλα και χρησιμοποιεί την διαπραγμάτευση πρωτοκόλλου για να επιλέξει ποιο πρωτόκολλο θα χρησιμοποιήσει με έναν συγκεκριμένο εξυπηρετητή.

3.2.2 WebSocket Συμβάντα (Events)

Το WebSocket API, ακολουθεί ένα ασύγχρονο μοντέλο προγραμματισμού, κάτι που σημαίνει ότι για όσο διάστημα παραμένει ανοιχτή μια σύνδεση WebSocket, η εφαρμογή απλά ακούει για συμβάντα. Για να αρχίσει η εφαρμογή να ακούει σε συμβάντα, θα πρέπει να προστεθούν μια σειρά προκαθορισμένων συναρτήσεων επανάκλησης(callback) στο αντικείμενο WebSocket που έχει δημιουργηθεί.

Το αντικείμενο WebSocket μπορεί να διεκπεραιώσει τέσσερα διαφορετικά συμβάντα:

- Open
- Message
- Error
- Close

3.2.2.1 Συμβάν Open

Μόλις ο εξυπηρετητής απαντήσει στο αίτημα σύνδεσης WebSocket, πυροδοτείται το συμβάν **Open** και η σύνδεση δημιουργείται. Η προκαθορισμένη συνάρτηση (callback function) που καλείται με την πυροδότηση του συμβάντος Open, είναι η **onopen**. Στο παρακάτω απόσπασμα κώδικα παρουσιάζεται η διαχείριση του συμβάντος open, όταν η σύνδεση WebSocket δημιουργηθεί.

```
ws.onopen = function(e) {
        console.log("Connection open...");
};
```

Από την στιγμή που πυροδοτηθεί το συμβάν open, η χειραψία έχει ολοκληρωθεί, και το WebSocket αντικείμενο είναι έτοιμο να στέλνει και να λαμβάνει δεδομένα. Αν η εφαρμογή πελάτη, αντιληφθεί το συμβάν open, αυτό σημαίνει ότι ο εξυπηρετητής WebSocket, χειρίστηκε επιτυχώς το αίτημα και συμφωνεί να επικοινωνεί με την εφαρμογή.

3.2.2.2 Συμβάν Message

Τα WebSocket μηνύματα περιέχουν τα δεδομένα που έρχονται από το εξυπηρετητή. Το συμβάν **Message** πυροδοτείται όταν λαμβάνεται ένα μήνυμα. Η προκαθορισμένη συνάρτηση(callback function) που καλείται με την πυροδότηση του συμβάντος Message, είναι η **onmessage**. Στο

παρακάτω απόσπασμα κώδικα παρουσιάζεται η διαχείριση του συμβάντος message, λαμβάνοντας ένα μήνυμα κειμένου.

```
ws.onmessage = function(e) {
	if(typeof e.data === "string"){
		console.log("String message received", e, e.data);
	} else {
		console.log("Other message received", e, e.data);
	}
};
```

3.2.2.3 Συμβάν Error

Το συμβάν **Error** πυροδοτείται ως απάντηση σε απροσδόκητα σφάλματα επικοινωνίας. Η προκαθορισμένη συνάρτηση(callback function) που καλείται με την πυροδότηση του συμβάντος Error, είναι η **onerror**. Τα σφάλματα προκαλούν επίσης το κλείσιμο της σύνδεσης WebSocket. Αν το συμβάν error γίνει αντιληπτό από την εφαρμογή, τότε περιμένει να ακολουθήσει σύντομα η πυροδότηση του συμβάντος Close. Ο χειριστής συμβάντων του error, είναι ένα κάλο σημείο στο κώδικα για να επιχειρηθεί επανασύνδεση στον εξυπηρετητή και να γίνει διαχείριση των εξευρέσεων που προέρχονται από το αντικείμενο WebSocket. Στο παρακάτω απόσπασμα κώδικα παρουσιάζεται η διαχείριση του συμβάντος error, όταν υπάρχει κάποιο σφάλμα.

```
ws.onerror = function(e) {
	console.log("WebSocket Error: " , e);
	//Προσαρμοσμένη συνάρτηση για τον χειρισμό σφαλμάτων.
	handleErrors(e);
};
```

3.2.2.4 Συμβάν Close

Το συμβάν **Close** πυροδοτείται όταν μια σύνδεση WebSocket τερματίζεται. Η προκαθορισμένη συνάρτηση(callback function) που καλείται με την πυροδότηση του συμβάντος Close, είναι η **onclose.** Μόλις η σύνδεση κλείσει, ο πελάτης και ο εξυπηρετητής, δεν μπορούν πλέον να στέλνουν και να λαμβάνουν μηνύματα. Στο παρακάτω απόσπασμα κώδικα παρουσιάζεται η διαχείριση του συμβάντος Close, όταν τερματίζεται μια σύνδεση.

```
ws.onclose = function(e) {
	console.log("Connection closed", e);
};
```

3.2.3 WebSocket μέθοδοι (Methods)

Ένα αντικείμενο WebSocket έχει δύο μεθόδους, την send() και close().

3.2.3.1 Μέθοδος Send

Μόλις δημιουργηθεί μια σύνδεση ταυτόχρονης αμφίδρομης επικοινωνίας (full - duplex, bidirectional) μεταξύ πελάτη και εξυπηρετητή, τότε είναι δυνατή η κλήση της μεθόδου **send()**, για όσο διάστημα η σύνδεση παραμένει ανοιχτή, δηλαδή αφού κληθεί η callback συνάρτηση onopen και πριν κληθεί η callback συνάρτηση onclose. Η μέθοδος send() χρησιμοποιείται για την αποστολή μηνυμάτων από τον πελάτη προς τον εξυπηρετητή. Μετά την αποστολή ενός ή περισσότερων μηνυμάτων, η σύνδεση μπορεί να παραμείνει ανοιχτή ή να τερματιστεί με την κλήση της μεθόδου close().Στο παρακάτω απόσπασμα κώδικα παρουσιάζεται η αποστολή μηνύματος κείμενου.

```
ws.send("Hello WebSocket!");
```

Η μέθοδος send() μεταδίδει τα δεδομένα, όταν η σύνδεση είναι ανοικτή. Εάν η σύνδεση

δεν είναι διαθέσιμη ή κλειστή, τότε εγείρει μια εξαίρεση για μη έγκυρη κατάσταση σύνδεσης. Πρώτου λάβει μέρος η αποστολή του πρώτου μηνύματος από ένα νέο-κατασκευασμένο αντικείμενο WebSocket, θα πρέπει πρώτα ο πελάτης να αντιληφθεί το συμβάν open (Βλέπε παρακάτω απόσπασμα κώδικα).

```
var ws = new WebSocket("ws://echo.websocket.org")

ws.onopen = function(e) {
        ws.send("Initial data");
}
```

3.2.3.2 Μέθοδος Close

Για να κλείσει μια σύνδεση WebSocket ή για να τερματιστεί η προσπάθεια σύνδεσης, χρησιμοποιείται η μέθοδος close(). Εάν η σύνδεση είναι ήδη κλειστή τότε η μέθοδος close() δεν κάνει καμία ενέργεια. Μετά την κλήση της μεθόδου close(), παύει να είναι πλέον δυνατή η αποστολή ή λήψη δεδομένων στο κλειστό WebSocket. Παρακάτω φαίνεται η κλήση της μεθόδου.

```
ws.close();
```

Η μέθοδος close(), δέχεται δύο προαιρετικές παραμέτρους, έναν αριθμητικό κωδικό κατάστασης και ένα λεκτικό. Ορίζοντας αυτές τις παραμέτρους, μεταδίδεται στον εξυπηρετητή, ο λόγος για τον οποίο ο πελάτης τερμάτισε την σύνδεση (Βλέπε παρακάτω απόσπασμα κώδικα).

```
ws.close(1000, "Closing normally");
```

3.2.4 Ιδιότητες Αντικειμένου WebSocket

Υπάρχουν πολλές ιδιότητες του αντικειμένου WebSocket, που μπορούν να χρησιμοποιηθούν προκειμένου να παράσχουν περισσότερες πληροφορίες για την κατάσταση του αντικειμένου.

3.2.4.1 Ιδιότητα readyState

Το αντικείμενο WebSocket, αναφέρει την κατάσταση της σύνδεσης μέσω της read-only ιδιότητας readyState. Η τιμή αυτής της ιδιότητας αλλάζει αυτόματα, ανάλογα με την κατάσταση της σύνδεσης και παρέχει χρήσιμες πληροφορίες σχετικά με την σύνδεση WebSocket. Στο πίνακα 3 περιγράφονται οι τέσσερις διαφορετικές τιμές που μπορεί να λάβει η ιδιότητα readyState, για να περιγράψει την κατάσταση της σύνδεσης. Όταν ένα αντικείμενο WebSocket πρώτο - δημιουργείται, η ιδιότητα readyState έχει την τιμή 0.

Σταθερά	Τιμή	Κατάσταση
WebSocket.CONNECTING	0	Η σύνδεση είναι σε εξέλιξη, αλλά δεν έχει ολοκληρωθεί.
WebSocket.OPEN	1	Η σύνδεση έχει εγκαθιδρυθεί. Τα μηνύματα μπορεί να ρέουν μεταξύ του πελάτη και του εξυπηρετητή.
WebSocket.CLOSING	2	Η σύνδεση που διέρχεται από το χειραψία κλεισίματος.
WebSocket.CLOSED	3	Η σύνδεση έχει κλείσει ή δεν μπορεί να ανοιχθεί.

Πίνακας 3: Τιμές της ιδιότητας readyState

3.2.4.2 Ιδιότητα bufferedAmount

Κατά τον σχεδιασμό της εφαρμογής, μπορεί να χρειάζεται να ελεγχθεί ο όγκος δεδομένων που υπάρχει στην ενδιάμεση προσωρινή μνήμη (buffer), για μετάδοση στον εξυπηρετητή, ειδικά εάν η εφαρμογή πελάτη, μεταδίδει μεγάλο όγκο δεδομένων. Ακόμα και αν η κλήση της μεθόδου send() είναι στιγμιαία, στην πραγματικότητα, η μετάδοση των δεδομένων στο διαδίκτυο δεν είναι το ίδιο στιγμιαία. Ο φυλλομετρητής θα αποθηκεύσει τα εξερχόμενα δεδομένα στην προσωρινή μνήμη για λογαριασμό της εφαρμογής πελάτη, ώστε να μπορεί να κληθεί η μέθοδος send() όσες φορές και με όσα δεδομένα είναι επιθυμητό. Αν θέλουμε να γνωρίζουμε, πόσο γρήγορα διαρρέουν τα δεδομένα από την προσωρινή μνήμη στο διαδίκτυο, το αντικείμενο WebSocket μπορεί να μας γνωρίσει το μέγεθος της προσωρινής μνήμης. Με την ιδιότητα bufferedAmount, μπορεί να ελεγχθεί ο αριθμός των bytes που έχουν μπει στην ουρά αναμονής και δεν έχουν μεταδοθεί ακόμα στον εξυπηρετητή.

3.2.4.3 Ιδιότητα Protocol

Η ιδιότητα Protocol, περιέχει το όνομα του πρωτοκόλλου που έχει επιλεγεί από τον εξυπηρετητή WebSocket κατά τη διαδικασία εγκαθίδρυσης σύνδεσης. Με άλλα λόγια η ιδιότητα protocol λέει ποιο πρωτόκολλο χρησιμοποιεί σε μια συγκεκριμένη WebSocket σύνδεση. Έως ότου ολοκληρωθεί η εγκαθίδρυση της σύνδεσης, η τιμή της ιδιότητας protocol αναφέρει ένα κενό λεκτικό, και παραμένει ως έχει αν ο εξυπηρετητής δεν επιλέγει ένα από τα πρωτόκολλα που προσφέρονται από τον πελάτη.

3.3 Χρησιμοποιούμενες Τεχνολογίες

Σε αυτή την ενότητα γίνεται μια συνοπτική παρουσίαση των τεχνολογιών που χρησιμοποιήθηκαν στην παρούσα εργασία για την ανάπτυξη μιας εφαρμογής κοινωνικής δικτύωσης. Οι τεχνολογίες που παρουσιάζονται στις επόμενες παραγράφους είναι:

- PHP πλαίσιο εργασίας Zend Framework
- Μη σχεσιακή (NoSql) βάση δεδομένων MongoDB
- PHP πλαίσιο εργασίας Doctrine ODM MongoDB
- Μη σχεσιακή (NoSql), in-memory βάση δεδομένων Redis
- Λογισμικό Node.js σε συνεργασία με την βιβλιοθήκη Socket.IO

3.3.1 Zend Framework;

Στην παρούσα διπλωματική εργασία, επιλέχθηκε το συγκεκριμένο πλαίσιο εργασίας ως ο βασικός πυρήνας για την υλοποίηση της εφαρμογής. Ουσιαστικά θα αποτελέσει το μέρος εκείνο της εφαρμογής που θα δέχεται και θα εξυπηρετεί HTTP αιτήματα εκτελώντας τον κώδικα επιχειρησιακής λογικής και πρόσβασης δεδομένων που θα γραφτεί υπό το συγκεκριμένο πλαίσιο εργασίας.

Το πρότυπο σχεδιασμού «Μοντέλο – Προβολή - Ελεγκτής» (Model - View - Controller ή MVC) που ενσωματώνει το Zend Framework, αποτελεί ένα κομψό τρόπο σχεδίασης και ανάπτυξης που ταιριάζει πολύ ωραία στις συνοριακές κλάσεις, στις ελεγκτικές κλάσεις και στις κλάσεις οντοτήτων που εμφανίζονται στην διαδικασία ICONIX που έχει επιλεχθεί για την παρούσα διπλωματική ως μεθοδολογία σχεδίασης και ανάπτυξης όπως παρουσιάζεται στο πέμπτο κεφάλαιο.

Επίσης, ένας ελκυστικός λόγος επιλογής του συγκεκριμένου πλαισίου εργασίας, αποτελεί η υποστήριξή αυτόματης κλήσης κλάσεων (autoloading) που ενσωματώνει, παρέχοντας με αυτό το τρόπο την δυνατότητα ομοιογενούς συνεργασίας με πρόσθετα πλαίσια εργασίας που χρησιμοποιούνται στην εφαρμογή της παρούσας διπλωματικής, όπως το πλαίσιο εργασίας για την αντιστοίχηση αντικειμένων σε έγγραφα «Doctrine ODM» για την αντικειμενοστραφής επικοινωνία με την βάση δεδομένων MongoDB και τη βιβλιοθήκη Predis για την επικοινωνία με την Redis βάση δεδομένων.

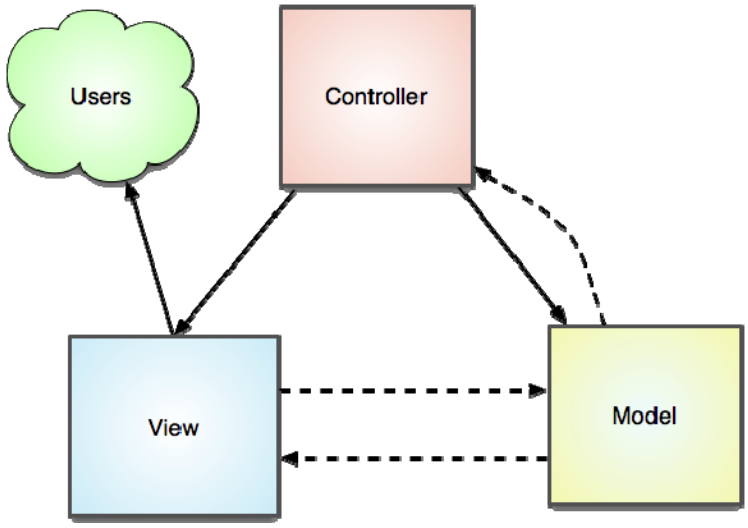
Γενικότερα, τα σχεδιαστικά πρότυπα (Front Controller, Registry, Adapter, Decorator, Strategy, Singleton, Dependency Injection) που διέπουν το Zend Framework επιτρέπουν τη συγγραφή δομημένου, επεκτάσιμου και συντηρήσιμου κώδικα.

Ορισμένες άλλες δυνατότητες που προσφέρει το Zend Framework και οι οποίες συνέβαλαν με τη σειρά τους στην επιλογή του, είναι ή ασφάλεια (sql injection), τα φιλικά URL (SEO Optimization) και η χρήση plugins που επιτρέπουν την τροποποίηση ή επέκταση του Zend Framework χωρίς να «πειραχθεί» ο κύριος κώδικας του.

Τέλος, το γεγονός και μόνο ότι το Zend Framework αποτελεί δημιούργημα του κορυφαίου προμηθευτή λογισμικού Zend, όντας ταυτόχρονα ένας εκ των δημιουργών της PHP, συνιστά σημαντικό λόγο επιλογής του, με πάρα πολύ καλή βιβλιογραφία και πολλά βοηθήματα στο διαδίκτυο να είναι διαθέσιμα.

3.3.1.1 Τι είναι το Zend Framework;

Το Zend Framework είναι ένα αντικειμενοστραφές (Object-Oriented) πλαίσιο εργασίας (Framework), που χρησιμοποιείται για την ανάπτυξη εφαρμογών διαδικτύου και το οποίο εκτελείται στην πλευρά του εξυπηρετητή μέσα στο περιβάλλον της PHP. Συχνά το Zend Framework καλείται «βιβλιοθήκη συστατικών» διότι έχει πολλά συστατικά, χαλαρής ζεύξης (loosely coupled), που μπορούν να χρησιμοποιηθούν σχεδόν ανεξάρτητα. Επίσης, παρέχει μια προηγμένη υλοποίηση του προτύπου Μοντέλο – Προβολή - Ελεγκτής (Model - View - Controller), η οποία μπορεί να χρησιμοποιηθεί για τη δημιουργία της βασικής δομής της εφαρμογής. Ο περισσότερος κώδικας μιας εφαρμογής, υπάγεται στις ακόλουθες τρεις κατηγορίες: Παρουσίαση (Presentation), Επιχειρησιακή Λογική (Business Login), Πρόσβαση Δεδομένων (Data Access). Η αρχιτεκτονική MVC μοντελοποιεί αυτό τον διαχωρισμό πολύ καλά. Το τελικό αποτέλεσμα είναι ότι ο κώδικας παρουσίασης είναι συγκεντρωμένος σε ένα σημείο της εφαρμογής και αντίστοιχα το ίδιο συμβαίνει και με τον κώδικα επιχειρησιακής λογικής και πρόσβασης δεδομένων.



3-4: Αρχιτεκτονική MVC

Μοντέλο (Model) - Σε αυτό το σημείο της εφαρμογής ορίζεται η βασική λειτουργικότητα πίσω από ένα σύνολο αφαιρέσεων. Ρουτίνες για την πρόσβαση δεδομένων και κώδικας επιχειρησιακής λογικής μπορεί να οριστεί στο μοντέλο.

Προβολή (View) - Οι προβολές καθορίζουν τι ακριβώς παρουσιάζεται στον τελικό χρήστη. Συνήθως οι ελεγκτές (Controllers) στέλνουν δεδομένα στις προβολές, ώστε να απεικονιστούν με μια μορφοποίηση. Οι προβολές, επίσης συλλέγουν δεδομένα από τον χρήστη. Οι προβολές είναι το μέρος της εφαρμογής που πιθανόν να υπάρχει ο περισσότερος HTML κώδικας.

Ελεγκτής (Controller) - Οι ελεγκτές συνδέουν ολόκληρο το πρότυπο σχεδιασμού (design pattern) MVC. Οι ελεγκτές διαχειρίζονται τα μοντέλα, αποφασίζουν ποια προβολή θα εμφανίσουν με βάση το αίτημα του χρήστη και άλλων παραγόντων, διοχετεύουν τα δεδομένα που χρειάζεται η κάθε προβολή ή στέλνουν τον έλεγχο σε έναν τελείως διαφορετικό ελεγκτή.

3.3.1.2 Η Δομή Εφαρμογής Zend Framework

Μια εφαρμογή Zend Framework αποτελείται από πολλούς καταλόγους(directories). Αυτό βοηθά ώστε να διασφαλιστεί ότι τα διάφορα μέρη της εφαρμογής, βρίσκονται σε ξεχωριστά σημεία. Στην παρακάτω εικόνα διακρίνεται μια τυπική δομή καταλόγων μιας Zend Framework εφαρμογής.

Υπάρχουν τέσσερις πολύ σημαντικοί κατάλογοι στο ανώτερο επίπεδο:

- application
- library
- test
- public

3.3.1.2.1 Κατάλογος application

Ο κατάλογος application περιέχει όλο τον κώδικα που απαιτείται για να τρέξει η εφαρμογή και δεν είναι άμεσα προσβάσιμος από τον εξυπηρετητή. Προκειμένου να υπάρχει σαφής διαχωρισμός της επιχειρησιακής λογικής, των προβολών και λογικής ελέγχου, υπάρχουν ακόμη τρείς κατάλογοι - controllers, models, views - , εντός του καταλόγου application. Επιπρόσθετοι κατάλογοι επίσης μπορούν να δημιουργηθούν αν απαιτηθεί, όπως ο κατάλογος configs.

3.3.1.2.2 Κατάλογος library

Σε μια εφαρμογή Zend Framework, το ίδιο το πλαίσιο εργασίας βρίσκεται στον κατάλογο library. Επίσης στον ίδιο κατάλογο, μπορεί να αποθηκευτούν επιπλέον προσαρμοσμένες βιβλιοθήκες, όπως ORM βιβλιοθήκες, π.χ. Doctrine.

3.3.1.2.3 Κατάλογος tests

Στον κατάλογο tests υπάρχουν όλες οι μονάδες ελέγχου που έχουν γραφτεί. Οι μονάδες ελέγχου χρησιμοποιούνται για να διασφαλίσουν ότι ο κώδικας εκτελείται σωστά, όσο μεγαλώνει και αλλάζει, καθ' όλη την διάρκεια της ανάπτυξης της εφαρμογής.

3.3.1.2.4 Κατάλογος public

Για να βελτιωθεί η ασφάλεια μιας διαδικτυακής εφαρμογής, ο εξυπηρετητής θα πρέπει να έχει άμεση πρόσβαση μόνο στα αρχεία που χρειάζεται να εξυπηρετήσει. Δεδομένου ότι το Zend Framework χρησιμοποιεί το πρότυπο σχεδιασμού Front Controller, όλα τα HTTP αιτήματα σε ένα μοναδικό αρχείο, που συνήθως ονομάζεται index.php. Αυτό το αρχείο είναι το μοναδικό php αρχείο, στο οποίο χρειάζεται να έχει πρόσβαση ο εξυπηρετητής και είναι αποθηκευμένο στον κατάλογο public. Άλλα, κοινά αρχεία, τα οποία είναι άμεσα προσβάσιμα είναι οι εικόνες, αρχεία CSS και αρχεία JavaScript, τα οποία είναι αποθηκευμένα σε ξεχωριστούς υπό- καταλόγους μέσα στον κατάλογο public.

3.3.1.3 Zend Framework & MVC

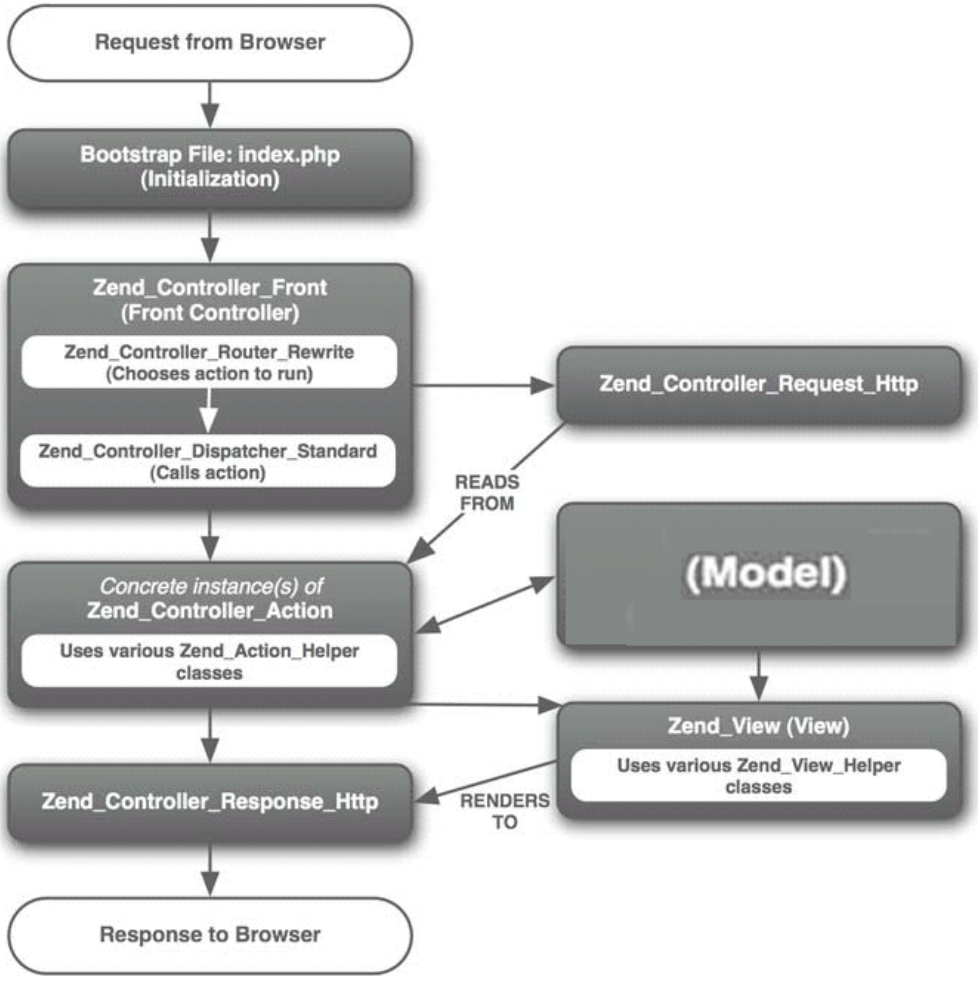
Ενώ φαίνεται να υπάρχουν πολλοί διαφορετικοί τρόποι δρομολόγησης αιτήσεων στον κώδικα της δικτυακής εφαρμογής, μπορούν να ταξινομηθούν σε δύο κατηγορίες: ελεγκτές σελίδας (page controllers) και εμπρόσθιοι ελεγκτές (front controllers). Ένας ελεγκτής σελίδα χρησιμοποιεί ξεχωριστά αρχεία για κάθε σελίδα που απαρτίζουν την ιστοσελίδα και είναι παραδοσιακά το πώς οι περισσότεροι PHP ιστοσελίδες έχουν κατασκευαστεί. Αυτό σημαίνει ότι ο έλεγχος της εφαρμογής είναι αποκεντρωμένος σε πολλά διαφορετικά αρχεία, γεγονός που μπορεί να οδηγήσει σε επαναλαμβανόμενο κώδικα ή, ακόμη χειρότερα, σε επαναλαμβανόμενο και ελαφρώς παραλλαγμένο κώδικα, οδηγώντας σε θέματα, όπως η απώλεια συνεδρίας, όταν ένα από τα αρχεία δεν εκτελεί την session_start () .

Ο εμπρόσθιος ελεγκτής (front controller), από την άλλη πλευρά, συγκεντρώνει όλα τα αιτήματα σε ένα ενιαίο αρχείο, ονόματι συνήθως index.php και το οποίο βρίσκεται στο αρχικό κατάλογο (root directory) της εφαρμογής. Υπάρχουν πολλά πλεονεκτήματα με αυτό τον μηχανισμό, με τα πιο προφανές να είναι ότι υπάρχει λιγότερο επαναλαμβανόμενος κώδικας και είναι ευκολότερο να διακριθούν τα URLs που έχει μια ιστοσελίδα από τον πραγματικό κώδικα που χρησιμοποιείται για να παράγει τις σελίδες. Συνήθως, οι σελίδες εμφανίζονται με δύο επιπλέον GET παραμέτρους που περνιούνται στο index.php αρχείο για να δημιουργήσει τις διευθύνσεις URL όπως index.php?controller=news&action=list για να εμφανίσει μια σελίδα καταλόγου (list).

3.3.1.3.1 Ο Εμπρόσθιος Ελεγκτής (Front Controller) του Zend Framework

Ο κώδικας του εμπρόσθιου ελεγκτή (Front Controller) του Zend Framework είναι κατανεμημένος σε έναν αριθμό κλάσεων (classes) που συνεργάζονται για να παρέχουν μια πιο ευέλικτη λύση στο πρόβλημα της δρομολόγησης ενός αιτήματος στο σωστό σημείο κώδικα.

Η κλάση Zend_Controller_Front είναι το βασικό συστατικό που επεξεργάζεται όλες τις αιτήσεις που λαμβάνονται από την εφαρμογή και μεταβιβάζει την πραγματική εργασία στους ελεγκτές δράσης (action controllers).



3-5: Η αλληλεπίδραση διαφόρων κλάσεων του Zend Framework που χρησιμοποιούνται για να δημιουργήσουν μια εφαρμογή MVC

3.3.1.3.2 Αίτημα (Request)

Το αίτημα ενθυλακώνεται μέσα σε ένα στιγμιότυπο της κλάσης Zend_Controller_Request_Http, το οποίο παρέχει πρόσβαση σε όλο το περιβάλλον αίτησης HTTP. Το περιβάλλον αίτησης HTTP, είναι όλες οι εξωτερικές μεταβλητές που λαμβάνονται από την εφαρμογή μαζί με σχετικές με τον ελεγκτή παραμέτρους, όπως η παράμετρος του ελεγκτή και της δράσης (action).

Το περιβάλλον αίτησης HTTP, περιέχει όλες τις υπέρ - γενικές (super globals) μεταβλητές ($_GET, $_POST, $_COOKIE, $_SERVER και $_ENV), μαζί με την βασική διαδρομή (base path) της εφαρμογής. Ο δρομολογητής τοποθετεί επίσης τα ονόματα μονάδας, ελεγκτή και δράσης στο αντικείμενο αίτησης μόλις τα επεξεργαστεί. Η κλάση Zend_Controller_Request_Http παρέχει την μέθοδο getParam() για να επιτρέπει στην εφαρμογή να συλλέξει τις μεταβλητές αιτήματος και έτσι το υπόλοιπο της εφαρμογής είναι προστατευμένο από μια αλλαγή στο περιβάλλον.

Σε γενικές γραμμές, το αντικείμενο αιτήματος θα πρέπει να αντιμετωπίζεται ως ένα αντικείμενο ανάγνωσης, καθώς οι τιμές που ορίζονται έμμεσα από τον χρήστη δεν πρέπει να αλλάζουν. Δεδομένου αυτού, το αντικείμενο αιτήματος, περιλαμβάνει επίσης, παραμέτρους που μπορεί να τεθούν στην φάση εκκίνησης της εφαρμογής και στην συνέχεια να ανακτηθεί η τιμή αυτών από τις μεθόδους δράσης, εφόσον χρειαστεί. Αυτό μπορεί να χρησιμοποιηθεί για να περαστούν επιπρόσθετες πληροφορίες από τον εμπρόσθιο ελεγκτή στις μεθόδους δράσης, αν απαιτηθεί.

3.3.1.3.3 Δρομολόγηση (Routing)

Δρομολόγηση είναι η διαδικασία καθορισμού ποια ενέργεια ελεγκτή (controller's action) πρέπει να τρέξει προκειμένου να ικανοποιήσει το αίτημα. Αυτό γίνεται την βοήθεια της κλάσης Zend_Controller_Router_Rewrite που υλοποιεί την διεπαφή Zend_Controller_Router_Interface και η οποία θα χειριστεί τις περισσότερες απαιτήσεις δρομολόγησης. Η δρομολόγηση (routing) λειτουργεί παίρνοντας το μέρος της διεύθυνσης URL, που ακολουθεί μετά την βασική διαδρομή και αποσυνθέτοντας το στη συνέχεια σε ξεχωριστές παραμέτρους. Σε μια τυπική URL διεύθυνση, όπως η http://example.com/index.php?controller=news&action=list, η αποσύνθεση γίνεται διαβάζοντας την μεταβλητή πίνακα $_GET και ψάχνει για στοιχεία controller και action. Βέβαια, το Zend Framework ως ένα σύγχρονο πλαίσιο εργασίας αναμένεται να χρησιμοποιεί φιλικά URLs, οπότε το παραπάνω URL παίρνει την μορφή http://example.com/news/list. Σε αυτή την περίπτωση, ο δρομολογητής (router) θα χρησιμοποιήσει την μεταβλητή $_SERVER['REQUEST_URI'] για να προσδιορίσει ποιος ελεγκτής (controller) και δράση (action) έχουν ζητηθεί.

3.3.1.3.4 Προώθηση (Dispatching)

Προώθηση είναι η διαδικασία κλήσης της σωστής μεθόδου, της σωστής κλάσης. Ο τυπικός διαβιβαστής (dispatcher) παρέχει αρκετή λειτουργικότητα για σχεδόν κάθε κατάσταση, άλλα αν απαιτείται κάτι ιδιαίτερο, μπορεί εύκολα να προστεθεί μια νέα λειτουργικότητα στον εμπρόσθιο ελεγκτή. Τα σημεία κλειδιά που ελέγχει ο διαβιβαστής είναι μορφοποίηση του ονόματος της κλάσης ελεγκτή (Class Controller), η μορφοποίηση του ονόματος της μεθόδου δράσης (Action Method) και η κλήση της ίδιας μεθόδους δράση.

Η κλάση Zend_Controller_Dispatcher_Standard είναι το μέρος όπου θα επιβληθούν οι όποιοι κανόνες, όπως ότι το όνομα του ελεγκτή πρέπει να έχει μια συγκεκριμένη μορφοποίηση, π.χ. περιέχει μόνο αλφαβητικούς χαρακτήρες. Η μέθοδος dispatch του διαβιβαστή είναι υπεύθυνη για α) την φόρτωση του αρχείου κλάσης ελεγκτή (controller class file), β) την δημιουργία ενός αντικειμένου της κλάσης και γ) την κλήση της μεθόδου δράσης της κλάσης.

3.3.1.3.5 Δράση (Action)

Η κλάση Zend_Controller_Action είναι μια αφημένη κλάση, την οποία κληρονομούν όλες οι κλάσεις ελεγκτές. Ο διαβιβαστής επιβάλει, ότι οι ελεγκτές δράσης κληρονομούν την κλάση Zend_Controller_Action για να διασφαλίσει ότι μπορεί να αναμένει ορισμένες μεθόδους να είναι διαθέσιμες. Η μέθοδος δράσης περιλαμβάνει ένα στιγμιότυπο αιτήματος για την ανάγνωση παραμέτρων και ένα στιγμιότυπο απάντησης για γράψιμο πάνω σε αυτό.

Οι μέθοδοι δράσης ονομάζονται με το λεκτικό «Action» να ακολουθεί το όνομα της δράσης, συνθέτοντας το ολοκληρωμένο όνομα της μεθόδου δράσης. Έτσι, μία κλάση ελεγκτή, αναμένεται να περιέχει μεθόδους δράσεις όπως indexAction(), viewAction(), editAction(), deleteAction() κ.α. Κάθε μια από αυτές τις διακριτές μεθόδους, εκτελούνται ως απάντηση σε αίτημα συγκεκριμένου URL. Υπάρχουν, ωστόσο, μια σειρά από εργασίες που μερικές φορές χρειάζεται να γίνουν, ανεξάρτητα από ποια μέθοδος δράσης εκτελείται. Η κλάση Zend_Controller_Action παρέχει δύο επίπεδα λειτουργικότητας για να φιλοξενήσει αυτή την απαίτηση: Οι μέθοδοι init() και pre/postDispatch(). Η μέθοδος init() καλείται όταν κατασκευάζεται ένα αντικείμενο της κλάσης ελεγκτή. Αυτό μοιάζει με της λειτουργικότητα που παρέχει ένας τυπικός κατασκευαστής με τη διαφορά ότι δεν δέχεται παραμέτρους και απαιτείται η κλήση της αντίστοιχής μεθόδου της γονικής κλάσης. Οι μέθοδοι preDispatch() και postDispatch() είναι ένα συμπληρωματικό ζεύγος μεθόδων που εκτελούνται πριν και μετά από την κλήση κάθε μεθόδου δράσης. Για μια εφαρμογή όπου μόνο μία μέθοδος δράση εκτελείται σε απάντηση σε ένα αίτημα, δεν υπάρχει διαφορά μεταξύ init() και preDispatch(), καθώς η κάθε μια εκτελείται μόνο μία φορά. Εάν, ωστόσο, η πρώτη μέθοδο δράσης χρησιμοποιεί την _forward() συνάρτηση για να περάσει τον έλεγχο σε μια άλλη μέθοδο δράσης, τότε η preDispatch() θα εκτελεστεί και πάλι, αλλά όχι και η init(). Με αυτό τον τρόπο, θα μπορούσε να χρησιμοποιηθεί η μέθοδος init() για να εξασφαλιστεί ότι μόνο στους διαχειριστές επιτρέπεται η πρόσβαση σε οποιαδήποτε μέθοδο δράσης στον ελεγκτή και η μέθοδος preDispatch() θα μπορούσε να χρησιμοποιηθεί για να θέσει το κατάλληλο υπόδειγμα μορφοποίησης (template) που θα εφαρμοστεί στην προβολή της μεθόδου δράσης.

3.3.1.3.6 Απάντηση (Response)

Ένα στιγμιότυπο της κλάσης Zend_Controller_Response_Http περιέχει τρεις τύπους πληροφοριών : κεφαλίδα (header), κύριο σώμα (body) και εξαιρέσεις (exceptions). Στα πλαίσια της απάντησης, οι κεφαλίδες είναι HTTP κεφαλίδες και όχι κεφαλίδες HTML. Κάθε κεφαλίδα είναι ένας πίνακας που περιέχει όνομα μαζί με την τιμή του και είναι δυνατόν να υπάρχουν δύο κεφαλίδες με το ίδιο όνομα αλλά διαφορετικές τιμές. Η απάντηση, έχει επίσης το κωδικό απόκρισης HTTP (όπως ορίζεται στο RFC 2616) το οποίο αποστέλλεται στον πελάτη στο τέλος της επεξεργασίας.

Το κύριο σώμα (body) της απάντηση, χρησιμοποιείται για να περιέχει οτιδήποτε άλλο πρέπει να σταλεί πίσω στον πελάτη. Για μια εφαρμογή διαδικτύου αυτό σημαίνει οτιδήποτε είναι ορατό κατά την προβολή του πηγαίου κώδικα μιας ιστοσελίδας, από τον φυλλομετρητή.

Οι εξαιρέσεις είναι ένας πίνακας που μπορεί να προστεθεί στην απάντηση καλώντας την μέθοδο setException της κλάσης Zend_Controller_Response_Http και χρησιμοποιείται από τον Zend_Controller_Front για να εξασφαλίσει ότι τα λάθη μέσα στον κώδικά δεν αποστέλλονται στον πελάτη, εκθέτοντας, ενδεχομένως, απόρρητες πληροφορίες που μπορεί να χρησιμοποιηθούν για να θέσουν σε κίνδυνο την εφαρμογή.

3.3.1.3.7 Πρόσθετα (Plug-ins) Εμπρόσθιου Ελεγκτή (Front Controller)

Η αρχιτεκτονική του εμπρόσθιου ελεγκτή προβλέπει ένα σύστημα πρόσθετων που επιτρέπει προσαρμοσμένο κώδικα (custom code) να εκτελεστεί αυτόματα σε ορισμένα σημεία τις διαδικασίας δρομολόγησης (Routing) και προώθησης (Dispatching). Όλα τα πρόσθετα (plug-ins) προέρχονται από την κλάση Zend_Controller_Plugin_Abstract και υπάρχουν έξι μέθοδοι συμβάντων που μπορεί να υπερκαλυφθούν (override) :

1. routeStartup() καλείται λίγο πριν από την εκκίνηση εκτέλεσης του δρομολογητή (router).
2. dispatchLoopStartup() καλείται λίγο πριν από την εκτέλεση του διαβιβαστή (dispatcher).
3. preDispatch() καλείται πριν από την εκτέλεση της μεθόδου δράσης.
4. postDispatch() καλείται μετά την εκτέλεση της μεθόδου δράσης.
5. dispatchLoopShutdown() καλείται αφού πρώρα όλες οι μέθοδοι δράσεις έχουν προωθηθεί.
6. routeShutdown() καλείται αφού η εκτέλεση του δρομολογητή ολοκληρωθεί.

Όπως φαίνεται παραπάνω, υπάρχουν τρία ζεύγη μεθόδων που εκτελούνται σε τρία διαφορετικά σημεία (δρομολόγηση, προώθηση, δράση), τα οποία επιτρέπουν τον όλο και πιο εις βάθος έλεγχο της διαδικασίας εξυπηρέτησης αίτηματος.

3.3.2 MongoDB

Η MongoDB είναι μια ισχυρή, ευέλικτη, επεκτάσιμη, εγγραφοκεντρική (document - oriented) αποθήκη δεδομένων. Συνδυάζει την ικανότητα να υποστηρίζει πολλά από τα πιο χρήσιμα χαρακτηριστικά των σχεσιακών βάσεων δεδομένων, όπως δευτερεύοντα ευρετήρια, ερωτήματα εύρους, και ταξινόμηση. Η MongoDB θα αποτελέσει τη βάση δεδομένων της εφαρμογής της παρούσα διπλωματικής εργασίας.

3.3.2.1 Γιατί MongoDB;

Οι κύριοι λόγοι για τους οποίους επιλέχθηκε να χρησιμοποιηθεί στην παρούσα διπλωματική εργασία είναι ότι :

- Υλοποιεί την ιδέα του ελεύθερου και δυναμικού σχήματος. Δεν απαιτείται να οριστεί η δομή των δεδομένων πριν ξεκινήσει η αποθήκευση, γεγονός που την καθιστά κατάλληλη για την αποθήκευση μη δομημένων δεδομένων.
- Είναι εξαιρετικά επεκτάσιμη. Έρχεται με μεγάλες δυνατότητες για να βοηθήσει να κρατήσει τη βέλτιστη απόδοση, όσο το μέγεθος και η κυκλοφορία των δεδομένων μεγαλώνει, με ελάχιστη ή καμία αλλαγή στο επίπεδο εφαρμογής.

Επίσης, η παρουσία και η χρησιμοποίηση της MongoDB σε πραγματικές εφαρμογές (real-world) την καταστούν, απ' ότι φαίνεται προτιμητέα τεχνολογία έναντι άλλων ανάλογα με την περίπτωση χρήσης της. Παρακάτω, αναφέρονται κάποιες από αυτές τις εφαρμογές.

- Craigslist: Το Craigslist είναι η πιο διάσημη ιστοσελίδα με μικρές αγγελίες. Χρησιμοποιεί τη MongoDB για να αρχειοθετήσει δισεκατομμύρια εγγραφές. Στην αρχή χρησιμοποιούσε τη βάση δεδομένων MySql, την οποία και αντικατέστησε με την MongoDB, γεγονός που επέτρεψε την αλλαγή σχήματος της βάσης χωρίς καθυστερήσεις στην αναβάθμιση και την δυνατότητα κλιμάκωσης με εύκολο τρόπο.
- Foursquare: Το Foursquare είναι μια τοποκεντρική (location based) διαδικτυακή εφαρμογή κοινωνικής δικτύωσης. Αποθηκεύει τη γεωγραφική θέση σημείων ενδιαφέροντος (εστιατόρια, καφετέριες, κ.α.) και καταγράφει όταν οι χρήστες

επισκέπτονται αυτά τα μέρη μαζί με τα σχόλια τους. Χρησιμοποιεί τη MongoDB για την αποθήκευση των σημείων και των πληροφοριών των χρηστών.

- CERN: Το διάσημο εργαστήριο φυσικής σωματιδίων, χρησιμοποιεί την MongoDB ώς μνήμη συσσωμάτωσης (aggregation cache) για το πείραμα του μεγάλου αδρονικού επιταχυντή. Τα αποτελέσματα για τα δαπανηρά ερωτήματα συσσωμάτωσης, που εκτελούνται σε τεράστιες ποσότητες δεδομένων, αποθηκεύονται στη MongoDB για μελλοντική χρήση.

3.3.2.2 Βασικά χαρακτηριστικά

3.3.2.2.1 Εγγράφο (document)

Στην καρδιά της MongoDB είναι η έννοια του εγγράφου (document). Ένα έγγραφο είναι η βασική μονάδα δεδομένων για την MongoDB, σχεδόν ισοδύναμη με μια γραμμή (row) σε ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων, αλλά πολύ πιο εκφραστικό. Στην mongoDB, τα έγγραφα αναπαριστώνται ως αντικείμενα παρόμοια με JSON, όπως φαίνεται στο παρακάτω παράδειγμα εγγράφου.

```
{"greeting" : "Hello, world!"}
```

3.3.2.2.2 Συλλογές (Collections)

Τα έγγραφα στην MongoDB μπορούν να οργανωθούν σε συλλογές (Collections). Μια συλλογή (collection) μπορεί να θεωρηθεί ανάλογη με έναν πίνακα σε ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων.

Οι συλλογές είναι ελεύθερης σχεδίασης (schema-free), κάτι που σημαίνει ότι τα έγγραφα εντός μιας ενιαίας συλλογής, μπορεί να έχουν διαφορετικό σχήμα μεταξύ τους. Για παράδειγμα, και τα δύο ακόλουθα έγγραφα μπορούν να αποθηκεύονται σε μια ενιαία συλλογή:

```
{"greeting" : "Hello, world!"}
{"foo" : 5}
```

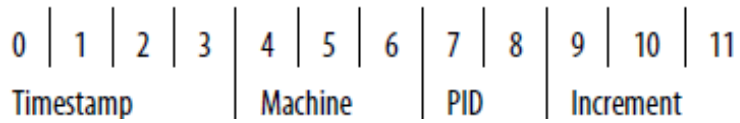
3.3.2.2.3 _id και ObjectId

Κάθε έγγραφο (document) που είναι αποθηκευμένο στη MongoDB πρέπει να έχει ένα "_id" κλειδί. Η τιμή του "_id" κλειδιού μπορεί να είναι οποιουδήποτε τύπου, αλλά προεπιλεγμένος τύπος είναι ο OBJECTID. Σε μια συλλογή (collection) κάθε έγγραφο πρέπει να έχει μοναδική τιμή για "_id", η οποία εξασφαλίζει ότι κάθε έγγραφο στη συλλογή είναι μοναδικό.

ObjectId είναι ο προκαθορισμένος τύπος του κλειδιού _id. Έχει σχεδιαστεί για να είναι ελαφρύ, ενώ εξακολουθεί να είναι εύκολο να παραχθεί με τρόπο καθολικά μοναδικό σε διαφορετικές μηχανές. Αυτός είναι ο κύριος λόγος για τον οποίο η MongoDB χρησιμοποιεί ObjectIds σε αντίθεση με κάτι πιο παραδοσιακό, όπως το αυτόματο πρωτεύων κλειδί (primary key) αυτόματης αρίθμησης (auto - increment). Είναι δύσκολο και χρονοβόρο να γίνει συγχρονισμός πρωτεύονταν κλειδιών αυτόματης αρίθμησης σε πολλούς εξυπηρετητές. Επειδή η MongoDB σχεδιάστηκε από την αρχή να είναι μια κατανεμημένη βάση δεδομένων, το να διαχειρίζεται πολλούς κόμβους είναι ένα σημαντικό στοιχείο. Ο τύπος OBJECTID, είναι εύκολο να παραχθεί σε ένα κοινόχρηστο (sharded) περιβάλλον.

Ο τύπος ObjectId χρησιμοποιεί 12 bytes αποθήκευσης, τα οποία δίνουν μια παράσταση συμβολοσειράς των 24 δεκαεξαδικών ψηφίων: 2 ψηφία για κάθε byte. Αυτό έχει ως αποτέλεσμα να φαίνονται μεγαλύτερα από ό, τι είναι. Εάν δημιουργηθούν διαδοχικά πολλαπλά νέα ObjectIds, μόνο τα τελευταία ψηφία αλλάζουν κάθε φορά. Επιπλέον, ένα ζευγάρι των ψηφίων στη μέση των ObjectIds θα αλλάξει (αν υπάρχει ένα χρονικό διάστημα δημιουργίας των ObjectIds). Αυτό είναι

λόγω του τρόπου με τον οποίο δημιουργούνται τα ObjectIds. Τα 12 bytes ενός ObjectId δημιουργούνται ως ακολούθως:

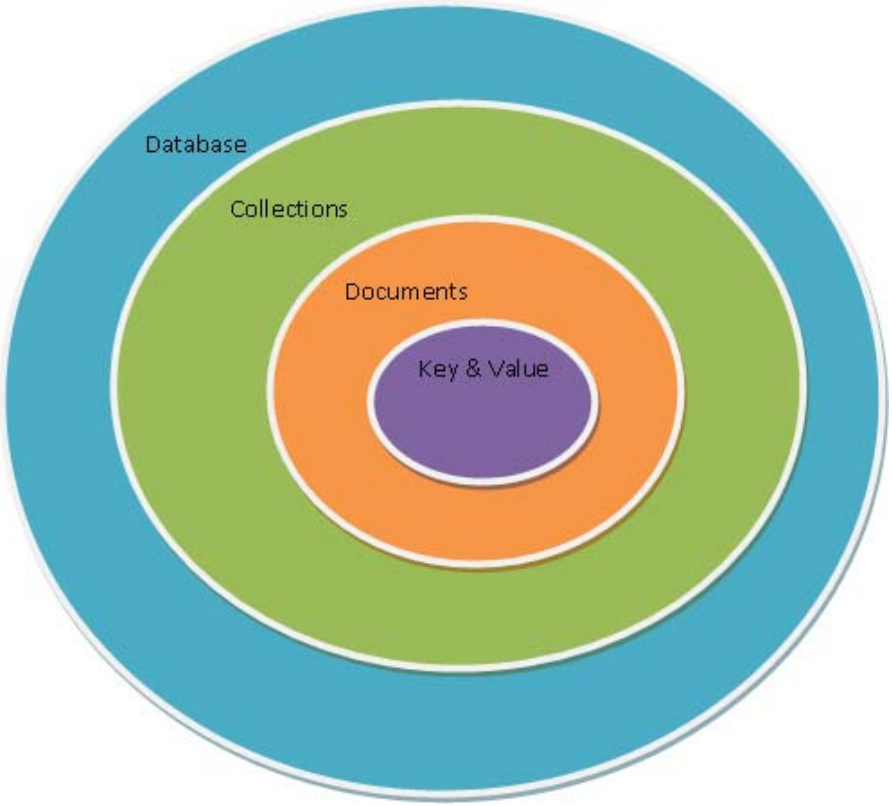


Τα πρώτα τέσσερα bytes είναι η χρονική σήμανση (timestamp) δημιουργίας ενός εγγράφου, σε δευτερόλεπτα. Τα επόμενα τρία bytes ενός ObjectId είναι ένα μοναδικό αναγνωριστικό της μηχανής στην οποία δημιουργήθηκε. Με τη συμπερίληψη αυτών των bytes, διασφαλίζεται ότι οι διάφορες μηχανές δεν θα δημιουργήσουν ίδια ObjectIds, προκαλώντας σύγκρουση (collision). Για την παροχή μοναδικότητας μεταξύ διαφόρων διαδικασιών που παράγουν ObjectIds ταυτόχρονα σε ένα μοναδικό μηχάνημα, τα επόμενα δύο bytes λαμβάνονται από το αναγνωριστικό της διαδικασίας (PID) που παράγει το ObjectId.

Αυτά τα πρώτα εννέα bytes ενός ObjectId εγγυώνται τη μοναδικότητά του σε επίπεδο μηχανημάτων και διαδικασιών για ένα δευτερόλεπτο. Τα τελευταία τρία bytes είναι απλά ένας αυτόματα αυξανόμενος μετρητή που είναι υπεύθυνος για την μοναδικότητα μέσα σε ένα δευτερόλεπτο σε μια διαδικασία.

3.3.2.2.4 Βάσεις Δεδομένων (Databases)

Εκτός από την ομαδοποίηση εγγράφων σε μια συλλογή, η MongoDB ομαδοποιεί επίσης τις συλλογές σε βάσεις δεδομένων. Ένα μόνο στιγμιότυπο της MongoDB μπορεί να φιλοξενήσει διάφορες βάσεις δεδομένων, κάθε μια από τις οποίες μπορεί να θεωρηθεί ως εντελώς ανεξάρτητη. Μια βάση δεδομένων έχει τα δικά τις δικαιώματα, και κάθε βάση δεδομένων είναι αποθηκευμένη σε ξεχωριστό αρχείο στο δίσκο.



3.3.2.2.5 BSON

BSON, είναι η δυαδική μορφή που χρησιμοποιείται για την αναπαράσταση των εγγράφων στην MongoDB. Η μορφή BSON δρα τόσο ως μορφή αποθήκευσης όσο και μορφή εντολής: όλα τα έγγραφα αποθηκεύονται στο δίσκο με την μορφή BSON και όλα τα ερωτήματα και οι εντολές καθορίζονται χρησιμοποιώντας BSON έγγραφα.

3.3.2.2.6 Κέλυφος Εντολών (Shell) της MongoDB

Η MongoDB έρχεται με ένα JavaScript κέλυφος, που επιτρέπει την αλληλεπίδραση με ένα στιγμιότυπο MongoDB από τη γραμμή εντολών. Το κέλυφος είναι πολύ χρήσιμο για την εκτέλεση λειτουργιών διαχείρισης, την επιθεώρηση ενός εκτελούμενου στιγμιότυπου, ή απλά για δοκιμές. Το mongo κέλυφος είναι ένα κρίσιμο εργαλείο για τη χρήση της MongoDB και η πραγματική δύναμη του έγκειται στο γεγονός ότι είναι επίσης ένα αυτόνομο πρόγραμμα πελάτη MongoDB.

Κατά την εκκίνηση, το κέλυφος συνδέεται με την προεγκατεστημένη βάση δεδομένων test στον MongoDB εξυπηρετητή και εκχωρεί αυτήν τη σύνδεση στην καθολική μεταβλητή db. Αυτή η μεταβλητή είναι το πρωταρχικό σημείο πρόσβασης στην MongoDB μέσα από το κέλυφος.

3.3.2.2.7 Αιτήματα και Συνδέσεις

Για κάθε σύνδεση με έναν εξυπηρετητή MongoDB (εξυπηρετητής στον οποίο εκτελείται η MongoDB), η βάση δεδομένων δημιουργεί μια ουρά για τα αιτήματα αυτής σύνδεσης. Όταν ο πελάτης στέλνει ένα αίτημα, θα τοποθετηθεί στο τέλος της ουράς της σύνδεσης. Οποιεσδήποτε μεταγενέστερες αιτήσεις πάνω από την ίδια σύνδεση, θα εξυπηρετηθούν μετά λειτουργία εξαγωγής στοιχείου από την ουρά (enqueued). Έτσι, μία μόνο σύνδεση, έχει στιγμιότυπο

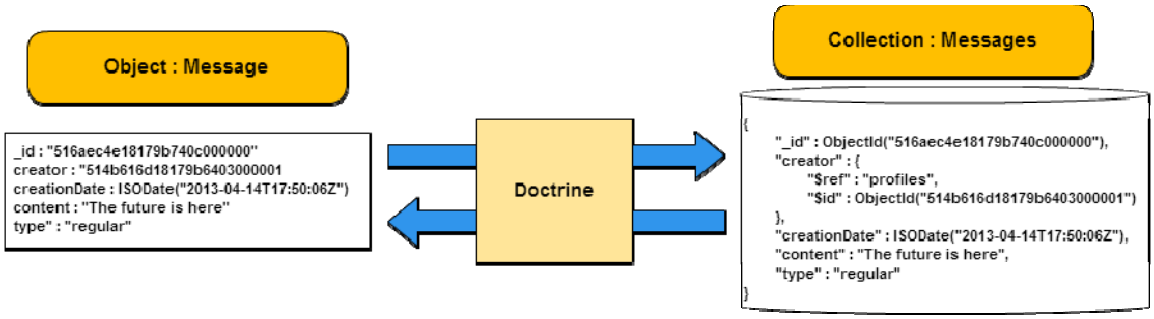
(snapshot) της βάσης δεδομένων από όπου μπορεί να διαβάζει και να γράφει. Για παράδειγμα, αν ανοιχτούν δύο κελύφη, θα υπάρχουν δύο συνδέσεις στη βάση δεδομένων. Αν εκτελεστεί μια εντολή εισαγωγής στο ένα κέλυφος, τότε η αμέσως επόμενη εκτέλεση ενός ερωτήματος στο άλλο κέλυφος δεν θα επιστρέφει το νέο εισαχθέν έγγραφο.

3.3.3 Doctrine MongoDB ODM (Object Document Mapper)

Μία από τις πιο κοινές και ενδιαφέρουσες εργασίες για μια εφαρμογή είναι η αποθήκευση και η ανάκτηση πληροφοριών σε και από μια βάση δεδομένων. Για αυτό το λόγο μέσα στο πλαίσιο εργασίας του Zend ενσωματώθηκε για τις ανάγκες της εφαρμογής και το πλαίσιο εργασίας Doctrine MongoDB ODM .

Το Doctrine αποτελεί μια βιβλιοθήκη της οποίας μοναδικός στόχος είναι να παρέχει ισχυρά εργαλεία στον σχεδιαστή, για να κάνει αυτές τις εργασίες πιο εύκολα. Ποιο συγκεκριμένα για την αποθήκευση δεδομένων στην MongoDB, χρησιμοποιείται η βιβλιοθήκη Doctrine MongoDB ODM. Το Doctrine επιτρέπει την εργασία με την MongoDB με ένα πολύ πιο ενδιαφέροντα τρόπο, από το να φορτώνονται απλά δεδομένα σε διάφορα σημεία της εφαρμογής σε πίνακες. Αντ 'αυτού, το Doctrine επιτρέπει να αποθηκεύονται (persist) ολόκληρα αντικείμενα στην MongoDB και να φορτώνει (fetch) ολόκληρα αντικείμενα από την MongoDB. Αυτό λειτουργεί με τον μηχανισμό χαρτογράφησης (mapping) μιας κλάσης PHP και των ιδιοτήτων της σε καταχωρήσεις μιας MongoDB συλλογής (collection).

Για να γνωρίζει το Doctrine, πως θα χαρτογραφήσει (map) μια κλάση και τις ιδιότητες τις, στην MongoDB, θα πρέπει να δημιουργηθούν κάποια μεταδεδομένα (metadata). Αυτά τα μεταδεδομένα μπορούν να καθοριστούν με διαφορετικές μορφές, συμπεριλαμβανομένων, αρχείων YAML, XML ή απευθείας στο εσωτερικό της κλάσης μέσω σημασιολογικών επισημειώσεων (annotations), με την τελευταία να έχει εφαρμοστεί στην εφαρμογή της παρούσας διπλωματικής (Βλέπε Παράρτημα Α).



3.3.4 Redis

Redis, είναι μια πάρα πολύ γρήγορη μη σχεσιακή (NoSql) βάση δεδομένων, προσωρινής αποθήκευσης (in-merory), η οποία αποθηκεύει την χαρτογράφηση κλειδιών σε πέντε διαφορετικούς τύπους δεδομένων (String, Lists, Hashes, Sets, Sorted Sets). Η Redis, υποστηρίζει προσωρινή (in-memory) και μόνιμη (persistent) αποθήκευση δεδομένων, δυνατότητα διατήρησης αντίγραφων πληροφοριών (replication) για να κλιμακώσει της απόδοση ανάγνωσης (reading) και δυνατότητα διαμερισμού (Sharding) πληροφοριών για να κλιμακώσει την απόδοση εγγραφής (writing).

Επίσης η Redis έχει εγγενή (native) υποστήριξη του πρότυπου Δημοσιεύσεων/Συνδρομών (Publish/subscribe) (Citrusbyte, n.d.) . Αυτός είναι και ο λόγος που χρησιμοποιείται στην παρούσα διπλωματική εργασία, καθώς θα αποτελέσει το σύστημα ειδοποιήσεων για την λήψη νέων μηνυμάτων.

Το πρότυπο Δημοσιεύσεων/Συνδρομών ορίζει τον τρόπο με τον οποίο υποψήφιοι συνδρομητές εγγράφονται σε κανάλια μετάδοσης μηνυμάτων που ταιριάζουν σε ένα συγκεκριμένο πρότυπο, καθώς επίσης, ορίζει και τον τρόπο με τον οποίο ο αποστολέα ή πομπός στέλνει μηνύματα σε ένα σύννεφο μηνυμάτων. Όταν ένα μήνυμα εισέλθει στο σύννεφο, οι πελάτες - συνδρομητές που γράφτηκαν σε μηνύματα αυτού του είδους θα λάβουν το μήνυμα. Με αυτό τον τρόπο, το πρότυπο Δημοσιεύσεων/Συνδρομών επιτρέπει οι αποστολείς και οι συνδρομητές να είναι χαλαρά συζευγμένοι (loosely coupled) - δεν χρειάζεται να γνωρίζει ο ένας τον άλλον, αρκεί να είναι σε θέση να στέλνουν μηνύματα σε ένα συγκεκριμένο πρότυπο και να λαμβάνουν μηνύματα που υπακούουν σε ένα συγκεκριμένο πρότυπο.

Η διαδικασία εγγραφής σε ένα κανάλι θα πραγματοποιείται στο WebSocket εξυπηρετητή με την υποστήριξη της Node.js, ενώ η διαδικασία δημοσίευσης θα πραγματοποιείται στον HTTP εξυπηρετητή μέσα στο πλαίσιο εργασίας Zend με την χρήστη της βιβλιοθήκης Predis.

Η Redis έχει άμεση υποστήριξη του πρότυπου Δημοσιεύσεων/Συνδρομών, που σημαίνει ότι επιτρέπει στους πελάτες να εγγράφονται σε συγκεκριμένα κανάλια που ταιριάζουν σε ένα συγκεκριμένο πρότυπο, και να δημοσιεύουν μηνύματα σε ένα συγκεκριμένο κανάλι. Έτσι, μπορεί εύκολα για παράδειγμα να δημιουργηθεί ένα κανάλι “user:antonis:timeline” με τα τελευταία νέα του χρήστη antonis. Η λειτουργικότητα πρότυπου Δημοσιεύσεων/Συνδρομών υποστηρίζεται από τις ακόλουθες εντολές Redis:

PUBLISH

Δημοσίευση σε ένα συγκεκριμένο κανάλι

SUBSCRIBE

Εγγραφή σε ένα συγκεκριμένο κανάλι

UNSUBSCRIBE

Διαγραφή από ένα συγκεκριμένο κανάλι

PSUBSCRIBE

Εγγραφή σε κανάλια που ταιριάζουν με ένα συγκεκριμένο πρότυπο

PUNSUBSCRIBE

Διαγραφή από κανάλια που ταιριάζουν με ένα συγκεκριμένο πρότυπο

3.3.5 Node.js & Socket.io

Οι περισσότεροι εξυπηρετητές συνήθως έχουν εγκατεστημένο λογισμικό που περιστρέφεται αποκλειστικά γύρω από το πρωτόκολλο HTTP. Επειδή όμως η τεχνολογία WebSocket χρησιμοποιεί το δικό της πρωτόκολλο, θα πρέπει στην πλευρά του εξυπηρετητή να προστεθούν επιπλέον βιβλιοθήκες (libraries) ή πρόσθετα (add-ons) που να υποστηρίζουν το πρωτόκολλο ws (websocket) και wss (websocket secure). Στην παρούσα εργασία χρησιμοποιήθηκε το εκτελούμενο στον εξυπηρετητή (server - side) λογισμικό Node.JS.

Το Node.js είναι ένα λογισμικό εξυπηρετητή (εκτελείται στον εξυπηρετητή), που δρα ως περίβλημα (wrapped) της γλώσσας JavaScript, για την ανάπτυξη επεκτάσιμων και οδηγούμενων από συμβάντα (event - driven) εφαρμογών. Ωστόσο πρέπει να διευκρινιστεί ότι το Node.js δεν είναι φτιαγμένο σε JavaScript, αλλά σε C++ και απλά, χρησιμοποιεί την JavaScript ως μια γλώσσα διερμήνευσης για την επεξεργασία του αιτήματος/απόκρισης στην πλευρά του εξυπηρετητή.

Αν και, η υλοποίηση ενός WebSocket εξυπηρετητή είναι δυνατή με την εγγενή υποστήριξη του Node.js, ωστόσο δεν είναι απαραίτητο. Πολλοί χαμηλού επιπέδου λεπτομέρειες πρέπει να ληφθούν υπόψη, πριν την υλοποίηση μιας πραγματική εφαρμογής, γεγονός που καθιστά τη χρήση μιας βιβλιοθήκης πολύ πιο πρακτική. Μια πρότυπη βιβλιοθήκη για την ανάπτυξη εφαρμογών WebSocket σε Node.JS που χρησιμοποιήθηκε και στην εφαρμογή της παρούσας διπλωματικής είναι η Socket.IO.

Η Βιβλιοθήκη Socket.IO δεν δρα μόνο ως βιβλιοθήκη περιβλήματος (wrapper library) που κάνει πολύ βολική την ανάπτυξη WebSocket εφαρμογών στην πλευρά του εξυπηρετητή, αλλά παρέχει και εφεδρικούς μηχανισμούς συμβατότητας (fallbacks), όπως long polling για πελάτες που δεν υποστηρίζουν το πρωτόκολλο WebSocket. Με αυτό τον τρόπο η βιβλιοθήκη Socket.IO δεν κάνει διάκριση μεταξύ ενός πελάτη που συνδέεται με την χρήση του πρωτοκόλλου WebSocket ή κάποιου άλλου μηχανισμού, παρέχει δηλαδή μια ενιαία διεπαφή προγραμματισμού, η οποία παραλείπει αυτές τις λεπτομέρειες υλοποίησης. Επιπλέον, η Socket.IO βιβλιοθήκη, έρχεται εξοπλισμένη με μια εύκολη και βολική διεπαφή προγραμματισμού (API) για την συγγραφή μέρους της εφαρμογής στην πλευρά του πελάτη (φυλλομετρητή), ενώ ταυτόχρονα παρέχει έναν μηχανισμό μηνυματοκεντρικής (message-oriented) επικοινωνίας. Αυτό είναι μια βελτίωση σε σχέση με το υποκείμενο WebSocket πρωτόκολλο, το οποίο δεν παρέχει πλαισίωση μηνύματος (message framing).

3.4 Σύνοψη

Σε αυτό το κεφάλαιο έγινε εισαγωγή στο πρωτόκολλο WebSocket της HTML5, καθώς επίσης παρουσιάστηκαν και οι βασικές έννοιες της προγραμματιστικής διεπαφής που το διέπει.

Τέλος, παρουσιάστηκαν οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής της παρούσας εργασίας.

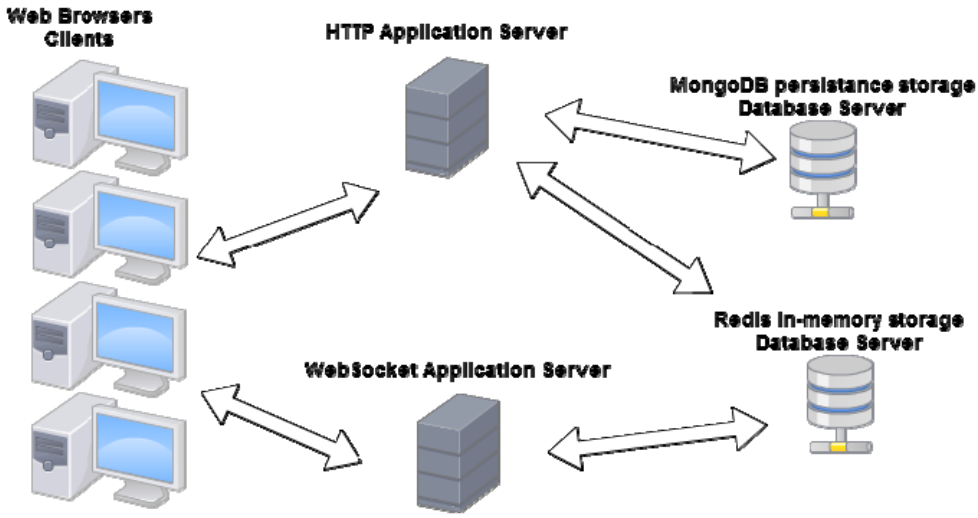
Στο επόμενο κεφάλαιο, θα περιγραφεί λεπτομερώς η αρχιτεκτονική του συστήματος της εφαρμογής που υλοποιήθηκε βάσει των τεχνολογιών που επιλεχτήκαν και η διαδικασία σχεδίασης με την μέθοδο ICONIX.

4 Εφαρμογή

4.1 Αρχιτεκτονική Συστήματος

Η πρότυπη εφαρμογή, ονόματι G-nnect, της παρούσας διπλωματικής εργασίας βασίζεται σε πολυεπίπεδη (muti-tier) αρχιτεκτονική, αποτελούμενη από τα παρακάτω πέντε διακριτά επίπεδα:

- Επίπεδο εξυπηρέτησης HTTP αιτημάτων (HTTP Application Server)
- Επίπεδο εξυπηρέτησης WebSocket αιτημάτων (WebSocket Application Server)
- Επίπεδο Μόνιμης Αποθήκευσης (Persistence Database Server)
- Επίπεδο Προσωρινής Αποθήκευσης (In-Merory Database Server)
- Επίπεδο Πελατών (Clients)



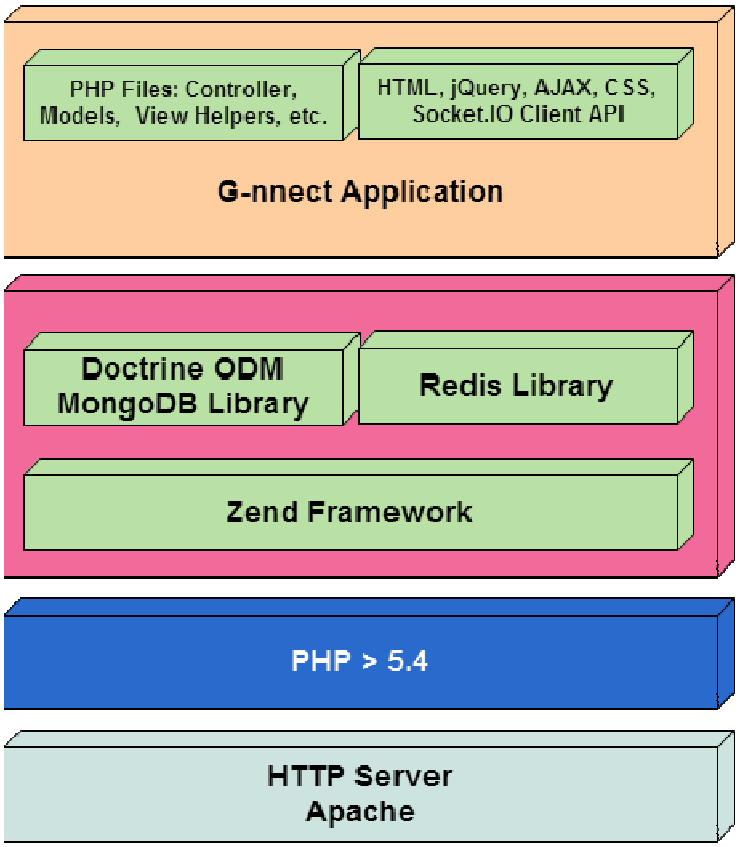
4-1: Αρχιτεκτονική εφαρμογής

4.1.1 Επίπεδο εξυπηρέτησης HTTP αιτημάτων (HTTP Application Server)

Σε αυτό το επίπεδο συναντάμε τον Apache HTTP ή απλά Apache. Ο Apache είναι ένα λογισμικό εξυπηρετητή παγκοσμίου ιστού (web). Όποτε ο χρήστης επισκέπτεται ένα ιστότοπο το πρόγραμμα πλοήγησης (browser) επικοινωνεί με τον εξυπηρετητή μέσω του πρωτοκόλλου HTTP, ο οποίος επεξεργάζεται αρχεία κώδικα PHP, παράγοντας ιστοσελίδες, τις οποίες και αποστέλλει στο πρόγραμμα πλοήγησης, διαμεσολαβώντας με αυτό τον τρόπο στην παροχή δεδομένων στην διεπαφή χρήστη (user interface) και παρουσιάζει τη λειτουργικότητα που παρέχει η εφαρμογή στον χρήστη.

Ο Apache, υποστηρίζει την εκτέλεση κώδικα PHP, με την χρήση του πρόσθετου (mod_php).

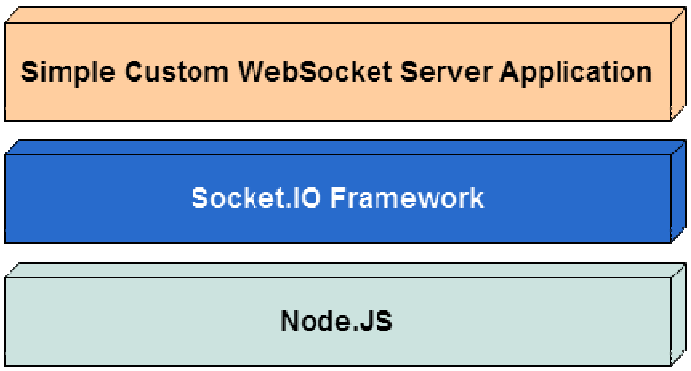
Σε αυτό το επίπεδο φιλοξενείται όλος ο κώδικας που έχει υλοποιηθεί υπό το πλαίσιο εργασίας της PHP, Zend Framework και των υποστηρικτικών βιβλιοθηκών Doctrine ODM MongoDB και Redis. Ο κώδικας επιχειρησιακής λογικής (business logic), οι ρουτίνες για την πρόσβαση δεδομένων και οι προβολές (views), επίσης φιλοξενούνται σε αυτό το σημείο.



4-2: HTTP Application Server

4.1.2 Επίπεδο εξυπηρέτησης WebSocket αιτημάτων (WebSocket Application Server)

Σε αυτό το επίπεδο είναι εγκατεστημένο το λογισμικό Node.js μαζί με την υποστηρικτική βιβλιοθήκη Socket.IO. Σε αυτό το σημείο φιλοξενούνται ο κατάλληλος κώδικας σε JavaScript ούτως ώστε ο WebSocket Server να είναι σε θέση δέχεται και εξυπηρετεί, από τους πελάτες, αιτήματα εγκαθίδρυσης WebSocket συνδέσεων. Επιπλέον, μετά την επιτυχή εγκαθίδρυση μιας WebSocket σύνδεσης, σε αυτό το επίπεδο λαμβάνει μέρος η διαδικασία έγγραφης (subscribe) του πελάτη σε κανάλι μετάδοσης δεδομένων που ενδιαφέρει το συγκεκριμένο πελάτη.



4-3: WebSocket Application Server

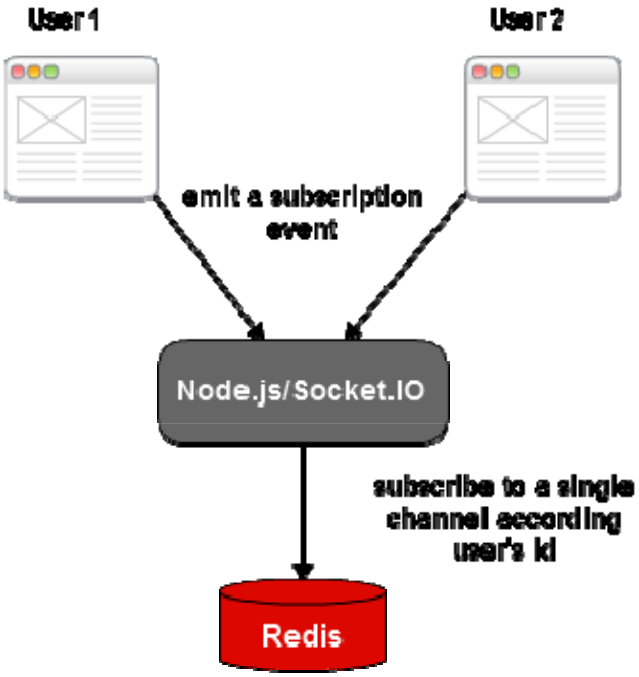
4.1.3 Επίπεδο Μόνιμης Αποθήκευσης (Persistence Database Server)

Αποτελεί το επίπεδο μόνιμης αποθήκευση δεδομένων παρέχοντας όλες τις απαραίτητες λειτουργίες για την αποθήκευση, ανάκτηση, ενημέρωση και συντήρηση των δεδομένων του συστήματος. Είναι το επίπεδο που συναντάμε την μη σχεσιακή βάση (NoSQL) δεδομένων MongoDB. Στην MongoDB αποθηκεύονται δεδομένα σχετικά με τους χρήστες, τα προφίλ, τα μηνύματα.

4.1.4 Επίπεδο Προσωρινής Αποθήκευσης (In-Memory Database Server)

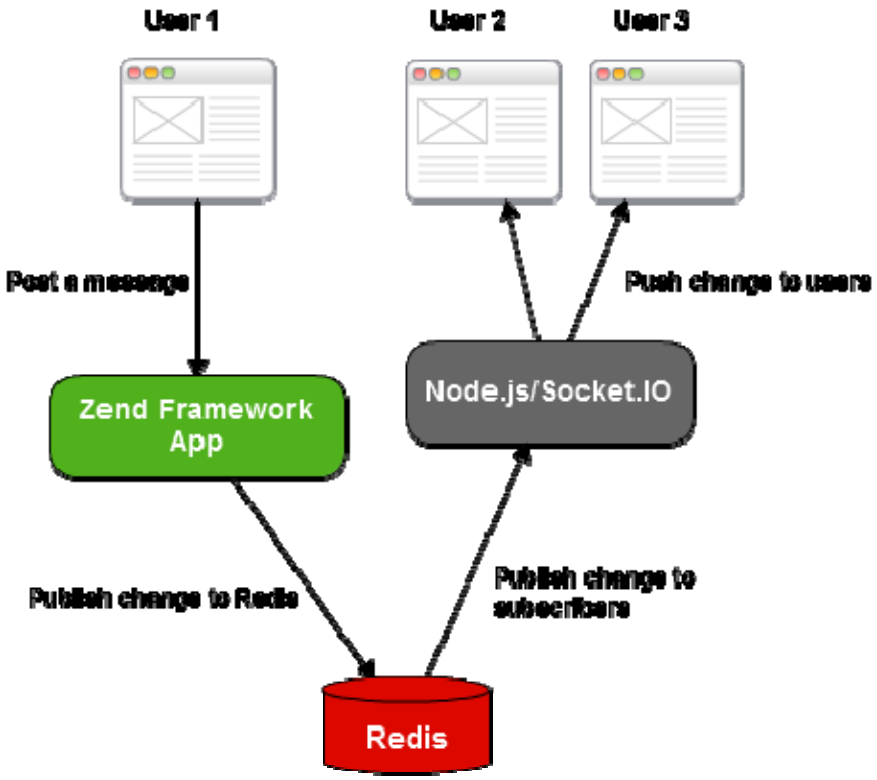
Σε αυτό το επίπεδο, ένα εκτελούμενο στιγμιότυπο της Redis, εξυπηρετεί την υλοποίηση του εσωτερικού μηχανισμού Δημοσιεύσεων/Συνδρομών (Publish/subscribe). Ο HTTP Application Server αλληλεπιδρά με αυτό το επίπεδο για δημοσίευση (Publish) σε συγκεκριμένα κανάλια. Ο WebSocket Application Server αλληλεπιδρά με αυτό το επίπεδο για εγγραφή (Subscribe) σε συγκεκριμένα κανάλια.

4.1.4.1 Πως θα λειτουργήσει;



Σχήμα 4-4: Διαδικασία εγγραφής σε κανάλι δεδομένων.

1. Όταν ο χρήστης μεταβαίνει στην κύρια σελίδα κάποιου προφίλ που διατηρεί, τότε το από το πρόγραμμα πλοήγησης εκπέμπεται μέσω του Socket.IO Client API, ένα συμβάν συνδρομής προς τον Node.JS Server.
2. Μέσω callback συνάρτησης, πραγματοποιείτε εγγραφή (subscription) του εκάστοτε χρήστη σε ένα μοναδικό κανάλι δημοσίευσης δεδομένων. Η μοναδικότητα του καναλιού προσδιορίζεται από το μοναδικό αναγνωριστικό του λογαριασμού πρόσβασης που διατηρεί ο χρήστης στη βάση δεδομένων mongoDB.



Σχήμα 4-5: Διαδικασία δημοσίευσης σε κανάλι δεδομένων.

1. Όταν ο χρήστης υποβάλει ένα νέο μήνυμα από το προφίλ του, η εφαρμογή δημοσιεύει (publish) μια ειδοποίηση για νέο μήνυμα μέσω της Redis προς όλους τους ενδιαφερόμενους (ακολούθους του εκάστοτε προφίλ); Όσοι είναι οι ακόλουθοι, τόσα είναι και τα κανάλια που θα λάβει μέρος η διαδικασία δημοσίευσης.
2. Στον Node Server, κάθε συνδεδεμένος χρήστης θα ενημερώνεται για νέες δημοσιεύσεις από το κανάλι δημοσίευσης που έχει πραγματοποιήσει εγγραφή.
3. Η ειδοποίηση για νέο μήνυμα προωθείτε προς τους χρήστες μέσω του Socket.IO (Server API)
4. Στην πλευρά του πελάτη, το Client API της Socket.IO βιβλιοθήκης λαμβάνει την ειδοποίηση και την προωθεί στην jQuery βιβλιοθήκη για την αναπαραστήσει στην οθόνη.

4.1.5 Επίπεδο Πελατών (Clients)

Αυτό το επίπεδο αποτελεί την επαφή του χρήστη με την εφαρμογή. Η επαφή αυτή μπορεί να είναι ένα πρόγραμμα πλοήγησης (browser). Το πρόγραμμα πλοήγησης επεξεργάζεται και προβάλει HTML πόρους, αποστέλλει HTTP αιτήματα για πόρους, επεξεργάζεται HTTP απαντήσεις, ξεκινάει αίτημα εγκαθίδρυσης WebSocket σύνδεσης, επεξεργάζεται ληφθέντα δεδομένα από μια WebSocket σύνδεση.

4.2 Ανάλυση και Σχεδιασμός Εφαρμογής

Σκοπός αυτού του κεφαλαίου είναι η παρουσίαση των σταδίων ανάπτυξης μιας διαδικτυακής εφαρμογής κοινωνικής δικτύωσης σύμφωνα με τη μεθοδολογία ICONIX.

Η μεθοδολογία ICONIX είναι επαναληπτική και χωρίζεται σε τέσσερα στάδια εργασιών. Σε κάθε στάδιο, οι εργασίες του προηγούμενου σταδίου αναθεωρούνται και ενημερώνονται. Παρακάτω παρουσιάζονται τα τέσσερα στάδια καθώς και οι εργασίες που δύναται να συμβαίνουν σε κάθε ένα από αυτά.

Στάδιο 1: Ανάλυση:

- Ορισμός λειτουργικών και μη απαιτήσεων. Οι λειτουργικές απαιτήσεις ορίζουν τι πρέπει να μπορεί να κάνει το σύστημα, ενώ οι μη- λειτουργικές απαιτήσεις (τεχνικές απαιτήσεις) ορίζουν πώς να το κάνει.
- Κατασκευή του μοντέλου του πεδίου προβλήματος (domain model). Το μοντέλο αυτό είναι μια γραφική απεικόνιση (διαγράμματα κλάσεων) των οντοτήτων/εννοιών (κλάσεις πεδίου) που χρησιμοποιούνται για την περιγραφή των απαιτήσεων του συστήματος καθώς και των σχέσεων μεταξύ τους.
- Ορισμός των περιπτώσεων χρήσης (use cases) του συστήματος. Οι περιπτώσεις χρήσης ορίζουν μια ακολουθία ενεργειών που ένας χρήστης του συστήματος (συνήθως πρόκειται για άνθρωπο αλλά μπορεί να είναι οποιαδήποτε εξωτερική οντότητα όπως ένα άλλο σύστημα) πραγματοποιεί στο σύστημα για να επιτύχει ένα συγκεκριμένο σκοπό.

Στάδιο 2: Ανάλυση - Αρχική Σχεδίαση

- Σχεδιασμός διαγραμμάτων ευρωστίας (Robustness Diagrams) για να αναπαραστήσουμε ένα προκαταρκτικό σχέδιο υλοποίησης των περιπτώσεων χρήσης του συστήματος.
- Αναθεώρηση του μοντέλου πεδίου προβλήματος αν απαιτείται με διαγραφή κλάσεων ή προσθήκη νέων κλάσεων.
- Προσθήκη ιδιοτήτων (attributes) στις κλάσεις.

Στάδιο 3: Αναλυτική Σχεδίαση

- Σχεδιασμός διαγραμμάτων ακολουθίας (Sequence Diagrams) για να αναπαραστήσουμε με λεπτομερειακό τρόπο το σχέδιο υλοποίησης των περιπτώσεων χρήσης του συστήματος.
- Προσθήκη μεθόδων (methods) στις κλάσεις.

Στάδιο 4: Υλοποίηση

4.2.1 Απαιτήσεις υψηλού επιπέδου

Κατά τη δημιουργία του πεδίου προβλήματος (domain model), μια καλή πηγή των κλάσεων του πεδίου, αποτελούν οι απαιτήσεις υψηλού επιπέδου - αυτές που συνήθως άλλα όχι πάντα είναι γραμμένες με την μορφή «Το σύστημα πρέπει να το κάνει "αυτό". Το σύστημα δεν πρέπει να κάνει "εκείνο"». Περάν του γεγονότος ότι οι απαιτήσεις υψηλού επιπέδου αποτελούν ταυτόχρονα και λειτουργικές απαιτήσεις, είναι χρήσιμο να αναλυθούν αυτές οι απαιτήσεις, για την εξαγωγή ουσιαστικών και ονοματικών φράσεων. Στη συνέχεια με μια επεξεργασία "εξυγίανσης" των εξαγόμενων αποτελεσμάτων μπορεί να δημιουργηθεί το αρχικό μοντέλο του πεδίου προβλήματος.

Παρακάτω παρουσιάζονται οι απαιτήσεις υψηλού επιπέδου της εφαρμογής G-nnect.

1. Το G-nnect, πρόκειται να είναι μια διαδικτυακή εφαρμογή.
2. Ο χρήστης πρέπει να διατηρεί ένα λογαριασμό πρόσβασης για την είσοδο του στο σύστημα.
    a. Λογαριασμός πρόσβασης αποτελεί το ζεύγος διεύθυνση ηλεκτρονικού ταχυδρομείου - κωδικός πρόσβαση
3. Ο χρήστης μπορεί να πραγματοποιήσει εγγραφή στο σύστημα για την απόκτηση λογαριασμού πρόσβασης δηλώνοντας διεύθυνση ηλεκτρονικού ταχυδρομείου και κωδικό πρόσβαση.
4. Ο χρήστης μπορεί να διατηρεί και να διαχειρίζεται (Δημιουργία νέου, Ενημέρωση, Διαγραφή) διακριτά, πολλαπλά προφίλ που υπόκεινται στον λογαριασμό του στο σύστημα.
    a. Το σύστημα ενσωματώνει δύο τύπους προφίλ: προσωπικό και εταιρικό.
    b. Ο χρήστης πρέπει να διατηρεί τουλάχιστον ένα προφίλ στο σύστημα.
    c. Ο χρήστης μπορεί να εναλλάσσεται μεταξύ των διαφορετικών προφίλ που διατηρεί.
    d. Ο χρήστης μπορεί να ορίσει κάποιο προεπιλεγμένο προφίλ, ουτώς ώστε κάθε φορά που πραγματοποιεί είσοδο στο σύστημα να μεταβαίνει αυτομάτως στην κύρια σελίδα αυτού του προφίλ.
5. Ο χρήστης, από την κύρια σελίδα του κάθε προφίλ που διατηρεί, θα μπορεί να γράφει και να υποβάλει μηνύματα μεγέθους έως 140 χαρακτήρες.
    a. Το σύστημα ενσωματώνει δύο τύπους μηνυμάτων :
        i. Προσωπικά μηνύματα : υποβάλλονται από προσωπικά προφίλ
        ii. Εταιρικά μηνύματα : υποβάλλονται από εταιρικά προφίλ.
6. Ο χρήστης, από την κύρια σελίδα του κάθε προφίλ που διατηρεί, θα μπορεί να βλέπει και να λαμβάνει, μηνύματα από τα προφίλ των χρηστών που ενδιαφέρεται ή «ακολουθεί».
    a. Για κάθε προφίλ, το σύστημα ενσωματώνει δύο λίστες ληφθέντων μηνυμάτων :
        i. Προσωπικά ληφθέντα μηνύματα, υποβληθέντα από προσωπικά προφίλ που «ακολουθεί»

ii. Εταιρικά ληφθέντα μηνύματα, υποβληθέντα από εταιρικά προφίλ που «ακολουθεί»

7. Για κάθε προφίλ, το σύστημα ενσωματώνει δύο λίστες από προφίλ άλλων χρηστών :
    a. Λίστα ακολουθουμένων: Σύνολο από προφίλ που το εκάστοτε προφίλ χρήστη «ακολουθεί» ώστε να ενημερώνεται για μηνύματα υποβαλλόμενα από αυτά.
    b. Λίστα ακολούθων: Δηλαδή σύνολο από προφίλ που «ακολουθούν» το εκάστοτε προφίλ χρήστη
8. Ο χρήστης θα πρέπει να ενημερώνεται/ειδοποιείται από το σύστημα για νέα υποβαλλόμενα μηνύματα από τα προφίλ των χρηστών που «ακολουθεί».
    a. Η ενημέρωση δύναται να πραγματοποιείται σε πραγματικό χρόνο και να λαμβάνει μέρος στην κύρια σελίδα του προφίλ του χρήστη, στο οποίο υπόκεινται τα προφίλ που «ακολουθεί».
    b. Ο χρήστης θα πρέπει να έχει την δυνατότερα κάνει λήψη νέων μηνυμάτων και κατόπιν δική του επιλογής (αλληλεπίδραση με το σύστημα) καθώς η ενημέρωση πραγματικού χρόνου μπορεί για κάποιο λόγο να μην λειτουργήσει.
9. Ο χρήστης μπορεί να αναζητήσει προφίλ άλλων χρηστών με βάση το ψευδώνυμο και στη συνέχεια να προβάλει οθόνη λεπτομερειών κάποιου προφίλ.
10. Από την οθόνη λεπτομερειών ενός προφίλ, ο χρήστης μπορεί να επιλέξει να προβάλει :
    a. Λίστα με τα υποβληθέντα μηνύματα του εκάστοτε προβαλλόμενου προφίλ.
    b. Λίστα με τα προφίλ που «ακολουθεί» το εκάστοτε προβαλλόμενο προφίλ.
    c. Λίστα με τα προφίλ που «ακολουθούν» το εκάστοτε προβαλλόμενο προφίλ.
11. Από την οθόνη λεπτομερειών ενός προφίλ, ο χρήστης μπορεί να επιλέξει να «ακολουθήσει» (αν δεν το «ακολουθεί» ήδη) ή να σταματήσει να «ακολουθεί» (αν το «ακολουθεί» ήδη) το εκάστοτε προβαλλόμενο προφίλ.
    a. Επίσης, ο χρήστης θα μπορεί να έχει την ίδια δυνατότητα από την προβολή οποιασδήποτε λίστας προφίλ, όπως αποτελέσματα αναζήτησης, λίστα ακολουθουμένων προφίλ, λίστα ακολούθων προφίλ.

4.2.2 Μοντέλο πεδίου προβλήματος

Από την περιγραφή απαιτήσεων υψηλού επιπέδου που δόθηκε παραπάνω, προκύπτει η ακόλουθη αρχική λίστα ουσιαστικών / ονοματικών φράσεων και πιθανών κλάσεων πεδίου προβλήματος (domain model) (έχοντας μετατρέψει το πρόσωπο σε πρώτο ενικό και έχοντας αποδώσει για λόγους ευκολότερης αναφοράς στη συνέχεια της σχεδίασης):

Λίστα ουσιαστικών/ Υποψήφιες κλάσεις	
Ουσιαστικό / Ονοματική φράση	**Εναλλακτική ξενόγλωσση ορολογία**
Διαδικτυακή εφαρμογή	Web App
Σύστημα	System
Χρήστης	User
Λογαριασμός πρόσβασης	User Account
Διεύθυνση ηλεκτρονικού ταχυδρομείου	Email
Κωδικός πρόσβασης	Password
Προφίλ	Profile
Προσωπικό Προφίλ	Regular Profile
Εταιρικό Προφίλ	Corporate Profile
Προεπιλεγμένο Προφίλ	Default Profile
Ψευδώνυμο	Nickname
Λίστα ακολουθουμένων	Following List
Λίστα ακολούθων	Follower List
Μήνυμα	Message
Προσωπικό Μήνυμα	Regular Message
Εταιρικό Μήνυμα	Corporate Message

Λίστα από προσωπικά ληφθέντα μηνύματα	Regular Received Messages List
Λίστα από εταιρικά ληφθέντα μηνύματα	Corporate Received Messages List
Λίστα από υποβληθέντα μηνύματα	Send Messages
Προβαλλόμενο προφίλ	Viewed Profile
Αποτελέσματα αναζήτησης	Search Results

Πίνακας 4: Αρχική λίστα από κλάσεις του πεδίου προβλήματος

Ουσιαστικά ή ονοματικές φράσεις (οθόνη λεπτομερειών, κεντρική σελίδα) που αναφέρονται σε οθόνες και στοιχειά του γραφικού περιβάλλοντος για αλληλεπίδραση με το δράστη (actor) του συστήματος παραλείπονται από τη αρχή καθότι δεν αποτελούν μέρος πεδίου προβλήματος.

4.2.2.1 Περιορισμός υποψηφίων κλάσεων

Η ανωτέρω λίστα μπορεί να περιοριστεί σημαντικά απαλείφοντας:

1. Αναφορές στο ίδιο το σύστημα λογισμικού που αναπτύσσουμε (G-nnect, Διαδικτυακή εφαρμογή, Σύστημα).
2. Αναφορές σε χειριστές του συστήματος που πρόκειται να αναπτύξουμε καθώς βρίσκονται «έξω» από τα όρια του συστήματος (Χρήστης).
3. Ουσιαστικά που πιθανόν να αποτελέσουν ιδιότητες άλλων κλάσεων. Η «διεύθυνση ηλεκτρονικού ταχυδρομείου» και ο «κωδικός πρόσβασης» δεν φαίνεται να αποτελούν ξεχωριστές κλάσεις στο πεδίο προβλήματος. Αντίθετα είναι πιο πιθανόν να αποτελούν ιδιότητες της κλάσης «Λογαριασμός πρόσβασης». Αντίστοιχα και το ψευδώνυμο πιθανόν να αποτελέσει ιδιότητα της κλάσης προφίλ.
4. Αναφορές που δεν φαίνεται προς το παρόν να μην έχουν σημαντικό ρόλο στο σύστημα και η έννοια τους είναι αμφιλεγόμενη (Προβαλλόμενο Προφίλ, Αποτελέσματα αναζήτησης).

Ωστόσο αν αποδειχθεί στη διαδικασία της ανάλυσης ότι κάποιες από αυτές τις οντότητες έχουν λειτουργίες απαραίτητες για το σύστημα, είναι απολύτως επιτρεπτό να εισαχθούν στη συνέχεια στο μοντέλο του πεδίου προβλήματος.

Νέα Λίστα ουσιαστικών/ Νέες Υποψήφιες κλάσεις	
Ουσιαστικό / Ονοματική φράση	**Εναλλακτική ξενόγλωσση ορολογία**
Λογαριασμός πρόσβασης	User Account
Προφίλ	Profile
Προσωπικό Προφίλ	Regular Profile
Εταιρικό Προφίλ	Corporate Profile
Λίστα ακολουθουμένων	Following List
Λίστα ακολούθων	Follower List
Μήνυμα	Message
Προσωπικό Μήνυμα	Regular Message
Εταιρικό Μήνυμα	Corporate Message
Λίστα από προσωπικά ληφθέντα μηνύματα	Regular Received Messages List
Λίστα από εταιρικά ληφθέντα μηνύματα	Corporate Received Messages List
Λίστα από υποβληθέντα μηνύματα	Send Messages

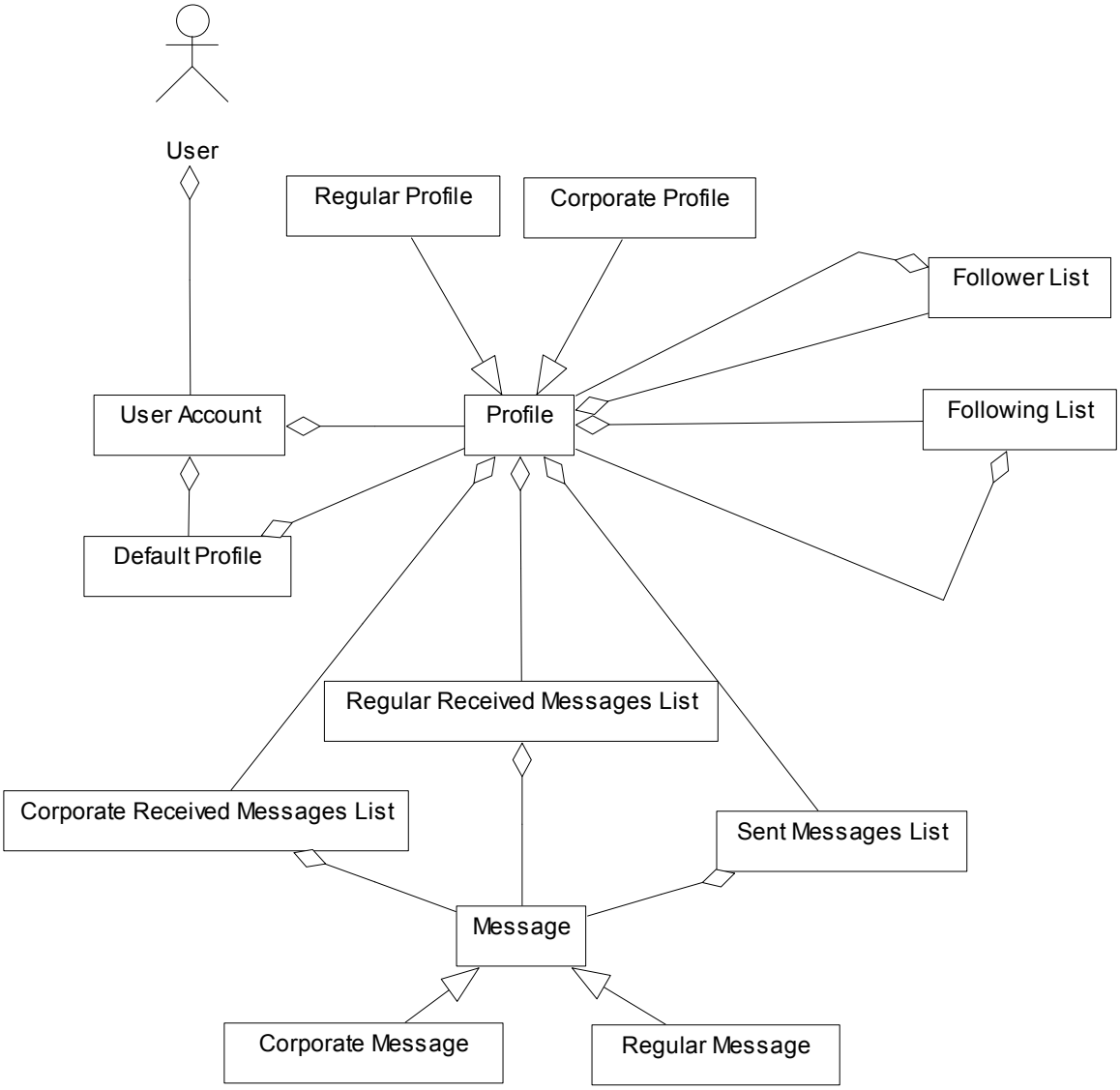
Πίνακας 5: Τελική λίστα από κλάσεις του πεδίου προβλήματος

4.2.2.2 Καθορισμός σχέσεων μεταξύ κλάσεων

Επόμενος στόχος είναι ο εντοπισμός σχέσεων μεταξύ των υποψηφίων κλάσεων του μοντέλου του πεδίου προβλήματος. Συνήθως αρκεί η απεικόνιση στο μοντέλο σχέσεων τύπου "έχει" (has) και σχέσεων τύπου "είναι" (is). Το πρώτο είδος αναφέρεται σε σχέσεις περιεκτικότητας μεταξύ δύο κλάσεων που στην απλούστερη μορφή καλούνται συναρμολογήσεις (ή συσσωματώσεις). Μια συναρμολόγηση υποδηλώνει ότι αντικείμενα μιας κλάσης περιέχουν αντικείμενα μιας άλλης κλάσης. Η συναρμολόγηση απεικονίζεται ως μια γραμμή μεταξύ των κλάσεων με έναν λευκό ρόμβο στο άκρο της περιέχουσας κλάσης. Το δεύτερο είδος αναφέρεται σε σχέσεις κληρονομικότητας μεταξύ κλάσεων. Μια σχέση κληρονομικότητας υποδηλώνει ότι μια κλάση (υποκλάση) αποτελεί ειδικότερη κατηγορία μιας άλλης (υπερκλάση) και κληρονομεί τις ιδιότητες και τη συμπεριφορά της. Η κληρονομικότητα απεικονίζεται ως μια γραμμή με ένα τρίγωνο στο άκρο της υπερκλάσης.

Από το κείμενο των απαιτήσεων υψηλού επιπέδου προκύπτει ότι ένας λογαριασμός εισόδου χρήστη μπορεί να περιλαμβάνει πολλαπλά προφίλ και επίσης περιλαμβάνει ένα προεπιλεγμένο προφίλ. Ένα προφίλ μπορεί να είναι (σχέση "είναι") είτε προσωπικό είτε εταιρικό. Ένα προφίλ περιλαμβάνει (σχέση "έχει") λίστα από προφίλ ακολούθων και λίστα από προφίλ ακολουθουμένων. Ένα προφίλ περιλαμβάνει (σχέση "έχει") λίστα από προσωπικά ληφθέντα μηνύματα, και λίστα από εταιρικά ληφθέντα μηνύματα. Ένα προφίλ περιλαμβάνει (σχέση "έχει") λίστα από υποβληθέντα μηνύματα. Ένα μήνυμα μπορεί να είναι (σχέση "είναι") είτε προσωπικό είτε εταιρικό.

Με βάση τις ανωτέρω πληροφορίες το αρχικό διάγραμμα του πεδίου προβλήματος που προκύπτει, φαίνεται στο σχήμα.



4-6: Διάγραμμα πεδίου προβλήματος (Αρχικό)

4.2.3 Μοντέλο Περιπτώσεων χρήσης

Στο μοντέλο περιπτώσεων χρήσης καταγράφονται οι απαιτήσεις των χρηστών διερευνώντας εξαντλητικά όλα τα πιθανά σενάρια χρήσης του συστήματος. Ο ορισμός μιας περίπτωσης χρήσης περιλαμβάνει όλες τις δυνατές συμπεριφορές, περιλαμβανομένης της κανονικής ή βασικής ροής (όπου για παράδειγμα εκτελούνται όλα όσα αναμένει ο χρήστης για την επίτευξη του στόχου) αλλά και όλων των "εναλλακτικών" ροών, όπου κάτι για παράδειγμα αποκλίνει από την "κανονική" συμπεριφορά. Κάθε περίπτωση χρήσης πρέπει να συνοδεύεται από κατάλληλη τεκμηρίωση, όπου καταγράφονται τόσο η βασική όσο και όλες οι εναλλακτικές ροές.

Από την ανάλυση απαιτήσεων προκύπτουν οι παρακάτω περιπτώσεις χρήσης (use cases).

Περιπτώσεις χρήσης (Use case) του δράστη (actor) «Χρήστης»	
Είσοδο στο σύστημα	Sign in
Εγγραφή στο σύστημα	Sign up
Διαχείριση (Προβολή, Δημιουργία, Ενημέρωση, Διαγραφή) προφίλ χρήστη	View, Create, Update, Delete Profile
Εναλλαγή προφίλ	Switch Profile
Ορισμός προεπιλεγμένου προφίλ	Set Default Profile
Υποβολή μηνύματος	Post Message
Προβολή ληφθέντων μηνυμάτων	View Received Messages
Αναζήτηση προφίλ	Search Profile
Ακολούθηση προφίλ	Follow
Παύση ακολούθησης προφίλ	Unfollow
Προβολή λεπτομερειών προφίλ	View Profile Details
Προβολή υποβληθέντων μηνυμάτων	View Sent Messages
Προβολή ακολούθων	View Followers
Προβολή ακολουθουμένων	View Followings
Λήψη νέων μηνυμάτων	Receive new Messages

Πίνακας 6: Περιπτώσεις χρήσης

4.2.4 Τεκμηρίωση Περιπτώσεων Χρήσης

Στη συνέχεια παρουσιάζεται η τεκμηρίωση των περιπτώσεων χρήσης του συστήματος. Η ανάλυση βασίζεται στις προδιαγραφές υψηλού επιπέδου αλλά και σε (υποθετική) διευκρίνιση ασαφειών που πραγματοποιήθηκε σε συνεργασία με τους τελικούς χρήστες του συστήματος.

4.2.4.1 Περίπτωση χρήσης: Είσοδο στο σύστημα

Βασική Ροή

- Το σύστημα εμφανίζει την οθόνη αυθεντικοποίησης στο χρήστη
- Ο χρήστης εισάγει την διεύθυνση ηλεκτρονικού ταχυδρομείου και τον κωδικό του και επιλέγει το πλήκτρο «Είσοδος».
- Το σύστημα κάνει αναζήτηση χρήστη ελέγχοντας διεύθυνση ηλεκτρονικού ταχυδρομείου και κωδικό πρόσβασης, επιτρέποντας την είσοδο του.
- Το σύστημα εμφανίζει την κύρια σελίδα προφίλ που έχει οριστεί ως προεπιλεγμένο

Εναλλακτική ροή 1

- Δεν συμπληρώθηκαν όλα τα υποχρεωτικά πεδία: το σύστημα επανεμφανίζει την οθόνη αυθεντικοποίησης ενημερώνοντας τον χρήστη με κατάλληλο μήνυμα.

Εναλλακτική ροή 2

- Ο χρήστης δε βρέθηκε: το σύστημα επανεμφανίζει την οθόνη αυθεντικοποίησης ενημερώνοντας το χρήστη με κατάλληλο μήνυμα.

4.2.4.2 Περίπτωση χρήσης: Εγγραφή στο σύστημα

Βασική Ροή

- Το σύστημα εμφανίζει την οθόνη εγγραφής στο χρήστη
- Ο χρήστης εισάγει διεύθυνση ηλεκτρονικού ταχυδρομείου, επιθυμητό κωδικό πρόσβασης, ψευδώνυμο και επιλέγει το πλήκτρο «Εγγραφή».
- Το σύστημα δημιουργεί έναν νέο χρήστη μαζί με ένα προεπιλεγμένο προσωπικό προφίλ για αυτό τον χρήστη
- Το σύστημα εμφανίζει την οθόνη αυθεντικοποίησης στον χρήστη

Εναλλακτική ροή 1

- Υπάρχει ήδη χρήστης με το ίδιο ψευδώνυμο: το σύστημα επανεμφανίζει την οθόνη εγγραφής ενημερώνοντας το χρήστη με κατάλληλο μήνυμα.

Εναλλακτική ροή 2

- Υπάρχει ήδη χρήστης με την ίδια διεύθυνση ηλεκτρονικού ταχυδρομείου: το σύστημα επανεμφανίζει την οθόνη εγγραφής ενημερώνοντας τον χρήστη με κατάλληλο μήνυμα.

Εναλλακτική ροή 3

- Δεν συμπληρώθηκαν όλα τα υποχρεωτικά πεδία: το σύστημα επανεμφανίζει την οθόνη εγγραφής ενημερώνοντας τον χρήστη με κατάλληλο μήνυμα.

4.2.4.3 Περίπτωση χρήσης: Διαχείριση προφίλ (Δημιουργία προφίλ)

Βασική Ροή

- Το σύστημα εμφανίζει την οθόνη δημιουργίας προφίλ
- Ο χρήστης εισάγει ψευδώνυμο, φωτογραφία, περιγραφή και επιλέγει το πλήκτρο «Δημιουργία».
- Το σύστημα δημιουργεί ένα νέο προφίλ χρήστη
- Το σύστημα επανεμφανίζει την οθόνη δημιουργίας νέου προφίλ για πιθανή δημιουργία επιπλέουν προφίλ

Εναλλακτική ροή 1

- Δεν συμπληρώθηκαν όλα τα υποχρεωτικά πεδία: το σύστημα επανεμφανίζει την οθόνη δημιουργίας προφίλ ενημερώνοντας τον χρήστη με κατάλληλο μήνυμα.

Εναλλακτική ροή 2

- Υπάρχει προφίλ άλλου χρήστη με το ίδιο ψευδώνυμο: το σύστημα επανεμφανίζει την οθόνη δημιουργίας προφίλ ενημερώνοντας τον χρήστη με κατάλληλο μήνυμα.

4.2.4.4 Περίπτωση χρήσης: Διαχείριση προφίλ (Ενημέρωση προφίλ)

Βασική Ροή

- Το σύστημα εμφανίζει την οθόνη ενημέρωσης προφίλ
- Ο χρήστης ενημερώνει τα πεδία ψευδώνυμο, φωτογραφία, περιγραφή και επιλέγει το πλήκτρο «Ενημέρωση».
- Το σύστημα ενημερώνει το προφίλ του χρήστη
- Το σύστημα επανεμφανίζει την οθόνη ενημέρωσης προφίλ για πιθανή εκ νέου ενημέρωση

Εναλλακτική ροή 1

- Δεν συμπληρώθηκαν όλα τα υποχρεωτικά πεδία: το σύστημα επανεμφανίζει την οθόνη ενημέρωσης προφίλ ενημερώνοντας τον χρήστη με κατάλληλο μήνυμα.

Εναλλακτική ροή 2

- Υπάρχει προφίλ άλλου χρήστη με το ίδιο ψευδώνυμο: το σύστημα επανεμφανίζει την οθόνη ενημέρωσης προφίλ ενημερώνοντας τον χρήστη με κατάλληλο μήνυμα.

4.2.4.5 Περίπτωση χρήσης: Διαχείριση προφίλ (Διαγραφή προφίλ)

Βασική Ροή

- Το σύστημα εμφανίζει την οθόνη με τα προφίλ του χρήστη
- Ο χρήστης επιλέγει το πλήκτρο «Διαγραφή» για το προφίλ που επιθυμεί να διαγράψει
- Το σύστημα διαγράφει το προφίλ και επανεμφανίζει την οθόνη με τα προφίλ του χρήστη.

Εναλλακτική ροή 1

- Το προφίλ είναι μοναδικό για τον χρήστη: το σύστημα δεν επιτρέπει την επιλογή του πλήκτρου «Διαγραφή».

Εναλλακτική ροή 2

- Το προφίλ έχει οριστεί ως σταθερά προκαθορισμένο: το σύστημα δεν επιτρέπει την επιλογή του πλήκτρου «Διαγραφή».

4.2.4.6 Περίπτωση χρήσης: Εναλλαγή προφίλ

- Το σύστημα εμφανίζει την οθόνη με τα προφίλ του χρήστη
- Ο χρήστης επιλέγει το πλήκτρο «Εναλλαγή» για το προφίλ που επιθυμεί να μεταβεί
- Το σύστημα εμφανίζει την κύρια σελίδα του προφίλ που επέλεξε

4.2.4.7 Περίπτωση χρήσης: Ορισμός προεπιλεγμένου προφίλ

- Το σύστημα εμφανίζει την οθόνη με τα προφίλ του χρήστη
- Ο χρήστης επιλέγει το πλήκτρο «Προεπιλογή» για το προφίλ που επιθυμεί να ορίσει ως προεπιλεγμένο
- Το σύστημα εμφανίζει την κύρια σελίδα του προφίλ που επέλεξε

4.2.4.8 Περίπτωση χρήσης: Υποβολή μηνύματος

Βασική Ροή

- Το σύστημα εμφανίζει την αρχική σελίδα του προφίλ από το οποίο θα υποβληθεί το μήνυμα
- Ο χρήστης εισάγει το μήνυμα και επιλέγει το πλήκτρο «Υποβολή»
- Το σύστημα ενημερώνει την λίστα υποβληθέντων μηνυμάτων και εμφανίζει το μήνυμα στην λίστα μηνυμάτων όπου εμφανίζονται τα μηνύματα που λαμβάνει από άλλους χρήστες

Εναλλακτική ροή 1

- Το μήνυμα ξεπερνάει το όριο των 140 χαρακτήρων: το σύστημα δεν επιτρέπει την επιλογή του πλήκτρου «Υποβολή».

Εναλλακτική ροή 2

- Το μήνυμα είναι κενό: το σύστημα δεν επιτρέπει την επιλογή του πλήκτρου «Υποβολή».

4.2.4.9 Περίπτωση χρήσης: Αναζήτηση προφίλ

Βασική Ροή

- Το σύστημα εμφανίζει την φόρμα αναζήτησης
- Ο χρήστης εισάγει ψευδώνυμο προς αναζήτηση και επιλέγει το πλήκτρο «Αναζήτηση»
- Το σύστημα αναζήτα προφίλ που ταιριάζουν και τα εμφανίζει στην σελιδα αποτελεσμάτων αναζήτησης

Εναλλακτική ροή 1

- Δε βρέθηκαν προφίλ να ταιριάζουν: το σύστημα εμφανίζει κενή λίστα αποτελεσμάτων ενημερώνοντας τον χρήστη με κατάλληλο μήνυμα.

Εναλλακτική ροή 2

- Δε συμπληρώθηκε το ψευδώνυμο προφίλ προς αναζήτηση : το σύστημα εμφανίζει κενή λίστα αποτελεσμάτων ενημερώνοντας τον χρήστη με κατάλληλο μήνυμα.

4.2.4.10 Περίπτωση χρήσης: Ακολούθηση προφίλ

- Ο χρήστης επιλέγει το πλήκτρο «Ακολούθηση» για το προφίλ που προβάλλεται από το σύστημα εκείνη την στιγμή είτε από την οθόνη πληροφοριών ενός προφίλ είτε από μια λίστα προβολής προφίλ.
- Το σύστημα προσθέτει στην λίστα ακολουθουμένων προφίλ του εκάστοτε προφίλ χρήστη, το προφίλ το οποίο επέλεξε να ακολουθήσει
- Το σύστημα προσθέτει το εκάστοτε προφίλ χρήστη στη λίστα ακολούθων του προφίλ που επέλεξε να ακολουθήσει

4.2.4.11 Περίπτωση χρήσης: Παύση ακολούθησης προφίλ

- Ο χρήστης επιλέγει το πλήκτρο «Παύση Ακολούθησης» για το προφίλ που προβάλλεται από το σύστημα εκείνη την στιγμή
- Το σύστημα αφαιρεί από την λίστα ακολουθουμένων προφίλ του εκάστοτε προφίλ χρήστη, το προφίλ το οποίο επέλεξε να μην ακολουθήσει
- Το σύστημα αφαιρεί το εκάστοτε προφίλ χρήστη από την λίστα ακολούθων του προφίλ που επέλεξε να μην ακολουθήσει

4.2.4.12 Περίπτωση χρήσης : Προβολή λεπτομερειών ενός προφίλ

- Το σύστημα εμφανίζει λίστα από προφίλ χρηστών με την μορφή συνδέσμων
- Ο χρήστης επιλέγει τον σύνδεσμο του προφίλ που επιθυμεί να προβάλει.
- Το σύστημα ανακτά πληροφορίες για αυτό το προφίλ
- Το σύστημα εμφανίζει στον χρήστη την σελίδα με τις πληροφορίες του προφίλ

4.2.4.13 Προβολή υποβληθέντων μηνυμάτων

- Το σύστημα εμφανίζει στον χρήστη την σελίδα με τις πληροφορίες του προβαλλόμενου προφίλ
- Ο χρήστης επιλέγει τον σύνδεσμο «Μηνύματα».
- Το σύστημα ανακτά τα υποβληθέντα μηνύματα για αυτό το προφίλ και τα εμφανίζει στην σελίδα

4.2.4.14 Προβολή λίστας ακολούθων

- Το σύστημα εμφανίζει στον χρήστη την σελίδα με τις πληροφορίες του προβαλλόμενου προφίλ
- Ο χρήστης επιλέγει τον σύνδεσμο «Ακόλουθοι».
- Το σύστημα ανακτά τους ακολούθους για αυτό το προφίλ και τους εμφανίζει στην σελίδα

4.2.4.15 Προβολή λίστας ακολουθουμένων

- Το σύστημα εμφανίζει στον χρήστη την σελίδα με τις πληροφορίες του προβαλλόμενου προφίλ
- Ο χρήστης επιλέγει τον σύνδεσμο «Ακόλουθούμενοι».
- Το σύστημα ανακτά τους ακολουθούμενους για αυτό το προφίλ και τους εμφανίζει στην σελίδα

4.2.4.16 Προβολή ληφθέντων μηνυμάτων

- Το σύστημα εμφανίζει στον χρήστη την κύρια σελίδα προφίλ
- Το σύστημα ανακτά τα ληφθέντα μηνύματα και τα εμφανίζει στην σελίδα

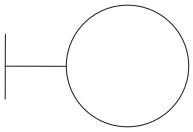
4.2.4.17 Λήψη νέων μηνυμάτων

- Το σύστημα εμφανίζει στον χρήστη την κύρια σελίδα προφίλ
- Ο χρήστης επιλέγει το πλήκτρο «Λήψη νέων μηνυμάτων».
- Το σύστημα ανακτά τα νέα μηνύματα και τα εμφανίζει στην σελίδα

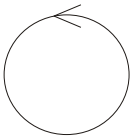
4.2.5 Ανάλυση Ευρωστίας

Η ανάλυση ευρωστίας (robustness analysis) αποτελεί μια τεχνική, για τη μετάβαση από τις περιπτώσεις χρήσης σε ένα λεπτομερές σχέδιο. Κατά την ανάλυση ευρωστίας το κείμενο των περιπτώσεων χρήσης μεταφράζεται σταδιακά σε μια γραφική απεικόνιση κλάσεων και συμπεριφοράς (διάγραμμα ευρωστίας). Κάθε οντότητα η οποία σύμφωνα με το μοντέλο πεδίου προβλήματος αποτελεί μια κλάση του συστήματος, απεικονίζεται στο διάγραμμα ευρωστίας, κατηγοριοποιώντας την με βάση ένα από τα ακόλουθα τρία στερεότυπα κλάσεων:

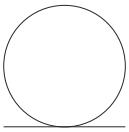
- Συνοριακές κλάσεις: κλάσεις που αποτελούν διασύνδεση μεταξύ του συστήματος και του "εξωτερικού κόσμου", δηλαδή των χειριστών του
- Κλάσεις οντοτήτων: κλάσεις του μοντέλου πεδίου προβλήματος
- Κλάσεις ελέγχου: κλάσεις που αποτελούν την "κόλλα" μεταξύ των συνοριακών και των κλάσεων οντοτήτων και αναπαριστούν τη συμπεριφορά του συστήματος.



(Συνοριακή κλάση) Διασύνδεση συστήματος με χρήστη



(Κλάση ελέγχου) Συμπεριφορά συστήματος



(Κλάση οντότητας) Κλάση του μοντέλου πεδίου προβλήματος

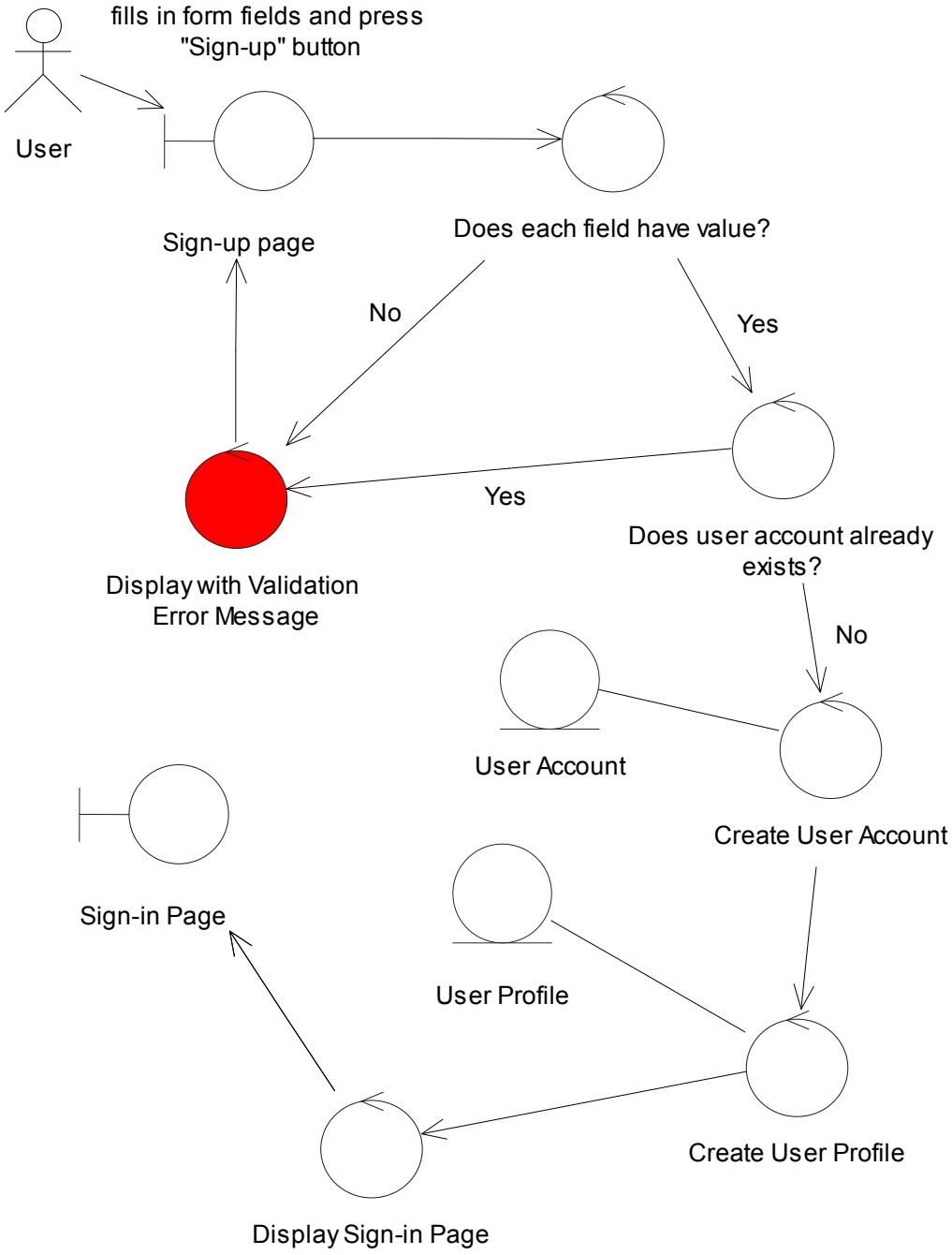
Σε ένα διάγραμμα ευρωστίας οι κλάσεις συσχετίζονται μεταξύ τους, υποδηλώνοντας την εννοιολογική συσχέτιση που υπάρχει ή υπονοείται στο κείμενο των περιπτώσεων χρήσης.

Κατά τη σχεδίαση διαγραμμάτων ευρωστίας οφείλουν να τηρούνται οι ακόλουθοι τρεις κανόνες:

- ουσιαστικά (χειριστές, συνοριακές κλάσεις και κλάσεις οντοτήτων) δεν μπορούν να συσχετίζονται απευθείας με άλλα ουσιαστικά
- ουσιαστικά μπορούν να συσχετίζονται με ρήματα (κλάσεις ελέγχου) και το αντίθετο
- ρήματα μπορούν να συσχετίζονται με άλλα ρήματα

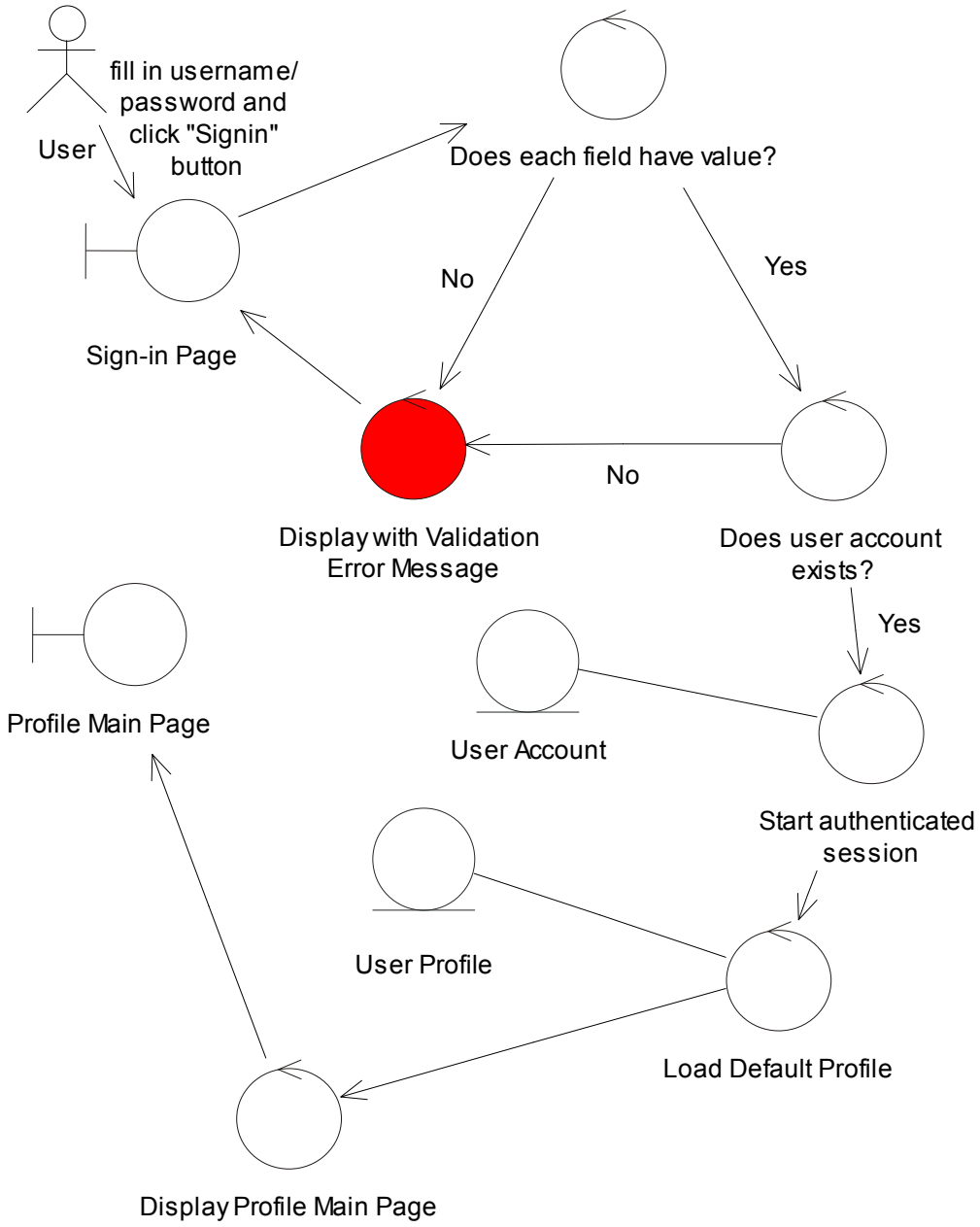
4.2.6 Διαγράμματα Ευρωστίας Συστήματος

4.2.6.1 Διάγραμμα ευρωστίας : Εγγραφή στο σύστημα



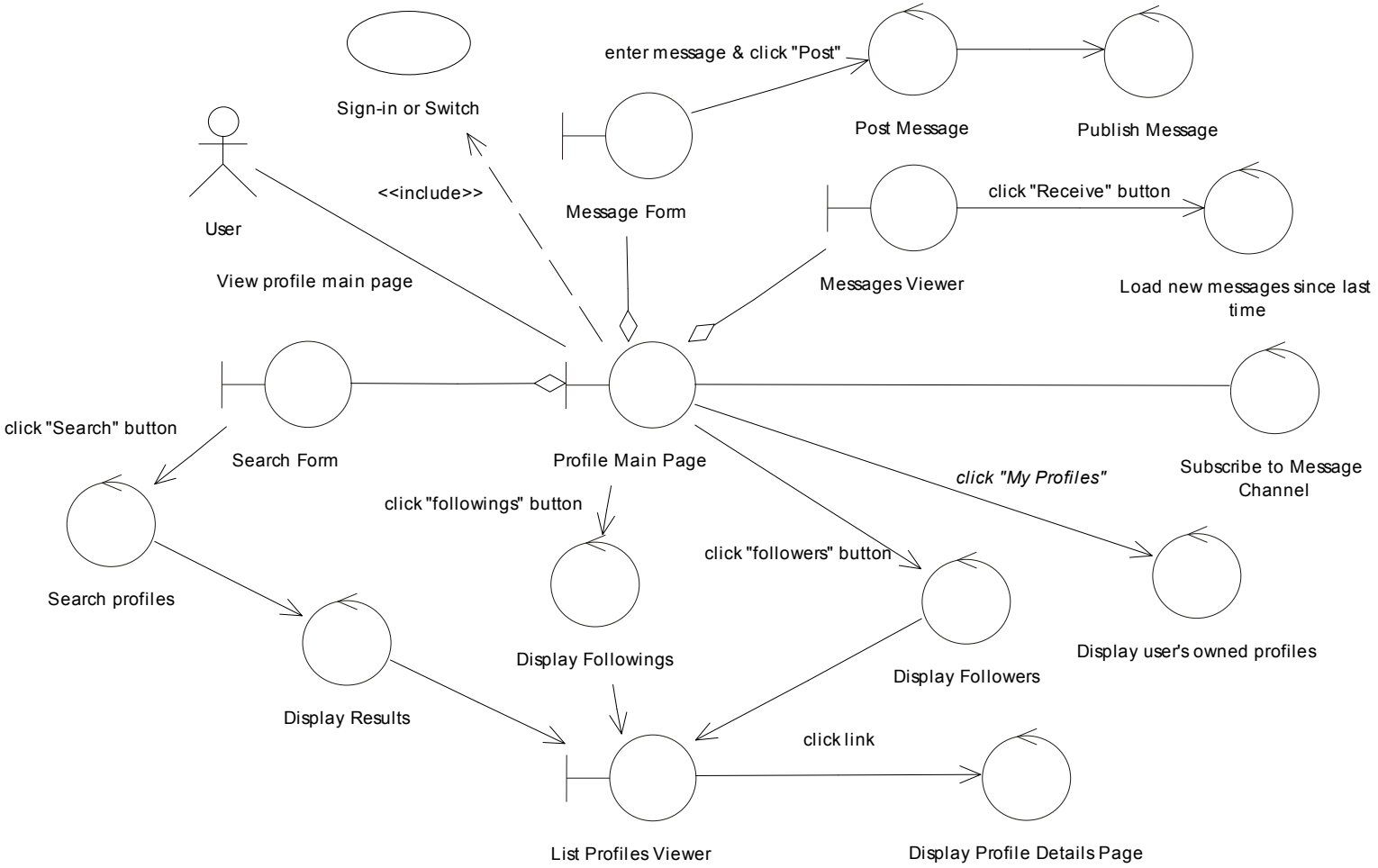
4-7: Διάγραμμα ευρωστίας για την περίπτωση χρήσης «Εγγραφή στο σύστημα»

4.2.6.2 Διάγραμμα ευρωστίας : Είσοδος στο σύστημα



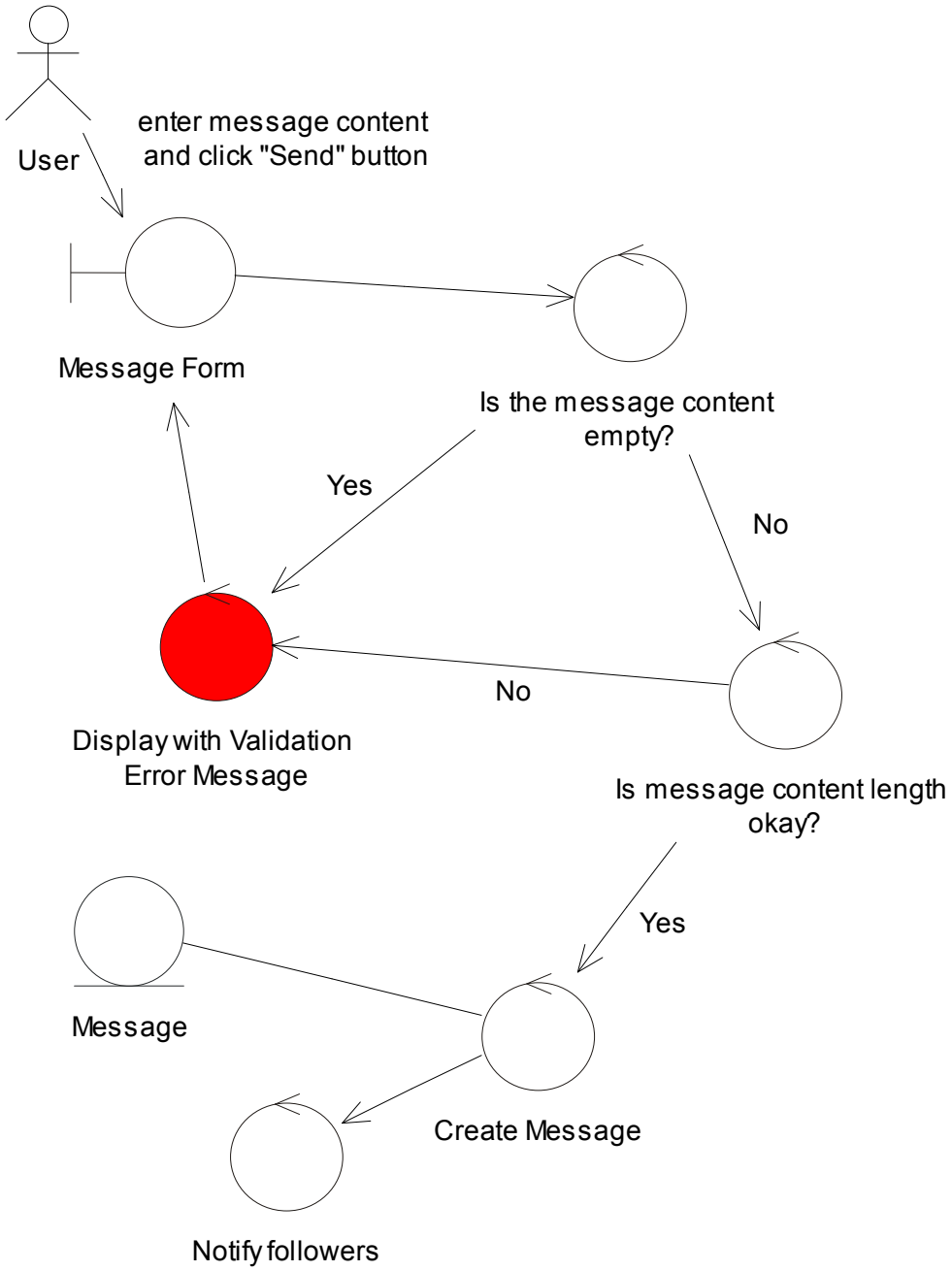
4-8: Διάγραμμα ευρωστίας για την περίπτωση χρήσης «Είσοδο στο σύστημα»

4.2.6.3 Διάγραμμα ευρωστίας : Γενική επισκόπιση περιπτώσεων χρήσης μετά από επιτυχή είσοδο στο σύστημα



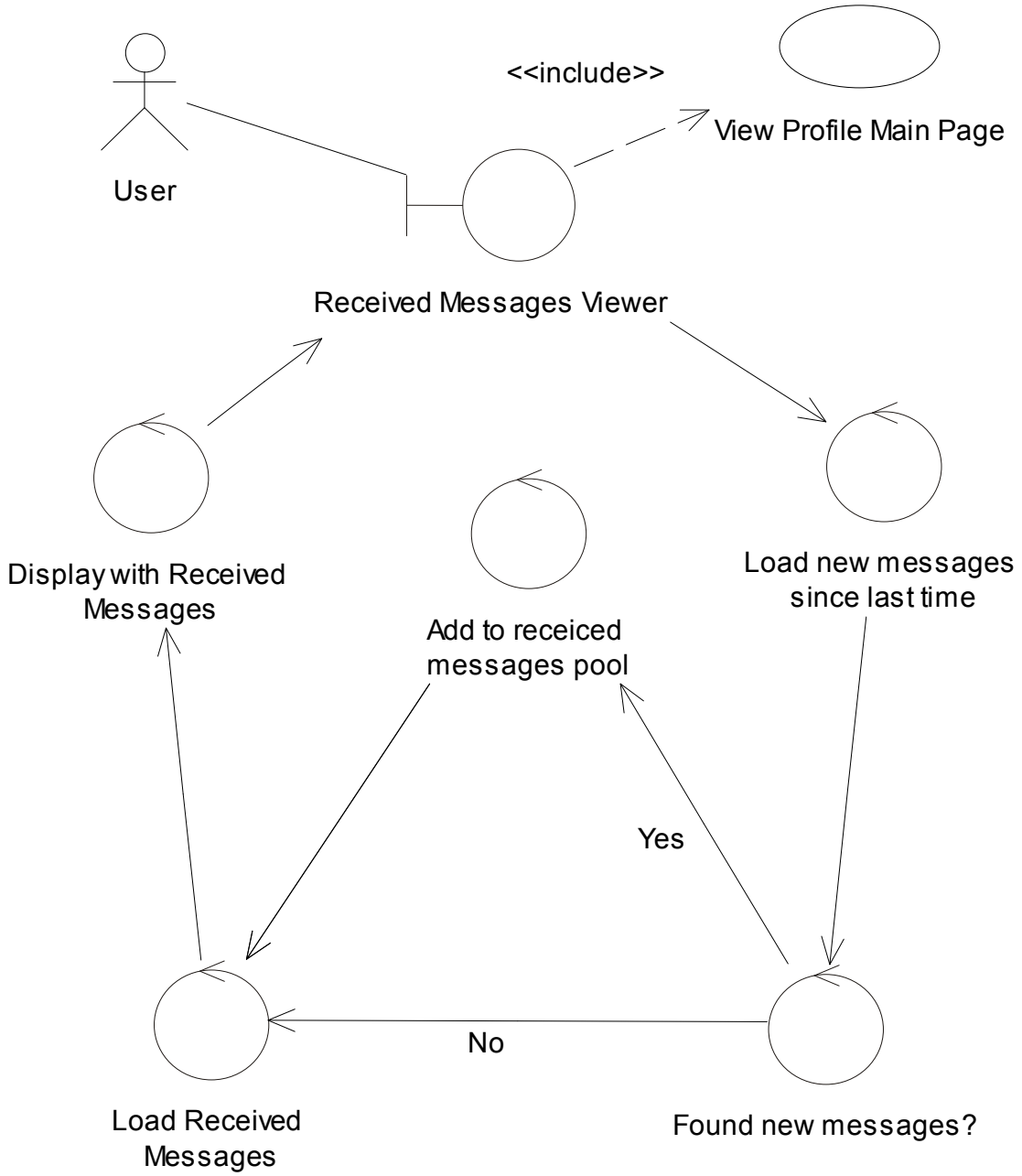
4-9: Διάγραμμα ευρωστίας για γενική επισκόπιση περιπτώσεων χρήσης

4.2.6.4 Διάγραμμα ευρωστίας : Υποβολή μηνύματος



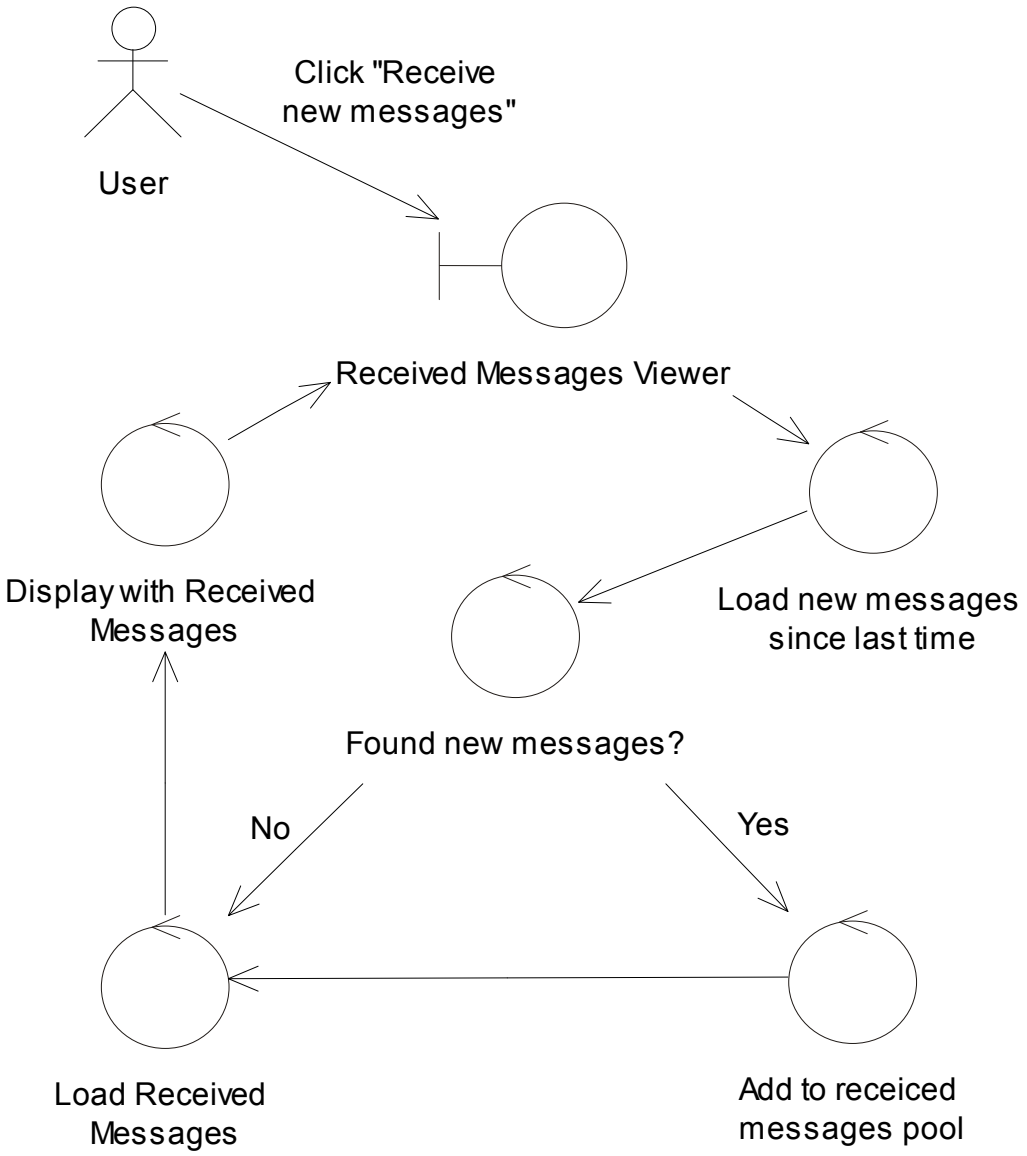
4-10: Διάγραμμα ευρωστίας για την περίπτωση χρήσης «Υποβολή μηνύματος»

4.2.6.5 Διαγραμμα ευρωστία : Προβολή ληφθέντων μηνυμάτων



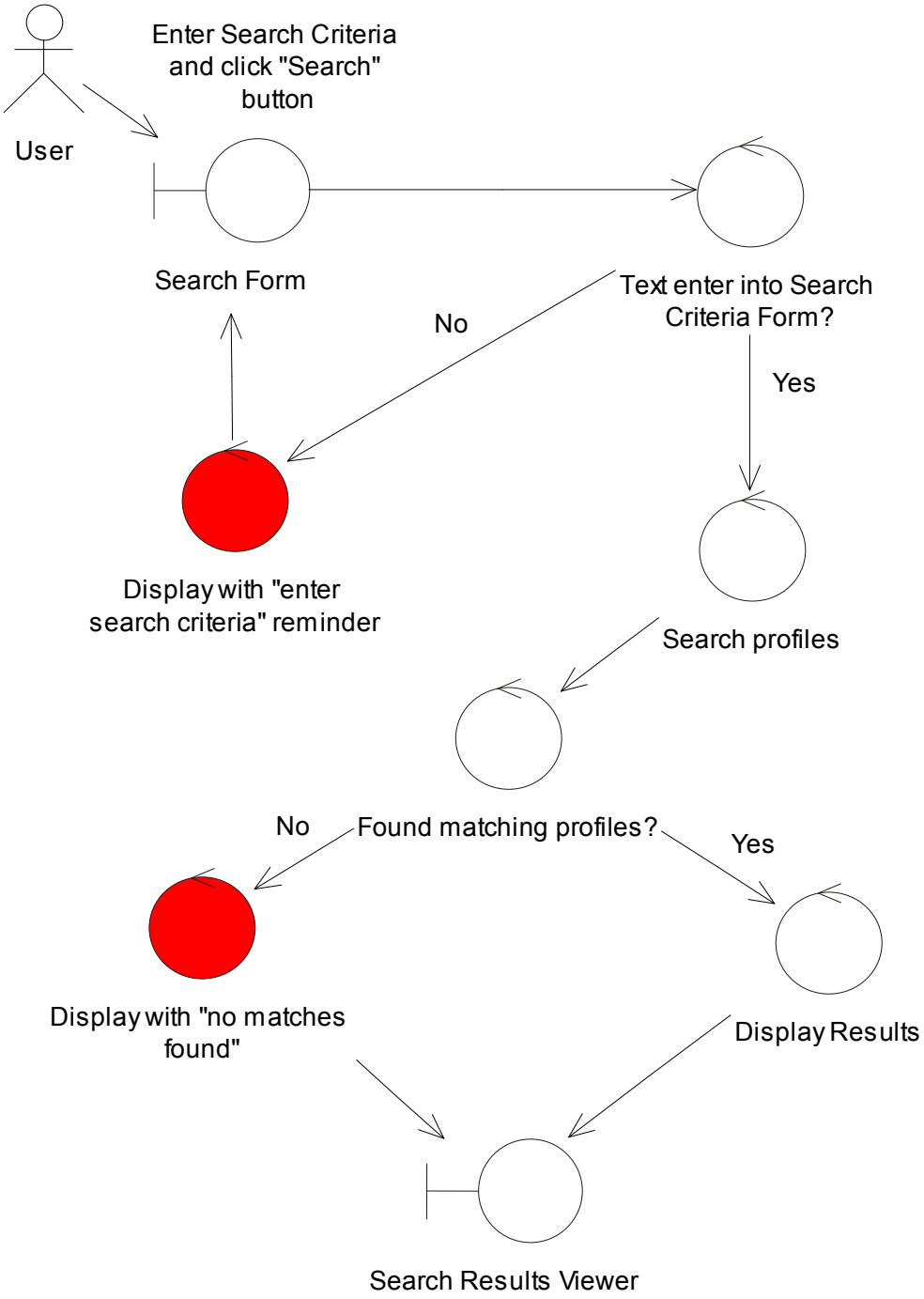
4-11: Διάγραμμα ευρωστίας για την περίπτωση χρήσης «Προβολή ληφθέντων μηνυμάτων»

4.2.6.6 Διαγραμμα ευρωστία : Λήψη νέων μηνυμάτων



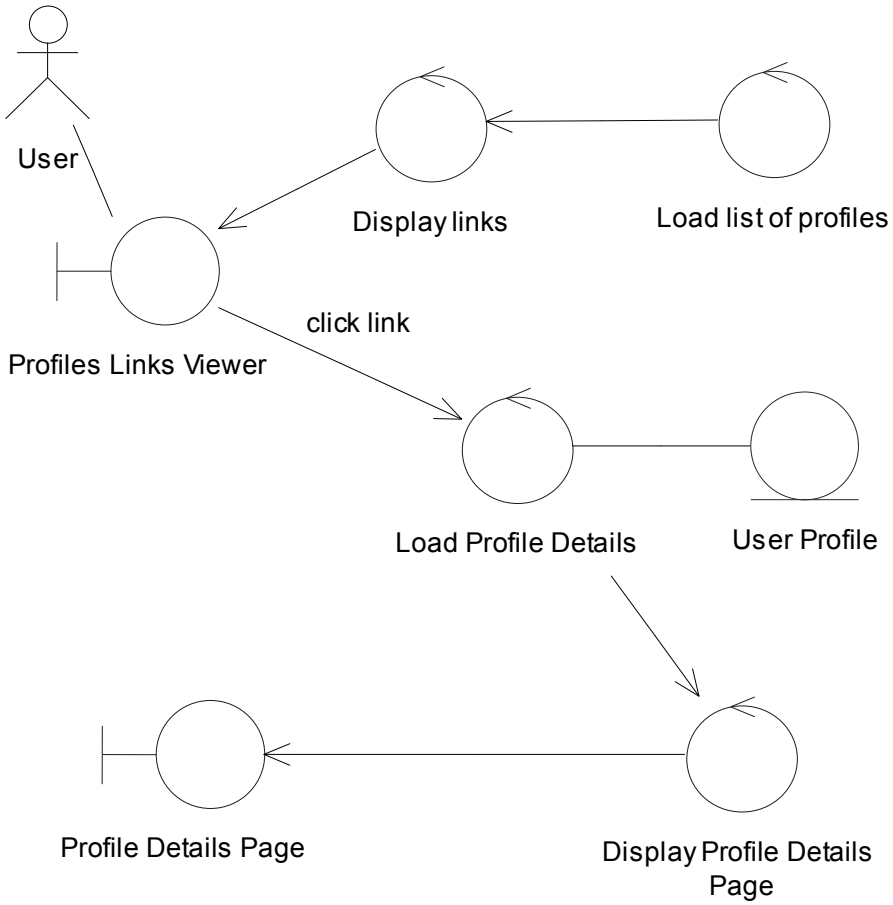
4-12 : Διάγραμμα ευρωστίας για την περίπτωση χρήσης «Λήψη νέων μηνυμάτων»

4.2.6.7 Διαγραμμα ευρωστίας : Αναζήτηση προφίλ



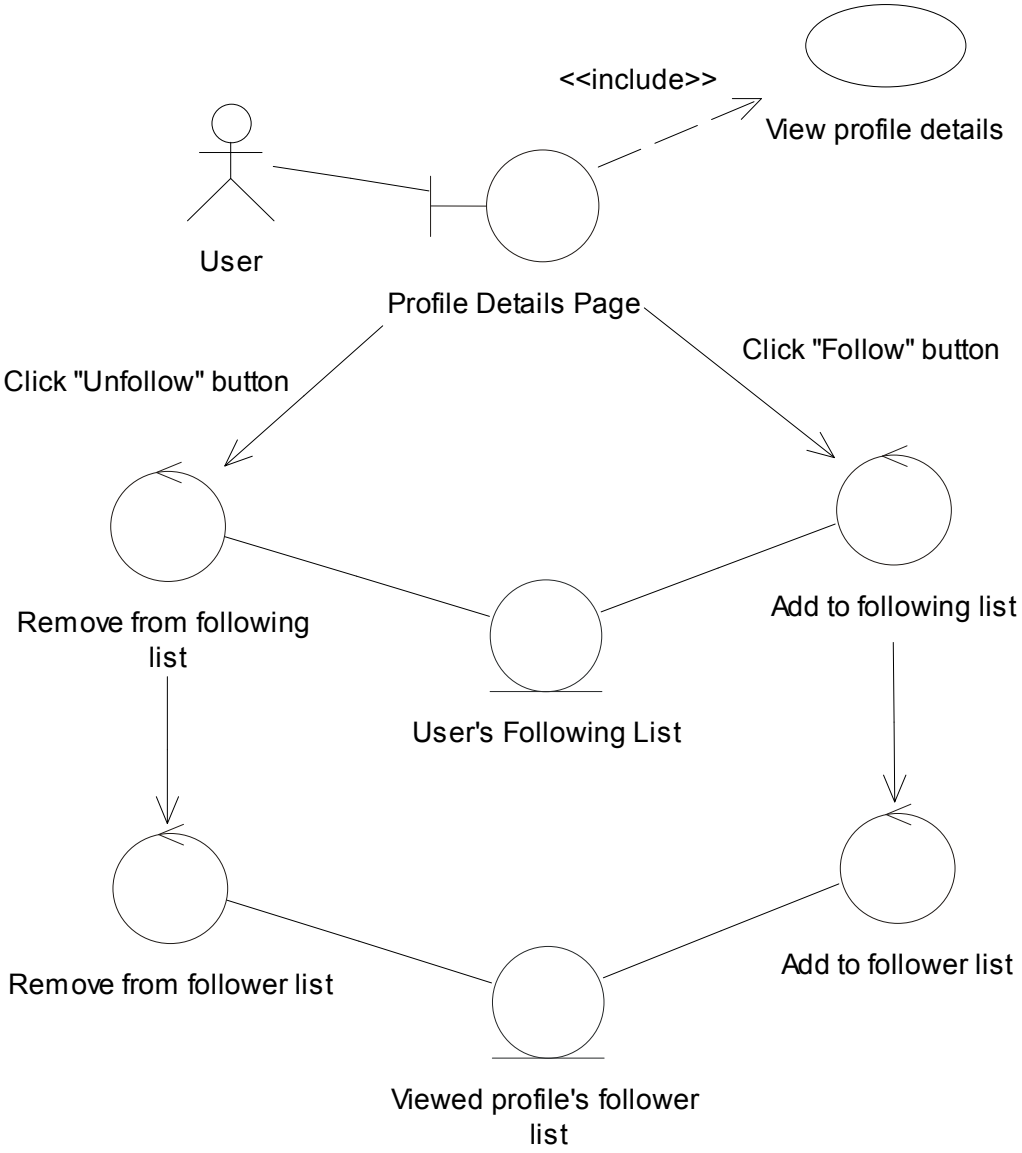
4-13: Διάγραμμα ευρωστίας για την περίπτωση χρήσης «Αναζήτηση»

4.2.6.8 Διαγραμμα ευρωστίας : Προβολή πληροφοριών προφίλ



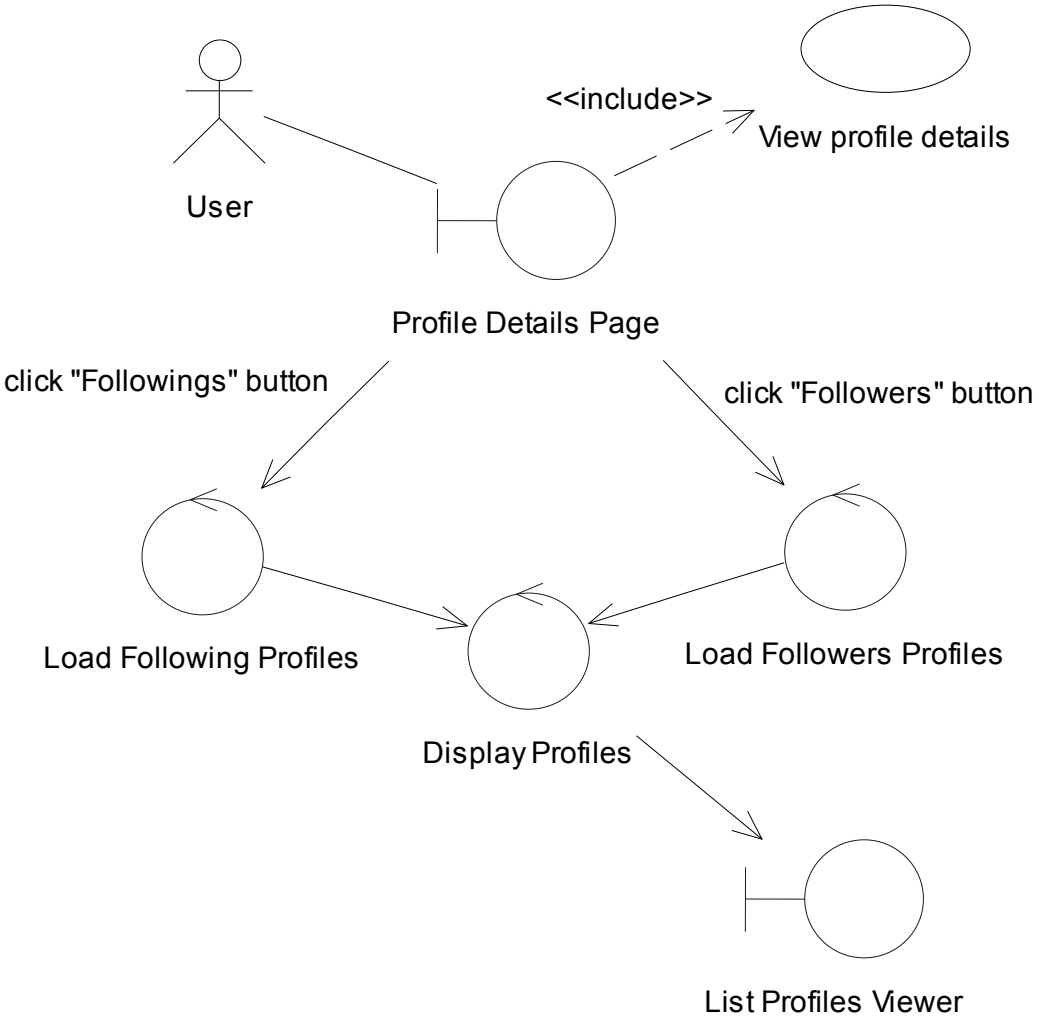
4-14: Διάγραμμα ευρωστίας για την περίπτωση χρήσης «Προβολή πληροφοριών προφίλ»

4.2.6.9 Διάγραμμα ευρωστίας : Ακολούθηση/ Παύση Ακολούθησης



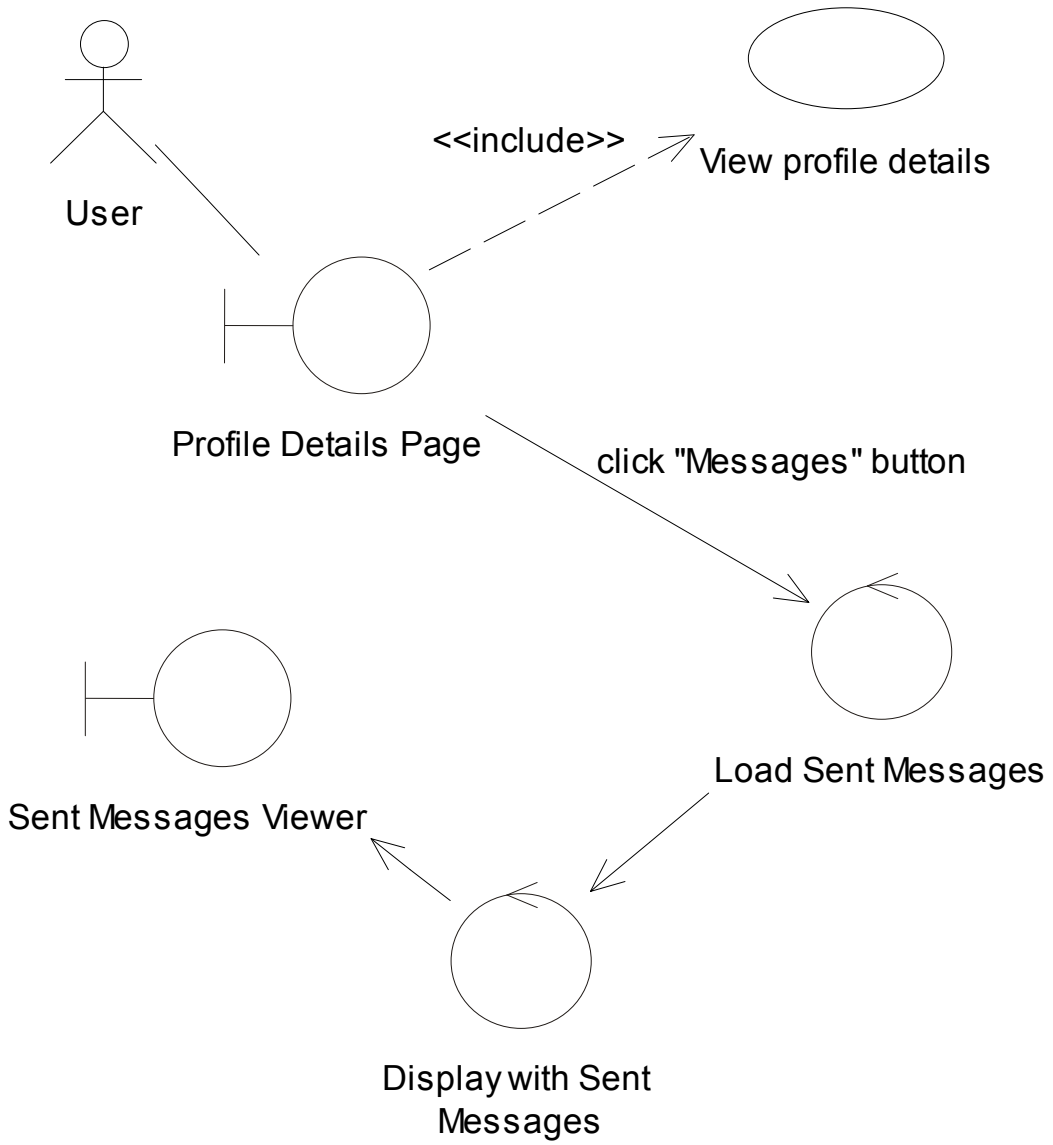
4-15 : Διάγραμμα ευρωστίας για την περίπτωση χρήσης «Ακολούθηση» & «Πάυση Ακολούθησης»

4.2.6.10 Διάγραμμα ευρωστίας : Προβολή Ακολούθων/ Ακολουθουμένων



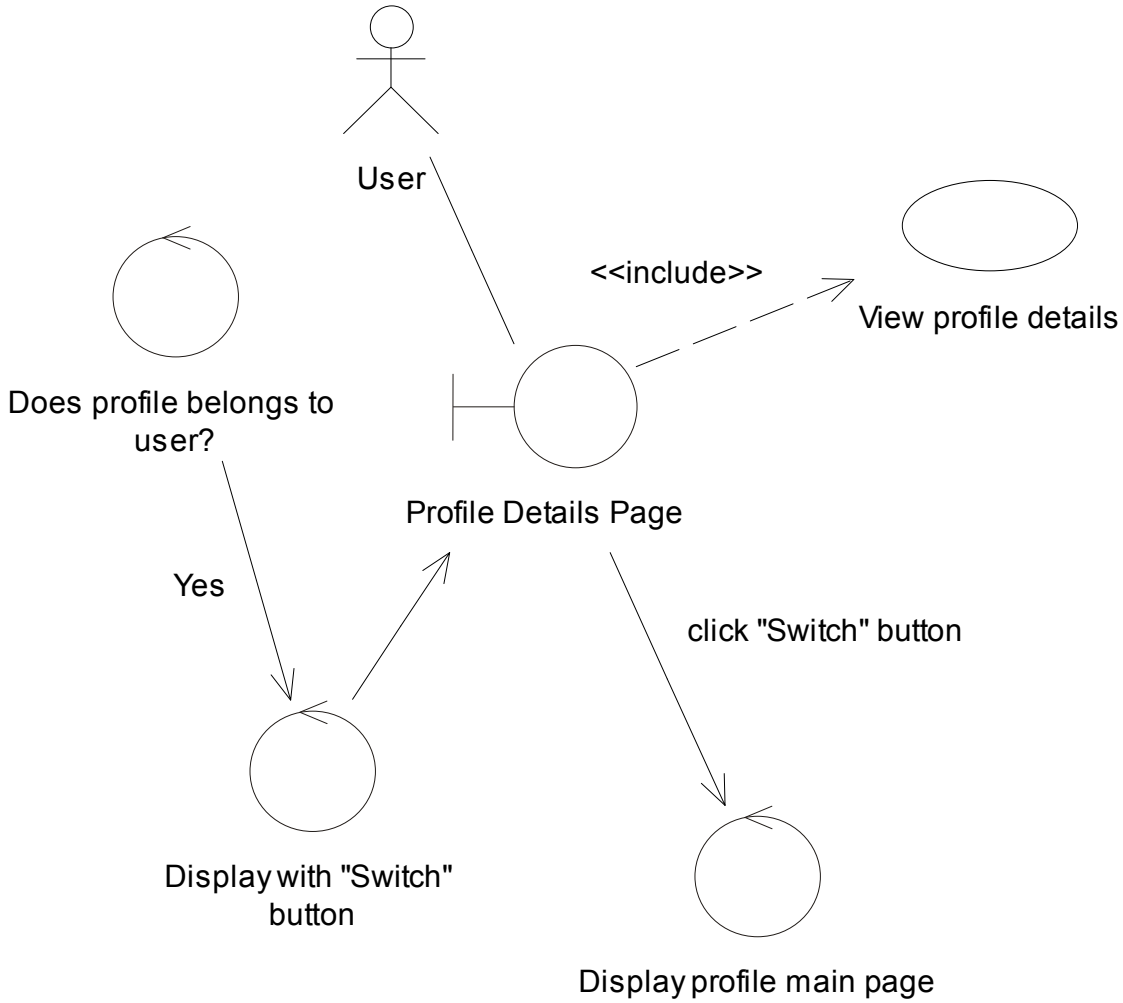
4-16: Διάγραμμα ευρωστίας για την περίπτωση χρήσης «Προβολή Ακολούθων/ Ακολουθουμένων»

4.2.6.11 Διάγραμμα ευρωστίας : Προβολή υποβληθέντων μηνυμάτων



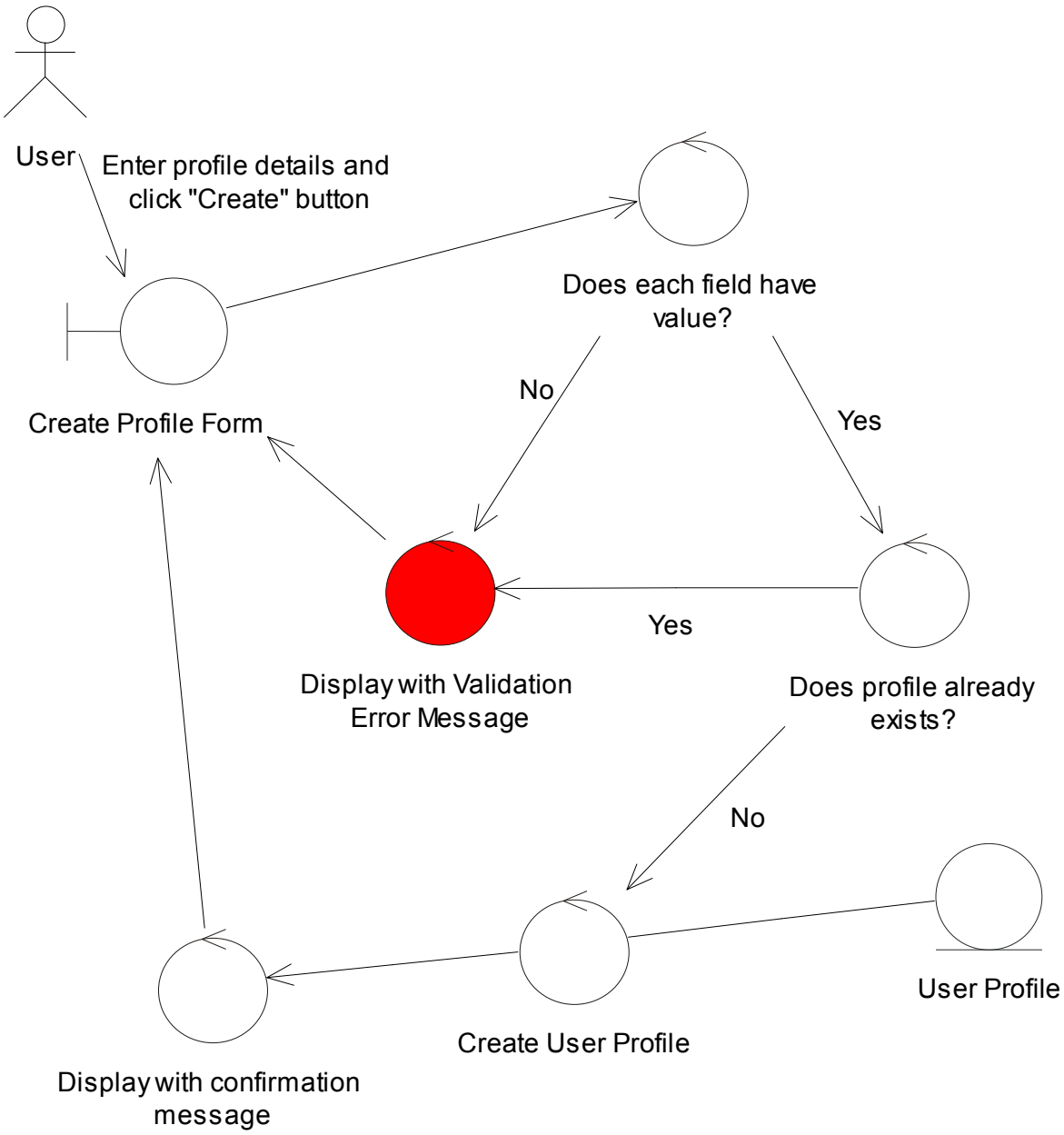
4-17: Διάγραμμα ευρωστίας για την περίπτωση χρήση «Προβολή υποβληθέντων μηνυμάτων»

4.2.6.12 Διάγραμμα ευρωστίας : Εναλλαγή προφίλ



4-18: Διάγραμμα ευρωστίας για την περίπτωση χρήσης «Εναλλαγή προφίλ»

4.2.6.13 Διάγραμμα ευρωστίας : Δημιουργία προφίλ



4-19: Διάγραμμα ευρωστίας για την περίπτωση χρήσης «Δημιουργία προφίλ»

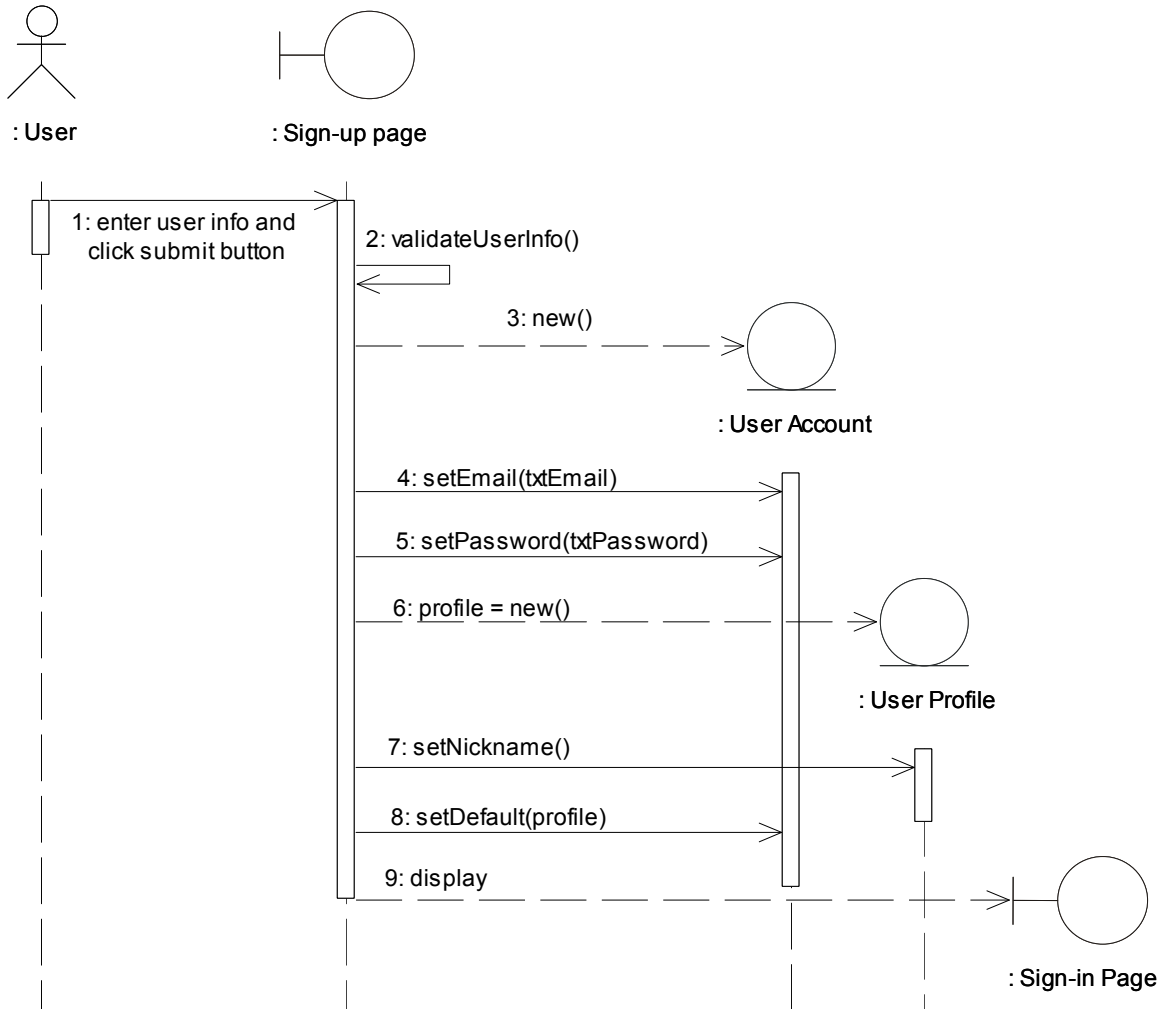
4.2.7 Αναλυτική Σχεδίαση

Μόλις ολοκληρωθεί η ανάλυση ευρωστίας, και έχει αφού έχει διεξαχθεί μια προκαταρκτική εξέταση της μελέτης, φτάνει η ώρα για να ξεκινήσει μια λεπτομερής σχεδιαστική προσπάθεια. Αν η αρχική σχεδίαση αφορά περισσότερο την ανακάλυψη κλάσεων, η αναλυτική σχεδίαση αφορά περισσότερο την κατανομή συμπεριφοράς.

Στο στάδιο της αρχικής σχεδίασης έγιναν κάποιες ανεπίσημες, πρώτες εικασίες για το πώς οι κλασεις θα αλληλεπιδρούν μεταξύ τους. Στο στάδιο της λεπτομερής σχεδίασης αυτές οι δηλώσεις θα πρέπει να γίνουν πιο ακριβής, ώστε να μετατραπούν σε ένα λεπτομερές σχέδιο που θα λειτουργέι εντός τις αρχιτεκτονικής που έχει οριστεί. Για να οδηγηθούμε σε αναλυτικό σχεδιασμό θα χρησιμοποιηθούν διαγράμματα ακολουθίας.

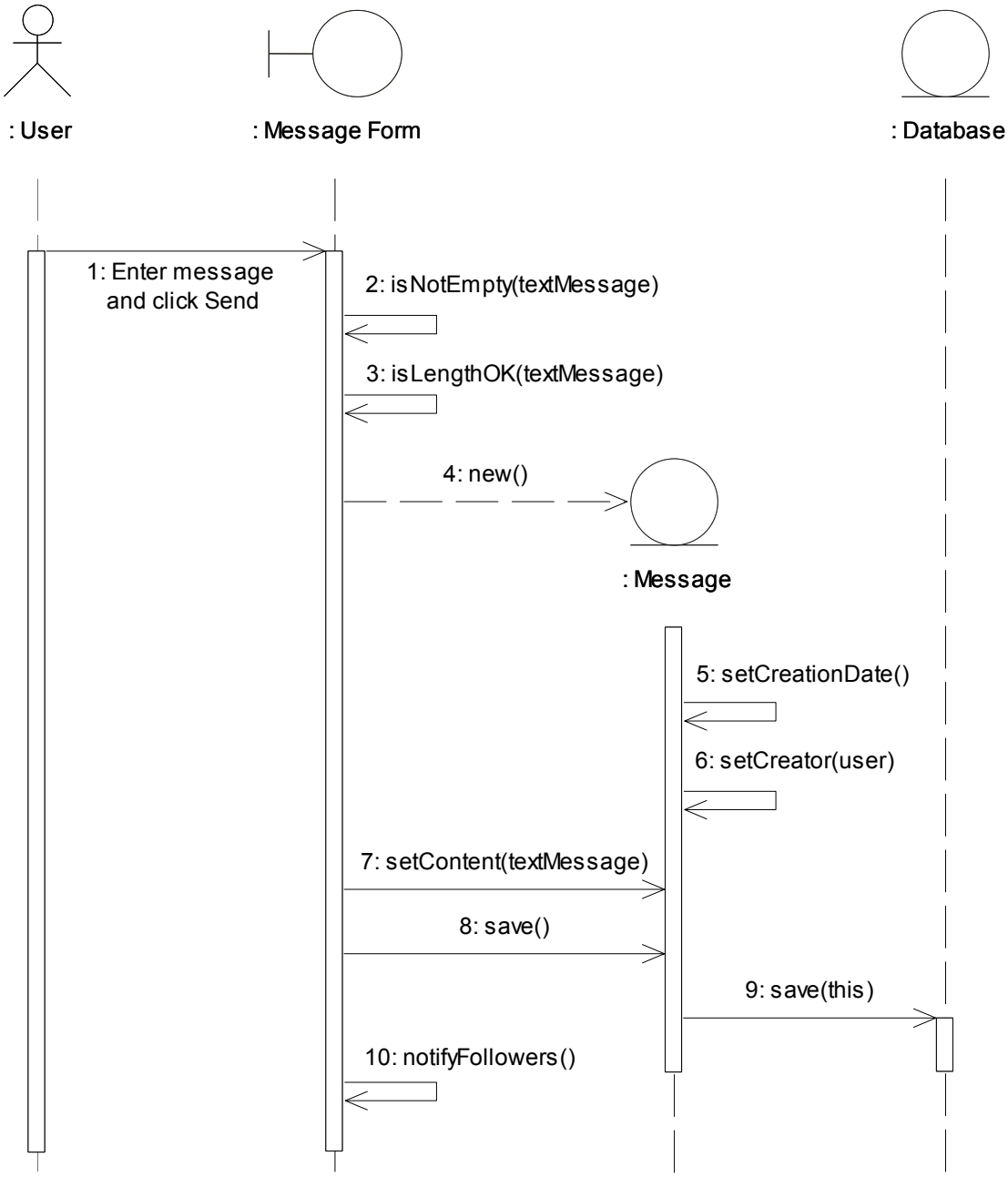
4.2.8 Διαγράμματα Ακολουθίας

4.2.8.1 Διάγραμμα ακολουθίας : Εγγραφή στο σύστημα



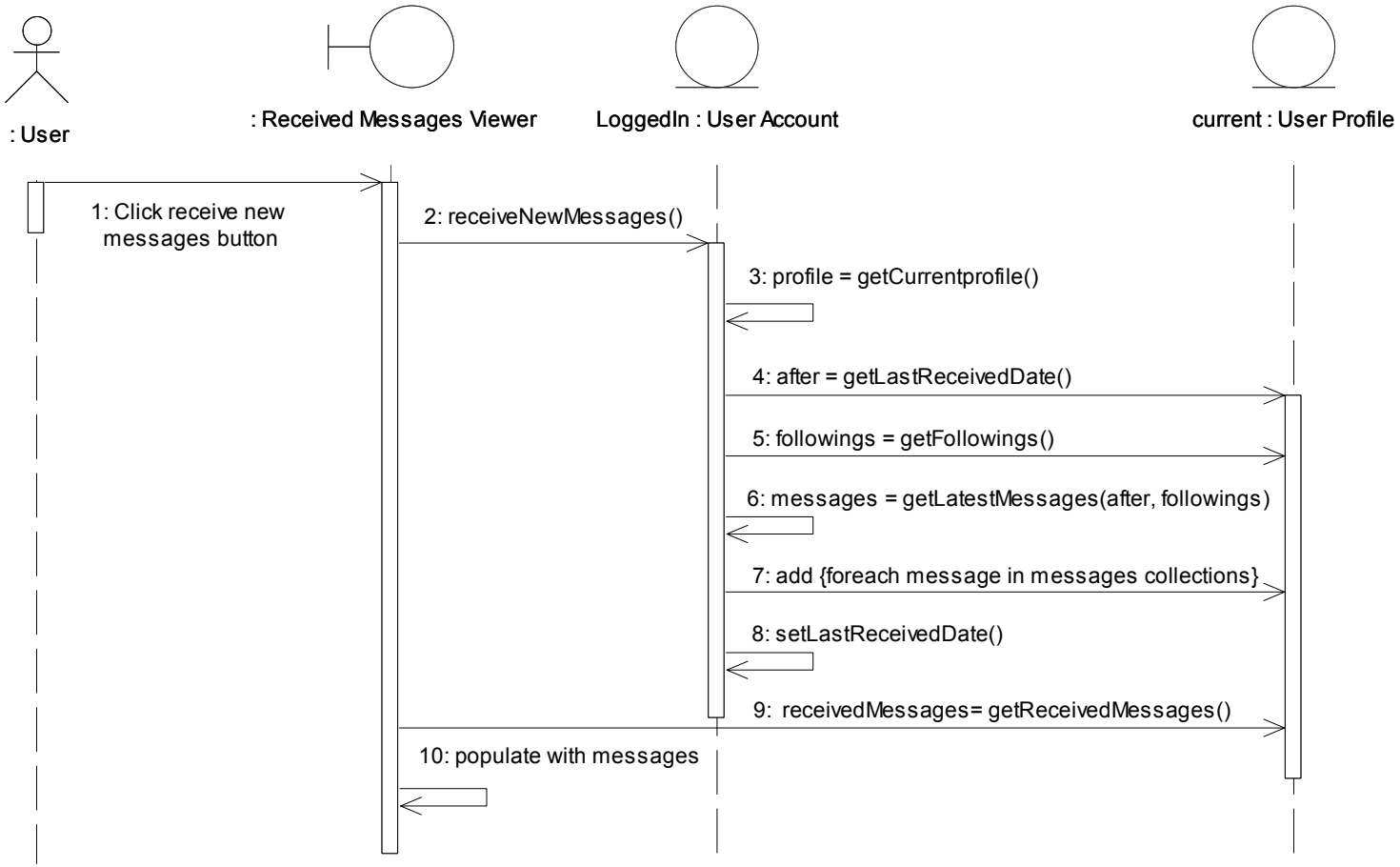
4-20: Διάγραμμα ακολουθίας για την περίπτωση χρήσης «Εγγραφή στο σύστημα»

4.2.8.2 Διάγραμμα ακολουθίας : Υποβολή μηνύματος



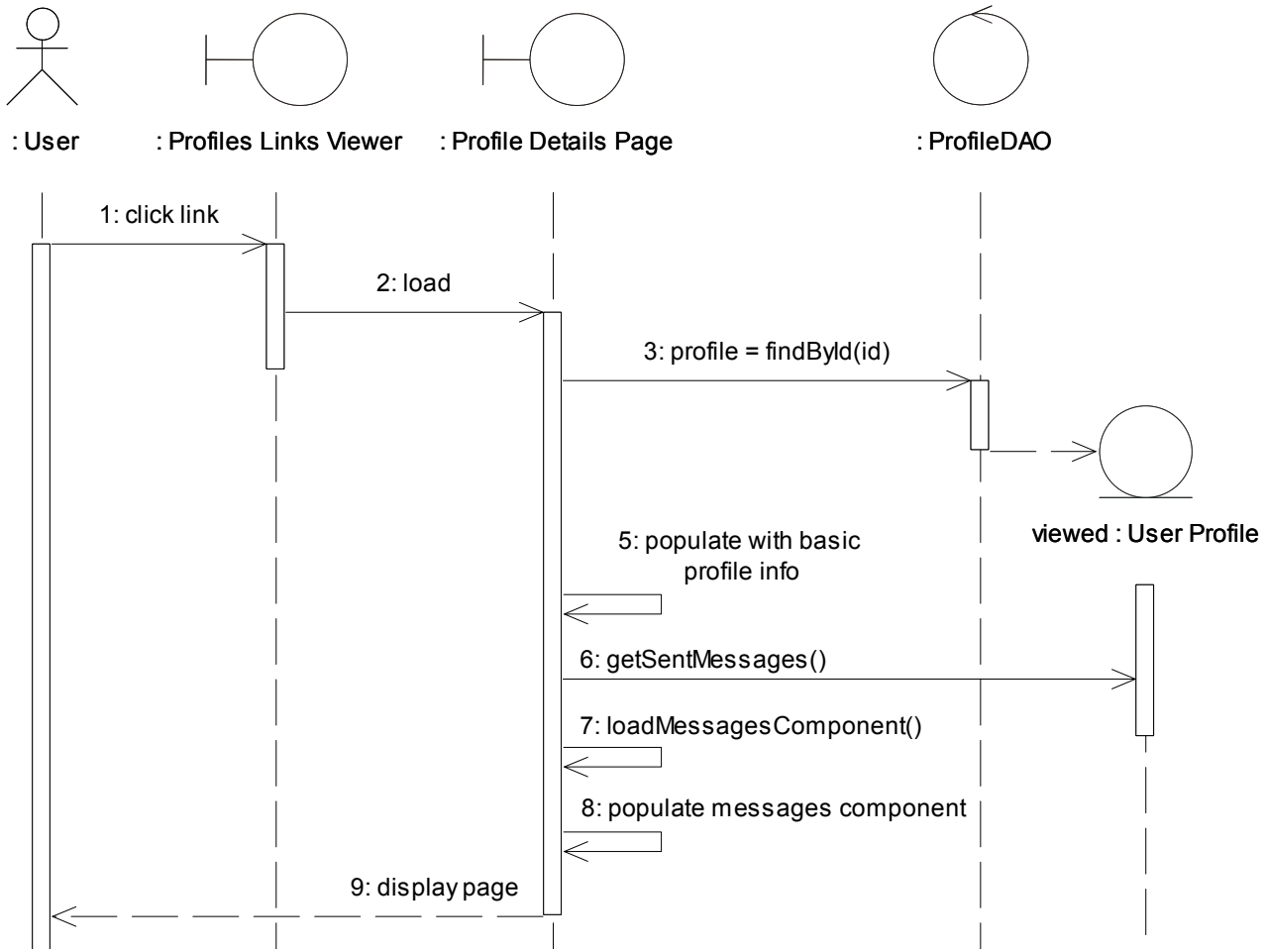
4-21: Διάγραμμα ακολουθίας για την περίπτωση χρήσης «Υποβολή μηνύματος»

4.2.8.3 Διαγραμμα ακολουθίας : Λήψη νέων μηνυμάτων



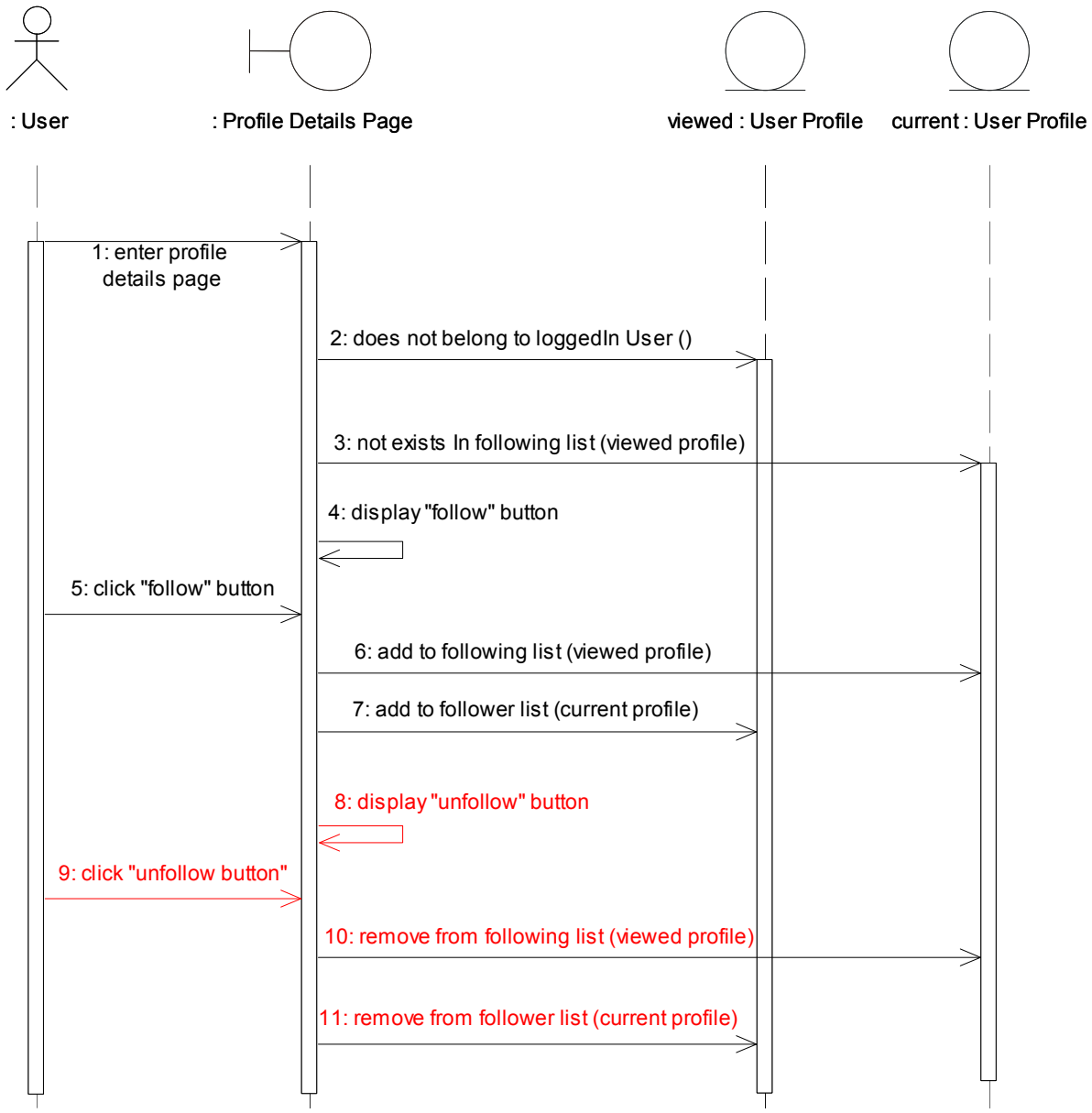
4-22: Διαγραμμα ακολουθίας για την περίπτωση χρήσης «Λήψη νέων μηνυμάτων»

4.2.8.4 Διαγραμμα ακολουθίας : Προβολή πληροφοριών προφίλ



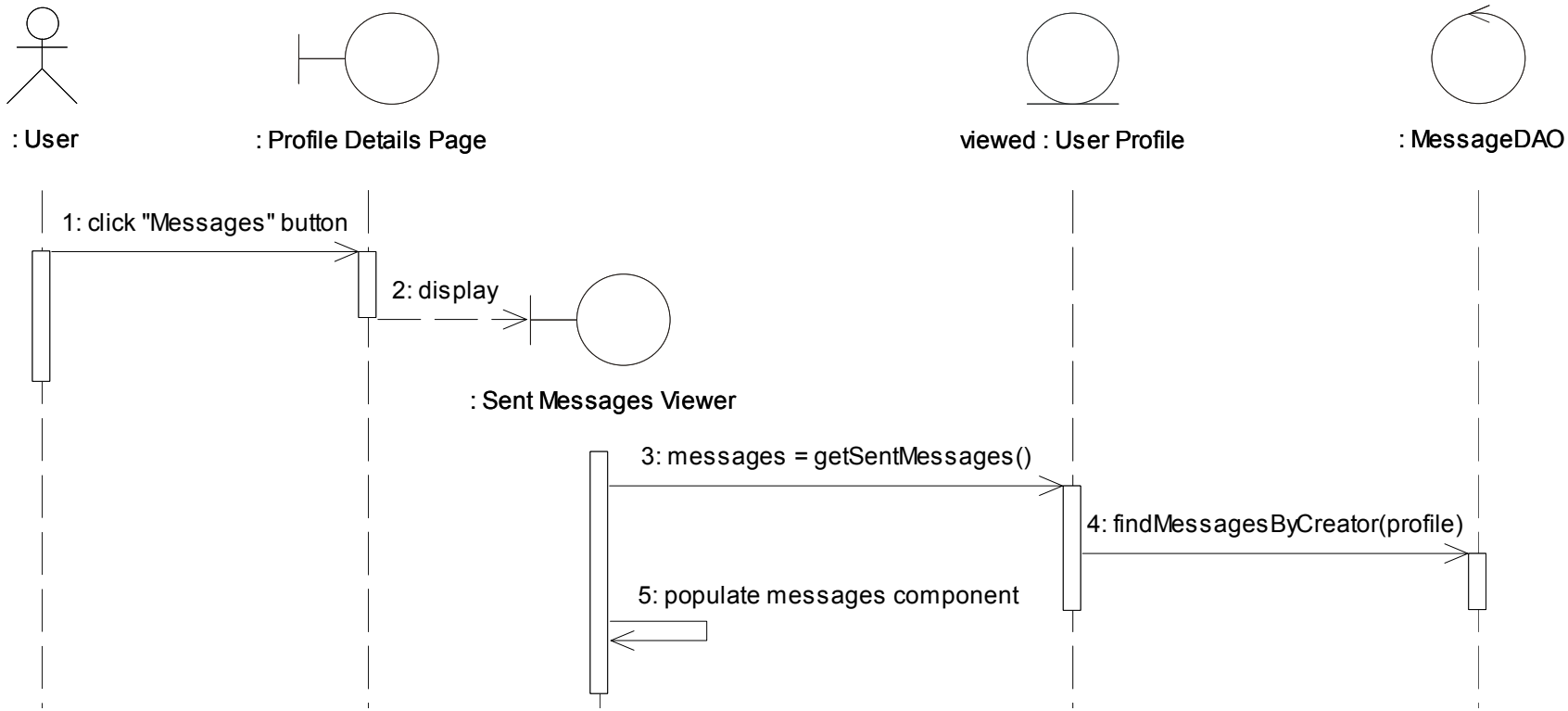
4-23: Διαγραμμα ακολουθίας για την περίπτωση χρήσης «Προβολή πληροφοριών προφίλ»

4.2.8.5 Διάγραμμα ακολουθίας : Ακολούθηση/ Παύση Ακολούθησης



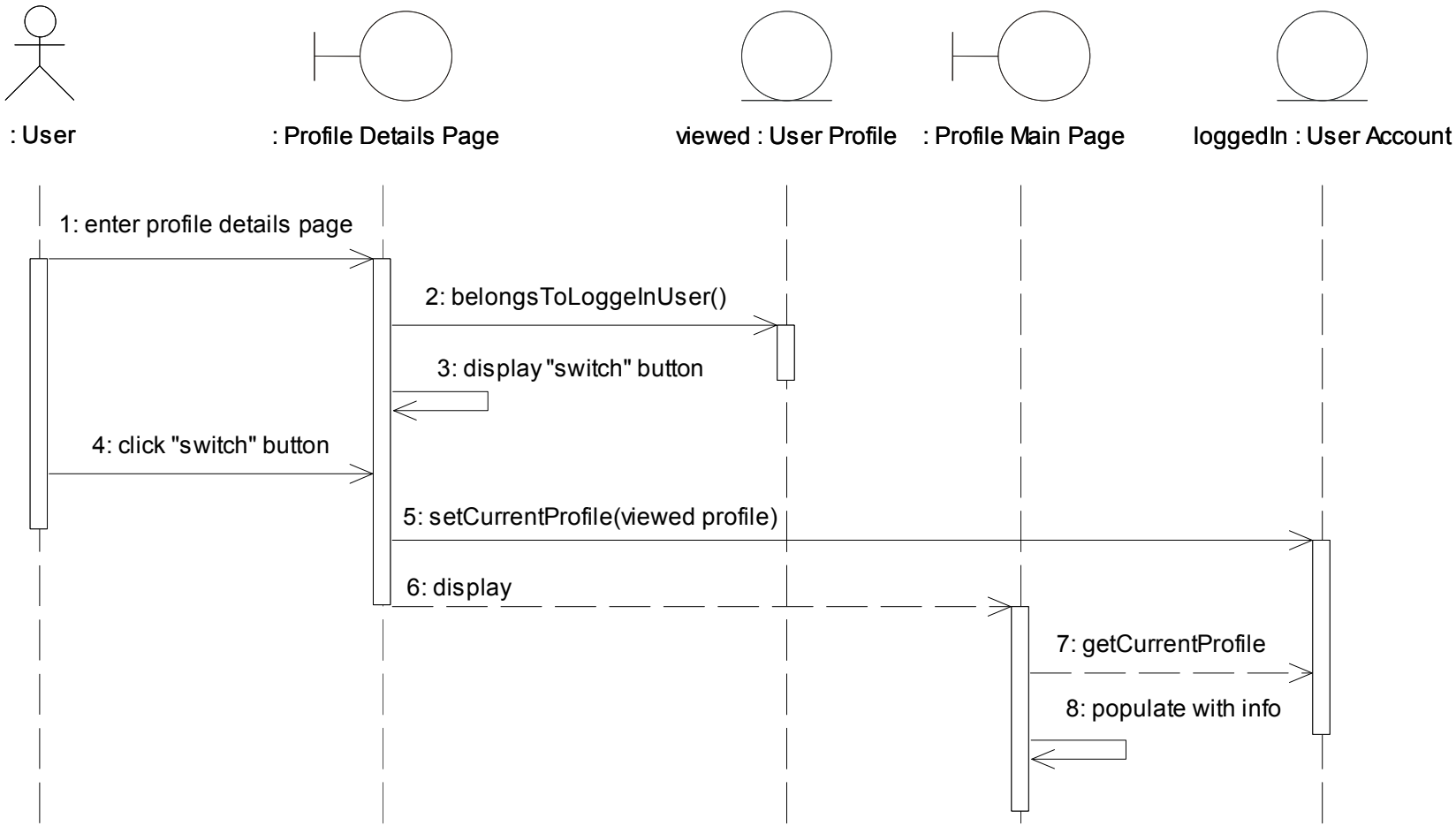
4-24: Διάγραμμα ακολουθίας για την περίπτωση χρήσης «Ακολούθηση/ Παύση Ακολούθησης»

4.2.8.6 Διάγραμμα ακολουθίας : Προβολή υποβληθέντων μηνυμάτων



4-25: Διάγραμμα ακολουθίας για την περίπτωση χρήσης «Προβολή υποβληθέντων μηνυμάτων»

4.2.8.7 Διάγραμμα ακολουθίας : Εναλλαγή προφίλ



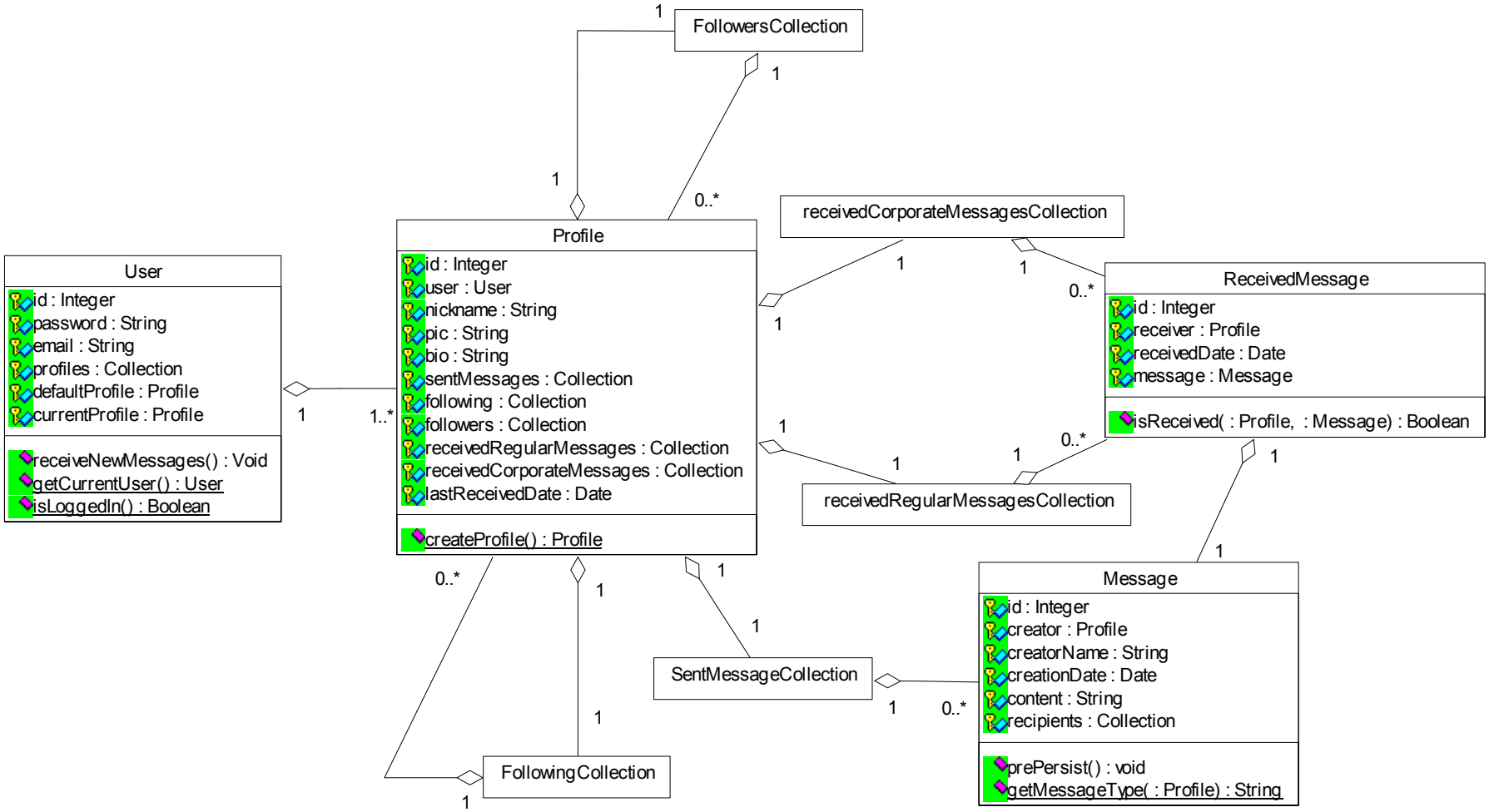
4-26: Διάγραμμα ακολουθίας για την περίπτωση χρήσης «Εναλλαγή προφίλ»

4.2.9 Διάγραμμα Κλάσεων

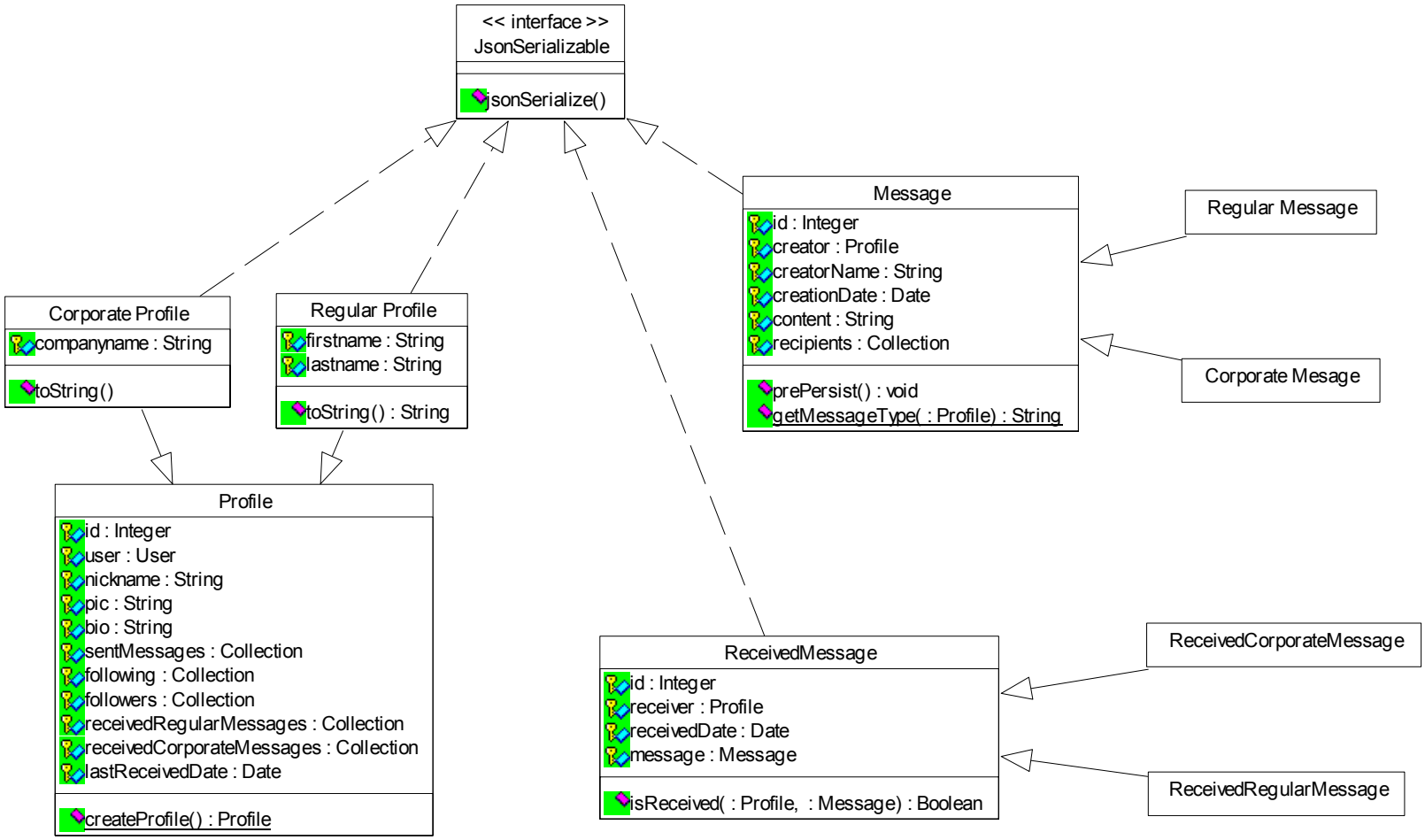
Ο κύριος σκοπός δημιουργίας των διαγραμμάτων ακολουθίας είναι η κατανομή της λειτουργικότητας (μεθόδων) στις κλάσεις του συστήματος και κατά δεύτερο λόγο ο εντοπισμός τυχόν κλάσεων, ιδιοτήτων και μεθόδων που δεν αναδείχθηκαν στο στάδιο της ανάλυσης. Στο

αναθεωρημένο διάγραμμα κλάσεων περιλαμβάνονται πλέον οι μέθοδοι κάθε κλάσης. Η λήψη ενός μηνύματος από ένα αντικείμενο μιας κλάσης σε ένα διάγραμμα ακολουθίας υποδηλώνει την ύπαρξη μιας μεθόδου στην κλάση που είναι ο αποδέκτης του μηνύματος. Η μέθοδος συνιστά τη λειτουργία που θα εκτελείται στην κλάση-αποδέκτη με τη λήψη του αντίστοιχου μηνύματος.

4.2.9.1 Αναθεωρημένο διάγραμμα κλάσεων



4-27: Διάγραμμα κλάσεων μοντέλο πεδίου προβλήματος (λαμβάνοντας υπόψη την πολλαπλότητα)



4-28: Διάγραμμα κλάσεων μοντέλο πεδίου προβλήματος (λαμβάνοντας υπόψη την κληρονομικότητα και την πραγμάτωση

5 Επίλογος

5.1 Σύνοψη και Συμπεράσματα

Σκοπός αυτής της εργασίας ήταν να αναλυθεί ο τρόπος με τον οποίο θα μπορούσε να αναπτυχθεί μία νέα γενιά διαδικτυακών εφαρμογών, οι οποίες θα χαρακτηρίζονται από συνεργατικότητα και ροή δεδομένων σε πραγματικό χρόνο.

Παρόλο που τέτοιου είδους εφαρμογές είχαν κάνει ήδη την εμφάνιση τους πολύ νωρίτερα από την εκπόνηση της παρούσας εργασίας, ωστόσο οι τεχνικές που χρησιμοποιούνταν ήταν προγραμματιστικά ιδιαίτερα δύστροπες ή απαιτούσαν τη χρήση προσθέτων (plugins) ή προωθούσαν την όχι και τόσο ορθολογική χρήση συστατικών του παγκόσμιού ιστού, όπως για παράδειγμα την ανορθόδοξη χρήση του πρωτοκόλλου HTTP.

Σε αυτή την εργασία πραγματοποιήθηκε μια πολύ ενδιαφέρουσα και εποικοδομητική έρευνα σχετικά με το μέλλον του διαδικτύου, το οποίο θα αλλάξει όχι μόνο ως προς τον τρόπο που σχεδιάζονται οι εφαρμογές αλλά και ως προς την αλληλεπίδραση των χρηστών με αυτές. Σε αυτό βέβαια συμβάλει και το γεγονός ότι τα τελευταία τρία χρόνια αν και δεν έγινε αμέσως αντιληπτό από τους σχεδιαστές, υπήρξε μια τεχνολογική επανάσταση στο χώρο της βιομηχανίας λογισμικού όσον αφορά στις εφαρμογές διαδικτύου. Αυτή η τεχνολογική έξαρση επιτρέπει όσο ποτέ άλλοτε, τη δυνατότητα ανάπτυξης καινοτόμων και συνεργατικών εφαρμογών πραγματικού χρόνου.

Η HTML5 με την ενσωματωμένη τεχνολογία WebSocket αντιπροσωπεύει το επόμενο εξελικτικό βήμα στην επικοινωνία στο διαδίκτυο σε σύγκριση με τις τεχνικές Comet και Ajax. Το σύστημα λογισμικού Node.JS, μαζί με τη βοηθητική βιβλιοθήκη Socket.IO παρέχει την δυνατότητα στους σχεδιαστές στο εγγύς μέλλον, να αναπτύσσουν τη νέα τάξη δυναμικών, διαδραστικών και «ζωντανών» εφαρμογών.

Η βάση δεδομένων MongoDB αποτελεί ιδανική και αξιόπιστη λύση για τύπους δεδομένων και περιπτώσεις χρήσης που χαρακτηρίζουν εφαρμογές κοινωνικής δικτύωσης, όπου οι χρήστες και η κυκλοφορία των δεδομένων αυξάνονται συνεχώς, καθιστώντας με αυτό το τρόπο επιβεβλημένη την ανάγκη κλιμάκωσης για την επίτευξή ταχύτητας σε εντολές εισόδου/ εξόδου.

Επιπλέον, αν και δεν αναλύθηκε εκτενώς στην παρούσα εργασία, ιδιαίτερο ενδιαφέρον είχε η αναφορά και η εφαρμογή του μοντέλου δημοσίευση/ συνδρομή (publish /subscribe), το οποίο αποτελεί σημαντική σχεδιαστική εξέλιξη για την υλοποίηση μηχανισμών ενημέρωσης και ειδοποιήσεων, ιδιαίτερα σε συστήματα που απαιτείται κλιμάκωση και υψηλή διαθεσιμότητα.

Τέλος, παρά το γεγονός ότι στην παρούσα εργασία παρουσιάζεται ένα αρκετά ενδιαφέρον θέμα, εξακολουθούν να υπάρχουν ανοιχτά θέματα, τα οποία πρέπει να αξιολογηθούν σε περαιτέρω έρευνα, όπως θέματα συνολικής απόδοσης και ασφάλειας.

Στην παρακάτω ενότητα παρακάτω ενότητα παρουσιάζονται ορισμένες βελτιώσεις και επεκτάσεις που επιδέχεται η εφαρμογή της παρούσας διπλωματικής εργασίας.

5.2 Μελλοντικές βελτιώσεις / επεκτάσεις

5.2.1 Αύξηση επιδόσεων της ΒΔ MongoDB

5.2.1.1 Δημιουργία ευρετηρίων (Indexing)

Με τη δημιουργία ευρετηρίων επιτυγχάνεται βελτίωση στην απόδοση των ερωτημάτων (queries). Στη βάση δεδομένων MongoDB, το ευρετήριο είναι μια ειδική δομή δεδομένων (B-trees), η οποία διατηρεί πληροφορίες σχετικά με τις τιμές συγκεκριμένων πεδίων στο έγγραφο μιας συλλογής. Όταν γίνεται ερώτημα στα πεδία αυτης της συλλογής, η mongoDB εξετάζει αυτή τη δομή δεδομένων προκειμένου τα ταξινομήσει και να διατάξει γρήγορα τα έγγραφα παρέχοντας τυχαία προσπέλαση στα δεδομένα της συλλογής.

5.2.1.2 Καταμερισμός (Sharding)

Η βάση δεδομένων MongoDB υποστηρίζει πραγματική οριζόντια κλιμάκωση δεδομένων μέσω της τεχνικής του καταμερισμού (Sharding). Σε συνδυασμό με το σύστημα δημιουργίας και κατανομής αντιγράφων (Replication), η MongoDB παρέχει πλήρη συνέπεια και υψηλής διαθεσιμότητας οριζόντια κλιμάκωση.

Καταμερισμός (Sharding) είναι η πρακτική κατάτμησης των δεδομένων, έτσι ώστε τα δεδομένα να είναι διαμοιρασμένα ανάμεσα σε πολλούς κόμβους. Πρόκειται για μια πολύ διαδεδομένη τεχνική, ιδιαίτερα σε ιστότοπους υψηλής επισκεψιμότητας και κυκλοφορίας δεδομένων.

Έτσι με αυτό το τρόπο, η MongoDB, αυτομάτως :

- Χωρίζει και διαμοιράζει τα δεδομένα ομοιόμορφα.
- Εξισορροπεί τα δεδομένα, όσο δημιουργούνται νέα δεδομένα.
- Αναδιανέμει τα δεδομένα όταν προστίθενται περισσότεροι κόμβοι.

Πλεονέκτημα της πρακτικής του καταμερισμού είναι η αύξηση των επιδόσεων της MongoDB σε ενέργειες ανάγνωσης και αποθήκευσης δεδομένων, καθώς αυτές πλέον, θα διαμοιράζονται σε πολλούς διαφορετικούς κόμβους, ενώ παράλληλα δεν απαιτείται να γίνει καμιά αλλαγή στον πηγαίο κώδικα της εφαρμογής. Αξίζει να σημειωθεί πως η διαδικτυακή εφαρμογή κοινωνικής δικτύωσης foursquare, εφαρμόζει αυτή τη τεχνική (Heymann, 2011) .

5.2.2 Επίγνωση θέσης του χρήστη (Location Aware)

Στην εφαρμογή θα ενσωματώνεται μηχανισμός επίγνωσης θέσης του χρήστη. Με αυτό τον τρόπο η εφαρμογή θα λαμβάνει ως είσοδο την γεωγραφική τοποθεσία του χρηστή με σκοπό την οπτική αναπαράσταση της θέσης του, στο περιβάλλον του χρήστη ή την παροχή υπηρεσιών προς τον χρήστη με βάση την θέση του, όπως προβολή άλλως χρηστών (followers, followings) που βρίσκονται εντός των ορίων καθορισμένης από τον χρήστη, ακτίνας.

Η βάση δεδομένων MongoDB έχει τις δυνατότητες για την αποθήκευση, την αναζήτηση και τη σύγκριση παραμέτρων (γεωγραφικό πλάτος/ μήκος) γεωγραφικής θέσης. Επίσης η MongoDB έχει δυνατότητα δημιουργίας ευρετηρίου γεωχωρικών δεδομένων (Geospatial Indexing), που την καθιστά ικανή να εκτελεί αποτελεσματικά ερωτήματα βάσει τοποθεσίας.

5.2.3 Χρήση πρωτοκόλλου OAuth (Open Authentication)

Η εφαρμογή G-nnect θα μπορεί να υλοποιεί ένα σύστημα το οποίο επιτρέπει στους χρήστες του, οι οποίοι έχουν ήδη κωδικούς πρόσβασης σε άλλες εφαρμογές κοινωνικής δικτύωσης (Facebook,

Twitter, MySpace, κ.α), να αποκτούν χωρίς πρόσθετες διαδικασίες πρόσβαση στην εφαρμογή. Αντίστοιχοι μηχανισμοί χρησιμοποιούνται σήμερα ευρέως στο διαδίκτυο, επιτρέποντας στους χρήστες να διασυνδέονται σε έναν δικτυακό τόπο χωρίς να χρειάζεται να εγγραφούν συγκεκριμένα σε αυτόν, αλλά εναλλακτικά να χρησιμοποιούν τα διακριτικά (username/ password) που ήδη έχουν έχουν σε κάποιον άλλον ιστότοπο. Για την παραπάνω διαδικασία χρησιμοποιείται το πρότυπο OAuth (Open Authentication).

Το OAuth είναι ένα πρωτόκολλο ταυτοποίησης χρηστών, επιτρέποντας σε χρήστες να μοιράζονται προσωπικά δεδομένα πιστοποίησης που διατηρούν σε έναν ιστότοπο στο διαδίκτυο, με άλλες διαδικτυακές εφαρμογές, χωρίς όμως την ανταλλαγή αυτών των στοιχείων.

6 Βιβλιογραφία

Aresh, A., 2012. *Real-Time Voice Chat Realized in JavaScript,* Lulea: Department of Computer Science, Electrical and Space Engineering.

Chodorow, K. & Dirolf, M., 2010. *MongoDB: The Definitive Guide.* s.l.:O'Reilly Media.

Citrusbyte, n.d. *Pub/Sub – Redis.* [Online]
Available at: http://redis.io/topics/pubsub

Fette, I. & Melnikov, A., 2011. *RCF 6455 The WebSocket Protocol.* [Online]
Available at: http://tools.ietf.org/html/rfc6455

Heymann, H., 2011. *Foursquare Customer Success with MongoDB.* [Online]
Available at: http://www.10gen.com/customers/foursquare

Höfler, T., 2013. *Enabling realtime collaborative data-intensive Web Applications - A case study using server-side JavaScript,* München: Fakultät für Informatik Der Technischen Universität München.

Holmstad, Ø., 2011. *Real-time annotation of video streams using staged processing,* s.l.: s.n.

Joyent, Inc, n.d. *node.js.* [Online]
Available at: http://nodejs.org/

Paudyal, U., 2011. *Scalable Web Application using Node.JS and CouchDB,* Uppsala: Uppsala University department of Information Technology.

PETER SÖDERGREN, BJÖRN ENGLUND, 2011. *Investigating NoSQL from a SQL Perspective,* Stockholm: Royal Institute of Technology.

Phil McCarthy, Dave Crane, 2008. *Comet and Reverse Ajax: The Next-Generation Ajax 2.0.* s.l.:Apress.

Rohit, R., 2013. *Socket.io Real-time Web Application Development.* Birmingham: Packt Publishing.

Russell, A., 2006. *Comet: Low Latency Data for the Browser.* [Online]
Available at: http://infrequently.org/2006/03/comet-low-latency-data-for-the-browser/

Teixeira, P., 2012. *Professional Node.js: Building Javascript Based Scalable Software.* s.l.:Wrox.

Wang, V., Salim, F. & Moskovits, P., 2013. *The Definitive Guide to HTML5 WebSocket.* s.l.:Apress.

Wikipedia., 2013d. *Java applet.* [Online]
Available at: http://en.wikipedia.org/wiki/Java_applet

Wikipedia, 2013a. *Αραβική Άνοιξη.* [Ηλεκτρονικό]
Available at: http://el.wikipedia.org/wiki/Αραβική_Άνοιξη

Wikipedia, 2013b. *Facebook.* [Online]
Available at: http://el.wikipedia.org/wiki/Facebook

Wikipedia, 2013c. *Twitter.* [Online]
Available at: http://el.wikipedia.org/wiki/Twitter

Wikipedia, 2013e. *Websocket.* [Online]
Available at: http://en.wikipedia.org/wiki/Websocket

Wikipedia, 2013f. *Server-sent events.* [Online]
Available at: http://en.wikipedia.org/wiki/Server-sent_events

Zend Technologies Ltd., 2013. *Programmer's Reference Guide - Zend Framework.* [Online]
Available at: http://framework.zend.com/manual/1.12/en/manual.html

Χατζηγεωργίου, Α., 2008. *Ανάπτυξη συστήματος λογισμικού βάσει της μεθοδολογίας ICONIX.* s.l.:ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ .

ΠΑΡΑΡΤΗΜΑ

Παράρτημα Α: Models

application/models/Documents/User.php

```
/**
 * @ODM\Document(db="gnnect", collection="users")
 */
class User
{
   /**
    * @ODM\Id
    */
   protected $id;

   /**
    * @ODM\String
    */
   protected $password;

   /**
    * @ODM\String
    */
   protected $email;

   /**
    * @ODM\ReferenceMany(targetDocument="Documents\Profile\Base", simple=true,
mappedBy="user")
    */
   protected $profiles;

   /**
    * @ODM\ReferenceOne(targetDocument="Documents\Profile\Base", simple=false)
    */
   protected $defaultprofile;

   protected $currentprofile;

   public function __construct() {

      $dm = \Zend_Registry::get('doctrine')->getDocumentManager();
      $this->profiles = new ArrayCollection();
      $dm->persist($this);
   }

   public function getId() {
      return $this->id;
   }

   public function setId($id) {
      $this->id = $id;
   }

   public function getPassword() {
      return $this->password;
   }

   public function setPassword($password) {
      $this->password = md5($password);
   }

   public function getEmail() {
```

```
        return $this->email;
    }

    public function setEmail($email) {
        $this->email = $email;
    }
    public function getProfiles() {
        return $this->profiles;
    }

    public function setProfiles($profiles) {
        $this->profiles = $profiles;
    }

    public function getDefaultprofile() {
        return $this->defaultprofile;
    }

    public function setDefaultprofile($profile) {

        if(!$this->profiles->contains($profile)) {
            $this->profiles->add($profile);
        }

        $this->defaultprofile = $profile;
    }

    public function getCurrentprofile() {

        $dm = \Zend_Registry::get('doctrine')->getDocumentManager();

        $session = new \Zend_Session_Namespace('profile');

        if(isset($session->currentprofileid))
        {
            $profile = $dm->find('Documents\Profile\Base', $session->currentprofileid);
        }
        else if ($this->getDefaultprofile() != null)
        {
            $profile = $this->getDefaultprofile();
            $this->setCurrentprofile($profile);
        }
        else
        {
            //throw exception
        }

        if ($this->getProfiles()->contains($profile))
        {
            return $profile;
        }
        else
        {
            //throw exception
        }
    }

    public function setCurrentprofile($profile) {
        $session = new \Zend_Session_Namespace('profile');
        $session->currentprofileid = $profile->getId();
    }

    public function receiveNewMessages()
    {
        $dm = \Zend_Registry::get('doctrine')->getDocumentManager();

        $profile = $this->getCurrentprofile();
```

```
        $trackedUsers = array_merge(
                array(),
                $profile->getFollowing()->getMongoData(),
                array(new \MongoId($profile->getId()))
        );

        /**
         * Get all messages since the last received date
         * and set as last received date the current
         */
        $trackedUserMessages = $dm->getRepository('Documents\Message\Base')
                            ->findMessages(array(
                               'creator'=> $trackedUsers,
                               'after'=>$profile->getLastReceivedDate()
                            ))->getQuery()->execute();

        foreach ($trackedUserMessages as $message)
        {
           if ($message instanceof \Documents\Message\Regular &&
!\Documents\Message\Received\Base::isReceived($this, $message))
           {
              $rmessage = new \Documents\Message\Received\Regular($profile, $message);

              $message->getRecipients()->add($rmessage);

              $profile->getReceivedRegularMessages()->add($rmessage);

              $dm->persist($rmessage);
           }
           else if ($message instanceof \Documents\Message\Corporate &&
!\Documents\Message\Received\Base::isReceived($this, $message))
           {
              $rmessage = new \Documents\Message\Received\Corporate($profile, $message);

              $message->getRecipients()->add($rmessage);

              $profile->getReceivedCorporateMessages()->add($rmessage);

              $dm->persist($rmessage);
           }
        }

        $profile->setLastReceivedDate(new \DateTime());

        $dm->flush();
     }

     public static function getCurrentUser()
     {
        $auth = \Zend_Registry::get('auth');
        $user =  $auth->getStorage()->read();
        return $user;
     }

     public static function isLoggedIn()
     {
        $auth = \Zend_Registry::get('auth');
         if ($auth->hasIdentity()){
            return true;
         }
         return false;
     }
}
```

application/models/Documents/Widget.php

```
/**
 * @ODM\Document(db="gnnect", collection="widgets")
```

```
 */
class Widget
{
    /**
     * @ODM\Id
     */
    protected $id;

    /**
     * @ODM\String
     */
    protected $name;

    /**
     * @ODM\String
     */
    protected $title;

    /**
     * @ODM\Boolean
     */
    protected $header;


    public function getName() {
        return $this->name;
    }

    public function setName($name) {
        $this->name = $name;
    }

    public function getTitle() {
        return $this->title;
    }

    public function setTitle($title) {
        $this->title = $title;
    }

    public function getHeader() {
        return $this->header;
    }

    public function setHeader($header) {
        $this->header = $header;
    }

    public function getId() {
        return $this->id;
    }
}
```

application/models/Documents/UserWidget.php

```
/**
 * @ODM\Document(db="gnnect", collection="users_widgets")
 */
class UserWidget
{
   /**
    * @ODM\Id
    */
   protected $id;

   /**
    * @ODM\String
    */
   protected $page;

   /**
    * @ODM\ReferenceOne(targetDocument="Documents\Widget", simple=true)
    */
   protected $widget;

   /**
    * @ODM\ReferenceOne(targetDocument="Documents\User", simple=true)
    */
   protected $user;

   /**
    * @ODM\Int
    */
   protected $column;

   /**
    * @ODM\Int
    */
   protected $order;

   protected $options;

   public function getPage() {
      return $this->page;
   }

   public function setPage($page) {
      $this->page = $page;
   }

   public function getWidget() {
      return $this->widget;
   }

   public function setWidget($widget) {
      $this->widget = $widget;
   }

   public function getColumn() {
      return $this->column;
   }

   public function setColumn($column) {
      $this->column = $column;
   }

   public function getOrder() {
      return $this->order;
   }
```

```
   public function setOrder($order) {
      $this->order = $order;
   }

   public function getUser() {
      return $this->user;
   }

   public function setUser($user) {
      $this->user = $user;
   }

   public function getOptions() {
      return $this->options;
   }

   public function setOptions($options) {
      $this->options = $options;
   }

   public static function loadUserWidget($widgetname, $defaultcolumn = 1, $defaultorder = 0,
$options = array())
   {
      $user = \Zend_Registry::get('currentUser');

      $dm = \Zend_Registry::get('doctrine')->getDocumentManager();

      $widget = $dm->getRepository('Documents\Widget')
               ->findOneBy(array("name"=>$widgetname));

      if (!isset($widget)){
         throw new \Exception("WidgetNotFound");
      }

      $request = \Zend_Controller_Front::getInstance()->getRequest();

      if (isset($options['pageName'])) {
         //
      }
      else {
         $options['pageName'] = $request->getModuleName() . '-' . $request-
>getControllerName() . '-' . $request->getActionName();
      }

      $userWidget = $dm->getRepository('Documents\UserWidget')
                  ->findOneBy(array(
                    "page" => $options['pageName'],
                    "user"  => $user->getId(),
                    "widget"=> $widget->getId())
                    );

      if (isset($userWidget))
      {
         $userWidget->setOptions($options);
         return $userWidget;
      }
      else
      {
         $class = __CLASS__;
         $userWidget = new $class();
         $userWidget->setPage($options['pageName']);
         $userWidget->setWidget($widget);
         $userWidget->setUser($user);
         $userWidget->setColumn($defaultcolumn);
         $userWidget->setOrder($defaultorder);
         $userWidget->setOptions($options);
```

```
            $dm->persist($userWidget);
            $dm->flush();

            return $userWidget;
        }
    }
}
```

application/models/Documents/Profile/Base.php

```
/**
 * @ODM\MappedSuperclass
 * @ODM\Document(db="gnnect", collection="profiles",
repositoryClass="Repositories\ProfileRepository")
 * @ODM\InheritanceType("SINGLE_COLLECTION")
 * @ODM\DiscriminatorField(fieldName="type")
 * @ODM\DiscriminatorMap({
 *  "regular"="Documents\Profile\Regular",
 *  "corporate"="Documents\Profile\Corporate"
 * })
 */
abstract class Base
{
    /**
     * @ODM\Id
     */
    protected $id;

    /**
     * @ODM\ReferenceOne(targetDocument="Documents\User", simple=true,
inversedBy="profiles")
     */
    protected $user;

    /**
     * @ODM\String
     */
    protected $nickname;

    /**
     * @ODM\String
     */
    protected $profilepic;

    protected $profilepiclink;

    /**
     * @ODM\String
     */
    protected $bio;

    /**
     * @ODM\String
     */
    protected $website;

    /**
     * @ODM\ReferenceMany(targetDocument="Documents\Message\Base", simple=true,
mappedBy="creator")
     */
    protected $sentMessages;

    /**
     * @ODM\ReferenceMany(targetDocument="Documents\Profile\Base", simple=true)
     */
    protected $following;

    /**
     * @ODM\ReferenceMany(targetDocument="Documents\Profile\Base", simple=true)
     */
    protected $followers;

    /**
     * @ODM\ReferenceMany(targetDocument="Documents\Message\Received\Regular",
```

```
simple=true, mappedBy="user")
    */
   protected $receivedRegularMessages;

   /**
    * @ODM\ReferenceMany(targetDocument="Documents\Message\Received\Corporate",
simple=true, mappedBy="user")
    */
   protected $receivedCorporateMessages;

   /**
    * @ODM\Date
    */
   protected $lastReceivedDate;

   public function __construct() {

      $dm = \Zend_Registry::get('doctrine')->getDocumentManager();

      $this->sentMessages = new ArrayCollection();

      $this->following = new ArrayCollection();
      $this->followers = new ArrayCollection();

      $this->receivedRegularMessages = new ArrayCollection();
      $this->receivedCorporateMessages = new ArrayCollection();

      $this->setLastReceivedDate(new \DateTime());

      $dm->persist($this);
   }

   public function getId() {
      return $this->id;
   }

   public function setId($id) {
      $this->id = $id;
   }

   public function getUser() {
      return $this->user;
   }

   public function setUser($user) {
      $this->user = $user;
   }

   public function getNickname() {
      return $this->nickname;
   }

   public function setNickname($nickname) {
      $this->nickname = $nickname;
   }

   public function getProfilepic() {
      if($this->profilepic === null) {
         return file_get_contents(APPLICATION_PATH.'/../public/img/default-avatar-
profile.png');
      } else {
         return $this->profilepic;
      }
   }

   public function setProfilepic($profilepic) {
      $this->profilepic = $profilepic;
   }
```

```
    public function getBio() {
        return $this->bio;
    }

    public function setBio($bio) {
        $this->bio = $bio;
    }

    public function getSentMessages() {
        return $this->sentMessages;
    }

    public function setSentMessages($sentMessages) {
        $this->sentMessages = $sentMessages;
    }

    public function getFollowing() {
        return $this->following;
    }

    public function setFollowing($following) {
        $this->following = $following;
    }

    public function getFollowers() {
        return $this->followers;
    }

    public function setFollowers($followers) {
        $this->followers = $followers;
    }

    public function getWebsite() {
        return $this->website;
    }

    public function setWebsite($website) {
        $this->website = $website;
    }

    public function getReceivedRegularMessages() {
        return $this->receivedRegularMessages;
    }

    public function setReceivedRegularMessages($receivedRegularMessages) {
        $this->receivedRegularMessages = $receivedRegularMessages;
    }

    public function getReceivedCorporateMessages() {
        return $this->receivedCorporateMessages;
    }

    public function setReceivedCorporateMessages($receivedCorporateMessages) {
        $this->receivedCorporateMessages = $receivedCorporateMessages;
    }

    public function getLastReceivedDate() {
        return $this->lastReceivedDate;
    }

    public function setLastReceivedDate($lastReceivedDate) {
        $this->lastReceivedDate = $lastReceivedDate;
    }

    public function getProfilepiclink()
    {
        $view = \Zend_Controller_Front::getInstance()->getParam('bootstrap')-
```

```
>getResource('view');
      $this->profilepiclink = $view->baseUrl('profile/' . $this->getId() . '/picture');
      return $this->profilepiclink;
   }

   public static function createProfile($user, $data)
   {
      if ($data['type'] === "corporate") {
         $classname = "\Documents\Profile\Corporate";
      }
      else if ($data['type'] === "regular") {
         $classname = "\Documents\Profile\Regular";
      }

      $profile = new $classname();

      if (isset($data['nickname']))
         $profile->setNickname($data['nickname']);
      else if (isset($data['companyname']))
        $profile->setCompanyname($data['companyname']);

      $profile->setUser($user);

      if(isset($profile)) {
         $dm = \Zend_Registry::get('doctrine')->getDocumentManager();
         $dm->persist($profile);
      }

      return $profile;
   }

   public function addFollowing($user){
      $this->following[] = $user;
   }

   public function jsonSerialize() {
      return array(
         'id'=> $this->getId(),
         'bio'=> $this->getBio(),
         'profilepiclink'=> $this->getProfilepiclink()
      );
   }
}
```

application/models/Documents/Profile/Regular.php

```
/**
 * @ODM\Document(repositoryClass="Repositories\ProfileRepository")
 */
class Regular extends \Documents\Profile\Base implements \JsonSerializable
{
    /**
     * @ODM\String
     */
    protected $firstname;

    /**
     * @ODM\String
     */
    protected $lastname;

    public function getFirstname() {
        return $this->firstname;
    }

    public function setFirstname($firstname) {
        $this->firstname = $firstname;
    }

    public function getLastname() {
        return $this->lastname;
    }

    public function setLastname($lastname) {
        $this->lastname = $lastname;
    }

    public function __toString() {
        return $this->nickname;
    }

    public function jsonSerialize() {
        return array(
            'nickname'=> $this->getNickname(),
            'firstname'=> $this->getFirstname(),
            'lastname'=>$this->getLastname(),
            'displayname'=> (string) $this,
            'type'=>'regular'
        )+ parent::jsonSerialize();
    }

}
```

application/models/Documents/Profile/Corporate.php

```php
/**
 * @ODM\Document(repositoryClass="Repositories\ProfileRepository")
 */
class Corporate extends \Documents\Profile\Base implements \JsonSerializable
{
    /**
     * @ODM\String
     */
    protected $companyname;

    public function getCompanyname() {
        return $this->companyname;
    }

    public function setCompanyname($companyname) {
        $this->companyname = $companyname;
    }

    public function __toString() {
        return $this->companyname;
    }

    public function jsonSerialize() {
        return array(
            'displayname'=> (string) $this,
            'type'=> 'corporate'
        ) + parent::jsonSerialize();
    }
}
```

application/models/Documents/Message/Base.php

```
/**
 * @ODM\Document(db="gnnect", collection="messages",
repositoryClass="Repositories\MessageRepository")
 * @ODM\InheritanceType("SINGLE_COLLECTION")
 * @ODM\DiscriminatorField(fieldName="type")
 * @ODM\DiscriminatorMap({
 *  "regular"="Documents\Message\Regular",
 *  "corporate"="Documents\Message\Corporate"
 * })
 */
abstract class Base implements \JsonSerializable
{
    /**
     * @ODM\Id
     */
    protected $id;

    /**
     * @ODM\ReferenceOne(targetDocument="Documents\Profile\Base", simple=false,
inversedBy="sentMessages", cascade={"all"})
     */
    protected $creator;

    /**
     * @ODM\String
     */
    protected $creatorname;

    /**
     * @ODM\Date
     */
    protected $creationDate;

    /**
     * @ODM\String
     */
    protected $content;

    /**
     * @ODM\ReferenceMany(targetDocument="Documents\ReceivedMessage", simple=true,
mappedBy="message", cascade={"all"})
     */
    protected $recipients;


    public function __construct() {

        $currentUser = \Zend_Registry::get('currentUser');

        $this->setCreator($currentUser->getCurrentprofile());
        $this->recipients = new ArrayCollection();
    }

    /** @ODM\PrePersist */
    public function prePersist()
    {
        $this->setCreationDate(new \DateTime());
    }

    public function getId() {
        return $this->id;
    }

    public function setId($id) {
        $this->id = $id;
```

```
    }

    public function getCreator() {
        return $this->creator;
    }

    public function setCreator($creator) {
        $this->creator = $creator;
    }

    public function getCreatorname() {
        return $this->creatorname;
    }

    public function setCreatorname($creatorname) {
        $this->creatorname = $creatorname;
    }

    public function getCreationDate() {
        return $this->creationDate->getTimestamp();
    }

    public function setCreationDate($creationDate) {
        $this->creationDate = $creationDate;
    }

    public function getContent() {
        return $this->content;
    }

    public function setContent($content) {
        $this->content = $content;
    }

    public function getRecipients() {
        return $this->recipients;
    }

    public function setRecipients($recipients) {
        $this->recipients = $recipients;
    }

    public static function getMessageType($profile) {

        if($profile instanceof \Documents\Profile\Corporate ) {
            return '\Documents\Message\Corporate';
        } else {
            return '\Documents\Message\Regular';
        }
    }

    // function called when encoded with json_encode
    public function jsonSerialize() {

        return array(
            'messageId' => $this->getId(),
            'creator'=> $this->getCreator(),
            'message'=> $this->getContent(),
            'createdAt'=> $this->getCreationDate()
        );
    }
}
```

application/models/Documents/Message/Regular.php

```
/**
 * @ODM\Document(repositoryClass="Repositories\MessageRepository")
 */
class Regular extends \Documents\Message\Base
{

}
```

application/models/Documents/Message/Corporate.php

```
/**
 * @ODM\Document(repositoryClass="Repositories\MessageRepository")
 */
class Corporate extends \Documents\Message\Base
{

}
```

application/models/Documents/Message/Received/Base.php

```
/**
 * @ODM\Document(db="gnnect", collection="received_messages",
repositoryClass="Repositories\MessageRepository")
 * @ODM\InheritanceType("SINGLE_COLLECTION")
 * @ODM\DiscriminatorField(fieldName="type")
 * @ODM\DiscriminatorMap({
 *  "regular"="Documents\Message\Received\Regular",
 *  "corporate"="Documents\Message\Received\Corporate"
 * })
 */
abstract class Base implements \JsonSerializable
{
    /**
     * @ODM\Id
     */
    protected $id;

    /**
     * @ODM\ReferenceOne(targetDocument="Documents\Profile\Base", simple=true)
     */
    protected $user;

    /**
     * @ODM\Date
     */
    protected $receivedDate;

    /**
     * @ODM\ReferenceOne(targetDocument="Documents\Message\Base", simple=true,
inversedBy="recipients")
     */
    protected $message;

    public function __construct($user, $message) {
        $this->setUser($user);
        $this->setMessage($message);
    }

    /** @ODM\PrePersist */
    public function prePersist()
    {
        $this->setReceivedDate(new \DateTime());
    }

    public function getId() {
        return $this->id;
    }

    public function setId($id) {
        $this->id = $id;
    }

    public function getUser() {
        return $this->user;
    }

    public function setUser($user) {
        $this->user = $user;
    }

    public function getReceivedDate() {
        return $this->receivedDate;
    }
```

```
    public function setReceivedDate($receivedDate) {
        $this->receivedDate = $receivedDate;
    }

    public function getMessage() {
        return $this->message;
    }

    public function setMessage($message) {
        $this->message = $message;
    }

    public static function isReceived($user, $message)
    {
        $dm = \Zend_Registry::get('doctrine')->getDocumentManager();

        $rmessages = $dm->getRepository(__CLASS__)->findBy(
            array("user"=>$user->getId(),
                "message"=>$message->getId())
        );

        if ($rmessages->count() > 0)
            return true;
        else if ($rmessages->count() == 0)
            return false;
    }

    public function jsonSerialize() {
        return $this->getMessage();
    }
}
```

application/models/Documents/Message/Received/Regular.php

```
/**
 * @ODM\Document(repositoryClass="Repositories\MessageRepository")
 */
class Regular extends \Documents\Message\Received\Base
{
    /**
     * @ODM\ReferenceOne(targetDocument="Documents\Profile\Base", simple=true,
inversedBy="receivedRegularMessages")
     */
    protected $user;
}
```

application/models/Documents/Message/Received/Corporate.php

```
/**
 * @ODM\Document(repositoryClass="Repositories\MessageRepository")
 */
class Corporate extends \Documents\Message\Received\Base
{
    /**
     * @ODM\ReferenceOne(targetDocument="Documents\Profile\Base", simple=true,
inversedBy="receivedCorporateMessages")
     */
    protected $user;
}
```

application/models/Repositories/ProfileRepository.php

```
class ProfileRepository Extends DocumentRepository
{
   public function findProfiles($criteria = array())
   {
      $qb = $this->createQueryBuilder($this->getDocumentName());
      $qb->field('following')->prime(true);

      if(isset($criteria['term']) && $criteria['term'] != '') {
         $qb->addOr($qb->expr()->field('nickname')->equals(new \MongoRegex('/.*'.
$criteria['term'] .'.*/i')))
            ->addOr($qb->expr()->field('companyname')->equals(new \MongoRegex('/.*'.
$criteria['term'] .'.*/i')));
      }

      if (isset($criteria['user']) && $criteria['user'] != '') {
         $qb->field('user')->equals($criteria['user']);
      }

      if (isset($criteria['type']) && $criteria['type'] != '') {
         $qb->field('type')->equals($criteria['type']);
      }

      if (isset($criteria['id']) && is_array($criteria['id'])) {
         $qb->field('id')->in($criteria['id']);
      }

      return $qb->getQuery()->execute();
   }
}
```

application/models/Repositories/MessageRepository.php

```
class MessageRepository Extends DocumentRepository
{
   public function getReceivedMessages($user, $criteria)
   {
      $qb = $this->createQueryBuilder($this->getDocumentName())
               ->field("user")->equals($user->getId());

      if(isset($criteria['after']) && $criteria['after'] != '') {

         if(isset($criteria['after']) && $criteria['after'] != '') {

            $after = $this->filterDate($criteria['after']);

            $qb->field('receivedDate')->gt($after);
         }
      }

      $qb->sort('receivedDate','desc');

      return $qb;
   }

   public function findMessages($criteria = array())
   {

      $qb = $this->createQueryBuilder($this->getDocumentName());

      if (isset($criteria['creator']) && is_array($criteria['creator'])) {

            $qb->field('creator.$id')->in($criteria['creator']);
      }
      else {
            $qb->field('creator.$id')->equals($criteria['creator']);
      }

      if(isset($criteria['after']) && $criteria['after'] != '') {

         $after = $this->filterDate($criteria['after']);

         $qb->field('creationDate')->gt($after);
      }

      $qb->sort('creationDate', 'desc');

      return $qb;
   }

   private function filterDate($date){

      if ($date instanceof \DateTime){
      }
      else {
         $date = new \MongoDate($date);
      }

      return $date;
   }
}
```

Παράρτημα Β: Controllers

application/controllers/AuthController.php

```
class AuthController extends Zend_Controller_Action
{
   protected $_dm;

   public function init()
   {
      $this->_dm = Zend_Registry::get('doctrine')->getDocumentManager();
   }

   public function indexAction()
   {
   }

   public function logoutAction()
   {
      Zend_Registry::get('auth')->clearIdentity();
      Zend_Session::destroy();
      $this->_helper->redirector('index', 'index');
   }
}
```

application/controllers/IndexController.php

```
class IndexController extends Zend_Controller_Action
{
   protected $_redirector = null;
   protected $_currentUser = null;
   protected $_currentProfile = null;

   public function init()
   {
      /* Initialize action controller here */
      $this->_redirector = $this->_helper->getHelper('Redirector');
   }

   public function preDispatch()
   {
      $auth = Zend_Registry::get('auth');

      if ($auth->hasIdentity())
      {
         $currentUser = Zend_Registry::get("currentUser");

         $this->_currentUser = $currentUser;
         $this->_currentProfile = $currentUser->getCurrentprofile();

         $this->view->currentUser = $currentUser;
         $this->view->currentProfile = $currentUser->getCurrentprofile();
      }
      else
      {
         $this->_redirector->gotoSimple('index', 'signin', 'default');
      }
   }

   public function indexAction()
   {
      $this->view->currentUser = $this->_currentUser;
      $this->view->currentProfile = $this->_currentProfile;
   }

   public function followersAction()
   {
      $this->view->currentUser = $this->_currentUser;
      $this->view->currentProfile = $this->_currentProfile;
   }

   public function followingAction()
   {
      $this->view->currentUser = $this->_currentUser;
      $this->view->currentProfile = $this->_currentProfile;
   }
}
```

application/controllers/ProfileController.php

```
class ProfileController extends Zend_Controller_Action
{
   protected $_profile = null;

   public function init()
   {
      $dm = Zend_Registry::get('doctrine')->getDocumentManager();

      $this->_profile = $dm->getRepository('Documents\Profile\Base')
                ->findOneBy(
                     array('id' => $this->getRequest()->getParam('uid'))
                 );
   }

   public function indexAction()
   {
      $this->_forward("tweets");
   }

   public function pictureAction()
   {
      $this->_helper->layout->disableLayout();
      $this->_helper->viewRenderer->setNoRender(TRUE);

      $this->getResponse()->setHeader('content-type', 'image/jpeg');

      echo $this->_profile->getProfilepic();
   }

   public function tweetsAction()
   {
      $this->view->profile = $this->_profile;
   }

   public function followersAction()
   {
      $this->view->profile = $this->_profile;
   }

   public function followingAction()
   {
      $this->view->profile = $this->_profile;
   }
}
```

application/controllers/ProfilesController.php

```
class ProfilesController extends Zend_Controller_Action
{
   protected $_flashMessenger = null;

   public function init()
   {
      /* Initialize action controller here */
      $this->_flashMessenger = $this->_helper->FlashMessenger;
   }

   public function indexAction()
   {
      //simple render the view script
   }

   public function addAction()
   {
      $dm = Zend_Registry::get('doctrine')->getDocumentManager();

      $user = Zend_Registry::get('currentUser');

      $profileType = $this->getRequest()->getParam('type');

      if ($this->getRequest()->isPost())
      {
         if ($profileType == 'regular'){
            $classname = 'Documents\Profile\Regular';
         }
         else if ($profileType == 'corporate'){
            $classname = 'Documents\Profile\Corporate';
         }

         $profile = new $classname();
         $profile->setUser($user);

         if ($profile instanceof Documents\Profile\Regular){
            $profile->setNickname($this->getRequest()->getParam("txtNickname"));
         }
         else if ($profile instanceof Documents\Profile\Corporate){
            $profile->setCompanyname($this->getRequest()->getParam("txtCompanyname"));
         }

         $profile->setBio($this->getRequest()->getParam("txtBio"));

         $dm->flush();

         $this->_flashMessenger->setNamespace('success')->addMessage('Your new profile
has been created successfully and is ready to use. You can create a new one');
      }

      $options = array(
         'url' => $this->view->url(array('module' => 'default', 'controller' => 'profiles', 'action'
=> 'add'), 'default', false),
         'profile'=> null,
         'label'=> 'Submit'
      );

      if ($profileType == 'regular'){
         $this->view->frmProfile = $this->view->partial('forms/_frmProfileRegular.phtml',
$options);
      }
      else if ($profileType == 'corporate'){
         $this->view->frmProfile = $this->view->partial('forms/_frmProfileCorporate.phtml',
$options);
      }
```

```
    }

    public function editAction()
    {
        $dm = Zend_Registry::get('doctrine')->getDocumentManager();

        $profile = $dm->getRepository('Documents\Profile\Base')->findOneBy(
              array('id' => $this->getRequest()->getParam('id'))
        );

        if ($profile instanceof Documents\Profile\Regular){
            $profileType = 'regular';
        }
        else if ($profile instanceof Documents\Profile\Corporate){
            $profileType = 'corporate';
        }

        if ($this->getRequest()->isPost())
        {
            if ($profile instanceof Documents\Profile\Regular){
                $profile->setNickname($this->getRequest()->getParam("txtNickname"));
            }
            else if ($profile instanceof Documents\Profile\Corporate){
                $profile->setCompanyname($this->getRequest()->getParam("txtCompanyname"));
            }

            $profile->setBio($this->getRequest()->getParam("txtBio"));

            $dm->flush();

            $this->_flashMessenger->setNamespace('success')->addMessage('Your profile has
been updated successfully');
        }

        $options = array(
            'url' => $this->view->url(array('module' => 'default', 'controller' => 'profiles', 'action'
=> 'edit'), 'default', false),
            'profile'=> $profile,
            'label'=> 'Update'
        );

        if ($profileType == 'regular'){
            $this->view->frmProfile = $this->view->partial('forms/_frmProfileRegular.phtml',
$options);
        }
        else if ($profileType == 'corporate'){
            $this->view->frmProfile = $this->view->partial('forms/_frmProfileCorporate.phtml',
$options);
        }
    }

    public function switchAction()
    {
        $user = Zend_Registry::get('currentUser');

        $dm = Zend_Registry::get('doctrine')->getDocumentManager();

        $profile = $dm->getRepository('Documents\Profile\Base')->findOneBy(
              array('id' => $this->getRequest()->getParam('to'))
        );

        if($user->getProfiles()->contains($profile)) {

            $user->setCurrentprofile($profile);

            $dm->flush();

            $this->_helper->redirector('index', 'index', 'default');
```

```
        } else {
            throw new Exception('ProfileAccessDenied');
        }
    }

    public function makedefaultAction()
    {
        $user = Zend_Registry::get('currentUser');

        $dm = Zend_Registry::get('doctrine')->getDocumentManager();

        $profile = $dm->getRepository('Documents\Profile\Base')->findOneBy(
              array('id' => $this->getRequest()->getParam('id'))
        );

        if($user->getProfiles()->contains($profile)) {

            $user->setDefaultprofile($profile);

            $dm->flush();

            $this->_helper->redirector('index', 'index', 'default');
        } else {
            throw new Exception('ProfileAccessDenied');
        }
    }

    public function deleteAction() {

        $user = Zend_Registry::get('currentUser');

        $dm = Zend_Registry::get('doctrine')->getDocumentManager();

        $profile = $dm->getRepository('Documents\Profile\Base')->findOneBy(
              array('id' => $this->getRequest()->getParam('id'))
        );

        if($user->getProfiles()->contains($profile)) {

            $dm->remove($profile);
            $dm->flush();

            $this->_helper->redirector('index', 'profiles', 'default');
        } else {
            throw new Exception('ProfileAccessDenied');
        }
    }

    public function searchAction(){
        $this->view->term = $this->getRequest()->getParam('term');
    }
}
```

application/controllers/SigninController.php

```
class SigninController extends Zend_Controller_Action
{
   protected $_redirector = null;
   protected $_dm = null;
   protected $_auth = null;
   protected $_flashMessenger = null;

   public function init()
   {
      $this->_redirector = $this->_helper->getHelper('Redirector');
      $this->_dm = Zend_Registry::get('doctrine')->getDocumentManager();
      $this->_auth = Zend_Registry::get('auth');
      $this->_flashMessenger = $this->_helper->FlashMessenger;
   }

   public function preDispatch()
   {
      if ($this->_auth->hasIdentity())
      {
         $identity = $this->_auth->getIdentity();

         if (isset($identity)) {
            $this->_redirector->gotoSimple('index', 'index', 'default');
         }
      }
   }

   public function indexAction()
   {
      $this->view->frmSignin = $this->view->partial('forms/_frmSignin.phtml');
   }

   public function processAction()
   {
      $request = $this->_request;

      if ($request->isPost())
      {
         $adapter = new Axiomes\Auth\Adapter\Odm();

         $adapter->setDocumentManager($this->_dm);

         $adapter->setClassName('Documents\User');
         $adapter->setIdentityField('email');
         $adapter->setCredentialField('password');
         $adapter->setIdentityValue($request->getParam("txtEmail"));
         $adapter->setCredentialValue($request->getParam("txtPassword"));
         $adapter->setCredentialTreatment('md5');

         $result = $this->_auth->authenticate($adapter);

         if($result->isValid())
         {
            $authStorage = $this->_auth->getStorage();
            $authStorage->write($result->getIdentity());
         }
         else
         {
            $this->_flashMessenger->setNamespace('error')->addMessage('Wrong Email and
password combination');
            $this->_redirector->gotoSimple('index', 'signin', 'default');
         }

         $this->_helper->redirector('index', 'index');
      }
```

```
    }
}
```

application/controllers/SignupController.php

```
class SignupController extends Zend_Controller_Action
{
    protected $_redirector = null;
    protected $_auth = null;
    protected $_dm = null;
    protected $_flashMessenger = null;

    public function init()
    {
        $this->_redirector = $this->_helper->getHelper('Redirector');
        $this->_auth = Zend_Registry::get('auth');
        $this->_dm = Zend_Registry::get('doctrine')->getDocumentManager();
        $this->_flashMessenger = $this->_helper->FlashMessenger;
    }

    public function preDispatch()
    {
        if ($this->_auth->hasIdentity())
        {
            $identity = $this->_auth->getIdentity();

            if (isset($identity)) {
                $this->_redirector->gotoSimple('index', 'index', 'default');
            }
        }
    }

    public function indexAction()
    {
        $options = array();
        $this->view->frmSignup = $this->view->partial('forms/_frmSignup.phtml', $options);
    }

    public function processAction()
    {
        if ($this->getRequest()->isPost())
        {

            $registedUser = new Documents\User();
            $registedUser->setEmail($this->getRequest()->getParam('txtEmail'));
            $registedUser->setPassword($this->getRequest()->getParam('txtPassword'));

            $registedUser->setDefaultprofile(
                  Documents\Profile\Base::createProfile(
                        $registedUser,
                        array('type'=>'regular', 'nickname'=>$this->getRequest()-
>getParam('txtNickname'))
                  )
            );

            $this->_dm->flush();

            $this->_flashMessenger->setNamespace('success')->addMessage('Your account has
been created successfully and is ready to use');

            $data = array("success"=>true);
            $this->_helper->json($data);
        }
```

```
    }
}
```

application/controllers/WidgetController.php

```
class WidgetController extends Zend_Controller_Action
{
    public function init()
    {
        $this->_helper->layout->disableLayout();
        $this->_helper->viewRenderer->setNoRender(TRUE);
    }

    public function moveAction()
    {
        $dm = Zend_Registry::get('doctrine')->getDocumentManager();

        foreach ($this->getRequest()->getParam('widgets') as $widget)
        {
            $info = preg_split("/[-]+/", $widget);

            $column = $info[2];
            $order = $info[3];
            $widgetname = $info[1];
            $pageName = $this->_request->getParam('pageName');

            $userwidget = Documents\UserWidget::loadUserWidget($widgetname, 1, 1,
array('pageName'=>$pageName));
            $userwidget->setColumn($column);
            $userwidget->setOrder($order);



            $dm->persist($userwidget);

            $dm->flush();
        }
    }
}
```

application/modules/ajax/controllers/MessagesController.php

```
class Ajax_MessagesController extends Zend_Controller_Action
{
   public function init()
   {
      $ajaxContext = $this->_helper->getHelper('AjaxContext');
      $ajaxContext->addActionContext('regular', array('html', 'json'))
               ->addActionContext('corporate', array('html', 'json'))
               ->initContext();

      Zend_Registry::get("currentUser")->receiveNewMessages();
      $this->getResponse()->setHeader('Tweet-Update-Time', time());
   }

   public function regularAction()
   {
      $dm = Zend_Registry::get('doctrine')->getDocumentManager();

      $requestedUserId = $this->getRequest()->getParam('uid');

      if (isset($requestedUserId)) {

         $tweets = $this->filter($dm->getRepository('Documents\Message\Base')-
>findMessages(
            array(
               'creator' => array(new MongoId($requestedUserId)),
               'after' => $this->getRequest()->getParam('after')
            )
         ));
      }
      else {
         $tweets = $this->filter($dm->getRepository('Documents\Message\Received\Regular')-
>getReceivedMessages(
            Zend_Registry::get("currentUser")->getCurrentprofile(),
            array(
               'after' => $this->getRequest()->getParam('after')
            )
         ));
      }

      $this->view->tweets = $tweets;
   }

   public function corporateAction()
   {
      $dm = Zend_Registry::get('doctrine')->getDocumentManager();

      $tweets = $this->filter($dm->getRepository('Documents\Message\Received\Corporate')-
>getReceivedMessages(
         Zend_Registry::get("currentUser")->getCurrentprofile(),
         array(
            'after' => $this->getRequest()->getParam('after')
         )
      ));

      $this->view->tweets = $tweets;
   }

   protected function filter($qb) {

      if ($this->getRequest()->getParam("after") != null){
         return $qb->getQuery()->execute();
      }
      else {
         Zend_Paginator::setDefaultItemCountPerPage(5);
```

```
        $adapter = new Axiomes\Paginator\Adapter\Odm($qb);

        $paginator = new Zend_Paginator($adapter);

        $paginator->setCurrentPageNumber($this->getRequest()->getParam('page', 1));

        return $paginator;
    }
  }
}
```

application/modules/ajax/controllers/ProfileController.php

```
class Ajax_ProfileController extends Zend_Controller_Action
{
   public function init()
   {
      $ajaxContext = $this->_helper->getHelper('AjaxContext');
      $ajaxContext->addActionContext('track', array('html', 'json'))
               ->setAutoJsonSerialization(false)
               ->initContext();
   }

   public function trackAction()
   {
      $dm = Zend_Registry::get('doctrine')->getDocumentManager();

      $user = Zend_Registry::get("currentUser");

      $profile = $user->getCurrentprofile();

      $trackedprofile = $dm->getRepository('Documents\Profile\Base')->find($this-
>getRequest()->getParam('id'));

      $action = null;

      if(!isset($trackedprofile))
      {
         throw new Exception("ProfileNotFound");
      }
      if(!$profile->getFollowing()->contains($trackedprofile))
      {
         $action = 'track';

         // Track
         $profile->getFollowing()->add($trackedprofile);
         $trackedprofile->getFollowers()->add($profile);
      }
      else
      {
         $action = 'untrack';

         // Untrack
         $profile->getFollowing()->removeElement($trackedprofile);
         $trackedprofile->getFollowers()->removeElement($profile);
      }

      $dm->flush();

      $this->view->profile = $trackedprofile;

      //notify
      try {

         $predis_cli = Zend_Registry::get('redis_cli');

         $data  = array($action.'for'=>$trackedprofile->getId() ,'by'=> $profile->getId());
```

```
            $predis_cli->publish("user:" . $trackedprofile->getUser()->getId() . ":". $trackedprofile-
>getId(), \json_encode($data));
        }
        catch(Exception $x){
            //nothing
        }
    }
}
```

application/modules/ajax/controllers/ProfilesController.php

```
class Ajax_ProfilesController extends Zend_Controller_Action
{
    private $profile;

    public function init()
    {
        $ajaxContext = $this->_helper->getHelper('AjaxContext');
        $ajaxContext->addActionContext('regular', array('html', 'json'))
                ->addActionContext('corporate', array('html', 'json'))
                ->addActionContext('following', array('html', 'json'))
                ->addActionContext('followers', array('html', 'json'))
                ->addActionContext('search', array('html', 'json'))
                ->setAutoJsonSerialization(false)
                ->initContext();
    }

    public function preDispatch()
    {
        $requestedProfileId = $this->getRequest()->getParam('uid');

        if (isset($requestedProfileId))
        {
            $dm = Zend_Registry::get('doctrine')->getDocumentManager();

            $profile = $dm->getRepository('Documents\Profile\Base')
                ->findOneBy(array(
                            'id'=> $requestedProfileId
                        ));

            $this->profile = $profile;
        }
        else
        {
            $user = Zend_Registry::get("currentUser");

            $this->profile = $user->getCurrentprofile();
        }
    }

    public function indexAction()
    {
        if(strlen($this->getRequest()->getParam('term')) < 1) {
            return;
        }

        $dm = Zend_Registry::get('doctrine')->getDocumentManager();

        $profiles = $dm->getRepository('Documents\Profile\Base')->findProfiles(
              array(
                'term' => $this->getRequest()->getParam('term')
              )
        );

        $data = array('count'=>$profiles->count());
```

```
        foreach($profiles as $profile){
            $data['profiles'][] = $profile->jsonSerialize();
        }

        $this->getHelper('json')->sendJson($data);
    }

    //ajax/profiles/regular
    public function regularAction()
    {
        $dm = Zend_Registry::get('doctrine')->getDocumentManager();

        $user = Zend_Registry::get("currentUser");

        $profiles = $dm->getRepository('Documents\Profile\Regular')->findProfiles(
                array(
                    'user' => $user->getId()
                )
        );

        $this->view->profiles = $profiles;
    }

    //ajax/profiles/corporate
    public function corporateAction()
    {
        $dm = Zend_Registry::get('doctrine')->getDocumentManager();

        $user = Zend_Registry::get("currentUser");

        $profiles = $dm->getRepository('Documents\Profile\Corporate')->findProfiles(
                array(
                    'user' => $user->getId()
                )
        );

        $this->view->profiles = $profiles;
    }

    public function followersAction()
    {
        $dm = Zend_Registry::get('doctrine')->getDocumentManager();


        $profiles = $dm->getRepository('Documents\Profile\Base')->findProfiles(
                array(
                    'id' => $this->profile->getFollowers()->getMongoData()
                )
        );

        $this->view->profiles = $profiles;
    }

    public function followingAction()
    {
        $dm = Zend_Registry::get('doctrine')->getDocumentManager();


        $profiles = $dm->getRepository('Documents\Profile\Base')->findProfiles(
                array(
                    'id' => $this->profile->getFollowing()->getMongoData()
                )
        );

        $this->view->profiles = $profiles;
    }

    public function searchAction(){
```

```
        $dm = Zend_Registry::get('doctrine')->getDocumentManager();

        if($this->getRequest()->getParam('term') != null &&
           strlen($this->getRequest()->getParam('term'))>0)
        {
           $profiles = $dm->getRepository('Documents\Profile\Base')->findProfiles(array(
              'term'=> $this->getRequest()->getParam('term')
           ));

           $this->view->profiles = $profiles;

        } else {

           $this->view->profiles = array();
        }
    }
}
```

application/modules/srv/controllers/MessageController.php

```
class Srv_MessageController extends Zend_Controller_Action
{
   public function init()
   {
      $this->_helper->layout->disableLayout();
      $this->_helper->viewRenderer->setNoRender(TRUE);
   }

   public function postAction()
   {
      $currentUser = Zend_Registry::get("currentUser");
      $currentProfile = $currentUser->getCurrentProfile();

      $classname = Documents\Message\Base::getMessageType(
                  $currentProfile
               );

      $message = new $classname();
      $message->setContent($this->getRequest()->getParam("txtMessage"));

      $dm = Zend_Registry::get('doctrine')->getDocumentManager();
      $dm->persist($message);
      $dm->flush();

      //TODO receive new messages - PIGGYBACKING
      $currentUser->receiveNewMessages();

      //TODO publish message TO REDIS channel
      try{

         $predis_cli = Zend_Registry::get('redis_cli');

         foreach($currentProfile->getFollowers() as $follower) {
            $userId = $follower->getUser()->getId();

            if ($message instanceof \Documents\Message\Regular){
               $type = "regular";
            }
            elseif ($message instanceof \Documents\Message\Corporate) {
               $type = "corporate";
            }

            $data  = array('messagefor'=>$follower->getId() ,'messageId'=> $message->getId(),
'messageType'=> $type);
```

```
            $predis_cli->publish("user:" . $userId . ":". $follower->getId(),
\json_encode($data));
         }
      }
      catch(Exception $x){
         //nothing
      }
   }
}
```

application/modules/srv/controllers/UserController.php

```
class Srv_UserController extends Zend_Controller_Action
{
   public function init()
   {
      $this->_helper->layout->disableLayout();
      $this->_helper->viewRenderer->setNoRender(TRUE);
   }

   public function nicknameAvailableAction()
   {
      $stringLengthValidator = new Zend_Validate_StringLength(
         array(
            'min'=> 3,
            'max'=>50,
            'messages' => array(
               Zend_Validate_StringLength::INVALID =>'Your Last Name must contain between
%min% and %max% characters',
               Zend_Validate_StringLength::TOO_LONG =>'Your Last Name cannot contain
more than %max% characters',
               Zend_Validate_StringLength::TOO_SHORT =>'Your Last Name must contain
more than %min% characters'
            )
         )
      );

      $alphaValidator = new Zend_Validate_Alpha(
          array(
            'allowWhiteSpace' => true,
            'messages' => array(
               Zend_Validate_Alpha::INVALID => 'fsdfsd',
               Zend_Validate_Alpha::NOT_ALPHA => 'sss'
            )
         )
      );

      $uniqueNicknameValidator = new My_Validate_UniqueNickname($this->getRequest()-
>getParam('exclude'));

      $nicknameValidatorChain = new Zend_Validate();
      $nicknameValidatorChain->addValidator($stringLengthValidator, true)
                     ->addValidator($alphaValidator, true)
                     ->addValidator($uniqueNicknameValidator) ;

      $validatorResult = $nicknameValidatorChain->isValid($this->getRequest()-
>getParam('nickname'));

      $response = array(
         'valid' => $validatorResult,
         'messages' =>
            call_user_func(function()
               use ($validatorResult, $nicknameValidatorChain, &$response)
            {
               if ($validatorResult)
                  return $response['messages'] = array('validNickname'=> 'Username is
available.');
```

```
                else
                    return $nicknameValidatorChain->getMessages();
            })
        );

        $this->getHelper('json')->sendJson($response);
    }

    public function emailAvailableAction()
    {
        $emptyValidator = new Zend_Validate_NotEmpty(
            array(
                Zend_Validate_NotEmpty::IS_EMPTY=> 'Is empty'
            )
        );

        $formatEmailValidator = new Zend_Validate_EmailAddress(
            array(
                'messages' => array(
                    Zend_Validate_EmailAddress::INVALID_FORMAT => 'Doesn\'t look like a valid
email'
                )
            )
        );

        $uniqueEmailValidator = new My_Validate_UniqueEmail();

        $emailValidatorChain = new Zend_Validate();
        $emailValidatorChain->addValidator($emptyValidator, true)
                    ->addValidator($formatEmailValidator, true)
                    ->addValidator($uniqueEmailValidator) ;

        $validatorResult = $emailValidatorChain->isValid($this->getRequest()-
>getParam('email'));

        $response = array(
            'valid' => $validatorResult,
            'messages' =>
                call_user_func(function()
                    use ($validatorResult, $emailValidatorChain, &$response)
                {
                    if ($validatorResult)
                        return $response['messages'] = array('validEmail'=> 'E-mail looks great.');
                    else
                        return $emailValidatorChain->getMessages();
                })
        );

        $this->getHelper('json')->sendJson($response);
    }
}
```

Παράρτημα Γ: Views

application/views/scripts/index/index.phtml

```
$this->headScript()->appendFile($this->baseUrl("js/components/main.js"));

$user = Zend_Registry::get("currentUser");
$profile = $user->getCurrentprofile();
$pageWidgets = array();

// Column 1
$col1order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('newmessage', 1, $col1order++,
array(
   'headerTitle'=> '<a href="' . $this->url(array('module'=>'default', 'controller'=>'profile',
'action'=>$profile->getId())) . '">' . (string) $profile . '</a>',
   'headerPhoto'=> $profile->getProfilepiclink(),
   'profile'=> $profile
));

$pageWidgets[] = Documents\UserWidget::loadUserWidget('profiles/regular', 1, $col1order++,
array('headerTools'=> '<a href="'.  $this->baseUrl('/profiles/add/type/regular') .'" class="btn
btn-mini btn-info"><i class="icon-plus-sign"></i> Create</a>') );
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profiles/corporate', 1,
$col1order++, array('headerTools'=> '<a href="'.  $this->baseUrl('/profiles/add/type/corporate')
.'" class="btn btn-mini btn-info"><i class="icon-plus-sign"></i> Create</a>') );

// Column 2
$tweetsOptions = array(
   'headerTools'=> '<a class="btn btn-mini btn-info btn-refresh"><i class="icon-refresh"></i>
Refresh</a>',
   'profile'=> $profile
);

$corpTweetsOptions = array(
   'headerTools'=> '<a class="btn btn-mini btn-info btn-refresh"><i class="icon-refresh"></i>
Refresh</a>',
   'profile'=> $profile
);

$col2order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('tweets', 2, $col2order++,
$tweetsOptions);
$pageWidgets[] = Documents\UserWidget::loadUserWidget('corptweets', 2, $col2order++,
$corpTweetsOptions);

$this->Widgets()->setPageWidgets($pageWidgets);
?>

<div class="container">

<div class="row" id="columns">

   <!-- 1st column -->
   <div id="column1" class="span4 column">
      <?php echo $this->Widgets(1); ?>
   </div>

   <!-- 2nd column -->
   <div id="column2" class="span8 column">
      <?php echo $this->Widgets(2); ?>
   </div>

</div>
```

```
</div>
```

application/views/scripts/index/following.phtml

```
$this->headScript()->appendFile($this->baseUrl("js/components/main.js"));

$pageWidgets = array();

// Column 1
$col1order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/nav', 1, $col1order++,
array('noWrapped'=>true, 'profile'=> $this->currentProfile));

// Column 2
$col2order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('following', 2, $col2order++, array(
'profile'=> $this->currentProfile));

$this->Widgets()->setPageWidgets($pageWidgets);
?>

<div class="container">

   <div class="row" id="columns">

      <!-- 1st column -->
      <div id="column1" class="span4 column">
         <?php echo $this->Widgets(1); ?>
      </div>

      <!-- 2nd column -->
      <div id="column2" class="span8 column">
         <?php echo $this->Widgets(2); ?>
      </div>


   </div>

</div>
```

application/views/scripts/index/followers.phtml

```
$this->headScript()->appendFile($this->baseUrl("js/components/main.js"));

$pageWidgets = array();

// Column 1
$col1order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/nav', 1, $col1order++,
array('noWrapped'=>true, 'profile'=> $this->currentProfile ));

// Column 2
$col2order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('followers', 2, $col2order++,
array('profile'=> $this->currentProfile));

$this->Widgets()->setPageWidgets($pageWidgets);
?>

<div class="container">

   <div class="row" id="columns">

      <!-- 1st column -->
```

```
      <div id="column1" class="span4 column">
         <?php echo $this->Widgets(1); ?>
      </div>

      <!-- 2nd column -->
      <div id="column2" class="span8 column">
         <?php echo $this->Widgets(2); ?>
      </div>

   </div>

</div>
```

application/views/scripts/profile/index.phtml

```
$this->headScript()->appendFile($this->baseUrl("js/components/main.js"));

$headerTitle = '@' . (string) $this->profile;

$pageWidgets = array();

$col1order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/nav', 1, $col1order++,
array('noWrapped'=> true,'profile'=> $this->profile));

$col2order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/view', 2, $col2order++,
array('header'=> false, 'profile'=> $this->profile, 'headerTitle'=>$headerTitle));
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/tweets', 2, $col2order++,
array('profile'=> $this->profile));

$this->Widgets()->setPageWidgets($pageWidgets);
?>

<div class="container">
<div class="row" id="columns">

   <!-- 1st column -->
   <div id="column1" class="span4 column">
      <?php echo $this->Widgets(1); ?>
   </div>

   <!-- 2nd column -->
   <div id="column2" class="span8 column">
      <?php echo $this->Widgets(2); ?>
   </div>

</div>
</div>
```

application/views/scripts/profile/following.phtml

```
$this->headScript()->appendFile($this->baseUrl("js/components/main.js"));

$pageWidgets = array();

// Column 1
$col1order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/nav', 1, $col1order++,
array('noWrapped'=> true, 'profile'=> $this->profile, 'active'=>'following'));

// Column 2
$col2order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/view', 2, $col2order++,
array('headerTitle'=> (string) $this->profile, 'profile'=> $this->profile));
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/following', 2, $col2order++,
array('profile'=> $this->profile));

$this->Widgets()->setPageWidgets($pageWidgets);
?>

<div class="container">
   <div class="row" id="columns">

      <!-- 1st column -->
      <div id="column1" class="span4 column">
         <?php echo $this->Widgets(1); ?>
      </div>

      <!-- 2nd column -->
      <div id="column2" class="span8 column">
         <?php echo $this->Widgets(2); ?>
      </div>


   </div>
</div>
```

application/views/scripts/profile/followers.phtml

```
$this->headScript()->appendFile($this->baseUrl("js/components/main.js"));

$pageWidgets = array();

// Column 1
$col1order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/nav', 1, $col1order++,
array('noWrapped'=> true, 'profile'=> $this->profile, 'active'=>'followers'));

// Column 2
$col2order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/view', 2, $col2order++,
array('headerTitle'=> (string) $this->profile, 'profile'=> $this->profile));
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/followers', 2, $col2order++,
array('profile'=> $this->profile));

$this->Widgets()->setPageWidgets($pageWidgets);
?>

<div class="container">

   <div class="row" id="columns">

      <!-- 1st column -->
      <div id="column1" class="span4 column">
         <?php echo $this->Widgets(1); ?>
```

```
      </div>

      <!-- 2nd column -->
      <div id="column2" class="span8 column">
         <?php echo $this->Widgets(2); ?>
      </div>

   </div>

</div>
```

application/views/scripts/profile/tweets.phtml

```
$this->headScript()->appendFile($this->baseUrl("js/components/main.js"));

$headerTitle = '@' . (string) $this->profile;

$pageWidgets = array();

$col1order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/nav', 1, $col1order++,
array('noWrapped'=> true,'profile'=> $this->profile, 'active'=>'tweets'));

$col2order = 0;
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/view', 2, $col2order++,
array('header'=> false, 'profile'=> $this->profile, 'headerTitle'=>$headerTitle));
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profile/tweets', 2, $col2order++,
array('profile'=> $this->profile));

$this->Widgets()->setPageWidgets($pageWidgets);
?>

<div class="container">
<div class="row" id="columns">

   <!-- 1st column -->
   <div id="column1" class="span4 column">
      <?php echo $this->Widgets(1); ?>
   </div>

   <!-- 2nd column -->
   <div id="column2" class="span8 column">
      <?php echo $this->Widgets(2); ?>
   </div>

</div>
</div>
```

application/views/scripts/profiles/index.phtml

```
$pageWidgets = array();

$pageWidgets[] = Documents\UserWidget::loadUserWidget('profiles/regular', 1, 0,
array('headerTools'=> '<a href="'.  $this->baseUrl('/profiles/add/type/regular') .'" class="btn
btn-mini btn-info"><i class="icon-plus-sign"></i> Create</a>') );

$pageWidgets[] = Documents\UserWidget::loadUserWidget('profiles/corporate', 2, 0,
array('headerTools'=> '<a href="'.  $this->baseUrl('/profiles/add/type/corporate') .'" class="btn
btn-mini btn-info"><i class="icon-plus-sign"></i> Create</a>') );

$this->widgets()->setPageWidgets($pageWidgets);
?>

<div class="row-fluid" id="columns">
   <!-- 1st column -->
```

```
   <div id="column1" class="span6">
   <?php echo $this->widgets(1); ?>
   </div>

   <!-- 2st column -->
   <div id="column2" class="span6">
   <?php echo $this->widgets(2); ?>
   </div>
</div>
```

application/views/scripts/profiles/add.phtml

```
echo $this->frmProfile;
```

application/views/scripts/profiles/edit.phtml

```
echo $this->frmProfile;
```

application/views/scripts/profiles/search.phtml

```
$this->headScript()->appendFile($this->baseUrl("js/components/main.js"));

$pageWidgets = array();
$pageWidgets[] = Documents\UserWidget::loadUserWidget('profiles/search', 1, 0,
array('term'=>$this->term) );


$this->widgets()->setPageWidgets($pageWidgets);
?>

<div class="container">
   <div class="row" id="columns">

      <!-- 1nd column -->
      <div id="column1" class="span12 column">
         <?php echo $this->Widgets(1); ?>
      </div>

   </div>
</div>
```

application/views/scripts/signin/index.phtml

```
echo $this->frmSignin;
```

application/views/scripts/signup/index.phtml

```
echo $this->frmSignup;
```

application/views/scripts/forms/_frmProfileCorporate.phtml

```
<?php
$this->headScript()->appendFile($this->baseUrl("js/components/validateInput.js"));
?>
<script type="text/javascript">

$(document).ready(function() {

   $('#txtCompanyname').validateInput({
      url: baseUrl + "/srv/user/nickname-available",
      paramName: 'nickname',
      exclude: "<?php echo isset($this->profile)? (string) $this->profile:""; ?>"
   });

});

</script>

<div class="white roundedBox formContainer">

   <h2>Create a Corporate Profile</h2>

   <form id="frmProfileRegular"
       name="frmProfileRegular"
       method="POST"
       action="<?php echo $this->url; ?>"
       >

      <div class="text-label">Company Name</div>
      <input type="text" name="txtCompanyname" id="txtCompanyname" class="input-xlarge"
value="<?php echo isset($this->profile)?$this->profile->getCompanyname():""; ?>">
      <span class="help-inline sidetip">
         <p class="text-info tip">Enter a Company Name</p>
         <p class="text-success ok"></p>
         <p class="text-error error"></p>
         <p class="muted checking">Validating...</p>
      </span>

      <div class="text-label">Bio</div>
      <textarea name="txtBio" id="txtBio" rows="5" class="input-xlarge"><?php echo
isset($this->profile)?$this->profile->getBio():""; ?></textarea>
      <span class="help-inline sidetip">
         <p class="text-info tip">Enter a Bio</p>
         <p class="text-success ok"></p>
         <p class="text-error error"></p>
      </span>

      <div class="clearfix"></div>

      <button type="submit" class="btn btn-primary input-xlarge"><?php echo $this->label;
?></button>

      <input type="hidden" name="id" id="id" value="<?php echo isset($this->profile)?$this-
>profile->getId():""; ?>">
   </form>

</div>
```

application/views/scripts/forms/_frmProfileRegular.phtml

```
<?php
$this->headScript()->appendFile($this->baseUrl("js/components/jquery.task.queue.js"));
$this->headScript()->appendFile($this->baseUrl("js/components/formValidation.js"));
$this->headScript()->appendFile($this->baseUrl("js/lib/jquery.timer.js"));

$this->headScript()->appendFile($this->baseUrl("js/lib/jquery.redirect.js"));

$flashMessenger = Zend_Controller_Action_HelperBroker::getStaticHelper('FlashMessenger');

if ($flashMessenger->hasCurrentMessages('success')):
   ?><div class="alert alert-success"><button type="button" class="close" data-
dismiss="alert">?</button><?php
   foreach ($flashMessenger->getCurrentMessages('success') as $msg):
      echo "<strong>" . $msg . "</strong>";
   endforeach;
   ?></div><?php
endif;

if ($flashMessenger->hasCurrentMessages('error')):
   ?><div class="alert alert-success"><button type="button" class="close" data-
dismiss="alert">?</button><?php
      foreach ($flashMessenger->getCurrentMessages('error') as $msg):
         echo "<strong>" . $msg . "</strong>";
      endforeach;
      ?></div><?php
endif;
?>
<script type="text/javascript">

   $(document).ready(function() {

      $('#frmProfileRegular').formValidation({
         onkeyup: true,
         delay: 500,
         fields: [
            {
               name: 'txtNickname',
               tip: 'Enter your nickname.',
               rules: [
                  {
                     name: 'required',
                     errorMessage: 'A nickname is required!',
                     onEvts: ['submit', 'keyup']
                  },
                  {
                     name: 'remote',
                     params: {
                        url: baseUrl + "/srv/user/nickname-available",
                        type: "get",
                        data: {
                           nickname: function() {
                              return $('#frmProfileRegular :input[id="txtNickname"]').val();
                           },
                           exclude: function() {
                              return "<?php echo isset($this->profile) ? $this->profile->getId() :
"0"; ?>";
                           }
                        }
                     }
                  }
               ]
            }
         ],
         tip: {
            onEvts: [
```

```
            {
               name: 'focus',
               conditions: [
                  function($element) {
                     return !$element.data('has-content');
                  }
               ]
            },
            {
               name: 'keydown'
            }
         ],
         handler: function($element, msg) {

            var $placeholder = $element.next(".help-inline:first").find("p:first");

            $placeholder.attr("class", "text-info tip");
            $placeholder.html(msg).show();

         }
      },
      callbackHandlers: {
         fieldRemoteChecking: function($element) {
            var $placeholder = $element.next(".help-inline:first").find("p:first");
            $placeholder.attr("class", "muted checking");
            $placeholder.html("Validating...").show();
         },
         fieldOk: function($element, msg) {
            var $placeholder = $element.next(".help-inline:first").find("p:first");
            $placeholder.attr("class", "text-success ok");
            $placeholder.html(msg).show();
         },
         fieldError: function($element, msg) {
            var $placeholder = $element.next(".help-inline:first").find("p:first");
            $placeholder.attr("class", "text-error error");
            $placeholder.html(msg).show();
         }
      },
      formSubmit: function($form) {

         $.fn.serializeObject = function()
         {
            var o = {};
            var a = this.serializeArray();
            $.each(a, function() {
               if (o[this.name]) {
                  if (!o[this.name].push) {
                     o[this.name] = [o[this.name]];
                  }
                  o[this.name].push(this.value || '');
               } else {
                  o[this.name] = this.value || '';
               }
            });
            return o;
         };

         $().redirect($form.attr("action"), $form.serializeObject());

      },
      preventFormSubmit: function($form) {

         $.fn.presence = function() {
            return this.length > 0 ? this : null;
         }

         var $f = $("input + .sidetip:has(p[class^='text-info'])").first().prev().presence()
               ||
```

```
                $("input + .sidetip:has(p[class=''])").first().prev().presence()
                ||
                $("input + .sidetip:has(p[class^='text-error'])").first().prev().presence();

            $f.focus();

            $(".buttonContainer").effect("shake", {times: 1, distance: 3}, 350);
        }

    });

    $('input#txtNickname:first').focus();

    //$('input#txtNickname:first').data('ValidationUnit').run();

  });

</script>

<div class="white roundedBox formContainer">

  <?php if (isset($this->profile)) : ?>
    <h2>Update your Profile</h2>
  <?php else : ?>
    <h2>Create a Regular Profile</h2>
  <?php endif; ?>


  <form id="frmProfileRegular"
      name="frmProfileRegular"
      class="white"
      method="POST"
      action="<?php echo $this->url; ?>"
      >

    <div class="text-label">Nickname</div>
    <input type="text" name="txtNickname" id="txtNickname" class="input-xlarge"
value="<?php echo isset($this->profile) ? $this->profile->getNickname() : ""; ?>">
    <span class="help-inline sidetip">
      <p class=""></p>
    </span>

    <div class="text-label">Bio</div>
    <textarea name="txtBio" id="txtBio" rows="5" class="input-xlarge"><?php echo
isset($this->profile) ? $this->profile->getBio() : ""; ?></textarea>


    <div class="clearfix"></div>

    <div class="buttonContainer"><button type="submit" class="btn btn-primary"><?php echo
$this->label; ?></button></div>

    <input type="hidden" name="id" id="id" value="<?php echo isset($this->profile) ? $this-
>profile->getId() : ""; ?>">
  </form>

</div>
```

application/views/scripts/forms/_frmSignin.phtml

```
<?php
$flashMessenger = Zend_Controller_Action_HelperBroker::getStaticHelper('FlashMessenger');

if ($flashMessenger->setNamespace('success')->hasMessages()):
   ?><div class="alert alert-success"><button type="button" class="close" data-
dismiss="alert">?</button><?php
   foreach ($flashMessenger->getMessages() as $msg):
      echo "<strong>" . $msg . "</strong>";
   endforeach;
   ?></div><?php
endif;

if ($flashMessenger->setNamespace('error')->hasMessages()):
   ?><div class="alert alert-error"><button type="button" class="close" data-
dismiss="alert">?</button><?php
      foreach ($flashMessenger->getMessages() as $msg):
         echo "<strong>" . $msg . "</strong>";
      endforeach;
      ?></div><?php
endif;
?>
<div class="white roundedBox formContainer">

   <h2>Login to G-nnect</h2>

   <form id="frmSignin"
        class="white"
        name="frmSignin"
        method="POST"
        action="<?php echo $this->url(array('module' => 'default', 'controller' => 'signin',
'action' => 'process'), 'default', false); ?>"
        >

      <input type="text" class="input-xlarge" placeholder="Email" id="txtEmail"
name="txtEmail">

      <div class="clearfix"></div>

      <input type="password" class="input-xlarge" placeholder="Password" id="txtPassword"
name="txtPassword">

      <div class="clearfix"></div>

      <button type="submit" class="btn btn-primary">Sign in</button>

   </form>

   <div class="clearfix">
      <br><br><br>
      <p class="signup-helper">
         <small><strong>New to G-nnect ?
         <a href="<?php echo $this->url(array('module' => 'default', 'controller' => 'signup',
'action' => 'index'), 'default', false); ?>">Sign up now »</a>
         </strong></small>
      </p>
   </div>

</div>
```

application/views/scripts/forms/_frmSignup.phtml

```
<?php
$this->headScript()->appendFile($this->baseUrl("js/components/jquery.task.queue.js"));
$this->headScript()->appendFile($this->baseUrl("js/components/formValidation.js"));
$this->headScript()->appendFile($this->baseUrl("js/lib/jquery.timer.js"));

$this->headScript()->appendFile($this->baseUrl("js/lib/jquery.redirect.js"));
?>
<script type="text/javascript">

   $(document).ready(function() {

      $('#frmSignup').formValidation({
         onkeyup: true,
         delay: 500,
         fields: [
            {
               name: 'txtFullname',
               tip: 'Enter your first and last name.',
               rules: [
                  {
                     name: 'required',
                     errorMessage: 'A name is required!',
                     successMessage: 'Name looks great.'
                  }
               ]
            },
            {
               name: 'txtEmail',
               tip: 'What is your email address?',
               rules: [
                  {
                     name: 'required',
                     errorMessage: 'An email is required!'
                  },
                  {
                     name: 'email',
                     errorMessage: 'Email is invalid!'
                  },
                  {
                     name: 'remote',
                     params: {
                        url: baseUrl + "/srv/user/email-available",
                        type: "get",
                        data: {
                           email: function() {
                              return $('#frmSignup :input[id="txtEmail"]').val();
                           }
                        }
                     }
                  }
               ]
            },
            {
               name: 'txtNickname',
               tip: 'Enter your nickname.',
               rules: [
                  {
                     name: 'required',
                     errorMessage: 'A nickname is required!',
                     onEvts: ['submit', 'keyup']
                  },
                  {
                     name: 'remote',
                     params: {
                        url: baseUrl + "/srv/user/nickname-available",
```

```
                        type: "get",
                        data: {
                           nickname: function() {
                              return $('#frmSignup :input[id="txtNickname"]').val();
                           }
                        }
                     }
                  }
               ]
            }
         ],
         tip: {
            onEvts: [
               {
                  name: 'focus',
                  conditions: [
                     function($element) {
                        return !$element.data('has-content');
                     }
                  ]
               },
               {
                  name: 'keydown'
               }
            ],
            handler: function($element, msg) {

               var $placeholder = $element.next(".help-inline:first").find("p:first");

               $placeholder.attr("class", "text-info tip");
               $placeholder.html(msg).show();

            }
         },
         callbackHandlers: {
            fieldRemoteChecking: function($element) {
               var $placeholder = $element.next(".help-inline:first").find("p:first");
               $placeholder.attr("class", "muted checking");
               $placeholder.html("Validating...").show();
            },
            fieldOk: function($element, msg) {
               var $placeholder = $element.next(".help-inline:first").find("p:first");
               $placeholder.attr("class", "text-success ok");
               $placeholder.html(msg).show();
            },
            fieldError: function($element, msg) {
               var $placeholder = $element.next(".help-inline:first").find("p:first");
               $placeholder.attr("class", "text-error error");
               $placeholder.html(msg).show();
            }
         },
         formSubmit: function($form) {

            var $request = $.ajax({
               url: $form.attr("action"),
               type: "POST",
               data: $form.serializeArray(),
               beforeSend: function(xhr) {
                  $('.formContainer > *').slideUp('fast', function(){
                     $('.formContainer').html("<i class='icon-spinner icon-spin'></i> Creating
your account...").show();
                  });
               }
            });

            $request.done(function(data, textStatus, xhr){

               if (data.success)
```

```
                $().redirect(baseUrl + '/signin', {}, 'GET');
                //window.location.replace(baseUrl + "/signin" );

            });
        },
        preventFormSubmit: function($form) {

            $.fn.presence = function() {
               return this.length > 0 ? this : null;
            }

            var $f = $("input + .sidetip:has(p[class^='text-info'])").first().prev().presence()
                ||
                $("input + .sidetip:has(p[class=''])").first().prev().presence()
                ||
                $("input + .sidetip:has(p[class^='text-error'])").first().prev().presence();

            $f.focus();

            $(".buttonContainer").effect("shake", {times: 1, distance: 3}, 350);

            //console.log('stop submiting');
        }

    });

    $('input#txtFullname').focus();
  });

</script>

<div class="white roundedBox formContainer">

  <h2>Join to G-nnect</h2>

  <form
    id="frmSignup"
    name="frmSignup"
    class="white"
    method="POST"
    action="<?php echo $this->url(array('module' => 'default', 'controller' => 'signup', 'action'
=> 'process'), 'default', false); ?>"
    >

    <div class="text-label">Full Name</div>
    <input type="text" name="txtFullname" id="txtFullname" class="input-xlarge">
    <span class="help-inline sidetip">
      <p class=""></p>
    </span>


    <div class="text-label">Email Address</div>
    <input type="text" name="txtEmail" id="txtEmail" class="input-xlarge">
    <span class="help-inline sidetip">
      <p class=""></p>
    </span>

    <div class="text-label">Password</div>
    <input type="password" name="txtPassword" id="txtPassword" class="input-xlarge">



    <div class="text-label">Nickname</div>
    <input type="text" name="txtNickname" id="txtNickname" class="input-xlarge">
    <span class="help-inline sidetip">
      <p class=""></p>
    </span>
```

```
      <div class="clearfix"></div>

      <div class="buttonContainer"><button type="submit" class="btn btn-primary">Create your
account</button></div>

   </form>

   <div class="clearfix">
      <br><br><br>
      <p class="signup-helper">
         <small><strong>Have an account ?
         <a href="<?php echo $this->url(array('module' => 'default', 'controller' => 'signin',
'action' => 'index'), 'default', false); ?>">Sign in »</a>
         </strong></small>
      </p>
   </div>

</div>
```

application/views/scripts/widgets/_template.phtml

```
<div id="<?php echo $this->widgetId; ?>"
    class="widget roundedBox"
    data-page-name="<?php echo $this->pageName; ?>"
    data-col="<?php echo $this->col; ?>"
    data-order="<?php echo $this->order; ?>">

   <?php if($this->header) : ?>
   <div class="media widget-head">

      <?php if(isset($this->headerPhoto) && $this->headerPhoto != ''): ?>
      <div class="pull-left">
      <a href="#" ><img class="img-rounded " src="<?php echo $this->headerPhoto; ?>"
width="32" height="32" /></a>
      </div>
      <?php endif; ?>

      <?php if(isset($this->headerTools) && $this->headerTools != ''): ?>
      <small class="pull-right head-tools"><?php echo $this->headerTools; ?></small>
      <?php endif; ?>

      <div class="media-body">

         <h2 class="media-heading head-title"><?php echo $this->title; ?></h2>

         <?php if(isset($this->headerDescr) && $this->headerDescr != ''): ?>
         <p></p>
         <?php endif; ?>

      </div>

   </div>
   <?php endif; ?>

   <div class="widget-content">
   <?php echo $this->widgetContent; ?>
   </div>

</div>
```

application/views/scripts/widgets/newmessage.phtml

```
<?php
$this->headScript()->appendFile($this->baseUrl("js/lib/jquery.form.js"));
$this->headScript()->appendFile($this->baseUrl("js/lib/maxlen.js"));
?>

<script type="text/javascript">

   $(document).ready(function() {

      $('.new-message').ajaxForm({
         type: 'POST',
         beforeSubmit: function() {

            var txtMessage = $('.new-message textarea[name="txtMessage"]');

            if (txtMessage.val() != '')
            {
               if (txtMessage.val().length <= 300) {
                  txtMessage.attr('disabled', 'disabled');
                  $('.new-message button[name="btnPostMessage"]').attr('disabled', 'disabled');
               }
               else {
                  return false;
               }
            }
            else {
               return false;
            }
         },
         success: function() {

            $(window).tweets('updateTweets', {});

            $('.new-message textarea[name="txtMessage"]').val('');
            $('.new-message textarea[name="txtMessage"]').removeAttr('disabled');
            $('.new-message button[name="btnPostMessage"]').removeAttr('disabled');

            $('ul#profile-stats-' + currentProfile).data("totalMessages", {"inc": 1});
         }
      });

      $('ul#profile-stats-' + currentProfile).on("setData", function(e, key, val){
         if (key=="totalMessages") {
            $element = $(this).find('a.totalMessages strong');
            $element.html(val);
         }
         else if (key=="totalFollowers") {
            $element = $(this).find('a.totalFollowers strong');
            $element.html(val);
         }
         else if (key=="totalFollowing") {
            $element = $(this).find('a.totalFollowing strong');
            $element.html(val);
         }
      });

      $('#txtMessage').maxlen({
         'counter': '.message-counter'
      });

      $("#txtMessage").on("setData", function(e, key, val) {
         if (key=="totalCharsCount") {
            if (val > 0)
               $('#btnPostMessage').removeClass('disabled');
```

```
            else
               $('#btnPostMessage').addClass('disabled');

         }
      });

   });

</script>

<ul id="profile-stats-<?php echo $this->profile->getId(); ?>"
   class="profile-stats"
   data-total-followers="<?php echo $this->profile->getFollowers()->count(); ?>"
   data-total-following="<?php echo $this->profile->getFollowing()->count(); ?>"
   data-total-messages="<?php echo $this->profile->getSentMessages()->count(); ?>"
   >

   <li><a href="<?php echo $this->baseUrl('profile/' . $this->profile->getId() . '/followers'); ?>"
class="totalFollowers"><strong><?php echo $this->profile->getFollowers()->count();
?></strong>Followers</a></li>

   <li><a href="<?php echo $this->baseUrl('profile/' . $this->profile->getId() . '/following'); ?>"
class="totalFollowing"><strong><?php echo $this->profile->getFollowing()->count();
?></strong>Following</a></li>

   <li><a href="<?php echo $this->baseUrl('profile/' . $this->profile->getId() . '/tweets'); ?>"
class="totalMessages"><strong><?php echo $this->profile->getSentMessages()->count();
?></strong>Messages</a></li>

</ul>

<div class="container-fluid">
   <div class="row-fluid">
      <div class="span12">

         <form class="new-message form-search"
             method="POST"
             action="<?php echo $this->url(array('module' => 'srv', 'controller' => 'message',
'action' => 'post'), 'default', false); ?>"
          >

            <br/>

            <textarea rows="3" class="span12" id="txtMessage" name="txtMessage" data-total-
chars-count="0" placeholder="Compose new message..."></textarea>

            <div class="message-button pull-right">
               <span class="muted message-counter"></span>
               <button type="submit" class="btn btn-info disabled" id="btnPostMessage"
name="btnPostMessage">Post</button>
            </div>

            <div class="clearfix"></div>

         </form>

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

application/views/scripts/widgets/tweets.phtml

```
<?php
//This view display the regular received tweets for current logged-in user profile

$this->headScript()->appendFile($this->baseUrl("js/components/tweets.js"));
```

```
?>

<script type="text/javascript">
   $(document).ready(function() {

      var tweetsSelector = 'div#regular-tweets-' + currentProfile;

      $(tweetsSelector).tweets({
         url: baseUrl + "/ajax/messages/regular?format=html",
         loaded: function() {
            $(tweetsSelector).tweets('initRefreshBtn');
         }
      });

      $(tweetsSelector).on('setData', function(e, key, val) {

         if (key == "streamItemsCount")
         {
            var itemsCount = $(this).data("streamItemsCount");

            if (itemsCount == 0) {
               $(tweetsSelector + ' .notify-container:first').remove();
               return;
            }

            var $notification = $(this).find('.notify-container:first .tweet-notify:first');

            if ($notification.length)
            {
               $notification.html(itemsCount + ' new tweets');
               $notification.parent().effect("highlight", {}, 3000);
            }
            else
            {
               var $notifyElement = $('<div class="media media-container notify-container"><a
href="#" class="tweet-notify">' + itemsCount + ' new tweets</a></div>');
               $(this).prepend($notifyElement);

               $notifyElement.show('fast').effect("highlight", {}, 3000);
            }

            $(this).find('.notify-container:first .tweet-notify:first').unbind('click').bind('click',
function() {
               $(tweetsSelector + ' .notify-container:first').remove();
               $(tweetsSelector).tweets('loadTweets', {}, false);
            });
         }
      });
   });
</script>

<div id="regular-tweets-<?php echo $this->profile->getId(); ?>"
    class="media-collection tweets-collection"
    data-stream-items-count="0"
    data-stream-items=""
    data-tweets-settings=""
    data-page="">

</div>
```

application/views/scripts/widgets/corptweets.phtml

```
<?php
//This view display the corporate received tweets for current logged-in user profile

$this->headScript()->appendFile($this->baseUrl("js/components/tweets.js"));
?>

<script type="text/javascript">

   $(document).ready(function() {

      var tweetsSelector = 'div#corporate-tweets-' + currentProfile;

      $(tweetsSelector).tweets({
         url: baseUrl + "/ajax/messages/corporate?format=html",
         loaded: function() {
            $(tweetsSelector).tweets('initRefreshBtn');
         }
      });

      $(tweetsSelector).on('setData', function(e) {
         var itemsCount = $(this).data("streamItemsCount");

         if (itemsCount == 0) {
            $(tweetsSelector + ' .notify-container:first').remove();
            return;
         }

         var notification = $(this).find('.notify-container:first .tweet-notify');

         if (notification.length) {
            notification.html(itemsCount + ' tweets');
            notification.parent().effect("highlight", {}, 3000);
         }
         else {
            var notifyElement = $('<div class="media media-container notify-container"><a
href="#" class="tweet-notify">' + itemsCount + ' tweets</a></div>');
            $(this).prepend(notifyElement);
            notifyElement.show('fast').effect("highlight", {}, 3000);
         }

         $(this).find('.notify-container:first .tweet-notify:first').unbind('click').bind('click',
function() {

            $(tweetsSelector).tweets('loadTweets', {preLoad: function() {
                  $(tweetsSelector).data()["streamItemsCount"] = 0;
                  $(tweetsSelector + ' .notify-container:first .tweet-
notify:first').parent().remove();
               }}, false);
         });
      });
   });

</script>

<div id="corporate-tweets-<?php echo $this->profile->getId(); ?>"
    class="media-collection tweets-collection"
    data-stream-items-count="0"
    data-stream-items=""
    data-tweets-settings="">
</div>
```

application/views/scripts/widgets/followers.phtml

```
<?php
//This view display the followers for current logged-in user profile

$this->headScript()->appendFile($this->baseUrl("js/components/profiles.js"));
?>

<script type="text/javascript">

   $(document).ready(function() {
      $('#profiles-followers-collection').profiles({
         url: baseUrl + "/ajax/profiles/followers?format=html",
         loaded: function() {
            $(".profile-container").profiles('initTrackBtn');
         }
      });
   });

</script>

<div id="profiles-followers-collection" class="media-collection tracking-content"></div>
```

application/views/scripts/widgets/following.phtml

```
<?php
//This view display the followings for current logged-in user profile

$this->headScript()->appendFile($this->baseUrl("js/components/profiles.js"));
?>

<script type="text/javascript">

   $(document).ready(function() {
      $('#profiles-following-collection').profiles({
         url: baseUrl + "/ajax/profiles/following?format=html",
         loaded: function() {
            $(".profile-container").profiles('initTrackBtn');
         }
      });
   });

</script>

<div id="profiles-following-collection" class="media-collection tracking-content"></div>
```

application/views/scripts/widgets/profile/nav.phtml

```
<?php
$user = Zend_Registry::get("currentUser");

$currentLoggedInUser = false;
if( $user->getProfiles()->contains($this->profile) )
   $currentLoggedInUser = true;

$currentLoggedInProfile = false;
if( $user->getCurrentprofile() == $this->profile )
   $currentLoggedInProfile = true;
?>
<ul class="nav nav-tabs nav-stacked profile-nav">

   <li class="<?php echo ($this->active=="following")?"active":""; ?>"><a href="<?php echo
$this->baseUrl( 'profile/' . $this->profile->getId() . '/following' ); ?>"><i class="icon-chevron-
right"></i>Following</a></li>

   <li class="<?php echo ($this->active=="followers")?"active":""; ?>"><a href="<?php echo
$this->baseUrl( 'profile/' . $this->profile->getId() . '/followers' ); ?>"><i class="icon-chevron-
right"></i>Followers</a></li>

   <li class="<?php echo ($this->active=="tweets")?"active":""; ?>"><a href="<?php echo $this-
>baseUrl( 'profile/' . $this->profile->getId() . '/tweets' ); ?>"><i class="icon-chevron-
right"></i>Tweets</a></li>

</ul>
```

application/views/scripts/widgets/profile/view.phtml

```
<?php
$this->headScript()->appendFile($this->baseUrl("js/components/profiles.js"));

$user = Zend_Registry::get("currentUser");

$currentLoggedInUser = false;
if ($user->getProfiles()->contains($this->profile))
   $currentLoggedInUser = true;

$currentLoggedInProfile = false;
if ($user->getCurrentprofile() == $this->profile)
   $currentLoggedInProfile = true;
?>

<script type="text/javascript" >
   $(document).ready(function() {

      $(".container-footer .profile-content").profiles('initTrackBtn');

      $('ul#profile-stats-' + currentProfile).on("setData", function(e, key, val) {
         if (key == "totalMessages") {
            $element = $(this).find('a.totalMessages strong');
            $element.html(val);
         }
         else if (key == "totalFollowers") {
            $element = $(this).find('a.totalFollowers strong');
            $element.html(val);
         }
         else if (key == "totalFollowing") {
            $element = $(this).find('a.totalFollowing strong');
            $element.html(val);
         }
      });

   });
</script>

<div class="container-fluid profile-view-header">

   <div class="row-fluid">
      <div class="offset4 span4" style="text-align: center;padding:10px;"><img class="img-
rounded profile-avatar" src="<?php echo $this->profile->getProfilepiclink(); ?>" alt="Picture for
@<?php echo (string) $this->profile; ?>" /></div>
   </div>

   <div class="row-fluid">
      <div class="span12">

         <div style="text-align: center; color: #fff; padding: 10px;">
            <div class="profile-bio"><?php echo $this->profile->getBio(); ?></div>
            <div class="profile-website"><a href="<?php echo $this->profile->getWebsite();
?>"><?php echo $this->profile->getWebsite(); ?></a></div>
         </div>

      </div>
   </div>

</div>

<div class="container-fluid container-footer">

   <div class="row-fluid">
      <div class="span12 profile-content" data-profile-id="<?php echo $this->profile->getId();
?>">
```

```
        <ul id="profile-stats-<?php echo $this->profile->getId(); ?>"
           class="profile-stats pull-left"
           data-total-followers="<?php echo $this->profile->getFollowers()->count(); ?>"
           data-total-following="<?php echo $this->profile->getFollowing()->count(); ?>"
           data-total-messages="<?php echo $this->profile->getSentMessages()->count(); ?>"
           >

           <li><a href="<?php echo $this->baseUrl((!$currentLoggedInProfile) ? 'profile/' .
$this->profile->getId() . '/followers' : 'followers' ); ?>" class="totalFollowers"><strong><?php
echo $this->profile->getFollowers()->count(); ?></strong>Followers</a></li>

           <li><a href="<?php echo $this->baseUrl((!$currentLoggedInProfile) ? 'profile/' .
$this->profile->getId() . '/following' : 'following' ); ?>" class="totalFollowing"><strong><?php
echo $this->profile->getFollowing()->count(); ?></strong>Following</a></li>

           <li><a href="<?php echo $this->baseUrl((!$currentLoggedInProfile) ? 'profile/' .
$this->profile->getId() . '/tweets' : 'tweets' ); ?>" class="totalMessages"><strong><?php echo
$this->profile->getSentMessages()->count(); ?></strong>Messages</a></li>

        </ul>

        <?php if (!$currentLoggedInUser) : ?>
           <div class="pull-right">
              <?php echo $this->btn()->trackProfile($this->profile); ?>
           </div>
        <?php endif; ?>

     </div>
   </div>

</div>
```

application/views/scripts/widgets/profile/tweets.phtml

```
<?php $this->headScript()->appendFile($this->baseUrl("js/components/tweets.js")); ?>

<script type="text/javascript">
$(document).ready(function() {

   var tweetsSelector = "div#tweets-<?php echo $this->profile->getId(); ?>";

   $(tweetsSelector).tweets({
      url: baseUrl+"/ajax/messages/regular?uid=<?php echo $this->profile->getId();
?>&format=html"
   });

});
</script>

<div id="tweets-<?php echo $this->profile->getId(); ?>" class="media-collection tweets-
collection"></div>
```

application/views/scripts/widgets/profile/followers.phtml

```
<?php $this->headScript()->appendFile($this->baseUrl("js/components/profiles.js")); ?>

<script type="text/javascript">
$(document).ready(function() {
   $('#profiles-followers-collection').profiles({
      url: baseUrl+"/ajax/profiles/followers?uid=<?php echo $this->profile->getId();
?>&format=html",

      loaded: function(){
         $(".profile-container").profiles('initTrackBtn');
      }
   });
});
</script>

<div id="profiles-followers-collection" class="media-collection tracking-content"></div>
```

application/views/scripts/widgets/profile/following.phtml

```
<?php $this->headScript()->appendFile($this->baseUrl("js/components/profiles.js")); ?>

<script type="text/javascript">
$(document).ready(function() {
   $('#profiles-following-collection').profiles({
      url: baseUrl+"/ajax/profiles/following?uid=<?php echo $this->profile->getId();
?>&format=html",

      loaded: function(){
         $(".profile-container").profiles('initTrackBtn');
      }
   });
});

</script>

<div id="profiles-following-collection" class="media-collection tracking-content"></div>
```

application/views/scripts/widgets/profiles/regular.phtml

```
<?php $this->headScript()->appendFile($this->baseUrl("js/components/profiles.js")); ?>

<script type="text/javascript">
$(document).ready(function() {
   $('#profiles-regular-collection').profiles({
      url: baseUrl+"/ajax/profiles/regular?format=html"
   });
});
</script>

<div id="profiles-regular-collection" class="media-collection tracking-content"></div>
```

application/views/scripts/widgets/profiles/corporate.phtml

```
<?php $this->headScript()->appendFile($this->baseUrl("js/components/profiles.js")); ?>

<script type="text/javascript">
$(document).ready(function() {
   $('#profiles-corporate-collection').profiles({
      url: baseUrl+"/ajax/profiles/corporate?format=html"
   });
});
</script>

<div id="profiles-corporate-collection" class="media-collection tracking-content"></div>
```

application/views/scripts/widgets/profiles/search.phtml

```
<?php $this->headScript()->appendFile($this->baseUrl("js/components/profiles.js")); ?>

<script type="text/javascript">

   $(document).ready(function() {
      $('#search-profiles-collection').profiles({
         url: baseUrl + "/ajax/profiles/search?term=<?php echo $this->term; ?>&format=html",
         loaded: function() {
            $(".profile-container").profiles('initTrackBtn');
         }
      });
   });

</script>

<div id="search-profiles-collection" class="media-collection tracking-content"></div>
```

application/modules/ajax/views/scripts/Messages/base-message.phtml

```
<?php
foreach ($this->tweets as $tweet) :

   if ($tweet instanceof \Documents\Message\Base){
      $tw = $tweet;
   }
   else if ($tweet instanceof \Documents\Message\Received\Regular){
      $tw = $tweet->getMessage();
   }
   else if ($tweet instanceof \Documents\Message\Received\Corporate){
      $tw = $tweet->getMessage();
   }
?>

<div class="media-container tweet-container media" id="<?php echo $tw->getId(); ?>">

   <a class="pull-left" href="<?php echo $this->url(array('module'=>'default',
'controller'=>'profile', 'action'=>$tw->getCreator()->getId())); ?>">
      <img alt="<?php echo (string) $tw->getCreator(); ?>" src="<?php echo $tw->getCreator()-
>getProfilepiclink(); ?>" class="img-rounded media-object profile-photo">
   </a>

   <small class="pull-right"><?php echo $this->shorttime($tw->getCreationDate()); ?></small>

   <div class="media-body">

      <h4 class="media-heading">
         <a href="<?php echo $this->url(array('module'=>'default', 'controller'=>'profile',
'action'=>$tw->getCreator()->getId())); ?>">
            <?php echo (string) $tw->getCreator(); ?>
         </a>
      </h4>

      <p><?php echo $tw->getContent(); ?></p>

   </div>

</div>

<?php
endforeach;

if ($this->tweets instanceof Zend_Paginator) {
   if ($this->tweets->getCurrentPageNumber() < $this->tweets->count()) {
      echo '<div class="media"><a href="javascript:void(0);" class="load_more_waves btn btn-
mini btn-info pull-right" style="margin: 0px 12px 9px 0px;"><i class="icon-download-alt"></i>
Load Older Messages</a><div>';
   }
}
```

application/modules/ajax/views/scripts/Messages/regular.ajax.phtml & application/modules/ajax/views/scripts/Messages/corporate.ajax.phtml

```
echo $this->render('Messages/base-message.phtml');
```

application/modules/ajax/views/scripts/Profiles/_user-profile.phtml

```
<?php
$user = Zend_Registry::get("currentUser");

foreach ($this->profiles as $profile) :
?>

<div class="media-container profile-container media" id="<?php echo $profile->getId(); ?>" >

   <a class="pull-left" href="<?php echo $this->url(array('module'=>'default',
'controller'=>'profile', 'action'=>$profile->getId())); ?>">
      <img alt="<?php echo (string) $profile; ?>" src="<?php echo $profile->getProfilepiclink();
?>" class="profile-photo img-rounded media-object" data-src="holder.js/64x64">
   </a>

   <?php if($user->getProfiles()->contains($profile)) : ?>

   <div class="btn-group pull-right">

      <a class="btn btn-mini" href="#">
         <i class="icon-user"></i>
      </a>

      <a class="btn btn-mini dropdown-toggle" data-toggle="dropdown" href="#"><span
class="caret"></span></a>

      <ul class="dropdown-menu">
         <?php echo $this->btn()->editProfile($profile); ?>
         <?php echo $this->btn()->switchToProfile($profile); ?>
         <?php echo $this->btn()->setDefaultProfile($profile); ?>
         <?php echo $this->btn()->deleteProfile($profile); ?>
      </ul>
   </div>

   <?php else: ?>

   <small class="pull-right"><?php echo $this->btn()->trackProfile($profile); ?></small>

   <?php endif; ?>

   <div class="media-body">

      <h4 class="media-heading">
         <a href="<?php echo $this->url(array('module'=>'default', 'controller'=>'profile',
'action'=>$profile->getId())); ?>">
            <?php echo (string) $profile; ?>
         </a>
      </h4>

      <p><?php echo $profile->getBio(); ?></p>

   </div>

</div>

<?php
endforeach;
```

application/modules/ajax/views/scripts/Profiles/regular.ajax.phtml & application/modules/ajax/views/scripts/Profiles/corporate.ajax.phtml

```
echo $this->render('Profiles/_user-profile.phtml');
```

application/modules/ajax/views/scripts/Profiles/_base-profile.phtml

```
<?php
$user = Zend_Registry::get("currentUser");

foreach ($this->profiles as $profile) :
?>

<div class="media-container profile-container media" id="<?php echo $profile->getId(); ?>"
data-profile-id="<?php echo $profile->getId(); ?>">

   <a class="pull-left" href="<?php echo $this->url(array('module'=>'default',
'controller'=>'profile', 'action'=>$profile->getId())); ?>">
      <img alt="<?php echo (string) $profile; ?>" src="<?php echo $profile->getProfilepiclink();
?>" class="img-rounded media-object" data-src="holder.js/64x64">
   </a>

   <?php if(!$user->getProfiles()->contains($profile)) : ?>
   <small class="pull-right"><?php echo $this->btn()->trackProfile($profile); ?></small>
   <?php endif; ?>

   <div class="media-body">

      <h4 class="media-heading">
         <a href="<?php echo $this->url(array('module'=>'default', 'controller'=>'profile',
'action'=>$profile->getId())); ?>">
            <?php echo (string) $profile; ?>
         </a>
      </h4>

      <p><?php echo $profile->getBio(); ?></p>

   </div>

</div>

<?php
endforeach;
```

application/modules/ajax/views/scripts/Profiles/followers.ajax.phtml & application/modules/ajax/views/scripts/Profiles/following.ajax.phtml

```
echo $this->render('Profiles/_base-profile.phtml');
```

application/modules/ajax/views/scripts/Profiles/search.ajax.phtml

```
<?php
$user = Zend_Registry::get("currentUser");

foreach ($this->profiles as $profile) :
?>

<div class="profile-container media" id="<?php echo $profile->getId(); ?>" data-profile-
id="<?php echo $profile->getId(); ?>">

   <a class="pull-left" href="<?php echo $this->url(array('module'=>'default',
'controller'=>'profile', 'action'=>$profile->getId())); ?>">
      <img alt="<?php echo (string) $profile; ?>" src="<?php echo $profile->getProfilepiclink();
?>" class="img-rounded media-object" data-src="holder.js/64x64">
   </a>

   <?php if($user->getProfiles()->contains($profile)) : ?>

   <div class="btn-group pull-right">
      <a class="btn dropdown-toggle" data-toggle="dropdown" href="#">
         <i class="icon-user"></i>
         Me
         <span class="caret"></span>
      </a>
      <ul class="dropdown-menu">
         <li><?php echo $this->btn()->switchToProfile($profile); ?></li>
         <li><?php echo $this->btn()->setDefaultProfile($profile); ?></li>
         <li><?php echo $this->btn()->deleteProfile($profile); ?></li>
      </ul>
   </div>

   <?php else: ?>

   <small class="pull-right"><?php echo $this->btn()->trackProfile($profile); ?></small>

   <?php endif; ?>

   <div class="media-body">

      <h4 class="media-heading">
         <a href="<?php echo $this->url(array('module'=>'default', 'controller'=>'profile',
'action'=>$profile->getId())); ?>">
            <?php echo (string) $profile; ?>
         </a>
      </h4>

      <p><?php echo $profile->getBio(); ?></p>

   </div>

</div>

<?php
endforeach;
```

Παράρτημα Δ: jQuery Custom Plugins

public/js/components/tweets.js

```
var tweetDivs = [];

(function($){

   var defaults = {
      url: baseUrl+"/ajax/message"
   };

   $.fn.tweets = function(method, options) {

      if ( methods[method] ) {
         return methods[ method ].apply( this, Array.prototype.slice.call( arguments, 1 ));
      } else if ( typeof method === 'object' || ! method ) {
         return methods.init.apply( this, arguments );
      } else {
         $.error( 'Method ' +  method + ' does not exist!' );
      }
   };

   var methods = {

      init: function(options) {
         var $element = $(this).first();
         $element.data('tweetsSettings', $.extend({}, defaults, options));

         $element.data('page', 1);
         tweetDivs.push($element.attr('id'));
         methods.loadTweets.apply(this, arguments);
      },

      loadTweets: function(options, nextPage) {

         var $element = $(this).first();
         var settings = $element.data('tweetsSettings');

         if($element.data('page') == 1) {
            nextPage = true;
         }
         else if(typeof nextPage == "undefined") {
            nextPage = false;
         }

         if (!nextPage) {
            $element.prepend('<p class="loading-tweets-icon" style="text-align: center;"><img
src="'+baseUrl+'/img/ajax-loader.gif" width="220" height="19" /></p>');
         }
         else {
            $element.append('<p class="loading-tweets-icon" style="text-align: center;"><img
src="'+baseUrl+'/img/ajax-loader.gif" width="220" height="19" /></p>');
         }

         var geturl;

         if (nextPage) {
            geturl = $.get(
               settings.url,
               {'page': $element.data('page')},
               function(data){
                  $element.find('.load_more_waves:first').remove();
                  $element.data('lastLoad', geturl.getResponseHeader('Tweet-Update-Time'));
                  $element.find('.loading-tweets-icon').remove();
                  $element.data('page', $element.data('page') + 1);
```

```
                $element.append(data);

                $element.find('.load_more_waves').on('click', function(){
                    $element.tweets('loadTweets', {}, true);
                });

            }
        );
    }
    else {
        geturl = $.get(
            settings.url,
            {'after': $element.data('lastLoad')},
            function(data){
                $element.find('.loading-tweets-icon').remove();

                if(typeof options != 'undefined' && typeof options['preLoad'] != 'undefined') {
                    options['preLoad'](data);
                }

                $element.data('lastLoad', geturl.getResponseHeader('Tweet-Update-Time'));
                $element.data("streamItemsCount", 0);

                $element.prepend(data);


                if(typeof options != 'undefined' && typeof options['postLoad'] != 'undefined') {
                    options['postLoad'](data);
                }
            }
        );
    }

    if(typeof options != 'undefined' && typeof options['loaded'] != 'undefined') {
        options['loaded']();
    }
},

updateTweets: function(options) {
    for(var i in tweetDivs) {
        $('#'+tweetDivs[i]).tweets('loadTweets', options);
    }
},

initRefreshBtn: function(){

    var $element = $(this).first();

    $element.parent().parent().find('.widget-head .btn-
refresh:first').unbind('click').bind('click', function(){
        $element.find('.notify-container:first').remove();
        $element.tweets('loadTweets', { }, false);
    });

}
};

})(jQuery);
```

public/js/components/profiles.js

```
(function($){

   var defaults = {
      url: baseUrl+"/ajax/profiles"
   };

   $.fn.profiles = function(method, options) {

      //"this" is already a jquery object

      if ( methods[method] ) {
         return methods[ method ].apply( this, Array.prototype.slice.call( arguments, 1 ));
      } else if ( typeof method === 'object' || ! method ) {
         return methods.init.apply( this, arguments );
      } else {
         $.error( 'Method ' +  method + ' does not exist jQuery.profiles' );
      }
   };

   var methods = {

      init: function(options) {
         var $element = $(this).first();
         $element.data('profilesSettings', $.extend({}, defaults, options));

         methods.loadProfiles.apply(this, arguments);
      },

      loadProfiles: function(options) {

         var $element = $(this).first();
         var settings = $element.data('profilesSettings');

         $element.html('<p class="loading-profiles-icon"><img src="'+baseUrl+'/img/ajax-
loader.gif" width="220" height="19" /></p>');

         var geturl;

         geturl = $.get(

            settings.url,
            {},
            function(data){

               $element.find('.loading-profiles-icon').remove();

               $element.append(data);

               if(typeof options != 'undefined' && typeof options['loaded'] != 'undefined') {
                  options['loaded']();
               }

            }
         );
      },

      initTrackBtn: function(){

         //"this" is already a jquery object
         $this = this;

         return this.each(function(){

            //"this" keyword refers to the native DOM element.
            //need to wrap to jQuery Object
```

```
            var $element = $(this);

            $element.find(".track-btn").unbind('click').bind('click', function(){

               var profileid = $(this).parent().parent().data('profileId');

               $(this).addClass('disabled');

               $.ajax({
                  type: 'POST',
                  data: {'id': profileid},
                  url: baseUrl+'/ajax/profile/track?format=html',
                  dataType: 'html',
                  success: function(data) {

                     //"this" is an AJAX settings Object so i cant call it inside
                     //"success" callback
                     $element.find(".track-btn").replaceWith(data);
                     $element.find(".track-btn").removeClass('disabled');

                     //$this.profiles('initTrackBtn');
                     $element.profiles('initTrackBtn');
                  }
               });
            });
         });
      }
   };

})(jQuery);
```

public/js/components/formValidation.js

```
(function($) {

   var defaults = {};

   $.fn.formValidation = function(method, options) {

      var self = this;

      if (methods[method]) {
         return methods[ method ].apply(this, Array.prototype.slice.call(arguments, 1));
      } else if (typeof method === 'object' || !method) {
         return methods.init.apply(this, arguments);
      } else {
         $.error('Method ' + method + ' does not exist!');
      }
   };

   var methods = {
      init: function(options) {

         var $form = $(this).first();

         //loop fields
         $.each(options.fields, function(fieldIdx, field) {

            var $field = $form.find("#" + field.name);
            var normalizedRules = [];

            $field.bind('keydown', function(e) {

               if (!methods.validKey.call(null, e.which))
                  return true;

               clearTimeout($field.data(e.type + "timer") || {});
```

```
                $field.removeData(e.type + "timer");

                var ms = 50;

                var keydown_timer = setTimeout(function() {

                   if ($field.val().length)
                      $field.data('has-content', true);
                   else
                      $field.data('has-content', false);

                }, ms);

                $field.data(e.type + "timer", keydown_timer);

             });

             //loop tip events
             $.each(options.tip.onEvts, function(tipEvtIdx, tipEvt) {

                $field.bind(tipEvt.name, function(e) {

                   if (!methods.validKey.call(null, e.which))
                      return true;

                   if (typeof tipEvt.conditions != 'undefined') {

                      if (tipEvt.conditions[0].call(null, $field))
                         options.tip.handler.call(null, $field, field.tip);
                   }
                   else if (typeof tipEvt.conditions == 'undefined') {
                      options.tip.handler.call(null, $field, field.tip);
                   }

                });

             });

             //loop rules
             $.each(field.rules, function(ruleIdx, rule) {

                var fixedRule = {
                   'method': rule.name,
                   'args': rule.params || {},
                   'errorMessage': rule.errorMessage || null,
                   'successMessage': rule.successMessage || null
                };

                normalizedRules.push(
                      function() {

                         var ret = checkMethods[ fixedRule.method ].call(null, $field.val(), $field,
fixedRule.args);

                         if (typeof ret == 'object' && ret.state())
                         {
                            $.when(ret.state() != 'undefined').then(function() {
                               options.callbackHandlers.fieldRemoteChecking($field);
                            });

                            ret.done(function(data) {
                               if (!data.valid) {

                                  var m = "";
                                  $.each(data.messages, function(k, v) {
                                     m += v + "\n";
                                  });
                                  options.callbackHandlers.fieldError($field,
```

```
fixedRule.errorMessage || m || "");
                                }
                                else
                                {
                                    var m = "";
                                    $.each(data.messages, function(k, v) {
                                        m += v + "\n";
                                    });
                                    options.callbackHandlers.fieldOk($field,
fixedRule.successMessage || m || "");
                                }
                            });
                        }
                        else if (typeof ret == 'boolean') {

                            if (ret) {

                                if (typeof fixedRule.successMessage != 'undefined')
                                    options.callbackHandlers.fieldOk($field,
fixedRule.successMessage);

                            }
                            else if (!ret) {

                                options.callbackHandlers.fieldError($field,
fixedRule.errorMessage);
                            }
                        }

                        return ret;

                    }
                );
            });//end loop rules

            var taskQueue = new $.taskQueue({tasks: normalizedRules});
            $.extend(taskQueue, options.onkeyup ? {onkeyup: true} : {onkeyup: false});

            $field.data('ValidationUnit', taskQueue);

            $field.bind('keyup', function(e) {

                if (!methods.validKey.call(null, e.which)) {
                    return true;
                }

                var taskValidateUnit = $field.data('ValidationUnit');

                if (taskValidateUnit.onkeyup)
                {
                    if ($field.data(e.type + "timer") instanceof $.timer) {
                        $field.data(e.type + "timer").clearTimer();
                        $field.removeData(e.type + "timer");
                    }

                    var timer = new $.timer(function() {
                        taskValidateUnit.run();
                    }, options.delay, true);

                    $field.data(e.type + "timer", timer);
                }
                else
                { }
            });

        }); //end loop fields

        $form.submit(function(e) {
```

```
            var taskQueues = [];
            var validationSuccess = [];

            //loop each field
            $.each(options.fields, function(fiedIdx, field) {

               var $field = $form.find("#" + field.name);
               var validationUnit = $field.data('ValidationUnit') || null;

               if (typeof validationUnit != null && validationUnit instanceof $.taskQueue) {

                  //has been already checked on keyup
                  if (validationUnit.onkeyup && validationUnit.completed) {
                     validationSuccess.push(validationUnit.success);
                  }
                  else {
                     taskQueues.push(validationUnit);
                  }
               }
            });

            if (taskQueues.length) {

               var taskQueue = new $.taskQueue({taskQueues: taskQueues, sync: false});

               $(taskQueue).on("completeTaskQueues", function(e, doneButError) {

                  validationSuccess.push(!doneButError);

                  for (var i = 0; i < validationSuccess.length; i++)
                  {
                     if (!validationSuccess[i])
                     {
                        options.preventFormSubmit($form);
                        return false;
                     }
                  }
                  options.formSubmit($form);
               });

               taskQueue.run();
            }
            else {

               for (var i = 0; i < validationSuccess.length; i++)
               {
                  if (!validationSuccess[i])
                  {
                     options.preventFormSubmit($form);
                     return false;
                  }
               }

               options.formSubmit($form);
            }

            return false;
         });

      },
      validKey: function(k) {

         if (k == 20 /* Caps lock */
               || k == 16 /* Shift */
               || k == 9 /* Tab */
               || k == 27 /* Escape Key */
               || k == 17 /* Control Key */
```

```
                || k == 91 /* Windows Command Key */
                || k == 19 /* Pause Break */
                || k == 18 /* Alt Key */
                || k == 93 /* Right Click Point Key */
                || (k >= 35 && k <= 40) /* Home, End, Arrow Keys */
                || k == 45 /* Insert Key */
                || (k >= 33 && k <= 34) /*Page Down, Page Up */
                || (k >= 112 && k <= 123) /* F1 - F12 */
                || (k >= 144 && k <= 145)) { /* Num Lock, Scroll Lock */
            return false;
         }

         return true;
      }
   };

   var checkMethods = {
      required: function(value, $element, params) {
         if ($element.val().length > 0)
            return true;
         else
            return false;
      },
      email: function(value, $element, params) {
         return /^((([a-z]|\d|[!#\$%&'\*\+\-\/=\?\^_`{\|}~]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-
\uFFEF])+(\.([a-z]|\d|[!#\$%&'\*\+\-\/=\?\^_`{\|}~]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-
\uFFEF])+)*)|((\x22)((((\x20|\x09)*(\x0d\x0a))?(\x20|\x09)+)?(([\x01-\x08\x0b\x0c\x0e-
\x1f\x7f]|\x21|[\x23-\x5b]|[\x5d-\x7e]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])|(\\([\x01-
\x09\x0b\x0c\x0d-\x7f]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-
\uFFEF])))*(((\x20|\x09)*(\x0d\x0a))?(\x20|\x09)+)?(\x22)))@((([a-z]|\d|[\u00A0-\uD7FF\uF900-
\uFDCF\uFDF0-\uFFEF])|(([a-z]|\d|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])([a-z]|\d|-
|\.|_|~|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])*([a-z]|\d|[\u00A0-\uD7FF\uF900-
\uFDCF\uFDF0-\uFFEF])))\.)+(([a-z]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])|(([a-
z]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])([a-z]|\d|-|\.|_|~|[\u00A0-\uD7FF\uF900-
\uFDCF\uFDF0-\uFFEF])*([a-z]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))$/i.test(value);
      },
      number: function(value, $element) {
         return /^-?(?:\d+|\d{1,3}(?:,\d{3})+)?(?:\.\d+)?$/.test(value);
      },
      // http://docs.jquery.com/Plugins/Validation/Methods/digits
      digits: function(value, $element) {
         return /^\d+$/.test(value);
      },
      // http://docs.jquery.com/Plugins/Validation/Methods/minlength
      minlength: function(value, $element, param) {
         var length = $.isArray(value) ? value.length : $.trim(value).length;
         return length >= param;
      },
      // http://docs.jquery.com/Plugins/Validation/Methods/maxlength
      maxlength: function(value, $element, param) {
         var length = $.isArray(value) ? value.length : $.trim(value).length;
         return length <= param;
      },
      // http://docs.jquery.com/Plugins/Validation/Methods/rangelength
      rangelength: function(value, $element, param) {
         var length = $.isArray(value) ? value.length : $.trim(value).length;
         return (length >= param[0] && length <= param[1]);
      },
      // http://docs.jquery.com/Plugins/Validation/Methods/min
      min: function(value, $element, param) {
         return value >= param;
      },
      // http://docs.jquery.com/Plugins/Validation/Methods/max
      max: function(value, $element, param) {
         return value <= param;
      },
      // http://docs.jquery.com/Plugins/Validation/Methods/range
      range: function(value, element, param) {
```

```
            return this.optional(element) || (value >= param[0] && value <= param[1]);
        },
        remote: function(value, element, params) {

            var xhr = $.ajax({
               'url': params.url,
               'data': params.data,
               'success': function(data) {
                  return data;
               },
               'fail': function() {
               }
            });

            return xhr;
        }
    };

})(jQuery);
```

public/js/components/validateInput.js

```
(function($) {

    var defaults = {};

    $.fn.validateInput = function(method, options) {

        if ( methods[method] ) {
            return methods[ method ].apply( this, Array.prototype.slice.call( arguments, 1 ));
        } else if ( typeof method === 'object' || ! method ) {
            return methods.init.apply( this, arguments );
        } else {
            $.error( 'Method ' +  method + ' does not exist!' );
        }
    };

    var methods = {

        init: function(options) {

            var $element = $(this).first();

            methods.initSide.apply(this, null);

            methods.initFocusEvent.apply(this, arguments);
            methods.initKeydownEvent.apply(this, arguments);
            methods.initKeyupEvent.apply(this, arguments);

        },

        initSide: function() {
            var $element = $(this).first();

            var $sideTip       = $element.next(".help-inline:first").find("p.tip:first");
            var $sideOk        = $element.next(".help-inline:first").find("p.ok:first");
            var $sideError     = $element.next(".help-inline:first").find("p.error:first");
            var $sideChecking  = $element.next(".help-inline:first").find("p.checking:first");

            $element.data('sideTip', $sideTip);
            $element.data('sideOk', $sideOk);
            $element.data('sideError', $sideError);
            $element.data('sideChecking', $sideChecking);
        },

        initFocusEvent: function(options) {
```

```
        var $element = $(this).first();

        $element.focus(function() {

            if (!$element.hasClass('has-error') && !$element.hasClass('has-content')){
                //showTip
                methods.showTip.call($element);
            }
            else if ($element.hasClass('has-error') && !$element.hasClass('has-content')){
                //showTip
                methods.showTip.call($element);
            }
            else if ($element.hasClass('has-error') && $element.hasClass('has-content')){
                //keep display error note
            }
        });
    },

    initKeydownEvent: function(options) {

        var $element = $(this).first();

        var keydown_timer;

        $element.keydown(function(e) {

            if (!methods.validKey.call(null, e.which))
                return true;

            //show tip
            methods.showTip.call($element);

            clearTimeout(keydown_timer);

            var ms = 50;

            keydown_timer = setTimeout(function() {

                if ($element.val().length)
                    $element.addClass('has-content');
                else
                    $element.removeClass('has-content');

            }, ms);
        });
    },

    initKeyupEvent: function(options) {

        var $element = $(this).first();

        var keyup_timer;

        $element.keyup(function(e) {

            if (!methods.validKey.call(null, e.which))
                return true;

            clearTimeout(keyup_timer);

            var ms = 600;

            keyup_timer = setTimeout(function() {

                methods.process.call($element, options);

            }, ms);
```

```
        });
    },

    process: function(options) {

        var $element = this;

        $element.data('sideTip').removeClass('visible');
        $element.data('sideOk').removeClass('visible');
        $element.data('sideError').removeClass('visible');
        $element.data('sideChecking').addClass('visible');

        //server side validation via ajax
        if ("url" in options)
        {
            var params = {};
            params[options.paramName] = $element.val();

            $.get(
                options.url,

                params,

                function(data){

                    $element.data('sideChecking').removeClass('visible');
                    $element.data('sideTip').removeClass('visible');

                    var $appendToElem;

                    if (data.valid) {
                        $appendToElem = $element.data('sideOk');
                        $element.removeClass('has-error');
                    }
                    else {
                        $appendToElem = $element.data('sideError');
                        $element.addClass('has-error');
                    }

                    $appendToElem.addClass('visible');

                    for(msgIdx in data.messages){
                        $appendToElem.html(data.messages[msgIdx]);
                    }
                }
            );
        }
        else if ("required" in options) //client side validation
        {
            $element.data('sideChecking').removeClass('visible');
            $element.data('sideTip').removeClass('visible');

            var data = {'valid': false, 'messages':{}};

            if ($element.val().length > 0){
                data.valid = true;
                data.messages.validFullname = "Name looks great";
            }
            else {
                data.valid = false;
                data.messages.inValidFullname = "Fullname is required";
            }

            var $appendToElem;

            if (data.valid) {
                $appendToElem = $element.data('sideOk');
                $element.removeClass('has-error');
```

```
            }
            else {
               $appendToElem = $element.data('sideError');
               $element.addClass('has-error');
            }

            $appendToElem.addClass('visible');

            for(msgIdx in data.messages){
               $appendToElem.html(data.messages[msgIdx]);
            }
         }
      },

      validKey: function(k) {

         if (k == 20 /* Caps lock */
         || k == 16 /* Shift */
         || k == 9 /* Tab */
         || k == 27 /* Escape Key */
         || k == 17 /* Control Key */
         || k == 91 /* Windows Command Key */
         || k == 19 /* Pause Break */
         || k == 18 /* Alt Key */
         || k == 93 /* Right Click Point Key */
         || ( k >= 35 && k <= 40 ) /* Home, End, Arrow Keys */
         || k == 45 /* Insert Key */
         || ( k >= 33 && k <= 34 ) /*Page Down, Page Up */
         || (k >= 112 && k <= 123) /* F1 - F12 */
         || (k >= 144 && k <= 145 )) { /* Num Lock, Scroll Lock */
            return false;
         }

         return true;
      },

      showTip: function(){

         this.data('sideOk').removeClass('visible');
         this.data('sideError').removeClass('visible');
         this.data('sideChecking').removeClass('visible');
         this.data('sideTip').addClass('visible');
      }

   };

})(jQuery);
```

public/js/components/query.task.queue.js

```
$.taskQueue = function(options) {
   this.settings = $.extend(true, {}, $.taskQueue.defaults, options);
   this.init();
};

$.extend($.taskQueue, {

   defaults: {
      tasks: [],
      taskQueues: [],
      is_processing: false,
      initial_queued_count: 0,
      transactionId: null,
      success: false,
      sync: true,
      completed: false,
      remain_taks:null
```

```
   },

   prototype: {

      init: function() {

         $(this).bind('completeTaskQueue', {context: this}, this.completeTaskQueueHandler);
         $(this).bind('successTaskQueue', {context: this}, this.successTaskQueueHandler);
         $(this).bind('failTaskQueue', {context: this}, this.failTaskQueueHandler);

         $(this).bind('completeTaskQueues', {context: this}, this.completeTaskQueuesHandler);
         $(this).bind('completeTaskQueuesButError', {context: this},
this.completeTaskQueuesButErrorHandler);
         $(this).bind('completeTaskQueuesSuccess', {context: this},
this.completeTaskQueuesSuccessHandler);

         this.tasks = this.settings.tasks || [];
         this.remainingTasks = this.tasks.length;

         this.taskQueues = this.settings.taskQueues || [];
         this.remainingTaskQueues = this.taskQueues.length;

         this.sync = this.settings.sync;

         this.success = null;
      },

      completeTaskQueueHandler: function(e, taskResult, taskIndex) {
         this.completed = true;
         this.remainingTasks = this.tasks.length - taskIndex;
         this.success = taskResult;

         if (this.completed && !this.remainingTasks && this.success) {

            //completed successfuly for all tasks
            $(this).trigger("successTaskQueue");
         }
         else if (this.completed && !this.success) {

            //completed but with errors
            $(this).trigger("failTaskQueue");
         }
      },

      successTaskQueueHandler: function(e) {},

      failTaskQueueHandler: function(e) {},

      completeTaskQueuesHandler: function(e, doneButError) {
         if (doneButError)
            $(this).trigger("completeTaskQueuesButError");
         else
            $(this).trigger("completeTaskQueuesSuccess");
      },

      completeTaskQueuesButErrorHandler: function(e) {},

      completeTaskQueuesSuccessHandler: function(e) {},

      run: function() {

         if (this.tasks.length){
            this.beginTask(0);
         }
         else if (this.taskQueues.length){
            this.beginTaskQueue(0);
         }
```

```
    },

    beginTask: function(index) {
       if (typeof this.tasks[index] !== 'undefined')
       {
          return this.handleTaskResult(
             this.tasks[index].call(this, null),
             ++index
             );
       }
       else {
          // all tasks has completed with  happy end!
          $(this).trigger("completeTaskQueue", [true, index]);

       }
    },

    handleTaskResult: function(taskResult, taskIndex) {

       var taskQueue = this;

       //is delayed proccess
       if (typeof taskResult === 'object' && taskResult instanceof $.timer) {
          $(taskResult).on("hit", function() {
             var f = this.action;
             this.stop();
             this.action = function() {
             };
             taskQueue.handleTaskResult(f.call(null), taskIndex);
          });
       }
       //is ajax process so {ret} must be checked on done event of request to proceed task
unit
       else if (typeof taskResult === 'object' && taskResult.state()) {

          taskResult.nextTask = taskIndex;
          taskResult.taskQueue = this;

          taskResult.done(function(data) {

             if (data.valid) {
                taskQueue.beginTask(taskResult.nextTask);
             }
             else if (!data.valid) {
                $(taskQueue).trigger("completeTaskQueue", [false, taskIndex]);
             }

          });
       }
       else if (typeof taskResult === 'boolean') {

          if (taskResult) {
             this.beginTask(taskIndex);
          }
          else if (!taskResult) {
             $(this).trigger("completeTaskQueue",  [false, taskIndex]);
          }
       }
    },

    beginTaskQueue: function(index) {

       var outerTaskQueue = this;

       if(this.sync === false)//execute anyway, ASYNCHRONOUS
       {
          var pendingTaskQueues = this.taskQueues.length;
```

```
            $.each(this.taskQueues, function(taskQueueIdx, taskQueueItem){

               var innerTaskQueue = taskQueueItem;

               $(innerTaskQueue).on("completeTaskQueue", function(e, taskResult, taskIndex)
{
                  pendingTaskQueues--;

                  if (pendingTaskQueues === 0) {

                     var doneButError = false;

                     $.each(outerTaskQueue.taskQueues, function(i, v) {

                        if (v.completed && v.success)
                           return true;
                        else {
                           doneButError = true;
                           return false;//break each loop
                        }
                     });

                     $(outerTaskQueue).trigger("completeTaskQueues", [ doneButError ]);
                  }
               });

               innerTaskQueue.run();
            });
         }
         else if (this.sync === true)//waiting for continue to next, SYNCHRONOUS
         {
            if (typeof this.taskQueues[index] !== 'undefined') {

               var innerTaskQueue = new $.taskQueue({
                  tasks: this.taskQueues[index].tasks
               });

               $(innerTaskQueue).on("completeTaskQueue", function(e, taskResult, taskIndex)
{
                  outerTaskQueue.beginTaskQueue(++taskIndex);
               });

               innerTaskQueue.run();
            }
            else {
               //console.log('complete all!!!!');
            }
         }
      }
   }
});
```

public/js/components/widgets.js

```
(function($){

   $.fn.widgets = function(method, options) {
      // Method calling logic
      if ( methods[method] ) {
         return methods[ method ].apply( this, Array.prototype.slice.call( arguments, 1 ));
      } else if ( typeof method === 'object' || ! method ) {
         return methods.init.apply( this, arguments );
      } else {
         $.error( 'Method ' +  method + ' does not exist on Kowor.waves' );
      }
   };
```

```
   var methods = {
      init: function(options) {
         options = $.extend({}, options, {
            handle: '.widget-head',
            opacity: 0.6,
            placeholder: 'widget-placeholder',
            start: methods.start,
            stop: methods.stop
         });
         $(this).sortable(options);
      },
      start: function(event, ui) {
         var column = ui.item.parent().parent().children().index(ui.item.parent()) + 1;
         var order = ui.item.index();
         ui.item.data('columnorder', { column: column, order: order });
      },
      stop: function(event, ui) {

         var data = "updateorder=1";
         var widgets = $(this).parent().parent().find('.widget');

         widgets.each(function(key, widget){

            var column = $(widget).parent().parent().children().index($(widget).parent()) + 1;
            var order = $(widget).index();
            $(widget).data('data-col', column);
            $(widget).data('data-order', order);

            data += "&widgets[]=" + $(widget).attr('id') + "-" + $(widget).data('data-col') + "-" +
$(widget).data('data-order');

         });

         var pageName = ui.item.attr('data-page-name');
         var widgetname = ui.item.attr('id');

         data += '&pageName=' + pageName + '&widgetname=' + widgetname;

         $.ajax({
            type: 'POST',
            data: data,
            url: baseUrl + '/Widget/move/'
         });
      }
   };
})( jQuery );
```

Παράρτημα Ε: WebSocket - Server Side

```
var io        = require("socket.io").listen(4001);
var redis      = require('redis');

io.configure(function(){

   io.set("log level", 1);

   /* Handshake/authorization mechanism
    *
    * Through this we can tell socket.io to invoke a user-defined function
    * whenever a new Websocket connection is incoming.
    * The important point is, that it is called,
    * before the connection is completed.
    *
    */
   io.set('authorization', function(data, callback) {

      if (data.headers.cookie) {
         callback(null, true);
      }
      else {
         return accept('No cookie transmitted.', false);
      }
   });
});

var cloud_people = io.of('/gnnect');

cloud_people.on('connection', function(socket){

   socket.on('subscribe', function (channel) {

      var redis_cli = redis.createClient();

      redis_cli.psubscribe("user:" + channel + ":*");

      redis_cli.on('psubscribe', function(pattern, count){
         //console.log('subscribe to ' + pattern);
      });

      redis_cli.on("pmessage", function (pattern, channel, key) {
         socket.emit('message', key);
      });
   });

   socket.on('disconnect', function () {

   });

   socket.on('reconnect', function () {
      //console.log("Try reconnecting");
   });

   socket.on('error', function (data) {
      //console.log(data);
   });
});
```

Παράρτημα Z: WebSocket - Client Side

```
$(document).ready(function() {

   try {
      var conn_options = {
         'force new connection': false,
         'sync disconnect on unload': true,
         'reconnect': false,
         'reconnection delay': 500
      };

      socket = io.connect('http://localhost:4001/gnnect', conn_options);

      socket.on('connect', function(data) {
         socket.emit('subscribe', userLoggedIn);
      });

      socket.on('disconnect', function() {
         console.log('disconnect');
      });

      socket.on('message', function(message) {

         var jsonMessage = jQuery.parseJSON(message);

         if ("messagefor" in jsonMessage)
         {
            var tweetsSelector = '';

            if (jsonMessage.messageType == "regular") {
               tweetsSelector = 'div#regular-tweets-' + currentProfile;
            }
            else if (jsonMessage.messageType == "corporate") {
               tweetsSelector = 'div#corporate-tweets-' + currentProfile;
            }

            if (jsonMessage.messagefor == currentProfile) {
               $(tweetsSelector).data("streamItemsCount", {'inc': 1});
            }
            else {

            }
         }
         else if ("trackfor" in jsonMessage)
         {
            if (jsonMessage.trackfor == currentProfile) {
               $('#profile-stats-' + currentProfile).data("totalFollowers", {'inc': 1});
            }
         }
         else if ("untrackfor" in jsonMessage)
         {
            if (jsonMessage.untrackfor == currentProfile) {
               $('#profile-stats-' + currentProfile).data("totalFollowers", {'decr': 1});
            }
         }

      });
   }
   catch (ex) {
      console.log(ex);
   }

   window.onunload = function() {
      socket.disconnect();
```

```
    }
});
```