Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάλυση συναισθημάτων σύντομων μηνυμάτων σε εφαρμογή Android με χρήση τεχνολογιών Google Cloud Messaging και Data Mining (Μέρος Β)** **Sentiment analysis of instant messages on android application using Google Cloud Messaging technologies and Data Mining (Part B)**
Ονοματεπώνυμο Φοιτητή	**ΜΠΟΖΙΟΝΕΛΟΣ ΣΤΕΦΑΝΟΣ**
Πατρώνυμο	**Κωνσταντίνος**
Αριθμός Μητρώου	**ΜΠΣΠ13070**
Επιβλέπων	**Αλέπης Ευθύμιος, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Σεπτέμβριος 2016**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Αλέπης Ευθύμιος Επίκουρος Καθηγητής	Πατσάκης Κωνσταντίνος Επίκουρος Καθηγητής	Τσιχριντζής Γεώργιος Καθηγητής

Περίληψη

Η διπλωματική αυτή εργασία αποτελεί το 2ο Μέρος της εργασίας με τίτλο «Ανάλυση συναισθημάτων σύντομων μηνυμάτων σε εφαρμογή Android με χρήση τεχνολογιών Google Cloud Messaging και Data Mining» και συνέχεια της εργασίας του Παρασκάκη Ιωάννη. Στη παρούσα διπλωματική εργασία, έχοντας θέσει το θεωρητικό υπόβαθρο της Πλατφόρμας Thesis παρουσιάζουμε τις συνιστώσες του εν λόγω πληροφοριακού συστήματος αλλά και τη διασύνδεση τους με τους διάφορους Classifiers.

Abstract

This thesis documentation is the second part of the dissertation titled “Sentiment analysis of instant messages on android application using Google Cloud Messaging technologies and Data Mining” and the continuation of Paraskakis Ioannis’ work. In this paper, having set the theoretical background of the Thesis Platform, we present the different components of the aforementioned platform and their interconnections and communication protocols.

Περιεχόμενα

Κεφάλαιο 1°: Εισαγωγή

Προσπαθώντας να εκμεταλλευτούμε όλα τα οφέλη που μας δίνει η χρήση των σύγχρονων κινητών συσκευών και ταυτόχρονα να συγκεντρώσουμε δεδομένα τα οποία θα μας βοηθήσουν να δημιουργήσουμε ένα μοντέλο αναγνώρισης συναισθημάτων αποφασίσαμε στα πλαίσια της παρούσας διπλωματικής εργασίας να δημιουργήσουμε την πλατφόρμα Thesis Messaging Platform. Συγκεκριμένα, στόχος ήταν η δημιουργία μίας εφαρμογής ανταλλαγής μηνυμάτων για σύγχρονες κινητές συσκευές, η οποία θα συνδύαζε την υλοποίηση αποστολής/λήψης σύντομων μηνυμάτων κειμένου και τη τροφοδότηση ενός μοντέλου αναγνώρισης συναισθημάτων, ενώ παράλληλα θα υλοποιούσε την καταγραφή δεδομένων από τους διαθέσιμους αισθητήρες των κινητών συσκευών ώστε να δημιουργηθεί ένα ασφαλές σύνολο δεδομένων για περεταίρω μελέτη.

Ένας από τους βασικούς στόχους μας εξ αρχής ήταν να υποστηρίξουμε άμεσα το μεγαλύτερο σύνολο σύγχρονων κινητών συσκευών. Για τον λόγο αυτό αποφασίσαμε η αρχική υλοποίηση να γίνει για το λειτουργικό σύστημα Android. Το Android κατέχει τα τελευταία χρόνια περίπου το 80% της παγκόσμιας αγοράς, με βασικό ανταγωνιστή του το iOS το οποίο κατέχει με τη σειρά του περίπου 15%. Αποφασίσαμε να υποστηρίξουμε από την έκδοση 15 έως την πιο πρόσφατη καθώς το πλήθος χρηστών που χρησιμοποιεί έκδοση μικρότερη της 15 δεν ξεπερνά το 2%. Με γνώμονα το πλήθος συσκευών που θα υποστηρίζονται καταλήξαμε και σε τεχνολογίες που θα πλαισιώνουν την εφαρμογή. Μία από τις βασικές επιλογές ήταν η χρήση του GCM (Google Cloud Messaging) το οποίο υποστηρίζει κοινοποιήσεις (push notifications) μεταξύ συσκευών, χαρακτηριστικό το οποίο είναι απαραίτητο στις σύγχρονες εφαρμογές ανταλλαγής μηνυμάτων. Το κύριο πλεονέκτημα του GCM είναι η υποστήριξη και των δύο κυρίαρχων λειτουργικών συστημάτων Android και iOS.

2.1 Διάρθρωση της διπλωματικής

Η παρούσα διπλωματική αποτελείται από 3 βασικά κεφάλαια. Στο επόμενο και δεύτερο κεφάλαιο της παρούσας διπλωματικής περιγράφουμε αναλυτικά το σύστημα αποτελεί τη πλατφόρμα Thesis και αναλύουμε το πρωτόκολλο επικοινωνίας μεταξύ των διάφορων εφαρμογών αλλά και τις διεργασίες που εκτελούνται σε κάθε βήμα του κάθε πρωτοκόλλου. Παρέχουμε κομμάτια κώδικα για κατανόηση των διαδικασιών και εξηγούμε την υλοποίηση και την προσαρμογή μοντέλων αναγνώρισης συναισθημάτων στην υλοποίηση μας.

Στο τρίτο και τελευταίο μέρος της εργασίας μας παρουσιάζουμε συγκεντρωτικά τα αποτελέσματα της χρήσης της εφαρμογής σε πραγματικό χρόνο και αναλύουμε τα συμπεράσματα που προέκυψαν τόσο κατά την υλοποίηση όσο και κατά τη δοκιμή.

Περιγραφή του συστήματος ανταλλαγής μηνυμάτων

3.1 Απαραίτητο θεωρητικό υπόβαθρο

Προκειμένου να κατανοήσει ο αναγνώστης το περιεχόμενο αυτής της διπλωματικής εργασίας και δεδομένης της λειτουργίας του πληροφοριακού συστήματος που ονομάζεται Thesis Platform, σε διαφορετικές πλατφόρμες, λειτουργικά συστήματα και συσκευές καθώς και τη χρήση διαφόρων εργαλείων, θα πρέπει να έχει κάποιο επίπεδο γνώσεων στα παρακάτω θέματα.

3.1.1 Java/Android

Το πρώτο και βασικότερο συστατικό της πλατφόρμας Thesis είναι η εφαρμογή για κινητές συσκευές Android και είναι γραμμένο σε γλώσσα προγραμματισμού JAVA σε περιβάλλον Android Studio. Ο αναγνώστης της παρούσας διπλωματικής θα πρέπει να γνωρίζει τις σύγχρονες βέλτιστες πρακτικές της ανάπτυξης εφαρμογών για android συσκευές και να είναι εξοικειωμένος σε θέματα δικτυακών εφαρμογών που υλοποιούν το πρωτόκολλο πελάτης-εξυπηρετητής με χρήση Web Services. Τέλος ο αναγνώστης θα πρέπει να γνωρίζει τις διαδικασίες και τις λειτουργίες των βιβλιοθηκών GooglePlay μέλος της οποίας αποτελεί η βιβλιοθήκη GMS που διαχειρίζεται τη διασύνδεση με την υπηρεσία Google Cloud Messaging.

3.1.2 PHP/Symfony Framework

Η υλοποίηση στη μεριά του εξυπηρετητή έχει γίνει με χρήση του Symfony Framework και ο εξυπηρετητής είναι γραμμένος σε γλώσσα PHP. Η σύνδεση με τη βάση δεδομένων γίνεται μέσω του Doctrine ORM. Ο αναγνώστης της παρούσας διπλωματικής θα πρέπει να είναι εξοικειωμένος με την αρχιτεκτονική του MVC (Model-View-Controller Architecture) και με το θεωρητικό υπόβαθρο της μετάδοσης μηνυμάτων μέσω του HTTP πρωτοκόλλου που διέπει την επικοινωνία ανάμεσα στον Server και την Android εφαρμογή.

3.1.3 Διαχείριση βάσεων δεδομένων

Το σύστημα μας κάνει εκτενή χρήση βάσεων δεδομένων προκειμένου να διατηρεί σε ασφαλές μέρος τα δεδομένα που απαιτούνται για τη σωστή λειτουργία του. Ο αναγνώστης θα πρέπει να έχει τις βασικές γνώσεις για τη διαχείριση, τη συσχέτιση πινάκων, την εισαγωγή και την εξαγωγή δεδομένων από αυτούς. Για την διατήρηση των δεδομένων χρησιμοποιούμε το σύστημα διαχείρισης βάσεων δεδομένων MySQL.

3.1.4 XML, JSON

Η επικοινωνία μεταξύ της εφαρμογής και των εξυπηρετητών υλοποιείται με την ανταλλαγή μηνυμάτων σε μορφή XML. Ο αναγνώστης θα πρέπει να έχει βασικές γνώσεις σε περιγραφικές γλώσσες όπως η XML αλλά και σε μορφοποιήσεις ανταλλαγής δεδομένων όπως η JSON. Παρότι η JSON δεν χρησιμοποιείται σε οποιαδήποτε λειτουργία της εφαρμογής, χρησιμοποιείται αρκετά στην καταγραφή των αισθητήρων των συσκευών και γενικότερα σε καταστάσεις όπου η εφαρμογή καταγράφει ακατέργαστα δεδομένα, όπως για παράδειγμα το σύνολο των τηλεφωνικών αριθμών των επαφών μίας συσκευής.

3.1.5 Server Configuration

Προκειμένου ο εξυπηρετητής να υποστηρίζει τη μετάδοση μηνυμάτων μέσω του Google Cloud Messaging έπρεπε να διαμορφωθεί με τέτοιο τρόπο ώστε να υποστηρίζεται η αυθεντικοποίηση και η διασύνδεση του με τις υπηρεσίες της Google. Ο αναγνώστης αυτής της διπλωματικής θα πρέπει να είναι εξοικειωμένος με το LAMP stack αλλά και με βασικές έννοιες διαχείρισης διανομών Linux για εξυπηρετητές.

Η πλατφόρμα Thesis στηρίζεται στην απλή αρχιτεκτονική του πελάτη-εξυπηρετητή μέσω διαδικτύου και αποτελείται από 3 βασικά μέρη. Έναν εξυπηρετητή ο οποίος διαχειρίζεται όλες τις αιτήσεις των πελατών, μία εφαρμογή πελάτη και ένα σύστημα αναγνώρισης συναισθημάτων από κείμενο.

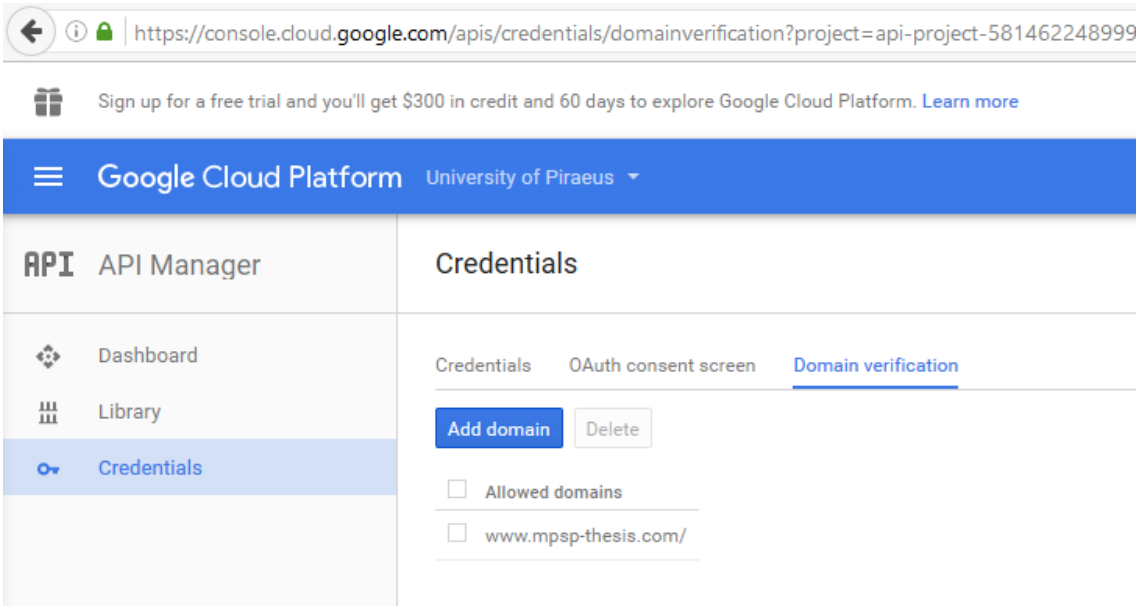
3.2 Λογισμικό

Το λογισμικό που χρησιμοποιήσαμε για τη δημιουργία του συστήματος που υποστηρίζει την εφαρμογή Thesis αλλά και για τη δημιουργία της ίδιας της εφαρμογής και των λειτουργιών της είναι:

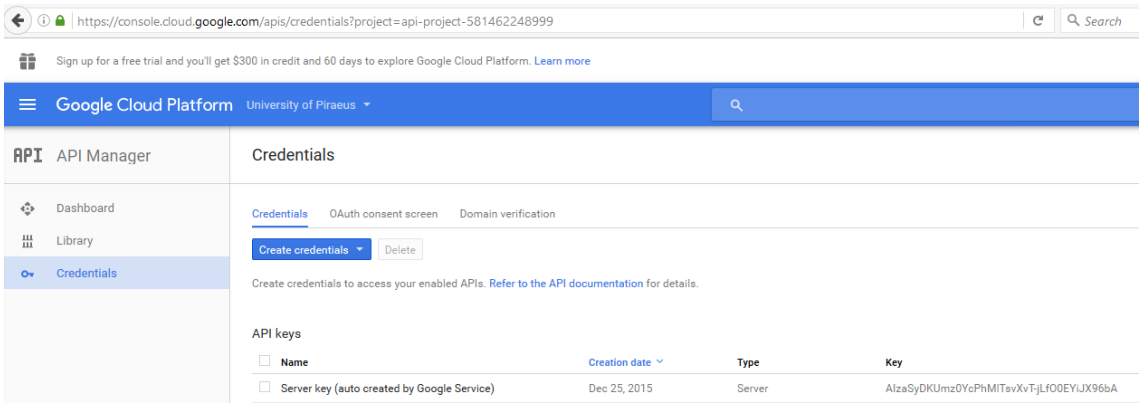
- Για την εφαρμογή του χρήστη χρησιμοποιούμε Android Studio 1.4 για SDK version 21.
- Ο εξυπηρετητής έχει εγκατεστημένο το LAMP stack (ubuntu 14.04LTS-server 64BITS) και έχει διαμορφωθεί ώστε να είναι συμβατός με το Symfony Framework
- Το Base Domain (mpsp-thesis.com) έχει αντιστοιχιστεί με τον προαναφερθέντα εξυπηρετητή και έχει πιστοποιηθεί από το Google ώστε να είναι συμβατό με το Google Cloud Messaging Service.
- Η βάση δεδομένων είναι MySQL.

3.3 Web Server

Η κεντρική εφαρμογή εξυπηρετητή που διαχειρίζεται όλα τα αιτήματα είναι γραμμένη σε γλώσσα PHP με χρήση του Symfony Framework 2.7 και βρίσκεται σε ένα διακομιστή που χρησιμοποιεί λειτουργικό σύστημα Ubuntu 14.04 και έχει εγκατεστημένο το LAMP stack. Ο εξυπηρετητής δέχεται αιτήματα με χρήση του πρωτοκόλλου HTTP και το Routing του Symfony Framework διαχειρίζεται τη δρομολόγηση των αιτημάτων αυτών μέσω των διάφορων Controller και Firewalls που έχουμε παραμετροποιήσει. Ο εξυπηρετητής περιέχει ένα Virtual Host configuration προκειμένου να γίνεται resolve το domain name "mpsp-thesis.com" στο οποίο είναι εγκατεστημένη η εφαρμογή του εξυπηρετητή. Ακόμα έχουν γίνει όλες οι απαραίτητες ενέργειες ώστε τόσο το domain όσο και ο διακομιστής να εγκριθούν και πιστοποιηθούν από τη Google μέσω του Google Developer's Console. Έχοντας λοιπόν παραμετροποιήσει τον εξυπηρετητή να μπορεί να συνδεθεί με τους Servers της Google μπορούμε να δεχόμαστε αιτήματα από τις κινητές συσκευές και αυτά να προωθούνται μέσω του www.mpsp-thesis.com στο GCM και να ενημερώνονται οι διάφοροι παραλήπτες αυτών των μηνυμάτων.



GCM Domain Verification



GCM Server Verification

Ο εξυπηρετητής είναι ακόμα συνδεδεμένος με μία βάση δεδομένων MySQL η οποία βρίσκεται επίσης στον διακομιστή και επικοινωνεί μαζί της μέσω του Doctrine ORM το οποίο είναι υπεύθυνο για τις αλληλεπιδράσεις του εξυπηρετητή με τη βάση δεδομένων. Ακόμα το ORM λειτουργεί σαν ένα στρώμα ασφαλείας ανάμεσα στην εφαρμογή και τη βάση δεδομένων. Η εφαρμογή του εξυπηρετητή είναι στη πραγματικότητα ένα σύνολο από Web Services τα οποία δέχονται αιτήματα από τις κινητές συσκευές (με συγκεκριμένες παραμέτρους) και τα εξυπηρετούν υλοποιώντας όλη τη λογική και τη λειτουργικότητα της πλατφόρμας. Οι απαντήσεις που παράγονται από τη κλήση των διάφορων endpoints των Web Services είναι σε XML μορφή και χρησιμοποιείται ο JMS Serializer για τη μετατροπή των αντικειμένων σε XML.

www.mpsp-thesis.com/api/createDevice?gcmtoken=fms2s8lg90Q%3AAPA91bHidvbF-gjKWWzHDHj45Zu15zDkmCbUE98tVlXRf8_DlNa3z3dVput8IaN7tF45Lz3Ck6ii9X2BE

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
− <Device device_id="493434b062336ef8" is_user_registered="false" id="1">
  − <gcm_token>
      fms2s8lg90Q:APA91bHidvbF-gjKWWzHDHj45Zu15zDkmCbUE98tVlXRf8_DlNa3z3dVput8IaN7tF45Lz3Ck6ii9X2BEg4j7VqMTNyjmQs4SY1Ct77NTC0WPAoTHeTKYK
    </gcm_token>
  − <User tkey="2ae644c5f82ef9ea8d0e0955ed3479d97e49f5c8" username="jparaskakis" cdate="2016-09-02T13:25:41+0200" avatar="http://62.210.36.88/thesis_assets/user1.jpg
      <phone>+306986894912</phone>
    − <Conversations>
      − <Conversation isregistered="true" identifier="Test Conversation" id="1">
        − <Messages>
          − <Message senderid="2" createdtimestamp="2016-09-04T12:13:46+0200" id="3">
            − <Receivers>
                <Receiver readbyreceiver="true" receiverid="1"/>
              </Receivers>
              <data>A simple message 2</data>
            </Message>
          − <Message senderid="2" createdtimestamp="2016-09-04T12:12:23+0200" id="2">
              <data>A simple message</data>
            </Message>
          − <Message senderid="1" createdtimestamp="2016-09-04T11:54:54+0200" id="7">
              <data>test300</data>
            </Message>
          − <Message senderid="1" createdtimestamp="2016-09-04T11:54:41+0200" id="6">
              <data>test30</data>
            </Message>
          − <Message senderid="1" createdtimestamp="2016-09-04T11:54:11+0200" id="5">
              <data>test3</data>
```


Server XML response

Ο εξυπηρετητής είναι τέλος συνδεδεμένος με την εφαρμογή του RapidMiner την οποία χρησιμοποιεί για να αναγνωρίσει τη γλώσσα στην οποία είναι γραμμένο το μήνυμα αρχικά και την κατηγοριοποίηση των συναισθημάτων που αυτό περιέχει στη συνέχεια κατά την αποστολή του μηνύματος.

3.3.1 Apache Tomcat

Ο Application Server χρησιμοποιείται στη θύρα 8080 του διακομιστή. Ο εν λόγω διακομιστής περιέχει μία εφαρμογή JAVA η οποία εκτελείται στον Apache Tomcat και η λειτουργία της χωρίζεται σε δύο (2) φάσεις. Η πρώτη φάση αποτελεί τη διαδικασία αναγνώρισης της γλώσσας του κειμένου, ανάμεσα σε 14 γλώσσες, ενώ η δεύτερη φάση αποτελεί την κατηγοριοποίηση του κειμένου σε μια από τις διαθέσιμες κλάσεις συναισθημάτων. Τόσο για τη πρώτη όσο και για τη δεύτερη φάση έχουν δημιουργηθεί τα αντίστοιχα μοντέλα και classifiers (περιγράφονται εκτενέστερα στο πρώτο μένος (Μέρος Α) της παρούσας διπλωματικής). Η επιλογή του Apache Tomcat έγινε καθώς το RapidMiner είναι γραμμένο σε JAVA και μας παρέχει όλες τις βιβλιοθήκες που είναι απαραίτητες για την εκτέλεση μοντέλων και classifiers μέσω του Apache Tomcat.

3.3.2 MySQL Database

Για την αποθήκευση όλων των απαραίτητων δεδομένων χρησιμοποιούμε μία βάση δεδομένων MySQL (5.0.91) η οποία βρίσκεται στον ίδιο διακομιστή και την επεξεργαζόμαστε μέσω του MySQL WorkBench. Η βάση δεδομένων περιέχει τις βασικές πληροφορίες των χρηστών και των συσκευών και όποια πληροφορία είναι απαραίτητη για την ανταλλαγή των μηνυμάτων. Ακόμα, έχουμε δημιουργήσει τους πίνακες συσχέτισης ανάμεσα στους χρήστες, τις κινητές συσκευές και τα μηνύματα δημιουργώντας στο σύνολο οχτώ (8) πινάκων. Η δομή της βάσης δεδομένων που δημιουργήσαμε παρουσιάζεται στο σχήμα 9. Οι πίνακες που την απαρτίζουν είναι οι ακόλουθοι:

- user: Περιέχει τις βασικές πληροφορίες του χρήστη όπως πρωτεύον κλειδί (A/A) στοιχεία επικοινωνίας (Όνομα, Επώνυμο, e-mail) , βασικές πληροφορίες (username, password σε μορφή hash) και τέλος πληροφορίες σχετικές με την εφαρμογή μας όπως συνδεδεμένους χρήστες (χρήστες που υπάρχουν στις επαφές και έχουν εγκατεστημένη την εφαρμογή Thesis στα κινητά τους) και τους αριθμούς των τηλεφώνων που βρίσκονται στις λίστες επαφών τους.

- userdevice: Περιέχει τις βασικές πληροφορίες της συσκευής του χρήστη όπως πρωτεύον κλειδί (A/A), συσχέτιση με το χρήστη στον οποίο ανήκει η συσκευή, το GCM token με το οποίο η συσκευή έχει εγγραφεί στον GCM server και ένα αναγνωριστικό της συσκευής. Η βάση δεδομένων υποστηρίζει πολλαπλές συσκευές εγγεγραμμένες στον ίδιο χρήστη. Η διασύνδεση εντός της βάσης γίνεται με βάση το πρωτεύων κλειδί (id) του χρήστη, όμως σαν επιχειρησιακή λογική η διασύνδεση και η καταγραφή νέου χρήστη γίνεται με βάση τον τηλεφωνικό του αριθμό.

- message: Περιέχει όλες εκείνες τις πληροφορίες που είναι απαραίτητες για κάθε μήνυμα που στέλνεται μέσω της εφαρμογής Thesis και αυτές είναι, πρωτεύον κλειδί (A/A) αποστολέας (συσχέτιση με τον πίνακα των χρηστών), χρονοσφραγιδα αποστολής. Περιέχει τέλος το κείμενο του μηνύματος και το αναγνωριστικό της συνομιλίας (thread).

- usermessage: Είναι ένας πίνακας συσχέτισης που περιλαμβάνει δύο δευτερεύοντα κλειδιά από τον πίνακα των users και των messages αντίστοιχα και ουσιαστικά χρησιμοποιείται για να διασυνδέσει τους χρήστες με τα μηνύματα που έχουν αποσταλεί. Βάση αυτού του πίνακα καταγράφεται αν ο χρήστης προορίζεται να λάβει αυτό το μήνυμα σε κάποια δεδομένη συνομιλία. Ο πίνακας αυτός περιέχει επίσης πληροφορίες όπως αν ο παραλήπτης έχει λάβει ή/και διαβάσει το μήνυμα, καθώς και τις αντίστοιχες χρονοσφραγίδες παραλαβής και ανάγνωσης. Ο πίνακας αυτός σε συνδυασμό με τον userconversation επιτρέπει στο σχήμα την υποστήριξη ομαδικής συνομιλίας.

- messagestatistics: Είναι ένας πίνακας συσχέτισης που περιλαμβάνει ένα πρωτεύον κλειδί (A/A) και ένα δευτερεύον κλειδί από τον πίνακα των messages. Στον πίνακα αυτό αποθηκεύονται όλα τα meta-δεδομένα ενός μηνύματος που εξάγονται από τους διαθέσιμους αισθητήρες της κινητής συσκευής. Τα δεδομένα από τους αισθητήρες είναι ακατέργαστα και αποθηκεύονται σε μορφή JSON. Ο κύριος λόγος μη προ επεξεργασίας

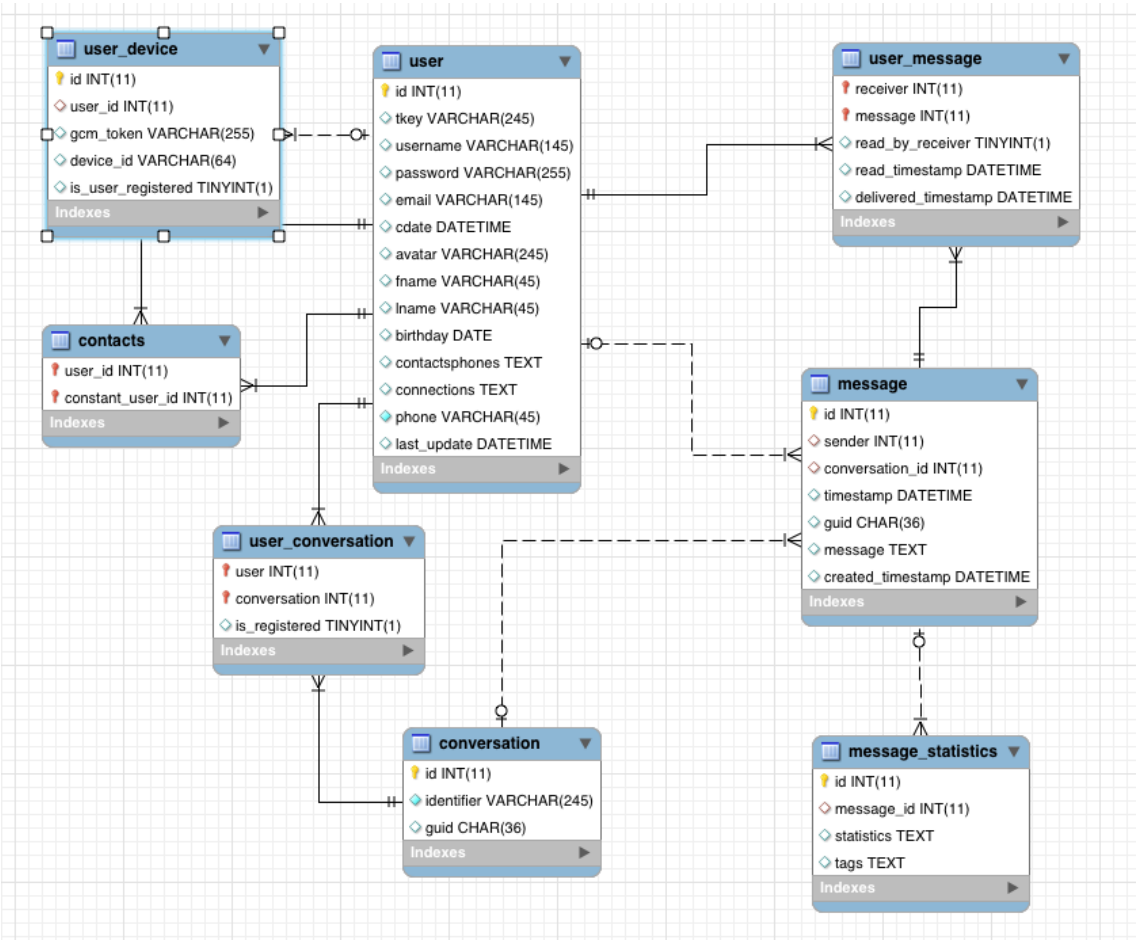
των δεδομένων αυτών είναι οι διαφοροποιήσεις που παρουσιάζουν από συσκευή σε συσκευή.

- contacts: Είναι ένας πίνακας συσχέτισης που περιλαμβάνει δύο δευτερεύοντα κλειδιά από τον πίνακα των users και ουσιαστικά χρησιμοποιείται για να διασυνδέσει τους χρήστες μεταξύ τους σύμφωνα με τις λίστες επαφών.

- conversation: Είναι ένας πίνακας συσχέτισης που χρησιμοποιείται για να διασυνδέσει τα μηνύματα με τα διάφορα conversations (message threads).

- userconversation: Είναι ένας πίνακας συσχέτισης που περιλαμβάνει δύο δευτερεύοντα κλειδιά από τον πίνακα των users και των conversations αντίστοιχα και ουσιαστικά χρησιμοποιείται για να διασυνδέσει τους χρήστες με τα διάφορα conversations (message threads). Σε αυτόν καταγράφεται αν κάποιος χρήστης είναι εγγεγραμμένος και ενεργός σε δεδομένη συνομιλία.

Αξίζει να σημειωθεί σε αυτό το σημείο αυτό ότι πρόσβαση στη βάση δεδομένων μπορεί να έχει μόνο ο εξυπηρετητής. Οποιαδήποτε καταχώρηση ή προβολή (UPDATE ή SELECT) είναι δυνατή μόνο μέσω του εξυπηρετητή και στη συνέχεια τα αποτελέσματα επιστρέφονται στους αιτούντες των λειτουργιών αυτών τους τρόπους που αναφέρουμε παραπάνω. Η βάση δεδομένων προστατεύεται από ανάχωμα προστασίας (firewall), που επιτρέπει συνδέσεις μόνο από την IP του εξυπηρετητή. Προκειμένου ο εξυπηρετητής να συνδεθεί στη βάση χρησιμοποιεί προ-εγκαταστημένο username και password.


user_device
id INT(11)
user_id INT(11)
gcm_token VARCHAR(255)
device_id VARCHAR(64)
is_user_registered TINYINT(1)
Indexes
user
id INT(11)
tkey VARCHAR(245)
username VARCHAR(145)
password VARCHAR(255)
email VARCHAR(145)
cdate DATETIME
avatar VARCHAR(245)
fname VARCHAR(45)
lname VARCHAR(45)
birthday DATE
contactsphones TEXT
connections TEXT
phone VARCHAR(45)
last_update DATETIME
Indexes
user_message
receiver INT(11)
message INT(11)
read_by_receiver TINYINT(1)
read_timestamp DATETIME
delivered_timestamp DATETIME
Indexes
contacts
user_id INT(11)
constant_user_id INT(11)
Indexes
message
id INT(11)
sender INT(11)
conversation_id INT(11)
timestamp DATETIME
guid CHAR(36)
message TEXT
created_timestamp DATETIME
Indexes
user_conversation
user INT(11)
conversation INT(11)
is_registered TINYINT(1)
Indexes
conversation
id INT(11)
identifier VARCHAR(245)
guid CHAR(36)
Indexes
message_statistics
id INT(11)
message_id INT(11)
statistics TEXT
tags TEXT
Indexes

3.4 Εφαρμογή Χρήστη

Η εφαρμογή του πελάτη που προορίζεται για τους χρήστες είναι γραμμένη σε JAVA για λειτουργικό Android. Το περιβάλλον που έχουμε δημιουργήσει είναι απλό, λειτουργικό και ελκυστικό για το χρήστη ακολουθώντας τις βέλτιστες πρακτικές και τις οδηγίες του material design. Όπως περιγράφεται στο σχήμα 16, η κεντρική σελίδα της εφαρμογής του χρήστη αποτελείται από δύο καρτέλες κύριων λειτουργιών. Αυτές αντιστοιχούν στις δύο βασικές λειτουργίες της εφαρμογής οι οποίες είναι οι εξής:

- Προβολή των επαφών και φιλτράρισμα αυτών που έχουν επίσης εγκατεστημένη την εφαρμογή στις κινητές τους συσκευές.

- Προβολή των συνομιλιών μεταξύ των χρηστών της εφαρμογής.

Εκτός από τις δύο βασικές καρτέλες η εφαρμογή διαθέτει και άλλες τρεις (3) οθόνες οι οποίες είναι οι εξής:

- Splash Screen : Πρόκειται για την αρχική οθόνη της εφαρμογής.

- Register Screen : Πρόκειται για την οθόνη εγγραφής του χρήστη αν αυτός δεν υπάρχει ήδη στο σύστημα.

- Message Screen : Πρόκειται για την οθόνη ανταλλαγής μηνυμάτων μεταξύ των χρηστών. Περιέχει το βασικό UI της ανταλλαγής μηνυμάτων τόσο μεταξύ δύο (2) όσο και περισσότερων χρηστών.

3.4.1 Splash Screen

Η πρώτη οθόνη την οποία αντικρίζει ο χρήστης όταν ανοίγει την εφαρμογή είναι η εικόνα που φαίνεται παρακάτω. Κατά τη διάρκεια που ο χρήστης παραμένει σε αυτή την οθόνη η εφαρμογή εκκινεί το πρωτόκολλο επικοινωνίας με τον εξυπηρετητή και αρχικοποιεί τις βασικές μεταβλητές και μεθόδους. Πιο συγκεκριμένα σε αυτό το στάδιο η εφαρμογή επικοινωνεί με τον εξυπηρετητή αποστέλλοντας πληροφορίες σχετικά με τη συσκευή και τον ίδιο το χρήστη (όπως π.χ. τα αναγνωριστικά της συσκευής, τα δεδομένα των επαφών του χρήστη, το τηλέφωνο της GSM σύνδεσης κ.α.). Ο εξυπηρετητής λαμβάνοντας τα δεδομένα μπορεί να απαντήσει με δύο διαφορετικούς τρόπους οι οποίοι είναι οι εξής:

- Αυθεντικοποίηση του χρήστη και επιστροφή των Επαφών (Contacts), Συνομιλιών (Conversations) και των μηνυμάτων (Messages) για τον εν λόγω χρήστη.

- Αδυναμία αυθεντικοποίησης του χρήστη και προτροπή στην εφαρμογή να προχωρήσει στην εγγραφή του χρήστη στο σύστημα.

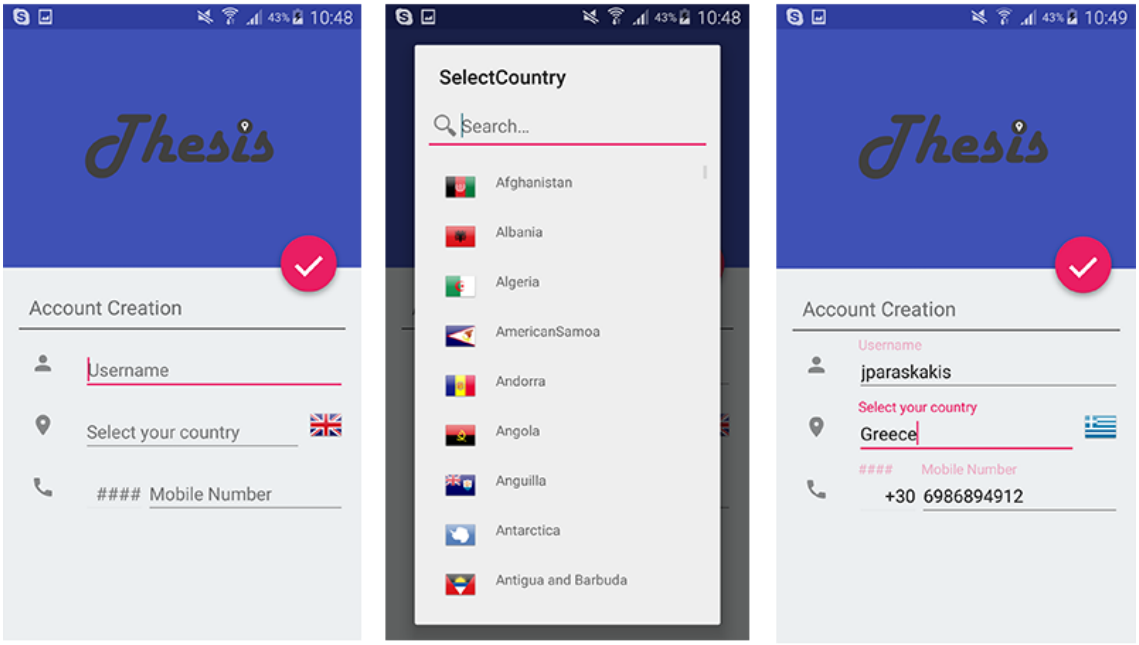
Splash Screen

```
Boolean userRegisteredSuccessfully =
Endpoints.registerUser(this, Globals.getCurrentDevice(),
phoneVerificationTxt.getText().toString());

if (userRegisteredSuccessfully) {
    progressCircle.beginFinalAnimation();
    Intent in = new Intent(this.getApplicationContext(),
MainActivity.class);
    in.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    this.getApplicationContext().startActivity(in);
    this.finish();
} else {
    progressCircle.hide();
    phoneVerificationTxt.setEnabled(true);
    registrationVerificationBtn.setEnabled(true);
}
```

Αναγνώριση του χρήστη και επιλογή Activity

3.4.2 Registration

Κατά τη πρώτη εκκίνηση της εφαρμογής (ο εξυπηρετητής δε μπορεί να αυθεντικοποιήσει το χρήστη στις υπάρχουσες εγγραφές στη βάση δεδομένων) ο χρήστης καλείται να εγγραφεί συμπληρώνοντας το όνομα χρήστη του, το τηλέφωνο του και επιλέγοντας τη χώρα του από μία λίστα με διαθέσιμες χώρες και τηλεφωνικούς κωδικούς. Όταν ολοκληρώσει την εισαγωγή των στοιχείων του ο χρήστης η εφαρμογή επικοινωνεί με τον εξυπηρετητή προκειμένου ο δεύτερος να αποθηκεύσει τα δεδομένα στη βάση και να δημιουργήσει τις απαραίτητες εγγραφές και συσχετίσεις. Όταν ολοκληρωθεί αυτή η διαδικασία τότε ο χρήστης ανακατευθύνεται στην κεντρική οθόνη της εφαρμογής.



Registration Screen

```
String username =
activity._UserNameTxt.getText().toString();
String countrycode =
activity._CountryCode.getText().toString();
String phone = activity._PhoneTxt.getText().toString();

TelephonyManager telephontManager = (TelephonyManager)
activity.getApplication().getSystemService(Context.TELEPHON
Y_SERVICE);

String deviceCountryCode = Environment.isEmulatorDevice() ?
"GR" : telephontManager.getSimCountryIso();

String phoneNumber = countrycode + phone;
if(Globals.getCurrentDevice().get_User() == null)
    Globals.getCurrentDevice().set_User(new User());
Globals.getCurrentDevice().get_User().set_Username(username
);
Globals.getCurrentDevice().get_User().set_Phone(Helpers.nor
malizeNumber
(phoneNumber, deviceCountryCode));

boolean request = Endpoints.requestCreateUser(activity,
Globals.getCurrentDevice());
```

Registration process

3.4.3 Κεντρική Οθόνη - Επαφές/Συνομιλίες

Η κεντρική οθόνη, στην οποία ανακατευθύνεται ο χρήστης είτε μετά από επιτυχημένη αυθεντικοποίηση από τον εξυπηρετητή είτε ύστερα από επιτυχημένη εγγραφή στο σύστημα, περιέχει δύο καρτέλες (Sliding Tabs). Η πρώτη καρτέλα εμφανίζει τις επαφές του χρήστη ενώ η δεύτερη εμφανίζει όλες τις συνομιλίες με τους άλλους χρήστες της εφαρμογής. Ο χρήστης έχει τη δυνατότητα να επιλέξει διαφορετικές προβολές της λίστας των επαφών ανάμεσα στις παρακάτω προβολές:

- Προβολή όλων των επαφών (Από όλες τις λίστες επαφών π.χ. τηλεφώνου, SIM, social).
- Προβολή μόνο των επαφών που έχουν εγκαταστημένη την εφαρμογή στις κινητές τους συσκευές.

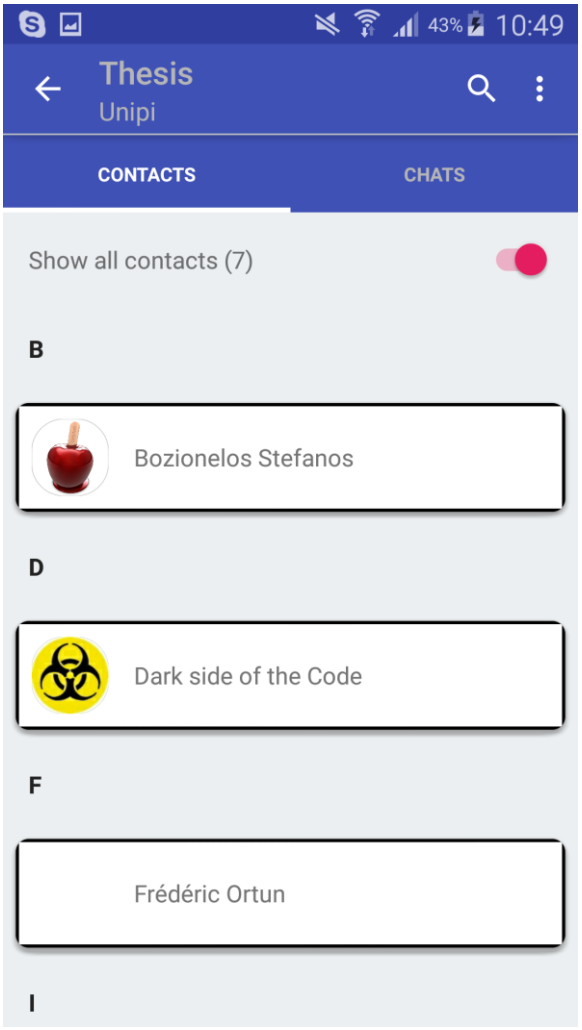
Η δεύτερη καρτέλα περιέχει όλες τις συνομιλίες στις οποίες συμμετέχει ο χρήστης της εφαρμογής. Ειδικά notifications ενημερώνουν το χρήστη για αδιάβαστα μηνύματα και για τους συμμετέχοντες της εκάστοτε συνομιλίας. Τα δεδομένα που συμπληρώνουν τη λίστα με τις συνομιλίες λαμβάνονται ύστερα από επικοινωνία της εφαρμογής με τον εξυπηρετητή.

```
recyclerView = (RecyclerView)
view1.findViewById(R.id.contact_recycler_view);
SwitchCompat showOnlyThesisContancts = (SwitchCompat)
view1.findViewById(R.id.thesis_only_switch);

final TextView thesisAllContactsTxt = (TextView)
view1.findViewById(R.id.thesis_all_contacts_text);
 container.addView(view1);

List<UserContact> userContacts = new
ArrayList<UserContact>(Globals.getAllContacts());
thesisAllContactsTxt.setText(getString(R.string.T_SHOW_ALL_CONTAC
TS) + " (" + userContacts.size() + ")");

final RecyclerViewContactsAdapter recyclerViewContactsAdapter =
new RecyclerViewContactsAdapter(getActivity(), userContacts);

recyclerView.setAdapter(recyclerViewContactsAdapter);
recyclerView.setClickable(true);
LinearLayoutManager mLinearLayoutManager = new
LinearLayoutManager(getActivity());
mLinearLayoutManager.setOrientation(LinearLayoutManager.VERTICAL)
;
recyclerView.setLayoutManager(mLinearLayoutManager);
```

Populate the Contacts Recycler view

```
view2 =
getActivity().getLayoutInflater().inflate(R.layout.conversation_r
ecycler_view, container, false);

conversationRecyclerView = (RecyclerView)
view2.findViewById(R.id.conversation_recycler_view);
container.addView(view2);

List<Conversation> userConversations = new
ArrayList<Conversation>(Globals.getAllConversations());

final RecyclerViewConversationsAdapter
recyclerViewConversationsAdapter = new
RecyclerViewConversationsAdapter(getActivity(),
userConversations);

conversationRecyclerView.setAdapter(recyclerViewConversationsAdap
ter);
conversationRecyclerView.setClickable(true);
LinearLayoutManager mLinearLayoutManager = new
LinearLayoutManager(getActivity());
mLinearLayoutManager.setOrientation(LinearLayoutManager.VERTICAL)
;
conversationRecyclerView.setLayoutManager(mLinearLayoutManager);
```

Populate Conversations Recycler View



Contacts Screen

3.4.4 Οθόνη ανταλλαγής μηνυμάτων

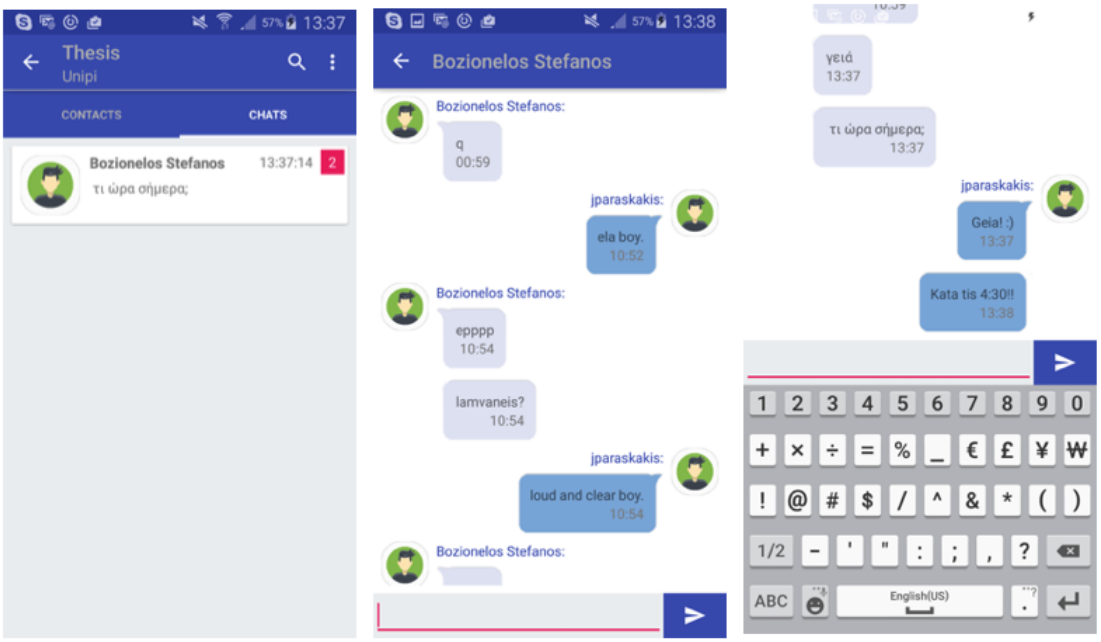
Αφού ένας χρήστης επιλέξει τη συνομιλία που επιθυμεί από την καρτέλα προβολής των συνομιλιών ανακατευθύνεται στη σελίδα ανταλλαγής άμεσων μηνυμάτων. Στην οθόνη αυτή μπορεί να δει το όνομα (αναγνωριστικό) της συνομιλίας καθώς και τα μηνύματα που έχουν αποσταλεί από όλους τους συμμετέχοντες. Όταν ο χρήστης συντάξει το μήνυμα που θέλει να στείλει τότε η εφαρμογή επικοινωνεί με τον εξυπηρετητή και του μεταφέρει το μήνυμα και τους παραλήπτες του μηνύματος. Ο εξυπηρετητής με τη σειρά του ενημερώνει το GCM Server και αυτός ενημερώνει τις συσκευές των παραληπτών για το νέο μήνυμα που έχουν λάβει.

```
String messageText = _SendText.getText().toString();
Message newmessage = new Message(messageText);
Globals.getSelectedConversation().get_Messages().add(newmessag
e);
Endpoints.sendMessage(newmessage,
Globals.getSelectedConversation());
((RecyclerViewMessagesAdapter)messageRecyclerView.getAdapter()
).
updateMessages(Globals.sortMessagesByDate(
Globals.getSelectedConversation().get_Messages()));

messageRecyclerView.smoothScrollToPosition(
Globals.getSelectedConversation().get_Messages().size());
```

Αποστολή μηνύματος

```
public void onMessageReceived(String from, Bundle data) {
    String type = data.getString("type");
    System.out.println(type);
    String message = data.getString("message");
    Integer messageID =
Integer.parseInt(data.getString("messageid"));
    Integer senderID =
Integer.parseInt(data.getString("senderid"));
    Integer conversationID =
Integer.parseInt(data.getString("conversationid"));
    String conversationIdentifier =
data.getString("conversationidentifier");
}
```

Παραλαβή μηνύματος μέσω GCM


13:37
Thesis
Unipi
CONTACTS
CHATS
Bozionelos Stefanos
13:37:14
2
τι ώρα σήμερα;
13:38
Bozionelos Stefanos
Bozionelos Stefanos:
q
00:59
jparaskakis:
ela boy.
10:52
Bozionelos Stefanos:
epppp
10:54
Iamvaneis?
10:54
jparaskakis:
loud and clear boy.
10:54
Bozionelos Stefanos:
γειά
13:37
τι ώρα σήμερα;
13:37
jparaskakis:
Geia! :)
13:37
Kata tis 4:30!!
13:38
ABC
English(US)

Chat Screen

3.5 Εφαρμογή Εξυπηρετητή - API

Όπως αναφέρουμε στις ενότητες 2.1.1 και 2.1.2 η εφαρμογή του εξυπηρετητή έχει υλοποιηθεί με τη βοήθεια του Symfony Framework και ουσιαστικά αποτελεί ένα σύνολο από web services (endpoints). Οι εφαρμογές των χρηστών μπορούν να καλέσουν αυτά τα web services μέσω του HTTP πρωτοκόλλου και τα τελευταία με τη σειρά τους εκτελούν όλη το business logic και επιστρέφουν τις απαραίτητες πληροφορίες στις κινητές συσκευές. Τα web services που είναι διαθέσιμα μέσω του API είναι τα εξής:

- api/sendMessage: Περιέχει όλη τη λειτουργικότητα που απαιτείται για την αποστολή, αποθήκευση αλλά και προώθηση των μηνυμάτων των χρηστών τόσο στη μεριά του εξυπηρετητή όσο και στην επικοινωνία με τον GCM Server. Σε αυτό το σημείο καταγράφονται και όλα τα δεδομένα από τους αισθητήρες της συσκευής. Η καταγραφή των δεδομένων αυτών, αν και επιτακτική για την δημιουργία του μοντέλου μας δεν θεωρείται αναγκαία για την εφαρμογή. Έτσι η αποτυχία καταγραφής των δεδομένων των αισθητήρων και οδηγεί σε αποτυχία εκτέλεσης της αποστολής/καταγραφής του μηνύματος. Η συνάρτηση αυτή με την ολοκλήρωση της επιστρέφει όλα τα στοιχεία του μηνύματος σε μορφή XML, καθώς και τα απαραίτητα στοιχεία της συνομιλίας στην οποία ανήκει.

```
$em = $this->getDoctrine()->getManager();
 $deviceRepository = $em->getRepository('AppBundle:UserDevice');
 $userRepository = $em->getRepository('AppBundle:User');

 $senderUser = $userRepository->find($sender);
 $participantsIds = array_unique(array_merge($receiversArray, array($sender)));
 $conversation = $this->getConversation($conversation_id, $conversation_name,
$conversation_guid, $participantsIds);

 $headers = array( 'Authorization: key=' . $this->getSApiKey(), 'Content-Type:
application/json' );

  $ch = curl_init();
  curl_setopt($ch, CURLOPT_URL, $this->getSApiEndpoint());
  curl_setopt($ch, CURLOPT_POST, true);
  curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
  curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($post));

  $result = curl_exec($ch);
```

Αποστολή μηνυμάτων μέσω του API

- /api/getConversation: Περιέχει όλη τη λειτουργικότητα που απαιτείται για την ανάκτηση και τη μεταφορά των συνομιλιών και των μηνυμάτων στις εφαρμογές των χρηστών. Η συνάρτηση αυτή χρησιμοποιείται για την ανάκτηση μία μόνο συνομιλίας, με σκοπό την ενημέρωση της αντίστοιχης συνομιλίας στη συσκευή του χρήστη. Επιστρέφονται όλα τα χαρακτηριστικά μία συνομιλίας, όπως το αναγνωριστικό της και οι συμμετέχοντες, καθώς επίσης και τα 30 τελευταία μηνύματα που έχουν καταγραφεί πριν από μία δεδομένη χρονική στιγμή. Για το λόγο αυτό η συνάρτηση αυτή χρησιμοποιείται και για την ανάκτηση παλαιότερων μηνυμάτων, καθώς ανά συνομιλία ο χρήστης μπορεί να δει αρχικά μόνο τα 30 πιο πρόσφατα μηνύματα.

```
$conversationid = $request->get('conversationid', null);
$userid = $request->get('userid', null);
$firstMessageDateTime = $request->get('firstmessagedatetime', null);
$firstMessageDateTime = (new \DateTime($firstMessageDateTime))-
>setTimezone(new \DateTimeZone('UTC'));

$em = $this->getDoctrine()->getManager();
$messageRepository = $em->getRepository('AppBundle:Message');
$conversationRepositoty = $em-
>getRepository('AppBundle:Conversation');
$userRepository = $em->getRepository('AppBundle:User');

$user = $userRepository->find($userid);

$conversation = $conversationRepositoty->find($conversationid);

$latestMessages = $messageRepository-
>findLatestUserMessagesInConversation($userid,
$conversation->getId(), $firstMessageDateTime);
$unreceivedMessageIds = $this-
>updateConversationReceived($latestMessages, $user);
$conversation->setLatestMessages($latestMessages);
$gcmTokens = array();
foreach ($conversation->getParticipants() as $participant) {
    if($participant->getUserId() != $userid) {
        $gcmTokens = array_merge($this->getUserGcmTokens(
        $participant->getUserId()), $gcmTokens);
    }
}
$this->notifyMessagesReceived($gcmTokens, $unreceivedMessageIds,
$userid, $conversation->getId());
```

Ανάκτηση συνομιλιών

- /api/createDevice: Η λειτουργικότητα αυτού του web service είναι διπλή. Από τη μία αυθεντικοποιεί το χρήστη και επιστρέφει ένα XML με όλες τις απαραίτητες πληροφορίες για τη σωστή λειτουργία της εφαρμογής ενώ από την άλλη αν η συσκευή δεν υπάρχει στις εγγραφές της βάσης δεδομένων τότε το εν λόγω web service αναλαμβάνει να δημιουργήσει την εγγραφή και να ενημερώσει την εφαρμογή να προχωρήσει στην εγγραφή του χρήστη.

```
//Get current device if persisted
$deviceRepository = $em->getRepository('AppBundle:UserDevice');
$messageRepository = $em->getRepository('AppBundle:Message');
$device = $deviceRepository->findOneBy(array(
    'deviceId' => $deviceId
));

if (isset($device)) {
    //if User exists updated his contacts
    if (isset($gcmToken) && strcmp($device->getGcmToken(),
$gcmToken) != 0) {
        $device->setGcmToken($gcmToken);
    }
    $em->persist($device);
    $em->flush();
    return $this->xmlResponse($device,
    SerializationContext::create()->setGroups(array('Device')));
}
//Create new UserDevice
$user_device = new UserDevice();
$user_device->setGcmToken($gcmToken);
$user_device->setDeviceId($deviceId);
$em->persist($user_device);
$em->flush();

return $this-
>xmlResponse($user_device,SerializationContext::create()->
enableMaxDepthChecks()->setGroups(array('Device')));
```

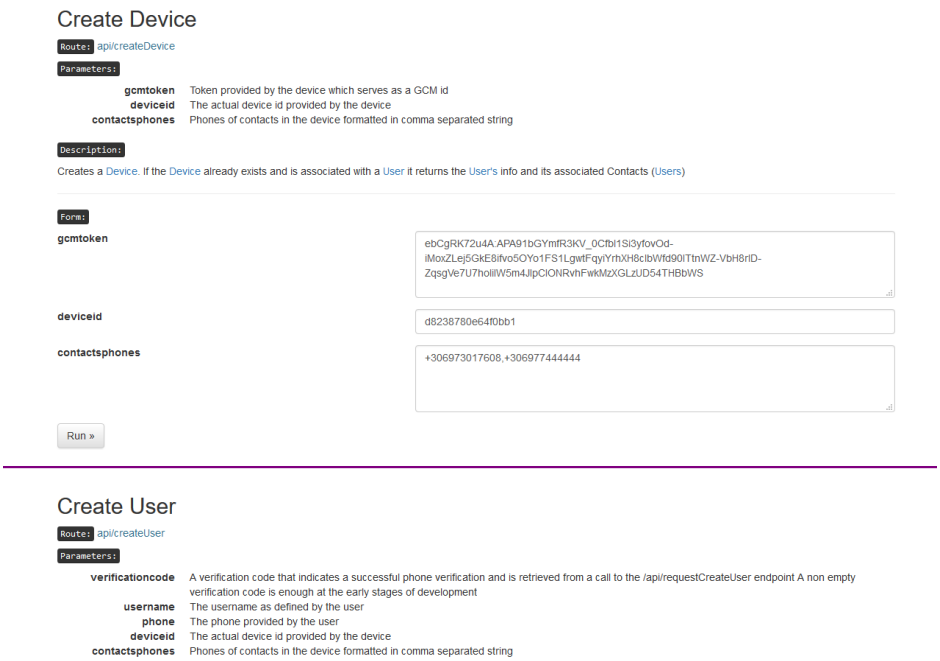
Εγγραφή συσκευής

- /api/createUser: Σε συνέχεια του web service «/api/createDevice» το «api/createUser» αναλαμβάνει να καταχωρήσει τις πληροφορίες που αφορούν το χρήστη της εφαρμογής όπως όνομα χρήστη, αριθμός τηλεφώνου κ.α.. Με τις δύο αυτές κλήσεις της εγγραφής της συσκευής και της εγγραφής του χρήστη ολοκληρώνεται η διαδικασία της εγγραφής στη πλατφόρμα Thesis.

```
$userRepository = $em->getRepository('AppBundle:User');
$user = $userRepository->findOneBy(array(
    'phone' => $phone
));
if (!isset($user)) {
    $user = new User();
    $user->setCdate((new DateTime())->setTimezone(new
\DateTimeZone('UTC')));
    $user->setTKey(sha1(date_format(new DateTime(), 'Y-m-d H:i:s') .
$phone));
    $user->setAvatar(self::DEFAULT_AVATAR_URL);
}

$user->setPhone($phone);
$user->setUsername($username);


$contactsPhonesArray = explode(',', $contactsPhones);
$updatedContactsPhones =
json_encode(array_values($contactsPhonesArray));
$user->setContactsPhones($updatedContactsPhones);
$user->setLastUpdate((new DateTime())->setTimezone(new
\DateTimeZone('UTC')));

$em->persist($user);
$em->flush();

$device->setUser($user);
$device->setIsUserRegistered(true);

$em->persist($device);
$em->flush();

return $this->xmlResponse($device,SerializationContext::create()
->enableMaxDepthChecks()->setGroups(array('Device')));
```

Εγγραφή χρήστη

Εκτός από τα παραπάνω web services έχουμε δημιουργήσει και μία σελίδα δοκιμών για το API προκειμένου να παρέχουμε ένα documentation που περιέχει τις παραμέτρους που απαιτεί κάθε web service για την ορθή λειτουργία του αλλά και ένα μέρος εκτέλεσης των τελευταίων ώστε να μπορούμε εύκολα να ελέγχουμε τις απαντήσεις στα διάφορα αιτήματα. Η σελίδα αυτή δεν αποτελεί σημείο ελέγχου της σωστής λειτουργίας όλων των συστημάτων, μπορεί όμως να παρέχει μία ολοκληρωμένη εικόνα του API στον αναγνώστη.



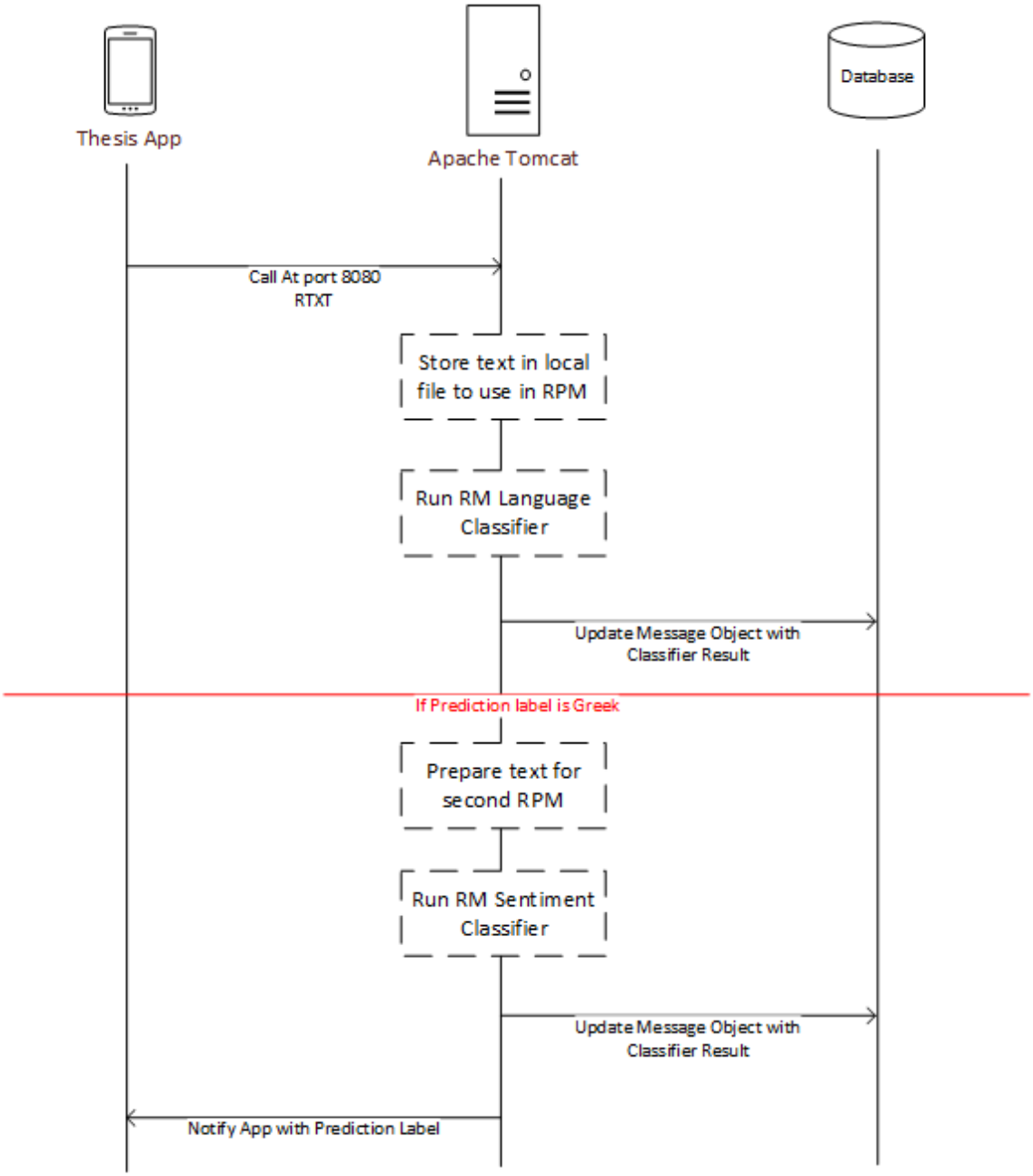
Περιβάλλον δοκιμής Web Services

3.6 Πρωτόκολλο επικοινωνίας

Στην παρούσα ενότητα αναλύεται το πρωτόκολλο με το οποίο επικοινωνεί ο εξυπηρετητής με τους πελάτες.

3.6.1 Επικοινωνία Χρήστη - Εξυπηρετητή

Στο σημείο αυτό θα αναλύσουμε το πρωτόκολλο επικοινωνίας ανάμεσα στο χρήστη και τον κεντρικό εξυπηρετητή το οποίο παρουσιάζεται στο σχήμα 25. Η σύνδεση ξεκινάει αμέσως όταν ο χρήστης ανοίξει την εφαρμογή και δει την αρχική οθόνη (Splash Screen). Τη στιγμή αυτή δημιουργείται μία σύνδεση με τον εξυπηρετητή στη θύρα 80 μέσω του πρωτοκόλλου HTTP και πραγματοποιείται μία αρχική κλήση στο web service api/createDevice. Η κλήση αυτή έχει ως στόχο το συγχρονισμό της κινητής συσκευής με τον εξυπηρετητή. Για το λόγο αυτό ο χρήστης στέλνει στον εξυπηρετητή το GCM Token που έχει ανακτήσει από τον Google Server, το αναγνωριστικό της συσκευής (μοναδικό για κάθε συσκευή) και μία λίστα με τις επαφές του. Αφού ο Server έχει λάβει τις αρχικές πληροφορίες του χρήστη, προσπαθεί να τον αυθεντικοποιήσει. Στη περίπτωση που ο εξυπηρετητής καταφέρει να ανακτήσει από τη βάση δεδομένων τον χρήστη, μέσω του αναγνωριστικού της συσκευής του, τότε ενημερώνει τις επαφές του εν λόγω χρήστη και το GCM Token του στη βάση και αποκρίνεται με τις συνομιλίες και τα μηνύματα τους σε μορφή XML Document. Στη περίπτωση που ο εξυπηρετητής αδυνατεί να ανακτήσει την εγγραφή του χρήστη από τη βάση δεδομένων τότε δημιουργεί την εγγραφή της συσκευής και αναμένει να λάβει τις πληροφορίες του χρήστη ώστε να ολοκληρώσει τη διαδικασία της εγγραφής. Στη συνέχεια ανάλογα με την απάντηση που θα λάβει η εφαρμογή του χρήστη από τον εξυπηρετητή είτε ανακατευθύνει το χρήστη στη κεντρική οθόνη των επαφών/συνομιλιών είτε στην οθόνη της εγγραφής του χρήστη. Στη περίπτωση που ο χρήστης δεν έχει ολοκληρώσει τη διαδικασία εγγραφής τότε συμπληρώνει τα πεδία της οθόνης εγγραφής και η εφαρμογή καλεί το web service /api/createUser αποστέλλοντας μαζί τις πληροφορίες του χρήστη. Όταν λάβει ο εξυπηρετητής τις πληροφορίες του χρήστη τότε δημιουργεί την εγγραφή του χρήστη στη βάση δεδομένων και τη διασύνδεση με την εγγραφή της συσκευής που έχει πραγματοποιήσει στο προηγούμενο βήμα. Με αυτό τον τρόπο ολοκληρώνεται η διαδικασία εγγραφής και συγχρονισμού των χρηστών με τον κεντρικό εξυπηρετητή. Έχοντας ανακατευθυνθεί ο χρήστης στην οθόνη επαφών/συνομιλιών μπορεί είτε να δει υπάρχουσες συνομιλίες είτε να εκκινήσει μία νέα. Στη περίπτωση που επιλέξει μία υπάρχουσα συνομιλία τότε η εφαρμογή καλεί το /api/getConversation με το αναγνωριστικό της συνομιλίας ως παράμετρο. Ο εξυπηρετητής με τη σειρά του ανακτά της συνομιλία από τη βάση και αποστέλλει τη συνομιλία μαζί με τα μηνύματα σε μορφή XML πίσω στην εφαρμογή, ταυτόχρονα ενημερώνει τη βάση για τη σήμανση των μηνυμάτων ως αναγνωσμένα. Τέλος όταν ο χρήστης επιλέξει να στείλει ένα μήνυμα σε μία συνομιλία τότε η εφαρμογή καλεί το /api/sendMessage αποστέλλοντας ταυτόχρονα το αναγνωριστικό του χρήστη που στέλνει το μήνυμα, τα αναγνωριστικά των παραληπτών, το αναγνωριστικό της συνομιλίας και το μήνυμα σε plaintext. Όταν ο εξυπηρετητής λάβει το μήνυμα του χρήστη δημιουργεί τις κατάλληλες εγγραφές στους πίνακες των μηνυμάτων και των συνομιλιών αλλά και τις μεταξύ τους συσχετίσεις. Στη συνέχεια αναζητά τα GCM Tokens των παραληπτών και αποστέλλει στον GCM server τα GCM Tokens των συσκευών και το μήνυμα που πρόκειται να αποσταλεί.


Thesis App
Apache Tomcat
Database
Call At port 8080
RTXT
Store text in local file to use in RPM
Run RM Language Classifier
Update Message Object with Classifier Result
If Prediction label is Greek
Prepare text for second RPM
Run RM Sentiment Classifier
Update Message Object with Classifier Result
Notify App with Prediction Label

Αποτελέσματα και συμπεράσματα

4.1 Αποτελέσματα

Το αποτέλεσμα της παρούσας διπλωματικής εργασίας είναι η δημιουργία μίας πλήρως λειτουργικής πλατφόρμας που όχι μόνο επιτρέπει την ανταλλαγή μηνυμάτων με χρήση cloud τεχνολογιών αλλά ταυτόχρονα αναγνωρίζει τα συναισθήματα που περιέχονται στα εν λόγω μηνύματα. Επιπροσθέτως δημιουργούμε μία συσχέτιση ανάμεσα σε μηνύματα και metadata των μηνυμάτων αυτών που προέρχονται από τους διάφορους αισθητήρες των κινητών συσκευών. Έτσι λοιπόν θεωρούμε ότι καταφέραμε να πραγματοποιήσουμε το στόχο που θέσαμε στην αρχή αυτής της διπλωματικής εργασίας.

4.2 Συμπεράσματα

Από την εκπόνηση της παρούσας διπλωματικής συμπεραίνουμε την ανάγκη αλλά ταυτόχρονα και την ευκαιρία, στο πλαίσιο του sentiment analysis (opinion mining) για δημιουργία εξειδικευμένων εφαρμογών που θα υποστηρίζουν την ερευνητική διαδικασία με δεδομένα που μπορεί να μας δώσουν χρήσιμες πληροφορίες για τα συναισθήματα των χρηστών. Ακόμα, η αξιοποίηση αυτών των meta-δεδομένων μπορεί να αποδειχθεί πολύ σημαντική στην έρευνα γύρω από το τομέα του Natural Language Processing NLP καθώς η ανάλυση τους μας δίνει μία πιο ολοκληρωμένη εικόνα γύρω από την ανθρώπινη επικοινωνία και την έκφραση της μέσω μηνυμάτων κειμένου. Επίσης, θεωρούμε ότι ιδιαίτερη προσοχή θα πρέπει να δίνεται στην προστασία των χρηστών, η ανάπτυξη εφαρμογών θα πρέπει να συνδυάζεται με υλοποίηση πολιτικών ασφαλείας που θα προστατεύουν την ιδιωτικότητα και θα εξασφαλίζουν την ανωνυμία των χρηστών.

4.3 Μελλοντική Εργασία

Από τη μελέτη της σχετικής βιβλιογραφίας προκύπτει ότι η πλατφόρμα Thesis αποτελεί μία καινοτόμα και πρωτοποριακή υλοποίηση, που επιτρέπει το συνδυασμό του opinion mining με τις νέες ροές δεδομένων που μπορούμε να έχουμε με την εξάπλωση και την εξέλιξη των έξυπνων συσκευών. Μελλοντικό μας σχέδιο είναι η βελτίωση και η επέκταση της πλατφόρμας Thesis με την πραγματοποίηση των παρακάτω ενεργειών:

- Πληρέστερη συλλογή μηνυμάτων για δημιουργία καλύτερων μοντέλων αναγνώρισης συναισθημάτων που θα περιλαμβάνει περισσότερα δείγματα από περισσότερες κλάσεις.
- Προσαρμογή στην πλατφόρμα μεγαλύτερων datasets για αναγνώριση περισσότερων συναισθημάτων και από περισσότερες γλώσσες εκτός της ελληνικής.
- Μετεγκατάσταση του εξυπηρετητή σε πιο ισχυρούς υπολογιστές προκειμένου να μειωθεί ο χρόνος απόκρισης για την εκτέλεση του μοντέλου αναγνώρισης (γλώσσας και συναισθήματος).
- Ενσωμάτωση ψηφιακών πιστοποιητικών για παροχή υπηρεσιών εμπιστευτικότητας ανάμεσα στα μέρη του συστήματος.
- Επέκταση της εφαρμογής του χρήστη για άλλες συσκευές και άλλα λειτουργικά συστήματα εκτός του Android.

Βιογραφικό Σημείωμα

Ο Μποζιονέλος Στέφανος γεννήθηκε στην Αττική το 1987. Είναι απόφοιτος του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς και μεταπτυχιακός φοιτητής του τμήματος Προηγμένα Συστήματα Πληροφορικής του Πανεπιστημίου Πειραιώς. Εργάζεται ως προγραμματιστής στην Αθήνα τα τελευταία τρία χρόνια.