ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία εφαρμογής για την αποτύπωση των κεραυνών στο χώρο της Ελλάδας σε πραγματικό χρόνο

Μεντζελόπουλος Παναγιώτης

Επιβλέποντες Καθηγητές:

Β. Δαλάκας, Π. Κατσαφάδος, Μ. Νικολαϊδου

ΑΘΗΝΑ

ΙΟΎΛΙΟΣ 2016

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία εφαρμογής για την αποτύπωση των κεραυνών στο χώρο της Ελλάδας σε πραγματικό χρόνο

Μεντζελόπουλος Παναγιώτης

AM:it21119

Επιβλέποντες Καθηγητές:

Β. Δαλάκας, Π. Κατσαφάδος, Μ. Νικολαϊδου

Περίληψη

Την σημερινή εποχή η ενημέρωση προέρχεται από μια πληθώρα επιλογών μεταξύ των οποίων και το διαδίκτυο. Με την διάδοση των έξυπνων κινητών τηλεφώνων, που διαθέτουν τις κατάλληλες τεχνολογίες για πρόσβαση στο διαδίκτυο, δημιουργήθηκαν εφαρμογές που βασίζονται σε αυτό το χαρακτηριστικό για την εύκολη πρόσβαση του χρήστη σε κάθε είδους πληροφορία.

Στα πλαίσια της συγκεκριμένης εργασίας, αναπτύχθηκε μία υβριδική εφαρμογή που έχει ως σκοπό την απεικόνιση της δραστηριότητας των κεραυνών στην επικράτεια της Ελλάδας σε πραγματικό χρόνο. Η εργασία μας επικεντρώθηκε στην υλοποίηση για συσκευές με λειτουργικό Android και iOS αλλά μπορεί με κάποιες τροποποιήσεις να χρησιμοποιηθεί και σε άλλα δημοφιλή λειτουργικά συστήματα. Για την πραγματοποίηση της παραπάνω εφαρμογής, δημιουργήθηκε παράλληλα μία RESTful υπηρεσία API για την εύκολη πρόσβαση της εφαρμογής στα δεδομένα της βάσης. Τα δεδομένα των κεραυνών προέρχονται από ένα σύστημα ανίχνευσης των θέσεων των κεραυνών, εγκατεστημένο στον χώρο του Χαροκόπειου Πανεπιστημίου, το οποίο ανιχνεύει και καταγράφει κεραυνούς σε ακτίνα περίπου 500 χιλιομέτρων.

Με την υλοποίηση της συγκεκριμένης εφαρμογής, δίνεται η δυνατότητα στους χρήστες να ενημερωθούν σε πραγματικό χρόνο για την κεραυνική δραστηριότητα σε περιοχές της επιλογής τους ή της τρέχουσας τοποθεσίας τους, η οποία προσδιορίζεται με χρήση της τεχνολογίας GPS (Global Positioning System) της συσκευής τους. Με τη χρήση του αλγόριθμου DBScan για την ομαδοποίηση των κεραυνών, παρουσιάζονται στον χρήστη οι περιοχές με τους περισσότερους κεραυνούς τις τελευταίες 24 ώρες στον ευρύτερο χώρο της Ελλάδας. Ακόμη, θα μπορούσε να θεωρηθεί εργαλείο αυτοπροστασίας, αφού με την έγκαιρη ενημέρωση του χρήστη για επικίνδυνα καιρικά φαινόμενα θα έχει το χρόνο να λάβει τα απαραίτητα μέτρα για την προστασία του.

Abstract

Nowadays information comes from a variety of options, including the Internet. Over the last years, smartphones, which have the proper technology to access the internet, have been spread widely. That led to the creation of many smartphone applications based on this feature, providing the user easy access to information.

The application developed in the context of the current thesis is a hybrid application that is designed to illustrate the activity of lightning in the territory of Greece in real time. Our work focused on implementation for Android and iOS devices, but can, with certain modifications, be used in other operating systems. As part of the implementation of this application, there was an API RESTful service, developed alongside to provide easy access to the database. Thunderbolts data are provided by a thunderbolts' position detection system installed on site of Harokopio University of Athens, which detects and records thunderbolts in radius of approximately 500 kilometers.

With the implementation of the application, users are allowed to be informed in real time lightning activity in areas of their choice or their current location. To determine the location of the user, GPS technology (Global Positioning System) of the device is being used. Furthermore, the application gives the opportunity to the user to be informed about the sites with the most thunderbolts in the last 24 hours in the broader area of Greece. The clustering of the thunderbolts data, to determine these sites, is being processed with the use of the DBScan algorithm. Moreover, it could be considered a self-defense tool, since the user can be informed in time for severe thunderbolt activity close by and will have the time to take measures for his safety.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Τα κινητά τηλέφωνα αποτελούν αναπόσπαστο πλέον κομμάτι της καθημερινότητας για τους περισσότερους ανθρώπους, έτσι έχουν εξελιχτεί από απλές συσκευές επικοινωνίας σε έξυπνα τηλέφωνα ή αλλιώς smartphones [1], δηλαδή συσκευές με λειτουργικό σύστημα παρόμοιο με των υπολογιστών. Τα πιο δημοφιλή λειτουργικά συστήματα έξυπνων συσκευών αυτή τη στιγμή είναι χωρίς αμφιβολία τα Android και iOS, τα οποία για το 2015 κατείχαν το 96,7% της παγκόσμιας αγοράς [2]. Τα smartphones προσφέρουν πληθώρα λειτουργιών, όπως μεταξύ άλλων, GPS και πλοήγηση στο διαδίκτυο. Οι τεχνολογίες αυτές επιτρέπουν την δημιουργία εφαρμογών που στηρίζονται σε γεωχωρικά δεδομένα και στη πρόσβαση στο διαδίκτυο για παροχή πληροφοριών σε πραγματικό χρόνο.

Πριν μερικά χρόνια, η υλοποίηση εφαρμογών ξεχωριστά για κάθε λειτουργικό σύστημα κινητών συσκευών (native mobile apps) ήταν μονόδρομος και πολλές φορές μάλιστα περιοριζόταν σε ένα μοναδικό, λόγω του απαιτούμενου κόστους και χρόνου. Πλέον όμως με τη δυνατότητα υλοποίησης της εφαρμογής ως υβριδικής (hybrid mobile app), παρέχεται η δυνατότητα στους προγραμματιστές να υλοποιούν την εφαρμογή για τα πιο δημοφιλή λειτουργικά συστήματα κινητών συσκευών ταυτόχρονα. Αν και δεν υπάρχουν έρευνες ή στατιστικά για το ποσοστό των υβριδικών εφαρμογών στην αγορά, το ενδιαφέρον και η ζήτηση για τα απαραίτητα frameworks για την υλοποίησή τους συνεχώς αυξάνονται. Το 2013 η Gartner Inc. έκανε την πρόβλεψη ότι μέχρι το 2016 περισσότερες από το 50% των εφαρμογών θα είναι υβριδικές [3].

Ανεξαρτήτως του τρόπου κατασκευής των εφαρμογών και του λειτουργικού συστήματος της εκάστοτε συσκευής, είναι γεγονός ότι η ζήτηση εφαρμογών για smartphones μεγαλώνει. Είναι προφανές άλλωστε από την πληθώρα των ήδη διαθέσιμων εφαρμογών σε πολλούς τομείς της σύγχρονης καθημερινότητας όπως είναι τα μέσα μεταφοράς, η ενημέρωση, η διασκέδαση και η αποτύπωση των διαφόρων φυσικών φαινομένων.

Συγκεκριμένα στον τομέα του καιρού, υπάρχουν παγκοσμίως εκατοντάδες εφαρμογές πρόγνωσης και παροχής πληροφοριών σε πραγματικό χρόνο. Ενδεικτικό είναι το γεγονός, ότι τόσο στο Google Play όσο και στο Apple Store, υπάρχει ειδική κατηγορία ‘Καιρός’ με πολλές από τις εφαρμογές να μετρούν εγκαταστάσεις σε εκατομμύρια. Ωστόσο αναζητώντας εφαρμογές με συγκεκριμένες ιδιότητες όπως η γραφική απεικόνιση κεραυνών σε πραγματικό χρόνο, που

μας απασχολεί στην συγκεκριμένη εργασία, οι επιλογές είναι πολύ περιορισμένες. Έπειτα από εξαντλητική αναζήτηση, για το λογισμικό iOS και Android βρέθηκαν ελάχιστες εξειδικευμένες εφαρμογές απεικόνισης κεραυνών, οι οποίες χρησιμοποιούν ως πηγή δεδομένων το δίκτυο blitzortung[1]. Συγκεκριμένα για το λογισμικό android βρέθηκε μία εφαρμογή. Ονομάζεται 'Blitzortung Lightning Monitor' [4] και διανέμεται δωρεάν. Εκμεταλλεύεται τα διαθέσιμα δεδομένα του δικτύου blitzortung και παρουσιάζει σε παγκόσμιο βελησεκές τους κεραυνούς. Ακόμη διαθέτει λειτουργία GPS και υπηρεσία ειδοποίησης σε περίπτωση καταιγίδας σε κοντινή απόσταση. Αντίστοιχες εφαρμογές για συσκευές με λειτουργικό συστημα iOS είναι η 'BlitzortungLive'[2] και η 'Live Lightning'[3].

Έτσι, στόχος της παρούσας πτυχιακής εργασίας είναι η δημιουργία μίας υβριδικής εφαρμογής για έξυπνες κινητές συσκευές, που θα απεικονίζει τους κεραυνούς στο χώρο της Ελλάδας τις τελευταίες ώρες σε πραγματικό χρόνο και θα καλύπτει το μεγαλύτερο ποσοστό χρηστών smartphone. Σκοπός είναι η πληροφόρηση του χρήστη ανά πάσα στιγμή για την δραστηριότητα των κεραυνών στην περιοχή που βρίσκεται ή τις περιοχές τον ενδιαφέρουν. Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν τα frameworks Apache Cordova, Ionic και CodeIgniter.

Πιο συγκεκριμένα η εφαρμογή χρησιμοποιεί τη τεχνολογία GPS για το καθορισμό της τοποθεσίας του χρήστη και παρουσιάζει τους κεραυνούς στην ευρύτερη περιοχή του τις τελευταίες ώρες. Επίσης υπάρχει επιλογή αποθήκευσης μέχρι τριών περιοχών της προτίμησής του. Να σημειώσουμε ότι οι κεραυνοί εμφανίζονται στον χάρτη με διαφορετικές αποχρώσεις ανάλογα με την χρονική στιγμή που δημιουργήθηκαν και ο χρήστης έχει τη δυνατότητα να επιλέξει για πόσες ώρες από την τρέχουσα στιγμή, επιθυμεί να προβάλλονται στον χάρτη κεραυνοί. Τέλος, ο χρήστης μπορεί να πληροφορηθεί ονομαστικά αλλά και οπτικά με τη βοήθεια χάρτη για τις περιοχές με τους περισσότερους κεραυνούς το τελευταίο 24ωρο. Παράλληλα για την επίτευξη των παραπάνω, υλοποιήθηκε μία RESTful API υπηρεσία για την μεταφορά των δεδομένων από τη βάση στην υβριδική εφαρμογή καθώς και για την ομαδοποίηση των δεδομένων.

---

[1] Blitzortung. [23 Ιουνίου 2016.] http://www.blitzortung.org

[2] BlitzortungLive. [23 Ιουνίου 2016.] https://itunes.apple.com/us/app/blitzortunglive/id933558423?mt=8.

[3] Live Lightning. [23 Ιουνίου 2016.] https://itunes.apple.com/us/app/live-lightning/id541468577?mt=8.

Οι κεραυνοί ή αλλιώς ‘ηλεκτρικές εκκενώσεις’ που παρουσιάζονται στην παρούσα εργασία, ανιχνεύονται και καταγράφονται από ένα σύστημα ανίχνευσης της θέσης των κεραυνών, εγκατεστημένο στο χώρο του Χαροκόπειου Πανεπιστημίου. Το σύστημα boltek καταγράφει τις πληροφορίες σε αρχείο binary το οποίο μέσω λογισμικού σε γλώσσα C μετατρέπεται σε ASCII, τα οποία σαφώς είναι ευκολότερα στην επεξεργασία και αναγνώσιμα από τον άνθρωπο.

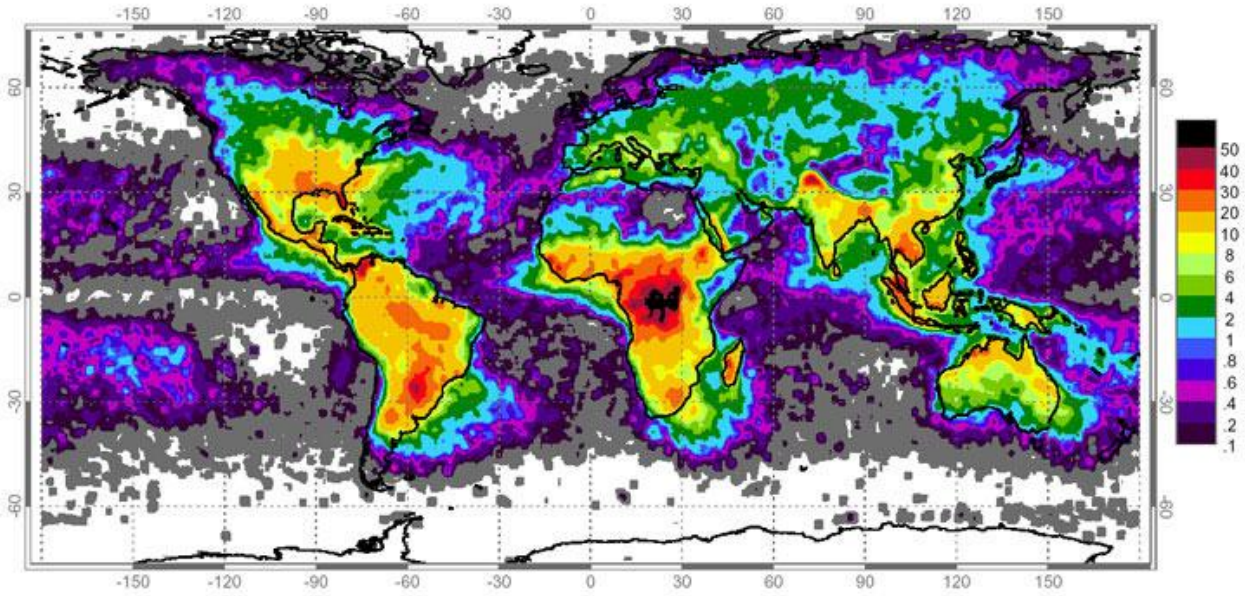
Κεφάλαιο 2

Θεωρητικό υπόβαθρο

2.1 Η δημιουργία ηλεκτρικών εκκενώσεων

Η ατμόσφαιρας της γης, έχει σαν αποτέλεσμα τη δημιουργία ενός μόνιμου ηλεκτρικού πεδίου το οποίο εκτείνεται έως και την ιονόσφαιρα. Το σύνολο των ατμοσφαιρικών στρωμάτων και ιδιαίτερα η ιονόσφαιρα, όπου παρατηρείται αυτή η διαφορά δυναμικού, ονομάζεται ηλεκτρόσφαιρα. Ο λόγος ύπαρξης αυτής της διαφοράς δυναμικού, οφείλεται σε δύο γεγονότα. Αρχικά, στον ιονισμό των ανώτερων στοιβάδων της ατμόσφαιρας λόγω των κοσμικών σωματιδίων και στην ύπαρξη φορτισμένων ιόντων λόγω φυσικών και χημικών διεργασιών στα κατώτερα στρώματα της ατμόσφαιρας. Το ηλεκτρικό πεδίο που περιγράψαμε παραπάνω, σε συνθήκες καλοκαιρίας αποτελείται από αρνητικό φορτίο κοντά στη γη και θετικό ψηλότερα στην ατμόσφαιρα.

Για να δημιουργηθεί λοιπόν ένα ηλεκτρικό φαινόμενο απαιτείται η διαφορά δυναμικού να ξεπεράσει την διηλεκτρική σταθερά της ατμόσφαιρας, η οποία ισούται με $3 \cdot 10^6$V/m. Σε συνθήκες καλοκαιρίας, η κάθετη διαφορά δυναμικού κυμαίνεται ανάμεσα σε 80-160V/m, η οποία δεν επιτρέπει τη δημιουργία ηλεκτρικών εκκενώσεων στα κατώτερα στρώματα της ατμόσφαιρας [5]. Για να ξεπεραστεί λοιπόν η τιμή της διηλεκτρικής σταθεράς από τη διαφορά δυναμικού, απαιτείται η ύπαρξη έντονων ανοδικών ρευμάτων [6]. Ως αποτέλεσμα έχουμε τη δημιουργία σωρειτομελανιών (καταιγιδοφόρα νέφη). Άλλες σπανιότερες αιτίες δημιουργίας μεγάλης διαφοράς δυναμικού, ικανής για τη δημιουργία ηλεκτρικών εκκενώσεων αποτελούν οι εκρήξεις ηφαιστείων, οι σκονοθύελλες και οι χιονοθύελλες. Οι περιοχές με έντονη κεραυνική δραστηριότητα, υποδηλώνουν την συχνή ύπαρξη έντονων ανοδικών ρευμάτων.



Εικόνα 1: Παγκόσμια κατανομή ηλεκτρικών εκκενώσεων σε μονάδες km^-2/yr-1 (NASA/GHRC/NSSTC Lightning Team, 2003)

Τα ηλεκτρικά φαινόμενα εμφανίζονται κυρίως στην ώριμη φάση των καταιγιδοφόρων νεφών και διακρίνονται στις εξής τρεις κατηγορίες: "IntraCloud" (IC) αποκαλούμε αυτά που εμφανίζονται εντός των νεφών, "Cloud-to-Cloud" (CC) αυτά που εμφανίζονται μεταξύ νεφών και τέλος τα "Cloud-to-Ground" (CG) που εμφανίζονται μεταξύ νέφους και εδάφους. Σπανιότερα, παρατηρούνται εκκενώσεις μεταξύ του περιβάλλοντα αέρα και νέφους. Οι εκκενώσεις της τρίτης κατηγορίας έχει επικρατήσει να αποκαλούνται κεραυνοί ενώ οι υπόλοιπες αστραπες.

Τα στάδια μίας εκκένωσης μεταξύ νέφους και εδάφους είναι τρία:

Αρχικά, με την συγκέντρωση αρνητικών φορτίων στη βάση των νεφών, τα ηλεκτρόνια απωθούνται κοντά στην επιφάνεια του εδάφους, με αποτέλεσμα το έδαφος να αποκτά θετικό φορτίο. Με την έλξη μεταξύ της αρνητικά φορτισμένης βάσης του νέφους και των θετικά φορτισμένων αντικειμένων του εδάφους να αυξάνεται, ηλεκτρόνια αρχίζουν να κατευθύνονται προς το έδαφος, δημιουργώντας μία 'διαδρομή'. Από την αντίθετη πλευρά, θετικά φορτία συνήθως από υψηλά αντικείμενα όπως κτίρια ή δέντρα αρχίζουν να κατευθύνονται προς τα επάνω με αντίστοιχο τρόπο.

Όταν οι δύο 'διαδρομές' συναντηθούν, ξεκινάει το δεύτερο στάδιο. Ο δίαυλος του κεραυνού ολοκληρώνεται και τα ηλεκτρόνια μεταφέρονται, ταχύτατα στο έδαφος. Το φως που βλέπουμε είναι ο φωτισμός του διαύλου, από κάτω προς τα πάνω, από τα πρώτα ηλεκτρόνια που φτάνουν στο έδαφος. Ο δίαυλος παύει να είναι φωτεινός όταν δεν υπάρχουν πια ηλεκτρόνια για να κινηθούν. Η διάρκεια του φαινομένου είναι μόλις μερικά χιλιοστά του δευτερολέπτου.

Στην τρίτη και τελευταία φάση μετά την πρώτη εκφόρτωση συνήθως ακολουθούν άλλες 3-4 κατά μήκος του ίδιου διαύλου, μεταφέροντας ωστόσο σαφώς μικρότερο φορτίο. Λόγω της πολύ μικρής διάρκειάς τους, ο άνθρωπος δεν είναι ικανός να τις ξεχωρίσει και τις βλέπει σαν μία η οποία 'τρεμοπαίζει' ελάχιστα, πιθανώς λόγω της διαφοράς χρόνου μεταξύ τους.

Ο ήχος που ακολουθεί το φαινόμενο, ονομάζεται βροντή. Δημιουργείται λόγω της έντονης θέρμανσης του αέρα (30.000ºK), στη διαδρομή που ακολουθεί το ηλεκτρικό ρεύμα με αποτέλεσμα τη δημιουργία ωστικού κύματος. Λόγω της διαφοράς στην ταχύτητα διάδοσης του φωτός και του ήχου, η βροντή ακούγεται μετά το φαινόμενο της ηλεκτρικής εκκένωσης ενώ με τη μέτρηση του χρόνου που μεσολαβεί μεταξύ των δύο φαινομένων, μπορεί να προσδιορισθεί η απόσταση μεταξύ παρατηρητή και τοποθεσίας πτώσης του κεραυνού. [7]

2.2 Το σύστημα ανίχνευσης ηλεκτρικών εκκενώσεων του Χαροκόπειου Πανεπιστημίου

Στο χώρο του Χαροκόπειου Πανεπιστημίου (37° 57' 39,55" N, 23° 42' 27,57" E) έχει εγκατασταθεί ένα σύστημα boltek ανίχνευσης και καταγραφής της θέσης των κεραυνών.

Οι Boltek ανιχνευτές είναι τεχνολογίας DF (Direction-Finding), δηλαδή χρησιμοποιούν κεραία εύρεσης κατεύθυνσης με στόχο την εξακρίβωση της κατεύθυνσης του σήματος. Γι το λόγο αυτό χρησιμοποιούνται γεωμετρικοί μέθοδοι για την εξακρίβωση της θέσης κάθε ηλεκτρικού φαινομένου καθώς και άλλοι αλγόριθμοι διόρθωσης αυτών των αποτελεσμάτων. Η διάταξη ενός τέτοιου συστήματος φαίνεται στην Εικόνα 2. Το σύστημα boltek χρησιμοποιεί μόνο ένα δέκτη και επιτρέπει σχετικά ακριβή ανίχνευση των φαινομένων. Ο δέκτης μέσω ηλεκτρονικής συσκευής επικοινωνεί με τον υπολογιστή, στον οποίο με εγκατεστημένο ειδικό λογισμικό υπάρχει δυνατότητα παραμετροποίησης, καταγραφής και οπτικοποίησης των δεδομένων.

Εικόνα 2: Διάταξη συστήματος ανίχνευσης ηλεκτρικών εκκενώσεων boltek

Το σύστημα στις εγκαταστάσεις του Χαροκόπειου Πανεπιστημίου λειτουργεί από το 2011 και βρίσκεται σε ύψος 11 μέτρα. Έχει δυνατότητα ανίχνευσης και καταγραφής κεραυνών σε απόσταση περίπου 500 χιλιομέτρων. [8] Με την συνεχή παρακολούθηση των δεδομένων που παράσχει και την σύγκρισή τους με άλλα συστήματα καταγραφής όπως το ZEUS, το blitzortung και το TOA σύστημα της ΕΜΥ, έχει καταστεί δυνατή η ποιοτική αξιολόγηση των αποτελεσμάτων. Έτσι, με τη συνεχή σύγκριση αποτελεσμάτων σε συνδιασμό με την παραμετροποίηση των ρυθμίσεων διόρθωσης του συστήματος, επιτυγχάνεται μία αρκετά ακριβής καταγραφή των ηλεκτρικών εκκενώσεων στο χώρο της Ελλάδας. Ωστόσο μέσα από την πολυετή παρακολούθηση προκύπτει επίσης η συστηματική εμφάνιση σφαλμάτων τόσο στον εντοπισμό της θέσης όσο και της συχνότητας εμφάνισης ηλεκτρικών εκκενώσεων σε συγκεκριμένες διευθύνσεις ανίχνευσης [9].

Τα πρωταρχικά δεδομένα παρέχονται από την Ομάδα Δυναμικής της Ατμόσφαιρας και του Κλίματος (ΟΔΑΚ) του Τμήματος Γεωγραφίας του Χαροκόπειου Πανεπιστημίου η οποία είναι υπεύθυνη για τη διαχείριση και διανομή τους από τον ανιχνευτή. Η διαδικασία ανάλυσης, παραμετροποίησης και καταγραφής των δεδομένων που θα χρησιμοποιήσουμε στην υλοποίηση της εφαρμογής μας, έγινε κατά την πτυχιακή εργασία του Γκίκα Νικόλαου με τίτλο ‘Ανάπτυξη αλγορίθμου για την αποτύπωση και ανάλυση της κεραυνικής δραστηριότητας στον Ελληνικό χώρο’ [9]. Κατά την εργασία του, ανέπτυξε λογισμικό σε γλώσσα C για την αποκωδικοποίηση των δεδομένων που παράγονται από το σύστημα ανίχνευσης. Τα δεδομένα μετατρέπονται από binary σε ASCII, τα οποία είναι σαφώς ευκολότερα στην επεξεργασία και αναγνώσιμα από τον άνθρωπο.

2.3 Hybrid mobile apps

Με τον όρο hybrid mobile apps [10] αναφερόμαστε στην ουσία σε μικρές ιστοσελίδες που τρέχουν σε περιβάλλον browser μέσα σε μία εφαρμογή και παράλληλα έχουν πρόσβαση στη πλατφόρμα και τις λειτουργίες της έξυπνης συσκευής. Πριν την δημιουργία των κατάλληλων εργαλείων για την υλοποίηση υβριδικών εφαρμογών, η δημιουργία εφαρμογών ξεχωριστά για κάθε λειτουργικό σύστημα έξυπνων συσκευών (native mobile apps) ήταν μονόδρομος για τους προγραμματιστές και τους ενδιαφερόμενους.

Η δημιουργία μιας τέτοιου είδους εφαρμογής έχει αρκετά πλεονεκτήματα σε σχέση με την ανάπτυξη εφαρμογής στοχευμένης σε συγκεκριμένο λειτουργικό σύστημα καθώς κερδίζουμε τόσο σε χρόνο ανάπτυξης όσο και υποστήριξης αφού μπορεί με κάποιες τροποποιήσεις να

χρησιμοποιηθεί σε διαφορετικές πλατφόρμες, όπως είναι οι ευρέως διαδεδομένες android, windows phone και iOS. Επιπλέον το κόστος ελαχιστοποιείται αφού για παράδειγμα αν θέλουμε να υλοποιήσουμε μία εφαρμογή για να καλύψουμε δύο λειτουργικά συστήματα όπως το Android και το iOS χρειάζεται μία ομάδα προγραμματιστών. Ακόμη συνδυάζει τις τεχνολογίες του διαδικτύου με τις τεχνολογίες των συσκευών.

Το μειονέκτημα μίας υβριδικής εφαρμογής σε σύγκριση με μία native είναι ο χρόνος απόκρισης. Μία υβριδική εφαρμογή επικοινωνεί με τον browser της συσκευής και ο browser με το λειτουργικό, ενώ μία native εφαρμογή επικοινωνεί απευθείας με το λειτουργικό. Έτσι στις υβριδικές εφαρμογές παρεμβαίνει ένα επιπλέον στάδιο, καθιστώντας τες πιο αργές [11].

Συμπερασματικά, η επιλογή δημιουργίας μιας εφαρμογής ως native είναι καλύτερη όταν θέλουμε να δημιουργήσουμε μία πολύπλοκη εφαρμογή, έχουμε ως στόχο την επίδοση και ο χρόνος και το κόστος δεν μας ενδιαφέρουν. Αντίθετα η επιλογή δημιουργίας μίας εφαρμογής ως υβριδική είναι προτιμότερη όταν θέλουμε γρήγορη ανάπτυξη του λογισμικού, ιδιαίτερα αν επιθυμούμε παράλληλη ανάπτυξη της εφαρμογής σε περισσότερες από μία πλατφόρμες, εύκολη συντήρηση και περιορισμούς στο κόστος.

Αν και δεν έχει ερευνηθεί το ποσοστό των υβριδικών εφαρμογών στην αγορά, το ενδιαφέρον για τα απαραίτητα frameworks για την υλοποίησή τους συνεχώς αυξάνεται. Το 2013 η Gartner Inc. έκανε την πρόβλεψη ότι μέχρι το 2016 περισσότερες από το 50% των εφαρμογών θα είναι υβριδικές [3].

2.4 JSON και GeoJSON

2.4.1 JSON

Το JSON [12] είναι ένα ελαφρύ πρότυπο ανταλλαγής δεδομένων. Για τους υπολογιστές είναι εύκολο να αναλυθεί και να παραχθεί ενώ παράλληλα για τους ανθρώπους να το διαβάσουν και να το γράψουν. Είναι βασισμένο σε ένα υποσύνολο της γλώσσας Javascript αλλά είναι τελείως ανεξάρτητο από γλώσσες προγραμματισμού, κάτι το οποίο το καθιστά κατάλληλο για την ανάλυση και καταγραφή δεδομένων.

Αξίζει να σημειωθεί ότι πρόκειται για το τύπο αρχείου που συναντάται πιο συχνά για ασύγχρονη μεταφορά δεδομένων μεταξύ διακομιστή και browser, αντικαθιστώντας μάλιστα το μέχρι πρότινος ιδιαίτερα δημοφιλές XML [13].

2.4.2 GeoJSON

Το GeoJSON [14] αποτελεί πρότυπο ανταλλαγής γεωχωρικών δεδομένων βασισμένο στο πρότυπο JSON, το οποίο υποστηρίζει τους παρακάτω γεωμετρικούς τύπους: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon και GeometryCollection. Κάθε feature σε ένα αρχείο geojson περιλαμβάνει ένα γεωμετρικό αντικείμενο και τις ιδιότητές του, ενώ το σύνολο τους ονομάζεται feature collection.

Παράδειγμα GeoJSON αρχείου:

```
{ "type": "FeatureCollection",
"features": [
      { "type": "Feature",
        "geometry": {"type": "Point", "coordinates": [102.0, 0.5]},
        "properties": {"prop0": "value0"}
        },
      { "type": "Feature",
        "geometry": {
          "type": "LineString",
          "coordinates": [ [102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]]
          },
        "properties": {"prop0": "value0","prop1": 0.0}
        }
]}
```

2.5 MySQL

Η MySQL [15] χρησιμοποιείται ως σχεσιακή βάση δεδομένων. Είναι από τις πλέον διαδεδομένες στο χώρο του διαδικτύου και το 2013 ήταν η πιο ευρέως χρησιμοποιούμενη ανοιχτού κώδικα σχεσιακή βάση δεδομένων. Η MySQL είναι διαθέσιμη υπό τους όρους της GNU General Public License και μερικών ιδιόκτητων συμφωνιών. Είναι απόλυτα συμβατή με διαδικτυακές εφαρμογές ή ιστοσελίδες που χρησιμοποιούν PHP ως γλώσσα προγραμματισμού και ως διακομιστή τον Apache, αφού αποτελεί και βασικό συστατικό του επίσης ανοιχτού λογισμικού LAMP (Linux,Apache,MySQL,PHP/Perl/Python) με στόχο την αρμονική συνύπαρξή τους. Αρχικά προωθούταν από την σουηδική εταιρεία MySQL AB, η οποία στη συνέχεια εξαγοράστηκε από την Oracle [7].

Για την γραφική απεικόνιση και διαχείριση των MySQL βάσεων δεδομένων έχει αναπτυχθεί το εργαλείο Phpmyadmin [16]. Μέσω του εργαλείου αυτού το οποίο είναι γραμμένο με τη χρήση της PHP και τρέχει διαδικτυακά στο server που είναι στημένη η βάση, μπορούμε να δούμε τους πίνακες, να τους τροποποιήσουμε, να δημιουργήσουμε χρήστες και να εκτελέσουμε οποιαδήποτε άλλη εντολή χρειαζόμαστε.

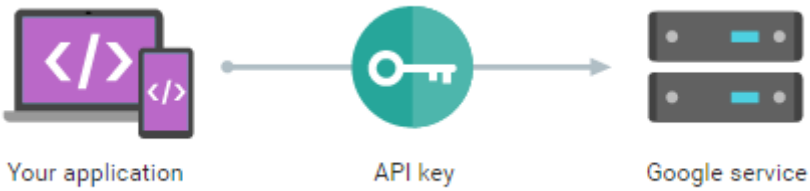
2.6 API

2.6.1 Γενικά

Ο όρος API [17], δηλαδή Application Programming Interface, μεταφράζεται στα ελληνικά ως Διεπαφή προγραμματισμού εφαρμογών. Ένα API μπορεί να παρέχεται από μία εφαρμογή, ένα λειτουργικό σύστημα, μία βιβλιοθήκη ή μία βάση δεδομένων ώστε να διευκολύνεται η ανταλλαγή αρχείων και η διασύνδεσή τους με άλλα προγράμματα. Ένα API ορίζει το σύνολο των λειτουργιών που παρέχει το εκάστοτε σύστημα, χωρίς όμως να επιτρέπει την πρόσβαση στον κώδικα που υλοποιούν τις υπηρεσίες αυτές.

Για παράδειγμα το λειτουργικό σύστημα Android παρέχει API για τη διασύνδεση της κάθε εφαρμογής με το υλικό της συσκευής όπως πχ. τη κάμερα, το gps κα.

2.6.2 Google APIs



Εικόνα 3: Το API key μεσολαβεί για την αυθεντικοποίηση της εφαρμογής στην υπηρεσία της Google

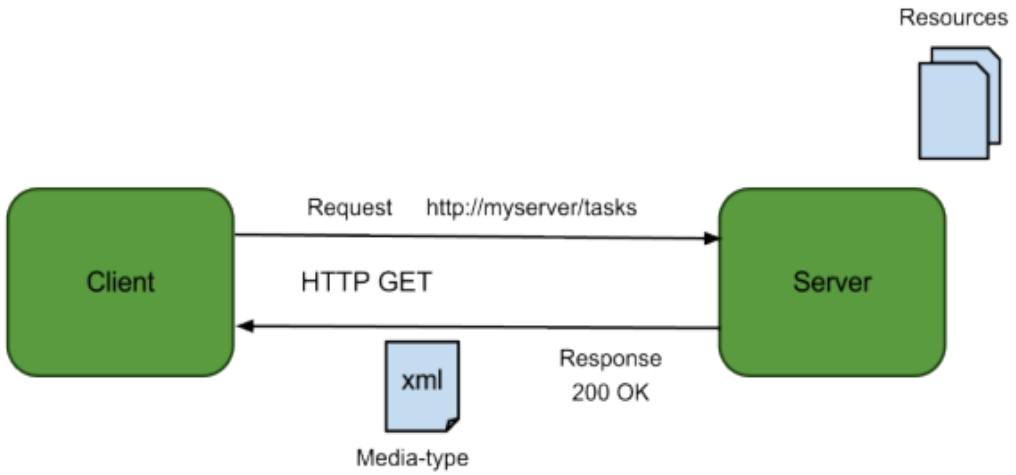
Google APIs [18] είναι οι διεπαφές προγραμματισμού εφαρμογών που έχει αναπτύξει η google για την επικοινωνία και τη χρησιμοποίηση των Google services από τρίτες εφαρμογές, καθώς και τη περαιτέρω ανάπτυξη των λειτουργιών τους. Μερικά από τα APIs της Google για να χρησιμοποιηθούν απαιτούν επαλήθευση ταυτότητας ή και αδειοδότηση. Ο προγραμματιστής πρέπει αρχικά να αποκτήσει πιστοποίηση από το Google API Console και με τη χρήση του API Key που παρέχεται, η εφαρμογή μπορεί στη συνέχεια να αιτηθεί τη χρησιμοποίηση της εκάστοτε API υπηρεσίας.

Συγκεκριμένα τα βήματα που πρέπει να πραγματοποιηθούν για την απόκτηση ενός API Key και τη χρησιμοποίησή του είναι τα εξής:

1. Είσοδος στην ιστοσελίδα https://console.developers.google.com/iam-admin/projects με λογαριασμό Google
2. Επιλέγουμε Create project και του δίνουμε την ονομασία X που επιθυμούμε
3. Στη σελίδα https://console.developers.google.com/apis/library?project=X όπου μεταφερόμαστε αυτόματα, επιλέγουμε Credentials από το μενού αριστερά.
4. Επιλέγουμε Create credentials
5. Από τις επιλογές που εμφανίζονται επιλέγουμε API Key και στη συνέχεια το τύπο που μας ενδιαφέρει. Στην περίπτωση της παρούσας εργασίας “Browser key”.
6. Δίνουμε ένα όνομα στο API Key μας και είμαστε έτοιμοι να χρησιμοποιήσουμε το API Key που μας δίνεται.
7. Επιστρέφουμε από το μενού στην επιλογή Overview και ενεργοποιούμε όσα API μας ενδιαφέρουν.

2.6.3 REST API

REST [19], δηλαδή Representational State Transfer είναι ένας τύπος αρχιτεκτονικής λογισμικού που χρησιμοποιείται για την υλοποίηση διαδικτυακών υπηρεσιών. Τα συστήματα που συμμορφώνονται με την αρχιτεκτονική REST, επικοινωνούν κυρίως με τη χρήση του πρωτοκόλλου HTTP, χρησιμοποιώντας τις μεθόδους POST, GET, DELETE και PUT που προσφέρονται για την ανάκτηση ιστοσελίδων αλλά και τη μεταφορά δεδομένων σε διακομιστές.



Εικόνα 4: Αρχιτεκτονική REST υπηρεσίας

Ένα σύστημα REST πρέπει να υπακούει στους εξής έξι περιορισμούς [19] [20]:

- **Uniform Interface**. Δηλαδή το διαχωρισμό ανάμεσα σε client και server. Όσο το interface μεταξύ τους παραμένει ίδιο, client και server μπορούν να αντικατασταθούν ή να αναπτυχθούν ξεχωριστά.
- **Client-Server**. Ο client δεν απασχολείται με τη μόνιμη αποθήκευση δεδομένων και ο server δεν ενδιαφέρεται για τη διεπαφή ή κατάσταση του χρήστη.
- **Stateless**. Καμία πληροφορία δεν αποθηκεύεται από τον διακομιστή σχετικά με τον client
- **Cacheable**. Ο client πρέπει να μπορεί να αποθηκεύει προσωρινά τις πληροφορίες που παρέχει το σύστημα και η υπηρεσία να δηλώνει αν η πληροφορία είναι cacheable, ώστε να αποτρέπεται η επαναχρησιμοποίηση της ή η χρήση λάθος δεδομένων.
- **Layered system**. Ο client δε πρέπει να γνωρίζει κατά πόσο συνδέεται και επικοινωνεί με τον τελικό διακομιστή ή κάποιον ενδιάμεσο.

- **Code on demand** (Προαιρετικά). Ο server είναι ικανός προσωρινά να διαμορφώσει ή να επεκτείνει την λειτουργικότητα του client μεταφέροντας προγραμματιστική λογική που μπορεί να εκτελεστεί στον client, όπως Javascript ή Java applets.

2.7 Clustering

2.7.1 Γενικά

Clustering [21] ή αλλιώς η πράξη της ομαδοποίησης εντοπίζει δομή σε ένα σύνολο δεδομένων και τα διαχωρίζει είτε με τη μορφή ομάδων αντικειμένων είτε με ιεραρχία ομάδων. Οι ομάδες αντικειμένων διαμορφώνονται σύμφωνα με ένα μέτρο απόστασης, με τέτοιο τρόπο ώστε αντικείμενα της ίδιας ομάδας να βρίσκονται όσο το δυνατόν πιο κοντά μεταξύ τους και παράλληλα απομακρυσμένα το μέγιστο από αντικείμενα άλλων ομάδων.

Τέσσερις βασικές οικογένειες αλγορίθμων ομαδοποίησης είναι οι εξής:

Ιεραρχικοί αλγόριθμοι: Δημιουργούν ιεραρχική δομή μεταξύ των αντικειμένων συνενώνοντας σε κάθε βήμα αντικείμενα που βρίσκονται κοντά. Οι συνενώσεις κάθε βήματος δεν μπορούν να μεταβληθούν.

Αλγόριθμοι τμηματοποίησης: Χωρίζουν το χώρο σε k περιοχές και τα αντικείμενα κάθε περιοχής αντιστοιχούν σε μία ομάδα.

Αλγόριθμοι βασισμένοι στην πυκνότητα: Σημεία που βρίσκονται κοντά μεταξύ τους σε σημαντική πυκνότητα συνενώνονται. Λόγω του τρόπου λειτουργίας τους οι αλγόριθμοι αυτοί εντοπίζουν ομάδες με σύνθετα σχήματα.

Γραφοθεωρητικοί αλγόριθμοι: Εντοπίζουν ομάδες αντικειμένων, βασισμένοι σε έννοιες της Θεωρείας Γράφων.

2.7.2 DBScan

Ο αλγόριθμος DBScan [21] είναι κατάλληλος για ομαδοποίηση σημείων με χαμηλή πυκνότητα, τα οποία μπορεί να διαχωρίζονται από άλλα σημεία χαμηλότερης πυκνότητας που αποτελούν

θόρυβο. Ο συγκεκριμένος αλγόριθμος προϋποθέτει ότι στη πυκνότητα των ομάδων δεν υπάρχουν μεγάλες διακυμάνσεις.

Για την εκτέλεσή του απαιτούνται δύο παράμετροι, Eps και minPts. Για κάθε ένα αντικείμενο – σημείο του συνόλου των δεδομένων ορίζεται η πυκνότητα ως ο αριθμός των σημείων, συμπεριλαμβανομένου του εαυτού του, που βρίσκονται σε ακτίνα μικρότερη ή ίση του Eps γύρω του.

Το αποτέλεσμα του DBScan εξαρτάται από τις τιμές των παραμέτρων Eps και minPts. Με βάση μία τιμή k βρίσκουμε για κάθε σημείο το k πλησιέστερό του σημείο όπως επίσης και την μεταξύ τους απόσταση. Ταξινομώντας τα σημεία ως προς την απόστασή τους από το k πλησιέστερο σημείο σε αυτά, καθορίζουμε τις τιμές Eps και minPts έτσι ώστε να διαχωρίζονται τα σημεία που ανήκουν σε ομάδες από τα θορυβώδη.

Ο αλγόριθμος DBScan διαχωρίζει τα δεδομένα σε τρεις κατηγορίες:

- Κεντρικά σημεία: Με πυκνότητα μεγαλύτερη ή ίση του minPts σε ακτίνα Eps
- Συνοριακά σημεία: Με πυκνότητα μικρότερη από minPts αλλά απέχει από ένα κεντρικό σημείο σε ακτίνα μικρότερη ή ίση από Eps.
- Θορυβώδη σημεία: Κάθε άλλο σημείο με πυκνότητα μικρότερη από minPts και απόσταση μεγαλύτερη από Eps από κάθε κεντρικό σημείο.

Βήματα εκτέλεσης

1. Χαρακτήρισε κάθε σημείο ως κεντρικό, συνοριακό ή θορυβώδες.
2. Αγνόησε τα θορυβώδη σημεία εντελώς.
3. Δημιούργησε ένα γράφο με μία κορυφή για κάθε κεντρικό ή συνοριακό σημείο
4. Πρόσθεσε μία ακμή για κάθε ζεύγος κεντρικών σημείων με απόσταση μικρότερη ή ίση από Eps.
5. Ανέθεσε στην ίδια ομάδα τα κεντρικά σημεία που ανήκουν σε συνδεδεμένη συνιστώσα του γράφου
6. Ανέθεσε κάθε συνοριακό σημείο στην ομάδα κάποιου κεντρικού σημείου από το οποίο απέχει απόσταση ίση ή μικρότερη από Eps. Αν υπάρχουν πάνω από ένα ένα τέτοια σημεία, εφάρμοσε δευτερεύον κριτήριο όπως για παράδειγμα την μικρότερη απόσταση

Ο ψευδοκώδικας υλοποίησης [22] [23] του αλγορίθμου DBScan όπως παρουσιάστηκε κατά την αρχική του δημοσίευση μπορεί να βρεθεί στο Παράρτημα Α: Ψευδοκώδικας αλγορίθμου DBScan στο τέλος της εργασίας.

Πλεονεκτήματα

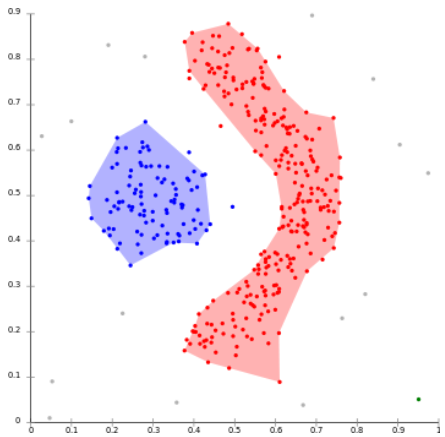
- Δεν απαιτεί εκ των προτέρων τον προσδιορισμό των ομάδων
- Έχει την ικανότητα εντοπισμού ομάδων με αυθαίρετα σχήματα
- Δεν επηρεάζεται από ακραίες τιμές
- Μικρή ευαισθησία ως προς την σειρά εμφάνισης των δεδομένων
- Σχετικά εύκολος προσδιορισμός των παραμέτρων έπειτα από μελέτη των δεδομένων

Μειονεκτήματα

- Δεν είναι αρκετά ντετερμινιστικός, δηλαδή τα περιθωριακά σημεία μίας ομάδας μπορεί να ανήκουν είτε σε αυτή είτε σε κάποια γειτονική
- Το αποτέλεσμα εξαρτάται εκτός των παραμέτρων Eps, minPts και από τη μετρική απόστασης που θα χρησιμοποιηθεί
- Ομάδες δεδομένων με διαφορετική πυκνότητα δεν ομαδοποιούνται σωστά

Πολυπλοκότητα

Ο αλγόριθμος DBScan επισκέπτεται κάθε σημείο πιθανόν περισσότερες από μία φορές, ωστόσο η πολυπλοκότητα του αλγορίθμου πηγάζει κυρίως από την πράξη εντοπισμού των γειτόνων κάθε σημείου. Έτσι στη χειρότερη περίπτωση και με δεδομένο ότι πρέπει να εντοπίσουμε τους γείτονες για κάθε ένα σημείο, η πολυπλοκότητα του αλγορίθμου είναι O(n^2).



Εικόνα 5: Παράδειγμα εκτέλεσης του DBScan αλγορίθμου

Κεφάλαιο 3

Λειτουργικές απαιτήσεις και αρχιτεκτονική εφαρμογής

Στο αυτό το κεφάλαιο παρουσιάζουμε τις απαιτήσεις του συστήματος που υλοποιήθηκε, δηλαδή τι χρειαζόμαστε αλλά και τι χρησιμοποιήσαμε στη συνέχεια για να επιτύχουμε το επιθυμητό αποτέλεσμα.

3.1 Λειτουργικές απαιτήσεις συστήματος

Στην συγκεκριμένη ενότητα παρουσιάζονται οι λειτουργικές απαιτήσεις του συστήματος που υλοποιήθηκε στην παρούσα εργασία. Οι λειτουργικές απαιτήσεις περιγράφουν τις αρμοδιότητες του συστήματος.

1. **Ομαδοποίηση δεδομένων**: Το σύστημα πρέπει να έχει τη δυνατότητα ομαδοποίησης δεδομένων και πιο συγκεκριμένα των κεραυνών.
2. **Παρουσίαση δεδομένων κεραυνών σε χάρτη**: Το σύστημα πρέπει να έχει τη δυνατότητα παρουσίασης των κεραυνών σε χάρτη.
3. **Ανανέωση δεδομένων κεραυνών στο χάρτη**: Το σύστημα πρέπει να έχει τη δυνατότητα ανανέωσης των κεραυνών στο χάρτη.
4. **Παρουσίαση ομάδων κεραυνών στο χάρτη**: Το σύστημα πρέπει να έχει τη δυνατότητα παρουσίασης των ομάδων των κεραυνών σε χάρτη.
5. **Εισαγωγή επιλογών από τον χρήστη**: Το σύστημα πρέπει να παρέχει τη δυνατότητα στον χρήστη να εισάγει περιοχές της επιλογής του και να μπορεί να επιλέξει το διάστημα ωρών που τον ενδιαφέρει.
6. **Αποθήκευση επιλογών του χρήστη**: Το σύστημα πρέπει να έχει τη δυνατότητα αποθήκευσης των επιλογών που εισάγει ο χρήστη.
7. **Εντοπισμός τοποθεσίας του χρήστη**: Το σύστημα πρέπει να έχει τη δυνατότητα εντοπισμού της γεωγραφικής θέσης του χρήστη τη τρέχουσα στιγμή και την προβολή του σε χάρτη.

3.2 Μη λειτουργικές απαιτήσεις συστήματος

Στην συγκεκριμένη ενότητα παρουσιάζονται οι μη λειτουργικές απαιτήσεις του συστήματος που υλοποιήθηκε στην παρούσα εργασία. Οι μη λειτουργικές απαιτήσεις περιγράφουν τις ιδιότητες

του συστήματος, δηλαδή το πώς ή το πόσο καλά το σύστημα θα υποστηρίξει τις λειτουργικές απαιτήσεις που ορίσθηκαν.

Φορητότητα:

1. Να καλύπτει το μεγαλύτερο φάσμα των χρηστών smartphone.

Επαναχρησιμοποίηση:

2. Να υπάρχει δυνατότητα επαναχρησιμοποίησης των δεδομένων από άλλες εφαρμογές στο μέλλον.

Αποδοτικότητα:

3. Να λειτουργεί αδιάλειπτα.
4. Να μην επιβαρύνεται ο χρήστης και η συσκευή του για την ομαδοποίηση των δεδομένων.
5. Ο χρόνος απόκρισης του συστήματος κατά την φόρτωση των δεδομένων στον χάρτη να μην υπερβαίνει τα 3 δευτερόλεπτα.
6. Ο χρόνος απόκρισης του συστήματος εντοπισμού τοποθεσίας να μην υπερβαίνει τα 4 δευτερόλεπτα.

Ευχρηστία:

7. Να μην επιβαρύνεται ο χρήστης με επιπλέον πληροφορία από αυτή που επιθυμεί.
8. Ενημέρωση του χρήστη αν δεν υπάρχει πρόσβαση στο διαδίκτυο και αποκοπή των υπηρεσιών για τις οποίες είναι απαραίτητη.
9. Ενημέρωση του χρήστη αν η λειτουργία GPS δεν είναι ενεργοποιημένη.

Ιδιωτικότητα:

10. Να μην αποθηκεύονται πληροφορίες για την τρέχουσα τοποθεσία του χρήστη, παρά μόνο προσωρινά και αποκλειστικά στην συσκευή.
11. Να αποθηκεύονται αποκλειστικά στη μνήμη της συσκευής οι προσωπικές επιλογές του χρήστη.

Ασφάλεια:

12. Η ομαδοποίηση των κεραυνών δεν θα πρέπει να μπορεί να επηρεαστεί από τον χρήστη.

3.3 Αρχιτεκτονική συστήματος

Το σύνολο του συστήματος της παρούσας εργασίας, με βάση τις απαιτήσεις που προέκυψαν και αναλύθηκαν στις ενότητες 3.1 Λειτουργικές απαιτήσεις συστήματος και 3.2 Μη λειτουργικές απαιτήσεις συστήματος, αποτελείται από δύο μέρη:

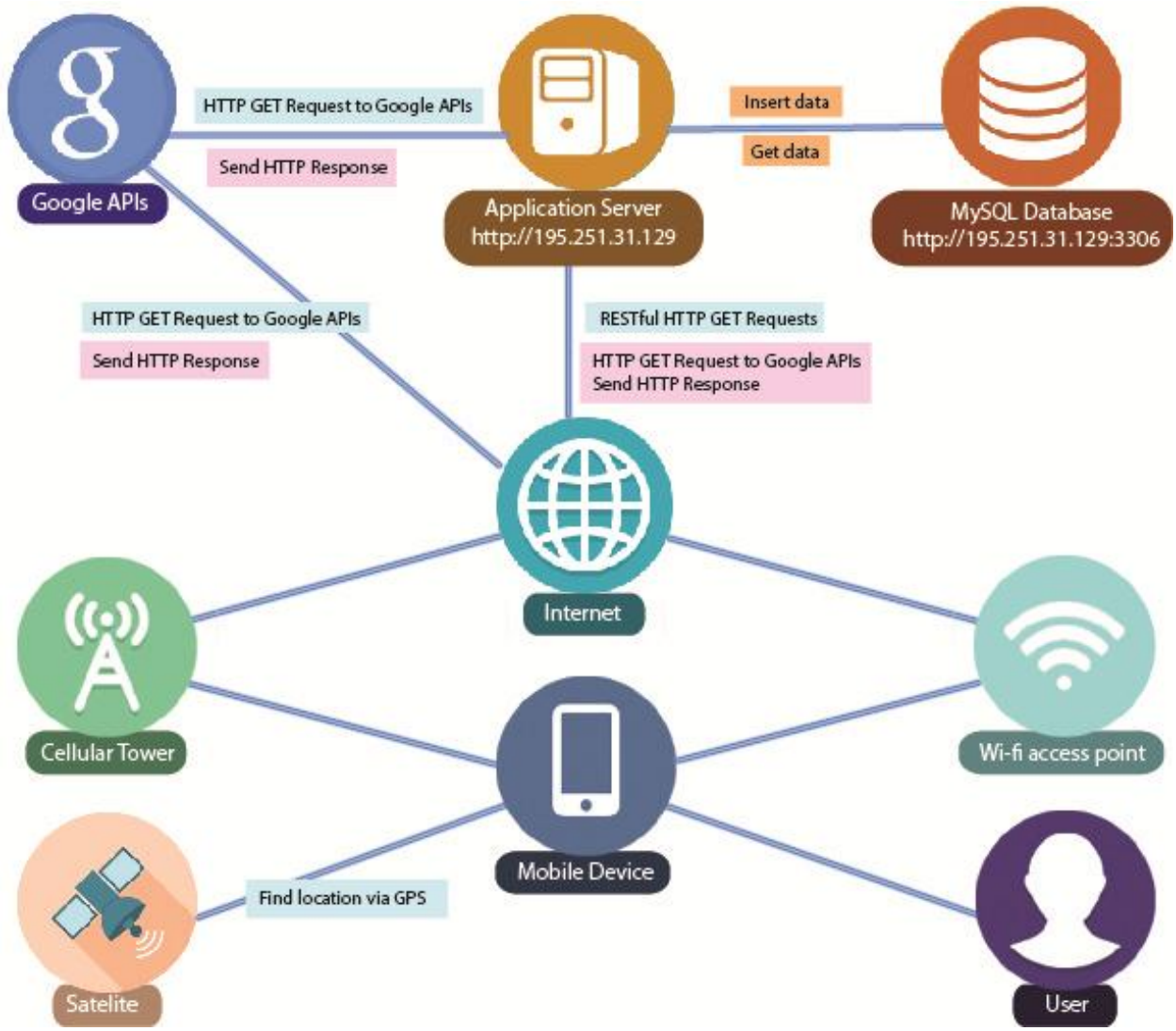
- Client App
- App Server

Η client app ή αλλιώς εφαρμογή-πελάτης που υλοποίησα αποτελείται από την υβριδική εφαρμογή για έξυπνα τηλέφωνα και ταμπλέτες. Είναι υπεύθυνη για την αποτύπωση των κεραυνών στο χώρο της Ελλάδας σε πραγματικό χρόνο καθώς και την παροχή πληροφορίας στο χρήστη. Για την υλοποίησή της χρησιμοποιήθηκαν τα Frameworks Apache Cordova (4.2.2.4 Apache Cordova) και Ionic (4.2.2.5 IONIC Framework) για λόγους που αναλύονται στο επόμενο κεφάλαιο.

Ο App server ή εξυπηρετητής εφαρμογής επικοινωνεί με την εφαρμογή-πελάτη και είναι υπεύθυνος για την συλλογή και αποθήκευση στοιχείων στη βάση δεδομένων καθώς επίσης στη συνέχεια για την ανάκτησή και ομαδοποίησή τους. Μέσω της REST υπηρεσίας που υλοποιήθηκε με τη βοήθεια του CodeIgniter Framework (

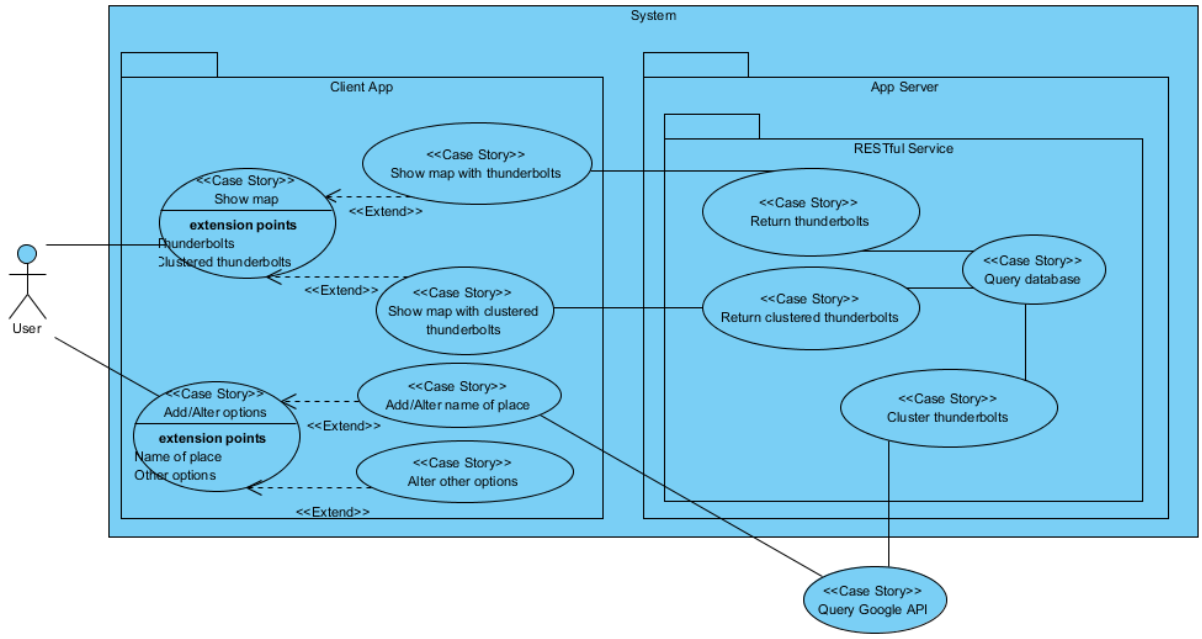
4.2.1.2 CodeIgniter) δύναται η παροχή πληροφοριών σχετικά με τους κεραυνούς, στην υβριδική εφαρμογή σε μορφή GeoJSON μέσω της GET μεθόδου της HTTP.
Ο app server που χρησιμοποιήθηκε βρίσκεται στη διεύθυνση http://195.251.31.119, με λειτουργικό σύστημα CentOS release 6.4 (final). Αναλυτικότερα σε αυτόν περιέχονται:

- Μία βάση δεδομένων MySQL 5.1.66 στην πόρτα 3306, όπου αποθηκεύονται τα δεδομένα των κεραυνών.
- Ένας Apache web server 2.2.15 (Unix) με εγκατεστημένη PHP 5.3.3 (cli) στον οποίο έχει υλοποιηθεί η REST υπηρεσία που είναι υπεύθυνη για τη παροχή πληροφοριών στην εφαρμογή-πελάτη. Επίσης εκεί υλοποιείται και ο αλγόριθμος ομαδοποίησης που χρησιμοποιείται με σκοπό την εύρεση των περιοχών με τους περισσότερους κεραυνούς στο διάστημα των τελευταίων 24 ωρών.



Εικόνα 6: Αρχιτεκτονική συστήματος

Όπως φαίνεται και στην Εικόνα 6, ο χρήστης χρησιμοποιεί την εφαρμογή-πελάτη στο έξυπνο κινητό του ή ταμπλέτα. Η εφαρμογή μέσω GPS έχει τη δυνατότητα να ανακτήσει τις συντεταγμένες του χρήστη. Συνδέεται στο διαδίκτυο μέσω wifi ή του δικτύου κινητής τηλεφωνίας και τότε μπορεί να επικοινωνήσει με τον app server. Με τη χρήση της GET μεθόδου λαμβάνει τα στοιχεία των κεραυνών σε μορφή GeoJSON. Στη συνέχεια με τη βοήθεια των κατάλληλων βιβλιοθηκών της google τα δεδομένα παρουσιάζονται σε χάρτες παρέχοντας την επιθυμητή πληροφορία στο χρήστη. Ακόμη, ο app server ανά τακτά χρονικά διαστήματα ανανεώνει τη βάση δεδομένων με τα νέα δεδομένα των κεραυνών. Τέλος εκτελείται και ομαδοποίηση των δεδομένων βάσει του DBScan αλγορίθμου με σκοπό την εξαγωγή συμπεράσματος για τις περιοχές με τους περισσότερους κεραυνούς τις τελευταίες ώρες.



Εικόνα 7: Use Case διάγραμμα συστήματος

Κεφάλαιο 4

Υλοποίηση

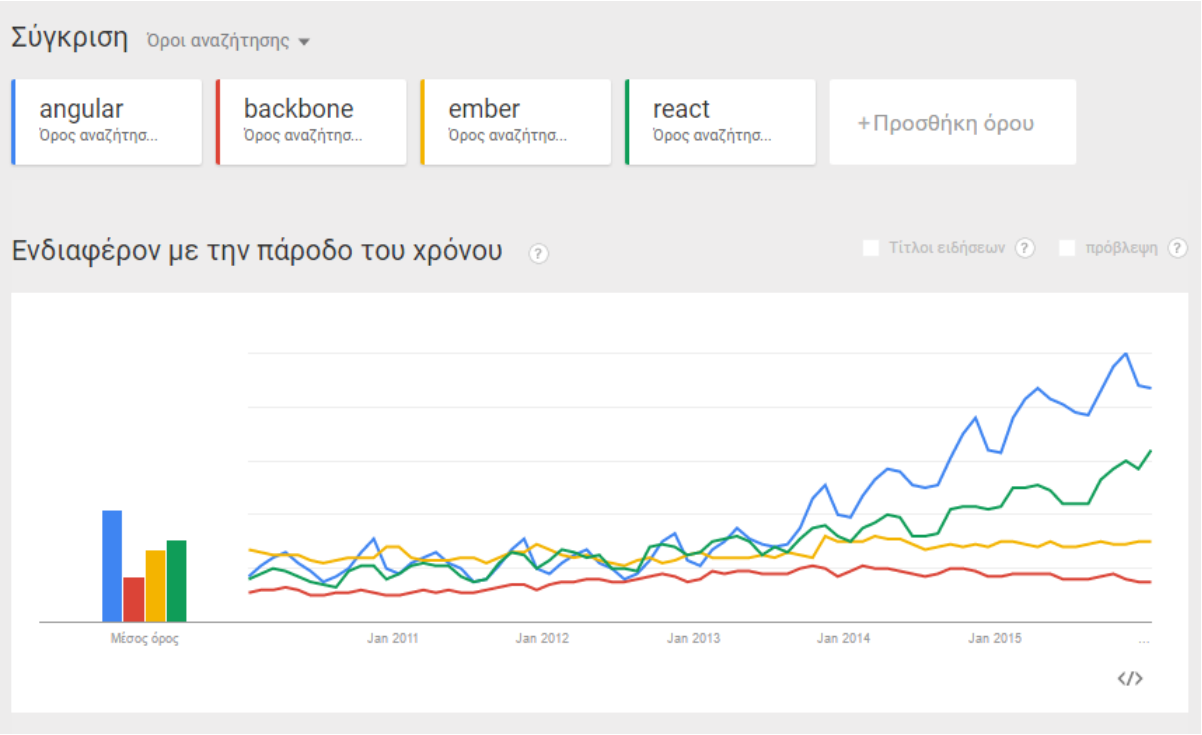
Γενικά

Στο παρόν κεφάλαιο θα παρουσιάσουμε το περιβάλλον στο οποίο υλοποιήθηκε η εργασία, τα frameworks που χρησιμοποιήθηκαν και τους λόγους επιλογής τους. Αποτελεί επίσης ένα σύντομο οδηγό για το στήσιμο του περιβάλλοντος ανάπτυξης του συστήματος που πραγματευόμαστε.

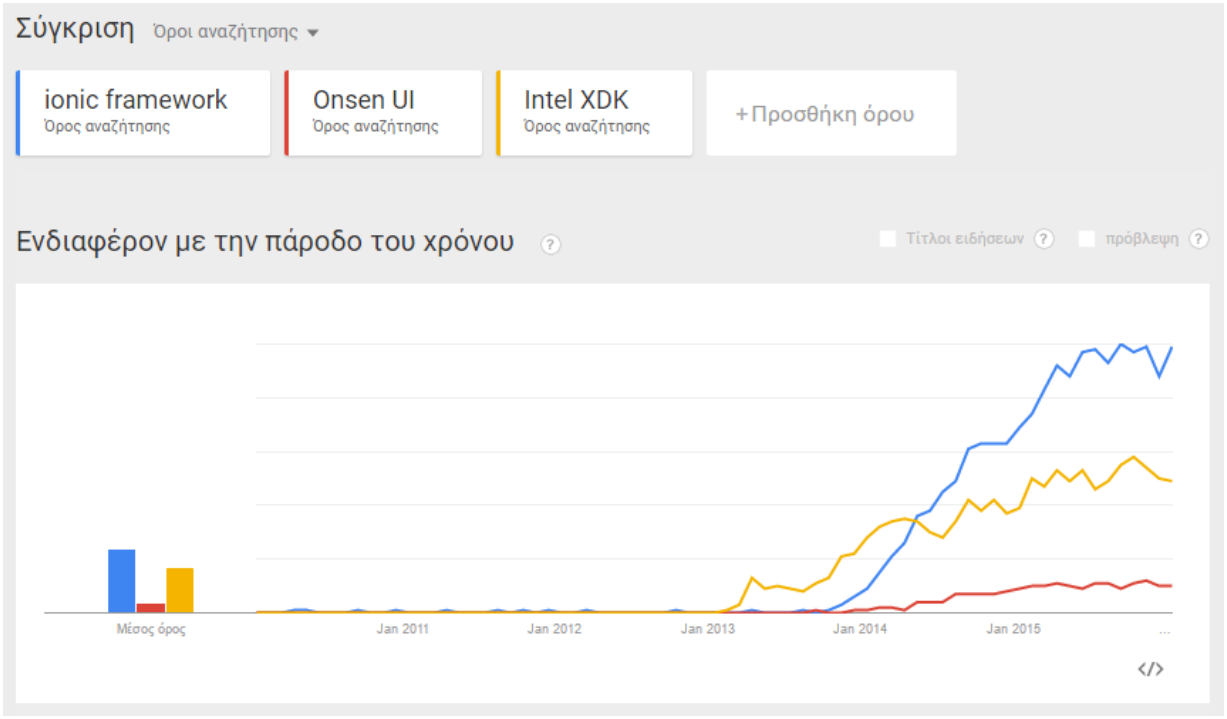
Η παρούσα εργασία όπως αναλύσαμε στο κεφάλαιο 4 αποτελείται από δύο μέρη. Το πρώτο αποτελεί την εφαρμογή-πελάτης, η οποία υλοποιήθηκε ως υβριδική ώστε να μην περιοριστεί σε ένα λειτουργικό σύστημα αλλά να υπάρχει η δυνατότητα χρήσης της από μεγαλύτερο φάσμα χρηστών. Συγκεκριμένα η εφαρμογή-πελάτης έχει δοκιμαστεί σε συσκευές με λειτουργικό Android 4.2, 4.3 και 5 και σε simulator με λειτουργικό σύστημα iOS 8. Για να επιτευχθεί αυτό, χρησιμοποιήθηκαν τα Cordova και Ionic frameworks με τα προαπαιτούμενά τους όπως θα αναλυθούν στη συνέχεια του κεφαλαίου.

Για την δημιουργία της RESTful υπηρεσίας, στον εξυπηρετητή της εφαρμογής, που μεταφέρει τα δεδομένα από τη βάση δεδομένων στην εφαρμογή-πελάτη σε μορφή GeoJSON, χρησιμοποιήθηκε το framework CodeIgniter λόγω του ότι είναι ελαφρύ, ταχύ και προσφέρει δεκάδες έτοιμες λύσεις σε καίρια ζητήματα.

Τα frameworks στα οποία θα αναφερθούμε σε αυτό το κεφάλαιο και χρησιμοποιήθηκαν στην υλοποίηση της εργασίας, τα χρησιμοποίησα για πρώτη φορά στο ‘2^{ο} θερινό σχολείο ανάπτυξης κώδικα ΕΛ/ΛΑΚ’ του Χαροκόπειου Πανεπιστημίου που πραγματοποιήθηκε στις 30/03/2015 με 06/04/2015. Ακόμη όπως φαίνεται και από τα διαγράμματα της Εικόνα 8 και Εικόνα 9, η γλώσσα angularJS που χρησιμοποιείται κατά κόρων από το Apache Cordova framework όπως και το Ionic framework αποτελούν αναμφισβήτητα την νούμερο ένα επιλογή σε σχέση με τους ανταγωνιστές τους για τη δημιουργία υβριδικών εφαρμογών. Έτσι μου προξένησαν το ενδιαφέρον και θέλησα να εμβαθύνω τις γνώσεις μου σε αυτά.



Εικόνα 8: Σύγκριση της AngularJS με τους ανταγωνιστές της, στο Google trends.



Εικόνα 9: Σύγκριση του Ionic framework και των ανταγωνιστών του, σύμφωνα με το Google trends

4.1 Εργαλεία ανάπτυξης λογισμικού και διαχείρισης

4.1.1 Netbeans IDE

Το Netbeans IDE αποτελεί εργαλείο ανάπτυξης λογισμικού υλοποιημένο σε γλώσα Java, το οποίο όμως υποστηρίζει την ανάπτυξη εφαρμογών εκτός από Java και Javascript σε PHP, C/C++ και HTML. Ακόμη είναι ανεξάρτητο λειτουργικού συστήματος και μπορεί να εγκατασταθεί σε Windows, Mac OS X, Linux, Solaris και σε άλλες πλατφόρμες οι οποίες υποστηρίζουν το JVM (Java Virtual Machine).

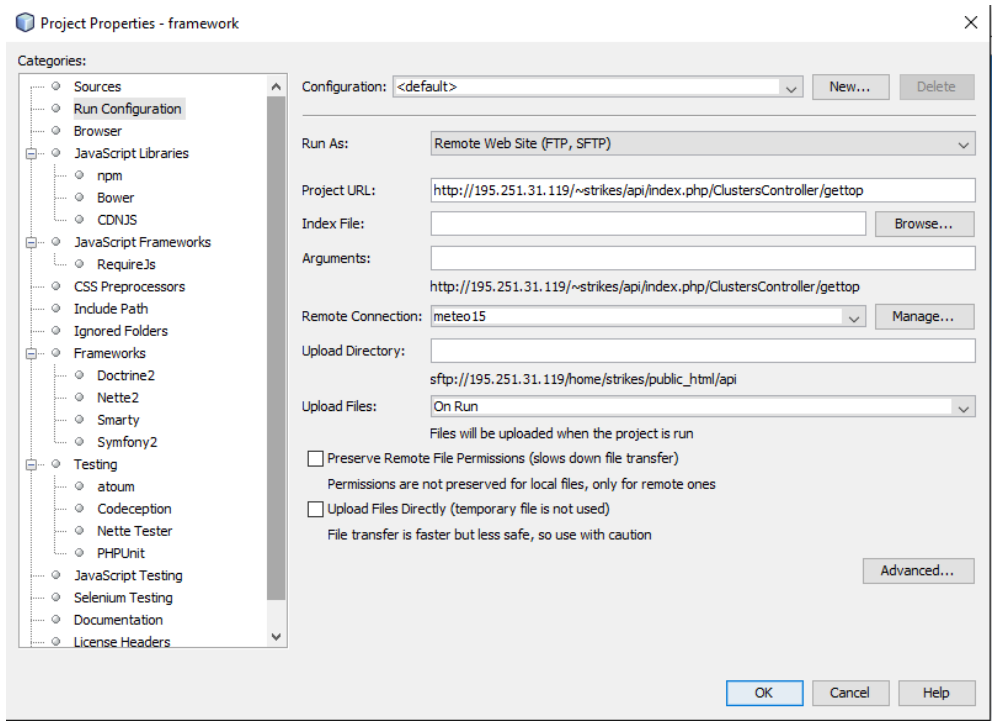
Το Netbeans [24] IDE (integrated development environment) προτιμήθηκε μεταξύ των διαθέσιμων εργαλείων ανάπτυξης λογισμικού καθώς για την ανάπτυξη της εφαρμογής πελάτη δεν υπάρχουν ιδιαίτερες απαιτήσεις πέρα από Javascript, HTML και CSS αφού η διαδικασία build γίνεται μέσω command line. Για την ανάπτυξη της RESTful υπηρεσίας σε γλώσσα PHP επίσης κρίνεται επαρκές, οπότε σε συνδιασμό και με την πρότερη εμπειρία μου πάνω στο συγκεκριμένο λογισμικό, αποφάσισα να το χρησιμοποιήσω.

Μεταξύ των διαθέσιμων εκδόσεων: Java SE, Java EE, C/C++, PHP και ALL, επιλέγουμε την έκδοση ALL. Αφού ολοκληρωθεί η μεταφόρτωση από την επίσημη ιστοσελίδα του λογισμικού και το εγκαταστήσουμε, είμαστε έτοιμοι να ξεκινήσουμε την ανάπτυξη του λογισμικού μας.

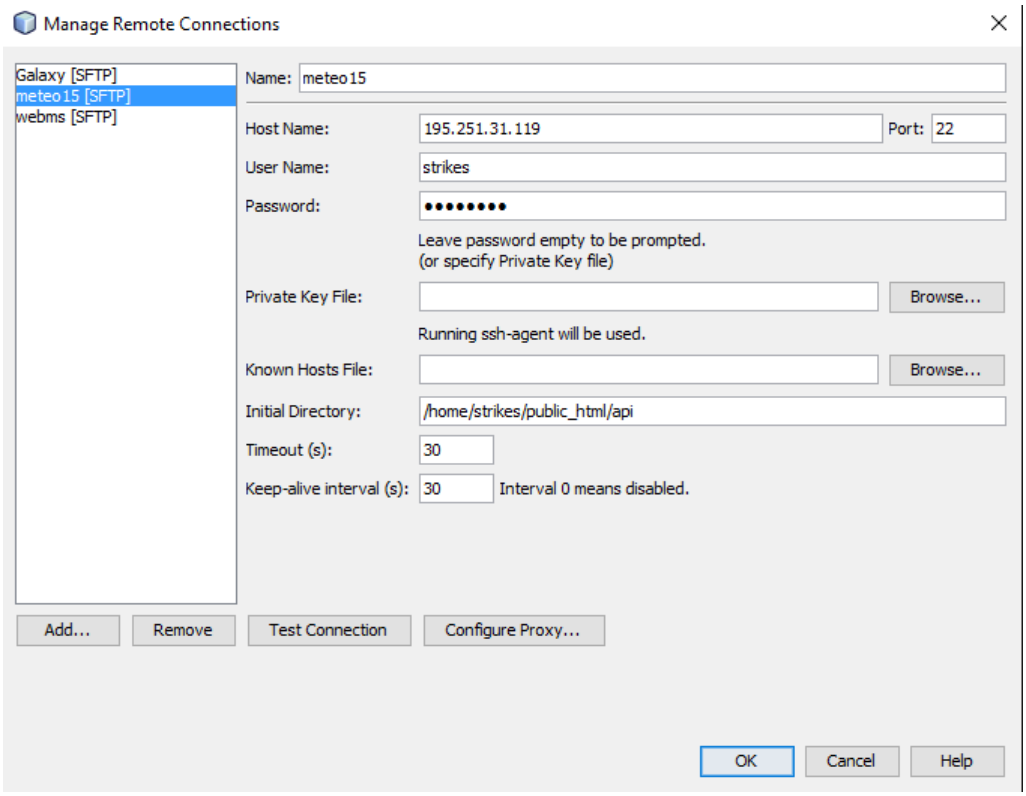
Για την αποθήκευση του server στον οποίο επιθυμούμε να μεταφορτώσουμε την RESTful υπηρεσία μας ακολουθούμε τα παρακάτω βήματα:

1. Ανοίγουμε τα properties του project μας και επιλέγουμε run configuration.
2. Όπως φαίνεται στην Εικόνα 10 επιλέγουμε :
    - Run as: Remote Web Site (FTP,SFTP).
    - Project URL: Δηλώνουμε το url στο οποίο επιθυμούμε να μεταβούμε κατά το run του project.
    - Upload Files: On Run.
3. Remote Connection: Επιλέγουμε Manage για την αποθήκευση των ρυθμίσεων του server μας.
    - Επιλέγουμε Add, στο παράθυρο που εμφανίζεται δίνουμε ένα όνομα για τον server μας και θέτουμε λειτουργία SFTP.

- Στη συνέχεια όπως φαίνεται και στην Εικόνα 11 δηλώνουμε τα στοιχεία του server μας και τον φάκελο του server στον οποίο επιθυμούμε να μεταφέρονται τα αρχεία του project μας. Τέλος πατάμε το κουμπί ‘OK’.



- **Εικόνα 10: Παράθυρο Run configuration από τις ιδιότητες του project**



- **Εικόνα 11: Στο παράθυρο Manage remote connections συμπληρώνουμε τα στοιχεία του server που θέλουμε να μεταφορτώσουμε την υπηρεσία μας**

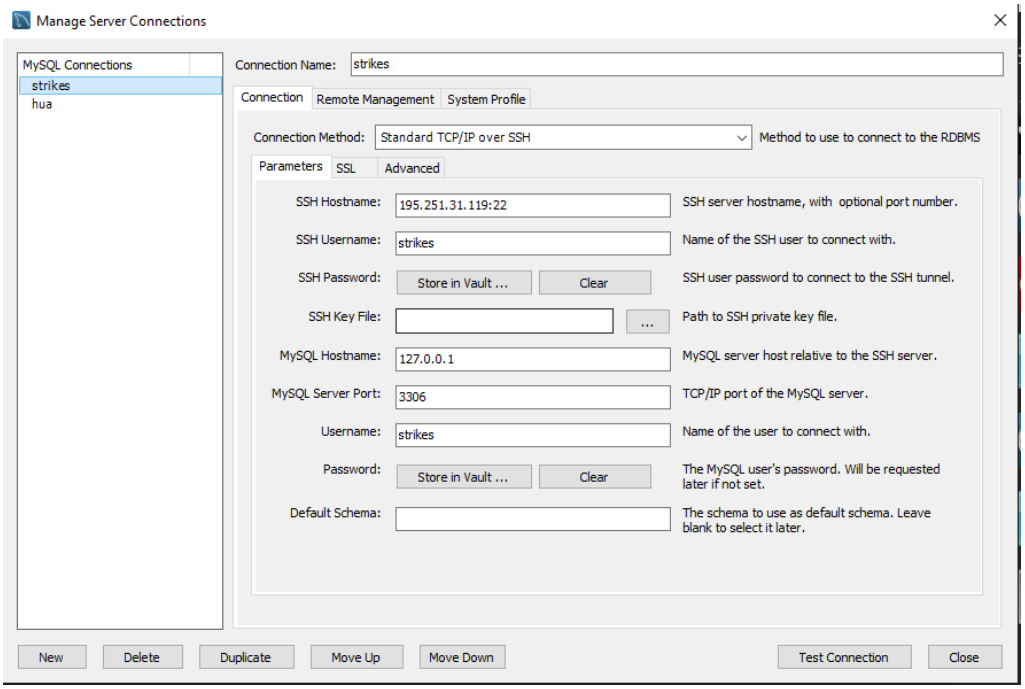
4.1.2 MySQL Workbench

Το MySQL Workbench [25] αποτελεί εργαλείο για την γραφική απεικόνιση και διαχείριση βάσεων δεδομένων, μεταξύ των οποίων και της MySQL. Εγκαθίσταται στο τοπικό μηχάνημα του προγραμματιστή / χρήστη και συνδέεται στη βάση δεδομένων μέσω ssh. Μέσω της διεπαφής του μπορούμε σε γραφικό περιβάλλον να δούμε τους πίνακες, να τους τροποποιήσουμε, να δημιουργήσουμε χρήστες και να εκτελέσουμε οποιαδήποτε άλλη εντολή χρειαζόμαστε.

Το εργαλείο MySQL Workbench προτιμήθηκε λόγω της πρότερης εμπειρίας που είχα με το συγκεκριμένο λογισμικό ενώ παράλληλα το επίσημο λογισμικό γραφικής απεικόνισης της MySQL PhpMyAdmin δεν ήταν εγκατεστημένο στον server.

Αφού ολοκληρωθεί η μεταφόρτωση από την επίσημη ιστοσελίδα του λογισμικού[4] και το εγκαταστήσουμε, είμαστε έτοιμοι να συνδεθούμε στη βάση μας.

Για να συνδεθούμε από τις διαθέσιμες επιλογές, επιλέγουμε Database > Manage connections > New και εισάγουμε τα αντίστοιχα στοιχεία του server και της βάσης μας με την Εικόνα 12.



Εικόνα 12: Παράθυρο Manage server connections του MySQL Workbench

[4] https://www.mysql.com/products/workbench/

4.2 Frameworks

Στον προγραμματισμό, framework [26] ονομάζεται το λογισμικό που παρέχει γενικές λειτουργίες και το οποίο μπορεί να τροποποιηθεί από τον προγραμματιστή κατάλληλα για την υλοποίηση συγκεκριμένης εργασία μέσα σε ένα μεγαλύτερο project. Ένα framework μπορεί να περιλαμβάνει μεταξύ άλλων βιβλιοθήκες κώδικα, μεταγλωττιστές και Application Programming Interfaces (APIs) που ενώνουν διαφορετικά συστατικά μιας εφαρμογής και επιτρέπουν την υλοποίησή της.

4.2.1 RESTful API υπηρεσία

Σε αυτή την ενότητα παρουσιάζεται το framework CodeIgniter καθώς και το λογισμικο Rhino που χρησιμοποιήθηκε κατά την υλοποίηση της REST υπηρεσίας για την εκτέλεση JavaScript κώδικα στον server.

4.2.1.1 Rhino

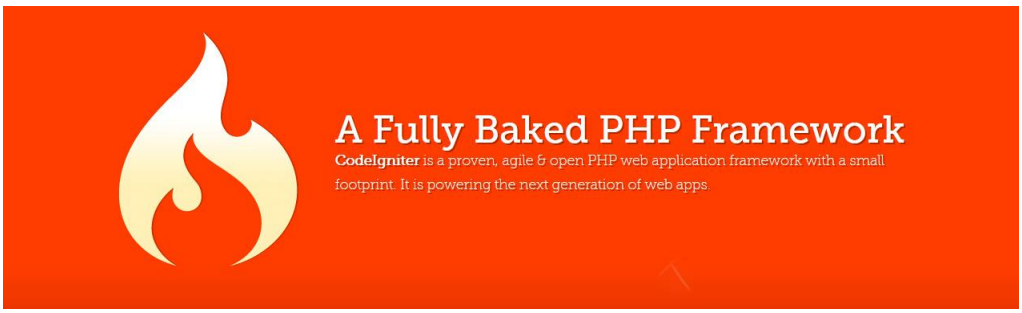
Γενικά

Το λογισμικό Rhino [27] αποτελεί μία ανοιχτού κώδικα μηχανή JavaScript γραμμένη εξ ολοκλήρου σε γλώσσα Java. Χρησιμοποιείται συνήθως για την εκτέλεση JavaScript σε server-side εφαρμογές, μετατρέποντάς τη σε κλάσεις.

Εγκατάσταση και χρήση

Κατά την υλοποίηση της παρούσας εργασίας μας ενδιέφερε η εκτέλεση JavaScript κώδικα μέσω script στον server. Για την συγκεκριμένη λειτουργία:

1. Κατεβάζουμε την έκδοση που μας ενδιαφέρει από την επίσημη ιστοσελίδα (https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino/Download_Rhino).
2. Μεταφέρουμε το αρχείο .jar στον server.
3. Εκτελούμε κώδικα JavaScript ως εξής: java -jar path/rhino-1.7.7.1.jar examplefile.js



4.2.1.2 CodeIgniter

Γενικά

Το λογισμικό CodeIgniter [28] είναι ένα framework που τελεί υπό την MIT άδεια. Έχει στόχο τη δημιουργία διαδικτυακών διαδραστικών εφαρμογών γρηγορότερα, παρέχοντας μία μεγάλη ποικιλία βιβλιοθηκών κώδικα για εργασίες που συναντώνται συχνά στις διαδικτυακές εφαρμογές. Το CodeIgniter σύμφωνα και με την επίσημη ιστοσελίδα του είναι πολύ ελαφρύ, έχει γρήγορες επιδόσεις, χρειάζεται ελάχιστη διαμόρφωση, έχει χαμηλές απαιτήσεις από τον διακομιστή (PHP version 5.4 ή νεότερη έκδοση) και υποστηρίζει πληθώρα βάσεων δεδομένων, μεταξύ αυτών και τη MySQL.

Αρχιτεκτονική

Βασίζεται στη προγραμματιστική δομή MVC, τα οποία είναι αρχικά για Model, View, Controller. Η προσέγγιση αυτή στηρίζεται στη λογική ότι η παρουσίαση των πληροφοριών στο χρήστη διαχωρίζεται από την υπόλοιπη εφαρμογή.

Model

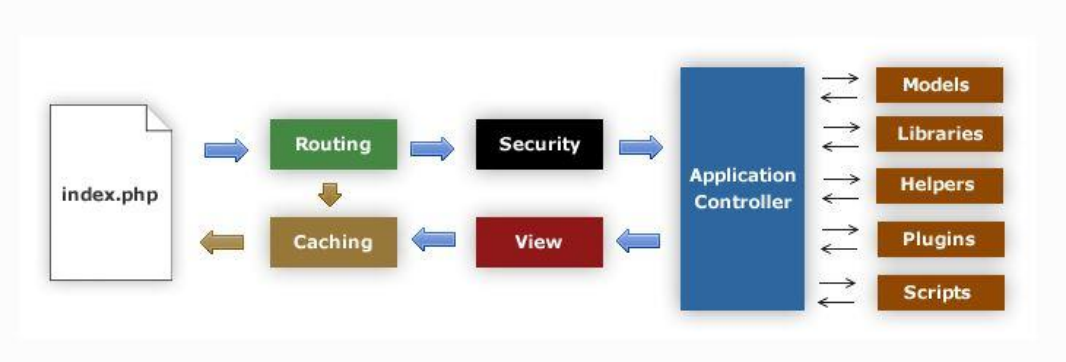
Αναπαριστά τη δομή των δεδομένων. Τυπικά περιλαμβάνει συναρτήσεις σχετικές με την ανάκτηση, εισαγωγή και την ενημέρωση των δεδομένων στη βάση.

View

Περιλαμβάνει τις πληροφορίες που θέλουμε να λάβει ο χρήστης. Συνήθως είναι μία ιστοσελίδα αλλά μπορεί να είναι οποιουδήποτε τύπου όπως xml ή json.

Controller

Λειτουργεί ως μεσολαβητής ανάμεσα σε model, view, scripts και οποιαδήποτε άλλη πηγή χρειάζεται για να παραχθεί η απάντηση στο HttpRequest.



Εικόνα 13: Ροή δεδομένων σε σύστημα με τη χρήση του CodeIgniter framework

Στην Εικόνα 13 παρουσιάζεται η ροή των δεδομένων και τα στάδια επεξεργασίας τους από την είσοδο έως την έξοδο, με τη χρήση του CodeIgniter. Αναλυτικότερα:

- Η σελίδα index.php λειτουργεί ως controller, αρχικοποιώντας τις όποιες πηγές χρειάζονται για να λειτουργήσει το framework
- Ο router εξετάζει το HttpRequest που ελήφθη για να αποφασισθεί τι πρέπει να γίνει ακριβώς.
- Αν υπάρχει cache αρχείο, αποστέλλεται κατευθείαν στον χρήστη μέσω του browser παρακάμπτοντας την υπόλοιπη διαδικασία
- Πριν κληθεί ο controller της εφαρμογής, ελέγχεται το αίτημα για λόγους ασφαλείας.
- Ο controller φορτώνει το model, τις βιβλιοθήκες και ό,τι άλλο απαιτείται.
- Τέλος, το view αποστέλλεται στον browser, έτοιμο για χρήση.

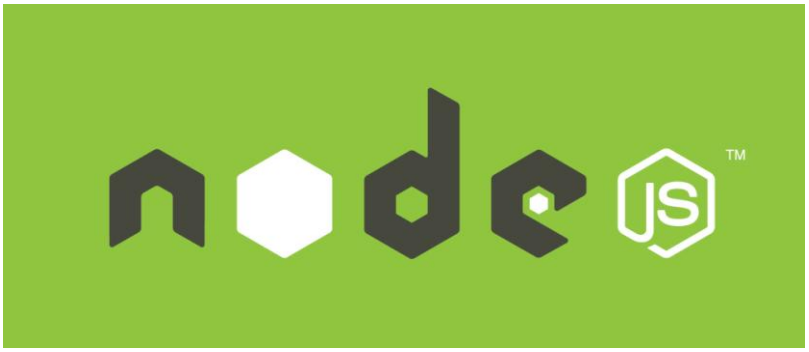
Εγκατάσταση

1. Αφού κατεβάσουμε την έκδοση του CodeIgniter που επιθυμούμε, κάνουμε unzip τα αρχεία
2. Τα ανεβάζουμε στον server σε φάκελο που υπάρχει πρόσβαση εξωτερικά μέσω browser, πχ. Public_html ή var/www, ανάλογα και με τον web server που έχουμε στήσει.
3. Στο αρχείο application/config/config.php ορίζουμε τη διεύθυνση URL που επιθυμούμε να έχει η εφαρμογή μας.
4. Στο αρχείο application/config/database.php ορίζουμε,αν έχουμε σκοπό να χρησιμοποιήσουμε, τα στοιχεία της βάσης δεδομένων μας.

Αφού ολοκληρώσουμε τα παραπάνω βήματα, η εγκατάσταση έχει ολοκληρωθεί και μπορούμε να αρχίσουμε την υλοποίηση της εφαρμογής μας με την κατάλληλη τροποποίηση των αρχείων μας.

4.2.2 Hybrid Mobile App

Σε αυτή την ενότητα παρουσιάζονται τα framework Apache Cordova, Ionic και AngularJS καθώς και τα προαπαιτούμενά τους Node.js και Git όπως χρησιμοποιήθηκαν κατά την υλοποίηση της υβριδικής εφαρμογής.



4.2.2.1 Node.js

Γενικά

Το Node.js [29] είναι μια, ανοιχτού κώδικα, πλατφόρμα ανάπτυξης λογισμικού βασισμένη σε Javascript για χρήση κυρίως σε διαδικτυακές διαδραστικές εφαρμογές. Χτίστηκε πάνω στη Chrome's V8 JavaScript engine της Google [29], που αποτελεί μια ανοιχτού κώδικα μηχανή Javascript, γραμμένη σε C++ που λόγω ταχύτητας χρησιμοποιείται και από τον Google Chrome Browser [30]. Το node.js δημιουργήθηκε το 2009 από τον Ryan Dahl και το 2011 συστάθηκε ο, όπως εξελίχθηκε σήμερα, μεγαλύτερος στο κόσμο διαχειριστής ανοιχτών βιβλιοθηκών κώδικα Javascript ή αλλιώς npm [29]. Με τη βοήθεια του npm, ο προγραμματιστής έχει τη δυνατότητα να διαχειριστεί τις βιβλιοθήκες που χρειάζεται η εφαρμογή του με μεγαλύτερη ευκολία, να εγκαταστήσει, να μοιράσει και να μοιραστεί κώδικα Javascript [31].

Εγκατάσταση

Η εγκατάστασή του είναι απαραίτητη για τα frameworks Cordova και Ionic που θα αναλύσουμε στην συνέχεια του κεφαλαίου 4. Αφού ολοκληρωθεί η λήψη του αρχείου από την επίσημη ιστοσελίδα (nodejs.org) και συγκεκριμένα για windows:

1. Εκτελούμε το αρχείο τύπου .msi που κατεβάσαμε
2. Δηλώνουμε ότι συμφωνούμε με τους όρους
3. Ορίζουμε που θέλουμε να εγκατασταθεί. Για παράδειγμα: C:\Program Files\nodejs\

4. Επιλέγουμε την εγκατάσταση όλων των στοιχείων, δηλαδή ‘node.js runtime’, ‘npm package manager’, ‘add to path’ και προαιρετικά το ‘online documentation shortcuts’.

Μόλις ολοκληρωθεί η εγκατάσταση είμαστε έτοιμοι να το χρησιμοποιήσουμε.

4.2.2.2 Git

Γενικά

Το Git [32] είναι ένα σύστημα ελέγχου εκδόσεων με έμφαση στην ταχύτητα και την ακεραιότητα των δεδομένων. Αναπτύχθηκε από τον Linus Torvalds το 2005, αρχικά για την ανάπτυξη του πυρήνα Linux. Έκτοτε έχει γίνει το πιο διαδεδομένο σύστημα ελέγχου εκδόσεων στην ανάπτυξη λογισμικού. [33] Όπως και τα Linux, έτσι και το Git διανέμεται ελεύθερα υπό τους όρους της GNU άδειας χρήσης.

Εγκατάσταση

Η εγκατάσταση του git είναι απαραίτητη αν εργαζόμαστε σε περιβάλλον Windows για τη χρήση των frameworks Cordova και Ionic που θα αναλύσουμε στην συνέχεια του κεφαλαίου 4. Αφού ολοκληρωθεί η λήψη του αρχείου από την επίσημη ιστοσελίδα (https://git-scm.com) και συγκεκριμένα για Windows:

1. Εκτελούμε το αρχείο τύπου .exe που κατεβάσαμε
2. Εγκαθιστούμε το Git με τις προτεινόμενες ρυθμίσεις.

Μόλις ολοκληρωθεί η εγκατάσταση είμαστε έτοιμοι να το χρησιμοποιήσουμε.

4.2.2.3 AngularJS

Γενικά

Η AngularJS [34] αποτελεί ένα framework κατάλληλο για τη δημιουργία δυναμικών διαδικτυακών εφαρμογών. Επιτρέπει στο χρήστη να χρησιμοποιήσει την HTML για τη δημιουργία του interface της εφαρμογής και παράλληλα την εμπλουτίζει με λειτουργίες που αφορούν το δυναμικό κομμάτι. Είναι επίσης σημαντικό να αναφερθεί ότι οι εκφράσεις της AngularJS σε μία σελίδα html γράφονται μέσα σε διπλές αγκύλες.

Στην επίσημη ιστοσελίδα της AngularJS αναφέρεται:

"*Angular is what HTML would have been,*

had it been designed for applications".

Βασικά συστατικά της AngularJS:

Scope

Το scope αποτελεί ένα αντικείμενο που μιμείται την DOM δομή της εφαρμογής και αφορά τη διασύνδεση μεταξύ των controllers και της διεπιφάνειας της εφαρμογής. Κάθε εφαρμογή Angular διατηρεί ένα μοναδικό rootScope αλλά μπορεί να έχει πολλαπλά childScopes. Σε ένα scope μπορεί να οριστούν διάφορες ιδιότητες καθώς και μέθοδοι για την χρησιμοποίησή τους στη συνέχεια στις HTML σελίδες της εφαρμογής δυναμικά.

Controllers

Ένας controller χρησιμοποιείται κυρίως για την αρχικοποίηση ενός scope αντικειμένου, την επεξεργασία του και τον ορισμό της συμπεριφοράς του, ενώ χρησιμοποιεί ένα αντικείμενο scope για τη χρήση των μεθόδων του στις HTML σελίδες.

Services

Services είναι αντικείμενα της Angular που επιτρέπουν την οργάνωση αλλά και τον διαμοιρασμό κοινού κώδικα για όλη την εφαρμογή. Η Angular διαθέτει αρκετά services όπως τα `$location` και `$http`, που χρησιμοποιήθηκαν στην υλοποίηση της παρούσας εργασίας και αναλύονται στην ενότητα

5.4.4.1 Services.js. Επιπλέον ο προγραμματιστής έχει τη δυνατότητα να δημιουργήσει δικά του.

Εγκατάσταση

Υλοποιόντας ένα Ionic app όπως στην παρούσα εργασία η AngularJS συμπεριλαμβάνεται αυτόματα στον φάκελο `lib/ng-cordova-master` του project και είναι έτοιμη για χρήση ενώ παράλληλα στο αρχείο `index.html` της εφαρμογής μας έχουν προστεθεί οι παρακάτω γραμμές κώδικα:

```
<!-- cordova script (this will be a 404 during development) -->

<script src="lib/ng-cordova-master/dist/ng-cordova.js">

</script> <script src="cordova.js"></script>
```

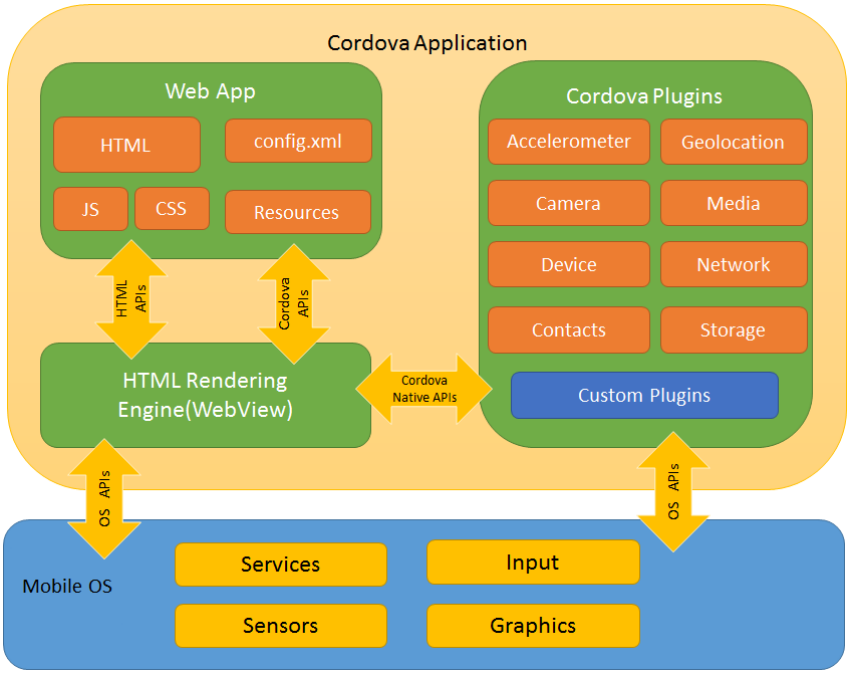
4.2.2.4 Apache Cordova

Γενικά

Η Apache Cordova [35] είναι ένα ανοιχτού κώδικα εργαλείο που επιτρέπει την ανάπτυξη εφαρμογών για κινητά τηλέφωνα χρησιμοποιώντας τις γνωστές τεχνολογίες του διαδικτύου HTML5, CSS και Javascript. Οι εφαρμογές που δημιουργούνται με τη χρήση της Apache Cordova στοχεύουν κάθε πλατφόρμα ξεχωριστά και στηρίζονται στο εκάστοτε API της

πλατφόρμας για πρόσβαση σε λειτουργίες της συσκευής όπως δεδομένα, αισθητήρες και άλλα. Τα λειτουργικά συστήματα που υποστηρίζονται μέχρι σήμερα είναι Android, Blackberry, iOS, Ubuntu και WindowsPhone. Για τη χρήση του είναι απαραίτητη η εγκατάσταση του 4.2.2.1 Node.js και του

Αρχιτεκτονική



Εικόνα 14: Αρχιτεκτονική εφαρμογής Apache Cordova

Μία εφαρμογή δημιουργημένη με τη χρήση του Apache Cordova framework, χωρίζεται σε 3 τομείς και ακολουθεί αρχιτεκτονική όπως φαίνεται στην Εικόνα 14.

WebView

Το WebView παράσχει στην εφαρμογή το interface, δηλαδή ό,τι βλέπει ο χρήστης στην οθόνη του κινητού ή tablet του. Σε κάποιες πλατφόρμες όμως, όπως το android, είναι δυνατό να χρησιμοποιηθεί ως συστατικό (Component) της εφαρμογής μαζί με άλλα συστατικά της ίδιας της πλατφόρμας.

Web App

Το Web App περιέχει όλο το κώδικα της εφαρμογής και τρέχει σε WebView μέσα σε ένα Native application wrapper.

Όπως φαίνεται και από την Εικόνα 14, το Web App περιέχει ένα αρχείο index.html στο οποίο δηλώνονται όλα τα απαραίτητα αρχεία όπως CSS, Javascript, εικόνες και οτιδήποτε άλλο απαραίτητο για την λειτουργία της εφαρμογής. Στα js αρχεία της παρούσας εργασίας περιλαμβάνονται τα controller.js, app.js και service.js τα οποία αναλύονται στο

5.4.4 JS αρχεία.

Ακόμη ένα πολύ σημαντικό αρχείο είναι το config.xml, στο οποίο δηλώνονται σημαντικές παράμετροι που επηρεάζουν την λειτουργία και την εμφάνιση της εφαρμογής. Όλα τα παραπάνω αρχεία βρίσκονται στον φάκελο www ενός apache cordova project.

Cordova Plugins

Τα plugins αποτελούν σημαντικό κομμάτι μιας εφαρμογής Cordova αφού μπορούν να παρέχουν interface αλλά και σύνδεση μεταξύ των συστατικών ή και του API της πλατφόρμας της συσκευής. Το ίδιο το project της Apache Cordova διατηρεί μία ομάδα plugins με όνομα "Core plugins", τα οποία επιτρέπουν πρόσβαση σε λειτουργίες της συσκευής όπως η κάμερα, το GPS, η μπαταρία και άλλα. Μερικά από αυτά χρησιμοποιήθηκαν και κατά την υλοποίηση της παρούσας εργασίας όπως θα δούμε στο 5.4.2 Cordova plugins. Πέραν όμως των core plugins, υπάρχει λειτουργία αναζήτησης περαιτέρω plugins μέσω του npm (4.2.2.1 Node.js) ή μέσω του επίσημου site της Apache Cordova (https://cordova.apache.org/plugins/). Τέλος υπάρχει δυνατότητα δημιουργίας νέων.

Τρόποι ανάπτυξης Apache Cordova εφαρμογής

Υπάρχουν δύο τρόποι ανάπτυξης εφαρμογής με τη χρήση της Cordova, ο πρώτος ονομάζεται Cross-platform workflow και ο δεύτερος Platform-centered workflow.

Cross-platform workflow ή αλλιώς **CLI**. Ο προγραμματιστής εργάζεται γύρω από τη Cordova με πολύ μικρή ανάμειξη σε χαμηλότερου επιπέδου ανάπτυξη κώδικα. Αυτός ο τρόπος εργασίας επιλέγεται συνήθως όταν ενδιαφερόμαστε να δημιουργήσουμε μια εφαρμογή που θα μπορεί να χρησιμοποιηθεί σε πολλές διαφορετικές πλατφόρμες. Το CLI αποτελεί ένα εργαλείο υψηλού επιπέδου που αναλαμβάνει να μετατρέψει το κώδικα της εφαρμογής μας σε εφαρμογή στοχευμένη στη πλατφόρμα που θα επιλέξουμε, κάνοντας τις απαραίτητες μετατροπές αυτοματοποιημένα, λύνοντας έτσι τα χέρια του προγραμματιστή. Αυτός ο τρόπος εργασίας είναι και ο προτεινόμενος εκτός και αν υπάρχει ανάγκη να εργαστούμε στοχεύοντας αποκλειστικά μια συγκεκριμένη πλατφόρμα και να επεξεργαστούμε την εφαρμογή σε χαμηλότερο επίπεδο. Τότε θα επιλεγεί το Platform-centered workflow που αναλύεται παρακάτω.

Platform-centered workflow. Ο προγραμματιστής εργάζεται εστιάζοντας αποκλειστικά σε μία πλατφόρμα και έχει πλέον τη δυνατότητα να επεξεργαστεί την εφαρμογή σε χαμηλότερο επίπεδο, μέσα στο SDK. Αν λοιπόν έχουμε σκοπό να χρησιμοποιήσουμε στοιχεία της Cordova

μαζί με φυσικά συστατικά της εκάστοτε πλατφόρμας πρέπει να επιλέξουμε το συγκεκριμένο τρόπο εργασίας. Αν και μπορεί να χρησιμοποιηθεί και για δημιουργία εφαρμογών σε περισσότερες από μία πλατφόρμα δεν συστήνεται καθώς θα πρέπει να γίνουν μετατροπές για κάθε μία ξεχωριστά.

Ξεκινώντας μία νέα εφαρμογή με τη χρήση της Cordova, ο προγραμματιστής μπορεί να επιλέξει το CLI, που είναι και το προτεινόμενο και αν κρίνει ότι χρειάζεται τη μεγαλύτερη ευχέρεια που προσφέρει το SDK μπορεί να αλλάξει σε Platform-centered εργαλεία και τρόπο ανάπτυξης. Είναι σημαντικό ωστόσο να αναφερθεί ότι αν γίνει η αλλαγή σε Platform-centered, μετά δεν μπορεί να επιστρέψει σε CLI. Το CLI διατηρεί ένα κοινό αρχείο κώδικα για όλες τις πλατφόρμες, το οποίο χρησιμοποιείται κατά τη διαδικασία build της εφαρμογής για τη δημιουργία του κώδικα της κάθε πλατφόρμας. Αν στη συνέχεια θέλει ο χρήστης να επεξεργαστεί το κώδικα της κάθε πλατφόρμας ξεχωριστά σε χαμηλότερο επίπεδο, δηλαδή SDK και οι αλλαγές του να διατηρηθούν θα πρέπει να χρησιμοποιήσει τα εργαλεία που προσφέρονται για Platform-centered τρόπο εργασίας, τα οποία αγνοούν το κοινό κώδικα και επικεντρώνονται στο κώδικα της συγκεκριμένης πλατφόρμας.

Στην παρούσα εργασία λαμβάνοντας υπόψη τα παραπάνω, επιλέξαμε να υλοποιήσουμε την εφαρμογή πελάτη με τον Cross-platform workflow τρόπο ανάπτυξης.

Εγκατάσταση

Η εγκατάσταση της Cordova προυποθέτει την εγκατάσταση των node.js και git. Αφού λοιπόν εγκατασταθούν, με τη χρήση του npm μπορούμε με μία εντολή να εγκαταστήσουμε την Cordova.

Windows: `npm install -g cordova`

OSX/Linux: `sudo npm install -g cordova`

Ανάπτυξη εφαρμογής για Android

Για την ανάπτυξη εφαρμογής για το λειτουργικό σύστημα android ***[36]***, εκτός της εγκατάστασης της Apache Cordova και των προαπαιτούμενών της, απαιτείται η εγκατάσταση των εργαλείων:

- Java Development Kit (JDK) 7 [5] ή νεότερου.

[5] Java Development Kit (JDK) 7 . [20 Ιουνίου 2016.]
http://www.oracle.com/technetwork/java/javase/overview/index.html

- Android Stand-alone SDK Tools [6] ή Android Studio.

Τέλος, πρέπει να δηλώσουμε τις μεταβλητές συστήματος JAVA_HOME ως την τοποθεσία εγκατάστασης του Java Development Kit (JDK) και ANDROID_HOME ως την τοποθεσία εγκατάστασης του ANDROID SDK. Στην μεταβλητή PATH πρέπει να προσθέσουμε τους φακέλους tools και platform-tools του ANDROID SDK.

Συγκεκριμένα σε λειτουργικό WINDOWS:

1. Δεξί κλικ στο εικονίδιο 'Ο υπολογιστής μου', επιλέγουμε 'Ιδιότητες'
2. Επιλέγουμε 'Ρυθμίσεις για προχωρημένους' > 'Μεταβλητές συστήματος'
3. Πραγματοποιούμε τις αλλαγές που απαιτούνται

Έχοντας εγκαταστήσει όλα τα προαπαιτούμενα και αφού προσθέσουμε την πλατφόρμα android, για την εκτέλεση της λειτουργίας build αρκεί μόλις μία εντολή στη βάση του project μας και το αρχείο .apk της εφαρμογής μας είναι έτοιμο.

Προσθήκη πλατφόρμας android: `cordova platform add android`

Build εφαρμογής για την πλατφόρμα android: `cordova build android`

Ανάπτυξη εφαρμογής για iOS

Για την ανάπτυξη εφαρμογής για το λειτουργικό iOS [37], εκτός της εγκατάστασης της Apache Cordova και των προαπαιτούμενών της, απαιτείται η εγκατάσταση του εργαλείου:

- Xcode.

Το συγκεκριμένο πρόγραμμα της Apple είναι συμβατό μόνο με συστήματα Mac βασισμένα σε Intel με λειτουργικό σύστημα OS X.

Υπάρχουν δύο τρόποι εγκατάστασης του λογισμικού Xcode:

- Από το App Store της Apple, αναζητούμε 'Xcode'.
- Από την σελίδα Apple Developer Downloads[7], όπου απαιτείται λογαριασμός επιβεβαιωμένος ως προγραμματιστής Apple.

---

[6] Android studio . [20 Ιουνίου 2016.] https://developer.android.com/studio/index.html

Αφού τελειώσει η μεταφόρτωση και εγκατασταθεί, ανοίγουμε το τερματικό και εκτελούμε τις παρακάτω εντολές:

Εγκατάσταση και ενεργοποίηση χρήσιμων εντολών για την Apache Cordova:

```
xcode-select –install
```

Οι παρακάτω εντολές επιτρέπουν την φόρτωση μέσω του τερματικού της iOS εφαρμογής στον simulator και στη συσκευή αντίστοιχα:

```
npm install -g ios-sim
```

```
npm install -g ios-deploy
```

Έχοντας εγκαταστήσει όλα τα προαπαιτούμενα και αφού προσθέσουμε την πλατφόρμα iOS, για την εκτέλεση της λειτουργίας build αρκεί μόλις μία εντολή στη βάση του project μας και δημιουργούνται τα κατάλληλα αρχεία.

Προσθήκη πλατφόρμας ios: `cordova platform add ios`

Build εφαρμογής για την πλατφόρμα ios: `cordova build ios`

4.2.2.5 IONIC Framework

Γενικά

Το Ionic [38] αποτελεί ένα εργαλείο ανοιχτού κώδικα, δημιουργίας HTML5 εφαρμογών έξυπνων κινητών τηλεφώνων. Η άδεια MIT [39] υπό την οποία τελεί, επιτρέπει τη χρήση του Ionic τόσο σε προσωπική όσο και εμπορική δωρεάν χρήση. Δημιουργήθηκε το 2013 από τους Max Lynch, Ben Sperry, and Adam Bradley of Drifty, υποστηρίζει τη χρήση HTML5, CSS,

---

[7] https://developer.apple.com/download

Javascript και αναλαμβάνει το front-end μιας υβριδικής εφαρμογής. Δεν αντικαθιστά σε καμία περίπτωση την Apache Cordova αλλά την ενισχύει.

Εγκατάσταση

Απαραίτητη είναι η εγκατάσταση του node.js και της Apache Cordova ενώ συστήνεται και η χρησιμοποίηση της AngularJS για πλήρη αξιοποίηση των δυνατοτήτων του. Με τη χρήση του npm εγκαθίσταται το ionic με μία εντολή

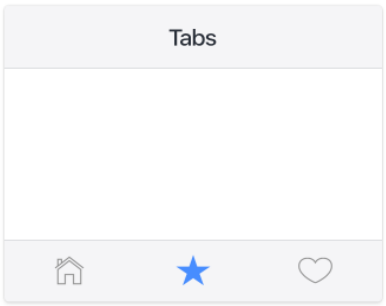
Winodws: `npm install -g ionic`

OSX/Linux: `sudo npm install -g ionic`

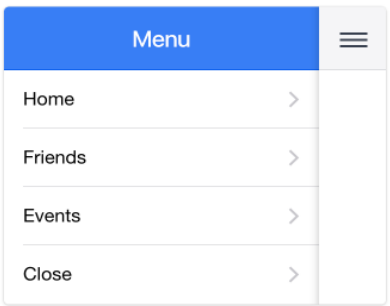
Δημιουργία Ionic project

Το Ionic framework παρέχει τη δυνατότητα δημιουργίας εφαρμογής από το μηδέν αλλά παράλληλα παράσχει και λύσεις με υλοποιημένες κάποιες λειτουργίες όπως μενού ή tabs. Για την δημιουργία του project από τη γραμμή εντολών του υπολογιστή μας τρέχουμε μία από τις παρακάτω εντολές.

`ionic start myApp blank`



`ionic start myApp tabs`



`ionic start myApp sidemenu`

Στη συνέχεια μπορούμε να ανοίξουμε και να επεξεργαστούμε το project μας σαν ένα HTML5 Application στον IDE της επιλογής μας.

Κεφάλαιο 5

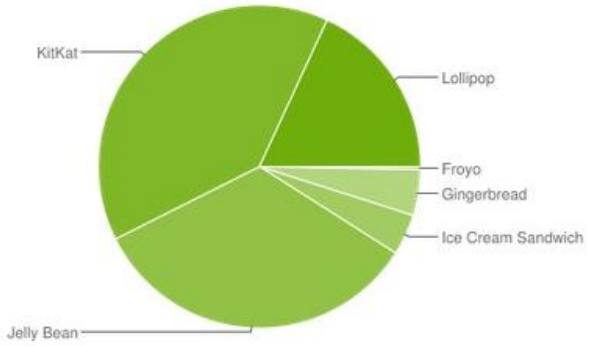
Παρουσίαση εφαρμογής

Γενικά

Όπως έχουμε ήδη αναφέρει η εφαρμογή υλοποιήθηκε με βασικό στόχο την υποστήριξη όσο το δυνατόν μεγαλύτερου μέρους των χρηστών smartphone. Για το λόγο αυτό, σύμφωνα και με την τα οποία για το 2015 κατείχαν το 96,7% της παγκόσμιας αγοράς [2]. αποφασίστηκε η υλοποίηση της εφαρμογής για τα λειτουργικά συστήματα Android και Ios, τα οποία για το 2015 κατείχαν το 96,7% της παγκόσμιας αγοράς [2]. Πιο συγκεκριμένα η εφαρμογή μας όσον αφορά το λειτουργικό Android, λειτουργεί σε συσκευές με εγκατεστημένο Android API 14 και πάνω, ενώ στοχεύει στις συσκευές επιπέδου Android API 19. Η απόφαση βασίστηκε στο γεγονός ότι τον Αύγουστο του 2015 η έκδοση KitKat (API 19) χρησιμοποιούταν από σχεδόν το 40% των χρηστών Android [40].

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	4.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	4.1%
4.1.x	Jelly Bean	16	13.0%
4.2.x		17	15.9%
4.3		18	4.7%
4.4	KitKat	19	39.3%
5.0	Lollipop	21	15.5%
5.1		22	2.6%

Data collected during a 7-day period ending on August 3, 2015.
Any versions with less than 0.1% distribution are not shown.

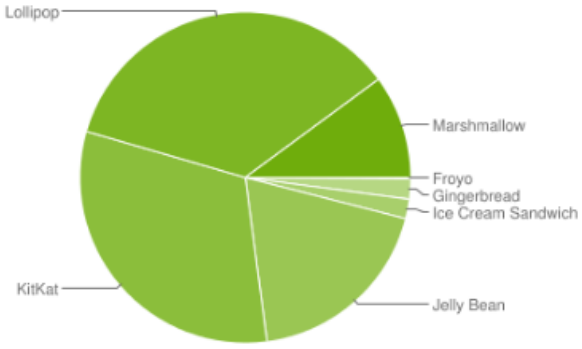


Εικόνα 15: Στατιστικά χρήσης εκδόσεων Android, τον Αύγουστο του 2015 [40]

Σήμερα (Ιούλιος 2016) η έκδοση KitKat εξακολουθεί να διατηρεί τα σκήπτρα με 31,6% όπως φαίνεται και από την Εικόνα 16, ενώ ακολουθεί η έκδοση Lollipop, η οποία συνολικά κατέχει το 35,4%, όμως διαχωρίζεται σε API 21 και API 22.

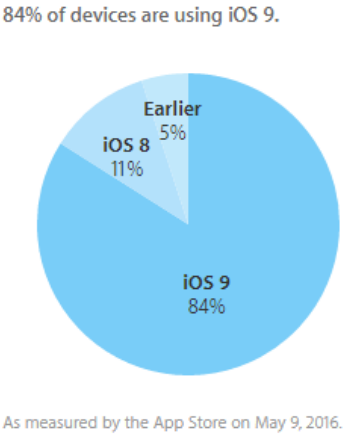
Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.9%
4.1.x	Jelly Bean	16	6.8%
4.2.x		17	9.4%
4.3		18	2.7%
4.4	KitKat	19	31.6%
5.0	Lollipop	21	15.4%
5.1		22	20.0%
6.0	Marshmallow	23	10.1%

Data collected during a 7-day period ending on June 6, 2016.
Any versions with less than 0.1% distribution are not shown.



Εικόνα 16: Στατιστικά χρήσης εκδόσεων Android, τον Ιούλιο του 2016 [41]

Όσον αφορά το λειτουργικό σύστημα iOS, τον Μάιο του 2016 η έκδοση iOS 9 χρησιμοποιείται από το 84% των χρηστών (Εικόνα 17). Για να καλύψουμε λοιπόν το μεγαλύτερο μέρος των χρηστών, αποφασίσαμε να μην υλοποιήσουμε κάποιες λειτουργίες για την έκδοση iOS αφού δεν υποστηρίζονταν από το λειτουργικό σύστημα. Οι λειτουργίες αφορούν την απευθείας μετάβαση του χρήστη στις ρυθμίσεις του κινητού του όταν το GPS είναι απενεργοποιημένο ή η σύνδεση στο διαδίκτυο αδύνατη.



Εικόνα 17: Στατιστικά χρήσης εκδόσεων iOS τον Μάιο του 2016 [42]

Στη σύνεχεια του κεφαλαίου 5 αναλύεται το σύστημα που υλοποιήθηκε, ξεκινώντας από την βάση δεδομένων και τον τρόπο με τον οποίο εισάγονται τα δεδομένα, συνεχίζοντας με την υλοποίηση της RESTful υπηρεσίας και τέλος την υβριδική εφαρμογή-πελάτη.

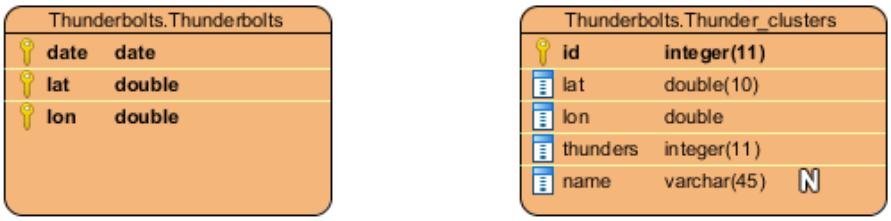
5.1 Βάση δεδομένων MySQL

Η MySQL βρίσκεται εγκατεστημένη όπως αναφέρθηκε στην ενότητα 3.3 Αρχιτεκτονική συστήματος στην διεύθυνση http://195.251.31.119 και πόρτα 3306. Δημιουργήσαμε μία νέα βάση δεδομένων με όνομα Thunderbolts, η οποία περιλαμβάνει δύο πίνακες, Thunderbolts και Thunder_clusters.

Επιλέγουμε character set **utf8** και collation **utf8_general_ci**, ώστε η βάση μας να δέχεται ειδικούς χαρακτήρες από διαφορετικές γλώσσες. Αυτό είναι σημαντικό, καθώς κατά την ομαδοποίηση αν το κέντρο μιας ομάδας κεραυνών βρίσκεται για παράδειγμα στο χώρο της Τουρκίας, το όνομα πιθανότατα δεν θα αποθηκευόταν σωστά.

Ακόμα δημιουργούμε ένα Event με όνομα `Clean_Older_Than_30_days_thunderbolts`, το οποίο εκτελείτε μία φορά την ημέρα και χρησιμεύει στην εκκαθάριση δεδομένων με ημερομηνία παλαιότερη από 30 μέρες.

```
CREATE EVENT IF NOT EXISTS `Clean_Older_Than_30_days_thunderbolts`
ON SCHEDULE
  EVERY 1 DAY_HOUR
  DO
    DELETE FROM Thunderbolts
    WHERE date < DATE_SUB(NOW(), INTERVAL 30 DAY);
```



Εικόνα 18: ER διάγραμμα βάσης δεδομένων

Αναλυτικότερα οι πίνακες που δημιουργήθηκαν για την εφαρμογή όπως φαίνονται και στην Εικόνα 18 είναι οι εξής:

1. **Thunderbolts**

   Πίνακας αποθήκευσης των χαρακτηριστικών κάθε κεραυνού που εισάγεται στην βάση δεδομένων

   Πεδία:

   - date: DATETIME

     Η χρονική καταγραφή του κεραυνού όπως μετατράπηκε από msec σε ημερομηνία και ώρα
   - lat: DOUBLE

     Το γεωγραφικό πλάτο στο οποίο καταγράφηκε ο κεραυνός
   - lon: DOUBLE

     Το γεωγραφικό μήκος στο οποίο καταγράφηκε ο κεραυνός
2. **Thunder_clusters**

   Πίνακας αποθήκευσης των ομαδοποιημένων κεραυνών.

   Πεδία:

   - id: INT(11) PRIMARY KEY

     Μοναδικός αριθμός εγγραφής ομάδας
   - lat: DOUBLE

     Η μέση τιμή γεωγραφικού πλάτους της ομάδας
   - lon: DOUBLE

     Η μέση τιμή γεωγραφικού μήκους της ομάδας
   - thunders: INT(11)

     Ο αριθμός κεραυνών που ανήκουν στην ομάδα
   - name: VARCHAR(45)

     Το όνομα της περιοχής στην οποία βρίσκεται το γεωγραφικό μήκος και πλάτος της ομάδας.

5.2 Εισαγωγή των δεδομένων στη βάση δεδομένων

Για την αδιάλειπτη εισαγωγή και επεξεργασία των σημείων-κεραυνών στη βάση δεδομένων υλοποιήθηκαν κατάλληλα scripts στον server τα οποία εκτελούνται ανά τακτά χρονικά διαστήματα με τη χρήση τριών crontabs.

Με τη χρήση των παρακάτω εντολών ορίζουμε ως editor τον nano και στη συνέχεια μπορούμε να επεξεργαστούμε τα crontabs.

```
EDITOR=nano
export EDITOR
crontab –e
```

Η βασική μορφή ενός crontab είναι η εξής:

```
minute hour day month day-of-week command-line-to-execute
```

Τα crontabs που υλοποιήθηκαν είναι τα παρακάτω:

1ο Crontab:

```
14,29,44,59 * * * * /home/strikes/scripts/nex2txt.sh >>
/home/strikes/scripts/log
```

Τα δεδομένα των κεραυνών που βρίσκονται αρχικά σε αρχείο τύπου .nex μετατρέπονται σε τύπου .txt κάθε ώρα στις xx:05,20,35,50. Το script παραδόθηκε από τον κ. Κατσαφάδο Πέτρο. Προστέθηκε μία επιπλέον εντολή στο τέλος που αποσκοπεί στην εκτέλεση του script sqlimport.sh που παρουσιάζεται και αναλύεται παρακάτω.

```
./sqlimport.sh >> /home/strikes/scripts/log
```

Η εκτέλεση του `sqlimport.sh` μετατρέπει το .txt αρχείο με τους κεραυνούς σε .csv τροποποιώντας το κατάλληλα και διαγράφει όλες τις στήλες δεδομένων που δεν μας ενδιαφέρουν κρατώντας μόλις τρεις: ημερομηνία, γεωγραφικό μήκος και πλάτος. Τέλος εισάγει τα δεδομένα στον πίνακα Thunderbolts στη βάση δεδομένων.

Κώδικας `sqlimport.sh` και ανάλυση:

```
#!/bin/bash
date
cd
cd ./hua
filename="$(date -d '3 hour ago' +"%Y%m%d")"
```

Μετατροπή του αρχείου .txt σε .csv διαγράφοντας τα επιπλέον κενά ανάμεσα στα πεδία:

```
sed 1d $filename.txt > Thund.csv
sed 's/  */ /g' Thund.csv > Thunderstmp.csv
sed -i 's/^ *//' Thunderstmp.csv
awk 'NF>=13' Thunderstmp.csv > Thunderbolts.csv
```

Μετατροπή της στήλης seconds σε μορφή ημερομηνίας YY-MM-DD HH:MM:SS:

```
awk -v date="$(date -d '3 hour ago' +"%Y-%m-%d")" '$1=date"
"int($1/3600)":"int($1%3600/60)":"$1%60' Thunderbolts.csv > Thund.csv
```

Διαχωρισμός των στηλών με ‘,’

```
tr ' ' ',' <Thund.csv >Thundercommas.csv
sed 's/,/ /' Thundercommas.csv > ThunderboltsAll.csv
```

Διαγραφή όλων των στηλών εκτός της ημερομηνίας, γεωγραφικού μήκους και πλάτους:

```
cut -f 1,10,11 -d, ThunderboltsAll.csv > Thunderbolts.csv
```

Εισαγωγή στην βάση δεδομένων στον πίνακα Thunderbolts:

```
mysqlimport --fields-terminated-by=',' --local -u username –ppassword
Thunderbolts Thunderbolts.csv
```

Διαγραφή όσων αρχείων δημιουργήθηκαν κατά την εκτέλεση του script:

```
rm Thund.csv
rm Thunderstmp.csv
rm Thundercommas.csv
rm ThunderboltsAll.csv
```

2ο Crontab:

```
52 * * * * /home/strikes/scripts/dbscansqlimport.sh >>
/home/strikes/scripts/log
```

Το script `dbscansqlimport.sh` εκτελείται μία φορά κάθε ώρα στις xx:52 και είναι υπεύθυνο για την εκτέλεση του αλγορίθμου DBScan για την ομαδοποίηση των κεραυνών-σημείων και την εισαγωγή αυτών στον πίνακα Thunder_clusters.

Κώδικας `dbscansqlimport.sh` και ανάλυση:

```
cd
cd ./scripts
date >> clustersLog.txt
```

Η παρακάτω εντολή καλεί την συνάρτηση index() του ClustersController.php και το αποτέλεσμα εκτυπώνεται στο αρχείο clustersOutput.js. Πρόκειται για ένα script που δίνει ως είσοδο στον αλγόριθμο DBScan τα δεδομένα των κεραυνών των τελευταίων 24 ωρών, εκτελεί τον αλγόριθμο και εισάγει τα αποτελέσματα στον πίνακα Thunder_clusters της βάσης.

```
php /home/strikes/public_html/api/index.php ClustersController >
clustersOutput.js
```

Τέλος με την επόμενη εντολή, το script που εκτυπώθηκε στο αρχείο clustersOutput.js εκτελείται με τη χρήση του rhino-1.7.7.1.jar για server-side Javascript. Το αποτέλεσμα της εκτέλεσης του script clustersOutput.js καταγράφεται στο αρχείο clustersLog.txt.

```
java -jar rhino-1.7.7.1.jar -opt -1 clustersOutput.js >> clustersLog.txt
```

Παράδειγμα κώδικα αρχείου clustersOutput.js:

```
load('env.rhino.1.2.js'); //load environment to run javascript on server

load('jquery-1.12.3.min.js'); //load jquery to post data

window.location = 'some.html'; //to use the environment, we need a simple html page

load('/home/strikes/public_html/api/application/third_party/jDBSCAN.js');

<script>

var points =[{"date":"2016-06-15
05:54:43","lat":"44.271","lon":"27.475"},{"date":"2016-06-15
05:55:06","lat":"43.219","lon":"25.097"},{"date":"2016-06-15
05:56:54","lat":"42.92","lon":"27.982"},{"date":"2016-06-15
05:57:57","lat":"44.501","lon":"24.913"},{"date":"2016-06-15
05:58:51","lat":"37.413","lon":"32.264"},{"date":"2016-06-15
06:00:53","lat":"38.764","lon":"30.367"},{"date":"2016-06-15
06:02:07","lat":"44.848","lon":"25.843"},{"date":"2016-06-15
06:02:42","lat":"44.234","lon":"26.596"}}];

var dbscanner = jDBSCAN().data(points); //run DBSCAN

var point_assignment_result = dbscanner();

var cluster_centers = dbscanner.getClusters();

console.log("successed jDBSCAN, Number of clusters: "+cluster_centers.length);

jQuery.support.cors = true; //enable cross origin for server usage

$(document).ready(function(){

$.ajax({

type: 'POST',

url:
'http://195.251.31.119/~strikes/api/index.php/ClustersController/insert_clusters',

data: {clusters : cluster_centers},

success: function(data){

console.log("successed POST");

},error: function(xhr, textStatus, error){

console.log(xhr.statusText);

console.log(textStatus);

console.log(error);

}

});

});
```

3ο Crontab:

```
59 23 * * * /home/strikes/scripts/midnightcopyfile.sh
```

Το 3ο και τελευταίο crontab που υλοποιήθηκε για την παρούσα εργασία αφορά την δημιουργία δύο αρχείων στον server με τα δεδομένα των κεραυνών της ημέρας σε μορφή .txt και .csv. Εκτελείται κάθε βράδυ στις 23:59.

Κώδικας `midnightcopyfile.sh` και ανάλυση:

```
#!/bin/bash
date="$(date +"%Y%m%d")"
cd
cd ./hua
cp $date.txt $date+midnightcopy.txt
unlink $date.txt
cp Thunderbolts.csv $date+Thunderbolts.csv
```

Αναλυτικότερα οι τρεις τελευταίες εντολές αποσκοπούν αντίστοιχα στα εξής:

Αντιγραφή του αρχείου .txt με τα δεδομένα σε δεύτερο αρχείο με όνομα ημερομηνία+`midnightcopy.txt`

Διαγραφή του αρχικού .txt αρχείου

Αντιγραφή του τροποποιημένου `Thunderbolts.csv` αρχείου σε αρχείο με όνομα ημερομηνία+ `midnightcopy.csv` ώστε το `Thunderbolts.csv` να χρησιμοποιηθεί εκ νέου για τους κεραυνούς της επόμενης ημέρας.

5.3 Rest API υπηρεσία

Με τη βοήθεια του CodeIgniter Framework δημιουργήσαμε μία RESTful υπηρεσία με στόχο την αποστολή δεδομένων από την βάση δεδομένων στην εφαρμογή-πελάτη, σε μορφή JSON και πιο συγκεκριμένα GeoJSON.

Οι λειτουργίες που έχουν υλοποιηθεί είναι οι εξής:

1. Ανάκτηση δεδομένων των κεραυνών
2. Ανάκτηση δεδομένων ομαδοποιημένων κεραυνών
3. Εκτέλεση DBScan για την ομαδοποίηση των κεραυνών

5.3.1 Controllers

Για τις ανάγκες της εργασίας δημιουργήθηκαν δύο αρχεία controllers ThunderboltsController.php και ClustersController.php.

ThunderboltsController.php

Το αρχείο ThunderboltsController.php υλοποιήθηκε για την ανάκτηση και μεταφορά των δεδομένων των κεραυνών στην εφαρμογή-πελάτη. Περιλαμβάνει μία συνάρτηση:

`index()`, η οποία φορτώνει το κατάλληλο model και view για την ανάκτηση και προβολή των δεδομένων από την βάση. Χρησιμοποιείται από την εφαρμογή-πελάτη.

ClustersController.php

Το αρχείο ClustersController.php υλοποιήθηκε για την ομαδοποίηση των κεραυνών καθώς και την ανάκτηση στη συνέχεια των αποτελεσμάτων και προβολή τους στην εφαρμογή-πελάτη. Περιλαμβάνει τρεις συναρτήσεις:

`index()`, φορτώνει το κατάλληλο model και καλεί για την εκτέλεση του αλγορίθμου DBScan. Καλείται μέσω cronjob ανά μία ώρα στον server όπως αναλύθηκε προηγουμένως στην ενότητα 5.2 Εισαγωγή των δεδομένων στη βάση δεδομένων.

`insert_clusters()`, φορτώνει το κατάλληλο model και καλεί για την εισαγωγή των ομαδοποιημένων δεδομένων στη βάση. Καλείται μέσω της **`index()`** από τον server.

`getTop()`, φορτώνει το κατάλληλο model για την ανάκτηση και προβολή των ομαδοποιημένων δεδομένων. Χρησιμοποιείται από την εφαρμογή-πελάτη.

5.3.2 Models

Στο αρχείο ThunderboltsModel.php περιλαμβάνονται οι παρακάτω συναρτήσεις:

`get_thunderbolts()`, αφορά την ανάκτηση των κεραυνών με βάση τις GET παραμέτρους που δόθηκαν από την εφαρμογή. Οι παράμετροι `$hours`, `$nelat`, `$nelng`, `$swlat`, `$swlng` είναι προαιρετικοί και αφορούν τις τελευταίες ώρες για τις οποίες θέλουμε να ανακτήσουμε δεδομένα καθώς επίσης και για ποιο συγκεκριμένο τμήμα του χάρτη. Πιο συγκεκριμένα δηλώνουμε τις συντεταγμένες της βορειοανατολικής και νοτιοδυτικής γωνίας του χάρτη που είναι ορατός στον χρήστη.

reverse_geocode($cluster,$result_type), καλεί το Google Maps Geocoding API

```
(marker = new google.maps.Marker({
      map: map,
      animation: google.maps.Animation.DROP,
      position: latLng,
      icon: './img/marker.png'
});
```

Events: Η Javascript εντός του browser οδηγείται από events, δηλαδή για την αλληλεπίδραση με το χρήστη και την εφαρμογή αντιδρά στα events που δημιουργούνται. Τα events χωρίζονται σε User events τα οποία σχετίζονται με τον χρήστη (για παράδειγμα 'click') και MVC state change notifications τα οποία αντικατοπτρίζουν αλλαγές στο Javascript API (για παράδειγμα 'zoom_changed'). Για να παρέμβουμε και να εκτελέσουμε κώδικα όταν συμβαίνει κάποιο event μπορεί να χρησιμοποιηθεί η συνάρτηση addListener().

Παράδειγμα:

```
//User event
marker.addListener('click', function() {
      map.setZoom(8);
      map.setCenter(marker.getPosition());
});


//MVC state change notifications
map.addListener('center_changed', function() {
      // 3 seconds after the center of the map has changed, pan back to the
      // marker.
      window.setTimeout(function() {
            map.panTo(marker.getPosition());
      }, 3000);
});
```

) για την ανάκτηση ονόματος της περιοχής που ανήκει κάθε ομάδα που δημιουργείται από τον αλγόριθμο DBScan. Δέχεται ως ορίσματα τη μεταβλητή cluster που περιλαμβάνει τα δεδομένα για κάθε ομάδα καθώς και τη μεταβλητή result_type. Η μεταβλητή result_type μπορεί να ισούται

με "administrative_area_level_2" ή "natural_feature", η πρώτη περίπτωση ορίζει ως προτιμώμενη απάντηση το όνομα μιας περιοχής στη στεριά ενώ η δεύτερη στη θάλασσα.

`insert_clusters()`, δέχεται τα δεδομένα των ομάδων που δημιουργήθηκαν από τον αλγόριθμο DBScan μέσω της POST μεθόδου της HTTP. Στη συνέχεια μαζί με το όνομα που προκύπτει για κάθε μία ομάδα από την **`reverse_geocode()`** εισάγει τα δεδομένα στον πίνακα Thunder_clusters της βάσης δεδομένων.

`dbscan()`, ανακτά τους κεραυνούς μέσω της **`get_thunderbolts()`** και εκτυπώνει με τη χρήση της `echo` ένα script το οποίο καλεί: 1. τον αλγόριθμο DBScan για την ομαδοποίηση των δεδομένων και 2. την συνάρτηση **`insert_clusters()`** για την ανανέωση του πίνακα Thunder_clusters στη βάση.

`get_top_clusters()`, είναι υπεύθυνη για την ανάκτηση των αποθηκευμένων στη βάση ομάδων σε φθίνουσα ταξινόμηση. Μέσω της προαιρετικής GET παραμέτρου $number μπορούμε να δηλώσουμε πόσες ομάδες θέλουμε να ανακτήσουμε.

5.3.3 Views

Δημιουργήθηκαν δύο αρχεία ThunderboltsView.php και Top3View.php. Τα δύο αρχεία αποτελούν το τελικό αποτέλεσμα των HTTP GET αιτημάτων από την εφαρμογή-πελάτη.

ThunderboltsView.php, παρουσιάζει σε μορφή GeoJSON τους κεραυνούς που αντιστοιχούν στο αίτημα που δέχτηκε η RESTful υπηρεσία.

Παράδειγμα αιτήματος για το διάστημα της τελευταίας ώρας:

http://195.251.31.119/~strikes/api/index.php/ThunderboltsController?hours=1

Παράδειγμα κώδικα απάντηση:

```
{ "type": "FeatureCollection",
    "features": [
      { "type": "Feature",
        "geometry":{"type": "Point","coordinates": [32.563,
38.626]},"properties": {
          "date": "2016-06-16 04:33:30"
          }
        }, { "type": "Feature",
        "geometry":{"type": "Point","coordinates": [32.409,
38.489]},"properties": {
          "date": "2016-06-16 04:39:32"
          }
        }, { "type": "Feature",
```

```
        "geometry":{"type": "Point","coordinates": [33.374,
36.819]},"properties": {
          "date": "2016-06-16 05:04:44"
          }
        }]}
```


Top3View.php, παρουσιάζει σε μορφή GeoJSON τις ομάδες κεραυνών που αντιστοιχούν στο αίτημα που δέχτηκε η RESTful υπηρεσία.

Παράδειγμα αιτήματος για τρεις ομάδες κεραυνών:

http://195.251.31.119/~strikes/api/index.php/ClustersController/gettop?number=3

Παράδειγμα κώδικα απάντηση:


```
{ "type": "FeatureCollection",
    "features": [
      { "type": "Feature",
        "geometry":{"type":     "Point","coordinates":     [24.6550779510022,
42.6485946547884]},"properties": {
          "parts": "449","name": "Karlovo, Bulgaria"
          }
        }, { "type": "Feature",
        "geometry":{"type":     "Point","coordinates":     [31.4942826086956,
38.5138478260869]},"properties": {
          "parts": "46","name": "AkÅŸehir/Konya, Turkey"
          }
        }, { "type": "Feature",
        "geometry":{"type":     "Point","coordinates":     [15.8274888888889,
38.6299333333333]},"properties": {
          "parts": "45","name": "Contrada Punta Tono, 89866 Ricadi VV, Italy"
          }
        }]}
```

5.3.4 Third Party

5.3.4.1 Υλοποίηση DBScan Αλγορίθμου

Για την υλοποίηση του DBScan αλγορίθμου βασίστηκα στην υλοποίηση του Corneliu S. [43] σε γλώσσα javascript, σύμφωνα με τον ψευδοκώδικα[8] της αρχικής δημοσίευσής του. Έγιναν κατάλληλες τροποποιήσεις και προσθήκη κώδικα για να συμβαδίσει με τον τύπο του αρχείου εισόδου και τον αυτόματο υπολογισμό των παραμέτρων Eps και MinPTS του αλγορίθμου.

Το αρχείο jDBScan.js που βρίσκεται στον φάκελο third_party της RESTful υπηρεσίας περιλαμβάνει τις εξής συναρτήσεις:

`haversine_distance(point1, point2)`: Έτοιμη συνάρτηση, υπολογίζει την απόσταση haversine μεταξύ δύο σημείων-κεραυνών. Τροποποιήθηκε κατάλληλα ώστε να συμβαδίζει με τη μορφή GeoJSON των σημείων-κεραυνών. Ως ορίσματα δέχεται δύο μεταβλητές, τις point1 και point2 που περιέχουν αντίστοιχα τις συντεταγμένες των δύο σημείων που εξετάζει κάθε φορά που καλείται.

`expand_cluster(point_idx, neighbours, cluster_idx)`: Έτοιμη συνάρτηση, υπεύθυνη για την ανάθεση κάθε σημείου-κεραυνού στη κατάλληλη ομάδα. Δέχεται ως ορίσματα τρεις μεταβλητές. **`point_idx`**: το id του συγκεκριμένου σημείου, `neighbours`: ένα πίνακα με τα σημεία που ορίζονται ως γείτονικά σε ακτίνα μικρότερη από Eps και `cluster_idx`: το id της ομάδας στην οποία τοποθετείται το σημείο-κεραυνός.

`get_region_neighbours(point_idx)`: Έτοιμη συνάρτηση, υπολογίζει ποια σημεία-κεραυνοί βρίσκονται σε ακτίνα μικρότερη ή ίση της παραμέτρου Eps, για κάθε ένα σημείο-κεραυνό. Δέχεται ως όρισμα το id του σημείου-κεραυνού που εξετάζουμε και επιστρέφει μία μεταβλητή-πίνακα με τα γειτονικά σημεία.

`count_nearest_neighbour(point_idx)`: Συνάρτηση που κατασκευάστηκε για την εύρεση των τριών κοντινότερων γειτόνων κάθε σημείου-κεραυνού και την αποθήκευση των μεταξύ τους αποστάσεων. Δέχεται ως όρισμα το id του σημείου-κεραυνού που εξετάζουμε και οι αποστάσεις αποθηκεύονται αντίστοιχα στις μεταβλητές:

---

[8] Ester, Martin; Kriegel, Hans-Pter; Sander, Jörg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M., eds. *A density-based algorithm for discovering clusters in large spatial databases with noise*. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231. ISBN 1-57735-004-9.

- `nearest[]`
- `second_nearest[]`
- `third_nearest[]`

`set_eps(neighbour)`: Συνάρτηση που υλοποιήθηκε με στόχο τον αυτόματο υπολογισμό της παραμέτρου Eps. Η παράμετρος Eps υπολογίζεται ως ο μέσος όρος απόστασης κάθε σημείου-κεραυνού από τον k κοντινότερό του ανάλογα με το σύνολο των κεραυνών. Παράλληλα θέτουμε και ένα μέγιστο όριο 200 χιλιομέτρων για απόσταση του k κοντινότερου γείτονα, κυρίως για όταν υπάρχουν ελάχιστα δεδομένα ώστε η τιμή του Eps να μην επηρεάζεται αρνητικά. Επιστρέφει την τιμή του Eps.

`set_neighbour(neighbour)`: Συνάρτηση που υλοποιήθηκε για τον υπολογισμό του κατάλληλου k κοντινότερου γείτονα, όπου ισχύει 1<=k<=3, του οποίου η απόσταση θα χρησιμοποιηθεί για την εύρεση της παραμέτρου Eps. Δέχεται ως όρισμα μία κενή μεταβλητή-πίνακα και την επιστρέφει τα κατάλληλα δεδομένα σύμφωνα με τα παρακάτω.

- Για λιγότερους από 30 κεραυνούς χρησιμοποιούμε τον κοντινότερο γείτονα.
- Για περισσότερους από 30 και λιγότερους από 400 κεραυνούς χρησιμοποιούμε τον δεύτερο κοντινότερο γείτονα.
- Για περισσότερους από 400 κεραυνούς χρησιμοποιούμε τον τρίτο κοντινότερο γείτονα.

`set_minpts()`: Συνάρτηση που δημιουργήθηκε για τον αυτόματο υπολογισμό της παραμέτρου minPts. Επιστρέφει την τιμή του minPts.

- Για λιγότερους από 100 κεραυνούς η παράμετρος ισούται με 1.
- Για περισσότερους από 100 και λιγότερους από 1000 κεραυνούς, ισούται με το ακέραιο αποτέλεσμα της πράξης Κεραυνοί/100+1.
- Για περισσότερους από 1000 κεραυνούς, ισούται με το ακέραιο αποτέλεσμα της πράξης Κεραυνοί/100-2.

`var dbscan = function ()`: Πρόκειται για την κεντρική συνάρτηση, η οποία καλεί τις παραπάνω συναρτήσεις, υλοποιεί τον αλγόριθμο DBScan.

`dbscan.getClusters = function ()`: Καλώντας τη συνάρτηση αυτή, αφού εκτελεστεί ο αλγόριθμος, επιστρέφει τη μεταβλητή var clusters_centers η οποία περιλαμβάνει όλες τις πληροφορίες σχετικά με τις ομάδες που δημιουργήθηκαν μεταξύ των οποίων και τις συντεταγμένες των κέντρων τους και πόσους κεραυνούς περιλαμβάνουν, τα οποία αποθηκεύονται στη βάση δεδομένων.

`dbscan.data = function (d):` Πρόκειται για τη συνάρτηση που δέχεται ως όρισμα τα δεδομένα στα οποία θέλουμε να εκτελέσουμε τον αλγόριθμο DBScan.

Παράδειγμα εκτέλεσης του αλγορίθμου καθώς και ανάκτησης των αποτελεσμάτων:

```
var dbscanner = jDBSCAN().data(points);
var point_assignment_result = dbscanner();
var cluster_centers = dbscanner.getClusters();
```

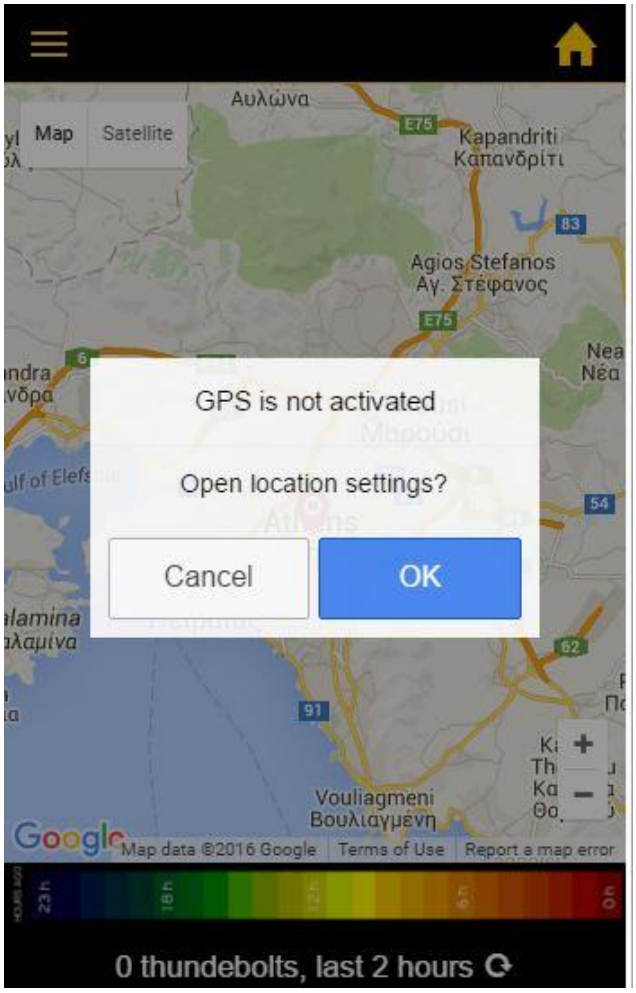
5.4 Hybrid mobile app

Με τη βοήθεια των Apache Cordova και Ionic frameworks δημιουργήσαμε μία υβριδική εφαρμογή κατάλληλη για τα σύγχρονα λειτουργικά συστήματα Android και iOS που χρησιμοποιούνται από έξυπνες συσκευές. Στην παρούσα εργασία λαμβάνοντας υπόψη τους δύο τρόπους ανάπτυξης εφαρμογής με τη χρήση της Apache Cordova, όπως αναλύθηκαν στην ενότητα Τρόποι ανάπτυξης Apache Cordova εφαρμογής, επιλέξαμε να υλοποιήσουμε την εφαρμογή πελάτη με τον Cross-platform workflow τρόπο ανάπτυξης.

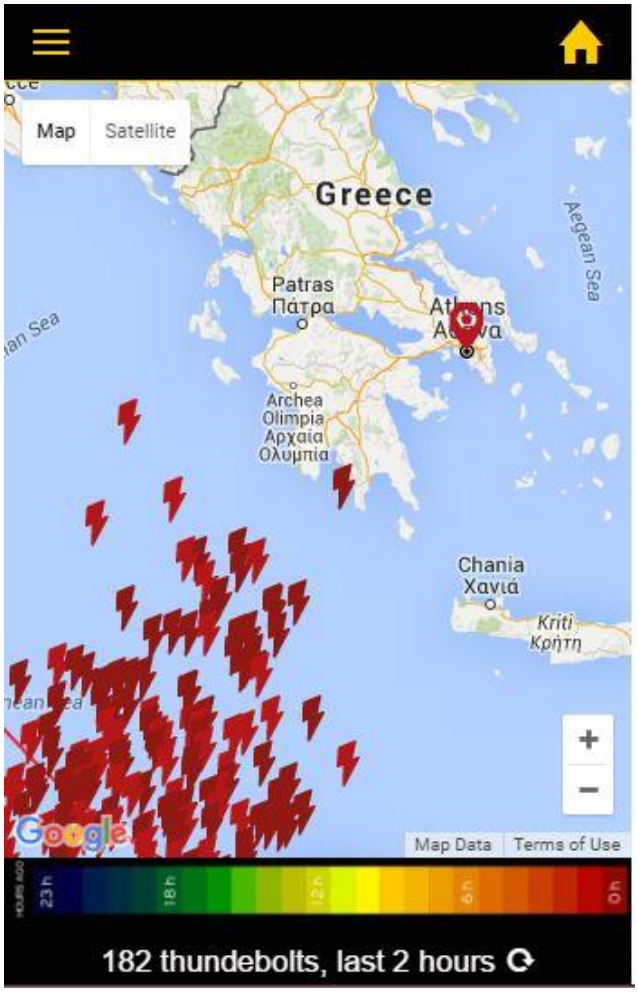
Η πρώτη σελίδα που βλέπει ο χρήστης κατά την έναρξη της εφαρμογής είναι η αρχική της εφαρμογής που φαίνεται στην Εικόνα 19. Ο χρήστης έχει τρεις επιλογές: Settings, My location, Top 3 places.

Εικόνα 19: Αρχική οθόνη

Επιλέγοντας My location, ο χρήστης μεταφέρεται στην σελίδα του χάρτη όπου με τη βοήθεια του GPS κεντράρεται στην γεωγραφική θέση του χρήστη (Εικόνα 21). Σε περίπτωση που η λειτουργία του GPS είναι απενεργοποιημένη, ο χρήστης ενημερώνεται και αν το επιλέξει μεταφέρεται στις ρυθμίσεις για να το ενεργοποιήσει (Εικόνα 20). Οι κεραυνοί που εμφανίζονται στην συγκεκριμένη λειτουργία είναι για τις τελευταίες δύο ώρες. Η εφαρμογή φορτώνει δεδομένα αποκλειστικά όσον αφορά την εμφανή περιοχή του χάρτη. Ωστόσο ο χρήστης έχει τη δυνατότητα ανανέωσης των δεδομένων πατώντας το κουμπί στο κάτω μέρος της οθόνης για όποιο σημείο του χάρτη τον ενδιαφέρει.

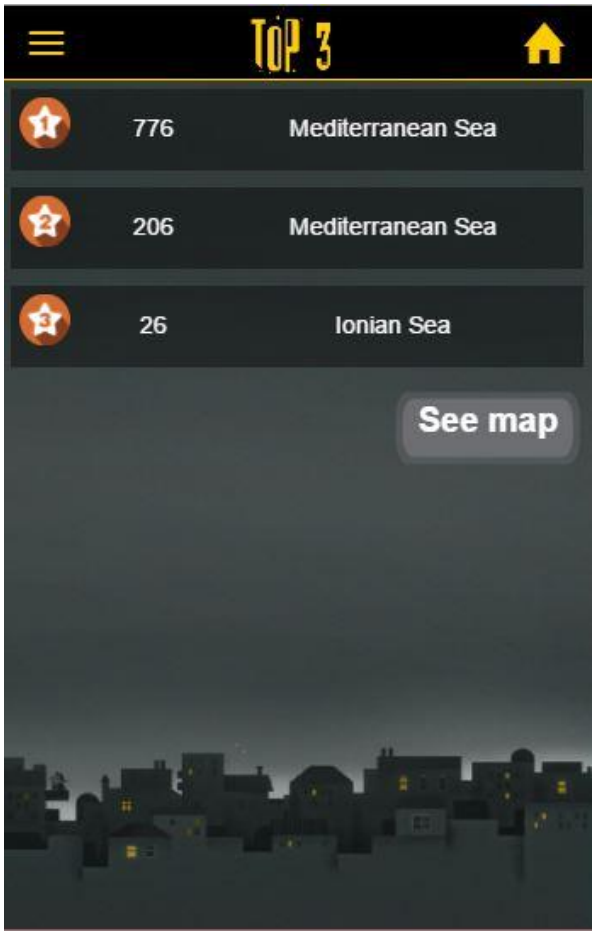


Εικόνα 20: Ενημερωτικό μήνυμα, όταν η λειτουργία GPS είναι απενεργοποιημένη



Εικόνα 21: Εμφάνιση χάρτη στη λειτουργία My location

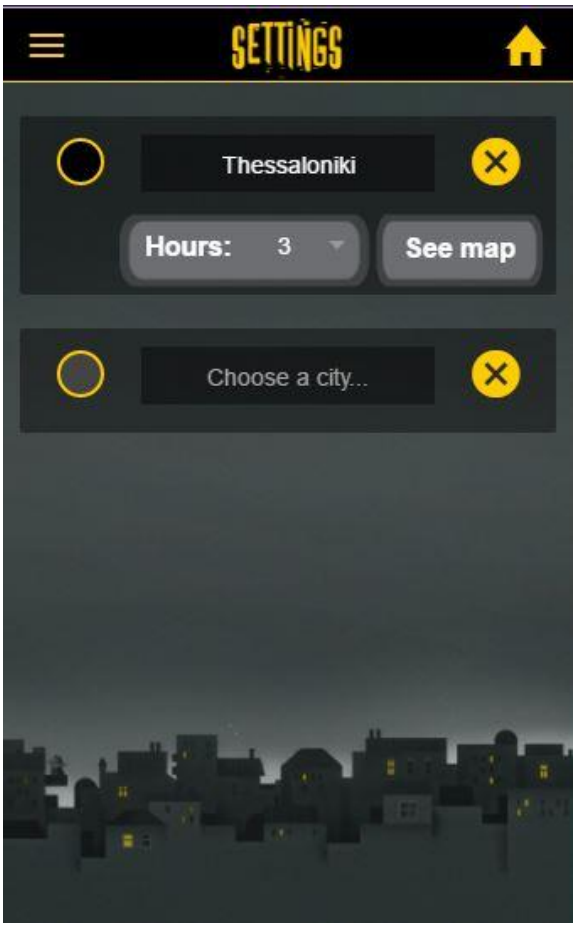
Επιλέγοντας Top 3 places, ο χρήστης μεταφέρεται στην σελίδα Top 3 όπου εμφανίζονται έως τρεις περιοχές με τους περισσότερους κεραυνούς τις τελευταίες 24 ώρες (Εικόνα 22). Επίσης με τη χρήση του 'See map' μπορεί να μεταφερθεί στον χάρτη όπου οι περιοχές θα είναι σημειωμένες με markers. Πατώντας πάνω στον marker κάθε περιοχής, μπορεί να ενημερωθεί για τον αριθμό των κεραυνών που ανήκουν στην ομάδα και το όνομα της περιοχής (Εικόνα 23).



Εικόνα 22: Σελίδα Top 3 με πληροφορίες για τις περιοχές με τους περισσότερους κεραυνούς τις τελευταίες 24 ώρες

Εικόνα 23: Χάρτης με τις περιοχές με τους περισσότερους κεραυνούς τις τελευταίες 24 ώρες

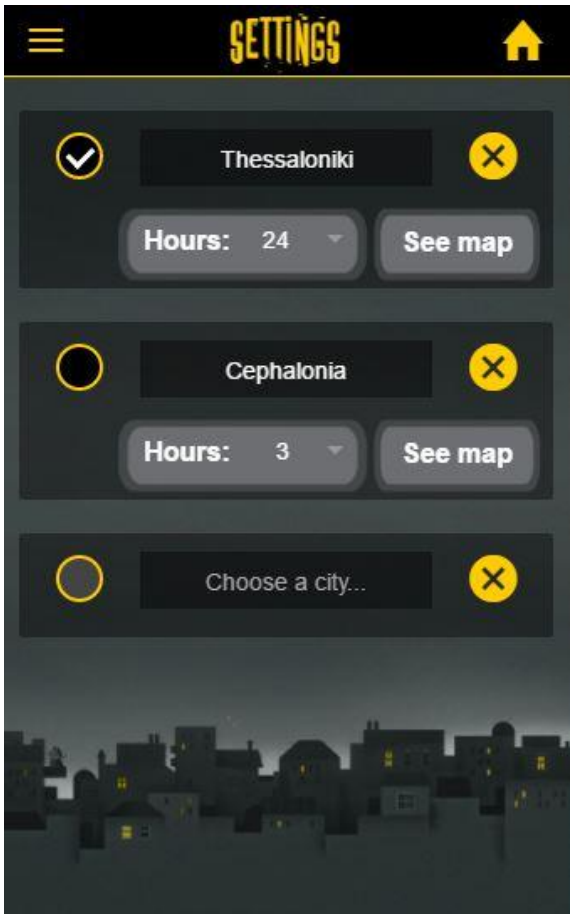
Επιλέγοντας Settings, ο χρήστης μεταφέρεται στην σελίδα των ρυθμίσεων (Εικόνα 24). Εκεί έχει τη δυνατότητα εισαγωγής έως τριών περιοχών της επιλογής του. Όταν επιλέξει να εισάγει μία περιοχή, μεταφέρεται στην σελίδα Search όπου καθώς πληκτρολογεί στο διαθέσιμο πεδίο, από κάτω εμφανίζονται οι διαθέσιμες επιλογές(Εικόνα 25).



Εικόνα 24: Σελίδα settings, όπου ο χρήστης μπορεί να προσθέσει περιοχές της επιλογής του

Εικόνα 25: Σελίδα search, όπου ο χρήστης με τη βοήθεια του Autocomplete GoogleAPI επιλέγει την περιοχή που επιθυμεί

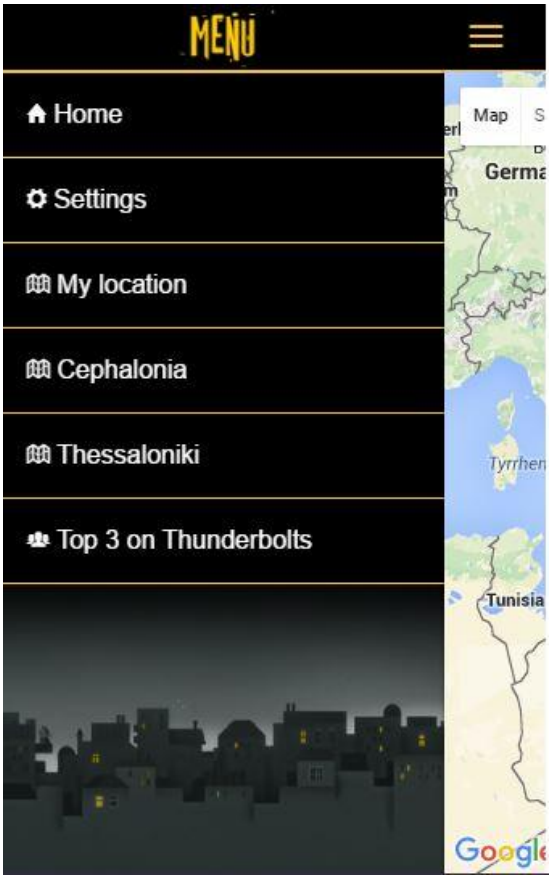
Επιλέγοντας την περιοχή που επιθυμεί, επιστρέφει στην σελίδα των ρυθμίσεων, η οποία διαμορφώνεται όπως φαίνεται στην εικόνα Εικόνα 26. Πλέον η επιλογή του είναι αποθηκευμένη. Παράλληλα ο χρήστης έχει τη δυνατότητα να επιλέξει το διάστημα ωρών για το οποίο θα εμφανίζονται κεραυνοί στο χάρτη της συγκεκριμένης περιοχής, αν επιθυμεί η συγκεκριμένη περιοχή να εμφανίζεται στην κεντρική οθόνη, κουμπί για την διαγραφή της περιοχής καθώς και ένα κουμπί 'See map' που οδηγεί απευθείας στον χάρτη με τις ιδιότητες που έχουν οριστεί. Επιστρέφοντας στην αρχική οθόνη και αν ο χρήστης έχει επιλέξει η περιοχή να εμφανίζεται στην αρχική, τότε διαμορφώνεται όπως φαίνεται στην Εικόνα 27, με τον χρήστη να έχει απευθείας πρόσβαση στις περιοχές που έχει επιλέξει.



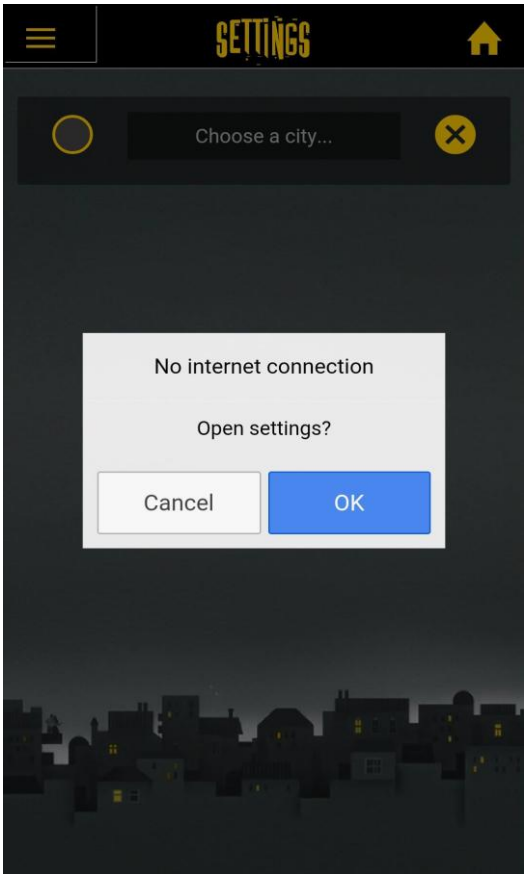
Εικόνα 26: Σελίδα settings, όπου πλέον έχει προστεθεί η νέα επιλογή του χρήστη

Εικόνα 27: Αρχική σελίδα, όπου πλέον εμφανίζονται οι νέες επιλογές του χρήστη

Παράλληλα στις σελίδες Settings, Top 3 και οποιουδήποτε διαθέσιμου χάρτη στην πάνω δεξιά γωνία εμφανίζεται κουμπί για απευθείας μεταφορά του χρήστη στην αρχική ενώ στην πάνω αριστερή γωνία εμφανίζεται κουμπί για την εμφάνιση του μενού της εφαρμογής. Από το μενού ο χρήστης μπορεί να μεταφερθεί σε οποιαδήποτε σελίδα της εφαρμογής, όπως και στις επιλεγμένες περιοχές του χρήστη, εύκολα και γρήγορα χωρίς να είναι υποχρεωμένος να επιστρέψει πρώτα στην αρχική (Εικόνα 28). Τέλος αν η σύνδεση στο διαδίκτυο διακοπεί ή είναι αδύνατη, εμφανίζεται μήνυμα που ενημερώνει τον χρήστη, προτρέποντάς τον να ελέγξει αν είναι ενεργοποιημένο (Εικόνα 29).



Εικόνα 28: Το μενού, όπου εμφανίζονται και οι προσωπικές επιλογές περιοχών του χρήστη



Εικόνα 29: Ενημερωτικό μήνυμα που εμφανίζεται αν δεν υπάρχει σύνδεση στο διαδίκτυο

5.4.1 Html5 pages

Πρόκειται για σελίδες υλοποιημένες σε γλώσσα HTML, CSS και AngularJS με τη βοήθεια του Ionic framework που αποτελούν την επιφάνεια διεπαφής της εφαρμογής με το χρήστη όπως παρουσιάστηκε στην προηγούμενη ενότητα (5.4 Hybrid mobile app). Δημιουργήθηκαν συνολικά έξι σελίδες HTML.

Home.html

Πρόκειται για την αρχική σελίδα της εφαρμογής που περιλαμβάνει με τη μορφή κουμπιών τις βασικές επιλογές της εφαρμογής. Σταθερά εμφανίζονται τρία κουμπιά που οδηγούν αντίστοιχα στη σελίδα των ρυθμίσεων, του χάρτη με τη χρήση GPS για τον ορισμό της τοποθεσίας και τέλος της ενημέρωσης σχετικά με τις περιοχές με τους περισσότερους κεραυνούς τις τελευταίες 24 ώρες. Ανάλογα με τις ρυθμίσεις του χρήστη εμφανίζονται μέχρι τρία ακόμα κουμπιά που οδηγούν στους αντίστοιχους χάρτες με τις περιοχές που έχει ορίσει ο χρήστης στις ρυθμίσεις.

Παράδειγμα κώδικα από τη σελίδα Home.html:

```
<div ng-repeat="i in [0,1,2]" class = "col" style="text-align: center" ng-
hide="!updatedCheckbox(i) || updatedCityName(i)==null">
      <img       id="map{{i}}Btn"       ng-src="img/map{{i}}Btn.png"       on-
      touch="onTouch('map',{{i}})" height='70px'>
      <div>
            <b>{{updatedCityName(i) | limitTo:15}}</b>
      </div>
 </div>
```

Στο παραπάνω κομμάτι κώδικα χρησιμοποιούμε τα εξής στοιχεία της AngularJS:
ng-repeat: για την δυναμική δημιουργία τριών κουμπιών, ένα για κάθε επιλογή του χρήστη.
ng-hide: για την απόκρυψη του εκάστοτε κουμπιού αν η επιλογή του χρήστη από τη σελίδα settings δεν είναι να εμφανίζεται η συγκεκριμένη πόλη στην αρχική οθόνη.
on-touch: για την εκτέλεση του κώδικα της συνάρτησης `onTouch ('map',{{i}})` του αρχείου controllers.js όταν ο χρήστης αγγίξει το κουμπί-εικόνα που έχει στόχο τη δημιουργία εφέ hover στην εικόνα και τη μεταφορά του χρήστη στον χάρτη που αιτήθηκε.

Map.html

Πρόκειται για την σελίδα που παρουσιάζει σε μορφή χάρτη τα δεδομένα των κεραυνών ή τις περιοχές με τους περισσότερους κεραυνούς τις τελευταίες ώρες. Στην πρώτη περίπτωση στο

κάτω μέρος της οθόνης εμφανίζεται η επιλογή ανανέωσης του χάρτη με τα αντίστοιχα δεδομένα για την περιοχή που είναι εμφανής τη κάθε στιγμή στον χρήστη.

Settings.html

Σε αυτή τη σελίδα ο χρήστης μπορεί να επιλέξει μέχρι τρεις περιοχές της αρεσκείας του προς αποθήκευση. Επιλέγει για πόσες τελευταίες ώρες θέλει να βλέπει δεδομένα στο χάρτη καθώς και αν επιθυμεί να εμφανίζεται το αντίστοιχο κουμπί για γρήγορη πρόσβαση στην αρχική σελίδα και στο μενού. Ανά πάσα στιγμή μπορεί να τροποποιήσει οποιαδήποτε επιλογή του ή και να διαγράψει πλήρως μία περιοχή. Τέλος έχει την επιλογή μεταφοράς του στη σελίδα του χάρτη.

Search.html

Πρόκειται για τη σελίδα στην οποία μεταφέρεται ο χρήστης από την Settings.html για τον ορισμό της τοποθεσίας που επιθυμεί να αποθηκεύσει. Ο χρήστης πληκτρολογεί την περιοχή που θέλει στο αντίστοιχο πεδίο, από κάτω εμφανίζονται οι διαθέσιμες επιλογές, ο χρήστης επιλέγει και μεταφέρεται πάλι πίσω στην σελίδα Settings.html για να συνεχίσει τις ρυθμίσεις του. Σε περίπτωση τροποποίησης ήδη υπάρχουσας περιοχής υπάρχει επιλογή επιστροφής χωρίς τη μεταβολή κάποιου στοιχείου.

Top3.html

Η συγκεκριμένη σελίδα αποσκοπεί στην πληροφόρηση του χρήστη για τις περιοχές με τους περισσότερους κεραυνούς τις τελευταίες 24 ώρες. Εμφανίζει μέχρι τρεις περιοχές σε φθίνουσα σειρά παρουσιάζοντας το όνομά της και πόσοι κεραυνοί έπεσαν στην γύρω περιοχή. Ακόμη υπάρχει επιλογή μεταφοράς στη σελίδα του χάρτη για την οπτική απεικόνιση των συγκεκριμένων περιοχών.

Menu.html

Πρόκειται για την υλοποίηση του μενού που εμφανίζεται στις σελίδες map.html, settings.html και top3.html. Περιλαμβάνει συνδέσμους προς όλες τις σελίδες της εφαρμογής ενώ αν το επιλέξει ο χρήστης, εμφανίζονται και οι περιοχές της αρεσκείας του που έχει δηλώσει στη σελίδα Settings.html.

5.4.2 Cordova plugins

Όπως αναλύσαμε στην ενότητα Cordova Plugins η Apache Cordova διαθέτει έτοιμα plugins για χρήση. Στη συγκεκριμένη εργασία χρησιμοποιήθηκαν τέσσερα από αυτά, τα οποία αναλύονται παρακάτω:

5.4.2.1 Whitelist plugin

Μέσω του Whitelist plugin [44] έχουμε τη δυνατότητα να ορίσουμε για λόγους ασφαλείας σε ποιες σελίδες επιτρέπουμε στο WebView να πλοηγηθεί, ποιες σελίδες επιτρέπουμε στην εφαρμογή να ζητήσει να ανοίξει και προς ποιες σελίδες επιτρέπεται να στείλει αιτήματα. Χωρίς το συγκεκριμένο plugin η εφαρμογή μας δεν είναι σε θέση να επικοινωνήσει με το υπόλοιπο διαδίκτυο.

Εγκατάσταση:

Εκτελούμε την παρακάτω εντολή στη βάση του project μας και το plugin είναι έτοιμο για χρήση

```
cordova plugin add cordova-plugin-whitelist
```

Η πλοήγηση στις σελίδες της εφαρμογής είναι επιτρεπτή χωρίς περαιτέρω ενέργειες μετά την εγκατάσταση του plugin αλλά για άλλα URLs πρέπει να προσθέσουμε στο αρχείο config.xml της εφαρμογής μας τα κατάλληλα tags.

Πιο συγκεκριμένα στην υλοποίηση της εφαρμογής μας:

Για να επιτρέψουμε αιτήματα προς την RESTful υπηρεσία που έχουμε δημιουργήσει στον http://195.251.31.119/ προσθέτουμε:

```
<access origin="http://*195.251.31.119*" />
```

Για να επιτρέψουμε αιτήματα προς τα GoogleAPIs προσθέτουμε:

```
<access origin="https://*.googleapi*" />
```

Και τέλος για να επιτρέψουμε την προβολή του χάρτη προσθέτουμε:

```
<allow-navigation href="*"/>
<allow-intent href="http://maps.google.com/maps/api*"/>
```

Παράλληλα στη σελίδα Map.html χρησιμοποιούμε content-policy [44] που επιτρέπει CSS, AJAX, object, frame, media κα. με την εξαίρεση ότι το CSS και τα scripts πρέπει να είναι αρχείο της εφαρμογής ή στην ίδια σελίδα:

```
<meta http-equiv="Content-Security-Policy" content="default-src ; style-src 'self' 'unsafe-inline'; script-src 'self' 'unsafe-inline' 'unsafe-eval'">
```

5.4.2.2 Network Information Plugin

Το Network information plugin [45] παρέχει πληροφορίες σχετικά με την κατάσταση σύνδεσης του κινητού τηλεφώνου σε wifi ή δίκτυο τηλεφωνίας καθώς και αν βρίσκεται συνδεδεμένο στο διαδίκτυο.

Εγκατάσταση:

Εκτελούμε την παρακάτω εντολή στη βάση του project μας και το plugin είναι έτοιμο για χρήση

```
cordova plugin add cordova-plugin-network-information
```

Το plugin μας παρέχει το αντικείμενο navigator.connection με την ιδιότητα type το οποίο μας παρέχει πληροφορίες για το δίκτυο κινητής τηλεφωνίας και τη wifi σύνδεση της συσκευής.

Οι πιθανές απαντήσεις είναι:

- `Connection.UNKNOWN`
- `Connection.ETHERNET`
- `Connection.WIFI`
- `Connection.CELL_2G`
- `Connection.CELL_3G`
- `Connection.CELL_4G`
- `Connection.CELL`
- `Connection.NONE`

Ακόμη παρέχονται event listeners που εκτελούνται όταν η συσκευή συνδέεται ή αποσυνδέεται από το διαδίκτυο αντίστοιχα.

Παραδείγματα εκτέλεσης:

```
document.addEventListener("online", yourCallbackFunction, false);
document.addEventListener("offline", yourCallbackFunction, false);
```

Συγκεκριμένα στην υλοποίηση της εφαρμογής μας χρησιμοποιήθηκαν για την ενημέρωση του χρήστη και αποκοπή πρόσβασης στις σελίδες της εφαρμογής που απαιτούν σύνδεση στο διαδίκτυο όταν αυτή δεν υπάρχει καθώς και για την επαναφορά και ανανέωση των σελίδων όταν η σύνδεση γίνεται ξανά εφικτή.

Παράδειγμα κώδικα από την εφαρμογή:

```
$rootScope.$on('$cordovaNetwork:online', function (event, networkState) {
      PopUp.hidePopUp();
      if ($location.path()==="/app/map") {
            checkLoaded(false);
      }else{
            $location.search('internet', 'true');
      }
});
$rootScope.$on('$cordovaNetwork:offline', function (event, networkState) {
```

```
	if(alertPopup===null){
		PopUp.createPopUp();
	}
});
```

5.4.2.3 Geolocation Plugin

Το Geolocation plugin [46] παρέχει πληροφορίες σχετικά με την τοποθεσία της συσκευής, δηλαδή το γεωγραφικό μήκος και πλάτος της. Τις συνήθεις πηγές για την εύρεση της τοποθεσίας της συσκευής αποτελούν το GPS σύστημα του κινητού και δεδομένα που προκύπτουν είτε από το δίκτυο κινητής τηλεφωνίας είτε από το διαδίκτυο όπως IP διεύθυνση, RFID και διευθύνσεις MAC του WiFi ή Bluetooth. Παρόλα αυτά η διεύθυνση που θα επιστραφεί δεν εγγυάται ότι θα είναι ακριβής και αληθής.

Εγκατάσταση:

Εκτελούμε την παρακάτω εντολή στη βάση του project μας και το plugin είναι έτοιμο για χρήση

```
cordova plugin add cordova-plugin-geolocation
```

Το plugin δημιουργεί, αν δεν υπάρχει ήδη από το λειτουργικό σύστημα, το αντικείμενο navigator.geolocation το οποίο υποστηρίζει τις μεθόδους:

- `navigator.geolocation.getCurrentPosition`
- `navigator.geolocation.watchPosition`
- `navigator.geolocation.clearWatch`

Παράδειγμα εκτέλεσης:

```
navigator.geolocation.getCurrentPosition(geolocationSuccess,
[geolocationError], [geolocationOptions]);
```

Συγκεκριμένα στην υλοποίηση της εφαρμογής χρησιμοποιήθηκε μόνο η πρώτη μέθοδος για την εύρεση της τοποθεσίας του χρήστη και την γραφική της απεικόνιση πάνω στο χάρτη στη σελίδα map.html.

Αρχικά ορίζουμε τις ρυθμίσεις με τις οποίες θέλουμε να χρησιμοποιήσουμε το plugin. Συγκεκριμένα του δίνουμε 4 δευτερόλεπτα να εντοπίσει την τοποθεσία. Ακόμη θέτοντας true την δεύτερη μεταβλητή δηλώνουμε ότι ζητάμε τη τοποθεσία με μέγιστη ακρίβεια, δηλαδή με τη χρήση GPS και όχι μέσω δικτύων όπως αναλύσαμε παραπάνω. Στη συνέχεια αποθηκεύουμε την τοποθεσία που επιστράφηκε στη μεταβλητή latlng και καλούμε τη συνάρτηση `createMap(refresh)` που θα αναλυθεί παρακάτω στο 5.4.4.2 App.js.

Παράδειγμα κώδικα από την εφαρμογή:

```
var gpsOptions = {timeout: 4000, enableHighAccuracy: true};
$cordovaGeolocation.getCurrentPosition(gpsOptions).then(function (position) {
      latLng      =      new      google.maps.LatLng(position.coords.latitude,
position.coords.longitude);
      createMap(refresh,function() {
            $ionicLoading.hide();
      });
 }
```

5.4.2.4 Native settings plugin

Το plugin native settings [47] υποστηρίζει το λειτουργικό σύστημα android και iOS8. Λόγω του ότι αυτή τη στιγμή όπως φαίνεται στην Εικόνα 17 το iOS 9 κυριαρχεί με 84% έναντι 11% του Ios 8, οι λειτουργίες του συγκεκριμένου plugin χρησιμοποιήθηκαν μόνο για την έκδοση Android. Επιτρέπει το άνοιγμα των ρυθμίσεων του κινητού μέσα από την εφαρμογή, παρέχοντας μάλιστα πληθώρα επιλογών για την συγκεκριμένη θέση που θέλουμε να ανοίξουμε. Μερικές από αυτές είναι οι: “settings”, “wifi”, ”location_source”, “airplane_mode” και δεκάδες άλλες.

Εγκατάσταση:

Εκτελούμε την παρακάτω εντολή στη βάση του project μας και το plugin είναι έτοιμο για χρήση

```
cordova plugin add https://github.com/selahssea/Cordova-open-native-settings
```

Παράδειγμα εκτέλεσης:

```
cordova.plugins.settings.openSetting(settingName,           success_callback,
failure_callback);
```

Συγκεκριμένα όταν δεν υπάρχει σύνδεση στο διαδίκτυο εμφανίζεται στον χρήστη ένα παράθυρο ενημέρωσης που του δίνει την επιλογή να ανοίξει τις ρυθμίσεις για ενεργοποίηση της σύνδεσης.

Παράδειγμα κώδικα από την εφαρμογή:

```
cordova.plugins.settings.openSetting("settings", function () {
      $timeout(function () { //reload page
            $location.search('internet', 'true');
      }, 1000);
}, function () {
      alert("Error opening settings");
      });
```

Επίσης όταν ο χρήστης επιθυμεί να χρησιμοποιήσει την επιλογή του χάρτη με τη χρήση της τοποθεσίας του αλλά το GPS είναι απενεργοποιημένο εμφανίζεται το αντίστοιχο παράθυρο ενημέρωσης με επιλογή μεταφοράς στις ρυθμίσεις GPS του κινητού.

Παράδειγμα κώδικα από την εφαρμογή:

```
cordova.plugins.settings.openSetting("location_source", function () {
      document.addEventListener("resume",        LocationSwap.go('map','gps'),
      false);
}
```

5.4.2.5 Splashscreen plugin

Το plugin Splashscreen [48] χρησιμοποιείται για την εμφάνιση και απόκρυψη μίας εικόνας ως splashscreen όταν ο χρήστης ενεργοποιεί την εφαρμογή.

Εγκατάσταση:

Εκτελούμε την παρακάτω εντολή στη βάση του project μας και το plugin είναι έτοιμο για χρήση

```
cordova plugin add cordova-plugin-splashscreen
```

Για την χρησιμοποίηση του plugin είναι απαραίτητες κάποιες προσθήκες στο αρχείο config.xml του project μας.

Αρχικά πρέπει να δηλώσουμε την εικόνα που θα χρησιμοποιηθεί ως splashscreen. Αυτό μπορεί να γίνει με δύο τρόπους:

1. Προσθέτουμε το κατάλληλο tag στο αρχείο config.xml

   Για παράδειγμα:

   ```
   <platform name="android">
   <splash     src="resources\android\splash\drawable-port-ldpi-screen.png"
   density="port-ldpi"/>
   </platform>
   ```

2. Αποθηκεύουμε στο φάκελο resources την εικόνα που θέλουμε να χρησιμοποιήσουμε ως splashscreen με όνομα splash και τύπο .png, .psd ή .ai.

   Στην βάση του project μας εκτελούμε την εντολή: `ionic resources –splash` και αυτόματα προστίθενται τα κατάλληλα tags στο αρχείο config.xml.

   Για παράδειγμα:

   ```
   <platform name="android">
       <splash            src="resources\android\splash\drawable-port-ldpi-
   screen.png" density="port-ldpi"/>
       <splash            src="resources\android\splash\drawable-port-mdpi-
   screen.png" density="port-mdpi"/>
       <splash            src="resources\android\splash\drawable-port-hdpi-
   screen.png" density="port-hdpi"/>
     </platform>
   ```

3. Στη συνέχεια προσθέσαμε τα εξής δύο tags:

```
<preference name="SplashScreen" value="screen"/>
<preference name="SplashScreenDelay" value="3000"/>
```

Με τις παραπάνω εντολές δηλώνουμε την χρήση του plugin καθώς και τον χρόνο που μεσολαβεί μέχρι την αυτόματη απόκρυψη του splashscreen.

5.4.3 Google APIs

Τα Google APIs που χρησιμοποιήθηκαν στην παρούσα εργασία είναι τρία. Το Google Maps Javascript API, το Google Maps Geocoding API και το Google Places API Web Service.

5.4.3.1 Google Maps Javascript API

Το Google Maps Javascript API [49] αποτελεί υπηρεσία της Google η οποία έχει σχεδιαστεί για την εισαγωγή χαρτών σε εφαρμογές. Αποσκοπεί στη γρήγορη φόρτωση και την αποδοτική λειτουργία του ειδικά για smartphones. Για την χρήση του απαιτείται η δήλωσή του μέσα σε script tag.

Παράδειγμα δήλωσης στο αρχείο index.html της εφρμογής:

```
<script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyC1RlpsPiVY1X4AR5l2xPKD
3A9bRkf3oq4" async defer></script>
```

Παράδειγμα δημιουργίας χάρτη από το αρχείο 5.4.4.2 App.js της εφαρμογής-πελάτη:

```
      mapOptions = {
            center: latLng,
            zoom: 10,
            mapTypeId: google.maps.MapTypeId.ROADMAP,
            streetViewControl: false
      };
map = new google.maps.Map(document.getElementById("map"), mapOptions);
```

Το Google Maps Javascript API προσφέρει διάφορες λειτουργίες μεταξύ των οποίων βρίσκονται οι markers, events και data layers που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής μας.

Marker: Ένας marker προσδιορίζει με μία εικόνα πάνω στο χάρτη μια τοποθεσία. Μεταξύ των ιδιοτήτων ενός marker μπορούμε να δηλώσουμε position, map, title, icon και άλλα.

Παράδειγμα εισαγωγής marker από το αρχείο 5.4.4.2 App.js της εφαρμογής-πελάτη:

```
marker = new google.maps.Marker({
      map: map,
      animation: google.maps.Animation.DROP,
      position: latLng,
      icon: './img/marker.png'
});
```

Events: Η Javascript εντός του browser οδηγείται από events, δηλαδή για την αλληλεπίδραση με το χρήστη και την εφαρμογή αντιδρά στα events που δημιουργούνται. Τα events χωρίζονται σε User events τα οποία σχετίζονται με τον χρήστη (για παράδειγμα 'click') και MVC state change notifications τα οποία αντικατοπτρίζουν αλλαγές στο Javascript API (για παράδειγμα 'zoom_changed'). Για να παρέμβουμε και να εκτελέσουμε κώδικα όταν συμβαίνει κάποιο event μπορεί να χρησιμοποιηθεί η συνάρτηση addListener().

Παράδειγμα:

```
//User event
marker.addListener('click', function() {
      map.setZoom(8);
      map.setCenter(marker.getPosition());
});


//MVC state change notifications
map.addListener('center_changed', function() {
      // 3 seconds after the center of the map has changed, pan back to the
      // marker.
      window.setTimeout(function() {
            map.panTo(marker.getPosition());
      }, 3000);
});
```

Data layers: Προσφέρει τη δυνατότητα αποτύπωσης γεωμετρικών ή στη συγκεκριμένη περίπτωση γεωγραφικών δεδομένων πάνω στον χάρτη. Διαθέτει μάλιστα τη συνάρτηση loadGeoJson() που επιτρέπει την εισαγωγή των δεδομένων στον χάρτη με μόλις μία γραμμή κώδικα.

Παράδειγμα από το αρχείο 5.4.4.2 App.js της εφαρμογής-πελάτη:

```
map.data.loadGeoJson('http://195.251.31.119/~strikes/api/index.php/ThunderboltsController');
```

5.4.3.2 Google Maps Geocoding API

Το Google Maps Geocoding API [50] αποτελεί υπηρεσία της Google που προσφέρει τόσο Geocoding όσο και Reverse Geocoding.

- Geocoding, είναι η διαδικασία μετατροπής μίας διεύθυνσης σε συντεταγμένες, δηλαδή η εύρεση γεωγραφικού μήκους και πλάτους.
- Reverse Geocoding, είναι η αντίστροφη διαδικασία, δηλαδή η μετατροπή γεωγραφικού μήκους και πλάτους σε ονομασία περιοχής.

Παράδειγμα HTTP αιτήματος προς την υπηρεσία Geocoding του Google Maps Geocoding API :

```
https://maps.googleapis.com/maps/api/geocode/json?address=athens
```

Πιο συγκεκριμένα στην εφαρμογή μας χρησιμοποιήσαμε τη λειτουργία Geocoding για την εύρεση της τοποθεσίας των περιοχών που επιλέγει ο χρήστης από τις σελίδες Settings.html και Search.html. Το Google Maps Geocoding API προσφέρει πολλές παραμέτρους που μπορούμε να προσθέσουμε με τη μέθοδο GET της HTTP στο αίτημά μας. Οι παράμετροι που χρησιμοποιήσαμε είναι οι εξής:

Key: Το API Key που έχουμε ενεργοποιήσει για το συγκεκριμένο API από το API Google Console. Υποχρεωτικό για τη χρησιμοποίηση των παραμέτρων region και bounds.
Region: Δηλώνουμε το κωδικό περιοχής που μας ενδιαφέρει, gr για την Ελλάδα.
Bounds: Δηλώνουμε τις συντεταγμένες της νοτιοδυτικής και βορειοανατολικής γωνίας του χάρτη στον οποίο επιθυμούμε να λάβουμε σχετικό αποτέλεσμα. Η παράμετρος αυτή επηρεάζει το αποτέλεσμα που θα λάβουμε αλλά αν υπάρχουν αποτελέσματα σχετικά εκτός των συντεταγμένων, μπορεί να συμπεριληφθούν στο αποτέλεσμα.
Address: Τη διεύθυνση για την οποία επιθυμούμε αποτέλεσμα

Παράδειγμα κώδικα από την εφαρμογή:

```
$http({
      url:"https://maps.googleapis.com/maps/api/geocode/json?key="+apiKey+"&region=gr&bounds=31.458014039186203,12.47032852382813|44.20686993524687,33.73985977382813&address=" + cityName,
      dataType: "json",
```

```
        method: "GET",
        headers: {
              "Content-Type": "application/json"
        }
})
```

5.4.3.3 Google Places API Web Service

Το Google Places API Web Service [51] αποτελεί υπηρεσία της Google η οποία ανάλογα με τα HTTP αιτήματα που λαμβάνει επιστρέφει πληροφορίες για μέρη όπως εγκαταστάσεις, τοποθεσίες και σημεία ενδιαφέροντος. Προσφέρει πληθώρα διαθέσιμων αιτημάτων περιοχών. Στην παρούσα εργασία χρησιμοποιήθηκε το Place Autocomplete [52], για την αυτόματη συμπλήρωση προτεινόμενων τοποθεσιών κατά την πληκτρολόγηση του χρήστη στη σελίδα Search.html.

Παράδειγμα HTTP αιτήματος προς την υπηρεσία Place Autocomplete του Google Places API Web Service:

```
https://maps.googleapis.com/maps/api/place/autocomplete/output?parameters
```

output: Δηλώνουμε το τύπο αρχείου που επιθυμούμε ως απάντηση. Μπορεί να είναι είτε JSON (προτεινόμενο) είτε XML

parameters: Οι παράμετροι key και input είναι υποχρεωτικές και δηλώνουν το APIKey της εφαρμογής και την είσοδο για την οποία επιθυμούμε να λάβουμε προτεινόμενα αποτελέσματα. Άλλες προαιρετικές παράμετροι που χρησιμοποιήθηκαν στην παρούσα εργασία αποτελούν οι region και bounds όπως αναλύθηκαν στην ενότητα marker = new google.maps.Marker({

```
map: map,

animation: google.maps.Animation.DROP,

position: latLng,

icon: './img/marker.png'
});
```

Events: Η Javascript εντός του browser οδηγείται από events, δηλαδή για την αλληλεπίδραση με το χρήστη και την εφαρμογή αντιδρά στα events που δημιουργούνται. Τα events χωρίζονται σε User events τα οποία σχετίζονται με τον χρήστη (για παράδειγμα ‘click’) και MVC state change notifications τα οποία αντικατοπτρίζουν αλλαγές στο Javascript API (για παράδειγμα

‘zoom_changed’). Για να παρέμβουμε και να εκτελέσουμε κώδικα όταν συμβαίνει κάποιο event μπορεί να χρησιμοποιηθεί η συνάρτηση addListener().

Παράδειγμα:

```
//User event
marker.addListener('click', function() {
	map.setZoom(8);
	map.setCenter(marker.getPosition());
});


//MVC state change notifications
map.addListener('center_changed', function() {
	// 3 seconds after the center of the map has changed, pan back to the
	// marker.
	window.setTimeout(function() {
		map.panTo(marker.getPosition());
	}, 3000);
});
```

.

Παράδειγμα κώδικα εφαρμογής:

```
$http({
	url:"https://maps.googleapis.com/maps/api/place/autocomplete/json?key="
+apiKey+"&region=gr&bounds=31.458014039186203,12.47032852382813|44.2068699352
4687,33.73985977382813&input=" + $scope.City[i].name,
	dataType: "json",
	method: "GET",
	headers: {
		"Content-Type": "application/json"
	}
})
```

5.4.4 JS αρχεία

Στην παρούσα εφαρμογή περιλαμβάνονται στον φάκελο js του project τρία αρχεία, αυτά είναι το App.js, Services.js και Controllers.js. Τα αρχεία αυτά είναι υπεύθυνα για οποιαδήποτε ενέργεια σχετίζεται με την αλληλεπίδραση του χρήστη και χρησιμοποιούν ως dependencies άλλες υπηρεσίες υλοποιημένες από εμάς ή έτοιμες υπηρεσίες που προσφέρει η Apache cordova.

Έτοιμες υπηρεσίες services που χρησιμοποιήθηκαν στην παρούσα εργασία είναι οι εξής:

$location: Μέσω της υπηρεσίας $location, ο προγραμματιστής έχει τη δυνατότητα να λάβει το URL της HTML σελίδας που βρίσκεται ο χρήστης αλλά και να επέμβει σε αυτό αλλάζοντάς το αφού οποιαδήποτε αλλαγή γίνει, είτε στο URL απευθείας είτε μέσω της $location, επιφέρει το ίδιο αποτέλεσμα στο άλλο [53].

$http: Αποσκοπεί στην επικοινωνία και μεταφορά δεδομένων μεταξύ ενός απομακρυσμένου διακομιστή και της εφαρμογής. Όταν καλείται, ανάλογα με την επιλογή του προγραμματιστή, δημιουργείται ένα αντικείμενο XMLHttpRequest ή η μεταφορά γίνεται μέσω της τεχνολογίας JSONP ***[53]***.

$ionicLoading: Πρόκειται για υπηρεσία του Ionic module που χρησιμοποιείται για να ενημερωθεί ο χρήστης ότι γίνεται κάποια διεργασία και να μπλοκαριστεί η διεπαφή της εφαρμογής μαζί του για σύντομο χρονικό διάστημα. Διαθέτει δύο μεθόδους, show για την χρησιμοποίησή του και hide για την απόκρυψή του όταν η διεργασία τερματιστεί [54].

$stateProvider: Μέσω της συγκεκριμένης υπηρεσίας μπορούμε να συνδέσουμε έναν controller με ένα template, δηλαδή τη σελίδα html στην οποία αντιστοιχεί ή τις ιδιότητές του [55].

Παράδειγμα κλήσης από την εφαρμογή:

```
$stateProvider

      .state('app.map', {

            url: '/map',

            views: {

                  'menuContent': {

                        templateUrl: "templates/map.html",

                        controller: 'MapCtrl'
```

```
                }
            }
        })
```

$urlRouterProvider: Η υπηρεσία $urlRouterProvider είναι επιφορτισμένη με την παρακολούθηση της υπηρεσίας $location και κάθε φορά που αλλάζει τρέχει μία λίστα από κανόνες μέχρι να βρει αυτόν που ταιριάζει [55].

$rootScope: Κάθε εφαρμογή διαθέτει έναν μοναδικό rootscope και όλα τα υπόλοιπα scopes αποτελούν child scopes του rootscope [55].

$timeout: Πρόκειται για wrapper της Angular για την συνάρτηση windows.setTimeout. Καθυστερεί την εκτέλεση μίας συνάρτησης για τον χρόνο που δηλώνουμε ως καθυστέρηση [53].

Παράδειγμα κλήσης από την εφαρμογή:

```
$timeout(function () {
      $location.search('internet', 'true');
}, 1000);
```

Τα αρχεία App.js, Services.js και Controllers.js περιλαμβάνουν providers υλοποιημένους μέσω του module API. Οι τύποι provider που χρησιμοποιήθηκαν είναι οι εξής:

Factory: Δημιουργεί μία νέα υπηρεσία (service) για την εφαρμογή μας που επιστρέφει ένα αντικείμενο, έχοντας μάλιστα τη δυνατότητα να χρησιμοποιήσει άλλες υπηρεσίες υλοποιημένες από εμάς ή έτοιμες, που προσφέρει η apache cordova [55] [56].

Config: Καλείται στην φάση αρχικοποίησης της εφαρμογής, πριν η AngularJS δημιουργήσει και καταστήσει προσπελάσιμες τις services και τους υπόλοιπους providers [55] [56].

Στη συνέχεια παρουσιάζονται τα αρχεία Services.js, App.js και Controllers.js αναλυτικά.

5.4.4.1 Services.js

Το αρχείο Services.js περιλαμβάνει τρεις providers, πιο συγκεκριμένα τρία υλοποιημένα μέσω του module API.

.factory('City', function (){…})

Το factory `City` αποτελεί υπηρεσία που υλοποιήθηκε με στόχο την αποθήκευση των επιλογών του χρήστη στη μνήμη της συσκευής καθώς και την ανάκτησή τους όταν απαιτείται. Αναλυτικότερα όταν συμπεριλαμβάνεται σε κάποιον controller ή service ως dependency ανακτά και αποθηκεύει σε μεταβλητές τις αποθηκευμένες επιλογές του χρήστη από την σελίδα settings.html μέσω των συναρτήσεων που υλοποιούνται εντός του return statement της υπηρεσίας και αναλύονται παρακάτω. Συγκεκριμένα δημιουργήθηκαν οι εξής συναρτήσεις:

`getCity``: function (i)`: Επιστρέφει την τιμή της μεταβλητής στην οποία έχουμε μεταφέρει το όνομα της i πόλης επιλογής του χρήστη, από την μνήμη της συσκευής.

`setCity``: function (city, i)`: Αποθηκεύει την τιμή της μεταβλητής city στην μνήμη της συσκευής, ανανεώνοντας την i πόλη επιλογή του χρήστη.

`getlat``: function (i)`: Επιστρέφει την τιμή της μεταβλητής στην οποία έχουμε μεταφέρει το γεωγραφικό πλάτος της I πόλης επιλογής του χρήστη, από την μνήμη της συσκευής.

`setlat``: function (lat, i)`: Αποθηκεύει την τιμή της μεταβλητής lat στην μνήμη της συσκευής, ανανεώνοντας το γεωγραφικό πλάτος της i πόλης επιλογής του χρήστη.

`getlon``: function (i)`: Επιστρέφει την τιμή της μεταβλητής στην οποία έχουμε μεταφέρει το γεωγραφικό μήκος της i πόλης επιλογής του χρήστη, από την μνήμη της συσκευής.

`setlon``: function (lon, i)`: Αποθηκεύει την τιμή της μεταβλητής lat στην μνήμη της συσκευής, ανανεώνοντας το γεωγραφικό μήκος της i πόλης επιλογής του χρήστη.

`gethours``: function (i)`: Επιστρέφει την τιμή της μεταβλητής στην οποία έχουμε μεταφέρει την επιλογή ώρες του χρήστη για την i πόλη –περιοχή, από την μνήμη της συσκευής.

`sethours``: function (hours, i)`: Αποθηκεύει την τιμή της μεταβλητής hours στην μνήμη της συσκευής, ανανεώνοντας τις ώρες της i πόλης επιλογής του χρήστη.

`getCheckbox``: function (i)`: Επιστρέφει την τιμή της μεταβλητής στην οποία έχουμε μεταφέρει την επιλογή του χρήστη να εμφανίζεται ή όχι στην κεντρική οθόνη η i πόλη –περιοχή, από την μνήμη της συσκευής.

`setCheckbox``: function (checkbox, i)`: Αποθηκεύει την τιμή της μεταβλητής checkbox στην μνήμη της συσκευής, ανανεώνοντας την επιλογή του χρήστη να εμφανίζεται ή όχι στην κεντρική οθόνη η i πόλη επιλογή του χρήστη.

Παράδειγμα κώδικα:

```
getCity: function (i) {
      return serviceCity[i];
},
setCity: function (city, i) {
      serviceCity[i] = city;
      window.localStorage['city' + i] = city;
}
```

.factory('LocationSwap', function ($location) {…})

Το factory `LocationSwap` αποτελεί υπηρεσία που υλοποιήθηκε με στόχο την μεταφορά του χρήστη από την μία html σελίδα στην άλλη, τροποποιόντας κατάλληλα το url, προσθέτωντας τις απαραίτητες GET παραμέτρους. Πιο συγκεκριμένα περιλαμβάνει μία μοναδική συνάρτηση.

`go: function (page, i):` Η συγκεκριμένη συνάρτηση δέχεται δύο παραμέτρους, η πρώτη αφορά την σελίδα στην οποία επιθυμούμε να μεταφερθούμε. Η δεύτερη είναι πάντα null εκτός αν θέλουμε να μεταφερθούμε στην σελίδα του χάρτη. Τότε έχουμε τέσσερις περιπτώσεις:

1. `i='gps'`: Το GPS της συσκευής ήταν απενεργοποιημένο και επιθυμούμε την ανανέωση της σελίδας ώστε να γίνει εκ νέου προσπάθεια εύρεσης της γεωγραφικής θέσης της συσκευής.

2. `i='gps2'`: Το GPS της συσκευής ήταν απενεργοποιημένο, ο χρήστης άνοιξε τις ρυθμίσεις για την ενεργοποίησή του και επιθυμούμε την ανανέωση της σελίδας ώστε να γίνει εκ νέου προσπάθεια εύρεσης της γεωγραφικής θέσης της συσκευής.
3. `i='top3'`: Μας μεταφέρει στο χάρτη με τις ομάδες κεραυνών
4. `i='0','1' ή '2'`: Δηλώνει ποια i πόλη επιλογή του χρήστη θέλουμε να δούμε στον χάρτη.

Παράδειγμα τμήμα κώδικα:

```
switch (page) {
      case 'map':
            if (i == 'gps') {
                  $location.path("app/map").search('gps', 'true');
            } else if (i == 'gps2') {
                  $location.path("app/map").search('gps2', 'true');
            } else if (i == 'top3') {
                  $location.path("app/map").search('top3', 'true');
            } else {
                  $location.search('top3', null);
                  $location.path("app/map").search('i',                    i);
            }
            break;
```

.factory('PopUp', function ($location, $ionicLoading, $ionicPopup, LocationSwap, $timeout) {…})

Το factory PopUp αποτελεί υπηρεσία που υλοποιήθηκε με στόχο την ενημέρωση του χρήστη μέσω pop up μηνύματος όταν η σύνδεση στο διαδίκτυο δεν είναι εφικτή ή διακόπτεται καθώς και όταν το GPS σύστημα της συσκευής είναι απενεργοποιημένο ή δεν καταφέρνει να ολοκληρώσει την εύρεση της τοποθεσίας της συσκευής επιτυχημένα. Αποτελεί και το μοναδικό

σημείο που η εφαρμογή Android διαφοροποιείται από την αντίστοιχη iOS αφού όπως ήδη αναφέραμε το λειτουργικό σύστημα iOS 9 δεν υποστηρίζεται από το 5.4.2.4 Native settings plugin. Πιο συγκεκριμένα το factory PopUp περιλαμβάνει τρεις συναρτήσεις εντός του return statement που αναλύονται παρακάτω. Οι συναρτήσεις **`createPopUp()`** και **`createGpsPopUp(gps)`**, υπεύθυνες για την δημιουργία των μηνυμάτων ενημέρωσης προς τον χρήστη, ανάλογα με το λειτουργικό σύστημα της συσκευής διαχωρίζουν τις ενέργειές τους.

Η παρακάτω γραμμή κώδικα αποθηκεύει στην μεταβλητή `deviceType` το λειτουργικό σύστημα της συσκευής:

```
var deviceType = (navigator.userAgent.match(/iPad/i))  == "iPad" ? "ios" :
(navigator.userAgent.match(/iPhone/i))  == "iPhone" ? "ios" :
(navigator.userAgent.match(/Android/i)) == "Android" ? "Android" : "null";
```

Android:

`createPopUp: function ():` Δημιουργεί pop up μήνυμα προς τον χρήστη όταν δεν υπάρχει σύνδεση στο διαδίκτυο. Ο χρήστης έχει δύο επιλογές:

- Cancel: Μεταφέρεται στην αρχική home.html σελίδα της εφαρμογής με εξαίρεση αν βρίσκεται στην σελίδα settings.html στην οποία θα παραμείνει αλλά οι επιλογές που απαιτούν σύνδεση στο διαδίκτυο δεν μπορούν να επιλεχθούν.
- Ok: Μεταφέρεται στις ρυθμίσεις της συσκευής για τυχόν ενεργοποίηση του wifi ή της χρήσης δεδομένων. Με την επιστροφή του από τις ρυθμίσεις στην εφαρμογή, αν η σύνδεση επανέλθει, η σελίδα θα ανανεωθεί αλλιώς το pop up παραμένει.

`createGpsPopUp: function (gps):` Δημιουργεί pop up μήνυμα προς τον χρήστη όταν το GPS της συσκευής είναι απενεργοποιημένο ή η εύρεση της γεωγραφικής περιοχής της συσκευής είναι αδύνατη. Το μήνυμα τροποποιείται ανάλογα με την παράμετρο gps που δέχεται η συνάρτηση. Ο χρήστης έχει δύο επιλογές:

- Cancel: Το pop up κλείνει και ο χρήστης παραμένει στη σελίδα του χάρτη ο οποίος δείχνει πλέον το κέντρο της Αθήνας ως τοποθεσία.
- Ok: Μεταφέρεται στις ρυθμίσεις της συσκευής για τυχόν ενεργοποίηση του GPS. Με την επιστροφή του από τις ρυθμίσεις στην εφαρμογή, γίνεται εκ νέου προσπάθεια χρήσης του GPS.

iOS:

`createPopUp: function ():` Δημιουργεί pop up μήνυμα προς τον χρήστη όταν δεν υπάρχει σύνδεση στο διαδίκτυο. Ο χρήστης έχει μία επιλογή:

- Ok: Μεταφέρεται στην αρχική home.html σελίδα της εφαρμογής με εξαίρεση αν βρίσκεται στην σελίδα settings.html στην οποία θα παραμείνει αλλά οι επιλογές που απαιτούν σύνδεση στο διαδίκτυο δεν μπορούν να επιλεχθούν.

`createGpsPopUp: function (gps)`: Δημιουργεί pop up μήνυμα προς τον χρήστη όταν το GPS της συσκευής είναι απενεργοποιημένο ή η εύρεση της γεωγραφικής περιοχής της συσκευής είναι αδύνατη. Το μήνυμα τροποποιείται ανάλογα με την παράμετρο gps που δέχεται η συνάρτηση. Ο χρήστης έχει μία επιλογή:

- Ok: Το pop up κλείνει και ο χρήστης παραμένει στη σελίδα του χάρτη ο οποίος δείχνει πλέον το κέντρο της Αθήνας ως τοποθεσία.

Κοινή για Android και iOS:

`hidePopUp: function ()`: Είναι υπεύθυνη για το κλείσιμο οποιουδήποτε pop up εμφανίζεται στην οθόνη αν η λειτουργία του GPS ενεργοποιηθεί ή η σύνδεση στο διαδίκτυο αποκατασταθεί. Παράλληλα απενεργοποιεί και τα μηνύματα που δημιουργούνται από τις συναρτήσεις loadingThunderboltsTemplate() και loadingLocation(). Τέλος, αν ο χρήστης βρίσκεται στην σελίδα settings.html ενεργοποιεί εκ νέου τα στοιχεία που για τη χρησιμοποίησή τους απαιτείται σύνδεση στο διαδίκτυο.

5.4.4.2 App.js

Το αρχείο app.js περιλαμβάνει τρεις providers, πιο συγκεκριμένα δύο factory και μία λετουργία config υλοποιημένα μέσω του module API.

.factory('GoogleMaps', function ($cordovaGeolocation, $location, City, $ionicLoading, PopUp) {…})

Το factory `GoogleMaps` αποτελεί υπηρεσία που υλοποιήθηκε με στόχο την συνολική λειτουργία του χάρτη της εφαρμογής. Είναι υπεύθυνο για την αρχικοποίηση του χάρτη, την προβολή των δεδομένων πάνω σε αυτόν και την τροποποίησή του όποτε χρειάζεται. Οι υπηρεσίες που δέχεται ως ορίσματα dependencies είναι: `$cordovaGeolocation, $location, City, $ionicLoading` και `PopUp`. Αναλυτικότερα περιλαμβάνει τις εξής συναρτήσεις:

`createMap(refresh)`: Δέχεται ως παράμετρο την μεταβλητή **`refresh`** και ανάλογα με την τιμή της, αν ισούται με false δημιουργείται ο χάρτης από την αρχή, ενώ αν είναι `true` σημαίνει ότι έχουμε απλή ανανέωση του χάρτη, δηλαδή του κέντρου του και των δεδομένων που προβάλλονται σε αυτόν.

Παράδειγμα τμήμα κώδικα: Δημιουργία χάρτη

```
mapOptions = {
      center: latLng, zoom: 10,
```

```
        mapTypeId: google.maps.MapTypeId.ROADMAP,
        streetViewControl: false
};
        map = new google.maps.Map(document.getElementById("map"), mapOptions);
```

getMapBounds(): Χρησιμοποιώντας το event listener 'bounds_changed' η συγκεκριμένη συνάρτηση αποθηκεύει στη global μεταβλητή bounds τα όρια του χάρτη κάθε φορά που αλλάζουν.

Παράδειγμα κώδικα:

```
function getMapBounds() {
        google.maps.event.addListener(map, 'bounds_changed', function () {
                bounds = map.getBounds();
        });
}
```

clearMarkers(): Η συνάρτηση αυτή είναι υπεύθυνη για την εκκαθάριση των κεραυνών από το χάρτη καθώς και την αφαίρεση των listener που δημιουργήθηκαν στη συνάρτηση createInfoWindow() κατά την ανανέωσή του.

Παράδειγμα τμήμα κώδικα:

```
map.data.forEach(function (feature) {
        map.data.remove(feature);
});
google.maps.event.removeListener(listenerHandle);
```

setfooter(clear): Η συγκεκριμένη συνάρτηση λαμβάνοντας υπόψη τις GET παραμέτρους του url της σελίδας ορίζει κατάλληλα το κείμενο στο footer ενημερώνοντας τον χρήστη για το πλήθος των κεραυνών στην οθόνη του καθώς και σε πόσες τελευταίες ώρες δημιουργήθηκαν.

setThunderStyle(top3): Η συνάρτηση αυτή είναι υπεύθυνη και επιστρέφει το εικονίδιο με το οποίο θα παρουσιαστεί κάθε κεραυνός ή ομάδα κεραυνών πάνω στον χάρτη. Η παράμετρος top3 που δέχεται ως όρισμα είναι `true` αν πρόκειται για χάρτη με τις ομάδες κεραυνών και `false` για χάρτη με κεραυνούς.

Για τις ομάδες κεραυνών χρησιμοποιούμε την εικόνα /img/marker.png.

Όσον αφορά όμως την απεικόνιση των κεραυνών, στο φάκελο img/thunder_icons υπάρχουν 24 εικόνες .png με ονόματα 00.png-23.png. Κάθε εικόνα αποτελείται από ένα κεραυνό σε διαφορετικό χρώμα, ξεκινώντας από τη 00.png που είναι σκούρο μπλε και φτάνοντας στην 23.png που είναι κόκκινο. Η συνάρτηση αντιστοιχεί το εικονίδιο ενός κεραυνού σύμφωνα με το πόσες ώρες πέρασαν από την καταγραφή του, με το κόκκινο να αποτελεί το πιο πρόσφατο και το μπλέ το παλαιότερο.

Παράδειγμα τμήμα κώδικα:

```
var pngName = parseInt(feature.getProperty('date').substring(11, 13).valueOf(), 10) + hoursTo23;

if (pngName > 23) {

	pngName = pngName - 23;

}

pngName = (pngName < 10 ? "0" : "") + pngName;  //get HH format

return {

	icon: './img/thunder_icons/' + pngName + '.png'

};
```

`createInfoWindow()`: Η συνάρτηση αυτή δημιουργεί το παράθυρο με τις πληροφορίες του κεραυνού ή της ομάδας κεραυνών όταν ο χρήστης πατήσει πάνω στο εικονίδιό τους. Για να επιτευχθεί το παραπάνω δημιουργεί και το κατάλληλο “click” listener.

`updateGeoJson()`: Είναι υπεύθυνη για την ανανέωση των δεδομένων στον χάρτη. Αυτό συνεπάγεται τον ορισμό του κατάλληλου url για το HTTP αίτημα στην RESTful υπηρεσία που έχουμε δημιουργήσει στον server http://195.251.31.119 και στη συνέχεια τη χρησιμοποίησή του μέσω της `map.data.loadGeoJson(apiUrl)`.Επιπλέον είναι επιφορτισμένη με την κλήση των συναρτήσεων `clearMarkers()` για την εκκαθάριση τυχόν προηγούμενων δεδομένων στον χάρτη, `setThunderStyle()` για τον ορισμό του κατάλληλου εικονιδίου για κάθε κεραυνό ή ομάδα κεραυνών και `createInfoWindow()` για τη δημιουργία των απαραίτητων παραθύρων πληροφορίας.

`loadingLocationTemplate()`: Με την κλήση της εμφανίζεται μήνυμα στο χρήστη με το οποίο ενημερώνεται ότι η εφαρμογή εντοπίζει την τοποθεσία στο χάρτη και απενεργοποιεί οποιαδήποτε αλληλεπίδραση μέχρι την ολοκλήρωση της διαδικασίας.

`loadingThunderboltsTemplate ():` Με την κλήση της εμφανίζεται μήνυμα στο χρήστη με το οποίο ενημερώνεται ότι η εφαρμογή φορτώνει τα δεδομένα των κεραυνών στο χάρτη και απενεργοποιεί οποιαδήποτε αλληλεπίδραση μέχρι την ολοκλήρωση της διαδικασίας ή το πέρας πέντε δευτερολέπτων.

`gpsOnSuccess: function (position,refresh):` Καλείται όταν η εφαρμογή χρησιμοποιεί την λειτουργία GPS της συσκευής για την εύρεση της τοποθεσίας του χρήστη και επιτυγχάνει. Ορίζει στη μεταβλητή `latLng` τις συντεταγμένες και καλεί την `createMap(refresh)` για τη δημιουργία του χάρτη.

`gpsError: function (refresh,error,gps):` Καλείται όταν η εφαρμογή χρησιμοποιεί την λειτουργία GPS της συσκευής για την εύρεση της τοποθεσίας του χρήστη και αποτυγχάνει. Ορίζει στη μεταβλητή `latLng` τις συντεταγμένες του κέντρου της Αθήνας και καλεί την `createMap(refresh)` για τη δημιουργία του χάρτη. Στη συνέχεια εμφανίζει αντίστοιχο μήνυμα pop up στον χρήστη για ενεργοποίηση του GPS στην περίπτωση που ήταν απενεργοποιημένο ή απλή ενημέρωση αν δεν ήταν δυνατή η εύρεση της τοποθεσίας παρότι η λειτουργία GPS ήταν ενεργοποιημένη.

`initMap: function (refresh):` Η συγκεκριμένη συνάρτηση υλοποιείται μέσα σε return statement ώστε να είναι δυνατή η κλήση της από άλλες υπηρεσίες services και πιο συγκεκριμένα την `Connection`. Αποτελεί την κεντρική συνάρτηση της υπηρεσίας `GoogleMaps` και είναι υπεύθυνη ανάλογα με τις GET παραμέτρους του url της σελίδας, για την κλήση των κατάλληλων συναρτήσεων για την αρχικοποίηση ή ανανέωση του χάρτη. Οι πιθανές περιπτώσεις είναι τρεις:

- Χάρτης περιοχής της επιλογής του χρήστη, αν η GET παράμετρος i δεν είναι null
- Χάρτης των περιοχών με τους περισσότερους κεραυνούς, αν οι GET παράμετροι i, top3 είναι null
- Χάρτης με τη χρήση GPS για τον εντοπισμό της γεωγραφικής θέσης της συσκευής, αν η GET παράμετρος top3 δεν είναι null

.factory('Connection', function ($location , $rootScope, GoogleMaps, PopUp) {…})

Το factory `Connection` αποτελεί υπηρεσία που υλοποιήθηκε με στόχο την παρακολούθηση της κατάστασης της σύνδεσης της συσκευής στο διαδίκτυο και την ενημέρωση του χρήστη αν η

σύνδεση διακοπεί. Οι υπηρεσίες που δέχεται ως ορίσματα είναι: `$location , $rootScope, GoogleMaps, PopUp`. Αναλυτικότερα περιλαμβάνει τις εξής συναρτήσεις:

`checkLoaded(refresh):` Όταν ο χρήστης επιλέξει να δει κάποιο χάρτη καθώς και όταν η σύνδεση στο διαδίκτυο επανέρχεται ελέγχει για την σωστή φόρτωση του 5.4.3.1 Google Maps Javascript API, ανάλογα και με την τιμή της παραμέτρου refresh που δέχεται. Πιο συγκεκριμένα ελέγχει αν η τιμή `google.maps` είναι δηλωμένη, δηλαδή αν έχει φορτωθεί το API για τους χάρτες και στη συνέχεια ελέγχει αν έχει φορτωθεί σωστά, δηλαδή αν επιστρέφει ένα αντικείμενο. Στη περίπτωση που το script δεν έχει φορτωθεί, προστίθεται στο <head> της σελίδας map.html.

Παράδειγμα κώδικα:

```
if (google.maps) {
      if (typeof google === 'object' && typeof google.maps === 'object' && GoogleMaps.mapOptions!==null && GoogleMaps.map!==null) {
            google.maps.event.addDomListener(window, 'load', GoogleMaps.initMap(refresh));
      }else{
            GoogleMaps.initMap(false);
      }
}else{
      var script = document.createElement("script");
      script.type = "text/javascript";
      script.onload = function () {
            script.onload = null; // Cleanup onload handler
            GoogleMaps.initMap(false);
      }
      script.src = "https://maps.googleapis.com/maps/api/js?key=AIzaSyC1RlpsPiVY1X4AR5l2xPKD3A9bRkf3oq4";
      // Add the script to the DOM
      (document.getElementsByTagName("head")[ 0 ]).appendChild(script);
}
```

`doWhenOffline():` Η συνάρτηση αυτή αναλαμβάνει τις εργασίες που πρέπει να γίνουν όταν δεν υπάρχει ή διακόπτεται η σύνδεση στο διαδίκτυο. Συγκεκριμένα εμφάνιση αντίστοιχου μηνύματος και προτροπή του χρήστη να μεταφερθεί στις ρυθμίσεις της συσκευής.

Παράδειγμα κώδικα:

```
function doWhenOffline() {
	if(alertPopup===null){
		PopUp.createPopUp();
	}
}
```

doWhenOnline(refresh): Η συνάρτηση αυτή αναλαμβάνει τις εργασίες που πρέπει να γίνουν όταν υπάρχει ή επανέρχεται η σύνδεση στο διαδίκτυο. Συγκεκριμένα απόκρυψη του μηνύματος ενημέρωσης του χρήστη και ανανέωση της σελίδας του χάρτη αν ο χρήστης βρισκόταν εκεί.

Παράδειγμα κώδικα:

```
function doWhenOnline(refresh) {
	PopUp.hidePopUp();
	if ($location.path()==="/app/map") {
		checkLoaded(false);
	}
}
```

addConnectivityListeners(): Η συγκεκριμένη συνάρτηση παρακολουθεί την κατάσταση της σύνδεσης στο διαδίκτυο, παρατηρώντας τη τιμή της $cordovaNetwork. Αυτή εναλλάσσεται ανάμεσα σε online και offline. Όταν η σύνδεση διακόπτεται ο χρήστης ενημερώνεται με αντίστοιχο μήνυμα που τον προτρέπει να μεταφερθεί στις ρυθμίσεις του κινητού του ενώ όταν η σύνδεση επανέρχεται το μήνυμα αποκρύπτεται και η σελίδα που βρισκόταν ο χρήστης ανανεώνεται.

Παράδειγμα κώδικα:

```
$rootScope.$on('$cordovaNetwork:online', function () {
	doWhenOnline(false);
	if ($location.path()!=="/app/map"){
		$location.search('internet', 'true');
	}});
$rootScope.$on('$cordovaNetwork:offline', function () {
	doWhenOffline();
});
```

`init: function (refresh):` Τέλος η συνάρτηση init δηλώνεται μέσα σε return statement ώστε να είναι δυνατή η κλήση της από τους controllers. Ελέγχει την αρχική κατάσταση της σύνδεσης με τη βοήθεια της υπηρεσίας `$cordovaNetwork` και αναλόγως καλεί την `doWhenOnline(refresh)` ή την `doWhenOffline()`. Επίσης καλεί την `addConnectivityListeners()` ώστε να ενεργοποιηθούν οι listeners που ανιχνεύουν αλλάγές στην κατάσταση της σύνδεσης.

Παράδειγμα κώδικα:

```
return {
	init: function (refresh) {
	if ($cordovaNetwork.isOnline()) { //internet enabled
		doWhenOnline(refresh);
	} else { //internet disabled
		doWhenOffline();
	}
	addConnectivityListeners(refresh);
	}
};
```

.config(function ($stateProvider, $urlRouterProvider) {…})

Το αρχείο app.js όπως ήδη αναφέραμε εκτός των δύο factory περιλαμβάνει ακόμη μία λειτουργία config. Πιο συγκεκριμένα στην εφαρμογή μας με τη χρήση της συγκεκριμένης config λειτουργίας επιτυγχάνουμε τη σύνδεση της κάθε σελίδας html με τον αντίστοιχο controller. Έχουμε έξι συναρτήσεις $stateProvider.state('X', {}), μία για κάθε μία σελίδα html της εφαρμογής μας.

Παράδειγμα κώδικα: Για τη σελίδα home.html

```
.state('app.home', {
	url: "/home",
	views: {
		'menuContent': {
			templateUrl: "templates/home.html",
			controller: 'HomeCtrl'
		}
	}
})
```

5.4.4.3 Controllers.js

Το αρχείο controllers.js περιλαμβάνει πέντε controllers οι οποίοι αναλύονται στην συνέχεια και αφορούν κυρίως την διαχείριση των scopes της κάθε σελίδας html.

.controller('MapCtrl', function ($scope, $location, City, Connection, LocationSwap) {…})

Πρόκειται για τον controller που είναι συνδεδεμένος με το template map.html. Καλεί για τον έλεγχο της σύνδεσης στο διαδίκτυο και την αρχικοποίηση της σελίδας μέσω της συνάρτησης `Connection.init(false)`, μεταφέρει στο scope τις συναρτήσεις της υπηρεσίας `LocationSwap` ενώ ορίζει και τον τίτλο της σελίδας που εμφανίζεται στον χρήστη αν έχει επιλέξει κάποια περιοχή πόλη της επιλογής του. Ακόμη με τη χρήση της `$scope.$on('$locationChangeSuccess', function () {…})` επιτυγχάνεται η ανανέωση του χάρτη όταν αυτό είναι απαραίτητο.

.controller('Top3Ctrl', function ($scope, $http, $location, Connection, LocationSwap) {…})

Πρόκειται για τον controller που είναι συνδεδεμένος με το template top3.html. Καλεί για τον έλεγχο της σύνδεσης στο διαδίκτυο και την αρχικοποίηση των δεδομένων της σελίδας μέσω της συνάρτησης `Connection.init(false)`, μεταφέρει στο scope τις συναρτήσεις της υπηρεσίας `LocationSwap` και ορίζει τον τίτλο της σελίδας. Διαθέτει επίσης μία συνάρτηση:

`getTop3Cities ()`: Η συνάρτηση στέλνει ένα HTTP αίτημα, στην RESTful υπηρεσία που έχουμε υλοποιήσει στον http://195.251.31.119, λαμβάνοντας ως απάντηση ένα GeoJSON αρχείο με μέχρι τρεις ομάδες κεραυνών. Οι ομάδες αυτές και οι ιδιότητές τους μεταφέρονται στον πίνακα top3[] του scope ώστε να είναι προσβάσιμα από την σελίδα top3.html.

`$scope.$on('$locationChangeSuccess', function () {…})`: Ακόμη με την χρήση της, καλείται εκ νέου η συνάρτηση getTop3Cities (). Έτσι επιτυγχάνεται η ανανέωση της σελίδας όταν η σύνδεση στο διαδίκτυο είχε διακοπεί και επανέρχεται.

.controller('SettingsCtrl', function ($scope, City, $http, $location, Connection, LocationSwap) {…})

Πρόκειται για τον controller που είναι συνδεδεμένος με τα templates settings.html και search.html. Καλεί για τον έλεγχο της σύνδεσης στο διαδίκτυο και την αρχικοποίηση των

δεδομένων της σελίδας μέσω της συνάρτησης `Connection.init(false)`, μεταφέρει στο scope τις συναρτήσεις της υπηρεσίας `LocationSwap`, τις τιμές των πόλεων επιλογών του χρήστη, την τιμή της GET παραμέτρου του url και τέλος ορίζει τον τίτλο της σελίδας. Περιλαμβάνει τις εξής συναρτήσεις που επιθυμούμε να έχουμε πρόσβαση από τις σελίδες html:

`$scope.clearInput=function(i):` Η συνάρτηση `clearInput()` καλείται στην σελίδα search.html όταν ο χρήστης πατήσει το κουμπί εκκαθάρισης της φόρμας.

`$scope.hide=function(i):` Η συγκεκριμένη συνάρτηση καλείται για κάθε πόλη i και επιστρέφει `true` ή `false` με στόχο την εμφάνιση των συμπληρωμένων πόλεων επιλογών του χρήστη και άλλη μία προς συμπλήρωση, σε σύνολο μέχρι τρεις.

`$scope.clearCity=function(i):` Καλείται για την διαγραφή της i πόλης επιλογής όταν ο χρήστης το επιλέξει από την σελίδα settings.html.

`$scope.setOrder=function():` Η συνάρτηση `setOrder()` είναι υπεύθυνη για την εμφάνιση σειράς των πόλεων επιλογών του χρήστη. Για παράδειγμα, αν έχει αποθηκευμένες τρεις επιλογές και διαγράψει την δεύτερη, θέλουμε την επόμενη φορά που θα εισέλθει ο χρήστης στις ρυθμίσεις, το ελεύθερο πεδίο να έρθει τρίτο στη σειρά.

`$scope.checkboxChecked = function (i):` Ελέγχει και επιστρέφει `true` ή `false` ανάλογα με την τιμή του checkbox της i πόλης.

`$scope.assignCheckbox = function (i):` Αλλάζοντας ο χρήστης την επιλογή checkbox της i πόλης από την σελίδα settings.html, η συνάρτηση ανανεώνει την τιμή στη μνήμη της συσκευής.

`$scope.assignHours = function (selectedHours, i):` Αλλάζοντας ο χρήστης την επιλογή ωρών της i πόλης από την σελίδα settings.html, η συνάρτηση ανανεώνει την τιμή στη μνήμη της συσκευής.

`$scope.assignLatLon = function (cityName,i):` Η συνάρτηση καλείται όταν ο χρήστης ανανεώσει το όνομα της i πόλης επιλογής μέσω της σελίδας serach.html. Στέλνει ένα HTTP αίτημα στην geocode υπηρεσία της Google και ανανεώνει τις τιμές του `$scope.City[i]` και της μνήμης της συσκευής.

`$scope.findAddresses = function (i):` Για κάθε ένα γράμμα που πληκτρολογεί ο χρήστης στη σελίδα search.html, η συνάρτηση `findAddresses(i)` στέλνει ένα HTTP αίτημα στην

autocomplete υπηρεσία της Google. Λαμβάνει ως απάντηση τις προβλέψεις για τη είσοδο που έχει παράσχει ο χρήστης και του τις εμφανίζει για να επιλέξει από αυτές.

`$scope.$on('$locationChangeSuccess', function () {..})` : Τέλος, με τη χρήση της επιτυγχάνεται η ανανέωση της πόλης επιλογής του χρήστη στην μνήμη της συσκευής, όταν αυτή αλλάζει από την σελίδα search.html, αλλά και στο αντίστοιχο πεδίο της σελίδας settings.html. Πιο συγκεκριμένα όταν ο χρήστης επιλέξει μία από τις προβλέψεις που προσφέρονται στην σελίδα search.html μεταφέρεται πίσω στην σελίδα settings.html. Στο url όμως πλέον υπάρχει η GET παράμετρος name εκτός της i. Η παράμετρος name αποθηκεύεται πλέον ως η πόλη επιλογή του χρήστη και παράλληλα καλείται η συνάρτηση `$scope.assignLatLon (cityName,i)` για την εύρεση της γεωγραφικής θέσης της επιλογής του χρήστη.

.controller('HomeCtrl', function ($ionicSideMenuDelegate, $scope, LocationSwap, $timeout) {…})

Πρόκειται για τον controller που είναι συνδεδεμένος με την αρχική σελίδα home.html. Μεταφέρει στο scope τις συναρτήσεις της υπηρεσίας `LocationSwap` και ορίζει τον τίτλο της σελίδας. Περιλαμβάνει μία συνάρτηση:

`$scope.onTouch = function (id,i)`: Η συγκεκριμένη συνάρτηση καλείται όταν ο χρήστης πατήσει ένα από τα διαθέσιμα κουμπιά της σελίδας. Προσθέτει ένα εφέ hover στο επιλεγμένο κουμπί για 0.075 δευτερόλεπτα και μεταφέρει τον χρήστη στην σελίδα που αιτήθηκε.

.controller('AppCtrl', function ($scope, City) {…})

Πρόκειται για τον controller που είναι συνδεδεμένος με το μενού δηλαδή το template menu.html. Μεταφέρει στο scope τις τιμές των πόλεων επιλογών του χρήστη και περιλαμβάνει δύο σχετικές συναρτήσεις:

`$scope.updatedCityName = function (i)`: Ανανεώνει και επιστρέφει την τιμή του ονόματος της i πόλης επιλογής του χρήστη στο scope όταν αυτή αλλάξει ώστε να είναι άμεσα ενημερωμένη στο μενού.

`$scope.updatedCheckbox = function (i)`: Ανανεώνει και επιστρέφει την τιμή του checkbox της i πόλης επιλογής του χρήστη στο scope όταν αυτή αλλάξει ώστε να εμφανιστεί ή να αποκρυφθεί άμεσα η πόλη από το μενού.

Κεφάλαιο 6

Συμπεράσματα και επεκτάσεις

6.1 Συμπεράσματα

Με την ολοκλήρωση της παρούσας εργασίας καταλήγουμε στο συμπέρασμα πως η επιλογή δημιουργίας της εφαρμογής ως υβριδικής ήταν επιτυχημένη, αφού με τη βοήθεια όλων των εργαλείων που αναλύσαμε στα προηγούμενα κεφάλαια καταφέραμε να υλοποιήσουμε την εφαρμογή τόσο για συσκευές με λειτουργικό Android όσο και iOS, καλύπτοντας σε πολύ μεγάλο βαθμό τους χρήστες smartphone.

Είναι γεγονός ότι η προετοιμασία για την υλοποίηση, δηλαδή η εγκατάσταση των απαραίτητων εργαλείων και SDKs για τη κάθε πλατφόρμα είναι περίπλοκη και μερικές φορές δύσκολο από άποψη υλικού. Για παράδειγμα για την υλοποίηση σε λειτουργικό iOS απαιτείται λειτουργικό σύστημα OS X καθώς και η εγκατάσταση του εργαλείου ανάπτυξης κώδικα Xcode. Ωστόσο τα frameworks που χρησιμοποιήσαμε παρέχουν όλα τα απαραίτητα έγγραφα ώστε όσα προβλήματα προκύψουν να ξεπεραστούν.

Κατά την ανάπτυξη της εφαρμογής-πελάτη αλλά και της RESTful API υπηρεσίας ήρθαμε σε επαφή με νέες τεχνολογίες και τεχνικές, εμβαθύναμε στον τρόπο λειτουργίας τους και αποκτήσαμε εμπειρία τόσο στο στήσιμο του απαιτούμενου περιβάλλοντος αλλά και στην σωστή χρήση τους.

6.2 Επεκτάσεις

Σίγουρα υπάρχουν επεκτάσεις που θα μπορούσαν να υλοποιηθούν για την διεύρυνση της εφαρμογής. Μερικές από τις αλλαγές που θα προτείναμε και θα μπορούσαν να γίνουν στο μέλλον, αποσκοπούν σε μεγαλύτερο κοινό, σε διαφορετικό τρόπο απεικόνισης της πληροφορίας ή την πληροφόρηση του χρήστη για περισσότερα φυσικά φαινόμενα.

6.2.1 Επέκταση στην υπόλοιπη Ευρώπη

Η εφαρμογή αυτή την στιγμή απευθύνεται κυρίως στο ελληνικό κοινό και σε κατοίκους πόλεων των γειτονικών χωρών που βρίσκονται κοντά στα σύνορα, σε ακτίνα περίπου 600 χιλιομέτρων από το Χαροκόπειο Πανεπιστήμιο. Αν λοιπόν τα δεδομένα μας προέρχονταν από περισσότερους

αισθητήρες σε άλλες πόλεις της Ευρώπης, η εφαρμογή θα μπορούσε να απευθύνεται στους κατοίκους όλης της Ηπείρου.

6.2.2 Διαφορετική απεικόνιση των ομάδων των κεραυνών

Αντί να παρουσιάζεται το κέντρο των ομάδων κεραυνών στον χρήστη, με κατάλληλες τροποποιήσεις στην RESTful υπηρεσία μας, θα μπορούσαμε να αποθηκεύουμε σε ποια ομάδα ανήκει κάθε κεραυνός και στην συνέχεια να εμφανίζουμε το σχήμα της ομάδας στο χάρτη.

6.2.3 Εξαγωγή και παρουσίαση στατιστικών

Αυτή τη στιγμή στη βάση δεδομένων μας δεν κρατάμε δεδομένα παλαιότερα των 30 ημερών. Ωστόσο στον server όπως αναφέραμε κρατάμε αρχεία τύπου .csv με τα δεδομένα κάθε ημέρας. Με την κατάλληλη ανάλυσή τους θα μπορούσαμε να εξάγουμε και να παρουσιάζουμε στον χρήστη διαφόρων ειδών στατιστικά. Μερικά από αυτά θα μπορούσαν να είναι: η συχνότητα με την οποία δημιουργείται ένας κεραυνός κάθε μήνα του χρόνου, ο μέσος όρος κεραυνών σε μία περιοχή ανά μήνα ή χρόνο κα.

6.2.4 Επέκταση σε διαφορετικά φυσικά φαινόμενα

Μία αρκετά εύκολη τροποποίηση θα αφορούσε την επέκταση της εφαρμογής σε περισσότερα φυσικά φαινόμενα που πληρούν την προϋπόθεση ότι συμβαίνουν σε ένα συγκεκριμένο γεωγραφικό μήκος και πλάτος για μία στιγμή ή περίοδο. Ένα τέτοιο φυσικό φαινόμενο είναι οι σεισμοί. Είναι λοιπόν πολύ εύκολο αν έχουμε πρόσβαση σε μία βάση δεδομένων που καταγράφει τους σεισμούς σε πραγματικό χρόνο με ελάχιστες τροποποιήσεις στην RESTful υπηρεσία και την εφαρμογή μας να τους παρουσιάσουμε στον χρήστη ή να τους ομαδοποιήσουμε.

6.2.5 Δημιουργία λειτουργίας παρασκηνίου για προειδοποίηση του χρήστη

Τέλος, η προσθήκη μιας λειτουργίας παρασκηνίου που θα προειδοποιούσε τον χρήστη σε περίπτωση κεραυνών στην περιοχή γύρω του, θα ήταν επίσης μία πιθανή επέκταση. Με αυτό τον τρόπο ο χρήστης θα μπορούσε να ενημερώνεται χωρίς να χρειάζεται να ανοίξει ο ίδιος την εφαρμογή.

Βιβλιογραφία

[1] «Smartphone,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Smartphone. [Πρόσβαση 23 Μαίου 2016].

[2] «Smartphone OS market share,» [Ηλεκτρονικό]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Πρόσβαση 23 Ιουνίου 2016].

[3] Gartner Inc, 2013. [Ηλεκτρονικό]. Available: http://www.gartner.com/newsroom/id/2324917. [Πρόσβαση 27 Ιουνίου 2016].

[4] «Blitzortung Lightning Monitor,» [Ηλεκτρονικό]. Available: https://play.google.com/store/apps/details?id=org.blitzortung.android.app. [Πρόσβαση 23 Ιουνίου 2016].

[5] H. R. G. Bennet A. J., «Variability in surface atmospheric electric field measurements,» *Journal of Physics,* τόμ. Conference Series 142, 2008.

[6] P. Arason, «Comparison of Data from a Lightning Location System and Atmospheric Parameters from a Numerical Weather Prediction Model,» Avignon France, 2004.

[7] «Κεραυνοί,» [Ηλεκτρονικό]. Available: http://www.meteo.gr/pdf/lightning.pdf. [Πρόσβαση 24 Ιουνίου 2016].

[8] «Meteoclima,» [Ηλεκτρονικό]. Available: http://meteoclima.hua.gr/index.php?option=com_content&view=article&id=157&Itemid=134. [Πρόσβαση 25 Ιουνίου 2016].

[9] Γ. Νικόλαος, «Ανάπτυξη αλγορίθμου για την αποτύπωση και ανάλυση της κεραυνικής δραστηριότητας στον Ελληνικό χώρο,» Αθήνα, 2013.

[10] «Ionic framework,» [Ηλεκτρονικό]. Available: http://ionicframework.com/docs/guide/preface.html. [Πρόσβαση 24 Μαίου 2016].

[11] «Quirksmode,» 2015. [Ηλεκτρονικό]. Available: http://www.quirksmode.org/blog/archives/2015/05/web_vs_native_l.html. [Πρόσβαση 30 Ιουνίου 2016].

[12] [Ηλεκτρονικό]. Available: http://www.json.org/json-el.html. [Πρόσβαση 30 Μαίου 2016].

[13] «JSON,» [Ηλεκτρονικό]. [Πρόσβαση 30 Μαίου 2016].

[14] [Ηλεκτρονικό]. Available: http://geojson.org/geojson-spec.html . [Πρόσβαση 17 Ιουνίου 2016].

[15] «MySQL,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/MySQL. [Πρόσβαση 30 Μαίου 2016].

[16] «PhpMyAdmin,» [Ηλεκτρονικό]. Available: https://www.phpmyadmin.net. [Πρόσβαση 30 Μαίου 2016].

[17] «API,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Application_programming_interface . [Πρόσβαση 4 Ιουνίου 2016].

[18] «Google APIs,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Google_APIs . [Πρόσβαση 4 Ιουνίου 2016].

[19] «REST,» 4 Ιουνίου 2016. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer .

[20] @. T. F. Pearson eCollege, «REST API Tutorial,» Αύγουστος 2013. [Ηλεκτρονικό]. Available: http://www.restapitutorial.com/resources.html. [Πρόσβαση 4 Ιουνίου 2016].

[21] «DBScan,» σε *Εισαγωγή στην εξόρυξη και τις αποθήκες δεδομένων*, Εκδόσεις Νέων Τεχνολογιών, Έκδοση 1η, σελίδες 176-177, 200-204, 2008.

[22] M. Ester, H.-P. Kriegel, J. Sander και X. Xu, «A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining,» AAAI Press, Μόναχο, 1996.

[23] «DBScan Wikipedia,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/DBSCAN. [Πρόσβαση 27 Ιουνίου 2016].

[24] «Netbeans,» [Ηλεκτρονικό]. Available: https://netbeans.org/features/. [Πρόσβαση 25 Ιουνίου 2016].

[25] «MySQL Workbench,» [Ηλεκτρονικό]. Available: https://www.mysql.com/products/workbench/. [Πρόσβαση 25 Ιουνίου 2016].

[26] «Framework Software,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Software_framework . [Πρόσβαση 28 Μαίου 2016].

[27] «Rhino,» [Ηλεκτρονικό]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino#Rhino_downloads. [Πρόσβαση 26 Ιουνίου 2016].

[28] «CodeIgniter,» [Ηλεκτρονικό]. Available: http://www.codeigniter.com/user_guide/. [Πρόσβαση 30 Μαίου 2016].

[29] «Node.js,» [Ηλεκτρονικό]. Available: https://nodejs.org/en/ . [Πρόσβαση 24 Μαίου 2016].

[30] «Chrome's V8 JavaScript engine,» [Ηλεκτρονικό]. Available: https://developers.google.com/v8/ . [Πρόσβαση 25 Μαίου 2016].

[31] «Npm js,» [Ηλεκτρονικό]. Available: https://www.npmjs.com/ . [Πρόσβαση 24 Μαίου 2016].

[32] «Short Historey of Git,» [Ηλεκτρονικό]. Available: https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git. [Πρόσβαση 25 Ιουνίου 2016].

[33] «Eclipse Community Survey 2014 Results,» 2014. [Ηλεκτρονικό]. Available: https://ianskerrett.wordpress.com/2014/06/23/eclipse-community-survey-2014-results/. [Πρόσβαση 25 Ιουνίου 2016].

[34] «Introduction,» [Ηλεκτρονικό]. Available: https://docs.angularjs.org/guide/ . [Πρόσβαση 28 Μαίου 2016].

[35] «Overview,» [Ηλεκτρονικό]. Available: https://cordova.apache.org/docs/en/latest/guide/overview/index.html. [Πρόσβαση 24 Μαίου 2016].

[36] «Android Platform Guide,» [Ηλεκτρονικό]. Available: https://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html. [Πρόσβαση 25 Ιουνίου 2016].

[37] «iOS Platform Guide,» [Ηλεκτρονικό]. Available: https://cordova.apache.org/docs/en/latest/guide/platforms/ios/index.html. [Πρόσβαση 25 Ιουνίου 2016].

[38] «Ionic overview,» [Ηλεκτρονικό]. Available: http://ionicframework.com/docs/overview/#license. [Πρόσβαση 24 Μαίου 2016].

[39] «MIT License,» [Ηλεκτρονικό]. Available: https://opensource.org/licenses/MIT. [Πρόσβαση 24 Μαίου 2016].

[40] «Android distribution 2015,» [Ηλεκτρονικό]. Available: http://www.androidauthority.com/android-distribution-august-2015-631077/. [Πρόσβαση 26 Ιουνίου 2016].

[41] «Android Dasgboards,» [Ηλεκτρονικό]. Available: https://developer.android.com/about/dashboards/index.html. [Πρόσβαση 26 Ιουνίου 2016].

[42] «Apple developer,» [Ηλεκτρονικό]. Available: https://developer.apple.com/support/app-store/. [Πρόσβαση 26 Ιουνίου 2016].

[43] C. S., «DBScan υλοποίηση σε γλώσσα Javascript,» [Ηλεκτρονικό]. Available: https://github.com/upphiminn/jDBSCAN. [Πρόσβαση 12 Ιουνίου 2016].

[44] «Whitelist plugin,» [Ηλεκτρονικό]. Available: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-whitelist/ . [Πρόσβαση 17 Ιουνίου 2016].

[45] «Network information pluginl,» [Ηλεκτρονικό]. Available: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-network-

information/index.htm. [Πρόσβαση 17 Ιουνίου 2016].

[46] «Geolocation plugin,» [Ηλεκτρονικό]. Available: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-geolocation/index.html . [Πρόσβαση 17 Ιουνίου 2016].

[47] «Native settings plugin,» [Ηλεκτρονικό]. Available: https://github.com/selahssea/Cordova-open-native-settings . [Πρόσβαση 17 Ιουνίου 2016].

[48] «Splashscreen plugin,» [Ηλεκτρονικό]. Available: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-splashscreen/ . [Πρόσβαση 17 Ιουνίου 2016].

[49] «Google Maps Javascript API,» [Ηλεκτρονικό]. Available: https://developers.google.com/maps/documentation/javascript/basics. [Πρόσβαση 1 Ιουνίου 2016].

[50] «Google Maps Geocoding API,» [Ηλεκτρονικό]. Available: https://developers.google.com/maps/documentation/geocoding/start . [Πρόσβαση 17 Ιουνίου 2016].

[51] «Google Places API Web Service,» [Ηλεκτρονικό]. Available: https://developers.google.com/places/web-service/intro . [Πρόσβαση 17 Ιουνίου 2016].

[52] «Place Autocomplete,» [Ηλεκτρονικό]. Available: https://developers.google.com/places/web-service/autocomplete . [Πρόσβαση 17 Ιουνίου 2016].

[53] «Services,» [Ηλεκτρονικό]. Available: https://docs.angularjs.org/api/ng/service . [Πρόσβαση 28 Μαίου 2016].

[54] «Ionic Loading service,» [Ηλεκτρονικό]. Available: http://ionicframework.com/docs/api/service/$ionicLoading/. [Πρόσβαση 19 Ιουνίου 2016].

[55] «URL Routing - AngularJS,» [Ηλεκτρονικό]. Available: https://github.com/angular-ui/ui-router/wiki/URL-Routing. [Πρόσβαση 25 Ιουνίου 2016].

[56] «Providers AngularJS,» [Ηλεκτρονικό]. Available: https://docs.angularjs.org/guide/providers . [Πρόσβαση 17 Ιουνίου 2016].

[57] «Live Lightning,» [Ηλεκτρονικό]. Available: https://itunes.apple.com/us/app/live-lightning/id541468577?mt=8. [Πρόσβαση 23 Ιουνίου 2016].

Πίνακας εικόνων

Παραρτήματα

Παράρτημα Α: Ψευδοκώδικας αλγορίθμου DBScan [22]

```
DBSCAN(D, eps, MinPts) {
   C = 0
   for each point P in dataset D {
      if P is visited
         continue next point
      mark P as visited
      NeighborPts = regionQuery(P, eps)
      if sizeof(NeighborPts) < MinPts
         mark P as NOISE
      else {
         C = next cluster
         expandCluster(P, NeighborPts, C, eps, MinPts)
      }
   }
}


expandCluster(P, NeighborPts, C, eps, MinPts) {
   add P to cluster C
   for each point P' in NeighborPts {
      if P' is not visited {
         mark P' as visited
         NeighborPts' = regionQuery(P', eps)
         if sizeof(NeighborPts') >= MinPts
            NeighborPts = NeighborPts joined with NeighborPts'
      }
      if P' is not yet member of any cluster
```

```
          add P' to cluster C
    }
}


regionQuery(P, eps)
    return all points within P's eps-neighborhood (including P)
```