ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ : Ψηφιακής Τεχνολογίας
ΤΜΗΜΑ : Τμήμα Πληροφορικής και Τηλεματικής
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ : Πληροφορική και Τηλεματική
ΚΑΤΕΥΘΥΝΣΗ : Τεχνολογίες και εφαρμογές ιστού

Τίτλος Εργασίας

Ανάπτυξη Διαδικτυακής Πλατφόρμας με σκοπό την παροχή Επιχειρηματικής Ευφυΐας από Μετρήσεις Γεωχωρικών Δεδομένων με χρήση τεχνολογιών Angular2

Όνομα φοιτητή

Λοντόρφος Νικόλαος,
Α.Μ. 15402

Αθήνα, 2017

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ : Ψηφιακής Τεχνολογίας
ΤΜΗΜΑ : Τμήμα Πληροφορικής και Τηλεματικής
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ : Πληροφορική και Τηλεματική
ΚΑΤΕΥΘΥΝΣΗ : Τεχνολογίες και εφαρμογές ιστού

Τριμελής Εξεταστική Επιτροπή

Δαλάκας Βασίλειος

Δημητρακόπουλος Γεώργιος

Τσαδήμας Ανάργυρος

Ο Λοντόρφος Νικόλαος

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει 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 τη 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.

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια του Μεταπτυχιακού Προγράμματος Σπουδών " Τεχνολογίες και εφαρμογές ιστού", του Τμήματος Πληροφορικής και Τηλεματικής του Χαροκοπείου Πανεπιστημίου υπό την επίβλεψη του Διδάσκοντα στο ΠΜΣ και Ε.ΔΙ.Π. του Τμήματος Δρ. Βασίλειου Δαλάκα. Θα ήθελα πρώτα απ' όλα λοιπόν να ευχαριστήσω αυτόν για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον αντικείμενο, το οποίο διεύρυνε τους επιστημονικούς μου ορίζοντες και για καθοδήγηση του καθ΄ όλη την διάρκεια.

Μαζί με τον Δρ. Δαλάκα, θα ήθελα να ευχαριστήσω τον Υποψήφιο Διδάκτορα, μέλος της τριμελούς επιτροπής και Ε.ΤΕ.Π. του Τμήματος κ. Ανάργυρο Τσαδήμα. Χωρίς τις τεχνικές και ακαδημαϊκές γνώσεις και συμβουλές που μου προσφέρανε, την στενή συνεργασία σε όλη την διάρκεια αλλά και την άπλετη υποστήριξη που μου δείξανε, δεν θα γινότανε ποτέ δυνατή η πραγματοποίηση της διπλωματικής αυτής εργασίας.

Θα ήθελα να ευχαριστήσω επίσης τον Επίκουρο Καθηγητή Δρ. Γεώργιο Δημητρακόπουλο, τρίτο μέλος της επιτροπής που συνέβαλε στην πραγματοποίηση αυτής της διπλωματικής.

Ευχαριστώ τους συμφοιτητές μου για τις πολύτιμες συμβουλές, τις γνώσεις, την στήριξη αλλά και την παρέα που μου παρείχαν σε όλη την διάρκεια των σπουδών στο Χαροκόπειο Πανεπιστήμιο.

Νοιώθω τέλος ότι πρέπει να εκφράσω τις ιδιαίτερες μου ευχαριστίες σε τέσσερα άτομα χωρίς τα οποία δύσκολα θα βρισκόμουν σε θέση να τελειώσω το Μεταπτυχιακό αυτό: τον φίλο μου Βασίλη που με τις τεχνικές του γνώσεις και την υπομονή του βοήθησε στο να σχεδιαστεί και να ξεκινήσει το project αυτό αλλά και για όλη την βοήθεια του κατά την διάρκεια του μεταπτυχιακού, την κοπέλα μου Σήλια, για την στήριξη και την υπομονή της τα 2 χρόνια του μεταπτυχιακού και την

βοήθεια της στο να μην χάσω το στόχο παρά τις δυσκολίες και φυσικά, ευχαριστώ τους γονείς μου για την διαχρονική συμπαράσταση τους και την υλική και ηθική υποστήριξη τους.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη

Στο παρών κείμενο παρουσιάζεται ο σχεδιασμός και η υλοποίηση της διπλωματικής εργασίας με αντικείμενο την ανάπτυξη μιας Διαδικτυακής Πλατφόρμας και σκοπό την παροχή Επιχειρηματικής Ευφυΐας (Business Intelligence) σε μετρήσεις γεωχωρικών δεδομένων . Ο στόχος είναι ο καλύτερη κατανόηση της σημασίας των δεδομένων αυτών, δίνοντας μια οπτικοποίηση τους. Με τον τρόπο αυτό πρότυπα, τάσεις και συσχετισμοί που μπορεί να περάσουν απαρατήρητα με κάποια απεικόνιση βασισμένη σε κείμενο, μπορούν εύκολα να αναγνωριστούν με τη σωστά οπτικοποίηση.

Τα γεωχωρικά δεδομένα που χρησιμοποιήσαμε παραχωρήθηκαν από τον Δρ. Βασίλειο Δαλάκα και αφορούν μετρήσεις απόδοσης ασυρμάτων δικτύων. Βρίσκονται αποθηκευμένα σε μια βάση δεδομένων από την οποία τα αντλεί η εφαρμογή μας για να εξάγει συμπεράσματα και να πραγματοποιήσει visualizations.

Η βάση που χρησιμοποιήσαμε ήταν τύπου PostgreSQL, object-relational δηλαδή, με εγκατεστημένη την επέκταση PostGIS. Η PostGIS δίνει τη δυνατότητα υποστήριξης γεωγραφικών objects, μας δίνει δηλαδή την δυνατότητα για SQL queries με βάση την τοποθεσία. Στα δεδομένα αυτά έχουμε πρόσβαση από τον client με την χρήση ενός Rest Web Service φτιαγμένο με το Java Spring Framework. Με την Angular 2 παίρνουμε τα δεδομένα αυτά και τα απεικονίζουμε στον χρήστη με την βοήθεια του Chart.js σε ευανάγνωστα και όμορφα γραφήματα. Επίσης για την δημιουργία χαρτών χρησιμοποιούμε ένα Geoserver για την ομαδοποίηση και διαμόρφωση των δεδομένων και το Google Maps για την απεικόνιση τους .

Στα κεφάλαια που ακολουθούν γίνεται μια ανάλυση των τεχνολογιών και των τεχνικών που χρησιμοποιήσαμε για την ανάπτυξη της Διαδικτυακής Πλατφόρμας μας.

Αναλυτικά, στην εισαγωγή αναφέρουμε τα προβλήματα που μας οδήγησαν στην δημιουργία της εφαρμογής μας, καθώς και την πρόταση που προτείνουμε για την επίλυση τους.

Στο κεφάλαιο 1 αναλύουμε τις τεχνολογίες που χρησιμοποιήσαμε για την δημιουργία της πλατφόρμας μας, χωρίζοντας τες σε Client-side και Server-side και λέγοντας λίγα λόγια για αυτές.

Στο κεφάλαιο 2 περιγράφουμε το λειτουργικό περιβάλλον της εφαρμογής μας και τις λειτουργικές τις απαιτήσεις. Αναλύουμε δηλαδή όλα τα εργαλεία και εφαρμογές που χρησιμοποιήσαμε στο σύστημα μας για την ανάπτυξη και λειτουργιά της εφαρμογής.

Στο κεφάλαιο 3 κάνουμε μια παρουσίαση της εφαρμογής και των οδηγιών χρήσης της. Γίνεται ακόμα μια ανάλυση της αρχιτεκτονικής της εφαρμογής και μια τεχνική ανάλυση επάνω σε θέματα όπως το σχήμα τις βάσης μας, πως στήθηκε το Rest Framework στο Java Spring, πως στήθηκε η Angular 2 στο client-side κ.α. Το κεφάλαιο αυτό ουσιαστικά αποτελεί ένα είδος

manual για το πως θα μπορέσει κάποιος να εγκαταστήσει για να χρησιμοποιήσει την εφαρμογή μας.

Τέλος στο κεφάλαιο 4 παρατίθενται τα συμπεράσματα που βγάλαμε από την εκπόνηση της διπλωματικής εργασίας, τα προβλήματα που αντιμετωπίσαμε και θέματα για μελλοντική έρευνα.

Abstract

In this document we present the planning and execution of our thesis project, with its subject being the development of a web-based platform with the goal of extracting Business Intelligence from geospatial data measurements. What we are trying to achieve is a better understanding of the meaning of this data by visualizing them. That way, we can identify patterns, trends and connections that can go unnoticed with traditional text-based presentations.

The geospatial data we used were given by Dr. Vasileios Dalakas and they refer to measurements of quality of service of mobile network providers. They are stored in a database our application has access to, in order to draw conclusions and perform visualizations.

This database type is PostgreSQL, which is object-relational and has the PostGIS extension installed. PostGIS is an open source software that adds support for geographic objects, enabling us to use location-based SQL queries. Our client has access to the data through a RESTful Web Service created with Java Spring Framework. Angular 2 "consumes" this data and with the help of Chart.js library, we present the user with beautiful and easily readable charts. For interactive map creation, we used Geoserver to process and serve the data and the Google Maps library to present them.

In the following chapters we present an in-depth analysis of the technologies and techniques used for the creation of our web-based platform.

Specifically, in the introduction we give a mention to the problems that lead us to the creation of the application and the solutions we propose to solve them.

In chapter 1 we present an analysis of the technologies we used for the creation of our platform, separating them to Client-side and Front-side and giving a short description of each one.

In chapter 2 we describe the operating environment of our application and its functional requirements. We analyze all the tools and frameworks we used in order to develop it, deploy it and have it running.

In chapter 3 we present our application and its instructions. We also present an outline of our platform architecture and a technical analysis on subjects like database schema, how we set up our REST framework in Spring, how we set up Angular 2 in client side etc. This chapter serves the purpose of a manual for setting up our application and using it.

Finally in Chapter 4 we present our conclusions and findings from the process of the creation of our web-based platform, the problems we faced and the needs for future research.

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

1 Εισαγωγή

Με την έννοια του Data Visualization αναφερόμαστε στην οπτική αναπαράσταση στατιστικών στοιχείων και άλλων αριθμητικών και μη αριθμητικών δεδομένων με την χρήση στατικών ή διαδραστικών εικόνων και γραφικών (Gatto, 2015). Ο στόχος του είναι να εξάγουμε σχέσεις, κενά, σχήματα και συνδέσεις τα οποία δεν είναι εύκολο να παρατηρήσουμε διαβάζοντας τα raw δεδομένα ή τεράστια κείμενα. Το Data Visualization βοηθά στην κατανόηση των δεδομένων από έμπειρους ερευνητές και επαγγελματίες αλλά και από ένα πολύ πιο ευρύ κοινό χωρίς ειδικές γνώσεις. Τελικά, μειώνει τα κενά γνώσης πάνω στην πληροφορία - ειδικά τα κενά σε ποσοτικά μετρήσιμες ιδιότητες. Τα λεκτικά και τα αριθμητικά δεδομένα σήμερα παράγονται με μεγαλύτερους ρυθμούς από ποτέ και με διάφορους τρόπους: υπολογίζεται ότι το 90% των υπαρχόντων δεδομένων σήμερα έχει παραχθεί από το 2010(Science Daily, 2015). Μέχρι το 2003 είχαν παραχθεί 5 Exabytes συνολικά, ενώ το 2013 ο ίδιος αριθμός δεδομένων παραγόταν καθημερινά (Gunelius, 2014). Αυτή η 'έκρηξη' στο όγκο των δεδομένων μπορεί να γίνει εύκολα κατανοητή αναλύοντας το internet και τα social media. Το 2012 η Google λάμβανε 2 εκατομμύρια αναζητήσεις το λεπτό. Το 2014 ο αριθμός αυτός διπλασιάστηκε. Το 2014, κάθε λεπτό 2.5 εκατομμύρια κομμάτια content γίνονταν share στο Facebook, γίνονταν 277.000 tweets, 216.000 φωτογραφίες ανέβαιναν στο Instagram, 72 ώρες βίντεο ανέβαιναν στο YouTube, 48.000 apps κατέβαιναν από το Apple Store και στέλνονταν πάνω από 204 εκατομμύρια e-mails (Gunelius, 2014). Όσο σαν κοινωνία ερχόμαστε καθημερινά σε επαφή με δεδομένα και η εξάρτηση μας από αυτά όλο και μεγαλώνει, η κατανόηση του νοήματος και των αριθμών χάνεται μέσα σε γραμμές και στήλες. Το Data Visualization επιτρέπει σε αυτόν που αναζητά την πληροφορία να μπορεί να την καταναλώσει σε μεγάλες ποσότητες, οργανώνοντας τα δεδομένα με έναν τρόπο ο οποίος αναδεικνύει τις σχέσεις, τα patterns και τα κενά. Όσο ο όγκος των δεδομένων αυξάνεται, τα visualizations, πέρα από ένα πρακτικό εργαλείο που βοηθά στην κατανόηση, γίνεται απαραίτητα. Επίσης οι άνθρωποι εξοικειώνονται όλο και περισσότερο με την οπτική απεικόνιση δεδομένων. Ενώ παλαιότερα η παραγωγή και έκθεση σε γραφικές απεικονίσεις περιοριζόταν σε επιστήμονες, ακαδημαϊκούς και στατιστικολόγους, οι περισσότεροι σήμερα ερχόμαστε σε επαφή με τουλάχιστον ένα data visualization την ημέρα: charts χρησιμοποιούνται συνέχεια στην τηλεόραση (πχ στο δελτίο καιρού ή σε προγνωστικά εκλογών) , στον γραπτό και τον ηλεκτρονικό Τύπο. Ακόμα και τα παιδιά έρχονται σε επαφή με visualizations καθημερινά. Με αυτό τον τρόπο η εξοικείωση με τέτοιες αναπαραστάσεις δεδομένων διαρκώς αυξάνεται.

Τελικά, αυτό που προσπαθούμε να επιτύχουμε με τα Data Visualizations είναι η απλοποίηση της παρουσίασης κάποιας υπόθεσης, θεωρίας ή ιστορίας, πράγμα πολύ δύσκολο με raw δεδομένα. Αυτό σημαίνει ότι η οπτικοποίηση επιτρέπει με απλό αλλά δυναμικό τρόπο την επεξήγηση και το μοίρασμα δεδομένων, ενώ ταυτόχρονα λειτουργεί και σαν εργαλείο για την ανάπτυξη της έρευνας.

1.1 Ορισμός του προβλήματος

Είχαμε μια σειρά από μετρήσεις οι οποίες αφορούν την απόδοση ασυρμάτων δικτύων φορέων όπως Cosmote, Wind, Vodafone κ.α.

Στόχος μας ήταν η κατασκευή μιας διαδικτυακής πλατφόρμας η οποία θα είναι σε θέση να προβάλει με γραφικό τρόπο στατιστικά και visualizations επάνω στα δεδομένα αυτά.

Το πρόβλημα που προέκυψε, ήταν κυρίως ο όγκος των δεδομένων, τα οποία ξεπερνούσαν τις 600.000 και η διαχείριση ενός τόσο μεγάλου data set.

Αυτή η αφθονία δεδομένων απαίτησε περισσότερους και πιο προσβάσιμους τρόπους επεξεργασίας της πληροφορίας. Την λύση μας δώσανε τα Data Visualizations. Με την χρήση τους, στοχεύσαμε στην βελτίωση της κατανόησης της γνώσης αυτής και τον εμπλουτισμό της. Επιπροσθέτως, με τις γραφικές αναπαραστάσεις προσπαθήσαμε να συνοψίσουμε το μεγάλο πλήθος των δεδομένων σε ευκολότερα κατανοητές μορφές, επιτρέποντας την αναγνώριση patterns, τις συγκρίσεις και την εύρεση διαφορών πιο εύκολα και γρήγορα.

Η πρόταση της διπλωματικής

Στην παρούσα διπλωματική, η πρόταση που προτείνεται είναι μια διαδικτυακή πλατφόρμα παρουσίασης Data Visualizations πάνω σε μεγάλα datasets. Στον χρήστη παρουσιάζεται μέσω του browser του μια πληθώρα Visualizations, όπως bar-charts, pie-charts αλλά και διαδραστικούς χάρτες, τα οποία βοηθούν στο να κατανοήσει ευκολότερα τα αποτελέσματα των μετρήσεων μας και να εξάγει πιο ολοκληρωμένα συμπεράσματα. Η αρχιτεκτονική της εφαρμογής μας (η οποία παρουσιάζεται αναλυτικά παρακάτω) είναι δυναμική, παρουσιάζει δηλαδή άμεσα στον χρήστη ότι αλλαγή πραγματοποιηθεί στο dataset και είναι φτιαγμένη έτσι ώστε να υποστηρίζει όσο μεγάλο αριθμό καταχωρήσεων και αν έχουμε.

Το τελικό αποτέλεσμα μπορεί να το δει κάποιος στο URI: http://test.hua.gr/datavisual/

2 Αρχιτεκτονική του συστήματος

Η δημιουργία μιας διαδικτυακής πλατφόρμας σαν την δικιά μας απαίτησε ένα συνδυασμό πολλών τεχνολογιών ανάπτυξης διαδικτύου. Οι τεχνολογίες αυτές διακρίνονται σε δύο βασικές κατηγορίες: **Client Side** και **Server Side**.

Το client side development γίνεται σχεδόν αποκλειστικά σε JavaScript (Javascript), χρησιμοποιώντας ταυτόχρονα κώδικα σε HTML (HTML) και CSS(CSS). Ο λόγος που αποκαλείται client side είναι επειδή τα scripts εκτελούνται στον υπολογιστή του τελικού χρήστη όταν φορτωθεί μια ιστοσελίδα.

Στο server side development , τα scripts μας τρέχουν στον Web Server ο οποίος φιλοξενεί την ιστοσελίδα και δημιουργούν δυναμικά HTML η οποία στέλνεται στον browser του client. Υπάρχει πληθώρα server side γλωσσών, όπως η Java (Java), η PHP (PHP) και η Ruby on Rails (Ruby on Rails).

Στην δική μας πλατφόρμα, το stack που επιλέξαμε είναι το Spring MVC (Spring) στο Server Side και η Angular 2 (Angular2) στο Client side.

Ο λόγος που διαλέξαμε αυτό το stack είναι ότι το Spring μαζί με την Angular αποτελούν ένα πολύ θελκτικό συνδυασμό για την κατασκευή data-intensive web applications.

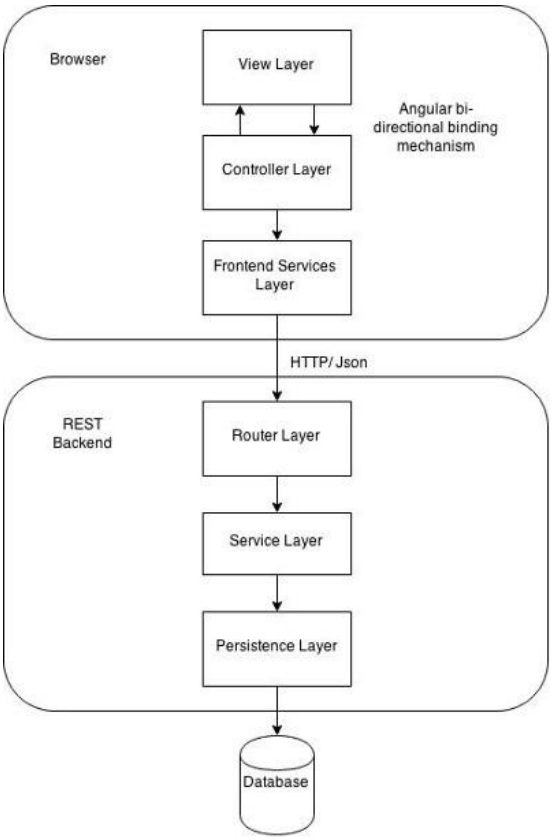
Η κύρια ιδέα σε σχέση με άλλες server side αρχιτεκτονικές είναι ότι χτίσαμε τον server σαν ένα σύνολο από stateless, επαναχρησιμοποιήσιμα REST services(REST) και από την οπτική της MVC λογικής (MVC), βγάλαμε την λογική του controller από το backend και την μεταφέραμε στον browser. Ο client είναι MVC-Capable και περιλαμβάνει όλη την λογική απεικόνισης της πληροφορίας, την οποία χωρίζει σε ένα view layer, ένα controller layer και ένα Services layer.

Ο κώδικας της εφαρμογής βρίσκεται στα repositories:

Angular: https://github.com/NickLont/Data-visualization-project

Spring: https://bitbucket.org/vdalakas/huatester

Στην εφαρμογή μπορεί κάποιος να έχει πρόσβαση από το URI: http://test.hua.gr/datavisual/



Εικόνα 1 - Αρχιτεκτονική

Μετά την εκκίνηση την εφαρμογής, ο client με τον server επικοινωνούν μόνο με JSON (JSON) μηνύματα.

Ο server που χρησιμοποιήσαμε είναι ο Tomcat(Apache Tomcat) και η βάση δεδομένων είναι η PostgreSQL(PostgreSQL) όπως προαναφέραμε.

Παρακάτω αναλύονται λεπτομερέστερα οι τεχνολογίες αυτές.

2.1 Client-Side τεχνολογίες

HTML 5

Η HTML (HTML) είναι μια markup γλώσσα, η οποία χρησιμοποιείται για την δόμηση και την παρουσίαση περιεχομένου στο Διαδίκτυο. Μαζί με την CSS και την JavaScript, αποτελούν τους ακρογωνιαίους λίθους του.

Ο σκοπός της είναι η περιγραφή της δομής των σελίδων του Web με τρόπο σημασιολογικό. Το βασικό της στοιχείο είναι τα Elements τα οποία μας παρέχουν τα μέσα για να

δημιουργήσουμε δομημένα έγγραφα. Ο τρόπος με τον οποίο το κάνει αυτό είναι δηλώνοντας την δομή αυτή με σημασιολογικό τρόπο, πχ εάν πρόκειται για κεφαλίδα, παράγραφο ή σύνδεσμο. Τα HTML elements δηλώνονται με tags. Κάποια tags όπως τα <img/> και <input/> εισάγουν κατευθείαν content στην σελίδα ενώ κάποια άλλα όπως το ... ενθυλακώνουν και παρέχουν πληροφορίες για το κείμενο που περικλείουν. Χρησιμοποιώντας HTML μπορούμε επίσης να ενσωματώνουμε στην σελίδα μας διαφόρων τύπων αντικείμενα όπως εικόνες ή διαδραστικές φόρμες αφού γίνει render. Η τρέχουσα έκδοση είναι η 5 η οποία παρουσιάστηκε τον Οκτώβριο του 2014 από τον οργανισμό World Wide Web Consortium (W3C) με σκοπό την καλύτερη υποστήριξη multimedia (βίντεο, audio κ.α.), διατηρώντας ταυτόχρονα την ευκολία ανάγνωσης της από ανθρώπους και της κατανόησης από υπολογιστές (web browsers, parsers κτλ). Παρουσίασε νέα elements και attributes για λειτουργίες οι οποίες είναι συναντώνται συχνά στις μοντέρνες ιστοσελίδες. Κάποια από αυτά δίνουν σημασιολογική έννοια σε block γενικής σημασίας (πχ. αντί για τα και elements, δίνει την δυνατότητα χρήσης <nav> ή <footer> blocks). Κάποια παλαιότερα attributes που αφορούσαν την παρουσίαση elements όπως <font> και καταργήθηκαν, αφού ξεπεράστηκαν από την ισχυρότερη σε τέτοια θέματα CSS.

CSS 3

Η CSS (Cascading Style Sheets) (CSS) είναι μια style sheet γλώσσα που χρησιμοποιείται για την περιγραφή της παρουσίασης περιεχομένου γραμμένου σε markup γλώσσες (όπως η HTML). Σχεδιάστηκε κυρίως για να διαχωρίσει την παρουσίαση και το περιεχόμενο και περιλαμβάνει επιλογές για το layout, τα fonts και τα χρώματα της σελίδας. Αυτός ο διαχωρισμός δίνει την δυνατότητα εμφάνισης της ίδιας σελίδας με διαφορετικά στυλ στο rendering, πχ. άλλη εμφάνιση στην οθόνη και στην εκτύπωση. Γενικά, οι αλλαγές στον γραφιστικό σχεδιασμό μιας σελίδας μπορούν να πραγματοποιηθούν πολύ γρήγορα, αλλάζοντας μόνο μερικές γραμμές στο CSS αρχείο που χρησιμοποιεί.

Η CSS3 είναι το τελευταίο standard και είναι backwards compatible με τις προηγούμενες εκδόσεις. Έφερε κάποιες πολυαναμενόμενες καινοτομίες όπως στρογγυλοποιημένες γωνίες, σκιάσεις, gradients και animations αλλά και καινούργια layouts όπως multi-columns, flexible boxes και grid layouts.

Typescript

Η Typescript (Typescript) είναι μια δωρεάν open-source γλώσσα προγραμματισμού που αναπτύχθηκε από την Microsoft. Είναι ένα υπερσύνολο της JavaScript στην οποία προσθέτει προαιρετικές δυνατότητες όπως static-typing (ο τύπος κάθε μεταβλητής δηλώνεται από τον προγραμματιστή ώστε να είναι γνωστός την ώρα του compile, βοηθώντας στην έγκαιρη αναγνώριση bugs) και class-based object-oriented προγραμματισμού. Χρησιμοποιείται για τον σχεδιασμό JavaScript applications που εκτελούνται στο client-side ή στο server-side. Γίνεται transcompile σε κανονική JavaScript στο runtime από τον Typescript compiler. Μιας και η Typescript αποτελεί υπερ-σύνολο της JavaScript, κάθε valid JavaScript πρόγραμμα είναι valid και στην Typescript.

Η Typescript υποστηρίζει επίσης definition files (ονομάζονται header files) που περιλαμβάνουν πληροφορίες για τον τύπο object που δημιουργήθηκε από κάποια υπάρχουσα JavaScript βιβλιοθήκη. Αυτό οδήγησε στην άμεση δημιουργία header files από τρίτες εταιρίες για την υποστήριξη δημοφιλών βιβλιοθηκών όπως η JQuery ή το D3.js.
Η ανάπτυξη της ξεκίνησε με σκοπό να καλύψει τις ελλείψεις της JavaScript στην δημιουργία applications μεγάλης κλίμακας στην Microsoft και στους συνεργάτες της. Οι developers της Typescript αναζήτησαν μια λύση η οποία δεν θα χαλάσει την συμβατότητα με τα standard και την υποστήριξη ανεξαρτήτως πλατφόρμας. Γνωρίζοντας ότι η ECMAScript είχε υποσχεθεί μελλοντική υποστήριξη για class-based προγραμματισμό, η Typescript βασίστηκε σε αυτή την πρόταση. Αυτό οδήγησε στην δημιουργία ενός JavaScript compiler με ένα σετ συντακτικών γλωσσικών επεκτάσεων, ο οποίος μεταφράζει τις επεκτάσεις αυτές σε απλή JavaScript. Προσέθεσαν επίσης προαιρετικό static-typing το οποίο δίνει την δυνατότητα στατικής ανάλυσης της γλώσσας, προσφέροντας έτσι υποστήριξη σε tools και IDE's.

Angular 2

Η Angular 2(Angular2) (ή σκέτο Angular πλέον) είναι μια open-source, front-end web application πλατφόρμα βασισμένη στην Typescript. Σχεδιάστηκε από την Angular Team της Google σε συνεργασία με μια κοινότητα ιδιωτών και εταιριών με σκοπό την βελτίωση του workflow των developers στην κατασκευή πολύπλοκων web applications. Αποτελεί έναν πλήρη επανασχεδιασμό της AngularJS (AngularJS) από την ομάδα που την δημιούργησε. Προωθεί μία component-based αρχιτεκτονική, χρησιμοποιώντας ταυτόχρονα τα νέα features της Typescript όπως κλάσεις και modules. Σχεδιάστηκε έχοντας πρώτα υπ' όψιν το development για φορητές συσκευές και τα προβλήματα στην απόδοση τους, μιας και είναι ευκολότερη η ανάπτυξη σε desktop περιβάλλον άμα λυθούν αυτά. Δεν χρησιμοποιεί πλέων "scopes" όπως η AngularJS αλλά στηρίζει την αρχιτεκτονική της στην ιεραρχία των components. Είναι φτιαγμένη μόνο για μοντέρνους browsers, οπότε υπάρχει μικρότερη ανάγκη για τον σχεδιασμό λύσεων για συμβατότητα με παλαιότερους browsers. Έχει βελτιωμένο dependency injection, δίνει την

δυνατότητα ασύγχρονης σύνθεσης templates, έχει απλό routing και υποστηρίζει reactive programming με την βιβλιοθήκη RxJS (RxJS).

Χρησιμοποιήσαμε επίσης κάποιες εξωτερικές βιβλιοθήκες:

-Chart.js (Chart.js): Ένα community-maintained project που βοηθάει στην δημιουργία animated, διαδραστικών και responsive γραφημάτων βασισμένων στην HTML5.

-GoogleMaps (angular2-google-maps): Ένα Web-mapping service σχεδιασμένο από την Google. Χρησιμοποιήθηκε για τον σχεδιασμό χαρτών επάνω στα οποία παρουσιάσαμε τα γεωχωρικά δεδομένα μας.

-Webpack(Webpack): Το Webpack είναι ένας Module bundler για Javascript applications. Όταν το ενσωματώνουμε στην εφαρμογή μας, δημιουργεί έναν γράφο με τα dependencies του κάθε module του application μας και έπειτα τα πακετάρει σε ένα bundle το οποίο φορτώνει στον browser. Βοηθάει ώστε το Web-application μας να είναι συντηρήσιμο, αξιόπιστο και αποδοτικό.

2.2 Server-Side τεχνολογίες

Apache Tomcat

Ο Apache Tomcat (Tomcat) είναι ένα open-source Java Servlet Container το οποίο αναπτύχθηκε από τον Apache Software Foundation. Υλοποιεί αρκετά specifications της Java Enterprise Edition, όπως Java Servlet, JavaServer Pages κ.α. και παρέχει ένα καθαρά JAVA HTTP web server περιβάλλον στο οποίο μπορεί να τρέξει κώδικας Java. Η τρέχουσα έκδοση είναι η 4.x η οποία εκδόθηκε περιλαμβάνοντας το Catalina (ένα servlet container), το Coyote (HTTP connector) και το Jasper (ένα JSP engine). Εμείς χρησιμοποιήσαμε το Catalina, το οποίο υλοποιεί τις προδιαγραφές της Sun Microsystems για servlets. Στον Tomcat, ένα Realm αντιπροσωπεύει μια "βάση δεδομένων" από usernames, passwords και ρόλους (όπως του Unix) τα οποία έχουν ανατεθεί σε χρήστες. Διαφορετικές υλοποιήσεις των Realms επιτρέπουν στο Catalina να ενσωματωθεί σε περιβάλλοντα στα οποία τέτοιες πληροφορίες που αφορούν authentication έχουν ήδη δημιουργηθεί και συντηρούνται και να χρησιμοποιήσει τις πληροφορίες αυτές για να υλοποιήσει το Container Managed Security, όπως αυτό περιγράφεται στα Servlet Specifications.

PostgreSQL

Η PostgreSQL (PostgreSQL), η οποία συχνά αναφέρεται και σαν Postgres, είναι μια object-relational database, μια relational βάση δηλαδή με δυνατότητα χρησιμοποίησης "objects"(RDBMS). Όταν χρησιμοποιείται σαν database server, οι βασικές της δυνατότητες είναι να αποθηκεύει με ασφάλεια δεδομένα και να τα επιστρέφει σαν απάντηση σε requests από άλλα software applications. Είναι ACID-compliant (ACID) και transactional και υποστηρίζει τους περισσότερους τύπους δεδομένων όπως INTEGER, NUMERIC, CHAR κτλ. Υποστηρίζει επίσης την αποθήκευση μεγάλων δυαδικών αντικειμένων (binary) όπως εικόνες, ήχος και βίντεο. Τρέχει σε όλα τα βασικά λειτουργικά, όπως UNIX, Linux και Windows. Αναπτύσσεται από το PostgreSQL Global Development Group και είναι δωρεάν και open source. Προέρχεται από το πακέτο POSTGRES, το οποίο φτιάχτηκε αρχικά στο Πανεπιστήμιο Berkeley στην Καλιφόρνια. Η ανάπτυξη της συνεχίζεται για δύο δεκαετίες και αποτελεί την πιο προχωρημένη βάση δεδομένων ανοιχτού κώδικα.

Στο application μας χρησιμοποιήσαμε το open source πρόγραμμα PostGIS (Postgis), μία επέκταση για γεωχωρικές βάσεις δεδομένων που προσθέτει υποστήριξη για γεωγραφικά objects, δίνοντας έτσι την δυνατότητα για SQL queries που βασίζονται στην τοποθεσία.

Java Spring Framework

Το Spring Framework (Spring) είναι η πιο δημοφιλής open-source Java πλατφόρμα αυτή την στιγμή. Χρησιμοποιείται από εκατομμύρια προγραμματιστές στον κόσμο για την παραγωγή κώδικα υψηλής απόδοσης, εύκολα ελέγξιμο και επαναχρησιμοποιήσιμο. Τα βασικά του features μπορούν να χρησιμοποιηθούν στο development οποιουδήποτε Java application αλλά υπάρχουν επεκτάσεις για την δημιουργία web applications επάνω από την πλατφόρμα της Java EE. Έχει σκοπό να κάνει το J2EE development ευκολότερο στη χρήση και προωθεί καλές πρακτικές εφαρμόζοντας ένα προγραμματιστικό μοντέλο βασισμένο στα POJO's. Αυτό δίνει την δυνατότητα χρήσης ενός απλού Servlet container όπως ο Tomcat αντί για εξεζητημένους application servers που να διαχειρίζονται EJB's. Είναι στηριγμένο στην λογική Model-View-Controller.

Geoserver

Ο Geoserver (Geoserver) είναι ένας open-source Server γραμμένος σε Java ο οποίος επιτρέπει στους χρήστες να διαμοιράζουν, να διαχειρίζονται και να επεξεργάζονται γεωχωρικά δεδομένα. Σχεδιάστηκε με βάση την διαλειτουργικότητα και μπορεί να κάνει publish δεδομένα από τις μεγαλύτερες πηγές γεωχωρικών δεδομένων χρησιμοποιώντας ανοιχτά standards. Αποτελεί μια εύκολη μέθοδο σύνδεσης υπάρχουσας πληροφορίας με virtual globes όπως το Google Earth αλλά και web-based maps όπως το OpenLayers και το Google Maps. Υλοποιεί τα

στάνταρ Web Feature Service, Web Map Service, Web Coverage Service and Web Processing Service σύμφωνα με τις προδιαγραφές του Open Geospatial Consortium(Open Geospatial Consortium).

3 Εγκατάσταση και παραμετροποίηση απαιτούμενου λογισμικού

Στο κεφάλαιο αυτό παρουσιάζουμε το περιβάλλον στο οποίο έγινε η ανάπτυξη της εφαρμογής και παραθέτουμε όλα τα προγράμματα και τα εργαλεία που χρησιμοποιήσαμε ώστε να καταστεί δυνατή η λειτουργία της. Αποτελεί έναν οδηγό για το στήσιμο του περιβάλλοντος αυτού ώστε να λειτουργεί σωστά η εφαρμογή μας και να μπορούμε να πραγματοποιήσουμε τις επιθυμητές λειτουργίες.

Tomcat

Ο Apache Tomcat (Tomcat), ο οποίος συχνά αναφέρεται ως Tomcat Server, είναι ένα open source Java Servlet Container όπως είδαμε παραπάνω. Τον χρησιμοποιήσαμε για να κάνουμε deploy τον Geoserver και το WAR του Java Spring κώδικα που γράψαμε ώστε να έχουμε πρόσβαση στο web application μας μέσω browser. Οι οδηγίες που δίνονται παρακάτω αφορούν το λειτουργικό περιβάλλον των Linux Mint αλλά στο manual του Tomcat (Tomcat Manual) δίνονται οδηγίες εγκατάστασης για οποιοδήποτε λειτουργικό. Να σημειωθεί ότι η τοπική εγκατάσταση του Server μας χρειάστηκε κατά της διάρκεια της ανάπτυξης της πλατφόρμας μας. Μετά την ολοκλήρωση της έγινε deploy στους servers του Χαροκοπείου στο URI test.hua.gr:8080, όπου μπορεί κάποιος να έχει πρόσβαση στον Geoserver και στο API χωρίς να χρειαστεί τοπική εγκατάσταση.

Οδηγίες εγκατάστασης:

1. Επιβεβαίωση ότι έχουμε εγκατεστημένη την σωστή έκδοση της Java στο λειτουργικό μας πληκτρολογώντας
   $java-version

```
nick@nick-VirtualBox ~ $ java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
```

Εικόνα 2 - Java version

Ο Tomcat 9 είναι σχεδιασμένος να τρέχει σε Java SE 8 και μεταγενέστερες εκδόσεις. Εάν δεν την έχουμε εγκατεστημένη ή έχουμε μικρότερη έκδοση θα πρέπει να εγκαταστήσουμε την Java 8.

```
$ sudo add-apt-repository ppa:webupd8team/java

$ sudo apt-get update

$ sudo apt-get install oracle-java8-installer
```

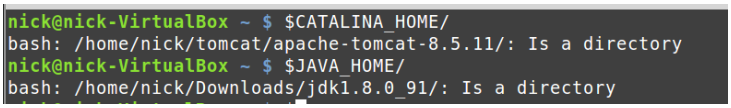
2. Κατεβάζουμε και αποσυμπιέζουμε το αρχείο εγκατάστασης :

```
$  cd /opt# wget http://www.us.apache.org/dist/tomcat/tomcat-9/v9.0.0.M1/bin/apache-tomcat-9.0.0.M1.tar.gz

$  tar xzf apache-tomcat-9.0.0.M1.tar.gz

$  mv apache-tomcat-9.0.0.M1 tomcat9
```

3. Ρυθμίζουμε το environmental variable CATALINA_HOME ώστε να βλέπει τον Tomcat και την Java:

4.
```
$ echo "export CATALINA_HOME="/opt/tomcat9"" >> /etc/environment
$ echo "export JAVA_HOME="/usr/lib/jvm/java-8-oracle"" >> /etc/environment

$ echo "export JRE_HOME="/usr/lib/jvm/java-8-oracle/jre"" >> /etc/environment

$source ~/.bashrc
```

Επιβεβαιώνουμε ότι εγκαταστάθηκαν:



Εικόνα 3 - Catalina and Java home

5. Πληκτρολογώντας

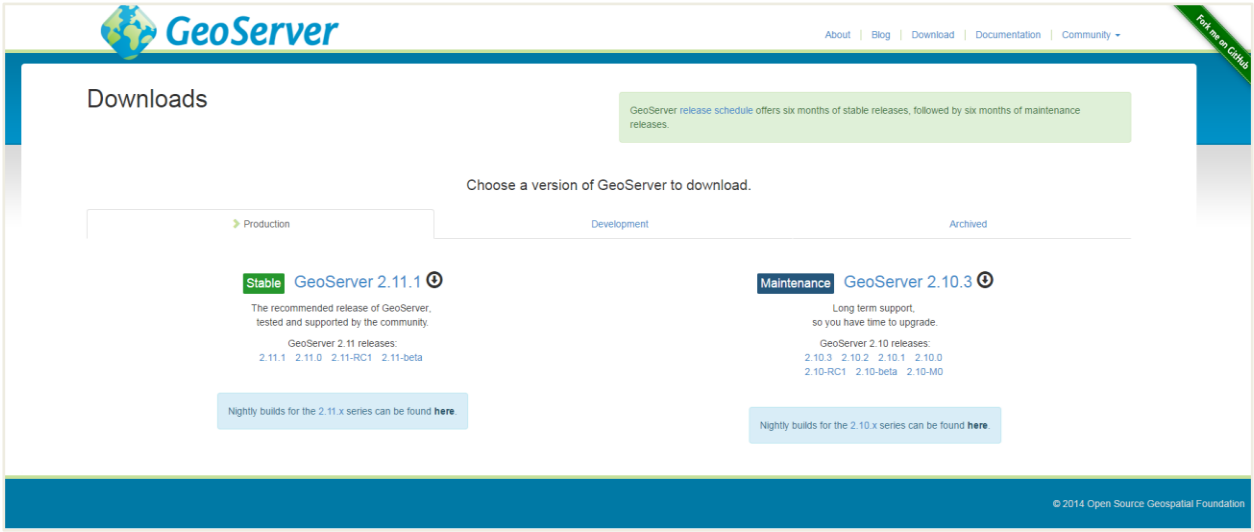
```
$ cd /opt/tomcat9

$ ./bin/startup.sh
```

Μπορούμε να ξεκινήσουμε τον Tomcat στο port 8080. Δεν θα το κάνουμε όμως γιατί θα διαχειριζόμαστε τα deployments μέσα από το IDE μας, το IDEA(Intellij Idea) όπως θα δούμε παρακάτω.

Geoserver

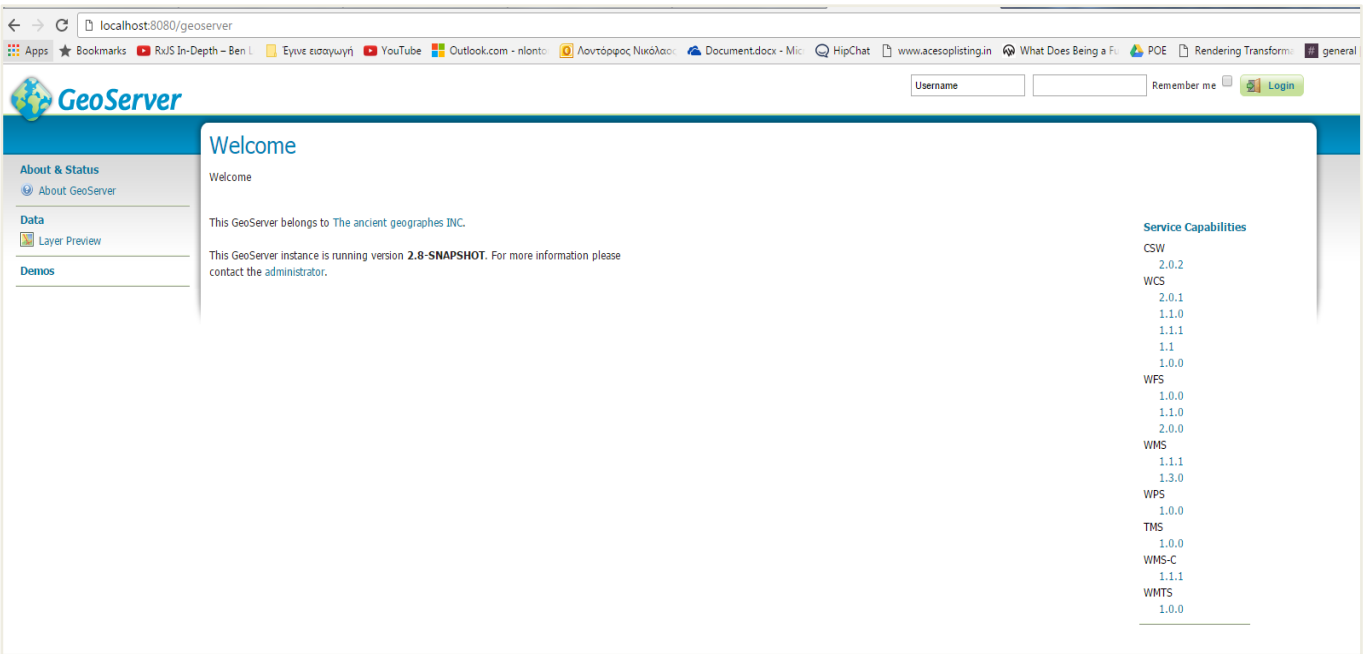
Είδαμε σε προηγούμενο κεφάλαιο τι είναι ο Geoserver και πως χρησιμοποιείται. Στην ενότητα αυτή εξηγούμε πως θα τον εγκαταστήσουμε καθώς και πως δημιουργούμε stores και layers καθώς και πως διαλέγουμε την μορφή του αρχείου που θα εξάγουμε.

1. Το πρώτο που πρέπει να κάνουμε είναι να κατεβάσουμε από το επίσημο site (http://geoserver.org/) την τελευταία Stable έκδοση σε μορφή WAR.



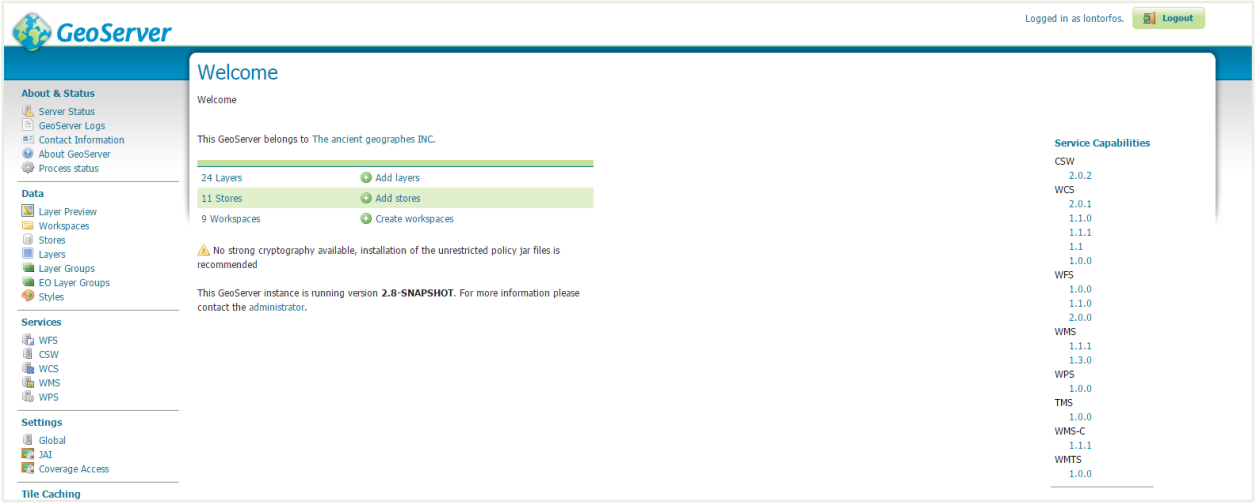
Εικόνα 4 - Geoserver Download

2. Τον κάνουμε deploy στον Tomcat. Όπως είπαμε πιο πριν, τα deployments μας τα διαχειριζόμαστε μέσω του Intellij Idea, οπότε θα δούμε παρακάτω πως γίνεται αυτό με απλό τρόπο. Εάν όλα πήγαν καλά, θα πρέπει εάν ο Tomcat είναι στο port 8080, να έχουμε πρόσβαση στον Geoserver στην διεύθυνση localhost:8080/geoserver



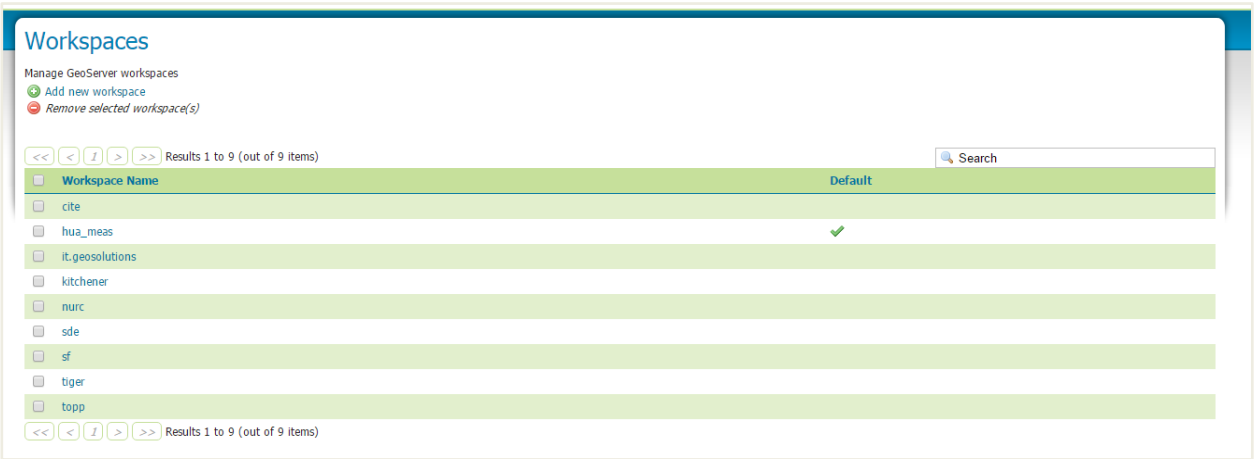
Εικόνα 5 - Geoserver home page

Τα default στοιχεία για να συνδεθούμε είναι admin/geoserver. Αφού συνδεθούμε μπορούμε να δημιουργήσουμε καινούργιο λογαριασμό.



Εικόνα 6 - Geoserver logged in

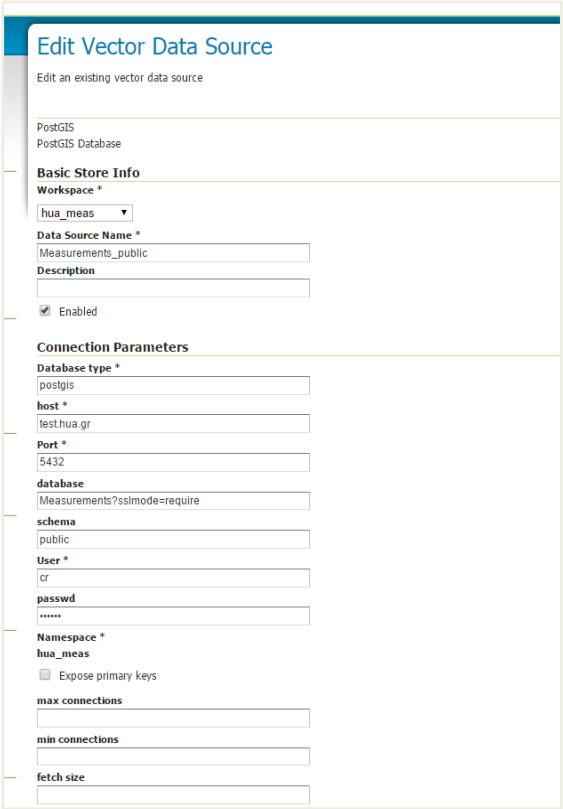
3. Πατάμε στο Workspaces για να φτιάξουμε ένα καινούργιο workspace στο οποίο θα δουλεύουμε. Εμείς φτιάξαμε ένα με την ονομασία hua_meas



Εικόνα 7 - Geoserver Workspaces

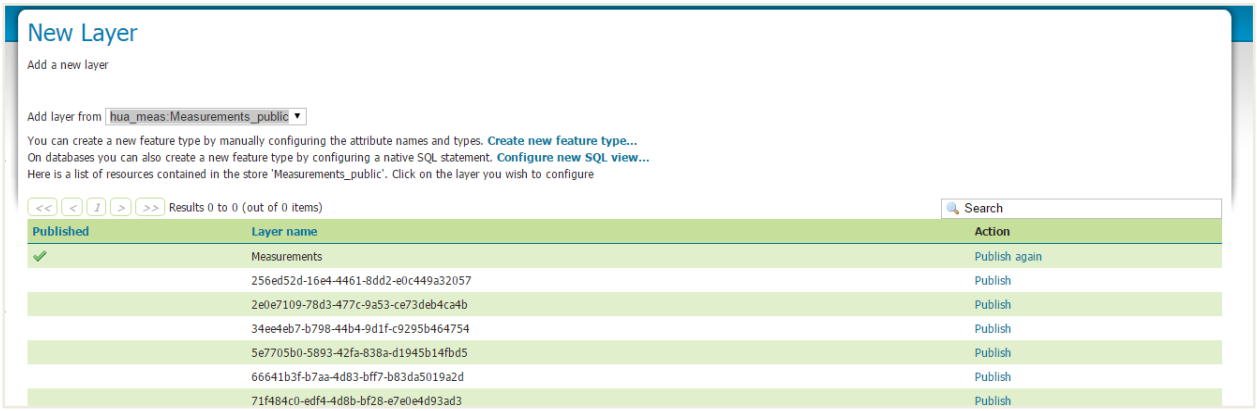
4. Έπειτα πρέπει να πραγματοποιήσουμε μια σύνδεση με την βάση μας. Πηγαίνουμε στο Stores και πατάμε Add new store. Στην επόμενη σελίδα New data source, διαλέγουμε τον τύπο της βάσης μας. Στην περίπτωση μας είναι PostGIS. Οδηγούμαστε στην επόμενη σελίδα όπου θα πρέπει να δώσουμε τα στοιχεία σύνδεσης με την βάση.

   Ονομάσαμε το Store μας Measurements_public. Να σημειωθεί εδώ ότι επειδή ο Geoserver δεν υποστηρίζει ssl προστασία για την βάση, εάν είναι ενεργοποιημένο θα πρέπει να το βάλουμε στο όνομα με την μορφή ?sslmode=require .



Εικόνα 8 - Geoserver Data Source

5. Στο επόμενο βήμα πατάμε στο Layers και Add new resource. Διαλέγουμε το hua_meas:Measurements_public (είναι το Workspace μας : το Store μας) και πατάμε Configure new SQL view...

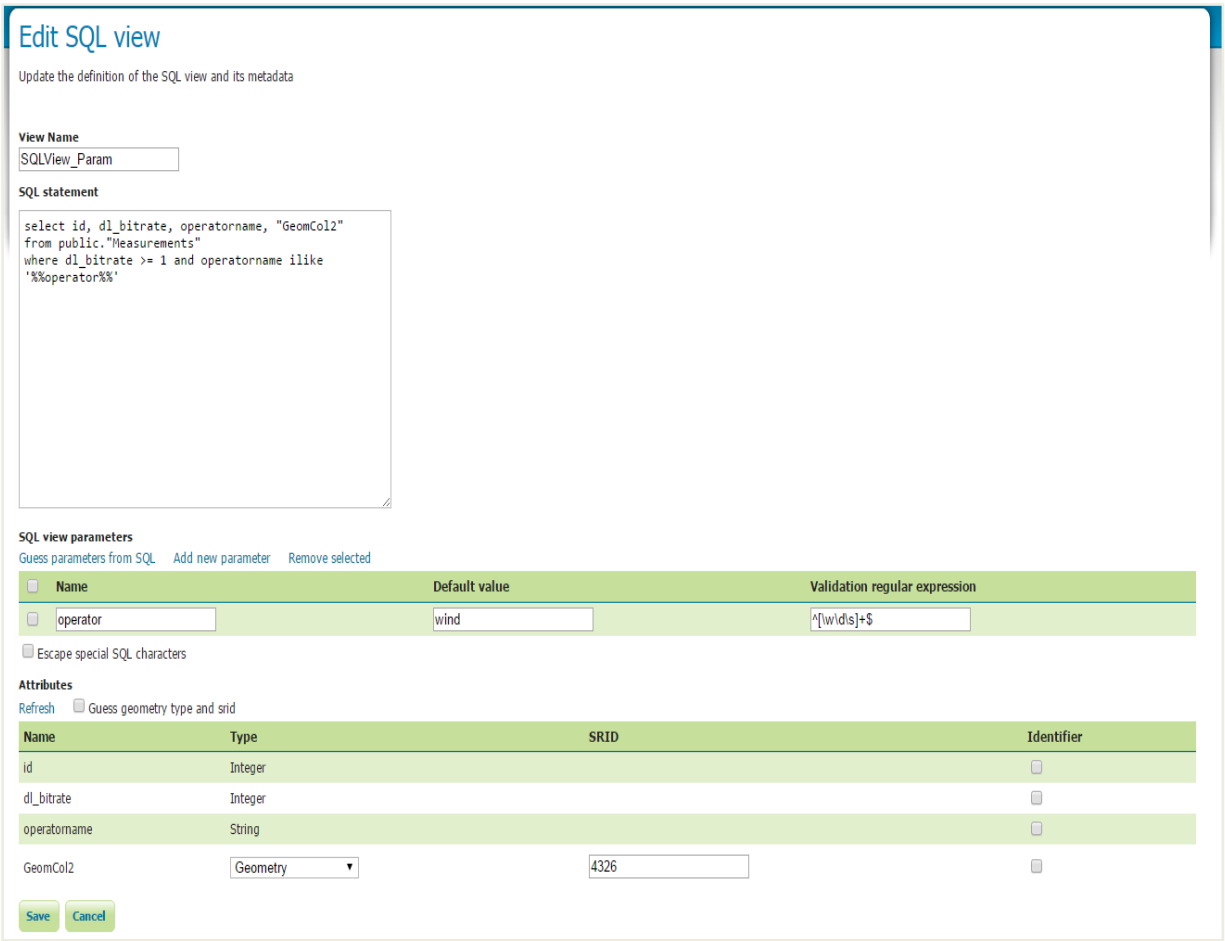


Εικόνα 9 - Geoserver layers

Για τις ανάγκες τις εργασίας φτιάξαμε 2 Layers, δηλαδή 2 παραμετρικά SQL Views, ένα για το Downlink και ένα για το Uplink. Παρακάτω βλέπουμε πως φτιάξαμε το view για το πεδίο dl_bitrate της βάσης μας.

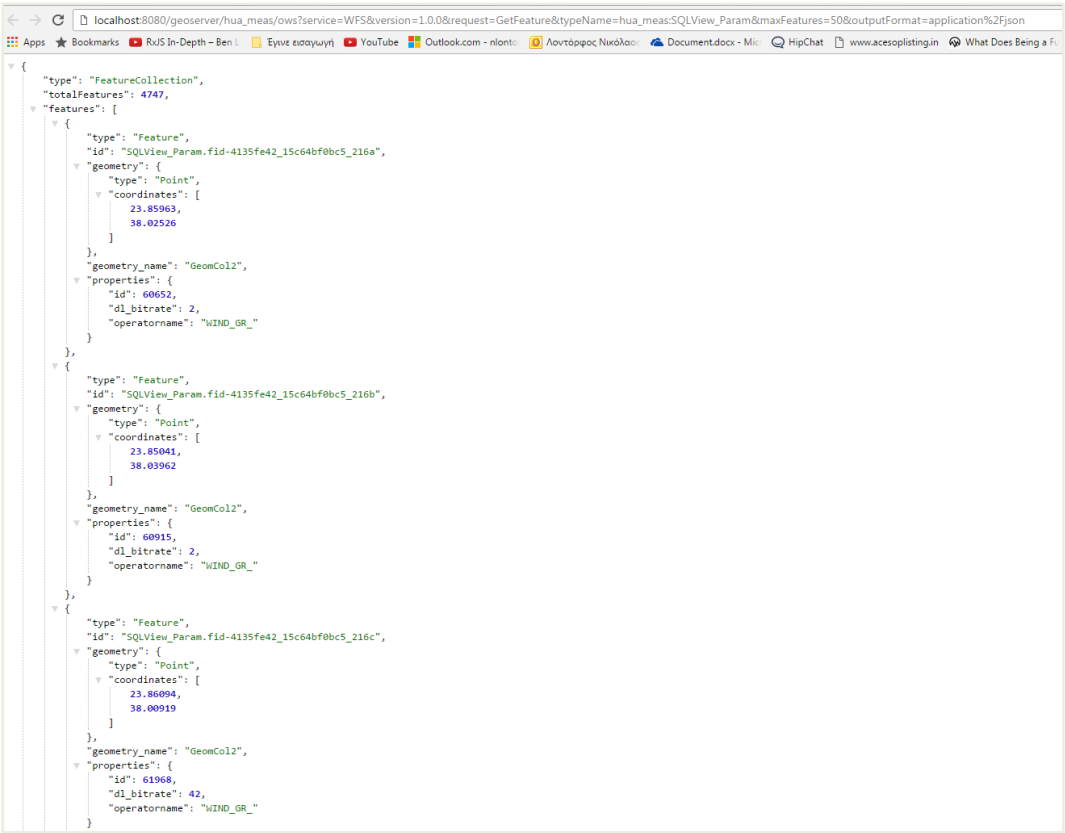
Η παράμετρος '%%operator%%' σημαίνει ότι θα μπορούμε παραμετρικά στο call μας στον Geoserver να θέσουμε την παράμετρο '%operator%', π.χ. wind ή vodafone. Τα δεύτερα σύμβολα "%" μέσα στα οποία είναι το operator σημαίνουν ότι το operator είναι case insensitive όταν θα γίνει το query στο store. Στο πεδίο sql view parameters βάζουμε

για default τιμή του operator την wind. Αυτή την τιμή θα την πάρει άμα στο call δεν βάλουμε καμία παράμετρο.



Εικόνα 10 - Geoserver SQL View

6. Τέλος πάμε στο Layer Preview για να δούμε το αποτέλεσμα μας. Βρίσκουμε το layer μας (στην περίπτωση μας hua_meas:SQLView_Param) και στο drop-down menu All Formats διαλέγουμε WFS - GeoJSON. Το αποτέλεσμα χρησιμοποιώντας ένα beautifier για να είναι ευανάγνωστο το JSON είναι αυτό:



Εικόνα 11 - Parametric JSON

Εάν αφαιρέσουμε από το URI την παράμετρο "&maxFeatures=50" μας επιστρέφεται το σύνολο των μετρήσεων για τον operator wind. Για να μας επιστραφεί κάποιος άλλος operator αρκεί να προσθέσουμε την παράμετρο "&viewparams=operator:τον operator που θέλουμε". Π.χ. test.hua.gr:8080/geoserver/hua_meas/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=hua_meas:SQLView_Param_ul&viewparams=operator:vodafone&outputFormat=application%2Fjson
Στο GeoJson που μας επιστρέφεται παίρνουμε πληροφορίες για το είδος του collection, τον αριθμό των καταχωρήσεων και στη λίστα features παίρνουμε μεμονωμένα την κάθε μέτρηση που πληροί τα κριτήρια μας.

Intellij Idea

Το Intellij Idea είναι ένα integrated development environment (IDE) για την ανάπτυξη Java εφαρμογών. Αναπτύσσεται από την εταιρία JetBrains και διατίθεται σε Community Edition κάτω από το Apache 2 License και εμπορική έκδοση, την Ultimate Edition. Εμείς χρησιμοποιήσαμε την Ultimate Edition εκμεταλλευόμενοι την δωρεάν άδεια ενός χρόνου που δίνει η JetBrains για τα προϊόντα της σε φοιτητές.

Το πρώτο πράγμα που πρέπει να κάνουμε για να το εγκαταστήσουμε είναι να κατεβάσουμε το .tar.gz αρχείο για Linux από το επίσημο site της εταιρίας (https://www.jetbrains.com/idea/#chooseYourEdition)

Κάνουμε unpack το αρχείο:

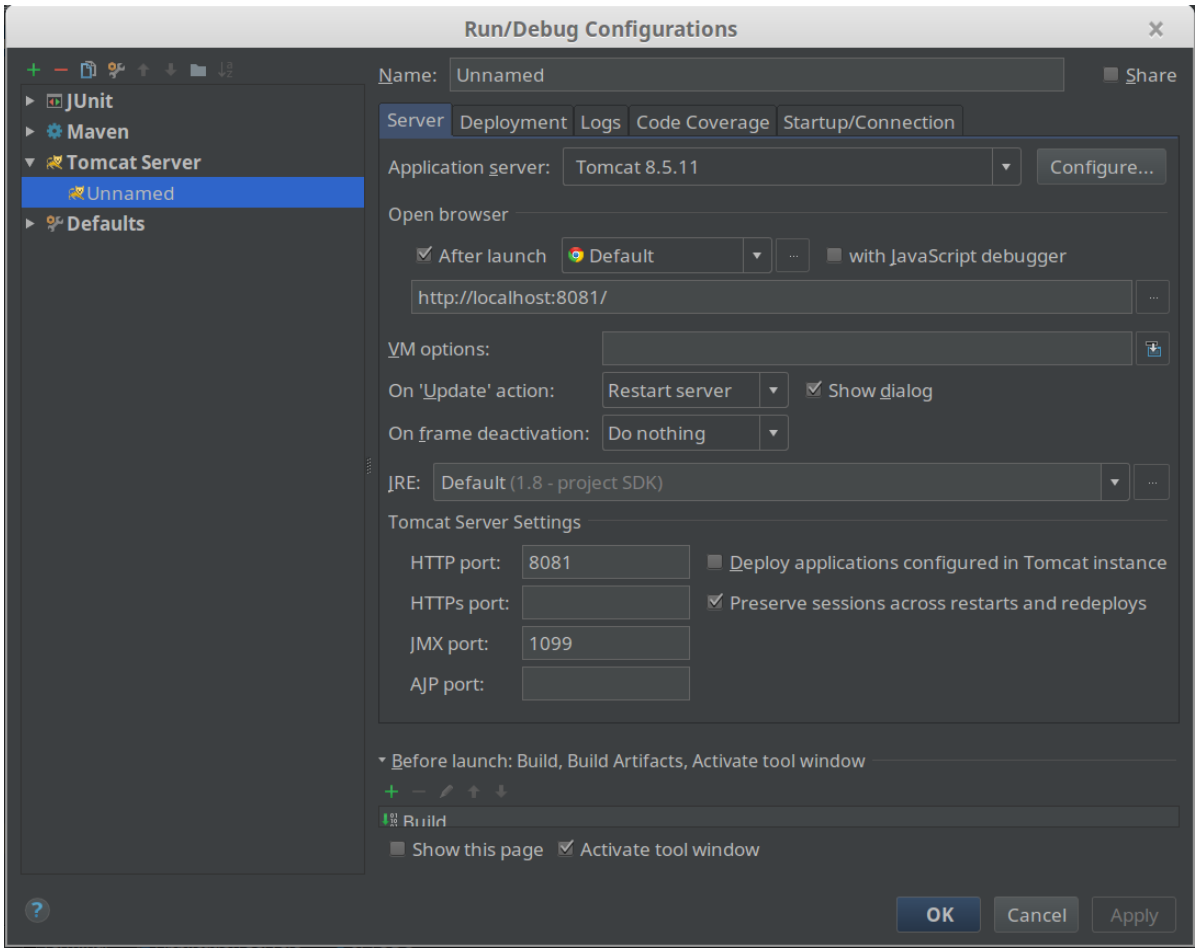
```
$ sudo tar xfz ideaIC -2017.1.3.tar.gz <new_archive_folder>
```

Και αφού μπούμε στο folder που δημιουργήσαμε πληκτρολογούμε

```
$ idea.sh
```

στο /bin directory.

Εφόσον πήγαν όλα καλά, τρέχουμε το πρόγραμμα και πηγαίνουμε στο Run - Edit Configurations ώστε να ρυθμίσουμε τον Tomcat σαν server, ποιό port θα χρησιμοποιεί και σε ποιόν browser.

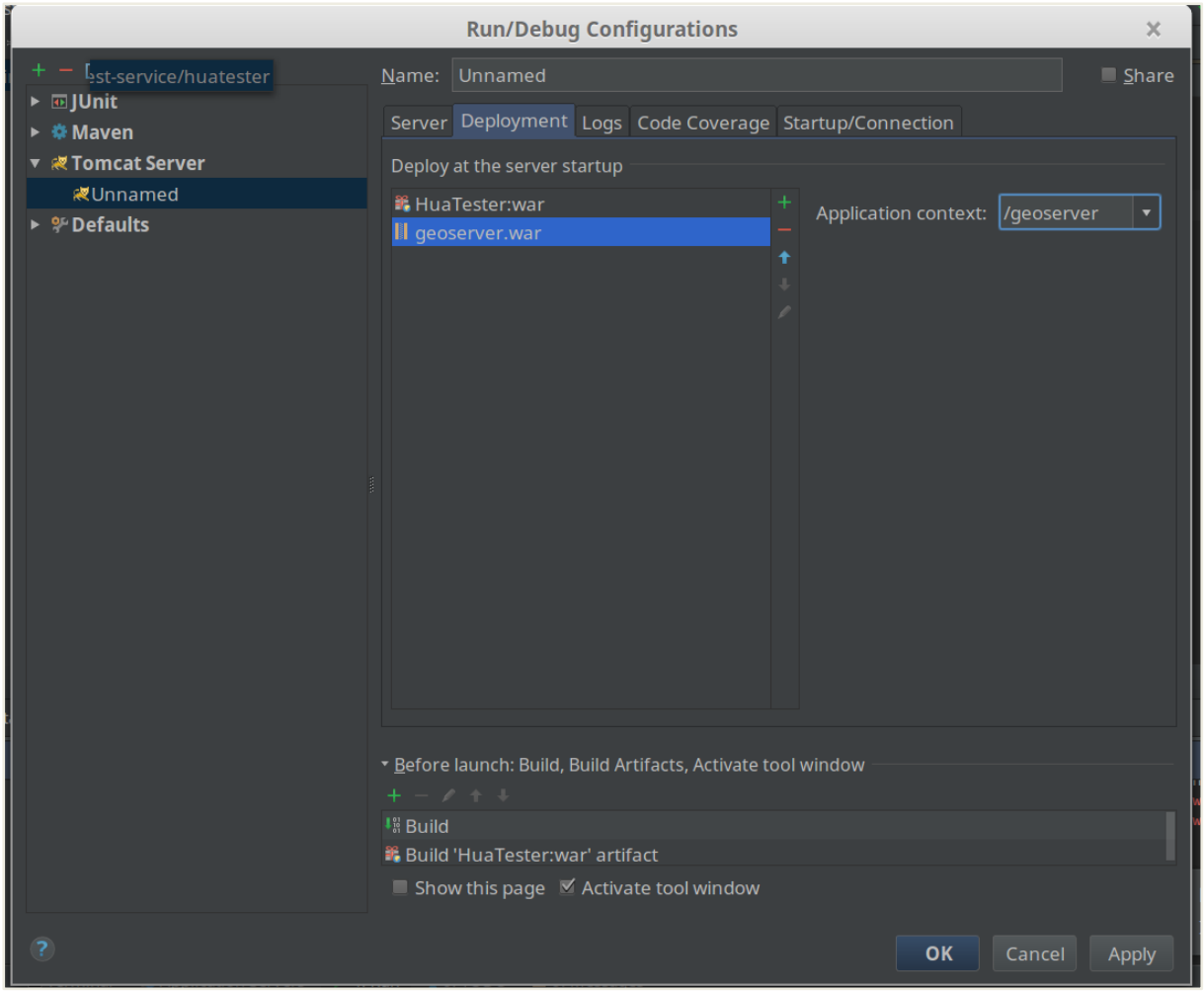


Εικόνα 12 - IDEA Tomcat

Έπειτα πηγαίνουμε στο πεδίο Deployment για να ρυθμίσουμε ποιά WAR θα γίνονται deploy.

Εκεί θα βρούμε ήδη ένα WAR αρχείο με το όνομα της εφαρμογής μας, στην περίπτωση μας HuaTester. Πατάμε το κουμπί με το πράσινο +, external source και διαλέγουμε το WAR αρχείο του Geoserver που κατεβάσαμε στο βήμα 1 του κεφαλαίου 3.2. Προσέχουμε να βάλουμε ως Application context /geoserver.

Πίσω τώρα στο Idea, όταν πατήσουμε Run - Run <Name> θα πρέπει να γίνει deploy και το Spring πρόγραμμα μας (το HuaTester) και ο Geoserver στον Tomcat. Για να τα δοκιμάσουμε αρκεί να γράψουμε στον browser localhost:8081 και localhost:8081/geoserver.



Εικόνα 13 - IDEA Deployment

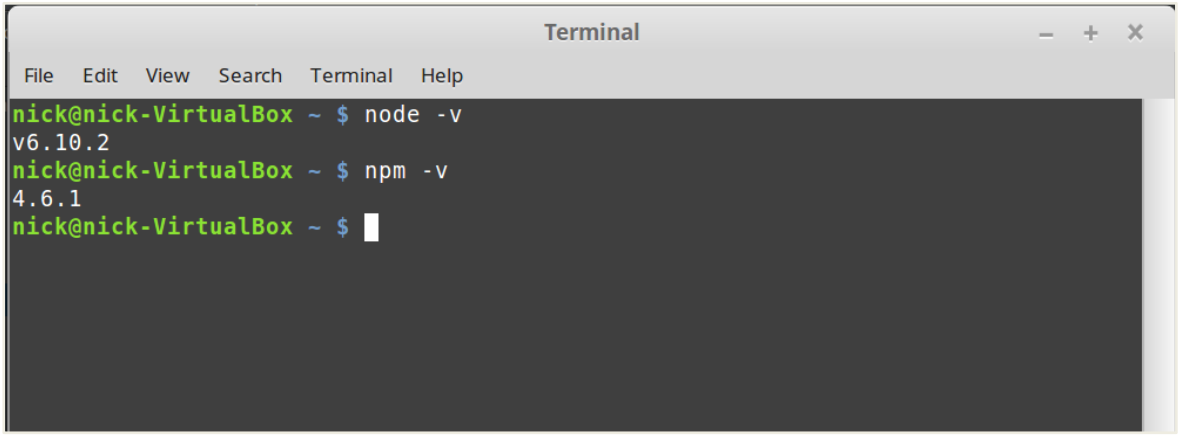
Node.js

Το Node.js (Node.js) είναι ένα open-source, cross-platform Javascript runtime περιβάλλον για την εκτέλεση Javascript κώδικα server-side. Παραδοσιακά η Javascript χρησιμοποιούνταν για την εκτέλεση client-side scripts, τα οποία ενθυλακώνονταν μέσα στην HTML και έτρεχαν στον browser του χρήστη. Το node.js επιτρέπει την χρήση Javascript σε server-side scripting, τρέχοντας τα scripts στον server με σκοπό την παραγωγή δυναμικών σελίδων *πριν* σταλεί η σελίδα στον browser του χρήστη. Είναι απαραίτητο στο μοντέρνο web development με την Angular και άλλες πλατφόρμες και παρέχει και αρκετά build tools.

Το npm είναι και αυτό απαραίτητο στην Angular. Είναι ο package manager του Node, αποτελεί το ίδιο ένα Node application και εγκαθιστά Javascript libraries.

Για την λειτουργία της εφαρμογής μας χρειαζόμαστε την έκδοση 4.x.x και πάνω του Node και την έκδοση 3.x.x και πάνω του npm.

Μπορούμε να ελέγξουμε ποιά έκδοση έχουμε πληκτρολογώντας node -v και npm -v στην κονσόλα.



Εικόνα 14 - Node version

Εάν δεν είναι εγκατεστημένα, για την έκδοση 6.x του Node πληκτρολογούμε:

$ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -

$ sudo apt-get install -y nodejs

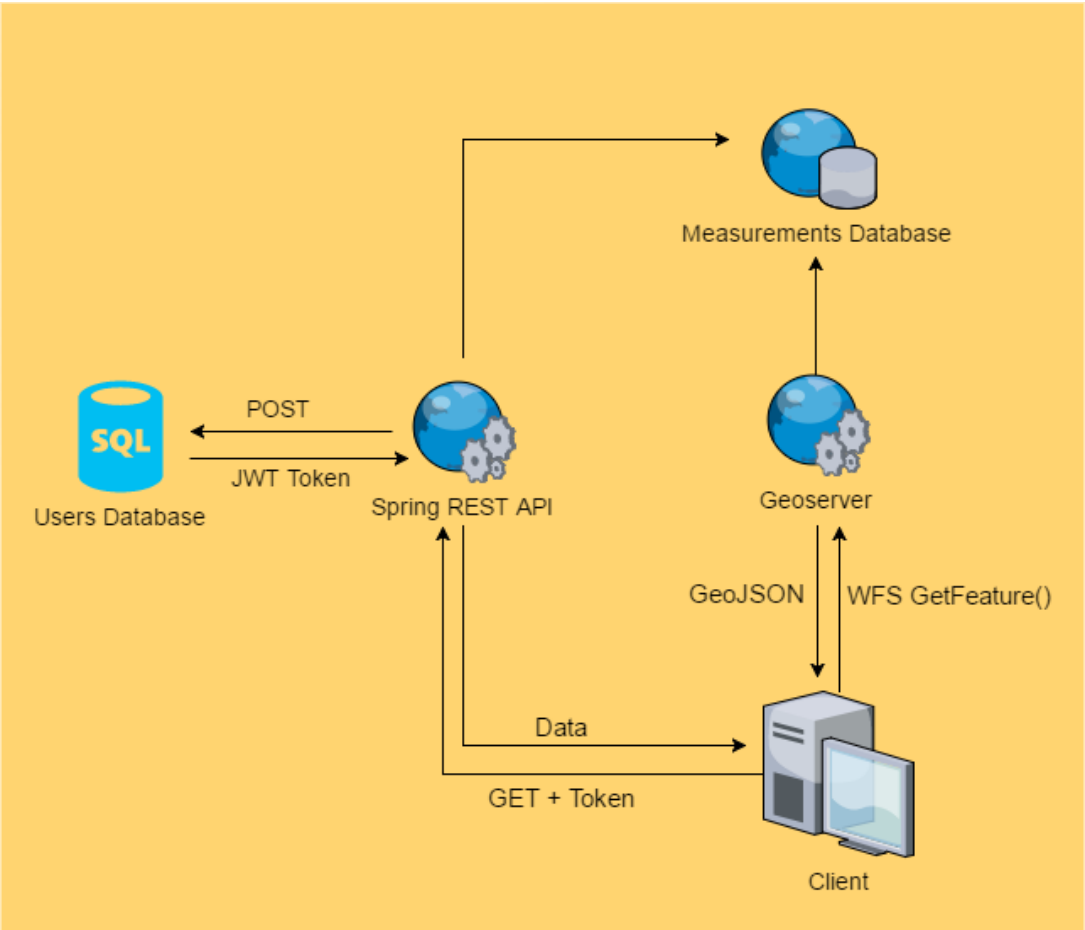
Το npm εγκαθίσταται αυτόματα.

Πληκτρολογώντας $ Nodejs -v και $ npm -v βλέπουμε εάν πήγαν όλα καλά και έχουμε τις σωστές εκδόσεις εγκατεστημένες.

4 Παρουσίαση εφαρμογής

Στο κεφάλαιο αυτό παρουσιάζεται η ανάλυση της πλατφόρμας σε τεχνικό επίπεδο καθώς και ο οδηγός χρήσης της εφαρμογής.

Πρώτα όμως θα πρέπει να πούμε δύο λόγια για την αρχιτεκτονική της εφαρμογής ώστε να καταστεί πιο εύκολη η κατανόηση της ανάλυσης αυτής.



Εικόνα 15 - Αρχιτεκτονική application

Στην παραπάνω εικόνα φαίνεται η ροή των δεδομένων μεταξύ των επιμέρους στοιχείων της εφαρμογής και ο τρόπος με τον οποίο συνεργάζονται.

Αναλυτικότερα ο χρήστης με το που εκκινήσει την εφαρμογή καλείται να δώσει ένα Username και ένα Password τα οποία θα πρέπει να είναι καταχωρημένα στην βάση Users. Εφόσον είναι σωστά, το Spring Security επιστρέφει ένα JWT Token (JSON Web Token) το οποίο αποθηκεύεται τοπικά στον browser του client. Σε κάθε call που κάνει ο χρήστης στο REST API πλέον, προσαρτάται στο Header του GET request το JWT Token και πριν σταλεί απάντηση ελέγχεται η εγκυρότητα του. Εάν είναι έγκυρο, αποστέλλεται στον χρήστη το response σε μορφή JSON. Εάν το JWT δεν είναι έγκυρο, επιστρέφεται μήνυμα λάθους και ο χρήστης είναι υποχρεωμένος να ξαναδώσει username και password.

Όλα τα δεδομένα που ζητάει ο χρήστης και κάνει expose το REST API, βρίσκονται αποθηκευμένα στην Postgres βάση δεδομένων Measurements. Η βάση αυτή αποτελεί το σημαντικότερο κομμάτι του Application μας αφού εκεί μέσα βρίσκονται αποθηκευμένα τα γεωχωρικά δεδομένα μας από τα οποία στόχος μας είναι να εξάγουμε επιχειρηματική ευφυΐα. Στην βάση αυτή είναι συνδεδεμένος και ο Geoserver, ο οποίος χρησιμοποιείται για την δημιουργία Heatmaps από τα δεδομένα μας. Το Heatmap είναι ένας Vector-to-Raster μετασχηματισμός, ο οποίος απεικονίζει ένα dataset σε μία επιφάνεια, λαμβάνοντας διαφορετικά χρώματα ανάλογα με την πυκνότητα και το βάρος των δεδομένων. Όταν ο Geoserver λάβει ένα request, επιστρέφει ένα GeoJSON αρχείο στον client και η Angular σε συνδυασμό με το Google Maps αναλαμβάνουν την εμφάνιση του Heatmap. Συνοπτικά, αυτή ήταν η περιγραφή της συνολικής λειτουργίας και της αρχιτεκτονικής του Web Application μας και της ροής δεδομένων του. Στα επόμενα κεφάλαια παρουσιάζουμε την τεχνική ανάλυση των επιμέρους μερών του application που είδαμε μέχρι τώρα και έναν οδηγό χρήσης της εφαρμογής.

4.1 Τεχνική ανάλυση

Βάση Δεδομένων

Η τεχνική ανάλυση της πλατφόρμας μας θα ξεκινήσει από το πιο βασικό κομμάτι της εφαρμογής, το εσωτερικό της βάσης δεδομένων μας. Η βάση Measurements είναι μια Postgres βάση δεδομένων, με εγκατεστημένο το extension PostGIS το οποίο δίνει την δυνατότητα για queries βασισμένα στην τοποθεσία. Τα δεδομένα που είναι αποθηκευμένα προήλθαν από καθηγητή μου, Δρ. Βασίλειο Δαλάκα. Το εργαλείο διαχείρισης που χρησιμοποιήσαμε είναι το DBeaver(DBeaver) και την στιγμή της εκπόνησης της παρούσας διπλωματικής στην βάση Measurements υπήρχαν 655.000 καταχωρήσεις.

Στην παρακάτω εικόνα φαίνεται ο πίνακας Measurements:

Εικόνα 16 - Measurements table

Πίνακας "Measurements"

Αναλυτικά τα πεδία του:

- **id:** Μοναδικός αύξοντας αριθμός για κάθε εγγραφή
- **GeomCol1:** Geometry τύπου byte
- **timestamp:** Timestamp με ακριβή ημερομηνία και ώρα λήψης μέτρησης
- **lon:** Γεωγραφικό μήκος
- **lat:** Γεωγραφικό πλάτος
- **level:** Η δύναμη του σήματος σε dB
- **speed:** Η ταχύτητα με την οποία κινούνταν η συσκευή την στιγμή της μέτρησης
- **operatorname:** Όνομα παρόχου κινητής τηλεφωνίας
- **mcc:** Mobile country code, Mobile country code, 3ψήφιος αριμός μοναδικός για κάθε χώρα.
- **mnc:** Mobile network code, 2ψήφιος ή 3ψήφιος αριμός μοναδικός γιά κάθε πάροχο στην χώρα
- **node:** Ο κόμβος του παρόχου
- **cellid:** Μοναδικό id της κυψέλης
- **lac:** Location area code
- **network_type:** Ο τύπος του δικτύου, πχ 2G, 3G
- **qual:** Η ποιότητα του σήματος
- **snr:** Signal to noise ratio
- **cqi:** Channel quality indicator
- **lterssi:** LTE Received Signal Strength Indicator
- **appversioncode:** Έκδοση της πλατφόρμας μετρήσεων
- **psc:** Public Service Commission
- **dl_bitrate:** Απόδοση ταχύτητας κατεβάσματος (downlink)
- **ul_bitrate:** Απόδοση ταχύτητας ανεβάσματος (uplink)
- **nlac1:** Βοηθητικό πεδίο που αφορά το local area code
- **ncellid1:** Βοηθητικό πεδίο που αφορά το cell id
- **nrxlev1:** Βοηθητικό πεδίο που αφορά το level
- **nlac2:** Βοηθητικό πεδίο που αφορά το local area code
- **ncellid2:** Βοηθητικό πεδίο που αφορά το cell id
- **nrxlev2:** Βοηθητικό πεδίο που αφορά το level
- **nlac3:** Βοηθητικό πεδίο που αφορά το local area code
- **ncellid3:** Βοηθητικό πεδίο που αφορά το cell id
- **nrxlev3:** Βοηθητικό πεδίο που αφορά το level
- **nlac4:** Βοηθητικό πεδίο που αφορά το local area code
- **ncellid4:** Βοηθητικό πεδίο που αφορά το cell id
- **nrxlev4:** Βοηθητικό πεδίο που αφορά το level
- **nlac5:** Βοηθητικό πεδίο που αφορά το local area code
- **ncellid5:** Βοηθητικό πεδίο που αφορά το cell id
- **nrxlev5:** Βοηθητικό πεδίο που αφορά το level
- **nlac6:** Βοηθητικό πεδίο που αφορά το local area code
- **ncellid6:** Βοηθητικό πεδίο που αφορά το cell id
- **nrxlev6:** Βοηθητικό πεδίο που αφορά το level
- **ctime:** Timestamp με ακριβή ημερομηνία και ώρα καταχώρησης μέτρησης
- **event:** Τυχών συμβάντα κατά την διάρκεια της μέτρησης (π.χ. errors)

- **accuracy:** Η ακρίβεια τοποθεσίας της μέτρησης
- **locationsource:** Η πηγή απόκτησης της τοποθεσίας
- **altitude:** Το υψόμετρο όπως μετρήθηκε από το GPS
- **conntype:** Ο τύπος σύνδεσης
- **conninfo:** Έξτρα πληροφορίες για την σύνδεση
- **avgping:** Μέσο ping στους server του Χαροκοπείου
- **minping:** Ελάχιστο ping στους server του Χαροκοπείου
- **maxping:** Μέγιστο ping στους server του Χαροκοπείου
- **stdevping:** Τυπική απόκλιση ping στους server του Χαροκοπείου
- **pingloss:** Απώλειες στο ping στους server του Χαροκοπείου
- **testdlbitrate:** Βοηθητικό πεδίο για την μέτρηση του downlink
- **testulbirate:** Βοηθητικό πεδίο για την μέτρηση του uplink
- **GeomCol2:** Geometry τύπου Point
- **DeviceManufacturer:** Ο κατασκευαστής του κινητού
- **DeviceModel:** Το μοντέλο του κινητού
- **DeviceName:** Το όνομα της συσκευής
- **VersionName:** Το όνομα της έκδοσης
- **Brand:** Το εμπορικό όνομα του λειτουργικού
- **AndroidVersion:** Η έκδοση Android της συσκευής
- **ServingCellTime:** Ο χρόνος εξυπηρέτησης της κυψέλης σε δευτερόλεπτα
- **os:** Το λειτουργικό σύστημα της συσκευής
- **os_id:** Το id του λειτουργικού της συσκευής
- **camp_id:** Το id του campaign
- **ssid:** Το service set identifier του wireless network

Java Spring RESTful API

Το επόμενο βήμα στην σχεδίαση της πλατφόρμας μας είναι η σχεδίαση του REST API.

Πρώτα όμως θα πρέπει να εξηγήσουμε τι είναι ένα REST API.

Ο όρος API σημαίνει Application Programmable Interface. Χρησιμοποιείται συνήθως για να περιγράψει τα χαρακτηριστικά μιας βιβλιοθήκης ή το πως να μπορούμε να αλληλεπιδράσουμε μαζί της. Οι βιβλιοθήκες αυτές συνήθως έχουν κάποιο Documentation στον οποίο εξηγούν ποιές functions είναι διαθέσιμες, πως τις καλούμε, τι ορίσματα χρειάζονται κτλ. Τα προβλήματα άρχισαν να εμφανίζονται όταν άρχισε ο καθένας να υλοποιεί το δικό του API. Χωρίς μια στάνταρ δομή στα URL, κρίθηκε απαραίτητη η ύπαρξη ενός documentation που να εξηγεί πως λειτουργεί το API. Για παράδειγμα ένα API το οποίο προβάλει widgets κάποιος θα μπορούσε να το γράψει `/view_widgets` και κάποιος άλλος `/widgets/all`.

Την λύση ήρθε να δώσει το REST. Τα αρχικά σημαίνουν Representational State Transfer και αποτελεί το standard για την δημιουργία HTTP APIs. Η αρχή του είναι ότι, για την πραγματοποίηση των συνηθέστερων actions που χρησιμοποιούνται στο Internet (view, create, edit, and delete) υπάρχει ακριβής αντιστοίχηση σε HTTP verbs τα οποία είναι ήδη υλοποιημένα (GET, POST, PUT, DELETE). Με αυτό τον τρόπο και χρησιμοποιώντας το URL σαν

resource (`/widgets` για όλα τα widgets ή `/widgets/123` για το widget με id=123) και για πρόθεμα ένα από τα 4 verbs μπορούμε με εύκολο τρόπο να περιγράψουμε την δράση που θέλουμε να πραγματοποιήσουμε.

Π.χ. `GET http://example.com/widgets/` ή

```
POST http://example.com/widgets
Data:
    name = Foobar
```

Αυτή είναι η μορφή ενός HTTP Request. Η απάντηση (το HTTP Response) συνήθως επιστρέφει στο format του JSON (JSON).

Το δικό μας REST API επιλέξαμε να το υλοποιήσουμε με το Spring Framework σε Java, σε συνδυασμό με κάποιες εξτρά επεκτάσεις όπως το Maven(Maven), το Hibernate (Hibernate) και το Spring Security(Spring Security) . Χρησιμοποιήσαμε το Spring Boot για ξεκινήσουμε να κάνουμε build το Project. Αυτό μας βοήθησε σε θέματα όπως το import και ο συγχρονισμός των libraries και τα deployment να γίνονται με πολύ εύκολο τρόπο. Την δυνατότητα αυτή μας την έδωσε το Maven το οποίοι βοηθάει στο να πακετάρουμε εκτελέσιμα Jars και Wars και να τρέχουμε το application μας παντού.

Maven

Το μόνο που χρειάζεται το Maven για να κάνει configure το build μας είναι ένα pom.xml αρχείο. Τα αρχείο αυτό χωρίζεται σε 3 sections: general properties, dependencies και το ίδιο το build. Οι πληροφορίες αυτές αρκούν ώστε να γίνει build το project μας όπως το θέλουμε.

Παράρτημα: pom.xml

Hibernate

Το Hibernate είναι ένα object-relational mapping εργαλείο για Java. Η βασική του λειτουργία είναι ότι κάνει map Java κλάσεις σε tables βάσεων δεδομένων και Java data types σε SQL data types. Αυτό μας έδωσε την δυνατότητα να μην χρειάζεται να ασχοληθούμε με την μετατροπές των αποτελεσμάτων των queries σε αντικείμενα και μας έδωσε επίσης έτοιμες μεθόδους για την πρόσβαση στα tables της βάσης όπως getters και setters.
Παράδειγμα από την persistent class μας:

```
package gr.huatester.model;

import java.io.Serializable;
import javax.persistence.*;

import com.fasterxml.jackson.annotation.JsonAutoDetect;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonInclude;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

import java.math.BigDecimal;
import java.sql.Timestamp;
```

```
/**
 * The persistent class for the "Measurements" database table.
 *
 */
@ApiModel
@Entity
@Table(name="\"Measurements\"")
@NamedQueries({ @NamedQuery(name = "Measurements.findAll", query = "SELECT m FROM Measurement
m"),
@NamedQuery(name = "Measurements.findByID", query = "SELECT m FROM Measurement m WHERE m.id =
:id")})

public class Measurement implements Serializable {
	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue(strategy=GenerationType.AUTO)
	@Column(unique=true, nullable=false)
	private Integer id;

	private Integer accuracy;

	private Integer altitude;


	public Measurement() {
	}

	public Integer getId() {
		return this.id;
	}

	public void setId(Integer id) {
		this.id = id;
	}
```

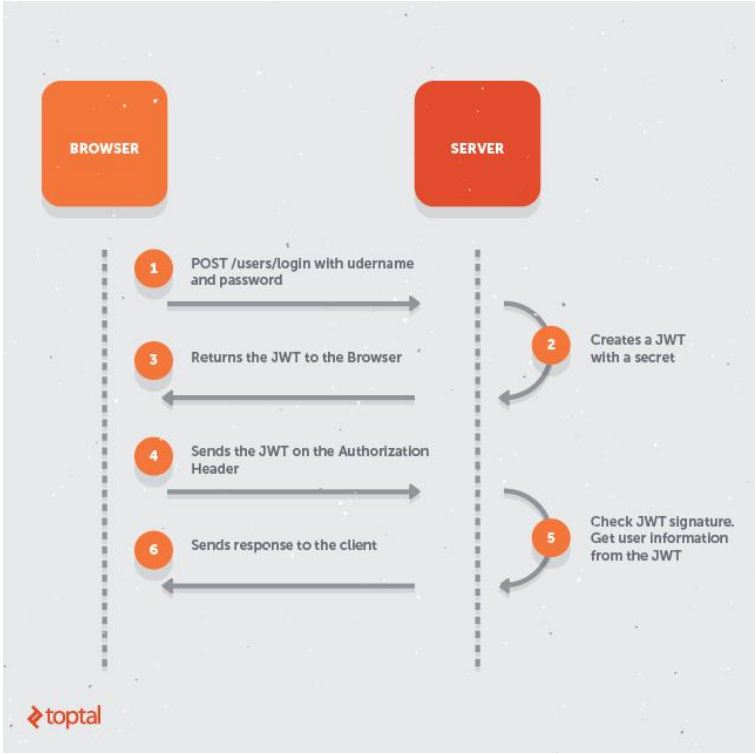
Με αυτό τον τρόπο μπορούμε να έχουμε πρόσβαση σε καταχωρήσεις της βάσης δεδομένων, ο οποίες μετατρέπονται σε Java αντικείμενα, υπόκεινται σε ότι επεξεργασία απαιτείται και επιστρέφονται μέσω του API σε μορφή JSON.

Spring Security

Το Spring security είναι ένα framework που παρέχει authentication, authorization και άλλα features ασφαλείας σε applications. Η δύναμη του βρίσκεται στο πόσο εύκολα μπορεί ένας χρήστης να το επεκτείνει στις προσωπικές του απαιτήσεις.

Τα RESTFul services και η ευκολία στην δημιουργία και την κατανάλωση τους, έφεραν μαζί και ένα σημαντικό μειονέκτημα: την πτώση του επιπέδου ασφαλείας. Κάποια παραδείγματα είναι το session hijacking και το cross-site request forgery (XSRF). Ο λόγος για τα προβλήματα είναι ότι το REST είναι stateless, πράγμα που σημαίνει ότι ο server δεν κρατάει πληροφορίες για την κατάσταση του χρήστη (sessions). Πολλά application που χρειαζόντουσαν αυτή την πληροφορία, άρχισαν να χρησιμοποιούν session cookies παρά το κενό ασφαλείας που

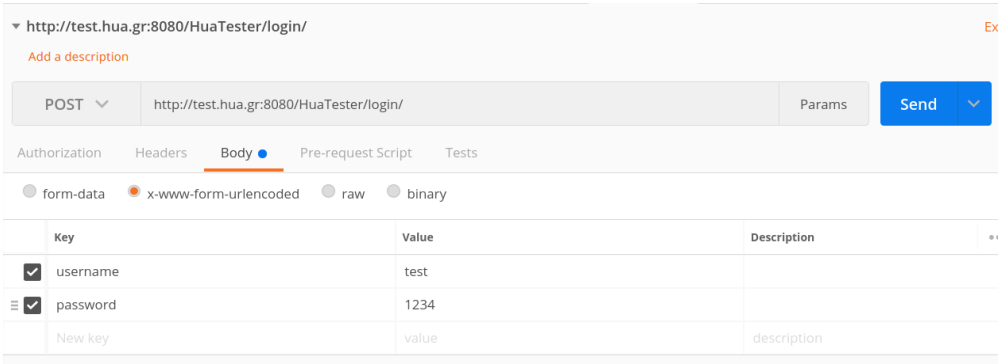
δημιουργούσαν. Το πρόβλημα αυτό λύθηκε με τα authentication tokens. Το standard των authorization tokens είναι το JWT (JWT). Έχει μικρό μέγεθος, είναι εύκολο στην χρήση και είναι ασφαλές για την μεταφορά του identity του χρήστη μεταξύ ενός identity provider και ενός service provider. Μπορεί ακόμα να κουβαλάει την πιστοποίηση της ταυτότητας ενός χρήστη, τα authorization data του δηλαδή, βοηθώντας τον server να μην πρέπει να επικοινωνεί με την βάση ή άλλα συστήματα για να αναγνωρίσει τον ρόλο του χρήστη. Παρακάτω φαίνεται η υλοποίηση του JWT:



Εικόνα 17 - JWT

Με αυτό τον τρόπο υλοποιήσαμε και εμείς το Authentication μας.

Ο χρήστης πρέπει πρώτα να επικοινωνήσει με το endpoint `/login` και να στείλει με ένα POST request στο body του μηνύματος ένα username και ένα password.



Εικόνα 18 - Login request

Εάν αυτά είναι σωστά (είναι καταχωρημένα στην βάση users δηλαδή), ο server επιστρέφει στο header του response ένα token. Αυτό το token θα πρέπει να το προσαρτήσουμε στο header κάθε request που θα κάνουμε στο υπόλοιπο API. Όταν ο server δεχτεί το request, ελέγχει την εγκυρότητα του token και εφόσον είναι σωστό, επιστρέφει την απάντηση του.

Body Cookies Headers (10) Tests Status: 200 OK Time: 137 ms

Pretty Raw Preview JSON Save Response


```
1 {
2   "email": "test@hua.gr",
3   "id": 2,
4   "country": null,
5   "password": "1234",
6   "userName": "test"
7 }
```


Εικόνα 19 - Login response body

Body Cookies Headers (10) Tests Status: 200 OK Time: 137 ms

Access-Control-Allow-Headers → token, x-requested-with, origin, Access-Control-Allow-Origin, Access-Control-Allow-Headers

Access-Control-Allow-Methods → POST, GET, OPTIONS, PUT

Access-Control-Allow-Origin → *

Access-Control-Expose-Headers → token, x-requested-with, Access-Control-Allow-Origin, Access-Control-Allow-Headers

Access-Control-Max-Age → 3600

Content-Type → application/json;charset=UTF-8

Date → Wed, 07 Jun 2017 15:12:08 GMT

Server → Apache-Coyote/1.1

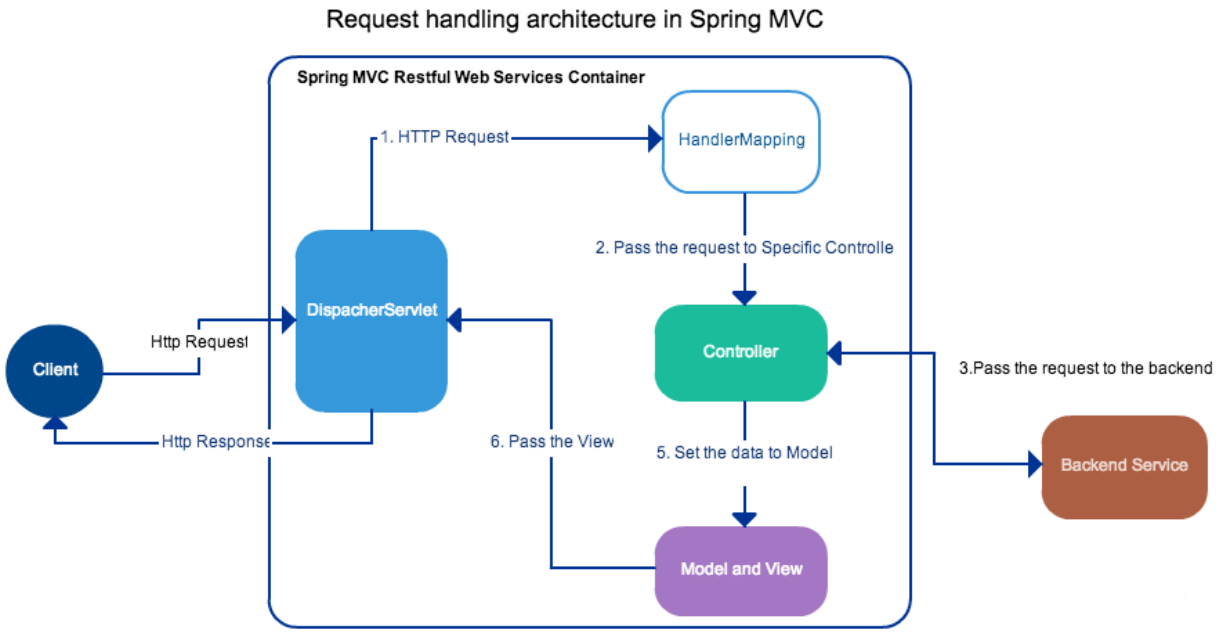
Token → eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ0ZXN0IiwianRpIjoiMiIsImV4cCI6MTQ5Njg2NjMyOCwiaXNzIjoiZ3IuaHVhdGVzdGVyIn0.XaBmroeqP7GHCd5iMsvXwkfJi9tWJmVaf-34pEr06aLIYElO0H7Iiue6vejhyplo8Ym04Mdnt_XnLYUYADq7jw

Transfer-Encoding → chunked

Εικόνα 20 - Login response header

Spring MVC

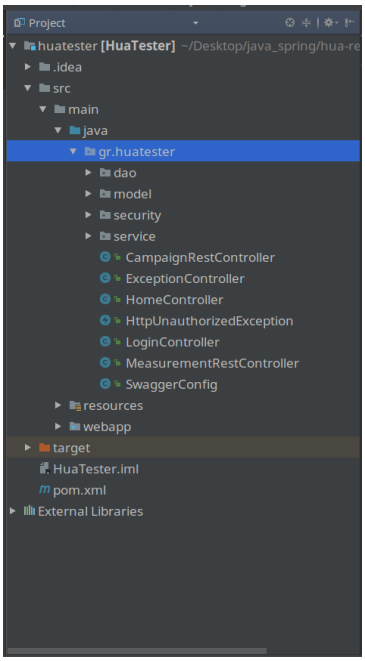
Ας δούμε την αρχιτεκτονική διαχείρισης των request από το Spring:



Εικόνα 21 - Spring MVC

Στο παραπάνω διάγραμμα βλέπουμε τον Controller ο οποίος είναι υπεύθυνος για την πραγματοποίηση των request. Αποτελεί καλή πρακτική το να μην γράφουμε κώδικα που αφορά το business logic στον controller. Η αρχιτεκτονική του Spring MVC γενικά επιβάλει τον διαχωρισμό της λογικής σε layers με κοινή λειτουργικότητα. Γι αυτό χρησιμοποιούμε το backend service (την βάση) για την παροχή business logic.

Παρακάτω βλέπουμε το Project structure μας, υλοποιημένο όπως είδαμε με Spring MVC και Hibernate. Χρησιμοποιήσαμε Maven project στο Intellij Idea:



Εικόνα 22 - Spring Structure

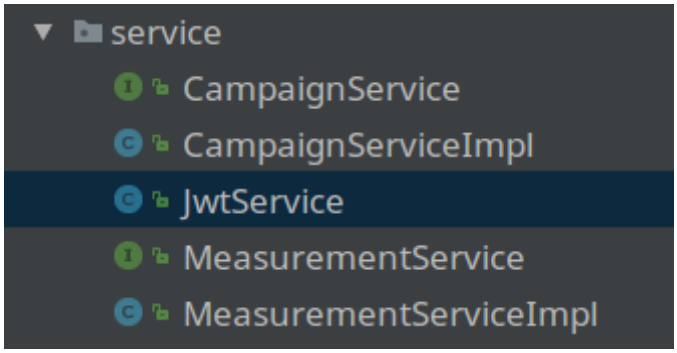
Τα layers δημιουργήθηκαν με βάση τον διαχωρισμό των αρμοδιοτήτων τους:

- **Controller Layer** (MeasurementRestController)

  Στο controller class δηλώσαμε τα request mappings χρησιμοποιώντας το annotation @RequestMapping. Με αυτό τον τρόπο κάνουμε redirect το request στο σωστό service του service layer. Έπειτα πρέπει τα δεδομένα αυτά να γίνουν inject σε ένα μοντέλο, το οποίο θα σταλεί στο view.

  Παράρτημα: MeasurementRestController

- **Service Layer** (ο φάκελος service)



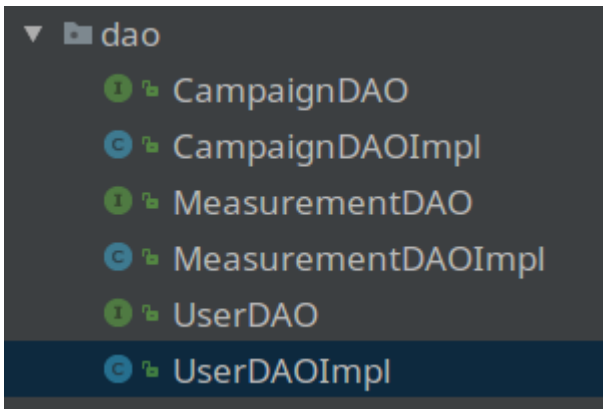
Εικόνα 23 - Service Layer

Στο layer αυτό χρησιμοποιήσαμε το JwtService για την παραγωγή του Jwt Token όπως είδαμε στο Spring Security καθώς και τα MeasurementService και MeasurementServiceImpl services, τα οποία σκοπό έχουν την διαχείριση των transactions με την βάση δεδομένων. Το @Transactional annotation δηλώνει ότι θέλουμε να αρχίσουμε το transaction πριν εκτελεστούν τα operations. Τον μηχανισμό αυτό τον διαχειρίζεται το Spring. Το μόνο που μας απασχολεί εμάς είναι οι ρυθμίσεις του Spring οι οποίες βρίσκονται στο αρχείο **servlet-context.xml**.

Το αρχείο αυτό δίνει οδηγίες στο spring framework για την δημιουργία instances του sessionFactory, όπως το πως να κάνει Inject references σε services (όπως το MeasurementServiceImpl) χρησιμοποιώντας κάποιο instance data access model (όπως το MeasurementsDAO που θα δούμε παρακάτω).

Παράρτημα: Servlet-context.xml, JWTService, MeasurementService, MeasurementServiceImpl

- Data Access Layer (ο φάκελος DAO)



Εικόνα 24 - Data Access Layer

Σε κάθε layer χρειαζόμαστε interfaces τα οποία διαχειρίζονται το functionality και τα αντίστοιχα implementation classes. Τα data access object σκοπό έχουν να πραγματοποιούν raw transactions με την βάση δεδομένων και να επιστρέφουν τύπους υψηλότερου επιπέδου (objects, collections κα.). Τα services καλούν τα DAO για την υλοποίηση business logic. Στην εφαρμογή μας χρησιμοποιήσαμε τα UserDAO και MeasurementDAO Interfaces και τις UserDAOImpl και MeasurementDAOImpl κλάσεις. Το UserDAO το χρησιμοποιούμε για την αποθήκευση και ανάκτηση χρηστών από την βάση μας και το MeasurementDAO γιά την ανάκτηση μετρήσεων από την βάση Measurements.

Να σημειωθεί οτί το Hibernate δημιούργησε μόνο του κάποιες βασικές μεθόδους όπως add και update Measurements ή αναζήτηση με ID. Τις υπόλοιπες μεθόδους τις δημιουργήσαμε εμείς με HQL (Hibernate Query Language), μια γλώσσα για query που χρησιμοποιεί το Spring με παρόμοια σύνταξη με την SQL η οποία χρησιμοποιεί class names αντί για table names. Ο κύριος λόγος που χρησιμοποιήσαμε HQL είναι ο όγκος των δεδομένων, όπως είδαμε και πιο πριν. Το να επιστρέψει το Spring όλες τις μετρήσεις ή έστω τις μετρήσεις από ένα μόνο provider αποδείχτηκε αρκετά χρονοβόρο καθώς ήτανε ένα αρχείο πολλών MB. Έτσι αναγκαστήκαμε να μεταφέρουμε το buisness logic στην βάση και να επιστρέφουμε μόνο την απάντηση στο ερώτημα με ένα αρχείο JSON μεγέθους λίγων kb.

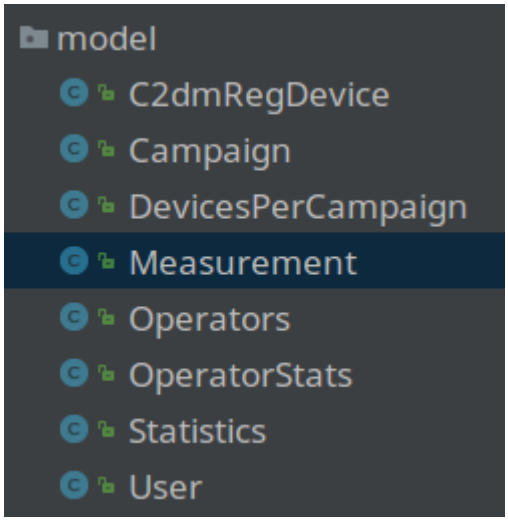
Για παράδειγμα η μέθοδος **public** List<Operators> findAll() που βλέπουμε παρακάτω στο MeasurementImpl service, επιστρέφει ένα JSON αρχείο της μορφής:

```
[
        { "operatorname": "COSMOTE", "value": 294203 },
        { "operatorname": "VODAFONE", "value": 336569 },
        { "operatorname": "WIND", "value": 23604 },
        { "operatorname": "OTHER", "value": 624 }
]
```

Στο αρχείο αυτό μας επιστρέφονται τα COUNT για κάθε distinct provider, με case-insensitive τρόπο και με ένα μόνο call. Αντίστοιχη λογική χρησιμοποιήσαμε και στις υπόλοιπες μεθόδους.

Παράρτημα: UserDAO, UserDAOImpl, MeasurementDAO, MeasurementDAOImpl

- Persistence Layer (ο φάκελος model)



Εικόνα 25 - Persistence Layer

Στο persistence layer έχουμε 2 είδη μοντέλων που χρησιμοποιήσαμε.
Τα μοντέλα Measurement και User χρησιμοποιήθηκαν για το mapping των αντίστοιχων objects στην σχεσιακή βάση δεδομένων μας. Με αυτό τον τρόπο η πρόσβαση στην βάση απλοποιήθηκε αφού αντί για άμεσα query, μπορούμε να χρησιμοποιούμε μεθόδους αντικειμένων σε υψηλό επίπεδο.
Τα μοντέλα Operators, OperatorStats και Statistics χρησιμοποιήθηκαν από τον controller μας. Όταν ένα νέο request γίνεται redirect στο αντίστοιχο service, τα δεδομένα που επιστρέφονται γίνονται inject σε ένα μοντέλο για να σταλούν μέσω του dispatcher στο response.
Για παράδειγμα, το URL `api/measurements/allproviders` επιστρέφει ένα JSON της μορφής:

```
[
        { "operatorname": "COSMOTE", "value": 294203 },
        { "operatorname": "VODAFONE", "value": 336569 },
        { "operatorname": "WIND", "value": 23604 },
        { "operatorname": "OTHER", "value": 624 }
]
```

Στην ουσία το JSON αυτό είναι ένας πίνακας που αποτελείται από 4 instances του μοντέλου Operator. Όπως φαίνεται παρακάτω, ένας Operator είναι απλά ένα αντικείμενο με 2 attributes: operatorname (String) και value (Long). Το MeasurementServiceImpl service ανέλαβε το query στην βάση (μέσω του MeasurementDAO) για να κάνει COUNT κάθε Distinct Operator και έπειτα έκανε map κάθε απάντηση σε ένα αντικείμενο τύπου Operator. Στον χρήστη επιστρέφεται τελικά μια συλλογή απ' όλα τα Operator objects που δημιουργήθηκαν.

Αντίστοιχα χρησιμοποιήθηκαν τα μοντέλα OperatorStats (με attributes Operatorname, min, max και avg) και Statistics (με attributes key και value, πχ Operating System και COUNT).

Παράρτημα: Measurement Model, User Model, Operators Model, OperatorStats Model, Statistics Model

Api Documentation

Το endpoint σε ένα REST Api είναι ένα μοναδικό URL το οποίο αντιπροσωπεύει ένα αντικείμενο ή μια συλλογή από αντικείμενα. Ουσιαστικά είναι μια αναφορά σε ένα URI το οποίο δέχεται web requests. Κρίνεται λοιπόν απαραίτητο κάθε Api να έχει ένα documentation, στο οποίο ο δημιουργός του να δίνει σαφείς οδηγίες χρήσης σε κάποιον που θα θέλει να το χρησιμοποιήσει (να το κάνει "consume").

Παρακάτω παραθέτουμε μια λίστα με όλα τα URL του Api μας, την μέθοδο του request, πιθανές παραμέτρους του request και γενικά οποιαδήποτε πληροφορία μπορεί να χρειαστεί κάποιος για να έχει πρόσβαση στα δεδομένα μας.

Login

Returns data about a single user plus JWT Token on Header.

- **URL**

  /login

- **Method:**

  POST

- **Authorization**

  None

- **URL Params**

  None

- **Data Params**

  username=[string],
  password=[string]

- **Success Response:**

  - **Code:** 200
  - **Content:**
  - {
  - "email": "test@hua.gr",
  - "id": 2,
  - "country": null,
  - "password": "1234",
  - "userName": "test"
  - }

- **Error Response:**

  - **Code:** 404 NOT FOUND

  OR

  - **Code:** 401 UNAUTHORIZED

All providers

Returns JSON data about all distinct Providers.

- **URL**

  /api/measurement/allproviders

- **Method:**

  `GET`

- **Authorization**

  None

- **URL Params**

  None

- **Data Params**

  None

- **Success Response:**

  - **Code:** 200
  - **Content:**

```
[
        { "operatorname": "COSMOTE", "value": 294203 },
        { "operatorname": "VODAFONE", "value": 336569 },
        { "operatorname": "WIND", "value": 23604 },
        { "operatorname": "OTHER", "value": 624 }
]
```

- **Error Response:**

  - **Code:** 404 NOT FOUND

  OR

  - **Code:** 401 UNAUTHORIZED

All providers with Network Type

Returns JSON data about all distinct Providers with network type parameter.

- **URL**

  /api/measurement/allproviders/{networkType}

- **Method:**

  `GET`

- **Authorization**

  None

- **URL Params**

Required:

`networkType = [string]`

- **Data Params**

  None

- **Success Response:**

  - **Code:** 200
  - **Content:**

```
[
      { "operatorname": "COSMOTE", "value": 6853 },
      { "operatorname": "VODAFONE", "value": 44239 },
      { "operatorname": "WIND", "value": 1723 }
]
```

- **Error Response:**

  - **Code:** 404 NOT FOUND

  OR

  - **Code:** 401 UNAUTHORIZED

Count

Returns JSON data about the total number of measurements.

- **URL**

  /api/measurement/count

- **Method:**

  `GET`

- **Authorization**

  None

- **URL Params**

  None

- **Data Params**

  None

- **Success Response:**

  - **Code:** 200
  - **Content:**

```
655000
```

- **Error Response:**

- **Code:** 404 NOT FOUND

OR

- **Code:** 401 UNAUTHORIZED

Downlink stats

Returns JSON data about downlink statistics with optional time range and network type.

- **URL**
- /api/measurement/downlinkStats/{r0}/{r1}/{networkType}
- **Method:**

  GET
- **Authorization**

  None
- **URL Params**

  Optional:

  r0, r1 = [string], Date, pattern "yyyy-MM-dd"

  networkType = [string]
- **Data Params**

  None
- **Success Response:**
  - **Code:** 200
  - **Content:**

```
[
    { "operatorname": "COSMOTE", "min": 1, "max": 12036, "avg": 243.77288046185856 },
    { "operatorname": "VODAFONE", "min": 1, "max": 16638, "avg": 99.31412120970634 },
    { "operatorname": "WIND", "min": 1, "max": 3651, "avg": 408.2795449757742 },
    { "operatorname": "OTHER",  "min": 1, "max": 1261,"avg": 54.726190476190474 }
]
```

- **Error Response:**
  - **Code:** 404 NOT FOUND

  OR

  - **Code:** 401 UNAUTHORIZED

Level stats

Returns JSON data about level statistics with optional time range and network type.

- **URL**
- /api/measurement/levelStats/{r0}/{r1}/{networkType}
- **Method:**

  GET
- **Authorization**

  None
- **URL Params**

  Optional:

  r0, r1 = [date], (pattern "yyyy-MM-dd")

  networkType = [string]
- **Data Params**

  None
- **Success Response:**
  - **Code:** 200
  - **Content:**

```
[
    { "operatorname": "COSMOTE", "min": -119, "max": -34,
    "avg": -78.48915368475463 },
    { "operatorname": "VODAFONE", "min": -117, "max": -42,
    "avg": -81.16892646450087 },
    { "operatorname": "WIND", "min": -118, "max": -42, "avg": -
    84.09871208269784 },
    { "operatorname": "OTHER", "min": -101, "max": -51, "avg": -
    86.97639123102867 }
]
```

- **Error Response:**
  - **Code:** 404 NOT FOUND

  OR

  - **Code:** 401 UNAUTHORIZED

Uplink stats

Returns JSON data about uplink statistics with optional time range and network type.

- **URL**
- /api/measurement/uplinkStats/{r0}/{r1}/{networkType}

- **Method:**

  `GET`

- **Authorization**

  None

- **URL Params**

  Optional:

  `r0, r1 = [date], (pattern "yyyy-MM-dd")`

  `networkType = [string]`

- **Data Params**

  None

- **Success Response:**

  - **Code:** 200
  - **Content:**

```
[
    { "operatorname": "COSMOTE", "min": 1, "max": 4407, "avg":
    103.51674020959736 },
    { "operatorname": "VODAFONE", "min": 1, "max": 3023, "avg":
    43.217843295184224 },
    { "operatorname": "WIND", "min": 1, "max": 2306, "avg":
    166.97368421052633 },
    { "operatorname": "OTHER", "min": 1, "max": 107, "avg":
    13.494736842105263 }
]
```

- **Error Response:**

  - **Code:** 404 NOT FOUND

  OR

  - **Code:** 401 UNAUTHORIZED

Network types

Returns JSON data about distinct detwork types.

- **URL**
- /api/measurement/networkTypes/
- **Method:**

  `GET`

- **Authorization**

  None

- **URL Params**

  None

- **Data Params**

  None

- **Success Response:**

  - **Code:** 200
  - **Content:**

```
[
  { "key": "_", "value": 925 },
  { "key": "2G", "value": 52815 },
  { "key": "3G", "value": 601257 }
]
```

- **Error Response:**

  - **Code:** 404 NOT FOUND

  OR

  - **Code:** 401 UNAUTHORIZED

Operating Systems

Returns JSON data about distinct operating systems.

- **URL**
- /api/measurement/operatingSystems/
- **Method:**

  GET

- **Authorization**

  None

- **URL Params**

  None

- **Data Params**

  None

- **Success Response:**

  - **Code:** 200
  - **Content:**

```
[
  { "key": "Unknown", "value": 426982 },
  { "key": "GINGERBREAD", "value": 77228 },
  { "key": "JZO54K", "value": 150790 }
```

```
]
```

- **Error Response:**
  - **Code:** 404 NOT FOUND

  OR

  - **Code:** 401 UNAUTHORIZED

Vendors

Returns JSON data about distinct phone vendors.

- **URL** /api/measurement/vendors/
- **Method:**

  GET
- **Authorization**

  None
- **URL Params**

  None
- **Data Params**

  None
- **Success Response:**
  - **Code:** 200
  - **Content:**

```
[
  { "key": "Unknown", "value": 426982 },
  { "key": "samsung", "value": 77228 },
  { "key": "LGE", "value": 150790 }
]
```

- **Error Response:**
  - **Code:** 404 NOT FOUND

  OR

  - **Code:** 401 UNAUTHORIZED

Angular Frontend

Το βασικό δομικό συστατικό της Angular είναι τα components. Ένα Angular application στην πραγματικότητα αποτελείται από ένα δέντρο από components. Έχουν πάντα ένα template και μόνο ένα component τη φορά μπορεί να γίνεται instantiate σε ένα template. Η βασική αρχή τους είναι ότι το Single responsibility principle, το κάθε component δηλαδή πραγματοποιεί μία μόνο συγκεκριμένη λειτουργία.

Στην πλατφόρμα μας, χωρίσαμε τα components σε 4 κατηγορίες:

- **Pages:** Components που προβάλουν μια σελίδα
- **Services:** Reusable data services, τα οποία μπορούμε κάνουμε inject σε όποιο component θέλουμε.
- **Guards:** Τα Guards προστατεύουν την πρόσβαση σε κάποιο route.
- **Models:** Μοντέλα για objects ή classes τα οποία χρησιμοποιούμε συχνά.

Τα service, guard και model components τα βάλαμε στα αντίστοιχα folders _guards, _services, _models. Τα page components τοποθετήθηκαν το καθένα στο δικό του folder.
Η δομή του project μας:

```
App
├──_guards
│    └── auth.guard.ts
│
├──_models
│    ├── measurements.ts
│    └── user.ts
│
├──_services
│    ├── authentication.service.ts
│    ├── data.service.ts
│    └── navBarService.service.ts
│
├──home
│    ├── home.component.ts
│    ├── home.component.css
│    └── home.component.html
│
├──login
│    ├── login.component.ts
│    └── login.component.html
│
├──maps
│    ├── maps.component.ts
│    ├── maps.component.css
│    └── maps.component.html
├──navbar
│    ├── navbar.component.ts
│    ├── navbar.component.css
│    └── navbar.component.html
│
├──networkStats
│    ├── networkStats.component.ts
│    ├── networkStats.component.css
│    └── networkStats.component.html
│
├──osStats
│    ├── osStats.component.ts
│    ├── osStats.component.css
│    └── osStats.component.html
│
├──providers
│    ├── providers.component.ts
│    ├── providers.component.css
│    └── providers.component.html
│
├──stat-barcharts
│    ├── stat-barcharts.component.ts
│    ├── stat-barcharts.component.css
│    └── stat-barcharts.component.html
```

```
│
├──vendorStats
│    ├── vendorStats.component.ts
│    ├── vendorStats.component.css
│    └── vendorStats.component.html
│
├──config
│
├──e2e
│
├──src
│
├──app.component.ts
├──app.module.ts
├──app.routing.ts
├──main.ts
├──src
├──index.html
├──package.json
├──CHANGELOG.md
├──favicon.ico
├──karma.conf.js
├──LICENCE
├──protractor.config.ts
├──README.md
├──styles.css
├──systemjs.config.extras.js
├── systemjs.config.js
├──tsconfig.json
├──tslint.json
└──webpack.config.js
```

Κάθε component folder περιλαμβάνει ένα typescript αρχείο (με κατάληξη .ts) με τον controller της Angular, ένα html αρχείο με το template του view και ένα προαιρετικό css αρχείο για το συγκεκριμένο template.
Το πως συνδέονται όλα τα κομμάτια του application, το αναλαμβάνει το Angular module class, το αρχείο `app.module.ts` δηλαδή στην εφαρμογή μας. Αυτό κάνει launch το application και περιλαμβάνει οδηγίες για το πως θα γίνει compile και θα τρέξει ο κώδικας. Αναγνωρίζει όλα τα component του module και κάνει import τυχόν εξωτερικές βιβλιοθήκες που θέλουμε να χρησιμοποιήσουμε.
Το Webpack ομαδοποιεί τα αρχεία του project σε bundles, πόρους δηλαδή που ταιριάζουν μαζί και θα πρέπει να επιστρέφονται στον client σε ένα response και ένα αρχείο. Αυτό το καταφέρνει σαρώνοντας την εφαρμογή για import statements, φτιάχνει ένα γράφο με τα dependencies και σχηματίζει ένα ή περισσότερα bundles. Χρησιμοποιεί ακόμα plugins για το preprocess και minify κώδικα που δεν είναι γραμμένος σε Javascript, όπως TypeScript, SASS κα.
Το αρχείο `webpack.config.js` περιλαμβάνει τις ρυθμίσεις του.
Τα σημαντικότερα αρχεία του project αναλύονται παρακάτω:

auth.guard.ts

Guard τύπου `canActivate()`, προστατεύει routes από πρόσβαση εάν ο χρήστης δεν είναι logged in.

measurements.ts

Το μοντέλο Measurements, το οποίο αποτελείται από τα πεδία `id, cellid, networkType, node, latitude, longtitude, level, operatorname, timestamp`. Το χρησιμοποιήσαμε σε περιπτώσεις που δεν μας ενδιέφεραν όλα τα πεδία των μετρήσεων που επέστρεφε η βάση και θέλαμε να τα μετασχηματίσουμε σε ένα interface με κάποια μόνο συγκεκριμένα πεδία.

user.ts

Το μοντέλο του User, με τα στοιχεία που κρατάμε αποθηκευμένα για κάθε χρήστη. Είναι τα `username, password, email, id και προεραιτικό country`.

authentication.service.ts

Το service που αναλαμβάνει το authentication. Περιλαμβάνει την μέθοδο `login(username: string, password: string)`, η οποία επικοινωνεί με το login api endpoint μας και εάν τα στοιχεία είναι αληθή αποθηκεύει το μοναδικό token του χρήστη και το username του στο local storage του browser.

data.service.ts

Το service που επικοινωνεί με το Rest api με ασύγχρονο τρόπο, επιστρέφει δεδομένα και τα κάνει inject στο component που το κάλεσε. Το component αυτό αναλαμβάνει σχεδόν εξ' ολοκλήρου την επικοινωνία με το back end (το μόνο άλλο component είναι το login).

navBarService.ts

Αυτό το service φροντίζει να παρακολουθεί για αλλαγές στο αποθηκευμένο username και οι αλλαγές αυτές να απεικονίζονται στο navigation bar.
Ο λόγος είναι ότι το navigation bar βρίσκεται εκτός routing και εμφανίζεται πάντα στο πάνω μέρος της οθόνης χωρίς να ανανεώνεται όποτε αλλάζει το περιεχόμενο των components. Με το injection του service αυτού το navbar component έχει πάντα την πληροφορία του χρήστη που είναι συνδεδεμένος.

home

Το component που αναλαμβάνει την εμφάνιση του home page μας.

login

Το component που αναλαμβάνει την εμφάνιση του login page μας. Αποτελείται από μια Angular form και περιλαμβάνει έλεγχους validation.

maps

Το component αυτό το χρησιμοποιήσαμε για την δημιουργία heat maps με βάση την τοποθεσία των δεδομένων μας και με βάρος το uplink και το downlink της κάθε μέτρησης. Χρησιμοποιεί την βιβλιοθήκη ngx-google-maps (Google Maps directives για Angular) και παίρνει τα δεδομένα μέσω των μεθόδων getDownlinkPoints(operator:string) και getUplinkPoints(operator:string) του data service το οποίο επικοινωνεί με τον Geoserver μας. Τα δεδομένα έρχονται σε μορφή GEOJson.

navbar

Το component που αναλαμβάνει την εμφάνιση του navigation bar, που εμφανίζεται στο επάνω μέρος της οθόνης.

networkStats

Το component αυτό χρησιμοποιεί την βιβλιοθήκη Chart.js για να σχεδιάσει ένα pie chart με τα διαφορετικά network types (2G, 3G κτλ.) που υπάρχουν στις μετρήσεις.

osStats

Εδώ, χρησιμοποιήθηκε η βιβλιοθήκη Chart.js για το σχεδιασμό ενός doughnut chart με τα διαφορετικά λειτουργικά συστήματα που ήταν εγκατεστημένα στα κινητά που πραγματοποίησαν μετρήσεις.

providers

Στο component αυτό, χρησιμοποιήθηκε το Chart.js για τον σχεδιασμό ενός pie chart με τους διαφορετικούς providers (vodafone, wind κτλ.) στις μετρήσεις μας. Μέσω ενός radio button δίνεται η δυνατότητα να εμφανιστούν οι μετρήσεις για συγκεκριμένο Network type.

stat-barcharts

Το component αυτό αναλαμβάνει χρησιμοποιώντας την βιβλιοθήκη Chart.js, να φτιάξει bar charts, με το minimum, average και maximum των μετρήσεων uplink, downlink και level. Η επιλογή της μέτρησης γίνεται μέσω μιας Angular form και δίνει ακόμα τις επιλογές για συγκεκριμένο Network Type και χρονικό διάστημα.

vendorStats

Το τελευταίο component , χρησιμοποιεί το Chart.js για την δημιουργία bar charts με τους διαφορετικούς κατασκευαστές κινητών συσκευών που λάβανε μέρος στις μετρήσεις.

package.json

Το package.json είναι το αρχείο που διαχειρίζεται όλα τα third-party packages που χρειάζονται τα Angular applications.

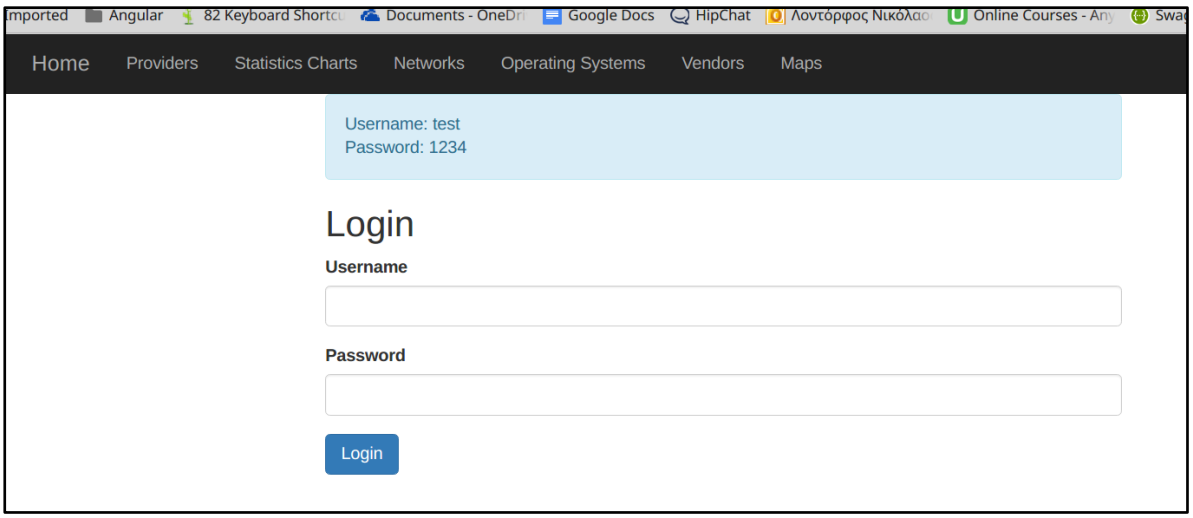
Τα χωρίζει σε 2 κατηγορίες, dependencies και devDependencies. Τα πρώτα χρειάζονται για να τρέξει η εφαρμογή και τα δεύτερα χρειάζονται μόνο στο development. Εγκαθίστανται με `npm install` εφόσον είναι εγκατεστημένο το Node.js.

Ο κώδικας της εφαρμογής μπορεί να βρεθεί στο repository: https://github.com/NickLont/Data-visualization-project

4.2 Οδηγός χρήσης της εφαρμογής

4.2.1 Σύνδεση (Log In Page)

Η πρώτη εικόνα που βλέπει ο χρήστης κατά την σύνδεση του είναι η σελίδα εισόδου όπως φαίνεται στην εικόνα. Εάν ο χρήστης είναι εγγεγραμμένος, εισάγει το όνομα και τον κωδικό του για να συνδεθεί στην εφαρμογή. Την εγγραφή νέου χρήστη μπορεί να την κάνει μόνο ένας διαχειριστής μέσα από την εφαρμογή.

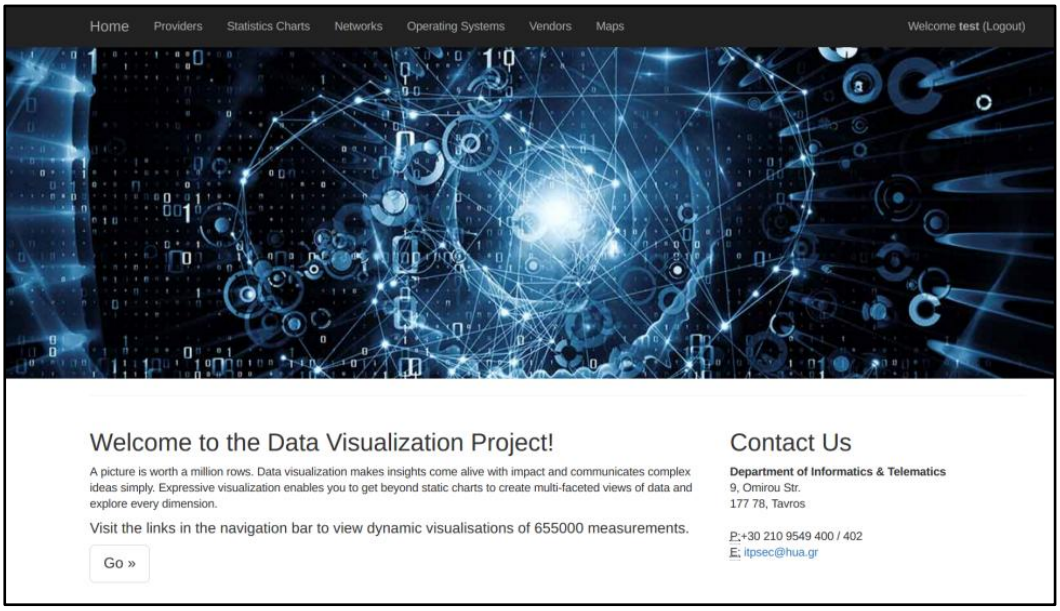


Εικόνα 26 - Login Page

Να σημειωθεί ότι εάν ο χρήστης δεν κάνει log in, δεν του επιτρέπεται η πρόσβαση σε κανένα link του navigation bar.

4.2.2 Κεντρική σελίδα (Home Page)

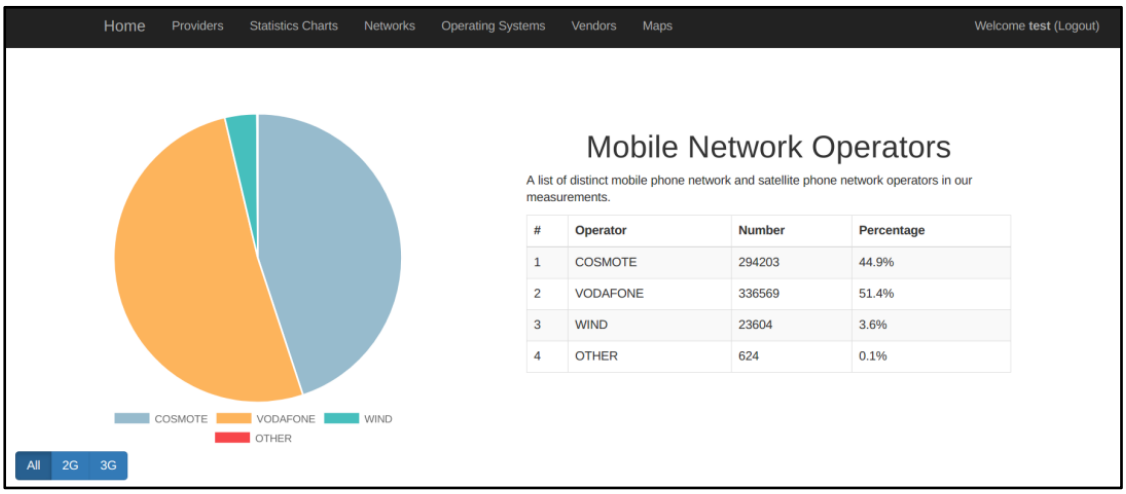
Η κεντρική σελίδα της εφαρμογής μας. Περιέχει ένα jumbotron με εικόνα, λίγα λόγια για το Data Visualization γενικά και στοιχεία επικοινωνίας με το Χαροκόπειο Πανεπιστήμιο. Στόχος της είναι να οδηγήσει τον χρήστη στα διαφορετικά visualizations που προσφέρουμε στην πλατφόρμα μας.



Εικόνα 27 - Home Page

4.2.3 Providers

Στην σελίδα αυτή παρέχει στοιχεία για τους διαφορετικούς Operators (Vodafone, Wind κτλ) που λαμβάνουν μέρος στις μετρήσεις. Απεικονίζονται με την μορφή pie chart αλλά και πίνακα ο οποίος περιέχει αναλυτικά τα στατιστικά.

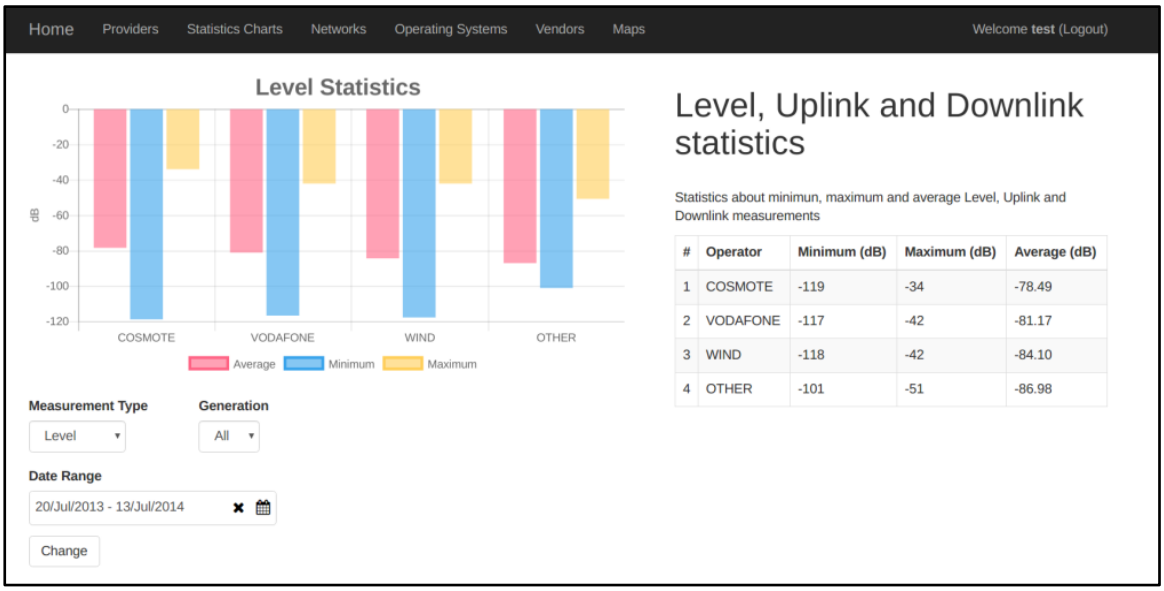


Εικόνα 28 - Providers Page

Το radio selector κάτω από το Pie chart δίνει την δυνατότητα επιλογής συγκεκριμένης τεχνολογίας δικτύου (2G, 3G κτλ.). Στην αλλαγή επιλογής, το pie chart και ο πίνακας επανασχεδιάζονται με τα καινούργια δεδομένα.

4.2.4 Statistics Charts

Στην σελίδα αυτή παρουσιάζονται στατιστικά για το Uplink, το Downlink και το Level (την ισχύ του σήματος), για κάθε μοναδικό provider στις μετρήσεις.

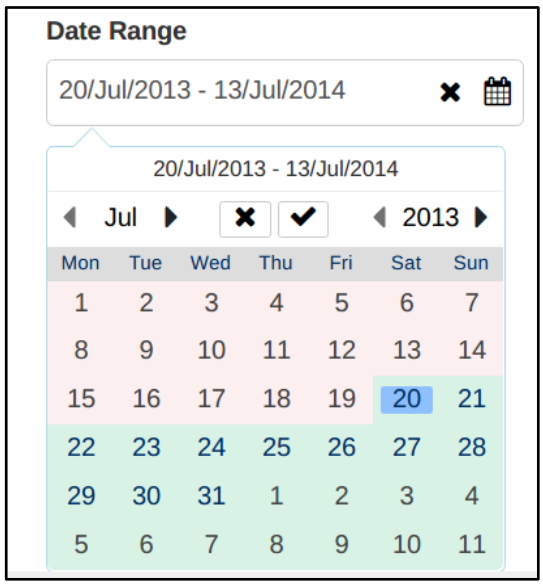


Εικόνα 29 - Statistics Page

Τα στατιστικά αυτά εμφανίζονται με την μορφή bar chart, όπου παρουσιάζεται το minimum, το average και το maximum για κάθε provider αλλά και αναλυτικά σε μορφή πίνακα στο δεξί μέρος της οθόνης.

Κάτω από το bar chart ο χρήστης μπορεί να ρυθμίσει custom παραμέτρους για τα στατιστικά που εμφανίζονται, όπως ποιά μέτρηση θέλει να εμφανίσει (level, uplink ή downlink), την γενιά του network technology (2G, 3G κτλ) και τον περιορισμό των αποτελεσμάτων σε ένα συγκεκριμένο χρονικό διάστημα.

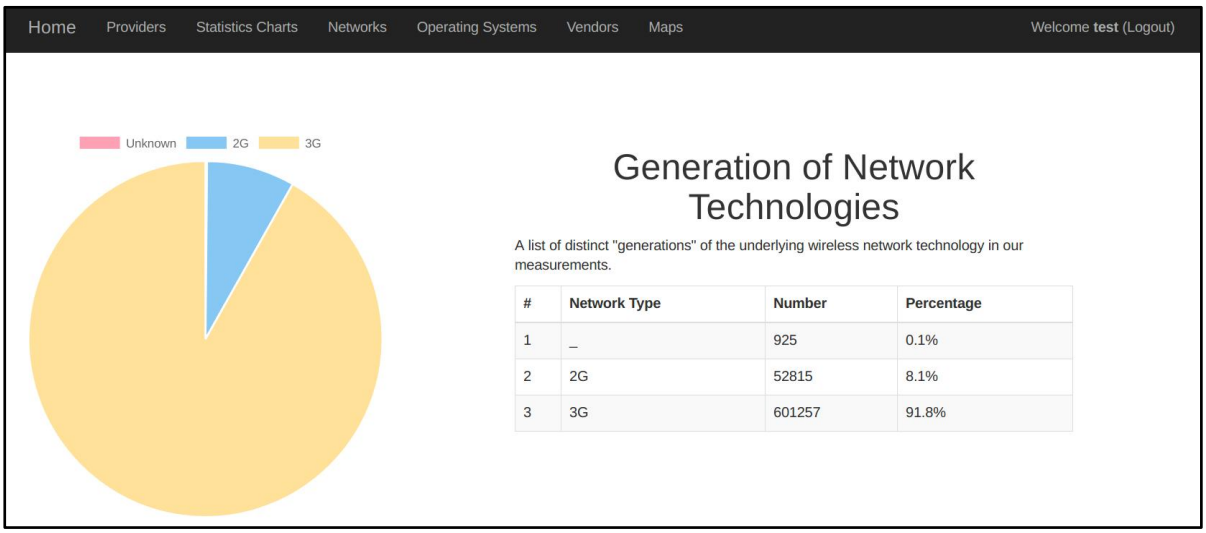
Η χρονική περίοδος ορίζεται από ένα date range picker όπως φαίνεται στην παρακάτω εικόνα. Να σημειωθεί ότι τα όρια του λαμβάνονται από την βάση δεδομένων, οπότε ο χρήστης δεν μπορεί να ορίσει ημερομηνία παλιότερη από την πιο παλιά μέτρηση.



Εικόνα 30 - Date picker

4.2.5 Networks

Η σελίδα αυτή εμφανίζει στοιχεία για τις γενιές δικτύου που εμφανίζονται στις μετρήσεις και στατιστικά που τις αφορούν.

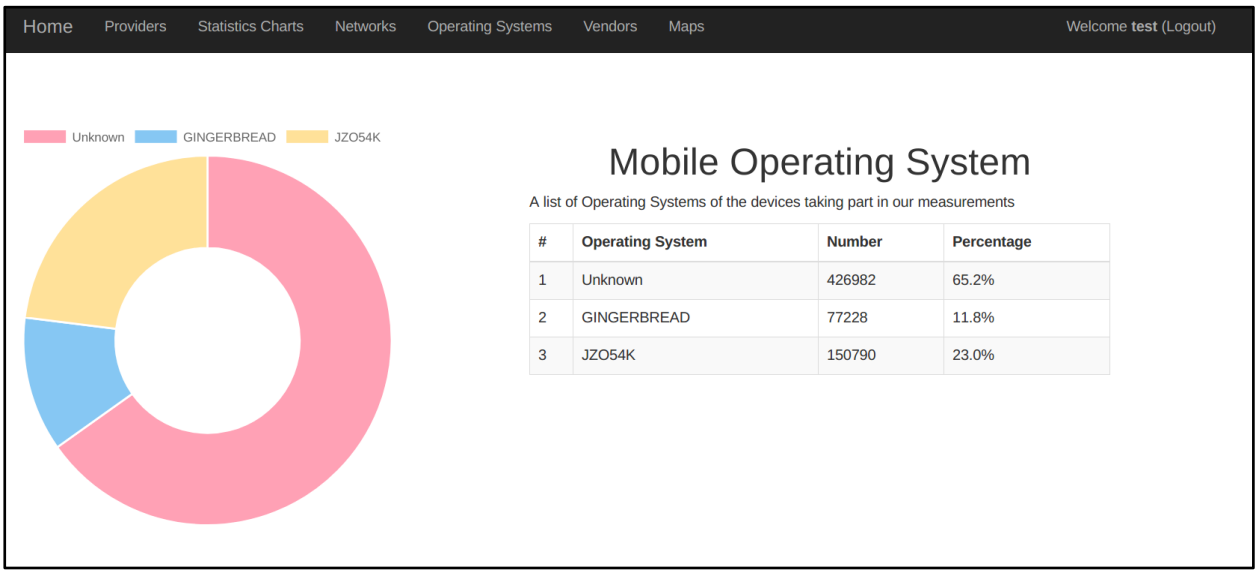


Εικόνα 31 - Networks Page

Η παρουσίαση τους γίνεται με την μορφή pie chart και αναλυτικού πίνακα.

4.2.6 Operating Systems

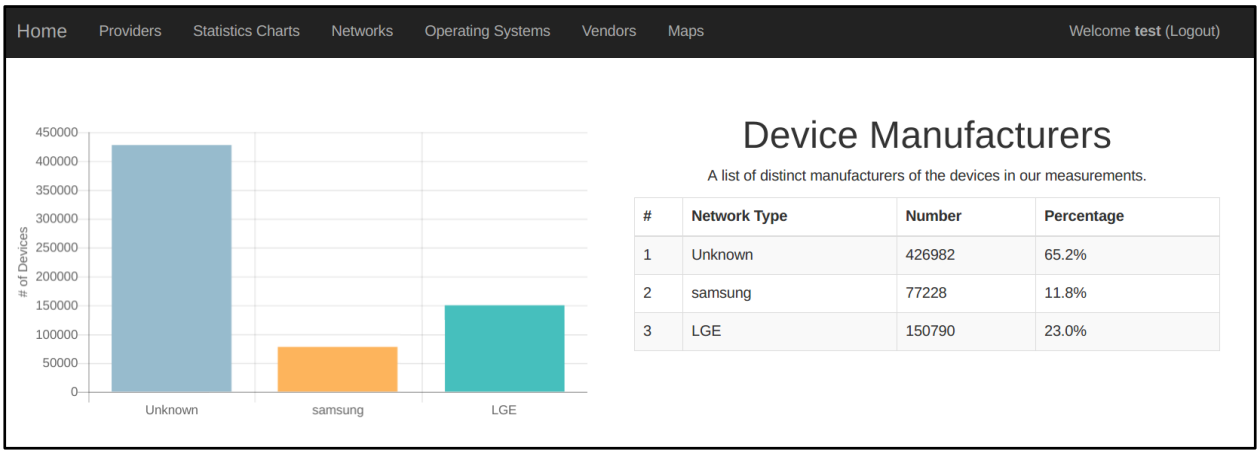
Εδώ παρουσιάζονται στοιχεία για τα λειτουργικά συστήματα που βρίσκονταν εγκατεστημένα στα κινητά που λάβανε μέρος στις μετρήσεις μας. Για την απεικόνιση τους χρησιμοποιήθηκε doughnut chart και αναλυτικός πίνακας.



Εικόνα 32 - Os's Page

4.2.7 Vendors

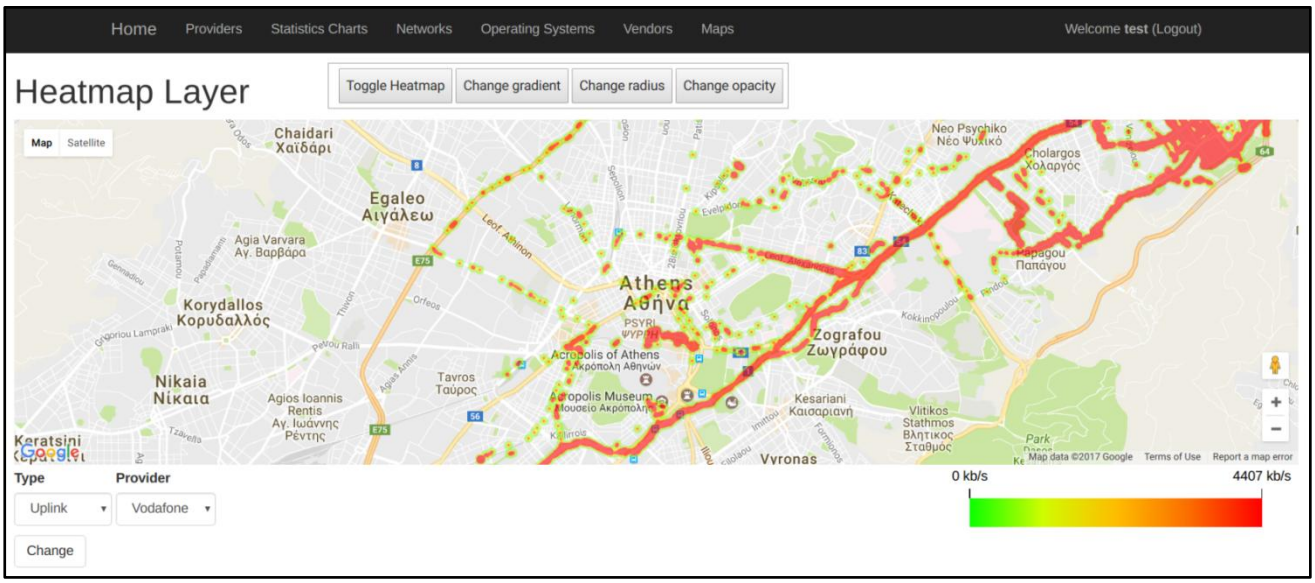
Σε αυτή την σελίδα βλέπουμε στατιστικά στοιχεία για τους κατασκευαστές των συσκευών των κινητών που λάβανε τις μετρήσεις. Εμφανίζονται με την μορφή bar chart και πίνακα.



Εικόνα 33 - Vendors Page

4.2.8 Maps

Η σελίδα αυτή, χρησιμοποιεί το Google Maps Api, με σκοπό την δημιουργία ενός Heat map από τα γεωχωρικά δεδομένα μας. Το heat map είναι μια γραφική αναπαράσταση των δεδομένων, όπου κάθε ξεχωριστή τιμή μέτρησης απεικονίζεται σαν ένα χρώμα. Για την ακρίβεια αποτελεί την οπτικοποίηση των αποτελεσμάτων queries σε ένα ορισμένο πολύγωνο, το οποίο αποτελείται από μικρότερα χρωματισμένα πολύγωνα. Βοηθάει στην καλύτερη κατανόηση μεγάλων datasets καθώς η χρωματική ένταση κάθε σχήματος καθώς και το μέγεθος του κάνουν εμφανή τα σημεία ενδιαφέροντος με μια ματιά. Η ένταση του χρώματος σε κάθε σημείο εξαρτάται από την πυκνότητα των μετρήσεων στο σημείο αυτό καθώς και το βάρος κάθε μέτρησης, το οποίο εξαρτάται από την τιμή της.



Εικόνα 34 - Maps Page

Στο κάτω δεξιά μέρος της οθόνης βλέπουμε το gradient legend που δηλώνει την αντιστοίχηση των τιμών των μετρήσεων σε σχέση με το χρώμα που απεικονίζεται στον χάρτη.

Κάτω αριστερά δίνονται επιλογές σχετικές με το ποιά σημεία θέλουμε να εμφανίσουμε στον χάρτη. Ο χρήστης μπορεί να επιλέξει να απεικονίσει δεδομένα σχετικά με το uplink ή το downlink καθώς και να επιλέξει συγκεκριμένο provider. Στο επάνω μέρος της οθόνης δίνονται ακόμα περισσότερες επιλογές για την εμφάνιση του heat map, όπως το toggle του (ενφάνιση και εξαφάνιση), η αλλαγή του gradient (αναστροφή των χρωμάτων), αύξηση της ακτίνας επιροής κάθε σημείου και μείωση της έντασης του opacity του χάρτη.

5 Συμπεράσματα

Απολογισμός και προβλήματα

Η ολοκλήρωση της παρούσας διπλωματικής μας βοήθησε στο να καταλήξουμε σε κάποια σημαντικά συμπεράσματα αλλά και να έρθουμε αντιμέτωποι με προβλήματα τα οποία δεν είχαμε προβλέψει πριν την δημιουργία της διαδικτυακής πλατφόρμας.Όσον αφορά τα visualizations, είδαμε έμπρακτα ότι η χρήση γραφημάτων και charts για την οπτικοποίηση μεγάλου όγκου δεδομένων βοηθά τον ανθρώπινο εγκέφαλο στην καλύτερη κατανόηση τους, σε σχέση με πολύ μεγάλα spreadsheets και reports. Η δυνατότητα αυτή, το να μπορούμε να δούμε πολλά δεδομένα δηλαδή με καθαρό και επεξηγηματικό τρόπο, οδηγεί στην γρηγορότερη ανάλυση της πληροφορίας και κατά συνέπεια στην απάντηση ερωτημάτων και την επίλυση προβλημάτων. Τα visualizations βοηθάνε ακόμη στην ανεύρεση σχέσεων σε παραμέτρους των δεδομένων οι οποίες δεν είναι εμφανείς σε πρώτο στάδιο μέχρι να απεικονιστούν γραφικά. Η ανεύρεση τέτοιων σχέσεων μπορεί να βοηθήσει σημαντικά έναν οργανισμό να εστιάσει σε τομείς οι οποίοι πιθανότατα θα επηρεάσουν τους άμεσους στόχους του. Τέλος, εφόσον πραγματοποιηθεί η εξαγωγή της γνώσης χρησιμοποιώντας visual analytics, είναι εύκολο αυτή να μεταφερθεί και να εξηγηθεί χρησιμοποιώντας charts, γραφήματα και άλλους τρόπους παρουσίασης οι οποίοι είναι ενδιαφέροντες και μπορούν εύκολα να επεξηγήσουν ένα συμπέρασμα.
Παρόλο όμως που data visualizations δημιουργήθηκαν για να βοηθήσουν τους αναλυτές να προβλέπουν τάσεις και να καταλαβαίνουν καλυτέρα τα δεδομένα, υπάρχουν και αρκετοί περιορισμοί οι οποίοι όσο μεγαλώνει ο όγκος των δεδομένων, τόσο σημαντικότεροι γίνονται. Ένας από αυτούς αποτελεί το γεγονός ότι τα visualizations, παρόλο που μπορούν να παραχθούν real-time, δεν παρέχουν κάποια επεξήγηση για την πραγματική σημασία τους. Αυτή η ανάλυση γίνεται από εξειδικευμένους αναλυτές και κοστίζει σε χρόνο και άρα σε χρήμα. Ακόμα και για έμπειρους χρήστες στην διαχείριση τέτοιων δεδομένων όμως υπάρχει πάντα ο κίνδυνος δύο διαφορετικοί χρήστες να εξάγουν διαφορετικά συμπεράσματα από τα ίδια visualizations, ανάλογα με την εμπειρία και την εξοικείωση τους καθενός. Αυτό μπορεί να οδηγήσει σε λάθος συμπεράσματα και εν' τέλει να θέσει σε κίνδυνο μια επιχείρηση η οποία θα στηριχτεί σε αυτά. Τα γραφήματα είναι πολύ χρήσιμα για την επεξήγηση μιας απλής ιδέας γρήγορα. Για να αναλυθεί όμως μια πολύπλοκη κατάσταση χρειάζονται λέξεις και προτάσεις, χρειάζεται ένα σύστημα δηλαδή το οποίο θα μπορεί να εξηγεί με λέξεις την διαδικασία με την οποία κατέληξε στο συμπέρασμα. Τα γραφικά μπορεί να οδηγήσουν τον χρήστη στο συμπέρασμα ότι παίρνει αποφάσεις στηριζόμενος σε δεδομένα ή ότι κατανοεί πλήρως τα στοιχεία που έχει, ενώ στην πραγματικότητα βλέπει μόνο μια εικόνα χωρίς να ξέρει όλη την πραγματικότητα.

Στο κομμάτι του development της πλατφόρμας μας τώρα, η αρχιτεκτονική που αποφασίσαμε να χρησιμοποιήσουμε είναι ενός client με δυναμική συμπεριφορά γραμμένος σε Angular 2, ο

οποίος παίρνει τα δεδομένα που χρειάζεται από ένα RESTful Api γραμμένο σε Java Spring. Το pattern αυτό μας έδωσε έναν ξεκάθαρο διαχωρισμό μεταξύ του front-end και του back-end. Το πρόβλημα προέκυψε όταν πήγαμε να εφαρμόσουμε την πιο απλή υλοποίηση του pattern αυτού, να "σερβίρουμε" όλες τις μετρήσεις δηλαδή από τον server στο front-end και να αφήσουμε την Angular να επεξεργαστεί τα δεδομένα. Ο όγκος των μετρήσεων (πάνω από 600.000) μετρήσεις κατέστησε απαγορευτική αυτή την προσέγγιση αφού η επιστροφή τόσου μεγάλου μεγέθους δεδομένων, όταν δεν αποτύγχανε λόγω time out, ήτανε φοβερά χρονοβόρα. Αυτό μας οδήγησε στην λύση του να αποδομήσουμε την πληροφορία που χρειαζόμασταν σε επί μέρους μικρότερα calls και το βάρος της επεξεργασίας, του μετασχηματισμού και της μορφοποίησης των δεδομένων να τα αναθέσουμε στην βάση δεδομένων. Το Spring έπειτα, έκανε wrap τα δεδομένα σε ένα JSON αρχείο, με μέγεθος συνήθως κάποια kilobyte και τα σέρβιρε στον client για επεξεργασία. Χρησιμοποιώντας τις δυνατότητες reactive programming (Reactive programming) της Angular, φτιάξαμε μια πλατφόρμα η οποία "αντιδρά" άμεσα, όχι μόνο σε inputs από τους χρήστες αλλά και σε αλλαγές του περιβάλλοντος, όπως πχ μια αλλαγή στα δεδομένα της βάσης.
Ένα σημαντικό πρόβλημα που αντιμετωπίσαμε στην ανάπτυξη της Διαδικτυακής μας Πλατφόρμας είναι ότι, η Angular 2 ήταν μια πολύ φρέσκια τεχνολογία στα front-end frameworks (η final έκδοση της έγινε release στα μέσα Σεπτεμβρίου του 2016). Όπως είδαμε πιο πριν, η Angular 2 χρησιμοποιεί TypeScript σε αντίθεση με την AngularJS που χρησιμοποιούσε JavaScript. Αυτό δημιούργησε πρόβλημα με τις δεκάδες βιβλιοθήκες που ήταν ήδη γραμμένες σε JavaScript και χρησιμοποιούνταν απ' όλα τα JS front-end frameworks. Οι πιο δημοφιλείς από αυτές εκδώσανε definition files τα οποία προσέθεσαν υποστήριξη για Typescript ή φτιαχτήκανε από χρήστες. Όταν κατά την διάρκεια της δημιουργίας της πλατφόρμας θελήσαμε να χρησιμοποιήσουμε την βιβλιοθήκη OpenLayers 3 (Open Layers) ανακαλύψαμε ότι δεν είχε βγει ακόμα επίσημο definition file παρά μόνο ένα φτιαγμένο από χρήστες. Δυστυχώς το documentation του ήταν αρκετά ελλιπές για τις εξεζητημένες διεργασίες που θέλαμε να πραγματοποιήσουμε όπως εισαγωγή σημείων από GEOJson και εφαρμογή transformations επάνω σε αυτά πριν την απεικόνιση τους. Αναγκαστήκαμε λοιπόν να χρησιμοποιήσουμε Google Maps, η οποία ήταν η μόνη βιβλιοθήκη εκείνη την περίοδο με επίσημα definition files για Typescript αλλά γενικά πιο περιορισμένες δυνατότητες από το OpenLayers. Το γεγονός αυτό μας οδήγησε στην χρήση αρκετών homebrew λύσεων, όπως το να χρησιμοποιήσουμε Heatmap για την εμφάνιση των σημείων στο χάρτη, αντί για κάποιον custom αλγόριθμο (Barnes Surface), που δεν είναι ιδανικό όταν θέλουμε να εστιάσουμε περισσότερο στο weight κάθε σημείου και όχι στο density σημείων σε περιοχές του χάρτη. Άλλο παράδειγμα είναι ότι το Google Maps δεν υποστήριζε το Geography data type Point που επέστρεφε ο Geoserver (μορφή Point (Lat, Long, SRID)), οπότε έπρεπε να το χωρίζουμε χειροκίνητα σε lat και lon, χάνοντας έτσι τελείως το νόημα του Geography.

Μελλοντικές επεκτάσεις

Οι επεκτάσεις που μπορούν να γίνουν στην εφαρμογή αφορούν κυρίως τεχνολογικούς περιορισμούς.
Η βασικότερη επέκταση που θα μπορούσε να γίνει το μέλλον αφορά την μεταφορά των Map Visualization από το Google Maps στο Open Layers 3, εφ' όσον αυτό υποστηριχθεί επίσημα. Η

υποδομή στο server side δεν χρειάζεται να αλλάξει καθόλου αφού είναι τελείως ξεχωριστό από το client side και το GEOJson που επιστρέφει ο GEOServer είναι ιδανικό για επεξεργασία στο front end.Η πληθώρα transformations που παρέχει το Open Layers θα βοηθούσε στην δημιουργία ακόμα περισσότερων map visualizations που θα βοηθούσαν στην καλύτερη κατανόηση της πληροφορίας.
Η δεύτερη επέκταση αφορά την ίδια την κατανόηση των visualizations και το πως απεικονίζουν αλλά δεν εξηγούν αυτό που παρουσιάζουν. Για την ακρίβεια η κατανόηση ενός visualization απαιτεί από τον χρήστη καλή γνώση των ίδιων των δεδομένων. Η λύση σε αυτό είναι η υιοθέτηση Natural Language Generation(NLG) λύσεων, οι οποίες περιγράφουν σε φυσική γλώσσα τα συμπεράσματα και της αλλαγές στα δεδομένα. Πολλές εταιρίες έχουν αρχίσει ήδη να μετατρέπουν τα δεδομένα τους σε συμπεράσματα φυσικής γλώσσας, χρησιμοποιώντας τεχνικές NLG και Artificial Intelligence αλλά η εφαρμογή τέτοιων τεχνικών στο web development είναι σε πολύ πρώιμο στάδιο ακόμα.

Παράρτημα

Κώδικας

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>HuaTester</groupId>
	<artifactId>HuaTester</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>war</packaging>
	<name>HuaTester</name>

	<properties>
		<java-version>1.8</java-version>
		<eclipselink.version>2.5.0</eclipselink.version>
		<org.springframework-version>4.3.3.RELEASE</org.springframework-version>
		<org.springframework.security-version>4.1.3.RELEASE</org.springframework.security-
version>
		<org.aspectj-version>1.8.1</org.aspectj-version>
		<org.slf4j-version>1.7.12</org.slf4j-version>
		<hibernate.version>5.2.4.Final</hibernate.version>
	</properties>

	<dependencies>
		<dependency>
			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<version>4.5</version>
			<scope>test</scope>
		</dependency>

		<dependency>
			<groupId>org.eclipse.persistence</groupId>
			<artifactId>eclipselink</artifactId>
			<version>${eclipselink.version}</version>
			<scope>compile</scope>
		</dependency>

		<dependency>
			<groupId>org.eclipse.persistence</groupId>
			<artifactId>javax.persistence</artifactId>
			<version>2.0.0</version>
			<scope>provided</scope>
<!-- since I'm running inside a Java EE container -->
		</dependency>

		<dependency>
			<groupId>javax.servlet</groupId>
			<artifactId>javax.servlet-api</artifactId>
			<version>3.1.0</version>
			<scope>provided</scope>
		</dependency>

		<dependency>
			<groupId>javax.servlet.jsp</groupId>
			<artifactId>javax.servlet.jsp-api</artifactId>
			<version>2.3.1</version>
			<scope>provided</scope>
		</dependency>
```

```
		<dependency>
			<groupId>javax.servlet.jsp.jstl</groupId>
			<artifactId>javax.servlet.jsp.jstl-api</artifactId>
			<version>1.2.1</version>
			<scope>compile</scope>
		</dependency>

		<!-- Spring -->
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-context</artifactId>
			<version>${org.springframework-version}</version>
			<exclusions>
<!-- Exclude Commons Logging in favor of SLF4j -->
				<exclusion>
					<groupId>commons-logging</groupId>
					<artifactId>commons-logging</artifactId>
				</exclusion>
			</exclusions>
		</dependency>

		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-core</artifactId>
			<version>${org.springframework-version}</version>
		</dependency>

		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-webmvc</artifactId>
			<version>${org.springframework-version}</version>
		</dependency>

		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-jdbc</artifactId>
			<version>${org.springframework-version}</version>
		</dependency>

		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-tx</artifactId>
			<version>${org.springframework-version}</version>
		</dependency>

		<!-- Spring ORM -->
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-orm</artifactId>
			<version>${org.springframework-version}</version>
		</dependency>

		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-
	entitymanager</artifactId>
			<version>${hibernate.version}</version>
		</dependency>

		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-spatial</artifactId>
			<version>${hibernate.version}</version>
```

```
</dependency>

<!-- Apache Commons DBCP -->
<dependency>
        <groupId>commons-dbcp</groupId>
        <artifactId>commons-dbcp</artifactId>
        <version>1.4</version>
</dependency>

<!-- AspectJ -->
<dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjrt</artifactId>
        <version>${org.aspectj-version}</version>
</dependency>

<!-- Logging -->
<dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>${org.slf4j-version}</version>
</dependency>

<dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>jcl-over-slf4j</artifactId>
        <version>${org.slf4j-version}</version>
        <scope>runtime</scope>
</dependency>

<dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>${org.slf4j-version}</version>
        <scope>runtime</scope>
</dependency>

<dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.16</version>
        <scope>runtime</scope>
</dependency>

<!-- Jackson JSON Procesor -->
<dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.5.3</version>
</dependency>

<!-- PostgreSQL -->
<dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <version>9.4.1212</version>
</dependency>

<dependency>
        <groupId>net.postgis</groupId>
        <artifactId>postgis-jdbc</artifactId>
        <version>2.2.1</version>
</dependency>
```

```
        <!-- Spring Security -->
        <dependency>
                <groupId>org.springframework.security</groupId>
                <artifactId>spring-security-core</artifactId>
                <version>${org.springframework.security-  version}</version>
        </dependency>

        <dependency>
                <groupId>org.springframework.security</groupId>
                <artifactId>spring-security-web</artifactId>
                <version>${org.springframework.security-
version}</version>
        </dependency>

        <dependency>
                <groupId>org.springframework.security</groupId>
                <artifactId>spring-security-config</artifactId>
                <version>${org.springframework.security-  version}</version>
        </dependency>

        <dependency>
                <groupId>jstl</groupId>
                <artifactId>jstl</artifactId>
                <version>1.2</version>
        </dependency>

        <!-- BeanValidation and Hibernate Validator. -->
        <dependency>
                <groupId>javax.validation</groupId>
                <artifactId>validation-api</artifactId>
                <version>1.1.0.Final</version>
        </dependency>

        <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-validator</artifactId>
                <version>${hibernate.version}</version>
        </dependency>

        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot</artifactId>
                <version>1.4.2.RELEASE</version>
        </dependency>

        <dependency>
                <groupId>javax.ws.rs</groupId>
                <artifactId>javax.ws.rs-api</artifactId>
                <version>2.1-m01</version>
        </dependency>

        <dependency>
                <groupId>javax.ejb</groupId>
                <artifactId>javax.ejb-api</artifactId>
                <version>3.2</version>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-devtools</artifactId>
                <version>1.4.2.RELEASE</version>
        </dependency>
        <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
            <version>1.4.2.RELEASE</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <version>1.4.2.RELEASE</version>
            <scope>provided</scope>
        </dependency>

        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.5.0</version>
        </dependency>

        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.5.0</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.data</groupId>
            <artifactId>spring-data-jpa</artifactId>
            <version>1.10.5.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt</artifactId>
            <version>0.7.0</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>6.0.5</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.5.1</version>
                <configuration>
                    <source>${java-version}</source>
                    <target>${java-version}</target>
                </configuration>
            </plugin>
            <plugin>
                <artifactId>maven-war-plugin</artifactId>
                <version>2.6</version>
                <configuration>
                    <!--<failOnMisingWebXml>false</failOnMisingWebXml>-->
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-resources-plugin</artifactId>
                <version>3.0.1</version>
                <configuration>
```

```
                <resources>
                    <resource>
                        <directory>src/main/resources/META-
                        INF</directory>
                        <!--src/test/resources/META-INF-->
                        <includes>
                            <include>persistence.xml</include>
                        </includes>
                    </resource>
                </resources>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <version>2.2</version>
            <configuration>
                <server>tomcat</server>
                <url>http://test.hua.gr:8080/manager/text</url>
            </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

MeasurementRestController

```
package gr.huatester;

import java.security.Timestamp;
import java.util.List;

import gr.huatester.model.OperatorStats;
import io.swagger.annotations.Api;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.util.UriComponentsBuilder;

import gr.huatester.model.Measurement;
import gr.huatester.model.Operators;
import gr.huatester.model.Statistics;

import gr.huatester.service.MeasurementService;
import io.swagger.annotations.ApiOperation;
```

```
@RestController
@RequestMapping(value = "/api/")
@Api(value = "measurementsapi", description = "Operations pertaining to measurements in
Measurements DB")
public class MeasurementRestController {

   @Autowired
   MeasurementService measurementService;
//Service which will do all data retrieval/manipulation work

         private static final Logger logger =
LoggerFactory.getLogger(MeasurementRestController.class);


   //------------------- Retrieve All Measurements -------------------------------------------------
   // @RequestMapping(value = "/measurement/", method = RequestMethod.GET)

   @RequestMapping(value = "/measurement/allproviders", method = RequestMethod.GET,
produces = MediaType.APPLICATION_JSON_VALUE)
   public ResponseEntity<List<Operators>> findAll() {
                  List<Operators> measurements = measurementService.findAll();
      if(measurements.isEmpty()){
         return new ResponseEntity<List<Operators>>(HttpStatus.NO_CONTENT);//You many
decide to return HttpStatus.NOT_FOUND
      }
      return new ResponseEntity<List<Operators>>(measurements, HttpStatus.OK);
   }

   //--------------Get Operators with network parameter-----------------------
   @RequestMapping(value = "/measurement/allproviders/{networkType}", method =
RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
   public ResponseEntity<List<Operators>>
findAllForNetwType(@PathVariable("networkType") String networkType) {
                  List<Operators> measurements = measurementService.findAllForNetwType(new
String (networkType));
      if(measurements.isEmpty()){
         return new ResponseEntity<List<Operators>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
      }
      return new ResponseEntity<List<Operators>>(measurements, HttpStatus.OK);
   }

   //--------------Get Statistics about Level for all distinct Providers--------------------------

   @RequestMapping(value = "/measurement/levelStats", method = RequestMethod.GET,
produces = MediaType.APPLICATION_JSON_VALUE)
   public ResponseEntity<List<OperatorStats>> findOperatorLevelStats() {
                  List<OperatorStats> measurements =
measurementService.findOperatorLevelStats();
      if(measurements.isEmpty()){
```

```
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }

    //--------------Get Statistics about Level for all distinct Providers with Network Type
parameter------------------

    @RequestMapping(value = "/measurement/levelStats/{networkType}", method =
RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<OperatorStats>>
findOperatorLevelStatsForNetwType(@PathVariable("networkType") String networkType) {
                    List<OperatorStats> measurements =
measurementService.findOperatorLevelStatsForNetwType(new String (networkType));
        if(measurements.isEmpty()){
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }


    //--------------Get Statistics about Level for all distinct Providers with time range-----------------
---------


    @RequestMapping(value = "/measurement/levelStats/{r0}/{r1}", method =
RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<OperatorStats>>
findOperatorLevelStatsWithParams(@PathVariable("r0") String r0, @PathVariable("r1") String
r1) {
                    List<OperatorStats> measurements =
measurementService.findOperatorLevelStatsWithParams(new String[] {r0, r1});
        if(measurements.isEmpty()){
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }

    //--------------Get Statistics about Level for all distinct Providers with time range and Network
Type parameter -------


    @RequestMapping(value = "/measurement/levelStats/{r0}/{r1}/{networkType}", method =
RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<OperatorStats>>
findOperatorLevelStatsWithParamsForNetwType(@PathVariable("r0") String r0,
@PathVariable("r1") String r1, @PathVariable("networkType") String networkType) {
```

```
        List<OperatorStats> measurements =
measurementService.findOperatorLevelStatsWithParamsForNetwType(new String[] {r0, r1,
networkType});
        if(measurements.isEmpty()){
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }

    //--------------Get Statistics about Uplink for all distinct Providers-------------------------

    @RequestMapping(value = "/measurement/uplinkStats", method = RequestMethod.GET,
produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<OperatorStats>> findOperatorUplinkStats() {
                    List<OperatorStats> measurements =
measurementService.findOperatorUplinkStats();
        if(measurements.isEmpty()){
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }

    //--------------Get Statistics about Uplink for all distinct Providers with Network Type
parameter------------------

    @RequestMapping(value = "/measurement/uplinkStats/{networkType}", method =
RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<OperatorStats>>
findOperatorUplinkStatsForNetwType(@PathVariable("networkType") String networkType) {
                    List<OperatorStats> measurements =
measurementService.findOperatorUplinkStatsForNetwType(new String (networkType));
        if(measurements.isEmpty()){
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }


    //--------------Get Statistics about Uplink for all distinct Providers with time range---------------
-----------


    @RequestMapping(value = "/measurement/uplinkStats/{r0}/{r1}", method =
RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<OperatorStats>>
findOperatorUplinkStatsWithParams(@PathVariable("r0") String r0, @PathVariable("r1")
String r1) {
                    List<OperatorStats> measurements =
measurementService.findOperatorUplinkStatsWithParams(new String[] {r0, r1});
```

```
        if(measurements.isEmpty()){
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }

    //--------------Get Statistics about Uplink for all distinct Providers with time range and
Network Type parameter -------


    @RequestMapping(value = "/measurement/uplinkStats/{r0}/{r1}/{networkType}", method =
RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<OperatorStats>>
findOperatorUplinkStatsWithParamsForNetwType(@PathVariable("r0") String r0,
@PathVariable("r1") String r1, @PathVariable("networkType") String networkType) {
        List<OperatorStats> measurements =
measurementService.findOperatorUplinkStatsWithParamsForNetwType(new String[] {r0, r1,
networkType});
        if(measurements.isEmpty()){
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }

    //--------------Get Statistics about Downlink for all distinct Providers--------------------------

    @RequestMapping(value = "/measurement/downlinkStats", method = RequestMethod.GET,
produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<OperatorStats>> findOperatorDownlinkStats() {
                    List<OperatorStats> measurements =
measurementService.findOperatorDownlinkStats();
        if(measurements.isEmpty()){
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);//You
many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }

    //--------------Get Statistics about Downlink for all distinct Providers with Network Type
parameter------------------

    @RequestMapping(value = "/measurement/downlinkStats/{networkType}", method =
RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<OperatorStats>>
findOperatorDownlinkStatsForNetwType(@PathVariable("networkType") String networkType)
{
                    List<OperatorStats> measurements =
measurementService.findOperatorDownlinkStatsForNetwType(new String (networkType));
        if(measurements.isEmpty()){
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);
```

```
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }


    //--------------Get Statistics about Downlink for all distinct Providers with time range-------------
--------------


    @RequestMapping(value = "/measurement/downlinkStats/{r0}/{r1}", method =
RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<OperatorStats>>
findOperatorDownlinkStatsWithParams(@PathVariable("r0") String r0, @PathVariable("r1")
String r1) {
                List<OperatorStats> measurements =
measurementService.findOperatorDownlinkStatsWithParams(new String[] {r0, r1});
        if(measurements.isEmpty()){
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }

    //--------------Get Statistics about Downlink for all distinct Providers with time range and
Network Type parameter -------


    @RequestMapping(value = "/measurement/downlinkStats/{r0}/{r1}/{networkType}", method
= RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<OperatorStats>>
findOperatorDownlinkStatsWithParamsForNetwType(@PathVariable("r0") String r0,
@PathVariable("r1") String r1, @PathVariable("networkType") String networkType) {
        List<OperatorStats> measurements =
measurementService.findOperatorDownlinkStatsWithParamsForNetwType(new String[] {r0, r1,
networkType});
        if(measurements.isEmpty()){
            return new ResponseEntity<List<OperatorStats>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<OperatorStats>>(measurements, HttpStatus.OK);
    }


    //------------------- Count All Measurements -----------------------------------------------------
    // This should be used as String with String.valueOf()
    @RequestMapping(value = "/measurement/count", method = RequestMethod.GET, produces
= MediaType.APPLICATION_JSON_VALUE)
    public long count() {
                long total = measurementService.count();
        return total;
    }
```

```
    //--------------Get Statistics about Network Type-------------------------

    @RequestMapping(value = "/measurement/networkTypes/", method = RequestMethod.GET,
produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<Statistics>> getNetworkTypes() {
        List<Statistics> measurements = measurementService.getNetworkTypes();
        if(measurements.isEmpty()){
            return new ResponseEntity<List<Statistics>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<Statistics>>(measurements, HttpStatus.OK);
    }

    //--------------Get Operating Systems-------------------------

    @RequestMapping(value = "/measurement/operatingSystems/", method =
RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<Statistics>> getOSs() {
        List<Statistics> measurements = measurementService.getOSs();
        if(measurements.isEmpty()){
            return new ResponseEntity<List<Statistics>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<Statistics>>(measurements, HttpStatus.OK);
    }

    //--------------Get Device Vendors-------------------------

    @RequestMapping(value = "/measurement/vendors/", method = RequestMethod.GET,
produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<Statistics>> getVendors() {
        List<Statistics> measurements = measurementService.getVendors();
        if(measurements.isEmpty()){
            return new ResponseEntity<List<Statistics>>(HttpStatus.NO_CONTENT);
//You many decide to return HttpStatus.NOT_FOUND
        }
        return new ResponseEntity<List<Statistics>>(measurements, HttpStatus.OK);
    }


}
```

Servlet-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:tx="http://www.springframework.org/schema/tx"
```

```
    xmlns:security="http://www.springframework.org/schema/security"
    xmlns:util="http://www.springframework.org/schema/util"
xmlns:jdbc="http://www.springframework.org/schema/jdbc"
    xmlns:repository="http://www.springframework.org/schema/data/repository"
    xmlns:jpa="http://www.springframework.org/schema/data/jpa"
    xsi:schemaLocation="http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-4.1.xsd
        http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-4.3.xsd
        http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd
        http://www.springframework.org/schema/data/repository
http://www.springframework.org/schema/data/repository/spring-repository-1.11.xsd
        http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util-4.3.xsd">
    <!-- DispatcherServlet Context: defines this servlet's request-processing
        infrastructure -->

    <!-- Enables the Spring MVC @Controller programming model -->
    <mvc:annotation-driven />

    <!-- Handles HTTP GET requests for /resources/** by efficiently serving
        up static resources in the ${webappRoot}/resources directory -->
    <mvc:resources mapping="/resources/**" location="/resources/" />

    <bean id="messageSource"
        class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
        <property name="basenames">
            <list>
                <value>classpath:validation</value>
            </list>
        </property>
    </bean>

    <!-- Resolves views selected for rendering by @Controllers to .jsp resources
        in the /WEB-INF/views directory -->
    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix">
            <value>/WEB-INF/views/</value>
        </property>
        <property name="suffix">
            <value>.jsp</value>
        </property>
```

```
    </bean>

    <!-- Hibernate 5 SessionFactory Bean definition -->
    <bean id="hibernate5AnnotatedSessionFactory"
      class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
      <property name="dataSource" ref="dataSource" />
      <property name="annotatedClasses">
        <list>
          <value>gr.huatester.model.Measurement</value>
          <value>gr.huatester.model.Campaign</value>
          <value>gr.huatester.model.C2dmRegDevice</value>
          <value>gr.huatester.model.DevicesPerCampaign</value>
        </list>
      </property>
      <property name="hibernateProperties">
        <props>
          <prop key="hibernate.dialect">org.hibernate.spatial.dialect.postgis.PostgisDialect
          </prop>
          <prop key="hibernate.show_sql">true</prop>
          <prop key="hibernate.hbm2ddl.auto">update</prop>
          <!-- prop key="hibernate.current_session_context_class">thread</prop -->
        </props>
      </property>
    </bean>

    <!-- Create DataSource Bean -->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
      destroy-method="close">
      <property name="driverClassName" value="org.postgis.DriverWrapper" />
      <property name="url"

value="jdbc:postgresql://test.hua.gr:5432/Measurements?ssl=true&amp;sslfactory=org.postgresq
l.ssl.NonValidatingFactory" />
      <property name="username" value="cr" />
      <property name="password" value="left_R" />
    </bean>

    <bean id="measurementDAO" class="gr.huatester.dao.MeasurementDAOImpl">
      <property name="sessionFactory" ref="hibernate5AnnotatedSessionFactory"></property>
    </bean>
    <bean id="campaignDAO" class="gr.huatester.dao.CampaignDAOImpl">
      <property name="sessionFactory" ref="hibernate5AnnotatedSessionFactory"></property>
    </bean>

    <bean id="measurementService" class="gr.huatester.service.MeasurementServiceImpl">
      <property name="measurementDAO" ref="measurementDAO"></property>
    </bean>

    <bean id="campaignService" class="gr.huatester.service.CampaignServiceImpl">
      <property name="campaignDAO" ref="campaignDAO"></property>
    </bean>
```

```
    <context:component-scan base-package="gr.huatester" />

    <!-- Enable annotation driven transaction management -->
    <tx:annotation-driven transaction-manager="transactionManager" />

    <!-- Configure the transaction manager bean -->
    <bean id="transactionManager"
        class="org.springframework.orm.hibernate5.HibernateTransactionManager">
        <property name="sessionFactory" ref="hibernate5AnnotatedSessionFactory"></property>
    </bean>

    <bean class="org.springframework.data.web.config.SpringDataWebConfiguration" />

    <mvc:resources mapping="/swagger-ui.html" location="classpath:/META-INF/resources/" />
    <mvc:resources mapping="/webjars/**"
        location="classpath:/META-INF/resources/webjars/" />

</beans>
```

JWTService

```
package gr.huatester.service;

import java.io.IOException;
import java.net.URISyntaxException;
import java.time.LocalDateTime;
import java.util.Date;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import gr.huatester.model.User;
import gr.huatester.security.SecretKeyProvider;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.JwtException;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;

import static java.time.ZoneOffset.UTC;

@Component
public class JwtService {

    private static final Logger logger = LoggerFactory.getLogger(JwtService.class);

    private static final String ISSUER = "gr.huatester";
    private SecretKeyProvider secretKeyProvider;

    @SuppressWarnings("unused")
    public JwtService() {
        this(null);
```

```
    }

    @Autowired
    public JwtService(SecretKeyProvider secretKeyProvider) {
        this.secretKeyProvider = secretKeyProvider;
    }

    public String tokenFor(User user) throws IOException, URISyntaxException {
        String secretKey = "testKEY";
        logger.info("------" + secretKey);


        Date expiration = Date.from(LocalDateTime.now().plusHours(2).toInstant(UTC));
        return Jwts.builder()
                .setSubject(user.getUserName())
                .setId(String.valueOf(user.getId()))
                .setExpiration(expiration)
                .setIssuer(ISSUER)
                .signWith(SignatureAlgorithm.HS512, secretKey)
                .compact();
    }

    public User parseToken(String token) throws Exception {
         String secretKey = "testKEY";
         logger.info("------INSIDE  parseToken");
        try {
            Claims body = Jwts.parser()
                    .setSigningKey(secretKey)
                    .parseClaimsJws(token)
                    .getBody();

            logger.info("------BODY " + body);
            User u = new User();
            u.setUserName(body.getSubject());
            u.setId(Integer.parseInt((String) body.getId()));

            logger.info("------>>>" + u.getUserName());

            return u;

        } catch (JwtException | ClassCastException e) {
             logger.info("------ inside jwt exception");
            return null;
        }
    }
}
```

MeasurementService

```
package gr.huatester.service;

import java.security.Timestamp;
```

```java
import java.util.List;

import gr.huatester.model.Measurement;
import gr.huatester.model.OperatorStats;
import gr.huatester.model.Operators;
import gr.huatester.model.Statistics;
import org.springframework.format.annotation.DateTimeFormat;

public interface MeasurementService {

    public void add(Measurement m);

    public void update(Measurement m);

    public List<Measurement> list();

    public Measurement getById(int id);

    public void remove(int id);

    public void deleteAll();

    public boolean exists(Measurement m);

    public List<Operators> findAll();

    public List<Operators> findAllForNetwType(String networkType);

    public List<OperatorStats> findOperatorLevelStats();

    public List<OperatorStats> findOperatorLevelStatsForNetwType(String networkType);

    public List<OperatorStats> findOperatorLevelStatsWithParams(String[] range);

    public List<OperatorStats> findOperatorLevelStatsWithParamsForNetwType(String[] range);

    public List<OperatorStats> findOperatorUplinkStats();

    public List<OperatorStats> findOperatorUplinkStatsForNetwType(String networkType);

    public List<OperatorStats> findOperatorUplinkStatsWithParams(String[] range);

    public List<OperatorStats> findOperatorUplinkStatsWithParamsForNetwType(String[]
range);

    public List<OperatorStats> findOperatorDownlinkStats();

    public List<OperatorStats> findOperatorDownlinkStatsForNetwType(String networkType);

    public List<OperatorStats> findOperatorDownlinkStatsWithParams(String[] range);
```

```
    public List<OperatorStats> findOperatorDownlinkStatsWithParamsForNetwType(String[]
range);

    public List<Measurement> findRange(int[] range);

    public long count();

    public String countMetrics(String osId);

    public List<Object[]> getMeasurements();

    public List<Statistics> getNetworkTypes();

    public List<Statistics> getOSs();

    public List<Statistics> getVendors();

}
```

MeasurementServiceImpl

```
package gr.huatester.service;

import java.security.Timestamp;
import java.util.List;

import gr.huatester.model.OperatorStats;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import gr.huatester.dao.MeasurementDAO;
import gr.huatester.model.Measurement;
import gr.huatester.model.Operators;
import gr.huatester.model.Statistics;

@Service
public class MeasurementServiceImpl implements MeasurementService {

    private MeasurementDAO measurementDAO;

    public void setMeasurementDAO(MeasurementDAO measurementDAO) {
        this.measurementDAO = measurementDAO;
    }

    @Override
    @Transactional
    public void add(Measurement m) {
        this.measurementDAO.add(m);
    }

    @Override
    @Transactional
```

```java
public void update(Measurement m) {
    this.measurementDAO.update(m);
}

@Override
@Transactional
public List<Measurement> list() {
    return this.measurementDAO.list();
}

@Override
@Transactional
public Measurement getById(int id) {
    return this.measurementDAO.getById(id);
}

@Override
@Transactional
public void remove(int id) {
    this.measurementDAO.remove(id);
}

@Override
@Transactional
public void deleteAll() {
    this.measurementDAO.deleteAll();
}

@Override
@Transactional
public boolean exists(Measurement m) {
    return this.measurementDAO.exists(m);
}

@Override
@Transactional
public List<Operators> findAll() {
    return this.measurementDAO.findAll();
}

@Override
@Transactional
public List<Operators> findAllForNetwType(String networkType) {
    return this.measurementDAO.findAllForNetwType(networkType);
}

@Override
@Transactional
public List<OperatorStats> findOperatorLevelStatsWithParams(String[] range) {
    return this.measurementDAO.findOperatorLevelStatsWithParams(range);
}
```

```
    @Override
    @Transactional
    public List<OperatorStats> findOperatorLevelStatsWithParamsForNetwType(String[] range)
{
        return this.measurementDAO.findOperatorLevelStatsWithParamsForNetwType(range);
    }


    @Override
    @Transactional
    public List<OperatorStats> findOperatorLevelStats() {
        return this.measurementDAO.findOperatorLevelStats();
    }

    @Override
    @Transactional
    public List<OperatorStats> findOperatorLevelStatsForNetwType(String networkType) {
        return this.measurementDAO.findOperatorLevelStatsForNetwType(networkType);
    }

    @Override
    @Transactional
    public List<OperatorStats> findOperatorUplinkStatsWithParams(String[] range) {
        return this.measurementDAO.findOperatorUplinkStatsWithParams(range);
    }

    @Override
    @Transactional
    public List<OperatorStats> findOperatorUplinkStatsWithParamsForNetwType(String[]
range) {
        return this.measurementDAO.findOperatorUplinkStatsWithParamsForNetwType(range);
    }


    @Override
    @Transactional
    public List<OperatorStats> findOperatorUplinkStats() {
        return this.measurementDAO.findOperatorUplinkStats();
    }

    @Override
    @Transactional
    public List<OperatorStats> findOperatorUplinkStatsForNetwType(String networkType) {
        return this.measurementDAO.findOperatorUplinkStatsForNetwType(networkType);
    }

    @Override
    @Transactional
    public List<OperatorStats> findOperatorDownlinkStatsWithParams(String[] range) {
        return this.measurementDAO.findOperatorDownlinkStatsWithParams(range);
    }
```

```
    @Override
    @Transactional
    public List<OperatorStats> findOperatorDownlinkStatsWithParamsForNetwType(String[]
range) {
        return
this.measurementDAO.findOperatorDownlinkStatsWithParamsForNetwType(range);
    }


    @Override
    @Transactional
    public List<OperatorStats> findOperatorDownlinkStats() {
        return this.measurementDAO.findOperatorDownlinkStats();
    }

    @Override
    @Transactional
    public List<OperatorStats> findOperatorDownlinkStatsForNetwType(String networkType) {
        return this.measurementDAO.findOperatorDownlinkStatsForNetwType(networkType);
    }

    @Override
    @Transactional
    public List<Measurement> findRange(int[] range) {
        return this.measurementDAO.findRange(range);
    }

    @Override
    @Transactional
    public long count() {
        return this.measurementDAO.count();
    }

    @Override
    @Transactional
    public String countMetrics(String osId) {
        return this.measurementDAO.countMetrics(osId);
    }

    @Override
    @Transactional
    public List<Object[]> getMeasurements() {
        return this.measurementDAO.getMeasurements();
    }

    @Override
    @Transactional
    public List<Statistics> getNetworkTypes() {
        return this.measurementDAO.getNetworkTypes();
    }

    @Override
```

```
    @Transactional
    public List<Statistics> getOSs() {
        return this.measurementDAO.getOSs();
    }

    @Override
    @Transactional
    public List<Statistics> getVendors() {
        return this.measurementDAO.getVendors();
    }
}
```

UserDAO

```
package gr.huatester.dao;

import java.util.List;
import gr.huatester.model.User;

//CRUD operations
public interface UserDAO {

    // Create
    public int save(User user);

    // Read
    public User getById(int id);

    // Update
    public void update(User user);

    // Delete
    public boolean deleteById(int id);

    // Get All
    public List<User> getAll();

    // login
    public User login(String username, String password);

}
```

UserDAOImpl

```
package gr.huatester.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.ArrayList;
```

```java
import java.util.List;
import java.util.Map;

import javax.sql.DataSource;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.dao.DataAccessException;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.jca.cci.InvalidResultSetAccessException;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.PreparedStatementCreator;
import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
import org.springframework.jdbc.support.GeneratedKeyHolder;
import org.springframework.jdbc.support.KeyHolder;

import gr.huatester.model.User;

public class UserDAOImpl implements UserDAO {

   private static final Logger logger = LoggerFactory.getLogger(UserDAOImpl.class);

   @Autowired
   @Qualifier("dataSource")
   private DataSource dataSource;

   public void setDataSource(DataSource dataSource) {
      this.dataSource = dataSource;
   }

   @Override
   public int save(User user) {
      JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);

      String INSERT_SQL = "insert into Users (name, email, country, password) values (?, ?, ?,
?)";
      KeyHolder keyHolder = new GeneratedKeyHolder();
      logger.info("--<<< before >>>>----");
      try {
         jdbcTemplate.update(new PreparedStatementCreator() {
            public PreparedStatement createPreparedStatement(Connection connection) throws
SQLException {
               PreparedStatement ps = connection.prepareStatement(INSERT_SQL, new String[] {
"id" });
               ps.setString(1, user.getUserName());
               ps.setString(2, user.getEmail());
               ps.setString(3, user.getCountry());
               ps.setString(4, user.getPassword());
```

```
                return ps;
            }
        }, keyHolder);
    } catch (Exception ex) {
        logger.info(ex.getMessage());
    }
    logger.info("--<<< after >>>>----");
    logger.info("--<<< after ID >>>>----" + keyHolder.getKey());

    return keyHolder.getKey().intValue();
}

@Override
public User getById(int id) {
    String query = "select * from Users where id = ?";
    JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);

    Object queryForObject = jdbcTemplate.queryForObject(query, new Object[] { id },
            new BeanPropertyRowMapper<User>(User.class));
    User user = (User) queryForObject;

    return user;
}

@Override
public void update(User user) {
    JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);
    jdbcTemplate.update("update Users set  username = ?, email = ?, country = ?, password = ?
where id = ?",
            user.getUserName(), user.getEmail(), user.getCountry(), user.getPassword(),
user.getId());

}

@Override
public boolean deleteById(int id) {
    Boolean state = false;
    try {
        JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);
        jdbcTemplate.update("delete from Users where id = ?", id);
        state = true;
    } catch (InvalidResultSetAccessException e) {
        state = false;
        logger.info("InvalidResultSetAccessException");

    } catch (DataAccessException e) {
        state = false;
        logger.info("DataAccessException " + e.getMessage());

    }

    return state;
```

```
    }

    @Override
    public List<User> getAll() {
        List<User> usersList = new ArrayList<User>();
        // JDBC Code - Start
        String query = "select * from Users";
        JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);

        List<Map<String, Object>> usersRows = jdbcTemplate.queryForList(query);

        for (Map<String, Object> userRow : usersRows) {
            User user = new User();
            user.setId(Integer.parseInt(String.valueOf(userRow.get("id"))));
            user.setUserName(String.valueOf(userRow.get("username")));
            user.setEmail(String.valueOf(userRow.get("email")));
            user.setCountry(String.valueOf(userRow.get("country")));
            usersList.add(user);
        }

        return usersList;
    }

    @Override
    public User login(String username, String password) {
        Boolean state = false;

        String query = "select * from Users where username = ? and password = ?";
        JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);
        try {
            Object queryForObject = jdbcTemplate.queryForObject(query, new Object[] { username,
password },
                    new BeanPropertyRowMapper<User>(User.class));

            User user = (User) queryForObject;

            return user;
        } catch (EmptyResultDataAccessException e) {
            return null;
        }
    }
}
```

MeasurementDAO

```
package gr.huatester.dao;

import java.security.Timestamp;
import java.util.List;

import gr.huatester.model.Measurement;
```

```
import gr.huatester.model.OperatorStats;
import gr.huatester.model.Operators;
import gr.huatester.model.Statistics;
import org.springframework.format.annotation.DateTimeFormat;

public interface MeasurementDAO {

    public void add(Measurement m);

    public void update(Measurement m);

    public List<Measurement> list();

    public Measurement getById(int id);

    public void remove(int id);

    public void deleteAll();

    public boolean exists(Measurement m);

    public List<Operators> findAll();

    public List<Operators> findAllForNetwType(String networkType);

    public List<OperatorStats> findOperatorLevelStats();

    public List<OperatorStats> findOperatorLevelStatsForNetwType(String networkType);

    public List<OperatorStats> findOperatorLevelStatsWithParams(String[] range);

    public List<OperatorStats> findOperatorLevelStatsWithParamsForNetwType(String[] range);

    public List<OperatorStats> findOperatorUplinkStats();

    public List<OperatorStats> findOperatorUplinkStatsForNetwType(String networkType);

    public List<OperatorStats> findOperatorUplinkStatsWithParams(String[] range);

    public List<OperatorStats> findOperatorUplinkStatsWithParamsForNetwType(String[]
range);

    public List<OperatorStats> findOperatorDownlinkStats();

    public List<OperatorStats> findOperatorDownlinkStatsForNetwType(String networkType);

    public List<OperatorStats> findOperatorDownlinkStatsWithParams(String[] range);

    public List<OperatorStats> findOperatorDownlinkStatsWithParamsForNetwType(String[]
range);

    public List<Measurement> findRange(int[] range);
```

```
    public long count();

    public String countMetrics(String osId);

    public List<Object[]> getMeasurements();

    public List<Statistics> getNetworkTypes();

    public List<Statistics> getOSs();

    public List<Statistics> getVendors();

}
```

MeasurementDAOImpl

```
package gr.huatester.dao;

import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import gr.huatester.model.OperatorStats;
import org.hibernate.ScrollMode;
import org.hibernate.ScrollableResults;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.StatelessSession;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.stereotype.Repository;

import gr.huatester.model.Measurement;
import gr.huatester.model.Operators;
import gr.huatester.model.Statistics;

import javax.persistence.TemporalType;

@Repository
public class MeasurementDAOImpl implements MeasurementDAO {

    private static final Logger logger = LoggerFactory.getLogger(MeasurementDAOImpl.class);

    private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sf) {
        this.sessionFactory = sf;
```

```
    }

    @Override
    public void add(Measurement m) {
        Session session = this.sessionFactory.getCurrentSession();
        session.persist(m);
        logger.info("Measurement saved successfully, Measurement Details=" + m);
    }

    @Override
    public void update(Measurement m) {
        Session session = this.sessionFactory.getCurrentSession();
        session.update(m);
        logger.info("Measurement updated successfully, Measurement Details=" + m);
    }

    @Override
    public List<Measurement> list() {
        Session session = this.sessionFactory.getCurrentSession();
        @SuppressWarnings("unchecked")
        List<Measurement> measurementsList = session.createQuery("from
Measurement").setMaxResults(10).getResultList();
        for (Measurement m : measurementsList) {
            logger.info("Measurement List::" + m);
        }
        StatelessSession slS = sessionFactory.openStatelessSession();
        try {
        slS.beginTransaction();
        @SuppressWarnings("deprecation")
        ScrollableResults sR = slS.createQuery("from
Measurement").setMaxResults(100000).scroll(ScrollMode.FORWARD_ONLY);
        int count = 0;
        while (sR.next()) {
            Measurement m = (Measurement) sR.get()[0];
            measurementsList.add(m);
            if (++count > 0 && count % 100 == 0) {
                logger.info("Fetched " + count + " entities");
            }
        }
        } finally{
            slS.close();
        }
        return measurementsList;
    }

    @Override
    public Measurement getById(int id) {
        Session session = this.sessionFactory.getCurrentSession();
        Measurement m = (Measurement) session.get(Measurement.class, new Integer(id));
        logger.info("Measurement loaded successfully, Measurement details=" + m);
        return m;
    }
```

```java
@Override
public void remove(int id) {
    Session session = this.sessionFactory.getCurrentSession();
    Measurement m = (Measurement) session.get(Measurement.class, new Integer(id));
    if (null != m) {
        session.delete(m);
    }
    logger.info("Measurement deleted successfully, measurement details=" + m);
}

@Override
public void deleteAll() {
    Session session = this.sessionFactory.getCurrentSession();
    session.createQuery("delete from Measurement").executeUpdate();
    logger.info("All measurements deleted succesfully!");
}

@Override
public boolean exists(Measurement m) {
    logger.info("Measurements existance check, Measurements details=" + m);
    return (getById(m.getId()) != null);
}

//count all distinct operators
@SuppressWarnings("unchecked")
@Override
public List<Operators> findAll() {
    Session session = this.sessionFactory.getCurrentSession();
    logger.info("Session" + session);
    String arg0 = "SELECT "
            + "CASE "
            + " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' "
            + " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' "
            + " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' "
            + "ELSE 'OTHER' END AS group_by_value, COUNT(*) AS group_b_count "
            + "FROM Measurement m "
            + "GROUP BY "
            + "CASE "
            + " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' "
            + " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' "
            + " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' "
            + "ELSE 'OTHER' END ";
    List<Object[]> measurementsList = null;
    List<Operators> result = null;
    try {
        measurementsList = session.createQuery(arg0).getResultList();
        result = new ArrayList<>(measurementsList.size());
        logger.info("List size" + measurementsList.size());
        for (Object[] o : measurementsList) {
            result.add(new Operators(
                    (String) o[0],
```

```
                (long) o[1] )
                );
        }
    } catch (Exception e) {
        logger.info("Error" + e.getMessage());
    }
    return result;
}

//count all distinct operators with Network Type parameter

@SuppressWarnings("unchecked")
@Override
public List<Operators> findAllForNetwType(String networkType) {
    Session session = this.sessionFactory.getCurrentSession();
    logger.info("Session" + session);
    String arg0 = "SELECT "
            + "CASE "
            + " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' "
            + " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' "
            + " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' "
            + "ELSE 'OTHER' END AS group_by_value, COUNT(*) AS group_b_count "
            + "FROM Measurement m "
            + "WHERE (m.networkType) = :network "
            + "GROUP BY "
            + "CASE "
            + " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' "
            + " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' "
            + " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' "
            + "ELSE 'OTHER' END ";
    List<Object[]> measurementsList = null;
    List<Operators> result = null;
    try {
        measurementsList = session.createQuery(arg0)
                .setParameter("network", networkType.toUpperCase())
                .getResultList();
        result = new ArrayList<>(measurementsList.size());
        logger.info("List size" + measurementsList.size());
        for (Object[] o : measurementsList) {
            result.add(new Operators(
                (String) o[0],
                (long) o[1] )
                );
        }
    } catch (Exception e) {
        logger.info("Error" + e.getMessage());
    }
    return result;
}

//Return min, max and avg Level for distinct providers
```

```
    @SuppressWarnings("unchecked")
    @Override
    public List<OperatorStats> findOperatorLevelStats() {
        Session session = this.sessionFactory.getCurrentSession();
        //language=HQL
        String arg0 ="SELECT " +
                "CASE " +
                " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
                " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
                " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
                "ELSE 'OTHER' " +
                "END AS group_by_value, avg(m.level) AS avg_level, min(m.level) AS min_level,
max(m.level) AS max_level\n" +
                "FROM Measurement m " +
                "GROUP BY " +
                "CASE " +
                " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
                " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
                " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
                "ELSE 'OTHER' END " ;
        List<Object[]> measurementsList = null;
        List<OperatorStats> result = null;
        try {
            measurementsList = session.createQuery(arg0).getResultList();
            result = new ArrayList<>(measurementsList.size());
            for (Object[] o : measurementsList) {
                result.add(new OperatorStats(
                        (String) o[0],
                        (int) o[2] ,
                        (int) o[3] ,
                        (double) o[1] )
                );
            }

        } catch (Exception e) {
            logger.info("Error" + e.getMessage());
        }
        return result;
    }

    //Return min, max and avg Level for distinct providers with network parameter

    @SuppressWarnings("unchecked")
    @Override
    public List<OperatorStats> findOperatorLevelStatsForNetwType(String networkType) {
        Session session = this.sessionFactory.getCurrentSession();
        //language=HQL
        String arg0 ="SELECT " +
                "CASE " +
                " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
                " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
                " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
```

```
            "ELSE 'OTHER' " +
            "END AS group_by_value, avg(m.level) AS avg_level, min(m.level) AS min_level,
max(m.level) AS max_level\n" +
            "FROM Measurement m " +
            "WHERE (m.networkType) = :network "+
            "GROUP BY " +
            "CASE " +
            " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
            " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
            " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
            "ELSE 'OTHER' END " ;
        List<Object[]> measurementsList = null;
        List<OperatorStats> result = null;
        try {
            measurementsList = session.createQuery(arg0)
                    .setParameter("network", networkType.toUpperCase())
                    .getResultList();
            result = new ArrayList<>(measurementsList.size());
            for (Object[] o : measurementsList) {
                result.add(new OperatorStats(
                        (String) o[0],
                        (int) o[2] ,
                        (int) o[3] ,
                        (double) o[1] )
                );
            }

        } catch (Exception e) {
            logger.info("Error" + e.getMessage());
        }
        return result;
    }

    //Return min, max and avg Level for distinct providers with date range parameter

    @SuppressWarnings("unchecked")
    @Override
    public List<OperatorStats> findOperatorLevelStatsWithParams(String[] range) {
        Session session = this.sessionFactory.getCurrentSession();
        String arg0 ="SELECT\n" +
            "CASE " +
            "WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
            " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
            " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
            "ELSE 'OTHER'\n" +
            "END AS group_by_value, avg(m.level) AS avg_level,\n" +
            "\tmin(m.level) AS min_level,\n" +
            "\tmax(m.level) AS max_level\n" +
            "FROM Measurement m " +
            "WHERE m.timestamp BETWEEN ? AND ?" +
            " group by " +
            "CASE\n" +
```

```
            "\tWHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
            "\tWHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
            "\tWHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
            "ELSE 'OTHER' END\n";

        List<Object[]> measurementsList = null;
        List<OperatorStats> result = null;
        try {
            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
            Date parsedStartDate = dateFormat.parse(range[0]);
            Timestamp StartTimestamp = new java.sql.Timestamp(parsedStartDate.getTime());
            Date parsedEndDate = dateFormat.parse(range[1]);
            Timestamp EndTimestamp = new java.sql.Timestamp(parsedEndDate.getTime());
            measurementsList = session.createQuery(arg0)
                    .setParameter(0, StartTimestamp)
                    .setParameter(1, EndTimestamp)
                    .getResultList();
            result = new ArrayList<>(measurementsList.size());
            for (Object[] o : measurementsList) {
                logger.info("Object List::" + o.toString());

                result.add(new OperatorStats(
                        (String) o[0],
                        (int) o[2] ,
                        (int) o[3] ,
                        (double) o[1] )
                );
            }

        } catch (Exception e) {
            logger.info("Error" + e.getMessage());
        }
        return result;
    }

    //Return min, max and avg Level for distinct providers with date range and network type
parameter

    @Override
    public List<OperatorStats> findOperatorLevelStatsWithParamsForNetwType(String[] range)
{
        Session session = this.sessionFactory.getCurrentSession();
        String arg0 ="SELECT\n" +
            "CASE " +
            "WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
            " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
            " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
            "ELSE 'OTHER'\n" +
            "END AS group_by_value, avg(m.level) AS avg_level,\n" +
            "\tmin(m.level) AS min_level,\n" +
            "\tmax(m.level) AS max_level\n" +
            "FROM Measurement m " +
```

```
            "WHERE m.timestamp BETWEEN (?1) AND (?2)" +
            "and (m.networkType) = (?3) "+
            " group by " +
            "CASE\n" +
            "\tWHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
            "\tWHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
            "\tWHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
            "ELSE 'OTHER' END\n";

    List<Object[]> measurementsList = null;
    List<OperatorStats> result = null;
    try {
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        Date parsedStartDate = dateFormat.parse(range[0]);
        Timestamp StartTimestamp = new java.sql.Timestamp(parsedStartDate.getTime());
        Date parsedEndDate = dateFormat.parse(range[1]);
        Timestamp EndTimestamp = new java.sql.Timestamp(parsedEndDate.getTime());
        measurementsList = session.createQuery(arg0)
                .setParameter(1, StartTimestamp)
                .setParameter(2, EndTimestamp)
                .setParameter(3, range[2].toUpperCase())
                .getResultList();
        result = new ArrayList<>(measurementsList.size());
        for (Object[] o : measurementsList) {

            result.add(new OperatorStats(
                    (String) o[0],
                    (int) o[2] ,
                    (int) o[3] ,
                    (double) o[1] )
            );
        }

    } catch (Exception e) {
        logger.info("Error" + e.getMessage());
    }
    return result;
}

//Return min, max and avg Uplink for distinct providers

@SuppressWarnings("unchecked")
@Override
public List<OperatorStats> findOperatorUplinkStats() {
    Session session = this.sessionFactory.getCurrentSession();
    //language=HQL
    String arg0 ="SELECT " +
            "CASE " +
            " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
            " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
            " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
            "ELSE 'OTHER' " +
```

```
            "END AS group_by_value, avg(m.ulBitrate) AS avg_Uplink, min(m.ulBitrate) AS
min_Uplink, max(m.ulBitrate) AS max_Uplink\n" +
            "FROM Measurement m " +
            "GROUP BY " +
            "CASE " +
            " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
            " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
            " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
            "ELSE 'OTHER' END " ;
        List<Object[]> measurementsList = null;
        List<OperatorStats> result = null;
        try {
            measurementsList = session.createQuery(arg0).getResultList();
            result = new ArrayList<>(measurementsList.size());
            for (Object[] o : measurementsList) {
                result.add(new OperatorStats(
                        (String) o[0],
                        (int) o[2] ,
                        (int) o[3] ,
                        (double) o[1] )
                );
            }

        } catch (Exception e) {
            logger.info("Error" + e.getMessage());
        }
        return result;
    }

    //Return min, max and avg Uplink for distinct providers with network type parameter

    @SuppressWarnings("unchecked")
    @Override
    public List<OperatorStats> findOperatorUplinkStatsForNetwType(String networkType) {
        Session session = this.sessionFactory.getCurrentSession();
        //language=HQL
        String arg0 ="SELECT " +
            "CASE " +
            " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
            " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
            " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
            "ELSE 'OTHER' " +
            "END AS group_by_value, avg(m.ulBitrate) AS avg_Uplink, min(m.ulBitrate) AS
min_Uplink, max(m.ulBitrate) AS max_Uplink\n" +
            "FROM Measurement m " +
            "WHERE (m.networkType) = :network "+
            "GROUP BY " +
            "CASE " +
            " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
            " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
            " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
            "ELSE 'OTHER' END " ;
```

```
        List<Object[]> measurementsList = null;
        List<OperatorStats> result = null;
        try {
            measurementsList = session.createQuery(arg0)
                    .setParameter("network", networkType.toUpperCase())
                    .getResultList();
            result = new ArrayList<>(measurementsList.size());
            for (Object[] o : measurementsList) {
                result.add(new OperatorStats(
                        (String) o[0],
                        (int) o[2] ,
                        (int) o[3] ,
                        (double) o[1] )
                );
            }

        } catch (Exception e) {
            logger.info("Error" + e.getMessage());
        }
        return result;
    }

    //Return min, max and avg Uplink for distinct providers with date range parameter

    @SuppressWarnings("unchecked")
    @Override
    public List<OperatorStats> findOperatorUplinkStatsWithParams(String[] range) {
        Session session = this.sessionFactory.getCurrentSession();
        String arg0 ="SELECT\n" +
                "CASE " +
                "WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
                " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
                " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
                "ELSE 'OTHER'\n" +
                "END AS group_by_value, avg(m.ulBitrate) AS avg_Uplink, min(m.ulBitrate) AS
min_Uplink, max(m.ulBitrate) AS max_Uplink\n" +
                "FROM Measurement m " +
                "WHERE m.timestamp BETWEEN ? AND ?" +
                " group by " +
                "CASE\n" +
                "\tWHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
                "\tWHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
                "\tWHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
                "ELSE 'OTHER' END\n";

        List<Object[]> measurementsList = null;
        List<OperatorStats> result = null;
        try {
            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
            Date parsedStartDate = dateFormat.parse(range[0]);
            Timestamp StartTimestamp = new java.sql.Timestamp(parsedStartDate.getTime());
            Date parsedEndDate = dateFormat.parse(range[1]);
```

```
        Timestamp EndTimestamp = new java.sql.Timestamp(parsedEndDate.getTime());
        measurementsList = session.createQuery(arg0)
                .setParameter(0, StartTimestamp)
                .setParameter(1, EndTimestamp)
                .getResultList();
        result = new ArrayList<>(measurementsList.size());
        for (Object[] o : measurementsList) {

            result.add(new OperatorStats(
                    (String) o[0],
                    (int) o[2] ,
                    (int) o[3] ,
                    (double) o[1] )
            );
        }

    } catch (Exception e) {
        logger.info("Error" + e.getMessage());
    }
    return result;
}

//Return min, max and avg Uplink for distinct providers with date range and network type
parameter

@Override
public List<OperatorStats> findOperatorUplinkStatsWithParamsForNetwType(String[]
range) {
    Session session = this.sessionFactory.getCurrentSession();
    String arg0 ="SELECT\n" +
            "CASE " +
            "WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
            " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
            " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
            "ELSE 'OTHER'\n" +
            "END AS group_by_value, avg(m.ulBitrate) AS avg_Uplink, min(m.ulBitrate) AS
min_Uplink, max(m.ulBitrate) AS max_Uplink\n" +
            "FROM Measurement m " +
            "WHERE m.timestamp BETWEEN (?1) AND (?2)" +
            "and (m.networkType) = (?3) "+
            " group by " +
            "CASE\n" +
            "\tWHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
            "\tWHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
            "\tWHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
            "ELSE 'OTHER' END\n";

    List<Object[]> measurementsList = null;
    List<OperatorStats> result = null;
    try {
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        Date parsedStartDate = dateFormat.parse(range[0]);
```

```
            Timestamp StartTimestamp = new java.sql.Timestamp(parsedStartDate.getTime());
            Date parsedEndDate = dateFormat.parse(range[1]);
            Timestamp EndTimestamp = new java.sql.Timestamp(parsedEndDate.getTime());
            measurementsList = session.createQuery(arg0)
                    .setParameter(1, StartTimestamp)
                    .setParameter(2, EndTimestamp)
                    .setParameter(3, range[2].toUpperCase())
                    .getResultList();
            result = new ArrayList<>(measurementsList.size());
            for (Object[] o : measurementsList) {

                result.add(new OperatorStats(
                        (String) o[0],
                        (int) o[2] ,
                        (int) o[3] ,
                        (double) o[1] )
                );
            }

        } catch (Exception e) {
            logger.info("Error" + e.getMessage());
        }
        return result;
    }

    //Return min, max and avg Downlink for distinct providers

    @SuppressWarnings("unchecked")
    @Override
    public List<OperatorStats> findOperatorDownlinkStats() {
        Session session = this.sessionFactory.getCurrentSession();
        //language=HQL
        String arg0 ="SELECT " +
                "CASE " +
                " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
                " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
                " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
                "ELSE 'OTHER' " +
                "END AS group_by_value, avg(m.dlBitrate) AS avg_Downlink, min(m.dlBitrate) AS
min_Downlink, max(m.dlBitrate) AS max_Downlink\n" +
                "FROM Measurement m " +
                "GROUP BY " +
                "CASE " +
                " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
                " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
                " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
                "ELSE 'OTHER' END " ;
        List<Object[]> measurementsList = null;
        List<OperatorStats> result = null;
        try {
            measurementsList = session.createQuery(arg0).getResultList();
            result = new ArrayList<>(measurementsList.size());
```

```
            for (Object[] o : measurementsList) {
                result.add(new OperatorStats(
                        (String) o[0],
                        (int) o[2] ,
                        (int) o[3] ,
                        (double) o[1] )
                );
            }

        } catch (Exception e) {
            logger.info("Error" + e.getMessage());
        }
        return result;
    }

    //Return min, max and avg Downlink for distinct providers with network type  parameter

    @SuppressWarnings("unchecked")
    @Override
    public List<OperatorStats> findOperatorDownlinkStatsForNetwType(String networkType) {
        Session session = this.sessionFactory.getCurrentSession();
        //language=HQL
        String arg0 ="SELECT " +
                "CASE " +
                " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
                " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
                " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
                "ELSE 'OTHER' " +
                "END AS group_by_value, avg(m.dlBitrate) AS avg_Downlink, min(m.dlBitrate) AS
min_Downlink, max(m.dlBitrate) AS max_Downlink\n" +
                "FROM Measurement m " +
                "WHERE (m.networkType) = :network "+
                "GROUP BY " +
                "CASE " +
                " WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE' " +
                " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND' " +
                " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE' " +
                "ELSE 'OTHER' END " ;
        List<Object[]> measurementsList = null;
        List<OperatorStats> result = null;
        try {
            measurementsList = session.createQuery(arg0)
                    .setParameter("network", networkType.toUpperCase())
                    .getResultList();
            result = new ArrayList<>(measurementsList.size());
            for (Object[] o : measurementsList) {
                result.add(new OperatorStats(
                        (String) o[0],
                        (int) o[2] ,
                        (int) o[3] ,
                        (double) o[1] )
                );
```

```
        }

    } catch (Exception e) {
        logger.info("Error" + e.getMessage());
    }
    return result;
}

//Return min, max and avg Downlink for distinct providers with date range parameter

@SuppressWarnings("unchecked")
@Override
public List<OperatorStats> findOperatorDownlinkStatsWithParams(String[] range) {
    Session session = this.sessionFactory.getCurrentSession();
    String arg0 ="SELECT\n" +
            "CASE " +
            "WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
            " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
            " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
            "ELSE 'OTHER'\n" +
            "END AS group_by_value, avg(m.dlBitrate) AS avg_Downlink, min(m.dlBitrate) AS
min_Downlink, max(m.dlBitrate) AS max_Downlink\n" +
            "FROM Measurement m " +
            "WHERE m.timestamp BETWEEN ? AND ?" +
            " group by " +
            "CASE\n" +
            "\tWHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
            "\tWHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
            "\tWHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
            "ELSE 'OTHER' END\n";

    List<Object[]> measurementsList = null;
    List<OperatorStats> result = null;
    try {
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        Date parsedStartDate = dateFormat.parse(range[0]);
        Timestamp StartTimestamp = new java.sql.Timestamp(parsedStartDate.getTime());
        Date parsedEndDate = dateFormat.parse(range[1]);
        Timestamp EndTimestamp = new java.sql.Timestamp(parsedEndDate.getTime());
        measurementsList = session.createQuery(arg0)
                .setParameter(0, StartTimestamp)
                .setParameter(1, EndTimestamp)
                .getResultList();
        result = new ArrayList<>(measurementsList.size());
        for (Object[] o : measurementsList) {

            result.add(new OperatorStats(
                    (String) o[0],
                    (int) o[2] ,
                    (int) o[3] ,
                    (double) o[1] )
            );
```

```
            }

        } catch (Exception e) {
            logger.info("Error" + e.getMessage());
        }
        return result;
    }

    //Return min, max and avg Downlink for distinct providers with date range and network type
parameter

    @Override
    public List<OperatorStats> findOperatorDownlinkStatsWithParamsForNetwType(String[]
range) {
        Session session = this.sessionFactory.getCurrentSession();
        String arg0 ="SELECT\n" +
                "CASE " +
                "WHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
                " WHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
                " WHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
                "ELSE 'OTHER'\n" +
                "END AS group_by_value, avg(m.dlBitrate) AS avg_Downlink, min(m.dlBitrate) AS
min_Downlink, max(m.dlBitrate) AS max_Downlink\n" +
                "FROM Measurement m " +
                "WHERE m.timestamp BETWEEN (?1) AND (?2)" +
                "and (m.networkType) = (?3) "+
                " group by " +
                "CASE\n" +
                "\tWHEN lower(operatorname) LIKE '%cosmote%'   THEN 'COSMOTE'\n" +
                "\tWHEN lower(operatorname) LIKE '%wind%'   THEN 'WIND'\n" +
                "\tWHEN lower(operatorname) LIKE '%vodafone%' THEN 'VODAFONE'\n" +
                "ELSE 'OTHER' END\n";

        List<Object[]> measurementsList = null;
        List<OperatorStats> result = null;
        try {
            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
            Date parsedStartDate = dateFormat.parse(range[0]);
            Timestamp StartTimestamp = new java.sql.Timestamp(parsedStartDate.getTime());
            Date parsedEndDate = dateFormat.parse(range[1]);
            Timestamp EndTimestamp = new java.sql.Timestamp(parsedEndDate.getTime());
            measurementsList = session.createQuery(arg0)
                    .setParameter(1, StartTimestamp)
                    .setParameter(2, EndTimestamp)
                    .setParameter(3, range[2].toUpperCase())
                    .getResultList();
            result = new ArrayList<>(measurementsList.size());
            for (Object[] o : measurementsList) {

                result.add(new OperatorStats(
                        (String) o[0],
                        (int) o[2] ,
```

```
                (int) o[3] ,
                (double) o[1] )
            );
        }

    } catch (Exception e) {
        logger.info("Error" + e.getMessage());
    }
    return result;
}


@SuppressWarnings("unchecked")
@Override
public List<Measurement> findRange(int[] range) {
    Session session = this.sessionFactory.getCurrentSession();
    List<Measurement> measurementsList = session.createQuery("from
Measurement").setFirstResult(range[0])
            .setMaxResults(range[1] - range[0] + 1).getResultList();
    for (Measurement m : measurementsList) {
        logger.info("Measurement List::" + m);
    }
    return measurementsList;
}

@Override
public long count() {
    Session session = this.sessionFactory.getCurrentSession();
    long total = (long)
session.createNamedQuery("Measurements.countAll").getSingleResult();
    return total;
}

// device's amount of mobile measurements in current month
@Override
public String countMetrics(String osId) {
    Session session = this.sessionFactory.getCurrentSession();
    String total = session
            .createQuery("SELECT count(*) FROM Measurement WHERE osId='" + osId + "'
AND networkType<>'' "
                    + "AND month(ctime)=month(CURRENT_TIMESTAMP) " + " AND
year(ctime)=year(CURRENT_TIMESTAMP) ")
            .getSingleResult().toString();
    return total;
}

// retrieve amount of measurements per campaign
@SuppressWarnings("unchecked")
@Override
public List<Object[]> getMeasurements() {
    // String hql = "SELECT m.campaign.campId, count(m) FROM Measurement m
    // GROUP BY campaign";
```

```
        Session session = this.sessionFactory.getCurrentSession();
        List<Object[]> measurementsList =
session.createNamedQuery("Measurements.groupByCampID").getResultList();
        for (Object[] m : measurementsList) {
        }
        return measurementsList;
    }
    //Retrieve distinct network types
    @SuppressWarnings("unchecked")
    @Override
    public List<Statistics> getNetworkTypes(){
        Session session = this.sessionFactory.getCurrentSession();

        String arg0="SELECT "+
                "DISTINCT m.networkType, count(*) " +
                "FROM Measurement m " +
                "WHERE m.networkType IS NOT NULL " +
                "GROUP BY m.networkType";

        List<Object[]> measurementsList = null;
        List<Statistics> result = null;

        try {
            measurementsList = session.createQuery(arg0).getResultList();
            result = new ArrayList<>(measurementsList.size());
            for (Object[] o : measurementsList) {

                result.add(new Statistics(
                        (String) o[0],
                        (Long) o[1])
                );
            }

        } catch (Exception e) {
            logger.info("Error" + e.getMessage());
        }
        return result;

    }
//  Get distinct Operating Systems
    @SuppressWarnings("unchecked")
    @Override
    public List<Statistics> getOSs(){
        Session session = this.sessionFactory.getCurrentSession();

        String arg0="SELECT "+
                "m.versionName, count(*) " +
                "FROM Measurement m " +
                "GROUP BY m.versionName";

        List<Object[]> measurementsList = null;
        List<Statistics> result = null;
```

```
        try {
            measurementsList = session.createQuery(arg0).getResultList();
            result = new ArrayList<>(measurementsList.size());
            for (Object[] o : measurementsList) {
                if (o[0]==null){
                    o[0]="Unknown";
                }
                result.add(new Statistics(
                        (String) o[0],
                        (Long) o[1])
                );
            }

        } catch (Exception e) {
            logger.info("Error" + e.getMessage());
        }
        return result;

    }

// Get distinct Device Vendors

    @SuppressWarnings("unchecked")
    @Override
    public List<Statistics> getVendors(){
        Session session = this.sessionFactory.getCurrentSession();

        String arg0="SELECT "+
                "m.deviceManufacturer, count(*) " +
                "FROM Measurement m " +
                "GROUP BY m.deviceManufacturer";

        List<Object[]> measurementsList = null;
        List<Statistics> result = null;

        try {
            measurementsList = session.createQuery(arg0).getResultList();
            result = new ArrayList<>(measurementsList.size());
            for (Object[] o : measurementsList) {
                if (o[0]==null){
                    o[0]="Unknown";
                }
                result.add(new Statistics(
                        (String) o[0],
                        (Long) o[1])
                );
            }
        } catch (Exception e) {
            logger.info("Error" + e.getMessage());
        }
        return result;
```

```
  }
}
```

Measurement Model

```
package gr.huatester.model;

import java.io.Serializable;
import javax.persistence.*;

import com.fasterxml.jackson.annotation.JsonAutoDetect;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonInclude;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

import java.math.BigDecimal;
import java.sql.Timestamp;


/**
 * The persistent class for the "Measurements" database table.
 *
 */
@ApiModel
@Entity
@Table(name="\"Measurements\"")
@NamedQueries({ @NamedQuery(name = "Measurements.findAll", query = "SELECT m
FROM Measurement m"),
   @NamedQuery(name = "Measurements.findByID", query = "SELECT m FROM
Measurement m WHERE m.id = :id"),
   @NamedQuery(name = "Measurements.findByACCURACY", query = "SELECT m FROM
Measurement m WHERE m.accuracy = :accuracy"),
   @NamedQuery(name = "Measurements.findByALTITUDE", query = "SELECT m FROM
Measurement m WHERE m.altitude = :altitude"),
   @NamedQuery(name = "Measurements.findByANDROIDVERSION", query = "SELECT m
FROM Measurement m WHERE m.androidVersion = :androidVersion"),
   @NamedQuery(name = "Measurements.findByAPPVERSIONCODE", query = "SELECT m
FROM Measurement m WHERE m.appversioncode = :appversioncode"),
   @NamedQuery(name = "Measurements.findByAVGPING", query = "SELECT m FROM
Measurement m WHERE m.avgping = :avgping"),
   @NamedQuery(name = "Measurements.findByBRAND", query = "SELECT m FROM
Measurement m WHERE m.brand = :brand"),
   @NamedQuery(name = "Measurements.findByCELLID", query = "SELECT m FROM
Measurement m WHERE m.cellid = :cellid"),
   @NamedQuery(name = "Measurements.findByCONNINFO", query = "SELECT m FROM
Measurement m WHERE m.conninfo = :conninfo"),
   @NamedQuery(name = "Measurements.findByCONNTYPE", query = "SELECT m FROM
Measurement m WHERE m.conntype = :conntype"),
```

```
    @NamedQuery(name = "Measurements.findByCQI", query = "SELECT m FROM
Measurement m WHERE m.cqi = :cqi"),
    @NamedQuery(name = "Measurements.findByCTIME", query = "SELECT m FROM
Measurement m WHERE date(m.ctime) = :ctime"),
    @NamedQuery(name = "Measurements.findByDEVICEMANUFACTURER", query =
"SELECT m FROM Measurement m WHERE m.deviceManufacturer = :deviceManufacturer"),
    @NamedQuery(name = "Measurements.findByDEVICEMODEL", query = "SELECT m
FROM Measurement m WHERE m.deviceModel = :deviceModel"),
    @NamedQuery(name = "Measurements.findByDEVICENAME", query = "SELECT m
FROM Measurement m WHERE m.deviceName = :deviceName"),
    @NamedQuery(name = "Measurements.findByDLBITRATE", query = "SELECT m FROM
Measurement m WHERE m.dlBitrate = :dlBitrate"),
    @NamedQuery(name = "Measurements.findByEVENT", query = "SELECT m FROM
Measurement m WHERE m.event = :event"),
    @NamedQuery(name = "Measurements.findByLAC", query = "SELECT m FROM
Measurement m WHERE m.lac = :lac"),
    @NamedQuery(name = "Measurements.findByLAT", query = "SELECT m FROM
Measurement m WHERE m.lat = :lat"),
    @NamedQuery(name = "Measurements.findByLEVEL", query = "SELECT m FROM
Measurement m WHERE m.level = :level"),
    @NamedQuery(name = "Measurements.findByLOCATIONSOURCE", query = "SELECT m
FROM Measurement m WHERE m.locationsource = :locationsource"),
    @NamedQuery(name = "Measurements.findByLON", query = "SELECT m FROM
Measurement m WHERE m.lon = :lon"),
    @NamedQuery(name = "Measurements.findByLTERSSI", query = "SELECT m FROM
Measurement m WHERE m.lterssi = :lterssi"),
    @NamedQuery(name = "Measurements.findByMAXPING", query = "SELECT m FROM
Measurement m WHERE m.maxping = :maxping"),
    @NamedQuery(name = "Measurements.findByMCC", query = "SELECT m FROM
Measurement m WHERE m.mcc = :mcc"),
    @NamedQuery(name = "Measurements.findByMINPING", query = "SELECT m FROM
Measurement m WHERE m.minping = :minping"),
    @NamedQuery(name = "Measurements.findByMNC", query = "SELECT m FROM
Measurement m WHERE m.mnc = :mnc"),
    @NamedQuery(name = "Measurements.findByNETWORKTYPE", query = "SELECT m
FROM Measurement m WHERE m.networkType = :networkType"),
    @NamedQuery(name = "Measurements.findByNODE", query = "SELECT m FROM
Measurement m WHERE m.node = :node"),
    @NamedQuery(name = "Measurements.findByOPERATORNAME", query = "SELECT m
FROM Measurement m WHERE m.operatorname = :operatorname"),
    @NamedQuery(name = "Measurements.findByPINGLOSS", query = "SELECT m FROM
Measurement m WHERE m.pingloss = :pingloss"),
    @NamedQuery(name = "Measurements.findByPSC", query = "SELECT m FROM
Measurement m WHERE m.psc = :psc"),
    @NamedQuery(name = "Measurements.findByQUAL", query = "SELECT m FROM
Measurement m WHERE m.qual = :qual"),
    @NamedQuery(name = "Measurements.findBySERVINGCELLTIME", query = "SELECT m
FROM Measurement m WHERE m.servingCellTime = :servingCellTime"),
    @NamedQuery(name = "Measurements.findBySNR", query = "SELECT m FROM
Measurement m WHERE m.snr = :snr"),
    @NamedQuery(name = "Measurements.findBySPEED", query = "SELECT m FROM
Measurement m WHERE m.speed = :speed"),
```

```
    @NamedQuery(name = "Measurements.findBySTDEVPING", query = "SELECT m FROM
Measurement m WHERE m.stdevping = :stdevping"),
    @NamedQuery(name = "Measurements.findByTESTDLBITRATE", query = "SELECT m
FROM Measurement m WHERE m.testdlbitrate = :testdlbitrate"),
    @NamedQuery(name = "Measurements.findByTESTULBITRATE", query = "SELECT m
FROM Measurement m WHERE m.testulbitrate = :testulbitrate"),
    @NamedQuery(name = "Measurements.findByTIMESTAMP", query = "SELECT m FROM
Measurement m WHERE m.timestamp = :timestamp"),
    @NamedQuery(name = "Measurements.findByULBITRATE", query = "SELECT m FROM
Measurement m WHERE m.ulBitrate = :ulBitrate"),
    @NamedQuery(name = "Measurements.findByOSID", query = "SELECT m FROM
Measurement m WHERE m.osId = :osId"),
    @NamedQuery(name = "Measurements.findBySsid", query = "SELECT m FROM
Measurement m WHERE m.ssid = :sid"),
    @NamedQuery(name = "Measurements.groupByCampID", query = "SELECT
m.campaign.campId, count(m) FROM Measurement m GROUP BY m.campaign.campId"),
    @NamedQuery(name = "Measurements.groupByNetworkType", query = "SELECT
m.networkType, count(m) FROM Measurement m GROUP BY m.networkType"),
    @NamedQuery(name = "Measurements.countAll", query = "SELECT count(m) FROM
Measurement m"),
    @NamedQuery(name = "Measurements.findByVERSIONNAME", query = "SELECT m
FROM Measurement m WHERE m.versionName = :versionName") })


public class Measurement implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(unique=true, nullable=false)
    private Integer id;

    private Integer accuracy;

    private Integer altitude;

    @Column(name="\"AndroidVersion\"", length=30)
    private String androidVersion;

    private Integer appversioncode;

    private Integer avgping;

    @Column(name="\"Brand\"", length=30)
    private String brand;

    @Column(length=11)
    private String cellid;

    @Column(length=25)
    private String conninfo;
```

```
    @Column(length=5)
    private String conntype;

    private Integer cqi;

    private Timestamp ctime;

    @Column(name="\"DeviceManufacturer\"", length=30)
    private String deviceManufacturer;

    @Column(name="\"DeviceModel\"", length=30)
    private String deviceModel;

    @Column(name="\"DeviceName\"", length=30)
    private String deviceName;

    @Column(name="dl_bitrate")
    private Integer dlBitrate;

    @Column(length=20)
    private String event;
/*
    @Column(name="\"GeomCol1\"")
    private byte[] geomCol1;

    @Column(name="\"GeomCol2\"")
    private Object geomCol2;
*/
    @Column(length=11)
    private String lac;

    @Column(precision=10, scale=5)
    private BigDecimal lat;

    private Integer level;

    @Column(length=2)
    private String locationsource;

    @Column(precision=10, scale=5)
    private BigDecimal lon;

    private Integer lterssi;

    private Integer maxping;

    @Column(length=3)
    private String mcc;

    private Integer minping;

    @Column(length=3)
```

```
private String mnc;

private Integer ncellid1;

private Integer ncellid2;

private Integer ncellid3;

private Integer ncellid4;

private Integer ncellid5;

private Integer ncellid6;

@Column(name="network_type", length=2)
private String networkType;

private Integer nlac1;

private Integer nlac2;

private Integer nlac3;

private Integer nlac4;

private Integer nlac5;

private Integer nlac6;

@Column(length=11)
private String node;

private Integer nrxlev1;

private Integer nrxlev2;

private Integer nrxlev3;

private Integer nrxlev4;

private Integer nrxlev5;

private Integer nrxlev6;

@Column(length=50)
private String operatorname;

@Column(length=30)
private String os;

@Column(name="os_id", length=128)
private String osId;
```

```
private Integer pingloss;

private Integer psc;

private Integer qual;

@Column(name="\"ServingCellTime\"")
private Integer servingCellTime;

private Integer snr;

private Integer speed;

@Column(length=128)
private String ssid;

private Integer stdevping;

private Integer testdlbitrate;

private Integer testulbitrate;

private Timestamp timestamp;

@Column(name="ul_bitrate")
private Integer ulBitrate;

@Column(name="\"VersionName\"", length=30)
private String versionName;

//bi-directional many-to-one association to Campaign
@JsonIgnore
@JsonIgnoreProperties(ignoreUnknown = true)
@ManyToOne
@JoinColumn(name="camp_id")
private Campaign campaign;

public Measurement() {
}

public Integer getId() {
   return this.id;
}

public void setId(Integer id) {
   this.id = id;
}

public Integer getAccuracy() {
   return this.accuracy;
}
```

```java
public void setAccuracy(Integer accuracy) {
    this.accuracy = accuracy;
}

public Integer getAltitude() {
    return this.altitude;
}

public void setAltitude(Integer altitude) {
    this.altitude = altitude;
}

public String getAndroidVersion() {
    return this.androidVersion;
}

public void setAndroidVersion(String androidVersion) {
    this.androidVersion = androidVersion;
}

public Integer getAppversioncode() {
    return this.appversioncode;
}

public void setAppversioncode(Integer appversioncode) {
    this.appversioncode = appversioncode;
}

public Integer getAvgping() {
    return this.avgping;
}

public void setAvgping(Integer avgping) {
    this.avgping = avgping;
}

public String getBrand() {
    return this.brand;
}

public void setBrand(String brand) {
    this.brand = brand;
}

public String getCellid() {
    return this.cellid;
}

public void setCellid(String cellid) {
    this.cellid = cellid;
}
```

```java
public String getConninfo() {
    return this.conninfo;
}

public void setConninfo(String conninfo) {
    this.conninfo = conninfo;
}

public String getConntype() {
    return this.conntype;
}

public void setConntype(String conntype) {
    this.conntype = conntype;
}

public Integer getCqi() {
    return this.cqi;
}

public void setCqi(Integer cqi) {
    this.cqi = cqi;
}

public Timestamp getCtime() {
    return this.ctime;
}

public void setCtime(Timestamp ctime) {
    this.ctime = ctime;
}

public String getDeviceManufacturer() {
    return this.deviceManufacturer;
}

public void setDeviceManufacturer(String deviceManufacturer) {
    this.deviceManufacturer = deviceManufacturer;
}

public String getDeviceModel() {
    return this.deviceModel;
}

public void setDeviceModel(String deviceModel) {
    this.deviceModel = deviceModel;
}

public String getDeviceName() {
    return this.deviceName;
}
```

```java
public void setDeviceName(String deviceName) {
    this.deviceName = deviceName;
}

public Integer getDlBitrate() {
    return this.dlBitrate;
}

public void setDlBitrate(Integer dlBitrate) {
    this.dlBitrate = dlBitrate;
}

public String getEvent() {
    return this.event;
}

public void setEvent(String event) {
    this.event = event;
}

public String getLac() {
    return this.lac;
}

public void setLac(String lac) {
    this.lac = lac;
}

public BigDecimal getLat() {
    return this.lat;
}

public void setLat(BigDecimal lat) {
    this.lat = lat;
}

public Integer getLevel() {
    return this.level;
}

public void setLevel(Integer level) {
    this.level = level;
}

public String getLocationsource() {
    return this.locationsource;
}

public void setLocationsource(String locationsource) {
    this.locationsource = locationsource;
}
```

```
    public BigDecimal getLon() {
        return this.lon;
    }

    public void setLon(BigDecimal lon) {
        this.lon = lon;
    }

    public Integer getLterssi() {
        return this.lterssi;
    }

    public void setLterssi(Integer lterssi) {
        this.lterssi = lterssi;
    }

    public Integer getMaxping() {
        return this.maxping;
    }

    public void setMaxping(Integer maxping) {
        this.maxping = maxping;
    }

    @ApiModelProperty(notes="Mobile Country Code - it is a 3 digit code specific for each
country")
    public String getMcc() {
        return this.mcc;
    }

    public void setMcc(String mcc) {
        this.mcc = mcc;
    }

    public Integer getMinping() {
        return this.minping;
    }

    public void setMinping(Integer minping) {
        this.minping = minping;
    }

    public String getMnc() {
        return this.mnc;
    }

    public void setMnc(String mnc) {
        this.mnc = mnc;
    }

    public Integer getNcellid1() {
```

```
    return this.ncellid1;
}

public void setNcellid1(Integer ncellid1) {
    this.ncellid1 = ncellid1;
}

public Integer getNcellid2() {
    return this.ncellid2;
}

public void setNcellid2(Integer ncellid2) {
    this.ncellid2 = ncellid2;
}

public Integer getNcellid3() {
    return this.ncellid3;
}

public void setNcellid3(Integer ncellid3) {
    this.ncellid3 = ncellid3;
}

public Integer getNcellid4() {
    return this.ncellid4;
}

public void setNcellid4(Integer ncellid4) {
    this.ncellid4 = ncellid4;
}

public Integer getNcellid5() {
    return this.ncellid5;
}

public void setNcellid5(Integer ncellid5) {
    this.ncellid5 = ncellid5;
}

public Integer getNcellid6() {
    return this.ncellid6;
}

public void setNcellid6(Integer ncellid6) {
    this.ncellid6 = ncellid6;
}

public String getNetworkType() {
    return this.networkType;
}

public void setNetworkType(String networkType) {
```

```
    this.networkType = networkType;
}

public Integer getNlac1() {
    return this.nlac1;
}

public void setNlac1(Integer nlac1) {
    this.nlac1 = nlac1;
}

public Integer getNlac2() {
    return this.nlac2;
}

public void setNlac2(Integer nlac2) {
    this.nlac2 = nlac2;
}

public Integer getNlac3() {
    return this.nlac3;
}

public void setNlac3(Integer nlac3) {
    this.nlac3 = nlac3;
}

public Integer getNlac4() {
    return this.nlac4;
}

public void setNlac4(Integer nlac4) {
    this.nlac4 = nlac4;
}

public Integer getNlac5() {
    return this.nlac5;
}

public void setNlac5(Integer nlac5) {
    this.nlac5 = nlac5;
}

public Integer getNlac6() {
    return this.nlac6;
}

public void setNlac6(Integer nlac6) {
    this.nlac6 = nlac6;
}

public String getNode() {
```

```
    return this.node;
}

public void setNode(String node) {
    this.node = node;
}

public Integer getNrxlev1() {
    return this.nrxlev1;
}

public void setNrxlev1(Integer nrxlev1) {
    this.nrxlev1 = nrxlev1;
}

public Integer getNrxlev2() {
    return this.nrxlev2;
}

public void setNrxlev2(Integer nrxlev2) {
    this.nrxlev2 = nrxlev2;
}

public Integer getNrxlev3() {
    return this.nrxlev3;
}

public void setNrxlev3(Integer nrxlev3) {
    this.nrxlev3 = nrxlev3;
}

public Integer getNrxlev4() {
    return this.nrxlev4;
}

public void setNrxlev4(Integer nrxlev4) {
    this.nrxlev4 = nrxlev4;
}

public Integer getNrxlev5() {
    return this.nrxlev5;
}

public void setNrxlev5(Integer nrxlev5) {
    this.nrxlev5 = nrxlev5;
}

public Integer getNrxlev6() {
    return this.nrxlev6;
}

public void setNrxlev6(Integer nrxlev6) {
```

```
    this.nrxlev6 = nrxlev6;
}

public String getOperatorname() {
    return this.operatorname;
}

public void setOperatorname(String operatorname) {
    this.operatorname = operatorname;
}

public String getOs() {
    return this.os;
}

public void setOs(String os) {
    this.os = os;
}

public String getOsId() {
    return this.osId;
}

public void setOsId(String osId) {
    this.osId = osId;
}

public Integer getPingloss() {
    return this.pingloss;
}

public void setPingloss(Integer pingloss) {
    this.pingloss = pingloss;
}

public Integer getPsc() {
    return this.psc;
}

public void setPsc(Integer psc) {
    this.psc = psc;
}

public Integer getQual() {
    return this.qual;
}

public void setQual(Integer qual) {
    this.qual = qual;
}

public Integer getServingCellTime() {
```

```
    return this.servingCellTime;
}

public void setServingCellTime(Integer servingCellTime) {
    this.servingCellTime = servingCellTime;
}

public Integer getSnr() {
    return this.snr;
}

public void setSnr(Integer snr) {
    this.snr = snr;
}

public Integer getSpeed() {
    return this.speed;
}

public void setSpeed(Integer speed) {
    this.speed = speed;
}

public String getSsid() {
    return this.ssid;
}

public void setSsid(String ssid) {
    this.ssid = ssid;
}

public Integer getStdevping() {
    return this.stdevping;
}

public void setStdevping(Integer stdevping) {
    this.stdevping = stdevping;
}

public Integer getTestdlbitrate() {
    return this.testdlbitrate;
}

public void setTestdlbitrate(Integer testdlbitrate) {
    this.testdlbitrate = testdlbitrate;
}

public Integer getTestulbitrate() {
    return this.testulbitrate;
}

public void setTestulbitrate(Integer testulbitrate) {
```

```
        this.testulbitrate = testulbitrate;
    }

    public Timestamp getTimestamp() {
        return this.timestamp;
    }

    public void setTimestamp(Timestamp timestamp) {
        this.timestamp = timestamp;
    }

    public Integer getUlBitrate() {
        return this.ulBitrate;
    }

    public void setUlBitrate(Integer ulBitrate) {
        this.ulBitrate = ulBitrate;
    }

    public String getVersionName() {
        return this.versionName;
    }

    public void setVersionName(String versionName) {
        this.versionName = versionName;
    }

    public Campaign getCampaign() {
        return this.campaign;
    }

    public void setCampaign(Campaign campaign) {
        this.campaign = campaign;
    }

}
```

User Model

```
package gr.huatester.model;

import java.io.Serializable;
import javax.persistence.*;

import io.swagger.annotations.ApiModelProperty;


/**
 * The persistent class for the users database table.
 *
 */
@Entity
```

```java
@Table(name="users")
@NamedQuery(name="User.findAll", query="SELECT u FROM User u")
public class User implements Serializable {
    private static final long serialVersionUID = 1L;
    private String username;
    private String email;
    private int id;
    private String country;
    private String password;
    private String phone;



    @Override
    public String toString() {
        return "Name=" + this.username + ", Email=" + this.email + ", Country=" + this.country;
    }

    public String getUserName() {
        return username;
    }

    public void setUserName(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @ApiModelProperty(notes="The user's Country")
    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public String getPassword() {
```

```
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
```

Operators Model

```
package gr.huatester.model;

import java.io.Serializable;

import io.swagger.annotations.ApiModelProperty;


/**
 * The class for Operators.
 *
 */
public class Operators implements Serializable {
    private static final long serialVersionUID = 1L;
    private String operatorname;
    private long value;

    public Operators(String operatorname, long value) {
        super();
        this.operatorname = operatorname;
        this.value = value;
    }

    public String getOperatorname() {
        return operatorname;
    }
    public void setOperatorname(String operatorname) {
        this.operatorname = operatorname;
    }
    public long getValue() {
        return value;
    }
    public void setValue(long value) {
        this.value = value;
    }



}
```

OperatorStats Model

```java
package gr.huatester.model;

import java.io.Serializable;

import io.swagger.annotations.ApiModelProperty;


/**
 * The  class for Operator Statistics.
 *
 */
public class OperatorStats implements Serializable {
   private static final long serialVersionUID = 1L;
   private String operatorname;
   private int min;
   private int max;
   private double avg;

   public OperatorStats(String operatorname, int min, int max, double avg) {
      super();
      this.operatorname = operatorname;
      this.min = min;
      this.max = max;
      this.avg = avg;
   }

   public String getOperatorname() {
      return operatorname;
   }
   public void setOperatorname(String operatorname) {
      this.operatorname = operatorname;
   }
   public int getMin() {
      return min;
   }
   public int getMax() { return max; }
   public double getAvg() { return avg; }

   public void setMin(int min) {
      this.min = min;
   }
   public void setMax(int max) { this.max = max; }
   public void setAvg(double avg) { this.avg = max; }

}
```

Statistics Model

```java
package gr.huatester.model;

import java.io.Serializable;
```

```java
/**
* A persistent class to hold statistics key-value pairs
 */
public class Statistics implements Serializable {
   private static final long serialVersionUID = 1L;
   private String key;
   private long value;


   public Statistics(String key, long value) {
      super();
      this.key = key;
      this.value = value;
   }

   public String getKey() {
      return key;
   }

   public void setKey(String key) {
      this.key = key;
   }

   public long getValue() {
      return value;
   }

   public void setValue(int value) {
      this.value = value;
   }
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

W3C. (n.d.). Ανάκτηση 2017, από https://en.wikipedia.org/wiki/World_Wide_Web_Consortium

ACID. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/ACID

Angular2. (n.d.). *Angular*. Ανάκτηση 2017, από https://angular.io/

angular2-google-maps. (n.d.). Ανάκτηση 2017, από https://angular-maps.com/

AngularJS. (n.d.). *Angular*. Ανάκτηση 2017, από https://angularjs.org/

Apache Tomcat. (n.d.). Ανάκτηση 2017, από http://tomcat.apache.org/

Barnes Surface. (n.d.). Ανάκτηση 2017, από https://connect.boundlessgeo.com/docs/suite/enterprise/latest/cartography/rt/barnes.html

Chart.js. (n.d.). Ανάκτηση 2017, από http://www.chartjs.org/

CSS. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/Cascading_Style_Sheets

DBeaver. (n.d.). Ανάκτηση 2017, από http://dbeaver.jkiss.org/

Gatto, M. A. (2015). *Making Research Useful: Current Challenges.* Reuters Institute for the Study of Journalism.

Geoserver. (n.d.). Ανάκτηση 2017, από http://geoserver.org/

Gunelius, S. (2014). *The Data Explosion in 2014 Minute by Minute:Infographic.* Ανάκτηση 1 2017, από newstex.com: http://newstex.com/2014/07/12/the-data-explosion-in-2014-minute-by-minute-infographic/

Hibernate. (n.d.). Ανάκτηση 2017, από http://hibernate.org/

HTML. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/HTML

Intellij Idea. (n.d.). Ανάκτηση 2017, από https://www.jetbrains.com/idea/

Java. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/Java

Javascript. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/JavaScript

JSON. (n.d.). *json.org*. Ανάκτηση 2017, από http://www.json.org/

JSON Web Token. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/JSON_Web_Token

JSON. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/JSON

JWT. (n.d.). Ανάκτηση 2017, από https://jwt.io

Maven. (n.d.). Ανάκτηση 2017, από https://maven.apache.org/

MVC. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

NLG. (n.d.). Ανάκτηση 2017, από https://en.wikipedia.org/wiki/Natural_language_generation

Node.js. (n.d.). Ανάκτηση 2017, από https://nodejs.org/en/

Open Geospatial Consortium. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/Open_Geospatial_Consortium

Open Layers. (n.d.). Ανάκτηση 2017, από https://openlayers.org/

PHP. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/PHP

Postgis. (n.d.). Ανάκτηση 2017, από http://www.postgis.net/

PostgreSQL. (n.d.). Ανάκτηση 2017, από https://www.postgresql.org/

PostgreSQL . (n.d.). Ανάκτηση 2017, από https://www.postgresql.org/

RDBMS. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/Relational_database_management_system

Reactive programming. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/Reactive_programming

REST. (n.d.). *Wikipedia*. Ανάκτηση 2017, από https://en.wikipedia.org/wiki/Representational_state_transfer

Ruby on Rails. (n.d.). *Ruby*. Ανάκτηση 2017, από http://rubyonrails.org/

RxJS. (n.d.). *RxJS*. Ανάκτηση 2017, από https://github.com/Reactive-Extensions/RxJS

Science Daily. (2015). Big Data, for Better or Worse: 90% of World's Data. *Science Daily* .

Spring. (n.d.). Ανάκτηση 2017, από https://spring.io/

Spring Security. (n.d.). Ανάκτηση 2017, από https://projects.spring.io/spring-security/

Spring. (n.d.). *Spring Framework*. Ανάκτηση 2017, από https://spring.io/

Tomcat. (n.d.). Ανάκτηση 2017, από http://tomcat.apache.org/

Tomcat. (n.d.). Ανάκτηση 2017, από http://tomcat.apache.org/

Tomcat Manual. (n.d.). Ανάκτηση 2017, από https://tomcat.apache.org/tomcat-7.0-doc/setup.html

Typescript. (n.d.). Ανάκτηση 2017, από https://www.typescriptlang.org/

Webpack. (n.d.). Ανάκτηση 2017, από https://webpack.js.org/