# ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ


ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗ


ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**Ανάλυση και Οπτικοποίηση Γεωχωρικών Δεδομένων με Χρήση του Django Framework**

**Geospatial Data Analytics and Visualization Using Django Framework**


**ΚΟΡΝΙΩΤΑΚΗΣ ΕΜΜΑΝΟΥΗΛ**

ΕΠΙΒΛΕΠΟΝΤΕΣ ΚΑΘΗΓΗΤΕΣ

**Β. Δαλάκας, Α. Τσαδήμας, Η. Βαρλάμης**



Αθήνα, 2019

Ο Κορνιωτάκης Εμμανουήλ

δηλώνω υπεύθυνα ότι:

1. Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και, απ' όσο γνωρίζω, η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλλει τα πνευματικά δικαιώματα τρίτων.

2. Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από την Ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο, καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

*Γνώσεσθε τὴν ἀλήθειαν,*
*καὶ ἡ ἀλήθεια*
*ἐλευθερώσει ὑμᾶς.*
*Ιω. 8,32*

# ΕΥΧΑΡΙΣΤΙΕΣ

Στο πολύτιμο αυτό ταξίδι της γνώσης στάθηκαν συμμέτοχοι και αρωγοί πολλοί άνθρωποι, χωρίς τη συμβολή των οποίων η διπλωματική αυτή δεν θα είχε πραγματοποιηθεί.

Αρχικά, θα ήθελα να ευχαριστήσω τον καθηγητή μου Δρ. Βασίλειο Δαλάκα, ο οποίος με την υπομονή του, τις συμβουλές του και την καθοδήγησή του συντέλεσε ουσιαστικά στην εκπόνηση της παρούσας εργασίας. Στο σημείο αυτό δε θα μπορούσα να μην αναφέρω πως το ένα μέρος (Frontend) το ενέταξε ο ίδιος στην εφαρμογή και υπό την καθοδήγησή του τελειοποιήθηκε. Παράλληλα, ευχαριστώ τον Δρ. Ανάργυρο Τσαδήμα, ο οποίος με τις γνώσεις του συνέβαλε σε καίρια σημεία και κατηύθυνε με τον τρόπο του την πορεία της εργασίας.

Στη συνέχεια, θα ήθελα να ευχαριστήσω τους γονείς μου και τα αδέρφια μου για τη συμπαράστασή τους, την υπομονή τους, αλλά και για την υλική τους στήριξη κατά τη διάρκεια των σπουδών μου.

Ευχαριστώ, επίσης, τη σύντροφό μου Εμμανουέλα για τη στήριξη, τη συμπαράσταση και την υπομονή της καθόλη τη διάρκεια των σπουδών καθώς και την κ. Μαριάννα για την πνευματική στήριξη και ενδυνάμωση. Τέλος, θα ήθελα να ευχαριστήσω θερμά τη σύντροφό μου Εμμανουέλα και την αγαπητή μου φίλη Κωνσταντίνα για την φιλολογική επιμέλεια της διπλωματικής.

# Περίληψη

Στην παρούσα διπλωματική εργασία παρουσιάζεται ο σχεδιασμός και η υλοποίηση μιας διαδικτυακής πλατφόρμας με τη χρήση του Django Framework, που έχει ως σκοπό την καλύτερη κατανόηση του τρόπου λειτουργίας του, όπως επίσης και την οπτικοποίηση των γεωχωρικών δεδομένων.

Οι μετρήσεις, που λαμβάνονται από τις κινητές συσκευές των χρηστών ως γεωχωρικά δεδομένα, παραχωρήθηκαν από τον Δρ. Βασίλειο Δαλάκα και βρίσκονται αποθηκευμένες σε μία βάση δεδομένων PostgreSQL.

Στη συνέχεια, γίνεται μία σύντομη παρουσίαση των κεφαλαίων, τα οποία περιγράφουν λεπτομερώς τη διαδικασία υλοποίησης της διαδικτυακής πλατφόρμας. Συγκεκριμένα, στο κεφάλαιο 1 περιγράφεται ο σκοπός για τον οποίο εκπονείται η διπλωματική αυτή εργασία.

Αμέσως μετά, στο κεφάλαιο 2, παρουσιάζονται οι τεχνολογίες, που χρησιμοποιήθηκαν και συνέβαλαν, ώστε να υλοποιηθεί η συγκεκριμένη εφαρμογή. Περιγράφονται, δηλαδή, η γλώσσα προγγραμματισμού Python, το Framework Django, όπως επίσης και οι τεχνολογίες από την πλευρά του χρήστη (Client side) και του server (Server side).

Στο κεφάλαιο 3, περιγράφονται αναλυτικά τα βήματα υλοποίησης της εφαρμογής, η εγκατάσταση της γλώσσας Python, καθώς και εκείνη των προαπαιτούμενων πακέτων. Επιπλέον, παρουσιάζεται ο τρόπος εγκατάστασης της βάσης δεδομένων PostgreSQL και της επέκτασης της PostGIS, όπως επίσης και το περιβάλλον ανάπτυξης Pycharm IDE.

Στο κεφάλαιο 4, γίνεται μία σύντομη αναφορά στα πιο σημαντικά πακέτα του Django, τα οποία εγκαταστάθηκαν και χρησιμοποιήθηκαν στην εφαρμογή.

Εν συνεχεία, στο κεφάλαιο 5, πραγματοποιείται μία σύντομη παρουσίαση της εφαρμογής, που επί της ουσίας είναι ο οδηγός χρήσης της.

Στο κεφάλαιο 6, παρατίθενται τα URLs της εφαρμογής, τα αποτελέσματα των οποίων εμφανίζονται στο παράρτημα Β σε μορφή JSON.

Τέλος, στο κεφάλαιο 7 παρουσιάζονται τα συμπεράσματα, τα οποία προέκυψαν κατά την υλοποίηση της εργασίας.

# Abstract

In this thesis, a description of the planning and implementation of a web platform by using Django Framework, with a view to the best comprehension of its functionality and the visualization of geospatial data, is made.

The geospatial data measurements –collected from mobile phones- were provided by Dr. Vassilios Dalakas and now are stored in a PostgreSQL database.

Afterwards, a brief presentation of the chapters follows. There, all the steps that have led to the implementation of the web platform, are described in detail.

Specifically, in chapter 1, the implementation purpose of the thesis is clarified.

After that, in chapter 2, there's a presentation of the technologies which have contributed to the existing application implementation.

More specifically, the Python programming language, Django Framework and the Client Side and Server Side technologies as well, are all analyzed in this chapter.

In chapter 3, a thorough description of the actualization steps of the app, Python language installation and also the pre-required packages is given. In addition, the PostgreSQL database and its extension PostGIS are presented and so is the Pycharm Integrated Development Environment (IDE).

In chapter 4, a brief reference to the most important Django packages, which were installed and used in the app, is made.

Then, in chapter 5, there's a mention concerning the app, which –in other words- constitutes the application's user guide.

In chapter 6, app's URLs, whose results appear in appendix B in JSON format, are quoted.

Finally, in chapter 7, both the conclusions –drawn during this thesis implementation- and future expansions are presented.

# Κατάλογος Εικόνων

# Πίνακας Περιεχομένων

# 1. Εισαγωγή

Στις μέρες μας, τα γεωχωρικά δεδομένα έχουν εισβάλλει στη ζωή μας με ποικίλους τρόπους. Ο πιο διαδεδομένος είναι οι χάρτες (Maps), όπου καθένας έχει τη δυνατότητα να ορίσει το σημείο (τοποθεσία), που βρίσκεται πάνω στο χάρτη, προκειμένου να πλοηγηθεί σε κάποιο διαφορετικό ή να βρει κάποιο κοντινό κατάστημα, που τον ενδιαφέρει, στην τοποθεσία που βρίσκεται. Επιπλέον, έχει τη δυνατότητα να δει την κυκλοφοριακή κίνηση στο δρόμο, έτσι ώστε να προγραμματίσει τον τρόπο μετάβασής του από ένα σημείο σε κάποιο άλλο. Όλα αυτά μπορεί κανείς να τα πραγματοποιήσει πλέον μέσω της κινητής συσκευής (smartphone, tablet, notebook κ.α.), που διαθέτει, χάρη στην εξέλιξη της τεχνολογίας.

Οι κινητές συσκευές διαθέτουν ενσωματωμένους αισθητήρες, όπως GPS (Global Positioning System), αξελερόμετρο (Accelerometer), γυροσκόπιο (Gyroscope) κ.α., που σε συνδυασμό με τα δεδομένα κινητής τηλεφωνίας (Mobile Broadband), όπως επίσης και τα ασύρματα δίκτυα (Wifi) δίνουν τη δυνατότητα στο χρήστη να εκμεταλλευτεί όλα τα παραπάνω. Αυτό έχει ως αποτέλεσμα οι πάροχοι των εκάστοτε εταιρειών τηλεπικοινωνίας να έχουν οικονομικό όφελος, λόγω του ότι η τοποθεσία, το γεωγραφικό μήκος και το γεωγραφικό πλάτος λαμβάνονται απευθείας από τις συσκευές των χρηστών. Επομένως, μία τέτοιου είδους πληροφορία αποκτά ουσία, όταν μπορεί να εμφανιστεί στην οθόνη της συσκευής του χρήστη [1].

Δεδομένα, όπως αυτά που προαναφέρθηκαν, μπορούν να εμφανιστούν στην οθόνη του χρήστη μέσω της χρήσης visualization στοιχείων, όπως είναι τα γραφήματα (charts), οι χάρτες (maps) και διάφορα άλλα μέσα οπτικοποίησης. Με τον τρόπο αυτό, το Data Visualization μετατρέπει την πολύπλοκη πληροφορία, που συλλέγεται από τη συσκευή του χρήστη, και την καθιστά προσβάσιμη, απλή και κατανοητή σε αυτόν. Αυτό έχει ως αποτέλεσμα, οι άνθρωποι τη σημερινή εποχή να είναι απολύτως εξοικειωμένοι με τέτοιου είδους έννοιες, αλλά και τεχνολογίες, διότι τις χρησιμοποιούν στην καθημερινότητά τους, πραγμα το οποίο εξυπηρετεί και τον τελικό σκοπό.

## 1.1 Σκοπός Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία αποτελεί προέκταση προηγούμενων διατριβών και έχει σκοπό την ανάπτυξη μιας διαδικτυακής εφαρμογής, χρησιμοποιώντας το Django Framework σε συνδυασμό με τη γλώσσα προγραμματισμού Python. Η προσπάθεια αυτή συνέβαλε στο να κατανοήσουμε καλύτερα τη λειτουργία του συγκεκριμένου Framework, καθώς και στο να εμβαθύνουμε σε αυτό, πράγμα που αποτελεί και το σκοπό της συγκεκριμένης εργασίας.

Στις προηγούμενες εργασίες χρησιμοποιήθηκαν διαφορετικές τεχνολογίες, αλλά και τεχνικές, όπως για παράδειγμα στις εργασίες του Νικολάου Κορωνιώτη και Αθανασίου Κουντουρά [2][3]. Σε αυτές δημιουργήθηκε μία εφαρμογή για έξυπνες κινητές συσκευές με τη χρήση του λειτουργικού Android και του λειτουργικού IOS αντίστοιχα, ενώ ως βάση συλλογής των δεδομένων χρησιμοποιήθηκε η MySQL. Έπειτα, ακολούθησαν ο Μωυσιάδης Αναστάσιος [4] και ο Gerti Nurka [5], οι οποίοι με τη σειρά τους επέκτειναν την ήδη υπάρχουσα εφαρμογή, χρησιμοποιώντας τεχνολογίες, όπως το λειτουργικό Android, καθώς και Java PrimeFaces. Τροποποιήθηκε, επίσης, η βάση MySQL και προστέθηκαν νέα πεδία σε αυτή, τα οποία αφορούσαν το χρήστη, την ακρίβεια μέτρησης, την ταχύτητα του δικτύου κ.α. Εν συνεχεία, στη διπλωματική του ο Λοντόρφος Νικόλαος [6] εφαρμόζοντας νέες τεχνολογίες στο Frontend, όπως είναι οι Angular2, Nodejs, HTML5 και Typescript, παρουσίασε με καλαίσθητο, αλλά και λειτουργικό τρόπο, τα δεδομένα στο χρήστη, χρησιμοποιώντας ένα REST API σε Spring, το οποίο επικοινωνεί με τη βάση (PostgreSQL) και υλοποιήθηκε από τον καθηγητή Δρ. Βασίλειο Δαλάκα.

Πάνω σε αυτές τις διπλωματικές εργασίες έρχεται με τη σειρά της να προστεθεί και η παρούσα, η οποία αποτελεί επέκταση των προαναφερθέντων εργασιών, εφαρμόζοντας νέες τεχνολογίες αιχμής. Βασιζόμενοι στη διπλωματική του Νικόλαου Λοντόρφου, μετατρέψαμε το Spring REST API σε ένα αντίστοιχο RESTfull API χρησιμοποιώντας το Django REST Framework.

Σχετικά τώρα με την πλευρά του χρήστη εφαρμόστηκε το Bootstrap Framework, καθώς και η βιβλιοθήκη Chartjs.

Επομένως, η συγκεκριμένη εφαρμογή στοχεύει στο να παρουσιάσει στον χρήστη με τρόπο όμορφο (αισθητικά), απλό και κατανοητό τα αποτελέσματα μιας σειράς μετρήσεων των τηλεπικοινωνιακών παρόχων, που βρίσκονται στην ελληνική επικράτεια.

# 2 Τεχνολογίες

## 2.1 Python - Django

PYTHON

Τη λίστα των γλωσσών προγραμματισμού υψηλού επιπέδου, όπως είναι η Java, C++, C#, Lisp κι άλλες, έρχεται να συμπληρώσει η Python. Η δημιουργία της ξεκίνησε το 1980 από τον Ολλανδό Guido van Rossum και η κυκλοφορία της πραγματοποιήθηκε το 1991. Πρόκειται για μία γλώσσα προγραμματισμού γενικού σκοπού, η οποία είναι εύκολη σε εκμάθηση, και γι' αυτό το λόγο καθίσταται ιδανική, για να κάνει κάποιος τα πρώτα του βήματα στο χώρο του προγραμματισμού. Ο σκοπός του δημιουργού της είναι να την καταστήσει ευχάριστη στους χρήστες, που θα την επιλέξουν, και γι' αυτό πήρε το όνομά της από τους γνωστούς Βρετανούς κωμικούς Monty Pythons.
Οι τρόποι, που απαιτούνται, ώστε ένα πρόγραμμα να μετατραπεί σε γλώσσα μηχανής υψηλού επιπέδου με σκοπό να γίνει κατανοητή και εκτελέσιμη από έναν ηλεκτρονικό υπολογιστή, είναι δύο. Πρώτο τρόπο αποτελεί η διαδικασία της **μεταγλώττισης** (compiling), κατά την οποία ο compiler αναλαμβάνει τον πηγαίο κώδικα (source code) και παράγει ένα πρόγραμμα (executable), το οποίο μπορούμε να εκτελέσουμε στη συνέχεια. Ως δεύτερος τρόπος ορίζεται η διαδικασία της **διερμηνείας** (interpreted), κατά την οποία ο interpreter μεταφράζει και εκτελεί το πρόγραμμα απευθείας, χωρίς να χρειαστεί πρώτα να παράξει κάποιο εκτελέσιμο αρχείο [7].

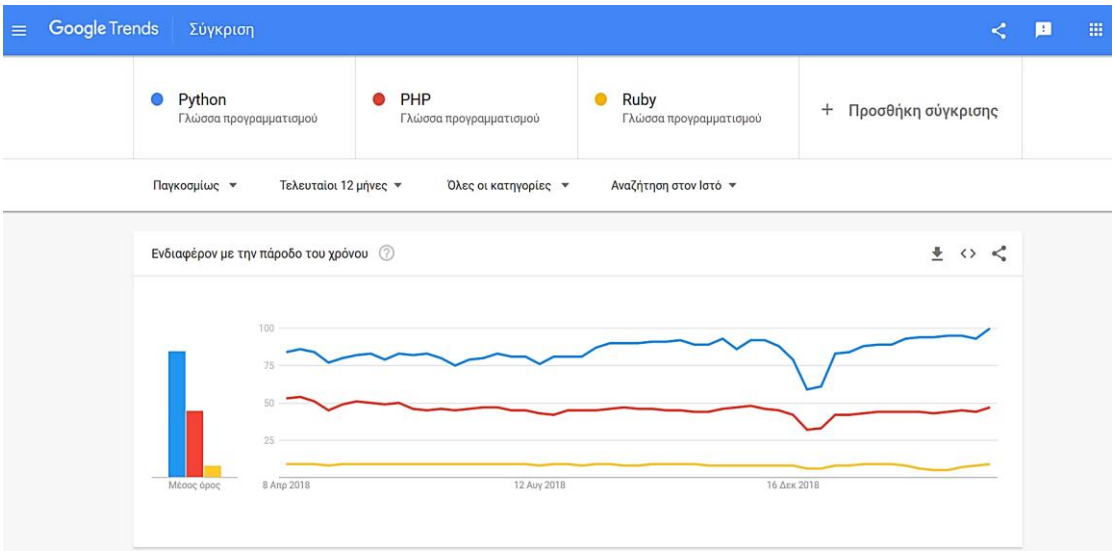
### 2.1.1 Βασικά Χαρακτηριστικά της Python

Η Python, επομένως, χρησιμοποιεί τη δεύτερη διαδικασία, όπως αυτή περιγράφηκε παραπάνω, για να μετατρέψει ένα πρόγραμμα σε γλώσσα μηχανής. Είναι, δηλαδή, μία **interpreted** γλώσσα προγραμματισμού.
Αυτό αποτελεί ένα από τα βασικά χαρακτηριστικά της, το οποίο την καθιστά γρήγορη, αλλά και ευέλικτη σε σχέση με άλλες γλώσσες προγραμματισμού.

Ταυτόχρονα, είναι μία αντικειμενοστραφής (**Object Oriented**) γλώσσα προγραμματισμού, είναι ανοικτού κώδικα (**Open Source**), πράγμα το οποίο σημαίνει πως μπορεί κανείς να την κατεβάσει ελεύθερα στον υπολογιστή του και να την εγκαταστήσει σε οποιοδήποτε λειτουργικό σύστημα επιθυμεί (Linux, Mac OS, Windows).

Η Python, επίσης, είναι ανεξάρτητη πλατφόρμας (**Platform Independent**). Αυτό σημαίνει πως ο κώδικας κάποιου προγράμματός της μπορεί να εκτελεστεί σε ποικίλα συστήματα, όπως για παράδειγμα στη γνωστή παιχνιδομηχανή Playstation. Χαρακτηρίζεται μάλιστα από τον δυναμικό τύπο του συστήματός της, καθώς και από την αυτόματη διαχείριση μνήμης την οποία διαθέτει [8].

Αυτά είναι μερικά από τα βασικά χαρακτηριστικά της γλώσσας προγραμματισμού Python. Περισσότερες λεπτομέρειες για τη γλώσσα, τη λειτουργικότητά της, καθώς και τις βιβλιοθήκες, που διαθέτει, είναι διαθέσιμες στον ιστότοπο **https://www.python.org/** .




Εικόνα 1

Στην εικόνα 1 εμφανίζονται διαγράμματα σύγκρισης επιλεγμένων γλωσσών προγραμματισμού ανοιχτού κώδικα. Συγκεκριμένα, τον τελευταίο χρόνο, παρατηρείται παγκοσμίως πως η γλώσσα Python χρησιμοποιείται κατά κόρον σε πολλούς τομείς (Data Mining, AI, Analytics, Telecommunications) με αποτέλεσμα να βρίσκεται στην πρώτη θέση σε σύγκριση με τις άλλες δύο γλώσσες PHP και Ruby.

## DJANGO

Το Django είναι ένα πανίσχυρο Web Framework, το οποίο βασίζεται στη γλώσσα προγραμματισμού Python. Η ονομασία του, την οποία έδωσαν οι δημιουργοί του Adrian Holovaty και Simon Willison το έτος 2005, προέρχεται από τον Βέλγο - Γάλλο κιθαρίστα Django Reinhardt. Αξίζει να αναφερθεί πως το Django Framework ξεκίνησε αρχικά σαν project, που δημοσιεύθηκε στην εφημερίδα Lawrence Journal-World στο Kansas των Ηνωμένων Πολιτειών [9].

Όπως τα σύγχρονα Frameworks χρησιμοποιούν την αρχιτεκτονική του λεγόμενου MVC (**Model** - **View** - **Controller**) ή MTV (**Model** - **Template** - **View**) σχήματος, έτσι και το Django βασίζεται στη συγκεκριμένη διάταξη. Παρακάτω δίνεται η περιγραφή των τριών μερών από τα οποία αποτελείται το συγκεκριμένο Framework:

- **Model**: Πρόκειται για την αναπαράσταση, αλλά και τη συσχέτιση των δεδομένων, που έχουν δηλωθεί ως κλάσεις στη γλώσσα Python και βρίσκονται στη βάση δεδομένων.

  Το Django κάνει χρήση της τεχνικής ORM (**Object Relational Mapper**) και έχει την ιδιότητα να μετατρέπει τα δεδομένα μεταξύ δύο μη συμβατών συστημάτων, χρησιμοποιώντας αντικειμενοστραφείς γλώσσες προγραμματισμού.
- **View**: Η αναπαράσταση των δεδομένων γίνεται σε μία HTML σελίδα, η οποία κάνει χρήση ενός ορισμένου, στη γλώσσα Python, κώδικα. Στην ουσία πρόκειται για το μέρος που αφορά στο χρήστη, αλλά και τη δυνατότητα που εκείνος αποκτά, ώστε να αλληλεπιδρά με το πρόγραμμα περιήγησης (Web Browser).
- **Controller**: Αποτελεί το πιο βασικό κομμάτι του συστήματος, καθώς είναι υπεύθυνος για τη διαχείριση της πληροφορίας, η οποία ανακτάται από τη βάση δεδομένων μέσω του Model, αλλά και για την πληροφορία που εμφανίζεται στην οθόνη μέσω του View [10]. Έχει, δηλαδή, την ικανότητα να διαχειρίζεται απαντήσεις κι ερωτήματα (**Requests and Responses**), καθώς και να δημιουργεί τη σύνδεση με τη βάση δεδομένων.

### 2.1.2 Κύρια χαρακτηριστικά του Django Framework

Ένα από τα κυριότερα χαρακτηριστικά, που διαθέτει το συγκεκριμένο Framework και το κάνει να ξεχωρίζει είναι η σελίδα της διεπαφής του διαχειριστή (**Admin Interface**). Μέσω της σελίδας αυτής παρέχεται η δυνατότητα στον χρήστη να τροποποιήσει έναν ήδη υπάρχοντα λογαριασμό (**User Account**), όπως και το να έρθει σε επαφή με τα μοντέλα (**Models**), τα οποία έχουν δημιουργηθεί στη βάση δεδομένων. Αυτό επιτυγχάνεται όταν το Django χρησιμοποιεί την κλάση, την οποία έχουμε δηλώσει στη γλώσσα Python μέσα στο αρχείο **admin.py**.

Παράλληλα, το Django περιλαμβάνει έναν πολύ ελαφρύ και συνάμα ταχύτατο **Web Server** για την ανάπτυξη (**Development**) μιας εφαρμογής, καθώς και για τον έλεγχο (**Testing**) αυτής [11]. Ακόμα, ένα χαρακτηριστικό, το οποίο θα μπορούσαμε να αναφέρουμε, είναι πως το Django διαθέτει κρυφή μνήμη (**Cache**). Αλλά τι σημαίνει αυτό;

Κάθε φορά που κάποιος χρήστης ζητάει μία σελίδα (Request), ο Web Server, από τον οποίο ο χρήστης τη ζητάει, πραγματοποιεί κάποιους υπολογισμούς (π.χ. ερωτήματα στη βάση δεδομένων, επιχειρηματική λογική κ.α.).

Με τον τρόπο αυτό αποφεύγεται η επανεκτέλεση ενός τέτοιου είδους υπολογισμού και το αποτέλεσμα αποθηκεύεται στη μνήμη Cache, επιστρέφοντας την απάντηση στο χρήστη σε σύντομο χρονικό διάστημα [12].

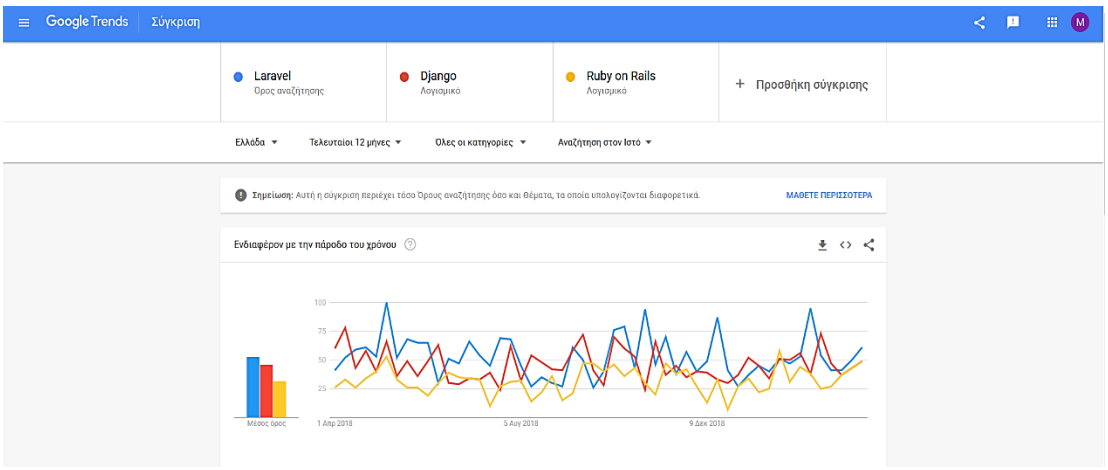
Επομένως, θα μπορούσε κανείς να υποστηρίξει πως το Django Framework είναι:

- **Γρήγορο**: Ο τρόπος, με τον οποίο έχει φτιαχτεί, εξυπηρετεί τις ανάγκες ακόμα και του πιο απαιτητικού προγραμματιστή (Web Developer) εύκολα και σε σύντομο χρονικό διάστημα.
- **Ασφαλές**: Δίνει τη δυνατότητα στον προγραμματιστή να κάνει Deploy την εφαρμογή του (Website) με τη «σιγουριά» που προσφέρει το HTTPS. Τα templates του Django έρχονται, επίσης, ενσωματωμένα με τους λεγόμενους χαρακτήρες διαφυγής (Escape Specific Characters), προστατεύοντας έτσι το χρήστη από XSS attack (**Cross Site Scripting**). Ακόμα, το Framework αυτό έχει την ιδιότητα να προστατεύει τους χρήστες του από την τεχνική **SQL Injection** [13].

- **Επεκτάσιμο**: Οποιαδήποτε προσθήκη ή αλλαγή κι αν κάνει κάποιος, όσον αφορά το κομμάτι του Hardware (π.χ. Web/Application Server), αυτή δεν επηρεάζει το συγκεκριμένο Framework [14].

Αξίζει, επιπλέον, να αναφερθεί ότι δεν είναι τυχαίο το γεγονός πως παγκοσμίου φήμης ιστότοποι, όπως είναι οι: Instagram, Pinterest, OpenStack, National Geographic, Mozilla κ.α., χρησιμοποιούν το συγκεκριμένο Framework [15].

Πληροφορίες σχετικα με το Django Framework είναι διαθέσιμες στον ιστότοπο https://www.djangoproject.com/, όπου είναι δυνατόν κανείς να γνωρίσει καλύτερα το συγκεκριμένο Framework, αλλά και να εμβαθύνει σε αυτό, φτιάχνοντας μέσα σε λίγα μόλις λεπτά το πρώτο του project όπως θα δούμε στη συνέχεια.




Εικόνα 2

Στην παραπάνω εικόνα παρατηρείται πως το Django Framework βρίσκεται παγκοσμίως στη δεύτερη θέση με ελάχιστη διαφορά από το πρώτο, που είναι το Laravel.

## 2.2 Τεχνολογίες Client Side

### 2.2.1 Django Template

Το ρόλο, για το πως θα εμφανίζεται η σελίδα στον εκάστοτε χρήστη, τον έχει αναλάβει το Django Template Language (είναι γνωστό κι ώς Jinja2) σε συνεργασία με το Framework Bootstrap 4, αλλά και το Material Kit. Για την ακρίβεια, το template είναι ένα απλό αρχείο σε μορφή text με την ιδιότητα να εμφανίζει δυναμικά μία HTML σελίδα. Αυτό αποδίδει στις αντίστοιχες μεταβλητές τις ανάλογες τιμές τους, προβάλλοντας το τελικό αποτέλεσμα στην έξοδο. Συνεπώς, ένα Django Template διαθέτει τέσσερις κατηγορίες, που συμβάλλουν στη σωστή δόμησή του. Αυτές αναφέρονται ακολούθως ονομαστικά:

- **Variables** - **{{ }}**
- **Tags** - **{% %}**
- **Filters** - **|**
- **Comments** - **{# #}**

### 2.2.2 Bootstrap

Το Bootstrap είναι ένα από τα πιο δημοφιλή και πανίσχυρα Frontend Frameworks που υπάρχουν στις μέρες μας. Δημιουργήθηκε από τους Mark Otto, Jacob Thornton και μία μικρή ομάδα προγραμματιστών και πρωτοκυκλοφόρησε στις 19 Αυγούστου του 2011. Η πρωταρχική του ονομασία ήταν Twitter Blueprint, καθώς βρισκόταν υπό ανάπτυξη για την υποστήριξη κάποιων εργαλείων της πλατφόρμας του Twitter. Στη συνέχεια, μετονομάστηκε σε Bootstrap και φέρει το όνομα αυτό μέχρι και σήμερα. Πρόκειται για ένα Framework ανοιχτού κώδικα (Open Source), με το οποίο μπορεί κανείς να αναπτύξει web εφαρμογές, καθώς και Web Sites (ιστοτόπους), χρησιμοποιώντας τεχνολογίες όπως HTML, CSS και JS (JavaScript). Διαθέτει έτοιμα Templates (πρότυπα) με πολύ όμορφα -αισθητικά- κουμπιά (Buttons), φόρμες (Forms), τυπογραφία, όπως επίσης και μοντέρνες διεπαφές πλοήγησης (Navigation). Υποστηρίζεται από όλους τους Browsers (φυλλομετρητές), όπως είναι οι Mozilla Firefox, Google Chrome, Internet Explorer, Opera και Safari [16].

## 2.3 Τεχνολογίες Server Side

### 2.3.1 Django Rest Framework (DRF)


Πρόκειται για ένα υψηλού επιπέδου Web Framework, που είναι γραμμένο σε γλώσσα Python και συνεισφέρει στη δημιουργία των RESTfull Web APIs. Είναι βασισμένο στο Representation State Transfer (REST), που ανέπτυξε ο Roy Fielding στη διδακτορική διατριβή του το έτος 2000. Το REST ακολουθεί μία διαφορετική αρχιτεκτονική λογισμικού, προκειμένου να δημιουργήσει ένα Web Service, χρησιμοποιώντας ένα σύνολο κανόνων, που έχει ορίσει το ίδιο [17]. Το DRF διαθέτει ποικίλους τρόπους, με τους οποίους μπορεί κανείς να δημιουργήσει ένα Web API σε πολύ σύντομο χρονικό διάστημα (μόλις λίγα λεπτά), ακολουθώντας τις οδηγίες, που περιγράφονται στον αντίστοιχο ιστότοπο https://www.django-rest-framework.org/ . Το γεγονός αυτό το καθιστά επεκτάσιμο και φιλικό προς τον εκάστοτε χρήστη – προγραμματιστή, που επιθυμεί να ασχοληθεί αλλά και να εντρυφήσει σε αυτό. Για το σκοπό της διπλωματικής αυτής εργασίας το Django Rest Framework χρησιμοποιήθηκε κυρίως στο Backend, επικοινωνώντας με τη βάση δεδομένων.

Ακολούθως αναφέρονται επιγραμματικά ορισμένες από τις δυνατότητες που διαθέτει το συγκεκριμένο Framework, καθώς επίσης κι ένα απλό παράδειγμα για τη δημιουργία ενός API.

**Ιδιότητες Django Rest Framework**

1. Serialization
2. Requests and Responses
3. Class Based Views
4. Authentication and Permissions
5. Relationships and Hyperlinked APIs
6. Viewsets and Routers
7. Schemas and Client Libraries

Αρχικά, πρέπει να δημιουργηθεί ένα καινούργιο αρχείο με την ονομασία **serializers.py** μέσα στο φάκελο της εφαρμογής (π.χ. **myapp/**). Στη συνέχεια ορίζεται μία **Serializer Class** μέσα σε αυτό, περιλαμβάνοντας τα αντίστοιχα πεδία του model (αρχείο model.py), που χρειάζεται να εμφανιστούν στην οθόνη. Έπειτα, στο αρχείο views.py δίνεται η δυνατότητα στον χρήστη να εφαρμόσει τις κατάλληλες CRUD εντολές (Create, Read, Update, Delete) καλώντας τον αντίστοιχο Serializer. Στο σημείο αυτό αξίζει να σημειωθεί πως στα πλαίσια της διπλωματικής χρησιμοποιήθηκαν κυρίως **ModelViewSet** κλάσεις, για την αναπαράσταση των Models, οι οποίες κληρονομούν τη λειτουργικότητά τους από τις **GenericAPIView**. Τέλος, μέσα στο αρχείο **urls.py** ορίζεται το κατάλληλο path, ώστε να εμφανίζονται στην οθόνη τα αποτελέσματα [18].

### 2.3.2 PostgreSQL - PostGIS


Το πιο σημαντικό κομμάτι, από το οποίο αποτελείται μία εφαρμογή, θα μπορούσε να πει κανείς πως είναι η βάση δεδομένων. Σε αυτή βρίσκονται τα δεδομένα, που προβάλλονται στην πλευρά του χρήστη, και γι' αυτήν είναι υπεύθυνος ο εκάστοτε προγραμματιστής. Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε η βάση δεδομένων **PostgreSQL**.

Πρωτοκυκλοφόρησε στις 29 Ιανουαρίου του 1997 από μία ομάδα προγραμματιστών κι εθελοντών με αρχική έκδοση (version 6.0) και ακολουθεί τη φιλοσοφία ανοικτού λογισμικού (**Free and Open Source Software**). Πρόκειται για μία ολοκληρωμένη αντικειμενοστραφή σχεσιακή βάση δεδομένων (**ORDBMS**), όπως π.χ. (Boolean, Character, Enum).

Ως προέκταση της **PostgreSQL** χρησιμοποιήθηκε η **PostGIS**, λόγω του ότι υποστηρίζει γεωγραφικά δεδομένα, πάνω στα οποία στηρίζεται και η εφαρμογή που υλοποιήθηκε.

Στην εικόνα που ακολουθεί εμφανίζεται το ER (**Entity Relationship Diagram**) διάγραμμα, καθώς και οι πίνακες που έχουν δημιουργηθεί στη βάση δεδμένων [19].

**ER Diagram**


Εικόνα 3

Στο σημείο αυτό είναι σημαντικό να επισημανθεί πως, για τη δημιουργία των ερωτημάτων (Queries) στη βάση, χρησιμοποιήθηκαν αυτά που υπάρχουν στη διπλωματική του Νικολάου Λοντόρφου, με τη διαφορά πως μετατράπηκαν αντιστοίχως στη γλώσσα **Python** (**παράρτημα Α**).

# 3 Υλοποίηση Εφαρμογής (Web Application)

## 3.1 Εγκατάσταση Django Framework

### 3.1.1 Εγκατάσταση Python - pip

Αρχικά, κρίνεται απαραίτητο να τονιστεί ότι για να εγκαταστήσει κανείς το συγκεκριμένο Web Framework θα πρέπει πρώτα να εγκαταστήσει τη γλώσσα Python. Η συγκεκριμένη εφαρμογή αναπτύχθηκε σε Linux περιβάλλον και για το λόγο αυτό δεν χρειάστηκε να γίνει εκ νέου εγκατάστασή της, διότι έρχεται προεγκατεστημένη στα περισσότερα συστήματα Linux.

Για την εγκατάσταση της Python σε άλλα λειτουργικά συστήματα (π.χ. Windows), μπορεί κανείς να επισκεφθεί τον παρακάτω ιστότοπο: https://www.python.org/downloads/.

Η έκδοση της Python που χρησιμοποιήθηκε για τη συγκεκριμένη εφαρμογή είναι η **3.7.1** .

Εν συνεχεία, χρειάζεται να γίνει εγκατάσταση ενός ακόμα εργαλείου, του οποίου την επίσημη έκδοση μπορεί να βρει κανείς εδώ: https://pip.pypa.io/en/stable/. Πρόκειται για το γνωστό διαχειριστή πακέτων pip (**Pip Installs Packages**), ο οποίος έχει την ιδιότητα να εγκαθιστά αλλά και να αφαιρεί πακέτα τα οποία έχουν γραφτεί στη γλώσσα Python [20] . Οι εντολές που εφαρμόστηκαν για την εγκατάστασή του είναι :

- **curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py**
- **python get-pip.py**

### 3.1.2 Εγκατάσταση του Virtualenv

Εφόσον έχουν, πλέον, εγκατασταθεί τα προηγούμενα, είναι έτοιμος κανείς να δημιουργήσει ένα εικονικό περιβάλλον χρησιμοποιώντας το Virtualenv (**Virtual Environment**).

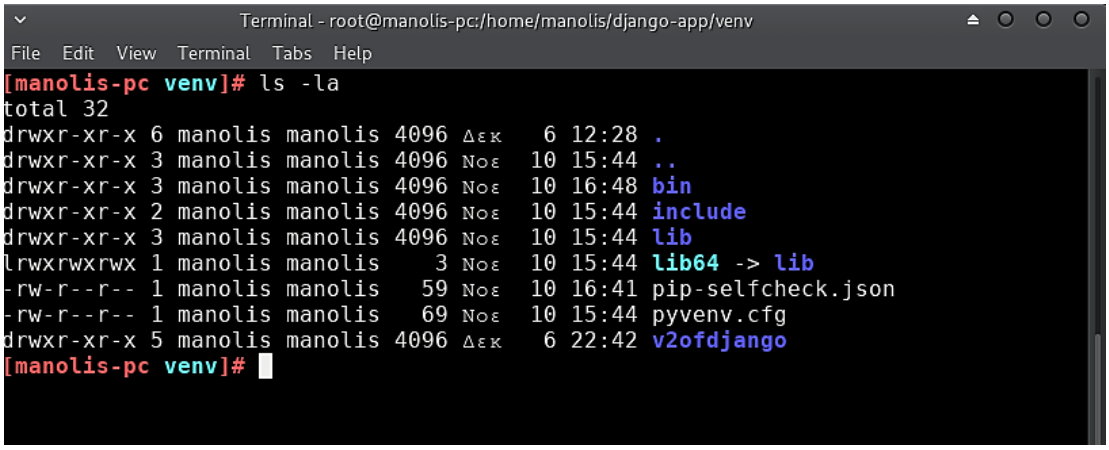
Το συγκεκριμένο εργαλείο έχει την δυνατότητα να δημιουργεί εικονικά περιβάλλοντα, τα οποία είναι μεμονωμένα (**Isolated**), χωρίς δηλαδή να επηρεάζει το

ένα το άλλο και που το καθένα θα έχει τα δικά του πακέτα, εξαρτήσεις (**Dependencies**) αντίστοιχα.

Οι εντολές είναι οι εξής :

- **apt-get install python-virtualenv**
- **$ [sudo] pip install virtualenv**
- **python3 -m venv  venv ή**
- **virtualenv -p /usr/bin/python2.7 ven**v

Οι δύο τελευταίες εντολές δημιουργούν ένα φάκελο (**Directory**) με την ονομασία **venv**, μέσα στον οποίο έχει εγκατασταθεί ένα εικονικό περιβάλλον με τις αντίστοιχες βιβλιοθήκες της Python, όπως φαίνεται στην ακόλουθη εικόνα.




Εικόνα 4

Στη συνέχεια, εφόσον πρώτα γίνει έλεγχος ότι ο χρήστης βρίσκεται μέσα στο σωστό φάκελο, γράφοντας στο τερματικό την εντολή :

**→ cd venv**

έχει τη δυνατότητα να ενεργοποιήσει το εικονικό περιβάλλον χρησιμοποιώντας την εντολή

**→ source bin/activate**

Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί πως η ενεργοποίηση του εικονικού περιβάλλοντος **δεν** καθίσταται υποχρεωτική, αλλά πραγματοποιείται από τον εκάστοτε χρήστη με σκοπό να τον διευκολύνει στην εκτέλεση των scripts, τα οποία

είναι γραμμένα στη γλώσσα Python. Αυτό σημαίνει πως αμέσως μετά την ενεργοποίηση του περιβάλλοντος, σαν **Environmental Path** ορίζεται αυτομάτως το μονοπάτι-διαδρομή του συγκεκριμένου εικονικού περιβάλλοντος μέσα από το οποίο γίνεται η ενεργοποίηση [21]. Ο χρήστης, εφόσον, το επιθυμεί μπορεί να απενεργοποιήσει το εικονικό περιβάλλον χρησιμοποιώντας την εντολή

→ **deactivate**

σε οποιαδήποτε διαδρομή-μονοπάτι κι αν βρίσκεται.

Κατά συνέπεια, πραγματοποιείται η εγκατάσταση του Django χρησιμοποιώντας την ακόλουθη εντολή:

→ **pip install Django**

Με αυτή την εντολή εγκαθίσταται η τελευταία έκδοση του Framework, η οποία είναι διαθέσιμη εκείνη τη χρονική στιγμή. Επιπλέον, μπορεί κανείς, για παράδειγμα, να εγκαταστήσει την έκδοση **2.0.7** του Django (την οποία τρέχει η εφαρμογή που φτιάξαμε) χρησιμοποιώντας την εντολή

→ **pip install -U Django==2.0.7** .

Αξίζει, επίσης, να επισημανθεί πως χάρη στη βοήθεια του διαχειριστή πακέτων pip δίνεται η δυνατότητα στο χρήστη να δημιουργήσει ένα αρχείο (**text file**) με την ονομασία **requirements.txt** με όλα τα πακέτα, αλλά και τις εξαρτήσεις (dependencies), τα οποία έχει εγκατεστημένα στην εφαρμογή του.

Αυτό επιτυγχάνεται με την εντολή:

→ **pip freeze > requirements.txt**

Έπειτα, μπορεί να εγκαταστήσει ξανά τα συγκεκριμένα πακέτα σε κάποιο άλλο λειτουργικό σύστημα ή ακόμα και να τα μοιραστεί με άλλους προγραμματιστές στην ίδια μορφή αρχείου, αρκεί μόνο να πληκτρολογήσει στο τερματικό την εντολή:

→ **pip install -r requirements.txt**.

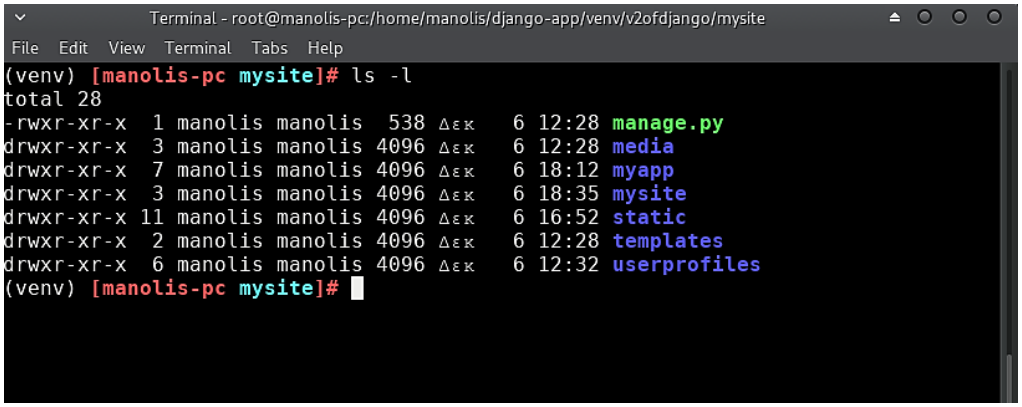
### 3.1.3 Δημιουργία ενός νέου project

Με την προϋπόθεση πως έχουν πραγματοποιηθεί τα προαναφερθέντα βήματα, σειρά τώρα παίρνει η δημιουργία ενός νέου project. Σε αυτό το σημείο ο χρήστης έχει

τη δυνατότητα να δημιουργήσει το δικό του project με οποιαδήποτε ονομασία επιθυμεί πληκτρολογώντας στο τερματικό την εντολή

**→ django-admin startproject mysite**

(για το σκοπό της διπλωματικής χρησιμοποιήθηκε η ονομασία φακέλου **mysite**). Ο φάκελος αυτός δημιουργείται μέσα στο ενεργοποιημένο πλέον εικονικό περιβάλλον του χρήστη, όπως φαίνεται στην εικόνα που ακολουθεί.




Εικόνα 5

Πληκτρολογώντας την προηγούμενη εντολή δημιουργούνται δύο φάκελοι με την ονομασία **mysite**. Ο εξωτερικός φάκελος μπορεί να λάβει οποιαδήποτε ονομασία επιθυμεί ο χρήστης, ενώ ο εσωτερικός φάκελος συμβολίζει το project που δημιουργήθηκε. Χρήσιμο είναι, επίσης, να αναφερθεί πως, όταν το εικονικό περιβάλλον είναι ενεργοποιημένο, εμφανίζεται στην κονσόλα μπροστά από το hostname το όνομα του περιβάλλοντος μέσα σε παρενθέσεις. Αυτό γίνεται για να μπορεί ο χρήστης εύκολα, ρίχνοντας μία γρήγορη ματιά, να διακρίνει αν έχει ενεργοποιήσει ή όχι το περιβάλλον του.

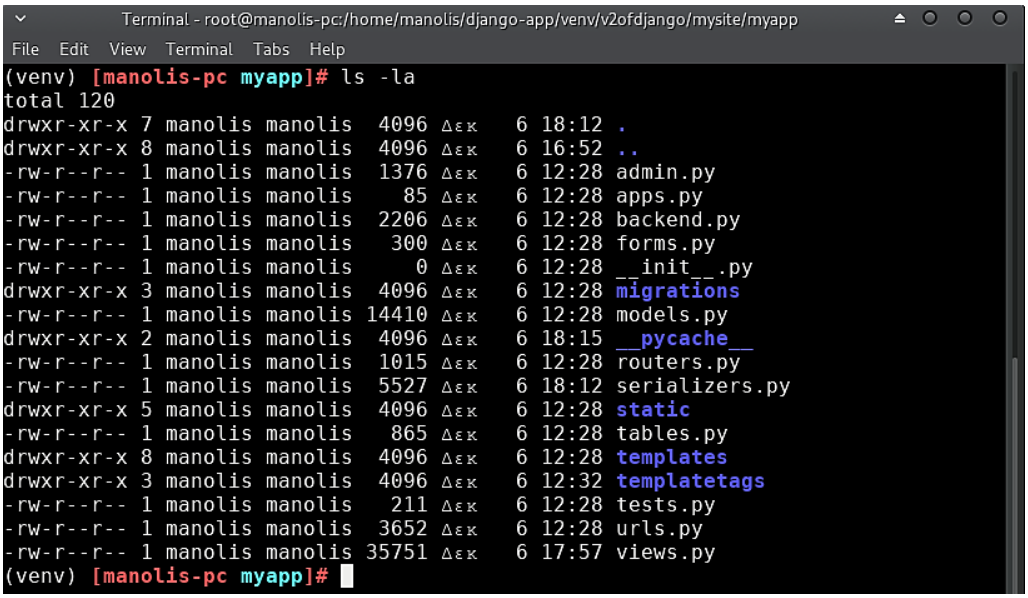
### 3.1.4 Δημιουργία μιας νέας εφαρμογής

Στη συνέχεια, εφόσον έχουν ολοκληρωθεί τα παραπάνω βήματα, με την εντολή

**→ python manage.py startapp myapp**

δημιουργείται μία νέα εφαρμογή (για το σκοπό της διπλωματικής η εφαρμογή ορίζεται ως **myapp**). Η επόμενη κίνηση, που πρέπει να πραγματοποιηθεί μετά τη δημιουργία της εφαρμογής, είναι να προστεθεί το όνομα αυτής στα **INSTALLED_APPS**

στο αρχείο **settings.py** ως εξής: **'myapp',** . Στην ακόλουθη εικόνα εμφανίζεται ο φάκελος myapp της εφαρμογής την οποία αναπτύξαμε.




Εικόνα 6

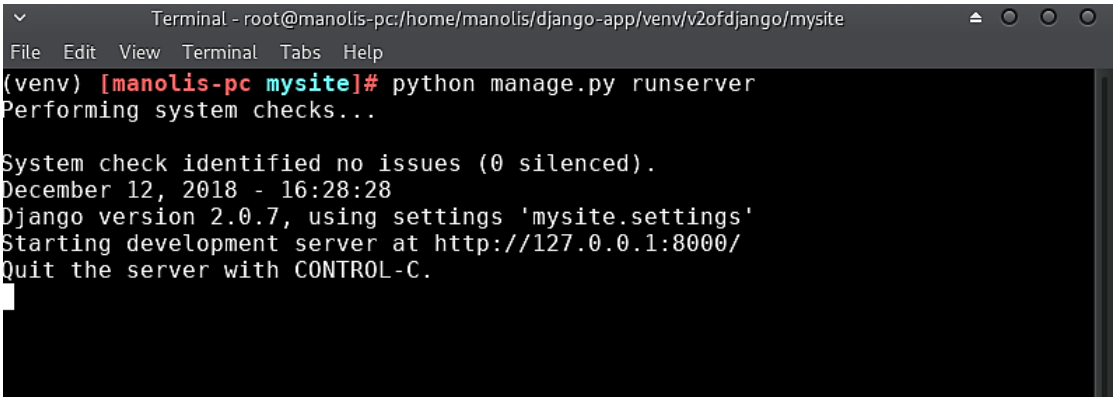
Ακολούθως, ο Development Server ενεργοποιείται τοπικά πληκτρολογώντας την εντολή

**→ python manage.py runserver**

Αυτός συνήθως "ακούει" στην πόρτα **8000,** όπως εμφανίζεται στην παρακάτω εικόνα. Στο σημείο αυτό, αξίζει να πούμε πως, αν και η παραπάνω πόρτα είναι η default επιλογή του server, μπορεί εύκολα να αλλάξει, αρκεί να πληκτρολογήσει κανείς την εντολή

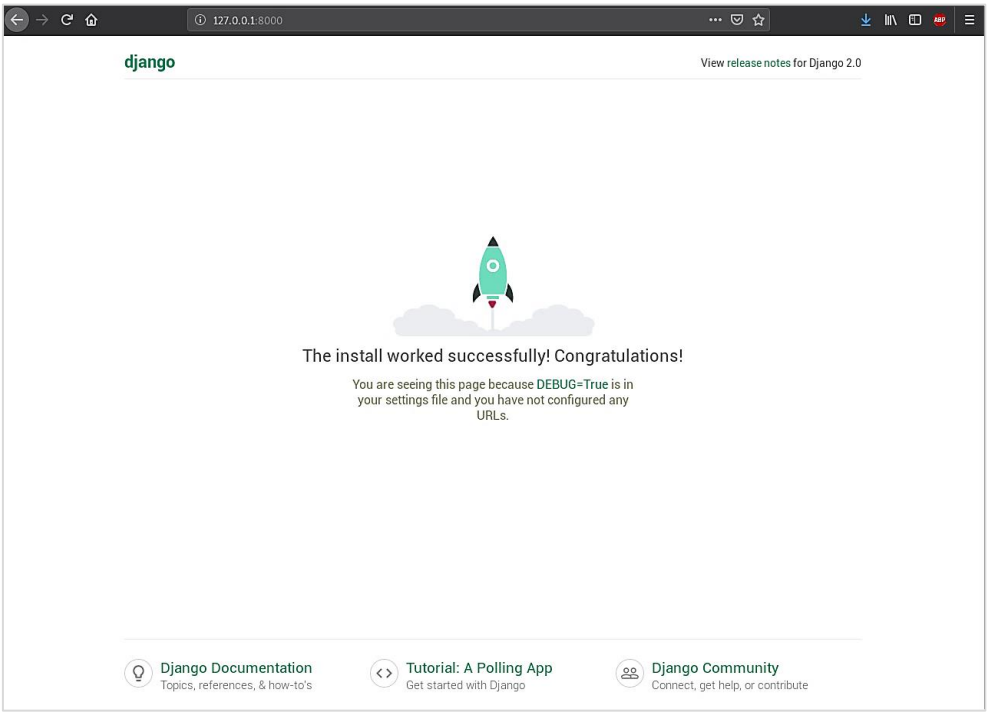
**→ python manage.py runserver 9000**

Έτσι, ο server ενεργοποιείται στην πόρτα **9000** περιμένοντας τα αντίστοιχα αιτήματα.




Εικόνα 7

Με το που θα ανοίξει κανείς τον φυλλομετρητή (browser) στη διαδρομή http://127.0.0.1:8000/ θα διαπιστώσει, διαβάζοντας το αντίστοιχο μήνυμα, πως έχει επιτυχώς εγκαταστήσει το Django και κατά συνέπεια μία καινούρια εφαρμογή, που λειτουργεί και μπορεί να την επεξεργαστεί περαιτέρω.




Εικόνα 8

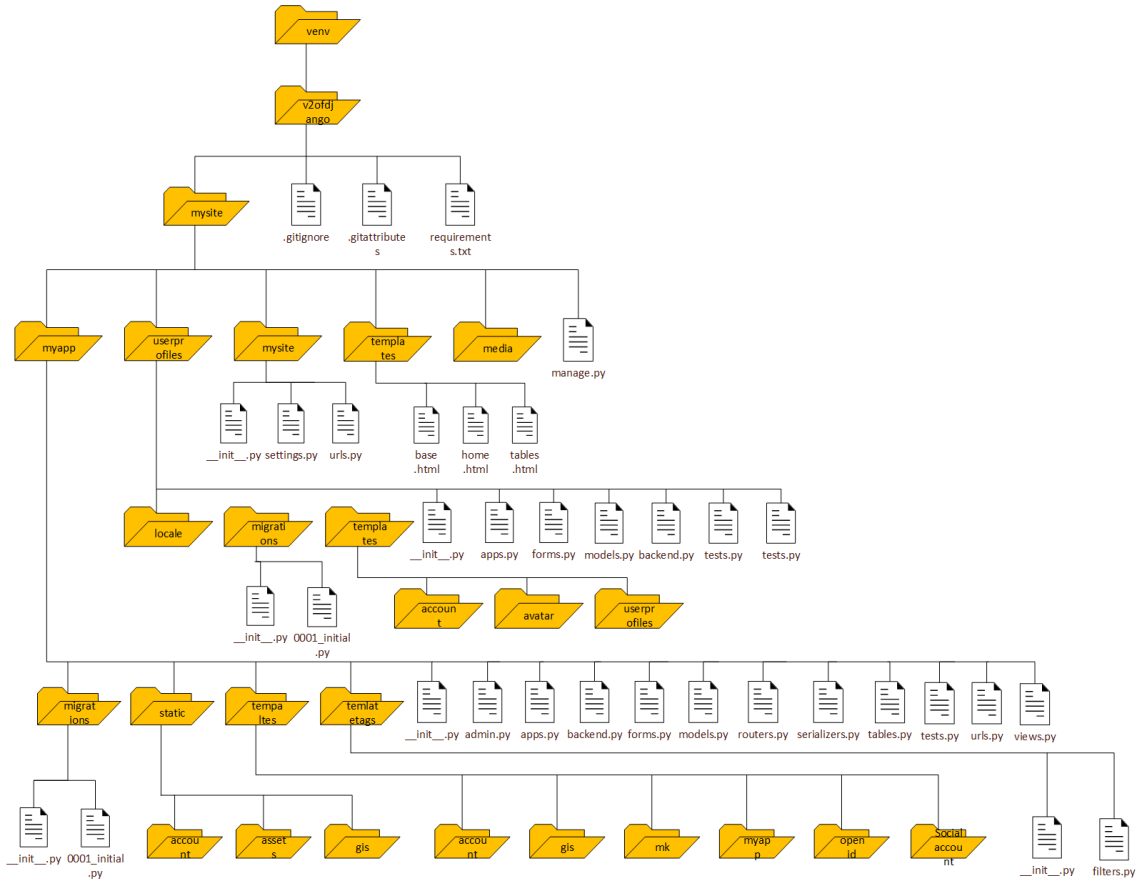
Κάτι το οποίο αποτελεί βασικό στοιχείο και χρήζει αναφοράς είναι η δημιουργία ενός νέου **superuser**, όπως ονομάζεται στην αργκό του Django. Με την εντολή

**→ python manage.py createsuperuser**

δημιουργούμε έναν νέο χρήστη, ο οποίος έχει δικαιώματα διαχειριστή (**admin**) της εφαρμογής, εξού και η ονομασία superuser. Με τα στοιχεία, που δήλωσε ο διαχειριστής προηγουμένως, θα είναι σε θέση να κάνει login στη σελίδα **Admin** του Django, η οποία αναλύεται στη συνέχεια.

### 3.1.5 Δομή της εφαρμογής

Στην εικόνα που ακολουθεί εμφανίζεται η δομή της εφαρμογής (project), που δημιουργήθηκε σταδιακά κατά την ανάπτυξή της. Αξίζει να πούμε πως ορισμένα αρχεία παραλείφθηκαν, προκειμένου να επιτευχθεί η βέλτιστη δομή του σχήματος .




Εικόνα 9

### 3.1.6 Django Admin

Πρόκειται για ένα από τα πιο ισχυρά χαρακτηριστικά-στοιχεία του Django, καθώς δίνει τη δυνατότητα στον χρήστη-διαχειριστή να έχει τη εποπτεία των Models, που δημιουργούνται στη βάση δεδομένων. Αρχικά, για να έχει πρόσβαση στη συγκεκριμένη σελίδα, θα πρέπει κανείς να συμπεριλάβει στο **INSTALLED_APPS** (στο αρχείο **settings.py**) τα παρακάτω:

- 'django.contrib.admin',
- 'django.contrib.auth',
- 'django.contrib.contenttypes',
- 'django.contrib.messages',
- 'django.contrib.sessions',

Εν συνεχεία, θα πρέπει κανείς να προσθέσει στα **TEMPLATES** στις επιλογές (**OPTIONS**) στο **context_processors** (στο ίδιο αρχείο) τα εξής:

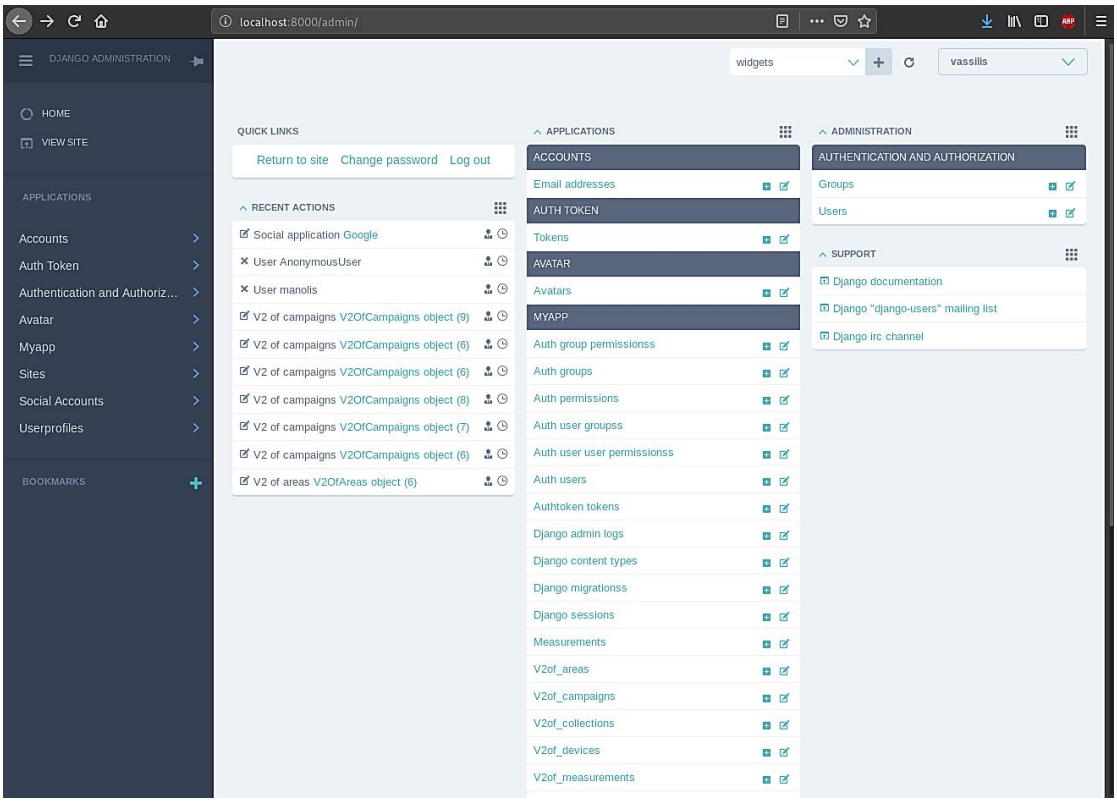
- 'django.contrib.auth.context_processors.auth',
- 'django.contrib.messages.context_processors.messages',

Τα παραπάνω συμπεριλαμβάνονται ήδη σε ένα project, εφόσον αυτό δημιουργηθεί, σύμφωνα με τον τρόπο, που περιγράφηκε προηγουμένως (ενότητα 3.1.3) [22].
Η δυνατότητα του να παρακολουθεί ο χρήστης τα μοντέλα, που διαθέτει η βάση δεδομένων, επιτυγχάνεται με την προσθήκη του κατάλληλου κώδικα στο αρχείο **admin.py** της εφαρμογής. Συγκεκριμένα, φτιάχνοντας μία κλάση μέσα στο αρχείο αυτό, η οποία θα περιλαμβάνει ένα **ModelAdmin** και αξιοποιώντας κατάλληλα την εντολή **admin.site.register()** μπορεί να συμπεριλάβει-εμφανίσει μέσα στη σελίδα admin μία τέτοια εγγραφή (Model). Ακόμα, μπορεί να πλοηγηθεί γραφικά και να παρατηρήσει, τροποποιήσει, προσθέσει, αλλά και να αφαιρέσει οποιαδήποτε εγγραφή επιθυμεί, η οποία σχετίζεται με τη βάση δεδομένων. Όλα αυτά μπορούν να πραγματοποιηθούν, εάν ο χρήστης σηκώσει τοπικά το Server, όπως αναφέρθηκε προηγουμένως και προσθέσει στο Browser τη διαδρομή (path):

**→ http://localhost:8000/admin**

Εκεί θα του ζητηθούν τα αντίστοιχα username και password, τα οποία δήλωσε κατά τη διάρκεια της δημιουργίας του superuser. Στην εικόνα που ακολουθεί εμφανίζονται

ορισμένα από τα Models της εφαρμογής, που αναπτύχθηκε στο πλαίσιο της διπλωματικής.




Εικόνα 10

Αυτά που μόλις αναφέρθηκαν είναι ορισμένα από τα χαρακτηριστικά της σελίδας αυτής του Django και αναδεικνύουν ένα μικρό μέρος των δυνατοτήτων της. Υπάρχουν, επίσης, τα λεγόμενα **Admin Actions,** καθώς και **Admin Documentation Generator,** όπως τα ονομάζει το Django. Στο πρώτο μπορεί κανείς να επιλέξει πολλά αντικείμενα (objects) και να πραγματοποιήσει μονομιάς την ίδια αλλαγή σε όλα, ενώ σε διαφορετική περίπτωση θα έπρεπε να επιλέξει ένα μόνο και να πραγματοποιήσει οποιαδήποτε αλλαγή μόνο σε αυτό. Όσον αφορά το δεύτερο, πρόκειται για μία εφαρμογή, η οποία εξάγει πληροφορίες για όλες τις εφαρμογές, που είναι εγκατεστημένες στα **INSTALLED_APPS** στο αρχείο **settings.py**, και τις κάνει προσβάσιμες μέσω του Django Admin σε μορφή document, εξού και η ονομασία αυτού.

### 3.1.7 Εγκατάσταση Βάσης Δεδομένων

Το πρώτο πράγμα, που πρέπει να σκεφτεί κανείς, εφόσον έχει εγκαταστήσει με επιτυχία την εφαρμογή του κι όλα βαίνουν καλώς, είναι να συνδεθεί με τη βάση στην οποία φιλοξενούνται τα δεδομένα, τα οποία χρήζουν διαχείρισης. Στη συγκεκριμένη εφαρμογή χρησιμοποιήθηκε η βάση **PostgreSQL,** όπως αναφέρθηκε σε προηγούμενο κεφάλαιο.
Ειδικότερα, για το σκοπό της διπλωματικής χρησιμοποιήθηκαν **δύο** βάσεις δεδομένων ίδιου τύπου, μία για το προφίλ των χρηστών (**globedb**) και μία για τις μετρήσεις (**Measurements**) [23].

Αφού ακολουθήσουμε την παραπάνω διαδικασία και εγκαταστήσουμε κάποιο project, διαπιστώνουμε ότι το Django έχει φροντίσει, ώστε να παρέχει στο χρήστη μία ενσωματωμένη βάση δεδομένων (default) και αυτή είναι η **SQLite**.

Σε περίπτωση, λοιπόν, που θελήσει κανείς να χρησιμοποιήσει μία διαφορετική βάση δεδομένων, θα πρέπει να εγκαταστήσει τα αντίστοιχα πακέτα αυτής, καθώς και να πραγματοποιήσει τις απαραίτητες αλλαγές μέσα στο αρχείο **settings.py** στο αντικείμενο **DATABASES 'default'**. Για παράδειγμα, για την βάση δεδομένων **PostgreSQL**, που χρησιμοποιήθηκε στην εργασία αυτή, χρειάστηκε να προσθέσουμε επιπλέον τα εξής πεδία:

```
DATABASES = {
  'default': {
     'ENGINE': 'django.contrib.gis.db.backends.postgis',
     'OPTIONS': {'options': '-c search_path=private,public',
              'sslmode': 'require',
              },
     'NAME': 'globedb',
     'USER': 'cr',
     'PASSWORD': '*******',
     'HOST': 'test.hua.gr',
     'PORT': '',
  },
```

```
'legacyDB': {

    'ENGINE': 'django.contrib.gis.db.backends.postgis',
    'OPTIONS': {'options': '-c search_path=public',
            'sslmode': 'require',
            },
    'NAME': 'Measurements',
    'USER': 'cr',
    'PASSWORD': '******',
    'HOST': 'test.hua.gr',
    'PORT': '',
  }
}
```

Το **NAME** αντιστοιχεί στο όνομα της βάσης δεδομένων και **USER** είναι το όνομα του χρήστη, που κατέχει τα απαραίτητα δικαιώματα, ώστε να διαχειρίζεται τη βάση αυτή εισάγοντας το κατάλληλο **PASSWORD**. Όπου **HOST** είναι ο αντίστοιχος εξυπηρετητής, που χρησιμοποιεί ο χρήστης και στη συγκεκριμένη περίπτωση πρόκειται για τον **test.hua.gr** του Χαροκοπείου Πανεπιστημίου (επί της ουσίας είναι ο αντίστοιχος server). Ακόμα, στο πεδίο **PORT** εισάγεται ο αριθμός της πόρτας, όταν επιτυγχάνεται η σύνδεση με τη βάση δεδομένων. Όσον αφορά το πεδίο **ENGINE**, σε αυτό εισάγεται το Backend κομμάτι της βάσης δεδομένων, που έχει χρησιμοποιηθεί. Επομένως, αν παρατηρήσει κανείς τα παραπάνω, θα διαπιστώσει πως η συγκεκριμένη διπλωματική έχει υλοποιηθεί με τη βάση δεδομένων **PostGIS** (πρόκειται για μία επέκταση της PostgreSQL), η οποία εφαρμόζεται κυρίως σε γεωχωρικά-γεωγραφικά δεδομένα.

### 3.1.8 Εγκατάσταση PostgreSQL και PostGIS

Σε πρώτο επίπεδο, χρειάστηκε να εφαρμοσθεί μια ακολουθία πραγμάτων προκειμένου να εγκατασταθεί η βάση δεδομένων. Ανάλογα με την έκδοση της γλώσσας Python θα πρέπει να εγκατασταθούν τα αντίστοιχα πακέτα.

Στη διπλωματική αυτή χρησιμοποιήθηκε η έκδοση **3.7.1**, όπως έχει ήδη αναφερθεί, επομένως με βάση αυτή πραγματοποιήθηκαν και οι κατάλληλες ενέργειες:

**→ sudo apt-get install python3-pip python3-dev libpq-dev postgresql postgresql-contrib**

Στη συνέχεια για να έχουμε πρόσβαση στη βάση δεδομένων εισάγονται οι ακόλουθες εντολές:

**→ sudo -u postgres**

**→ psql**

Έπειτα, ο εκάστοτε χρήστης είναι σε θέση να δημιουργήσει την αντίστοιχη βάση, καθώς και τους αντίστοιχους πίνακες χρησιμοποιώντας τις κατάλληλες **SQL** εντολές. Για τη σύνδεση της βάσης με το Django Framework χρειάζεται να εγκατασταθεί ο αντίστοιχος driver:

**→ pip install django psycopg2**

Ακόμα, σε τοπικό (local) επίπεδο πραγματοποιούνται ορισμένες αλλαγές στο αρχείο **postgresql.conf**, ώστε να επιτευχθεί απομακρυσμένη πρόσβαση με τη βάση. Το αρχείο αυτό βρίσκεται στο path **/etc/postgresql/10/main/postgresql.conf** (το συγκεκριμένο path διαφέρει ανάλογα με τη διανομή του Linux αλλά και την έκδοση της βάσης). Χρησιμοποιώντας έναν οποιονδήποτε editor, στο πεδίο "CONNECTIONS AND AUTHENTICATION" εισάγεται η γραμμή: **listen_addresses = '*'**. Στο αρχείο **pg_hba.conf** γίνονται μερικές επιπλέον προσθήκες, ώστε να ενεργοποιηθεί η σύνδεση με τη βάση. Μία από αυτές είναι η ακόλουθη:

```
# Allow Remote Connections

host    all         all         0.0.0.0/0           md5
```

Με την παραπάνω αλλαγή επιτρέπονται οι συνδέσεις στη βάση από όλες τις IP's και συγκεκριμένα για τις **IPv4**. Αν για παράδειγμα κάποιος χρήστης θελήσει να επιτρέπονται οι **IPv6** συνδέσεις, πρέπει να πραγματοποιήσει την εξής αντικατάσταση σύμφωνα με το παραπάνω, από 0.0.0.0/0 **σε** ::0/0 .

Όσον αφορά την επέκταση PostGIS, είναι αναγκαίο να εγκατασταθούν τα ακόλουθα πακέτα:

**→ sudo apt-get install binutils libproj-dev gdal-bin python-gdal libgeoip1**

Απαραίτητη, επίσης, είναι η εγκατάσταση των βιβλιοθηκών GEOS, PROJ.4, και GDAL [24]. Αξίζει να αναφερθεί πως οι παραπάνω βιβλιοθήκες προηγούνται σε εγκατάσταση της PostGIS. Περισσότερες λεπτομέρειες μπορεί να βρει κανείς στο επίσημο site: https://postgis.net/ .

Σε αυτό το σημείο δεν έχει ολοκληρωθεί ακόμα η δημιουργία της βάσης δεδομένων. Για να πραγματοποιηθεί αυτό θα πρέπει να πληκτρολογήσει αρχικά ο χρήστης την εντολή

**→ python manage.py makemigrations**.

Με την εντολή αυτή εμφανίζονται οι αλλαγές, που θα γίνουν στα μοντέλα του αρχείου models.py. Έπειτα, ο χρήστης πληκτρολογεί στην κονσόλα την εντολή

**→ python manage.py migrate**,

με την οποία **εφαρμόζονται** οι αλλαγές, που πραγματοποιήθηκαν στα μοντέλα. Ουσιαστικά, αυτό που κάνει η συγκεκριμένη εντολή, είναι να ρίχνει μία ματιά στο αρχείο **settings.py** στο **INSTALLED_APPS** και να δημιουργεί τους κατάλληλους πίνακες σύμφωνα με τη βάση δεδομένων την οποία έχουμε ορίσει. Μια πολύ σημαντική λειτουργία, η οποία προσφέρεται από το Django, είναι πως, όταν υπάρχει μια βάση δεδομένων με ήδη υπάρχοντες πίνακες, μπορεί να μετατρέψει αυτόματα τους πίνακες σε μοντέλα (Models) χωρίς να τα δημιουργήσει εξαρχής γράφοντάς τα ένα προς ένα, αρκεί να πληκτρολογήσει την εντολή:

**→ python manage.py inspectdb > models.py**

Στην παραπάνω εντολή χρησιμοποιείται το σύμβολο **>** ώστε να δημιουργηθούν οι πίνακες σε γλώσσα Python στο αρχείο **models.py**.

## 3.2 Εργαλεία Ανάπτυξης Λογισμικού της Εφαρμογής

### 3.2.1 Pycharm IDE 

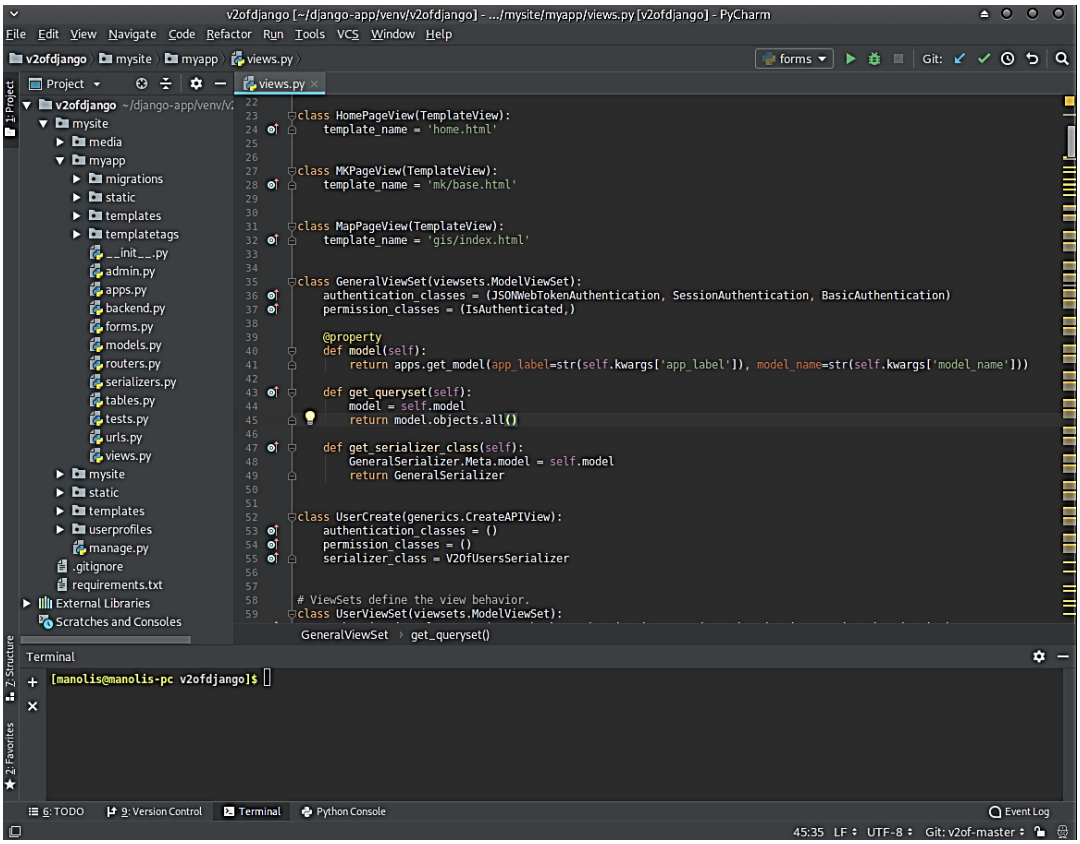
Πρόκειται για ένα περιβάλλον ανάπτυξης κώδικα, το οποίο δημιουργήθηκε από μια εταιρεία Τσεχικής προέλευσης ονόματι Jetbrains και πρωτοκυκλοφόρησε τον Ιούλιο του 2010. Υποστηρίζει κυρίως τη συγγραφή κώδικα σε γλώσσα Python, καθώς και την ανάπτυξη του Django Framework. Ορισμένες από τις δυνατότητες, που παρέχει το Pycharm IDE (το οποίο διατίθεται με γραφικό περιβάλλον), είναι η ανάλυση του κώδικα, ο ενσωματωμένος debugger, αλλά και το ενσωματωμένο εργαλείο διαχείρισης του λογισμικού, που αναπτύσσεται σε αυτό (Version Control System). Με το τελευταίο ο χρήστης έχει τον πλήρη έλεγχο του κώδικα (ενημέρωση, διαγραφή), τον οποίο διαθέτει σε κάποιο αποθετήριο, όπως είναι για παράδειγμα το Github, πατώντας απλά τα απαραίτητα κλικς. Επίσης, αξίζει να αναφέρουμε πως το Pycharm μπορεί να εγκατασταθεί σε διαφορετικά λειτουργικά συστήματα όπως είναι τα Windows, το Linux αλλά και το Mac OS. Είναι, με άλλα λόγια, μια multi-platform ή cross-platform εφαρμογή. Για την ανάπτυξη της εφαρμογής στην παρούσα διπλωματική χρησιμοποιήθηκε η έκδοση σε Linux και ακριβέστερα η Full-featured IDE. Πρωτίστως, θα πρέπει κανείς να κατεβάσει το πρόγραμμα από την επίσημη ιστοσελίδα https://www.jetbrains.com/pycharm/download/#section=linux .

Έπειτα, τα βήματα για την εγκατάσταση είναι τα εξής:

- **$ sudo tar -xzf pycharm-professional-2018.2.4.tar.gz** (αποσυμπίεση αρχείου)
- **$ cd pycharm-professional-2018.2.4/bin/** (αλλαγή καταλόγου)
- **$ sh pycharm.sh** (εκτέλεση της εφαρμογής)

Η εικόνα που ακολουθεί είναι ενδεικτική του περιβάλλοντος του Pycharm (χρησιμοποιήθηκε το Darcula Theme). Ένα ακόμα χαρακτηριστικό, το οποίο κρίνεται σκόπιμο να επισημανθεί είναι ότι το τερματικό (**Terminal**), το οποίο διαθέτει το Pycharm, δίνει την δυνατότητα στον χρήστη να πληκτρολογεί τις απαραίτητες εντολές

μέσα σε αυτό κρατώντας άμεση επαφή με τον κώδικα, χωρίς δηλαδή την εναλλαγή παραθύρων [25].



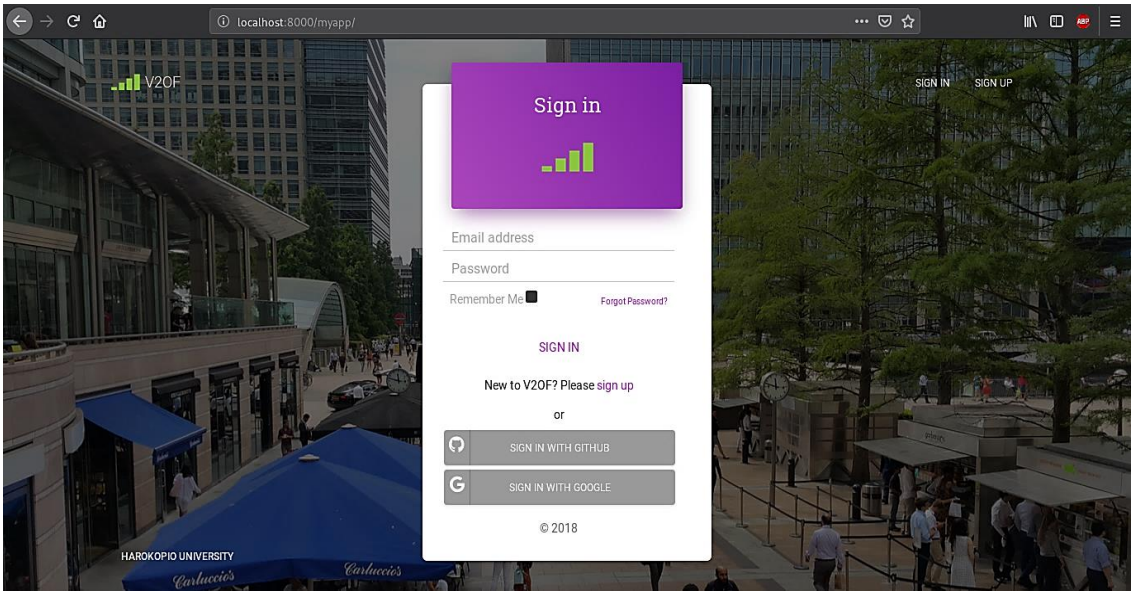

Εικόνα 11

# 4. Django Packages

## 4. 1 Django Allauth

Με το συγκεκριμένο πακέτο μπορεί κανείς να συνδεθεί με το λογαριασμό, που διαθέτει σε κάποιο από τα κοινωνικά δίκτυα (π.χ. Facebook, Twitter). Χρησιμοποιείται κυρίως με σκοπό την αυθεντικοποίηση των χρηστών στο διαδίκτυο με έναν ήδη υπάρχοντα λογαριασμό στα διάφορα κοινωνικά μέσα.

Στην εφαρμογή, που αναπτύχθηκε, δίνεται η δυνατότητα στο χρήστη να κάνει login χρησιμοποιώντας το λογαριασμό του **Github** ή της **Google**, κάνοντας κλικ σε ένα από τα δύο αντίστοιχα εικονίδια, τα οποία βρίσκονται στη σελίδα login, όπως φαίνεται στην παρακάτω εικόνα. Σημειώνεται εδώ πως είναι απαραίτητο ο χρήστης να διαθέτει λογαριασμό σε μία από τις δύο πλατφόρμες [26].




Εικόνα 12

**Εγκατάσταση Django - allauth**

Για την εγκατάσταση του allauth χρειάζονται οι ακόλουθες ενέργειες:

Εγκατάσταση Python package: **pip install django-allauth**

Μέσα στο αρχείο **settings.py** θα πρέπει να προστεθούν τα ακόλουθα:

**settings.py:**

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                # Already defined Django-related contexts here
                # `allauth` needs this from django
                'django.template.context_processors.request',
            ],
        },
    },
]
```

```
AUTHENTICATION_BACKENDS = (
   ...
   # Needed to login by username in Django admin, regardless of `allauth`
   'django.contrib.auth.backends.ModelBackend',
   # `allauth` specific authentication methods, such as login by e-mail
   'allauth.account.auth_backends.AuthenticationBackend',
   ...
)
INSTALLED_APPS = (
…
          'allauth',
   'allauth.account',
   'allauth.socialaccount',
   'allauth.socialaccount.providers.github',
   'allauth.socialaccount.providers.google',
)
SITE_ID = 1
```

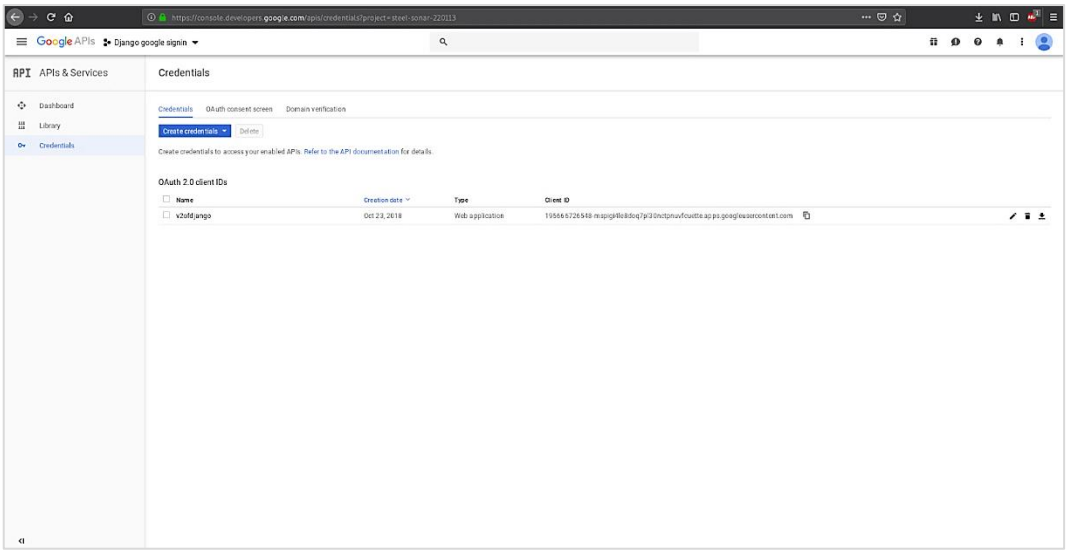
Στη συνέχεια, μέσα στο αρχείο **urls.py** της εφαρμογής θα πρέπει να προστεθεί το αντίστοιχο url:

```
urlpatterns = [
   ...
   url(r'^accounts/', include('allauth.urls')),
   ...
]
```

**Δημιουργία Google Social Account**

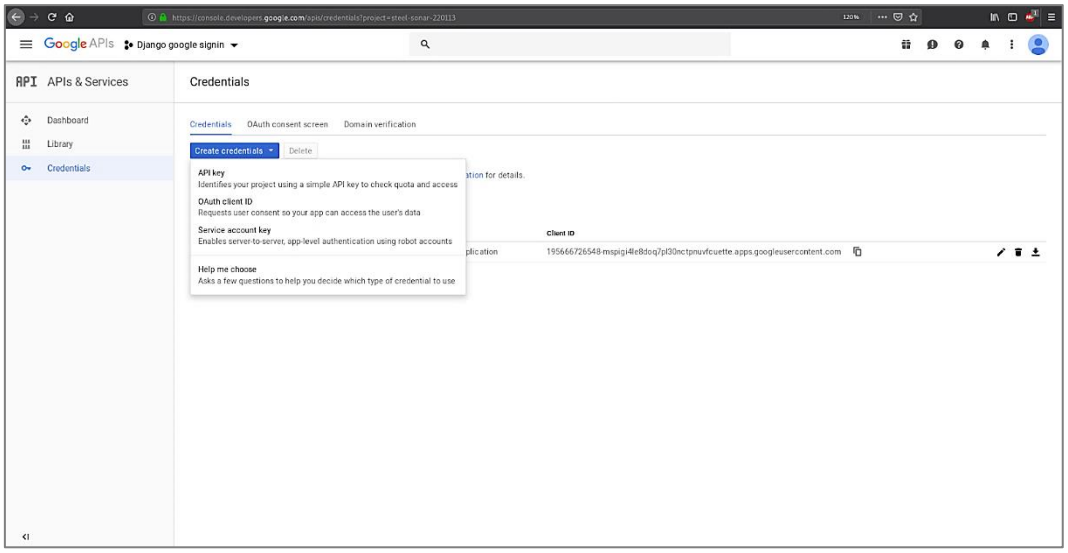
Προκειμένου ο προγραμματιστής να προσθέσει ένα κοινωνικό μέσο σε μία εφαρμογή, θα πρέπει ήδη να διαθέτει ένα λογαριασμό στο **Google Social Account**. Ακολουθεί το παράδειγμα της προσθήκης του κουμπιού (button) στην πλατφόρμα Google.

Αρχικά, ο διαχειριστής θα πρέπει να συνδεθεί με τα στοιχεία του στην πλατφόρμα Google API Console [27], η οποία εμφανίζεται στην ακόλουθη εικόνα.



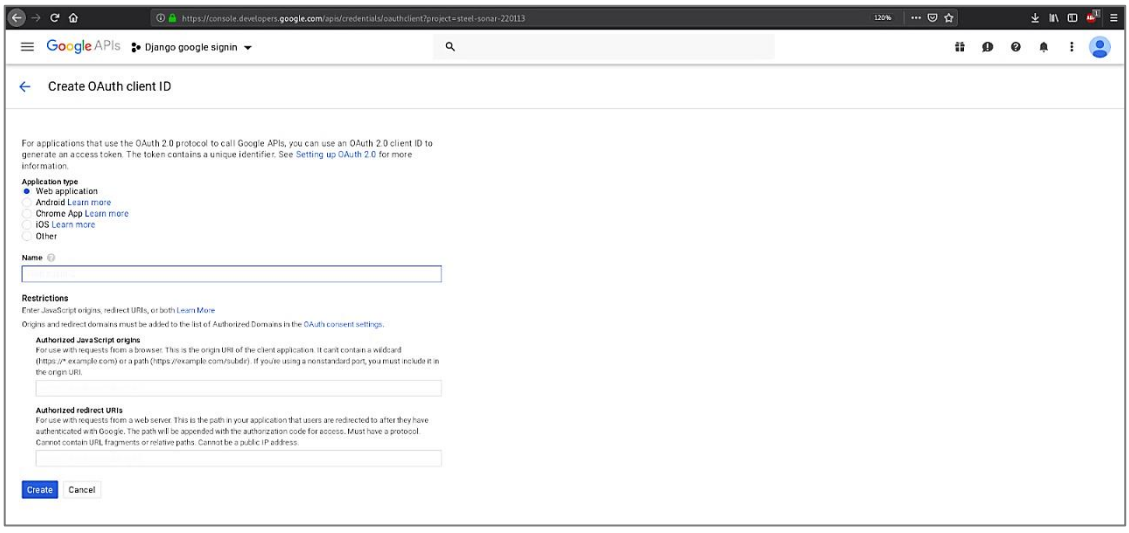

Εικόνα 13

Έπειτα, θα πρέπει να επιλέξει το κουμπί **Credentials,** το οποίο βρίσκεται στην καρτέλα **APIs & Services**, που εμφανίζεται στο αριστερό μέρος της οθόνης. Στη συνέχεια, αφού μεταφερθεί στην καρτέλα Credentials, επιλέγει το πλαίσιο **Create Credentials** και αμέσως μετά, στη λίστα που εμφανίζεται, επιλέγει το **OAuth Client ID**, όπως φαίνεται στην πιο κάτω εικόνα.




Εικόνα 14

Ανάλογα με την εφαρμογή, που επιθυμεί να δημιουργήσει, επιλέγει ένα από τα αντίστοιχα είδη τα οποία εμφανίζονται στην εικόνα. Στη συγκεκριμένη περίπτωση, επιλέχθηκε το **Web application**, λόγω του ότι η εφαρμογή, που αναπτύχθηκε στα πλαίσια της διπλωματικής, είναι διαδικτυακή.




Εικόνα 15

Παρακάτω εμφανίζονται ορισμένα νέα πεδία, τα οποία ζητούν από το χρήστη να δηλώσει ένα όνομα (πεδίο **Name**), που θα συμβολίζει τον τύπο της εφαρμογής, την

οποία επέλεξε πιο πάνω, όπως επίσης και τα/το αντίστοιχα/ο URIs της εφαρμογής (πεδίο **Authorized Redirect URIs**).

Το δεύτερο πεδίο είναι και το πιο σημαντικό, διότι σε αυτό δηλώνεται η διαδρομή (path) την οποία ακολουθεί η συγκεκριμένη εφαρμογή και η οποία (διαδρομή) είναι υπεύθυνη στο να ανακατευθύνει το χρήστη στην αρχική σελίδα. Για παράδειγμα, η συγκεκριμένη εφαρμογή τρέχει τοπικά στο **localhost** και "ακούει" στην πόρτα **8000**. Επομένως, το αντίστοιχο μονοπάτι (path) που έπρεπε να δηλωθεί στο πεδίο Authorized Redirect URIs ήταν το εξής: **http://localhost:8000/myapp/account/google/login/callback/ .** Στο σημείο αυτό θα πρέπει να διευκρινιστεί πως, εκτός από τον host και την πόρτα στην οποία "ακούει" η εφαρμογή, αμέσως μετά προστέθηκε στην ήδη υπάρχουσα διαδρομή και η σελίδα στην οποία θα ανακατευθυνθεί ο χρήστης.

Είναι σημαντικό, επίσης, να αναφέρουμε πως μέσα στο αρχείο **settings.py** πραγματοποιείται η δήλωση της σελίδας ως εξής: **LOGIN_REDIRECT_URL = "home"**. Συγκεκριμένα, πρόκειται για τη σελίδα, στην οποία οδηγείται ο χρήστης, εφόσον η σύνδεσή του στην πλατφόρμα της Google πραγματοποιηθεί με επιτυχία.

Τέλος, πατώντας το κουμπί **create** εμφανίζεται ένα παράθυρο στο οποίο είναι εμφανή ένα **Client ID** και ένα **Client secret**. Αυτά τα δύο στοιχεία θα χρειαστούν στην συνέχεια, καθώς θα τα εισάγουμε στην σελίδα του Django Admin. Με αυτόν τον τρόπο, λοιπόν, δημιουργούμε ένα **OAuth Client ID**.

Επιστρέφοντας στην σελίδα **Credentials**, διαπιστώνουμε πως εμφανίζεται ο Client, που δημιουργήσαμε προηγουμένως κάτω από την ετικέτα **OAuth 2.0 client IDs**. Αξίζει να επισημάνουμε εδώ πως έχουμε τη δυνατότητα να τροποποιήσουμε έναν τέτοιο client κάνοντας κλικ πάνω σε αυτόν.

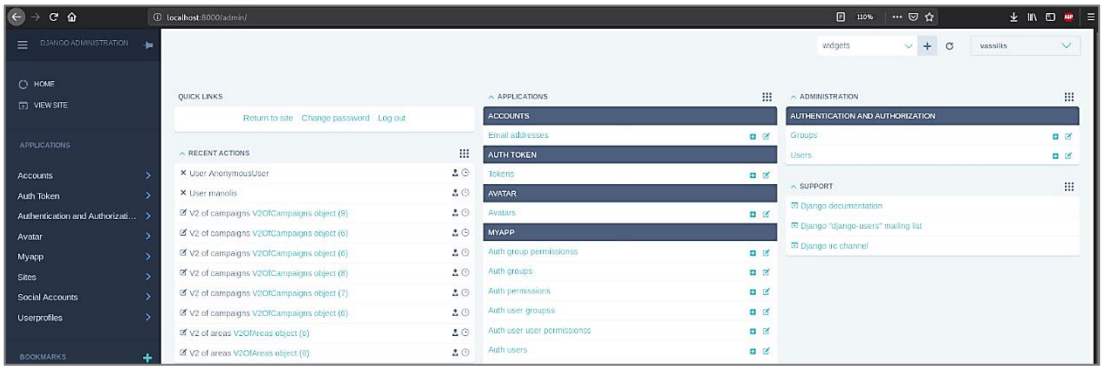
**Admin Google - social accounts**

Τρέχοντας τοπικά τον server με την εντολή:

**→ python manage.py runserver**

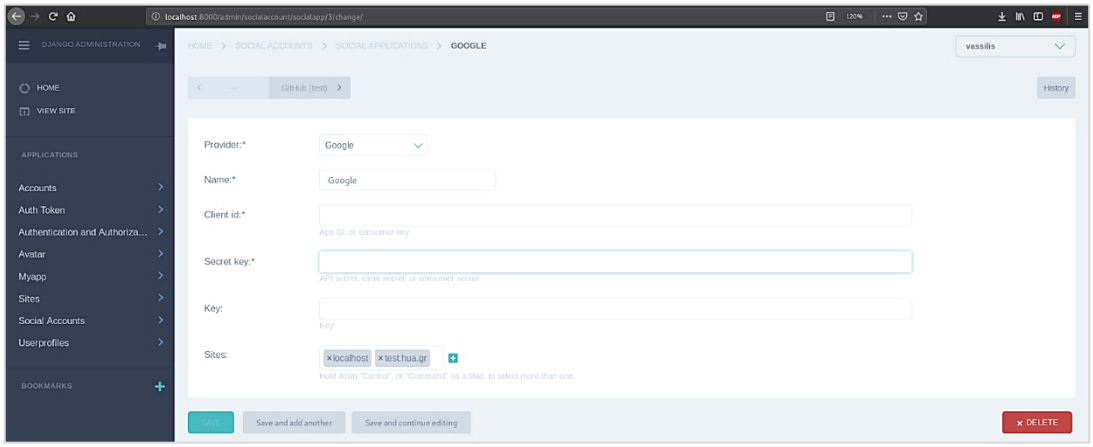
μπορεί κανείς να έχει πρόσβαση στην εφαρμογή σαν **administrator** (διαχειριστής), πληκτρολογώντας στο φυλλομετρητή (Browser) τη διαδρομή **http://localhost:8000/admin/**, όπως έχει προαναφερθεί.

Εισάγοντας τα κατάλληλα διαπιστευτήρια ο εκάστοτε διαχειριστής μεταφέρεται στην αρχική σελίδα. Εκεί, αρκεί να πλοηγηθεί στην καρτέλα **Social Accounts** επιλέγοντάς τη από το μενού, που εμφανίζεται αριστερά στην οθόνη, όπως φαίνεται στην παρακάτω εικόνα.




Εικόνα 16

Έπειτα, επιλέγοντας την καρτέλα **Social applications** και πατώντας στο κουμπί **Add social application** ο διαχειριστής μπορεί να προσθέσει ένα κοινωνικό μέσο. Παρακάτω εξηγούνται πιο αναλυτικά τα πεδία, τα οποία εμφανίζονται στην εικόνα:




Εικόνα 17

- **Provider**: Το πεδίο αυτό είναι άρρηκτα συνδεδεμένο με το αρχείο **settings.py,** καθότι σε αυτό δηλώνεται το εκάστοτε κοινωνικό μέσο. Μέσα στο αρχείο, στην κατηγορία **INSTALLED_APPS**, γίνεται η εξής δήλωση:

**‘allauth.socialaccount.providers.google’** . Χάρη σε αυτό εμφανίζεται η επιλογή **Google** στο πεδίο του Provider.

- **Name**: Εδώ δηλώνεται το όνομα που δίνει ο διαχειριστής στην εφαρμογή (συνήθως λαμβάνει το όνομα του κοινωνικού μέσου).

- **Client id**: Στο πεδίο αυτό δηλώνεται το αντίστοιχο **Client ID** του OAuth 2.0 client, που αναφέρθηκε προηγουμένως, και το οποίο είναι μοναδικό για κάθε client που δημιουργείται.

- **Secret key**: Αφορά το αντίστοιχο Client secret και είναι εξίσου μοναδικό για την εφαρμογή.

- **Sites**: Αυτό το πεδίο αφορά στον host, στον οποίο τρέχει η εφαρμογή που αναπτύχθηκε. Αξίζει να αναφέρουμε πως στη συγκεκριμένη διπλωματική, εκτός από τοπικά (localhost), η εφαρμογή τρέχει και στον server του πανεπιστημίου. Για το λόγο αυτό προσθέσαμε δύο hosts, το **localhost** και το **test.hua.gr**

Έπειτα, πατώντας το κουμπί **save** αποθηκεύονται οι ενέργειες, που πραγματοποιήθηκαν, και με τον τρόπο αυτό προστίθεται μία κοινωνική εφαρμογή στη σελίδα του admin της εφαρμογής που δημιουργήθηκε. Τέλος, αυτό που μένει είναι να προστεθεί ο host (domain), στον οποίο τρέχει η εφαρμογή. Αυτό γίνεται εφικτό με τη μεταφορά του διαχειριστή στη σελίδα **http://localhost:8000/admin/sites/site/** . Εκεί, πατώντας το κουμπί **add site** θα πρέπει να δηλωθεί ένα **Domain name,** καθώς κι ένα **Display name**. Στο πρώτο πεδίο δηλώνεται το όνομα του domain της εφαρμογής, ενώ στο δεύτερο ένα ενδεικτικό όνομα, το οποίο συμβολίζει κυρίως την εφαρμογή. Πατώντας στη συνέχεια το κουμπί **save** δημιουργείται ένα νέο site.

Περισσότερες οδηγίες για τη δημιουργία ενός OAuth client μπορεί να αντλήσει κάποιος από τον επίσημο ιστότοπο https://developers.google.com/identity/protocols/OAuth2

## 4. 2 Django Rest Framework

Στο **κεφάλαιο 2.3** αναφέρεται το θεωρητικό κομμάτι του Django Rest Framework, ενώ εδώ θα δούμε το πρακτικό. Ουσιαστικά, περιγράφεται ο τρόπος εγκατάστασής του στην εφαρμογή, καθώς και οι απαραίτητες ενέργειες που το καθιστούν λειτουργικό. Η εντολή που χρησιμοποιήθηκε είναι:

**→ pip install djangorestframework**

Θα πρέπει, επίσης, να προστεθεί στο αρχείο INSTALLED_APPS:

**settings.py:**

```
INSTALLED_APPS = [
   'djangorestframework',
]
```

Ακόμα, σε περίπτωση που κάποιος ενδιαφέρεται να χρησιμοποιήσει το browsable API, θα πρέπει να προσθέσει το αντίστοιχο url στον κατάλογο του project:

**urls.py:**

```
urlpatterns = [

   url(r'^api-auth/', include('rest_framework.urls')) ]
```

## 4. 3 Django Avatar

Η συγκεκριμένη εφαρμογή διαθέτει την ιδιότητα να ορίζει ένα avatar profile (πρόκειται για την εικόνα που χρησιμοποιεί ο χρήστης στο προφίλ του) σε οποιονδήποτε χρήστη της εφαρμογής. Επιπλέον, παρέχει τη δυνατότητα στο χρήστη να χρησιμοποιήσει ένα ήδη υπάρχον avatar από κάποια τρίτη υπηρεσία, όπως για παράδειγμα είναι το Facebook [28].

Η εγκατάσταση του πακέτου πραγματοποιείται με την εντολή:

**→ pip install django-avatar**

Στη συνέχεια θα πρέπει να προστεθούν τα παρακάτω:

**settings.py:**

```
INSTALLED_APPS = [
   'avatar',
]
```

**urls.py:**

```
urlpatterns = [
    ...
   url('avatar/', include('avatar.urls')),
]
```

Το παραπάνω url προστίθεται στον κατάλογο του project.

Ακόμα χρειάζεται και η εντολή

**→ python manage.py migrate**

ώστε να δημιουργηθούν οι αντίστοιχοι πίνακες στη βάση.

## 4. 4 Django Hstore

Η βάση δεδομένων **PostgreSQL** διαθέτει μία επέκταση ονόματι **Hstore,** η οποία είναι συνυφασμένη με τη λογική του ζεύγους key-value για τους διάφορους τύπους των δεδομένων. Ταυτόχρονα, υποστηρίζει και την επέκταση **PostGIS** που χρησιμοποιείται στην εφαρμογή της παρούσας διπλωματικής. Ακόμα ένα χαρακτηριστικό του είναι πως διαθέτει το Django Admin widget το οποίο προσφέρει στην διαχειριστή ένα όμορφο περιβάλλον, μέσα από το οποίο μπορεί να εποπτεύει τη βάση [29].

Η αντίστοιχη εντολή για την εγκατάστασή του είναι:

**→ pip install django-hstore**.

Επίσης χρειάζονται τα ακόλουθα:

**settings.py:**

```
INSTALLED_APPS = [
   'django_hstore',
]
```

→ python manage.py collectstatic

Η παραπάνω εντολή χρησιμεύει στη συλλογή των static files του admin widget.

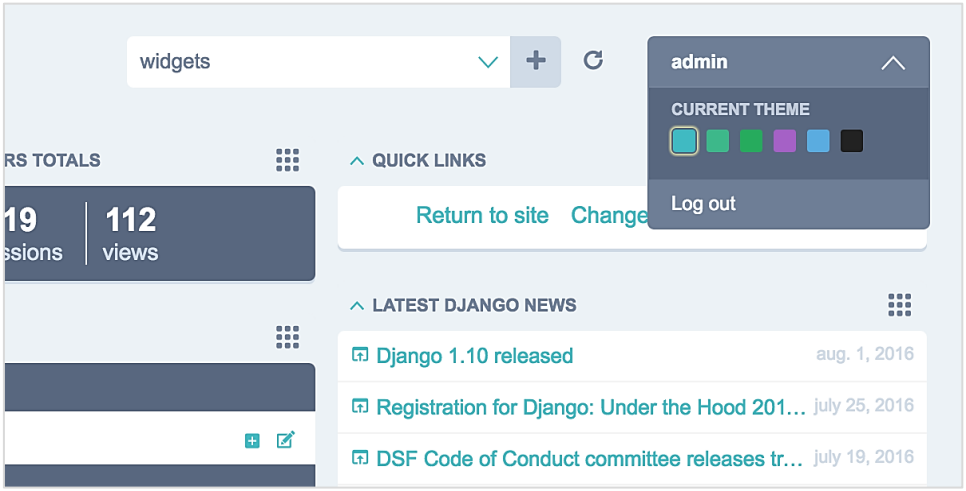
## 4. 5 Django jet

Όπως έχει ήδη ειπωθεί το Django Framework διαθέτει τη σελίδα admin μέσα από την οποία μπορεί κανείς να εποπτεύει τα πακέτα, τα οποία είναι εγκατεστημένα στην εφαρμογή. Το Django Jet έρχεται με τη σειρά του να προσθέσει ένα όμορφο περιβάλλον για το χρήστη admin, προσφέροντας του διάφορα θέματα (themes), ένα συμπαγές μενού (compact menu) κι άλλες λειτουργικότητες. Μία από αυτές είναι το **Google Analytics widget**, το οποίο παρέχει ποικίλους τρόπους προβολής των δεδομένων της εφαρμογής. Για να το εγκαταστήσει κανείς αρκεί να πληκτρολογήσει την εντολή

**→ pip install google-api-python-client==1.4.1**

Για την εγκατάσταση όλου του πακέτου χρειάζεται να πραγματοποιηθούν οι παρακάτω ενέργειες:

**→ pip install django-jet**




Εικόνα 18

**settings.py:**

```
INSTALLED_APPS = (
   'jet',
   'django.contrib.admin',)
```

Θα ήταν καλό, επίσης, να ελεγχθεί πως το **django.template.context_processors.request** context είναι ενεργοποιημένο:

```
TEMPLATES = [
  {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
              'context_processors': [
                    ...
                    'django.template.context_processors.request',
                    …
              ],
        },
  },
]
```

Δημιουργούνται τα αντίστοιχα **url-patterns** στο αρχείο **urls.py** του καταλόγου project:

**urls.py:**

```
urlpatterns = patterns(
  url(r'^jet/', include('jet.urls', 'jet')),  # Django JET URLS
  url(r'^admin/', include(admin.site.urls)),
)
```

Στη συνέχεια, κάνουμε **migrate** ώστε να δημιουργηθούν οι απαιτούμενοι πίνακες στη βάση:

**→ python manage.py migrate jet**

Μετά την εγκατάσταση του Jet, για την ενεργοποίηση του **Jet Dashboard** απαιτούνται τα εξής βήματα:

**settings.py:**

```
INSTALLED_APPS = (
   'jet.dashboard',
   'jet',
   'django.contrib.admin',
)
```

Τα url-patterns προστίθενται στο αρχείο urls.py του καταλόγου του project:

**urls.py:**

```
urlpatterns = patterns(
   url(r'^jet/', include('jet.urls', 'jet')),  # Django JET URLS
   url(r'^jet/dashboard/', include('jet.dashboard.urls', 'jet-dashboard')),
   # Django JET   dashboard URLS
   url(r'^admin/', include(admin.site.urls)),)
```

Έπειτα, κάνουμε migrate ώστε να δημιουργηθούν οι απαιτούμενοι πίνακες στη βάση:

Εντολή εγκατάστασης

**→ python manage.py migrate jet**

Τέλος, για τη χρήση των διαφόρων θεμάτων, που προσφέρει το Jet, χρειάζεται να προστεθούν τα ακόλουθα στο αρχείο **settings.py** του καταλόγου project:

**settings.py:**

```
JET_THEMES = [
     {
      'theme': 'default', # theme folder name
      'color': '#47bac1', # color of the theme's button in user menu
      'title': 'Default' # theme title
```

```
    },
    {
     'theme': 'green',
     'color': '#44b78b',
     'title': 'Green'
    },
    {
     'theme': 'light-green',
     'color': '#2faa60',
     'title': 'Light Green'
    },
    {
     'theme': 'light-violet',
     'color': '#a464c4',
     'title': 'Light Violet'
    },


    {
     'theme': 'light-blue',
     'color': '#5EADDE',
     'title': 'Light Blue'
    },
    {
     'theme': 'light-gray',
     'color': '#222',
     'title': 'Light Gray'
     }
]
```

Αυτά είναι μερικά από τα χαρακτηριστικά και τις δυνατότητες του Django Jet. Για περισσότερες πληροφορίες μπορεί να απευθυνθεί κανείς στον επίσημο ιστότοπο: https://jet.readthedocs.io/en/latest/index.html [30].

### 4. 6 Django  Template Check

Με τη χρήση του Django template, παρατηρείται πως είναι αρκετά εύκολο να γίνουν συντακτικά λάθη. Για το λόγο αυτό υπάρχει το **django-template-check** package, που, όπως προδίδει και το όνομά του, ελέγχει συντακτικά λάθη που ενδέχεται να προκύψουν κατά τη διαδικασία της συνεχούς ενσωμάτωσης (continuous integration) [31]. Για την εγκατάστασή του απαιτούνται οι ακόλουθες ενέργειες:
Εντολή εγκατάστασης:

**→ pip install django-template-check**

**settings.py:**

```
INSTALLED_APPS = (
   'django_template_check',
)
```

Αφότου ολοκληρωθεί η εγκατάσταση του πακέτου, εκτελείται η ακόλουθη εντολή ώστε να τεθεί σε εφαρμογή:

**→ python manage.py templatecheck**

## 4. 7 Django CORS

Ένα από τα πιο σημαντικά πακέτα της εφαρμογής είναι το **Cross-Origin Resource Sharing**. Πρόκειται για έναν ειδικό μηχανισμό, που λειτουργεί μεταξύ δύο διαφορετικών Domain Servers, επιτρέποντας τη μεταφορά των **headers** από τον ένα στον άλλο. Στην ουσία, χρησιμοποιώντας επιπρόσθετα headers πάνω από το HTTP επιτρέπεται στον web browser του ενός domain (origin) να παραχωρήσει σε μία εφαρμογή τα απαραίτητα δικαιώματα σε επιλεγμένους πόρους, του άλλου domain [32] [33].

Για την εγκατάσταση του πακέτου απαιτούνται τα εξής:

**→ pip install django-cors-headers**

**settings.py:**

```
INSTALLED_APPS = (
    'corsheaders',
)
```

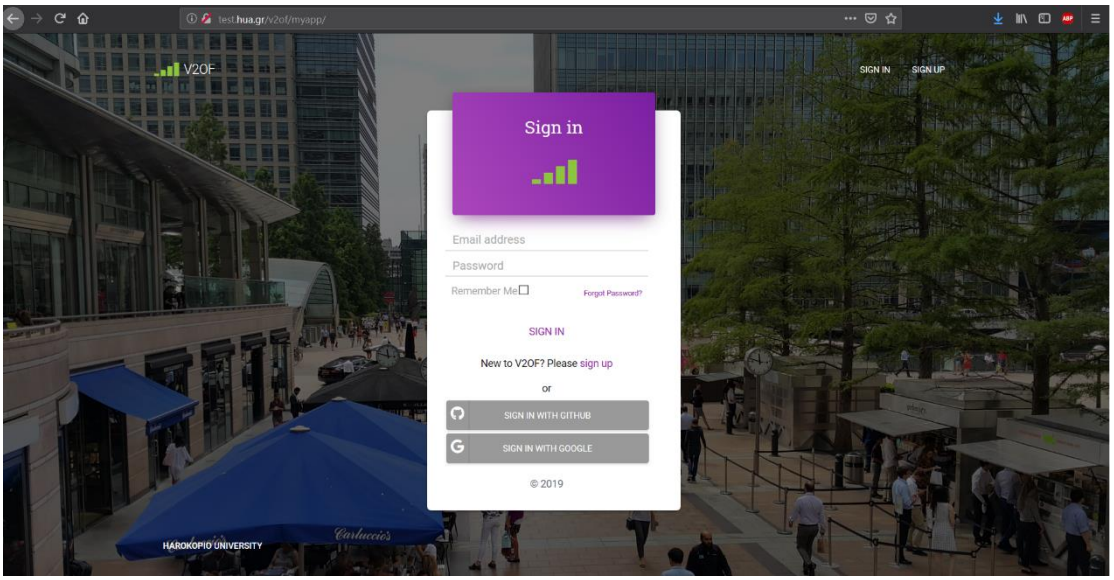
Στο ίδιο αρχείο θα πρέπει να προστεθεί και η MIDDLEWARE κλάση:

```
MIDDLEWARE = [
# Or MIDDLEWARE_CLASSES on Django < 1.10
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',


]
```

# 5. Οδηγός Χρήσης της Εφαρμογής

Αρχικά θα πρέπει να αναφέρουμε πως η εφαρμογή φιλοξενείται στον server του Χαροκοπείου Πανεπιστημίου. Επομένως, προκειμένου να πλοηγηθεί κανείς στην εφαρμογή, αρκεί να πληκτολογήσει την ακόλουθη διεύθυνση: http://test.hua.gr/v2of/myapp/ στον browser της επιλογής του.

## 5.1 Login Page




Εικόνα 19

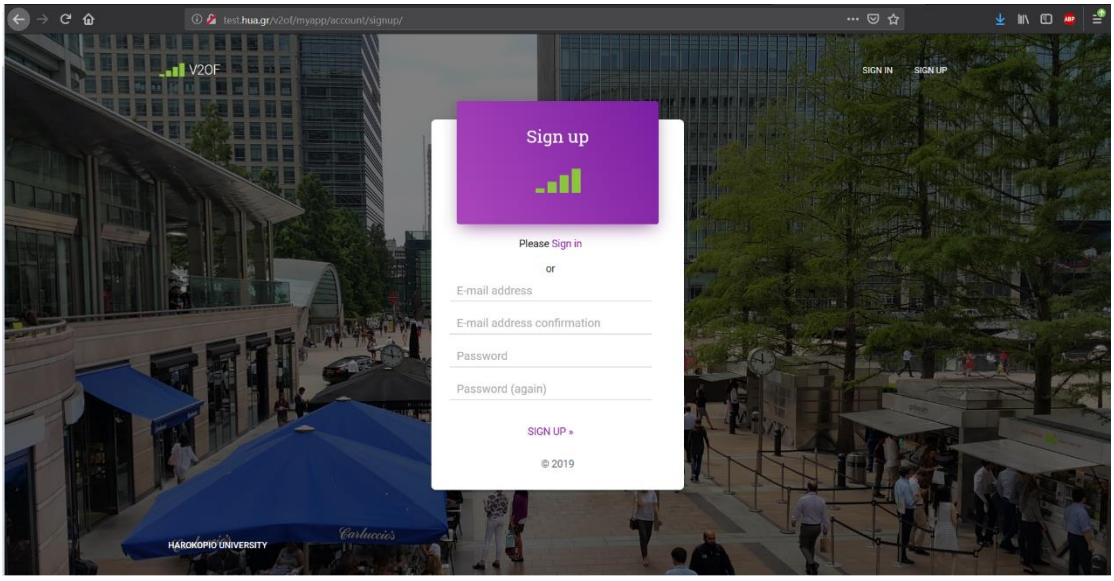
Η πρώτη επαφή, που έχει ο χρήστης με την εφαρμογή, εμφανίζεται στην πιο πάνω εικόνα. Εκεί, του δίνεται η δυνατότητα να κάνει login, εφόσον διαθέτει κάποιον ήδη υπάρχοντα λογαριασμό. Σε αντίθετη περίπτωση, μπορεί, είτε να δημιουργήσει έναν καινούριο, πατώντας το κουμπί **Sign up**, είτε να κάνει login με κάποιο λογαριασμό, που διαθέτει στην πλατφόρμα της **Google** ή του **Github** αντίστοιχα.

## 5.2 Σελίδα Εγγραφής (Sign up Page)

Στην περίπτωση που κάποιος χρήστης θελήσει να δημιουργήσει έναν καινούριο λογαριασμό πατώντας το κουμπί **Sign up** θα μεταφερθεί στο περιβάλλον, το οποίο εμφανίζεται στην πιο κάτω εικόνα. Εκεί θα του ζητηθούν τα εξής:

- E-mail address
- E-mail address confirmation
- Password
- Password (again)

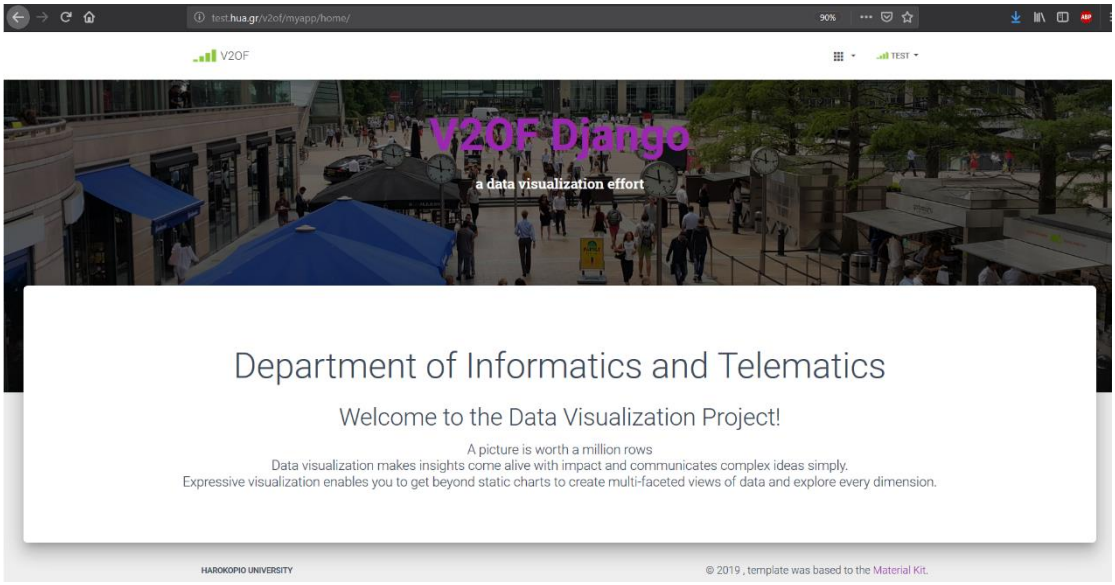
Συμπληρώνοντας τη φόρμα με τα παραπάνω στοιχεία μπορεί να κάνει εύκολα login στην εφαρμογή.




Εικόνα 20

## 5.3 Αρχική Σελίδα (Home Page)

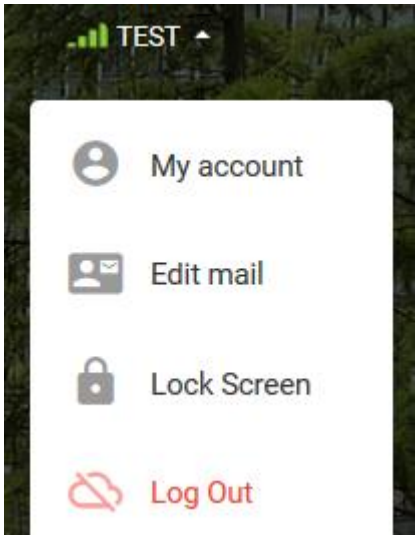
Εν συνεχεία, ο χρήστης μεταφέρεται στην αρχική σελίδα της εφαρμογής, όπου από το μενού που βρίσκεται πάνω δεξιά στην εικόνα μπορεί είτε να διαχειριστεί το **προφίλ** του, είτε να μεταβεί στο **Dashboard**. Στην εικόνα, που ακολουθεί, φαίνεται η αρχική σελίδα.




Εικόνα 21

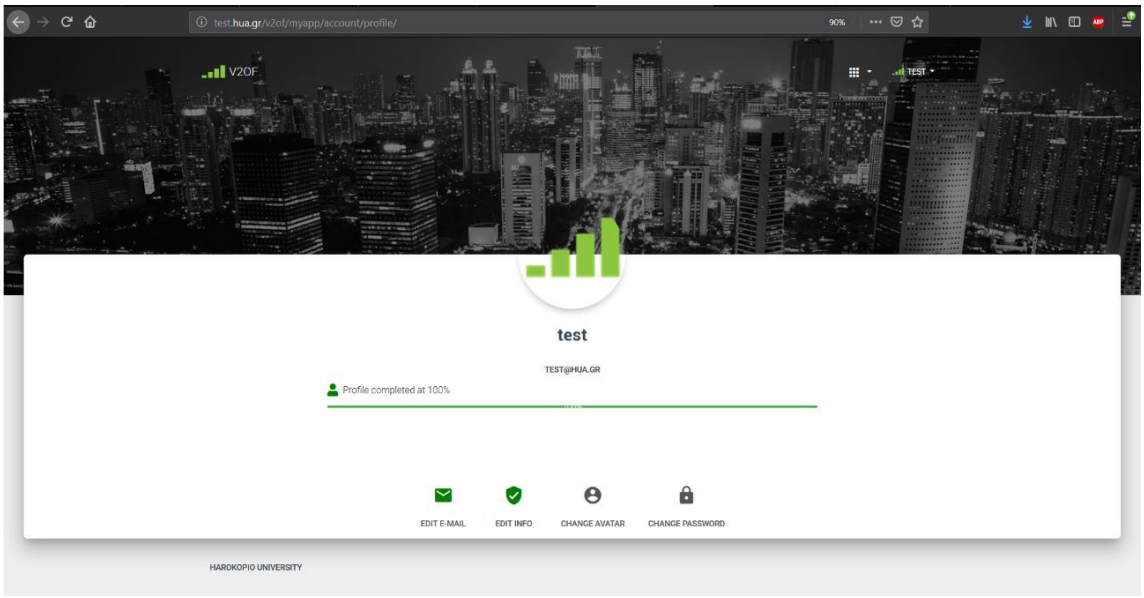
## 5.4 Διαχείριση Λογαριασμού (Account Management)

Ο χρήστης έχει τη δυνατότητα να διαχειριστεί το προφίλ του πατώντας το κουμπί **My account,** όπως φαίνεται στην εικόνα που ακολουθεί.




Εικόνα 22

Μεταφέρεται, έτσι, στη σελίδα του προφίλ, απ’ όπου μπορεί να πληροφορηθεί για τα ήδη δηλωμένα στοιχεία, ακόμα και να τα τροποποιήσει.

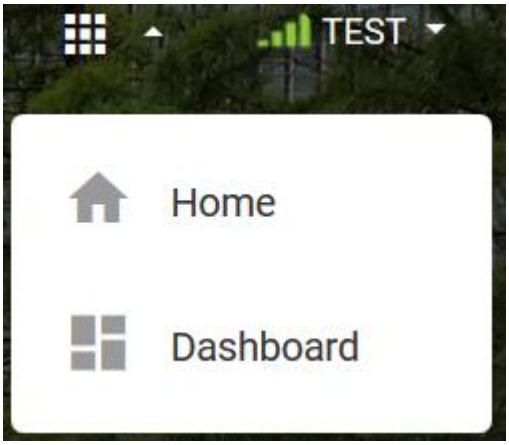



Εικόνα 23

- **Edit E-mail**: Επεξεργασία του E-mail του χρήστη.
- **Edit Info**: Από εδώ ο χρήστης μπορεί να αλλάξει τα προσωπικά του στοιχεία.
- **Change Avatar**: Εδώ ο χρήστης μπορεί είτε να προσθέσει μία φωτογραφία για το προφίλ του είτε να την αντικαταστήσει με κάποια νέα.
- **Change Password**: Εδώ ο χρήστης έχει τη δυνατότητα να αντικαταστήσει τον παλιό κωδικό του με ένα νέο.

## 5.5 Dashboard

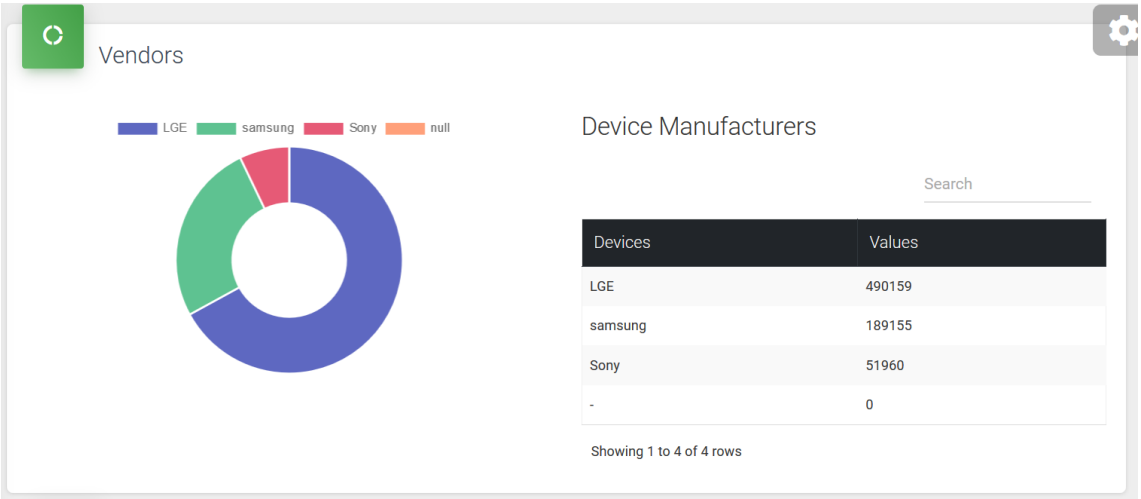
Πρόκειται για τη σελίδα που περιέχει την πιο σημαντική πληροφορία της εφαρμογής και στην οποία ο χρήστης μπορεί να έχει πρόσβαση πατώντας το κουμπί, που εμφανίζεται στην εικόνα 24. Στη συνέχεια παρουσιάζονται αναλυτικότερα οι αντίστοιχες καρτέλες της σελίδας.




Εικόνα 24

### 5.5.1 Vendors

Με τη χρήση του διαγράμματος **pie chart** σε αυτήν την καρτέλα εμφανίζονται οι κατασκευαστές των κινητών συσκευών, από τα οποία ελήφθησαν οι μετρήσεις. Τα αποτελέσματα των μετρήσεων εμφανίζονται και σε μορφή πίνακα.




Εικόνα 25

Ενδεικτικά, γι' αυτή την καρτέλα, παρουσιάζεται ακολούθως ο τρόπος με τον οποίο δημιουργήθηκε τεχνικά το διάγραμμα και ο πίνακας των δεδομένων. Τα αντίστοιχα κομμάτια κώδικα είναι τα εξής:

**Python**:

**views.py**:

```
class VendorsView(viewsets.ModelViewSet):
   authentication_classes = (JSONWebTokenAuthentication, SessionAuthentication,
  BasicAuthentication)
   permission_classes = (IsAuthenticatedOrReadOnly,)
   queryset =
Measurements.objects.values('devicemanufacturer').annotate(value=Count('device
manufacturer'))

   serializer_class = VendorsSerializer

   def get_queryset(self):
      """ allow rest api to filter by Device Manufacturer """
      queryset =
Measurements.objects.values('devicemanufacturer').annotate(value=Count('device
manufacturer'))

 devicemanufacturer = self.request.query_params.get('devicemanufacturer', None)
      if devicemanufacturer is not None:
         queryset = queryset.filter(devicemanufacturer=devicemanufacturer)
      return queryset
```

**Javascript**:

```
var GetChartData = function () {
   $.ajax({
      url: '../vendorStats/',
      method: 'GET',
      dataType: 'json',
      beforeSend: function (xhr) {
         xhr.setRequestHeader ("Authorization", "Basic " + btoa(username + ":" +
        password));
      },
      success: function (d) {
        var labels = d.results.map(function(e) {
           return e.key;
        });
        var data = d.results.map(function(e) {
           return e.value;
        });
        var ctx = document.getElementById("myChart");
        var config = {
         type: 'doughnut',
```

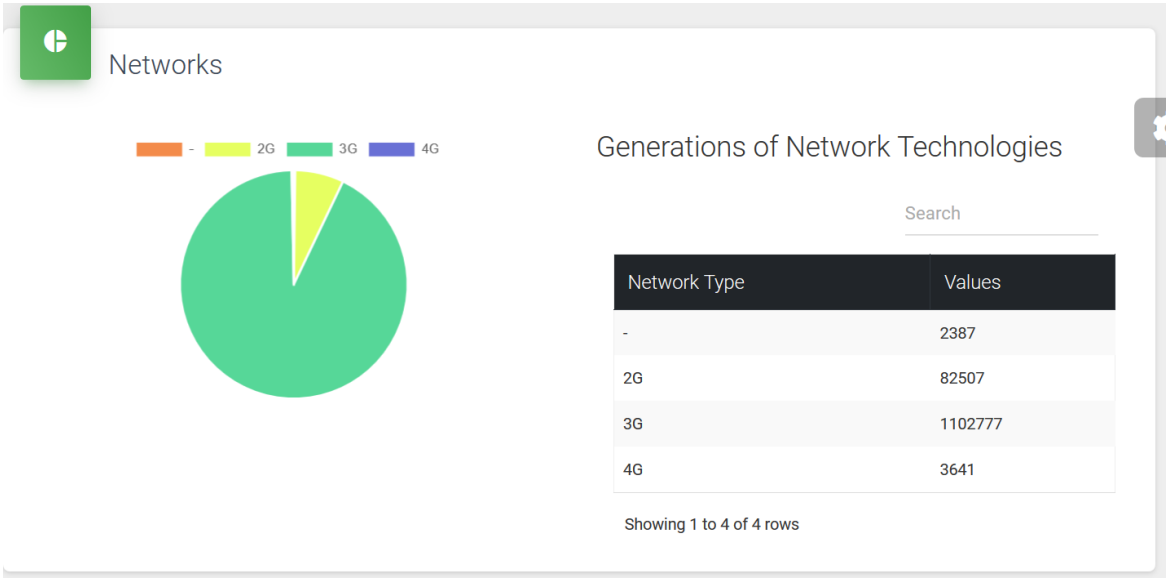
```
        data: {
          labels: labels,
          datasets: [{
            //label: 'Graph Line',
            data: data,
            backgroundColor: [
              'rgba(25, 40, 166, 0.7)', //LGE
              'rgba(25, 168, 98, 0.7)',
              'rgba(220,20,60 ,0.7)',
              'rgba(255,160,122 ,1)', //Samsung
            ],
          }]
        }
      };
           var chart = new Chart(ctx, config);

           $('#VendorsTable').bootstrapTable({
                 data: [{
                   labels: labels[0],
                   data: data[0]
                 }, {
                   labels: labels[1],
                   data: data[1]

                 }, {
                   labels: labels[2],
                   data: data[2]

                 }, {
                   labels: labels[3],
                   data: data[3]

                 }]
           })
      }
   });
};

$(document).ready(function() {
   GetChartData();
});
```

**HTML**:

```
</div>
      <h4 class="card-title">Vendors</h4>
</div>
<div class="card-body">
     <div class="row">
          <div class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
               <br>
               <canvas id="myChart"></canvas>
          </div>
          <div class="col-lg-6 col-md-6 col-sm-6 col-xs-6">
            <div class="table-responsive">
                 <h3> Device Manufacturers</h3>
                 <table class="table table-bordered table-hover table-striped"
                          data-search="true"
                          data-pagination="true"
                          id="VendorsTable">
                     <thead class="thead-dark">
                     <tr>
                        <th data-field="labels">Devices</th>
                        <th data-field="data">Values</th>
                     </tr>
                     </thead>
                 </table>
            </div>
          </div>
     </div>
</div>
```

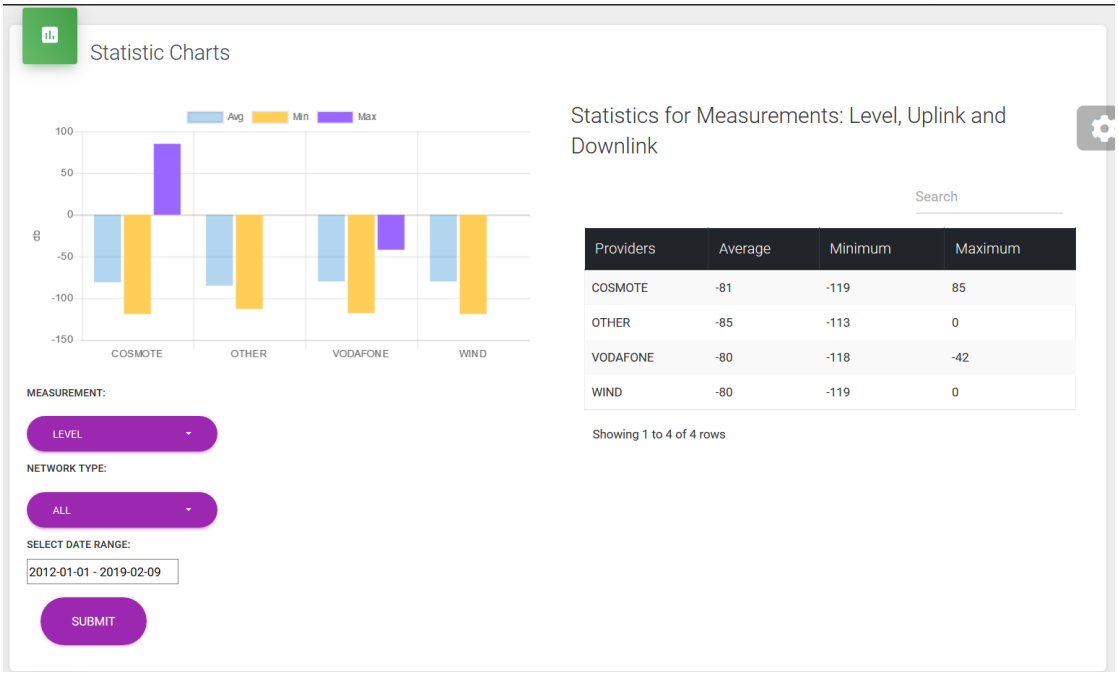
### 5.5.2 Networks

Σε αυτήν την καρτέλα, εμφανίζονται οι αντίστοιχες γενιές των κινητών δικτύων με τη μορφή διαγράμματος **pie chart**. Στον πίνακα παραπλεύρως παρουσιάζονται τα αντίστοιχα αποτελέσματα.



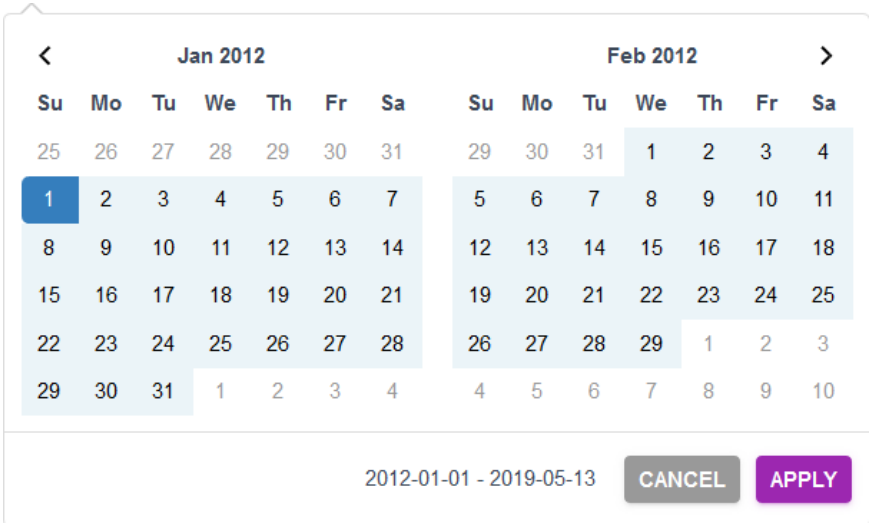

Εικόνα 26

### 5.5.3 Statistic Charts




Εικόνα 27

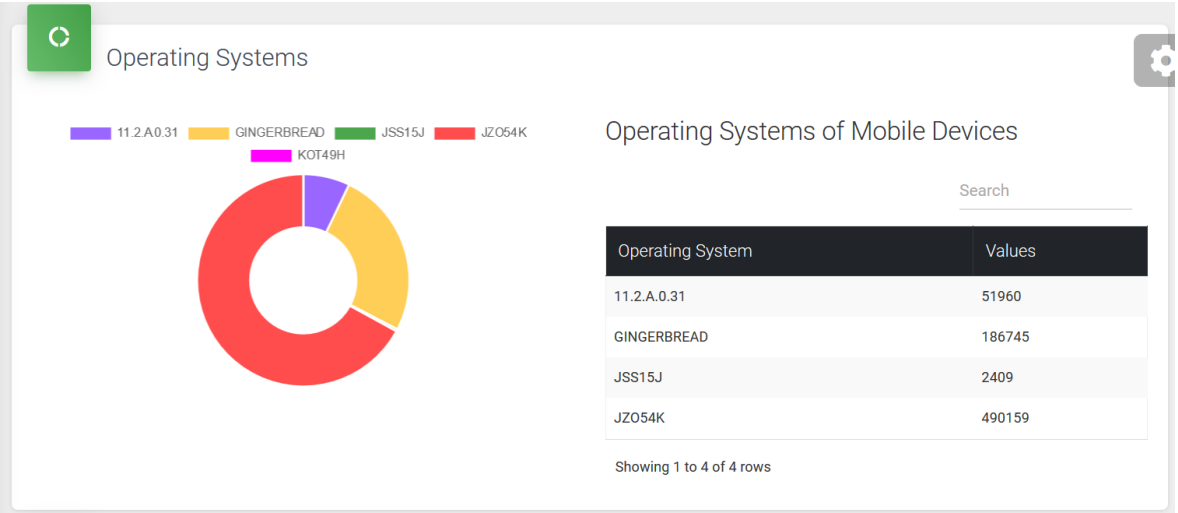
Η συγκεκριμένη καρτέλα είναι ίσως η πιο σημαντική. Ο χρήστης έχει τη δυνατότητα να επιλέξει μεταξύ των μετρήσεων **Level**, **Uplink**, **Downlink** (αφορούν στην ισχύ του σήματος) σε συνδυασμό με τον τύπο δικτύου (**2G**, **3G**, **4G**), καθώς και με το εύρος της ημερομηνίας. Αυτές απεικονίζονται με τη μορφή διαγραμμάτων **bar chart** σε συνδυασμό με τους εκάστοτε τηλεπικοινωνιακούς παρόχους. Για την επιλογή της ημερομηνίας χρησιμοποιήθηκε η βιβλιοθήκη **daterangepicker**, όπως εμφανίζεται στην εικόνα. Στο σημείο αυτό αξίζει να σημειωθεί πως η εφαρμογή λαμβάνει μετρήσεις από τη βάση, επομένως ο χρήστης δεν μπορεί να επιλέξει ημερομηνία παλαιότερη από αυτή που έχει οριστεί εξαρχής (**2012-01-01**).




Εικόνα 28

### 5.5.4 Operating Systems

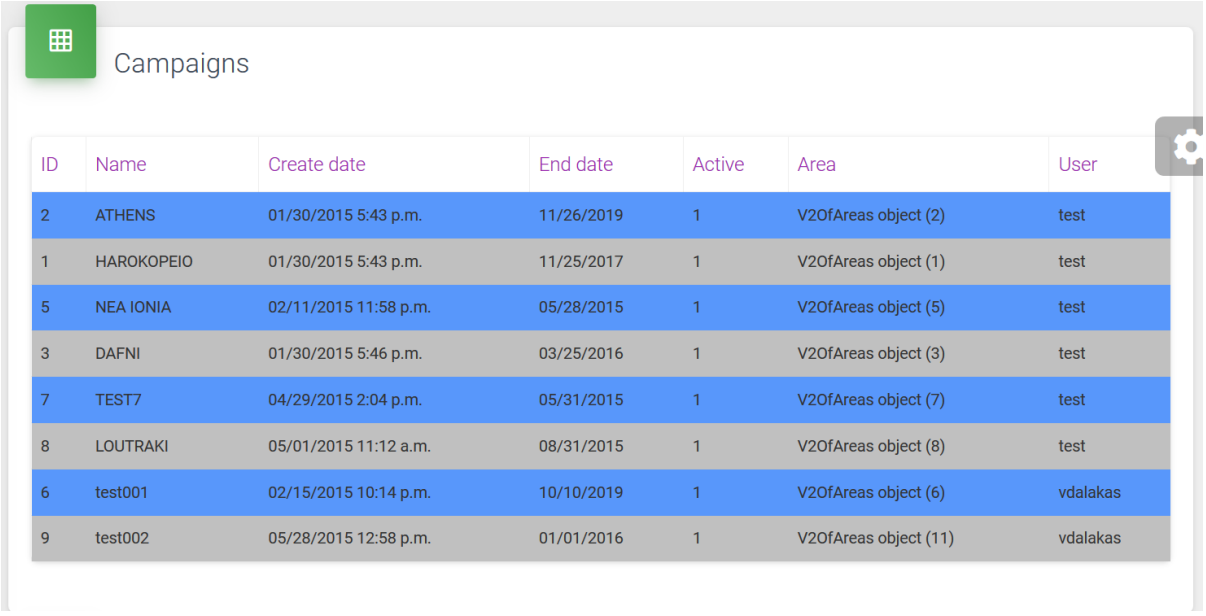
Εδώ, παρουσιάζονται τα λειτουργικά συστήματα (λογισμικό) των κινητών συσκευών από τις οποίες ελήφθησαν οι μετρήσεις, όπως και ο αντίστοιχος πίνακας που τις απεικονίζει.




Εικόνα 29

### 5.5.5 Campaigns

Στην καρτέλα αυτή εμφανίζονται ορισμένες περιοχές ενδιαφέροντος προς μέτρηση, οι λεγόμενες καμπάνιες. Εμφανίζονται, επίσης, ο χρήστης ο οποίος δημιούργησε μία καμπάνια, η αντίστοιχη περιοχή και η ημερομηνία.




Εικόνα 30

### 5.5.6 MNOS

Στην καρτέλα που ακολουθεί, εμφανίζονται όλες οι μετρήσεις που υπάρχουν στη βάση δεδομένων και αφορούν στον εκάστοτε πάροχο τηλεπικοινωνίας.

MNOs

| Id | Geomcol1 | Timestamp | Lon | Lat | Level | Speed | Operatorname | Mcc | Mnc | Node | Cellid |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1156868 | <memory at 0x7fe6cd2334c8> | 07/07/2014 8:42 a.m. | 23.82670 | 38.00380 | – | 3.0 | – | – | – | – | – |
| 1177418 | <memory at 0x7fe6cd233588> | 07/09/2014 7:37 a.m. | 23.70145 | 37.96340 | -93 | 28.0 | COSMOTE | 202 | 01 | 520 | 1531 |
| 1191314 | <memory at 0x7fe6cd233708> | 12/10/2018 1:37 p.m. | 23.70089 | 37.96227 | -113 | 0.0 | GR COSMOTE | 202 | 01 | – | 702468 |
| 1189659 | <memory at 0x7fe6cd233888> | 10/19/2013 9:02 a.m. | 23.82462 | 38.01712 | -87 | 3.0 | GR_COSMOTE | 202 | 01 | 522 | 27893 |
| 1191320 | <memory at 0x7fe6cd233a08> | 02/13/2019 7:46 p.m. | -3.62538 | 37.19196 | -113 | 0.0 | Movistar | 214 | 07 | – | 461399 |

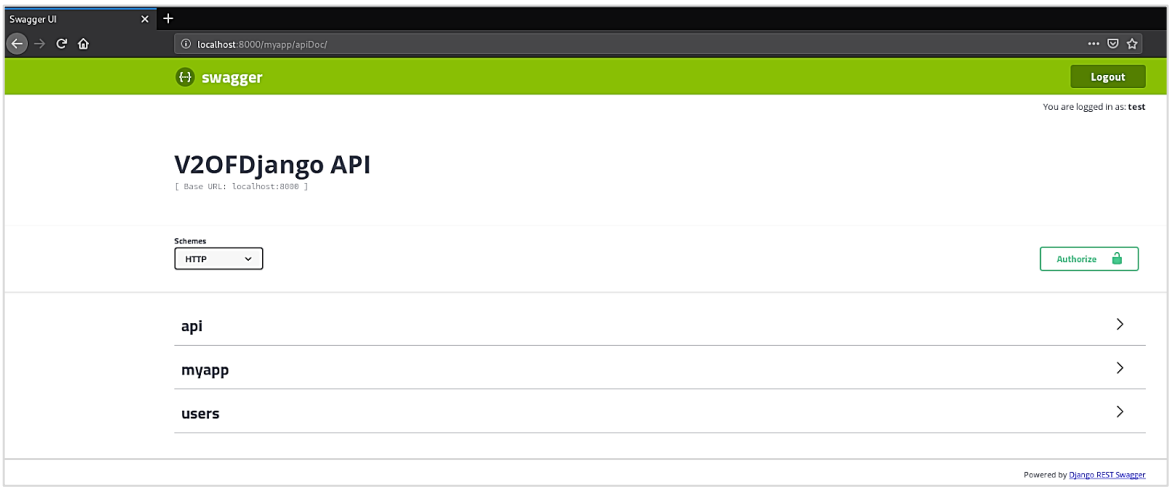
Εικόνα 31

# 6. API Documentation

Ο τρόπος, με τον οποίο μπορεί ένας προγραμματιστής να αλληλεπιδράσει με ένα **Application Programming Interface** (API) και να το χρησιμοποιήσει, δίνεται μέσω του εγχειριδίου του. Αυτό παρέχει όλες τις απαραίτητες πληροφορίες, που χρειάζεται ο χρήστης, από τις κλάσεις που χρησιμοποιούνται μέχρι και το περιεχόμενο που επιστρέφει, τον τύπο των αιτημάτων (**GET**, **POST**, **PUT**, **DELETE**) κ.α. Υπάρχουν διαφόρων ειδών εργαλεία (tools) που αναλαμβάνουν την διαχείριση ενός API, από τα οποία σημειώνεται το ευρέως γνωστό Postman. Για την εφαρμογή αυτή χρησιμοποιήθηκε το **Swagger** [34], το οποίο είναι εγκατεστημένο στα **INSTALLED_APPS** (settings.py) του προγράμματος με την ακόλουθη εντολή

**→ pip install django-rest-swagger**

**Swagger**




Εικόνα 32

Επίσης, θα πρέπει να προστεθούν τα ακόλουθα:

**settings.py:**

```
INSTALLED_APPS = [
   'rest_framework_swagger',
]
```

**views.py:**

```
from django.conf.urls import url
from rest_framework_swagger.views import get_swagger_view
schema_view = get_swagger_view(title='Pastebin API')
```

**urls.py:**

```
urlpatterns = [
    url(r'^$', schema_view)
]
```

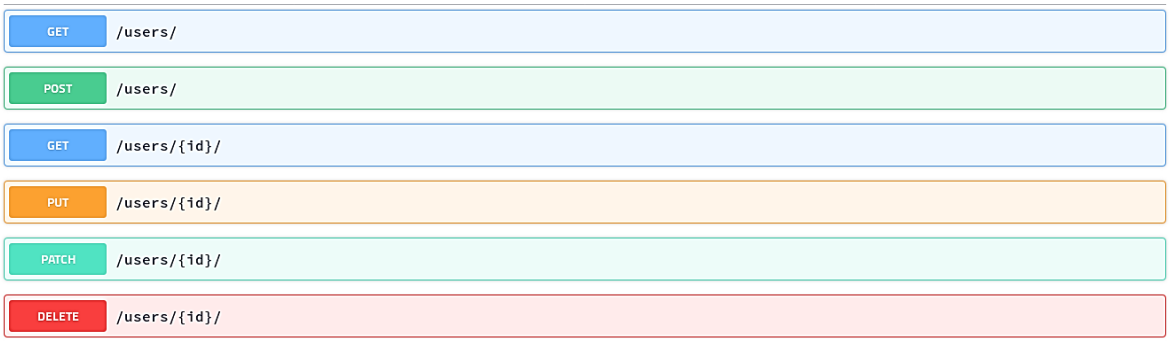
Το παραπάνω url προστίθεται στον κατάλογο του project.

**My App API**:

Στην εικόνα, που ακολουθεί, εμφανίζονται τα URL του Application Server, του οποίου τα αποτελέσματα παρατίθενται στο **παράρτημα Β** σε μορφή JSON.


Εικόνα 33

**Users API**:




Εικόνα 34

# 7. Συμπεράσματα

Κατά τη διάρκεια της υλοποίησης της εφαρμογής ήρθαμε αντιμέτωποι με ποικίλα προβλήματα. Αρχικά, η βάση δεδομένων έπρεπε να τροποποιηθεί, ώστε να μπορέσουμε να εντάξουμε τους χρήστες μέσα στην εφαρμογή, όπως επίσης και στη σελίδα Admin του Django. Τα διάφορα προβλήματα στο Frontend, όπως η εμφάνιση μεγάλου όγκου των δεδομένων, επιλύθηκαν χρησιμοποιώντας **AJAX** κλήσεις (calls). Όσον αφορά στο Backend χρησιμοποιήθηκε **pagination**, ώστε να είναι εμφανή τα αποτελέσματα των μετρήσεων που βρίσκονται στη βάση. Στο σημείο αυτό είναι σημαντικό να αναφέρουμε πως, χωρίς το πακέτο **CORS Headers,** δεν μπορούσαν να πραγματοποιηθούν **AJAX calls** και κατά συνέπεια τα αποτελέσματα στην πλευρά του χρήστη δεν εμφανίζονταν. Αντιμετωπίσαμε, επιπλεόν, δυσκολία στο να εμφανίσουμε τα αποτελέσματα των γραφημάτων στους αντίστοιχους πίνακες. Αρχικά, προσπαθήσαμε να αποδώσουμε τα αποτελέσματα σε πίνακες με τη χρήση των **Django Tables**, όμως κάτι τέτοιο δεν επετεύχθη. Καταλήξαμε, λοιπόν, στο να χρησιμοποιήσουμε τη βιβλιοθήκη, που παρέχεται στο **Bootstrap Framework**. Στο σημείο αυτό, αξίζει να επισημάνουμε πως οι μετρήσεις, που προβάλλονται στο χρήστη, προέρχονται από την παλιά βάση. Αντιθέτως, το **authentication** του χρήστη πραγματοποιείται από την καινούρια βάση. Σε περίπτωση που συμπληρωθεί ο απαιτούμενος αριθμός των μετρήσεων, θα γίνει **migration** εξολοκλήρου στην καινούρια βάση.

## 7.1 Επεκτάσεις

Οι επεκτάσεις, που μπορούν να υλοποιηθούν στην εφαρμογή, αφορούν κυρίως στην ένταξη της τελευταίας έκδοσης του **openlayers** στο Frontend της εφαρμογή, όπως επίσης και στο **Admin** περιβάλλον του Django Framework. Τέλος, οι μετρήσεις που βρίσκονται στη βάση δεδομένων θα πρέπει να είναι συνυφασμένες σύμφωνα με το **ISO 19156:2011**.

# Βιβλιογραφία


[1] 3GPP. Ανακτήθηκε Δεκέμβριο, 2018 από το Google: http://www.3gpp.org/

[2] Κορωνιώτης, Ν. (2014). *Ανάπτυξη εφαρμογής και δικτυακής πλατφόρμας μέτρησης απόδοσης ασύρματων δικτύων για συσκευές Android*. Χαροκόπειο Πανεπιστήμιο, Αθήνα.

[3] Αθανάσιος Κ. (2014). *Ανάπτυξη εφαρμογής και δικτυακής πλατφόρμας (sensing platform) μέτρησης απόδοσης ασύρματων δικτύων για συσκευές iOS*. Χαροκόπειο Πανεπιστήμιο, Αθήνα.

[4] Μωυσιάδης Α. (2015). *Εφαρμογή συλλογής γεωχωρικών δεδομένων απόδοσης ασύρματων δικτύων με χρήση υπηρεσίας παρασκηνίου για έξυπνες συσκευές*.

[5] Gerti N. (2015). *Διαδικτυακή Πλατφόρμα Συντονισμού για Σύντηξη Γεωχωρικών Δεδομένων από Μετρήσεις Απόδοσης Ασύρματων Δικτύων*

[6] Λοντόρφος Ν. (2017). *Ανάπτυξη Διαδικτυακής Πλατφόρμας με σκοπό την παροχή Επιχειρηματικής Ευφυΐας από Μετρήσεις Γεωχωρικών Δεδομένων με χρήση τεχνολογιών Angular2*. Χαροκόπειο Πανεπιστήμιο, Αθήνα.

[7] Python (programming language). Ανακτήθηκε Νοέμβριο, 2018 από το Wiki: https://en.wikipedia.org/wiki/Python_(programming_language)

[8] Δημήτριος Α. Καρολίδης (2016). *Μαθαίνετε Εύκολα Python*. Αθήνα: Εκδόσεις Καρολίδη.

[9] Daniel Rubio (2017). *Beginning Django: Web Application Development and Deployment with Python.* Mexico: Publications Apress

[10] The Model-View-Controller Design Pattern. (Νοέμβριο, 2018). Ανακτήθηκε από το Google: https://djangobook.com/model-view-controller-design-pattern/

[11] Django (web framework). Ανακτήθηκε Νοέμβριο, 2018 από το Wiki: https://en.wikipedia.org/wiki/Django_(web_framework)

[12] Django. The web framework for perfectionists with deadlines (Django' s cache framework). Ανακτήθηκε Νοέμβριο, 2018 από https://docs.djangoproject.com/en/2.1/topics/cache/

[13] Django. The web framework for perfectionists with deadlines (Django' s cache framework). Ανακτήθηκε Νοέμβριο, 2018 από https://docs.djangoproject.com/en/2.1/topics/security/

[14] Django. The web framework for perfectionists with deadlines (Django' s cache framework). (FAQ: General). Ανακτήθηκε Νοέμβριο, 2018 από https://docs.djangoproject.com/en/2.1/faq/general/#does-django-scale

[15] Django. The web framework for perfectionists with deadlines (Django' s cache framework). Ανακτήθηκε Νοέμβριο, 2018 από https://www.djangoproject.com/start/overview/

[16] Bootstrap (front-end framework). Ανακτήθηκε Νοέμβριο, 2018 από το Wiki: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

[17] REST. Representational State Transfer. Ανακτήθηκε Δεκέμβριο, 2018 από το Wiki: https://en.wikipedia.org/wiki/Representational_state_transfer

[18] Django Rest Framework. Ανακτήθηκε Νοέμβριο, 2018 από το Google: https://www.django-rest-framework.org/

[19] PostgreSQL. Ανακτήθηκε Νοέμβριο, 2018 από το Google: https://en.wikipedia.org/wiki/PostgreSQL

[20] pip (package manager). Ανακτήθηκε Νοέμβριο, 2018 από το Wiki: https://en.wikipedia.org/wiki/Pip_(package_manager)

[21] User Guide (Usage). Ανακτήθηκε Νοέμβριο, 2018 από το Google: https://virtualenv.pypa.io/en/latest/userguide/

[22] Writing your first Django app, part 2. Ανακτήθηκε Νοέμβριο, 2018 από το Google: https://docs.djangoproject.com/en/2.1/intro/tutorial02/

[23] Chapter 18: Integrating with Legacy Databases and Applications. Ανακτήθηκε Νοέμβριο, 2018 από το Google: https://django-book.readthedocs.io/en/latest/chapter18.html

[24] Installing Geospatial Libraries. Ανακτήθηκε Δεκέμβριο, 2018 από το Google: https://docs.djangoproject.com/en/2.2/ref/contrib/gis/install/geolibs/

[25] PyCharm. Ανακτήθηκε Δεκέμβριο, 2018 από το Wiki: https://en.wikipedia.org/wiki/PyCharm

[26] Django allauth (Welcome to django-allauth). Ανακτήθηκε Νοέμβριο, 2018 από το Google: https://django-allauth.readthedocs.io/en/latest/index.html

[27] Google API Console. Ανακτήθηκε Νοέμβριο, 2018 από το Google: https://developers.google.com/identity/protocols/OAuth2

[28] django-avatar. Ανακτήθηκε Νοέμβριο, 2018 από το Google: https://django-avatar.readthedocs.io/en/latest/

[29] Django hstore-documentation. Ανακτήθηκε Νοέμβριο, 2018 από το Google: https://django-hstore.readthedocs.io/en/latest/

[30] Django JET. Ανακτήθηκε Νοέμβριο, 2018 από το Google: https://jet.readthedocs.io/en/latest/

[31] Python Software Foundation (django-template-check 0.3.1). Ανακτήθηκε Νοέμβριο, 2018 από το Google: https://pypi.org/project/django-template-check/

[32] Cross-Origin Resource Sharing. Ανακτήθηκε Ιανουάριο, 2019 από το Google: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

[33] django-cors-headers. Ανακτήθηκε Ιανουάριο, 2019 από το Github: https://github.com/ottoyiu/django-cors-headers/

[34] Django REST Swagger. Ανακτήθηκε Ιανουάριο, 2019 από το Google: https://django-rest-swagger.readthedocs.io/en/latest/

# Παραρτήματα

## Παράρτημα Α : SQL – Python Queries

https://github.com/mKorniotakis/V2ofDjango/blob/7681b0bd652424af69fff0dbb60d8faa7d133477/mysite/myapp/views.py#L125-L650

## Παράρτημα Β : Αποτελέσματα API

- **/myapp/providers/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/providers/

***Server response***

200

**Response body**


```
{
 "count": 4,
 "next": null,
 "previous": null,
 "results": [
  {
   "key": "COSMOTE",
   "value": 671596
  },
  {
   "key": "OTHER",
   "value": 1000
  },
  {
   "key": "VODAFONE",
   "value": 429063
  },
  {
```

```
    "key": "WIND",
    "value": 89653
   }
  ]
}
```


- **myapp/downlinkStats/**

  **Request: GET**

  **Request URL**

  http://localhost:8000/myapp/downlinkStats/

  **Server response**

  200

  **Response body**


```
{
 "count": 4,
 "next": null,
 "previous": null,
 "results": [
  {
   "key": "COSMOTE",
   "avg": 183,
   "min": 1,
   "max": 12036
  },
  {
   "key": "OTHER",
   "avg": 28,
   "min": 0,
   "max": 1261
  },
  {
   "key": "VODAFONE",
   "avg": 99,
```

```
    "min": 1,
    "max": 16638
   },
   {
    "key": "WIND",
    "avg": 191,
    "min": 0,
    "max": 3651
   }
 ]
}
```


- **/myapp/downlinkStats/{network_type}/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/downlinkStats/2g/

**Server response**

200

**Response body**


```
{
 "count": 3,
 "next": null,
 "previous": null,
 "results": [
  {
   "key": "COSMOTE",
   "avg": 30,
   "min": 1,
   "max": 206
  },
  {
   "key": "VODAFONE",
```

```
    "avg": 31,
    "min": 1,
    "max": 800
   },
   {
    "key": "WIND",
    "avg": 23,
    "min": 1,
    "max": 230
   }
 ]
}
```


**Request: GET**

**Request URL**

http://localhost:8000/myapp/downlinkStats/3g/

**Server response**

200

**Response body**


```
{
 "count": 4,
 "next": null,
 "previous": null,
 "results": [
  {
   "key": "COSMOTE",
   "avg": 190,
   "min": 1,
   "max": 12036
  },
  {
   "key": "OTHER",
   "avg": 54,
```

```
    "min": 1,
    "max": 1261
   },
   {
    "key": "VODAFONE",
    "avg": 120,
    "min": 1,
    "max": 16638
   },
   {
    "key": "WIND",
    "avg": 214,
    "min": 0,
    "max": 3651
   }
  ]
}
```


**Request: GET**

**Request URL**

http://localhost:8000/myapp/downlinkStats/4g/

**Server response**

200

**Response body**


```
{
  "count": 3,
  "next": null,
  "previous": null,
  "results": [
   {
    "key": "COSMOTE",
    "avg": null,
    "min": null,
```

```
    "max": null
   },
   {
    "key": "OTHER",
    "avg": null,
    "min": null,
    "max": null
   },
   {
    "key": "WIND",
    "avg": null,
    "min": null,
    "max": null
   }
  ]
}
```


Στο σημείο αυτό, αξίζει να σημειωθεί πως στα **{start}** και **{end}** επιλέχθηκαν τυχαία δύο τιμές (ημερομηνίες) όπως επίσης και στο **{network_type}** εφαρμόστηκε ενδεικτικά μία τιμή (**2g**) προκειμένου να εξυπηρετηθεί ο σκοπός της επιστροφής των αποτελεσμάτων.

- **/myapp/downlinkStats/{start} – {end}/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/downlinkStats/2012-01-01 – 2018-01-01/

**Server response**

200

**Response body**


```
{
  "count": 4,
  "next": null,
  "previous": null,
  "results": [
```

```
  {
    "key": "COSMOTE",
    "avg": 183,
    "min": 1,
    "max": 12036
  },
  {
    "key": "OTHER",
    "avg": 28,
    "min": 0,
    "max": 1261
  },
  {
    "key": "VODAFONE",
    "avg": 99,
    "min": 1,
    "max": 16638
  },
  {
    "key": "WIND",
    "avg": 191,
    "min": 0,
    "max": 3651
  }
 ]
}
```


- **/myapp/downlinkStats/{start} - {end}/{network_type}/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/downlinkStats/2012-01-01 – 2018-01-01/2g

**Server response**

200

**Response body**


```
{
   "count": 3,
   "next": null,
   "previous": null,
   "results": [
      {
         "key": "COSMOTE",
         "avg": 30,
         "min": 1,
         "max": 206
      },
      {
         "key": "VODAFONE",
         "avg": 31,
         "min": 1,
         "max": 800
      },
      {
         "key": "WIND",
         "avg": 23,
         "min": 1,
         "max": 230
      }
   ]
}
```


- **/myapp/levelStats/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/levelStats/

**Server response**

200

**Response body**


```
{
  "count": 4,
  "next": null,
  "previous": null,
  "results": [
    {
      "key": "COSMOTE",
      "avg": -81,
      "min": -119,
      "max": 85
    },
    {
      "key": "OTHER",
      "avg": -85,
      "min": -113,
      "max": 0
    },
    {
      "key": "VODAFONE",
      "avg": -80,
      "min": -118,
      "max": -42
    },
    {
      "key": "WIND",
```

```
    "avg": -80,
    "min": -119,
    "max": 0
   }
 ]
}
```


- **/myapp/levelStats/{network_type}/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/levelStats/2g/

***Server response***

200

**Response body**


```
{
 "count": 3,
 "next": null,
 "previous": null,
 "results": [
  {
   "key": "COSMOTE",
   "avg": -81,
   "min": -109,
   "max": -51
  },
  {
   "key": "VODAFONE",
   "avg": -87,
   "min": -113,
   "max": -51
  },
```

```
  {
    "key": "WIND",
    "avg": -83,
    "min": -113,
    "max": 0
  }
 ]
}
```


**Request: GET**

**Request URL**

http://localhost:8000/myapp/levelStats/3g/

**Server response**

200

**Response body**


```
{
 "count": 4,
 "next": null,
 "previous": null,
 "results": [
  {
    "key": "COSMOTE",
    "avg": -81,
    "min": -119,
    "max": -34
  },
  {
    "key": "OTHER",
    "avg": -85,
    "min": -101,
    "max": -51
```

```
  },
  {
    "key": "VODAFONE",
    "avg": -79,
    "min": -118,
    "max": -42
  },
  {
    "key": "WIND",
    "avg": -79,
    "min": -119,
    "max": -42
  }
 ]
}
```


**Request: GET**

**Request URL**

http://localhost:8000/myapp/levelStats/4g/

**Server response**

200

**Response body**


```
{
 "count": 3,
 "next": null,
 "previous": null,
 "results": [
  {
    "key": "COSMOTE",
    "avg": -101,
    "min": -113,
```

```
    "max": 85
   },
   {
    "key": "OTHER",
    "avg": -75,
    "min": -113,
    "max": 0
   },
   {
    "key": "WIND",
    "avg": -96,
    "min": -113,
    "max": 0
   }
  ]
}
```


- **/myapp/levelStats/{start} - {end}/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/levelStats/2012-01-01 – 2018-01-01/

**Server response**

200

**Response body**


```
{
 "count": 4,
 "next": null,
 "previous": null,
 "results": [
  {
   "key": "COSMOTE",
```

```
    "avg": -81,
    "min": -119,
    "max": -34
   },
   {
    "key": "OTHER",
    "avg": -85,
    "min": -113,
    "max": -51
   },
   {
    "key": "VODAFONE",
    "avg": -80,
    "min": -118,
    "max": -42
   },
   {
    "key": "WIND",
    "avg": -80,
    "min": -119,
    "max": 0
   }
 ]
}
```


- **/myapp/levelStats/{start} – {end}/{network_type}/**

**Request: GET**

***Request URL***

http://localhost:8000/myapp/levelStats/2012-01-01 – 2018-01-01/2g/

**Server response**

200

**Response body**


```
{
  "count": 3,
  "next": null,
  "previous": null,
  "results": [
    {
      "key": "COSMOTE",
      "avg": -81,
      "min": -109,
      "max": -51
    },
    {
      "key": "VODAFONE",
      "avg": -87,
      "min": -113,
      "max": -51
    },
    {
      "key": "WIND",
      "avg": -83,
      "min": -113,
      "max": -51
    }
  ]
}
```


- **/myapp/uplinkStats/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/uplinkStats/

***Server response***

200

**Response body**


```
{
  "count": 4,
  "next": null,
  "previous": null,
  "results": [
    {
      "key": "COSMOTE",
      "avg": 77,
      "min": 1,
      "max": 4407
    },
    {
      "key": "OTHER",
      "avg": 7,
      "min": 0,
      "max": 107
    },
    {
      "key": "VODAFONE",
      "avg": 41,
      "min": 1,
      "max": 3023
    },
    {
      "key": "WIND",
      "avg": 87,
      "min": 0,
      "max": 4509
    }
  ]
}
```


- **/myapp/uplinkStats/{network_type}/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/uplinkStats/2g/

**Server response**

200

**Response body**


```
{
  "count": 3,
  "next": null,
  "previous": null,
  "results": [
    {
      "key": "COSMOTE",
      "avg": 14,
      "min": 1,
      "max": 290
    },
    {
      "key": "VODAFONE",
      "avg": 10,
      "min": 1,
      "max": 230
    },
    {
      "key": "WIND",
      "avg": 19,
      "min": 1,
      "max": 221
    }
  ]
}
```

**Request: GET**

**Request URL**

http://localhost:8000/myapp/uplinkStats/3g/

**Server response**

200

**Response body**


```
{
  "count": 4,
  "next": null,
  "previous": null,
  "results": [
    {
      "key": "COSMOTE",
      "avg": 80,
      "min": 1,
      "max": 4407
    },
    {
      "key": "OTHER",
      "avg": 13,
      "min": 1,
      "max": 107
    },
    {
      "key": "VODAFONE",
      "avg": 51,
      "min": 1,
      "max": 3023
    },
    {
      "key": "WIND",
```

```
    "avg": 100,
    "min": 0,
    "max": 4509
   }
 ]
}
```


**Request: GET**

**Request URL**

http://localhost:8000/myapp/uplinkStats/4g/

**Server response**

200

**Response body**


```
{
 "count": 3,
 "next": null,
 "previous": null,
 "results": [
  {
   "key": "COSMOTE",
   "avg": null,
   "min": null,
   "max": null
  },
  {
   "key": "OTHER",
   "avg": null,
   "min": null,
   "max": null
  },
  {
   "key": "WIND",
```

```
    "avg": null,
    "min": null,
    "max": null
   }
 ]
}
```


- **/myapp/uplinkStats/{start} – {end}/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/uplinkStats/2012-01-01 - 2018-01-01/

***Server response***

200

**Response body**


```
{
 "count": 4,
 "next": null,
 "previous": null,
 "results": [
  {
   "key": "COSMOTE",
   "avg": 77,
   "min": 1,
   "max": 4407
  },
  {
   "key": "OTHER",
   "avg": 7,
   "min": 0,
   "max": 107
  },
```

```
  {
    "key": "VODAFONE",
    "avg": 41,
    "min": 1,
    "max": 3023
  },
  {
    "key": "WIND",
    "avg": 87,
    "min": 0,
    "max": 4509
  }
 ]
}
```


- **/myapp/uplinkStats/{start} – {end}/{network_type}/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/uplinkStats/2012-01-01 - 2018-01-01/2g/

**Server response**

200

**Response body**


```
{
 "count": 3,
 "next": null,
 "previous": null,
 "results": [
  {
    "key": "COSMOTE",
    "avg": 14,
    "min": 1,
```

```
    "max": 290
   },
   {
    "key": "VODAFONE",
    "avg": 10,
    "min": 1,
    "max": 230
   },
   {
    "key": "WIND",
    "avg": 19,
    "min": 1,
    "max": 221
   }
 ]
}
```


- **/myapp/vendorStats/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/vendorStats/

**Server response**

200

**Response body**


```
{
 "count": 4,
 "next": null,
 "previous": null,
 "results": [
  {
   "key": "LGE",
```

```
    "value": "490159"
   },
   {
    "key": "samsung",
    "value": "189155"
   },
   {
    "key": "Sony",
    "value": "51960"
   },
   {
    "key": null,
    "value": "0"
   }
 ]
}
```


- **/myapp/osStats/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/osStats/

**Server response**

200

**Response body**


```
{
 "count": 6,
 "next": "http://localhost:8000/myapp/osStats/?limit=5&offset=5",
 "previous": null,
 "results": [
  {
   "key": "11.2.A.0.31",
```

```
    "value": "51960"
   },
   {
    "key": "GINGERBREAD",
    "value": "186745"
   },
   {
    "key": "JSS15J",
    "value": "2409"
   },
   {
    "key": "JZO54K",
    "value": "490159"
   },
   {
    "key": "KOT49H",
    "value": "1"
   }
 ]
}
```


- **/myapp/networks/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/networks/

**Server response**

200

**Response body**


```
{
  "count": 4,
  "next": null,
```

```
 "previous": null,
 "results": [
  {
   "key": "-",
   "value": 2387
  },
  {
   "key": "2G",
   "value": 82507
  },
  {
   "key": "3G",
   "value": 1102777
  },
  {
   "key": "4G",
   "value": 3641
  }
 ]
}
```


- **/myapp/measurements/**

**Request: GET**

**Request URL**

http://localhost:8000/myapp/measurements/

**Server response**

200

**Response body**


```
{
 "count": 50,
 "next": "http://localhost:8000/myapp/measurements/?limit=5&offset=5",
```

```
"previous": null,
"results": [
 {
  "id": 293560,
  "geomcol1": "AQEAACDmEAAAtyizQSbhM0A6OxkcJbtDQA==",
  "timestamp": "2014-08-08T18:23:15Z",
  "lon": "19.87949",
  "lat": "39.46207",
  "level": null,
  "speed": 32,
  "operatorname": "GR_COSMOTE",
  "mcc": "202",
  "mnc": "01",
  "node": "",
  "cellid": "_",
  "lac": "_",
  "network_type": "_",
  "qual": 0,
  "snr": 0,
  "cqi": 0,
  "lterssi": 0,
  "appversioncode": 104,
  "psc": 0,
  "dl_bitrate": null,
  "ul_bitrate": null,
  "nlac1": 0,
  "ncellid1": 0,
  "nrxlev1": 0,
  "nlac2": 0,
  "ncellid2": 0,
  "nrxlev2": 0,
  "nlac3": 0,
```

```
"ncellid3": 0,
"nrxlev3": 0,
"nlac4": 0,
"ncellid4": 0,
"nrxlev4": 0,
"nlac5": 0,
"ncellid5": 0,
"nrxlev5": 0,
"nlac6": 0,
"ncellid6": 0,
"nrxlev6": 0,
"ctime": "2014-08-08T18:27:18.247000Z",
"event": "",
"accuracy": 9,
"locationsource": "G",
"altitude": 39,
"conntype": "M",
"conninfo": "10",
"avgping": null,
"minping": null,
"maxping": null,
"stdevping": null,
"pingloss": null,
"testdlbitrate": null,
"testulbitrate": null,
"geomcol2": {
  "type": "Point",
  "coordinates": [
    19.87949,
    39.46207
  ]
},
```

```
    "devicemanufacturer": "LGE",
    "devicemodel": "LG-E430",
    "devicename": "VEE3E",
    "versionname": "JZO54K",
    "brand": "lge",
    "androidversion": "0.0",
    "servingcelltime": 16,
    "os": null,
    "os_id": null,
    "ssid": null,
    "camp": null
  }
```


- **/users/**

**Request: GET, POST**

**Request URL**

http://test.hua.gr/users/

**Server response**

200

**Response body**


```
{
 "count": 2,
 "next": null,
 "previous": null,
 "results": [
  {
   "firstname": "test",
   "lastname": "testidis",
   "username": "test",
   "email": "test@hua.gr"
  },
  {
```

```
    "firstname": "Vassilis",
    "lastname": "Dalakas",
    "username": "vdalakas",
    "email": "vdalakas@gmail.com"
   }
 ]
}
```


- **/users/{id}/**

**Request: GET, PUT, PATCH, DELETE**

**Request URL**

http://test.hua.gr/users/1/

**Server response**

200

***Response body***


```
{
 "firstname": "Vassilis",
 "lastname": "Dalakas",
 "username": "vdalakas",
 "email": "vdalakas@gmail.com"
}
```


## Παράρτημα Γ : Κώδικας Εφαρμογής

Στον ακόλουθο σύνδεσμο βρίσκεται ο κώδικας ολόκληρης της εφαρμογής:

https://github.com/mKorniotakis/V2ofDjango