Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Data analytics σε real-time βάση δεδομένων στο Google Firebase Framework.** **Data analytics on a real-time database developed in Google Firebase Framework.**
Ονοματεπώνυμο Φοιτητή	**Γεώργιος Μαθιός**
Πατρώνυμο	**Κωνσταντίνος**
Αριθμός Μητρώου	**ΜΠΣΠ13061**
Επιβλέπων	**Κωνσταντίνος Πατσάκης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **30 Οκτωβρίου 2017**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Κωνσταντίνος Πατσάκης Επίκουρος Καθηγητής	Ευθύμιος Αλέπης Επίκουρος Καθηγητής	Γεώργιος Τσιχριντζής Πρόεδρος Τμήματος Πληροφορικής Πανεπιστημίου Πειραιώς

Περιεχόμενα

Περίληψη

Η διπλωματική εργασία που παρουσιάζεται παρακάτω δημιουργήθηκε και συντάχθηκε για την ολοκλήρωση της φοίτησης στο μεταπτυχιακό πρόγραμμα σπουδών «Προηγμένα Συστήματα Πληροφορικής» του Πανεπιστημίου Πειραιώς. Στις σελίδες που έπονται θα γίνει μία παρουσίαση ενός καινούργιου και πολύ σημαντικού κλάδου τα τελευταία χρόνια στην πληροφορική. Αυτός είναι τα Data Analytics αλλά και η χρησιμότητα τους για τους επιστήμονες σήμερα.

Η συλλογή τεράστιων βάσεων δεδομένων για πολλούς τομείς της καθημερινότητας, αλλά και η αλληλεπίδραση με αυτές είναι το κύριο ζήτημα που θα αναπτυχθεί. Η εργασία βασίζεται στην παραδοχή ότι κάποιος δημιούργησε μία real-time βάση δεδομένων με πολλά raw data. Αυτή η βάση παραδόθηκε στον ερευνητή με σκοπό εκείνος να χωρίς να πειράξει τα δεδομένα αυτά, να εκτελέσει data analytics (με τρόπο που εκείνος κρίνει) ώστε να γίνει μία εξαγωγή σειράς συμπερασμάτων. Στην προκειμένη περίπτωση η βάση αυτή δημιουργήθηκε συλλέγοντας δεδομένα μέσω μίας εφαρμογής που αναπτύχθηκε για κινητά τηλέφωνα και τάμπλετ.

Τα δεδομένα τα οποία συλλέχθηκαν και αποθηκεύτηκαν στην βάση δεδομένων προέρχονται από τα διάφορα αισθητήρια των κινητών συσκευών, αλλά και από την διεπαφή της εφαρμογής με το εκάστοτε επιβατικό όχημα ΙΧ του χρήστη. Η βάση δεδομένων που χρησιμοποιήθηκε αναπτύχθηκε στο framework της Google Firebase. Για να μπορέσει να οπτικοποιηθεί το αποτέλεσμα της συλλογής των δεδομένων δημιουργήθηκε ένας ιστότοπος. Αφού έγιναν οι απαραίτητες μετατροπές, εκτελέστηκαν μέσω πηγαίου κώδικα queries με σκοπό την εξαγωγή διάφορων συμπερασμάτων με βάση τα δεδομένα.

Με βάση τα παραπάνω αποδείχθηκε πως η χρήση τέτοιων εφαρμογών με σκοπό την συλλογή δεδομένων μπορεί να οδηγήσει μελλοντικούς ερευνητές σε μελέτες πάνω στον τομέα των Data Analytics που μπορούν να επηρεάσουν σε σημαντικό βαθμό τους διάφορους τομείς της τεχνολογίας ακόμα και σε βιομηχανικό επίπεδο.

Abstract

The diploma thesis presented below was created and compiled for the completion of the postgraduate program "Advanced Informatics Systems" of the University of Piraeus. In the following pages will be a presentation of a new and very important area of computer science in recent years. This is the sector of Data Analytics and their usefulness for scientists today.

The collection of huge databases for many areas of everyday life, as well as the interaction with them, is the main issue to be discussed. The thesis is based on the assumption that someone created a real-time database with many raw data. This database was then given to the researcher so as to perform data analytics – without manipulating the data itself (in any way he wishes or finds suitable), present the results and make a series of useful conclusions. In this case, this database was created by collecting data through an application that was developed for mobile phones and tablets.

The data collected and stored in the database come not only from various sensors of the mobile devices, but also from the interface of the application with the user's own passenger vehicle. The database used was constructed within the Google Firebase framework. In order to be able to visualize the result of the collection of data, a website was created. After the necessary conversions were made, the researcher executed queries (through source code) in order to perform Data Analytics that led him in extracting various conclusions for the database.

Based on the above, it has been shown that the use of such applications for the purpose of collecting data may lead future researchers to studies on the Data Analytics sector, which can significantly affect the various technology sectors even on industrial level.

Εισαγωγή

Μία βάση δεδομένων αποτελεί στην ουσία μία οργανωμένη συλλογή δεδομένων. Οι βάσεις δεδομένων μπορούν να αποθηκευτούν είτε τοπικά σε έναν προσωπικό υπολογιστή είτε μία απομακρυσμένη υπηρεσία cloud-storage. Μία real-time βάση δεδομένων έχει ως κύριο γνώρισμα ότι τα δεδομένα θα πρέπει να αποθηκεύονται αλλά και να γίνεται η ανάκτηση τους με πολύ μεγάλη ταχύτητα. Ο στόχος αυτής της εργασίας, είναι πέραν από το να γίνει μία αναλυτική παρουσίαση του Google Firebase API, να εκτελεστούν και Data Analytics για μία υπάρχουσα βάση δεδομένων. Αυτό θα γίνει με τη βοήθεια queries τα οποία κατασκευάστηκαν από τον ερευνητή. Μέσω αυτών, παράχθηκαν χρήσιμα συμπεράσματα σε σχέση με τα δεδομένα που έχει αποθηκευμένα η βάση στο εσωτερικό της.

Στο πρώτο κεφάλαιο, θα γίνει μία προσπάθεια να αποσαφηνιστούν οι τεχνικοί όροι των Data Analytics και των Big Data. Επίσης, θα γίνει και μία μικρή αναφορά στα θέματα ασφάλειας που έχουν οι σύγχρονες βάσεις δεδομένων στον τομέα της πληροφορικής.

Στο δεύτερο κεφάλαιο θα γίνει μία υπέρ αναλυτική παρουσίαση του Google Firebase Framework, όπου θα παρουσιαστούν όλες οι υπηρεσίες που αυτό προσφέρει στον σύγχρονο developer ώστε εκείνος να το χρησιμοποιήσει για την εκάστοτε εφαρμογή που εκείνος αναπτύσσει, Αλλά και ένας μικρός οδηγός για το πώς δημιουργείται ένα project και συνδέεται με τις αναπτυσσόμενες εφαρμογές. Στη συνέχεια και έχοντας χτίσει το απαραίτητο θεωρητικό υπόβαθρο θα γίνει μία μικρή παρουσίαση των raw data που παραδόθηκαν στον ερευνητή με σκοπό την περαιτέρω διερεύνηση τους. Στόχος για τον αναγνώστη θα είναι να κατανοήσει την μορφή αλλά και τη δομή της βάσης δεδομένων αλλά και να αντιληφθεί την προέλευση και τι σημασία της κάθε εγγραφής.

Στο τρίτο και τελευταίο κεφάλαιο, ξεκινώντας θα γίνει μία συνοπτική παρουσίαση για τον ιστότοπο τον οποίο κατασκεύασε ερευνητής (με σκοπό να οπτικοποιήσει καλύτερα για τον μέσω αναγνώστη) τα αποτελέσματα μετά από την εκτέλεση των queries. Στα δύο εναπομείναντα υποκεφάλαια, θα γίνει αρχικά επεξήγηση της λειτουργίας της υπηρεσίας για το Road Matching και δευτερευόντως του πηγαίου κώδικα που συντάχθηκε, με σκοπό την εκτέλεση τον queries στην βάση δεδομένων. Κλείνοντας θα παρουσιαστούν τα αποτελέσματα που επέστρεψαν τα queries τόσο σε επίπεδο raw data αλλά και μέσα από την απεικόνιση τους στον ιστότοπο.

Αντί επιλόγου, θα γίνει μία επισκόπηση της εργασίας αναφέροντας τα προβλήματα που παρουσιάστηκαν κατά την υλοποίηση, αλλά και μία μικρή ποιοτική ανάλυση, όσον αφορά τα αποτελέσματα των queries μαζί με προτάσεις για μελλοντικό έργο.

ΚΕΦΑΛΑΙΟ 1

1.1 Τι σημαίνει Data Analytics.

Τα Data Analytics είναι η διεργασία με την οποία εξετάζουμε διάφορα σετ δεδομένων με την βοήθεια εξειδικευμένων συστημάτων και λογισμικού. Ο σκοπός είναι να καταλήξουμε σε συμπεράσματα σε σχέση με την πληροφορία που περιέχουν τα δεδομένα αυτά. Η τεχνολογία των Data Analytics εφαρμόζεται σήμερα σε πάρα πολλές εμπορικές βιομηχανίες με σκοπό τη καλύτερη λήψη αποφάσεων που βασίζεται στην καλύτερη προγενέστερη πληροφόρηση. Επίσης χρησιμοποιείται από ερευνητές για την επιβεβαίωση ή την απόρριψη επιστημονικών μοντέλων, θεωριών και υποθέσεων.

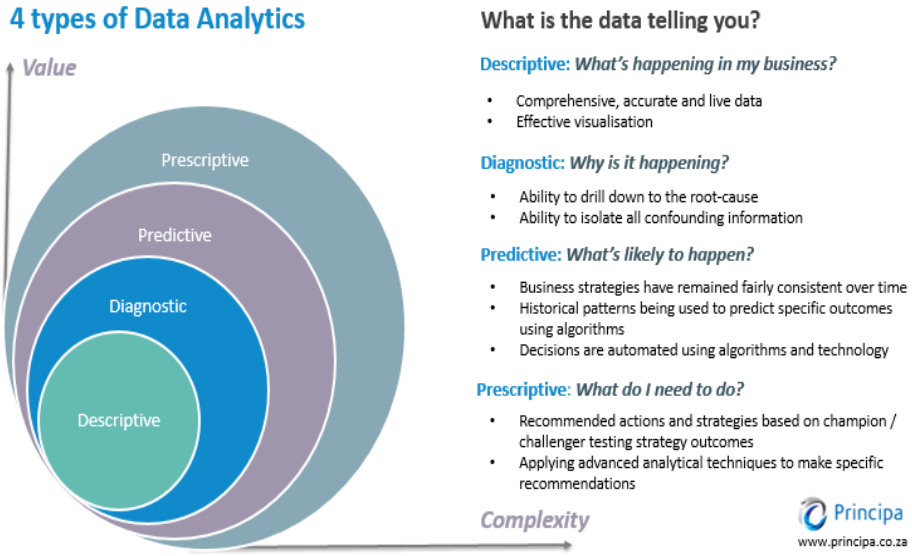
Σαν όρος τα Data Analytics έχουν τη ρίζα τους, κυρίως στον τομέα του business intelligence παράγοντας αναφορές και online analytical processing σε διάφορες μορφές μέσω προηγμένων analytics. Αυτό είναι και το μοντέλο το οποίο έρχεται συνήθως στον νου του καθενός όταν κάποιος αναφέρεται σε αυτά. Συνήθως οι οργανισμοί ή εταιρίες οι οποίες έχουν πάρει την πρωτοβουλία να ασχοληθούν με τον τομέα του Data Analytics, βοηθήθηκαν στο να αυξήσουν τα κέρδη τους, να βελτιώσουν την επιχειρησιακή τους λειτουργία, να κάνουν ακόμα καλύτερες τις διαφημιστικές τους καμπάνιες, να βελτιώσουν σε μεγάλο βαθμό την εξυπηρέτηση πελατών τους, να αντιδράσουν πιο γρήγορα σε ενδεχόμενες αλλαγές της αγοράς, ώστε να έχουνε στρατηγικό πλεονέκτημα σε σχέση με τους ανταγωνιστές τους. Ανάλογα πάντοτε με την φύση της εφαρμογής, τα δεδομένα τα οποία θα αναλυθούν, μπορούν να αποτελούνται είτε από ιστορικά δεδομένα, είτε από καινούργια δεδομένα τα οποία έχουν επεξεργαστεί για real-time analytics. Επιπρόσθετα, μπορούν να γίνουν και μελέτες από μία μίξη εσωτερικών συστημάτων και εξωτερικών πηγών δεδομένων.

Οι μεθοδολογίες των Data Analytics έχουν να κάνουν με την ανεύρεση ακολουθιών συμπεριφοράς και σχέσεων ανάμεσα στα εξεταζόμενα δεδομένα στα οποία μπορεί να εφαρμοστεί στατιστική ανάλυση και να αποδειχθεί αν μία υπόθεση εργασίας για ένα σετ δεδομένων είναι αληθής ή ψευδής. Θα μπορούσε να παρομοιάσει κανείς τη συγκεκριμένη δουλειά με τη δουλειά ενός ντετέκτιβ. Τα Data Analytics συνήθως χωρίζονται σε δύο κύριες κατηγορίες. Αυτές είναι η ποιοτική ανάλυση των δεδομένων η ποσοτική ανάλυση αυτών. Η πρώτη κατηγορία συνήθως περιλαμβάνει αριθμητικά δεδομένα με μετρήσιμες μεταβλητές οι οποίες μπορούν να συγκριθούν η να μετρηθούν μέσω στατιστικής. Η ποσοτική προσέγγιση είναι περισσότερο ερμηνευτικού περιεχομένου και επικεντρώνεται στην κατανόηση του περιεχομένου των δεδομένων (συνήθως μη αριθμητικών) όπως είναι για παράδειγμα κείμενα, εικόνες, αρχεία ήχου ή βίντεο, τα οποία συνήθως περιλαμβάνουν φράσεις, θέματα ,αλλά και σημεία αναφοράς.

Από την άλλη το κομμάτι που έχει να κάνει με το business intelligence, παρέχει στους διευθυντές και σε άλλα εργαζόμενα μέλη μιας εταιρείας, την δυνατότητα να έχουνε πρακτική πληροφορία σε σχέση με κρίσιμους στατιστικούς δείκτες απόδοσης όπως είναι οι λειτουργίες της εταιρείας, το πελατολόγιο κτλ. Αρκεί να ανατρέξουμε στο πρόσφατο παρελθόν για να συνειδητοποιήσουμε ότι τα data queries αλλά και οι αναφορές συνήθως δημιουργούνταν για τους τελικούς χρήστες από developers οι οποίοι εργάζονταν στο κομμάτι του IT. Οι σύγχρονοι οργανισμοί χρησιμοποιούν όλο και περισσότερο ατομικής χρήσης εργαλεία, τα οποία επιτρέπουν στους διευθυντές, στους ειδικούς αναλυτές και στο λοιπό προσωπικό να διεξάγουν τα δικά τους queries και να φτιάξουν από μόνοι τους δικές τους αναφορές.

Πιο προηγμένες τεχνικές Data Analytics περιλαμβάνουν το data mining, το οποίο συνήθως ασχολείται με το σοτάρισμα σε μεγάλα σετ δεδομένων, τα οποία τους επιτρέπουν να αναγνωρίζουν καινούργιες μόδες και ακολουθίες συμπεριφοράς ή σχεσιακά μοντέλα. Τα predictive analytics αναφέρονται στους προβλεπτικούς αλγόριθμους και συνήθως προσπαθούν να προβλέψουν την πιθανή συμπεριφορά ενός πελάτη, αστοχίες στους εξοπλισμούς και πιθανά μελλοντικά γεγονότα. Έτσι ακριβώς λειτουργεί το machine learning που είναι ένας κλάδος που εμπίπτει στο πεδίο της τεχνητής νοημοσύνης και χρησιμοποιεί αυτοματοποιημένους αλγορίθμους για να διαβάσει τα σετ δεδομένων πιο γρήγορα από όσο μπορούν οι ερευνητές που χρησιμοποιούν συμβατικά αναλυτικά μοντέλα. Τα Big Data analytics έχουν άμεση σχέση με το data mining, που αναφέρθηκε νωρίτερα και συνήθως εμπίπτουν σε σετ δεδομένων, τα οποία

έχουν μη δομημένα ή ημι-δομημένα δεδομένα σαν εγγραφές. Αντίστοιχα το text mining είναι ένα μέσο για ανάλυση αναγνώσιμων αρχείων, μνημάτων ηλεκτρονικού ταχυδρομείου , μνημάτων κινητής τηλεφωνίας κτλ. Σήμερα ένα μεγάλο μέρος των σύγχρονων επιχειρήσεων αναζητά, υποστήριξη στην υπηρεσία των Data Analytics (εικόνα 1).



Εικόνα 1: Οι τέσσερεις τύποι των Data Analytics.

Απλά παραδείγματα αποτελούν οι τραπεζικοί οργανισμοί οι οποίοι εφαρμόζουν παρακολουθήσεις ακολουθίας συμπεριφοράς στα συστήματα αναλήψεων των λογαριασμών και των χρεώσεων των πιστωτικών καρτών, για να αποτρέψουν πιθανές απάτες η κλοπή προσωπικών δεδομένων. Ακόμα ένα παράδειγμα αποτελούν οι εταιρείες ηλεκτρονικού εμπορίου, αλλά και οι πάροχοι υπηρεσιών μάρκετινγκ, οι οποίοι αναλύουν τα click-stream για να ταυτοποιήσουν ποιοι είναι οι επισκέπτες τους, τι αναζητούν από τα προϊόντα τους και τι είναι πιο πιθανόν να αγοράσουν, βασιζόμενοι στο ιστορικό πλοήγησης των εφαρμογών ή των ιστοτόπων τους. Αντιστοίχως οι πάροχοι κινητής τηλεφωνίας εξετάζουν την κίνηση αλλά και τον όγκο των δεδομέων των πελατών τους σε σχέση με την πρόγνωση του καιρού ή με τις μετακινήσεις του πληθυσμού, για να προβλέψουν τυχόν αστοχίες στην επικοινωνία λόγω συγκεντρώσης μεγάλου αριθμό χρηστών κάτω από την ίδια κυψέλη με αποτέλεσμα να υπάρξει bootleneck.

Οι διαδικασίες που χρησιμοποιούνται σε εφαρμογές Data Analytics περιλαμβάνουν πολύ περισσότερες διεργασίες από την απλή ανάλυση των δεδομένων. Ιδιαίτερα σε πολύ προηγμένα πρότζεκτ, συνήθως χρειάζεται πολύ μεγάλη προπαρασκευή πριν το τελικό αποτέλεσμα. Αυτό περιλαμβάνει την συλλογή, την ενσωμάτωση και το τεστάρισμα των αναλυτικών μοντέλων πάνω στα δεδομένα ώστε να είναι 100% σίγουρο, ότι αυτά παράγουν σωστά αποτελέσματα Σε αυτή τη διαδικασία συμμετέχουν τόσο οι αναλυτές όσο και οι μηχανικοί, οι οποίοι βοηθούν με τη σειρά τους τους πρώτους να ετοιμάσουν σωστά τα σετ δεδομένων για επεξεργασία.

Συνήθως, η διαδικασία ξεκινά με τη συλλογή των δεδομένων, τα οποία οι ερευνητές αναλύουν με σκοπό να αναγνωρίσουν την πληροφορία, για την οποία θέλουν να διεξάγουν την έρευνα. Αν χρειαστεί η συλλογή δεδομένων από διαφορετικά συστήματα, για να μπορέσει να γίνει η μίξη των δεδομένων, θα χρειαστεί να χρησιμοποιηθούν ρουτίνες ενσωμάτωσης και να μετατραπούν τα δεδομένα, σε μία κοινή μορφή και να φορτωθούν σε ένα σύστημα ανάλυσης όπως είναι για παράδειγμα ο Hadoop cluster ή η NoSQL.

Όταν τα δεδομένα φορτωθούν και είναι στη σωστή μορφή, επιλύονται τυχόν προβλήματα που έχουν να κάνουν με την ποιότητα και την ακρίβεια της ανάλυσης των εφαρμογών. Αυτή η διαδικασία περιλαμβάνει την διεξαγωγή του data profiling και του data cleansing, έτσι ώστε να εξαλειφθεί η πιθανότητα ότι τα δεδομένα να είναι ασυνεχή και ότι τυχόν πιθανά λάθη από διπλο-εγγραφές έχουν διαγραφεί. Αντίστοιχη δουλειά γίνεται στο data preparation με σκοπό το

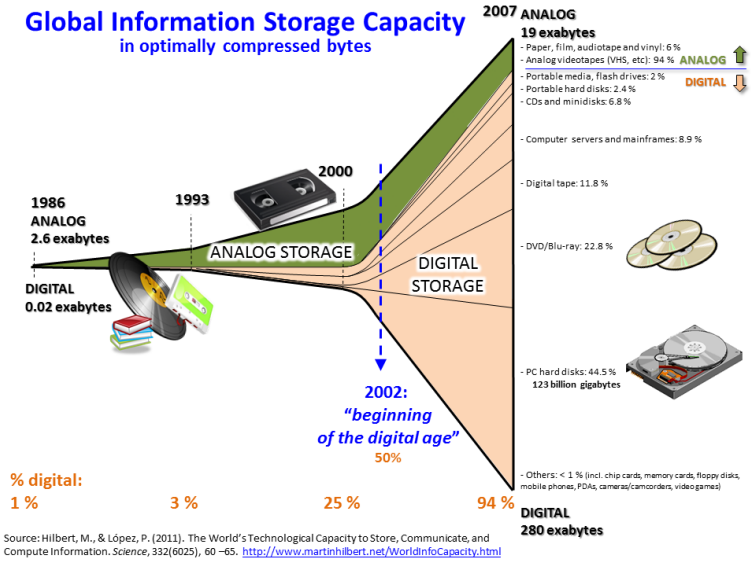
διαχείρηση και την οργάνωση των δεδομένων σύμφωνα με τις data governance πολιτικές που έχουν προεπιλεγεί να εφαρμοστούν.

Από αυτό το σημείο και μετά ξεκινάει η πραγματική διαδικασία, όπου ο ερευνητής καθορίζει το αναλυτικό μοντέλο το οποίο επιθυμεί, χρησιμοποιώντας εργαλεία προβλεπτικών μοντέλων με το αντίστοιχο λογισμικό και ανάλογες γλώσσες προγραμματισμού όπως Python, Scala, R και SQL. Τα μοντέλα αρχικά τρέχουν ενατίον ενός δοκιμαστικού dataset για να διαπιστωθεί η ακρίβεια. Αυτό συνήθως περιγράφεται ως προπόνηση του μοντέλου και συνεχίζεται μέχρι να αρχίσει να λειτουργεί αυτό στο επιθυμητό επίπεδο. Εν τέλει το μοντέλο μπαίνει σε λειτουργία εναντίον του φουλ σετ δεδομένων που εξετάζεται.

Σε κάποιες περιπτώσεις οι εφαρμογές analytics μπορούν να ενεργοποιήσουν αυτόματα δράσεις, όπως για παράδειγμα σε περίπτωση που θα χρειαζόταν να γίνουν πωλήσεις ή αγορές μετοχών σε ένα χρηματιστήριο. Σε αντίθετη περίπτωση το τελευταίο βήμα της διαδικασίας είναι η κοινοποίηση των αποτελεσμάτων που παρήχθησαν από τα αναλυτικά μοντέλα στους προισταμένους και στους διευθυντές ώστε αυτά να τους βοηθήσουν στη χάραξη της αντίστοιχης στρατηγικής. Αυτό συνήθως γίνεται με την βοήθεια του data visualization, το οποίο είναι μία διαδικασία που παράγει και δημιουργεί διαγράμματα και στατιστικές αναλύσεις, ώστε να είναι πιο εύκολο στην κατανόηση το τελικό αποτέλεσμα. Τα data vizualizations είναι συνήθως διαθέσιμα στο business intelligence dashboard της εφαρμογής και μπορούν να απεικονίσουν τα δεδομένα, σε μία και μόνο οθόνη αλλά και τις αλλαγές αυτών σε πραγματικό χρόνο, όσο καινούργιες πληροφορίες ρέουν συνεχώς προς το σύστημα. [1],[2],[3]

1.2 Big Data – Το επόμενο μεγάλο στοίχημα στις βάσεις δεδομένων.

Με τον όρο Big Data αναφερόμαστε σε σετ δεδομένων, τα οποία είναι τόσο μεγάλα σε μέγεθος και τόσο πολύπλοκα δομημένα, όπου τα παραδοσιακά λογισμικά διαχείρισης δεδομένων δεν είναι αδύνατον να τα διαχειριστούν. Οι σύγχρονες προκλήσεις της ενασχόλησης μετά Big Data περιλαμβάνουν τομείς όπως την συλλογή, αποθήκευση, ανάλυση, οπτικοποίηση των δεδομένων καθώς επίσης και τον διαμοιρασμό ή την εκτέλεση queries με σκοπό την περαιτέρω ανάλυση αυτών. Τελευταία ο όρος συνηθίζεται να χρησιμοποιείται για τους τομείς της πληροφορικής που έχουν να κάνουν με την ανάλυση συμπεριφοράς των χρηστών , τους προβλεπτικούς αλγορίθμους γεγονότων ή με τον κλάδο των Data Analytics, που μπορούν να εξάγουν υπεραξία για τον αναλυτή πάντα σε σχέση με το μέγεθος του σετ των δεδομένων (εικόνα 3).



Εικόνα 2: Εικόνα αύξησης ρυθμού αποθηκευμένων δεδομένων παγκοσμίως.

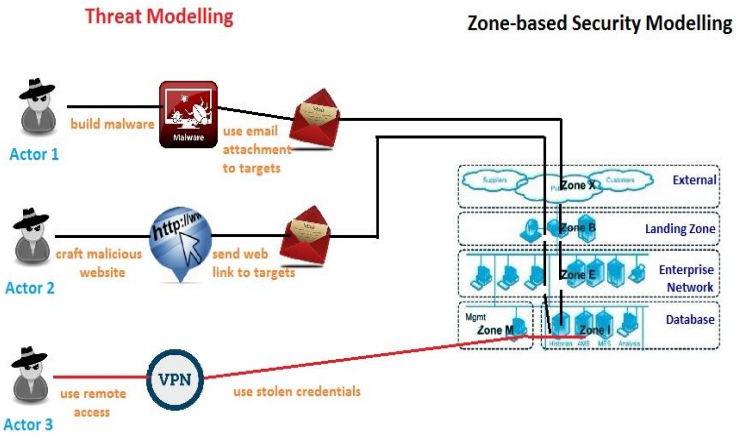
Η συλλογή και ανάλυση τέτοιον σετ δεδομένων γίνεται σχεδόν από όλους τους τομείς της κοινωνίας μας. Κυβερνήσεις, ιδιωτικοί και δημόσιοι οργανισμοί, αλλά και απλοί πολίτες αντιμετωπίζουν αντίστοιχα διάφορα προβλήματα σε όλους τους τομείς της καθημερινής τους δραστηριότητας, με την διαχείριση ενός τεράστιου όγκου δεδομένων, που έχει να κάνει από οικονομικές συναλλαγές, μέχρι μελέτες πάνω στην βιολογία και την περιβαλλοντική διαχείριση.

Με την παροχή γρήγορου ίντερνετ τα τελευταία χρόνια σε όλο τον σύγχρονο δυτικό κόσμο, τα σετ δεδομένων που συλλέγονται καθημερινά αυξάνονται με αλματώδη ρυθμό, κυρίως λόγω του ότι η συλλογή τους γίνεται πια από ιδιαίτερα φτηνές και καθημερινές συσκευές που όλοι χρησιμοποιούν. Μόνο με την χρήση των κινητών συσκευών σε όλο τον κόσμο (που έχουν σήμερα τα περισσότερα από αυτά έχουν δυνατότητες ενός μικρού υπολογιστή τσέπης) και τους ενσωματωμένους αισθητήρες που αυτά περιλαμβάνουν παράγονται περίπου 2,5 exabytes δεδομένων τον χρόνο. Τα κλασικά συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων συνήθως παρουσιάζουν δυσκολία στο να διαχειριστούν Big Data. Η διαχείριση μπορεί συχνά να απαιτεί λογισμικό το οποίο να λειτουργεί παράλληλα σε δεκάδες, εκατοντάδες, ή ακόμα και χιλιάδες σέρβερ. Για πολλούς οργανισμούς αυτή η διαχείριση εκατοντάδων gigabyte δεδομένων για πρώτη φορά μπορεί να τις οδηγήσει σε διαδικασία αναζήτησης διαφορετικού συστήματος διαχειριστής δεδομένων. Αλλά και σε εν γένει σε διαφοροποίηση της οπτικής σε θέματα στρατηγικής ανάπτυξης. Τα πέντε βασικά χαρακτηριστικά των Big Data συνοψίζονται παρακάτω:

- **Volume =** Είναι η ποσότητα των δεδομένων που παράγονται και αποθηκεύονται. Το μέγεθος των δεδομένων καθορίζει το αν αυτά θεωρούνται Big Data ή όχι.
- **Variety =** Είναι ουσιαστικά ο τύπος και η φύση των δεδομένων. Αυτό συνήθως βοηθάει τους ανθρώπους που τα αναλύουν να μπορέσουν να τα αντιμετωπίσουν με την ίδια αποτελεσματική διορατικότητα.
- **Velocity =** Είναι ουσιαστικά η ταχύτητα του ρυθμού παραγωγής των δεδομένων αλλά και η ικανότητα αυτά να επεξεργαστούν κατάλληλα ώστε να μπορέσουν να πιάσουν κάποια συγκεκριμένα στάνταρ.
- **Variability =** Η διακοπή της συνέχειας των δεδομένων σε ένα σετ δεδομένων μπορεί Να δημιουργήσει προβλήματα στην διαχείρισή του.
- **Veracity =** Η ποιότητα των δεδομένων που συλλέγονται μπορεί να διαφέρει σε τέτοιο βαθμό που αυτό μπορεί να δημιουργήσει προβλήματα στην ακριβή ανάλυση αυτών. [4]

1.3 Θέματα ασφαλείας στις σύγχρονες βάσεις δεδομένων.

Σε καθημερινή πια βάση χάκερς από όλο τον κόσμο, εξαπολύουν επιθέσεις για να κλέψουν απόρρητα ή ευαίσθητα δεδομένα από βάσεις δεδομένων οργανισμών. Οι βάσεις δεδομένων είναι ένας από τους μεγαλύτερους αποδέκτες επιθέσεων μέσω διαδικτύου σύμφωνα με την τελευταία μελέτη της Verisign το 2017(εικόνα 3).



Εικόνα 3: Μοντέλο απειλής τριών διαφορετικών επιθέσεων σε βάση δεδομένων.

Ο λόγος που οι βάσεις δεδομένων αποτελούν τόσο συχνό στόχο, είναι πολύ απλός. Ουσιαστικά αποτελούν την καρδιά οποιουδήποτε οργανισμού και συνήθως έχουν αποθηκευμένα στοιχεία πελατών ή άλλες εμπιστευτικές πληροφορίες οι οποίες στην πλειονότητα των περιπτώσεων, δεν προστατεύονται σωστά με αποτέλεσμα να είναι ευάλωτα.

Όταν κάποιος χάκερ ή άτομα που έχουν προσληφθεί για εσωτερική κατασκοπεία, αποκτήσουν πρόσβαση σε ευαίσθητα δεδομένα, μπορούν πολύ γρήγορα να προκαλέσουν ζημιά σε πολλά κομμάτια μιας επιχείρησης και των λειτουργιών αυτής. Αυτό συνήθως γίνεται με σκοπό το άμεσο κέρδος. Οι απειλές που έχουν ταυτοποιηθεί τα τελευταία δύο χρόνια είναι περίπου οι ίδιες και συνεχίζουν σαν μάστιγα να ταλαιπωρούν ακόμα και σήμερα τους εκάστοτε οργανισμούς. Ακολουθεί μία αλφαβητική λίστα με τα πιο συνηθισμένα ήδη επιθέσεων – απειλών:

- **Database injection attacks =** Δύο είναι οι κύριοι τύποι επιθέσεων με database injection. Ο πρώτος είναι SQL injections, οι οποίες έχουν στόχο τυποποιημένα συστήματα βάσεων δεδομένων. Ο δεύτερος είναι NoSQL injections, οι οποίες με τη σειρά τους έχουν στόχο Big Data πλατφόρμες, ασχέτως της παραδοχής, ότι τα συστήματα Big Data είναι συνήθως πολύ καλά ασφαλισμένα και πρακτικά αδιαπέραστα. Από την στιγμή που χρησιμοποιούν SQL τεχνολογία είναι πάντοτε πιθανόν να δεχτούν ιδίου τύπου επιθέσεις, καθώς ανήκουν στην ίδια γενική κλάση.
- **Excessive privileges =** Είναι το σύνηθες φαινόμενο όπου κάποιος εργαζόμενος μιας επιχείρησης, απέκτησε πρόσβαση σε μία βάση δεδομένων και με τη σειρά του ξεπέρασε την χρήση των δικαιωμάτων πρόσβασης που είχαν σκοπό την εργασία του. Για παράδειγμα, ένας υπάλληλος τραπέζης που ενώ η δουλειά του απαιτεί, την δυνατότητα να αλλάξει το όνομα του δικαιούχου σε ένα λογαριασμό ταμιευτηρίου, εκείνος εκμεταλλεύτηκε το κενό ασφαλείας και αύξησε το όριο καθημερινής ανάληψης ενός πελάτη γιατί είναι φίλοι. Αυτό συνήθως συμβαίνει, γιατί οι επιχειρήσεις αποτυγχάνουν να κάνουν συχνά ενημερώσεις για την κατάσταση των δικαιωμάτων των υπαλλήλων τους. Όπως για παράδειγμα όταν αυτοί αλλάζουν αντικείμενο εντός της εταιρείας λόγω εσωτερικής μετακίνησης ή απλά αποχωρούν από αυτή.
- **Exploitation of vulnerable databases =** Δυστυχώς οι περισσότεροι οργανισμοί παλεύουν με το πρόβλημα, ώστε παραμείνουν τα συστήματα ασφαλείας τους ενημερωμένα up-to-date, ακόμα και όταν οι εκάστοτε επιδιορθώσεις είναι διαθέσιμες. Γενικά οι περισσότεροι οργανισμοί χάνουν σημαντικό χρονικό διάστημα μέχρι να μπορέσουν να κάνουν ενημέρωση στις βάσεις δεδομένων τους και αυτό τις καθιστά άμεσα ευάλωτες. Συνήθως οι χάκερς γνωρίζουν το πώς να κάνουν exploit μία unpatched βάση δεδομένων που έχει ίσως ακόμα μέσα της περασμένες τις εργοστασιακές ρυθμίσεις. Το σύνηθες πρόβλημα είναι οι εντατικοί ρυθμοί δουλειάς που έχουν συνήθως οι διαχειριστές βάσεων δεδομένων, η πολυπλοκότητα και η χρονοβόρα διαδικασία του να τσεκαριστεί μία ενημέρωση επιδιόρθωσης και η δυσκολία, του να βρεθεί ένα παράθυρο χρόνου, για να μπορέσει να γίνει η κατάλληλη συντήρηση σε ένα κρίσιμο σύστημα της υποδομής.
- **Malware =** Η αιωνόβια απειλή, που συνήθως χρησιμοποιείται για υποκλοπή ευαίσθητων δεδομένων με θύματα συνήθως εξουσιοδοτημένους χρήστες, λόγω μολυσμένων συσκευών, που συνήθως χρησιμοποιούν στους χώρους εργασίας.
- **Storage media exposure =** Συνήθως, η υποδομή που χρησιμοποιείται για αποθήκευση των δεδομένων αλλά και για δυνατότητα redundancy (κάνοντας αντίγραφα ασφαλείας) είναι τις περισσότερες φορές τελείως απροστάτευτη από επιθέσεις. Ως αποτέλεσμα έχουν καταγραφεί τα τελευταία χρόνια αμέτρητες παραβιάσεις που συνήθως κατέληγαν στην κλοπή σκληρών δίσκων ή άλλων αποθηκευτικών μέσων που χρησιμοποιούνταν για το backup. Επιπρόσθετα η αποτυχία να παρακολουθούνται στενά οι ενέργειες του εκάστοτε διαχειριστή που έχει χαμηλών δικαιωμάτων πρόσβαση σε ευαίσθητη πληροφορία, μπορεί να μεγιστοποιήσει τον κίνδυνο έκθεσης εμπιστευτικών δεδομένων. Παίρνοντας τα απαραίτητα μέτρα για να προστατέψουμε τα αντίγραφα ασφαλείας, αλλά και παρακολουθώντας ακόμα και τους πιο υψηλά εξουσιοδοτημένους χρήστες, όχι μόνο

εφαρμόζουμε στην πράξη την καλύτερη δυνατή πρακτική αλλά και υπακούμε στους υποχρεωτικούς κανονισμούς ασφαλείας.

- **Unmanaged sensitive data =** Οι περισσότερες εταιρείες δυσκολεύονται στο να διατηρήσουν ένα ακριβές αρχείο με τις βάσεις δεδομένων και των κρίσιμων δεδομένων που αυτές περιλαμβάνουν. Ξεχασμένες βάσεις δεδομένων συνήθως περιέχουν ευαίσθητες πληροφορίες και μπορούν να παραμείνουν, χωρίς επίβλεψη από την ομάδα ασφαλείας. Αυτό έχει ως αποτέλεσμα τα ευαίσθητα δεδομένα να εκτεθούν σε απειλές αν δεν παρθούν τα κατάλληλα μέτρα ασφαλείας.

Ο ανθρώπινος παράγων περίπου το 30% όλων των συμβάντων περιλαμβάνει αμέλεια. Αυτό συνήθως συμβαίνει λόγω έλλειψης κατάρτισης ή εξειδίκευσης στο να εφαρμοστούν πρωτόκολλα ασφαλείας είναι επιβληθούν οι αντίστοιχες πολιτικές αυτών και να συντάσσονται συχνά αναφορές συμβάντων. Έτσι προτείνεται μία πολυεπίπεδη σειρά λύσεων ασφάλειας. Ο καλύτερος αμυντικός μηχανισμός είναι η εφαρμογή των προτεινόμενων πρακτικών αλλά και οι επανειλημμένοι εσωτερικοί έλεγχοι για να μπορούν να προστατευτούν οι βάσεις δεδομένων ενός οργανισμού.

Η συχνή αξιολόγηση οποιασδήποτε ευπάθειας των βάσεων δεδομένων και η αναγνώριση τυχόν αφύλακτων σημείων πρόσβασης καθώς και σωστή ταξινόμηση των ευαίσθητων δεδομένων είναι μονόδρομος. Το σωστό μανατζάρισμα όσο αναφορά τα δικαιώματα των χρηστών και διαγραφή των excessive privileges αλλά και των ανενεργών χρηστών είναι εξίσου αναγκαία. Με συχνή παρακολούθηση της δραστηριότητας της βάσης δεδομένων και εφαρμογή προφίλ χρήσεως σε πραγματικό χρόνο (για να διαπιστωθεί τυχόν διαρροή πληροφορίας, ή μη εξουσιοδοτημένες SQL και Big Data συναλλαγές καθώς επίσης Και αντίμετρα σε επιθέσεις πρωτοκόλλων ή συστήματος αποτελεί επίσης μία λύση.

Χρειάζεται ακόμα μπλοκάρισμα των μολυσμένων αιτημάτων από το διαδίκτυο και εφαρμογή αυτοματοποιημένου ελέγχου. Ακόμα να γίνει αρχειοθέτηση των δεδομένων που αποθηκεύονται απομακρυσμένα σημεία και κρυπτογράφηση αυτών. Με εκπαίδευση των υπαλλήλων σε τεχνικές μετριασμού των κινδύνων και της αναγνώρισης των πιο συνηθισμένων κυβερνο-απειλών (όπως για παράδειγμα η επίθεση spear-phishing) και συνάμα με συχνή ενημέρωση όσον αφορά τις προτεινόμενες πρακτικές σχετικά με το Ιντερνέτ, την σωστή χρήση των ηλεκτρονικών ταχυδρομείων αλλά και τη διαχείριση των προσωπικών κωδικών μπορούμε να φτάσουμε σε ένα ικανοποιητικό επίπεδο ασφάλειας. [5]

ΚΕΦΑΛΑΙΟ 2

2.1 Παρουσίαση του Google Firebase Framework.

2.1.1 Εισαγωγή.

Σε αυτό το κομμάτι της εργασίας θα γίνει μία συνολική αναφορά για το Google Firebase Framework. Η startup εταιρεία Envolve (την οποία δημιούργησαν ο James Tamplin και ο Andrew Lee το 2011) κατασκεύασε ένα API (Application Programming Interface) το οποίο έδινε την δυνατότητα στους developers να ενεργοποιήσουν την υπηρεσία του online chat στους ιστότοπους τους. Λίγο καιρό μετά έγινε γνωστό ότι τελικά η υπηρεσία χρησιμοποιείτο από τους developers ώστε να γίνει συγχρονισμός δεδομένων σε πραγματικό χρόνο μεταξύ των χρηστών. Αυτό ώθησε τους δημιουργούς να εν τέλη να διαχωρίσουν το σύστημα του chat messaging με την υπηρεσία αρχιτεκτονικής πραγματικού χρόνου που είχαν δημιουργήσει και να ιδρύσουν την Firebase ως ξεχωριστή εταιρία. Μετά από δύο συνεχόμενους εράνους με επιλογή αγοράς μετοχών (το 2012 και το 2013) η καινούργια εταιρία τελικά εξαγοράζεται το 2014 από την Google η οποία την συγχωνεύει με την επίσης εξαγορασμένη εταιρία Divshot με σκοπό την δημιουργία μίας ενοποιημένης πλατφόρμας με πολλές υπηρεσίες για τους developers εφαρμογών κινητών συσκευών (εικόνα 4). [6]

Εικόνα 4: Το logo της Google Firebase.

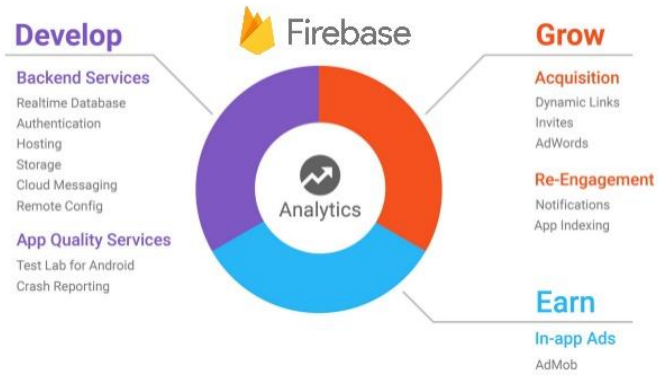
Η Firebase πια, είναι στην ουσία ένα API το οποίο παρέχει η Google με σκοπό την αποθήκευση βάσεων δεδομένων και έχει την δυνατότητα συγχρονισμού με εφαρμογές για διαδικτύου, Android ή iOS. Συνήθως χρησιμοποιείται σαν backend από developers για εφαρμογές σε Android OS. Έχει την δυνατότητα να λειτουργεί σαν real-time βάση δεδομένων που την κάνει πιο δυνατή σε σχέση με τις παραδοσιακές λύσεις (SQL, SQLite, shared preference κτλ.). Πρακτικά αυτό της δίνει την δυνατότητα να μπορεί να αποθηκεύει, αλλά και να τραβάει τιμές σε πραγματικό χρόνο, όταν και όποτε αυτό χρειαστεί ή ζητηθεί από την εκάστοτε εφαρμογή. Από τα βασικά πλεονεκτήματα που προσφέρει σε όσους την χρησιμοποιούν είναι η δυνατότητα ενσωμάτωσης της στην εφαρμογή με μόνο λίγες γραμμές κώδικα. Επίσης για να μπορέσει να είναι λειτουργική η εφαρμογή που θα δημιουργηθεί, θα πρέπει στο σετάρισμα της βάσης, να ακολουθηθούν όλα τα βήματα ενεργοποίησης με αποτέλεσμα να εξαλείφεται η πιθανότητα λάθους. Τα δεδομένα αποθηκεύονται σε μορφή JSON (Javascript Object Notation) και όπως προαναφέραμε είναι προσβάσιμα και διαθέσιμα από όλους τους τύπους πλατφόρμας.

Η Firebase υποστηρίζει μία σειρά από υπηρεσίες όπως αποθήκευση (storage), φιλοξενία (hosting), μνημάτων κτλ. στα οποία θα αναφερθούμε εκτενέστερα παρακάτω. Η υπηρεσία μισθώνεται με χρέωση στην Google και τα πρώτα 200 MB αποθηκευτικού χώρου δίνονται δωρεάν από την εταιρεία στον χρήστη. Η Google βελτιώνει συνεχώς αλλά και προσθέτει συνέχεια καινούργιες λειτουργίες στις υπηρεσίες της Firebase, όπως για παράδειγμα το AdMob (API για δημιουργία διαφημιστικής πλατφόρμας) που είναι ακόμα σε δοκιμαστική έκδοση (beta version). [7],[8]

2.1.2 Τα κυριότερα χαρακτηριστικά της Firebase.

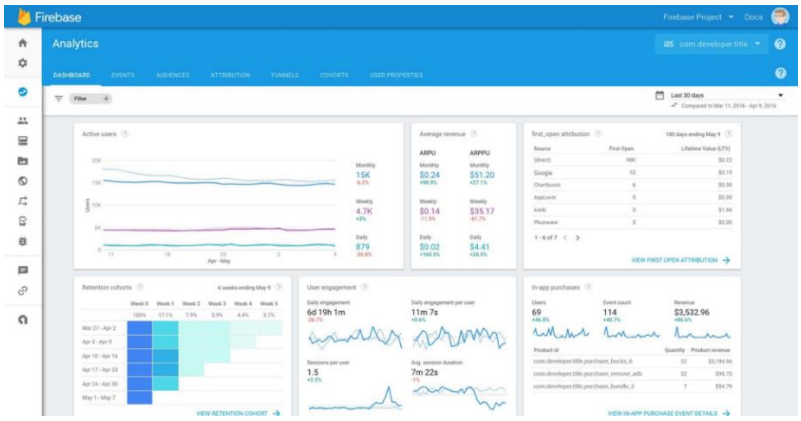
Όπως αναφέρθηκε και νωρίτερα η Firebase προσφέρει μία σειρά από έτοιμες υπηρεσίες οι οποίες δίνουν διάφορες δυνατότητες στους developers, μόλις εκείνοι τις συνδέσουν με την εκάστοτε εφαρμογή που εκείνοι αναπτύσσουν. Ακολουθεί μία συνοπτική παρουσίαση αυτών με αλφαβητική σειρά:

- **AdMob =** Πρόκειται για μία υπηρεσία που δημιουργεί μία διαφημιστική πλατφόρμα με σκοπό την παραγωγή κέρδους για τον δημιουργό, από την χορηγία μέσω διαφημιστικού υλικού. Η ταυτόχρονη χρήση μαζί με την υπηρεσία των Analytics δίνει την δυνατότητα για ανάλυση της συμπεριφοράς των χρηστών πάνω στην εφαρμογή (εικόνα 5). [9]



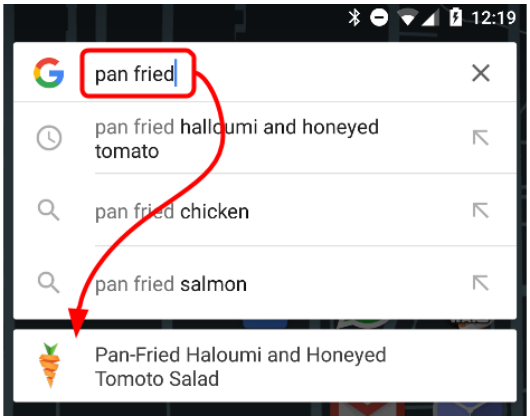
Εικόνα 5: Επεξήγηση διαγράμματος υπηρεσιών.

- **Analytics =** Αυτό το εργαλείο βοηθά σημαντικά τον developer να αντιληφθεί την συμπεριφορά των χρηστών όσο αναφορά την αλληλεπίδραση τους με την εφαρμογή. Το SDK έχει την δυνατότητα να καταγράψει δράσεις και γεγονότα από μόνο του με αποτέλεσμα να είναι δυνατόν να παραχθούν προσποιημένα δεδομένα. Ακόμα το dashboard δίνει πληροφορίες για τους πιο ενεργούς χρήστες, ποια χαρακτηριστικά της εφαρμογής χρησιμοποιούνται πιο συχνά, δημογραφικά στοιχεία και γενικά μία σειρά από συγκεντρωτικά στατιστικά. Με αυτό το εργαλείο ένας δημιουργός μπορεί να πάρει στρατηγικές αποφάσεις όσο αναφορά το μάρκετινγκ με σκοπό να προσελκύσει πελάτες μεγαλύτερης αξίας. Σε περίπτωση που ο developer το επιθυμεί μπορεί να ενώσει τα δικά του Analytics με την υπηρεσία BigQuery που του επιτρέπει πολύ μεγαλύτερη πολυπλοκότητα στην ανάλυση και προτείνεται συνήθως για μεγάλα σετ δεδομένων και από πολλές και διαφορετικές πηγές (εικόνα 6). [10]



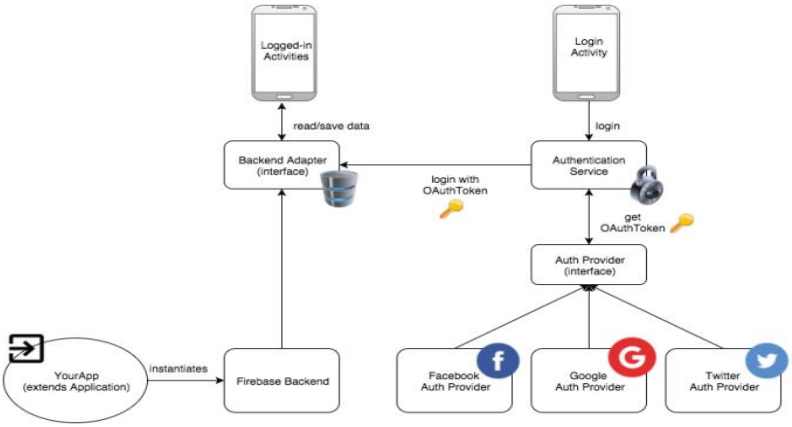
Εικόνα 6: Παράδειγμα εικόνας στο dashboard για τα Analytcis.

- **App Indexing =** Χρησιμοποιώντας αυτό το API σε Android OS ενισχύεται η αποδοτικότητα της βελτίωσης της βαθμολογίας για τους συνδέσμους που εμφανίζονται στην εφαρμογή και προσφέρεται δυνατότητα αυτοσυμπλήρωσης – πρότασης (σαν να ψάχνει κάποιος κάτι από την αναζήτηση του Google στο διαδίκτυο) βασισμένη στις προηγούμενες αναζητήσεις του χρήστη που έχουν καταγραφεί στα αρχεία καταγραφής. Πρακτικά η υπηρεσία αναζήτησης του Google σκανάρει από την χρήση των συνδέσμων τόσο στο διαδίκτυο όσο και στην εφαρμογή και τα αποτελέσματα τα σερβίρει όταν αυτά ζητηθούν. Αυτό επίσης, μπορεί να προστεθεί και σαν API δημιουργίας δείκτη προσωπικού περιεχομένου που δένεται με την συσκευή και τον προσωπικό λογαριασμό του χρήστη. Για παράδειγμα αρκεί να σκεφτεί κανείς τις σημειώσεις που μπορεί να κρατάει ένας χρήστης όσο αναφορά την ανάγνωση ενός εγγράφου την ώρα που χρησιμοποιεί την εφαρμογή (εικόνα 7). [11]



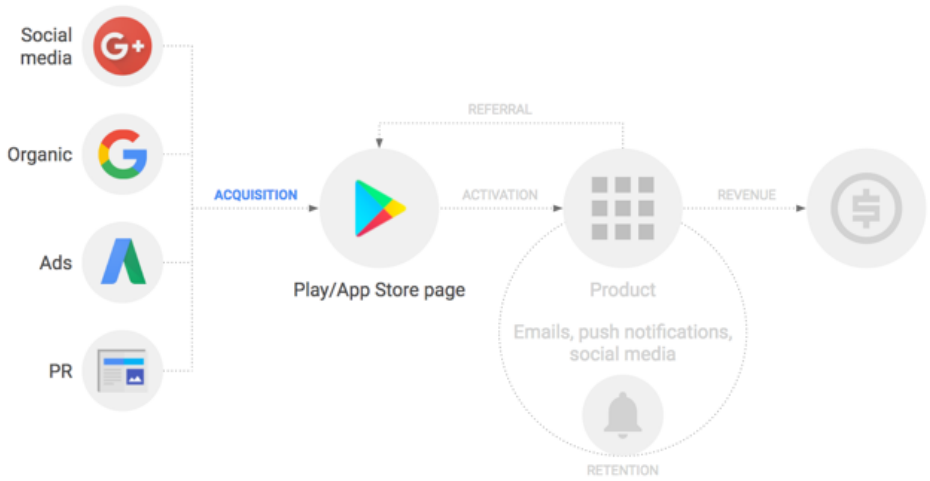
Εικόνα 7: Παράδειγμα εμφάνισης αυτόματης συμπλήρωσης.

- **Authentication =** Οι περισσότερες εφαρμογές σήμερα συνήθως ζητούν από τον χρήστη να προχωρήσει σε εγγραφή για να μπορέσουν να σώσουν με ασφάλεια τα δεδομένα της δραστηριότητας του και να προσφέρουν συγχρονισμό και προσωποποιημένη cross-platform εμπειρία ανάμεσα σε διαφορετικές συσκευές. Δύναται λοιπόν δυνατότητα μέσω της backend υπηρεσίας (με ένα εύχρηστο SDK) και με έτοιμες UI βιβλιοθήκες, ώστε να γίνει ταυτοποίηση του χρήστη. Αυτό μπορεί να γίνει, είτε με χρήση τηλεφωνικού αριθμού, είτε μέσω προσωπικού κωδικού, είτε με την συνηθισμένη ποια ταυτοποίηση μέσω social media όπως το Google (Gmail), Facebook, Twitter κτλ. Αξίζει να σημειωθεί πως υπάρχει πλήρης συμβατότητα με τα πρότυπα που υπάρχουν ήδη στην αγορά όπως το OpenID Connect και το OAuth 2.0 (εικόνα 8). [12]



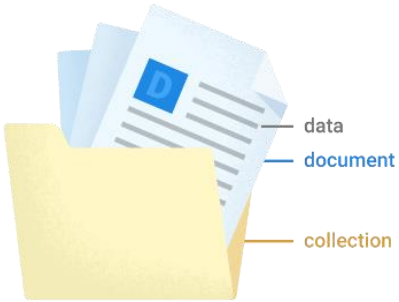
Εικόνα 8: Παράδειγμα ταυτοποίησης χρήστη με χρήση social media accounts.

- **AdWords =** Με το δέσιμο της υπηρεσίας AdWords με την Firebase, μπορεί να δημιουργηθεί και μία σειρά από λίστες βασιζόμενες σε πελατειακό κοινό που αντλείται από τα Analytics. Συνήθως το κοινό αυτό από προεπιλογή, είναι είτε από χρήστες που έχουν αγοράσει την εφαρμογή ή έχουν κάνει κάποια αγορά μέσα από την εφαρμογή, είτε από απλούς χρήστες που απλά εγκατέστησαν την εφαρμογή. Στην προκείμενη περίπτωση δίνεται η δυνατότητα να δημιουργηθούν λίστες με συνδυασμούς που μπορεί να βασίζονται σε άλλες δράσεις που εκτελούν οι χρήστες, τις επιλογές που έχουν στα properties της εφαρμογής. Έχει άμεση σχέση και λειτουργεί παράλληλα με το AdMob που αναφέρθηκε νωρίτερα (εικόνα 9). [13]



Εικόνα 9: Σχεδιάγραμμα απεικόνισης λειτουργίας με χρήση AdWords .

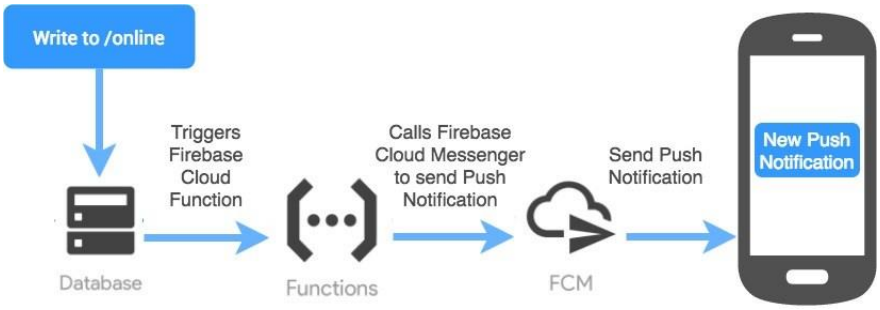
- **Cloud Firestore =** Πρόκειται για υπηρεσία που επιτρέπει την δημιουργία μίας cloud-hosted NoSQL βάσης δεδομένων που μπορούν να έχουν πρόσβαση όλων των ειδών οι εφαρμογές (Android, iOS και web) απευθείας μέσω των native SDKs. Η αποθήκευση γίνεται μέσα σε έγγραφα τα οποία περιέχουν πεδία που χαρτογραφούν τις τιμές. Τα έγγραφα με την σειρά τους αποθηκεύονται σε συλλογές που βοηθούν στην οργάνωση και των δεδομένων και στην δημιουργία queries (εικόνα 10).



Εικόνα 10: Collection template σε Firestore.

Υποστηρίζει πολλά διαφορετικά data types από απλά strings και αριθμούς μέχρι πολύπλοκα nested αντικείμενα. Όπως και στην Realtime Database είναι δυνατός ο cross-platform συγχρονισμός μέσω τοποθέτησης Realtime listeners και υπάρχει δυνατότητα offline υποστήριξης τόσο για mobile ή web. Αυτό διευκολύνει τις περιπτώσεις που υπάρχει network latency ή πρόβλημα connectivity. Προσφέρεται για ανάπτυξη και σε Node.js, Java, Python και GoSDKs παράλληλα με τα REST και RPC APIs. [14]

- **Cloud Functions =** Τα Cloud Function είναι η δυνατότητα να τρέξει κάποιος πηγαίο κώδικα στο backend κομμάτι όταν υπάρχει event που ενεργοποιείται από κάποιο feature ή λόγω κάποιου HTTP request. Ο κώδικας είναι αποθηκευμένος στο Google Cloud, εκτελείται σε ελεγχόμενο περιβάλλον και δεν χρειάζεται τσεκάρονται οι δυνατότητες του εκάστοτε server. Όταν το φορτίο γίνεται υψηλό η Google αυτόματα αυξάνει τον αριθμό τον Virtual Machines (instances) που χρειάζονται ώστε να μπορεί να εκτελεστεί άμεσα η function. Ακόμα και αν ο developer αλλάξει ή ανανεώσει μέρος του κώδικα που εκτελεί τα VM καθαρίζονται αυτόματα και αντικαθίστανται με νέα που εκτελούν ξανά την διαδικασία (εικόνα 11). [15]



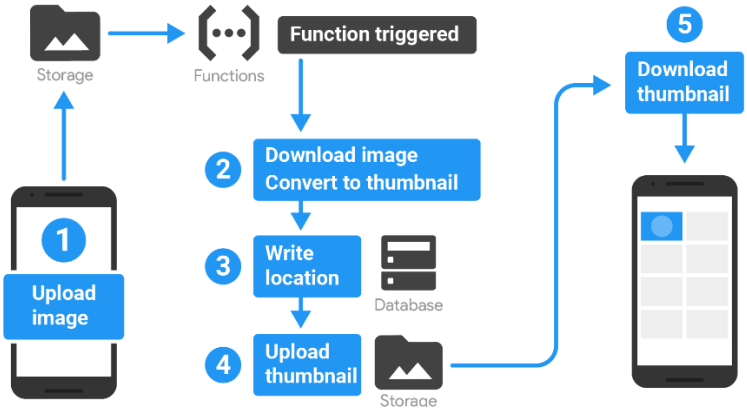
Εικόνα 11: Παράδειγμα χρήσης Cloud Functions για Push Notification.

- **Cloud Messaging =** Χρησιμοποιείται για την αποστολή ειδοποιήσεων στην client εφαρμογή. Σκοπός είναι να ειδοποιηθεί ο χρήστης για τον συγχρονισμό των δεδομένων του ή για την λήψη ενός μνήματος ηλεκτρονικού ταχυδρομείου πχ με τους όρους χρήσης. Έχει δύο βασικά στοιχεία λειτουργίας για λήψη και αποστολή. Πρώτον την ύπαρξη ενός εμπιστεύσιμου περιβάλλοντος όπου θα συντάσσονται και θα αποστέλλονται τα μνήματα (πχ ένας app σέρβερ). Δεύτερον την ύπαρξη μίας function στην client εφαρμογή που θα λαμβάνει αυτά τα μηνύματα (συνήθως JavaScript αν είναι web). Το μοντέλο φαίνεται στην παρακάτω εικόνα 12.

Εικόνα 12: Αρχιτεκτονική Messaging.

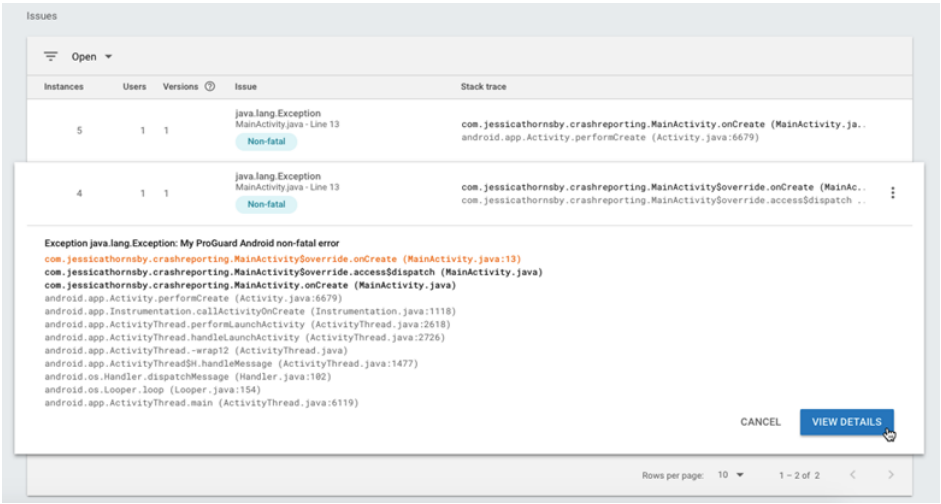
Σε εφαρμογές που χρησιμοποιούν την ιδιότητα του instant messaging το εκάστοτε μήνυμα μπορεί να μεταφερθεί με μέγιστο payload 4 KB στην κάθε μία client εφαρμογή χωρίς να υπάρξει επιπλέον χρέωση (εικόνα). [16]

- **Cloud Storage =** Αποτελεί έναν απλό, πανίσχυρο και χαμηλού κόστους τρόπο για την αποθήκευση αντικειμενοστερεφές περιεχομένου. Με την χρησιμοποίηση της υπηρεσίας είναι δυνατή η προσθήκη ασφάλισης στο περιεχόμενο με αποτέλεσμα να μην είναι δυνατή η προβολή του από μη εξουσιοδοτημένους χρήστες. Επίσης παρέχεται η ευκολία να γίνεται upload ή download του περιεχομένου στις κινητές συσκευές αλλά και η server-side επεξεργασία αυτού (πχ φιλτράρισμα εικόνων, κωδικοποίηση video κτλ.). Συνήθως το περιεχόμενο είναι φωτογραφίες, βίντεο ή υλικό παραχθέν από τον χρήστη. Σε περίπτωση δε που η σύνδεση είναι κακή δίνεται η δυνατότητα επανάληψης της διαδικασίας από το σημείο που αυτή σταμάτησε με ώστε να σώνεται πολύτιμος χρόνος και data ή bandwidth usage από τους χρήστες (εικόνα 13). [17]



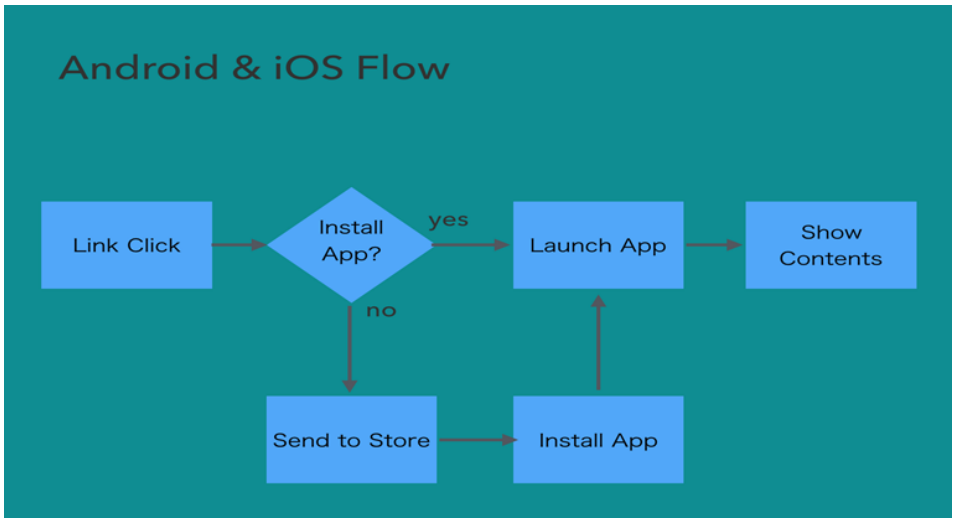
Εικόνα 13: Παράδειγμα μεταφοράς εικόνας μέσω του Storage.

- **Crash Reporting =** Το crash reporting είναι μία υπηρεσία η οποία δημιουργεί ολοκληρωμένες αναφορές για τα λάθη τα οποία συμβαίνουν κατά τη διάρκεια της χρήσης της εφαρμογής. Τα σφάλματα γκρουπάρονται σε προβλήματα τα οποία έχουν συνήθως την ίδια συμπεριφορά και το βασικό αναγνωριστικό τους είναι το πρόβλημα το οποίο προκαλούν στους τελικούς χρήστες της εφαρμογής. Επιπρόσθετα υπάρχει η δυνατότητα να γίνουν αυτόματα αναφορές συμβάντων τα οποία μέσω των αρχείων καταγραφής τους μπορούν να βρούμε επακριβώς το αρχικό πρόβλημα το οποίο δημιούργησε την πτώση της εφαρμογής και να προχωρήσει το ανάλογο debugging (εικόνα 14). [18]



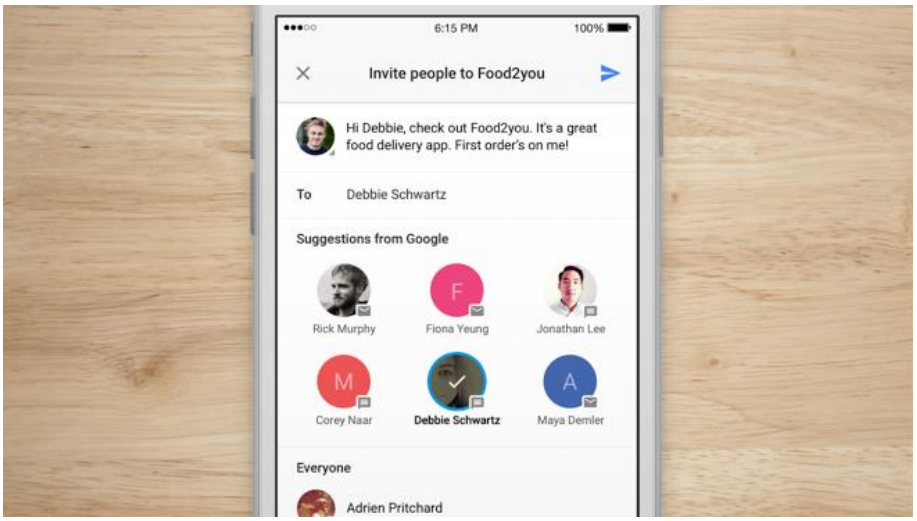
Εικόνα 14: Παράδειγμα Crash Repoting.

- **Dynamic Links =** Με την δημιουργία ενός δυναμικού συνδέσμου είτε μέσω της κονσόλας της firebase, είτε χρησιμοποιώντας ένα REST API (ισχύει το ίδιο και σε iOS, android ή web) μπορούμε να έχουμε συγκεκριμένες παραμέτρους συμπεριφοράς της εφαρμογής. Με την υποστήριξη του cross-platform ο χρήστης ασχέτως της της κινητής συσκευής θα μπορεί να ανοίξει το σύνδεσμο μέσω της δικής μας εφαρμογής. Σε περίπτωση που ο χρήστης δεν έχει εγκαταστήσει την εφαρμογή θα γίνεται redirect στο google store με αποτέλεσμα να ζητηθεί πρώτα από να κάνει εγκατάσταση την εφαρμογή και στη συνέχεια αφού συμβεί αυτό να μπορέσει να ξανανοίξει τον σύνδεσμο πάλι μέσω της αυτής (εικόνα 15). [19]



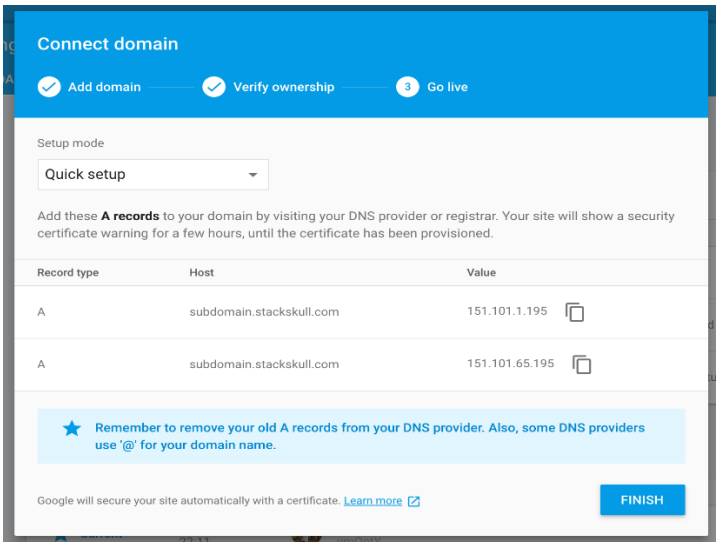
Εικόνα 15: Διάγραμμα ροής διαδικασίας ανοίγματος Dynamic Link.

- **Invites =** Άλλο ένα εργαλείο προώθησης και μάρκετινγκ της εφαρμογής μπορούν να είναι οι προσκλήσεις που μπορεί να αποστείλει ένας ήδη υπάρχον χρήστης. Σαφέστατα η διαφήμιση από στόμα σε στόμα, αποτελεί ένα όπλο με το οποίο ο developer μπορεί να κάνει την εφαρμογή του ιδιαίτερα εμπορική. Σε συνέχεια μετά Dynamic links που αναφέραμε προηγουμένως ο χρήστης αποστέλλει σε όλες τις επαφές που εκείνος θα επιλέξει, το Dynamic link το οποίο όταν ο παραλήπτης ανοίξει, λειτουργεί με την ίδια διαδικασία που περιεγράφηκε νωρίτερα (εικόνα 16). [20]



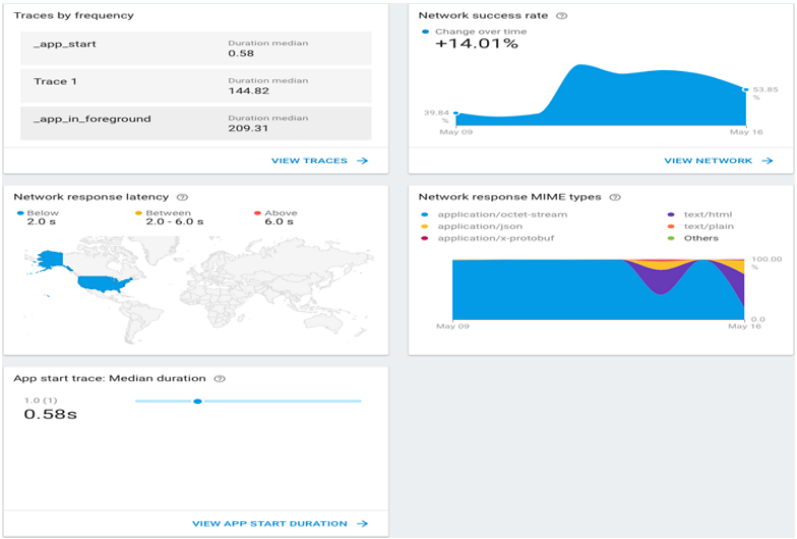
Εικόνα 16: Παράδειγμα Invites.

- **Hosting =** Με την αύξηση της χρήσης front-end JavaScript frameworks όπως το Angular ή static generator tools όπως το Jekyll η χρήση static sites είναι πιο σημαντική από ποτέ. Ασχέτως με την πολυπλοκότητα της εφαρμογής το hosting μας παρέχει την κατάλληλη υποδομή για να μπορέσουμε να κάνουμε upload αρχεία ακόμα και από τα local directories του προσωπικού μας υπολογιστή απευθείας στον hosting σέρβερ. Τα αρχεία σερβίρονται πάντα από τον κοντινότερο σέρβερ μέσω μιας ασφαλούς SSL σύνδεσης με κλειδί το οποίο παράγεται αυτόματα για το εκάστοτε domain. Επίσης, προσφέρεται μία πόλη ελαφριά εργαλειοθήκη επιλογών, με την οποία είναι πανεύκολο να ξαναγραφτούν ακόμα και τα URL για client-side δρομολόγηση ή να σεταριστούν προσωποποιημένες επικεφαλίδες (εικόνα 17).[21]



Εικόνα 17: Μενού setup για hosting.

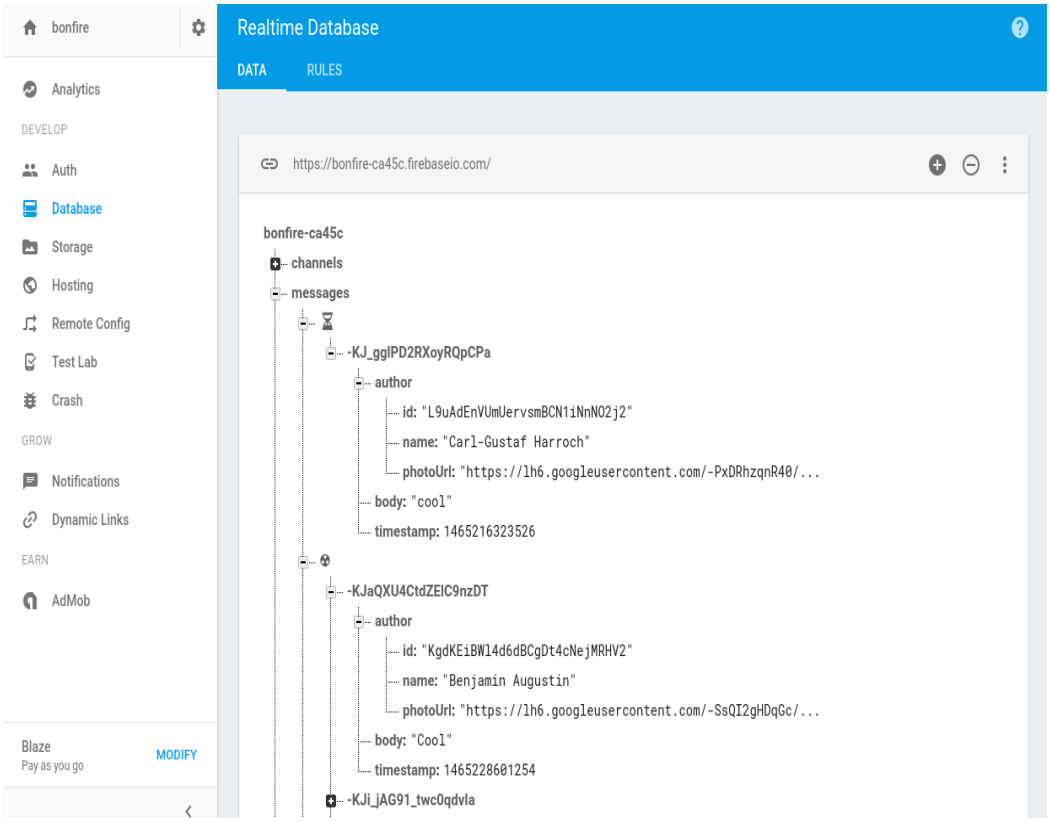
- **Performance Monitoring =** Αποτελεί το βασικό εργαλείο με το οποίο μπορούμε να κάνουμε μία ανάλυση της συμπεριφοράς της εφαρμογής. Αναλύοντας τα δεδομένα (μέσω trace) για όλα τα https requests σε σχέση με την εφαρμογή, παράγεται μία αναφορά με χρονικό προσδιορισμό ανάμεσα σε δύο χρονικές περιόδους (εικόνα 18).



Εικόνα 18: Εικόνα από Performance Monitoring εφαρμογής.

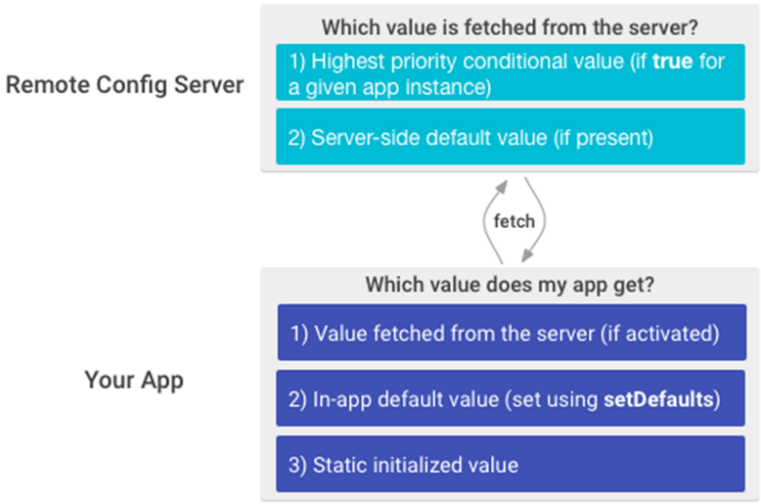
Υπάρχει δυνατότητα να δημιουργηθούν πέραν των προεπιλεγμένων, προσωποποιημένα traces και αντίστοιχα το κάθε trace, να μπορεί να έχει και περαιτέρω επιλογές με σκοπό να καταγραφούν counters και για events. Τα βασικά metrics τα οποία απασχολούν τον developer είναι ο χρόνος αντίδρασης (δηλαδή ο χρόνος που πέρασε μεταξύ ενός request που έγινε και του σημείου το οποίο αυτό απαντήθηκε). Το δεύτερο είναι το μέγεθος του payload, το οποίο δείχνει το συνολικό φόρτου του δικτύου, το οποίο έχει χρησιμοποιήσει εφαρμογή και τέλος, το τρίτο είναι το ποσοστό επιτυχίας, το οποίο δείχνει πόσο επί τοις εκατό είχαμε επιτύχει responses ώστε να γίνει καταγραφή τυχόν αποτυχίας στο δίκτυο ή στον server. [22]

- **Realtime Database =** Η Realtime Database είναι μία NoSQL Cloud-Hosted βάση δεδομένων και ως επακόλουθο έχει διαφορετικά χαρακτηριστικά και συνθήκες λειτουργίας όταν την συγκρίνουμε με σχεσιακή βάση δεδομένων. Η expression-based γλώσσα που χρησιμοποιείται η οποία επιτρέπει να προκαθοριστεί ο τρόπος με τον οποίον θα κατασκευάζονται τα data αλλά και ποτέ ή πως αυτά θα μπορούν να διαβαστούν ή να εγγραφούν. Το API είναι σχεδιασμένο με κύριο στόχο να εκτελούνται πολύ γρήγορα οι διεργασίες και αυτό μας επιτρέπει να έχουμε μία καταπληκτική real-time εμπειρία η οποία μπορεί να εξυπηρετήσει εκατομμύρια χρήστες ταυτόχρονα χωρίς να επηρεάζεται η αποκριτικότητα. Τα αρχικά δεδομένα αποθηκεύονται τοπικά αποθηκεύονται σε μορφή JSON στην συσκευή, οπότε και αν ακόμα κάποιος χρήστης βγει offline, τα Real Time events συνεχίζουν να λειτουργούν. Το αποτέλεσμα είναι ότι όταν επανέλθει η σύνδεση η βάση θα προχωρήσει σε συγχρονισμό μεταξύ του της τοπικής και της απομακρυσμένης βάσης κάνοντας συγχώνευση στις όποιες διαφορές (εικόνα 19). [23]



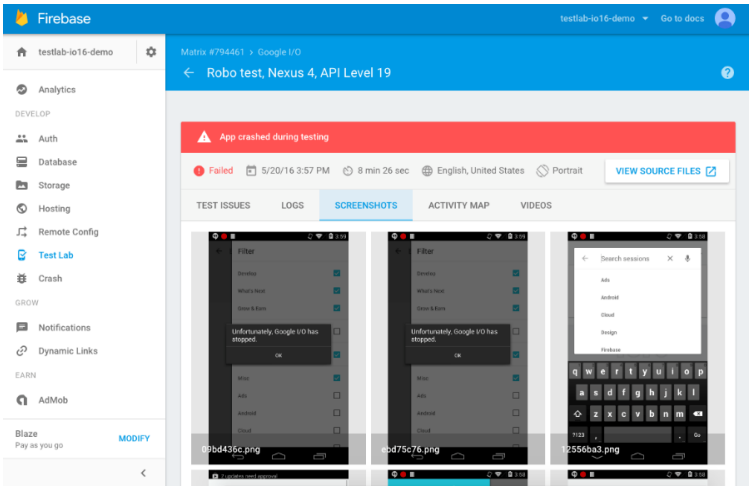
Εικόνα 19: Μορφή εικόνας Realtime Database.

- **Remote Config =** Περιλαμβάνει μία βιβλιοθήκη η οποία διαχειρίζεται όλα τα σημαντικά tasks όπως το fetching και το catching των δεδομένων κάνοντας απλό έλεγχο όταν καινούργιες τιμές προκύψουν. Αυτό επιτρέπει τη σωστή διαχείριση του χρονισμού της εφαρμογής, όταν προκύψει οποιαδήποτε αλλαγή. Χρησιμοποιεί τις ίδιες μεθόδους get τόσο στο server-side κομμάτι όσο και στο client χρησιμοποιώντας την ίδια λογική με αποτέλεσμα να μην χρειάζεται μεγάλο κομμάτι ανάπτυξης πηγαίου κώδικα (εικόνα 20). [24]



Εικόνα 20: Διάγραμμα προτεραιότητας στο Remote Config

- **Test Lab for Android =** Η Google παρέχει μέσω του test Lab μία cloud-based υποδομή για το τεστάρισμα android εφαρμογών σε πραγματικό περιβάλλον. Με μία και μόνο διαδικασία μπορούμε να τεστάρουμε την εφαρμογή μας μέσα από μία τεράστια συλλογή συσκευών με διαφορετικά χαρακτηριστικά. Τα αποτελέσματα των τεστ περιλαμβάνουν αρχεία καταγραφής, βίντεο και στιγμιότυπα οθόνης, τα οποία είναι διαθέσιμα για το εκάστοτε project μέσω της κονσόλας της Firebase. Ακόμα και αν δεν έχει δημιουργηθεί δοκιμαστικός κώδικας για το τεστάρισμα της εφαρμογής το Test Lab μπορεί να τον εκτελέσει αυτόματα ψάχνοντας για τυχόν bugs. Οι δοκιμαστικές συσκευές βρίσκονται σε ένα απομακρυσμένο Datacenter της Google το οποίο είναι ενημερωμένο με τα τελευταία android API. Μέσω του Robot test γίνεται η αυτόματη εκτέλεση της εφαρμογής σε Τεστ περιβάλλον (εικόνα 21). [25]

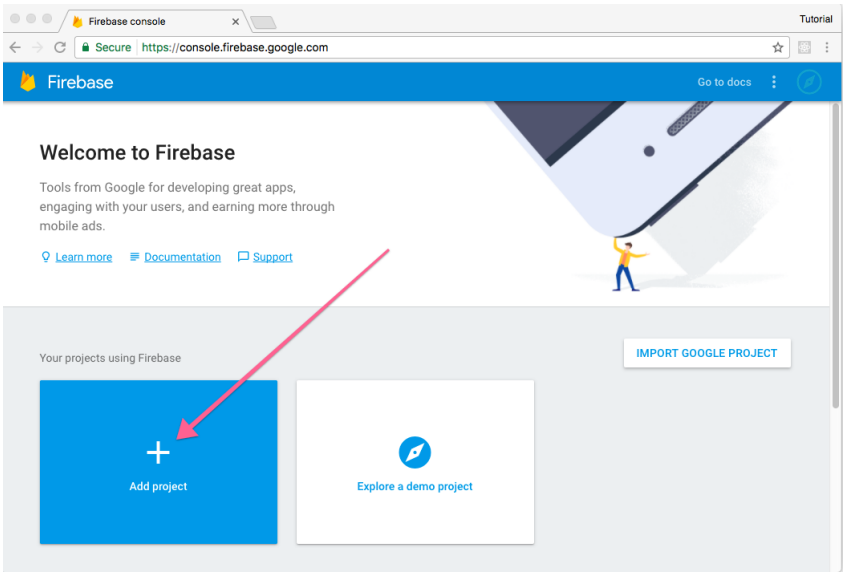


Εικόνα 21: Περιβάλλον Test Lab.

2.1.3 Δημιουργώντας ένα καινούργιο project στην Firebase.

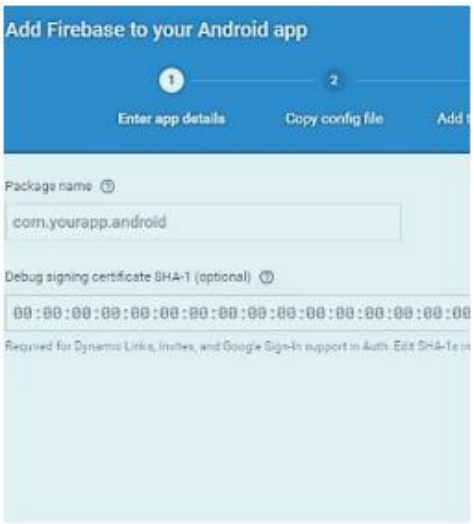
Έχοντας αναλύσει στο προηγούμενο κεφάλαιο τα βασικά χαρακτηριστικά των υπηρεσιών που προσφέρει το framework, δημιουργήθηκε ένας σύντομος οδηγός που δείχνει το πως δημιουργείτε ένα project και πως γίνεται ένα πρώτο σετάρισμα. Το API είναι δυνατόν να προστεθεί σε οποιαδήποτε υπάρχουσα εφαρμογή η οποία τρέχει Android 2.3 (Gingerbread) ή νεότερο λειτουργικό και η συσκευή να έχει εγκατεστημένη την εφαρμογή Google Play 9.6.1 ή νεότερη. Ακολουθούν βήμα - βήμα τα σημεία κλειδιά:

1. Κάνουμε click στο πεδίο Add Project και στο μενού που ανοίγει περνάμε στα πεδία το όνομα της εφαρμογής (μπορεί να είναι διαφορετικό από το όνομα της εφαρμογής) και την τοποθεσία μας (εικόνα 22).



Εικόνα 22: Δημιουργία καινούργιου project.

2. Στην συνέχεια ζητάτε το όνομα του πακέτου και η εισαγωγή του SHA-1 code (optional) που έχει παραχθεί από το android studio (εικόνα 23).



Εικόνα 23: Δήλωση ονόματος και SHA-1 code για την εφαρμογή.

3. Με την ολοκλήρωση των παραπάνω βημάτων δημιουργείται και γίνεται download ένα πακέτο google-services.json το οποίο είναι ξανά από το ίδιο σημείο ξανά διαθέσιμο προς κατέβασμα.
4. Αυτό το πακέτο θα πρέπει να προστεθεί στον φάκελο που είναι το application module για να μπορέσει να λειτουργήσει.
5. Στην συνέχεια θα πρέπει να γίνει ενσωμάτωση των βιβλιοθηκών της Firebase με σκοπό να μπορούν να χρησιμοποιηθούν από την εφαρμογή. Αυτό γίνεται προσθέτοντας τον κανόνα στο build.gradlefile για το plugin:

```
buildscript {
        // ….
        dependencies {
      //….
      classpath ‘ com.google.gms:google-services:3.0.0’
      }
}
```

6. Μερικά από τα dependencies που μπορούν να προστεθούν φαίνονται στην εικόνα 24.

Gradle Dependency Line
com.google.firebase:firebase-core:9.6.1
com.google.firebase:firebase-database:9.6.1
com.google.firebase:firebase-storage:9.6.1
com.google.firebase:firebase-crash:9.6.1
com.google.firebase:firebase-auth:9.6.1
com.google.firebase:firebase-messaging:9.6.1

Εικόνα 24: Πιθανά dependencies που μπορεί να προστεθούν στην εφαρμογή.

Για την περίπτωση που επιλέξαμε να εξετάσουμε εμείς θα δώσουμε δύο παραδείγματα για τα dependencies της βάσης δεδομένων και του χώρου αποθήκευσης. Άρα λοιπόν για να λειτουργήσουν οι συγκεκριμένες δύο υπηρεσίες θα πρέπει να γίνουν οι εξής μικρές αλλαγές στον κώδικα. Στην πρώτη περίπτωση της βάσης δεδομένων θα προστεθεί για παράδειγμα το παρακάτω κομμάτι κώδικα με σκοπό να μπορούν να εγγραφούν μη δομημένα δεδομένα στην βάση.

```
FirebaseDatabase database =FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");
myRef.setValue("Παράδειγμα Database!");
```

Η προσθήκη μπορεί να γίνει με χρήση μίας από τις παρακάτω τέσσερεις μεθόδους:

- **setValue() –** Είναι η function που κάνει εγγραφή ή αντικατάσταση των δεδομένων στην υπάρχουσα DatabaseReference.
- **push() –** Είναι η function που προσθέτει list entries στην υπάρχουσα DatabaseReference αλλά με ένα τυχαίο και μοναδικό ID.
- **updateChildren() –** Είναι η function που κάνει ενημέρωση μόνο σε κάποια keys της reference χωρίς να κάνει αντικατάσταση όλων τους.

- **runTransaction –** Είναι η function η οποία κάνει ενημέρωση με μεγάλη προσοχή λόγω εκτέλεσης παράλληλων διεργασιών ώστε να μην γίνουν τα δεδομένα corrupted.

Αντίστοιχα με την προσθήκη ενός listener (πχ ValueEventListener) μπορούμε να προσπελάσουμε τα ανάλογα δεδομένα ώστε να τα εμφανίσουμε πχ σε κάποιο dialog. Ακολουθεί ένα παράδειγμα κώδικα. [26], [27]

```
myRef.addValueEventListener(newValueEventListener(){
    @Override
    publicvoid onDataChange(DataSnapshot dataSnapshot){
       String value = dataSnapshot.getValue(String.class);
       // data is stored in string value and can be used according to
need
    }
    @Override
    publicvoid onCancelled(DatabaseError error){
        // if data is not fetched this function is used
    }});
```

Αντιστοίχως είναι και για την function του Child με έναν ChildEventListerner που περιλαμβάνει τέσσερεις functions. Ακολουθεί ένα παράδειγμα κώδικα.

```
ChildEventListener childEventListener =newChildEventListener(){
    @Override
publicvoid onChildAdded(DataSnapshot dataSnapshot,String
previousChildName){ //
retrieve list of children and listens to addition of items into list
}    @Override
publicvoid onChildChanged(DataSnapshot
dataSnapshot,StringpreviousChildName){
       // listens for a change to items of list
}
    @Override
publicvoid onChildRemoved(DataSnapshot dataSnapshot){
//listens to removal of any child from the list
           }
    @Override
 publicvoid onChildMoved(DataSnapshot dataSnapshot,String
previousChildName){
         // this function listens to change in order list}
@Override
    publicvoid onCancelled(DatabaseError error){
        // if data is not fetched this function is used
    }};
```

Σε περίπτωση επιθυμίας διαγραφής δεδομένων θα μπορούσαμε απλά να καλέσουμε την removeValue() function. Στην περίπτωση που θέλαμε να κάνουμε και σορτάρισμα θα έπρεπε ακολουθήσουμε μία από τις 3 πιθανές λύσει που είναι:

1. orderByChild() – Εμφάνιση αποτελεσμάτων με σειρά την τιμή του child key.
2. orderByKey() – Εμφάνιση αποτελεσμάτων με σειρά του child key.
3. orderByChild() – Εμφάνιση αποτελεσμάτων με σειρά του child.

Στην δεύτερη περίπτωση της αποθήκευσης θα πρέπει να δοθεί μεγάλη βάση στο κομμάτι του χειρισμού σε περίπτωση απώλειας της σύνδεσης και λειτουργείας σε offline mode. Θα πρέπει να προβλεφθεί η δημιουργία ενός σημείου αναφοράς το θα εκτελεί επαναφορά από την προηγούμενη κατάσταση. Αυτό θα μας χρησιμεύσει στην περίπτωση που θα θέλαμε να κάνουμε upload ή download κάποιο αρχείο. Ακολουθεί ένα παράδειγμα κώδικα:

- Προσθήκη dependency για την δημιουργία instance στην firebase:

```
FirebaseStorage storageobject =FirebaseStorage.getInstance();
```

- Δημιουργία σημείου reference:

```
StorageReference FileRef = storageRef.child("filePath");
```

- Δημιουργία upload task με πιθανές functions όπως την putBytes(), την putFile(), την putData() ή την putStream():

```
UploadTask uploadTask =FileRef.putBytes(data);
```

- Έλεγχος της κατάστασης με την προσθήκη listener στην περίπτωση διακοπής του upload:

```
uploadTask.addOnFailureListener(newOnFailureListener(){
    @Override
    publicvoid onFailure(@NonNullException exception){
        // Handle unsuccessful uploads
    }
}).addOnSuccessListener(newOnSuccessListener<UploadTask.TaskSnapshot
>(){
    @Override
    publicvoid onSuccess(UploadTask.TaskSnapshot taskSnapshot){
        // handles successful uploads
    }
```

- Αντίστοιχη δημιουργία κώδικα για το download. Οι functions διαχείρισης μπορεί να είναι getBytes(), get Stream() κτλ.

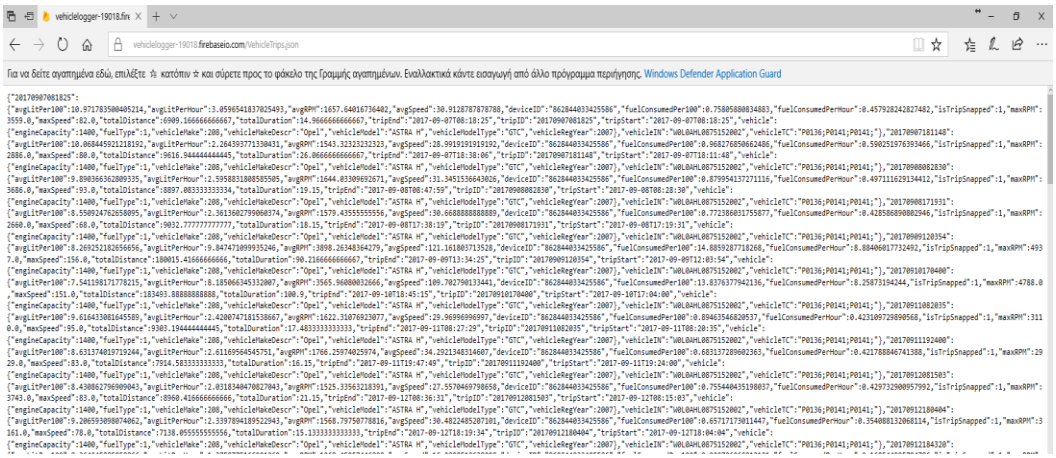
```
StorageReference FileRef =storageRef.child("images/island.jpg");
FileRef.getBytes().addOnSuccessListener(newOnSuccessListener<byte[]>
(){
    @Override
    publicvoid onSuccess(byte[] bytes){
        // handles successfully downloaded data    }
}).addOnFailureListener(newOnFailureListener(){
    @Override
    publicvoid onFailure(@NonNullException exception){
              // Handle any errors
    }});
```

[28]

2.2 Ανάλυση βάσης δεδομένων.

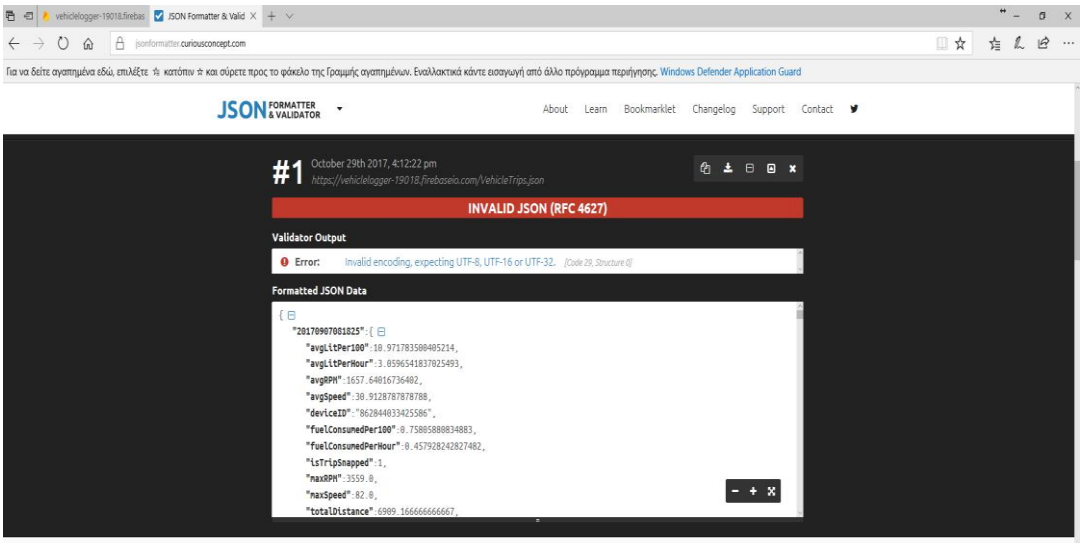
Έχοντας αναλύσει το κομμάτι που έχει να κάνει με το θεωρητικό κομμάτι της Firebase ήρθε η στιγμή να αναλύσουμε το κομμάτι που έχει να κάνει με τα raw-data. Χτυπώντας καθένα από τα 2 παρακάτω end point θα πάρουμε μία εικόνα την εικόνα 25 που είναι ουσιαστικά τα raw -data.

- https://vehiclelogger-19018.firebaseio.com/VehicleTrips.json
- https://vehiclelogger-19018.firebaseio.com/VehicleTripDetails.json



Εικόνα 25: Οπτικό αποτέλσμα εκτλέλεσης του endpoint σε broswer.

Αν περάσουμε τα αποτελέσματα από έναν JSON validator τα αποτελέσματα γίνονται πιο ευανάγνωστα και φαίνονται όπως στην εικόνα 26.



Εικόνα 26: Χρησιμοποίηση JSON validator.

Με την παραπάνω διαδικασία είναι προφανές και το κλειδί της οντότητας που είναι της μορφής YYYYMMDDHHss. Αντίστοιχα φαίνονται τα attributes σαν εγγραφή από το δρομολόγιο. Σε ανάπτυξη στο notepad ++ η μορφή θα φαίνεται όπως παρακάτω. Για να βοηθήσουμε τον αναγνώστη να αντιληφθεί την σημασία της κάθε τιμής έχουμε προχωρήσει και σε ανάλυση αυτών.

```
"20170907081825":{
  "avgLitPer100 (Μέση κατανάλωση καυσίμου σε L / 100 Km)":10.971783500405214,
  "avgLitPerHour (Μέση κατανάλωση καυσίμου σε L / Hour)":3.0596541837025493,
  "avgRPM (Μέση τιμή Σ.α.Ω κινητήρα)":1657.64016736402,
  "avgSpeed (Μέση τιμή ταχύτητας ταξιδιού)":30.9128787878788,
  "deviceID (IMEI Κινητής συσκευής)":"862844033425586",
  "fuelConsumedPer100 (Συνολική τιμή λίτρων κατανάλωσης καυσίμου βάση της
κατανάλωσης/100) ":0.75805880834883,
  "fuelConsumedPerHour (Συνολική τιμη λίτρων κατανάλωσης καυσίμου βάση της
κατανάλωσης/Hour)":0.457928242827482,
  "maxRPM (Μέγιστη τιμή Σ.α.Ω κινητήρα)":3559.0,
  "maxSpeed (Μέγιστη τιμή ταχύτητας Ταξιδίου)":82.0,
  "totalDistance (Συνολικό μήκος διαδρομής που διανύθηκε σε
μέτρα)":6909.166666666667,
  "totalDuration (Συνολικός χρόνος διάρκειας Ταξιδίου σε
λεπτά)":14.9666666666667,
  "tripEnd (Ημερομηνία και ώρα λήξης Ταξιδίου)":"2017-09-07T08:18:25",
  "tripID (Κωδικός Ταξιδίου)":"20170907081825",
  "tripStart (Ημερομηνία και ώρα έναρξης Ταξιδίου)":"2017-09-07T08:18:25",
  "vehicle":{
        "engineCapacity (Κυβικά κινητήρα)":1400,
        "fuelType (Τύπος Καυσίμου)":1,
        "vehicleMake (Κωδικός Κατασκευαστή)":208,
        "vehicleMakeDescr (Μάρκα κατασκευαστή)":"Opel",
        "vehicleModel (Μοντέλο Έκδοσης)":"ASTRA H",
        "vehicleModelType (Τύπος Μοντέλου) ":"GTC",
        "vehicleRegYear (Έτος εκδόσεως άδειας κυκλοφορίας)":2007
  },
  "vehicleIN (Αριθμός πλαισίου κινητήρα)":"W0L0AHL0875152002",
  "vehicleTC(Κωδικοί βλαβών που παρουσιάζει το συγκεκριμένο
όχημα)":"P0136;P0141;P0141;"
}
```


Ακριβώς η ίδια απεικόνιση υπάρχει και για τις τιμές της οντότητας του Vechiletrip details. Έχει γίνει αντίστοιχα η επεξήγηση και εδώ.

```
{
      "accTot": "Το συνολικό magnitude του αισθητηρίου του accelerometer",
      "accX": "Η στιγμιαία τιμή του άξονα X του accelerometer",
      "accY": "Η στιγμιαία τιμή του άξονα Y του accelerometer",
      "accZ": "Η στιγμιαία τιμή του άξονα Z του accelerometer",
      "alt": "Η στιγμιαία τιμή υψόμετρου (GPS)",
      "engineLoad": "Η στιγμιαία τιμή του φορτίου του κινητήρα (σε %)",
      "gyrX": "Η στιγμιαία τιμή του άξονα X του gyroscope",
      "gyrY": "Η στιγμιαία τιμή του άξονα Y του gyroscope",
      "gyrZ": "Η στιγμιαία τιμή του άξονα Z του gyroscope",
      "intakeAirTemp": "Η στιγμιαία τιμή της θερμοκρασίας της εισαγωγής αέρα",
      "lat": "Η στιγμιαία τιμή του Latitude (GPS)",
      "lng": "Η στιγμιαία τιμή του Longitude (GPS)",
      "lper100": "Η στιγμιαία τιμή της κατανάλωσης καυσίμου L / 100 km",
      "lperhour": "Η στιγμιαία τιμή της κατανάλωσης καυσίμου L / Hour",
      "ltft1": "Η στιγμιαία τιμή του αισθητήρα Long Term Fuel Trim",
      "maf": "Η στιγμιαία τιμή του αισθητήρα mass (air) flow sensor ",
      "provider": "Η προέλευση των δεδομένων στιγματος (Network or GPS)",
      "rpm": "Η στιγμιαία τιμή των Σ.α.Ω. του κινητήρα",
      "speed": "Η στιγμιαία τιμή της ταχύτητας",
      "stft1": "Η στιγμιαία τιμή του αισθητήρα Short Term Fuel Trim",
      "temp": "Η στιγμιαία τιμή της θερμοκρασίας του κινητήρα (σε βαθμούς
κελσίου)",
      "throttle": "Η στιγμιαία τιμή της θέσης της πεταλούδας γκαζιού (σε %)",
      "timestamp": "Η ημερομηνία και ώρα της εγγραφής",
      "tripID": "Ο κωδικός της parent εγγραφής της διαδρομής"
}
```

Η εικόνα εγγραφής που έχει περάσει από road matching είναι η ακόλουθη. Χαρακτηριστικό αυτής είναι η ύπαρξη τιμής στα πεδία matched, snapedLat και snaaped Lng.


```
   {
      "accX":"0.07888045161962509",
      "accY":"0.015986040234565735",
      "alt":"0.11372458189725876",
      "engineLoad":"5.490196228027344 %",
      "lat":"38.05086166666667",
      "lng":"23.823406666666664",
      "lper100":33.779028852920476,
      "maf":"3.0",
      "matched":"true",
      "provider":"0.0948619619011879",
      "rpm":"1095.0",
      "snappedLat":"38,05086",
      "snappedLng":"23,82339",
      "snappedName":"",
      "speed":"3",
      "temp":"25 C",
      "throttle":"6.666666507720947",
      "timestamp":"07/09/2017 08:19:00"
          },
```


Αντίστοιχη είναι η τιμή που γίνεται άσσος από μηδέν και ονομάζεται isTripsnapted.

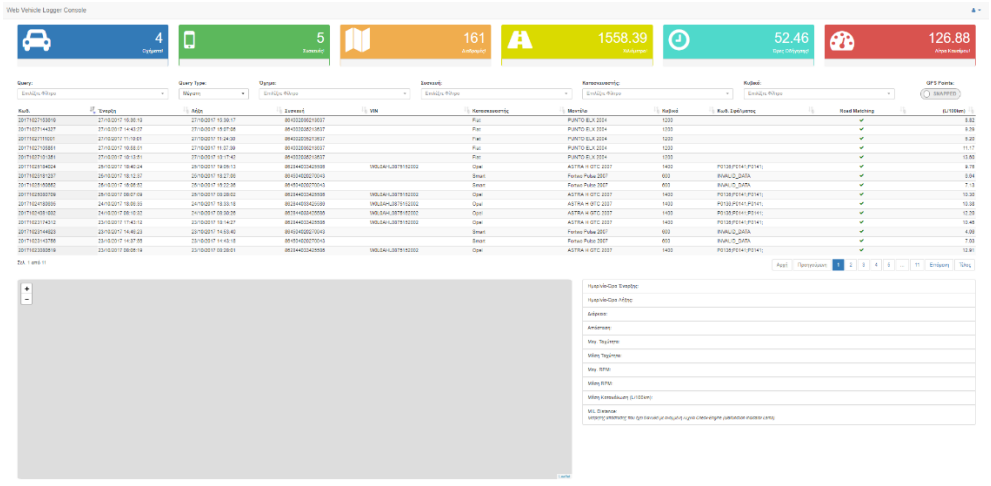

```
"20170907081825": {
              "avgLitPer100": 10.971783500405214,
              "avgLitPerHour": 3.0596541837025493,
              "avgRPM": 1657.64016736402,
              "avgSpeed": 30.9128787878788,
              "deviceID": "862844033425586",
              "fuelConsumedPer100": 0.75805880834883,
              "fuelConsumedPerHour": 0.457928242827482,
              "isTripSnapped": 1,
              "maxRPM": 3559.0,
              "maxSpeed": 82.0,
              "totalDistance": 6909.166666666667,
              "totalDuration": 14.9666666666667,
              "tripEnd": "2017-09-07T08:18:25",
              "tripID": "20170907081825",
              "tripStart": "2017-09-07T08:18:25",
              "vehicle": {
                     "engineCapacity": 1400,
                     "fuelType": 1,
                     "vehicleMake": 208,
                     "vehicleMakeDescr": "Opel",
                     "vehicleModel": "ASTRA H",
                     "vehicleModelType": "GTC",
                     "vehicleRegYear": 2007
              },
              "vehicleIN": "W0L0AHL0875152002",
              "vehicleTC": "P0136;P0141;P0141;"
```

ΚΕΦΑΛΑΙΟ 3

Όπως περιγράψαμε στο προηγούμενο κεφάλαιο η βάση δεδομένων που μας παραδόθηκε είναι σε μορφή raw data. Πριν από την εκτέλεση των queries θα αναλύσουμε δύο πολύ σημαντικά θέματα. Το πρώτο είναι να κάνουμε μια συνολική παρουσίαση του ιστότοπου τον οποίο δημιουργήσαμε για να βοηθήσουμε στην οπτικοποίηση του αποτελέσματος και το δεύτερο είναι ο τρόπος, με τον οποίο λειτουργεί το GPS Road Matching. Στο τρίτο κομμάτι θα παρουσιαστεί η μέθοδος που εκτελεί τα querries και τα αποτελέσματα αυτών.

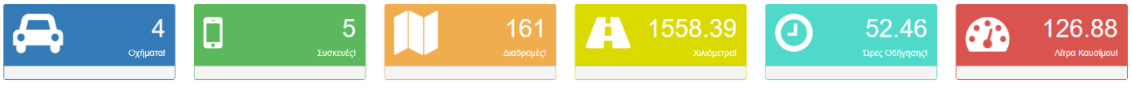
3.1 Παρουσίαση ιστοτόπου απεικόνισης.

Αρχικά λοιπόν δημιουργήθηκε ένας ιστότοπος με σκοπό τα raw data να μπορούν να απεικονιστούν οπτικά και μέσω ενός graphical user interface. Η ανάπτυξη του website έγινε καθαρά για λόγους καλύτερης απεικόνισης των αποτελεσμάτων των queries και δεν αποτελεί αντικείμενο που θα αναπτυχθεί στην παρούσα εργασία. Συνοπτικά θα αναφέρουμε κάποια από τα εργαλεία ανάπτυξης που χρησιμοποιήθηκαν. Για την δημιουργία της ASP.Net MVC web πλατφορμας χρησιμοποιήθηκε Microsoft Visual Studio 2017 και η γλώσσα προγραμματισμού C# με .Net Framework 4.6.2. Για την client-side απεικόνιση των δεδομένων χρησιμοποιήθηκαν αρκετά javascript και jquery plugins όπως, Bootstrap για την κύρια δομή της σελίδας, LeafletJS για τους χάρτες, Datatables για την εμφάνιση των δρομολογίων σε πίνακες, Highcharts για την εμφάνιση της ταλάντωσης σε γράφημα, JSLinq για το φιλτράρισμα και την εφαρμογή linq ερωτημάτων στα collections και κάποια ακόμα εργαλεία. Το site έγινε publish σε Windows Server 2008 R2 και είναι hosted σε IIS v.6.1 στην freeware υπηρεσία του https://okeanos.grnet.gr/home/ η οποία μας παρέχεται δωρεάν από την συνεργασία που έχει το Πανεπιστήμιο με την υπηρεσία. Ανοίγοντας τον σύνδεσμο http://snf-472303.vm.okeanos.grnet.gr/ από έναν browser βλέπουμε στην εικόνα 27 τον ιστοτόπο.



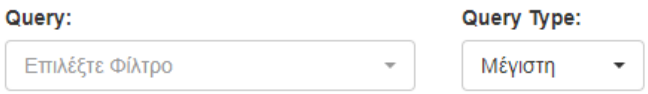
Εικόνα 27: Γενική εικόνα ιστοτόπου.

Στο πάνω μέρος όπως φαίνεται στην εικόνα 28 είναι συνολικές πληροφορίες τα δεδομένα της βάσης. Φαίνονται 4 οχήματα που έχουν κάνει εγγραφές, 5 κινητές συσκευές, 161 δρομολόγια, 1558 χιλιόμετρα καλυμμένης απόστασης, 52,4 ώρες καταγραφής και 126.8 λίτρα καυσίμου να έχουν καταναλωθεί.

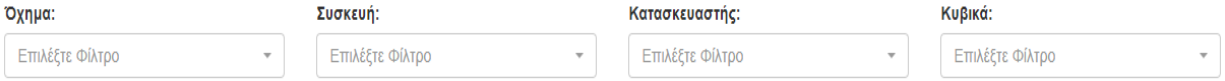


Εικόνα 28: Συγκεντρωτικοί δείκτες.

Στο αμέσως από κάτω επίπεδο υπάρχουν 6 μενού καταρράκτες όπου ο χρήστης μπορεί στα δύο πρώτα να επιλέξει το querie που θέλει να εκτελέσει και να ζητήσει αντίστοιχα την μέγιστη ή ελάχιστη τιμή (εικόνες 29,30). Τα επόμενα 4 είναι για την τοποθέτηση φίλτρου για το επόμενο επίπεδο.

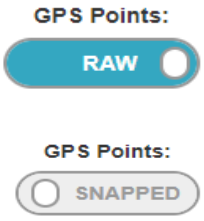


Εικόνα 29: Διπλό μενού για εκτέλεση queries.



Εικόνα 30: Τετραπλό μενού φίλτρων αποτελεσμάτων.

Ακόμα δεξιότερα είναι το κουμπί το οποίο κάνει κανονικοποίηση στα λανθασμένα στίγματα του GPS. Στην πρώτη περίπτωση γίνεται αποτύπωση με βάση τα raw-data και στην δεύτερη με βάση την κανονικοποίηση, την ακριβή διαδικασία θα την περιγράψουμε σε επόμενο κεφάλαιο (εικόνα 31).



Εικόνα 31: Οι δυο καταστάσεις του κομπιού του GPS Snapping.

Ακολουθεί το βασικότερο πεδίο που είναι το μενού επιλογής διαδρομών απεικονίζονται οι κωδικοί των διαδρομών, χρονικά timestamps, τα IMEI, τα VIN, ο κατασκευαστής, το μοντέλο, τα κυβικά οι κωδικοί σφάλματος (αν έχει το αμάξι), αν στο δρομολόγιο έχει γίνει road matching και η κατανάλωση του δρομολογίου. Υπάρχει και κάτω δεξιά μενού αναζήτησης προγενέστερων διαδρομών(εικόνα 32).

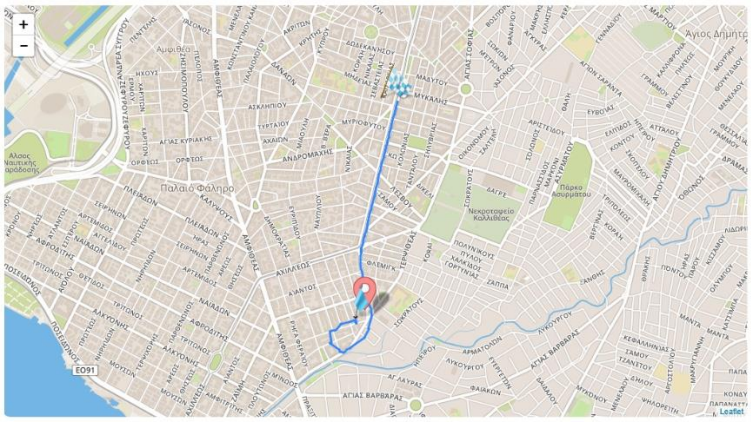
Κωδ.	Έναρξη	Λήξη	Συσκευή	VIN	Κατασκευαστής	Μοντέλο	Κυβικά	Κωδ. Σφάλματος	Road Matching	(L/100km)
20171027153619	27/10/2017 15:36:19	27/10/2017 15:39:17	864002035213637		Fiat	PUNTO ELX 2004	1200	INVALID_DATA	✔	8.82
20171027144327	27/10/2017 14:43:27	27/10/2017 15:07:06	864002035213637		Fiat	PUNTO ELX 2004	1200	INVALID_DATA	✔	9.29
20171027111001	27/10/2017 11:10:01	27/10/2017 11:24:30	864002035213637		Fiat	PUNTO ELX 2004	1200	INVALID_DATA	✔	6.20
20171027105851	27/10/2017 10:58:51	27/10/2017 11:07:39	864002035213637		Fiat	PUNTO ELX 2004	1200	INVALID_DATA	✔	11.17
20171027101351	27/10/2017 10:13:51	27/10/2017 10:17:42	864002035213637		Fiat	PUNTO ELX 2004	1200	INVALID_DATA	✔	13.60
20171025184024	25/10/2017 18:40:24	25/10/2017 19:05:13	862844033425586	W0L0AHL0875152002	Opel	ASTRA H GTC 2007	1400	P0136;P0141;P0141;	✔	9.76
20171025181237	25/10/2017 18:12:37	25/10/2017 18:27:06	864504020270043		Smart	Fortwo Pulse 2007	600	INVALID_DATA	✔	6.64
20171025160652	25/10/2017 16:06:52	25/10/2017 16:22:36	864504020270043		Smart	Fortwo Pulse 2007	600	INVALID_DATA	✔	7.13
20171025080709	25/10/2017 08:07:09	25/10/2017 08:28:02	862844033425586	W0L0AHL0875152002	Opel	ASTRA H GTC 2007	1400	P0136;P0141;P0141;	✔	10.30
20171024180635	24/10/2017 18:06:35	24/10/2017 18:33:18	862844033425586	W0L0AHL0875152002	Opel	ASTRA H GTC 2007	1400	P0136;P0141;P0141;	✔	10.38
20171024081032	24/10/2017 08:10:32	24/10/2017 08:30:26	862844033425586	W0L0AHL0875152002	Opel	ASTRA H GTC 2007	1400	P0136;P0141;P0141;	✔	12.20
20171023174312	23/10/2017 17:43:12	23/10/2017 18:14:27	862844033425586	W0L0AHL0875152002	Opel	ASTRA H GTC 2007	1400	P0136;P0141;P0141;	✔	10.46
20171023144623	23/10/2017 14:46:23	23/10/2017 14:53:40	864504020270043		Smart	Fortwo Pulse 2007	600	INVALID_DATA	✔	4.06
20171023143756	23/10/2017 14:37:56	23/10/2017 14:43:18	864504020270043		Smart	Fortwo Pulse 2007	600	INVALID_DATA	✔	7.03
20171023080619	23/10/2017 08:06:19	23/10/2017 08:28:01	862844033425586	W0L0AHL0875152002	Opel	ASTRA H GTC 2007	1400	P0136;P0141;P0141;	✔	12.91

Σελ. 1 από 11

Αρχή Προηγούμενη 1 2 3 4 5 ... 11 Επόμενη Τέλος

Εικόνα 32: Κεντρικός πίνακας επιλογής διαδρομής προς απεικόνηση.

Ακολουθεί κάτω αριστερά ο χάρτης απεικόνισης του δρομολογίου στον χάρτη (εικόνα 33).



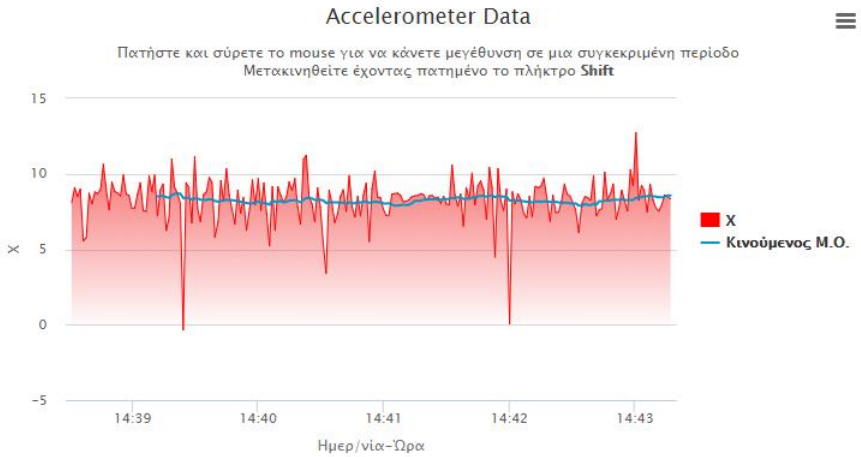
Εικόνα 33: Εκτελεσμένο δρομολόγιο.

Στα δεξιά είναι ο πίνακας με τις τιμές ενός ολοκληρωμένου δρομολογίου (εικόνα 34).

Ημερ/νία-Ώρα Έναρξης:	**23/10/2017 14:38:30**
Ημερ/νία-Ώρα Λήξης:	**23/10/2017 14:43:17**
Διάρκεια:	**5.37 min**
Απόσταση:	**1.61 km**
Μεγ. Ταχύτητα:	**42.00 Km/h**
Μέση Ταχύτητα:	**18.76 Km/h**
Μεγ. RPM:	**2894.00 RPM**
Μέση RPM:	**1656.84 RPM**
Μέση Κατανάλωση (L/100km):	**7.03 L/100km**
MIL Distance: *Μετρητής απόστασης που έχει διανυθεί με αναμμένη λυχνία Check-Engine (Malfunction Indicator Lamb)*	**NODATA km**

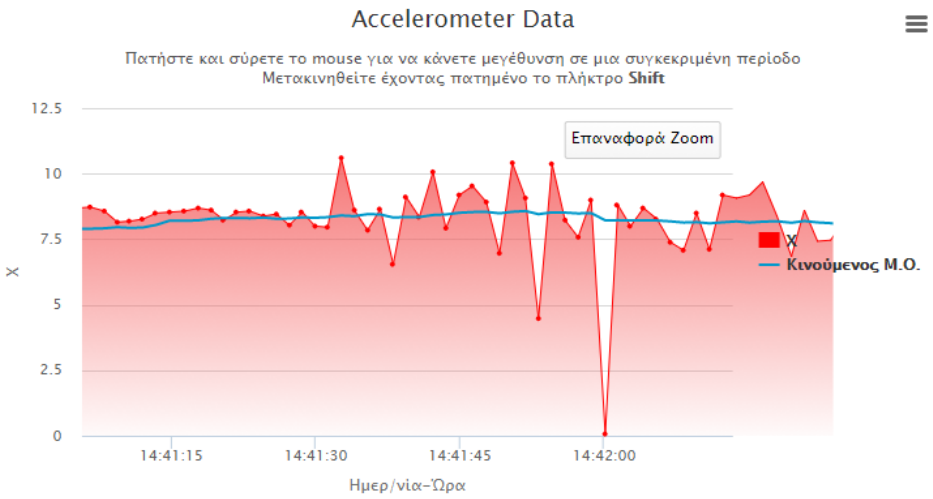
Εικόνα 34: Εικόνα τιμών ολοκληρωμένου δρομολογίου.

Ακριβώς από κάτω βρίσκεται η απεικόνιση του γραφήματος του accelerometer (εικόνα 35).



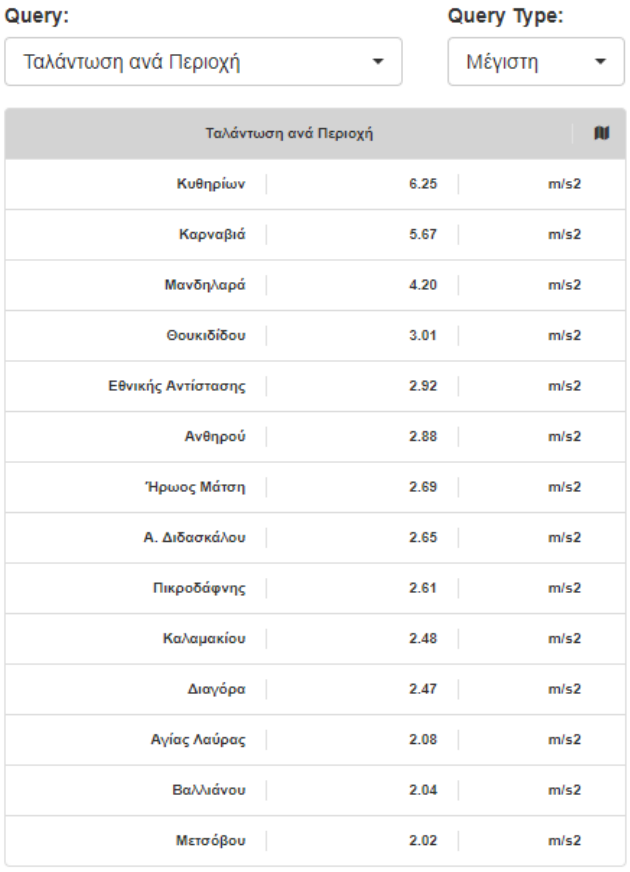
Εικόνα 35: Γράφημα τιμών accelerometer για ολοκληρωμένο δρομολόγιο.

Σε περίπτωση δε που κάποιος κάνει κλικ επάνω του ενεργοποιείτε η ιδιότητα του zoom (εικόνα 36).



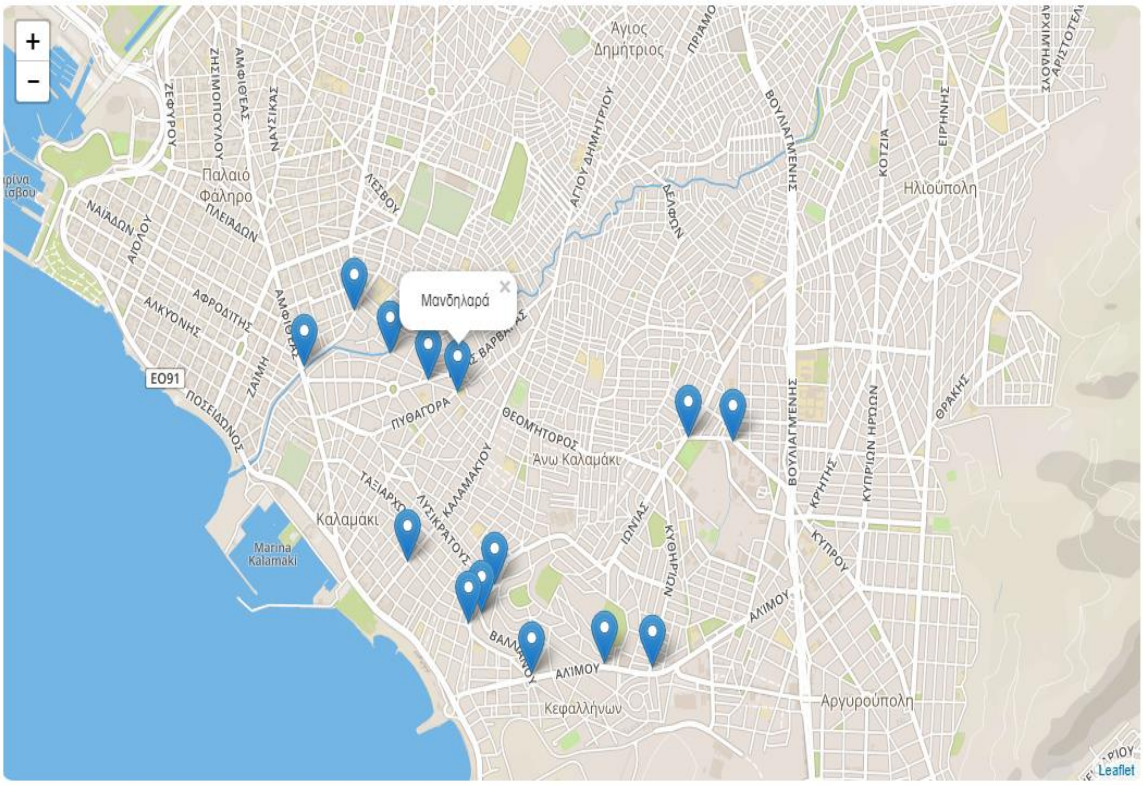
Εικόνα 36: Ενεργοποίηση ιδιότητας zoom.

Αν πάλι κάποιος εκτελέσει για παράδειγμα ένα query η εικόνα από τα αποτελέσματα μπορεί να είναι αυτή που ακολουθεί (εικόνα 37).



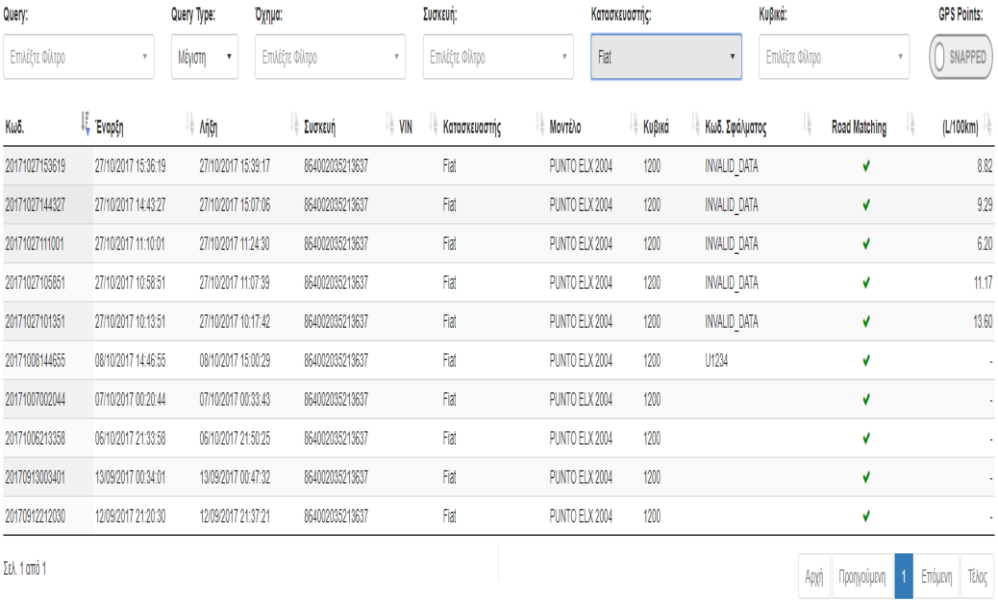
Εικόνα 37: Εικόνα αποτελεσμάτων query.

Αν κάποιος δε κλικάρει στην προηγούμενη εικόνα επάνω δεξιά στο σημαιάκι τότε τοποθετούνται με pin τα σημεία στον χάρτη (εικόνα 38).



Εικόνα 38: Τοποθέτηση pin στον χάρτη.

Τέλος αν γίνει επιλογή κάποιου φίλτρου όπως για παράδειγμα αυτό του κατασκευαστή το αποτέλεσμα θα είναι όπως στην εικόνα 39.



Κωδ.	Έναρξη	Λήξη	Συσκευή	VIN	Κατασκευαστής	Μοντέλο	Κυβικά	Κωδ. Σφάλματος	Road Matching	(L/100km)
20171027153619	27/10/2017 15:36:19	27/10/2017 15:39:17	864002005213637		Fiat	PUNTO ELX 2004	1200	INVALID_DATA	✓	8.82
20171027144327	27/10/2017 14:43:27	27/10/2017 15:07:06	864002005213637		Fiat	PUNTO ELX 2004	1200	INVALID_DATA	✓	9.29
20171027111001	27/10/2017 11:10:01	27/10/2017 11:24:30	864002005213637		Fiat	PUNTO ELX 2004	1200	INVALID_DATA	✓	6.20
20171027105851	27/10/2017 10:58:51	27/10/2017 11:07:39	864002005213637		Fiat	PUNTO ELX 2004	1200	INVALID_DATA	✓	11.17
20171027101351	27/10/2017 10:13:51	27/10/2017 10:17:42	864002005213637		Fiat	PUNTO ELX 2004	1200	INVALID_DATA	✓	13.60
20171008144655	08/10/2017 14:46:55	08/10/2017 15:00:29	864002005213637		Fiat	PUNTO ELX 2004	1200	U1234	✓	-
20171007002044	07/10/2017 00:20:44	07/10/2017 00:33:43	864002005213637		Fiat	PUNTO ELX 2004	1200		✓	-
20171006213358	06/10/2017 21:33:58	06/10/2017 21:50:25	864002005213637		Fiat	PUNTO ELX 2004	1200		✓	-
20170913003401	13/09/2017 00:34:01	13/09/2017 00:47:32	864002005213637		Fiat	PUNTO ELX 2004	1200		✓	-
20170912212030	12/09/2017 21:20:30	12/09/2017 21:37:21	864002005213637		Fiat	PUNTO ELX 2004	1200		✓	-

Σελ 1 από 1 Αρχή Προηγούμενη 1 Επόμενη Τέλος

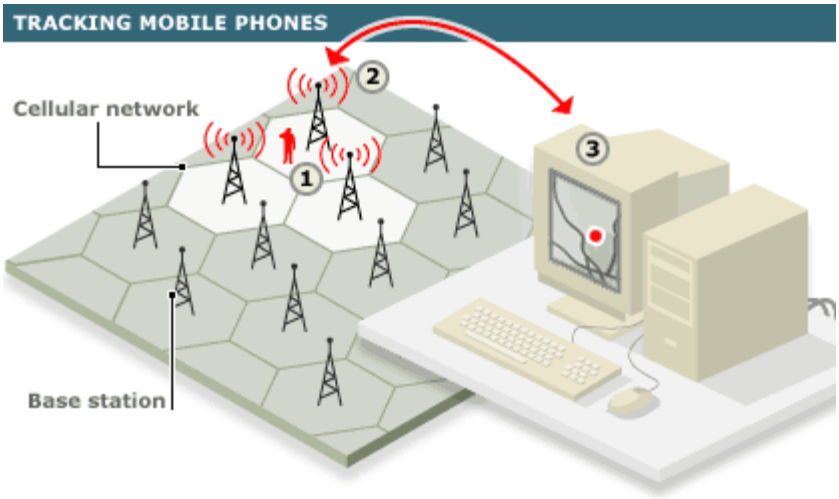
Εικόνα 39: Εικόνα αποτελεσμάτων μετά από επιλογή φίλτρου κατασκευαστή.

3.2 Επεξήγηση λειτουργείας GPS Snapping.

Όπως προαναφέραμε και νωρίτερα η μόνη επέμβαση που έγινε στα raw data τα οποία μας παραδόθηκαν έχει να κάνει με την διόρθωση των σημείων που κατέγραψε το gps των κινητών συσκευών. Σε αυτό το σημείο θα επεξηγηθεί αναλυτικά η λειτουργία του κουμπιού για το GPS Snapping.

3.2.1 Θεωρητικό υπόβαθρο.

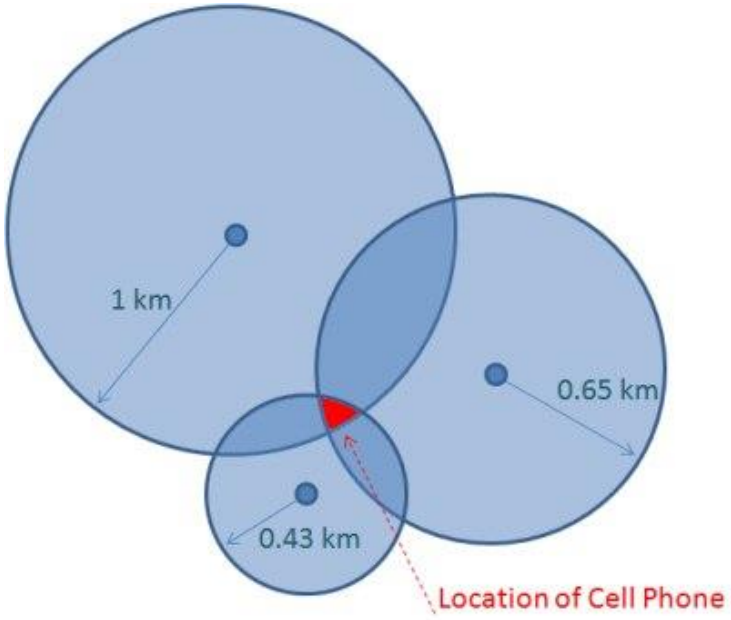
Επειδή ακριβώς μιλάμε για καταγραφή σημείων πάνω σε οδικό δίκτυο (στο οποίο υπάρχει συνεχής κίνηση) είναι απόλυτα φυσιολογικό κάποιες φορές να το αισθητήριο καταγραφής μίας κινητής συσκευής να έχει απώλεια επικοινωνίας με τον δορυφόρο με αποτέλεσμα οι μέτρηση που εκτελεί να έχει μεγάλη απόκλιση σε σχέση με το σημείο το οποίο βρίσκεται κάποιος πραγματικά. Αυτό συμβαίνει για δύο βασικούς λόγους. Ο πρώτος είναι γιατί για λόγους ασφαλείας ποτέ ένα GPS δεν δίνει το ακριβές στίγμα μίας συσκευής αλλά το σημείο με απόκλιση maximum 30 μέτρα. Ο δεύτερος έχει να κάνει με εναλλαγή την οποία κάνει μία εφαρμογή στην συλλογή μέτρησης από το σήμα κινητής τηλεφωνίας ή του Wi-Fi. Αυτό πρακτικά σημαίνει ότι όταν δεν είναι διαθέσιμο για sampling το GPS τότε το service το οποίο εκτελεί την υπηρεσία του geolocation προσπαθεί να καταλάβει την θέση του χρήστη με την διαδικασία του τριγωνισμού από κεραίες της κινητής τηλεφωνίας (εικόνα 40).



Εικόνα 40: Μέθοδος εντοπισμού θέσης μέσω κεραιών κινητής τηλεφωνίας.

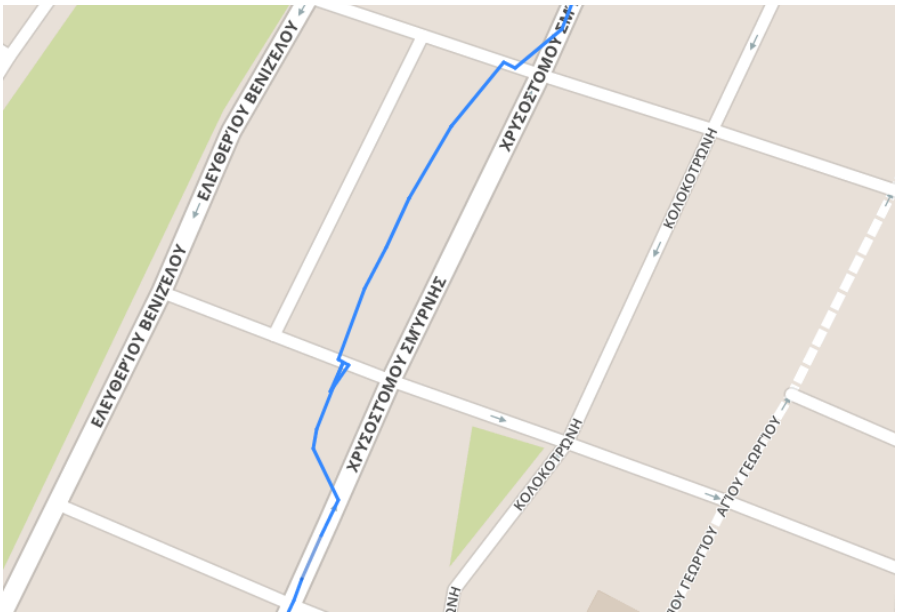
Η διαδικασία είναι σχετικά απλή. Η συσκευή για να μπορέσει να έχει σήμα στέλνει συνεχόμενα requests προς όλες της κατευθύνσεις. Κάθε σταθμός βάσης (κεραία κινητής τηλεφωνίας) έχει υπό την επίβλεψη του μία συγκεκριμένη γεωγραφική περιοχή που ονομάζεται κυψέλη. Όταν η συσκευή βρίσκεται εντός αυτού του γεωγραφικού χώρου η κεραία αναλαμβάνει να γίνει όταν λάβει το request από την συσκευή να κάνει την προεπιλεγμένη πύλη για το κομμάτι του δικτύου. Με την λογική όμως του ότι όλες η συσκευές κινούνται περιμετρικά μίας κεραίας δεν είναι δυνατόν να υπολογιστεί με ακρίβεια το σημείο στο οποίο βρίσκεται κάποιος. Ο λόγος είναι προφανής διότι αν ενώναμε θεωρητικά με κέντρο το σημείο που βρίσκεται η κεραία αναπτύσσεται ένας κύκλος με ακτίνα την απόσταση που έχει θεωρητικά κάποιος από αυτήν το οποίο δημιουργεί φαινομενικά άπειρα σημεία στα οποία αυτός μπορεί να βρίσκεται πάνω στην περίμετρο του κύκλου αυτού. Συνεπώς η διαδικασία εντοπισμού δεν γίνεται από μια κεραία και μόνο αλλά χρειάζονται τουλάχιστον ακόμα 2 για να μπορεί κάποιος με μεγάλη ακρίβεια να τοποθετήσει το ακριβές στίγμα πάνω σε έναν χάρτη. Η κεραία που έχει υπό την επίβλεψη της (γιατί βρίσκεται εντός της κυψέλης της) την συσκευή στέλνει request προς 2 ακόμα σταθμούς βάσης – συνήθως τους κοντινότερους – και ζητάει από αυτούς να κάνουν eco requests (κάτι σαν ping) προς την

συσκευή που εκείνη διαχειρίζεται. Εκείνες αφού εκτελέσουν την διαδικασία ουσιαστικά επιστρέφουν προς τα πίσω την απόσταση της ακτίνας του δικού τους πιθανού κύκλου σημείων. Με την χρήση του κατάλληλου αλγορίθμου δημιουργούνται 3 εικονικοί κύκλοι με κέντρο κάθε έναν σταθμό βάσης, οι οποίοι έχουν κάποιο σημείο επικάλυψης περιοχής. Το κοινό σημείο επικάλυψης είναι και το σημείο που ουσιαστικά βρίσκεται η κινητή συσκευή (εικόνα 41).



Εικόνα 41: Εύρεση κοινής περιοχής μέσω επικάλυψης.

Έτσι λοιπόν σε περίπτωση που κάποιος προσπαθήσει να απεικονίσει οπτικά τα raw data σε χάρτη χωρίς πρώτα να απομονώσει από την μέτρηση τις τιμές γεωγραφικού μήκους και πλάτους από σήμα κινητής τηλεφωνίας καθώς και να κάνει normalize τα data από το gps θα έχει ένα αποτέλεσμα όπως φαίνεται στην εικόνα 42. [29]



Εικόνα 42: Απεικόνιση στίγματος gps με raw data χωρίς normalization.

Για να μην συμβαίνει αυτό και να μπορεί να απεικονιστεί σωστά το αποτέλεσμα των μετρήσεων εφαρμόστηκε η λύση του road matching μέσω του API που παρέχει η εταιρεία Mapbox. Το API χρησιμοποιεί το service της υπηρεσίας OpenStreetMap που έχει Open Database License. Υπάρχουν αντίστοιχα service και άλλων εταιριών όπως η Google Maps, η Bing Maps, η MapQuest, το Here, η Yandex Maps και η Apple Maps (για συσκευές με λειτουργικό iOS).

Ο λόγος που χρησιμοποιήθηκε η συγκεκριμένη υπηρεσία και όχι κάποιας άλλης εταιρείας είναι τα παρακάτω χαρακτηριστικά:

- Έχει availability σε όλες τις χώρες του κόσμου.
- Υποστηρίζεται για IE7+, Mozilla Firefox 3.5+, Google Chrome 4+, Safari 4+ web browsers.
- Έχει διαθέσιμες όλες τις γλώσσες και υποστηρίζονται πάνω από 50 διαθέσιμες μεταφράσεις.

Το API για map matching της Mapbox κάνει snaping τις ανακριβείς μετρήσεις του gps και χρησιμοποιεί το path και road network του directions API. Τα μόνα limitations που έχει αφορούν τα maximum 60 requests ανά λεπτό, την μετατροπή maximum 100 συντεταγμένων ανά request και την προβολή του χάρτη στα libraries του SDK. Θα μπορούσε να χρησιμοποιηθεί κάποιο άλλο API όπως αυτό της Google αλλά υπήρχε περιορισμός για maximum 50 request (μετά υπήρχε χρέωση για την υπηρεσία). Επίσης το Mapbox δίνει αρχικά μέχρι 50000 request το μήνα δωρεάν. Αυτά είναι υπεραρκετά για το πλήθος των διαδρομών τα οποία εξετάζουμε. [30],[31]

Ένα παράδειγμα request περιγράφετε στον παρακάτω σύνδεσμο: https://api.mapbox.com/matching/v5/mapbox/driving/-117.1728265285492,32.71204416018209;-117.17288821935652,32.712258556224;-117.17293113470076,32.712443613445814;-117.17292040586472,32.71256999376694;-117.17298477888109,32.712603845608285;-117.17314302921294,32.71259933203019;-117.17334151268004,32.71254065549407?access_token=your-access-token

Επίσης αυτές είναι οι αρχικές παράμετροι:

Παράμετρος URL	**Περιγραφή**
`Προφίλ`	Προφιλ ID οδηγιών;είτε `mapbox/οδήγησης` , `mapbox/οδήγησης για κίνηση`, `mapbox/περπατήματος` , or `mapbox/ποδηλασίας`
`συντεταγμένων`	Semicolon-separated list of `{γ. πλάτος},{γ. μήκος}` κατεύθυνση ζευγαριών με την σειρά; μεταξύ 2 και 100 συντεταγμένων.

Και το αποτέλεσμα που επιστρέφει:

```
"code": "Ok",
  "durations": [
    [ 0,    77.3, null ],
    [ 75.7, 0,    null ],
    [ null, null, 0    ]
  ],
  "destinations": [
    {
      "location": [
        -6.80897,
        62.000075
      ],
      "name": "Kirkjubøarvegur"
    },
```

[32],[33],[34]

3.2.2 Υλοποίηση του κώδικα για το Road Matching.

Αφού περιεγράφηκε παραπάνω η υπηρεσία που θα αναλάβει την ομαλοποίηση των raw δεδομένων του GPS, σε δεδομένα τα οποία αντιστοιχούν σε σημεία πραγματικών οδών (μόνο για χαρτογραφημένες περιοχές), σε αυτή τη φάση θα περιγραφεί η διαδικασία SnapTripDetailsToRoadsAsync μέσω της οποίας πραγματοποιείται η κλήση της υπηρεσίας μέσα από την διαδικτυακή πλατφόρμα που υλοποιήθηκε.

Η αρχιτεκτονική δομή της διαδικασίας αυτής είναι πολύ απλή και περιγράφεται ως εξής: Η συνάρτηση λαμβάνει ως όρισμα τον κωδικό δρομολογίου και τον τύπο δεδομένων του GPS που χρειάζονται να φιλτραριστούν. Έτσι, εφόσον ο αριθμός δρομολογίου είναι έγκυρος, η συνάρτηση επικοινωνεί μέσω ενός Firebase Client (FireSharp v.2.0.3) με την real-time database του endpoint το οποίο διατηρεί αποθηκευμένα τα δεδομένα των διαδρομών, για να ανακτήσει όλες τις εγγραφές του συγκεκριμένου δρομολογίου.

```
IFirebaseConfig config = new FirebaseConfig { BasePath =
"https://vehiclelogger-
      19018.firebaseio.com/" };
IFirebaseClient client = new FirebaseClient(config);
FirebaseResponse response = await client.GetAsync("VehicleTripDetails/"
+ tripID);
```

Εφόσον επιστραφούν οι εγγραφές για το συγκεκριμένο δρομολόγιο, σειρά έχει το φιλτράρισμά τους βάσει του τύπου δεδομένων του GPS. Πιο συγκεκριμένα, όταν για κάποιο λόγο δεν υπάρχουν διαθέσιμες εγγραφές στίγματος απευθείας από το δορυφόρο (π.χ. κατά την αρχικοποίηση του GPS δεκτή, κατά τη διέλευση μέσα από ένα τούνελ, κ.α.), το λειτουργικό σύστημα των κινητών συσκευών λαμβάνει το στίγμα μέσω του διαδικτύου. Δυστυχώς όμως, αυτή η μέθοδος είναι αρκετά ανακριβής και τα σημεία που προέρχονται από αυτή τη μέθοδο, καταφέρνουν και παραμορφώνουν το σύνολο των σημείων της διαδρομής και μπορούν να οδηγήσουν σε πολύ λανθασμένα αποτελέσματα. Συνεπώς, αφού φιλτραριστούν και απομακρυνθούν όλες αυτές οι εγγραφές από το σύνολο των δεδομένων της διαδρομής, η συνάρτηση προχώρα έτσι ώστε να υλοποιηθεί η ομαλοποίηση των σημείων.

Όπως περιεγράφηκε παραπάνω, η υπηρεσία δεν θα επιστρέψει σωστά αποτελέσματα εάν ως είσοδο λάβει πολλά ίδια σημεία. Πρακτικά αυτό σημαίνει ότι ένα όχημα το οποίο περιμένει στο φανάρι ή όταν μία καταγραφή διαδρομής η οποία έχει ξεκινήσει ενώ το αυτοκίνητο δεν κινείται, παράγει πολλά ίδια στίγματα. Αυτά τα σημεία λοιπόν ομαδοποιούνται και αποστέλλονται μόνο οι μοναδικές τιμές τους, έτσι ώστε τα αποτελέσματα της υπηρεσίας Road Matching να μην επηρεαστούν.

```
if (!string.IsNullOrEmpty(providerflag))
{
if (normalizedTripDetails.Where(x => x.provider == providerflag).Any())
   normalizedTripDetails =
      normalizedTripDetails.Where(x       =>       x.provider       ==
providerflag).ToList();
}
```

Όπως επίσης έχει αναφερθεί κατά την περιγραφή της υπηρεσίας σε προηγούμενη ενότητα, η υπηρεσία έχει ως όριο 100 ζεύγη στίγματος [Latitude, Longitude] ανά request. Αυτό σημαίνει πως αφού ομαδοποιηθούν όλα τα στίγματα και σχηματιστεί μια συλλογή από μοναδικά ζεύγη σημείων, αυτά θα πρέπει να σπάσουν σε υποομάδες των 100 ζευγών. Συνεπώς, μέσα από μια επαναληπτική διαδικασία, η συνάρτηση παίρνει όλα τα σημεία ανά 100 και σχηματίζει ένα νέο request σε κάθε επανάληψη.

```
for (int i = 0; i < normalizedTripDetails.Count(); i = i + 100)
{
  var points = string.Join(";", normalizedTripDetails.Skip(i)
  .Take(100)
  .Select(x => x.lng.Replace(",", ".") + "," + x.lat.ToString()
  .Replace(",", ".")).Distinct());

  var timestamps = string.Join(";", normalizedTripDetails.Skip(i)
  .Take(100)
  .GroupBy(x => x.lng.Replace(",", ".") + "," + x.lat.ToString()
  .Replace(",", "."))
  .Select(g => g.First()).ToList()
  .Select(x                                                    =>
((DateTimeOffset)DateTime.Parse(x.timestamp)).ToUnixTimeSeconds()));

  var requestUri =
      string.Format($"https://api.mapbox.com/matching/v5/mapbox/drivin
g/{points}?tidy
      =false&access_token={APIKey}");
}
```

Στο request αυτό, εκχωρούνται ως όρισμα όλα τα ζεύγη στιγμάτων σε μορφή concatenated string της μορφής “Lat[1];Long[1],Lat[2];Long[2], …, Lat[100];Long[100]”, ο τύπος των data που θα ζητηθούν (driving, cycling, walking) και το API key του λογαριασμού της υπηρεσίας. Αφού κληθεί η υπηρεσία και επιστρέψει αποτελέσματα, αυτά ελέγχονται εάν είναι έγκυρα ή εάν περιέχουν κάποιο κωδικό σφάλματος. Εφόσον είναι έγκυρα, το JSON response γίνεται deserialize σε C# object τύπου MatchResponse και ελέγχονται τα περιεχόμενά του. Τα normalized στίγματα περιέχονται μέσα στο nested object Tracepoints, το οποίο περιέχει μια συλλογή με nested objects τύπου Location και με property Name η οποία περιέχει το όνομα της οδού που αντιστοιχεί στο συγκεκριμένο στίγμα (εφόσον υπάρχει).

```
"tracepoints": [
{
  "alternatives_count": 0,
  "waypoint_index": 0,
  "location": [
      24.172836,
      32.712041
  ],
  "name": "Ερμού",
  "matchings_index": 0
}, ...
```

Εφόσον η συλλογή των normalized σημείων που επιστράφηκε έχει το ίδιο μέγεθος με τα σημεία που δόθηκαν ως όρισμα, ξεκινά η διαδικασία εκχώρησης των normalized σημείων στις εγγραφές του δρομολογίου. Καθώς απαιτείται να παραμείνει η πληροφορία των raw data της διαδρομής, τα normalized σημεία δεν θα αντικαταστήσουν τα αρχικά αλλά σε κάθε εγγραφή θα προστεθούν 4 νέα properties όπως φαίνεται παρακάτω:

```
var tracepoint = matchingRes.tracepoints[tripdetail.index];
if (tracepoint != null
      && tracepoint.location != null
            && tracepoint.waypoint_index >= 0)
{
```

```
        tripdetail.value.matched = "true";
        tripdetail.value.snappedLat = tracepoint.location[1].ToString();
        tripdetail.value.snappedLng = tracepoint.location[0].ToString();
        tripdetail.value.snappedName = tracepoint.name;
}
```

Συνεπώς, σε κάθε εγγραφή προστίθενται τα properties matched, snappedLat, snappedLng και snappedName και έτσι μετά το τέλος όλων των επαναλήψεων, όλες οι εγγραφές της συλλογής θεωρούνται ως normalized.

Πριν ολοκληρωθεί η διαδικασία και επιστραφεί η συλλογή, απαιτείται να πραγματοποιηθεί μια ένωση μεταξύ των normalized σημείων της συλλογής και των αρχικών σημείων της διαδρομής, μέσα στα οποία περιέχονται και τα φιλτραρισμένα data τα οποία είχαν απομακρυνθεί στο πρώτο βήμα. Αυτό απαιτείται διότι ακόμα και οι εγγραφές που περιέχουν στίγματα τα οποία δεν προέρχονται από δορυφόρο, περιέχουν χρήσιμες πληροφορίες για την διαδρομή (ταχύτητα, κατανάλωση, κ.λπ.) οπότε και δεν πρέπει να απομακρυνθούν από την αρχική συλλογή δεδομένων της διαδρομής ακόμα και αν δεν έχουν γίνει normalized.

```
tripdt.details =
tripDetails.Union(normalizedTripDetails,                                    new
FirebaseVehicleTripDetailComparer())
.Distinct(new FirebaseVehicleTripDetailComparer())
.ToArray();


return tripdt;
```

Με τις παραπάνω εντολές, πραγματοποιείται ένα union μεταξύ της αρχικής λίστας εγγραφών της διαδρομής και της normalized λίστας και ως αποτέλεσμα περιέχεται μία λίστα η οποία περιέχει όλα τα σημεία normalized και μη, της διαδρομής. Η παραπάνω διαδικασία δεν θα είχε καμία χρησιμότητα εάν δεν γινόταν σε κανένα σημείο της εφαρμογής η αξιοποίησή της. Καθώς δεν θεωρείται αποδεκτή λύση η χειροκίνητη κλήση της ανά δρομολόγιο μέσω κάποιου UI, έπρεπε με κάποια μέθοδο αυτή η διαδικασία να εκτελείται για κάθε νέο δρομολόγιο που προστίθεται στην Firebase και ολοκληρώνεται. Αυτό επιτυγχάνεται με την χρήση ενός listener ο οποίος αρχικοποιείται κατά την έναρξη λειτουργίας της διαδικτυακής πλατφόρμας μέσα στην κλάση Global.asax και κατά το event Application_Start.

```
IFirebaseConfig config = new FirebaseConfig { BasePath =
"https://vehiclelogger-
      19018.firebaseio.com/" };
IFirebaseClient client = new FirebaseClient(config);
EventStreamResponse response =
      await client.OnAsync("VehicleTrips",
            added: async (sender, args, context) =>
            { … }
```

Με τον παραπάνω κώδικα αρχικοποιείται ο client της Firebase και «δένεται» ένας listener ο οποίος παρακολουθεί την κίνηση των εγγραφών στο Path .\VehicleTrips ο οποίος γίνεται trigger όταν μια εγγραφή τύπου VehicleTrip προστίθεται (onAdded). Κατά την προσθήκη ενός δρομολογίου λοιπόν πυροδοτείται το παραπάνω event μέσα στο οποίο ελέγχεται η εγκυρότητα του κωδικού δρομολογίου ο οποίος χρησιμοποιείται για να ανακτηθεί ολόκληρη η εγγραφή. Αφού ανακτηθεί το δρομολόγιο, αρχικά ελέγχεται η ύπαρξη του property tripEnd μέσα από το οποίο γίνεται η αναγνώριση μιας ολοκληρωμένης διαδρομής.

Εφόσον η διαδρομής είναι νέα, δεν θα έχει τιμή στην ιδιότητα αυτή, οπότε ο listener αγνοεί το συγκεκριμένο δρομολόγιο, έως ότου αυτό ολοκληρωθεί. Εφόσον το δρομολόγιο

ολοκληρώνεται, εκτελείται η παραπάνω διαδικασία και ελέγχεται η ύπαρξη του property isTripSnapped το οποίο προστίθεται σε κάθε δρομολόγιο που γίνεται normalize μέσω της SnapTripDetailsToRoadsAsync. Εφόσον δεν υπάρχει τιμή σε αυτή την ιδιότητα, σημαίνει πως το δρομολόγιο δεν έχει γίνει normalized και καλείται η function SnapTripDetailsToRoadsAsync για να ξεκινήσει η διαδικασία που περιγράφηκε στην προηγούμενη ενότητα. Καθώς ολοκληρώνεται η διαδικασία επιτυχώς, τα normalized πλέον detail του δρομολογίου επιστρέφονται στον listener, μέσα από τον οποίο αυτά γίνονται upload στο path VehicleTripDetails κάτω από τον κωδικό της εγγραφής του δρομολογίου και αντικαθιστούν όλα τα υπάρχοντα.

Ως αποτέλεσμα, η Firebase περιέχει όλα τα δεδομένα που περιείχε κατά την ολοκλήρωση του δρομολογίου, με την διαφορά πως σε όσες εγγραφές βρέθηκαν snapped roads, περιέχεται και η πληροφορία των normalized Latitude & Longitude ζευγών μαζί με την ονομασία της οδού που αναγνωρίστηκε. Συνεπώς, η διαδικασία αυτή θα επαναλαμβάνεται αυτόματα κάθε φορά που ολοκληρώνεται ένα δρομολόγιο σε όλες τις διαδρομές τα ζεύγη των στιγμάτων θα γίνονται normalized.

3.3 Εκτέλεση queries και παρουσίαση αποτελεσμάτων.

Έχοντας καλύψει το θεωρητικό υπόβαθρο που λειτουργεί η βάση δεδομένων φτάσαμε στο σημείο το οποίο είναι και ο σκοπός της εργασίας. Παρακάτω θα εξηγήσουμε τις 2 main functions που δημιουργήθηκαν για να εκτελεστούν τα queries και στην συνέχεια θα παρουσιάσουμε τα αποτελέσματα που επιστρέφουν.

3.3.1 Υλοποίηση κώδικα για εκτέλεση queries.

Οι βασικές λειτουργίες της διαδικτυακής πλατφόρμας είναι να δίνει μια οπτική μορφή στα δεδομένα των καταγεγραμμένων δρομολογίων που βρίσκονται στην Firebase και να τα παρουσιάζει μέσω ποικίλων γραφικών στοιχείων στην οθόνη του χρήστη. Για να πραγματοποιηθούν όλα τα παραπάνω όμως, πρώτα πρέπει να ανακτηθούν τα δεδομένα από την Firebase.

Για την ανάκτηση των δεδομένων από την real-time database της Firebase υπάρχουν αρκετοί τρόποι. Οι κυριότεροι είναι μέσω του REST API που παράγει αυτόματα το framework ή μέσω του client. Το κυριότερο σημείο που πρέπει να γίνει κατανοητό είναι πως τα data ανακτώνται μέσω παραμέτρων που προστίθενται στο βασικό URL του endpoint. Στην περίπτωση της παρούσας εργασίας, όπου και παρακολουθείται το endpoint https://vehiclelogger-19018.firebaseio.com/ . Δύο είναι τα κύρια paths με τα data των δρομολογίων, το path VehicleTrips που περιέχει τα headers των δρομολογίων και το path VehicleTripDetails που περιέχει τα detail κάτω από κάθε κωδικό δρομολογίου.

Αν μέσω μιας REST GET κλήσης κληθεί το URL https://vehiclelogger-19018.firebaseio.com/VehicleTrips/.json , θα επιστραφούν όλα τα headers των δρομολογίων σε μορφή JSON ενώ εάν στο παραπάνω path προστεθεί ο κωδικός ενός δρομολογίου πχ https://vehiclelogger-19018.firebaseio.com/VehicleTrips/20171027153619.json θα επιστραφούν τα δεδομένα μόνο αυτού του δρομολογίου. Οι ίδιοι κανόνες ισχύουν αντίστοιχα και για το path VehicleTripDetails και για οποιοδήποτε άλλο path προστεθεί με οποιοδήποτε είδους data.

Για να υλοποιηθούν τα queries για τα data των VehicleTrips και των VehicleTripDetails υλοποιήθηκαν δύο generic functions οι οποίες μπορούν να εξυπηρετήσουν πολλαπλά ερωτήματα με δυναμικό τρόπο. Αυτές περιγράφονται παρακάτω.

1. **VehicleTrips.GetMaxMinSumAvgByPropertyAsync**

Αυτή η function χρησιμοποιείται για να αναπαράγει αποτελέσματα σε ερωτήματα τύπου Μέγιστη/Ελάχιστη τιμή και Άθροισμα/Μέσος Όρος μιας τιμής, εξυπηρετώντας έτσι τους βασικότερους τύπους queries.

Ως ορίσματα δέχεται το όνομα της ιδιότητας (propname) στην οποία θα εφαρμοστούν τα παραπάνω ερωτήματα, το όνομα της ιδιότητας (groupbypropname) επάνω στην οποία θα γίνει

το grouping (εφόσον απαιτείται), ο τύπος του query (querytype 1: Max, 2: Min, 3: Sum, 4: Average) και ο αριθμός των αποτελεσμάτων (limitresults) που πρέπει να επιστραφούν.

Εφόσον περαστούν τα παραπάνω ορίσματα, η function επικοινωνεί με την real-time database της Firebase και ανακτά όλα τα headers των δρομολογίων. Εφόσον επιστραφούν τα αποτελέσματα σε μορφή JSON, γίνεται deserialize των data σε objects τύπου VehicleTrip και σχηματίζεται μια συλλογή δρομολογίων. Έπειτα σχηματίζονται δυναμικά αντικείμενα τύπου PropertyDescriptor για τις ιδιότητες propname και groupbypropname. Εφόσον οι τιμές σε αυτές τις ιδιότητες υπάρχουν μέσα στα αντικείμενα, τότε πραγματοποιείται το φιλτράρισμα της συλλογής χρησιμοποιώντας απλές εντολές LINQ.

Για παράδειγμα, για να υλοποιηθεί ένα query της μορφής «επέστρεψε τις 10 μεγαλύτερες καταναλώσεις ανά Κατασκευαστή οχήματος» θα πρέπει πρώτα να ανακτηθούν όλα τα δρομολόγια με τις παρακάτω εντολές:

```
var response = await GetFirebaseVehicleTripsAsync();
if (response != null)
{
      var vehicleTrips =
      (List<WebVehicleLoggerDAL.Entities.VehicleTrip>)((JsonResult)res
ponse).Data;
}
```

Εφόσον όλα τα δρομολόγια ανακτηθούν και γίνει με επιτυχία το deserialize τους σε λίστα με αντικείμενα, έπειτα θα πρέπει να πραγματοποιηθεί grouping βάσει της groupbypropname ιδιότητας. Στην περίπτωση του παραδείγματος, θα πρέπει να γίνει grouping βάσει του Κατασκευαστή και έπειτα θα πρέπει να υπολογισθεί ο Μ.Ο. της κατανάλωσης για κάθε group. Αυτό επιτυγχάνεται με τις παρακάτω εντολές:

```
if (groupbyprop != null && !string.IsNullOrEmpty(groupbyprop.Name))
{
      grouping = vehicleTrips
      .Where(x => groupbyprop.GetValue(x) != null && prop.GetValue(x)
      != null)
      .GroupBy(x => groupbyprop.GetValue(x))
      .Select(g => new
      {
            GroupByPropName = groupbypropname,
            AvgPropName = propname,
            GroupByProp = g.Key,
            AvgProp = g.Average(s => (decimal)prop.GetValue(s)),
            Cnt = g.Count()
      }).ToList();
}
```

Στο παραπάνω κομμάτι κώδικα, πραγματοποιείται το grouping και σχηματίζεται μια λίστα από αντικείμενα τα οποία περιέχουν το όνομα της ιδιότητας επάνω στην οποία εφαρμόστηκε το grouping, το όνομα της ιδιότητας επάνω στην οποία εφαρμόστηκε η aggregate function, η τιμή της ιδιότητας του grouping η οποία για το παραπάνω παράδειγμα θα εμφανίζει το λεκτικό του κατασκευαστή, η ζητούμενη τιμή της ιδιότητας της μέσης κατανάλωσης για αυτό το group και το πλήθος των δρομολογίων που περιέχει αυτό το group.

Έτσι, εάν η παράμετρος querytype εκφράζει πχ τις μέγιστες καταναλώσεις, τότε η παραπάνω λίστα ταξινομείται με φθίνουσα σειρά από την μεγαλύτερη προς την μικρότερη κατανάλωση και εφόσον η τιμή της παραμέτρου limitresults έχει οριστεί π.χ. στα 10 αποτελέσματα, τότε επιστρέφεται μια λίστα με τα 10 πρώτα αποτελέσματα, δηλαδή τους 10 κατασκευαστές με τις μεγαλύτερες καταναλώσεις. Αυτό επιτυγχάνεται με τις παρακάτω εντολές:

```
switch (querytype)
{
```

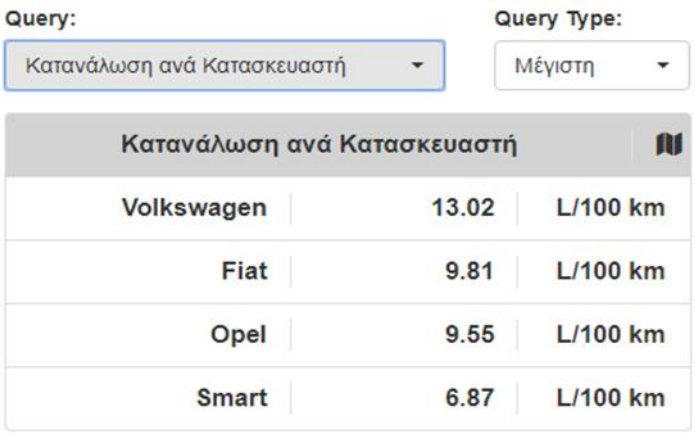
```
  case 1:
      // max
      {
            if (grouping.Any())
            {
                  grouping = grouping
                        .OrderByDescending(x => x.AvgProp)
                        .Take(limitresults.Value).ToList();

                  return new JsonResult()
                  {
                        Data = grouping,
                        JsonRequestBehavior                                    =
JsonRequestBehavior.AllowGet,
                        MaxJsonLength = int.MaxValue
                  };
            }
      }
}
```

Τέλος, εφόσον σχηματιστεί η συλλογή, επιστρέφεται με μορφή JSON αντικειμένων και οποιοσδήποτε client μπορεί να απεικονίσει τα αποτελέσματα με οποιοδήποτε τρόπο (πχ πίνακας, γράφημα, text, κλπ.). Για το παραπάνω παράδειγμα για τους 10 κατασκευαστές με τις μεγαλύτερες καταναλώσεις, το αποτέλεσμα είναι το ακόλουθο:


```
[{
      "GroupByPropName": "VTHD_VehicleMake",
      "AvgPropName": "VTHD_AVGLitPer100",
      "GroupByProp": "Volkswagen",
      "AvgProp": 13.021318582935164,
      "Cnt": 1
},
{
      "GroupByPropName": "VTHD_VehicleMake",
      "AvgPropName": "VTHD_AVGLitPer100",
      "GroupByProp": "Fiat",
      "AvgProp": 9.81464188309308478,
      "Cnt": 5
},
{
      "GroupByPropName": "VTHD_VehicleMake",
      "AvgPropName": "VTHD_AVGLitPer100",
      "GroupByProp": "Opel",
      "AvgProp": 9.554997068525590150943396226,
      "Cnt": 106
},
{
      "GroupByPropName": "VTHD_VehicleMake",
      "AvgPropName": "VTHD_AVGLitPer100",
      "GroupByProp": "Smart",
      "AvgProp": 6.868272961427901732558139534 9,
      "Cnt": 43
}]
```


Σύμφωνα με το παραπάνω JSON result, η ιδιότητα με την οποία έγινε το grouping και για το παράδειγμα αφορά τον κατασκευαστή, έχει την ονομασία VTHD_VehicleMake ενώ η ιδιότητα της κατανάλωσης έχει το όνομα VTHD_AVGLitPer100. Πρώτος κατασκευαστής στη λίστα εμφανίζεται η Volkswagen έχοντας όμως μόνο μία διαδρομή καταγεγραμμένη, με μέση κατανάλωση 13.021 L / 100 km. τα JSON data επιστρέφονται σε όποιον client χρησιμοποιήσει την αντίστοιχη function και για την παρούσα εργασία, παρουσιάζονται με την εικόνα 43:



Εικόνα 43: Εικόνα κατανάλωσης ανά κατασκευαστή από το GUI.

Όπως γίνεται κατανοητό, με την παραπάνω διαδικασία είναι εφικτό να αναπαραχθούν αρκετά queries όπως π.χ. Μέγιστη / Ελάχιστη Κατανάλωση ανά μοντέλο, Κατανάλωση ανά κατασκευαστή, Στροφές ανά Λεπτό ανά κατασκευαστή, Μέση Ταχύτητα ανά Κατασκευαστή, Μοντέλα με βλάβες, Βλάβες ανά μοντέλα, κ.λπ.

2. VehicleTripDetails.GetMaxMinSumAvgByPropertyAsync.

Η παραπάνω λογική της διαδικασίας χρησιμοποιείται και για τα queries που αφορούν τις αναλυτικές εγγραφές των δρομολογίων. Η υλοποίηση είναι ακριβώς η ίδια απλά διαφοροποιείται για τους ξεχωριστούς τύπους δεδομένων.

Σε αυτή την περίπτωση πρέπει να ανακτηθούν όλα τα αναλυτικά δεδομένα των δρομολογίων, γεγονός που όπως προαναφέρθηκε προκαλεί μια αναμονή μέχρι να ολοκληρωθεί η διαδικασία. Εφόσον ανακτηθούν όλες οι αναλυτικές εγγραφές από όλα τα δρομολόγια και σε αυτή την περίπτωση εφαρμόζονται groupings και aggregate functions ενώ και εδώ επιστρέφονται JSON data σε όποιον client κάνει consume αυτή τη function.

Επίσης, άλλη μία διαφοροποίηση είναι πως εφόσον τα queries απαιτούν grouping βάσει Περιοχής/Οδού, μέσα σε κάθε object που επιστρέφεται περιλαμβάνεται και το πρώτο object κάθε group το οποίο περιέχει πληροφορία ενός ζεύγους συντεταγμένων [Latitude,Longitude] της συγκεκριμένης οδού, έτσι ώστε να υπάρχει και η γεωκωδικοποίηση της πληροφορίας σε περίπτωση όπου ο client επιθυμεί να απεικονίσει την πληροφορία σε κάποιο χάρτη. Αυτό επιτυγχάνεται με τις παρακάτω εντολές:

```
if (groupbyprop != null && !string.IsNullOrEmpty(groupbyprop.Name))
{
      grouping = vehicleTripDetails
      .Where(x => groupbyprop.GetValue(x) != null && prop.GetValue(x)
!= null)
      .GroupBy(x => groupbyprop.GetValue(x))
      .Select(g => new
      {
            GroupByPropName = groupbypropname,
            AvgPropName = propname,
            GroupByProp = g.Key,
                  AvgProp = g.Average(s =>
                        decimal.Parse(prop.GetValue(s).ToString().Repl
                  ace(".", ","),
                        System.Globalization.NumberStyles.Float)),
```

```
                Cnt = g.Count(),
                Point = new
                {
                        LatLng = g.TakeWhile(s =>
                        !string.IsNullOrEmpty(s.lat) &&
                                !string.IsNullOrEmpty(s.lng)).Select(x =>
                        x.lat + ";" +     x.lng).First(),
                        AddressName = g.TakeWhile(s =>
                                !string.IsNullOrEmpty(s.snappedName))
                        .First().snappedName
                }
        }).ToList();
}
```

Παραπάνω, διακρίνεται η διαφοροποίηση στο nested object Point στο οποίο εκχωρείται το πρώτο ζεύγος Latitude, Longitude του group μαζί με το όνομα της οδού. Έτσι, τα αποτελέσματα έχουν την παρακάτω μορφή:

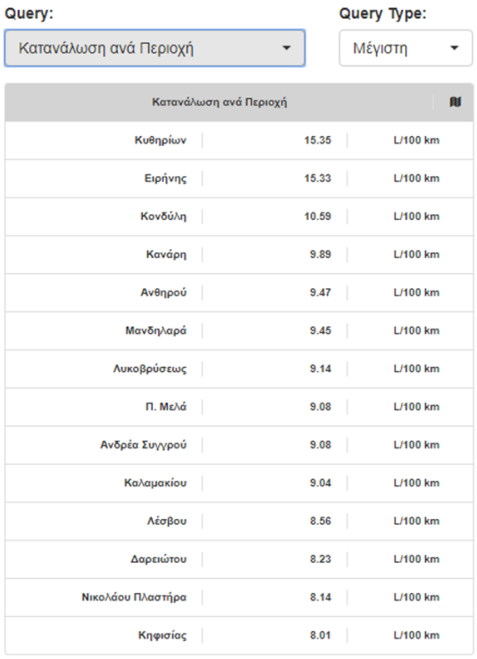
```
{
        "GroupByPropName": "snappedName",
        "AvgPropName": "lper100",
        "GroupByProp": "Κυθηρίων",
        "AvgProp": 15.354104042053223,
        "Cnt": 1,
        "Point": {
                "LatLng": "37.90755;23.733144999999997",
                "AddressName": "Κυθηρίων"
        }

}
```

Σε μορφή raw JSON data οι δέκα οδοί με την μεγαλύτερη κατανάλωση θα απεικονίζονται όπως στην εικόνα.

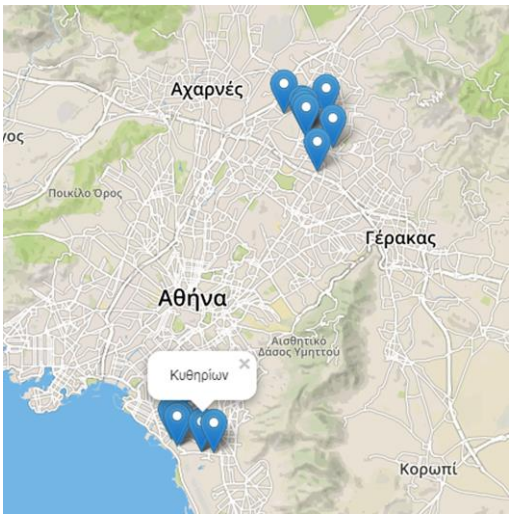
Εικόνα 44: Οπτική απεικόνιση αποτελεσμάτων raw data μετά από εκτέλεση queries.

Αντιστοίχως θα είναι το αποτέλεσμα αν αυτό ζητηθεί κάνοντας click και επιλέγοντας την κατάλληλη παράμετρο από το μενού καταρράκτη στο GUI του website (εικόνα 45).



Εικόνα 45: Το ίδιο αποτέλεσμα με εμφάνιση στο GUI.

Επιπρόσθετα, και εφόσον σε κάθε object, υπάρχει nested το Point object με το ζεύγος Latitude, Longitude της οδού, η κάθε οδός απεικονίζεται πλέον και με markers επάνω στο χάρτη όπως στην εικόνα 46 παρακάτω:



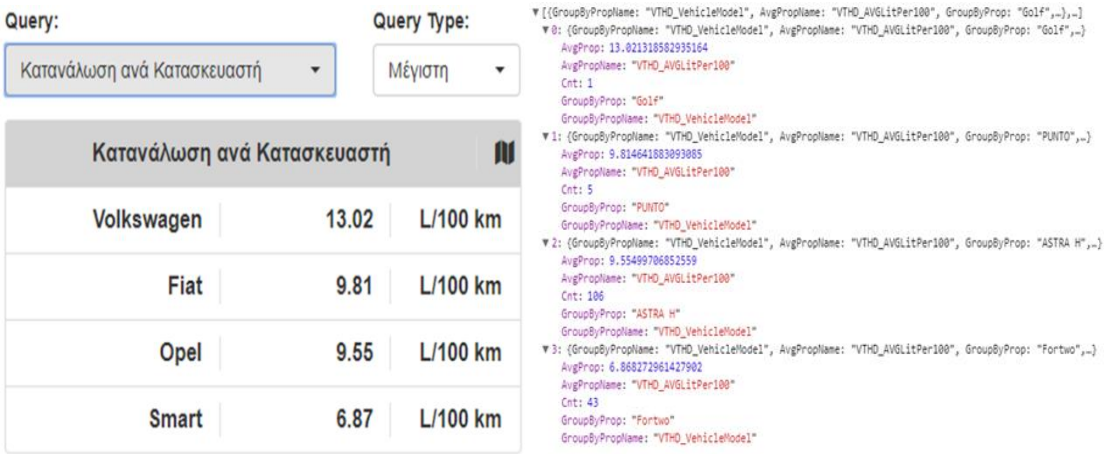
Εικόνα 46: Απεικόνιση τοποθέτησης markers στον χάρτη.

Όπως γίνεται κατανοητό, με την παραπάνω διαδικασία είναι εφικτό να αναπαραχθούν αρκετά queries και για τα αναλυτικά data των δρομολογίων όπως πχ η Ταλάντωση ανά Περιοχή/Οδό, η Ταχύτητα ανά Περιοχή/Οδό, κλπ.

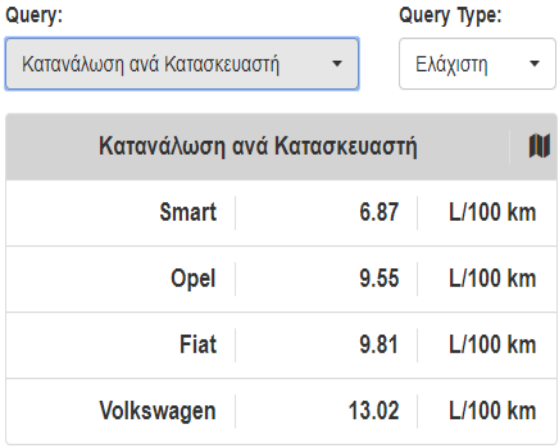
3.3.2 Αποτελέσματα των εκτελεσθέντων queries.

Συνολικά εκτελέστηκαν οχτώ διαφορετικά queries εκ των οποίων τα 6 από αυτά και με την παράμετρο μέγιστης και ελάχιστης τιμής. Σε κάθε query υπάρχει screenshot από το οπτικό αποτέλεσμα που εμφανίζεται στο website αλλά και εικόνα από το αποτέλεσμα που εμφανίστηκε σε JSON raw data. Παρακάτω παρουσιάζονται τα αποτελέσματα.

1. **Μέγιστη και ελάχιστη κατανάλωση ανά κατασκευαστή.**



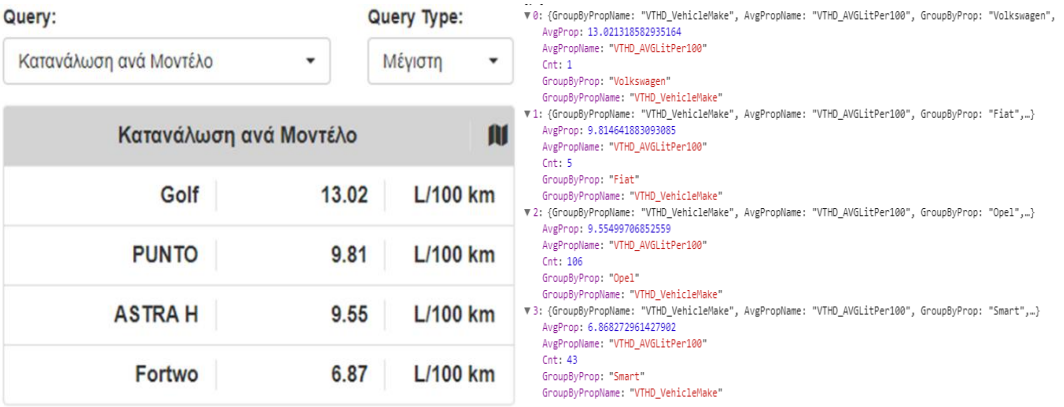
Εικόνα 47: Εμφάνιση αποτελέσματος μέγιστης κατανάλωσης ανά κατασκευαστή σε GUI και raw data.



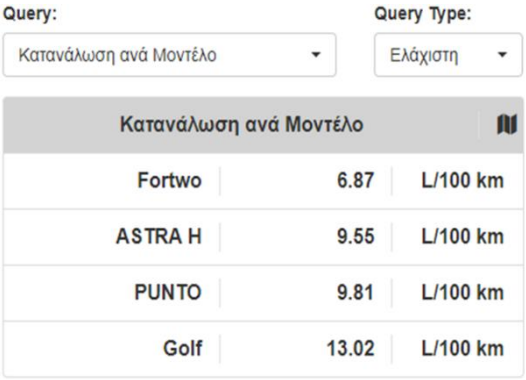
Εικόνα 48: Εμφάνιση αποτελέσματος ελάχιστης κατανάλωσης ανά κατασκευαστή στο GUI.

Είναι ουσιαστικά το ανάποδο αποτέλεσμα. Αυτό συμβαίνει και στα περισσότερα querries.

2. Μέγιστη και ελάχιστη κατανάλωση ανά μοντέλο κατασκευαστή.

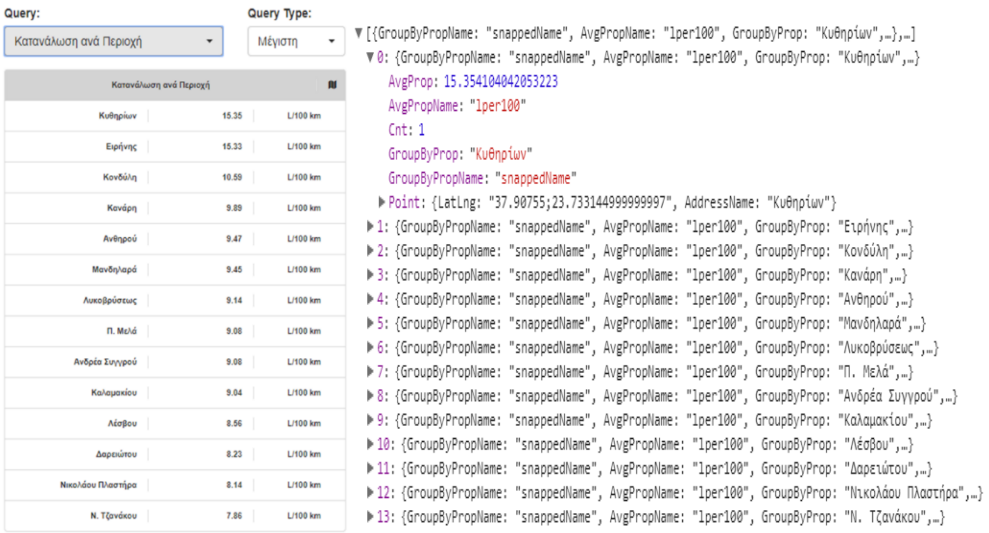


Εικόνα 49: Εμφάνιση αποτελέσματος μέγιστης κατανάλωσης ανά μοντέλο σε GUI και raw data.

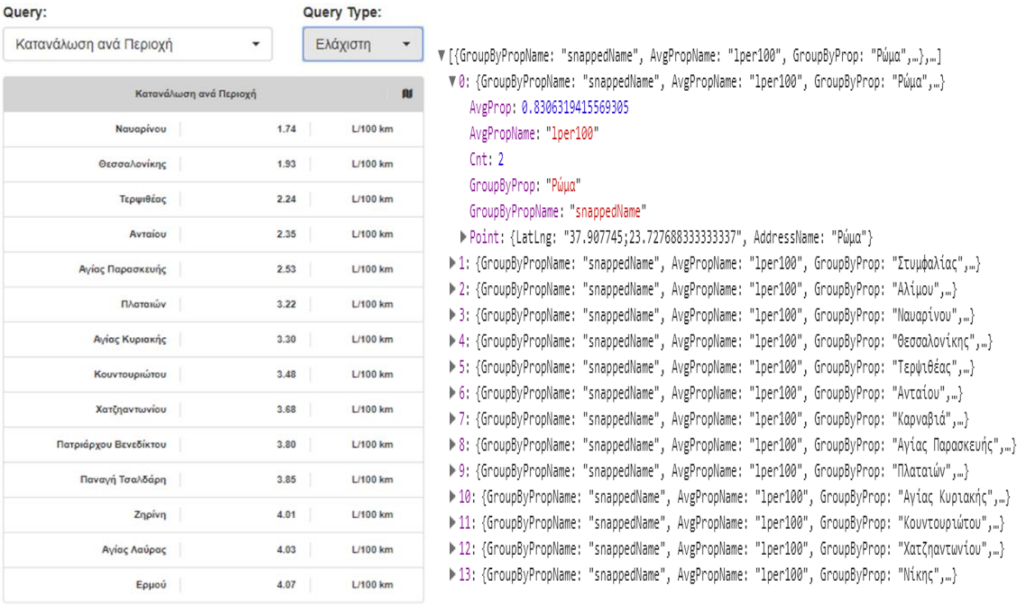


Εικόνα 50: Εμφάνιση αποτελέσματος ελάχιστης κατανάλωσης ανά μοντέλο στο GUI.

3. Μέγιστη και ελάχιστη κατανάλωση ανά περιοχή.

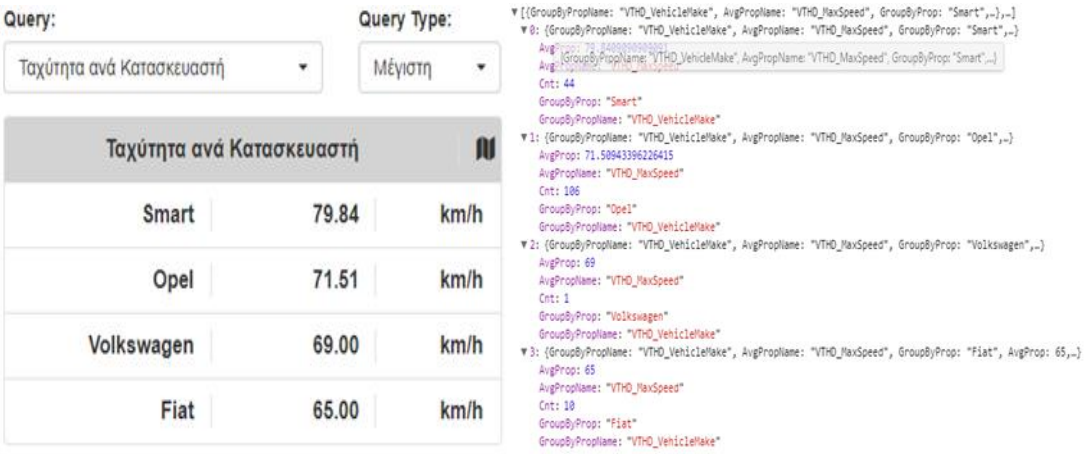


Εικόνα 51: Εμφάνιση αποτελέσματος μέγιστης κατανάλωσης ανά περιοχή σε GUI και raw data.

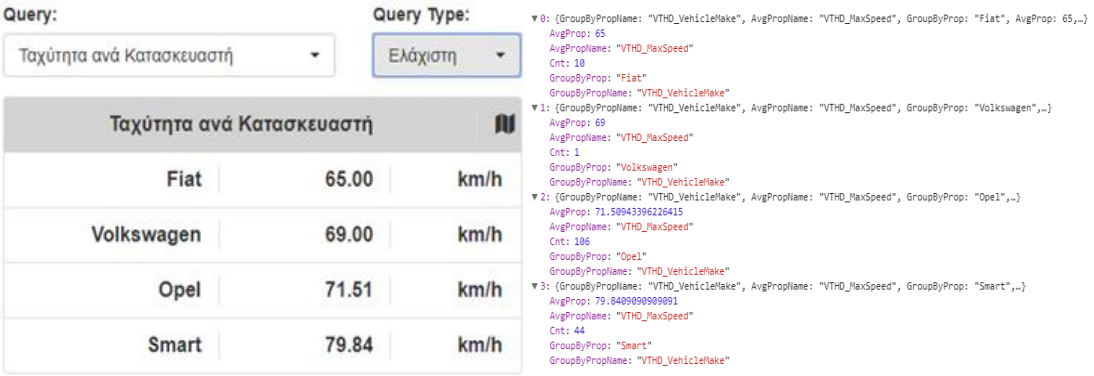


Εικόνα 52: Εμφάνιση αποτελέσματος ελάχιστης κατανάλωσης ανά περιοχή σε GUI και raw data.

4. Μέγιστη και ελάχιστη μέση ταχύτητα ανά κατασκευαστή.

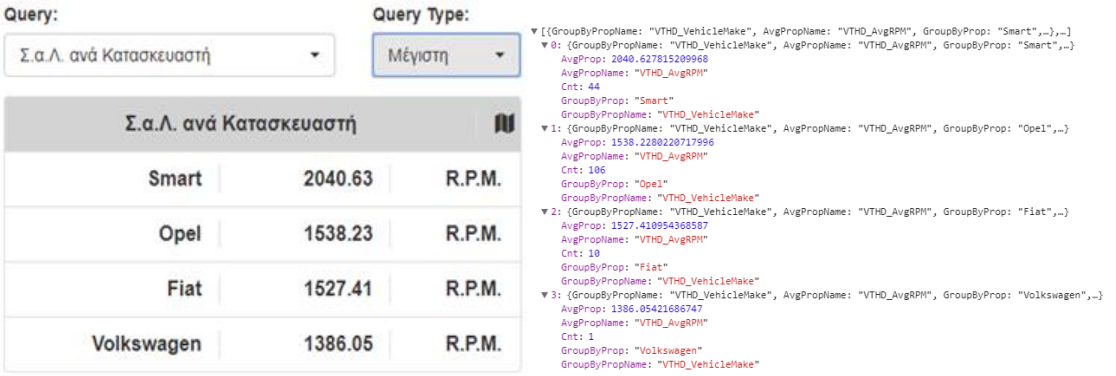


Εικόνα 53: Εμφάνιση αποτελέσματος μέγιστης μέσης ταχύτητας ανά κατασκευαστή σε GUI και raw data.

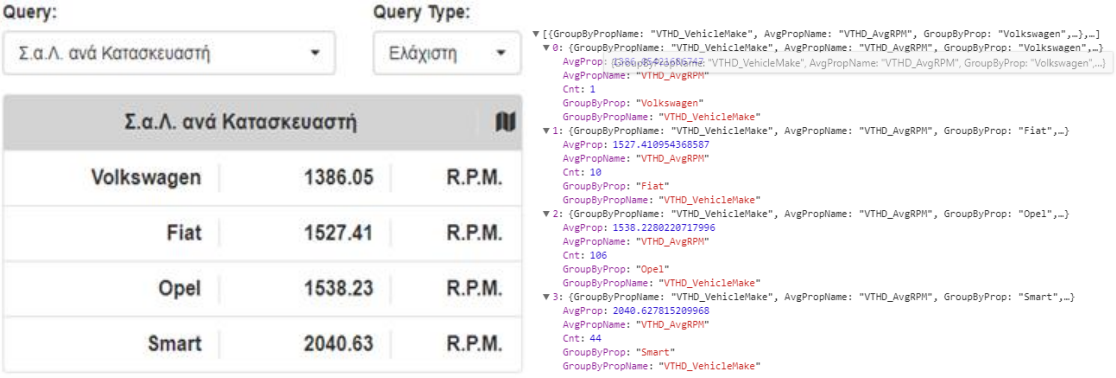


Εικόνα 54: Εμφάνιση αποτελέσματος ελάχιστης μέσης ταχύτητας ανά κατασκευαστή σε GUI και raw data.

5. Μέγιστη και ελάχιστη μέση τιμή στροφών κινητήρα ανά λεπτό ανά κατασκευαστή.

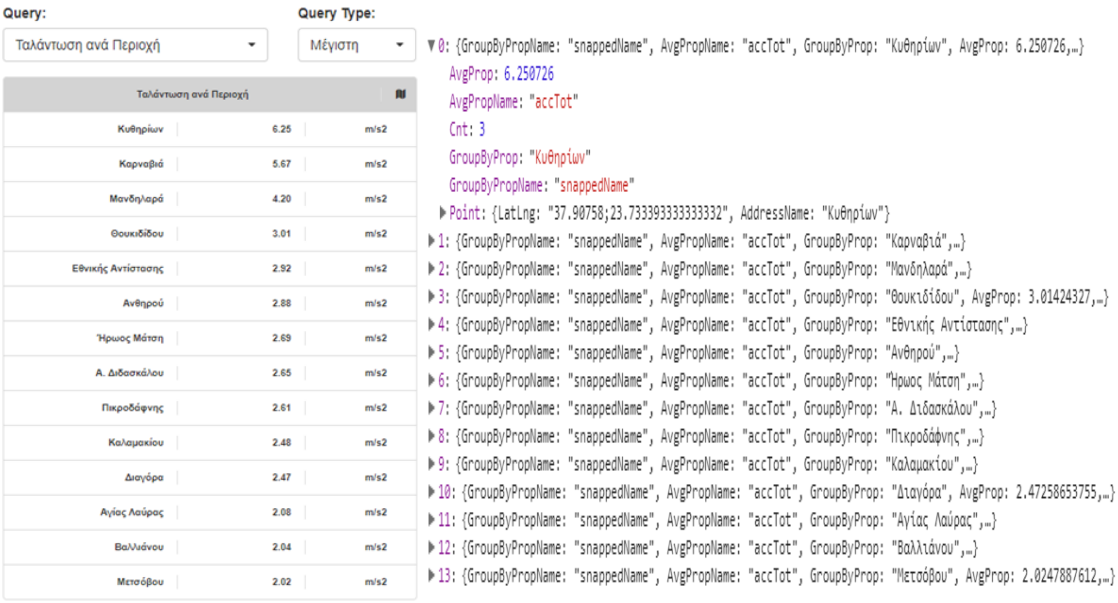


Εικόνα 55: Εμφάνιση αποτελέσματος μέγιστης μέσης τιμής στροφών κινητήρα ανά λεπτό ανά κατασκευαστή σε GUI και raw data.

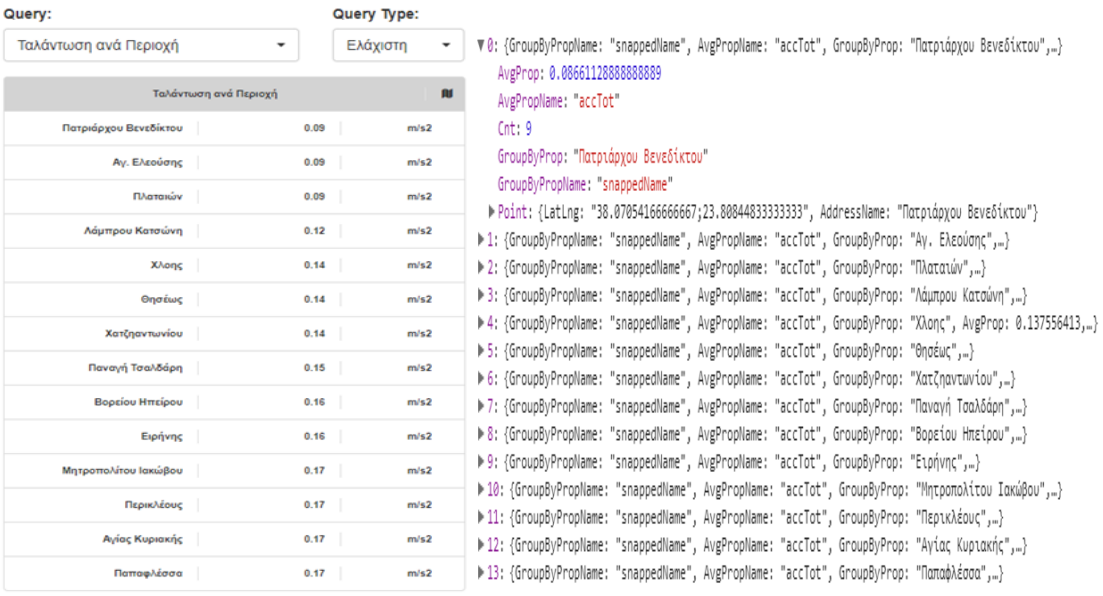


Εικόνα 56: Εμφάνιση αποτελέσματος ελάχιστης μέσης τιμής στροφών κινητήρα ανά λεπτό ανά κατασκευαστή σε GUI και raw data.

6. Μέγιστη και ελάχιστη ταλάντωσης accelerometer ανά περιοχή.

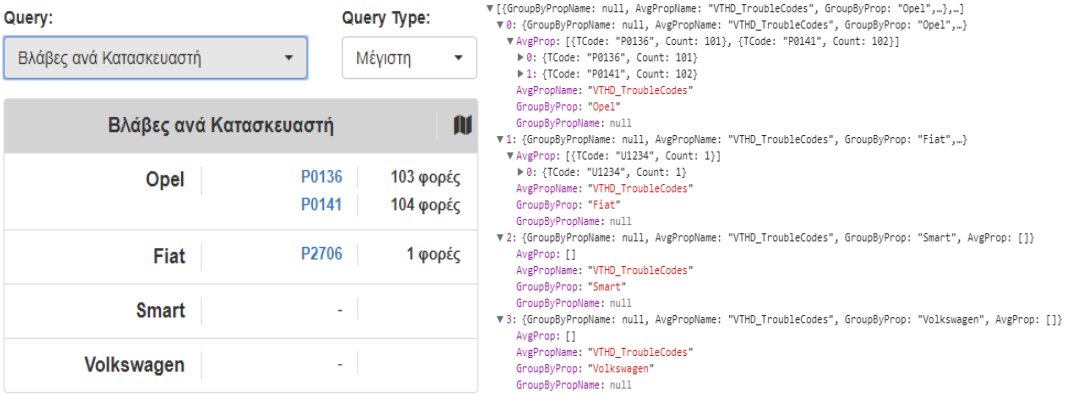


Εικόνα 57: Εμφάνιση αποτελέσματος μέγιστης ταλάντωσης ανά περιοχή σε GUI και raw data.

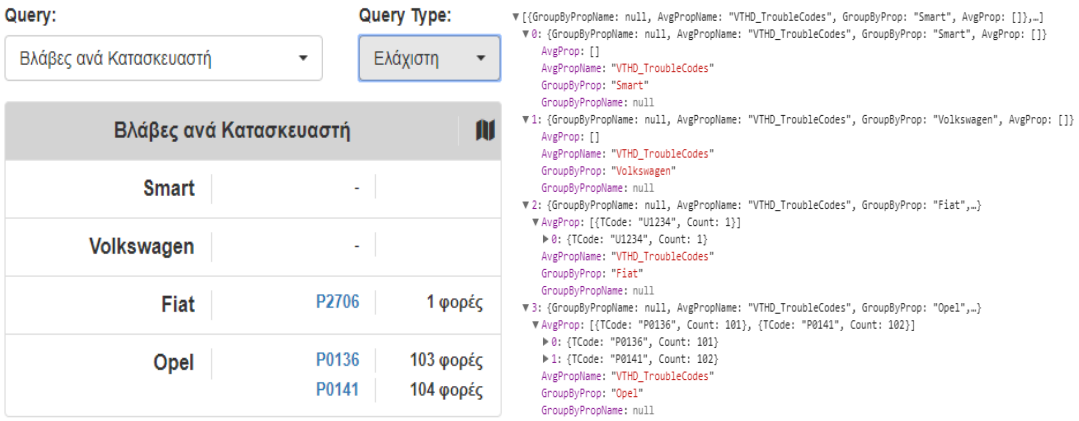


Εικόνα 58: Εμφάνιση αποτελέσματος μέγιστης ταλάντωσης ανά περιοχή σε GUI και raw data.

7. Μέγιστη και ελάχιστη τιμή βλαβών ανά κατασκευαστή.

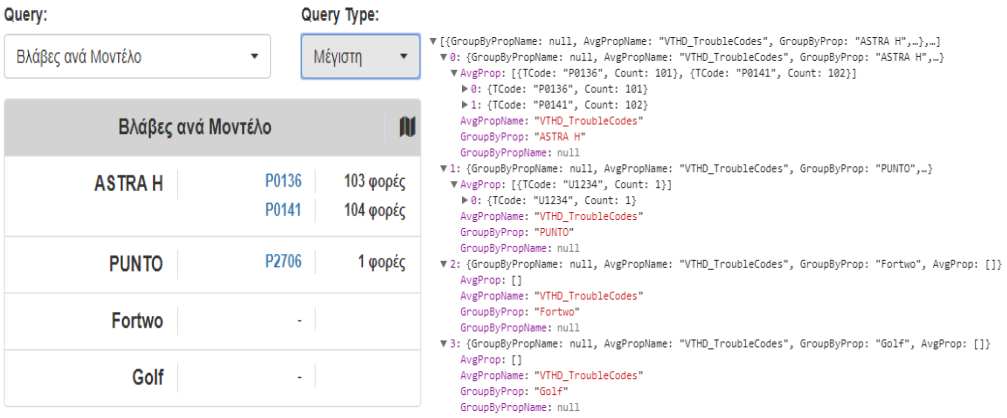


Εικόνα 59: Εμφάνιση αποτελέσματος μέγιστης τιμής βλαβών ανά κατασκευαστή σε GUI και raw data.

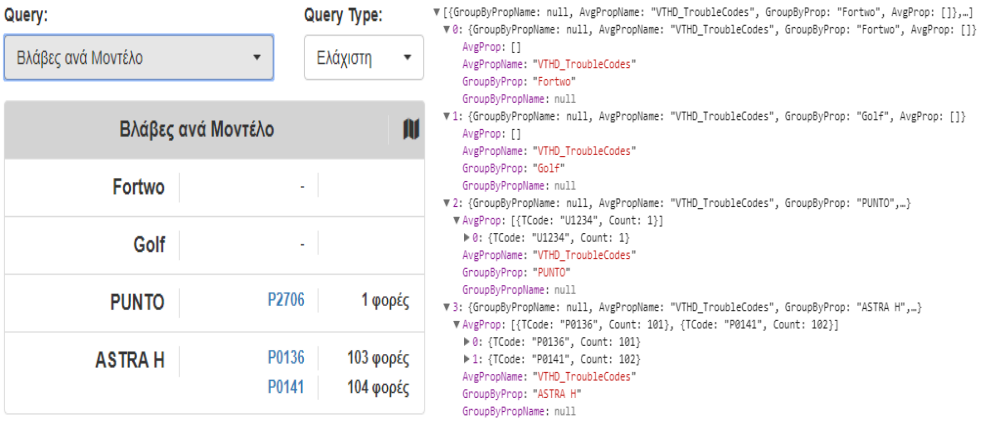


Εικόνα 60: Εμφάνιση αποτελέσματος μέγιστης τιμής βλαβών ανά κατασκευαστή σε GUI και raw data.

8. Μέγιστη και ελάχιστη μέση τιμή βλαβών ανά μοντέλο.



Εικόνα 61: Εμφάνιση αποτελέσματος μέγιστης τιμής βλαβών ανά μοντέλο σε GUI και raw data.



Εικόνα 62: Εμφάνιση αποτελέσματος ελάχιστης τιμής βλαβών ανά μοντέλο σε GUI και raw data.

Συμπεράσματα και προτάσεις για μελλοντικό έργο.

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας έγιναν αντιληπτές οι δυνατότητες που μπορούν να προσφέρουν τέτοιου είδους μελέτες. Καθώς η real-time βάση δεδομένων της Firebase δεν μπορεί να ανταποκριθεί σε complex queries ή σε queries που απαιτούνται απλές aggregation functions π.χ. sum, average, κ.λπ. όπως επίσης ούτε σε queries που περιέχουν αντικείμενα με πολλαπλά επίπεδα. Δυστυχώς για να εκτελεστούν οι εργασίες της συγκεκριμένης εργασίας, απαιτείται η ανάκτηση όλων των δεδομένων από την Firebase και η περαιτέρω επεξεργασία και το φιλτράρισμά τους από την εφαρμογή.

Αυτό πρακτικά σημαίνει πως για να αναλυθούν όλα τα δεδομένα των εγγραφών κάθε διαδρομής και να εξαχθούν χρήσιμα συμπεράσματα, θα πρέπει πρώτα να γίνουν «download» όλες οι εγγραφές, γεγονός που δυσκολεύει και καθυστερεί την διαδικασία και καθιστά την χρήση της Firebase όχι ιδανική λύση για τέτοιου είδους data. Παρόλα αυτά, η πλατφόρμα λειτουργεί αποδοτικά σε ένα ποσοστό 80% και μπορεί να παρουσιάσει μια καθυστέρηση μερικών λεπτών όταν ζητηθούν πιο περίπλοκα queries τα οποία αφορούν αποκλειστικά τα detail data των διαδρομών, όπως π.χ. η ανάλυση της ταλάντωσης ανά περιοχή (ή οδό).

Άλλο ένα θέμα ήταν το γενικά μικρό δείγμα των δεδομένων, τα queries επέστρεψαν λίγα αποτελέσματα κατά συνέπεια. Σαφέστατα η επανάληψη της διαδικασίας από έναν στόλο οχημάτων αλλά και από πολλές διαφορετικές κινητές συσκευές. Θα αναδεικνύονταν πολλές και σημαντικές διαφορές. Όπως και στο τωρινό παράδειγμα μόνο ένα αμάξι είχε βλάβη κρατημένη στην μνήμη του εγκεφάλου του. Για την ακρίβεια η βλάβη ήταν στους αισθητήρες λάμδα. [36]

Πολύ σημαντικό θα ήταν επίσης κάποιος να μπορέσει να ασχοληθεί και με την μεταφορά του πηγαίου κώδικα σε επίπεδο άλλης πλατφόρμας. Η ανάπτυξη του σε λειτουργικό iOS για συσκευές Apple αλλά και Windows Phone είναι από μόνο του μία πρόκληση καθώς και το κόστος του αναπτυξιακού αλλά η μεταγλώττιση θα είναι μία πολύπλοκη διαδικασία. Οι αλγόριθμοι επικοινωνίας αλλά και μεταφοράς των δεδομένων θα έπρεπε πιθανόν να γραφούν και σε μεταγενέστερες γλώσσες προγραμματισμού. Ακόμα και η μεταφορά της σε σχεσιακή βάση δεδομένων σε κάποια άλλη πλατφόρμα όπως MySQL θα χρειαστεί περαιτέρω μελέτη έτσι ώστε τα δεδομένα να παραμείνουν ανέπαφα και να μην χαθούν τιμές.

Άλλωστε σκοπός δεν θα ήταν να βοηθηθεί από την παραγόμενη πληροφορία μόνο ο απλός ιδιώτης αλλά και ο επαγγελματίας που έχει το όχημα ως μέσω επιβίωσης. Οι κατασκευαστές που μπορεί να έχουν μέρος αυτών των data δεν έχουν και τις αντίστοιχες μετρήσεις από τις κινητές συσκευές των καταναλωτών τους.

Παράρτημα – Λέξεις κλειδιά

Android	Το **Android** είναι ένα λειτουργικό σύστημα το οποίο βασιζόμενο στον πυρήνα του λειτουργικού linux, υποστηρίζει συσκευές κινητών τηλεφώνων, tablets ενώ προσφάτως ανακοινώθηκε και η υποστήριξη συσκευών αρχιτεκτονικής x86. Αναπτύχθηκε από την Google
API	(**Application Programming Interface**), όρος με τον οποίο συνήθως περιγράφεται μια έτοιμη βιβλιοθήκη η οποία περιέχει ρουτίνες, δομές δεδομένων, κλάσεις και μεταβλητές και μπορεί να χρησιμοποιηθεί ως πρόσθετο σε κάποιο λογισμικό.
Firebase	Σχετικά νέα πλατφόρμα της Google η οποία προσφέρει ένα ολοκληρωμένο πακέτο υπηρεσιών (database, hosting, storage, notifications, cloud, analytics, etc.) για την ανάπτυξη εφαρμογών, ιδιαίτερα για εφαρμογές κινητών συσκευών αλλά και web.
GPS	(**Global Positioning System**), Αναφέρεται στο Παγκόσμιο Σύστημα Στιγματοθέτησης μέσα από το οποίο μπορεί να εντοπιστεί η γεωγραφική θέση ενός κινητού ή ακίνητου σημείου.
GUI	(**Graphical User Interface**), Αποτελεί το γραφικό περιβάλλον μια εφαρμογής που χρησιμοποιείται για να αλληλοεπιδράσει με τον χρήστη
IDE	(**Integrated Development Environment**), είναι μία σουίτα λογισμικού που βοηθάει στην ανάπτυξη προγραμμάτων υπολογιστή. Συνήθως ένα IDE περιλαμβάνει κάποιον επεξεργαστή πηγαίου κώδικα, έναν μεταγλωττιστή, εργαλεία αυτόματης παραγωγής κώδικα, αποσφαλματωτή, συνδέτη, σύστημα ελέγχου εκδόσεων και εργαλεία κατασκευής γραφικών διασυνδέσεων χρήστη για τις υπό ανάπτυξη εφαρμογές.
JSON	(**JavaScript Object Notation**), είναι η μορφή αναπαράστασης δεδομένων η οποία παρουσιάζει απλές δομές σε αναγνώσιμη από τον άνθρωπο μορφή. Χρησιμοποιείται κυρίως για μετάδοση δεδομένων μέσω πρωτοκόλλων διαδικτύου.
NoSQL	(**Non SQL**) Βάση δεδομένων η οποία δεν εφαρμόζει σχεσιακούς κανόνες αποθήκευσης δεδομένων
REST	(**Representational State Transfer**), αποτελεί μία μορφή αρχιτεκτονικής λογισμικού για κατανεμημένα συστήματα όπως ο παγκόσμιος ιστός και βασίζεται κατά πολύ στο πρωτόκολλο HTTP.
SDK	(**Software Development Kit**), είναι ένα πακέτο Ανάπτυξης Λογισμικού το οποίο περιέχει ένα σύνολο εργαλείων ανάπτυξης που επιτρέπουν σε έναν προγραμματιστή να δημιουργήσει λογισμικό εφαρμογών για ένα συγκεκριμένο πακέτο λογισμικού, πλατφόρμα, παιχνιδομηχανή, λειτουργικά συστήματα, κ.λπ.
VIN	(**Vehicle Identification Number**), είναι ένας μοναδικός κωδικός πλαισίου ο οποίος χρησιμοποιείται για την αναγνώριση οχημάτων

Βιβλιογραφία:

1. https://en.wikipedia.org/wiki/Data_analysis
2. http://searchdatamanagement.techtarget.com/definition/data-analytics
3. https://en.wikipedia.org/wiki/Real-time_database
4. https://en.wikipedia.org/wiki/Big_data
5. https://www.shrm.org/resourcesandtools/hr-topics/risk-management/pages/top-database-security-threats.aspx
6. https://en.wikipedia.org/wiki/Firebase
7. https://leanpub.com/firebase-android/
8. https://firebase.google.com/docs/
9. https://firebase.google.com/docs/admob/admob-firebase
10. https://firebase.google.com/docs/analytics/
11. https://firebase.google.com/docs/app-indexing/
12. https://firebase.google.com/docs/auth/
13. https://firebase.google.com/docs/adwords/
14. https://firebase.google.com/docs/firestore/
15. https://firebase.google.com/docs/functions/
16. https://firebase.google.com/docs/cloud-messaging/
17. https://firebase.google.com/docs/storage/
18. https://firebase.google.com/docs/crash/
19. https://firebase.google.com/docs/dynamic-links/
20. https://firebase.google.com/docs/invites/
21. https://firebase.google.com/docs/hosting/
22. https://firebase.google.com/docs/perf-mon/
23. https://firebase.google.com/docs/database
24. https://firebase.google.com/docs/remote-config/
25. https://firebase.google.com/docs/test-lab
26. https://www.ijircce.com/upload/2016/september/133_Study.pdf
27. https://www.tutorialspoint.com/firebase/index.htm
28. https://cseweb.ucsd.edu/classes/wi16/cse110-a/applications/ln/cse110-discussion6.pdf
29. https://en.wikipedia.org/wiki/Mobile_phone_tracking
30. https://en.wikipedia.org/wiki/Comparison_of_web_map_services
31. https://www.mapbox.com/api-documentation/#map-matching
32. https://www.mapbox.com/api-documentation/#retrieve-a-match
33. https://www.mapbox.com/api-documentation/#match-response-object
34. https://www.mapbox.com/api-documentation/#map-matching-errors
35. https://developer.android.com/reference/android/hardware/SensorEvent.html#values
36. http://trouble-codes.com/