ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΥΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Σχεδίαση και Ανάπτυξη Δικτυακής Εφαρμογής για Κοινωνική Δικτύωση σε Κοντινή Απόσταση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΖΕΝΙΟΥ ΖΕΝΙΟΥ

Επιβλέπων : Δημήτριος Ασκούνης
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΥΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΖΕΝΙΟΥ ΖΕΝΙΟΥ

Επιβλέπων : Δημήτριος Ασκούνης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 27η Φεβρουαρίου 2018.

.....	.....	.....
Δημήτριος Ασκούνης	Ψαρράς Ιωάννης	Δούκας Χρυσόστομος
Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2018

.....

ΖΕΝΙΟΣ ΖΕΝΙΟΥ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στην παρούσα διπλωματική εργασία, σχεδιάστηκε και υλοποιήθηκε μία εφαρμογή κοινωνικής δικτύωσης για κινητά τηλέφωνα, τύπου Android. Αρχικά παρουσιάζεται μία σχετική έρευνα περί μέσων κοινωνικής δικτύωσης. Στη συνέχεια αναλύονται τα απαιτούμενα και ο σχεδιασμός της εν λόγω εφαρμογής, από μία τεχνολογικά αγνωστικιστική όψη. Ακολουθεί, αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν στα πλαίσια αυτής της εργασίας και στον τρόπο που εφαρμόστηκαν. Στη συνέχεια περιγράφεται η σχεδίαση και υλοποίηση της εφαρμογής. Εδώ δίνεται έμφαση στις τεχνολογίες και στα αρχιτεκτονικά χαρακτηριστικά που απαρτίζουν την εφαρμογή. Κλείνοντας γίνεται αξιολόγηση της εφαρμογής, μέσω δοκιμής χρηστών και συλλογής απόψεων. Τέλος, παρουσιάζονται τα συμπεράσματα της παρούσας διπλωματικής εργασίας και πιθανές μελλοντικές προεκτάσεις.

Λέξεις Κλειδιά: Κοινωνικό Δίκτυο, Android, REST, Django, React Native

Abstract

This diploma thesis captures the process of designing and implementing a social networking application, for Android smart-phones. Firstly, relevant research is presented on the subject of social networking and media. Then, the requirements and design of said application are analyzed, from a technology-agnostic point of view. The tecnhologies used for the development are then presented, along with their respective ways of application. Then follows the documentation for the design and implementation of the application. Emphasis is given here, to the architectural and technological characteristics of the application. Finally, an evaluation of the application takes place, through user-testing and surveying of users. In closing, a conclusion is presented along with possible future extensions.

Keywords: Social Network, Android, REST, Django, React Native

Ευχαριστίες

Ευχαριστώ τον Καθηγητή κ. Δημήτριο Ασκούνη για την ανάθεση της παρούσας διπλωματικής εργασίας, καθώς και τον Διδάκτωρ κ. Ιωσήφ Αλβέρτη για την υποστήριξη και καθοδήγηση που μου πρόσφερε κατά την εκπόνηση της εργασίας.

Ένα πολύ μεγάλο ευχαριστώ οφείλω επίσης στους συμφοιτητές και φίλους μου, Παναγιώτη Μουλλωτού και Χρυσόστομο Χρίστου, των οποίων η συνεργασία και στήριξη έκανε δυνατή την ολοκλήρωση των σπουδών μου στο Εθνικό Μετσόβιο Πολυτεχνείο. Θα ήθελα επίσης να αναφέρω την πολύτιμη φιλοξενία και στήριξη του φίλου μου Νεόφυτου Κολοκοτρώνη, κατά τα τελευταία στάδια της φοίτησής μου.

Τέλος, στη μητέρα και στήριγμά μου Φλώρα Ζένιου, θέλω να εκφράσω την ευγνωμοσύνη μου για την ακράδαντη εμπιστοσύνη και απαράμιλλη υποστήριξη της. Δεν υπάρχουν λόγια ισάξια των όσων μου έχεις χαρίσει. Η διπλωματική αυτή είναι αφιερωμένη σε εσένα.

Περιεχόμενα

1. Εισαγωγή

1.1 Πρόλογος

Η ραγδαία εξέλιξη της τεχνολογίας κατά τα τέλει του 20ου και αρχή του 21ου αιώνα έχει οδηγήσει στην ανάπτυξη του τομέα των τηλεπικοινωνιών. Η ανάπτυξη αυτή έχει φέρει μαζί της την εξέλιξη και διάδοση των κινητών τηλεφωνικών συσκευών. Πλέον το κάθε άτομο έχει στην διάθεση του, πολλαπλές φορητές συσκευές (smart-phones, tablets κλπ).

Παράλληλα με την σμίκρυνση και εξέλιξη των φορητών συσκευών, παρατηρούμε την εμφάνιση των κοινωνικών δικτύων ως αναπόσπαστο κομμάτι τους. Εν έτη 2018, είναι κοινώς αποδεκτό το γεγονός, ότι σαν άτομα ξοδεύουμε ένα μεγάλο κομμάτι της καθημερινότητάς μας χρησιμοποιώντας εφαρμογές που μας επιτρέπουν να συμμετέχουμε σε κοινωνικά δίκτυα.

Εφαρμογές όπως είναι το Facebook, Twitter και Instagram έχουν αλλάξει σημαντικά τον τρόπο με τον οποίο αλληλεπιδρούμε μεταξύ μας. Κατά τα πρώτα έτη του 21ου αιώνα, είδαμε στον δυτικό κόσμο την μετατόπιση των κοινωνικών δικτύων, από τον φυσικό κόσμο στον ηλεκτρονικό κόσμο. Διαδικασίες και δραστηριότητες που δεν μπορούσαν παρά να είναι περιορισμένες από φυσικά στοιχεία όπως είναι η απόσταση, παίρνουν πλέον μέρος στον κυβερνοχώρο χωρίς τους εν λόγω περιορισμούς. Ενημέρωση και επικοινωνία, παγίωση και επέκταση του επαγγελματικού κύκλου, αναζήτηση εκδηλώσεων και υπηρεσιών , είναι λίγες από τις δραστηριότητες που απαρτίζουν τον κόσμο των κοινωνικών δικτύων σήμερα.

Παρά την αδιαμφισβήτητη πρόοδο και διευκόλυνση που προσφέρει όμως η ψηφιακή κοινωνική δικτύωση, είναι συνήθως συνοδευόμενη από άυλα κόστη. Τα σημαντικότερα είναι ίσως η παραχώρηση προσωπικών δεδομένων αλλά και δεδομένων χρήσης. Οι πλείστες πλατφόρμες κοινωνικής δικτύωσης σήμερα αφαιρούν σε κάποιο βαθμό την δυνατότητα από τους χρήστες, να επιλέξουν το κοινό στο οποίο είναι εμφανής οι πληροφορίες τους. Επίσης, δεν υπάρχει κανένας έλεγχος από τον χρήστη, σχετικά με το πως και σε τι βαθμό θα γίνει εκμετάλλευση των δεδομένων χρήσης τους.

1.2 Σκοπός

Λαμβάνοντας υπόψη τα παραπάνω, η παρούσα διπλωματική εργασία εκπονήθηκε με σκοπό την ανάπτυξη μίας ψηφιακής πλατφόρμας κοινωνικής δικτύωσης, η οποία δίνει στους χρήστες την εκτεταμένη δυνατότητα ελέγχου των προσωπικών τους δεδομένων αλλά και την αλληλεπίδραση βάσει τοποθεσίας. Οι χρήστες θα έχουν την ευχέρεια και επιλογή να απέχουν από την πλατφόρμα,

να ορίσουν τις φυσικές αποστάσεις στις οποίες θα δραστηριοποιούνται, αλλά και να περιορίσουν την ορατότητα των δεδομένων τους ως προς τους υπόλοιπους χρήστες. Παράλληλα, η πλατφόρμα αυτή θα επιτρέπει στους χρήστες να δραστηριοποιούνται με γνώριμες λειτουργίες όπως είναι η δημοσίευση status, η συμμετοχή σε συζητήσεις και η αντίδραση μέσω αντιδράσεων “like” ως προς τις δραστηριότητες άλλων χρηστών εντός του κύκλου τους.

1.3 Δομή Διπλωματικής Εργασίας

Κεφάλαιο 1

Παρόν κεφάλαιο, σκοπό έχει να κατατοπίσει τον αναγνώστη ως προς το θέμα, δομή και περιεχόμενο της παρούσας διπλωματικής εργασίας

Κεφάλαιο 2

Σε αυτό το κεφάλαιο γίνεται μία σχετική έρευνα περί θεωρητικών θεμάτων που αφορούν την εργασία. Τέτοια είναι η θεωρία κοινωνικών δικτύων, η επιρροή των κινητών συσκευών(smart-phones, tablets κλπ) στην κοινωνία και το τρέχων τοπίο όσο αφορά την ψηφιακή δικτύωση και τις υπηρεσίες που την καθιστούν δυνατή.

Κεφάλαιο 3

Στη συνέχεια, καταγράφονται και αναλύονται οι απαιτήσεις και τα χαρακτηριστικά της εφαρμογής που υλοποιήθηκε. Δίνεται έμφαση σε χαρακτηριστικά που δεν αφορούν τεχνολογίες. Αντ’αυτού εστιάζουμε στην οπτική γωνία του χρήστη, τις δυνατότητες που θα του/ης δίνει η εφαρμογή και όλες τις λειτουργίες που θα είναι επιτρεπτές.

Κεφάλαιο 4

Εδώ γίνεται μία εκτενής καταγραφή του κύκλου υλοποίησης της εφαρμογής. Παρουσιάζουμε αρχικά τις τεχνολογίες που επιλεγήκαν για την υλοποίηση, αναλύεται η αρχιτεκτονική του όλου συστήματος και καταγράφεται η δομή των οντοτήτων που το απαρτίζουν.

Κεφάλαιο 5

Με την ολοκλήρωση της εφαρμογής, προχωράμε στην αξιολόγηση του από την οπτική γωνία του χρήστη. Παρουσιάζουμε αρχικά την μέθοδο αξιολόγησης που επιλέγηκε και στην συνέχεια δίνεται το ερωτηματολόγιο με το οποίο οι χρήστες κλήθηκαν να μοιραστούν την άποψή τους. Αναλύονται στη συνέχεια, τα αποτελέσματα της και παρουσιάζονται επίσης τα συμπεράσματα που προέκυψαν από την αξιολόγηση.

Κεφάλαιο 6

Τέλος, γίνεται μία ανασκόπηση των όσων προαναφέρθηκαν και παρουσιάζονται πιθανές μελλοντικές προεκτάσεις.

2. Σχετική Έρευνα

2.1 Κοινωνικά Δίκτυα

Ο όρος *κοινωνικό δίκτυο* αναφέρεται σε έναν τύπο κοινωνικής δομής, ο οποίος αποτελείται από ένα σύνολο κοινωνικών οντοτήτων και ένα σύνολο δυαδικών δεσμών μεταξύ των οντοτήτων. Οι οντότητες αυτές μπορεί να είναι είναι άτομα, κοινωνικές ομάδες, πολιτικά κινήματα κλπ. Η ιδέα του κοινωνικού δικτύου επιτρέπει την χρήση ενός συνόλου μεθόδων με σκοπό την ανάλυση κοινωνικών δομών, η οποία με τη σειρά της επιτρέπει την αναγνώριση τοπικών και καθολικών προτύπων, τον εντοπισμό οντοτήτων με επιρροή και την εξέταση της κοινωνικής δυναμικής. [1]

Αν και οι πρώτες αναφορές σε κοινωνικά δίκτυα έγιναν στα τέλη του 19ου αιώνα, σημαντική ανάπτυξη στο πεδίο παρατηρήθηκε κατά την δεκαετία 1930 από διαφορετικές ομάδες ερευνητών στα πεδία της ψυχολογίας, ανθρωπολογίας και των μαθηματικών ξεχωριστά. Η ανάλυση κοινωνικών δικτύων άρχισε να γίνεται όλο και πιο διάσημη στην δεκαετία των 1990, κατά την οποία αναπτύχθηκαν και εφαρμόστηκαν μοντέλα και μέθοδοι στα δεδομένα που άρχισαν να γίνονται διαθέσιμα μέσω του διαδικτύου και συγκεκριμένα μέσω των υπηρεσιών κοινωνικής δικτύωσης.

Τα οφέλη της ανάλυσης κοινωνικών δικτύων γίνονται όλο και πιο εμφανή καθώς τα δίκτυα μεγαλώνουν, εφόσον μπορούν προσομοιώσουν ένα σύνολο με μεγαλύτερη ακρίβεια. Παρόλα αυτά όμως, η ανάλυση ενός καθολικού δικτύου δεν είναι εφικτή λόγω πρακτικών περιορισμών όπως είναι για παράδειγμα η υπολογιστική ισχύς. [2] Για τον λόγο αυτό, τα κοινωνικά δίκτυα αναλύονται σε διαφορετικές κλίμακες, αναλόγως του σκοπού της ανάλυσης. Τα επίπεδα αυτά μπορεί να είναι ένα εκ των παρακάτω.

- **Μικρού(ή Μικροσκοπικού) Επιπέδου:** Στο επίπεδο αυτό η ανάλυση ξεκινά στο ατομικό επίπεδο ή στο επίπεδο μίας μικρής ομάδας ατόμων, και επεκτείνεται καθώς εντοπίζονται κοινωνικές σχέσεις μεταξύ των οντοτήτων

- **Μεσαίου Επιπέδου:** Στο επίπεδο αυτό η ανάλυση ξεκινά με ένα μέγεθος πληθυσμού που εμπίπτει μεταξύ του μικρού και μεγάλου επιπέδου. Παρόλα αυτά, το επίπεδο αυτό μπορεί να αναφέρεται σε ανάλυση που σκοπό έχει τον εντοπισμό σχέσεων μεταξύ των δύο αυτών επιπέδων.

- **Μεγάλου(ή Μακροσκοπικού) Επιπέδου:** Αντί παρακολούθησης διαπροσωπικών αλληλεπιδράσεων, μια ανάλυση του τύπου αυτού ακολουθεί συνήθως τις επιδράσεις αλλαγών στο περιβάλλον ενός μεγάλου σε αριθμό πληθυσμού.

2.2 Υπηρεσίες Κοινωνικής Δικτύωσης

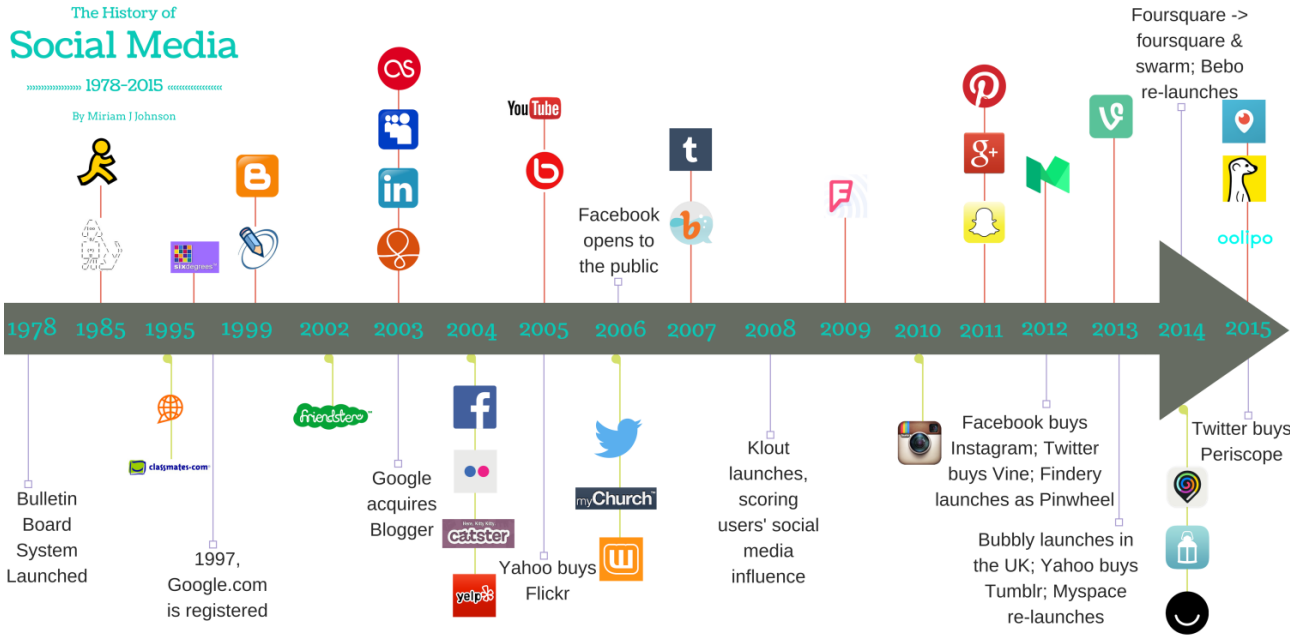
Ο όρος *υπηρεσία(ή μέσο) κοινωνικής δικτύωσης(ΥΚΔ ή ΜΚΔ)* αναφέρεται σε μία διαδικτυακή πλατφόρμα που επιτρέπει στους χρήστες της να ενταχθούν και να συμμετάσχουν σε κοινωνικά δίκτυα βάσει ενδιαφερόντων, δραστηριοτήτων, κοινωνικών υποβάθρων ή και πραγματικών προϋπαρχόντων σχέσεων. [3] Σε πολλές περιπτώσεις όμως ο ορισμός αυτός δεν αρκεί για να περιγράψει το σύνολο των ποικίλων εφαρμογών και πλατφορμών που είναι διαθέσιμες σήμερα. Τα χαρακτηριστικά όμως που θα μπορούσαμε να πούμε με βεβαιότητα πως περιγράφουν όλες τις πλατφόρμες σήμερα, είναι η διαδικτυακή τους φύση και η δημοσίευση περιεχομένου από τους ίδιους τους χρήστες.

Οι πλατφόρμες κοινωνικής δικτύωσης όπως τις ξέρουμε σήμερα, ξεκίνησαν στα μέσα της δεκαετίας του 1990 σαν διαδικτυακές κοινότητες οι οποίες δραστηριοποιούνταν μέσω υπηρεσιών δωματίων συζήτησης(chat rooms), όπως το theGlobe. Κατά τα τέλει της δεκαετίας, το *προφίλ χρήστη*, ένα από τα κοινά χαρακτηριστικά των σημερινών ΜΚΔ έκανε την εμφάνισή του, μαζί με την ιδέα της λίστας φίλων, μέσω της οποίας οι χρήστες είχαν την επιλογή να συμπεριλάβουν άλλους χρήστες στον κύκλο τους. [4] Αυτή η μορφή ΜΚΔ, συνοδευόμενη από τον σημαντικό βαθμό υιοθέτησης του διαδικτύου από το κοινό, κατέστησε τις υπηρεσίες κοινωνικής δικτύωσης αναπόσπαστο κομμάτι της ολικής εμπειρίας του διαδικτύου. Τέτοιες υπηρεσίες ήταν τα SixDegrees το 1997 και Friendster αργότερα το 2002. Ένα χρόνο μετά είδαμε την ίδρυση του LinkedIn κατά το 2003, μία ΥΚΔ η οποία σκοπό είχε την κοινωνική δικτύωση σε επαγγελματικό επίπεδο, μέσω ανταλλαγής μηνυμάτων και δημοσίευσης περιεχομένου. Παράλληλα το MySpace έκανε την εμφάνιση του, και 2 μετά ήταν ένα από τα διασημότερα ΜΚΔ. Πολύ διάσημο ήταν επίσης κατά το ίδιο χρονικό διάστημα το hi5, το οποίο είχε παρόμοια χαρακτηριστικά με το προαναφερθέν MySpace.[5]

Το 2004 στο Πανεπιστήμιο Χάρβαρντ λανσάρεται το Facebook (τότε TheFacebook) από τους Mark E. Zuckerberg και Eduardo L. Saverin, ως μέσο κοινωνικής δικτύωσης για φοιτητές εντός του πανεπιστημίου. Επεκτάθηκε όμως γρήγορα σε άλλα σχολεία εντός και εκτός της χώρας, και μέχρι το 2009 ήταν πλέον η μεγαλύτερη υπηρεσία κοινωνικής δικτύωσης και εξακολουθεί να είναι σήμερα.

Τον Φεβρουάριο του 2005 ξεκίνησε επίσης το YouTube, ως πλατφόρμα δημοσίευσης βίντεο. Το YouTube δίνει την δυνατότητα στους χρήστες να δημοσιεύσουν και να παρακολουθήσουν υλικό από άλλους, να αξιολογήσουν, να σχολιάσουν το εν λόγω υλικό αλλά και να “ακολουθήσουν” συγκεκριμένους παραγωγούς υλικού.[6] Η εταιρεία εξαγοράστηκε από την

Google LLC περίπου ενάμισι χρόνο μετά το λανσάρισμα της ιστοσελίδας τον Νοέμβριο του 2006. Η ιστοσελίδα εξακολουθεί να είναι η δημοφιλέστερη υπηρεσία έκδοσης βίντεο, και η δεύτερη δημοφιλέστερη ιστοσελίδα ανά το παγκόσμιο. [7]

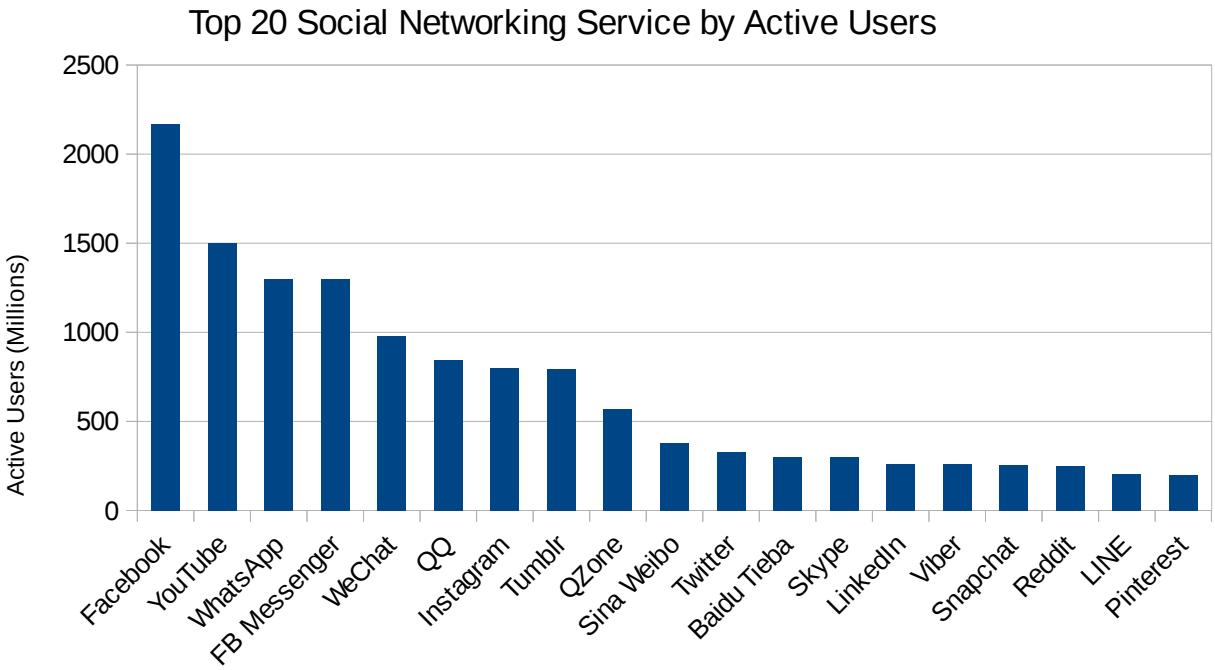


Σχήμα 2.1: Η ιστορία των μέσων κοινωνικής δικτύωσης(1978-2015) Miriam J. Johnson

Η άνθηση του τομέα των ΜΚΔ συνέχισε στα επόμενα χρόνια, με την είσοδο νέων και καινοτόμων ιδεών στην αγορά. Το 2006 λανσαρίστηκε το Twitter, μία πλατφόρμα ενημέρωσης και κοινωνικής δικτύωσης όπου οι χρήστες επικοινωνούσαν με μικρά μηνύματα των 140 (αργότερα 280) χαρακτήρων γνωστά ως “tweet” (μετάφραση: “τιτίβισμα”). Το 2007 ιδρύθηκε το Tumblr μία υπηρεσία micro-blogging. Το WhatsApp μία εφαρμογή ανταλλαγής μηνυμάτων μεταξύ χρηστών και ομάδων έκανε την εμφάνισή του το 2009. Το Instagram και Pinterest ξεκίνησαν το 2010 ως πλατφόρμες δημοσίευσης εικόνας και βίντεο. Το Instagram εξαγοράστηκε από το Facebook τον Απρίλιο του 2012 μετά από ενάμισι χρόνο λειτουργίας.

Άλλες αξιοσημείωτες υπηρεσίες κοινωνικής δικτύωσης είναι φυσικά το Snapchat, μία πλατφόρμα ανταλλαγής μηνυμάτων υπό την μορφή εικόνας και βίντεο. Η κεντρική ιδέα του Snapchat είναι το ότι όλα τα μηνύματα και εικόνες, είναι διαθέσιμα στον παραλήπτη για ένα μικρό χρονικό διάστημα. Παρόμοια πλατφόρμα είναι το Vine, που επιτρέπει την δημοσίευση βίντεο μικρής διάρκειας (6 δευτερόλεπτων) και την εγγραφή των χρηστών σε “εκδότες” που τους ενδιαφέρουν. Τέλος, η εφαρμογή Tinder η οποία λανσαρίστηκε τον Σεπτέμβριο του 2012 επιτρέπει στους χρήστες να κτίσουν ένα προφίλ που αποτελείται από ένα μικρό κείμενο, και ένα περιορισμένο αριθμό φωτογραφιών. Στη συνέχεια η χρήστες μπορούν να ενωθούν με άτομα που βρίσκονται στην ίδια γεωγραφική περιοχή, αφού τα δύο άτομα “αλληλο-εγκριθούν” βάση των προφίλ τους.

Σήμερα υπάρχουν περισσότερες από 200 κοινώς γνωστές υπηρεσίες κοινωνικής δικτύωσης και πολλές ακόμα λιγότερο διαδεδομένες. Τα μέσα κοινωνικής δικτύωσης είναι πλέον αναπόσπαστο κομμάτι της εμπειρίας του διαδικτύου, και με την εξέλιξη των τηλεπικοινωνιακών και έξυπνων συσκευών μεγάλο κομμάτι της καθημερινότητάς μας, αφού οι πλείστες ΥΚΔ σήμερα επιτρέπουν χρήση μέσω smart-phones.



Σχήμα 2.2: Δημοφιλέστερες υπηρεσίες κοινωνικής δικτύωσης (Ιανουάριος 2018)

2.3 Ζητήματα Υπηρεσιών Κοινωνικής Δικτύωσης

Υπάρχουν πολλά θέματα και ζητήματα που αξίζουν ανάλυσης, παρόλα αυτά μια εκτενής έρευνα των ζητημάτων περί υπηρεσιών κοινωνικής δικτύωσης, ξεφεύγει από το πλαίσια μίας διπλωματικής εργασίας. Βαρύτητα θα δώσουμε παρακάτω σε θέματα που έχουν να κάνουν με τα υποκείμενα δεδομένα και πληροφορίες που συνιστούν οι ΥΚΔ. Θα αναφερθούν όμως εδώ κάποια από τα ζητήματα που δεν αναλύονται, για λόγους πληρότητας.

Τέτοια, μπορεί να είναι οι συνέπειες ψυχικής υγείας που εμφανίζονται λόγω της υπερβολικής χρήσης των ΜΚΔ, οι επιπτώσεις της χρήσης των ΜΚΔ στην καθημερινή ζωή και αντιστρόφως, η δυσκολία διαχωρισμού της καθημερινής(φυσικής) ζωής από το περιβάλλον των ΜΚΔ. Επίσης, ο κίνδυνος ασφάλειας των νεαρών ατόμων από κακόβουλους χρήστες, η κοινωνική πίεση και υπερκόπωση που μπορεί να προκαλέσει η βαρύτητα που δίνουμε σαν κοινωνία στα

ΜΚΔ. Αυτά και πολλά άλλα είναι σημαντικά ζητήματα που έχουν να κάνουν με την ευημερία των ατόμων και της κοινωνίας, ως προς την ενσωμάτωση των ΜΚΔ στην ζωή μας.[5]

2.3.1 Μυστικότητα

Η μυστικότητα και ασφάλεια προσωπικών δεδομένων είναι ίσως το σημαντικότερο θέμα που περιτριγυρίζει τα ΜΚΔ. Σε τι βαθμό είναι ασφαλές για ένα χρήστη η αποκάλυψη προσωπικών δεδομένων και γραφικού περιεχομένου; Παρά τις προσπάθειες των υπηρεσιών αυτών να εξασφαλίσουν την ασφάλεια δεδομένων για τους χρήστες τους, ο κίνδυνος κλοπής είτε από κακόβουλους τρίτους ή ιούς ελλοχεύει στα μάτια του κοινού. Σε πολλές περιπτώσεις μεγάλες υπηρεσίες συνεργάζονται με όργανα του νόμου για να εξασφαλίσουν την αποφυγή τέτοιων συμβάντων. Πέραν αυτού, όμως υπάρχουν πολλοί παράγοντες που μπορούν να υπονομεύσουν την μυστικότητα σε τέτοιες υπηρεσίες, όπως για παράδειγμα η μεταπώληση ανώνυμων δεδομένων σε τρίτους. Αν και ο παραλήπτης τέτοιας πληροφορίας δεν είναι σε θέση να συνδέσει να την συνδέσει με το συγκεκριμένο άτομο που αφορά, δεν παύει να είναι προσωπικής φύσεως. [5][8][9]

2.3.2 Εξόρυξη Δεδομένων

Με σκοπό την βελτίωση της εμπειρίας χρήσης, οι ΥΚΔ πολλές φορές χρησιμοποιούν μεθόδους εξόρυξης δεδομένων, με σκοπό να δημιουργήσουν ένα προφίλ για τον κάθε χρήστη. Το προφίλ αυτό μπορεί με τη σειρά του να χρησιμοποιηθεί για την προσαρμογή του περιβάλλοντος του χρήστη εντός της πλατφόρμας. Αν για παράδειγμα ένας χρήστης κάνει συχνά έρευνες για δημοσιεύσεις σχετικές με κάποιο ενδιαφέρον του, το προφίλ που προαναφέρθηκε μπορεί να αντικατοπτρίσει αυτό του το ενδιαφέρον και να παρουσιάσει στον χρήστη περισσότερο υλικό σχετικά με το ενδιαφέρον αυτό. Με τον ίδιο τρόπο η υπηρεσία μπορεί να κάνει καλύτερη τοποθέτηση διαφημίσεων ή να επηρεάσει παθητικά την γνώμη του χρήστη σχετικά με ένα θέμα.

Το ιδίωμα *“If you are not paying for it, you are the product.”* είναι εύλογα συνδεδεμένο σήμερα με την χρήση των μέσων κοινωνικής δικτύωσης. Οι πλείστες ΥΚΔ σήμερα προσφέρουν μέχρι ένα σημείο το προϊόν τους δωρεάν στο κοινό, πρέπει επομένως να προσφύγουν σε διαφήμιση προκειμένου να έχουν έσοδα. Αυτό οδηγεί σε τακτικές που μπορεί μεν να μην είναι παράνομες, αλλά πολύ πιθανόν να βρίσκονται σε γκρίζες περιοχές από την οπτική πλευρά του χρήστη. Ας θεωρήσουμε για παράδειγμα την περίπτωση όπου μία υπηρεσία καταγράφει δεδομένα χρήσης σχετικά με την λειτουργία scrolling (κύλισης), στην κύρια σελίδα της πλατφόρμας της. Αν ένας χρήστης σταματάει την κύλιση σε συγκεκριμένου είδους υλικό, αυτό είναι μία πολύ δυνατή ένδειξη σχετικά με τον χαρακτήρα και τα ενδιαφέροντα του κάθε χρήστη. Παρόλο που τέτοιες τακτικές δεν είναι απαραίτητα κακόβουλες, βάζουν στα χέρια των υπηρεσιών σημαντική πληροφορία που τους επιτρέπει να επηρεάσουν την άποψη του κοινού σε μεγάλη κλίμακα. Αν για παράδειγμα ένας χρήστης σε συγκεκριμένη υπηρεσία, δέχεται υλικό προσαρμοσμένο στις πολιτικές του απόψεις, ο χρήστης αυτός είναι λιγότερο πιθανόν να αλλάξει τις απόψεις του εφόσον η όλη εμπειρία χρήσης

του περιορίζεται σε μία ιδεατή σφαίρα, σχεδιασμένη να του προσφέρει ικανοποίηση και επιβεβαίωση.

Άλλες υπηρεσίες μπορεί ακόμα και να παρακολουθήσουν την δραστηριότητα των χρηστών τους εκτός της ίδιας της πλατφόρμας που προσφέρουν. Ένα παράδειγμα είναι το Facebook Beacon, κομμάτι του διαφημιστικού συστήματος του Facebook που επέτρεπε την συλλογή πληροφοριών από εξωτερικές ιστοσελίδες προς το Facebook, με σκοπό την στοχευμένη διαφήμιση. Η διαδικασία αυτή ήταν εν λειτουργία ακόμα και αν ο χρήστης δεν ήταν συνδεδεμένος στο Facebook. Η υπηρεσία αυτή διακόπηκε το 2009 μετά από αγωγή που κινήθηκε προς την εταιρεία σχετικά με το θέμα.[10]

2.3.3 Πρόσβαση Πληροφοριών

Ένα επίσης πολύ σημαντικό ζήτημα είναι η πρόσβαση πληροφοριών του χρήστη από άλλους χρήστες ή τρίτους. Ο χρήστης πολλές φορές θέλει να περιορίζει το υλικό που δημοσιεύει σε ένα συγκεκριμένο σύνολο ατόμων. Επιπροσθέτως, ίσως κάποιος να θέλει να αποκρύψει ακόμα και το γεγονός ότι χρησιμοποιεί την εν λόγω υπηρεσία. Οι ΥΚΔ προσφέρουν πολλές φορές επιλογές, που επιτρέπουν στο χρήστη τέτοιου είδους ενέργειες, καθώς η πρόσβαση προσωπικών πληροφοριών από συγκεκριμένα άτομα, πιθανόν να έχει σοβαρό αντίκτυπο στον χρήστη. Σε πολλές περιπτώσεις, υποψήφιοι για θέσεις εργασίας απορρίφθηκαν και εργαζόμενοι απολύθηκαν εξαιτίας της δραστηριότητας τους σε υπηρεσίες κοινωνικής δικτύωσης. [5] Τέτοιες περιπτώσεις έχουν δημιουργήσει σημαντικές επιπλοκές σχετικά με την εμπλοκή εταιρειών και την δραστηριότητα των εργαζομένων τους σε ΥΚΔ.

3. Απαιτούμενα & Σχεδιασμός

3.1 Κύρια Χαρακτηριστικά

3.1.1 Προφίλ Χρήστη

Ο κάθε χρήστης θα μπορεί να δημιουργήσει ένα προφίλ το οποίο θα διατηρείται μεταξύ sessions. Το προφίλ αυτό θα περιλαμβάνει πληροφορίες όπως το όνομα και επίθετο του χρήστη και η ημερομηνία γέννησης. Οι υπόλοιποι χρήστες δεν θα μπορούν να δουν την ακριβή ημερομηνία γεννήσεως κάποιου άλλου χρήστη, αλλά μόνο την ηλικία του. Επίσης, ο χρήστης θα πρέπει να δώσει μία ηλεκτρονική διεύθυνση και ένα αναγνωριστικό username, το οποίο θα χρησιμοποιείται μόνο για λειτουργικούς σκοπούς και δεν θα είναι ορατό σε άλλους χρήστες. Τέλος, ο χρήστης θα έχει μία εικόνα προφίλ που θα τον αντιπροσωπεύει και θα είναι ορατή στους υπόλοιπους χρήστες

3.1.2 Δραστηριότητες / Ενδιαφέροντα

Πέραν των προσωπικών πληροφοριών, στο προφίλ του κάθε χρήστη θα είναι προσκολλημένες οι δραστηριότητες(Activities) που τον ενδιαφέρουν. Ο χρήστης θα έχει την δυνατότητα να εξερευνήσει μία συλλογή δραστηριοτήτων και να ακολουθήσει(Follow) αυτές που τον ενδιαφέρουν. Οι δραστηριότητες αυτές θα είναι ορατές στους υπόλοιπους χρήστες, μέσω του προφίλ του χρήστη.

3.1.3 Σύνδεση Χρηστών

Οι χρήστες θα μπορούν συνδεθούν με άλλους χρήστες μέσω της λειτουργίας ακολούθησης(Follow). Για να θεωρηθεί πλήρης η σχέση μεταξύ δύο χρηστών, θα πρέπει να ακολουθούν ο ένας τον άλλον. Αρχικά λοιπόν πρέπει κάποιος χρήστης να εντοπίσει το προφίλ κάποιου άλλου χρήστη και να αρχίσει να τον ακολουθεί. Στη συνέχεια ο δεύτερος χρήστης θα πρέπει αντίστοιχα να ακολουθήσει τον πρώτο, πριν θεωρηθεί πλήρης η σχέση τους, και γίνουν διαθέσιμοι περαιτέρω τρόποι αλληλεπίδρασης.

Κάθε χρήστης μπορεί να εντοπίσει νέους χρήστες με 2 τρόπους. Ο πρώτος τρόπος είναι μέσω της λειτουργίας εξερεύνησης(Discover). Η λειτουργία αυτή θα επιτρέπει στον κάθε χρήστη να ανακαλύψει προφίλ άλλων χρηστών που βρίσκονται σε απόσταση που ο ίδιος μπορεί να ελέγξει. Η ίδια απόσταση θα είναι επίσης το όριο στο οποίο το προφίλ του χρήστη θα είναι ορατό σε άλλους χρήστες μέσω της λειτουργίας(Discover). Ο δεύτερος τρόπος θα είναι μέσω κάποιας δραστηριότητας(Activity). Εάν για παράδειγμα ένας χρήστης θέλει να συνδεθεί με άλλους χρήστες μέσω κοινών ενδιαφερόντων, αρκεί ο χρήστης να καθορίσει την εν λόγω δραστηριότητα και θα έχει την δυνατότητα να ακολουθήσει άλλους χρήστες που την έχουν σαν κοινό.

3.1.4 Δημοσίευση Περιεχομένου

Οι χρήστες θα έχουν την δυνατότητα να δημοσιεύσουν περιεχόμενο υπό την μορφή κειμένου μικρού μήκους, μέσω της λειτουργίας Feed. Η ιδέα είναι ότι κάθε Feed θα είναι συσχετισμένο με μία δραστηριότητα, σαν ενημέρωση προς τον κύκλο του χρήστη ότι εμπλέκεται την συγκεκριμένη στιγμή με την εν λόγω δραστηριότητα. Επίσης πέραν του κειμένου, ο χρήστης θα έχει την δυνατότητα να μοιραστεί γραφικό υλικό μέσω του Feed, χωρίς όμως αυτό να είναι απαραίτητο.

Στη συνέχεια ο κάθε χρήστης θα μπορεί να δει και να αλληλεπίδραση με τα Feed που μοιράστηκαν τα άτομα εντός του κύκλου τους. Αυτό θα γίνεται με δύο τρόπους, την λειτουργία αρέσκειας(Like) και την λειτουργία σχολίων (Comment).

3.1.5 Κατάσταση Χρήσεως

Τέλος, ο χρήστης θα μπορεί να ελέγξει το κοινό που θα έχει πρόσβαση στο προφίλ του και στο περιεχόμενο που έχει δημοσιεύσει. Αυτό θα γίνεται με δύο “διακόπτες” που θα ελέγχουν διαφορετικές λειτουργίες. Αρχικά, ο διακόπτης τοποθεσίας θα μπορεί να ελέγξει κατά πόσο το προφίλ του χρήστη θα εμφανίζεται στην λειτουργία Discover άλλων χρηστών. Εάν είναι απενεργοποιημένος το προφίλ του χρήστη δεν θα εμφανίζεται κατά την λειτουργία Discover στους άλλους χρήστες, και αντίστοιχα ο χρήστης δεν θα μπορεί να χρησιμοποιήσει την λειτουργία Discover. Επίσης, ο διακόπτης κατάστασης θα μπορεί να ελέγξει την ολική παρουσία του χρήστη στην πλατφόρμα. Εάν ο διακόπτης κατάστασης είναι απενεργοποιημένος, τότε τα Feed που έχει δημοσιεύσει ο χρήστης δεν θα είναι εμφανή στους υπόλοιπους χρήστες και αντίστοιχα ο χρήστης δεν θα μπορεί να δει Feeds από άλλους χρήστες.

3.2 Λειτουργικές Απαιτήσεις

Ακολουθεί μία εκτενής λίστα το των λειτουργιών που υποστηρίζει η εφαρμογή. Οι λειτουργίες αυτές καταγράφονται από την οπτική γωνία του χρήστη, επομένως δεν αναλύονται ως προς τον τρόπο με τον οποίο θα υλοποιηθούν, αλλά με την εμπειρία χρήσης που θα προσφέρουν.

- **Δημιουργία Προφίλ**
  Ο χρήστης θα έχει την δυνατότητα να δημιουργήσει το προφίλ του/ης, το οποίο θα περιλαμβάνει username(αναγνωριστικό χρήστη), κωδικός πρόσβασης, όνομα, επίθετο, ηλεκτρονική διεύθυνση, ημερομηνία γέννησης και εικόνα προφίλ.

- **Ενημέρωση Προφίλ**
  Ο χρήστης θα έχει την δυνατότητα να αλλάξει τα προσωπικά στοιχεία που έδωσε κατά την δημιουργία προφίλ. Εξαιρείται το username του χρήστη το οποίο είναι μοναδικό και οριστικοποιείται κατά την δημιουργία προφίλ.

- **Αλλαγή Κατάστασης**
  Ο χρήστης θα έχει την δυνατότητα να αλλάξει την κατάσταση χρήσης της εφαρμογής, χρησιμοποιώντας τον διακόπτη κατάστασης και τον διακόπτη τοποθεσίας. Εάν ο διακόπτης

τοποθεσίας είναι απενεργοποιημένος ο χρήστης δεν είναι ορατός σε άλλους μέσω της λειτουργίας Discover. Εάν ο διακόπτης κατάστασης είναι απενεργοποιημένος οι δημοσιεύσεις του χρήστη δεν είναι ορατές από άλλους μέσω της λειτουργίας Feed.

- **Λειτουργία Discover**
  Ο χρήστης θα έχει την δυνατότητα να ανακαλύψει προφίλ χρηστών που βρίσκονται εντός συγκεκριμένης απόστασης από την τρέχουσα τοποθεσία του.
- **Φίλτρα Discover**
  Ο χρήστης θα έχει την δυνατότητα να μεταβάλει την μέγιστη απόσταση στην οποία ανακαλύπτει άλλους χρήστες, και ταυτόχρονα την απόσταση στην οποία τον ανακαλύπτουν άλλοι χρήστες. Επίσης, θα μπορεί να φιλτράρει περαιτέρω τα προφίλ που εμφανίζονται μέσω πεδίων όπως είναι η ηλικία.
- **Follow & Unfollow Χρηστών**
  Ο χρήστης θα πρέπει να έχει την δυνατότητα να ακολουθήσει (Follow) κάποιον άλλο χρήστη, όπως και επίσης να σταματήσει να τον ακολουθεί (Unfollow).
- **Επισκόπηση Λιστών Followers & Following**
  Ο χρήστης θα έχει την δυνατότητα ανά πάσα στιγμή να δει ποιοι χρήστες τον ακολουθούν και ποιους χρήστες ακολουθεί ο ίδιος.
- **Προβολή Προφίλ**
  Η εφαρμογή θα πρέπει να υποστηρίζει την προβολή ενός συγκεκριμένου προφίλ χρήστη ως μία σελίδα. Η λειτουργία αυτή θα είναι χρήσιμη κατά τις λειτουργίες Follow/Unfollow/Discover κλπ. Το προφίλ θα περιλαμβάνει όναμα, επώνυμο, ηλικία, απόσταση από τον χρήστη, την εικόνα προφίλ, και τις δραστηριότητες στις οποίες είναι εγγεγραμμένος ο χρήστης.
- **Activity / Δραστηριότητα**
  Η εφαρμογή θα πρέπει να προσφέρει την έννοια της δραστηριότητας. Η δραστηριότητα θα αντιπροσωπεύει για παράδειγμα ένα από τα ενδιαφέροντα του χρήστη και θα είναι επίσης συνδεδεμένο με την έννοια της δημοσίευσης(ή Feed) που εξαιτάζεται παρακάτω.
- **Εξερεύνηση & Follow Δραστηριοτήτων**
  Ο χρήστης θα μπορεί να εξερευνήσει μία λίστα δραστηριοτήτων. Στη συνέχεια θα μπορεί να ακολουθήσει(Follow) αυτές που τον ενδιαφέρουν, παρομοίως με την λειτουργία Follow χρηστών.
- **Unfollow Δραστηριοτήτων**
  Ο χρήστης θα έχει επίσης την δυνατότητα να σταματήσει να ακολουθεί(Unfollow) κάποια δραστηριότητα.

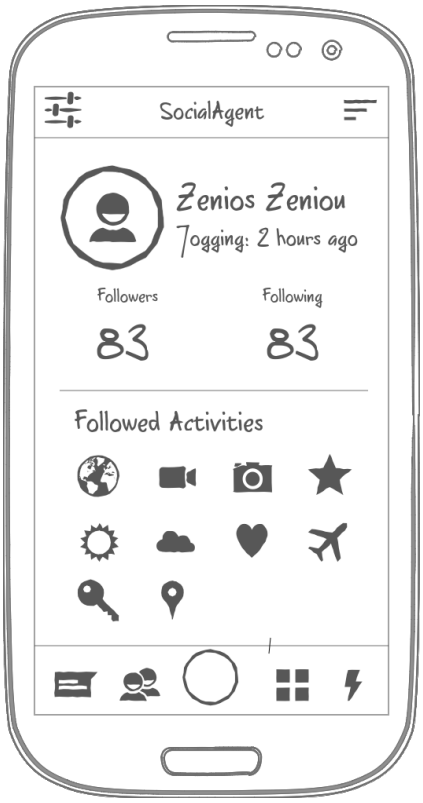
- **Επισκόπηση Λίστας Δραστηριοτήτων**
  Ο χρήστης θα μπορεί να δει μία λίστα με τις δραστηριότητες στις οποίες είναι εγγεγραμμένος, και να τις κάνει Unfollow εάν επιθυμεί.
- **Προβολή Δραστηριότητας**
  Η εφαρμογή θα πρέπει να υποστηρίζει την προβολή συγκεκριμένης δραστηριότητας ως μία σελίδα. Η λειτουργία αυτή θα είναι χρήσιμη κατά τις λειτουργίες Follow/Unfollow δραστηριοτήτων. Η σελίδα θα περιλαμβάνει το όνομα της δραστηριότητας, μία περιγραφή, ένα χαρακτηριστικό εικονίδιο και τον αριθμό των χρηστών που είναι εγγεγραμμένοι σε αυτήν.
- **Λειτουργία Feed**
  Ο χρήστης θα μπορεί να ξεφυλλίσει μία λίστα με “ενημερώσεις” που προέρχονται από τον ίδιο ή άλλους χρήστες. Οι ενημερώσεις αυτές τις οποίες θα ονομάζουμε επίσης Feed Items, είναι η κύρια μορφή επικοινωνίας μεταξύ των χρηστών. Θα είναι συνδεδεμένες με μία δραστηριότητα, και θα συμβολίζουν την εμπλοκή αν θέλετε του χρήστη, με αυτή την δραστηριότητα. Θα περιλαμβάνουν φυσικά ένα μικρού μεγέθους κείμενο, και προαιρετικά μία εικόνα. Κατά την λειτουργία Feed, χρήστης θα μπορεί να δει από ποιον χρήστη προέρχεται το feed item, πόσος χρόνος έχει περάσει από την δημοσίευση του, και σε πόση απόσταση έγινε η δημοσίευση αυτή.
- **Δημοσίευση Feed Item**
  Ο χρήστης θα έχει την δυνατότητα να δημοσιεύσει τα δικά του feed items, τα οποία θα μπορούν τα άτομα εντός του κύκλου του να δουν κατά την λειτουργία Feed.
- **Comment & Like**
  Οι χρήστες, κατά την λειτουργία Feed θα μπορούν να “αντιδράσουν” στα feed items που βλέπουν με δύο τρόπους. Ο πρώτος τρόπος είναι η λειτουργία Like, μέσω της οποίας ο χρήστης εκφράζει την αρέσκειά του. Ο δεύτερος τρόπος θα είναι η λειτουργία Comment, μέσω της οποίας ο χρήστης θα μπορεί να αντιδράσει σε ένα feed item χρησιμοποιώντας κείμενο μικρού μήκους
- **Επισκόπηση & Διαγραφή Προσωπικών Feed**
  Ο χρήστης θα μπορεί να δει μία λίστα με τα feed items τα οποία έχει ο ίδιος δημοσιεύσει. Θα έχει επίσης την δυνατότητα να διαγράψει κάποια από τα feed items εάν επιθυμεί.

3.3 Περιβάλλον Χρήστη

Εδώ παρατίθενται τα αρχικά σχέδια(mockups) που περιγράφουν το περιβάλλον και την εμπειρία χρήστη, της εφαρμογής. Αν και η τελική εφαρμογή διαφέρει σε μεγάλο βαθμό από τα σχέδια που ακολουθούν, χρησιμοποιήθηκαν για την κατεύθυνση της υλοποίησης διεπαφής.

3.3.1 Σελίδα Home – Κεντρική Σελίδα

Η σελίδα Home θα είναι η πρώτη σελίδα που θα συναντά ο χρήστης μετά την σύνδεση και είσοδο του στην εφαρμογή. Θα προσφέρει μία επισκόπηση του προφίλ χρήστη, καθώς και την δυνατότητα επιλογής του κοινού για τον χρήστη μέσω του διακόπτη τοποθεσίας και διακόπτη κατάστασης. Θα έχει επίσης, σημεία εισόδου όπως είναι η σελίδα ρυθμίσεων στην πάνω αριστερά γωνία, και η λειτουργία συζήτησης στην πάνω δεξιά γωνία(Η λειτουργία αυτή δεν υλοποιήθηκε τελικά, λόγω επιλογής άλλων χαρακτηριστικών).



Σχήμα 3.1 Σελίδα Home – Πρώτη Έκδοση

Σχήμα 3.2 Σελίδα Home – Δεύτερη Έκδοση

Δίνονται παραπάνω δύο διαφορετικές εκδόσεις της σελίδας Home. Η πρώτη έκδοση, δίνει περισσότερο έμφαση στην ιδέα ότι η κεντρική σελίδα είναι το προφίλ του χρήση και ότι θα είναι παρόμοια, αν όχι ίδια με αυτήν, που θα βλέπουν οι υπόλοιποι χρήστες σαν το προφίλ του εν λόγω χρήστη. Η μεν δεύτερη έκδοση, γέρνει περισσότερο προς την ιδέα ότι η σελίδα αυτή θα λειτουργεί σαν dashboard για τον χρήστη, δίνοντας του μία επισκόπηση και παράλληλα την δυνατότητα ελέγχου της όλης εφαρμογής.

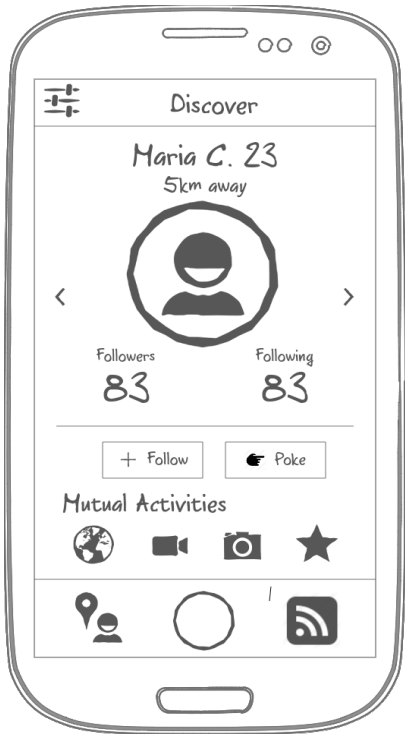
3.3.2 Σελίδες Followers & Following

Σχήμα 3.3 Σελίδα Followers

Μέσω της κεντρικής σελίδας, ο χρήστης θα μπορεί να μεταβεί στην σελίδα Followers και στην σελίδα Following. Οι δύο αυτές σελίδες θα είναι ουσιαστικά ίδιες, απλά με διαφορετικό περιεχόμενο.

Οι σελίδες αυτές θα καταγράφουν την λίστα των ατόμων που ακολουθούν (Follow) τον χρήστη ή αντίστοιχα που ακολουθούνται από τον χρήστη. Μέσω κάθε εικονιδίου ο χρήστης θα μπορεί να μεταβεί στην σελίδα προφίλ, η οποία δίνει περισσότερες λεπτομέρειες σχετικά με το κάθε άτομο.

3.3.3 Σελίδα Discover



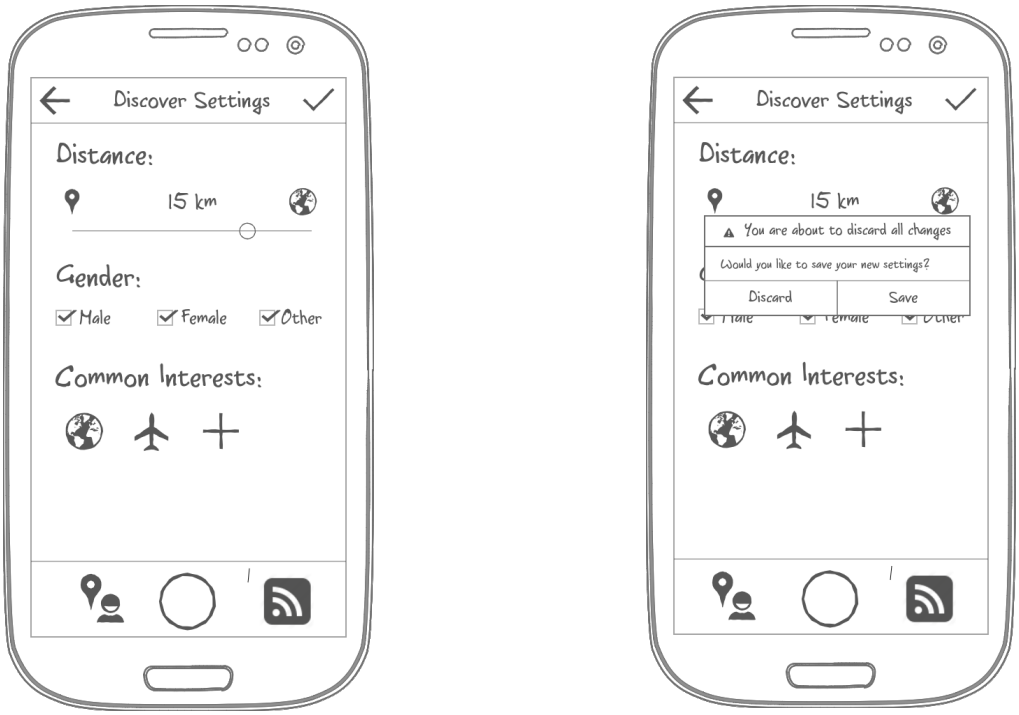
Σχήμα 3.4 Σελίδα Discover

Η σελίδα Discover θα προσφέρει στον χρήστη την δυνατότητα να ανακαλύψει και να δικτυωθεί με χρήστες που βρίσκονται κοντά του.

Κάποιες λεπτομέρειες σχετικά με τους χρήστες αυτούς θα είναι ορατές εδώ, όπως είναι το όνοματεπώνυμο, ηλικία, απόσταση, αριθμός followers / following καθώς και οι κοινές δραστηριότητες με τον χρήστη. Ο χρήστης θα μπορεί μέσω της σελίδας Discover να μεταβεί στην σελίδα ρυθμίσεων της λειτουργίας Discover.

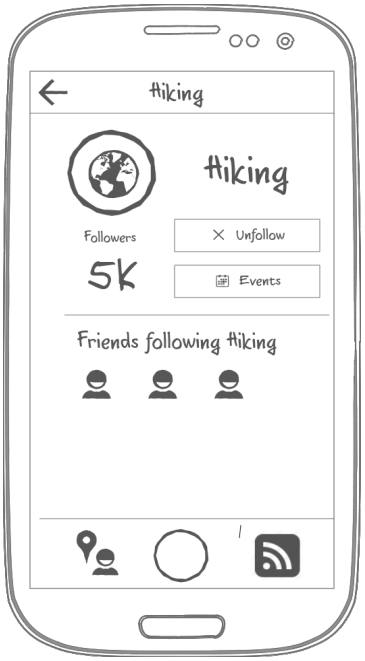
3.3.4 Σελίδα Discover Settings

Εδώ ο χρήστης θα μπορεί να ρυθμίσει το πεδίο εντός του οποίου θα ανακαλύπτει νέα άτομα μέσω της λειτουργίας Discover. Ο χρήστης θα μπορεί να επικυρώσει τις αλλαγές ρυθμίσεων(πάνω αριστερά) ή να τις ακυρώσει(πάνω δεξιά).



Σχήμα 3.5 Σελίδα Discover Settings

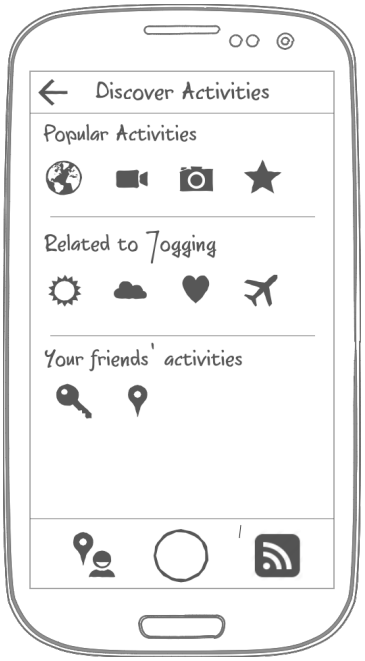
3.3.5 Σελίδα Activity



Σχήμα 3.6 Σελίδα Activity

Η σελίδα Activity θα προσφέρει μία επισκόπηση συγκεκριμένης δραστηριότητας και θα επιτρέπει στον χρήστη να ακολουθήσει(Follow) την εν λόγω δραστηριότητα. Πέραν αυτού θα απεικονίζει λεπτομέρειες σχετικά με την δραστηριότητα όπως ο αριθμός των χρηστών που την ακολουθούν, άτομα εντός του κύκλου του χρήστη που ακολουθούν την δραστηριότητα, λεπτομέρειες κ.α.

3.3.6 Σελίδα Activity Picker



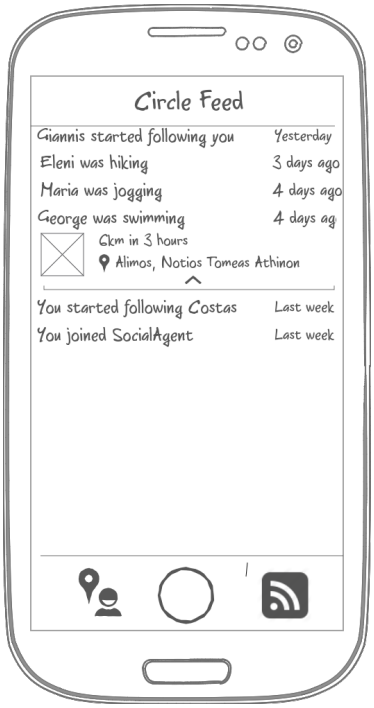
Σχήμα 3.7 Σελίδα Activity Picker

Σελίδα όπου ο χρήστης μπορεί να εξερευνήσει δραστηριότητες που θα δεν ακολουθεί ήδη, και να μεταβεί στην σελίδα Activity μέσω κάθε εικονιδίου.

3.3.7 Σελίδα Feed

Παρακάτω ακολουθούν δύο εκδόσεις της σελίδας Feed, μέσω της οποίας γίνεται δυνατή η λειτουργία Feed. Εδώ ο χρήστης θα βλέπει δημοσιεύσεις από τον κύκλο του και θα μπορεί επίσης να δημοσιεύσει ο ίδιος υλικό. Κάθε δημοσίευση θα συνοδεύεται από το όνομα του χρήστη-συγγραφέα, τον χρόνο που πέρασε από την δημοσίευση, την απόσταση κ.α.

Σχήμα 3.8 Σελίδα Feed – Πρώτη Έκδοση



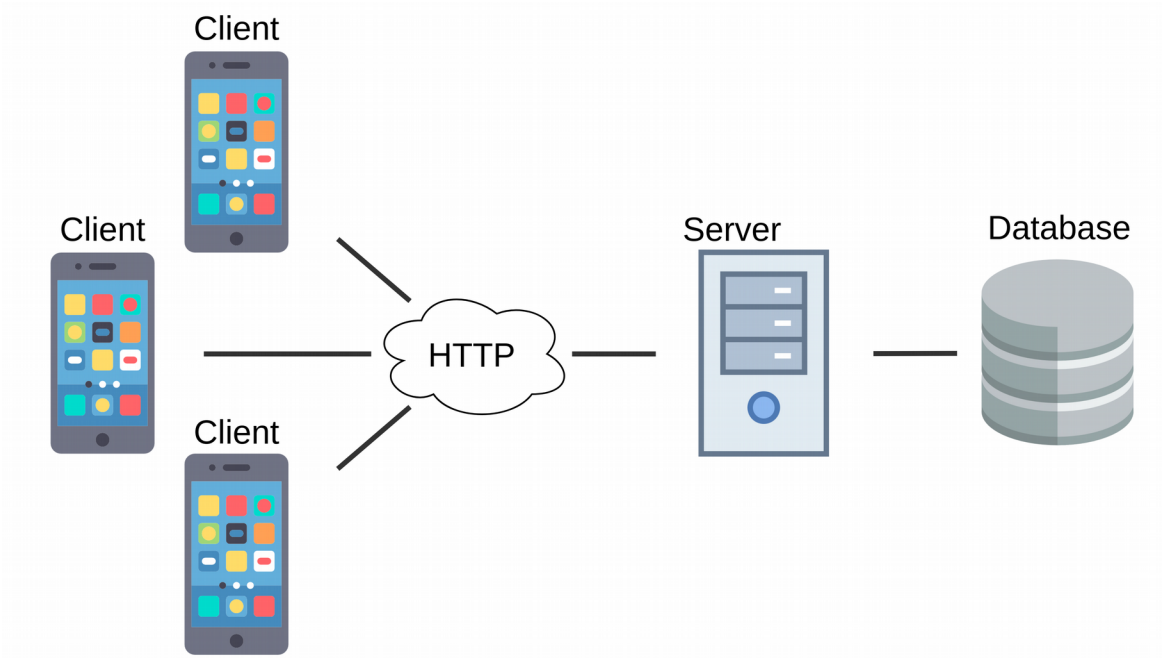
Σχήμα 3.9 Σελίδα Feed – Δεύτερη Έκδοση

4. Υλοποίηση

4.1 Αρχιτεκτονική & Τεχνολογίες

4.1.1 Επισκόπηση

Η όλη αρχιτεκτονική του συστήματος ακολουθεί το μοντέλο Διακομιστή-Πελάτη (Server-Client). Το μοντέλο αυτό αποτελείται από δύο κύριες οντότητες. Ο **διακομιστής(server)** επικοινωνεί με όλους τους **πελάτες(clients)**, εξυπηρετώντας αιτήματα τα οποία δέχεται από αυτούς μέσω του προτόκολλου HTTP.



Σχήμα 4.1 Αρχιτεκτονική Συστήματος

Ο κάθε πελάτης είναι μία περίπτωση χρήστη που χρησιμοποιεί την εφαρμογή μέσω του κινητού του. Ο διακομιστής επικοινωνεί με μία σχεσιακή βάση δεδομένων, η οποία βρίσκεται στο ίδιο μηχάνημα, και η οποία διαχειρίζεται όλα τα δεδομένα που είναι απαραίτητα για την λειτουργία του συστήματος. Για την υλοποίηση της εφαρμογής πελάτη, επιλέξαμε React Native το οποίο είναι JavaScript framework, για την υλοποίηση του διακομιστή χρησιμοποιήσαμε Django και το Django REST Framework, τα οποία είναι βασιμένα σε Python, και για λόγους απλότητας χρησιμοποιήσαμε για την βάση δεδομένων SQLite, παρόλο που αυτό μπορεί εύκολα να αντικατασταθεί από διαφορετική τεχνολογία.

4.1.2 React Native

Για την υλοποίηση της εφαρμογής πελάτη χρησιμοποιήσαμε την βιβλιοθήκη React Native, η οποία επιτρέπει την κατασκευή mobile cross-platform εφαρμογών μέσω JavaScript. Η βιβλιοθήκη αυτή καθώς και το ReactJS στο οποίο βασίστηκε, ξεκίνησαν στην Facebook η οποία και διατηρεί σήμερα την βιβλιοθήκη μαζί με το Instagram και ένα αριθμό ανεξάρτητων προγραμματιστών. Η βιβλιοθήκη ReactJS έγινε διαθέσιμη τον Μάρτιο του 2013 ενώ το React-Native τον Ιανουάριο του 2015. Η διαφορά των δύο είναι ότι ενώ το ReactJS επιτρέπει την δημιουργία user-interface(διεπαφής χρήστη) για διαδικτυακές εφαρμογές, το React-Native χρησιμοποιεί την λογική πίσω από το ReactJS για να "μεταφράσει" JavaScript κώδικα σε native εφαρμογές για smart-phones τόσο σε Android όσο και σε iOS ή UWP. Οι τελικές εφαρμογές δεν διαφέρουν καθόλου ως προς τις εφαρμογές που υλοποιήθηκαν με την χρήση καθιερωμένων πρακτικών μέσω C++ ή Java.[11][12]

Το React-Native περιστρέφεται γύρω από την χρήση δηλωτικής λογικής μέσω αντικειμένων τα οποία ονομάζονται Components(Εξαρτήματα). Κάθε component αποτελεί ένα μικρό κομμάτι της διεπαφής χρήστη υλοποιώντας κάποιες συγκεκριμένες λειτουργίες. Δίνονται παρακάτω κάποιες βασικές έννοιες απαραίτητες για την κατανόηση του πως λειτουργεί μία εφαρμογή React-Native.

- **JSX**

  Σημαίνει JavaScript XML και είναι μία επέκταση της JavaScript που επιτρέπει την περιγραφή των components που θα συγκροτήσουν την εφαρμογή. Η χρήση JSX είναι εντελώς προαιρετική καθώς μετά την προ-επεξεργασία της καταλήγει σε κώδικα JavaScript, μπορεί επομένως κάποιος εύκολα να το αποφύγει. Η σύνταξη JSX φαίνεται παρακάτω με ένα απλό παράδειγμα όπου θέλουμε να δηλώσουμε ένα HTML στοιχείο τύπου h1.

  ```
  const element = <Text>Hello, world!</Text>;
  ```

  Η μεταβλητή element αντιπροσωπεύει ένα component τύπου Text, που χρησιμοποιείται για την προβολή κειμένου. [13]

- **Props**

  Όλα σχεδόν τα components μπορούν να προσαρμοστούν κατά την δήλωσή τους μέσω παραμέτρων που ονομάζουμε props. Οι παράμετροι αυτοί είναι διαθέσιμοι εντός των components, και μπορούν να χρησιμοποιηθούν για τον έλεγχο της όψης και συμπεριφοράς τους. Ένα component τύπου Image για παράδειγμα χρησιμοποιείται για την προβολή εικόνας, και κάποια από τα κύρια props του είναι το source, που χρησιμοποιείται για τον εντοπισμό της εικόνας που θα προβληθεί, και το style που χρησιμοποιείται για την οπτική προσαρμογή της εικόνας. Τα props ενός component τίθενται κατά την δήλωση του, και δεν μπορούν να μεταβληθούν από κώδικα εντός του component.[14]

- **State**

  Τα components είναι επίσης stateful, έχουν δηλαδή state(κατάσταση) που συμβολίζεται με ένα JavaScript αντικείμενο. Η κατάσταση ενός component χρησιμοποιείται για τον ίδιο

σκοπό που που χρησιμοποιούνταιι τα props, την προσαρμογή του component. Αντίθετα όμως με τα props, το state τίθεται αρχικά εντός του constructor και μπορεί να μεταβληθεί από τον κώδικα εντός του component, μέσω μίας μεθόδου που ονομάζεται setState().[15]

- **Style**
  Για το styling του κάθε component στο React-Native δεν χρησιμοποιείται κάποιο ξεχωριστής μορφής αρχείο όπως για παράδειγμα CSS. Αντ'αυτού χρησιμοποιούμαι επίσης JavaScript. Όλα τα components δέχονται ένα βασικό prop που ονομάζεται style και είναι ένα αντικείμενο που περιλαμβάνει μία ή περισσότερες κλάσεις στυλ που μπορούν να χρησιμοποιηθούν στο component.[16]

Το React-Native σε αντίθεση με το ReactJS δεν χρησιμοποιεί στοιχεία HTML, αλλά native components τα οποία είναι έτοιμα για χρήση χωρίς ιδιαίτερο κόπο. Τέτοια components είναι το βασικό View, ένα component που σκοπό έχει την τοποθέτηση άλλων εξαρτημάτων στην οθόνη, το Text που χρησιμοποιείται για προβολή κειμένου ή το Picker που είναι το αντίστοιχο ενός dropdown box.

4.1.3 Django

Για την υλοποίηση του back-end της εφαρμογής χρησιμοποιήθηκε το Django framework, που επιτρέπει την δημιουργία web applications μέσω της γλώσσας Python. Το Django δημοσιεύθηκε αρχικά τον Ιούλιο του 2005 και παραμένει σήμερα ένα από τα δημοφιλέστερα frameworks για την υλοποίηση database-driven σελίδων και εφαρμογών διαδικτύου. Ακολουθεί την αρχιτεκτονική model-view-controller(MVC) ή model-view-template(MVT). Η MVC αρχιτεκτονική αποτελείται από 3 κύρια αφαιρετικά στρώματα.[17][18]

- **Model(M)**
  Το M στο ακρόνυμο MVC συμβολίζει το μοντέλο(ή μοντέλα) που αντιπροσωπεύει τα δεδομένα του συστήματος. Δεν πρόκειται για τα πραγματικά δεδομένα αλλά μία περιγραφή ή προσχέδιό τους. Το μοντέλο αυτό επιτρέπει την εξαγωγή δεδομένων από την βάση δεδομένων του συστήματος, χωρίς την ενασχόληση με λεπτομέρειες που έχουν να κάνουν με την βάση που χρησιμοποιείται. Κατά αυτό τον τρόπο μπορούμε να αλλάξουμε την υποκείμενη βάση δεδομένων χωρίς ιδιαίτερο κόπο.[19]

- **View(V)**
  Το γράμμα V συμβολίζει την λέξη View και πρόκειται για την παρουσίαση του μοντέλου δεδομένων που προαναφέρθηκε. Το επίπεδο αυτό περιγράφει τον τρόπο με τον οποίο τα δεδομένα παρουσιάζονται στον χρήστη. Σε κάποιες περιπτώσεις, όπως και στην δική μας, αυτό το επίπεδο μπορεί να περιγράφει μία δομή δεδομένων αντί μίας όψης ή σελίδας. Οπότε αντί για απευθείας “κατανάλωση” των δεδομένων από τον χρήστη, μέσω μίας δομής τα δεδομένα μεταφέρονται σε άλλες εφαρμογές για περαιτέρω επεξεργασία και εν τέλει οπτικοποίηση.[19]

- **Controller(C)**

  Το τελευταίο επίπεδο που συμβολίζεται από το γράμμα C για Controller, είναι υπεύθυνο για τον έλεγχο ροής δεδομένων. Μέσω προγραμματισμένης λογικής, αποφασίζει ποια δεδομένα θα εξαχθούν από την βάση μέσω του μοντέλου. Πέραν αυτού προ-επεξεργάζεται τα δεδομένα αυτά πριν τα στείλει στην όψη(view) και είναι υπεύθυνο για την αποδοχή δεδομένων από τον χρήστη μέσω της όψης και την προώθηση αλλαγών στην βάση δεδομένων.[19]

Πέραν του Django χρησιμοποιήθηκε το Django REST Framework, μία εργαλειοθήκη για την υλοποίηση διαδικτυακών διεπαφών που περιγράφεται παρακάτω.

4.1.4 REST

Είναι ένα σύνολο σχεδιαστικών περιορισμών, για την υλοποίηση διαδικτυακών υπηρεσιών. Το ακρώνυμο προέρχεται από την ονομασία Representational State Transfer, και σκοπό έχει την προώθηση αποτελεσματικών, αξιόπιστων και επεκτάσιμων συστημάτων που μπορούν με ευκολία να επικοινωνούν μεταξύ τους.[20] Κάποια από τα χαρακτηριστικά της αρχιτεκτονικής REST είναι τα παρακάτω.

- **Διαχωρισμός πελάτη & διακομιστή**

  Στην αρχιτεκτονική REST η υλοποίηση του πελάτη και η υλοποίηση του διακομιστή μπορεί να γίνει ανεξάρτητα, χωρίς να γνωρίζουν ο ένας κάτι για τον άλλο. Αυτό σημαίνει ότι ο καθένας εκ των δύο μπορεί να αλλάξει ως προς τον τρόπο που εκτελεί τις λειτουργίες του, χωρίς να επηρεασθεί ο άλλος. Το μόνο που απαιτείται, είναι και η δύο να γνωρίζουν την μορφή των μηνυμάτων που θα ανταλλάζουν. Κατά αυτό τον τρόπο οι δύο αυτές οντότητες μπορούν να εξελιχθούν ανεξάρτητα. Επίσης, οι πελάτες μπορούν να είναι εντελώς διαφορετικές εφαρμογές κάτι που μας δίνει ελαστικότητα. Τέλος τα εξαρτήματα που συγκροτούν τον διακομιστή μπορούν να απλοποιηθούν και να τυποποιηθούν βελτιώνοντας έτσι την επεκτασιμότητα.[21]

- **Απουσία Κατάστασης (Statelessness)**

  Τα συστήματα που ακολουθούν την αρχιτεκτονική REST δεν έχουν “κατάσταση”, που σημαίνει πως ο διακομιστής δεν γνωρίζει σε ποια κατάσταση βρίσκεται ο πελάτης, και αντιστρόφως. Κατά αυτό τον τρόπο και οι δύο οντότητες καταλαβαίνουν τα μηνύματα που λαμβάνονται, χωρίς την ανάγκη πληροφοριών σχετικά με προηγούμενες ανταλλαγές μηνυμάτων. Ο περιορισμός αυτός επιβάλλεται μέσω της χρήσης πόρων αντί της χρήσης εντολών. Όλα τα μηνύματα προς τον διακομιστή περιγράφουν επομένως τέτοιους πόρους(ή κάποια χαρακτηριστικά τους) έναντι κατηγορηματικών διαδικασιών για το τι πρέπει να κάνει ο διακομιστής. Το κάθε μήνυμα περιέχει κάποια πληροφορία σχετικά με κάποιον

πόρο, και το πως θα γίνουν τυχόν απαραίτητες αλλαγές, είναι καθαρά θέμα υλοιποίησης του διακομιστή.[21]

- **Επικοινωνία πελάτη & διακομιστή**

  Η επικοινωνία στην αρχιτεκτονική REST, αποτελείται από αιτήματα που στέλνουν οι πελάτες προς τον διακομιστή, και στα οποία ο διακομιστής δίνει απαντήσεις. Το κάθε αίτημα αποτελείται από ένα ρήμα που αντιπροσωπεύει την αλληλεπίδραση με τον πόρο που αφορά.[22] Τέτοια είναι:

  - **GET** – ανάκτηση πόρου
  - **POST –** δημιουργία πόρου
  - **PUT –** τροποποίηση πόρου
  - **PATCH** – μερική τροποποίηση πόρου
  - **DELETE** – διαγραφή πόρου

  Το αίτημα περιλαμβάνει επίσης πληροφορίες σχετικά με το ίδιο το αίτημα(στο header), το μονοπάτι που χρησιμοποιείται για τον εντοπισμό του πόρου, και προαιρετικά το “σώμα” του αιτήματος όπου περιλαμβάνονται δεδομένα που καταναλώνονται για να ολοκληρωθεί το αίτημα. Οι απαντήσεις στα αιτήματα συνοδεύονται από ένα status code (κωδικό κατάστασης) το οποίο είναι ένας τριψήφιος δεκαδικός αριθμός, που ενδεικνύει το αποτέλεσμα του αιτήματος. Η λίστα κωδικών είναι αρκετά μεγάλη οπότε παρατίθενται παρακάτω οι ευρύτερες κατηγορίες των κωδικών, οι οποίες συμβολίζονται με το πρώτο ψηφίο του αριθμού.[22]

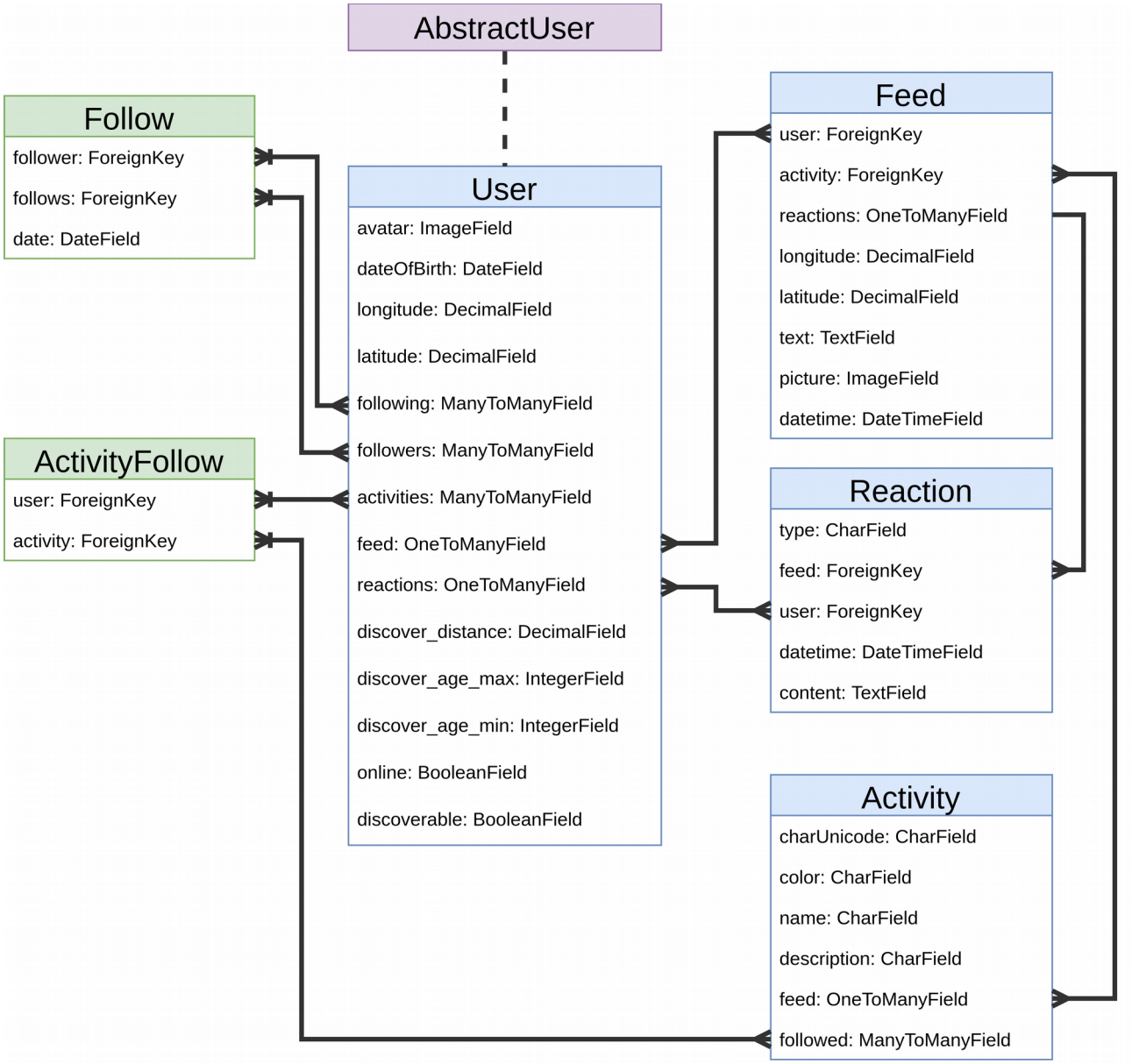
  - **1XX –** Πληροφορία
  - **2XX –** Επιτυχία
  - **3XX –** Ανακατεύθυνση
  - **4XX –** Λάθος Πελάτη
  - **5XX –** Λάθος Διακομιστή

Όπως προαναφέρθηκε, για την υλοποίηση της διεπαφής(API – Application Programming Interface) μεταξύ του διακομιστή και των πελατών, χρησιμοποιήσαμε πέρα από το Django, την εργαλειοθήκη Django REST Framework. Η βιβλιοθήκη προσφέρει διάφορες κλάσεις, μεθόδους και λειτουργίες που μπορούν να χρησιμοποιηθούν out-of-the-box για την υλοποίηση διεπαφών που ακολουθούν το πρότυπο REST. Τέτοια είναι η δυνατότητα περιήγησης του API μέσω κάποιου browser που κάνει την διαδικασία υλοποίησης λιγότερο επίπονη για τους προγραμματιστές, πακέτα πιστοποίησης συμπεριλαμβανομένων των OAuth1 και OAuth2 και κλάσεις σειριοποίησης που μπορούν να καταναλώσουν τυπικά αρκετό χρόνο για να υλοποιηθούν. Δίνονται παρακάτω λεπτομέρειες σχετικά με την χρήση της εργαλειοθήκης κατά την περιγραφή του back-end.

4.2 Server(Διακομιστής) & API

4.2.1 Μοντέλα Δεδομένων

Ακολουθεί ένα διάγραμμα ER(οντοτήτων-σχέσεων) του μοντέλου δεδομένων που περιγράφει τις διάφορε δομές δεδομένων που χρησιμοποιούνται από τον διακομιστή. Το παρακάτω μοντέλο μεταφράζεται από το Django σε λογική SQL η οποία χρησιμοποιείται για την αποθήκευση, διαγραφή ή αλλαγή δεδομένων σε μία βάση SQLite. Η επιλογή χρήσης SQLite έγινε για λόγους απλότητας, αφού μας επιτρέπει να διατηρήσουμε τα δεδομένα μας σε ένα μόνο αρχείο και είναι η default βάση που διατίθεται από το Django χωρίς επιπρόσθετες προσπαθειες.



Σχήμα 4.2 Μοντέλο Δεδομένων

- **User**

  Μοντέλο που χρησιμοποιείται για την αποθήκευση του προφίλ ενός χρήστη. Κληρονομεί από την κλάση AbstractUser του Django, η οποία περιλαμβάνει βασικά πεδία χρήστη όπως όνομα, επίθετο, username, κωδικό πρόσβασης κλπ. Περαιτέρω πεδία:

  - **avatar:** Εικόνα προφίλ του χρήση.
  - **dateOfBirth:** Ημερομηνία γεννήσεως του χρήστη.
  - **longitude:** Γεωγραφικό μήκος της τελευταίας γνωστής τοποθεσίας του χρήστη.
  - **latitude:** Γεωγραφικό πλάτος της τελευταίας γνωστής τοποθεσίας του χρήστη.
  - **online:** Κατάσταση του διακόπτη χρήσης για τον χρήστη.
  - **discoverable:** Κατάσταση του διακόπτη τοποθεσίας για τον χρήστη.
  - **discover_distance:** Μέγιστη απόσταση λειτουργίας Discover του χρήστη(χιλιόμετρα).
  - **discover_age_max:** Μέγιστη ηλικία λειτουργίας Discover του χρήστη.
  - **discover_age_min**: Ελάχιστη ηλικία λειτουργίας Discover του χρήστη.
  - **followers:** Λίστα προφίλ χρηστών οι οποίοι ακολουθούν τον χρήστη.
  - **following**: Λίστα προφίλ χρηστών τους οποίους ακολουθεί ο χρήστης.
  - **activities:** Λίστα δραστηριοτήτων τις οποίες ακολουθεί ο χρήσης.
  - **feed:** Λίστα αντικειμένων feed που δημοσιεύθηκαν από τον χρήστη.
  - **reactions:** Λίστα αντιδράσεων(reactions) που δημοσιεύθηκαν από τον χρήση σε αντικείμενα feed.

- **Activity**

  Μοντέλο που χρησιμοποιείται για την αποθήκευση δεδομένων για μία συγκεκριμένη δραστηριότητα. Πεδία:

  - **charUnicode:** Χαρακτήρας που χρησιμοποιείται για την απεικόνιση ενός εικονιδίου που αντιστοιχεί στην δραστηριότητα.
  - **color:** 6 χαρακτήρες (base16) που χρησιμοποιούνται για τον συμβολισμό ενός χρώματος που αντιστοιχεί στην δραστηριότητα.
  - **name:** Όνομα δραστηριότητας.
  - **description:** Μία μικρού μήκους περιγραφή της δραστηριότητας.
  - **feed:** Λίστα αντικειμένων feed σχετικά με την δραστηριότητα.

- **followed:** Λίστα χρηστών που ακολουθούν την δραστηριότητα.

- **Feed**

  Μοντέλο που αντιπροσωπεύει μία δημοσίευση (αντικείμενο Feed) που έγινε από κάποιον χρήστη.

  - **longitude:** Γεωγραφικό μήκος της τοποθεσίας όπου έγινε η δημοσίευση.
  - **latitude:** Γεωγραφικό πλάτος της τοποθεσίας όπου έγινε η δημοσίευση.
  - **text:** Το περιεχόμενο κείμενου που συνοδεύει την δημοσίευση.
  - **picture:** Το περιεχόμενο εικόνας που συνοδεύει την δημοσίευση.
  - **datetime:** Η ημερομηνία και ώρα δημοσίευσης.
  - **user:** Ο χρήστης που έκανε την δημοσίευση.
  - **activity:** Η δραστηριότητα που αφορά την συγκεκριμένη δημοσίευση.
  - **reactions:** Μία λίστα αντιδράσεων(αντικειμένων reaction) από τους χρήστες.

- **Reaction**

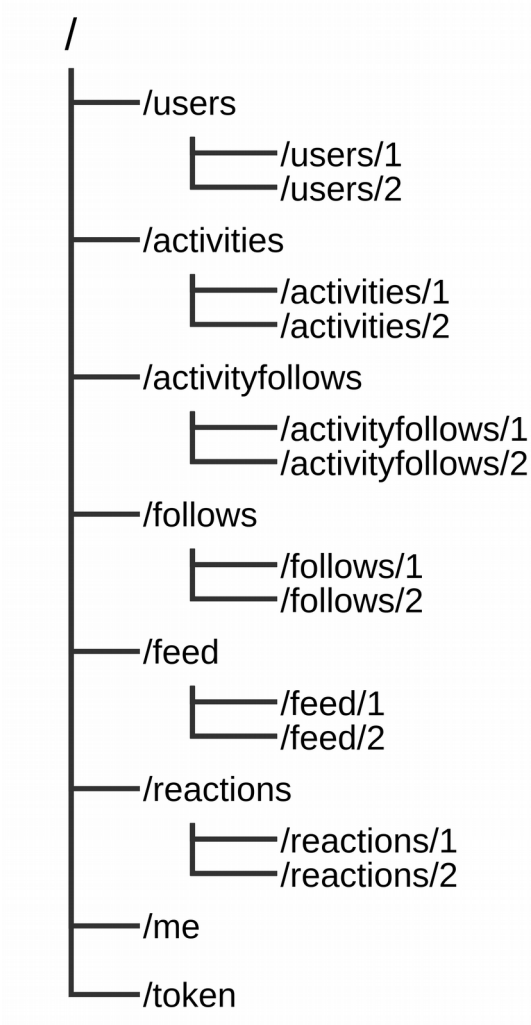
  Μοντέλο που αντιπροσωπεύει μία αντίδραση (αντικείμενο Reaction) κάποιου χρήστη προς μία συγκεκριμένη δημοσίευση.

  - **type:** Μικρού μήκους ακολουθία χαρακτήρων που παίρνει δύο μόνο τιμές, “Like” ή “Comment”, και χρησιμοποιείται για να διακρίνουμε τις δύο αυτές αντιδράσεις.
  - **feed:** Το αντικείμενο Feed το οποίο η συγκεκριμένη αντίδραση.
  - **user:** Ο χρήστης που δημοσίευσε την αντίδραση.
  - **datetime:** Η ημερομηνία και ώρα δημοσίευσης της αντίδρασης.
  - **content:** Στην περίπτωση που πρόκειται για αντίδραση τύπου “Comment”, το πεδίο αυτό κρατάει το περιεχόμενο του σχολίου.

Τα μοντέλα Follow και ActivityFollow είναι ενδιάμεσες κλάσεις, που επιτρέπουν την διαχείριση, και χρήση των σχεσιακών πεδίων μεταξύ μοντέλων. Θα μπορούσαν να παραλειφθούν χωρίς συνέπειες στην σχεσιακή λογική μεταξύ χρηστών ή μεταξύ χρηστών και δραστηριοτήτων. Το πεδίο **date** στο μοντέλο Follow επιτρέπει την αποθήκευση της ημερομηνίας κατά την οποία εγκαθιδρύθηκε μία σχέση Follow μεταξύ δύο χρηστών. Πότε δηλαδή άρχισε κάποιος χρήστης να ακολουθεί έναν άλλο.

4.2.2 Τοπολογία API

Παρουσιάζουμε εδώ τις όψεις(views) που υλοποιούν το API του διακομιστή, και εξερευνούμε την URL τοπολογία του. Η δομή της τοπολογίας παρουσιάζεται με ένα συνοπτικό διάγραμμα παρακάτω, και κάθε άκρο της αναλύεται με περισσότερη λεπτομέρεια παρακάτω.



Σχήμα 4.3 Τοπολογία API

Αν και το Django REST Framework προσφέρει browsable API, που διευκολύνει σημαντικά την διαδικασία υλοποίησης, θα δώσουμε σημασία στην μορφή απαντήσεων JSON, η οποία και θα χρησιμοποιείται από την εφαρμογή πελάτη. Όπως θα γίνει εμφανές παρακάτω η τοπολογία είναι σε ευθυγράμμιση με τα ίδια τα μοντέλα. Αυτό οφείλεται στην απλότητα που προσφέρει το Django REST Framework.

Κατά την υλοποίηση δίνεται η επιλογή απόρροιας των όψεων από τα ίδια τα μοντέλα, και στη συνέχεια της απόρροιας των διευθύνσεων URL από τις όψεις αυτές. Κατά αυτό τον τρόπο έχουμε μία αξιόλογη βάση για το API με ελάχιστο κόπο, στην οποία μπορούμε να κτίσουμε περαιτέρω λειτουργικότητα και να προσαρμόσουμε την λογική. Παρόλο λοιπόν που όλες οι όψεις και τα άκρα του API έχουν υποστεί αλλαγές, είναι εύκολα κατανοητές αν κάποιος γνωρίζει τα μοντέλα δεδομένων που χρησιμοποιήθηκαν.

Παρακάτω αναλύονται όλες οι όψεις, σχετικά με τα πεδία που εκθέτουν, τις λειτουργίες που επιτρέπουν και τον σκοπό που εξυπηρετούν.

4.2.3 Όψεις API

- **/users/**
  - **Σκοπός:** Υλοποιεί την λειτουργία Discover, επιτρέποντας την εξερεύνηση προφίλ χρηστών εντός συγκεκριμένης γεωγραφικής εμβέλειας.
  - **Πεδία:** url, username, first_name, last_name, avatar, dateOfBirth, latitude, longitude, following, followers, activities
  - **Μεθόδοι:**
    - **GET**
      Επιστρέφει μία λίστα αντικειμένων User, καθένα εκ των οποίων αντιστοιχεί σε ένα χρήστη. Οι χρήστες αυτοί έχουν τους διακόπτες τοποθεσίας και κατάστασης ενεργοποιημένους, βρίσκονται εντός της εμβέλειας Discover του χρήστη που κάνει την κλήση, ο χρήστης βρίσκεται εντός της δικής τους εμβέλειας και ο χρήστης που κάνει την κλήση **δεν** τους ακολουθεί.
- **/users/{id}/**
  - **Σκοπός:** Επιτρέπει την ανάκτηση του δεδομένων για το προφίλ ενός συγκεκριμένουχρήστη, όπου id είναι το αναγνωριστικό του εν λόγω χρήστη.
  - **Πεδία:** url, username, first_name, last_name, avatar, dateOfBirth, latitude, longitude, following, followers, activities
  - **Μεθόδοι:**
    - **GET**

      Επιστρέφει το αντικείμενο User, που αντιστοιχεί στον χρήστη με αναγνωριστικό id.

Παράδειγμα λίστας αντικειμένων User:


```
[
    {
        "url": "http://localhost:8000/users/15/",
        "username": "zenios",
        "first_name": "Zenios",
        "last_name": "Zeniou",
        "avatar": "http://localhost:8000/media/users/IMG_20171209_120631_Y9rHlHD.jpg",
        "dateOfBirth": "1991-08-30",
        "latitude": "47.546300",
        "longitude": "7.619048",
        "following": [
            {
                "url": "http://localhost:8000/follows/62/",
                "follower": "http://localhost:8000/users/15/",
                "followee": "http://localhost:8000/users/17/",
                "date": "2018-02-11"
            }
        ],
```

```
        "followers": [
            {
                "url": "http://localhost:8000/follows/50/",
                "follower": "http://localhost:8000/users/22/",
                "followee": "http://localhost:8000/users/15/",
                "date": "2018-02-09"
            }
        ],
        "activities": [
            {
                "url": "http://localhost:8000/activityfollows/27/",
                "user": "http://localhost:8000/users/15/",
                "activity": "http://localhost:8000/activities/96/"
            }
        ]
    },
    { ... },
    { ... }
]
```


- **/activities/**
    - **Σκοπός:** Υλοποιεί την λειτουργία εξερεύνησης νέων δραστηριοτήτων
    - **Πεδία:** url, charUnicode, color, name, description, followed
    - **Μεθόδοι:**
        - **GET**

          Επιστρέφει μία λίστα δραστηριοτήτων τις οποίες ο χρήστης που κάνει το αίτημα, **δεν** ακολουθεί.
- **/activities/{id}/**
    - **Σκοπός:** Επιτρέπει την ανάκτηση δεδομένων για μία συγκεκριμένη δραστηριοτήτων.
    - **Πεδία:** url, charUnicode, color, name, description, followed
    - **Μεθόδοι:**
        - **GET**

          Επιστρέφει το αντικείμενο δραστηριότητας στην οποία αντιστοιχεί το αναγνωριστικό id.

Παράδειγμα λίστας αντικειμένων User:


```
[
    {
        "url": "http://localhost:8000/activities/106/",
        "charUnicode": "",
        "color": "#168C2F",
        "name": "Ballet",
        "description": "Soaring like a swan.",
        "followed": [
            "http://localhost:8000/users/20/"
        ]
```

```
    },
    { ... },
    { ... },
    { ... }
]
```

- **/activityfollows/**
  - **Σκοπός:** Υλοποιεί την λειτουργία Follow δραστηριοτήτων για έναν χρήστη.
  - **Πεδία:** user, activity
  - **Μεθόδοι:**
    - **POST**

      Δημιουργία ενός νέου αντικειμένου activity follow μεταξύ του χρήστη και της δραστηριότητας που περιλαμβάνονται στο σώμα του αιτήματος
- **/activityfollows/{id}**
  - **Σκοπός:** Υλοποιεί την λειτουργία Unfollow δραστηριοτήτων για έναν χρήστη.
  - **Πεδία:** url, user, activity
  - **Μεθόδοι:**
    - **GET**

      Ανάκτηση του αντικειμένου ActivityFollow στο οποίο αντιστοιχεί το αναγνωριστικό id.
    - **DELETE**

      Διαγραφή του αντικειμένου ActivityFollow στο οποίο αντιστοιχεί το αναγνωριστικό id.

Παράδειγμα αντικειμένου ActivityFollow:

```
{
    "url": "http://localhost:8000/activityfollows/27/",
    "user": "http://localhost:8000/users/15/",
    "activity": "http://localhost:8000/activities/96/"
}
```

- **/follows/**
  - **Σκοπός:** Υλοποιεί την λειτουργία Follow μεταξύ των χρηστών.
  - **Πεδία:** follower, followee

- **Μεθόδοι:**
  - **POST**

    Δημιουργία ενός νέου αντικειμένου follow μεταξύ του χρήστη follower και του χρήστη followee που περιλαμβάνονται στο σώμα του αιτήματος.

- **/follows/{id}**
  - **Σκοπός:** Υλοποιεί την λειτουργία Unfollow μεταξύ των χρηστών
  - **Πεδία:** url, follower, followee, date
  - **Μεθόδοι:**
    - **GET**

      Ανάκτηση του αντικειμένου Follow στο οποίο αντιστοιχεί το αναγνωριστικό id.
    - **DELETE**

      Διαγραφή του αντικειμένου Follow στο οποίο αντιστοιχεί το αναγνωριστικό id.

Παράδειγμα αντικειμένου Follow:

```
{
  "url": "http://localhost:8000/follows/10/",
  "follower": "http://localhost:8000/users/16/",
  "followee": "http://localhost:8000/users/17/",
  "date": "2017-10-30"
}
```

- **/feed/**
  - **Σκοπός:** Υλοποιεί την λειτουργία εξερεύνησης Feed για έναν συγκεκριμένο χρήστη.
  - **Πεδία:** url, user, activity, datetime, latitude, longitude, text, picture, reactions
  - **Μεθόδοι:**
    - **GET**

      Επιστρέφει μία λίστα αντικειμένων Feed, δεδομένου ότι ο χρήστης που κάνει το αίτημα έχει τον διακόπτη κατάστασης ενεργοποιημένο, ακολουθεί και ακολουθείται από τον χρήστη που δημοσίευσε το κάθε αντικειμένου Feed στην λίστα,.
    - **POST**

      Δημιουργεί ένα νέο αντικείμενο Feed χρησιμοποιώντας τα πεδία που περιλαμβάνονται στο σώμα του αιτήματος.

- **/feed/{id}/**

- **Σκοπός:** Επιτρέπει την ανάκτηση, διαγραφή και ενημέρωση ενός αντικειμένου Feed.
- **Πεδία:** url, user, activity, datetime, latitude, longitude, text, picture, reactions
- **Μεθόδοι:**
  - **GET**

    Ανάκτηση του αντικειμένου Feed στο οποίο αντιστοιχεί το αναγνωριστικό id.
  - **DELETE**

    Διαγραφή του αντικειμένου Feed στο οποίο αντιστοιχεί το αναγνωριστικό id.
  - **PUT**

    Ενημέρωση του αντικειμένου Feed στο οποίο αντιστοιχεί το αναγνωριστικό id, χρησιμοποιώντας τα πεδία στο σώμα του αιτήματος.
  - **PATCH**

    Ενημέρωση του αντικειμένου Feed στο οποίο αντιστοιχεί το αναγνωριστικό id, χρησιμοποιώντας τα πεδία στο σώμα του αιτήματος(κάποια από τα πεδία μπορούν να απουσιάζουν).

Παράδειγμα αντικειμένου Feed:


```
{
    "url": "http://localhost:8000/feed/15/",
    "user": "http://localhost:8000/users/21/",
    "activity": "http://localhost:8000/activities/107/",
    "datetime": "2018-02-07T11:34:53.485175Z",
    "latitude": "50.127859",
    "longitude": "14.466141",
    "text": "Pancakes anyone?",
    "picture": null,
    "reactions": [
        {
            "url": "http://localhost:8000/reactions/27/",
            "user": "http://localhost:8000/users/21/",
            "feed": "http://localhost:8000/feed/15/",
            "datetime": "2018-02-07T11:35:11.397755Z",
            "content": null,
            "type": "Like"
        }
    ]
}
```


- **/reactions/**
  - **Σκοπός:** Επιτρέπει την δημοσίευση “αντιδράσεων” από τους χρήστες, ως προς κάποιο αντικείμενο Feed.
  - **Πεδία:** url, user, feed, datetime, content, type

- **Μεθόδοι:**
  - **POST**

    Δημιουργία ενός νέου αντικειμένου Reaction από τον χρήστη user ως προς το αντικείμενο Feed, χρησιμοποιώντας τα πεδία που περιλαμβάνονται στο σώμα του αιτήματος.

- **/reactions/{id}/**
  - **Σκοπός:** Επιτρέπει την ανάκτηση και διαγραφή ενός αντικειμένου Reaction
  - **Πεδία:** url, user, feed, datetime, content, type
  - **Μεθόδοι:**
    - **GET**

      Ανάκτηση του αντικειμένου Reaction στο οποίο αντιστοιχεί το αναγνωριστικό id.
    - **DELETE**

      Διαγραφή του αντικειμένου Reaction στο οποίο αντιστοιχεί το αναγνωριστικό id.

Παράδειγμα αντικειμένου Feed:

```
{
    "url": "http://localhost:8000/reactions/23/",
    "user": "http://localhost:8000/users/15/",
    "feed": "http://localhost:8000/feed/14/",
    "datetime": "2018-02-06T21:50:45.704220Z",
    "content": "Some comment text.",
    "type": "Comment"
}
```

- **/token/**
  - **Σκοπός:** Επιτρέπει σε έναν χρήστη να ανακτήσει ένα Token το οποίο θα χρησιμοποιήσει στην συνέχεια για όλες τα αιτήματα προς τον διακομιστή.
  - **Πεδία:** username, password
  - **Μεθόδοι:**
    - **POST**

      Προμηθεύοντας τα πεδία username και password στο σώμα του αιτήματος, ένας χρήστης/πελάτης μπορεί να πιστοποιήσει την ταυτότητά του και να λάβει σαν απάντηση ένα token που χρησιμοποιεί για μετέπειτα αιτήματα.

```
{
    "username": "zenios",
    "password": "mypassword012345"
}
```

- **/me/**
  - **Σκοπός:** Επιτρέπει σε έναν πελάτη/χρήστη να δημιουργήσει, να ανακτήσει και να ενημερώσει το προφίλ του.
  - **Πεδία:** url, id, username, password, first_name, last_name, email, avatar, dateOfBirth, latitude, longitude, following, followers, activities, discover_distance, discover_age_max, discover_age_min, feed, discoverable, online, reactions
  - **Μεθόδοι:**
    - **POST**

      Δημιουργία προφίλ χρήστη, χρησιμοποιώντας τα πεδία απαραίτητα πεδία από το σώμα του αιτήματος.
    - **GET**

      Ανάκτηση προφίλ χρήστη.
    - **PATCH**

      Ενημέρωση του προφίλ χρήστη χρησιμοποιώντας μόνο όσα πεδία βρίσκονται στο σώμα του αιτήματος.

4.2.3 Πιστοποίηση Χρήστη

Για την πιστοποίηση χρήστη χρησιμοποιούμε κέρματα(token authentication). Αυτό σημαίνει πως για την πιστοποίηση κάθε αιτήματος, το αίτημα πρέπει να εμπεριέχει μία ακολουθία χαρακτήρων(το κέρμα) στο header του. Για να προμηθευτεί ένας πελάτης με token, πρέπει να κάνει αίτημα POST στον διακομιστή μέσω της όψης /token/. Το αίτημα αυτό θα περιλαμβάνει στο σώμα του, τα πεδία username και password, τα οποία ο διακομιστής θα επαλήθευση της ταυτότητας του χρήστη. Εάν η επαλήθευση είναι επιτυχής, ο χρήστης θα απαντήσει με ένα token το οποίο ο πελάτης θα χρησιμοποιήσει για όλα τα επόμενα αιτήματα. Το token χρησιμοποιείται στο ***Authorization*** header του αιτήματος όπως φαίνεται παρακάτω.

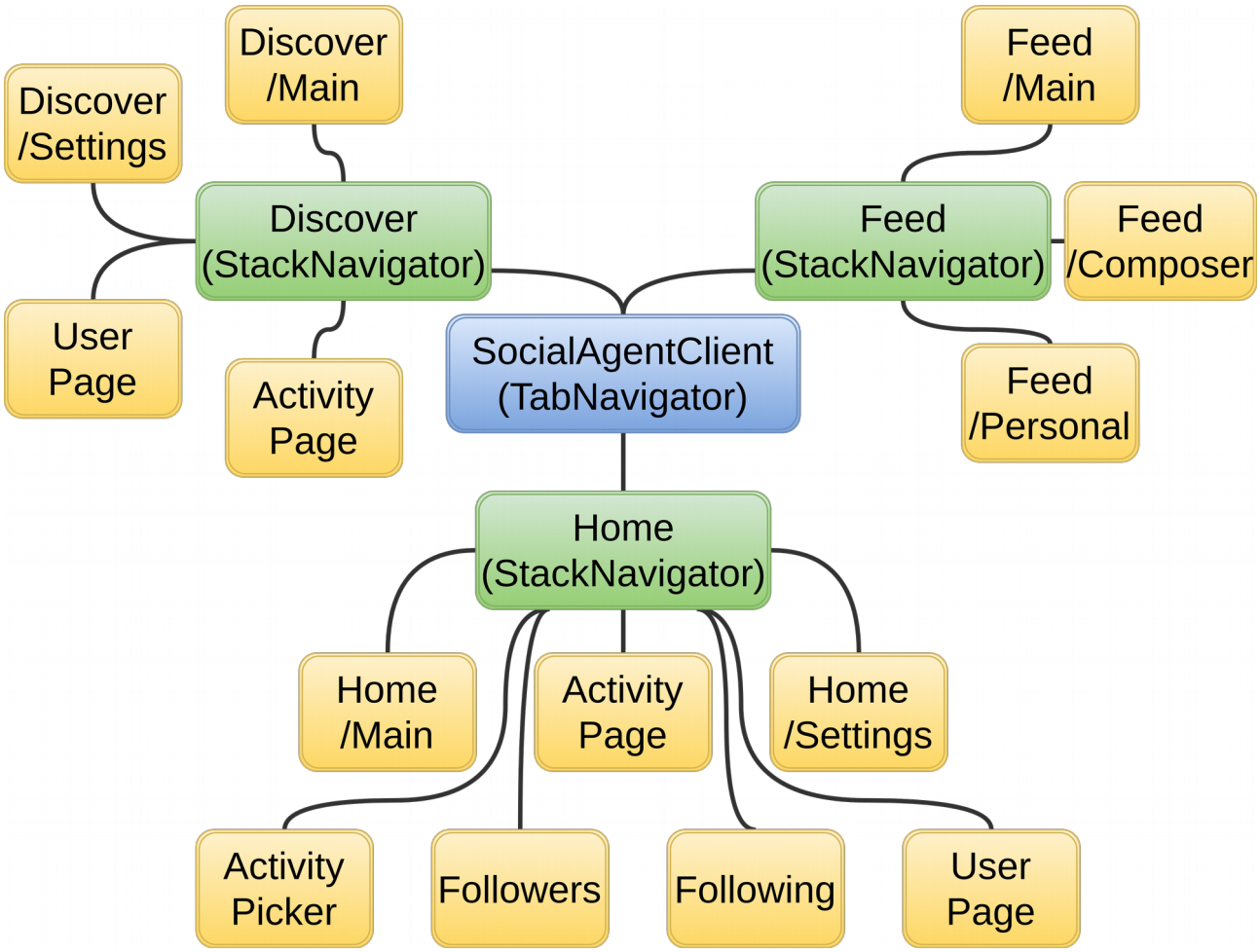
```
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

Τα κέρματα αυτά δημιουργούνται τυπικά κατά την εξυπηρέτηση του αιτήματος POST προς την όψη /token/, και έχουν περιορισμένη διάρκεια ζωής. Όταν ένα token λήξει, ο πελάτης πρέπει να ακολουθήσει εκ νέου την ίδια διαδικασία για να δημιουργηθεί ένα νέο token από τον διακομιστή και να του το προμηθεύσει. Στην δική μας περίπτωση τα κέρματος είναι μόνιμα για λόγους απλότητας της εφαρμογής πελάτη. Για την ενεργοποίηση όμως αυτής της λογικής στην πλευρά του διακομιστή, απαιτούνται ελάχιστες αλλαγές.

4.3 Εφαρμογή Πελάτη

4.3.1 Περιήγηση και Τοπολογία

Η τελική εφαρμογή που υλοποιήθηκε απαρτίζεται από 15 διαφορετικές σελίδες στις οποίες μπορεί να βρεθεί ο χρήστης. Δίνεται παρακάτω ένα διάγραμμα που περιγράφει την τοπολογία πλοήγησης της εγαρμογής.



Σχήμα 4.4 Τοπολογία Πλοήγησης

Για την κατάρτιση της παραπάνω τοπολογίας και τον έλεγχο πλοήγησης μεταξύ των σελίδων, χρησιμοποιήθηκε το πακέτο React Navigation. Το πακέτο αυτό προσφέρει κάποιες out-of-the-box λύσεις, που επιτρέπουν την πλοήγηση σε React εφαρμογές.[23] Στην δική μας περίπτωση χρησιμοποιήθηκαν δύο τέτοιες λύσεις:

- **Tab Navigator**

  Υλοποιεί έναν πλοηγό τύπου “tab”, που περιλαμβάνει έναν αριθμό σελίδων οι οποίες τοποθετούνται πλάι πλάι, εκ των οποίων μία είναι εμφανής ανά πάσα στιγμή. Ο χρήστης μπορεί να πλοηγηθεί μεταξύ των σελίδων αυτών, χρησιμοποιώντας κάποιου είδους “μπάρα”.[24] Ο πλοηγός αυτός χρησιμοποιήθηκε για την κατάτμηση της εφαρμογής σε τρία κύρια κομμάτια και την πλοήγηση μεταξύ αυτών.

- **Stack Navigator**

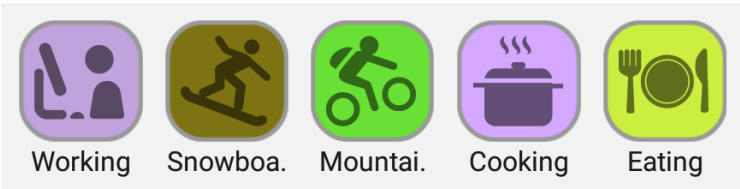
  Υλοποιεί έναν πλοηγό τύπου “σωρού”, που περιλαμβάνει έναν αριθμό σελίδων μεταξύ των οποίων ο χρήστης μπορεί να πλοηγηθεί μέσω προκαθορισμένων σημείων. Σε κάθε δράση πλοήγησης, είτε γίνεται “push” μία καινούργια σελίδα στον σωρό η οποία και είναι εμφανής στο χρήστη, είτε γίνεται “pop” η ενεργή σελίδα επιτρέποντας στο χρήστη να πλοηγηθεί στην προηγούμενη σελίδα. [25] Φωλιασμένος σε κάθε κομμάτι του πλοηγού τύπου tab που προαναφέρθηκε βρίσκεται ένας πλοηγός τύπου stack, για την περαιτέρω επέκταση των της τοπολογίας.

4.3.2 Components

Παρατίθενται εδώ τα επιμέρους components που απαρτίζουν την εφαρμογή. Για το καθένα δίνεται μία σύντομη επισκόπηση των props και του state, και περιγράφονται περιληπτικά οι λειτουργίες που υλοποιεί. Αξίζει να σημειωθεί εδώ ότι δεν αναφέρονται εξαντλητικά όλα τα πεδία των state και props, αλλά μόνο αυτά που είναι σχετικά με τις λειτουργίες υψηλού επιπέδου.

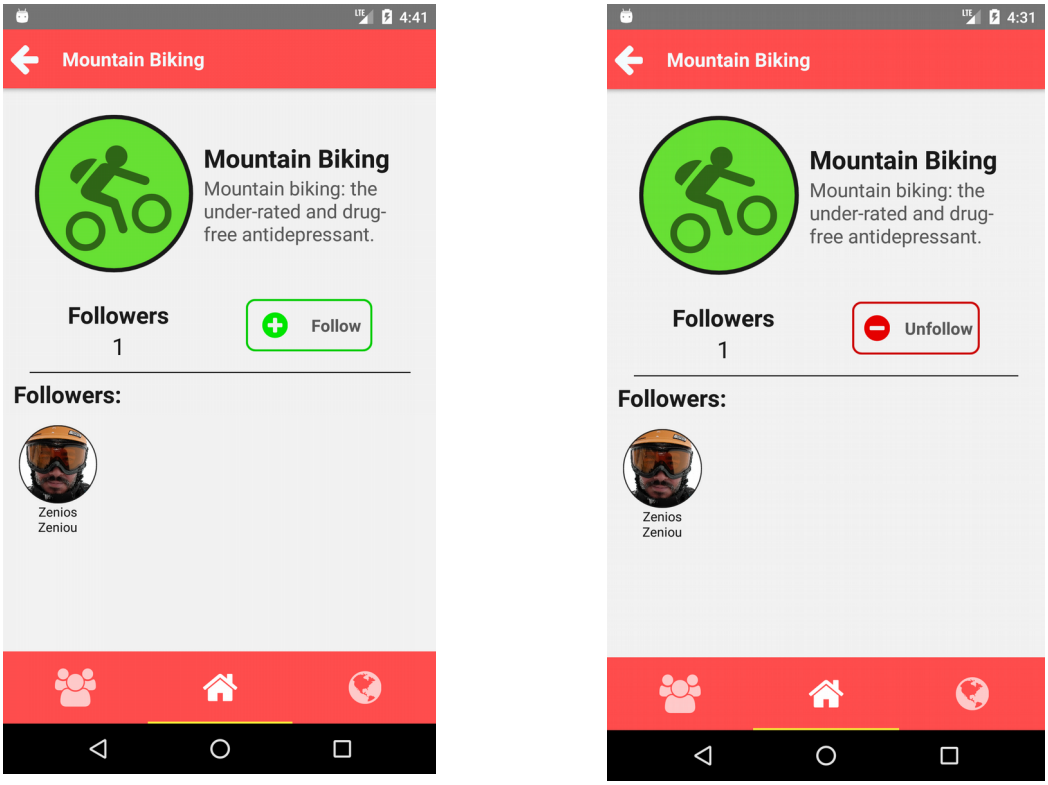
- **ActivityIcon**
  - **State**
    - activity(Object): Κρατάει τα δεδομένα της δραστηριότητας όπως αυτά λαμβάνονται από τον διακομιστή
  - **Props**
    - uri(String): Διεύθυνση URL προς τον διακομιστή, για την ανάκτηση των δεδομένων δραστηριότητας.

  Το component αυτό αποτελείται από το εικονίδιο το οποίο συμβολίζει μία δραστηριότητα και το όνομα της δραστηριότητας. Όταν ο χρήστης αγγίξει το εικονίδιο, γίνεται πλοήγηση προς την σελίδα(ActivityPage) της δραστηριότητας.



Σχήμα 4.5 Πέντε components τύπου ActivityIcon

- **ActivityPage**
  - **State**
    - followed(Boolean): Είναι True εάν ο τρέχων χρήστης ακολουθεί την δραστηριότητα, διαφορετικά είναι False.
  - **Props**
    - activity(Object): Κρατάει τα δεδομένα της δραστηριότητας όπως αυτά λαμβάνονται από τον διακομιστή.



Σχήμα 4.6 Σελίδα ActivityPage

Σελίδα δραστηριότητας. Αποτελείται από το εικονίδιο της δραστηριότητας, το όνομα και περιγραφή της, έναν μετρητή για το πόσοι χρήστες την ακολουθούν και εικονίδια αυτών των χρηστών(UserIcon). Τέλος ένα κουμπί Follow/Unfollow που επιτρέπει στον χρήστη να ακολουθήσει την εν λόγω δραστηριότητα ή να σταματήσει να την ακολουθεί.

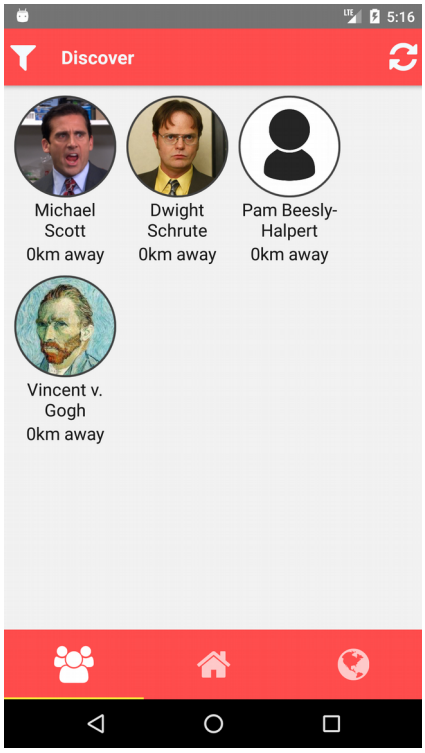
- **ActivityPicker**

Σχήμα 4.7 Σελίδα ActivityPicker

- **State**
  - activities(Array<Object>): Λίστα που περιέχει όλα τα αντικείμενα δραστηριοτήτων που δεν ακολουθεί ο χρήστης, όπως ανακτάται από τον διακομιστή.

Μία σελίδα που αποτελείται από components τύπου ActivityIcon, τα οποία αντιπροσωπεύουν όλες τις δραστηριότητες που δεν ακολουθεί ο χρήστης. Η σελίδα αυτή εξυπηρετεί την απαίτηση ο χρήστης να μπορεί να ανακαλύψει και να ακολουθήσει νέες δραστηριότητες.

- **Discover / Main**

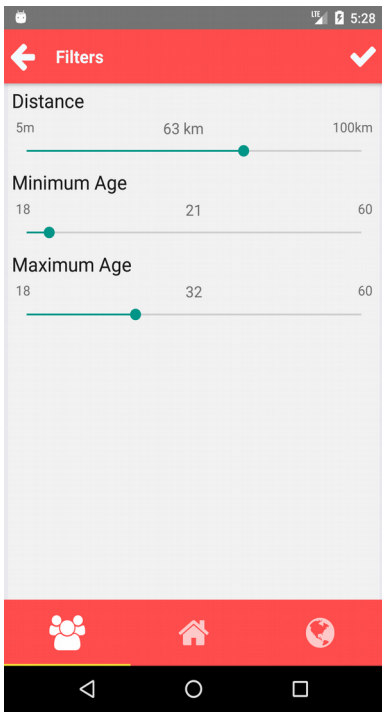


Σχήμα 4.8 Σελίδα Discover/Main

- **State**
  - people(Array<String>): Λίστα που περιέχει διευθύνσεις URL προς τον διακομιστή, για την ανάκτηση προφίλ χρηστών.

Μία σελίδα που υλοποιεί την λειτουργία Discover και αποτελείται από components τύπου UserIconLarge, τα οποία αντιπροσωπεύουν όλους τους χρήστες που δεν ακολουθεί ο χρήστης. Τα προφίλ αυτών των χρηστών τηρούν τις απαιτήσεις της λειτουργίας Discover όπως περιγράφηκαν στο υποκεφάλαιο 4.2.3. Η λίστα αυτή μπορεί να ανανεωθεί μέσω του συμβόλου ανανέωσης στην πάνω δεξιά γωνία της σελίδα. Ο χρήστης μπορεί να πατήσει το εικονίδιο στην πάνω αριστερά γωνία της σελίδας, και να μεταβεί στην σελίδα Discover/Settings όπου έχει την δυνατότητα να θέσει τα φίλτρα για την λειτουργία Discover.

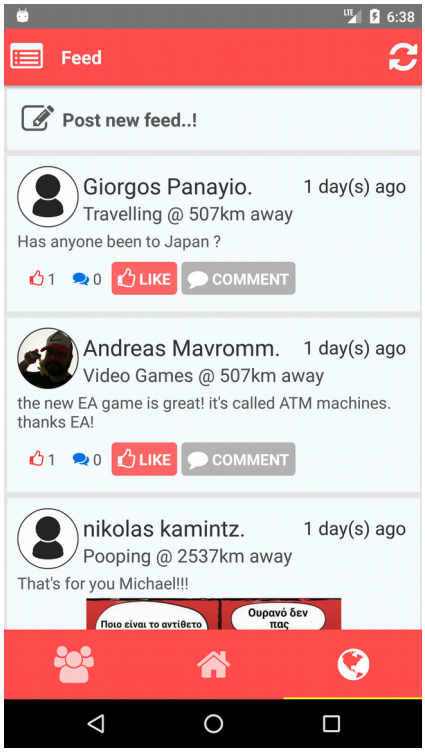
- **Discover / Settings**



Σχήμα 4.9 Σελίδα Discover/Settings

- **State**
  - discover_distance(Number): Φίλτρο απόστασης στην οποία είναι ενεργή η λειτουργία Discover.
  - discover_age_min(Number): Φίλτρο ελάχιστης ηλικίας χρηστών για την λειτουργία Discover.
  - discover_age_max(Number): Φίλτρο μέγιστης ηλικίας χρηστών για την λειτουργία Discover.
- Η σελίδα αυτή επιτρέπει στον χρήστη να θέσει τα φίλτρα για την λειτουργία Discover.

- **Feed / Main**

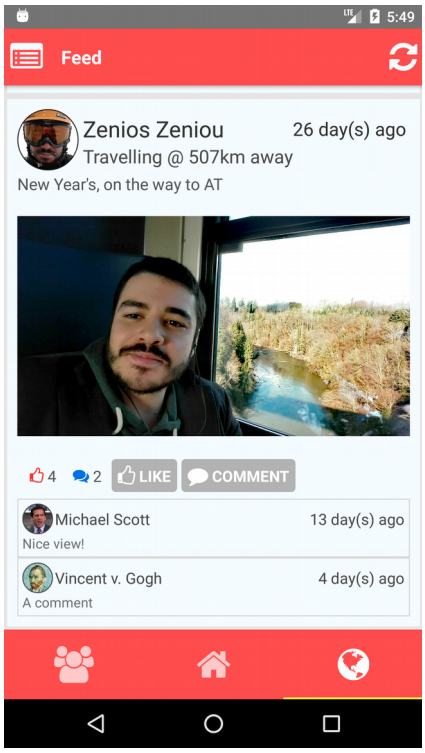


Σχήμα 4.10 Σελίδα Feed/Main

- **State**
  - feed_list(Array<Object>): Λίστα αντικειμένων Feed όπως αυτή ανακτάται από τον διακομιστή.

Η σελίδα αυτή υλοποιεί την λειτουργία Feed η οποία επιτρέπει στον χρήστη να δει και να αντιδράσει με τις δημοσιεύσεις των χρηστών εντός του κύκλου του. Η λίστα των δημοσιεύσεων μπορεί να ανανεωθεί μέσω του συμβόλου ανανέωσης στην πάνω δεξιά γωνία της σελίδα. Το εικονίδιο στην πάνω δεξιά γωνία της σελίδας, επιτρέπει στον χρήστη να πλοηγηθεί στην σελίδα Feed/FeedPersonal. Οι δημοσιεύσεις που υλοποιούνται μέσω components FeedItem τοποθετούνται σε ένα component τύπου ScrollVIew που επιτρέπει κύλιση προς τα κάτω. Στην αρχή της λίστας βρίσκεται πάντα ένα sub-component που επιτρέπει την πλοήγηση στην σελίδα Feed/FeedComposer, όπου είναι δυνατή η δημοσίευση νέου αντικειμένου Feed.

- **FeedItem**

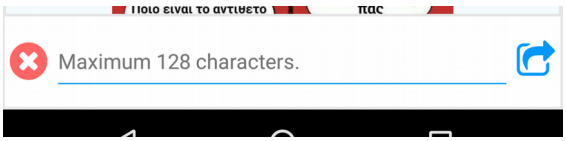


Σχήμα 4.11 Component τύπου FeedItem

- **State**
  - user(Object): Αντικείμενο που περιέχει όλα τα δεδομένα για τον χρήστη που έκανε την δημοσίευση.
  - activity(Object): Αντικείμενο που περιέχει όλα τα δεδομένα για την δραστηριότητα που αφορά η δημοσίευση.
  - latitude(Number): Γεωγραφικό πλάτος του τρέχων χρήστη.
  - longitude(Number): Γεωγραφικό μήκος του τρέχων χρήστη.
  - liked(Boolean): Είναι True στην περίπτωση που ο χρήστης έχει αντιδράσει μέσω “Like” με την δημοσίευση, αλλιώς False.

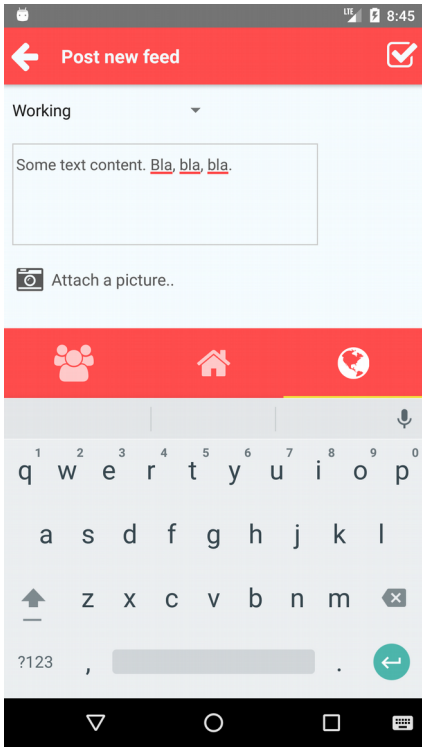
- **Props**
  - feed(Object): Αντικείμενο που περιέχει όλες τις πληροφορίες σχετικά με την δημοσίευση, όπως αυτό ανακτάται από τον διακομιστή.

Component για την οπτικοποίηση ενός αντικειμένου Feed(δημοσίευσης). Το άνω κομμάτι του component περιλαμβάνει την φωτογραφία “προφίλ” του χρήστη που έκανε την δημοσίευση, το όνομα και επίθετό του, τον χρόνο που μεσολάβησε από την δημοσίευση μέχρι την παρούσα στιγμή, το όνομα σχετικής δραστηριότητας και την απόσταση από την τοποθεσία που έγινε η δημοσίευση. Στο κέντρο βρίσκεται το περιεχόμενο της δημοσίευσης, το οποίο περιλαμβάνει απαραίτητα ένα κείμενο μήκους εως και 200 χαρακτήρων κείμενο και ίσως μία συνημμένη εικόνα. Ακολουθεί στο κάτω μέρος η “μπάρα” αντιδράσεων που περιλαμβάνει, τον μετρητή αντιδράσεων “Like”, τον μετρητή αντιδράσεων “Comment”, και τα κουμπιά Like και Comment που επιτρέπουν τις αντίστοιχες λειτουργίες. Για την δημοσίευση ενός σχολίων(comment), όταν ο χρηστής πατήσει το κουμπί comment, μία μικρή μπάρα εμφανίζεται στο κάτω μέρος της σελίδας για τον σκοπό αυτό.



Τέλος, έχουμε την λίστα σχολίων με τους αντίστοιχους χρήστες και τον χρόνο δημοσίευσης.

- **Feed / Composer**

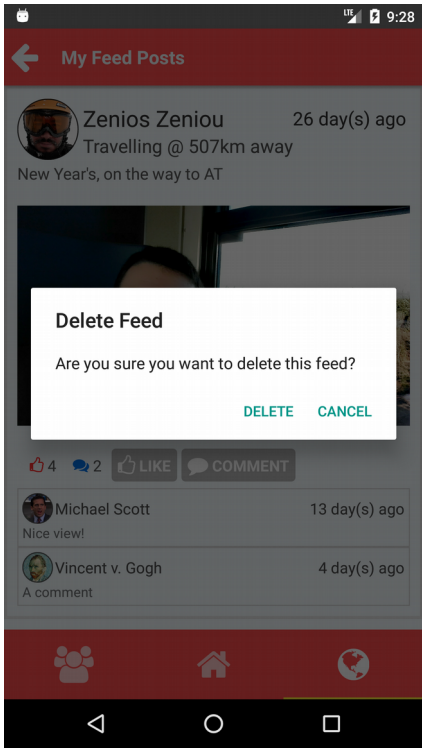


Σχήμα 4.12 Σελίδα Feed /Composer

- **State**
  - user(Object): Αντικείμενο που περιέχει όλα τα δεδομένα για τον τρέχων χρήστη.
  - picture(Object): Αντικείμενο που περιέχει όλα τα απαραίτητα πεδία για την μεταφορά περιεχομένου εικόνας μέσω HTTP.
  - activities_list(Array<Object>): Λίστα δεδομένων των δραστηριοτήτων που ακολουθεί η χρήστης. Χρησιμοποιείται για να επιλέξει ο χρήστης την σχετική δραστηριότητα για την δημοσίευση.
  - activity(String): Δραστηριότητα που επέλεξε ο χρήστης για την δημοσίευση.
  - text(String): Κείμενο δημοσίευσης.

Component για την δημοσίευση ενός νέου αντικειμένου Feed(δημοσίευσης).

- **Feed / Personal**

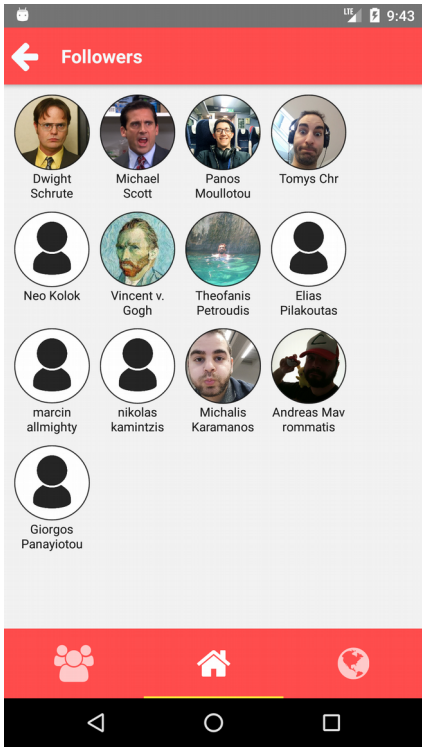


Σχήμα 4.13 Σελίδα Feed / Personal

- **State**
  - feed_list(Array<Object>): Λίστα αντικειμένων Feed όπως αυτή ανακτάται από τον διακομιστή, μέσω του πεδίου feed της όψης /me/.

Η σελίδα αυτή υλοποιεί επιτρέπει στον χρήστη την επισκόπηση των αντικειμένων Feed, που δημοσίευσε ο ίδιος. Περαιτέρω, ο χρήστης μπορεί να διαγράψει εάν επιθυμεί κάποια από αυτά τα αντικείμενα. Αυτό επιτυγχάνεται μέσω ενός “long-press” σε κάποιο από τα αντικείμενα Feed, όπως δείχνει το σχήμα στα δεξιά.

- **Followers / Following**

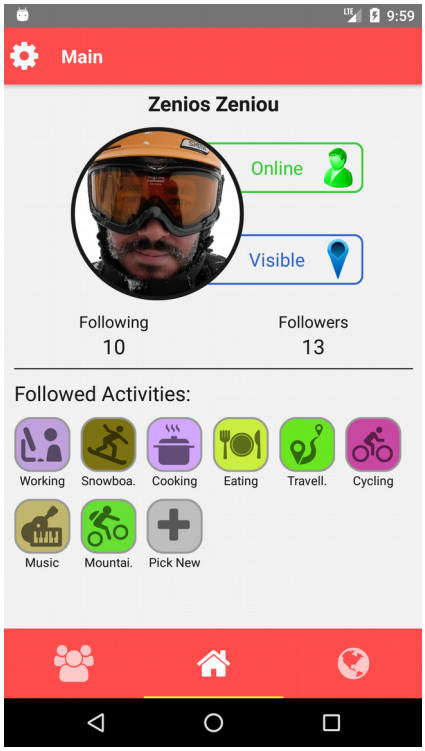


Σχήμα 4.14 Σελίδα Followers

- **State**
  - followers(Array<String>): Λίστα που περιέχει διευθύνσεις URL προς τον διακομιστή, για την ανάκτηση προφίλ χρηστών.

Οι σελίδες Followers & Following αποτελούνται από μία λίστα αντικειμένων UserIcon, κάθε ένα εκ των οποίων αντιστοιχεί στο προφίλ ενός χρήστη που ακολουθεί ή ακολουθείται αντίστοιχα από τον τρέχων χρήστη. Κάθε αντικείμενο UserIcon μπορεί να πατηθεί από τον χρήστη για να πλοηγηθεί στην αντίστοιχη σελίδα UserPage.

- **Home / Main**



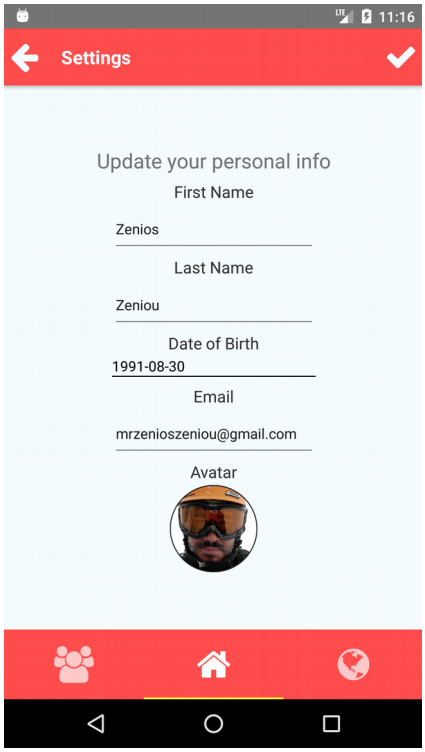
Σχήμα 4.15 Σελίδα Home/Main

- **State**
  - active(Boolean): Boolean πεδίο που περιέχει την τιμή του διακόπτη κατάστασης.
  - location(Boolean): Boolean πεδίο που περιέχει την τιμή του διακόπτη τοποθεσίας.
  - user(Object): Αντικείμενο που περιέχει το προφίλ του τρέχων χρήστη, όπως αυτό ανακτάται από τον διακομιστή μέσω της όψης /me/

Η κεντρική σελίδα της εφαρμογής, η οποία προσφέρει μία επισκόπηση του προφίλ χρήστη. Ο χρήστης μπορεί να πατήσει το εικονίδιο στην πάνω αριστερά γωνία της σελίδας, και να μεταβεί στην σελίδα Main / Settings όπου έχει την δυνατότητα να ενημερώσει τα στοιχεία του. Ο χρήστης μπορεί να αγγίξει τα τους δεικτών κατάστασης και τοποθεσίας για να αλλάξει την κατάσταση προφίλ του, ή τους

μετρητές Following & Followers για να μεταβεί στις αντίστοιχες σελίδες. Τέλος, στο κάτω μέρος της σελίδες βρίσκεται μία λίστα components τύπου ActivityIcon, καθένα εκ των οποίων αντιστοιχεί σε μία δραστηριότητα που ακολουθεί ο χρήστης. Στο τέλος της λίστας δραστηριοτήτων βρίσκεται πάντα ένα εικονίδιο “Pick New”, το οποίο επιτρέπει πλοήγηση στην σελίδα ActivityPicker για την ακολούθηση νέων δραστηριοτήτων.

- **Home / Settings**



Σχήμα 4.16 Σελίδα Home/Settings

- **State**
  - first_name(String): Όνομα χρήστη.
  - last_name(String): Επίθετο χρήστη.
  - email(String): Ηλεκτρονική διεύθυνση χρήστη.
  - dateOfBirth(String): Ημερομηνία γεννήσεως χρήστη.
  - avatar(Object): Αντικείμενο που περιέχει όλα τα απαραίτητα πεδία για την μεταφορά περιεχομένου εικόνας μέσω HTTP.

Η σελίδα αυτή επιτρέπει στον χρήστη να ενημέρωση των στοιχείων του. Με το εικονίδιο στην πάνω δεξιά γωνία της σελίδας ο χρήστης μπορεί να εφαρμόσει τις όποιες αλλαγές στο προφίλ του, ενώ με το εικονίδιο στην πάνω αριστερά γωνία απορρίπτει τις εν λόγω αλλαγές.

- **SocialAgentClient**

Το component αυτό αποτελεί το σημείο εισαγωγής στα υπόλοιπα components και περιλαμβάνει επίσης τον πλοηγό TabNavigator υψηλού επιπέδου για την εναλλαγή μεταξύ των τριών κυρίων κομματιών της εφαρμογής. Πέραν αυτού το component αυτό υλοποιεί τις εξής τρεις σελίδες που επιτρέπουν την εισαγωγή στην εφαρμογή:

- **Σελίδα Login**

  Η σελίδα αυτή επιτρέπει στον χρήστη να πιστοποιήσει την ταυτότητα του χρησιμοποιώντας το username και τον κωδικό πρόσβασης του. Γίνεται μέσω αυτής της σελίδας, κλήση προς την όψη /token/ του διακομιστή με τα πεδία που προαναφέρθηκαν και ανάκτηση ενός token το οποίο αποθηκεύεται για μετέπειτα χρήση κατά την

επικοινωνία με τον διακομιστή. Ο χρήστης μπορεί επίσης να μεταβεί μέσω ενός συνδέσμου στην σελίδα Register η οποία περιγράφεται παρακάτω.

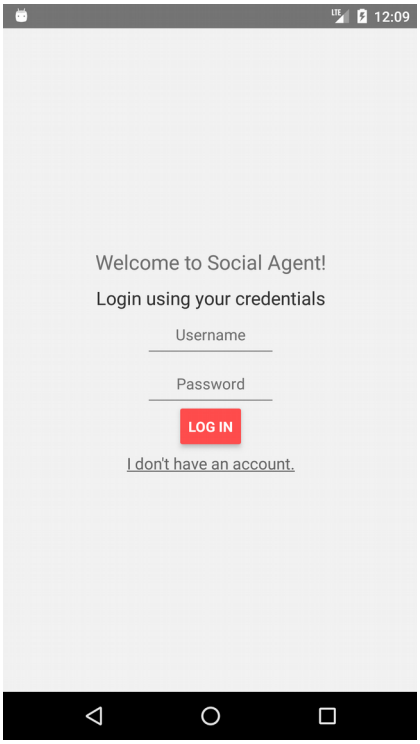
- **Σελίδα Enter**

  Για την αποφυγή ταλαιπωρίας του χρήστη λόγω ανάγκης για επαναλαμβανόμενη ταυτοποίηση, η εφαρμογή έχει την δυνατότητα διατήρησης ενός token ακόμα και μετά το κλείσιμό της. Έτσι όταν ο χρήστης ανοίξει ξανά την εφαρμογή δεν χρειάζεται να περάσει από την σελίδα Login, αλλα του παρουσιάζεται εναλλακτικά αυτή η σελίδα όπου έχει την επιλογή να εισέλθει στην εφαρμογή με την χρήση το αποθηκευμένου token.

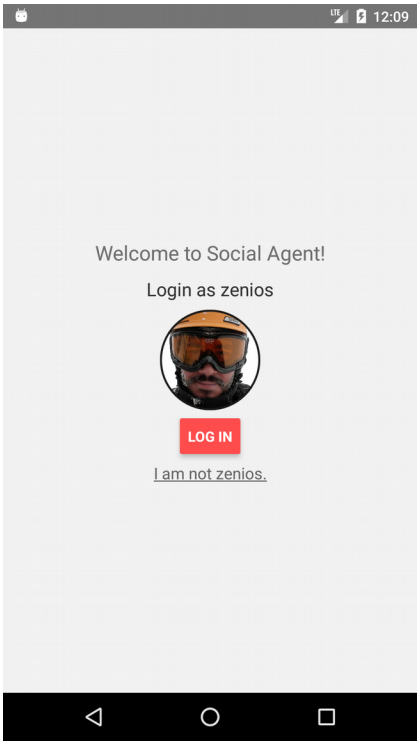
- **Σελίδα Register**

  Η σελίδα αυτή επιτρέπει στον χρήστη να δημιουργήσει το προφίλ του συμπληρώνοντας τα απαραίτητα πεδία, κατά την πρώτη φορά που θα χρησιμοποιήσει την εφαρμογή. Ο χρήστης μπορεί επίσης να μεταβεί μέσω ενός συνδέσμου στην σελίδα Login η οποία περιγράφεται παραπάνω.

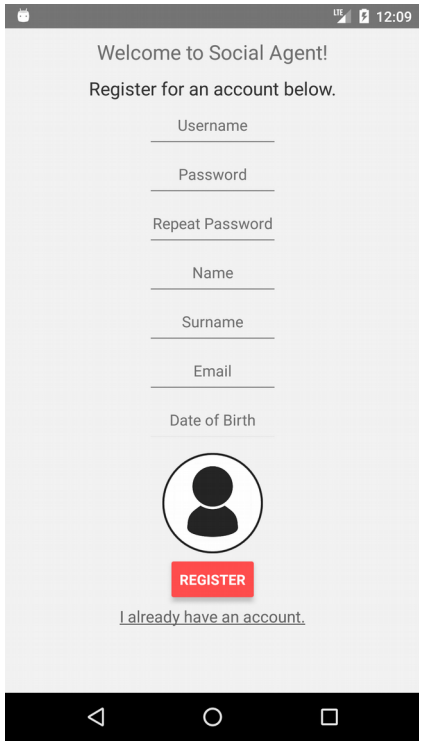
Οι τρεις διαφορετικές σελίδες που προαναφέρθηκαν παρουσιάζονται παρακάτω. Αξίζει να σημειωθεί ότι παρά τις διαφορετικές όψεις, υλοποιούνται μέσω του ιδίου component.



Σχήμα 4.17 Σελίδα Login



Σχήμα 4.18 Σελίδα Enter



Σχήμα 4.19 Σελίδα Register

5. Αξιολόγηση

5.1 Μέθοδος Αξιολόγησης

Για την αξιολόγηση του όλου συστήματος, διεξάχθηκαν δοκιμές με πραγματικούς χρήστες και στην συνέχεια συλλέχθηκαν σχόλια και απόψεις από τους εν λόγω χρήστες. Για την καταγραφή και κατανόηση της εμπειρίας των χρηστών, χρησιμοποιήσαμε ένα ερωτηματολόγιο το οποίο επιτρέπει την γρήγορη αξιολόγηση της εν λόγω εμπειρίας. Η μορφή του ερωτηματολογίου επιτρέπει στους χρήστες να εκφράσουν άμεσα τα συναισθήματα και εντυπώσεις που προκύπτουν κατά την χρήση ενός διαδραστικού προϊόντος.[26]

Το ερωτηματολόγιο αποτελείται από ένα αριθμό κλιμάκων αντίθεσης, όπως είναι για παράδειγμα τα ζευγάρια “δημιουργικό-βαρετό”, “καλό-κακό”, “συνηθισμένο-προχωρημένο” κλπ. Για την κάθε κλίμακα ζητείται από τον χρήστη να τοποθετήσει την εμπειρία χρήσης του σε μία κλίμακα από το ένα μέχρι το εφτά (1-7) όπου για παράδειγμα 1 σημαίνει “πλήρως ασαφές ” ενώ 7 σημαίνει “πλήρως ξεκάθαρο”. Οι κλίμακες αυτές, συλλαμβάνουν την συνολική εικόνα της εμπειρίας των χρηστών, μετρώντας τόσο κλασσικές πτυχές χρηστικότητας (αποδοτικότητα, αξιοπιστία, σαφήνεια) όσο και πτυχές εμπειρίας (εφευρετικότητα, διέγερση ενδιαφέροντος).[26]

Πέρα των ερωτήσεων κλίμακας, στο ερωτηματολόγιο περιλαμβάνονται οι εξής προαιρετικά κλασσικές ερωτήσεις για του χρήστες που επιθυμούν να μοιραστούν την εμπειρία τους σε βάθος.

5.2 Ερωτηματολόγιο

Δίνονται παρακάτω οι κλίμακες και ερωτήσεις που προαναφέρθηκαν. Όλες οι ερωτήσεις και κλίμακες είναι στα Αγγλικά καθώς το σύνολο χρηστών που συμμετείχαν στην δοκιμή δεν γνώριζε εξ΄ολοκλήρου Ελληνικά. Παρόλα αυτά πρόκειται για απλό λεξιλόγιο οπότε δεν μπαίνουμε στην διαδικασία να προσφέρουμε και την μετάφραση σε Ελληνικά

5.2.1 Κλίμακες

	1	2	3	4	5	6	7		
annoying	O	O	O	O	O	O	O	enjoyable	1
not understandable	O	O	O	O	O	O	O	understandable	2
creative	O	O	O	O	O	O	O	dull	3
easy to learn	O	O	O	O	O	O	O	difficult to learn	4
valuable	O	O	O	O	O	O	O	inferior	5
boring	O	O	O	O	O	O	O	exciting	6
not interesting	O	O	O	O	O	O	O	interesting	7
unpredictable	O	O	O	O	O	O	O	predictable	8
fast	O	O	O	O	O	O	O	slow	9
inventive	O	O	O	O	O	O	O	conventional	10
obstructive	O	O	O	O	O	O	O	supportive	11
good	O	O	O	O	O	O	O	bad	12
complicated	O	O	O	O	O	O	O	easy	13
unlikable	O	O	O	O	O	O	O	pleasing	14
usual	O	O	O	O	O	O	O	leading edge	15
unpleasant	O	O	O	O	O	O	O	pleasant	16
secure	O	O	O	O	O	O	O	not secure	17
motivating	O	O	O	O	O	O	O	demotivating	18
meets expectations	O	O	O	O	O	O	O	does not meet expectations	19
inefficient	O	O	O	O	O	O	O	efficient	20
clear	O	O	O	O	O	O	O	confusing	21
impractical	O	O	O	O	O	O	O	practical	22
organized	O	O	O	O	O	O	O	cluttered	23
attractive	O	O	O	O	O	O	O	unattractive	24
friendly	O	O	O	O	O	O	O	unfriendly	25
conservative	O	O	O	O	O	O	O	innovative	26

5.2.2 Ερωτήσεις

- What did you like most about the app?
- What did you dislike most about the app?
- Do you have any comments and/or suggestions?

Το ερωτηματολόγιο υλοποιήθηκε, διανεμήθηκε και συμπληρώθηκε με την χρήση Google Forms.

5.3 Αποτελέσματα

Στην έρευνα αξιολόγησης πήραν μέρος συνολικά 14 άτομα. Τα άτομα αυτά εγκατέστησαν και χρησιμοποίησαν την εφαρμογή στις προσωπικές τους Android συσκευές, και στην συνέχεια ολοκλήρωσαν το ερωτηματολόγιο εμπειρίας χρήσης. Δίνονται παρακάτω τα αποτελέσματα που συλλέχθηκαν μέσω των ερωτήσεων κλίμακας, όπου κάθε στήλη αντιστοιχεί σε μία κλίμακα, και κάθε γραμμή στις απαντήσεις ενός χρήστη.

Item	Mean	Variance	Std. Dev.	No.	Left	Right	Category
1	1.6	0.7	0.9	14	annoying	enjoyable	Attractiveness
2	2.6	0.3	0.5	14	not understandable	understandable	Perspicuity
3	1.4	1.6	1.3	14	creative	dull	Novelty
4	2.6	0.6	0.7	14	easy to learn	difficult to learn	Perspicuity
5	1.5	1.2	1.1	14	valuable	inferior	Stimulation
6	0.9	1.2	1.1	14	boring	exciting	Stimulation
7	1.6	1.3	1.2	14	not interesting	interesting	Stimulation
8	1.3	1.3	1.1	14	unpredictable	predictable	Dependability
9	1.6	1.0	1.0	14	fast	slow	Efficiency
10	1.1	2.9	1.7	14	inventive	conventional	Novelty
11	1.5	0.9	0.9	14	obstructive	supportive	Dependability
12	2.4	0.4	0.6	14	good	bad	Attractiveness
13	2.4	0.4	0.6	14	complicated	easy	Perspicuity
14	1.9	0.7	0.8	14	unlikable	pleasing	Attractiveness
15	0.3	2.2	1.5	14	usual	leading edge	Novelty
16	2.0	0.5	0.7	14	unpleasant	pleasant	Attractiveness
17	1.3	2.2	1.5	14	secure	not secure	Dependability
18	1.7	1.0	1.0	14	motivating	demotivating	Stimulation
19	1.9	0.6	0.8	14	meets expectations	does not meet expectations	Dependability
20	1.8	1.0	1.0	14	inefficient	efficient	Efficiency
21	1.9	2.5	1.6	14	clear	confusing	Perspicuity
22	1.6	1.2	1.1	14	impractical	practical	Efficiency
23	1.8	2.8	1.7	14	organized	cluttered	Efficiency
24	1.6	1.6	1.3	14	attractive	unattractive	Attractiveness
25	2.4	0.9	0.9	14	friendly	unfriendly	Attractiveness
26	0.4	2.6	1.6	14	conservative	innovative	Novelty

Για την ερμηνεία των αποτελεσμάτων δεν χρησιμοποιούμε μία καθολική μετρική. Λόγω της μορφής του ερωτηματολογίου, ένα τέτοιο συνολικό σκορ δεν θα είχε νόημα, καθώς η τιμή αυτή δεν θα μπορούσε να ερμηνευθεί σωστά.[26] Αντιθέτως, κατηγοριοποιούμε τις κλίμακες βάσει των ευρύτερων χαρακτηριστικών που αντιπροσωπεύουν, και κατευθύνουμε το ενδιαφέρον μας προς την ερμηνεία αυτών των κατηγοριών.

Το αρχικό σύνολο αποτελεσμάτων αποτελείται από τιμές που κυμαίνονται μεταξύ των 1 και 7. Για την αξιολόγηση όμως χρησιμοποιούμε το πεδίο -3 έως +3, όπου -3 θεωρείται η χειρότερη δυνατή τιμή ενώ +3 η καλύτερη δυνατή. Αντίστοιχα, το 0 θεωρείται ουδέτερο. Δίνουμε παρακάτω μία ομαδοποίηση των κλιμάκων και τις αντίστοιχες τιμές τους.

Categories	
Attractiveness	1.964
Perspicuity	2.393
Efficiency	1.714
Dependability	1.482
Stimulation	1.411
Novelty	0.804

Αξίζει να σημειώσουμε εδώ, ότι λόγω του μικρού δείγματος χρηστών τα αποτελέσματα ίσως να μην είναι όσο αντιπροσωπευτικά όσο θα θέλαμε. Όπως παρατηρούμε, σε καμία κατηγορία δεν είχαμε αρνητική ή τουλάχιστον ουδέτερη αξιολόγηση, κάτι που αντικειμενικά είναι αμφίβολο. Δεν θα δώσουμε λοιπόν τόση σημασία στις τιμές καθαυτές αλλά, στις κατηγορίες που απέχουν από την μέση τιμή των κατηγοριών, που τυχαίνει να είναι 1.628.

Αμέσως λοιπόν ξεχωρίζουμε την κατηγορία "**σαφήνειας"**(Perspicuity), η οποία απέχει αρκετά από την μέση τιμή. Η κατηγορία αυτή μας λέει ουσιαστικά ότι οι χρήστες βρήκαν την εφαρμογή εύχρηστη και κατανοητή, χωρίς δυσνόητα χαρακτηριστικά. Η μετρική αυτή σε συνδυασμό με την κατηγορία **"ελκυστικότητας"**(η οποία αποκλίνει επίσης από τον μέσο όρο), μπορεί να ερμηνευθεί ως θετική αντίληψη του περιβάλλοντος χρήστη. Μπορούμε λοιπόν να θεωρήσουμε ότι ο σχεδιασμός και υλοποίηση της εφαρμογής, όσο αφορά την πλοήγηση και την χρήση των λειτουργιών που προσφέρονται ήταν σχετικά επιτυχής.

Παρατηρούμε επίσης ότι η τιμή της κατηγορίας **"καινοτομίας"**(Novelty) είναι αρκετά πιο χαμηλή σε σχέση με τις υπόλοιπες. Μπορούμε λοιπόν να θεωρήσουμε πως οι χρήστες δεν αναγνώρισαν κάποια ιδιαίτερα αυθεντικά χαρακτηριστικά στην εφαρμογή. Τουλάχιστο σε σχέση με τις παρούσες υπηρεσίες κοινωνικής δικτύωσης.

Τέλος, δίνονται παρακάτω κάποια από τα σχόλια που άφησαν οι χρήστες.

- **What did you like most about the app?**
  - “The implementation (UI)”
  - “I can find people with the same interests as me”
  - “The UI”
  - “The fact that I could identify people with the same type of interests. It’s a good idea.”
  - “Easy to use”
  - “Sharing of common activities”
- **What did you dislike most about the app?**
  - “Kind of slow”
  - “The user experience is not that great.”
  - “The graphics”
  - “Some functions(like deleting a post) are not so clear”
  - “The layout is a bit plain and boring. Doesn’t look as *up-to-date* as it could be”
- **Do you have any comments and/or suggestions?**
  - “Could be faster”
  - “Selecting categories could be simpler. Once clicked on an activity it should add it, without me having to click the follow button.”
  - “Great app idea”
  - “Maybe add a section(function) for singles/dating. It could be a could way to attract new users”
  - “No suggestions. Overall good job”
  - “Include chat in the future.”

Γίνεται εμφανές, ότι το κοινό ήταν γενικά ευχαριστημένο με τον σχεδιασμό και λειτουργία των στοιχείων του περιβάλλοντος, όχι όμως τόσο με την αισθητική του. Παρατηρούμαι επίσης πως αρκετοί χρήστες βρήκαν ελκυστική την ιδέα κοινωνικής δικτύωσης βάσει δραστηριοτήτων και ενδιαφερόντων. Τέλος αξίζει να αναφέρουμε εδώ, ότι κανένας εκ των των χρηστών δεν ανέφερε περιπτώσεις συντριβής ή σφαλμάτων τα οποία κατέστησαν την χρήση της εφαρμογής αδύνατη.

6. Επίλογος

6.1 Σύνοψη

Στη διπλωματική αυτή εργασία, μελετήθηκε ο τομέας των υπηρεσιών κοινωνικής δικτύωσης και αναπτύχθηκε μία τέτοια υπηρεσία. Το σύστημα που υλοποιήθηκε αποτελείται από έναν διακομιστή(back-end) και μία εφαρμογή πελάτη Android. Για τον διακομιστή χρησιμοποιήσαμε το MVC πλαίσιο Django και την εργαλειοθήκη Django REST Framework. Για την ανάπτυξη της εφαρμογής πελάτη, έγινε χρήση του πλαισίου React Native μέσω του οποίου συγγράφηκαν περισσότερα από 20 διαφορετικά "εξαρτήματα" τα οποία απαρτίζουν την όλη Android εφαρμογή.

Με το πέρας της υλοποίησης, ενοικιάστηκαν πόροι μέσω υπηρεσίας web hosting για την χρήση και αξιολόγηση της εφαρμογής από εθελοντές χρήστες, μέσω ενός ερωτηματολογίου. Οι χρήστες αντέδρασαν θετικά στα πλείστα χαρακτηριστικά της εφαρμογής και σε αρκετές περιπτώσεις υπέδειξαν σημεία τα οποία χαίρουν βελτίωσης. Η απουσία περιπτώσεων αναφοράς σφαλμάτων ή συντριβών της εφαρμογής από τους χρήστες, ήταν αρκετά ενθαρρυντική.

6.2 Πιθανές επεκτάσεις

Με την ποικιλία υπηρεσιών κοινωνικής δικτύωσης σήμερα, θα μπορούσαμε να παρουσιάσουμε εδώ ένα μεγάλο αριθμό λειτουργικών επεκτάσεων που θα είναι δυνατό να υιοθετηθούν από την δική μας πλατφόρμα. Θα αναφέρουμε όμως εδώ τα πιο σχετικά.

Αρχικά θα ήταν πολύ χρήσιμη η υλοποίηση μίας λειτουργίας συνομιλίας μεταξύ των χρηστών, έτσι ώστε μπορούν να επικοινωνήσουν μεταξύ τους μέσω μίας άμεσης γραμμής αλληλεπίδρασης.

Ένα από τα χαρακτηριστικά στα οποία αντέδρασαν θετικά οι χρήστες ήταν η δικτύωση βάσει των κοινών δραστηριοτήτων. Για την επέκταση τις ιδέας αυτής, θα μπορούσαμε να υλοποιήσουμε μία λειτουργία "activity feed" όπου οι χρήστες θα έχουν την δυνατότητα να δημοσιεύσουν και να δουν αντικείμενα τύπου Feed τα οποία είναι σχετικά σε μία μόνο δραστηριότητα. Κατ' αυτό τον τρόπο θα οργανώνονταν σιγά σιγά ομάδες ατόμων γύρω από αυτές τις δραστηριότητας. Η ιδέα αυτή θα μπορούσε να επεκταθεί περαιτέρω με την δημιουργία συμβάντων("events") όπου οι ομάδες θα είχαν την δυνατότητα να δραστηριοποιηθούν εκτός της πλατφόρμας. Τέλος, στα φίλτρα της λειτουργάς Discover θα μπορούσαμε να προσθέσουμε μία

λίστα δραστηριοτήτων οι οποίες θα ήταν κοινές μεταξύ του χρήστη και των προφίλ που ανακαλύπτονται.

Μία άλλη πιθανή επέκταση η οποία ήταν στην πραγματικότητα εντός των αρχικών ιδεών για την πλατφόρμα- είναι η σύνδεση της πλατφόρμας με άλλες υπηρεσίες κοινωνικής δικτύωσης. Θα μπορούσε στην συνέχεια ο χρήστης να μοιραστεί τις δημοσιεύσεις στην πλατφόρμα, και σε άλλα δίκτυα στα οποία δραστηριοποιείται. Πέραν αυτού, διάφορες εφαρμογές θα μπορούσαν να προσφέρουν εισαγόμενο υλικό στην πλατφόρμα. Μία εφαρμογή fitness για παράδειγμα όπως είναι το Endomondo θα επέτρεπε στον χρήστη να μοιραστεί την στατιστικά της εξάσκησης του μέσω της δικής μας πλατφόρμας. Παρομοίως, υλικό θα μπορούσε να εισαχθεί και να εξαχθεί από και προς άλλες υπηρεσίες κοινωνικής δικτύωσης.

Κάποιες λιγότερο δραστικές επεκτάσεις θα ήταν, ο εμπλουτισμός της τοποθεσίας Feed με ονόματα τοποθεσιών χρησιμοποιώντας το GoogleMaps ή OpenStreet API, η δυνατότητα δημοσίευσης βίντεο, ή ετικέτα χρηστών τύπου tag παρομοίως με το Facebook, Twitter κ.α. Και τέλος θα ήταν η προσπάθεια υλοποίησης της εφαρμογής για iOS κινητά, κάτι που θα ήταν στην πραγματικότητα πολύ εύκολο, εφόσον οι πλειοψηφία των εξαρτημάτων που υλοποιήσαμε είναι cross-platform. Το μόνο που χρειαζόμαστε είναι το iOS APK το οποίο απαιτεί για την εξασφάλισή του την αγορά ενός προϊόντος Apple.

7. Παράρτημα

7.1 Κώδικας Διακομιστή

Για πρακτικούς λόγους δεν παραθέτουμε όλα τα αρχεία κώδικα που απαρτίζουν την εφαρμογή. Αντιθέτως δίνονται παρακάτω, μόνο τα αρχεία που μας ενδιαφέρουν, όπως είναι τα μοντέλα δεδομένων, οι όψεις κλπ. Το πλήρες σύνολο κώδικα είναι διαθέσιμο στο GitHub repository του συγγραφέα: https://github.com/mrzenioszeniou/SocialAgent

models.py

Στο αρχείο models.py περιλαμβάνεται ο κώδικας που ορίζει τα μοντέλα δεδομένων.

```
1   from django.db import models
2   from django.contrib.auth.models import AbstractUser
3   from django.conf import settings
4   from django.db.models.signals import post_save
5   from django.dispatch import receiver
6   from rest_framework.authtoken.models import Token
7
8
9   class Activity(models.Model):
10      charUnicode = models.CharField(max_length=6)
11      color = models.CharField(max_length=7)
12      name = models.CharField(max_length=25)
13      description = models.CharField(max_length=100)
14
15      class Meta:
16          ordering = ('name',)
17
18      def __unicode__(self):
19          return self.name
20
21
22  class User(AbstractUser):
23      avatar = models.ImageField(upload_to='users/', default='users/default_avatar.png')
24      dateOfBirth = models.DateField(null=True, blank=True)
25      longitude = models.DecimalField(max_digits=9, decimal_places=6,null=True, blank=True)
26      latitude = models.DecimalField(max_digits=9, decimal_places=6,null=True, blank=True)
27      following = models.ManyToManyField(settings.AUTH_USER_MODEL, related_name= 'followers',
28                  through='Follow', through_fields=('follower', 'followee'))
29      activities = models.ManyToManyField('Activity', related_name= 'followed',
30                  through='ActivityFollow', through_fields=('user','activity'))
31      discover_distance = models.DecimalField(max_digits=7, decimal_places=3, default=9999.999)  # In KM
32      discover_age_max = models.IntegerField(default=61)
33      discover_age_min = models.IntegerField(default=18)
34      discoverable = models.BooleanField(default=True)
35      online = models.BooleanField(default=True)
36      class Meta:
37          ordering = ('id',)
38          unique_together = ('email',)
39
40
41  class Follow(models.Model):
42      follower = models.ForeignKey(settings.AUTH_USER_MODEL, related_name='follows')
43      followee = models.ForeignKey(settings.AUTH_USER_MODEL, related_name='followed_by')
44      date = models.DateField(auto_now_add=True)
45
46      class Meta:
47          ordering = ('id',)
48          unique_together = (("follower", "followee"),)
49
50
51  class Feed(models.Model):
52      SOURCE_CHOICES = (
53          ("Native", "Native"),
54          ("Facebook", "Facebook")
55      )
56      id = models.AutoField(primary_key=True)
57      source = models.CharField(max_length=8,choices=SOURCE_CHOICES, default="Native")
58      user = models.ForeignKey(settings.AUTH_USER_MODEL, related_name='feed')
59      activity = models.ForeignKey('Activity', default=1, related_name='feed')
60      longitude = models.DecimalField(max_digits=9, decimal_places=6,
61                  null=True, blank=True)
```

```
62      latitude = models.DecimalField(max_digits=9, decimal_places=6,
63                  null=True, blank=True)
64      text = models.TextField(max_length=256,null=True, blank=True)
65      picture = models.ImageField(upload_to='feed/',null=True, blank=True)
66      datetime = models.DateTimeField(auto_now_add=True)
67
68      def __unicode__(self):
69          return self.user.username + ':' +  (self.text[:8] + '..') if len(self.text) > 10 else self.text
70
71      class Meta:
72          ordering = ('-datetime',)
73
74
75  class ActivityFollow(models.Model):
76      user = models.ForeignKey(settings.AUTH_USER_MODEL)
77      activity = models.ForeignKey('Activity')
78
79      class Meta:
80          ordering = ('id',)
81
82
83  class Reaction(models.Model):
84      TYPE_CHOICES = (
85          ("Like", "Like"),
86          ("Comment", "Comment")
87      )
88      id = models.AutoField(primary_key=True)
89      type = models.CharField(max_length=8, choices=TYPE_CHOICES, default='Like')
90      feed = models.ForeignKey('Feed', related_name='reactions')
91      user = models.ForeignKey(settings.AUTH_USER_MODEL, related_name='reactions')
92      datetime = models.DateTimeField(auto_now_add=True)
93      content = models.TextField(max_length=128, null=True, blank=True)
94
95
96  @receiver(post_save, sender=settings.AUTH_USER_MODEL)
97  def create_auth_token(sender, instance=None, created=False, **kwargs):
98      if created:
99          Token.objects.create(user=instance)
100
101
```

serializers.py

Στο αρχείο serializers.py περιλαμβάνεται ο κώδικας που ορίζει τους serializers(σειριοποιητές). Οι serializers είναι υπεύθυνοι για την ερμηνεία την “μετάφραση” μοντέλων δεδομένων σε μορφή κειμένου.

```
1   from rest_framework import serializers
2   from api.models import *
3
4
5   class ActivitySerializer(serializers.HyperlinkedModelSerializer):
6       class Meta:
7           model = Activity
8           fields = ('url', 'charUnicode', 'color', 'name', 'description','followed')
9
10
11  class ActivityFollowSerializer(serializers.HyperlinkedModelSerializer):
12      class Meta:
13          model = ActivityFollow
14          fields = ('url', 'user', 'activity')
15
16
17  class FollowSerializer(serializers.HyperlinkedModelSerializer):
18      class Meta:
19          model = Follow
20          fields = ('url','follower','followee','date')
21
22
23  class ReactionSerializer(serializers.HyperlinkedModelSerializer):
24      class Meta:
25          model = Reaction
26          fields = ('url', 'user', 'feed', 'datetime', 'content', 'type')
27
28
29  class FeedSerializer(serializers.HyperlinkedModelSerializer):
30      reactions = ReactionSerializer(many=True, read_only=True)
31
32      class Meta:
```

```
33          model = Feed
34          fields = ('url','source','user','activity','datetime','latitude','longitude','text','picture','source','reactions')
35
36
37  class UserSerializer(serializers.HyperlinkedModelSerializer):
38      activities = ActivityFollowSerializer(source='activityfollow_set', many=True,read_only=True)
39      following = FollowSerializer(source='follows',many=True,read_only=True)
40      followers = FollowSerializer(source='followed_by',many=True,read_only=True)
41
42      class Meta:
43          model = User
44          fields = ('url', 'username', 'first_name', 'last_name', 'avatar', 'dateOfBirth',
45                    'latitude', 'longitude', 'following', 'followers', 'activities')
46
47
48  class CurrentUserSerializer(serializers.HyperlinkedModelSerializer):
49      activities = ActivityFollowSerializer(source='activityfollow_set', many=True, read_only=True)
50      following = FollowSerializer(source='follows',many=True, read_only=True)
51      followers = FollowSerializer(source='followed_by',many=True, read_only=True)
52      reactions = ReactionSerializer(many=True, read_only=True)
53      class Meta:
54          model = User
55          fields = ('url', 'id', 'username', 'password', 'first_name', 'last_name', 'email', 'avatar', 'dateOfBirth',
56                    'latitude', 'longitude', 'following', 'followers', 'activities', 'discover_distance',
57                    'discover_age_max', 'discover_age_min', 'feed', 'discoverable', 'online', 'reactions')
58
59
```

views.py

Στο αρχείο views.py περιλαμβάνεται ο κώδικας που ορίζει τις όψεις του API. Κάθε όψη περιλαμβάνει τον τρόπο διαχείρισης συγκεκριμένων αιτημάτων, τόσο ως προς το περιεχόμενο όσο και ως προς την παρουσίαση, με την χρήση των serializers.

```
1   from api.models import *
2   from api.serializers import *
3   from rest_framework import viewsets, status, mixins, generics
4   from rest_framework.response import Response
5   from rest_framework.permissions import AllowAny
6   from custom import getDistanceFromLatLonInKm, getAgeFromDateOfBirth
7
8
9   class ActivityViewSet(viewsets.ReadOnlyModelViewSet):
10      queryset = Activity.objects.all()
11      serializer_class = ActivitySerializer
12
13      def list(self, request, *args, **kwargs):
14          if request.auth:
15              user = request.user
16              queryset = Activity.objects.exclude(followed=user)
17          else:
18              queryset = Activity.objects.all()
19          serializer = ActivitySerializer(queryset,context={'request':request},many=True)
20          return Response(serializer.data)
21
22
23  class UserViewSet(viewsets.ReadOnlyModelViewSet):
24      queryset = User.objects.exclude(username='admin')
25      serializer_class = UserSerializer
26
27      def list(self, request, *args, **kwargs):
28          if request.auth:
29              user = request.user
30              if user.discoverable:
31                  queryset = [ u for u in
User.objects.exclude(username__in=['admin',user.username]).exclude(followers=user)
32                      if getDistanceFromLatLonInKm(user.latitude,user.longitude,u.latitude,u.longitude) <=
user.discover_distance
33                      and getDistanceFromLatLonInKm(user.latitude,user.longitude,u.latitude,u.longitude) <=
u.discover_distance
34                       and u.discoverable and (user.discover_age_min <= getAgeFromDateOfBirth(u.dateOfBirth) <=
user.discover_age_max)]
35              else:
36                  queryset = []
37          else:
38              queryset = User.objects.exclude(username='admin')
39          serializer = UserSerializer(queryset,context={'request': request}, many=True)
40          return Response(serializer.data)
41
```

```
42
43   class ReactionViewSet(mixins.RetrieveModelMixin, mixins.DestroyModelMixin, mixins.CreateModelMixin,
viewsets.GenericViewSet):
44       queryset = Reaction.objects.all()
45       serializer_class = ReactionSerializer
46
47
48   class ActivityFollowViewSet(mixins.RetrieveModelMixin, mixins.DestroyModelMixin, mixins.CreateModelMixin,
viewsets.GenericViewSet):
49       queryset = ActivityFollow.objects.all()
50       serializer_class = ActivityFollowSerializer
51
52
53   class FollowViewSet(mixins.RetrieveModelMixin, mixins.DestroyModelMixin, mixins.CreateModelMixin,
viewsets.GenericViewSet):
54       queryset = Follow.objects.all()
55       serializer_class = FollowSerializer
56
57
58   class FeedViewSet(viewsets.ModelViewSet):
59       queryset = Feed.objects.all()
60       serializer_class = FeedSerializer
61
62       def list(self, request, *args, **kwargs):
63           if request.auth:
64               user = request.user
65               if user.online:
66                   # The first commented line represents normal functionality. For testing purposes we ignore
some functionality
67                   # queryset = [f for f in Feed.objects.all() if ((f.user in user.following.all() and f.user in
user.followers.all() and f.user.online) or f.user == user)]
68                   queryset = [f for f in Feed.objects.all() if ((f.user in user.following.all() and
f.user.online) or f.user == user)]
69               else:
70                   queryset = [f for f in Feed.objects.all() if f.user == user]
71           else:
72               queryset = Feed.objects.all()
73           if len(queryset) > 20:
74               queryset = queryset[0:20]
75           serializer = FeedSerializer(queryset, context={'request': request}, many=True)
76           return Response(serializer.data)
77
78
79   class CurrentUserView(mixins.CreateModelMixin, generics.GenericAPIView,viewsets.ViewSet):
80       queryset = User.objects.all()
81       serializer_class = CurrentUserSerializer
82       permission_classes = (AllowAny,)
83
84       def get(self, request):
85           if request.auth:
86               user = request.user
87               serializer = CurrentUserSerializer(user, context={'request': request})
88               return Response(serializer.data, status=status.HTTP_200_OK,)
89           else:
90               body = {
91                   "detail" : "Authentication credentials were not provided.",
92                   "help" : "Provide 'Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b' header."
93               }
94               return Response(body, status=status.HTTP_401_UNAUTHORIZED)
95
96       def patch(self, request):
97           if request.auth:
98               user = request.user
99               for attr,value in request.data.iteritems():
100                  setattr(user,attr,value)
101              user.save()
102              serializer = CurrentUserSerializer(user, context={'request': request})
103              return Response(serializer.data, status=status.HTTP_200_OK,)
104          else:
105              body = {
106                  "detail" : "Authentication credentials were not provided.",
107                  "help" : "Provide 'Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b' header."
108              }
109              return Response(body, status=status.HTTP_401_UNAUTHORIZED)
110
111      def perform_create(self, serializer):
112          instance = serializer.save()
113          instance.set_password(instance.password)
114          instance.save()
115
116
```

urls.py

Στο αρχείο urls.py περιλαμβάνεται ο κώδικας που ορίζει την τοπολογία URL του API.

```
1   from django.conf import settings
2   from django.conf.urls import url, include
3   from django.conf.urls.static import static
4   from api import views
5   from rest_framework.routers import DefaultRouter
6   from rest_framework.authtoken import views as auth_views
7
8   router = DefaultRouter()
9   router.register(r'activities', views.ActivityViewSet)
10  router.register(r'users', views.UserViewSet)
11  router.register(r'activityfollows', views.ActivityFollowViewSet)
12  router.register(r'follows', views.FollowViewSet)
13  router.register(r'feed', views.FeedViewSet)
14  router.register(r'reactions', views.ReactionViewSet)
15
16  urlpatterns = [
17      url(r'^', include(router.urls)),
18      url(r'^me/', views.CurrentUserView.as_view({'get':'get','patch':'patch','post':'create'})),
19      url(r'^token/', auth_views.obtain_auth_token),
20  ]
21
22  urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
23  urlpatterns +=  static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
24
25
```

settings.py

Στο αρχείο settings.py περιλαμβάνεται ο κώδικας που θέτει ορισμένες ρυθμίσεις σχετικά με την λειτουργία του δακομιστή.

```
1   import os
2
3   # Build paths inside the project like this: os.path.join(BASE_DIR, ...)
4   BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
5
6   # SECURITY WARNING: keep the secret key used in production secret!
7   SECRET_KEY = '-$zjzhjo%_2r0lrag)vcfq(k#_ulydk9m8qzfgu_8ahfprsh4i'
8
9   # SECURITY WARNING: don't run with debug turned on in production!
10  DEBUG = True
11
12  ALLOWED_HOSTS = ['localhost', '127.0.0.1','10.0.2.2', '*']
13
14  # Application definition
15
16  INSTALLED_APPS = [
17      'django.contrib.admin',
18      'django.contrib.auth',
19      'django.contrib.contenttypes',
20      'django.contrib.sessions',
21      'django.contrib.messages',
22      'django.contrib.staticfiles',
23      'rest_framework',
24      'api.apps.ApiConfig',
25      'rest_framework.authtoken',
26  ]
27
28  MIDDLEWARE_CLASSES = [
29      'django.middleware.security.SecurityMiddleware',
30      'django.contrib.sessions.middleware.SessionMiddleware',
31      'django.middleware.common.CommonMiddleware',
32      'django.middleware.csrf.CsrfViewMiddleware',
33      'django.contrib.auth.middleware.AuthenticationMiddleware',
34      'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
35      'django.contrib.messages.middleware.MessageMiddleware',
36      'django.middleware.clickjacking.XFrameOptionsMiddleware',
37  ]
38
39  ROOT_URLCONF = 'SocialAgentServer.urls'
40
41  TEMPLATES = [
42      {
43          'BACKEND': 'django.template.backends.django.DjangoTemplates',
```

```
44          'DIRS': [],
45          'APP_DIRS': True,
46          'OPTIONS': {
47              'context_processors': [
48                  'django.template.context_processors.debug',
49                  'django.template.context_processors.request',
50                  'django.contrib.auth.context_processors.auth',
51                  'django.contrib.messages.context_processors.messages',
52              ],
53          },
54      },
55  ]
56
57  WSGI_APPLICATION = 'SocialAgentServer.wsgi.application'
58
59  # Database
60  DATABASES = {
61      'default': {
62          'ENGINE': 'django.db.backends.sqlite3',
63          'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
64      }
65  }
66
67
68  # Internationalization
69  LANGUAGE_CODE = 'en-us'
70  TIME_ZONE = 'UTC'
71  USE_I18N = True
72  USE_L10N = True
73  USE_TZ = True
74
75  # Static files (CSS, JavaScript, Images)
76  MEDIA_ROOT = 'media/'
77  MEDIA_URL = '/media/'
78  STATIC_ROOT = 'static/'
79  STATIC_URL = '/static/'
80
81  # Default model for authrrization
82  AUTH_USER_MODEL = 'api.User'
83
84  # REST Framework Settings
85  REST_FRAMEWORK = {
86      'DEFAULT_AUTHENTICATION_CLASSES': (
87          'rest_framework.authentication.TokenAuthentication',
88      ),
89      'DEFAULT_PERMISSION_CLASSES': (
90          'rest_framework.permissions.IsAuthenticated',
91      )
92  }
93
94
```

custom.py

Στο αρχείο custom.py ορίζονται κάποιες βοηθητικές μέθοδοι που χρησιμοποιούνται για από τον υπόλοιπο κώδικα.

```
1   import math, csv, random, datetime
2   from api.models import Activity
3
4
5   def getDistanceFromLatLonInKm(lat1, lon1, lat2, lon2):
6       r = 6371
7       dlat = math.radians(lat2 - lat1)
8       dlon = math.radians(lon2 - lon1)
9       a = math.pow(math.sin(dlat / 2), 2) + math.cos(math.radians(lat1)) * math.cos(math.radians(lat2)) *
math.pow(
10          math.sin(dlon / 2), 2)
11      c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
12      d = r * c
13      return d
14
15
16  def addActivitiesFromCsv(filename):
17      reader = csv.DictReader(open(filename))
18      for i in range(0, 52):
19          row = reader.next()
20          activity = Activity(
21              charUnicode=unichr(int(row['unicode'])),
22              color='#' + ''.join(random.choice('0123456789ABCDEF') for i in range(6)),
```

```
23                 name=row['name'],
24                 description=row['description']
25             )
26             activity.save()
27
28     def getAgeFromDateOfBirth(dateOfBirth):
29         return (datetime.datetime.now().date() - dateOfBirth).days // 365.25
30
31
```

7.2 Κώδικας Εφαρμογής(Android) Πελάτη

Παρομοίως με την περίπτωση του διακομιστή, δεν παρατίθενται εδώ όλα τα αρχεία που απαρτίζουν την εφαρμογή πελάτη. Αντ'αυτού δίνουμε μόνο τα αρχεία στα οποία ορίζονται τα components της εφαρμογής, τα οποία αντιστοιχούν σε κάτι λιγότερο από 4000 γραμμές κώδικα. Πέραν αυτού παραλείπεται από όλα τα components ο κώδικας styling, ο οποίος καθορίζει τον τρόπο με τον οποίο προσαρμόζονται οπτικά τα υπο-components. Με την εξαίρεση του αρχείου **index.android.js** το οποίο βρίσκεται στην ρίζα του κώδικα, όλα τα αρχεία που δίνονται παρακάτω, βρίσκονται στον φάκελο Components. Το πλήρες σύνολο κώδικα είναι διαθέσιμο στο GitHub repository του συγγραφέα: https://github.com/mrzenioszeniou/SocialAgent

index.android.js

Περιέχει το υψηλότερου επιπέδου component SocialAgentClient.

```
1   import React, { Component } from 'react';
2   import {
3     ActivityIndicator,
4     AppRegistry,
5     AsyncStorage,
6     Button,
7     DatePickerAndroid,
8     Image,
9     Picker,
10    StyleSheet,
11    ScrollView,
12    Text,
13    TextInput,
14    TouchableOpacity,
15    ToastAndroid,
16    View
17  } from 'react-native';
18  import { TabNavigator } from 'react-navigation';
19  import Home from './components/Home/';
20  import Discover from './components/Discover/';
21  import Feed from './components/Feed/';
22  import config from './assets/config.json';
23
24  export default class SocialAgentClient extends Component {
25    constructor (props) {
26      super(props);
27      this.state = {
28        isPageReady: false,
29        accepted: false, // If true we render navigator
30        authentication: 'login', // 'login' or 'register'
31        user: null,
32        latitude: null,
33        longitude: null,
34        login_field_username: "",
35        login_field_password: "",
36        register_field_username: "",
37        register_field_password1: "",
38        register_field_password2: "",
39        register_field_first_name: "",
40        register_field_last_name: "",
41        register_field_email: "",
42        register_field_dateOfBirth: "",
43        register_field_avatar_uri: "",
44        register_field_avatar_name: "",
45        register_field_avatar_type: "",
46      };
47    }
48
49    async componentWillMount(){
50      // Try fetching GPS coordinates
51      navigator.geolocation.getCurrentPosition(
52        (position) => {
53          this.setState({
```

```
54            latitude: position.coords.latitude,
55            longitude: position.coords.longitude
56          });
57        },
58        (error) => {console.log(error);},
59        { enableHighAccuracy: true, timeout: 20000},
60      );
61
62      try{ // Attempt to fetch user's profile
63        const token = await AsyncStorage.getItem('@SocialAgent:token');
64        if(this.state.latitude && this.state.longitude){
65          var body = {
66            latitude: this.state.latitude,
67            longitude: this.state.longitude
68          };
69        }else{
70          var body = {};
71        }
72        if(token==null) throw("No token was found");
73        let response = await fetch(
74          'http://'+config['server-ip']+':'+config['server-port']+'/me/',
75          {
76            method: 'PATCH',
77            headers: {
78              'Accept': 'application/json',
79              'Content-Type': 'application/json',
80              'Authorization': 'Token ' + token,
81            },
82            body: JSON.stringify(body)
83          });
84        if(!response.ok) {
85          ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
86        }else{
87          let user = await response.json();
88          await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(user));
89          this.setState({
90            user: user
91          });
92        }
93      }catch(error){ // Failed. Prompt for login/register credentials
94        console.log(error);
95      }
96
97      try{ // Save server address for future calls
98        await AsyncStorage.setItem('@SocialAgent:server-address',
99          'http://'+config['server-ip']+':'+config['server-port']+'/');
100     }catch(error){
101       console.error(error);
102     }
103     this.setState({isPageReady: true});
104     return;
105   }
106
107   async _loginWithCredentials(){
108     try{
109       const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
110       let body = {
111         username: this.state.login_field_username,
112         password: this.state.login_field_password,
113       }
114       let response = await fetch(
115         server_address+'token/',
116         {
117           method: 'POST',
118           headers: {
119             'Accept': 'application/json',
120             'Content-Type': 'application/json'
121           },
122           body: JSON.stringify(body)
123         });
124       if(!response.ok) {
125         ToastAndroid.show('Something went wrong. STATUS('+response.status+')',ToastAndroid.SHORT);
126         return;
127       }
128       const token = (await response.json()).token;
129       await AsyncStorage.setItem('@SocialAgent:token',token);
130       if(this.state.latitude && this.state.longitude){
131         body = {
132           latitude: this.state.latitude,
133           longitude: this.state.longitude
134         };
135       }else{
136         body = {};
137       }
```

```
138         response = await fetch(
139           server_address + 'me/',
140           {
141             method: 'PATCH',
142             headers: {
143               'Accept': 'application/json',
144               'Content-Type': 'application/json',
145               'Authorization': 'Token ' + token,
146             },
147             body: JSON.stringify(body)
148           });
149         if(!response.ok) {
150           ToastAndroid.show('Something went wrong. STATUS('+response.status+')',ToastAndroid.SHORT);
151           return;
152         }
153         let user = await response.json();
154         await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(user));
155         this.setState({user: user});
156       }catch(error){
157         ToastAndroid.show('Something went wrong..', ToastAndroid.SHORT);
158         console.log(error);
159       }
160       return;
161     }
162
163     async _registerAccount(){
164       if(this.state.register_field_password1 != this.state.register_field_password2){
165         ToastAndroid.show('Password entries do not match.', ToastAndroid.SHORT);
166         return;
167       }
168       const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
169       try{
170         let body = new FormData();
171         body.append('username', this.state.register_field_username);
172         body.append('password', this.state.register_field_password1);
173         body.append('first_name', this.state.register_field_first_name);
174         body.append('last_name', this.state.register_field_last_name);
175         body.append('email', this.state.register_field_email);
176         body.append('dateOfBirth', this.state.register_field_dateOfBirth);
177         body.append('latitude', this.state.latitude.toFixed(6));
178         body.append('longitude', this.state.longitude.toFixed(6));
179         if(this.state.register_field_avatar_uri!=''){
180           body.append('avatar', {
181             uri: this.state.register_field_avatar_uri,
182             type: this.state.register_field_avatar_type, // or photo.type
183             name: this.state.register_field_avatar_name
184           });
185         }
186         let response = await fetch(
187           server_address+'me/',
188           {
189             method: 'POST',
190             headers: {
191               'Accept': 'application/json',
192               'Content-Type': 'multipart/form-data'
193             },
194             body: body
195           });
196         if(!response.ok) {
197           if(response.status === 400){
198             ToastAndroid.show('Username and/or email already registered. If you need a password reset contact
the adminstrator.',ToastAndroid.LONG);
199           }else{
200             ToastAndroid.show('Something went wrong.
"'+response.body+'"STATUS('+response.status+')',ToastAndroid.SHORT);
201           }
202           return;
203         }
204         let user = await response.json();
205         body = {
206           "username": this.state.register_field_username,
207           "password": this.state.register_field_password1
208         };
209         response = await fetch(
210           server_address+'token/',
211           {
212             method: 'POST',
213             headers: {
214               'Accept': 'application/json',
215               'Content-Type': 'application/json'
216             },
217             body: JSON.stringify(body)
218           });
219         if(!response.ok) {
```

```
220         ToastAndroid.show('Something went wrong. STATUS('+response.status+')',ToastAndroid.SHORT);
221         return;
222       }
223       const token = (await response.json()).token;
224       await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(user));
225       await AsyncStorage.setItem('@SocialAgent:token',token);
226       this.setState({user: user});
227     }catch(error){
228       console.error(error);
229     }
230     return;
231   }
232
233   _selectAvatarFromGallery(){
234     var ImagePicker = require('react-native-image-picker');
235     var options = {
236       title: 'Select Avatar',
237       storageOptions: {
238         skipBackup: true,
239         path: 'images'
240       }
241     };
242     ImagePicker.showImagePicker(options, (response) => {
243       if (response.didCancel) {
244         console.log('User cancelled image picker');
245       }
246       else if (response.error) {
247         console.log('ImagePicker Error: ', response.error);
248       }
249       else {
250         this.setState({
251           register_field_avatar_uri: response.uri,
252           register_field_avatar_type: response.type,
253           register_field_avatar_name: response.fileName
254         });
255       }
256     });
257   }
258
259   _switchToRegistrationPage(){
260     this.setState({
261       authentication: 'register'
262     });
263     return;
264   }
265
266   _switchToLoginPage(){
267     this.setState({
268       authentication: 'login'
269     });
270     return;
271   }
272
273   async _clearUser(){
274     this.setState({
275       user: null,
276       login_field_username: '',
277       login_field_password: '',
278     });
279     await AsyncStorage.removeItem('@SocialAgent:user');
280     await AsyncStorage.removeItem('@SocialAgent:token');
281   }
282
283   async _loginEnterApp(){
284     this.setState({
285       accepted: true
286     })
287   }
288
289   async changeDateOfBirth(){
290     try {
291       const {action, year, month, day} = await DatePickerAndroid.open({
292         date: Date.now()
293       });
294       if (action !== DatePickerAndroid.dismissedAction) {
295         this.setState({ register_field_dateOfBirth: year+'-'+(month+1)+'-'+day});
296       }
297     } catch ({code, message}) {
298       console.warn('Cannot open date picker', message);
299     }
300   }
301
302   render() {
303     if(!this.state.isPageReady){
```

```
304         return(
305           <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
306             <ActivityIndicator color={'#ff4d4d'}/>
307           </View>
308         );
309       }
310       if(this.state.accepted){
311         const TabNav = TabNavigator(
312           {
313             Home: {
314               screen: Home,
315             },
316             Discover: {
317               screen: Discover,
318             },
319             Feed: {
320               screen: Feed,
321             },
322           },{
323             lazy: true,
324             tabBarPosition: 'bottom',
325             order: ['Discover','Home','Feed'],
326             initialRouteName: 'Home',
327             swipeEnabled: false,
328             animationEnabled: false,
329             tabBarOptions: {
330               labelStyle: {
331                 fontSize:36,
332                 fontFamily: 'awesome',
333               },
334               style : {
335                 backgroundColor: '#ff4d4d'
336               },
337             }
338           }
339         );
340         return <TabNav/>;
341       }else if (this.state.user) { // Account is present. Prompot to launch interface
342         return(
343           <View style={styles.container}>
344               <Text style={styles.welcome}>Welcome to Social Agent!</Text>
345               <Text style={styles.instructions}>
346                 Login as {this.state.user.username}
347               </Text>
348               <Image style={styles.avatar} source={{uri: this.state.user.avatar}}/>
349               <Button
350                 title="Log In"
351                 color="#ff4d4d"
352                 onPress={() => this._loginEnterApp()}
353               />
354               <Text
355                 style={styles.subtext}
356                 onPress={() => this._clearUser()}
357                 >I am not {this.state.user.username}.</Text>
358           </View>
359         );
360       }else if (this.state.authentication=='login') { // Account is not present. Prompt for login
361         return(
362           <View style={styles.container}>
363               <Text style={styles.welcome}>Welcome to Social Agent!</Text>
364               <Text style={styles.instructions}>
365                 Login using your credentials
366               </Text>
367               <TextInput
368                 style={styles.text_input}
369                 placeholder={"Username"}
370                 onChangeText={(text) => this.setState({login_field_username: text})}
371                 value={this.state.login_field_username}
372               />
373               <TextInput
374                 style={styles.text_input}
375                 placeholder={"Password"}
376                 secureTextEntry={true}
377                 onChangeText={(text) => this.setState({login_field_password: text})}
378                 value={this.state.login_field_password}
379               />
380               <Button
381                 title="Log In"
382                 color="#ff4d4d"
383                 onPress={() => {this._loginWithCredentials();}}
384               />
385               <Text
386                 style={styles.subtext}
387                 onPress={() => {this._switchToRegistrationPage();}}
```

```
388                >I don't have an account.</Text>
389            </View>
390          );
391        }else if (this.state.authentication=='register'){ // Account is not present. Prompt for registration
392          return(
393            <View style={styles.scrollOuterContainer}>
394                <ScrollView style={styles.scrollContainer} contentContainerStyle={styles.contentContainer}>
395                  <Text style={styles.welcome}>Welcome to Social Agent!</Text>
396                  <Text style={styles.instructions}>
397                    Register for an account below.
398                  </Text>
399                  <TextInput
400                    style={styles.text_input}
401                    placeholder={"Username"}
402                    onChangeText={(text) => this.setState({register_field_username: text})}
403                    value={this.state.register_field_username}
404                  />
405                  <TextInput
406                    style={styles.text_input}
407                    placeholder={"Password"}
408                    secureTextEntry={true}
409                    onChangeText={(text) => this.setState({register_field_password1: text})}
410                    value={this.state.register_field_password1}
411                  />
412                  <TextInput
413                    style={styles.text_input}
414                    placeholder={"Repeat Password"}
415                    secureTextEntry={true}
416                    onChangeText={(text) => this.setState({register_field_password2: text})}
417                    value={this.state.register_field_password2}
418                  />
419                  <TextInput
420                    style={styles.text_input}
421                    placeholder={"Name"}
422                    onChangeText={(text) => this.setState({register_field_first_name: text})}
423                    value={this.state.register_field_first_name}
424                  />
425                  <TextInput
426                    style={styles.text_input}
427                    placeholder={"Surname"}
428                    onChangeText={(text) => this.setState({register_field_last_name: text})}
429                    value={this.state.register_field_last_name}
430                  />
431                  <TextInput
432                    keyboardType={'email-address'}
433                    style={styles.text_input}
434                    placeholder={"Email"}
435                    onChangeText={(text) => this.setState({register_field_email: text})}
436                    value={this.state.register_field_email}
437                  />
438                  <TouchableOpacity onPress={() => {this.changeDateOfBirth();}}>
439                    <TextInput
440                      editable={false}
441                      style={[styles.text_input,{color: 'black'}]}
442                      placeholder={"Date of Birth"}
443                      onChangeText={(text) => {}}
444                      value={this.state.register_field_dateOfBirth}
445                    />
446                  </TouchableOpacity>
447                  <TouchableOpacity onPress={() => {this._selectAvatarFromGallery();}}>
448                    <Image style={styles.avatar} source={{uri:
449                      this.state.register_field_avatar_uri == ''
450                      ? 'default_avatar'
451                      : this.state.register_field_avatar_uri
452                    }}/>
453                  </TouchableOpacity>
454                  <Button
455                    title="Register"
456                    color="#ff4d4d"
457                    onPress={() => {this._registerAccount();}}
458                  />
459                  <Text
460                    style={styles.subtext}
461                    onPress={() => {this._switchToLoginPage();}}
462                    >I already have an account.</Text>
463                </ScrollView>
464                </View>
465          );
466        }else{ // Should never reach this point
467          return(
468            <View style={styles.container}>
469                <Text style={styles.welcome}>Welcome to Social Agent!</Text>
470                <Text style={styles.instructions}>
471                  Something is wrong here...
```

```
472             </Text>
473         </View>
474       );
475     }
476   }
477 }
478
479 const styles = StyleSheet.create({
    ...
530 });
531
532 AppRegistry.registerComponent('SocialAgentClient', () => SocialAgentClient);
```

ActivityIcon/index.js

```
1   import React, {Component} from 'react';
2   import {
3     ActivityIndicator,
4     AsyncStorage,
5     StyleSheet,
6     View,
7     Text,
8     TouchableOpacity
9   } from 'react-native';
10
11  export default class ActivityIcon extends Component {
12
13    constructor(props){
14      super(props);
15      this.state = {
16        activity : null
17      };
18      this._navigateParent = this._navigateParent.bind(this);
19    }
20
21    _navigateParent() {
22      this.props.callBack(this.state.activity);
23      return;
24    }
25
26    async componentWillMount() {
27        const token = await AsyncStorage.getItem('@SocialAgent:token');
28        try{
29          let response = await fetch(
30            this.props.uri,
31            {
32              method: 'GET',
33              headers: {
34                'Accept': 'application/json',
35                'Content-Type': 'application/json',
36                'Authorization': 'Token ' + token,
37              },
38            });
39          if(!response.ok) return;
40          let responseJson = await response.json();
41          this.setState({activity: responseJson});
42        }catch(error){
43          console.error(error);
44        }
45        return;
46    }
47
48    render() {
49      if(this.state.activity == null) {
50        return (
51          <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
52            <ActivityIndicator color={'#ff4d4d'}/>
53          </View>
54        );
55      }else{
56        return (
57          <View style={styles.mainContainer}>
58            <TouchableOpacity onPress={() => {
59                if("backPreCall" in this.props){
60                  this.props.navigation.navigate('ActivityPage',{
61                    activity: this.state.activity,
62                    backPreCall: this.props.backPreCall
63                  });
64                }else{
65                  this.props.navigation.navigate('ActivityPage',{activity: this.state.activity});
66                }
67              }}>
```

```
68                <View style={[styles.charContainer,{backgroundColor:this.state.activity.color}]}>
69                  <Text style={styles.activityChar}>{this.state.activity.charUnicode}</Text>
70                </View>
71                <Text style={styles.activityName}>{this.state.activity.name.length > 7 ?
this.state.activity.name.substring(0,7) + '.' : this.state.activity.name}</Text>
72              </TouchableOpacity>
73            </View>
74          );
75        }
76      }
77
78    }
79
80    const styles= StyleSheet.create({
        ...
107   });
108
```

ActivityPage/index.js

```
1     import React, {Component} from 'react';
2     import { NavigationActions } from 'react-navigation';
3     import {
4       AsyncStorage,
5       ScrollView,
6       StyleSheet,
7       InteractionManager,
8       View,
9       Image,
10      Text,
11      TouchableOpacity,
12      ToastAndroid,
13      ActivityIndicator
14    } from 'react-native';
15    import UserIcon from '../UserIcon/';
16
17    export default class ActivityPage extends Component {
18
19      constructor(props){
20        super(props);
21        this.state = {
22          pageDidMount: false,
23          activity: props.navigation.state.params.activity,
24          followed: null
25        };
26        this._handleFollowPush = this._handleFollowPush.bind(this);
27      }
28
29      static navigationOptions = function(props) {
30        return({
31          title: props.navigation.state.params.activity.name,
32          headerLeft:
33              <TouchableOpacity onPress={ () => {
34                    if( "backPreCall" in props){
35                      props.backPreCall();
36                    }else if ( "backPreCall" in props.navigation.state.params) {
37                      props.navigation.state.params.backPreCall();
38                  }
39                  props.navigation.dispatch(NavigationActions.back());
40                }}>
41                <Text style={styles.headerButton}>{'\uF060'}</Text>
42              </TouchableOpacity>,
43          headerStyle: styles.header,
44          headerTitleStyle: styles.headerTitle,
45          headerBackTitleStyle: styles.headerButton,
46          });
47      }
48
49      async _handleFollowPush(){
50        try{
51          const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
52          const token = await AsyncStorage.getItem('@SocialAgent:token');
53          var user = await AsyncStorage.getItem('@SocialAgent:user');
54          user = JSON.parse(user);
55          const activity_url = this.state.activity.url;
56          if(this.state.followed){ // Unfollow
57            var url = user.activities.filter(function( obj ) {
58              return obj.activity == activity_url;
59            })[0].url;
60            let response = await fetch(
61              url,
62              {
```

```
63              method: 'DELETE',
64              headers: {
65                'Accept': 'application/json',
66                'Content-Type': 'application/json',
67                'Authorization': 'Token ' + token,
68              },
69            });
70            if(response.ok){
71              this.setState({followed:false});
72              let response = await fetch(
73                server_address+'me/',
74                {
75                  method: 'GET',
76                  headers: {
77                    'Accept': 'application/json',
78                    'Content-Type': 'application/json',
79                    'Authorization': 'Token ' + token,
80                  },
81                });
82              let responseJson = await response.json();
83              await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(responseJson));
84              ToastAndroid.show("Unfollowed.",ToastAndroid.SHORT);
85            }else{
86              ToastAndroid.show("Something went wrong..",ToastAndroid.SHORT);
87            }
88        }else{ // Follow
89          let response = await fetch(
90            server_address + 'activityfollows/',
91            {
92              method: 'POST',
93              headers: {
94                'Accept': 'application/json',
95                'Content-Type': 'application/json',
96                'Authorization': 'Token ' + token,
97              },
98              body: JSON.stringify({
99                user: user.url,
100               activity: activity_url
101             })
102           });
103           if(response.ok){
104             this.setState({followed:true});
105             let response = await fetch(
106               server_address+'me/',
107               {
108                 method: 'GET',
109                 headers: {
110                   'Accept': 'application/json',
111                   'Content-Type': 'application/json',
112                   'Authorization': 'Token ' + token,
113                 },
114               });
115             let responseJson = await response.json();
116             await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(responseJson));
117             ToastAndroid.show("Followed.",ToastAndroid.SHORT);
118           }else{
119             ToastAndroid.show("Something went wrong..",ToastAndroid.SHORT);
120           }
121       }
122     }catch(error){
123       console.error(error);
124     }
125   }
126
127   async componentWillMount() {
128     try{
129       const user = await AsyncStorage.getItem('@SocialAgent:user');
130       var followed_activities = JSON.parse(user).activities;
131       if(followed_activities.some((e) => e.activity === this.state.activity.url)){
132         this.setState({followed: true});
133       }else{
134         this.setState({followed: false})
135       }
136     }catch(error){
137       console.error(error);
138     }
139     return;
140   }
141
142   componentDidMount() {
143     InteractionManager.runAfterInteractions(()=>{
144       this.setState({pageDidMount: true});
145     });
146   }
```

```
147
148     render() {
149       if(this.state.pageDidMount){
150         return (
151           <View style={styles.mainContainer}>
152             <ScrollView style={styles.scrollContainer} contentContainerStyle={styles.contentContainers}>
153               <View style={[styles.horizontalContainer,
154                            {margin:20,justifyContent:'space-between'}]}>{/*Activity Icon + Name*/}
155                 <View style={[styles.activityIconWrapper,
156                     { backgroundColor:this.state.activity.color}]}>
157                   <Text style={styles.activityIcon}>{this.state.activity.charUnicode}</Text>
158                 </View>
159                 <View style={[styles.verticalContainer,styles.activityNameWrapper]}>
160                   <Text style={[styles.activityName]}>{this.state.activity.name}</Text>
161                   <Text style={[styles.activityDescription]}>{this.state.activity.description}</Text>
162                 </View>
163               </View>
164               <View style={[styles.horizontalContainer,
165                            {marginHorizontal:20,paddingBottom:10,justifyContent:'space-around',
166                             borderBottomWidth: 1,borderColor:'#1a1a1a'}]}>{/*Followers + Buttons*/}
167                 <View style={styles.verticalContainer}>{/*Followers*/}
168                   <Text style={styles.followersText}>Followers</Text>
169                   <Text style={styles.followersNumber}>{this.state.activity.followed.length}</Text>
170                 </View>
171                 <View>{/*Buttons*/}
172                   <TouchableOpacity onPress={this._handleFollowPush.bind(this)}>
173                     <View style={this.state.followed?styles.unfollowButtonWrapper:styles.followButtonWrapper}>
174                       <Text style={[styles.scrollItemMargin,this.state.followed?
styles.unfollowButtonIcon:styles.followButtonIcon]}>
175                         {this.state.followed?'\uF056':'\uF055'}
176                       </Text>
177                       <Text style={styles.followButtonText}>
178                         {this.state.followed ?'Unfollow':'Follow'}
179                       </Text>
180                     </View>
181                   </TouchableOpacity>
182                 </View>
183               </View>{/*End of Followers + Buttons*/}
184               <Text style={[styles.title]}>Followers:</Text>
185               <View style={styles.followersList}>
186                 {
187                   this.state.activity.followed.map(function(follower, index){
188                     return <UserIcon navigation={this.props.navigation} key={index} uri={follower}
showDistance={false}/>;
189                   },this)
190                 }
191               </View>
192             </ScrollView>
193           </View>
194         );
195       }else{
196         return(
197           <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
198             <ActivityIndicator color={'#ff4d4d'}/>
199           </View>
200         );
201       }
202     }
203
204   }
205
206   const styles= StyleSheet.create({
        ...
335   });
336
```

ActivityPicker/index.js

```
1     import React, { Component } from 'react';
2     import { NavigationActions } from 'react-navigation';
3     import {
4       ActivityIndicator,
5       AsyncStorage,
6       StyleSheet,
7       ScrollView,
8       Text,
9       View,
10      Image,
11      TouchableOpacity
12    } from 'react-native';
13    import ActivityIcon from '../ActivityIcon/';
14
```

```
15  export default class ActivityPicker extends Component  {
16
17    constructor(props){
18      super(props);
19      this.state = {
20        activities: null
21      };
22    }
23
24    async componentWillMount(){
25      const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
26      const token = await AsyncStorage.getItem('@SocialAgent:token');
27      try{
28        let response = await fetch(
29          server_address + 'activities/',
30          {
31            method: 'GET',
32            headers: {
33              'Accept': 'application/json',
34              'Content-Type': 'application/json',
35              'Authorization': 'Token ' + token,
36            },
37          });
38        if(!response.ok) return;
39        let responseJson = await response.json();
40        this.setState({activities: responseJson});
41      }catch(error){
42        console.error(error);
43      }
44      return;
45    }
46
47    static navigationOptions = function(props) {
48      return({
49        title: 'Find Activities',
50        headerStyle: styles.header,
51        headerTitleStyle: styles.headerTitle,
52        headerBackTitleStyle: styles.headerButton,
53        headerLeft:
54            <TouchableOpacity onPress={ () => {
55              if( "backPreCall" in props){
56                props.backPreCall();
57              }
58              props.navigation.dispatch(NavigationActions.back());
59            }}>
60              <Text style={styles.headerButton}>{'\uF060'}</Text>
61            </TouchableOpacity>,
62        });
63    }
64
65    render() {
66      if(this.state.activities == null){
67        return(
68          <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
69            <ActivityIndicator color={'#ff4d4d'}/>
70          </View>
71        );
72      }else if (this.state.activities.length == 0) {
73        return(
74          <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
75            <Text>No unfollowed activities available.</Text>
76          </View>
77        );
78      }else{
79        return(
80          <View style={styles.mainContainer}>
81            <ScrollView>
82            <View style={styles.followersList}>
83              {
84                this.state.activities.map(function(activity, index){
85                  return <ActivityIcon
86                    navigation={this.props.navigation}
87                    key={activity.url}
88                    uri={activity.url}
89                    backPreCall={this.props.navigation.state.params.backPreCall}
90                  />;
91                },this)
92              }
93            </View>
94            </ScrollView>
95          </View>
96        );
97      }
98    }
```

```
99
100 }
101
102 const styles= StyleSheet.create({
      ...
129 });
```

Discover/index.js

```
1   import React, { Component } from 'react';
2   import {
3     AsyncStorage,
4     StyleSheet,
5     Text,
6     View,
7     InteractionManager,
8     ActivityIndicator,
9   } from 'react-native';
10  import { StackNavigator } from 'react-navigation';
11  import Main from './Main/';
12  import Settings from './Settings/';
13  import ActivityPage from '../ActivityPage/';
14  import UserPage from '../UserPage/';
15
16  export default class Discover extends Component {
17
18    constructor(props){
19      super(props);
20      this.state = {
21        pageDidMount: false,
22      };
23    }
24
25    componentDidMount() {
26      InteractionManager.runAfterInteractions(()=>{
27        this.setState({pageDidMount: true});
28      });
29    }
30
31    static navigationOptions = {
32      tabBarLabel: '\uF0C0',
33    }
34
35    render() {
36
37      if(this.state.pageDidMount){
38        const StackNav = StackNavigator({
39            Main: {
40              screen: Main
41            },
42            Settings: {
43              screen: Settings
44            },
45            ActivityPage: {
46              screen: ActivityPage
47            },
48            UserPage: {
49              screen: UserPage
50            },
51          },
52          {
53            initialRouteName: 'Main',
54            headerMode: 'screen'
55          }
56        );
57      return <StackNav/>;
58    }else{
59      return (
60      <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
61        <ActivityIndicator color={'#ff4d4d'}/>
62      </View>
63      );
64    }
65    }
66  }
67
68  const styles = StyleSheet.create({
69    container: {
70      flex: 1,
71      justifyContent: 'center',
72      alignItems: 'center',
73      backgroundColor: '#F5FCFF',
```

```
74      },
75      welcome: {
76        fontSize: 20,
77        textAlign: 'center',
78        margin: 10,
79      },
80      instructions: {
81        fontSize: 16,
82        textAlign: 'center',
83        color: '#333333',
84        marginBottom: 5
85      }
86    });
87
```

Discover/Main/index.js

```
1     import React, { Component } from 'react';
2     import {
3       ActivityIndicator,
4       AsyncStorage,
5       InteractionManager,
6       ScrollView,
7       StyleSheet,
8       Text,
9       TouchableOpacity,
10      View,
11    } from 'react-native';
12    import UserIconLarge from '../../UserIconLarge/';
13
14    export default class Main extends Component{
15
16      constructor(props){
17        super(props);
18        this.state = {
19          people: null
20        };
21        this.refreshPeople = this.refreshPeople.bind(this);
22      }
23
24      async refreshPeople(){
25        const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
26        const token = await AsyncStorage.getItem('@SocialAgent:token');
27        try{
28          let response = await fetch(
29            server_address + 'users/',
30            {
31              method: 'GET',
32              headers: {
33                'Accept': 'application/json',
34                'Content-Type': 'application/json',
35                'Authorization': 'Token ' + token,
36              },
37            });
38          if(!response.ok) return;
39          let responseJson = await response.json();
40          this.setState({people: responseJson});
41        }catch(error){
42          console.error(error);
43        }
44        return;
45      }
46
47      componentDidMount() {
48        this.props.navigation.setParams({ refreshPeople: this.refreshPeople });
49      }
50
51      componentWillMount(){
52        this.refreshPeople();
53        return;
54      }
55
56      render(){
57        if(this.state.people == null){
58          return(
59            <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
60              <ActivityIndicator color={'#ff4d4d'}/>
61            </View>
62          );
63        }else{
64          if(this.state.people.length == 0){
65            return(
```

```
66            <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
67              <Text>
68                No other users nearby or your status is set to invisible.
69              </Text>
70            </View>
71          );
72        }else{
73          return (
74            <View style={styles.mainContainer}>
75              <ScrollView>
76              <View style={styles.followersList}>
77                {
78                    this.state.people.map(function(person, index){
79                      return <UserIconLarge navigation={this.props.navigation} key={person.url}
uri={person.url} backPreCall={this.refreshPeople}/>;
80                    },this)
81                }
82              </View>
83              </ScrollView>
84            </View>
85          );
86        }
87      }
88    }
89
90    static navigationOptions = function(props) {
91      return({
92        title: 'Discover',
93        headerStyle: styles.header,
94        headerTitleStyle: styles.headerTitle,
95        headerBackTitleStyle: styles.headerButton,
96        headerLeft:
97            <TouchableOpacity onPress={ () => props.navigation.navigate("Settings",{backPreCall:
props.navigation.state.params.refreshPeople})}>
98              <Text style={styles.headerButton}>{'\uF0B0'}</Text>
99            </TouchableOpacity>,
100       headerRight:
101       <TouchableOpacity onPress={ () => props.navigation.state.params.refreshPeople()}>
102         <Text style={styles.headerButton}>{'\uF021'}</Text>
103       </TouchableOpacity>,
104     });
105   }
106 }
107
108 const styles= StyleSheet.create({
      ...
135 });
136
```

Discover/Settings/index.js

```
1   import React, { Component } from 'react';
2   import { NavigationActions } from 'react-navigation';
3   import {
4     ActivityIndicator,
5     AsyncStorage,
6     Image,
7     ScrollView,
8     Slider,
9     StyleSheet,
10    Text,
11    ToastAndroid,
12    TouchableOpacity,
13    View,
14  } from 'react-native';
15
16
17  export default class Settings extends Component {
18
19    constructor(props){
20      super(props);
21      this.state = {
22        done: false
23      }
24      this.updateSettings = this.updateSettings.bind(this);
25    }
26
27    static navigationOptions = function(props) {
28      return({
29        title: 'Filters',
30        headerStyle: styles.header,
```

```
31          headerTitleStyle: styles.headerTitle,
32          headerBackTitleStyle: styles.headerButton,
33          headerLeft:
34              <TouchableOpacity onPress={ () => {
35                  if( "backPreCall" in props){
36                    props.backPreCall();
37                  }else if ( "backPreCall" in props.navigation.state.params) {
38                    props.navigation.state.params.backPreCall();
39                  }
40                  props.navigation.dispatch(NavigationActions.back());
41                }}>
42                <Text style={styles.headerButton}>{'\uF060'}</Text>
43              </TouchableOpacity>,
44          headerRight:
45              <TouchableOpacity onPress={ () => {props.navigation.state.params.updateSettings();}}>
46                <Text style={styles.headerButton}>{'\uF00C'}</Text>
47              </TouchableOpacity>,
48          });
49      }
50
51      componentDidMount() {
52        this.props.navigation.setParams({ updateSettings: this.updateSettings });
53      }
54
55      async updateSettings(){
56        if(this.state.max_age < this.state.min_age){
57          ToastAndroid.show('Maximum age is smaller than the minimum age. Change and try
again..',ToastAndroid.LONG);
58          return;
59        }
60        const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
61        const token = await AsyncStorage.getItem('@SocialAgent:token');
62        try{
63          var body = {
64            discover_distance: String(this.getActualDistanceFromState(this.state.distance)),
65            discover_age_max: this.state.max_age > 60 ? 100 : this.state.max_age,
66            discover_age_min: this.state.min_age
67          };
68          let response = await fetch(
69            server_address+'me/',
70            {
71              method: 'PATCH',
72              headers: {
73                'Accept': 'application/json',
74                'Content-Type': 'application/json',
75                'Authorization': 'Token ' + token,
76              },
77              body: JSON.stringify(body)
78            });
79          if(!response.ok) {
80            ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
81            return;
82          }
83          let responseJson = await response.json();
84          await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(responseJson));
85          this.props.navigation.state.params.backPreCall();
86          this.props.navigation.dispatch(NavigationActions.back());
87        }catch(error){
88          console.error(error);
89        }
90        return;
91
92      }
93
94      async componentWillMount(){
95        try{
96          const user = JSON.parse(await AsyncStorage.getItem('@SocialAgent:user'));
97          this.setState({
98            done: true,
99            distance: this.getStateFromActualDistance(user.discover_distance),
100           max_age: user.discover_age_max,
101           min_age: user.discover_age_min,
102         });
103       }catch(error){
104         console.error(error);
105       }
106       return;
107     }
108
109     getStateFromActualDistance(d){
110       d = parseFloat(d);
111       switch(true){
112         case (d==0.001):
113           return 1;
```

```
114         break;
115       case (d==0.010):
116         return 2;
117         break;
118       case (d==0.050):
119         return 3;
120         break;
121       case (d>=0.1 && d<=0.5):
122         return (d*10) + 3;
123         break;
124       case (d>=1 && d<=100):
125         return d+8
126         break;
127       default:
128         return 109;
129         break;
130     }
131   }
132
133   getActualDistanceFromState(d){
134     switch(true){
135       case (d==1):
136         return 0.005;
137         break;
138       case (d==2):
139         return 0.010;
140         break;
141       case (d==3):
142         return 0.050;
143         break;
144       case (d>=4 && d<=8):
145         return (d-3)/10;
146         break;
147       case (d>=9 && d<=108):
148         return (d-8)* 1.0
149         break;
150       default:
151         return 9999.999;
152         break;
153     }
154   }
155
156   getStringDistanceFromState(d){
157     switch(true){
158       case (d==1):
159         return '5 meters';
160         break;
161       case (d==2):
162         return '10 meters';
163         break;
164       case (d==3):
165         return '50 meters';
166         break;
167       case (d>=4 && d<=8):
168         return (d-3)*100 + ' meters';
169         break;
170       case (d>=9 && d<=108):
171         return d-8 + ' km'
172         break;
173       default:
174         return '100+ km'
175         break;
176     }
177   }
178
179   render() {
180     if(!this.state.done){
181       return(
182         <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
183           <ActivityIndicator color={'#ff4d4d'}/>
184         </View>
185       );
186     }else{
187       return(
188           <ScrollView style={[styles.mainContainer,{marginHorizontal:4}]}
contentContainerStyle={styles.contentContainers}>
189               <View style={{flex:1}}>
190                 <Text style={styles.itemTitle}>Distance</Text>
191               </View>
192               <View style={[styles.horizontalContainer,{margin:4}]}>
193                 <Text>{'5m'}</Text>
194                 <Text style={{fontSize: 16 ,flex:1,textAlign:'center'}}>
195                 {this.getStringDistanceFromState(this.state.distance)}</Text>
196                 <Text>{'100km'}</Text>
```

```
197                 </View>
198                 <Slider
199                   maximumValue={109}
200                   minimumValue={1}
201                   step={1}
202                   onValueChange={(value) => {
203                     this.setState({distance: value});
204                   }}
205                   value={this.state.distance}
206                   disabled={false}
207                   style={{marginBottom: 10}}
208                 />
209                 <View style={{flex:1}}>
210                   <Text style={styles.itemTitle}>Minimum Age</Text>
211                 </View>
212                 <View style={[styles.horizontalContainer,{margin:4}]}>
213                   <Text>18</Text>
214                   <Text style={{fontSize: 16 ,flex:1,textAlign:'center'}}>
215                     {this.state.min_age>60?'60+':this.state.min_age}
216                   </Text>
217                   <Text>60</Text>
218                 </View>
219                 <Slider
220                   maximumValue={61}
221                   minimumValue={18}
222                   step={3}
223                   onValueChange={(value) => {this.setState({min_age: value===61 ? 100 : value});}}
224                   value={this.state.min_age > 60 ? 61 : this.state.min_age}
225                   disabled={false}
226                   style={{marginBottom: 10}}
227                 />
228                 <View style={{flex:1}}>
229                   <Text style={styles.itemTitle}>Maximum Age</Text>
230                 </View>
231                 <View style={[styles.horizontalContainer,{margin:4}]}>
232                   <Text>18</Text>
233                   <Text style={{fontSize: 16 ,flex:1,textAlign:'center'}}>
234                     {this.state.max_age>60?'60+':this.state.max_age}
235                   </Text>
236                   <Text>60</Text>
237                 </View>
238                 <Slider
239                   maximumValue={61}
240                   minimumValue={18}
241                   step={1}
242                   onValueChange={(value) => {this.setState({max_age: value});}}
243                   value={this.state.max_age}
244                   disabled={false}
245                   style={{marginBottom: 10}}
246                 />
247             </ScrollView>
248         );
249       }
250     }
251   }
252
253   const styles = StyleSheet.create({
        ...
334   });
335
```

Feed/index.js

```
1   import React, { Component } from 'react';
2   import {
3     StyleSheet,
4     FlatList,
5     Text,
6     View,
7     InteractionManager,
8     ActivityIndicator,
9   } from 'react-native';
10  import { StackNavigator } from 'react-navigation';
11  import Main from './Main/';
12  import FeedComposer from './FeedComposer/';
13  import FeedPersonal from './FeedPersonal/';
14
15  export default class Feed extends Component {
16
17    static navigationOptions = {
18      tabBarLabel: '\uF0AC',
19    }
```

```
20
21      constructor(props){
22        super(props);
23        this.state = {
24          pageDidMount: false,
25        };
26      }
27
28      componentDidMount() {
29        InteractionManager.runAfterInteractions(()=>{
30          this.setState({pageDidMount: true});
31        });
32      }
33
34      render(){
35        if(this.state.pageDidMount){
36          const StackNav = StackNavigator(
37            {
38              Main: {
39                screen: Main,
40              },
41              FeedComposer: {
42                screen: FeedComposer,
43              },
44              FeedPersonal: {
45                screen: FeedPersonal,
46              },
47            },{
48              initialRouteName: 'Main',
49              headerMode: 'screen'
50            }
51          );
52          return <StackNav/>;
53
54        }else{
55          return (
56          <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
57            <ActivityIndicator color={'#ff4d4d'}/>
58          </View>
59          );
60        }
61
62      }
63    }
64
```

Feed/Main/index.js

```
1     import React, { Component } from 'react';
2     import {
3       AsyncStorage,
4       ActivityIndicator,
5       ScrollView,
6       StyleSheet,
7       Text,
8       TouchableOpacity,
9       View,
10    } from 'react-native';
11    import FeedItem from '../../FeedItem/';
12
13    export default class Main extends Component {
14
15      constructor(props){
16        super(props);
17        this.state = {
18          feed_list: null
19        };
20        this.refreshFeed = this.refreshFeed.bind(this);
21      }
22
23      static navigationOptions = function(props) {
24        return({
25          title: 'Feed',
26          headerRight:
27              <TouchableOpacity onPress={ () => {
28                props.navigation.state.params.refreshFeed();
29              }}>
30                <Text style={styles.headerButton}>{'\uF021'}</Text>
31              </TouchableOpacity>,
32          headerLeft:
33              <TouchableOpacity onPress={ () => props.navigation.navigate("FeedPersonal", {backPreCall:
props.navigation.state.params.refreshFeed})}>
```

```
34              <Text style={styles.headerButton}>{'\uF022'}</Text>
35            </TouchableOpacity>,
36        headerStyle: styles.header,
37        headerTitleStyle: styles.headerTitle
38        });
39    }
40
41    async refreshFeed(){
42      console.log('refreshFeed called')
43      const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
44      const token = await AsyncStorage.getItem('@SocialAgent:token');
45      try{
46        let response = await fetch(
47          server_address + 'feed/',
48          {
49            method: 'GET',
50            headers: {
51              'Accept': 'application/json',
52              'Content-Type': 'application/json',
53              'Authorization': 'Token ' + token,
54            },
55          });
56        if(!response.ok){
57          ToastAndroid.show('Something went wrong. STATUS('+response.status+')',ToastAndroid.SHORT);
58          return;
59        }
60        let responseJson = await response.json();
61        this.setState({feed_list: responseJson});
62      }catch(error){
63        ToastAndroid.show('Something went wrong. Couldn\'t fetch feed.',ToastAndroid.SHORT);
64        console.log(error);
65      }
66      return;
67    }
68
69    componentDidMount() {
70      this.props.navigation.setParams({ refreshFeed: this.refreshFeed });
71    }
72
73    componentWillMount(){
74        this.refreshFeed();
75        return
76    }
77
78    render() {
79      if(this.state.feed_list!=null && this.state.feed_list.length != 0){
80        return (
81          <View style={styles.container}>
82            <ScrollView>
83              <TouchableOpacity onPress={ () => this.props.navigation.navigate("FeedComposer", {backPreCall:
this.props.navigation.state.params.refreshFeed}) }>
84                <View style={styles.newFeedContainer}>
85                  <Text style={styles.newFeedIcon}>{'\uF044'}</Text>
86                  <Text style={styles.newFeedText}>Post new feed..!</Text>
87                </View>
88              </TouchableOpacity>
89              {
90                this.state.feed_list.map(function(item, index){
91                  return <FeedItem key={item.url} feed={item} refreshFeed={this.refreshFeed}/>;
92                },this)
93              }
94            </ScrollView>
95          </View>
96        );
97      }else if (this.state.feed_list!=null) {
98        return (
99          <View style={styles.container}>
100            <ScrollView>
101              <TouchableOpacity onPress={ () => this.props.navigation.navigate("FeedComposer", {backPreCall:
this.props.navigation.state.params.refreshFeed}) }>
102                <View style={styles.newFeedContainer}>
103                  <Text style={styles.newFeedIcon}>{'\uF044'}</Text>
104                  <Text style={styles.newFeedText}>Post new feed..!</Text>
105                </View>
106              </TouchableOpacity>
107              <View style={{alignItems:'center',justifyContent:'center',flex:1}}>
108                <Text>No feed items available.</Text>
109              </View>
110            </ScrollView>
111          </View>
112
113        );
114      }else{
115        return (
```

```
116          <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
117            <ActivityIndicator color={'#ff4d4d'}/>
118          </View>
119        );
120      }
121    }
122  }
123
124  const styles = StyleSheet.create({
       ...
165  });
166
```

Feed/FeedComposer/index.js

```
1    import React, { Component } from 'react';
2    import { NavigationActions } from 'react-navigation';
3    import {
4      AsyncStorage,
5      ActivityIndicator,
6      Image,
7      Picker,
8      ScrollView,
9      StyleSheet,
10     Text,
11     TextInput,
12     ToastAndroid,
13     TouchableOpacity,
14     View,
15   } from 'react-native';
16   import FeedItem from '../../FeedItem/';
17
18   export default class FeedComposer extends Component {
19
20     constructor(props){
21       super(props);
22       this.state = {
23         user: null,
24         picture: null,
25         activities_list: [],
26         activity: '',
27         text: ''
28       };
29       this._postNewFeed = this._postNewFeed.bind(this);
30     }
31
32     static navigationOptions = function(props) {
33       return({
34         title: 'Post new feed',
35         headerLeft:
36           <TouchableOpacity onPress={ () => {
37               if ( "backPreCall" in props.navigation.state.params) {
38                 props.navigation.state.params.backPreCall();
39               }
40               props.navigation.dispatch(NavigationActions.back());
41             }}>
42             <Text style={styles.headerButton}>{'\uF060'}</Text>
43           </TouchableOpacity>,
44        headerRight:
45          <TouchableOpacity onPress={ () => {props.navigation.state.params._postNewFeed();}}>
46            <Text style={styles.headerButton}>{'\uF046'}</Text>
47          </TouchableOpacity>,
48         headerStyle: styles.header,
49         headerTitleStyle: styles.headerTitle
50         });
51     }
52
53     _selectPictureFromGallery(){
54       var ImagePicker = require('react-native-image-picker');
55       var options = {
56         title: 'Select Avatar',
57         storageOptions: {
58           skipBackup: true,
59           path: 'images'
60         }
61       };
62       ImagePicker.showImagePicker(options, (response) => {
63         if (response.didCancel) {
64           console.log('User cancelled image picker');
65         }
66         else if (response.error) {
67           console.log('ImagePicker Error: ', response.error);
```

```
68          }
69          else {
70            this.setState({
71              picture: {
72                uri: response.uri,
73                type: response.type,
74                name: response.fileName
75              }
76            });
77          }
78        });
79      }
80
81      async _refreshStateData(){
82        try{
83          const user = JSON.parse(await AsyncStorage.getItem('@SocialAgent:user'));
84          const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
85          const token = await AsyncStorage.getItem('@SocialAgent:token');
86          let activities_list = [];
87          let response = null;
88          for(let i=0;i<user.activities.length;i++){
89            response = await fetch(
90              user.activities[i].activity,
91              {
92                method: 'GET',
93                headers: {
94                  'Accept': 'application/json',
95                  'Content-Type': 'application/json',
96                  'Authorization': 'Token ' + token,
97                },
98              }
99            );
100           if(!response.ok){
101             ToastAndroid.show('Something went wrong. STATUS('+response.status+')',ToastAndroid.SHORT);
102             return;
103           }else{
104             activities_list.push(await response.json());
105           }
106         }
107         this.setState({
108           user: user,
109           activities_list: activities_list
110         });
111       }catch(error){
112         console.error(error);
113       }
114       return;
115     }
116
117     async _postNewFeed(){
118       console.log(this.state);
119       if(this.state.activity == ''){
120         ToastAndroid.show('Feed\'s relevevant activity cannot be empty.',ToastAndroid.SHORT);
121         return;
122       }
123       if(this.state.text == ''){
124         ToastAndroid.show('Feed\'s text content cannot be empty.',ToastAndroid.SHORT);
125         return;
126       }
127
128       const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
129       const token = await AsyncStorage.getItem('@SocialAgent:token');
130       try{
131         const body = new FormData();
132         body.append('user', this.state.user.url);
133         body.append('activity', this.state.activity);
134         body.append('text', this.state.text);
135         body.append('latitude', this.state.user.latitude);
136         body.append('longitude', this.state.user.longitude);
137         if(this.state.picture){
138           body.append('picture', {
139             uri: this.state.picture.uri,
140             type: this.state.picture.type,
141             name: this.state.picture.name
142           });
143         }
144         let response = await fetch(
145           server_address+'feed/',
146           {
147             method: 'POST',
148             headers: {
149               'Accept': 'application/json',
150               'Content-Type': 'multipart/form-data',
151               'Authorization': 'Token ' + token,
```

```
152           },
153           body: body
154         });
155       if(!response.ok) {
156         ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
157         return;
158       }
159       response = await fetch(
160         server_address + 'me/',
161         {
162           method: 'GET',
163           headers: {
164             'Accept': 'application/json',
165             'Content-Type': 'application/json',
166             'Authorization': 'Token ' + token,
167           }
168         });
169       if(!response.ok) {
170         ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
171         return;
172       }
173       let user = await response.json();
174       await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(user));
175       this.props.navigation.state.params.backPreCall();
176       this.props.navigation.dispatch(NavigationActions.back());
177     }catch(error){
178       console.error(error);
179     }
180     return;
181   }
182
183   componentDidMount() {
184     this.props.navigation.setParams({ _postNewFeed: this._postNewFeed });
185   }
186
187   componentWillMount(){
188       this._refreshStateData();
189       return
190   }
191
192   render() {
193     if(this.state.user){
194       return (
195         <View style={styles.container}>
196           <Picker
197             mode={'dropdown'}
198             prompt={"Please pick an activity"}
199             style={styles.picker}
200             selectedValue={this.state.activity}
201             onValueChange={(itemValue, itemIndex) => this.setState({activity: itemValue})}>
202             <Picker.Item label="Select Activity.." value='' />
203             {
204               this.state.activities_list.map(function(activity, index){
205                 return <Picker.Item label={activity.name} value={activity.url} key={activity.url}/>;
206               },this)
207             }
208           </Picker>
209           <TextInput
210             style={styles.textInput}
211             multiline={true}
212             maxLength={256}
213             underlineColorAndroid={'transparent'}
214             placeholder={'Tell your followers about it!'}
215             value={this.state.text}
216             onChangeText={(text) => this.setState({text: text})}
217           />
218           <TouchableOpacity
219             onPress={ () => {this._selectPictureFromGallery();}}
220             style={styles.pictureWrapper}
221           >
222             {   !this.state.picture &&
223               <View style={styles.picturePromptContainer}>
224                 <Text style={styles.picturePromptIcon}>{'\uF083'}</Text>
225                 <Text style={styles.picturePromptText}>Attach a picture..</Text>
226               </View>
227             }
228             { this.state.picture &&
229               <Image style={styles.picture} resizeMode={'contain'} source={{uri: this.state.picture.uri}}/>
230             }
231           </TouchableOpacity>
232         </View>
233       );
234     }else{
235       return (
```

```
236          <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
237            <ActivityIndicator color={'#ff4d4d'}/>
238          </View>
239        );
240      }
241    }
242  }
243
244  const styles = StyleSheet.create({
       ...
303  });
304
```

Feed/FeedPersonal/index.js

```
1    import React, { Component } from 'react';
2    import { NavigationActions } from 'react-navigation';
3    import {
4      Alert,
5      AsyncStorage,
6      ActivityIndicator,
7      Image,
8      Picker,
9      ScrollView,
10     StyleSheet,
11     Text,
12     TextInput,
13     ToastAndroid,
14     TouchableOpacity,
15     View,
16   } from 'react-native';
17   import FeedItem from '../../FeedItem/';
18
19   export default class FeedPersonal extends Component {
20
21     constructor(props){
22       super(props);
23       this.state = {
24         user: null,
25         feed_list: [],
26       };
27     }
28
29     static navigationOptions = function(props) {
30       return({
31         title: 'My Feed Posts',
32         headerLeft:
33           <TouchableOpacity onPress={ () => {
34             if ( "backPreCall" in props.navigation.state.params) {
35               props.navigation.state.params.backPreCall();
36             }
37               props.navigation.dispatch(NavigationActions.back());
38             }}>
39             <Text style={styles.headerButton}>{'\uF060'}</Text>
40           </TouchableOpacity>,
41         headerStyle: styles.header,
42         headerTitleStyle: styles.headerTitle
43         });
44     }
45
46
47     async _refreshFeedData(){
48       try{
49         const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
50         const token = await AsyncStorage.getItem('@SocialAgent:token');
51         let response = await fetch(
52           server_address + 'me/',
53           {
54             method: 'GET',
55             headers: {
56               'Accept': 'application/json',
57               'Content-Type': 'application/json',
58               'Authorization': 'Token ' + token,
59             }
60           });
61         if(!response.ok) {
62           ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
63           return;
64         }
65         let user = await response.json();
66         await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(user));
67         let feed_list = [];
```

```
69          let feed_url_list =  Array.from(new Set(user.feed));
70          for(let i=0;i<feed_url_list.length;i++){
71            response = await fetch(
72              feed_url_list[i],
73              {
74                method: 'GET',
75                headers: {
76                  'Accept': 'application/json',
77                  'Content-Type': 'application/json',
78                  'Authorization': 'Token ' + token,
79                }
80              }
81            );
82            if(!response.ok){
83              ToastAndroid.show('Something went wrong. STATUS('+response.status+')',ToastAndroid.SHORT);
84              return;
85            }else{
86              feed_list.push(await response.json());
87            }
88          }
89          this.setState({
90            user: user,
91            feed_list: feed_list
92          });
93        }catch(error){
94          console.error(error);
95        }
96        return;
97      }
98
99      async _removeFeed(feed_item){
100       Alert.alert(
101         'Delete Feed',
102         'Are you sure you want to delete this feed?',
103         [
104           {text: 'Delete', onPress: async () => {
105             if(feed_item.user !== this.state.user.url) {
106               ToastAndroid.show('Only the owner of a feed can delete it.', ToastAndroid.SHORT);
107               return;
108             }
109             const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
110             const token = await AsyncStorage.getItem('@SocialAgent:token');
111             try{
112               let response = await fetch(
113                 feed_item.url,
114                 {
115                   method: 'DELETE',
116                   headers: {
117                     'Accept': 'application/json',
118                     'Content-Type': 'application/json',
119                     'Authorization': 'Token ' + token,
120                   }
121                 });
122               if(!response.ok) {
123                 ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
124                 return;
125               }
126               response = await fetch(
127                 server_address + 'me/',
128                 {
129                   method: 'GET',
130                   headers: {
131                     'Accept': 'application/json',
132                     'Content-Type': 'application/json',
133                     'Authorization': 'Token ' + token,
134                   }
135                 });
136               if(!response.ok) {
137                 ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
138                 return;
139               }
140               let user = await response.json();
141               await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(user));
142               this._refreshFeedData();
143             }catch(error){
144               console.error(error);
145             }
146           }},
147           {text: 'Cancel', onPress: () => {}, style: 'cancel'},
148         ],
149         { cancelable: false }
150       );
151       return;
152     }
```

```
153
154     componentWillMount(){
155         this._refreshFeedData();
156         return
157     }
158
159     render() {
160       if(this.state.feed_list!=null && this.state.feed_list.length != 0){
161         return (
162           <View style={styles.container}>
163             <ScrollView>
164               {
165                 this.state.feed_list.sort((a,b) => b.datetime > a.datetime ? 1 : -1).map(function(item, index){
166                   return (
167                     <TouchableOpacity key={item.url} onLongPress={()=> this._removeFeed(item)}>
168                       <FeedItem key={item.url} feed={item}/>
169                     </TouchableOpacity>
170                   );
171                 },this)
172               }
173             </ScrollView>
174           </View>
175         );
176       }else if (this.state.feed_list!=null) {
177         return (
178           <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
179             <Text>No feed items available.</Text>
180           </View>
181         );
182       }else{
183         return (
184           <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
185             <ActivityIndicator color={'#ff4d4d'}/>
186           </View>
187         );
188       }
189     }
190   }
191
192   const styles = StyleSheet.create({
        ...
233   });
234
```

FeedItem/index.js

```
1     import React, { Component } from 'react';
2     import {
3       ActivityIndicator,
4       AsyncStorage,
5       Button,
6       Image,
7       Modal,
8       StyleSheet,
9       Text,
10      TextInput,
11      TouchableHighlight,
12      TouchableOpacity,
13      ToastAndroid,
14      View,
15    } from 'react-native';
16    import { getElapsedTimeFromTimestamp } from '../../assets/support.js';
17    import { getDistanceFromLatLonInKm } from '../../assets/support.js';
18    import Comment from './FeedComment/';
19
20    export default class FeedItem extends Component {
21
22      constructor(props){
23        super(props);
24        this.state = {
25          user: null,
26          activity: null,
27          latitude: null,
28          longitude: null,
29          liked: null,
30          commenting: false,
31          commentText: ''
32        }
33        this._handleLikePress = this._handleLikePress.bind(this);
34        this._handleCommentPress = this._handleCommentPress.bind(this);
35      }
36
```

```
37      async componentWillMount(){
38        const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
39        const token = await AsyncStorage.getItem('@SocialAgent:token');
40        const curr_user = JSON.parse(await AsyncStorage.getItem('@SocialAgent:user'));
41        this.setState({
42          latitude: curr_user.latitude,
43          longitude: curr_user.longitude,
44        });
45        try{
46          let response = await fetch(
47            this.props.feed.user,
48            {
49              method: 'GET',
50              headers: {
51                'Accept': 'application/json',
52                'Content-Type': 'application/json',
53                'Authorization': 'Token ' + token,
54              },
55            });
56          if(!response.ok){
57            ToastAndroid.show('Something went wrong. STATUS('+response.status+')',ToastAndroid.SHORT);
58            return;
59          }
60          const user = await response.json();
61          response = await fetch(
62            this.props.feed.activity,
63            {
64              method: 'GET',
65              headers: {
66                'Accept': 'application/json',
67                'Content-Type': 'application/json',
68                'Authorization': 'Token ' + token,
69              },
70            });
71          if(!response.ok){
72            ToastAndroid.show('Something went wrong. STATUS('+response.status+')',ToastAndroid.SHORT);
73            return;
74          }
75          const activity = await response.json();
76          if(this.props.feed.reactions.filter(r => r.type === 'Like' && r.user === curr_user.url).length > 0){
77            this.setState({liked: true});
78          }else{
79            this.setState({liked: false});
80          }
81          this.setState({user: user});
82          this.setState({activity: activity});
83        }catch(error){
84          ToastAndroid.show('Something went wrong. Couldn\'t fetch feed.',ToastAndroid.SHORT);
85          console.log(error);
86        }
87      }
88
89      async _handleLikePress(){
90        try{
91          const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
92          const token = await AsyncStorage.getItem('@SocialAgent:token');
93          const curr_user = JSON.parse(await AsyncStorage.getItem('@SocialAgent:user'));
94          if(this.state.liked){
95            const reaction = this.props.feed.reactions.filter(r => r.type === 'Like' && r.user === curr_user.url)
[0];
96            let response = await fetch(
97              reaction.url,
98              {
99                method: 'DELETE',
100               headers: {
101                 'Accept': 'application/json',
102                 'Content-Type': 'application/json',
103                 'Authorization': 'Token ' + token,
104               },
105             });
106           if(!response.ok){
107             ToastAndroid.show('Something went wrong. STATUS('+response.status+')',ToastAndroid.SHORT);
108             return;
109           }else{
110             this.setState({liked: false})
111             this.props.refreshFeed();
112           }
113         }else{
114           const body = {
115             user: curr_user.url,
116             feed: this.props.feed.url,
117             type: 'Like'
118           }
119           let response = await fetch(
```

```
120             server_address+'reactions/',
121             {
122               method: 'POST',
123               headers: {
124                 'Accept': 'application/json',
125                 'Content-Type': 'application/json',
126                 'Authorization': 'Token ' + token,
127               },
128               body: JSON.stringify(body)
129             });
130           if(!response.ok) {
131             ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
132             return;
133           }else{
134             this.setState({liked: true})
135             this.props.refreshFeed();
136           }
137         }
138       }catch(error){
139         ToastAndroid.show('Something went wrong.',ToastAndroid.SHORT);
140         console.log(error);
141       }
142       return;
143     }
144
145     async _handleCommentPress(){
146       if(this.state.commentText.replace(/\s/g, '') === ''){
147         ToastAndroid.show('Empty comments are not allowed',ToastAndroid.SHORT);
148       }
149       try{
150         const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
151         const token = await AsyncStorage.getItem('@SocialAgent:token');
152         const curr_user = JSON.parse(await AsyncStorage.getItem('@SocialAgent:user'));
153         const body = {
154           user: curr_user.url,
155           feed: this.props.feed.url,
156           type: 'Comment',
157           content: this.state.commentText
158         }
159         let response = await fetch(
160           server_address+'reactions/',
161           {
162             method: 'POST',
163             headers: {
164               'Accept': 'application/json',
165               'Content-Type': 'application/json',
166               'Authorization': 'Token ' + token,
167             },
168             body: JSON.stringify(body)
169           });
170         if(!response.ok) {
171           ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
172           return;
173         }else{
174           this.setState({commenting: false});
175           this.props.refreshFeed();
176         }
177       }catch(error){
178         ToastAndroid.show('Something went wrong.',ToastAndroid.SHORT);
179         console.log(error);
180       }
181       return;
182     }
183
184     _formattedName(){
185       let name = this.state.user.first_name + ' ' + this.state.user.last_name;
186       if(name.length > 16){
187         return name.slice(0,15) + '.';
188       }else{
189         return name;
190       }
191     }
192
193     render() {
194       if(this.state.user && this.state.activity){
195         return(
196           <View style={styles.mainContainer}>
197             <View style={styles.title}>
198               <Image style={styles.avatar} source={{uri: this.state.user.avatar}}/>
199               <View style={{flexDirection:'column', padding: 4, flex:1, justifyContent:'space-between'}}>
200                 <View style={{flexDirection:'row', flex:1, justifyContent:'space-between',
alignItems:'center'}}>
201                   <Text style={styles.name}>{this._formattedName()}</Text>
202                   <Text style={styles.time}>{getElapsedTimeFromTimestamp(this.props.feed.datetime)}</Text>
```

```
203                 </View>
204                 <Text style={styles.activityAndLocation}>{this.state.activity.name} @ {
205                   getDistanceFromLatLonInKm(
206                     this.state.latitude,this.state.longitude,this.props.feed.latitude,this.props.feed.longitude
207                   )} away</Text>
208               </View>
209             </View>
210             <View style={styles.content}>
211               <Text style={styles.contentText}>{this.props.feed.text}</Text>
212                 {this.props.feed.picture && <Image resizeMode={'contain'} style={styles.picture} source={{uri:
this.props.feed.picture}} />}
213             </View>
214             <View style={styles.reactionsContainer}>
215               <View style={styles.reactionCountContainer}>
216                 <Text style={styles.likeCountIcon}>{'\uF087'}</Text>
217                 <Text style={styles.reactionCountText}>{this.props.feed.reactions.reduce((acc,cur) => cur.type
=== 'Like' ? acc + 1 : acc ,0)}</Text>
218               </View>
219               <View style={styles.reactionCountContainer}>
220                 <Text style={styles.commentCountIcon}>{'\uF086'}</Text>
221                 <Text style={styles.reactionCountText}>{this.props.feed.reactions.reduce((acc,cur) => cur.type
=== 'Comment' ? acc + 1 : acc ,0)}</Text>
222               </View>
223               <TouchableOpacity onPress={this._handleLikePress}>
224                 <View style={[styles.reactionsSubContainer,{backgroundColor:
225                   this.state.liked ? '#ff6666' : '#b3b3b3'}]}>
226                   <Text style={styles.reactionsIcon}>{'\uF087'}</Text>
227                   <Text style={styles.reactionsText}>LIKE</Text>
228                 </View>
229               </TouchableOpacity>
230               <TouchableOpacity onPress={() => this.setState({commenting: true})}>
231                 <View style={[styles.reactionsSubContainer,{backgroundColor: '#b3b3b3'}]}>
232                   <Text style={styles.reactionsIcon}>{'\uF075'}</Text>
233                   <Text style={styles.reactionsText}>COMMENT</Text>
234                 </View>
235               </TouchableOpacity>
236
237             </View>
238             <View>
239               {
240                 this.props.feed.reactions.filter((r) => r.type === 'Comment').map(function(reaction, index){
241                   return <Comment reaction={reaction} key={index}/>;
242                 },this)
243               }
244             </View>
245             <Modal
246               transparent={true}
247               visible={this.state.commenting}
248               animationType={'slide'}
249               onRequestClose={() => this.setState({commenting: false})}
250             >
251               <View style={{flex:1}}/>
252               <View style={styles.modalContainer}>
253                 <TouchableOpacity onPress={() => this.setState({commenting: false})}>
254                   <Text style={styles.modalCancelIcon}>{'\uF057'}</Text>
255                 </TouchableOpacity>
256                 <TextInput
257                   maxLength={128}
258                   style={styles.modalText}
259                   placeholder={"Maximum 128 characters."}
260                   onChangeText={(text) => this.setState({commentText: text})}
261                   value={this.state.commentText}
262                   multiline={true}
263                   underlineColorAndroid={'#0099ff'}
264                 />
265                 <TouchableOpacity onPress={this._handleCommentPress}>
266                   <Text style={styles.modalPostIcon}>{'\uF045'}</Text>
267                 </TouchableOpacity>
268               </View>
269             </Modal>
270           </View>
271         )
272       }else{
273         return (
274           <View style={{alignItems:'center',justifyContent:'center',flex:1}}>
275             <ActivityIndicator color={'#ff4d4d'}/>
276           </View>
277         );
278       }
279     }
280   }
281
282   const styles = StyleSheet.create({
        ...
```

```
398 });
```

FeedItem/FeedComment/index.js

```
1    import React, { Component } from 'react';
2    import {
3      ActivityIndicator,
4      AsyncStorage,
5      Image,
6      StyleSheet,
7      Text,
8      TouchableOpacity,
9      ToastAndroid,
10     View,
11   } from 'react-native';
12   import { getElapsedTimeFromTimestamp } from '../../../assets/support.js';
13
14   export default class Comment extends Component {
15
16     constructor(props){
17       super(props);
18       this.state = {
19         user: null
20       };
21     }
22
23     async componentWillMount(){
24       const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
25       const token = await AsyncStorage.getItem('@SocialAgent:token');
26       let response = await fetch(
27         this.props.reaction.user,
28         {
29           method: 'GET',
30           headers: {
31             'Accept': 'application/json',
32             'Content-Type': 'application/json',
33             'Authorization': 'Token ' + token,
34           },
35         });
36       if(!response.ok){
37         ToastAndroid.show('Something went wrong. STATUS('+response.status+')',ToastAndroid.SHORT);
38         return;
39       }else{
40         this.setState({
41           user: await response.json()
42         });
43       }
44     }
45
46     render(){
47       if(this.state.user === null){
48         return (
49           <View style={{alignItems:'center',justifyContent:'center',flex:1}}>
50             <ActivityIndicator color={'#ff4d4d'}/>
51           </View>
52         );
53       }else{
54         return(
55           <View style={styles.commentContainer}>
56             <View style={styles.commentHeader}>
57               <Image style={styles.commentAvatar} source={{uri: this.state.user.avatar}}/>
58               <Text style={styles.commentTitle}>{this.state.user.first_name} {this.state.user.last_name}</Text>
59               <Text style={[styles.commentTitle,{flex:1, textAlign:
'right'}]}>{getElapsedTimeFromTimestamp(this.props.reaction.datetime)}</Text>
60             </View>
61             <Text style={styles.commentText}>{this.props.reaction.content}</Text>
62           </View>
63         );
64       }
65     }
66   }
67
68   const styles = StyleSheet.create({
       ...
108  });
109
```

Followers/index.js

```
1    import React, { Component } from 'react';
```

```
2    import { NavigationActions } from 'react-navigation';
3    import {
4      ActivityIndicator,
5      AsyncStorage,
6      StyleSheet,
7      ScrollView,
8      Text,
9      View,
10     Image,
11     TouchableOpacity
12   } from 'react-native';
13   import UserIcon from '../UserIcon/';
14
15   export default class Followers extends Component  {
16
17     constructor(props){
18       super(props);
19       this.state = {
20         followers: null
21       };
22     }
23
24     async componentWillMount(){
25       try{
26         const response = await AsyncStorage.getItem('@SocialAgent:user');
27         const user = await JSON.parse(response);
28         this.setState({
29           followers: user.followers
30         });
31       }catch(error){
32         console.error(error);
33       }
34       return;
35     }
36
37     static navigationOptions = function(props) {
38       return({
39         title: 'Followers',
40         headerLeft:
41             <TouchableOpacity onPress={ () => {
42                   if( "backPreCall" in props){
43                     props.backPreCall();
44                   }else if ( "backPreCall" in props.navigation.state.params) {
45                     props.navigation.state.params.backPreCall();
46                  }
47                  props.navigation.dispatch(NavigationActions.back());
48                }}>
49                <Text style={styles.headerButton}>{'\uF060'}</Text>
50              </TouchableOpacity>,
51         headerStyle: styles.header,
52         headerTitleStyle: styles.headerTitle,
53         headerBackTitleStyle: styles.headerButton,
54         });
55     }
56
57     render() {
58
59       if(this.state.followers!==null){
60         return(
61           <View style={styles.mainContainer}>
62             <View style={styles.followersList}>
63               {
64                 this.state.followers.map(function(follower, index){
65                   return <UserIcon navigation={this.props.navigation} key={follower.follower}
uri={follower.follower}/>;
66                 },this)
67               }
68             </View>
69           </View>
70         );
71       }else{
72         return(
73           <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
74             <ActivityIndicator color={'#ff4d4d'}/>
75           </View>
76         );
77       }
78     }
79
80   }
81
82   const styles= StyleSheet.create({
       ...
109  });
```

Following/index.js

```
1    import React, { Component } from 'react';
2    import { NavigationActions } from 'react-navigation';
3    import {
4      ActivityIndicator,
5      AsyncStorage,
6      StyleSheet,
7      ScrollView,
8      Text,
9      View,
10     Image,
11     TouchableOpacity
12   } from 'react-native';
13   import UserIcon from '../UserIcon/';
14
15   export default class Following extends Component  {
16
17     constructor(props){
18       super(props);
19       this.state = {
20         followers: null
21       };
22     }
23
24     async componentWillMount(){
25       try{
26         const response = await AsyncStorage.getItem('@SocialAgent:user');
27         const user = await JSON.parse(response);
28         this.setState({
29           followers: user.following
30         });
31       }catch(error){
32         console.error(error);
33       }
34       return;
35     }
36
37     static navigationOptions = function(props) {
38       return({
39         title: 'Following',
40         headerLeft:
41             <TouchableOpacity onPress={ () => {
42                   if( "backPreCall" in props){
43                     props.backPreCall();
44                   }else if ( "backPreCall" in props.navigation.state.params) {
45                      props.navigation.state.params.backPreCall();
46                  }
47                  props.navigation.dispatch(NavigationActions.back());
48                }}>
49                <Text style={styles.headerButton}>{'\uF060'}</Text>
50              </TouchableOpacity>,
51         headerStyle: styles.header,
52         headerTitleStyle: styles.headerTitle,
53         headerBackTitleStyle: styles.headerButton,
54         });
55     }
56
57     render() {
58       if(this.state.followers!==null){
59         return(
60           <View style={styles.mainContainer}>
61             <View style={styles.followersList}>
62               {
63                 this.state.followers.map(function(follower, index){
64                   return <UserIcon navigation={this.props.navigation} key={follower.followee}
uri={follower.followee}/>;
65                 },this)
66               }
67             </View>
68           </View>
69         );
70       }else{
71         return(
72           <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
73             <ActivityIndicator color={'#ff4d4d'}/>
74           </View>
75         );
76       }
77     }
78   }
79
80   const styles= StyleSheet.create({
       ...
```

```
107 });
```

Home/index.js

```
1   import React, { Component } from 'react';
2   import {
3     AsyncStorage,
4     StyleSheet,
5     Text,
6     View,
7     InteractionManager,
8     ActivityIndicator,
9   } from 'react-native';
10  import { StackNavigator } from 'react-navigation';
11  import Main from './Main/';
12  import Settings from './Settings/';
13  import ActivityPicker from '../ActivityPicker';
14  import ActivityPage from '../ActivityPage/';
15  import UserPage from '../UserPage/';
16  import Followers from '../Followers/';
17  import Following from '../Following/';
18
19  export default class Home extends Component {
20
21    constructor(props){
22      super(props);
23      this.state = {
24        pageDidMount: false,
25      };
26    }
27
28    componentDidMount() {
29      InteractionManager.runAfterInteractions(()=>{
30        this.setState({pageDidMount: true});
31      });
32    }
33
34    static navigationOptions = {
35      tabBarLabel: '\uF015',
36    }
37
38    render() {
39      if(this.state.pageDidMount){
40        const StackNav = StackNavigator({
41            Main: {
42              screen: Main
43            },
44            Settings: {
45              screen: Settings
46            },
47            ActivityPicker: {
48              screen: ActivityPicker
49            },
50            ActivityPage: {
51              screen: ActivityPage
52            },
53            Followers: {
54              screen: Followers
55            },
56            Following: {
57              screen: Following
58            },
59            UserPage: {
60              screen: UserPage
61            },
62          },
63          {
64            initialRouteName: 'Main',
65            headerMode: 'screen'
66          }
67        );
68      return <StackNav/>;
69    }else{
70      return (
71      <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
72        <ActivityIndicator color={'#ff4d4d'}/>
73      </View>
74      );
75    }
76    }
77  }
78
```

```
79  const styles = StyleSheet.create({
       ...
97  });
```

Home/Main/index.js

```
1   import React, { Component } from 'react';
2   import {
3     AsyncStorage,
4     ActivityIndicator,
5     Image,
6     ScrollView,
7     StyleSheet,
8     Text,
9     ToastAndroid,
10    View,
11    TouchableOpacity,
12
13  } from 'react-native';
14  import ActivityIcon from '../../ActivityIcon/';
15
16  export default class Main extends Component {
17
18    constructor(props){
19      super(props);
20      this.state = {
21        active: null,
22        location: null,
23        user: null,
24      };
25      this.refreshUser = this.refreshUser.bind(this);
26    }
27
28    static navigationOptions = function(props) {
29      return({
30        title: 'Main',
31        headerLeft:
32            <TouchableOpacity onPress={ () => props.navigation.navigate("Settings",{backPreCall:
props.navigation.state.params.refreshUser})}>
33              <Text style={styles.headerButton}>{'\uF013'}</Text>
34            </TouchableOpacity>,
35        headerStyle: styles.header,
36        headerTitleStyle: styles.headerTitle
37        });
38    };
39
40    _handleActivityStatusChange(newUserObject){
41      this.setState({user: newUserObject});
42    }
43
44    async _handlePressStatus() {
45      const active = !this.state.active;
46      const location = !this.state.active && this.state.location;
47      const token = await AsyncStorage.getItem('@SocialAgent:token');
48      const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
49      const body = {
50        online: active,
51        discoverable: location
52      };
53      try{
54        let response = await fetch(
55          server_address+ 'me/',
56          {
57            method: 'PATCH',
58            headers: {
59              'Accept': 'application/json',
60              'Content-Type': 'application/json',
61              'Authorization': 'Token ' + token,
62            },
63            body: JSON.stringify(body)
64          });
65        if(!response.ok) {
66          ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
67        }else{
68          let user = await response.json();
69          await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(user));
70          this.setState({
71            user: user,
72            active: active,
73            location: location,
74          });
75          let text = this.state.active?'online.':'offline.';
```

```
76          ToastAndroid.show('You are now '+text,ToastAndroid.SHORT);
77        }
78      }catch(error){
79        ToastAndroid.show('Something went wrong..'+text,ToastAndroid.SHORT);
80        console.log(error);
81      }
82    }
83
84    async _handlePressLocation() {
85      if(!this.state.active) return;
86      const location = !this.state.location;
87      const token = await AsyncStorage.getItem('@SocialAgent:token');
88      const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
89      const body = {
90        discoverable: location
91      };
92      try{
93        let response = await fetch(
94          server_address+ 'me/',
95          {
96            method: 'PATCH',
97            headers: {
98              'Accept': 'application/json',
99              'Content-Type': 'application/json',
100             'Authorization': 'Token ' + token,
101           },
102           body: JSON.stringify(body)
103         });
104       if(!response.ok) {
105         ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
106       }else{
107         let user = await response.json();
108         await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(user));
109         this.setState({
110           user: user,
111           location: location,
112         });
113         let text = this.state.location?'visible.':'invisible.';
114         ToastAndroid.show('You are now '+text,ToastAndroid.SHORT);
115       }
116     }catch(error){
117       ToastAndroid.show('Something went wrong..'+text,ToastAndroid.SHORT);
118       console.log(error);
119     }
120   }
121
122   async refreshUser(){
123     try{
124       const response = await AsyncStorage.getItem('@SocialAgent:user');
125       const user = await JSON.parse(response);
126       this.setState({
127         user: user,
128         active: user.online,
129         location: user.discoverable,
130       });
131     }catch(error){
132       console.error(error);
133     }
134     return;
135   }
136
137   componentDidMount() {
138     this.props.navigation.setParams({ refreshUser: this.refreshUser });
139   }
140
141   componentWillMount() {
142     this.refreshUser();
143     return;
144   }
145
146
147   render() {
148     var statusIcon = this.state.active ? 'online' : 'offline';
149     var locationIcon = this.state.location ? 'locationon' : 'locationoff';
150     if(this.state.user != null){
151     return (
152       <View style={styles.mainContainer}>
153               <ScrollView style={styles.scrollContainer} contentContainerStyle={styles.contentContainer}>
154                 <View style={styles.scrollItemMargin}>
155                   <Text style={styles.profileName}>{this.state.user.first_name}
{this.state.user.last_name}</Text>
156                 </View>
157                 <View style={[styles.avatarContainer,styles.scrollItemMargin]}>
158                   <View style={[styles.activityButtonWrapper,
```

```
159                         this.state.active
160                         ? {borderColor: '#33cc33'}
161                         : {borderColor: '#808080'}
162                         ]}>
163                       <TouchableOpacity onPress={this._handlePressStatus.bind(this)}>
164                         <View style={{flexDirection:'row'}}>
165                         <Text
166                           style={this.state.active
167                               ? styles.activityButtonTextOn
168                               : styles.activityButtonTextOff}
169                         >{this.state.active ? 'Online' : 'Offline'}</Text>
170                           <Image style={{height:40,width:40}} source={{uri: statusIcon}}/>
171                         </View>
172                       </TouchableOpacity>
173                     </View>
174                     <View style={[styles.locationButtonWrapper,
175                         this.state.location
176                         ? {borderColor: '#3366cc'}
177                         : {borderColor: '#808080'}
178                         ]}>
179                       <TouchableOpacity onPress={this._handlePressLocation.bind(this)}>
180                         <View style={{flexDirection:'row'}}>
181                           <Text
182                             style={this.state.location
183                                 ? styles.locationButtonTextOn
184                                 : styles.locationButtonTextOff}
185                           >{this.state.location ? 'Visible' : 'Invisible'}</Text>
186                           <Image style={{height:40,width:40}} source={{uri: locationIcon}}/>
187                         </View>
188                       </TouchableOpacity>
189                     </View>
190                     <Image style={styles.avatar} source={{uri: this.state.user.avatar}}/>
191                   </View>
192                   <View style={[styles.followsContainer,styles.horizontalContainer,styles.scrollItemMargin]}>
193                     <TouchableOpacity onPress={ () => {
194                       this.props.navigation.navigate("Following", {
195                         backPreCall: this.refreshUser
196                       });}}
197                       style={{flex:1}}>
198                       <View style={[styles.verticalContainer,{flex:1}]}>
199                         <Text style={{textAlign:'center',color: '#1a1a1a',fontSize:16}}>Following</Text>
200                         <Text style={{textAlign:'center',color:
'#1a1a1a',fontSize:20}}>{this.state.user.following.length}</Text>
201                       </View>
202                     </TouchableOpacity>
203                     <TouchableOpacity onPress={ () => {
204                       this.props.navigation.navigate("Followers", {
205                         backPreCall: this.refreshUser
206                       });}}
207                       style={{flex:1}}>
208                       <View style={[styles.verticalContainer,{flex:1}]}>
209                         <Text style={{textAlign:'center',color: '#1a1a1a',fontSize:16}}>Followers</Text>
210                         <Text style={{textAlign:'center',color:
'#1a1a1a',fontSize:20}}>{this.state.user.followers.length}</Text>
211                       </View>
212                     </TouchableOpacity>
213                   </View>
214                   <View style={[styles.horizontalContainer,styles.scrollItemMargin]}>
215                     <Text style={[styles.itemTitle,{flex:1}]}>Followed Activities:</Text>
216                     {/*<Text style={{height:38,fontSize: 38,color:
'#1a1a1a',textAlignVertical:'bottom'}}>+</Text>*/}
217                   </View>
218                   <View style={styles.activitiesList}>
219                     {
220                       this.state.user.activities.map(function(activity, index){
221                         return <ActivityIcon
222                                 navigation={this.props.navigation}
223                                 key={activity.activity}
224                                 uri={activity.activity}
225                                 backPreCall={this.refreshUser}
226                               />;
227                       },this)
228                     }
229                     <TouchableOpacity onPress={ () => this.props.navigation.navigate("ActivityPicker",
{backPreCall: this.refreshUser})}>
230                       <View style={styles.iconMainContainer}>
231                           <View style={styles.charContainer}>
232                             <Text style={[styles.activityChar,{fontFamily:'awesome'}]}>{'\uF067'}</Text>
233                           </View>
234                           <Text style={styles.activityName}>Pick New</Text>
235                       </View>
236                     </TouchableOpacity>
237                   </View>
238                 </ScrollView>
```

```
239                 </View>
240       );}else{
241         return(
242           <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
243             <ActivityIndicator color={'#ff4d4d'}/>
244           </View>
245         );
246       }
247     }
248   }
249
250   const styles = StyleSheet.create({
        ...
391     });
```

Home/Settings/index.js

```
1     import React, { Component } from 'react';
2     import { NavigationActions } from 'react-navigation';
3     import {
4       AsyncStorage,
5       ActivityIndicator,
6       Button,
7       DatePickerAndroid,
8       Image,
9       StyleSheet,
10      Text,
11      TextInput,
12      TouchableOpacity,
13      ToastAndroid,
14      View,
15    } from 'react-native';
16
17    export default class Main extends Component {
18
19      constructor(props){
20        super(props);
21        this.state = {
22          done: false,
23          avatarChanged: false
24        };
25        this.updatePersonalInfo = this.updatePersonalInfo.bind(this);
26        this.selectAvatar = this.selectAvatar.bind(this);
27      }
28
29      static navigationOptions = function(props) {
30        return({
31          title: 'Settings',
32          headerStyle: styles.header,
33          headerTitleStyle: styles.headerTitle,
34          headerBackTitleStyle: styles.headerButton,
35          headerLeft:
36            <TouchableOpacity onPress={ () => {
37                if( "backPreCall" in props){
38                  props.backPreCall();
39                }else if ( "backPreCall" in props.navigation.state.params) {
40                  props.navigation.state.params.backPreCall();
41                }
42                props.navigation.dispatch(NavigationActions.back());
43              }}>
44              <Text style={styles.headerButton}>{'\uF060'}</Text>
45            </TouchableOpacity>,
46          headerRight:
47            <TouchableOpacity onPress={ () => {props.navigation.state.params.updatePersonalInfo();}}>
48              <Text style={styles.headerButton}>{'\uF00C'}</Text>
49            </TouchableOpacity>,
50          });
51      }
52
53      async updatePersonalInfo(){
54        const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
55        const token = await AsyncStorage.getItem('@SocialAgent:token');
56        try{
57          const body = new FormData();
58          body.append('first_name', this.state.first_name);
59          body.append('last_name', this.state.last_name);
60          body.append('email', this.state.email);
61          body.append('dateOfBirth', this.state.dateOfBirth);
62          if(this.state.avatarChanged){
63            body.append('avatar', {
64              uri: this.state.avatar.uri,
65              type: this.state.avatar.type, // or photo.type
```

```
66            name: this.state.avatar.name
67          });
68        }
69        let response = await fetch(
70          server_address+'me/',
71          {
72            method: 'PATCH',
73            headers: {
74              'Accept': 'application/json',
75              'Content-Type': 'multipart/form-data',
76              'Authorization': 'Token ' + token,
77            },
78            body: body
79          });
80        if(!response.ok) {
81          console.log(response.json());
82          ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
83          return;
84        }
85        let responseJson = await response.json();
86        await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(responseJson));
87        this.props.navigation.state.params.backPreCall();
88        this.props.navigation.dispatch(NavigationActions.back());
89      }catch(error){
90        ToastAndroid.show('Something went wrong..',ToastAndroid.SHORT);
91        console.error(error);
92      }
93      return;
94    }
95
96    selectAvatar(){
97      var ImagePicker = require('react-native-image-picker');
98      var options = {
99        title: 'Select Avatar',
100       storageOptions: {
101         skipBackup: true,
102         path: 'images'
103       }
104     };
105     ImagePicker.showImagePicker(options, (response) => {
106       console.log('Response = ', response);
107       if (response.didCancel) {
108         console.log('User cancelled image picker');
109       }
110       else if (response.error) {
111         console.log('ImagePicker Error: ', response.error);
112       }
113       else {
114         //let source = { uri: response.uri };
115         // You can also display the image using data:
116         // let source = { uri: '' + response.data };
117         this.setState({
118           avatarChanged: true,
119           avatar: {
120             uri: response.uri,
121             type: response.type,
122             name: response.fileName
123           }
124         });
125       }
126     });
127   }
128
129   componentDidMount() {
130     this.props.navigation.setParams({ updatePersonalInfo: this.updatePersonalInfo });
131   }
132
133   async componentWillMount(){
134     try{
135       const user = JSON.parse(await AsyncStorage.getItem('@SocialAgent:user'));
136       this.setState({
137         done: true,
138         first_name: user.first_name,
139         last_name: user.last_name,
140         email: user.email,
141         dateOfBirth: user.dateOfBirth,
142         avatar: {
143           uri: user.avatar
144         }
145       })
146     }catch(error){
147       console.error(error);
148     }
149     return;
```

```
150     }
151
152     async changeDate(){
153       try {
154         const {action, year, month, day} = await DatePickerAndroid.open({
155           // Use `new Date()` for current date.
156           // May 25 2020. Month 0 is January.
157           date: Date.parse(this.state.dateOfBirth)
158         });
159         if (action !== DatePickerAndroid.dismissedAction) {
160           this.setState({ dateOfBirth: year+'-'+(month+1)+'-'+day});
161         }
162       } catch ({code, message}) {
163         console.warn('Cannot open date picker', message);
164       }
165     }
166
167     render() {
168       if(!this.state.done){
169         return(
170           <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
171             <ActivityIndicator color={'#ff4d4d'}/>
172           </View>
173         );
174       }else{
175         return (
176           <View style={styles.container}>
177             <Text style={styles.title}>
178               Update your personal info
179             </Text>
180             <Text style={styles.subtitle}>
181               First Name
182             </Text>
183             <TextInput
184               style={styles.input}
185               value={this.state.first_name}
186               onChangeText={(text) => this.setState({first_name: text})}
187               />
188             <Text style={styles.subtitle}>
189               Last Name
190             </Text>
191             <TextInput
192               style={styles.input}
193               value={this.state.last_name}
194               onChangeText={(text) => this.setState({last_name: text})}
195               />
196             <Text style={styles.subtitle}>
197               Date of Birth
198             </Text>
199             <TouchableOpacity onPress={() => {this.changeDate();}}>
200               <Text style={[styles.input,{color: '#000000', borderBottomWidth: 1, marginBottom: 6}]}>
201                 {this.state.dateOfBirth}
202               </Text>
203             </TouchableOpacity>
204             <Text style={styles.subtitle}>
205               Email
206             </Text>
207             <TextInput
208               style={styles.input}
209               value={this.state.email}
210               onChangeText={(text) => this.setState({email: text})}
211               onFocus={() => this.changeDate()}
212               />
213             <Text style={styles.subtitle}>
214               Avatar
215             </Text>
216             <TouchableOpacity onPress={() => {this.selectAvatar();}}>
217               <Image style={styles.avatar} source={{uri: this.state.avatar.uri}}/>
218             </TouchableOpacity>
219           </View>
220         );
221       }
222     }
223   }
224
225   const styles = StyleSheet.create({
        ...
268   });
269
```

UserIcon/index.js

```
1    import React, { Component } from 'react';
2    import {
3      AsyncStorage,
4      ActivityIndicator,
5      StyleSheet,
6      Text,
7      View,
8      Image,
9      TouchableOpacity
10   } from 'react-native';
11
12   export default class UserIcon extends Component {
13     constructor(props){
14       super(props);
15       this.state = {
16          person: null,
17          user: null
18       };
19     }
20
21     async componentWillMount(){
22       const token = await AsyncStorage.getItem('@SocialAgent:token');
23       try{
24         let response = await fetch(
25           this.props.uri,
26           {
27             method: 'GET',
28             headers: {
29               'Accept': 'application/json',
30               'Content-Type': 'application/json',
31               'Authorization': 'Token ' + token,
32             },
33           });
34         if(!response.ok) return;
35         let responseJson = await response.json();
36         this.setState({person: responseJson});
37       }catch(error){
38         console.error(error);
39       }
40       return;
41     }
42
43     render(){
44       if(this.state.person == null) {
45         return (
46           <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
47             <ActivityIndicator color={'#ff4d4d'}/>
48           </View>
49         );
50       }else{
51         return(
52           <TouchableOpacity onPress={() => {
53             if("backPreCall" in this.props){
54               this.props.navigation.navigate('UserPage',{
55                 person: this.state.person,
56                 backPreCall: this.props.backPreCall
57               });
58             }else{
59               this.props.navigation.navigate('UserPage',{person: this.state.person});
60             }
61           }}>
62             <View style={styles.followerContainer}>
63               <Image style={styles.followerAvatar} source={{uri:this.state.person.avatar}}/>
64               <Text style={styles.followerName}>{this.state.person.first_name}
{this.state.person.last_name}</Text>
65             </View>
66           </TouchableOpacity>
67         );
68       }
69     }
70   }
71
72   const styles= StyleSheet.create({
       ...
91   });
```

UserIconLarge/index.js

```
1    import React, { Component } from 'react';
```

```
2    import {
3      AsyncStorage,
4      ActivityIndicator,
5      StyleSheet,
6      Text,
7      View,
8      Image,
9      TouchableOpacity
10   } from 'react-native';
11   import { getDistanceFromLatLonInKm } from '../../assets/support.js';
12
13   export default class UserIconLarge extends Component {
14     constructor(props){
15       super(props);
16       this.state = {
17          person: null,
18          user: null
19       };
20     }
21
22     async componentWillMount(){
23       const token = await AsyncStorage.getItem('@SocialAgent:token');
24       const user = await AsyncStorage.getItem('@SocialAgent:user');
25       this.setState({user: JSON.parse(user)})
26       try{
27         let response = await fetch(
28           this.props.uri,
29           {
30             method: 'GET',
31             headers: {
32               'Accept': 'application/json',
33               'Content-Type': 'application/json',
34               'Authorization': 'Token ' + token,
35             },
36           });
37         if(!response.ok) return;
38         let responseJson = await response.json();
39         this.setState({person: responseJson});
40       }catch(error){
41         console.error(error);
42       }
43       return;
44     }
45
46     render(){
47       if(this.state.person == null || this.state.user == null) {
48         return (
49           <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
50             <ActivityIndicator color={'#ff4d4d'}/>
51           </View>
52         );
53       }else{
54         return(
55           <TouchableOpacity onPress={() => {
56             if("backPreCall" in this.props){
57               this.props.navigation.navigate('UserPage',{
58                 person: this.state.person,
59                 backPreCall: this.props.backPreCall
60               });
61             }else{
62               this.props.navigation.navigate('UserPage',{person: this.state.person});
63             }
64           }}>
65             <View style={styles.followerContainer}>
66               <Image style={styles.followerAvatar} source={{uri:this.state.person.avatar}}/>
67               <Text style={styles.followerName}>{this.state.person.first_name}
{this.state.person.last_name}</Text>
68               <Text style={styles.followerName}>{
69                 getDistanceFromLatLonInKm(
70                   this.state.user.latitude,
71                   this.state.user.longitude,
72                   this.state.person.latitude,
73                   this.state.person.longitude)} away
74               </Text>
75             </View>
76           </TouchableOpacity>
77         );
78       }
79     }
80   }
81
82   const styles= StyleSheet.create({
       ...
101  });
```

102

UserPage/index.js

```
1   import React, { Component } from 'react';
2   import { NavigationActions } from 'react-navigation';
3   import {
4     ActivityIndicator,
5     AsyncStorage,
6     ScrollView,
7     StyleSheet,
8     Text,
9     View,
10    Image,
11    TouchableOpacity,
12    ToastAndroid
13  } from 'react-native';
14  import { getDistanceFromLatLonInKm } from '../../assets/support.js';
15  import { getAgeFromDateOfBirth } from '../../assets/support.js';
16  import ActivityIcon from '../ActivityIcon/';
17
18  export default class UserPage extends Component {
19    constructor(props){
20      super(props);
21      this.state ={
22        person: props.navigation.state.params.person,
23        user: null,
24        followed: null
25      };
26    }
27
28    async _handleFollowPush(){
29      try{
30        const server_address = await AsyncStorage.getItem('@SocialAgent:server-address');
31        const token = await AsyncStorage.getItem('@SocialAgent:token');
32        var user = this.state.user;
33        const person_url = this.state.person.url;
34
35        if(this.state.followed){ // Unfollow
36          var url = user.following.filter(function( obj ) {
37            return obj.followee == person_url;
38          })[0].url;
39          let response = await fetch(
40            url,
41            {
42              method: 'DELETE',
43              headers: {
44                'Accept': 'application/json',
45                'Content-Type': 'application/json',
46                'Authorization': 'Token ' + token,
47              },
48            });
49            if(response.ok){
50              let response = await fetch(
51                server_address+'me/',
52                {
53                  method: 'GET',
54                  headers: {
55                    'Accept': 'application/json',
56                    'Content-Type': 'application/json',
57                    'Authorization': 'Token ' + token,
58                  },
59                });
60              let responseJson = await response.json();
61              this.setState({
62                followed:false,
63                user: responseJson
64              });
65              await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(responseJson));
66              ToastAndroid.show("Unfollowed.",ToastAndroid.SHORT);
67            }else{
68              ToastAndroid.show("Something went wrong..",ToastAndroid.SHORT);
69            }
70        }else{ // Follow
71          let response = await fetch(
72            server_address + 'follows/',
73            {
74              method: 'POST',
75              headers: {
76                'Accept': 'application/json',
77                'Content-Type': 'application/json',
78                'Authorization': 'Token ' + token,
```

```
79              },
80              body: JSON.stringify({
81                follower: user.url,
82                followee: person_url
83              })
84            });
85            if(response.ok){
86              let response = await fetch(
87                server_address+'me/',
88                {
89                  method: 'GET',
90                  headers: {
91                    'Accept': 'application/json',
92                    'Content-Type': 'application/json',
93                    'Authorization': 'Token ' + token,
94                  },
95                });
96              let responseJson = await response.json();
97              this.setState({
98                followed: true,
99                user: responseJson
100             });
101             await AsyncStorage.setItem('@SocialAgent:user',JSON.stringify(responseJson));
102             ToastAndroid.show("Followed.",ToastAndroid.SHORT);
103           }else{
104             ToastAndroid.show("Something went wrong..",ToastAndroid.SHORT);
105           }
106       }
107     }catch(error){
108       console.error(error);
109     }
110   }
111
112   async componentWillMount(){
113     try{
114       const user = await AsyncStorage.getItem('@SocialAgent:user');
115       var followed_users = JSON.parse(user).following;
116       if(followed_users.some((e) => e.followee === this.state.person.url)){
117         this.setState({
118           followed: true,
119           user: JSON.parse(user)
120         });
121       }else{
122         this.setState({
123           followed: false,
124           user: JSON.parse(user)
125       });
126       }
127     }catch(error){
128       console.error(error);
129     }
130     return;
131   }
132
133   static navigationOptions = function(props) {
134     return({
135       title: 'User',
136       headerStyle: styles.header,
137       headerTitleStyle: styles.headerTitle,
138       headerLeft:
139           <TouchableOpacity onPress={ () => {
140                 if( "backPreCall" in props){
141                   props.backPreCall();
142                 }else if("backPreCall" in props.navigation.state.params){
143                   props.navigation.state.params.backPreCall();
144                 }
145                 props.navigation.dispatch(NavigationActions.back());
146               }}>
147               <Text style={styles.headerButton}>{'\uF060'}</Text>
148             </TouchableOpacity>,
149       headerBackTitleStyle: styles.headerButton,
150       });
151   }
152
153   render() {
154     if(this.state.user != null){
155       return (
156         <View style={styles.mainContainer}>
157           <ScrollView>
158           <View style={styles.scrollItemMargin}>
159             <Text style={styles.profileName}>{ this.state.person.first_name}
160                {" "+this.state.person.last_name}
{getAgeFromDateOfBirth(this.state.person.dateOfBirth)}</Text>
161           </View>
```

```
162            <View style={styles.scrollItemMargin}>
163              <Text style={[styles.profileName,{fontWeight:'normal'}]}>{getDistanceFromLatLonInKm(
164                    this.state.user.latitude,
165                    this.state.user.longitude,
166                    this.state.person.latitude,
167                    this.state.person.longitude)} away</Text>
168            </View>
169            <View style={[styles.avatarContainer,styles.scrollItemMargin,styles.horizontalContainer]}>
170              <Image style={styles.avatar} source={{uri: this.state.person.avatar}}/>
171            </View>
172            <View style={[styles.followsContainer,styles.horizontalContainer,styles.scrollItemMargin]}>
173              <View style={[styles.verticalContainer,{flex:1}]}>
174                <Text style={{textAlign:'center',color: '#1a1a1a',fontSize:16}}>Following</Text>
175                <Text style={{textAlign:'center',color:
'#1a1a1a',fontSize:20}}>{this.state.person.following.length}</Text>
176              </View>
177              <View style={[styles.verticalContainer,{flex:1}]}>
178                <Text style={{textAlign:'center',color: '#1a1a1a',fontSize:16}}>Followers</Text>
179                <Text style={{textAlign:'center',color:
'#1a1a1a',fontSize:20}}>{this.state.person.followers.length}</Text>
180              </View>
181            </View>
182            <View style={[styles.verticalContainer,styles.horizontalContainer,
183                          styles.scrollItemMargin,{justifyContent:'space-around'}]}>
184              <TouchableOpacity onPress={ this._handleFollowPush.bind(this) }>
185                <View style={this.state.followed?styles.unfollowButtonWrapper:styles.followButtonWrapper}>
186                  <Text style={[styles.scrollItemMargin,this.state.followed?
styles.unfollowButtonIcon:styles.followButtonIcon]}>
187                    {this.state.followed?'\uF056':'\uF055'}
188                  </Text>
189                  <Text style={styles.followButtonText}>
190                    {this.state.followed ?'Unfollow':'Follow'}
191                  </Text>
192                </View>
193              </TouchableOpacity>
194            </View>
195            <View style={[styles.horizontalContainer,styles.scrollItemMargin]}>
196              <Text style={[styles.itemTitle,{flex:1}]}>Activities:</Text>
197            </View>
198            <View style={[styles.horizontalContainer,styles.scrollItemMargin,{flexWrap:'wrap'}]}>
199              {
200                this.state.person.activities.map(function(activity, index){
201                  return <ActivityIcon navigation={this.props.navigation} key={activity.activity}
uri={activity.activity}/>;
202                },this)
203              }
204            </View>
205          </ScrollView>
206          </View>
207        );
208      }else{
209        return (
210          <View style={{alignItems:'center',justifyContent:'center',flex:1,backgroundColor:'#f2f2f2'}}>
211            <ActivityIndicator color={'#ff4d4d'}/>
212          </View>
213        );
214      }
215    }
216  }
217
218  const styles= StyleSheet.create({
       ...
322  });
```

Βιβλιογραφία

[1]: Wasserman, S. & Faust, K., Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences), 1994

[2]: Kadushin, C., Understanding social networks: Theories, concepts, and findings., 2012

[3]: Obar, Jonathan A. & Wildman, Steve, Social media definition and the governance challenge: An introduction to the special issue, 2015

[4]: Romm-Livermore, C. & Setzekorn, K., Social Networking Communities and E-Dating Services: Concepts and Implications, 2008

[5]: Social Networking Service - Wikipedia , https://wikipedia.org/wiki/Social_networking_service

[6]: Wikipedia - Youtube , https://wikipedia.org/wiki/YouTube

[7]: Alexa Internet - Youtube , https://www.alexa.com/siteinfo/youtube.com

[8]: D. Rosenblum, What Anyone Can Know: The Privacy Risks of Social Networking Sites, 2007

[9]: Henry Jenkins & Danah Boyd, Discussion: MySpace and Deleting Online Predators Act (DOPA), 2006-05-26

[10]: Wikipedia - Facebook Beacon , https://en.wikipedia.org/wiki/Facebook_Beacon

[11]: React Native - Facebook GitHub IO , https://facebook.github.io/react-native/

[12]: Wikipedia - React , https://en.wikipedia.org/wiki/React_(JavaScript_library)

[13]: React - Intoducing JSX , https://reactjs.org/docs/introducing-jsx.html

[14]: React Native - Props , https://facebook.github.io/react-native/docs/props.html

[15]: React Native - State , https://facebook.github.io/react-native/docs/state.html

[16]: React Native - Style , https://facebook.github.io/react-native/docs/style.html

[17]: Django Documentation , https://docs.djangoproject.com/en/2.0/

[18]: Wikipedia - Django (Web-Framework) , https://en.wikipedia.org/wiki/Django_(web_framework)

[19]: DjangoBook - MVC Design Pattern , https://djangobook.com/model-view-controller-design-pattern/

[20]: REST - MDN Web Docs , https://developer.mozilla.org/en-US/docs/Glossary/REST

[21]: What is Rest? - Code Academy , https://www.codecademy.com/articles/what-is-rest

[22]: Understanging REST , https://spring.io/understanding/REST

[23]: React Navigation , https://reactnavigation.org/

[24]: Tab Navigation - React Navigation , https://reactnavigation.org/docs/tab-navigator.html

[25]: Stack Navigator - React Navigation , https://reactnavigation.org/docs/stack-navigator.html

[26]: Use Experience Questionnaire (UEQ) , http://www.ueq-online.org/