Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ημερολόγιο τοποθεσιών για android**
Ονοματεπώνυμο Φοιτητή	**Βασίλειος Φουρούλης**
Πατρώνυμο	**Γεώργιος**
Αριθμός Μητρώου	**ΜΠΠΛ13079**
Επιβλέπων	**Επ. Καθηγητής Αλέπης Ευθύμιος**
Βοηθός Επιβλέποντος	**Τρούσσας Χρήστος**

Ημερομηνία Παράδοσης

22/04/2016

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

Περιεχόμενα

ABSTRACT - ΠΡΟΛΟΓΟΣ 5
INTRODUCTION - ΕΙΣΑΓΩΓΗ 7
ΑΝΑΣΚΟΠΗΣΗ ΠΕΔΙΟΥ 8
ΠΑΡΟΥΣΙΑΣΗ ΚΑΙ ΧΡΗΣΗ ΕΦΑΡΜΟΓΗΣ 9
ΔΙΑΓΡΑΜΜΑΤΑ UML 10
Διαγραμμα Περιπτωσεων Χρησης (use case) 10
Διαγραμμα δραστηριοτητας (Activity Diagram) 11
FRONTEND – ANDROID APPLICATION 12
Εισαγωγη στο Android 12
Διασυνδεση με το δικτυο 13
Εισαγωγή *13*
Βιβλιοθήκη Volley *14*
Ευρεση τοποθεσιας στο Android 15
Location API *15*
Google Play Service *16*
Google Maps *17*
Services 19
Διαχειριση χρηστων 20
Ημερολογιο 22
Συρομενο Menu 23
Λίστα στοιχείων *25*
BACKEND API 29
Εισαγωγη 29
Πλεονεκτηματα Τεχνολογιας node.js 29
REST API 30
Web Service – Routing 31
Αρχιτεκτονικη των Web Services 32
Gloud Services 37
ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ 38
Εισαγωγη 38
Driver node.js και mongodb 39
ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ 40
Συμπερασματα 40

Περιορισμοι 40
Μελλοντικες Επεκτασεις 41
ΠΑΡΑΡΤΗΜΑ **42**
Εικονες 42
Grandle Build System *42*
Τροποι εξυπηρετησης αιτηματων 43
ΒΙΒΛΙΟΓΡΑΦΙΑ **44**

Abstract - Πρόλογος

In this Dissertation, we introduce an android application and it's stack (back-end and database). This app, can perform auto check-in and save the location of the user in a database. Also, it can display the location that has been saved in a Calendar and on Google Map.

The technologies that has been selected for the back-end platform are node.js and mongodb. This is a perfect much for this application because all the back – end platform uses the same programming language, JavaScript. In addition, this stack is very popular for creating the back – end API for android or in general in mobile applications and web applications, because it uses JSON format that is very easy to be parsed from the client.

In order the app could work via the internet, the platform of Okeanos has been used, which gave us the opportunity to have this back – end API accessible from the internet. Okeanos is a GRNET's cloud service, for the Greek Research and Academic Community. For this dissertation a virtual Machine of Windows Server 2012 has been used.

This thesis is divided in 2 parts, Theoretical part, where the technologies is being explained, UMLs are shown, some pieces of code are being introduced and bibliography. The second part of the application is the implementation of android code and of the back-end code.

Likewise, the theoretical part is divided in 3 parts. The part of Front end, where the application of the android is explained, the part of web services, where it gives the possibility for android application and the database to communicate and finally, the third part, the mongodb database. In each piece the architecture and the bibliography are being analyzed.

The second component of this thesis, is the piece of code, where it is being displayed the most important code. This code can be found on Github in the link **https://github.com/basilisfou/Dissertation**. Moreover, UML diagrams are used. Finally, as last part of this document, is bibliography. Bibliography is style of IEEE.

Not only this thesis awards the Master's degree of Computer Science of the University of Piraeus, but also we hope that it could be a good reference for someone who wants to use android Locations, node.js, mongodb and their combination.

Η παρούσα εργασία εκπονήθηκε για την απόκτηση του Μεταπτυχιακού διπλώματος ΠΜΣ Πληροφορική του Πανεπιστημίου Πειραιώς και αντιπροσωπεύει την Μεταπτυχιακή διατριβή του 4ου Εξαμήνου του προγράμματος σπουδών. Ευελπιστώ να αποτελέσει ένα βοήθημα για κάποιον που θέλει να κάνει τα πρώτα βήματά του στον χώρου του Android, nodejs και mongodb.

Πιο συγκεκριμένα, στην παρούσα γίνεται η παρουσίαση μίας εφαρμογής για την Πλατφόρμα του Android. Επιπλέον, γίνεται η παρουσίαση όλου του πακέτου (stack) της εφαρμογής, δηλαδή η πλευρά του Server (web services και βάση δεδομένων).

Η Εφαρμογή που θα παρουσιαστεί και θα αναλυθεί παρουσιάζει ένα σύστημα αυτόματης αποθήκευσης τοποθεσιών και απεικόνισής τους στον χρήστη ως Ημερολόγιο. Η αποθήκευση γίνεται στην βάση δεδομένων που είναι αποθηκευμένη στον server.

Οι τεχνολογίες που επιλέχθηκαν για την πλευρά του Server είναι οι node.js για τα web services και mongodb για την βάση δεδομένων. Επιλέχθηκαν αυτές οι τεχνολογίες διότι ο συνδυασμός τους κάνει ένα καλό ταίριασμα χρησιμοποιώντας την ίδια γλώσσα προγραμματισμού (Javascript). Επιπλέον, αυτό το πακέτο (stack) είναι πολύ δημοφιλές πλέον και χρησιμοποιείται σε συνδυασμό με εφαρμογές για κινητά και ιστοσελίδες.

Για την αποθήκευση του Server και της βάσης δεδομένων χρησιμοποιήθηκε ο Ωκεανός, που αποτελεί μία Cloud υπηρεσία για την ακαδημαϊκή και ερευνητική κοινότητα της Ελλάδος και δίνει την δυνατότητα διατήρησης εικονικών μηχανών και απόδοσης διεύθυνσης IP.

Η εργασία χωρίζεται σε 2 βασικά μέρη. Στο θεωρητικό μέρος ,και στο πρακτικό μέρος.

Στο θεωρητικό κομμάτι αναλύονται και επεξηγούνται οι τεχνολογίες και η αρχιτεκτονική που χρησιμοποιήθηκαν για την κατασκευή της εφαρμογής και του Server. Επίσης, αναλύεται η

βιβλιογραφία και χρησιμοποιείται για να υποστηρίξει το θεωρητικό και το πρακτικό κομμάτι της παρούσας και τέλος γράφονται οι περιορισμοί τα συμπεράσματα και η μελλοντική συνέχεια της έρευνας.

Το θεωρητικό κομμάτι της εφαρμογής χωρίζεται σε 3 κομμάτια που ουσιαστικά απαρτίζουν και τις τεχνολογίες που χρησιμοποιήθηκαν στην παρούσα εργασία. Αυτά τα 3 κομμάτια είναι , το Front end , δηλαδή το application που αλληλοεπιδρά ο χρήστης ο server με τα web services(node.js) και η βάση δεδομένων (MongoDB). Σε κάθε κομμάτι αναλύεται η αρχιτεκτονική που χρησιμοποιήθηκε για να φτιαχτεί και αναλύεται βιβλιογραφικά η θεωρεία που το αντιπροσωπεύει.

Το επόμενο μέρος είναι το κομμάτι της υλοποίησης της εφαρμογής που επισυνάπτονται στην παρούσα θεωρητική εργασία και είναι διαθέσιμα στον αναγνώστη μέσω Github στην διεύθυνση **https://github.com/basilisfou/Dissertation**. Ουσιαστικά, παρουσιάζονται τα κυριότερα και πιο σημαντικά μέρη της του κώδικα της εφαρμογής αλλά και του server. Επίσης παρουσιάζονται τα διαγράμματα χρήσης (UML) της εφαρμογής του android. Τέλος, παρουσιάζεται η βάση δεδομένων και τα σχετικά διαγράμματα που την απαρτίζουν. Τέλος, ακολουθεί η βιβλιογραφική αναφορά της παρούσας.

Introduction - Εισαγωγή

The rapid growth of mobile devices the recent years has transform mobiles and can be compared as PCs in CPU and RAM possibilities. Moreover, mobiles obtained many integrated sensors such as Accelerometer, GPS etc. Using, a very strong hardware, people can enjoy a PC in their hands, that is telephone, GPS, photograph machine etc. Likewise, the tablets had the same revolution. Tablet devices contain the same software and hardware as mobile devices but they have bigger screen, as a result they form perfect devices for internet and e-readers. For this reason, tablet have replaced netbooks, magazines and maybe books, because the user has the opportunity to download e-books. In Smartphone and tablets market, there are 4 operation systems, android, iOS, Windows Phone and blackberry. Both of them has an application market, where the user can buy and download many applications. Applications are programs, which are installed onto the OS of the smartphone and they offer a better experience to the user. For this reason, millions of application have been created, such us filter for photographs, web browsers, social media etc.

Τα τελευταία χρόνια τα κινητά τηλέφωνα απέκτησαν πολλές δυνατότητες εφάμιλλές ενός PC. Επίσης, τα κινητά τηλέφωνα απέκτησαν πολλούς ενσωματωμένους σένσορες, όπως το GPS, το αξελερόμετρο κ.α. Χρησιμοποιώντας δυνατό hardware οι άνθρωποι απέκτησαν έναν υπολογιστή στα χέρια τους που είναι συνάμα τηλέφωνο, GPS, φωτογραφική μηχανή, βιντεοκάμερα και πολλά άλλα. Με το ίδιο τρόπο εξελίχθηκαν και τα tablets τα οποία ουσιαστικά περιέχουν σχεδόν το ίδιο hardware και software με τα κινητά αλλά διαθέτουν μεγαλύτερη οθόνη, με αποτέλεσμα να είναι άριστες ιντερνετικές συσκευές αλλά και e-readers. Τα tablets με την σειρά τους αντικατέστησαν τα netbooks, τα περιοδικά αλλά ίσως και το βιβλίο, καθώς ο χρήστης έχει την δυνατότητα να κατεβάσει βιβλία σε ηλεκτρονική μορφή. Στην αγορά των Smartphones και των Tablets έχουν επικρατήσει 4 λειτουργικά συστήματα, τα Android, τα iOS για συσκευές Apple, Microsoft Phone και blackberry OS για συσκευές blackberry. Και τα 4 λειτουργικά συστήματα χρησιμοποιούν ένα διαδικτυακό κατάστημα, όπου ο χρήστης έχει την δυνατότητα να κατεβάζει applications. Τα applications ή mobile apps ουσιαστικά είναι προγράμματα τα οποία εγκαθίστανται στο λειτουργικό σύστημα και προσφέρουν πρόσθετη εμπειρία στον χρήστη. Έτσι αναπτύχθηκαν εκατομμύρια applications τα οποία μπορούν να καλύψουν κάθε γούστο, όπως φίλτρα για την φωτογραφική μηχανή, web browsers , κοινωνικά δίκτια κ.τ.λ..

Ανασκόπηση πεδίου

Google Location History

Η εφαρμογή Location History της Google βοηθά τον χρήστη που την χρησιμοποιεί να καταγράψει τις τοποθεσίες που βρίσκεται και να κάνει ανασκόπησή των τοποθεσιών μέσα από ένα χρονολόγιο (timeline) . Οι συσκευές αυτές μπορεί να είναι κινητά ή tablets, όμως ο χρήστης θα πρέπει να έχει ανοικτό τον καταγραφέα τοποθεσιών έτσι ώστε να λαμβάνει τοποθεσίες. Επίσης ο χρήστης έχει το δικαίωμα να διαγράφει τοποθεσίες και να βλέπει στατιστικά και χρήση του ιστορικού του. [1]

Όπως και στην παρούσα εφαρμογή, ο χρήστης έχει πρόσβαση στις τοποθεσίες του μέσω ενός χάρτη αλλά και μέσω ενός χρονολογίου (Timeline) το οποίο μπορεί να φιλτράρει και να δει ανάλογα την ημερομηνία σε ποια τοποθεσία ή τοποθεσίες βρισκόταν. Το βασικότερο είναι, ότι ο χρήστης έχει μόνο αυτός πρόσβαση στο χρονολόγιο του και οι συσκευές που έχουν τον λογαριασμό του εγγεγραμμένο μπορούν να καταγράψουν αυτόματα την τοποθεσία του. Επιπροσθέτως, ο χρήστης έχει κάποιες δυνατότητες, όπως:

- Μπορεί να κατεβάσει τις τοποθεσίες του χρονολογίου του στον υπολογιστή του σε μόρφη JSON [2]
- Μπορεί να επεξεργαστεί τις υπάρχουσες καταγεγραμμένες τοποθεσίες του.

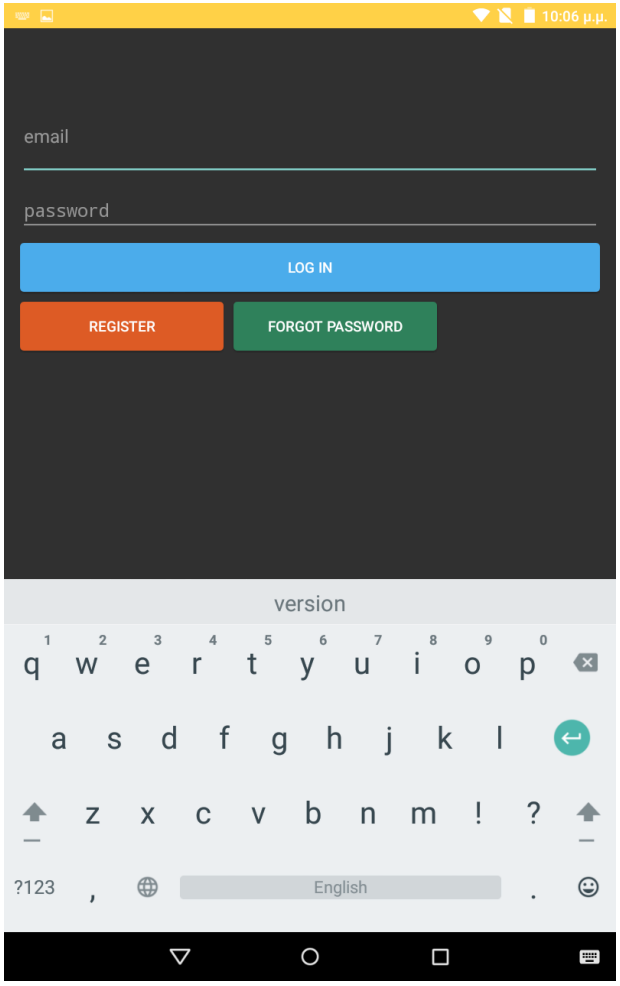
Swarm

Η εφαρμογή Swarm προσφέρετε για κινητά , android, iOS και windows phone. Σε αυτή την εφαρμογή ο χρήστης μπορεί να αποθηκεύσει μόνος του την τοποθεσία κάνοντας ένα “check in”. Επίσης μπορεί παράλληλα να επιλέξει να μπορούν οι φίλοι του να δουν που ακριβώς βρίσκεται στον χάρτη. Υπάρχει ένα timeline και ένας χάρτης ως γραφικές διεπαφές που προσφέρουν στον χρήστη ευκολία για να βλέπει το ιστορικό του και το ιστορικό των φίλων του. [3]

Παρουσίαση και χρήση εφαρμογής

Η παρούσα εφαρμογή έχει σαν σκοπό να καταγράφει αυτόματα την τοποθεσία του χρήστη ανά μισή ώρα. Αυτή η λειτουργία γίνεται στο background, έτσι ο χρήστης δεν χρειάζεται να κάνει τίποτα παρά μόνο να μην τερματίσει την εφαρμογή. Η τοποθεσία καταγράφεται από το κινητό του χρήστη και αυτόματα αποθηκεύεται στην βάση δεδομένων μέσω των web services. Καθώς ο χρήστης κάνει εγκατάσταση την εφαρμογή θα πρέπει να κάνει σύνδεση στην υπηρεσία ή να εγγράψει τον εαυτό του. Επίσης έχει και την δυνατότητα να κάνει reset το κωδικό πρόσβασης έτσι ώστε άμα τον ξεχάσει να μπορεί να συνδεθεί με καινούργιο.

Ο χρήστης πρέπει να συνδεθεί στην εφαρμογή βάζοντας το email του, που έχει δημιουργήσει τον λογαριασμό του και τον σωστό κωδικό του. Ύστερα πρέπει να πατήσει Log in, έτσι ώστε να χρησιμοποιήσει την εφαρμογή και να περάσει στην επόμενη οθόνη. Σε περίπτωση που δεν υπάρξει επιτυχία στην σύνδεση θα εκτυπωθεί στην οθόνη ένα μήνυμα λάθους. Αφού ο χρήστης συνδεθεί με τον λογαριασμό, αυτόματα καταγράφεται και η τοποθεσία του και αρχίζει ένα αντίστροφο χρονόμετρο στο background μισής ώρας για να καταγράψει την επόμενη τοποθεσία του (Traking). Επίσης κάθε φορά που καταγράφεται η τοποθεσία του ένα μήνυμα εμφανίζεται στην οθόνη και ένας ήχος τον ειδοποιεί.



Εικόνα 1: Log in screen

Ο χρήστης έχει διάφορες επιλογές να κάνει μέσα στην εφαρμογή. Πρώτον έχει διαθέσιμο ένα hamburger menu στο οποίο μπορεί να προηγηθεί στην εφαρμογή. Η πρώτη επιλογή έχει να κάνει με την διαχείριση του λογαριασμού του. Σε αυτήν την επιλογή μπορεί να αλλάξει τον λογαριασμό, αφού πρώτα δώσει κάποια στοιχεία του λογαριασμού του και αυτά επαληθεύουν. Η δεύτερη επιλογή που μπορεί να κάνει είναι να αποσυνδεθεί από την εφαρμογή , δηλαδή να αποσυνδεθεί από τον προσωπικό του λογαριασμό. Μόλις γίνει αυτό ο χρήστης θα αντικρίσει πάλι την log In screen και δεν θα έχει στην διάθεσή του καμία λειτουργία την εφαρμογής μέχρι να συνδεθεί πάλι. Η Τρίτη επιλογή είναι ο χάρτης της google. Σε αυτήν την οθόνη μπορεί να δει το ιστορικό του πάνω στον χάρτη. Δηλαδή με Pins μπορεί να δει ποια μέρη έχει επισκεφθεί. Τέλος η τέταρτη επιλογή μπορεί να δει το timeline του. Δηλαδή ποια μέρη επισκέφθηκε ανάλογα με την ημερομηνία. Έχει στην διάθεσή του ένα ημερολόγιο και πατώντας πάνω σε συγκεκριμένη ημερομηνία η εφαρμογή τον ενημερώνει σε ποιες τοποθεσίες βρέθηκε.

Διαγράμματα UML

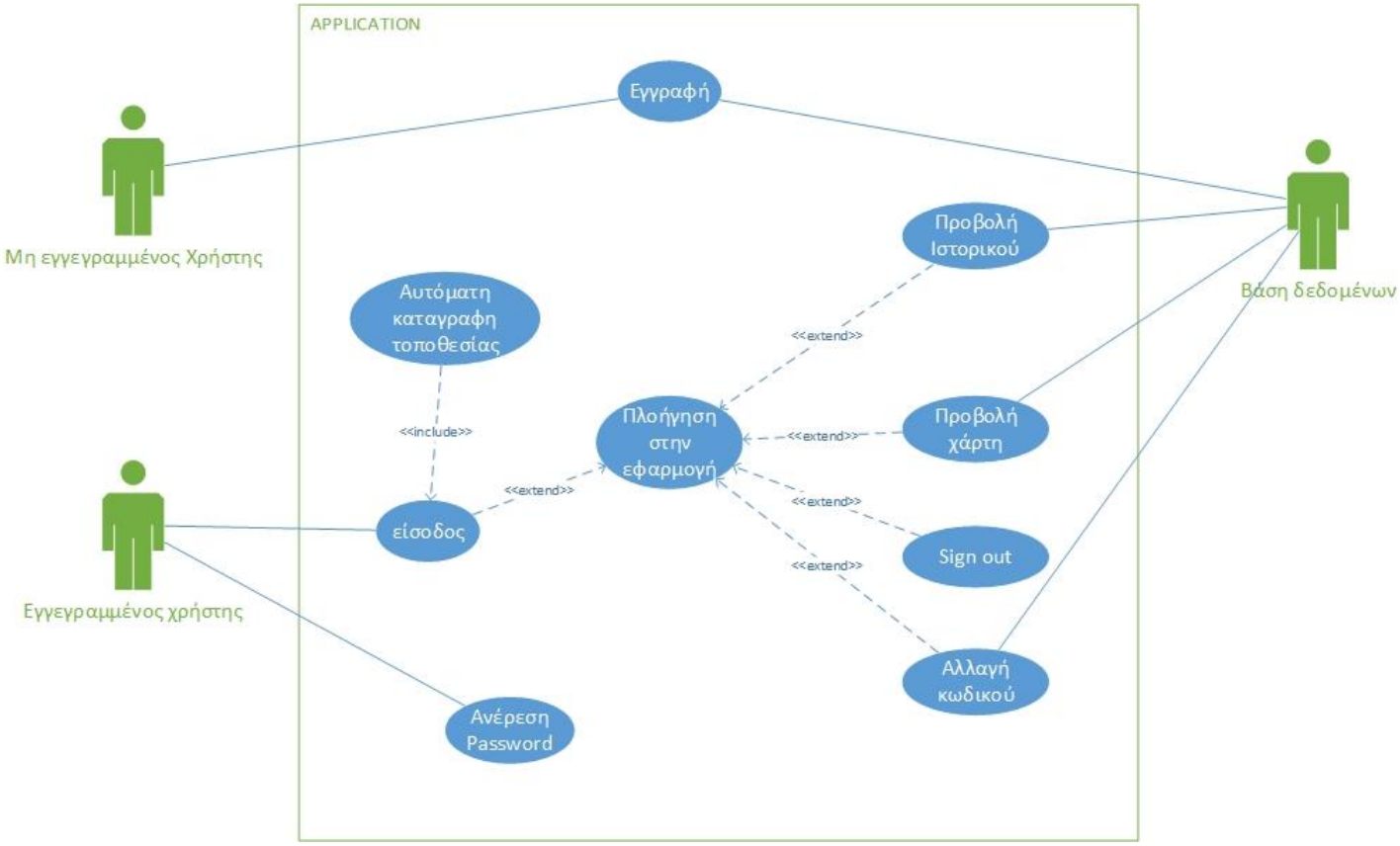
Στο παρόν κεφάλαιο θα καταγραφούν δυο σημαντικά σχεδιαγράμματα του Λογισμικού. Το UML διάγραμμα Περιπτώσεων Χρήσης (use case) και το διάγραμμα τάξεων (Class diagram).

Διάγραμμα Περιπτώσεων Χρήσης (use case)

Το διάγραμμα περίπτωσης χρήσης απεικονίζει την αλληλεπίδραση του συστήματος με τους χρήστες του ή και με άλλα συστήματα. Επίσης, περιγράφει με σχηματικό τρόπο τους χρήστες του συστήματος και τον τρόπο με τον οποίο αναμένουν να αλληλεπιδρούν με αυτό.

Οι περιπτώσεις χρήσης είναι ένα σύνολο σεναρίων που συνδέονται με ένα συγκεκριμένο σκοπό του χρήστη. Στην πραγματικότητα το use case είναι οι λειτουργίες του συστήματος οι οποίες παριστάνονται με ένα πιο οργανωμένο τρόπο. [4]
Τα διαγράμματα Περίπτωσης Χρήσης στοχεύουν στο να:
1) Καθοριστούν και να περιγραφούν οι λειτουργικές απαιτήσεις του συστήματος
2) Δίνουν μια σαφή και συνεπή περιγραφή για το τι θα πρέπει να κάνει το σύστημα
3) Παρέχουν την ικανότητα να εντοπίζονται οι λειτουργικές απαιτήσεις μέσα στις κλάσεις και τις λειτουργίες του συστήματος.

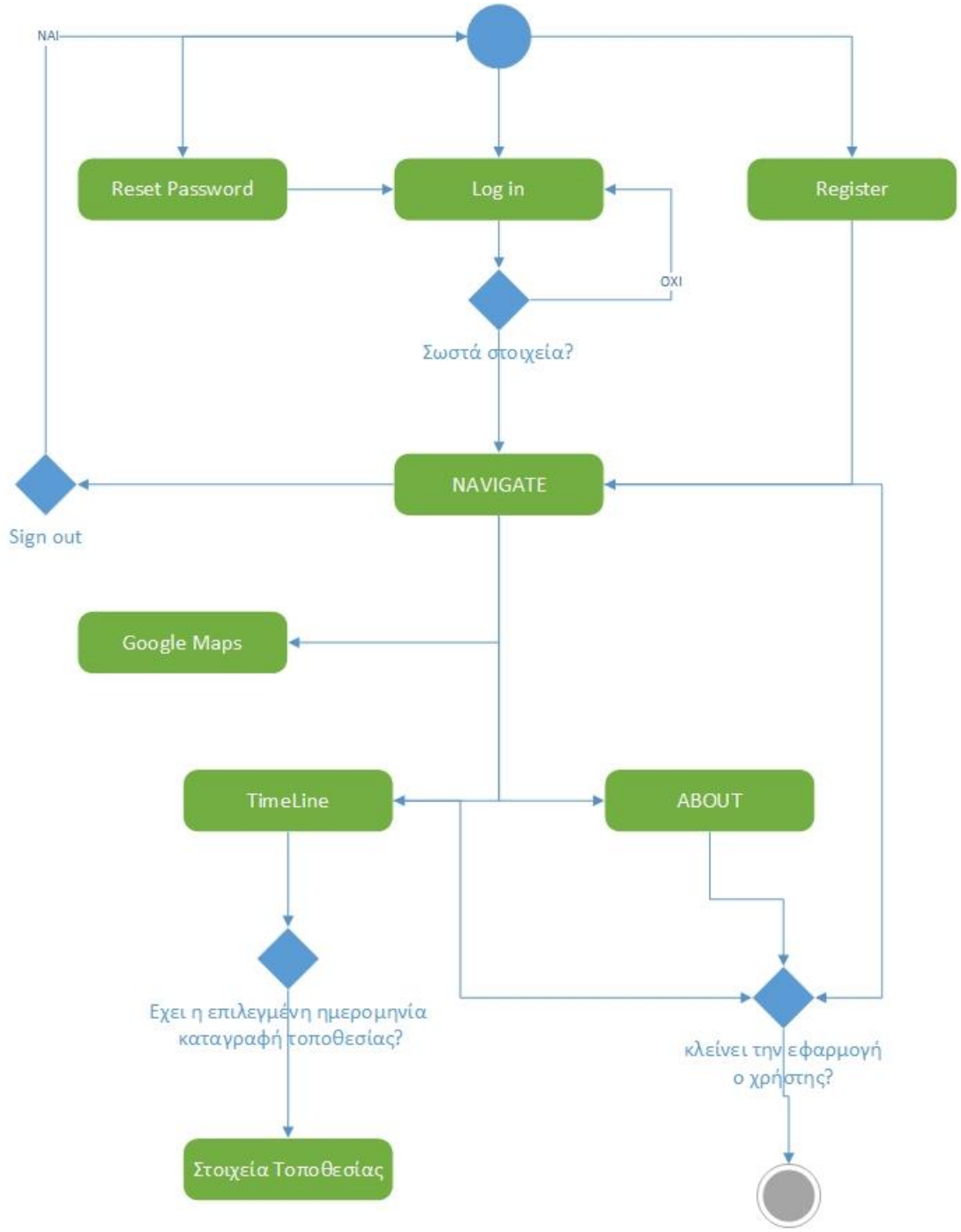
Οι χαρακτήρες (actors) είναι τα αλληλοεπιδρώμενα με το σύστημα μέλη. Οι χαρακτήρες αυτοί ενδέχεται να είναι είτε φυσικά πρόσωπα είτε άψυχα αντικείμενα.



Εικόνα 2 - Use Case Διάγραμμα

Διάγραμμα δραστηριότητας (Activity Diagram)

Το διάγραμμα δραστηριοτήτων είναι μια ειδική κατάσταση διαγράμματος στο οποίο όλες (ή τουλάχιστον οι περισσότερες) από τις μεταβάσεις προκαλούνται από την ολοκλήρωση των δράσεων των πηγαίων καταστάσεων. Σκοπός του διαγράμματος αυτού είναι να εστιάσει σε ροές οι οποίες προκαλούνται από την εξωτερική επεξεργασία. Χρησιμοποιείται στις περιπτώσεις όπου όλα ή τα περισσότερα συμβάντα αναπαριστούν την ολοκλήρωση εσωτερικά παραγόμενων δράσεων ,δηλ. την διαδικαστική ροή ελέγχου. [5]



Εικόνα 3: - Activity Diagram

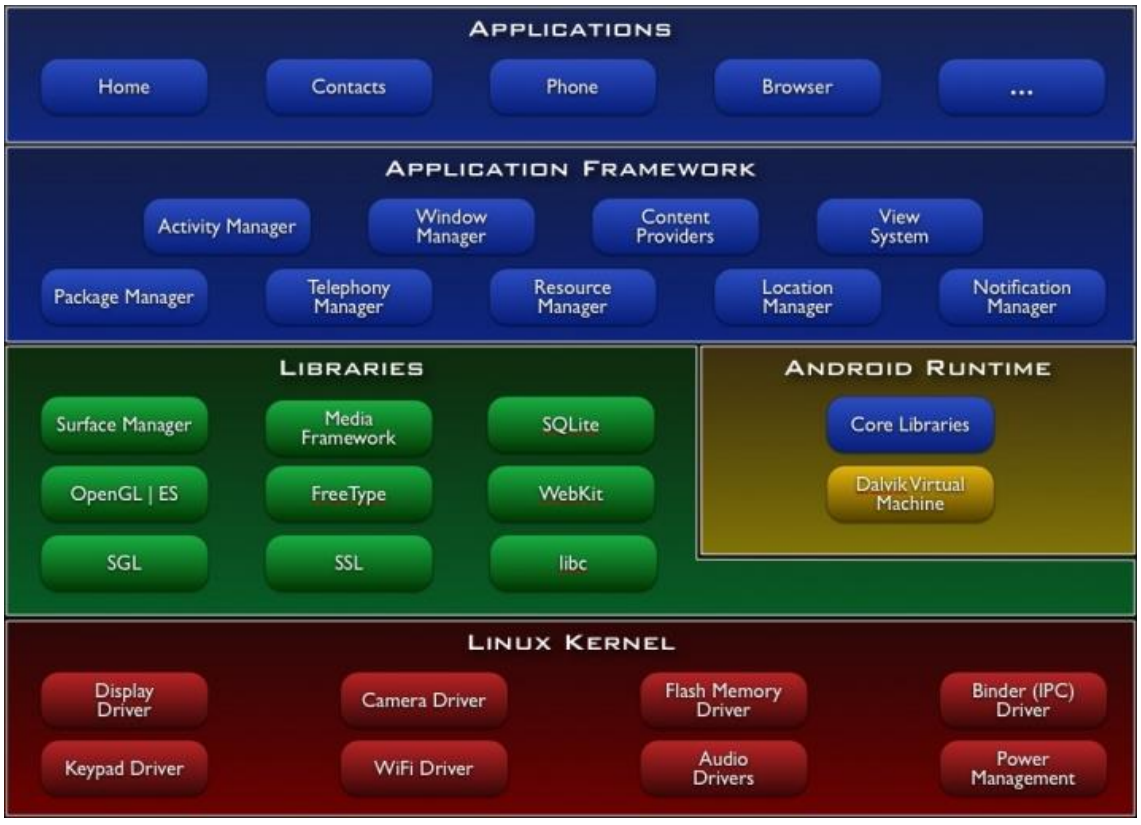
Frontend – Android Application

Εισαγωγή στο Android

Το Android είναι ένα λειτουργικό σύστημα που βασίζεται στον πυρήνα του Linux και σχεδιάστηκε για smartphones, tablet PCs και άλλες embedded συσκευές (π.χ. google watch κ.α.) [6]. Κατασκευάστηκε από την Google και από τα μέλη της Open Handset Alliance και αποτελεί από τα πρώτα mobile Λ.Σ. ανοικτού κώδικα. Το Λ.Σ. Android ήταν από τις πρώτες πλατφόρμες που πραγματικά διαχώρισαν το Software από το Hardware. [7] στις συσκευές, με αποτέλεσμα να το ενσωματώσουν όλο και περισσότεροι κατασκευαστές mobile συσκευών, σε αντίθεση με το Λ.Σ. iOS της Apple που χρησιμοποιείται αποκλειστικά για συσκευές Apple (iPhone , iPad , iPod κ.α.). Σήμερα, κατέχει την πρώτη θέση στην αγορά των smartphones. Το γεγονός αυτό καθιστά την αγορά των εφαρμογών Android μία πολλά υποσχόμενη περιοχή για ανάπτυξη επιχειρηματικής δραστηριότητας. Το store του Android λέγετε Google Play και φιλοξενεί εκατομμύρια application, τα οποία γράφονται σε γλώσσα java.

Το 2007 κυκλοφόρησε ο κώδικα του Android από την Google υπό την άδεια Apache[1]. Το 2008 κυκλοφόρησε στην αγορά το πρώτο κινητό με Android OS, το G1 phone της HTC [7]. Έπειτα, ακολούθησαν πολλές εταιρίες κινητής τηλεφωνίας που ενσωμάτωσαν το Android OS στις συσκευές τους με αποτέλεσμα να είναι το πιο δημοφιλές Λ.Σ. για tablets και για κινητά τηλέφωνα. Τελευταία έκδοση του λειτουργικού Android είναι η έκδοση 6 (Marshmallow).

Το Android αποτελείται από 4 επίπεδα όπως φαίνεται και στην παρακάτω εικόνα:



Εικόνα 4: Αρχιτεκτονική του Android - *http://elinux.org/Android_Architecture*

Στο πρώτο επίπεδο βρίσκεται ο πυρήνας Linux ο οποίος είναι υπεύθυνος για: τη διαχείριση των drivers (οθόνης, wifi, κάμερα κ.τ.λ.), τον έλεγχο πρόσβασης στους πόρους του

[1] http://www.apache.org/licenses/

συστήματος, τη διαχείριση μνήμης και γενικά όλες τις υπηρεσίες που παρέχει ένα λειτουργικό σύστημα. Στο δεύτερο επίπεδο βρίσκεται το Android Runtime[2] που φιλοξενεί την εικονική μηχανή Dalvik Virtual Machine (DVM) και την υλοποίηση των βασικών βιβλιοθηκών της java που προσφέρονται στον προγραμματιστή κατά την διαδικασία ανάπτυξης των δικών του εφαρμογών. Επίσης σε αυτό το επίπεδο υπάρχουν βασικές βιβλιοθήκες (Native) του συστήματος που είναι γραμμένες σε C/C++. Κάθε Application γραμμένο σε java τρέχει στην εικονική μηχανή Dalvik. Στο τρίτο επίπεδο βρίσκεται η βάση προγραμματισμού των applications, οι android developers τον περισσότερο χρόνο τους δουλεύουν με το πλαίσιο αυτό. Το επίπεδο αυτό προσφέρει στον προγραμματιστή μια πληθώρα δυνατοτήτων σχετιζόμενων με το λειτουργικό σύστημα και την συσκευή που το φιλοξενεί, όπως πρόσβαση στα διάφορα περιφερικά της συσκευής κ.τ.λ.. Το επίπεδο αυτό περιλαμβάνει τον Activity manager, Windows manager, Notification manager, Location manager κ.τ.λ. Τέλος, στο τελευταίο στρώμα της στοίβας βρίσκονται οι εφαρμογές για το περιβάλλον Android, κατασκευασμένες σε Java. [8]

Πλέον χρησιμοποιείται το android studio για την δημιουργεία applications, όπως και χρησιμοποιήθηκε για την κατασκευή του application της εργασίας αυτής. Το android studio χρησιμοποιεί το Grandle για την κατασκευή, αποσφαλμάτωση και μεταγλώττιση των εφαρμογών android. Στην εικόνα του παραρτήματος, επισημαίνεται ο κύκλος λειτουργίας του Grandle.

Όπως γίνεται φανερό το input του Grandle συστήματος είναι ο κώδικας και τα resources της εφαρμογής και μαζί με διάφορες κλάσεις μεταγλωττίζεται ο κώδικας από το διερμηνευτή της java και παράγεται ένα αρχείο .class . Τελικό αποτέλεσμα την όλης διαδικασίας είναι ένα αρχείο APK το όποιο τρέχει μόνο σε λογισμικό android. Για να ανεβεί η εφαρμογή στο Google market χρειάζονται να γίνουν κάποιες διαδικασίες, έτσι ώστε να συμπιεσθεί κάπως το APK και να μην είναι τόσο μεγάλο σε μέγεθος. Αυτή η διαδικασία παράγει το signed-apk και μόνο αυτό μπορεί να ανεβεί στο market.

Όντας ένα λειτουργικό σύστημα ανοιχτού κώδικα, το Android, δεν θα μπορούσε παρά να διαθέτει και ένα σύνολο εργαλείων ανάπτυξης που επίσης είναι ανοιχτού κώδικα. Η προγραμματιστική πλατφόρμα για android ονομάζεται Android software development kit (SDK), υποστηρίζει εφαρμογές με πλούσια λειτουργικότητα, δηλαδή οι εφαρμογές μπορούν να αξιοποιήσουν τις δυνατότητες των κινητών, π.χ. χρήση κάμερας, GPS, γυροσκόπιο, 3D graphics κ.τ.λ. [9] Ορισμένες φορές δεν χρειάζεται καν ένα κινητό με Android OS διότι το SDK περιέχει έναν προσομοιωτή (emulator) που προσομοιώνει ένα κινητό που τρέχει Android. Επίσης, το SDK μπορεί να εγκατασταθεί σε Windows , Linux και Mac OS.

Διασύνδεση με το δίκτυο

Εισαγωγή

Το networking[3] δεν γίνεται άμεσα. Για έναν web Server μπορεί να πάρει 1 με 2 δευτερόλεπτα για να ανταποκριθεί και το κατέβασμα ενός αρχείου μπορεί να πάρει ακόμα περισσότερο. Επειδή το networking μπορεί να κρατήσει τόσο πολύ, το android δεν αφήνει τις λειτουργίες networking να εκτελεστούν στην κεντρική διεργασία του android (Main thread)[4] [10].

Για να γίνει αντιληπτό αυτό, θα πρέπει να εξηγηθεί τι ακριβώς είναι ένα thread. Thread είναι μία μοναδική ακολουθία εκτέλεσης. Ο κώδικας που εκτελείται μέσα σε ένα τέτοια thread εκτελείται συριακά. Όμως το Main thread δεν δουλεύει έτσι, αλλά κάθετε σε ένα άπειρο loop (επανάληψη) και περιμένει για γεγονότα (events) από τον χρήστη ή από το σύστημα. Οπότε εκτελεί κώδικα ως ανταπάντηση σε αυτά τα γεγονότα, όταν αυτά βέβαια πραγματοποιηθούν [10].Σύμφωνα με τα παραπάνω, άμα το android χρησιμοποιούσε το main thread για λειτουργίες

[2] Android runtime: Βασικός μηχανισμός που απαιτείται για την εκτέλεση των εφαρμογών που αναπτύσσονται για το περιβάλλον του Android.

[3] Networking: Διασύνδεση πελάτη με τον Server για την ανταλλαγή δεδομένων

[4] Main thread: Όταν ένα συστατικό στοιχείο (component) του android ξεκινά, τα αρχεία συστήματος του android ξεκινάνε μία διεργασία Linux για το στοιχείο αυτό. Κατά σύμβαση, όλα τα στοιχεία της ίδιας εφαρμογής τρέχουν σε μία μονή διεργασία, η οποία ονομάζεται (Main thread). Το Main Thread είναι υπεύθυνο για την λειτουργία του γραφικού περιβάλλοντος της εφαρμογής.

επίπονες, όπως είναι και η λειτουργία του networking, τότε ο χρήστης θα είχε μία εμπειρία ANR[5], που αυτή η εμπειρία θα τον απέτρεπε να χρησιμοποιήσει οποιαδήποτε εφαρμογή. Η λύση για να μην επιφορτίζεται με πολλές λειτουργίες η Main thread είναι η χρησιμοποίηση worker thread[6], που κάνουν όλη την επίπονη δουλειά σε διαφορετική διεργασία. Με αυτόν το τρόπο, το networking γίνεται ομαλά και χωρίς να γίνεται αντιληπτό από τον χρήστη. Ο τρόπος για να γίνει αυτό είναι με διάφορες κλάσεις που παρέχει το android, όπως η κλάση AsyncTask. Επίσης υπάρχουν και βιβλιοθήκες όπως το Retrofit ή το Volley. Η βιβλιοθήκη Volley είναι μία βιβλιοθήκη HTTP και είναι η προτεινόμενη λύση της google για την μετάδοση δεδομένων στο android.

Βιβλιοθήκη Volley

Η βιβλιοθήκη Volley είναι μία βιβλιοθήκη HTTP και είναι η προτεινόμενη λύση της google για την μετάδοση δεδομένων στο android. Οι λόγοι που επιλέχθηκε στην εφαρμογή η βιβλιοθήκη αυτή είναι οι εξής [11] :

1. Αυτόματος χρονοπρογραμματισμός (scheduling) των διαδικτυακών αιτημάτων (request).
2. Πολλαπλές ταυτόχρονες διαδικτυακές συνδέσεις
3. Αυτόματη αποθήκευση των απαντήσεων (response) στην μνήμη cache
4. Ιεράρχηση των αιτημάτων
5. Δυνατότητα ακύρωσης ενός αιτήματος
6. Αυτόματη δημιουργία ξεχωριστού thread που

Με την βιβλιοθήκη αυτή ένα application (client) μπορεί να επικοινωνήσει με έναν Server με διάφορες μεθόδους χρησιμοποιώντας το πρωτόκολλο HTTP . Το κινητό(client) μπορεί να λάβει πληροφορίες σε μορφή JSON ή XML , επίσης μπορεί να λάβει ή να στείλει εικόνες και βίντεο. Στην παρούσα εργασία η επικοινωνία ανάμεσα σε Server και client γίνεται με JSON.

Στον παρακάτω κομμάτι κώδικα (snipper), παρουσιάζεται ένα αίτημα που γίνεται στον server για να μπορέσει ο χρήστης να εγγραφεί στην Υπηρεσία.

[5] ANR: application not responding (περίπτωση που η εφαρμογή δεν υπακούει τις εντολές του χρήστη λόγο φόρτου εργασίας, πχ σε ένα event όπως το πάτημα ενός κουμπιού)
[6] Worker Thread: Διεργασία εργάτης, διεργασία που τρέχει στο παρασκήνιο

code snippet 1: Register.java Αίτημα του application έτσι ώστε να εγγραφεί ένας χρήστης

```
private final static String URL = "http://83.212.113.58:3000/users/register";
final JSONObject jsonBody;

try {
    jsonBody = new JSONObject("{\"name\":\""+nameString+"\", " +
"\"surname\":\""+surnameString+"\",\"email\":\""+emailString+"\",\"userName\":\""+
usernameString+"\"," +"\"passWord\":\""+passwordString+"\"}");
JsonObjectRequest request = new JsonObjectRequest(Request.Method.POST,URL,
        jsonBody, new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                try{
                    messageResponce = response.getString("response");
                    res = response.getBoolean("res");
                    token = response.getString("token"); // token for the new user
                }catch (JSONException e){
                    e.printStackTrace();
                }
                 }
            }, new Response.ErrorListener(){

            @Override
            public void onErrorResponse(VolleyError error) {

           }
        });
       Volley.newRequestQueue(context).add(request);
} catch (JSONException e){
    e.printStackTrace();
    Log.d("billy",e.toString());
}
```

Εύρεση τοποθεσίας στο Android

Location API

Το android παρέχει τα βασικά εργαλεία (Location - API) για τις τοποθεσίες, έτσι ώστε να μπορεί μία εφαρμογή να εποπτεύει την τοποθεσία του κινητού από διάφορες πηγές, όπως το GPS radio ή τις κεραίες της κινητής τηλεφωνίας . Αυτό το API υπάρχει από τότε υπάρχει το android και βρίσκεται στο πακέτο android.location [10].

Το android επιλέγει πηγές ανάλογα με τι σήμα λαμβάνει από αυτές. Για παράδειγμα το GPS είναι το καλύτερο σε θέμα ακρίβειας, άμα δεν υπάρχει καλό σήμα GPS τότε το σήμα από τις κεραίες κινητής τηλεφωνίας μπορεί να παρέχει μεγαλύτερη ακρίβεια στον χρήστη.

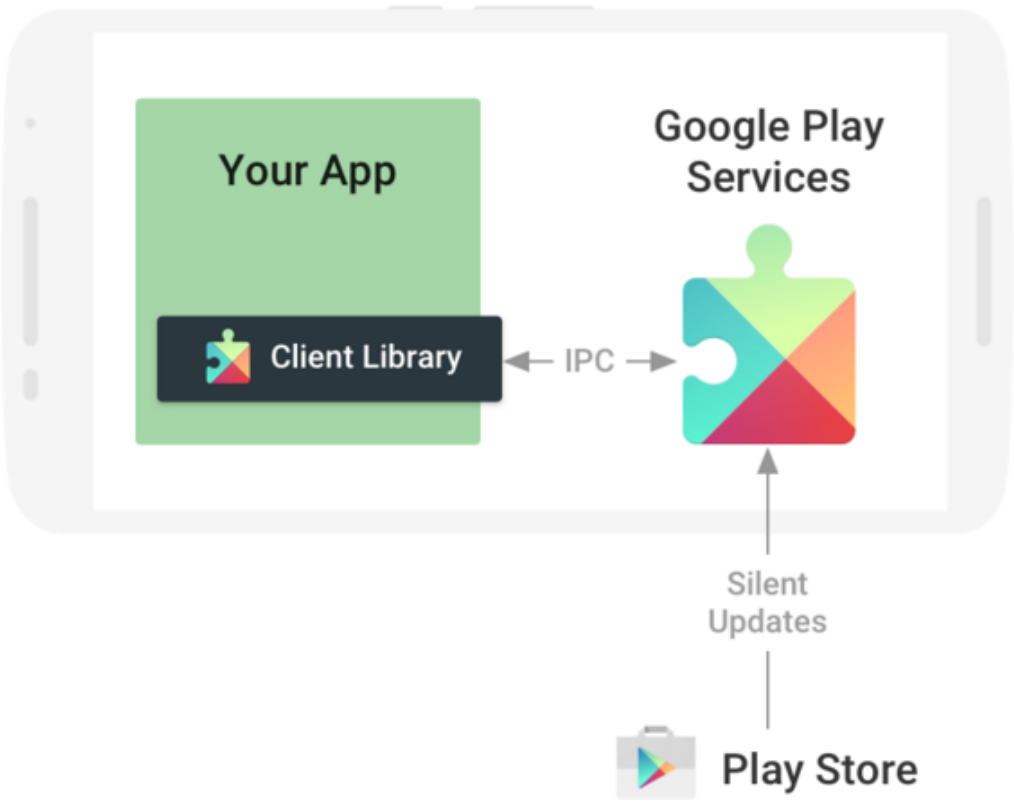
Το συγκεκριμένο API έχει κάποιες ατέλειες. Ο λόγος είναι ότι στον πραγματικό κόσμο, τα application κάνουν αιτήματα και προσπαθούν να βρουν την ακριβής τοποθεσία (Fine Location), όμως αυτή η προσπάθεια κοστίζει σε μπαταρία. Αυτό αποτέλεσε πρόβλημα όταν οι συσκευές μετακινούνται έξω [10].

Google Play Service

Χρειαζόταν ένα καλύτερο API για την καταγραφή των τοποθεσιών, αλλά δεν θα μπορούσε να προέλθει από την βασική βιβλιοθήκη του android, όπως το Location API, διότι θα έπαιρνε χρόνια για να το χρησιμοποιήσουν οι προγραμματιστές.

Ευτυχώς, η Google είχε και άλλον τρόπο, αντί να ενσωματώσει τον καινούργιο κώδικα στην βασική βιβλιοθήκη του android. Ο τρόπος αυτός είναι με τα Google play services. Τα Google play services είναι μία σειρά υπηρεσιών τα όποια εγκαθίστανται μαζί με το Google play store application. Έτσι η google για να δώσει στους προγραμματιστές και ένα διαφορετικό API για καταγραφή τοποθεσιών, ενσωμάτωσε μία καινούργια υπηρεσία στα Google paly services με όνομα Fused Location Provider [10]. Αυτή η υπηρεσία θα χρησιμοποιηθεί και στην παρούσα εφαρμογή.

Όμως, το πρόβλημα είναι ότι εφόσον ο κώδικας βρίσκεται σε άλλη εφαρμογή θα πρέπει να υπάρχει στο κινητό, έτσι ώστε να λειτουργεί η εφαρμογή που δημιουργεί ο προγραμματιστής. Αυτό σημαίνει ότι μόνο η συσκευές με Google Play Store είναι εγκαταστημένο και αναβαθμισμένο θα μπορούν να χρησιμοποιούν τις εφαρμογές που περιέχουν την υπηρεσία Fused Location Provider. Μπορεί να διαπιστωθεί και από την παρακάτω εικόνα.



Εικόνα 5: Google play Services - Πηγή: https://developers.google.com/android/guides/overview#the_google_play_services_apk

Τέλος, το παρακάτω κομμάτι κώδικα παρουσιάζει τον τρόπο που χρησιμοποιήθηκε η παραπάνω κλάση που περιεγράφηκε. Χρησιμοποιεί σαν κριτήριο την υψηλή ακρίβεια για την αποτύπωση της τοποθεσίας, όμως αυτό έχει σαν κόστος την εξάντληση της μπαταρίας.

```
public void getLocation(){
    mCallbacks = new GoogleApiClient.ConnectionCallbacks() {
        @Override
        public void onConnected(Bundle bundle) {
            mLocationRequest = LocationRequest.create();

mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
            mLocationRequest.setNumUpdates(1);
            mLocationRequest.setInterval(0);

LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, locationListener);
        }

        @Override
        public void onConnectionSuspended(int i) {

        }
    };
    onConnectionFailedListener = new
GoogleApiClient.OnConnectionFailedListener() {
        @Override
        public void onConnectionFailed(ConnectionResult connectionResult) {

        }
    };

    mGoogleApiClient = new GoogleApiClient.Builder(context).
            addApi(LocationServices.API)
            .addConnectionCallbacks(mCallbacks)
            .addOnConnectionFailedListener(onConnectionFailedListener)
            .build();


    mGoogleApiClient.connect();
    locationListener = new com.google.android.gms.location.LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            Log.d(TAG,"onLocationChanged ");
            latitude = location.getLatitude();
            longitude = location.getLongitude();
            Log.d(TAG, String.valueOf(latitude) + " " +
String.valueOf(longitude));
            saveLocation();
        }
    };
}
```

code snippet 2: - AutoCheckin.java καταγραφή της τοποθεσίας του κινητού με την χρήση google Play Services

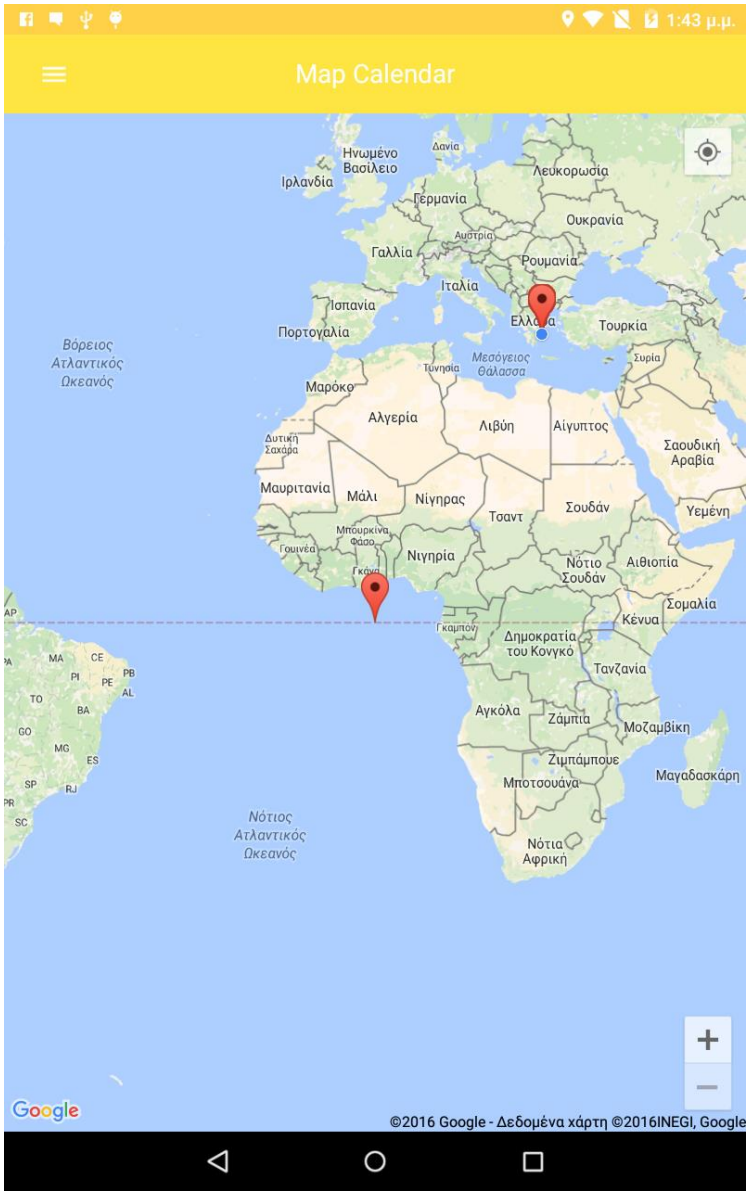
Google Maps

Οι χάρτες της Google στο android δεν ανήκουν στο API του android, παρόλο που υπάρχουν από την αρχή του android. Η πιο πρόσφατη έκδοση των google maps προέρχεται και αυτή από το google play services, όπως αναφέραμε παραπάνω [10]. Οπότε για να χρησιμοποιηθούν οι χάρτες

θα πρέπει ο χρήστης να έχει την πιο πρόσφατη έκδοση χαρτών. Επίσης θα πρέπει η εφαρμογή να διαθέτει ένα hash Key[7] το οποίο να ορίζεται στο manifest της εφαρμογής.

Η κλάση που περιέχει του χάρτες της Google είναι η *com.google.android.gms.maps* και περιέχει όλα τα απαραίτητα interfaces και κλάσεις που χρειάζονται για να χρησιμοποιηθούν οι χάρτες από ένα android κινητό [12]. Στο android οι χάρτες αναπαράγονται σε ένα mapView, το map view είναι σαν όλα τα άλλα views[8] του android. Στην παρούσα εργασία ο χάρτης ζει σε ένα mapFragment.

Παράδειγμα ενός χάρτη στην εφαρμογή ακολουθεί παρακάτω με κομμάτι κώδικα, όπου αρχικοποιείται στη εφαρμογή και ακολουθεί εικόνα από την εφαρμογή.



Εικόνα 6: Χάρτης της παρούσας εφαρμογής.

[7] Hash Key: Κλειδί που πρέπει να δημιουργηθεί από την κονσόλα Google Developer Console.
[8] Views: Καταλαμβάνει κομμάτι της οθόνης και αποτελεί γραφικό στοιχείο ενός activity ή fragment, παράδειγμα ενός view είναι ένα button ή LinearLayout

```
com.google.android.gms.maps.MapFragment map;
GoogleMap googleMap;
map = ((com.google.android.gms.maps.MapFragment)
getChildFragmentManager().findFragmentById(R.id.map));
googleMap = map.getMap();

//zoom controls in the map
googleMap.getUiSettings().setZoomControlsEnabled(true);
//control of the current location , pointer with the current location
googleMap.setMyLocationEnabled(true);
//compass enable
googleMap.getUiSettings().isCompassEnabled();
```

code snippet 3: MapFragment.java κομμάτι κώδικα που αρχικοποιήται ένα αντικείμενο τύπου google map και ορίζονται κάποιες ιδιότητες του.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class = "com.google.android.gms.maps.MapFragment"
        android:layout_alignParentTop="true"
        android:layout_alignParentBottom="true" />

</RelativeLayout>
```

code snippet 4: map_fragment το layout που ορίζεται το mapView , ως Fragment .

Services

Υπάρχουν δύο κατηγορίες Services, τα intentServices και τα Services. Τα Services είναι δομικά στοιχεία του android που μπορούν να εκτελέσουν μεγάλες διεργασίες στο παρασκήνιο και δεν χρειάζεται να υπάρχει UI[9], όπως χρειάζεται με τα activities. Τα services μπορούν να ξεκινήσουν από κάποιο άλλο δομικό στοιχείο του android, όπως activities, broadcast receivers κ.τ.λ. [13]. Επίσης, μπορούν να συνεχίσουν να τρέχουν ακόμα και αν σταματήσει η εφαρμογή που τα ξεκίνησε. Τα services, χρησιμοποιούν τα intents τα οποία είναι μηνύματα που περιέχουν τις οδηγίες για να ξεκινήσει ένα service [14], επίσης μπορούν να περιέχουν και δεδομένα που μεταφέρουν στα services. Τα intents των Services ονομάζονται Commands. Κάθε εντολή – Command είναι μία οδηγία στο service για να κάνει κάτι. Ανάλογα με το τύπο του Service κάθε εντολή μπορεί να εξυπηρετηθεί με διάφορους τρόπους [10].

Τα intentServices αποτελούν και αυτά δομικά στοιχειά του Android. Είναι σαν τα Activities, δηλαδή κληρονομούν από την κλάση του android Context και απαντούνε σε μηνύματα

[9] UI: γραφική διεπαφή

Intenets. Τα intentService δημιουργούν μία διεργασία εργάτη, η οποία διεργασία εκτελεί όλα τα intents τα οποία παραδίδονται στην onStartCommand του intentService. Αυτή η λειτουργεία γίνεται ξεχωριστά από τη κεντρική διεργασία της εφαρμογής (application's main thread) [13].

Τα intentServices καταναλώνουν εντολές - Commands με την μέθοδο της ουράς FIFO[10], οι καινούργιες εντολές πάνε στο τέλος της σειράς και εξυπηρετούνται οι παλιότερες από την μέθοδο onHandleIntenet [13]. Ακολουθεί το παράδειγμα από την παρούσα εργασία

code snippet 5 : onHandleIntent – autoCheckin.java

```
// Invoked on the worker thread
// Do some work in background without affecting the UI thread
@Override
protected void onHandleIntent(Intent intent) {
    context = getApplicationContext();
    intent.getExtras();
    token = intent.getStringExtra("token");
    scheduler.scheduleWithFixedDelay(runnable, 0, TIMEOUT, TimeU-
nit.MILLISECONDS);
}
```

Όταν τελειώσουν οι εντολές τα intentServices καταστρέφονται ή σταματάνε τον εαυτό τους [10].

Το πλεονέκτημα που προσφέρει το IntentService είναι ότι εκτελεί τις εντολές στο παρασκήνιο σε ξεχωριστή διεργασία, με αποτέλεσμα να μην επηρεάζει την ομαλή ροή της εφαρμογής. Έτσι είναι χρήσιμο όπως αναφέρθηκε και παραπάνω για μεγάλες διεργασίες, όπως π.χ. για ανέβασμα μιας φωτογραφίας στο internet ή για κατέβασμα αρχείων JSON ή XML, κτλ. .Έτσι στην παρούσα εργασία θεωρήθηκε σωστό να χρησιμοποιηθεί ένα IntentService για την αυτόματη καταγραφή της τοποθεσίας του χρήστη και την αποθήκευση της στον server του Okeanou , μέσω ενός service.

Έτσι, αυτόματα ανά μισή ώρα το intentService κάνει αίτημα για τοποθεσία και την αποθηκεύει στον server. Αυτό γίνεται με την βοήθεια ενός Timer τύπου ScheduledExecutorService. Ο ScheduledExecutorService είναι ένας timer που μπορεί να προγραμματίσει εντολές για να εκτελούνται μετά από συγκεκριμένο χρόνο στο μέλλον ή να εκτελούνται περιοδικά [15].

Με την παραπάνω δομή που αποτυπώθηκε ο χρήστης μπορεί να περιηγηθεί σε άλλες εφαρμογές, όμως ταυτόχρονα η εφαρμογή ανά μισή ώρα καταγράφει την τοποθεσία του και την αποθηκεύει στον server. Βέβαια βασική προϋπόθεση είναι να έχει ανοικτή την τοποθεσία, έτσι ώστε να είναι δυνατή η καταγραφή.

Διαχείριση χρηστών

Για την διαχείριση των χρηστών στην παρούσα εφαρμογή έγινε με την χρήση των *SharedPreferences*. Αυτή η κλάση γράφει δεδομένα στον δίσκο με έναν

[10] FIFO: Η ουρά FIFO (First in – First out) αποτελεί έναν ακόμη θεμελιώδη ΑΤΔ που είναι παρόμοιος με τη στοίβα ώθησης προς τα κάτω, αλλά χρησιμοποιεί τον αντίθετο κανόνα για να αποφασίσει ποιο στοιχείο θα αφαιρεθεί με την μέθοδο remove. Αντί να αφαιρεθεί το στοιχείο που προστέθηκε, αφαιρείται το στοιχείο που έχει μείνει παραπάνω στην ουρά. [32]

SharedPreferences.Editor και μπορεί να διαβάζει τα δεδομένα σε οποιοδήποτε σημείο της εφαρμογής. Το μόνο που χρειάζεται είναι να δίδεται μία τιμή μορφής string, η οποία είναι το κλειδί για την ταυτοποίηση της μεταβλητής που αποθηκεύτε στον δίσκο του κινητού. [16] Ο παρακάτω κώδικας αποτελεί παράδειγμα εγγραφής στον δίσκο στην παρούσα εφαρμογή.

code snippet 6: LogIn.java - αποθήκευση του session του χρήστη

```
//into the app

private SharedPreferences pref;
private SharedPreferences.Editor editor;

pref = context.getSharedPreferences("sessionUser", 0);// 0 - for private
mode
editor = pref.edit();

if(res) {
    //Save the token for this user in order to maintain session
    editor.putString("token",token);
    editor.putString("username",username);
    editor.commit();
}
```

Για να διαβαστούν οι μεταβλητές από τον δίσκο το παρακάτω παράδειγμα είναι αντιπροσωπευτικό.

code snippet 7: Διάβασμα από τον δίσκο - ο χρήστης έχει πραγματοποιήσει LogIn

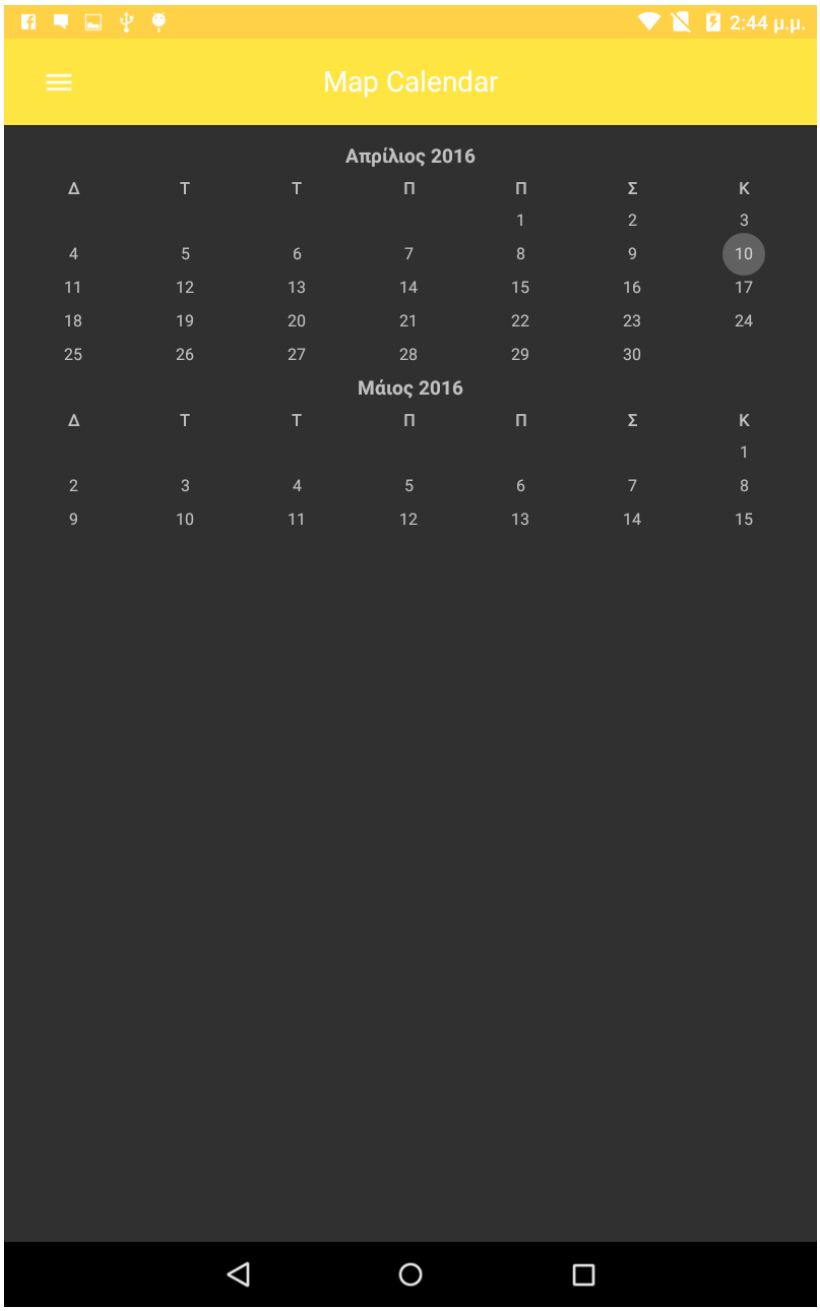
```

private SharedPreferences pref;
pref = context.getSharedPreferences("sessionUser", 0);// 0 - for private
mode
token = pref.getString("token",token);
```

Στην παρούσα εφαρμογή, όταν ο χρήστης κάνει sign in ή sign up, εγγράφεται το στον δίσκο το token του που προέρχεται από τον Server και το username του. Κάθε φορά που ο χρήστης κλείνει την εφαρμογή αυτές οι τιμές διατηρούνται. Έτσι κάθε φορά που ανοίγει ο χρήστης την εφαρμογή εξετάζεται άμα είναι συνδεδεμένος με τον λογαριασμό του και η οθόνη Log in παρακάμπτεται, όμως αν πραγματοποιήσει ο χρήστης sign out τότε αυτές οι τιμές γίνονται clear οπότε την επόμενη φορά που θα ανοίξει ο χρήστης την εφαρμογή θα δει την log in screen. Τέλος, άμα η εφαρμογή απεγκατασταθεί από το σύστημα οι τιμές αυτές σβήνονται από τον δίσκο.

Ημερολόγιο

Για να μπορέσει ο χρήστης να δει σε ποιες τοποθεσίες βρισκόταν χρησιμοποιήθηκε η κλάση CalendarView. Η κλάση αυτή είναι ένα widget το οποίο απεικονίζει ημέρες με την μορφή ενός ημερολογίου. Επίσης ο χρήστης μπορεί να επιλέξει και ημέρες από το ημερολόγιο. Επίσης ο χρήστης μπορεί να σκρολάρει έτσι ώστε να βρει την επιθυμητή ημερομηνία. [17]

Παράδειγμα χρήσης αυτού του Widget φαίνεται στην παρακάτω εικόνα η οποία είναι της παρούσας εφαρμογής.



Εικόνα 7: Ημερολόγιο που ο χρήστης μπορεί να επιλέξει οποία ημερομηνία θέλει και να δει πληροφορίες γι' αυτή την ημερομηνία.

```
public void initializeCalendar(View view){
    calendar = (CalendarView)view.findViewById(R.id.calendar);

    // sets whether to show the week number.
    calendar.setShowWeekNumber(false);

    // sets the first day of the of the week according the Calendar ,
here we set the MONDAY
    calendar.setFirstDayOfWeek(2);

    //setting the Seperator between weeks
    calendar.setWeekSeparatorLineColor(Color.WHITE);

    //setting the selected week color

calendar.setSelectedWeekBackgroundColor(getResources().getColor(R.col
or.SelectedWeekColor));


calendar.setFocusedMonthDateColor(getResources().getColor(R.color.Sel
ectMonthColor));
    //sets the listener to be notified upon selected date change.
    calendar.setOnDateChangeListener(new OnDateChangeListener() {
        @Override
        public void onSelectedDayChange(CalendarView view, int year,
int month , int dayOfMonth) {

            try {
                if(clicked) {
                    clicked = false;
                    month = month+1;
                    getCoordinates(dayOfMonth + "/" + month + "/" +
year);

                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
            textView.setText(String.valueOf(dayOfMonth) +"
"+String.valueOf(month)+"
"+String.valueOf(year),TextView.BufferType.NORMAL);
        }
    });
}
```

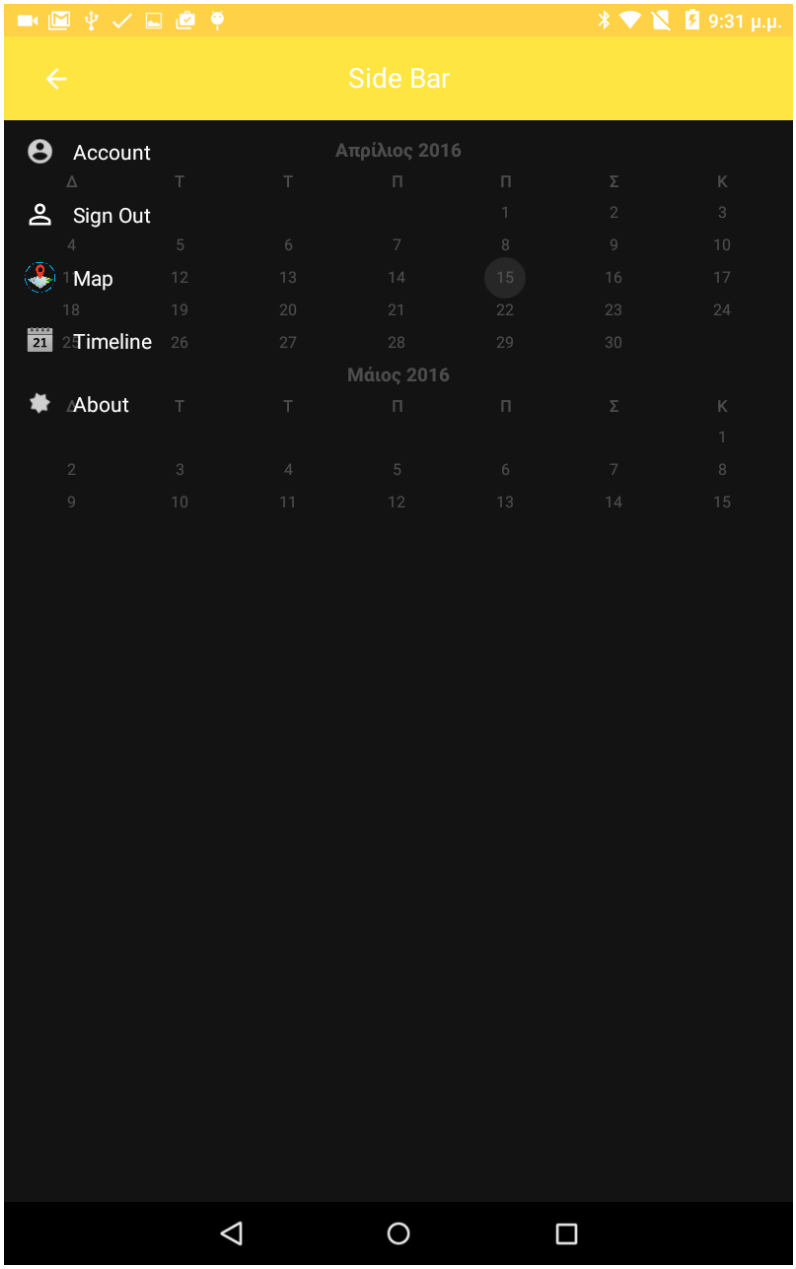
code snippet 8: CalendarFragment.java Παραμετροποίηση της κλάσης CalendarView

Συρόμενο Menu

Το συρόμενο menu ή αλλιώς navigation drawer ή hamburger menu χρησιμεύει για την πλοήγηση του χρήστη στην εφαρμογή. Αποτελεί μέρος της γραφικής διεπαφής της Main Activity και αποτελείται από links, που όταν τα πατά ο χρήστης μπορεί να προηγηθεί σε διαφορετική οθόνη που υποδεικνύει το link που πάτησε. Κάθε, link οδηγεί σε διαφορετικό Fragment, δηλαδή, οθόνη η οποία αποτελεί μέρος του MainActivity. Ο ρόλος των Fragments εξηγείται λίγο ποιο κάτω.

Το navigation drawer είναι ένα panel που απεικονίζει τις βασικές επιλογές πλοήγησης της εφαρμογής. Συνήθως βρίσκεται στην αριστερή πλευρά της εφαρμογής. [18] Τις περισσότερες

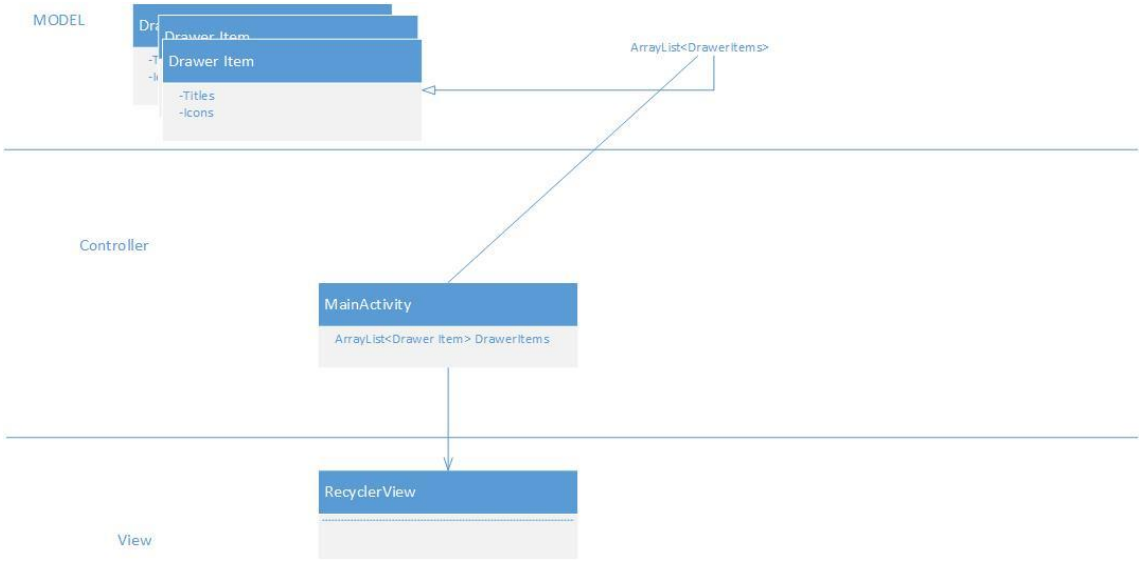
φορές δεν φαίνεται στον χρήστη παρά μόνο αν πατήσει τις 3 γραμμές που βρίσκονται στο Action Bar της εφαρμογής. Στην εικόνα που ακολουθεί μπορούμε να δούμε το Navigation Drawer της παρούσας εφαρμογής, το οποίο είναι ανοικτό.



Εικόνα 8 - Navigation Drawer

Μπορούμε να διακρίνουμε 5 αντικείμενα να υπάρχουν στο navigation drawer. Αυτά είναι, Account, Sign out, Map, Timeline και About. Στην δημιουργία των αντικειμένων έχει χρησιμοποιηθεί μία λίστα[11] , πιο συγκεκριμένα έχει δημιουργηθεί ένα Recycler View.

[11] Μια λίστα είναι ένα σύνολο στοιχείων όπου κάθε στοιχείο αποτελεί τμήμα ενός κόμβου (node) ο οποίος περιέχει ένα σύνδεσμό προς κάποιο κόμβο. [32]



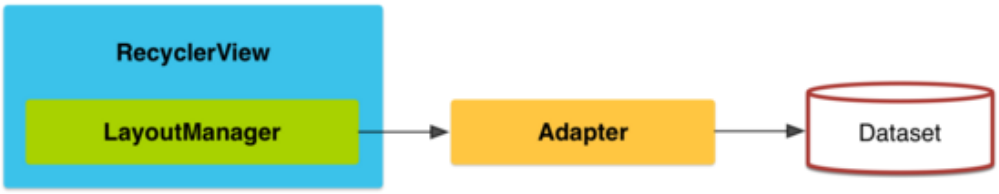
Εικόνα 9: Πηγή [9] , παράδειγμα προσαρμοσμένο για τις ανάγκες της παρούσας εφαρμογής.

Λίστα στοιχείων

Recycler View είναι ένα widget του android και λειτουργεί ως ένα δοχείο και απεικονίζει μεγάλες σειρές δεδομένων (data sets) και δίνει την δυνατότητα στον χρήστη να τα σκρολάρει με άνεση. [19]

Στην περίπτωση της εφαρμογής χρησιμοποιούμε το μοτίβο (Pattern) MVC[12]. Το Μοτίβο αυτό είναι ένα μοτίβο για την υλοποίηση γραφικής διεπαφής. Διαιρεί ένα πρόγραμμα σε 3 μέρη, έτσι ώστε να διαχωρίσει την εσωτερική απεικόνιση των πληροφοριών από τον τρόπο που αυτές οι πληροφορίες παρουσιάζονται στον χρήστη.

Το πρώτο μέρος του μοντέλου αυτού αποτελεί το controller. Ο controller στο android είναι ένα activity ή fragment, ουσιαστικά εκεί θα απεικονιστεί η λίστα. Το δεύτερο μέρος του μοντέλου αυτού είναι ο adapter. Ο adapter είναι υπεύθυνος να γεμίζει τα αντικείμενα τύπου της model κλάσης που υπάρχει.Τέλος το τρίτο μέρος είναι η κλάση model η οποία αποτελεί το δοχείο που θα μπουν τα δεδομένα που θέλουμε να απεικονίσουμε Στην παρακάτω εικόνα μπορεί να γίνει αντιληπτή αυτή η έννοια.



Εικόνα 10 - MVC Recycler View **[19]**

Το μοντέλο αντικείμενο το οποίο και αποτελεί ένα αντικείμενο της λίστας δίδεται από τον παρακάτω κώδικα. Αποτελείται από 2 πεδία , mTitle και mIcon. Το πεδίο mTitle αντιπροσωπεύει το όνομα του αντικειμένου που θα φαίνεται στο panel του navigation drawer και το mIcon αντιπροσωπεύει την εικόνα που θα φαίνεται αριστερά από το κείμενο. Οι εικόνες αυτές είναι

[12] MVC: model-view-controller pattern

αποθηκευμένες στον φάκελο drawable[13] του project. Επίσης μπορούμε να διακρίνουμε τις μεθόδους getter και setter και τον constructor της κλάσης. Οι μέθοδοι getter και setter χρησιμοποιούνται για να ορίσουν τιμές στις private μεταβλητές (setter) ή να πάρουν τιμές από τις μεταβλητές αυτές (getter) [20]. Τέλος η κλάση αυτή κληρονομεί από ένα interface, που ονομάζεται Serializable[14], ουσιαστικά με αυτό το interface μπορούμε να μετακινούμε την λίστα μέσω intents και bundles.

```
public class DrawerItem implements Serializable {
    private String mTitle;
    private int mIcom;
    private static final long serialVersionUID = 1L;

    public DrawerItem(String mTitle, int mIcom) {
        this.mTitle = mTitle;
        this.mIcom = mIcom;
    }

    public int getIcom() {
        return mIcom;
    }

    public void setIcoc(int mIcom) {
        this.mIcom = mIcom;
    }

    public String getTitle() {
        return mTitle;
    }

    public void setTitle(String mTitle) {
        this.mTitle = mTitle;
    }
}
```

code snippet 9 - Κλάση model DrawerItem.java

Ο adapter είναι αυτός που χρησιμεύει για να συνδέσει την Recycler View με τα δεδομένα [10]. Ουσιαστικά, αναλαμβάνει την λειτουργία να συνδέσει τα δεδομένα από το Model Layer που περιγράψαμε παραπάνω, με την γραφική διεπαφή κάθε αντικειμένου έτσι όπως αυτή ορίζεται στην εφαρμογή και να αναπαρασταθούν μέσα στην RecyclerVIew. Η κλάση χρησιμοποιεί το μοτίβο του ViewHolder[15], όπου ο adapter δημιουργεί τους απαιτούμενους ViewHolders και συνδέει τους ViewHolders με τα δεδομένα από το Model Layer.

[13] Drawable: Φάκελος που ανήκει στα Resources, ταξινομούνται στο android project σαν ακέραιοι και περιέχουν όλα τα αντικείμενα (assets) τα οποία μπορούν να ζωγραφιστούν στην οθόνη του κινητού.
[14] Serializable: Αυτό το interface βοηθά μόνο για serialization και deserialization, δηλαδή το αντικείμενο μπορεί να αναπαρασταθεί ως μια σειρά από bytes που περιέχουν τα δεδομένα του αντικειμένου, το τύπο του αντικειμένου και διάφορες πληροφορίες για το αντικείμενο αυτό. Αυτά τα αντικείμενα γράφονται σε ένα αρχείο και μετά ακολουθεί η λειτουργεία του deserialization που κάνει ακριβώς το αντίθετο πράγμα. [35]
[15] ViewHolder: Ο ViewHolder περιγράφει μία απεικόνιση (view) ενός αντικειμένου (item) και ως metadata την θέση του αντικειμένου αυτού μέσα στην λίστα. Ο ViewHolder ορίζεται ως inner class στον adapter της Recycler View. [36]

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/transparent">
    <LinearLayout
        android:id="@+id/inner_layout"
        android:layout_width="40dp"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/nv_title"
        android:layout_alignTop="@+id/nv_title"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">

        <ImageView
            android:layout_width="25dp"
            android:layout_height="25dp"
            android:id="@+id/nv_image"
            android:background="@android:color/transparent"
            android:scaleType="fitXY"
            android:layout_marginLeft="15dp"
            android:layout_gravity="center"/>
    </LinearLayout>
    <TextView
        android:id="@+id/nv_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="13dp"

android:minHeight="?android:attr/listPreferredItemHeightSmall"

android:textAppearance="?android:attr/textAppearanceListItemSmall"
        android:gravity="center_vertical"
        android:layout_centerVertical="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_toEndOf="@+id/inner_layout"
        android:layout_gravity="center"
        android:singleLine="true"
        android:ellipsize="end"
        android:layout_toRightOf="@+id/inner_layout" />
</RelativeLayout>
```

Εικόνα 11 - View του navigation - item

Για να φτιαχτεί ο adapter πρέπει να κληρονομεί από την κλάση RecyclerView.Adapter [10]. Ο adapter θα τυλίξει την λίστα με τα αντικείμενα DrawerItem και όταν η RecyclerVIew θέλει να απεικονίσει ένα αντικείμενο στην οθόνη, θα έχει μία συνομιλία με τον adapter της. Η RecyclerView καλεί την μέθοδο createViewHolder(ViewHolder,int) του adapter που είναι override[16] για να δημιουργηθεί ο ViewHolder μαζί με το view του αντικειμένου της RecyclerView. Ύστερα, η recyclerView καλεί την μέθοδο onBindViewHolder του adapter που είναι override. Η RecyclerView περνάει έναν ViewHolder σε αυτήν την μέθοδο παράλληλα με την θέση του στην λίστα. Ο adapter θα ενσωματώσει τα δεδομένα σε αυτήν την θέση στον ViewHolder και θα πάρει τα δεδομένα από την λίστα χρησιμοποιώντας την model κλάση (Drawer Item) και θα γεμίσει το View του αντικειμένου. Αφού τελειώσει αυτή η διαδικασία ο adapter βάζει ένα αντικείμενο της λίστας στην οθόνη [10]. Τέλος, η μέθοδος που είναι override getItemCount() μετρά τα αντικείμενα που υπάρχουν στην λίστα και έτσι ορίζεται το μέγεθος της Recycler View.

[16] Override: Όταν μία κλάση κληρονομεί από μία άλλη και η κλάση πατέρας έχει μία συνάρτηση όπου καλείται στην κλάση παιδί με το ίδιο όνομα και τις ίδιες παραμέτρους, η υλοποίηση στην κλάση πατέρα επικρατεί

```
public class RecyclerViewerAdapter extends
RecyclerView.Adapter<RecyclerViewerAdapter.CommentViewHolder> {
    private ArrayList<DrawerItem> mList;
    public static OnItemCustomClickListener listener;// Define listener member
variable ( Composition )

    /** Defines the listener interface - inner interface */
    public interface OnItemCustomClickListener {
        void onItemClick(View itemView, int position);
    }
    /**Sets the custom Listener - Define the method that allows the parent activity or
fragment to define the listener*/
    public void setOnItemCustomClickListener(OnItemCustomClickListener listener){
        this.listener = listener;
    }
    //Constructor initialise the list
    public RecyclerViewerAdapter(ArrayList<DrawerItem> pList){
        this.mList = pList;
    }
    /** View holder - holds the views of the navigation items*/
    public class CommentViewHolder extends RecyclerView.ViewHolder {
        public ImageView imageview;
        public TextView textView;
        //constructor of the inner class
        public CommentViewHolder(final View itemView){
            super(itemView);
            // Find the TextView in the LinearLayout
            imageview = (ImageView)itemView.findViewById(R.id.nv_image);
            textView = (TextView)itemView.findViewById(R.id.nv_title);
            itemView.setOnClickListener(new View.OnClickListener(){
                @Override
                public void onClick(View v){
                    // Triggers click upwards to the adapter on click
                    if(listener != null)
                        listener.onItemClick(itemView,getAdapterPosition());
                }

            });
        }
    }
    /** Replace the contents of a view (invoked by the layout manager */
    @Override
    public RecyclerViewerAdapter.CommentViewHolder onCreateViewHolder(ViewGroup
viewGroup, int i) {
        View view;
        view =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.navigation_item,
viewGroup, false);
        CommentViewHolder commentViewHolder = new CommentViewHolder(view);
        return commentViewHolder;
    }
    @Override
    public void onBindViewHolder(CommentViewHolder Holder, int position) {
        final  CommentViewHolder commentViewHolder2 = Holder;
        DrawerItem item2 = mList.get(position);
        commentViewHolder2.textView.setText(item2.getTitle());
        commentViewHolder2.imageview.setImageResource(item2.getIcom());
    }
    @Override
    public int getItemCount() {
        return mList.size();
    }
}
```

code snippet 10: Adapter του RecyclerView του Navigation Drawer, RecycleViewerAdapter.java

Backend API

Εισαγωγή

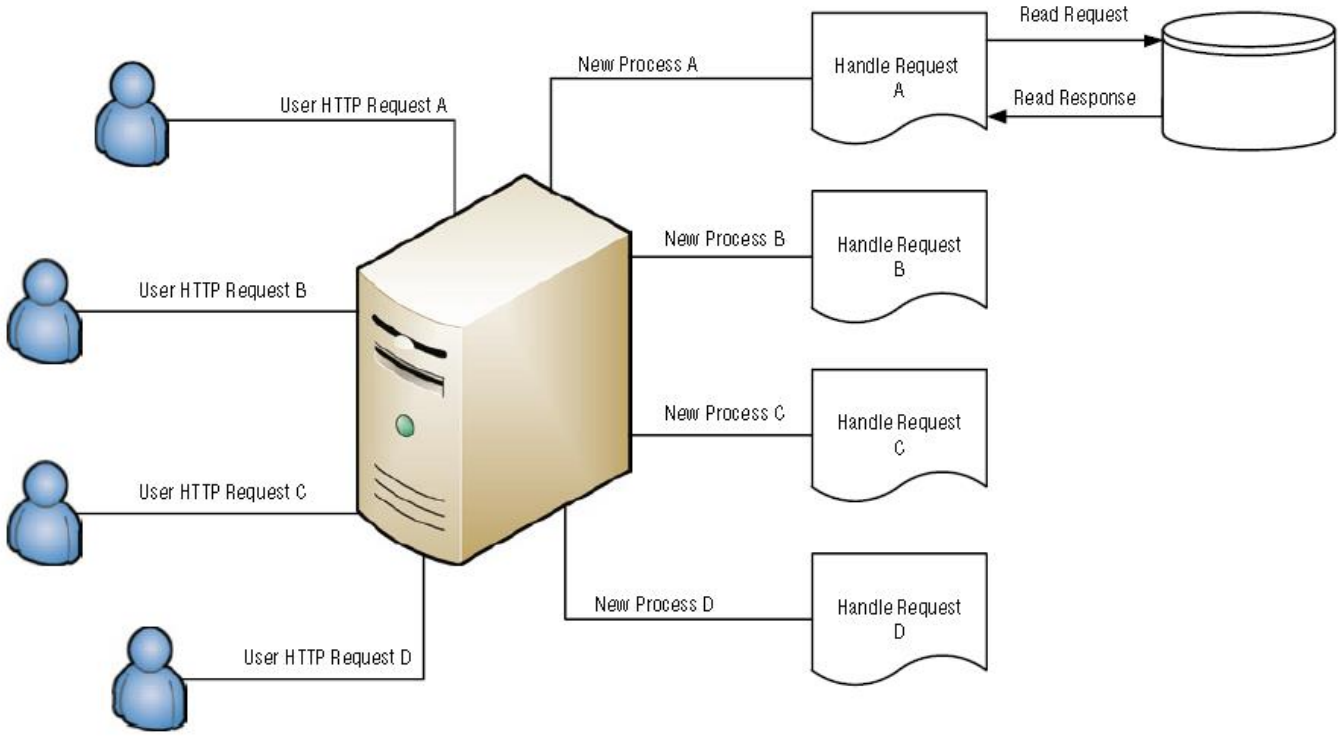
Για την δημιουργία του Back End χρησιμοποιήθηκε η μέθοδος MEAN STACK [21]. MEAN Stack είναι ακρωνύμιο και σημαίνει:

- M = MongoDB/Mongoose.js, μία Βάση δεδομένων noSQL.
- E = Express.js, ένα εργαλείο WEB φτιαγμένο σε node.js .
- A = Angular.js, στην περίπτωση του API που παρουσιάζεται στην παρούσα εργασία θα χρησιμοποιηθεί ένα android Application άρα συμβολικά για την εργασία θα μετατραπεί το ακρώνυμο σε ME(a)N Stack.
- N = Node.js, είναι ένας διερμηνευτής (interpreter), γραμμένος σε JavaScript , από την πλευρά του Server.

Ο λόγος που δημιουργήθηκε το Back End με την μέθοδο αυτή είναι ότι ο Server απαντά στον Client με JSON, άρα η επιλογή της MongoDB είναι η καλύτερη δυνατή διότι δεν αποθηκεύει τα δεδομένα σε πίνακες αλλά σε objects που έχουν την μορφή JSON και κάθε στοιχείο μπορεί να έχει διαφορετικά χαρακτηριστικά (Πεδία). Επίσης όλες οι τεχνολογίες είναι στην ίδια γλώσσα προγραμματισμού, σε JavaScript (express.js , node.js), παρακάτω εξηγούνται τα πλεονεκτήματα της τεχνολογίας node.js.

Πλεονεκτήματα Τεχνολογίας node.js

Οι παραδοσιακοί web servers χρησιμοποιούν μία διεργασία για κάθε αίτημά. Το να χρησιμοποιείται μία διεργασία για κάθε αίτημά είναι μία πολύ ακριβή διαδικασία [22] και σε όρους επεξεργαστή (CPU) και σε όρους μνήμης RAM. Ο λόγος που συμβαίνει αυτό μπορεί να αναλυθεί στην παρακάτω εικόνα.



Εικόνα 12: Παραδοσιακός web server που χρησιμοποιεί διεργασίες **[22]**

Στο παραδοσιακό web Server, για να απαντήσει ο web Server σε ένα αίτημα (π.χ. στο αίτημα A), χρειάζονται κάποια δεδομένα από την βάση , όπως φαίνεται και στην εικόνα του παραρτήματος. Αυτή η ανάγνωσή για να ολοκληρωθεί μπορεί να πάρει πολύ χρόνο. Για αυτή την διαδικασία θα απασχολείται ο επεξεργαστής και η μνήμη γιατί θα αναμένει (idle time) μία απάντηση από την βάση δεδομένων.

Πλέον οι web Server χρησιμοποιούν ένα νήμα από μία πισίνα νημάτων (thread pool). Όταν ένα αίτημα έρθει στον server , συνδυάζεται ένα νήμα για το εξυπηρετήσει. Αυτό το νήμα φυλάσσεται για το αίτημα σε όλη την διάρκεια που το αίτημα εξυπηρετείται. Αυτή η μέθοδος είναι πολύ καλύτερη διότι το νήμα είναι πιο ελαφρύ από μία διεργασία, παρόλα ταύτα και αυτή η μέθοδος εξυπηρέτησης αιτημάτων έχει τα μειονεκτήματά της , διότι σπαταλάτε μνήμη RAM ανάμεσα στα νήματα και επίσης το λειτουργικό χρειάζεται επεξεργαστική ισχύς για να εναλλάσει τα νήματα, έτσι σπαταλούνται και πόροι του επεξεργαστή. [22]

Από τα παραπάνω, γίνεται κατανοητό ότι και με τους δύο τρόπους σπαταλάμε πόρους του συστήματος. Ο τρόπος που λειτουργεί η τεχνολογία node.js είναι η μονονηματική διαχείριση των αιτημάτων. Αυτός ο τρόπος λέγεται Nginx. Με αυτόν τον τρόπο δεν υπάρχει καθόλου waiting time και μπορεί να εξυπηρετήσει ο server μεγαλύτερο αριθμό αιτημάτων. Σε κάθε αίτημα μία callback[17] εκτελείται αλλά αν δεν υπάρχει κάποια εργασία για να πραγματοποιηθεί η node.js κοιμάται [23]. Τελικά δεν υπάρχουν διεργασίες E/E (I/O), άρα δεν υπάρχει περίπτωση να μπλοκάρει η διαδικασία. Δηλαδή, ο τρόπος που λειτουργεί το node.js είναι ασύγχρονος. Τέλος, έχει ένα από τα μεγαλύτερα οικοσυστήματα πακέτων[18] που ονομάζεται npm. [23]

REST API

Η αρχιτεκτονική του Back End API βασίστηκε σε REST API (Representational State Transfer). REST API είναι μία τεχνολογία που μπορεί ο Server να επικοινωνεί με οποιαδήποτε συσκευή ή υπολογιστή, πχ android ή iPhone ή μέσω web services. Πλέων, σχεδόν όλα τα web services χρησιμοποιούν την τεχνική του REST API.

πηγή: [24]

Στην παραπάνω εικόνα απεικονίζεται ο Client(Android Phone, iOS Phone ή Website), ο οποίος στέλνει Request και Response στο Web Service. Το Web API είναι η “Web Service” που “ακούει” απευθείας στα αιτήματα(Request) και στις Ανταποκρίσεις (Response) του Client. Φυσικά το πρωτόκολλό που χρησιμοποιείται για την επικοινωνία είναι HTTP[19] πρωτόκολλο.
Ο Client στέλνει στον Server HTTP verbs, μαζί με ένα URL και παράμετρούς. Το URL περιγράφει το αντικείμενο σε μορφή (JSON) στο API που παρουσιάζεται στην παρούσα εργασία, έτσι ώστε ο Server να απαντήσει στον Client με ένα JSON[20].
HTTP verbs μπορούν να περιγράφουν κάποιες διαδικασίες επικοινωνίας που κάνει ο Client προς τον Server. Στον πίνακα [25] που ακολουθεί περιγράφονται σε γενικές γραμμές τα ρήματα του πρωτοκόλλου αυτού .

[17] Callback: Λέγεται ένα κομμάτι εκτελέσιμου κώδικα το οποίο περνιέται ως παράμετρο σε ένα άλλο κώδικα, ο οποίος αναμένεται να εκτελέσει αυτήν την παράμετρο (callback).
[18] Τα πακέτα περιέχουν πολλές βιβλιοθήκες
[19] HTTP: The hyper Text Transfer Protocol, είναι ένα πρωτόκολλο, message-based με το οποίο επικοινωνούν οι υπολογιστές μεταξύ τους [21]
[20] JSON: JavaScript Object Notation

POST	Create	Ο Client θέλει να εισάγει ή να δημιουργήσει ένα αντικείμενο
GET	Read	Ο Client θέλει να διαβάσει ένα αντικείμενο
PUT	Update	Ο Client θέλει να αναβαθμίσει ένα αντικείμενο
DELETE	Delete	Ο Client θέλει να διαγράψει ένα αντικείμενο

Μερικοί κωδικοί αποτελεσμάτων(http result codes) μπορούν να περιγράφουν στον παρακάτω πίνακα.

200	**OK**
201	Δημιουργήθηκε (χρησιμοποιείται με το Post)
400	Bad Request , κακό αίτημα , μπορεί να λύπουν κάποιοι παράμετροι
401	Unauthorized, λύπουν οι παράμετροι πιστοποίησης
403	Forbidden, έχει γίνει πιστοποίηση αλλά η σύνδεση στερείται δικαιωμάτων
404	Not Found

Web Service – Routing

Ο όρος routing (διαδρομή) αναφέρεται στο πώς ένα πρόγραμμα (Server) αποκρίνεται (responds) στο αίτημα (request) του πελάτη (Client), σε ένα συγκεκριμένο άκρο (endpoint), το οποίο είναι ένα URI[21] ή διαδρομή (Path), μαζί ένα συγκεκριμένο HTTP verb (GET, POST, PUT , DELETE).

Κάθε διαδρομή μπορεί να έχει μια ή παραπάνω συναρτήσεις χειρισμού (handler functions), οι οποίες εκτελούνται όταν υπάρχει ταίριασμα στο request του πελάτη με το respond του Server. Τα αιτήματα που μπορεί να κάνει ο Client στον HTTP Server περιγράφονται στον παρακάτω πίνακα:

[21] Uniform Resource Identifier, είναι ένα αλφαριθμητικό το οποίο χρησιμεύει για τον εντοπισμό ενός ονόματος ενός στοιχείου

Route(Διαδρομή)	Verb(Ρήμα HTTP)	Description(Περιγραφή)	Variables(JSON Body)	Δράση στην Βάση Δεδομένων
/users/register	POST	Εγγραφή νέου χρήστη	First Name, Last Name , email, username, password	Δημιουργία ενός αρχείου τύπου user με μοναδικό id
/users/resetpass	POST	Επαναφορά κωδικού και αποστολή στον χρήστη e-mail με το νέο κωδικό	Email, username	Αναζήτηση του αρχείου user και αλλαγή του password
/users/changepass	POST	Αλλαγή κωδικού πρόσβασης	userName,email, oldPassword,newPassword	Εύρεση του αρχείου User που ταιριάζει στο e-mail και αλλαγή του κωδικού
/user/login	POST	Σύνδεση στον λογαριασμό υπάρχοντος χρήστη	email, password	Αναζήτηση ενός αρχείου user με βάση το e-mail
/gpc/saveCoord	POST	Αποθήκευση της τοποθεσίας του χρήστη χρησιμοποιώντας το token του για ταυτοποίηση	token,latitude,longitude	Δημιουργία ενός νέου αρχείου τύπου Locations
/gpc/getCoord	POST	Επιστρέφει όλα τα location του χρήστη με το συγκεκριμένο token	token	Αναζήτηση στο collection locations του token
gpc/getCoordDate	POST	Επιστρέφει τα location του χρήστη με το συγκεκριμένο token την συγκεκριμένη ημέρα	Token, date	Αναζήτηση στο collection locations του token και της ημέρας.

Αρχιτεκτονική των Web Services

Το API του Back End αποτελείται από πολλά αρχεία. Για να δομηθεί χρησιμοποιήθηκε το εργαλείο express [26]. Αυτό το εργαλείο είναι ένα ελαφρύ node.js πρόγραμμα το οποίο παρέχει πολλές δυνατότητες για εφαρμογές web και mobile.

Μία από αυτές τις δυνατότητες είναι ότι δημιουργεί, όταν εγκατασταθεί την ραχοκοκαλιά του back End, δημιουργώντας την δομή των φακέλων. Αυτό μπορεί να πραγματοποιηθεί με λίγες εντολές στην κονσόλα του λειτουργικού συστήματος που θέλουμε να εγκαταστήσουμε το back end API. Για παράδειγμα με την εντολή

```
$ npm init
```

Δημιουργείται ένα αρχείο json που ονομάζεται package.json , το οποίο ουσιαστικά αντικατοπτρίζει τις ιδιότητες του server [27] .

Ύστερα δημιουργεί το κυρίως αρχείο του back end API, που στην περίπτωσή μας ονομάζεται index.js . Σε αυτό το αρχείο καλούνται οι απαραίτητες βιβλιοθήκες που έχουν αποθηκευτεί στον

φάκελο node_modules με την εντολή npm init. Γίνεται η σύνδεση με την βάση στην κατάλληλη πόρτα που είναι ορισμένη να ακούει και σηκώνεται ο server node js στην επιθυμητή πόρτα. Στο παρακάτω κώδικα φαίνεται το αρχείο index.js. Η μεταβλητή app είναι η μεταβλητή που χρησιμοποιείται για να αρχίσει να λειτουργεί ο server στην πόρτα 3000 και η μεταβλητή mongoose είναι η μεταβλητή που χρησιμοποιείται για να συνδεθεί ο server node.js με την βάση mongodb.

Οι βιβλιοθήκες που χρησιμοποιήθηκαν στο API αυτό έγιναν install με το πακέτο npm και έχουν εκχωρηθεί ως dependencies στο αρχείο package.json. όπου την δομή του μπορούμε να την δούμε στο παρακάτω κομμάτι κώδικα.

```
{
  "name": "api",
  "version": "2.0.0",
  "description": "api for dissertation",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "vasilis",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.13.3",
    "express": "^4.13.3",
    "mongoose": "^4.1.2",
    "debug": "~2.2.0",
    "connect": "~2.14.4",
    "nodemailer":"~0.6.3",
    "gravatar":"~1.0.6",
    "csprng" : "~0.1.1"
  }
}
```

code snippet 11 – Αρχείο package.json , ορισμός του application

Όπως φαίνεται παραπάνω χρησιμοποιούνται διάφορες βιβλιοθήκες όπως η body-parser ή η express. Αυτές οι βιβλιοθήκες έχουν αποθηκευτεί στον φάκελο node-modules και είναι διαθέσιμες από οποιοδήποτε αρχείο μέσα στον server.

Τέλος, για να εκτελεστεί άλλη η εφαρμογή και να τρέξει ο server χρειάζεται να εκτελεστεί η εντολή node index έτσι ώστε να γίνει compile ο κώδικας και να αρχίσει να λειτουργεί ο Server στην πόρτα 3000.

```
var express = require('express'); // using express framework
var mongoose = require('mongoose');
var app = express(); //getting an instance of the express framework
var connect = require('connect');
var express = require('express'); // using express framework
var app = express(); //getting an instance of the express framework
var bodyParser = require("body-parser"); // using body - parser for parsing
//using modules
app.use(bodyParser.json()); // parsing json
app.use(bodyParser.urlencoded({"extended" : false})); //This will send all of our data to the Node server as query strings

var router = require('./router')(app); // instance of the file router
//log modules
app.use(connect.logger('dev'));
app.use(connect.json());
app.use(connect.urlencoded());
//connect to the Database
mongoose.connect('mongodb://localhost:27017',function(err){
  if(err){
    console.log('connection error',err);
  }else{
    console.log('connection to the database successful');
  }
});

//assign the port in the Server - Starting the Server
var server = app.listen(3000, function() {
  console.log("listening to port 3000");
});
```

code snippet 12 : index.js

Δεύτερο ποιο σημαντικό αρχείο, είναι το αρχείο router.js το οποίο περιέχει όλα τα web services έτσι όπως περιεγράφηκαν στον πίνακα με τις διαδρομές. Εκεί ορίζονται οι μέθοδοι που θα εκτελεστούν σε περίπτωση που ένας client κάνει μία κλήση προς αυτά τα URI και το JSON body. Αυτό το αρχείο καλεί τα αρχεία Javascript τα οποία περιέχουν το κώδικα για να πραγματοποιηθεί η εκτέλεση των αιτημάτων των clients, να γίνει κάποια αλλαγή στην βάση δεδομένων εάν είναι απαραίτητο και τέλος να στείλουν την απάντηση σε μορφή json στους clients. Στο παρακάτω κομμάτι κώδικα φαίνεται ένα web Service μαζί με το body του και μία κλήση μεθόδου(code snippet 12) σε ένα αρχείο Javasctript, που εκτελεί τις επιθυμητές ενέργειες (code snippet 13).

```
var register = require('./node_modules/users/register');
```

Στην παραπάνω γραμμή βλέπουμε την ανάθεση του αρχείου register.js που βρίσκεται στην διαδρομή node_modules/users στην μεταβλητή register, αυτή η μεταβλητή θα χρησιμοποιηθεί στο παράδειγμα του (code snippet 12) για να καλέσει την μέθοδο register που βρίσκεται στο αρχείο register.js .

```
//*************************************************************************************
//Add new user in MongoDB - register
//*************************************************************************************
app.post('/users/register',function(req, res){

  var name = req.body.name;
  var surname = req.body.surname;
  var email = req.body.email;
  var userName = req.body.userName;
  var passWord = req.body.passWord;

  register.register(name, surname , email, userName, passWord, function (callback){
      console.log(callback);
      res.json(callback);
  });
});
//*************************************************************************************
```

code snippet 13 Παράδειγμα web Service , /users/register που καλεί την συνάρτηση register που βρίσκεται στο αρχείο register.js

Η συνάρτηση register στο αρχείο register.js χρησιμοποιεί τον τρόπο της εξαγωγής, export. Η πρόταση export χρησιμοποιείται για την εξαγωγή συναρτήσεων, αντικειμένων από κάποιο συγκεκριμένο αρχείο ή κομμάτι κώδικα (module) [28].

Επίσης όπως βλέπουμε και από το code snippet 12 το body που στέλνει ο client ενσωματώνεται μέσα στο request (req). Ο client θα πρέπει να έχει ορίσει ακριβώς τις ίδιες ονομασίες για τα κλειδιά του body (key values), ακριβώς με τις ονομασίες που έχουν οριστεί στο web service ειδάλλως το web service δεν θα μπορεί να τα δει.

Οι μεταβλητές του body εκχωρούνται σε τοπικές μεταβλητές που με την σειρά τους περνάνε με τον τρόπο export στην μέθοδο export της export.js. Τέλος, όπως φαίνεται και από το σχήμα, γίνεται επιστροφή μιας callback (res). Η callback είναι ένα JSON που μπορεί να περιέχει ένα μήνυμα ή διάφορα στοιχεία από την βάση. Πχ. Στην περίπτωση της register μεθόδου γίνεται επιστροφή ενός μηνύματος και του token του καινούργιου χρήστη. Στην παρακάτω εικόνα φαίνεται η απάντηση από τον server.

{

"response":"Successfully Registered",

"res":true,

"token":"0888cd99f03f1d32c5fa30b9c1f8a459c0aca2bc9c17b2a83ff732c18ed368f2c
a6a9073e449b92bcd55998cadcc89cb2dbfde5261752d5384871e4e6c39d7d7"

}

Εικόνα 13 - JSON απάντηση του Server ύστερα από ενέργεια register

Ο client μπορεί να χρησιμοποιήσει όπως θέλει την δεδομένη απάντηση, στην παρούσα android εφαρμογή γίνεται ο έλεγχος εάν το res είναι true και έτσι προχωρά ο χρήστης στην επόμενη οθόνη της εφαρμογής και εγγράφεται στην βάση δεδομένων της mongoDB.

Ο ορισμός της συνάρτησης register στο αρχείο register.js φαίνεται από το παρακάτω κομμάτι κώδικα (code snippet 13). Όπως φαίνεται από τον κώδικα γίνονται διάφοροι έλεγχοι στα στοιχεία που δίνει ο χρήστης, όπως το password και το email να ακολουθούν ένα μοτίβο κτλ.

Επίσης ορίζεται μία μεταβλητή mailOptions που περιέχει τις ιδιότητες του ηλεκτρονικού μηνύματος που θα σταλεί στον χρήστη που θα δώσει το mail του. Η μεταβλητή newUser

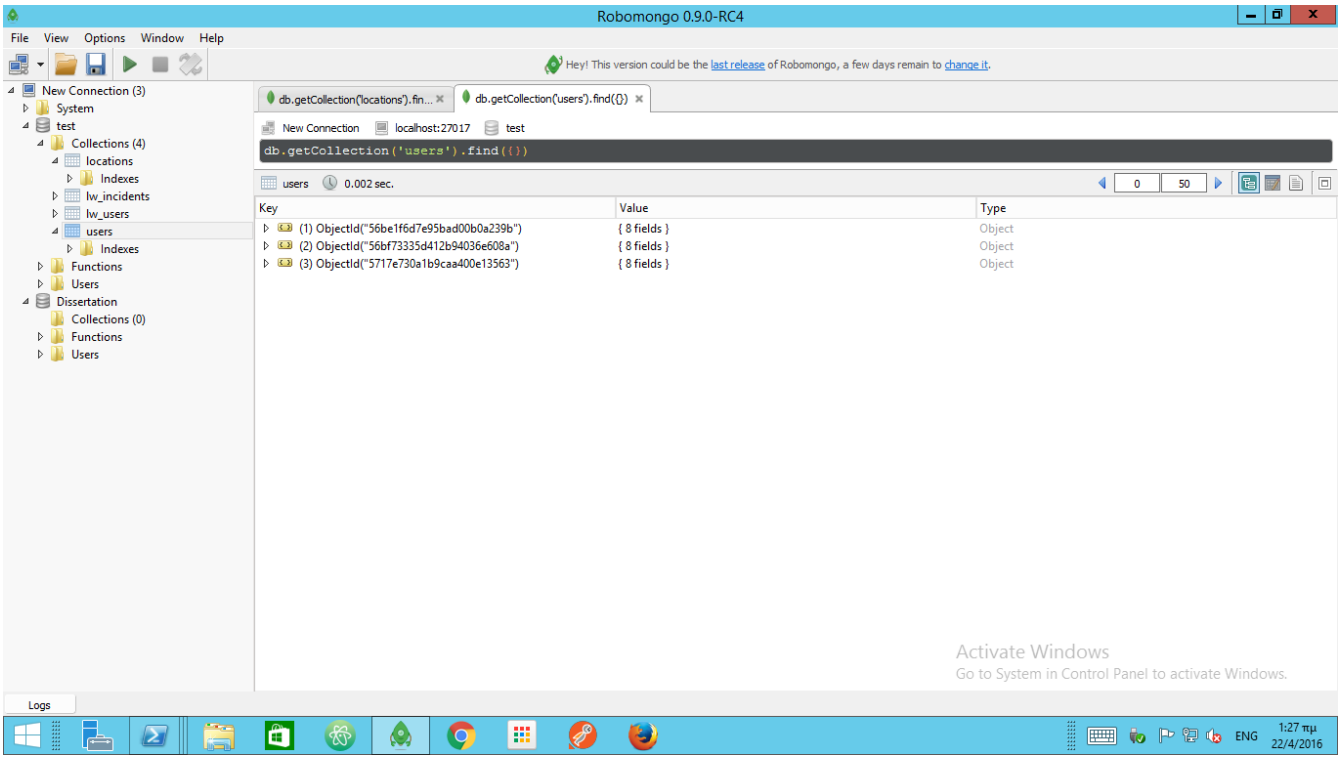
```
exports.register = function(name, surname , email, userName, password,callback) {

  //validations of the user's request for to register
  if (!(email.indexOf("@") == email.length)) {
    if (password.match(/([a-z].*[A-Z])|([A-Z].*[a-z])/) &&
        password.length > 4 && password.match(/[0-9]/) && password.match(/.[!,@,#,$,%,^,&,*,?,_,~]/)) {
          //creating more secure password and email
          var token = crypto.createHash('sha512').update(email).digest("hex"); //hashed token with email
          var hashed_password = crypto.createHash('sha512').update(password).digest("hex");

          var mailOptions = {
            from : "check in map Calendar <vasilisfouroulis@gmail.com>",
            to : email,
            subject: "Welcome to Map Calendar",
            text : "hello " + userName + " Welcome to the App. The Map Calendar Team :)"
          };

          //a new user
          var newUser = new mongoUser ({
              name : name,
              surname : surname,
              email : email,
              userName : userName,
              passWord : hashed_password,
              token : token
          });


          // find all users with email
          mongoUser.find({email:email}, function(err,users){
            //count the users
            var length = users.length;
            if(length == 0 ){
              //send the mail
              newUser.save(function(err) {
                smtpTransport.sendMail(mailOptions)
                //the user takes his token
                callback({'response':"Sucessfully Registered",
                          'res':true,
                          'token': token });
              });

            } else { //the user allready exists
                callback({'response':"Email already Registered",'res':false});
            }
          });
          //the user is not Registered

        } else {
          //validation of Password
          callback({'response':"Password Weak",'res':false});
        }
  } else {
    //validation of email
    callback({'response':"Email Not Valid",'res':false});
  }


}
```

code snippet 14 - register.js ορισμός της συνάρτησης register

αποθηκεύτε στην βάση δεδομένων. Αυτή η μεταβλητή μπορεί να είναι δυναμική και να περιέχει διαφορετικά πεδία κάθε φορά. Τέλος το αρχείο register.js γυρίζει μία callback στο αρχείο router που τελικά επιστρέφεται στον client.

Gloud Services

Για την αποθήκευση του Cloud Service χρησιμοποιήθηκε ο Ωκεανός (Okeanos). Ο Ωκεανός είναι η cloud υπηρεσία του GRNET. Χρησιμοποιήθηκε ένα VM Windows Server 2012 R2 server με σκληρό δίσκο 40g , έναν επεξεργαστή και 4g μέγεθος μνήμης. Επίσης δημιουργήθηκε Public IP (83.212.113.58), η οποία είναι static. Στην παρακάτω εικόνα που ακολουθεί φαίνεται ένα στιγμιότυπο μέσα από το VM και ποιο συγκεκριμένα φαίνεται ένας client της mongoDB με γραφική διεπαφή, ο οποίος ονομάζεται ROBOMONGO.



Βάση δεδομένων

Εισαγωγή

Η Βάση δεδομένων του συστήματος βασίστηκε στην τεχνολογία mongo DB. Η βάση mongo DB είναι μία βάση δεδομένων NoSQL. Η NoSQL βάσεις είναι πολύ χρήσιμες σε περιπτώσεις που η βάση δεδομένων δεν είναι δομημένη και πάρα πολύ μεγάλη και έχουν πολύ μεγαλύτερη αποδοτικότητα από τις σχεσιακές βάσεις. [29].

Τα δεδομένα στις τυπικές βάσεις δεδομένων SQL ανήκουν συνήθως με ένα schema. Αντικείμενα με τα ίδια χαρακτηριστικά ομαδοποιούνται μαζί κάνοντας τα δομημένα δεδομένα. Έτσι κάθε σειρά του πίνακα που απαρτίζουν, έχει ακριβώς το ίδιο αριθμό και τύπο δεδομένων (fields). Οι σχεσιακές βάσεις κανονικοποιούνται και αυτό έχει ως αποτέλεσμα να δημιουργούνται πολλοί πίνακες. Για να γίνουν ερωτήματα (queries) προς την βάση χρειάζεται να έρθουν (fetching) και να συνδυαστούν πολλά δεδομένα από πολλούς πίνακες. Πολλές φορές για να γίνει αυτό πρέπει να γίνουν πολύπλοκες διαδικασίες , ενώσεις πινάκων (join) ή να ταιριάξουν τα ξένα κλειδιά και τα πρωτεύοντα κλειδιά. Όλα αυτά κοστίζουν σε χρόνο και σε πόρους του συστήματος και όσο πιο μεγάλο το σχήμα της βάσης τόσο πιο πολύς ο χρόνος για να εκτελεστούν λειτουργίες.

Οι βάσεις NoSQL , όπως είναι και η βάση που χρησιμοποιείται στην παρούσα εργασία, μπορεί να αποθηκεύσει δεδομένα τα οποία είναι ημιδομημένα, δηλαδή τα αντικείμενα μπορούν να ομαδοποιηθούν αλλά έχουν διαφορετικά χαρακτηριστικά [29]. Τέτοια δεδομένα μπορούν να βρεθούν σε κινητά smartphones διότι χρησιμοποιούν τεχνολογίες τύπου JSON και XML που περιέχουν ημι-δομημένα δεδομένα.

Οι βάσεις NoSQL, οργανώνουν τα δεδομένα σε ζευγάρια τιμών-κλειδιών. Το κλειδί χρησιμοποιείται για να αναγνωρίζεται ένα μοναδικό δεδομένο και η τιμή μπορεί να είναι ένα απλό κείμενο ή αριθμός ή και περίπλοκος τύπος δεδομένων [29]. Δεν υπάρχει τυπική γλώσσα για ερωτήματα και γενικά οι λειτουργίες είναι πιο απλές και πιο προσιτές.

Πιο συγκεκριμένα στην MongoDB μία εγγραφή είναι και ένα αρχείο, το οποίο είναι μία δομή δεδομένων που αποτελείται από πεδία και ζευγάρια τιμών [25]. Τα αρχεία της MongoDB είναι παρόμοια με JSON αντικείμενα.

Η mongo DB αποθηκεύει τα αρχεία σε συλλογές (collections). Οι συλλογές είναι ανάλογες των πινάκων αλλά η συλλογή δεν υποχρεώνει τα αρχεία να έχουν το ίδιο σχήμα και όπως και σε όλες τις βάσεις NoSQL, κάθε αρχείο πρέπει να έχει ένα μοναδικό πεδίο (_id) [25].

Driver node.js και mongodb

Για να μπορεί να επικοινωνεί η βάση με το back – end , στην περίπτωση μας τον server node.js πρέπει να χρησιμοποιηθεί ένας driver. Στην παρούσα πτυχιακή χρησιμοποιήθηκε ο driver moongose. [30].

Η moongose είναι ένας driver που εισάγεται στην Node.js και παρέχει μία ευκολονόητη schema-based[22] λύση για την μοντελοποίηση των δεδομένων. Στον παρακάτω κομμάτι κώδικα μπορούμε να δούμε πως ορίζεται ο User ο οποίος κάθε φορά που καλείται και αποθηκεύεται αποτελεί μία σειρά στον πίνακα ή πιο σωστά ένα αρχείο στην noSQL βάση δεδομένων .

```
var mongoose = require('mongoose');

//creating a new user
var users = new mongoose.Schema({
  name: String,
  surname : String,
  email : String,
  userName : String,
  passWord : String,
  token : String
});

module.exports = mongoose.model('users', users);
```

code snippet 15 - Ορισμός ενός user και αποθηκευσή του στον πίνακα users

[22] Σχέδιο που καταγράφεται και ορίζεται

Συμπεράσματα και μελλοντικές επεκτάσεις

Συμπεράσματα

Μέσα από την υλοποίηση του front – End και του back – End και της θεωρητικής προσέγγισης το συμπέρασμα που μπορεί να εξαχθεί είναι ότι το συγκεκριμένο stack που χρησιμοποιήθηκε, δηλαδή το mean – stack [23] είναι μία πολύ καλή προσέγγιση για τον σχεδιασμό, την υλοποίηση, την συντήρηση και την αποθήκευση του κώδικα στο cloud (Okeanos).

Η Javascript σαν γλώσσα προσφέρει μεγάλη ευελιξία και μέσα σε λιγοστό χρόνο μπορεί να δημιουργηθεί ένα αξιόλογο back – End για οποιαδήποτε εφαρμογή. Επίσης η mongoDB και η node.js χρησιμοποιούν την ίδια γλώσσα, Javascript. Σε άλλη περίπτωση θα έπρεπε να χρησιμοποιηθεί και η γλώσσα SQL για την αποθήκευση – αλλαγή – διαγραφή των δεδομένων στην βάση δεδομένων.

Τέλος με το nodejs γίνεται εύκολη η υλοποίηση των web services. Οπότε η nodejs είναι πολύ καλή προσέγγιση για την υλοποίηση ενός back – End για εφαρμογές mobile. Ο λόγος που ισχύει αυτό είναι ότι οι mobile εφαρμογές δεν μπορούν να μιλήσουν απευθείας στην βάση δεδομένων λόγο ασφάλειας, έτσι έχουν ανάγκη τα web services. Επίσης για κάποιον προγραμματιστή που γνωρίζει android, δηλαδή java, είναι ποιο εύκολο να κατανοήσει την γλώσσα προγραμματισμού JavaScript παρά άλλες κλασικές γλώσσες προγραμματισμού για back – end όπως την PHP.

Όσον αφορά την πλευρά του cloud, η node.js τεχνολογία είναι ποιο αποδοτική, διότι είναι αυτόνομη, δεν έχει ανάγκη από extra server όπως η php (Apache) ή (Tomcat) που έχει η Java. Αυτό διότι στην παρούσα εργασίας χρησιμοποιήθηκε ένα VM του Okeanos με ελάχιστα χαρακτηριστικά, όπως ένα CPU core και 4gb ram. Το ίδιο ισχύει και για την mongoDB, η οποία είναι πιο ελαφριά από την MySQL ή την OracleSQL κ.τ.λ. .

Όσον αφορά το Front- End η χρησιμοποίηση των google play services (Fused Location Provider) για την καταγραφή της τοποθεσίας, βοηθά το κινητό του χρήστη για την εξοικονόμηση ενέργειας και μπαταρίας.

Περιορισμοί

Στην παρούσα εργασία σκοπός ήταν να φτιαχτεί μία εφαρμογή η οποία να αποθηκεύει την τοποθεσία του χρήστη. Βέβαια, όταν ο χρήστης τερματίσει την εφαρμογή το service που αποθηκεύει στον server την τοποθεσία του χρήστη τερματίζεται. Αυτό, διότι δεν είναι σκοπός της εφαρμογής η αθέμιτη καταγραφή της τοποθεσίας. Ο χρήστης πρέπει να το γνωρίζει ότι γίνεται αυτή η ενέργεια στο background. Έτσι, η εφαρμογή λειτουργεί μόνο όταν βρίσκεται στο foreground ή στο background. Επίσης, δεν δίδεται στον χρήστη η δυνατότητα να διαγράψει μία εγγραφή. Αυτή την δυνατότητα την έχει μόνο ο διαχειριστής της βάσης.

Αυτή η εφαρμογή χρησιμοποιεί κώδικα έξω από το API του android για την καταγραφή της τοποθεσίας. Χρησιμοποιεί το API των google play services και ποιο συγκεκριμένα τον τρόπο του Fused Location Provider. Αυτό από την μία είναι θετικό διότι είναι πολύ καλύτερο API από αυτό του Location manager, σε ορισμούς μπαταρίας και εξοικονόμησης ενέργειας αλλά σε περίπτωση που ο χρήστης δεν έχει google play services στο κινητό του, τότε δεν θα μπορέσει να

[23] http://mean.io/#!/

λειτουργήσει ποτέ η εφαρμογή. Το ίδιο ισχύει και για τους χάρτες της google που χρησιμοποιεί η εφαρμογή αυτή. Άμα ο χρήστης δεν έχει ενημερωμένη την εφαρμογή των google maps δεν θα μπορέσει να δει τον χάρτη και στην παρούσα εφαρμογή. Οπότε είναι πολύ σημαντικό ο χρήστης που θα χρησιμοποιήσει την εφαρμογή αυτή να έχει ενημερωμένα αυτά που αναφέρθηκαν παραπάνω. Βέβαια το ποιο σημαντικό απ' όλα είναι ότι η εφαρμογή αυτή προορίζεται μόνο για λογισμικά με API 17 (Jelly bean) και πάνω.

Τέλος, το ποιο σημαντικό απ' όλα, εάν το λειτουργικό αισθανθεί πίεση σε θέμα hardware , δηλαδή ram , μπορεί να τερματίσει από μόνο του το background Service και να σταματήσει η διαδικασία της καταγραφής τοποθεσιών.

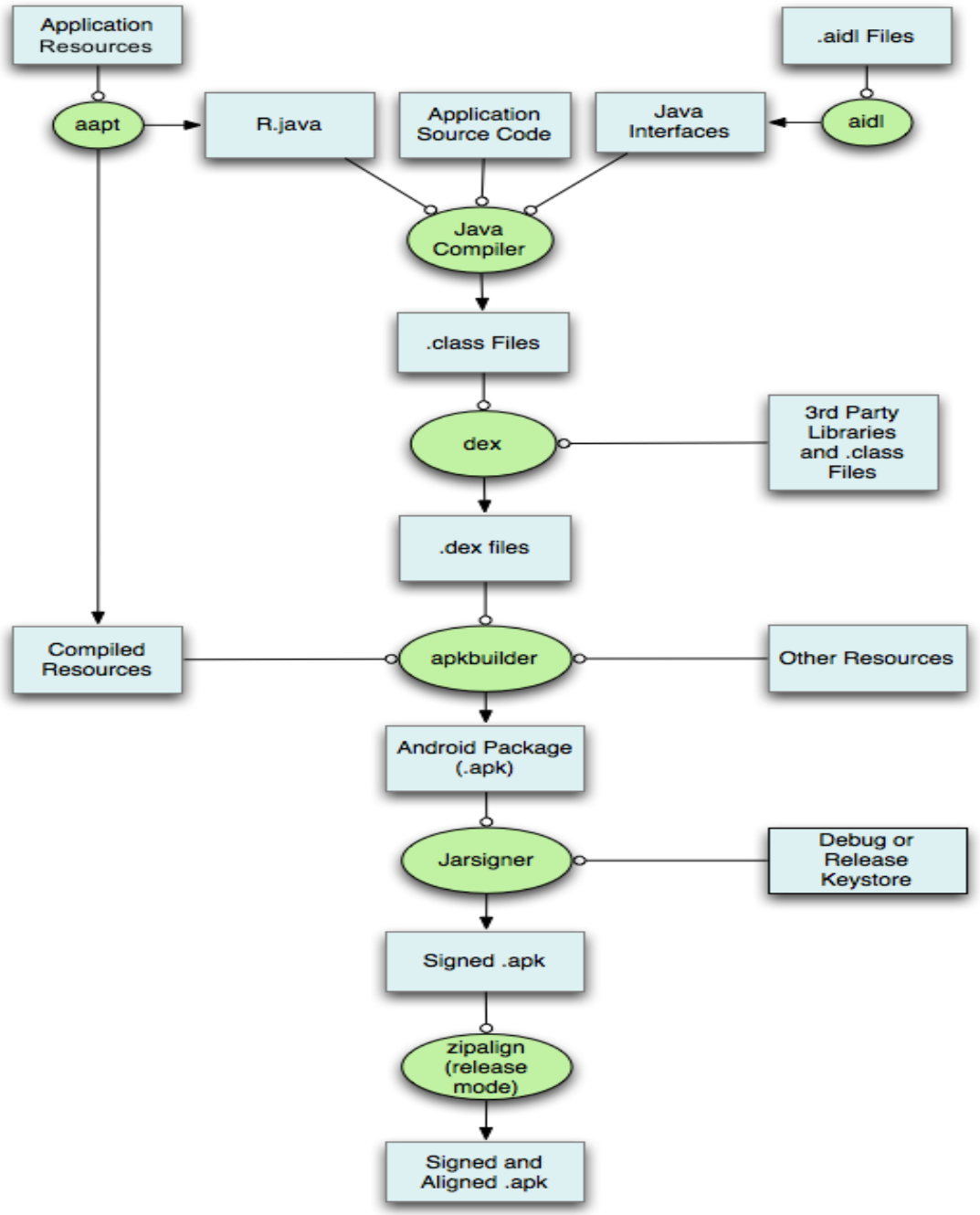
Μελλοντικές Επεκτάσεις

Όπως αναφέρθηκε και στους περιορισμούς, η καταγραφή της τοποθεσίας μπορεί να γίνει μόνο όταν η εφαρμογή είναι στο background ή στο foreground. Για να μπορέσει να λειτουργήσει ή εφαρμογή υπό την συνθήκη που η εφαρμογή είναι κλειστή, θα μπορούσε να χρησιμοποιηθεί ένα service START_STICKY . Το πρόβλημα μπορεί να αποφευχθεί έτσι αλλά δημιουργείται ένα άλλο, ότι όταν κλείνει η εφαρμογή που δημιούργησε το service, τότε το service κάνει restart, με αποτέλεσμα το χρονόμετρο που χρησιμοποιείται στην εφαρμογή μας να κάνει και αυτό restart. Μία πολύ καλή προσέγγιση θα ήταν να χρησιμοποιηθεί ένας Alarm Manager. Ο Alarm Manager είναι ένα service του συστήματος, που χρονοπρογραμματίζεται και ακούει την CPU του συστήματος για να μπορεί να κάνει την αντίστροφη μέτρηση, άρα είναι αρκετά αξιόπιστο. Αυτός ο Alarm Manager μπορεί να ξυπνήσει ένα Broadcast Receiver που με την σειρά του θα ξυπνήσει ένα service που θα καταγράψει την τοποθεσία. Η διαδιακασία αυτή δίνει αξιοπιστία και θα μπορούσε η εφαρμογή αυτή να χρησιμοποιηθεί ως children check ή για λόγους ασφαλείας για οποιονδήποτε. Βέβαια, θα πρέπει να χρησιμοποιηθεί με προσοχή, διότι ο Alarm Manager θέλει διαφορετικές μεθόδους για λειτουργικά Lollipop και πάνω.

Παράρτημα

Εικόνες

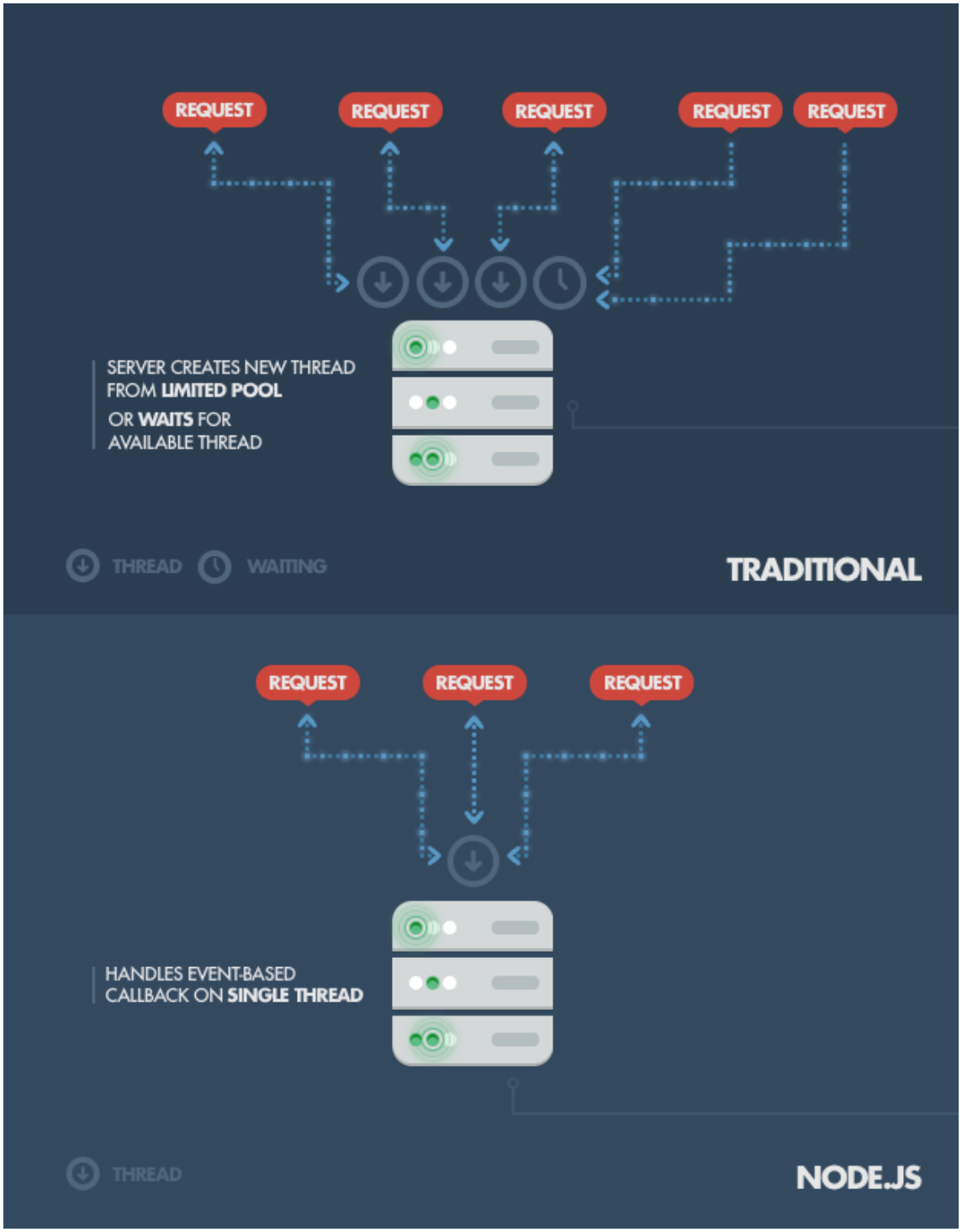
Grandle Build System



Εικόνα 2 - Grandle Build System **[31]**

Τρόποι εξυπηρέτησης αιτημάτων

Διαφορές Παραδοσιακού τρόπου εξυπηρέτησης αιτημάτων ανάμεσα σε παραδοσιακή τεχνολογία και node.js



Εικόνα 14: Παραδοσιακός τρόπος εξυπηρέτησης αιτημάτων με νήματα σε σχέση με τον τρόπο της τεχνολογίας node.js

Βιβλιογραφία

[1] Google, «https://www.google.com/maps/,» Google, 2016. [Ηλεκτρονικό]. Available: https://www.google.com/maps/timeline.

[2] Google, «takeout.google.com,» Google, 2016. [Ηλεκτρονικό]. Available: https://takeout.google.com/settings/takeout/custom/location_history?hl=en&gl=GR&expflags.

[3] F. Labs. [Ηλεκτρονικό]. Available: https://www.swarmapp.com/.

[4] D. Clark, Beginning C# Object-Oriented Programming, 2013.

[5] R. S. Wazlawick, Object-Oriented Analysis and Design for Information Systems , Modeling with UML, OCL and IFML, Elsevier Inc, 2014.

[6] J. Y. Lee and J. K. Lee , "Android Programming Techniques for Improving Performance," in *Awareness Science and Technology (iCAST) 2011 3rd International Conference*, Abuja, Nigeria, 2011.

[7] M. Gargenta και M. Nakamura, Learning Android second Edition, O'Reilly Media, 2014.

[8] Wei Hu, Dan Han, Abram Hindle and Kenny Wong, "The Build Dependency Perspective of Android's Concrete Architecture," in *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference*, June 2012.

[9] D. Gavalas και D. Economou , «Development Platforms for Mobile applications: Status and Trends,» *IEEE SOFTAWARE,* τόμ. 28, αρ. 1, pp. 77 - 86 , January/February 2011.

[10] B. Phillips, C. Stewart, B. Hardy και K. Marsicano, Android Programming THE BIG NERD RANCH GUIDE 2ND EDITION, Big Nerd Ranch, 2015.

[11] D. Android, «Transmitting Network Data Using Volley,» Google, 2016. [Ηλεκτρονικό]. Available: http://developer.android.com/training/volley/index.html.

[12] Google, «Google Developers,» [Ηλεκτρονικό]. Available: https://developers.google.com/android/reference/com/google/android/gms/maps/package-summary#interfaces.

[13] Google, «Developer Android,» [Ηλεκτρονικό]. Available: http://developer.android.com/guide/components/services.html.

[14] Google, «http://developer.android.com/,» 2016. [Ηλεκτρονικό]. Available: http://developer.android.com/guide/components/intents-filters.html.

[15] Google, «http://developer.android.com/,» 2016. [Ηλεκτρονικό]. Available: http://developer.android.com/reference/java/util/concurrent/ScheduledExecutorService.html.

[16] Google, «http://developer.android.com/,» [Ηλεκτρονικό]. Available: http://developer.android.com/training/basics/data-storage/shared-preferences.html#GetSharedPreferences.

[17] Google, «http://developer.android.com,» [Ηλεκτρονικό]. Available: http://developer.android.com/reference/android/widget/CalendarView.html.

[18] http://developer.android.com/, Google, 2016. [Ηλεκτρονικό]. Available: http://developer.android.com/training/implementing-navigation/nav-drawer.html.

[19] Google, «developer android,» [Ηλεκτρονικό]. Available: http://developer.android.com/training/material/lists-cards.html.

[20] W. Savitch, Java: An Introduction to Problem Solving and Programming - 7th Edition, Pearson, 2012.

[21] A. Poulter, S. Johnston και S. Cox, «Using the MEAN Stack to implement a RESTful service for an Internet of Things Application,» 2015.

[22] B. A. Syed, Beginning Node.js, Apress, 2013.

[23] Node.js, "Node," 2015. [Online]. Available: https://nodejs.org. [Accessed December 2015].

[24] M. Masse, REST API Design Rulebook, O'Reilly, 2012.

[25] mongodb, «mongodb,» 2015. [Ηλεκτρονικό]. Available: https://www.mongodb.com. [Πρόσβαση December 2015].

[26] N. Foundation, «express,» [Ηλεκτρονικό]. Available: http://expressjs.com/.

[27] npm, «npmjs,» [Ηλεκτρονικό]. Available: https://docs.npmjs.com/files/package.json.

[28] MDN, «developer.mozilla.org,» [Ηλεκτρονικό]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/export.

[29] P. Zachary, S. Vrbsky and S. Poe, "Comparing NoSQL MongoDB to an SQL DB," p. 6, 2013.

[30] «mongoosejs,» [Ηλεκτρονικό]. Available: http://mongoosejs.com/.

[31] developer.Android, "Android Developers," Google, 2016. [Online]. Available: http://developer.android.com/sdk/installing/studio-build.html.

[32] R. Sedgewick, Αλγόριθμοι σε JAVA - Τρίτη αμερικάνικη έκδοση, Κλειδαριθμος, 2009.

[33] Goolge, «http://developer.android.com/,» [Ηλεκτρονικό]. Available: http://developer.android.com/reference/java/util/Calendar.html.

[34] A. Denis, B. H. Wixom και D. Tegarden, Ανάλυση & Σχεδιασμός Συστημάτων με την UML 2.0 , Μια αντικειμενοστρεφής προσέγγιση, 2009.

[35] Oracle, «docs.oracle.com,» Oracle, 2016. [Ηλεκτρονικό]. Available: https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html.

[36] Google, «http://developer.android.com/,» [Ηλεκτρονικό]. Available: http://developer.android.com/reference/android/support/v7/widget/RecyclerView.ViewHolder.html.