|  |  |
| --- | --- |
| http://www.paso.gr/wp-content/uploads/2013/01/%CE%A0%CE%B1%CE%BD%CE%B5%CF%80%CE%B9%CF%83%CF%84%CE%AE%CE%BC%CE%B9%CE%BF-%CE%9A%CF%81%CE%AE%CF%84%CE%B7%CF%82.jpg | Πανεπιστήμιο Κρήτης  Τμήμα Επιστήμης Υπολογιστών |

Ανάπτυξη REST API και διαδικτυακής εφαρμογής οπτικοποίησης ανοικτών δεδομένων που αφορούν την καταγεγραμμένη σεισμική δραστηριότητα από το Εθνικό Αστεροσκοπείο Αθηνών.

Development of a REST API and a web-based client application for the visualization of open data related to the recorded seismic activity by the National Observatory of Athens.

**Χαράλαμπος Ευριπίδου**

Επιβλέπων: ΠΙΤΙΚΑΚΗΣ ΜΑΡΙΟΣ

Επόπτης: ΤΖΙΤΖΙΚΑΣ ΙΩΑΝΝΗΣ

Μέλος της Επιτροπής Παρακολούθησης/Αξιολόγησης: ΠΛΕΞΟΥΣΑΚΗΣ ΔΗΜΗΤΡΙΟΣ

Σεπτέμβριος 2022

**Περιεχόμενα**

[1 Εισαγωγή 3](#_Toc115105823)

[1.1 Γενική περιγραφή 4](#_Toc115105824)

[1.2 Σκοπός συστήματος 4](#_Toc115105825)

[2 Σενάρια Χρήσης και Τύποι Χρηστών 5](#_Toc115105826)

[2.1 Περιγραφή 5](#_Toc115105827)

[2.2 Σενάρια Χρήσης 6](#_Toc115105828)

[2.3 Τύποι χρήστη 7](#_Toc115105829)

[3 Απαιτήσεις 8](#_Toc115105830)

[4 Οθόνες της Εφαρμογής 8](#_Toc115105831)

[4.1 Swagger 9](#_Toc115105832)

[4.2 Postman 11](#_Toc115105833)

[4.3 Main Page (index.html) 12](#_Toc115105834)

[5 Σχεδίαση και Μοντελοποίηση 15](#_Toc115105835)

[5.1 Σχεδιασμός Βάσης Δεδομένων 15](#_Toc115105836)

[5.2 Σχεδίαση των project 17](#_Toc115105837)

[5.3 Μοντελοποίηση 17](#_Toc115105838)

[5.3.1 Επεξεργασία αρχείων 17](#_Toc115105839)

[5.3.2 Βάση Δεδομένων 18](#_Toc115105840)

[5.3.3 Front End, Back End and User Interface 19](#_Toc115105841)

[5.3.3.1 Front End 20](#_Toc115105842)

[*5.3.3.2* *Back End* 20](#_Toc115105843)

[6 Αρχιτεκτονική του Συστήματος και Εργαλεία 21](#_Toc115105844)

[6.1 Environment setup and deployment 21](#_Toc115105845)

[6.1.1 IDE and Application Server 21](#_Toc115105846)

[6.1.2 Project Creation 21](#_Toc115105847)

[6.2 Τεχνολογίες/frameworks used 22](#_Toc115105848)

[6.2.1 Spring Framework and Spring Boot 22](#_Toc115105849)

[6.2.2 Eclipse Jersey 22](#_Toc115105850)

[6.2.3 Swagger 23](#_Toc115105851)

[6.2.4 Leaflet 23](#_Toc115105852)

[6.2.5 D3.js 23](#_Toc115105853)

[6.2.6 Postman 23](#_Toc115105854)

[7 Δοκιμή Συστήματος 23](#_Toc115105855)

[8 Οδηγίες Χρήσης του Συστήματος 24](#_Toc115105856)

[8.1 Swagger 24](#_Toc115105857)

[8.2 Postman 29](#_Toc115105858)

[8.3 Database Update/Login Form 34](#_Toc115105859)

[8.4 Πώς τρέχουμε τα projects? 36](#_Toc115105860)

[8.5 Dependencies 37](#_Toc115105861)

[9 Επίλογος 39](#_Toc115105862)

[10 Πηγές 41](#_Toc115105863)

# Εισαγωγή

Στα πλαίσια της διπλωματικής εργασίας στο Τμήμα Επιστήμης Υπολογιστών του Πανεπιστήμιο Κρήτης σχεδιάστηκε και αναπτύχθηκε ένα REST API (web service) για την ανοικτή διάθεση κυρίως ιστορικών δεδομένων σχετικά με την καταγεγραμμένη σεισμική δραστηριότητα στην Ελλάδα (1964-σήμερα). Τα δεδομένα αυτά είναι διαθέσιμα από το Εθνικό Ατεροσκοπείο Αθηνών (https://www.gein.noa.gr/en/services-products/earthquake-catalogs/ ) και είναι ήδη διαθέσιμα σε ανοικτό μηχαναγνώσιμο μορφότυπο, αλλά όχι τόσο ευανάγνωστο για τον απλό χρήστη. Παρακάτω παρουσιάζονται σκοπός ανάπτυξης του συγκεκριμένου συστήματος μαζί με τα χαρακτηριστικά, τις απαιτήσεις, περιορισμούς, τις διεπαφές του συστήματος καθώς και τις λειτουργίες από τις οποίες απαρτίζεται.

## Γενική περιγραφή

Στα πλαίσια της πτυχιακής εργασίας υλοποιήθηκαν υπηρεσίες που στόχευσαν:

**α)** ανάπτυξη ενός REST API για την ανοικτή τους διάθεσή τους σε διαφορετικές/εναλλακτικές μορφές: HTML, XML, JSON

**β)** στην εύκολη εισαγωγή, διαμόρφωση, παραμετροποίηση και ενημέρωση του REST API με ανοικτά δεδομένα και γενικότερα την διαχείριση και επεκτασιμότητα του συστήματος.

**γ)** στην ανάπτυξη μιας web-based client εφαρμογής που θα χρησιμοποιεί το παραπάνω REST API και θα παρέχει υπηρεσίες αναζήτησης/φιλτραρίσματος και οπτικοποίησης των δεδομένων (π.χ. γραφήματα ανά έτος, γεωγραφική διασπορά σε χάρτη κλπ)

Η εφαρμογή έχει αναπτυχθεί ως αυτόνομη υπηρεσία (εγκατάσταση σε ανεξάρτητο μηχάνημα) και η διαχείριση/αποθήκευση των δεδομένων έγινε σε μια ΒΔ (εισαγωγή/ενημέρωση/διαγραφή).

## Σκοπός συστήματος

Το λογισμικό του συστήματος που θα αναπτυχθεί θα εξυπηρετήσει τις υπηρεσίες αναζήτησης και απεικόνισης πληροφοριών. Στόχοι είναι οι χρήστες να μπορούν να κατανοήσουν εύκολα και με ελάχιστη προσπάθεια τα οπτικοποιημένα δεδομένα που υπάρχουν σε αρχεία csv, που προσφέρονται απο το Εθνικό Αστεροσκοπείο Αθηνών - τα οποία επίσης επεξεργάζονται κατάλληλα – χωρίς να μπούνε στον κόπο να τα κατεβάσουνε και να τα ανοίξουνε. Τα οπτικοποιημένα δεδομένα παρουσιάζονται εν τέλη σε ένα χάρτη με τις κατάλληλες ενδείξεις, αλλά επίσης και σε γραφικές παραστάσεις. Ο όγκος δεδομένων που χρησιμοποιείται (1964-2021) χωρίς ενημέρωση της βάσης δεδομένων είναι 322,693 εγγραφές (rows) στην βάση δεδομένων. Τα αρχεία που είναι διαθέσιμα στο Εθνικό Ατεροσκοπείο Αθηνών (<https://www.gein.noa.gr/en/services-products/earthquake-catalogs/> ) είναι σε μορφή csv, αλλά δεν περιέχουν delimiter characters ( Διαχωριστικούς χαρακτήρες π.χ “,” ”|”,κ.τ.λ.) που είναι απαραίτητοι για να εισαχθούν τέτοια αρχεία στη βάση δεδομένων, άρα το σύστημα τα επεξεργάζεται αναλόγως και στη συνέχεια τα εισάγει στη βάση δεδομένων. Η ενημέρωση της βάσης δεδομένων θα μπορεί να γίνει εύκολα από κάποιον admin με τα κατάλληλα προνόμια, κατεβάζοντας το αρχείο και προσθέτοντάς το με τη χρήση του UI. Μέσω την χρήση του swagger και του postman θα μπορούν να προβάλλονται οι λειτουργίες του REST. Με τη χρήση του Postman ο χρήστης μπορεί να πάρει αποτέλεσμα μιας μορφής δεδομένων που επιλέγει (html, xml, json), αλλά και επειδή αρκετά queries έχουν τεράστια αποτελέσματα, που είναι αναμενόμενο με δεδομένα αυτού του τεράστιου μεγέθους, λόγο του ότι το swagger χρησιμοποιείται σε browser, δεν είναι αρκετα Optimized για rendering δεδομένων μεγάλης κλίμακας. Αυτά τα queries θα έχουν προειδοποιήσεις.

# Σενάρια Χρήσης και Τύποι Χρηστών

## Περιγραφή

Τα σενάρια χρήσης είναι αρκετά απλά, αφού τα δεδομένα που έχουμε αφορούν σεισμούς. Ο κάθε καταγεγραμμένος σεισμός περιέχει 6 χαρακτηριστικά:

1. Date (YYYY/MM/DD)
2. Time
3. Latitude
4. Longitude
5. Depth
6. Magnitude

Άρα ο χρήστης θα αναζητά σεισμούς (ή έναν συγκεκριμένο σεισμό) με βάση τουλάχιστον ενός απο τα παραπάνω χαρακτηριστικά. Συνολικά υπάρχουν 19 σενάρια χρήσης του συστήματος εκ των οποίων το 1 είναι κρυμμένο αφού είναι αυτό που επιστρέφει τα πάντα (όλες τις καταγραφές στη βάση δεδομένων) που είναι ένα τεράστιο response body. Στο Postman, το συγκεκριμένο query μπορεί να προστεθεί manually σαν get method ή απλά να τρέξει το συγκεκριμένο get command ο χρήστης και θα πάρει το response body κανονικά σαν τα υπόλοιπα queries. Για τους τύπους χρήστη θα πρέπει να ξεκαθαρίσουμε τι είναι ένα REST API.

REST

Το REST ουσιαστικά είναι ένα σύνολο αρχιτεκτονικών περιορισμών.

Όταν ένα αίτημα πελάτη γίνεται μέσω ενός RESTful API, μεταφέρει μια αναπαράσταση της κατάστασης του πόρου στον αιτούντα ή στο τελικό σημείο. Αυτές οι πληροφορίες, ή αναπαράσταση, παραδίδονται σε μία από τις διάφορες μορφές. Το JSON είναι η πιο δημοφιλής μορφή αρχείου που χρησιμοποιείται γενικά, επειδή, παρά το όνομά του, είναι αγνωστική ως προς τη γλώσσα, καθώς και ευανάγνωστη τόσο από ανθρώπους όσο και από μηχανήματα.

API

Ένα API είναι ένα σύνολο ορισμών και πρωτοκόλλων για τη δημιουργία και την ενοποίηση λογισμικού εφαρμογών. Άν θέλουμε να αλληλεπιδράσουμε με έναν υπολογιστή ή ένα σύστημα για να ανακτήσουμε πληροφορίες ή να εκτελέσουμε μια λειτουργία, ένα API μάς βοηθά να επικοινωνήσουμε αυτό που θέλουμε σε αυτό το σύστημα, ώστε να μπορεί να κατανοήσει και να εκπληρώσει το αίτημα.

Είναι επίσης ένας τρόπος για έναν οργανισμό να μοιράζεται πόρους και πληροφορίες διατηρώντας παράλληλα την ασφάλεια, τον έλεγχο και τον έλεγχο ταυτότητας—καθορίζοντας ποιος έχει πρόσβαση σε τι.

Ένα άλλο πλεονέκτημα ενός API είναι ότι δεν χρειάζεται να γνωρίζουμε τις ιδιαιτερότητες της προσωρινής αποθήκευσης.

REST API

Το REST API (γνωστό και ως RESTful API) είναι μια διεπαφή προγραμματισμού εφαρμογών (API ή web API) που συμμορφώνεται με τους περιορισμούς του αρχιτεκτονικού στυλ REST και επιτρέπει την αλληλεπίδραση με τις υπηρεσίες ενός RESTful web.

RESTful Web Services

Οι υπηρεσίες Ιστού RESTful είναι βασικά Υπηρεσίες Ιστού που βασίζονται στην αρχιτεκτονική REST. Στο REST Architecture τα πάντα είναι ένας πόρος. Οι υπηρεσίες ιστού RESTful είναι μικρού βάρους, εξαιρετικά επεκτάσιμες και διατηρούμενες και χρησιμοποιούνται πολύ συχνά για τη δημιουργία API για εφαρμογές που είναι web-based.

## Σενάρια Χρήσης

1. /Earthquakes/Year?year=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που καταγράφηκαν μια συφκεκριμένη χρονιά).
2. /Earthquakes/YearMonth?year=...&month=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που καταγράφηκαν σε έναν συγκεκριμένο μήνα ενός συγκεκριμένου χρόνου ).
3. Earthquakes/YearMagnitude?year=... &magnitude=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που καταγράφηκαν σε έναν συγκεκριμένο χρόνο και έχουνε συγκεκριμένο magnitude ή εύρος (<1) (π.χ magnitude=5 -> επιστρέφει όλους τους σεισμούς με μεγέθη απο 5.00-5.99)).
4. /Earthquakes/YearCity?year=minim dolore&latitude=... &longitude=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που καταγράφηκαν σε μια πόλη της Ελλάδας για ένα συγκεκριμένο έτος).
5. /Earthquakes/YearCityMagnitude?year=...&latitude=...&longitude=...&magnitude=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που καταγράφηκαν σε μια πόλη της Ελλάδας για ένα συγκεκριμένο έτος και συγκεκριμένο μέγεθος σεισμού ή συγκεκριμένο εύρος μεγέθου σεισμών).
6. /Earthquakes/SingleEQ?date=...&time=...&latitude=...&longitude=...&depth=94066729&magnitude=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για έναν συγκεκριμένο σεισμό).
7. /Earthquakes/Month?month=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για όλους τους σεισμούς που καταγράφηκαν έναν συγκεκριμένο μήνα. Για όλα τα έτη).
8. /Earthquakes/Magnitude?magnitude=… ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που έχουν συγκεκριμένο μέγεθος ή συγκεκριμένο εύρος μεγέθου (π.χ magnitude=5 -> επιστρέφει όλους τους σεισμούς με μεγέθη απο 5.00-5.99)).
9. /Earthquakes/Depth?depth=… ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που έχουν συγκεκριμένο βάθος).
10. /Earthquakes/DepthMagnitude?depth=...&magnitude=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που έχουν συγκεκριμένο βάθος και μέγεθος).
11. /Earthquakes/Date?date=... =... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που καταγράφηκαν σε μια συγκεκριμένη ημερομηνία).
12. /Earthquakes/DateTime?date=... &time=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που καταγράφηκαν σε μια συγκεκριμένη ημερομηνία και ώρα).
13. /Earthquakes/DateMagnitude?date=...&magnitude=… =... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που καταγράφηκαν σε μια συγκεκριμένη ημερομηνία και έχουν ένα συγκεκριμένο μέγεθος).
14. /Earthquakes/DateDepth?date=...&depth=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που καταγράφηκαν σε μια συγκεκριμένη ημερομηνία και έχουν ένα συγκεκριμένο βάθος).
15. /Earthquakes/DateCoordsDepthMagnitude?date=...&latitude=...&longitude=...&depth=...&magnitude=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που καταγράφηκαν σε μια συγκεκριμένη ημερομηνία και έχουν ένα συγκεκριμένες συντεταγμένες, βάθος και μέγεθος).
16. /Earthquakes/CoordsMagnitude?latitude=...&longitude=...&magnitude=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς με συγκεκριμένες συντεταγμένες και μέγεθος).
17. /Earthquakes/Coordinates?latitude=...&longitude=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για όλους τους σεισμούς που καταγράφηκαν στις συγκεκριμένες συντεταγμένες).
18. /Earthquakes/CityMagnitude?latitude=...&longitude=...&magnitude=... ( Ο χρήστης επιλέγει να εμφανιστούν όλες οι πληροφορίες για τους σεισμούς που καταγράφηκαν σε μια συγκεκριμένη πόλη της Ελλάδας( μικρό εύρος συντεταγμένων) και που έχουν ένα μικρό εύρος ως προς το μέγεθός τους).
19. (Κρυμμένο QUERY) /Earthquakes/EVERYTHING (Επιστρέφει όλους τους εγγεγραμμένους σεισμούς που βρίσκονται μέσα στη βάση δεδομένων).

## Τύποι χρήστη

Οι χρήστες που θα χρησιμοποιήσουν το σύστημα μπορεί να είναι είτε κανονικοί χρήστες με μηδενική γνώση για το τί είναι REST ή REST API, μπορεί να είναι developers που να έχουν ως πρόθεση να εκμεταλλευτούν την χρήση του REST API για ανάκτηση δεδομένων, ή μπορεί να είναι admin, άτομο με συγκεκριμένα προνόμια που είναι υπεύθυνος για διαχείριση, επιτήρηση, ενημέρωση και διαγραφή δεδομένων από τη βάση δεδομένων.

* **Περιστασιακός χρήστης:**  Χρησιμοποιεί την κύρια σελίδα του web application για να υποβάλει δεδομένα τα οποία τον ενδιαφέρουν και παρουσιάζονται οπτικοποιημένα μπροστά του με τη βοήθεια ενός χάρτη με τις κατάλληλες ενδείξεις (markers) διαφορετικών χρωμάτων. Δέν έχει απαραίτητα γνώσεις ενός developer. Επίσης για ακόμα πιο λεπτομερείς ένδειξη των δεδομένων υπάρχει και μία γραφική παράσταση pie chart, αλλά και όλες οι πληροφορίες καταγραμμένες.
* **Developer:** Χρήστης που με τη βοήθεια του documentation θα μπορέσει να χρησιμοποιήσει όλες τις λειτουργίες του RESTful API. Στο συγκεκριμένο REST API θα χρησιμοποιήσει τη βοήθεια του Swagger, αλλά και του Postman. Το Swagger προσφέρει πιο ξεκάθαρη περιγραφή της εφαρμογής, ενώ το Postman βελτιστοποιημένα responses και επιλογές για το response body (xml, html, json, text).
* **Admin:** Χρήστης με προνόμια που μπορεί να έχει την δυνατότητα να ενημερώσει ή να διαγράψει δεδομένα από τη βάση δεδομένων. Χρησιμοποιεί το καταλληλο user interface αφού κάνει login για να επιβεβαιώσει τα προνόμια που έχει και στη συνέχεια ανεβάζει ένα αρχείο για να γίνει ενημέρωση στη βάση δεδομένων. Αν το αρχείο που θέλει να ανεβάσει είναι της τωρινής χρονίας, και η χρονιά δεν έχει τελειώσει, τότε στο μέλλον όταν θα ξανα ενημερώσει την βάση, θα πρέπει να διαγράψει τα δεδομένα της προηγούμενης φοράς πρίν ανεβάσει το νέο αρχείο. Αυτό γίνεται πολύ εύκολα με τη χρήση του user interface επίσης. Πιο γενικά, ο admin έχει πρόσβαση και έλεγχο ολόκληρου του συστήματος και της βάσης δεδομένων.

# Απαιτήσεις

Απαιτήσεις που έχουμε είναι κατ’αρχάς το σύστημα να είναι διαθέσιμο πάντα για να μην υπάρχουν ασυνέπειες. Λόγω του μεγάλου όγκου δεδομένων που χρησιμοποιεί το σύστημα πρέπει να είναι ξεκάθαρο για τους developers πότε μπορούνε να κάνουνε χρήση του Swagger χωρίς πρόβλημα rendering απο το browser, αν και κατα προτίμηση να χρησιμοποιείται το Swagger παραπάνω σαν documentation και τα requests να τρέχουν στο Postman. Πρέπει να είναι εφικτή η εγκατάσταση του συστήματος σε άλλα servers. Επίσης πρέπει να γίνεται συχνή ενημέρωση δεδομένων, αφού τα δεδομένα στο Εθνικό Αστεροσκοπείο Αθηνών ενημερώνονται καθημερινά. Ανάλογα από την κίνηση με την οποία οι χρήστες χρησιμοποιούν το σύστημα, συστήνεται η συχνή συντήρηση και ενημέρωση του συστήματος.

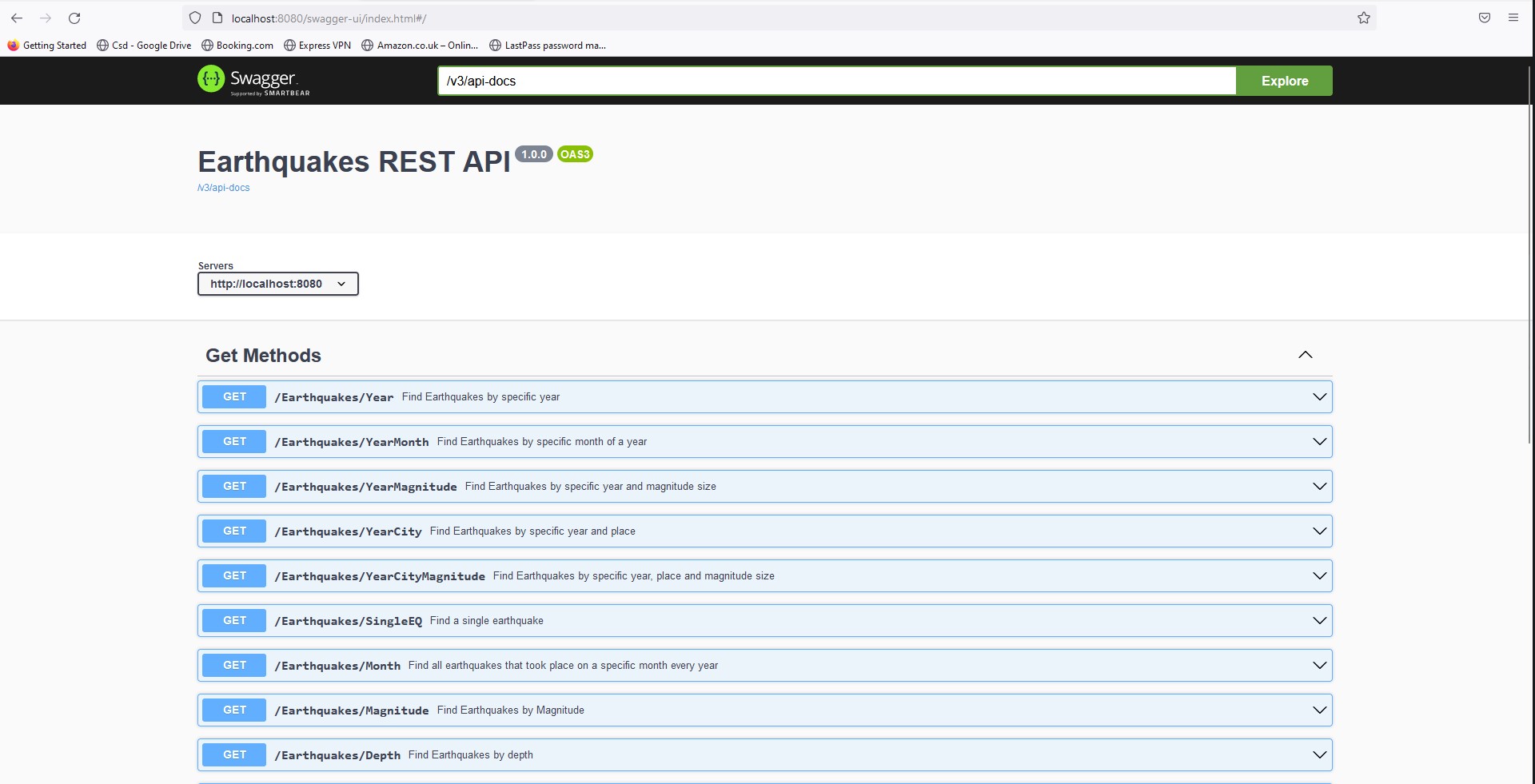
Για την ασφάλεια του συστήματος, πρόσβαση και διαχείριση βάσης δεδομένων πρέπει να έχει μόνο κάποιος admin για διατήρηση ακεραιότητας της βάσης και όλων των δεδομένων της. Επίσης, σχετικά με την ενημέρωση της βάσης, λόγω του ότι το αρχείο στην ιστοσελίδα του Αστεροσκοπείου ενημερώνεται καθημερινά, το URL αλλάζει και άρα ο υπεύθυνος πρέπει να κατεβάσει το αρχείο χειροκίνητα. Η αποθήκευση των κατεβασμένων αρχείων και των καινούργιων που δημιουργούνται μετά απο την επεξεργασία τους πρέπει να αποθηκεύονται μέσα στον server, έτσι ώστε να μην χρειάζεται να γράψουμε hard code για local storing. Έτσι τα αρχεία δεν θα είναι προσβάσιμα εύκολα για τους πάντες. Το σύστημα θα πρέπει να μπορεί να χρησιμοποιηθεί εύκολα απο τους developers αφού διαβάσουνε το documentation για κάθε λειτουργία του, άρα η περιγραφή των λειτουργιών θα πρέπει να είναι αρκετά ξεκάθαρη.

# Οθόνες της Εφαρμογής

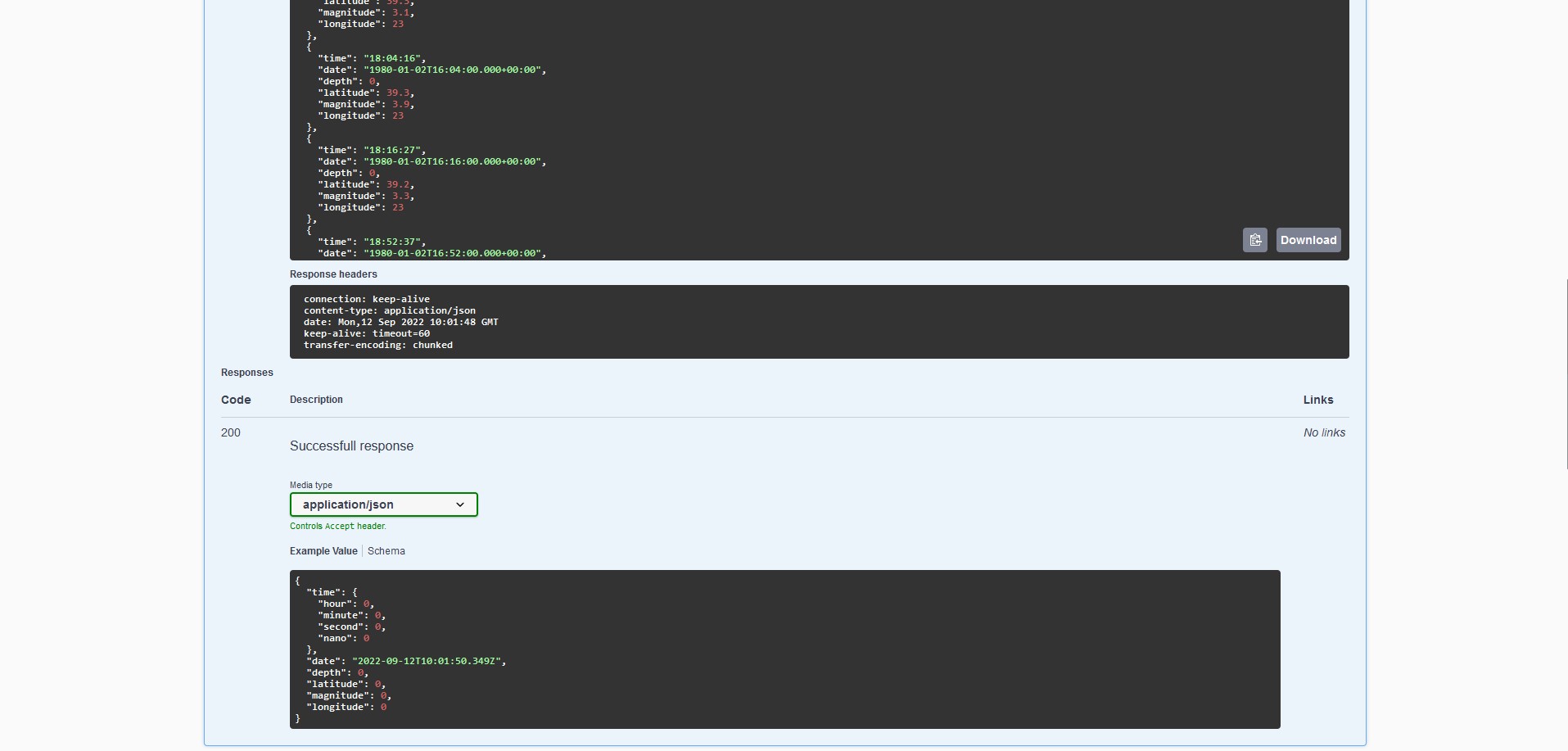
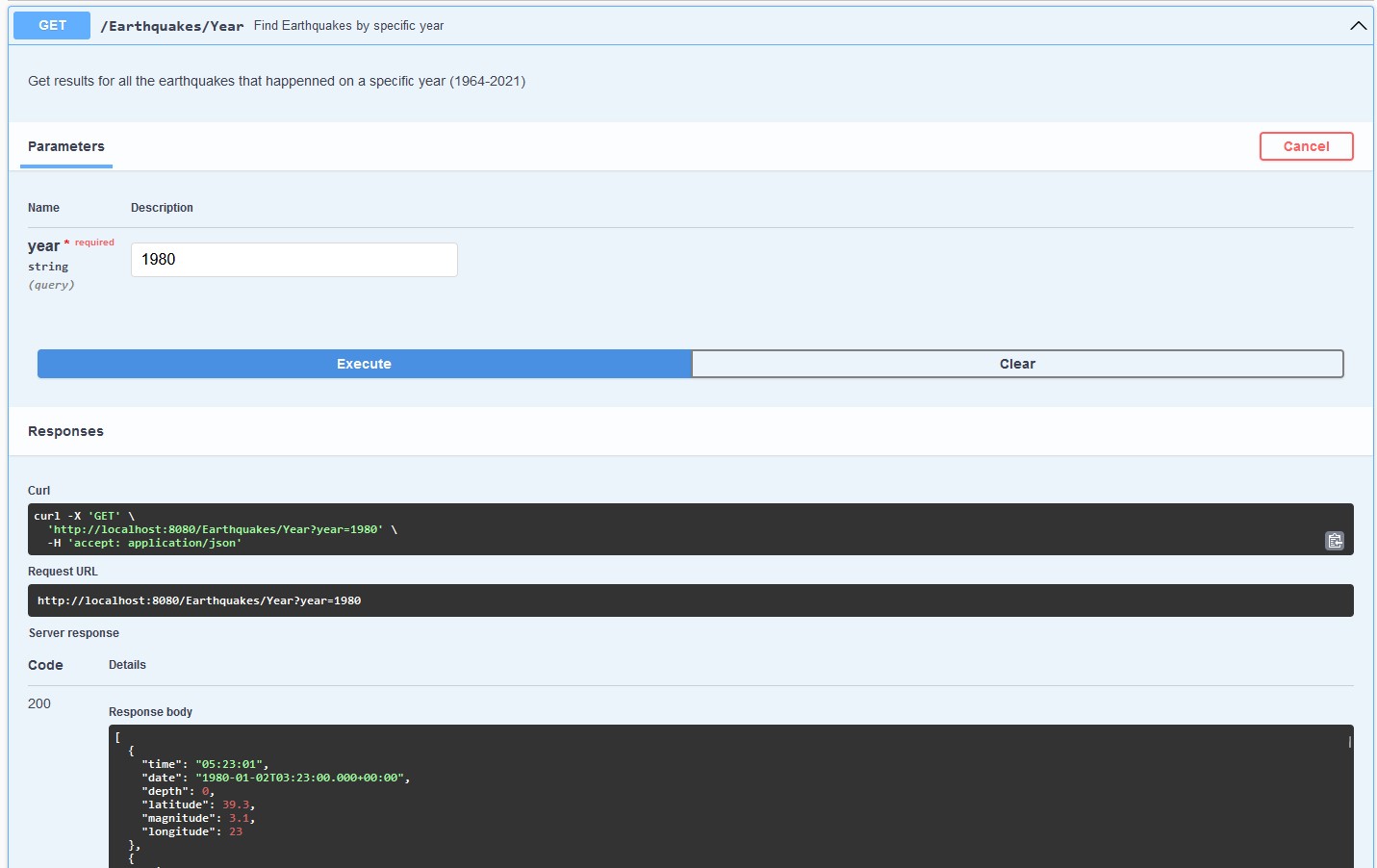
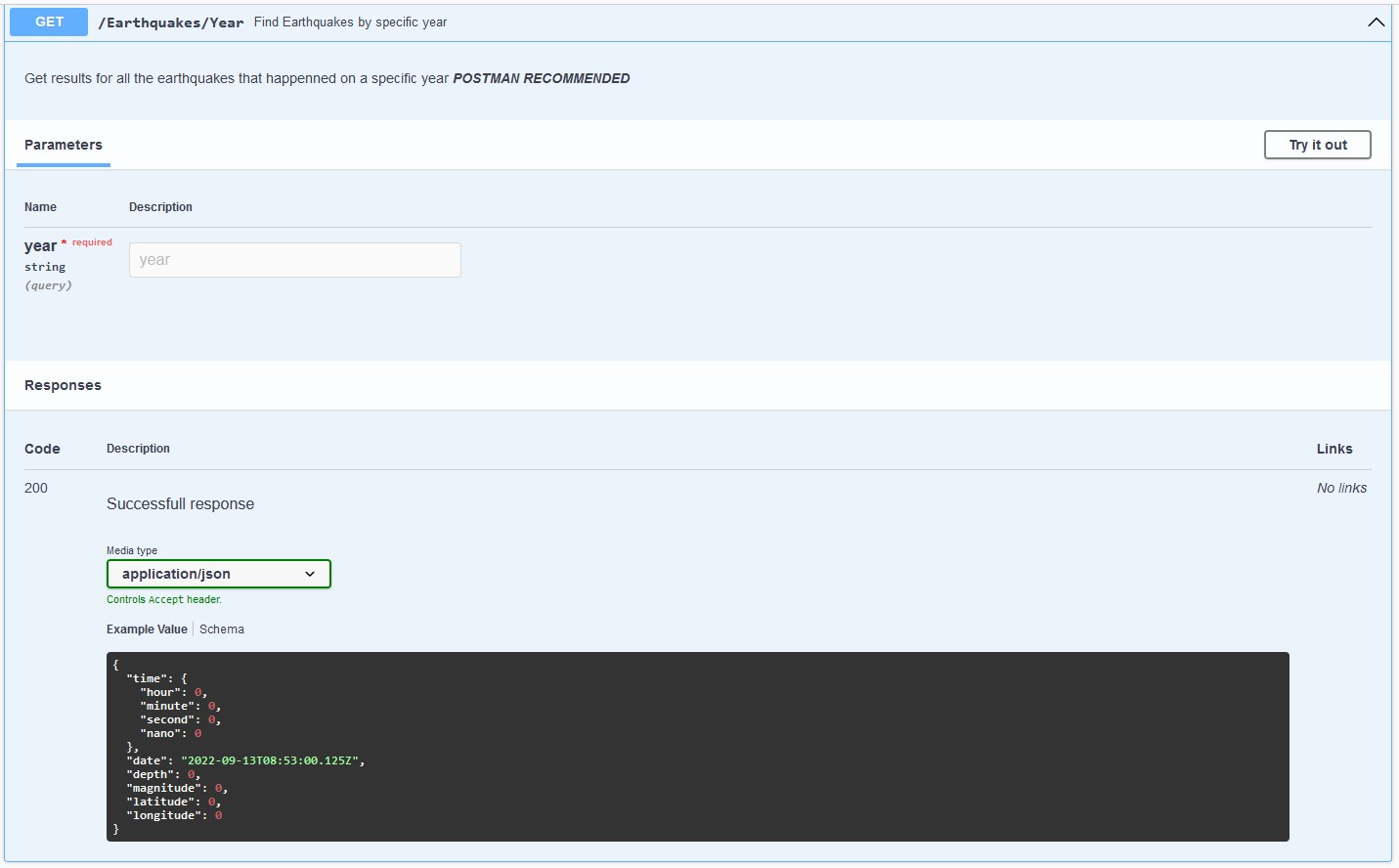
Οι κύριες οθόνες της εφαρμογής μας είναι το Swagger, το Postman και η κύρια σελίδα η οποία είναι εύκολη να την χρησιμοποιήσει οποιοσδήποτε χρήστης.

## Swagger

Το Swagger είναι ένα εργαλείο που μας βοηθά να δημιουργήσουμε, να περιγράψουμε τη δομή του API μας, αλλά και να δοκιμάσουμε τις RESTful web services του. Μας βοηθάει επίσης στο να δημιουργήσουμε ένα διαδραστικό documentation για το API. Στην σελίδα του Swagger έχουμε πρόσβαση πηγαίνοντας στο link τύπου {server name}/swagger-ui/index.html ή απο πατώντας τον σύνδεσμο απο την κύρια σελίδα του συστήματος. Η σελίδα του Swagger είναι η παρακάτω και τώρα μπορεί ο χρήστης να δεί όλες τις λειτουργίες του συστήματος.

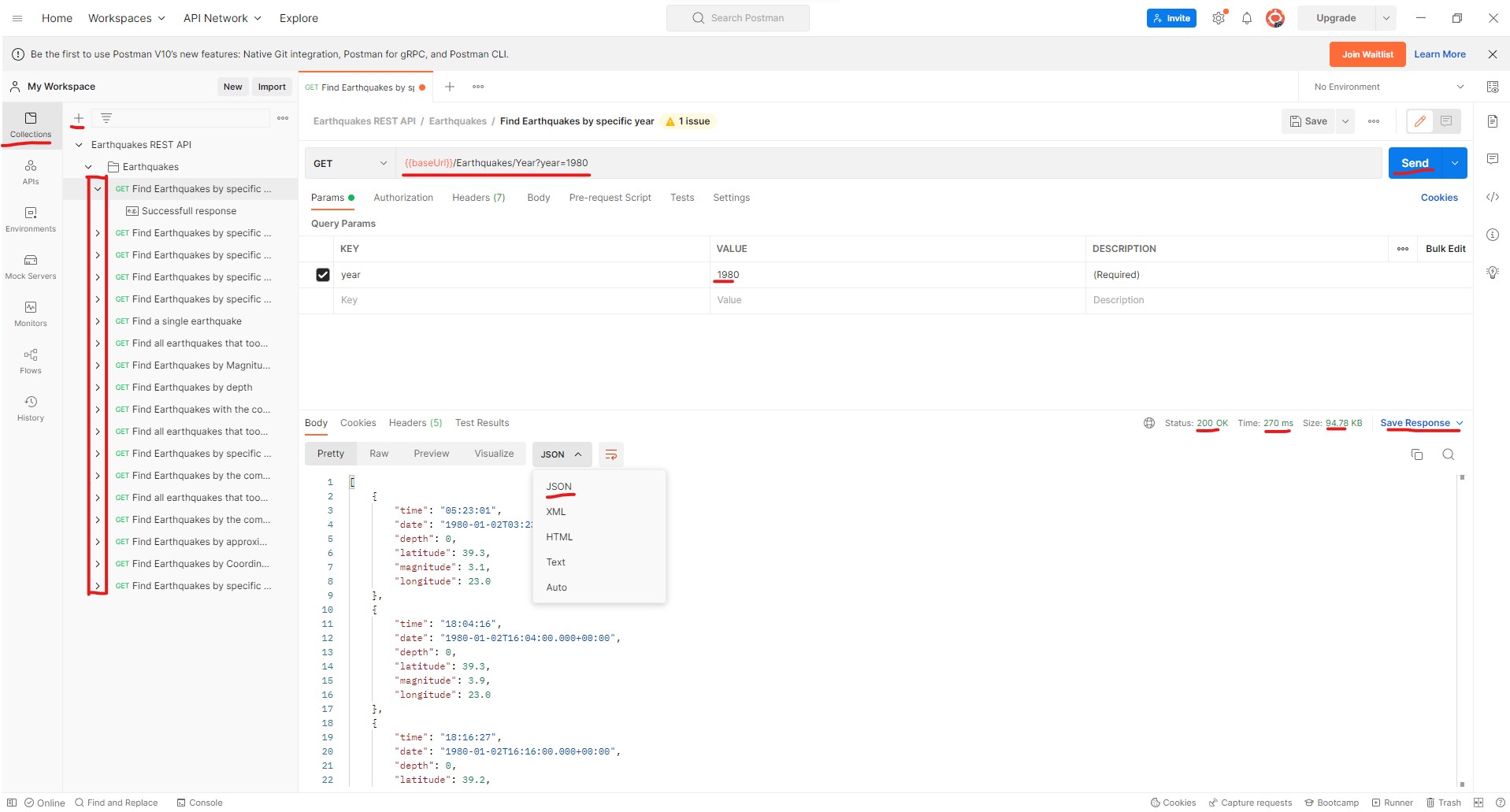


Με ένα κλικ πάνω στην κάθε λειτουργία μπορούμε να δούμε λεπτομέρειες για το πώς ακριβώς μπορούμε να εισάγουμε τιμές στα κατάλληλα πεδία, αλλα επίσης βλέπουμε και την ολόκληρη την περιγραφή της λειτουργίας που είναι σημαντικό αφού εδώ είναι που περιέχεται η σύσταση για το αν θα χρειαστεί το Postman για την λειτουργία ή αν είμαστε οκ να την δοκιμάσουμε κανονικά. Πιο κάτω επίσης φαίνεται ένα παράδειγμα του response body σε περίπτωση επιτυχής επιστροφής αντικειμένων. Μπορούμε δοκιμάσουμε την λειτουργία κάνοντας κλικ στο κουμπί “Try it out” στα δεξιά και τώρα μας αφήνει να εισάγουμε τιμές στα πεδία που υπάρχουν για την κάθε λειτουργία.



## Postman

Το Postman είναι μια πλατφόρμα API για δημιουργία και χρήση API. Το Postman απλοποιεί κάθε βήμα του κύκλου ζωής του API και βελτιστοποιεί τη συνεργασία, ώστε να μπορούμε να δημιουργήσουμε καλύτερα API—πιο γρήγορα. Αφού εγκαταστήσουμε την εφαρμογή στον υπολογιστή μας και προσθέσουμε το API μας μέσα, μπορούμε εύκολα να χρησιμοποιήσουμε όλες τις λειτουργίες του, ακόμα και τις πιο χρονοβόρες, σε πολύ πιο βέλτιστο χρόνο απο το Swagger. Επίσης μπορούμε να διαλέξουμε το response body type και να κατεβασουμε στον υπολογιστή μας ολόκληρο το response.

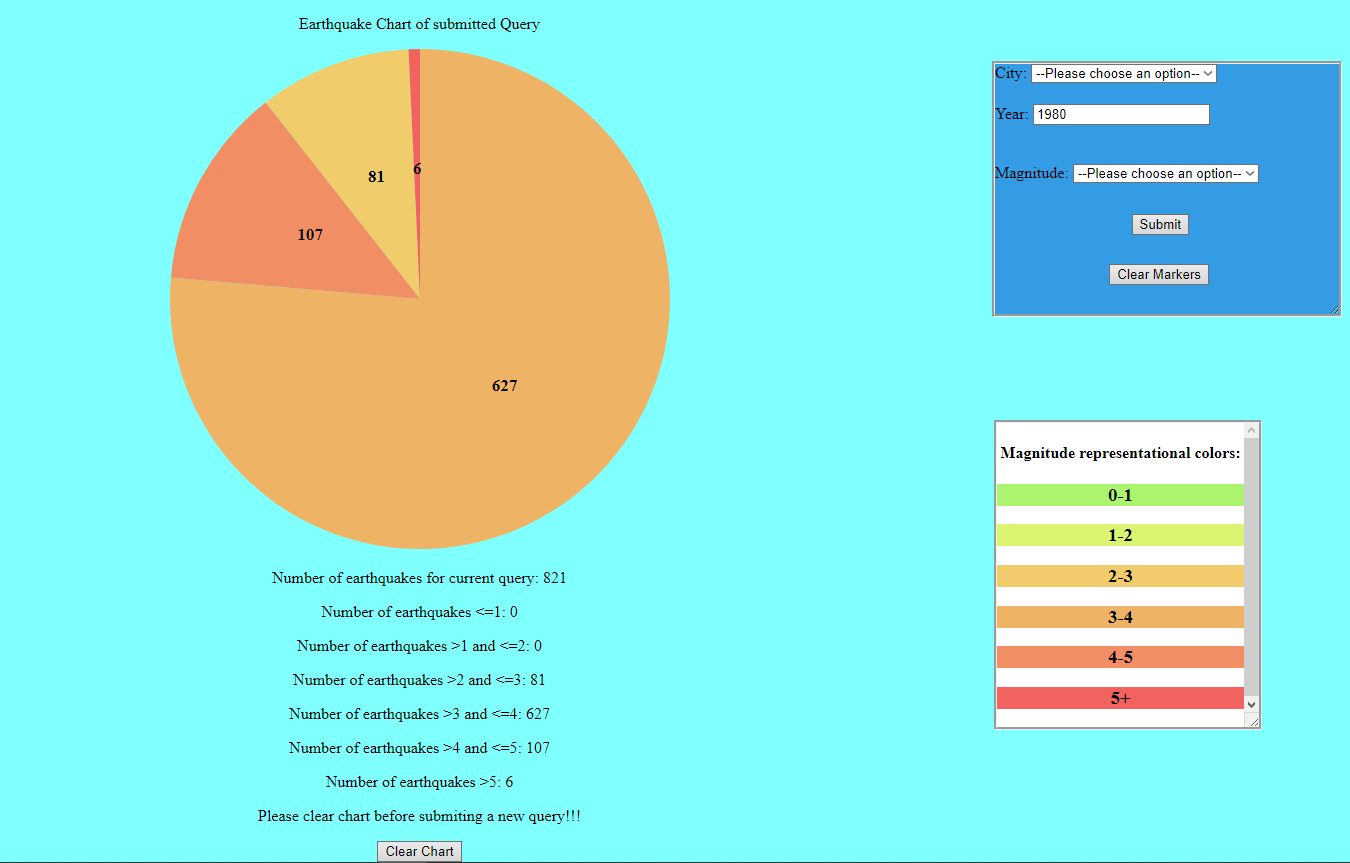


## Main Page (index.html)

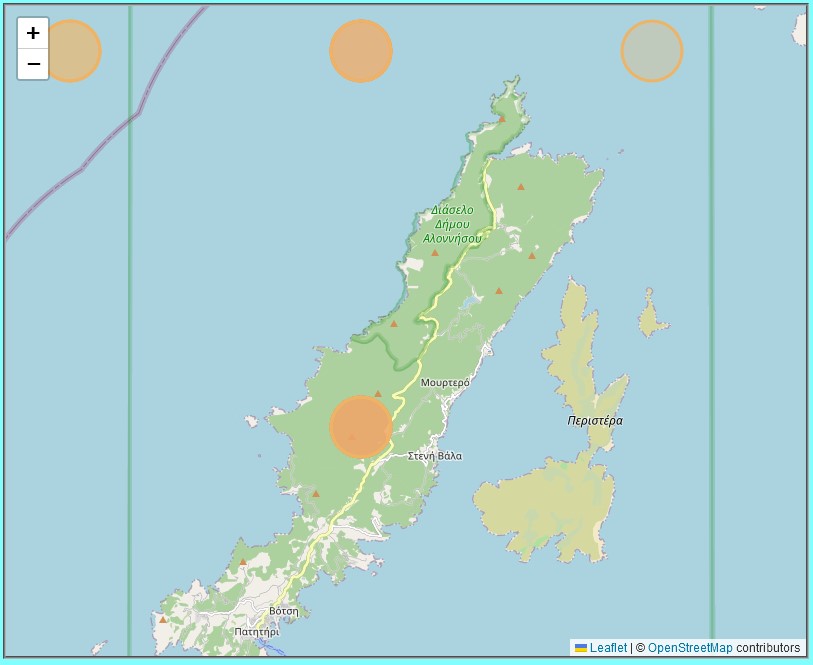
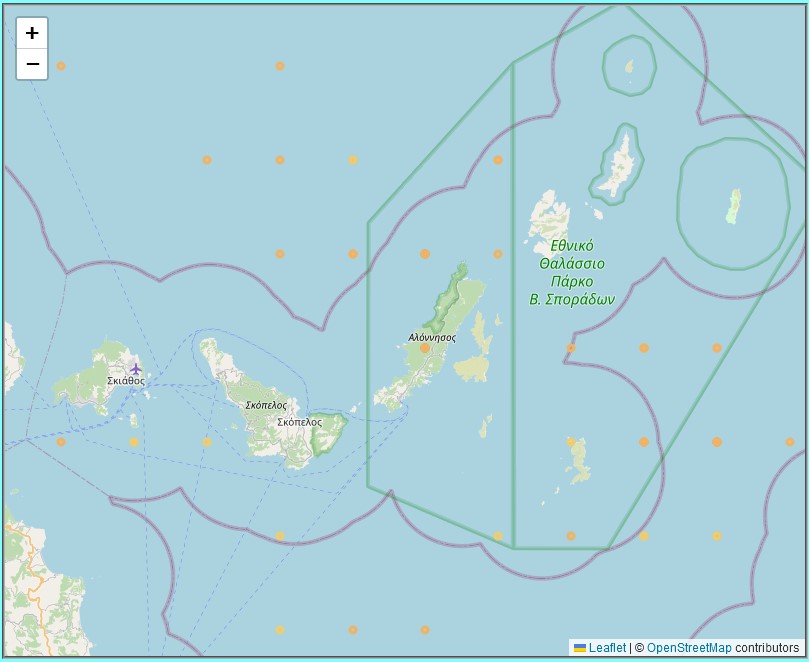
Η κύρια σελίδα μας περιέχει τον τίτλο της εφαρμογής μας, ενα σύνδεσμο για το Swagger και έναν χάρτη αρχικοποιημένο στην περιοχή που βρίσκεται η Ελλάδα. Στα δεξιά βρίσκονται μέσα στο μπλε κουτάκι οι 3 επιλογές που ο χρήστης μπορεί να συνδυάσει για να κάνει τα queries που τον ενδιαφέρουν. Οι επιλογές που έχει για να εμφανιστούν ενδείξεις στον χάρτη είναι να συμπληρώσει τουλάχιστον ένα από τις 3 επιλογές ( City, Year, Magnitude ) και να πατήσει το κουμπί “Submit”. Κάτω από αυτό το κουτί βλέπουμε τα χρώματα που εκπροσωπούν την κάθε ένδειξη που εμφανίζεται στον χάρτη.



Υποβάλλοντας τα στοιχεία που επιθυμεί ο χρήστης προκαλεί τα διάφορα marker να εμφανιστούν στον χάρτη. Επίσης θα δημιουργηθεί το Pie Chart κάτω από τον χάρτη με λεπτομέρειες και κάτω από το γράφημα οι λεπτομέρειες θα φαίνονται καθαρογραμμένες. Ο χρήστης μπορεί να ‘καθαρίσει’ τα markers απο τον χάρτη και το Pie Chart από κάτω εάν το επιθυμεί (βλ. Κουμπιά ‘Clear Markers’, ‘Clear Chart’).



Κάνοντας ζούμ στον χάρτη βλέπουμε πως όσο και να είναι το πλήθος των marker - που αρκετές φορές είναι μεγάλο – γίνεται render καλά και γρήγορα. Επίσης αν κάνουμε αρκετό ζούμ το μέγεθος των marker μεγαλώνει για να φαίνονται ξεκάθαρα.

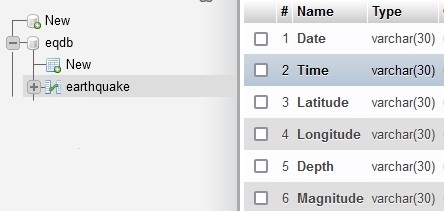


# Σχεδίαση και Μοντελοποίηση

## Σχεδιασμός Βάσης Δεδομένων

Η βάση δεδομένων δημιουργείται αρχικά παίρνοντας τα αρχεία καταλόγων που υπάρχουν στην ιστοσελίδα του Εθνικού Αστεροσκοπείου Αθηνών και στη συνέχεια επεξεργάζοντάς τα κατάλληλα προσθέτουμε τα δεδομένα κάθε καταλόγου σε ένα table. Το table μας έχει 6 στήλες οι οποίες αντιπροσωπεύουν τα χαρακτηριστικά που έχει ο κάθε σεισμός. Τα χαρακτηριστικά αυτά είναι:

1. Date (YYYY/MM/DD)
2. Time
3. Latitude
4. Longitude
5. Depth
6. Magnitude



Η βάση δεδομένων δημιουργήθηκε με την εντολή: “ CREATE DATABASE eqdb; “.

Το table(πίνακας) για τους σεισμούς δημιουργήθηκε με την εντολή:

“ CREATE TABLE earthquake(Date VARCHAR(30), Time VARCHAR(30), Latitude VARCHAR(30), Longitude VARCHAR(30), Depth VARCHAR(30), Magnitude VARCHAR(30)); ”

Σημείωση: Μετατρέπονται στους κατάλληλους τύπους όταν χρησιμοποιούμε Java Objects πιο μετά.

Τα πρώτα αρχεία που εισάγονται στη βάση είναι οι κατάλογοι απο το 1964 μέχρι και του 2021. Για αυτά τα αρχεία έχουμε προνοήσει να τα κατεβάζουμε αυτόματα σε ένα directory που δημιουργήσαμε μέσα στον server (config folder) και στη συνέχεια επεξεργαζόμαστε το καθένα για την κατάλληλη εισαγωγή των δεδομένων τους στη βάση δεδομένων. Για την εισαγωγή και επέκταση δεδομένων στη βάση μέσω του αρχείου αρκεί μετά την επιλογή του χρήστη μέσω του UI και την επεξεργασία του, να εκτελείται η εντολή:

insert into earthquake(Date,Time,Latitude,Longitude,Depth,Magnitude)values(?,?,?,?,?,?);

Το αρχείο που δίνει ο χρήστης πρώτα πρέπει να επεξεργαστεί επειδή δεν υπάρχουν διαχωριστικοί χαρακτήρες, γι’αυτό το επεξεργαζόμαστε και παράγουμε ένα νέο αρχείο το οποίο αποθηκεύεται σε ένα directory του server μας.

Η περίπτωση που θέλουμε να διαγράψουμε δεδομένα απο τη βάση, είναι η περίπτωση που έχουμε ενημερώσει τη βάση με μη συμπληρωμένο αρχείο (δηλαδή αρχείο που δεν συμπεριλαμβάνει όλους τους σεισμούς ολόκληρης της χρονιάς) και θέλουμε να την ξανα ενημερώσουμε με πιο πρόσφατο αρχείο που περιέχει και παλιά αλλά και νέα δεδομένα. Θέλουμε δηλαδή να διαγράψουμε όλα τα δεδομένα που υπάρχουν για το έτος που θέλουμε να ενημερώσουμε και στη συνέχεια να κάνουμε την ενημέρωση. Έτσι αποφεύγουμε το να υπάρχουν διπλά δεδομένα στη βάση. Για να διαγράψουμε λοιπόν τα δεδομένα, ο χρήστης αρχικά, μέσω του UI γράφει τον αριθμό του έτους που θέλει να ενημερώσει/διαγράψει και το υποβάλλει. Με αυτή την πληροφορία μπορούμε να εκτελέσουμε την εντολή:

DELETE FROM earthquake WHERE Date LIKE '%"+s+"%';

Όπου s η χρονολογία που υπέβαλε ο χρήστης.

## Σχεδίαση των project

Το σύστημα είναι χωρισμένο σε 3 ‘projects’.

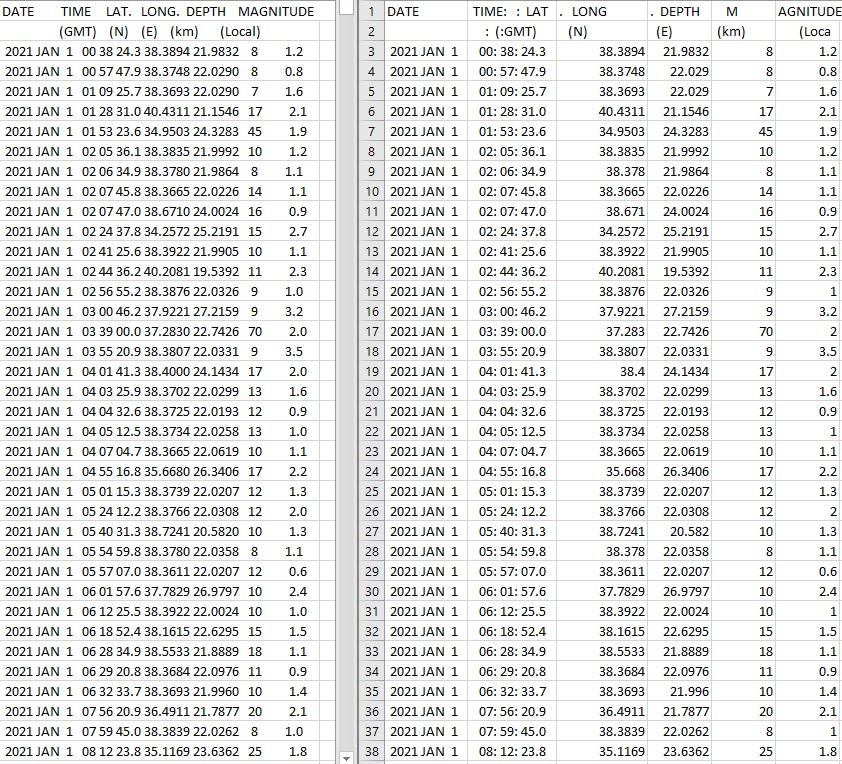
* + - 1. Το πρώτο είναι αυτό που κατεβάζει και επεξεργάζεται τα αρχεία που περιέχουν τα δεδομένα που αμέσως μετά θα εισαχθούν στη βάση δεδομένων. Αυτό το project απλά τρέχει μόνο την πρώτη φορά και σταματά.
      2. Το δεύτερο είναι το κύριο κομμάτι του συστήματος. Δηλαδή είναι το κομμάτι στο οποίο δημιουργείται το REST API, το Swagger και η κύρια σελίδα. Αυτό το project πρέπει να τρέχει συνέχεια έτσι ώστε να υπάρχει συνεχής διαθεσιμότητα.
      3. Το τρίτο είναι υπεύθυνο για το Login του admin και την ενημέρωση της βάσης δεδομένων. Αυτό το project θα πρέπει να τρέχει για όσο επιθυμεί ο admin, αφού μπορεί να θέλει να ενημερώσει πολλαπλές φορές τη βάση δεδομένων.

## Μοντελοποίηση

### Επεξεργασία αρχείων

Θέλουμε να αλλάξουμε τον τρόπο που υπάρχουν τα δεδομένα μέσα στα αρχεία επειδή δεν είναι χωρισμένα με διαχωριστικούς χαρακτήρες και άρα εκτός του ότι δεν είναι τόσο ευανάγνωστα, δεν μπορούν να καταχωρηθούν στην βάση δεδομένων όπως πρέπει. Βλέπουμε στην Εικόνα 5.3.1.1 πιο κάτω την διαφορά ενός μή επεξεργασμένου αρχείου που κατεβάσαμε απο την ιστοσελίδα του Αστεροσκοπείου (αριστερά), με το πώς φαίνονται τα δεδομένα στο ίδιο αρχείο μετά την επεξεργασία τους(δεξιά).

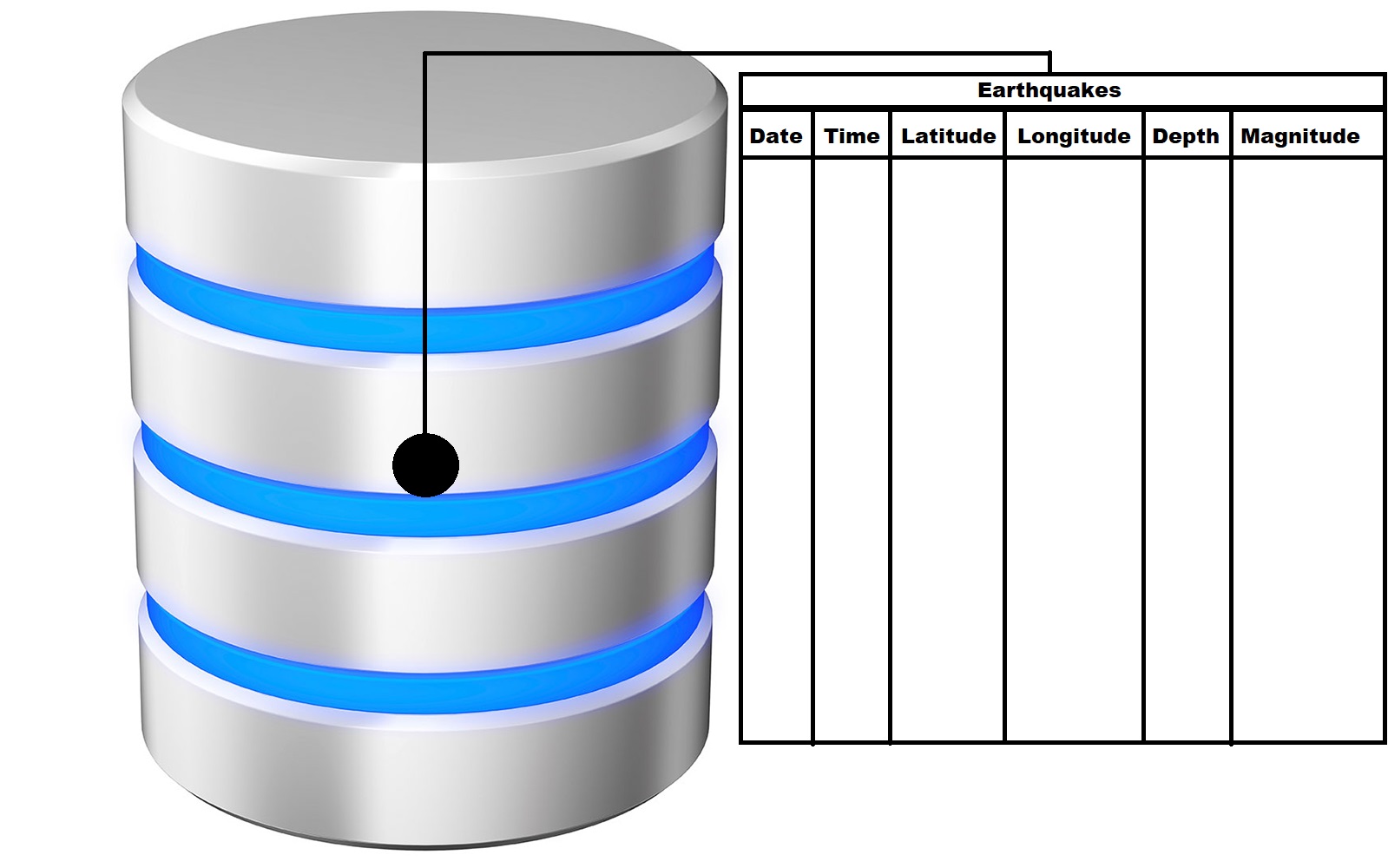
Εικόνα 5.3.1.1



### Βάση Δεδομένων

Η μοντελοποίηση της βάσης είναι αρκετά απλή αφού υπάρχει μόνο ένας πίνακας με 6 στήλες που θα περιέχει όλους τους καταγεγραμμένους σεισμούς απο το 1964 μέχρι και σήμερα. Βλέπουμε την εικόνα 5.3.2.1 πιο κάτω:

Εικόνα 5.3.2.1

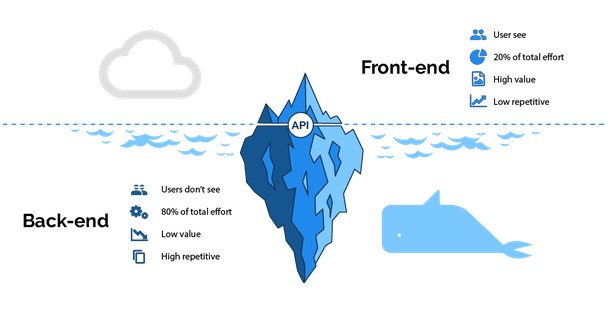


## Front End, Back End and User Interface

Κατ’αρχάς ας εξηγήσουμε τους όρους Front και Back end.

Το Frontend και το Backend είναι οι δύο πιο δημοφιλείς όροι που χρησιμοποιούνται στην ανάπτυξη Ιστού. Αυτοί οι όροι είναι πολύ σημαντικοί για την ανάπτυξη ιστού, αλλά είναι αρκετά διαφορετικοί μεταξύ τους. Κάθε πλευρά πρέπει να επικοινωνεί και να λειτουργεί αποτελεσματικά με την άλλη ως ενιαία μονάδα για τη βελτίωση της λειτουργικότητας του ιστότοπου. Για μια γενική εικόνα βλέπουμε την Εικόνα 5.3.3.1.

Εικόνα 5.3.3.1



## Front End

Το τμήμα ενός ιστότοπου με το οποίο αλληλεπιδρά απευθείας ο χρήστης ονομάζεται front end. Αναφέρεται επίσης ως «πελάτης» της εφαρμογής. Περιλαμβάνει όλες τις αλληλεπιδράσεις που έχουνε οι χρήστες(π.χ. χρώματα και στυλ κειμένου, εικόνες, γραφήματα, πίνακες, κουμπιά, χρώματα χάρτες κ.τ.λ). Η δομή, η σχεδίαση, η συμπεριφορά και το περιεχόμενο όλων όσων εμφανίζονται στις οθόνες του προγράμματος περιήγησης όταν ανοίγουν ιστότοποι, εφαρμογές ιστού ή εφαρμογές για κινητά, υλοποιούνται από προγραμματιστές Front End. Η ανταπόκριση και η απόδοση είναι δύο κύριοι στόχοι του Front End. Ο προγραμματιστής πρέπει να διασφαλίσει ότι ο ιστότοπος αποκρίνεται, δηλαδή εμφανίζεται σωστά σε συσκευές όλων των μεγεθών, κανένα μέρος του ιστότοπου δεν πρέπει να συμπεριφέρεται ασυνήθιστα ανεξάρτητα από το μέγεθος της οθόνης.

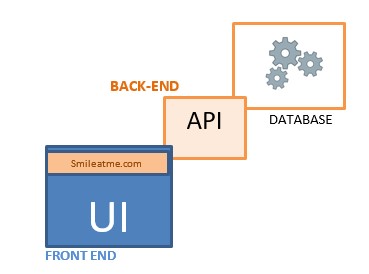
## *Back End*

Το backend είναι η πλευρά του διακομιστή του ιστότοπου (server-side). Αποθηκεύει και τακτοποιεί δεδομένα, διασφαλίζοντας επίσης ότι όλα στην πλευρά του πελάτη του ιστότοπου (client-side) λειτουργούν όπως πρέπει. Είναι το μέρος του ιστότοπου που δεν μπορεί να δει ο χρήστης και να αλληλεπιδράσει μαζί του. Τα εξαρτήματα και τα χαρακτηριστικά που αναπτύχθηκαν από σχεδιαστές backend είναι έμμεσα προσβάσιμα από τους χρήστες μέσω μιας εφαρμογής front-end. Δραστηριότητες, όπως η σύνταξη API, η δημιουργία βιβλιοθηκών και η εργασία με στοιχεία συστήματος χωρίς διεπαφές χρήστη ή ακόμη και συστήματα επιστημονικού προγραμματισμού, περιλαμβάνονται επίσης στο backend.

***Ένωση Front End, Back End και Βάσης Δεδομένων***

Ολόκληρη η εφαρμογή/σύστημα χωρίζεται σε 2 μέρη δηλαδή τα οποία στο τέλος είναι συνδεδεμένα. Παρουσιάζεται παρακάτω ένα σχεδιάγραμμα που ξεκαθαρίζει το το πώς απεικονίζονται (Εικόνα 5.3.3.2).

Εικόνα 5.3.3.2



# Αρχιτεκτονική του Συστήματος και Εργαλεία

## Environment setup and deployment

Το setup και deployment του περιβάλλοντος χρειάστηκε αρκετή διαδικασία. Παραπάνω επειδή έπρεπε να βρεθούν όλα τα dependencies και τα απαραίτητα versions τους για να δουλεύουν όλα μεταξύ τους. Με τη βοήθεια του Spring Boot (για την κύρια φάση του project) πολλά προστεθηκαν εν τέλη αυτόματα αλλά υπήρξαν και αρκετά αλλα που χρειάστηκε να τα προσθέσω μόνος μου χειροκίνητα απο το Maven Repository.

### IDE and Application Server

Ο IDE που χρησιμοποιήθηκε είναι ο Eclipse και το συγκεκριμένο version είναι το Eclipse IDE for Enterprise Java and Web Developers - 2022-03. Για Application Server χρησιμοποιήθηκε ο Tomcat 9.0.62. Το java version που χρησιμοποιήθηκε στους υπολογιστές που έγινε το σύστημα είναι 1.8.0\_333. Αφού εγκαθιστούμε τα 3 αυτά απαραίτητα λογισμικά στον υπολογιστή μας, ανοίγουμε το Eclipse για να προσθέσουμε και δηλαδή να ενώσουμε τον Tomcat με το Eclipse. Πιο συγκεκριμένα τα βήματα που ακολουθούμε είναι:

Servers tab->New->Servers->{Επιλέγουμε το version του Tomcat που θέλουμε να χρησιμοποιήσουμε(9.0 στη δική μας περίπτωση)}->Next->Browse->{Βρίσκουμε τον φάκελο στον οποίο εγκαταστήσαμε τον Tomcat}->Next->Add->Finish.

### Project Creation

Τώρα είμαστε έτοιμοι να δημιουργήσουμε το πρώτο μας project που είναι αυτό στο οποίο κατεβάζουμε όλους τους καταλόγους (1964-2021) απο την ιστοσελίδα του Αστεροσκοπείου και τα επεξεργαζόμαστε, δημιουργώντας τα ξανά σε καλύτερη και πιο ευανάγνωστη μορφή έτσι ώστε να μπορούν να εισαχθούν κατάλληλα στη βάση δεδομένων.

Δημιουργήθηκαν συνολικά 3 projects για το σύστημά μας. Το πρώτο θα τρέξει μόνο μια φορά και θα σταματήσει. Το δεύτερο είναι το κύριο κομμάτι του συστήματος (δηλαδή το REST API μαζί με την κύρια σελίδα με τα γραφικά και το swagger), το οποίο θα τρέχει συνεχώς. Το τρίτο είναι το κομμάτι που είναι υπεύθυνο για την ενημέρωση της βάσης δεδομένων, στο οποίο απαιτείται ένα Login απο έναν admin.

Τα δύο projects είναι Maven Projects. Το πρώτο δημιουργήθηκε κανονικά με τα default settings που απαιτούνται για να χρησιμοποιηθεί ένα Maven Project. Το δεύτερο δημιουργήθηκε με την βοήθεια του spring boot initializr και στη συνέχεια απλά το προσθέτουμε στο Eclipse σαν maven project. Το τρίτο project είναι Dynamic Web Project.

Πιο συγκεκριμένα, για το πρώτο και τρίτο Project κάνουμε την εξής διαδικασία:

* File setting->New->Dynamic Web Project->Next (default settings, απλά προσέχουμε να βάλουμε τίτλο και να επιλέξουμε τον Tomcat 9.0)->Next->Tick option to create web.xml(Optional)->Finish. (Δεν χρειάζεται να κάνουμε κάτι παραπάνω για το τρίτο project)
* Δεξί κλικ στο project μας->Configure->Convert to Maven Project->Finish. (Αυτό το βήμα είναι απαραίτητο για το πρώτο project)

Για το δεύτερο project, απο το springboot initializr ( <https://start.spring.io> ) επιλέγουμε τα εξής options:

* Project -> Maven Project
* Language -> Java
* Spring Boot -> 2.7.3
* Packaging -> Jar
* Java -> 11
* Dependencies -> Spring Web, Jersey, MySQL Driver

Στο τέλος πατάμε το κουμπί ‘Generate’ και ξεκινάει το κατέβασμα του αρχείου μας αυτόματα.

Κάνουμε unzip το project κατα προτίμηση μέσα στον φάκελο του workspace του Eclipse, όπου βρίσκονται και τα υπόλοιπα projects. Στη συνέχεια, μέσα στο Eclipse IDE πρέπει να κάνουμε import το project με τον εξής τρόπο:

* File setting->Import->Existing Maven Projects->Next->Browse->{ Επιλέγουμε τον φάκελο του project }-> Finish.

Τέλος, για κάθε project πρέπει να προσθέσουμε συγκεκριμένα dependencies για τα frameworks που χρειαζόμαστε, στο pom.xml.

## Τεχνολογίες/frameworks used

### Spring Framework and Spring Boot

Το Spring Framework παρέχει ένα ολοκληρωμένο μοντέλο προγραμματισμού και διαμόρφωσης για σύγχρονες εταιρικές εφαρμογές που βασίζονται σε Java - σε κάθε είδους πλατφόρμα ανάπτυξης.

Ένα βασικό στοιχείο του Spring είναι η υποστήριξη υποδομής σε επίπεδο εφαρμογών: Το Spring εστιάζει στο "plumbing" των εταιρικών εφαρμογών, έτσι ώστε οι ομάδες να μπορούν να επικεντρωθούν στην επιχειρηματική λογική σε επίπεδο εφαρμογής, χωρίς περιττούς δεσμούς με συγκεκριμένα περιβάλλοντα ανάπτυξης.

Το Spring Boot ουσιαστικά διευκολύνει τη δημιουργία αυτόνομων, production-grade Spring based εφαρμογών που μπορούμε απλά να ‘τρέξουμε’.

Λαμβάνουμε μια γνώμη σχετικά με την πλατφόρμα Spring και τις βιβλιοθήκες τρίτων, ώστε να μπορούμε να ξεκινήσουμε με ελάχιστη φασαρία (Όπως δηλαδή να ψάχνουμε χειροκίνητα για dependencies και τα συγκεκριμένα versions τους στο Maven Repository) .

### Eclipse Jersey

Το Eclipse Jersey ανήκει στην κατηγορία των Jersey RESTful Web Services και είναι ένα open source framework που χρησιμοποιείται για την ανάπτυξη τέτοιων RESTful Web Services υλοποιημένες σε Java.

### Swagger

Το Swagger είναι μια τεχνολογία με την οποία περιγράφουμε την δομή του API μας. Περιγράφοντας την δομή του API μας, δίνεται τη δυνατότητα στα μηχανήματα να μπορούν να το διαβάσουν. Διαβάζοντας την δομή του μας επιτρέπεται η δημιουργία ενός documentation του, το οποίο είναι όμορφο, οργανωμένο και κατανοητό. Χρησιμοποιώντας το Swagger επίσης μας επιτρέπεται να χρησιμοποιήσουμε το API μας και σε άλλες εφαρμογές, όπως και κάνουμε στο σύστημά μας με την χρήση του Postman.

### Leaflet

Το Leaflet είναι η κορυφαία open source βιβλιοθήκη JavaScript για διαδραστικούς χάρτες που είναι επίσης mobile friendly. Περιέχει όλα τα χαρακτηριστικά χαρτογράφησης που χρειάζονται οι περισσότεροι προγραμματιστές.

Το Leaflet έχει σχεδιαστεί με γνώμονα την απλότητα, την απόδοση και τη χρηστικότητα. Λειτουργεί αποτελεσματικά σε όλες τις μεγάλες πλατφόρμες για επιτραπέζιους υπολογιστές και κινητές συσκευές, μπορεί να επεκταθεί με πολλά plugins, έχει ένα όμορφο, εύχρηστο και well-documented API και έναν απλό, ευανάγνωστο source code.

### D3.js

Το D3.js είναι μια βιβλιοθήκη JavaScript για χειρισμό εγγράφων που βασίζονται σε δεδομένα. Το D3 μάς βοηθά να φέρουμε τα δεδομένα μας σε μια μορφή αναπαράστασης χρησιμοποιώντας HTML, SVGs και CSS. Η D3 συνδυάζει στοιχεία οπτικοποίησης και μια προσέγγιση που βασίζεται σε δεδομένα (όπως αναφέραμε παραπάνω) για τη χειραγώγηση DOM Elements. Δηλαδή πιο συχνά βλέπουμε την D3 να χρησιμοποιείται για τη δημιουργία γραφικών παραστάσεων που εξαρτώνται από τα δεδομένα που προσφέρει π.χ. ένα API.

### Postman

Το Postman είναι μια πλατφόρμα API για δημιουργία και χρήση API. Το Postman απλοποιεί κάθε βήμα του κύκλου ζωής του API και βελτιστοποιεί τη συνεργασία, ώστε να μπορούμε να δημιουργήσουμε καλύτερα API—πιο γρήγορα. Απαραίτητη είναι η εγκατάσταση της εφαρμογής στον υπολογιστή μας και η δημιουργία ενός λογαριασμού. Με τη βοήθεια του Swagger, η χρήση του Postman φαίνεται να είναι η πιο βέλτιστη σε θέμα response time και επιλογής response body.

# Δοκιμή Συστήματος

Το σύστημά μας δοκιμάστηκε σε έναν φορητό υπολογιστή με τα εξής χαρακτηριστικά:

* Λειτουργικό Σύστημα: Windows 11 Home
* Επεξεργαστής: AMD Ryzen 7 4800H, Base speed: 2.90 GHz, 8 Cores
* Κάρτα γραφικών: NVIDIA GeForce GTX 1650 Ti 4GB
* RAM: 16GB
* SSD: Samsung 512GB
* Browsers: Opera, Firefox (Recommended)

Το σύστημα δοκιμάστηκε πλήρως τουλάχιστον 5 φορές στον υπολογιστή σε 2 διαφορετικούς browsers και παρατηρήθηκε πως ο firefox κάνει data rendering (Swagger, Leaflet markers) με παραπάνω ευκολία. Το Swagger όμως σε κανένα απο τα 2 browser δεν ήταν ικανό να εκτελέσει μερικές λειτουργίες (5 οι οποίες αναφέρονται στο documentation και συστήνεται η χρήση του Postman). Το Leaflet map με τα markers εκτελεί όλα τα queries ανεξαρτήτως μεγέθου σε βέλτιστο χρόνο και στα 2 προγράμματα περιήγησης ιστού. Το D3.js pie chart το ίδιο! Λόγω του όγκου αρχείων και δεδομένων κάθε αρχείου (ειδικά τα έτη μετά το 2010) η δημιουργία των νέων αρχείων που δημιουργούνται λόγω απαραίτητης επεξεργασίας των δεδομένων τους απαιτεί δυστυχώς αρκετό χρόνο. Από την άλλη ‘όμως είναι ένα κόστος που πληρώνουμε μόνο την πρώτη φορά που τρέχουμε το project, κατα την εγκατάσταση του συστήματος. Το σύστημα εκτός από csv αρχεία κατά την ενημέρωσή του μπορεί να δεχτεί και άλλου είδους αρχεία (π.χ. php - που είναι το format του καταλόγου που ενημερώνεται καθημερινά στην ιστοσελίδα του Αστεροσκοπείου).

# Οδηγίες Χρήσης του Συστήματος

## Swagger

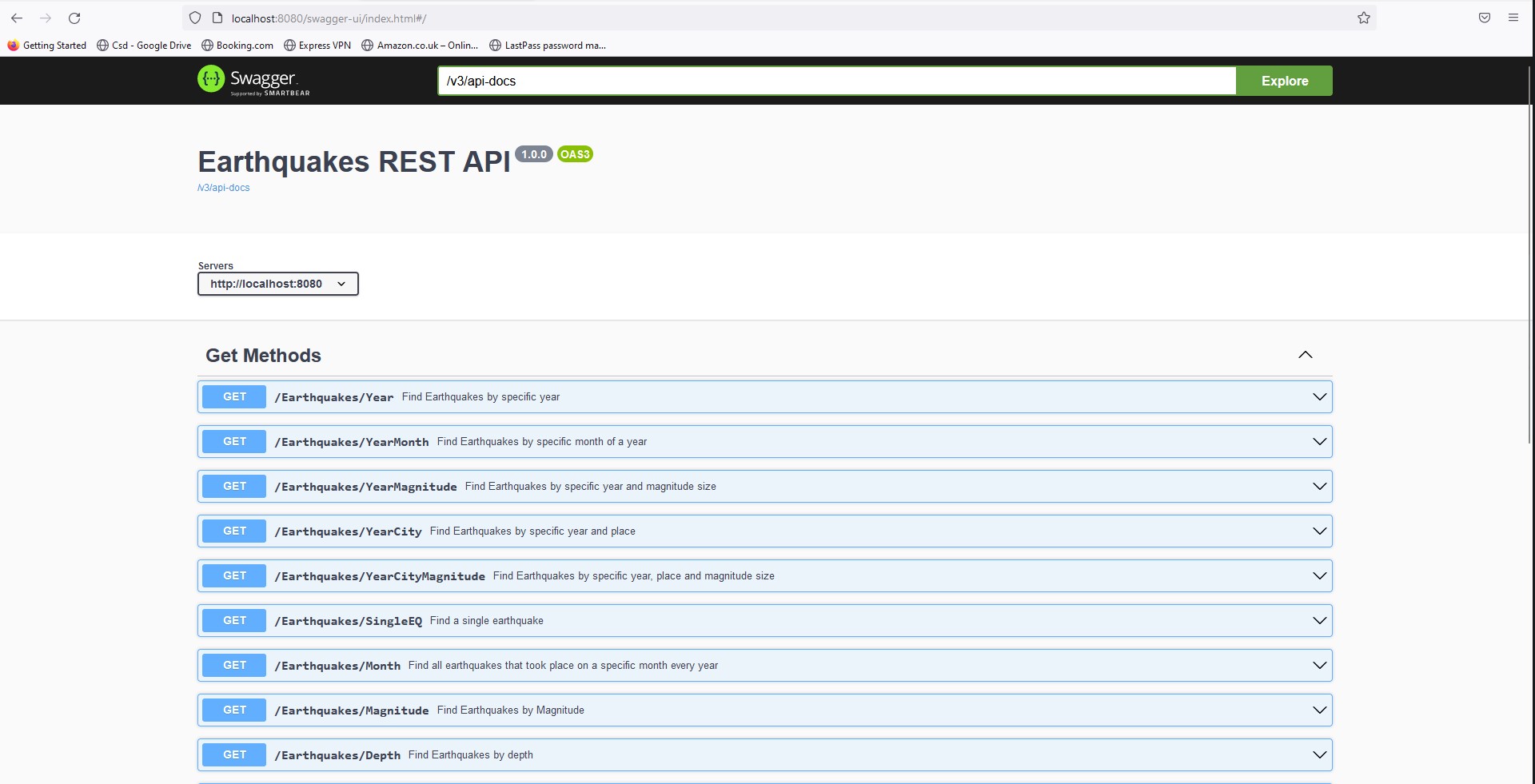
Το Swagger δουλεύει ουσιαστικά σαν το documentation του συστήματός μας, αφού υπάρχει αναλυτική περιγραφή για κάθε λειτουργία του. Επίσης μας επιτρέπει να δοκιμάσουμε τις λειτουργίες του και η χρήση του είναι ευνόητη.

Αρχικά, υπάρχουν 2 τρόποι για να περιηγηθούμε στο Swagger.

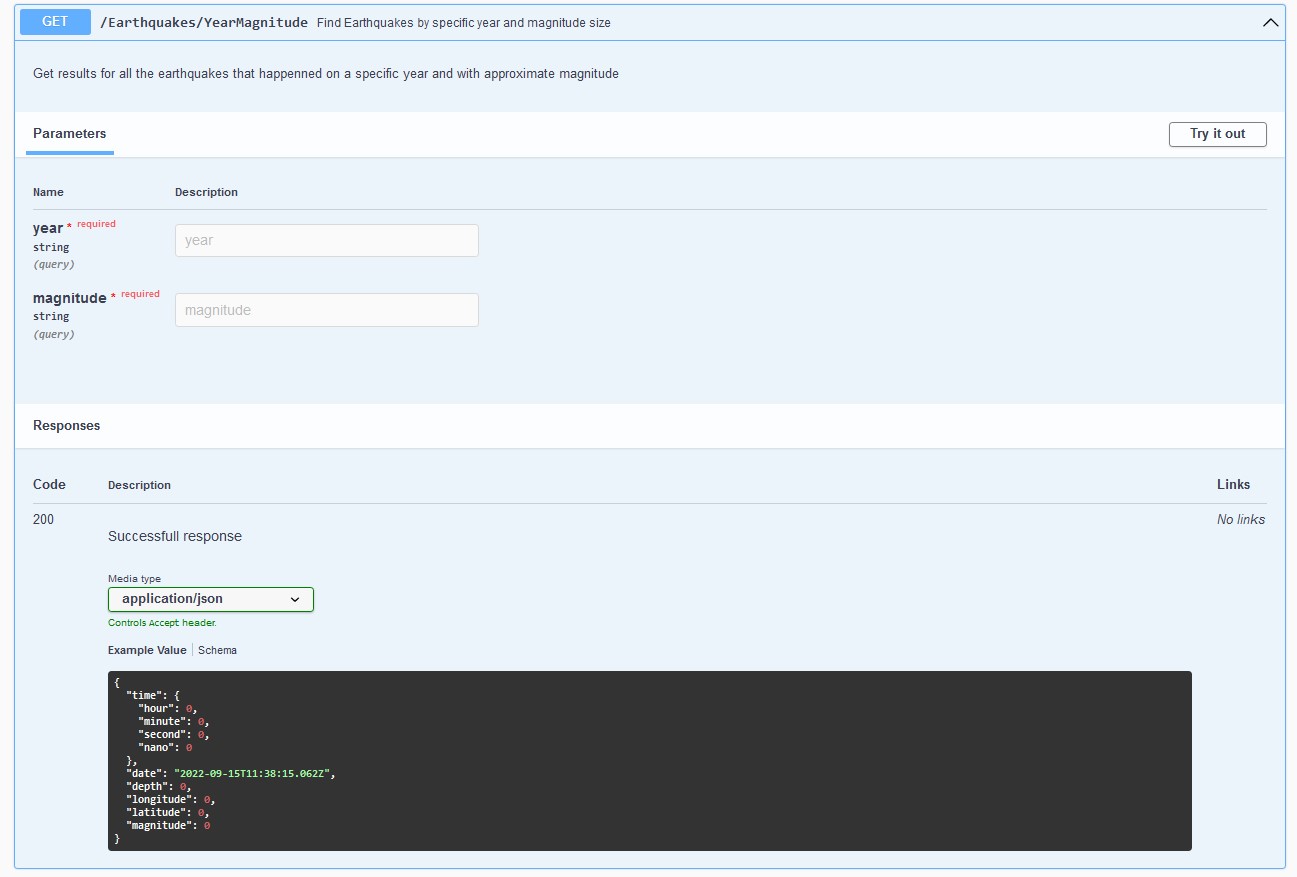
1. Να ακολουθήσουμε τον σύνδεσμο στην κύρια σελίδα
2. Να πάμε στο link {server\_name:port}/swagger-ui/index.html



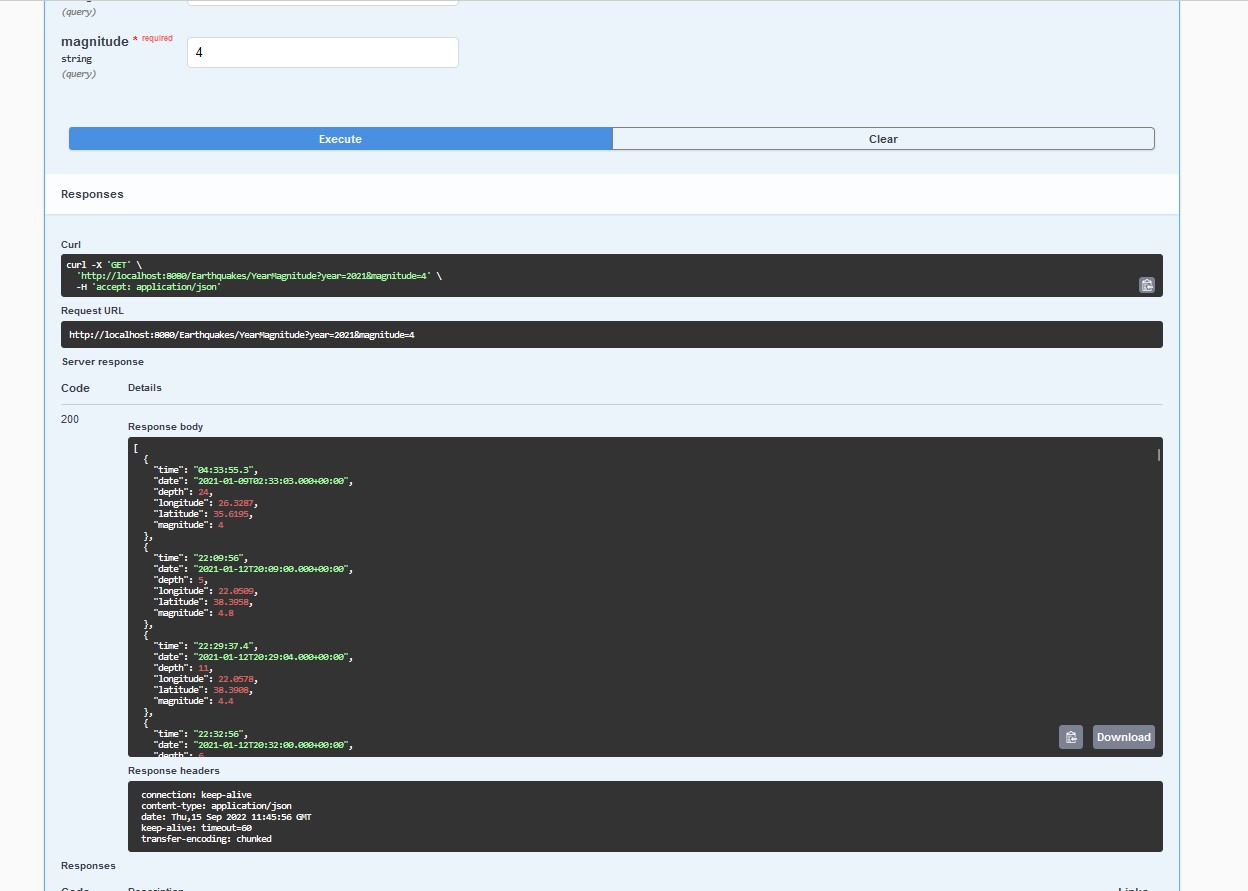
Στη συνέχεια θα εισαχθούμε στην σελίδα του Swagger.



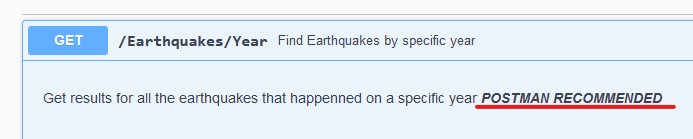
Κάνοντας κλικ πάνω σε μια λειτουργία, εμφανίζονται οι πληροφορίες και οδηγίες χρήσης της.



Φαίνονται επίσης ένα παράδειγμα επιτυχής ανταπόκρισης πιο κάτω. Για να μας επιτραπεί η είσοδος τιμών στις παραμέτρους, μπορούμε απλά να πατήσουμε το κουμπί ‘Try it out’. Έπειτα αφού βάλουμε επιτρεπτές τιμές, πατάμε το κουμπί ‘Execute’ για να στείλουμε το request μας. Αν όλα πάνε καλά και δεν έχουμε άδειο αποτέλεσμα, θα έχουμε ένα response body σε json.

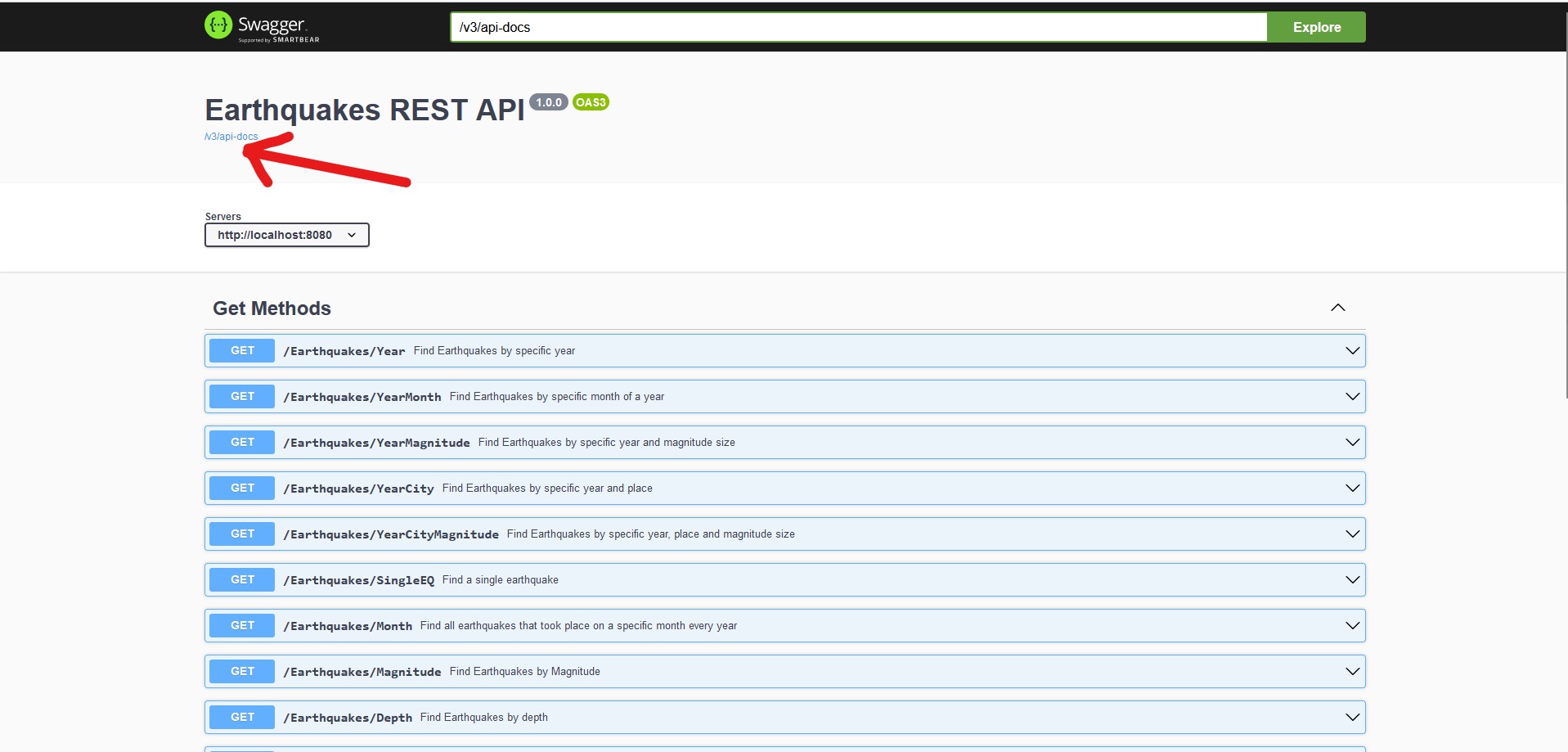


Μετά αν θέλουμε να δοκιμάσουμε κάποια άλλη λειτουργία, συστήνεται – έτσι ώστε να μην κάνουμε τον browser μας πιο αργό – να πατάμε το κουμπί ‘Clear’ που βρίσκεται δίπλα από το ‘Execute’ και στη συνέχεια το παραπάνω κουμπί ‘Cancel’. Αυτό δεν θα πρέπει να είναι πρόβλημα για τους παραπάνω υπολογιστές και browsers, γι’αυτό και μπορείτε να το αγνοήσετε. Μπορούμε να αποθηκεύσουμε το response body μετά την εκτέλεση μιας λειτουργίας πατώντας το κουμπί ‘Download’ κάτω δεξιά στο response body. Το κουμπί στα αριστερά του είναι αυτό που κάνει το response body Copy to Clipboard. Προειδοποίηση: Μερικές λειτουργίες λειτουργούν καλύτερα - αν όχι και μόνο – στο Postman. Για αυτές τις λειτουργίες υπάρχει η κατάλληλη προειδοποίηση στην περιγραφή κάθε λειτουργίας.



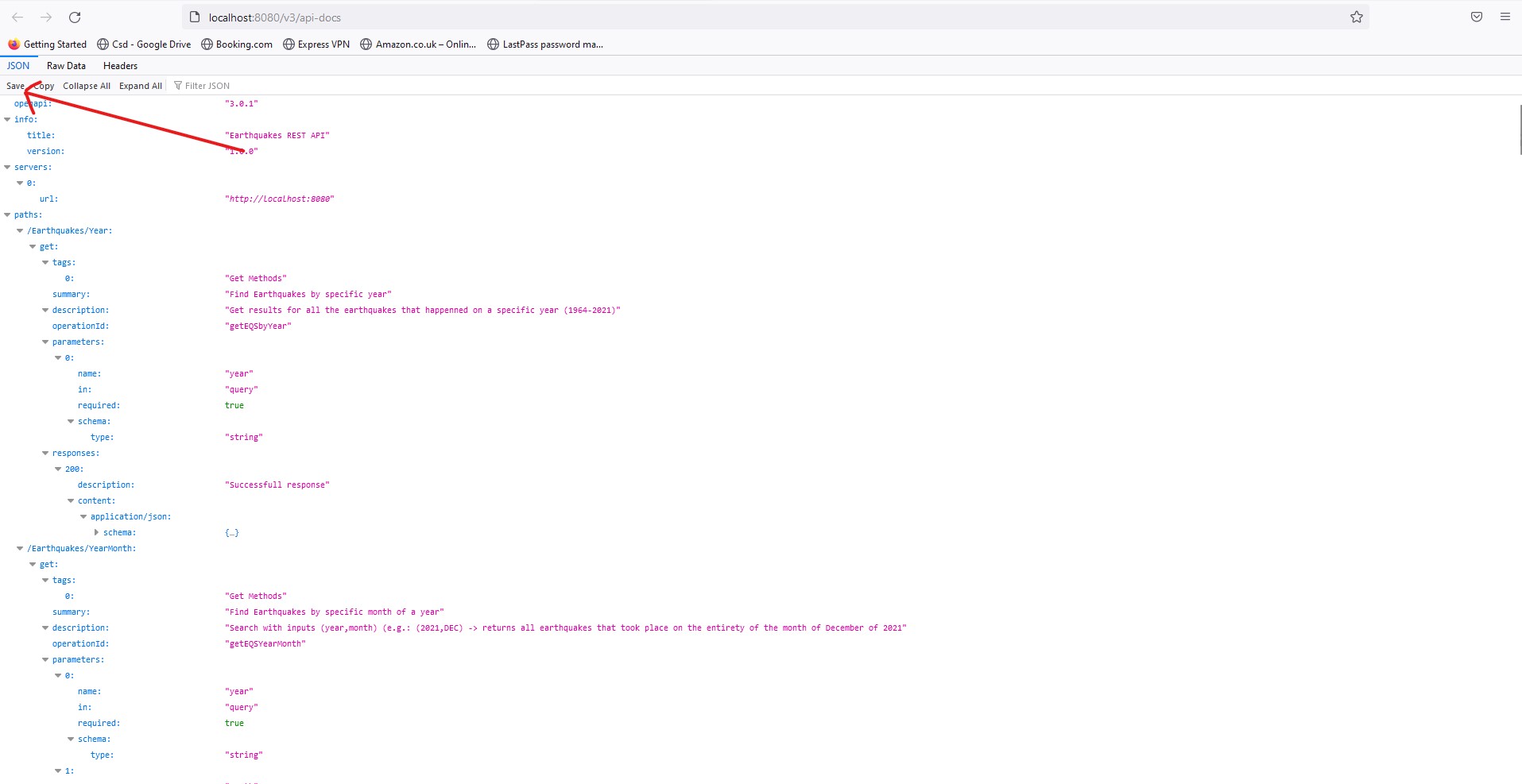
Τέλος, κάτι που θα χρειαστούμε για το Postman, για να κατεβάσουμε το api-documentation του swagger σε μορφή json ακολουθούμε τον σύνδεσμο στην αρχή της σελίδας ‘ [/v3/api-docs](http://localhost:8080/v3/api-docs) ‘.

Εικόνα 8.1.6



Στη συνέχεια αποθηκεύουμε στον υπολογιστή μας το αρχείο.

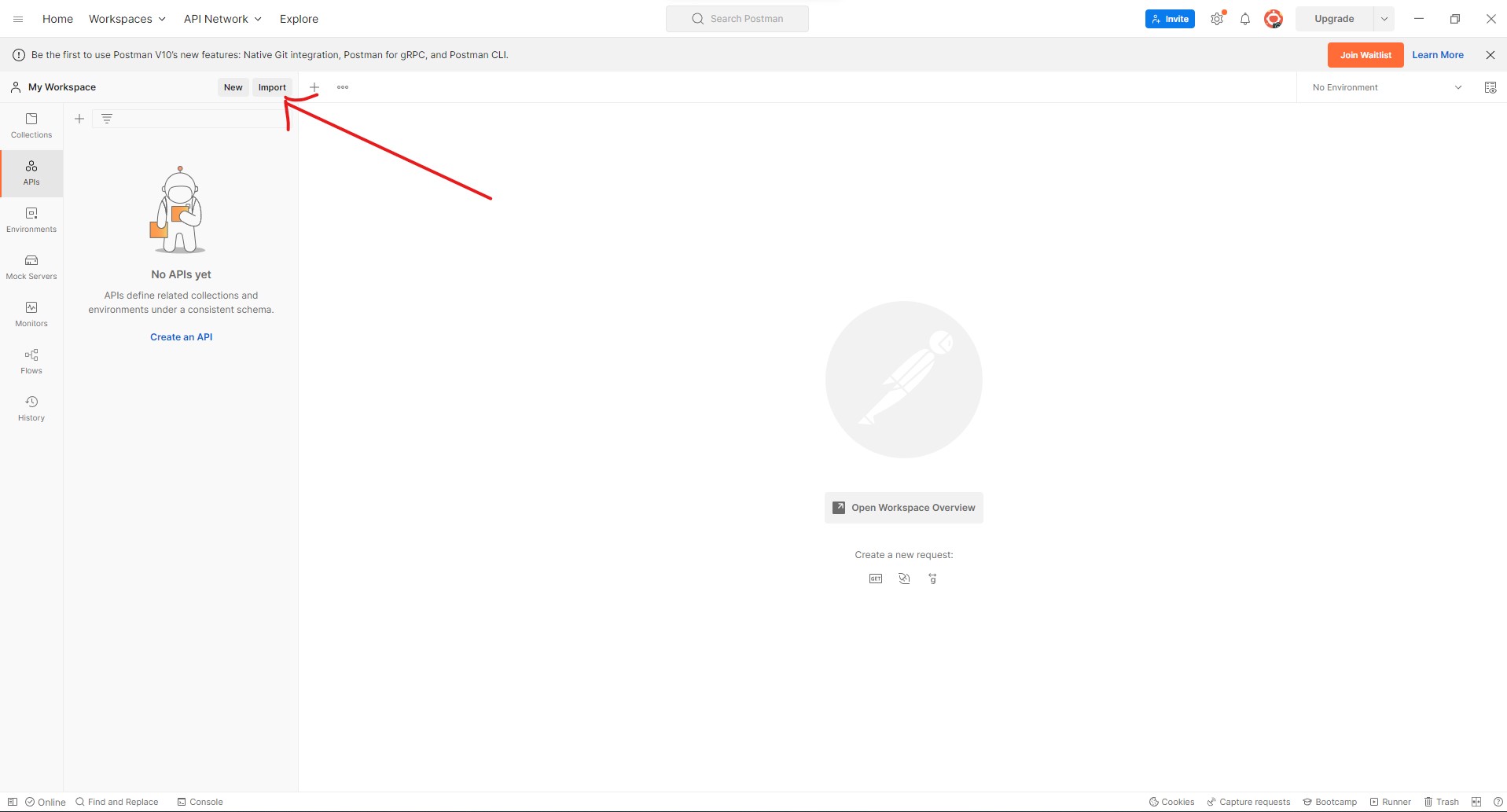
Εικόνα 8.1.7



## Postman

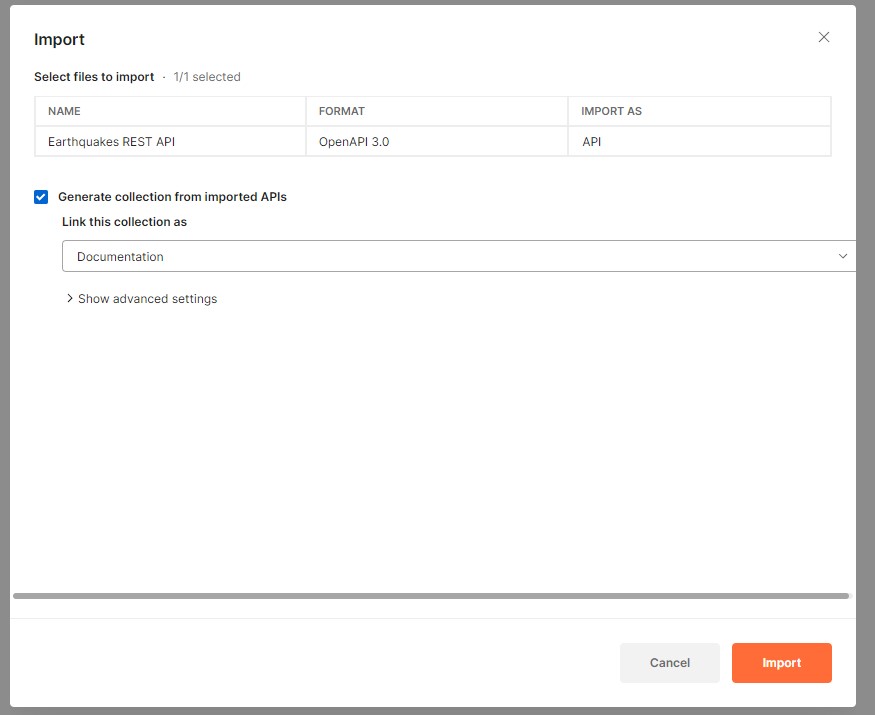
Για να χρησιμοποιήσουμε το Postman πρέπει αρχικά να κατεβάσουμε την εφαρμογή απο τη ιστοσελίδα (<https://www.postman.com/downloads/>) για Windows και να δημιουργήσουμε λογαριασμό ( ή απλά να κάνουμε σύνδεση μέσω άλλης πλατφόρμας, δίνονται 3-4 επιλογές όπως Google και Github ). Στη συνέχεια ακολουθούμε τα βήματα που αναφέρονται παραπάνω στις εικόνες 8.1.6 και 8.1.7 για να κατεβάσουμε το api-doc μας. Μετά από αυτό προσθέτουμε το api-doc στο Postman πατώντας το κουμπί ‘Import’ πάνω αριστερά, στη στήλη APIs.

Εικόνα 8.2.1



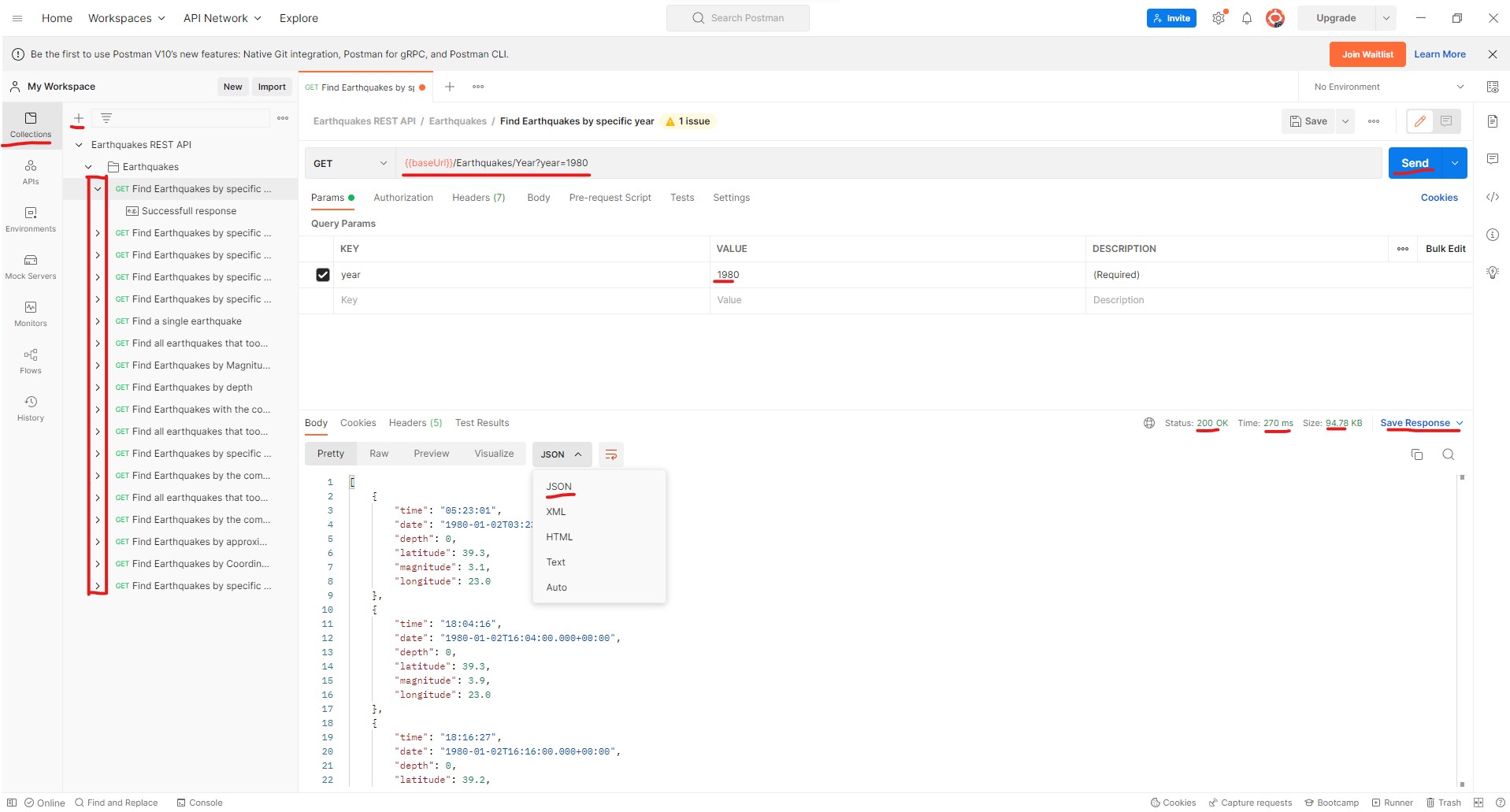
Μετά είτε κάνουμε drag and drop το doc-api μας, είτε πατάμε το ‘upload file’ και το επιλέγουμε. Θα δούμε ότι το Postman αναγνωρίζει το API μας πριν επιβεβαιώσουμε την προσθήκη του. Πατάμε το ‘Import’.

Εικόνα 8.2.2



Είμαστε έτοιμοι! Μπορούμε τώρα να βρούμε τις λειτουργίες στο Collections Tab στα αριστερά. Πιο κάτω απεικονίζεται το Postman μετά από μια επιτυχημένη ανταπόκριση μιας λειτουργίας του συστήματός μας.

Εικόνα 8.2.3



Πάμε να εξηγήσουμε τι ακριβώς βλέπουμε πιο πάνω(βλ. Εικόνα 8.2.3)

Στα αριστερά φαίνονται όλες οι λειτουργίες του συστήματος. Εμείς όπως φαίνεται πιο πάνω εκτελέσαμε τη λειτουργία που μας επιστρέφει όλους τους σεισμούς μιας συγκεκριμένης χρονιάς. Στο Value βάλαμε την τιμή 1980 και το link του request αλλάζει όπως φαίνεται πιο πάνω ({{baseUrl}}/Earthquakes/Year?year=1980). Δηλαδή όποτε θέλουμε να αλλάξουμε τις τιμές των παραμέτρων, τις αλλάζουμε στα κουτιά κάτω από την στήλη Value.Πατάμε λοιπόν το κουμπί ‘Send’ στα δεξιά και από κάτω στο ‘Body’ μπορούμε να δούμε την επιτυχές ανταπόκριση της λειτουργίας. Στην ίδια γραμμή με το ‘Body’ μπορούμε να δούμε και άλλες χρήσιμες πληροφορίες( Body, Cookies, Headers, Test Results, Status, Time, Size, Save response βλ. Εικόνα 8.2.4). Από κάτω μπορούμε να δούμε το response body, αλλά και να αλλάξουμε την μορφή και οπτικοποιηση του (Απο περιποιημένο κείμενο json σε raw, επιλογές XML,HTML,JSON,TEXT βλ. Εικόνα 8.2.5).

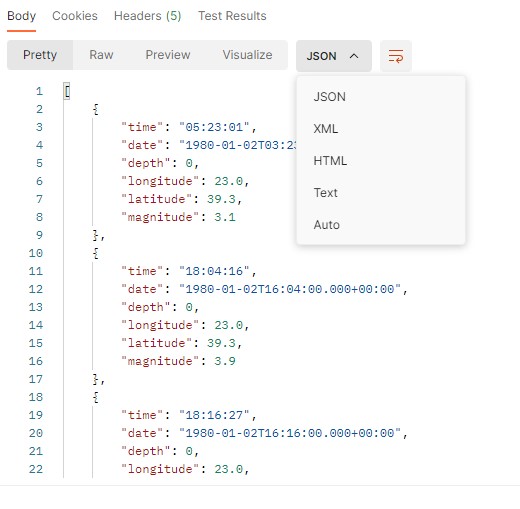
Εικόνα 8.2.4

--------------------------------------------------------------------------------------------------------------



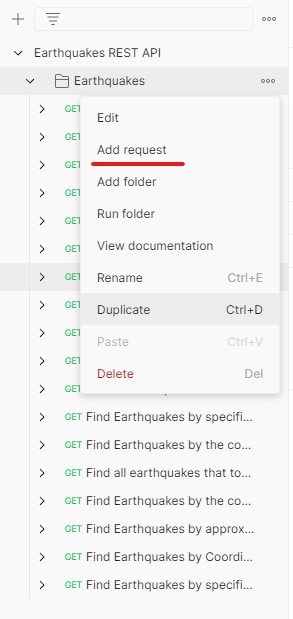
--------------------------------------------------------------------------------------------------------------

Εικόνα 8.2.5

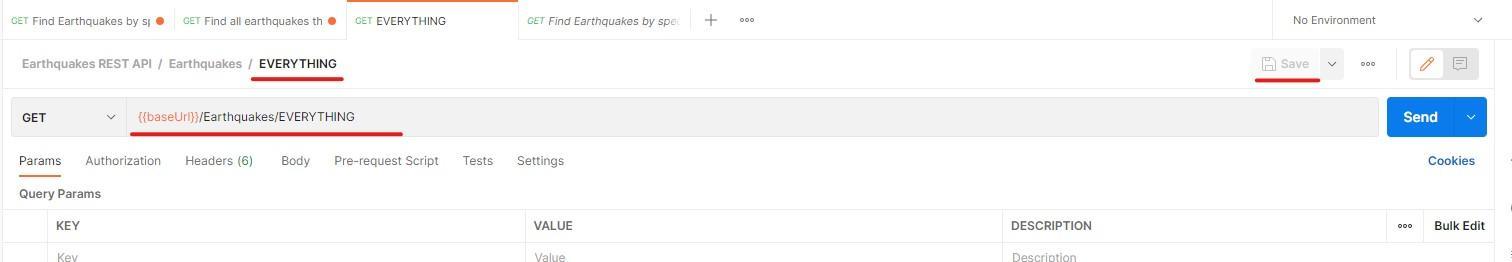


Υπάρχει μια επιπλέον λειτουργία που είναι Hidden και αν και μπορούμε να την χρησιμοποιήσουμε και εκείνη κανονικά, καλύτερα είναι να την προσθέσουμε και αυτήν μαζί με τις άλλες. Για να το κάνουμε αυτό πατάμε τις 3 τελείες (. . .) δίπλα από τον φάκελο Earthquakes παι επιλέγουμε το Add Request (βλ. Εικόνα 8.2.6). Συμπληρώνουμε το request URL και πατάμε το Save στα δεξιά. Έχουμε και την επιλογή να αλλάξουμε τον τίτλο της νέας λειτουργίας που προσθέσαμε (βλ. Εικόνα 8.2.7). Η λειτουργία αυτή δεν χρειάζεται παραμέτρους, άρα αρκεί να τρέξουμε το REQUEST URL.

Εικόνα 8.2.6



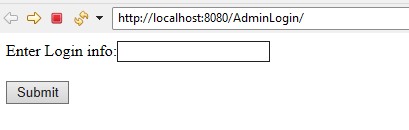
Εικόνα 8.2.7



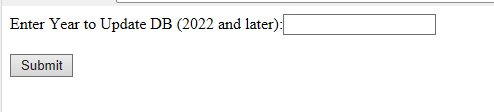
### Database Update/Login Form

Όπως αναφέραμε, για να μπορεί κάποιος να ενημερώσει τη βάση δεδομένων με ένα νέο αρχείο, θα πρέπει να έχει προνόμια admin. Με αυτά τα προνόμια μπορεί να κάνει login (βλ. Εικόνα 8.3.1), στη συνέχεια να γράψει στο textbox το έτος που θέλει να ενημερώσει (βλ. Εικόνα 8.3.2)(αξίζει να σημειωθεί εδώ πως όταν το κάνει αυτό ο χρήστης, διαγράφονται όλα τα δεδομένα της χρονιάς άν είναι εγγραμμένα μέσα στη βάση) και μετά θα του δοθεί η επιλογή μέσω του User Interface να κάνει upload το αρχείο που κατέβασε από την ιστοσελίδα του Εθνικού Αστεροσκοπείου Αθηνών (βλ. Εικόνα 8.3.3). Όταν πατήσει το κουμπί ‘Browse’, ο χρήστης θα έχει την επιλογή να διαλέξει το αρχείο κάνοντας είτε διπλό κλικ πάνω στο αρχείο, είτε ένα κλικ και πατώντας το κουμπί ‘Open’ κάτω δεξιά. Στην συνέχεια πατάμε το κουμπί ‘Save’ και το αρχείο γίνεται upload στον server και ξεκινά επίσης η επεξεργασία του. Μετά από την επεξεργασία γίνεται η εισαγωγή των δεδομένων στην βάση. Στο τέλος εμφανίζεται στην οθόνη ο φάκελος που περιέχει τα αρχεία μας ο οποίος βρίσκεται στον server, αλλά επίσης μπορούμε να δούμε τα μηνύματα επιτυχίας στο κάτω μέρος της κονσόλας.

Εικόνα 8.3.1



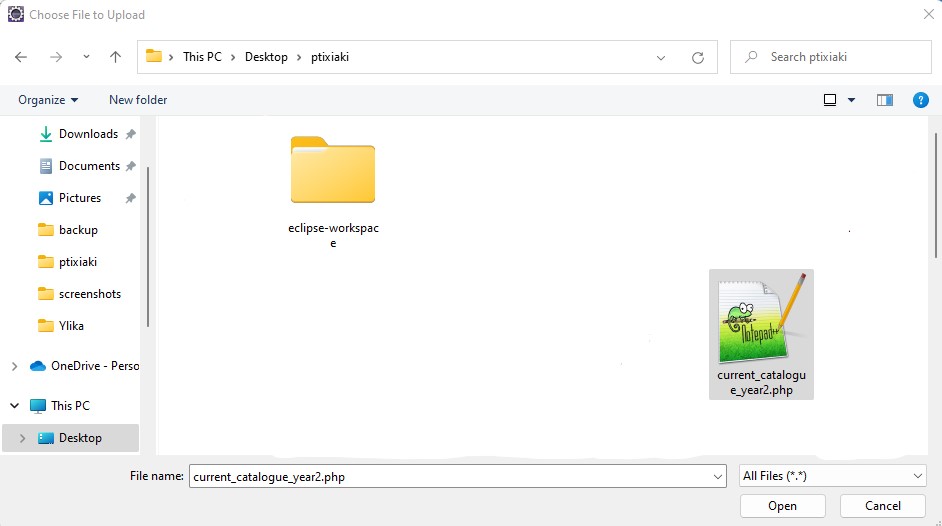
Εικόνα 8.3.2



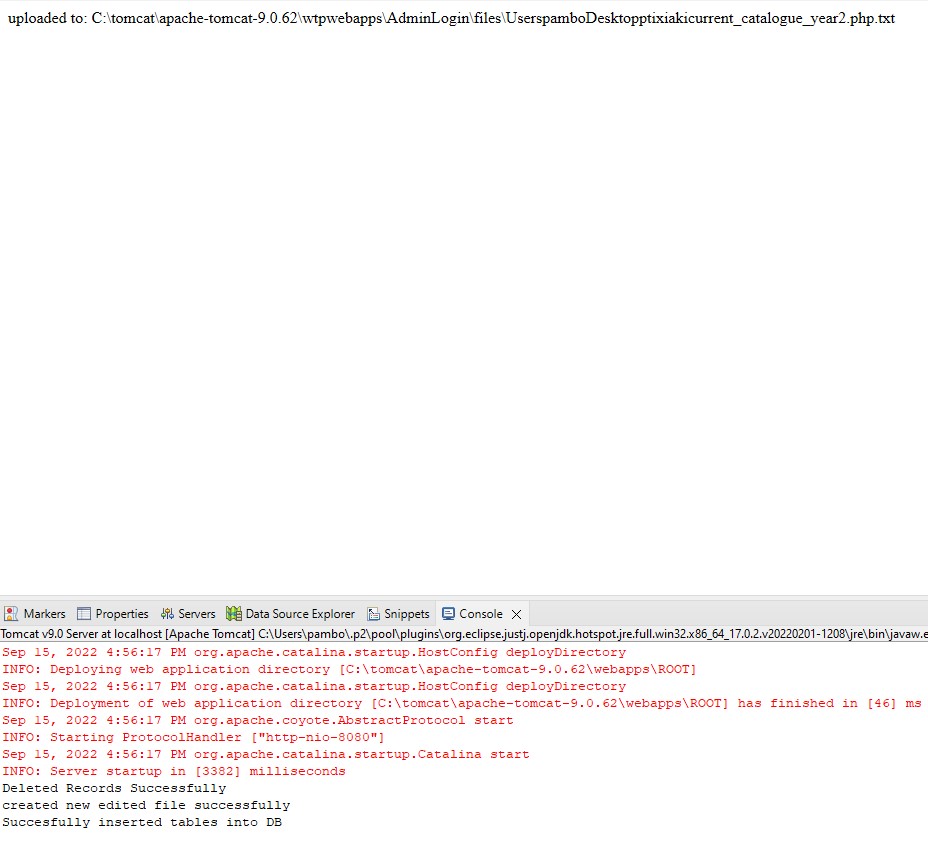
Εικόνα 8.3.3



Εικόνα 8.3.4



Εικόνα 8.3.5



## Πώς τρέχουμε τα projects?

Για το πρώτο project (P1) πολύ απλά κάνουμε δεξί κλικ στο project->run as->Java Application.

Στο δεύτερο project κάνουμε το ίδιο αλλά κάνοντας δεξί κλικ στο class ‘EarthquakeRestApplication.java’ αντί στο όνομα του project (EarthquakeREST). Τρέχει κανονικά στον Tomcat 9.0 αλλά αυτό το τακτοποιεί αυτόματα το Spring Boot. Σε περίπτωση που το Eclipse μας αναγκάσει να επιλέξουμε πώς θα τρέξει το πρόγραμμα, η επιλογή μας θα είναι ‘Spring Boot Application’.

Το τρίτο project είναι διαφορετικό αφού το τρέχουμε απευθείαν στον server. Δηλαδή κάνουμε δεξί κλικ στο όνομα του project(AdminLogin)->Run as->Run on Server.

## Dependencies

Παρακάτω φαίνονται τα pom.xml των πρώτων 2 Maven Project.

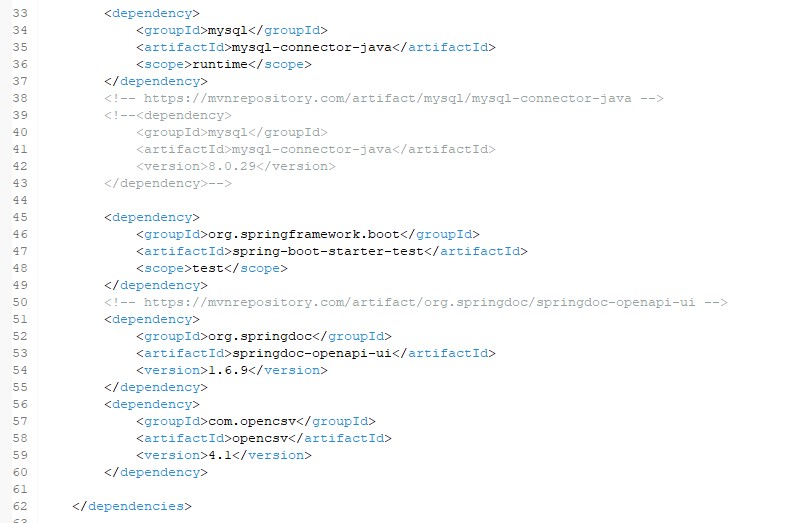
Πρώτο project (P1 – File download and editing – Εικόνα 8.5.1)

Εικόνα 8.5.1

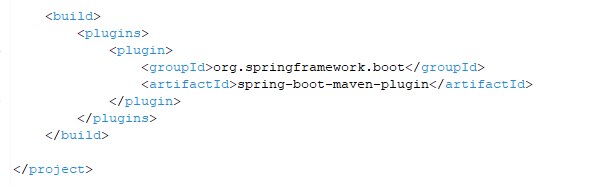


Δεύτερο project (EarthquakeREST – Κύριο PROJECT – Spring Boot Application – Εικόνα 8.5.2 και 8.5.3)

Εικόνα 8.5.2



Εικόνα 8.5.3



# Επίλογος

***Σύνοψη λειτουργικότητας του συστήματος***

Το σύστημα , σε πρώτη φάση, δέχεται αρχεία (από το Εθνικό Ατεροσκοπείο Αθηνών https://www.gein.noa.gr/en/services-products/earthquake-catalogs/ ) με δεδομένα που επεξεργάζονται σε μια καλύτερη μορφή και δημιουργούνται τα νέα αρχεία σε φάκελο που βρίσκεται μέσα στον server. Στη συνέχεια καταχωρούνται τα δεδομένα μέσα στην βάση δεδομένων, η οποία περιέχει έναν πίνακα με τα 6 απαραίτητα χαρακτηριστικά που έχει ο κάθε σεισμός.

Στην δεύτερη φάση δημιουργείται το RESTful web service μας. Η υπηρεσία μας παίρνει τα δεδομένα και με τη χρήση των Object της Java μπορούμε να αναπαραστήσουμε κάθε σεισμό σαν ένα java object, το οποίο θα εμφανιστεί στο request body όταν εκτελεστεί μια λειτουργία. Δημιουργήθηκαν στο σύνολο 19 λειτουργίες. Η μια είναι κρυμμένη (αλλά documented) και οι 5 από αυτές (για τις οποίες υπάρχουν κατάλληλες προειδοποιήσεις) δουλεύουνε βέλτιστα στην εφαρμογή Postman. Επίσης δημιουργήθηκε το (κυρίως για developers) Swagger που δουλεύει σαν documentation για την κάθε λειτουργία αλλά και για να δοκιμάσουμε τις λειτουργίες του συστήματος. Με τη βοήθεια του Swagger μπορούμε να χρησιμοποιήσουμε και το Postman για πιο αποτελεσματικές χρήσεις των λειτουργιών του συστήματος. Δημιουργήθηκε σε αυτή τη φάση επίσης και η κύρια σελίδα του συστήματος (index.html->HTML,CSS) στην οποία μπορεί ένας casual user να κάνει διάφορα queries τα οποία παρουσιάζονται οπτικοποιημένα σε έναν χάρτη (Leaflet) με markers διαφορετικών χρωμάτων, με περιγραφή για το κάθε χρώμα. Επίσης στην κύρια σελίδα παρουσιάζεται ένα pie chart (D3.js) μετά από κάθε υποβολή καινουργιου query μαζί με λεπτομερείς κείμενο. Τέλος, στην κύρια σελίδα υπάρχει επίσης ένας σύνδεσμος για περιήγηση στο Swagger.

Στην Τρίτη φάση δημιουργήθηκε ένα Admin Login form στο οποίο ένας χρήστης με τα κατάλληλα προνόμια μπορεί να ενημερώσει τη βάση δεδομένων. Για να το κάνει αυτό πρέπει να κατεβάσει το αρχείο που επιθυμεί απο την ιστοσελίδα του Εθνικού Αστεροσκοπείου Αθηνών και στη συνεχεια να το κάνει upload στον server. Σε αυτή την φάση χρησιμοποιήθηκαν JSPs και Java Servlets.

***Αδυναμίες και Μελλοντική Εργασία***

Η κύρια αδυναμία του συστήματος είναι στην πρώτη φάση όταν επεξεργαζόμαστε τα αρχεία του Αστεροσκοπείου και ουσιαστικά δημιουργούμε καινούργια αρχεία που είναι πιο ευανάγνωστα, αλλά και κατάλληλα για εισαγωγή δεδομένων στην βάση δεδομένων. Επειδή παίρνουμε τα αρχεία απο το 1964-2021, και μετά το 2010 τα αρχεία έχουν περίπου 25,000 εγγραφές, το πρόγραμμα είναι χρονοβόρο(5-6 λεπτά ανάλογα με τον υπολογιστή στον οποίο τρέχει). Επίσης επειδή κατεβάζει όλα τα αρχεία ένα ένα (αν και υπάρχει ένας κατάλογος με όλα μαζεμένα, αλλά το αρχείο είναι υπερβολικά μεγάλο για να γίνει render σε οποιονδήποτε υπολογιστή στο Eclipse), ο χρόνος που θα τρέξει το πρώτο project (P1) εξαρτάται από τα χαρακτηριστικά του υπολογιστή μας και την ταχύτητα δικτύου μας. Στο μέλλον κοιτάμε για βελτιστοποίηση.

Η δεύτερη αδυναμία του συστήματος είναι λόγο του τεράστιου όγκου δεδομένων, δεν μπορούμε να δοκιμάσουμε όλες τις λειτουργίες στο Swagger. Αυτό δυστυχώς προς το παρόν είναι αναπόφευκτο, αλλά δεν μας ενοχλεί επειδή υπάρχει το Postman, για το οποίο εξήγησα αναλυτικά πως μπορεί να χρησιμοποιηθεί. Είναι ένα πολύ καλό εργαλείο που χρησιμοποιείται πλέον σήμερα από εκατομμύρια χρήστες για λειτουργίες διάφορων API.

Η τρίτη αδυναμία είναι ότι αποφάσισα να κάνω το admin login σε διαφορετικό ανεξάρτητο project χρησιμοποιώντας jsp και servlets. Θα μπορούσε να ήταν στην 2η φάση της εργασίας και απλά να υπάρχει ένας σύνδεσμος για το Admin Login Page.

Τέλος, θα μπορούσε να είναι αισθητικά λίγο καλύτερο το front end και η κύρια σελίδα της εφαρμογής (με καλύτερη χρήση css).

# Πηγές

<https://mvnrepository.com>

<https://start.spring.io>

<https://www.geeksforgeeks.org>

<https://stackoverflow.com>

<https://d3js.org>

<https://leafletjs.com>

<https://tomcat.apache.org>

<https://inspector.swagger.io/builder>

<https://editor.swagger.io/>

<https://www.postman.com>

<https://eclipse-ee4j.github.io/jersey/>

<https://study.com/academy/lesson/what-is-backend-development.html>

<https://blog.back4app.com/best-backends-to-build-an-api-for-your-mobile-app/>