Πανεπιστήμιο Ιωαννίνων

Πληροφορικής & Τηλεπικοινωνιών

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Διαδικτυακός Ταξιδιωτικός οδηγός για κινητά τηλέφωνα

Νικόλαος Κοντοθανάσης Α.Μ. 14025

Επιβλέπων: Ιωάννης Τσούλος

Επίκουρος Καθηγητής, M.Sc., Ph.D. Πανεπιστήμιο Ιωαννίνων

Άρτα, Ιανουάριος, 2019

ΔΙΑΔΙΚΤΥΑΚΟΣ ΤΑΞΙΔΙΩΤΙΚΟΣ ΟΔΗΓΟΣ ΓΙΑ ΚΙΝΗΤΑ ΤΗΛΕΦΩΝΑ

NETWORKING TRAVEL ASSISTANT FOR SMARTPHONES

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 2019

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής
   Όνομα Επίθετο,
   Τίτλος, βαθμίδα

2. Επιβλέπων καθηγητής
   Όνομα Επίθετο,
   Τίτλος, βαθμίδα

3. Επιβλέπων καθηγητής
   Όνομα Επίθετο,
   Τίτλος, βαθμίδα

Ο/Η Προϊστάμενος/η του Τμήματος

Όνομα Επίθετο

Τίτλος, βαθμίδα

Υπογραφή

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Επίθετο, Όνομα:

Κοντοθανάσης Νικόλαος

Υπογραφή:

Κοντοθανάσης Ν.

Ημερομηνία:

Δευτέρα, 11 Φεβρουαρίου 2019

ΠΕΡΙΛΗΨΗ

Το θέμα της παρούσας πτυχιακής εργασίας είναι η κατασκευή ενός διαδικτυακού ταξιδιωτικού οδηγού για κινητά τηλέφωνα. Η πτυχιακή εργασία εστιάζει στον τρόπο κατασκευής μιας εφαρμογής που μπορεί να χρησιμοποιηθεί ως ταξιδιωτικός οδηγός, συγκεκριμένα σε iOS συσκευές, με την λήψη απαραίτητων πληροφοριών για τα σημεία ενδιαφέροντος από το διαδίκτυο σε μορφή JSON, καθώς και την ενσωμάτωση ενός μοντέλου ταξινόμησης φωτογραφιών για την άμεση αναγνώριση σημείων ή αντικειμένων ενδιαφέροντος όσον αφορά τις ληφθείσες τοποθεσίες. Για την διεκπεραίωση της εργασίας, χρησιμοποιήθηκε ως κύριο εργαλείο το προγραμματιστικό περιβάλλον της εταιρίας Apple, με γλώσσα ανάπτυξης της εφαρμογής την Swift 4.2. Για να πραγματοποιηθεί η σωστή λειτουργία της εφαρμογής χρησιμοποιήθηκαν επίσης υπηρεσίες όπως είναι το Firebase, Firebase Database, Alamofire και Geofire για την αποστολή/λήψη και διαχείριση δεδομένων μέσω του διαδικτύου. Πιο συγκεκριμένα η υπηρεσία Firebase που προφέρεται δωρεάν από την Google παρέχει την βάση δεδομένων , η Firebase Database για την δημιουργία και τη διαχείριση σε απευθείας σύνδεση και σε πραγματικό χρόνο της βάσης που χρειάζεται η εφαρμογή. Επίσης η υπηρεσία Alamofire για την λήψη του αρχείου JSON, ένας δυναμικός τρόπος λήψης πληροφορίων και τέλος η υπηρεσία Geofire για την δημιουργία και διαχείριση τοποθεσιών με συντεταγμένες. Η υπηρεσία Geofire είναι επέκταση της Firebase.
Όλες οι απαραίτητες γνώσεις για τις διαδικασίες πηγάζουν από τον επίσημο οδηγό της Apple για τους προγραμματιστές της.

Λέξεις-κλειδιά: Προγραμματισμός, Κινητά τηλέφωνα iPhone, Xcode, Swift, Παγκόσμιο Σύστημα Στιγματοθέτησης, Machine Learning

ABSTRACT

The theme of this thesis is to build an online travel guide application for mobile phones. The thesis focuses on how to build an application that can be used as a travel guide, specifically on iOS devices, by gathering necessary information about Web sites in JSON format, as well as integrating a photo sorting model for instant recognition points or objects of interest in the locations received. In order to build this application, Apple's programming environment was used as the main tool, and Swift 4.2 as the programming language. In addition, services such as Firebase, Firebase Database, Alamofire and Geofire were also used to send and receive data over the Internet. In particular, the Firebase service, pronounced free of charge by Google, provides the Firebase in order to create and manage on-line and real-time the database that the application needs. Also, the Alamofire service, a dynamic way to get information for downloading the JSON file, and finally the Geofire service to create and manage coordinates. Geofire is an extension of Firebase. All the necessary knowledge about the process stems from the official Apple driver for its developers.

Keywords: Programming, mobile phones, smartphones,GPS, iPhone, Xcode, Swift, Machine Learning

Περιεχόμενα

Κεφάλαιο 1

1.1. Εισαγωγή

Δισεκατομμύρια άνθρωποι χρησιμοποιούν κινητές συσκευές σε όλο τον κόσμο. Μέχρι σήμερα χρησιμοποιούνται πάνω από επτά δισεκατομμύρια κινητές συσκευές με συνεχή πρόσβαση στο Διαδίκτυο. Το κινητό τηλέφωνο δεν αποτελούν πλέον μόνο ένα τηλέφωνο, αλλά μια φορητή συσκευή πολυμέσων. Η συνεχής ζήτηση από τους χρήστες για υπηρεσίες διαδικτύου, πολυμέσα και άλλες εφαρμογές, οδήγησαν στην ταχεία βελτίωση των κινητών τηλεφώνων. Η ταχεία αύξηση της υπολογιστικής ισχύος, της χωρητικότητας μνήμης, του μεγέθους της οθόνης και της ανάλυσης αυξάνει τις δυνατότητες των συσκευών. Οι προγραμματιστές εφαρμογών για κινητά τηλέφωνα επωφελούνται από αυτές τις βελτιώσεις κατά τη δημιουργία νέων εφαρμογών, προσφέροντας προηγμένες δυνατότητες στους χρήστες τόσο στην προσωπική όσο και την επαγγελματική τους ζωή. Υπάρχει πλέον η δυνατότητα τα κινητά τηλέφωνα να έχουν ενσωματωμένες υπηρεσίες GPS και εφαρμογές που εντοπίζουν την τοποθεσία ή που χρησιμοποιούν την τοποθεσία του χρήστη για να προσφέρουν μια πληθώρα άλλων υπηρεσιών.

1.2. Δομή της εργασίας

Η εργασία είναι χωρισμένη σε 6 κεφάλαια τα οποία είναι δομημένα ως εξής:

- Στο δεύτερο κεφάλαιο γίνεται εισαγωγή μια ιστορική αναδρομή στην εξέλιξη των κινητών τηλεφώνων, εμβαθύνοντας στα τηλέφωνα της εταιρίας Apple, τα iPhone. Επίσης παρατίθενται χρήσιμες πληροφορίες για το προγραμματιστικό περιβάλλον Xcode.
- Το τρίτο κεφάλαιο αφορά τη γλώσσα που έχει αναπτύξει η εταιρία Apple, την Swift, με βάση τον προκάτοχό της την Objective-C. Αναφέρονται επίσης παραδείγματα χρήσης την γλώσσας από γνωστές εφαρμογές και τέλος τα πλεονεκτήματα της γλώσσας Swift.
- Το τέταρτο κεφάλαιο αφορά το πακέτο λογισμικού Core ML, την δημιουργία και εκπαίδευση εξατομικευμένου μοντέλου ταξινόμησης φωτογραφιών και τα αρχεία τύπου JSON και την χρήση τους μέσα στο Xcode.

- Το πέμπτο κεφάλαιο αναφέρεται στην τεχνολογία των GPS και την χρήση του, εμβαθύνει στην χρήση των τοποθεσιών του GPS με το Xcode για την ανάπτυξη εφαρμογών που κάνουν χρήση τοποθεσίας. Τέλος γίνεται αναφορά στο Apple Maps, μια εφαρμογή χαρτογράφησης για την πλοήγηση του χρήστη, δημιουργημένη από την Apple.
- Η πτυχιακή εργασία ολοκληρώνεται με το έκτο κεφάλαιο, στο οποίο παρουσιάζεται ο κώδικας της εφαρμογής ενός διαδικτυακού ταξιδιωτικού οδηγού για κινητά τηλέφωνα iOS.

ΚΕΦΑΛΑΙΟ 2

2.1. Ιστορία των κινητών τηλεφώνων

2.1.1. Η εξέλιξη των κινητών τηλεφώνων ως την έλευση των smartphones

Η ιστορία των δημόσιων κινητών τηλεφώνων αρχίζει στη δεκαετία του 1940 μετά τον Δεύτερο Παγκόσμιο Πόλεμο. Παρόλο που υπήρχαν πρωτόγονα κινητά τηλέφωνα και πριν από τον πόλεμο, χρησιμοποιήθηκαν περισσότερο σαν αμφίδρομοι ραδιοφωνικοί πομποί από την κυβέρνηση ή τη βιομηχανία, ενώ η μετάδοση της κλήσης γινόταν χειροκίνητα μέσω του τηλεφωνικού δικτύου χερσαίων γραμμών. Μετά τον Δεύτερο Παγκόσμιο Πόλεμο, θα μπορούσαν τελικά να αντιμετωπιστούν οι παραμελημένες τα προηγούμενα χρόνια ανάγκες για επικοινωνία. Πολλές πόλεις μετατράπηκαν σε ερείπια. Οι υποδομές τους χρειάζονταν χρόνια
ώστε να ανασυγκροτηθούν. Οι ταχυδρομικές, τηλεφωνικές, τηλεγραφικές υπηρεσίες και οι ιδιωτικές τηλεφωνικές εταιρείες επικεντρώθηκαν στην παροχή σταθερών τηλεφώνων και υπηρεσιών, ωστόσο οι έρευνες σχετικά μετά κινητά τηλέφωνα συνεχίστηκαν. Οι Αμερικανοί οδηγούν αυτό το κίνημα “χαμηλής προτεραιότητας”. Οι Ηνωμένες Πολιτείες ήταν φυσικά άθικτες μετά τον πόλεμο, τα Bell Telephone Laboratories είχαν μια μεγάλη ομάδα ραδιομηχανών και επιστημόνων και η εταιρεία Motorola είχε μεγαλώσει σημαντικά κατά τη διάρκεια του Β 'Παγκοσμίου Πολέμου. Η ζήτηση των καταναλωτών, οι ερευνητικές εγκαταστάσεις και οι ικανότητες κατασκευής σχετικά με την κινητή τηλεφωνία υπήρχαν στις ΗΠΑ. Αλλά ήταν αρκετά; Και τι είδους κινητό σύστημα θα δημιουργηθεί;
Στις 28 Ιουλίου 1945 περιγράφηκε για πρώτη φορά ένα κυψελοειδές σύστημα. Ο επικεφαλής της Ομοσπονδιακής Επιτροπής Επικοινωνιών των Ηνωμένων Πολιτειών (FCC), περιέγραψε ένα κυψελοειδές σύστημα που θα είχε την επαναχρησιμοποίηση συχνότητας σε μια μικρή περιοχή, ως το κύριο στοιχείο του. Εκατομμύρια χρήστες, όπως είπε, θα μπορούσαν να χρησιμοποιούν τα ίδια κανάλια σε ολόκληρη τη χώρα. Οι μεταδότες χαμηλής ισχύος που χρησιμοποιούν ραδιοσυχνότητες υψηλής συχνότητας θα έδιναν τη δυνατότητα στα σήματα κοντινών πόλεων να παρεμβαίνουν μεταξύ τους. Παρά τον αρχικό ενθουσιασμό του Jett, η FCC δεν χορήγησε ποτέ το φάσμα που απαιτείται για την υπηρεσία αυτή. Βέβαια, οι ραδιομηχανικοί συνέχισαν να

σκέφτονται ένα τέτοιο σύστημα, ακόμα κι αν δεν ήταν εύκολο να επιτευχθεί τη δεδομένη χρονική στιγμή.

Περίπου ένα χρόνο μετά, ξεκίνησε η πρώτη αμερικανική εμπορική κινητή ραδιοτηλεφωνική υπηρεσία. Στις 17 Ιουνίου 1946 στο Saint Louis, Missouri, η AT & T και μία από τις περιφερειακές τηλεφωνικές εταιρείες, η South-Western Bell ξεκίνησαν τη λειτουργία της πρώτης υπηρεσίας κινητής τηλεφωνίας (1). Η Motorola δημιούργησε τους φορητούς ασύρματους και η Bell System ανέλαβε την εγκατάσταση. Το μοντέλο λειτουργούσε ως εξής: Μια κεραία είναι κεντρικά τοποθετημένη και τα κινητά τηλέφωνα κινούνται σε μια μεγάλη περιοχή γύρω από αυτή. Τα κινητά τηλέφωνα, μεταδίδουν σε αρκετούς δέκτες που βρίσκονται γύρω από την πόλη. Η κίνηση από τους δέκτες στον πομπό γίνεται από ένα χειριστή σε ένα κεντρικό τηλεφωνικό κέντρο. Χρησιμοποιήθηκαν έξι κανάλια στη ζώνη των 150 MHz με απόσταση μεταξύ τους 60 kHz. Οι απρόσμενες παρεμβολές μεταξύ των καναλιών ανάγκασαν τη Bell System να χρησιμοποιήσει μόνο τρία κανάλια. Λίστες αναμονής αναπτύχθηκαν αμέσως σε κάθε μία από τις είκοσι πέντε πόλεις.

Τον Δεκέμβριο του 1947 ο DH Ring της Bell Laboratories, με τη βοήθεια της WR Young, ανέπτυξε ένα αληθινό κυψελοειδές ραδιοσύστημα για κινητή τηλεφωνία (2). Το σύστημα αποτελούνταν από μικρές γεωγραφικές περιοχές που ονομάζονται κελιά, από έναν πομπό βάσης σε κάθε μια, η κυψελοειδής κυκλοφορία ελέγχεται από έναν κεντρικό διακόπτη, οι συχνότητες επαναχρησιμοποιούνται από διαφορετικά κελιά και ούτω καθεξής.

Το 1947 η Bell Systems ζήτησε από την FCC περισσότερες συχνότητες. Η Επιτροπή διέθεσε μερικά ακόμη κανάλια το 1949, αλλά έκανε και κάτι αναπάντεχο. Έδωσαν τις μισές από αυτές τις συχνότητες σε άλλες εταιρείες που επιθυμούσαν να πουλήσουν υπηρεσίες κινητής τηλεφωνίας. Έτσι δημιουργήθηκε ανταγωνισμός επιτρέποντας παράλληλα την αύξηση της χωρητικότητας. Ως απόδειξη της ανταγωνιστικότητάς τους, αυτές οι μικρές επιχειρήσεις εξυπηρέτησαν 80.000 κινητές μονάδες μέχρι το 1978, διπλάσιες της AT & T. Την 1η Μαρτίου 1948 άρχισε να λειτουργεί η πρώτη πλήρως αυτόματη ραδιοτηλεφωνική υπηρεσία στο Ρίτσμοντ της Ινδιάνας (3) ενώ η AT & T δεν παρείχε αυτόματη κλήση μέχρι το 1964. Όμως, ενώ αυτές οι μικρές, ανεξάρτητες ασύρματες εταιρείες μπορούσαν να παρέχουν υπηρεσίες σε λίγες δεκάδες

πελάτες κάθε φορά, δεν είχαν τα χρήματα ή τους πόρους για έρευνα, το σχεδιασμό και στη συνέχεια την κατασκευή ενός συστήματος κινητής τηλεφωνίας υψηλής χωρητικότητας.

Την 1η Ιουλίου 1948, η Bell System αποκάλυψε το τρανζίστορ, μια κοινή εφεύρεση των επιστημόνων των Bell Laboratories William Shockley, John Bardeen και Walter Brattain. Θα φέρει επανάσταση σε κάθε πτυχή της τηλεφωνικής βιομηχανίας και όλων των επικοινωνιών. Οι εύθραυστοι και ογκώδεις σωλήνες κενού θα τελικά αντικατασταθούν από τρανζίστορ. Ωστόσο, οι σωλήνες κενού θα κυριαρχούν στη βιομηχανία τηλεφώνου για άλλα είκοσι χρόνια.

Εκτός των Ηνωμένων Πολιτειών, η ανάπτυξη της κινητής τηλεφωνίας ήρθε αργά. Οι περισσότερες κυβερνήσεις δεν επέτρεψαν τα δημόσια ραδιοτηλέφωνα. Υπήρχαν εξαιρέσεις. Το 1949, το ολλανδικό εθνικό ραδιοτηλεφωνικό δίκτυο εγκαινίασε το πρώτο παγκόσμιο δημόσιο σύστημα ραδιοτηλεφωνίας παγκοσμίως. Και το 1951επιστήμονες στη Σουηδία σχεδίασαν ένα νέο αυτόματο σύστημα κινητής τηλεφωνίας. Ένα παρόμοιο σύστημα δημιουργήθηκε σύντομα στο Γκέτεμποργκ. Όπως και με όλα τα ραδιοτηλέφωνα, ο εξοπλισμός ήταν τεράστιος και απαιτούσε μεγάλη ισχύ.

Στην άλλη πλευρά του πλανήτη, ένας γίγαντας γεννιέται. Το 1952 η Ιαπωνία κέρδισε την ανεξαρτησία της, επτά χρόνια μετά τον Δεύτερο Παγκόσμιο Πόλεμο. Η Nippon Telephone and Telegraph ιδιωτικοποιήθηκε, ενισχύθηκε ο κλάδος της έρευνας και διάφορα κρατικά επιχορηγούμενα εργαστήρια δίνουν τη δυνατότητα για σπουδές σχετικές με την τηλεφωνία. Παρόλο που δεν επιτρέπονται τα ιδιωτικά ραδιοτηλέφωνα, η ζήτηση των καταναλωτών για ραδιοφωνικά και τηλεοπτικά προϊόντα θα έφθανε γρήγορα και οι Ιάπωνες σύντομα θα προσπαθούσαν να χρησιμοποιήσουν αυτό τον εξοπλισμό για εξαγωγές.

Τον Ιούλιο του 1958 ο Jack Kilby εφηύρε το ολοκληρωμένο κύκλωμα στην Texas Instruments στο Ντάλας του Τέξας. Ένα τεμάχιο γερμανίου μεγέθους δοντιού περιείχε το πλήρες ηλεκτρονικό κύκλωμα. Έδειξε επίσης πώς οι αντιστάτες, οι πυκνωτές, οι δίοδοι και τα τρανζίστορ θα μπορούσαν να συνυπάρχουν σε ένα μόνο μπλοκ ημιαγωγών και ότι όλα θα μπορούσαν να κατασκευαστούν από το ίδιο υλικό. Όπως λέει και η Texas Instruments, "Οι ρίζες σχεδόν κάθε ηλεκτρονικής συσκευής που θεωρούμε δεδομένη σήμερα μπορούν να αναχθούν στο Ντάλλας πριν από περισσότερα από 40 χρόνια".

Το 1958 η καινοτόμος εταιρεία ραδιοτηλεφώνων Richmond βελτίωσε το σύστημα αυτόματης κλήσης. Πρόσθεσαν νέα χαρακτηριστικά σε αυτό, συμπεριλαμβανομένων των άμεσων κινητών επικοινωνιών. Άλλες ανεξάρτητες τηλεφωνικές εταιρείες πραγματοποίησαν παρόμοια, σταδιακή πρόοδο στην κινητή τηλεφωνία κατά τη δεκαετία του 1950 και του 1960.

Στα τέλη της δεκαετίας του 1950 ολοκληρώθηκε μικρή έρευνα και ανάπτυξη κυψελοειδούς ραδιοφώνου, αλλά δεν ήταν οικονομικά εφικτό να κατασκευαστεί ένα κυψελοειδές σύστημα μεγάλης χωρητικότητας στις Ηνωμένες Πολιτείες. Παρόλα αυτά, δύο σημαντικές εργασίες των υπαλλήλων της Bell System δημοσιεύθηκαν το 1960. Τα άρθρα ασχολήθηκαν με τη μεταβίβαση μιας κλήσης από το ένα κύτταρο στο άλλο, με διαφορετικές συχνότητες που χρησιμοποιούνται σε γειτονικά κύτταρα (4). Αυτή ήταν η πρώτη φορά που ολόκληρη η έννοια του κυτταρικού συστήματος περιγράφηκε σε έντυπη μορφή στο αναγνωστικό κοινό παγκοσμίως.

Το 1961, η θυγατρική της Svenska Radio Aktieboget της Ericsson, αναδιοργανώθηκε για να επικεντρωθεί στην κατασκευή ραδιοσυστημάτων. Η SRA θα συνεχίσει να αποτελεί κεντρικό τμήμα της Ericsson, βοηθώντας στη δημιουργία της ασύρματης επικοινωνίας.

Το 1964 η Bell System εισήγαγε την Improved Mobile Telephone Service (IMTS), με το οποίο αντικατέστησαν το πεπαλαιωμένο Σύστημα Κινητής Τηλεφωνίας (5). Με το IMTS οι άνθρωποι δεν έπρεπε να πατούν ένα κουμπί για να μιλήσουν. Οι συζητήσεις μπορούσαν να πραγματοποιηθούν όπως σε ένα κανονικό τηλέφωνο. Το IMTS έκανε τελικά εφικτή την άμεση κλήση, την επιλογή αυτόματου καναλιού και το μειωμένο εύρος ζώνης από 25 έως 30 kHz. Ορισμένες περιφερειακές τηλεφωνικές εταιρείες όπως η Pacific Bell, που ανήκει στην AT & T, χρειάστηκαν σχεδόν είκοσι χρόνια για να αντικαταστήσουν τα παλαιά τους συστήματα. Και πάλι, αν και η ζήτηση ήταν μεγάλη, δεν υπήρχαν αρκετά κανάλια για να φιλοξενήσουν περισσότερους χρήστες.

Άλλες χώρες στα μέσα της δεκαετίας του '60 αντικατέστησαν επίσης τα πρώτα τους συστήματα κινητής τηλεφωνίας, όπως η σουηδική υπηρεσία τηλεπικοινωνιών. Ο Ragnar Berglund ανέπτυξε ένα νέο σύστημα, χάρη στο οποίο κατέστη δυνατή η χρήση μικρότερων τηλεφώνων που απαιτούσαν λιγότερη ισχύ και επομένως ήταν λιγότερο δαπανηρά.

Το 1967 η Nokia επέκτεινε τη διεύρυνση του τμήματος ηλεκτρονικών καλωδίων μιας Φινλανδικής Καλωδιακής εταιρείας ώστε να συμπεριλάβει έρευνα σχετική με τους ημιαγωγούς. Αυτές οι μελέτες στις αρχές της δεκαετίας του 1970 βοήθησαν την Nokia να αναπτύξει ψηφιακούς σταθερούς τηλεφωνικούς διακόπτες.

Στα τέλη της δεκαετίας του 1960 είναι βέβαιο ότι κάθε μεγάλη εταιρεία τηλεπικοινωνιών γνώριζε για την ιδέα του κυψελοειδούς συστήματος. Το 1967, για παράδειγμα, η NT & T ξεκίνησε έρευνα για ένα παγκόσμιο κυψελοειδές σύστημα στα 900 MHz για την Ιαπωνία (6). Αλλά πώς θα μπορούσε να λειτουργήσει τεχνικά και οικονομικά μια τέτοια προσπάθεια; Δεν υπήρχε τρόπος να

εξελιχθεί η υπάρχουσα υποδομή ραδιοτηλεφώνων σε κυψελοειδή. Απαιτούνταν νέος εξοπλισμός για τους σταθμούς βάσης και νέα κινητά τηλέφωνα για τους πελάτες. Αντί για μία κεντρική κεραία με έναν αρκετά απλό πομποδέκτη, θα απαιτούνταν πλέον αρκετές έως δεκάδες κυψέλες, οι οποίες θα χρειάζονταν το δικό τους πομποδέκτη, οι οποίοι θα έπρεπε να είναι διασυνδεμένοι μεταξύ τους με ένα διακόπτη δικτύου για τη διαχείριση της κυκλοφορίας, ώστε τα πάντα να λειτουργούν. Το κόστος θα ήταν τεράστιο.

Η Ομοσπονδιακή Επιτροπή Επικοινωνιών στις Ηνωμένες Πολιτείες το 1968 επανεξέτασε το δεκαετές αίτημα της Bell Systems για περισσότερες συχνότητες. Με μια προσωρινή απόφαση το 1970 αποφάσισαν να τους παραχωρήσουν τις συχνότητες και έτσι η Bell Systems υπέβαλε μια πρόταση για ένα κυψελοειδές σύστημα επαναχρησιμοποίησης συχνότητας.

Τον Ιανουάριο του 1969 η Bell Systems καθιέρωσε για πρώτη φορά το εμπορικό κυψελοειδές ραδιοτηλέφωνο, χρησιμοποιώντας τη μέθοδο επαναχρησιμοποίησης της συχνότητας σε ένα σύστημα μικρών ζωνών. Οι επιβάτες των τρένων Metroliner που εκτελούσε δρομολόγια μεταξύ New York City και Washington, DC διαπίστωσαν ότι θα μπορούσαν να πραγματοποιούν τηλεφωνικές κλήσεις, ενώ κινούνται σε περισσότερα από 160 χιλιόμετρα την ώρα.

Η εποχή του σωλήνα για τα ραδιοτηλέφωνα τελείωσε. Το σήμα «Mark 12» της Motorola ήταν ένα τηλέφωνο IMTS που σχεδιάστηκε για να λειτουργεί στη ζώνη των 450 MHz. Αυτό το τρανζίστορ ήταν ακόμα ογκώδες και τοποθετήθηκε σε ένα όχημα. Είναι η χρονική στιγμή που κάνουν την εμφάνισή τους τα πρώτα εμπορικά φορητά ραδιοτηλέφωνα στις Ηνωμένες Πολιτείες. Το 1969 ή το 1970, η SCM Melabs, που ανήκει στον Smith Corona, παρήγαγε ένα τηλέφωνο Attache, ένα πλήρες τηλέφωνο MTS ενσωματωμένο σε ένα χαρτοφύλακα. Σχεδόν αμέσως η Canyon Communications και η Livermore Data βγήκαν με τα τηλέφωνα τους. Όλα αυτά ήταν MTS, παρόλο που το IMTS είχε εισαχθεί το 1964. Μόνο μικρές επιχειρήσεις έκαναν αυτές τις μονάδες. Η Harris, η Motorola και η GE δεν το έκαναν ποτέ. Όλα αυτά τα τηλέφωνα έγιναν ουσιαστικά με το χέρι (7).

Τον Νοέμβριο του 1971 η Intel εισήγαγε τον πρώτο εμπορικό μικροεπεξεργαστή, το 4004, έναν μικροσκοπικό υπολογιστή σε τσιπ πυριτίου. Περιείχε 2.300 μετασχηματιστές και έκανε 60.000 λειτουργίες το δευτερόλεπτο. Οι σημερινοί μικροεπεξεργαστές μπορούν να περιέχουν 5,5 εκατομμύρια τρανζίστορ, εκτελώντας εκατοντάδες εκατομμύρια υπολογισμών κάθε δευτερόλεπτο. Το 4004 της Intel σχεδιάστηκε αρχικά για έναν επιτραπέζιο υπολογιστή, αλλά οι

μικροεπεξεργαστές βελτιώθηκαν σύντομα και τελικά τέθηκαν σε όλα τα είδη ηλεκτρονικών, συμπεριλαμβανομένων των κινητών τηλεφώνων.

Στις 17 Οκτωβρίου 1973, η Motorola υπέβαλε δίπλωμα ευρεσιτεχνίας για το δικό της κυψελοειδές ραδιοσύστημα (8). Παρόλο που η Motorola είχε προμηθεύσει τη Bell Systems με ραδιοτηλέφωνα για δεκαετίες, η AT & T θεωρήθηκε πλέον απειλή και όχι φίλος. Έτσι ένας ανταγωνισμός αναπτύχθηκε μεταξύ των δύο εταιρειών. Το 1973, αφού ολοκλήρωσε το πρώτο κινητό τηλέφωνο της η Motorola, ο Δρ. Martin Cooper κάλεσε τους ανταγωνιστές του στο Bell Labs. Ο Ferranti λέει ότι "ο Cooper δεν μπόρεσε να αντισταθεί στο να καταδείξει με πολύ πρακτικό τρόπο ποιος είχε κερδίσει" (9). Η ομάδα του Cooper εφηύρε το πρώτο συμπαγές και εύχρηστο φορητό κινητό τηλέφωνο, αλλά όχι το ίδιο το κινητό τηλέφωνο. Αυτό είχε ήδη γίνει.

Την 1η Μαΐου 1974 η FCC ενέκρινε επιπλέον 115 megahertz φάσματος για μελλοντική χρήση κινητού τηλεφώνου. Η Cellular βγαίνει μπροστά, και οι Αμερικανοί κατασκευαστές ραδιοτηλεφώνων και οι επιχειρήσεις αρχίζουν να σχεδιάζουν για το μέλλον. Το 1976 μόνο 545 πελάτες στη Νέα Υόρκη είχαν κινητά τηλέφωνα Bell System, με 3700 πελάτες στη λίστα αναμονής. Στις Ηνωμένες Πολιτείες συνολικά, 44.000 συνδρομητές είχαν κινητά τηλέφωνα, αλλά 20.000 άτομα βρίσκονταν σε λίστες αναμονής πέντε έως δέκα ετών. Η ζήτηση υπήρχε, αλλά το ραδιοτηλεοπτικό φάσμα για να τα φιλοξενήσει δεν υπήρχε.

Το 1975 η FCC άφησε τη Bell System να ξεκινήσει μια δοκιμή. Όμως, μέχρι το Μάρτιο του 1977, η FCC δεν ενέκρινε το αίτημα της AT & T για λειτουργία του κυψελοειδούς συστήματος. Μια νέα βιομηχανία ασύρματης επικοινωνίας αναπτύχθηκε στην Αμερική και η FCC προσπάθησε να ελέγξει κάθε πτυχή της. Θα αποφάσιζαν τον αριθμό των ασύρματων μεταφορέων σε κάθε αγορά, τις εταιρείες που θα επέτρεπαν να λειτουργούν, τα πρότυπα για τον εξοπλισμό, τις εκχωρήσεις συχνοτήτων, τα διαστήματα των καναλιών, και τα επόμενα βήματα (10).

Με το πρόβλημα της γραφειοκρατίας να μην είναι τόσο μεγάλο, οι κατασκευαστές Ιαπωνίας και Σκανδιναβίας δοκίμασαν τα πρώτα εμπορικά αναλογικά κυψελοειδή συστήματα. Η ομάδα NMT πραγματοποίησε μια ικανοποιητική δοκιμή στη Στοκχόλμη στα τέλη του 1977 έως τις αρχές του 1978. Η Nippon Telephone and Telegraph πιθανότατα ξεκίνησε πειράματα στο Τόκιο ήδη από το 1975 (11).

Η NTT παρήγαγε τα πρώτα κυψελοειδή συστήματα για την Ιαπωνία, χρησιμοποιώντας όλο τον ιαπωνικό εξοπλισμό. Οι Ιάπωνες συνέβαλαν επίσης σημαντικές μελέτες στην έρευνα σχετικά με το κυψελοειδές σύστημα.

Στα μέσα και στα τέλη της δεκαετίας του 1970, ο στόχος της Ιαπωνίας να παράγει ηλεκτρονικά αγαθά χωρίς ελαττώματα ανάγκασε τους κατασκευαστές σε ολόκληρο τον κόσμο να αναρωτηθούν αν μπορούσαν να ανταγωνιστούν.

Τον Μάιο του 1978 η τηλεφωνική εταιρεία Batelco άρχισε να λειτουργεί το πρώτο εμπορικό κινητό τηλεφωνικό σύστημα. Το απλό σύστημα δύο κυψελών είχε 250 συνδρομητές, λειτουργούσε σε 20 κανάλια στη ζώνη των 400 MHz και χρησιμοποίησε όλο τον εξοπλισμό Matsushita (Panasonic) (12).

Τον Ιούλιο του 1978 η Advanced Mobile Phone Service (AMPS) άρχισαν να λειτουργούν κοντά σε δύο αμερικανικές πόλεις. Δέκα κύτταρα που κάλυπταν 21.000 τετραγωνικά μίλια αποτελούσαν το σύστημα του Σικάγου. Η Oki Electric παρείχε τα κινητά τερματικά. Η δοκιμή ξεκίνησε με 90 υπαλλήλους της Bell System που δρούσαν ως πελάτες. Μετά από έξι μήνες, στις 20 Δεκεμβρίου 1978, άρχισε μια δοκιμή αγοράς με συνδρομητές. Το σύστημα χρησιμοποίησε τη νεοσυσταθείσα ζώνη 800 MHz (13). Το πρώιμο δίκτυο, η χρήση ολοκληρωμένων κυκλωμάτων μεγάλης κλίμακας, ο αποκλειστικός υπολογιστής και το σύστημα μεταγωγής, τα έτοιμα κινητά τηλέφωνα και οι κεραίες, έδειξαν ότι θα μπορούσε να λειτουργήσει ένα μεγάλο κυψελοειδές σύστημα.

Η παγκόσμια εμπορική κυψελοειδής ανάπτυξη συνέβη στα τέλη της δεκαετίας του 1970 και στη συνέχεια συνεχίστηκε στις αρχές της δεκαετίας του 1980. Ένα σύστημα 88 κυψελών ξεκίνησε τον Δεκέμβριο του 1979 στο Τόκυο. Το πρώτο εμπορικό σύστημα της Βόρειας Αμερικής ξεκίνησε τον Αύγουστο του 1981 στην πόλη του Μεξικού. Το πρώτο παγκόσμιο δίκτυο κινητής τηλεφωνίας του Βορρά ξεκίνησε την 1η Σεπτεμβρίου του 1981 στη Σαουδική Αραβία. Χρησιμοποιούσε 20 κελιά και λειτουργούσε στα 450 MHz. Τον επόμενο μήνα, ξεκινώντας από την 1η Οκτωβρίου 1981, και προχωρώντας σταδιακά έως τον Μάρτιο του 1982, η Σουηδία, η Νορβηγία, η Δανία και η Φινλανδία άρχισαν να λειτουργούν ένα σκανδιναβικό δίκτυο NMT. Λειτουργούσε επίσης στα 450 MHz και χρησιμοποιούσε τρεις διακόπτες της Ericsson. Το πρώτο πολυεθνικό κυψελοειδές σύστημα, το NMT450 είχε 600 κύτταρα. Καθώς οι Σκανδιναβοί λειτουργούσαν το πιο εξελιγμένο κυψελοειδές σύστημα στον κόσμο, η ανάπτυξη του κυψελοειδούς ραδιοτηλεφώνου στην Αμερική σταμάτησε και πάλι από την κυβερνητική γραφειοκρατία.

Στις 25 Μαρτίου 1980, ο Richard Anderson, γενικός διευθυντής της Data Division της Hewlett-Packard, συγκλόνισε τους Αμερικανούς παραγωγούς τσιπ, λέγοντας ότι η εταιρεία του θα αγόραζε πλέον τα περισσότερα τσιπ από την Ιαπωνία. Η HP ανακάλυψε ότι οι αμερικανικές κάρτες μνήμης

είχαν ποσοστό αποτυχίας έξι φορές μεγαλύτερο από τον χειρότερο ιαπωνικό κατασκευαστή. Οι αμερικανικές επιχειρήσεις δεν ήταν μόνο που χρειάζονταν να ανανεώσουν.

Το 1987, η Panasonic ανέλαβε ένα εργοστάσιο της Ericsson στο Kumla της Σουηδίας, 120 μίλια δυτικά της Στοκχόλμης. Οι Meurling και Jeans εξηγούν: "Η Panasonic έφερε εντελώς νέα πρότυπα ποιότητας" (14).

Στις αρχές της δεκαετίας του 1980 η τηλεπικοινωνία άλλαξε για πάντα. Το 1982, η Bell Systems, που εξυπηρετούσε το 80% του αμερικανικού πληθυσμού διαλύθηκε.

Νέες εταιρείες, προϊόντα και υπηρεσίες εμφανίστηκαν αμέσως σε όλους τους τομείς της αμερικανικής τηλεπικοινωνίας, καθώς ένα νέο, ανταγωνιστικό πνεύμα σάρωνε τη χώρα. Το κλείσιμο της AT & T οδήγησε τα έθνη σε όλο τον κόσμο να επανεξετάσουν τις κρατικές και λειτουργικές τηλεφωνικές εταιρείες τους, με στόχο την ενίσχυση του ανταγωνισμού στις χώρες τους.

Η Ευρώπη είδε το κυψελοειδές σύστημα το 1981, όταν άρχισε να λειτουργεί το Δίκτυο κινητής τηλεφωνίας Nordic ή NMT450 στη Δανία, τη Σουηδία, τη Φινλανδία και τη Νορβηγία στην περιοχή των 450 MHz. Ήταν το πρώτο πολυεθνικό κυψελοειδές σύστημα. Το 1985 η Μεγάλη Βρετανία άρχισε να χρησιμοποιεί το σύστημα επικοινωνιών συνολικής πρόσβασης ή το TACS στα 900 MHz. Αργότερα, η γερμανική C-Netz, η γαλλική Radiocom 2000 και η ιταλική RTMI / RTMS βοήθησαν στην κατασκευή εννέα ασυμβίβαστων αναλογικών ραδιοτηλεφωνικών συστημάτων στην Ευρώπη.

Τα πρώτα τηλέφωνα C-Netz, TAC, εισήχθησαν για πρώτη φορά εμπορικά στη Βαλτιμόρη και στην Ουάσιγκτον DC. Η AMPS και η Dyna-Tac, εντός τριών ετών λειτουργούσαν σε καθεμία από τις ενενήντα μεγαλύτερες αγορές της Αμερικής (15).

Η δημοτικότητα της Cellular στις Ηνωμένες Πολιτείες ήταν απροσδόκητα ισχυρή. Η αύξηση κατά 100% κάθε χρόνο προσελκύει παραγωγούς απο το εξωτερικό. Η Ericsson διέθεσε διακόπτες και εξοπλισμό σταθμών βάσης, ενώ εταιρείες όπως η Nokia πωλούν συσκευές χειρός. Τα συστήματα AMPS πωλήθηκαν σε όλο τον κόσμο.

Αναλογικά κυψελοειδή αναπτύσσονταν επίσης στην Ευρώπη στα μέσα της δεκαετίας του '80. Το κύριο πρόβλημα ήταν ότι τα συστήματα λειτουργούσαν καλά από μόνα τους αλλά δεν συνεργάζονταν μεταξύ τους. Ένας Γερμανός πελάτης, για παράδειγμα, δεν μπορούσε να χρησιμοποιήσει το κινητό του στην Ιταλία. Ο σχεδιασμός στην Ευρώπη στις αρχές της δεκαετίας

του 1980 ήταν να δημιουργηθεί μια ενιαία ευρωπαϊκή ψηφιακή κινητή υπηρεσία με προηγμένα χαρακτηριστικά και εύκολη περιαγωγή.

Οι Ευρωπαίοι είδαν τα πράγματα διαφορετικά. Αποφάσισαν να δημιουργήσουν μια νέα τεχνολογία σε ένα νέο εύρος συχνοτήτων. Κινητό ραδιοτηλέφωνο αλλά πλήρως ψηφιακό που δε θα έχει συμβατότητα με τα υπάρχοντα συστήματα. Σχεδίαζαν το νέο πρότυπο ασύρματων επικοινωνιών τους μετά τις απαιτήσεις σταθερής τηλεφωνίας για ISDN, ελπίζοντας να κάνουν ένα ασύρματο σύστημα αντίστοιχο με αυτό. Η νέα υπηρεσία ονομάστηκε GSM.

Μέσα σε ένα σπάνιο τρίμηνο ευρωπαϊκής ενότητας, τα επιτεύγματα του GSM έγιναν “ένα απο τα πιο πειστικά επιχειρήματα για το τι μπορεί να επιτύχει η συνεργασία σε ολόκληρη την ευρωπαϊκή βιομηχανία στην παγκόσμια αγορά." Ο σχεδιασμός άρχισε σοβαρά και συνεχίστηκε για αρκετά χρόνια.

Στα τέλη της δεκαετίας του 1980 η αμερικανική βιομηχανία ασύρματων επικοινωνιών άρχισε να ψάχνει για ένα σύστημα υψηλότερης χωρητικότητας. Στα τέλη της δεκαετίας του 1980 η Ιαπωνία μελέτησε επίσης την επόμενη γενιά των κυψελοειδών συστημάτων. Τα συστήματα πρώτης γενιάς τους διαμορφώθηκαν μετά από AMPS, αλλά δεν ήταν σαφές εάν τα δεύτερα συστήματα θα ήταν αναλογικά ή ψηφιακά.

Τον Μάρτιο του 1990, το κυψελοειδές δίκτυο της Βόρειας Αμερικής υιοθέτησε το ψηφιακό πρότυπο: IS-54. Αυτή η επιλογή κέρδισε το Narrowband AMPS ή το NAMPS της Motorola, ένα πρόγραμμα ανάλυσης που αύξησε την χωρητικότητα μειώνοντας το μέγεθος του καναλιού. Ένας χειριστής είχε μεγάλη ευελιξία με το IS-54. Θα μπορούσε να μετατρέψει οποιοδήποτε από τα αναλογικά κανάλια της φωνής του σε ψηφιακό. Οι πελάτες είχαν ψηφιακές υπηρεσίες όπου αυτές ήταν διαθέσιμες και αναλογικές όπου δεν ήταν. Οι υπάρχοντες πελάτες δεν παρέμειναν χωρίς τις υπηρεσίες. Απλά δεν μπορούσαν να έχουν πρόσβαση στα νέα χαρακτηριστικά του IS-54.

Τα εμπορικά δίκτυα GSM άρχισαν να λειτουργούν στα μέσα του 1991 στην Ευρώπη. Το όλο ψηφιακό GSM αύξησε την χωρητικότητα τρεις φορές πάνω από το αναλογικό. Κάθε κινητό πλέον περιέχει κρυπτογράφηση ή αποκτά πρόσβαση σε κρυπτογράφηση για την αποτροπή της παρακολούθησης, ελέγχου ταυτότητας για την αποφυγή απάτης, υπηρεσίες σύντομων μηνυμάτων ή SMS και κάρτας SIM για εύκολη προσθήκη λογαριασμών. Το GSM θα εγκατασταθεί σε όλο τον κόσμο και θα γίνει η πιο δημοφιλής κυψελωτή ραδιοτηλεφωνική υπηρεσία. Τον Φεβρουάριο του 2004 ανακοινώθηκε ότι το GSM είχε ένα δισεκατομμύριο πελάτες.

Τον Αυγούστου του 1996, η Nokia παρουσίασε τον Communicator, ένα κινητό τηλέφωνο GSM και φορητό υπολογιστή μαζί. Είχε ένα πληκτρολόγιο QWERTY και ενσωματωμένα προγράμματα επεξεργασίας κειμένου και ημερολογίου. Εκτός από την αποστολή και τη λήψη φαξ, μπορούσε να ελέγξει το ηλεκτρονικό ταχυδρομείο και να έχει πρόσβαση στο διαδίκτυο με περιορισμένο τρόπο. Ωστόσο, η αποτελεσματικότητά του ήταν περιορισμένη, δεδομένου ότι τα κυψελοειδή δίκτυα επελέγησαν για τη φωνή, όχι για τα δεδομένα.

Μέχρι τα μέσα της δεκαετίας του '90, η παροχή ποιοτικής ομιλίας ήταν το ζητούμενο και εξασφαλίστηκε με τα κυψελοειδή συστήματα. Η φωνή, με τις ρυθμίσεις, ήταν όσο καλή έπρεπε να είναι. Με την επίλυση του προβλήματος της ομιλίας, τα δεδομένα έγιναν το πρώτο ενδιαφέρον των σχεδιαστών τηλεπικοινωνιακών συστημάτων.

Η φωνή παρέμενε η βασική υπηρεσία για τη μεγάλη πλειοψηφία των κινητών τηλεφώνων, αλλά η ανάπτυξη καλύτερων και ταχύτερων δικτύων δεδομένων μέσω των κυψελοειδών συστημάτων έγινε η προτεραιότητα.

Για καλύτερες υπηρεσίες φωνής χρησιμοποιήθηκε η εναλλαγή κυκλωμάτων, όπως γινόταν και με το σταθερό τηλεφωνικό δίκτυο. Αλλά τα δεδομένα δε μεταδίδονται αποτελεσματικά με την εναλλαγή κυκλωμάτων. Κατά συνέπεια, χρειάστηκε μια θεμελιώδης αλλαγή, από την εναλλαγή κυκλωμάτων στην εναλλαγή πακέτων. Και το είδος της μεταγωγής πακέτων που χρειάστηκε ήταν προφανές από την αρχή.

Το διαδίκτυο έγινε εμπορικό στα μέσα της δεκαετίας του ’90. Η ανάπτυξη του διαδικτύου ήρθε σε σύγκρουση με την τηλεφωνία μεταξύ του 1995 και του 2000. Το διαδίκτυο λειτουργεί με το καλώς οριζόμενο πρωτόκολλο Internet (IP), ουσιαστικά μια τεχνική μεταγωγής πακέτων. Η σημερινή γενική υπηρεσία πακέτων ραδιοσυχνοτήτων (GPRS), η βελτίωσή της, η EDGE και τα ασύρματα δίκτυα μικρής εμβέλειας, όπως το Blue-tooth, χρησιμοποιούν όλα το IP. Όλα τα συστήματα 3G χρησιμοποιούν το IP καθώς όλοι μας κατευθυνόμαστε προς τον "παγκόσμιο κόσμο IP".

Μέχρι τα μέσα της δεκαετίας του 1990 το κινητό έγινε όσο το δυνατόν πιο μικρό. Το πληκτρολόγιο και η οθόνη περιόρισαν την περαιτέρω μείωση του μεγέθους. Τα κυκλώματα κινητής τηλεφωνίας άρχισαν να ενσωματώνονται σε φορητούς υπολογιστές και PDA και σε όργανα όπως το Blackberry, αναγκάζοντάς μας να ξανασκεφτούμε τι είναι ένα κινητό τηλέφωνο. Τα τηλέφωνα εξελίσσονται για να παρέχουν μια ποικιλία υπηρεσιών, κυρίως μη φωνητικών, όπως ήχους κλήσης, λήψη εικόνων, μηνύματα κειμένου, παιχνίδια κ.ο.κ. Ενώ οι υπηρεσίες κινητής

τηλεφωνίας φαίνονται περιορισμένες μόνο από τη φαντασία, τα συστήματα που περνούν καθίστανται λιγότερα.
Τα συστήματα GSM και CDMA θα συνεχίσουν να εγκαθίστανται σε όλο τον κόσμο, αλλά μέχρι το 2005 δεν θα προκύψει νέο ραδιοτηλεφωνικό σύστημα.

2.1.2. Η περίοδος των Smartphones

Το 1999 το πρώτο smartphone (Ericsson R380) παρουσιάστηκε επίσημα. Οι χρήστες του R380 δεν χρειάζονταν να μεταφέρουν πολλές κινητές συσκευές, δεδομένου ότι όλες αυτές οι συσκευές είχαν εγκλωβιστεί σε αυτό το μικρό τηλέφωνο. Συνδεδεμένο σε περισσότερες από 120 χώρες διεθνώς σε 5 ηπείρους μέσω υπηρεσιών WAP παρέχονταν πρόσβαση στο Διαδίκτυο. Με αυτό το κινητό τηλέφωνο, οι χρήστες θα μπορούσαν να επικοινωνούν και να εργάζονται οπουδήποτε, ανά πάσα στιγμή. Με την οθόνη αφής και τον γραφικό πλούτο, παρέχονταν οργανωτικές δυνατότητες και οι υπηρεσίες WAP στο R380 επιτρέπουν στους χρήστες να λαμβάνουν ή να στέλνουν μηνύματα ηλεκτρονικού ταχυδρομείου ή να επισκέπτονται τις ιστοσελίδες τους ή ακόμα και να λαμβάνουν πληροφορίες σχετικά με τις καιρικές συνθήκες και νέα. Το R380 χρησιμοποιεί το λειτουργικό σύστημα Symbian, ένα σύστημα σχεδιασμένο ειδικά για συσκευές ασύρματης επικοινωνίας. Οι διαφορές μεταξύ των πρώιμων Smartphones με αυτά του σήμερα είναι ότι τα πρώτα smartphones προορίζονταν κατά κύριο λόγο για εταιρικές συσκευές, ενώ ήταν υπερβολικά ακριβά για τους καταναλωτές [16]. Η εποχή των Smartphones χωρίζεται σε τρεις κύριες φάσεις. Κατά τη διάρκεια της πρώτης φάσης όλα τα smartphones στόχευαν τις εταιρείες και τα χαρακτηριστικά και οι λειτουργίες τους ήταν σύμφωνα με τις εταιρικές απαιτήσεις. Το Blackberry θεωρείται ως η επαναστατική συσκευή αυτής της εποχής, εισήγαγε πολλά χαρακτηριστικά, όπως τα email, Internet, φαξ, περιήγηση στο Web, κάμερα.[16] [17] [18]. Η δεύτερη φάση της εποχής των Smartphones ξεκίνησε με την έλευση του iPhone το 2007. Εκείνη την εποχή, η δημοτικότητα του Android αυξανόταν και άρχιζε να χτίζει γερά θεμέλια. Ωστόσο, με την εμφάνιση του Apple iPhone, όλα μοιάζουν να αλλάζουν για πάντα. Αυτή ήταν η πρώτη φορά που η βιομηχανία εισήγαγε τα smartphones στην αγορά [19]. Στο τέλος του 2007 η Google παρουσίασε το λειτουργικό της σύστημα Android με πρόθεση να προσεγγίσει την αγορά των Smartphones. Κατά τη διάρκεια αυτής της χρονικής περιόδου δόθηκε έμφαση στην εισαγωγή χαρακτηριστικών που απαιτεί ο μέσος καταναλωτής και παράλληλα στη διατήρηση του κόστους σε σχετικά χαμηλό

επίπεδο, με στόχο την προσέλκυση όλο και περισσότερων πελατών. Χαρακτηριστικά όπως το ηλεκτρονικό ταχυδρομείο, τα μέσα κοινωνικής δικτύωσης , η χρήση ήχου / βίντεο, η πρόσβαση στο διαδίκτυο, η συνομιλία μαζί με τα γενικά χαρακτηριστικά του τηλεφώνου ήταν μέρος του smartphone [19] [20] [21] [22]. Η τρίτη φάση των Smartphones ήταν η φάση που βελτιώθηκε η ποιότητα της οθόνης, η τεχνολογία του smartphone, το λειτουργικό σύστημα, οι μπαταρίες, το περιβάλλον χρήστη και όλες γενικά οι λειτουργίες των έξυπνων αυτών συσκευών. Αυτή η φάση άρχισε το 2008 με τις αναβαθμίσεις στο λειτουργικό σύστημα των κινητών τηλεφώνων και μέσα στην τελευταία πενταετία υπήρξαν αρκετές αναβαθμίσεις στα λειτουργικά συστήματα Apple iOS και Android. Τα πιο δημοφιλή λειτουργικά συστήματα κινητής τηλεφωνίας (iOS, Android, Blackberry OS, Windows Mobile) και οι βασικές εταιρείες πώλησης Smartphones (Apple, Samsung, HTC, Motorola, Nokia, LG, Sony κ.λπ.) επικεντρώνονται σε λειτουργίες που θα συναρπάσουν τόσο των επιχειρηματικό κόσμο όσο και τους καταναλωτές. Ο ρόλος του Android ήταν τεράστιος κατά τη διάρκεια αυτής της χρονικής περιόδου, δεδομένου ότι πρόσφερε μια μεγάλη ευκαιρία σε όλους τους προμηθευτές να κατασκευάσουν συσκευές χρησιμοποιώντας τη μεγάλη τεχνολογία ανοικτού κώδικα Android [19] [20] [21].

2.2. Τα κινητά τηλέφωνα iPhone

2.2.1. Η ιστορία του iPhone

Από το 2007, όταν ο Steve Jobs παρουσίασε το the iPhone, η Apple κατάφερε να πουλήσει πάνω από 1.5 δισεκατομμύρια συσκευές χαρακτηρίζοντας έτσι το iPhone όχι μόνο το τηλέφωνο με τις μεγαλύτερες πωλήσεις αλλά και το πιο ιδανικό τηλέφωνο για τον κάθε άνθρωπο. Με αυτό τον τρόπο η Apple δημιούργησε έναν επιχειρηματικό κολοσσό για προγραμματιστές εφαρμογών αλλά και για εταιρίες κατασκευής αξουσουάρ. Εκατοντάδες άνθρωποι χρησιμοποιούν το iPhone τους αντί για υπολογιστή, κάμερα, GPS, ως μέσο για πληρωμές (αντί για πιστωτική/χρεωστική κάρτα) και φυσικά για μουσική.

Πριν από τα iPhone τα περισσότερα τηλέφωνα έτειναν να αντιγράψουν τα BlackBerry, όμως πλέον μετά την παρουσίαση ενός νέου iPhone όλη η αγορά γεμίζει με συσκευές που μοιάζουν σε αυτό, μεγάλη οθόνη, κομψός σχεδιασμός, βελτιωμένη κάμερα, ακόμα και το notch είναι χαρακτηριστικά που εμφανίστηκαν στην αγορά ακριβώς μετά από κάθε νέο iPhone.

Χάρη σε αυτό το τηλέφωνο η Apple κατάφερε να μπει στην καθημερινότητα του κόσμου και να βρει λύσεις για να κάνει τον κόσμο καλύτερο. Βέβαια πολλοί από εμάς ανησυχούμε που περνάμε αρκετές ώρες με το κινητό μας τηλέφωνο και για να κάνουμε οτιδήποτε το έχουμε πάντα στο χέρι μας. Όμως αναμφισβήτητα τα iPhone έχουν αλλάξει την ζωή μας, και πάντα με μια τέτοια ριζική αλλαγή λύνουμε υπάρχοντα προβλήματα και δημιουργούνται νέα.

Αρχίζοντας με το the original iPhone (το πρώτο iPhone) τον Ιανουάριο του 2007, στην σκήνη του συνεδρίου Macworld, έγινε η παρουσίαση του πρωτοποριακού τηλεφώνου εξηγώντας τις λειτουργίες του μια προς μια. Έξι μήνες μετά πραγματοποιήθηκε η εξαγωγή στην αγορά και πούλησε 270.000 iPhone το πρώτο σαββατοκύριακο που ήταν διαθέσιμο, έφτασε 1 εκατομμύριο στην Labor Day στην Αμερική.

Ένα χρόνο μετά η Apple δημοσίευσε το iPhone 3G με υποστήριξη των δικτύων 3G που προσέφερε μεγαλύτερη και γρηγορότερη πρόσβαση στο διαδίκτυο και στα email. Το τηλέφωνο επίσης είχε ενσωματωμένο το App Store που μπορούσε ο χρήστης του να αποκτήσει, δωρεάν ή επι πληρωμή εφαρμογές που θα του έλυναν τα χέρια μέσα από μεγάλη ποικιλία. Έτσι πολλοί προγραμματιστές είχα την δυνατότητα να δουλέψουν και να πουλήσουν τα λογισμικά τους στο ευρύ κοινό, με αποτέλεσμα να αλλάξει ο τρόπος που επικοινωνούμε, δουλεύουμε, ψυχαγωγούμαστε ακόμα και μετακινούμαστε.

Από τότε κάθε χρόνο έρχεται ένα νέο iPhone, με πιο μεγάλη οθόνη, πιο καλή κάμερα, πιο ωραίο σχεδιασμό που κάθε ένας θα ήθελε να έχει.

Το 2017, δέκα χρόνια μετά το 1ο iPhone, η Apple θέλησε να δώσει στην αγορά ένα μοναδικό και πολύ ιδιαίτερο μοντέλο που θα χάραζε την πορεία των επόμενων γενεών. Έτσι παρουσίασε το iPhone X, το πρώτο τηλέφωνο με οθόνη από άκρη σε άκρη, χωρίς home button, με εξαιρετική διπλή κάμερα, πανίσχυρο επεξεργαστή και φυσικά με Face ID μια τεχνολογία για ξεκλείδωμα του τηλεφώνου με την όψη του προσώπου του κάθε χρήστη, που επιτεύχθηκε τοποθετώντας την μπροστινή κάμερα ενσωματωμένη στο notch μαζί με άλλους αισθητήρες. Παρά την ανεβασμένη τιμή και τον περιορισμένο χρόνο παραγωγής του από την εταιρία το iPhone X πουλήθηκε πάνω από 16 δισεκατομμύρια άτομα παγκοσμίως μέχρι την άνοιξη του 2017. Τον Σεπτέμβρη του 2018 η Apple κυκλοφόρησε το νέο iPhone XS/ XS Max και XR με ακόμα περισσότερη επεξεργαστική ισχύ και λειτουργία κάμερας από τον προκάτοχο του, σε παρόμοιο όμως κομψό και ιδιαίτερο σχεδιασμό.

2.2.2 Η εξέλιξη των iPhone από το 2007 μέχρι σήμερα

29 Ιουνίου, 2007 - iPhone 2G (the original iPhone)

Με μοναδικό και πρωτοποριακό σχεδιασμό για την εποχή του το iPhone 2G κυκλοφόρησε σε 3 εκδόσεις ανάλογα με τη χωρητικότητα του σε 4/8/16GB.

11 Ιουλίου, 2008 - iPhone 3G

Το δεύτερο σε σειρά iPhone, αλλάζοντας ελάχιστα τον σχεδιασμό του και προσθέτοντας δυνατότητα σύνδεσης με 3G δίκτυα ,GPS. Αναβαθμισμένο iOS με ειδοποιήσεις στα email και δυνατότητα πλοήγησης βήμα-βήμα καθώς και σύνδεση με το App Store για απόκτηση εφαρμογών.

19 Ιουλίου, 2009 - iPhone 3GS

Ίδιος σχεδιασμός με τον προκάτοχό του όμως με γρηγορότερο επεξεργαστή, υψηλότερη ανάλυση κάμερας που έφτανε τα 480p για εγγραφή βίντεο και φωνητικές εντολές, το iPhone 3GS έκλεψε τις εντυπώσεις.

24 Ιουνίου 2010 - iPhone 4

Ήταν το πρώτο iPhone με αλλαγμένο σχεδιασμό, επίπεδη όψη και από τις 2 πλευρές, κάμερα στα 5 megapixel και μπροστινή κάμερα για δυνατότητα βιντεοκλήσης, υψηλής ανάλυσης οθόνη γνωστή ως Retina, σχεδιασμένο για ταυτόχρονη εκτέλεση πολλών ενεργειών με νέο επεξεργαστή γενιάς A4. Χαρακτηρίστηκε ως το καλύτερο iPhone για την εποχή καθώς και το πιο λεπτό τηλέφωνο στον πλανήτη.

14 Οκτώβρη 2011 - iPhone 4s

Διατηρώντας τον ίδιο σχεδιασμό, με κάμερα αναβαθμισμένη στα 8megapixel, βιντεοσκόπηση στα 1080p, νέο επεξεργαστή διπύρηνο A5 και ενσωματωμένο το Siri καθώς και πολλές νέες λειτουργίες όπως iCloud, iMessage, Notification Center κ.α.

21 Σεπτέμβρη 2012 - iPhone 5

Το iPhone 5 αμέσως μετά την κυκλοφορία του έκανε ρεκόρ σε πωλήσεις. Με οθόνη μεγαλύτερη των 4", νέο φορτιστή, διπύρηνο επεξεργαστή γενιάς A6, αλουμινένιο φινίρισμα και υποστήριξη LTE δικτύων.

20 Σεπτέμβρη 2013 - iPhone 5S & iPhone 5C

Για πρώτη φορά η Apple κάνει δημοσίευση 2 διαφορετικών μοντέλων. Το iPhone 5C ήταν μια αναβάθμιση του iPhone 5 σε 5 διαφορετικά χρώματα και νέο σχεδιασμό. Το iPhone 5S έρχεται με τις διπλάσιες επιδόσεις από το 5 με επεξεργαστή A7 - 64bit, M7 Motion Control coprocessor και δακτυλικό αποτύπωμα στο home button για γρήγορο και ασφαλή ξεκλείδωμα από τον χρήστη. Διπλό LED φλας στην κάμερα και φυσικά το νέο λογισμικό iOS 7.

19 Σεπτέμβρη 2014 - iPhone 6 & iPhone 6 Plus

Αυτή την χρονιά η Apple έκανε πάλι την διαφορά όχι μόνο για τα 2 διαφορετικά μοντέλα αλλά για την διαφορά τους στην οθόνη, 4,7" & 5,5" αντίστοιχα με νέα τεχνολογία οθόνης για ακόμα καλύτερη οπτική γωνία. Με ενσωματωμένο το νέο επεξεργαστή A8 64bit, νέα M8 motion coprocessor και βελτιωμένο δακτυλικό αποτύπωμα. Επιπλέον τα 2 νέα μοντέλα είχαν NFC λειτουργία για δυνατότητα πληρωμών μέσω apple pay.

25 Σεπτέμβρη 2015 - iPhone 6S $ iPhone 6S Plus

Ίδιος σχεδιασμός με τα μοντέλα του 2014 όμως όλα αλλάζουν στο εσωτερικό τους. Με νέο επεξεργαστή A9 64bit, νέο M9 motion coprocessor για εξαιρετικές επιδόσεις και με νέο χαρακτηριστικό το 3D Touch που αντιλαμβάνεται την πίεση στην οθόνη και είναι σε θέση να εκτελέσει διάφορες λειτουργίες όπως για παράδειγμα προβολή για ανάγνωσή ενός email χωρίς να το ανοίξουμε, και τέλος αναβάθμισή κάμερας.

31 Μαρτίου 2016 - iPhone SE

Ένα μοντέλο ειδικής έκδοσης για αυτούς που προτιμάνε το μέγεθος του 5/5S με δυνατότητες ενός 6S iPhone.

16 Σεπτεμβρίου 2016 - iPhone 7 & iPhone 7 Plus

Ένα από τα καλύτερα μοντέλα iPhone, με την καλύτερη διπλή κάμερα της εποχής, η Apple ανέβασε το επίπεδο για τους ανταγωνιστές της, νέος επεξεργαστής A10 quad-core chip for maximum processing δύναμη. Χωρίς όμως θέση για τα "αναλογικά" ακουστικά με 3.5mm θύρα.

22 Σεπτεμβρίου 2017 - iPhone 8 & iPhone 8 Plus

Με τον ίδιο σχεδιασμό, λίγο καλύτερες επιδόσεις σε κάμερα κ επεξεργαστή και ακόμα χωρίς 3.5mm θύρα για ακουστικά η Apple πάλι κάνει την διαφορά φέρνοντας την ασύρματη φόρτηση στο νέο iPhone.

3 Νοεμβρίου 2017 - iPhone X

Με OLED οθόνες που καλύπτουν (σχεδόν) όλο το μπροστά μέρος του τηλεφώνου, η Apple χαράζει νέα πορεία με το iPhone X, ένα συλλεκτικό θα έλεγε κάποιος τηλέφωνο για τα 10 χρόνια iPhone και πολλές νέες δυνατότητες όπως Face ID.

21 Σεπτεμβρίου 2018 - iPhone XS & iPhone XS Max

Με παρόμοιο σχεδιασμό όμως για ακόμη μια φορά πολύ διαφορετικό εσωτερικά το νέο iPhone με επεξεργαστή A12 Bionic 7nm για 6 πυρήνες που δίνουν απίστευτή εμπειρία χρήσης ακόμα και στις πιο απαιτητικές λειτουργίες.

26 Οκτώβρη 2018 - iPhone XR

Το τελευταίο μέχρι τώρα τηλέφωνο της Apple με liquid οθόνες σε μεγαλύτερο μέγεθος 6,1’’, ίδια χαρακτηριστικά με τα iPhone XS & iPhone XS Max, ισχυρίζεται πως είναι το πιο ακριβές τηλέφωνο σε χρώματα στην αγορά [23].

2.3. Το εργαλείο προγραμματισμού Xcode

Το Xcode είναι το IDE (integrated development environment) που παρέχει η Apple σε προγραμματιστές που επιθυμούν να δημιουργήσουν λογισμικό για τις ακόλουθες συσκευές:
iOS, tvOS, watchOS (Αυτό προστέθηκε με την έκδοση 7.0), macOS.
Το Xcode κυκλοφόρησε αρχικά το 2003 και η πιο πρόσφατη έκδοση του πακέτου είναι η έκδοση 8. Είναι σημαντικό να θυμόμαστε ότι η έκδοση 8.0 είναι σε λειτουργία beta. Εάν γίνει κανείς εγγεγραμμένος προγραμματιστής OS, το Xcode είναι ένα δωρεάν προϊόν που μπορεί να κατεβάσει από το Apple Store.
Υπάρχει ωστόσο η επιλογή, να το κατεβάσει κανείς χειροκίνητα, μέσω της σελίδας προγραμματιστή της Apple. Ένας επίσημος προγραμματιστής μπορεί να έχει πρόσβαση σε εκδόσεις beta και επίσης σε προηγούμενες εκδόσεις. Αυτό είναι πολύτιμο εάν η συμβατότητα με τις προηγούμενες εκδόσεις είναι σημαντικό στοιχείο για τον προγραμματιστή όταν αναπτύσσει τις εφαρμογές του.
Το Xcode έχει σχεδιαστεί για να δώσει στους προγραμματιστές ένα ενιαίο παράθυρο στο οποίο θα δουλεύουν. Το πως θα μοιάζει αυτό το παράθυρο εξαρτάται από την εργασία που εκτελεί ο

χρήστης εκείνη τη στιγμή. Για παράδειγμα, εάν επιλέξει να επεξεργαστεί ένα αρχείο, η διεπαφή θα αλλάξει έτσι ώστε να είναι διαθέσιμος ο κατάλληλος επεξεργαστής για τον συγκεκριμένο τύπο αρχείου. Η διεπαφή είναι πολύ ευέλικτη. Επιτρέπει να είναι ανοιχτές πολλές καρτέλες ταυτόχρονα.

Το Xcode συνοδεύεται από ελεγκτή πηγαίου κώδικα. Αυτό το εργαλείο λειτουργεί σε πραγματικό χρόνο. Αυτό σημαίνει ότι ενώ ο προγραμματιστής πληκτρολογεί, το Xcode επισημαίνει τυχόν σφάλματα. Εκτός από αυτό, όταν είναι σε θέση, το IDE προτείνει στον προγραμματιστή πώς να διορθώσει τα σφάλματα που βρίσκει. Επίσης δαπανάται λιγότερος χρόνος στην πληκτρολόγηση, χάρη στη λειτουργία της αυτόματης συμπλήρωσης του Xcode. Ο έλεγχος σφαλμάτων είναι διαθέσιμος για C ++, Objective-C, Swift και C.

Αρκετός χρόνος μπορεί να εξοικονομηθεί επίσης αποφεύγοντας την επαναφορά των περιττών αποσπασμάτων του κώδικα. Το εργαλείο έρχεται με διάφορα αποσπάσματα κώδικα που χρησιμοποιούνται συχνά και πρότυπα που μπορούν να χρησιμοποιηθούν ώστε η διαδικασία της ανάπτυξης μιας εφαρμογής να γίνει ακόμα πιο γρήγορα. Ο προγραμματιστής μπορεί επίσης να αποθηκεύσει τα δικά του αποσπάσματα κώδικα και πρότυπα.

Ο επεξεργαστής του Xcode επιτρέπει να μπορεί να δει κανείς ταυτόχρονα περισσότερα από ένα αρχεία. Αυτό, σε συνδυασμό με τη δυνατότητα εύρεσης και αντικατάστασης, δίνει στους προγραμματιστές τη δυνατότητα να κάνουν γενικές αλλαγές σε εκατοντάδες γραμμές κώδικα χωρίς να χρειάζεται να ανοίγουν ξεχωριστά αρχεία και να εκτελούν την ίδια ενέργεια ξανά και ξανά. Το αποτέλεσμα αυτού του γεγονότος είναι λιγότερες ώρες δουλειάς και ευκολία χρήσης.

Το Xcode συνοδεύεται από ένα γραφικό εργαλείο σχεδίασης διεπαφών που ονομάζεται Interface Builder. Μπορεί να το χρησιμοποιήσει ο προγραμματιστής για να σχεδιάσει μενού, να τοποθετεί τα παράθυρα μαζί και να σχεδιάσει διάφορα οπτικά στοιχεία. Μπορεί να τα πάρει από υπάρχοντα αντικείμενα σε μια ενσωματωμένη βιβλιοθήκη ή να αναπτύξει τα δικά του. Η διεπαφή του storyboard επιτρέπει στον προγραμματιστή να σχεδιάζει την εφαρμογή του έτσι ώστε να μπορεί να καθορίσει τον τρόπο με τον οποίο μία ενέργεια του χρήστη αναγκάζει την εφαρμογή να μεταβεί από μια λειτουργία σε μια άλλη. Αυτές οι μεταβάσεις μπορούν στη συνέχεια να σταλούν στον πηγαίο κώδικα.

Με το Auto Layout μπορεί κανείς να δημιουργήσει περιορισμούς για τα αντικείμενα UI, τα οποία στη συνέχεια θα πάρουν αυτόματα το σωστό μέγεθος και θέση στην οθόνη.

Το Xcode περιλαμβάνει ένα πλούσιο κατάλογο στοιχείων. Τα γραφικά, τα εικονίδια και άλλα οπτικά στοιχεία είναι κάποια από αυτά. Εδώ περιλαμβάνονται οι γνωστές εικόνες εκκίνησης που που βλέπει κανείς σε ήδη υπάρχουσες εφαρμογές σε συσκευές Apple.

Για να προσθέσει κανείς ένα στοιχείο 'oh wow!' στις εφαρμογές του, μπορεί να κάνει χρήση του εργαλείου επεξεργασίας Scene Kit. Το Scene Kit επιτρέπει την προσθήκη τρισδιάστατων στοιχείων σε μια εφαρμογή. Με το Particle Emmiter μπορεί κανείς να προσθέσει καπνό, αστέρια και άλλα κινούμενα σχέδια στην εφαρμογή του.

Μια εφαρμογή που δεν έχει δοκιμαστεί επαρκώς πριν βγει στο κοινό, είναι βέβαιο ότι θα "βυθιστεί" πριν αποκτήσει οποιαδήποτε έλξη. Ευτυχώς, το Xcode έρχεται με ένα ολοκληρωμένο βοηθητικό πρόγραμμα εντοπισμού σφαλμάτων. Αυτό επιτρέπει στον προγραμματιστή να εκτελεί την εφαρμογή του σε πραγματικό χρόνο, ενώ ταυτόχρονα εντοπίζει τα σφάλματα που υπάρχουν. Καθώς εκτελείται η εφαρμογή, μπορεί να ακολουθήσει τον κώδικα κατά γραμμή και ακόμα να ελέγξει την τιμή συγκεκριμένων μεταβλητών.

Δεδομένου ότι η απόδοση είναι τόσο σημαντική για τους χρήστες κινητών συσκευών, το βοηθητικό πρόγραμμα εντοπισμού σφαλμάτων παρέχει ενημέρωση για το πόση CPU καταναλώνει η εφαρμογή και πόσα μέσα χρησιμοποιεί από τη συσκευή σε σύγκριση με άλλες λειτουργίες που εκτελούνται εκείνη τη στιγμή.

Μόλις η εφαρμογή περάσει τη δοκιμή εντοπισμού σφαλμάτων, μπορεί να αρχίσει η επόμενη φάση των δοκιμών. Με το Xcode, μπορεί κανείς να γράψει απλά τις δοκιμές που θέλει να εκτελέσει και στη συνέχεια, να εκτελέσει αυτές τις δοκιμές χρησιμοποιώντας το Test Navigator.

Η λειτουργία αυτόματης αποθήκευσης του Xcode σημαίνει ότι υπάρχει μικρός κίνδυνος να χαθούν οι αλλαγές που έγιναν στο προτζεκτ ή στα αρχεία προέλευσης. Δεν χρειάζεται κάποια διαδικασία ρύθμισης για να ενεργοποιηθεί αυτή τη λειτουργία. Αυτό συμβαίνει αυτόματα και υπάρχει επιλογή αναίρεσης και επαναφοράς της λειτουργίας.

Το Xcode συνοδεύεται από μια πολύ λεπτομερή ανάλυση(documentation) και μια βοηθητική βιβλιοθήκη. Αυτό περιλαμβάνει τεκμηρίωση και πληροφορίες SDK, οδηγούς κωδικοποίησης, αναφορές API και δείγματα κώδικα. Το τεχνικό προσωπικό της Apple έχει επίσης δημιουργήσει χρήσιμα βίντεο για την εκπαίδευση υποψήφιων προγραμματιστών. Τέλος, σχετικά με τις ενημερώσεις, όταν είναι διαθέσιμη μια νέα ενημέρωση, το πρόγραμμα θα ενημερωθεί αυτόματα.

Το Xcode IDE διευκολύνει αυτό που διαφορετικά θα ήταν ένα πολύ περίπλοκο έργο. Αυτό το επιτυγχάνει κάνοντας πολύ πιο εύκολα τα βήματα ώστε μια εφαρμογή να μπορεί να είναι

διαθέσιμη στο κατάστημα εφαρμογών (App Store). Ο προγραμματιστής μπορεί να χρησιμοποιήσει το member center web tool για να παρακολουθεί τις συμβάσεις, τις πωλήσεις και πολλά άλλα. Οι ρυθμίσεις παραμέτρων της εφαρμογής, δίνουν ακόμα και τη δυνατότητα να δοκιμαστεί η εφαρμογή από Beta testers. Τέλος, με χρήση της σύνδεσης του στο iTunes μπορεί οποιοσδήποτε να κάνει λήψη της εφαρμογής.

Κεφάλαιο 3

3.1. Η γλώσσα προγραμματισμού Swift και η ιστορία της

3.1.1. Εισαγωγή

Η Swift είναι ένας εξαιρετικός τρόπος για να δημιουργεί κανείς λογισμικό, είτε πρόκειται για τηλέφωνα, υπολογιστές, διακομιστές ή για οτιδήποτε άλλο τρέχει κώδικα. Είναι μια ασφαλής, γρήγορη και δια δραστική γλώσσα προγραμματισμού που συνδυάζει τα καλύτερα στοιχεία των σύγχρονων γλωσσών προγραμματισμού με τη σοφία από την ευρύτερη τεχνολογική κουλτούρα της Apple και τις ποικίλες συνεισφορές της κοινότητας ανοιχτού κώδικα. Ο μεταγλωττιστής έχει βελτιστοποιηθεί για να επιτυγχάνει καλύτερη απόδοση και η γλώσσα βελτιστοποιήθηκε για την πιο αποτελεσματική ανάπτυξη κώδικα χωρίς συμβιβασμούς σε κανέναν από τους δύο τομείς.
Η Swift είναι φιλική προς τους νέους προγραμματιστές. Το γράψιμο του κώδικα σε Swift επιτρέπει στον προγραμματιστή να πειραματιστεί με τον κώδικα και να δει τα αποτελέσματα αμέσως, χωρίς να απαιτούνται τα έξοδα για την κατασκευή και τη λειτουργία μιας εφαρμογής.
Η Swift βοηθά στην αποφυγή κοινών σφαλμάτων υιοθετώντας σύγχρονα πρότυπα προγραμματισμού:

- Οι μεταβλητές αρχικοποιούνται πάντα πριν από τη χρήση.
- Οι δείκτες των πινάκων ελέγχονται για σφάλματα εκτός ορίων.
- Οι ακέραιοι αριθμοί ελέγχονται για υπερχείλιση.
- Οι nil values αντιμετωπίζονται με σαφήνεια.
- Η μνήμη διαχειρίζεται αυτόματα.
- Ο χειρισμός των σφαλμάτων επιτρέπει την ελεγχόμενη ανάκτηση από απροσδόκητες αποτυχίες.

Η Swift έχει σχεδιαστεί και βελτιστοποιηθεί για να αξιοποιήσει στο έπακρο το σύγχρονο hardware. Το συντακτικό και η βιβλιοθήκη (standard library) έχουν σχεδιαστεί με βάση την αρχή ότι σχετικά με τη γραφή κώδικα ο προφανής τρόπος είναι και ο καλύτερος. Ο συνδυασμός της

ασφάλειας και της ταχύτητας καθιστούν την Swift μια εξαιρετική επιλογή για όλα. Από το απλό "Hello, world!", σε ένα ολόκληρο λειτουργικό σύστημα.
Η Swift επιτρέπει την εκπόνηση σύνθετων ιδεών με σαφή και συνοπτικό τρόπο. Ως αποτέλεσμα, ο κώδικας δεν είναι απλώς πιο εύκολο να γραφεί, αλλά είναι και ευκολότερος στην ανάγνωση [24].

3.1.2. Η Ιστορία της Swift

Η ανάπτυξη της γλώσσας Swift ξεκίνησε τον Ιούλιο του 2010 από τον Chris Lattner, με την συνεργασία πολλών άλλων προγραμματιστών της εταιρίας Apple. Η γλώσσα πήρε ιδέες από άλλες γλώσσες όπως είναι Objective-C, Rust, Haskell, Ruby, Python, C#, CLU και πολλές άλλες. Τον Ιούνιο του 2014 η Apple Worldwide Developers Conference (WWDC) εφαρμογή έγινε το πρώτο δημοσιευμένο app γραμμένο σε γλώσσα Swift. Έπειτα από αυτό μια δοκιμαστική έκδοση της γλώσσας έγινε διαθέσιμη στους προγραμματιστές της Apple, όμως η εταιρία δεν μπόρεσε να υποσχεθεί ότι ο πηγαίος κώδικας θα ήταν συμβατός με την δοκιμαστική έκδοση, αν και σκόπευε να δημιουργήσει μετατροπείς διαθέσιμους αν χρειαζόταν για την τελική έκδοση.

Το εγχειρίδιο χρήσης με όνομα The Swift Programming Language, ένα βιβλίο 500 σελίδων, δημοσιεύτηκε στο WWDC και έγινε διαθέσιμο στο iBooks Store και στην επίσημη ιστοσελίδα.

Η πρώτη επίσημη έκδοση της Swift βγήκε στις 9 Σεπτέμβρη του 2014 με το Gold Master of Xcode 6.0 for iOS. Τον Οκτώβρη στις 22, 2014 έγινε δημοσίευση της Swift 1.1 μαζί με το Xcode 6.1. Πλέον τρέχει η έκδοση Swift 4.1 από τις 29 Μαρτίου, 2018.

Η γλώσσα Swift κέρδισε αμέσως την πρώτη θέση ως η πιο αγαπημένη γλώσσα προγραμματισμού στο Stack Overflow Developer Survey το 2015 και την δεύτερη θέση το 2016.

Τον Δεκέμβρη του 2015, Η εταιρία IBM ανακοίνωσε την ιστοσελίδα Swift Sandbox, με το οποίο επιτρέπεται στους προγραμματιστές να γράφουν Swift κώδικα σε ένα παράθυρο και εμφανίζεται σε ένα άλλο. Όμως αποδοκιμάστηκε τον Ιανουάριο του 2018.

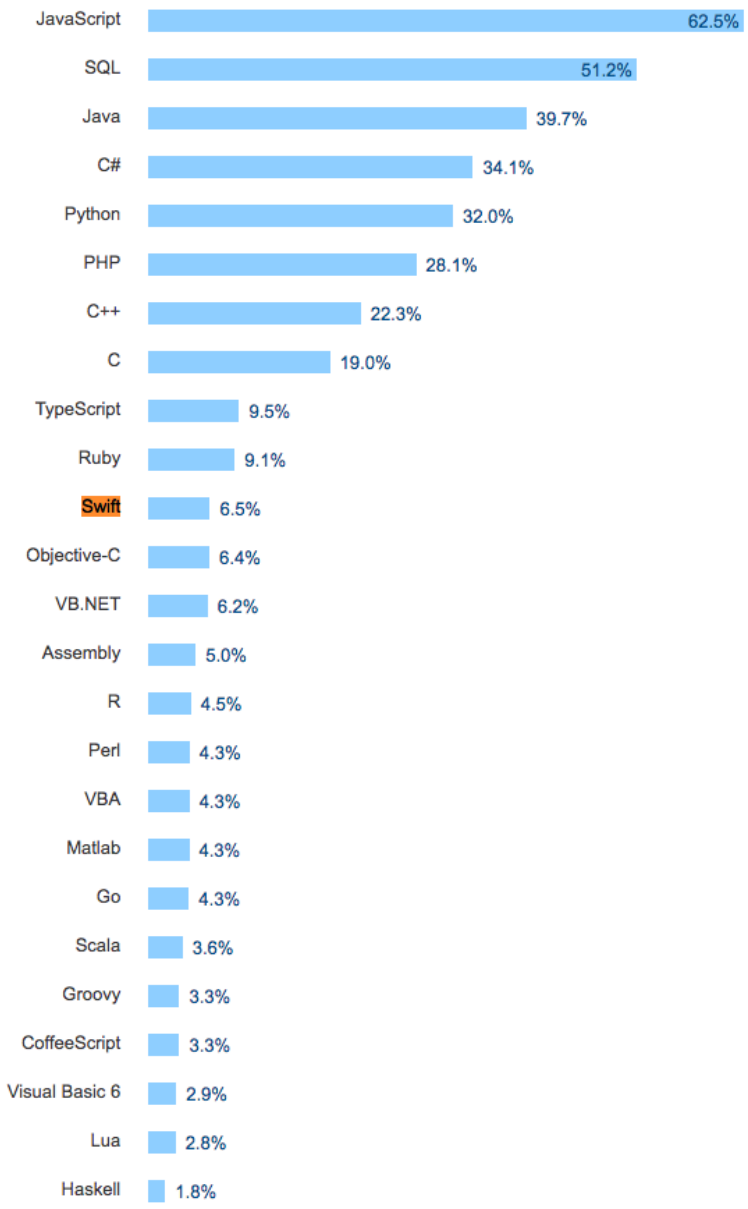
Το 2016 κατά την διάρκεια του WWDC, η Apple ανακοίνωσε μια εφαρμογή ειδικά για συσκευές iPad, με όνομα Swift Playgrounds, σχεδιασμένο για εκμάθηση της γλώσσας Swift. Η εφαρμογή αυτή παρουσιάζεται σε ένα παιχνίδι τριών διαστάσεων, το οποίο παρέχει σχόλια όταν γραμμές κώδικα τοποθετούνται σε συγκεκριμένη σειρά και εκτελούνται.

Πίνακας με τις εκδόσεις της Swift.

Swift 1.0	2014-09-09
Swift 1.1	**2014-10-22**
Swift 1.2	**2015-04-08**
Swift 2.0	**2015-09-21**
Swift 3.0	**2016-09-13**
Swift 4.0	**2017-09-19**
Swift 4.1	**2018-03-29**
Swift 4.2	**2018-09-17**

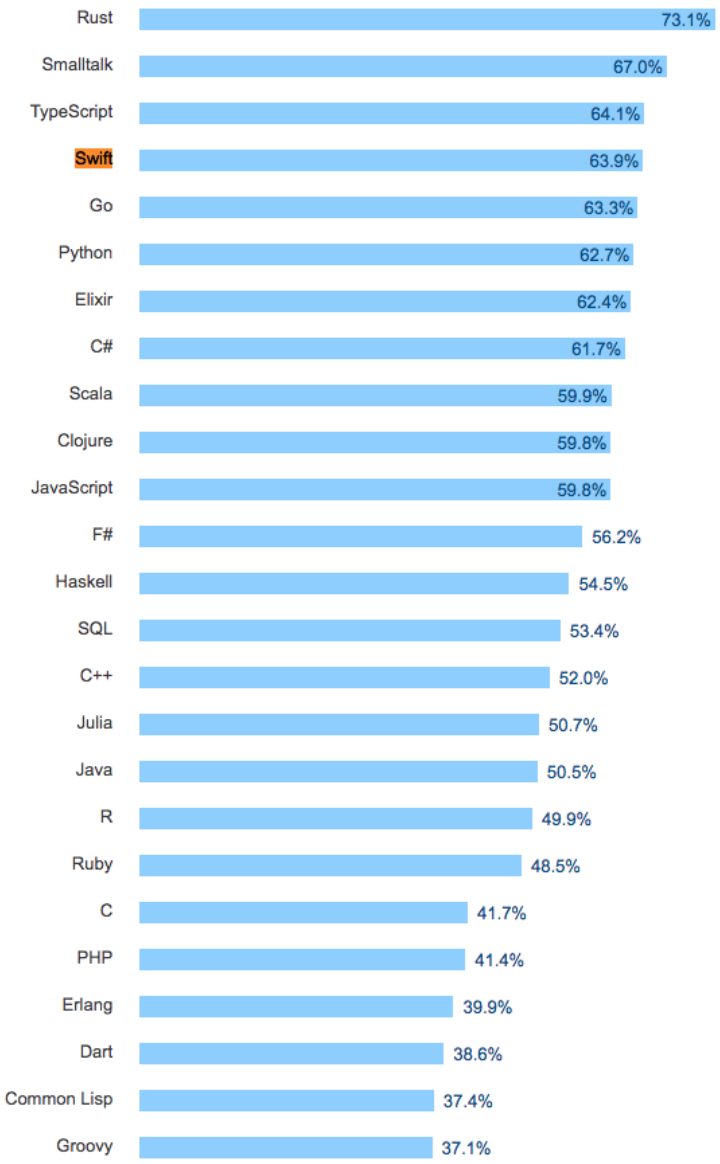
3.1.3. Στατιστικά στοιχεία

Σύμφωνα με το StackOverflow οι γλώσσες προγραμματισμού με τη μεγαλύτερη ζήτηση το 2017 φαίνονται στο διάγραμμα 1. Η έρευνα βασίστηκε στις ερωτήσεις που έγιναν κατά τη διάρκεια της χρονιάς στο StackOverflow.



Η Swift καταλαμβάνει την 11η θέση κάτι που είναι αρκετά σημαντικό αν αναλογιστεί κανείς ότι είναι μια γλώσσα που έχει χρόνο ζωής μια πενταετία.

Στο διάγραμμα 2 απεικονίζεται το ποσοστό των προγραμματιστών που επιλέγουν τη συγκεκριμένη γλώσσα και ενδιαφέρονται να συνεχίσουν να εξελίσσονται σε αυτή.



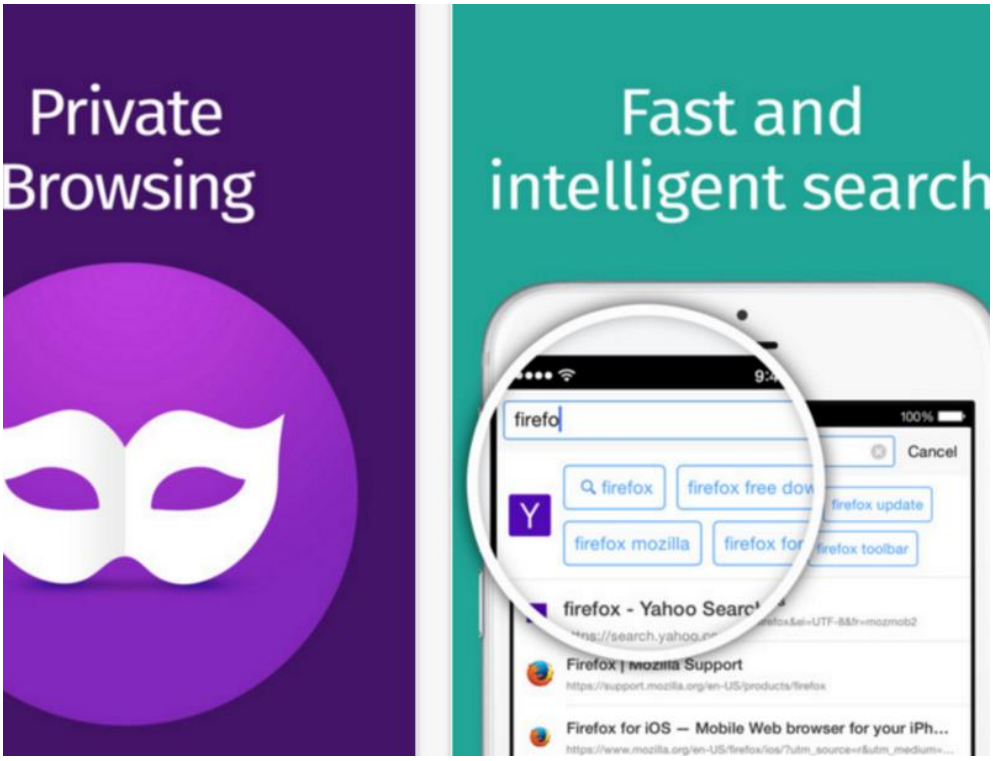
% of developers who are developing with the language or technology and have expressed interest in continuing to develop with it

Είναι ξεκάθαρο ότι η Swift είναι μια γλώσσα που αρέσει πολύ στους προγραμματιστές, οι οποίοι δηλώνουν ότι θέλουν να συνεχίσουν να προγραμματίζουν σε Swift. Καταλαμβάνει την 4η θέση ανάμεσα σε 25 γλώσσες προγραμματισμού.

3.2. Εφαρμογές ανοιχτού κώδικα γραμμένες σε Swift

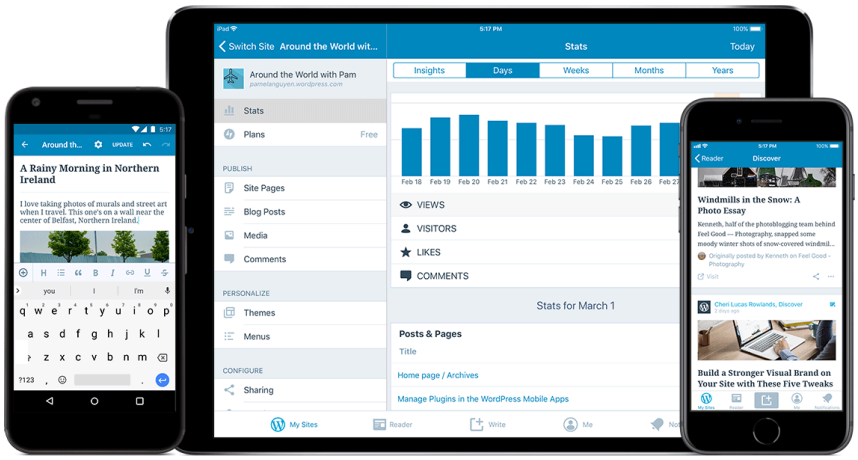
- Firefox iOS app (GitHub Stars: 5,906).

Το Firefox iOS app είναι μια εφαρμογή για συσκευές που τρέχουν iOS από την Mozilla. Είναι το πρώτο app από τον γνωστό φυλλομετρητή Firefox που δεν χρησιμοποιεί το λογισμικό Gecko της Mozilla λόγω περιορισμών ασφαλείας τος iOS από την Apple. Η εφαρμογή χρησιμοποιεί για την υλοποίηση της το WebKit της Apple αντί για το Gecko. Τέλος η εφαρμογή έχει υποστήριξη για συγχρονισμό των φυλλομετρητών Firefox μέσω του Firefox Sync [26].



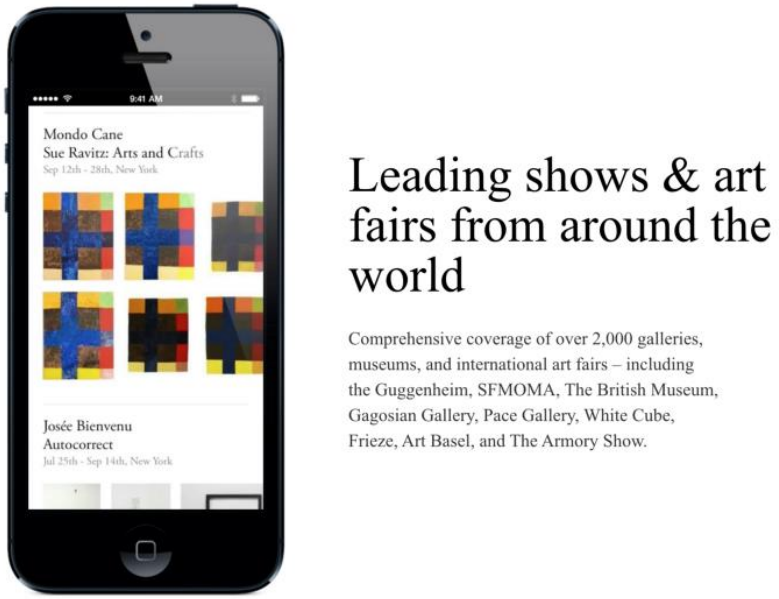
- WordPress for iOS (GitHub Stars: 1,225).

Το WordPress for iOS είναι μια εφαρμογή για διαχείριση και σχεδίαση ιστοσελίδων της γνωστής υπηρεσίας WordPress. Μέσα από το συγκεκριμένο app ο χρήστης έχει την δυνατότητα να δημοσιεύσει, να δημιουργήσει και να αγοράσει περιεχόμενο γράφοντας και επεξεργάζοντας τις ιστοσελίδες που διαθέτει μέσα από την υπηρεσία όπως ακριβώς και στα υπόλοιπα λογισμικά του WordPress [28].



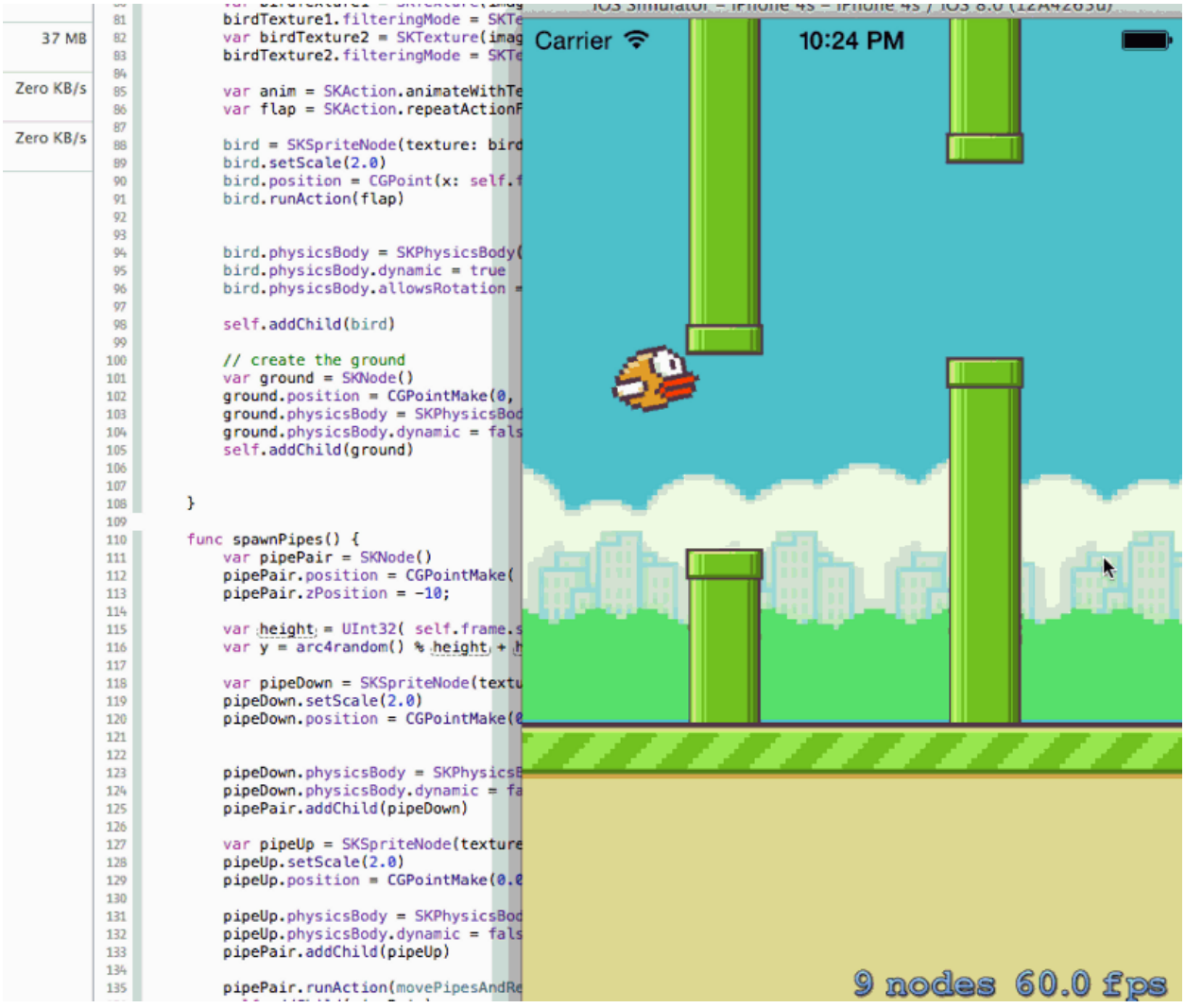
- Artsy: Auction App for Arts written in Swift (GitHub Stars: 1,302) Courtesy of Artsy

Το Artsy είναι μια εφαρμογή με την μεγαλύτερη βάση από μοντέρνα και ιστορικά έργα τέχνης από πάνω από 40.000 καλλιτέχνες, 2.000 γκαλερί και μουσεία που μπορεί κανείς να τα θαυμάσει αλλά και να τα αγοράσει μέσα από την εφαρμογή [27].



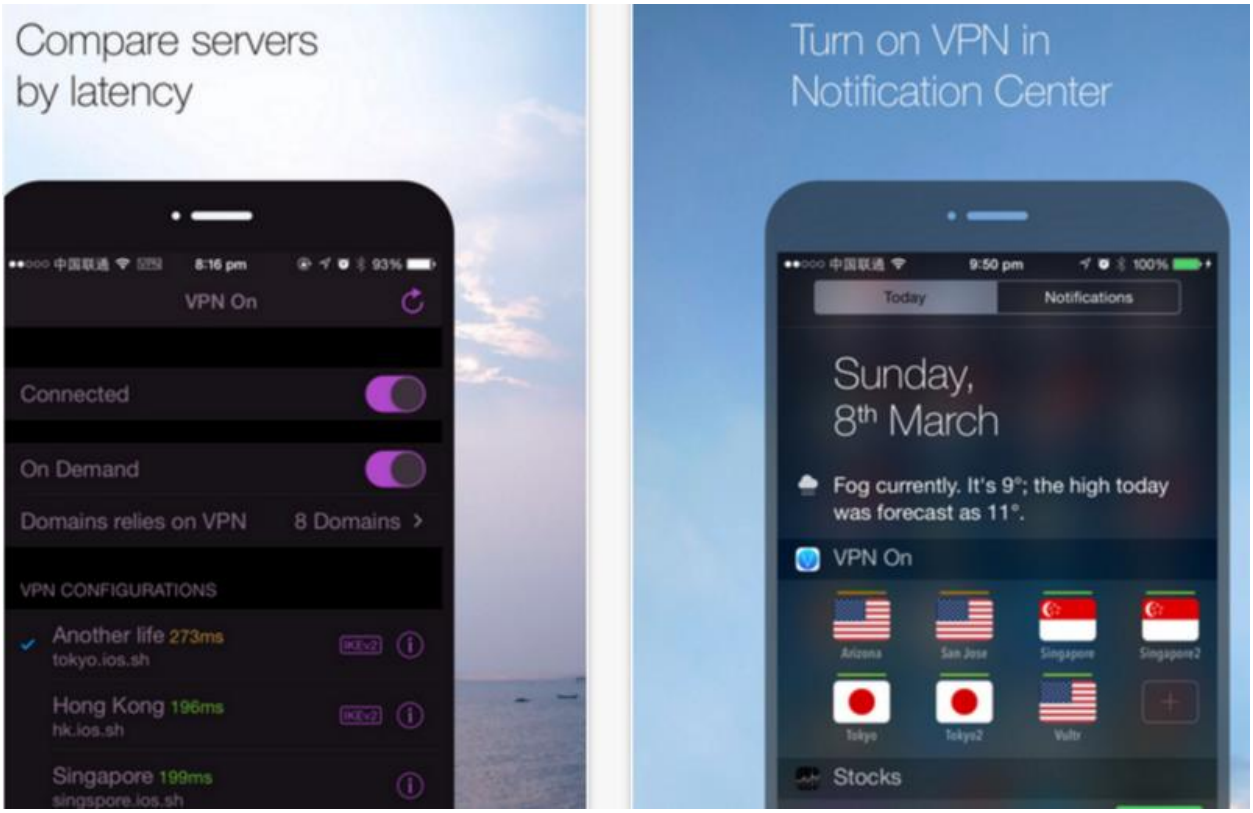
- Flappy iOS App in Swift (GitHub Stars: 7,854).

Το Flappy Bird είναι μια εφαρμογή παιχνιδιού σχεδιασμένη από τον Dong Nguyen από την εταιρία dotGears, στην οποία ο παίκτης προσπαθεί να κρατήσει τον έλεγχο ενός πτηνού και να καταφέρει να το περάσει μέσα κενά που δημιουργούν οι σωλήνες τους οποίους δεν πρέπει να ακουμπήσει γιατί χάνει. Το παιχνίδι δημιουργήθηκε μέσα σε αρκετές ημέρες και το πρωταγωνιστικό πτηνό χρησιμοποιήθηκε από άλλη εφαρμογή που ακυρώθηκε το 2012. Αφαιρέθηκε όμως απο τον δημιουργό της τον Φεβρουάριο του 2014 [29].



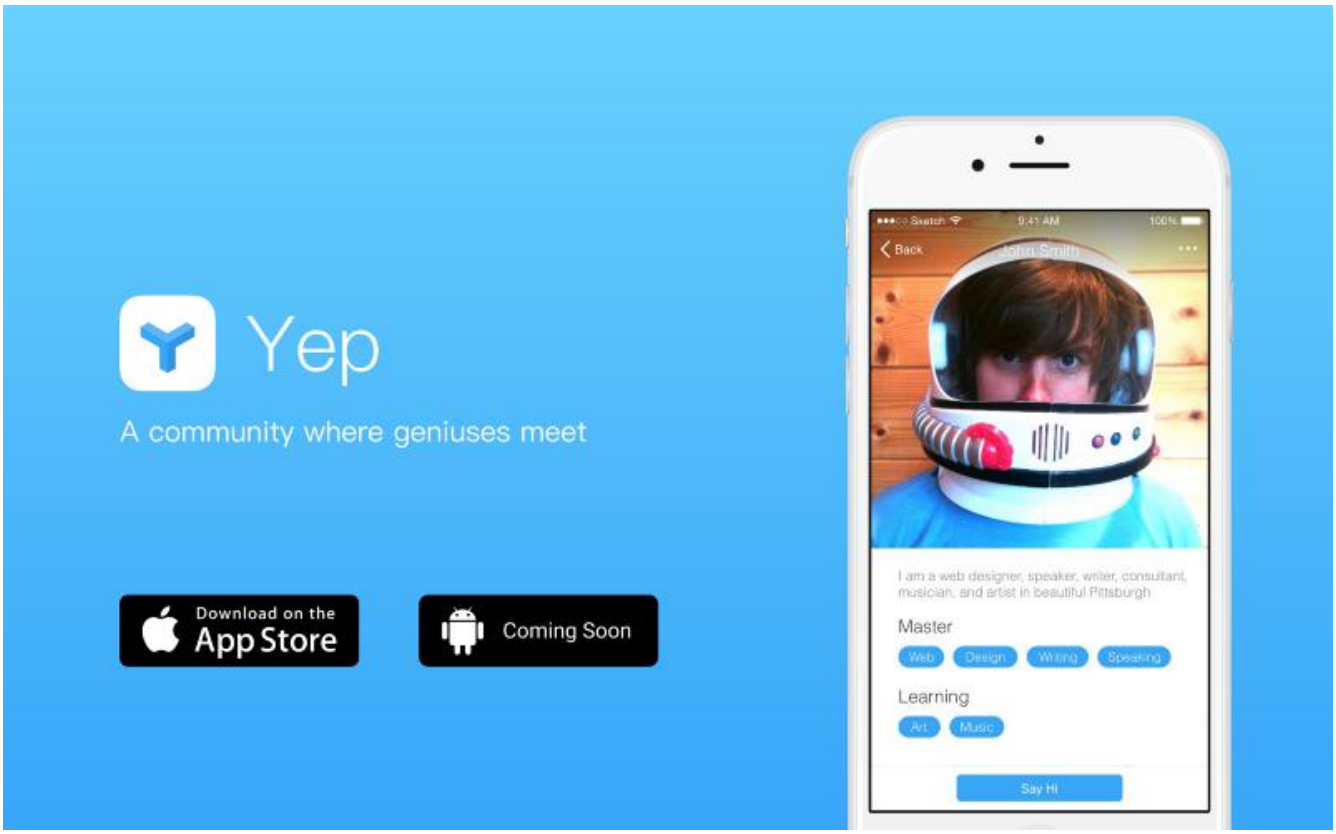
- Turn On your VPN like a hero. (GitHub Stars: 2,523).

Η εφαρμογή Turn On your VPN like a hero δίνει την δυνατότητα στον χρήστη να αποφύγει την διαδικασία του iOS για σύνδεση σε VPN εγκαθιστώντας ένα widget πάνω στο Notification Center της συσκευής που κάνει δυνατό να συνδέεται στο VPN σε 3 δευτερόλεπτα, καθώς και συνδέεται αυτόματα όταν επισκέπτεσαι κάποιο τομέα (domain) μέσα στην εφαρμογή [30].



- Yep: A community where geniuses meet (GitHub Stars: 4,547).

Η Yep είναι μια εφαρμογή κοινωνικής δικτύωσης, ευχάριστη όπως χαρακτηρίζεται που περιστρέφεται γύρω από το θέμα της “συνάντησης ιδιοφυΐας” επιτρέποντας στους χρήστες να βρουν ειδικούς εμπειρογνώμονες ή άλλους γνώστες ενός συγκεκριμένου τομέα. Η αρχιτεκτονική του προτζεκτ είναι εύκολη στην κατανόηση, ακολουθώντας το κοινό μοτίβο λογισμικού MVC, χωρίζοντας το UI, την επιχειρησιακή λογική και το μοντέλο δεδομένων [31].



3.3. Πλεονεκτήματα της Swift

- Το πρώτο και πολύ σημαντικό πλεονέκτημα στην επιλογή της Swift είναι αναμφισβήτητα το καθαρό συντακτικό της, γεγονός που καθιστά ευκολότερη την ανάγνωση και την εγγραφή. Ο αριθμός των γραμμών κώδικα που απαιτούνται για την εκτέλεση μιας λειτουργίας στη Swift είναι πολύ μικρότερος από ότι στην Objective-C. Ο λόγος είναι επειδή η Swift δε χρησιμοποιεί κάποια σύμβολα, όπως τα ερωτηματικά στις γραμμές λήξης ή τις παρενθέσεις που περιβάλλουν τις υπό όρους εκφράσεις μέσα σε if / else δηλώσεις. Μια άλλη σημαντική αλλαγή είναι ότι οι κλήσεις μεθόδων δεν γίνονται η μια μέσα στην άλλη με αποτέλεσμα μια πολύ μπερδεμένη έκφραση. Αντίθετα, οι κλήσεις μεθόδων και συναρτήσεων στη Swift χρησιμοποιούν μια λίστα παραμέτρων χωρισμένη με κόμματα εντός παρενθέσεων. Ως αποτέλεσμα, ο κώδικας είναι καθαρότερος με απλοποιημένη σύνταξη.

- Η Objective-C δεν είναι δυνατό να εξελιχθεί αν δεν εξελίσσεται πρώτα η C. Αντίθετα, η Swift δεν έχει αυτές τις εξαρτήσεις, γεγονός που καθιστά πολύ πιο εύκολο να διατηρηθεί. Η C απαιτεί από τους προγραμματιστές να διατηρούν δύο αρχεία κώδικα για να βελτιώσουν το χρόνο κατασκευής και την αποδοτικότητα του κώδικα. Η Swift, ωστόσο, δεν έχει αυτή την απαίτηση για τα δύο αρχεία, συνδυάζοντας την κεφαλίδα (.h) και τα αρχεία υλοποίησης (.m) σε ένα ενιαίο αρχείο κώδικα (.swift).
- Στην ανταγωνιστική αγορά των εφαρμογών για κινητά, η ανάπτυξη μιας ασφαλούς εφαρμογής αποτελεί προτεραιότητα. Το συντακτικό και οι γλωσσικές δομές της Swift αποκλείουν τους διάφορους τύπους σφαλμάτων που είναι πιθανά στην Objective-C . Αυτό σημαίνει ότι θα υπάρξουν λιγότερες συγκρούσεις και περιπτώσεις προβληματικής συμπεριφοράς. Δε σημαίνει ότι εμποδίζει τους προγραμματιστές να γράψουν κακό κώδικα, αλλά ότι καθιστά λιγότερο πιθανό το ενδεχόμενο λάθους. Με τη Swift μπορούν να διορθωθούν τα σφάλματα κατά τη σύνταξη του κώδικα, κάτι που δεν είναι εφικτό με την Objective-C. Ως αποτέλεσμα, η Swift λειτουργεί καλύτερα και πιο γρήγορα σε σύγκριση με την Objective-C όταν πρόκειται για δοκιμές σφαλμάτων και έτσι μπορεί να θεωρηθεί ως μια ασφαλής γλώσσα προγραμματισμού.
- Με την Objective-C, υπάρχουν πολλά προβλήματα που προκαλούν διακοπές λειτουργίας. Η Swift παρέχει κώδικα που είναι λιγότερο επιρρεπής σε σφάλματα, λόγω της inline υποστήριξης της στον χειρισμό των συμβολοσειρών κειμένου και των δεδομένων. Επιπλέον, οι κλάσεις δεν χωρίζονται σε δύο μέρη τη διεπαφή και την υλοποίηση. Αυτό μειώνει τον αριθμό των αρχείων στο μισό, γεγονός που καθιστά πολύ πιο εύκολη τη διχείρηση.
- Η Swift παρέχει επίσης το πλεονέκτημα της ταχύτητας κατά τη διάρκεια της ανάπτυξης μιας εφαρμογής, που με τη σειρά του, μειώνει το κόστος. Μια σύνθετη ταξινόμηση, για παράδειγμα, θα τρέξει 3,9 φορές ταχύτερα από την εφαρμογή του ίδιου αλγορίθμου στην Python. Η συγκεκριμένη επίδοση είναι καλύτερη και από την Objective-C, η οποία είναι 2,8 φορές ταχύτερη από την Python. Η απόδοση της πλησιάζει σε εκείνη της C ++, η οποία θεωρείται ως ο ταχύτερος αλγόριθμος υπολογιστικής αριθμητικής.
- Οι δυναμικές βιβλιοθήκες είναι εκτελέσιμα κομμάτια κώδικα που μπορούν να συνδεθούν με μια εφαρμογή. Αυτή η λειτουργία επιτρέπει στις τρέχουσες εφαρμογές με Swift να συνδέονται με νεότερες εκδόσεις της γλώσσας Swift καθώς εξελίσσονται με την πάροδο του χρόνου. Οι

δυναμικές βιβλιοθήκες στη Swift μεταφορτώνονται απευθείας στη μνήμη, μειώνοντας το αρχικό μέγεθος της εφαρμογής και αυξάνοντας τελικά την απόδοση της εφαρμογής.

- Τα Playgrounds είναι ένα χαρακτηριστικό που επιτρέπει στους προγραμματιστές να δοκιμάσουν έναν νέο αλγόριθμο χωρίς να χρειάζεται να δημιουργήσουν ολόκληρη την εφαρμογή. Η Apple έχει ενσωματώσει κώδικα εκτέλεσης στα Playgrounds για να βοηθήσει τους προγραμματιστές να δημιουργήσουν ένα κομμάτι κώδικα ή να γράψουν έναν αλγόριθμο, ενώ λαμβάνουν ανατροφοδότηση στην πορεία. Αυτός ο βρόχος ανατροφοδότησης μπορεί να βελτιώσει την ταχύτητα με την οποία μπορεί να γραφτεί ο κώδικας με τη βοήθεια της απεικόνισης δεδομένων. Τα Playgrounds και η Swift επιβεβαιώνουν τις προσπάθειες της Apple να καταστήσει την ανάπτυξη εφαρμογών ευκολότερη και πιο προσιτή.
- Η Swift το 2015 ανακοινώθηκε ότι γίνεται ανοικτού κώδικα, ανοίγοντας το δρόμο για να χρησιμοποιηθεί σε διάφορες πλατφόρμες και για backend δομές. Η Open-Source Swift σημαίνει ότι η Apple θα μπορέσει να πάρει ανατροφοδότηση από την κοινότητα ανοιχτού κώδικα για να κάνει βελτιώσεις καθώς οι ανεξάρτητοι προγραμματιστές συμβάλλουν στην επιτυχία της γλώσσας. Η Swift απογειώθηκε όχι μόνο επειδή είναι μια καλά δομημένη και σχεδιασμένη γλώσσα, αλλά και επειδή πολλοί προγραμματιστές τη στήριξαν [25].

Κεφάλαιο 4

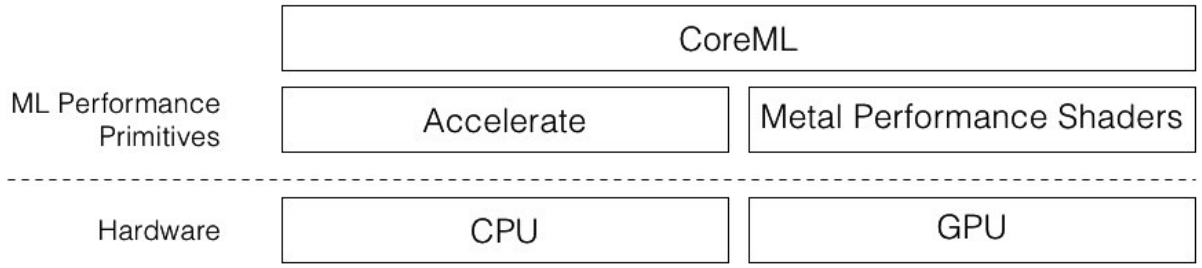
4.1. Το Core ML

4.1.1. Εισαγωγή στο Core ML

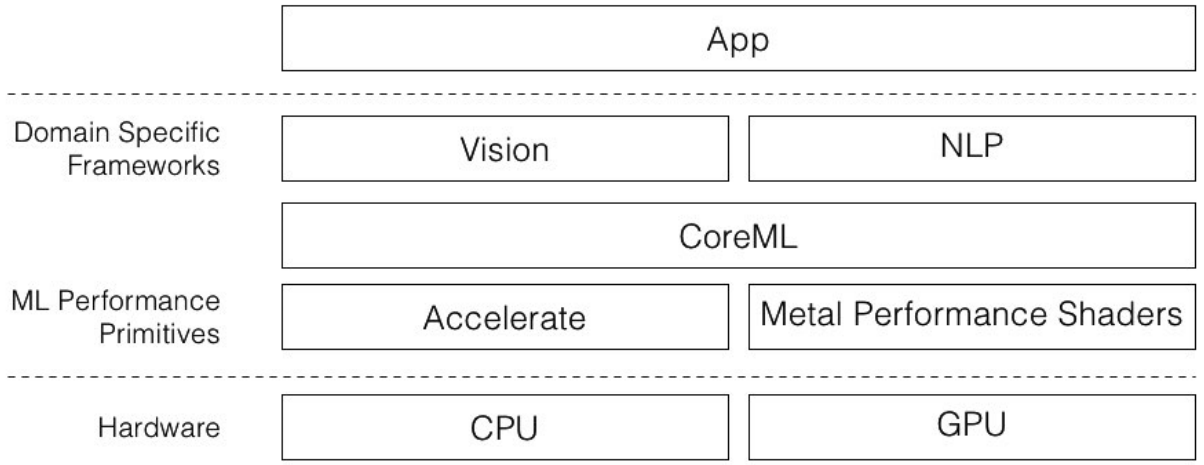
Το Core ML, το οποίο εισήχθη από την Apple, είναι ένα πλαίσιο μηχανικής μάθησης (machine learning) που δίνει τη δυνατότητα στους προγραμματιστές εφαρμογών iOS να ενσωματώσουν την τεχνολογία μηχανικής μάθησης στις εφαρμογές τους. Υποστηρίζει την επεξεργασία φυσικής γλώσσας (NLP), την ανάλυση εικόνας και διάφορα άλλα συμβατικά μοντέλα για να παρέχει κορυφαία απόδοση με ελάχιστη μνήμη και κατανάλωση ενέργειας.

Αυτό το άρθρο είναι ένα απόσπασμα που προέρχεται από το βιβλίο Machine Learning with Core ML που έγραψε ο Joshua Newnham. Σε αυτό το άρθρο, θα γνωρίσετε τα βασικά στοιχεία του CoreML και της τυπικής ροής εργασιών του.

Με την κυκλοφορία των iOS 11 και Core ML, η συμπερασματολογία γίνεται απλώς θέμα μερικών γραμμών κώδικα. Η εξαγωγή συμπερασμάτων ήταν δυνατή και πριν από το iOS 11, αλλά χρειαζόταν επιπλέον να πάρει κανείς ένα προ-εκπαιδευμένο μοντέλο και να το μεταφέρει χρησιμοποιώντας ένα υπάρχον πλαίσιο, όπως το Accelerate ή τα metal performance shaders (MPSes). Το Accelerate και τα MPSes εξακολουθούν να χρησιμοποιούνται κάτω από την ομπρέλα του Core ML, αλλά το Core ML αποφασίζει ποιο πλαίσιο θα πρέπει να χρησιμοποιεί το μοντέλο (Το Accelerate χρησιμοποιεί την CPU για εργασίες με μεγάλη μνήμη και τα MPSes χρησιμοποιούν τη GPU για υπολογιστικές εργασίες). Φροντίζει επίσης να αφαιρέσει τις πολλές λεπτομέρειες. Αυτό το στρώμα της διαδικασίας αφαίρεσης φαίνεται στο παρακάτω διάγραμμα:



Επίσης υπάρχουν κάποια επιπλέον στρώματα. Το iOS 11 έχει εισαγάγει και επεκτείνει στρώματα ειδικά για το συγκεκριμένο τομέα, τα οποία αφαιρούν περαιτέρω εργασίες σχετικές με την επεξεργασία δεδομένων εικόνας και κειμένου, όπως η ανίχνευση προσώπου, η παρακολούθηση αντικειμένων, η μετάφραση γλωσσών και αναγνώριση οντοτήτων (NER). Αυτά τα συγκεκριμένα πεδία είναι ενσωματωμένα στο Vision και στα πλαίσια επεξεργασίας φυσικής γλώσσας (NLP).



Αξίζει να σημειωθεί ότι αυτά τα στρώματα είναι σύνηθες να χρησιμοποιούνται μαζί, ειδικά εκείνα τα πλαίσια που προσφέρουν χρήσιμες μεθόδους προεπεξεργασίας και προετοιμασίας των δεδομένων πριν την αποστολή σε ένα μοντέλο Core ML.

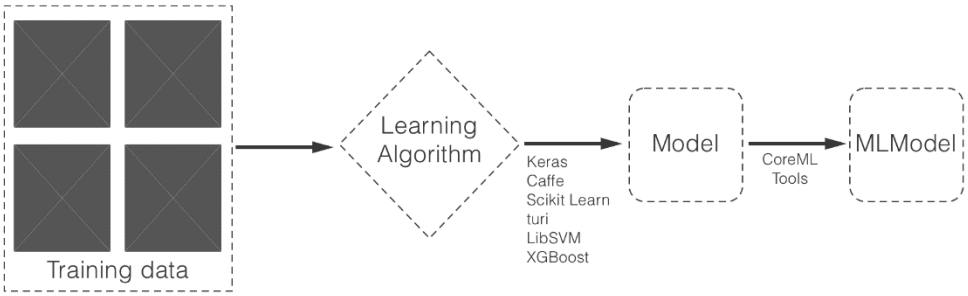
Τι ακριβώς όμως είναι το Core ML; Μπορεί κανείς να το φανταστεί ως μια σειρά από εργαλεία που χρησιμοποιούνται για τη διευκόλυνση της προσθήκης μοντέλων ML σε iOS, έτσι ώστε να υπάρχει πρόσβαση σε αυτά μέσω μιας διεπαφής και να μπορούν να χρησιμοποιηθούν στον κώδικα[33].

4.1.2. Τυπική ροή εργασίας με το Core ML

Όπως περιγράφηκε προηγουμένως, τα δύο κύρια καθήκοντα μιας τυπικής ροής εργασίας ML συνίστανται στην εκπαίδευση και τη συμπερασματολογία. Το στάδιο της εκπαίδευσης περιλαμβάνει την απόκτηση και την προετοιμασία των δεδομένων, τον καθορισμό του μοντέλου και στη συνέχεια την πραγματική εκπαίδευση. Τη στιγμή που το μοντέλο έχει επιτύχει

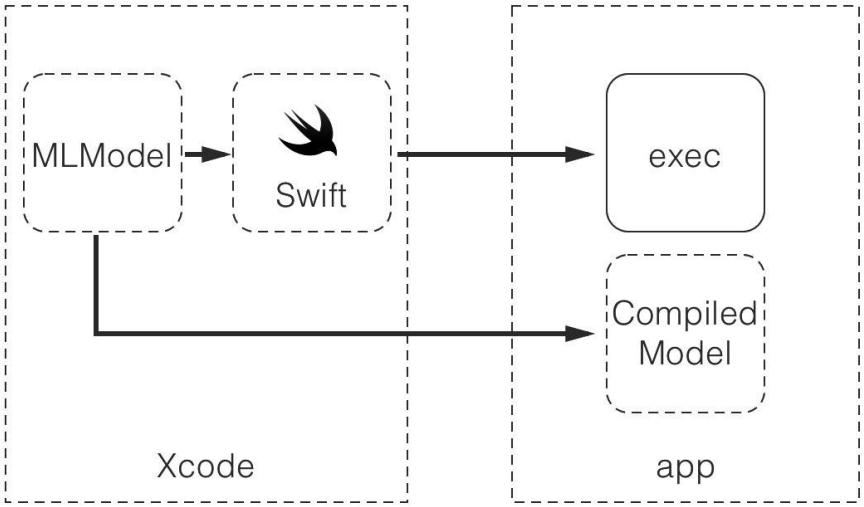
ικανοποιητικά αποτελέσματα κατά τη διάρκεια της εκπαίδευσης και είναι σε θέση να εκτελέσει επαρκείς προβλέψεις (συμπεριλαμβανομένων των δεδομένων που δεν έχει ξαναδεί), μπορεί να χρησιμοποιηθεί για συμπερασματολογία χρησιμοποιώντας δεδομένα διαφορετικά από εκείνα που χρησιμοποιήθηκαν για την εκπαίδευσή του. Το Core ML παρέχει εργαλεία που διευκολύνουν την είσοδο ενός εκπαιδευμένου μοντέλου στο iOS, ένα από τα οποία είναι το Python packaged (ονομάζεται Core ML Tools). Το εργαλείο εξάγει ένα αρχείο .mlmodel, το οποίο στη συνέχεια μπορεί να εισαχθεί στο Xcode προτζεκτ.

Μόλις εισαχθεί το μοντέλο, το Xcode θα δημιουργήσει μια διεπαφή για αυτό, καθιστώντας το εύκολα προσβάσιμο μέσω του κώδικα. Μια περίληψη της διαδικασίας δημιουργίας του μοντέλου παρουσιάζεται στο ακόλουθο διάγραμμα:



Creating the model (offline)

Το προηγούμενο διάγραμμα απεικονίζει τη διαδικασία δημιουργίας του .mlmodel, είτε χρησιμοποιώντας ένα υπάρχον μοντέλο από τα υποστηριζόμενα πλαίσια, είτε ένα μοντέλο που εκπαιδεύεται από το μηδέν. Τα εργαλεία core ML υποστηρίζουν τα περισσότερα από τα πλαίσια, συμπεριλαμβανομένων των Keras, Turi, Caffe, scikit-learn, LibSVN και XGBoost. Η Apple έχει επίσης μετατρέψει αυτό το πακέτο σε ανοικτού κώδικα για πιο εύκολη προσαρμογή τόσο σε άλλα πλαίσια, όσο και από τον ίδιο τον προγραμματιστή. Η διαδικασία εισαγωγής του μοντέλου απεικονίζεται στο διάγραμμα που ακολουθεί:



Model as code

Μόλις εισαχθεί το μοντέλο, με τη διαδικασία που αναφέρθηκε προηγουμένως, το Xcode δημιουργεί μια διεπαφή για το μοντέλο, τις εισόδους του μοντέλου και τις εξόδους [33].

4.1.3. Εκπαίδευση ενός μοντέλου ML για αναγνώριση εικόνων

Το αντικείμενο ταξινόμησης εικόνων (image classifier) είναι ένα μοντέλο ML που εκπαιδεύεται στο να αναγνωρίζει εικόνες. Όταν του δοθεί μια φωτογραφία, αυτό απαντάει με μια ετικέτα για την εικόνα αυτή.

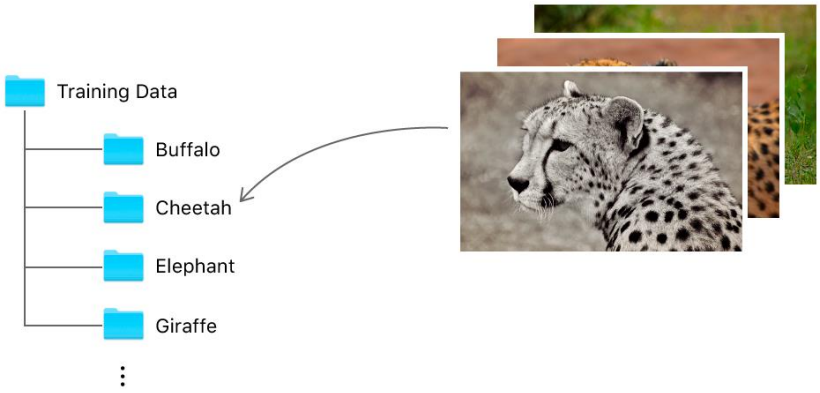
Image classifier

Ο ταξινομητής εικόνων εκπαιδεύεται με παραδείγματα εικόνων που αντιπροσωπεύουν την επιθυμητή ετικέτα. Για παράδειγμα, το μοντέλο μπορεί να εκπαιδευτεί να αναγνωρίζει άγρια ζώα, δίνοντας του μια μεγάλη ποικιλία από φωτογραφίες που απεικονίζουν τα ζώα αυτά.

4.1.3.1. Προετοιμασία των δεδομένων

Η διαδικασία αρχίζει προετοιμάζοντας τα δεδομένα που θα χρησιμοποιηθούν για την εκπαίδευση και την αξιολόγηση του ταξινομητή. Δημιουργείται ένα πακέτο δεδομένων για εκπαίδευση με περίπου το 80% των διαθέσιμων εικόνων για κάθε μια ετικέτα. Στη συνέχεια δημιουργείται ένα άλλο πακέτο για δοκιμή από τα υπόλοιπα δεδομένα. Καλό είναι να αποφεύγονται τα διπλότυπα στις εικόνες. Έπειτα τα δεδομένα οργανώνονται έτσι ώστε να έχουν συμβατή μορφή για το MLImageClassifier.DataSource. Ένας τρόπος για να γίνει αυτό είναι να δοθεί όνομα τους φακέλους: Training Data για το πακέτο εικόνων που είναι για την εκπαίδευση και Testing Data για εκείνα που θα χρησιμοποιηθούν για τη δοκιμή του μοντέλου. Κάθε φάκελος πρέπει να περιέχει υπό φακέλους με ετικέτες για το όνομα από κάθε είδος. Το ακριβές όνομα που θα δοθεί δεν έχει σημασία, αρκεί βέβαια να βγαίνει νόημα. Όσο για το όνομα της κάθε εικόνας , δεν παίζει κάποιο ρόλο στην διαδικασία. Ο ελάχιστος αριθμός που μπορεί να δοθεί είναι 10 φωτογραφίες για κάθε πακέτο δεδομένων, βέβαια όσες περισσότερες φωτογραφίες δοθούν τόσο καλύτερο θα είναι το μοντέλο. Επίσης τα πακέτα δεδομένων χρειάζεται να έχουν μια ισορροπία στον όγκο δεδομένων. Αυτό σημαίνει ότι πρέπει να αποφεύγεται για παράδειγμα μια κατάσταση κατά την οποία 10 φωτογραφίες δίνονται για το ένα πακέτο και 1000 για το άλλο. Οι φωτογραφίες μπορούν να έχουν

κάποια από τις γνωστές μορφές όπως είναι το JPEG και PNG και πρέπει να είναι το λιγότερο 299x299 pixels. Δεν χρειάζεται να είναι στο ίδιο μέγεθος, για την ακρίβεια είναι καλύτερο να είναι σε διάφορα μεγέθη και ανάλυση, φως και οπτικές γωνίες [32].



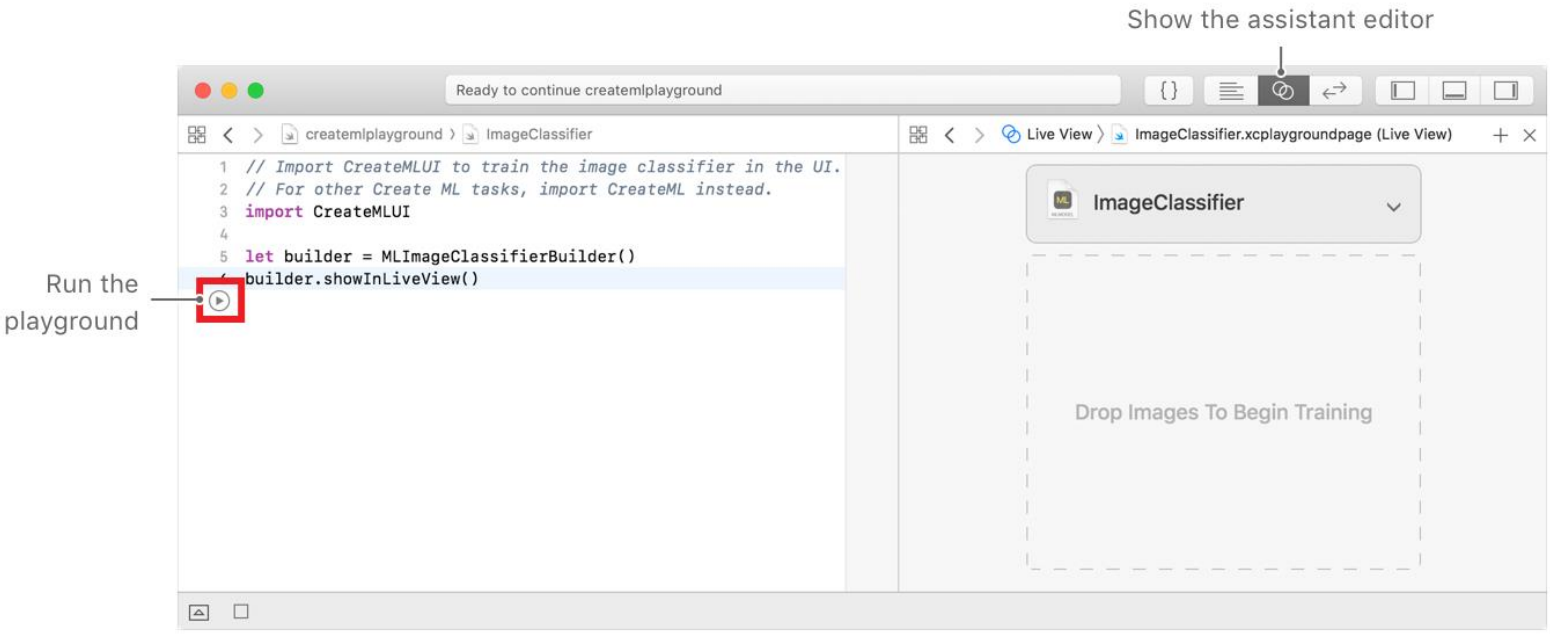
4.1.3.2. Εμφάνιση ενός ταξινομητή εικόνων στο Playground

Εφόσον είναι έτοιμα τα δεδομένα, ο προγραμματιστής δημιουργεί ένα νέο έργο (project) στο Xcode Playground με στόχο (target) macOS. Με το Playground για να δημιουργεί μια MLImageClassifierBuilder αναφορά όπως φαίνεται παρακάτω:

```
// Import CreateMLUI to train the image classifier in the UI.
// For other Create ML tasks, import CreateML instead.
import CreateMLUI

let builder = MLImageClassifierBuilder()
builder.showInLiveView()
```

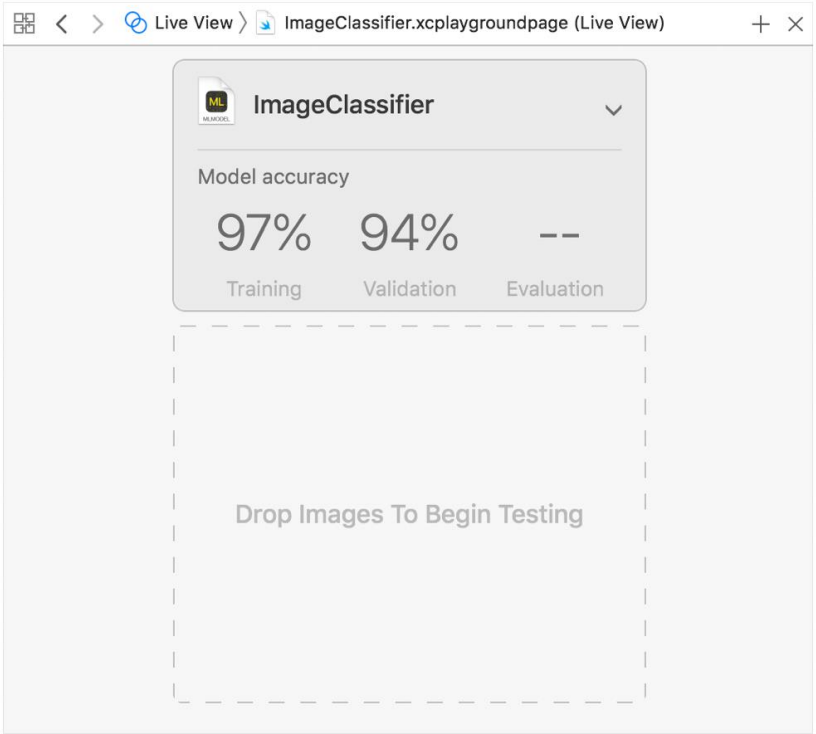
Έπειτα, εμφανίζει τον βοηθητικό συντάκτη του Xcode και τρέχει τον κώδικα στο Playground, και τότε παρουσιάζεται ένας γραφικός ταξινομητής φωτογραφιών:



Έτσι μόνο με 3 γραμμές κώδικα το Xcode δίνει την δυνατότητα στον προγραμματιστή να εκπαιδεύσει ένα δικό του ταξιμομητή εικόνων[32].

4.1.3.3. Εκπαίδευση

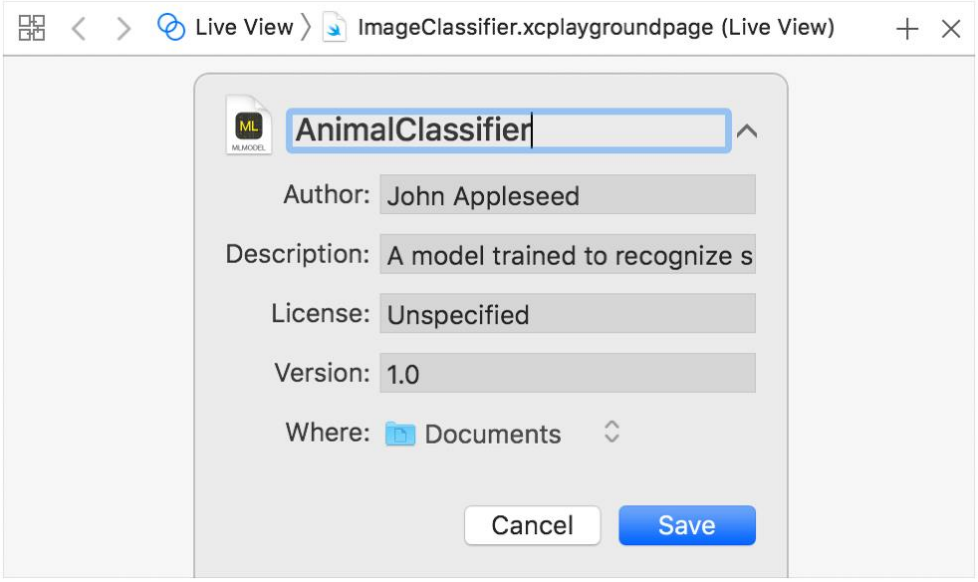
Ο προγραμματιστής μπορεί να σύρει το φάκελο Training Data μέσα στο γραφικό περιβάλλον που εμφανίστηκε στο Playground και τότε η διαδικασία της ανάλυσης των εικόνων αρχίζει αυτόματα. Σαν μέρος της διαδικασίας, τα δεδομένα χωρίζονται αυτόματα από τον φάκελο Training Data σε πακέτο εκπαίδευσης και πακέτο αξιολόγησης. Τα 2 αυτά πακέτα επηρεάζουν τη διαδικασία αλλά με διαφορετικό τρόπο.
Όταν ολοκληρωθεί η διαδικασία, εμφανίζεται το ποσοστό ακρίβειας της εκπαίδευσης και της αξιολόγησης.



Έπειτα ο προγραμματιστής τοποθετεί τον φάκελο Test Data μέσα στο γραφικό περιβάλλον για να προχωρήσει η διαδικασία της δοκιμής του μοντέλου, το οποίο επεξεργάζεται όλες τις φωτογραφίες, κάνει πρόβλεψη για την καθεμία και τέλος εμφανίζει την ακρίβεια των αξιολογήσεων.
Αν κάτι δεν προχωρήσει σωστά, τότε πρέπει να γίνει επανάληψη της διαδικασίας.

4.1.3.4. Αποθήκευση του μοντέλου

Αφού ολοκληρωθεί η διαδικασία επιτυχώς τότε μπορεί να γίνει αποθήκευση ώστε να χρησιμοποιηθεί στην εφαρμογή. Δίνεται ένα νέο όνομα αλλάζοντας το προεπιλεγμένο ImageClassifier. Επίσης ο προγραμματιστής μπορεί να δώσει επιπλέον πληροφορίες για το μοντέλο. Τέλος επιλέγει το Save και το μοντέλο αποθηκεύεται σαν αρχείο .mlmodel στον φάκελο που επιλέχθηκε.



4.1.3.5. Προσθήκη του μοντέλου στην εφαρμογή

Ο προγραμματιστής μπορεί να χρησιμοποιήσει το μοντέλο αυτό αντικαθιστώντας το με το Classifying Images with Vision and Core ML του δοκιμαστικού κώδικα. Έτσι τώρα είναι σε θέση να αναγνωρίζει εικόνες με βάση το μοντέλο που δημιουργήθηκε πριν. Αφού ανοίξει το Xcode project, τοποθετεί μέσα το μοντέλο στο τμήμα πλοήγησης. Για να το χρησιμοποιήσει, απλά αλλάζει μια γραμμή κώδικα, αντικαθιστώντας αυτό:

```
let model = try VNCoreMLModel(for: MobileNet().model)
```

με αυτό:

```
let model = try VNCoreMLModel(for: AnimalClassifier().model)
```

Όπου στη θέση του AnimalClassifier().model τοποθετείται το όνομα που έχει επιλεγεί κατά την αποθήκευση του μοντέλου.

4.2. JSON αρχεία και η χρήση τους στο Xcode

4.2.1. Τι είναι το αρχείο JSON

Το JSON είναι μια μορφή αρχείου, που χρησιμοποιείται πολύ συχνά για την επικοινωνία μεταξύ εξυπηρετητή - πελάτη (server - client). Είναι πολύ δημοφιλές επειδή μπορεί να χρησιμοποιηθεί

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

από οποιαδήποτε πλατφόρμα, όπως iOS, Android, Windows Phone, φυλλομετρητές καθώς και επειδή είναι εύκολα αναγνώσιμο από τον χρήστη.

Όπως είναι εμφανές, ένα αρχείο JSON είναι ευανάγνωστο και κατανοητό. Η δομή ενός αρχείου JSON αποτελείται από δύο τύπους δεδομένων, Dictionaries και πίνακες (Array). Τα Dictionaries περιέχουν ένα ή περισσότερα ζεύγη κλειδιού-τιμής τα οποία κλείνονται μέσα σε { }. Οι πίνακες περιέχουν μια λίστα από ταξινομημένα αντικείμενα και κλείνονται μέσα σε [ ]. Σχεδόν κάθε

γλώσσα προγραμματισμού περιλαμβάνει αυτούς τους τύπους δεδομένων, κάτι που εξηγεί γιατί τα αρχεία JSON υποστηρίζονται από κάθε γλώσσα. Οι ακόλουθοι τύποι δεδομένων υποστηρίζονται απο JSON:

- Αλφαριθμητικά
- Αριθμοί (ακέραιοι, πραγματικοί κλπ)
- Λογικές μεταβλητές
- Πίνακες
- Dictionaries

Ένας από τους λόγους που το JSON είναι τόσο δημοφιλές είναι ότι μπορεί εύκολα να διαβαστεί από τον άνθρωπο και να αναλυθεί από τις μηχανές. Κατά την διαδικασία της ανάλυσης, το μηχάνημα παίρνει ακατέργαστα δεδομένα και τα μετατρέπει σε δεδομένα που μπορούν να χρησιμοποιηθούν από την εφαρμογή.
Η Swift διαθέτει επεκτάσεις για να διαχειρίζεται τέτοια αρχεία τύπου .json. Έτσι όταν η εφαρμογή επικοινωνεί με μια διαδικτυακή εφαρμογή μέσω κάποιου εξυπηρετητή (server) και η πληροφορία επιστρέφει σε μορφή json, μπορεί να χρησιμοποιηθεί η κλάση JSONSerialization για να μετατραπεί το αρχείο JSON σε τύπους δεδομένων εύκολα διαχειρίσιμους μέσω της Swift, όπως είναι: Dictionary, Array, String, Number,Bool.

4.2.2. Εξαγωγή τιμών από το JSON

Η μέθοδος της κλάσης JSONSerialization jsonObject επιστρέφει μια τιμή τύπου Any και βγάζει σφάλμα αν δεν ήταν δυνατή η ανάλυση των δεδομένων.

```
import Foundation

let data: Data // received from a network request, for example
let json = try? JSONSerialization.jsonObject(with: data, options: [])
```

Ο προγραμματιστής μπορεί να χρησιμοποιήσει την προαιρετική δέσμευση και τον τελεστή as? σε μια εντολή if ή guard για να εξαγάγει μια τιμή τύπου σταθεράς. Για να λάβει μια τιμή Dictionary από ένα αντικείμενο JSON , μπορεί να το κάνει ως εξης [String: Any], ενώ για να λάβει μια τιμή

Array από έναν πίνακα JSON, ως [Any] (ή έναν πίνακα με έναν πιο συγκεκριμένο τύπο στοιχείου, όπως [String]).

```
// Example JSON with object root:
/*
        {
                "someKey": 42.0,
                "anotherKey": {
                        "someNestedKey": true
                }
        }
*/
if let dictionary = jsonWithObjectRoot as? [String: Any] {
        if let number = dictionary["someKey"] as? Double {
                // access individual value in dictionary
        }

        for (key, value) in dictionary {
                // access all key / value pairs in dictionary
        }

        if let nestedDictionary = dictionary["anotherKey"] as? [String: Any] {
                // access nested dictionary values by key
        }
}

// Example JSON with array root:
/*
        [
                "hello", 3, true
        ]
```

```
*/
if let array = jsonWithArrayRoot as? [Any] {
        if let firstObject = array.first {
                // access individual object in array
        }

        for object in array {
                // access all objects in array
        }

        for case let string as String in array {
                // access only string values in array
        }
}
```

Τα ενσωματωμένα γλωσσικά χαρακτηριστικά της Swift διευκολύνουν την ασφαλή εξαγωγή και επεξεργασία δεδομένων JSON που αποκωδικοποιούνται με τα APIs του Foundation, χωρίς την ανάγκη για εξωτερική βιβλιοθήκη ή πλαίσιο [34].

Κεφάλαιο 5

5.1 GPS - Παγκόσμιο Σύστημα Στιγματοθέτησης

Το Παγκόσμιο Σύστημα Στιγματοθέτησης ή όπως το ξέρουμε όλοι, GPS (Global Positioning System) είναι ένα παγκόσμιο σύστημα εντοπισμού γεωγραφικής θέσης, το οποίο βασίζεται σε ένα "πλέγμα" εικοσιτεσσάρων δορυφόρων της Γης, που ανήκει στην κυβέρνηση των Ηνωμένων Πολιτειών και λειτουργεί από την Πολεμική Αεροπορία των Ηνωμένων Πολιτειών. Πρόκειται για ένα παγκόσμιο δορυφορικό σύστημα πλοήγησης που παρέχει πληροφορίες γεωγραφικής κατανομής και χρόνου σε ένα δέκτη GPS οπουδήποτε στη Γη ή κοντά στη Γη, όπου υπάρχει απεριόριστη οπτική επαφή με τέσσερις ή περισσότερους δορυφόρους. Το GPS δεν απαιτεί από τον χρήστη να μεταδίδει δεδομένα και λειτουργεί ανεξάρτητα από οποιαδήποτε τηλεφωνική ή διαδικτυακή λήψη, αν και αυτές οι τεχνολογίες μπορούν να ενισχύσουν ττην ακρίβεια των πληροφοριών του GPS . Το GPS παρέχει δυνατότητες τοποθεσίας σε στρατιωτικούς, πολιτικούς και εμπορικούς χρήστες σε όλο τον κόσμο. Η κυβέρνηση των Ηνωμένων Πολιτειών δημιούργησε το σύστημα, το διατηρεί και το καθιστά ελεύθερα προσιτό σε οποιονδήποτε έχει δέκτη GPS. Η βασική ιδέα της τεχνολογίας του GPS βασίζεται στον χρόνο και τη γνωστή θέση των ειδικών δορυφόρων τύπου GPS. Οι δορυφόροι αυτοί φέρουν πολύ σταθερά ατομικά ρολόγια τα οποία συγχρονίζονται τόσο μεταξύ τους όσο και με τα επίγεια ρολόγια. Οποιαδήποτε μετατόπιση από τον πραγματικό χρόνο διορθώνεται καθημερινά. Με τον ίδιο ακριβώς τρόπο, οι δορυφορικές θέσεις είναι γνωστές με πολύ μεγάλη ακρίβεια. Οι δέκτες τύπου GPS έχουν επίσης ρολόγια, αλλά είναι λιγότερο σταθερά και λιγότερο ακριβή. Κάθε δορυφόρος GPS μεταδίδει συνεχώς ένα ραδιοφωνικό σήμα που περιέχει την τρέχουσα ώρα και δεδομένα σχετικά με τη θέση του. Είναι δεδομένο ότι η ταχύτητα αυτών των ραδιοκυμάτων είναι επίσης σταθερή και ανεξάρτητη από την δορυφορική ταχύτητα, η χρονική καθυστέρηση μεταξύ του δορυφορικού σήματος και του δέκτη είναι ανάλογη με την απόσταση από τον δορυφόρο στον δέκτη. Ένας δέκτης GPS παρακολουθεί πολλούς δορυφόρους και λύνει εξισώσεις για να καθορίσει την ακριβή θέση του δέκτη και την απόκλιση από τον πραγματικό χρόνο. Τουλάχιστον τέσσερις δορυφόροι πρέπει να είναι ορατοί απο τον δέκτη για να υπολογίσει τις τέσσερις άγνωστες ποσότητες: τρεις συντεταγμένες θέσης και τη χρονική απόκλιση από τη δορυφορική ώρα.

Ο εντοπισμός μέσω κινητού τηλεφώνου είναι μια διαδικασία για τον εντοπισμό της θέσης του κινητού τηλεφώνου, σε σταθερή ή και κινούμενη θέση. Η ταυτοποίηση μπορεί να συμβεί είτε μέσω της πολλαπλής εναπόθεσης ραδιοσημάτων μεταξύ των κυψελών του δικτύου και του τηλεφώνου είτε απλώς μέσω του GPS. Για να εντοπιστεί η θέση ενός κινητού τηλεφώνου χρησιμοποιώντας την πολλαπλότητα του ραδιοσήματος, πρέπει να εκπέμπει τουλάχιστον το σήμα περιαγωγής για να επικοινωνήσει με την επόμενη κοντινή κεραίας. Το Παγκόσμιο Σύστημα Κινητών Επικοινωνιών (GSM) βασίζεται στην ισχύ του σήματος του τηλεφώνου σε κοντινούς ιστούς κεραιών. Ο εντοπισμός θέσης μέσω κινητού τηλεφώνου μπορεί να περιλαμβάνει υπηρεσίες βάσει τοποθεσίας που αποκαλύπτουν τις πραγματικές συντεταγμένες ενός κινητού τηλεφώνου, μια τεχνολογία που χρησιμοποιείται από τις εταιρείες τηλεπικοινωνιών για την προσέγγιση της θέσης ενός κινητού τηλεφώνου και συνεπώς και του χρήστη του.

Η τοποθεσία ενός κινητού τηλεφώνου μπορεί να καθοριστεί χρησιμοποιώντας :
Τα δίκτυα τηλεφωνίας. Η τοποθεσία ενός κινητού τηλεφώνου μπορεί να καθοριστεί χρησιμοποιώντας την υποδομή του δικτύου του παρόχου υπηρεσιών. Το πλεονέκτημα των τεχνικών που βασίζονται στο δίκτυο, από την άποψη του παρόχου υπηρεσιών, είναι ότι μπορούν να υλοποιηθούν χωρίς παρενοχλήσεις, χωρίς να επηρεάζονται τα κινητά τηλέφωνα. Η ακρίβεια των τεχνικών που βασίζονται στο δίκτυο ποικίλλει, με την ταυτοποίηση των κυψελών να έχει την ελάχιστη ακρίβεια την τριγωνοποίηση να έχει μέτρια ακρίβεια και τις νεότερες μεθόδους χρονισμού να είναι οι πιο ακριβείς.

- Handset-based (κινητό τηλέφωνο), αυτή η τεχνική καθορίζει τη θέση του κινητού τηλεφώνου θέτοντας τη θέση του μέσω της αναγνώρισης κυψέλης και των ισχυρών σημάτων τόσο του κεντρικού όσο και των γειτονικών κελιών. Επιπλέον, αν το κινητό είναι εξοπλισμένο με GPS τότε μπορούν να αποστέλλονται σημαντικά ακριβέστερες πληροφορίες θέσης από το κινητό τηλέφωνο στον φορέα. Το βασικό μειονέκτημα των τεχνικών που βασίζονται στο κινητό τηλέφωνο, είναι η ανάγκη εγκατάστασης λογισμικού στη συσκευή. Αυτό απαιτεί την ενεργή συνεργασία του συνδρομητή κινητής τηλεφωνίας καθώς και του λογισμικού που πρέπει να είναι σε θέση να χειρίζεται τα διάφορα λειτουργικά συστήματα των φορητών συσκευών. Συνήθως, τα smartphones, όπως το Symbian, το Windows Mobile, το Windows Phone, το BlackBerry OS, το iOS ή το Android, μπορούν να τρέχουν τέτοιου είδους λογισμικά.

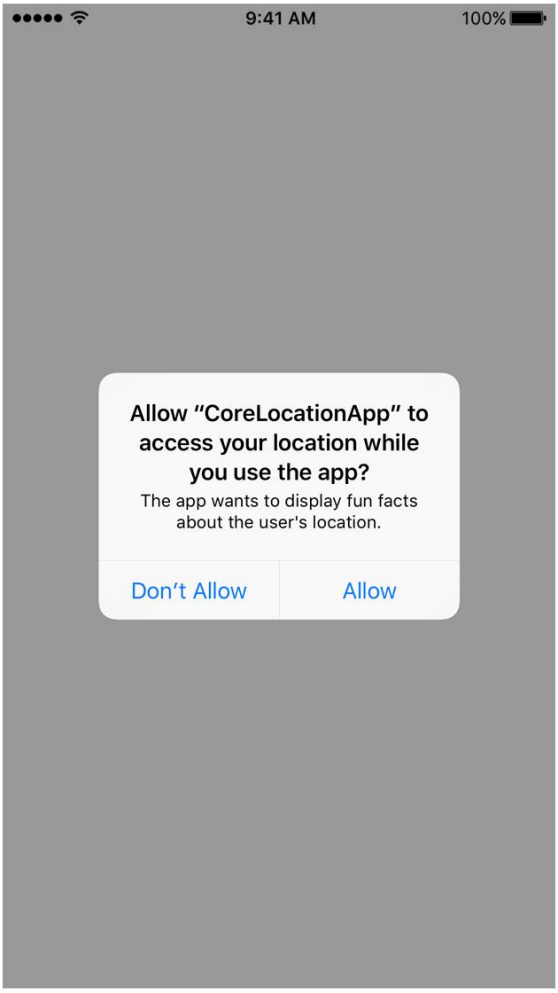
- Με κάρτα SIM, χρησιμοποιώντας τη μονάδα ταυτοποίησης συνδρομητή GSM και το Universal Mobile Telecommunications System (UMTS) των κινητών τηλεφώνων, είναι δυνατή η λήψη ακατέργαστων ραδιομετρήσεων από τη συσκευή. Οι διαθέσιμες μετρήσεις περιλαμβάνουν το αναγνωριστικό των κυψελών εξυπηρέτησης, το χρόνο και την ισχύ του σήματος. Ο τύπος πληροφοριών που λαμβάνεται μέσω της κάρτας SIM μπορεί να διαφέρει από αυτόν που είναι διαθέσιμος από το κινητό τηλέφωνο.

- Με δεδομένα Wi-Fi, τα οποία μπορούν επίσης να χρησιμοποιηθούν για τον προσδιορισμό της θέσης του κινητού τηλεφώνου. Η κακή απόδοση των μεθόδων που βασίζονται στο GPS σε εσωτερικούς χώρους και η αυξανόμενη δημοτικότητα του Wi-Fi έχουν οδηγήσει τις εταιρείες στο να σχεδιάσουν νέες και πιο αποδοτικές μεθόδους εντοπισμού της τοποθεσίας με βάση το Wi-Fi σε εσωτερικούς χώρους. Τα περισσότερα smartphones συνδυάζουν τα παγκόσμια δορυφορικά συστήματα πλοήγησης, όπως το GPS και το GLONASS, με συστήματα εντοπισμού Wi-Fi.

- Τα υβριδικά συστήματα, χρησιμοποιούν έναν συνδυασμό τεχνολογιών βασισμένων σε δίκτυα και τεχνολογιών που βασίζονται σε κινητά τηλέφωνα για τον καθορισμό της θέσης. Και οι δύο τύποι δεδομένων χρησιμοποιούνται από το τηλέφωνο για να κάνουν την τοποθεσία ακριβέστερη (δηλ. A-GPS). Εναλλακτικά, ο εντοπισμός της θέσης με τον συνδυασμό των δύο συστημάτων μπορεί να επιτευχθεί και απευθείας από τους δορυφόρους και στη συνέχεια οι πληροφορίες να αποστέλλονται μέσω του δικτύου στο πρόσωπο που προσπαθεί να εντοπίσει το τηλέφωνο. Τέτοια συστήματα περιλαμβάνουν τους Χάρτες Google. Υπάρχουν επίσης υβριδικά συστήματα εντοπισμού θέσης που συνδυάζουν πολλές διαφορετικές μεθόδους προσέγγισης τοποθεσίας κινητών συσκευών με Wi-Fi, WiMAX, GSM, LTE, διευθύνσεις IP και δεδομένα δικτύων[42], [43].

5.2 Ανάπτυξη εφαρμογών GPS με την χρήση Xcode

Για να αποκτήσουν τη γεωγραφική θέση και τον προσανατολισμό μιας συσκευής η Apple προσφέρει στους προγραμματιστές της το Core Location μέσω της γλώσσας Swift. Το Core Location παρέχει υπηρεσίες για τον προσδιορισμό της γεωγραφικής θέσης μιας συσκευής, του υψομέτρου, του προσανατολισμού ή της θέσης σε σχέση με ένα κοντινό iBeacon (είναι ένα πρωτόκολλο που αναπτύχθηκε από την Apple και παρουσιάστηκε στο συνέδριο της Apple Worldwide Developers Conference το 2013.). Το πλαίσιο (framework) χρησιμοποιεί όλο το διαθέσιμο ενσωματωμένο υλικό, όπως Wi-Fi, GPS, Bluetooth, μαγνητόμετρο, βαρόμετρο και υλικό από τις κυψέλεγια τη συλλογή δεδομένων. Την πρώτη φορά που η εφαρμογή ζητά την εξουσιοδότηση από το χρήστη, η κατάσταση εξουσιοδότησης είναι απροσδιόριστη και το σύστημα ζητά από τον χρήστη να παραχωρήσει ή να απορρίψει το αίτημα όπως φαίνεται στο σχήμα.



Το σύστημα καταγράφει την απόκριση του χρήστη και δεν εμφανίζει αυτόν τον πίνακα σε μεταγενέστερα αιτήματα. Αφού ζητηθεί άδεια και προσδιοριστεί αν υπάρχουν διαθέσιμες υπηρεσίες, μόνο τότε είναι δυνατό να χρησιμοποιηθεί το αντικείμενο CLLocationManager και λάβει ο προγραμματιστής τα αποτελέσματα στο σχετικό αντικείμενο.

Το Core Location προσφέρει τρεις διαφορετικές υπηρεσίες για την εξαγωγή της τοποθεσίας του χρήστη. Κάθε υπηρεσία έχει διαφορετικά οφέλη και διαθέτει διαφορετικές απαιτήσεις ισχύος και εξουσιοδότησης. Ο χρήστης μπορεί να χρησιμοποιήσει μια ενιαία υπηρεσία ή πολλές υπηρεσίες σε διαφορετικές χρονικές στιγμές, ανάλογα με τις ανάγκες του.

- Υπηρεσία τοποθεσίας με βάση τις επισκέψεις που πραγματοποιεί ο χρήστης: Είναι ο πιο αποδοτικός τρόπος για τη συγκέντρωση δεδομένων τοποθεσίας. Αυτή η υπηρεσία παρέχει ενημερώσεις σχετικά με την τοποθεσία όταν ο χρήστης μετά από κάποιο χρόνο σε μία τοποθεσία αρχίζει να κινείται. Κάθε ενημέρωση περιλαμβάνει τόσο την τοποθεσία όσο και το χρόνο που έμεινε σε εκείνη την τοποθεσία. Αυτή η υπηρεσία δεν προορίζεται για πλοήγηση ή άλλες δραστηριότητες σε πραγματικό χρόνο. Αντ 'αυτού, χρησιμοποιείται για να προσδιορίσει τη συμπεριφορά του χρήστη και να εφαρμόσει αυτές τις γνώσεις σε άλλα σημεία της εφαρμογής. Για παράδειγμα, μια εφαρμογή μουσικής μπορεί να προετοιμάσει ένα playlist προπόνησης όταν ο χρήστης φύγει από το σπίτι του και πηγαίνει προς το γυμναστήριο. Αυτή η υπηρεσία απαιτεί πάντα έγκριση. Για να ξεκινήσει ο προγραμματιστής την υπηρεσία αυτή, καλεί τη μέθοδο startMonitoringVisits () του διαχειριστή τοποθεσίας. Με αυτή την υπηρεσία, ο διαχειριστής τοποθεσίας αγνοεί τις τιμές των distanceFilter και desiredAccuracy, επομένως δεν χρειάζεται να τις διαμορφώσει. Ο διαχειριστής τοποθεσίας παρέχει ενημερώσεις στη μέθοδο locationManager (_: didVisit :).

```
func startReceivingVisitChanges() {
    let authorizationStatus = CLLocationManager.authorizationStatus()
    if authorizationStatus != .authorizedAlways {
        // User has not authorized access to location information.
        return
    }

    if !CLLocationManager.locationServicesEnabled() {
        // This service is not available.
        return
    }
    locationManager.startMonitoringVisits()
}
```

- Υπηρεσία σημαντικών αλλαγών στην τοποθεσία του χρήστη: Πρόκειται για μια εναλλακτική λύση για εφαρμογές που δεν χρειάζεται συχνές ενημερώσεις ή την ακρίβεια που προσφέρει το GPS. Αυτή η υπηρεσία βασίζεται σε εναλλακτικές λύσεις χαμηλότερης ισχύος για τον προσδιορισμό της θέσης του χρήστη και παρέχει ενημερώσεις μόνο όταν προκύπτουν σημαντικές αλλαγές στη συγκεκριμένη τοποθεσία. Χρησιμοποιείται κυρίως για να προτείνει κοντινά σημεία ενδιαφέροντος στο χρήστη όταν περπατάει. Αυτή η υπηρεσία απαιτεί πάντα έγκριση. Για να ξεκινήσει, καλείται η μέθοδος startMonitoringSignificantLocationChanges () του διαχειριστή τοποθεσίας. Με αυτήν την υπηρεσία, ο διαχειριστής τοποθεσίας αγνοεί τις τιμές στις ιδιότητες distanceFilter και desiredAccuracy, επομένως δεν χρειάζεται να τις διαμορφώσει. Ο διαχειριστής τοποθεσίας παρέχει ενημερώσεις στη μέθοδο locationManager (_: didUpdateLocations :) :).

```
func startReceivingSignificantLocationChanges() {
    let authorizationStatus = CLLocationManager.authorizationStatus()
    if authorizationStatus != .authorizedAlways {
        // User has not authorized access to location information.
        return
    }

    if !CLLocationManager.significantLocationChangeMonitoringAvailable() {
        // The service is not available.
        return
    }
    locationManager.delegate = self
    locationManager.startMonitoringSignificantLocationChanges()
}
```

- Υπηρεσία τυπικής τοποθεσίας: Πρόκειται για μια λύση σχετικά με τη λήψη της θέσης του χρήστη σε πραγματικό χρόνο. Αυτή η υπηρεσία χρησιμοποιεί σημαντικά περισσότερη ισχύ από τις άλλες υπηρεσίες, αλλά παρέχει τις πιο ακριβείς και άμεσες πληροφορίες. Καλό είναι να χρησιμοποιείται αυτή η υπηρεσία μόνο όταν υπάρχει πραγματικά ανάγκη για ακρίβεια στις πληροφορίες, όπως σε οδηγίες πλοήγησης ή για την καταγραφή της διαδρομής ενός χρήστη σε μια πεζοπορία. Αυτή η υπηρεσία απαιτεί πάντα έγκριση. Για να ξεκινήσει διαμορφώνεται το αντικείμενο CLLocationManager και καλείται η μέθοδος startUpdatingLocation (). Ο διαχειριστής τοποθεσίας παρέχει ενημερώσεις στο διαχειριστή locationManager (_: didUpdateLocations :) :) .

```
let locationManager = CLLocationManager()
func startReceivingLocationChanges() {
   let authorizationStatus = CLLocationManager.authorizationStatus()
   if authorizationStatus != .authorizedWhenInUse && authorizationStatus != .auth
      // User has not authorized access to location information.
      return
   }
   // Do not start services that aren't available.
   if !CLLocationManager.locationServicesEnabled() {
      // Location services is not available.
      return
   }
   // Configure and start the service.
   locationManager.desiredAccuracy = kCLLocationAccuracyHundredMeters
   locationManager.distanceFilter = 100.0  // In meters.
   locationManager.delegate = self
   locationManager.startUpdatingLocation()
}
```

Η Apple προτείνει να επιλέγεται πάντα η πιο αποδοτική υπηρεσία που εξυπηρετεί τις ανάγκες της εφαρμογής. Για την εξοικονόμηση ενέργειας, καλό είναι να απενεργοποιούνται οι υπηρεσίες τοποθεσίας ή να γίνεται μετάβαση σε μια εναλλακτική λύση χαμηλής κατανάλωσης, όταν δεν χρειάζονται τα δεδομένα θέσης που προσφέρει η υπηρεσία.

5.3 Apple Maps

Στις 11 Ιουνίου 2012, κατά τη διάρκεια της διάσκεψης Apple Worldwide Developers Conference, η Apple ανακοίνωσε την αρχική έκδοση των χαρτών Apple και αποκάλυψε ότι η εφαρμογή θα αντικαταστήσει τους Χάρτες Google ως την προεπιλεγμένη υπηρεσία χαρτογράφησης από iOS 6 και μετά. Η Apple ανακοίνωσε επίσης ότι η εφαρμογή θα περιλαμβάνει πλοήγηση "turn-by-turn", 3D χάρτες, Flyovers και τον εικονικό βοηθό Siri. Επιπλέον, δήλωσε ότι οι χρήστες iPhone θα μπορούσαν να πλοηγηθούν στο Apple Maps ενώ βρίσκονται στην κλειδωμένη οθόνη. Η υπηρεσία χαρτογράφησης κυκλοφόρησε στις 19 Σεπτεμβρίου 2012. Μετά την κυκλοφορία, το Apple Maps επικρίθηκε έντονα, γεγονός που οδήγησε σε δημόσια συγνώμη από τον CEO της Apple, Tim Cook στα τέλη Σεπτεμβρίου και την αναχώρηση δύο βασικών υπαλλήλων της Apple. Πριν από την εκκίνηση της εφαρμογής Apple Maps ως προεπιλεγμένης εφαρμογής χαρτογράφησης στο iOS, οι Χάρτες Google κατείχαν αυτή τη θέση από την έκδοση της πρώτης γενιάς του iPhone το 2007. Στα τέλη του 2009, οι εντάσεις μεταξύ της Google και της Apple άρχισαν να αυξάνονται όταν η έκδοση Android των Χαρτών Google χαρακτήρισε τη πλοήγηση "turn-by-turn", μια λειτουργία που η έκδοση του iOS δεν είχε. Εκείνη την εποχή, η Apple ισχυρίστηκε ότι η Google συνέλεξε πάρα πολλά δεδομένα από τους χρήστες. Όταν η Apple έκανε το iOS 6 διαθέσιμο, οι Χάρτες Google ήταν προσβάσιμοι από τους χρήστες του iOS 6 μόνο μέσω του διαδικτύου. Παρόλο που η Google δεν ξεκίνησε αμέσως μια δική της εφαρμογή χαρτογράφησης , λίγο μετά την ανακοίνωση των Χαρτών της Apple, η Google πρόσθεσε ένα ισοδύναμο χαρακτηριστικό γνώρισμα με το Flyover των Χαρτών της Apple στην εφαρμογή Google Earth. Τρεις μήνες αργότερα, τον Δεκέμβριο του 2012, οι Χάρτες Google κυκλοφόρησαν στο App Store. Αυτή η έκδοση των Χαρτών Google, σε αντίθεση με την προηγούμενη έκδοση, περιείχε πλοήγηση "turn-by-turn". Λίγο μετά, οι Χάρτες Google ήταν η πιο δημοφιλής δωρεάν εφαρμογή στο App Store.

Στο πρώτο έτος μετά την κυκλοφορία τους, οι Χάρτες της Apple έλαβαν ορισμένες βελτιώσεις με τις οποίες επιλύθηκαν διάφορα σφάλματα στην εφαρμογή. Ακόμη οι αλλαγές περιλάμβαναν την προσθήκη περισσότερων δορυφορικών εικόνων και τη διάθεση της πλοήγησης σε περισσότερες πόλεις. Το 2013, η Apple εξαγόρασε κάποιες εταιρείες για να βελτιώσει τους Χάρτες της, όπως την HopStop, την Embark, την WifiSlam και την Locations. Κατά τη διάρκεια του WWDC, τον Ιούνιο του 2013, η Apple ανακοίνωσε τη νέα έκδοση των χαρτών Apple στο iOS 7. Αυτή η νέα έκδοση είχε μια νέα εμφάνιση και διάφορες νέες λειτουργίες, όπως η λειτουργία πλήρους οθόνης,

η νυχτερινή λειτουργία, οι πληροφορίες κυκλοφορίας σε πραγματικό χρόνο, η πλοήγηση για πεζούς και η λειτουργία “Συχνές τοποθεσίες”. Το τελευταίο χαρακτηριστικό, το οποίο μπορεί να ενεργοποιηθεί και να απενεργοποιηθεί, εισήχθη για να καταγράψει τους προορισμούς που επισκέπτονται πιο συχνά οι χρήστες, προκειμένου να βελτιωθεί το Apple Maps. Επιπλέον, προστέθηκαν νέες δορυφορικές εικόνες. Στις 18 Σεπτεμβρίου 2013, η Apple κυκλοφόρησε iOS 7. Την εποχή εκείνη, το νέο iPhone 5S περιελάβανε έναν νέο μηχανισμό κίνησης, τον M7, ο οποίος μπορούσε να προσδιορίσει εάν ο χρήστης περπατάει ή οδηγεί ώστε να ρυθμίζει ανάλογα τη λειτουργία πλοήγησης.

Στις 8 Ιουνίου 2015, ο Craig Federighi, ανώτερος αντιπρόεδρος της Apple Software Engineering, ανακοίνωσε ότι η νέα έκδοση των χαρτών Apple στο iOS 9 θα έχει πληροφορίες σχετικά με τις δημόσιες συγκοινωνίες σε πολλές παγκόσμιες πόλεις. Η λειτουργία έγινε επίσης διαθέσιμη για το OS X El Capitan και το watchOS 2. Επιπλέον, η Apple πρόσθεσε τη λειτουργία "Nearby", η οποία δείχνει κοντινά σημεία ενδιαφέροντος σε διάφορες κατηγορίες. Με την συνεχή ενημέρωση, η εφαρμογή επιλέγει μια παράκαμψη σε περίπτωση καθυστέρησης λόγω αυξημένης κυκλοφορίας. Οι τρεις νέες εκδόσεις των λειτουργικών συστημάτων έγιναν διαθέσιμες τον Σεπτέμβριο του 2015. Εκτός από αυτές τις νέες κυκλοφορίες, η Apple εξαγόρασε κάποιες εταιρείες το 2015 για να βελτιώσει ακόμη περισσότερο την εφαρμογή χαρτογράφησης. Την άνοιξη του 2015, η Apple απέκτησε την εταιρία Coherent Navigation, η οποία παρέχει ακριβή δεδομένα θέσης μέσω του High Integrity GPS και την εταιρεία Mapsense, η οποία ανέπτυξε λογισμικό για να οργανώνει μεγάλους όγκους δεδομένων τοποθεσίας. Το 2016, οι χάρτες της Apple ανανεώθηκαν στη νέα έκδοση του λειτουργικού συστήματος και έλαβαν μια σειρά από νέες δυνατότητες, συμπεριλαμβανομένου του "Nearby" που είχε αποκλειστικά για το iOS. Τέσσερις μήνες αργότερα, ο CEO της Apple, Tim Cook, εγκαινίασε τα νέα γραφεία σε συνεργασία με την εταιρεία πληροφορικής RMSI Noida στην πανεπιστημιούπολη WaveRock στην ινδική πόλη Hyderabad. Το κέντρο ανάπτυξης επικεντρώνεται στην ανάπτυξη των χαρτών της Apple και απασχολεί 4.000 άτομα. Σύμφωνα με το ZDNet, το γραφείο των 250.000 τετραγωνικών ποδιών κοστίζει 25 εκατομμύρια δολάρια. Τον Σεπτέμβριο, κυκλοφόρησε το iOS 10. Η ενημέρωση του λειτουργικού συστήματος της Apple συνοδεύτηκε από ένα νέο σχεδιασμό του Apple Maps. Επιπλέον, η εφαρμογή ανοίχθηκε στους προγραμματιστές και απέκτησε μερικά νέα χαρακτηριστικά: μπορεί να προτείνει στο χρήστη τοποθεσίες με βάση την προηγούμενη χρήση της εφαρμογής, να θυμάται τη θέση όπου ο χρήστης σταθμεύει το όχημά του, επιτρέπει στον χρήστη να φιλτράρει τις

προτάσεις αναζήτησης , και η πλοήγηση "turn-by-turn" βελτιώθηκε. Η πλοήγηση μεγεθύνει αυτόματα τα zoom in και zoom out, δείχνει την κίνηση στους δρόμους και επιτρέπει στους χρήστες να αναζητούν σημεία ενδιαφέροντος κατά μήκος της διαδρομής. Αυτά τα χαρακτηριστικά είναι διαθέσιμα και για το CarPlay.

Το 2018, η Apple ανακοίνωσε ότι οι Χάρτες μπορούν να βοηθήσουν τους χρήστες να βρουν σταθμούς ποδηλάτων σε περισσότερες από 175 πόλεις σε 36 χώρες, όπως: Σαν Φρανσίσκο, Νέα Υόρκη, Μόντρεαλ, Λονδίνο και Brisbane. Το καλοκαίρι του 2018, ο ηγέτης των χαρτών Eddy Cue ανακοίνωσε μια φιλόδοξη εξέλιξη της εφαρμογής των χαρτών της Apple, η οποία χτίστηκε από την αρχή. Νέοι χάρτες θα διατίθενται για την περιοχή του Σαν Φρανσίσκο από τις αρχές Ιουλίου στο επόμενο iOS 12 beta. Ωστόσο, οι ενημερώσεις δεν θα απαιτούν το iOS 12 και θα εμφανίζονται στο Apple Watch και στους Mac. Οι ενημερωμένοι χάρτες με τα νέα δεδομένα είναι διαθέσιμοι σε όλη τη Βόρεια Καλιφόρνια από το iOS 12 beta 2 και μετά. Η Apple θα στείλει επίσης οχήματα χαρτογράφησης σε όλες τις ΗΠΑ και το Ηνωμένο Βασίλειο για να συγκεντρώσει τα δικά της δεδομένα και να προσθέσει λεπτομέρειες στους χάρτες.

Οι Χάρτες της Apple χρησιμοποιούν διανυσματικά γραφικά, τα οποία επιτρέπουν στην εφαρμογή να χρησιμοποιεί λιγότερα δεδομένα από ό, τι ο ανταγωνιστής της, το Google Maps. Ο χάρτης διαθέτει τέσσερα διαθέσιμα επίπεδα: κανονικό χάρτη, δορυφορική προβολή, υβριδική προβολή (συνδυασμός κανονικής και δορυφορικής προβολής) και δημόσια συγκοινωνία. Ο κύριος πάροχος των δεδομένων των χαρτών είναι η TomTom, αλλά τα δεδομένα παρέχονται επίσης από τα δεδομένα πλοήγησης των αυτοκινήτων, την Hexagon AB, τις τεχνολογίες Intermap, το OpenStreetMap και το Waze. Η Apple ανανέωσε τη συμφωνία της με την TomTom το 2015. Η TomTom είναι η μητρική εταιρεία της Tele Atlas, η οποία χρησιμοποιείται επίσης από τον ανταγωνιστή των Χαρτών της Apple, τους Χάρτες Google. Οι δορυφορικές εικόνες προέρχονται από το DigitalGlobe. Τα iPhones που βρίσκονται στην Κίνα χρησιμοποιούν αντ 'αυτού δεδομένα από το AutoNavi. Οι χάρτες της Apple μπορούν να χρησιμοποιηθούν για τον προγραμματισμό διαδρομών. Η υπηρεσία πλοήγησης διαθέτει περιηγητή περιστροφής προς τα πίσω και οδηγίες με ομιλία για οχήματα, πεζούς και μέσα μαζικής μεταφοράς. Σύμφωνα με την Apple, η λειτουργία πλοήγησης είναι διαθέσιμη σε 56 χώρες σε όλο τον κόσμο. Οι χάρτες Apple μπορούν επίσης να χρησιμοποιηθούν για να δώσουν πληροφορίες σχετικά με την κυκλοφορία σε πραγματικό χρόνο. Επιπλέον, ο εικονικός βοηθός της Apple, Siri, είναι ενσωματωμένος στους Χάρτες της Apple. Ο

χάρτης παρουσιάζει σημεία ενδιαφέροντος που παρέχονται από περίπου είκοσι εταιρείες, συμπεριλαμβανομένων των ξενοδοχείων Booking.com, Foursquare, TripAdvisor και Yelp. Τα δεδομένα από το Foursquare προστέθηκαν στα τέλη του 2015. Τέλος, οι χρήστες μπορούν να αποθέσουν καρφίτσες στο χάρτη για να αποθηκεύσουν θέσεις για ανάκτηση κάποια μεταγενέστερη χρονική στιγμή [35], [36].

Κεφάλαιο 6

6.1. Τα μέρη της εφαρμογής

Στην κλάση MapViewController που βρίσκεται το αρχείο MapViewController.swift υπάρχουν αρχικά τα τέσσερα βασικά import στις βιβλιοθήκες που χρειαζόμαστε για τις βασικές εντολές στο UIKit, για τις μεθόδους του MapKit, για την σύνδεση με την βάση Firebase Database και με το Alamofire για την λήψη του json.

```
import UIKit
import MapKit
import FirebaseDatabase
import Alamofire
```

Η κλάση μας κληρονομεί τις ακόλουθες κλάσεις:

```
class MapViewController: UIViewController, MKMapViewDelegate,
    CLLocationManagerDelegate,UITableViewDelegate,UITableViewDataSource {
```

Τα IBOutlet παρακάτω είναι ειδικές μεταβλητές που συνδέουν τα UIViews με τον κώδικα της εφαρμογής ώστε να είμαστε σε θέση να προσαρμόζουμε το καθένα ξεχωριστά συμφώνα με τις λειτουργίες που θέλουμε να εκτελεί το καθένα. Εδώ χρειάζεται να συνδέσουμε το κουμπί για την αναγνώριση φωτογραφιών, τον πίνακα με τα σημεία ενδιαφέροντος και τέλος τον χάρτη.

```
@IBOutlet var MLButton: RoundedShadowButton!
@IBOutlet var tableView: UITableView!
@IBOutlet var mapView: MKMapView!
```

Global Μεταβλητές για να είναι ορατές σε όλες τις μεθόδους.

```
let locationManager = CLLocationManager()
let authorizationStatus = CLLocationManager.authorizationStatus()
let regionRadius: Double = 100
var passedImage: UIImage! = nil
var mapHasCenteredOnce = false
var geoFire: GeoFire!
var geoFireRef: DatabaseReference!
var location : Location!
var locations2 = [Location]()
var spots = [CLLocation]()
```

Η viewDidLoad() είναι η κύρια συνάρτηση της κλάσεις, στην οποία αρχικοποιούνται μέθοδοι αλλά και μεταβλητές.

```
override func viewDidLoad() {
    super.viewDidLoad()
    MLButoon.layer.cornerRadius = 15

    //mapView
    mapView.showsCompass = true
    mapView.showsScale = true
    mapView.delegate = self
    mapView.userTrackingMode = MKUserTrackingMode.follow
    geoFireRef = Database.database().reference()
    geoFire = GeoFire(firebaseRef: geoFireRef)
    configureLocationServices()
    downloadLocationsDetails{ }

    //tableView
    tableView.dataSource = self
    tableView.delegate = self
    self.tableView.reloadData()

    //3D Force touch
    registerForPreviewing(with: self, sourceView: tableView!)
}
```

Η μέθοδος downloadLocationsDetails(completed: @escaping DownloadComplete) εκτελεί την λήψη του JSON file για τις τοποθεσίες με τις ιδιότητές τους.

```
func downloadLocationsDetails(completed: @escaping DownloadComplete) {
    Alamofire.request(JsonURL).responseJSON { response in
        let result = response.result
        print(response)
        if let jsondict = result.value as? Dictionary<String, AnyObject> {
             if let list = jsondict["list"] as? [Dictionary<String, AnyObject>] {
                for locObj in list {
                    let name = locObj["name"] as? String
                    let id = locObj["id"] as? Int
                    let lon = locObj["lon"] as? CLLocationDegrees
                    let lat = locObj["lat"] as? CLLocationDegrees
                    let iconURL = locObj["iconURL"] as? String
                    let imageURL = locObj["imageURL"] as? String
                    let description = locObj["description"] as? String

                    //for distance value
                    let currentLoc = CLLocation(latitude: lat!, longitude: lon!)
                    //print(currentLoc,"**********")
                    let distanceFromLocation: CLLocationDistance
                    distanceFromLocation = (self.locationManager.location?.distance(from: currentLoc))!
                    let location = Location(name: name!,id: id!, longitude: lon!, latitude: lat!, iconURL:
                        iconURL!, imageURL: imageURL!, description: description!, distance:
                        distanceFromLocation)
                    self.locations2.append(location)

                    //create Location on GeoFire
                    self.createSighting(forLongitude: lon!, forLatitude: lat!, withID: id!)
        }
        //sort by nearest distance
        self.locations2.sort(by: {$0.distance < $1.distance})
        self.tableView.reloadData()
        completed()
            }
        }
    }
```

Η μέθοδος centerMapOnUserLocation() τοποθετεί στο κέντρο του χάρτη την ακριβή τοποθεσία του χρήστη.

```
func centerMapOnUserLocation() {
    guard let coordinate = locationManager.location?.coordinate else { return }
    let coordinateRegion = MKCoordinateRegion(center: coordinate, latitudinalMeters: regionRadius * 2.0,
        longitudinalMeters: regionRadius * 2.0)
    mapView.setRegion(coordinateRegion, animated: true)
}
```

Η μέθοδος configureLocationServices() είναι υπεύθυνη για την αποδοχή του χρήστη στην χρήση της τοποθεσίας του ως προσωπικό δεδομένο, χρειάζεται πάντα να είναι επιτρεπτό από τον χρήστη.

```
func configureLocationServices() {
    if authorizationStatus == .notDetermined {
        locationManager.requestAlwaysAuthorization()
    } else {
        return
    }
}
```

Η μέθοδος locationAuthStatus() ζητά την άδεια από τον χρήστη για χρήση της τοποθεσίας του.

```
func locationAuthStatus() {
    if CLLocationManager.authorizationStatus() == .authorizedWhenInUse {
        mapView.showsUserLocation = true
    }else {
        locationManager.requestWhenInUseAuthorization()
    }
```

Η locationManager() είναι υπεύθυνη για τον έλεγχο σε περίπτωση αλλαγής της κατάστασης για την άδεια του χρήστη.

```
func locationManager(_ manager: CLLocationManager, didChangeAuthorization status: CLAuthorizationStatus) {
    if status == .authorizedWhenInUse {
        mapView.showsUserLocation = true
    }
}
```

Η μέθοδος centerMapOnLocation() προσαρμόζει την εμβέλειά του χάρτη με κέντρο την συγκεκριμένη τρέχουμε τοποθεσία του χρήστη.

```
func centerMapOnLocation(location: CLLocation) {
    let coordinateRegion = MKCoordinateRegion.init(center: location.coordinate, latitudinalMeters: 100,
        longitudinalMeters: 100)
    mapView.setRegion(coordinateRegion, animated: true)
}
```

Η mapView(_ mapView: MKMapView, didUpdate userLocation: MKUserLocation) ελέγχει αυτόματα αν υπάρχει αλλαγή στην τρέχουσα τοποθεσία του χρήστη καθώς μετακινείται και ενημερώνει αντίστοιχα τον χάρτη με την νέα τοποθεσία.

```
func mapView(_ mapView: MKMapView, didUpdate userLocation: MKUserLocation) {
    if let loc = userLocation.location {
        if !mapHasCenteredOnce {
            centerMapOnLocation(location: loc)
            mapHasCenteredOnce = true
        }
```

Η createSighting() δεχεται σαν όρισμα μεταβλητες τύπου CLLocationDegrees και δημιουργεί, αν δεν υπάρχουν, στην βάση της υπηρεσίας Firebase μια νεα τοποθεσία με τις συντεταγμένες.

```
func createSighting(forLongitude lon: CLLocationDegrees, forLatitude lat: CLLocationDegrees, withID Id: Int) {
    let location = CLLocation(latitude: lat, longitude: lon)
    geoFire.setLocation(location, forKey: "\(Id)")
}
```

Η mapView(_ mapView: MKMapView, regionWillChangeAnimated animated: Bool) καλείται στην έναρξη και αλλάζει την ορατή περιοχή του χάρτη.

```
func mapView(_ mapView: MKMapView, regionWillChangeAnimated animated: Bool) {
    let loc = CLLocation(latitude: mapView.centerCoordinate.latitude, longitude:
        mapView.centerCoordinate.longitude)
    showSightingsOnMap(location: loc)
}
```

Η showSightingsOnMap(location: CLLocation) δέχεται σαν παράμετρο μια μεταβλητή τύπου CLLocation και κάθε φορά που καλείται δημιουργεί ένα αίτημα στο Firebase για λάβει την συγκεκριμένη τοποθεσία και να την εισάγει σαν πινέζα στον χάρτη.

```
func showSightingsOnMap(location: CLLocation) {
    let circleQuery = geoFire.query(at: location, withRadius: 2.5)
    _ = circleQuery.observe(.keyEntered, with: { (key: String!, location: CLLocation!) in
        if let _ = key, let location = location {
            let anno = MKPointAnnotation()
            anno.coordinate = location.coordinate
            self.mapView.addAnnotation(anno)

        }
    })
}
```

Η mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) -> MKAnnotationView?, εμφανίζει ένα annotation (μια κουκίδα) για την τοποθεσία του χρήστη.

```
func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) -> MKAnnotationView? {
    var annotationView: MKPinAnnotationView?
    if annotation.isKind(of: MKUserLocation.self) {
        return nil
    }
    return annotationView
}
```

Η numberOfSections() επιστρέφει τον αριθμό των section που θέλουμε στον πίνακα.

```
func numberOfSections(in tableView: UITableView) -> Int {
    return 1
}
```

Η tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int μας επιστρέφει τον αριθμό γραμμών που θέλουμε για τον πίνακά ανάλογα με το πλήθος των τοποθεσιών που έχουμε κάθε φορά.

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return locations2.count
}
```

Η tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell επιστρέφει ένα UITableViewCell διαμορφωμένο κατάλληλα με τα επιθυμητά δεδομένα για κάθε μια τοποθεσία ξεχωριστά σε κάθε γραμμή.

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    if let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath) as? LocationsCell{
        let location = locations2[indexPath.row]
        cell.updateTableViewUI(location: location)
        return cell
    }else{
        return UITableViewCell()
    }
}
```

Η tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) αναλαμβάνει να προωθήσει τον χρήστη στην επόμενη activity ανάλογα με ποια γραμμή του πίνακα επέλεξε για να λάβει επιπλέον πληροφορίες για την συγκεκριμένη τοποθεσία.

```
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        guard let detailVC = storyboard?.instantiateViewController(withIdentifier: "DVC") as? DetailViewController
            else { return }
        detailVC.initData(forLocation: locations2[indexPath.row])
        present(detailVC, animated: true, completion: nil)
    }
}
```

Η μέθοδος previewingContext(_ previewingContext: UIViewControllerPreviewing, viewControllerForLocation location: CGPoint) -> UIViewController? και η previewingContext(_ previewingContext: UIViewControllerPreviewing, commit viewControllerToCommit: UIViewController) χρησιμοποιείται ως επέκταση της αρχικής κλάσης και δίνει την δυνατότητα (σε περίπτωση που υποστηρίζεται από την τρέχουσα συσκευή) του 3D Force Touch, μιας λειτουργίας για άμεση απεικόνιση την επόμενης οθόνης, με ένα πιο "δυνατό άγγιγμα" στην οθόνη της συσκευής.

```
extension MapViewController: UIViewControllerPreviewingDelegate {
    func previewingContext(_ previewingContext: UIViewControllerPreviewing, viewControllerForLocation location:
        CGPoint) -> UIViewController? {
        guard let indexPath = tableView?.indexPathForRow(at: location), let cell = tableView?.cellForRow(at:
            indexPath) else { return nil }
        guard let DetailVC = storyboard?.instantiateViewController(withIdentifier: "DVC") as? DetailViewController
            else { return nil }
        DetailVC.initData(forLocation: locations2[indexPath.row])
        previewingContext.sourceRect = cell.contentView.frame
        return DetailVC
    }


   func previewingContext(_ previewingContext: UIViewControllerPreviewing, commit viewControllerToCommit:
       UIViewController) {
       show(viewControllerToCommit, sender: self)
   }
```

Η μέθοδος @IBAction func MLButton(_ sender: Any) αντιστοιχεί στο κουμπί ML που βρίσκεται στην αριστερή περιοχή του χάρτη και μεταβαίνει στην επόμενη activity για αναγνώριση εικόνας μέσω της πίσω κάμερας.

```
@IBAction func MLButton(_ sender: Any) {
    guard let CameraVC = storyboard?.instantiateViewController(withIdentifier: "camVC") as? CameraVC else
        { return }
    present(CameraVC, animated: true, completion: nil)
}
```

Η @IBAction func detectBtn(_ sender: Any) αντιστοιχεί στο κουμπί που βρίσκεται στην κάτω δεξιά περιοχή του χάρτη για εντοπισμό του χρήστη και τοποθέτηση στο κέντρο του χάρτη.

```
  @IBAction func detectBtn(_ sender: Any) {
      if authorizationStatus == .authorizedAlways || authorizationStatus == .authorizedWhenInUse {
          centerMapOnUserLocation()
      }
  }
```

Στην κλάση CameraVC που βρίσκεται στο αρχείο CameraVC.swift υπάρχουν αρχικά τα τέσσερα βασικά import στις βιβλιοθήκες που χρειαζόμαστε για της βασικές εντολές στο UIKit, για τις μεθόδους του AVFoundation για αφήγηση του Siri, και τέλος του CoreML και Vision για την αναγνωρίση φωτογραφιών.

```
import UIKit
import AVFoundation
import CoreML
import Vision
```

Τα IBOutlet είναι ειδικές μεταβλητές που συνδέουν τα UIViews με τον κώδικα της εφαρμογής ώστε να είμαστε σε θέση να προσαρμόζουμε το καθένα ξεχωριστά συμφώνα με τις λειτουργίες που θέλουμε να εκτελεί το καθένα.

```
@IBOutlet weak var cameraView: UIView!
@IBOutlet var capturedImage: RoundedShadowImageView!
@IBOutlet weak var identificationLbl: UILabel!
@IBOutlet weak var confidenceLbl: UILabel!
@IBOutlet weak var roundedLblView: RoundedShadowView!
@IBOutlet weak var spinner: UIActivityIndicatorView!
```

Global Μεταβλητές.

```
var captureSession: AVCaptureSession!
var cameraOutput: AVCapturePhotoOutput!
var previewLayer: AVCaptureVideoPreviewLayer!
var photoData: Data?
var speechSynthesizer = AVSpeechSynthesizer()
```

Η viewDidLoad() είναι η κύρια συνάρτηση της κλάσεις, στην οποία αρχικοποιούνται μέθοδοι αλλά και μεταβλητές. Στην συγκεκριμένη χρειάζεται μόνο το super.viewDidLoad().

```
override func viewDidLoad() {
    super.viewDidLoad()
}
```

Η μέθοδος @IBAction func backBtn(_ sender: Any) αντιστοιχεί σε κουμπί για την πολτοποίηση της λειτουργίας να μεταφερθεί ο χρήστης στην προηγουμένη οθόνη.

```
@IBAction func backBtn(_ sender: Any) {
    dismiss(animated: true, completion: nil)

}
```

Η μέθοδος override func viewDidAppear(_ animated: Bool) εμφανίζει την κάμερα.

```
override func viewDidAppear(_ animated: Bool) {
    super.viewDidAppear(animated)
    previewLayer.frame = cameraView.bounds
    speechSynthesizer.delegate = self
    spinner.isHidden = true
}
```

Η override func viewWillAppear(_ animated: Bool) αναλαμβάνει να ανοίξει, να φωτογραφίσει το κάδρο της κάμερας και να κρατήσει την εικόνα.

```
override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(animated)

    let tap = UITapGestureRecognizer(target: self, action: #selector(didTapCameraView))
    tap.numberOfTapsRequired = 1
    captureSession = AVCaptureSession()
    captureSession.sessionPreset = AVCaptureSession.Preset.hd1920x1080

    let backCamera = AVCaptureDevice.default(for: AVMediaType.video)
    do {
        let input = try AVCaptureDeviceInput(device: backCamera!)
        if captureSession.canAddInput(input) == true {
            captureSession.addInput(input)
        }
        cameraOutput = AVCapturePhotoOutput()
        if captureSession.canAddOutput(cameraOutput) == true {
            captureSession.addOutput(cameraOutput!)

            previewLayer = AVCaptureVideoPreviewLayer(session: captureSession!)
            previewLayer.videoGravity = AVLayerVideoGravity.resizeAspect
            previewLayer.connection?.videoOrientation = AVCaptureVideoOrientation.portrait

            cameraView.layer.addSublayer(previewLayer!)
            cameraView.addGestureRecognizer(tap)
            captureSession.startRunning()
        }
    } catch {
        debugPrint(error)
    }
}
```

Η @objc func didTapCameraView() είναι μια μέθοδός με ενσωμάτωση Objective-C που αναλαμβάνει να ‘παγώσει’ το UI του χρήστη μέχρι να ολοκληρωθεί η αναγνώριση της εικόνας.

```
@objc func didTapCameraView() {
    self.cameraView.isUserInteractionEnabled = false
    self.spinner.isHidden = false
    self.spinner.startAnimating()

    let settings = AVCapturePhotoSettings()
    let previewPixelType = settings.availablePreviewPhotoPixelFormatTypes.first!
    let previewFormat = [kCVPixelBufferPixelFormatTypeKey as String: previewPixelType, kCVPixelBufferWidthKey as
        String: 160, kCVPixelBufferHeightKey as String: 160]
    settings.previewPhotoFormat = previewFormat
    cameraOutput.capturePhoto(with: settings, delegate: self)
}
```

Η resultsMethod(request: VNRequest, error: Error?) αναλαμβάνει να μας επιστρέψει όσο το δυνατόν πιο ακριβές αποτέλεσμα για το τι είναι αυτό που αναγνώρισε και πόσο σίγουρο είναι αυτό.

```
func resultsMethod(request: VNRequest, error: Error?) {
    guard let results = request.results as? [VNClassificationObservation] else { return }

    for classification in results {
        if classification.confidence < 0.7 {
            let unknownObjectMessage = "No result. Please try again."
            self.identificationLbl.text = unknownObjectMessage
            synthesizeSpeech(fromString: unknownObjectMessage)
            self.confidenceLbl.text = ""
            break
        } else {
            let identification = classification.identifier
            let confidence = Int(classification.confidence * 100)
            self.identificationLbl.text = identification
            self.confidenceLbl.text = "CONFIDENCE: \(confidence)%"
            let completeSentence = "This looks like a \(identification) and I'm \(confidence) percent sure."
            synthesizeSpeech(fromString: completeSentence)
            break
        }
    }
}
```

Η synthesizeSpeech(fromString string: String) αφηγείται ένα αλφαριθμητικό με το αποτέλεσμα από την προηγούμενη μέθοδο.

```
    func synthesizeSpeech(fromString string: String) {
        let speechUtterance = AVSpeechUtterance(string: string)
        speechSynthesizer.speak(speechUtterance)
    }
}
```

Η μέθοδος photoOutput(_ output: AVCapturePhotoOutput, didFinishProcessingPhoto photo: AVCapturePhoto, error: Error?) χρησιμοποιείται ως επέκταση της αρχικής κλάσης και κληρονομεί την AVCapturePhotoCaptureDelegate με την οποία δέχεται την φωτογραφία που έβγαλε ο χρήστης και με βάση το εκπαιδευμένο μοντέλο τύπου ImageClassifier αναλαμβάνει να αναγνωρίσει το αντικείμενο που απεικονίζεται στην φωτογραφία.

```
extension CameraVC: AVCapturePhotoCaptureDelegate {
    func photoOutput(_ output: AVCapturePhotoOutput, didFinishProcessingPhoto photo: AVCapturePhoto, error: Error?)
        {
        if let error = error {
            debugPrint(error)
        } else {
            photoData = photo.fileDataRepresentation()

            do {
                let model = try VNCoreMLModel(for: MyImageClassifier().model)
                let request = VNCoreMLRequest(model: model, completionHandler: resultsMethod)
                let handler = VNImageRequestHandler(data: photoData!)
                try handler.perform([request])
            } catch {
                debugPrint(error)
            }
            let image = UIImage(data: photoData!)
            self.capturedImage.image = image
        }
    }
}
```

Ομοίως και η speechSynthesizer(_ synthesizer: AVSpeechSynthesizer, didFinish utterance: AVSpeechUtterance) παγώνει το UI του χρήστη μέχρις ότου να ολοκληρωθεί η αφήγηση.

```
extension CameraVC: AVSpeechSynthesizerDelegate {
    func speechSynthesizer(_ synthesizer: AVSpeechSynthesizer, didFinish utterance: AVSpeechUtterance) {
        self.cameraView.isUserInteractionEnabled = true
        self.spinner.isHidden = true
        self.spinner.stopAnimating()
    }
}
```

Στην κλάση DetailViewController που βρίσκεται στο αρχείο DetailViewController.swift υπάρχουν αρχικά τα δύο βασικά import στις βιβλιοθήκες που χρειαζόμαστε για της βασικές εντολές στο UIKit, για τις μεθόδους του AVFoundation για αφήγηση του Siri.

```
import UIKit
import AVFoundation
```

Η κλάση κληρονομεί τις ακόλουθες κλάσεις:

```
class DetailViewController: UIViewController, UIGestureRecognizerDelegate {
```

Τα IBOutlet είναι ειδικές μεταβλητές που συνδέουν τα UIViews με τον κώδικα της εφαρμογής ώστε να είμαστε σε θέση να προσαρμόζουμε το καθένα ξεχωριστά συμφώνα με τις λειτουργίες που θέλουμε να εκτελεί το καθένα για την κεντρική εικόνα, το κουμπί close, το όνομα και την περιγραφή του αξιοθέατου και τέλος την αφήγηση της περιγραφής μέσω του Siri.

```
@IBOutlet weak var mainImage: UIImageView!
@IBOutlet var closeBtnOutlet: UIButton!
@IBOutlet weak var name: UILabel!
@IBOutlet var SiriSpeech: UIButton!
@IBOutlet weak var descriptionLbl: UILabel!
```

Global Μεταβλητές για να είναι ορατες σε όλες τις μεθόδους.

```
let locationManager = CLLocationManager()
var speechSynthesizer = AVSpeechSynthesizer()
var passedImage: String!
var passedName: String!
var passedDescription: String!
var passedLatitude: CLLocationDegrees!
var passedLongitude: CLLocationDegrees!
var blurEffect = UIBlurEffect(style: UIBlurEffect.Style.light)
var blurEffectView: UIVisualEffectView!
```

Η initData(forLocation location: Location) αρχικοποιεί τις μεταβλητές που προέρχονται από την MapViewController κλάση.

```
func initData(forLocation location: Location) {
    self.passedName = location.name
    self.passedDescription = location.description
    self.passedLatitude = location.latitude
    self.passedLongitude = location.longitude
    self.passedImage = "\(location.id)"
}
```

Η viewDidLoad() είναι η κύρια συνάρτηση της κλάσεις, στην οποία αρχικοποιούνται οι μέθοδοι αλλά και οι μεταβλητές παίρνους τις σωστές τιμές.

```
override func viewDidLoad() {
    super.viewDidLoad()
    name.layer.masksToBounds = true
    descriptionLbl.layer.masksToBounds = true
    name.layer.cornerRadius = 15
    descriptionLbl.layer.cornerRadius = 15
    mainImage.image = UIImage(named: passedImage)
    name.text = passedName
    descriptionLbl.text = passedDescription
    addDoubleTap()
}
```

Η addDoubleTap() αναλαμβάνει να αναγνωρίσει τον αριθμό των χτύπων πάνω στην οθόνη.

```
func addDoubleTap() {
    let doubleTap = UITapGestureRecognizer(target: self, action: #selector(screenWasDoubleTapped))
    doubleTap.numberOfTapsRequired = 2
    doubleTap.delegate = self
    view.addGestureRecognizer(doubleTap)
}
```

Η @objc func screenWasDoubleTapped() είναι μια μέθοδός με ενσωμάτωση Objective-C που αναλαμβάνει να κλείσει την τρέχουσα οθόνη και να μεταβιβάσει τον χρήστη στην αρχική activity.

```
@objc func screenWasDoubleTapped() {
    dismiss(animated: true, completion: nil)
    self.speechSynthesizer.stopSpeaking(at: .immediate)
}
```

Η @IBAction func learnMoreBtn(_ sender: Any) αντιστοιχεί στο κουμπί πάνω δεξιά για εμφάνιση επιπλέον πληροφοριών που έχει λάβει η εφαρμογή από το JSON αρχείο.

```
@IBAction func learnMoreBtn(_ sender: Any) {
    blurEffectView = UIVisualEffectView(effect: blurEffect)
    blurEffectView.frame = mainImage.bounds
    blurEffectView.autoresizingMask = [.flexibleWidth, .flexibleHeight]
    mainImage.addSubview(blurEffectView)
    blurEffectView.isHidden = false
    closeBtnOutlet.isHidden = false
    descriptionLbl.isHidden = false
    SiriSpeech.isHidden = false

}
```

Η @IBAction func getDirections(_ sender: Any) αντιστοιχεί στο κουμπί κάτω αριστερά για λήψη οδηγιών μεταβαίνοντας στο Apple Maps με την επιθυμητή τοποθεσία.

```
@IBAction func getDirections(_ sender: Any) {
    let source = MKMapItem(placemark: MKPlacemark(coordinate: (locationManager.location?.coordinate)!))
    source.name = "Current Location"
    let destination = MKMapItem(placemark: MKPlacemark(coordinate: CLLocationCoordinate2D(latitude:
        passedLatitude , longitude: passedLongitude)))
    destination.name = passedName
    MKMapItem.openMaps(with: [source, destination], launchOptions: [MKLaunchOptionsDirectionsModeKey:
        MKLaunchOptionsDirectionsModeDriving])
}
```

Η @IBAction func closeBtn(_ sender: Any) κλείνει το view με τις πληροφορίες.

```
@IBAction func closeBtn(_ sender: Any) {
    self.speechSynthesizer.stopSpeaking(at: .immediate)
    closeBtnOutlet.isHidden = true
    descriptionLbl.isHidden = true
    SiriSpeech.isHidden = true
    blurEffectView.isHidden = true
}
```

Η synthesizeSpeech(fromString string: String) δέχεται σαν είσοδο και αφηγείται ένα αλφαριθμητικό.

```
func synthesizeSpeech(fromString string: String) {
    let speechUtterance = AVSpeechUtterance(string: string)
    speechSynthesizer.speak(speechUtterance)
}
```

Η @IBAction func siriSpeech(_ sender: Any) αντιστοιχεί σε κουμπί για να ξεκινήσει η αφήγησή της περιγραφής με τις πληροφορίες για την τοποθεσία.

```
@IBAction func siriSpeech(_ sender: Any) {
    synthesizeSpeech(fromString: passedDescription)
    }
}
```

Η μέθοδος speechSynthesizer(_ synthesizer: AVSpeechSynthesizer, didFinish utterance: AVSpeechUtterance) χρησιμοποιείται ως επέκταση της αρχικής κλάσης και κληρονομεί την AVSpeechSynthesizerDelegate με την οποία αναλαμβάνει να ενεργοποιήσει το κουμπί για να μπορέσει ο χρήστης να διακόψει την αφήγηση αν το επιθυμεί.

```
extension DetailViewController: AVSpeechSynthesizerDelegate {
    func speechSynthesizer(_ synthesizer: AVSpeechSynthesizer, didFinish utterance: AVSpeechUtterance) {
        closeBtnOutlet.isUserInteractionEnabled = true
    }
}
```

Στην κλάση Location που βρίσκεται στο αρχείο Locations.swift υπάρχουν αρχικά τα δύο βασικά import στις βιβλιοθήκες που χρειαζόμαστε.

```
import Foundation
import Alamofire
```

Ιδιωτικές Μεταβλητές.

```
private var _name: String!
private var _id: Int!
private var _description: String!
private var _longitude: CLLocationDegrees!
private var _latitude: CLLocationDegrees!
private var _iconURL: String!
private var _imageURL: String!
private var _distance: CLLocationDistance!
```

Αρχικοποίηση των ιδιοτήτων μεταβλητών.

```
var name:String {
    if _name == nil {
        _name = ""
    }
    return _name
}
var id:Int {
    if _id == nil {
        _id = 0
    }
    return _id
}
var iconURL:String {
    if _iconURL == nil {
        _iconURL = ""
    }
    return _iconURL
}
var imageURL:String {
    if _imageURL == nil {
        _imageURL = ""
    }
    return _imageURL
}

var description:String {
    if _description == nil {
        _description = ""
    }
    return _description
}
```

```
var latitude:CLLocationDegrees {
    if _latitude == nil {

    }
    return _latitude
}

var longitude:CLLocationDegrees {
    if _longitude == nil {

    }
    return _longitude
}

var distance:CLLocationDistance {
    if _distance == nil {

    }
    return _distance
}
```

Και τέλος περιέχει μια συνάρτηση αρχικοποίησης init() με είσοδο: όνομα τύπου String, ID τύπου Int, συντεταγμένες τύπου CLLocationDegrees(longitude & latitude), το URL για το εικονίδιο και την εικόνα τύπου String και τα δύο, περιγραφή τύπου String και τέλος την απόσταση τύπου CLLocationDistance με την οποία συνάρτηση αρχικοποίησης η κλάσηLocation αρχικοποιεί ένα αντικείμενο τύπου Location για κάθε μια τοποθεσία που λαμβάνει από το JSON αρχείο που κατεβάζει.

```
init(name: String,id: Int, longitude: CLLocationDegrees, latitude: CLLocationDegrees, iconURL: String, imageURL:
    String, description: String,distance: CLLocationDistance) {
    _name = name
    _id = id
    _longitude = longitude
    _latitude = latitude
    _iconURL = iconURL
    _imageURL = imageURL
    _description = description
    _distance = distance
}
```

Στην κλάση LocationsCell που βρίσκεται στο αρχείο LocationsCell.swift υπάρχουν αρχικά το βασικό import στην βιβλιοθήκη που χρειάζεται.

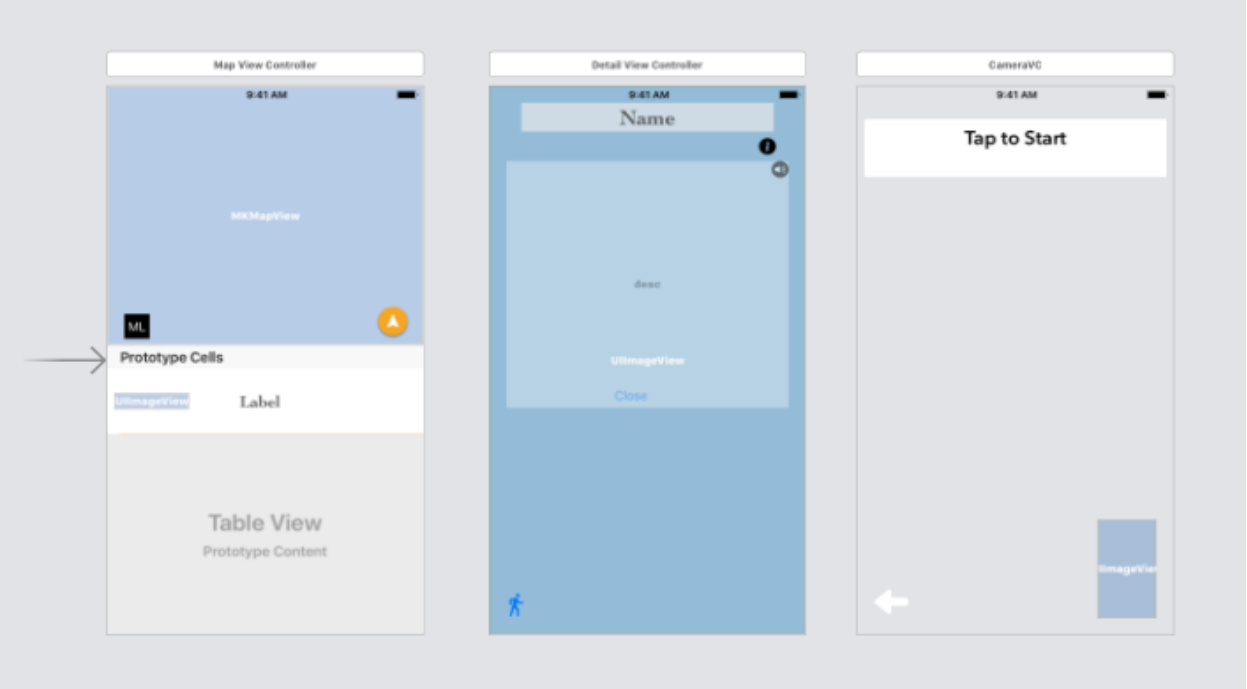
```
import UIKit
```

Τα IBOutlet είναι ειδικές μεταβλητές που συνδέουν τα UIViews με τον κώδικα της εφαρμογής ώστε να είμαστε σε θέση να προσαρμόζουμε το καθένα ξεχωριστά συμφώνα με τις λειτουργίες που θέλουμε να εκτελεί το καθένα για την εικόνα, το όνομα και την απόσταση του σημείου/αξιοθέατου απο την τοποθεσία του χρήστη.

```
@IBOutlet var photo: UIImageView!
@IBOutlet var distance: UILabel!
@IBOutlet var name: UILabel!
```

Η μέθοδος updateTableViewUI(location: Location)

```
func updateTableViewUI(location: Location) {
    self.distance.text = "\(String(format: "%.01f km",(location.distance)/1000.0))"
    self.name.text = location.name
    let url = URL(string: location.iconURL)!
   DispatchQueue.global().async {
        do {
            let data = try Data(contentsOf: url)
            DispatchQueue.global().sync {
                self.photo.image = UIImage(data: data)
            }
        } catch  {
            //handle the error
        }
    }
}
```



Τέλος στο αρχείο Main.storyboard έχουμε την σχεδιασμένη εφαρμογή στο Xcode.

6.2. Μελλοντικές επεκτάσεις

Η εφαρμογή αυτή είναι ακόμα σε πειραματικό στάδιο και φυσικά μπορεί να δεχτεί αρκετές αλλαγές ώστε να φτάσει σε ένα ικανοποιητικό επίπεδο για να μπορεί ο χρήστης να περιηγηθεί με άνεση σε όλη την Αθήνα ή και οποιαδήποτε άλλη περιοχή της Ελλάδας ή ακόμα και του εξωτερικού.

Σε πρώτο χρόνο θα μπορούσαν να ενσωματωθούν περισσότερα σημεία ενδιαφέροντος με περισσότερες πληροφορίες ή και ακόμα και ειδικά διαμορφωμένα 3D αντικείμενα ώστε χρησιμοποιώντας το AR Kit της Apple ο χρήστης να είναι σε θέση να βλέπει μπροστά του το επιλεγμένο σημείο ενδιαφέροντος όπου και αν βρίσκεται και φυσικά να μπορεί η εφαρμογή να αναγνωρίσει κάθε αντικείμενο με το Core ML Kit της Apple φωτογραφίζοντας το και να παρέχει πληροφορίες γενικού ενδιαφέροντος στον χρήστη.

Σε δεύτερο χρόνο, ο χρήστης μπορεί να έχει την δυνατότητα να στείλει δικές του φωτογραφίες από την κάθε τοποθεσία που επισκέπτεται ώστε να βλέπουν οι υπόλοιποι τι θα συναντήσουν στην κάθε τοποθεσία όταν την επισκεφτούν. Καθώς να αφήνει την δική του περιγραφή σαν σχόλιο με την προσωπική του άποψη.

Η παρούσα εφαρμογή θα μπορούσε να συνδεθεί ακόμη και με κεντρικές βάσεις δεδομένων ανά τον κόσμο και να επεκταθεί σε ένα παγκόσμιο ταξιδιωτικό οδηγό ή ακόμα και σε πλατφόρμα εκμάθησης με διαδραστικό τρόπο για σημεία ενδιαφέροντος, οπού ο χρήστης θα μπορεί να δει, να ακούσει, να διαβάσει για όποιο σημείο ενδιαφέροντος ή αξιοθέατο επιθυμεί.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Peterson, A C, Jr. Vehicle Radiotelephony Becomes a Bell System Practice. Bell Laboratories Record, 137, April, 1947.
[2] Roessner, D et al. The Role of NSF's Support of Engineering in Enabling Technological Innovation: Phase II, Chapter 4: The Cell Phone. Final report to the National Science Foundation. Arlington, Virginia: SRI International, 89, 1998. citing Ring, D H, "Mobile Telephony – Wide Area Coverage," Bell Laboratories Technical Memorandum, December 11, 1947. Online: http://www.sri.com/ policy/stp/techin2/chp4.html
[3] McDonald, R. Dial Direct: Automatic Radiotelephone System. IRE Transactions on Vehicle Communications, 80, July, 1958.
[4] Lewis, W D. Coordinated Broadband Mobile Telephone System. IRE Transactions, 43, May, 1960; and Schulte, H J Jr. and W A
Cornell. Multi-area Mobile Telephone System. IRE Transactions, 49, May, 1960.
[5] Douglas, V A. The MJ Mobile Radio Telephone System. Bell Laboratories Record, 383, December, 1964.
[6] Ikegami, F. Mobile Radio Communications in Japan. IEEE Transactions On Communications, 744, Vol. Com-20 No. 4, August, 1972.
[7] Geoff Fors. Personal correspondence.
[8] US Patent Number 3,906,166, granted September16, 1975.
[9] Ferranti, M. Father of Cell Phone Eyes a Revolution. IDG News Service, New York Bureau, 14 (31), October 12, 1999.
[10] Online: http://www.fcc.gov/Bureaus/OGC/Reports/cellr.txt
[11] Ito, Sadao and Yasushi Matsuzaka. 800 MHz Band Land Mobile Telephone System – Overall View. IEEE Transactions on Vehicular Technology, 205, Volume VT-27, No. 4, November 1978, as reprinted from Nippon Telegraph and Telephone's The Review of the Electrical Communication Laboratories, Vol. 25, nos 11–12, November-December, 1977. (English and Japanese)
[12] Gibson, Stephen W. Cellular Mobile Radiotelephones. Englewood Cliffs, Prentice Hall, 141, 1987. See also online:
http://www.privateline.com/PCS/Bahrain.htm
[13] Blecher, F. Advanced Mobile Phone Service. IEEE Transactions on Vehicle Communications, Vol. VT-29, No. 2, May, 1980.
[14] Meurling, John and Richard Jeans. The Ugly Duckling: Mobile phones from Ericsson. Stockholm, Ericsson Radio Systems AB, 46, 1997.
[15] Gibson, Stephen W. Cellular Mobile Radiotelephones. Englewood Cliffs, Prentice Hall, 19–22, 1987. 26) Online: http://ctia.org/research_statistics/index.cfm/AID/10030
[16] Brad Reed, 2010, ―A brief history of Smartphone's‖,
http://www.networkworld.com/slideshows/2010/061510-smartphone-history.html#slide1
[17] James Niccolai, Nancy Gohring, 2010,‖A Brief History of Palm‖,
http://www.pcworld.com/article/195199/article.html
[18] Wikipedia, 2012, ―Blackberry‖, http://en.wikipedia.org/wiki/BlackBerry
[19] Hamza Querashi, 2012, ―Apple: from iPhone 1 to iPhone 5 – Evolution, Features and Future Review‖ ,http://www.thenewstribe.com/2012/07/16/apple-from-iphone-1-to-iphone-5-evolution-features-and-future-review/
[20] Sam Costello,retrieved on 2012, ―First-Generation iPhone Review‖,

http://ipod.about.com/od/iphoneproductreviews/fr/iphone_review.htm
[21] Chris Ziegler, 2011, ―Android: A visual history‖,
http://www.theverge.com/2011/12/7/2585779/android-history
Impact of Smartphone‘s on Society 225
[22] Mark Prigg, 2012,‖ Microsoft launches Windows 8 Phone software - and hopes apps and Jessica Alba will help it take on Apple and Google‖, http://www.dailymail.co.uk/sciencetech/article-2225149/Windows-8-phone-software-launch-Microsoft-hopes-Jessica-Alba-help-Apple-Google.html

[23]
https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/index.html
[24] Hillyer, N. (2016). The Ultimate Guide to Choosing Objective-C or Swift for Your Project. Available at: https://savvyapps.com/blog/ultimate-guide-choosing-objective-c-or-swift. Cite 22.02.2017.
[25] Solt, P. (2015). Swift vs. Objective-C: 10 reasons the future favors Swift. Available at: http://www.infoworld.com/article/2920333/mobile-development/swift-vs-objective-c-10-reasons- the-future-favors-swift.html. Cite 22.02.2017.
https://en.wikipedia.org/wiki/Swift_(programming_language)#Version_History
[26] https://en.wikipedia.org/wiki/Firefox_for_iOS
[27] https://en.wikipedia.org/wiki/Gecko_(software)
[28] https://wordpress.org/mobile/
[29] https://en.wikipedia.org/wiki/Flappy_Bird
[30] https://github.com/lexrus/VPNOn
[31] https://github.com/CatchChat/Yep
[32] https://developer.apple.com/documentation/createml/creating_an_image_classifier_model
[33] https://hub.packtpub.com/what-is-core-ml/
[34] https://developer.apple.com/swift/blog/?id=37
[35] https://www.macrumors.com/roundup/apple-maps/
[36] https://en.wikipedia.org/wiki/Apple_Maps
[37]https://developer.apple.com/documentation/corelocation/getting_the_user_s_location
[38]https://developer.apple.com/documentation/corelocation
[39]https://developer.apple.com/documentation/corelocation/getting_the_user_s_location/using_the_significant-change_location_service
[40]https://developer.apple.com/documentation/corelocation/getting_the_user_s_location/using_the_standard_location_service
[41]https://developer.apple.com/documentation/corelocation/getting_the_user_s_location/using_the_visits_location_service
[42] https://el.wikipedia.org/wiki/Global_Positioning_System
[43] https://en.wikipedia.org/wiki/Mobile_phone_tracking