# ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

## Πρόγραμμα Μεταπτυχιακών Σπουδών

## «ΠΜΣ ΠΛΗΡΟΦΟΡΙΚΗ»

**Μεταπτυχιακή Διατριβή**

| | |
|---|---|
| Τίτλος Διατριβής | **Υλοποίηση εφαρμογής restful web-service σε spring framework με οπτικοποίηση metrics σε prometheus και grafana.**<br><br>**Data processing by implementation of restful api web service in spring framework with metrics visualization in prometheus and grafana.** |
| Ονοματεπώνυμο Φοιτητή | **Ανδρέας Κρεούζος** |
| Πατρώνυμο | **Απόστολος** |
| Αριθμός Μητρώου | **ΜΠΠΛ20040** |
| Επιβλέπων | **Ευθύμιος Αλέπης, Αν. Καθηγητής** |

Ημερομηνία Παράδοσης **Νοέμβριος 2022**

**Τριμελής Εξεταστική Επιτροπή**

| | | |
|---|---|---|
| Ευθύμιος Αλέπης<br>Αναπληρωτής Καθηγητής | Μαρία Βίρβου<br>Καθηγήτρια | Κωνσταντίνος Πατσάκης<br>Αναπληρωτής Καθηγητής |

**Σύνδεσμοι Επικοινωνίας**

LinkedIn: www.linkedin.com/in/andreas-kreouzos

GitHub: https://github.com/Andreas-Kreouzos

# ΕΥΧΑΡΙΣΤΙΕΣ

Στο σημείο αυτό θα ήθελα να ευχαριστήσω θερμά τον αναπληρωτή καθηγητή κ. Ευθύμιο Αλέπη, για την ανάθεση και επίβλεψη της παρούσας διπλωματικής εργασίας, ο οποίος με το ενδιαφέρον του και τον απαράμιλλο ζήλο του βοήθησε στην εκπόνηση της.

Θα ήθελα επίσης να εκφράσω ιδιαίτερη μνεία στον κ. Χατζηγιαννάκη Νικόλαο, διευθυντή του καταστήματος Παλ. Φαλήρου της εταιρίας Β. Καυκάς Α.Ε, στον κ. Φράγκο Παναγιώτη, Energy Solutions Manager της εταιρίας Β. Καυκάς Α.Ε. καθώς και στους συναδέλφους μου, οι οποίοι με θυσίες, μου επέτρεψαν την συμμετοχή μου στο συγκεκριμένο πρόγραμμα μεταπτυχιακών σπουδών.

Ιδιαίτερη ευγνωμοσύνη θα ήθελα να εκφράσω προς την κα. Mary Grygleski, Streaming Developer Advocate, Java Champion και Chicago JUG President, με την οποία είχα την τιμή να συνομιλήσω μέσω βιντεοκλήσης για θέματα που αφορούσαν στο IoT input data.

Ανάλογη τιμή αισθάνομαι, μετά την συνομιλία μου με τον κ. Tom Donohue, Senior Field Engineer (Presales), Grafana Labs, για την πολύτιμη τεχνική του βοήθεια ως προς τις ρυθμίσεις της πλατφόρμας Prometheus.

Απεριόριστη χαρά και δέος, οφείλω στον κ. Mama Samba Braima Nelson Djalo, ιδρυτή της πλατφόρμας εκμάθησης software development Amigoscode.com, για την συνολική του υποστήριξη και συμβουλές στο έργο μου αυτό.

Τέλος, θα ήθελα να ευχαριστήσω ιδιαίτερα την οικογένεια μου για την ενθάρρυνση και την υποστήριξη τους και που χωρίς την βοήθεια των οποίων δεν θα είχα καταφέρει να πετύχω τους στόχους μου.

Κρεούζος Ανδρέας
Μάϊος 2022

# ΠΕΡΙΕΧΟΜΕΝΑ

# ΠΕΡΙΛΗΨΗ

Η δημιουργία του Διαδικτύου[1] αποτέλεσε το λίκνο για την επιστήμη της μετάδοσης της πληροφορίας, αρχής γενομένης στα τέλη της δεκαετίας του 60 με την δημιουργία του project ARPANET. Η εξέλιξη του τελευταίου καθώς και η δημιουργία του Παγκόσμιου Ιστού από τον Tim Berners Lee στα τέλη της δεκαετίας του 80, έφερε την επανάσταση, το μέγεθος της οποίας συνεχίζει να ανακαλύπτεται και σήμερα με νέες τεχνολογίες και υπηρεσίες στην διάθεση των ανθρώπων.

Αναμφισβήτητα το Διαδίκτυο θεωρείται ως το πλέον κοινωνικό αγαθό στις υπηρεσίες των πολιτών ενώ προβλέπονται ειδικές μέριμνες έτσι ώστε να γίνεται παντού προσβάσιμο. Ειδικά κονδύλια και επενδύσεις έχουν ήδη δοθεί και συνεχίζουν δίδονται από κυβερνητικούς και οργανισμούς τηλεπικοινωνιών, με σκοπό την ανάπτυξη σύγχρονων δικτύων υψηλών ταχυτήτων με κύρια τεχνολογία αυτής των οπτικών ινών κατ' οίκων (Fiber To The Home - FTTH).

Όλα τα ανωτέρω σε συνδυασμό με την είσοδο της ανθρωπότητας στην 4η Βιομηχανική Επανάσταση, η οποία συντελεί στην όλο και μεγαλύτερη ψηφιοποίηση των υπηρεσιών και των διαδικασιών, όπου διέπονται απλές και καθημερινές λειτουργίες του ανθρώπινου πληθυσμού, κατέστησε εφικτή την δημιουργία του Διαδικτύου των Πραγμάτων (IoT). Η ανάπτυξη αυτή, απαιτεί με την σειρά της, μια ακόμη μεγαλύτερη ροή πληροφοριών από και προς σε συσκευές, οι οποίες απεικονίζουν και επεξεργάζονται τα δεδομένα αυτά, με στόχο την έγκαιρη ενημέρωση των χρηστών, απρόσκοπτα και σε πραγματικό χρόνο.

Καταλυτικό ρόλο στην επίτευξη των στόχων της 4ης Βιομηχανικής Επανάστασης αλλά και του Διαδικτύου των Πραγμάτων, εκτός των πολλών διαφορετικών τεχνολογιών, έπαιξε και συνεχίζει να παίζει η ανάπτυξη των υπηρεσιών διαδικτύου μέσω των πρωτοκόλλων SOAP και REST. Υπηρεσίες με ιδιαίτερη βαρύτητα στην μετάδοση και επεξεργασία δεδομένων, με σκοπό την επίτευξη των προαναφερθέντων στόχων.

Στην παρούσα διατριβή, θα αποτυπωθεί επι τω πρακτέω η υλοποίηση μιας υπηρεσίας διαδικτύου με χρήση της γλώσσας προγραμματισμού JAVA υπό το πρωτόκολλο REST, κάνοντας εφαρμογή του πιο διαδεδομένου περιβάλλοντος δημιουργίας υπηρεσιών διαδικτύου, ήτοι του Spring Boot. Κύριος σκοπός μέσα από την υλοποίηση αυτή, είναι η δημιουργία μιας εφαρμογής, η οποία θα μπορεί να λαμβάνει στην είσοδο της, δεδομένα απ' οποιαδήποτε εξωτερική διεπαφή API, να τα επεξεργάζεται και να αποτυπώνει στην έξοδο της, αποτελέσματα που αφορούν το πεδίο ενδιαφέροντος των εκάστοτε χρηστών.

Υπό τις παραδοχές αυτές, η συγγραφή του παρόντος εγγράφου ξεκινά με το πρώτο κεφάλαιο να αναφέρεται σε εισαγωγικές έννοιες περί των υπηρεσιών JAVA WEB, με ιδιαίτερη έμφαση στα δύο πρωτοκόλλα που αναφέρθηκαν στις προηγούμενες παραγράφους. Θα πραγματοποιηθεί στοιχειοθέτηση ως προς την τελική χρήση του πρωτοκόλλου REST και εν συνεχεία, θα γίνει μια σύντομη εισαγωγή στο framework Spring και τα οφέλη που προκύπτουν από την χρήση του1. Για λόγους ευκολότερης πρόσβασης στην βιβλιογραφία της παρούσας διατριβής, θα γίνεται χρήση bookmarks τα οποία θα λειτουργούν ως υπερσύνδεσμοι προς την τελευταία.

Το δεύτερο κεφάλαιο συνθέτει την εγκαθίδρυση των στόχων του παρόντος έργου, αρχής γενομένης με την υλοποίηση μιας ανάλυσης απαιτήσεων. Κατά την διεξαγωγή τους, θα προκύψει

[1] Έχοντας ως δεδομένες τις έννοιες των πρωτοκόλλων REST και SOAP καθώς και του framework Spring, οι αναλύσεις θα είναι όσο το δυνατόν περιεκτικές βάσει της βιβλιογραφίας που έχει χρησιμοποιηθεί.

η ανάγκη για την κατανάλωση δύο εξωτερικών APIs. Μέσα από το συνονθύλευμα αυτό, θα κατασκευασθεί ένα διάγραμμα τύπου χάρτη, για την κατανόηση των βημάτων και της αλληλουχίας των συστατικών που θα απαρτίσουν το τελικό αποτέλεσμα.

Το τρίτο κεφάλαιο, αποτελεί το σημείο τομής μεταξύ της θεωρίας μέχρι και την τελική υλοποίηση, αφού εκεί καταγράφονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν. Θα δοθεί ιδιαίτερη βαρύτητα στην ασφάλεια της εφαρμογής, με υλοποίηση Spring Security σε πρωτόκολλο HTTPS και συστήματος αυθεντικοποίησης χρηστών μέσω συστήματος Email Verification. Σε ειδική ενότητα, θα αποτυπωθούν τα είδη των κλάσεων από τα οποία συνήθως αποτελείται ένα project Spring Boot και θα δοθούν επεξηγήσεις στα annotations που έλαβαν χώρα.

Στο τέταρτο κεφάλαιο, αναλαμβάνεται η κατασκευή DevOps διαδικασιών με τις πλατφόρμες Prometheus και Grafana, να έχουν πρωταγωνιστικό ρόλο. Θα δοθούν λεπτομερείς επεξηγήσεις ως προς την χρήση αυτών μέσω Docker και τα οφέλη που προκύπτουν με την αποτύπωση metrics και γραφικών dashboards. Η συνολική υλοποίηση έγινε μέσω WSL Ubuntu σε Windows 11 OS.

Τέλος στο πέμπτο κεφάλαιο, γίνεται η τελική χρήση των δύο API υπηρεσιών που καταναλώθηκαν εντός της εφαρμογής. Με εικόνες τύπου print screen δίνονται οι κατευθυντήριες γραμμές και η ακολουθία των HTML σελίδων πάνω στις οποίες απεικονίζονται τα καταναλισκόμενα αποτελέσματα από τις δύο APIs.

# EXECUTIVE SUMMARY

The creation of the Internet[1] has been identified as the cradle of the science of information transmission, beginning in the late 1960s with the creation of the ARPANET project. The evolution of the latter as well as the creation of the World Wide Web by Tim Berners Lee in the late 80's, brought the revolution, the magnitude of which continues to be discovered today with new technologies and services available to the people.

Undoubtedly, the Internet is considered as the most social good in the services of the citizens while special arrangements are provided so that it becomes accessible everywhere. Special funds and investments have already been given and continue to be given by government and telecommunications organizations, with the aim of developing modern high-speed networks with the main technology of this, the fiber optics at home (Fiber to the Home - FTTH).

All of the above, combined with the entry of humanity into the Industry 4.0 era, which contributes to the increasing digitization of services and processes governed by simple and daily operations of the human population, made possible the creation of the Internet of Things (IoT). This development, in turn, requires an even greater flow of information to and from devices that display and process this data, with the aim of informing users in a timely manner, seamlessly and in real time.

Apart from the many different technologies, the development of internet services through the SOAP and REST protocols has played and continues to play a catalytic role in achieving the goals of the Industry 4.0 and the Internet of Things. Services with special emphasis on data transmission and processing, in order to achieve the above objectives.

In this dissertation, the implementation of a web service using the JAVA programming language under the REST protocol will be practically take place, by using the most widespread framework, namely the Spring Boot. The main purpose of this implementation is to create an application, which will be able to receive in its input data from any type of IoT sensor, then process them and capture the output results that concern the field of interest of each user.

With these assumptions, the writing of this document begins with the first chapter referring to introductory concepts about JAVA WEB services, with particular emphasis on the two protocols mentioned in the previous paragraphs. The end-use of the REST protocol will be concluded and then there will be a brief introduction to the Spring framework and the benefits of using it[2]. To accommodate the access to the bibliography of this dissertation, bookmarks will be used that will act as hyperlinks to the latter.

The second chapter synthesizes the establishment of the objectives of the present project, starting with the implementation of a requirements analysis. During their execution, the need to consume two external APIs will arise. Through this patchwork, a map-like diagram will be constructed, to understand the steps and sequence of components that will make up the final result.

The third chapter is the point of intersection between the theory and the final implementation, since the technologies and tools used, are recorded there. Special emphasis will be given to the

[2] Given the fact that REST and SOAP protocols as well as Spring Framework, are already known in the science of Information Technology, the references will be as comprehensive as possible based on the bibliography used to create this dissertation.

security of the application, with the implementation of Spring Security in HTTPS protocol and user authentication through Email Verification system. In a special section, the types of classes, that a Spring Boot project usually consists of, will be captured and the annotations that took place will be explained.

In the fourth chapter, the construction of DevOps processes is undertaken with the Prometheus and Grafana platforms having a leading role. Detailed explanations will be given as to the usage of these through Docker and the benefits that arise from capturing metrics and graphic dashboards. The overall implementation was done via WSL Ubuntu on Windows 11 OS.

Finally, in the fifth chapter, the final usage of the two API services consumed within the application is made. The guidelines and the sequence of the HTML pages, on which the consumed results from the two APIs are displayed, are given with print screen images.

# ΛΙΣΤΑ ΠΙΝΑΚΩΝ

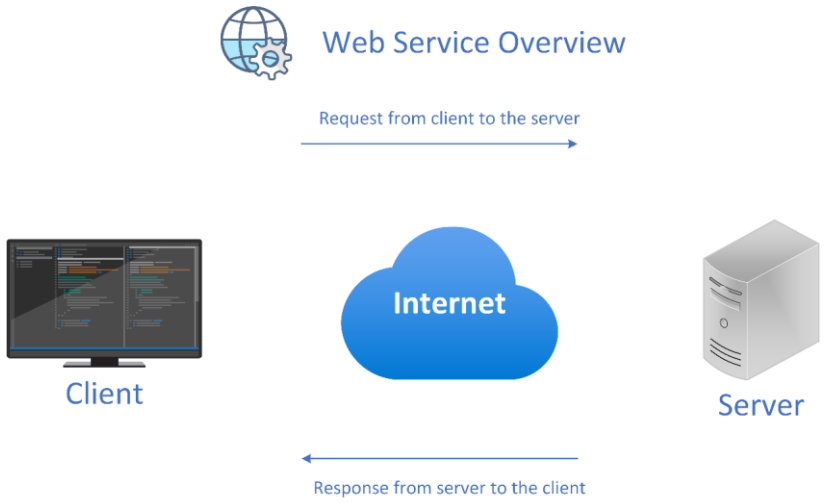
# ΛΙΣΤΑ ΕΙΚΟΝΩΝ

# 1 ΕΙΣΑΓΩΓΉ ΣΤΙΣ ΥΠΗΡΕΣΙΕΣ JAVA WEB

Όπως αναφέρθηκε και στην περίληψη του παρόντος εγγράφου, σκοπός του παρόντος κεφαλαίου είναι η εισαγωγή σε συνοπτικές έννοιες των τεχνολογιών που θα χρησιμοποιηθούν στα επόμενα κεφάλαια της παρούσας διατριβής. Οι έννοιες αυτές θα αναλωθούν στις υπηρεσίες web ως οντότητα, στα πρωτόκολλα SOAP – REST και τέλος στο framework Spring.

## 1.1 ΟΙ ΥΠΗΡΕΣΊΕΣ WEB (WEB SERVICES)[2] , [3]

Μια απλή αναζήτηση στο Διαδίκτυο αρκεί για να εντοπίσει κανείς, ότι ο ορισμός μιας υπηρεσίας διαδικτύου δεν είναι πολύ ακριβής. Ο τελευταίος κρύβει πολλές έννοιες με αντίστοιχες σύνθετες παραμέτρους, ωστόσο για λόγους ορθότητας θεωρείται συνετό να αναφερθεί ο ορισμός Web Service ως ένα οποιοδήποτε τμήμα λογισμικού το οποίο είναι διαθέσιμο μέσω του διαδικτύου, σκοπός ύπαρξης του οποίου είναι η επίλυση διαφόρων εφαρμογών των χρηστών. Επί της ουσίας είναι ένα κατανεμημένο σύστημα λογισμικού, το οποίο παραδίδεται μέσω του πρωτοκόλλου HTTP και υπό καθαρά τεχνική έννοια, μπορεί να εκτελεστεί σε φυσικές διακριτές συσκευές που βρίσκονται συνδεδεμένες σε δίκτυο.




**Εικόνα 1. Επισκόπηση μιας υπηρεσίας διαδικτύου (Πηγή: Δημιουργία στο Microsoft Visio[12])**

Η τελευταία θεώρηση, θα μπορούσε να αναπτυχθεί ακόμα περισσότερο, αναφέροντας την υπηρεσία διαδικτύου, ως το σταθερό μέσο το οποίο βοηθά στην αποτελεσματική επικοινωνία μεταξύ ενός πελάτη (client) και μιας εφαρμογής, η οποία βρίσκεται σε κάποιον διακομιστή (server) στο διαδίκτυο.

Η εικόνα 1 αποτυπώνει την διαπίστωση αυτή, στην πιο απλή της μορφή καθώς μεγάλο εύρος των υπηρεσιών διαδικτύου διαθέτουν πιο περίπλοκες δομές από την προαναφερθείσα. Παράδειγμα μιας περίπλοκης δομής υπηρεσίας διαδικτύου, θα ήταν όταν αυτή έχει περισσότερους του ενός πελατών, οι οποίοι δημιουργούν τα αιτήματα προς αυτήν, την στιγμή που η τελευταία αποτελείται από ακόμη περισσότερες υπηρεσίες, για να παράσχει το τελικό αποτέλεσμα.

Όπως αναφέρθηκε σε προηγούμενη παράγραφο, η λειτουργία μιας υπηρεσίας διαδικτύου υλοποιείται μέσω του πρωτοκόλλου HTTP, κατά συνέπεια το τελευταίο αποτελεί την μέθοδο επικοινωνίας και αλληλεπίδρασης μεταξύ του πελάτη με τον διακομιστή μέσω μηνυμάτων τύπου request. Η ίδια φιλοσοφία ακολουθείται όταν πλέον ο διακομιστής είναι έτοιμος να απαντήσει στο αίτημα του πελάτη, με αντίστοιχο μήνυμα τύπου response. Ο συγκεκριμένος τρόπος επικοινωνίας τύπου request / response ανήκει σε μια ευρύτερη κατηγορία αλληλεπιδράσεων που λαμβάνουν χώρα στις υπηρεσίες διαδικτύου και θεωρείται ο πιο κοινός μεταξύ αυτών.

| Κατηγορίες Αλληλεπιδράσεων | Επεξηγήσεις |
|---|---|
| Request / Response | Μήνυμα τύπου request από τον πελάτη το οποίο αναμένει απάντηση τύπου response από τον διακομιστή |
| Solicit / Response | Εκκίνηση μηνύματος (Solicit) πρώτα από τον server, το οποίο αναμένει μήνυμα από τον πελάτη (Response) |
| One-Way | Αποστολή μηνύματος από τον πελάτη χωρίς την απαίτηση απάντησης από τον διακομιστή |
| Notification | Αποστολή μηνύματος από τον διακομιστή χωρίς την απαίτηση απάντησης από τον πελάτη |

**Πίνακας 1. Κατηγορίες αλληλεπιδράσεων μηνυμάτων μεταξύ πελάτη - διακομιστή**

Ο πίνακας 1 αποτυπώνει συγκεντρωτικά τους τρόπους επικοινωνίας που διέπουν τις πλέον σύγχρονες υπηρεσίες διαδικτύου.

### 1.1.1 Υπηρεσίες Διαδικτύου μέσω πρωτοκόλλου SOAP[8]

Ο συγκεκριμένος τύπος υπηρεσίας διαδικτύου χρησιμοποιεί την λογική της ανεξάρτητης μεταφοράς μηνυμάτων πρωτοκόλλου και είναι προϊόν της Microsoft. Το συγκεκριμένο πρωτόκολλο χρησιμοποιεί την γλώσσα XML υπό συγκεκριμένη και στοιχειοθετημένη γραμματική, της οποίας η ύπαρξη είναι απαραίτητη προκειμένου το αρχείο που την εμπεριέχει, να αναγνωρίζεται ως συμβατό με το πρωτόκολλο SOAP. Το αρχείο αυτό αποτελεί το μήνυμα του πρωτοκόλλου και εμπεριέχει τα δεδομένα στην προαναφερθείσα γλώσσα με την μετάδοση των μηνυμάτων να γίνεται υπό την βοήθεια των HTML και SMTP. Επίσης οι υπηρεσίες αυτές, διαθέτουν εσωτερικό σύστημα χειρισμού λαθών που μπορεί να προκύψουν κατά την διαδικασία μετάδοσης, όπου κατά την αποστολή των απαντήσεων, εμπεριέχεται και η πληροφορία λάθους για τον εντοπισμό των τελευταίων και την επίλυση τους.

Ένα από τα βασικά πλεονεκτήματα της υπηρεσίας αυτής είναι ότι χαρακτηρίζεται ως γλώσσα ανοικτού τύπου, το οποίο χαρακτηριστικό βοηθά στην διαλειτουργικότητα μεταξύ των διαφόρων υπηρεσιών καθότι ερμηνεύεται από οποιαδήποτε γλώσσα προγραμματισμού. Ακόμα σημαντικό πλεονέκτημα, έγκειται στον τομέα της ασφάλειας με το πρωτόκολλο αυτό να κάνει χρήση των SSL (Secure Socket Layer) και WS-Security.

Η βασική του ιδέα κατά την διάρκεια της δημιουργίας του πρωτοκόλλου SOAP ήταν η διασφάλιση της ασφαλούς ανταλλαγής δεδομένων μεταξύ προγραμμάτων τα οποία έχουν κατασκευαστεί σε διαφορετικές πλατφόρμες και γλώσσες προγραμματισμού.

### 1.1.2 Υπηρεσίες Διαδικτύου μέσω αρχιτεκτονικής REST[4], [8]

Ο συγκεκριμένος τύπος υπηρεσίας διαδικτύου δεν ανήκει στην κατηγορία κάποιου πρωτοκόλλου αλλά είναι στυλ λογισμικής αρχιτεκτονικής που σχεδιάστηκε ως οδηγός για την ανάπτυξη και δομή συστημάτων υπερμέσων όπου τα κείμενα, οι γραφικές αποτυπώσεις, ήχοι και βίντεο αποθηκεύονται σε κάποιο δίκτυο και ενώνονται με την βοήθεια υπερσυνδέσμων. Παράδειγμα που ακολουθεί αυτή τη μορφή είναι ο Παγκόσμιος Ιστός.

Δημιουργός του είναι ο περίφημος Roy Fielding, ο οποίος και το πρωτοανέφερε στην διδακτορική του διατριβή με τίτλο «Architectural Styles and the Design of Network-based Software Architectures» το 2000. Η συγκεκριμένη αρχιτεκτονική θεωρείται σημείο αναφοράς στην κοινότητα δημιουργίας λογισμικού περί ανάπτυξης αξιόπιστων εφαρμογών διαδικτύου.

Εν αντιθέσει με το πρωτόκολλο SOAP που χρησιμοποιεί μόνο την γλώσσα XML, η αρχιτεκτονική REST κάνει χρήση εκτός της προαναφερθείσας, και απλού κειμένου, HTML καθώς

και της δομής JSON. Σε περιπτώσεις που είναι απαραίτητο η αρχιτεκτονική REST μπορεί να χρησιμοποιήσει το πρωτόκολλο SOAP αλλά όχι το αντίθετο. Βασικό της πλεονέκτημα είναι η κατανάλωση λιγότερων πόρων και εύρους ζώνης και η παροχή μεγαλύτερης ευελιξίας σε εφαρμογές που έχουν γραφτεί σε διαφορετικές γλώσσες προγραμματισμού. Ο χάρτης υλοποίησης της αρχιτεκτονικής REST ακολουθεί τους παρακάτω κανόνες:

Πόροι: Γίνεται εντολή στον διακομιστή διαδικτύου να παράσχει όλες τις πληροφορίες που απαιτούνται.

Εντολές Ρημάτων: Οι τελευταίες περιγράφουν οτιδήποτε επιθυμεί ο χρήστης να κάνει με τους συγκεκριμένους πόρους.

Επικεφαλίδες: Οι τελευταίες στέλνονται ως επιπρόσθετες οδηγίες μαζί με τις απαιτήσεις των χρηστών για να καθορίσουν τον τύπο της απαίτησης.


**Εικόνα 2. Λογότυπο REST (Πηγή: Δημιουργία στο Microsoft Visio[12])**

- Σώμα Απαίτησης: Το σώμα της απαίτησης περιέχει όλες τις απαραίτητες πληροφορίες ενός συγκεκριμένου πόρου, οι οποίες πρέπει να προστεθούν στον διακομιστή.
- Σώμα Απάντησης: Εδώ παρατίθεται το κυρίως σώμα της απάντησης που λαμβάνεται από τον διακομιστή.
- Κωδικοί Καταστάσεων Απάντησης: Οι κωδικοί αυτοί επιστρέφονται μαζί με την απάντηση από τον διακομιστή για να δώσουν την ισχύουσα δεδομένη κατάσταση που λαμβάνει χώρα. Παρακάτω παρατίθεται ο πίνακας 2 με τους κωδικούς κατάστασης που λαμβάνουν χώρα.

| Κωδικός | Περιγραφή | Επεξήγηση |
|---|---|---|
| 200 | OK | Αίτημα OK |

| | | |
|---|---|---|
| 303 | See Other | Ανακατεύθυνση |
| 400 | Bad Request | Εσφαλμένο Αίτημα |
| 401 | Unauthorized | Σφάλμα εξουσιοδότησης |
| 403 | Forbidden | Άρνηση Αιτήματος |
| 404 | Not Found | Μη Ευρισκόμενος Πόρος |
| 405 | Method Not Allowed | Μη υποστηριζόμενη μέθοδος |
| 415 | Unsupported Media Type | Μη αναγνωρίσιμος τύπος |
| 500 | Internal Server Error | Διαδικασία αιτήματος απέτυχε |

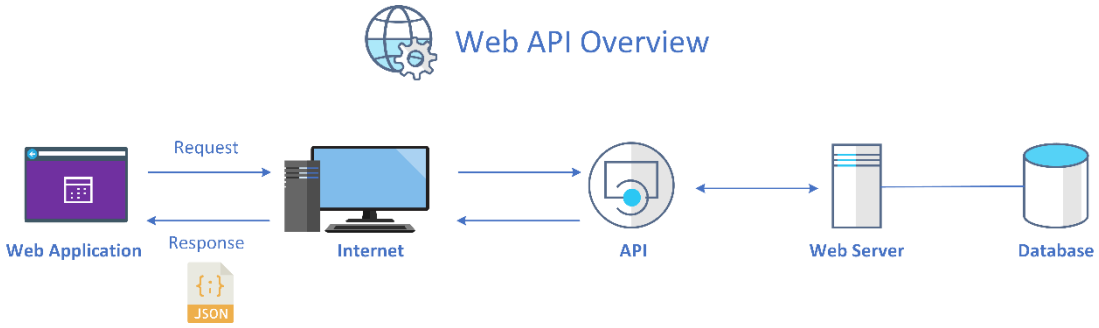
**Πίνακας 2. Κωδικοί Καταστάσεων HTTP**

### 1.1.3 Application Programming Interface (API)[7] , [9]

Ο τίτλος της παρούσας ενότητας μαρτυρά την αναφορά στον ορισμό μιας διεπαφής προγραμματισμού εφαρμογών (API) σε ελεύθερη ελληνική μετάφραση, η οποία επιτελεί παρόμοια, αν όχι σχεδόν ίδια διαδικασία μ αυτή μιας εφαρμογής διαδικτύου. Με μια απλή αναζήτηση στο Διαδίκτυο, εύκολα μπορεί να εντοπισθεί ότι αυτή ορίζεται ως η σύνδεση μεταξύ υπολογιστών με σκοπό την επικοινωνία μιας εφαρμογής με μία άλλη. Επί της ουσίας αποτελεί τον μηχανισμό για την αλληλεπίδραση μεταξύ δύο εφαρμογών κάνοντας χρήση συγκεκριμένων κανόνων.

Η υλοποίηση των διεπαφών προγραμματισμού εφαρμογών αποτελούνται από δύο κύριους βασικούς άξονες. Ο πρώτος αφορά τις τεχνικές προδιαγραφές που περιγράφουν την ανταλλαγή των δεδομένων μεταξύ των λογισμικών υπό τη μορφή αίτησης πρωτοκόλλων μετάδοσης αυτών. Ο δεύτερος αφορά την συγγραφή διεπαφής προς τις τεχνικές προδιαγραφές που θα την εκπροσωπούν.

Με την υλοποίηση των ανωτέρω, το λογισμικό που επιθυμεί την πρόσβαση σε δεδομένα ή λειτουργίες ενός άλλου λογισμικού, θα πραγματοποιήσει κλήση της διεπαφής API του, δίνοντας ταυτόχρονα τις προδιαγραφές με τις οποίες τα δεδομένα θα πρέπει να του παρασχεθούν πίσω. Το δεύτερο λογισμικό θα απαντήσει με την πρόσδοση των ζητούμενων δεδομένων από την αρχική εφαρμογή βάσει των προδιαγραφών που τέθηκαν. Η διεπαφή με την οποία γίνεται η επικοινωνία των δύο εφαρμογών είναι αυτή που καθορίζεται από την API.




**Εικόνα 3. Επισκόπηση ενός API διαδικτύου (Πηγή: Δημιουργία στο Microsoft Visio[12])**

Ο μηχανισμός αυτός προσδίδει μεγάλά πλεονεκτήματα στους μηχανικούς ανάπτυξης λογισμικού καθότι τους δίνει την δυνατότητα να προσθέσουν αυξημένου τύπου λειτουργικότητες στις εφαρμογές, αποφεύγοντας την συγγραφή μεγάλου όγκου κώδικα. Εκτός αυτού παρέχουν επίσης την δυνατότητα πρόσβασης σε δεδομένα διαμέσου άλλων εφαρμογών. Ένα παράδειγμα της συγκεκριμένης λειτουργικότητας είναι η ενσωμάτωση περιεχομένου από εφαρμογές κοινωνικής δικτύωσης σε ιστοσελίδες διαφορετικού περιεχομένου και ενδιαφέροντος, ως widgets.

### 1.1.4 Υπηρεσία Διαδικτύου ως REST API[5] , [6]

Από την σύντομη ανάλυση της προηγούμενης ενότητας, εύκολα θα μπορούσε να προκύψει σύγχυση ως προς το ποια είναι η διαφορά που διέπει τις εφαρμογές διαδικτύου από τις διεπαφές προγραμματισμού εφαρμογών. Πράγματι και τα δύο είδη ως υπηρεσίες παρέχουν επικοινωνία μεταξύ υπολογιστών και εφαρμογών, ωστόσο όπως αναφέρθηκε στην εισαγωγή της παραγράφου 1.1, οι υπηρεσίες διαδικτύου υλοποιούνται αποκλειστικά μέσω σύνδεσης στο διαδίκτυο. Η τελευταία θεώρηση, δεν ισχύει πάντοτε για τα APIs, τα οποία μπορούν να εκτελεσθούν και μέσω τοπικών αρχείων (π.χ. αρχείο τύπου JAR ενός προγράμματος σε γλώσσα Java), για να επιτρέψουν την επικοινωνία μεταξύ δύο εφαρμογών στον ίδιο τοπικό υπολογιστή.

Με την αποτύπωση των εννοιών μεταξύ των Web Services και των API και την διαπίστωση περί της διαφοράς τους, εύκολα μπορεί να βγει το συμπέρασμα ότι ένα API αποτελεί την προέκταση ενός Web Service. Επίσης από την στιγμή που οι τύποι υλοποίησης των Web Services ανάγονται στο πρωτόκολλο SOAP και στην αρχιτεκτονική REST, τότε προκύπτουν υπηρεσίες διαδικτύου είτε τύπου SOAP API είτε REST API.

Ειδικά για την υπηρεσία Web REST API, ενσωματώνονται όλες οι λειτουργίες που αναφέρθηκαν στις προηγούμενες παραγράφους, δηλαδή η υλοποίηση αιτημάτων μέσω HTTP μεθόδων διαμέσου εντός δικτύου. Για την πραγμάτωση των αιτημάτων αυτών, ο πελάτης (client) καλείται να επιτελέσει δύο παραμέτρους:

1. Ονοματίζει την στοχευμένη πηγή απ' την οποία θα αντλήσει την πληροφορία που ζητά, δίνοντας το URI της, υπό την μορφή του συνδέσμου URL.
2. Καθορίζει ένα ρήμα στην μέθοδο HTTP, το οποίο δίνει την εντολή που ο τελευταίος επιθυμεί να επιτελέσει. Οι πιο γνωστές εντολές αφορούν στην ανάγνωση περιεχομένου, στην δημιουργία νέου, στην μορφοποίηση υπάρχοντος καθώς και στην διαγραφή αυτού.

Τα ρήματα αυτά και κατ' επέκταση οι εντολές που δίνουν, υπακούν στο πρότυπο λειτουργιών CRUD (Create, Read, Update, Delete), το οποίο είναι αναπόσπαστό κομμάτι των RESTful Web Services και που αναλύεται στον επόμενο πίνακα.

| HTTP Ρήμα Εντολής | CRUD Λειτουργία |
|---|---|
| POST | Δημιουργία |
| GET | Ανάγνωση |
| PUT | Μορφοποίηση |
| DELETE | Διαγραφή |

**Πίνακας 3. Επεξήγηση βασικών HTTP εντολών με τις λειτουργίες τους**

Η δομή των RESTful Web Services, έγκειται στο γεγονός ότι το πρωτόκολλο HTTP, παρά την σημασία στην έννοια της «μεταφοράς» που περικλείεται στο όνομα του, δρα ως διεπαφή προγραμματισμού εφαρμογών (API) και όχι ως απλό μεταφορικό μέσο πληροφορίας. Αξίζει να σημειωθεί ότι οι εντολές του πίνακα 3, είναι οι πιο συνήθεις και βασικές στην υλοποίηση RESTful API Web Services. Επίσης ισχύουν και εδώ, οι κωδικοί κατάστασης που αναλύθηκαν στον Πίνακας 2.

## 1.2 Spring Framework[10]

Το κύριο μέρος της παρούσας διατριβής, θα εστιασθεί στο framework Spring, η ηλικία του οποίου πλησιάζει τις δύο δεκαετίες, την περίοδο συγγραφής του εγγράφου αυτού, και που το οποίο πλέον αποτελεί σημείο αναφοράς για την ανάπτυξη Java εφαρμογών.

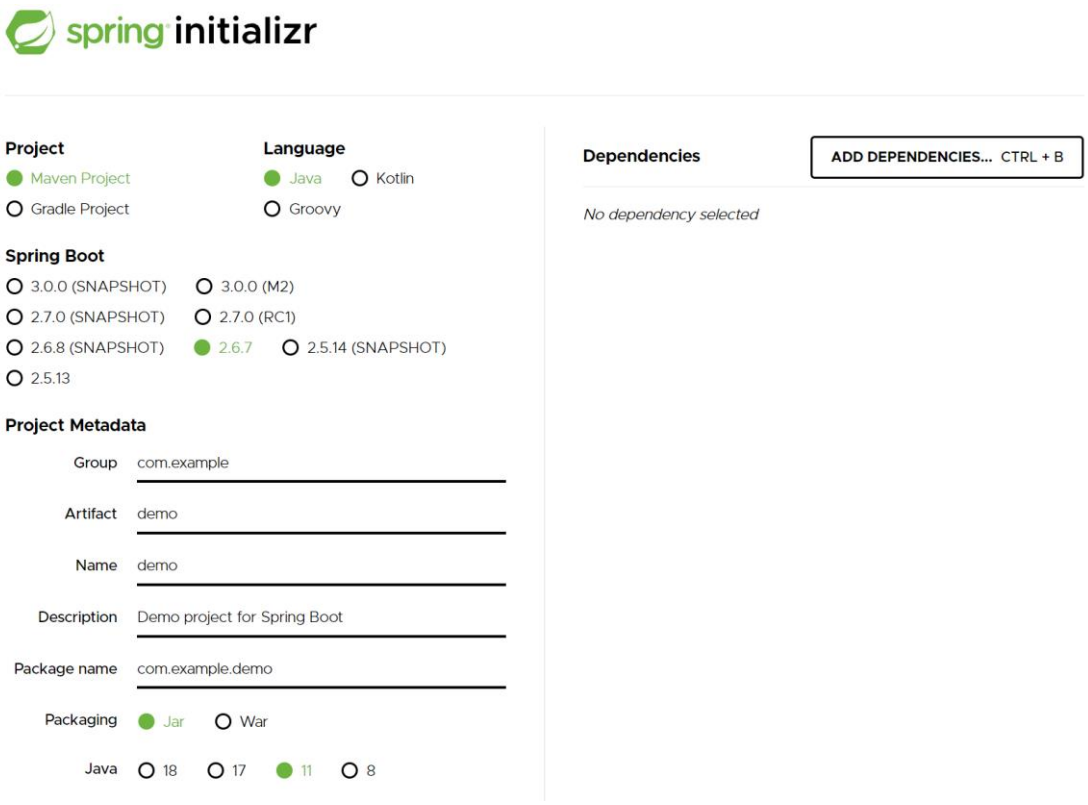
### 1.2.1 Εισαγωγή

Το Spring Framework, παρέχει τα απαραίτητα εργαλεία και την υποστήριξη υποδομής για την ανάπτυξη εφαρμογών σε γλώσσα προγραμματισμού Java. Η αρχική του μορφή παρουσιάστηκε στο βιβλίο «Expert One-on-One J2EE Design and Development» του συγγραφέα Rod Johnson, το έτος 2002 και αποτέλεσε την βάση για την περαιτέρω ανάπτυξη των web services καθώς και των microservices που προορίζονται για cloud εφαρμογές με διατήρηση εγγραφών σε μια πληθώρα βάσεων δεδομένων.


**Εικόνα 4. Λογότυπο Spring Framework**

Στον βασικό του πυρήνα το Spring παρέχει την κύρια δομή του, γνωστή ως Spring Application Context, μέσα στην οποία μπορούν να δημιουργηθούν και να διαχειριστούν όλα τα συστατικά (components), τα οποία απαρτίζουν το σύνολο της εφαρμογής κατά Spring. Η επιλογή των, γίνεται από τον εκάστοτε προγραμματιστή λογισμικού, με πολλούς τρόπους με πιο κύρια αυτού

της εφαρμογής Spring Initializr, και κατόπιν ώριμων σκέψεων και σχεδιασμού της εφαρμογής. Η τελευταία θεώρηση εναπόκειται στο γεγονός, ότι πρέπει να έχει προηγηθεί μια σωστή καταγραφή των αναγκών, μέσω ανάλυσης απαιτήσεων, για την ανάπτυξη της τελικής εφαρμογής και βάσει αυτών θα γίνει η επιλογή των components.




**Εικόνα 5. Αρχική σελίδα Spring Initializr (Πηγή: https://start.spring.io/)**

Τα τελευταία στην γλώσσα του Spring Framework ονομάζονται beans και η μεταξύ τους επικοινωνία είναι απαραίτητη για την σύσταση της τελικής εφαρμογής. Αυτό το είδος επικοινωνίας αναφέρεται στην βιβλιογραφία ως wiring ενώ η δράση με την οποία ο προγραμματιστής θα την πράξει ονομάζεται ως dependency injection[13].

Υπό την έννοια dependency injection, νοείται η τεχνική σχεδίασης λογισμικού κατά την οποία, ένα αντικείμενο μπορεί να λάβει άλλα αντικείμενα από τα οποία εξαρτάται η ύπαρξη του και η λειτουργία του. Η τεχνική διασφαλίζει την λογική της χρήσης υπηρεσιών από ένα αντικείμενο, χωρίς να απαιτείται το τελευταίο να γνωρίζει πως θα τις κατασκευάσει. Το τελευταίο

επιτυγχάνεται με αποσύνδεση της χρήσης του αντικείμενου από αυτή της δημιουργίας του και αυστηρά μέσα στο Spring Application Context το οποίο λειτουργεί ως container διασφάλισης όλων των λειτουργιών που απαρτίζουν την τελική εφαρμογή.

Ο τελευταίος ορισμός αποτελεί έναν απο τους ακρογωνιαίους λίθους στην ανάπτυξη λογισμικού, διότι:

- Βοηθά στην ανεξαρτητοποίηση μιας κλάσης από την διαδικασία δημιουργίας των αντικειμένων, απ' τα οποία εξαρτάται.
- Στην διαδικασία υποστήριξης διαφορετικών configurations από μία εφαρμογή και των αντικειμένων της.
- Στην αλλαγή των εξαγόμενων αποτελεσμάτων ενός τμήματος κώδικα, χωρίς την άμεση τροποποίηση αυτού.

Έχοντας λοιπόν, ως θεμέλιο για την ανάπτυξη της τελικής εφαρμογής, το Spring Application Context αλλά και επιμέρους βιβλιοθήκες, το framework προσφέρει λειτουργίες όπως: Ενσωμάτωση τομέα ασφάλειας της εφαρμογής, ενοποιήσεις με άλλα συστήματα, υποστήριξη microservices και μοντέλων αντιδραστικού προγραμματισμού, τα οποία είναι απαραίτητα στην ανάπτυξη σύγχρονων εφαρμογών.

### 1.2.2 Spring Boot[11]

Με την ανάπτυξη των microservices, η ομάδα ανάπτυξης του framework Spring, προέβη στην απλοποίηση αυτού εισάγοντας στην σκακιέρα το Spring Boot. Το τελευταίο αποτελεί μια επέκταση του framework Spring, το οποίο προσδίδει τροποποιήσεις και απλούστευση στην διαδικασία ανάπτυξης μιας εφαρμογής, με λιγότερη συγγραφή κώδικα.

**Εικόνα 6. Λογότυπο Spring Boot**

Κάνοντας χρήση ειδικών modules, διαδικασίες όπως η εισαγωγή δεδομένων σε μια αντίστοιχη βάση, απαιτούν μόλις λίγες γραμμές κώδικα, την στιγμή που η ίδια διαδικασία πριν την δημιουργία του Spring Boot, θα χρειάζονταν πολύ περισσότερο χρόνο και συγγραφή του τελευταίου για να υλοποιηθεί. Σε επόμενο κεφάλαιο, θα αναλυθεί η δομή του project και οι ευκολίες που παρέχονται από το ίδιο το framework στο στήσιμο του.

Το Spring Framework και ειδικά το Spring Boot, αποτελούν τα σημείο τομής στους ανθρώπους που απασχολούνται ως developers και θεωρούνται αναπόσπαστα μέρη στην ανάπτυξη εφαρμογών. Δεν είναι τυχαίο που η επιλογή του framework αυτού, έγινε από την εταιρία Netflix για την υποστήριξη των back-end συστημάτων της.

# 2 ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ – ΧΑΡΤΗΣ ΕΦΑΡΜΟΓΗΣ

Έχοντας θέσει τις βάσεις και τους ορισμούς των βασικών τεχνολογιών και των υλικών συσκευών που συνθέτουν το περιεχόμενο και αντικείμενο του παρόντος εγγράφου, σειρά έχει η αποτύπωση του σκοπού και των λειτουργιών της εφαρμογής που θα αναπτυχθεί με σκοπό την πρόσδοση πολύτιμης πληροφορίας προς τους τελικούς χρήστες. Εν προκειμένω, θα γίνει μια σύλληψη απαιτήσεων που θα αναγνωρίζει τις ανάγκες για την δημιουργία και τους λόγους ύπαρξης του τελικού προϊόντος.

Εν συνεχεία, θα γίνει μια γραφική αποτύπωση της εφαρμογής που θα λάβει χώρα. Με το όρο «γραφική», νοείται η δημιουργία και αναπαράσταση ενός γραφήματος, το οποίο θα απεικονίσει τα διάφορα επίπεδα (layers) της τελικής εφαρμογής, τα οποία σε υποενότητες θα εξειδικευτούν, έτσι ώστε να δοθούν όλες οι απαραίτητες πληροφορίες του τρόπου δράσης των και στο τι εξυπηρετεί το καθένα απ' αυτά.

Επίσης, σε ενότητα που θα ακολουθήσει, θα δοθεί ένας «χάρτης», η εξαγωγή του οποίου θα γίνει αφού ολοκληρωθεί, η διαδικασία της ανάλυσης απαιτήσεων. Το συγκεκριμένο γράφημα, θα αποτελέσει την ραχοκοκαλιά που θα ακολουθηθεί για την ολοκλήρωση του πρακτικού μέρους της εφαρμογής. Σαφώς, θα γίνει πλήρης επεξήγηση αυτού με βήματα, για την καλύτερη αποτύπωση των λειτουργιών και της αλληλοεπίδρασης των χρηστών με την εφαρμογή.

Τέλος, για λόγους οικειότητας και μεγαλύτερης ευκολίας, θα γίνει χρήση της Rational Unified Process, η οποία διδάχθηκε κατά το Γ’ Εξάμηνο του ΠΜΣ.

## 2.1 Σκοπός της Εφαρμογής

Στις επόμενες δύο υποενότητες, δίνεται ο στόχος της εφαρμογής που θα κατασκευασθεί, παραθέτοντας δύο σενάρια, διαφορετικά μεταξύ τους, που απαρτίζουν την καθημερινότητα των χρηστών. Οι εφαρμογές αυτές, οι οποίες επιλέχθηκαν από τον συγγραφέα φοιτητή, έγιναν με βάση καθαρά αρέσκειας αλλά και γεγονότων, καταστάσεων και συνηθειών που λαμβάνουν χώρα στην καθημερινή ζωή όλων.

### 2.1.1 Συλλογή Ειδήσεων περί Κρυπτονομισμάτων

Τα κρυπτονομίσματα έχουν μπει εδώ και αρκετά χρόνια στις ζωές αρκετών χρηστών έχοντας αποτελέσει και συνεχίζοντας να αποτελούν, αντικείμενο έρευνας και συζητήσεων στους ευρείς κύκλους οικονομικών αναλύσεων και όχι μόνο. Με έντονη την αλληλεπίδραση τους στην καθημερινότητα, αποδεικνύεται ένα φημισμένο είδος ψηφιακών συναλλαγών, το μέλλον του οποίο προμηνύεται μακρύ.

Έχοντας ως βάση, λοιπόν τα προαναφερθέντα, αποφασίσθηκε η κατασκευή μιας εφαρμογής επισκόπησης ειδήσεων αναφορικά με τα κρυπτονομίσματα με στόχο την έγκαιρη ενημέρωση των ενδιαφερόμενων χρηστών.

Το σκέλος αυτής της εφαρμογής, περιλαμβάνει την κατανάλωση μιας εξωτερικής υπηρεσίας διαδικτύου, με τα δεδομένα αυτής να έρχονται σε κωδικοποίηση JSON. Εν συνεχεία, μέσω του ανάλογου business logic και την N Tier αρχιτεκτονικής που θα υλοποιηθεί, θα γίνει η αποθήκευση τους σε βάση δεδομένων PostgreSQL. Τέλος, θα πραγματοποιηθεί ανάκτηση των δεδομένων και η αποστολή τους σε front-end τμήμα της εφαρμογής τύπου HTML, για την αποτύπωση των ειδήσεων. Οι χρήστες κάνοντας κλήση της εφαρμογής, θα πρέπει να οδηγούνται σε μια ιστοσελίδα, με τις ειδήσεις ταξινομημένες και υπό μορφή υπερσυνδέσμων. Η χρήση των τελευταίων θα επιτρέπει την μεταπήδηση στην επίσημη πηγή (redirect) που έχει αρχικώς δημοσιοποιήσει την ανάλογη είδηση.

### 2.1.2 Best Box Office Ταινίες

Ο κινηματογράφος ως γνωστόν αποτελεί την 7η τέχνη, με μία μακρά ιστορία άνω των 200 ετών και με τεράστια επιρροή στο κοινωνικό γίγνεσθαι. Αναμφισβήτητα αποτελεί μια τέχνη, όπου λίγο ή πολύ έχει μεγάλη απήχηση στον ανθρώπινο πολιτισμό με την δημιουργία εκατοντάδων έργων κάθε χρόνο και τζίρους που κυμαίνονται στα επίπεδα των δισεκατομμυρίων δολαρίων. Η εξέλιξη της τεχνολογίας, δεν θα μπορούσε να αφήσει ανεπηρέαστο τον τομέα αυτό, όχι μόνο στον τρόπο δημιουργίας των ταινιών αλλά και σ’ αυτούς της κατανάλωσης των, από τους τελικούς χρήστες.

Συνοψίζοντας τα ανωτέρω, κρίθηκε απαραίτητη η κατασκευή μιας εφαρμογής, η κλήση της οποίας, θα ενημερώνει τους τελικούς χρήστες για τις ταινίες με τις μεγαλύτερες εισπράξεις σε παγκόσμιο επίπεδο. Η είσοδος των δεδομένων, θα γίνει με την κατανάλωση εξωτερικής API σε μορφή τύπου JSON και η αναπαράσταση τους σε σελίδα HTML.

## 2.2 Rational Unified Process[17]

### 2.2.1 Ορισμοί και Έννοιες

Η διαδικασία Rational Unified Process αποτελεί την πιο γνωστή και ευρέως διαδεδομένη τεχνική υλοποίησης περί της ανάπτυξης λογισμικού. Παρέχει μια πειθαρχημένη προσέγγιση στον τομέα της ανάληψης και διαμοιρασμού εργασιών και ευθυνών σ' έναν οργανισμό ανάπτυξης λογισμικού.


**Εικόνα 7. Η Rational Unified Process**

Βασικός της στόχος είναι η διασφάλιση της παραγωγής λογισμικού υψηλής ποιότητας, το οποίο θα εξυπηρετεί πλήρως τις ανάγκες των τελικών του χρηστών, βάσει ενός προβλεπόμενου χρονοδιαγράμματος και χρηματικού προϋπολογισμού.

Περιγράφει επίσης, την ανάπτυξη αποτελεσματικών και εμπορικά αποδεδειγμένων προσεγγίσεων για την ανάπτυξη λογισμικού και τις ομάδες που το αναπτύσσουν. Η τελευταία διαπίστωση είχε ως αποτέλεσμα την μετουσίωση της σε έξι βέλτιστες πρακτικές, οι οποίες είναι άρρηκτα συνδεδεμένες στην παραγωγή ποιοτικού λογισμικού από μεγάλους οργανισμούς ανάπτυξης και όχι μόνο. Με την χρήση των, γίνεται παροχή σε κάθε μέλος της ομάδας των κατευθυντήριων γραμμών, των προτύπων και των απαραίτητων εργαλείων για την παραγωγή βέλτιστου λογισμικού. Παρακάτω δίνεται μια συνοπτική περιγραφή των βέλτιστων πρακτικών.

1. Ανάπτυξη λογισμικού με επαναληπτικότητα

- ❖ Χρήση μιας επαναληπτικής προσέγγισης που επιτρέπει την αυξανόμενη κατανόηση του προβλήματος μέσω διαδοχικών βελτιώσεων και την σταδιακή ανάπτυξη μιας αποτελεσματικής λύσης σε πολλαπλές επαναλήψεις.

2. Διαχείριση των απαιτήσεων
   - ❖ Περιγραφή του τρόπου λειτουργίας, οργάνωσης και τεκμηρίωσης της απαιτούμενης λειτουργικότητας και των περιορισμών. Καταγραφή των αποφάσεων καθώς και επικοινωνία των επιχειρηματικών απαιτήσεων.

3. Χρήση κοινών συστατικών αρχιτεκτονικών
   - ❖ Εστίαση στην πρώιμη ανάπτυξη και τη δημιουργία βάσης, μιας ισχυρής εκτελέσιμης αρχιτεκτονικής, πριν την δέσμευση πόρων για την ανάπτυξη λογισμικού πλήρους κλίμακας. Περιγράφει τον σχεδιασμό μιας ανθεκτικής αρχιτεκτονικής, η οποία θα πρέπει να είναι ευέλικτη, να δέχεται αλλαγές, να είναι διαισθητικά κατανοητή και να προωθεί την αποτελεσματικότερη επαναχρησιμοποίηση του λογισμικού.

4. Οπτική μοντελοποίηση λογισμικού
   - ❖ Η συγκεκριμένη πρακτική δείχνει πως πρέπει να γίνει η μοντελοποίηση του λογισμικού ως προς τον οπτικό άξονα με σκοπό να γίνει η καταγραφή της δομής και της συμπεριφοράς των αρχιτεκτονικών του στοιχείων. Αυτό επιτρέπει την απόκρυψη των λεπτομερειών και την συγγραφή κώδικα με χρήση «γραφικών δομικών στοιχείων».

5. Επικύρωση της ποιότητας λογισμικού
   - ❖ Μια ενδεχόμενη κακή απόδοση ενός παραγόμενου λογισμικού καθώς και η χαμηλή του αξιοπιστία, θεωρούνται κοινοί παράγοντες που εμποδίζουν δραματικά την αποδοχή χρήσης του. Η RUP βοηθά στον προγραμματισμό, την υλοποίηση, την εκτέλεση και την αξιολόγηση αυτών των τύπων δοκιμών.

6. Έλεγχος των αλλαγών σε λογισμικό
   - ❖ Η ικανότητα διαχείρισης της αλλαγής, ο έλεγχος δηλαδή της αποδοχής της και η παρακολούθηση των αλλαγών, είναι απαραίτητη σ' ένα περιβάλλον στο οποίο, η τελευταία είναι αναπόφευκτη. Στον τομέα αυτό περιγράφεται ο τρόπος ελέγχου και παρακολούθησης των αλλαγών για την επιτυχή επαναληπτική ανάπτυξη.

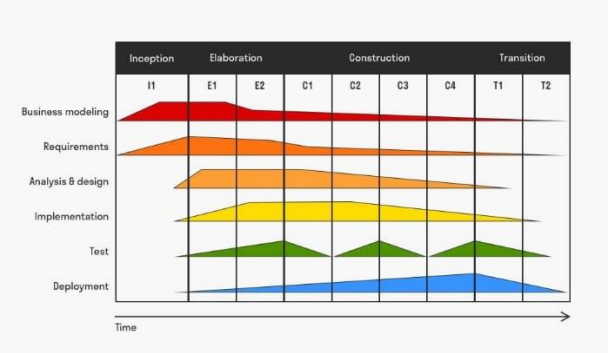
Επίσης, φέρνει μια ομάδα ανάπτυξης λογισμικού σε μια ενιαία οντότητα μέσω της δημιουργίας management πρακτικών.

### 2.2.2 Οπτικοποίηση σε Άξονες και Φάσεις**[18]**

Η διαδικασία της Rational Unified Process περιγράφεται σε δύο διαστάσεις ή καλύτερα δύο άξονες:

- Ο οριζόντιος άξονας αναπαριστά τον χρόνο και δείχνει την δυναμική πτυχή της διαδικασίας καθώς αυτή εκτελείται και η οποία εκφράζεται με όρους κύκλων, φάσεων, επαναλήψεων και σημείων ορόσημων.

- Ο κάθετος άξονας αναπαριστά την στατική πτυχή της διαδικασίας, ήτοι την περιγραφή υπό όρους δραστηριοτήτων και ροών έργου.




**Εικόνα 8. Διάγραμμα υλοποίησης της RUP στους άξονες του χρόνου και των δραστηριοτήτων[18]**

Στην ενότητα αυτή γίνεται η δυναμική οργάνωση των διαδικασιών της Rational Unified Process στον άξονα του χρόνου. Ο κύκλος ζωής ενός λογισμικού, χωρίζεται σε κύκλους ή φάσεις, καθένας απ' τους οποίους εργάζεται πάνω σε μια διαφορετική εκδοχή του τελικού προϊόντος. Η RUP διαιρεί κάθε κύκλο ανάπτυξης λογισμικού σε τέσσερις συνεχείς φάσεις, οι οποίες είναι οι κάτωθι.

- Φάση Έναρξης (Inception Phase)
- Φάση Εκπόνησης Μελέτης (Elaboration Phase)
- Φάση Κατασκευής (Construction Phase)
- Φάση Μετάβασης (Transition Phase)

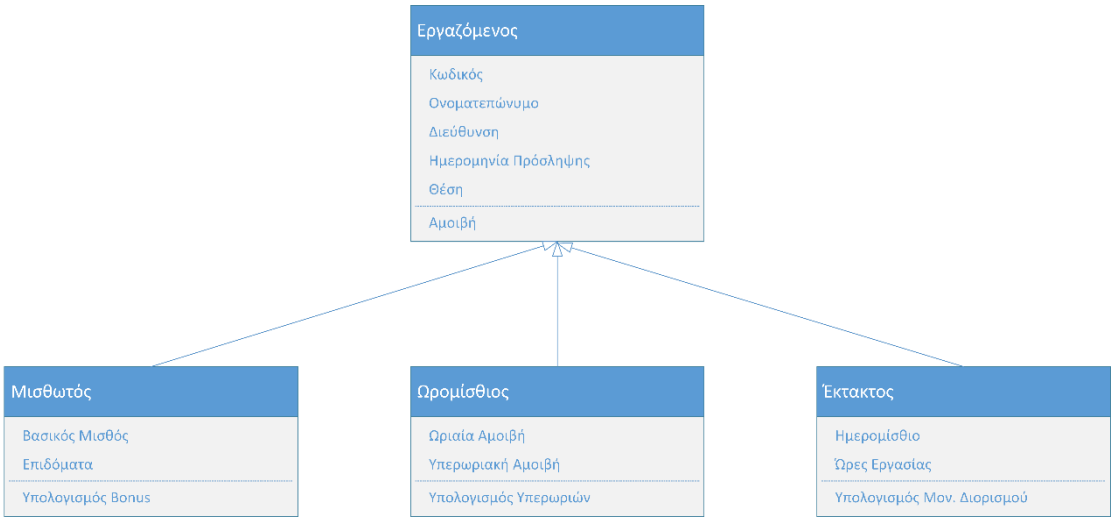
### 2.2.3 Αντικειμενοστραφής Ανάπτυξη**[19]**

Σύμφωνα με το βιβλίο «Πληροφοριακά Συστήματα Διοίκησης» των Kenneth C. Laudon και Jane Price Laudon[19], οι δομημένες μέθοδοι αντιμετωπίζουν τα δεδομένα και τις διεργασίες ως λογικά χωριστές οντότητες, ενώ στον πραγματικό κόσμο ένας τέτοιος διαχωρισμός φαίνεται αφύσικος. Γίνεται χρήση διαφορετικών συμβάσεων μοντέλων για την ανάλυση και τον σχεδιασμό ενός συστήματος.

Η αντικειμενοστραφής ανάπτυξη αντιμετωπίζει αυτά τα ζητήματα, κάνοντας χρήση του όρου «αντικείμενο», το οποίο συνδυάζει τα δεδομένα και τις συγκεκριμένες διεργασίες που ενεργούν πάνω σ αυτά. Τα τελευταία βρίσκονται ενθυλακωμένα στο αντικείμενο και μπορούν να προσπελαστούν και να τροποποιηθούν από τις πράξεις και τις μεθόδους που είναι άρρηκτα συνδεδεμένες με το αντικείμενο. Οι τελευταίες ενεργοποιούνται μέσω μηνυμάτων που λαμβάνονται από εξωτερικά προγράμματα. Το αποτέλεσμα είναι η δημιουργία ενός συστήματος όπου στο εσωτερικό του απαρτίζεται από πολλά τέτοια αντικείμενα ως συλλογή αλλά και τις μεταξύ τους σχέσεις για την μεταξύ τους αλληλεπίδραση.

Η αντικειμενοστραφής λογική βασίζεται στις έννοιες της κλάσης (ή τάξης) καθώς και της κληρονομικότητας μεταξύ αυτών. Αντικείμενα που ανήκουν σε μία κλάση έχουν τα γνωρίσματα και τις μεθόδους αυτής της κλάσης. Οι κλάσεις αντικειμένων κληρονομούν, με τη σειρά τους, όλη τη δομή και τις συμπεριφορές μιας πιο γενικής κλάσης, και στη συνέχεια προσθέτουν μεταβλητές και συμπεριφορές μοναδικές για κάθε αντικείμενο.

Η εικόνα 10 παρουσιάζει γραφικά την κληρονομικότητα μεταξύ της υπερ-κλάσης «Εργαζόμενος» και των υπό-κλάσεων «Μισθωτός», «Ωρομίσθιος» και «Έκτακτος» με σκοπό να αποτυπώσει τις σχέσεις μεταξύ αυτών και τον τρόπο πληρωμής τους.




**Εικόνα 9. Παράδειγμα κληρονομικότητας μεταξύ κλάσεων**

## 2.3 Ανάλυση Απαιτήσεων**[21]**

Με πρόθεση την απεικόνιση του κύκλου ζωής ενός λογισμικού, γίνεται χρήση του μοντέλου των φάσεων της μεθοδολογίας RUP (Πίνακας 4), από τον οποίο φαίνεται ότι το πρώτο βήμα προς την πραγμάτωση του, είναι η ανάπτυξη της ανάλυσης απαιτήσεων, από την οποία θέτονται οι βάσεις για τον ορθό και σωστό σχεδιασμό του. Η τελευταία αποτελεί κύριο άξονα της Μηχανικής των Απαιτήσεων που αναφέρεται στην αντιστοίχιση των στόχων των λειτουργιών και των περιορισμών των συστημάτων από τον πραγματικό κόσμο, σε έναν αντίστοιχο λογισμικό.

| ΦΑΣΗ | ΕΡΓΑΣΙΑ |
|---|---|
| Έναρξη (Inception) | Σύλληψη Απαιτήσεων |
| | Ανάλυση – Σχεδιασμός |
| Εκπόνηση Μελέτης (Elaboration) | Ανάλυση - Σχεδιασμός |
| | Υλοποίηση - Έλεγχος |
| Κατασκευή (Construction) | Ανάλυση - Σχεδιασμός |
| | Υλοποίηση - Έλεγχος |
| Μετάβασης (Transition) | Υλοποίηση - Έλεγχος |
| | Εκπαίδευση |

**Πίνακας 4. Συνοπτική απεικόνιση φάσεων μεθοδολογίας RUP**

Ο σκοπός της ανάλυσης απαιτήσεων είναι η συλλογή των απαιτήσεων μιας επαγγελματικής και μη οντότητας (οργανισμός, εταιρία, χρήστες κτλ.), με σκοπό αυτοί να χρησιμοποιηθούν στην ανάπτυξη του μοντέλου δεδομένων του λογισμικού συστήματος. Από τον ορισμό που μόλις δόθηκε, είναι θεμιτή η διάκριση αυτών σε λειτουργικές και μη λειτουργικές απαιτήσεις.

Οι μεν πρώτες, καθορίζουν την λειτουργικότητα του λογισμικού, το οποίο θα κατασκευασθεί από την υπεύθυνη ομάδα ανάπτυξης, με απώτερο στόχο το προϊόν να επιτρέπει στους χρήστες του, να διεκπεραιώνουν τα καθήκοντα τους. Ένας ακόμη ισοδύναμος ορισμός που μπορεί να εντοπισθεί ως ανάλυση αυτών που ειπώθηκαν είναι οι απαιτήσεις συμπεριφοράς (Behavioral Requirements).

Οι μεν δεύτερες, καθορίζουν τους τρόπους με τους οποίους το σύστημα θα πρέπει να λειτουργεί. Στην κατηγορία αυτή, εμπίπτουν χαρακτηριστικά ποιότητας που αναφέρονται στην ευχρηστία και την ταχύτητα του συστήματος, στην συχνότητα με την οποία αυτό αποτυγχάνει καθώς και την διαχείριση απρόβλεπτων συνθηκών, στις οποίες μπορεί αυτό να βρεθεί. Η τήρηση τους είναι καθοριστικής σημασίας στην επιτυχία ή αποτυχία των λογισμικών συστημάτων και για την επίτευξη τους είναι σημαντική, η διερεύνηση των προσδοκιών ποιότητας των τελικών χρηστών.

### 2.3.1 Υλοποίηση Ανάλυσης Απαιτήσεων

Συνοψίζοντας τα παραπάνω, το βήμα αυτό προϋποθέτει, την συλλογή πληροφοριών που αφορούν τις διαδικασίες, τις οντότητες (κατηγορίες δεδομένων) και τις διάφορες μονάδες και τμήματα ενός οργανισμού. Με την ολοκλήρωση αυτής της διαδικασίας, τα εξαγόμενα δεδομένα θα χρησιμοποιηθούν στην σχεδίαση της εφαρμογής, με τη βοήθεια διαγραμμάτων που απεικονίζουν μέσω συσχετίσεων τις διαδικασίες και τα δεδομένα. Ο στόχος αυτής της φάσης συνοψίζεται στις εξής παραδοχές:

- Κατανόηση των τρόπων με τους οποίους δουλεύει ένας εταιρικός οργανισμός και τις επιθυμίες των τελικών χρηστών.
- Να έρθουν στην επιφάνεια τα προβλήματα και οι περιορισμοί που διέπονται στην καθημερινή εργασία των χρηστών (end users)
- Κατανόηση των στόχων που ο οργανισμός και οι τελικοί χρήστες, θέλουν να επιτύχουν με την υλοποίηση του έργου.
- Καθορισμός του πεδίου εφαρμογής και των ορίων του έργου. Τα τελευταία είναι απαραίτητα έτσι ώστε να γίνει σαφής η δημιουργία του λογισμικού συστήματος όπως αυτό εξ αρχής ορίσθηκε.

Η συλλογή της απαραίτητης πληροφορίας για την διεξαγωγή της ανάλυσης απαιτήσεων, πραγματοποιείται μέσω συνεντεύξεων, που διενεργούνται σε τελικούς χρήστες του εκάστοτε οργανισμού καθώς και με την ανάγνωση πηγών (manuals), τα οποία ήδη χρησιμοποιούνται εντός του.

Η προαναφερθείσα διαδικασία, βοηθά στην εξαγωγή πολύτιμης γνώσης, η οποία με την σειρά της θα βοηθήσει την υλοποίηση των υπόλοιπων διεργασιών του συστήματος. Συνοψίζοντας τα παραπάνω προκύπτουν οι κάτωθι παραδοχές αναφορικά με την ανάλυση απαιτήσεων:

- Επικοινωνία μέσω συνεντεύξεων με τους τελικούς χρήστες
- Αναγνώριση απαραίτητης «αληθινής» πληροφορίας
- Απομάκρυνση περιττής και μη σημαντικής πληροφορίας
- Διευκρίνιση ασαφών δηλώσεων που διέπονται στην μετάδοση πληροφορίας
- Ένωση των κενών σημείων που δημιουργούνται κατά την διεξαγωγή των συζητήσεων για την αποκόμιση της πληροφορίας
- Διάκριση δεδομένων και λειτουργιών

Με βάση όλα τα παραπάνω λεγόμενα γίνεται κατανοητό ότι η ανάλυση απαιτήσεων όπως και ο εννοιολογικός σχεδιασμός, που θα ακολουθήσει, στοχεύουν στην συγκέντρωση σκέψεων και συζητήσεων για την εγκαθίδρυση των θεμελίων που θα ορίσουν το οικοδόμημα του λογισμικού συστήματος.

#### 2.3.1.1 Λειτουργικές Απαιτήσεις

Στην ενότητα αυτή γίνεται η καταγραφή των λειτουργικών απαιτήσεων με τυχαία καταγραφή.

- ✓ Οι τελικοί πελάτες αναζητούν την δυνατότητα να τους παρέχονται περαιτέρω πληροφορίες στον τομέα του cryptocurrency.

- ✓ Την ίδια δυνατότητα αναζητούν πελάτες, οι οποίοι αρέσκονται στην 7η τέχνη ήτοι του κινηματογράφου. Επιθυμία τους, λοιπόν είναι η συλλογή δεδομένων σε τομέα του συγκεκριμένου θέματος.

- ✓ Δεν υπάρχει συγκεκριμένη απαίτηση για τον τρόπο και από πού, θα λαμβάνονται οι πληροφορίες αυτές.

- ✓ Οι πληροφορίες μπορούν να προσφέρονται απο κάποια ή κάποιες διεπαφές (API) από το διαδίκτυο, η οποία παρέχονται από τους εκάστοτε κατασκευαστές. Σκοπός της είναι η παροχή των δεδομένων προς τους developers, που επιθυμούν την επεξεργασία των δεδομένων και την παροχή συμβατών αποτελεσμάτων και συμπερασμάτων.

- ✓ Οι περαιτέρω πληροφορίες που θα προκύψουν πρέπει να αντικατοπτρίζουν και να είναι συνυφασμένες με τις λειτουργίες της εκάστοτε διεπαφής.

- ✓ Οι πληροφορίες θα πρέπει να παρέχονται στους χρήστες απρόσκοπτα και σε real time χρόνο.

#### 2.3.1.2 Μη Λειτουργικές Απαιτήσεις

Στην ενότητα αυτή γίνεται η καταγραφή των μη λειτουργικών απαιτήσεων με τυχαία καταγραφή. Οι συγκεκριμένες καλούνται να υποστηρίξουν την ορθή λειτουργία των λειτουργικών απαιτήσεων που περιγράφηκαν μόλις.

- ➢ Η εφαρμογή θα πρέπει να είναι σε θέση να παρέχει real time δεδομένα στη συσκευές των χρηστών.

- ➢ Για τον διαχωρισμό των δεδομένων που θα εισάγονται και εξάγονται, μπορεί να γίνει χρήση εξωτερικής API.

- ➢ Τα δεδομένα των διεπαφών θα είναι σε μορφή JSON και θα εγγράφονται σε βάση δεδομένων.

- Εφόσον ολοκληρωθεί η κατανάλωση των διεπαφών, θα γίνεται δημιουργία των πεδίων σε βάση δεδομένων, τα οποία θα είναι και τα στοιχεία που θα εξάγονται πίσω στους χρήστες.

- Υλοποίηση DevOps διαδικασιών κατά τις οποίες θα γίνει η προσάρτηση της πλατφόρμας Prometheus, η οποία ειδικεύεται στην κατανάλωση των metrics τηw εφαρμογής.

- Περαιτέρω εξειδίκευση της διαδικασίας με ένωση της Prometheus με την πλατφόρμα Grafana, η οποία παίρνει τα δεδομένα απο την τελευταία και τα μετατρέπει σε γραφήματα.

- Για την υλοποίηση της DevOps διαδικασίας, καλή θα ήταν η χρήση containerized software μέσω Docker. Αν χρειαστεί, να γίνει χρήση Linux OS μέσω WSL.

- Ο χρόνος απόκρισης του συστήματος δεν θα πρέπει να υπερβαίνει τον χρόνο των 5 δευτερολέπτων. Αυτό ισχύει για οποιοδήποτε λειτουργία, λαμβάνει χώρα.

- Η εφαρμογή θα πρέπει να υλοποιεί την τεχνική N-Tier Architecture[22], κατά την οποία πραγματοποιείται διαχωρισμός των ευθυνών και των εξαρτήσεων μέσω ορισμού επιπέδων (layers). Τα επίπεδα θα πρέπει να περιλαμβάνουν την επικοινωνία με τους πελάτες, την διαχείριση των ερωτημάτων κατά CRUD, το επίπεδο λογικής και επεξεργασίας των δεδομένων και το επίπεδο επικοινωνίας με τη βάση δεδομένων.

- Για την ασφάλεια της εφαρμογής, θα πρέπει να υλοποιηθεί Spring Security με σύστημα καταγραφής των χρηστών σε βάση δεδομένων και αυθεντικοποίηση αυτών μέσω Email Verification.

- Στα πλαίσια της προηγούμενης απαίτησης θα πρέπει να αλλάξει το πρωτόκολλο HTTP σε HTTPS, με την κατασκευή Self-Signed Certificate.

- Υλοποίηση Exception Handling σε global scope θα βοηθούσε στην συνολική διαχείριση των σφαλμάτων από μία κλάση της εφαρμογής.

- Υλοποίηση JUnit Tests σ' όσο το δυνατόν μεγαλύτερο μέρος του συγγραφόμενου κώδικα για την εξασφάλιση ορθής λειτουργίας των μεθόδων που θα προκύψουν. Εφόσον καταστεί εφικτό, ας εφαρμοστεί η τεχνική, Test Driven Development.

- Αποτύπωση των ζητούμενων δεδομένων σε front end τμήμα της εφαρμογής με την βοήθεια HTML σελίδων. Καλή θα ήταν η χρήση Thymeleaf templates καθώς και CSS, για την όσο το δυνατόν καλύτερη απεικόνιση τους.

- Υλοποίηση Git Versioning προκειμένου να γίνεται ο έλεγχος των αλλαγών που λαμβάνουν και εφόσον χρειαστεί revert σε προηγούμενο χρονικό σημείο της εφαρμογής.

Όλα τα ανωτέρω, αποτελούν μία σύνθεση λειτουργιών μέσα από την οποία στήνεται η επίτευξη των στόχων της τελικής εφαρμογής.

## 2.4 Διαγραμματική Απεικόνιση Εφαρμογής

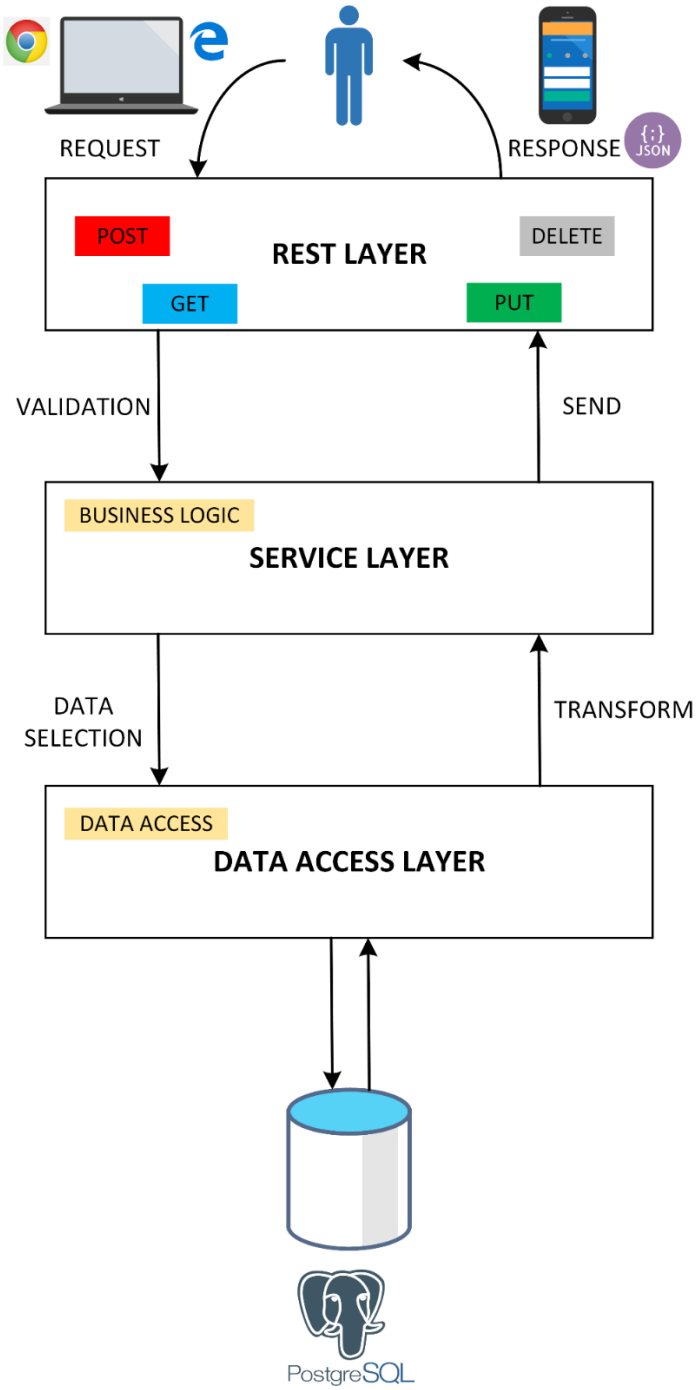
Η ολοκλήρωση των λεγόμενων της προηγούμενης παραγράφου, αποτελεί το θεμέλιο λίθο για την απαρχή της δημιουργίας της εφαρμογής. Στην συγκεκριμένη παράγραφο, γίνονται οι προσπάθειες αποτύπωσης σε γραφική μορφή των επιπέδων από τα οποία θα αποτελείται, η τελική εφαρμογή. Στόχος είναι η δημιουργία ενός χάρτη στον οποίο θέτονται οι συντεταγμένες επικοινωνίας μεταξύ των στρωμάτων της εφαρμογής και πως αυτές θα υλοποιηθούν στον IDE IntelliJ IDEA μέσω της μεθόδου dependency injection.

### 2.4.1 Μορφή Διαγράμματος Εφαρμογής RESTful API

Η εικόνα 11, απεικονίζει την μορφή κατά την οποία μπορεί να στηθεί μια back-end εφαρμογή υπηρεσίας web, με την βοήθεια του Spring Framework. Το σημείο έναρξης υλοποιείται από την πλευρά του πελάτη, ο οποίος με την χρήση τερματικών συσκευών όπως προσωπικοί υπολογιστές και smartphones, επιθυμεί να εκτελέσει κάποιες λειτουργίες. Παράδειγμα λειτουργίας θα μπορούσε να είναι η εκτύπωση του φορολογικού σημειώματος εκκαθάρισης του από την υπηρεσία Taxis Net. Η πλοήγηση γίνεται ως γνωστόν μέσω του browser στο front-end τμήμα της υπηρεσίας, όπου ο τελευταίος περιηγείται.

Μέσω της μηχανής φυλλομετρητή, ο πελάτης αποστέλλει ερωτήματα με τη βοήθεια του πρωτοκόλλου HTTP, σκοπός των οποίων είναι να ενημερώσουν την υπηρεσία για την ανάγκη του. Τα ερωτήματα αυτά, ακολουθούν το πρότυπο CRUD και αναλύθηκαν συνοπτικά στην ενότητα 1.1.4. Η δε επεξεργασία τους αναλαμβάνεται από το επίπεδο REST.

Εν συνεχεία ακολουθεί το επίπεδο Service το οποίο είναι υπεύθυνο για την διαχείριση του business logic των δεδομένων που εισάγονται στην υπηρεσία και που τα οποία θα πρέπει να διαβιβασθούν στο επόμενο επίπεδο υπό την ονομασία Data Access. Το τελευταίο εφόσον τα λάβει θα πρέπει να κάνει την τελευταία μεταβίβαση προς τη σχεσιακή ή μη, βάση δεδομένων, με την διαδικασία να ακολουθείται αντιστρόφως.




**Εικόνα 10. Διάγραμμα απεικόνισης Client – RESTful API (Πηγή: Δημιουργία στο Microsoft Visio[12])**

### 2.4.2 Χάρτης Εφαρμογής

Στα πλαίσια υλοποίησης του θέματος της παρούσας εργασίας, σκοπός είναι η πραγματοποίηση των λεγόμενων της προηγούμενης παραγράφου, με τη σημαντική διαφοροποίηση ότι τα δεδομένα στην είσοδο της, θα πρέπει να λαμβάνονται από εξωτερικές API's, με σκοπό την επεξεργασία τους και την πρόσδοση των αποτελεσμάτων πίσω στους χρήστες. Η εικόνα 11, απεικονίζει έναν χάρτη της εφαρμογής προς κατασκευή, όπως αυτός στοιχειοθετήθηκε από την ανάλυση απαιτήσεων που έλαβε χώρα σε προηγούμενη παράγραφο.

Προβλέπονται λοιπόν οι εξής παραδοχές για την κατασκευή της εφαρμογής αυτής με στόχο την επίτευξη των ζητούμενων αποτελεσμάτων:

1. Χρήση της θύρας 8443 στην οποία θα εκκινεί η εφαρμογή υπό το πρωτόκολλο HTTPS.

2. Ταυτόχρονα με την θύρα 8443, η εφαρμογή θα εκκινεί στην θύρα 8080, την δυνατότητα εξαγωγής endpoints, από τα οποία μπορούν να εξαχθούν metric πληροφορίες.

3. Εκκίνηση της διαδικασίας αυθεντικοποίησης των χρηστών, με την απεικόνιση μιας φόρμας HTML σύνδεσης αυτών.

4. Η προηγούμενη φόρμα θα εμπεριέχει ένα μπουτόν «Register», για την μετάβαση των χρηστών στην σελίδα HTML εισαγωγής των στοιχείων τους.

5. Οι χρήστες εισάγουν τα στοιχεία τους και χρησιμοποιούν το πλήκτρο «Register» για την καταχώρηση των δεδομένων τους στην βάση.

6. Λαμβάνεται φόρμα επιβεβαίωσης ότι τα στοιχεία έχουν καταχωρηθεί επιτυχώς.

7. Το σημείο αυτό είναι κομβικό καθώς εκκινεί η υπηρεσία αυθεντικοποίησης της δηλωμένης διεύθυνσης mail των χρηστών. Έτσι, θα σταλεί αυτοματοποιημένο μήνυμα στην διεύθυνση αυτή.

8. Ο εκάστοτε χρήστης οφείλει να συνδεθεί στον client διαχείρισης του mail box του και να πατήσει το πλήκτρο «Activate Now».

9. Γίνεται αυτόματο redirection σε σελίδα HTML, η οποία θα επιβεβαιώσει το επιτυχές verification. Ταυτόχρονα, εμπεριέχει και πλήκτρο εκ νέου προς την σελίδα καλωσορίσματος.

10. Ο χρήστης μπορεί πλέον να εισάγει τα στοιχεία του στην σελίδα αυτή, για να χρησιμοποιήσει την εφαρμογή.

11. Η εφαρμογή εκκινεί και στο σημείο αυτό, γίνεται κατανάλωση δύο εξωτερικών APIs. Η πρώτη θα καταναλώνει δεδομένα παροχής ειδήσεων περί του cryptocurrency. Η δεύτερη στοιχεία από την ιστοσελίδα της IMDb, αναφορικά με τις καλύτερες Box Office ταινίες.

12. Η κατανάλωση των δύο αυτών υπηρεσιών, θα γίνεται με διαδραστικό τρόπο μέσω CSS. Συγκεκριμένα, θα υπάρχουν δύο παράθυρα τα οποία κατά το hover event του ποντικιού θα μπορούν να κάνουν zoom out ενώ πάνω τους θα υπάρχει σχετικό logo και εικόνα.

13. Οι χρήστες θα μπορούν να αλληλοεπιδρούν με τα παράθυρα αυτά για να μεταβούν στις εκάστοτε υπηρεσίες πάντοτε σε σελίδες HTML. Για την αποτύπωση των δεδομένων, από την αντίστοιχη βάση, θα γίνει χρήση Thymeleaf.

14. Τα απεικονιζόμενα δεδομένα, ενημερώνονται σε πραγματικό από τις εξωτερικές APIs. Το σημείο αυτό, σηματοδοτεί την ολοκλήρωση κατανάλωσης της υπηρεσίας.

15. Με την εκκίνηση της υπηρεσίας που έγινε στο βήμα 11, εκκινεί και η απαίτηση των DevOps διαδικασιών. Με χρήση της πλατφόρμας Docker γίνεται η εγκατάσταση των Prometheus και Grafana ως containerized software. Οι εντολές λαμβάνουν χώρα σε περιβάλλον WSL Ubuntu.

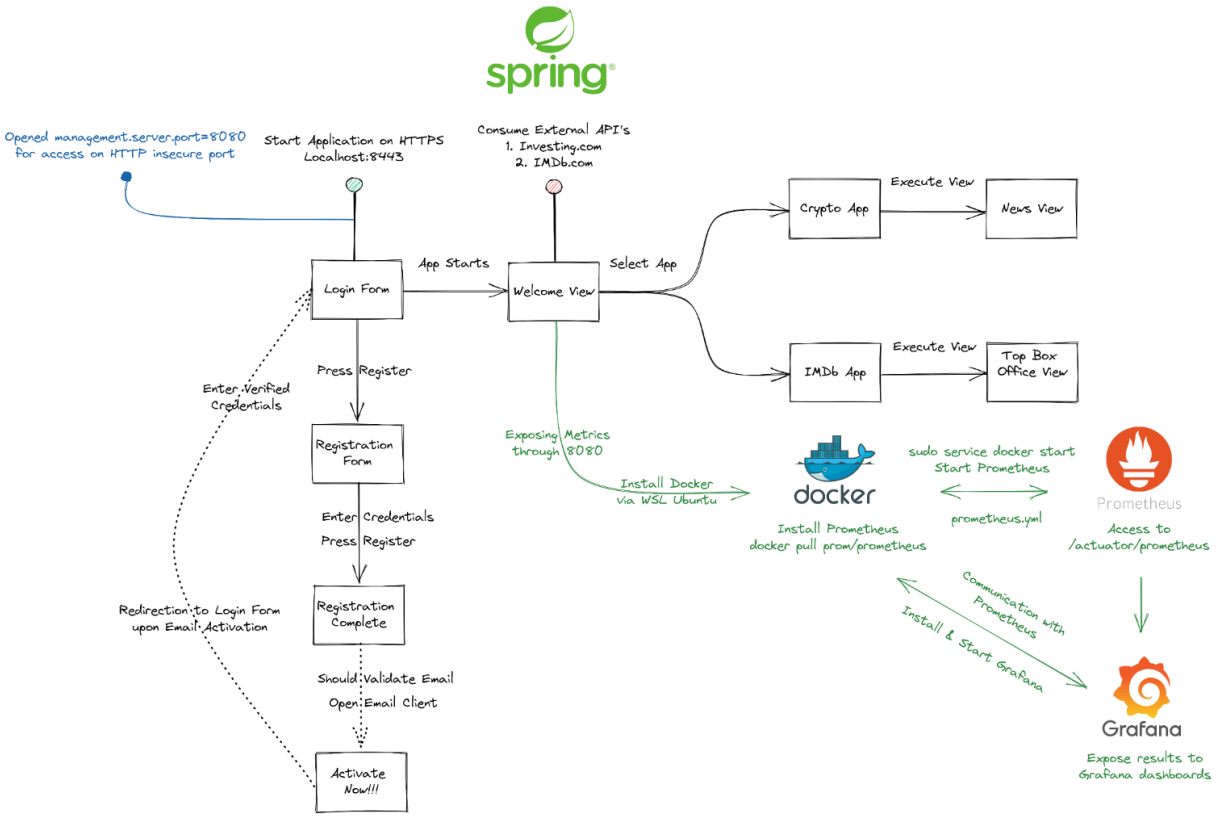
16. Με χρήση των θυρών 9090 και 3000, γίνεται η server επικοινωνία με τις πλατφόρμες Prometheus και Grafana, αντίστοιχα.

Με δεδομένο ότι το πεδίο υπηρεσίας που εξετάζεται είναι τύπου RESTful API, η μορφή των δεδομένων δεν θα μπορούσε να είναι άλλη από την Javascript Object Notation, ήτοι JSON, που θεωρείται από τις πιο διάσημες στον τομέα μεταφοράς και ανταλλαγής δεδομένων σε σύγχρονες υπηρεσίες διαδικτύου και κινητών συσκευών. Ο συγκεκριμένος τύπος δεδομένων, θα χρησιμοποιηθεί και στην είσοδο της εφαρμογής όπως και στην απόδοση των αποτελεσμάτων.

Η μεταφορά εφόσον περάσει απ' όλα τα προαναφερθέντα επίπεδα, θα πρέπει να ολοκληρωθεί με την απόθεση των δεδομένων στη αντίστοιχη βάση που θα στηθεί. Το είδος της βάσης μπορεί να είναι είτε σχεσιακό είτε μη. Είναι γνωστό, ωστόσο, ότι η λειτουργία βάσεων δεδομένων με αρχεία τύπου JSON, είναι μεν εφικτή αλλά όχι εύκολα διαχειρίσιμη. Για τους λόγους αυτούς, λοιπόν, ως προς την χειραγώγηση των, θα γίνει χρήση σχεσιακής βάσης με χρήση POJO κλάσεων.

Με την απόθεση των δεδομένων, θα χρειαστεί να γίνει ανάκτηση αυτών, όταν αυτό ζητείται από τους χρήστες, μέσω του ανάλογου κώδικα. Το πέρας του σταδίου αυτού, σηματοδοτεί και την επιστροφή των αποτελεσμάτων πίσω στους πελάτες με την αντίστροφη διαδικασία, όπως αυτή παρουσιάζεται στο διάγραμμα της εικόνας 10. Η υλοποίηση των επιμέρους επιπέδων, θα γίνει στον IDE IntelliJ IDEA, υπό μορφή κλάσεων ενώ η σύνδεση τους, όπως αναφέρθηκε και πριν, θα γίνει με χρήση της μεθόδου dependency injection.

Κατά την διάρκεια συγγραφής του κώδικα, θα γίνουν οι ανάλογες επεξηγήσεις των μεθόδων που λαμβάνουν χώρα καθώς και των επιπρόσθετων λογισμικών που ενδεχομένως χρησιμοποιηθούν στο εν λόγω project.




**Εικόνα 11. Χάρτης Εφαρμογής προς κατασκευή (Πηγή: Δημιουργία στο Excalidraw)**

# 3 ΧΡΗΣΙΜΟΠΟΙΟΥΜΕΝΕΣ ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΕΡΓΑΛΕΙΑ ΕΦΑΡΜΟΓΗΣ

Στο προηγούμενο κεφάλαιο, ολοκληρώθηκε το θεωρητικό υπόβαθρο και η τοποθέτηση των στόχων της RESTful εφαρμογής, δηλαδή αυτούς που θα επιτελούνται και τα αποτελέσματα που θα επιστρέφονται στους χρήστες. Στο παρόν, καλείται να γίνει η επεξήγηση των τεχνολογιών και των εργαλείων των οποίων θα γίνει χρήση, για την επίτευξη των τελευταίων. Με συνοπτικές επεξηγήσεις και ερμηνείες, αναλύονται τα εργαλεία και τα οφέλη που αυτά προσέφεραν κατά την συγγραφή του κώδικα της εφαρμογής.

Το κεφάλαιο αυτό όπως και το επόμενο, βασίζεται στον «χάρτη» της εικόνας 11 με μία περισσότερο εξειδίκευση σε όρους, τεχνικές και ερμηνείες, όπου δεν ήταν δυνατόν, να απεικονισθούν εκεί. Τέλος, σε ξεχωριστή ενότητα, γίνεται αναφορά στην ραχοκοκαλιά της εφαρμογής (backbone), η οποία αναφέρεται στον σκελετό κλάσεων σε γλώσσα Java, που συνέθεσαν το συνολικό οικοδόμημα.

## 3.1 Τεχνολογίες & Εργαλεία

Πριν την οποιαδήποτε τεχνική ανάπτυξη λογισμικού, είναι ιδιαιτέρως σημαντικό να αναφερθούν τα εργαλεία και οι τεχνολογίες που έλαβαν χώρα ανά εφαρμογή και πως αυτά σε συνδυασμό εννοείται με το αντίστοιχο κεφάλαιο 1, συνέδραμαν στο τελικό αποτέλεσμα. Στόχος είναι να δοθούν όσο το δυνατόν αναλυτικές πληροφορίες χωρίς όμως ταυτόχρονα να επιτυγχάνεται εμβάθυνση, καθότι αυτό ξεφεύγει από τους σκοπούς της παρούσας διπλωματικής εργασίας. Στις υποενότητες που θα ακολουθήσουν, παρουσιάζονται οι έννοιες των τεχνολογιών και πως αυτές διαδραμάτισαν σημαντικό ρόλο στην παροχή της RESTful εφαρμογής.

### 3.1.1 JSON**[26]**

Ο ορισμός JSON προέρχεται από τις λέξεις JavaScript Object Notation. Στην επιστήμη της Πληροφορικής ορίζεται ως μορφότυπο το οποίο έχει την ικανότητα να μεταβιβάζει πληροφορία και δεδομένα από και προς διάφορες υπηρεσίες του διαδικτύου. Η δομή του είναι τέτοια ώστε η πληροφορία να γίνεται κατανοητή από τον απλό άνθρωπο αλλά ταυτόχρονα να μπορεί να γίνεται η εύκολη διαχείριση της από τους ηλεκτρονικούς υπολογιστές.

Η αποτύπωση της πληροφορίας γίνεται με την μορφή «λεξικού», που στην ουσία διασπά την πληροφορία σε κλειδιά και σε τιμές αυτών. Αν για παράδειγμα μια πληροφορία κλειδιού είναι η λέξη «όνομα», τότε η αξία αυτής θα είναι τα ονόματα όλων των ανθρώπων (π.χ. “name”: “Andreas”).

```
[
  {
    "gameId": 5104,
    "drawId": 2466,
    "drawTime": 1657825200000,
    "status": "active",
    "drawBreak": 1800000,
    "visualDraw": 2466,
    "pricePoints": {
      "amount": 0.5
    },
    "prizeCategories": [
      {
        "id": 1,
        "divident": 0,
        "winners": 0,
        "distributed": 0,
        "jackpot": 204241.01,
        "fixed": 0,
        "categoryType": 0,
        "gameType": "Normal",
        "minimumDistributed": 600000
      },
```


**Εικόνα 12. Παράδειγμα JSON τιμών απο την API του ΟΠΑΠ**

Η παραπάνω εικόνα, αποτυπώνει την πραγματική έξοδο δεδομένων μιας εκ των διαθέσιμων API του Οργανισμού Προγνωστικών Αγώνων Ποδοσφαίρου (ΟΠΑΠ). Σ αυτήν αποτυπώνονται συνοπτικά μέρος των δεδομένων, τα οποία αποτελούνται από κλειδιά και τιμές. Ως αντικείμενο JSON ορίζεται καθένα που ξεκινά με αριστερό άγκιστρο { και τελειώνει με δεξί άγκιστρο }.

Ένα κλειδί, μπορεί να απαρτίζεται από επιμέρους κλειδιά με τις επιμέρους τους αξίες. Τέτοιο παράδειγμα στην παραπάνω εικόνα αποτελεί το κλειδί "prizeCategories", το οποίο εντός του έχει περισσότερα του ενός αντικείμενα τα οποία διαχωρίζονται με τα κλειδιά "id" και αντιστοίχως λαμβάνουν τις δικές τους τιμές. Όλα αυτά θα πρέπει να περικλείονται σε λίστα που ξεκινά με αριστερή αγκύλη [ και κλείνει με δεξιά ]. Οι αγκύλες ξεκινούν και αναφέρονται στο κλειδί "prizeCategories".

### 3.1.2 Postman**[27]**

Κατά την χρήση, κατασκευή και κλήση εξωτερικών και όχι API's, μπορεί να γίνει χρήση του κλασσικού browser, που κάθε χρήστης χρησιμοποιεί για να περιηγηθεί στο διαδίκτυο. Τα αποτελέσματα της εικόνας στην προηγούμενη ενότητα παράχθηκαν με χρήση του url https://api.opap.gr/draws/v3.0/5104/last/2, το οποίο καταχωρήθηκε στην γραμμή αναζήτησης του Google Chrome του συγγραφέα φοιτητή. Η συγκεκριμένη διεύθυνση διαδικτύου παρήγαγε επιτυχώς την άνωθεν μορφοποίηση JSON, με επιτυχία χωρίς προβλήματα ή περιορισμούς. Στις περισσότερες όμως των περιπτώσεων, λαμβάνει χώρα το ρητό «that's not the case, in this one», το οποίο ανάγεται σε περιπτώσεις κατά τις οποίες η πρόσβαση σε μια API απαιτεί επιπρόσθετες παραμέτρους που υπακούν σε πρότυπα ασφάλειας. Στις συνθήκες αυτές, η απλή πληκτρολόγηση ενός URL δεν θα δώσει τα αναμενόμενα αποτελέσματα.

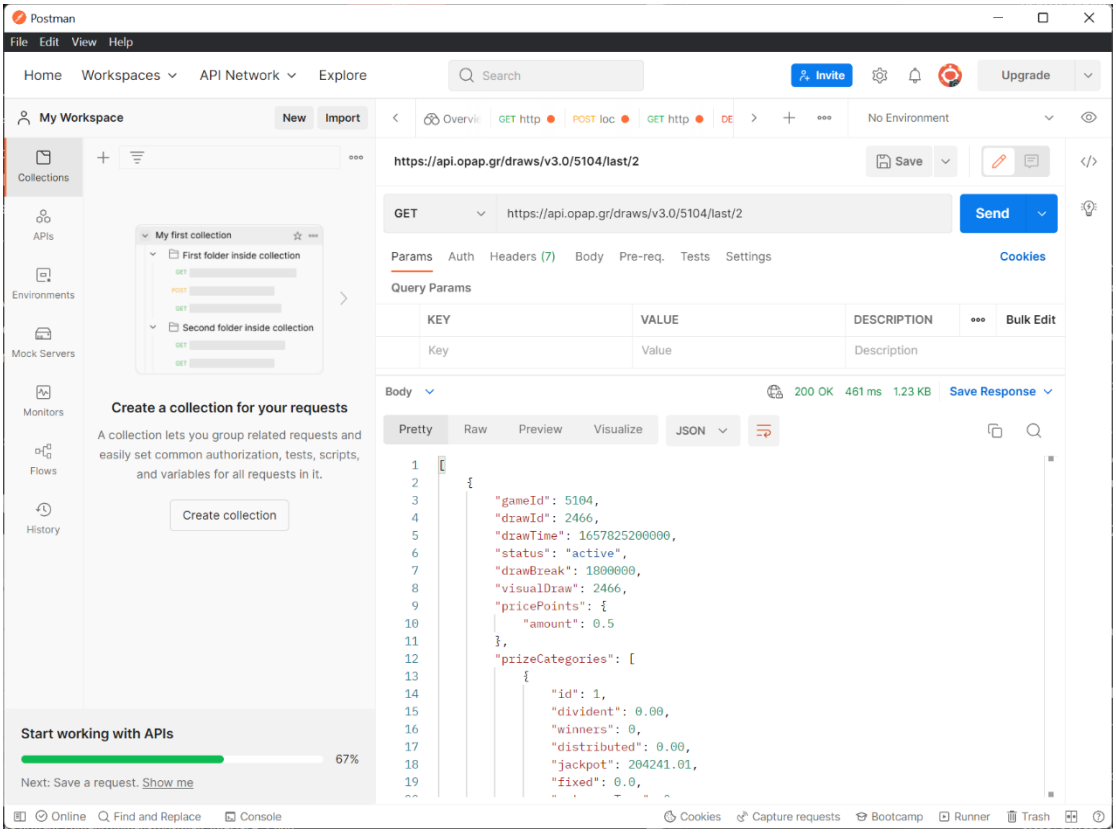

**Εικόνα 13. Λογότυπο Postman**

Ο λόγοι που συντρέχουν στο πρόβλημα αυτό, έχουν να κάνουν κατά κύριο λόγο στην σωστή, ορθή και ασφαλή χρήση μιας API, η οποία μπορεί να βρεθεί εκτεθειμένη στα πληκτρολόγια επιτήδειων, οι οποίοι θα μπορούσαν να προκαλέσουν ανεπανόρθωτες βλάβες. Έτσι λοιπόν, πολλές φορές θα προκύψουν περιπτώσεις χρήσης APIs, οι οποίες απαιτούν την εγγραφή με τα credentials του χρήστη στην ανάλογη βάση δεδομένων τους, με σκοπό να του δοθεί η ανάλογη πρόσβαση σ αυτή. Με την ολοκλήρωση της διαδικασίας αυτής, παρέχονται από την API υπηρεσία κάποια περαιτέρω credentials με το πιο κοινό και γνωστό, να ακούει στο όνομα APIKEY. Το τελευταίο αποτελεί μοναδικό και προσωπικό κωδικό ανά χρήστη και απαιτείται η χρήση του κατά

την κλήση της API. Στις επόμενες παραγράφους θα αντιμετωπισθεί μια τέτοια περίπτωση και εκεί θα γίνει μία εν πρακτέω ανάλυση.

Ενδεχομένως, γίνεται κατανοητό ότι η χρήση ενός url που ακούει σε μια υπηρεσία API μαζί με την χρήση APIKEY, δεν μπορεί να γίνει εφικτή με την χρήση των συνηθισμένων browsers. Την λύση στο πρόβλημα αυτό, την δίνει η εφαρμογή Postman η οποία αποτελεί εργαλείο διαχείρισης, ανάπτυξης και μορφοποίησης APIs.

Στην παραπάνω εικόνα, δίνεται η παραθυρική εφαρμογή της Postman μέσα στην οποία φαίνεται καθαρά η κλήση της ίδιας API της προηγούμενης παραγράφου. Η εφαρμογή αυτή, γίνει την δυνατότητα στους χρήστες, την εισαγωγή περαιτέρω συνθηκών για την αποτελεσματική κλήση APIs, όπως APIKEYs, Headers και λοιπών parameters.




**Εικόνα 14. Χρήση υπηρεσίας API σε περιβάλλον Postman**

### 3.1.3 Thymeleaf**[28]**

Ένα πολύ βασικό κομμάτι για την πραγμάτωση της εφαρμογής της παρούσας διπλωματικής εργασίας, είναι η αποτύπωση των αποτελεσμάτων της αναπτυσσόμενης API πίσω στους χρήστες. Απαιτείται λοιπόν, η δημιουργία ενός front-end τμήματος, το οποίο θα πρέπει να ανταποκρίνεται στις απαιτήσεις και να αποδίδει την πληροφορία ορθά και σωστά. Έχοντας στην σκέψη, ότι το βασικό αντικείμενο της εφαρμογής εναπόκειται στο backend τμήμα, δεν θα δοθεί η ιδιαίτερη βαρύτητα στο front-end, ωστόσο η επικοινωνία μεταξύ αυτών, θεωρείται ένας από τους δυνατούς κρίκους της αλυσίδας, για να επιτευχθούν οι στόχοι του λογισμικού.


**Εικόνα 15. Λογότυπο Thymeleaf**

Ένα λοιπόν, από τα βασικά ερωτήματα που ανέκυψαν κατά την συγγραφή του κώδικα, ήταν ο τρόπος με τον οποίο θα μεταφέρεται η πληροφορία από την βάση δεδομένων προς το frontend τμήμα, με στόχο την αλληλεπίδραση των χρηστών μ αυτήν. Η λύση δόθηκε με την Thymeleaf η οποία αποτελεί μια μοντέρνα server-side template μηχανή, που επιτρέπει το fetch των δεδομένων από την βάση δεδομένων και την αποτύπωση των σε αρχεία HTML. Εκτός αυτού, η συγκεκριμένη μηχανή είναι ικανή στην διαχείριση κώδικα XML, JavaScript και CSS.

Στις ενότητες που θα ακολουθήσουν, θα αναλυθεί ο κώδικας που χρησιμοποιήθηκε για τις επιμέρους υποεφαρμογές και θα φανούν ξεκάθαρα οι επιπτώσεις της συγκεκριμένης μηχανής στα τελικά αποτελέσματα. Όπως θα γίνει αντιληπτό, ο τρόπος γραφής και σύνταξης της συγκεκριμένης μηχανής είναι κοινός με την HTML, με μερικές τροποποιήσεις.

### 3.1.4 Spring Security

Ο τομέας της ασφάλειας, ανέκαθεν διαδραμάτισε κομβικό ρόλο στην επίτευξη εφαρμογών διαδικτύου. Πλέον στη σημερινή εποχή, με τον αριθμό των κυβερνοεπιθέσεων να αυξάνεται δραματικά, η ασφάλεια θεωρείται το κύριο συστατικό δόμησης μιας εφαρμογής προς ανάπτυξη.

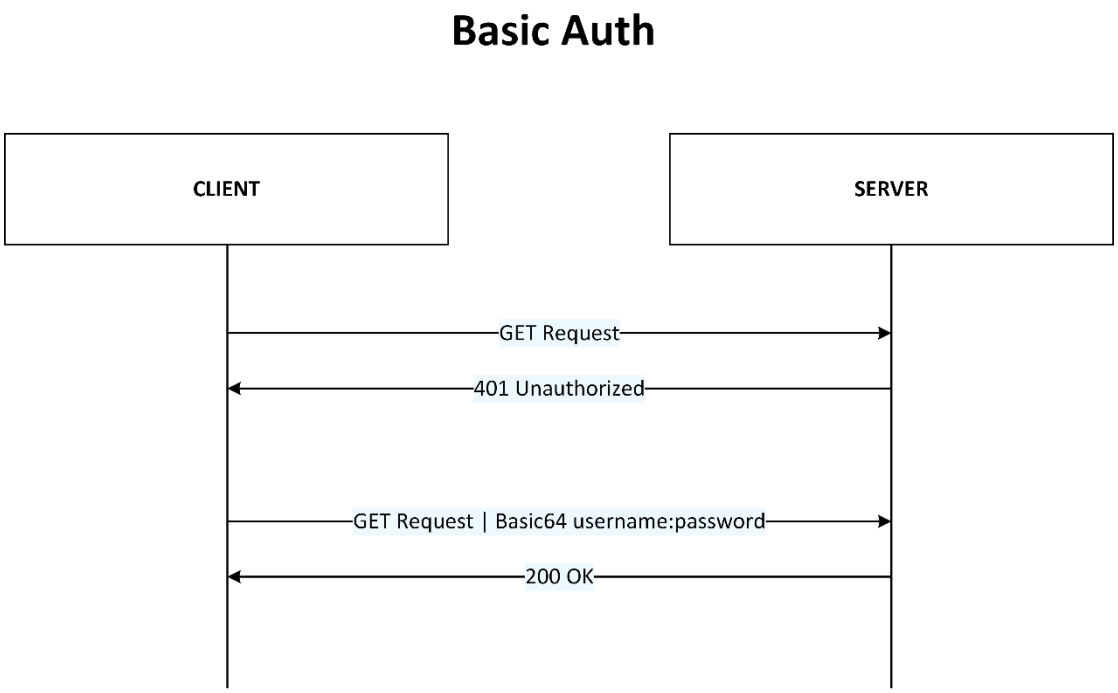
Το τομέα αυτόν, έρχεται το ίδιο το framework να το καλύψει με την εισαγωγή του Spring Security. Το τελευταίο αποτελεί, λοιπόν, ένα αυτοτελές framework που λειτουργεί εντός του Spring, και είναι υπεύθυνο για την παροχή υπηρεσιών ασφαλείας μέσω ελέγχου αυθεντικοποίησης χρηστών (authentication) καθώς και της ανάλογης πρόσβασης αυτών.

**Εικόνα 16. Λογότυπο Spring Security**

Οι βασικοί τρόποι με τους οποίους επιτυγχάνεται ο έλεγχος πρόσβασης και αυθεντικοποίησης στην υπό δημιουργία εφαρμογή αναλύονται στις επόμενες δύο υποενότητες.

#### 3.1.4.1 Basic Auth

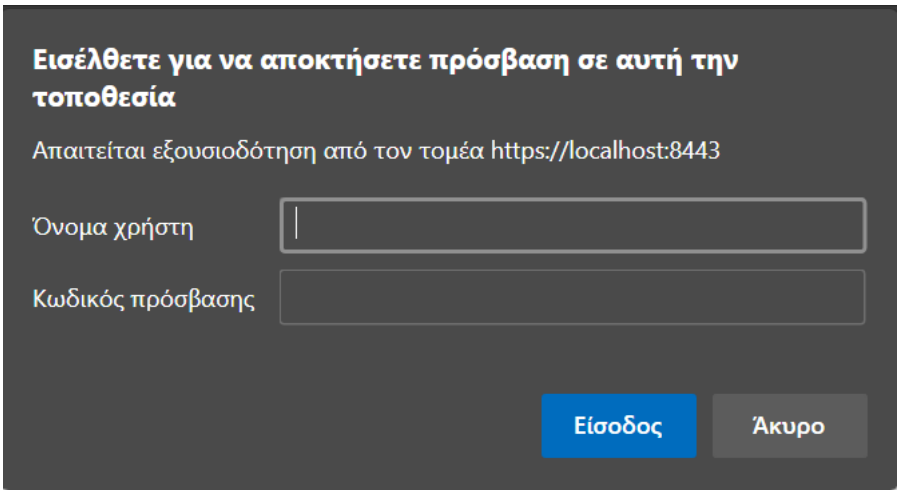
Ο πρώτος τρόπος έγκειται στην εφαρμογή της «Βασικής Αυθεντικοποίησης» (Basic Auth) ενώ στην επόμενη εικόνα απεικονίζεται το διάγραμμα υλοποίησης της.




**Εικόνα 17. Διάγραμμα Υλοποίησης Basic Auth**

Στην περίπτωση αυτή, ο χρήστης απαιτεί την πρόσβαση σε κάποιον διακομιστή με χρήση ερωτημάτων τύπου CRUD. Η υλοποίηση της Basic Auth, θα αποτρέψει την πρόσβαση στον διακομιστή, επιστρέφοντας μια απάντηση τύπου 401, η οποία ανάγεται στην μη εξουσιοδοτημένη πρόσβαση. Προκειμένου η πρόσβαση να πραγματοποιηθεί θα πρέπει στην κεφαλίδα (header) του CRUD ερωτήματος, να μεταφερθεί η πληροφορία του ονόματος χρήστη καθώς και του κωδικού του πρόσβασης υπό κωδικοποίηση Basic64. Με την μεταβίβαση αυτής της πληροφορίας, η Basic Auth, θα επιτρέψει την πρόσβαση στον χρήστη επιστρέφοντας τον πολυπόθητο κωδικό κατάστασης 200.

Κατά την συγγραφή του απαραίτητου κώδικα, ο οποίος θα θέσει σε λειτουργία την Basic Auth, θα προκύψει το αποτέλεσμα της παρακάτω εικόνας. Κατά την πληκτρολόγηση, λοιπόν, του url του ανάλογου server, το Spring Security απορρίπτει την προσπάθεια εισόδου, ζητώντας ταυτόχρονα από τον χρήστη τα ανάλογα διακριτικά για την είσοδο του. Ο κωδικός πρόσβασης παρέχεται από τον IDE και είναι μοναδικός ανά συνεδρία ενώ για το όνομα χρήστη, αρκεί η λέξη «user» για να ολοκληρώσει την διαδικασία σύνδεσης. Σαφώς υπάρχει η δυνατότητα χρήσης κωδικών που είναι αποθηκευμένοι σε βάση δεδομένων.

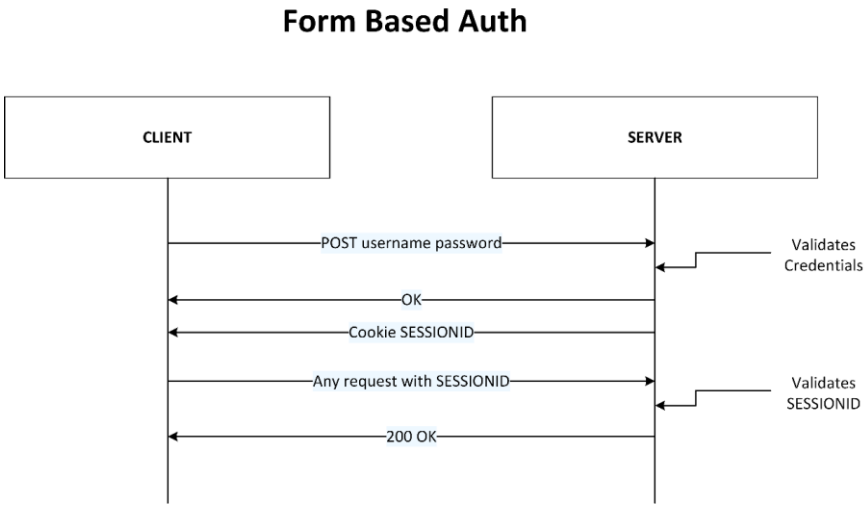



**Εικόνα 18. Η Basic Auth κατά την εφαρμογή της στον browser**

Το βασικό πλεονέκτημα της μεθόδου αυτής ανάγεται στην πολύ απλή και γρήγορη χρήση της. Στον αντίποδα, δεν προσφέρει στους χρήστες την δυνατότητα αποσύνδεσης από την εφαρμογή.

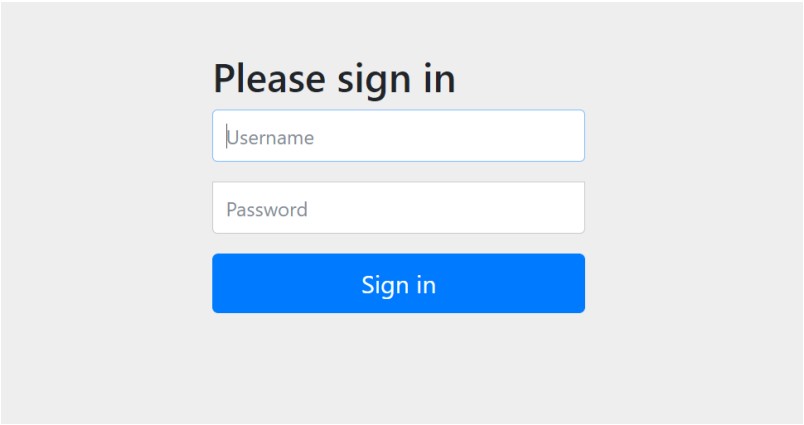
#### 3.1.4.2 Form Based Auth

Θεωρείται αδιαμφισβήτητα ως ο πλέον διαδεδομένος τρόπος προστασίας και ελέγχου εφαρμογών διαδικτύου. Είναι μάλιστα, ο by default ενεργοποιημένος τρόπος προστασίας μιας εφαρμογής, από το ίδιο το framework, απο την στιγμή που θα γίνει η εγκατάσταση του Spring Security και χωρίς να έχει γραφτεί ούτε μία γραμμή κώδικα στο project.




**Εικόνα 19. Διάγραμμα Form Based Auth**

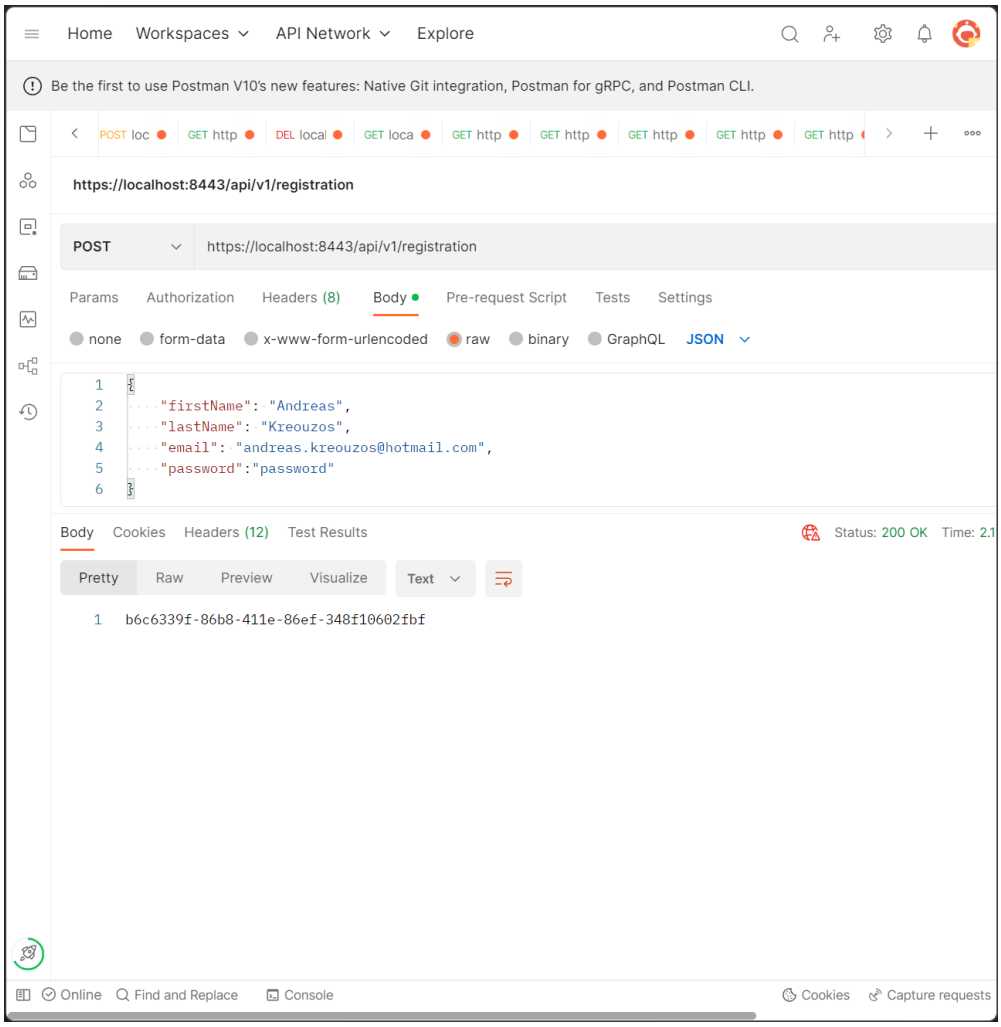
Η περίπτωση αυτή, απαιτεί την δημιουργία ενός ερωτήματος τύπου POST, το οποίο μεταφέρει τα στοιχεία εισόδου χρήστη και κωδικού πρόσβασης. Από την μεριά του διακομιστή, διενεργείται η εγκυρότητα των στοιχείων αυτών και εφόσον αυτά είναι σωστά, αποστέλλεται θετική απάντηση μαζί μ' ένα cookie, στο οποίο εμπεριέχεται ένας μοναδικός κρυπτογραφημένος κωδικός. Ο τελευταίος κατέχει αναγνωριστικό χαρακτήρα και η διάρκεια ζωής του εκτείνεται είτε όσο διαρκέσει η συνεδρία αλληλεπίδρασης του χρήστη με την εφαρμογή ή με το πέρας 30 λεπτών πλήρους αδράνειας. Η παρακάτω εικόνα αποτελεί την φόρμα εισόδου στην εφαρμογή χωρίς να έχει προηγηθεί κάποια παραμετροποίηση από τον χρήστη.




**Εικόνα 20. Η Form Based Auth κατά την εφαρμογή στον browser**

### 3.1.4.3 Ολοκληρωμένο Σύστημα Εισόδου μέσω Email Πιστοποίησης

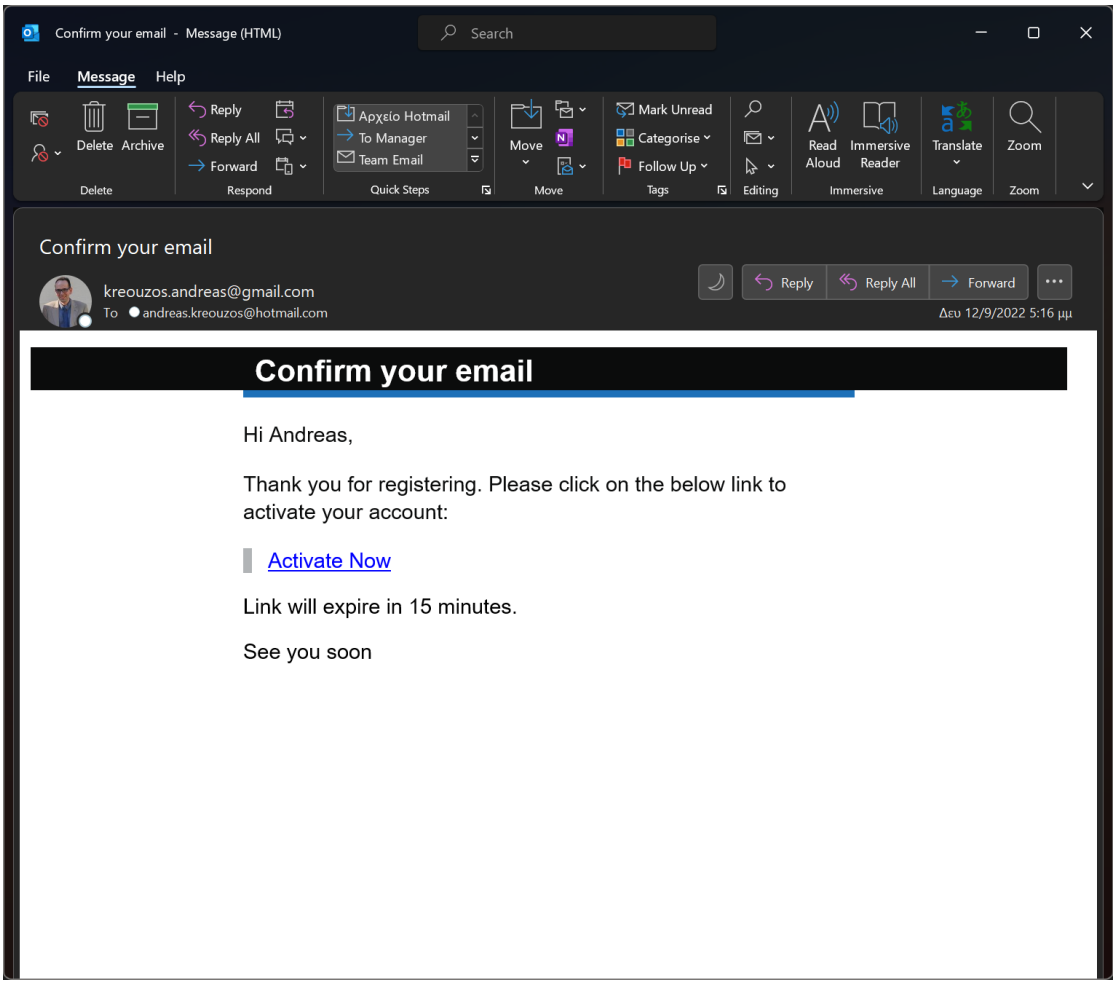
Για τους σκοπούς της παρούσας διπλωματικής εργασίας, επιλέχθηκε η τεχνική «Form Based Auth» ως ο τρόπος ελέγχου εισόδου των χρηστών στην εφαρμογή. Πέραν τούτης, κατασκευάσθηκε σύστημα ελέγχου καταγραφής των στοιχείων των χρηστών, τα οποία εισάγουν οι ίδιοι στην εφαρμογή με στόχο την αποθήκευση τους στη βάση δεδομένων. Για λόγους ευκολότερων δοκιμών, χρησιμοποιήθηκε η εφαρμογή POSTMAN, στην οποία καταχωρήθηκε ως έγγραφο JSON, τα στοιχεία του συγγραφέα φοιτητή, για λόγους καθαρά επίδειξης (παρακάτω εικόνα).




**Εικόνα 21. Δοκιμή εισαγωγής στοιχείων χρήστη μέσω ερωτήματος τύπου POST**

Με την αποστολή των στοιχείων αυτών μέσω ερωτήματος POST, αφενός γίνεται η καταχώρηση τους στη βάση δεδομένων που έχει στηθεί, αφετέρου δημιουργείται ένας μοναδικός αριθμός token, ο οποίος και φαίνεται στην εικόνα της εφαρμογής Postman. Ο τελευταίος είναι μοναδικός και τυχαίος κωδικός ανά χρήστη.

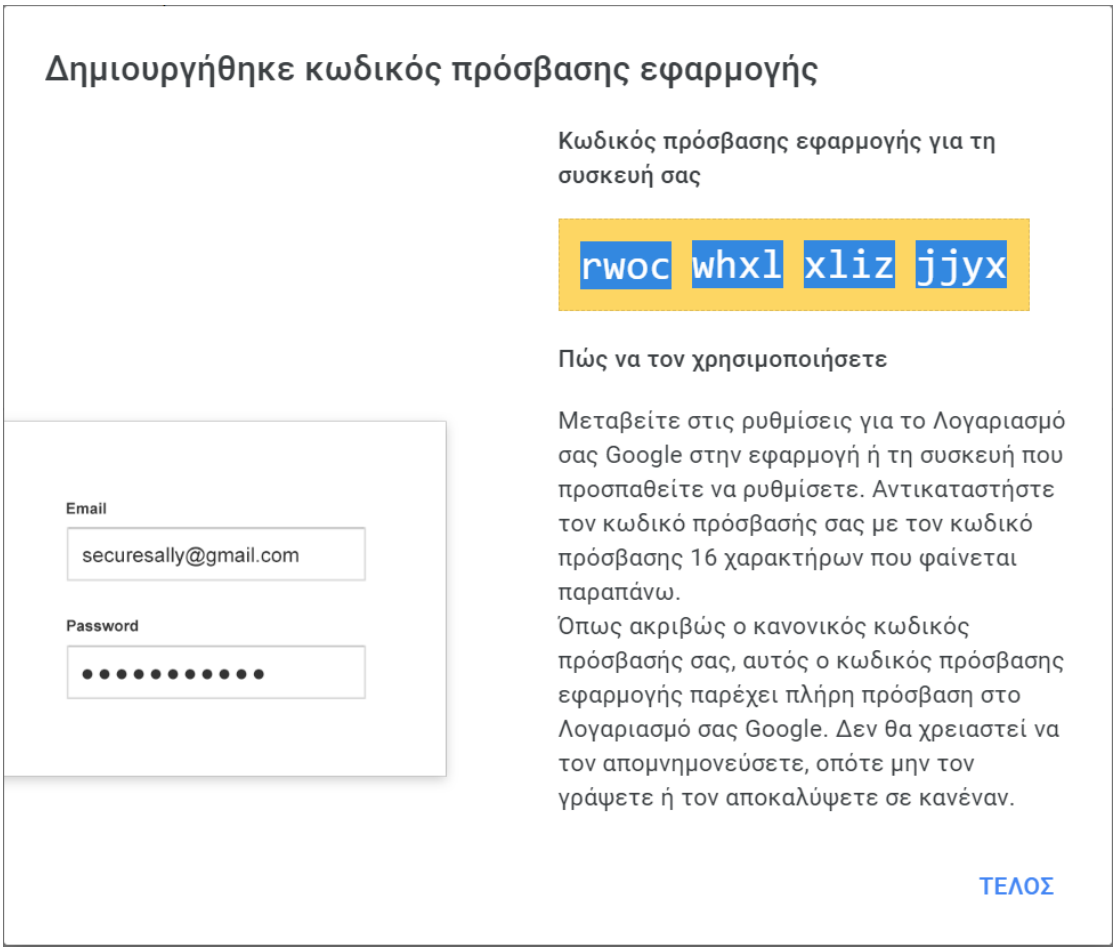
Με την ολοκλήρωση της αναφερθείσας διαδικασίας, το σύστημα θα προβεί στην αποστολή μηνύματος ηλεκτρονικού ταχυδρομείου στην διεύθυνση που έχει εισαχθεί, με σκοπό την τελική επιβεβαίωση της διεύθυνσης αυτής. Το μήνυμα αυτό εμπεριέχει σύνδεσμο τελικής ενεργοποίησης, ο οποίος λήγει έπειτα από 15 λεπτά.




**Εικόνα 22. Απεσταλμένο μήνυμα επιβεβαίωσης δηλωθείσας διεύθυνσης mail**

Για την υλοποίηση του ανωτέρου συστήματος έγινε χρήση του προσωπικού λογαριασμού Gmail του συγγραφέα φοιτητή. Απαιτήθηκαν τροποποιήσεις στο λογαριασμό Google με κυριότερη, αυτή της πρόσδοσης κωδικού, ο οποίος επέτρεψε την επικοινωνία της εφαρμογής Spring Boot με τον λογαριασμό αυτό. Η επόμενη εικόνα παρέχει μια τελική αποτύπωση της διαδικασίας αυτής, με οδηγίες ως προς τον τρόπο χρήσης της.

Με δεδομένο ωστόσο, ότι η εφαρμογή αναφέρεται σε περιβάλλον προγραμματισμού και δει σε Spring framework, απαιτήθηκαν παραμετροποιήσεις στο αρχείο application.properties, οι οποίες φαίνονται στον πίνακα που ακολουθεί.



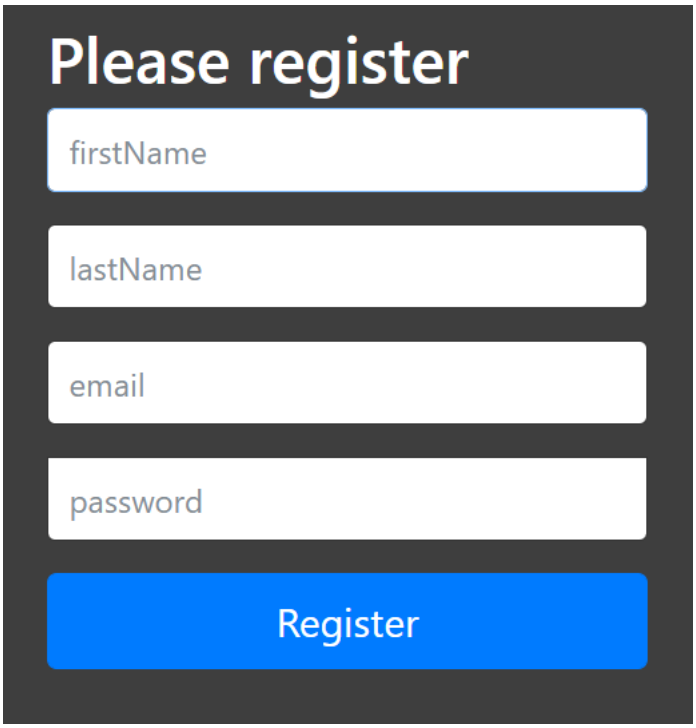

**Εικόνα 23. Κωδικός Πρόσβασης Χρήσης Gmail απο εξωτερική εφαρμογή**

| ΕΝΤΟΛΕΣ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗΣ ΧΡΗΣΗΣ GMAIL |
| --- |
| spring.mail.host=smtp.gmail.com<br>spring.mail.port=587<br>spring.mail.username=kreouzos.andreas@gmail.com<br>spring.mail.password=rwocwhxlxlizjjyx |

**Πίνακας 5. Εντολές Παραμετροποίησης στο αρχείο Application.Properties**

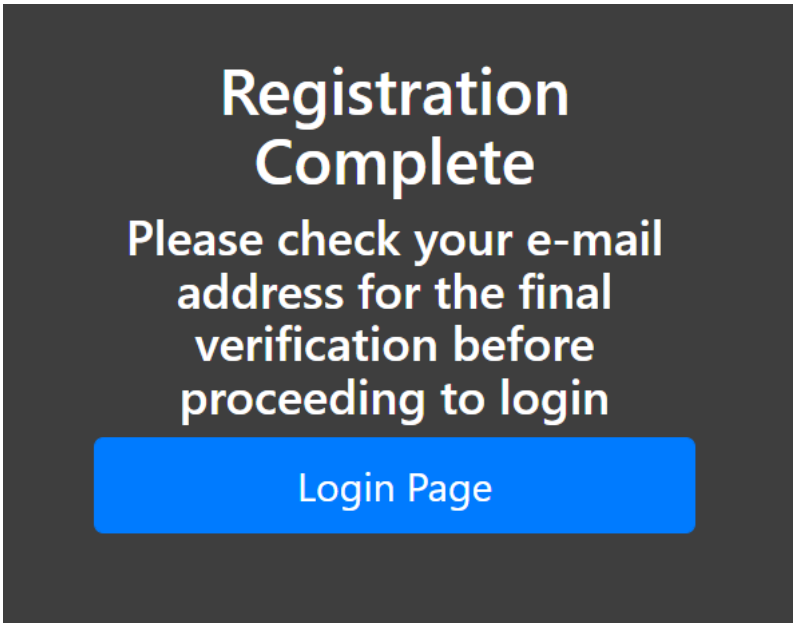
Περαιτέρω παραμετροποιήσεις τέθηκαν σε ισχύ για την ολοκλήρωση της διαδικασίας, οι οποίες αποτυπώνονται εντός του project.

Πριν την τελική παράδοση του project, κρίθηκε σκόπιμη η αντικατάσταση της εφαρμογής Postman για την καταχώρηση των στοιχείων των χρηστών, με view που παραπέμπει σε αρχείο HTML, μέσω κλάσης υπό το annotation @Controller (ενότητα 3.3.2.9). Αντί λοιπόν, τα στοιχεία του εκάστοτε χρήστη να καταχωρούνται με τον τρόπο της εικόνας 22, έγινε χρήση αρχείου HTML, όπως φαίνεται στην εικόνα 25, για την καταχώρηση αυτών σε φόρμα.



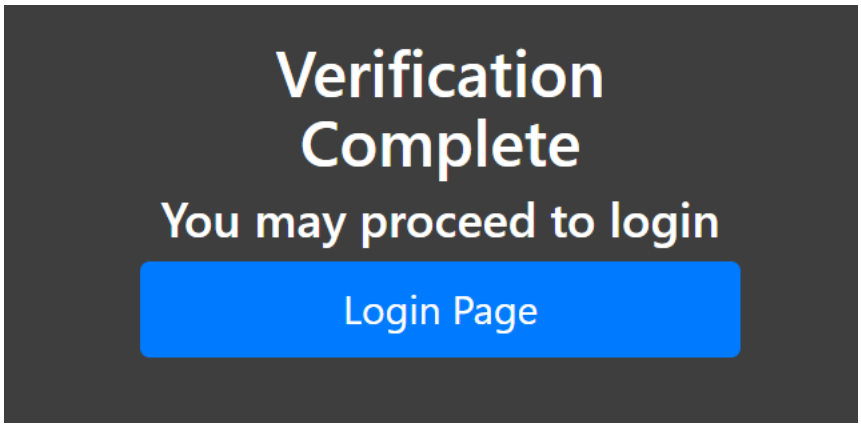

**Εικόνα 24. Φόρμα HTML δήλωσης στοιχείων χρήστη**

Με την ολοκλήρωση της καταχώρησης των τελευταίων, ο χρήστης μπορεί με πάτημα του πλήκτρου «Register», να προχωρήσει στο επόμενο στάδιο, που αφορά την ενεργοποίηση της διεύθυνσης mail που δήλωσε.




**Εικόνα 25. Φόρμα HTML ενημέρωσης περί ολοκλήρωσης καταχώρησης στοιχείων χρήστη**

Η εικόνα 26, δίνει την ενημέρωση της επιτυχούς καταχώρησης αλλά ταυτόχρονα και την διαδικασία ελέγχου του mailbox εκ μέρους του χρήστη για να ενεργοποιηθεί το τελευταίο. Το πλήκτρο «Login Page», δίνει τυπικά την επιλογή δρομολόγησης στην αρχική σελίδα σύνδεσης της εφαρμογής.




**Εικόνα 26. Φόρμα HTML ενημέρωσης επιτυχούς ενεργοποίησης διεύθυνσης mail**

Εφόσον ο χρήστης εντοπίσει το mail ενεργοποίησης του λογαριασμού του (εικόνα 22), το πάτημα της επιλογής «Activate Now» θα οδηγήσει τον τελευταίο στο αρχείο HTML της εικόνας 26. Το άνοιγμα της φόρμας αυτής, θα γίνει στον προεπιλεγμένο φυλλομετρητή διαδικτύου του υπολογιστή του χρήστη μέσω redirection από το mail ενεργοποίησης.

### 3.1.5 Βιβλιοθήκη Lombok

Το Project Lombok ή απλά Lombok, αποτελεί μια πολύ ισχυρή βιβλιοθήκη της Java, βασικός σκοπός της οποίας είναι η μείωση του επαναλαμβανόμενου και δεδομένου (boilerplate) κώδικα, που απαρτίζει μια εφαρμογή κατά Spring Boot. Ένα βασικό παράδειγμα χρήσης της είναι στις κλάσεις POJO, όπου επιτρέπει την διαγραφή του κώδικά των μεθόδων getters και setters όπως και την toString(), με ταυτόχρονή αντικατάσταση του από annotations, τα οποία προσφέρουν τις ίδιες λειτουργίες. Με την δημιουργία της, να ανάγεται εκτός του οικοσυστήματος του Spring αλλά με μια ισχυρή κοινότητα, η οποία την υποστηρίζει, αποτελεί βασικό εργαλείο για την ανάπτυξη εφαρμογών. Μερικά από τα annotations που χρησιμοποιεί είναι τα ακόλουθα:

**@Getter**

Όταν μια κλάση χαρακτηρισθεί με το συγκεκριμένο annotation, η Lombok θα προβεί στην αυτόματη δημιουργία των μεθόδων Getter.

**@Setter**

Όταν μια κλάση χαρακτηρισθεί με το συγκεκριμένο annotation, η Lombok θα προβεί στην αυτόματη δημιουργία των μεθόδων Setter. Σημειώνεται ότι η κλάση μπορεί να λάβει και τα δύο προαναφερθέντα annotations.

**@NoArgsConstructor**

Στην περίπτωση του annotation αυτού, θα δημιουργηθεί από την Lombok, ο default constructor, ο οποίος ως γνωστόν δεν λαμβάνει παραμέτρους εντός.

**@AllArgsConstructor**

Το annotation συγκεκριμένο θα κατασκευάσει τον constructor της κλάσης έχοντας εντός του, όλες τις μεταβλητές που έχουν δηλωθεί στην POJO κλάση.

**@RequiredArgsConstructor**

Τέλος το annotation αυτό οδηγεί στην δημιουργία constructor με παραμέτρους που απαιτούν ειδικό χειρισμό. Μεταβλητές που έχουν δηλωθεί με το keyword «final» αποτελούν τις πλέον υποψήφιες ως παραμέτρους στον τελικό constructor.

**@ToString**

Ακόμα ένα τμήμα δεδομένου κώδικα κατά την δημιουργία μιας κλάσης POJO είναι η μέθοδος toString(). Το annotation αυτό προβαίνει στη δημιουργία του.

### 3.1.6 HTTPS & SSL**[29]** , **[30]**

Θεωρώντας πλέον την χρήση του Spring Security δεδομένη, θεωρήθηκε σημαντική η χρήση της επέκτασης του πρωτοκόλλου HTTP, ήτοι του HTTPS. Ως γνωστόν το τελευταίο χρησιμοποιείται για την διασφάλιση της επικοινωνίας μεταξύ των διαφόρων υπολογιστών μέσα σ' ένα δίκτυο. Στην περίπτωση της παρούσας εφαρμογής, η επικοινωνία θα υλοποιηθεί με HTTPS ενώ η κρυπτογράφηση θα γίνει με την βοήθεια του Secure Sockets Layer (SSL). Τα βασικά στάδια, που έλαβαν στην επίτευξη του συγκεκριμένου στόχου αναλύονται παρακάτω.


**Εικόνα 27. Δημιουργία SSL Πιστοποιητικού**

1. Δημιουργία πιστοποιητικού τύπου SSL προσωπικά υπογεγραμμένο από τον συγγραφέα φοιτητή (self-signed SSL Certificate). Ο συγκεκριμένος τρόπος ενδείκνυται όταν η εφαρμογή βρίσκεται στο στάδιο της ανάπτυξης και επίδειξης. Στην περίπτωση που αυτή οριστικοποιηθεί, θα πρέπει πριν την παράδοση σε τελικό διακομιστή, να γίνει χρήση SSL πιστοποίησης από κάποιον αρμόδιο φορέα (Certificate Authority). Το στάδιο αυτό περιλαμβάνει την δημιουργία ενός key pair αρχείου με την βοήθεια της εφαρμογής Keytool που περιλαμβάνεται μέσα στο JDK της γλώσσας JAVA. Το αρχείο αυτό αποτελεί ένα ζεύγος κλειδιών κρυπτογράφησης, το οποίο χρησιμοποιείται για την παραγωγή του SSL πιστοποιητικού μέσω της αποθήκευσης του σε μία βάση δεδομένων ονόματι Keystore. Το τελευταίο θα ενσωματωθεί στην εφαρμογή για να ενεργοποιήσει το πιστοποιητικό σ αυτήν. Η παραπάνω εικόνα αποτυπώνει μέρος της διαδικασίας, ενώ στον παρακάτω πίνακα δίνονται οι εντολές που δόθηκαν στον terminal για την δημιουργία των ανωτέρω.

| ΕΝΤΟΛΗ ΔΗΜΙΟΥΡΓΙΑΣ KEY PAIR |
|---|
| keytool -genkeypair -alias local_ssl -keyalg RSA -keysize 2048 -storetype PKCS12 -keystore local-ssl.p12 -validity 365 -ext san=dns:localhost |
| ΕΝΤΟΛΗ ΕΞΑΓΩΓΗΣ SSL ΠΙΣΤΟΠΟΙΗΤΙΚΟΥ |
| keytool -export -keystore local-ssl.p12 -alias local_ssl -file local-cert.crt |

**Πίνακας 6. Εντολές Command Prompt για SSL Πιστοποίηση**

2. Παραμετροποίηση της αναπτυσσόμενης εφαρμογής στον IDE για την μετατροπή του localhost server από HTTP σε HTTPS. Το στάδιο αυτό περιλαμβάνει την τοποθέτηση του εξαγόμενου αρχείου του προηγούμενου βήματος εντός της εφαρμογής.

| ΕΝΤΟΛΕΣ ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗΣ ΧΡΗΣΗΣ SSL |
|---|

```
server.ssl.enabled=true
server.ssl.key-store-type=PKCS12
server.ssl.key-store=classpath:local-ssl.p12
server.ssl.key-store-password=
server.ssl.key-password=
server.servlet.context-path=/
server.ssl.key-alias=local_ssl
server.port=8443
```

**Πίνακας 7. Εντολές στο αρχείο application.properties**

Επίσης πραγματοποιήθηκαν αλλαγές και στο αρχείο ιδιοτήτων της εφαρμογής (application.properties), προκείμενου το συγκεκριμένο αρχείο να μπορεί να διαβαστεί από αυτήν. Υλοποιήθηκε επίσης αλλαγή της θύρας δοκιμής του διακομιστή από 8080 σε 8443.

3. Δημιουργία και εγκατάσταση πιστοποιητικού στον τοπικό υπολογιστή, όπου και δημιουργήθηκε η εφαρμογή, έτσι ώστε κατά την εκτέλεση της, ο browser να εμπιστεύεται το τρέχων πιστοποιητικό και κατ' επέκταση την εκτέλεση της εφαρμογής. Η διαδικασία αυτή ολοκληρώνεται με την εξαγωγή του πιστοποιητικού από το Keystore.

## 3.2 Δομικός Σκελετός (Backbone)

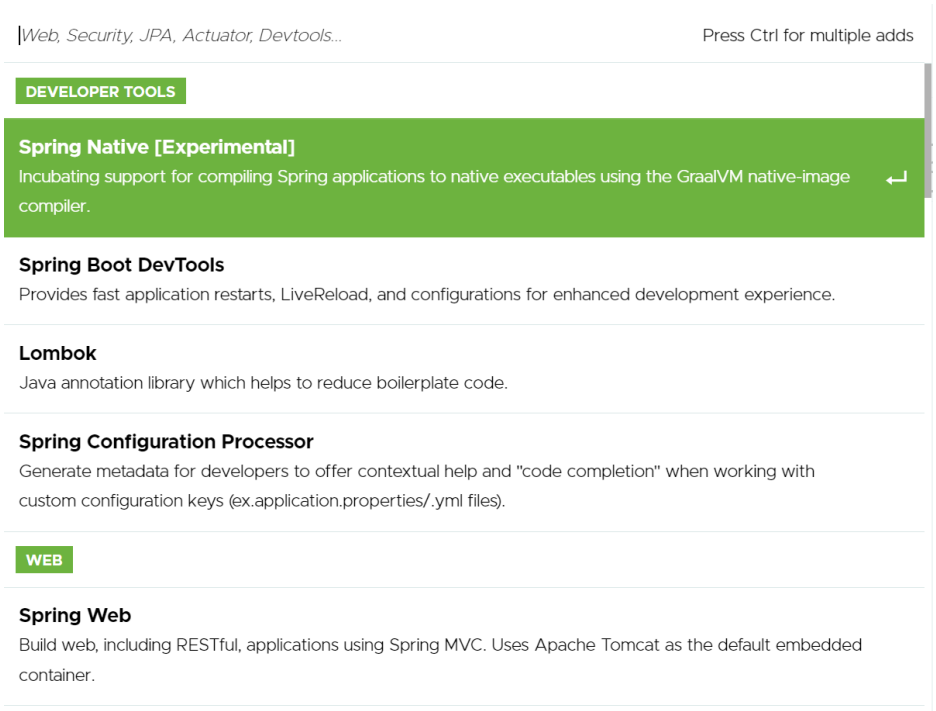
Στην ενότητα αυτή, αναλύεται το σώμα του project όπως αυτό περιγράφηκε στην ενότητα 2.3.2. Αρχής γενομένης με το τμήμα εισόδου των δεδομένων, γίνονται επεξηγήσεις των πακέτων και των κλάσεων που θα δημιουργηθούν και τα οποία θα δομήσουν το τελικό εγχείρημα. Έτσι, λοιπόν, η ενότητα αυτή θα χωριστεί σε επιμέρους υποενότητες με σκοπό την επεξήγηση των όσων διαδραματίζονται ανά στρώμα επικοινωνίας και διαχείρισης κώδικα.

### 3.2.1 Spring Initializr

Για την εκκίνηση του project, γίνεται χρήση της πλατφόρμας του Spring Framework ήτοι το Spring Initializr (https://start.spring.io/), μια εικόνα του οποίου δίνεται στην ενότητα 1.2. Στο περιβάλλον αυτό, έγιναν οι κάτωθι επιλογές:

1. Τύπος Project: Maven
2. Γλώσσα Προγραμματισμού: Java
3. Spring Boot Version: 2.7.0
4. Project Metadata
    a. Group: com.andrekreou
    b. Artifact: iot
    c. Name: iot
    d. Description: MSc Thesis Project
    e. Package Name: com.andrekreou.iot (Autocompleted)
5. Packaging: JAR
6. Java Version: 17

Εν συνεχεία θα πρέπει να προστεθούν οι εξαρτήσεις (dependencies), οι οποίες αποτελούν το δομικό σκελετό λειτουργιών του project. Στο σημείο αυτό, η μόνη εξάρτηση που τοποθετήθηκε ήταν το Spring Web, το οποίο επιτρέπει την κατασκευή και δοκιμή, εφαρμογών διαδικτύου RESTful με ενσωματωμένη την server λειτουργία του Apache Tomcat.




**Εικόνα 28. Επιλογή dependencies (Πηγή: https://start.spring.io/)**

Ο λόγος που δεν επιλέχθηκαν περισσότερα dependencies ανάγεται σε δύο κύρια γεγονότα:

1. Η λίστα των dependencies είναι πολύ μεγάλη και αυτό αποτυπώνει την δημοφιλία του Spring ως framework στην αγορά του software development. Ταυτόχρονα όμως με την έλλειψη εμπειρίας του συγγραφέα φοιτητή, θεωρήθηκε καλό να αποφευχθεί, η είσοδος τεχνολογιών που θα μετέτρεπαν το project πιο βαρύ στην εκτέλεση του.
2. Μπορεί να επιτραπεί και περαιτέρω εισαγωγή dependencies κατά την διάρκεια υλοποίησης του project στον IntelliJ.

Για λόγους καλύτερης οπτικοποίησης του κώδικα που θα αναπτυχθεί, θα γίνει χρήση της βιβλιοθήκης Lombok, η οποία βοηθά στην απλοποίηση του κώδικα αφαιρώντας μεθόδους όπως των getters and setters καθώς και των constructors των κλάσεων που θα δημιουργηθούν. Για την βιβλιοθήκη αυτή, γίνεται αναφορά στην ενότητα 3.1.5. των λειτουργιών της, με τα ανάλογα annotations και τα οφέλη που προκύπτουν.

### 3.2.2 Δομικό Στήσιμο στον IntelliJ IDEA

Η ολοκλήρωση του προηγούμενου βήματος, δίνει την δυνατότητα στον εκάστοτε χρήστη να εξάγει το project από την online πλατφόρμα σε αρχείο τύπου zip. Εφόσον το αρχείο αυτό αποσυμπιεσθεί, θα πρέπει να εισαχθεί στον IDE που έχει επιλεχθεί. Στην περίπτωση του παρόντος project, θα είναι ο IntelliJ IDEA.

Ακολουθώντας την N Tier Architecture δόμηση, το project θα διασπαστεί σε επιμέρους πακέτα (packages), καθένα από τα οποία θα αντιπροσωπεύει μια λειτουργία της συνολικής εφαρμογής.

#### 3.2.2.1 Main Application Package

Το συγκεκριμένο πακέτο δημιουργήθηκε ήδη από τον συγγραφέα φοιτητή μέσω της online πλατφόρμας Spring Initializr και η ονομασία που επιλέχθηκε ήταν η com.andrekreou.iot. Μέσα στο συγκεκριμένο πακέτο θα πρέπει να δομηθούν οι τρεις υποεφαρμογές που αναλύθηκαν στις ενότητες 3.1.1, 3.1.2 και 3.1.3.

#### 3.2.2.2 Secondary Application Packages

Εφόσον έχουν τεθεί τα θεμέλια αρχικοποίησης της εφαρμογής της εφαρμογής, με την δημιουργία του main πακέτου, σειρά έχουν τα πακέτα των τριών υπό εφαρμογών. Στο πλαίσιο αυτό, θα κατασκευασθούν 3 πακέτα εντός του main πακέτου, ένα ανά εφαρμογή και με ονομασία κατάλληλη, ως προς την διαφοροποίηση. Προς διευκόλυνση της κατανόησης των ανωτέρω,

| | |
|---|---|
| com.andrekreou.iot | |
| - | crypto |
| - | movies |

**Πίνακας 8. Διάγραμμα Πακέτων Εφαρμογής**

Με την ολοκλήρωση της διαδικασίας φακελοποίησης, όπως αυτή σημειώνεται στον παραπάνω πίνακα, θα λάβει χώρα η δημιουργία νέων υπό πακέτων ανά εφαρμογή, μέσα στα οποία θα στεγαστούν οι κλάσεις που θα δομήσουν την λειτουργία τους. Κάθε εφαρμογή που υλοποιείται σε περιβάλλον Spring Boot, συνηθίζεται να έχει τις εξής κλάσεις σε δικά τους πακέτα:

1. Controller Classes
2. Configuration Classes
3. POJO ή Model Classes
4. Repository Interfaces
5. Service Classes

Οι υποενότητες που θα ακολουθήσουν θα αναλύσουν τις λειτουργίες των ανωτέρω κλάσεων με μεγαλύτερη σαφήνεια.

#### 3.2.2.3 Controller Class

Στο περιβάλλον του Spring Boot, μια Κλάση Ελέγχου HTTP Ερωτημάτων, είναι υπεύθυνη για την διαχείριση των εισερχόμενων ερωτημάτων που θέτονται απο τους χρήστες στην είσοδο της υπηρεσίας. Επίσης εφόσον διαχειριστούν τα ερωτήματα αυτά, θα πρέπει να προετοιμάσει το μοντέλο πληροφορίας της εφαρμογής καθώς επίσης και την τελική προβολή αυτού πίσω στους χρήστες, ως απάντηση στο αρχικό τους ερώτημα.

Στη περίπτωση του παρόντος project, όπου η αναπτυσσόμενη εφαρμογή υλοποιείται με χρήση browser, η κλάση ελέγχου απαντά παραθέτοντας τα μοντελοποιημένα δεδομένα και μεταφέροντας τα σε μία τελική προβολή HTML, η οποία και επιστρέφεται στον εκάστοτε browser.

#### 3.2.2.4 POJO / Model Class

Όπως αναλύθηκε σε προηγούμενες παραγράφους, η ανάπτυξη της παρούσας διατριβής περιλαμβάνει την δημιουργία τριών υπό εφαρμογών. Για λόγους ορθής υλοποίησης θέλοντας να γίνει χρήση πραγματικών δεδομένων, πραγματοποιείται κλήση εξωτερικών APIs, προκειμένου αυτά να παρθούν ως είσοδος στην εφαρμογή του συγγραφέα φοιτητή. Τα εξωτερικά δεδομένα, θα ληφθούν υπό μορφή JSON όπως αναλύθηκε στην ενότητα 3.2.1. Ένα πραγματικό παράδειγμα δεδομένων σε μορφή JSON (https://bitpay.com/api/rates) απεικονίζεται στον πίνακα 8.

Όπως γίνεται αντιληπτό, μια συστοιχία JSON αποτελείται από κλειδιά και τις αξίες αυτών. Στην περίπτωση του κάτωθι πίνακα, οι λέξεις «code», «name» και «rate», αποτελούν κλειδιά ενώ οι αξίες αυτών αναγράφονται ξεχωριστά ανά αντικείμενο της συστοιχίας JSON. Ο διαχωρισμός των αντικειμένων γίνεται με άγκιστρα. Ο κύριος σκοπός που πρέπει να επιτευχθεί είναι η αντιγραφή των δεδομένων των κλειδιών και της αξίας αυτών, εντός της υπό ανάπτυξης εφαρμογής. Κύριο αρωγό στην διαδικασία αυτή αποτελούν κλάσης μοντελοποίησης ή POJO class.

```
[
    {
        "code": "BTC",
        "name": "Bitcoin",
```

```
        "rate": 1
    },
    {
      "code": "BCH",
      "name": "Bitcoin Cash",
      "rate": 161.43}
]
```


**Πίνακας 9. Δεδομένα JSON απο την BitPay.com**

Η λέξη POJO αποτελεί αρχικά της έκφρασης Plain Old Java Object και ορίζεται ως ένα απλό αντικείμενο στη γλώσσα Java, το οποίο χρησιμοποιείται για την ευκολότερη αναγνωσιμότητα και χρησιμότητα ενός προγράμματος. Η βασική του όμως λειτουργία ανάγεται στην δυνατότητα ορισμού μια οντότητας δεδομένων, η οποία οντότητα μπορεί να καταχωρηθεί σε βάση δεδομένων.

Η κλάση αυτή επομένως, είναι υπεύθυνη για την χαρτογράφηση των κλειδιών και των αξιών των δεδομένων από τα έγγραφα τύπου JSON που θα εισαχθούν στις υπό εφαρμογές. Η αναφερθείσα διαδικασία, μάλιστα, δεν είναι απαραίτητο να γίνει για όλα τα κλειδιά, παρά σε όσα μόνο ο προγραμματιστής επιθυμεί. Στον πίνακα 9, παρατίθεται ένα σύντομο στιγμιότυπο μιας κλάσης POJO για το προαναφερθέν έγγραφο JSON του πίνακα 8.

```
public class Crypto
{
   public String code;
   public String name;
   private double rate;

   //Create argument constructor for field initialization
```

```
    //Create getters and setters
}
```

**Πίνακας 10. Στιγμιότυπο κώδικα κλάσης POJO για το JSON της BitPay**

Στο σημείο αυτό είναι σημαντικό να αναφερθεί ότι η POJO κλάση περικλείει την προγραμματιστική λογική και επικοινωνία που λαμβάνει από την κλάση υπηρεσίας (Service Class), η οποία αναφέρεται σε παρακάτω ενότητα. Η επικοινωνία ξεκινά από την Controller Class, συνεχίζει στην Service, μετά στην POJO και από κει καταλήγει στην βάση δεδομένων. Τέλος είναι εξαιρετικής σημασίας το γεγονός ότι τα ονόματα των μεταβλητών στην κλάση αυτή, θα πρέπει να είναι τα ίδια μ αυτά των κλειδιών των δεδομένων στα αρχεία JSON. Ο πίνακας 9 αποτυπώνει ακριβώς το λεγόμενο αυτό, εν συναρτήσει το έγγραφο JSON του πίνακα 8.

#### 3.2.2.5 Configuration Class

Κατά την δημιουργία ενός project σε περιβάλλον Spring Boot, ο προγραμματιστής καλείται να δημιουργήσει τα λεγόμενα «Beans». Τα τελευταία, όπως αναλύθηκαν στην ενότητα 1.2, αποτελούν τα εργαλεία, η χρήση των οποίων δομούν το οικοδόμημα της εφαρμογής. Μια κλάση τύπου Configuration, υποδηλώνει ότι έχει την δυνατότητα κλήσης και εκμετάλλευσης ενός ή περισσοτέρων Bean μεθόδων. Η διαχείριση της κλάσης αυτής μπορεί να γίνει από το Spring Application Context, για την αναπαραγωγή των ορισμών των μεθόδων των «Bean» καθώς και των ερωτημάτων στα οποία αυτά αλληλοεπιδρούν κατά την εκτέλεση της εφαρμογής.

Μια κλάση που παρέχεται κατά την δημιουργία ενός Spring Boot project είναι η κεντρική κλάση διαχείρισης και εκκίνησης «Main». Συνίσταται ωστόσο βάσει του επίσημου documentation, να γίνεται αποφυγή μιας τέτοιας κίνησης και να ενσωματώνεται η λογική αυτή σε απομονωμένη κλάση. Όπως θα φανεί στη συνέχεια, στη συγκεκριμένη κλάση γίνεται η εκτέλεση των εντολών για την εκκίνηση συλλογής των δεδομένων από τις εξωτερικές API και η αποθήκευση τους στη βάση δεδομένων μέσω instantiation των διαφόρων «beans» που έχουν κατασκευασθεί.

#### 3.2.2.6 Repository Interface

Η ύπαρξη βάσης δεδομένων στην υπό ανάπτυξη θεωρείται απολύτως δεδομένη όπως και στο σύνολο σχεδόν των applications που αναπτύσσονται παγκοσμίως. Η ύπαρξη επομένως μιας αντίστοιχης κλάσης για τον χειρισμό και την επικοινωνία των δεδομένων με τη βάση, θεωρείται

ειλημμένη. Το κενό αυτό έρχεται, λοιπόν, να καλυφθεί από ένα interface, το οποίο μέσω της κληρονομικότητας λαμβάνει μεθόδους και λειτουργίες χειρισμού δεδομένων σε βάση.

Με βάση την τελευταία θεώρηση, το interface αυτό θα επικοινωνεί με το Spring Data JPA μέσω του JpaRepository interface, που αποτελεί τον συνδετικό κρίκο για την διαχείριση των δεδομένων που επιθυμεί ο χρήστης, να εισάγει στην βάση δεδομένων.

#### 3.2.2.7 Service Class

Μια κλάση, η οποία χαρακτηρίζεται ως Service, κατέχει τον σημαντικό ρόλο του business logic, εντός μια εφαρμογής σε περιβάλλον Spring Boot. Επί της ουσίας, επιτελεί χρέη ενός παρόχου υπηρεσιών με σκοπό την εξυπηρέτηση κάποιων λειτουργιών που διέπονται κατά την ανάπτυξη της εκάστοτε εφαρμογής.

Στα πλαίσια της εφαρμογής της παρούσας διπλωματικής, το business logic που λαμβάνει χώρα αφορά κυρίως στην δημιουργία μεθόδων για την συλλογή των δεδομένων σε λίστες και η απεικόνιση τους στους χρήστες με την βοήθεια των controller κλάσεων.

#### 3.2.2.8 Global Exception Handling Classes

Στα πλαίσια διαχείρισης των exceptions που θα προκύψουν στην εφαρμογή, έγινε η δημιουργία σε ξεχωριστό package, δύο κλάσεων προς εξυπηρέτηση του σκοπού αυτού. Η πρώτη εξ αυτών, είναι υπεύθυνη για την κατασκευή του response μηνύματος που θα λαμβάνει ο χρήστης για την ενημέρωση του, ως προς τι πήγε λάθος. Το μήνυμα αυτό, αντικαθιστά την εμφάνιση του exception και εντός του IntelliJ IDEA. Θα πρέπει όμως να γίνει η πρακτική του υλοποίηση και εδώ έρχεται η δεύτερη κλάση.

Στη κλάση αυτή, συγκεντρώνονται όλες οι εξαιρέσεις (exceptions) και υποδηλώνονται ανά μέθοδο ξεχωριστά η μία από την άλλη. Μέσα στην κλάση αυτή, θα κατασκευασθεί και μία μέθοδος τύπου «Response Entity», όπου σα παράμετρο της θα δέχεται το αντικείμενο response της πρώτης κλάσης. Το τελευταίο θα γίνεται instantiate και inject ως μήνυμα λάθους ανά μέθοδο χειρισμού όλων των exceptions. Στην ενότητα 3.2.2.10, αναγράφονται όλα τα annotations που χρησιμοποιήθηκαν για την κατασκευή των κλάσεων αυτών.

#### 3.2.2.9 JUnit Testing

Στα πλαίσια υλοποίησης της εφαρμογής έγιναν προσπάθειες εκτέλεσης tests στις κλάσεις των Controller, Service καθώς και στην Email Validation. Η τελευταία κλάση ελέγχει την εγκυρότητα μορφής της εισαγόμενης διεύθυνσης mail του χρήστη κατά την διαδικασία του registration. Η εγκυρότητα ελέγχεται μέσω του προτύπου Regex RFC822. Ο κώδικας των tests ακολουθεί την φιλοσοφία ελέγχου των μεθόδων που είτε έχουν ήδη αναπτυχθεί, είτε θα αναπτυχθούν εφόσον ακολουθηθεί η μέθοδος Test Driven Development.

Στην περίπτωση των κλάσεων controller, γίνεται η δημιουργία ενός αντικειμένου κλάσης MockMvc, το οποίο έχει την ικανότητα μέσω της μεθόδου του υπό το όνομα «perform», να παρακολουθεί την αλληλουχία και διαδοχή των HTML σελίδων, να επιτηρεί τα status codes που επιστρέφονται από αυτές και μέσω της μεθόδου «Expect», να αναμένει ένα συγκεκριμένο αποτέλεσμα που θα ορισθεί από τον εκάστοτε developer.

Για την κλάση πιστοποίησης της διεύθυνσης mail, κατασκευάσθηκε ένα test, το οποίο στην είσοδο του, τοποθετεί μια έγκυρη διεύθυνση mail και μέσω της μεθόδου «assert True», εκτελεί την productive μέθοδο της κλάσης για να πιστοποιήσει ότι η είσοδος είναι συμβατή με το πρότυπο RFC822.

Τέλος για τις κλάσεις τύπου Service, έγινε έλεγχος των μεθόδων επικοινωνίας τους με τις κλάσεις repositories με την χειροκίνητη είσοδο δεδομένων από τον συγγραφέα φοιτητή. Στο πρώτο είδος test, ελέγχεται αν η hardcoded είσοδος έχει πράγματι σωθεί μέσω της μεθόδου «save», των repository κλάσεων, ενώ στο δεύτερο test, ελέγχεται αν στην δημιουργηθείσα καταχώρηση, εμπεριέχεται πράγματι ένα τμήμα τύπου String της εισόδου. Με δεδομένο ότι εμπλέκεται η τεχνική dependency injection, έγινε χρήση ειδικών annotations που περιγράφονται στην ενότητα 3.2.2.10.

#### 3.2.2.10 Dependencies (POM.XML)

Το αρχείο Project Object Model, αποτελεί το βασικό αρχείο δόμησης ενός project τύπου Maven, σύμφωνα με την επίσημη ιστοσελίδα του ινστιτούτου Apache[23]. Η γλώσσα που χρησιμοποιείται, είναι η XML, και μέσω αυτής γίνεται η ενημέρωση προς το build system του Maven, των παραμέτρων που θα χρησιμοποιηθούν σ αυτό. Αξίζει να αναφερθεί ότι το Maven αποτελεί το θεμέλιο πάνω στο οποίο θα στηθεί το υπόλοιπο οικοδόμημα της εφαρμογής.

Εν αρχή, το αρχείο αυτό παραθέτει την έκδοση της γλώσσας XML, την προέλευση του με τοποθέτηση του ανάλογου url προς το ινστιτούτο Apache, ενώ μέσα στην συνθήκη <parent> τίθενται ο αριθμός έκδοσης του Spring Boot, που έχει χρησιμοποιηθεί. Από το πεδίο <groupId> έως και το </description>, απεικονίζονται τα δεδομένα που τοποθετήθηκαν μέσω της πλατφόρμας Spring Initializr ενώ στο πεδίο <properties> γίνεται εγγραφή της έκδοσης της γλώσσας Java που χρησιμοποιήθηκε.

Οι εξαρτήσεις, δηλαδή τα dependencies που θα χρησιμοποιηθούν στο project ακολουθούν αμέσως μετά. Το πρώτο dependency, αφορά το Spring Web, που επιλέχθηκε κατά την διαδικασία του Spring Initializr, το οποίο, όπως ειπώθηκε, παρέχει την δυνατότητα δημιουργίας web εφαρμογών και την χρήση του Tomcat server, μεταξύ άλλων για την λειτουργία των. Όπως αναφέρθηκε και σε προηγούμενη παράγραφο, δεν επιλέχθηκε άλλη εξάρτηση από την πλατφόρμα.

Το επόμενο dependency, παρέχεται ευθύς εξ αρχής από την δημιουργία ενός project σε περιβάλλον Spring Boot, και αναφέρεται στην δυνατότητα δημιουργίας και εκτέλεσης κλάσεων που περιέχουν κώδικα προς δοκιμή (test) της κύριας εφαρμογής. Θεωρείται πολύ σημαντική η υλοποίησης δοκιμών στον κώδικα που έχει κατασκευασθεί, για να διαπιστωθούν τυχόν λάθη και βελτιώσεις που μπορούν να γίνουν, πριν το deploy της εφαρμογής στους τελικούς χρήστες της.

Με δεδομένο ότι η εφαρμογή θα συνδεθεί με βάση δεδομένων, επιλέχθηκε το επόμενο και πολύ σημαντικό dependency που αναφέρεται στο Spring Data, που αποτελεί τον βασικό κορμό στην πρόσβαση και κατ' επέκταση στη διαχείριση των δεδομένων που θα προκύψουν. Ειδικά για την τελευταία διατύπωση γίνεται εξειδίκευση με την χρήση του module JPA, η βασική λειτουργία του οποίου αναλύθηκε στην ενότητα 3.3.2.6.

Συνέχεια στο προηγούμενο dependency, είναι και η τοποθέτηση αυτού που αναφέρεται στην ίδια την βάση δεδομένων. Προστίθεται, λοιπόν, συγκεκριμένος κώδικας XML που θα αφορά την βάση δεδομένων της PostgreSQL.

Τέλος, το τελευταίο dependency υπό την ονομασία Jackson Databind, αναφέρεται στην δυνατότητα της χαρτογράφησης των δεδομένων μέσα από έγγραφα τύπου JSON καθώς και στην μετατροπή τους σε αντικείμενα Java ακόμα και αν το έγγραφο εμπεριέχει εμφωλευμένες εκδοχές (nested JSON).

| |
|---|
| <dependency><br><groupId>org.springframework.boot</groupId><br><artifactId>spring-boot-starter-web</artifactId><br></dependency> |
| <dependency><br><groupId>org.springframework.boot</groupId><br><artifactId>spring-boot-starter-test</artifactId><br><br><scope>test</scope><br></dependency> |
| <dependency><br><groupId>org.springframework.boot</groupId><br><artifactId>spring-boot-starter-data-jpa</artifactId><br></dependency> |
| <dependency><br><groupId>org.postgresql</groupId><br><artifactId>postgresql</artifactId><br></dependency> |
| <dependency><br><groupId>com.fasterxml.jackson.core</groupId><br><artifactId>jackson-databind</artifactId><br></dependency> |

**Πίνακας 11. Τα αρχικώς χρησιμοποιούμενα dependencies της εφαρμογής**

#### 3.2.2.11 Annotations

Ένα από τα κύρια χαρακτηριστικά του Spring Framework είναι η χρήση «επισημάνσεων κώδικα» ή καλύτερα με τον αγγλικό όρο «annotations». Τα τελευταία αποτελούν σημείο αναφοράς για την δόμηση του κώδικα μιας εφαρμογής Spring Boot, καθότι αποτελούν τα στίγματα με τα οποία το framework αναγνωρίζει συγκεκριμένες λειτουργίες που πρέπει να υλοποιηθούν. Κλάσεις, οι οποίες χαρακτηρίζονται με κάποιο ή κάποια annotations λαμβάνουν «εσωτερικές σημάνσεις» και με τον τρόπο αυτόν, αναλαμβάνουν ειδικούς ρόλους εντός του εκάστοτε project, τους οποίους το framework αναγνωρίζει με την χρήση των.

Έτσι, όταν μια κλάση χαρακτηριστεί ως @Service, αυτό σημαίνει ότι έχει αναλάβει τον ρόλο μιας service class, που αναλύθηκε στην ενότητα 3.2.2.7. Όπως ήδη διατυπώθηκε, η χρήση των annotations γίνεται με χρήση του συμβόλου «@» και εν συνεχεία το όνομα αυτού. Η διατύπωση

αυτή πρέπει να μπαίνει πάνω στο όνομα μιας κλάσης ή και των μεθόδων αυτής. Ο παρακάτω πίνακας, δίνει ένα τυπικό παράδειγμα της ανωτέρω θεώρησης.

```
@Service
public class Service
{
   public List<String> getData() {
      return Collections.singletonList(someRepo.findall());
   }
}
```

**Πίνακας 12. Παράδειγμα χρήσης annotation σε κλάση**

Στις επόμενες παραγράφους, δίνονται συνοπτικά πληροφορίες για τα annotations που χρησιμοποιήθηκαν για την υλοποίηση της παρούσας διατριβής.

**@Entity**

Με την χρήση του annotation αυτού, μια κλάση χαρακτηρίζεται, ως οντότητα και απαραίτητο στοιχείο προκειμένου να χτιστεί μέσω της κλάσης αυτής, η βάση δεδομένων του τελικού προγράμματος. Το annotation αυτό, χρησιμοποιήθηκε στις POJO κλάσεις, έτσι ώστε να σημανθούν αυτές ως οντότητες.

**@Table**

Θα μπορούσε να ειπωθεί ότι το annotation αυτό αποτελεί προέκταση του προηγουμένου. Με την σήμανση αυτή, η κλάση αυτή στοχοποιείται ως πίνακας για την βάση δεδομένων που θα δημιουργηθεί. Εφόσον η κλάση στοχοποιηθεί ως οντότητα, το annotation αυτό αποτελεί τον αντίστοιχο πίνακα αυτής.

**@JsonIgnoreProperties(ignoreUnknown = true)**

Κατά την διαδικασία συλλογής των δεδομένων από το έγγραφο JSON, στοιχεία τα οποία διαφέρουν των μεταβλητών που έχουν δημιουργηθεί στη POJO κλάση, θα αγνοούνται. Με το annotation δηλαδή αυτό, δηλώνεται στο framework ότι τα δεδομένα που θα περασθούν στη βάση θα είναι μόνο αυτών των μεταβλητών που έχουν ορισθεί στη POJO Class. Το annotation αυτό λαμβάνει παράμετρο σε παρένθεση, ήτοι το ignoreUnknown ίσο με true, για την ολοκλήρωση της διαδικασίας.

**@Id**

Με την δημιουργία της σχεσιακής βάσης δεδομένων, θα πρέπει να γίνει ο ανάλογος χαρακτηρισμός μιας μεταβλητής ως πρωτεύων κλειδί. Το annotation αυτό επιτελεί αυτή τη διαδικασία.

**@Bean**

Σε προηγούμενες παραγράφους, πραγματοποιήθηκε αναφορά στον ρόλο των beans και πως αυτά στοιχειοθετούν μια πλήρη εφαρμογή σε περιβάλλον Spring Boot. Η επισήμανση τους γίνεται με το ανάλογο annotation, κατά κύριο λόγο, εντός Configuration κλάσεων.

**@Autowired**

Η συγκεκριμένη επισήμανση θεωρείται από τις πιο σημαντικές στο framework του Spring καθότι είναι αυτή που υλοποιεί την τεχνική dependency injection. Με το annotation γίνονται inject τα αντικείμενα που επιθυμεί ο χρήστης από κλάση σε κλάση, σύμφωνα με τα λεγόμενα της ενότητας 1.2.

**@Configuration**

Η ενότητα 3.3.2.5, εισήγαγε την έννοια και την σημασία των Configuration κλάσεων σε μια εφαρμογή Spring Boot και των χειρισμών μεθόδων οι οποίες αποτελούν τα beans της εφαρμογής. Η επισήμανση τους γίνεται με το συγκεκριμένο annotation.

**@RestController**

Το annotation αυτό, αποτελεί μια εξειδίκευση του @Controller και χρησιμοποιείται σε υπηρεσίες RESTful για τον χειρισμό των CRUD ερωτημάτων. Εν αντιθέσει με το τελευταίο, το annotation αυτό δεν μπορεί να επιστρέψει στους χρήστες κάποια προβολή, ενώ με τη χρήση του νοείται η ταυτόχρονη χρήση του annotation @ResponseBody κάτι το οποίο δεν είναι εφικτό να γίνει με το @Controller. Το @ResponseBody ενημερώνει τον controller ότι το επιστρεφόμενο αντικείμενο θα ψηφιοποιηθεί σε έγγραφο τύπου JSON και θα σταλεί πίσω ως αντικείμενο HTTP απάντησης.

**@Controller**

Στην ενότητα 3.3.2.3, έγινε λόγος για την ύπαρξη των Controller κλάσεων και την σπουδαιότητα τους στον χειρισμό ερωτημάτων των χρηστών. Η επισήμανση τους γίνεται με το συγκεκριμένο annotation, έτσι ώστε το framework να τους αναθέσει το αντίστοιχο ρόλο. Το annotation αυτό αποτελεί προέκταση του @Component.

**@GetMapping**

Το annotation αυτό, αποτελεί προϊόν απο την έκδοση Spring 4.3 και μετά, ενώ είναι παρακλάδι του παλαιοτέρου annotation @RequestMapping. Σαν λειτουργίες, τα τελευταία χρησιμοποιούνται για την χαρτογράφηση των ερωτημάτων σε μια εφαρμογή Spring Boot. Το @GetMapping λοιπόν, χαρτογραφεί ερωτήματα τύπου GET. Αντίστοιχα annotation υπάρχουν για τους υπόλοιπους τύπους ερωτημάτων ήτοι: @PostMapping, @PutMapping, @DeleteMapping και @PatchMapping.

**@Value**

Πολύ σημαντικό και ιδιαίτερο annotation αποτελεί το , το οποίο είναι υπεύθυνο για την ανάθεση default σε μεταβλητές. Στα πλαίσια της παρούσας εργασίας, χρησιμοποιείται σε μεταβλητή String, η τιμή της οποίας γίνεται inject από το αρχείο application.properties σε κλάση της εφαρμογής μέσου του annotation αυτού.

**@Repository**

Στην ενότητα 3.3.2.6, ειπώθηκε η σημασία των Repository κλάσεων για τον χειρισμό και την επικοινωνία των δεδομένων από και προς τις αντίστοιχες βάσεις. Ο χαρακτηρισμός μιας τέτοιας κλάσης με το συγκεκριμένο annotation, είναι δεδομένος προκειμένου το framework να καταλάβει τον ρόλο που αυτή διαδραματίζει. Το annotation αυτό αποτελεί προέκταση του @Component.

**@Service**

Στην ενότητα 3.3.2.7, έγινε αναφορά στις Service κλάσεις και στον χειρισμό του business logic που αυτές διέπουν. Για να αντιληφθεί το framework, τον ρόλο αυτό, γίνεται χρήση του συγκεκριμένου annotation. Όπως και με τα @Controller και @Repository, το συγκεκριμένο annotation αποτελεί προέκταση του @Component.

**@ControllerAdvice**

Το annotation αυτό χρησιμοποιήθηκε για την διαχείριση των exceptions που προέκυψαν στην εφαρμογή. Το τελευταίο προσδίδει χαρακτηριστικά χειρισμού των εξαιρέσεων στο global επίπεδο της εφαρμογής, αποφεύγοντας έτσι την διπλοεγγραφή κώδικα σε διάφορα σημεία της. Παρέχει τα εξής σημαντικά πλεονεκτήματα:

- Δίνει στους developers πλήρη έλεγχο του σώματος απάντησης όπως και του κωδικού κατάστασης του μηνύματος.
- Προσφέρει καταγραφή διαφόρων exceptions και την ενσωμάτωση τους, σε μία μέθοδο για τον ολικό τους χειρισμό.
- Δίνει την δυνατότητα χρήσης αντικειμένων της κλάσης «Response Entity» για λεπτομερέστερη καταγραφή των μηνυμάτων.

**@ExceptionHandler**

Λειτουργεί συμπληρωματικά με τον προηγούμενο annotation και τοποθετείται στο επίπεδο μιας μεθόδου, όχι της κλάσης συνολικά. Μια μέθοδος λοιπόν, με το annotation αυτό, κάνει χειρισμό του exception που θα δηλωθεί εντός των brackets αυτού, αλλά όχι σε global επίπεδο

παρά μόνο για την μέθοδο στην οποία ορίσθηκε. Μαζί με τη χρήση του annotation @ControllerAdvice επιτυγχάνεται το global scope.

**@Mock**

Η χρήση του συγκεκριμένου annotation ήταν καθοριστικής σημασίας για την ανάπτυξη tests, παρουσία ενός project που υλοποιείται με την μέθοδο dependency injection. Το τελευταίο λοιπόν, ανήκει στο framework Mockito, και δημιουργεί instances των αντικειμένων στα οποία ορίζεται, χωρίς να χρειάζεται αυτά να κληθούν προκειμένου να διεξαχθούν τα tests και η ορθή λειτουργία του productive κώδικα.

**@InjectMocks**

Η λειτουργία του annotation αυτού είναι με συμπληρωματική με του @Mock και δίνει την δυνατότητα υλοποίησης injections των πεδίων κάποιων κλάσεων, στη κλάση την οποία εξετάζεται. Στο πλαίσιο του παρόντος έργου, έγινε χρήση αυτού για τα δύο repositories εντός της κλάσης Service.

**@DisplayName**

Το annotation αυτό αντικαθιστά το όνομα της κλάσης ενός test, με όνομα υπό τη μορφή string, που θα δοθεί από τον developer.

# 4 DEVOPS OPERATIONS - MONITORING

Το κεφάλαιο αυτό, αναλώνεται σ ένα επίσης πολύ σημαντικό κομμάτι της διπλωματικής εργασίας, το οποίο αφορά στην παρακολούθηση των metrics που εξάγονται από την εφαρμογή. Επιπροσθέτως, η παρακολούθηση αυτή προχωράει ένα βήμα παραπέρα με την χρήση εργαλείων DevOps με σκοπό την χαρτογράφηση των metrics και την γραφική απεικόνιση τους.

Συγκεκριμένα, στο κεφάλαιο αυτό, αναλύεται η χρήση του Spring Actuator ως βασικού εργαλείου για την εξαγωγή των endpoints της εφαρμογής, της open-source πλατφόρμας Prometheus, υπεύθυνης για την συλλογή και το monitoring των endpoints αυτών και τέλος της διαδικτυακής εφαρμογής Grafana, η οποία θα βοηθήσει στην γραφική οπτικοποίηση των αποτελεσμάτων. Το στήσιμο του τμήματος αυτού, έγινε με την χρήση containers μέσω της πλατφόρμας λογισμικού Docker σε περιβάλλον WSL Ubuntu εκτελεσμένο Windows 11 OS.

## 4.1 Spring Actuator

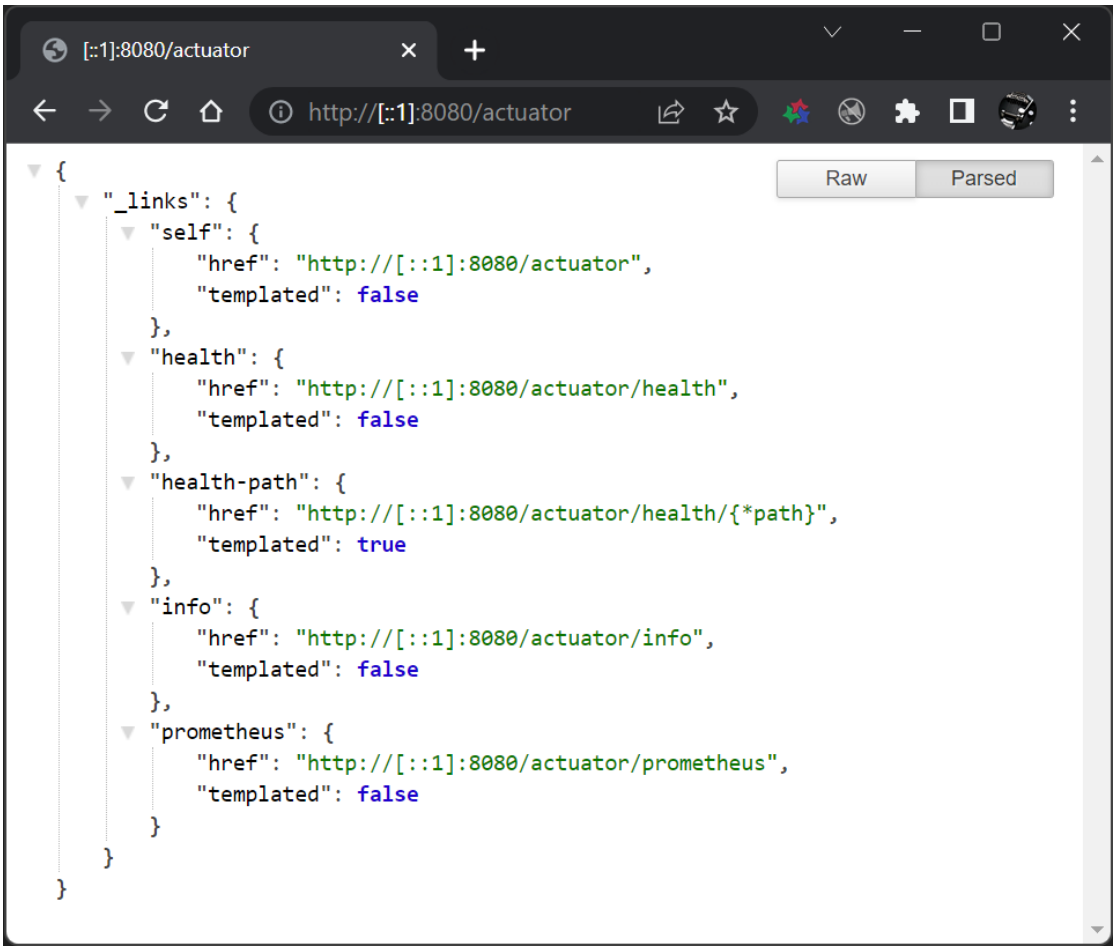
Η λειτουργία Actuator έχει συσταθεί από τον Απρίλιο του 2014, από την πρώτη δηλαδή έκδοση του Spring Boot και έκτοτε έχει λάβει πολλές και ενδιαφέρουσες παραμετροποιήσεις. Το βασικό της έργο, αφορά στην πρόσδοση πολύτιμων χαρακτηριστικών metrics, τα οποία αφορούν στην κατανάλωση συστημικών πόρων, στην αποτύπωση της συνολικής «υγείας » της εφαρμογής ενώ παρέχει και την δυνατότητα κατασκευής custom metrics από τον εκάστοτε χρήστη.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

**Πίνακας 13. Dependency Εγκατάστασης Spring Actuator**

Για την εγκατάσταση της αρκεί η τοποθέτηση του ανωτέρω dependency στο αρχείο POM.XML. Από το σημείο εκείνο, η δημιουργηθείσα εφαρμογή έχει αποκτήσει ένα περαιτέρω path ήτοι την διεύθυνση /actuator, η πληκτρολόγηση της οποίας θα δώσει τα αποτελέσματα της παρακάτω εικόνας. Όπως γίνεται αντιληπτό, στην εφαρμογή αυτή έχουν γίνει expose, δηλαδή έκθεση, 5 endpoints, τα οποία αφορούν στην παροχή πληροφοριών της εφαρμογής (info), στην

κατάσταση υγείας της εφαρμογής (health) καθώς και στην σύνδεση επιπρόσθετων metrics μέσω της Prometheus. Το τελευταίο κομμάτι, αναλύεται σε ξεχωριστή ενότητα.



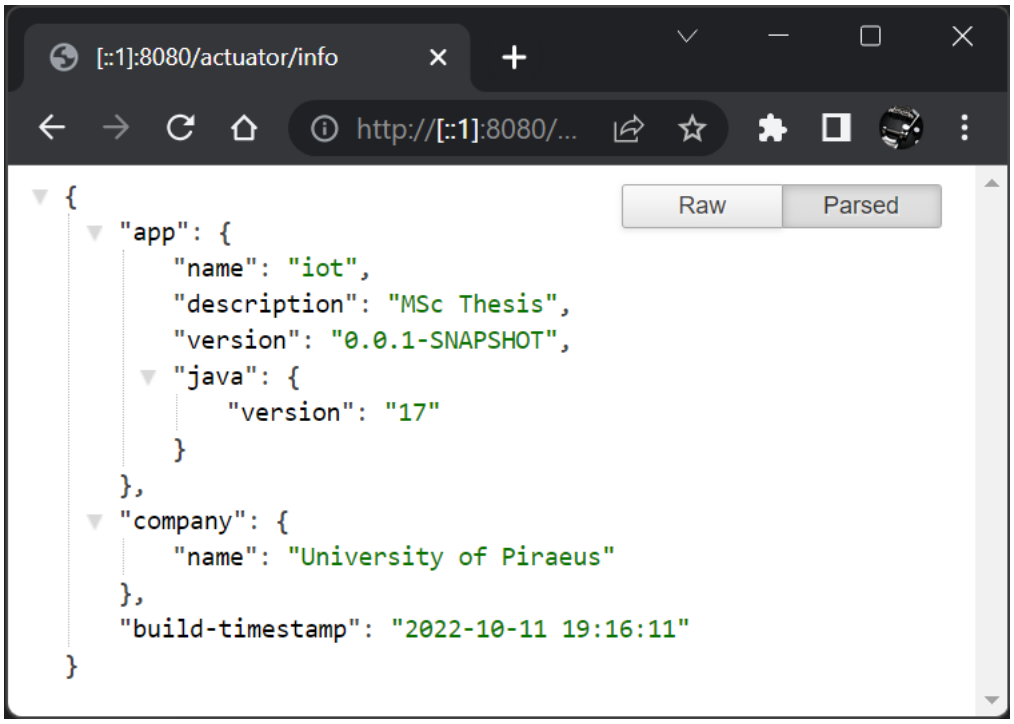

**Εικόνα 29. Endpoints Εφαρμογής Spring Boot**

Ως παράδειγμα χρήσης ενός endpoint, μπορεί κανείς να χρησιμοποιήσει αυτό των πληροφορίων, ήτοι το info. Το τελευταίο παρέχει στους χρήστες αυτό που υπονοεί και το όνομα του, δηλαδή την παροχή πληροφοριών. Εν αρχή, το συγκεκριμένο endpoint όταν πατηθεί θα δώσει μια κενή λίστα σε αρχείο τύπου JSON. Προκείμενου να ληφθούν πληροφορίες, απαιτείται η πρόσδοση αυτών από τον developer στο αρχείο application.properties της εφαρμογής.

```
#Information exposed to actuator/info endpoint
info.app.name=@project.name@
info.app.description=@project.description@
info.app.version=@project.version@
info.app.java.version=@java.version@
info.company.name=University of Piraeus
info.build-timestamp=@maven.build.timestamp@
```

**Εικόνα 30. Παροχή Πληροφοριών για το Info Endpoint**

Η συγκεκριμένη παραμετροποίηση θα έχει ως αποτέλεσμα την παροχή των πληροφοριών που απεικονίζονται παρακάτω.




**Εικόνα 31. Απεικόνιση του info endpoint**

## 4.2 Prometheus

Όπως ειπώθηκε και στην εισαγωγή του παρόντος κεφαλαίου, η Prometheus αποτελεί μια πλατφόρμα ανοικτού λογισμικού, υπεύθυνης για την διαδικασία monitoring καθώς και alerting των χρηστών. Η σύσταση της οριοθετείται το 2012 και πλέον εφαρμόζεται σε ποικίλες εφαρμογές παρατήρησης και καλής λειτουργίας των εφαρμογών, πολλών εταιριών ανά τον κόσμο.

Παρέχει δυνατότητες ενημέρωσης στους χρήστες, σε περιπτώσεις που οι τελευταίοι, οφείλουν να αναλάβουν δράση όταν κάποια διεργασία αστοχήσει. Το τελευταίο γίνεται με στοιχεία που συλλέγονται από την Prometheus αποθηκεύονται ως δεδομένα σε χρονική σειριακή μορφή με την χρήση timestamp. Στην παρούσα εργασία θα γίνει χρήση της Prometheus, η οποία θα λαμβάνει στην είσοδο της custom metrics, τα οποία κατασκευάσθηκαν από τον συγγραφέα φοιτητή. Πριν γίνει όμως αυτό, θα γίνει μια ανάλυση στον τρόπο λειτουργίας της Prometheus και πως έγινε η σύνδεση της με την παρούσα εφαρμογή.

### 4.2.1 Αλλαγή Θύρας Metrics Exposing

Μια πολύ σημαντική παραμετροποίηση, η οποία έλαβε χώρα στην εφαρμογή, αφορούσε την αλλαγή του port στο οποίο θα γίνεται το expose των endpoints και κατ' επέκταση των metrics. Η συγκεκριμένη σύμβαση, θεωρήθηκε απαραίτητη διότι ως γνωστόν, έχει εφαρμοστεί το πρωτόκολλο HTTPS με την βοήθεια Self Signed Certificate υπό την υλοποίηση του Spring Security. Υπό την σκοπιά αυτή, η θύρα από την οποία γίνεται χρήση για την εκτέλεση της εφαρμογή είναι η 8443 σε πρωτόκολλο HTTPS.

Ωστόσο κατά την χρήση της Prometheus, η οποία αναλύεται στις επόμενες παραγράφους, παρουσιάστηκε πρόβλημα στην επικοινωνία με την εφαρμογή εντός του Spring Framework καθότι απαιτούσε πιστοποίηση κατά TLS. Έγιναν προσπάθειες επικοινωνίας υπό τις συνθήκες αυτές ωστόσο, δεν κατέστη εφικτή η ολοκλήρωση της διαδικασίας, λόγω εμπλοκών στο web configuration της Prometheus σε περιβάλλον Docker Container.

Στο σημείο αυτό, έγινε αντιληπτό ότι το Framework παρέχει την δυνατότητα στους developers, να προβούν σε αλλαγή του port μεμονωμένα και μόνο για τα endpoints. Αυτό σημαίνει ότι η εφαρμογή εκκινεί κανονικά από την θύρα 8443 με όλα τα πρωτόκολλα ασφαλείας ενεργοποιημένα αλλά ανοίγει και μια επιπρόσθετη θύρα, που καθορίστηκε στον αριθμό 8080, για να μπορέσει να γίνει το fetch των endpoints από την Prometheus σε πρωτόκολλο HTTP, με ταυτόχρονη απενεργοποίηση της ασφάλειας SSL.

Γίνεται αμέσως αντιληπτό, ότι το εν λόγω exposing δημιουργεί ευπάθειες (vulnerabilities) στην εφαρμογή, οι οποίες την θέτουν σε κίνδυνο. Ωστόσο, ο συγγραφέας φοιτητής προχώρησε με την σύμβαση αυτή για λόγους καθαρά demonstration του όλου εγχειρήματος. Η παραμετροποίηση αυτή έγινε με εντολές στον τομέα management.server εντός του αρχείου application.properties του project με την επόμενη εικόνα τις παρουσιάζει.

```
#Setting to expose endpoints on HTTP port
management.server.ssl.enabled=false
management.server.port=8080
management.endpoints.web.exposure.include=health,info,prometheus
management.endpoint.info.enabled=true
management.info.env.enabled=true
```

**Εικόνα 32. Δημιουργία Θύρας HTTP**

### 4.2.2 Εγκατάσταση Prometheus Dependency

Με τις αλλαγές της ενότητας 4.2.1, ακολουθεί η εισαγωγή του dependency για το Spring Actuator, κάτι που επεξηγήθηκε στην ενότητα 4.1. Εν συνεχεία, το επόμενο βήμα, απαιτεί την εγκατάσταση για το micrometer της πλατφόρμας Prometheus. Με τον όρο Micrometer νοείται το εργαλείο συλλογής των metrics, κάτι το οποίο υπάρχει μεν και στο Spring Actuator ωστόσο απαιτείται η περαιτέρω εξειδίκευση του με τρόπο, τον οποίο διαβάζεται από την πλατφόρμα Prometheus.

```
<dependency>
  <groupId>io.micrometer</groupId>
  <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

**Πίνακας 14. Dependency Εγκατάστασης Prometheus**

Η εγκατάσταση του ανωτέρω dependency δίνει πρόσβαση στο endpoint με διεύθυνση http://localhost:8080/actuator/prometheus. Στην τελευταία όπως μπορεί να γίνει αντιληπτό εμφανίζονται τα metrics που συλλέγει η πλατφόρμα και θα απεικονισθούν σε επόμενη ενότητα.

### 4.2.3 Εισαγωγή Custom Metric

Παρότι η Prometheus παρέχει μια πληθώρα metrics, απευθείας out-of-the-box, θα υπάρξουν στιγμές που θα γίνει η απαίτηση για κατασκευή καινούργιων custom υλοποιήσεων. Ευτυχώς, παρέχεται η δυνατότητα κατασκευής νέων metrics ενώ ταυτόχρονα, το είδος αυτών επεκτείνεται σε 3 υποκατηγορίες, ήτοι gauge, counter timer metrics.

| Είδος Metric | Χρήση |
|---|---|
| Gauge | Μέτρηση χρήσης πόρων. Τιμές που μπορούν να λάβουν αύξουσες και φθίνουσες τιμές με ανώτατα όρια |
| Counter | Καταμέτρηση events και δράσεων. Τιμές που λαμβάνουν αύξουσα πορεία |
| Timer | Καταμέτρηση μικρών χρονικών διαστημάτων και της συχνότητας τους |

**Πίνακας 15. Είδη Custom Metrics**

Ο ανωτέρω πίνακας τις αποτυπώνει, δίνοντας μια πιο συνοπτική εικόνα στο είδος λειτουργίας που αυτές παρέχουν. Για λόγους επίδειξης, θα γίνει χρήση δύο custom metrics για να απεικονισθούν οι δυνατότητες της Prometheus σε συνδυασμό με το Spring Framework. Σε επόμενη ενότητα μπαίνει στην υλοποίηση και η πλατφόρμα Grafana για να ολοκληρωθεί η οπτικοποίηση των εξαγόμενων δεδομένων.

#### 4.2.3.1 Timer Custom Metric

Στην υλοποίηση αυτή, επιλέχθηκε η κατασκευή ενός metric το οποίο θα είναι τύπου timer, ήτοι θα καταμετρά το χρονικό διάστημα το οποίο απαιτείται για να εκτελεστεί μια μέθοδος της εφαρμογής. Προπομπός αυτής, είναι η χρήση του κάτωθι dependency στο αρχείο POM.XML, το οποίο επιτρέπει την χρήση custom timer σε μεθόδους.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-aop</artifactId>
</dependency>
```

**Πίνακας 16. Dependency AOP**

Η μέθοδος που επιλέχθηκε είναι αυτή του κάτωθι πίνακα και που στην ουσία είναι υπεύθυνη για την αποτύπωση των εγγραφών των ταινιών της βάσης δεδομένων προς το front end τμήμα της εφαρμογής σε Thymeleaf template.

```
@Timed(value = “show-all-movies.time”, description = “Time taken to return Movies”)
@GetMapping(“/show-movies-contents”)
public String showAllMovies(HttpServletRequest request)
{
   request.setAttribute("movies", mainService.showAllMovies());
      return “movies-db-contents”;
}
```

**Πίνακας 17. Υλοποίηση Custom Metric σε Μέθοδο**

Όπως γίνεται αντιληπτό, έχει γίνει χρήση ενός annotation @Timed, το οποίο για να μπορέσει να υλοποιηθεί απαιτεί την παρουσία ενός @Bean Timed Aspect, το οποίο τοποθετήθηκε στην Main class της εφαρμογής. Counter Custom Metric.

```
@Bean
public TimedAspect timedAspect(MeterRegistry registry) {
return new TimedAspect(registry);
}
```

**Πίνακας 18. Timed Aspect bean**

#### 4.2.3.2 Counter Custom Metric

Στην δεύτερη υλοποίηση, επιλέχθηκε η κατασκευή ενός metric το οποίο θα είναι τύπου counter, ήτοι θα καταμετρά τα χτυπήματα τα οποία δέχεται μια μέθοδος της εφαρμογής κατά το runtime. Στην περίπτωση αυτή, δεν απαιτείται η εισαγωγή κάποιου dependency, παρά μόνο η κατασκευή ενός αντικειμένου τύπου Counter με ανάλογη προσαρμογή στον constructor της κλάσης.

```
@Controller
public class MainController {

    private final MainService mainService;
    private final Counter hitCounter;

    @Autowired
    public MainController(MainService mainService, MeterRegistry meterRegistry) {
        this.mainService = mainService;
        this.hitCounter = Counter.builder("hit_counter")
                .description("Number of Hits")
                .register(meterRegistry);
    }
```

**Εικόνα 33. Εισαγωγή Counter σε Controller Κλάση**

Ενώ στη συνέχεια, γίνεται instantiation του αντικειμένου Counter στην μέθοδο που επιθυμεί ο developer να μετρήσει τα δεδομένα, με χρήση της εσωτερικής μεθόδου increment( ).

```
@GetMapping("/show-news-contents")
public String showAllRates(HttpServletRequest request){
    request.setAttribute("rates", mainService.showAllRates());
    hitCounter.increment();
    return "news-db-contents";
}
```

**Εικόνα 34. Υλοποίηση Counter σε Μέθοδο**

Με την εισαγωγή όλων των ανωτέρω παραμέτρων, μπορεί η εφαρμογή να εκκινήσει ενώ με μετάβαση στη διεύθυνση http://localhost:8080/actuator/prometheus, αποτυπώνονται όλα τα metrics της εφαρμογής μαζί με τα custom που μόλις δημιουργήθηκαν.

```
# HELP show_all_movies_time_seconds_max Time taken to return Movies
# TYPE show_all_movies_time_seconds_max gauge
show_all_movies_time_seconds_max{class="com.andrekreou.iot.control.controller.MainController",exception="none",method="showAllM
ovies",} 0.0178342
# HELP show_all_movies_time_seconds Time taken to return Movies
# TYPE show_all_movies_time_seconds summary
show_all_movies_time_seconds_count{class="com.andrekreou.iot.control.controller.MainController",exception="none",method="showAl
lMovies",} 1.0
show_all_movies_time_seconds_sum{class="com.andrekreou.iot.control.controller.MainController",exception="none",method="showAllM
ovies",} 0.0178342
```

**Εικόνα 35. Time Custom Metric**

Στην παραπάνω εικόνα, φαίνονται τα στοιχεία του metric που αφορά στον χρόνο εκτέλεσης της μεθόδου showAllMovies, ενώ στην αμέσως επόμενη ο αριθμός χτυπημάτων για την μέθοδο showAllRates.

```
# HELP hit_counter_total Number of Hits
# TYPE hit_counter_total counter
hit_counter_total 1.0
```

**Εικόνα 36. Counter Custom Metric**

Στο σημείο αυτό αξίζει να αναφερθεί ότι αν δεν εκτελεστεί η μέθοδος κατά το runtime της εφαρμογής, τότε το ανάλογο metric δεν θα εμφανιστεί.

### 4.2.4 Containerized Prometheus via Docker

Η επόμενη μεγάλη πρόκληση, που κλήθηκε ο συγγραφέας φοιτητής να αντιμετωπίσει ήταν η μεταφορά και απεικόνιση των ανωτέρω metrics στο περιβάλλον της Prometheus ήτοι σε τοπικό της server. Για να γίνει αυτό εφικτό, θα έπρεπε να κατασκευασθεί ένα instance αυτής. Αρωγός στην διαδικασία αυτή, αποτέλεσε η πλατφόρμα ανοικτού λογισμικού Docker, η οποία προσφέρει υπηρεσίες στην ανάπτυξη, αποστολή και εκτέλεση των εφαρμογών που κατασκευάζονται από developers.

Πιο συγκεκριμένα, βασική της λειτουργία αποτελεί η ικανότητα της τοποθέτησης μιας εφαρμογής σε κάποιο πακέτο (package), το οποίο ζει πλέον σ' ένα απομονωμένο περιβάλλον, δίνοντας τις δυνατότητες της προηγούμενης παραγράφου. Βασικό της πλεονέκτημα αποτελεί το γεγονός ότι με τον τρόπο αυτό, δεν απαιτείται η εγκατάσταση μιας τρίτης εφαρμογής τοπικά αλλά η χρήση της, ως container, απομακρυσμένα.

Με δεδομένη λοιπόν, την χρήση της Prometheus ως software container μέσω Docker, σειρά έχει η παραμετροποίηση της επικοινωνίας της με την εφαρμογή Spring Boot. Για να γίνει αυτό εφικτό, θα πρέπει να δοθούν κανόνες scraping προς την containerized Prometheus, οι οποίοι στην ουσία, αποτελούν τις οδηγίες με τις οποίες θα λάβει και θα διαβάσει τα δεδομένα από την εφαρμογή.

Η διαδικασία αυτή, γίνεται με την βοήθεια ενός επιπρόσθετου αρχείου επέκτασης yml, το οποίο κατασκευάσθηκε και τοποθετήθηκε εντός του project. Στον επόμενο πίνακα παρατίθενται τα περιεχόμενα του αρχείου αυτού.

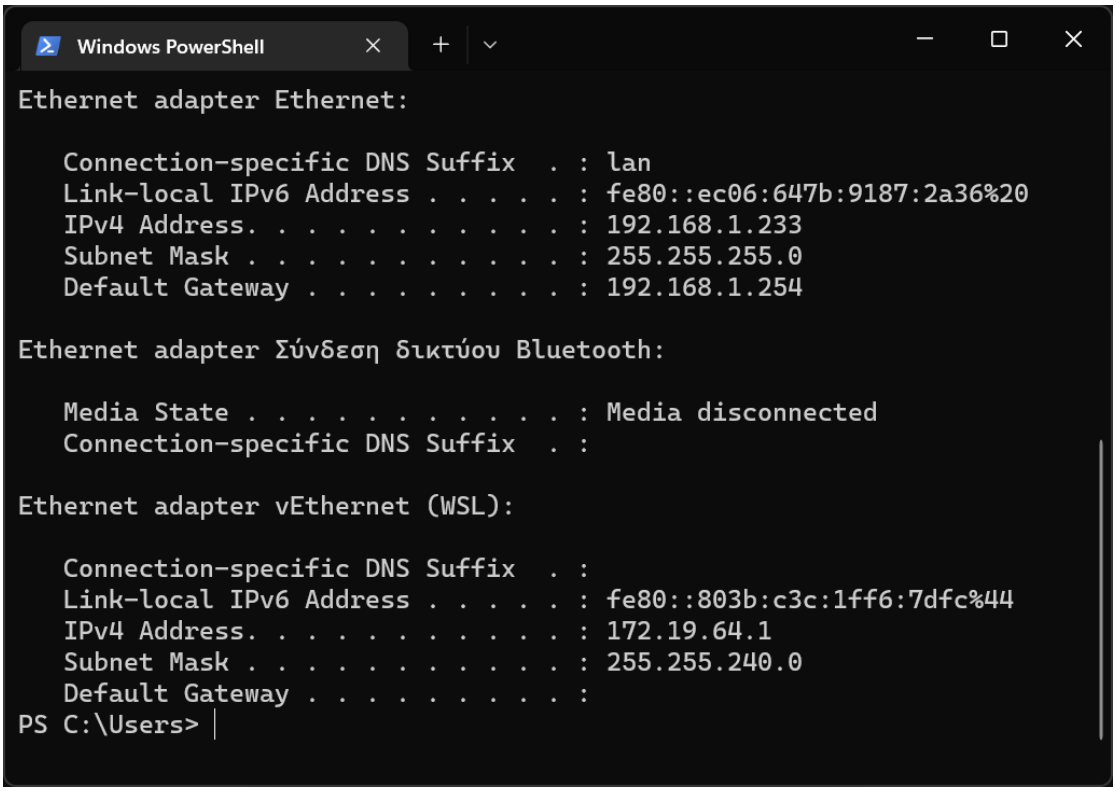
```
global:
  scrape_interval:    15s
  evaluation_interval: 15s
```

```
scrape_configs:

 - job_name: 'prometheus'
   scrape_interval: 5s
   static_configs:
    - targets: ['localhost:9090']

 - job_name: 'Spring Boot Application Input'
   metrics_path: '/actuator/prometheus'
   scrape_interval: 2s
   scheme: http
   static_configs:
    - targets: ['192.168.1.233:8080']
    -   labels:
    -      application: 'Msc Project Thesis'
```

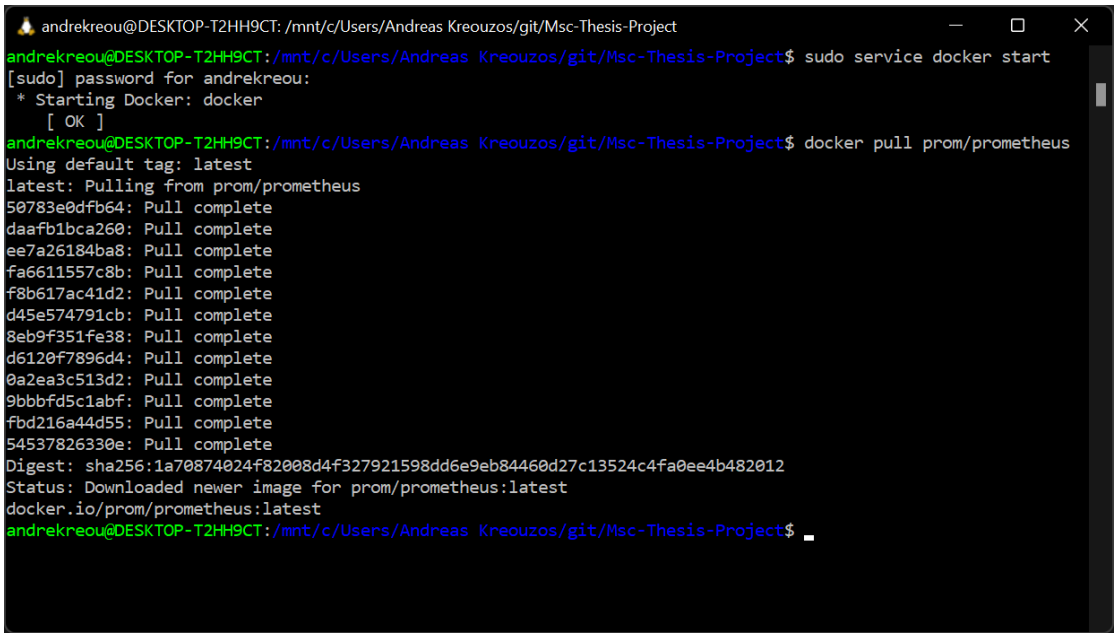
**Πίνακας 19. Περιεχόμενα Αρχείου prometheus.yml**




**Εικόνα 37. Διεύθυνση IP μέσω CMD**

Με το αρχείο ρυθμίσεων prometheus.yml να έχει γραφτεί, στην ουσία μεταβιβάζεται η πληροφορία ότι η Prometheus θα πρέπει να κάνει scrape των metrics κάθε δύο δευτερόλεπτα. Η διαπίστωση αυτή ισχύει για το job το οποίο ονομάσθηκε «Spring Boot Application Input», το οποίο ακούει στην θύρα 8080. Σημαντικό να αναφερθεί το γεγονός ότι αντί για την ονομασία localhost, πρέπει να γίνει χρήση της διεύθυνσης IP του τερματικού όπου συντάσσεται το αρχείο. Η εικόνα 38 που ακολουθεί τον πίνακα, δείχνει την εκτέλεση της εντολής ipconfig σε περιβάλλον cmd. Τέλος το αρχείο yml, εμπεριέχει ακόμη ένα job, που αναφέρεται στα metrics της ίδιας της Prometheus, στην θύρα 9090, όπου είναι και η default.

Με το στήσιμο του αρχείου αυτού, σειρά είχε η εκτέλεση εντολών που θα κάνει την ενοποίηση αυτή. Η διαδικασία πραγματοποιήθηκε μέσω WSL Ubuntu με Windows 11 OS, με την διαδικασία να έχει ως εξής:

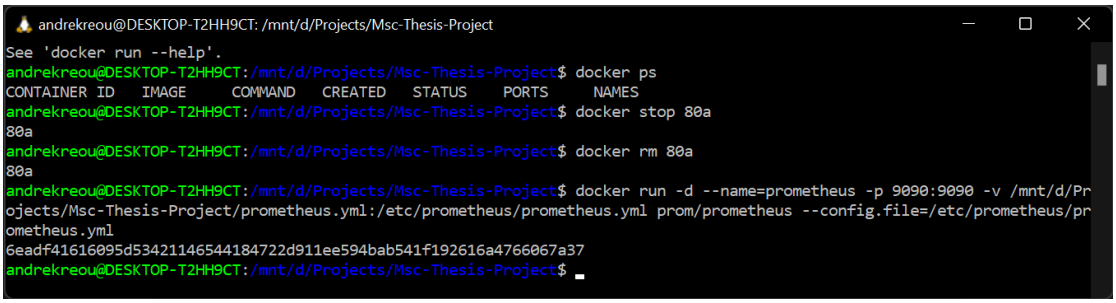



**Εικόνα 38. Εγκατάσταση Prometheus Container μέσω Docker**

1. Εκκίνηση της πλατφόρμας Docker μέσω της εντολής sudo service docker start
2. Εισαγωγή του κωδικού χρήστη
3. Χρήση της εντολής docker pull prom/Prometheus, η οποία μπορεί να βρεθεί στην ιστοσελίδα της πλατφόρμας Docker.
4. Χρήσης της κάτωθι εντολής για την υπόδειξη του αρχείου yml:

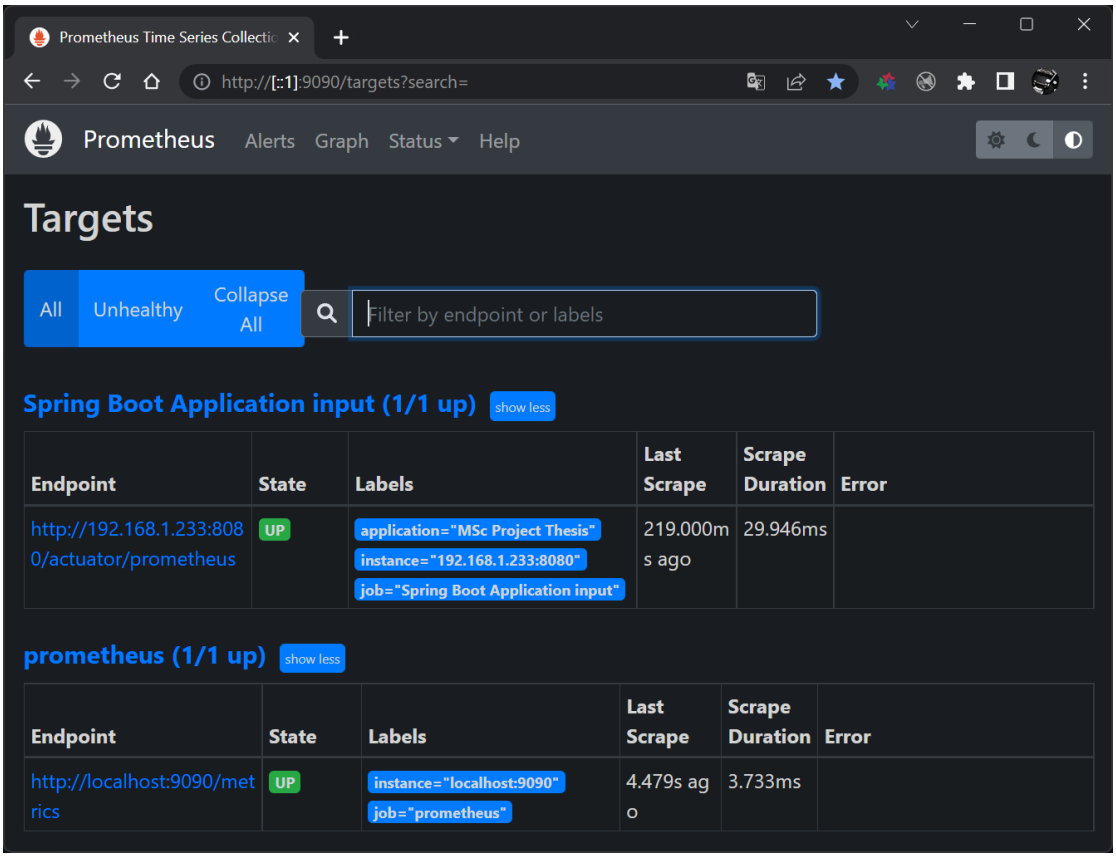
   docker run -d --name=prometheus -p 9090:9090 -v /mnt/d/Projects/Msc-Thesis-Project/prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus --config.file=/etc/prometheus/prometheus.yml

   - Το σημείο αυτό κρίνεται σημαντικό καθότι εδώ γίνεται το mounting του αρχείου που κατασκευάσθηκε από τον developer στην πλατφόρμα της Prometheus. Από το σημείο της εντολής /mnt και μετά, θα πρέπει να τοποθετηθεί το absolute path που βρίσκεται το αρχείο αυτό.




**Εικόνα 39. Κατάσταση Ενεργοποίησης Software Container Prometheus**

Εφόσον όλα έχουν γίνει σωστά, το εξαγόμενο αποτέλεσμα θα έχει την μορφή της εικόνας 40, η οποία και υποδεικνύει ότι ο Prometheus Server είναι ενεργοποιημένος στην θύρα 9090. Πράγματι το αποτέλεσμα της εικόνας 41, δικαιώνει τα προηγούμενα λεγόμενα.

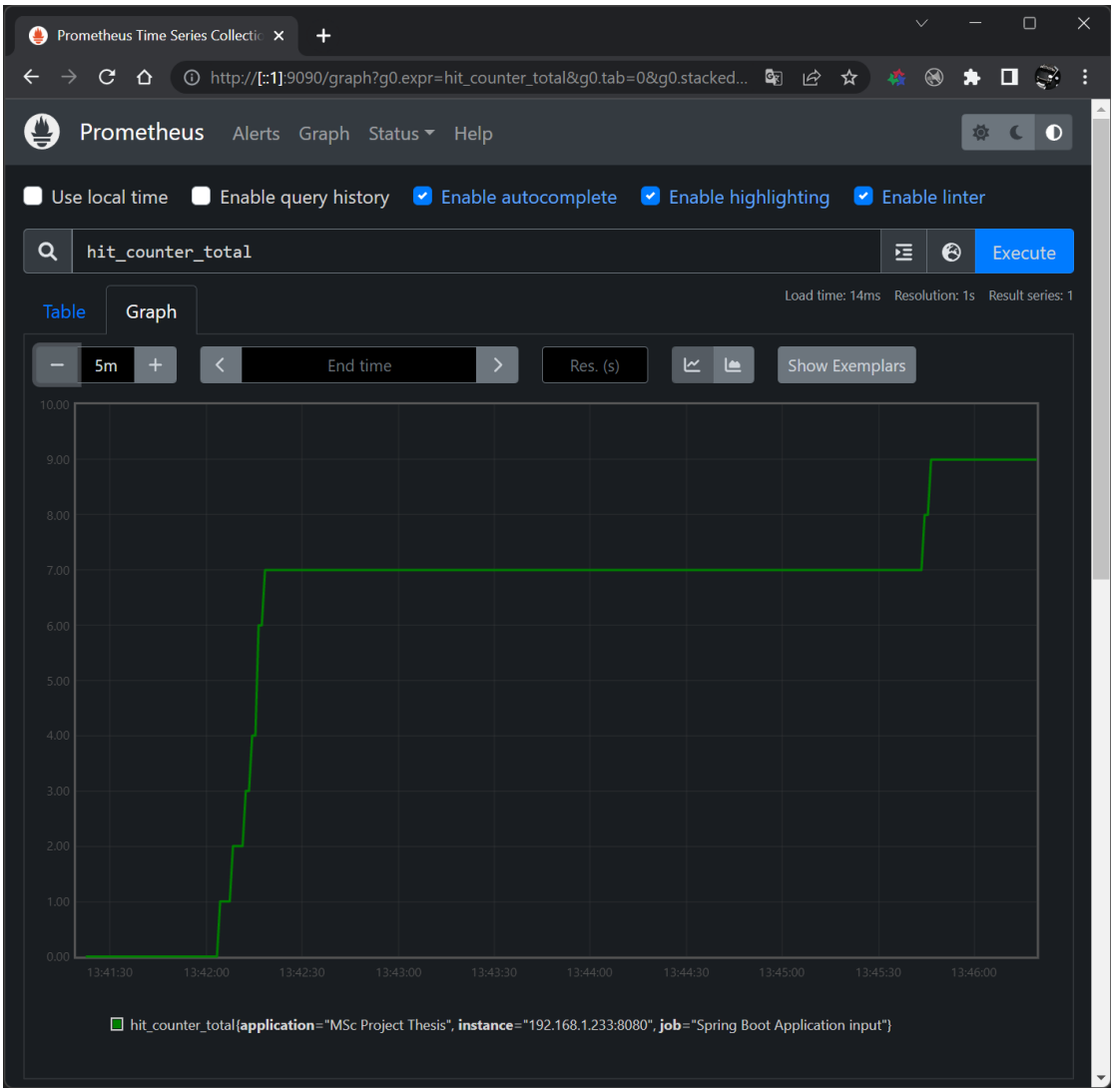



**Εικόνα 40. Prometheus Dashboard**

Στην συγκεκριμένη εικόνα, απεικονίζονται τα endpoints όπως αυτά τέθηκαν στο αρχείο διαχείρισης Prometheus.yml. Η κατάσταση State είναι δηλωμένη στο UP, το οποίο σημαίνει ότι η Prometheus είναι σε θέση να διαβάζει τα endpoints που έχουν γίνει expose στην Spring Boot εφαρμογή.

### 4.2.5 Απεικόνιση Custom Metrics στην Prometheus

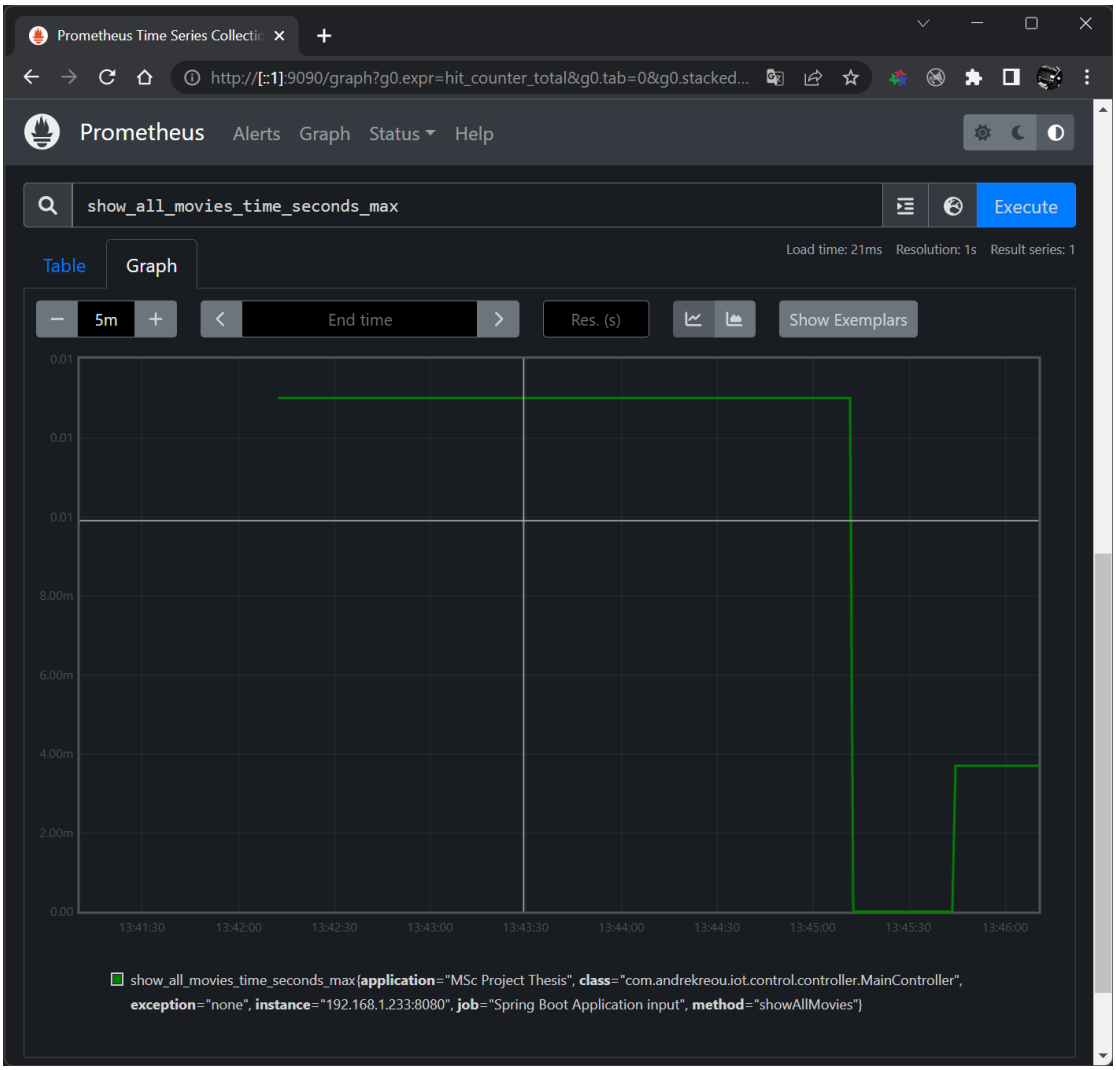
Έχοντας ολοκληρώσει το στάδιο περάσματος των metrics στον Prometheus server, σειρά έχει η απεικόνιση των δυνατοτήτων που αυτός παρέχει. Στο πεδίο «Graph», μπορεί ο εκάστοτε χρήστης να μεταβεί και με την χρήση των metrics να εξάγει τα πρώτα γραφικά αποτελέσματα από τον server.




**Εικόνα 41. Hit Counter Metric on Prometheus**

Η εικόνα 42 απεικονίζει το γράφημα που αφορά στο custom metric counter που δημιουργήθηκε στην ενότητα 4.2.3.2. Όλα τα metrics μπορούν να απεικονισθούν σε λίστα αρκεί να πατηθεί το σύμβολό της «Γης», δίπλα από το «Execute».

Εφόσον το metric επιλεχθεί, με το πάτημα του τελευταίου πλήκτρου, παρέχεται το γράφημα αυτό. Για λόγους καλύτερης απεικόνισης, ο χρήστης μπορεί να αλλάξει το χρονικό εύρος στο οποίο έχουν ληφθεί οι μετρήσεις.




**Εικόνα 42. Timer Metric on Prometheus**

Όμοια περίπτωση, φαίνεται και στην εικόνα 43, αυτή τη φορά για το timer custom metric που υλοποιήθηκε στην ενότητα 4.2.3.1. Τα δεδομένα αυτά, εμφανίζονται όσο η εφαρμογή είναι υπό εκτέλεση. Σε περίπτωση που γίνει κλείσιμο ή επανεκκίνηση της, η λειτουργία του Prometheus Server, δεν παύει να υφίσταται, αλλά στα εξαγόμενα διαγράμματα, θα υπάρχουν χρονικά κενά που ισοδυναμούν με τους χρόνους κατά τους οποίους η εφαρμογή ήταν κλειστή ή επανεκκινούσε.

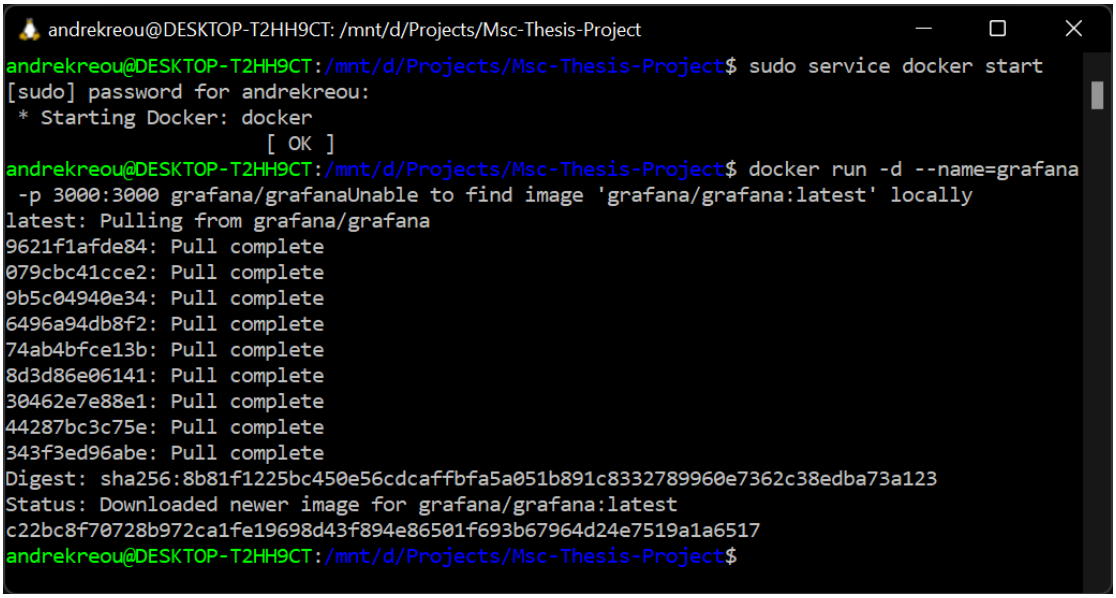
## 4.3 Grafana

Όπως ειπώθηκε και στην αρχή του παρόντος κεφαλαίου, η Grafana αποτελεί πλατφόρμα ανοικτού λογισμικού για την διαδραστική απεικόνιση δεδομένων. Επιτρέπει στους χρήστες της, να βλέπουν τα δεδομένα τους υπό μορφή γραφημάτων τα οποία μπορούν να βρίσκονται ενοποιημένα σε έναν πίνακα, υπό την ονομασία dashboard.

Στα πλαίσια της παρούσας διπλωματικής εργασίας, θα γίνει χρήση της πλατφόρμας αυτής, για να δοθούν οι δυνατότητες της.

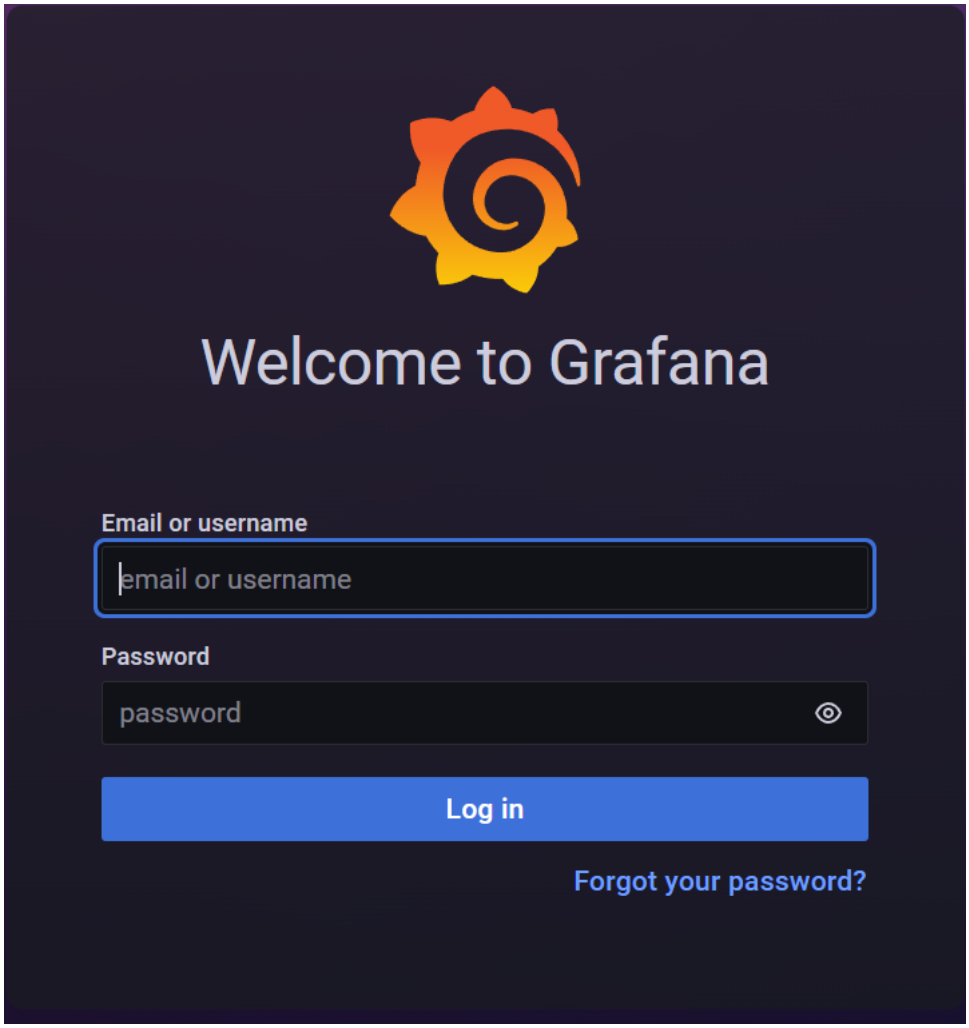
### 4.3.1 Εγκατάσταση

Για την εγκατάσταση της Grafana, η διαδικασία είναι λίγο πιο απλή εν αντιθέσει με την Prometheus, καθότι απαιτείται μονάχα μια εντολή μέσω της πλατφόρμας docker και καμία παραμετροποίηση στην Spring Boot εφαρμογή. Αυτό είναι λογικό καθότι η Grafana θα πρέπει να επικοινωνήσει πλέον με την Prometheus, και ως γνωστόν η επικοινωνία της τελευταίας με την εφαρμογή, πραγματοποιήθηκε σε προηγούμενη ενότητα.




**Εικόνα 43. Εντολή Εγκατάστασης Grafana**

Η εικόνα 44, δείχνει την εντολή που χρησιμοποιήθηκε για την εγκατάσταση της Grafana. Η default θύρα στην οποία μπορεί να απεικονισθεί ο server της Grafana είναι η 3000. Πράγματι με πληκτρολόγηση της διεύθυνσης http://localhost:3000 , λαμβάνεται η σελίδα καλωσορίσματος της Grafana της εικόνας 45.



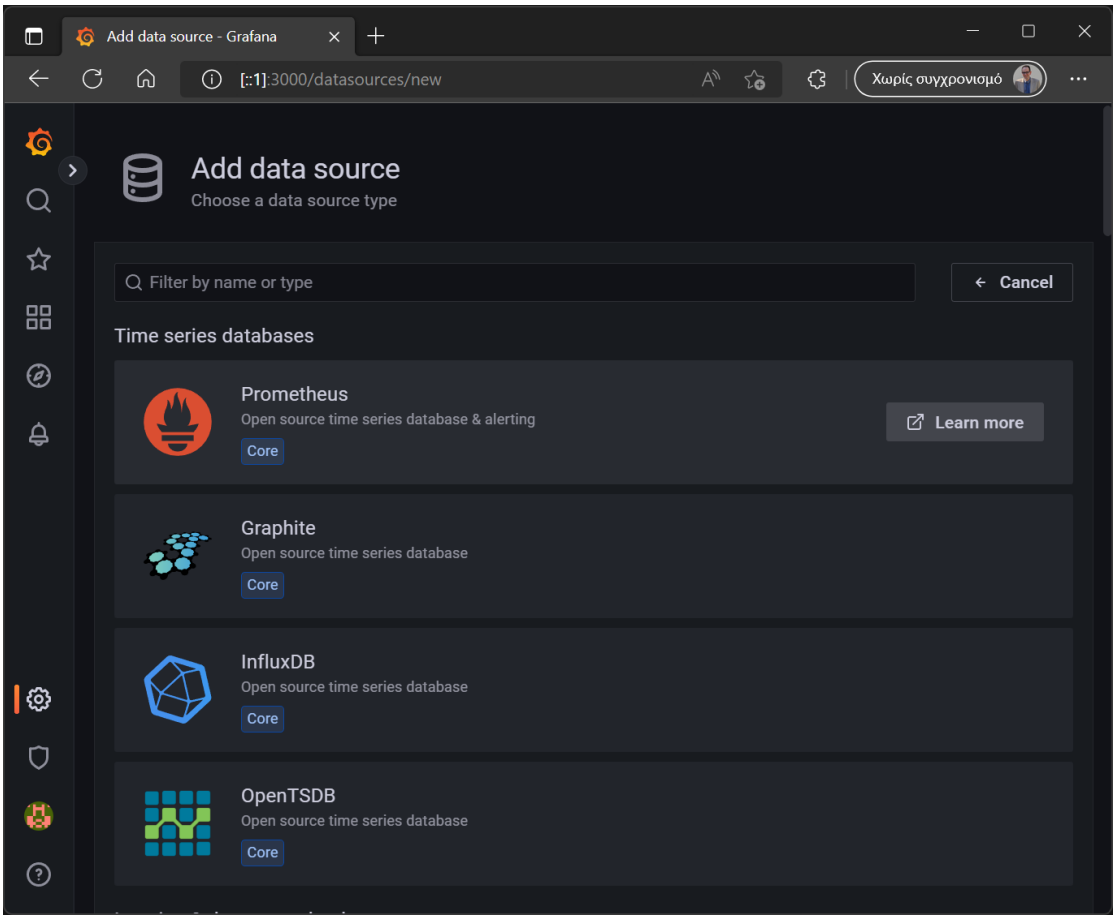

**Εικόνα 44. Grafana Welcome Screen**

### 4.3.2 Εισαγωγή Metrics από Prometheus

Με την ολοκλήρωση της εγκατάστασης και με δεδομένο ότι ο server είναι ενεργός, μπορεί να γίνει χρήση της λέξης «admin» για το πεδίο «username» αλλά και για το «password», προκειμένου να αποφευχθεί η διαδικασία ανοίγματος λογαριασμού, αν δεν είναι απαραίτητος. Η

Grafana θα καλωσορίσει τον χρήστη στο γραφικό της περιβάλλον, παρέχοντας ένα κατακόρυφο μενού επιλογών, τα οποία θα πρέπει να χρησιμοποιήσει για την επίτευξη των στόχων του.

Εν προκειμένω, ο στόχος είναι η εισαγωγή των metrics που αυτή τη στιγμή υπάρχουν στη Prometheus. Για την ολοκλήρωση του, θα πρέπει να πατηθεί το σύμβολο του «γραναζιού», και να επιλεχθεί η επιλογή «Data Sources». Στο αμέσως επόμενο μενού, επιλέγεται το πλήκτρο «Add Data Source» και με τον τρόπο αυτό, θα εμφανιστεί το μενού της εικόνας 46, με όλες τις υποστηριζόμενες από την Grafana πλατφόρμες.

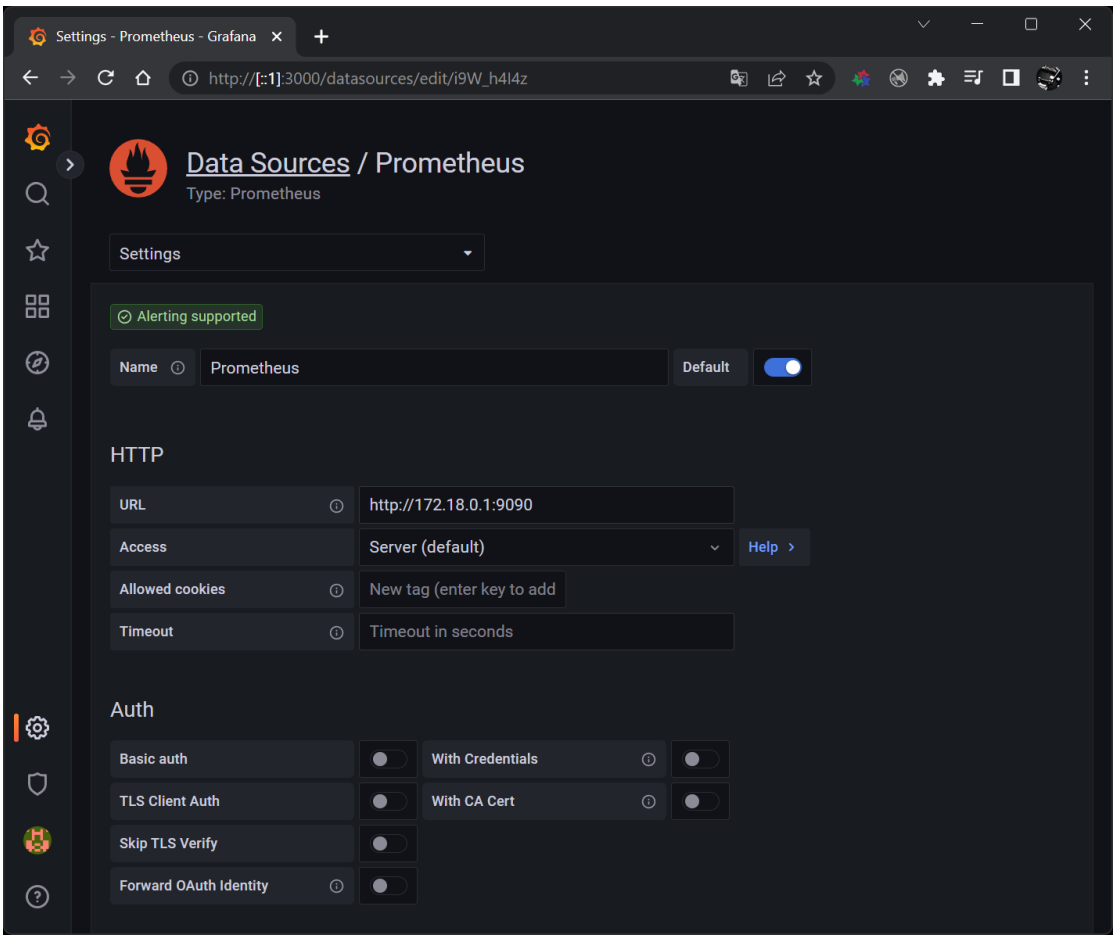



**Εικόνα 45. Εισαγωγή Prometheus ως Data Source**

Με επιλογή της Prometheus, η οποία εμφανίστηκε πρώτη στη λίστα, θα απεικονισθεί η εικόνα 47, στην οποία θα γίνει η ένωση των δύο πλατφορμών. Η μόνη ρύθμιση που χρειάζεται στο σημείο αυτό, είναι η http διεύθυνση στην οποία βρίσκεται ο server της Prometheus. Ταυτόχρονα όμως,

το σημείο αυτό ήταν επίσης μια πρόκληση για τον συγγραφέα φοιτητή καθότι η απλή πληκτρολόγηση της διεύθυνσης http://localhost:9090, έφερε αποτελέσματα αδυναμίας σύνδεσης.

Έπειτα από ενδελεχή έρευνα, καθώς και χρήση διαφορετικών εντολών, οι οποίες αντικαθιστούσαν την λέξη «localhost», στην ανωτέρω διεύθυνση, προέκυψε ότι το πρόβλημα οφειλόταν στο γεγονός μη επιτυχούς εισαγωγής IP διεύθυνσης του container της Prometheus. Για να γίνει αυτό αντιληπτό, έπρεπε να γίνει κάποια περαιτέρω κατανόηση του τρόπου με τον οποίο λειτουργεί η πλατφόρμα Docker έχοντας εντός της, πλέον, δύο containers.



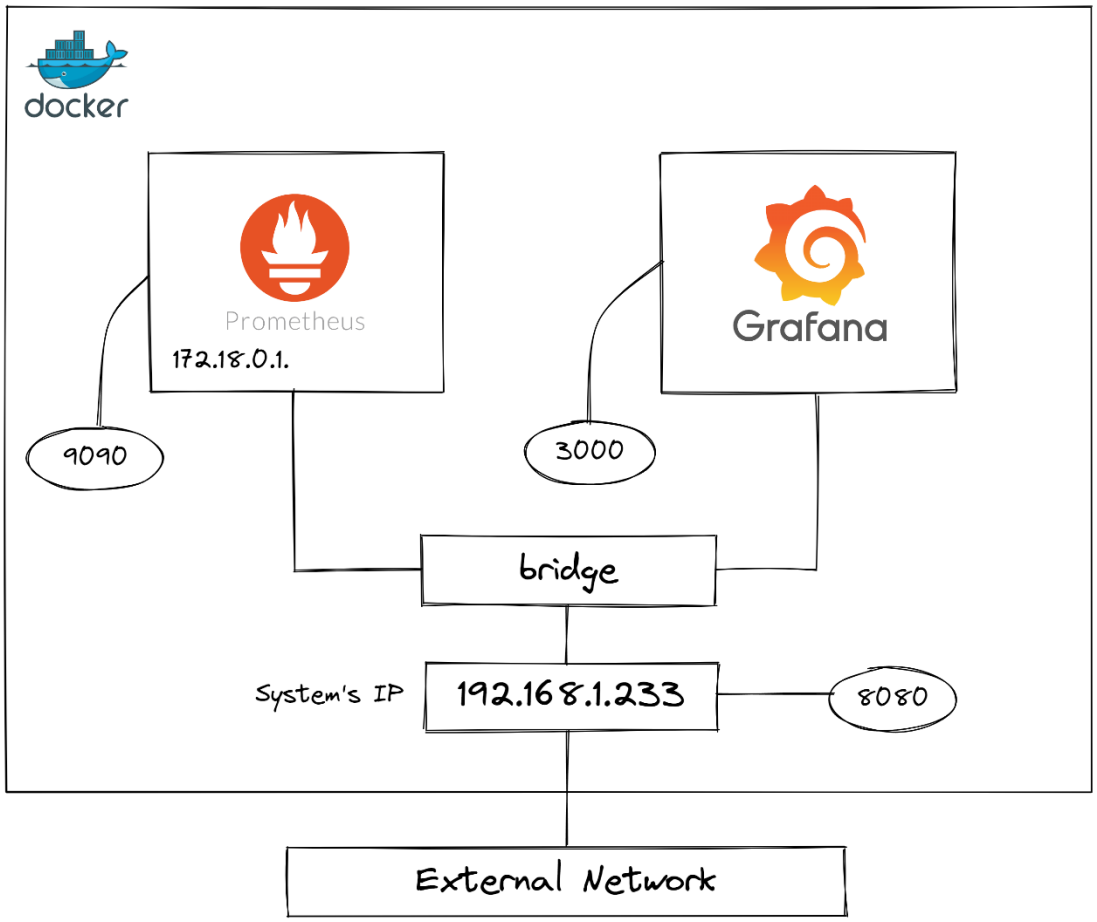

**Εικόνα 46. Prometheus IP Container Address**

Για λόγους καλύτερης απεικόνισης, δημιουργήθηκε το διάγραμμα της εικόνας 48 μέσα από την online εφαρμογή Excalidraw. Στο διάγραμμα αυτό, απεικονίζεται ο host ο οποίος εκτελείται

σε περιβάλλον Docker. Από τις προηγούμενες παραγράφους, έγινε γνωστό ότι μπορεί να γίνει χρήση τρίτων εφαρμογών με χρήση containerized software αποφεύγοντας έτσι, τοπικές εγκαταστάσεις των Prometheus και Grafana. Το διάγραμμα λοιπόν, αποτυπώνει τα container των εξωτερικών εφαρμογών, καθεμιά από τις οποίες «ακούει» στην δική της θύρα.

Εξ αρχής η πλατφόρμα Docker ορίζει ένα δίκτυο επικοινωνίας με την ονομασία bridge, το οποίο επιτρέπει την επικοινωνία μεταξύ των containers. Το δίκτυο αυτό, επικοινωνεί με την σειρά του, με την διεύθυνση IP του τερματικού στην οποία εκτελείται. Όπως φαίνεται και στο διάγραμμα αυτό, η διεύθυνση IP ακούει στην θύρα 8080, πάνω στην οποία εκτελείται η εφαρμογή Spring Boot.

Το ζητούμενο λοιπόν, ήταν να υλοποιηθεί η επικοινωνία μεταξύ των δύο containers που βρίσκονταν ήδη μέσα στον Docker host. Για να γίνει αυτό, ήταν απαραίτητη η εύρεση της διεύθυνσης IP του container που περιβάλλει την εφαρμογή της Prometheus.

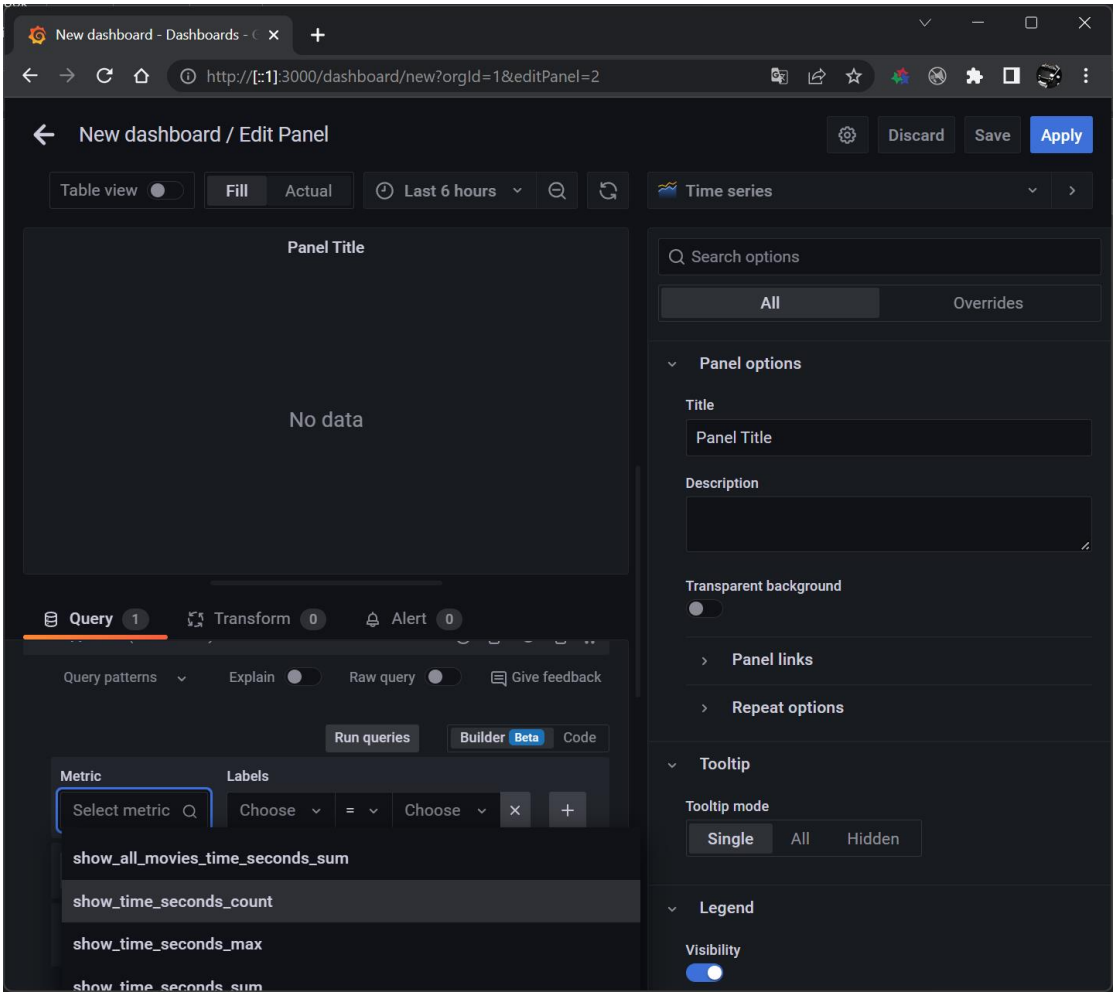



**Εικόνα 47. Docker Network Architecture**

Για την εύρεση της, έγινε χρήση της εντολής docker inspect ακολουθούμενη από το μοναδικό κωδικό ID που αντιπροσωπεύει το container της Prometheus. Αξίζει να σημειωθεί ότι τα IDs όλων των εγκατεστημένων containers μπορούν να απεικονισθούν με την εντολή «docker ps». Η εκτέλεση της docker inspect <Container ID>, θα δώσει ένα εκτενές αρχείο τύπου JSON, όπου στο τέλος του οποίου υπάρχει το κλειδί «Network» και εντός του η IP διεύθυνση για το συγκεκριμένο container. Η σωστή λοιπόν, διεύθυνση η οποία τοποθετήθηκε στις ρυθμίσεις data source ήταν η http://172.18.0.1.:9090.

### 4.3.3 Δημιουργία Dashboard

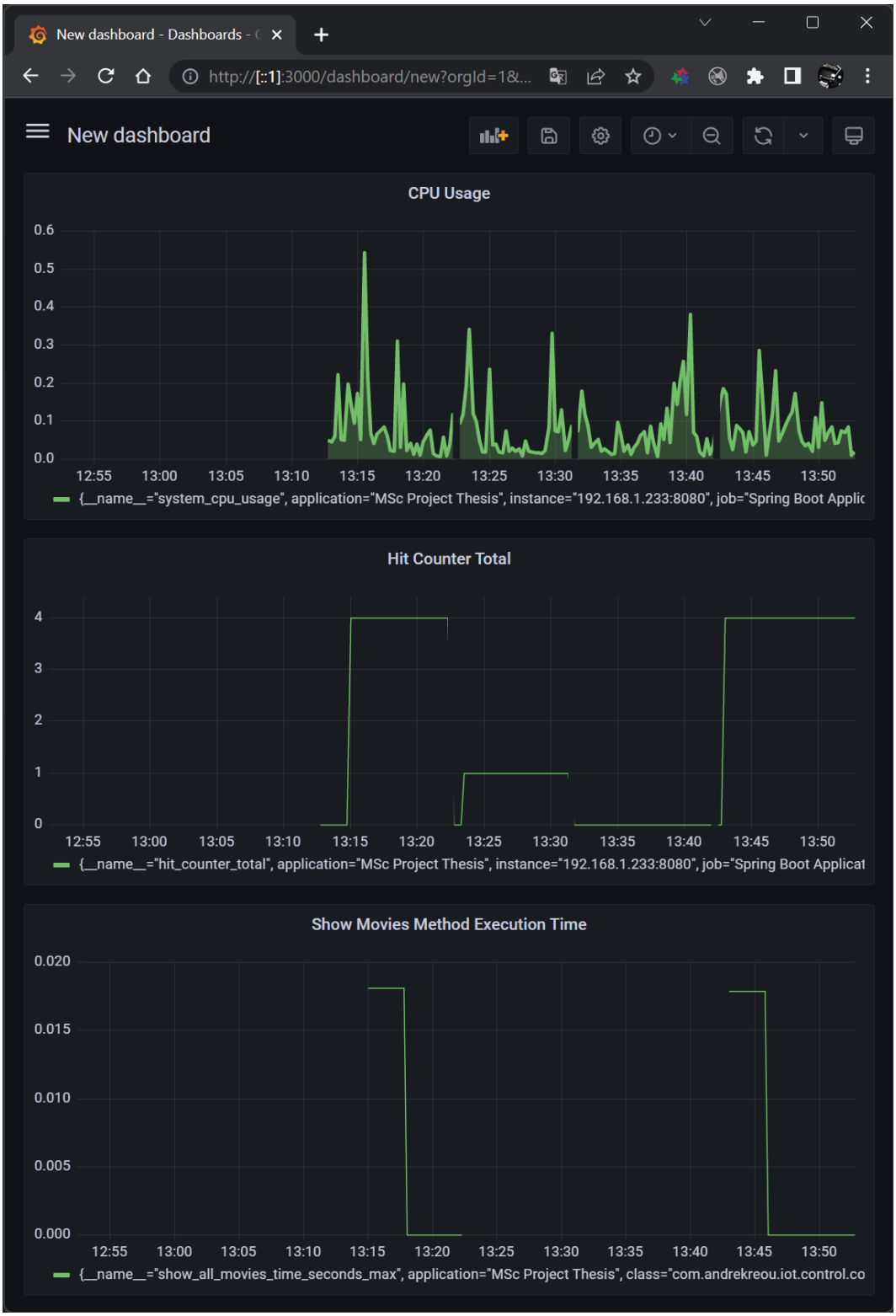
Η Grafana παρέχει ένα εύχρηστο και διαδραστικό μενού, για την υλοποίηση των λειτουργιών που παρέχει. Μέσω της επιλογής «Dashboards» στα αριστερά της οθόνης, ο χρήστης μπορεί να προσθέσει νέο dashboard. Εκτός της κατασκευής, παρέχεται η δυνατότητα εισαγωγής έτοιμου dashboard μέσω των templates που παρέχονται από την ιστοσελίδα https://grafana.com/grafana/dashboards/.




**Εικόνα 48. Παραμετροποίηση Panel ανά Metric**

Με την συγκεκριμένη επιλογή, ο χρήστης θα οδηγηθεί στο επόμενο μενού, το οποίο και ζητά το είδος του dashboard που εκείνος επιθυμεί (panel, row κτλ.). Με την επιλογή Panel, θα προκύψει η εικόνα 49, στην οποία θα πρέπει να επιλεχθεί το είδος του metric προς απεικόνιση, όπως ακριβώς έγινε και στην Prometheus. Εδώ για λόγους επίδειξης έχει επιλεγεί το custom timer metric που δημιουργήθηκε στην εφαρμογή Spring Boot μαζί με το hit counter metric και το CPU usage. Στο δεξί μέρος του μενού αυτού, περιλαμβάνεται μια μεγάλη ποικιλία εργαλείων διαμόρφωσης του εκάστοτε πάνελ, όπως την μορφή του, αν θα είναι συνεχής γραμμής ή μπάρες.

Με την ολοκλήρωση της διαδικασίας διαμόρφωσης, θα πρέπει να πατηθεί το πλήκτρο «Apply» έτσι ώστε να εφαρμοσθούν οι αλλαγές και να σωθεί το νέο πάνελ, στο dashboard. Η διαδικασία αυτή πρέπει να γίνει ξεχωριστά για κάθε metric, το οποίο με τη σειρά του θα έχει το δικό του panel, στο συνολικό dashboard. Ως αποτέλεσμα, των ανωτέρω ενεργειών, θα προκύψει το dashboard της εικόνας 50.


New dashboard - Dashboards -
http://[::1]:3000/dashboard/new?orgId=1&...
New dashboard
CPU Usage
0.6
0.5
0.4
0.3
0.2
0.1
0.0
12:55 13:00 13:05 13:10 13:15 13:20 13:25 13:30 13:35 13:40 13:45 13:50
{__name__="system_cpu_usage", application="MSc Project Thesis", instance="192.168.1.233:8080", job="Spring Boot Applic
Hit Counter Total
4
3
2
1
0
12:55 13:00 13:05 13:10 13:15 13:20 13:25 13:30 13:35 13:40 13:45 13:50
{__name__="hit_counter_total", application="MSc Project Thesis", instance="192.168.1.233:8080", job="Spring Boot Applicat
Show Movies Method Execution Time
0.020
0.015
0.010
0.005
0.000
12:55 13:00 13:05 13:10 13:15 13:20 13:25 13:30 13:35 13:40 13:45 13:50
{__name__="show_all_movies_time_seconds_max", application="MSc Project Thesis", class="com.andrekreou.iot.control.co

**Εικόνα 49. Ολοκληρωμένο Dashboard**

Με την εισαγωγή ολοκληρωμένου template, αποκτάται πρόσβαση σε περισσότερα metrics, έχοντας κάνει την μικρότερη δυνατή παραμετροποίηση. Η εικόνα 50, δείχνει το εύρος δυνατοτήτων απεικόνισης της Grafana.


**Εικόνα 50. Template Dashboard Ολοκληρωμένων Metrics**

# 5 ΠΡΑΚΤΙΚΗ ΥΛΟΠΟΙΗΣΗ

Με όλες τις βάσεις να έχουν τεθεί στα προηγούμενα κεφάλαια, ήρθε η στιγμή να αποτυπωθούν τα αποτελέσματα με πρακτική υλοποίηση όλων των υποεφαρμογών, που επιλέχθηκαν από τον συγγραφέα φοιτητή. Ανά ξεχωριστές ενότητες, θα αναπτυχθούν οι λειτουργίες και τα αποτελέσματα αυτών, μέσω εικόνων τύπου print screen, για την όσο το δυνατόν καλύτερη διατύπωση και αναπαράσταση. Ειδικά σ ότι αφορά την πρώτη ενότητα 4.1, θα υπάρξει τμήμα ειδικό στο οποίο θα αναλυθεί πρόβλημα το οποίο, ο συγγραφέας φοιτητής δεν κατόρθωσε να επιλύσει επιδεικνύοντας ωστόσο τις προσπάθειες που έκανε, για το αντίθετο.

Για λόγους σαφήνειας και καλύτερης οπτικοποίησης, θα αποφευχθεί η γραπτή αναπαράσταση του κώδικα, του οποίου έγινε η συγγραφή. Ωστόσο ο τελευταίος, θα αναφερθεί σε ειδική ενότητα στο παράρτημα του παρόντος εγγράφου, καθαρά για πληρέστερη απεικόνιση.

## 5.1 Υπηρεσία Cryptocurrency News

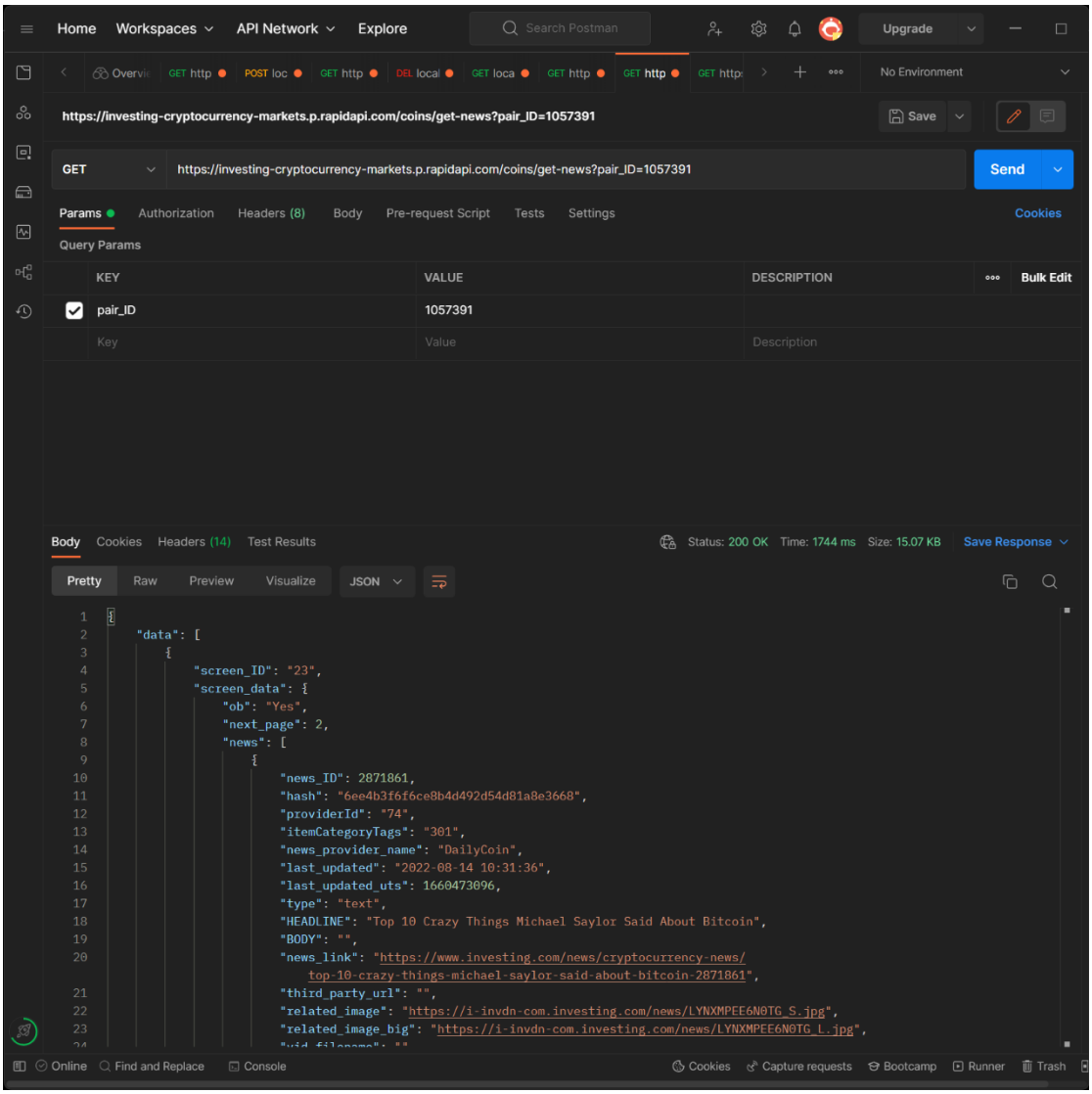
Στην ενότητα αυτή, θα πραγματοποιηθεί η κατανάλωση μιας υπηρεσίας παροχής δεδομένων στον τομέα του cryptocurrency, βασικός σκοπός της οποίας είναι η παροχή ειδήσεων στον τομέα αυτόν. Η υπηρεσία αυτή, παρέχεται από τον σύνδεσμο

https://investing-cryptocurrency-markets.p.rapidapi.com/coins/get-news?pair_ID=1057391

της εταιρίας Investing, η οποία ειδικεύεται στην παροχή λύσεων και υπηρεσιών στον τομέα των κρυπτονομισμάτων. Το δοθέν URL παρέχει την πληροφορία παροχής ειδήσεων αναφορικά με τον τομέα του cryptocurrency. Εντός του εγγράφου JSON, το οποίο φαίνεται στην παρακάτω εικόνα της εφαρμογής Postman, παρέχονται σημαντικά πεδία όπως:

- Το μοναδικό ID number ανά είδηση.
- Το URL που μπορούν οι χρήστες να οδηγηθούν σ αυτό για να πληροφορηθούν για την εκάστοτε είδηση.
- Αρχεία εικόνας για την καλύτερη οπτικοποίηση των ειδήσεων σε ιστοσελίδες.
- Τον αρχικό πάροχο ή ειδησεογραφικό πρακτορείο από τον οποίο προήλθε η τελική είδηση και αναδιατυπώθηκε στην σελίδα https://www.investing.com/.

- Timestamp που αφορά στην ημερομηνία έκδοσης μιας είδησης.




**Εικόνα 51. Έγγραφο JSON Κατανάλωσης της Σελίδας Investing.com**

Αξίζει να αναφερθεί ότι η απευθείας χρήση του προαναφερθέντος URL στον εκάστοτε browser, δεν θα δώσει τα αποτελέσματα της παραπάνω εικόνας διότι η συγκεκριμένη API προστατεύεται με χρήση κλειδιών και λοιπών headers τα οποία αναγράφονται παρακάτω.

X-RapidAPI-Key: 605a619252msh709d3d6fa6fbdcdp155ef4jsn6cd36a1a4195

X-RapidAPI-Host: investing-cryptocurrency-markets.p.rapidapi.com

Η παράμετρος X-RapidAPI-Key δόθηκε στον συγγραφέα φοιτητή κατά την εγγραφή του στην ιστοσελίδα Investing για την δυνατότητα κατανάλωσης της API της τελευταίας. Οι ανωτέρω παράμετροι, εφόσον τοποθετηθούν στην καρτέλα «Headers» της εφαρμογής Postman, μαζί εννοείται με το κυρίως URL κλήσης της εξωτερικής API, θα δώσει τα απεικονιζόμενα αποτελέσματα. Τα επόμενα βήματα περιλαμβάνουν την εισαγωγή κάποιων από τα δεδομένα αυτά, στην αναπτυσσόμενη εφαρμογή Spring Boot και η τοποθέτηση τους σε βάση δεδομένων.

### 5.1.1 Επιλογή Δεδομένων προς Απεικόνιση

Από το αρχείο JSON της προηγούμενης εικόνας, επιλέχθηκαν τα πεδία news_id news_provider_name, Headline, news_link και related_image. Σκοπός του συγγραφέα φοιτητή ήταν η τοποθέτηση τους σε πίνακα σε αρχείο HTML για την καλύτερη δυνατή απεικόνιση τους. Ο πίνακας αυτός, λοιπόν, θα εμπεριέχει το όνομα του ειδησεογραφικού πρακτορείου, τον τίτλο της είδησης, τον υπερσύνδεσμο που θα οδηγεί σ αυτήν κατ' επέκταση σε ιστοσελίδα διαφορετική από αυτήν της εφαρμογής και τέλος ένα αρχείο εικόνας που συνοδεύει κάθε είδηση ξεχωριστά.

Η δέσμευση των δεδομένων των πεδίων αυτών έγινε με χρήση μια κλάσης τύπου POJO, υπεύθυνης για το mapping των κλειδιών του αρχείου JSON, κρατώντας τα ονόματα των πεδίων του κώδικα, ακριβώς ίδια μ αυτά του αρχείου JSON.

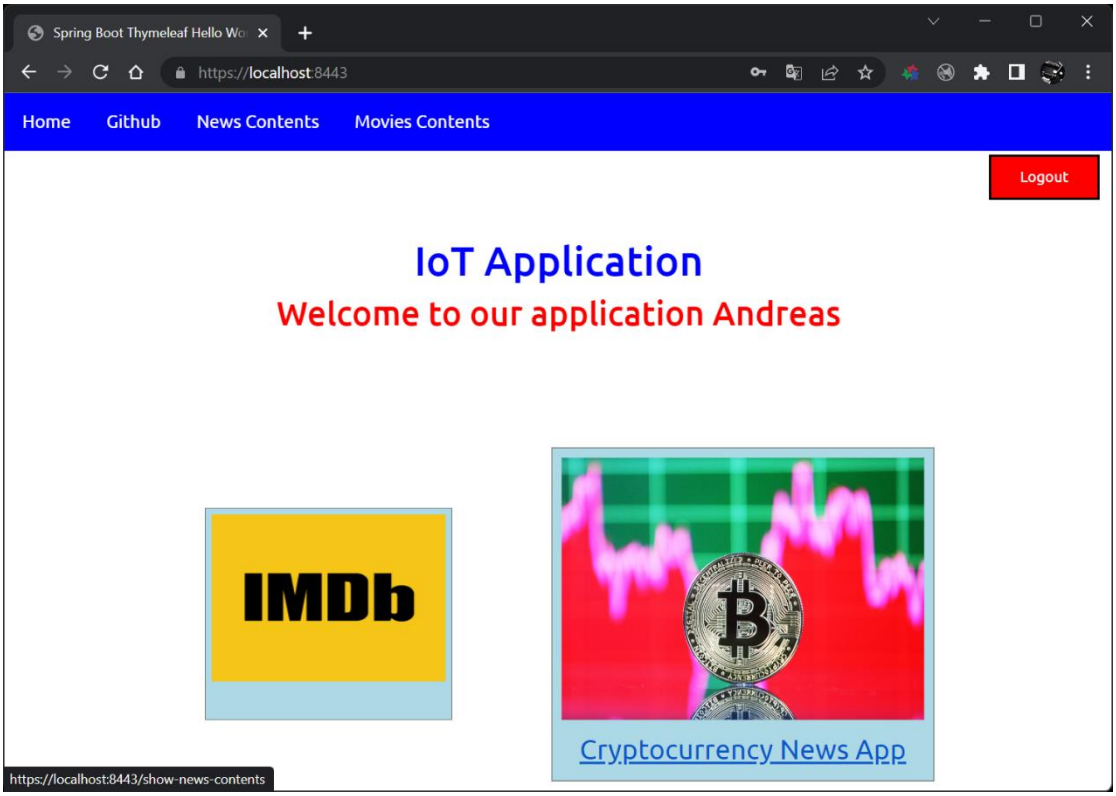
```
@Id
private Integer news_id;
private String news_provider_name;
@JsonProperty("headline")
private String HEADLINE;
private String news_link;
private String related_image;
```

**Εικόνα 52. Επιλεγμένα Κλειδιά Κλάσης POJO Cryptocurrency News**

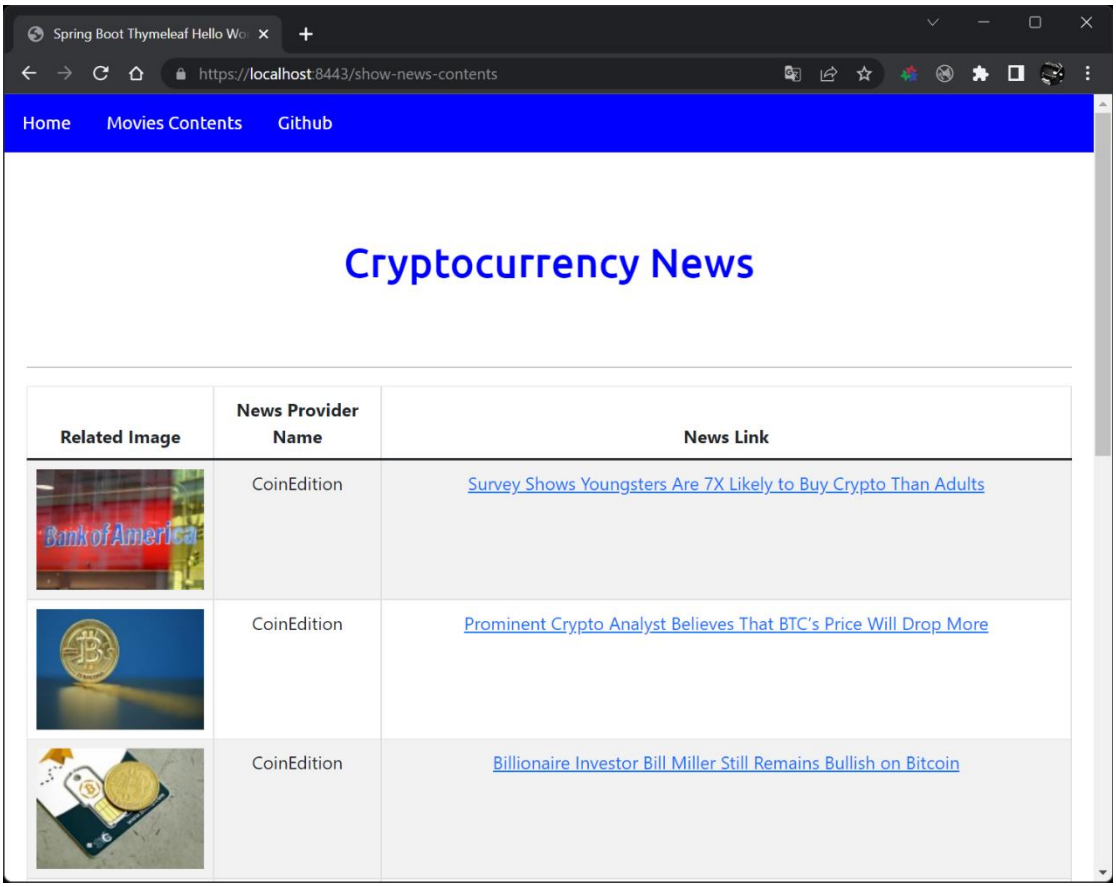
### 5.1.2 Εκτέλεση Cryptocurrency News Υπηρεσίας

Για την είσοδο του εκάστοτε χρήστη στην εφαρμογή, απαιτείται η αυθεντικοποίηση του, με την διαδικασία που περιγράφηκε στην ενότητα .3.1.4.3. Εφόσον η διαδικασία κυλήσει κανονικά, θα βρεθεί στην αρχική οθόνη της εφαρμογής, όπως αυτή απεικονίζεται στην παρακάτω εικόνα. Όπως φαίνεται εκεί, γίνεται αποτύπωση και των δύο υπηρεσιών που λαμβάνουν χώρα, ωστόσο για τους σκοπούς της παραγράφου αυτής, επιλέγεται η υπηρεσία «Cryptocurrency News App».

Με πάτημα στο ανάλογο παράθυρο της εφαρμογής, στο πεδίο της ονομασίας της υπηρεσίας, γίνεται η μετάβαση στο επόμενο HTML Thymeleaf View, το οποίο φαίνεται στην εικόνα 54. Στην πρώτη στήλη φαίνεται το related image ανά είδηση, στην δεύτερη το ειδησεογραφικό πρακτορείο και στην τρίτη, ο υπερσύνδεσμος όπου με το πάτημα του οποίου, γίνεται η μετάβαση στην είδηση, εκτός της παρούσας εφαρμογής.




**Εικόνα 53. Welcome Screen - Cryptocurrency News App Selected**




**Εικόνα 54. Λίστα ειδήσεων περί του cryptocurrency**

### 5.1.3 Πρόβλημα URL Redirection

Κατά την υλοποίηση της παρούσας υπηρεσίας, αντιμετωπίσθηκε πρόβλημα κατά την επανακατεύθυνση των χρηστών από το εκάστοτε URL της είδησης. Αντί λοιπόν, η τελευταία να γίνεται απευθείας στον URL σύνδεσμο της, πραγματοποιείται redirection στην αρχική σελίδα της εταιρίας Investing. Δοκιμάστηκαν διάφορες μέθοδοι και πραγματοποιήθηκε ερώτημα στην πλατφόρμα Stack Overflow για το ζήτημα αυτό. Το τελικό συμπέρασμα ήταν ότι η ιστοσελίδα της εταιρίας Investing, έχει την ικανότητα να μπλοκάρει αιτήματα τύπου cross origin resource sharing τα οποία προέρχονται από domain με το όνομα localhost.

Κατ' επέκταση, το πάτημα όλων των εικονιζόμενων URLs θα δρομολογήσει τους χρήστες στην αρχική σελίδα www.investing.com και όχι στο URL της αντίστοιχης είδησης. Ωστόσο κατά το hover event με το ποντίκι πάνω από το URL, φαίνεται ξεκάθαρα ο πλήρης σύνδεσμος ενώ με το δεξί κλικ και εν συνεχεία αντιγραφή και επικόλληση αυτού, η είδηση εμφανίζεται ως έπρεπε.

## 5.2 Υπηρεσία IMDb

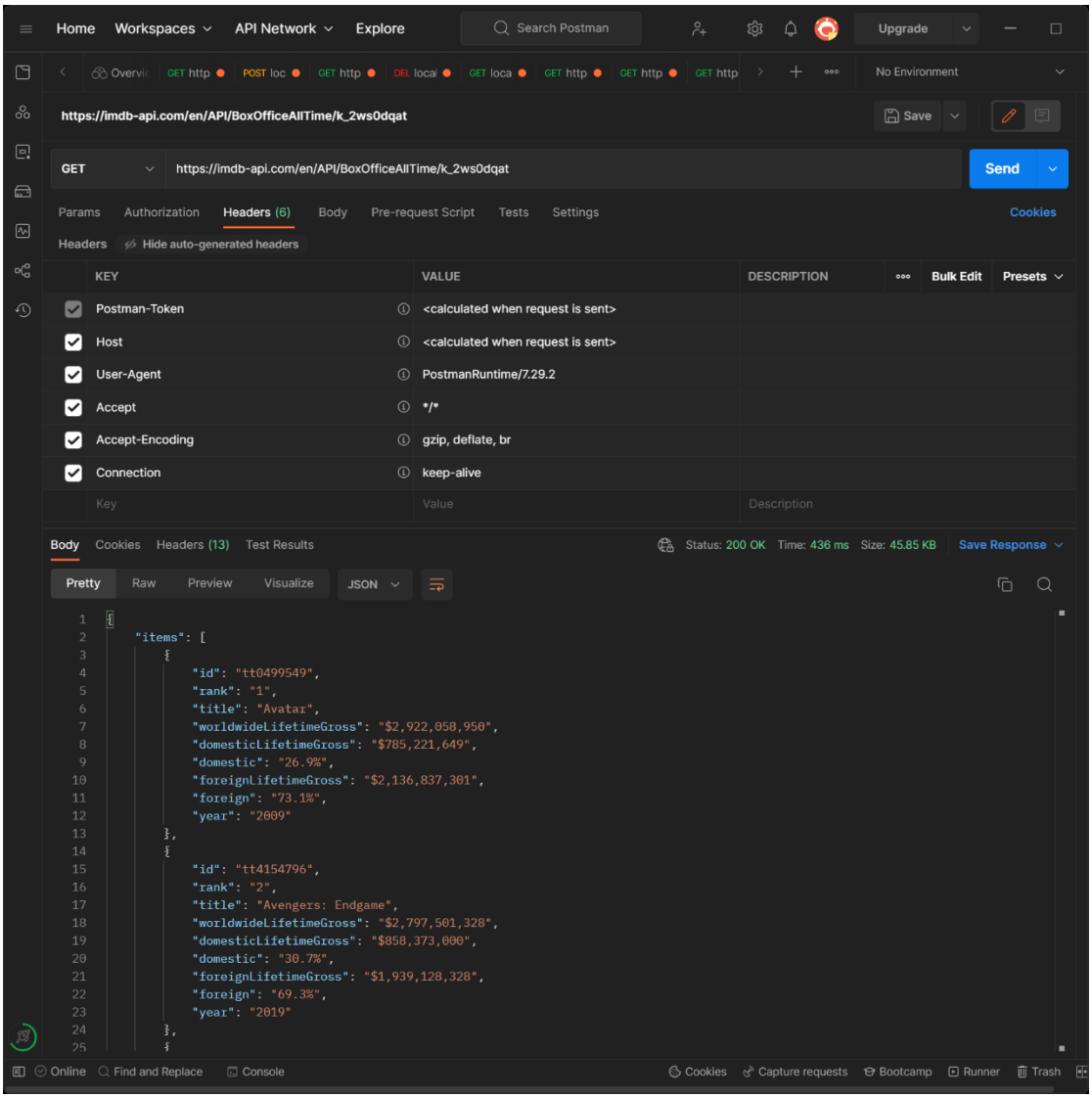
Στην ενότητα αυτή, θα πραγματοποιηθεί η κατανάλωση μιας υπηρεσίας παροχής δεδομένων στον τομέα του κινηματογράφου, βασικός σκοπός της οποίας είναι η παροχή δεδομένων στον τομέα των καλύτερων Box Office ταινιών. Η υπηρεσία αυτή, παρέχεται από τον σύνδεσμο

https://imdb-api.com/en/API/BoxOfficeAllTime/k_2ws0dqat

της εταιρίας IMDb, η οποία ειδικεύεται στην παροχή πληροφοριών στον τομέα της 7ης τέχνης. Εντός του εγγράφου JSON, το οποίο φαίνεται στην παρακάτω εικόνα της εφαρμογής Postman, παρέχονται σημαντικά πεδία όπως:

- Το μοναδικό ID number ανά ταινία.
- Τον τίτλο της ταινίας.
- Την κατάταξη της ταινίας βάσει του ποσού που έχει συγκεντρώσει σε εισπράξεις.
- Το ποσό εισπράξεων υπολογισμένο σε παγκόσμιο επίπεδο.
- Το έτος όπου η ταινία βγήκε στις κινηματογραφικές αίθουσες.
- Τα ποσοστά εισπράξεων σε παγκόσμιο και εθνικό επίπεδο

Μια σημαντική παρατήρηση έχει να κάνει στο γεγονός ότι εν αντιθέσει με την προηγούμενη υπηρεσία, δεν υπάρχουν οι ίδιοι περιορισμοί ασφαλείας στην API της IMDb. Το μόνο χαρακτηριστικό ασφαλείας είναι ο αριθμός API Token, ο οποίος λήφθηκε με την εγγραφή του χρήστη στην υπηρεσία και τοποθετήθηκε στο τέλος του δοθέντος υπερσυνδέσμου, ήτοι ο k_2ws0dqat.




**Εικόνα 55. Έγγραφο JSON απο την IMDb υπηρεσία**

### 5.2.1 Επιλογή Δεδομένων Υπηρεσίας IMDb προς Απεικόνιση

Από το αρχείο JSON της προηγούμενης εικόνας, επιλέχθηκαν τα πεδία title, worldwideLifetimeGross και year. Όπως και στην προηγούμενη ενότητα, σκοπός ήταν η τοποθέτηση τους σε πίνακα σε αρχείο HTML για την καλύτερη δυνατή απεικόνιση τους. Ο πίνακας αυτός, λοιπόν, θα εμπεριέχει τον τίτλο της ταινίας, το συνολικό ποσό εισπράξεων σε παγκόσμιο επίπεδο καθώς και το έτος που αυτή βγήκε στις αίθουσες.

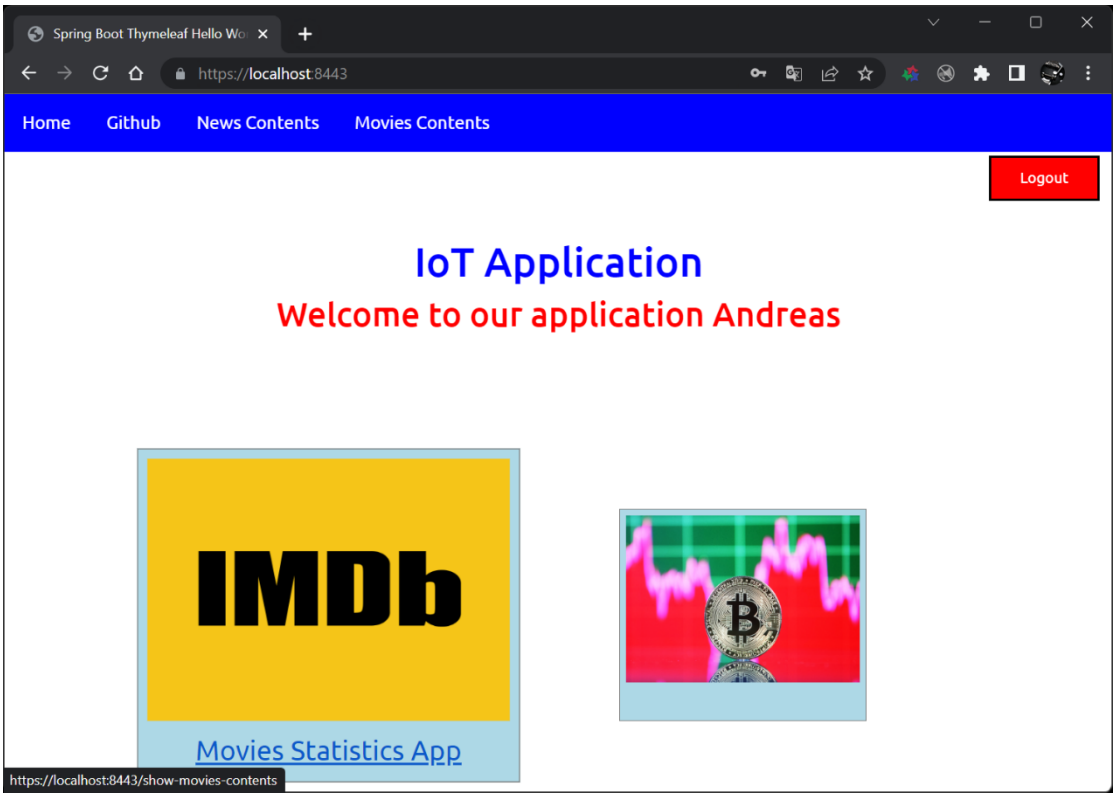
Όπως και στην προηγούμενη υπηρεσία, έγινε χρήση μιας κλάσης τύπου POJO για την χαρτογράφηση των πεδίων από το αρχείο JSON, κρατώντας τα ονόματα ίδια με του αρχείου. Στην εικόνα που ακολουθεί, έχει προστεθεί και ένα επιπρόσθετο πεδίο υπό το όνομα id, το οποίο θα χρησιμοποιηθεί από την βάση δεδομένων ως πρωτεύον κλειδί με auto increment τιμές.

```
@Id
private String id;
private String title;
private String worldwideLifetimeGross;
private Integer year;
```

**Εικόνα 56. Επιλεγμένα Κλειδιά Κλάσης POJO IMDb**

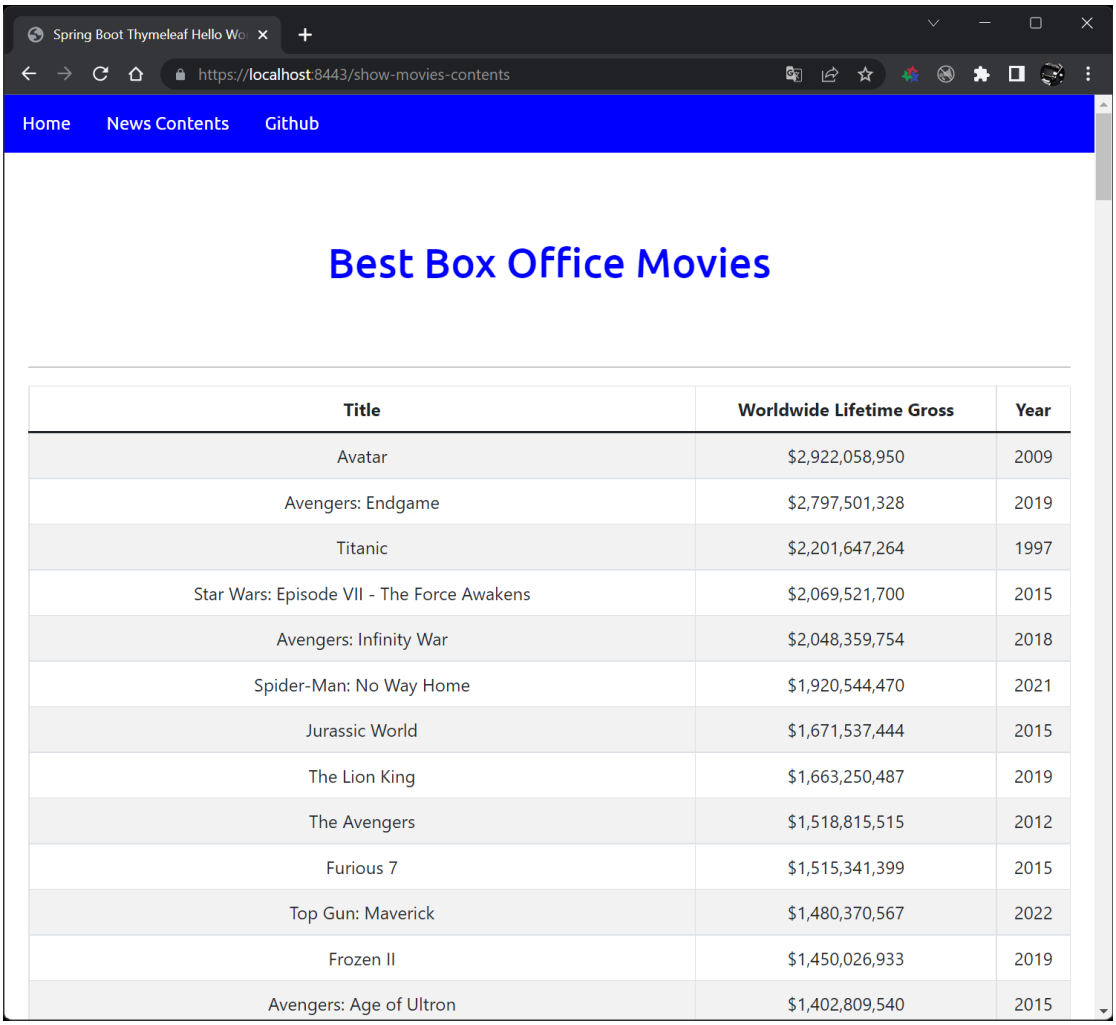
### 5.2.2 Εκτέλεση Υπηρεσίας IMDb

Η εκτέλεση της υπηρεσίας αυτής ακολουθεί την ίδια διαδικασία μ αυτή που εκτελέσθηκε στην παράγραφο 5.1.2. Με την ολοκλήρωση της αυθεντικοποίησης του εκάστοτε χρήστη, προκύπτει η παρακάτω εικόνα, αυτή τη φορά έχοντας επιλεγμένη την υπηρεσία IMDb.




**Εικόνα 57. Welcome Screen - IMDb Service Selected**

Με πάτημα στο ανάλογο παράθυρο της εφαρμογής, στο πεδίο της ονομασίας της υπηρεσίας, γίνεται η μετάβαση στο επόμενο HTML Thymeleaf View, το οποίο φαίνεται στην παρακάτω εικόνα. Στην πρώτη στήλη φαίνεται ο τίτλος της ταινίας, στην δεύτερη το συνολικό ποσό εισπράξεων σε παγκόσμιο επίπεδο και στην τρίτη, το έτος κατά το οποίο η εκάστοτε ταινία προβλήθηκε στις κινηματογραφικές αίθουσες.




**Εικόνα 58. Best Box Office Movies List**

# ΒΙΒΛΙΟΓΡΑΦΙΑ

1. History of the Internet. (χ.χ.). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/History_of_the_Internet

2. Παπαργύρη, Τ. (2012, Ιούνιος). Πανεπιστήμιο Πατρών. Ανάκτηση από Πανεπιστήμιο Πατρών: https://nemertes.library.upatras.gr/jspui/bitstream/10889/6122/3/Nimertis_Papargyri(ele).pdf

3. Kalin, M. (2013). Java Web Services, Up and Running (2nd εκδ.). Sebastopol: O'Reilly Media, Inc. Ανάκτηση από https://www.oreilly.com/library/view/java-web-services/9781449373856/

4. Representational state transfer. (χ.χ.). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Representational_state_transfer

5. API vs Web Service: What's the Difference? (χ.χ.). Ανάκτηση από Last Call. The RapidAPI Blog: https://rapidapi.com/blog/api-vs-web-service/

6. Difference between REST API and SOAP API. (χ.χ.). Ανάκτηση από GeeksforGeeks: https://www.geeksforgeeks.org/difference-between-rest-api-and-soap-api/?ref=lbp

7. Types of APIs. (χ.χ.). Ανάκτηση από Last Call. The RapidAPI Blog: https://rapidapi.com/blog/types-of-apis/

8. Types of Web Services. (χ.χ.). Ανάκτηση από Educba: https://www.educba.com/types-of-web-services/

9. What is an API? (χ.χ.). Ανάκτηση από Altexsoft. Software R&D Engineering: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/

10. Walls, C. (2022). Spring In Action (Sixth εκδ.). Shelter Island, NY: Manning Publications Co. Ανάκτηση από https://www.manning.com/books/spring-in-action-sixth-edition?query=Craig%20Walls

11. A Comparison Between Spring and Spring Boot. (χ.χ.). Ανάκτηση από Baeldung: https://www.baeldung.com/spring-vs-spring-boot

12. Microsoft Integration Stencils Pack V2.0 for Visio 2016/2013 is now available. (χ.χ.). Ανάκτηση από Sandro Pereira Blog: https://blog.sandro-pereira.com/2016/06/16/microsoft-integration-stencils-pack-v2-0-for-visio-20162013-is-now-available/

13. Dependency Injection. (χ.χ.). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Dependency_injection

14. Greengard, S. (2021). The Internet of Things. Revised and Updated Edition, Cambridge, England: The MIT Press. Ανάκτηση από https://mitpress.mit.edu/books/internet-things

15. Παπασταθοπούλου, Α. (2017, Φεβρουάριος). Πανεπιστήμιο Μακεδονίας. Ανάκτηση από Πανεπιστήμιο Μακεδονίας: https://dspace.lib.uom.gr/bitstream/2159/20157/4/PapastathopoulouAlexandraMsc2017.pdf

16. Big Data. (χ.χ.). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Big_data

17. Rational Unified Process (RUP). (χ.χ.). Ανάκτηση από Toolshero: https://www.toolshero.com/information-technology/rational-unified-process-rup/

18. Rational Unified Process. (χ.χ.). Ανάκτηση από Wikipedia: https://en.wikipedia.org/wiki/Rational_Unified_Process

19. Laudon, Kenneth C. – Laudon, Jane P. (2009). Πληροφοριακά Συστήματα Διοίκησης (8η Αμερικάνικη Έκδοση). Αθήνα: Εκδόσεις Κλειδάριθμος. Ανάκτηση από https://www.klidarithmos.gr/plhroforiaka-systhmata-dioikhshs-8h-amerikanikh-ekdosh

20. Seidl, Martina. – Scholz, Marion. – Huemer, Christian. – Kappel, Gerti (2015). UML @ Classroom (2η Έκδοση). Heidelberg, Germany: Springer. Ανάκτηση από https://link.springer.com/book/10.1007/978-3-319-12742-2

21. Τριάντης, Κ. (2013). Πανεπιστήμιο Πειραιά. Ανάλυση Απαιτήσεων για την Ανάπτυξη Πληροφοριακών Συστημάτων. Μεθοδολογίες Ανάλυσης Απαιτήσεων στο Πλαίσιο Εναλλακτικών Κύκλων Ζωής Έργων Πληροφοριακών Συστημάτων. Διενέργεια Σχετικής Μελέτης Περίπτωσης. Ανάκτηση από Πανεπιστήμιο Πειραιά: https://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/5921/Triantis.pdf?sequence=2&isAllowed=y

22. What is N-Tier Architecture? How It Works, Examples, Tutorials, and More. (χ.χ.). Ανάκτηση από Stackify: https://stackify.com/n-tier-architecture/

23. Introduction to the POM. (χ.χ.). Ανάκτηση από Apache Maven Project: https://maven.apache.org/guides/introduction/introduction-to-the-pom.html

24. Mitchell, Robin (2020). HTTP vs MQTT - Which Communication Protocol Should You Use in Your IoT Application?. (χ.χ.). Ανάκτηση από Eelectromaker.io: https://www.electromaker.io/blog/article/http-vs-mqtt

25. MQTT: The Standard for IoT Messaging. (χ.χ.). Ανάκτηση από MQTT: https://mqtt.org/

26. Introducing JSON. (χ.χ.). Ανάκτηση από: https://www.json.org/json-en.html

27. Introduction to Postman for API Development. (χ.χ.). Ανάκτηση από GeeksforGeeks: https://www.geeksforgeeks.org/introduction-postman-api-development/

28. Thymeleaf. (χ.χ.). Ανάκτηση από: https://www.thymeleaf.org/

29. Configure HTTPS for Spring Boot application on localhost with self-signed certificate (χ.χ.). Ανάκτηση από https://www.youtube.com/watch?v=eBEq0Kv7vsw

30. Vitale, Thomas (2017). How to enable HTTPS in a Spring Boot Java application. (χ.χ.). Ανάκτηση από: https://www.thomasvitale.com/https-spring-boot-ssl-certificate/

31. Zanini, Antonello (2021). A Complete Guide to Lombok. (χ.χ.). Ανάκτηση από: https://auth0.com/blog/a-complete-guide-to-lombok/

32. Carlos Valero Sánchez, José (2022). Spring Boot Actuator. (χ.χ.). Ανάκτηση από: https://www.baeldung.com/spring-boot-actuators
33. Prometheus Overview. (χ.χ.). Ανάκτηση από: https://prometheus.io/docs/introduction/overview/
34. Dashboard anything. Observe everything. (χ.χ.). Ανάκτηση από: https://grafana.com/grafana/
35. Donohue, Tom (2022). Spring Boot app metrics - with Prometheus and Micrometer. (χ.χ.). Ανάκτηση από: https://www.tutorialworks.com/spring-boot-prometheus-micrometer/
36. Costa, Marcelo (2020). How to Get A Docker Container IP Address - Explained with Examples. (χ.χ.). Ανάκτηση από: https://www.freecodecamp.org/news/how-to-get-a-docker-container-ip-address-explained-with-examples/