



# «Ανάπτυξη εφαρμογής βασισμένη στο σύστημα *RethinkDB*»

---

**Αναστασάς Αναστάσιος**

[AM: 3166 – Email: csd3166@csd.uoc.gr]

Επιβλέπων Καθηγητής:

**Κος Δημήτριος Πλεξουσάκης**

“Ο σκοπός της αναφοράς αυτής είναι να παρουσιάσει την εφαρμογή (PushUP) η οποία είναι βασισμένη στην λειτουργία ‘Live feeds’ που προσφέρει το σύστημα RethinkDB. Θα αναλυθούν τα χαρακτηριστικά, ο τρόπος λειτουργίας της εφαρμογής καθώς και οι τεχνολογίες που χρησιμοποιήθηκαν.”

## Περιεχόμενα

1	Εισαγωγή .....	3
1.1	Περιγραφή .....	3
1.2	Τι ακολουθεί; .....	3
2	Τεχνική Σχεδίαση & Υλοποίηση .....	4
2.1	Web Application ΜΗ συνδεδεμένου χρήστη .....	4
2.1.1	Εγγραφή Χρήστη (Signup) .....	4
2.1.2	Είσοδος Χρήστη (Login) .....	5
2.2	Web Application συνδεδεμένου χρήστη .....	5
2.2.1	Ταμπλό (Dashboard) .....	5
2.2.2	Ρυθμίσεις (Settings) .....	6
2.2.3	Πρόσθετα (Extras) .....	8
3	Σενάρια Χρήσης .....	9
3.1	Αρχική σελίδα .....	9
3.2	Εγγραφή χρήστη (Signup) .....	10
3.3	Είσοδος χρήστη .....	12
3.4	Λειτουργίες συνδεδεμένου χρήστη .....	13
3.4.1	Dashboard .....	13
3.4.2	Group Settings .....	16
3.4.3	Account settings .....	17
3.4.4	Extras .....	18
4	Αρχιτεκτονική & απαιτήσεις συστήματος .....	19
4.1	RethinkDB <sup>[1]</sup> .....	19
4.2	Front-end .....	21
4.2.1	AngularJS <sup>[2]</sup> .....	21
4.2.2	Bootstrap 4 <sup>[3]</sup> .....	22
4.2.3	Επικοινωνία Client-Server .....	22
4.3	Back-end .....	26
4.3.1	NodeJS <sup>[4]</sup> .....	26
4.3.2	Security <sup>[5], [6]</sup> .....	27
4.4	Socket.IO <sup>[7]</sup> .....	27
4.5	Απαιτήσεις συστήματος .....	29
4.5.1	Προετοιμασία του περιβάλλοντος .....	29
4.5.2	Εκκίνηση συστήματος .....	30
5	Επίλογος .....	31
5.1	Συμπεράσματα .....	31
5.1.1	Σχεδιασμός .....	31
5.1.2	Υλοποίηση .....	31
5.2	Μελλοντικές βελτιστοποιήσεις και νέοι στόχοι .....	32
6	Βιβλιογραφία .....	34

# 1 Εισαγωγή

## 1.1 Περιγραφή

Το PUP (Push UP) είναι μια εφαρμογή συγχρονισμού δεδομένων που εκτελείται σε φυλλομετρητή. Οι χρήστες θα μπορούν κάνοντας αρχικά εγγραφή με τα ελάχιστα δυνατά στοιχεία τους και έπειτα είσοδο στην εφαρμογή, να μοιράσουν άμεσα τα δεδομένα τους μεταξύ των διαφορετικών συσκευών τους. Οι χρήστες θα έχουν την δυνατότητα να φτιάξουν πολλαπλές ομάδες, πράγμα που θα τους επιτρέπει να στείλουν διαφορετικά δεδομένα ή αρχεία ανά ομάδα. Συνεπώς, εκτός από το να διαμοιράσουν δεδομένα θα μπορέσουν και να τα οργανώσουν σε ομάδες ανάλογα με τις ανάγκες τους. Επιπλέον, διασφαλίζεται και η μονιμότητα των δεδομένων τους για όσο χρονικό διάστημα χρειαστεί εκτός εάν προβούν σε κάποια ενέργεια διαγραφής όπου, η οποία είναι μόνιμη και μη αναστρέψιμη. Επιπλέον, προσφέρεται η δυνατότητα στους χρήστες να κάνουν τροποποιήσεις στον λογαριασμό τους, καθώς επίσης και στις ομάδες που έχουν δημιουργήσει ανά πάσα στιγμή ([§2.2.2](#)).

Σκοπός της εφαρμογής είναι η άμεση αποστολή μιας πληροφορίας, ενός συνδέσμου ή ενός αρχείου από ένα smartphone σε έναν υπολογιστή, χωρίς την άμεση υλική επαφή των δύο ηλεκτρονικών συσκευών.

## 1.2 Τι ακολουθεί;

Στην παρούσα αναφορά θα γίνει μια λειτουργική και τεχνική παρουσίαση της εφαρμογής. Πιο συγκεκριμένα :

- **Κεφάλαιο 2<sup>ο</sup>:** περιγραφή και ανάλυση των λειτουργικών χαρακτηριστικών της εφαρμογής.
- **Κεφάλαιο 3<sup>ο</sup>:** εικονογραφημένη παρουσίαση των σεναρίων χρήσης της εφαρμογής.
- **Κεφάλαιο 4<sup>ο</sup>:** περιγραφή της αρχιτεκτονικής του συστήματος, αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν και με ποιον τρόπο και τέλος, αναφορά στις απαιτήσεις λειτουργίας του συστήματος.
- **Κεφάλαιο 5<sup>ο</sup>:** περιγραφή των χαρακτηριστικών και αναβαθμίσεων που θα ενσωματωθούν σε μελλοντικές εκδόσεις.

## 2 Τεχνική Σχεδίαση & Υλοποίηση

### 2.1 Web Application ΜΗ συνδεδεμένου χρήστη

Ένας νέος χρήστης που επισκέπτεται είτε για πρώτη φορά είτε είναι είδη μέλος της εφαρμογής αντικρίζει την σελίδα καλωσορίσματος ([Σενάριο 1<sup>ο</sup>](#)) όπου από εκεί μπορεί να επιλέξει και να κάνει 2 ενέργειες:

- εγγραφή στην εφαρμογή εφόσον δεν διαθέτει λογαριασμό και επισκέπτεται πρώτη φορά την εφαρμογή
- είσοδο στον λογαριασμό του εάν είναι είδη μέλος ή έπειτα από μια επιτυχημένη εγγραφή.

Στο κάτω μέρος τις εκάστοτε σελίδας (home, login, signup, aboutUs) θα βρει κατάλληλα links προς της παραπάνω ενέργειες καθώς και link για να επιστρέψει στην αρχική σελίδα εάν το επιθυμεί ή να διαβάσει την σελίδα 'About Us'.

#### 2.1.1 Εγγραφή Χρήστη (Signup)

Κατά την επίσκεψη στην σελίδα εγγραφής θα δει κανείς, μια φόρμα με 4 υποχρεωτικά πεδία που πρέπει να συμπληρωθούν προκειμένου να γίνει επιτυχώς η εγγραφή ([Σενάριο 2<sup>ο</sup>](#)). Αυτά είναι:

- **Nickname:** ένα όνομα το οποίο θα εμφανίζεται στα μηνύματα τα οποία θα στέλνει ο χρήστης. Τα κριτήρια ορθότητας είναι:
  - Να περιέχει τουλάχιστον 3 χαρακτήρες (επιτρέπονται σύμβολα & αριθμοί).
  - Να μην υπερβαίνει τους 15 χαρακτήρες.
- **Email:** ένα έγκυρο email με το οποίο θα συνδέεται στην εφαρμογή. Τα κριτήρια ορθότητας είναι:
  - Έγκυρο θεωρείται ένα email τις μορφής example@example.gr
  - Να μην υπάρχει άλλος λογαριασμός με το ίδιο email (αφού πληκτρολογηθεί ένα έγκυρο email τότε γίνεται έλεγχος στην βάση δεδομένων για την μοναδικότητα του).
- **Password:** ο κωδικός σύνδεσης στην εφαρμογή. Τα κριτήρια ορθότητας είναι:
  - Να περιέχει τουλάχιστον 8 χαρακτήρες (επιτρέπονται σύμβολα & αριθμοί).
  - Να μην υπερβαίνει τους 20 χαρακτήρες.
  - Να περιέχει τουλάχιστον 1 κεφαλαίο ΚΑΙ έναν πεζό χαρακτήρα.
  - Να περιέχει τουλάχιστον 1 σύμβολο Ή αριθμό.
- **Confirm Password:** να είναι ίδιος με τον κωδικό πρόσβασης που πληκτρολόγησε ακριβώς πιο πάνω στην φόρμα εγγραφής.

Τέλος, πατώντας το κουμπί 'Sign UP' που βρίσκεται στο κάτω μέρος της φόρμας και δεδομένου ότι έχουν πληκτρολογηθεί σωστά όλα τα στοιχεία που απαιτούνται ([Σενάριο 4<sup>ο</sup>](#)),

PUP – PushUP

Data synchronizer between two or more devices.

γίνεται η εγγραφή και ο χρήστης μπορεί πλέον να συνδεθεί στην εφαρμογή. Κατάλληλο μήνυμα θα εμφανιστεί κατά την επιτυχή εγγραφή του χρήστη και μετά από 5 δευτερόλεπτα γίνεται ανακατεύθυνση στην σελίδα εισόδου της εφαρμογής ([Σενάριο 5<sup>ο</sup>](#)). Επίσης, μηνύματα λάθους εμφανίζονται κατά την διάρκεια συμπλήρωσης της φόρμα ([Σενάριο 3<sup>ο</sup>](#)) αλλά και στην περίπτωση που κάτι πάει στραβά με την εγγραφή.

### 2.1.2 Είσοδος Χρήστη (Login)

Κατά την επίσκεψη στην σελίδα εισόδου της εφαρμογής, ο χρήστης θα βρει μια φόρμα ([Σενάριο 6<sup>ο</sup>](#)). Η φόρμα αυτή αποτελείται από τα πεδία email και password τα οποία απαιτούνται για την είσοδο του χρήστη και τα οποία εκείνος όρισε κατά την εγγραφή του. Αφού πατηθεί το κουμπί 'Login' στο κάτω μέρος της σελίδας τότε τα πιθανά σενάρια είναι:

- **Επιτυχημένη** σύνδεση! Αυτό σημαίνει ότι ο χρήστης συνδέεται στην εφαρμογή και γίνεται ανακατεύθυνση στην Αρχική Σελίδα ([Σενάριο 8<sup>ο</sup>](#)). Επίσης κατά την σύνδεση του δημιουργείται και ένα cookie ώστε ο χρήστης να παραμένει συνδεδεμένος. Υπάρχουν δύο επιλογές που μπορεί να κάνει ο χρήστης για την διάρκεια του cookie πριν κάνει είσοδο, αυτές είναι:

- Εάν **δεν επιλεγεί** το 'Remember me' τότε το cookie έχει διάρκεια όσο είναι ανοιχτός ο browser,
- Εάν **επιλεγεί** το 'Remember me' τότε το cookie που δημιουργείται έχει διάρκεια 7 ημέρες από την ημέρα τις σύνδεσης.

Να σημειώσουμε εδώ ότι τα παραπάνω ακυρώνονται είτε στην περίπτωση που ο χρήστης κάνει έξοδο από μόνος του από την σελίδα είτε από την σκόπιμη αποσύνδεση του στην περίπτωση αλλαγής κωδικού πρόσβασης του λογαριασμού του από κάποια άλλη συσκευή όπως περιγράφεται στην παράγραφο [2.2.2.2](#).

- **Αποτυχημένη** σύνδεση ! Αυτό σημαίνει ότι τα στοιχεία που έχει πληκτρολογήσει ο χρήστης είναι λάθος και θα πρέπει να προσπαθήσει πάλι. Κατάλληλο μήνυμα θα εμφανιστεί. ([Σενάριο 7<sup>ο</sup>](#))

## 2.2 Web Application συνδεδεμένου χρήστη

### 2.2.1 Ταμπλό (Dashboard)

Το ταμπλό ή dashboard αποτελεί τον βασικό πυρήνα της εφαρμογής. Μέσω του οποίου θα έχει την δυνατότητα ο χρήστης να δημιουργεί νέες ομάδες καθώς και να δει μια λίστα με όλες τις ομάδες που διαθέτει ([Σενάριο 9<sup>ο</sup>](#)). Επίσης μετά από μια ανανέωση της σελίδας ή νέα σύνδεση από άλλη συσκευή, στο dashboard θα προβάλλονται οι καρτέλες των ομάδων που παρέμειναν ανοιχτές από προηγούμενη σύνδεση. Τέλος, στην εργαλειοθήκη αριστερά της εφαρμογής, κάτω από το Dashboard, φιλοξενούνται οι λειτουργίες:

- Create group ([Σενάριο 10<sup>ο</sup>](#))
- My groups ([Σενάριο 16<sup>ο</sup>](#))

### 2.2.1.1 Δημιουργία Ομάδας (Create Group)

Μόλις πατηθεί το κουμπί 'Create Group' θα εμφανιστεί ένα πάνελ, όπου ο χρήστης μπορεί να πληκτρολογήσει το όνομα τις ομάδας που θέλει να δημιουργήσει ([Σενάριο 10<sup>ο</sup>](#)). Μοναδικός περιορισμός είναι:

- Το όνομα πρέπει να περιέχει τουλάχιστον 3 χαρακτήρες.

Με την δημιουργία της ομάδας θα ανοίξει μια νέα καρτέλα στο dashboard που θα αφορά την ομάδα που μόλις δημιουργήθηκε ([Σενάριο 11<sup>ο</sup>](#)). Επίσης θα γίνει προσθήκη της ομάδας στη λίστα των ομάδων του χρήστη ([Σενάριο 16<sup>ο</sup>](#)). Να σημειώσουμε εδώ, ότι κάθε ομάδα θα εμφανίζει ως πρώτο μήνυμα την ημέρα και ώρα δημιουργίας της, καθώς και τον χρήστη που την δημιούργησε. Αυτό το μήνυμα θα είναι αδύνατον να διαγραφεί, σε αντίθεση με τα υπόλοιπα που θα μπορεί να τα διαγράψει εάν το επιθυμεί ([Σενάριο 15<sup>ο</sup>](#)).

Τέλος στην κορυφή κάθε καρτέλα μιας ομάδας, υπάρχει μια μικρή φόρμα με την οποία ο χρήστης θα μπορεί να ανεβάσει κείμενα ή αρχεία. Πληκτρολογώντας αυτό που θέλει να μοιράσει στις συσκευές του και πατώντας το κουμπί 'ΥΡ' το μήνυμα αποστέλλεται στις συσκευές με τις οποίες είναι συνδεδεμένος εκείνη την στιγμή, καθώς επίσης και στη βάση δεδομένων όπου και αποθηκεύεται ([Σενάριο 12<sup>ο</sup>](#)). Επιπρόσθετα, δεξιά της φόρμας αυτής, υπάρχει ένας σελιδοδείκτης με τον οποίο ο χρήστης μπορεί να επισυνάψει ένα ή περισσότερα αρχεία ([Σενάριο 13<sup>ο</sup>](#)).

### 2.2.1.2 Λίστα των ομάδων μου (My groups)

Όπως αναφέρθηκε πιο πάνω ο χρήστης μπορεί να επιλέγει ποιές από τις ομάδες του επιθυμεί να έχει ανοιχτές. Αυτό σημαίνει ότι μπορεί να ανοίγει και να κλείνει οποιοδήποτε ομάδα. Επομένως, θα πρέπει να υπάρχουν όλες οι ομάδες του χρήστη διαθέσιμες κάπου ώστε να μπορεί να τις ξανά βρει. Αυτό γίνεται με την βοήθεια του My Groups (Dropdown) που βρίσκεται στην εργαλειοθήκη στα αριστερά ([Σενάριο 16<sup>ο</sup>](#)), πατώντας το ανοίγει μια λίστα ταξινομημένη αλφαβητικά με τις ομάδες του χρήστη.

### 2.2.2 Ρυθμίσεις (Settings)

Σε αυτήν την κατηγορία ο χρήστης έχει τις εξής 2 ενέργειες:

- Επεξεργασία των ομάδων του ([Σενάριο 17<sup>ο</sup>](#)). Αυτή η ενέργεια περιλαμβάνει τη δυνατότητα μετονομασίας ή διαγραφής μιας ομάδας.
- Επεξεργασία των πληροφοριών του λογαριασμού του ([Σενάριο 21<sup>ο</sup>](#)). Αυτή η ενέργεια περιλαμβάνει τη δυνατότητα αλλαγής του ονόματος του χρήστη (Nickname) ή αλλαγή του προσωπικού κωδικού του (Password).

### 2.2.2.1 Ρυθμίσεις ομάδων (Groups settings)

Σε αυτήν την σελίδα θα εμφανίζεται ένας πίνακας με όλες της ομάδες του χρήστη ταξινομημένες αλφαβητικά. Για κάθε μια από τις ομάδες ο χρήστης θα έχει δύο επιλογές ([Σενάριο 18<sup>ο</sup>](#)):

- **Αλλαγή ονόματος.** Πατώντας το ανάλογο κουμπί θα βλέπει ένα αναδυόμενο παράθυρο στο οποίο υπάρχει μια φόρμα με το παρόν όνομα της ομάδας καθώς και ένα πεδίο για να πληκτρολογήσει το νέο όνομα της ομάδας ([Σενάριο 19<sup>ο</sup>](#)). Για το νέο όνομα ισχύουν οι κανόνες που περιγράφηκαν στην ενότητα [2.2.1.1](#) – Create Group. Επιλέγοντας ένα νέο όνομα και εφαρμόζοντας την αλλαγή ενημερώνεται ανάλογα η βάση δεδομένων, ο πίνακας με τις ομάδες που αναφέρθηκε παραπάνω καθώς και οι άλλες συσκευές του χρήστη, για την αλλαγή του ονόματος.
- **Διαγραφή.** Επιλέγοντας την διαγραφή εμφανίζεται ένα αναδυόμενο παράθυρο στο οποίο ο χρήστης πρέπει να επιβεβαιώσει την διαγραφή της ομάδας ([Σενάριο 20<sup>ο</sup>](#)). Να σημειωθεί εδώ, ότι η διαγραφή είναι μόνιμη και μη αναστρέψιμη. Όταν μία ομάδα διαγραφεί, τότε, γίνεται και η διαγραφή του πίνακα τόσο από την μεριά του client αλλά και την βάση δεδομένων, όλα τα μηνύματα της ομάδας διαγράφονται και αυτά.  
Τέλος, οι συνδεδεμένες συσκευές του χρήστη ενημερώνονται για την διαγραφή της ομάδας.

### 2.2.2.2 Ρυθμίσεις λογαριασμού χρήστη (Account settings)

Σε αυτήν την σελίδα υπάρχει μια φόρμα που επιτρέπει την τροποποίηση των στοιχείων του χρήστη, εκτός από το Email το οποίο δεν μπορεί να αλλάξει ([Σενάριο 21](#)). Αναλυτικότερα μέσω αυτής τις φόρμας μπορούν να συντελεστούν οι παρακάτω διαφορετικές ενέργειες:

- **Αλλαγή ονόματος (Nickname).** Ο χρήστης έχει την δυνατότητα να αλλάξει το όνομα του όσες φορές επιθυμεί. Για να γίνει αυτό, θα πρέπει να πληκτρολογήσει ένα έγκυρο όνομα και στην συνέχεια τον ισχύοντα κωδικό πρόσβασής του έτσι ώστε να μπορέσει να αλλάξει επιτυχώς το όνομα. Ωστόσο, και εδώ ισχύουν οι ίδιες απαιτήσεις και κανόνες με αυτούς που ίσχυαν και κατά την διαδικασία της εγγραφής ([§2.1.1](#)).
- **Αλλαγή κωδικού πρόσβασης (Password).** Ο χρήστης μπορεί να αλλάξει τον ισχύοντα κωδικό πρόσβασής του. Για να γίνει αυτό είναι απαραίτητο να πληκτρολογήσει τον νέο κωδικό δύο φορές όπως φαίνεται και στο [σενάριο 21](#). Ισχύουν οι κανόνες ορθότητας που περιγράφηκαν στην [§2.1.1](#) για το Password. Έπειτα, θα πρέπει και πάλι να πληκτρολογήσει τον ισχύοντα κωδικό του.

- **Αλλαγή ονόματος και κωδικού πρόσβασης.** Ο χρήστης μπορεί να αλλάξει το όνομα και τον κωδικό του ταυτόχρονα. Θα πρέπει να γίνουν όλες οι ενέργειες σύμφωνα με την παραπάνω περιγραφή.

Τέλος, να σημειώσουμε εδώ ότι στην περίπτωση αλλαγής του κωδικού του χρήστη, τότε αποσυνδέεται από οποιαδήποτε συσκευή είναι συνδεδεμένος εκείνη την στιγμή, καθώς και από οποιαδήποτε συσκευή έχει αποθηκευτεί το cookie. Ακόμη κι αν προσπαθήσει να συνδεθεί αργότερα, πάλι θα αποσυνδεθεί καθώς τα στοιχεία του χρήστη που θα υπάρχουν στο cookie θα είναι λανθασμένα.

### 2.2.3 Πρόσθετα (Extras)

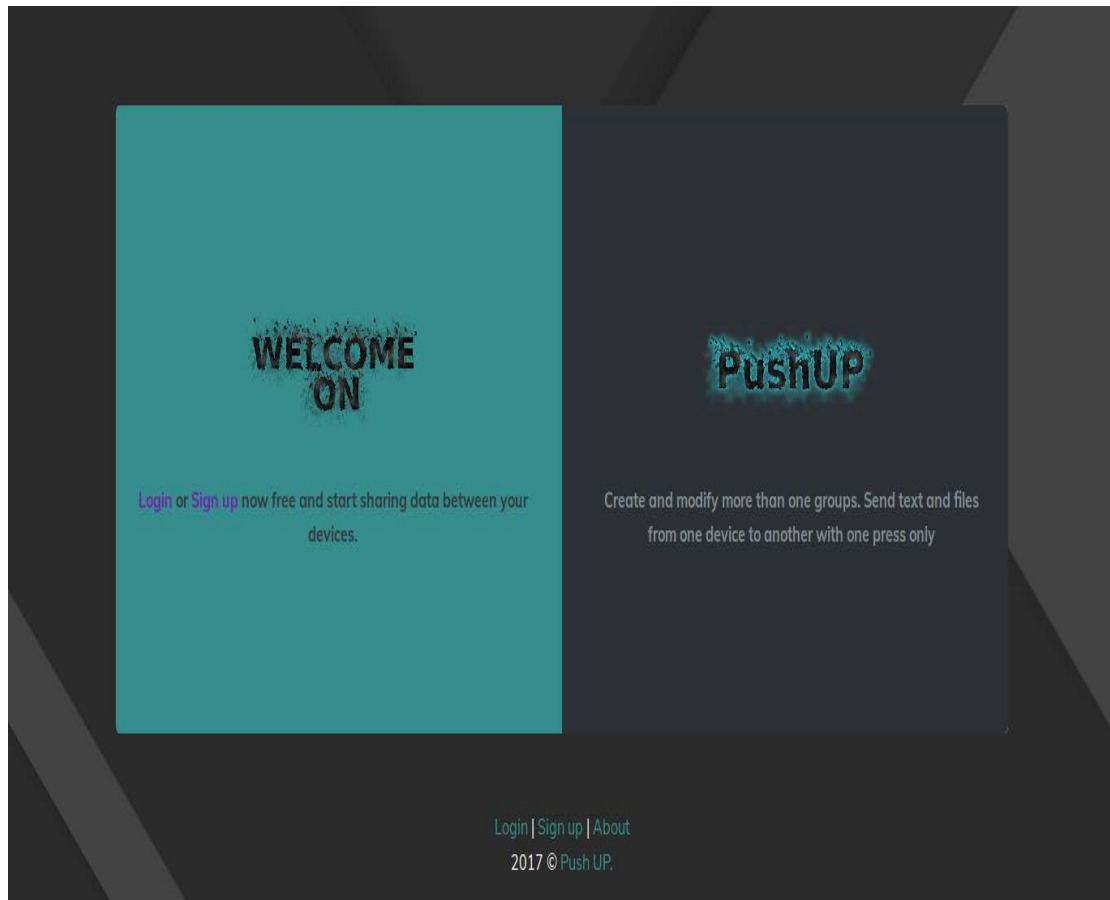
#### 2.2.3.1 Σχετικά με εμάς (About us)

Στην σελίδα αυτήν υπάρχει μια μικρή περιγραφή της εφαρμογής καθώς και μερικές πληροφορίες για τον δημιουργό της εφαρμογής ([Σενάριο 22<sup>ο</sup>](#)).



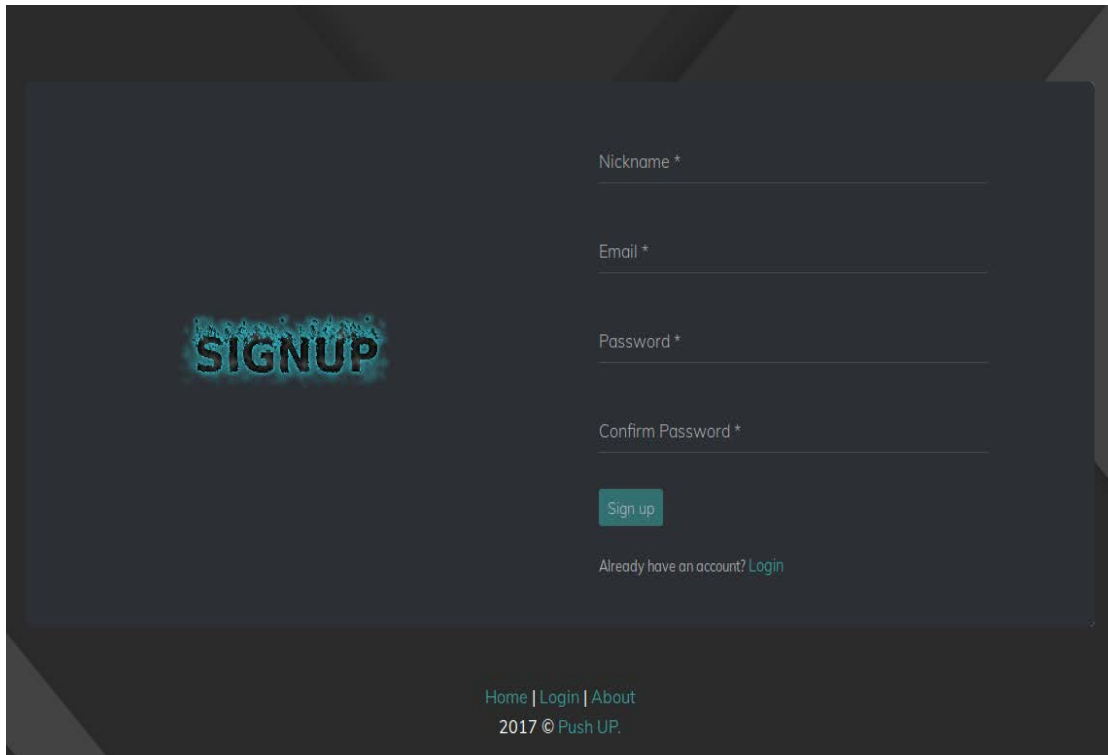
## 3 Σενάρια Χρήσης

### 3.1 Αρχική σελίδα



Σενάριο 1: Η αρχική σελίδα που αντικριζει ένας χρήστης.

## 3.2 Εγγραφή χρήστη (Signup)



**SIGNUP**

Nickname \*

Email \*

Password \*

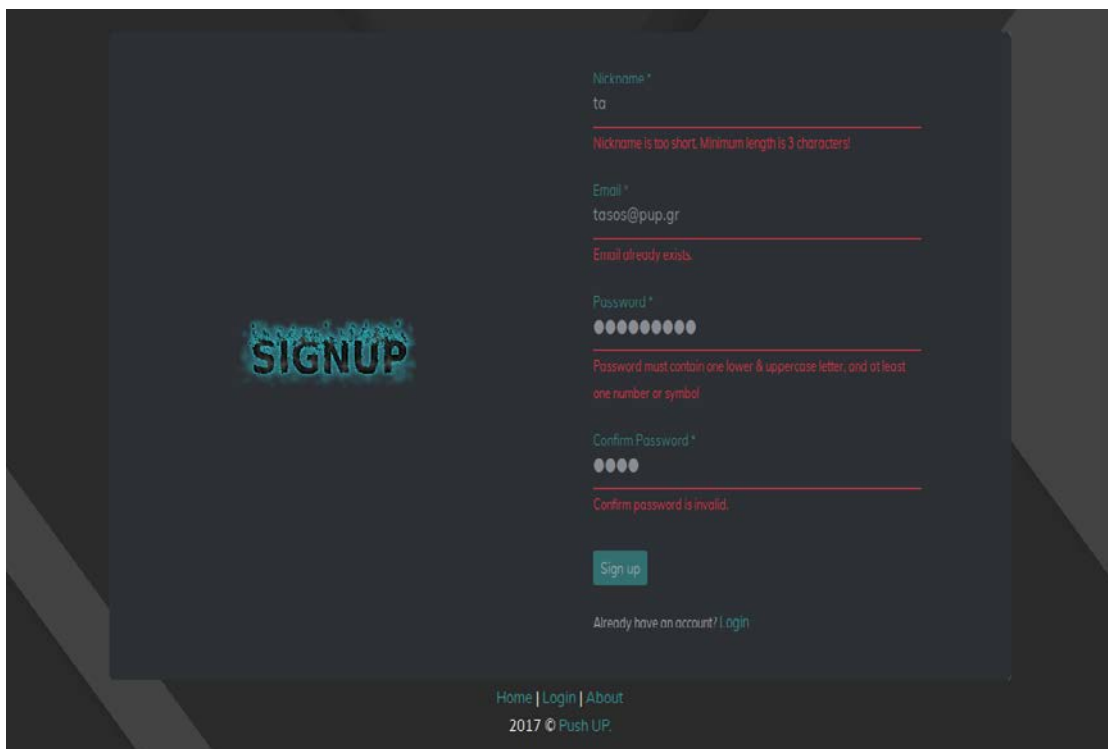
Confirm Password \*

Sign up

Already have an account? [Login](#)

[Home](#) | [Login](#) | [About](#)  
2017 © Push UP.

Σενάριο 2: Φόρμα εγγραφής. Όλα τα πεδία της είναι υποχρεωτικά.



**SIGNUP**

Nickname \*

ta

Nickname is too short. Minimum length is 3 characters!

Email \*

tasos@pup.gr

Email already exists.

Password \*

●●●●●●●●

Password must contain one lower & uppercase letter, and at least one number or symbol

Confirm Password \*

●●●●

Confirm password is invalid.

Sign up

Already have an account? [Login](#)

[Home](#) | [Login](#) | [About](#)  
2017 © Push UP.

Σενάριο 3: Μη έγκυρη φόρμα εγγραφής με εμφάνιση κατάλληλων μηνυμάτων λάθους.

Nickname \*

tasos

Email \*

tasos@pup.gr

Password \*

Confirm Password \*

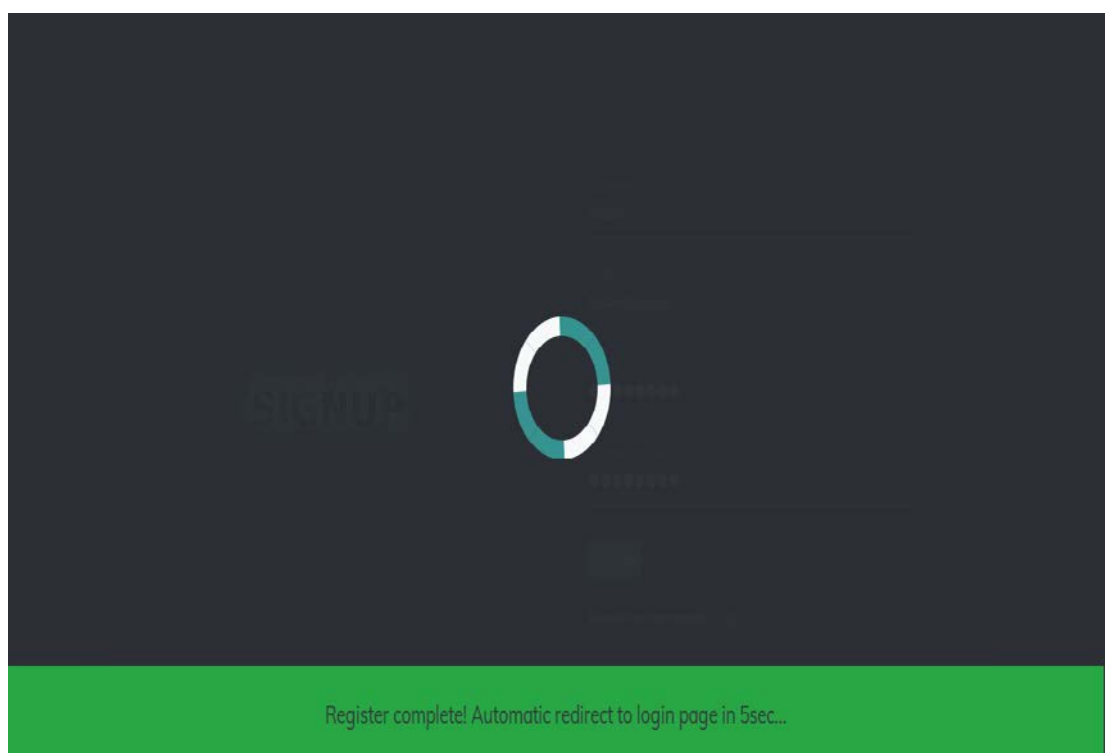
Sign up

Already have an account? [Login](#)

[Home](#) | [Login](#) | [About](#)

2017 © Push UP.

Σενάριο 4: Έγκυρη φόρμα εγγραφής.



Σενάριο 5: Ολοκλήρωση εγγραφής.

### 3.3 Είσοδος χρήστη

LOGIN

Email

Password

☐ Remember me

Login

Do not have an account? [Sign up](#)

[Home](#) | [Sign up](#) | [About](#)  
2017 © Push UP.

Σενάριο 6: Φόρμα εισόδου.

LOGIN

Email  
tasos@pup.gr

Password  
●●●●●●●●

☐ Remember me

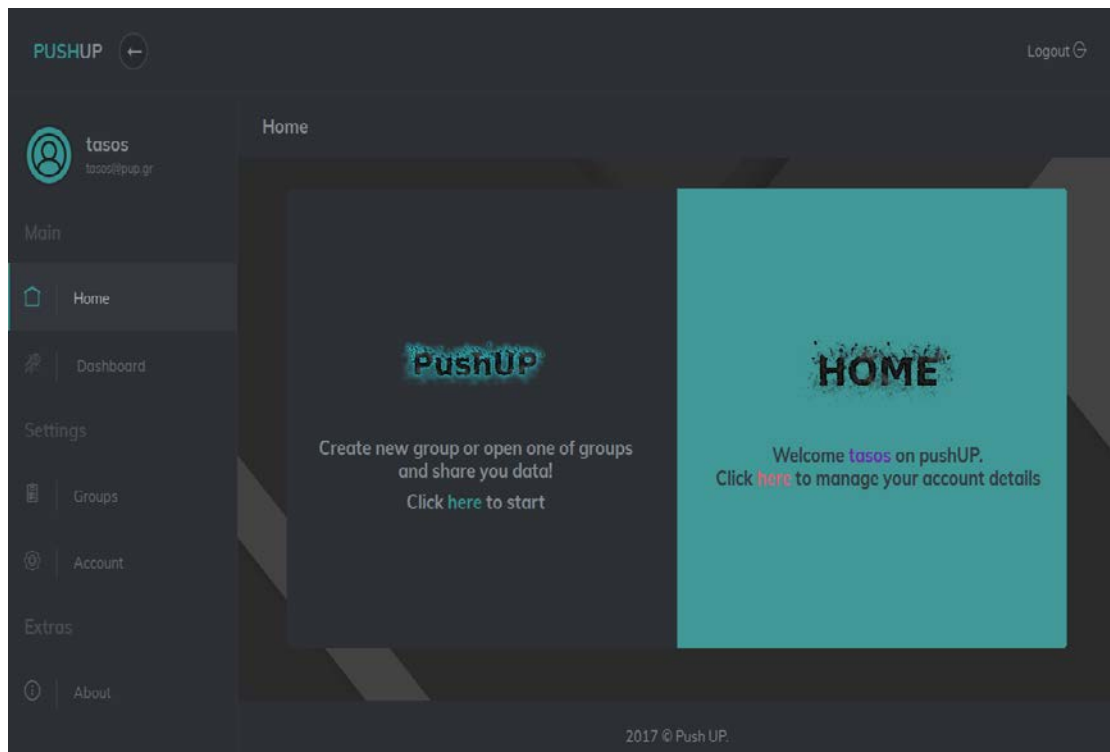
Login

Do not have an account? [Sign up](#)

Email or password do not matched.

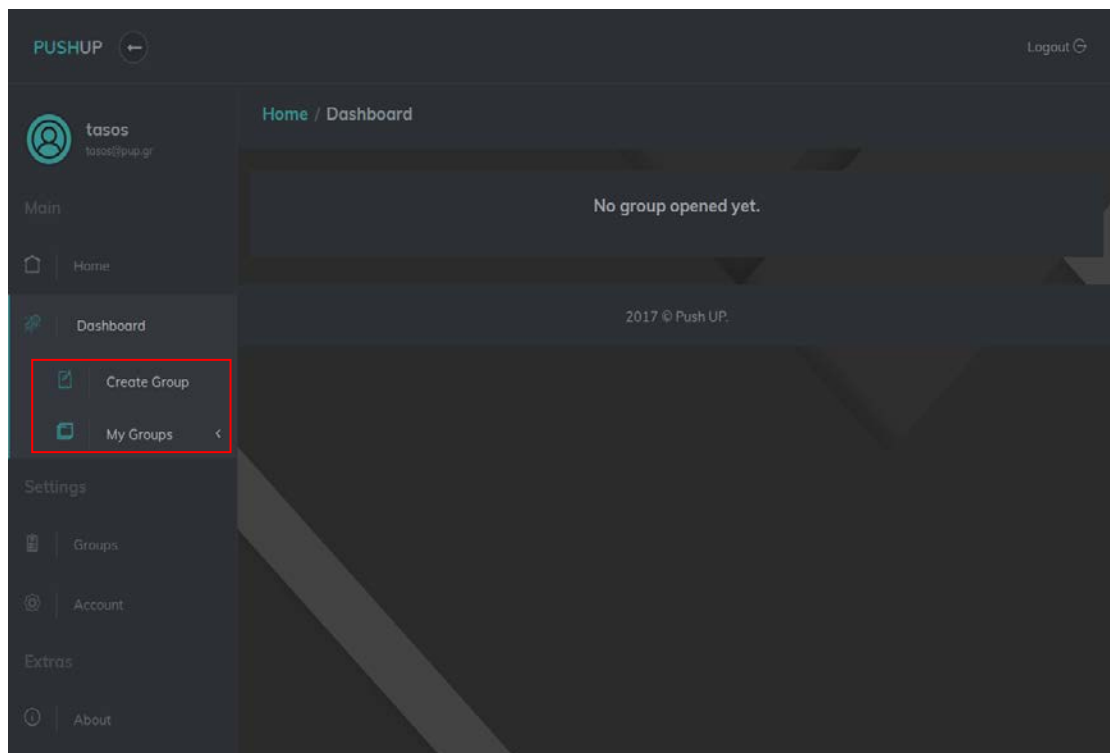
Σενάριο 7: Προσπάθεια εισόδου με μη-έγκυρα στοιχεία.

### 3.4 Λειτουργίες συνδεδεμένου χρήστη.

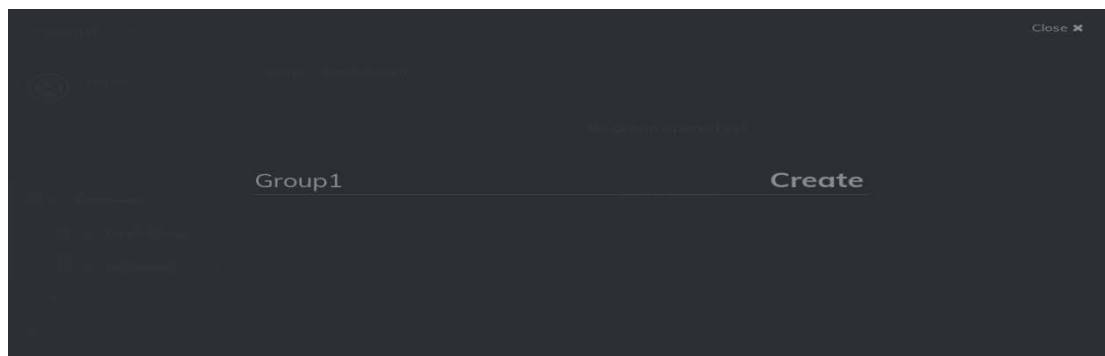


Σενάριο 8: Η αρχική σελίδα της εφαρμογής μετά από μια επιτυχημένη είσοδο.

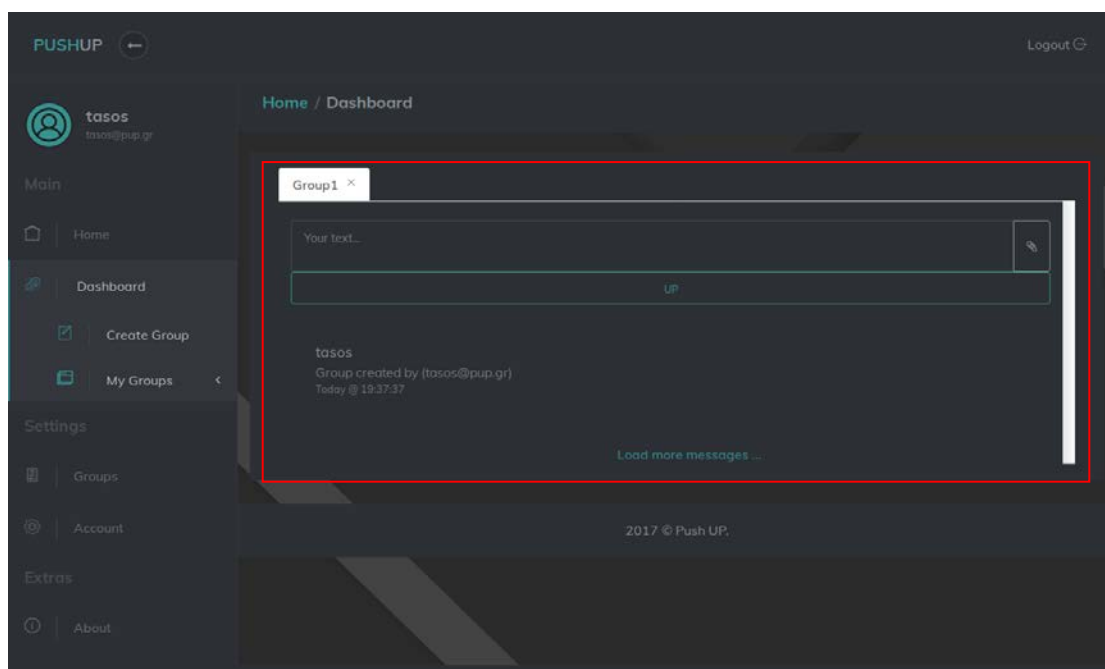
#### 3.4.1 Dashboard



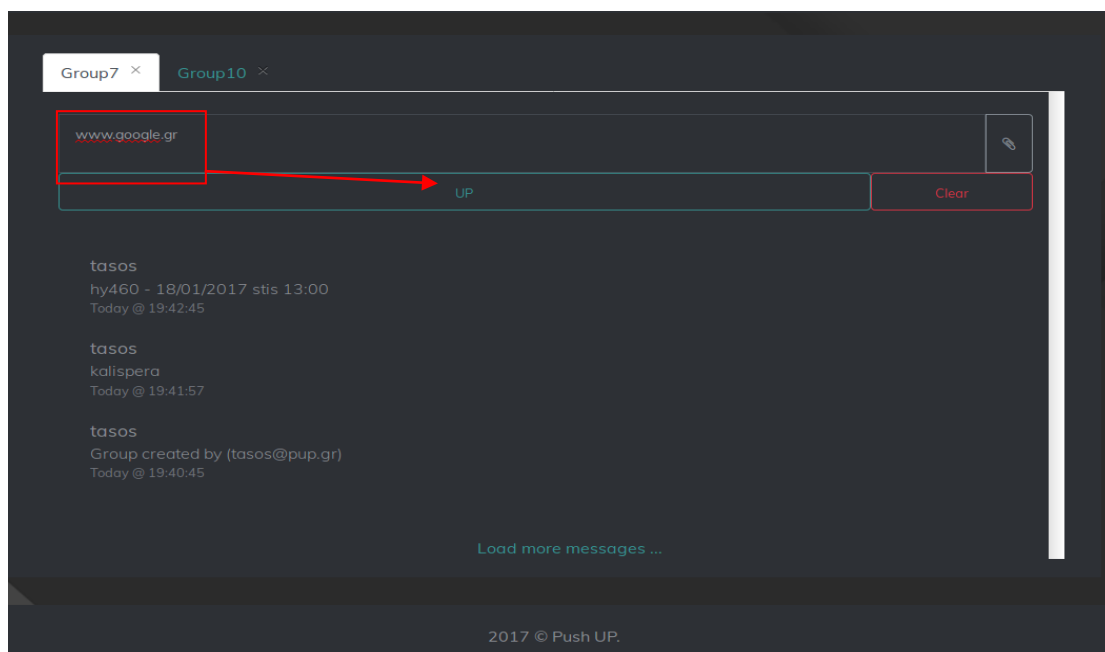
Σενάριο 9: Η βασική σελίδα για το dashboard χωρίς την ύπαρξη κάποιων ανοιχτών ομάδων. Κάτω από το dashboard υπάρχουν οι δύο επιλογές όπως περιγράφηκαν παραπάνω.



Σενάριο 10: Έχοντας πατήσει το Create Group εμφανίζεται το παραπάνω πάνελ για την πληκτρολόγηση του ονόματος της ομάδας.

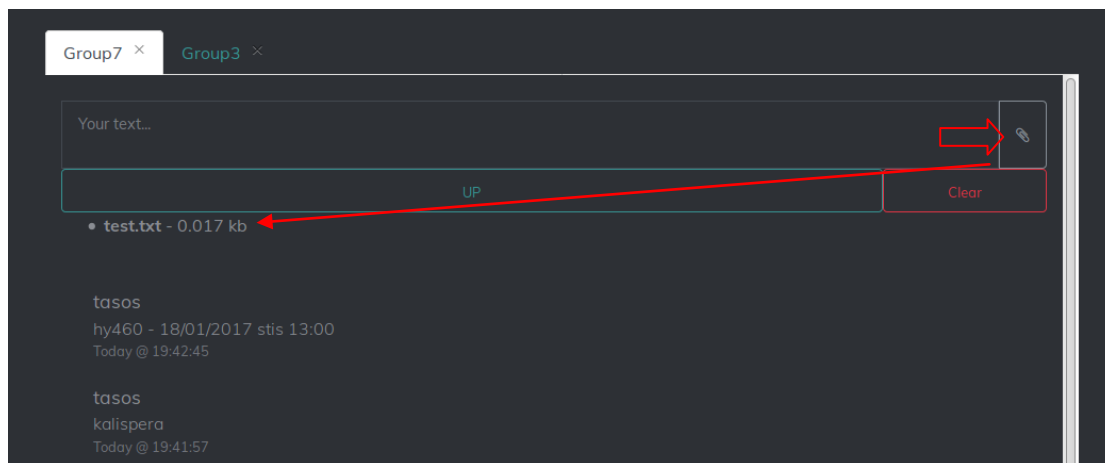


Σενάριο 11: Εμφάνιση μια νέας καρτέλας με το όνομα τις ομάδας που μόλις φτιάξαμε.

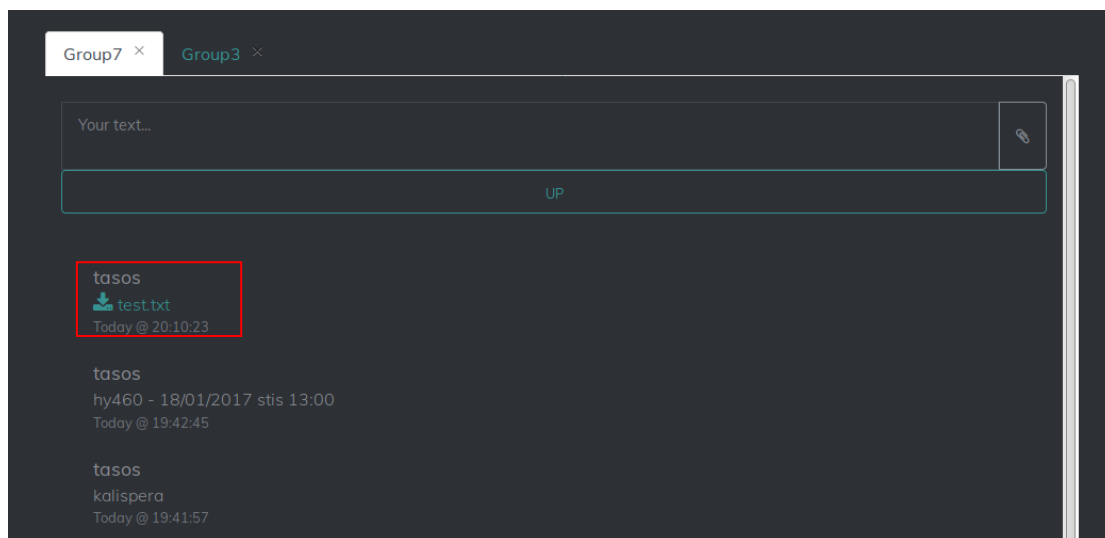


Σενάριο 12: Πληκτρολογώντας κάτι που θέλουμε να μοιράσουμε στις άλλες συσκευές μας. Για αποστολή πατάμε το UP.

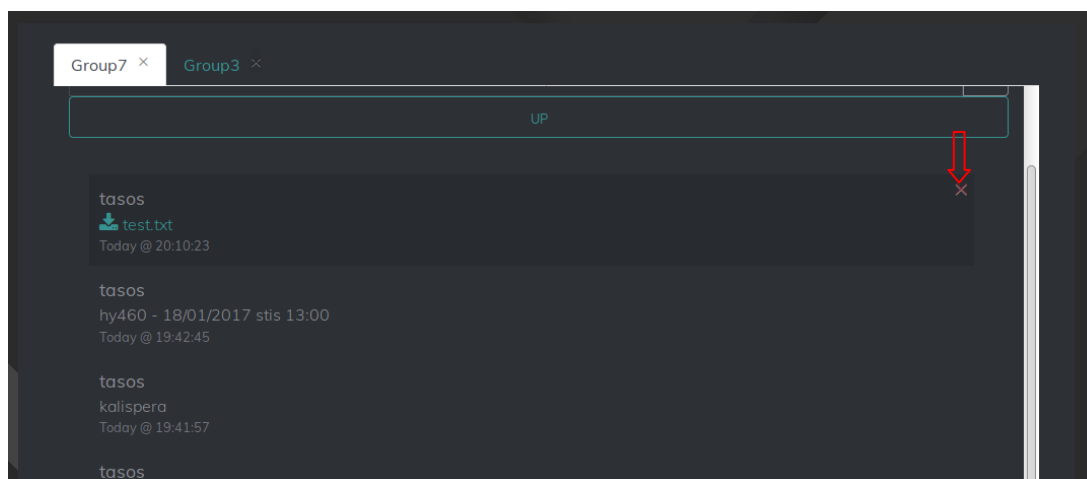
PUP – PushUP  
Data synchronizer between two or more devices.



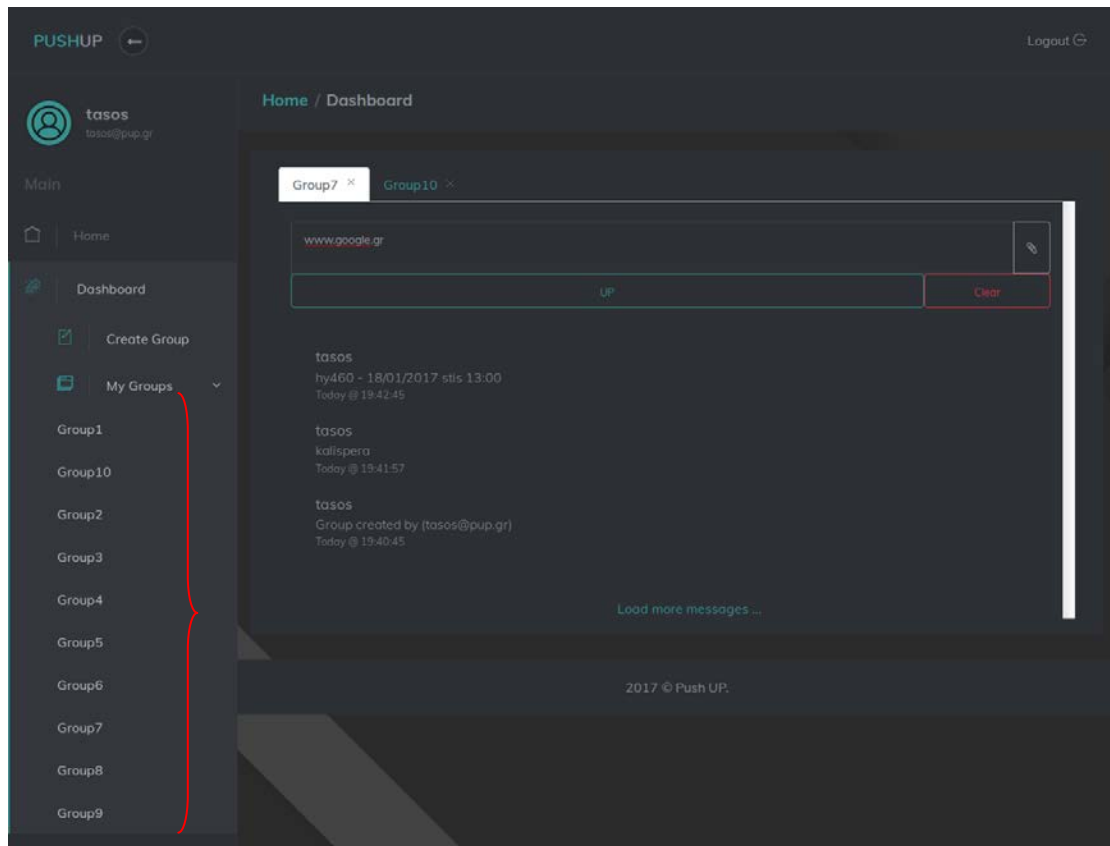
Σενάριο 13: Έχοντας επιλέξει το αρχείο test.txt προς αποστολή αφού πρώτα το επισυνάψαμε πατώντας τον σελιδοδείκτη.



Σενάριο 14: Το αρχείο μόλις έχει σταλεί και είναι έτοιμο προς λήψη κάνοντας το κλικ.

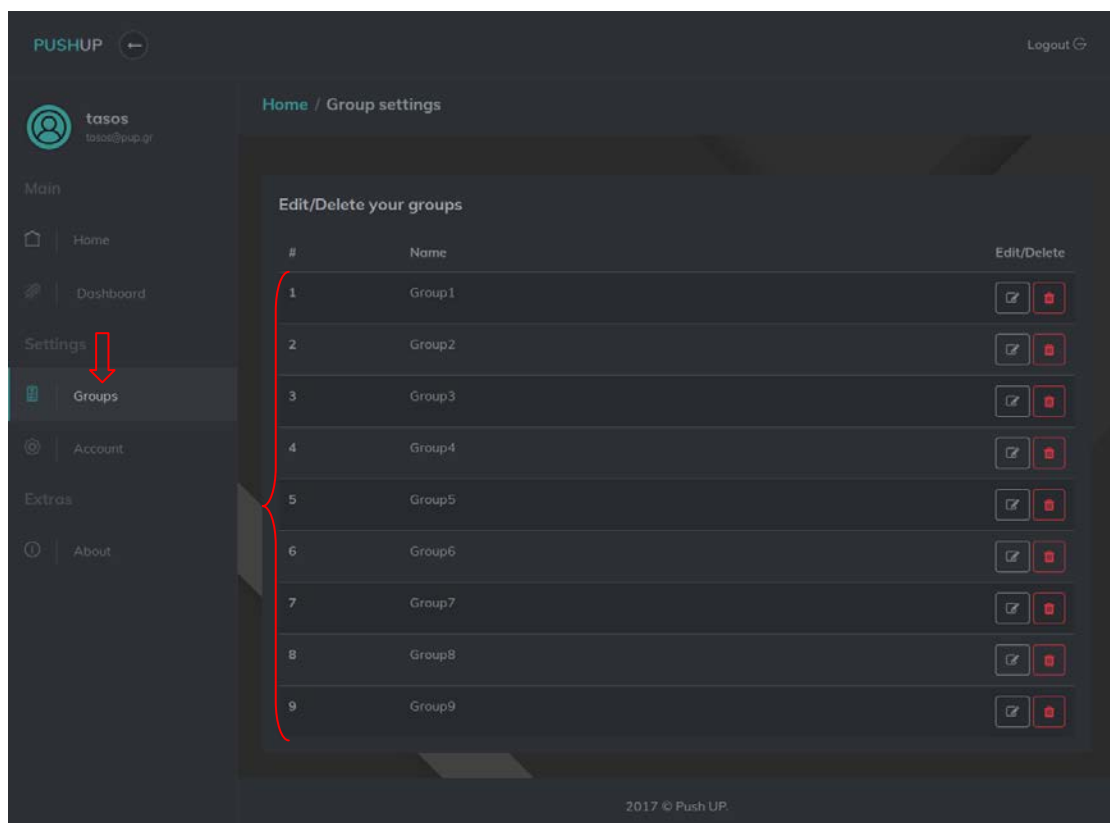


Σενάριο 15: Τοποθετώντας τον δείκτη του ποντικιού σε ένα από τα μηνύματα, εμφανίζεται στα δεξιά ένα κουμπί X με το οποίο μπορούμε να διαγράψουμε το μήνυμα.





Σενάριο 16: Πατώντας το My Groups εμφανίζεται μια λίστα με όλα μας τα groups.

### 3.4.2 Group Settings



Σενάριο 17: Πίνακας με όλες τις ομάδες όπου μπορούμε να επεξεργαστούμε καθένα από τα groups.



#	Name	Edit/Delete
1	Group_1	 

Σενάριο 18: Κάθε ομάδα στα δεξιά τις έχει τα κουμπιά edit & delete.

Edit group name.

Group1

Group\_1

Apply

Cancel

Σενάριο 19: Έχοντας πατήσει το κουμπί για επεξεργασία του ονόματος μας εμφανίζεται το παραπάνω παράθυρο.

Delete group.

Are you sure you want to delete the group?

Delete

Cancel

Σενάριο 20: Παράθυρο όπου πρέπει να επιβεβαιώσουμε την διαγραφή μιας από τις ομάδες μας.

### 3.4.3 Account settings

Home / Account settings

Edit your account details.

Login email

tasos@pup.gr

Nickname

tasos

New Nickname

Type your new nickname

Current password

Type your current password

New password

Type your new password

Confirm new password

Retype your new password

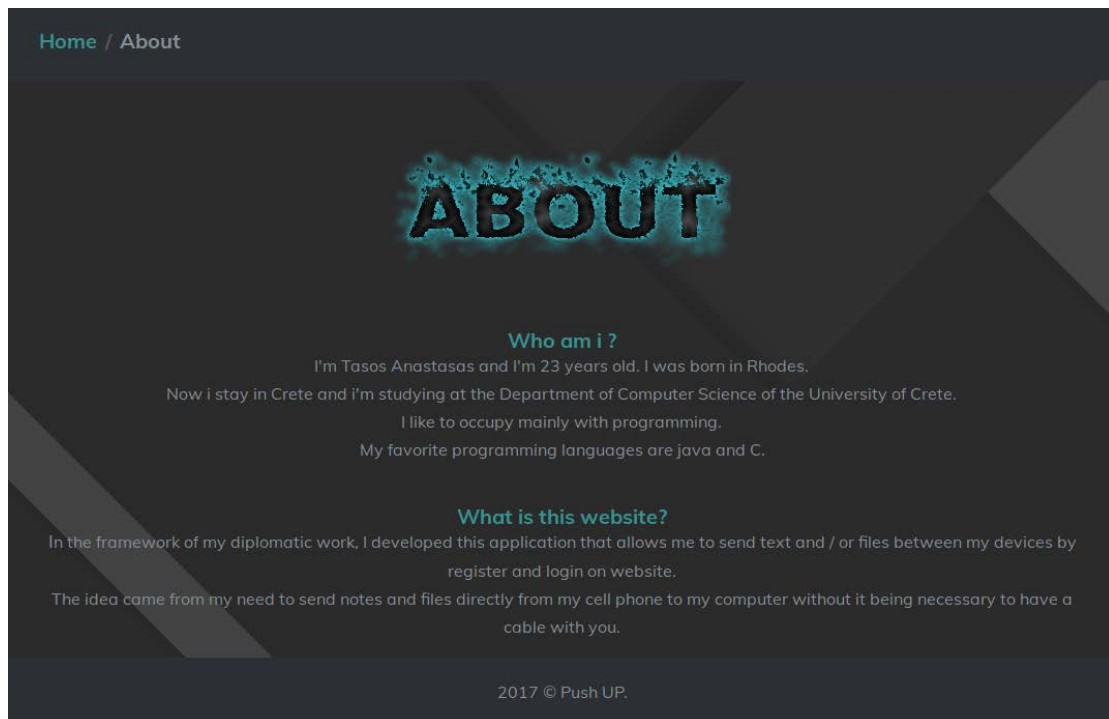
Save changes

2017 © Push UP.

Σενάριο 21: Φόρμα για την αλλαγή των στοιχείων του χρήστη.

### 3.4.4 Extras

#### 3.4.4.1 About



Σενάριο 22: Σελίδα με πληροφορίες του χρήστη και της εφαρμογής.

## 4 Αρχιτεκτονική & απαιτήσεις συστήματος

### 4.1 RethinkDB [1]

#### *Τι είναι η βάση δεδομένων RethinkDB*

---

Η RethinkDB είναι η πρώτη βάση δεδομένων JSON ανοιχτού κώδικα και είναι ιδανική για real time ιστοσελίδες. Αντιστρέφει την παραδοσιακή αρχιτεκτονική της βάσης δεδομένων, εκθέτοντας ένα συναρπαστικό μοντέλο νέας πρόσβασης – αντί για polling για αλλαγές, ο προγραμματιστής μπορεί να λαμβάνει συνεχώς ενημερωμένα αποτελέσματα των επερωτήσεων του σε πραγματικό χρόνο (live feeds). Η αρχιτεκτονική της για την συνεχή λήψη ενημερωμένων αποτελεσμάτων σε πραγματικό χρόνο μειώνει δραματικά τον χρόνο και την προσπάθεια που απαιτούνται για την ανάπτυξη επεκτάσιμων εφαρμογών πραγματικού χρόνου. Η RethinkDB είναι μια εξαιρετική επιλογή όταν οι εφαρμογές χρειάζεται να επωφεληθούν από νέα δεδομένα σε πραγματικό χρόνο.

#### *Πως χρησιμοποιήθηκε η RethinkDB*

---

Όπως αναφέρθηκε και παραπάνω, η βάση δεδομένων RethinkDB έχει την δυνατότητα να λαμβάνει και να μοιράζει οποιαδήποτε αλλαγή γίνει σε ένα πεδίο, ένα πίνακα ή σε ολόκληρη την βάση. Χάρης σε αυτήν την λειτουργία ο client μπορεί να λαμβάνει σε πραγματικό χρόνο τα νέα δεδομένα που εισήχθησαν σε έναν πίνακα. Πιο συγκεκριμένα, κατά την είσοδο του χρήστη στην εφαρμογή με την βοήθεια τεχνολογίας Socket.io (§4.4) ο server ενημερώνεται για την είσοδό του αυτή. Τότε με τις κατάλληλες επερωτήσεις παίρνει από την βάση δεδομένων τις πληροφορίες του χρήστη καθώς και τις ανοιχτές ομάδες που είχε από την τελευταία του σύνδεση. Έτσι, για κάθε μια ομάδα αρχίζει να παρακολουθεί τον αντίστοιχο πίνακα για εισαγωγή ή διαγραφή δεδομένων. Αυτό ονομάζεται Live feeds και γίνεται με την χρήση της συνάρτησης Changes() που προσφέρεται από το API της RethinkDB.

#### *Η σχεδίαση της βάσης δεδομένων.*

---

Η βάση δεδομένων αρχίζει με 2 πίνακες οι οποίοι δημιουργούνται στην εκκίνηση του server. Όμως κατά την δημιουργία μιας ομάδας αυτόματα δημιουργείται και ένας πίνακας που αντιστοιχίζεται στο αναγνωριστικό της ομάδας. Επομένως οι πίνακες της βάσης αυξομειώνονται δυναμικά με την πάροδο του χρόνου, ανάλογα των δραστηριοτήτων του εκάστοτε χρήστη. Πιο συγκεκριμένα έχουμε τους παρακάτω πίνακες:

- **Accounts**, ο οποίος αποθηκεύει τους χρήστες και τις πληροφορίες που χρειαζόμαστε για κάθε χρήστη όπως περιγράφεται παρακάτω:
  - **Email** : [string] το email του χρήστη.
  - **Nickname**: [string] το όνομα του χρήστη.

PUP – PushUP

Data synchronizer between two or more devices.

- Password: [string] ο κωδικός του χρήστη (έχει γίνει MD5 κωδικοποίηση πριν κατά την εγγραφή).
- Groups: [list] μια λίστα από τα ID των ομάδων που έχει ο χρήστης.
- OpenedGroups: [list] μια λίστα με τα ID των ανοιχτών ομάδων του χρήστη.

➤ **Groups**, ο πίνακας αυτός αποθηκεύει όλες τις ομάδες που έχουν φτιαχτεί από όλους του χρήστες. Αυτό μας επιτρέπει να μπορούμε να δημιουργούμε ένα μοναδικό ID με το οποίο αντιστοιχίζεται κάθε πίνακας. Ο πίνακας περιέχει τα πεδία:

- **ID**: [string] δημιουργείται αυτόματα και είναι μοναδικό. Αυτό το ID μετατρέπεται σε έγκυρο όνομα για πίνακα μετατρέποντας τις '-' σε '\_'.
- Name: [string] Το όνομα της ομάδας.
- User: [string] Το email του χρήστη.

➤ **Group\_id**: (το ID είναι της μορφής **12345abc\_123f\_456f\_890f\_123456789abc** και είναι ένα από τα ID του πίνακα Groups. Ένας τέτοιος πίνακας δημιουργείται κάθε φορά που γίνεται Create Group):

- **ID**: [string] το id του εκάστοτε μηνύματος.
- Data: [string] το κείμενο του μηνύματος ή το όνομα του αρχείου (εξαρτάται από τον τύπο που έχει η κάθε πλειάδα).
- Type: [string] ο τύπος του μηνύματος, εάν είναι 'text' τότε είναι απλό κείμενο αλλιώς θα είναι ο τύπος του αρχείου.
- File: [string] τα δεδομένα του αρχείου, αυτό το πεδίο υπάρχει μόνο όταν ο τύπος του μηνύματος είναι αρχείο.
- Time: [integer] η ώρα αποστολής του μηνύματος.

Η διαδικασία δημιουργίας νέου πίνακα αρχίζει με την δημιουργία νέας ομάδας από τον χρήστη ([Σενάριο 10<sup>ο</sup>](#)), στην συνέχεια γίνονται οι παρακάτω διαδικασίες:

- (1) Γίνεται εισαγωγή μιας νέας καταχώρισης στον πίνακα Groups
- (2) Παίρνοντας το ID της πλειάδας που μόλις δημιουργήθηκε, μετατρέπουμε τις παύλες '-' σε κάτω παύλες '\_' και δημιουργούμε το ID'.
- (3) Κάνουμε δημιουργία ενός πίνακα με όνομα το ID'.
- (4) Τέλος εισάγεται το ID' στις λίστες Groups και OpenedGroups του χρήστη.

Η διαδικασία διαγραφής πίνακα πυροδοτείται από την διαγραφή μιας ομάδας από τον πίνακα ρυθμίσεων ομάδων ([Σενάριο 17<sup>ο</sup>](#) & [Σενάριο 20<sup>ο</sup>](#)). Έπειτα ακολουθεί η αντίστροφη διαδικασία όπως αυτή περιγράφηκε παραπάνω με την διαφορά ότι εδώ γίνεται διαγραφή στοιχείων αντί για δημιουργία.

Στην βάση δεδομένων δημιουργείται για κάθε πίνακα ένα ευρετήριο στο πρωτεύων κλειδί. Η δημιουργία του ευρετηρίου γίνεται αυτόματα από την RethinkDB κατά την δημιουργία του πίνακα. Επιπλέον γίνεται και η δημιουργία δευτερευόντων ευρετηρίων για τους σκοπούς της εφαρμογής. Αναλυτικότερα έχουμε:

- Για τον πίνακα **'groups'** δημιουργείται το ευρετήριο με όνομα **'userAndNames'** στα πεδία **'user'** και **'names'** του πίνακα. Ο λόγος ύπαρξης του ευρετηρίου είναι η ανάγκη ανάκτησης ενός ή περισσότερων ονομάτων μιας ομάδας ενός συγκεκριμένου χρήστη. Το ευρετήριο αυτό φτιάχνεται μια φορά κατά την εκκίνηση του server για πρώτη φορά και εφόσον δεν υπάρχει ίδει.
- Για **κάθε νέα ομάδα** που φτιάχνει ο χρήστης δημιουργείται και ένας πίνακας στην βάση δεδομένων. Η ανάκτηση των μηνυμάτων από τον πίνακα γίνεται με βάση τον χρόνο που έχουν σταλθεί, γιατί γίνεται η χρήση κατάλληλων φίλτρων ώστε να εμφανίζονται αρχικά κάποια από τα μηνύματα και όχι όλα, για λόγους ταχύτητας. Επομένως υπήρχε η ανάγκη για την δημιουργία ενός δευτερεύοντος ευρετηρίου με όνομα **'time'** στο πεδίο **'time'** του εκάστοτε πίνακα.

## 4.2 Front-end

### 4.2.1 AngularJS [2]

#### *Τι είναι η AngularJS*

---

Τα αρχεία HTML είναι εξαιρετικά για την δήλωση στατικών εγγράφων, αλλά παρουσιάζουν αδυναμίες όταν προσπαθούμε να τα χρησιμοποιήσουμε για να δηλώσουμε δυναμικές προβολές ιστοσελίδων. Εκεί εμφανίζεται η AngularJS η οποία επεκτείνει το λεξιλόγιο της HTML. Έτσι δημιουργείται ένα εξαιρετικά εκφραστικό, ευανάγνωστο και γρήγορο περιβάλλον. Το σημαντικότερο πλεονέκτημα της Angular είναι η επεκτασιμότητά της εξαιτίας του modularity που προσφέρει. Αυτό είναι το βασικό συστατικό για την συντήρηση και την επέκταση μιας εφαρμογής.

#### *Πως χρησιμοποιήθηκε η AngularJS ?*

---

Το μεγαλύτερο ποσοστό του client είναι γραμμένο σε AngularJS. Υπάρχουν πολλά διαφορετικά module τα οποία όλα μαζί συνθέτουν τον πυρήνα της εφαρμογής. Η AngularJS διαθέτει πληθώρα υπηρεσιών οι οποίες κάνουν την ανάπτυξη της εφαρμογής ευκολότερη. Ένα χαρακτηριστικό service είναι το \$route, με το οποίο υπάρχει η

δυνατότητα ανάλογα με τη διεύθυνση που έχει πληκτρολογηθεί να φορτώνεται το ανάλογο HTML template με τον ανάλογο controller. Ακόμα ένα χαρακτηριστικό της Angular είναι τα directive τα οποία είναι οδηγίες σε ένα στοιχείο DOM. Χαρακτηριστικά directive είναι ngView, το οποίο συνδυάζεται με το \$route και κάθε φορά που φορτώνεται ένα νέο template λόγω του route αυτό τοποθετείται στο στοιχείο που περιέχει αυτό το directive. Επίσης σημαντικό directive είναι και το ngModel, το οποίο προσφέρει την δυνατότητα να γίνονται binding τιμές μεταξύ HTML και controller.

Η δομή του client είναι:

```
- app
----- common
----- filters
----- services
----- components
----- about
----- login
----- signup
----- welcome
----- home
----- about
----- common
----- dashboard
----- settings
----- account
----- groups
```

#### 4.2.2 Bootstrap 4 [3]

Για την δημιουργία της διεπαφής έγινε η χρήση του bootstrap twitter framework. Το Bootstrap είναι ένα δωρεάν και open-source front-end framework για το σχεδιασμό ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει HTML και CSS με βάση πρότυπα σχεδίασης για την τυπογραφία, τις μορφές, τα κουμπιά πλοήγησης και άλλα στοιχεία του περιβάλλοντος, καθώς και προαιρετικά τη Javascript για επεκτάσεις και για δημιουργία των αναδυόμενων παραθύρων τα οποία παρουσιάζονται στην web εφαρμογή. Υποστηρίζει και είναι συμβατή με τις τελευταίες εκδόσεις των Google Chrome, Firefox, Internet Explorer, Opera και Safari browsers.

#### 4.2.3 Επικοινωνία Client-Server

Προκειμένου να λειτουργήσει η εφαρμογή για πολλούς και διαφορετικούς χρήστες είναι αναγκαίος ένας server όπου θα εξυπηρετεί κατάλληλα τα αιτήματα του εκάστοτε χρήστη. Επομένως, η επικοινωνία των δύο πλευρών γίνεται με το παρακάτω API, όπου η εκάστοτε συνάρτηση δέχεται κάποια ορίσματα, έπειτα κάνει κατάλληλο HTTP request στον server και επιστρέφει την απάντηση που έλαβε από αυτόν. Το API βρίσκεται στο path: «*diplomaticProject / src / client / app / common / services / http.service.js*»

Το AngularJS module (http.service.js) προσφέρει τις εξής συναρτήσεις:

PUP – PushUP

Data synchronizer between two or more devices.

- **accountGetUserInfo:** ζητάει από τον server τις πληροφορίες ενός χρήστη. Λεπτομέρειες συνάρτησης:
  - Method: GET
  - Url: `/account/info`
  - Params: το email του χρήστη
  
- **accountCreate:** ζητάει από τον server να δημιουργήσει έναν νέο λογαριασμό χρήστη. Λεπτομέρειες συνάρτησης:
  - Method: POST
  - Url: `/account/create`
  - Data: JSON object το οποίο περιέχει τα πεδία:
    - uNickname – το όνομα του χρήστη.
    - uEmail – το email του χρήστη.
    - uPassword – ο κωδικός του χρήστη.
  
- **accountAuthenticate:** ζητάει επιβεβαίωση στοιχείων του χρήστη, επιπλέον δημιουργείται και έναν cookie στην περίπτωση που τα στοιχεία είναι σωστά. Λεπτομέρειες συνάρτησης:
  - Method: GET
  - Url: `/account/authenticate`
  - Params: JSON object με τα πεδία:
    - uEmail: το email του χρήστη
    - uPassword: ο κωδικός του χρήστη
    - rememberMe: ένα flag το οποίο καθορίζει την διάρκεια του cookie, μια εβδομάδα όταν είναι true , αλλιώς μέχρι να κλείσει ο browser.
  
- **accountUpdateNickname:** ζητάει από τον server να αλλάξει το όνομα του χρήστη. Λεπτομέρειες συνάρτησης:
  - Method: POST
  - Url: `/account/update/nickname`
  - Data: JSON object με τα πεδία:
    - curPassword: ο ισχύον κωδικός του χρήστη
    - nickname: το νέο όνομα του χρήστη
  
- **accountUpdatePassword:** ζητάει από τον server να αλλάξει τον κωδικό του χρήστη. Λεπτομέρειες συνάρτησης:
  - Method: POST
  - Url: `/account/update/password`
  - Data: JSON object με τα πεδία:
    - curPassword: ο ισχύον κωδικός του χρήστη
    - password: ο νέος κωδικός του χρήστη

- **accountUpdateAll:** ζητάει από τον server να αλλάξει και τον κωδικό και το όνομα του χρήστη. Λεπτομέρειες συνάρτησης:
  - Method: POST
  - Url: '/account/update/all'
  - Data: JSON object με τα πεδία:
    - curPassword: ο ισχύον κωδικός του χρήστη
    - nickname: το νέο όνομα του χρήστη
    - password: ο νέος κωδικός του χρήστη
- **groupAddData:** ζητάει από τον server να εισάγει νέα δεδομένα για μια ομάδα
  - Method: POST
  - Url: '/group/add/data'
  - Data: JSON object με τα πεδία:
    - gID: το ID της ομάδας που θέλουμε να εισάγουμε τα δεδομένα
    - value: η τιμή των δεδομένων
    - time: η ώρα που έγινε η αποστολή του μηνύματος
    - type: ο τύπος του μηνύματος
    - file \*: τα δεδομένα του αρχείου, το πεδίο υπάρχει όταν ο τύπος δεν είναι text
- **groupRetrieveData:** ζητάει από τον server τα μηνύματα από ένα group
  - Method: GET
  - Url: '/group/retrieve/data'
  - Params: JSON object με τα πεδία
    - gID: το ID του group που θέλουμε να πάρουμε τα δεδομένα
    - afterFrom: ένα timestamp το οποίο καθορίζει από πιο χρονικό διάστημα και μετά θα επιστρέψουμε δεδομένα
    - limitVal: περιορισμός του πλήθους των επιστρεφόμενων μηνυμάτων
- **groupFileValue:** ζητάει από τον server το value ενός αρχείου
  - Method: GET
  - Url: '/group/retrieve/data'
  - Params: JSON object με τα πεδία
    - gID: το ID της ομάδας
    - mID: το ID του μηνύματος
- **groupRetrieveName:** ζητάμε από τον server το όνομα ενός group
  - Method: GET
  - Url: '/group/retrieve/name'
  - Params: JSON object με τα πεδία
    - gID: το id της ομάδας

PUP – PushUP

Data synchronizer between two or more devices.



- **groupCreate:** ζητάμε από τον server να φτιάξει μια νέα ομάδα
  - Method: POST
  - Url: '/group/create'
  - Data: JSON object με τα πεδία:
    - gName: το όνομα της ομάδας
- **groupDelete:** ζητάμε από τον server να διαγράψει μια ομάδα
  - Method: GET
  - Url: '/group/delete'
  - Params: JSON object με τα πεδία:
    - gID: το αναγνωριστικό της ομάδας
    - gName: το όνομα της ομάδας
- **groupUpdateName:** ζητάμε από τον server να ενημερώσει το όνομα μιας ομάδας
  - Method: POST
  - Url: '/group/update/name'
  - Data: JSON object με τα πεδία:
    - gID: το id της ομάδας
    - gName: το νέο όνομα της ομάδας
- **groupInsertToOpenedList:** ζητάμε από τον server να εισάγει το group στην λίστα με τα ενεργά groups
  - Method: POST
  - Url: '/group/openedList/insert'
  - Data: JSON object με τα πεδία:
    - gID: το id της ομάδας
- **groupRemoveFromOpenedList:** ζητάμε από τον server να αφαιρέσει την ομάδα από την λίστα των ενεργών ομάδων
  - Method: POST
  - Url: '/group/openedList/remove'
  - Data: JSON object με τα πεδία:
    - gID: το αναγνωριστικό της ομάδας
- **groupDeleteMessage:** ζητάμε από τον server να διαγράψει ένα μήνυμα
  - Method: POST
  - Url: '/group/delete/message'
  - Data: JSON object με τα πεδία:
    - gID: το αναγνωριστικό της ομάδας
    - mID: το αναγνωριστικό του μηνύματος.

## 4.3 Back-end

### 4.3.1 NodeJS [4]

#### *Τι είναι το Node JS*

---

Το NodeJS είναι ένα Javascript runtime built επάνω στο Chrome's V8 JavaScript engine. Το Node χρησιμοποιεί event-driven, non-blocking I/O μοντέλα τα οποία το καθιστούν ελαφρύ και αποτελεσματικό. Είναι σχεδιασμένο για την δημιουργία επεκτάσιμων εφαρμογών δικτύου. Σε κάθε σύνδεση ενεργοποιείται μία συνάρτηση, αλλά εάν δεν υπάρχει εργασία, το Node θα αδρανοποιηθεί (κοιμηθεί). Αυτό έρχεται σε αντίθεση με το πιο συνηθισμένο μοντέλο συγχρονισμού σήμερα, όπου χρησιμοποιούνται τα νήματα του λειτουργικού. Επιπλέον οι χρήστες του Node δεν χρειάζεται να ανησυχούν για dead-locks της διαδικασίας, καθώς δεν υπάρχουν locks. Σχεδόν καμία συνάρτηση στο Node δεν εκτελεί απευθείας I/O, οπότε η διαδικασία ποτέ δεν μπλοκάρει. Επειδή τίποτα δεν μπλοκάρει, τα επεκτάσιμα συστήματα είναι πολύ λογικό να αναπτυχθούν στο Node.

#### *Πως χρησιμοποιήθηκε το NodeJS?*

---

Στο Node είναι βασισμένο όλο το κομμάτι του back-end. Ο server που εξυπηρετεί όλα τα αιτήματα των χρηστών είναι υλοποιημένο σε Javascript και βασίζεται στις τεχνολογίες που προσφέρει το Node. Κάποιες από τις τεχνολογίες του Node που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής είναι:

- **Express:** στο οποίο μπορούμε να ορίσουμε διάφορα routes ανάλογα με την διεύθυνση και το είδος των requests. Επίσης η χρήση των επόμενων τεχνολογιών έγινε μέσω του express.
- **Cookie-parser:** το οποίο κάνει parse τα cookies από κάθε request και έτσι μπορούμε ανά πάσα στιγμή να τα εξάγουμε για έλεγχους ασφαλείας.
- **Body-parser:** το οποίο κάνει parse το κύριο μέρος του μηνύματος και το μετατρέπει σε JSON object. Επίσης μπορούμε να ορίσουμε μέγιστο μέγεθος του εκάστοτε request, κάτι που έχει ως αποτέλεσμα ο χρήστης να μπορεί να στέλνει αρχεία συνολικού μεγέθους το πολύ 50mb.
- **HTTP:** με το οποίο μπορούμε να φτιάξουμε έναν HTTP server σε όποια IP,port εμείς θέλουμε.

#### 4.3.2 Security [5], [6]

##### 4.3.2.1 CSRF ή XSRF (Cross-site request forgery) [5]

Είναι ένας τύπος κακόβουλης εκμετάλλευσης ενός ιστότοπου όπου μεταδίδονται μη εξουσιοδοτημένες εντολές από έναν χρήστη που εμπιστεύεται η εφαρμογή. Υπάρχουν πολλοί τρόποι με τους οποίους ένας κακόβουλος ιστότοπος μπορεί να μεταδώσει τέτοιες εντολές, όπως ειδικά επεξεργασμένες ετικέτες εικόνας, κρυφές φόρμες και Javascript XMLHttpRequests, που για παράδειγμα, μπορούν να λειτουργήσουν χωρίς την αλληλεπίδραση του χρήστη ή ακόμα και με την γνώση του. Σε αντίθεση με το cross-site scripting (XSS), το οποίο εκμεταλλεύεται την εμπιστοσύνη που έχει ένας χρήστης για έναν συγκεκριμένο ιστότοπο, η CSRF εκμεταλλεύεται την εμπιστοσύνη που έχει ένας ιστότοπος στο πρόγραμμα περιήγησης ενός χρήστη.

Δεδομένων των παραπάνω έχει φτιαχτεί ένα middleware module το οποίο ελέγχει το request που ο server έλαβε. Εάν περάσει τον έλεγχο, τότε το request συνεχίζει την πορεία του αλλιώς ανακόπτεται και επιστρέφεται μήνυμα λάθους (error : 400) .

##### 4.3.2.2 JSON Vulnerability [6]

Μια ευπάθεια για JSON επιτρέπει σε third-party ιστοσελίδες να μετατρέψει τη διεύθυνση πόρων του JSON σε αίτηση JSONP υπό ορισμένες προϋποθέσεις. Για να αντιμετωπιστεί αυτό, ο server πρέπει να προσθέτει σε όλες τις αιτήσεις JSON την ακόλουθη συμβολοσειρά «}}}}, \n». Η AngularJS θα καταργήσει αυτό το πρόβλημα αυτόματα πριν το επεξεργαστεί ως JSON.

Για την αντιμετώπιση της ευπάθειας αυτής έχει δημιουργηθεί το κατάλληλο middleware που προσθέτει στο μήνυμα την παραπάνω συμβολοσειρά.

##### 4.3.2.3 Encryption

Έχει υλοποιηθεί ένα σύστημα κρυπτογράφησης και αποκρυπτογράφησης με σκοπό να γίνεται κρυπτογράφηση του cookie με τέτοιο τρόπο, ώστε να είναι αρκετά δύσκολη η εξαγωγή των πληροφοριών του χρήστη που αποθηκεύονται σε αυτό.

#### 4.4 Socket.IO [7]

##### *Τι είναι το Socket.io*

---

Το Socket.io επιτρέπει την αμφίδρομη επικοινωνία σε πραγματικό χρόνο η οποία είναι βασισμένη σε γεγονότα. Λειτουργεί σε κάθε πλατφόρμα, πρόγραμμα περιήγησης ή συσκευή, εστιάζοντας εξίσου στην αξιοπιστία και την ταχύτητα.

Το Socket.io χρησιμοποιήθηκε και από τις δύο πλευρές (client και server). Αρχικά ένας client μόλις συνδεθεί κάνει σύνδεση στο socket που αυτό έχει ως αποτέλεσμα ο server να εξάγει από το cookie του request που έλαβε μέσω του socket τις πληροφορίες του χρήστη και αφού πάρει από την βάση τις απαραίτητες πληροφορίες για τις ανοιχτές ομάδες καλεί τις συναρτήσεις που χρειαζόμαστε για την υλοποίηση του live feeds.

Για την αποστολή ενός γεγονότος μέσω του socket γίνεται η χρήση του “emit” πχ socket.emit(<το όνομα του event>, <δεδομένα>), ενώ για την λήψη κάποιο γεγονότος χρησιμοποιείται η εντολή “on” πχ socket.on(<το όνομα του event>, <η συνάρτηση που θα καλεστεί όταν λάβει κάτι>).

Ο **client** μπορεί να στείλει τα παρακάτω events:

- **openGroup**: αυτό σηματοδοτεί το άνοιγμα της ομάδας, επομένως ο server θα πρέπει να καλέσει την κατάλληλη συνάρτηση που θα παρακολουθεί την ομάδα για τυχόν προσθήκη ή διαγραφή μηνυμάτων.
- **closeGroup**: μόλις λάβει αυτό το event ο server θα πρέπει να σταματήσει να κοιτάει για αλλαγές στα μηνύματα της ομάδας. Αυτό γίνεται με το κλείσιμο της σύνδεση που δημιουργήθηκε κατά το event: openGroup.
- **deleteGroup**: θα πρέπει ο server να κλείσει όλα τα connection που αφορούσαν την ομάδα που μόλις διαγράφηκε.
- **disconnect** ή **logout**: ο χρήστης είτε έκανε logout ή το socket αποσυνδέθηκε εξαιτίας αδρανοποίησης (π.χ. όταν είναι συνδεδεμένος από το κινητό και κλειδωθεί το κινητό τότε γίνεται disconnect). Ο server είναι υποχρεωμένος να κλείσει όλες τις ενεργές συνδέσεις που έχει με την βάση.

Ο **server** μπορεί και αυτός με την σειρά του να στείλει τα δικά του events τα οποία πυροδοτούνται από το query result που επιστρέφει η συνάρτηση changes() που προσφέρει η RethinkDB για τα live feeds:

- **groupCreate**: μια νέα ομάδα μόλις προστέθηκε πιθανότατα από άλλη συσκευή, ο server ενημερώνει τον client ο οποίος εκτελεί τις κατάλληλες ενέργειες ώστε να ενημερώσει τον χρήστη αλλά και την σελίδα για την προσθήκη της ομάδας.
- **groupDelete**: μια ομάδα διαγράφηκε και θα πρέπει να την σβήσει ο client από τα δεδομένα του καθώς και να ενημερώσει τον χρήστη για την αλλαγή.
- **groupDataAdd**: ο server αναγνωρίζει μια εισαγωγή μηνύματος σε έναν πίνακα και ενημερώνει κατάλληλα τον client. Ο client με την σειρά του όταν λάβει αυτό το γεγονός θα εισάγει το νέο μήνυμα στο κατάλληλο group και θα το κάνει render στην σελίδα.

- **groupDataRemove:** όμοια με το 'groupDataAdd' μόνο που αντί για εισαγωγή θα γίνεται διαγραφή μηνύματος.
- **groupNameChange:** ο server αναγνωρίζει μια αλλαγή ονόματος μιας ομάδας και στέλνει αυτήν την αλλαγή στον client, ο οποίος αντικαθιστά το παλιό όνομα με το νέο και κάνει τις απαραίτητες αλλαγές στην εμφάνιση της σελίδας.
- **accountNameChange:** ο server έλαβε μια αλλαγή (από την συνάρτηση changes()) για το όνομα του χρήστη και ενημερώνει τον client και αυτός με την σειρά του ενημερώνει το κατάλληλο πεδίο και τον χρήστη.
- **accountPasswordChange:** ο server ενημερώνει τον client ότι άλλαξε ο κωδικός του χρήστη και ο client αποσυνδέει το χρήστη ο οποίος πρέπει να συνδεθεί και πάλι με τον νέο κωδικό για να μπορέσει να κάνει χρήση της εφαρμογής.

## 4.5 Απαιτήσεις συστήματος

Για να έχει πρόσβαση ένας χρήστης στην εφαρμογή χρειάζεται πρόσβαση στο ίντερνετ και έναν browser, κατά προτίμηση Google chrome, Mozilla firefox ή Safari.

### 4.5.1 Προετοιμασία του περιβάλλοντος

Το ιδανικότερο περιβάλλον για να εκτελέσουμε την βάση δεδομένων και τον server είναι κάποια διανομή Linux βασισμένη σε Ubuntu.

- Για την βάση δεδομένων θα πρέπει να κάνουμε εγκατάσταση την RethinkDB <sup>[11]</sup>. Εγκατάσταση της βάσης μέσω command line εκτελώντας τις παρακάτω εντολές:
  - `source /etc/lsb-release && echo "deb http://download.rethinkdb.com/apt $DISTRIB_CODENAME main" | sudo tee /etc/apt/sources.list.d/rethinkdb.list`
  - `wget -qO- https://download.rethinkdb.com/apt/pubkey.gpg | sudo apt-key add -`
  - `sudo apt-get update`
  - `sudo apt-get install rethinkdb`
- Για τον server θα πρέπει να εγκαταστήσουμε το nodeJS <sup>[12]</sup>. Παρακάτω περιγράφονται οι εντολές που χρειάζονται για την εγκατάσταση του:

PUP – PushUP

Data synchronizer between two or more devices.

- `curl -sL https://deb.nodesource.com/setup_9.x | sudo -E bash -`
  - `sudo apt-get install -y nodejs`
  - `sudo apt-get install -y build-essential`
- Ακόμα θα χρειαστεί να εγκαταστήσουμε κάποια dependencies που υπάρχουν για τον client και τον server.
- Ανοίγουμε ένα terminal και αφού έχουμε μεταβεί στον φάκελο server που βρίσκεται στο path «**src/server**» τότε πρέπει να εκτελέσουμε την παρακάτω εντολή:
    - `npm install`
 Η οποία θα κατεβάσει όλες τις απαραίτητες βιβλιοθήκες που χρησιμοποιήθηκαν για την υλοποίηση του server.
  - Όμοια θα κάνουμε και για τον client αφού μεταφερθούμε στον φάκελο assets που βρίσκεται στο path «**src/client/assets**». Έπειτα εκτελούμε ξανά την εντολή:
    - `npm install`

#### 4.5.2 Εκκίνηση συστήματος

Αφού έχει γίνει η εγκατάσταση των άνωθεν επιτυχώς είμαστε σε θέση να εκτελέσουμε το σύστημα κάνοντας τα παρακάτω:

- Αρχικά τρέχουμε την βάση δεδομένων ανοίγοντας ένα terminal και πληκτρολογώντας την εντολή :
  - `rethinkdb`
- Στην συνέχεια για να εκτελέσουμε τον server ανοίγουμε άλλο terminal και αφού μεταβούμε στον βασικό φάκελο τις εφαρμογής πρέπει να μεταβούμε στον υπο-φάκελο server και να ξεκινήσουμε την εφαρμογή πληκτρολογώντας:
  - `cd src/server`
  - `npm start`

Ανοίγοντας έναν browser και πληκτρολογώντας: **http://localhost:3000** έχουμε πρόσβαση στην εφαρμογή.

## 5 Επίλογος

### 5.1 Συμπεράσματα

Κατά τη διάρκεια του σχεδιασμού και της υλοποίησης εξήχθησαν τα παρακάτω συμπεράσματα.

#### 5.1.1 Σχεδιασμός

Μία εφαρμογή ανεξαρτήτως μεγέθους σκοπού και λειτουργικότητας απαιτεί καλά ορισμένο σχεδιασμό και ανάλυση πριν να αρχίσει η υλοποίηση της. Μια λεπτομερή ανάλυση απαιτήσεων και σκοπού σε συνδυασμό με μια αποδοτική αρχιτεκτονική που κάνει χρήση των κατάλληλων σχεδιαστικών προοπτικών μπορεί να συρρικνώσει δραματικά τον χρόνο υλοποίησης της. Ο προγραμματιστής δεν αναγκάζεται να επιλύσει σχεδιαστικά προβλήματα on-the-fly και αποφεύγεται ο πονοκέφαλος των διαρκών refactoring στον υπάρχον κώδικα.

Συγκεκριμένα στο PushUP παρατηρήθηκε το εξής φαινόμενο. Αν και η σχεδίαση του συστήματος βοήθησε αρκετά στην γρηγορότερη παραγωγή καλύτερης ποιότητας κώδικα κατά τα την διάρκεια του σταδίου τις υλοποίησης έπρεπε να υπάρξει μια άλφα τριβή με τις εντελώς νέες και άγνωστες σε εμένα τεχνολογίες που χρησιμοποιήθηκαν. Αυτό είχε ως αποτέλεσμα να αφιερωθεί αρκετός χρόνος στην εγκατάσταση, μελέτη και δοκιμή τόσο του Nodejs, της AngularJS και αργότερα του Socket.IO. Αυτό σε συνδυασμό με την έλλειψη χρόνου είχε ως αποτέλεσμα να μην έχει αναπτυχθεί πλήρως η εφαρμογή και να υπάρχουν κάποια σημεία βελτιώσεις και επέκτασης. Μεγάλο ρόλο στην γρηγορότερη ανάπτυξη και στο refactoring έπαιξε η χρήση ενός git repository, με το οποίο μπορούσε να υπάρχει ένας έλεγχος των αλλαγών, ανίχνευση και επίλυση bugs και δημιουργία διαφορετικών branches για την δημιουργία νέων λειτουργιών χωρίς να χάνεται ο κώδικας που εκτελούταν σωστά.

#### 5.1.2 Υλοποίηση

##### 5.1.2.1 Web application

Η επιλογή χρήσης του web application αντί κάποιου άλλου είδους εφαρμογής έγινε εξαιτίας του εύρους των διαφορετικών συσκευών που μπορούν να έχουν πρόσβαση σε μια ιστοσελίδα. Εάν είχαμε επιλέξει την ανάπτυξη μιας εφαρμογής android ή IOS τότε θα περιοριζόμασταν μόνο στην εκτέλεση σε ηλεκτρονικές συσκευές τσέπης (κινητά). Σκοπός της εφαρμογής είναι να μπορεί να συγχρονίσει δεδομένα μεταξύ διαφορετικών συσκευών και ποσό μάλλον εάν αυτές οι συσκευές είναι εντελώς διαφορετικού είδους, καθώς η αποστολή δεδομένων μεταξύ υπολογιστή – κινητό δεν είναι τόσο απλή όσο είναι μέσω του PushUP.

### 5.1.2.2 NodeJS – AngularJS - RethinkDB

Η επιλογή χρήσης της Angular αντί της παραδοσιακής jQuery & Javascript μπορεί αρχικά να μας κόστισε σε χρόνος μέχρι να γίνει κατανοητός ο τρόπος λειτουργίας της αλλά και οι διαφορετικές λειτουργίες που προσέφερε, ωστόσο αργότερα κερδίστηκε περισσότερος χρόνος. Αυτό οφείλετε στο γεγονός ότι υπήρξε η δυνατότητα να φτιαχτούν διαφορετικά module ανάλογα τις ανάγκες που υπήρχαν κάθε φορά, έτσι έγινε πιο επεκτάσιμος και συντηρήσιμος ο κώδικας, μειώνοντας δραματικά τον χρόνο του refactoring. Ακόμα μπόρεσα και χρησιμοποίησα το service \$routes που προσφέρει η angular που μου επέτρεψε να φτιάξω πολλαπλά templates και controllers για κάθε διαφορετικό component της εφαρμογής, κάνοντας έτσι την HTML σελίδα πιο ξεκάθαρη και ευανάγνωστη. Σαφώς και οι υπόλοιπες ανέσεις που προσφέρει για την επεξεργασία μιας φόρμας και την επαναληπτική διαδικασία για την εισαγωγή κώδικα HTML έπαιξαν καθοριστικό ρόλο για την ολοκλήρωση του συστήματος.

Για την υλοποίηση του server έγινε η επιλογή του node αντί των κλασικών servlet, JSP, apache κλπ τα οποία ήταν ίδωι γνωστά για εμένα. Η επιλογή έγινε με κριτήριο τις δυνατότητες σε συνδυασμό με την απλότητα της υλοποίησης. Στο node υπήρχαν πολλοί έτοιμοι αυτοματισμοί που σου επέτρεπαν να κάνεις αυτό που ήθελες άμεσα. Έτσι μπόρεσα να εστιάσω σε άλλα θέματα όπως η ασφάλεια. Φυσικά σημαντικό παράγοντα στην επιλογή του ήταν και η γλώσσα ανάπτυξης του που είναι από τις γνωστότερες και αρκετά απλή στην χρήση, αλλά και η ανάγκη για εκμάθηση κάτι διαφορετικού στα πλαίσια της διπλωματικής μου εργασίας.

Τέλος κάνοντας χρήση του RethinkDB κατάλαβα ότι έχει αρκετές περισσότερες δυνατότητες από αυτές που αξιοποιήθηκαν για την ανάπτυξη της εφαρμογής. Προφανώς η χρήση της συγκεκριμένης βάσης έπαιξε καθοριστικό ρόλο στην επιλογή και ανάπτυξη αυτού του είδους της εφαρμογής, εξαιτίας της τρομερής δυνατότητας που προσέφερε στο να μπορείς να λαμβάνεις ειδοποιήσεις για δεδομένα που αλλάζουν real-time σε έναν πίνακα.

### 5.1.2.3 Επίτευξη σκοπού

Σύμφωνα με την αρχική ιδέα που υπήρξε για το είδος και τις δυνατότητες της εφαρμογής, μπορούμε να πούμε ότι έχει δημιουργηθεί κατά 90%. Οι βασικοί στόχοι που είχαν οριστεί και υλοποιήθηκαν κατά την σχεδίαση της εφαρμογής ήταν:

- Η εφαρμογή να μπορεί να τρέχει από διαφορετικές συσκευές όπως υπολογιστή ή κινητά.
- Την δυνατότητα ύπαρξης πολλαπλών ομάδων για την αποστολή διαφορετικών δεδομένων ανά ομάδα.
- Τη δυνατότητα αποστολή τόσο κειμένων αλλά και αρχείων καθώς και η διαγραφή τους.

## 5.2 Μελλοντικές βελτιστοποιήσεις και νέοι στόχοι

Δεδομένου ότι η εφαρμογή υλοποιήθηκε στα πλαίσια ατομικής πτυχιακής εργασίας δεν υπήρχε δυνατότητα να συμπεριληφθούν περαιτέρω λειτουργικότητες και χαρακτηριστικά. Μερικές σημαντικές προσθήκες και τροποποιήσεις που θα μπορούσαν να πραγματοποιηθούν σε μελλοντικό χρόνο αναφέρονται παρακάτω και μερικά από τα οποία



θα υλοποιηθούν μέχρι το τέλος του έτους 2018 εκτός των πλαισίων αυτής της πτυχιακής εργασίας.

- (1) **Δυνατότητα share των ομάδων σε άλλους χρήστες.** Αυτή η λειτουργικότητα υπήρχε στους αρχικούς στόχους. Ωστόσο δεν υλοποιήθηκε καθώς απαιτούσε την προσθήκη πολλών νέων πεδίων στην βάση δεδομένων, αλλά και την λήψη περισσότερων περιπτώσεων κατά την διαγραφή/μετονομασία μιας ομάδας αλλά και την διαγραφή μηνυμάτων. Ένα χαρακτηριστικό παράδειγμα είναι, όταν μια ομάδα του χρήστη Α μοιραστεί σε κάποιον χρήστη Β τότε ο χρήστης Β δεν θα πρέπει να έχει δυνατότητα διαγραφής ή αλλαγή του ονόματος της ομάδας. Επίσης δεν θα του επιτρέπεται να διαγράψει μηνύματα τα οποία δεν τα έχει στείλει εκείνος.
- (2) **Δυνατότητα επεξεργασίας των μηνυμάτων.** Επίσης και αυτό υπήρχε στους αρχικούς στόχους. Αυτό είναι κάτι το οποίο στην παρούσα φάση μπορεί να υλοποιηθεί με την προσθήκη μικρών έξτρα συναρτήσεων.
- (3) **Μετατροπή του server από http σε https.** Από την στιγμή που υπάρχει σύστημα εισόδου/εγγραφή του χρήστη θα ήταν καλό να είναι η σύνδεση https. Να σημειωθεί ότι έγιναν μικρές προσπάθειες για δημιουργία του server σε https αλλά απέτυχαν.
- (4) Εμφάνιση μικρών **thumbnail** για φωτογραφίες και **frame** των ιστοσελίδων που έχει στείλει ο χρήστης σαν μήνυμα.
- (5) **Βελτιστοποιήσει στον server** ώστε να μπορεί να έχει την δυνατότητα να εμφανίζει κατάλληλες σελίδες σε περιπτώσεις που γίνει πρόσβαση σε σελίδα που δεν υπάρχει ή υπάρξει κάποιο σφάλμα από την μεριά του server.
- (6) **Βελτιστοποίηση στην βάση δεδομένων** ώστε να επιστρέφει τα λιγότερα δυνατά δεδομένα ανάλογα την επερώτηση. Καθώς και optimization σε περίπτωση που υπάρχει καθυστέρηση λόγω φόρτου εργασίας όταν υπάρχουν πολλοί και ταυτόχρονοι χρήστες.
- (7) Δυνατότητα **αναζήτησης** μιας ομάδας του χρήστη. Δεδομένου ότι θα μπορούσε ένας χρήστη να έχει όσες ομάδες θέλει, θα ήταν τρομερά κουραστικό να ψάχνει μια ομάδα κάνοντας scroll πολύ κάτω.
- (8) Δυνατότητα επιλογής **avatar** για εικόνα profile του χρήστη το οποίο θα μπορούσε να εμφανίζεται αργότερα και στα μηνύματα που στέλνει.
- (9) Δημιουργία της **εφαρμογής σε android application**, ώστε να προσφέρεται η καλύτερη δυνατή εμπειρία για τους χρήστες κινητών συσκευών.

## 6 Βιβλιογραφία

- [1] RethinkDB – <https://www.rethinkdb.com/>
- [2] AngularJS v1 – <https://angularjs.org/>
- [3] Bootstrap 4 – <https://getbootstrap.com/>
- [4] Node JS v9.4.0 – <https://nodejs.org/en/>
- [5] CSRF - [https://en.wikipedia.org/wiki/Cross-site\\_request\\_forgery](https://en.wikipedia.org/wiki/Cross-site_request_forgery)
- [6] JSON vulnerability - [https://docs.angularjs.org/api/ng/service/\\$http#json-vulnerability-protection](https://docs.angularjs.org/api/ng/service/$http#json-vulnerability-protection)
- [7] Socket.io - <https://socket.io/>
- [8] Stack overflow - <https://stackoverflow.com/>
- [9] W3Schools - <https://www.w3schools.com/>
- [10] Github - <https://github.com/>
- [11] RethinkDB installation - <https://www.rethinkdb.com/docs/install/ubuntu/>
- [12] NodeJS installation - <https://nodejs.org/en/download/package-manager/>