ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Διπλωματική Εργασία

Έξυπνο Σύστημα Παρακολούθησης του Περιβάλλοντος με χρήση Τεχνολογιών Μικροϋπηρεσιών και Εικονικοποίησης βασισμένης σε Περιέκτες

Συγγραφέας: Βασιλάκος Θεμιστοκλής
Αριθμός Μητρώου: 19390031

Επιβλέπων Καθηγητής: Βασίλειος Μάμαλης
Συνεπιβλέπων: Απόστολος Αναγνωστόπουλος

Αθήνα, Μάρτιος 2024

Smart Environmental Monitoring System using Microservices and Container-based Virtualization

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	Ονοματεπώνυμο	Βαθμίδα/Ιδιότητα	Ψηφιακή Υπογραφή
1	Μάμαλης Βασίλειος	Καθηγητής Πανεπιστημίου Δυτικής Αττικής	Vasileios Mamalis Digitally signed by Vasileios Mamalis Date: 2024.03.23 03:09:08 +02'00'
2	Ιωάννα Καντζάβελου	Επίκουρη Καθηγήτρια Πανεπιστημίου Δυτικής Αττικής	Ioanna Kantzavelou Digitally signed by Ioanna Kantzavelou Date: 2024.03.24 15:10:45 +02'00'
3	Παναγιώτης Καρκαζής	Αναπληρωτής Καθηγητής Πανεπιστημίου Δυτικής Αττικής	Panagiotis Karkazis Digitally signed by Panagiotis Karkazis Date: 2024.03.26 11:44:17 +02'00'

Ημερομηνία εξέτασης 11/03/2024

Smart Environmental Monitoring System using Microservices and Container-based Virtualization

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος **Βασιλάκος Θεμιστοκλής** του **Ιωάννη**, με αριθμό μητρώου **19390031** φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

(υπογραφή)

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία αναλύει ένα εξαιρετικά ενδιαφέρον και δύσκολο γνωστικό αντικείμενο για το οποίο απαιτήθηκε αρκετός χρόνος και προσπάθεια τόσο για την έρευνα όσο και για την υλοποίησή της. Γι'αυτό λοιπόν θα ήθελα να ευχαριστήσω πρώτα τον επιβλέπων καθηγητή μου, κ. Βασίλειο Μάμαλη και τον συν-επιβλέπων καθηγητή μου, κ. Απόστολο Αναγνωστόπουλο, οι οποίοι με καθοδήγησαν και μου έδωσαν τις απαραίτητες συμβουλές για την ορθή και ολοκληρωμένη ανάπτυξη της εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου που ήταν υποστηρικτικοί και ενθαρρυντικοί καθόλη την διάρκεια των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Αντικείμενο της διπλωματικής εργασίας αποτέλεσε η μελέτη, ο σχεδιασμός και η ανάπτυξη ενός έξυπνου συστήματος παρακολούθησης του περιβάλλοντος (environmental monitoring) με χρήση τεχνικών εικονικοποίησης βασισμένης σε περιέκτες (container-based virtualization) και αρχιτεκτονικής μικροϋπηρεσιών (microservices architecture).

Αρχικά, μελετήθηκαν οι απαιτήσεις ενός συστήματος παρακολούθησης του περιβάλλοντος ως προς τη συλλογή των απαιτούμενων δεδομένων (για την ποιότητα του αέρα, του νερού, του εδάφους κ.α. μετρήσεων) με χρήση ειδικού εξοπλισμού/συσκευών (sensors κλπ).

Στη συνέχεια αναλύθηκαν σε θεωρητικό επίπεδο σύγχρονες τεχνολογίες και εργαλεία που χρησιμοποιούνται τόσο στην ανάπτυξη λογισμικού γενικότερα, όσο και στα πλαίσια της εργασίας αυτής.

Τέλος, υλοποιήθηκε μία εφαρμογή συγκέντρωσης και επεξεργασίας των δεδομένων αυτών (με σκοπό την παροχή υπηρεσιών πληροφόρησης και τη δρομολόγηση απαιτούμενων ενεργειών, σε άμεσο, βραχυπρόθεσμο και μακροπρόθεσμο επίπεδο), η οποία βασίστηκε στα εργαλεία Docker και Kubernetes (με απώτερο στόχο την αποδοτική υποστήριξη και ενορχήστρωση των παρεχόμενων υπηρεσιών σε κατανεμημένο περιβάλλον.

Abstract

The subject of this thesis was the study, design and development of an intelligent environmental monitoring system using container-based virtualization techniques and microservices architecture.

Initially, the requirements of an environmental monitoring system were studied in terms of collecting the required data (air quality, water quality, soil quality, and other measurements) using specific equipment/devices (sensors, etc.).

Subsequently, modern technologies and tools used both in software development in general and in the context of this work were analyzed on a theoretical level.

Finally, an application for the collection and processing of this data (in order to provide information services and the routing of required actions, in the immediate, short and long term) was implemented, based on Docker and Kubernetes tools (with the ultimate goal of efficient support and orchestration of the services provided in a distributed environment.

Ενότητες

1. Εισαγωγή

Καθώς οι επιπτώσεις της ανθρώπινης δραστηριότητας στο περιβάλλον είναι πλέον παγκόσμιες και εκτείνονται από τη ρύπανση του αέρα και των υδάτων μέχρι την υπερθέρμανση του πλανήτη, η ανάγκη για προηγμένες τεχνολογίες παρακολούθησης και παρέμβασης είναι πιο επείγουσα από ποτέ. Γι΄αυτό λοιπόν η προστασία του περιβάλλοντος και η παρακολούθηση της ποιότητάς του έχουν γίνει κρίσιμες ανησυχίες στην εποχή μας.

Η ραγδαία ανάπτυξη της τεχνολογίας προσφέρει νέες δυνατότητες για την παρακολούθηση και τη διαχείριση του περιβάλλοντος. Η χρήση προηγμένων συστημάτων αισθητήρων, τηλεπισκόπησης, και πληροφορικής μπορεί να παρέχει στους ερευνητές και το κοινό πληροφορίες σε πραγματικό χρόνο, που επιτρέπουν την άμεση αντίδραση και την υιοθέτηση αποτελεσματικών πολιτικών προστασίας του περιβάλλοντος.

Στο πλαίσιο αυτό, αυτή η διατριβή επικεντρώνεται στη σχεδίαση, την ανάπτυξη και την εφαρμογή ενός έξυπνου συστήματος παρακολούθησης του περιβάλλοντος, χρησιμοποιώντας τεχνικές εικονικοποίησης βασισμένες σε περιέκτες και αρχιτεκτονική μικροϋπηρεσιών.

1.1 Στόχος Διατριβής

Αυτή η διατριβή είναι μια προσπάθεια να συμβάλλει στην ανάπτυξη καινοτόμων λύσεων για την παρακολούθηση του περιβάλλοντος, χρησιμοποιώντας σύγχρονες τεχνολογίες και μεθόδους. Αναμένεται να προσφέρει σημαντικές εφαρμογές στον τομέα της περιβαλλοντικής επιστήμης και της πολιτικής προστασίας του περιβάλλοντος.

1.2 Υπόβαθρο Γνώσεων

Διαβάζοντας το 1ο Κεφάλαιο της εισαγωγής ο αναγνώστης θα είναι σε θέση να καταλάβει το κίνητρο για την εκπόνηση της διπλωματικής εργασίας. Έπειτα, στο 3ο και στο κεφάλαιο 4ο θα λάβει το θεωρητικό υπόβαθρο που χρειάζεται για την κατανόηση των τεχνικών και τεχνολογιών που θα αποτελέσουν το θεμέλιο λίθο του συστήματος παρακολούθησης του περιβάλλοντος.

1.3 Δομή Εργασίας

Αρχικά, θα αναλυθούν τα υπάρχον συστήματα περιβαλλοντικής παρακολούθησης και τα δεδομένα που μετράνε ανά κατηγορία ρύπανσης. Έπειτα, θα γίνει αναφορά σε τεχνικές ανάπτυξης σύγχρονου λογισμικού και σε τεχνολογίες που βοηθούν στην υλοποίησή τους. Τέλος, θα αναπτυχθούν οι λειτουργικές απαιτήσεις ενός τέτοιου συστήματος, θα παρουσιαστεί η αρχιτεκτονική και η υλοποίηση του συστήματος, καθώς και η διεπαφή του από τη πλευρά ενός χρήστη.

2. Συστήματα Περιβαλλοντικής Παρακολούθησης

Τα τρέχοντα συστήματα περιβαλλοντικής παρακολούθησης περιλαμβάνουν ένα ευρύ φάσμα τεχνολογιών και μεθοδολογιών που είναι προσαρμοσμένα για την αξιολόγηση και τη διαχείριση περιβαλλοντικών παραμέτρων σε διάφορους τομείς. Αυτά τα συστήματα χρησιμοποιούν συνήθως έναν συνδυασμό αισθητήρων, πλατφορμών συλλογής δεδομένων και αναλυτικών εργαλείων για τη συλλογή, επεξεργασία και ερμηνεία περιβαλλοντικών δεδομένων. Ορισμένα υπάρχων συστήματα τέτοιου είδους είναι τα εξής:

- **AirNow:** Το AirNow, που αναπτύχθηκε από την Υπηρεσία Προστασίας του Περιβάλλοντος των ΗΠΑ, παρέχει δεδομένα και προβλέψεις για την ποιότητα του αέρα σε πραγματικό χρόνο για πόλεις και περιοχές σε όλες τις Ηνωμένες Πολιτείες.

- **Παγκόσμιο Σύστημα Συστημάτων Παρατήρησης της Γης (GEOSS):** Το GEOSS είναι μια συνεργατική πρωτοβουλία υπό την ηγεσία της Ομάδας για τις Παρατηρήσεις της Γης για την ενσωμάτωση δεδομένων περιβαλλοντικής παρακολούθησης από διάφορες πηγές παγκοσμίως. Στόχος της είναι η παροχή ανοικτής πρόσβασης σε περιβαλλοντικά δεδομένα, εργαλεία και υπηρεσίες για τη λήψη αποφάσεων και τη βιώσιμη ανάπτυξη.

- **Ευρωπαϊκός Οργανισμός Περιβάλλοντος (ΕΟΠ):** Ο ΕΟΠ παρέχει πληροφορίες σχετικά με την ποιότητα του αέρα για τις ευρωπαϊκές χώρες μέσω της πλατφόρμας του Δείκτη Ποιότητας Αέρα (AQI). Συγκεντρώνει δεδομένα από εθνικά δίκτυα παρακολούθησης για την αξιολόγηση των επιπέδων ποιότητας του αέρα και την παροχή συστάσεων που σχετίζονται με την υγεία του κοινού.

- **Πρωτοβουλία Ωκεάνιων Παρατηρητηρίων (OOI):** Η OOI είναι ένα δίκτυο πλατφόρμων και αισθητήρων παρατήρησης των ωκεανών που αναπτύσσονται σε διάφορες ωκεάνιες περιοχές για την παρακολούθηση φυσικών, χημικών, βιολογικών και γεωλογικών παραμέτρων. Παρέχει δεδομένα σε πραγματικό χρόνο και μακροπρόθεσμα δεδομένα για τη μελέτη της δυναμικής των ωκεανών, των θαλάσσιων οικοσυστημάτων και των επιπτώσεων της κλιματικής αλλαγής.

2.1 Σημασία της Περιβαλλοντικής Παρακολούθησης

¨Οπως αναφέρθηκε και στο κεφάλαιο της εισαγωγής, η σημασία της περιβαλλοντικής παρακολούθησης είναι κρίσιμη για την κατανόηση της κατάστασης του φυσικού περιβάλλοντος και των επιπτώσεών της ανθρώπινης δραστηριότητας. Μέσω της παρακολούθησης των περιβαλλοντικών παραμέτρων, μπορούμε να αξιολογήσουμε την υγεία του οικοσυστήματος και να αντιληφθούμε τις τάσεις που απαιτούν δράση. Η περιβαλλοντική παρακολούθηση παρέχει επίσης τη βάση για την ανάπτυξη περιβαλλοντικών πολιτικών, τη λήψη αποφάσεων και την αξιολόγηση της απόδοσης των μέτρων που λαμβάνονται. Συνολικά, η περιβαλλοντική παρακολούθηση είναι ουσιώδης για τη διατήρηση της βιώσιμης ανάπτυξης και την προστασία του περιβάλλοντος για τις επόμενες γενιές [1].

2.2 Τύποι Περιβαλλοντικών Δεδομένων

Στα πλαίσια αυτής της διπλωματικής εργασίας, έχουν επιλεχθεί τριών ειδών κατηγορίες μετρήσεων. Πιο συγκεκριμένα, θα παρακολουθείτε η ποιότητα του αέρα, του νερού και του εδάφους. Τα εκάστοτε είδη αποτελούνται από τα δικά τους μοναδικά χαρακτηριστικά, τα οποία θα αναλυθούν παρακάτω.

2.2.1 Δεδομένα παρακολούθησης αέρα

Σχετικά με την ποιότητα του αέρα, θα μετριούνται τα εξής δεδομένα [2]:

- **Temperature:** Μέτρο της θερμοκρασίας ή του ψύχους του αέρα. Συνήθως μετριέται σε βαθμούς Κελσίου (°C) ή Φαρενάιτ (°F).

- **Humidity:** Η υγρασία αναφέρεται στην ποσότητα των υδρατμών που υπάρχουν στον αέρα. Εκφράζεται συνήθως ως ποσοστό. Η υψηλή υγρασία υποδηλώνει περισσότερη υγρασία στον αέρα, ενώ η χαμηλή υγρασία υποδηλώνει ξηρότερο αέρα.

- **CO2:** Διοξείδιο του άνθρακα είναι ένα άχρωμο και άοσμο αέριο που υπάρχει φυσικά στην ατμόσφαιρα της Γης. Παράγεται από την αναπνοή των ζωντανών οργανισμών και από την καύση ορυκτών καυσίμων. Η παρακολούθηση των επιπέδων CO2 είναι σημαντική σε εσωτερικούς χώρους για τη διασφάλιση καλής ποιότητας αέρα.

- **VOCs:** Οι πτητικές οργανικές ενώσεις είναι μια ομάδα οργανικών χημικών ουσιών που μπορούν εύκολα να εξατμιστούν στον αέρα. Εκπέμπονται από διάφορες πηγές, συμπεριλαμβανομένων των οικιακών προϊόντων, των χρωμάτων, των ειδών καθαρισμού και των οικοδομικών υλικών. Ορισμένες πτητικές οργανικές ενώσεις μπορούν να συμβάλλουν στη ρύπανση του αέρα και να έχουν πιθανές επιπτώσεις στην υγεία.

- **PM2.5:** Τα PM2.5 αναφέρονται σε μικροσωματίδια με διάμετρο 2,5 μικρομέτρων ή μικρότερη. Τα σωματίδια αυτά μπορεί να προέρχονται από διάφορες πηγές, όπως τα καυσαέρια των οχημάτων, οι βιομηχανικές εκπομπές και η καύση ορυκτών καυσίμων. Τα PM2.5 προκαλούν ανησυχία επειδή μπορούν να διεισδύσουν βαθιά στο αναπνευστικό σύστημα και να έχουν δυσμενείς επιπτώσεις στην υγεία.

- **CO:** Το μονοξείδιο του άνθρακα είναι ένα άχρωμο και άοσμο αέριο που σχηματίζεται όταν τα καύσιμα που περιέχουν άνθρακα (όπως η βενζίνη, το ξύλο ή το φυσικό αέριο) καίγονται ατελώς. Τα υψηλά επίπεδα μονοξειδίου του άνθρακα μπορεί να είναι επικίνδυνα, καθώς μπορεί να επηρεάσουν την ικανότητα του σώματος να μεταφέρει οξυγόνο.

2.2.2 Δεδομένα παρακολούθησης νερού

Σχετικά με την ποιότητα του νερού, θα μετριούνται τα εξής δεδομένα [3]:

- **Dissolved oxygen:** Το διαλυμένο οξυγόνο αναφέρεται στην ποσότητα του αερίου οξυγόνου που είναι διαλυμένο στο νερό. Είναι ζωτικής σημασίας για την επιβίωση των υδρόβιων οργανισμών. Τα επαρκή επίπεδα διαλυμένου οξυγόνου υποστηρίζουν τα ψάρια και άλλους υδρόβιους οργανισμούς, ενώ τα χαμηλά επίπεδα μπορούν να οδηγήσουν σε υποξία, βλάπτοντας τα υδάτινα οικοσυστήματα.

- **Oxidation-reduction potential:** Το Δυναμικό οξείδωσης-αναγωγής είναι ένα μέτρο της ικανότητας ενός διαλύματος να οξειδώνει ή να ανάγει ουσίες. Στο πλαίσιο της ποιότητας του νερού, παρέχει πληροφορίες σχετικά με τη συνολική χημική δραστηριότητα και τη δυνατότητα του νερού να κερδίζει ή να χάνει ηλεκτρόνια. Το

ORP χρησιμοποιείται συχνά για την αξιολόγηση της αποτελεσματικότητας των διεργασιών επεξεργασίας νερού.

- **PH:** Το PH είναι ένα μέτρο της οξύτητας ή της αλκαλικότητας ενός διαλύματος. Βασίζεται σε μια κλίμακα από το 0 έως το 14, όπου ένα pH 7 είναι ουδέτερο, τιμές κάτω του 7 είναι όξινο και τιμές άνω του 7 είναι αλκαλικό. Το pH του νερού είναι μια σημαντική παράμετρος, καθώς μπορεί να επηρεάσει τις χημικές αντιδράσεις, τη διαθεσιμότητα θρεπτικών συστατικών και την υγεία των υδρόβιων οργανισμών.

- **Turbidity:** Η θολερότητα είναι ένα μέτρο της θολερότητας ή της θολότητας ενός υγρού που προκαλείται από μεγάλο αριθμό μεμονωμένων σωματιδίων. Στο νερό, η θολότητα προκαλείται συχνά από αιωρούμενα στερεά, όπως ιλύ, άργιλος και οργανική ύλη. Η υψηλή θολότητα μπορεί να επηρεάσει τη διείσδυση του φωτός, γεγονός που μπορεί να επηρεάσει την ανάπτυξη των υδρόβιων φυτών και τη συνολική υγεία των υδάτινων οικοσυστημάτων.

- **Total dissolved solids:** Τα Ολικά διαλυμένα στερεά είναι ένα μέτρο της συνολικής συγκέντρωσης των διαλυμένων ουσιών στο νερό, συμπεριλαμβανομένων των ορυκτών, των αλάτων και της οργανικής ύλης. Συνήθως εκφράζεται σε μέρη ανά εκατομμύριο (ppm) ή χιλιοστόγραμμα ανά λίτρο (mg/L). Τα υψηλά επίπεδα TDS μπορούν να επηρεάσουν τη γεύση του νερού και μπορεί να υποδηλώνουν μόλυνση.

- **Temperature:** Η θερμοκρασία του νερού είναι μια θεμελιώδης παράμετρος που μπορεί να επηρεάσει διάφορες φυσικές και βιολογικές διεργασίες στα υδάτινα οικοσυστήματα. Επηρεάζει τη διαλυτότητα των αερίων, τους μεταβολικούς ρυθμούς των υδρόβιων οργανισμών και τη συνολική υγεία των υδάτινων οικοσυστημάτων. Η θερμοκρασία παρακολουθείται συχνά για την αξιολόγηση της καταλληλότητας του νερού για διάφορες χρήσεις και για την ανίχνευση τυχόν μη φυσιολογικών μεταβολών.

2.2.3 Δεδομένα παρακολούθησης εδάφους

Σχετικά με την ποιότητα του εδάφους, θα μετριούνται τα εξής δεδομένα [4]:

- **Soil Temperature:** Η θερμοκρασία του εδάφους αναφέρεται στη θερμοκρασία σε ορισμένο βάθος κάτω από την επιφάνεια. Επηρεάζει την ανάπτυξη των φυτών, τη μικροβιακή δραστηριότητα και τη διαθεσιμότητα των θρεπτικών στοιχείων. Η παρακολούθηση της βοηθά στην κατανόηση των εποχικών διακυμάνσεων και στη βελτιστοποίηση των γεωργικών πρακτικών.

- **Soil Moisture:** Η εδαφική υγρασία είναι ένα μέτρο της ποσότητας του νερού που υπάρχει στο έδαφος. Συχνά εκφράζεται σε μονάδες όπως τα centibars (cbar), τα οποία υποδηλώνουν την τάση που απαιτείται για την εξαγωγή νερού από το έδαφος. Η παρακολούθηση της εδαφικής υγρασίας είναι ζωτικής σημασίας για την αποτελεσματική διαχείριση της άρδευσης και την κατανόηση της διαθεσιμότητας εδαφικού νερού για τα φυτά.

- **Electronic Conductivity:** Η ηλεκτρονική αγωγιμότητα, η οποία συχνά μετράται σε χιλιοστά του χιλιοστού ανά μέτρο (mS/m), είναι ένας δείκτης της ικανότητας του εδάφους να άγει ηλεκτρικό ρεύμα. Σχετίζεται με τη συγκέντρωση των διαλυμένων αλάτων στο έδαφος. Η υψηλή αγωγιμότητα μπορεί να υποδηλώνει αυξημένα επίπεδα αλατότητας, επηρεάζοντας την ανάπτυξη των φυτών.

- **Volumetric Water Content:** Η ογκομετρική περιεκτικότητα σε νερό είναι το ποσοστό του όγκου ενός εδάφους που καταλαμβάνεται από νερό. Συχνά μετράται σε deciSiemens ανά μέτρο (dS/m) και χρησιμοποιείται για την αξιολόγηση της ικανότητας του εδάφους να άγει ηλεκτρικό ρεύμα, παρέχοντας πληροφορίες για την αλατότητα του εδάφους.

- **Salinity:** Αλατότητα είναι η συγκέντρωση διαλυμένων αλάτων στο έδαφος ή στο νερό. Η υπερβολική αλατότητα μπορεί να είναι επιζήμια για την ανάπτυξη των φυτών, καθώς επηρεάζει την πρόσληψη νερού από τις ρίζες των φυτών. Συνήθως μετριέται σε μέρη ανά χίλια (ppt) ή μονάδες ηλεκτρικής αγωγιμότητας (EC).

- **Total Suspended Solids (TSS):** Ενώ τα TSS συνδέονται συνήθως με την ποιότητα των υδάτων, μπορεί επίσης να είναι σημαντικά για την αξιολόγηση της ποιότητας του εδάφους, ιδίως σε περιοχές που είναι επιρρεπείς στη διάβρωση. Τα υψηλά επίπεδα αιωρούμενων στερεών στο έδαφος μπορεί να υποδηλώνουν προβλήματα διάβρωσης, όπου τα σωματίδια του εδάφους παρασύρονται από την απορροή του νερού ή τον άνεμο. Η παρακολούθηση των TSS στο έδαφος μπορεί να βοηθήσει στην κατανόηση των ρυθμών διάβρωσης, της απώλειας εδάφους και της αποτελεσματικότητας των μέτρων ελέγχου της διάβρωσης.

- **Flow/Volume:** Στο πλαίσιο του εδάφους, οι μετρήσεις ροής και όγκου είναι λιγότερο σημαντικές σε σύγκριση με τα συστήματα νερού. Ωστόσο, η κατανόηση της κίνησης του νερού μέσω του εδάφους, όπως οι ρυθμοί διήθησης και τα χαρακτηριστικά αποστράγγισης, είναι ζωτικής σημασίας για τη διαχείριση του εδάφους, τον σχεδιασμό της άρδευσης και την εκτίμηση του κινδύνου πλημμύρας.

- **Nitrate:** Τα νιτρικά είναι μια μορφή αζώτου που είναι απαραίτητη για την ανάπτυξη των φυτών. Ωστόσο, τα υπερβολικά επίπεδα νιτρικών στο έδαφος μπορεί να οδηγήσουν σε περιβαλλοντικά ζητήματα όπως η ρύπανση των υδάτων και ο ευτροφισμός. Τα επίπεδα νιτρικών μετριούνται συχνά σε χιλιοστόγραμμα ανά λίτρο (mg/L) ή μέρη ανά εκατομμύριο (ppm).

2.3 Συγκέντρωση δεδομένων

Τα παραπάνω δεδομένα συγκεντρώνονται από σένσορες που είναι τοποθετημένοι σε μία πληθώρα γεωγραφικών περιοχών, ώστε να μπορέσουν να καλύψουν όσο περισσότερο φάσμα μπορούν και μιλούν απευθείας με κάποιο δίκτυο. Πιο συγκεκριμένα, ανήκουν στην κατηγορία του Διαδικτύου των Αντικειμένων (IoT) [5], το οποίο θα αναλυθεί σε επόμενο κεφάλαιο.

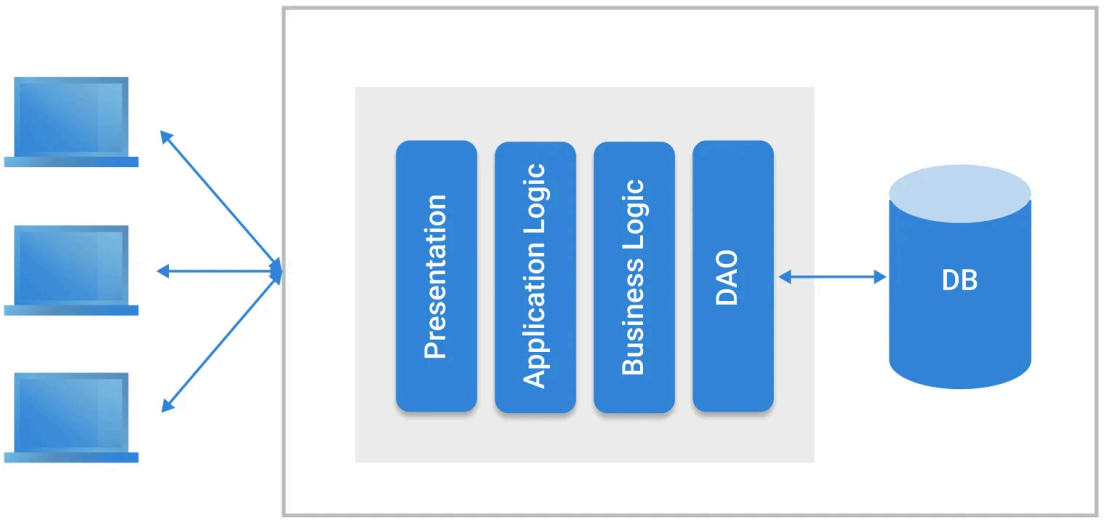
3. Ανάπτυξη Λογισμικού και Τεχνολογικές Προσεγγίσεις

Αυτό το κεφάλαιο εξετάζει διάφορες τεχνολογικές προσεγγίσεις. Πρώτα, θα διερευνήσει τις αρχιτεκτονικές μονόλιθων και μικροϋπηρεσιών, καθώς και τη σημασία του Διαδικτύου των Αντικειμένων (IoT), της εικονοποίησης (virtualization) και της ενορχήστρωσης περιεκτών (container orchestration) .Με αυτόν τον τρόπο, το κεφάλαιο θα παρέχει μια συνολική εικόνα των προσεγγίσεων που χρησιμοποιούνται για την σύγχρονη ανάπτυξη λογισμικού.

3.1 Μονολιθική αρχιτεκτονική

Η μονολιθική αρχιτεκτονική [6] αναφέρεται σε μια παραδοσιακή προσέγγιση ανάπτυξης λογισμικού όπου μια εφαρμογή κατασκευάζεται ως μια ενιαία, αδιαίρετη μονάδα. Σε αυτή την αρχιτεκτονική, όλα τα στοιχεία της εφαρμογής, συμπεριλαμβανομένης της διεπαφής χρήστη, της επιχειρησιακής λογικής και των επιπέδων πρόσβασης σε δεδομένα, ενσωματώνονται στενά σε μια ενιαία βάση κώδικα. Η εφαρμογή συνήθως αναπτύσσεται ως ένα ενιαίο εκτελέσιμο αρχείο ή ως ένα σύνολο στενά συνδεδεμένων ενοτήτων που εκτελούνται εντός της ίδιας διεργασίας. Έρχεται σε αντίθεση με τα σύγχρονα αρχιτεκτονικά στυλ, όπως οι μικροϋπηρεσίες, οι οποίες αποσυνθέτουν μια εφαρμογή σε μικρότερες, χαλαρά συνδεδεμένες υπηρεσίες.

[Εικόνα 1]: [Απεικόνιση Διαγράμματος Μονολιθικής Εφαρμογής]



3.1.1 Πλεονεκτήματα και μειονεκτήματα μονολιθικής αρχιτεκτονικής

Όσον αφορά τα πλεονεκτήματα της μονολιθικής αρχιτεκτονικής έχουμε τα εξής:

- **Απλότητα:** Η μονολιθική αρχιτεκτονική απλοποιεί την διαδικασία της ανάπτυξης, καθώς ολόκληρη η εφαρμογή περιέχεται σε μια ενιαία βάση κώδικα. Με αυτόν τον τρόπο μπορεί να μειωθεί ο χρόνος ανάπτυξης και γενικά τα έξοδα, ειδικά για μικρά και μεσαία έργα.

- **Ευκολία ανάπτυξης:** Με όλα τα συστατικά στενά ενσωματωμένα, οι προγραμματιστές μπορούν εύκολα να περιηγηθούν και να κατανοήσουν τη δομή και τις εξαρτήσεις της εφαρμογής, διευκολύνοντας την υλοποίηση νέων χαρακτηριστικών, τη διόρθωση σφαλμάτων και τη συντήρηση της βάσης κώδικα.

Αντιθέτως, σχετικά με τα μειονεκτήματα, έχουμε αντίστοιχα:

- **Προκλήσεις επεκτασιμότητας:** Οι μονολιθικές αρχιτεκτονικές μπορούν να δημιουργήσουν προκλήσεις όσον αφορά την επεκτασιμότητα, καθώς η εφαρμογή αυξάνεται σε μέγεθος και πολυπλοκότητα, η κλιμάκωση γίνεται πιο δύσκολη, αφού όλα τα στοιχεία πρέπει να κλιμακώνονται μαζί. Αυτό μπορεί να οδηγήσει σε αναποτελεσματικότητα και υπερπρομήθεια πόρων.

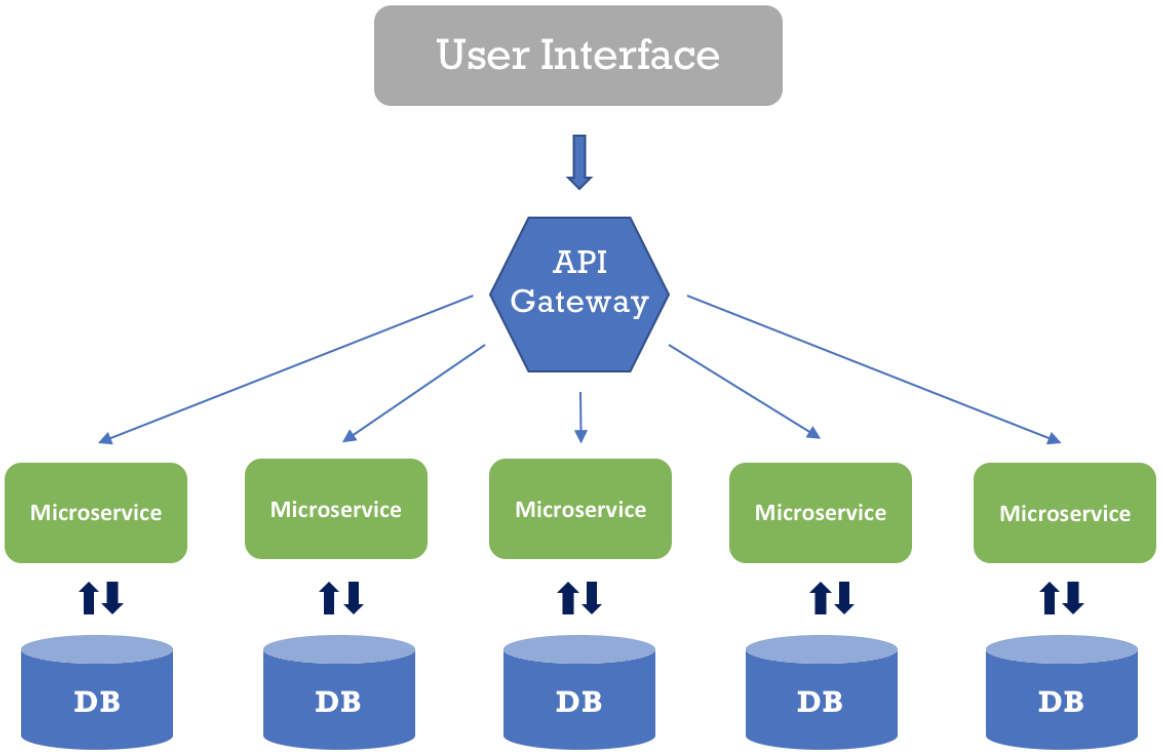
- **Περιορισμένη τεχνολογική ευελιξία:** Δεδομένου ότι όλα τα στοιχεία είναι στενά συνδεδεμένα, μπορεί να είναι δύσκολο να υιοθετηθούν νέες τεχνολογίες ή να αναβαθμιστούν οι υπάρχουσες χωρίς να επηρεαστεί ολόκληρη η εφαρμογή. Αυτό μπορεί να εμποδίσει την καινοτομία και να δυσχεράνει την αξιοποίηση των πιο πρόσφατων εργαλείων και πλαισίων.

3.2 Αρχιτεκτονική μικροϋπηρεσιών

Η αρχιτεκτονική μικροϋπηρεσιών [7] είναι μια σύγχρονη προσέγγιση στην ανάπτυξη λογισμικού που δομεί μια εφαρμογή ως μια συλλογή χαλαρά συνδεδεμένων υπηρεσιών, καθεμία από τις οποίες αντιπροσωπεύει μια συγκεκριμένη επιχειρηματική λειτουργία ή δυνατότητα. Κάθε μικροϋπηρεσία αναπτύσσεται και συντηρείται ανεξάρτητα, με τη δική της βάση κώδικα, αποθήκευση δεδομένων και πρωτόκολλα επικοινωνίας. Η αρχιτεκτονική των μικροϋπηρεσιών προωθεί την ευελιξία, την επεκτασιμότητα και την ανθεκτικότητα, επιτρέποντας στις ομάδες να αναπτύσσουν ανεξάρτητα, να κλιμακώνουν τα στοιχεία

ξεχωριστά και να υιοθετούν τεχνολογίες που ταιριάζουν καλύτερα στις απαιτήσεις κάθε υπηρεσίας.

[Εικόνα 2]: [Απεικόνιση Διαγράμματος Εφαρμογής μικροϋπηρεσιών]



3.2.1 Πλεονεκτήματα και μειονεκτήματα αρχιτεκτονικής μικροϋπηρεσιών

Όσον αφορά τα πλεονεκτήματα της αρχιτεκτονικής μικροϋπηρεσιών έχουμε τα εξής:

- **Επεκτασιμότητα και ευελιξία:** Η αρχιτεκτονική των μικροϋπηρεσιών επιτρέπει στους οργανισμούς να κλιμακώνουν και να εξελίσσουν τις εφαρμογές τους πιο αποτελεσματικά. Δεδομένου ότι κάθε υπηρεσία λειτουργεί ανεξάρτητα, οι ομάδες μπορούν να αναπτύσσουν ενημερώσεις και να εισάγουν νέα χαρακτηριστικά χωρίς να επηρεάζουν ολόκληρη την εφαρμογή, οδηγώντας σε ταχύτερους κύκλους ανάπτυξης και μεγαλύτερη ευελιξία.

- **Τεχνολογική ποικιλομορφία:** Οι μικροϋπηρεσίες επιτρέπουν στους οργανισμούς να χρησιμοποιούν μια ποικιλία τεχνολογιών και γλωσσών προγραμματισμού για διαφορετικές υπηρεσίες, ανάλογα με τις συγκεκριμένες απαιτήσεις τους. Αυτή η ευελιξία επιτρέπει στις ομάδες να επιλέγουν τα καταλληλότερα εργαλεία και πλαίσια για κάθε στοιχείο, προωθώντας την καινοτομία και την προσαρμοστικότητα.

- **Απομόνωση σφαλμάτων και ανθεκτικότητα:** Με την απομόνωση των υπηρεσιών μεταξύ τους, η αρχιτεκτονική μικροϋπηρεσιών βελτιώνει την ανοχή σε σφάλματα και την ανθεκτικότητα. Εάν μια υπηρεσία αποτύχει ή αντιμετωπίσει προβλήματα, αυτό δεν επηρεάζει απαραίτητα ολόκληρη την εφαρμογή, καθώς άλλες υπηρεσίες μπορούν να συνεχίσουν να λειτουργούν ανεξάρτητα.

Αντιθέτως, σχετικά με τα μειονεκτήματα, έχουμε αντίστοιχα:

- **Αυξημένη πολυπλοκότητα:** Η διαχείριση ενός κατανεμημένου συστήματος που αποτελείται από πολλαπλές μικροϋπηρεσίες εισάγει πολυπλοκότητα όσον αφορά την ανάπτυξη, την παρακολούθηση και την ενορχήστρωση. Οι οργανισμοί πρέπει να επενδύσουν σε ισχυρές πρακτικές και εργαλεία για την αποτελεσματική διαχείριση και συντήρηση εφαρμογών που βασίζονται σε αυτές.

- **Λειτουργική επιβάρυνση:** Η αρχιτεκτονική μικροϋπηρεσιών μπορεί να οδηγήσει σε αυξημένο λειτουργικό φόρτο, καθώς οι ομάδες πρέπει να διαχειριστούν μεγαλύτερο αριθμό υπηρεσιών, καθεμία με τις δικές της απαιτήσεις ανάπτυξης, παρακολούθησης και κλιμάκωσης. Αυτό μπορεί να οδηγήσει σε υψηλότερο κόστος υποδομής και διοικητική πολυπλοκότητα.

- **Προκλήσεις κατανεμημένων συστημάτων:** Η επικοινωνία μεταξύ μικροϋπηρεσιών εισάγει καθυστέρηση και επιβάρυνση δικτύου, ιδίως σε κατανεμημένα περιβάλλοντα. Οι οργανισμοί πρέπει να σχεδιάζουν προσεκτικά τις αλληλεπιδράσεις των υπηρεσιών και να εφαρμόζουν ανθεκτικά πρότυπα επικοινωνίας για να μετριάσουν αυτές τις προκλήσεις.

3.2.2 Πρωτόκολλα επικοινωνίας μικροϋπηρεσιών

Στην αρχιτεκτονική μικροϋπηρεσιών, η επικοινωνία μεταξύ των υπηρεσιών [8] είναι ζωτικής σημασίας για την εξασφάλιση απρόσκοπτης αλληλεπίδρασης και συντονισμού. Δύο συχνά χρησιμοποιούμενα πρωτόκολλα για την επικοινωνία τους είναι το HTTP (Hypertext Transfer Protocol) και το AMQP (Advanced Message Queuing Protocol). Το HTTP χρησιμοποιείται ευρέως λόγω της απλότητάς του, της συμβατότητας με τα πρότυπα ιστού και της ευκολίας υλοποίησής του. Διευκολύνει τη σύγχρονη επικοινωνία μεταξύ υπηρεσιών μέσω αλληλεπιδράσεων αίτησης-απόκρισης, καθιστώντας το κατάλληλο για σενάρια όπου

απαιτούνται απαντήσεις σε πραγματικό χρόνο, όπως κλήσεις API και εφαρμογές ιστού. Από την άλλη πλευρά, το AMQP είναι ένα πρωτόκολλο ανταλλαγής μηνυμάτων σχεδιασμένο για ασύγχρονη επικοινωνία, όπου τα μηνύματα ανταλλάσσονται μεταξύ υπηρεσιών μέσω ουρών αναμονής. Το AMQP επιτρέπει αποσυνδεδεμένα μοτίβα επικοινωνίας, επιτρέποντας στις υπηρεσίες να λειτουργούν ανεξάρτητα και ασύγχρονα, γεγονός που είναι πλεονεκτικό για σενάρια που απαιτούν αρχιτεκτονικές που βασίζονται σε συμβάντα, χρονοπρογραμματισμό εργασιών και αξιόπιστη παράδοση μηνυμάτων. Αξιοποιώντας τόσο το πρωτόκολλο HTTP όσο και το πρωτόκολλο AMQP, οι αρχιτεκτονικές μικροϋπηρεσιών μπορούν να πετύχουν μια ισορροπία μεταξύ σύγχρονης και ασύγχρονης επικοινωνίας, καλύπτοντας διάφορες απαιτήσεις εφαρμογών και εξασφαλίζοντας ισχυρές και αποδοτικές αλληλεπιδράσεις υπηρεσιών.

3.3 Διαδίκτυο των Αντικειμένων - Internet of Things

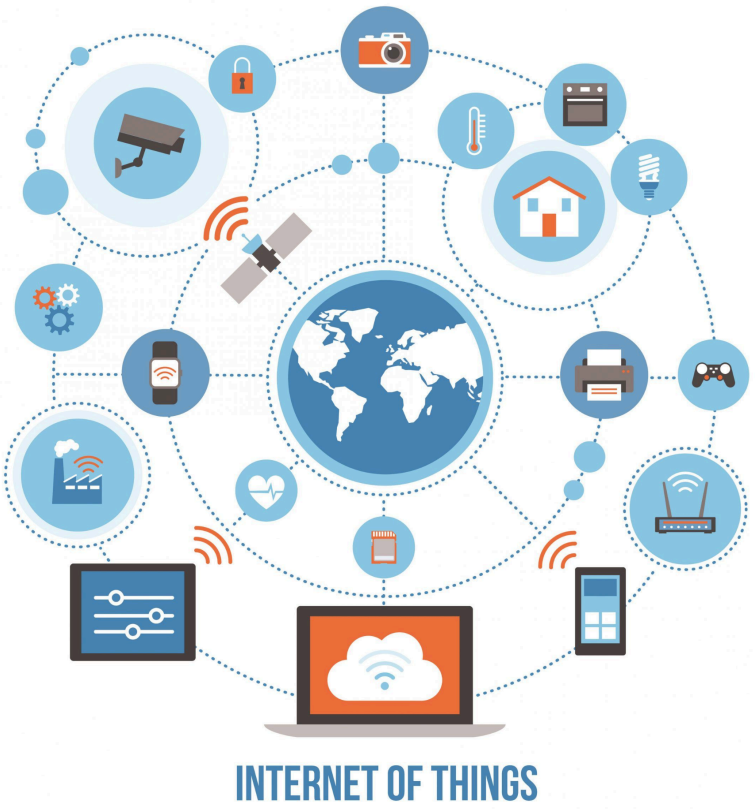
Ο όρος IoT (Internet of Things) αναφέρεται σε ένα δίκτυο συνδεδεμένων συσκευών, αισθητήρων και αντικειμένων καθημερινής χρήσης με το διαδίκτυο, δίνοντάς τους τη δυνατότητα να συλλέγουν, να ανταλλάσσουν και να αναλύουν δεδομένα αυτόνομα. Στη σύγχρονη εποχή, το IoT έχει αποκτήσει ολοένα και μεγαλύτερη σημασία λόγω του βαθύτατου αντίκτυπού του σε διάφορες πτυχές της ζωής μας και των βιομηχανιών.

Οι τεχνολογίες IoT επιτρέπουν τη δημιουργία έξυπνων περιβαλλόντων και συστημάτων, από έξυπνα σπίτια και πόλεις μέχρι βιομηχανικούς αυτοματισμούς και λύσεις υγείας.Επιπλέον, το IoT διαδραματίζει κρίσιμο ρόλο στην αντιμετώπιση σύγχρονων προκλήσεων όπως η αστικοποίηση, η διαχείριση των πόρων και η περιβαλλοντική βιωσιμότητα. Οι πρωτοβουλίες έξυπνων πόλεων αξιοποιούν αισθητήρες IoT και αναλύσεις δεδομένων για την παρακολούθηση και τη διαχείριση των αστικών υποδομών, τη μείωση της κατανάλωσης ενέργειας και τη βελτίωση των δημόσιων υπηρεσιών. Παρομοίως, τα συστήματα περιβαλλοντικής παρακολούθησης που υποστηρίζονται από το IoT διευκολύνουν την παρακολούθηση της ποιότητας του αέρα και του νερού σε πραγματικό χρόνο, συμβάλλοντας στον μετριασμό της ρύπανσης και την προστασία των φυσικών πόρων. Εκτός από τις βιομηχανικές και περιβαλλοντικές εφαρμογές του, το IoT έχει σημαντικές επιπτώσεις στην υγειονομική περίθαλψη, με τη δυνατότητα να φέρει επανάσταση στη φροντίδα των ασθενών, τη διαχείριση των ασθενειών και την ιατρική έρευνα. Οι φορητές συσκευές, τα συστήματα απομακρυσμένης παρακολούθησης και τα έξυπνα ιατρικά εμφυτεύματα επιτρέπουν τη

συνεχή παρακολούθηση της υγείας, την έγκαιρη ανίχνευση προβλημάτων υγείας και τις εξατομικευμένες θεραπευτικές προσεγγίσεις, οδηγώντας σε βελτιωμένα αποτελέσματα υγείας και μειωμένες δαπάνες υγειονομικής περίθαλψης.

Συνολικά, η σημασία του IoT στη σύγχρονη εποχή έγκειται στην ικανότητά του να προωθεί την καινοτομία, να βελτιώνει την αποτελεσματικότητα και να αντιμετωπίζει τις κοινωνικές προκλήσεις σε διάφορους τομείς. Καθώς οι τεχνολογίες IoT συνεχίζουν να εξελίσσονται και να πολλαπλασιάζονται, αναμένεται να διαδραματίσουν ολοένα και πιο κεντρικό ρόλο στη διαμόρφωση του διασυνδεδεμένου κόσμου μας και στην προώθηση του ψηφιακού μετασχηματισμού σε όλους τους κλάδους και τις κοινωνίες.

[Εικόνα 3]: [Απεικόνιση IoT]



3.4 Εικονοποίηση - Virtualization

Η εικονικοποίηση είναι μια θεμελιώδης τεχνολογία [9] που επιτρέπει τη δημιουργία εικονικών εκδόσεων υπολογιστικών πόρων, συμπεριλαμβανομένων διακομιστών, συσκευών

αποθήκευσης και δικτύων. Με την αφαίρεση του φυσικού υλικού από τις εφαρμογές λογισμικού, η εικονικοποίηση επιτρέπεται η εκτέλεση πολλαπλών εικονικών στιγμιοτύπων σε ένα μόνο φυσικό μηχάνημα, γνωστό ως hypervisor. Αυτή η τεχνολογία έχει φέρει επανάσταση στον τομέα του IT βελτιώνοντας τη χρήση των πόρων, την επεκτασιμότητα και την ευελιξία. Επιτρέπει την ενοποίηση των φόρτων εργασίας, μειώνοντας το κόστος του υλικού και την κατανάλωση ενέργειας, ενώ παράλληλα βελτιώνει την αποδοτικότητα του διακομιστή. Επιπλέον, διευκολύνει την διαχείριση και παροχή πόρων, επιτρέποντας στις ομάδες πληροφορικής να κατανέμουν δυναμικά υπολογιστικούς πόρους ανάλογα με τη ζήτηση. Επιπλέον, η εικονικοποίηση ενισχύει την ανθεκτικότητα του συστήματος και τις δυνατότητες αποκατάστασης από καταστροφές, καθώς οι εικονικές μηχανές μπορούν εύκολα να μεταφερθούν ή να αντιγραφούν σε φυσικούς κεντρικούς υπολογιστές.

Συνολικά, η εικονικοποίηση έχει γίνει ακρογωνιαίος λίθος των σύγχρονων αρχιτεκτονικών κέντρων δεδομένων, δίνοντας τη δυνατότητα στους οργανισμούς να βελτιστοποιούν την υποδομή IT τους, να εξορθολογίζουν τις λειτουργίες τους και να προσαρμόζονται στις εξελισσόμενες επιχειρηματικές ανάγκες.

3.4.1 Περιέκτες - Containers

Οι περιέκτες είναι μια ελαφριά μορφή εικονικοποίησης που πακετάρει τις εφαρμογές και τις εξαρτήσεις τους σε απομονωμένα, φορητά περιβάλλοντα. Σε αντίθεση με την παραδοσιακή εικονικοποίηση, η οποία εικονικοποιεί ολόκληρο το λειτουργικό σύστημα, τα containers μοιράζονται τον πυρήνα του κεντρικού λειτουργικού συστήματος, οδηγώντας σε χαμηλότερη επιβάρυνση και ταχύτερους χρόνους εκκίνησης. Οι περιέκτες χρησιμοποιούν πλατφόρμες εμπορευματοποίησης όπως το Docker (θα αναλυθεί σε επόμενο κεφάλαιο) για την ενθυλάκωση εφαρμογών, βιβλιοθηκών και αρχείων ρυθμίσεων, εξασφαλίζοντας συνέπεια και φορητότητα σε διαφορετικά υπολογιστικά περιβάλλοντα.

Επιτρέπουν στους προγραμματιστές να δημιουργούν, να δοκιμάζουν και να αναπτύσσουν εφαρμογές ταχύτερα και αξιόπιστα, καθώς οι εφαρμογές που τρέχουν σε περιέκτες μπορούν εύκολα να αναπτυχθούν σε διαφορετικά περιβάλλοντα, από φορητούς υπολογιστές ανάπτυξης έως διακομιστές παραγωγής. Ωστόσο, παρουσιάζουν προκλήσεις που σχετίζονται με την ασφάλεια, τη δικτύωση και την πολυπλοκότητα της διαχείρισης, απαιτώντας από τους οργανισμούς να εφαρμόσουν ισχυρές λύσεις ενορχήστρωσης. Συνολικά, τα containers έχουν

αναδειχθεί ως μια μετασχηματιστική τεχνολογία για τη σύγχρονη ανάπτυξη και εγκατάσταση εφαρμογών.

3.5 Ενορχήστρωση Περιεκτών - Container Orchestration

Η ενορχήστρωση περιεκτών [10] είναι μια κρίσιμη πτυχή της αποτελεσματικής διαχείρισης και κλιμάκωσης εφαρμογών με containers. Περιλαμβάνει την αυτοματοποίηση της ανάπτυξης, της κλιμάκωσης και της διαχείρισης των φορτίων εργασίας που έχουν ενσωματωθεί σε περιέκτες σε ένα σύμπλεγμα υπολογιστικών πόρων. Με λίγα λόγια, παρομοιάζουν τον ρόλο ενός μαέστρου που είναι υπεύθυνος για την ορθή λειτουργία ενός πληροφοριακού συστήματος, όπου όλες οι υπηρεσίες του είναι χωρισμένα σε περιέκτες.Από το προηγούμενο καταλαβαίνουμε πως είναι κάτι απίστευτα χρήσιμο κατά την ανάπτυξη λογισμικού με την αρχιτεκτονική μικροϋπηρεσιών, οι οποίες έχουν υλοποιηθεί με την χρήση containers.

4. Τεχνολογίες που χρησιμοποιήθηκαν

Το κεφάλαιο αυτό εξετάζει τις κύριες τεχνολογίες που χρησιμοποιήθηκαν κατά την ανάπτυξη του περιβαλλοντικού συστήματος παρακολούθησης. Περιλαμβάνει μία πληθώρα εργαλείων και αναλύει επίσης τις προκλήσεις και τα πλεονεκτήματα κάθε τεχνολογίας και τη σημασία της επιλογής των κατάλληλων τεχνολογιών για την επιτυχή υλοποίηση του συστήματος.

4.1 Τεχνολογίες ανάπτυξης μικροϋπηρεσιών

4.1.1 Java

Η Java [11] είναι μια ευέλικτη και ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού που φημίζεται κυρίως για την ανεξαρτησία της από το περιβάλλον ανάπτυξής της. Αρχικά, δημιουργήθηκε από τη Sun Microsystems το 1995 και έκτοτε έχει γίνει ακρογωνιαίος λίθος της ανάπτυξης επιχειρηματικού λογισμικού, της εφαρμογών για κινητές συσκευές και ιστοσελίδων. Τα βασικά χαρακτηριστικά της περιλαμβάνουν την αντικειμενοστραφή δομή, την αυτόματη διαχείριση της μνήμης (μέσω της συλλογής σκουπιδιών) και τη δυνατότητα "γράψε μια φορά, τρέξε οπουδήποτε", που επιτρέπει το μοντέλο μεταγλώττισης bytecode. Επιπλέον, το τεράστιο οικοσύστημα βιβλιοθηκών και πλαισίων τρίτων κατασκευαστών την καθιστούν δημοφιλή επιλογή για προγραμματιστές σε διάφορους τομείς.

4.1.2 Spring Boot

Το Spring Boot [12] είναι ένα ισχυρό και ευρέως χρησιμοποιούμενο πλαίσιο για τη δημιουργία διαδικτυακών εφαρμογών και μικροϋπηρεσιών βασισμένων σε Java. Αναπτύχθηκε από την Pivotal Software και κυκλοφόρησε το 2014, το Spring Boot παρέχει στους προγραμματιστές έναν βελτιωμένο τρόπο για τη δημιουργία αυτόνομων εφαρμογών παραγωγικής ποιότητας με ελάχιστη εγκατάσταση και διαμόρφωση. Προσφέρει μια σειρά από χαρακτηριστικά, όπως ενσωματωμένους διακομιστές, διαχείριση εξαρτήσεων, διαχείριση ρυθμίσεων και χαρακτηριστικά έτοιμα για παραγωγή, όπως μετρήσεις, έλεγχοι υγείας και ασφάλεια. Η αρθρωτή αρχιτεκτονική της επιτρέπει στους προγραμματιστές να προσθέτουν ή να αφαιρούν εύκολα στοιχεία ανάλογα με τις απαιτήσεις του έργου. Με το Spring Boot, οι προγραμματιστές μπορούν να αναπτύσσουν και να αναπτύσσουν γρήγορα εφαρμογές, εστιάζοντας περισσότερο στην επιχειρησιακή λογική και λιγότερο στις ανησυχίες για την υποδομή. Συνολικά, το Spring Boot έχει γίνει η επιλογή για την κατασκευή ισχυρών και κλιμακούμενων εφαρμογών Java στη σύγχρονη ανάπτυξη λογισμικού.

4.1.3 Maven

Το Apache Maven [13] είναι ένα ευρέως διαδεδομένο εργαλείο αυτοματοποίησης δημιουργίας που χρησιμοποιείται κυρίως για έργα Java. Χρησιμοποιεί μια δηλωτική προσέγγιση για τη διαμόρφωση του έργου μέσω αρχείων POM (Project Object Model) βασισμένων σε XML, τα οποία καθορίζουν τα μεταδεδομένα του έργου, τις εξαρτήσεις, τις φάσεις κατασκευής και τα πρόσθετα. Με το συγκεκριμένο έχει τη δυνατότητα, να κατεβάζει και περιλαμβάνει αυτόματα εξαρτήσεις έργου από απομακρυσμένα αποθετήρια, απλοποιώντας την ενσωμάτωση βιβλιοθηκών τρίτων.

4.1.4 MongoDB

Η MongoDB [14] είναι ένα δημοφιλές, ανοικτού κώδικα σύστημα διαχείρισης βάσεων δεδομένων NoSQL, σχεδιασμένο για ευελιξία, επεκτασιμότητα και υψηλές επιδόσεις. Παρουσιάστηκε το 2009 από την MongoDB Inc. και αποκλίνει από τα παραδοσιακά συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων χρησιμοποιώντας ένα μοντέλο δεδομένων προσανατολισμένο σε έγγραφα αντί για πίνακες και γραμμές. Αποθηκεύει δεδομένα σε ευέλικτα έγγραφα τύπου JSON, καθιστώντας την κατάλληλη για το χειρισμό αδόμητων ή ημιδομημένων δεδομένων και την προσαρμογή σε εξελισσόμενα σχήματα. Ένα από τα βασικά χαρακτηριστικά της, είναι η ικανότητά της να κλιμακώνεται οριζόντια σε πολλαπλούς κόμβους, επιτρέποντας την απρόσκοπτη επέκταση για την προσαρμογή σε αυξανόμενο όγκο δεδομένων και φορτία κίνησης. Προσφέρει πλούσιες δυνατότητες ερωτημάτων, συμπεριλαμβανομένης της υποστήριξης ad-hoc ερωτημάτων, ευρετηρίασης και αγωγών συγκέντρωσης, διευκολύνοντας την αποτελεσματική ανάκτηση και ανάλυση δεδομένων. Το δυναμικό σχήμα και το ευέλικτο μοντέλο δεδομένων της MongoDB δίνουν τη δυνατότητα στους προγραμματιστές να επαναλαμβάνουν γρήγορα, να προσαρμόζονται στις μεταβαλλόμενες απαιτήσεις και να δημιουργούν εφαρμογές πιο αποτελεσματικά.

4.1.5 RabbitMQ

Το RabbitMQ [15] είναι ένα ευρέως χρησιμοποιούμενο λογισμικό διαμεσολαβητή μηνυμάτων ανοικτού κώδικα που διευκολύνει την επικοινωνία μεταξύ διαφορετικών τμημάτων κατανεμημένων εφαρμογών.Παρέχει μια ισχυρή υποδομή ανταλλαγής μηνυμάτων που βασίζεται στο πρότυπο Advanced Message Queuing Protocol (AMQP), καθιστώντας το ιδανικό για τη δημιουργία κλιμακούμενων και αξιόπιστων συστημάτων. Υποστηρίζει

πολλαπλά μοτίβα ανταλλαγής μηνυμάτων, όπως point-to-point, publish/subscribe, request/reply και δρομολόγηση, επιτρέποντας στους προγραμματιστές να σχεδιάζουν σύνθετες ροές επικοινωνίας που να ανταποκρίνονται σε συγκεκριμένες ανάγκες της εφαρμογής. Προσφέρει προηγμένα χαρακτηριστικά, όπως επιβεβαιώσεις μηνυμάτων, ανθεκτικότητα μηνυμάτων και δρομολόγηση μηνυμάτων βάσει κανόνων που ορίζονται από το χρήστη. Το RabbitMQ ενσωματώνεται επίσης απρόσκοπτα με διάφορες γλώσσες προγραμματισμού και πλαίσια μέσω βιβλιοθηκών-πελατών και υποστηρίζει τη διαλειτουργικότητα με άλλα πρωτόκολλα ανταλλαγής μηνυμάτων, όπως το MQTT και το STOMP. Με τις υψηλές επιδόσεις, την επεκτασιμότητα και την ευελιξία του, το RabbitMQ έχει γίνει ακρογωνιαίος λίθος της σύγχρονης αρχιτεκτονικής μικροϋπηρεσιών, τροφοδοτώντας λύσεις ανταλλαγής μηνυμάτων σε ένα ευρύ φάσμα βιομηχανιών και περιπτώσεων χρήσης, όπως cloud computing, microservices, IoT και άλλα.

4.2 Τεχνολογίες ανάπτυξης διεπαφής χρήστη

4.2.1 JavaScript

Η JavaScript [16] είναι μια ευέλικτη και ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού, γνωστή κυρίως για το ρόλο της στην ανάπτυξη ιστοσελίδων. Πρόκειται για μια διερμηνευόμενη γλώσσα υψηλού επιπέδου και εκτελείται από τους φυλλομετρητές ιστού, επιτρέποντας τη δημιουργία σεναρίων από την πλευρά του πελάτη για τον δυναμικό χειρισμό της HTML, της CSS και του Μοντέλου Αντικειμένου Εγγράφου (DOM). Αυτή η δυνατότητα επιτρέπει στους προγραμματιστές να δημιουργούν δυναμικές ιστοσελίδες, να χειρίζονται τις αλληλεπιδράσεις των χρηστών και να επικυρώνουν φόρμες εισόδου σε πραγματικό χρόνο χωρίς να απαιτείται επεξεργασία από την πλευρά του διακομιστή. Το πλούσιο οικοσύστημα βιβλιοθηκών και πλαισίων της JavaScript, συμπεριλαμβανομένων των React, Angular και Vue.js, ενισχύει περαιτέρω τις δυνατότητές της για τη δημιουργία σύνθετων εφαρμογών ιστού, εφαρμογών μιας σελίδας (SPA) και προοδευτικών εφαρμογών ιστού (PWA). Με την πανταχού παρούσα παρουσία της, την ευελιξία της και την εκτεταμένη υποστήριξη της κοινότητας, η JavaScript έχει γίνει απαραίτητο εργαλείο για τους προγραμματιστές ιστού παγκοσμίως, οδηγώντας την καινοτομία και τροφοδοτώντας το σύγχρονο οικοσύστημα του διαδικτύου.

4.2.2 React

Η React [17] είναι μια ισχυρή και ευρέως χρησιμοποιούμενη βιβλιοθήκη JavaScript για την κατασκευή διεπαφών χρήστη (UI), ιδίως για εφαρμογές μιας σελίδας (SPA) και εφαρμογές ιστού με σύνθετα, διαδραστικά στοιχεία UI. Αναπτύχθηκε από το Facebook,κυκλοφόρησε το 2013 και έχει αποκτήσει τεράστια δημοτικότητα μεταξύ των προγραμματιστών λόγω της δηλωτικής και βασισμένης σε στοιχεία προσέγγισής του στην ανάπτυξη UI. Χρησιμοποιεί ένα εικονικό DOM (Document Object Model) για την αποτελεσματική ενημέρωση και απόδοση των στοιχείων, με αποτέλεσμα βελτιωμένη απόδοση και απόκριση σε σύγκριση με τις παραδοσιακές προσεγγίσεις χειρισμού του DOM. Ένα από τα βασικά χαρακτηριστικά της είναι η σύνταξη JSX, η οποία επιτρέπει στους προγραμματιστές να γράφουν κώδικα που μοιάζει με HTML απευθείας μέσα στην JavaScript, βελτιώνοντας την αναγνωσιμότητα και τη συντηρησιμότητα του κώδικα. Με την απλότητα, την επεκτασιμότητα και το εύρωστο οικοσύστημα, η React έχει γίνει μία από τις πιο διάσημες επιλογές για τη δημιουργία σύγχρονων και διαδραστικών διεπαφών χρήστη, τροφοδοτώντας εφαρμογές για ένα ευρύ φάσμα βιομηχανιών και περιπτώσεων χρήσης.

4.3 Τεχνολογίες εικονικοποίησης και ενορχήστωσης

4.3.1 Docker

Το Docker [18] είναι μια ισχυρή πλατφόρμα και ένα σύνολο εργαλείων για την κατασκευή, αποστολή και εκτέλεση εφαρμογών μέσα σε ελαφρούς, φορητούς περιέκτες. Κυκλοφόρησε το 2013 και έφερε επανάσταση στην ανάπτυξη και ανάπτυξη λογισμικού εισάγοντας την τεχνολογία containerization. Τα containers είναι αυτοτελή, απομονωμένα περιβάλλοντα που πακετάρουν μια εφαρμογή και τις εξαρτήσεις της, συμπεριλαμβανομένων των βιβλιοθηκών και των αρχείων διαμόρφωσης, σε μια ενιαία μονάδα. Το Docker παρέχει μια τυποποιημένη μορφή για τη δημιουργία, τη διανομή και την εκτέλεση περιεκτών, εξασφαλίζοντας συνέπεια σε διαφορετικά περιβάλλοντα, από την ανάπτυξη έως την παραγωγή. Ένα από τα βασικά χαρακτηριστικά του είναι η χρήση των εικόνων περιεκτών, οι οποίες χρησιμεύουν ως σχεδιαγράμματα και μπορούν να δημιουργηθούν χρησιμοποιώντας Dockerfiles (αρχεία ρυθμίσεων βασισμένα σε κείμενο που καθορίζουν τα βήματα για τη δημιουργία μιας εικόνας). Το Docker Hub, ένα μητρώο βασισμένο στο cloud, φιλοξενεί ένα τεράστιο αποθετήριο προκατασκευασμένων εικόνων που οι προγραμματιστές μπορούν να αντλήσουν

και να χρησιμοποιήσουν για να βελτιώσουν τις ροές εργασίας τους στην ανάπτυξη. Επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές μία φορά και να τις εκτελούν οπουδήποτε, ανεξάρτητα από την υποκείμενη υποδομή, είτε πρόκειται για διακομιστές στις εγκαταστάσεις, είτε για εικονικές μηχανές, είτε για πλατφόρμες cloud. Επιπλέον, απλοποιεί τη διαδικασία διαχείρισης και κλιμάκωσης των εφαρμογών, καθώς τα containers μπορούν εύκολα να ενορχηστρωθούν και να διαχειριστούν χρησιμοποιώντας πλατφόρμες ενορχήστρωσης εμπορευματοκιβωτίων όπως το Kubernetes ή το Docker Swarm. Με έμφαση στη φορητότητα, την αποδοτικότητα και την επεκτασιμότητα, το Docker έχει γίνει απαραίτητο εργαλείο για τη σύγχρονη ανάπτυξη λογισμικού, επιτρέποντας στους οργανισμούς να επιταχύνουν τις πρακτικές DevOps και να βελτιώσουν την ανάπτυξη αρχιτεκτονικών που βασίζονται σε μικρουπηρεσίες.

4.3.2 Kubernetes

Το Kubernetes [19], συχνά συντομευμένο ως K8s, είναι μια πλατφόρμα ορχήστρωσης ανοικτού κώδικα για εφαρμογές σε επίπεδο περιεκτών, που αρχικά αναπτύχθηκε από τη Google και αργότερα δωρίστηκε στο Cloud Native Computing Foundation. Κυκλοφόρησε το 2014 και αυτοματοποιεί την ανάπτυξη, την επιλογή κλίμακας και τη διαχείριση των εφαρμογών που εκτελούνται σε περιέκτες. Μια από τις βασικές του λειτουργίες είναι η δηλωτική προσέγγιση στη διαχείριση των εφαρμογών, επιτρέποντας στους χρήστες να ορίζουν τις επιθυμητές καταστάσεις χρησιμοποιώντας αρχεία YAML και φροντίζει για τη διατήρηση της πραγματικής κατάστασης ώστε να αντιστοιχεί στην επιθυμητή κατάσταση. Το K8s παρέχει ισχυρά πρωτεύοντα για την ανάπτυξη εφαρμογών, τη διαχείριση αποθήκευσης, το δίκτυο και την κλίμακα των πόρων βάσει της ζήτησης. Υποστηρίζει διάφορες στρατηγικές αναπτύξεων όπως αναβαθμίσεις κύλισης, αναβαθμίσεις blue-green και αναβαθμίσεις canary, εξασφαλίζοντας ενημερώσεις χωρίς διακοπές και υψηλή διαθεσιμότητα. Προσφέρει επίσης ενσωματωμένη ανακάλυψη υπηρεσιών και ισορροπία φορτίου, επιτρέποντας την άψογη επικοινωνία μεταξύ μικροϋπηρεσιών εντός ενός συστήματος. Με τη δυνατότητά του να απλοποιεί και να αυτοματοποιεί τη διαχείριση εφαρμογών containers, το Kubernetes έχει γίνει ο πρότυπος κώδικας για την ενορχήστρωση τους, επιτρέποντας στις οργανώσεις να αναπτύσσουν και να κλιμακώνουν ανθεκτικές και εφαρμογές cloud-native αποτελεσματικά.

4.4 Περιβάλλον Ανάπτυξης Λογισμικού

4.4.1 Intellij IDEA

Το IntelliJ IDEA [20] είναι ένα δημοφιλές και υψηλά εκτιμημένο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που αναπτύχθηκε από την JetBrains. Κυκλοφόρησε αρχικά το 2001 και έχει γίνει ένα απαραίτητο εργαλείο για τους προγραμματιστές Java και χρησιμοποιείται επίσης ευρέως για την ανάπτυξη εφαρμογών σε άλλες γλώσσες προγραμματισμού όπως η Kotlin, η Scala, η Groovy και άλλες. Προσφέρει μια ευρεία γκάμα χαρακτηριστικών σχεδιασμένων για να βελτιώσουν την παραγωγικότητα των προγραμματιστών και να διευκολύνουν τη διαδικασία ανάπτυξης λογισμικού. Μερικά από τα κύρια του χαρακτηριστικά περιλαμβάνουν έξυπνη συμπλήρωση κώδικα, προηγμένα εργαλεία αναδιαμόρφωσης κώδικα, επιθεωρήσεις κώδικα και έξυπνη ανάλυση κώδικα, τα οποία βοηθούν τους προγραμματιστές να γράφουν πιο καθαρό, αποδοτικότερο κώδικα με λιγότερα σφάλματα.

Συνολικά, η ευφυής διεπαφή, το πλούσιο σύνολο χαρακτηριστικών και οι εκτεταμένες επιλογές προσαρμογής καθιστούν το IntelliJ IDEA την προτιμώμενη επιλογή για επαγγελματίες προγραμματιστές που αναζητούν ένα ισχυρό και αποδοτικό περιβάλλον ανάπτυξης.

4.4.2 Visual Studio Code

Το Visual Studio Code (VS Code) [21] είναι ένα δωρεάν πρόγραμμα επεξεργασίας κώδικα που αναπτύχθηκε από τη Microsoft. Κυκλοφόρησε το 2015 και σύντομα έγινε ένα από τα πιο δημοφιλή περιβάλλοντα ανάπτυξης σε πολλές πλατφόρμες, συμπεριλαμβανομένων των Windows, mac OS και Linux. Το VSCode είναι γνωστό για την ελαφρότητά του και την γρήγορη απόδοσή του, καθιστώντας το κατάλληλο για διάφορες γλώσσες προγραμματισμού και εργασίες ανάπτυξης. Προσφέρει ένα πλούσιο σύνολο χαρακτηριστικών σχεδιασμένων για να βελτιώσουν την παραγωγικότητα των προγραμματιστών, συμπεριλαμβανομένης της έξυπνης συμπλήρωσης κώδικα, του συντονισμού σύνταξης και της πλοήγησης κώδικα. Ένα από τα εξαιρετικά χαρακτηριστικά του, είναι η επεκτασιμότητα του μέσω χρήσης plugins. Είτε πρόκειται για προσθήκη υποστήριξης για μια νέα γλώσσα προγραμματισμού, ενσωμάτωση με ένα συγκεκριμένο πλαίσιο, είτε για βελτίωση των δυνατοτήτων του

επεξεργαστή με νέα χαρακτηριστικά, πιθανότατα υπάρχει διαθέσιμο Plugin για να καλύψει τις ανάγκες των προγραμματιστών.

Συνολικά, η συνδυαστική απόδοση, τα χαρακτηριστικά και η επεκτασιμότητα του VSCode έχουν στερεοποιήσει τη θέση του ως μιας κορυφαίας επιλογής μεταξύ των προγραμματιστών παγκοσμίως.

5. Σχεδιασμός και Ανάπτυξη του Συστήματος

5.1 Ανάλυση Λειτουργικών Απαιτήσεων

Μετά από μελέτη και ανάλυση των υπάρχων συστημάτων παρακολούθησης του περιβάλλοντος, οι λειτουργικές απαιτήσεις για την εφαρμογή που θα αναπτυχθεί στα πλαίσια της εργασίας αυτής είναι οι εξής:

- **Αναζήτηση live δεδομένων:** Το σύστημα πρέπει να επιτρέπει την αναζήτηση και προβολή live δεδομένων από τους αισθητήρες για άμεση παρακολούθηση της κατάστασης ανά γεωγραφική περιοχή.

- **Προβολή μετρήσεων από ημερομηνίες που αναζητά ο χρήστης προηγούμενου χρόνου:** Το σύστημα πρέπει να επιτρέπει στους χρήστες να προβάλλουν μετρήσεις από προηγούμενες ημερομηνίες, προσφέροντας έτσι ιστορικό των δεδομένων για ανάλυση.

- **Σύγκριση και ανάλυση μετρήσεων σε βάθος χρόνου:** Μετά την αναζήτηση παλαιών χρόνου, το σύστημα πρέπει να επιτρέπει τη σύγκριση των μετρήσεων σε διαφορετικά χρονικά διαστήματα για ανάλυση της αλλαγής των παραμέτρων με τον χρόνο.

- **Εξασφάλιση υψηλής ακρίβειας των δεδομένων:** Το σύστημα πρέπει να διασφαλίζει την υψηλή ακρίβεια των δεδομένων μέσω ελέγχων κατά την παραγωγή τους από τους αισθητήρες και μηχανισμούς χειρισμού σφαλμάτων.

- **Ειδοποίηση και ενημέρωση χρήστη:** Το σύστημα πρέπει να επιτρέπει την ειδοποίηση των χρηστών μέσω notification, SMS ή email για οποιαδήποτε συνθήκη που ορίζει ο χρήστης, όπως υπέρβαση θερμοκρασίας σε ορισμένη γεωγραφική περιοχή.

- **Επεκτασιμότητα για διαχείριση πολλών ταυτόχρονων αιτημάτων:** Το σύστημα πρέπει να είναι επεκτάσιμο και να μπορεί να διαχειρίζεται πολλαπλούς χρήστες και αιτήματα ταυτόχρονα χωρίς να μειώνεται η απόδοσή του.

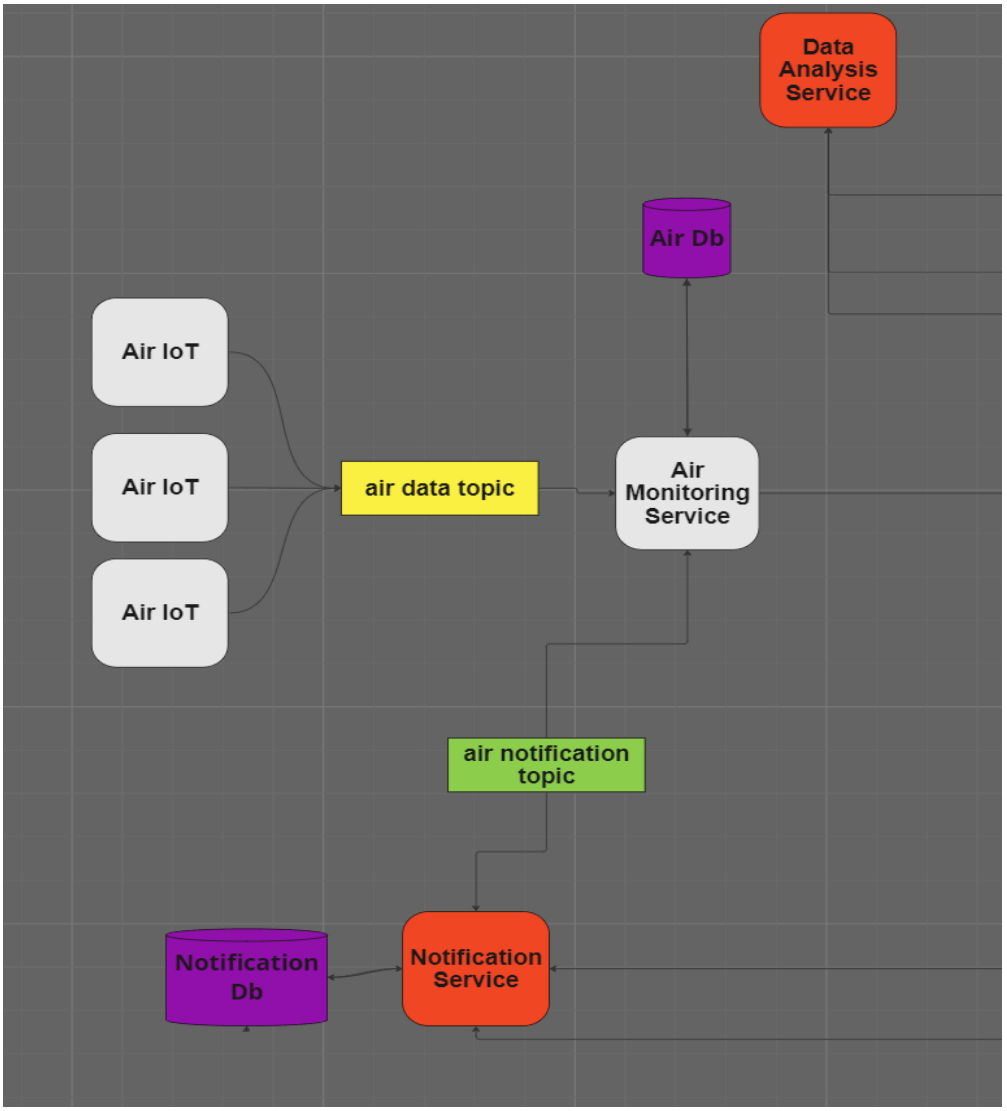
- **Backup συστήματος:** Το σύστημα πρέπει να παρέχει backup σε 2η βάση δεδομένων για να διασφαλίσει την ασφάλεια των δεδομένων και τη συνεχή λειτουργία σε περίπτωση αποτυχίας του κύριου συστήματος.

- **Εξαγωγή δεδομένων αναζήτησης σε μορφή dataset:** Το σύστημα πρέπει να επιτρέπει την εξαγωγή των δεδομένων που προέκυψαν από τις αναζητήσεις σε μορφή αξιοποιήσιμου dataset, προκειμένου να μπορούν να αξιοποιηθούν για περαιτέρω ανάλυση ή επεξεργασία.

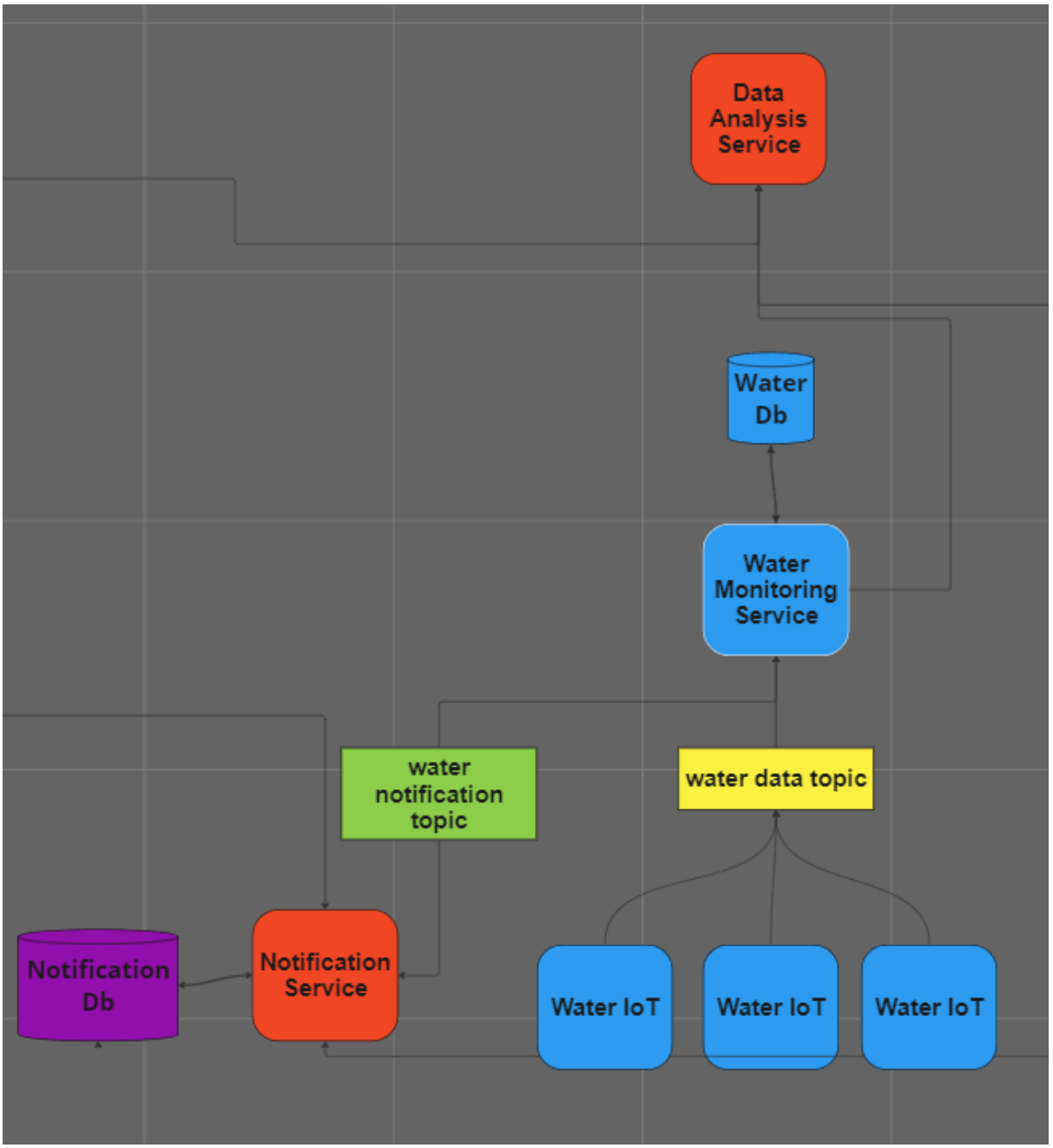
5.2 Αρχιτεκτονική Συστήματος

Παρακάτω φαίνεται το τελικό διάγραμμα της αρχιτεκτονικής του συστήματος, χωρισμένο σε 3 διαφορετικές εικόνες, ανά υπηρεσίες που συνεργάζονται μαζί.

[Εικόνα 4]: [Διάγραμμα μικροϋπηρεσιών αέρα-ανάλυσης -ειδοποίησης]

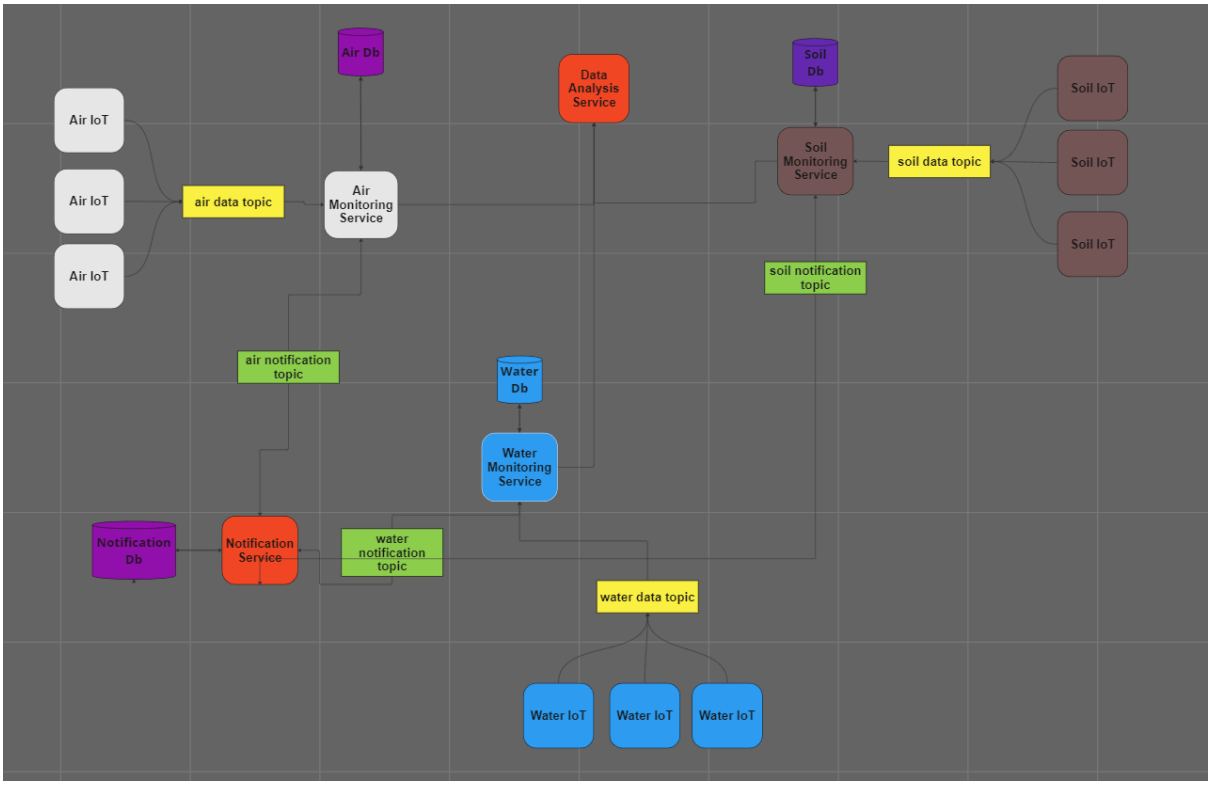


[Εικόνα 5]: [Διάγραμμα μικροϋπηρεσιών νερού-ανάλυσης -ειδοποίησης]



[Εικόνα 6]: [Διάγραμμα μικροϋπηρεσιών εδάφους-ανάλυσης-ειδοποίησης]

[Εικόνα 7]: [Συνολικό διάγραμμα συστήματος]



Πιο συγκεκριμένα, όπου αναγράφεται **Air IoT**, **Water IoT** και **Soil IoT**, αντιπροσωπεύονται οι σένσορες που μετράνε τα δεδομένα και τα προωθούν σε ουρές με θέμα **air data topic**, **water data topic** και **soil data topic** αντίστοιχα. Στις ουρές αυτές είναι συνδεδεμένες οι μικροϋπηρεσίες του αέρα (**Air Monitoring Microservice**), του νερού (**Water Monitoring Microservice**) και του εδάφους (**Soil Monitoring Microservice**), όπου το καθένα αποθηκεύει σε ξεχωριστή βάση δεδομένων (**MongoDB**) τα αποτελέσματα που μετράνε οι σένσορες. Επιπλέον, οι μικροϋπηρεσίες αυτές συνδέονται σε άλλες ουρές **air notification topic**, **water notification topic** και **soil notification topic** για επικοινωνία με το **Notification Microservice**, όταν αυτό κριθεί απαραίτητο. Η ασύγχρονη αυτή επικοινωνία των υπηρεσιών αυτών γίνεται μέσω AMQP υλοποιημένη με RabbitMQ.

Επιπροσθέτως, οι μικροϋπηρεσίες του αέρα (**Air Monitoring Microservice**), του νερού (**Water Monitoring Microservice**) και του εδάφους (**Soil Monitoring Microservice**) παρέχουν HTTP κλήσεις για αναζήτηση δεδομένων από αυτές, όπως και στο **Data Analysis Microservice**, όταν επιλέγεται από το χρήστη. Θα αναλυθούν παρακάτω στην επεξήγηση κάθε υπηρεσίας ξεχωριστά.

5.3 Ανάπτυξη Συστήματος

Για την ανάπτυξη του συστήματος να σημειωθεί πως οι σένσορες IoT είναι εικονικοί στα πλαίσια της εργασίας αυτής (γι'αυτό δεν θα αναλυθούν ξεχωριστά) αλλά υπάρχει η δυνατότητα εναλλαγής τους με πραγματικούς, όπου απλά θα προωθούν τα δεδομένα που μετράνε στις κατάλληλες ουρές που αναφέρθηκαν.

Επιπροσθέτως, για κάθε υπηρεσία θα παρέχεται το αρχείο **pom.xml** που θα αναφέρει όλα τα dependencies που χρειάστηκαν για την ανάπτυξή του, το **application.properties** που είναι απαραίτητο για τις global μεταβλητές (credentials βάσεις και ουρών), τα σημαντικά σημεία στον κώδικα που δείχνουν τη λειτουργικότητα της υπηρεσίας (θα εξηγούνται αναλυτικά), το **Dockerfile.yaml** , μαζι με το **Docker-Compose.yaml** και τις εντολές που χρειάζονται για την δημιουργία των εικόνων και την εκτέλεση των περιεκτών. Τέλος, τα configuration αρχεία του **K8s** για την ενορχήστρωση των containers. Η μεγαλύτερη ανάλυση θα γίνει πρώτα στην υπηρεσία του αέρα, καθώς το μοτίβο που ακολουθούν οι υπόλοιπες είναι το ίδιο.

5.3.1 Μικροϋπηρεσία Αέρα

Παρακάτω παρατίθενται τα βασικά αρχεία για την μικροϋπηρεσία του αέρα και η λογική υλοποίησης των κύριων σημείων της λειτουργικότητας του.

pom.xml

```
<dependency>
            <groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
```

```
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

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

        <dependency>
            <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId
>
            <version>4.1.0</version>
        </dependency>
```

application.properties

```
# General Config
spring.application.name=air-microservice
server.port=0

# Mongodb config
spring.data.mongodb.authentication-database=${SPRING_DATA_MONGODB_AUTHEN
TICATION}
spring.data.mongodb.database=${SPRING_DATA_MONGODB_DATABASE}
spring.data.mongodb.port=${SPRING_DATA_MONGODB_PORT}
spring.data.mongodb.host=${SPRING_DATA_MONGODB_HOST}

# Rabbitmq config
spring.rabbitmq.host=${SPRING_RABBITMQ_HOST}
spring.rabbitmq.port=${SPRING_RABBITMQ_PORT}
spring.rabbitmq.username=${SPRING_RABBITMQ_USERNAME}
spring.rabbitmq.password=${SPRING_RABBITMQ_PASSWORD}

rabbitmq.air.queue.name=${RABBITMQ_AIR_QUEUE_NAME}
rabbitmq.exchange.name=${RABBITMQ_EXCHANGE_NAME}
```

```
rabbitmq.air.routing.key=${RABBITMQ_AIR_ROUTING_KEY}

# Service discover config
eureka.client.service-url.defaultZone=${EUREKA_CLIENT_SERVICE_URL}
eureka.instance.prefer-ip-address=true
```

Οι τιμές των μεταβλητών δίνονται μέσα από τα αρχεία του docker-compose και του K8s για μεγαλύτερη ασφάλεια, όπως γίνεται και σε ένα παραγωγικό περιβάλλον.

Air Data Model

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Document(collection = "air-data")
public class AirData {

    private Double airTemperature;
    private Double airHumidity;
    private Double airC02;
    private Double airVOCs;
    private Double airPM25;
    private Double airC0;
    @CreatedDate
    private Instant timestamp;
    private String coordinates;

}
```

Παραπάνω παρουσιάζεται το μοντέλο των δεδομένων που θα λαμβάνεται από τους σένσορες και θα αποθηκεύεται στη βάση δεδομένων.

Air Data Service

```
 public List<AirDataDto> findByLocationAndDates(String
coordinates, Instant  start, Instant  end){
        List<AirData> res = null;

        res =
airDataRepository.findByTimestampAndCoordinatesOOrderByTimestampAi
rHumidityAsc(start,end,coordinates);
```

```
        return res.stream()
                .map(this::mapAirDataToAirDataDto)
                .collect(Collectors.toList());
    }

    public AirDataDto findLastEntry(String location){
        return
mapAirDataToAirDataDto(airDataRepository.findFirstByCoordinatesOrd
erByTimestampDesc(location));
    }
```

Παραπάνω παρουσιάζονται οι μέθοδοι που μας επιστρέφουν είτε την τελευταία εγγραφή στη βάση (δηλαδή live τιμή), είτε κάνει αναζήτηση σε παρελθοντικές τιμές και επιστρέφει μία λίστα τιμών. Τα παραπάνω προωθούνται στον "έξω κόσμο" με http μεθόδους όπως φαίνεται εδώ:

Air Controller

```
@RestController
@RequestMapping("/air")
@RequiredArgsConstructor
public class AirDataController {

    private final AirDataService airDataService;

    @GetMapping("/search-all")
    public ResponseEntity<List<AirDataDto>> getData(
            @RequestParam String location,
            @RequestParam @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) Instant start,
            @RequestParam @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) Instant  end
    ){

        return new
ResponseEntity<>(airDataService.findByLocationAndDates(location,st
art,end), HttpStatus.OK);
    }
```

```
    @GetMapping("/search")
    public ResponseEntity<AirDataDto> getData(
            @RequestParam String location
    ){
        return new
ResponseEntity<>(airDataService.findLastEntry(location),
HttpStatus.OK);
    }

}
```

RabbitMQ configuration

```
@Configuration
public class RabbitMQConfig {

    @Value("${rabbitmq.air.queue.name}")
    private String queue;

    @Value("${rabbitmq.exchange.name}")
    private String exchange;

    @Value("${rabbitmq.air.routing.key}")
    private String routingKey;

    @Bean
    public Queue queue(){
        return  new Queue(queue);
    }

    @Bean
    public DirectExchange exchange(){
        return new DirectExchange(exchange);
    }

    @Bean
    public Binding binding(){
        return BindingBuilder.bind(queue())
                .to(exchange())
                .with(routingKey);
    }
```

```
    @Bean
    public MessageConverter converter(){
        return new Jackson2JsonMessageConverter();
    }

    @Bean
    public  AmqpTemplate amqpTemplate(ConnectionFactory
connectionFactory){
        RabbitTemplate rabbitTemplate = new
RabbitTemplate(connectionFactory);
        rabbitTemplate.setMessageConverter(converter());
        return  rabbitTemplate;
    }

}
```

Παραπάνω, γίνεται η σύνδεση με τις ουρές, λαμβάνοντας υπόψη τις global μεταβλητές που έχουμε ορίσει στο application.properties

Air User Model

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Document(collection = "user-to-notify")
public class UserToNotify {

    @Id
    private String id;

    private String email;
    private String coordinates;

    private Double lessAirTemperature;
    private Double moreAirTemperature;

    private Double lessAirHumidity;
    private Double moreAirHumidity;
```

```
    private Double lessAirC02;
    private Double moreAirC02;

    private Double lessAirVOCs;
    private Double moreAirVOCs;

    private Double lessAirPM25;
    private Double moreAirPM25;

    private Double lessAirC0;
    private Double moreAirC0;
}
```

Παραπάνω φαίνεται το μοντέλο ενός χρήστη όπου μπορεί να επιλέξει όσες τιμές θέλει για να ειδοποιηθεί και το email του, όπου θα στέλνεται η ειδοποίησή του.

Air User Service

```
 public UserToNotifyService(UserToNotifyRepository
userToNotifyRepository, RabbitTemplate rabbitTemplate) {
        this.userToNotifyRepository = userToNotifyRepository;
        this.rabbitTemplate = rabbitTemplate;
    }

    public UserToNotifyDto userSubToAirData(UserToNotifyDto
userToNotifyDto){

userToNotifyRepository.save(mapUserToNotifyDtoToUserToNotify(userToNotif
yDto));
        return userToNotifyDto;
    }

    public void deleteUser(String email){
        List<UserToNotify> userToNotifyList =
userToNotifyRepository.findByEmail(email);
        userToNotifyRepository.deleteAll(userToNotifyList);
    }

    public void getNotifyUserList(AirData airData){
        List<UserToNotify> userToNotifyList =
userToNotifyRepository.findByCoordinatesAndLessAirTemperature(airData.ge
```

```
tCoordinates(), airData.getAirTemperature());

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAndMoreA
irTemperature(airData.getCoordinates(), airData.getAirTemperature()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAndLessA
irHumidity(airData.getCoordinates(), airData.getAirHumidity()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAndMoreA
irHumidity(airData.getCoordinates(), airData.getAirHumidity()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAndLessA
irC02(airData.getCoordinates(), airData.getAirC02()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAndMoreA
irC02(airData.getCoordinates(), airData.getAirC02()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAndLessA
irVOCs(airData.getCoordinates(), airData.getAirVOCs()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAndMoreA
irVOCs(airData.getCoordinates(), airData.getAirVOCs()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAndLessA
irPM25(airData.getCoordinates(), airData.getAirPM25()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAndMoreA
irPM25(airData.getCoordinates(), airData.getAirPM25()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAndLessA
irC0(airData.getCoordinates(), airData.getAirC0()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAndMoreA
irC0(airData.getCoordinates(), airData.getAirC0()));

        List<String> emails = userToNotifyList.stream()
                .map(UserToNotify::getEmail).toList();

        AirQueueNotifyUsersDto airQueueNotifyUsersDto = new
AirQueueNotifyUsersDto();

airQueueNotifyUsersDto.setAirData(mapAirDataToAirQueueDataDto(airData));
        airQueueNotifyUsersDto.setEmails(emails);

        if(!emails.isEmpty()){
```

```
rabbitTemplate.convertAndSend(exchange,routingKey,airQueueNotifyUsersDto
);
        }

    }
```

Παραπάνω βρίσκονται οι μέθοδοι που αποθηκεύουν στη βάση δεδομένων τις τιμές για κάθε χρήστη και επίσης τη μέθοδο που στέλνει στην ουρά για επικοινωνία με το **Notification Microservice**. Επίσης, οι μέθοδοι για αποθήκευση ενός χρήστη προωθούνται στον έξω κόσμο με http εδώ:

Air User Controller

```
@RestController
@RequestMapping("/air")
@RequiredArgsConstructor
public class UserToNotifyController {

    private final UserToNotifyService userToNotifyService;

    @PostMapping("/notify")
    public ResponseEntity<UserToNotifyDto>
userSubToAirData(@RequestBody UserToNotifyDto userToNotifyDto){
        return new
ResponseEntity<>(userToNotifyService.userSubToAirData(userToNotify
Dto), HttpStatus.OK);
    }

    @DeleteMapping ("/notify")
    public void userSubToAirData(@RequestParam String email){
        userToNotifyService.deleteUser(email);
    }
}
```

Dockerfile

```
FROM openjdk:17
EXPOSE 8080:8080
ADD target/air-microservice.jar air-microservice.jar
ENTRYPOINT ["java","-jar","/air-microservice.jar"]
```

Στο παραπάνω dockerfile δηλώνεται η έκδοση της java που θέλουμε να χτιστεί η εικόνα της

εφαρμογής μας, η πόρτα στην οποία θέλουμε να τρέχει τόσο στο φυσικό, όσο στο εικονικό μηχάνημα και το εκτελέσιμο αρχείο μαζί με την εντολή που θα ξεκινάει την υπηρεσία. Χρησιμοποιώντας την παρακάτω εντολή θα δημιουργηθεί η εικόνα:

```
docker build -t air-microservice .
```

Dockerfile Compose

```
version: '3.8'

services:
  air-microservice:
    image: air-microservice:latest
    ports:
      - "8080"
    environment:
      SPRING_DATA_MONGODB_HOST: mongodb
      SPRING_DATA_MONGODB_PORT: 27017
      SPRING_DATA_MONGODB_DATABASE: air
      SPRING_DATA_MONGODB_AUTHENTICATION: admin
      SPRING_RABBITMQ_HOST: rabbitmq
      SPRING_RABBITMQ_PORT: 5672
      SPRING_RABBITMQ_USERNAME: guest
      SPRING_RABBITMQ_PASSWORD: guest
      RABBITMQ_AIR_QUEUE_NAME: air.notification.queue
      RABBITMQ_EXCHANGE_NAME: notification.exchange
      RABBITMQ_AIR_ROUTING_KEY: air.queue.user.notification
      EUREKA_CLIENT_SERVICE_URL: http://eureka-server:8761/eureka/
    depends_on:
      - mongodb
    networks:
      - environment-pollution-network

  mongodb:
    image: mongo:latest
    ports:
      - "27017:27017"
    volumes:
      - type: bind
        source: Documents/pollution/air-db
        target: /pollution/air-db
    networks:
      - environment-pollution-network
```

```
networks:
  environment-pollution-network:
    external: true
    name: project_environment-pollution-network
```

Με το παραπάνω docker compose δημιουργείται ο περιέκτης της υπηρεσίας του αέρα, δηλώνοντας όλες τις global μεταβλητές που χρειάζονται για το application.properties, η εικόνα της βάσης και ένα δίκτυο ώστε να μπορεί να επικοινωνήσει με τα υπόλοιπα containers. Παρακάτω δηλώνονται οι εντολές για δημιουργία δικτύου και εκτέλεσης του περιέκτη:

```
docker  network create environment-pollution-network
docker compose -f air-compose.yaml up
```

Εναλλακτικά, για να τρέξουν περισσότερα στιγμιότυπα της εφαρμογής, υπάρχει η επιλογή:

```
docker compose -f air-compose.yaml up --scale air-microservice=2
```

Προς το παρόν η υπηρεσία μας μπορεί να τρέξει κανονικά, αλλά για λόγους καλύτερης κλιμάκωσης θα αφήσουμε το K8s να αναλάβει την ενορχήστρωση. Το μόνο που χρειάζεται είναι η εικόνα της υπηρεσίας που δημιουργήσαμε πριν και κάποια έξτρα αρχεία που θα παρατεθούν παρακάτω:

K8s deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: air-microservice
spec:
  replicas: 2
  selector:
    matchLabels:
      app: air-microservice
  template:
    metadata:
      labels:
        app: air-microservice
```

```
    spec:
      containers:
        - name: air-microservice
          image: "themisvas/air-microservice:latest"
          ports:
            - containerPort: 8080
          env:
            - name: SPRING_DATA_MONGODB_HOST
              value: air-mongodb
            - name: SPRING_DATA_MONGODB_PORT
              value: "27017"
            - name: SPRING_DATA_MONGODB_DATABASE
              value: air
            - name: SPRING_DATA_MONGODB_AUTHENTICATION
              value: admin
            - name: SPRING_RABBITMQ_HOST
              value: rabbitmq
            - name: SPRING_RABBITMQ_PORT
              value: "5672"
            - name: SPRING_RABBITMQ_USERNAME
              valueFrom:
                secretKeyRef:
                  name: rabbitmq-secret
                  key: rabbitmq-username
            - name: SPRING_RABBITMQ_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: rabbitmq-secret
                  key: rabbitmq-password
            - name: RABBITMQ_AIR_QUEUE_NAME
              value: air.notification.queue
            - name: RABBITMQ_EXCHANGE_NAME
              value: notification.exchange
            - name: RABBITMQ_AIR_ROUTING_KEY
              value: air.queue.user.notification
```

K8s Service

```
apiVersion: v1
kind: Service
metadata:
```

```
  name: air-microservice
spec:
  type: LoadBalancer
  selector:
    app: air-microservice
  ports:
    - port: 80
      targetPort: 8080
```

K8s Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: rabbitmq-secret
type: Opaque
data:
  rabbitmq-username: Z3Vlc3Q=
  rabbitmq-password: Z3Vlc3Q=
```

K8s DB Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: air-mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: air-mongodb
  template:
    metadata:
      labels:
        app: air-mongodb
    spec:
      containers:
        - name: air-mongodb
          image: mongo:latest
          ports:
            - containerPort: 27017
          volumeMounts:
            - name: air-mongodb-data
              mountPath: /data/air-db
  volumes:
```

```
    - name: air-mongodb-data
      persistentVolumeClaim:
        claimName: air-mongodb-data
```

K8s DB Service

```
apiVersion: v1
kind: Service
metadata:
  name: air-mongodb
spec:
  selector:
    app: air-mongodb
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
```

K8s DB Persistence Volume

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: air-mongodb-data
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

Τα παραπάνω yaml αρχεία δηλώνουν τα εξής:

- **K8s Deployment:**
    - Διαχειρίζεται έναν αντίστοιχο σύνολο από pods.
    - Ορίζει την επιθυμητή κατάσταση για έναν ομάδα από ταυτόσημα pods, διασφαλίζοντας ένα καθορισμένο αριθμό αντιγράφων να εκτελούνται.
    - Ορίζει το πρότυπο ελέγχου του container, συμπεριλαμβανομένης της εικόνας, των θυρών και των μεταβλητών περιβάλλοντος.

- **K8s Service:**
    - Παρέχει ένα συνεπές τρόπο πρόσβασης σε ένα σύνολο από pods.
    - Ο τύπος Load Balancer εκθέτει την υπηρεσία εξωτερικά χρησιμοποιώντας έναν εξισορροπητή φορτίου του παροχέα νέφους.
    - Δρομολογεί την κίνηση προς τα pods βάσει των ετικετών.

- **K8s Secret:**
    - Αποθηκεύει ευαίσθητες πληροφορίες όπως κωδικοί πρόσβασης, τεκμηριώσεις OAuth και κλειδιά SSH ασφαλώς στον αρθρωτό χώρο του Kubernetes.
    - Τα δεδομένα κωδικοποιούνται σε Base64 και αποθηκεύονται, τα οποία αποκωδικοποιούνται όταν προσπελαύνονται από εξουσιοδοτημένους χρήστες.
- **K8s DB Deployment:**
    - Διαχειρίζεται την κατανομή του MongoDB, ενός συστήματος βάσης δεδομένων, στον χώρο του Kubernetes.
    - Ορίζει τον αριθμό των αντιγράφων, το πρότυπο του container και τα σημεία τοποθέτησης όγκου.
- **K8s DB Service:**
    - Παρέχει πρόσβαση δικτύου στα pods MongoDB εντός του χώρου του Kubernetes.
    - Δρομολογεί την κίνηση προς τα pods MongoDB βάσει των ετικετών.
- **K8s DB Persistence Volume:**
- Αιτείται πόρους αποθήκευσης από την υποκείμενη υποδομή (όπως παροχείς νέφους ή συστήματα αποθήκευσης εντός εγκατάστασης).
- Παρέχει διαρκή αποθήκευση για τα δεδομένα MongoDB.
- Εξασφαλίζει τη διατήρηση των δεδομένων ακόμη και εάν τα pods επανεκκινούνται ή επαναπρογραμματίζονται.

Το περιβάλλον του K8s θα τρέξει τοπικά με τη βοήθεια του minikube (εργαλείο για δοκιμή K8s τοπικά και όχι σε cloud περιβάλλον), με τις εξής εντολές:

```
minikube start
minikube stop
kubectl apply -f air-deployment.yaml
```

Με τα παραπάνω έχουμε καταφέρει η υπηρεσία μας να τρέχει σε κατανεμημένο περιβάλλον ενορχηστρωμένο μέσω K8s.

5.3.2 Μικροϋπηρεσία Νερού

Παρακάτω παρατίθενται τα βασικά αρχεία για την μικροϋπηρεσία του νερού και η λογική υλοποίησης των κύριων σημείων της λειτουργικότητας του.

pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-mongodb</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-amqp</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
```

```
<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
            <version>4.1.0</version>
        </dependency>
    </dependencies>
```

application.properties

```
# General Config
spring.application.name=water-microservice
server.port=0

# Mongo db config
spring.data.mongodb.authentication-database=${SPRING_DATA_MONGODB_
AUTHENTICATION}
spring.data.mongodb.database=${SPRING_DATA_MONGODB_DATABASE}
spring.data.mongodb.port=${SPRING_DATA_MONGODB_PORT}
spring.data.mongodb.host=${SPRING_DATA_MONGODB_HOST}

# Rabbit mq config
spring.rabbitmq.host=${SPRING_RABBITMQ_HOST}
spring.rabbitmq.port=${SPRING_RABBITMQ_PORT}
spring.rabbitmq.username=${SPRING_RABBITMQ_USERNAME}
spring.rabbitmq.password=${SPRING_RABBITMQ_PASSWORD}

rabbitmq.water.queue.name=${RABBITMQ_WATER_QUEUE_NAME}
rabbitmq.exchange.name=${RABBITMQ_EXCHANGE_NAME}
rabbitmq.water.routing.key=${RABBITMQ_WATER_ROUTING_KEY}

# Service discover config
eureka.client.service-url.defaultZone=${EUREKA_CLIENT_SERVICE_URL}
eureka.instance.prefer-ip-address=true
```

Water Data Model

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Document(collection = "water-data")
public class WaterData {

    private Double dissolvedOxygen;
    private Double oxidationReductionPotential;
```

```
    private Double pH;
    private Double turbidity;
    private Double totalDissolvedSolids;
    private Double temperature;

    @CreatedDate
    private Instant timestamp;
    private String coordinates;
}
```

Water Data Service

```
  public List<WaterDataDto> findByParameter(String coordinates, Instant
start, Instant end){
        List<WaterData> res = null;

        res =
waterDataRepository.findByTimestampAndCoordinates(start,end,coordinates)
;

        return
res.stream().map(this::mapWaterDataToWaterDataDto).collect(Collectors.to
List());
    }

    public WaterDataDto findLastEntry(String location){
        return
mapWaterDataToWaterDataDto(waterDataRepository.findFirstByCoordinatesOrd
erByTimestampDesc(location));
    }
```

Water Controller

```
@RestController
@RequestMapping("/water")
@RequiredArgsConstructor
public class WaterDataController {

    private final WaterDataService waterDataService;

    @GetMapping("/search-all")
    public ResponseEntity<List<WaterDataDto>> getData(
            @RequestParam String location,
            @RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
Instant start,
```

```
            @RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
Instant end
    ){

        return new
ResponseEntity<>(waterDataService.findByParameter(location,start,end),
HttpStatus.OK);
    }

    @GetMapping("/search")
    public ResponseEntity<WaterDataDto> getData(
            @RequestParam String location
    ){
        return new
ResponseEntity<>(waterDataService.findLastEntry(location),
HttpStatus.OK);
    }

}
```

RabbitMQ configuration

```
@Configuration
public class RabbitMQConfig {

    @Value("${rabbitmq.water.queue.name}")
    private String queue;

    @Value("${rabbitmq.exchange.name}")
    private String exchange;

    @Value("${rabbitmq.water.routing.key}")
    private String routingKey;

    @Bean
    public Queue queue(){
        return  new Queue(queue);
    }

    @Bean
    public DirectExchange exchange(){
        return new DirectExchange(exchange);
    }
```

```
    @Bean
    public Binding binding(){
        return BindingBuilder.bind(queue())
                .to(exchange())
                .with(routingKey);
    }

    @Bean
    public MessageConverter converter(){
        return new Jackson2JsonMessageConverter();
    }

    @Bean
    public  AmqpTemplate amqpTemplate(ConnectionFactory
connectionFactory){
        RabbitTemplate rabbitTemplate = new
RabbitTemplate(connectionFactory);
        rabbitTemplate.setMessageConverter(converter());
        return  rabbitTemplate;
    }

}
```

Water User Model

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Document(collection = "user-to-notify")
public class UserToNotify {
    @Id
    private String id;
    private String email;
    private String coordinates;

    private Double lessDissolvedOxygen;
    private Double moreDissolvedOxygen;

    private Double lessOxidationReductionPotential;
    private Double moreOxidationReductionPotential;
```

```
    private Double lessPH;
    private Double morePH;

    private Double lessTurbidity;
    private Double moreTurbidity;

    private Double lessTotalDissolvedSolids;
    private Double moreTotalDissolvedSolids;

    private Double lessTemperature;
    private Double moreTemperature;
}
```

Water User Service

```
public UserToNotifyDto userSubToWaterData(UserToNotifyDto
userToNotifyDto){

userToNotifyRepository.save(mapUserToNotifyDtoToUserToNotify(userT
oNotifyDto));
        return userToNotifyDto;
    }

    public void deleteUser(String email){
        List<UserToNotify> userToNotifyList =
userToNotifyRepository.findByEmail(email);
        userToNotifyRepository.deleteAll(userToNotifyList);
    }


    public void getNotifyUserList(WaterData waterData){
        List<UserToNotify> userToNotifyList =
userToNotifyRepository.findByCoordinatesAndLessDissolvedOxygen(wat
erData.getCoordinates(), waterData.getDissolvedOxygen());

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAn
dMoreDissolvedOxygen(waterData.getCoordinates(),
waterData.getDissolvedOxygen()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAn
dLessOxidationReductionPotential(waterData.getCoordinates(),
waterData.getOxidationReductionPotential()));
```

```
userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAn
dMoreOxidationReductionPotential(waterData.getCoordinates(),
waterData.getOxidationReductionPotential()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAn
dLessPH(waterData.getCoordinates(), waterData.getPH()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAn
dMorePH(waterData.getCoordinates(), waterData.getPH()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAn
dLessTurbidity(waterData.getCoordinates(),
waterData.getTurbidity()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAn
dMoreTurbidity(waterData.getCoordinates(),
waterData.getTurbidity()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAn
dLessTotalDissolvedSolids(waterData.getCoordinates(),
waterData.getTotalDissolvedSolids()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAn
dMoreTotalDissolvedSolids(waterData.getCoordinates(),
waterData.getTotalDissolvedSolids()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAn
dLessTemperature(waterData.getCoordinates(),
waterData.getTemperature()));

userToNotifyList.addAll(userToNotifyRepository.findByCoordinatesAn
dMoreTemperature(waterData.getCoordinates(),
waterData.getTemperature()));

        List<String> emails = userToNotifyList.stream()
                .map(UserToNotify::getEmail).toList();

        WaterQueueNotifyUsersDto waterQueueNotifyUsersDto = new
WaterQueueNotifyUsersDto();

waterQueueNotifyUsersDto.setWaterData(mapWaterDataToWaterQueueData
```

```
Dto(waterData));
        waterQueueNotifyUsersDto.setEmails(emails);

        if(!emails.isEmpty()){

rabbitTemplate.convertAndSend(exchange,routingKey,waterQueueNotify
UsersDto);
        }
    }
```

Air User Controller

```
@RestController
@RequestMapping("/water")
@RequiredArgsConstructor
public class UserToNotifyController {

    private final UserToNotifyService userToNotifyService;

    @PostMapping("/notify")
    public ResponseEntity<UserToNotifyDto>
userSubToWaterData(@RequestBody UserToNotifyDto userToNotifyDto){
        return new
ResponseEntity<>(userToNotifyService.userSubToWaterData(userToNoti
fyDto), HttpStatus.OK);
    }

    @DeleteMapping ("/notify")
    public void userSubToWaterData(@RequestParam String email){
        userToNotifyService.deleteUser(email);
    }
}
```

Dockerfile

```
FROM openjdk:17
EXPOSE 8080:8080
ADD target/water-microservice.jar water-microservice.jar
ENTRYPOINT ["java","-jar","/water-microservice.jar"]
```

Dockerfile Compose

```yaml
version: '3.8'

services:
  water-microservice:
    image: water-microservice:latest
    ports:
      - "8080"
    environment:
      SPRING_DATA_MONGODB_HOST: mongodb
      SPRING_DATA_MONGODB_PORT: 27018
      SPRING_DATA_MONGODB_DATABASE: water
      SPRING_DATA_MONGODB_AUTHENTICATION: admin
      SPRING_RABBITMQ_HOST: rabbitmq
      SPRING_RABBITMQ_PORT: 5672
      SPRING_RABBITMQ_USERNAME: guest
      SPRING_RABBITMQ_PASSWORD: guest
      RABBITMQ_WATER_QUEUE_NAME: water.notification.queue
      RABBITMQ_EXCHANGE_NAME: notification.exchange
      RABBITMQ_WATER_ROUTING_KEY: water.queue.user.notification
      EUREKA_CLIENT_SERVICE_URL: http://eureka-server:8761/eureka/
    depends_on:
      - mongodb
    networks:
      - environment-pollution-network

  mongodb:
    image: mongo:latest
    ports:
      - "27018:27017"
    volumes:
      - type: bind
        source: Documents/pollution/water-db
        target: /pollution/water-db
    networks:
      - environment-pollution-network

networks:
  environment-pollution-network:
    external: true
    name: project_environment-pollution-network
```

K8s deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: water-microservice
spec:
  replicas: 2
  selector:
    matchLabels:
      app: water-microservice
  template:
    metadata:
      labels:
        app: water-microservice
    spec:
      containers:
        - name: water-microservice
          image: "themisvas/water-microservice:latest"
          ports:
            - containerPort: 8081
          env:
            - name: SPRING_DATA_MONGODB_HOST
              value: water-mongodb
            - name: SPRING_DATA_MONGODB_PORT
              value: "27018"
            - name: SPRING_DATA_MONGODB_DATABASE
              value: water
            - name: SPRING_DATA_MONGODB_AUTHENTICATION
              value: admin
            - name: SPRING_RABBITMQ_HOST
              value: rabbitmq
            - name: SPRING_RABBITMQ_PORT
              value: "5672"
            - name: SPRING_RABBITMQ_USERNAME
              valueFrom:
                secretKeyRef:
                  name: rabbitmq-secret
                  key: rabbitmq-username
            - name: SPRING_RABBITMQ_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: rabbitmq-secret
```

```
            key: rabbitmq-password
          - name: RABBITMQ_WATER_QUEUE_NAME
            value: water.notification.queue
          - name: RABBITMQ_EXCHANGE_NAME
            value: notification.exchange
          - name: RABBITMQ_WATER_ROUTING_KEY
            value: water.queue.user.notification
```

K8s Service

```
apiVersion: v1
kind: Service
metadata:
  name: water-microservice
spec:
  type: LoadBalancer
  selector:
    app: water-microservice
  ports:
    - port: 80
      targetPort: 8081
```

K8s DB Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: water-mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: water-mongodb
  template:
    metadata:
      labels:
        app: water-mongodb
    spec:
      containers:
        - name: water-mongodb
          image: mongo:latest
          ports:
            - containerPort: 27018
          volumeMounts:
            - name: water-mongodb-data
              mountPath: /data/water-db
  volumes:
    - name: water-mongodb-data
      persistentVolumeClaim:
        claimName: water-mongodb-data
```

K8s DB Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: water-mongodb
spec:
  selector:
    app: water-mongodb
  ports:
    - protocol: TCP
      port: 27018
      targetPort: 27018
```

K8s DB Persistence Volume

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: water-mongodb-data
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

5.3.3 Μικροϋπηρεσία Εδάφους

Παρακάτω παρατίθενται τα βασικά αρχεία για την μικροϋπηρεσία του εδάφους και η λογική υλοποίησης των κύριων σημείων της λειτουργικότητας του.

pom.xml

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-amqp</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId
>
            <version>4.1.0</version>
        </dependency>
    </dependencies>
```

application.properties

```
# General Config
spring.application.name=soil-microservice
server.port=0

# Mongo db config
spring.data.mongodb.authentication-database=${SPRING_DATA_MONGODB_
AUTHENTICATION}
spring.data.mongodb.database=${SPRING_DATA_MONGODB_DATABASE}
spring.data.mongodb.port=${SPRING_DATA_MONGODB_PORT}
spring.data.mongodb.host=${SPRING_DATA_MONGODB_HOST}

# Rabbit mq config
spring.rabbitmq.host=${SPRING_RABBITMQ_HOST}
spring.rabbitmq.port=${SPRING_RABBITMQ_PORT}
spring.rabbitmq.username=${SPRING_RABBITMQ_USERNAME}
spring.rabbitmq.password=${SPRING_RABBITMQ_PASSWORD}

rabbitmq.soil.queue.name=${RABBITMQ_SOIL_QUEUE_NAME}
rabbitmq.exchange.name=${RABBITMQ_EXCHANGE_NAME}
rabbitmq.soil.routing.key=${RABBITMQ_SOIL_ROUTING_KEY}

# Service discover config
eureka.client.service-url.defaultZone=${EUREKA_CLIENT_SERVICE_URL}
```

```
eureka.instance.prefer-ip-address=true
```

Soil Data Model

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Document(collection = "soil-data")
public class SoilData {

    private double soilTemperature;
    private double soilMoisture;
    private double electronicConductivity;
    private double volumetricWaterContent;
    private double salinity;
    private double totalSuspendedSolids;
    private double flowVolume;
    private double nitrate;
    @CreatedDate
    private Instant timestamp;
    private String coordinates;
}
```

Soil Data Service

```
public List<SoilDataDto> findByParameter(String coordinates,
Instant start, Instant end) {
        List<SoilData> res = null;

        res =
soilDataRepository.findByTimestampAndCoordinatesOOrderByTimestampA
irHumidityAsc(start, end, coordinates);

        return res.stream()
                .map(this::mapSoilDataTSoilDataDto)
                .collect(Collectors.toList());
    }

    public SoilDataDto findLastEntry(String location){
        return
mapSoilDataTSoilDataDto(soilDataRepository.findFirstByCoordinatesO
```

```
rderByTimestampDesc(location));
    }
```

Soil Controller

```
@RestController
@RequestMapping("/soil")
@RequiredArgsConstructor
public class SoilDataController {

    private final SoilDataService soilDataService;

    @GetMapping("/search-all")
    public ResponseEntity<List<SoilDataDto>> getData(
            @RequestParam String location,
            @RequestParam(required = false) @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) Instant start,
            @RequestParam(required = false) @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) Instant end
    ){

        return new
ResponseEntity<>(soilDataService.findByParameter(location,start,en
d), HttpStatus.OK);
    }

    @GetMapping("/search")
    public ResponseEntity<SoilDataDto> getData(
            @RequestParam String location
    ){
        return new
ResponseEntity<>(soilDataService.findLastEntry(location),
HttpStatus.OK);
    }

}
```

RabbitMQ configuration

```
@Configuration
public class RabbitMQConfig {

    @Value("${rabbitmq.soil.queue.name}")
```

```
    private String queue;

    @Value("${rabbitmq.exchange.name}")
    private String exchange;

    @Value("${rabbitmq.soil.routing.key}")
    private String routingKey;

    @Bean
    public Queue queue(){
        return  new Queue(queue);
    }

    @Bean
    public DirectExchange exchange(){
        return new DirectExchange(exchange);
    }

    @Bean
    public Binding binding(){
        return BindingBuilder.bind(queue())
                .to(exchange())
                .with(routingKey);
    }

    @Bean
    public MessageConverter converter(){
        return new Jackson2JsonMessageConverter();
    }

    @Bean
    public  AmqpTemplate amqpTemplate(ConnectionFactory
connectionFactory){
        RabbitTemplate rabbitTemplate = new
RabbitTemplate(connectionFactory);
        rabbitTemplate.setMessageConverter(converter());
        return  rabbitTemplate;
    }

}
```

Soil User Model

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Document(collection = "user-to-notify")
public class UserToNotify {
    @Id
    private String id;
    private String email;
    private String coordinates;

    private Double lessSoilTemperature;
    private Double moreSoilTemperature;

    private Double lessSoilMoisture;
    private Double moreSoilMoisture;

    private Double lessElectronicConductivity;
    private Double moreElectronicConductivity;

    private Double lessVolumetricWaterContent;
    private Double moreVolumetricWaterContent;

    private Double lessSalinity;
    private Double moreSalinity;

    private Double lessTotalSuspendedSolids;
    private Double moreTotalSuspendedSolids;

    private Double lessFlowVolume;
    private Double moreFlowVolume;

    private Double lessNitrate;
    private Double moreNitrate;
}
```

Soil User Service

```
public List<SoilDataDto> findByParameter(String coordinates,
Instant start, Instant end) {
        List<SoilData> res = null;
```

```
        res =
soilDataRepository.findByTimestampAndCoordinatesOOrderByTimestampA
irHumidityAsc(start, end, coordinates);

        return res.stream()
                .map(this::mapSoilDataTSoilDataDto)
                .collect(Collectors.toList());
    }

    public SoilDataDto findLastEntry(String location){
        return
mapSoilDataTSoilDataDto(soilDataRepository.findFirstByCoordinatesO
rderByTimestampDesc(location));
    }
```

Soil User Controller

```
@RestController
@RequestMapping("/soil")
@RequiredArgsConstructor
public class SoilDataController {

    private final SoilDataService soilDataService;

    @GetMapping("/search-all")
    public ResponseEntity<List<SoilDataDto>> getData(
            @RequestParam String location,
            @RequestParam(required = false) @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) Instant start,
            @RequestParam(required = false) @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) Instant end
    ){

        return new
ResponseEntity<>(soilDataService.findByParameter(location,start,end),
HttpStatus.OK);
    }

    @GetMapping("/search")
    public ResponseEntity<SoilDataDto> getData(
            @RequestParam String location
    ){
        return new
ResponseEntity<>(soilDataService.findLastEntry(location),
```

```
HttpStatus.OK);
    }

}
```

Dockerfile

```
FROM openjdk:17
EXPOSE 8080:8080
ADD target/soil-microservice.jar soil-microservice.jar
ENTRYPOINT ["java","-jar","/soil-microservice.jar"]
```

Dockerfile Compose

```
version: '3.8'

services:
  soil-microservice:
    image: soil-microservice:latest
    ports:
      - "8080"
    environment:
      SPRING_DATA_MONGODB_HOST: mongodb
      SPRING_DATA_MONGODB_PORT: 27019
      SPRING_DATA_MONGODB_DATABASE: soil
      SPRING_DATA_MONGODB_AUTHENTICATION: admin
      SPRING_RABBITMQ_HOST: rabbitmq
      SPRING_RABBITMQ_PORT: 5672
      SPRING_RABBITMQ_USERNAME: guest
      SPRING_RABBITMQ_PASSWORD: guest
      RABBITMQ_SOIL_QUEUE_NAME: soil.notification.queue
      RABBITMQ_EXCHANGE_NAME: notification.exchange
      RABBITMQ_SOIL_ROUTING_KEY: soil.queue.user.notification
      EUREKA_CLIENT_SERVICE_URL: http://eureka-server:8761/eureka/
    depends_on:
      - mongodb
    networks:
      - environment-pollution-network

  mongodb:
    image: mongo:latest
    ports:
      - "27019:27017"
    volumes:
      - type: bind
        source: Documents/pollution/soil-db
        target: /pollution/soil-db
```

```
    networks:
      - environment-pollution-network

networks:
  environment-pollution-network:
    external: true
    name: project_environment-pollution-network
```

K8s deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: soil-microservice
spec:
  replicas: 2
  selector:
    matchLabels:
      app: soil-microservice
  template:
    metadata:
      labels:
        app: soil-microservice
    spec:
      containers:
        - name: soil-microservice
          image: "themisvas/soil-microservice:latest"
          ports:
            - containerPort: 8082
          env:
            - name: SPRING_DATA_MONGODB_HOST
              value: soil-mongodb
            - name: SPRING_DATA_MONGODB_PORT
              value: "27019"
            - name: SPRING_DATA_MONGODB_DATABASE
              value: soil
            - name: SPRING_DATA_MONGODB_AUTHENTICATION
              value: admin
            - name: SPRING_RABBITMQ_HOST
              value: rabbitmq
            - name: SPRING_RABBITMQ_PORT
```

```
            value: "5672"
          - name: SPRING_RABBITMQ_USERNAME
            valueFrom:
              secretKeyRef:
                name: rabbitmq-secret
                key: rabbitmq-username
          - name: SPRING_RABBITMQ_PASSWORD
            valueFrom:
              secretKeyRef:
                name: rabbitmq-secret
                key: rabbitmq-password
          - name: RABBITMQ_SOIL_QUEUE_NAME
            value: soil.notification.queue
          - name: RABBITMQ_EXCHANGE_NAME
            value: notification.exchange
          - name: RABBITMQ_SOIL_ROUTING_KEY
            value: soil.queue.user.notification
```

K8s Service

```
apiVersion: v1
kind: Service
metadata:
  name: soil-microservice
spec:
  type: LoadBalancer
  selector:
    app: soil-microservice
  ports:
    - port: 80
      targetPort: 8082
```

K8s DB Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: soil-mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: soil-mongodb
```

```
  template:
    metadata:
      labels:
        app: soil-mongodb
    spec:
      containers:
        - name: soil-mongodb
          image: mongo:latest
          ports:
            - containerPort: 27019
          volumeMounts:
            - name: soil-mongodb-data
              mountPath: /data/soil-db
  volumes:
    - name: soil-mongodb-data
      persistentVolumeClaim:
        claimName: soil-mongodb-data
```

K8s DB Service

```
apiVersion: v1
kind: Service
metadata:
  name: soil-mongodb
spec:
  selector:
    app: soil-mongodb
  ports:
    - protocol: TCP
      port: 27019
      targetPort: 27019
```

K8s DB Persistence Volume

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: soil-mongodb-data
spec:
  accessModes:
    - ReadWriteMany
  resources:
```

```
    requests:
      storage: 1Gi
```

5.3.4 Μικροϋπηρεσία Ειδοποίησης Ενδιαφερόμενου

Η συγκεκριμένη υπηρεσία είναι υπεύθυνη για να ειδοποιεί τον κάθε ενδιαφερόμενο σε περίπτωση επιθυμητής τιμής που έχει ορίσει στην εφαρμογή. Συνδέεται άμεσα με τις μικροϋπηρεσίες του Αέρα, του Νερού και του Εδάφους μέσω ουρών, οι οποίες προωθούν ένα αίτημα ειδοποίησης σε περίπτωση που λάβουν τιμή από τους σένσορες που έχει ορίσει ο χρήστης στο σύστημα.

Παρακάτω παρατίθενται τα configuration αρχεία για το Docker και το K8s. Για την υλοποίηση του κώδικα βλέπε Παράρτημα 8.1 .

Dockerfile

```
FROM openjdk:17
EXPOSE 8080:8080
ADD target/notification-microservice.jar
notification-microservice.jar
ENTRYPOINT ["java","-jar","/notification-microservice.jar"]
```

Dockerfile Compose

```
version: '3.8'

services:
  notification-microservice:
    image: notification-microservice:latest
    ports:
      - "8080"
    environment:
      SPRING_RABBITMQ_HOST: rabbitmq
      SPRING_RABBITMQ_PORT: 5672
      SPRING_RABBITMQ_USERNAME: guest
      SPRING_RABBITMQ_PASSWORD: guest
      RABBITMQ_EXCHANGE_NAME: notification.exchange
      RABBITMQ_AIR_QUEUE_NAME: air.notification.queue
      RABBITMQ_AIR_ROUTING_KEY: air.queue.user.notification
      RABBITMQ_WATER_QUEUE_NAME: water.notification.queue
      RABBITMQ_WATER_ROUTING_KEY: water.queue.user.notification
      RABBITMQ_SOIL_QUEUE_NAME: soil.notification.queue
```

```
      RABBITMQ_SOIL_ROUTING_KEY: soil.queue.user.notification
      SPRING_MAIL_HOST: smtp.gmail.com
      SPRING_MAIL_PORT: 587
      SPRING_MAIL_USERNAME: th3m.them@gmail.com
      SPRING_MAIL_PASSWORD: ckonidsgtbylakhh
      EUREKA_CLIENT_SERVICE_URL: http://eureka-server:8761/eureka/

    networks:
      - environment-pollution-network

networks:
  environment-pollution-network:
    external: true
    name: project_environment-pollution-network
```

K8s deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: notification-microservice
spec:
  replicas: 1
  selector:
    matchLabels:
      app: notification-microservice
  template:
    metadata:
      labels:
        app: notification-microservice
    spec:
      containers:
        - name: notification-microservice
          image: "themisvas/notification-microservice:latest"
          ports:
            - containerPort: 8083
          env:
            - name: SPRING_RABBITMQ_HOST
              value: rabbitmq
            - name: SPRING_RABBITMQ_PORT
              value: "5672"
            - name: SPRING_RABBITMQ_USERNAME
              valueFrom:
                secretKeyRef:
```

```
        name: rabbitmq-secret
        key: rabbitmq-username
  - name: SPRING_RABBITMQ_PASSWORD
    valueFrom:
      secretKeyRef:
        name: rabbitmq-secret
        key: rabbitmq-password
  - name: SPRING_MAIL_HOST
    valueFrom:
      secretKeyRef:
        name: email-secret
        key: SPRING_MAIL_HOST
  - name: SPRING_MAIL_PORT
    valueFrom:
      secretKeyRef:
        name: email-secret
        key: SPRING_MAIL_PORT
  - name: SPRING_MAIL_USERNAME
    valueFrom:
      secretKeyRef:
        name: email-secret
        key: SPRING_MAIL_USERNAME
  - name: SPRING_MAIL_PASSWORD
    valueFrom:
      secretKeyRef:
        name: email-secret
        key: SPRING_MAIL_PASSWORD
  - name: RABBITMQ_EXCHANGE_NAME
    value: notification.exchange
  - name: RABBITMQ_AIR_QUEUE_NAME
    value: air.notification.queue
  - name: RABBITMQ_AIR_ROUTING_KEY
    value: air.queue.user.notification
  - name: RABBITMQ_WATER_QUEUE_NAME
    value: water.notification.queue
  - name: RABBITMQ_WATER_ROUTING_KEY
    value: water.queue.user.notification
  - name: RABBITMQ_SOIL_QUEUE_NAME
    value: soil.notification.queue
  - name: RABBITMQ_SOIL_ROUTING_KEY
    value: soil.queue.user.notification
```

K8s Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: notification-microservice
spec:
  type: LoadBalancer
  selector:
    app: notification-microservice
  ports:
    - port: 80
      targetPort: 8083
```

K8s Secret

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: email-secret
type: Opaque
data:
  SPRING_MAIL_HOST: c210cC5nbWFpbC5jb20=
  SPRING_MAIL_PORT: NTg3
  SPRING_MAIL_USERNAME: dGgzbS50aGVtQGdtYWlsLmNvbQ==
  SPRING_MAIL_PASSWORD: Y2tvbmlkc2d0YnlsYWtoaA==
```

5.3.5 Μικροϋπηρεσία Ανάλυσης Δεδομένων

Παρακάτω παρατίθενται τα βασικά αρχεία για την μικροϋπηρεσία της ανάλυσης δεδομένων και η λογική υλοποίησης των κύριων σημείων της λειτουργικότητας του.

pom.xml

```xml
<dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
```

```
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-math3</artifactId>
            <version>3.6.1</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
            <version>4.1.0</version>
        </dependency>
    </dependencies>
```

application.properties

```
# General Config
spring.application.name=analytics-microservice
server.port=0

# Service discover config
eureka.client.service-url.defaultZone=${EUREKA_CLIENT_SERVICE_URL}
eureka.instance.prefer-ip-address=true
```

Statistics Calculator

```
@Service
public class StatsCalculator {


    public Double calculateMean(List<Double> numbers){
        if (numbers.isEmpty()) {
            throw new IllegalArgumentException("List is empty");
```

```
        }

        DescriptiveStatistics statistics = new DescriptiveStatistics();
        numbers.forEach(statistics::addValue);

        return statistics.getMean();
    }

    public Double calculateMedian(List<Double> numbers) {
        if (numbers.isEmpty()) {
            throw new IllegalArgumentException("List is empty");
        }

        DescriptiveStatistics stats = new DescriptiveStatistics();
        numbers.forEach(stats::addValue);

        return stats.getPercentile(50);
    }

    public Double calculateRange(List<Double> numbers) {
        if (numbers.isEmpty()) {
            throw new IllegalArgumentException("List is empty");
        }

        DescriptiveStatistics stats = new DescriptiveStatistics();
        numbers.forEach(stats::addValue);

        return stats.getMax() - stats.getMin();
    }

    public Double calculateStandardDeviation(List<Double> numbers) {
        if (numbers.isEmpty()) {
            throw new IllegalArgumentException("List is empty");
        }

        DescriptiveStatistics stats = new DescriptiveStatistics();
        numbers.forEach(stats::addValue);

        return stats.getStandardDeviation();
    }
```

Η παραπάνω κλάση υπολογίζει βασικά στατιστικά όπως το μέσο όρο, τη διάμεσο, την τυπική απόκλιση κλπ. Για τον υπολογισμό των δεδομένων υπάρχουν προς τον έξω κόσμο μέθοδοι που ακούνε με HTTP:

Air Stats Controller

```
@RestController
@RequestMapping("/analytics")
@RequiredArgsConstructor
public class BasicAirDataStatisticalResultController {

    private final BasicAirDataStatisticalResultService
dataStatisticalResultService;

    @PostMapping("/basic-air")
    public ResponseEntity<BasicAirDataStatisticalResultDto>
calculateBasicStatistics(@RequestBody List<AirDataDto> airDataDtoList){

        return new
ResponseEntity<>(dataStatisticalResultService.calculateBasicAirStats(air
DataDtoList), HttpStatus.OK);
    }

}
```

Water Stats Controller

```
@RestController
@RequestMapping("/analytics")
@RequiredArgsConstructor
public class BasicWaterDataStatisticalResultController {

    private final BasicWaterDataStatisticalResultService
dataStatisticalResultService;

    @PostMapping("/basic-water")
    public ResponseEntity<BasicWaterDataStatisticalResultDto>
calculateBasicStatistics(@RequestBody List<WaterDataDto>
waterDataDtoList){

        return new
ResponseEntity<>(dataStatisticalResultService.calculateBasicWaterStats(w
aterDataDtoList), HttpStatus.OK);
    }


}
```

Soil Stats Controller

```
@RestController
@RequestMapping("/analytics")
```

```
@RequiredArgsConstructor
public class BasicSoilDataStatisticalResultController {

    private final BasicSoilDataStatisticalResultService
dataStatisticalResultService;

    @PostMapping("/basic-soil")
    public ResponseEntity<BasicSoilDataStatisticalResultDto>
calculateBasicStatistics(@RequestBody List<SoilDataDto>
soilDataDtoList){

        return new
ResponseEntity<>(dataStatisticalResultService.calculateBasicSoilSt
ats(soilDataDtoList), HttpStatus.OK);
    }


}
```

Dockerfile

```
FROM openjdk:17
EXPOSE 8080:8080
ADD target/analysis-microservice.jar analysis-microservice.jar
ENTRYPOINT ["java","-jar","/analysis-microservice.jar"]
```

Dockerfile Compose

```
version: '3.8'

services:
  analysis-microservice:
    image: analysis-microservice:latest
    ports:
      - "8080"
    environment:
      EUREKA_CLIENT_SERVICE_URL: http://eureka-server:8761/eureka/
    networks:
      - environment-pollution-network

networks:
  environment-pollution-network:
    external: true
    name: project_environment-pollution-network
```

K8s deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: analysis-microservice
spec:
  replicas: 2
  selector:
    matchLabels:
      app: analysis-microservice
  template:
    metadata:
      labels:
        app: analysis-microservice
    spec:
      containers:
        - name: analysis-microservice
          image: "themisvas/analysis-microservice:latest"
          ports:
            - containerPort: 8084
```

K8s Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: analysis-microservice
spec:
  type: LoadBalancer
  selector:
    app: analysis-microservice
  ports:
    - port: 80
      targetPort: 8084
```

5.3.6 Μικροϋπηρεσία Api-Gateway

Η συγκεκριμένη υπηρεσία [22] λειτουργεί ως ένα σημείο εισόδου για πολλαπλές διεπαφές προγραμματισμού εφαρμογών παρέχοντας διάφορες λειτουργίες, συμπεριλαμβανομένων:

- **Routing:** Καθοδήγηση των αιτημάτων του πελάτη στην κατάλληλη υπηρεσία backend βάσει του του περιεχομένου του αιτήματος.

- **Authentication and Authorization:** Επιβολή μέτρων ασφαλείας, όπως η επαλήθευση των ταυτοτήτων των χρηστών ή ο έλεγχος πρόσβασης.

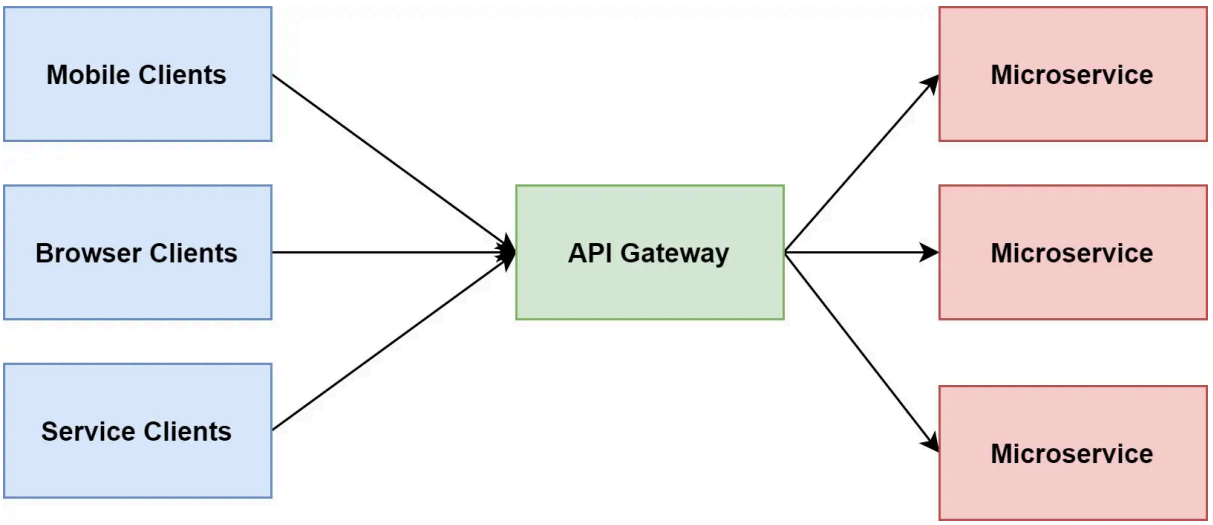
- **Request Filtering:** Προσαρμογή των αιτημάτων και των απαντήσεων για ασφάλεια του backend από κακόβουλες κλήσεις.

- **Load Balancing:** Διανομή εισερχόμενων αιτημάτων σε πολλαπλές περιπτώσεις των υπηρεσιών backend για τη βελτιστοποίηση της απόδοσης και της διαθεσιμότητας.

- **Monitoring and Analytics:** Συλλογή δεδομένων σχετικά με τη χρήση του API, τις μετρήσεις απόδοσης και τα ποσοστά σφαλμάτων για τη διευκόλυνση της παρακολούθησης.

- **Rate Limiting:** Έλεγχος του ρυθμού με τον οποίο οι πελάτες μπορούν να έχουν πρόσβαση στα APIs προκειμένου να αποτραπεί η κατάχρηση ή η υπερφόρτωση του backend.

[Εικόνα 8]: [Λειτουργία Api-Gateway]



pom.xml

```
 <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-gateway</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId
>
            <version>4.1.0</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

application.properties

```
# General Config
spring.application.name=api-gateway
```

```
server.port=8086

eureka.client.service-url.defaultZone=http://eureka-server:8761/eu
reka/
eureka.instance.prefer-ip-address=true

spring.cloud.gateway.routes[0].id=air-microservice
spring.cloud.gateway.routes[0].uri=lb://air-microservice
spring.cloud.gateway.routes[0].predicates[0].=Path=air/**
spring.cloud.gateway.routes[0].filters[0]=SetPath=/air/**

spring.cloud.gateway.routes[1].id=water-microservice
spring.cloud.gateway.routes[1].uri=lb://water-microservice
spring.cloud.gateway.routes[1].predicates[0].=Path=water/**
spring.cloud.gateway.routes[1].filters[0]=SetPath=/water/**

spring.cloud.gateway.routes[2].id=soil-microservice
spring.cloud.gateway.routes[2].uri=lb://soil-microservice
spring.cloud.gateway.routes[2].predicates[0].=Path=soil/**
spring.cloud.gateway.routes[2].filters[0]=SetPath=/soil/**

spring.cloud.gateway.routes[3].id=analytics-microservice
spring.cloud.gateway.routes[3].uri=lb://analytics-microservice
spring.cloud.gateway.routes[3].predicates[0].=Path=/analytics**
spring.cloud.gateway.routes[3].filters[0]=SetPath=/analytics/**

# Enable global CORS configuration
spring.cloud.gateway.globalcors.cors-configurations.[/**].allowed-
origins=http://localhost:5173
spring.cloud.gateway.globalcors.cors-configurations.[/**].allowed-
methods=GET, POST, PUT, DELETE, OPTIONS
spring.cloud.gateway.globalcors.cors-configurations.[/**].allowed-
headers=*
spring.cloud.gateway.globalcors.cors-configurations.[/**].allow-cr
edentials=true

spring.cloud.gateway.discovery.locator.enabled=true
spring.cloud.gateway.discovery.locator.lower-case-service-id=true
```

Στο παραπάνω αρχείο ρυθμίζουμε το api-gateway μας να δέχεται κλήσεις μόνο από το front end της εφαρμογής μας και όχι από εξωτερικές πηγές και ορίζουμε κάποια φίλτρα για να προωθούνται οι κλήσεις στα σωστά microservices.

Dockerfile

```
FROM openjdk:17
EXPOSE 8086
ADD target/api-gateway.jar api-gateway.jar
ENTRYPOINT ["java","-jar","/api-gateway.jar"]
```

Dockerfile Compose

```
version: '3.8'

services:
  api-gateway:
    image: api-gateway:latest
    ports:
      - "8086:8086"
    environment:
      EUREKA_CLIENT_SERVICE_URL: http://eureka-server:8761/eureka/
    networks:
      - environment-pollution-network

networks:
  environment-pollution-network:
    external: true
    name: project_environment-pollution-network
```

K8s deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api-gateway
spec:
  replicas: 1
  selector:
    matchLabels:
      app: api-gateway
  template:
    metadata:
```

```
    labels:
      app: api-gateway
  spec:
    containers:
      - name: api-gateway
        image: "themisvas/api-gateway:latest"
        ports:
          - containerPort: 8086
```

K8s Service

```
apiVersion: v1
kind: Service
metadata:
  name: api-gateway
spec:
  type: LoadBalancer
  selector:
    app: api-gateway
  ports:
    - port: 80
      targetPort: 8086
```

5.3.7 Μικροϋπηρεσία Service Discovery

Η ανίχνευση υπηρεσιών σε ένα περιβάλλον μικροϋπηρεσιών αναφέρεται στη διαδικασία εντοπισμού και επικοινωνίας μεταξύ των διαφόρων υπηρεσιών που αποτελούν το συνολικό σύστημα. Κάθε υπηρεσία μπορεί να έχει διαφορετική τοποθεσία, διεύθυνση IP ή άλλα δυναμικά χαρακτηριστικά που αλλάζουν κατά την εκτέλεση, και η ανίχνευση υπηρεσιών βοηθάει στο να γίνεται αυτόματα ο εντοπισμός και η επικοινωνία με αυτές. Μπορεί να επιτευχθεί με τη χρήση ειδικών εργαλείων ή πρωτοκόλλων. Στην παρούσα εργασία θα αναπτυχθούν δύο τρόποι για service discovery. Ο πρώτος θα είναι σαν μία κανονική μικροϋπηρεσία με όνομα eureka-server, όπως φαίνεται και σαν dependency στο application.properties των προηγούμενων μικροϋπηρεσιών και χειρίζεται την συγκεκριμένη λειτουργία. Ο δεύτερος είναι ο αυτόματος τρόπος του K8s όταν βρίσκονται στο ίδιο κατανεμημένο σύστημα, χωρίς την ανάγκη για την μικροϋπηρεσία του eureka server. Στην

περίπτωση αυτή, απλά θα αφαιρέσουμε το dependency σε κάθε υπηρεσία που το χρησιμοποιεί.

pom.xml

```
<dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId
>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

application.properties

```
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

Dockerfile

```
FROM openjdk:17
EXPOSE 8761
ADD target/eureka-server.jar eureka-server.jar
ENTRYPOINT ["java","-jar","/eureka-server.jar"]
```

Dockerfile Compose

```
version: '3.8'

services:
  eureka-server:
    image: eureka-server:latest
    ports:
      - "8761:8761"
    networks:
```

```
      - environment-pollution-network

  rabbitmq:
    image: rabbitmq:3.12.11-management
    ports:
      - "5672:5672"
    networks:
      - environment-pollution-network

networks:
  environment-pollution-network:
    external: true
    name: project_environment-pollution-network
```

5.3.8 Διεπαφή Χρήστη

Για την διεπαφή χρήστη υλοποιήθηκαν components που χρησιμεύουν για την αναζήτηση τιμών που επιθυμεί ο ενδιαφερόμενος, την προβολή των δεδομένων, φόρμες για εισαγωγή στοιχείων ειδοποίησης και επεξήγηση των όλων των μετρούμενων δεδομένων.

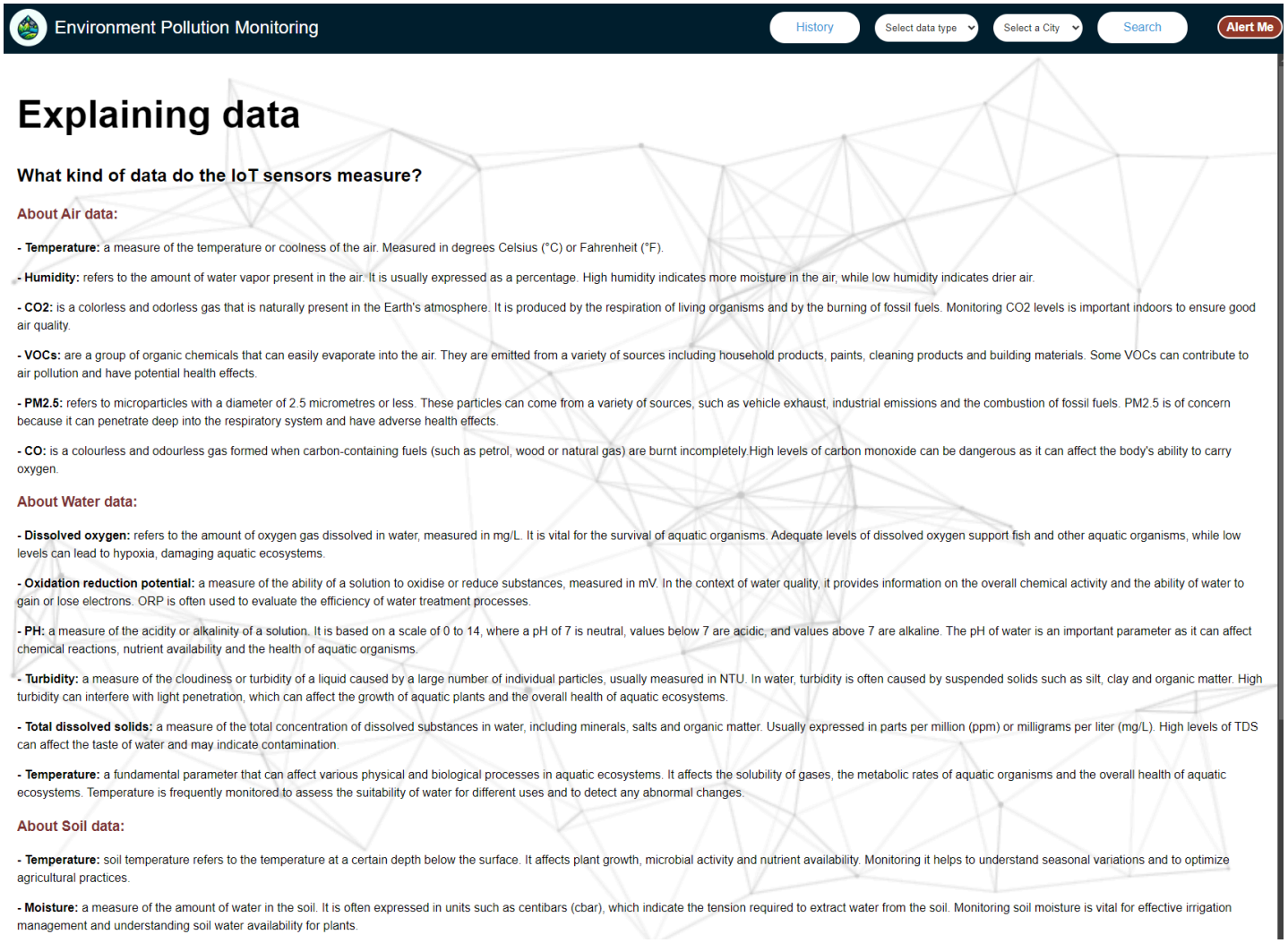
Για την υλοποίηση του κώδικα βλέπε Παράρτημα 8.2 .

5.4 Παρουσίαση Εφαρμογής

5.4.1 Αρχική Σελίδα

Στην αρχική σελίδα υπάρχει μία μικρή επεξήγηση για την λειτουργία της εφαρμογής, δύο εικόνες, μία μπάρα αναζήτησης στο πάνω μέρος και ένας σύνδεσμος.

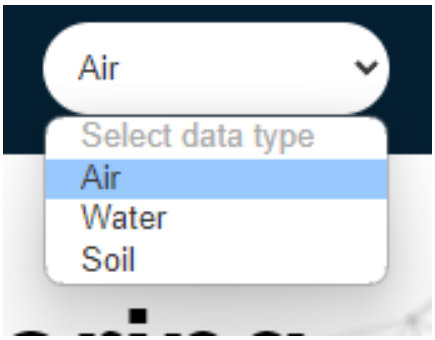
Πατώντας τον σύνδεσμο για επεξήγηση των δεδομένων που μετράει η εφαρμογή μεταφερόμαστε παρακάτω:

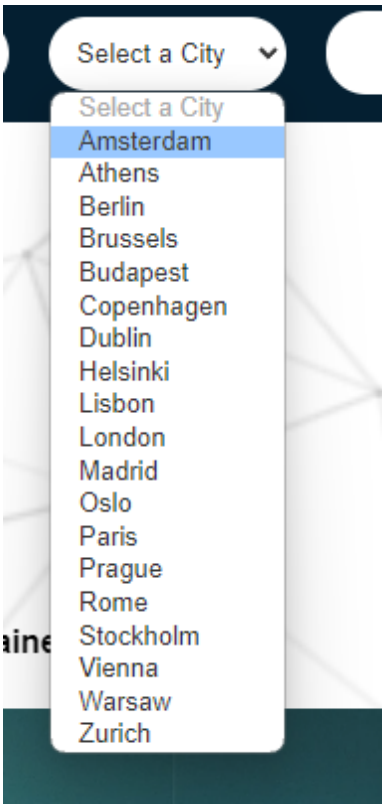


Στην παραπάνω σελίδα επεξηγούνται αναλυτικά όλοι οι τύποι δεδομένων που μετριούνται ανά κατηγορία, ώστε οι χρήστες να αποκτήσουν μία σφαιρική εικόνα για την ποιότητα των μετρήσεων.

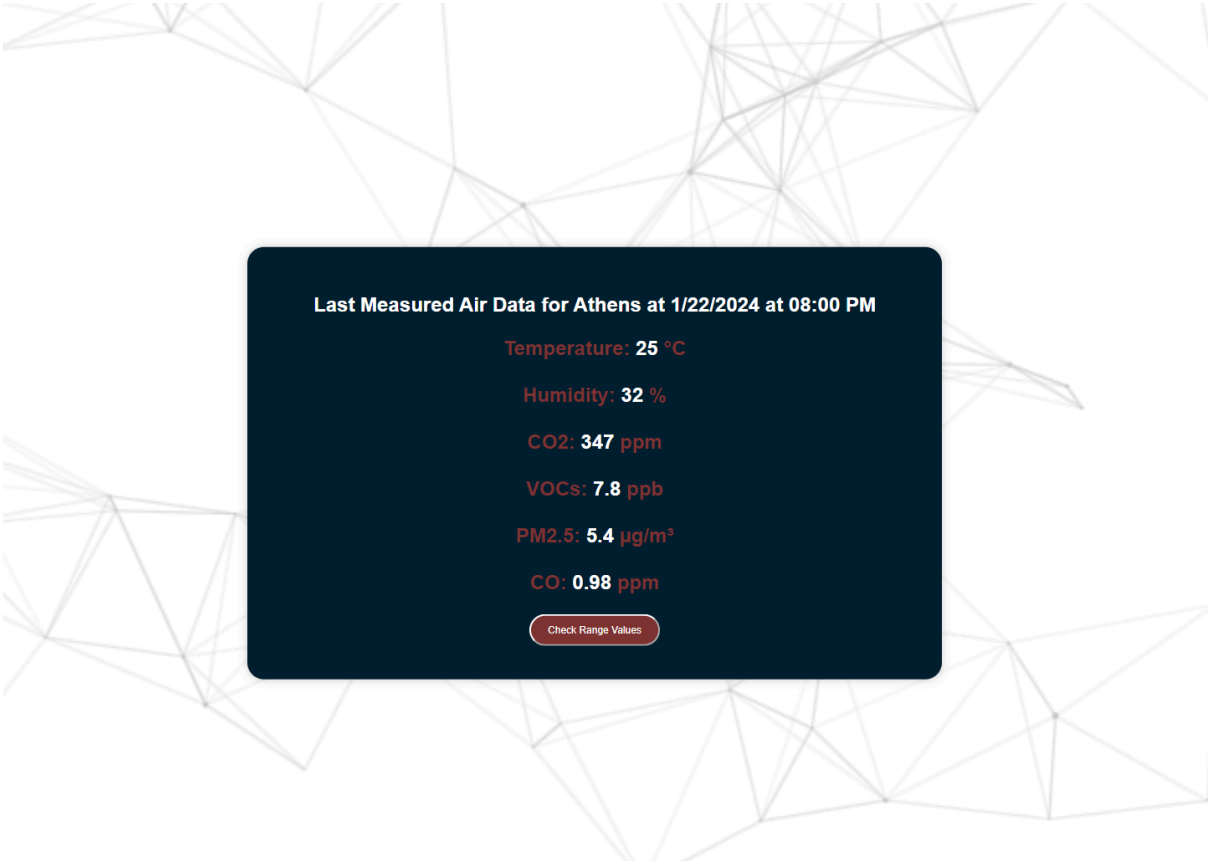
5.4.2 Αναζήτηση ζωντανών δεδομένων

Παρατηρώντας τη μπάρα αναζήτησης βλέπουμε κάποια φίλτρα για επιλογή αναζητήσεων. Αρχικά, η προκαθορισμένη λειτουργία είναι η αναζήτηση ζωντανών δεδομένων επιλέγοντας απλά τον επιθυμητό τύπο δεδομένων και την περιοχή.





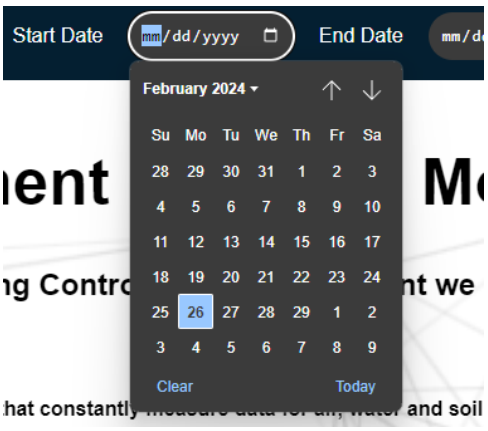
Συνεχίζοντας, πατάμε το κουμπί Search και αυτόματα μεταφερόμαστε σε μία άλλη σελίδα και εμφανίζονται σε μία λίστα οι μετρήσεις της αναζήτησής μας. Κάτω από τις μετρήσεις υπάρχει επιπλέον ένα κουμπί με την ονομασία Check Range Values.



Πατώντας το κουμπί αυτό εμφανίζεται ένα πινακάκι με τα πεδία φυσιολογικών τιμών ανά δεδομένο, ώστε να καταλαβαίνει ο χρήστης σε τι επίπεδο βρίσκονται οι μετρήσεις.

5.4.3 Αναζήτηση παρελθοντικών δεδομένων

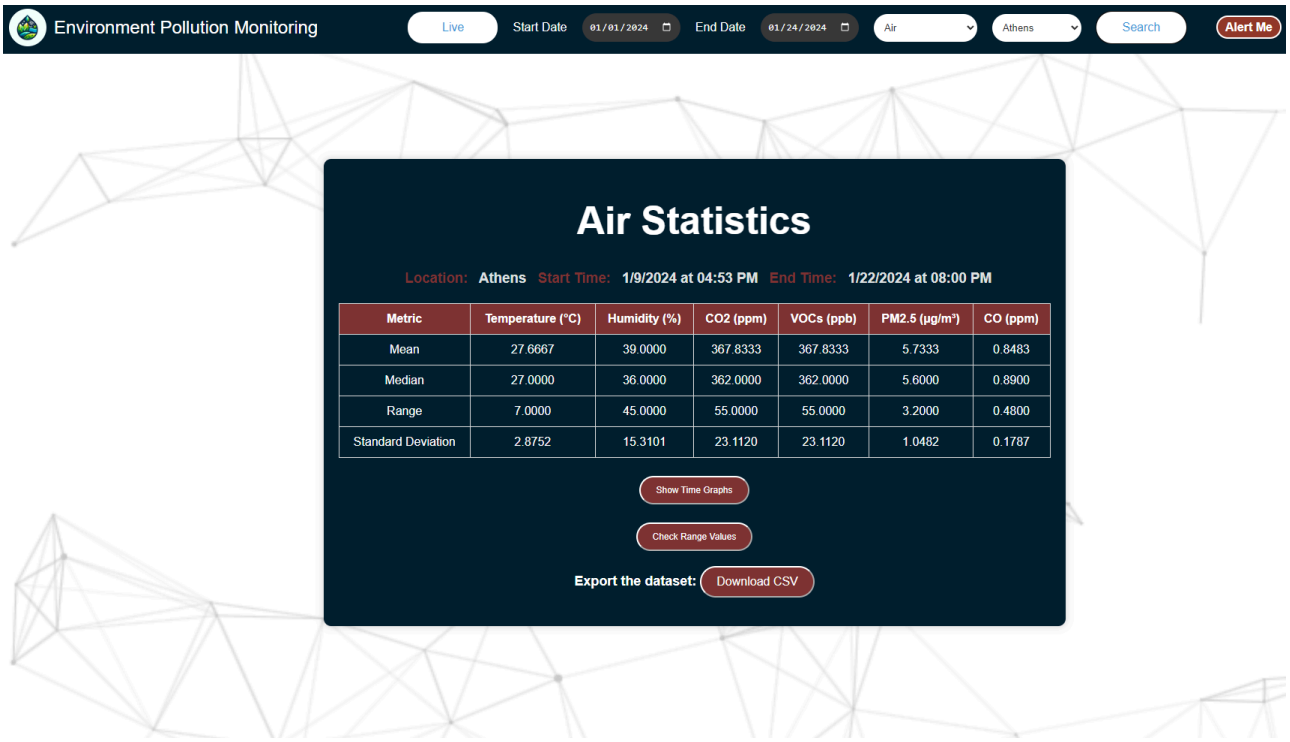
Για να αναζητήσει ο χρήστης παρελθοντικά δεδομένα, πατάει το κουμπί History στη μπάρα αναζήτησης για να εμφανιστούν τα φίλτρα για τις επιθυμητές ημερομηνίες.



Έπειτα, πατώντας πάλι το κουμπί Search συνεχίζουμε στην ανάλυση δεδομένων.

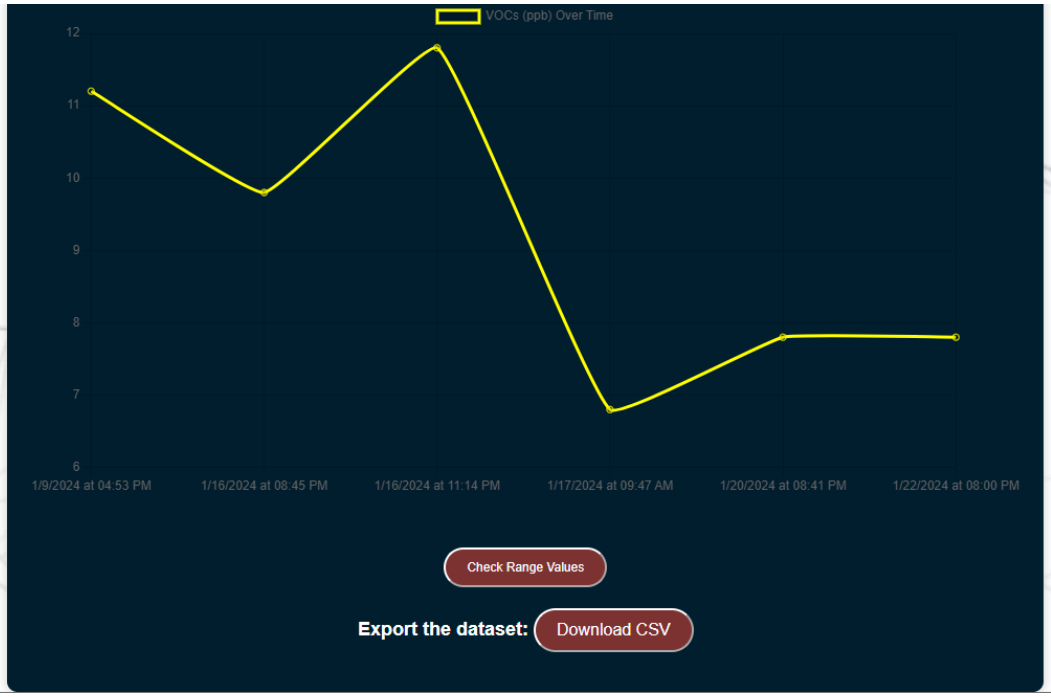
5.4.4 Ανάλυση Δεδομένων

Στην σελίδα ανάλυσης δεδομένων εμφανίζεται πρώτα ένα πινακάκι με στατιστικά για τις ημερομηνίες που επιλέξαμε στα φίλτρα, όπως ο μέσος όρος, η διάμεσος, η τυπική απόκλιση και άλλα ανά μέτρηση. Επιπροσθέτως, υπάρχουν τρία κουμπιά, το πρώτο Check Range Values όπως αναφέρθηκε προηγουμένως, το Show Time Graphs και το Download CSV.

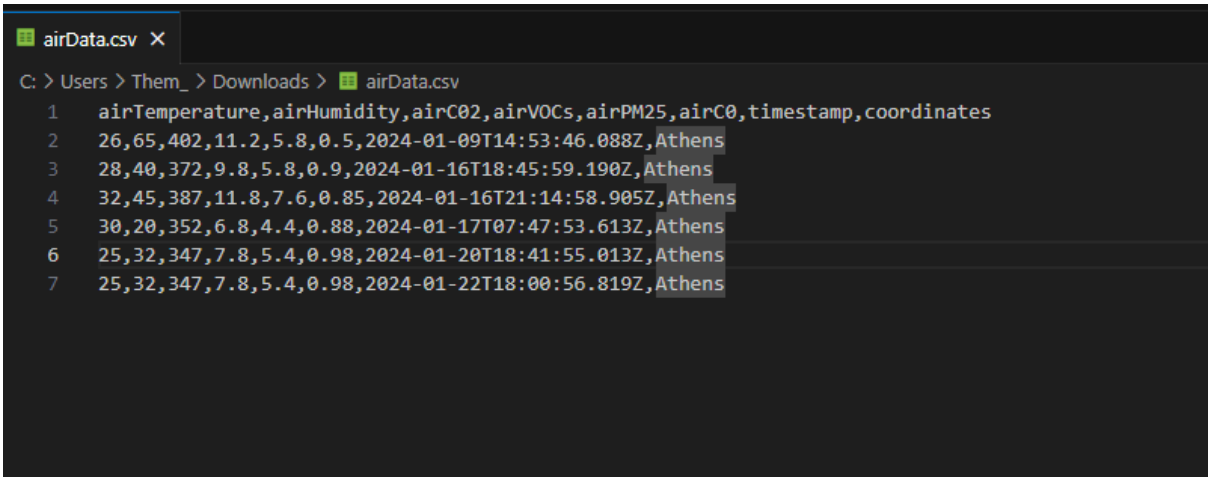


Πατώντας το δεύτερο κουμπί Show Time Graphs θα εμφανιστούν γραφικές παραστάσεις που απεικονίζουν τη μεταβολή των τιμών κάθε μέτρησης στο χρόνο που ορίσαμε κατά την αναζήτησή μας.


Close Time Graphs
Temperature (°C) Over Time
32
31
30
29
28
27
26
25
1/9/2024 at 04:53 PM
1/16/2024 at 08:45 PM
1/16/2024 at 11:14 PM
1/17/2024 at 09:47 AM
1/20/2024 at 08:41 PM
1/22/2024 at 08:00 PM
Humidity (%) Over Time
65
60
55
50
45
40
35
30
25
20
1/9/2024 at 04:53 PM
1/16/2024 at 08:45 PM
1/16/2024 at 11:14 PM
1/17/2024 at 09:47 AM
1/20/2024 at 08:41 PM
1/22/2024 at 08:00 PM
CO2 (ppm) Over Time
410
400

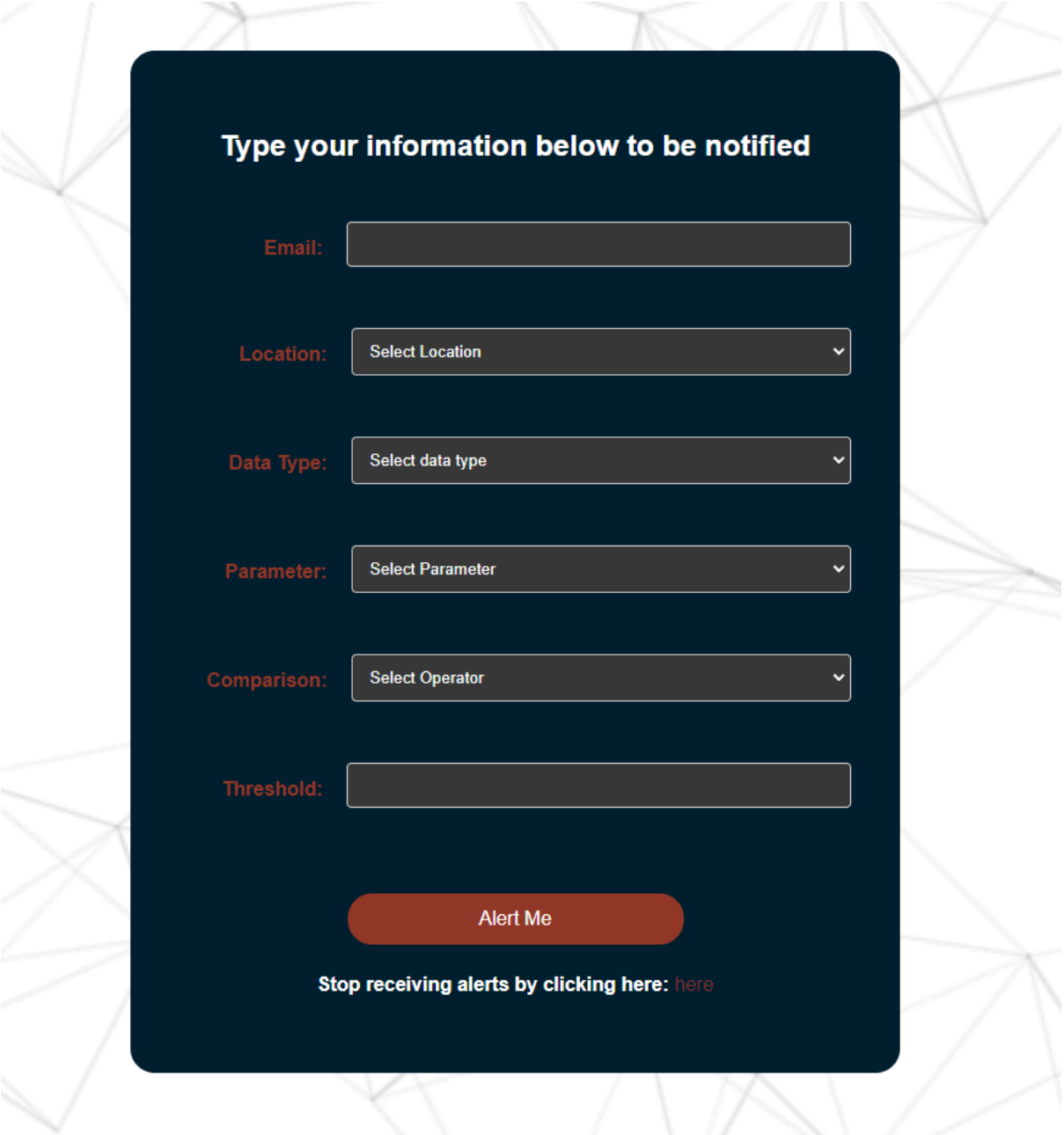

VOCs (ppb) Over Time
12
11
10
9
8
7
6
1/9/2024 at 04:53 PM
1/16/2024 at 08:45 PM
1/16/2024 at 11:14 PM
1/17/2024 at 09:47 AM
1/20/2024 at 08:41 PM
1/22/2024 at 08:00 PM
Check Range Values
Export the dataset:
Download CSV

Τέλος, πατώντας το τελευταίο κουμπί Download CSV κατεβαίνει ένα αρχείο τοπικά στον υπολογιστή μας μορφής csv, όπου περιέχει όλες τις τιμές που μετρήθηκαν στο διάστημα αναζήτησής μας σε περίπτωση που κάποιος θέλει να προβεί σε περαιτέρω ανάλυση των δεδομένων αυτών.



5.4.5 Ειδοποίηση Ενδιαφερόμενου

Η φόρμα για ειδοποίηση του ενδιαφερόμενου βρίσκεται πατώντας στο κουμπί alert στη μπάρα αναζήτησης, όπου μας μεταφέρει στην παρακάτω σελίδα και ο χρήστης έχει τη δυνατότητα να επιλέξει την επιθυμητή κατηγορία, περιοχή και τιμή που θέλει να λαμβάνει ειδοποιήσεις.



Σε περίπτωση ειδοποίησης το email που στέλνεται στον ενδιαφερόμενο έχει την παρακάτω μορφή:

Air Quality Data

Parameter	Value	Unit
Temperature	26.0	°C
Humidity	65.0	%
CO2	402.0	ppm
VOCs	11.2	ppb
PM2.5	5.8	µg/m³
CO	0.5	ppm
Timestamp	2024-01-09T14:53:46.088680500Z	
Location	Athens	

6. Συμπεράσματα

Συμπερασματικά, η παρακολούθηση του περιβάλλοντος αναδεικνύεται ως ουσιώδης για τη διατήρηση της οικολογικής ισορροπίας και την προστασία της φύσης. Ωστόσο, η πρόοδος στον τομέα της τεχνολογίας έχει ανοίξει νέους ορίζοντες μέσω της ενσωμάτωσης του Internet of Things (IoT) στα συστήματα παρακολούθησης. Αυτό επιτρέπει την αυτόματη συλλογή δεδομένων από αισθητήρες και την παρακολούθηση του περιβάλλοντος σε πραγματικό χρόνο.

Ο σχεδιασμός και η χρήση σύγχρονων τεχνολογιών αποτελούν κρίσιμους παράγοντες για τη δημιουργία ευέλικτων και αποδοτικών συστημάτων παρακολούθησης. Ειδικότερα, η εικονικοποίηση με περιέκτες και η αρχιτεκτονική μικροϋπηρεσιών αναδεικνύονται ως ορόσημα σε αυτόν τον τομέα, προσφέροντας απαραίτητη ευελιξία και απόδοση.

Εξίσου σημαντική είναι και η συνεχής εκπαίδευση και ενημέρωση σε νέες τεχνολογίες, η οποία αποτελεί θεμέλιο πυλώνα για την επιτυχή υλοποίηση προηγμένων έργων παρακολούθησης. Η διαρκής εξέλιξη του τομέα απαιτεί σταθερή ενημέρωση και προετοιμασία.

Τέλος, η εφαρμογή των παραπάνω σε κατανεμημένα συστήματα επιτρέπει την αποτελεσματική λειτουργία και διαχείριση των πόρων σε ευρύτερη κλίμακα. Η ολοκληρωμένη προσέγγιση αυτών των παραγόντων αναδεικνύεται ως κρίσιμη για τη διατήρηση της βιωσιμότητας και την προστασία του περιβάλλοντος για τις επόμενες γενιές.

7. Βιβλιογραφία

[1] Silvia Liberata Ullo, G. R. Sinha 2, 2020: Advances in Smart Environment Monitoring Systems Using IoT and Sensors
https://www.mdpi.com/1424-8220/20/11/3113
[2] IoT solutions for monitoring air quality
https://www.paessler.com/iot/air-quality-monitoring
[3] IoT in Water Management with Smart Water Quality Monitors
https://blues.com/use-cases/iot-in-water-management/
[4] Soil Monitoring with IoT – Smart Agriculture
https://www.manxtechgroup.com/soil-monitoring-with-iot-smart-agriculture/
[5] IoT-Based Environmental Monitoring: How IoT Technology Can Help Businesses Go Green
https://caburntelecom.com/iot-environmental-monitoring/
[6] What is Monolithic Architecture?
https://www.integrate.io/glossary/what-is-monolithic-architecture/
[7] Microservices vs. monolithic architecture
https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith
[8] Communication in a microservice architecture
https://learn.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/communication-in-microservice-architecture
[9] What is Virtualization?
https://aws.amazon.com/what-is/virtualization/
[10] What is container orchestration?
https://www.vmware.com/topics/glossary/content/container-orchestration.html
[11] What is Java?
https://aws.amazon.com/what-is/java/
[12] What is Java Spring Boot?
https://www.ibm.com/topics/java-spring-boot
[13] What is Maven: Here's What You Need to Know
https://www.simplilearn.com/tutorials/maven-tutorial/what-is-maven
[14] What is MongoDB?
https://aws.amazon.com/documentdb/what-is-mongodb/

[15] Make your apps scalable and resilient with enterprise RabbitMQ
https://tanzu.vmware.com/rabbitmq
[16] What is JavaScript?
https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
[17] Getting started with React
https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started
[18] Docker tutorial for beginners
https://docker-curriculum.com/
[19] What is Kubernetes?
https://www.armosec.io/glossary/kubernetes/
[20] IntelliJ IDEA
https://www.techopedia.com/definition/7755/intellij-idea
[21] Introduction to Visual Studio Code
https://www.educba.com/what-is-visual-studio-code/
[22] What Is an API Gateway?
https://www.nginx.com/learn/api-gateway/
[23] Service Discovery in Microservices
https://www.baeldung.com/cs/service-discovery-microservices

8. Παράρτημα

8.1 Υλοποίηση Μικροϋπηρεσίας Ειδοποίησης

Παρακάτω βρίσκεται ο κώδικας της μικροϋπηρεσίας ειδοποίησης χρήστη.

pom.xml

```
<dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-amqp</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.amqp</groupId>
            <artifactId>spring-rabbit-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-mail</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId
>
            <version>4.1.0</version>
```

```
        </dependency>
    </dependencies>
```

application.properties

```
# General Config
spring.application.name=notification-microservice
server.port=0

# Rabbit mq config
spring.rabbitmq.host=${SPRING_RABBITMQ_HOST}
spring.rabbitmq.port=${SPRING_RABBITMQ_PORT}
spring.rabbitmq.username=${SPRING_RABBITMQ_USERNAME}
spring.rabbitmq.password=${SPRING_RABBITMQ_PASSWORD}

rabbitmq.exchange.name=${RABBITMQ_EXCHANGE_NAME}
rabbitmq.air.queue.name=${RABBITMQ_AIR_QUEUE_NAME}
rabbitmq.air.routing.key=${RABBITMQ_AIR_ROUTING_KEY}
rabbitmq.water.queue.name=${RABBITMQ_WATER_QUEUE_NAME}
rabbitmq.water.routing.key=${RABBITMQ_WATER_ROUTING_KEY}
rabbitmq.soil.queue.name=${RABBITMQ_WATER_QUEUE_NAME}
rabbitmq.soil.routing.key=${RABBITMQ_WATER_QUEUE_NAME}

# Email SMTP config
spring.mail.host=${SPRING_MAIL_HOST}
spring.mail.port=${SPRING_MAIL_PORT}
spring.mail.username=${SPRING_MAIL_USERNAME}
spring.mail.password=${SPRING_MAIL_PASSWORD}
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.starttls.required=true

# Service discover config
eureka.client.service-url.defaultZone=${EUREKA_CLIENT_SERVICE_URL}
eureka.instance.prefer-ip-address=true
```

RabbitMQ configuration

```
@Bean
    public DirectExchange exchange(){
        return new DirectExchange(exchange);
    }

    @Bean
```

```
public Queue airQueue(){
    return  new Queue(airQueue);
}

@Bean
public Binding airBinding(){
    return BindingBuilder.bind(airQueue())
            .to(exchange())
            .with(airRoutingKey);
}

@Bean
public Queue waterQueue(){
    return  new Queue(waterQueue);
}

@Bean
public Binding waterBinding(){
    return BindingBuilder.bind(waterQueue())
            .to(exchange())
            .with(waterRoutingKey);
}

@Bean
public Queue soilQueue(){
    return  new Queue(soilQueue);
}

@Bean
public Binding soilBinding(){
    return BindingBuilder.bind(waterQueue())
            .to(exchange())
            .with(soilRoutingKey);
}

@Bean
public MessageConverter converter(){
    return new Jackson2JsonMessageConverter();
}

@Bean
public AmqpTemplate amqpTemplate(ConnectionFactory
```

```
connectionFactory){
        RabbitTemplate rabbitTemplate = new
RabbitTemplate(connectionFactory);
        rabbitTemplate.setMessageConverter(converter());
        return  rabbitTemplate;
    }
```

Air Queue Consumer

```
@Service
public class AirQueueConsumer {

    private final EmailSender emailSender;

    public AirQueueConsumer(EmailSender emailSender) {
        this.emailSender = emailSender;
    }

    @RabbitListener(queues = {"${rabbitmq.air.queue.name}"})
    public void consume(AirQueueNotifyUsersDto airQueueNotifyUsersDto)
throws MessagingException {

        System.out.println("Received From Air Queue-> "+
airQueueNotifyUsersDto.toString());

        List<String> emails = airQueueNotifyUsersDto.getEmails();
        AirQueueDataDto airQueueDataDto =
airQueueNotifyUsersDto.getAirData();
        String location = airQueueDataDto.getCoordinates();
        String time = airQueueDataDto.getTimestamp().substring(0, 23);
        String subject = "Alert: Air Data - Detected: "+location+ " -
Time: "+time;

        for (String email : emails) {
            emailSender.sendAirDataEmail(email, subject,
airQueueDataDto);
        }

    }

}
```

Water Queue Consumer

```
@Service
public class WaterQueueConsumer {
```

```
    private final EmailSender emailSender;

    public WaterQueueConsumer(EmailSender emailSender) {
        this.emailSender = emailSender;
    }

    @RabbitListener(queues = {"${rabbitmq.water.queue.name}"})
    public void consume(WaterQueueNotifyUsersDto
waterQueueNotifyUsersDto) throws MessagingException {

        System.out.println("Received From Water Queue-> "+
waterQueueNotifyUsersDto.toString());

        List<String> emails =
waterQueueNotifyUsersDto.getEmails();
        WaterQueueDataDto waterQueueDataDto =
waterQueueNotifyUsersDto.getWaterData();
        String location = waterQueueDataDto.getCoordinates();
        String time =
waterQueueDataDto.getTimestamp().substring(0, 23);
        String subject = "Alert: Water Data - Detected:
"+location+ " - Time: "+time;

        for (String email : emails) {
            emailSender.sendWaterDataEmail(email, subject,
waterQueueDataDto);
        }

    }

}
```

Soil Queue Consumer

```
@Service
public class SoilQueueConsumer {

    private final EmailSender emailSender;

    public SoilQueueConsumer(EmailSender emailSender) {
```

```
        this.emailSender = emailSender;
    }

    @RabbitListener(queues = {"${rabbitmq.soil.queue.name}"})
    public void consume(SoilQueueNotifyUsersDto
soilQueueNotifyUsersDto) throws MessagingException {

        System.out.println("Received From Soil Queue-> "+
soilQueueNotifyUsersDto.toString());

        List<String> emails = soilQueueNotifyUsersDto.getEmails();
        SoilQueueDataDto soilQueueDataDto =
soilQueueNotifyUsersDto.getSoilData();
        String location = soilQueueDataDto.getCoordinates();
        String time = soilQueueDataDto.getTimestamp().substring(0,
23);
        String subject = "Alert: Soil Data - Detected: "+location+
" - Time: "+time;

        for (String email : emails) {
            emailSender.sendSoilDataEmail(email, subject,
soilQueueDataDto);
        }

    }

}
```

Email Sender

```
@Service
public class EmailSender {
    @Autowired
    private JavaMailSender mailSender;

    public void sendEmail(String toEmail, String subject, String
body){
        SimpleMailMessage message = new SimpleMailMessage();
        message.setFrom("th3m.them@gmail.com");
        message.setTo(toEmail);
        message.setText(body);
        message.setSubject(subject);
```

```
        mailSender.send(message);
    }

    public void sendAirDataEmail(String toEmail, String subject,
AirQueueDataDto airData) throws MessagingException {

        String body = generateHtmlTableForAirData(airData);

        MimeMessage message = mailSender.createMimeMessage();
        MimeMessageHelper helper = new MimeMessageHelper(message,
true, "UTF-8");

        try {
            helper.setFrom("th3m.them@gmail.com");
            helper.setTo(toEmail);
            helper.setText(body, true);
            helper.setSubject(subject);

            mailSender.send(message);
        } catch (MessagingException e) {
            e.printStackTrace(); // Handle the exception according
to your needs
        }
    }

    public void sendWaterDataEmail(String email, String subject,
WaterQueueDataDto waterQueueDataDto) throws MessagingException {

        String body =
generateHtmlTableForWaterData(waterQueueDataDto);

        MimeMessage message = mailSender.createMimeMessage();
        MimeMessageHelper helper = new MimeMessageHelper(message,
true, "UTF-8");

        try {
            helper.setFrom("th3m.them@gmail.com");
            helper.setTo(email);
            helper.setText(body, true);
            helper.setSubject(subject);
```

```
            mailSender.send(message);
        } catch (MessagingException e) {
            e.printStackTrace(); // Handle the exception according
to your needs
        }
    }

    public void sendSoilDataEmail(String email, String subject,
SoilQueueDataDto soilQueueDataDto) throws MessagingException {

        String body =
generateHtmlTableForSoilData(soilQueueDataDto);

        MimeMessage message = mailSender.createMimeMessage();
        MimeMessageHelper helper = new MimeMessageHelper(message,
true, "UTF-8");

        try {
            helper.setFrom("th3m.them@gmail.com");
            helper.setTo(email);
            helper.setText(body, true);
            helper.setSubject(subject);

            mailSender.send(message);
        } catch (MessagingException e) {
            e.printStackTrace(); // Handle the exception according
to your needs
        }
    }

    private String generateHtmlTableForSoilData(SoilQueueDataDto
soilQueueDataDto) {
        StringBuilder htmlBody = new
StringBuilder("<html><head><style>"
                + "table { width: 100%; border-collapse: separate;
border-spacing: 0; margin-top: 20px; }"
                + "th, td { padding: 10px; text-align: left;
border-bottom: 1px solid #ddd; }"
                + "th { background-color: #f8f8f8; }"
                + "tr:nth-child(even) { background-color: #f2f2f2;
}"
                + "td:nth-child(2) { padding-right: 5px; color:
```

```
#007BFF; }" // Color for the "Value" column
                + "td:nth-child(3) { color: #28A745; }" // Color
for the "Unit" column
                + "</style></head><body><h2>Soil Quality
Data</h2>");

        // Create an HTML table with the soil data and measurement
units

htmlBody.append("<table><tr><th>Parameter</th><th>Value</th><th>Un
it</th></tr>");

htmlBody.append("<tr><td>Temperature</td><td>").append(soilQueueDa
taDto.getSoilTemperature()).append("</td><td>°C</td></tr>");

htmlBody.append("<tr><td>Moisture</td><td>").append(soilQueueDataD
to.getSoilMoisture()).append("</td><td>cbar</td></tr>");
        htmlBody.append("<tr><td>Electronic
Conductivity</td><td>").append(soilQueueDataDto.getElectronicCondu
ctivity()).append("</td><td>mS/m</td></tr>");
        htmlBody.append("<tr><td>Volumetric Water
Content</td><td>").append(soilQueueDataDto.getVolumetricWaterConte
nt()).append("</td><td>dS/m</td></tr>");

htmlBody.append("<tr><td>Salinity</td><td>").append(soilQueueDataD
to.getSalinity()).append("</td><td></td></tr>");
        htmlBody.append("<tr><td>Total Suspended
Solids</td><td>").append(soilQueueDataDto.getTotalSuspendedSolids(
)).append("</td><td></td></tr>");

htmlBody.append("<tr><td>Flow/Volume</td><td>").append(soilQueueDa
taDto.getFlowVolume()).append("</td><td>m3</td></tr>");

htmlBody.append("<tr><td>Nitrate</td><td>").append(soilQueueDataDt
o.getNitrate()).append("</td><td>mg/L</td></tr>");

htmlBody.append("<tr><td>Timestamp</td><td>").append(soilQueueData
Dto.getTimestamp()).append("</td><td></td></tr>");

htmlBody.append("<tr><td>Location</td><td>").append(soilQueueDataD
to.getCoordinates()).append("</td><td></td></tr>");
        htmlBody.append("</table></body></html>");
```

```
        return htmlBody.toString();
    }

    private String generateHtmlTableForWaterData(WaterQueueDataDto
waterQueueDataDto) {
        StringBuilder htmlBody = new
StringBuilder("<html><head><style>"
                + "table { width: 100%; border-collapse: separate;
border-spacing: 0; margin-top: 20px; }"
                + "th, td { padding: 10px; text-align: left;
border-bottom: 1px solid #ddd; }"
                + "th { background-color: #f8f8f8; }"
                + "tr:nth-child(even) { background-color: #f2f2f2;
}"
                + "td:nth-child(2) { padding-right: 5px; color:
#007BFF; }" // Color for the "Value" column
                + "td:nth-child(3) { color: #28A745; }" // Color
for the "Unit" column
                + "</style></head><body><h2>Water Quality
Data</h2>");

        // Create an HTML table with the water data and
measurement units

htmlBody.append("<table><tr><th>Parameter</th><th>Value</th><th>Un
it</th></tr>");
        htmlBody.append("<tr><td>Dissolved
Oxygen</td><td>").append(waterQueueDataDto.getDissolvedOxygen()).a
ppend("</td><td>mg/L</td></tr>");
        htmlBody.append("<tr><td>Oxidation-Reduction
Potential</td><td>").append(waterQueueDataDto.getOxidationReductio
nPotential()).append("</td><td>mV</td></tr>");

htmlBody.append("<tr><td>pH</td><td>").append(waterQueueDataDto.ge
tPH()).append("</td><td></td></tr>");

htmlBody.append("<tr><td>Turbidity</td><td>").append(waterQueueDat
aDto.getTurbidity()).append("</td><td>NTU</td></tr>");
        htmlBody.append("<tr><td>Total Dissolved
Solids</td><td>").append(waterQueueDataDto.getTotalDissolvedSolids
()).append("</td><td>ppm</td></tr>");
```

```
htmlBody.append("<tr><td>Temperature</td><td>").append(waterQueueD
ataDto.getTemperature()).append("</td><td>°C</td></tr>");

htmlBody.append("<tr><td>Timestamp</td><td>").append(waterQueueDat
aDto.getTimestamp()).append("</td><td></td></tr>");

htmlBody.append("<tr><td>Location</td><td>").append(waterQueueData
Dto.getCoordinates()).append("</td><td></td></tr>");
        htmlBody.append("</table></body></html>");

        return htmlBody.toString();
    }

    private String generateHtmlTableForAirData(AirQueueDataDto
airData) {
        StringBuilder htmlBody = new
StringBuilder("<html><head><style>"
                + "table { width: 100%; border-collapse: separate;
border-spacing: 0; margin-top: 20px; }"
                + "th, td { padding: 10px; text-align: left;
border-bottom: 1px solid #ddd; }"
                + "th { background-color: #f8f8f8; }"
                + "tr:nth-child(even) { background-color: #f2f2f2;
}"
                + "td:nth-child(2) { padding-right: 5px; color:
#007BFF; }" // Color for the "Value" column
                + "td:nth-child(3) { color: #28A745; }" // Color
for the "Unit" column
                + "</style></head><body><h2>Air Quality
Data</h2>");

        // Create an HTML table with the sensor data and
measurement units

htmlBody.append("<table><tr><th>Parameter</th><th>Value</th><th>Un
it</th></tr>");

htmlBody.append("<tr><td>Temperature</td><td>").append(airData.get
AirTemperature()).append("</td><td>°C</td></tr>");

htmlBody.append("<tr><td>Humidity</td><td>").append(airData.getAir
```

```
Humidity()).append("</td><td>%</td></tr>");

htmlBody.append("<tr><td>CO2</td><td>").append(airData.getAirC02()
).append("</td><td>ppm</td></tr>");

htmlBody.append("<tr><td>VOCs</td><td>").append(airData.getAirVOCs
()).append("</td><td>ppb</td></tr>");

htmlBody.append("<tr><td>PM2.5</td><td>").append(airData.getAirPM2
5()).append("</td><td>µg/m³</td></tr>");

htmlBody.append("<tr><td>CO</td><td>").append(airData.getAirC0()).
append("</td><td>ppm</td></tr>");

htmlBody.append("<tr><td>Timestamp</td><td>").append(airData.getTi
mestamp()).append("</td><td></td></tr>");

htmlBody.append("<tr><td>Location</td><td>").append(airData.getCoo
rdinates()).append("</td><td></td></tr>");
        htmlBody.append("</table></body></html>");

        return htmlBody.toString();
    }
```

8.2 Υλοποίηση Διεπαφής Χρήστη

Παρακάτω βρίσκεται ο κώδικας όλων των διαφορετικών εξαρτημάτων που αποτελούν την διεπαφή χρήστη.

Home

```
import Navbar from './Navbar.jsx'
import UserGuide from './UserGuide.jsx'


function Home() {
    return (
        <div className="app-container">
            <Navbar />
            <UserGuide />
        </div>
    );
}

export default Home;
```

Το βασικό component home αποτελεί την κεντρική σελίδα της εφαρμογής και καλεί ένα navbar που λειτουργεί σαν περιηγητής για να κάνει ο χρήστης τις απαραίτητες αναζητήσεις.

Navbar

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import "../styles/Navbar.css";

function Navbar(){

    const [location, setLocation] = useState('');
    const [startDate, setStartDate] = useState('');
    const [endDate, setEndDate] = useState('');
    const [pollutionType, setPollutionType] = useState("");

    const navigate = useNavigate()

    const [showDates, setShowDates] = useState(true); //
```

```
Initialize showDates state

    const handleToggleDates = () => {
        setShowDates(!showDates); // Toggle showDates state when
the button is clicked
    };

    const europeanCapitals = [
        "Amsterdam",
        "Athens",
        "Berlin",
        "Brussels",
        "Budapest",
        "Copenhagen",
        "Dublin",
        "Helsinki",
        "Lisbon",
        "London",
        "Madrid",
        "Oslo",
        "Paris",
        "Prague",
        "Rome",
        "Stockholm",
        "Vienna",
        "Warsaw",
        "Zurich"
    ];

    const handleSearch = async () => {
        if(location.trim() === "") {
            alert("Please choose a location.");
            window.location.reload();
            return
        }

        if( pollutionType.trim() === "") {
          alert("Please choose a data type");
          window.location.reload();
          return
      }
```

```
        if ((startDate.trim() === "" && endDate.trim() === "") ||
(startDate.trim() !== "" && endDate.trim() !== "")) {

            if(pollutionType.trim() ==="air" && (startDate.trim()
=== "" && endDate.trim() === "")){
                try {
                    const response = await
fetch(`http://localhost:8086/air-microservice/air/search?location=
${location}`);
                    if (!response.ok) {
                      throw new Error('Failed to fetch data');
                    }
                    const data = await response.json();

                    navigate('/search-air', { state: { data } });

                  } catch (error) {
                    console.error('Error fetching data:', error);
                    alert("An error occurred while fetching
data");
                  }
            }

            if(pollutionType.trim() ==="air" && (startDate.trim()
!== "" && endDate.trim() !== "")){
                const startCPM = new Date(startDate);
                const endCPM = new Date(endDate);
                const today = new Date();

                if (startCPM >= endCPM) {
                    alert("Start date should be less than end
date");
                    return;
                }

                if (endCPM > today) {
                    alert("End date cannot be greater than the
present day");
                    return;
                }

                const start = startDate+"T00:00:00.000%2B00:00"
```

```
                const end = endDate+"T23:59:59.000%2B00:00"
                try {
                    const response = await
fetch(`http://localhost:8086/air-microservice/air/search-all?locat
ion=${location}&start=${start}&end=${end}`);
                    if (!response.ok) {
                      throw new Error('Failed to fetch data');
                    }
                    const data = await response.json();

                    navigate('/search-air', { state: { data } });

                  } catch (error) {
                    console.error('Error fetching data:', error);
                    alert("An error occurred while fetching
data");
                  }

            }

            if(pollutionType.trim() ==="water" &&
(startDate.trim() === "" && endDate.trim() === "")){
                try {
                    const response = await
fetch(`http://localhost:8086/water-microservice/water/search?locat
ion=${location}`);
                    if (!response.ok) {
                      throw new Error('Failed to fetch data');
                    }
                    const data = await response.json();

                    navigate('/search-water', { state: { data }
});

                  } catch (error) {
                    console.error('Error fetching data:', error);
                    alert("An error occurred while fetching
data");
                  }
            }

            if(pollutionType.trim() ==="water" &&
```

```
(startDate.trim() !== "" && endDate.trim() !== "")){
                const startCPM = new Date(startDate);
                const endCPM = new Date(endDate);
                const today = new Date();

                if (startCPM >= endCPM) {
                    alert("Start date should be less than end
date");
                    return;
                }

                if (endCPM > today) {
                    alert("End date cannot be greater than the
present day");
                    return;
                }

                const start = startDate+"T00:00:00.000%2B00:00"
                const end = endDate+"T23:59:59.000%2B00:00"
                try {
                    const response = await
fetch(`http://localhost:8086/water-microservice/water/search-all?l
ocation=${location}&start=${start}&end=${end}`);
                    if (!response.ok) {
                      throw new Error('Failed to fetch data');
                    }
                    const data = await response.json();

                    navigate('/search-water', { state: { data }
});

                  } catch (error) {
                    console.error('Error fetching data:', error);
                    alert("An error occurred while fetching
data");
                  }

            }

            if(pollutionType.trim() ==="soil" && (startDate.trim()
=== "" && endDate.trim() === "")){
                try {
```

```
                    const response = await
fetch(`http://localhost:8086/soil-microservice/soil/search?locatio
n=${location}`);
                    if (!response.ok) {
                      throw new Error('Failed to fetch data');
                    }
                    const data = await response.json();

                    navigate('/search-soil', { state: { data } });

                  } catch (error) {
                    console.error('Error fetching data:', error);
                    alert("An error occurred while fetching
data");
                  }
            }

            if(pollutionType.trim() ==="soil" && (startDate.trim()
!== "" && endDate.trim() !== "")){
                const startCPM = new Date(startDate);
                const endCPM = new Date(endDate);
                const today = new Date();

                if (startCPM >= endCPM) {
                    alert("Start date should be less than end
date");
                    return;
                }

                if (endCPM > today) {
                    alert("End date cannot be greater than the
present day");
                    return;
                }

                const start = startDate+"T00:00:00.000%2B00:00"
                const end = endDate+"T23:59:59.000%2B00:00"
                try {
                    const response = await
fetch(`http://localhost:8086/soil-microservice/soil/search-all?loc
ation=${location}&start=${start}&end=${end}`);
                    if (!response.ok) {
```

```
                    throw new Error('Failed to fetch data');
                  }
                  const data = await response.json();

                  navigate('/search-soil', { state: { data } });

                } catch (error) {
                  console.error('Error fetching data:', error);
                  alert("An error occurred while fetching
data");
                }

            }


        }else{
            alert("Dates are invalid! Specify both start and end
dates or none at all to check live data");
            window.location.reload();
        }


    };

    return(
        <nav className="navbar">
            <div className="navbar-brand">
                <img src="/logo.png" alt="Environment Monitoring
Logo" />
                <a href="/" className="app-name"> Environment
Pollution Monitoring</a>
            </div>
            <div className="search-container">

                <button onClick={handleToggleDates}
className='calendar-btn'>{!showDates ? 'Live' :
'History'}</button>

                {!showDates && (
                    <><label htmlFor="startDate">Start
Date</label><input
                        type="date"
```

```
                        id="startDate"
                        value={startDate}
                        onChange={(e) =>
setStartDate(e.target.value)} /><label htmlFor="endDate">End
Date</label><input
                            type="date"
                            id="endDate"
                            value={endDate}
                            onChange={(e) =>
setEndDate(e.target.value)} /></>
                )}

                <select value={pollutionType} onChange={(e) =>
setPollutionType(e.target.value)}>
                    <option value="" disabled>Select data
type</option>
                    <option value="air">Air</option>
                    <option value="water">Water</option>
                    <option value="soil">Soil</option>
                </select>

                <select
                    value={location}
                    onChange={(e) => setLocation(e.target.value)}
                >
                    <option value="" disabled>Select a
City</option>
                    {europeanCapitals.map((capital, index) => (
                        <option key={index}
value={capital}>{capital}</option>
                    ))}
                </select>

                <button onClick={handleSearch}>Search</button>

                <div className="about-container">
                    <a href="/alert" className="about-link">Alert
Me</a>
                </div>

            </div>
        </nav>
```

```
    );
}

export default Navbar;
```

AirSearch

```
import React, { useEffect, useState } from 'react';
import Navbar from './Navbar.jsx';
import '../styles/AirSearch.css';
import { useLocation } from 'react-router-dom';
import { Line } from 'react-chartjs-2';
import { Chart } from 'react-chartjs-2';
import 'chart.js/auto';
import FileSaver from 'file-saver';

function AirSearch(props) {
    const [chart, setChart] = useState(null);
    const location = useLocation();
    const jsonData = location.state?.data || {};
    const [stats, setStats] = useState([]);

    const tableData = [
        { metric: 'Mean', temperature:
stats.airTemperatureMean?.toFixed(4), humidity:
stats.airHumidityMean?.toFixed(4), co2:
stats.airC02Mean?.toFixed(4), vocs: stats.airC02Mean?.toFixed(4),
pm25: stats.airPM25Mean?.toFixed(4) , co:
stats.airC0Mean?.toFixed(4)},
        { metric: 'Median', temperature:
stats.airTemperatureMedian?.toFixed(4), humidity:
stats.airHumidityMedian?.toFixed(4), co2:
stats.airC02Median?.toFixed(4), vocs:
stats.airC02Median?.toFixed(4), pm25:
stats.airPM25Median?.toFixed(4) , co:
stats.airC0Median?.toFixed(4) },
        { metric: 'Range', temperature:
stats.airTemperatureRange?.toFixed(4), humidity:
stats.airHumidityRange?.toFixed(4), co2:
stats.airC02Range?.toFixed(4),  vocs:
stats.airC02Range?.toFixed(4), pm25:
```

```
stats.airPM25Range?.toFixed(4) , co: stats.airC0Range?.toFixed(4)
},
        { metric: 'Standard Deviation', temperature:
stats.airTemperatureStandardDeviation?.toFixed(4), humidity:
stats.airHumidityStandardDeviation?.toFixed(4), co2:
stats.airC02StandardDeviation?.toFixed(4), vocs:
stats.airC02StandardDeviation?.toFixed(4), pm25:
stats.airPM25StandardDeviation?.toFixed(4) , co:
stats.airC0StandardDeviation?.toFixed(4) },
    ];

    const startTimestamp = jsonData.length > 0 ?
jsonData[0].timestamp : 'N/A';
    const endTimestamp = jsonData.length > 0 ?
jsonData[jsonData.length - 1].timestamp : 'N/A';
    const coordinates = jsonData.length > 0 ?
jsonData[0].coordinates : 'N/A';
    let content;

    const infoContent = (
        <div>
            <h3><b className="info-section">Location:</b>
{coordinates}<b className="info-section"> Start Time:</b>
{formatTimestamp(startTimestamp)}<b className="info-section"> End
Time:</b> {formatTimestamp(endTimestamp)}</h3>
        </div>
    );

    const [isModalOpen, setIsModalOpen] = useState(false);
    const openModal = () => {
        setIsModalOpen(true);
    };

    // Function to close the modal
    const closeModal = () => {
        setIsModalOpen(false);
    };

    const [isGraphOpen, setisGraphOpen] = useState(false);
    const isGraph = () => {
        setisGraphOpen(true);
    };
```

```
    // Function to close the modal
    const closeIsGraphOpen = () => {
        setisGraphOpen(false);
    };

    useEffect(() => {
        // Auto-close the modal when the page changes
        closeModal();
        closeIsGraphOpen();
    }, [location]);

    const rangeValues = (
        <><div className="range-chart" id="temperature">
            <h2>Temperature Range (°C)</h2>
            <br></br>
            <div className="range-bar">
                <div className="range very-low">
                    <span className="label">Very Low: &lt;
10°C</span>
                </div>
                <div className="range low">
                    <span className="label">Low: 10°C -
15°C</span>
                </div>
                <div className="range normal">
                    <span className="label">Normal: 15°C -
25°C</span>
                </div>
                <div className="range high">
                    <span className="label">High: 25°C -
30°C</span>
                </div>
                <div className="range very-high">
                    <span className="label">Very High: &gt;
30°C</span>
                </div>
            </div>
        </div><div className="range-chart" id="humidity">
                <h2>Humidity Range (%)</h2>
                <br></br>
                <div className="range-bar">
```

```
                    <div className="range very-low">
                        <span className="label">Very Low: &lt;
20%</span>
                    </div>
                    <div className="range low">
                        <span className="label">Low: 20% -
40%</span>
                    </div>
                    <div className="range normal">
                        <span className="label">Normal: 40% -
60%</span>
                    </div>
                    <div className="range high">
                        <span className="label">High: 60% -
80%</span>
                    </div>
                    <div className="range very-high">
                        <span className="label">Very High: &gt;
80%</span>
                    </div>
                </div>
            </div><div className="range-chart" id="co">
                <h2>CO Range (ppm)</h2>
                <br></br>
                <div className="range-bar">
                    <div className="range very-low">
                        <span className="label">Very Low: &lt; 1
ppm</span>
                    </div>
                    <div className="range low">
                        <span className="label">Low: 1 - 5
ppm</span>
                    </div>
                    <div className="range normal">
                        <span className="label">Normal: 5 - 10
ppm</span>
                    </div>
                    <div className="range high">
                        <span className="label">High: 10 - 20
ppm</span>
                    </div>
                    <div className="range very-high">
```

```
                        <span className="label">Very High: &gt; 20
ppm</span>
                    </div>
                </div>
            </div><div className="range-chart" id="co2">
                <h2>CO2 Range (ppm)</h2>
                <br></br>
                <div className="range-bar">
                    <div className="range very-low">
                        <span className="label">Very Low: &lt; 400
ppm</span>
                    </div>
                    <div className="range low">
                        <span className="label">Low: 400 - 800
ppm</span>
                    </div>
                    <div className="range normal">
                        <span className="label">Normal: 800 - 1000
ppm </span>
                    </div>
                    <div className="range high">
                        <span className="label">High: 1000 - 2000
ppm</span>
                    </div>
                    <div className="range very-high">
                        <span className="label">Very High: &gt;
2000 ppm</span>
                    </div>
                </div>
            </div><div className="range-chart" id="pm25">
                <h2>PM2.5 Range (µg/m³)</h2>
                <br></br>
                <div className="range-bar">
                    <div className="range very-low">
                        <span className="label">Very Low: &lt; 10
µg/m³</span>
                    </div>
                    <div className="range low">
                        <span className="label">Low: 10 - 20
µg/m³</span>
                    </div>
                    <div className="range normal">
```

```
                        <span className="label">Normal: 20 - 30
µg/m³</span>
                    </div>
                    <div className="range high">
                        <span className="label">High: 30 - 50
µg/m³</span>
                    </div>
                    <div className="range very-high">
                        <span className="label">Very High: &gt; 50
µg/m³</span>
                    </div>
                </div>
            </div><div className="range-chart" id="vocs">
                <h2>VOCs Range (ppb)</h2>
                <br></br>
                <div className="range-bar">
                    <div className="range very-low">
                        <span className="label">Very Low: &lt; 5
ppb</span>
                    </div>
                    <div className="range low">
                        <span className="label">Low: 5 - 20
ppb</span>
                    </div>
                    <div className="range normal">
                        <span className="label">Normal: 20 - 50
ppb</span>
                    </div>
                    <div className="range high">
                        <span className="label">High: 50 - 100
ppb</span>
                    </div>
                    <div className="range very-high">
                        <span className="label">Very High: &gt;
100 ppb</span>
                    </div>
                </div>
            </div></>
    );

    function formatTimestamp(timestamp) {
        const dateObject = new Date(timestamp);
```

```
        const formattedDate = dateObject.toLocaleDateString();
        const formattedTime = dateObject.toLocaleTimeString([], {
hour: '2-digit', minute: '2-digit' });
        return `${formattedDate} at ${formattedTime}`;
    }

    function convertJsonToCsv(jsonData) {
        const fields = [
          'airTemperature',
          'airHumidity',
          'airC02',
          'airVOCs',
          'airPM25',
          'airC0',
          'timestamp',
          'coordinates',
        ];

        const csvRows = [];
        // Add the header row with field names
        csvRows.push(fields.join(','));

        // Iterate over the JSON data and create CSV rows
        jsonData.forEach((data) => {
          const values = fields.map((field) => data[field]);
          csvRows.push(values.join(','));
        });

        return csvRows.join('\n');
      }

      function downloadCSV() {
        try {
          const csvData = convertJsonToCsv(jsonData);
          const blob = new Blob([csvData], { type:
'text/csv;charset=utf-8;' });
          FileSaver.saveAs(blob, 'airData.csv');
        } catch (error) {
          console.error('Error while downloading CSV:', error);
        }
      }
```

```
    const humidityChartData = {
        labels: Array.isArray(jsonData) ? jsonData.map(item =>
formatTimestamp(item.timestamp)) : [],
        datasets: [
          {
            label: 'Humidity (%) Over Time',
            data: Array.isArray(jsonData) ? jsonData.map(item =>
item.airHumidity) : [],
            fill: false,
            borderColor:  'rgb(255, 0, 0)',
            tension: 0.1,
          },
        ],
      };

    const c02ChartData = {
        labels: Array.isArray(jsonData) ? jsonData.map(item =>
formatTimestamp(item.timestamp)) : [],
        datasets: [
          {
            label: 'CO2 (ppm) Over Time',
            data: Array.isArray(jsonData) ? jsonData.map(item =>
item.airC02) : [],
            fill: false,
            borderColor: 'rgb(0, 0, 255)',
            tension: 0.1,
          },
        ],
    };

    const vocsChartData = {
        labels: Array.isArray(jsonData) ? jsonData.map(item =>
formatTimestamp(item.timestamp)) : [],
        datasets: [
          {
            label: 'VOCs (ppb) Over Time',
            data: Array.isArray(jsonData) ? jsonData.map(item =>
item.airVOCs) : [],
            fill: false,
            borderColor: 'rgb(255, 255, 0)',
```

```
            tension: 0.1,
          },
        ],
    };

    const pm2ChartData = {
        labels: Array.isArray(jsonData) ? jsonData.map(item =>
formatTimestamp(item.timestamp)) : [],
        datasets: [
          {
            label: 'PM2.5 (µg/m³) Over Time',
            data: Array.isArray(jsonData) ? jsonData.map(item =>
item.airPM25) : [],
            fill: false,
            borderColor: 'rgb(128, 0, 128)',
            tension: 0.1,
          },
        ],
    };

    const c0ChartData = {
        labels: Array.isArray(jsonData) ? jsonData.map(item =>
formatTimestamp(item.timestamp)) : [],
        datasets: [
          {
            label: 'C0 (ppm) Over Time',
            data: Array.isArray(jsonData) ? jsonData.map(item =>
item.airPM25) : [],
            fill: false,
            borderColor: 'rgb(0, 128, 0)',
            tension: 0.1,
          },
        ],
    };

    const temperatureChartData = {
        labels: Array.isArray(jsonData) ? jsonData.map(item =>
formatTimestamp(item.timestamp)) : [],
        datasets: [
            {
                label: 'Temperature (°C) Over Time',
                data: Array.isArray(jsonData) ? jsonData.map(item
```

```
=> item.airTemperature) : [],
                fill: false,
                borderColor: 'rgb(75, 192, 192)',
                tension: 0.1,
            },
        ],
    };

    useEffect(() => {

        async function postDataToBackend() {
            const url =
'http://localhost:8086/analytics-microservice/analytics/basic-air'
;

            try {
                const response = await fetch(url, {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                    },
                    body: JSON.stringify(jsonData), // Assuming
you have jsonData defined
                });

                if (response.ok) {
                    const data = await response.json();
                    setStats(data)

                    if (chart) {
                        chart.destroy();
                    }
                    const canvasId =
`chart-${Math.floor(Math.random() * 10000)}`;
                    // Create a new chart and set it in the state
                    const newChart = new
Chart(document.getElementById(canvasId), {
                        type: 'line',
                        data:
temperatureChartData,humidityChartData,c02ChartData,c0ChartData,pm
2ChartData,vocsChartData
                    });
```

```
                    setChart(newChart);

                } else {
                    console.error('Error:', response.statusText);
                }
            } catch (error) {
                console.error('Error:', error);
            }
        }

        if (Array.isArray(jsonData) &&
Object.keys(jsonData).length > 0) {
            postDataToBackend();
        }

    }, [jsonData, chart]);

    if (Array.isArray(jsonData)) {

        console.log('Received jsonData:', jsonData);

        content = (
        <div className="stats-table-container">
            <h1>Air Statistics</h1>
            {infoContent}
            <table className="stats-table">
                <thead>
                    <tr>
                        <th>Metric</th>
                        <th>Temperature (°C)</th>
                        <th>Humidity (%)</th>
                        <th>CO2 (ppm)</th>
                        <th>VOCs (ppb)</th>
                        <th>PM2.5 (µg/m³)</th>
                        <th>CO (ppm)</th>
                    </tr>
                </thead>
                <tbody>
                    {tableData.map((row, index) => (
                        <tr key={index}>
                            <td>{row.metric}</td>
                            <td>{row.temperature}</td>
```

```
                            <td>{row.humidity}</td>
                            <td>{row.co2}</td>
                            <td>{row.vocs}</td>
                            <td>{row.pm25}</td>
                            <td>{row.co}</td>
                        </tr>
                    ))}
                </tbody>
            </table>

            <br></br>
            {!isGraphOpen &&<button className="range-btn"
onClick={isGraph}>Show Time Graphs</button>}
            {isGraphOpen &&
            <><button className="range-btn"
onClick={closeIsGraphOpen}>Close Time Graphs</button><Line
data={temperatureChartData} /><Line data={humidityChartData}
/><Line data={c02ChartData} /><Line data={c0ChartData} /><Line
data={pm2ChartData} /><Line data={vocsChartData} /></>}
            <br></br>
            <br></br>
            {!isModalOpen &&<button className="range-btn"
onClick={openModal}>Check Range Values</button>}
                {isModalOpen && (
                <div className="modal">
                    <div className="modal-content">
                        {/* Close button */}
                        <button className="range-btn"
onClick={closeModal}>Close Range Values</button>
                        {rangeValues}
                    </div>
                </div>
            )}
             <h3 className="export-header">
            Export the dataset: <button
onClick={downloadCSV}>Download CSV</button>
            </h3>
        </div>

    );

    }else if (typeof jsonData === 'object' &&
```

```
Object.keys(jsonData).length > 0) {

        content = (
            <div className="data-single">
                <h1>Last Measured Air Data for
{jsonData.coordinates} at
{formatTimestamp(jsonData.timestamp)}</h1>
                <h2>Temperature: <b
className="unit">{jsonData.airTemperature}</b> °C</h2>
                <h2>Humidity: <b
className="unit">{jsonData.airHumidity}</b> %</h2>
                <h2>CO2: <b className="unit">{jsonData.airC02}</b>
ppm</h2>
                <h2>VOCs: <b
className="unit">{jsonData.airVOCs}</b> ppb</h2>
                <h2>PM2.5: <b
className="unit">{jsonData.airPM25}</b> µg/m³</h2>
                <h2>CO: <b className="unit">{jsonData.airC0}</b>
ppm</h2>
                {!isModalOpen &&<button className="range-btn"
onClick={openModal}>Check Range Values</button>}
                {isModalOpen && (
                <div className="modal">
                    <div className="modal-content">
                        {/* Close button */}
                        <button className="range-btn"
onClick={closeModal}>Close</button>
                        {rangeValues}
                    </div>
                </div>
            )}
            </div>
        );

    } else {

        content = <div className="data-single"><h1><b>No Data
Available</b></h1></div>;

    }

    return (
```

```
        <div className="app-container">
        <Navbar />
        {content}
        </div>
    );
}

export default AirSearch;
```

Το παραπάνω component είναι υπεύθυνο για να αντλεί τα δεδομένα από την αναζήτηση του χρήστη και να τα στέλνει στο Data Analysis Microservice, όπου θα υπολογιστούν τα στατιστικά. Επιπλέον, από αυτό μπορούμε να κατεβάσουμε το dataset που επιλέχθηκε. Αντίστοιχη λειτουργία έχουν το WaterSearch και το SoilSearch.

Alert

```
import React, { useState } from 'react';
import Navbar from './Navbar.jsx'
import "../styles/Alert.css";

function Alert() {
    const [formData, setFormData] = useState({
        email: '',
        location: '',
        dataType: '',
        parameter: '',
        comparisonOperator: '',
        thresholdValue: '',
    });

    const [jsonAirData, setJsonAirDataData] = useState({
        email: '',
        coordinates: '',
        lessAirTemperature: '',
        moreAirTemperature: '',
        lessAirHumidity: '',
        moreAirHumidity: '',
        lessAirC02: '',
        moreAirC02: '',
        lessAirVOCs: '',
        moreAirVOCs: '',
```

```
    lessAirPM25: '',
    moreAirPM25: '',
    lessAirC0: '',
    moreAirC0: '',
});

const [jsonWaterData, setJsonWaterDataData] = useState({
    email: '',
    coordinates: '',
    lessDissolvedOxygen: '',
    moreDissolvedOxygen: '',
    lessOxidationReductionPotential: '',
    moreOxidationReductionPotential: '',
    lessPH: '',
    morePH: '',
    lessTurbidity: '',
    moreTurbidity: '',
    lessTotalDissolvedSolids: '',
    moreTotalDissolvedSolids: '',
    lessTemperature: '',
    moreTemperature: '',
});

const [jsonSoilData, setJsonSoilDataData] = useState({
    email: '',
    coordinates: '',
    lessSoilTemperature: '',
    moreSoilTemperature: '',
    lessSoilMoisture: '',
    moreSoilMoisture: '',
    lessVolumetricWaterContent: '',
    moreVolumetricWaterContent: '',
    lessSalinity: '',
    moreSalinity: '',
    lessTotalSuspendedSolids: '',
    moreTotalSuspendedSolids: '',
    lessFlowVolume: '',
    moreFlowVolume: '',
    lessNitrate: '',
    moreNitrate: '',
});
```

```
const europeanCapitals = [
    "Amsterdam",
    "Athens",
    "Berlin",
    "Brussels",
    "Budapest",
    "Copenhagen",
    "Dublin",
    "Helsinki",
    "Lisbon",
    "London",
    "Madrid",
    "Oslo",
    "Paris",
    "Prague",
    "Rome",
    "Stockholm",
    "Vienna",
    "Warsaw",
    "Zurich"
];

const dataTypes = ['Air', 'Water', 'Soil'];

const airParameters = [
    'Temperature',
    'Humidity',
    'C02',
    'VOCs',
    'PM25',
    'C0',
];

const waterParameters = [
    'DissolvedOxygen',
    'Oxidation Reduction Potential',
    'PH',
    'Turbidity',
    'Total Dissolved Solids',
    'Temperature',
];
```

```
    const soilParameters = [
        'Temperature',
        'Moisture',
        'Electronic Conductivity',
        'Volumetric Water Content',
        'Salinity',
        'Total Suspended Solids',
        'Flow Volume',
        'Nitrate',
    ];

    const handleInputChange = async (e) => {
        const { name, value } = e.target;
        setFormData({
            ...formData,
            [name]: value,
        });
    };

    const parameterOptions = formData.dataType === 'Air'
        ? airParameters
        : formData.dataType === 'Water'
            ? waterParameters
            : soilParameters;

    const handleSubmit = async (e) => {
        e.preventDefault();

        const emailRegex =
/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i;
        if (!formData.email.match(emailRegex)) {
            alert('Please enter a valid email address.');
            return;
        }

        // Check if other inputs are not empty
        if (!formData.location || !formData.parameter ||
!formData.comparisonOperator || !formData.thresholdValue) {
            alert('Please fill in all fields.');
            return;
        }
```

```
if(formData.dataType==="Air"){
    let choice = null;
    if(formData.comparisonOperator==="less"){
        console.log("less");
        if(formData.parameter==="Temperature"){
            choice="lessAirTemperature";
        }else if(formData.parameter==="Humidity"){
            choice="lessAirHumidity";
        }
        else if(formData.parameter==="C02"){
            choice="lessAirC02";
        }
        else if(formData.parameter==="VOCs"){
            choice="lessAirVOCs";
        }
        else if(formData.parameter==="PM25"){
            choice="lessAirPM25";
        }else if(formData.parameter==="C0"){
            choice="lessAirC0";
        }

    }else{
        console.log("more");
        if(formData.parameter==="Temperature"){
            choice="moreAirTemperature";
        }else if(formData.parameter==="Humidity"){
            choice="moreAirHumidity";
        }
        else if(formData.parameter==="C02"){
            choice="moreAirC02";
        }
        else if(formData.parameter==="VOCs"){
            choice="moreAirVOCs";
        }
        else if(formData.parameter==="PM25"){
            choice="moreAirPM25";
        }else if(formData.parameter==="C0"){
            choice="moreAirC0";
        }
    }
    const updatedJsonAirData = {
        ...jsonAirData,
```

```
                email: formData.email,
                coordinates: formData.location,
            };

            updatedJsonAirData[choice] = formData.thresholdValue;

            setJsonAirDataData(updatedJsonAirData);

            try {
                const response = await
fetch('http://localhost:8086/air-microservice/air/notify', {
                    method: 'POST',
                    headers: {
                    'Content-Type': 'application/json',
                    },
                    body: JSON.stringify(updatedJsonAirData),
                });

                if (response.status === 200) {
                    alert('Successfully subscribed for
notifications!');
                    window.location.reload();
                } else {
                    alert('Failed to subscribe for
notifications.');
                }
                } catch (error) {
                console.error('Error sending alert:', error);
                alert('Error sending alert. Please try again
later.');
                }
        }

        if(formData.dataType==="Water"){
            let choice = null;
            if(formData.comparisonOperator==="less"){
                console.log("less");
                if(formData.parameter==="DissolvedOxygen"){
                    choice="lessDissolvedOxygen";
                }else if(formData.parameter==="Oxidation Reduction
Potential"){
                    choice="lessOxidationReductionPotential";
```

```
                }
                else if(formData.parameter==="PH"){
                    choice="lessPH";
                }
                else if(formData.parameter==="Turbidity"){
                    choice="lessTurbidity";
                }
                else if(formData.parameter==="Total Dissolved
Solids"){
                    choice="lessTotalDissolvedSolids";
                }else if(formData.parameter==="Temperature"){
                    choice="lessTemperature";
                }

            }else{
                console.log("more");
                if(formData.parameter==="DissolvedOxygen"){
                    choice="moreDissolvedOxygen";
                }else if(formData.parameter==="Oxidation Reduction
Potential"){
                    choice="moreOxidationReductionPotential";
                }
                else if(formData.parameter==="PH"){
                    choice="morePH";
                }
                else if(formData.parameter==="Turbidity"){
                    choice="moreTurbidity";
                }
                else if(formData.parameter==="Total Dissolved
Solids"){
                    choice="moreTotalDissolvedSolids";
                }else if(formData.parameter==="Temperature"){
                    choice="moreTemperature";
                }
            }
            const updatedJsonWaterData = {
                ...jsonWaterData,
                email: formData.email,
                coordinates: formData.location,
            };

            updatedJsonWaterData[choice] =
```

```
formData.thresholdValue;

            setJsonWaterDataData(updatedJsonWaterData);

            try {
                const response = await
fetch('http://localhost:8086/water-microservice/water/notify', {
                    method: 'POST',
                    headers: {
                    'Content-Type': 'application/json',
                    },
                    body: JSON.stringify(updatedJsonWaterData),
                });

                if (response.status === 200) {
                    alert('Successfully subscribed for
notifications');
                    window.location.reload();
                } else {
                    alert('Failed to subscribe for
notifications.');
                }
                } catch (error) {
                console.error('Error sending alert:', error);
                alert('Error sending alert. Please try again
later.');
                }
        }

        if(formData.dataType==="Soil"){
            let choice = null;
            if(formData.comparisonOperator==="less"){
                console.log("less");
                if(formData.parameter==="Temperature"){
                    choice="lessSoilTemperature";
                }
                else if(formData.parameter==="Moisture"){
                    choice="lessSoilMoisture";
                }
                else if(formData.parameter==="Electronic
Conductivity"){
                    choice="lessElectronicConductivity";
```

```
                }
                else if(formData.parameter==="Volumetric Water
Content"){
                    choice="lessVolumetricWaterContent";
                }else if(formData.parameter==="Salinity"){
                    choice="lessSalinity";
                }
                else if(formData.parameter==="Total Suspended
Solids"){
                    choice="lessTotalSuspendedSolids";
                }
                else if(formData.parameter==="Flow Volume"){
                    choice="lessFlowVolume";
                }
                else if(formData.parameter==="Nitrate"){
                    choice="lessNitrate";
                }

            }else{
                console.log("more");
                if(formData.parameter==="Temperature"){
                    choice="moreSoilTemperature";
                }
                else if(formData.parameter==="Moisture"){
                    choice="moreSoilMoisture";
                }
                else if(formData.parameter==="Electronic
Conductivity"){
                    choice="moreElectronicConductivity";
                }
                else if(formData.parameter==="Volumetric Water
Content"){
                    choice="moreVolumetricWaterContent";
                }else if(formData.parameter==="Salinity"){
                    choice="moreSalinity";
                }
                else if(formData.parameter==="Total Suspended
Solids"){
                    choice="moreTotalSuspendedSolids";
                }
                else if(formData.parameter==="Flow Volume"){
                    choice="moreFlowVolume";
```

```
                }
                else if(formData.parameter==="Nitrate"){
                    choice="moreNitrate";
                }
            }
            const updatedJsonSoilData = {
                ...jsonSoilData,
                email: formData.email,
                coordinates: formData.location,
            };

            updatedJsonSoilData[choice] = formData.thresholdValue;

            setJsonSoilDataData(updatedJsonSoilData);

            try {
                const response = await
fetch('http://localhost:8086/soil-microservice/soil/notify', {
                    method: 'POST',
                    headers: {
                    'Content-Type': 'application/json',
                    },
                    body: JSON.stringify(updatedJsonSoilData),
                });

                if (response.status === 200) {
                    alert('Successfully subscribed for
notifications');
                    window.location.reload();
                } else {
                    alert('Failed to subscribe for
notifications.');
                }
                } catch (error) {
                console.error('Error sending alert:', error);
                alert('Error sending alert. Please try again
later.');
                }
        }
        window.location.reload();
    };
```

```
    return (
        <div className="alert-general">
            <Navbar />
            <div className="alert-container">
                <h2 className="h2-data"><b>Type your information
below to be notified</b></h2>
                <br />
                <form className="alert-form"
onSubmit={handleSubmit}>
                    <div className="form-group">
                        <label
className="flabel"><b>Email:</b></label>
                        <input
                            type="email"
                            name="email"
                            value={formData.email}
                            onChange={handleInputChange}
                            required
                        />
                    </div>
                    <div className="form-group">
                        <label
className="flabel"><b>Location:</b></label>
                        <select
                        name="location"
                        value={formData.location}
                        onChange={handleInputChange}
                        required
                        >
                        <option value="" disabled>Select
Location</option>
                        {europeanCapitals.map((capital, index) =>
(
                            <option key={index} value={capital}>
                            {capital}
                            </option>
                        ))}
                        </select>
                    </div>
                    <div className="form-group">
                        <label className="flabel"><b>Data
```

```
Type:</b></label>
                        <select
                            name="dataType"
                            value={formData.dataType}
                            onChange={handleInputChange}
                        >
                            <option value="" disabled>Select data
type</option>
                            {dataTypes.map((type) => (
                                <option key={type} value={type}>
                                    {type}
                                </option>
                            ))}
                        </select>
                    </div>
                    <div className="form-group">
                        <label
className="flabel"><b>Parameter:</b></label>
                        <select
                            name="parameter"
                            value={formData.parameter}
                            onChange={handleInputChange}
                            required
                        >
                            <option value="" disabled>Select
Parameter</option>
                            {parameterOptions.map((param) => (
                                <option key={param} value={param}>
                                    {param}
                                </option>
                            ))}
                        </select>
                    </div>
                    <div className="form-group">
                        <label
className="flabel"><b>Comparison:</b></label>
                        <select
                            name="comparisonOperator"
                            value={formData.comparisonOperator}
                            onChange={handleInputChange}
                            required
                        >
```

```
                            <option value="" disabled>Select
Operator</option>
                            <option value="less">Less or Equal
Than</option>
                            <option value="greater">Greater or
Equal Than</option>
                        </select>
                    </div>
                    <div className="form-group">
                        <label
className="flabel"><b>Threshold:</b></label>
                        <input
                            type="number"
                            name="thresholdValue"
                            value={formData.thresholdValue}
                            onChange={handleInputChange}
                            required
                        />
                    </div>
                    <button className="alert-btn"
type="submit">Alert Me</button>
                    <h4 className="h4-data">Stop receiving alerts
by clicking here:
                     <a href="/alert-stop" className="stop-alert">
here</a></h4>
                </form>
            </div>
        </div>
    );
}

export default Alert;
```