



ΠΑΝΕΠΙΣΤΗΜΙΟ  
ΠΑΤΡΩΝ  
UNIVERSITY OF PATRAS

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ  
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

Χρήση τεχνικών οπτικοποίησης δεδομένων  
χρονοσειρών από αισθητήρες πτηνοτροφικού  
περιβάλλοντος

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΤΡΙΜΙΝΤΖΙΟΥ ΗΛΙΑ

ΑΜ 1047200

Επιβλέπων: Σιούτας Σπυρίδων  
Καθηγητής

Συνεπιβλέπων: Μαχρής Χρήστος  
Αναπληρωτής Καθηγητής

Συνεπιβλέπων: Τσώλης Δημήτριος  
Επίκουρος Καθηγητής

Πάτρα, Νοέμβριος 2022



Πανεπιστήμιο Πατρών  
Πολυτεχνική Σχολή  
Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Copyright ©—All rights reserved Τριμίντζιος Ηλίας, 2022.

Με την επιφύλαξη παντός δικαιώματος.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα.

Το περιεχόμενο αυτής της εργασίας δεν απηχεί απαραίτητα τις απόψεις του Τμήματος, του Επιβλέποντα, ή της επιτροπής που την ενέχρινε.

### Τπεύθυνη Δήλωση

Βεβαιώνω ότι είμαι συγγραφέας αυτής της διπλωματικής εργασίας, και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην διπλωματική εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η διπλωματική εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής του Πανεπιστημίου Πατρών.

(Υπογραφή)

.....  
Τριμίντζιος Ηλίας

# Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω όμερα τον καθηγητή κ. Σιούτα Σπύρο για την ευκαιρία που μου έδωσε να συνεργαστώ μαζί του πάνω σε ένα τόσο ενδιαφέρον και πολλά υποσχόμενα ερευνητικό έργο. Τον αναπληρωτή καθηγητή κ. Μακρή Χρήστο και τον επίκουρο καθηγητή κ. Τσώλη Δημήτριο για την στήριξη και το ενδιαφέρον που έδειξαν κατά την περάτωση της διπλωματικής εργασίας. Επίσης θα ήθελα να δώσω ιδιαίτερες ευχαριστίες στους Γιάνναρο Αναστάσιο και Δόμαλη Γεώργιο για τις ατέλειωτες ώρες στήριξης και απέραντης προσθυμίας τους να με βοηθήσουν στην υλοποίηση της εργασίας με σημειώσεις και υποδείξεις, ανεξαρτήτου ώρας και περιστάσεων. Τέλος να εκφράσω την ευγνωμοσύνη μου στην οικογένεια μου για την συνεχή στήριξη και το κουράγιο που έδειξαν καθ' όλη την διάρκεια φοίτησης μου στο πανεπιστήμιο.

# Περίληψη

Η ανάπτυξη και ο σχεδιασμός μια διαδικτυακής πλατφόρμας που διαχειρίζεται και αναλύει δεδομένα από αισθητήρες πτηνοτροφικού περιβάλλοντος, αποσκοπεί στον εντοπισμό συσχετίσεων μεταξύ των δεδομένων. Η πλατφόρμα αυτή παρέχει στους χρήστες (παραγωγούς έτοιμων ζωοτροφών, κτηνοτρόφους, συνεταιρισμούς) υπηρεσίες που υποστηρίζουν την παρουσίαση των απαραίτητων πληροφοριών σχετικά με τα διάφορα δεδομένα που έχουν εισαχθεί, την παρουσίαση κατάλληλων γραφημάτων με βάση τα χριτήρια του χρήστη και τις απαραίτητες οδηγίες για την κατανόηση και την εξοικείωση του. Η συγκεκριμένη πλατφόρμα που συλλέγει τα δεδομένα τόσο από τους αισθητήρες όσο και χειροκίνητα από τον χρήστη, είναι σε θέση να παρέχει εργαλεία διαχείρισης των δεδομένων που υπάρχουν στο σύστημα και να υποστηρίζει τεχνικές οπτικοποίησης (visualization) των δεδομένων. Τα δεδομένα των αισθητήρων, προτού γίνει το ανέβασμα (upload) στην πλατφόρμα υπόκεινται σε αναλυτική επεξεργασία και προετοιμασία ώστε να γίνουν αποδεκτά από το σύστημα. Τέλος η αρχιτεκτονική της web based πλατφόρμας αποτελείται από δύο βασικές οντότητες:

- α) Τα εργαλεία τα οποία ψάχνουν τη διεπαφή διάδρασης των χρηστών με την πλατφόρμα,
- β) Τις υπηρεσίες οι οποίες δουλεύουν στο παρασκήνιο και περιγράφονται αναλυτικά, διασφαλίζουν τη σωστή λειτουργία της πλατφόρμας.

# **Abstract**

The development and design of an online platform that manages and analyzes data from poultry environment sensors, aims to identify correlations between the data. This platform provides users (ready-to-feed producers, breeders, cooperatives) with services that support the presentation of the necessary information about the various data entered, the presentation of appropriate graphs based on user criteria and the necessary instructions for a better understanding and use of the system. This platform, which collects data both from the sensors and manually by the user, is able to provide tools for managing the data in the system and to support data visualization techniques. The sensor data, before being uploaded to the platform, is subjected to detailed processing and preparation in order to be accepted by the system. Finally the web based platform architecture consists of two main entities:

- a)Tools that provide the user interactive interface with the platform,
- b)Services that run in the background and are described in detail, to ensure the proper operation of the platform.

*Στην οικογένεια μου*

# Περιεχόμενα

Ευχαριστίες	i
Περίληψη	ii
Abstract	iii
Περιεχόμενα	vi
Κατάλογος Πινάκων	vii
<b>1 Εισαγωγή</b>	<b>1</b>
1.1 Η Τεχνολογία στη Πτηνοτροφία . . . . .	1
1.2 Το Phytobiotics στη Πτηνοτροφία . . . . .	1
1.3 Στόχοι Διπλωματικής Εργασίας . . . . .	2
<b>2 Εργαλεία</b>	<b>3</b>
2.1 Docker . . . . .	3
2.1.1 Ορισμός του Docker . . . . .	3
2.1.2 Container . . . . .	3
2.1.3 Image . . . . .	5
2.1.4 Dockerfile - Docker Compose . . . . .	5
2.1.5 Network . . . . .	9
2.1.6 Docker Deployment . . . . .	9
2.2 ElasticSearch . . . . .	10
2.2.1 Τι είναι το Elasticsearch . . . . .	10
2.2.2 Δομή: Έγγραφα & Ευρετήρια . . . . .	11
2.2.3 Mapping . . . . .	11
2.2.4 Dynamic Mapping . . . . .	11
2.2.5 Τύπος Πεδίου Ημερομηνία . . . . .	12
2.2.6 Explicit Mapping . . . . .	16
2.2.7 Τροποποίηση Αντιστοίχησης . . . . .	16
2.3 Kibana . . . . .	18
2.3.1 Ορισμός του Kibana . . . . .	18

2.3.2 Security στα Kibana - Elasticsearch <sup>18</sup> . . . . .	19
2.4 Django . . . . .	19
2.4.1 Τι είναι το Django . . . . .	19
<b>3 Υλοποίηση</b>	<b>21</b>
3.1 Προετοιμασία Δεδομένων (Data Preparation) . . . . .	21
3.1.1 Είδη Δεδομένων . . . . .	21
3.1.2 Data Processing & Upload . . . . .	23
3.1.3 Data Processing . . . . .	23
3.1.4 Ανέβασμα Αρχείων - Δεδομένων . . . . .	30
3.2 Kibana . . . . .	34
3.2.1 Εφαρμογή των χαρακτηριστικών ασφαλείας . . . . .	34
3.2.2 Data Search <sup>3536</sup> . . . . .	43
3.2.3 Dev Tools . . . . .	43
3.2.4 Rest API . . . . .	44
3.2.5 Queries . . . . .	45
3.2.6 Aggregations . . . . .	50
3.3 Django . . . . .	57
3.3.1 Εγκατάσταση Django . . . . .	57
3.3.2 Το Django σε Virtual-Env . . . . .	57
3.3.3 Η μεταφορά του Django σε Docker . . . . .	58
3.3.4 Λειτουργίες της Πλατφόρμας . . . . .	60
3.3.5 Βασικές Δυνατότητες Πλατφόρμας . . . . .	60
3.3.6 Ανέβασμα Αρχείων στο Elasticsearch (Uploading) . . . . .	62
3.3.7 Η οπτικοποίηση των δεδομένων στη πλατφόρμα . . . . .	66
3.3.8 Matplotlib - Seaborn . . . . .	66
3.3.9 Bokeh . . . . .	69
3.3.10 Plotly & Dash . . . . .	71
3.3.11 Kibana Dashboard . . . . .	73
3.3.12 Προβλήματα που παρουσιάστηκαν . . . . .	74
<b>4 Πλοήγηση Στην Πλατφόρμα</b>	<b>76</b>

# Κατάλογος Πινάκων

2.1 Τύποι Πεδίων Δεδομένων . . . . .	12
--------------------------------------	----

# Κεφάλαιο 1

## Εισαγωγή

### 1.1 Η Τεχνολογία στη Πτηνοτροφία

Η τεχνολογική πρόοδος στο τομέα της πτηνοτροφίας έχει αποφέρει τεράστια οφέλη στο πτηνοτρόφο. Η εισαγωγή τεχνολογιών στην καθημερινότητα του, έχει διευκολύνει αισθητά τον εργασιακό του βίο, με αποτέλεσμα τη βελτίωση της ποσότητας και της ποιότητας της παραγωγής του. Παράλληλα όμως με τους πτηνοτρόφους, παρατηρείται ότι και στα ίδια τα ζώα έχουν παρουσιαστεί θετικές αλλαγές, γεγονός που αποδεικνύεται από την ποιότητα τους. Η ανάπτυξη πραγματοποιήθηκε σταδιακά και χρειάστηκαν χρόνια για να υπάρξουν αισθητά αποτελέσματα. Καθημερινά γίνονται προσπάθειες για την επίτευξη νέων στόχων που θα εξελίξουν ακόμη περισσότερο τη πτηνοτροφία. Όμως, παρ όλη την τεχνολογική εξέλιξη, υπάρχουν σοβαρά προβλήματα τα οποία πλήττουν καθημερινά το τομέα αυτό. Αρρώστιες που δεν μπορούν να αντιμετωπιστούν εγκαίρως, κακές αναλογίες σε αντιβιοτικά καθώς και η λαθεμένη επιλογή τοποθεσίας ανάπτυξης ενός πτηνοτροφικού κέντρου αποτελούν μερικά από τα εμπόδια που παρουσιάζονται στην πτηνοτροφία. Συνεπώς, η ιδέα τοποθέτησης αισθητήρων και η χρήση τεχνικών οπτικοποίησης δεδομένων, έχει ως απότερο σκοπό να περιορίσει στο ελάχιστο αν όχι εξ ολοκλήρου οποιοδήποτε νέο εμπόδιο παρουσιαστεί. Αυτό όμως προϋποθέτει και την εξοικείωση των γεωργών σε τέτοιου είδους τεχνολογίες.

### 1.2 To Phytobiotics στη Πτηνοτροφία

Στα πλαίσια του έργου Phytobiotics με θέμα την ανάπτυξη καινοτόμων ζωοτροφών εμπλουτισμένες με βιοδραστικούς παράγοντες από εδώδιμα φυτά και βότανα για αντιμετώπιση των ασθενειών της ζωικής παραγωγής και μείωση της χρήσης αντιβιοτικών, υλοποιείται μια νέα πλατφόρμα με την οποία θα αξιολογούμε της φυσική κατάσταση των ζώων και του χώρου παραγωγής. Η πλατφόρμα καταρχάς θα εφαρμόζει τεχνικές και μεθοδολογίες μηχανικής μάθησης και αναγνώρισης προτύπων με στόχο τον εντοπισμό συσχετίσεων μεταξύ των δεδομένων που παράγει η έρευνα και την εκπαίδευση ενός μοντέλου που θα ταξινομεί τα δεδομένα αυτά. Η πλατφόρμα επίσης θα διαχειρίζεται και θα αναλύει τα δεδομένα και τα αποτελέσματα της έρευνας, έτσι ώστε να παρέχει στους χρήστες τις απαραίτητες πληροφορίες. Για την υλοποίηση

της συγκεκριμένης πλατφόρμας όταν αξιοποιηθούν τεχνικές και τεχνολογίες για την ανάπτυξης διαδικτυακών εφαρμογών τεχνικές μηχανικής μάθησης (machine learning) και ανάλυσης δεδομένων (data analytics) στοχεύοντας στην εξόρυξη νέας γνώσης, στην αναγνώριση νέων συσχετίσεων μεταξύ των δεδομένων (linked data) και κατά συνέπεια τη βελτίωση των μεθοδολογιών εξαγωγής συμπερασμάτων και ερευνητικών αποτελεσμάτων.

### 1.3 Στόχοι Διπλωματικής Εργασίας

Στόχος της παρούσας διπλωματικής είναι η μελέτη, η έρευνα, η περιγραφή και η υλοποίηση της διαδικτυακής πλατφόρμας. Θα πραγματοποιηθεί λεπτομερή ανάλυση όλων τεχνολογιών που όταν χρησιμοποιηθούν για το έργο αυτό τόσο στο back-end (Docker, Python, Curl) όσο και του front-end (Kibana, Elasticsearch). Επιπλέον όταν γίνεται αναφορά στη προετοιμασία των δεδομένων (Data Preparation) που γίνεται μετά την λήψη τους από τους αισθητήρες και πριν το ανέβασμα στην πλατφόρμα καθώς και τα scripts που πρέπει να τρέζουν για να επιτευχθεί η συγκεκριμένη λειτουργία. Επίσης να σημειώσουμε ότι όταν καλύψουμε την σύνδεση και την επικοινωνία ανάμεσα σε back-end και front-end τεχνολογίες που χρησιμοποιούμε και όταν παρουσιάσουμε τις δυνατότητες που μπορεί να εκμεταλλευτεί ο εκάστοτε χρήστης κατά την περιήγηση του στην πλατφόρμα. Τέλος, όταν αναλυθεί ο κλάδος της οπτικοποίησης των δεδομένων (data visualization) ο οποίος εστιάζει στο πώς ο χρήστης όταν είναι σε θέση να ερμηνεύσει τα γραφήματα που παράγονται έπειτα από την διαδικασία του ανεβάσματος (uploading) και η τυχόν επεξεργασία τους.

## Κεφάλαιο 2

# Εργαλεία

Στο κεφάλαιο αυτό γίνεται αναφορά στα εργαλεία που θα χρησιμοποιηθούν για την επίτευξη της υλοποίησης του παρόντος έργου. Όπως προαναφέρθηκε, τα εργαλεία που θα χρησιμοποιηθούν για την ανάπτυξη του έργου χωρίζονται στις κατηγορίες front-end και back-end. Στην πρώτη κατηγορία ανήκουν τα εργαλεία που σχετίζονται με το περιβάλλον διεπαφής του χρήστη (Django και Kibana) ενώ στη δεύτερη ανήκουν τα εργαλεία που εκτελούνται υποστηρικτικά στο παρασκήνιο (Elasticsearch και Docker).

### 2.1 Docker

#### 2.1.1 Ορισμός του Docker

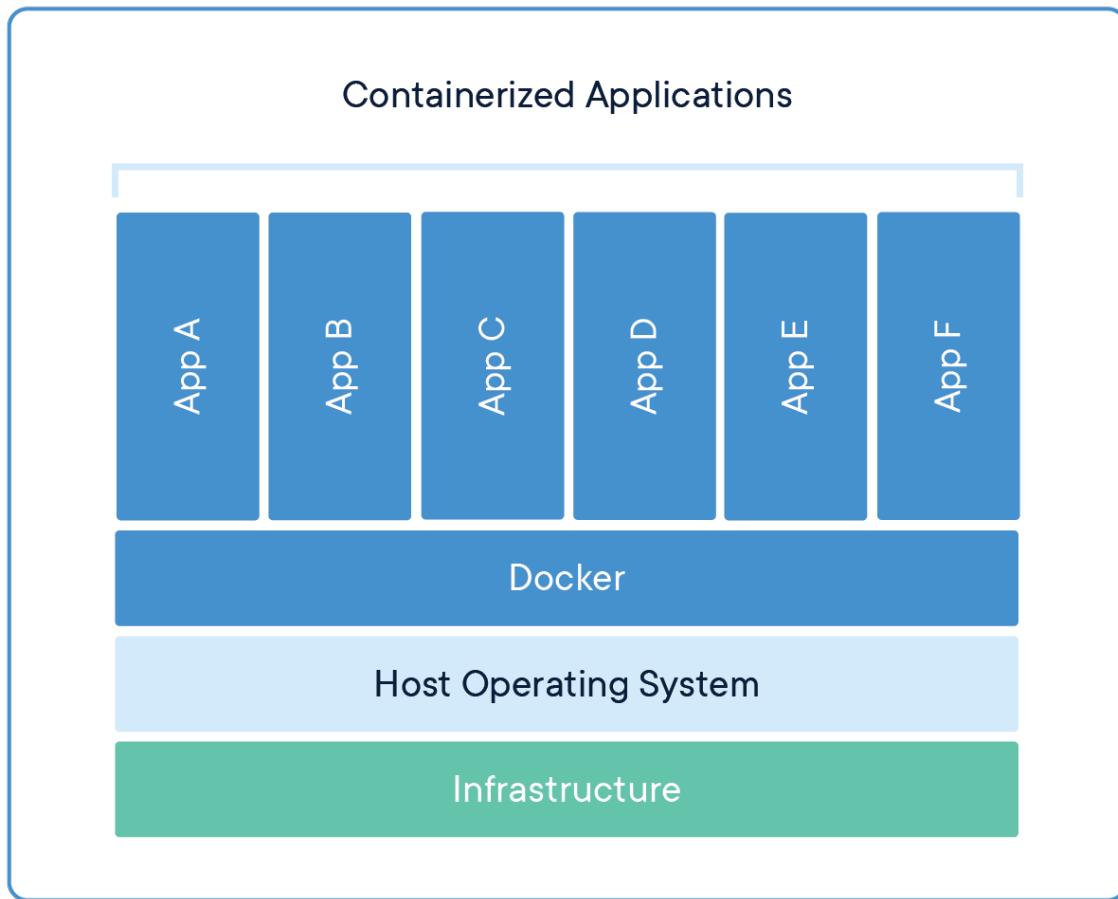
Το Docker<sup>1</sup> είναι μια πλατφόρμα λογισμικού ανοιχτού κώδικα (Open Source) που χρησιμοποιεί Εικονοποίηση Επιπέδου Λειτουργικού Συστήματος (OS-Level Virtualization). Σκοπός του Docker είναι η παροχή ενός περιβάλλοντος απομονωμένης λειτουργίας για τις εφαρμογές σε απομονωμένα πακέτα (Packages) που ονομάζονται Containers έτσι ώστε να μην επηρεάζει η μία την λειτουργία της άλλης. Αφορά κυρίως προγραμματιστές και προχωρημένους χρήστες και επιτρέπει την αυτοματοποίηση της ανάπτυξης εφαρμογών ως φορητών (portable applications) που μπορούν να εκτελούνται στο Cloud ή σε απομονωμένο περιβάλλον. Το Docker μπορεί να εκτελεστεί σε οποιοδήποτε λειτουργικό σύστημα (Windows, Unix) είτε μέσω του περιβάλλοντος διεπαφής (User Interface) γνωστό ως Docker Desktop<sup>2</sup> ή μέσω της γραμμής εντολών (terminal). Τα βασικά τμήματα που απαρτίζουν το Docker είναι: To Container, το Dockerfile, το Image και το Network τα οποία αναλύονται στα επόμενα υποκεφάλαια.

#### 2.1.2 Container

Ένα Container<sup>3</sup> είναι μια τυπική μονάδα λογισμικού που περιέχει κώδικα και όλες τις εξαρτήσεις του, ώστε η εφαρμογή να εκτελείται γρήγορα και αξιόπιστα προσφέροντας δυνατότητες μεταφερσιμότητας (portability) από το ένα σύστημα στο άλλο αγνοώντας το λειτουργικό σύστημα στο οποίο αυτό εκτελείται. Πολλαπλά Containers μπορούν να εκτελούνται ταυτόχρονα στον ίδιο υπολογιστή και να μοιράζονται τον πυρήνα του λειτουργικού συστήματος, καθένα

από τα οποία εκτελείται ως απομονωμένη διεργασία. Ένα Docker Container χωρίζεται σε 3 υποκατηγορίες:

- Standard:** Το βιομηχανικό πρότυπο για Containers, ώστε να μπορούν να είναι φορητά οπουδήποτε.
- Lightweight:** Τα Containers μοιράζονται τον πυρήνα του λειτουργικού συστήματος του μηχανήματος και επομένως δεν απαιτούν λειτουργικό σύστημα ανά εφαρμογή, οδηγώντας σε υψηλότερη απόδοση διακομιστή και μειώνοντας το κόστος διακομιστή και άδειας χρήσης.
- Secure:** Οι εφαρμογές είναι πιο ασφαλείς σε Containers και το Docker παρέχει τις ισχυρότερες προεπιλεγμένες δυνατότητες απομόνωσης.



Σχήμα 2.1: Η Δομή ενός Container

Η δημιουργία ενός Container είναι εφικτή από τη γραμμή εντολών με την χρήση ενός συνόλου εντολών ενώ η προβολή τους είναι δυνατή και από την εφαρμογή Docker Desktop. Οι βασικότερες εντολές είναι οι ακόλουθες:

- **docker run -d docker.elastic.co/elasticsearch/elsaticsearch:8.0.0**

Η εντολή αυτή δημιουργεί και ξεκινά την λειτουργία ενός νέου Container από ένα ήδη υπάρχων docker image. Η επέκταση -d εκτελεί το container στο background και το image που δίνεται είναι το κομμάτι docker.elastic.co/elasticsearch/elsaticsearch:8.0.0 που πρόκειται για ενα επίσημο dockerfile μιας εφαρμογής που χρησιμοποιούμε στα πλαίσια του έργου.

```
C:\Users\user\Desktop>docker run -d docker.elastic.co/elasticsearch/elsaticsearch:8.0.0
42357fed2c8dd01c1a2b9103ba002c22a7b417fb1865cd76324de67e2dc707e
```

Σχήμα 2.2: Το αποτέλεσμα της εντολής docker run

- **docker ps**

Η συγκεκριμένη εντολή επιστρέφει όλες τις πληροφορίες σχετικά με ένα Container που είναι σε λειτουργία, από το image που έχει χρησιμοποιηθεί, πότε δημιουργήθηκε μέχρι και το όνομα που έχει οριστεί από τον χρήστη.

```
C:\Users\user\Desktop>docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS      PORTS          NAMES
F84eebb3ced4  docker.elastic.co/kibana/kibana:8.2.0   "/bin/tini -- /usr/l..."  2 weeks ago  Up 6 seconds  0.0.0.0:5601->5601/tcp   kibana01
769ebb857197  iltrimmer/project-ph:django1  "python manage.py ru..."  2 weeks ago  Up 5 seconds  0.0.0.0:8000->8000/tcp   django01
12e12cd1caa2  docker.elastic.co/elasticsearch/elasticsearch:8.2.0  "/bin/tini -- /usr/l..."  2 weeks ago  Up 6 seconds  0.0.0.0:9200->9200/tcp, 9300/tcp   elastic01
```

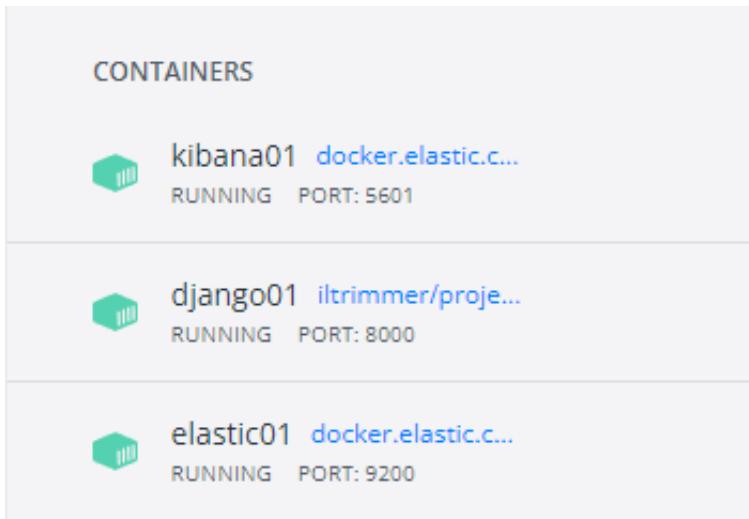
Σχήμα 2.3: Το αποτέλεσμα της εντολής docker ps

## 2.1.3 Image

Μια Docker εικόνα (Image)<sup>4</sup> είναι ένα αυτόνομο, εκτελέσιμο πακέτο λογισμικού που περιλαμβάνει όλα όσα χρειάζονται για την εκτέλεση μιας εφαρμογής: κώδικας, χρόνος εκτέλεσης, εργαλεία συστήματος, βιβλιοθήκες συστήματος και ρυθμίσεις. Ένα Docker Image γίνεται Container κατά την διάρκεια εκτέλεσης του αρχείου. Πρόκειται για ένα τύπο αρχείου που σπάνια υπόκειται σε επεξεργασία από τον χρήστη και εμπεριέχει επίσημο, αναγνωρισμένο κώδικα από την εκάστοτε εταιρία λογισμικού που αναζητά ο χρήστης για την εφαρμογή του.

## 2.1.4 Dockerfile - Docker Compose

Το Dockerfile<sup>5</sup> είναι ένα έγγραφο κειμένου που περιέχει όλες τις εντολές που ένας χρήστης θα μπορούσε να καλέσει στη γραμμή εντολών για να συναρμολογήσει μια εικόνα (Image) δημιουργώντας μια αυτοματοποιημένη έκδοση που εκτελεί πολλές εντολές γραμμής εντολών διαδοχικά. Ένα Dockerfile για να μπορεί να χρησιμοποιηθεί, πρέπει πρώτα να «χτιστεί» με



Σχήμα 2.4: Τα Containers μέσα από την εφαρμογή Docker Desktop

τη βοήθεια της εντολής ”build”. Επιπλέον, το Dockerfile πρέπει να ωκελουθεί ένα σύνολο οδηγιών και να περιέχει ορισμένους τύπους εντολών. Παραδοσιακά, το Dockerfile βρίσκεται στη ρίζα του περιβάλλοντος. Το Docker - Compose<sup>6</sup> είναι ένα εργαλείο για τον καθορισμό και την εκτέλεση πολλαπλών Docker Container. Με το Compose, χρησιμοποιείται ένα αρχείο τύπου yml<sup>7</sup> για να διαμορφωθούν οι υπηρεσίες της εφαρμογής. Στη συνέχεια, με μία μόνο εντολή, δημιουργείται και ξεκινούν όλες οι υπηρεσίες. Η χρήση του Docker - Compose είναι βασικά μια διαδικασία τριών βημάτων:

1. Καθορισμός του περιβάλλοντος της εφαρμογής με ένα Dockerfile ώστε να μπορεί να αναπαραχθεί οπουδήποτε.
2. Καθορισμός των υπηρεσιών που απαρτίζουν την εφαρμογή στο docker-compose.yml, ώστε να μπορούν να εκτελούνται μαζί σε ένα απομονωμένο περιβάλλον.
3. Εκτέλεση της εντολής ”docker-compose-up -d” και ξεκινά ολόκληρη η εφαρμογή ή εναλλακτικά εκτελούμε την εντολή χρησιμοποιώντας το δυαδικό docker-compose αρχείο που βρίσκεται μέσα στο περιβάλλον του container. Στα πλαίσια του έργου Phytobiotics για την υλοποίηση της πλατφόρμας χρησιμοποιούνται τρία είδη λογισμικού μέσω της τεχνολογίας του Docker. Η επικοινωνία των τεχνολογιών αυτών γίνεται με την χρήση ενός docker-compose.yml αρχείου το οποίο αναλύεται παραχάτω.

```
version: "2"
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.2.0
    container_name: elastic01
    restart: "no"
```

```
environment:
  - node.name=node1
  - xpack.security.enabled=true
  - discovery.type=single-node
  - bootstrap.memory_lock=true
  - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
  - xpack.ml.enabled=true
ulimits:
  memlock:
    soft: -1
    hard: -1
 nofile:
    soft: 65536
    hard: 65536
cap_add:
  - IPC_LOCK
volumes:
  - elasticsearch-data:/usr/share/elasticsearch/data
  - shared:/usr/share/elasticsearch/shared
ports:
  - 9200:9200
networks:
  - es-network

kibana:
  container_name: kibana01
  image: docker.elastic.co/kibana/kibana:8.2.0
  restart: "no"
  environment:
    - ELASTICSEARCH_HOSTS=https://elasticsearch:9200
  ports:
    - 5601:5601
  depends_on:
    - elasticsearch
  volumes:
    - shared:/usr/share/elasticsearch/shared
networks:
  - es-network

django:
  container_name: django01
  image: iltrimmer/project-phy:django1
```

```

restart: "no"
ports:
  - 8000:8000
depends_on:
  - elasticsearch
networks:
  - es-network
volumes:
  elasticsearch-data:
    driver: local
  shared:
    driver: local
networks:
  es-network:
    driver: bridge

```

Η δομή του συγκεκριμένου docker-compose.yml αρχείου χωρίζεται σε 4 κατηγορίες: "version", "services", "volumes", "networks".

- **version:** Αφορά την έκδοση του yml αρχείου και το πως θα αναγνωρίζεται από το Docker. Νεότερες εκδόσεις σημαίνουν και περισσότερες λειτουργίες αλλά απαραίτητη προϋπόθεση είναι να συμβαδίζει με την έκδοση του Docker.
- **services:** Στη κατηγορία αυτή περιέχονται όλα τα στοιχεία σχετικά με τα Containers που θα χρησιμοποιήσει ο χρήστης για να χτίσει την εφαρμογή του. Όπως φαίνεται και στο σχήμα 2.4 χρησιμοποιούνται τρία Containers (elastic01, kibana01, django01). Η ονομασία τους προκύπτει από την γραμμή κώδικα container\_name: που υπάρχει σε κάθε section των services. Επιπλέον με την εντολή image, καθορίζεται ποια θα είναι η εικόνα που θα χρησιμοποιηθεί για την δημιουργία του συγκεκριμένου container. Είναι εφικτό να χρησιμοποιηθεί μια επίσημη εικόνα ή ένα Dockerfile αρχείο προσαρμοσμένο από τον εκάστοτε χρήστη. Για τη παρούσα διπλωματική, γίνεται χρήση των επίσημων εικόνων των τεχνολογιών που απαιτούνται. Εξαίρεση αποτελεί η περίπτωση του django η οποία βασίζεται σε custom Dockerfile (βλ. <https://github.com/GvasTheGreek/Thesis-Phytobiotics-Repo/blob/master/Dockerfile/docker-compose.yml> λόγω του γεγονότος ότι η διαδικτυακή εφαρμογή (web application) που δημιουργείται χρησιμοποιεί βιβλιοθήκες (libraries) και πρόσθετα (extensions) τα οποία δεν προσφέρονται με το επίσημο Dockerfile. Εν συνεχείᾳ στη εντολή environment: τοποθετούνται οι ρυθμίσεις και τα χαρακτηριστικά που χρειάζεται το εκάστοτε Container να έχει. Για παράδειγμα, οι εντολές -xpack.security.enabled=true και -xpack.ml.enabled=true ενεργοποιούν την δυνατότητα προσθήκης ασφάλειας και machine Learning αντίστοιχα στο Container του Elasticsearch, ενώ από την άλλη η εντολή ELASTICSEARCH\_HOSTS=https://elasticsearch:9200 είναι υψηστης σημασίας καθώς γίνεται εφικτή η σύνδεση των δύο Containers.

- **volumes:** Η προσθήκη αυτή γίνεται για να μπορέσουμε να ορίσουμε το μονοπάτι που θα αποθηκευτούν τα δεδομένα μέσα στο Container.
- **networks:** Επιτρέπει και καθορίζει τον τρόπο που επικοινωνούν τα Containers

## 2.1.5 Network

Το δίκτυο του Docker ή αλλιώς το Docker Network<sup>8</sup> είναι πρακτικά η λειτουργία που επιτρέπει την σύνδεση μεταξύ των Containers. Η τοποθέτηση τους στο docker-compose.yml αρχείο είναι απαραίτητη και καθορίζει με ποιο τρόπο επικοινωνούν τα Containers μεταξύ τους. Η λειτουργικότητα του δικτύου εξαρτάται από τον οδηγό (driver) που θα οριστεί στο yml αρχείο.

- **Γέφυρα (Bridge):** Η πιο απλή μορφή δικτύου που χρησιμοποιείται σε ανεξάρτητες εφαρμογές οι οποίες πρέπει να επικοινωνήσουν μεταξύ τους. Το είδος αυτό χρησιμοποιείται στα πλαίσια της διπλωματικής. Αναλυτικότερα, ένα δίκτυο γέφυρας χρησιμοποιεί μια γέφυρα λογισμικού (software bridge) που επιτρέπει στα κοντέινερ που είναι συνδεδεμένα στο ίδιο δίκτυο γέφυρας να επικοινωνούν, ενώ παρέχει απομόνωση από κοντέινερ που δεν είναι συνδεδεμένα σε αυτό το δίκτυο γέφυρας. Το πρόγραμμα οδήγησης Docker bridge εγκαθιστά αυτόματα κανόνες στον κεντρικό υπολογιστή, έτσι ώστε τα κοντέινερ σε διαφορετικά δίκτυα γεφυρών να μην μπορούν να επικοινωνούν απευθείας μεταξύ τους. Μια γέφυρα μπορεί να είναι μια συσκευή υλικού ή μια συσκευή λογισμικού που εκτελείται μέσα στον πυρήνα μιας κεντρικής μηχανής.
- **Host:** Μια πιο σύνθετη μορφή δικτύου στην οποία ο χρήστης δίνει ο ίδιος το αναγνωριστικό ID στο Container, με αποτέλεσμα η εφαρμογή του Container να τρέχει στη πύλη που έχει ορίσει ο χρήστης. Χρησιμεύει για επίτευξη μέγιστης απόδοσης με δυνατότητα αφομοίωσης πολλών πυλών δικτύου Ports. Η χρήση της επιλογής Host, είναι εφικτή μόνο σε λειτουργικά Linux.
- **none:** Η χρήση του σε κάποιο Container καθιστά αδύνατη την επικοινωνία με άλλα Containers.

## 2.1.6 Docker Deployment

Αφού έχει πραγματοποιηθεί η προεργασία που απαιτείται από το χρήστη για να φέρει εις πέρας την αρχική εγκατάσταση και ρύθμιση και παραμετροποίηση των τεχνολογιών μέσω Docker, απομένει το Deployment δηλαδή η διαδικασία για να ενεργοποιηθεί και να τρέξει ομαλά το πρόγραμμα. Στα πλαίσια της παρούσας διπλωματικής εργασίας, γίνεται χρήση του docker-compose.yml αρχείου για το Deployment. Η χρήση του αρχείου αυτού κάνει πιο εύκολη την όλη διαδικασία καθώς επιτρέπει στον χρήστη με λίγες εντολές να παράξει το επιθυμητό αποτέλεσμα χωρίς δυσκολία λόγω της πολυπλοκότητας που επιφέρει η χρήση πολλών Dockerfile ταυτόχρονα. Ακολουθούν οι εντολές που χρησιμοποιούνται για την συγκεκριμένη εργασία:

1. **docker compose up -d**: Η παρούσα εντολή δημιουργεί και ξεκινά τα Containers που υπάρχουν στο services section του docker-compose.yml. Η εντολή δεν παράγει αποτέλεσμα αν δεν υπάρχει ένα σωστά δομημένο yml αρχείο όπως αυτό που υπάρχει σε προηγούμενη παράγραφο. Αυτό που κάνει ιδιαίτερα εύχρηστη την μέθοδο του docker-compose αλλά και την εντολή αυτή είναι ότι δεν χρειάζεται να προηγηθεί η δημιουργία των images. Εφόσον η αναφορά σε επίσημη εικόνα έχει γίνει σωστά και το image δεν υπάρχει στο τοπικό περιβάλλον η εντολή αυτή πριν την δημιουργία των Containers κάνει λήψη των εικόνων.
2. **docker compose start**: Όπως προδιαθέτει και η εντολή η ίδια, αυτή ξεκινά την λειτουργία των Containers. Απαραίτητη συνθήκη είναι να υπάρχουν Containers καθώς, σε αντίθεση με τη προηγούμενη μέθοδο, δεν δημιουργεί καινούργια.
3. **docker compose stop**: Σταματά την λειτουργία των Containers.

## 2.2 ElasticSearch

### 2.2.1 Τι είναι το Elasticsearch

Το Elasticsearch<sup>910</sup> είναι μια κατανεμημένη μηχανή αναζήτησης και ανάλυσης. Είναι το μέρος στο οποίο πραγματοποιείται η διαδικασία της ευρετηρίασης (Indexing), της αναζήτησης και της ανάλυσης. Το Elasticsearch παρέχει αναζήτηση και αναλυτικά στοιχεία σχεδόν σε πραγματικό χρόνο για όλους τους τύπους δεδομένων. Είτε πρόκειται για δομημένο είτε μη δομημένο κείμενο, αριθμητικά δεδομένα ή γεωχωρικά δεδομένα (geospatial data), το Elasticsearch μπορεί να τα αποθηκεύσει και να τα ευρετηριάσει αποτελεσματικά με τρόπο που να υποστηρίζει γρήγορες αναζητήσεις. Μπορεί να προχωρήσει πολύ πέρα από την απλή ανάζηση δεδομένων και να συγχεντρώσει πληροφορίες για να ανακαλύψει τάσεις και μοτίβα στα δεδομένα. Καθώς αυξάνεται ο όγκος των δεδομένων και των ερωτήσεων, η κατανεμημένη φύση του Elasticsearch επιτρέπει την διαδικασία να αναπτύσσεται απρόσκοπτα παράλληλα με αυτό. Προσφέρει ταχύτητα και διαλλακτικότητα στον χειρισμό των δεδομένων και ένα μεγάλο εύρος χρήσεων:

1. Δυνατότητα προσθήκης διεπαφής (Interface) σε τρίτες εφαρμογές και ιστοσελίδες.
2. Αποθήκευση και ανάλυση αρχείων καταγραφής, μετρήσεων και δεδομένων συμβάντων ασφαλείας.
3. Χρήση μηχανικής μάθησης για την αυτόματη μοντελοποίηση της συμπεριφοράς των δεδομένων.
4. Ανάλυση χωρικών πληροφοριών (spatial information) με χρήση του Elasticsearch ως γεωγραφικό σύστημα πληροφοριών (GIS).

### 2.2.2 Δομή: Έγγραφα & Ευρετήρια

Το Elasticsearch είναι ένα κατανεμημένο σύστημα αποθήκευσης εγγράφων. Αντί να αποθηκεύει τις πληροφορίες χωρίς καμία ταξινόμηση και ως σειρές στηλών δεδομένων το Elasticsearch αποθηκεύει σύνθετες δομές δεδομένων που έχουν σειριοποιηθεί ως έγγραφα JSON. Όταν υπάρχουν πολλοί κόμβοι σε ένα σύμπλεγμα (nodes in a cluster), τα αποθηκευμένα έγγραφα διανέμονται σε όλο το σύμπλεγμα (cluster) και είναι άμεσα προσβάσιμα από οποιονδήποτε κόμβο.

Όταν ένα έγγραφο αποθηκεύεται, καταχωρείται σε ευρετήριο (Index) και μπορεί να αναζητηθεί πλήρως σε σχεδόν πραγματικό χρόνο ‘εντός 1 δευτερολέπτου’ (near real time)<sup>11</sup>. Το Elasticsearch χρησιμοποιεί μια δομή δεδομένων που ονομάζεται ανεστραφμένο ευρετήριο<sup>1213</sup> που υποστηρίζει πολύ γρήγορες αναζητήσεις πλήρους κειμένου. Ένα ανεστραφμένο ευρετήριο παραθέτει κάθε μοναδική λέξη που εμφανίζεται σε οποιοδήποτε έγγραφο και προσδιορίζει όλα τα έγγραφα στα οποία εμφανίζεται κάθε λέξη.

Ένα ευρετήριο μπορεί να θεωρηθεί ως μια βελτιστοποιημένη συλλογή εγγράφων και κάθε έγγραφο είναι μια συλλογή πεδίων, τα οποία είναι τα ζεύγη κλειδιών-τιμών που περιέχουν τα δεδομένα σας. Από προεπιλογή, το Elasticsearch ευρετηριάζει όλα τα δεδομένα σε κάθε πεδίο και κάθε πεδίο με ευρετήριο έχει μια αποκλειστική, βελτιστοποιημένη δομή δεδομένων.

Πρόκειται για μια τεχνολογία με ευρεία γκάμα δυνατοτήτων. Στα πλαίσια όμως τόσο της διπλωματικής εργασίας όσο και του project Phytobiotics δεν γίνεται χρήση όλων των εργαλείων του. Οι παράγραφοι που ακολουθούν επικεντρώνονται στις λειτουργίες που χρησιμοποιούνται για το πέρας του έργου.

### 2.2.3 Mapping

Η αντιστοίχηση (Mapping) είναι η διαδικασία καθορισμού του τρόπου με τον οποίο ένα έγγραφο και τα πεδία που περιέχει, αποθηκεύονται και ευρετηριάζονται (Indexed). Κάθε έγγραφο είναι μια συλλογή πεδίων, που το καθένα έχει τον δικό του τύπο δεδομένων. Κατά την αντιστοίχηση των δεδομένων μας, δημιουργούμε έναν ορισμό αντιστοίχησης, ο οποίος περιέχει μια λίστα πεδίων που σχετίζονται με το έγγραφο. Χωρίζεται σε δύο κατηγορίες προσέγγισης, Dynamic Mapping και Explicit Mapping.

### 2.2.4 Dynamic Mapping

Η περίπτωση της δυναμικής αντιστοίχησης (Dynamic Mapping) είναι ιδανική για νέους χρήστες στο περιβάλλον του Elasticsearch. Το Elasticsearch προσθέτει αυτόματα νέα πεδία, μόνο με την ευρετηρίαση ενός εγγράφου, αλλά δίνει και την δυνατότητα τροποποίησης του αυτόματα δημιουργηθέντος Mapping, διαδικασία που μπορεί να γίνει μέσω της διεπαφής του Kibana είτε μέσω της γραμμής εντολών χρησιμοποιώντας την εντολή curl. Όταν πραγματοποιείται η διαδικασία της δυναμικής αντιστοίχησης, οι τύποι των πεδίων δεδομένων μπορούν

να πάρουν μια συγκεκριμένη τιμή, βασισμένη στο είδος του δεδομένου που έχει διαβαστεί κατά το ανέβασμα του json αρχείου στην πλατφόρμα. Ο πίνακας που ακολουθεί διατυπώνει τους τύπους δεδομένων που αναγνωρίζει το Elasticsearch κατά την αναγνώριση των δεδομένων του αρχείου.

<b>JSON data type</b>	<b>Elasticsearch data type</b>
null	No field added
true or false	boolean
double	float
long	long
object	object
array	Depends on the first non-null value in the array
string that passes date detection	date
string that passes numeric detection	float or long
string that doesn't pass date detection	text with a .keyword sub-field
string that doesn't pass numeric detection	text with a .keyword sub-field

Table 2.1: Τύποι Πεδίων Δεδομένων

Το πόρισμα που προκύπτει από το πίνακα 2.1, είναι ότι σε περίπτωση που το json αρχείο περιέχει αριθμούς ο τύπος τους κατά την ευρετηρίαση τους είναι ζεχανθαρος, σε αντίθεση με την αναγνώριση αλφαριθμητικών, όπου το Elasticsearch θα πρέπει να πραγματοποιήσει περαιτέρω ελέγχους.

### 2.2.5 Τύπος Πεδίου Ημερομηνία

Αξιοσημείωτη είναι η περίπτωση της ημερομηνίας, για την οποία εξαιτίας του γεγονότος ότι τα json αρχεία δεν έχουν στην δομή τους τύπο ‘ημερομηνίας’, το Elasticsearch αναγνωρίζει την παρουσία τέτοιου τύπου πεδίου με τους παρακάτω τρόπους:

- Συμβολοσειρές που περιέχουν μορφοποιημένες ημερομηνίες.  
πχ. ‘2011-01-01’ και ‘2011/11/11 11:11:11’
- Έναν αριθμό που αντιπροσωπεύει χιλιοστά του δευτερολέπτου από την epoch<sup>14</sup> στιγμή ή αλλιώς την ημερομηνία Πέμπτη 1 Ιανουάριου 1970 00:00:00 UTC (Epoch θεωρείται η ημερομηνία την οποία χαρακτηρίζει το Unix ως απαρχή των πάντων και τη χρησιμοποιεί για να υπολογίζει τη ‘χρονική απόσταση’ από κάποια άλλη ημερομηνία).
- Έναν αριθμό που αντιπροσωπεύει δευτερόλεπτα από την epoch στιγμή.

Έχοντας λοιπόν ως βάση τις συγκεκριμένες μορφές ημερομηνίας μπορούμε να επεξεργαστώ καταλλήλως το συγκεκριμένο πεδίο του mapping ενός ευρετηρίου ή να τροποποιήσω το ίδιο το αρχείο πριν ξεκινήσει το Upload με τα απαραίτητα Script όπως αναλύεται στο κεφάλαιο 3.

```
PUT my-index-01
{
  "mappings": {
    "properties": {
      "date": {
        "type": "date",
        "format": "yyyy-MM-dd HH:mm:ss"
      }
    }
  }
}
```

Κώδικας 2.1: Mapping του πεδίου ημερομηνίας

Στο παράδειγμα του κώδικα 2.1 γίνεται επεξεργασία του πεδίου της ημερομηνίας του Mapping του ευρετηρίου με όνομα my-index-01. Του δίνουμε την μορφή ”yyyy-MM-dd HH:mm:ss” που αποτελεί μια από τις αποδεκτές από το elasticsearch μορφές. Στη συνέχεια παρουσιάζεται και επεξηγείται η δομή του Mapping ενός αρχείου Json που χρησιμοποιείται στα πλαίσια της εργασίας.

```
{
{
  "Timestamp": "2022-03-02 09:53:46",
  "Entity Name": "Phytobiotics-RTH1",
  "CO2 (ppm)": 2319,
  "NH3 (ppm)": 2,
  "ambient_light (lux)": 30,
  "ambient_noise (dB)": 71,
  "humidity (%)": 80,
  "temperature (°C)": 21.38
},
{
  "Timestamp": "2022-03-02 09:58:56",
  "Entity Name": "Phytobiotics-RTH1",
  "CO2 (ppm)": 2285,
  "NH3 (ppm)": 2,
  "ambient_light (lux)": 30,
  "ambient_noise (dB)": 72,
  "humidity (%)": 79,
  "temperature (°C)": 21.23
},
{
  "Timestamp": "2022-03-02 10:04:05",
  "Entity Name": "Phytobiotics-RTH1",
  "CO2 (ppm)": 2300,
  "NH3 (ppm)": 2,
  "ambient_light (lux)": 30,
  "ambient_noise (dB)": 73,
  "humidity (%)": 81,
  "temperature (°C)": 21.35
}
}
```

```

"Entity Name": "Phytobiotics-RTH1",
"C02 (ppm)": 2304,
"NH3 (ppm)": 2,
"ambient_light (lux)": 30,
"ambient_noise (dB)": 72,
"humidity (%)": 80,
"temperature (°C)": 21.23
},
{
"Timestamp": "2022-03-02 10:43:26",
"Entity Name": "Phytobiotics-RTH1",
"C02 (ppm)": 1941, "NH3 (ppm)": 2,
"ambient_light (lux)": 30,
"ambient_noise (dB)": 71,
"humidity (%)": 78,
"temperature (°C)": 20.79
}
}

```

Κώδικας 2.2: Δείγμα Json για την διαδικασία Mapping

```

1 {
2   "my-index" : {
3     "mappings" : {
4       "properties" : {
5         "C02 (ppm)" : {
6           "type" : "long"
7         },
8         "Entity Name" : {
9           "type" : "text",
10          "fields" : {
11            "keyword" : {
12              "type" : "keyword",
13              "ignore_above" : 256
14            }
15          }
16        },
17        "NH3 (ppm)" : {
18          "type" : "long"
19        },
20        "Timestamp" : {
21          "type" : "text",
22          "fields" : {
23            "keyword" : {
24              "type" : "keyword",
25              "ignore_above" : 256
26            }
27          }
28        }
29      }
30    }
31  }
32}

```

```

27     }
28   },
29   "ambient_light (lux)" : {
30     "type" : "long"
31   },
32   "ambient_noise (dB)" : {
33     "type" : "long"
34   },
35   "humidity (%)" : {
36     "type" : "long"
37   },
38   "temperature (°C)" : {
39     "type" : "float"
40   }
41 }
42 }
43 }
44 }

```

Κώδικας 2.3: Το αποτέλεσμα του Mapping

Ο κώδικας 2.3 αποτελεί την αντιστοίχηση (Mapping) των δεδομένων του Json αρχείου, του οποίου ένα δείγμα παρουσιάζεται στον κώδικα 2.2. Πρόκειται για το αποτέλεσμα της αυτόματης αντιστοίχησης (Dynamic Mapping) που πραγματοποιεί το Elasticsearch αφού πραγματοποιηθεί το ανέβασμα στη πλατφόρμα του εκάστοτε αρχείου.

Αναλυτικότερα:

- (Γραμμές 3-5):** Παρατηρείται ότι η δομή του Mapping ξεκινά πρώτα με την αναφορά του ευρετηρίου που αντιστοιχεί η συγκεκριμένη αντιστοίχηση και έπειτα γίνεται αναφορά στους τύπους δεδομένων και στις ιδιότητες τους ("mappings" , "properties").
- (Γραμμές 6-40):** Γίνεται η αντιστοίχηση του κάθε τύπου δεδομένου που υπάρχει στο αρχείο μας με τον τύπο που ανατίθεται από το Elasticsearch. Είναι ξεκάθαρο ότι στα παρακάτω πεδία η αντιστοίχηση έχει πραγματοποιηθεί επιτυχώς.

- \* "CO2 (ppm)" → long
- \* "NH3 (ppm)" → long
- \* "ambient\_light (lux)" → long
- \* "ambient\_noise (dB)" → long
- \* "humidity (%)" → long
- \* "temperature (°C)" → float

Σε αντίθεση τα πεδία "Timestamp" και "Entity Name" έχουν αντιστοιχηθεί ως "Text with a .keyword sub-field", τίτλος που υποδηλώνει ότι επρόκειτο για συμβολοσειρές που δεν έχουν περάσει την διαδικασία αναγνώρισης. Αυτό οφείλεται στο γεγονός ότι η μορφή των δεδομένων στο json αρχείο δεν συνάδει με την μορφή που αναγνωρίζει το Elasticsearch.

Καταλήγοντας παράγεται το συμπέρασμα ότι το αποτέλεσμα της αυτόματης αντιστοίχησης (Dynamic Mapping) σε κάποια από τα πεδία να μην είναι αυτό που επιδιώκει ο χρήστης. Για τον λόγο αυτό, υπάρχει και η εναλλακτική προσέγγιση της ‘σαφής αντιστοίχησης’ (Explicit Mapping).

### 2.2.6 Explicit Mapping

Η περίπτωση της ‘σαφής αντιστοίχησης’ ή (Explicit Mapping) επιτρέπει στον χρήστη να καθορίσει ο ίδιος την αντιστοίχηση και τα πεδία της. Του δίνεται η δυνατότητα να επιλέξει ποια πεδία θα αναγνωριστούν ως συμβολοσειρές και ποια αριθμητικά πεδία θα αναγνωριστούν ως αριθμοί, ημερομηνίες ή γεωγραφικές τοποθεσίες. Επιπλέον, είναι εφικτός ο καθορισμός της μορφής της ημερομηνίας που θα σπεύσει να αναγνωρίσει το Elasticsearch, μια δυνατότητα που διευκολύνει κυρίως το χρόνο εκτέλεσης και τους πόρους που δεσμεύονται μιας και τα json αρχεία δεν παρέχουν τύπο ημερομηνίας και κάνουν τη δουλειά του Elasticsearch πιο δύσκολη.

Αναλυτικότερα, το Explicit Mapping αποδεικνύεται μια διαδικασία πιο απαιτητική σε σχέση με το Dynamic Mapping καθώς στην περίπτωση αυτή η αντιστοίχηση πρέπει να γίνει χειροκίνητα και αυτό είναι εφικτό με δύο τρόπους:

- **Δημιουργία του mapping πριν το ανέβασμα των δεδομένων.** Στα πλαίσια της δημιουργίας του ευρετηρίου που θα περαστούν τα δεδομένα, υλοποιείται και το custom mapping για τα δεδομένα αυτά. Χρονικά η στιγμή που δημιουργείται στο Elasticsearch το Mapping είναι αμέσως μετά την δημιουργία του ευρετηρίου και πριν το ανέβασμα των δεδομένων. Χαρακτηριστικό παράδειγμα ο κώδικας 3.9.
- **Δημιουργία του mapping μέσω διεπαφής Kibana.** Το Mapping μπορεί να φτιαχτεί μέσω του Kibana (κεφ. 3) ακολουθώντας τους κανόνες που ισχύουν για τη δημιουργία Mapping και φυσικά να υπάρχει ήδη το ευρετήριο στο οποίο αναφέρεται. Να τονιστεί όμως ότι σε περίπτωση δημιουργίας λαθεμένου Mapping που δεν συνάδει με τα δεδομένα, θα προκαλέσει την αδυναμία ανεβάσματος των δεδομένων στο Elasticsearch.

### 2.2.7 Τροποποίηση Αντιστοίχησης

Εκτός από τις υποστηριζόμενες παραμέτρους, η τροποποίηση της αντιστοίχησης (mapping) ή του τύπου ενός πεδίου του δεν συνιστάται. Η αλλαγή ενός πεδίου μπορεί να προκαλέσει την διαγραφή ή την ακύρωση δεδομένων που έχουν ευρετηριαστεί. Σε περίπτωση που είναι απαραίτητη η αλλαγή της αντιστοίχησης, τότε επαναλαμβάνεται όλη η διαδικασία από την αρχή η οποία εμπεριέχει το ανέβασμα δεδομένων στη πλατφόρμα σε νέο ευρετήριο με τη σωστή αντιστοίχηση.

Ακολουθεί το αποτέλεσμα που προκύπτει έπειτα από την εφαρμογή του explicit mapping.

```

1  {
2      "the-name-of-the-index" : {
3          "mappings": {
4              "properties": {
5                  "@timestamp": {
6                      "type": "date"
7                  },
8                  "CO2 (ppm)": {
9                      "type": "long"
10                 },
11                  "Entity Name": {
12                      "type": "keyword"
13                  },
14                  "NH3 (ppm)": {
15                      "type": "long"
16                  },
17                  "Timestamp": {
18                      "type": "date",
19                      "format" : "yyyy-MM-dd HH:mm:ss"
20                  },
21                  "ambient_light (lux)": {
22                      "type": "long"
23                  },
24                  "ambient_noise (dB)": {
25                      "type": "long"
26                  },
27                  "humidity (%)": {
28                      "type": "long"
29                  },
30                  "temperature (°C)": {
31                      "type": "double"
32                  }
33              }
34          }
35      }
36  }

```

Κώδικας 2.4: Αποτέλεσμα της διαδικασίας Explicit Mapping

Συγκριτικά με τον κώδικα 2.3 που εφαρμόζεται το Dynamic Mapping, σε αυτή την περίπτωση παρατηρούνται διαφορές στα πεδία "Timestamp" και "Entity Name". Αρχικά το Timestamp όντας ανέφικτο για το Elasticsearch να αναγνωρίσει τη μορφή των δεδομένων, στο explicit mapping γίνεται σαφής ορισμός. Αρχικά συμπεραίνεται ότι λαμβάνει τύπο ημερομηνίας ("Date") και η δοθείσα μορφή είναι ίδια με αυτή που υπάρχει και στο json αρχείο. Αυτό έχει ως αποτέλεσμα οι τιμές των ημερομηνιών να διαβάζονται σωστά και να είναι εφικτή πλέον η δημιουργία γραφημάτων χρονοσειρών. Τέλος το πεδίο Entity Name γνωρίζοντας ότι περιέχει πρακτικά έναν τίτλο ως περιεχόμενο λαμβάνει τον τύπο 'keyword'.

## 2.3 Kibana

### 2.3.1 Ορισμός του Kibana

Το Kibana<sup>15</sup> είναι μια εφαρμογή η οποία προσφέρει περιβάλλον χρήσης προσβάσιμο από κάποιον web browser. Βρίσκεται στην κορυφή του Elastic Stack<sup>16</sup>, παρέχοντας δυνατότητες αναζήτησης και οπτικοποίησης δεδομένων για δεδομένα που ευρετηριάζονται στο Elasticsearch. Κοινώς γνωστό ως εργαλείο χαρτογράφησης για το Elasticsearch, το Kibana λειτουργεί επίσης ως διεπαφή χρήστη για την παρακολούθηση, τη διαχείριση και την ασφάλιση ενός Elasticsearch Stack cluster.<sup>17</sup>

Η ενσωμάτωση του Kibana με το Elasticsearch το καθιστούν ιδανικό για την υποστήριξη των παρακάτω:

1. Αναζήτηση, προβολή και οπτικοποίηση δεδομένων που ευρετηριάζονται (indexed) στο Elasticsearch και ανάλυση των δεδομένων μέσω της δημιουργίας γραφημάτων ράβδων, γραφημάτων πίτας, πινάκων, ιστογραμμάτων και χαρτών. Ο πίνακας (Dashboard) συνδυάζει αυτά τα οπτικά στοιχεία για κοινή χρήση παρουσιάζοντας τα μέσω του προγράμματος περιήγησης για την παροχή αναλυτικών προβολών σε πραγματικό χρόνο, σε μεγάλους όγκους δεδομένων για υποστήριξη περιπτώσεων χρήσης όπως:
  - (α') Καταγραφή και αναλυτικά στοιχεία καταγραφής.
  - (β') Μετρήσεις υποδομής και παρακολούθηση των Containers.
  - (γ') Παρακολούθηση απόδοσης εφαρμογής (Application Performance Monitoring - APM).
  - (δ') Ανάλυση και οπτικοποίηση γεωχωρικών δεδομένων.
  - (ε') Αναλυτικά στοιχεία ασφαλειας.
2. Παρακολούθηση, διαχείριση και διασφάλιση προστασίας των Elasticsearch, Kibana μέσω διασύνδεσης web.
3. Κεντρική πρόσβαση για ενσωματωμένες λύσεις που αναπτύχθηκαν στο Elastic Stack για εφαρμογές παρατηρητικότητας και ασφάλειας.

Το Kibana απευθύνεται σε διαχειριστές, αναλυτές και απλούς χρήστες. Ως διαχειριστές, ο ρόλος μας είναι να διαχειριζόμαστε την πλατφόρμα, από τη δημιουργία της έως τη λήψη δεδομένων από το Elasticsearch στο Kibana και, στη συνέχεια, τη διαχείριση αυτών των δεδομένων. Ως αναλυτές, αναζητούμε να ανακαλύψουμε πληροφορίες στα δεδομένα, να οπτικοποιήσουμε τα δεδομένα μας σε πίνακες και να μοιραστούμε τα ευρήματά σας. Ως απλοί χρήστες, θέλουμε να δούμε τους υπάρχοντες πίνακες (Dashboard) και να εμβαθύνουμε σε λεπτομέρειες.

Επιπρόσθετα, το Kibana λειτουργεί με όλους τους τύπους δεδομένων. Τα δεδομένα μας μπορεί να είναι δομημένα ή μη δομημένα κείμενα, αριθμητικά δεδομένα, δεδομένα χρονοσειρών, γεωχωρικά δεδομένα, αρχεία καταγραφής, μετρήσεις, συμβάντα που σχετίζονται με την ασφάλεια και άλλα. Ανεξάρτητα από το είδος τους, το Kibana μπορεί να μας βοηθήσει να ανακαλύψουμε μοτίβα και σχέσεις και να οπτικοποιήσουμε τα αποτελέσματα.

Στις επόμενες ενότητες επικεντρωνόμαστε στις δυνατότητες που προσφέρει η πλατφόρμα του Kibana οι οποίες αξιοποιούνται στα πλαίσια του έργου Phytobiotics.

## 2.3.2 Security στα Kibana - Elasticsearch<sup>18</sup>

Ο τομέας του Security είναι ένα από τα πιο σημαντικά χαρακτηριστικά που οφείλουν να αξιοποιήσουν οι χρήστες των Kibana και Elasticsearch. Στο παρόν κεφάλαιο, γίνεται αναφορά και υλοποίηση των απαραίτητων ενεργειών για την εξασφάλιση ασφάλειας στο σύστημα του Kibana και του ElasticSearch. Οι ενέργειες αυτές προϋπονθέτουν γνώση τόσο του Docker όσο και του Elasticsearch.

Η προστασία των Elasticsearch, Kibana και των δεδομένων που περιέχουν είναι υψίστης σημασίας. Η εφαρμογή μιας στρατηγικής άμυνας σε βάθος παρέχει πολλαπλά επίπεδα ασφάλειας που βοηθούν στην προστασία του συστήματός μας. Οι αρχές που παρουσιάζονται στη συνέχεια παρέχουν τη βάση για την εκτέλεση της πλατφόρμας με ασφαλή τρόπο που συμβάλλει στην αποφυγή μεγάλου μέρους των επιθέσεων στο σύστημά μας σε πολλαπλά επίπεδα. Να τονιστεί ότι η εκτέλεση ενός Elasticsearch cluster στο οποίο δεν έχουν εφαρμοστεί καλές πρακτικές διασφάλισης ασφάλειας, προσφέρει τη δυνατότητα σε οποιονδήποτε να στείλει κίνηση δικτύου στο Elasticsearch επιτρέποντας τους κατά επέκταση να κατεβάσουν, να τροποποιήσουν ή να διαγράψουν δεδομένα στο σύστημα. Εφαρμόζοντας καλές πρακτικές ασφάλειας σε ένα σύστημα που εξυπηρετεί το Elasticsearch και το Kibana, αποτρέπουμε τη μη εξουσιοδοτημένη πρόσβαση και διασφαλίζουμε την ασφαλή επικοινωνία μεταξύ των κόμβων.

## 2.4 Django

### 2.4.1 Τι είναι το Django

Το Django<sup>19</sup> είναι ένα framework<sup>20</sup> βασισμένο σε Python που ενθαρρύνει την ταχεία ανάπτυξη και τον καθαρό σχεδιασμό της εκάστοτε εφαρμογής κάνοντας τη διαδικασία ανάπτυξης πιο βατή και ευκολότερα κατανοητή στον χρήστη. Διευκολύνει τη διαδικασία ανάπτυξης εφαρμογής παγκοσμίου ιστού προσφέροντας μία σειρά από εργαλεία και έτοιμα κομμάτια κώδικα. Το Django περιλαμβάνει δεκάδες πρόσθετα που μπορούν να χρησιμοποιηθούν για εργασίες ανάπτυξης εφαρμογών ιστού. Το Django φροντίζει για την ταυτοποίηση των χρηστών και τη διαχείριση περιεχομένου σε περίπτωση που ο χρήστης δεν επιθυμεί να δημιουργήσει δικά του συστήματα για τις λειτουργίες αυτές. Το Django λαμβάνει σοβαρά υπόψη την ασφάλεια και βοηθά τους προγραμματιστές να αποφεύγουν πολλά κοινά λάθη ασφαλείας, όπως η έγχυ-

ση SQL (SQL injection)<sup>21</sup>, η δημιουργία δέσμης ενεργειών μεταξύ διαδικτυακών τοποθεσιών (cross-site scripting)<sup>22</sup>, η πλαστογράφηση αιτημάτων μεταξύ διαδικτυακών τοποθεσιών (cross-site request forgery)<sup>23</sup> και η κακόβουλη τεχνική εξαπάτησης ενός χρήστη για να κάνει κλικ σε κάτι (click-jacking)<sup>24</sup>. Το σύστημα ελέγχου ταυτότητας χρήστη παρέχει έναν ασφαλή τρόπο διαχείρισης λογαριασμών χρηστών και κωδικών πρόσβασης.

# Κεφάλαιο 3

## Υλοποίηση

Στο κεφάλαιο αυτό γίνεται αναφορά στη υλοποίηση και στα βήματα που ακολουθούνται για την επίτευξη της υλοποίησης του παρόντος έργου.

### 3.1 Προετοιμασία Δεδομένων (Data Preparation)

Στην ενότητα αυτή περιγράφεται η διαδικασία που πρέπει να γίνει πριν τα δεδομένα περαστούν στο front-end της πλατφόρμας. Αναλύονται οι τύποι δεδομένων που μπορεί να δεχτεί η πλατφόρμα, οι τρόποι προώθησης των data αλλά και τα scripts που χρειάζεται να τρέξει ο χρήστης ανάλογα με των είδος της πληροφορίας που λαμβάνει.

#### 3.1.1 Είδη Δεδομένων

To format των δεδομένων που μπορεί να δεχτεί και να επεξεργαστεί η πλατφόρμα είναι τα παρακάτω:

1. **.json (JavaScript Object Notation):** To JavaScript Object Notation<sup>25</sup> είναι μια ελαφριά μορφή ανταλλαγής δεδομένων. Ελαφριά γιατί περιέχει κυρίως τα δεδομένα που πρέπει να μεταφερθούν, αντί να χρησιμοποιείται πολλή σήμανση για τον καθορισμό της δομής των δεδομένων. Τα αρχεία JSON αποθηκεύουν απλές δομές δεδομένων σε μια οργανωμένη και εύκολα προσβάσιμη μορφή. Είναι μια συλλογή από ζεύγη ονόματος/τιμών (key/values), ενώ είναι και εντελώς ανεξάρτητη γλώσσα αλλά χρησιμοποιεί συμβάσεις που είναι γνωστές στους προγραμματιστές. Το JSON βασίζεται σε δύο δομές:

- Μια συλλογή ζευγών ονόματος/τιμών
- Μια ταξινομημένη λίστα τιμών.

Το JSON είναι μια μορφή που βασίζεται σε κείμενο, στην οποία η πρόσβαση γίνεται μέσω οποιουδήποτε προγράμματος επεξεργασίας κειμένου ή επεξεργασίας γραμμής εντολών (π.χ. Vim) σε λειτουργικά συστήματα εκτός των Windows.

Τα δεδομένα που χρησιμοποιούνται στα πλαίσια του έργου και της διπλωματικής εργασίας ακολουθούν τη δομή json που ακολουθεί.

```
{
  Timestamp: 2022-02-10 11:58:30 ,
  DeviceName: Phytobiotics-RTH2 ,
  CO2 : 2173 ,
  NH3 : 2 ,
  ambient_light (lux): 30 ,
  ambient_noise (dB): 49 ,
  humidity (%): 34 ,
  temperature: 29.21
}
{
  "Timestamp": "2022-03-15 15:09:04",
  "Entity Name": "Phytobiotics-RTH1",
  "CO2 (ppm)": 2180 ,
  "NH3 (ppm)": 7 ,
  "ambient_light (lux)": 30 ,
  "ambient_noise (dB)": 71 ,
  "humidity (%)": 73 ,
  "temperature": 21.1
}
```

Κώδικας 3.1: Τα δεδομένα που χρησιμοποιούνται στα πλαίσια της διπλωματικής

**2. .csv (Comma - Separated Values):** Τα αρχεία κειμένου CSV<sup>26</sup> χρησιμοποιούν κόμματα ή παρόμοια σύμβολα για να διαχωρίσουν τιμές. Συνήθως, αποθηκεύουν δεδομένα σε πίνακα (αριθμούς και κείμενο) σε απλό κείμενο. Μπορούν να ανοίξουν με οποιοδήποτε πρόγραμμα επεξεργασίας κειμένου, όπως το Notepad++. Η μορφή αρχείου CSV δεν είναι πλήρως τυποποιημένη, αν και υπάρχουν ορισμένοι συγκεκριμένοι κανόνες και συστάσεις. Σε περιπτώσεις όπου τα δεδομένα πεδίου περιέχουν επίσης κόμματα ή ενσωματωμένες διακοπές γραμμής, είναι μια κοινή προσέγγιση να χρησιμοποιούνται εισαγωγικά για να περιβάλλουν το πεδίο.

Μια από τις μορφές που μπορεί να έχει ένα csv αρχείο παρουσιάζεται παρακάτω.

- Timestamp; Entity Name; CO2(ppm); NH3(ppm); ambient\_light(lux); ambient\_noise(dB); humidity(%); temperature (°C)  
 2022-02-10 11:58:30; Phytobiotics-RTH2; 2173; 2; 30; 49; 34; 29.21  
 2022-02-21 09:14:16; Phytobiotics-RTH2; 3321; 2; 30; 60; 65; 27.46
- Timestamp, Entity Name, CO2(ppm), NH3(ppm), ambient\_light(lux), ambient\_noise(dB), humidity(%), temperature (°C)  
 2022-01-15 18:35:22, Phytobiotics-RTH1, 2863, 2, 30, 62, 39, 29.59  
 2022-02-20 00:19:47, Phytobiotics-RTH1, 2265, 2, 30, 40, 75, 18.21

Στα παραπάνω csv τύπου δείγματα παρατηρούμε ότι ο διαχωρισμός μεταξύ των τιμών και των κεφαλίδων γίνεται είτε με με κόμμα είτε με το ελληνικό ερωτηματικό (semicolon), αλλά η γενικότερη μορφή παραμένει ίδια. Σε περίπτωση μετατροπής ενός csv αρχείου σε json (βλ. κεφάλαιο 3.2.1), απαραίτητη προϋπόθεσή είναι η αναφορά του κατάλληλου διαχωριστή.

3. **.xls / .xlsx (Spreadsheets)**: Τα υπολογιστικά φύλλα<sup>27</sup> είναι μια εφαρμογή προγράμματος που επιτρέπει την ανάλυση και την αποθήκευση δεδομένων σε μορφή πίνακα. Το υπολογιστικό φύλλο είναι ένα αρχείο που αποτελείται από κελιά σε σειρές και στήλες. Στα υπολογιστικά φύλλα, οι μορφές δεδομένων ποικίλλουν από αριθμητικές τιμές έως κείμενο, τύπους, αναφορές και συναρτήσεις. Εκτός από την αποθήκευση δεδομένων σε πίνακα και την εκτέλεση βασικών αριθμητικών και μαθηματικών συναρτήσεων, υπολογιστικά φύλλα όπως το Excel παρέχουν ενσωματωμένες συναρτήσεις για στατιστικές πράξεις. Παρά την δυνατότητα τους στην υποστήριξη μεγάλου όγκου δεδομένου δεν είναι άμεσα συμβατές με τις τεχνολογίες που χρησιμοποιούμε για την υλοποίηση της πλατφόρμας. Για αυτό και σε περίπτωση που υπάρχει ένα τέτοιου είδους αρχείου προχωρούμε στην μετατροπή του σε .json ή .csv αρχεία που υποστηρίζονται πλήρως.

### 3.1.2 Data Processing & Upload

Στην ενότητα αυτή αναλύονται οι πηγαίοι κώδικες (scripts) που χρησιμοποιούνται για να επεξεργαστούν και να τροποποιηθούν ορισμένα στοιχεία των αρχείων που λαμβάνει ο χρήστης. Η επεξεργασία και τροποποίηση ορισμένων στοιχείων αποτελεί απαραίτητη διαδικασία στην περίπτωση που ο χρήστης έχει διαφορετικού τύπου αρχείου ή και σε περίπτωση που το json αρχείο που δίνεται δεν συμφωνεί με το τύπο που δέχεται το σύστημα (βλ. Κεφάλαιο 3.1). Επιπλέον γίνεται αναφορά στους τρόπους που μπορεί να γίνεται το Uploading (Ανέβασμα) των δεδομένων αυτών και τι προαπαιτείται από τον χρήστη. Η γλώσσα προγραμματισμού που χρησιμοποιείται για την υλοποίηση των διαδικασιών είναι Python.

### 3.1.3 Data Processing

```

1 import pandas
2 import json
3 # Read csv document
4 csv_data = pandas.read_csv('csv File Name.csv')
5 # Convert csv to string (define orientation of document from up to down)
6 json_file = csv_data.to_json(orient='records')
7 # Print out the result if you would like to see the result before
     conversion
8 # print('csv to JSON:\n', json_file)
9 # Make the string into a list to be able to input in to a JSON-file
10 json_list = json.loads(json_file)

```

```

11 # Define file to write to and 'w' for write option -> json.dump() defining
   the list to
12 # write from and file to write to
13 with open('json file name.json', 'w') as json_file:
14     json.dump(json_list, json_file)

```

### Κώδικας 3.2: CSV to JSON

Το παραπόνω κομμάτι κώδικα χρησιμοποιείται για την μετατροπή ενός csv αρχείου σε τύπου json. Θα περιγράψουμε βήμα-βήμα τι κάνει ο κώδικας μας.

- (**Γραμμές 1-2**): Η εισαγωγή των βιβλιοθηκών Pandas, Json, απαραίτητες για επεξεργασία τέτοιου είδους αρχείων.
- (**Γραμμή 4**): Εισαγωγή και διάβασμα του csv αρχείου. Σε περίπτωση που το αρχείο δεν βρίσκεται στον ίδιο φάκελο με το εκτελέσιμο πρόγραμμα χρειάζεται το Full Path του αρχείου.
- (**Γραμμές 6,10**): Οι εντολές αυτές ζεκινούν να διαχωρίζουν το csv αρχείο με βάση την επιλογή "orient=records". Αξίζει να σημειωθεί ότι η συγκεκριμένη επιλογή διαφέρει τόσο με την μορφή που έχει το csv αρχείο όσο και με το πως επιθυμούμε να δομηθεί το τελικό json αρχείο. Στην γραμμή 10 μετατρέπουμε τη λίστα σε String για να εισαχθεί στο τελικό αρχείο. Εναλλακτικές επιλογές αντί του "records" ακολουθούν.
  1. **'split'**: Όταν τελικό json που επιθυμούμε είναι της μορφής:  
Dictionary(Λεξικό)  
{‘index’ -> [index], ‘columns’ -> [columns], ‘data’ -> [values]}
  2. **‘index’**: Μια διαφορετική μορφή τύπου Dictionary.  
{index -> {column -> value}}
  3. **‘columns’**: Τύπου Dictionary με εναλλαγή θέσεων των κατηγοριών.  
{column -> {index -> value}}
  4. **‘values’**: Αυτή η μορφή αρχείου περιέχει μόνο τιμές.  
[[“value\_a”, “value\_b”], [“value\_c”, “value\_d”]]
  5. **‘table’**: Dictionary Style χρησιμοποιείται σπανιότερα από τις υπόλοιπες επιλογές.  
{‘schema’: {schema}, ‘data’: {data}}
  6. **‘records’**: Η επιλογή αυτή επιστρέφει αρχείο json μορφής λίστας(list) και είναι αυτή που επιλέγουμε στα πλαίσια της διπλωματικής εργασίας.  
[column -> value, … , column -> value]
- (**Γραμμές 13,14**): Οι τελευταίες εντολές εκτελούν την μέθοδο dump() η οποία μετατρέπει αντικείμενα της Python σε κατάλληλα json αντικείμενα και τοποθετούνται σε νέο αρχείο. Η εντολή open() πρωτικά επιτρέπει το άνοιγμα του αρχείου και με την επιλογή ’w’ δίνεται η δυνατότητα εγγραφής σε αυτό.

```

1 import pandas
2 import json
3 # Read excel document
4 excel_data_df = pandas.read_excel('My Excel File.xlsx')
5 # Convert excel to string
6 this_is_json = excel_data_df.to_json(orient='records')
7 # Make the string into a list to be able to input in to a JSON-file
8 this_is_json_dict = json.loads(this_is_json)
9 # Define file to write to and 'w' for write option -> json.dump() defining
10 # the list to
11 # write from and file to write to
12 with open('data.json', 'w') as json_file:
    json.dump(this_is_json_dict, json_file)

```

Κώδικας 3.3: xlsx/xls to JSON

Το παραπάνω κομμάτι κώδικα χρησιμοποιείται για την μετατροπή ένος xls/xlsx αρχείου σε τύπου json. Η μόνη διαφορά με τον κώδικα 3.2 αναλύεται στη συνέχεια.

- (**Γραμμή 3**): Η εντολή αυτή χρησιμοποιεί τη μέθοδο `read_excel()` η οποία επιτρέπει το διάβασμα ενός .xlsx αρχείου σε σχέση με τη εντολή `read_csv()` του κώδικα 3.2.

```

1 import csv
2 import json
3
4 def csv_to_json(csv_File_Path, json_File_Path):
5     jsonArray = []
6
7     #read csv file
8     with open(csv_File_Path, encoding='utf-8') as csv_f:
9         #load csv file data using csv library's dictionary reader
10        csvReader = csv.DictReader(csv_f)
11
12        #convert each csv row into python dict
13        for row in csvReader:
14            #add this python dict to json array
15            jsonArray.append(row)
16
17        #convert python jsonArray to JSON String and write to file
18        with open(json_File_Path, 'w', encoding='utf-8') as json_f:
19            json_String = json.dumps(jsonArray, indent=4)
20            json_f.write(json_String)
21
22 csv_File_Path = r'My csv file.csv'
23 json_File_Path = r'my json file.json'
24
25 csv_to_json(csv_File_Path, json_File_Path)

```

Κώδικας 3.4: CSV to JSON

Στον κώδικα 3.4 παρουσιάζεται μια εναλλακτική προσέγγιση μετατροπής ενός csv σε json αρχεία με την χρήση της βιβλιοθήκης CSV έναντι της pandas που χρησιμοποιούμε στους κώδικες 3.2 και 3.3.

- (**Γραμμές 1,2**): Η δήλωση των απαραίτητων βιβλιοθηκών csv, json για την επεξεργασία τέτοιων αρχείων.
- (**Γραμμή 4**): Δήλωση συνάρτησης στην οποία θα πραγματοποιηθεί η διαδικασία μετατροπής ενός csv αρχείου σε json. Σαν ορίσματα έχουμε το Full Path των δύο αρχείων.
- (**Γραμμές 8-15**): Αρχικά χρησιμοποιούμε την μέθοδο open() για να είναι εφικτό το διάβασμα του csv αρχείου. Στην συνέχεια με την εντολή csv.DictReader διαβάζεται το περιεχόμενο. Πραγματοποιώντας μια επαναληπτική διαδικασία, κάθε σειρά που διαβάζεται εισάγεται σαν τιμή πίνακα στο τελικό json.
- (**Γραμμές 17-20**): Στο σημείο αυτό επιλέγοντας ξανά την μέθοδο open() αλλά με τη δυνατότητας εγγραφής, εξού και το "w", μετατρέπουμε το array σε String και το περνάμε στο τελικό έγγραφο.
- (**Γραμμές 22-25**): Στις γραμμές 22,23 ορίζουμε τις μεταβλητές για το πλήρες μονοπάτι (Full Path) των δύο αρχείων που χρειαζόμαστε. Η τελευταία γραμμή πρακτικά καλεί την συνάρτηση με τα ορίσματα της που δημιουργήθηκε στις παραπάνω γραμμές του κώδικα.

```

1 # Creating an indexed Json file
2 import json
3
4 i = 0
5 f = open ('My_File.json', "r")
6 # Reading from file
7 data = json.loads(f.read())
8 # print(len(data))
9
10 # Iterating through the json list
11 #for i in range(len(data)):
12 #    print(data[i])
13
14 # Closing file
15 f.close()
16
17 with open('My_File.json', 'w') as f:
18     for i in range(len(data)):
19         json.dump({"index": {"_id": i}}, f)
20         f.write('\n')
21         json.dump((data[i]), f)
22         f.write('\n')
```

Κώδικας 3.5: Δημιουργία ενός Ευρετηριασμένου (Indexed) Json Αρχείου

Ο κώδικας 3.5 χρησιμοποιείται όταν το αρχείο json που επεξεργαζόμαστε δεν περιέχει `_id`. Η μορφή του αρχείου json που αναγνωρίζει το Elasticsearch οφείλει να έχει ένα ξεχωριστό `_id` (Identity) για κάθε πακέτο δεδομένων που λαμβάνει κατά το ανέβασμα ανεξάρτητα αν το αρχείο αυτό είναι δομημένο κατά στήλες ή γραμμές. Πρόκειται για μια ειδική περίπτωση που δεν εφαρμόζεται κάθε φορά καθώς:

1. Σωστά δομημένα json αρχεία έχουν από πριν το χαρακτηριστικό αυτό, επομένως ποφεύγεται η πρόσθετη εργασία.
  2. Η ίδια διαδικασία μπορεί να εφαρμοστεί κατά τη διάρκεια του ανεβάσματος (Uploading) του αρχείου στην πλατφόρμα.
  3. Το ίδιο το Elasticsearch αναθέτει δικό της τυχαίο `_id` στα πακέτα των δεδομένων σε περίπτωση που ο χρήστης δεν έχει αναθέσει. Παρόλα αυτά, όντας τυχαίο (random) αλφαριθμητικό καθιστά δύσκολή την αναζήτηση αποτελεσμάτων με βάση αυτό.
- (**Γραμμές 2-7**): Κάνοντας εισαγωγή της βιβλιοθήκης json και με την μέθοδο `open()` δίνεται η δυνατότητα φόρτωσης και ανάγνωσης του αρχείου json. Με την εντολή `json.loads(f.read)` διαβάζονται τα δεδομένα ως string για να γίνει σωστά η τροποποίηση.
  - (**Γραμμές 17-22**): Στο κομμάτι αυτό του κώδικα πραγματοποιείται η επεξεργασία του json αρχείου. Έχοντας την δυνατότητα επεξεργασίας και εγγραφής με την εντολή `open()` και την επιλογή `'w'` -> `write`, προσθέτουμε τα απαραίτητα δεδομένα που εμφανίζονται στην γραμμή 19. Με αυτή την προσθήκη η πλατφόρμα θα είναι σε θέση να ερμηνεύσει και να λάβει σωστά τα δεδομένα του αρχείου. Τα στοιχεία που προστίθενται στην περίπτωση μας είναι η ονομασία του δείκτη `index` και το μοναδικό `_id` που η τιμή του προκύπτει από το αριθμό των επαναλήψεων της εντολής στην γραμμή 18. Τέλος στην γραμμή 21 γίνεται προσθήκη των δεδομένων της σειράς του αρχικού αρχείου που βρίσκεται ο δείκτης της επανάληψης.

```
{
  "_index" : "trial",
  "_id" : "898",
  "_score" : 1.0,
  "_source" : {
    "Timestamp" : "2022-01-19 08:32:50",
    "Entity Name" : "Phytobiotics-RTH1",
    "CO2 (ppm)" : 2640,
    "NH3 (ppm)" : 2,
    "ambient_light (lux)" : 30,
    "ambient_noise (dB)" : 75,
    "humidity (%)" : 41,
    "temperature (°C)" : 30.03
  }
},
```

Σχήμα 3.1: Αποτέλεσμα Αναζήτησης με Custom ID

```
{
  "_index" : "via-kibana",
  "_id" : "LNLS4oABRvsyaSXg9a0g",
  "_score" : 1.0,
  "_source" : {
    "CO2 (ppm)" : 1519,
    "humidity (%)" : 54,
    "temperature (°C)" : 21.64,
    "Timestamp" : "2022-02-19 09:23:25",
    "Entity Name" : "Phytobiotics-RTH1",
    "NH3 (ppm)" : 2,
    "ambient_light (lux)" : 30,
    "@timestamp" : "2022-02-19T09:23:25.000+02:00",
    "ambient_noise (dB)" : 71
  }
},
```

Σχήμα 3.2: Αποτέλεσμα Αναζήτησης με Random ID

```
{"index": {"_id": 0}}
{
  "Timestamp": "2022-02-10 08:58:30",
  "Entity Name": "Phytobiotics-RTH1",
  "CO2 (ppm)": 2173,
  "NH3 (ppm)": 2,
  "ambient_light (lux)": 30,
  "ambient_noise (dB)": 49,
  "humidity (%)": 34,
  "temperature (°C)": 29.21
}
{"index": {"_id": 1}}
{
  "Timestamp": "2022-02-21 06:14:16",
  "Entity Name": "Phytobiotics-RTH1",
  "CO2 (ppm)": 3321,
  "NH3 (ppm)": 2,
  "ambient_light (lux)": 30,
  "ambient_noise (dB)": 60,
  "humidity (%)": 65,
  "temperature (°C)": 27.46
}
{"index": {"_id": 2}}
{
  "Timestamp": "2022-02-25 06:38:38",
  "Entity Name": "Phytobiotics-RTH1",
  "CO2 (ppm)": 1814,
  "NH3 (ppm)": 4,
  "ambient_light (lux)": 30,
  "ambient_noise (dB)": 68,
  "humidity (%)": 56,
  "temperature (°C)": 24.21
}
{"index": {"_id": 3}}
{
  "Timestamp": "2022-03-03 07:55:10",
  "Entity Name": "Phytobiotics-RTH1",
  "CO2 (ppm)": 2577,
  "NH3 (ppm)": 29,
  "ambient_light (lux)": 30,
  "ambient_noise (dB)": 71,
```

```

    "humidity (%)": 72 ,
    "temperature (°C)": 23.26
}

```

Κώδικας 3.6: Μορφή ενός Indexed Json File.

Στο κώδικα 3.6 παρατηρούμε το αποτέλεσμα της εκτέλεσης του κώδικα 3.5 (Δημιουργία ενός Indexed Json File). Η ουσιαστική διαφορά με προηγούμενες μορφές Json αρχείου που έχουν αναφερθεί παραπάνω είναι η προσθήκη του Unique \_id μετά από κάθε σειρά δεδομένων καθώς και η προσθήκη ονόματος στο δείκτη.

```

1 # This is to be used when a csv is separated by ; in a single column
2 import pandas as pd
3
4 csv_data = pd.read_csv("my_csv_file.csv", sep=";")
5 csv_data.to_json("my_new_json_file.json", orient="records")

```

Κώδικας 3.7: Απλοποιημένη εκδοχή μετατροπής CSV σε JSON

- Το συγκεκριμένο Script των 5 γραμμών αποτελεί την πιο απλοποιημένη μορφή μετατροπής ενός csv to json αρχείο στα πλαίσια τόσο του Project Phytoiotics όσο και της διπλωματικής εργασίας. Αυτό συμβαίνει διότι η μορφή του csv που χρησιμοποιήθηκε είναι πλήρως δομημένο σε μία και μόνο στήλη. Επιπλέον παρατηρούμε ότι έχει επιλεχτεί το σύμβολο ";" ως διαχωριστής των δεδομένων, επιλογή αποδεκτή όπως προαναφέρθηκε στο Κεφάλαιο 3.1 στην ανάλυση csv αρχείων. Στη περίπτωση όμως που χρησιμοποιηθεί το σύμβολο "," σε παρόμοιας δομής csv αρχείο η χρήση του παραπάνω κώδικα θα ήταν αποδεκτή.

### 3.1.4 Ανέβασμα Αρχείων - Δεδομένων

```

1 import requests
2 import time
3
4 time.sleep(5)
5
6
7 payload = open("my_json_file.json", 'rb')
8 headers = {
9     'content-type': 'application/json'
10 }
11
12 r = requests.post('http://host.docker.internal:9200/my_index_name/_bulk?
    pretty', headers=headers, data=payload)

```

Κώδικας 3.8: Ανέβασμα μέσω της βιβλιοθήκης Requests στο Elasticsearch

- (Γραμμές 1-3):** Πραγματοποιείται η εισαγωγή των βιβλιοθηκών που χρησιμοποιούνται στο κώδικα μας. Η βιβλιοθήκη requests επιτρέπει τη δημιουργία μιας HTTP αίτησης με την οποίη εγκαθιδρύεται μια επικοινωνία μεταξύ του χρήστη και του server, στην περίπτωση μας το elasticsearch.
  - (Γραμμές 7-10):** Μέσω της συνάρτησης open() ορίζουμε το επιθυμητό json αρχείο το οποίο διαβάζεται ως binary. Η μεταβλητή headers χρησιμοποιείται με σκοπό να οριστεί το είδος του περιεχομένου που θα περαστεί στην πλατφόρμα.
  - (Γραμμή 12):** Η εντολή requests.post() είναι υπεύθυνη για το ανέβασμα το δεδομένων μας στη πλατφόρμα. Περιέχει μια σειρά από ορίσματα τα οποία εξυπηρετούν διαφορετικές λειτουργίες:
    - \* `http://host.docker.internal:9200`: Πρόκειται για την διεύθυνση του server που έχει στηθεί η πλατφόρμα. Στα πλαίσια της διπλωματικής εργασίας η όλη διαδικασία έχει υλοποιηθεί σε τοπικό περιβάλλον μέσω docker για αυτό και στο domain έχει σημειωθεί host.docker.internal:9200, όπου το 9200 είναι η πύλη (port) επικοινωνίας όπως έχει διατυπωθεί και στο yml αρχείο στη 2η ενότητα.
    - \* `/my_index_name/_bulk?pretty`: Η συνέχεια της προηγούμενης εντολής, στην οποία σημειώνεται το όνομα του index της αρεσκείας. Τέλος η προσθήκη bulk?pretty υποδηλώνει ότι η αποστολή των δεδομένων θα γίνει σαν σύνολο όλο το αρχείο.
    - \* `headers=headers, data=payLoad`: Τα ορίσματα αυτά δηλώνουν το είδος του περιεχομένου και το ίδιο το αρχείο αντίστοιχα.

```
1 import requests
2 import time
3 from requests.auth import HTTPBasicAuth
4 from requests.structures import CaseInsensitiveDict
5
6 time.sleep(5)
7
8 response = requests.get('https://host.docker.internal:9200',
9                         verify=False,
10                        auth = HTTPBasicAuth('Username', 'Password'))
11
12 # print request object
13 print(response)
14
15
16 payload = open("my_json_file.json", 'rb')
17 headers = {
18     'content-type': 'application/json'
19 }
20 r = requests.post('https://host.docker.internal:9200/my_index_name/_bulk?',
21                   pretty', headers=headers, data=payload, verify=False, auth=(('Username', 'Password')))
```

```
21 print(r.json)
```

Κώδικας 3.9: Ανέβασμα μέσω Requests - Authentication Required

Το παραπάνω έγγραφο κώδικα παρουσιάζει σημαντικές διαφορές σε σχέση με τον κώδικα 3.8. Αρχικά η επικοινωνία μεταξύ χρήστη και server γίνεται μετά από επιτυχή έλεγχο των διαπιστευτηρίων (Authentication Process) του χρήστη. Εφόσον είναι επιτυχής ο έλεγχος, το Upload γίνεται και αυτό με το παρόμοια διαδικασία ελέγχου ταυτότητας. Στη συνέχεια διατυπώνονται οι καίριες διαφορές στο κώδικα που επιτρέπουν την επικοινωνία με το server που έχει ένα βασικό επίπεδο προστασίας.

- (**Γραμμές 8-10**): Παρατηρείται ότι η χρήση της βιβλιοθήκης requests ακολουθείται από τη μέθοδο Get. Στην περίπτωση που υπάρχει Basic Authentication είναι απαραίτητο να γίνει η σύνδεση προκαταβολικά πριν το ανέβασμα των δεδομένων. Συνεπώς στο όρισμα HTTPBasicAuth ζητούνται από το χρήστη τα στοιχεία σύνδεσης του στην πλατφόρμα. Σε περίπτωση που δεν υπάρχουν στη βάση δεδομένων τους η εκτέλεση του υπόλοιπου κώδικα δεν θα παράξει αποτέλεσμα. Η προσθήκη του ορίσματος verify=False έχει να κάνει με τον έλεγχο των πιστοποιητικών για ασφαλής επικοινωνία (Πρωτόκολλο "https"). Στα πλαίσια της διπλωματικής εργασίας τα πιστοποιητικά έχουν δημιουργηθεί σε τοπικό επίπεδο και έτσι δεν θεωρούνται επίσημα και αναγνωρίσιμα έγγραφα, με αποτέλεσμα η τιμή της μεταβλητής να είναι "false". Η επιλογή True είναι εφικτή μόνο όταν τα πιστοποιητικά είναι εγκεκριμένα από επίσημους φορείς.
- (**Γραμμές 16-19**): Μέσω της συνάρτησης open() ορίζουμε το επιθυμητό τλθσον αρχείο το οποίο διαβάζεται ως binary. Η μεταβλητή headers χρησιμοποιείται με σκοπό να οριστεί το είδος του περιεχομένου που θα περαστεί στην πλατφόρμα.
- (**Γραμμή 20**): Παρατηρείται ότι η συγκεκριμένη εντολή έχει αρκετά παρόμοια στοιχεία με αυτή του κώδικα 3.7. Οι βασικές διαφορές είναι η αλλαγή του πρωτοκόλλου σε https, η προσθήκη του ορίσματος verify για το έλεγχο των πιστοποιητικών και τέλος το πόρισμα για το έλεγχο των credentials του χρήστη για ασφαλή σύνδεση.

```
1 from elasticsearch import Elasticsearch
2 import json
3 import urllib3
4 urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
5
6 es = Elasticsearch(['https://localhost:9200'], http_auth=('Username', 'Password'), verify_certs=False)
7
8 body = {
9     "mappings": {
10         "properties": {
11             "@timestamp": {
12                 "type": "date"
13             },
14             "CO2 (ppm)": {
```

```

15         "type": "long"
16     },
17     "Entity Name": {
18         "type": "keyword"
19     },
20     "NH3 (ppm)": {
21         "type": "long"
22     },
23     "Timestamp": {
24         "type": "date",
25         "format": "yyyy-MM-dd HH:mm:ss"
26     },
27     "ambient_light (lux)": {
28         "type": "long"
29     },
30     "ambient_noise (dB)": {
31         "type": "long"
32     },
33     "humidity (%)": {
34         "type": "long"
35     },
36     "temperature (°C)": {
37         "type": "double"
38     }
39 }
40 }
41 }
42 }
43
44
45 def function():
46     p = 0
47     es.indices.create(index='trial', body=body, params=None, headers=None,
48     ignore=400)
49
50     with open('Sensor_Data_RTH1.json') as f:
51         data = json.loads(f.read())
52         for row in range(len(data)):
53             # put document into elasticsearch
54             es.index(index="trial", body=data[row], id=p)
55             # print(obj)
56             p = p + 1
57
58 function()

```

Κώδικας 3.10: Ανέβασμα μέσω Elasticsearch Βιβλιοθήκης -  
 Απαιτείται Ταυτοποίηση Χρήστη - Προσαρμοσμένη Αντιστοίχηση  
 (Mapping)

Η παρούσα υλοποίηση προσεγγίζει την διαδικασία του Upload με διαφορετικό τρόπο συγχριτικά με τις προηγούμενες υλοποιήσεις. Αρχικά αντί της βιβλιοθήκης Requests χρησιμοποιούμε την βιβλιοθήκη Elasticsearch για την επικοινωνία με το Server και δημιουργώ το δικό μου mapping για την αναγνώριση τύπου του κάθε δεδομένου από την πλατφόρμα. Το κομμάτι του Mapping αποτελεί βασικό πυλώνα του Elasticsearch, οπότε η επεξήγηση του γίνεται στη Κεφάλαιο 4. Στην συνέχεια ακολουθεί η ανάλυση του κώδικα.

- (**Γραμμές 1-4**): Γίνεται η εισαγωγή των απαραίτητων βιβλιοθηκών Elasticsearch, json, urllib3 που είναι απαραίτητες για την επικοινωνία με την πλατφόρμα, την δυνατότητα επεξεργασίας του αρχείου και την απόρριψη Warnings που πιθανότατα εμφανιστούν κατά την εκτέλεση του κώδικα.
- (**Γραμμή 6**): Η συγκεκριμένη εντολή χρησιμοποιείται για την ίδρυση επικοινωνίας και πιστοποίησης των διαπιστευτηρίων του χρήστη. Όπως και με την χρήση της μεθόδου Requests ζητείται η συμπλήρωση του URL της πλατφόρμας (Στην περίπτωση αυτή είναι το τοπικό δίκτυο), και τα στοιχεία του χρήστη. Το verify\_certs=False έχει να κάνει με την εγκυρότητα των πιστοποιητικών και την επισημότητα τους. Στα πλαίσια της διπλωματικής εργασίας δεν χρίνεται απαραίτητο και έτσι παίρνει την τιμή αυτή.
- (**Γραμμές 8-42**): Το κομμάτι αυτό του κώδικα αφορά την δήλωση της αντιστοίχησης mapping που θα διαβάσει και θα εφαρμόσει το Elasticsearch στα δεδομένα μόλις περάσουν στην πλατφόρμα. Αυτό που αξίζει να σημειωθεί είναι ότι οι μεταβλητές που ορίζονται μέσα στην μεταβλητή body είναι ίδιες με εκείνες που υπάρχουν στο json που χρησιμοποιείται ως παράδειγμα. Σε περίπτωση αλλαγής αρχείου είναι αναγκαία η δόμηση νέου mapping που συνάδει με τα στοιχεία του εκάστοτε json αρχείου.
- (**Γραμμές 45-58**): Ορίζεται μια συνάρτηση στην οποία πραγματοποιείται η διαδικασία ανεβάσματος το δεδομένων του αρχείου στην πλατφόρμα. Αρχικά στην γραμμή 47 δημιουργείται το ευρετήριο (index) που θα περαστούν τα αρχεία. Πρόκειται για μια βασική διαφορά έναντι της εντολής requests που γίνονταν όλα σε μια εντολή. Εν συνεχεία το αρχείο διαβάζεται ως string και μέσα από μία επαναληπτική διαδικασία περνιέται ανά σειρά στο index (γραμμή 53). Να τονιστεί ότι σαν ορίσματα επιλέγονται το όνομα του index που θα περαστούν τα δεδομένα αλλά και ο τρόπος με τον οποίο διαβάζονται, ανά σειρά.

## 3.2 Kibana

### 3.2.1 Εφαρμογή των χαρακτηριστικών ασφαλείας

Πρώτον και κυριότερο, για να είναι εφικτή η εφαρμογή του χαρακτηριστικού της ασφάλειας είναι απαραίτητη και η συμπεριληφθή του μέσω απλής αναφοράς στο docker-compose.yml. Η εντολή **xpack.security.enabled=true** που βρίσκεται στο environment section του Elasticsearch Container προσφέρει την δυνατότητα εφαρμογής χαρακτηριστικών ασφαλείας στο Elasticsearch και κατ' επέκταση στο Kibana.

```

1 version: "2"
2 services:
3   elasticsearch:
4     image: docker.elastic.co/elasticsearch/elasticsearch:8.2.0
5     container_name: elastic01
6     restart: "no"
7     environment:
8       - node.name=node1
9       - xpack.security.enabled=true
10      - discovery.type=single-node
11      - bootstrap.memory_lock=true
12      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
13      - xpack.ml.enabled=true
14      - "node.roles = -data, -ingest -ml -transform"

```

Κώδικας 3.11: Η ασφάλεια είναι ενεργοποιημένη

Για την εφαρμογή των όποιων ρυθμίσεων ασφαλείας, απαραίτητη προϋπόθεση είναι η εκτέλεση του εκάστοτε container.

- **docker exec -it -u 0 ”Container-Name” bash:** Όπως έχει ήδη αναφερθεί στο κεφάλαιο 2.2, το Container είναι μια αυτόνομη μονάδα λογισμικού που περιέχει εκτελέσιμο κώδικα. Η τρέχουσα εντολή λοιπόν δίνει την δυνατότητα στο χρήστη να εκτελέσει και να εισέλθει στα containers των Elasticsearch - Kibana και να πραγματοποιήσει κάθε είδους τροποποίηση. Η εντολή αυτή εκτελείται μέσω του περιβάλλοντος γραφμής εντολών ‘cmd’. Η παράμετρος -it χρησιμοποιείται για να παραμείνει ανοιχτή η σύνδεση ακόμα και αν δεν δούθει κάποια εντολή για αρκετή ώρα αποφεύγοντας έτσι το φαινόμενο του τερματισμού της σύνδεσης λόγω αδράνειας. Ταυτόχρονα, κάνει πιο ομαλό και κατανοητό το περιβάλλον διεπαφής. Τέλος η παράμετρος ”-u 0” παρέχει δικαιώματα και δυνατότητες διαχειριστή.

```
C:\Users\user>docker exec -it -u 0 es01 bash
root@e3f099ca6d44:/usr/share/elasticsearch#
```

Σχήμα 3.3: Εκτέλεση Εντολής docker exec στο Elasticsearch

Στο σχήμα 3.3 παρατηρούμε το αποτέλεσμα εκτέλεσης της εντολής docker exec. Βλέποντας το root στην αρχή της επόμενης γραφμής συμπεραίνουμε ότι έχουμε δικαιώματα admin<sup>28</sup>, και στο μονοπάτι /usr/share/elasticsearch είναι όλα τα Configuration files του Elasticsearch. Το ίδιο ισχύει και στην περίπτωση του Kibana.

Αξίζει να αναφέρουμε ότι τα es01 - kib01 αποτελούν τις ονομασίες των Containers των Elasticsearch και Kibana αντίστοιχα που επιλέχθηκαν κατά την εκτέλεση της εντολής έναρξης των container (σχήμα 3.4). Η εντολή μπορεί να εκτελεστεί μόνο όταν το Container στο οποίο

```
C:\Users\user>docker exec -it -u 0 kib01 bash
root@461bf7d16cfb:/usr/share/kibana# _
```

Σχήμα 3.4: Εκτέλεση Εντολής docker exec στο Kibana

επιθυμούμε να μπούμε είναι εν ενεργεία. Σε περίπτωση κάποιου σφάλματος του συστήματος ή του ίδιου του Container η σύνδεση θα χαθεί. Τέλος για να βγούμε από το περιβάλλον του container εκτελούμε την εντολή exit.

```
root@e3f099ca6d44:/usr/share/elasticsearch/config# exit
exit
C:\Users\user>
```

Σχήμα 3.5: Εκτέλεση Εντολής docker exit μέσα σε Container

Αρχικά στη γραμμή εντολών εκτελούμε την εντολή docker exec για να πραγματοποιήσουμε την είσοδο στο Container του Elasticsearch και ακολουθούμε βήμα-βήμα την παρακάτω διαδικασία:

- 1. apt-get update:** Η εντολή αυτή κατεβάζει τις λίστες πακέτων από τα αποθετήρια (repositories) και τις «ενημερώνει» για να λάβει πληροφορίες για τις πιο πρόσφατες εκδόσεις πακέτων και τις εξαρτήσεις τους.<sup>29</sup> Η διαδικασία αυτή γίνεται για όλα τα repositories. Η χρήση της μας επιτρέπει να εκτελέσουμε την επόμενη εντολή.

```
C:\Users\user\Desktop>docker exec -it -u 0 es01 bash
root@04d6a7e27729:/usr/share/elasticsearch# apt-get update
Get:1 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [27.5 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [1331 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [881 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [30.3 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2420 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1161 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [1411 kB]
Get:16 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [1979 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [27.1 kB]
Get:18 http://archive.ubuntu.com/ubuntu focal-backports/main amd64 Packages [54.2 kB]
Fetched 22.8 MB in 30s (746 kB/s)
Reading package lists... Done
root@04d6a7e27729:/usr/share/elasticsearch# _
```

Σχήμα 3.6: Εκτέλεση Εντολής apt-get update μέσα σε Container

- 2. apt-get install nano:** Πρακτικά η συγκεκριμένη εντολή εκτελείται με σκοπό να ομαλοποιήσει τη χρήση του περιβάλλοντος του container εγκαθιστώντας ένα λογισμικό κειμενογράφου. Το πρόγραμμα Nano<sup>30</sup> πρόκειται για έναν κειμενογράφο (editor) ο οποίος θα βοηθήσει σε οποιαδήποτε διαδικασία επεξεργασίας προβούμε κατά τη διάρκεια εγκατάστασης των Security Features.

```
[root@04d6a7e27729:/usr/share/elasticsearch# apt-get install nano
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  nano
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 42 not upgraded.
Need to get 269 kB of archives.
After this operation, 868 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 nano amd64 4.8-1ubuntu1 [269 kB]
Fetched 269 kB in 1s (393 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package nano.
(Reading database ... 4800 files and directories currently installed.)
Preparing to unpack .../nano_4.8-1ubuntu1_amd64.deb ...
Unpacking nano (4.8-1ubuntu1) ...
Setting up nano (4.8-1ubuntu1) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor (editor) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/editor.1.gz because associated file /usr/share/man/man1/nano.1.gz (of link group editor) doesn't exist
update-alternatives: using /bin/nano to provide /usr/bin/pico (pico) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/pico.1.gz because associated file /usr/share/man/man1/nano.1.gz (of link group pico) doesn't exist
root@04d6a7e27729:/usr/share/elasticsearch#
```

Σχήμα 3.7: Εκτέλεση Εντολής apt-get isntall nano μέσα σε Container

3. **./elasticsearch-certutil ca:** Πρωταρχικά πριν την εκτέλεση της εντολής, ελέγχουμε το path που βρισκόμαστε να είναι **/usr/share/elasticsearch/bin**. Σε οποιαδήποτε άλλη περίπτωση χρησιμοποιούμε την εντολή (*cd 'change directory'*).<sup>31</sup> Η εκτέλεση της εντολής παράγει ένα αρχείο το οποίο είναι ένα δημόσιο πιστοποιητικό (Certificate Authority) για το σύστημα μας και ένα ιδιωτικό κλειδί για την αυθεντικότητα του πιστοποιητικού. Πριν ολοκληρωθεί η εκτέλεση της εντολής δίνεται η δυνατότητα στο χρήστη να δώσει το δικό του όνομα και κωδικό πρόσβασης στο αρχείο. Η παραγωγή και χρήση πιστοποιητικού συνίσταται μόνο αν το σύνολο του έργου πρόκειται να προχωρήσει σε στάδιο παραγωγής.

```
root@04d6a7e27729:/usr/share/elasticsearch# cd bin
root@04d6a7e27729:/usr/share/elasticsearch/bin# ./elasticsearch-certutil ca
This tool assists you in the generation of X.509 certificates and certificate
signing requests for use with SSL/TLS in the Elastic stack.

The 'ca' mode generates a new 'certificate authority'
This will create a new X.509 certificate and private key that can be used
to sign certificate when running in 'cert' mode.

Use the 'ca-dn' option if you wish to configure the 'distinguished name'
of the certificate authority

By default the 'ca' mode produces a single PKCS#12 output file which holds:
  * The CA certificate
  * The CA's private key

If you elect to generate PEM format certificates (the -pem option), then the output will
be a zip file containing individual files for the CA certificate and private key

Please enter the desired output file [elastic-stack-ca.p12]:
Enter password for elastic-stack-ca.p12 :
root@04d6a7e27729:/usr/share/elasticsearch/bin#
```

Σχήμα 3.8: Εκτέλεση Εντολής elasticsearch-certutil -ca

4. **./elasticsearch-certutil cert --ca elastic-stack-ca.p12:** Στο ίδιο μονοπάτι **/usr/share/elasticsearch/bin** που εκτελέστηκε η προηγούμενη εντολή, τρέχουμε την τρέχουσα εντολή. Όπως και προηγουμένως, η εκτέλεση της εντολής αυτής παράγει ιδιωτικά κλειδιά και πιστοποιητικά για το σύστημα μας, με την διαφορά ότι τώρα συμπεριλαμβάνουμε το αρχείο που παράχθηκε από την προηγούμενη εντολή **elastic-stack-ca.p12**. (Η εντολή **./elasticsearch-certutil cert --ca elastic-stack-ca.p12**: συμπεριλαμβάνει το

αρχείο που παράχθηκε. Το αρχείο είναι το elastic-stack-ca.p12 και έχει τα κλειδιά μέσα. Η διαφορά είναι ότι η πρώτη δημιουργεί το αρχείο με CA (Certificate authority) για βασική TLS-SSL security. Η δεύτερη χρησιμοποιεί τη CERT μέθοδο. Έχει και αυτό τη βασική TLS-SSL με την δυνατότητα για παράλληλη χρήση σε άλλα container). Επίσης η εντολή αυτή εξασφαλίζει την δημιουργία πιστοποιητικών για πολλαπλούς κόμβους σε περίπτωση που υπάρχουν στο δίκτυο. Τέλος πριν τελειώσει η διαδικασία, ζητείται από τον χρήστη ο κωδικός πρόσβασης του αρχείου elastic-stack-ca.p12 και αν επιθυμεί να δώσει κωδικό πρόσβασης για το νέο αρχείο που θα προκύψει.

```
root@04d6a7e27729:/usr/share/elasticsearch/bin# elasticsearch-certutil cert --ca elastic-stack-ca.p12
This tool assists you in the generation of X.509 certificates and certificate
signing requests for use with SSL/TLS in the Elastic stack.

The 'cert' mode generates X.509 certificate and private keys.
  * By default, this generates a single certificate and key for use
    on a single instance.
  * The '-multiple' option will prompt you to enter details for multiple
    instances and will generate a certificate and key for each one
  * The '-in' option allows for the certificate generation to be automated by describing
    the details of each instance in a YAML file

  * An instance is any piece of the Elastic Stack that requires an SSL certificate.
    Depending on your configuration, Elasticsearch, Logstash, Kibana, and Beats
    may all require a certificate and private key.
  * The minimum required value for each instance is a name. This can simply be the
    hostname, which will be used as the Common Name of the certificate. A full
    distinguished name may also be used.
  * A filename value may be required for each instance. This is necessary when the
    name would result in an invalid file or directory name. The name provided here
    is used as the directory name (within the zip) and the prefix for the key and
    certificate files. The filename is required if you are prompted and the name
    is not displayed in the prompt.
  * IP addresses and DNS names are optional. Multiple values can be specified as a
    comma separated string. If no IP addresses or DNS names are provided, you may
    disable hostname verification in your SSL configuration.

  * All certificates generated by this tool will be signed by a certificate authority (CA)
    unless the --self-signed command line option is specified.
    The tool can automatically generate a new CA for you, or you can provide your own with
    the --ca or --ca-cert command line options.

By default the 'cert' mode produces a single PKCS#12 output file which holds:
  * The instance certificate
  * The private key for the instance certificate
  * The CA certificate

If you specify any of the following options:
  * -pem (PEM formatted output)
  * -multiple (generate multiple certificates)
  * -in (generate certificates from an input file)
then the output will be a zip file containing individual certificate/key files

Enter password for CA (elastic-stack-ca.p12) :
Please enter the desired output file [elastic-certificates.p12]:
Enter password for elastic-certificates.p12 :

Certificates written to /usr/share/elasticsearch/elastic-certificates.p12

This file should be properly secured as it contains the private key for
your instance.
```

Σχήμα 3.9: Εκτέλεση Εντολής elasticsearch-certutil -ca elastic-stack-ca.p12

5. **mv elastic-certificates.p12 /usr/share/elasticsearch/config:** Επιστρέφοντας σε πρώτη φάση στο path `/usr/share/elasticsearch`, εκτελούμε την εντολής μετακίνησης "mv" η οποία μας επιτρέπει να μεταφέρουμε το αρχείο που δημιουργήθηκε (`elastic-certificates.p12`), στο φάκελο όπου βρίσκεται το `elasticsearch.yml` αρχείο, μέσα στο οποίο θα πραγματοποιηθεί μια σειρά αλλαγών. Εκτελώντας στην συνέχεια την εντολή "ls" διαπιστώνουμε αν η μετακίνηση ήταν επιτυχής.

```
root@04d6a7e27729:/usr/share/elasticsearch/config# ls
elastic-certificates.p12      elasticsearch.keystore    jvm.options      log4j2.file.properties  role_mapping.yml   users
elasticsearch-plugins.example.yml  elasticsearch.yml    jvm.options.d    log4j2.properties    roles.yml        users_roles
```

Σχήμα 3.10: Το αρχείο `elastic-certificates.p12` στο επιθυμητό μονοπάτι.

6. **nano elasticsearch.yml:** Χρησιμοποιώντας τον Editor που εγκαταστήσαμε σε προηγούμενο βήμα ανοίγουμε το αρχείο `elasticsearch.yml` που βρίσκεται στο μονοπάτι `/usr/share/elasticsearch/config`, και προσθέτουμε χωρίς να διαγράψουμε κάτι από τον ήδη υπάρχον κώδικα, τις παρακάτω γραμμές κώδικα. Για την αποθήκευση του αρχείου πατάμε `CTRL+X` και `"Y"`.

```
1   cluster.name: "docker-cluster"
2   node.name: "node1"
3   xpack.security.enabled: true
4   xpack.security.transport.ssl.enabled: true
5   xpack.security.transport.ssl.verification_mode: certificate
6   xpack.security.transport.ssl.keystore.path: elastic-certificates.
p12
7   xpack.security.transport.ssl.truststore.path: elastic-certificates
.p12
8   xpack.security.transport.ssl.client_authentication: required
9
```

Κώδικας 3.12: Γραμμές κώδικα που προστίθενται στο `elasticsearch.yml`

- Έχοντας αυτές τις γραμμές κώδικα στο αρχείο ενεργοποιείται η δυνατότητα επικοινωνίας μεταξύ πολλαπλών κόμβων σε περίπτωση που έχουν στηθεί παραπάνω από ένας και παρέχεται πρόσβαση στα πιστοποιητικά που έχουμε δημιουργήσει. Λόγω της χρήσης πιστοποιητικού τύπου "p12", στην γραμμή 5 επιλέγουμε την τιμή "certificate". Εναλλακτικά, η δυνατότητα επιλογής "Full" για την οποία η διαδικασία παραγωγής πιστοποιητικού είναι διαφορετική κατά την οποία το πιστοποιητικό επαληθεύεται με βάση την IP του μηχανήματος στο οποίο εκτελούνται τα containers.
7. **chown root:elasticsearch /usr/share/elasticsearch/config/elastic-certificates.p12**  
**chmod 755 /usr/share/elasticsearch/config/elastic-certificates.p12**

Εκτελώντας τις παραπάνω εντολές τη μια μετά την άλλη, με την εντολή `chown32` ορίζουμε ως ιδιοκτήτη του πιστοποιητικού που παράχθηκε τον χρήστη `root` που ανήκει στο

group (ομάδα) χρηστών elasticsearch. Σε πολλές περιπτώσεις, η εντολή αυτή μπορεί να εκτελεστεί μόνο από τον υπερχρήστη. Έπειτα, με την δεύτερη εντολή chmod<sup>33</sup> αλλάζουμε τα δικαιώματα πρόσβασης στο αρχείο. Στην προκειμένη, το νούμερο 755 υποδηλώνει ότι ο ιδιοκτήτης του αρχείου έχει δικαιώματα επεξεργασίας, εκτέλεσης και ανάγνωσης του αρχείου. Οποιοσδήποτε άλλος χρήστης έχει τη δυνατότητα να εκτελέσει και να διαβάσει το αρχείο. Το αποτέλεσμα των εντολών μπορεί να ελεγχθεί απλά με την εκτέλεση της εντολής ”ls -ltr” η οποία παρουσιάζει τους ιδιοκτήτες και τι δικαιώματα αντιστοιχούν για κάθε αρχείο σε ένα συγκεκριμένο μονοπάτι.

```
root@04d6a7e27729:/usr/share/elasticsearch/config# ls -ltr
total 64
-rw-rw-r-- 1 root      root      2992 Feb  3 16:47 jvm.options
-rw-rw-r-- 1 root      root      1042 Feb  3 16:47 elasticsearch-plugins.example.yml
-rw-rw-r-- 1 root      root      0 Feb  3 16:51 users_roles
-rw-rw-r-- 1 root      root      0 Feb  3 16:51 users
-rw-rw-r-- 1 root      root     197 Feb  3 16:51 roles.yml
-rw-rw-r-- 1 root      root     473 Feb  3 16:51 role_mapping.yml
-rw-rw-r-- 1 root      root   16480 Feb  3 16:52 log4j2.file.properties
drwxrwxr-x 1 elasticsearch root    4096 Feb  3 16:52 jvm.options.d
-rw-rw-r-- 1 root      root   11078 Feb  3 22:53 log4j2.properties
-rw-rw---- 1 elasticsearch root    199 Jun 27 10:07 elasticsearch.keystore
-rwxr-xr-x 1 root      elasticsearch 3596 Jun 27 14:03 elastic-certificates.p12
-rw-rw-r-- 1 root      root     327 Jun 27 19:31 elasticsearch.yml
```

Σχήμα 3.11: Αποτέλεσμα της ls -ltr

8. **./elasticsearch-setup-passwords interactive:** Σε αυτό το σημείο δημιουργούμε τους κωδικούς πρόσβασης για τους default χρήστες που υπάρχουν ήδη στο σύστημα. Επιστρέφοντας στο μονοπάτι /usr/share/elasticsearch/bin εκτελούμε την εντολή. Κατά την εκτέλεση της το σύστημα ζητά από τον χρήστη να δώσει τους δικούς του κωδικούς πρόσβασης για όλες τις υπηρεσίες που παρέχει το Elk Stack ανεξάρτητα αν χρησιμοποιούνται ή όχι από το χρήστη. Για δική μας διευκόλυνση ορίζουμε τους παρακάτω κωδικούς πρόσβασης για τους built-in χρήστες.

```
* user [elastic] : password [elastic]
* user [apm_system] : password [apmsystem]
* user [kibana] : password [kibana]
* user [logstash_system] : password [logstash]
* user [beats_system] : password [beatssystem]
* user [remote_monitoring_user] : password [remotemonitoring]
```

Είναι ξεκάθαρο φυσικά ότι ο ορισμός εύκολων κωδικών και ιδίως κωδικών οι οποίοι είναι ίδιοι ή πανομοιότυποι με το εκάστοτε όνομα χρήστη είναι μία πρακτική που δεν πρέπει να ακολουθείται σε συστήματα.

Σε περίπτωση που ο χρήστης επιθυμεί να αποφύγει αυτή την διαδικασία επιλογής κωδικού για κάθε έναν από τους χρήστες, μπορεί να εκτελέσει μια παραλλαγή της προηγούμενης εντολής. Εκτελώντας ./elasticsearch-setup-passwords auto παράγονται αυτόματα από

το σύστημα μια σειρά αλφαριθμητικών που αντιστοιχούν στους κωδικούς πρόσβασης των προαναφερθέντων χρηστών.

9. **nano elasticsearch.yml:** Επιστρέφουμε στο αρχείο elasticsearch.yml στο path /usr/share/elasticsearch/config και προσθέτουμε τις παρακάτω γραμμές κώδικα που θα μας επιτρέψουν να ενεργοποιήσουμε το πρωτοκόλλο HTTPS<sup>34</sup>.

```

1 xpack.security.http.ssl.enabled: true
2 xpack.security.http.ssl.keystore.path: elastic-certificates.p12
3 xpack.security.http.ssl.truststore.path: elastic-certificates.p12
4 xpack.security.http.ssl.client_authentication: optional
5

```

Κώδικας 3.13: Γραμμές κώδικα που προστίθενται στο elasticsearch.yml

10. **openssl pkcs12 -in elastic-certificates.p12 -out newfile.crt.pem -clcerts -nokeys openssl pkcs12 -in elastic-certificates.p12 -out newfile.key.pem -nocerts -nodes**

Στη συνέχεια εκτελούμε αυτές τις δύο εντολές διαδοχικά στο μονοπάτι /usr/share/elasticsearch/config, παράγοντας έτσι τα απαραίτητα πιστοποιητικά και κλειδιά για το Kibana. Να τονίσουμε βέβαια ότι η ενέργεια αυτή πραγματοποιείται μέσα στο Container του Elasticsearch, γι' αυτό τα παραχθέντα αρχεία θα μεταφερθούν στο Container του Kibana.

11. Σε αυτό το βήμα πραγματοποιείται η μετακίνηση των πιστοποιητικών που δημιουργήθηκαν στο βήμα 10, από το Container του Elasticsearch σε αυτό του Kibana. Εξ ορισμού το Docker δεν υποστηρίζει την επικοινωνία των Container σε επίπεδο OS. Συνεπώς, για λυθεί το πρόβλημα αυτό δημιουργούμε ένα νέο, κοινό μοναδικό μονοπάτι (File Path) μέσω του docker-compose.yml αρχείου. Μέσα από το Volumes Section του yml αρχείου μπορούμε να ορίσουμε το κοινό μονοπάτι και στα δύο Containers που επιτρέπει την μεταξύ τους μεταφορά αρχείων. Έχοντας δηλώσει το επιθημένο μονοπάτι, στην περίπτωση μας το /usr/share/elasticsearch/shared προχωράμε στην μεταφορά των πιστοποιητικών.

- **mv newfile.\* /usr/share/elasticsearch/shared:** Πηγαίνοντας στο μονοπάτι που βρίσκονται τα πιστοποιητικά εκτελούμε την εντολή. Ο αστερίσκος στην εντολή υποδηλώνει ότι μεταφέρει όλα τα αρχεία που ζεκινούν με την λέξη "newfile.". Οι ονομασίες των πιστοποιητικών στην περίπτωση μας είναι **newfile.crt.pem & newfile.key.pem**.

```

root@03784cd75c8a:/usr/share/elasticsearch# cd shared
root@03784cd75c8a:/usr/share/elasticsearch/shared# ls
newfile.crt.pem  newfile.key.pem

```

Σχήμα 3.12: Τα πιστοποιητικά στο κοινό Path μετά την εκτέλεσης της mv.

12. Εφόσον η μεταφορά των απαραίτητων αρχείων είναι επιτυχής συνεχίζω στο Container του Kibana. Εκτελώντας την εντολή ”docker exec -it -u 0 kibana01 bash” έχω πρόσβαση στο container. Εν συνεχεία βρίσκω τα αρχεία που μετέφερα σε προηγούμενο βήμα στο μονοπάτι /usr/share/elasticsearch/shared με την χρήση της εντολής change Directory, **cd /usr/share/elasticsearch/shared**.

```
root@f00923800c59:/usr/share/kibana# cd /usr/share/elasticsearch/shared
root@f00923800c59:/usr/share/elasticsearch/shared# ls
newfile.crt.pem  newfile.key.pem
root@f00923800c59:/usr/share/elasticsearch/shared#
```

Σχήμα 3.13: Τα κοινά αρχεία στο Container του Kibana.

13. **chown root:kibana /usr/share/elasticsearch/shared/newfile\***  
**chmod 755 /usr/share/elasticsearch/shared/newfile\***

Όπως και στα πιστοποιητικά του Elasticsearch στο βήμα 7, έτσι και σε αυτά, η αλλαγή δικαιωμάτων χρήσης και ιδιοκτησίας είναι αναγκαία για να μπορέσουμε να τα χρησιμοποιήσουμε.

```
root@f00923800c59:/usr/share/elasticsearch/shared# chown root:kibana /usr/share/elasticsearch/shared/newfile*
root@f00923800c59:/usr/share/elasticsearch/shared# chmod 755 /usr/share/elasticsearch/shared/newfile*
root@f00923800c59:/usr/share/elasticsearch/shared# ls -ltr
total 8
-rwxr-xr-x 1 root kibana 1338 Jun 29 14:12 newfile.crt.pem
-rwxr-xr-x 1 root kibana 1849 Jun 29 14:12 newfile.key.pem
root@f00923800c59:/usr/share/elasticsearch/shared#
```

Σχήμα 3.14: Αλλαγή δικαιωμάτων στο Container του Kibana.

14. **mv newfile.\* /usr/share/kibana/config:** Μετακινούμε τα πιστοποιητικά στο συγκεκριμένο μονοπάτι καθώς πρέπει να γίνει η αναφορά τους από το yml αρχείο του kibana. Ο φάκελος config του Container θα έχει τα παρακάτω αρχεία.

```
root@f00923800c59:/usr/share/kibana/config# ls
kibana.yml  newfile.crt.pem  newfile.key.pem  node.options
```

Σχήμα 3.15: Μεταφορά πιστοποιητικών.

15. **apt-get update & apt-get install nano**

Η χρήση ενός κειμενογράφου (editor) για την επεξεργασία του yml αρχείου είναι αναγκαία, συνεπώς εκτελούμε τις παραπάνω εντολές ξεχωριστά για το έλεγχο ενημερώσεων και αναβαθμίσεων των πακέτων. Στην συνέχεια εγκαθιστούμε το λογισμικό με την 2η εντολή. Τα containers όντας αυτόνομες και ανεξάρτητες οντότητες, το γεγονός ότι η εκτέλεση κάποιων εντολών πραγματοποιήθηκε στο Container του Elasticsearch, δεν εγγυάται την αυτόματη ταυτόχρονη εκτέλεση τους και στο συγκεκριμένο Container, εξού και η επανεκτέλεση τους.

16. **nano kibana.yml:** Ελέγχοντας πρώτα ότι βρισκόμαστε στο σωστό μονοπάτι, (/usr/share/kibana/config) ανοίγουμε το αρχείο kibana.yml με τον κειμενογράφο και προσθέτουμε τις παρακάτω εντολές:

```

1
2     elasticsearch.hosts: [ "https://localhost:9200" ]
3     elasticsearch.username: "kibana"
4     elasticsearch.password: "kibana"
5     server.ssl.enabled: true
6     server.ssl.certificate: /usr/share/kibana/config/newfile.crt.pem
7     server.ssl.key: /usr/share/kibana/config/newfile.key.pem
8     elasticsearch.ssl.verifyMode: none
9

```

Κώδικας 3.14: Γραμμές κώδικα που προστίθενται στο kibana.yml

Με τις παραπάνω προσθήκες διασφαλίζουμε χρυπτογραφημένη επικοινωνία μεταξύ του φυλλομετρητή και του Kibana. Έχοντας δημιουργήσει κλειδί και πιστοποιητικό, το Kibana χρησιμοποιεί το πιστοποιητικό και το αντίστοιχο κλειδί κατά τη λήψη συνδέσεων από προγράμματα περιήγησης ιστού. Να αναφέρουμε ότι στην γραμμή 2 του κώδικα 3.14 έχει δηλωθεί η διεύθυνση του ιστότοπου του Elasticsearch. Φυσικά σε περίπτωση που γίνει γίνει μεταφορά του συστήματος εκτός localhost, το localhost θα αντικατασταθεί με την IP διεύθυνση του server που έχει στηθεί η εφαρμογή. Τέλος στις γραμμές 3-4 παρατηρούμε το όνομα και τον κωδικό ενός από τους χρήστες που δημιουργήσαμε σε προηγούμενο βήμα. Αυτό διασφαλίζει την εκκίνηση του Kibana χωρίς να χρειαστεί ο χρήστης να ανατρέξει πρώτα στο Elasticsearch.

Σε αυτό το σημείο έχουν ολοκληρωθεί οι ρυθμίσεις ασφαλείας. Με την σωστή εκτέλεση των παραπάνω βημάτων έχουμε εξασφαλίσει χρυπτογραφημένη επικοινωνία μεταξύ Elasticsearch και Kibana μέσω των πρωτοκόλλων (SSL - TLS) και του πρωτοκόλλου HTTPS σε Elasticsearch και Kibana καθώς και χρυπτογραφημένη επικοινωνία μεταξύ περιηγητή (Browser) και Kibana.

### 3.2.2 Data Search<sup>35</sup><sup>36</sup>

Σε αυτή την ενότητα θα αναλύσουμε το πως πραγματοποιείται η αναζήτηση δεδομένων. Αρχικά λαμβάνουμε υπόψη ότι έχει γίνει σωστό ανέβασμα των δεδομένων στο σύστημα, είτε μέσω της ίδιας της διεπαφής του Kibana είτε μέσω κατάλληλων script που έχουμε επεξηγήσει σε προηγούμενο κεφάλαιο. Η αναζήτηση των δεδομένων αυτών είναι εφικτή μέσω της διεπαφής του Kibana και το εργαλείο Dev Tools αλλά και μέσω της εντολής CURL<sup>37</sup> μέσα από την γραμμή εντολών cmd.

### 3.2.3 Dev Tools

Το παρών εργαλείο είναι προσβάσιμο από την πλατφόρμα του Kibana. Πληκτρολογώντας την διεύθυνση <https://localhost:5601> στον περιηγητή ανοίγει η σελίδα του Kibana. Συμπληρώνοντας το **Username: elastic & Password: elastic** έχουμε πρόσβαση στο Kibana

με πλήρη δικαιώματα. Επιλέγοντας το Menu στα αριστερά αναζητούμε και επιλέγουμε το Dev Tools. Το Dev Tools περιέχει εργαλεία που μπορούμε να χρησιμοποιήσουμε για να αλληλεπιδράσουμε με τα δεδομένα μας.

- **Κονσόλα:** Αλληλεπίδραση με τα REST API των Elasticsearch και Kibana, συμπεριλαμβανομένης της αποστολής αιτημάτων και της προβολής τεκμηρίωσης API.
- **Αναζήτηση Προφίλ:** Επιθεώρηση και ανάλυση των ερωτημάτων αναζήτησης.
- Δυνατότητα ανάλυσης και επεξεργασίας κώδικα.

### 3.2.4 Rest API

Το Representational State Transfer (REST) είναι μια διεπαφή εφαρμογών προγραμματισμού που ορίζει ένα σύνολο περιορισμών που θα χρησιμοποιηθούν για τη δημιουργία υπηρεσιών web. Το REST API είναι ένας τρόπος πρόσβασης σε υπηρεσίες web με απλό και ευέλικτο τρόπο χωρίς καμία επεξεργασία. Ορισμένες δυνατότητες του Kibana παρέχονται μέσω ενός REST API, το οποίο είναι ιδιαίτερο για τη δημιουργία μιας ενοποίησης με το Kibana.

Οι κλήσεις API είναι αδήλωτες. Κάθε αίτημα που κάνουμε πραγματοποιείται μεμονωμένα από άλλες κλήσεις και πρέπει να περιλαμβάνει όλες τις απαραίτητες πληροφορίες ώστε το Kibana να εκπληρώσει το αίτημα. Το API ζητά επιστροφή εξόδου JSON, η οποία είναι μια μορφή που είναι αναγνώσιμη από μηχανή και λειτουργεί καλά για διεργασίες αυτοματισμού. Οι κλήσεις προς τα τελικά σημεία API απαιτούν διαφορετικές λειτουργίες. Για να αλληλεπιδράσουμε με το API Kibana, χρησιμοποιούμε τις ακόλουθες λειτουργίες:

- GET - Ανακτά τις πληροφορίες.
- POST - Προσθέτει νέες πληροφορίες.
- PUT - Ενημερώνει τις υπάρχουσες πληροφορίες.
- DELETE - Αφαιρεί τις πληροφορίες.

Στη συνέχεια αναλύουμε μια σειρά από παραδείγματα που παρουσιάζουν τις βασικότερες δυνατότητες του Dev Tools στο κομμάτι της αναζήτησης. Υπάρχουν δύο βασικοί τρόποι αναζήτησης στο Elasticsearch. Queries και Aggregations.

### 3.2.5 Queries

Τα Querries (Ερωτήματα) επιστρέφουν δεδομένα που ικανοποιούν συγκεκριμένα κριτήρια.

```
GET /_search
{
  "query": {
    "match_all": {}
  }
}
```

Κώδικας 3.15: API αίτηση για αναζήτηση Δεδομένων

Το παραπάνω κομμάτι κώδικα ζητά από το σύστημα να επιστρέψει κάθε δεδομένο που είναι αποθηκευμένο στη βάση του Elasticsearch. Το αποτέλεσμα της εντολής αυτής παράγει ένα Json μορφής αποτέλεσμα.

The screenshot shows the Elasticsearch Dev Tools interface with the 'Console' tab selected. The query '{ \"query\": { \"match\_all\": {} } }' is entered in the text input field. The results pane displays two document hits. The first hit is from the index 'via-kibana' with ID '10051E89Mhcdcc2el'. It contains fields like '\_id', '\_index', '\_score', '\_source', 'CO2 (ppm)', 'humidity (%)', 'temperature (°C)', 'timestamp', 'Web (ppm)', and 'ambient\_light (lux)'. The second hit is from the index 'ui-kibana' with ID '10051E89Mhcdcc2el'. It contains similar fields. Both hits have a score of 1.0 and an '\_index' of 'ui-kibana'.

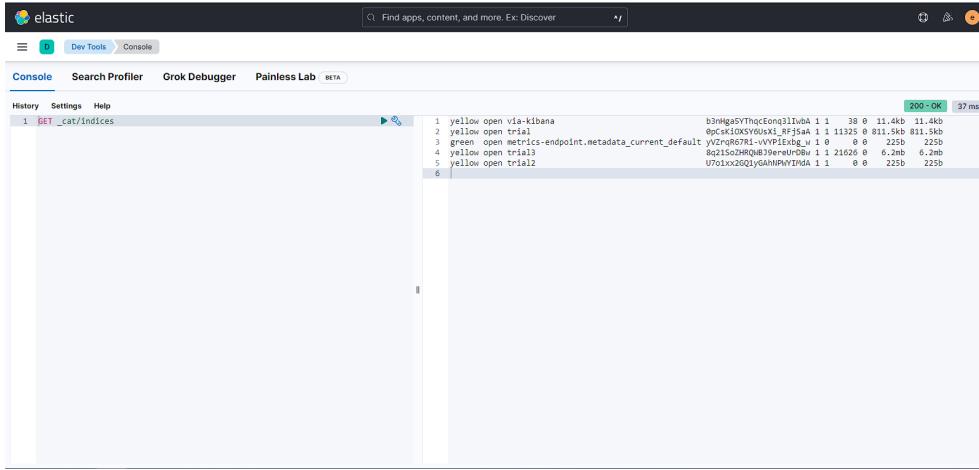
```
1: {
  "took": 10,
  "timed_out": false,
  "_shards": 1,
  "_total": 1,
  "_score": 1,
  "_source": {
    "CO2 (ppm)": 2321,
    "humidity (%)": 65,
    "temperature (°C)": 29.21,
    "timestamp": "2022-02-10 11:58:38",
    "Web (ppm)": 2,
    "ambient_light (lux)": 30,
    "timestamp": "2022-02-10T09:58:30.000+02:00",
    "ambient_noise (dB)": 49
  }
}
{
  "_index": "ui-kibana",
  "_id": "10051E89Mhcdcc2el",
  "_score": 1.0,
  "_source": {
    "CO2 (ppm)": 2321,
    "humidity (%)": 65,
    "temperature (°C)": 27.46,
    "timestamp": "2022-02-10 09:34:16",
    "entity_name": "Phytobiomics-RTH1",
    "Web (ppm)": 2,
    "ambient_light (lux)": 30,
    "timestamp": "2022-02-10T09:34:16.000+02:00",
    "ambient_noise (dB)": 49
  }
}
```

Σχήμα 3.16: Αποτέλεσμα αναζήτησης στο Dev Tools.

```
GET /_cat/indices
```

Κώδικας 3.16: Επιστροφή όλων των ευρετηρίων

Η εντολή αυτή μας επιστρέφει όλα τα ευρετήρια που υπάρχουν στη βάση δεδομένων του Elasticsearch. Παρέχει το αριθμό των γραμμών που έχει κάθε ευρετήριο καθώς και το χώρο που καταλαμβάνει.



Σχήμα 3.17: Εμφάνιση όλων των ευρετηρίων.

```
GET via-kibana/_search
{
  "query": {
    "match_all": {}
  },
  "size": 20
}
```

Κώδικας 3.17: Επιστροφή όλων το δεδομένων ενός ευρετηρίου

Η αναζήτηση που κάνουμε στο κώδικα 3.17 αφορά το ευρετήριο με όνομα "via-kibana". Η ερώτηση (query) που κάνουμε στο σύστημα μας επιστρέφει όλα τα δεδομένα ενός συγκεκριμένου ευρετηρίου. Το size : 20 είναι μια επιπλέον προσθήκη η οποία επιτρέπει την εμφάνιση περισσότερων αποτελεσμάτων στο πλαϊνό πίνακα του Dev Tools. Από προεπιλογή το Kibana εμφανίζει τα 10 πρώτα.

```

1+ [
2   "took" : 0,
3   "timed_out" : false,
4   "shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : {
11    "total" : {
12      "value" : 38,
13      "relation" : "eq"
14    },
15    "max_score" : 1.0,
16    "hits" : [
17      {
18        "_index" : "via-kibana",
19        "_id" : "1@posiEBXPwKcdqccZel",
20        "_score" : 1.0,
21        "_source" : {
22          "CO2 (ppm)" : 2173,
23          "humidity (%)" : 34,
24          "temperature (°C)" : 29.21,
25          "@timestamp" : "2022-02-10 11:58:30",
26          "Entity Name" : "Phytobiotics-RTH1",
27          "NO2 (ppm)" : 2,
28          "ambient_light (lux)" : 30,
29          "@timestamp" : "2022-02-10T11:58:30.000+02:00",
30          "ambient_noise (dB)" : 49
31        }
32      },
33      {
34        "_index" : "via-kibana",
35        "_id" : "ED09zEBXPwKcdqccZel",
36        "_score" : 1.0,
37        "_source" : {
38          "CO2 (ppm)" : 3321,
39          "humidity (%)" : 65,
40          "temperature (°C)" : 27.46,
41          "@timestamp" : "2022-02-21 09:14:16",
42          "Entity Name" : "Phytobiotics-RTH1",
43          "NO2 (ppm)" : 2,
44          "ambient_light (lux)" : 30,
45          "@timestamp" : "2022-02-21T09:14:16.000+02:00",
46          "ambient_noise (dB)" : 60
47        }
}

```

Σχήμα 3.18: Εμφάνιση των δεδομένων ενός ευρετηρίου.

```

GET via-kibana/_search
{
  "query": {
    "match": {
      "ambient_noise (dB)": "71"
    }
  },
  "size": 100
}

```

Κώδικας 3.18: Επιστροφή δεδομένων με παράγοντες

```

GET trial/_search
{
  "query": {
    "match": {
      "_id" : "10"
    }
  }
}

```

Κώδικας 3.19: Επιστροφή δεδομένων με συγκεκριμένο ID

Στους κώδικες 3.18 και 3.19 παρατηρείται ότι είναι εφικτή η αναζήτηση με βάση μια συγκεκριμένη τιμή ή ένα συγκεκριμένο ID. Έτσι ο κώδικας 3.18 επιστρέφει όλες τις γραμμές του ευρετηρίου via-kibana που η τιμή του ambient noise είναι 71, ενώ στην 2η περίπτωση επιστρέφονται τα στοιχεία του ευρετηρίου trial που έχουν id 10.

```
GET trial/_search
{
  "size": 1000,
  "query": {
    "range": {
      "temperature (°C)": {
        "gte": 17,
        "lte": 19
      }
    }
  }
}
```

Κώδικας 3.20: Χρήση φίλτρου για τα δεδομένα

Στο παρόδειγμα εισάγεται μια νέα δυνατότητα, αυτή του πεδίου (range). Σε περίπτωση αναζήτησης αποτελέσματος σε ένα εύρος θερμοκρασίας, εφαρμόζοντας το range μας δίνεται η δυνατότητα να οριστεί μια μέγιστη και ελάχιστη τιμή για το εύρος. Ως αποτέλεσμα θα ληφθούν οι σειρές του ευρετηρίου trial όπου η θερμοκρασία είναι ανάμεσα στις τιμές 17 και 19.

```
GET trial/_search
{
  "size": 1000,
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "ambient_noise (dB)": "71"
          }
        },
        {
          "range": {
            "temperature (°C)": {
              "gte": 17,
              "lte": 19
            }
          }
        }
      ]
    }
  }
}
```

```
}
```

### Κώδικας 3.21: Σύνθετη αναζήτηση

Στην περίπτωση αυτή κάνουμε μια πιο σύνθετη αναζήτηση με πολλαπλά ορίσματα. Η εφαρμογή του bool μας επιτρέπει να αναζητήσουμε δεδομένα με μεγαλύτερη ακρίβεια κάνοντας την πιο λεπτομερή. Η χρήση του bool επιτρέπει την έμμεση χρήση των τελεστών (AND,OR,NOT). Οι τελεστές αυτοί μπορούν να χρησιμοποιηθούν για τη βελτίωση των ερωτημάτων αναζήτησής μας προκειμένου να παρέχουμε πιο σχετικά ή συγκεκριμένα αποτελέσματα. Αυτό υλοποιείται στο API αναζήτησης ως ερώτημα bool, το οποίο δέχεται τις εξής παραμέτρους:

- "must". Ισοδύναμο με AND
- "should". Ισοδύναμο με OR
- "must\_not". Ισοδύναμο με NOT

Συνεπώς στο παράδειγμα μας το "must" υποδηλώνει ότι τα αποτελέσματα που εμφανίζονται ικανοποιούν ταυτόχρονα και τις δύο συνθήκες. Να προσθέσουμε ότι η ύπαρξη δύο παραμέτρων τύπου bool είναι επιτρεπτή αρκεί να τηρηθεί η σωστή δομή του query όπως φαίνεται στο παράδειγμα 3.21.

```
GET trial/_search
{
  "size": 1000,
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "ambient_noise (dB)": "71"
          }
        },
        {
          "range": {
            "temperature (°C)": {
              "gte": 17,
              "lte": 19
            }
          }
        }
      ],
      "must_not": [
        {
          "match": {

```

```

        "NH3 (ppm)": "3"
    }
}
]
}
}
}
```

Κώδικας 3.22: Σύνθετη αναζήτηση με πολλαπλές παραμέτρους

### 3.2.6 Aggregations

Σε περίπτωση που ο χρήστης επιθυμεί να βρει το μέγιστο ή το ελάχιστο μιας τιμής, το άθροισμα ή τον μέσο όρο χρησιμοποιούνται τα Aggregations. Η χρήση των Aggregations μας δίνει την δυνατότητα να παρουσιάσουμε τη σύνοψη των δεδομένων μας ως μετρήσεις, στατιστικά στοιχεία και άλλα αναλυτικά στοιχεία.

```
GET trial/_search
{
  "aggs": {
    "my_agg": {
      "sum": {
        "field": "temperature (°C)"
      }
    }
  }
}
```

Κώδικας 3.23: Χρήση συναθροίσεων (Aggregations) για εύερση αθροίσματος

Χρησιμοποιώντας τη παράμετρο aggs αντί του query είμαστε σε θέση να εφαρμόζουμε ερωτήσεις μετρήσεων. Στο παρόντο παράδειγμα αυτό ζητάμε το άθροισμα των θερμοκρασιών στο ευρετήριο trial. Το αποτέλεσμα του aggregation έχει την μορφή `{ "sum": { "field": "temperature (°C)" } }`, πάντα στο τέλος του json-formatted κείμενο που παράγει το Dev Tools.

```

1 ▾ {
2   "took" : 14,
3   "timed_out" : false,
4 ▾ "_shards" : {
5   "total" : 1,
6   "successful" : 1,
7   "skipped" : 0,
8   "failed" : 0
9 ▾ },
10 ▾ "hits" : { },
169 ▾ "aggregations" : {
170 ▾ "my_agg" : {
171   "value" : 272622.94
172 ▾ }
173 ▾ }
174 ▾ }
175

```

Σχήμα 3.19: Το αποτέλεσμα ενός aggregation.

Παρόμοια στους κώδικες 3.24 και 3.25 που ακολουθούν, αναζητούμε την ελάχιστη και μέγιστη τιμή της θερμοκρασίας και της υγρασίας των ευρετηρίων που επιλέξαμε. Οι μετρήσεις αυτές επιτυγχάνονται με την χρήση των παραμέτρων "min" και "max".

```

GET via-kibana/_search
{
  "aggs": {
    "my_agg": {
      "min": {
        "field": "temperature (°C)"
      }
    }
  }
}

```

Κώδικας 3.24: Εύρεση ελαχίστου ενός πεδίου

```

GET trial/_search
{
  "aggs": {
    "my_agg": {
      "max": {
        "field": "humidity (%)"
      }
    }
  }
}

```

```

    }
}
}
```

Κώδικας 3.25: Εύρεση μεγίστου ενός πεδίου

Παρατηρούμε όμως ότι η εύρεση τιμών ξεχωριστά μπορεί να αποβεί μια χρονοβόρα διαδικασία. Το παρακάτω παράδειγμα μας δίνει την δυνατότητα να βρούμε τέτοιου είδους πληροφορίες χωρίς να χρειάζεται να αλλάξουμε τις παραμέτρους ανάλογα με τις ανάγκες.

```
GET trial/_search
{
  "aggs": {
    "my_agg": {
      "stats": {
        "field": "temperature (°C)"
      }
    }
  }
}
```

Κώδικας 3.26: Ταυτόχρονη επιστροφή ελάχιστου, μεγίστου, μέσου όρου, αύριοίσματος

Βλέπουμε στην παρακάτω εικόνα ότι η παρόμετρος stats επιστρέφει την τιμή count που αντιπροσωπεύει τις σειρές του ευρετηρίου που υπάρχει τιμή θερμοκρασίας, την ελάχιστη και μέγιστη θερμοκρασία που έχει καταγραφεί, τη μέση τιμή καθώς και το άθροισμα αυτής.

```

1 [
2   "took" : 7,
3   "timed_out" : false,
4   "_shards" : {
5     "total" : 1,
6     "successful" : 1,
7     "skipped" : 0,
8     "failed" : 0
9   },
10  "hits" : { },
169  "aggregations" : {
170    "my_agg" : {
171      "count" : 11325,
172      "min" : 7.46,
173      "max" : 34.69,
174      "avg" : 24.07266578366446,
175      "sum" : 272622.94
176    }
177  }
178 }
179

```

Σχήμα 3.20: Συνάθροιση (Aggregation) όλα σε ένα.

Τέλος τα aggregations μπορούν να συνδυαστούν με τα queries πραγματοποιώντας έτσι πιο σύνθετες και εξειδικευμένες αναζητήσεις, όπως στο παρόντα που ακολουθεί.

```

GET trial/_search
{
  "size": 0,
  "query": {
    "match": {
      "humidity (%)": "72"
    }
  },
  "aggs": {
    "avg_temperature_where_humitidy_is_": {
      "avg": {
        "field": "temperature (°C)"
      }
    }
  }
}

```

```

        }
    }
}

```

Κώδικας 3.27: Συνδυαστική αναζήτηση με Ερώτηση και Συνάθροιση (Query &amp; Aggregation)

```

1 ▾ {
2     "took" : 2,
3     "timed_out" : false,
4 ▾   "_shards" : {
5       "total" : 1,
6       "successful" : 1,
7       "skipped" : 0,
8       "failed" : 0
9 ▾     },
10 ▾   "hits" : {
11     "total" : {
12       "value" : 416,
13       "relation" : "eq"
14     },
15     "max_score" : null,
16     "hits" : [ ]
17   },
18 ▾   "aggregations" : {
19     "avg_temperature_where_humidity_is_" : {
20       "value" : 21.287211538461538
21     }
22   }
23 }
24

```

Σχήμα 3.21: Το αποτέλεσμα ενός query και agg.

Η αναζήτηση που έγινε επιστρέφει το ζητούμενο, που είναι η μέση τιμή της θερμοκρασίας των 416 δειγμάτων στα οποία η υγρασία (humidity) αγγίζει το 72%.

Σε αυτό το σημείο να αναφέρουμε ότι οι παραπάνω κώδικες προϋπονθέτουν την ύπαρξη ευρετηρίου και δεδομένων. Κατά κύριο λόγο τόσο η δημιουργία όσο και η ύπαρξη δεδομένων καλύπτεται με τρόπους που έχουμε αναφέρει σε προηγούμενα κεφάλαια. Όμως το Kibana δίνει την δυνατότητα στον χρήστη να δημιουργήσει νέο ευρετήριο και να προσθέσει χειροκίνητα δικά του στοιχεία. Η προσθήκη νέων στοιχείων σε ήδη υπάρχων ευρετήριο θα πρέπει να συμβαδίζει με τις παραμέτρους του mapping του ευρετηρίου. Αντιθέτως αν το ευρετήριο δεν προϋπάρχει η αντιστοίχηση (mapping) δημιουργείται αυτόματα με την πρώτη εισαγωγή δεδομένων. Εκτελώντας αρχικά στο Dev Tools την μιας σειράς εντολή **PUT new-index/?pretty**, δημιουργείται το ευρετήριο το οποίο μπορούμε να ελέγξουμε με την εντολή **GET \_cat/indices**.

```
PUT new-index/_doc/0
{
  "name": "John",
  "surname": "Smith",
  "email" : "mail@gmail.com",
  "phone_number": 6900000000
}
```

Κώδικας 3.28: Εισαγωγή δεδομένων σε ευρετήριο (Index)

Η είσοδος νέων στοιχείων στο ευρετήριο που δημιουργήσαμε γίνεται με το POST. Επιλέγοντας το όνομα του ευρετηρίου, ακολουθεί στη συνέχεια ο τύπος του ευρετηρίου και τέλος το μοναδικό ID. Εφόσον εκτελεστεί ο κώδικας δημιουργείται και η αντιστοίχηση (mapping) κάτι το οποίο μπορούμε να δούμε με την εντολή **GET new-index/\_mapping**.

Παρατηρώντας το σχήμα 3.22, συμπεραίνουμε ότι το Elasticsearch με την είσοδο δεδομένων στο ευρετήριο δημιουργησε αυτόματα την αντιστοίχηση, λαμβάνοντας υπόψη το είδος της τιμής του κάθε πεδίου.

```
1 {  
2   "new-index" : {  
3     "mappings" : {  
4       "properties" : {  
5         "email" : {  
6           "type" : "text",  
7           "fields" : {  
8             "keyword" : {  
9               "type" : "keyword",  
10              "ignore_above" : 256  
11            }  
12          }  
13        },  
14        "name" : {  
15          "type" : "text",  
16          "fields" : {  
17            "keyword" : {  
18              "type" : "keyword",  
19              "ignore_above" : 256  
20            }  
21          }  
22        },  
23        "phone_number" : {  
24          "type" : "long"  
25        },  
26        "surname" : {  
27          "type" : "text",  
28          "fields" : {  
29            "keyword" : {  
30              "type" : "keyword",  
31              "ignore_above" : 256  
32            }  
33          }  
}
```

Σχήμα 3.22: Η αντιστοίχηση (mapping) ενός ευρετηρίου.

Τέλος, η διαγραφή ευρετηρίων και σειρών/δεδομένων ενός ευρετηρίου είναι εφικτή μέσα από την διεπαφή του Kibana. Εκτελώντας τις εντολές **DELETE new-index** και **DELETE new-index/\_doc/0**, διαγράφουμε στην πρώτη περίπτωση το συγκεκριμένο ευρετήριο με όλα του τα δεδομένα και στη συνέχεια διαγράφεται η σειρά του ευρετηρίου με το ID που δίνεται στο τέλος. Να αναφέρουμε εδώ ότι σε περίπτωση που το ID δεν έχει οριστεί από τον χρήστη, το Elasticsearch, ορίζει αυτόματα ένα τυχαίο κάνοντας έτσι δυσκολότερη την εύρεση του επιθυμητού στοιχείου που επιθυμούμε να διαγράψουμε.

## 3.3 Django

### 3.3.1 Εγκατάσταση Django

Στα πλαίσια της παρούσας διπλωματικής εργασίας για λόγους που αναφέρονται στο επόμενο κεφάλαιο, το Django εγκαθίσταται και χρησιμοποιείται σε εικονικό περιβάλλον (Virtual-Environment) καθώς όλη την διάρκεια της ανάπτυξης (Development), ενώ η τελική έκδοση μεταφέρεται σε Docker.

### 3.3.2 Το Django σε Virtual-Env

Ένα εικονικό περιβάλλον είναι ένα εργαλείο που βοηθά στη διατήρηση των εξαρτήσεων (Βιβλιοθήκες, πρόσθετα) που απαιτούνται από διαφορετικά έργα ξεχωριστά, δημιουργώντας για αυτά απομονωμένα εικονικά περιβάλλοντα python. Είναι ένα από τα πιο σημαντικά εργαλεία που χρησιμοποιούν οι περισσότεροι προγραμματιστές Python και ενδείκνυται σε περιπτώσεις που υπάρχουν πολλαπλά έργα στο ίδιο σύστημα. Η διαδικασία εγκατάστασης πραγματοποιείται σε λειτουργικό Windows.

Έχοντας επιλέξει το φάκελο που θα γίνει η εγκατάσταση του εικονικού περιβάλλοντος ο χρήστης στη γραμμή εντολών (Terminal) εκτελεί τις ακόλουθες εντολές.

- **python -m venv my\_env:** Η συγκεκριμένη εντολή δημιουργεί και εγκαθιστά όλα τα απαραίτητα εκτελέσιμα αρχεία για τη χρήση των πακέτων που θα χρειαζόταν ένα έργο που αναπτύσσεται με τη γλώσσα προγραμματισμού Python. Εναλλακτικά μπορεί να χρησιμοποιηθεί η προσέγγιση μέσω pip<sup>38</sup> (σύστημα διαχείρισης πακέτων βιβλιοθηκών) εκτελώντας αντί της προαναφερθείσας εντολής τις επόμενες δύο.

1. **pip install virtualenv**
2. **virtualenv my\_env**

- **my\_env\Scripts\activate.bat:** Η εντολή αυτή ενεργοποιεί το εικονικό περιβάλλον για τον χρήστη και μέσα σε αυτό υλοποιείται το έργο. (Πρόκειται για ένα script που στην ουσία αποτελείται από ένα σύνολο εντολών). Εκτελώντας την εντολή **deactivate** απενεργοποιείται το εικονικό περιβάλλον.

```
C:\Users\user\Desktop\testing>my_env\Scripts\activate.bat
(my_env) C:\Users\user\Desktop\testing>
```

Σχήμα 3.23: Η ενεργοποίηση του εικονικού περιβάλλοντος

- **pip install Django** : Γίνεται εγκατάσταση του Django μέσα στο εικονικό περιβάλλον.
- **django-admin startproject my\_project\_name** : Εκτελώντας την συγκεκριμένη εντολή δημιουργείται το project με το όνομα που έχει δώσει ο χρήστης. Ο συγκεκριμένος φάκελος ή app όπως αποκαλούνται οι υποφάκελοι ενός Django Project περιέχει όλα τα απαραίτητα αρχεία όπως Settings , urls αλλά και απαραίτητες πληροφορίες διαχειριστή , που αποτελούν το πυρήνα το έργου και δίνουν την δυνατότητα στο χρήστη να αναπτύξει την εφαρμογή του.
- **python manage.py migrate** : Στο σημείο αυτό και έχοντας περάσει στο φάκελο του έργου (π.χ (my\_env) C:\Users\user\Desktop\testing\my\_project\_name) εκτελείται η εντολή η οποία εγκαθιστά την βάση δεδομένων sqlite3<sup>39</sup> με όλους τους απαραίτητους πίνακες και πεδία για τον διαχειριστή (Administration Tables).
- **python manage.py makemigrations** : Η χρήση της είναι απαραίτητη όταν πραγματοποιούνται αλλαγές (Εισαγωγή, Ενημέρωση, Διαγραφή) στη βάση δεδομένων του Project.
- **python manage.py runserver** : Πραγματοποιείται η εκτέλεση της εφαρμογής (application). Το Django από προεπιλογή έχει ως θύρα την 8000 για πρόσβαση στο γραφικό περιβάλλον της διαδικτυακής εφαρμογής. Συνεπώς αν το σύστημα τρέχει σε τοπικό περιβάλλον (localhost) και όχι σε κάποιον server, το έργο είναι προσβάσιμο πληκτρολογώντας σε ένα φυλλομετρητή (Browser) την σελίδα **localhost:8000**.
- **python manage.py startapp app's\_name** : Η συγκεκριμένη εντολή προσφέρει την δυνατότητα ανάπτυξης και συνύπαρξης ενός νέου app μέσα στο ήδη υπάρχον project. Αναγκαία η χρήση της σε έργα μεγάλου εύρους.

Οι παραπάνω εντολές καλύπτουν τα πρώτα απαραίτητα βήματα που πρέπει να ακολουθηθούν για την δημιουργία ενός εικονικού περιβάλλοντος και της εγκατάστασης του Django σε αυτό. Να σημειωθεί ότι το εικονικό περιβάλλον δεν αποτελεί απαραίτητο εργαλείο για την λειτουργία του Django καθώς μπορεί να λειτουργήσει εξίσου καλά και χωρίς αυτό με την προϋπόθεση να μην εκτελούνται παραπάνω από ένα έργο Python την ίδια χρονική στιγμή.

### 3.3.3 Η μεταφορά του Django σε Docker

Η μεταφορά του Django σε Docker γίνεται με σκοπό την εκμετάλλευσή της φορητότητας που προσφέρει το Docker μεταξύ συστημάτων (ηαρδωαρε καθώς και λειτουργικών συστημάτων). Ένα Project στην τελική του μορφή μπορεί να καταλαμβάνει πολύ χώρο στη μνήμη

ενώ μπορεί να αξιοποιεί ταυτόχρονα πολλές τεχνολογίες και να έχει αρκετές εξαρτήσεις από άλλα λογισμικά (dependencies) γεγονός που καθιστά την μεταφορά σε άλλο σύστημα αρκετά χρονοβόρα. Έτσι, η ανάπτυξη ενός Django έργου σε ένα Docker Container θεωρείται καλή πρακτική. Δημιουργώντας ένα Dockerfile στο Directory του Django δημιουργείται το κατάλληλο Docker Image και το project θεωρείται πλέον φορητό.

```

1 # pull the official base image
2 FROM python:3.9.13
3
4 # set work directory
5 WORKDIR /usr/src/app
6
7 # set environment variables
8 ENV PYTHONDONTWRITEBYTECODE 1
9 ENV PYTHONUNBUFFERED 1
10
11 # install dependencies
12 RUN pip install --upgrade pip
13 COPY ./requirements.txt /usr/src/app
14 RUN pip install -r requirements.txt
15
16 # copy project
17 COPY . /usr/src/app
18
19 EXPOSE 8000
20
21 CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

Κώδικας 3.29: Δημιουργία του Django Dockerfile για το image.

Με την χρήση του παραπάνω κώδικα επιτυγχάνεται η δημιουργία του Docker Image, το οποίο πρακτικά περικλείει ολόκληρο το έργο που έχει ήδη δημιουργηθεί στο εικονικό περιβάλλον. Με αυτό τον τρόπο η μεταφορά σε άλλο σύστημα ή η χρήση του από κάποιο άλλον χρήστη είναι αρκετά πιο εύκολη. Επιπρόσθετα με το Container που θα δημιουργηθεί από το συγκεκριμένο image, αποφεύγεται η εκτέλεση των πολλαπλών εντολών για την ενεργοποίηση του Project στη γραμμή εντολών (Terminal) αυτοματοποιώντας με τον τρόπο αυτό την έναρξη του έργου. Αναλυτικότερα:

- (**Γραμμές 5,17**): Δηλώνεται το μονοπάτι στο οποίο θα περαστούν όλα τα αρχεία του έργου που υπάρχουν στο εικονικό περιβάλλον. Το Container που προκύπτει στην συνέχεια περιέχει αρχεία και μονοπάτια (paths) που δεν έχουν σχέση με το Django οπότε για διευκόλυνση ορίζεται ένα μονοπάτι ευρέως γνωστό.
- (**Γραμμές 12-14**): Πραγματοποιείται η εγκατάσταση των βιβλιοθηκών. Κατά την διάρκεια υλοποίησης της διπλωματικής εργασίας, εγκαταστάθηκε ένας μεγάλος αριθμός τρίτων βιβλιοθηκών που είναι απαραίτητοι για την επίτευξη του επιθυμητού αποτελέσματος. Οι βιβλιοθήκες αυτές είναι απαραίτητο να μεταφερθούν και στο image που θα

δημιουργηθεί. Αυτό πραγματοποιείται αντιγράφοντας το αρχείο requirements.txt στο μονοπάτι /usr/src/app. Η μεταφορά των βιβλιοθηκών πραγματοποιείται με την εκτέλεση της εντολής **pip freeze -> requirements.txt** στο εικονικό περιβάλλον. Τέλος η εγκατάσταση τους ζεχινά στην γραμμή 14.

- (**Γραμμή 21**): Η εντολή CMD σε ένα Dockerfile δηλώνει το σύστημα ότι επρόκειτο για εντολή που τρέχει στην γραμμή εντολών. Συνεπώς στην περίπτωση αυτή εκτελείται αυτόματα με την εκκίνηση του Container που θα δημιουργηθεί, η εντολή **python manage.py runserver** στο localhost με πύλη 8000.

### 3.3.4 Λειτουργίες της Πλατφόρμας

Στην ενότητα αυτή γίνεται αναφορά στις βασικές λειτουργίες της πλατφόρμας που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας για το Project Phytobiotics. Η πλατφόρμα αποτελείται από βασικές σελίδες δημιουργίας λογαριασμού, ταυτοποίησης και ενημέρωσης Profile, ανέβασμα αρχείων του χρήστη σε Elasticsearch - Kibana και το πιο βασικό, αναπαράσταση γραφημάτων με χρήση βιβλιοθηκών της Python βασισμένα στα ανεβασμένα αρχεία με τα οποία τροφοδοτεί ο χρήστης τη πλατφόρμα.

### 3.3.5 Βασικές Δυνατότητες Πλατφόρμας

#### Εγγραφή (Register) - Είσοδος (Login)<sup>404142</sup>

Ο χρήστης κατά την είσοδο του στην πλατφόρμα, καλείται να δημιουργήσει ένα λογαριασμό δίνοντας κάποια στοιχεία έτσι ώστε να είναι διαθέσιμες σε αυτόν οι σημαντικότερες λειτουργίες της εφαρμογής. Η φόρμα ακολουθεί βασικούς κανόνες συμπλήρωσης στοιχείων που καθοδηγούν τον χρήστη στην συμπλήρωση αποδεκτών από το σύστημα στοιχείων. Παράλληλα με την εγγραφή στη πλατφόρμα του Django, εκτελείται με αυτοματοποιημένο τρόπο στο παρασκήνιο και η εγγραφή στο Kibana. Έτσι, με τα ίδια δεδομένα εγγραφής, παρέχεται στον χρήστη άμεση πρόσβαση στις λειτουργίες του Kibana αποφεύγοντας εν συνεχεία την επανάληψη της διαδικασίας εγγραφής.

```

1 def register(request):
2     if request.method == 'POST':
3         form = UserRegisterForm(request.POST)
4         if form.is_valid():
5             form.save()
6             username = form.cleaned_data.get('username')
7             password = form.cleaned_data.get('password1')
8             email = form.cleaned_data.get('email')
9             #return HttpResponse(password)
10            register_kibana_user(username, password, email)
11
12            return redirect('Login')
13        else:
14            form = UserRegisterForm()
15            return render(request, 'users/register.html',{'form': form})

```

Κώδικας 3.30: Συνάρτηση εγγραφής στην πλατφόρμα του Django.

- Ο παραπάνω κώδικας ελέγχει την εγκυρότητα της φόρμας αξιοποιώντας το πρόσθετο εργαλείο Crispy Forms<sup>43</sup> και πριν την ολοκλήρωση της καλείται η συνάρτηση που αφορά την εγγραφή στο Kibana. Εφόσον η διαδικασία κριθεί επιτυχής, γίνεται ανακατεύθυνση (redirect) στην σελίδα εισόδου (Login Page). Σε αντίθετη περίπτωση, δεν πραγματοποιείται καμία ενέργεια μέχρις ότου ο χρήστης να κάνει σωστά την εγγραφή.

```

1 def register_kibana_user(username, password, email):
2     response = requests.get('https://host.docker.internal:9200',
3                             verify=False,
4                             auth=HTTPBasicAuth('elastic', 'elastic'))
5
6
7     data = {
8         "password": password,
9         "enabled": True,
10        "roles": ["superuser", "kibana_admin"],
11        "full_name": username,
12        "email": email,
13        "metadata": {
14            "intelligence": 7
15        }
16    }
17
18    headers = {
19        'content-type': 'application/json'
20    }
21
22    name = username
23    r = requests.post('https://host.docker.internal:9200/_security/user/' +
24                      name, json=data, headers=headers, verify=False, auth=('elastic', 'elastic'))

```

Κώδικας 3.31: Συνάρτηση εγγραφής στο Kibana.

- (**Γράμμη 2**): Γίνεται εγκαθίδρυση επικοινωνίας μεταξύ Django και Elasticsearch. Η διεύθυνση που δίνεται εξαρτάται από το που έχει στηθεί το Elasticsearch. Στην προκειμένη, το Elasticsearch βρίσκεται σε τοπικό περιβάλλον. Τα διαπιστευτήρια που δίνονται για τη συνάρτηση ελέγχου ταυτότητας χρήστη βασικής μορφής HTTPBasicAuth προκύπτουν από την δημιουργία των των χαρακτηριστικών ασφαλείας που περιγράφονται στο κεφάλαιο του Docker (Κεφ 2). Εφόσον η επικοινωνία μεταξύ Django και Elasticsearch είναι επιτυχής, το σύστημα επιστρέφει Response 200.
- (**Γραμμές 7-20**): Συμπληρώνονται όλα τα απαραίτητα στοιχεία για την εγγραφή στο Kibana. Τα δεδομένα που έχει εισάγει ο χρήστης περνούν σε Python Dictionaries<sup>44</sup>. Παρατηρείται επίσης ότι εκτός από τα δεδομένα του χρήστη (Κωδικός Πρόσβασης, όνομα, email) δίνονται και κάποιοι ρόλοι (Roles) για το προφίλ στο (Kibana).
- (**Γραμμή 23**): Πρόκειται για την εντολή που πραγματοποιεί την δημιουργία χρήστη (user). Στην πραγματικότητα το Kibana παρέχει API για την αυτόματη δημιουργία χρήστη. Η ‘μετατροπή’ του API αυτού σε γλώσσα Python γίνεται μέσω της βιβλιοθήκης request.

### Είσοδος (Login) - Προφίλ (Profile)

Μετά την διαδικασία εγγραφής σε Django και Kibana ο χρήστης καλείται να πραγματοποιήσει την είσοδο του στην πλατφόρμα για να μπορέσει να εκμεταλλευτεί στο μέγιστο τις δυνατότητες της, όπως το ανέβασμα των αρχείων του αλλά και αναπαράσταση γραφημάτων. Επιπρόσθετα του δίνεται η δυνατότητα ανανέωσης των δεδομένων που έχει εισάγει και γενικότερα μια πλήρη επεξεργασία του προφίλ του.

#### 3.3.6 Ανέβασμα Αρχείων στο Elasticsearch (Uploading)

Με την πραγματοποίηση της εισόδου στην πλατφόρμα ο χρήστης είναι σε θέση να ανεβάσει τα αρχεία που επιθυμεί για να είναι εφικτή η αναπαράσταση γραφημάτων. Επιλέγοντας το κατάλληλο αρχείο (τύπου json για την υλοποίηση που έγινε στα πλαίσια της παρούσας διπλωματικής), το σύστημα το ανεβάζει στη πλατφόρμα του Django και στο Elasticsearch. Απαραίτητη προϋπόθεση αποτελεί η επιλογή ονόματος ευρετηρίου από τον χρήστη στο οποίο θα αποθηκευτούν τα δεδομένα του αρχείου που επιλέχθηκε. Έπειτα, πριν ολοκληρωθεί η πρώτη θηση του αρχείου στο Elasticsearch, γίνεται αναγνώριση του τύπου δεδομένων του αυτού, το λεγόμενο mapping. Πρόκειται για αναγκαίο βήμα καθώς έτσι το Elasticsearch καθιστά τα δεδομένα διαθέσιμα για οπτικοποίηση. Με το πέρας της διαδικασίας αυτής, ο χρήστης παροτρύνεται να δει τα γραφήματα που δημιουργούνται από τα δεδομένα του.

```

1 @login_required
2 def BookUploadView(request):
3     if request.method == 'POST':
4         form = UploadFilesForm(request.POST, request.FILES)
5         if form.is_valid():
6             form.save()
7             # Here is upload to elk function
8             index= form.cleaned_data['Index_Name']
9             files = form.cleaned_data['files'].name
10
11             bokeh_upload_file(files)
12             body = json_data_types(files)
13
14             ##### FOR THE ORIGINAL VERSION #####
15             #to_elastic(index,files,body)
16             ##### FOR THE SECOND VERSION #####
17             temp = to_elastic(index,files,body)
18             ##### FOR THE SECOND VERSION #####
19             delete_file(temp)
20
21             create_data_view(index)
22
23             context = {
24                 'form':form,
25                 'success': 'yes'
26             }
27         else:
28             form = UploadFilesForm(request.POST, request.FILES)
29             context = {
30                 'form':form,
31                 'success': 'no'
32             }
33
34     return render(request, 'elk/upload_to_elastic.html', context)

```

Κώδικας 3.32: Η φόρμα ανεβάσματος αρχείου από τον χρήστη.

- (**Γραμμή 4**): Πραγματοποιείται η αποθήκευση των δεδομένων εισόδου στο σύστημα. Επιστρέφονται για το ανέβασμα στο elasticsearch.
- (**Γραμμές 5-9**): Αν η φόρμα είναι έγκυρη και ο χρήστης έδωσε σωστά τα στοιχεία, η φόρμα αποθηκεύεται και στη συνέχεια περνιούνται σε ξεχωριστές μεταβλητές οι τιμές εισόδου.
- (**Γραμμή 12**): Στο σημείο αυτό ξεκινάει η διαδικασία εύρεσης του τύπου δεδομένων του αρχείου του χρήστη, περνώντας ως είσοδο την μεταβλητή files. Η ανάλυση της παρούσας συνάρτησης πραγματοποιείται στην συνέχεια.

- (**Γραμμές 17,21**): Το κάλεσμα της συνάρτησης που πραγματοποιεί το ανέβασμα του αρχείου στο Elasticsearch. Στην γραμμή 21 πραγματοποιείται η δημιουργία ενός Data View το οποίο αποτελεί στην ουσία ένα πιστοποιητικό επαλήθευσης εγκυρότητας το οποίο επιτρέπει στο kibana να πάρει το ευρετήριο από το Elasticsearch επιτρέποντας έτσι την οπτικοποίηση του από τον χρήστη.
- (**Γραμμές 27-32**): Σε περίπτωση ανεπιτυχούς προσπάθειες το σύστημα φορτώνει εκ' νέου και ο χρήστης επαναλαμβάνει την διαδικασία.

```

1 def to_elastic(variable1,variable2, variable3):
2     i =0
3
4     variable22 = "to_be_deleted_" + variable2
5     default_storage.open(os.path.join('files', variable22), 'w')
6     shutil.copy(default_storage.path(os.path.join('files', variable2)),
7                  default_storage.path(os.path.join('files', variable22)))
8     with default_storage.open(os.path.join('files', variable22)) as f:
9         # Reading from file
10        data = json.loads(f.read())
11
12
13    with default_storage.open(os.path.join('files', variable22), 'w') as f:
14        for i in range(len(data)):
15            json.dump({"index": {"_id": i}}, f)
16            f.write('\n')
17            json.dump((data[i]), f)
18            f.write('\n')
19
20
21    response = requests.get('https://host.docker.internal:9200', verify=
22                           False, auth=HTTPBasicAuth('elastic', 'elastic'))
23
24
25    payload = default_storage.open(os.path.join('files', variable22), 'rb')
26    headers = {
27        'content-type': 'application/json'
28    }
29    r = requests.put('https://localhost:9200/' + variable1+ '?pretty',
30                      verify=False, auth=('elastic', 'elastic'))
31
32    r = requests.put('https://localhost:9200/' + variable1 + '/_mapping',
33                      headers=headers, verify=False, auth=('elastic', 'elastic'), json=
34                      variable3)
35
36    r = requests.post('https://localhost:9200/dummy_index/_bulk?pretty',
37                      headers=headers, data=payload, verify=False, auth=('elastic', 'elastic'
38 ))
39
40
41    new_index = {
42        "source": {
43

```

```

34         "index": "dummy_index",
35         "query": {
36             "match_all": {}
37         }
38     },
39
40     "dest": {
41         "index": variable1
42     }
43 }
44
45 time.sleep(5)
46 r = requests.post('https://localhost:9200/_reindex', headers=headers,
47 json=new_index, verify=False, auth=HTTPBasicAuth('elastic', 'elastic'))
48
49 return variable2

```

Κώδικας 3.33: Το ανέβασμα του αρχείου στο Elasticsearch.

- (**Γραμμή 1**): Οι μεταβλητές που ορίζονται αντιπροσωπεύουν το ευρετήριο, το αρχείο και το σώμα του mapping.
- (**Γραμμές 3-8**): Στο σημείο αυτό πραγματοποιείται η δημιουργία ενός προσωρινού αντιγράφου του αρχείου του χρήστη. Αυτό συμβαίνει διότι το αρχείο πρέπει πρώτα να ευρετηριαστεί (κάθε γραμμή δεδομένων να λάβει ένα μοναδικό αναγνωριστικό ID) πριν περάσει στο Elasticsearch. Η ευρετηρίαση δεν γίνεται στο βασικό αρχείο γιατί δεν θα είναι εφικτή στη συνέχεια η οπτικοποίηση των δεδομένων από τις βιβλιοθήκες γραφημάτων της Python. Αυτό οφείλεται στον τρόπο με τον οποίο πραγματοποιείται η διαδικασία ευρετηρίασης στο Elasticsearch. Πιο συγκεκριμένα, κατά την ευρετηρίαση ενός json αρχείου, το Elasticsearch προσθέτει στο αρχείο αυτό επιπλέον στοιχεία τα οποία είναι απαραίτητα για την εκπλήρωση της ευρετηρίασης όπως το όνομα του ευρετηρίου και το μοναδικό ID. Έτσι, η Python δεν μπορεί να διαβάσει το τροποποιημένο json αρχείο. Σαν εναλλακτική λύση της δημιουργίας προσωρινού αντιγράφου του json αρχείου προς ευρετηρίαση είναι η εκ νέου ανάγνωση του αρχείου μετά την ευρετηρίαση και η επιλογή των γραμμών εκείνων που περιέχουν την αναγκαία πληροφορία. Δοκιμές όμως έδειξαν ότι η μεθόδος αυτή ήταν πιο χρονοβόρα της πρώτης μεθόδου σε αρχεία μεγάλου όγκου δεδομένων. Σε περιπτώσεις μεγάλου όγκου αρχείων και πιο συγκεκριμένα αρχείου των 11000 γραμμών, παρατηρήθηκε καθυστέρηση μέχρι και 5 δευτερολέπτων σε δοκιμαστικό περβάλλον προσωπικού υπολογιστή με hardware τα ακόλουθα μοντέλα (Επεξεργαστής Ryzen 5-2600, μνήμη RAM στα 8GB) και διεργασίες να τρέχουν στο παρασκήνιο όπως browser, Docker-Desktop, Antivirus.
- (**Γραμμές 11-16**): Εδώ παρατηρείται η διαδικασία ευρετηρίασης του προσωρινού αρχείου. Πιο συγκεκριμένα, στην γραμμή 13 γίνεται η τοποθέτηση μοναδικού ID για κάθε γραμμή αρχείου.

- (**Γραμμές 18- 29**): Ακολουθεί μια σειρά εντολών που χρησιμοποιούν το API του Kibana. Αρχικά εγκαθιδρύεται η σύνδεση με το Kibana. Εν συνεχεία στη γραμμή 25 δημιουργείται ένα κενό ευρετήριο με το δοιθέν όνομα του χρήστη και στην γραμμή 27 προστίθεται το mapping του αρχείου στο ευρετήριο αυτό. Τέλος στην γραμμή 29 δημιουργείται ένα προσωρινό ευρετήριο με όνομα dummy\_index στο οποίο εισάγονται τα επιθυμητά δεδομένα.
- (**Γραμμές 32-46**): Χρησιμοποιώντας ξανά το API του Kibana και μέσω της βιβλιοθήκης requests ξεκινά η διαδικασία αναπροσαρμογής του ευρετηρίου. Τα δεδομένα του προσωρινού ευρετηρίου περνούν στο τελικό ευρετήριο το οποίο ενημερώνεται παράλληλα στο Elasticsearch. Η μεταφορά από το αρχικό στο τελικό αρχείο ορίζεται στο Dictionary στη μεταβλητή new\_index.

### 3.3.7 Η οπτικοποίηση των δεδομένων στη πλατφόρμα

Στην παρούσα ενότητα προβάλλεται στο χρήστη το περιεχόμενο του αρχείου του μέσα από μια σειρά γραφημάτων. Επιπλέον, γίνεται αναφορά στις βιβλιοθήκες γραφημάτων της python που χρησιμοποιήθηκαν για την δημιουργία των γραφημάτων, τα χαρακτηριστικά της κάθε βιβλιοθήκης καθώς και το αποτέλεσμα της δημιουργίας των πινάκων (Dashboard) του Kibana και τις δυνατότητες αυτό που προσφέρει στον χρήστη.

### 3.3.8 Matplotlib - Seaborn

Το Matplotlib είναι μια βιβλιοθήκη για τη δημιουργία στατικών, κινούμενων και διαδραστικών γραφημάτων στην Python.<sup>4546</sup> Στα πλαίσια της διπλωματικής εργασίας η βιβλιοθήκη Matplotlib χρησιμοποιείται για την αναπαράσταση γραφημάτων στην πλατφόρμα του Django με βάση την είσοδο του χρήστη. Για να πραγματοποιηθεί με επιτυχία η αναπαράσταση των γραφημάτων στο Django απαιτείται η προσθήκη συγκεκριμένων εντολών<sup>47</sup>. Ακολουθεί ένα παράδειγμα γραφήματος με την χρήση της Matplotlib.

```

1 def stacked_area_time():
2
3     def get_file_list():
4         return os.listdir(os.path.join(MEDIA_ROOT, "files_bokeh"))
5
6     def get_columns():
7         files_list = []
8         for col in get_file_list():
9             files_list.append(col)
10    return files_list[0]
11
12 tips = pd.read_json(default_storage.open(os.path.join('files_bokeh',
13                                         get_columns())))
14
15 # set the style
16 sns.set_style("whitegrid")

```

```

17     # Color palette
18     blue, = sns.color_palette("muted", 1)
19
20     x = tips["Timestamp"]
21     y = tips["temperature"]
22     z = tips["humidity"]
23
24     # Make the plot
25     fig, ax = plt.subplots()
26     ax.plot(x, y, color=blue, lw=1)
27     ax.plot(x, z, color="red", lw=1)
28     plt.legend(['temperature', 'humidity'])
29
30     ax.fill_between(x, 0, y, alpha=.3)
31     ax.fill_between(x, 0, z, alpha=.3)
32
33     plt.xlabel('Time')
34     plt.ylabel('temperature & humidity')
35
36     # Show the graph
37     imgdata = BytesIO()
38     figure.savefig(imgdata, format='jpeg')
39     imgdata.seek(0)
40
41     data = base64.b64encode(imgdata.getvalue()).decode()
42
43     return data

```

Κώδικας 3.34: Παράδειγμα δημιουργίας γραφήματος με χρήση Matplotlib.

- (**Γραμμές 24-34**): Στο συγκεκριμένο σημείο δημιουργείται το γράφημα μέσω της Matplotlib. Το χαρακτηριστικό που ξεχωρίζει είναι η χρήση του plt object που προέρχεται από το ορισμό της βιβλιοθήκης Matplotlib μέσω της εντολής import matplotlib.pyplot as plt<sup>48</sup> ορίζοντας έτσι στην ουσία την χρήση της συγκεκριμένης βιβλιοθήκης.
- (**Γραμμές 36-41**): Για την αναπαράσταση του γραφήματος χρησιμοποιείται η βιβλιοθήκη io της Python<sup>4950</sup>. Τα δεδομένα αποθηκεύονται ως αντικείμενο τύπου Bytes και στη συνέχεια παίρνει μορφή jpeg. Τέλος, αποκωδικοποιείται η base64 κωδικοποίηση και παράγεται το τελικό αποτέλεσμα.

```

1 def jointplot_hum_temp():
2
3     #px = 1=plt.rcParams['figure.dpi'] # pixel in inches
4     #figure = plt.figure(figsize=(625*px, 450*px))
5     #mpl.rcParams['path.simplify'] = True
6
7
8     def get_file_list():
9         return os.listdir(os.path.join(MEDIA_ROOT, "files_bokeh"))
10
11    def get_columns():
12        files_list = []
13        for col in get_file_list():
14            files_list.append(col)
15        return files_list[0]
16
17
18    tips = pd.read_json(default_storage.open(os.path.join('files_bokeh',
19                                              get_columns())))
20
21    figure = sns.jointplot(data=tips, x="humidity", y="temperature", hue="ambient_noise",
22                           rasterized=True)
23    plt.suptitle("Your title here")
24
25    imgdata = io.StringIO()
26    figure.savefig(imgdata, format='svg', block=True)
27    imgdata.seek(0)
28
29    data = imgdata.getvalue()
30
31
32    return data

```

Κώδικας 3.35: Παράδειγμα δημιουργίας γραφήματος με χρήση Seaborn.

- (**Γραμμές 3-5**): Αναπαράσταση του γραφήματος με συγκεκριμένες διαστάσεις σε pixel. Η Seaborn<sup>51</sup> περνιέται στο django σαν στατική εικόνα (image) χωρίς τη δυνατότητα αλληλεπίδρασης από τον χρήστη. Συνεπώς, αν το δείγμα είναι μικρό, είναι εφικτός ο ορισμός διαστάσεων πριν την δημιουργία της. Στο συγκεκριμένο παράδειγμα μπαίνουν σε σχόλια καθώς το δείγμα είναι όλο το αρχείο 11000 στοιχείων. (βλ. [https://github.com/GvasTheGreek/Thesis-Phytobiotics-Repo/blob/master/sample\\_files/Sensor\\_Data\\_RTH1.csv](https://github.com/GvasTheGreek/Thesis-Phytobiotics-Repo/blob/master/sample_files/Sensor_Data_RTH1.csv))
- (**Γραμμές 8-16**): Οι δύο συναρτήσεις εκτελούν λειτουργίες εύρεσης του αρχείου του χρήστη. Αρχικά επιστρέφεται το μονοπάτι που αποθηκεύτηκε το αρχείο και στη συνέχεια επιλέγεται το νέο εισαχθέν αρχείο.
- (**Γραμμή 18**): Μέσω της βιβλιοθήκης Pandas<sup>52</sup> διαβάζεται το αρχείο και το περιεχόμενο του είναι προσπελάσιμο από τη Seaborn.

- (**Γραμμές 20,21**): Εδώ δημιουργείται το γράφημα που παρουσιάζεται στην πλατφόρμα. Συγκεκριμένα στην περίπτωση αυτή επιλέγονται ως άξονες η υγρασία και η θερμοκρασία με 3η μεταβλητή το όγραφο.
- (**Γραμμές 23-27**): Το παρών κομμάτι κώδικα πραγματοποιεί το ουσιαστικό πέρασμα των δεδομένων στο django. Παρατηρείται ότι το αποτέλεσμα του γραφήματος διαβάζεται ως αλφαριθμητικό (String) και αποθηκεύεται σε μορφή svgs. Παίρνοντας και επιστρέφοντας μέσω της συνάρτησης το περιεχόμενο του στην γραμμή 27, επιτυγχάνεται η σύνδεση μεταξύ Seaborn - django.

Σε γενικές γραμμές, όσον αφορά την λειτουργικότητα η Matplotlib χρησιμοποιείται για την δημιουργία βασικών γραφημάτων έναντι της Seaborn που προσφέρει περισσότερες δυνατότητες. Από την άλλη η Seaborn είναι πιο κατανοητή και εύκολη χρησιμοποιώντας μικρότερες και απλές εντολές. Επιπρόσθετα, η matplotlib είναι σε θέση να υλοποιήσει πολλαπλά γραφήματα ταυτόχρονα, σε αντίθεση με την Seaborn που μπορεί να παρουσιάσει θέματα με την μνήμη. Η επικοινωνία με τρίτες βιβλιοθήκες είναι εξίσου εύκολη και στις δύο και τέλος, χαρακτηριστική διαφορά είναι η ευελιξία της matplotlib με την πλήρη επεξεργασιμότητα της σε σχέση με τη Seaborn αλλά και ο τρόπος κλήσης τους (**import matplotlib.pyplot as plt** και **import seaborn as sns**). Τέλος παρουσιάστηκαν δύο διαφορετικοί τρόποι αναπαράστασης του γραφήματος στο Django. Ο html κώδικας διαφέρει ανάλογα με το τρόπο προσέγγισης. Στην περίπτωση που επιλεχθεί το **StringIO** module της βιβλιοθήκης IO τότε μπορεί να χρησιμοποιηθεί το template της django με χρήση των Double Curly Braces ({{ "python variable" }}). Αντιθέτως με την επιλογή του **BytesIO** module της ίδιας βιβλιοθήκης χρησιμοποιείται το κομμάτι που ακολουθεί: <img src='data:image/jpeg;base64,{{python variable}}'/>

### 3.3.9 Bokeh

Συνεχίζοντας στην επόμενη βιβλιοθήκη, η Bokeh<sup>53</sup> αποτελεί μια πιο σύνθετη και εξελιγμένη βιβλιοθήκη αναπαράστασης γραφημάτων στη Python που συμβάλει στην δημιουργία διαδραστικών γραφημάτων με εφικτή την αλληλεπίδραση από το χρήστη. Επιπλέον προσφέρει την δυνατότητα δημιουργίας οπτικοποιήσεων μέσω Javascript, χαρακτηριστικό το οποίο χρησιμοποιείται για το πέρασμα των γραφημάτων στη πλατφόρμα.

```

1 def f7d():
2     def get_file_list():
3         return os.listdir(os.path.join(MEDIA_ROOT, "files_bokeh"))
4
5     def get_columns():
6         files_list = []
7         #columns_list = []
8         for col in get_file_list():
9             files_list.append(col)
10    return files_list[0]
11

```

```

12     tips = pd.read_json(default_storage.open(os.path.join('files_bokeh',
13         get_columns()), 'r'))
14     x = tips["Timestamp"]
15
16     source = ColumnDataSource(data = dict(
17         x = tips["Timestamp"],
18         y1 = tips["temperature"],
19         y2 = tips["humidity"],
20         y3 = tips["ambient_noise"],
21         ))
22
23
24
25     # create three plots
26     # S1
27     s1 = figure(width=1750, height=350, background_fill_color="#fafafa",
28         output_backend="webgl", tools = "box_select,lasso_select,help")
29     s1.line('x', 'y1', legend_label="Temperature", color="blue", line_width
30         =2,source = source)
31     s1.circle('x', 'y1',legend_label="Temperature", size=3, color="#53777a"
32         , alpha=0.8,source=source)
33     s1.xaxis[0].formatter = DatetimeTickFormatter(days="%Y %m/%d %H:%M",
34         months="%Y %m/%d %H:%M",hours="%Y %m/%d %H:%M",minutes="%Y %m/%d %H:%M"
35         )
36
37     s1.add_tools(HoverTool(tooltips = [('@x', '@x{%-Y-%m-%d %H:%M:%S}') , ('y1',
38         '@y1{0.00}')],
39         formatters={'@x':'datetime','@y1':'numeral'},))
40
41     range_slider = DateRangeSlider(
42         title="Adjust x-axis range", # a title to display above the slider
43         start=x[0], # set the minimum value for the slider
44         end=x[len(x)-1], # set the maximum value for the slider
45         step=1, # increments for the slider
46         value=(x[0], x[len(x)-1]), # initial values for slider
47         )
48
49     range_slider.js_link("value", s1.x_range, "start", attr_selector=0)
50     range_slider.js_link("value", s1.x_range, "end", attr_selector=1)
51
52     tlayout = layout([
53         [range_slider],
54         [column(s1)],
55     ])
56
57     script, div = components(tlayout)
58     return script,div

```

Κώδικας 3.36: Μέρος κώδικα δημιουργίας γραφήματος με χρήση Bokeh.

- (**Γραμμές 2-13**): Όπως και στα προηγούμενα παραδείγματα οι συγκεκριμένες γραμμές επιστρέφουν το μονοπάτι του αρχείου του χρήστη, το επιλέγουν και χρησιμοποιώντας τη pandas βιβλιοθήκη διαβάζονται τα δεδομένα.
- (**Γραμμές 15-20**): Σημαντικό σημείο για την δημιουργία γραφημάτων με Bokeh. Τα δεδομένα αφού αναλυθούν με την Pandas πρέπει να μεταφερθούν μέσω του ColumnDataSource module της Bokeh βιβλιοθήκης. Πρόκειται για ένα μοντέλο της βιβλιοθήκης Bokeh που μορφοποιηθεί τα δεδομένα με τέτοιο τρόπο ώστε να είναι προσπελάσμα από την εντολή που αφορά το γράφημα. Η είσοδος πρέπει αυστηρά να είναι Python Dictionary ή Pandas DataFrame.
- (**Γραμμές 27-44**): Το παρών κομμάτι αποτελεί τον πυρήνα της δημιουργίας του γραφήματος. Καίριο σημείο αποτελεί η εντολή figure στην 27 με την οποία ορίζεται πρακτικά το γράφημα. Το είδος του γραφήματος ορίζεται με ξεχωριστή εντολή στις γραμμές 28,29. Επιπροσθέτως στο παράδειγμα δημιουργείται ένας ρυθμιστής εύρους (Range Slider).
- (**Γραμμές 47-53**): Στο σημείο αυτό γίνεται η τελική αναφορά και η μεταφορά στο django. Άξιο αναφοράς είναι το ότι εάν το γράφημα που υλοποιείται περιέχει σύνθετα μέρη εκτός του βασικού γραφήματος (π.χ. Slider, επιλογή ημερομηνίας, μεγέθυνση, επιλογές τιμών από τον χρήστη) συνιστάται το πέρασμα να γίνει μέσω του Bokeh μοντέλου Layout το οποίο 'συμμαζεύει' και ομορφαίνει το γράφημα χρησιμοποιώντας ενσωματωμένα χαρακτηριστικά (element attributes) όπως height, width, position. Στοιχίζοντας τα στοιχεία σαν στήλες επιτυγχάνεται το επιυψημένο αποτέλεσμα. Τέλος μέσω του μοντέλου components, το layout ενσωματώνεται (γίνεται embedded) στις δύο μεταβλητές που συμπεριφέρονται σαν html div και script.

### 3.3.10 Plotly & Dash

H Dash αποτελεί ένα Framework μέσω του οποίου δημιουργούνται διαδραστικά γραφήματα της βιβλιοθήκης Plotly χωρίς να χρειαστεί η άμεση χρήση της JavaScript και HTML. Με άλλα λόγια, η Dash είναι υπεύθυνη για την αναπαράσταση των εργαλείων που προσφέρουν την διαδραστικότητα και την αλληλεπίδραση του χρήστη και η Plotly παρέχει τα είδη των γραφημάτων. Μέσα από το παράδειγμα που ακολουθεί γίνεται ξεκάθαρη η χρήση των δύο άλλα και πως πραγματοποιείται η σύνδεση με το Django.

```

1 from dash import dcc, Input, Output, Dash, html
2 from dash.dependencies import Input, Output
3 from django_plotly_dash import DjangoDash
4 import plotly.express as px
5 import pandas as pd
6 import os
7 from django.core.files.storage import default_storage
8 from phytobiotics_v2.settings import MEDIA_ROOT
9
10 app = DjangoDash('prwto_grafima_dash')
11

```

```

12 def serve_layout():
13
14     def get_file_list():
15         return os.listdir(os.path.join(MEDIA_ROOT, "files"))
16
17     return html.Div([
18         html.H4('Interactive heat plot accompanied by a histogram that
19             is based on the options'),
20         html.P("Filter by selection:"), 
21         html.Div([
22             dcc.RangeSlider(
23                 id='range-slider',
24                 min=40, max=115, step=0.1,
25                 marks={0: '40', 115: '115'},
26                 value=[0, 115]
27             ),
28             dcc.Dropdown(
29                 get_file_list(),
30                 '',
31                 id='yaxis-column'
32             )
33         ]),
34         dcc.Graph(id="scatter-plot")
35     ])
36
37
38 app.layout = serve_layout
39
40 @app.callback(
41     Output("scatter-plot", "figure"),
42     Input("range-slider", "value"),
43     Input("yaxis-column", "value"),)
44
45 def update_bar_chart(slider_range, yaxis_column_name):
46     tips = pd.read_json(default_storage.open(os.path.join('files',
47         yaxis_column_name)))
48     low, high = slider_range
49     mask = (tips['ambient_noise'] > low) & (tips['ambient_noise'] < high)
50     fig = px.scatter(
51         tips[mask], x="temperature", y="humidity",
52         color="ambient_noise", size='ambient_noise',
53         hover_data=['ambient_noise'],
54         marginal_x="histogram", marginal_y="rug"
55     )
56
57     return fig

```

Κώδικας 3.37: Αναπαράσταση διαδραστικού γραφήματος με χρήση Dash-Plotly.

- (**Γραμμές 1-8**): Στην δήλωση των βιβλιοθηκών φαίνεται η ξεχωριστή αναφορά των dash και Plotly καθώς και τα είδη των εργαλείων τους που χρησιμοποιούνται. Στην γραμμή 3 εισάγεται το πακέτο (Package) `django_plotly_dash` μέσω του οποίου πραγματοποιείται η επικοινωνία με το Django. Παράλληλα στα settings της πλατφόρμας είναι απαραίτητη η προσθήκη της γραμμής `'django_plotly_dash.apps.DjangoPlotlyDashConfig'`. Οι εναπομέναντες βιβλιοθήκες συμβάλλουν στην σωστή λειτουργία του κώδικα.
- (**Γραμμές 10,36**): Με την χρήση του πακέτου `django_plotly_dash` ορίζεται το layout του γραφήματος με την ονομασία του στην γραμμή 10 αλλά και το παραχθέν αποτέλεσμα στην γραμμή 36.
- (**Γραμμές 12-34**): Στο σημείο αυτό πραγματοποιείται το στήσιμο του layout με χρήση των εργαλείων της dash. Η ξεκάθαρη αναφορά στα εργαλεία της html, Div δηλώνει το ρόλο της Dash ως Framework για διευκόλυνση του χρήστη στην αποφυγή χρήσης HTML & Javascript κώδικα αλλά και στην καλύτερη σύνδεση της με την Plotly. Το Dash framework μπορεί να χρησιμοποιηθεί και για την αποφυγή σύνταξης κώδικα HTML και avascript καθώς προσφέρει εναλλακτικές εντολές για χρήση HTML elements και Javascript εντολών τις οποίες μπορεί να αξιοποιήσει ο προγραμματιστής.
- (**Γραμμές 38-52**): Χρησιμοποιώντας την λειτουργία των επανακλήσεων (Callbacks) που προσφέρει η Plotly, επιτυγχάνεται η διαδραστικότητα και η ανανέωση του γραφήματος, έχοντας ως είσοδο τα features του dash και ως έξοδο το γράφημα. Το γράφημα ορίζεται από το πακέτο της Plotly στην γραμμή 46.

### 3.3.11 Kibana Dashboard

Όσον αφορά τους πίνακες (Dashboards) του kibana η ενσωμάτωση τους στη πλατφόρμα του Django πραγματοποιείται σε δύο στάδια. Αρχικά χρησιμοποιώντας το API του Kibana γίνεται εξαγωγή των Dashboard που υπάρχουν στην εφαρμογή του Kibana. Η διαδικασία αυτή επιστρέφει json μορφής δεδομένα τα οποία περιέχουν πληροφορίες για τα Dashboards. Αυτά αποθηκεύονται προσωρινά και έπειτα από επεξεργασία λαμβάνονται τα ID του κάθε Dashboard. Στην συνέχεια ακολουθεί η μεταφορά στο Django. Έχοντας το μοναδικό αναγνωριστικό (Unique ID) και με την χρήση της βιβλιοθήκης BeautifulSoup<sup>54</sup> διαμορφώνεται ξεχωριστά το κάθε Dashboard με τέτοιο τρόπο ώστε να είναι επεξεργασμένα από τον χρήστη. Με το τελικό αποτέλεσμα ο χρήστης έχει την δυνατότητα να περιηγηθεί και να ελέγξει τις δυνατότητες που προσφέρει το Dashboard με πλήρη ελευθερία στην επεξεργασία και δημιουργία γραφημάτων αλλά και στην επιλογή Dataset. Τέλος, να σημειωθεί ότι στο Django εμφανίζονται τα Dashboard που έχουν ήδη δημιουργηθεί στο Kibana και ανανεώνονται σε πραγματικό χρόνο. Ο χρήστης μπορεί να δημιουργήσει εκ νέου το δικό του Dashboard με τις δικές του ρυθμίσεις μόνο από το περιβάλλον του Kibana.

### 3.3.12 Προβλήματα που παρουσιάστηκαν

Εν κατακλείδι, κατά την διάρκεια υλοποίησης και ανασκόπησης της παρούσας διπλωματικής εργασίας παρουσιάστηκε μια σειρά εμποδίων που κάνουν αισθητή την παρουσία τους προκαλώντας μικρές δυσλειτουργίες κατά την εκτέλεση της πλατφόρμας. Στη υπό-ενότητα αυτή γίνεται αναφορά στα προβλήματα αυτά και στους τρόπους αντιμετώπισης τους με βάση τα πλαίσια στα οποία έχει υλοποιηθεί το έργο.

#### 1. Η διάρκεια ανεβάσματος αρχείου

Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο ο χρήστης έχει την δυνατότητα να ανεβάσει το αρχείο του στο Elasticsearch. Η διαδικασία αυτή αποδεικνύεται αρκετά χρονοβόρα. Στα πλαίσια της διπλωματικής εργασίας το αρχείο των 11000 γραμμών που χρησιμοποιείται ανεβαίνει σε χρονική διάρκεια μερικών λεπτών. Αυτό συμβαίνει διότι το αρχείο πρέπει να ευρετηριαστεί (indexed) και μετά να περάσει στο Elasticsearch, διαδικασία που πραγματοποιούνταν ανά γραμμή (Ευρετηρίαση μιας γραμμής και ανέβασμα). Για επίτευξη καλύτερου χρόνου πραγματοποιήθηκε μια νέα προσέγγιση με ευρετηρίαση όλου του αρχείου εξ-αρχής και στην συνέχεια ανέβασμα σαν σύνολο. Ο χρόνος εκτέλεσης μειώθηκε αλλά χωρίς ουσιαστική διαφορά για μεγάλου όγκου αρχεία. Αντιθέτως, όσο μικρότερα τα αρχεία τόσο πιο αισθητή η διαφορά στη ταχύτητα ευρετηρίασης αυτών. Στο GitHub repository του έργου (<https://github.com/GvasTheGreek/Thesis-Phytobiotics-Repo>) υπάρχουν και οι δύο προσεγγίσεις με ανάλυση των διαφορών τους. Τέλος, η βελτίωση τόσο του hardware όσο και η χρήση ενός server όπου βελτίωναν την απόδοση σε μεγάλο βαθμό.

#### 2. Η απόδοση στις βιβλιοθήκες Matplotlib και Seaborn

Οι βιβλιοθήκες Matplotlib και Seaborn, δυσκολεύονται να παράξουν το επιθυμητό γράφημα από μεγάλου όγκου δεδομένα με αποτέλεσμα να υπάρχει καθυστέρηση στην εμφάνιση όλης της σελίδας για αρκετά λεπτά. Το συμπέρασμα πάρθηκε μετά από πειράματα στα οποία έγινε μη ταυτόχρονη εμφάνιση περισσότερων από ένα γραφημάτων την φορά. Οι βιβλιοθήκες αυτές χρησιμοποιούνται συνήθως για αναπαράσταση στατικών γραφημάτων που η αλληλεπίδραση του χρήστη δεν υπάρχει και με περιορισμένα δεδομένα. Συνεπώς, για την αντιμετώπιση του θέματος πραγματοποιήθηκε η μείωση του όγκου δεδομένου που αναπαριστά το γράφημα μέσα από τυχαία λήψη. Παρόλα αυτά, το πρόβλημα εμφανίζεται σε γραφήματα πιο σύνθετα που μπορεί να περιέχουν διασταυρώσεις και συσχετίσεις δεδομένων όπως Distribution και Pair plots.

#### 3. Out Of Memory Error (Εξάντληση Μνήμης Τυχαίας Προσπέλασης)

Πρόκειται για ένα σφάλμα που εμφανίζεται κατά τη χρήση των γραφημάτων που έχουν παραχθεί στο Django. Έχοντας χρησιμοποιήσει το αρχείο των 11000 γραμμών για κάθε είδους γράφημα που υπάρχει στην πλατφόρμα, η ανανέωση της σελίδας ή η αλληλεπίδραση από τον χρήστη μπορούν να πυροδοτήσουν αυτό το σφάλμα. Το γεγονός αυτό οφείλεται στο ότι το υπολογιστικό σύστημα που τρέχει η πλατφόρμα δεν έχει εκείνη την χρονική στιγμή τους απαραίτητους πόρους για να εξυπηρετήσει το συγκεκριμένο αίτημα.

Όπως και σε προηγούμενο πρόβλημα, η μείωση του δείγματος συμβάλλει στην αποφυγή του σφάλματος, το κλείσιμο περιττών εφαρμογών για απελευθέρωση μνήμης RAM αλλά και καλύτερο υλικό hardware θα βοηθήσει στην αποφυγή επανεμφάνισης του.

#### 4. Cookie Session in Kibana

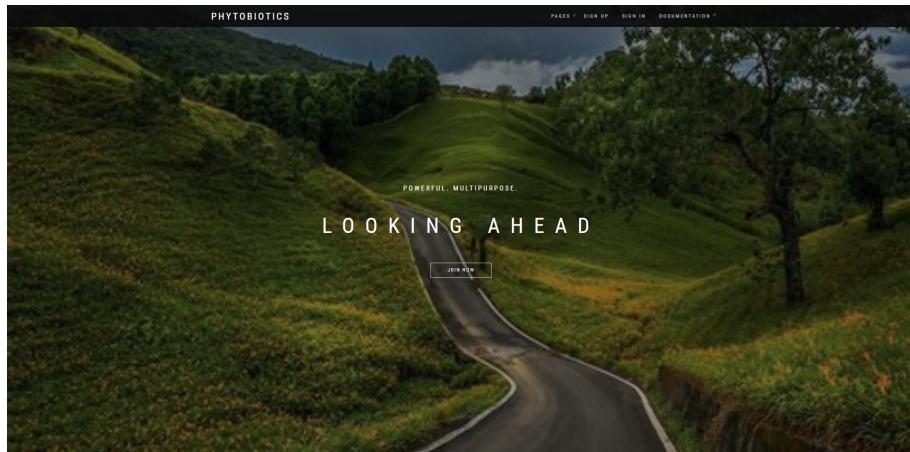
Το παρόν πρόβλημα παρουσιάζεται κατά την περιήγηση στα Dashboard του Kibana στο Django. Σε περίπτωση που ένας νέος χρήστης εισέλθει είτε για πρώτη φορά είτε μετά από καθαρισμό του ιστορικού στην σελίδα των Dashboard, το σύστημα αδυνατεί να του παράξει το αποτέλεσμα. Το Documentation του Kibana αναφέρει ότι δεν είναι εφικτή η εκμετάλλευση των πόρων χωρίς σύνδεση. Η σύνδεση στο Kibana μέσα από το dashboard πραγματοποιείται με επιτυχία αλλά λόγω του ότι γίνεται στο django αυτή η ενέργεια και όχι στη πλατφόρμα του kibana δεν εγκαθίσταται το απαραίτητο Cookie Session<sup>55</sup> για να ενεργοποιηθούν στην ουσία οι λειτουργίες του kibana. Η ιδανικότερη λύση στο πρόβλημα αυτό είναι να γίνει η σύνδεση του χρήστη στο Kibana μέσω ενός API call. Δυστυχώς, στις νεότερες εκδόσεις της εφαρμογής η λειτουργία αυτή καταργήθηκε, οπότε προσωρινά συνιστάται στο χρήστη μέσω ενός μηνύματος να πραγματοποιήσει την σύνδεση στην πλατφόρμα του Kibana και να επιστρέψει στην συνέχεια στα Dashboards του Django κάνοντας ανανέωση της σελίδας.

## Κεφάλαιο 4

# Πλοήγηση Στην Πλατφόρμα

Στο κεφάλαιο πραγματοποιείται μια πλοήγηση (Walk-through) της πλατφόρμας από την πλευρά του χρήστη. Μέσα από εικόνες, θα παρουσιάζονται οι βασικότερες λειτουργίες που προσφέρει η πλατφόρμα αλλά και πως ο χρήστης μπορεί να πλοηγηθεί σε αυτή.

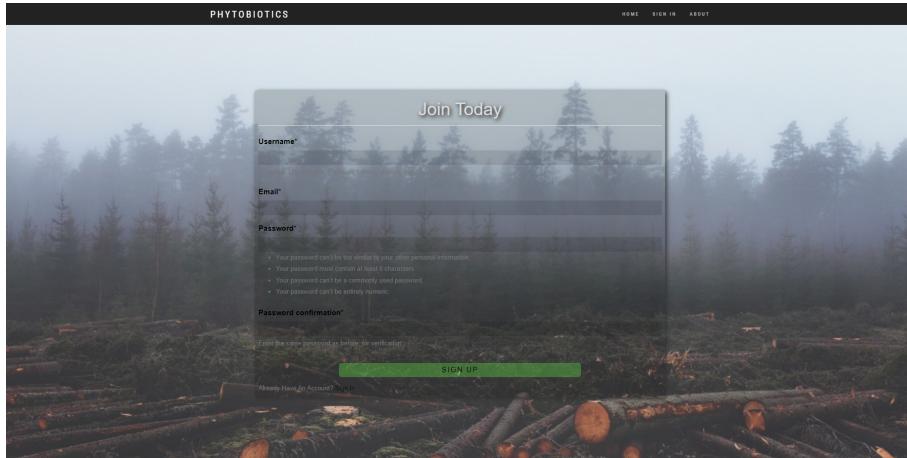
### Σελίδα Καλωσορίσματος (Welcoming Page)



Σχήμα 4.1: Η σελίδα καλωσορίσματος

Η παρούσα σελίδα είναι το πρώτο πρόγμα που αντικρίζει ο χρήστης με την είσοδο του στην πλατφόρμα. Συνιστάται από την εφαρμογή να προχωρήσει στην είσοδο ή στην εγγραφή του αλλά και να μάθει περισσότερα για την πλατφόρμα και το PhytoBiotics.

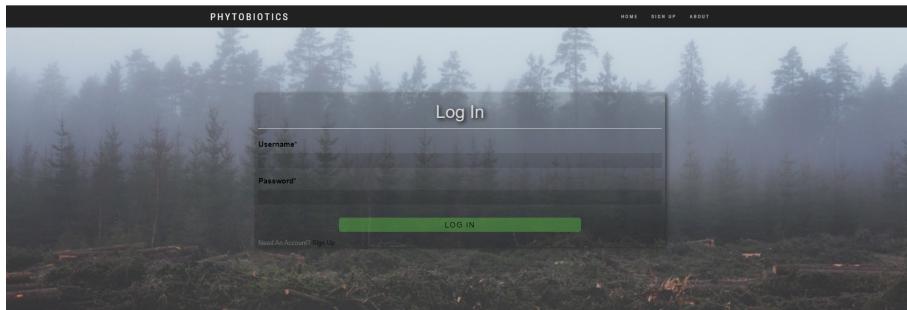
## Σελίδα Εγγραφής (Register Page)



Σχήμα 4.2: Η σελίδα εγγραφής στην πλατφόρμα

Η σελίδα εγγραφής ωθεί τον χρήστη να συμπληρώσει τα απαραίτητα στοιχεία με βάση τους κανόνες που εμφανίζονται στην φόρμα συμπλήρωσης. Εφόσον η διαδικασία χριστεί επιτυχής μπορεί στην συνέχεια να συνδεθεί. Σε περίπτωση που υπάρχει ήδη εγγραφή μπορεί απευθείας να συνδεθεί επιλέγοντας το εικονίδιο σύνδεσης (Sign-In).

## Σελίδα Σύνδεσης (Login Page)



Σχήμα 4.3: Η σελίδα σύνδεσης στην πλατφόρμα

Εφόσον ο χρήστης εγγραφεί επιτυχώς ή έχει ήδη λογαριασμό μπορεί να πραγματοποιήσει είσοδο στην εφαρμογή δίνοντας το ψευδώνυμο (Username) και τον κωδικό πρόσβασης (Password).

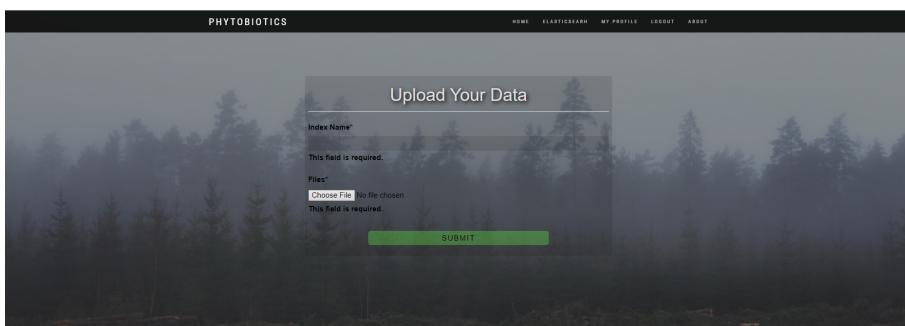
## Κεντρική Σελίδα (Home Page)



Σχήμα 4.4: Η κεντρική σελίδα

Η Αρχική σελίδα (Home Page) της εφαρμογής αποτελεί το κεντρικό χομμάτι της πλατφόρμας στο οποίο ο χρήστης έχει άμεση πρόσβαση σε όλες τις λειτουργίες της εφαρμογής. Στη μπάρα πλοήγησης (Navigation Bar) μπορεί να επιλέξει οποιαδήποτε από τα χαρακτηριστικά (features) της πλατφόρμας. Συνεχίζοντας, στην ίδια σελίδα μπορεί να διαβάσει για τις υπηρεσίες που προσφέρει η πλατφόρμα, να δει ένα μικρό δείγμα γραφικών αναπαραστάσεων και τέλος να μάθει περισσότερα για την ίδια την εφαρμογή και το (Phytobiotics) αλλά και για τον δημιουργό της.

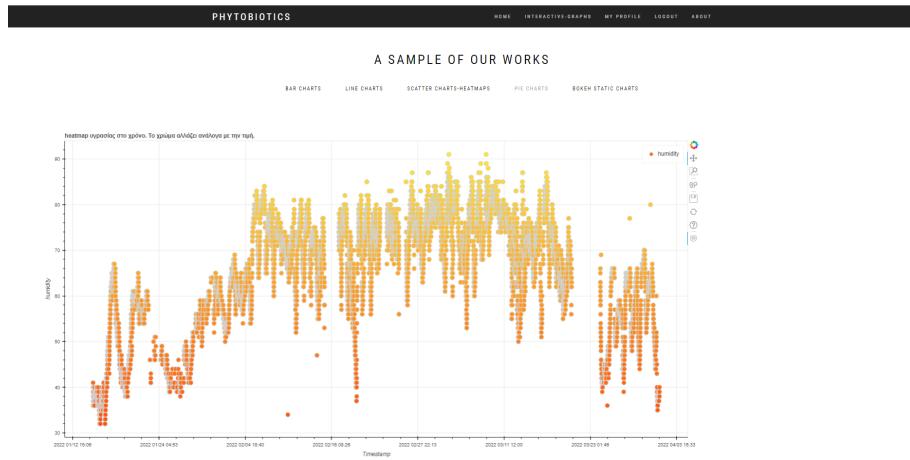
## Σελίδα Ανεβάσματος Αρχείου (Upload Page)



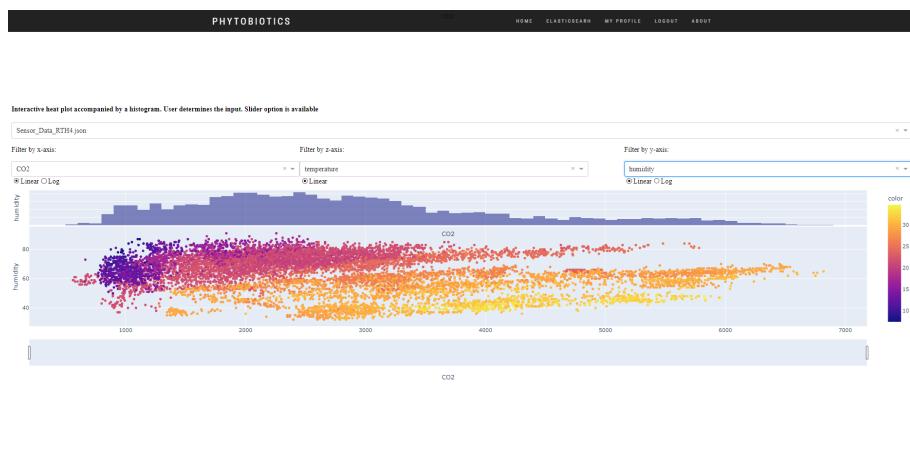
Σχήμα 4.5: Η σελίδα ανεβάσματος αρχείου

Στο σημείο αυτό ο χρήστης καλείται να ανεβάσει το αρχείο του στο Elasticsearch για την δημιουργία των γραφικών αναπαραστάσεων. Δίνοντας ως δεδομένα το όνομα του ευρετηρίου (Index) και το επιθυμητό .json αρχείο ξεκινά η διαδικασία του Upload. Αν η διαδικασία κριθεί επιτυχής εμφανίζεται στον χρήστη μια νέα επιλογή προβολής των γραφημάτων.

## Σελίδες Στατικών και Διαδραστικών Γραφημάτων



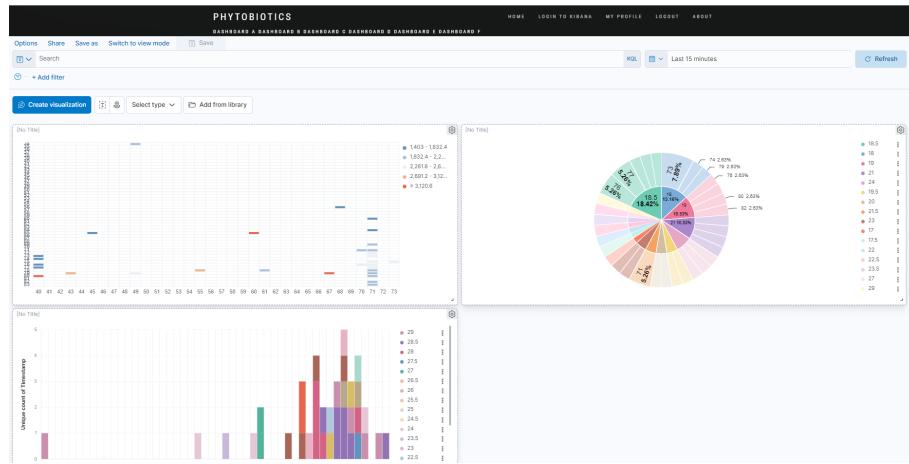
Σχήμα 4.6: Η σελίδα στατικών γραφημάτων



Σχήμα 4.7: Η σελίδα διαδραστικών γραφημάτων

Μετά την επιτυχή διαδικασία ανεβάσματος του αρχείου ο χρήστης εισέρχεται στην σελίδα στατικών και διαδραστικών γραφημάτων, στις οποίες είναι σε θέση να δει γραφήματα που έχουν δημιουργηθεί με την χρήση των βιβλιοθηκών δημιουργίας γραφημάτων (Graph Libraries) αλλά και με βάση το αρχείο που επέλεξε να ανεβάσει.

## Σελίδα Πινάκων (Dashboard Page)



Σχήμα 4.8: Η σελίδα Dashboard του Kibana

Ο χρήστης, εισερχόμενος στην σελίδα αυτή, έχει την δυνατότητα να δημιουργήσει δικές του παραλλαγές γραφημάτων με βάση τα αρχεία που έχει ανεβάσει στο elasticsearch εκμεταλλευόμενος όλες τις δυνατότητες που προσφέρει το Kibana μέσα από το ενσωματωμένο πίνακα (embedded Dashboard) στη σελίδα του Django.

# Bibliography

1. *Docker overview* Accessed 9 Sep. 2022. <https://docs.docker.com/get-started/overview/>.
2. *Docker Desktop* Accessed 9 Sep. 2022. <https://www.docker.com/products/docker-desktop/>.
3. "What is a Container? - Docker." Accessed 9 Sep. 2022. <https://www.docker.com/resources/what-container/>.
4. "Docker image reference - Docker Documentation." Accessed 9 Sep. 2022. <https://docs.docker.com/engine/reference/builder/>.
5. "Dockerfile reference - Docker Documentation." Accessed 9 Sep. 2022. <https://docs.docker.com/engine/reference/builder/>.
6. "Docker compose reference - Docker Documentation." Accessed 9 Sep. 2022. <https://docs.docker.com/compose/>.
7. "YML File Extension - What is a .yml file" Accessed 9 Sep. 2022. <https://fileinfo.com/extension/yml>.
8. "Networking overview - Docker Documentation." Accessed 9 Sep. 2022. <https://docs.docker.com/network/>.
9. "Welcome to Elastic Docs." Accessed 9 Sep. 2022. <https://www.elastic.co/docs>.
10. "What is Elasticsearch? — Elastic." Accessed 9 Sep. 2022. <https://www.elastic.co/what-is/elasticsearch>.
11. "Near real-time search — Elasticsearch Guide — Elastic." Accessed 9 Sep. 2022. <https://www.elastic.co/guide/en/elasticsearch/reference/current/near-real-time.html>.
12. Salton, G., Fox, E. A. & Wu, H. Extended boolean information retrieval. *Communications of the ACM* **26**, 1022–1036 (1983).
13. Buckley, C., Salton, G. & Allan, J. *Automatic Retrieval With Locality Information Using* in *The First Text REtrieval Conference (TREC-1)* **500** (1993), 59.
14. "Changing Unix time to readable Timestamp in Db2 - IBM." Accessed 9 Sep. 2022. <https://www.ibm.com/support/pages/changing-unix-time-readable-timestamp-db2>.

15. "What is Kibana? — Elastic." Accessed 9 Sep. 2022. <https://www.elastic.co/what-is/kibana>.
16. "The ELK Stack: From the Creators of Elasticsearch — Elastic." Accessed 9 Sep. 2022. <https://www.elastic.co/what-is/elk-stack>.
17. "Add and remove nodes in your cluster — Elasticsearch Guide." Accessed 9 Sep. 2022. <https://www.elastic.co/guide/en/elasticsearch/reference/current/add-elasticsearch-nodes.html>.
18. "Start the Elastic Stack with security enabled automatically - Elastic." Accessed 20 Oct. 2022. <https://www.elastic.co/guide/en/elasticsearch/reference/current/configuring-stack-security.html>.
19. "Django overview." Accessed 9 Sep. 2022. <https://www.djangoproject.com/overview/>.
20. "What is a Framework in Programming and Why You Should Use One." Accessed 9 Sep. 2022. <https://www.netsolutions.com/insights/what-is-a-framework-in-programming/>.
21. Halfond, W. G., Viegas, J., Orso, A., et al. *A classification of SQL-injection attacks and countermeasures* in *Proceedings of the IEEE international symposium on secure software engineering* **1** (2006), 13–15.
22. Nentwich, F., Jovanovic, N., Kirda, E., Kruegel, C. & Vigna, G. *Cross-site scripting prevention with dynamic data tainting and static analysis* in *In Proceeding of the Network and Distributed System Security Symposium (NDSS'07)* (2007).
23. "Cross Site Request Forgery (CSRF) - OWASP Foundation." Accessed 20 Oct. 2022. <https://owasp.org/www-community/attacks/csrf>.
24. Huang, L.-S., Moshchuk, A., Wang, H. J., Schechter, S. & Jackson, C. *Clickjacking: Attacks and defenses* in *21st USENIX Security Symposium (USENIX Security 12)* (2012), 413–428.
25. "JSON.org." Accessed 9 Sep. 2022. <https://www.json.org/>.
26. "CSV File Format - CSV Reader." Accessed 9 Sep. 2022. [https://www.csvreader.com/csv\\_format.php](https://www.csvreader.com/csv_format.php).
27. "File Formats: Microsoft Excel Spreadsheet (XLSX/XLS) - LeadTools." Accessed 9 Sep. 2022. <https://www.leadtools.com/help/sdk/v21/dh/to/file-formats-microsoft-excel-spreadsheet-xlsx-xls.html>.
28. "Root account - IBM." Accessed 9 Sep. 2022. [https://www.ibm.com/docs/ssw\\_aix\\_72/security/root\\_account.html](https://www.ibm.com/docs/ssw_aix_72/security/root_account.html).
29. "Upgrade Ubuntu desktop." Accessed 9 Sep. 2022. <https://ubuntu.com/tutorials/upgrading-ubuntu-desktop>.
30. "nano – Text editor." Accessed 9 Sep. 2022. <https://www.nano-editor.org/>.

31. "UsingTheTerminal - Community Help Wiki." Accessed 9 Sep. 2022. <https://help.ubuntu.com/community/UsingTheTerminal>.
32. "chown - change file owner and group - Ubuntu Manpage." Accessed 9 Sep. 2022. <https://manpages.ubuntu.com/manpages/xenial/man1/chown.1.html>.
33. "FilePermissions - Community Help Wiki." Accessed 9 Sep. 2022. <https://help.ubuntu.com/community/FilePermissions>.
34. "https - Cloudflare." Accessed 9 Sep. 2022. <https://www.cloudflare.com/learning/ssl/what-is-https/>.
35. "Search your data — Elasticsearch Guide [8.4] — Elastic." Accessed 20 Oct. 2022. <https://www.elastic.co/guide/en/elasticsearch/reference/current/search-your-data.html>.
36. "Aggregations — Elasticsearch Guide [master] — Elastic." Accessed 20 Oct. 2022. <https://www.elastic.co/guide/en/elasticsearch/reference/master/search-aggregations.html>.
37. "Command line basics - Everything curl." Accessed 9 Sep. 2022. <https://everythingcurl.dev/cmdline>.
38. "Using Python's pip to Manage Your Projects' Dependencies." Accessed 9 Sep. 2022. <https://realpython.com/what-is-pip/>.
39. "sqlite3 — DB-API 2.0 interface for SQLite databases — Python 3.10 ..." Accessed 9 Sep. 2022. <https://docs.python.org/3/library/sqlite3.html>.
40. "Using the Django authentication system." Accessed 20 Oct. 2022. <https://docs.djangoproject.com/en/4.1/topics/auth/default/>.
41. "Writing your Django app." Accessed 20 Oct. 2022. <https://docs.djangoproject.com/en/4.1/intro/tutorial07/>.
42. "Using the Django authentication system." Accessed 20 Oct. 2022. <https://docs.djangoproject.com/en/4.1/topics/auth/default/>.
43. "Forms have never been this crispy — django-crispy ... - Read the Docs." Accessed 20 Oct. 2022. <https://django-crispy-forms.readthedocs.io/>.
44. "5. Data Structures — Python 3.11.0 documentation." Accessed 20 Oct. 2022. <https://docs.python.org/3/tutorial/datastructures.html>.
45. "matplotlib: plotting with Python - GitHub." Accessed 9 Sep. 2022. <https://github.com/matplotlib/matplotlib>.
46. "Matplotlib 3.6.0 documentation." Accessed 9 Sep. 2022. <https://matplotlib.org/stable/index.html>.
47. "Embed matplotlib graph in Django webpage" Accessed 20 Oct. 2022. <https://stackoverflow.com/a/62123328>.

48. "5. The import system — Python 3.11.0 documentation." Accessed 1 Nov. 2022. <https://docs.python.org/3/reference/import.html>.
49. "io — Core tools for working with streams — Python ...." Accessed 20 Oct. 2022. <https://docs.python.org/3/library/io.html>.
50. "cpython/io.rst at main · GitHub." Accessed 20 Oct. 2022. <https://github.com/python/cpython/blob/master/Doc/library/io.rst>.
51. "mwaskom/seaborn: Statistical data visualization in Python · GitHub." Accessed 20 Oct. 2022. <https://github.com/mwaskom/seaborn>.
52. "pandas: powerful Python data analysis toolkit · GitHub." Accessed 20 Oct. 2022. <https://github.com/pandas-dev/pandas>.
53. "bokeh: Interactive Data Visualization in the browser ... · GitHub." Accessed 20 Oct. 2022. <https://github.com/bokeh/bokeh>.
54. "Beautiful Soup Documentation — Beautiful Soup ... - Read the Docs." Accessed 20 Oct. 2022. <https://beautiful-soup-4.readthedocs.io/en/latest/>.
55. "Window.sessionStorage - Web APIs - MDN Web Docs." Accessed 1 Nov. 2022. <https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage>.

