ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών
Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών
Ειδίκευση Λογισμικού και Πληροφοριακών Συστημάτων,

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Εφαρμογής Έξυπνων Κάδων με χρήση Τεχνολογιών Μικροϋπηρεσιών και Εικονικοποίησης βασισμένης σε Περιέκτες

Ιωάννης Κ. Θεοδωρόπουλος
Α.Μ. 20005

Στυλιανός Θ. Καραγιαννάκης
Α.Μ. 20026

Επιβλέπων: Βασίλειος Μάμαλης
Συν επιβλέπων: Απόστολος Αναγνωστόπουλος

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Εφαρμογής Έξυπνων Κάδων με χρήση Τεχνολογιών Μικροϋπηρεσιών και Εικονικοποίησης βασισμένης σε Περιέκτες

Ιωάννης Κ. Θεοδωρόπουλος
Α.Μ. 20005

Στυλιανός Θ. Καραγιαννάκης
Α.Μ. 20026

Επιβλέπων: Βασίλειος Μάμαλης
Συν επιβλέπων: Απόστολος Αναγνωστόπουλος

Εξεταστική Επιτροπή:

Αντώνιος Μπόγρης, Καθηγητής
Βασίλειος Μάμαλης, Καθηγητής
Παναγιώτης Καρκαζής, Αναπληρωτής Καθηγητής

Antonios Bogris
Digitally signed by Antonios Bogris
Date: 2023.04.19 23:49:45 +03'00'

Vasileios Mamalis
Digitally signed by Vasileios Mamalis
Date: 2023.04.13 13:16:52 +03'00'

Panagiotis Karkazis
Digitally signed by Panagiotis Karkazis
Date: 2023.04.18 12:13:33 +03'00'

Ημερομηνία εξέτασης
10 / 04 / 2023

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΩΝ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Οι κάτωθι υπογεγραμμένοι Θεοδωρόπουλος Ιωάννης του Κωνσταντίνου με αριθμό μητρώου 20005 και Καραγιαννάκης Στυλιανός του Θωμά με αριθμό μητρώου 20026 φοιτητές του Προγράμματος Μεταπτυχιακών Σπουδών Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνουμε ότι: «Είμαστε οι συγγράφεις αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες κάναμε 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, βεβαιώνουμε ότι αυτή η εργασία έχει συγγραφεί από εμάς αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μας, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μας ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μας».

Οι Δηλώντες

Θεοδωρόπουλος Ιωάννης

Καραγιαννάκης Στυλιανός

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία ολοκληρώθηκε μετά από επίμονες και επίπονες προσπάθειες, σε ένα άκρως ενδιαφέρον αλλά άγνωστο γνωστικό αντικείμενο σε εμάς, όπως αυτό της επικοινωνίας μεταξύ έξυπνων κάδων με σκοπό την επεξεργασία και αποθήκευση των τιμών μέτρησης εικονικών αισθητήρων του κάθε κάδου. Την προσπάθειά μας αυτή υποστήριξαν οι επιβλέποντες καθηγητές μας, τους οποίους θα θέλαμε να ευχαριστήσουμε δεόντως. Μεγάλη συμπαράσταση σε όλη αυτή την διαδρομή και βοήθεια για την επίτευξη του στόχου μου, παρείχαν η γυναίκα μου Μαριλένα Βαγγελάτου και οι γονείς της Νίκος και Ελένη - Ανδρομάχη, η μητέρα μου Παναγιώτα Καπλάνη και τα αδέλφια μου Παύλος, Ρήγας και Πάνος στους οποίους αφιερώνω την επιτυχία αυτή. Επίσης ένα μεγάλο ευχαριστώ στον συνάδελφο και φίλο Στυλιανό Καραγιαννάκη και στην συνάδελφο Χριστίνα Νίκα που με έπεισαν και με ώθησαν να παρακολουθήσω το συγκεκριμένο μεταπτυχιακό καθώς και στον προϊστάμενο και φίλο Βασίλη Θεολογίδη που με στήριξε όλο αυτό το διάστημα.

Θεοδωρόπουλος Ιωάννης

Η παρούσα διπλωματική εργασία ολοκληρώθηκε μετά από επίμονες και επίπονες προσπάθειες, σε ένα άκρως ενδιαφέρον αλλά άγνωστο γνωστικό αντικείμενο σε εμάς, όπως αυτό της επικοινωνίας μεταξύ έξυπνων κάδων με σκοπό την επεξεργασία και αποθήκευση των τιμών μέτρησης εικονικών αισθητήρων του κάθε κάδου. Την προσπάθειά μας αυτή υποστήριξαν οι επιβλέποντες καθηγητές μας, τους οποίους θα θέλαμε να ευχαριστήσουμε δεόντως. Ένα μεγάλο ευχαριστώ στην σύζυγό μου Αρχοντία Μπατσή που με ώθησε να παρακολουθήσω το συγκεκριμένο μεταπτυχιακό και με στήριξε όλο αυτό το διάστημα μαζί με τα τέκνα μας Θωμά, Γεωργία. Μεγάλη «ευθύνη» στην όλη αυτή διαδρομή και βοήθεια για την επίτευξη του στόχου μου, είχε όλη η οικογένεια μου, οι γονείς μου Θωμάς, Ελένη και η μητέρα της συζύγου μου Γεωργία, στους οποίους αφιερώνω την επιτυχία αυτή. Ιδιαίτερη αφιέρωση στα τέκνα μου (Θωμά, Γεωργία) και εύχομαι η δια βίου μάθηση να τους είναι παράδειγμα προς μίμηση. Από τις ευχαριστίες δεν γίνεται να λείψει και ο φίλος και συνάδελφος Θεοδωρόπουλος Γιάννης που ήταν καταλυτικός παράγοντας για την επιτυχή κατάληξη του μεταπτυχιακού αυτού.

Καραγιαννάκης Στυλιανός

ΠΕΡΙΛΗΨΗ

Η διπλωματική εργασία αυτή αποτελεί ένα λιθαράκι στην δημιουργία ενός ολοκληρωμένου συστήματος συλλογής και επεξεργασίας πληροφοριών που αφορούν την διαχείριση έξυπνων κάδων στα πλαίσια ενός ευρύτερου πλάνου βελτιστοποίησης των λειτουργιών μια έξυπνης πόλης. Ασχολείται με την διερεύνηση των υποστηρικτικών τεχνολογιών εικονικοποίησης (virtualization) στην υπολογιστική νέφους (cloud computing), με έμφαση στην τεχνική της εικονικοποίησης σε επίπεδο λειτουργικού συστήματος (os-level virtualization) η οποία αποτελεί μια από τις πλέον σύγχρονες τάσεις. Το αντικείμενο της είναι η ανάπτυξη εφαρμογής έξυπνων κάδων απορριμμάτων που θα επικοινωνούν με ένα κεντρικό σημείο - κάδο και με το υπολογιστικό νέφος με χρήση τεχνολογιών μικροϋπηρεσιών και εικονικοποίησης βασισμένης σε περιέκτες.

Η πρώτη μικροϋπηρεσία θα παράγει τιμές από τους εικονικούς αισθητήρες και θα τις αποστέλλει στην κεντρική μικροϋπηρεσία. Η κεντρική μικροϋπηρεσία που θα διαχειρίζεται τον κεντρικό κάδο θα λαμβάνει αποφάσεις, θα αποθηκεύει τις πληροφορίες σε βάση δεδομένων ταχείας προσπέλασης (redis) και σε βάση δεδομένων μακράς διαρκείας (mongoDB) και ανάλογα την απόφαση θα αποστέλλει εντολή στην αρχική μικροϋπηρεσία για εκκίνηση ενεργοποιητή (actuator). Άλλη μικροϋπηρεσία θα εξυπηρετεί τον ρόλο της διεπαφής προγραμματισμού εφαρμογών (API) που θα αντλεί δεδομένα από την βάση δεδομένων mongoDB και θα τα παρέχει στην επόμενη μικροϋπηρεσία. Η τελευταία μικροϋπηρεσία θα εμφανίζει τα δεδομένα των κάδων σε όμορφο και λειτουργικό περιβάλλον ιστοσελίδας με δυνατότητες διαχείρισης τους.

Διερευνήθηκε-μελετήθηκε η χρήση των περιεκτών (containers) (ως υποστηρικτική υποδομή φιλοξενίας) σε συνδυασμό με την τεχνολογία ανάπτυξης εφαρμογών με την αρχιτεκτονική των μικροϋπηρεσιών σε επίπεδο περιέκτη, και αναπτύχθηκε και αξιολογήθηκε-μελετήθηκε ενδεικτική εφαρμογή-σενάριο χρήσης στο χώρο του Διαδικτύου των Αντικειμένων (IoT). Διερευνήθηκαν και παρουσιάζονται τα χαρακτηριστικά, η χρησιμότητα και η λειτουργικότητα των αισθητήρων που μπορούν να τοποθετηθούν σε έξυπνους κάδους. Διερευνήθηκαν επίσης και παρουσιάζονται συγκριτικά τα χαρακτηριστικά και δυνατότητες των κυριότερων εργαλείων-προϊόντων που διατίθενται στην αγορά για υποστήριξη-διαχείριση και ενορχήστρωση των παρεχόμενων υπηρεσιών (Kubernetes, Docker Swarm, Nomad).

ABSTRACT

This thesis explores the use of virtualization technologies in cloud computing, specifically the os-level virtualization technique, to develop a smart waste bin application that communicates with a central point and the cloud using microservices and container-based virtualization. The thesis aims to contribute to the development of an integrated information collection and processing system for smart waste bins as part of a broader plan to optimize the functions of a smart city.

The first microservice will generate values from virtual sensors and send them to the central microservice. The central microservice that manages the central bin will make decisions, store information in a high-speed database (Redis) and in a long-term database (MongoDB), and based on the decision, send a command to the initial microservice to start an actuator. Another microservice will serve as the application programming interface (API) interface that retrieves data from the MongoDB database and provides it to the next layer of the application. The final microservice will display the data of the bins in a beautiful and functional website environment with management capabilities.

The use of containers as a hosting infrastructure was explored and studied in combination with application development technology with microservices architecture at the container level, and an indicative use case scenario application was developed and evaluated in the Internet of Things (IoT) space. The characteristics, usefulness, and functionality of sensors that can be placed in smart bins were also explored and presented. Additionally, the characteristics and capabilities of the main tools and products available in the market to support, manage, and orchestrate the provided services (Kubernetes, Docker Swarm, Nomad) were also explored and comparatively presented.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Ανάπτυξη λογισμικού πληροφορικής και υποδομή υπολογιστικών συστημάτων.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Cloud Computing, υπολογιστική νέφους, redis, mongoDB, Kubernetes, Docker Swarm, Nomad, IoT, virtualization, εικονικοποίηση, εφαρμογή έξυπνων κάδων, μικροϋπηρεσία, IaaS, PaaS, SaaS, Containers, Microservices, Docker ,Orchestration, Ενορχήστρωση, API, Containers, Περιέκτες, Artificial intelligence, Application Programming Interface, DOM, DevOps, WSL, XaaS, XYTA, EaaS, Document Object Model, Διαχείριση Αποβλήτων, Waste management, Middleware, Ενδιάμεσο λογισμικό, Διαδίκτυο των Αντικειμένων, Internet of things, Ενορχήστρωση περιεκτών, Container Orchestration, Microservices Dockerization, GitHub, DockerHub, Linux, Ubuntu, Alpine, NodeJS, Express, Axios, Socket.io, Websocket, Remote Dictionary Server, PostMan, Nginx, Vue.js, Apptainer, Singularity, Fire sensor, Ultrasonic sensor, Temperature sensor, Visual Studio Code, Blockchain.

Πίνακας περιεχομένων

Περιεχόμενα

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

AI - Artificial intelligence

API - Application Programming Interface

Cloud Computing - Υπολογιστική Νέφους

DOM (Document Object Model) Μοντέλο Αντικειμένων Εγγράφου

DevOps - software development (Dev) and IT operations (Ops)

EaaS - Environment as a Service

EBITDA - Earnings Before Interest, Tax, Depreciation, and Amortization

HTTP - Hypertext Transfer Protocol

IaaS - Infrastructure as a Service

ICT Information and Communications Technology

IDC - International Data Group

IoT Internet of Things

IP - Internet Protocol

IPCC The Intergovernmental Panel on Climate Change

NIST - National Institute of Standards and Technology

NPM - Node Package Manager

PaaS - Platform as a service

SaaS - Software as a Service

SOA - service-oriented architecture

SPA - Single Page Web Applications

Virtualization - Εικονικοποίησης

VLAN Virtual Local Area Network

WSL - (Windows Subsystem for Linux)

XaaS - Anything as a service

ΑΣΑ Αστικά Στερεά Απόβλητα

ΧΥΤΑ - Χώροι Υγειονομικής Ταφής Απορριμμάτων

Ανάπτυξη Εφαρμογής Έξυπνων Κάδων με χρήση Τεχνολογιών Μικροϋπηρεσιών και Εικονικοποίησης βασισμένης σε Περιέκτες

1. Εισαγωγή-Υπόβαθρο

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της διπλωματικής εργασίας και γίνεται μια ιστορική αναδρομή γύρω από τις μεθόδους που έχουν παρουσιαστεί σε αυτήν την περιοχή.

1.1 Περιγραφή του αντικειμένου της διπλωματικής εργασίας

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η Ανάπτυξη Εφαρμογής Έξυπνων Κάδων αποκομιδής απορριμμάτων με χρήση Τεχνολογιών Μικροϋπηρεσιών και Εικονικοποίησης βασισμένης σε Περιέκτες. Θα γίνει διερεύνησης των υποστηρικτικών τεχνολογιών Εικονικοποίησης (virtualization) στην Υπολογιστική Νέφους (cloud computing). Δίνεται έμφαση στην τεχνική του os-level virtualization η οποία αποτελεί μια από τις πλέον σύγχρονες τάσεις στον τομέα της ανάπτυξης εφαρμογών πληροφορικής.

Θα διερευνηθεί-μελετηθεί η χρήση των περιεκτών (ως υποστηρικτική υποδομή φιλοξενίας) σε συνδυασμό με την τεχνολογία ανάπτυξης εφαρμογών με την αρχιτεκτονική των μικροϋπηρεσιών, και θα αναπτυχθούν και αξιολογηθούν-μελετηθούν ενδεικτικές εφαρμογές-σενάρια χρήσης στο χώρο του Διαδικτύου των Αντικειμένων (IoT).

Θα προσπαθήσουμε βάσει των νέων και τελευταία εξελιγμένων τεχνολογιών της πληροφορικής να υλοποιήσουμε μια εφαρμογή η όποια μέσω της εικονικοποίησης των μικροϋπηρεσιών και των περιεκτών θα λειτουργεί διαδραστικά στο IoT και θα μπορεί να δώσει πληροφορίες και να πάρει αποφάσεις για την εκτέλεση κάποιων ενεργειών. Όλη η εφαρμογή θα επιχειρηθεί σε μια ομάδα από κάδους αποκομιδής οικιακών απορριμμάτων με την χρήση των προαναφερθέντων τεχνολογιών και θα παρουσιαστεί ένα δείγμα της λειτουργίας τους και των ενεργειών τους.

1.2 Ιστορική αναδρομή

1.2.1 Έξυπνες Πόλεις - Smart Cities

Έξυπνη πόλη είναι ένα μέρος όπου τα παραδοσιακά δίκτυα και οι υπηρεσίες γίνονται πιο αποτελεσματικά με τη χρήση ψηφιακών λύσεων προς όφελος των κατοίκων και των επιχειρήσεων.

Η έξυπνη πόλη υπερβαίνει τη χρήση ψηφιακών τεχνολογιών για καλύτερη χρήση των πόρων και λιγότερες εκπομπές το οποίο όπως καταλαβαίνουμε σημαίνει αναβαθμισμένες εγκαταστάσεις ύδρευσης και διάθεσης απορριμμάτων και πιο αποτελεσματικούς τρόπους φωτισμού και θέρμανσης κτιρίων καθώς και εξυπνότερα δίκτυα αστικών μεταφορών. Επίσης ισοδυναμεί με μια πιο διαδραστική και ανταποκρινόμενη διοίκηση της πόλης με ασφαλέστερους δημόσιους χώρους και κάλυψη των αναγκών ενός γηράσκοντος πληθυσμού.

Ένας δήμος Έξυπνη πόλη χρησιμοποιεί τεχνολογία πληροφοριών και επικοινωνιών ΤΠΕ (ICT) για τη βελτίωση της λειτουργικής αποτελεσματικότητας, την ανταλλαγή πληροφοριών με το κοινό και την παροχή καλύτερης ποιότητας κρατικών υπηρεσιών και ευημερίας των πολιτών.

Ο κύριος στόχος μιας έξυπνης πόλης είναι η βελτιστοποίηση των λειτουργιών της πόλης και η προώθηση της οικονομικής ανάπτυξης βελτιώνοντας παράλληλα την ποιότητα ζωής των πολιτών χρησιμοποιώντας έξυπνες τεχνολογίες και ανάλυση δεδομένων. Η αξία έγκειται στον τρόπο χρήσης αυτής της τεχνολογίας και όχι απλώς στο πόση τεχνολογία είναι διαθέσιμη.

Η εξυπνάδα μιας πόλης προσδιορίζεται χρησιμοποιώντας ένα σύνολο χαρακτηριστικών, όπως:

- Μια υποδομή που βασίζεται στην τεχνολογία
- Περιβαλλοντικές πρωτοβουλίες
- Αποτελεσματικά και εξαιρετικά λειτουργικά μέσα μαζικής μεταφοράς
- Σχέδια πόλης με αυτοπεποίθηση και προοδευτικά
- Άνθρωποι ικανοί να ζουν και να εργάζονται μέσα στην πόλη, χρησιμοποιώντας τους πόρους της

Εφαρμογές του IoT σε Έξυπνες Πόλεις που μπορούν να συμβάλουν σε όλα αυτά που αναφερόμαστε και θα είχαν έναν θετικό αντίκτυπο στους πολίτες, την καθημερινότητα αλλά και την οικολογία είναι κάποιες όπως οι:

- Έξυπνη Υποδομή.
- Διαχείριση Ποιότητας Αέρα.
- Διαχείριση Κυκλοφορίας.
- Έξυπνο πάρκινγκ.
- Έξυπνη διαχείριση απορριμμάτων.

Λόγω του εύρους των τεχνολογιών που έχουν εφαρμοστεί με τον όρο έξυπνη πόλη, είναι δύσκολο να καταλήξουμε σε έναν ακριβή ορισμό της έξυπνης πόλης. Οι Deakin και Al Waer [25] απαριθμούν τέσσερις παράγοντες που συμβάλλουν στον ορισμό της έξυπνης πόλης:

- Η εφαρμογή ενός ευρέος φάσματος ηλεκτρονικών και ψηφιακών τεχνολογιών σε κοινότητες και πόλεις.
- Η χρήση των ΤΠΕ για τη μετατροπή της ζωής και του εργασιακού περιβάλλοντος στην περιοχή.
- Η ενσωμάτωση τέτοιων Τεχνολογιών Πληροφορικής και Επικοινωνιών στα κυβερνητικά συστήματα.
- Η ενσωμάτωση πρακτικών που φέρνει κοντά τις ΤΠΕ και τους ανθρώπους για να ενισχύσουν την καινοτομία και τη γνώση που προσφέρουν.

Ο Deakin [25] ορίζει την έξυπνη πόλη ως αυτή που χρησιμοποιεί τις ΤΠΕ για να καλύψει τις απαιτήσεις της αγοράς (τους πολίτες της πόλης) και δηλώνει ότι η συμμετοχή της κοινότητας στη διαδικασία είναι απαραίτητη για μια έξυπνη πόλη. Μια έξυπνη πόλη θα ήταν επομένως μια πόλη που όχι μόνο διαθέτει τεχνολογία ΤΠΕ σε συγκεκριμένες περιοχές, αλλά έχει επίσης εφαρμόσει αυτήν την τεχνολογία με τρόπο που επηρεάζει θετικά την τοπική κοινότητα.

Το Smart Cities Marketplace δημιουργήθηκε από τη συγχώνευση δύο προηγούμενων πλατφορμών, του «Marketplace of the European Innovation Partnership on Smart Cities and Communities (EIP-SCC Marketplace)» και του «Smart Cities Information System (SCIS)».

Είναι ένα σημαντικό εγχείρημα που αλλάζει την αγορά και στοχεύει να φέρει κοντά πόλεις, βιομηχανίες, ΜΜΕ, επενδυτές, τράπεζες, ερευνητές και πολλούς άλλους παράγοντες έξυπνων πόλεων. Έχει πολλούς ακόλουθους από όλη την Ευρώπη και όχι μόνο, πολλοί από τους οποίους έχουν εγγραφεί ως μέλη. Κοινοί στόχοι τους είναι η βελτίωση της ποιότητας ζωής των πολιτών, η αύξηση της ανταγωνιστικότητας των ευρωπαϊκών πόλεων και της βιομηχανίας καθώς και η επίτευξη ευρωπαϊκών στόχων για την ενέργεια και το κλίμα.

1.2.2 Διαχείριση των Αποβλήτων - Waste management

Ορισμός:

Η διαχείριση ή η διάθεση αποβλήτων περιλαμβάνει τις διαδικασίες και τις ενέργειες που απαιτούνται για τη διαχείριση των αποβλήτων από την έναρξή τους (τοποθέτηση σε κάδο) έως την τελική τους διάθεσή. Αυτό περιλαμβάνει τη συλλογή, τη μεταφορά, την επεξεργασία και τη διάθεση των αποβλήτων, αλλά και την παρακολούθηση και τη ρύθμιση της διαδικασίας διαχείρισης των αποβλήτων και τους νόμους, τις τεχνολογίες, τους οικονομικούς μηχανισμούς που σχετίζονται με τα απόβλητα[1].

Τα απόβλητα μπορεί να είναι στερεά, υγρά ή αέρια και κάθε τύπος έχει διαφορετικές μεθόδους διάθεσης και διαχείρισης. Η διαχείριση αποβλήτων ασχολείται με όλα τα είδη αποβλήτων, συμπεριλαμβανομένων των βιομηχανικών, βιολογικών, οικιακών, αστικών, οργανικών, βιοϊατρικών, ραδιενεργών αποβλήτων. Σε ορισμένες περιπτώσεις, τα απόβλητα μπορούν να αποτελέσουν απειλή για την ανθρώπινη υγεία. Τα θέματα προστασίας της υγείας συνδέονται σε όλη τη διαδικασία διαχείρισης απορριμμάτων. Θέματα υγείας μπορεί επίσης να προκύψουν έμμεσα ή άμεσα. Άμεσα, μέσω του χειρισμού στερεών αποβλήτων, και έμμεσα μέσω της κατανάλωσης νερού, εδάφους και τροφίμων. Τα απόβλητα παράγονται από ανθρώπινη δραστηριότητα, για παράδειγμα, την εξόρυξη και την επεξεργασία πρώτων υλών. Η διαχείριση των απορριμμάτων αποσκοπεί στη μείωση των αρνητικών επιπτώσεων των αποβλήτων στην ανθρώπινη υγεία, το περιβάλλον, τους πλανητικούς πόρους και την αισθητική.

Στόχος της διαχείρισης αποβλήτων είναι να μειωθούν οι επικίνδυνες επιπτώσεις τέτοιων αποβλήτων στο περιβάλλον και την ανθρώπινη υγεία. Ένα μεγάλο μέρος της διαχείρισης απορριμμάτων ασχολείται με τα αστικά στερεά απόβλητα, τα οποία δημιουργούνται από βιομηχανική, εμπορική και οικιακή δραστηριότητα.

Οι πρακτικές διαχείρισης αποβλήτων δεν είναι ομοιόμορφες μεταξύ των χωρών (ανεπτυγμένες και αναπτυσσόμενες χώρες). Περιφέρειες (αστικές και αγροτικές περιοχές) και οι οικιστικοί και βιομηχανικοί τομείς μπορούν όλοι να υιοθετήσουν διαφορετικές προσεγγίσεις.

Η σωστή διαχείριση των απορριμμάτων είναι σημαντική για την οικοδόμηση βιώσιμων και βιώσιμων πόλεων, αλλά παραμένει πρόκληση για πολλές αναπτυσσόμενες χώρες και πόλεις. Έχει διαπιστωθεί ότι η αποτελεσματική διαχείριση των απορριμμάτων είναι σχετικά δαπανηρή και συνήθως περιλαμβάνει το 20%-50% των δημοτικών προϋπολογισμών. Η λειτουργία αυτής

της βασικής δημοτικής υπηρεσίας απαιτεί ολοκληρωμένα συστήματα που είναι αποτελεσματικά, βιώσιμα και κοινωνικά υποστηριζόμενα. Ένα μεγάλο μέρος των πρακτικών διαχείρισης αποβλήτων ασχολείται με τα αστικά στερεά απόβλητα (ΑΣΑ) τα οποία αποτελούν το μεγαλύτερο μέρος των αποβλήτων που δημιουργούνται από οικιακές, βιομηχανικές και εμπορικές δραστηριότητες. Σύμφωνα με τη Διακυβερνητική Επιτροπή για την Κλιματική Αλλαγή (IPCC), τα αστικά στερεά απόβλητα αναμένεται να φτάσουν περίπου τα 3,4 Gt έως το 2050. Ωστόσο, οι πολιτικές και η νομοθεσία μπορούν να μειώσουν την ποσότητα των απορριμμάτων που παράγονται σε διάφορες περιοχές και πόλεις του κόσμου. Τα μέτρα διαχείρισης αποβλήτων περιλαμβάνουν μέτρα για ολοκληρωμένους τεχνοοικονομικούς μηχανισμούς κυκλικής οικονομίας, αποτελεσματικές εγκαταστάσεις διάθεσης, έλεγχο εξαγωγών και εισαγωγών και βέλτιστο βιώσιμο σχεδιασμό των προϊόντων που παράγονται.

Οι χαρακτηριστικές δραστηριότητες της διαχείρισης απορριμμάτων περιλαμβάνουν:

1. Συλλογή, μεταφορά, επεξεργασία και διάθεση αποβλήτων,
2. Έλεγχος, παρακολούθηση και ρύθμιση της παραγωγής, συλλογής, μεταφοράς, επεξεργασίας και διάθεσης αποβλήτων και
3. Πρόληψη της παραγωγής αποβλήτων μέσω τροποποιήσεων στη διαδικασία, επαναχρησιμοποίησης και ανακύκλωσης.

Θέλοντας λοιπόν να ορίσουμε τον σκοπό της διαχείρισης απορριμμάτων θα λέγαμε πως η διαχείριση των απορριμμάτων αποτελεί σημαντικό στοιχείο της προστασίας του περιβάλλοντος και σκοπός της είναι να παρέχει υγιεινή, αποτελεσματική και οικονομική αποθήκευση στερεών αποβλήτων, συλλογή, μεταφορά και επεξεργασία ή διάθεση αποβλήτων χωρίς να μολύνει την ατμόσφαιρα, το έδαφος ή το υδάτινο σύστημα. Θα μπορούσαμε να συμπεριλάβουμε ακόμα και την απόρριψη προς το διάστημα των αποβλήτων πράγμα το οποίο ακούμε συχνά τα τελευταία χρόνια.

Υπάρχουν διάφοροι τύποι διαχείρισης απορριμμάτων κάποιοι από αυτούς περιλαμβάνουν: Χώροι Υγειονομικής Ταφής Απορριμμάτων (ΧΥΤΑ) – Αποτέφρωση - Συμπίεση απορριμμάτων – Κομποστοποίηση.

1.3 Δομή Μεταπτυχιακής Εργασίας

Η μεθοδολογική προσέγγιση για την εκπόνηση της παρούσας εργασίας είναι η εξής. Αρχικά διερευνήθηκε το εύρος των εφαρμογών που περιλαμβάνουν υλοποιήσεις έξυπνων κάδων οι οποίες περιλαμβάνουν 3 μέρη, 1ον το κύκλωμα του μικροελεγκτή και των αισθητήρων, 2ον την επικοινωνία μεταξύ των κάδων και του κέντρου ελέγχου και 3ον την αξιοποίηση τεχνολογιών τεχνητής νοημοσύνης ή επιχειρηματικής νοημοσύνης για την εκμετάλλευση από άλλες εφαρμογές των δεδομένων. Στην συνέχεια διερευνήθηκαν οι μέθοδοι επικοινωνίας μέσω πρωτοκόλλων του Web και ποια εργαλεία θα υλοποιήσουν τις τεχνολογίες αυτές. Ύστερα έγινε σχεδιασμός των τεχνικών χαρακτηριστικών που θα διέπουν την εφαρμογή που πρόκειται να υλοποιηθεί. Επίσης δημιουργήθηκε το περιβάλλον ανάπτυξης και ελέγχου του κώδικα της εφαρμογής. Διερευνήθηκε η χρήση των περιεκτών ως υποστηρικτική υποδομή φιλοξενίας και συνδυάστηκε με την τεχνολογία ανάπτυξης εφαρμογών και την αρχιτεκτονική των μικροϋπηρεσιών. Τέλος δημιουργήθηκε εφαρμογή επίδειξης που χρησιμοποιεί τις αναφερόμενες τεχνολογίες – εργαλεία και εξομοιώνει την παραγωγή και μεταφορά τιμών αισθητήρων από ένα κάδο σε ένα κεντρικό σημείο (άλλος κάδος) καθώς και την αποθήκευση και εμφάνισής τους σε ένα ευανάγνωστο περιβάλλον διαχείρισης.

Στο κεφάλαιο 1 αναφέρονται η περιγραφή και το αντικείμενο που εκπονείται με αυτή την διπλωματική εργασία, η ιστορική αναδρομή στα αντικείμενα αυτά καθώς και η δομή της εργασίας με αναφορά στις τεχνολογίες που θα εφαρμοστούν και θα υλοποιηθούν.

Στο κεφάλαιο 2 αναφέρονται οι βασικές τεχνολογίες που πρέπει να γνωρίζουμε για την εργασία και περιγράφονται οι ιδιότητες τους στο βάθος που είναι αναγκαίο για να μπορέσει ο αναγνώστης να παρακολουθήσει την πορεία της εκπόνησης.

Στο κεφάλαιο 3 παρουσιάζονται αναλυτικά και στο βάθος που απαιτείται τα εργαλεία και οι εφαρμογές που χρησιμοποιούνται για την ανάπτυξη της εφαρμογής επίδειξης, όπως αναφέρεται στην περιγραφή του αντικειμένου (Υποκεφάλαιο 1.1), αλλά και ο τρόπος λειτουργίας τους, με την μεταξύ τους διασύνδεση - πάντρεμα.

Στο κεφάλαιο 4 παρουσιάζεται ο σχεδιασμός της εφαρμογής που υλοποιήθηκε με τα απαραίτητα για αυτή εργαλεία καθώς και το λογικό διάγραμμα που εξηγεί τον αλγόριθμο της λειτουργίας της. Επίσης αναλύονται οι λειτουργίες και τα χαρακτηριστικά των τριών αισθητήρων που χρησιμοποιούνται και παρατίθεται πίνακας με τα κύρια χαρακτηριστικά των υπόλοιπων αισθητήρων που μπορούν να ενσωματωθούν σε ένα έξυπνο κάδο.

Στο κεφάλαιο 5 παρουσιάζεται η υλοποίηση της εφαρμογής που αποτελείται από δύο προγράμματα παραγωγής και εμφάνισης δεδομένων, δύο βάσεις δεδομένων στη μνήμη και σε αποθηκευτικό χώρο, ένα πρόγραμμα διαχείρισης της επικοινωνίας και της μεταφοράς δεδομένων προς και από τις βάσεις δεδομένων και λήψης αποφάσεων, και ένα πρόγραμμα διεπαφής προγραμματισμού εφαρμογών για την διαχείριση και διοχέτευση των δεδομένων της βάσης.

Στο κεφάλαιο 6 παρατίθενται τα ιδιαιτέρως ενδιαφέροντα συμπεράσματα που προέκυψαν καθώς η δημιουργία εφαρμογής επικοινωνίας έξυπνων κάδων είναι ένα δύσκολο εγχείρημα γιατί συνδυάζει πολλές διαφορετικές τεχνολογίες πληροφορικής που πρέπει να παντρευτούν αποτελεσματικά με επιτυχία. Επίσης διατυπώνονται τα επόμενα βήματα που προκύπτουν ως μελλοντικές δράσεις.

Ακολουθεί το παράρτημα με τους πίνακες εντολών και οι πηγές βιβλιογραφίας, κώδικα και εικόνων.

1.4 Αναφορά τεχνολογιών που θα χρησιμοποιηθούν

Για την εκπόνηση της εργασίας μας, χρειάζεται η γνώση συγκεκριμένων τεχνολογιών, σχετικά πρόσφατα ερευνημένων και δοσμένων προς χρήση στο ευρύ κοινό, προς γνώση της ανθρωπότητας οι οποίες πραγματικά μας ωφελούν και μας πάνε βήματα μπροστά, εξελίσσοντας τον τρόπο ζωής και την καθημερινότητα μας προσπαθώντας να εξαλείψουν κάθε δυσχέρεια που μπορεί να υπάρξει από τον τρόπο επικοινωνίας μας μέχρι και την ανάπτυξη του κόσμου μας. Οι πιο σημαντικές και βασικές τεχνολογίες που αναφέραμε παραπάνω είναι οι εξής:

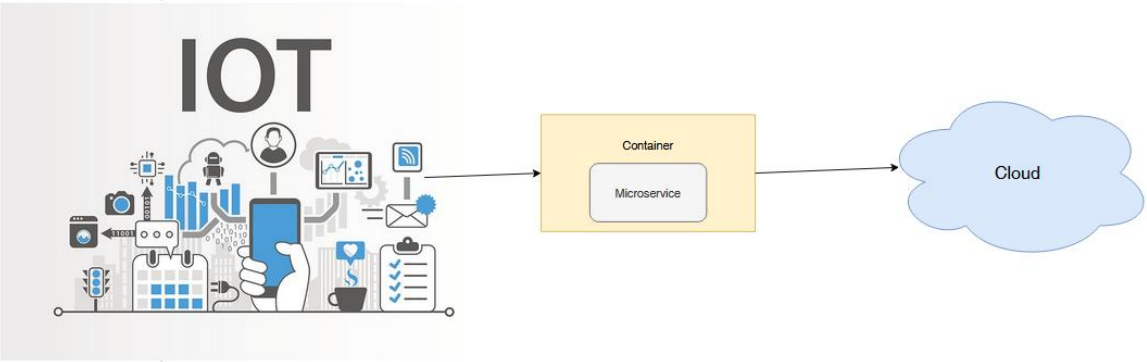
- IoT
- Cloud Computing
- Containers – Περιέκτες
- Virtualization - Εικονικοποίηση
- Orchestration - Ενορχήστρωση
- Microservices
- Middleware - Χρήση ενδιάμεσου λογισμικού.

Σε παρακάτω κεφάλαιο θα δούμε αναλυτικά όσα γνωρίζουμε για τη κάθε μια τεχνολογία ξεχωριστά αλλά και τα εργαλεία τα οποία θα χρησιμοποιήσουμε βάση των εν λόγω τεχνολογιών.

2. Θεωρία – Περιγραφή τεχνολογιών

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναφερθούμε στις τεχνολογίες και θα περιγράψουμε την θεωρία που τις διέπει οι οποίες χρησιμοποιήθηκαν σε αυτή την διπλωματική εργασία. Θα ξεκινήσουμε με την γενική περιγραφή των τεχνολογιών, ώστε στην συνέχεια αφού ο αναγνώστης θα έχει αποκτήσει ένα υπόβαθρο θα προχωρήσουμε σε πιο εξειδικευμένες γνώσεις πάνω στα γνωστικά αντικείμενα που θα χρησιμοποιηθούν.



Σχήμα 2.1: Internet of things και Cloud

2.2 Διαδίκτυο των Αντικειμένων - Internet of things

Το Διαδίκτυο των αντικειμένων (Internet of things - IoT) περιγράφει φυσικά αντικείμενα (ή ομάδες τέτοιων αντικειμένων) με αισθητήρες, ικανότητα επεξεργασίας, λογισμικό και άλλες τεχνολογίες που συνδέουν και ανταλλάσσουν δεδομένα με άλλες συσκευές και συστήματα μέσω του Διαδικτύου ή άλλων δικτύων επικοινωνιών. Το Διαδίκτυο των αντικειμένων έχει θεωρηθεί λανθασμένη ονομασία επειδή οι συσκευές δεν χρειάζεται να είναι συνδεδεμένες στο δημόσιο Διαδίκτυο, πρέπει μόνο να είναι συνδεδεμένες σε ένα δίκτυο και να μπορούν να διευθυνσιοδοτηθούν μεμονωμένα[2].. Ένα πράγμα στο διαδίκτυο των αντικειμένων μπορεί να είναι ένα άτομο με εμφύτευμα παρακολούθησης καρδιάς, ένα ζώο φάρμας με αναμεταδότη βιοτσίπ, ένα αυτοκίνητο που έχει ενσωματωμένους αισθητήρες για να ειδοποιεί τον οδηγό όταν η πίεση των ελαστικών είναι χαμηλή ή οποιοδήποτε άλλο φυσικό ή ανθρωπογενές αντικείμενο

στο οποίο μπορεί να εκχωρηθεί μια διεύθυνση Πρωτοκόλλου Διαδικτύου (IP) και μπορεί να μεταφέρει δεδομένα μέσω δικτύου.[3] Το IoT θέλει τα συστήματα ή τις «μηχανές», να αλληλοεπιδρούν μεταξύ τους και, να λαμβάνουν αυτόματα σε πολλές περιπτώσεις, μία σειρά αποφάσεων ακόμα και ζωτικής σημασίας. Ένα καλό παράδειγμα είναι το εμφύτευμα καρδίας που προαναφέραμε σε συνδυασμό με κάποια εντολή του κινητού τηλεφώνου του ασθενή προς κάποιον οικείο του η και ακόμα με κλήση προς τον ιατρό του!

Όλο και περισσότερο, οι οργανισμοί σε διάφορους κλάδους χρησιμοποιούν το IoT για να λειτουργούν πιο αποτελεσματικά, να κατανοούν καλύτερα τους πελάτες για να παρέχουν βελτιωμένη εξυπηρέτηση πελατών, να βελτιώνουν τη λήψη αποφάσεων και να αυξάνουν την αξία της επιχείρησης.[4] Το πεδίο έχει εξελιχθεί λόγω της σύγκλισης πολλαπλών τεχνολογιών, συμπεριλαμβανομένων των απανταχού υπολογιστών, των αισθητήρων εμπορευμάτων, των ολοένα και πιο ισχυρών ενσωματωμένων συστημάτων και της μηχανικής μάθησης. να ενεργοποιήσουν ανεξάρτητα και συλλογικά το Διαδίκτυο των αντικειμένων. Στην καταναλωτική αγορά, η τεχνολογία IoT είναι πιο συνώνυμη με προϊόντα που σχετίζονται με την έννοια του «έξυπνου σπιτιού», συμπεριλαμβανομένων συσκευών (όπως φωτιστικά, θερμοστάτες, συστήματα οικιακής ασφάλειας, κάμερες και άλλες οικιακές συσκευές) που υποστηρίζουν ένα ή πιο κοινά οικοσυστήματα και μπορεί να ελεγχθεί μέσω συσκευών που σχετίζονται με αυτό το οικοσύστημα, όπως έξυπνα κινητά τηλέφωνα και έξυπνα ηχεία.[5] Το IoT χρησιμοποιείται επίσης σε συστήματα υγειονομικής περίθαλψης.

Υπάρχουν πολλές ανησυχίες σχετικά με τους κινδύνους στην ανάπτυξη των τεχνολογιών και προϊόντων IoT, ειδικά στους τομείς της ιδιωτικής ζωής και της ασφάλειας, και κατά συνέπεια, έχουν ξεκινήσει ήδη κινήσεις της βιομηχανίας και της κυβέρνησης για την αντιμετώπιση αυτών των ανησυχιών, συμπεριλαμβανομένης της ανάπτυξης διεθνών και τοπικών προτύπων, κατευθυντήριων γραμμών , και κανονιστικά πλαίσια.

2.2.1 Τρόπος λειτουργίας του IoT

Ένα «οικοσύστημα» IoT αποτελείται από έξυπνες συσκευές με δυνατότητα σύνδεσης στον ιστό, που χρησιμοποιούν ενσωματωμένα συστήματα, όπως επεξεργαστές, αισθητήρες και υλικό επικοινωνίας, για τη συλλογή, αποστολή και δράση σε δεδομένα που αποκτούν από το περιβάλλον τους. Οι συσκευές IoT μοιράζονται τα δεδομένα αισθητήρων που συλλέγουν συνδέοντας μια πύλη IoT ή άλλη συσκευή αιχμής όπου τα δεδομένα είτε αποστέλλονται στο cloud για ανάλυση ή ανάλυση τοπικά. Μερικές φορές, αυτές οι συσκευές επικοινωνούν με

άλλες σχετικές συσκευές και ενεργούν βάσει των πληροφοριών που λαμβάνουν η μία από την άλλη. Οι συσκευές κάνουν το μεγαλύτερο μέρος της εργασίας χωρίς ανθρώπινη παρέμβαση, αν και οι άνθρωποι μπορούν να αλληλοεπιδράσουνε με τις συσκευές - για παράδειγμα, για να τις ρυθμίσουν, να τους δώσουν οδηγίες ή να αποκτήσουν πρόσβαση στα δεδομένα. Τα πρωτόκολλα συνδεσιμότητας, δικτύωσης και επικοινωνίας που χρησιμοποιούνται με αυτές τις συσκευές με δυνατότητα web εξαρτώνται σε μεγάλο βαθμό από τις συγκεκριμένες εφαρμογές IoT που αναπτύσσονται.

Το IoT μπορεί επίσης να κάνει χρήση της τεχνητής νοημοσύνης (AI) και της μηχανικής μάθησης για να διευκολύνει και να κάνει πιο δυναμικές τις διαδικασίες συλλογής δεδομένων

2.2.2 Σημαντικότητα του IoT

Το Διαδίκτυο των αντικειμένων βοηθά τους ανθρώπους να ζουν και να εργάζονται πιο έξυπνα, καθώς και να αποκτούν τον απόλυτο έλεγχο της ζωής τους. Εκτός από την προσφορά έξυπνων συσκευών για την αυτοματοποίηση των σπιτιών, το IoT είναι απαραίτητο για τις επιχειρήσεις. Το IoT παρέχει στις επιχειρήσεις μια ματιά σε πραγματικό χρόνο για το πώς λειτουργούν πραγματικά τα συστήματά τους, παρέχοντας πληροφορίες για τα πάντα, από την απόδοση των μηχανών έως τις λειτουργίες της αλυσίδας εφοδιασμού.

Το IoT δίνει τη δυνατότητα στις εταιρείες να αυτοματοποιούν τις διαδικασίες και να μειώνουν το κόστος εργασίας. Επίσης, περιορίζει τα απόβλητα και βελτιώνει την παροχή υπηρεσιών, καθιστώντας λιγότερο δαπανηρή την κατασκευή και την παράδοση αγαθών, καθώς και προσφέροντας διαφάνεια στις συναλλαγές των πελατών.

Ως εκ τούτου, το IoT είναι μια από τις πιο σημαντικές τεχνολογίες της καθημερινής ζωής και θα συνεχίσει να αυξάνεται καθώς περισσότερες επιχειρήσεις συνειδητοποιούν τις δυνατότητες των συνδεδεμένων συσκευών να τις διατηρήσουν ανταγωνιστικές.

Το διαδίκτυο των αντικειμένων προσφέρει πολλά οφέλη στους οργανισμούς. Ορισμένα πλεονεκτήματα αφορούν συγκεκριμένα τον εκάστοτε κλάδο και ορισμένα ισχύουν σε πολλούς κλάδους. Μερικά από τα κοινά οφέλη του IoT επιτρέπουν στις επιχειρήσεις να:

- παρακολουθούν τις συνολικές επιχειρηματικές τους διαδικασίες·
- βελτίωση της εμπειρίας του πελάτη (CX)·
- εξοικονομούν χρόνο και χρήμα.

- ενίσχυση της παραγωγικότητας των εργαζομένων·
- ενσωμάτωση και προσαρμογή επιχειρηματικών μοντέλων·
- λήψη καλύτερων επιχειρηματικών αποφάσεων. και
- αποφέρει περισσότερα έσοδα.

Με το IoT οι εταιρείες ενθαρρύνονται στην επανεξέταση των τρόπων με τους οποίους προσεγγίζουν τις επιχειρήσεις τους και τους δίνει τα εργαλεία για να βελτιώσουν τις επιχειρηματικές τους στρατηγικές.

Γενικά, το IoT είναι πιο άφθονο στην κατασκευή, τις μεταφορές και τους οργανισμούς κοινής ωφέλειας, χρησιμοποιώντας αισθητήρες και άλλες συσκευές IoT. Ωστόσο, έχει βρει επίσης περιπτώσεις χρήσης για οργανισμούς στους τομείς της γεωργίας, των υποδομών και του οικιακού αυτοματισμού, οδηγώντας ορισμένους οργανισμούς στον ψηφιακό μετασχηματισμό.

Το IoT μπορεί να ωφελήσει τους αγρότες στη γεωργία διευκολύνοντας τη δουλειά τους. Οι αισθητήρες μπορούν να συλλέγουν δεδομένα σχετικά με τις βροχοπτώσεις, την υγρασία, τη θερμοκρασία και την περιεκτικότητα του εδάφους, καθώς και άλλους παράγοντες, που θα βοηθούσαν στην αυτοματοποίηση των γεωργικών τεχνικών όπως ποτίσματος, ραντίσματος η ακόμα και κάλυψης των εκτάσεων τους για αποφυγή παγετού.

Ένας παράγοντας στον οποίο μπορεί να βοηθήσει το IoT είναι η ικανότητα παρακολούθησης λειτουργιών γύρω από την υποδομή. Για παράδειγμα, θα μπορούσαν να χρησιμοποιηθούν αισθητήρες για την παρακολούθηση γεγονότων ή αλλαγών σε δομικά κτίρια, γέφυρες και άλλες υποδομές. Αυτό συνεπάγεται πλεονεκτήματα, όπως εξοικονόμηση κόστους, εξοικονόμηση χρόνου, αλλαγές στη ροή εργασιών ποιότητας ζωής και ροή εργασίας χωρίς χαρτιά.

Το IoT γενικά αγγίζει κάθε κλάδο, συμπεριλαμβανομένων των επιχειρήσεων στον τομέα της υγειονομικής περίθαλψης, των οικονομικών, της βιομηχανικής παραγωγής, στις τηλεπικοινωνίες, τις μεταφορές, την ενέργεια, του λιανικού εμπορίου και της κατασκευής.

2.3 Υπολογιστικό Νέφος - Cloud Computing

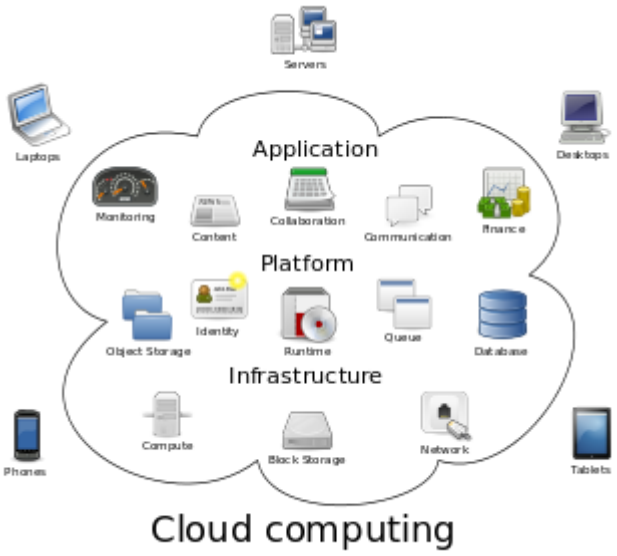
Cloud computing ονομάζεται η απαίτηση για διαθεσιμότητα πόρων συστήματος του υπολογιστή, και ειδικότερα αποθήκευσης δεδομένων και υπολογιστικής ισχύος, χωρίς την

άμεση διαχείριση από κάποιον χρήστη. Ο συγκεκριμένος όρος χρησιμοποιείται γενικά για να περιγραφεί η διάθεση κέντρων δεδομένων σε πολλούς χρήστες μέσω του Διαδικτύου.

Τα μεγάλα νέφη, που είναι και τα συνηθέστερα σήμερα, συχνά έχουν κατανεμημένες λειτουργίες σε πολλές τοποθεσίες από κεντρικούς διακομιστές. Σε περίπτωση που η σύνδεση με τον χρήστη είναι σχετικά κοντά, μπορεί να οριστεί διακομιστής άκρης.

Τα νέφη ενδέχεται να περιορίζονται σε έναν μόνο οργανισμό (εταιρικά νέφη) ή να είναι διαθέσιμα σε πολλούς οργανισμούς (δημόσιο νέφος).

Το cloud computing βασίζεται στην κοινή χρήση πόρων για την επίτευξη συνοχής και οικονομιών κλίμακας.[6]



Σχήμα 2.2: Cloud Computing

Το Cloud computing είναι η κατ' απαίτηση διαθεσιμότητα πόρων του συστήματος του υπολογιστή, ιδίως η αποθήκευση δεδομένων (αποθήκευση στο σύννεφο) και η υπολογιστική ισχύς, χωρίς άμεση ενεργή διαχείριση από τον χρήστη. Τα μεγάλα σύννεφα συχνά έχουν λειτουργίες κατανεμημένες σε πολλαπλές τοποθεσίες, με κάθε τοποθεσία να είναι ένα κέντρο δεδομένων. Το cloud computing βασίζεται στην κοινή χρήση πόρων για την επίτευξη συνοχής και συνήθως χρησιμοποιώντας ένα μοντέλο «πληρώνεις καθώς προχωράς» - ("pay-as-you-go") που μπορεί να βοηθήσει στη μείωση των κεφαλαιουχικών δαπανών, αλλά μπορεί επίσης να οδηγήσει σε απροσδόκητα λειτουργικά έξοδα για χρήστες που δεν γνωρίζουν. Με απλά λόγια, το cloud computing είναι η παροχή υπολογιστικών υπηρεσιών — συμπεριλαμβανομένων διακομιστών, αποθήκευσης, βάσεων δεδομένων, δικτύωσης,

λογισμικού, αναλυτικών στοιχείων και νοημοσύνης— μέσω του Διαδικτύου («το σύννεφο») για να προσφέρει ταχύτερη καινοτομία, ευέλικτους πόρους και οικονομικές κλίμακας. Συνήθως πληρώνετε μόνο για υπηρεσίες cloud που χρησιμοποιείτε, βοηθώντας σας να μειώσετε το λειτουργικό κόστος, να εκτελείτε την υποδομή σας πιο αποτελεσματικά και να «κλιμακώνεστε» καθώς αλλάζουν οι ανάγκες της επιχείρησής σας.

Σχήμα 2.3: Cloud Computing - παροχή υπολογιστικών υπηρεσιών

Οι υποστηρικτές των δημόσιων και υβριδικών cloud ισχυρίζονται ότι το cloud computing επιτρέπει στις εταιρείες να αποφεύγουν ή να ελαχιστοποιούν το αρχικό κόστος υποδομής πληροφορικής. Οι υποστηρικτές ισχυρίζονται επίσης ότι το cloud computing επιτρέπει στις επιχειρήσεις να ξεκινούν και να λειτουργούν πιο γρήγορα τις εφαρμογές τους, με βελτιωμένη διαχειρισιμότητα και λιγότερη συντήρηση, και ότι επιτρέπει στις ομάδες IT να προσαρμόζουν ταχύτερα τους πόρους για να ανταποκρίνονται στις κυμαινόμενες και απρόβλεπτες απαιτήσεις, παρέχοντας δυνατότητα υπολογισμού ριπής: υψηλή υπολογιστική ισχύ σε ορισμένες περιόδους αιχμής ζήτησης.

Σύμφωνα με την IDC, οι παγκόσμιες δαπάνες για υπηρεσίες υπολογιστικού νέφους έχουν φθάσει τα 706 δισεκατομμύρια δολάρια και αναμένεται να φτάσουν τα 1,3 τρισεκατομμύρια δολάρια έως το 2025. Ενώ η Gartner εκτίμησε ότι οι παγκόσμιες δαπάνες των τελικών χρηστών για υπηρεσίες cloud προβλέπεται να φτάσουν τα 600 δισεκατομμύρια δολάρια μέχρι το 2023.

Σύμφωνα με την McKinsey & Company έκθεση [26], μοχλοί βελτιστοποίησης κόστους στο cloud και υποθέσεις επιχειρηματικής χρήσης προσανατολισμένες στην αξία προβλέπουν περισσότερα από 1 τρισεκατομμύρια δολάρια σε τρέχον EBITDA σε όλες τις εταιρείες του Fortune 500 έως το 2030. Το 2022, περισσότερα από 1,3 τρισεκατομμύρια δολάρια σε επιχειρηματικές δαπάνες πληροφορικής διακυβεύονται από τη μετάβαση στο cloud, αυξάνοντας σε σχεδόν 1,8 τρισεκατομμύρια δολάρια το 2025.

Το cloud computing είναι μια μεγάλη αλλαγή από τον παραδοσιακό τρόπο που οι επιχειρήσεις σκέφτονται τους πόρους πληροφορικής. Οι επτά συνήθεις λόγοι για τους οποίους οι οργανισμοί στρέφονται στις υπηρεσίες υπολογιστικού νέφους:

1. Κόστος
2. Παγκόσμια κλίμακα
3. Ταχύτητα
4. Εκτέλεση
5. Παραγωγικότητα
6. Αξιοπιστία
7. Ασφάλεια

2.3.1 Μοντέλα υπηρεσιών

Αν και η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική υποστηρίζει το Τα Πάντα σαν Υπηρεσία "Everything as a service" (με τα ακρωνύμια EaaS ή XaaS, ή απλά aas), οι πάροχοι υπολογιστικού νέφους προσφέρουν τις "υπηρεσίες" τους σύμφωνα με διαφορετικά μοντέλα, εκ των οποίων τα τρία τυπικά μοντέλα ανά NIST είναι το Infrastructure as a Service (IaaS), το Platform as a Service (PaaS) και το Software as a Service (SaaS). Αυτά τα μοντέλα συχνά απεικονίζονται ως επίπεδα σε μια στοίβα υποδομή, πλατφόρμα και λογισμικό ως υπηρεσία, αλλά αυτά δεν χρειάζεται να σχετίζονται. Για παράδειγμα, μπορεί κανείς να παρέχει SaaS που υλοποιείται σε φυσικές μηχανές (μέταλλο), χωρίς να χρησιμοποιεί υποκείμενα επίπεδα PaaS ή IaaS, και αντιστρόφως μπορεί να τρέξει ένα πρόγραμμα στο IaaS και να έχει πρόσβαση απευθείας σε αυτό, χωρίς να το τυλίξει ως SaaS.

2.3.2 Η υποδομή ως υπηρεσία (IaaS)

Η "υποδομή ως υπηρεσία" (IaaS) αναφέρεται σε διαδικτυακές υπηρεσίες που παρέχουν API υψηλού επιπέδου που χρησιμοποιούνται για την περίληψη διαφόρων λεπτομερειών χαμηλού επιπέδου υποκείμενης υποδομής δικτύου, όπως φυσικούς υπολογιστικούς πόρους, τοποθεσία, κατάτμηση δεδομένων, κλιμάκωση, ασφάλεια, δημιουργία αντιγράφων ασφαλείας κ.λπ. Ο hypervisor εκτελεί τις εικονικές μηχανές ως επισκέπτες. Οι ομάδες hypervisor εντός του λειτουργικού συστήματος cloud μπορούν να υποστηρίξουν μεγάλο αριθμό εικονικών μηχανών και τη δυνατότητα κλιμάκωσης των υπηρεσιών προς τα πάνω και προς τα κάτω σύμφωνα με τις διαφορετικές απαιτήσεις των πελατών. Οι περιέκτες Linux τρέχουν σε μεμονωμένα διαμερίσματα ενός πυρήνα Linux που εκτελείται απευθείας στο φυσικό υλικό. Οι cgroups και οι χώροι ονομάτων Linux είναι οι υποκείμενες τεχνολογίες πυρήνα Linux που χρησιμοποιούνται για την απομόνωση, την ασφάλεια και τη διαχείριση των περιεκτών. Η αποθήκευση σε περιέκτες προσφέρει υψηλότερη απόδοση από την εικονικοποίηση, επειδή δεν υπάρχει επιβάρυνση hypervisor. Τα σύννεφα IaaS προσφέρουν συχνά πρόσθετους πόρους, όπως βιβλιοθήκη εικόνων δίσκου εικονικής μηχανής, αποθήκευση ακατέργαστου μπλοκ, αποθήκευση αρχείων ή αντικειμένων, τείχη προστασίας, εξισορροπητές φορτίου, διευθύνσεις IP, εικονικά τοπικά δίκτυα (VLAN) και πακέτα λογισμικού.

Ο ορισμός του NIST για το cloud computing περιγράφει το IaaS ως «όπου ο καταναλωτής μπορεί να αναπτύξει και να εκτελέσει αυθαίρετο λογισμικό, το οποίο μπορεί να περιλαμβάνει λειτουργικά συστήματα, αναπτυγμένες εφαρμογές και πιθανώς περιορισμένο έλεγχο επιλεγμένων στοιχείων δικτύου (π.χ. τείχη προστασίας κεντρικού υπολογιστή)».

Οι πάροχοι IaaS-cloud παρέχουν αυτούς τους πόρους κατ' απαίτηση από τις μεγάλες δεξαμενές εξοπλισμού τους που είναι εγκατεστημένοι σε κέντρα δεδομένων. Για συνδεσιμότητα ευρείας περιοχής, οι πελάτες μπορούν να χρησιμοποιήσουν είτε το Διαδίκτυο είτε τα cloud cloud (αποκλειστικά εικονικά ιδιωτικά δίκτυα). Για να αναπτύξουν τις εφαρμογές τους, οι χρήστες cloud εγκαθιστούν image λειτουργικού συστήματος και το λογισμικό εφαρμογής τους στην υποδομή cloud. Σε αυτό το μοντέλο, ο χρήστης του cloud διορθώνει και συντηρεί τα λειτουργικά συστήματα και το λογισμικό εφαρμογής. Οι πάροχοι cloud συνήθως χρεώνουν τις υπηρεσίες IaaS σε υπολογιστική βάση: το κόστος αντικατοπτρίζει τον αριθμό των πόρων που διατίθενται και καταναλώνονται.

2.3.3 Πλατφόρμα ως υπηρεσία (PaaS)

Ο ορισμός του NIST για την υπολογιστική νέφους, ορίζει την Πλατφόρμα ως Υπηρεσία (PaaS) ως: Η δυνατότητα που παρέχεται στον καταναλωτή να αναπτύσσει στην υποδομή cloud εφαρμογές που δημιουργήθηκαν ή αποκτήθηκαν από τους καταναλωτές που δημιουργήθηκαν χρησιμοποιώντας γλώσσες προγραμματισμού, βιβλιοθήκες, υπηρεσίες και εργαλεία που υποστηρίζονται από τον πάροχο. Ο καταναλωτής δεν διαχειρίζεται ούτε ελέγχει την υποκείμενη υποδομή cloud, συμπεριλαμβανομένου του δικτύου, των διακομιστών, των λειτουργικών συστημάτων ή της αποθήκευσης, αλλά έχει τον έλεγχο των εφαρμογών που έχουν αναπτυχθεί και πιθανώς τις ρυθμίσεις διαμόρφωσης για το περιβάλλον φιλοξενίας εφαρμογών.

Οι προμηθευτές PaaS προσφέρουν ένα περιβάλλον ανάπτυξης στους προγραμματιστές εφαρμογών. Ο πάροχος συνήθως αναπτύσσει εργαλειοθήκη και πρότυπα για την ανάπτυξη και κανάλια διανομής και πληρωμής. Στα μοντέλα PaaS, οι πάροχοι cloud παρέχουν μια υπολογιστική πλατφόρμα, που συνήθως περιλαμβάνει ένα λειτουργικό σύστημα, ένα περιβάλλον εκτέλεσης στη γλώσσα προγραμματισμού, τη βάση δεδομένων και τον διακομιστή Ιστού. Οι προγραμματιστές εφαρμογών αναπτύσσουν και εκτελούν το λογισμικό τους σε μια πλατφόρμα cloud αντί να αγοράζουν και να διαχειρίζονται απευθείας τα υποκείμενα στρώματα υλικού και λογισμικού. Με κάποιο PaaS, ο υποκείμενος υπολογιστής και οι πόροι αποθήκευσης κλιμακώνονται αυτόματα ώστε να ταιριάζουν με τη ζήτηση εφαρμογών, έτσι ώστε ο χρήστης του cloud να μην χρειάζεται να εκχωρεί πόρους με μη αυτόματο τρόπο.

Ορισμένοι πάροχοι ενοποίησης και διαχείρισης δεδομένων χρησιμοποιούν επίσης εξειδικευμένες εφαρμογές του PaaS ως μοντέλα παράδοσης δεδομένων. Τα παραδείγματα περιλαμβάνουν το iPaaS (Integration Platform as a Service) Πλατφόρμα Ολοκλήρωσης ως Υπηρεσία και το dPaaS (Data Platform as a Service) Πλατφόρμα Δεδομένων ως Υπηρεσία. Το iPaaS επιτρέπει στους πελάτες να αναπτύσσουν, να εκτελούν και να διοικούν τις ροές ενοποίησης. Σύμφωνα με το μοντέλο ενσωμάτωσης iPaaS, οι πελάτες καθοδηγούν την ανάπτυξη και την ανάπτυξη ενσωματώσεων χωρίς εγκατάσταση ή διαχείριση υλικού ή ενδιάμεσου λογισμικού. Το dPaaS παρέχει προϊόντα ολοκλήρωσης και διαχείρισης δεδομένων ως μια πλήρως διαχειριζόμενη υπηρεσία. Σύμφωνα με το μοντέλο dPaaS, ο πάροχος PaaS, όχι ο πελάτης, διαχειρίζεται την ανάπτυξη και την εκτέλεση προγραμμάτων δημιουργώντας εφαρμογές δεδομένων για τον πελάτη. Οι χρήστες dPaaS έχουν πρόσβαση σε δεδομένα μέσω εργαλείων οπτικοποίησης δεδομένων.

2.3.4 Λογισμικό ως υπηρεσία (SaaS)

Ο ορισμός του NIST για την υπολογιστική νέφους, ορίζει το λογισμικό ως υπηρεσία (SaaS) ως εξής: Η δυνατότητα που παρέχεται στον καταναλωτή είναι να χρησιμοποιεί τις εφαρμογές του παρόχου που εκτελούνται σε μια υποδομή cloud. Οι εφαρμογές είναι προσβάσιμες από διάφορες συσκευές-πελάτες είτε μέσω μιας λεπτής διεπαφής πελάτη, όπως ένα πρόγραμμα περιήγησης ιστού (π.χ. μέσω ηλεκτρονικού ταχυδρομείου που βασίζεται στον ιστό), είτε μέσω μιας διεπαφής προγράμματος. Ο καταναλωτής δεν διαχειρίζεται ούτε ελέγχει την υποκείμενη υποδομή cloud, συμπεριλαμβανομένων των δικτύων, των διακομιστών, των λειτουργικών συστημάτων, της αποθήκευσης ή ακόμη και των δυνατοτήτων μεμονωμένων εφαρμογών, με πιθανή εξαίρεση περιορισμένων ρυθμίσεων διαμόρφωσης εφαρμογών για συγκεκριμένους χρήστες.

Στο μοντέλο λογισμικού ως υπηρεσίας (SaaS), οι χρήστες αποκτούν πρόσβαση σε λογισμικό εφαρμογών και βάσεις δεδομένων. Οι πάροχοι υπηρεσιών νέφους cloud διαχειρίζονται την υποδομή και τις πλατφόρμες που εκτελούν τις εφαρμογές. Το SaaS αναφέρεται μερικές φορές ως "λογισμικό κατ' απαίτηση" και συνήθως τιμολογείται βάσει πληρωμής ανά χρήση ή με χρέωση συνδρομής. Στο μοντέλο SaaS, οι πάροχοι υπηρεσιών νέφους cloud εγκαθιστούν και λειτουργούν λογισμικό εφαρμογών στο νέφος cloud και οι χρήστες έχουν πρόσβαση στο λογισμικό από πελάτες υπηρεσιών νέφους cloud. Οι χρήστες του cloud δεν διαχειρίζονται την υποδομή και την πλατφόρμα cloud όπου εκτελείται η εφαρμογή. Αυτό εξαλείφει την ανάγκη εγκατάστασης και εκτέλεσης της εφαρμογής στους υπολογιστές του ίδιου του χρήστη νέφους cloud, κάτι που απλοποιεί τη συντήρηση και την υποστήριξη. Οι εφαρμογές νέφους cloud διαφέρουν από άλλες εφαρμογές ως προς την επεκτασιμότητα τους η οποία μπορεί να επιτευχθεί με την κλωνοποίηση εργασιών σε πολλαπλές εικονικές μηχανές κατά τη διάρκεια της εκτέλεσης για την κάλυψη της μεταβαλλόμενης ζήτησης εργασίας. Οι συσκευές εξισορρόπησης φορτίου κατανέμουν την εργασία στο σύνολο των εικονικών μηχανών. Αυτή η διαδικασία είναι διαφανής για τον χρήστη του cloud, ο οποίος βλέπει μόνο ένα μόνο σημείο πρόσβασης. Για να φιλοξενήσουν μεγάλο αριθμό χρηστών cloud, οι εφαρμογές cloud μπορούν να είναι πολυενοικιαζόμενες, πράγμα που σημαίνει ότι οποιοδήποτε μηχάνημα μπορεί να εξυπηρετεί περισσότερους από έναν οργανισμούς χρηστών cloud.

Το μοντέλο τιμολόγησης για εφαρμογές SaaS είναι συνήθως μια μηνιαία ή ετήσια πάγια χρέωση ανά χρήστη, επομένως οι τιμές γίνονται κλιμακούμενες και προσαρμόσιμες εάν προστεθούν ή αφαιρεθούν χρήστες σε οποιοδήποτε σημείο. Μπορεί επίσης να είναι δωρεάν.

Οι υποστηρικτές ισχυρίζονται ότι το SaaS δίνει σε μια επιχείρηση τη δυνατότητα να μειώσει το λειτουργικό κόστος πληροφορικής αναθέτοντας σε εξωτερικούς συνεργάτες τη συντήρηση και υποστήριξη υλικού και λογισμικού στον πάροχο cloud. Αυτό δίνει τη δυνατότητα στην επιχείρηση να ανακατανέμει το κόστος των λειτουργιών πληροφορικής μακριά από τις δαπάνες υλικού/λογισμικού και από τις δαπάνες προσωπικού, προς την επίτευξη άλλων στόχων. Επιπλέον, με τις εφαρμογές που φιλοξενούνται κεντρικά, οι ενημερώσεις μπορούν να κυκλοφορήσουν χωρίς να απαιτείται η εγκατάσταση νέου λογισμικού από τους χρήστες. Ένα μειονέκτημα του SaaS είναι η αποθήκευση των δεδομένων των χρηστών στον διακομιστή του παρόχου cloud. Ως αποτέλεσμα, θα μπορούσε να υπάρξει μη εξουσιοδοτημένη πρόσβαση στα δεδομένα. Παραδείγματα εφαρμογών που προσφέρονται ως SaaS είναι παιχνίδια και λογισμικό παραγωγικότητας όπως τα Έγγραφα Google και το Office Online. Οι εφαρμογές SaaS ενδέχεται να ενσωματωθούν με υπηρεσίες αποθήκευσης cloud ή φιλοξενίας αρχείων, όπως συμβαίνει με τα Έγγραφα Google που είναι ενσωματωμένα στο Google Drive και το Office Online με το OneDrive.

2.3.5 Αρχιτεκτονική υπολογιστικού νέφους

Η αρχιτεκτονική υπολογιστικής νέφους, cloud computing αναφέρεται στα στοιχεία και τα υποσυστήματα που απαιτούνται για αυτό. Αυτά τα στοιχεία αποτελούνται συνήθως από μια πλατφόρμα front-end (fat client, thin client, mobile device), back end platform (servers, storage), μια cloud-based παράδοση και ένα δίκτυο (Internet, Intranet, Intercloud). Σε συνδυασμό, αυτά τα στοιχεία αποτελούν την αρχιτεκτονική υπολογιστικού νέφους[7]

2.4 Περιέκτες - Containers

Η χρήση περιεκτών (containers) έχει γίνει μια μεγάλη τάση στην ανάπτυξη λογισμικού ως εναλλακτική λύση ή «σύντροφος» στην εικονικοποίηση. Περιλαμβάνει εγκλεισμό ή συσκευασία κώδικα λογισμικού και όλων των εξαρτήσεων του, ώστε να μπορεί να λειτουργεί ομοιόμορφα και με συνέπεια σε οποιαδήποτε υποδομή. Η τεχνολογία ωριμάζει γρήγορα, με αποτέλεσμα μετρήσιμα οφέλη για προγραμματιστές και ομάδες επιχειρήσεων, καθώς και συνολική υποδομή λογισμικού.[8]

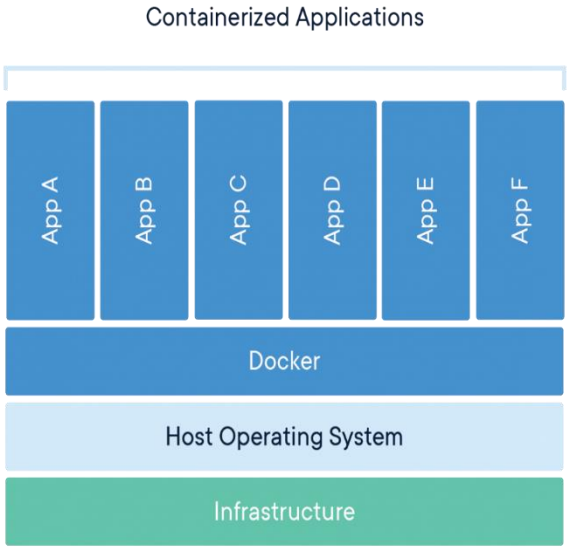
2.3. Εικονικές μηχανές έναντι περιεκτών Docker

Μια εικόνα περιέκτη (container) είναι ένα ελαφρύ, αυτόνομο, εκτελέσιμο πακέτο ενός λογισμικού που περιλαμβάνει όλα όσα χρειάζονται για την εκτέλεση του.

Το Docker είναι η υπηρεσία για την εκτέλεση πολλαπλών περιεκτών σε ένα μηχάνημα (κόμβος) που μπορεί να βρίσκεται σε μια μηχανή virtual ή σε μια φυσική μηχανή.

Μια εικονική μηχανή είναι ένα ολόκληρο λειτουργικό σύστημα (το οποίο συνήθως δεν είναι ελαφρύ).

Docker Containers



Σχήμα 2.4: Docker Containers

2.5 Εικονικοποίηση - Virtualization

Στην πληροφορική, η εικονικοποίηση αναφέρεται στην πράξη δημιουργίας μιας εικονικής (και όχι πραγματικής) έκδοσης κάποιου υλικού, συμπεριλαμβανομένων πλατφορμών υλικού εικονικού υπολογιστή, συσκευών αποθήκευσης και πόρων δικτύου υπολογιστών.

Η εικονικοποίηση είναι τεχνολογία που μας επιτρέπει να δημιουργούμε χρήσιμες υπηρεσίες πληροφορικής χρησιμοποιώντας πόρους που «παραδοσιακά» συνδέονται με το υλικό. Επιτρέπει την χρήση πλήρως της χωρητικότητας μιας φυσικής μηχανής κατανέμοντας τις δυνατότητές της σε πολλούς χρήστες ή περιβάλλοντα.

Πρακτικά έχουμε 3 φυσικούς διακομιστές με ατομικούς αποκλειστικούς σκοπούς. Ο ένας είναι διακομιστής αλληλογραφίας, ο άλλος είναι διακομιστής ιστού και ο τελευταίος εκτελεί εσωτερικές εφαρμογές παλαιού τύπου. Κάθε διακομιστής χρησιμοποιείται με περίπου 30% χωρητικότητα σε μόλις ένα κλάσμα των δυνατοτήτων λειτουργίας τους. Αλλά επειδή οι εφαρμογές παλαιού τύπου παραμένουν σημαντικές για τις εσωτερικές λειτουργίες, πρέπει να τις διατηρούμε και τον τρίτο διακομιστή που τις φιλοξενεί.

Συνήθως ήταν ευκολότερο και πιο αξιόπιστο να εκτελούνται μεμονωμένες εργασίες σε μεμονωμένους διακομιστές: 1 διακομιστής, 1 λειτουργικό σύστημα, 1 εργασία. Δεν ήταν εύκολο να δώσεις σε 1 διακομιστή πολλαπλούς εγκεφάλους. Αλλά με την εικονικοποίηση, μπορεί να χωρίσουμε τον διακομιστή αλληλογραφίας σε 2 μοναδικούς που μπορούν να χειριστούν ανεξάρτητες εργασίες, ώστε οι εφαρμογές παλαιού τύπου να μπορούν να μετεγκατασταθούν. Είναι το ίδιο υλικό, απλώς χρησιμοποιείτε περισσότερο από αυτό πιο αποτελεσματικά.

Έχοντας υπόψη την ασφάλεια, θα μπορούσαμε να χωρίσουμε ξανά τον πρώτο διακομιστή ώστε να μπορεί να χειριστεί μια άλλη εργασία—αυξάνοντας τη χρήση του από 30% σε 60% ή 90%. Μόλις γίνει αυτό, οι πλέον άδειοι διακομιστές θα μπορούσαν να επαναχρησιμοποιηθούν για άλλες εργασίες ή να αποσυρθούν εντελώς για να μειωθεί το κόστος ψύξης και συντήρησης.

Η εικονικοποίηση μπορεί να αυξήσει την ευελιξία και την επεκτασιμότητα της πληροφορικής, ενώ παράλληλα δημιουργεί σημαντική εξοικονόμηση κόστους. Μεγαλύτερη κινητικότητα φόρτου εργασίας, αυξημένη απόδοση και διαθεσιμότητα πόρων, αυτοματοποιημένες λειτουργίες είναι κάποια από τα πλεονεκτήματα της εικονικοποίησης που κάνουν το IT πιο απλό στη διαχείριση και λιγότερο δαπανηρό στην κατοχή και τη λειτουργία του.

Το λογισμικό που ονομάζεται Hypervisors διαχωρίζει τους φυσικούς πόρους από τα εικονικά περιβάλλοντα, τα πράγματα που χρειάζονται αυτούς τους πόρους. Οι υπερεπόπτες μπορούν να κάθονται πάνω από ένα λειτουργικό σύστημα (όπως σε φορητό υπολογιστή) ή να εγκατασταθούν απευθείας σε υλικό (όπως ένας διακομιστής), με τον οποίο οι περισσότερες επιχειρήσεις εικονικοποιούν. Οι υπερεπόπτες παίρνουν τους φυσικούς πόρους και τους διαιρούν έτσι ώστε τα εικονικά περιβάλλοντα να μπορούν να τους χρησιμοποιούν.

Οι πόροι κατανέμονται ανάλογα με τις ανάγκες από το φυσικό περιβάλλον στα πολλά εικονικά περιβάλλοντα. Οι χρήστες αλληλοεπιδρούν και εκτελούν υπολογισμούς μέσα στο εικονικό περιβάλλον (που συνήθως ονομάζεται επισκέπτης ή εικονική μηχανή). Η εικονική μηχανή λειτουργεί ως ένα ενιαίο αρχείο δεδομένων. Και όπως κάθε ψηφιακό αρχείο, μπορεί να μετακινηθεί από τον έναν υπολογιστή στον άλλο, να ανοίξει σε έναν από τους δύο και αναμένεται να λειτουργεί το ίδιο.

Όταν εκτελείται το εικονικό περιβάλλον και ένας χρήστης ή ένα πρόγραμμα εκδίδει μια εντολή που απαιτεί πρόσθετους πόρους από το φυσικό περιβάλλον, ο υπερεπόπτης αναμεταδίδει το αίτημα στο φυσικό σύστημα και αποθηκεύει τις αλλαγές στην κρυφή μνήμη κάτι που συμβαίνει με ταχύτητα πλησίον της εγγενούς ταχύτητας (ιδιαίτερα εάν το αίτημα αποστέλλεται μέσω ενός υπερεπόπτη ανοιχτού κώδικα που βασίζεται στο KVM, την εικονική μηχανή που βασίζεται στον πυρήνα).

Συνεπώς τα VM χρησιμοποιούν εικονικοποίηση υλικού, ενώ τα docker χρησιμοποιούν εικονικοποίηση λογισμικού και επομένως έχουν καλύτερη απόδοση (σε μια περίπτωση που τρέχει ένα Dockerized Linux σε μια μηχανή Windows).

Το Docker δεν κάνει εικονικοποίηση. Χρησιμοποιεί χώρους ονομάτων πυρήνα για να επιτύχει ένα εφέ παρόμοιο όχι μόνο για το ριζικό σύστημα αρχείων, αλλά για πληροφορίες διεργασίας (χώρος ονομάτων PID), σημεία προσάρτησης, δικτύωση, IPC (κοινόχρηστη μνήμη), πληροφορίες UTS (όνομα κεντρικού υπολογιστή) και αναγνωριστικά χρήστη.

Οι περιέκτες μοιράζονται τον πυρήνα με τον κεντρικό υπολογιστή. Για ασφάλεια, το Docker χρησιμοποιεί δυνατότητες AppArmor/SELinux, Linux και seccomp για να φιλτράρει τις κλήσεις του συστήματος. Οι ομάδες ελέγχου γνωστές ως cgroups χρησιμοποιούνται για τη λογιστική διεργασίας και για την επιβολή ορίων στους πόρους.

Αυτό σημαίνει ότι δεν μπορεί να εκτελεστεί ένας περιέκτης linux στα Windows ή ένα περιέκτης windows στο linux χωρίς να χρησιμοποιείται κάποιο είδος εικονικοποίησης (Virtualbox, Hyper-v...) Είναι εντάξει να γίνεται στον φορητό υπολογιστή μας κατά την ανάπτυξη, αλλά στην παραγωγή θα πρέπει να επιλεγεί η κατάλληλη αρχιτεκτονική για τους περιέκτες μας.

2.6 Ενορχήστρωση περιεκτών - Container Orchestration

Η ενορχήστρωση περιεκτών (containers) αυτοματοποιεί την ανάπτυξη, διαχείριση, κλιμάκωση και δικτύωση των περιεκτών. Οι επιχειρήσεις που πρέπει να αναπτύξουν και να διαχειριστούν εκατοντάδες ή χιλιάδες περιεκτών και κεντρικούς υπολογιστές Linux® μπορούν να επωφεληθούν από την ενορχήστρωση περιεκτών. Η ενορχήστρωση περιεκτών (containers) μπορεί να χρησιμοποιηθεί σε οποιοδήποτε περιβάλλον όπου χρησιμοποιείται περιέκτης (container). Μπορεί να σας βοηθήσει να αναπτύξετε την ίδια εφαρμογή σε διαφορετικά περιβάλλοντα χωρίς να χρειάζεται να τον επανασχεδιάσετε. Και οι μικροϋπηρεσίες περιεκτών (containers) διευκολύνουν την ενορχήστρωση υπηρεσιών, όπως αποθήκευση, δικτύωση και ασφάλεια.[9]

Οι περιέκτες (containers) παρέχουν στις εφαρμογές σας που βασίζονται στα microservices μια ιδανική μονάδα ανάπτυξης εφαρμογών και αυτόνομο περιβάλλον εκτέλεσης. Καθιστούν δυνατή την εκτέλεση πολλαπλών τμημάτων μιας εφαρμογής ανεξάρτητα σε microservices , στο ίδιο υλικό, με πολύ μεγαλύτερο έλεγχο σε μεμονωμένα κομμάτια και κύκλους ζωής.

Η διαχείριση του κύκλου ζωής των περιεκτών (containers) με ενορχήστρωση υποστηρίζει επίσης ομάδες DevOps που την ενσωματώνουν σε ροές εργασίας CI / CD. Μαζί με τις διεπαφές προγραμματισμού εφαρμογών (API) και τις ομάδες DevOps, οι μικροϋπηρεσίες περιεκτών είναι το θεμέλιο για εφαρμογές εγγενών νεφών.

Ενορχήστρωση περιεκτών (containers)που χρησιμοποιείται για:

1. Παροχή και ανάπτυξη
2. Διαμόρφωση και προγραμματισμός
3. Κατανομή των πόρων
4. Διαθεσιμότητα container
5. Κλιμάκωση ή αφαίρεση περιεκτών με βάση την εξισορρόπηση του φόρτου εργασίας στην υποδομή σας
6. Ισορροπία φορτίων και δρομολόγηση κυκλοφορίας
7. Παρακολούθηση της υγείας των περιεκτών (containers)
8. Διαμόρφωση εφαρμογών με βάση το container στο οποίο θα εκτελεστούν
9. Διατήρηση ασφαλών αλληλεπιδράσεων μεταξύ περιεκτών

2.7 Μικροϋπηρεσίες - Microservices

Οι μικροϋπηρεσίες είναι μια αρχιτεκτονική και οργανωτική προσέγγιση στην ανάπτυξη λογισμικού όπου το λογισμικό αποτελείται από μικρές ανεξάρτητες υπηρεσίες που επικοινωνούν μέσω καλά καθορισμένων API. Αυτές οι υπηρεσίες ανήκουν σε μικρές, αυτόνομες ομάδες.

Οι αρχιτεκτονικές μικροϋπηρεσιών διευκολύνουν την κλίμακα και την ταχύτερη ανάπτυξη των εφαρμογών, επιτρέποντας την καινοτομία και επιταχύνοντας το χρόνο διάθεσης νέων χαρακτηριστικών στην αγορά.

Μια αρχιτεκτονική microservice – μια παραλλαγή του δομικού στυλ SOA (service-oriented architecture) Αρχιτεκτονική Προσανατολισμού Υπηρεσιών – οργανώνει μια εφαρμογή ως μια συλλογή από χαλαρά συζευγμένες υπηρεσίες. Σε μια αρχιτεκτονική μικροϋπηρεσιών, οι υπηρεσίες είναι λεπτομερείς και τα πρωτόκολλα είναι ελαφριά. Ο στόχος είναι οι ομάδες να μπορούν να «ζωντανεύουν» τις υπηρεσίες τους ανεξάρτητα από άλλες. Οι απαιτήσεις επικοινωνίας μειώνονται. Αυτά τα οφέλη έχουν κόστος για τη διατήρηση της αποσύνδεσης. Οι διεπαφές πρέπει να σχεδιάζονται προσεκτικά και να αντιμετωπίζονται ως δημόσιο API. Μια τεχνική που χρησιμοποιείται είναι η ύπαρξη πολλαπλών διεπαφών στην ίδια υπηρεσία ή πολλαπλών εκδόσεων της ίδιας υπηρεσίας, έτσι ώστε να μην ενοχλούνται οι υπάρχοντες χρήστες του κώδικα.

2.7.1 Σύγκριση Μονολιθικής Αρχιτεκτονικής με Microservices.

Με μονολιθικές αρχιτεκτονικές, όλες οι διεργασίες συνδέονται στενά και εκτελούνται ως ενιαία υπηρεσία. Αυτό σημαίνει ότι εάν μια διεργασία της εφαρμογής παρουσιάσει αύξηση της ζήτησης, ολόκληρη η αρχιτεκτονική πρέπει να κλιμακωθεί. Η προσθήκη ή η βελτίωση των δυνατοτήτων μιας μονολιθικής εφαρμογής γίνεται πιο περίπλοκη όσο μεγαλώνει η βάση κώδικα. Αυτή η πολυπλοκότητα περιορίζει τον πειραματισμό και καθιστά δύσκολη την εφαρμογή νέων ιδεών. Οι μονολιθικές αρχιτεκτονικές προσθέτουν κίνδυνο για τη διαθεσιμότητα της εφαρμογής επειδή πολλές εξαρτημένες και στενά συνδεδεμένες διεργασίες αυξάνουν τον αντίκτυπο μιας μεμονωμένης αποτυχημένης διαδικασίας.

Με μια αρχιτεκτονική microservices, μια εφαρμογή δημιουργείται ως ανεξάρτητο στοιχείο που εκτελεί κάθε διαδικασία εφαρμογής ως υπηρεσία. Αυτές οι υπηρεσίες επικοινωνούν μέσω μιας καλά καθορισμένης διεπαφής χρησιμοποιώντας ελαφριά API. Οι υπηρεσίες έχουν

δημιουργηθεί για επιχειρηματικές δυνατότητες και κάθε υπηρεσία εκτελεί μία μόνο λειτουργία. Επειδή εκτελούνται ανεξάρτητα, κάθε υπηρεσία μπορεί να ενημερωθεί, να αναπτυχθεί και να κλιμακωθεί για να καλύψει τη ζήτηση για συγκεκριμένες λειτουργίες μιας εφαρμογής.

2.7.2 Οφέλη των Microservices

- Ευκινησία

Τα Microservices ενθαρρύνουν μια οργάνωση μικρών, ανεξάρτητων ομάδων που αναλαμβάνουν την κυριότητα των υπηρεσιών τους. Οι ομάδες ενεργούν μέσα σε ένα μικρό και καλά κατανοητό πλαίσιο και έχουν τη δυνατότητα να εργάζονται πιο ανεξάρτητα και πιο γρήγορα. Αυτό συντομεύει τους χρόνους του κύκλου ανάπτυξης.

- Ευέλικτη Κλιμάκωση

Οι μικροϋπηρεσίες επιτρέπουν σε κάθε υπηρεσία να κλιμακώνεται ανεξάρτητα για να καλύψει τη ζήτηση για τη δυνατότητα εφαρμογής που υποστηρίζει. Αυτό δίνει τη δυνατότητα στις ομάδες να έχουν τις ανάγκες υποδομής σωστού μεγέθους, να μετρούν με ακρίβεια το κόστος μιας λειτουργίας και να διατηρούν τη διαθεσιμότητα εάν μια υπηρεσία παρουσιάσει αύξηση της ζήτησης.

- Εύκολη ανάπτυξη

Οι μικροϋπηρεσίες επιτρέπουν τη συνεχή ενσωμάτωση και τη συνεχή παράδοση, καθιστώντας εύκολη τη δοκιμή νέων ιδεών και την επαναφορά αν κάτι δεν λειτουργεί. Το χαμηλό κόστος της αποτυχίας επιτρέπει τον πειραματισμό, διευκολύνει την ενημέρωση κώδικα και επιταχύνει την κυκλοφορία νέων δυνατοτήτων στην αγορά.

- Τεχνολογική Ελευθερία

Οι αρχιτεκτονικές μικροϋπηρεσιών δεν ακολουθούν μια προσέγγιση "ένα μέγεθος για όλους". Οι ομάδες έχουν την ελευθερία να επιλέξουν το καλύτερο εργαλείο για την επίλυση των συγκεκριμένων προβλημάτων τους. Κατά συνέπεια, οι ομάδες που δημιουργούν μικροϋπηρεσίες μπορούν να επιλέξουν το καλύτερο εργαλείο για κάθε εργασία.

- Επαναχρησιμοποιήσιμος κώδικας.

Η διαίρεση του λογισμικού σε μικρές, καλά καθορισμένες ενότητες επιτρέπει στις ομάδες να χρησιμοποιούν λειτουργίες για πολλαπλούς σκοπούς. Μια υπηρεσία γραμμένη για μια συγκεκριμένη λειτουργία μπορεί να χρησιμοποιηθεί ως δομικό στοιχείο για ένα άλλο χαρακτηριστικό. Αυτό επιτρέπει σε μια εφαρμογή να κάνει bootstrap από μόνη της, καθώς οι προγραμματιστές μπορούν να δημιουργήσουν νέες δυνατότητες χωρίς να γράφουν κώδικα από την αρχή.

- Ελαστικότητα

Η ανεξαρτησία της υπηρεσίας αυξάνει την αντίσταση μιας εφαρμογής στην αποτυχία. Σε μια μονολιθική αρχιτεκτονική, εάν ένα μεμονωμένο στοιχείο αποτύχει, μπορεί να προκαλέσει αποτυχία ολόκληρης της εφαρμογής. Με τις μικροϋπηρεσίες, οι εφαρμογές χειρίζονται την πλήρη αποτυχία της υπηρεσίας υποβαθμίζοντας τη λειτουργικότητα και δεν διακόπτουν τη λειτουργία ολόκληρης της εφαρμογής.

Τα οφέλη της αποσύνθεσης μιας εφαρμογής σε διαφορετικές μικρότερες υπηρεσίες είναι πολλά:

- Modularity

Αυτό καθιστά την εφαρμογή ευκολότερη στην κατανόηση, την ανάπτυξη, τη δοκιμή και την καθιστά πιο ανθεκτική στη διάβρωση της αρχιτεκτονικής.[6] Αυτό το όφελος συχνά υποστηρίζεται σε σύγκριση με την πολυπλοκότητα των μονολιθικών αρχιτεκτονικών.

- Επεκτασιμότητα

Εφόσον οι μικροϋπηρεσίες υλοποιούνται και αναπτύσσονται ανεξάρτητα η μία από την άλλη, δηλαδή εκτελούνται εντός ανεξάρτητων διεργασιών, μπορούν να παρακολουθούνται και να κλιμακώνονται ανεξάρτητα.

Ενσωμάτωση ετερογενών και παλαιών συστημάτων: οι μικροϋπηρεσίες θεωρούνται βιώσιμο μέσο για τον εκσυγχρονισμό της υπάρχουσας μονολιθικής εφαρμογής λογισμικού. Υπάρχουν αναφορές εμπειρίας πολλών εταιρειών που έχουν αντικαταστήσει επιτυχώς (τμήματα) του υπάρχοντος λογισμικού τους με μικροϋπηρεσίες ή βρίσκονται στη διαδικασία να το κάνουν. Η διαδικασία εκσυγχρονισμού λογισμικού παλαιού τύπου εφαρμογών γίνεται χρησιμοποιώντας μια σταδιακή προσέγγιση.

Κατανεμημένη ανάπτυξη: παραλληλίζει την ανάπτυξη δίνοντας τη δυνατότητα σε μικρές αυτόνομες ομάδες να αναπτύξουν, να αναπτύξουν και να κλιμακώσουν τις αντίστοιχες υπηρεσίες τους ανεξάρτητα. Επιτρέπει επίσης στην αρχιτεκτονική μιας μεμονωμένης

υπηρεσίας να αναδυθεί μέσω συνεχούς ανακατασκευής. Οι αρχιτεκτονικές που βασίζονται σε μικροϋπηρεσίες διευκολύνουν τη συνεχή ενοποίηση, τη συνεχή παράδοση και ανάπτυξη.

2.7.3 Ενσωμάτωση μικροϋπηρεσιών σε περιέκτες (Microservices Dockerization)

Το Docker είναι μια παγκοσμίως κορυφαία πλατφόρμα CaaS (Container-as-a-Service), είναι το πιο κυρίαρχο εργαλείο στο οικοσύστημα περιεκτών (containers) αυτή τη στιγμή. Ο περιέκτης είναι ένας τρόπος συσκευασίας λογισμικού μαζί με δυαδικά αρχεία και ρυθμίσεις που απαιτούνται για να γίνει το λογισμικό που εκτελείται απομονωμένο κατά την κοινή χρήση ενός λειτουργικού συστήματος. Φυσικά, τα δοχεία έχουν πολλά οφέλη — μαζί με ορισμένες επιπλοκές και ανησυχίες.[10]

- Περιβαλλοντική συνέπεια: Οι εφαρμογές που εκτελούνται σε περιέκτη συμπεριφέρονται με συνέπεια σε διαφορετικά περιβάλλοντα. Αυτό εξαλείφει τις περιβαλλοντικές ασυνέπειες και καθιστά τη δοκιμή και τον εντοπισμό σφαλμάτων λιγότερο περίπλοκη και λιγότερο χρονοβόρα.
- Ταχύτερη ανάπτυξη: Ένας περιέκτης είναι ελαφρύς και ξεκινά και σταματά σε λιγότερο από ένα δευτερόλεπτο, καθώς δεν απαιτούν εκκίνηση λειτουργικού συστήματος. Αυτό μας βοηθά τελικά να επιτύχουμε ταχύτερη ανάπτυξη και υψηλή διαθεσιμότητα.
- Απομόνωση: Τα δοχεία που λειτουργούν στο ίδιο μηχάνημα χρησιμοποιώντας τους ίδιους πόρους είναι απομονωμένα μεταξύ τους. Έτσι, εάν μια εφαρμογή διακοπεί, δεν θα επηρεάσει την άλλη εφαρμογή.
- Φορητότητα: Ένα σημαντικό πλεονέκτημα των περιεκτών (containers) είναι η φορητότητά τους. Ένας περιέκτης ολοκληρώνει μια εφαρμογή με όλα όσα χρειάζεται για να εκτελεστεί, όπως αρχεία διαμόρφωσης και εξαρτήσεις. Αυτό σας δίνει τη δυνατότητα να εκτελείτε εύκολα και αξιόπιστα εφαρμογές σε διαφορετικά περιβάλλοντα, όπως η τοπική επιφάνεια εργασίας, οι φυσικοί διακομιστές, οι εικονικοί διακομιστές, οι δοκιμές, η εγκατάσταση, τα περιβάλλοντα παραγωγής και τα δημόσια ή ιδιωτικά σύννεφα. Αυτή η φορητότητα παρέχει στους οργανισμούς μεγάλη ευελιξία, επιταχύνει τη διαδικασία ανάπτυξης και διευκολύνει τη μετάβαση σε άλλο περιβάλλον cloud ή πάροχο, εάν χρειάζεται.

- Επεκτασιμότητα: Ένα σημαντικό πλεονέκτημα των περιεκτών είναι ότι προσφέρουν τη δυνατότητα οριζόντιας κλιμάκωσης, που σημαίνει ότι προσθέτετε περισσότερα πανομοιότυπα δοχεία σε ένα σύμπλεγμα για να κλιμακωθούν. Με την έξυπνη κλιμάκωση, όπου εκτελείται μόνο ο περιέκτης που χρειάζεστε σε πραγματικό χρόνο, μπορείτε να μειώσετε δραστικά το κόστος των πόρων σας και να επιταχύνετε την απόδοση της επένδυσής σας.

Για να κατανοήσουμε καλύτερα το σύστημα περιεκτών (containers), πρέπει να εξετάσουμε τις τρεις ακόλουθες περιοχές.

- Πολλαπλοί περιέκτες και πώς θα τα καταφέρουμε όταν έχουμε πολλούς περιέκτες
- Πολλαπλοί κεντρικοί υπολογιστές και πώς μπορούμε να διανείμουμε την κίνηση από αυτούς σε μεμονωμένους περιέκτες
- Δικτύωση και πώς μπορούν πολλοί περιέκτες να επικοινωνούν μεταξύ τους μεμονωμένα

Μπορούμε να αντιμετωπίσουμε τα παραπάνω προβλήματα με διαφορετικούς τρόπους, αλλά η επιλογή του αποτελεσματικού τρόπου μπορεί να βοηθήσει στην αξιοποίηση των δοχείων.

Η αρχιτεκτονική Microservice χωρίζει μια μονολιθική εφαρμογή σε στοιχεία που λειτουργούν στα δικά τους δοχεία και μπορούν να κλιμακωθούν και να κλιμακωθούν ανεξάρτητα. Οι μικροϋπηρεσίες επιτυγχάνουν την πραγματική επεκτασιμότητα.

Στη συνέχεια, υπάρχουν πολλοί περιέκτες με την ίδια εικόνα Docker για να επιτευχθεί υψηλή διαθεσιμότητα. Εάν ένας από τους περιέκτες πέσει κάτω, τότε άλλα περιέκτες μπορούν να είναι διαθέσιμοι για την εξυπηρέτηση αιτημάτων.

Η εκτέλεση πολλών περιεκτών και πολλαπλών αντιγράφων του ίδιου περιέκτη καθιστούν δύσκολη τη διαχείριση και την επικοινωνία. Το Docker δίνει σε κάθε περιέκτη το δικό του σύνολο συσκευών εικονικού δικτύου και μια μοναδική διεύθυνση IP.

2.8 Χρήση ενδιάμεσου λογισμικού - Middleware

Το Middleware χρησιμοποιείται εκτενώς σε εφαρμογές Express, για εργασίες από την εξυπηρέτηση στατικών αρχείων έως τον χειρισμό σφαλμάτων και τη συμπίεση αποκρίσεων HTTP. Ενώ οι συναρτήσεις διαδρομής τερματίζουν τον κύκλο αίτησης-απόκρισης HTTP

επιστρέφοντας κάποια απόκριση στον υπολογιστή-πελάτη HTTP, οι συναρτήσεις ενδιάμεσου λογισμικού συνήθως εκτελούν κάποια λειτουργία στο αίτημα ή την απόκριση και στη συνέχεια καλούν την επόμενη συνάρτηση στη "στοίβα", η οποία μπορεί να είναι περισσότερο ενδιάμεσο λογισμικό ή διαδρομή χειριστής. Η σειρά με την οποία καλείται το ενδιάμεσο λογισμικό εξαρτάται από τον προγραμματιστή της εφαρμογής.

Σημείωση: Το ενδιάμεσο λογισμικό μπορεί να εκτελέσει οποιαδήποτε λειτουργία, να εκτελέσει οποιονδήποτε κώδικα, να κάνει αλλαγές στο αντικείμενο αίτησης και απόκρισης και μπορεί επίσης να τερματίσει τον κύκλο αίτησης-απόκρισης. Εάν δεν τελειώσει ο κύκλος, τότε πρέπει να καλέσει την εντολή next() για να περάσει τον έλεγχο στην επόμενη συνάρτηση ενδιάμεσου λογισμικού (ή το αίτημα θα μείνει κλειστό).

Οι περισσότερες εφαρμογές θα χρησιμοποιούν ενδιάμεσο λογισμικό τρίτου κατασκευαστή για να απλοποιήσουν κοινές εργασίες ανάπτυξης ιστού, όπως εργασία με cookies, περιόδους σύνδεσης, έλεγχο ταυτότητας χρήστη, πρόσβαση σε δεδομένα αιτημάτων POST και JSON, καταγραφή κ.λπ. Μπορείτε να βρείτε μια λίστα με πακέτα ενδιάμεσου λογισμικού που διατηρεί η ομάδα Express (το οποίο περιλαμβάνει και άλλα δημοφιλή πακέτα τρίτων). Άλλα πακέτα Express είναι διαθέσιμα στον διαχειριστή πακέτων NPM.

3. Εργαλεία και Εφαρμογές

3.1 Διαδικτυακή πλατφόρμα ανάπτυξης λογισμικού GitHub

Το GitHub είναι μια διαδικτυακή πλατφόρμα ανάπτυξης λογισμικού. Χρησιμοποιείται για την αποθήκευση, την παρακολούθηση και τη συνεργασία σε έργα λογισμικού.

Διευκολύνει τους προγραμματιστές να μοιράζονται αρχεία κώδικα και να συνεργάζονται με άλλους προγραμματιστές σε έργα ανοιχτού κώδικα. Το GitHub χρησιμεύει επίσης και ως ένας ιστότοπος κοινωνικής δικτύωσης όπου οι προγραμματιστές μπορούν ανοιχτά να δικτυωθούν, να συνεργαστούν και να παρουσιάσουν τη δουλειά τους.

Από την ίδρυσή του το 2008, το GitHub έχει αποκτήσει εκατομμύρια χρήστες και έχει καθιερωθεί ως μια πλατφόρμα για συνεργατικά έργα λογισμικού. Αυτή η δωρεάν υπηρεσία διαθέτει πολλές χρήσιμες λειτουργίες για κοινή χρήση κώδικα και εργασία με άλλους σε πραγματικό χρόνο.

Πέρα από τις λειτουργίες που σχετίζονται με τον κώδικα, το GitHub ενθαρρύνει τους χρήστες να δημιουργήσουν ένα προσωπικό προφίλ και μια επωνυμία για τον εαυτό τους. Μπορείτε να επισκεφτείτε το προφίλ οποιουδήποτε και να δείτε ποιων έργων είναι κάτοχος και ποια η συνεισφορά τους. Αυτό καθιστά το GitHub έναν τύπο κοινωνικού δικτύου για προγραμματιστές και ενθαρρύνει μια συλλογική προσέγγιση στην ανάπτυξη λογισμικού και ιστοτόπων.

Λειτουργία GitHub

Οι χρήστες του GitHub δημιουργούν λογαριασμούς, ανεβάζουν αρχεία και δημιουργούν έργα κωδικοποίησης. Αλλά η πραγματική δουλειά του GitHub έχει έννοια όταν οι χρήστες αρχίζουν να συνεργάζονται.

Ενώ ο καθένας μπορεί να κωδικοποιεί ανεξάρτητα, ομάδες ανθρώπων κατασκευάζουν τα περισσότερα έργα ανάπτυξης. Μερικές φορές αυτές οι ομάδες βρίσκονται όλες σε ένα μέρος ταυτόχρονα, αλλά πιο συχνά λειτουργούν ασύγχρονα. Υπάρχουν πολλές προκλήσεις για τη δημιουργία συνεργατικών έργων με κατανεμημένες ομάδες. Το GitHub κάνει αυτή τη διαδικασία πολύ πιο απλή με μερικούς διαφορετικούς τρόπους.

GitHub platform home

Καταρχάς, όλος ο κώδικας και η τεκμηρίωση βρίσκονται σε ένα μέρος. Αυτό περιορίζει τα προβλήματα πρόσβασης για οποιονδήποτε θέλει να συνεισφέρει σε ένα έργο. Κάθε αποθετήριο

περιέχει επίσης οδηγίες και άλλες λεπτομέρειες που βοηθούν στην περιγραφή των στόχων και των κανόνων του έργου.

Στη συνέχεια, η κωδικοποίηση είναι πιο δημιουργική και αφηρημένη από ό,τι πιστεύουν οι περισσότεροι μη τεχνικοί. Για παράδειγμα, ας πούμε ότι δύο προγραμματιστές εργάζονται σε διαφορετικά κομμάτια κώδικα. Αυτά τα δύο κομμάτια κώδικα θα πρέπει να συνεργάζονται. Αλλά μερικές φορές ένα κομμάτι κώδικα μπορεί να κάνει τον άλλο κώδικα να αποτύχει. Ή ένα κομμάτι κώδικα μπορεί να έχει απροσδόκητη επίδραση στον τρόπο λειτουργίας του άλλου κώδικα.

Το GitHub επιλύει αυτά τα προβλήματα δείχνοντας πώς και τα δύο αρχεία θα αλλάξουν τον κύριο κλάδο. Εντοπίζει αυτά τα σφάλματα πριν πιέσει τις αλλαγές, κάνοντας τη διαδικασία κωδικοποίησης πιο αποτελεσματική.

Το GitHub διευκολύνει επίσης την παρακολούθηση των αλλαγών και την επιστροφή στις προηγούμενες εκδόσεις ενός έργου. Για να το εξηγήσουμε αυτό, θα πρέπει να κατανοήσουμε την τεχνολογία στην οποία βασίζεται το GitHub, το Git και να μιλήσουμε για τον έλεγχο έκδοσης.

Το Git είναι λογισμικό ελέγχου έκδοσης ανοιχτού κώδικα, που χρησιμοποιείται για τη διαχείριση και την παρακολούθηση αναθεωρήσεων αρχείων. Μπορεί να χρησιμοποιείται το Git με οποιονδήποτε τύπο αρχείου, αλλά η συχνότερη χρήση του είναι για αρχεία κώδικα παρακολούθησης.

Το Git είναι το πιο ευρέως χρησιμοποιούμενο σύστημα ελέγχου εκδόσεων στην ανάπτυξη λογισμικού και το GitHub αξιοποιεί αυτήν την τεχνολογία για την υπηρεσία του, εξ ου και το όνομά του.

3.2 Επίσημο Μητρώο εικόνων Docker DockerHub

Το Docker Hub είναι το επίσημο μητρώο του Docker που βασίζεται στο Cloud για εικόνες Docker. Η χρήση του Docker Hub - του επίσημου μητρώου περιεκτών του Docker - έχει εκτοξευθεί στα ύψη. Από τον Νοέμβριο του 2019 έως το τέλος Ιουλίου 2020, οι συνολικές λήψεις (λήψη μιας εικόνας Docker) από το Docker Hub αυξήθηκαν από 130 δισεκατομμύρια σε 242 δισεκατομμύρια. Αυτός είναι ένας δείκτης του πόσο δημοφιλής έχει γίνει η μεταφορά περιεκτών (containers) γενικά αλλά και το Docker ειδικότερα.

Δεν είναι μυστικό ότι το Docker βρίσκεται παντού, αλλά η κατανόηση των ιδιαιτεροτήτων του συστήματος μπορεί να είναι δύσκολη.

Όπως θα περίμενε κανείς, με δεδομένο ότι το Docker Hub είναι το επίσημο μητρώο του Docker, είναι και το προεπιλεγμένο μητρώο όταν εγκαθιστάτε το Docker. Φιλοξενεί πάνω από 100.000 εικόνες, συμπεριλαμβανομένων επίσημων εικόνων για MongoDB, nginx, Apache, Ubuntu και MySQL που έχουν ληφθεί πάνω από ένα δισεκατομμύριο φορές η καθεμία τους.

Εκτός από τα δημόσια αποθετήρια από τα οποία μπορεί να αντλήσει ο καθένας, το Docker Hub προσφέρει ιδιωτικά αποθετήρια όπου άτομα ή ομάδες μπορούν να φιλοξενήσουν εικόνες στις οποίες επιθυμούν να περιορίσουν την πρόσβαση. Το Docker Hub προσφέρει επίσης λειτουργίες όπως ενσωματώσεις GitHub και Bitbucket που βοηθούν στην αυτοματοποίηση των διαδικασιών δημιουργίας ανάπτυξης και υποστήριξη για webhook, που μπορούν να λειτουργήσουν ως ενεργοποιητές για τα πάντα, από αυτοματοποιημένες δοκιμές έως ειδοποιήσεις.

Εδώ είναι μερικά από τα βασικά πλεονεκτήματα του Docker Hub:

- Μια μεγάλη βιβλιοθήκη αξιόπιστων εικόνων - οι εικόνες Docker Certified, οι εικόνες Verified Publisher (οι οποίες είναι πιστοποιημένες με Docker και επαληθεύονται από τον εκδότη) και οι Επίσημες εικόνες που δημοσιεύονται από το Docker προσθέτουν ένα επίπεδο εμπιστοσύνης στους χρήστες. Με εκατομμύρια — ή σε ορισμένες περιπτώσεις δισεκατομμύρια — λήψεις για πολλές εικόνες που χρησιμοποιούνται συνήθως, μπορούμε να βασιστούμε σε μια αξιόπιστη βασική εικόνα όταν χρησιμοποιείται το Docker hub. Αν και αυτό είναι υπέροχο από την πλευρά του χρήστη, ωφελεί επίσης τους εκδότες, καθώς η φιλοξενία μιας εικόνας στο Docker Hub μπορεί να δώσει στο έργο σας μεγαλύτερη έκθεση.
- Ένα δωρεάν επίπεδο- Επί του παρόντος, το δωρεάν πρόγραμμα του Docker προσφέρει απεριόριστους δημόσιους χώρους αποθήκευσης και 1 ιδιωτικό χώρο αποθήκευσης με έως και 3 συνεργάτες. Αυτό είναι χρήσιμο για βασικές δοκιμές και εξοικείωση με την πλατφόρμα.
- Ενσωματωμένα χαρακτηριστικά ασφαλείας- Όλοι οι λογαριασμοί μπορούν να επωφεληθούν από τοπικές σαρώσεις ευπάθειας εικόνας. Οι λογαριασμοί "ομάδας" αποκτούν επίσης πρόσβαση στα αρχεία καταγραφής ελέγχου και στον έλεγχο ταυτότητας πολλαπλών παραγόντων για την περαιτέρω ασφάλεια των αποθετηρίων.
- Ενσωματώσεις και δυνατότητες που ενεργοποιούν το CI/CD– Το Docker Hub υποστηρίζει επίσης ενσωματώσεις GitHub & Bitbucket, αυτοματοποιημένες δοκιμές, ενεργοποιήσεις build και webhook για να βοηθήσουν στην αυτοματοποίηση των αγωγών ανάπτυξης και να ενεργοποιήσουν το CI/CD (συνεχής ενοποίηση/συνεχής παράδοση). wiki

Φυσικά, δεν υπάρχει ποτέ μια λύση που ταιριάζει σε όλους, οπότε το Docker Hub δεν θα είναι η σωστή απάντηση για κάθε περίπτωση χρήσης.

Το Docker Hub είναι το μεγαλύτερο αποθετήριο εικόνων περιεκτών στον κόσμο με μια σειρά από πηγές περιεχομένου, συμπεριλαμβανομένων προγραμματιστών κοινότητας περιεκτών, έργων ανοιχτού κώδικα και ανεξάρτητων προμηθευτών λογισμικού (ISV) που δημιουργούν και διανέμουν τον κώδικά τους σε περιέκτες. Οι χρήστες έχουν πρόσβαση σε δωρεάν δημόσια αποθετήρια για αποθήκευση και κοινή χρήση εικόνων ή μπορούν να επιλέξουν πρόγραμμα συνδρομής για ιδιωτικά αποθετήρια.

3.3 Linux

3.3.1 Εισαγωγή

Το Linux είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα που μοιάζει με Unix που βασίζεται στον πυρήνα του Linux. Έναν πυρήνα λειτουργικού συστήματος που κυκλοφόρησε για πρώτη φορά στις 17 Σεπτεμβρίου 1991, από τον Linus Torvalds.[11]

Το πακέτο περιλαμβάνει τον πυρήνα Linux και το λογισμικό υποστήριξης του συστήματος τις βιβλιοθήκες, πολλές από τις οποίες παρέχονται από το έργο GNU. Πολλές διανομές Linux χρησιμοποιούν τη λέξη "Linux" στο όνομά τους, αλλά το Ίδρυμα Ελεύθερου Λογισμικού χρησιμοποιεί το όνομα "GNU/Linux" για να τονίσει τη σημασία του λογισμικού GNU, προκαλώντας κάποιες διαμάχες.

Οι δημοφιλείς διανομές Linux περιλαμβάνουν το Debian, το Fedora Linux και το Ubuntu, το οποίο από μόνο του έχει πολλές διαφορετικές διανομές και τροποποιήσεις, συμπεριλαμβανομένων των Lubuntu και Xubuntu. Οι εμπορικές διανομές περιλαμβάνουν το Red Hat Enterprise Linux και το SUSE Linux Enterprise. Οι διανομές Desktop Linux περιλαμβάνουν ένα σύστημα παραθύρων όπως το X11 ή το Wayland και ένα περιβάλλον επιφάνειας εργασίας όπως το GNOME ή το KDE Plasma. Οι διανομές που προορίζονται για διακομιστές ενδέχεται να παραλείπουν εντελώς τα γραφικά ή να περιλαμβάνουν μια στοίβα λύσεων όπως το LAMP. Επειδή το Linux μπορεί να αναδιανεμηθεί ελεύθερα, οποιοσδήποτε μπορεί να δημιουργήσει μια διανομή για οποιονδήποτε σκοπό.

Το Linux αναπτύχθηκε αρχικά για προσωπικούς υπολογιστές με βάση την αρχιτεκτονική Intel x86, αλλά έκτοτε έχει μεταφερθεί σε περισσότερες πλατφόρμες από οποιοδήποτε άλλο λειτουργικό σύστημα. Λόγω της κυριαρχίας του Android που βασίζεται σε Linux σε

smartphone, το Linux, συμπεριλαμβανομένου του Android, έχει τη μεγαλύτερη εγκατεστημένη βάση από όλα τα λειτουργικά συστήματα γενικής χρήσης, από τον Μάιο του 2022. Παρόλο που το Linux χρησιμοποιείται, από τον Μάιο του 2022, μόνο από περίπου 2,3 ποσοστό των επιτραπέζιων υπολογιστών, το Chromebook, το οποίο εκτελεί το Chrome OS που βασίζεται σε πυρήνα Linux, κυριαρχεί στην αγορά εκπαίδευσης K-12 των ΗΠΑ και αντιπροσωπεύει σχεδόν το 20 τοις εκατό των πωλήσεων φορητών υπολογιστών κάτω των 300 $ στις ΗΠΑ. Το Linux είναι το κορυφαίο λειτουργικό σύστημα σε διακομιστές (πάνω από το 96,4% των λειτουργικών συστημάτων του 1 εκατομμυρίου κορυφαίων διακομιστών Ιστού είναι Linux), ηγείται άλλων μεγάλων συστημάτων όπως οι υπολογιστές mainframe και είναι το μόνο λειτουργικό σύστημα που χρησιμοποιείται σε υπερυπολογιστές TOP500 (από τον Νοέμβριο του 2017, έχοντας σταδιακά αποκλείσει όλους τους ανταγωνιστές).

Το Linux εκτελείται επίσης σε ενσωματωμένα συστήματα, δηλαδή συσκευές των οποίων το λειτουργικό σύστημα είναι συνήθως ενσωματωμένο στο υλικολογισμικό και είναι ιδιαίτερα προσαρμοσμένο στο σύστημα. Αυτό περιλαμβάνει δρομολογητές, χειριστήρια αυτοματισμού, έξυπνες οικιακές συσκευές, κάμερες IP, κονσόλες βιντεοπαιχνιδιών, τηλεοράσεις (οι έξυπνες τηλεοράσεις Samsung και LG χρησιμοποιούν Tizen και WebOS, αντίστοιχα), αυτοκίνητα (Tesla, Audi, Mercedes-Benz, Hyundai και Toyota όλα βασίζονται σε Linux), Spacecraft (τα αεροηλεκτρονικά του Falcon 9 και του Dragon 2 χρησιμοποιούν μια προσαρμοσμένη έκδοση του Linux) και Rovers (The Mars 2020 Mission μετέφερε 3 υπολογιστές Linux στον Άρη, συμπεριλαμβανομένου του ελικοπτέρου Ingenuity).

Το Linux είναι ένα από τα πιο χαρακτηριστικά παραδείγματα συνεργασίας ελεύθερου λογισμικού και λογισμικού ανοιχτού κώδικα. Ο πηγαίος κώδικας μπορεί να χρησιμοποιηθεί, να τροποποιηθεί και να διανεμηθεί εμπορικά ή μη από οποιονδήποτε σύμφωνα με τους όρους των αντίστοιχων αδειών του, όπως η Γενική Δημόσια Άδεια GNU (GPL). Ο πυρήνας Linux, για παράδειγμα, έχει άδεια χρήσης σύμφωνα με το GPLv2, με ειδική εξαίρεση για κλήσεις συστήματος, καθώς χωρίς την εξαίρεση κλήσης συστήματος, οποιοδήποτε πρόγραμμα που καλεί τον πυρήνα θα θεωρείται παράγωγο και επομένως η GPL θα πρέπει να ισχύει για αυτό το πρόγραμμα.

3.3.2 Ubuntu Linux

Το Ubuntu είναι μια διανομή Linux που βασίζεται στο Debian και αποτελείται κυρίως από δωρεάν λογισμικό ανοιχτού κώδικα. Το Ubuntu κυκλοφορεί επίσημα σε τρεις εκδόσεις:

Desktop, Server και Core για συσκευές και ρομπότ Internet of things. Όλες οι εκδόσεις μπορούν να εκτελεστούν μόνο στον υπολογιστή ή σε μια εικονική μηχανή. Το Ubuntu είναι ένα δημοφιλές λειτουργικό σύστημα για cloud computing, με υποστήριξη για OpenStack. Η προεπιλεγμένη επιφάνεια εργασίας του Ubuntu είναι το GNOME από την έκδοση 17.10.

Το Ubuntu κυκλοφορεί κάθε έξι μήνες, με εκδόσεις μακροπρόθεσμης υποστήριξης (LTS) κάθε δύο χρόνια. Από τις 21 Απριλίου 2022, η πιο πρόσφατη έκδοση μακροπρόθεσμης υποστήριξης είναι η 22.04 ("Jammy Jellyfish").

Το Ubuntu αναπτύσσεται από τη βρετανική εταιρεία Canonical και μια κοινότητα άλλων προγραμματιστών, σύμφωνα με ένα αξιοκρατικό μοντέλο διακυβέρνησης. Η Canonical παρέχει ενημερώσεις ασφαλείας και υποστήριξη για κάθε έκδοση του Ubuntu, ξεκινώντας από την ημερομηνία κυκλοφορίας και έως ότου η κυκλοφορία φτάσει στην καθορισμένη ημερομηνία λήξης ζωής (EOL). Η Canonical δημιουργεί έσοδα μέσω της πώλησης premium υπηρεσιών που σχετίζονται με το Ubuntu και δωρεών από όσους κατεβάζουν το λογισμικό Ubuntu.

Το Ubuntu πήρε το όνομά του από τη φιλοσοφία Nguni του ubuntu, την οποία η Canonical υποδεικνύει ότι σημαίνει «ανθρωπιά προς τους άλλους» με την έννοια «Είμαι αυτό που είμαι εξαιτίας αυτού που είμαστε όλοι»

3.3.3 Linux Alpine

Το Alpine Linux είναι μια έκδοση Linux που έχει σχεδιαστεί να είναι μικρή, απλή και ασφαλής. Το Alpine Linux χρησιμοποιεί musl, BusyBox και OpenRC αντί των συχνά χρησιμοποιούμενων glibc, GNU Core Utilities και systemd αντίστοιχα. Το Alpine Linux είναι προσανατολισμένο στην ασφάλεια, όντας ελαφριά διανομή Linux που βασίζεται στη βιβλιοθήκη musl libc και στην πλατφόρμα βοηθητικών προγραμμάτων BusyBox αντί για το GNU. Λειτουργεί σε υλικό «γυμνού μετάλλου», σε VM ή ακόμα και σε Raspberry Pi.

Λόγω του μικρού του μεγέθους και της γρήγορης εκκίνησης, χρησιμοποιείται συνήθως σε περιέκτες που παρέχουν γρήγορους χρόνους εκκίνησης, σε εικονικές μηχανές καθώς και σε πραγματικό υλικό σε ενσωματωμένες συσκευές, όπως δρομολογητές, διακομιστές αλλά και NAS.

Για ασφάλεια, το Alpine μεταγλωττίζει όλα τα δυαδικά αρχεία χώρου χρήστη ως εκτελέσιμα αρχεία ανεξάρτητα από τη θέση με προστασία κατά της στοίβας.

Ο πυρήνας του Alpine Linux με μια ανεπίσημη θύρα ενημερωμένης έκδοσης κώδικα grsecurity, με το LibreSSL Secure Sockets Layer και το σύστημα αρχικοποίησης OpenRC (init), συμβάλλουν σε μια ασφαλή διανομή. Ο χρήστης του Alpine Linux θα βρει τα περισσότερα πράγματα απενεργοποιημένα ή μη εγκατεστημένα από προεπιλογή, μια άλλη στρατηγική ασφάλειας για το λειτουργικό σύστημα. Άλλα χαρακτηριστικά ασφαλείας αποτρέπουν την υπερχείλιση του buffer στοίβας και την καταστροφή της μνήμης για κάθε πακέτο.

Η διανομή Alpine Linux καταλαμβάνει περίπου 300 MB για το τυπικό μέγεθος αποθηκευτικού χώρου. Χρησιμοποιεί τον διαχειριστή πακέτων apk, ο οποίος προσθέτει, διαγράφει και επιδιορθώνει πακέτα. Ενώ το Alpine Linux είναι γενικής χρήσης, ο βασικός σχεδιασμός είναι αραιός χωρίς βοηθητικά προγράμματα GNU, για να διατηρείται ένα μικρό αποτύπωμα. Ο χρήστης αναμένεται να επιλέξει από χιλιάδες πακέτα και να εγκαταστήσει αυτά που ανταποκρίνονται στις ανάγκες μιας συγκεκριμένης εργασίας. Ένας χρήστης μπορεί επίσης να δημιουργήσει ένα προσαρμοσμένο και μοναδικό πακέτο.

Το Alpine Linux είναι μια δημοφιλής επιλογή λειτουργικού συστήματος για την εκτέλεση περιεκτών, αν και δεν είναι ειδικά σχεδιασμένο για αυτήν την εργασία. Το περιβάλλον του περιέκτη έχει μικρό αποτύπωμα, ωστόσο, το Alpine Linux απαιτεί σημαντική προσπάθεια κατά την εγκατάσταση για να λειτουργήσει σωστά το Docker. Η σύνδεση δικτύου Alpine Linux για λειτουργίες περιέκτη περιλαμβάνει τη μη αυτόματη δημιουργία του αρχείου /etc/network/interfaces, για παράδειγμα. Επειδή χρησιμοποιεί εναλλακτικά στοιχεία Linux, το Alpine Linux μπορεί να είναι λιγότερο οικείο για έναν διαχειριστή περιεκτών από το Red Hat Enterprise Linux Atomic Host ή τον Windows Server 2016.

Για να χρησιμοποιήσετε το Alpine Linux, ο διαχειριστής πρέπει να είναι εξοικειωμένος με τον επεξεργαστή κειμένου vi, που είναι κοινός στις διανομές Linux. Ως εκ τούτου, μπορεί να είναι δύσκολο για τους χρήστες Windows και Mac να το παραλάβουν.

3.3.4 Τα μειονεκτήματα του Ubuntu έναντι του Alpine

Ενώ μια βασική εικόνα του Ubuntu είναι πλεονεκτική από πολλές απόψεις, το Alpine Linux μπορεί να είναι καλύτερη επιλογή σε ορισμένες περιπτώσεις. Τα μειονεκτήματα του Ubuntu σε σύγκριση με το Alpine περιλαμβάνουν:

- Μεγαλύτερο μέγεθος εικόνας: Τα images βάσης Alpine συνολικά καταλαμβάνουν περίπου 5,5 Megabyte – πολύ μικρότερες από τα περίπου 75 Megabyte που καταλαμβάνει το

Ubuntu. Ως αποτέλεσμα, οι περιέκτες που δημιουργούνται χρησιμοποιώντας το Alpine ως βασική εικόνα θα είναι επίσης μικρότεροι. Με τη σειρά τους, οι περιέκτες που βασίζονται σε Alpine θα χρειαστούν λιγότερο χρόνο για λήψη, σάρωση και (στις περισσότερες περιπτώσεις) εκτέλεση.

- Ευρύτερη επιφάνεια επίθεσης: Επειδή τα images του Ubuntu περιλαμβάνουν πολύ περισσότερα βοηθητικά προγράμματα και βιβλιοθήκες από προεπιλογή από το Alpine, ενέχουν υψηλότερο κίνδυνο ασφάλειας. Πρόκειται για συνολικά 81 βοηθητικά προγράμματα στο /bin – όχι ασήμαντο αριθμό, αλλά ακόμα πολύ λιγότερα από τα 294 του Ubuntu.

Επιπλέον, αν κοιτάξουμε πιο προσεκτικά, θα ανακαλύψουμε ότι τα περισσότερα από τα βοηθητικά προγράμματα της Alpine είναι στην πραγματικότητα απλώς συμβολικοί σύνδεσμοι προς το BusyBox, ένα εκτελέσιμο που παρέχει έναν αριθμό εργαλείων Unix μέσα σε μία μόνο εφαρμογή. Έτσι, τεχνικά μιλώντας, το Alpine περιέχει πραγματικά μόνο ένα βοηθητικό πρόγραμμα - το BusyBox. Υπό αυτή την έννοια, το Alpine είναι ακόμα πιο αδύνατο να «σπάσει» και πιο εύκολο να ασφαλιστεί και τα μόνα τρωτά σημεία για τα οποία συντρέχει λόγος ανησυχίας είναι αυτά που επηρεάζουν το BusyBox. Αυτό δεν συμβαίνει στο Ubuntu, όπου καθένα από τα εκατοντάδες εργαλεία που είναι εγκατεστημένα από προεπιλογή θα μπορούσε να υπόκειται σε οποιονδήποτε αριθμό ευπαθειών.

Συμπέρασμα: Σε περιπτώσεις χρήσης όπου η ασφάλεια είναι κορυφαία προτεραιότητα ή/και όπου η εφαρμογή έχει πολύ λίγες εξαρτήσεις, είναι πιο λογικό να χρησιμοποιείται το Alpine ως βασική εικόνα από το Ubuntu ή μια παρόμοια διανομή Linux "πλήρους βάρους".

3.4 Υποσύστημα των Windows για Linux - WSL

Πρόκειται για μια δυνατότητα του λειτουργικού συστήματος Windows η οποία δίνει τη δυνατότητα εκτέλεσης ενός συστήματος αρχείων Linux, με εργαλεία γραμμής εντολών Linux και εφαρμογές GUI, άμεσα στα Windows, παράλληλα με την επιφάνεια εργασίας και τις εφαρμογές των Windows.

Απευθύνεται κυρίως για προγραμματιστές, καθώς είναι ένα εργαλείο και ειδικότερα προγραμματιστές ιστού, οι οποίοι εργάζονται σε ανοιχτό κώδικα ή αναπτύσσουν περιβάλλοντα διακομιστών Linux. Το WSL είναι για αυτούς που τους αρέσει να χρησιμοποιούν τα Bash,

εργαλεία Linux (sed, awk, κ.λπ.) και πλαίσια Linux-first (Ruby, Python, κ.λπ.), αλλά και εργαλεία των Windows.

Με το WSL δίνεται η δυνατότητα να εκτελείται το Linux σε Bash με την επιλογή σε Ubuntu, Debian, OpenSUSE, Kali, Alpine, κ.λπ. Χρησιμοποιώντας Bash, μπορείτε να εκτελέσετε εργαλεία και εφαρμογές Linux σε γραμμή εντολών. Επίσης γίνεται να αποκτηθεί πρόσβαση στο σύστημα αρχείων του τοπικού μηχανήματος μέσω του Linux Bash

3.5 NodeJS

Το Node (ή πιο επίσημα Node.js) είναι ένα περιβάλλον χρόνου εκτέλεσης ανοιχτού κώδικα, πολλαπλών πλατφορμών που επιτρέπει στους προγραμματιστές να δημιουργούν όλα τα είδη εργαλείων και εφαρμογών από την πλευρά του διακομιστή σε JavaScript.[12] Ο χρόνος εκτέλεσης προορίζεται για χρήση εκτός περιβάλλοντος προγράμματος περιήγησης (δηλαδή εκτέλεση απευθείας σε υπολογιστή ή λειτουργικό σύστημα διακομιστή). Ως εκ τούτου, το περιβάλλον παραλείπει τα API JavaScript για συγκεκριμένα προγράμματα περιήγησης και προσθέτει υποστήριξη για πιο παραδοσιακά API λειτουργικού συστήματος, συμπεριλαμβανομένων των βιβλιοθηκών HTTP και συστημάτων αρχείων.

Από την άποψη της ανάπτυξης διακομιστή ιστού, το Node έχει μια σειρά από πλεονεκτήματα:

- Καταπληκτικές επιδόσεις. Το Node σχεδιάστηκε για τη βελτιστοποίηση της απόδοσης και της επεκτασιμότητας σε εφαρμογές Ιστού και είναι μια καλή λύση για πολλά κοινά προβλήματα ανάπτυξης εφαρμογών ιστού (π.χ. εφαρμογές ιστού σε πραγματικό χρόνο).
- Ο κώδικας είναι γραμμένος σε "απλό παλιό JavaScript", που σημαίνει ότι αφιερώνεται λιγότερος χρόνος για την αντιμετώπιση της "μετατόπισης περιβάλλοντος" μεταξύ των γλωσσών όταν γράφετε κώδικα τόσο από την πλευρά του πελάτη όσο και από τον διακομιστή.
- Η JavaScript είναι μια σχετικά νέα γλώσσα προγραμματισμού και επωφελείται από βελτιώσεις στον σχεδιασμό της γλώσσας σε σύγκριση με άλλες παραδοσιακές γλώσσες διακομιστή ιστού (π.χ. Python, PHP, κ.λπ.) Πολλές άλλες νέες και δημοφιλείς γλώσσες μεταγλωττίζονται/μετατρέπονται σε JavaScript, ώστε να μπορείτε επίσης να χρησιμοποιήσετε το TypeScript, CoffeeScript, ClojureScript, Scala, LiveScript, κ.λπ.
- Ο διαχειριστής πακέτων κόμβου (NPM) παρέχει πρόσβαση σε εκατοντάδες χιλιάδες επαναχρησιμοποιήσιμα πακέτα. Έχει επίσης την καλύτερη ανάλυση εξάρτησης στην

κατηγορία του και μπορεί επίσης να χρησιμοποιηθεί για την αυτοματοποίηση του μεγαλύτερου μέρους της αλυσίδας εργαλείων κατασκευής.

- Το Node.js είναι φορητό. Είναι διαθέσιμο σε Microsoft Windows, macOS, Linux, Solaris, FreeBSD, OpenBSD, WebOS και NonStop OS. Επιπλέον, υποστηρίζεται καλά από πολλούς παρόχους φιλοξενίας ιστοσελίδων, οι οποίοι συχνά παρέχουν συγκεκριμένη υποδομή και τεκμηρίωση για τη φιλοξενία τοποθεσιών Node.
- Διαθέτει ένα πολύ ενεργό οικοσύστημα και κοινότητα προγραμματιστών τρίτων, με πολλούς ανθρώπους που είναι πρόθυμοι να βοηθήσουν.

Μπορείτε να χρησιμοποιήσετε το Node.js για να δημιουργήσετε έναν απλό διακομιστή web χρησιμοποιώντας το πακέτο Node HTTP.

3.5.1 Express

Το Express είναι το πιο δημοφιλές πλαίσιο web για το Node.js και είναι η υποκείμενη βιβλιοθήκη για μια σειρά από άλλα δημοφιλή πλαίσια web για το Node.js.[13] Παρέχει μηχανισμούς για να κάνετε τα παρακάτω:

Να γράψετε προγράμματα χειρισμού αιτημάτων με διαφορετικά ρήματα HTTP σε διαφορετικές διαδρομές URL (διαδρομές).

Να ενσωματώσετε τις μηχανές απόδοσης "προβολής" για να δημιουργήσετε αποκρίσεις εισάγοντας δεδομένα σε πρότυπα.

Να ορίσετε κοινές ρυθμίσεις εφαρμογών web, όπως τη θύρα που θα χρησιμοποιηθεί για τη σύνδεση και τη θέση των προτύπων που χρησιμοποιούνται για την απόδοση της απόκρισης.

Να προσθέσετε πρόσθετο "ενδιάμεσο λογισμικό" επεξεργασίας αιτημάτων σε οποιοδήποτε σημείο εντός του αγωγού διαχείρισης αιτημάτων.

Ενώ η ίδια η Express είναι αρκετά μινιμαλιστική, οι προγραμματιστές έχουν δημιουργήσει συμβατά πακέτα ενδιάμεσου λογισμικού για να αντιμετωπίσουν σχεδόν οποιοδήποτε πρόβλημα ανάπτυξης ιστού. Υπάρχουν βιβλιοθήκες για εργασία με cookies, περιόδους σύνδεσης, συνδέσεις χρηστών, παραμέτρους URL, δεδομένα POST, κεφαλίδες ασφαλείας και πολλά άλλα. Μπορείτε να βρείτε μια λίστα με πακέτα ενδιάμεσου λογισμικού που διατηρεί η ομάδα Express στο Express Middleware (μαζί με μια λίστα με μερικά δημοφιλή πακέτα τρίτων).

Αυτή η ευελιξία είναι ένα δίκοπο μαχαίρι. Υπάρχουν πακέτα ενδιάμεσου λογισμικού για την αντιμετώπιση σχεδόν οποιουδήποτε προβλήματος ή απαίτησης, αλλά η επεξεργασία των σωστών πακέτων για χρήση μπορεί μερικές φορές να είναι μια πρόκληση. Επίσης, δεν υπάρχει "σωστός τρόπος" δομής μιας εφαρμογής και πολλά παραδείγματα που μπορεί να βρείτε στο Διαδίκτυο δεν είναι βέλτιστα ή δείχνουν μόνο ένα μικρό μέρος του τι πρέπει να κάνετε για να αναπτύξετε μια εφαρμογή ιστού.

Σε έναν παραδοσιακό ιστότοπο που βασίζεται σε δεδομένα, μια εφαρμογή Ιστού περιμένει αιτήματα HTTP από το πρόγραμμα περιήγησης Ιστού (ή άλλο πελάτη). Όταν λαμβάνεται ένα αίτημα, η εφαρμογή επεξεργάζεται ποια ενέργεια χρειάζεται με βάση το μοτίβο διεύθυνσης URL και πιθανώς τις σχετικές πληροφορίες που περιέχονται στα δεδομένα POST ή στα δεδομένα GET. Ανάλογα με το τι απαιτείται, μπορεί στη συνέχεια να διαβάσει ή να γράψει πληροφορίες από μια βάση δεδομένων ή να εκτελέσει άλλες εργασίες που απαιτούνται για την ικανοποίηση του αιτήματος. Στη συνέχεια, η εφαρμογή θα επιστρέψει μια απάντηση στο πρόγραμμα περιήγησης ιστού, δημιουργώντας συχνά δυναμικά μια σελίδα HTML για την εμφάνιση του προγράμματος περιήγησης, εισάγοντας τα ανακτημένα δεδομένα σε σύμβολα κράτησης θέσης σε ένα πρότυπο HTML.

Η Express παρέχει μεθόδους για τον καθορισμό της συνάρτησης που καλείται για ένα συγκεκριμένο ρήμα HTTP (GET, POST, SET, κ.λπ.) και μοτίβο διεύθυνσης URL ("Διαδρομή") και μεθόδους για τον καθορισμό του μηχανισμού προτύπου ("προβολή") που χρησιμοποιείται, που βρίσκονται τα αρχεία προτύπων και ποιο πρότυπο να χρησιμοποιήσετε για την απόδοση μιας απάντησης. Μπορείτε να χρησιμοποιήσετε το ενδιάμεσο λογισμικό Express για να προσθέσετε υποστήριξη για cookies, περιόδους σύνδεσης και χρήστες, να λάβετε παραμέτρους POST/GET κ.λπ. Μπορείτε να χρησιμοποιήσετε οποιονδήποτε μηχανισμό βάσης δεδομένων που υποστηρίζεται από το Node (το Express δεν καθορίζει καμία συμπεριφορά σχετική με τη βάση δεδομένων).

3.5.2 Axios

Είναι μια βιβλιοθήκη HTTP που ενεργεί την αποστολή ασύγχρονων αιτημάτων HTTP στα επιθυμητά σημεία REST ώστε να εκτελούνται σωστά λειτουργίες CRUD. Το τελικό σημείο/API REST μπορεί να είναι ένα εξωτερικό API (Google API, GitHub API κα) ή ακόμα και ο δικός μας κόμβος υποστήριξης. Το Axios είναι βασισμένο σε «υποσχέσεις» για το node.js και το πρόγραμμα περιήγησης. Μπορεί να τρέξει στο πρόγραμμα περιήγησης και στο node.js

με την ίδια βάση κώδικα. Στην πλευρά του διακομιστή χρησιμοποιεί την εγγενή λειτουργική μονάδα http node.js, ενώ στου πελάτη (πρόγραμμα περιήγησης) χρησιμοποιεί XMLHttpRequests.

3.5.3 Socket.io

Το WebSocket είναι ένα πρωτόκολλο επικοινωνίας που παρέχει αμφίδρομη επικοινωνία μεταξύ πελάτη και διακομιστή μέσω σύνδεσης TCP. Μένει ανοιχτό καθ' όλη τη διάρκεια έτσι ώστε να επιτρέπεται η μεταφορά δεδομένων σε πραγματικό χρόνο.[14] Όταν ο πελάτης στέλνει κάποιο αίτημα στον διακομιστή, η σύνδεση δεν κλείνει μετά τη λήψη της απάντησης. αλλά παραμένει μέχρι να τερματίσει το αίτημα είτε ο πελάτη είτε ο διακομιστής.

Το Socket.IO είναι μια βιβλιοθήκη που παρέχει σύνδεση μέσω TCP και επιτρέπει την επικοινωνία σε πραγματικό χρόνο και full-duplex μεταξύ του Client και των διακομιστών Web. Χρησιμοποιεί το πρωτόκολλο WebSocket για τη διεπαφή. Χωρίζεται σε δύο μέρη. Το WebSocket όπως και το Socket.io είναι «βιβλιοθήκες» που βασίζονται σε συμβάντα.

Το WebSocket είναι η τεχνολογία, ενώ το Socket.io είναι μια βιβλιοθήκη για WebSockets.

Το Socket.IO αποτελείται από:

- έναν διακομιστής Node.js
- μια βιβλιοθήκη πελάτη JavaScript για το πρόγραμμα περιήγησης (ή έναν πελάτη Node.js)

Υπάρχουν υλοποιήσεις του Socket.io σε γλώσσες που είναι διαθέσιμες όπως Java, C++, Swift, Dart, Python, .NET.

Τα κύρια χαρακτηριστικά του Socket.IO είναι :

- Αξιοπιστία
- Υποστήριξη αυτόματης επανασύνδεσης
- Ανίχνευση αποσύνδεσης
- Δυαδική υποστήριξη
- Απλό και βολικό API
- Cross-browser
- Υποστήριξη πολυπλεξίας

Βασικά χαρακτηριστικά του Socket.IO είναι ότι βοηθά στη μετάδοση σε πολλά sockets ταυτόχρονα και χειρίζεται τη σύνδεση με διαφάνεια. Λειτουργεί σε όλες τις πλατφόρμες, είτε συσκευή είτε διακομιστή και διασφαλίζει ταχύτητα, ισότητα και αξιοπιστία. Αναβαθμίζει

αυτόματα την απαίτηση αν χρειαστεί σε WebSocket και είναι μια εφαρμογή πρωτοκόλλου μεταφοράς σε πραγματικό χρόνο ξεπερνώντας τα άλλα πρωτόκολλα. Χρειάζονται και οι δύο βιβλιοθήκες στην πλευρά του πελάτη και μια βιβλιοθήκη στην πλευρά του διακομιστή.

Το IO λειτουργεί με τον μηχανισμό των συμβάντων που ενεργοποιούνται βάση των εργασιών που εκτελούνται. Τα συμβάντα υπάρχουν και στην πλευρά του πελάτη όπως Connect, Connect-error, Connect-timeout and Reconnect και στην πλευρά του διακομιστή, όπως Connect, Message, Disconnect, Ping and Reconnect.

3.6 Docker

Το Docker ορίζεται σαν ένα σύνολο προϊόντων πλατφόρμας ως υπηρεσία (PaaS) που χρησιμοποιούν εικονικοποίηση σε επίπεδο λειτουργικού συστήματος για την παράδοση λογισμικού σε πακέτα που ονομάζονται περιέκτες (containers).[15] Η υπηρεσία έχει τόσο δωρεάν όσο και premium επίπεδα χρήσης. Το λογισμικό που φιλοξενεί τους περιέκτες (containers) ονομάζεται Docker Engine. Ξεκίνησε το 2013 και αναπτύχθηκε από την Docker, Inc. Οι περιέκτες (containers) είναι απομονωμένοι ο ένας από τον άλλο και συνδυάζουν το δικό τους λογισμικό, βιβλιοθήκες και αρχεία διαμόρφωσης και επικοινωνούν μεταξύ τους μέσω καθορισμένων καναλιών. Επειδή όλοι οι περιέκτες (containers) μοιράζονται τις υπηρεσίες ενός ενιαίου πυρήνα λειτουργικού συστήματος, χρησιμοποιούν λιγότερους πόρους από τις εικονικές μηχανές.

Το Docker μπορεί να συσκευάσει μια εφαρμογή και τις εξαρτήσεις της σε ένα εικονικό περιέκτη που μπορεί να εκτελεστεί σε οποιονδήποτε υπολογιστή Linux, Windows ή macOS. Αυτό δίνει τη δυνατότητα στην εφαρμογή να εκτελείται σε διάφορες τοποθεσίες, όπως εσωτερική εγκατάσταση, δημόσια (βλ. αποκεντρωμένα συστήματα, κατανεμημένα συστήματα και συστήματα νέφους) ή και σε κάποιο ιδιωτικό cloud. Επειδή οι περιέκτες του Docker είναι ελαφροί, ένας διακομιστής ή μια εικονική μηχανή μπορεί να τρέξει πολλούς περιέκτες ταυτόχρονα. Σε μια ανάλυση του 2018 διαπιστώθηκε ότι μια τυπική περίπτωση χρήσης Docker περιλαμβάνει τη λειτουργία οκτώ περιεκτών ανά κεντρικό υπολογιστή και ότι το ένα τέταρτο των οργανισμών που αναλύθηκαν εκτελούν 18 ή περισσότερα ανά κεντρικό υπολογιστή.

Το Docker εφαρμόζει ένα API υψηλού επιπέδου για να παρέχει ελαφρούς περιέκτες που εκτελούν διαδικασίες μεμονωμένα. Οι περιέκτες (containers) Docker είναι τυπικές διεργασίες, επομένως είναι δυνατή η χρήση χαρακτηριστικών του πυρήνα για την παρακολούθηση της

εκτέλεσής τους συμπεριλαμβανομένης για παράδειγμα της χρήσης εργαλείων όπως το strace για την παρατήρηση και τη μεσολάβηση στις κλήσεις συστήματος

Εργαλεία του Docker είναι το Docker Swarm, το Docker Compose και το Docker Volume.

3.7 Docker Compose

Το Docker Compose είναι ένα εργαλείο για την εκτέλεση εφαρμογών πολλαπλών περιεκτών στο Docker που ορίζονται χρησιμοποιώντας τη μορφή αρχείου Compose και αναπτύχθηκε για να βοηθήσει στον καθορισμό και την κοινή χρήση εφαρμογών πολλαπλών περιεκτών. Με το Compose, μπορούμε να δημιουργήσουμε ένα αρχείο YAML για να ορίσουμε τις υπηρεσίες και με μία μόνο εντολή, να περιστρέψουμε τα πάντα ή να τα καταστρέψουμε όλα. Χρησιμοποιεί αρχεία YAML για τη διαμόρφωση των υπηρεσιών της εφαρμογής και εκτελεί τη διαδικασία δημιουργίας και εκκίνησης όλων των περιεκτών με μία μόνο εντολή. Το βοηθητικό πρόγραμμα docker-compose CLI επιτρέπει στους χρήστες να εκτελούν εντολές σε πολλους περιέκτες ταυτόχρονα, για παράδειγμα, δημιουργία εικόνων, κλιμάκωση περιεκτών, εκτέλεση περιεκτών που έχουν σταματήσει και πολλά άλλα. Εντολές που σχετίζονται με χειρισμό εικόνας ή επιλογές αλληλεπίδρασης με το χρήστη, δεν είναι σχετικές στο Docker Compose επειδή απευθύνονται σε ένα περιέκτη. Το αρχείο docker-compose.yml χρησιμοποιείται για τον καθορισμό των υπηρεσιών μιας εφαρμογής και περιλαμβάνει διάφορες επιλογές διαμόρφωσης. Για παράδειγμα, η επιλογή δημιουργίας ορίζει επιλογές διαμόρφωσης όπως η διαδρομή Dockerfile. Η επιλογή εντολής επιτρέπει σε κάποιον να παρακάμψει τις προεπιλεγμένες εντολές Docker και πολλά άλλα. Ένα αρχείο Compose χρησιμοποιείται για να ορίσει τον τρόπο διαμόρφωσης ενός ή περισσότερων περιεκτών που απαρτίζουν την εφαρμογή σας. Μόλις έχετε ένα αρχείο Compose, μπορείτε να δημιουργήσετε και να ξεκινήσετε την εφαρμογή σας με μία μόνο εντολή: docker compose up.

Η πρώτη δημόσια έκδοση beta του Docker Compose (έκδοση 0.0.1) κυκλοφόρησε στις 21 Δεκεμβρίου 2013. Η πρώτη έκδοση έτοιμη για παραγωγή (1.0) έγινε διαθέσιμη στις 16 Οκτωβρίου 2014.

3.8 Ενορχηστρωτές (Docker Swarm – Kubernetes - Nomad)

3.8.1 Docker Swarm

Το Docker Swarm παρέχει λειτουργία εγγενούς ομαδοποίησης για περιέκτες Docker, η οποία μετατρέπει μια ομάδα μηχανών Docker σε μια ενιαία εικονική μηχανή Docker.[16] Στο Docker 1.12 και νεότερη έκδοση, η λειτουργία Swarm είναι ενσωματωμένη στο Docker Engine. Το βοηθητικό πρόγραμμα docker swarm CLI επιτρέπει στους χρήστες να εκτελούν περιέκτες Swarm, να δημιουργούν διακριτικά εντοπισμού, να καταγράφουν κόμβους στο σύμπλεγμα και πολλά άλλα. Το βοηθητικό πρόγραμμα docker node CLI επιτρέπει στους χρήστες να εκτελούν διάφορες εντολές για τη διαχείριση κόμβων σε ένα σμήνος, για παράδειγμα, καταχώριση των κόμβων σε ένα σμήνος, ενημέρωση κόμβων και κατάργηση κόμβων από το σμήνος. Ο Docker διαχειρίζεται τα σμήνη χρησιμοποιώντας τον αλγόριθμο συναίνεσης Raft. Σύμφωνα με τον Raft, για να εκτελεστεί μια ενημέρωση, η πλειοψηφία των κόμβων Swarm πρέπει να συμφωνήσουν σχετικά με την ενημέρωση.

3.8.2 Kubernetes

Το Kubernetes είναι μια φορητή, επεκτάσιμη πλατφόρμα ανοιχτού κώδικα για τη διαχείριση φόρτου εργασίας και υπηρεσιών με περιέκτες (containers), που διευκολύνει τόσο την παραμετροποίηση μέσω δηλώσεων όσο και τον αυτοματισμό.[17] Έχει ένα μεγάλο, ταχέως αναπτυσσόμενο οικοσύστημα. Οι υπηρεσίες, η υποστήριξη και τα εργαλεία Kubernetes είναι ευρέως διαθέσιμα.

3.8.3 Nomad

Το Nomad είναι ένας ευέλικτος ενορχηστρωτής φόρτου εργασίας που επιτρέπει σε έναν οργανισμό να αναπτύσσει και να διαχειρίζεται εύκολα οποιαδήποτε εφαρμογή με περιέκτες (containers) ή εφαρμογές παλαιού τύπου, χρησιμοποιώντας μια ενιαία, ενοποιημένη ροή εργασίας.[18] Το Nomad μπορεί να εκτελέσει έναν ποικίλο φόρτο εργασίας εφαρμογών Docker, μη περιεκτών, μικροϋπηρεσιών και παρτίδων και επιτρέπει στους προγραμματιστές

να χρησιμοποιούν δηλωτική υποδομή ως κώδικα (infrastructure-as-a-code) για την ανάπτυξη εφαρμογών.

3.8.4 Συγκριτικά Kubernetes, Nomad, Docker Swarm.

Παρόμοια Χαρακτηριστικά:

- Αυτόματη κλιμάκωση

Αξιοποιώντας ένα σύστημα κατάταξης που βασίζεται σε προκαθορισμένες λειτουργίες, τόσο το Nomad όσο και το Kubernetes αναζητούν αυτόματα κατάλληλους κεντρικούς υπολογιστές για την εκτέλεση μιας εφαρμογής

- Μηχανισμοί αυτόματης ανάκτησης και αυτοθεραπείας

Και οι δύο πλατφόρμες ενορχήστρωσης περιλαμβάνουν διάφορα εργαλεία και μηχανισμούς για επανεκκίνηση εφαρμογών και ανάκτηση δεδομένων σε περίπτωση αποτυχίας.

- Στρατηγικές διάθεσης και επαναφοράς

Για τη διαχείριση μεγάλων αναπτύξεων συμπλέγματος, τόσο το Nomad όσο και το Kubernetes υποστηρίζουν κυλιόμενες ενημερώσεις out of the box.

- Ενορχήστρωση αποθήκευσης

Τόσο το Nomad όσο και το Kubernetes υποστηρίζουν διάφορες προσθήκες τρίτων που συμμορφώνονται με το πρότυπο Container Storage Interface (CSI), επιτρέποντας στις εφαρμογές να συνδέονται με εξωτερικούς τόμους αποθήκευσης. Επιτρέποντας την κατανάλωση αποθηκευτικού χώρου από δημόσια/ιδιωτικά σύννεφα ή υποδομές γυμνού σε μέταλλο εγκαταστάσεων (bare-metal on-premises), και οι δύο πλατφόρμες επιτρέπουν την αποτελεσματική ενορχήστρωση αποθήκευσης για τη διατήρηση του φορτίου εργασίας σε ιδανική κατάσταση.

Διαφορές:

- Ευέλικτη υποστήριξη φόρτου εργασίας:

Το Kubernetes έχει κατασκευαστεί για να ενορχηστρώνει περιέκτες που φιλοξενούν φόρτους εργασίας, ενώ το Nomad θεωρείται περισσότερο ως ένας προγραμματιστής φόρτου εργασίας που μπορεί να χρησιμοποιηθεί για τη διαχείριση της ανάπτυξης εφαρμογών παλαιού τύπου και περιεκτών (containers), καθώς και για τον προγραμματισμό εργασιών δέσμης.

- Ενορχήστρωση περιεκτών από άκρη σε άκρη

Το kubernetes προσφέρει υπηρεσίες ενορχήστρωσης περιεκτών από άκρο σε άκρο, όπως εξισορρόπηση φορτίου, διαχείριση διαμόρφωσης, δρομολόγηση, πύλες δυνατοτήτων και ανακάλυψη υπηρεσιών.

Το Nomad είναι κυρίως μια πλατφόρμα προγραμματισμού εργασιών που μπορεί να ενορχηστρώσει διαφορετικούς τύπους φόρτου εργασίας ως εκτεταμένο χαρακτηριστικό.

- Επεκτασιμότητα

Και οι δύο πλατφόρμες έχουν κατασκευαστεί για αυτόματη κλιμάκωση σε μεγάλες αναπτύξεις, ωστόσο το Nomad θεωρείται ευνοϊκό για την κλίμακα μεγαλύτερης κλίμακας από το Kubernetes. Ενώ το Kubernetes έχει κατασκευαστεί για να υποστηρίζει συμπλέγματα με έως και 5.000 κόμβους που ενορχηστρώνουν έως και 300.000 περιεκτών, το Nomad μπορεί να κλιμακώσει συμπλέγματα που υπερβαίνουν τους 10.000 κόμβους στην παραγωγή και ξεπέρασε το σημείο αναφοράς για την πρόκληση των δύο εκατομμυρίων περιεκτών.

- Συνεπής ανάπτυξη.

Το Kubernetes προσφέρει διαφορετικά περιβάλλοντα ανάπτυξης και διαχείρισης για να διευκολύνει τους οργανισμούς να εκτελούν ανάπτυξη και δοκιμές πριν ωθήσουν τις εφαρμογές στην ανάπτυξη. Αυτά μπορεί να δημιουργήσουν ασυνέπειες στη διαμόρφωση και τις δυνατότητες κατά τη μετάδοση ζωντανά. Το Nomad χρησιμοποιεί μια ενιαία, φορητή βιβλιοθήκη που μπορεί να αναπτυχθεί σε οποιοδήποτε περιβάλλον για να παρέχει την ίδια εμπειρία χρήστη σε διαφορετικές πλατφόρμες.

Πλεονεκτήματα του Kubernetes:

- Τεράστια κοινότητα
- Υγιή εργαλεία και οικοσύστημα ενοποίησης τρίτων
- Συνεχής καινοτομία μέσω της συνεχούς κυκλοφορίας νέων εκδόσεων
- Ευρεία διαθεσιμότητα ταλέντων και ευκαιριών κατάρτισης

- Το καλύτερο εάν ο οργανισμός σας χρειάζεται μια ολοκληρωμένη λύση ενορχήστρωσης περιεκτών.

Μειονεκτήματα του Kubernetes:

- Δύσκολη η διαμόρφωση και η ανάπτυξη
- Περιορίζεται σε φόρτους εργασίας με περιέκτες (containers)
- Είναι μια πολύ πιο σύνθετη πλατφόρμα από το Nomad

Πλεονεκτήματα του Nomad:

- Εύκολο στην ανάπτυξη, επιτρέποντας συνεπείς παραμετροποιήσεις ανεξάρτητα από το λειτουργικό σύστημα
- Binary και πακέτα διαθέσιμα για Linux, macOS και Linux
- Γενικά, είναι μια απλούστερη πλατφόρμα από την Kubernetes
- Ιδανικό για διαχείριση φόρτου εργασίας εικονικοποιημένων, περιεκτών και αυτόνομων εφαρμογών κάτω από μια ενιαία πλατφόρμα.

Μειονεκτήματα του Nomad:

- Μικρότερη κοινότητα
- Λιγότερα διαθέσιμα εργαλεία και ενσωματώσεις
- Λόγω των πιο περιορισμένων ευκαιριών εκπαίδευσης, μπορεί να είναι πιο δύσκολο να βρείτε ταλέντα εξειδικευμένα στο Nomad σε σύγκριση με την Kubernetes

Το Kubernetes είναι ένα αυτόνομο εργαλείο ενορχήστρωσης με πολλές ενσωματωμένες υπηρεσίες που παρέχουν όλες τις δυνατότητες που χρειάζεστε για να εκτελέσετε μια εφαρμογή που βασίζεται σε περιέκτη. Διαθέτει την πιο εντυπωσιακή κοινότητα συνεργατών και υποστήριξη cloud, παρέχοντας μια πλούσια εργαλειοθήκη και έναν μεγάλο αριθμό λύσεων εκτός συσκευασίας. Ωστόσο, είναι δύσκολο να ρυθμιστεί χειροκίνητα και έχει σχεδιαστεί μόνο για εφαρμογές σε περιέκτες.

Αντίθετα, το Nomad είναι εύκολο στην εγκατάσταση και λειτουργία, επειδή εστιάζει μόνο στη διαχείριση συμπλέγματος. Υποστηρίζει επίσης διάφορους τύπους φόρτου εργασίας, αλλά προσφέρει περιορισμένη λειτουργικότητα, η οποία απαιτεί την εγκατάσταση εργαλείων τρίτων για την επίλυση εργασιών που εφαρμόζει η Kubernetes από προεπιλογή.

Εάν οι εφαρμογές σας απαιτούν πρόσθετες λειτουργίες και αισθάνεστε άνετα να αφιερώσετε χρόνο για να μάθετε το εργαλείο, τότε το Kubernetes μπορεί να είναι η καλύτερη

επιλογή. Ωστόσο, εάν προτιμάτε μια πιο απλή ροή εργασίας χωρίς πρόσθετη λειτουργικότητα, τότε το Nomad μπορεί να είναι κατάλληλο για τις εφαρμογές σας. Τελικά, ποιο εργαλείο θα επιλέξετε εξαρτάται από την περίπτωση χρήσης και την ικανότητά σας να εφαρμόσετε το εργαλείο στον κύκλο παραγωγής σας. Φροντίστε να λάβετε υπόψη τους παραπάνω παράγοντες όταν λάβετε την απόφασή σας για το 2022.

Ωστόσο, οι μοναδικότητα των διαχειριζόμενων υπηρεσιών καθιστούν δύσκολη την υποστήριξη του multi-cloud. Απαιτεί ουσιαστική αναδιαμόρφωση για τη μεταφορά φόρτου εργασίας από το EKS στο GKE, λέει ο Li. Με ορισμένα επίπεδα πάνω από το Kubernetes, όταν κάτι πάει στραβά, θα μπορούσε να χρειαστεί διπλή προσπάθεια για να καταλάβουμε προβλήματα, προσθέτει. Οι διαχειριζόμενες υπηρεσίες Kubernetes ή άλλα εργαλεία ενορχήστρωσης περιεκτών που κρύβουν λειτουργική πολυπλοκότητα δεν πρέπει να προκαλούν πρόσθετη αντιμετώπιση προβλημάτων.

«Οι οργανισμοί τείνουν να επιλέγουν έναν ενορχηστρωτή με βάση το προϊόν, τον προϋπολογισμό και το χρονοδιάγραμμά τους», είπε ο Li. «Δεν είναι ποτέ ένα εργαλείο που ταιριάζει σε όλους».

Κατά τη σύγκριση του Docker Swarm εναντίον Kubernetes, καθίσταται προφανές ότι οι ρίζες και των δύο πλατφορμών έχουν διαδραματίσει βασικούς ρόλους στη διαμόρφωση των χαρακτηριστικών και των κοινοτήτων τους σήμερα.

Το Docker, συνειδητοποιώντας τη δύναμη της τεχνολογίας container, αποφάσισε να δημιουργήσει μια πλατφόρμα που θα διευκόλυνε τους χρήστες του Docker να αρχίσουν να ενορχηστρώνουν το φόρτο εργασίας των περιεκτών (containers)σε πολλούς κόμβους. Ωστόσο, η επιθυμία τους να διατηρήσουν αυτόν τον στενό σύνδεσμο μπορεί να θεωρηθεί ότι περιορίζουν την επεκτασιμότητα της πλατφόρμας.

Το Kubernetes, από την άλλη πλευρά, πήρε βασικές έννοιες που ελήφθησαν από το Google Borg και, από μια προοπτική υψηλού επιπέδου, αποφάσισε να ταιριάζει με το υπάρχον μοντέλο ενορχήστρωσης φόρτου εργασίας της πρώην πλατφόρμας. Αυτό είχε ως αποτέλεσμα την έμφαση του Kubernetes στην αξιοπιστία, μερικές φορές στο κόστος της απλότητας και της απόδοσης.

3.9 Redis (Remote Dictionary Server)

Το Redis (Remote Dictionary Server) είναι ένας χώρος αποθήκευσης δομών δεδομένων στη μνήμη, αναφέρεται συχνά ως διακομιστής δομών δεδομένων, που χρησιμοποιείται ως

κατανεμημένη βάση δεδομένων κλειδιού-τιμής στη μνήμη, προσωρινής μνήμης και κατεύθυνσης μηνυμάτων, με προαιρετική αντοχή. Αυτό σημαίνει ότι το Redis παρέχει πρόσβαση σε μεταβλητές δομές δεδομένων μέσω ενός συνόλου εντολών, οι οποίες αποστέλλονται χρησιμοποιώντας ένα μοντέλο διακομιστή-πελάτη με υποδοχές TCP και ένα απλό πρωτόκολλο. Έτσι, διαφορετικές διαδικασίες μπορούν να υποβάλουν ερωτήματα και να τροποποιήσουν τις ίδιες δομές δεδομένων με κοινόχρηστο τρόπο.

Το Redis υποστηρίζει διαφορετικά είδη αφηρημένων δομών δεδομένων, όπως συμβολοσειρές, λίστες, χάρτες, σύνολα, ταξινομημένα σύνολα, HyperLogLogs, bitmaps, ροές και χωρικούς δείκτες. Το έργο αναπτύχθηκε και συντηρήθηκε από τον Salvatore Sanfilippo, ξεκινώντας το 2009. Από το 2015 έως το 2020, ηγήθηκε μιας βασικής ομάδας έργου που χρηματοδοτήθηκε από την Redis Labs. Ο Salvatore Sanfilippo άφησε το Redis ως συντηρητής το 2020. Είναι λογισμικό ανοιχτού κώδικα που κυκλοφορεί με άδεια BSD 3 ρητρών. Το 2021, λίγο καιρό μετά την αποχώρηση του αρχικού συγγραφέα και κύριου συντηρητή, η Redis Labs αφαίρεσε το Labs από το όνομά του και τώρα είναι γνωστό απλώς ως "Redis".

Το Redis έκανε δημοφιλή την ιδέα ενός συστήματος που μπορεί να θεωρηθεί store και cache μνήμη ταυτόχρονα. Σχεδιάστηκε έτσι ώστε τα δεδομένα να τροποποιούνται και να διαβάζονται πάντα από την κύρια μνήμη του υπολογιστή, αλλά και να αποθηκεύονται στο δίσκο σε μορφή που δεν είναι κατάλληλη για τυχαία πρόσβαση στα δεδομένα. Τα μορφοποιημένα δεδομένα ανακατασκευάζονται στη μνήμη μόνο μετά την επανεκκίνηση του συστήματος.

Το Redis παρέχει επίσης ένα μοντέλο δεδομένων που είναι πολύ ασυνήθιστο σε σύγκριση με ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS). Οι εντολές χρήστη δεν περιγράφουν ένα ερώτημα που πρέπει να εκτελεστεί από τη μηχανή βάσης δεδομένων, αλλά συγκεκριμένες λειτουργίες που εκτελούνται σε δεδομένους αφηρημένους τύπους δεδομένων. Επομένως, τα δεδομένα πρέπει να αποθηκεύονται με τρόπο που να είναι κατάλληλος αργότερα για γρήγορη ανάκτηση. Η ανάκτηση γίνεται χωρίς βοήθεια από το σύστημα βάσης δεδομένων με τη μορφή δευτερευόντων ευρετηρίων, συναθροίσεων ή άλλων κοινών χαρακτηριστικών του παραδοσιακού RDBMS. Η υλοποίηση Redis χρησιμοποιεί σε μεγάλο βαθμό την κλήση συστήματος fork, για να αντιγράψει τη διαδικασία που διατηρεί τα δεδομένα, έτσι ώστε η βασική διαδικασία να συνεχίσει να εξυπηρετεί τους πελάτες ενώ η θυγατρική διαδικασία δημιουργεί ένα αντίγραφο στη μνήμη των δεδομένων στο δίσκο.

Η υλοποίηση δομών δεδομένων δίνει έμφαση στην αποδοτικότητα της μνήμης, επομένως οι δομές δεδομένων μέσα στο Redis πιθανότατα θα χρησιμοποιούν λιγότερη μνήμη σε

σύγκριση με την ίδια δομή δεδομένων που έχει μοντελοποιηθεί χρησιμοποιώντας μια γλώσσα προγραμματισμού υψηλού επιπέδου.

Ένα καλό παράδειγμα είναι να σκεφτούμε το Redis ως μια πιο σύνθετη έκδοση του memcached, όπου οι λειτουργίες δεν είναι μόνο SET και GET, αλλά λειτουργίες που λειτουργούν με σύνθετους τύπους δεδομένων όπως Λίστες, Σύνολα, δομές διατεταγμένων δεδομένων κ.λπ.

Το Redis εισήγαγε το clustering τον Απρίλιο του 2015 με την κυκλοφορία της έκδοσης 3.0. Η προδιαγραφή συμπλέγματος υλοποιεί ένα υποσύνολο εντολών Redis: όλες οι εντολές ενός κλειδιού είναι διαθέσιμες, οι λειτουργίες πολλαπλών κλειδιών (εντολές που σχετίζονται με ενώσεις και διασταυρώσεις) περιορίζονται σε κλειδιά που ανήκουν στον ίδιο κόμβο και εντολές που σχετίζονται με λειτουργίες επιλογής βάσης δεδομένων δεν είναι διαθέσιμες. Ένα σύμπλεγμα Redis μπορεί να κλιμακωθεί σε έως και 1.000 κόμβους, να επιτύχει "αποδεκτή" ασφάλεια εγγραφής και να συνεχίσει τις λειτουργίες όταν ορισμένοι κόμβοι αποτυγχάνουν.

Πολλές γλώσσες προγραμματισμού έχουν συνδέσμους γλώσσας για Redis όπως: ActionScript, C, C++, C#, Chicken, Clojure, Common Lisp, Crystal, D, Dart, Elixir, Erlang, Go, Haskell, Haxe, Io, Java, Nim, JavaScript (Node.js), Julia, Lua, Objective-C, OCaml, Perl, PHP, Pure Data, Python, R, Racket, Ruby, Rust, Scala, Smalltalk, Swift και Tcl.

3.10 MongoDB

Η MongoDB είναι μια βάση δεδομένων εγγράφων ανοιχτού κώδικα που βασίζεται σε μια αρχιτεκτονική οριζόντιας κλίμακας που χρησιμοποιεί ένα ευέλικτο σχήμα για την αποθήκευση δεδομένων[19]. Η MongoDB, που ιδρύθηκε το 2007, έχει παγκόσμιους ακόλουθους στην κοινότητα προγραμματιστών.

Αντί να αποθηκεύονται δεδομένα σε πίνακες σειρών ή στηλών, όπως βάσεις δεδομένων SQL, κάθε εγγραφή σε μια βάση δεδομένων MongoDB είναι ένα έγγραφο που περιγράφεται στο BSON, μια δυαδική αναπαράσταση των δεδομένων. Στη συνέχεια, οι εφαρμογές μπορούν να ανακτήσουν αυτές τις πληροφορίες σε μορφή JSON.

Η MongoDB αναπτύχθηκε από την MongoDB Inc. και διαθέτει άδεια χρήσης βάσει της δημόσιας άδειας χρήσης από την πλευρά του διακομιστή (SSPL), η οποία θεωρείται μη δωρεάν από πολλούς.

Η MongoDB παρέχει υψηλή διαθεσιμότητα για σύνολα αντιγράφων. Ένα σύνολο αντιγράφων αποτελείται από δύο ή περισσότερα αντίγραφα των δεδομένων. Κάθε μέλος σετ

αντιγράφων μπορεί να ενεργεί ως πρωτεύον ή δευτερεύον αντίγραφο ανά πάσα στιγμή. Όλες οι εγγραφές και οι αναγνώσεις γίνονται στο πρωτεύον αντίγραφο από προεπιλογή. Τα δευτερεύοντα αντίγραφα διατηρούν ένα αντίγραφο των δεδομένων του πρωτεύοντος χρησιμοποιώντας ενσωματωμένη αναπαραγωγή. Όταν ένα πρωτεύον αντίγραφο αποτυγχάνει, το σύνολο αντιγράφων διεξάγει αυτόματα μια εκλογική διαδικασία για να καθορίσει ποιο δευτερεύον θα γίνει κύριο.

Εάν η αναπαραγόμενη MongoDB έχει μόνο ένα δευτερεύον μέλος, ένας ξεχωριστός «δαίμονας» που έχει το ρόλο του διαιτητή πρέπει να προστεθεί στο σύνολο. Έχει μια ενιαία ευθύνη, που είναι να επιλύσει την εκλογή του νέου πρωτεύοντος αντιγράφου. Κατά συνέπεια, μια ιδανική κατανεμημένη εγκατάσταση MongoDB απαιτεί τουλάχιστον τρεις ξεχωριστούς διακομιστές, ακόμη και στην περίπτωση ενός μόνο κύριου και ενός δευτερεύοντος.

Η MongoDB μπορεί να χρησιμοποιηθεί ως σύστημα αρχείων, που ονομάζεται GridFS, με δυνατότητες εξισορρόπησης φορτίου και αναπαραγωγής δεδομένων σε πολλαπλές μηχανές για την αποθήκευση αρχείων.

Αυτή η λειτουργία, που ονομάζεται σύστημα αρχείων πλέγματος, περιλαμβάνεται στα προγράμματα οδήγησης MongoDB. Η MongoDB περιλαμβάνει λειτουργίες για χειρισμό αρχείων και περιεχόμενο στους προγραμματιστές. Το GridFS μπορεί να προσπελαστεί χρησιμοποιώντας το βοηθητικό πρόγραμμα mongofiles ή πρόσθετα για το Nginx και το lighttpd.

Λόγω της προεπιλεγμένης διαμόρφωσης ασφαλείας της MongoDB, που επιτρέπει σε οποιονδήποτε να έχει πλήρη πρόσβαση στη βάση δεδομένων, έχουν κλαπεί δεδομένα από δεκάδες χιλιάδες εγκαταστάσεις MongoDB. Επιπλέον, πολλοί διακομιστές MongoDB έχουν κρατηθεί για λύτρα.

Τον Σεπτέμβριο του 2017, ενημερώθηκε τον Ιανουάριο του 2018, σε μια επίσημη απάντηση, ο Davi Ottenheimer, επικεφαλής Ασφάλεια Προϊόντων στη MongoDB, δήλωσε ότι η MongoDB έλαβε μέτρα για την άμυνα έναντι αυτών των κινδύνων.

Από την έκδοση MongoDB 2.6 και μετά, τα δυαδικά αρχεία από τα επίσημα πακέτα MongoDB RPM και DEB συνδέονται με τον localhost από προεπιλογή. Από το MongoDB 3.6, αυτή η προεπιλεγμένη συμπεριφορά επεκτάθηκε σε όλα τα πακέτα MongoDB σε όλες τις πλατφόρμες. Ως αποτέλεσμα εχει όλες οι δικτυωμένες συνδέσεις στη βάση δεδομένων θα απορριφθούν, εκτός εάν ρυθμιστούν ρητά από έναν διαχειριστή.

3.11 PostMan

Το Postman είναι μια πλατφόρμα-εφαρμογή API για δημιουργία και χρήση API καθώς απλοποιεί κάθε βήμα του κύκλου ζωής του API και βελτιστοποιεί τη συνεργασία, ώστε να μπορείτε να δημιουργήσετε καλύτερα API και πιο γρήγορα.[20]

Θεωρείται το μεγαλύτερο δίκτυο API και μπορείτε να διαμοιραστείτε ό,τι δημιουργείτε με προγραμματιστές σε όλο τον πλανήτη.

Πρόκειται λοιπόν για μια εφαρμογή που χρησιμοποιείται για την δοκιμή API.

Είναι ένας client HTTP που τεστάρει αιτήματα HTTP, χρησιμοποιώντας ένα γραφικό περιβάλλον, μέσω του οποίου λαμβάνουμε διαφορετικούς τύπους απαντήσεων που πρέπει στη συνέχεια να επικυρωθούν.

Προσφέρει αρκετές μεθόδους αλληλεπίδρασης και μερικες από τις πιο χρησιμοποιούμενες είναι οι GET POST PUT PATCH DELETE.

Στην λειτουργία λοιπόν του Postman το τεστάρισμα των API, συνήθως λαμβάνουμε διαφορετικούς κωδικούς απόκρισης. Μερικά από τα πιο σύνηθες περιλαμβάνουν:

100 Series Χρονικές αποκρίσεις, για παράδειγμα, «102 Processing».

200 Series > Απαντήσεις όπου ο πελάτης αποδέχεται το αίτημα και ο διακομιστής το επεξεργάζεται με επιτυχία, για παράδειγμα, «200 Ok».

300 Series > Απαντήσεις που σχετίζονται με την ανακατεύθυνση URL, για παράδειγμα, "301 Moved Permanently."

400 Series > Αποκρίσεις σφαλμάτων πελάτη, για παράδειγμα, «400 Bad Request».

500 Series > Αποκρίσεις σφαλμάτων διακομιστή, για παράδειγμα, "500 Internal Server Error".

Στην εφαρμογή δίνεται και η δυνατότητα ομαδοποίησης διαφορετικών αιτημάτων, γνωστή ως «συλλογή» για την καλύτερη και σωστότερη οργάνωση των δοκιμών.

Πρόκειται για φακέλους όπου αποθηκεύονται τα αιτήματα και δομούνται κατά τρόπο τέτοιο ώστε η ομάδα να είναι ικανοποιημένη. Υπάρχει η δυνατότητα εξαγωγής και εισαγωγής τους από την εφαρμογή.

Επίσης η εφαρμογή μας επιτρέπει την δημιουργία διαφορετικών περιβαλλόντων μέσω της χρήσης μεταβλητών, όπως για παράδειγμα, μια μεταβλητή URL που στοχεύει σε διαφορετικά περιβάλλοντα δοκιμής, οπού μας επιτρέπει να εκτελούμε δοκιμές σε διαφορετικά περιβάλλοντα χρησιμοποιώντας κάποια υπάρχοντα αιτήματα.

Τα οφέλη από τη χρήση του Postman για τις δοκιμές API, είναι πάρα πολλές. Κάποιες όμως πιο σημαντικές; είναι και οι εξής:

1. Προσβασιμότητα από cloud. Όντας συνδεδεμένοι στο λογαριασμό του ο εκάστοτε χρήστης δύναται να εκτελέσει δοκιμή API μέσω Postman οποτεδήποτε, οπουδήποτε.
2. Συνεργασία. Στο Postman η εισαγωγή και εξαγωγή διευκολύνει την κοινή χρήση αρχείων με άλλα μέλη της ομάδας, και έτσι η συνεργασία τους είναι σε στενότερη κλίμακα.
3. Δημιουργία Τεστ. Για την επίτευξη πιο ολοκληρωμένης κάλυψης δοκιμών υπάρχει η δυνατότητα η οποία βοηθάει τις ομάδες να προσθέσουν δοκιμαστικά σημεία ελέγχου, πχ. η επαλήθευση επιτυχών καταστάσεων απόκρισης HTTP στις κλήσεις API του Postman.
4. Αυτοματοποιημένη δοκιμή API. Υπάρχουν λειτουργίες όπως το Collection Runner, για την αυτοματοποίηση των δοκιμών API στο Postman, προς εξοικονόμηση χρόνου και πόρων.
5. Απλούστερη διαδικασία εύρεσης σφάλματος. Η εφαρμογή Postman απλοποιεί την διαδικασία εύρεσης σφαλμάτων σε δοκιμές API, δίνοντας τεράστια βοήθεια στις ομάδες να ελέγξουν τα δεδομένα που ανακτήθηκαν.
6. Συλλογές. Οι λειτουργίες Collections επιτρέπουν την ομαδοποίηση πολλών σχετικών API, για την οργάνωση δοκιμαστικών σουιτών.

3.12 Visual Studio Code

Το Visual Studio Code είναι ένα πρόγραμμα τελευταίας εξέλιξης για την επεξεργασία κώδικα το οποίο υποστηρίζει λειτουργίες ανάπτυξης, εντοπισμό σφαλμάτων, τρέξιμο εργασιών και έλεγχο έκδοσης. Στόχος του VSC είναι να παρέχει μόνο τα εργαλεία που χρειάζεται ένας προγραμματιστής για ταχύτερη δημιουργία κώδικα, άμεσο εντοπισμό σφαλμάτων και άλλα.

Το VS Code τρέχει σε macOS, Linux και Windows και είναι δωρεάν. Έχει πολλά χαρακτηριστικά και μπορεί κανείς να αναπτύξει τις εφαρμογές σε διάφορες γλώσσες όπως C#, VB.Net, C++ και ούτω καθεξής.

Είναι δημοφιλές λόγω της «εξάπλωσής του τομέα ανάπτυξης ιστοσελίδων αυτά τα χρόνια και λόγω της ανάγκης των προγραμματιστών να έχουν έναν ελαφρύ, καλά κατασκευασμένο πρόγραμμα επεξεργασίας, με λίγες δυνατότητες μα και λιγότερο περίπλοκο από τους υπόλοιπους της αγοράς

Ο κώδικας του Visual Studio δεν περιορίζεται σε μια συγκεκριμένη γλώσσα, επομένως είναι δύσκολη η διαχείριση των πρόσθετων όταν χρησιμοποιείται για πολλούς σκοπούς σε διαφορετικά περιβάλλοντα όπως python ή C++.

3.13 Nginx

Το Nginx είναι λογισμικό ανοιχτού κώδικα για web service, reverse proxy, caching, load balancing, media streaming και πολλά άλλα. Ξεκίνησε ως διακομιστής ιστού σχεδιασμένος για μέγιστη απόδοση και σταθερότητα.[21] Εκτός από τις δυνατότητες του διακομιστή HTTP, το Nginx μπορεί επίσης να λειτουργήσει ως διακομιστής μεσολάβησης για email (IMAP, POP3 και SMTP) και αντίστροφος διακομιστής μεσολάβησης και εξισορρόπησης φορτίου για διακομιστές HTTP, TCP και UDP.

Ο στόχος πίσω από το Nginx ήταν να δημιουργήσει τον ταχύτερο διακομιστή ιστού και να διατηρήσει αυτής την «αριστεία» εξακολουθώντας να αποτελεί κεντρικό στόχο του έργου. Το Nginx κερδίζει σταθερά τον Apache και άλλους διακομιστές σε σημεία αναφοράς που μετρούν την απόδοση του διακομιστή ιστού. Από την αρχική κυκλοφορία του Nginx, ωστόσο, οι ιστότοποι έχουν επεκταθεί από απλές σελίδες HTML σε δυναμικό, πολύπλευρο περιεχόμενο. Το Nginx αναπτύχθηκε μαζί τους και τώρα υποστηρίζει όλα τα στοιχεία του σύγχρονου Ιστού, συμπεριλαμβανομένων των WebSocket, HTTP/2, gRPC και ροής πολλαπλών μορφών βίντεο (HDS, HLS, RTMP και άλλα).

Αν και ο Nginx έγινε διάσημος ως ο ταχύτερος διακομιστής ιστού, η κλιμακούμενη υποκείμενη αρχιτεκτονική έχει αποδειχθεί ιδανική για πολλές εργασίες ιστού πέρα από την εξυπηρέτηση περιεχομένου. Επειδή μπορεί να χειριστεί μεγάλο όγκο συνδέσεων, το Nginx χρησιμοποιείται συνήθως ως αντίστροφος διακομιστής μεσολάβησης και εξισορρόπηση φορτίου για τη διαχείριση της εισερχόμενης κίνησης και τη διανομή της σε πιο αργούς διακομιστές upstream – από διακομιστές βάσεων δεδομένων παλαιού τύπου έως μικροϋπηρεσίες.

Το Nginx τοποθετείται επίσης συχνά μεταξύ πελατών και ενός δεύτερου διακομιστή ιστού, για να χρησιμεύσει ως τερματικός SSL/TLS ή επιταχυντής Ιστού. Λειτουργώντας ως ενδιάμεσος, το Nginx χειρίζεται αποτελεσματικά εργασίες που ενδέχεται να επιβραδύνουν τον διακομιστή ιστού, όπως η διαπραγμάτευση SSL/TLS ή η συμπίεση και αποθήκευση περιεχομένου για τη βελτίωση της απόδοσης. Οι δυναμικά κατασκευασμένοι ιστότοποι, χρησιμοποιώντας οτιδήποτε από Node.js έως PHP, συνήθως αναπτύσσουν το Nginx ως

προσωρινή μνήμη περιεχομένου και αντίστροφο διακομιστή μεσολάβησης για να μειώσουν το φόρτο στους διακομιστές εφαρμογών και να κάνουν πιο αποτελεσματική χρήση του υποκείμενου υλικού.

Το Nginx Plus και το Nginx είναι οι καλύτεροι στην κατηγορία σε λύσεις παράδοσης διακομιστή ιστού και εφαρμογών που χρησιμοποιούνται από ιστότοπους υψηλής επισκεψιμότητας όπως το Dropbox, το Netflix και το Zynga. Περισσότεροι από 350 εκατομμύρια ιστότοποι παγκοσμίως βασίζονται στο Nginx Plus και στο Nginx Open Source για να παρέχουν το περιεχόμενό τους γρήγορα, αξιόπιστα και με ασφάλεια.

Ως εξισορρόπηση φορτίου all-in-one μόνο για λογισμικό, διακομιστή ιστού, πύλη API και αντίστροφο διακομιστή μεσολάβησης που έχει σχεδιαστεί για εγγενείς αρχιτεκτονικές στο cloud, το Nginx βοηθά να επιταχύνετε τις προσπάθειες των υποδομών πληροφορικής και εκσυγχρονισμού εφαρμογών. Το Nginx Plus παρέχει δυνατότητες εταιρικής ποιότητας που παρέχουν ισχυρή αξιοπιστία και ασφάλεια.

Το Nginx είναι πολυλειτουργικό εργαλείο. Με το Nginx, μπορείτε να χρησιμοποιήσετε το ίδιο εργαλείο με το πρόγραμμα εξισορρόπησης φορτίου, τον αντίστροφο διακομιστή μεσολάβησης, την προσωρινή μνήμη περιεχομένου και τον διακομιστή ιστού, ελαχιστοποιώντας τον όγκο των εργαλείων και των διαμορφώσεων που χρειάζεται να διατηρήσει ο οργανισμός σας. Το Nginx προσφέρει τεκμηρίωση και μια ευρεία γκάμα ηλεκτρονικών βιβλίων, διαδικτυακών σεμιναρίων και βίντεο. Το Nginx Plus περιλαμβάνει υποστήριξη πελατών ταχείας απόκρισης, ώστε να μπορείτε εύκολα να λάβετε βοήθεια για τη διάγνωση οποιουδήποτε τμήματος της στοίβας σας, που χρησιμοποιεί Nginx ή Nginx Plus.

Το Nginx συνεχίζει να εξελίσσεται. Την τελευταία δεκαετία το Nginx βρισκόταν στην πρώτη γραμμή της ανάπτυξης του σύγχρονου Ιστού και συνέβαλε στην πρωτοπορία σε οτιδήποτε, από το HTTP/2 έως την υποστήριξη μικροϋπηρεσιών. Καθώς η ανάπτυξη και η παράδοση εφαρμογών Ιστού συνεχίζουν να εξελίσσονται, το Nginx Plus συνεχίζει να προσθέτει λειτουργίες για να επιτρέψει την άψογη παράδοση εφαρμογών, από υποστήριξη για διαμόρφωση με χρήση εφαρμογής JavaScript προσαρμοσμένης για Nginx, έως υποστήριξη δυναμικών μονάδων. Η χρήση του Nginx Plus διασφαλίζει ότι θα παραμείνετε στην αιχμή στο θέμα της απόδοσης ιστού.

Από τον Ιούνιο του 2022, ο αριθμός των διακομιστών ιστού της W3Tech από όλους τους ιστότοπους κατέταξε το Nginx πρώτο με 33,6%. Ο Apache ήταν δεύτερος με 31,4% και ο Cloudflare Server τρίτος με 21,6%. Από τον Μάρτιο του 2022, η Netcraft υπολόγισε ότι το Nginx εξυπηρετούσε το 22,01% μεταξί εκατομμυρίων των πιο πολυσύχναστων ιστότοπων με τον Apache λίγο πιο μπροστά στο 23,04%. Το Cloudflare με 19,53% και το Microsoft Internet

Information Services στο 5,78% ολοκλήρωσαν τους τέσσερις πρώτους διακομιστές για τους πιο πολυσύχναστους ιστότοπους. Ορισμένα από τα άλλα στατιστικά στοιχεία της Netcraft δείχνουν το Nginx μπροστά από τον Apache.

Στην έκδοση 5.2 του OpenBSD (Νοέμβριος 2012), το Nginx έγινε μέρος του βασικού συστήματος OpenBSD, παρέχοντας μια εναλλακτική λύση του συστήματος Apache 1.3, το οποίο προοριζόταν να αντικαταστήσει, αλλά αργότερα στην έκδοση 5.6 (Νοέμβριος 2014) καταργήθηκε χάρη του httpd του OpenBSD (8).

Το Nginx είναι εύκολο να διαμορφωθεί για να εξυπηρετεί στατικό περιεχόμενο ιστού ή να λειτουργεί ως διακομιστής μεσολάβησης. Το Nginx μπορεί να αναπτυχθεί για να εξυπηρετήσει επίσης δυναμικό περιεχόμενο στο δίκτυο χρησιμοποιώντας FastCGI, χειριστές SCGI για σενάρια, διακομιστές εφαρμογών WSGI ή λειτουργικές μονάδες Phusion Passenger και μπορεί να χρησιμεύσει ως εξισορρόπηση φόρτου λογισμικού. Το Nginx χρησιμοποιεί μια ασύγχρονη προσέγγιση που βασίζεται σε συμβάντα, αντί για νήματα, για να χειριστεί αιτήματα. Η αρθρωτή αρχιτεκτονική του Nginx που βασίζεται σε συμβάντα μπορεί να παρέχει προβλέψιμη απόδοση κάτω από υψηλό φόρτο.

Το Nginx φτιάχτηκε με ρητό στόχο την καλύτερη απόδοση του διακομιστή ιστού Apache. Εξυπηρετώντας στατικά αρχεία, το Nginx χρησιμοποιεί πολύ λιγότερη μνήμη από τον Apache και μπορεί να χειριστεί περίπου τέσσερις φορές περισσότερα αιτήματα ανά δευτερόλεπτο. Ωστόσο, αυτή η ενίσχυση απόδοσης έχει κόστος τη μειωμένη ευελιξία, όπως η δυνατότητα παράκαμψης των ρυθμίσεων πρόσβασης σε όλο το σύστημα ανά αρχείο (το Apache το επιτυγχάνει με ένα αρχείο .htaccess, ενώ το Nginx δεν έχει ενσωματωμένο τέτοιο χαρακτηριστικό).

Παλαιότερα, η προσθήκη λειτουργικών μονάδων τρίτου κατασκευαστή στο Nginx απαιτούσε την εκ νέου «μεταγλώττιση» της εφαρμογής από την πηγή με τις ενότητες συνδεδεμένες στατικά. Αυτό ξεπεράστηκε εν μέρει στην έκδοση 1.9.11 τον Φεβρουάριο του 2016, με την προσθήκη δυναμικής φόρτωσης λειτουργικών μονάδων. Ωστόσο, οι ενότητες πρέπει ακόμα να μεταγλωττίζονται ταυτόχρονα με το Nginx και δεν είναι όλες οι μονάδες συμβατές με αυτό το σύστημα. Ορισμένα απαιτούν την παλαιότερη διαδικασία στατικής σύνδεσης.

Το Nginx θεωρείται γενικά λιγότερο σταθερό στον Windows Server από ότι στο Linux, ενώ ο Apache έχει την ίδια υποστήριξη και για τα δύο.

3.14 Vue.js

Το Vue είναι μία δομή JavaScript για τη δημιουργία διεπαφών χρήστη. Είναι βασισμένο σε HTML, CSS και JavaScript και παρέχει ένα μοντέλο προγραμματισμού που βοηθά στην αποτελεσματική ανάπτυξη των διεπαφών χρήστη, είτε είναι απλές είτε σύνθετες.[22]

Πρόκειται λοιπόν για ένα JavaScript ανοιχτού κώδικα που χρησιμοποιείται κυρίως για τη δημιουργία UI και εφαρμογών σελίδας. Είναι συμβατό με τις περισσότερες σύγχρονες τεχνολογίες και λόγω της εύκολης εκμάθησης και επεκτασιμότητας, κέρδισε μεγάλη δημοτικότητα. Ακολουθεί το μοτίβο Model-View-ViewModel (MVVM), όπου το ViewModel έχει μια παρουσία «Vue» και το View και το Model δεσμεύονται από αμφίδρομη σύνδεση δεδομένων. Χρησιμοποιεί εικονικό DOM και όσον αφορά το API και τη σχεδίαση, το Vue είναι εύκολο στην εκμάθηση σε σύγκριση με το AngularJS κ.α.

Το Vue παρέχει διάφορους τρόπους για την εφαρμογή εφέ μετάβασης όταν εισάγονται, ενημερώνονται ή αφαιρούνται στοιχεία από το DOM. Αυτό περιλαμβάνει εργαλεία για:

1. Αυτόματη εφαρμογή κλάσεων για μεταβάσεις και κινούμενα σχέδια CSS
2. Ενσωματώστε βιβλιοθήκες κινούμενων σχεδίων CSS τρίτων, όπως το Animate.css
3. Χρησιμοποιήστε JavaScript για να χειριστείτε απευθείας το DOM κατά τη διάρκεια των αγκίστρων μετάβασης
4. Ενσωματώστε βιβλιοθήκες κινούμενων σχεδίων JavaScript τρίτων, όπως το Velocity.js

Όταν ένα στοιχείο “τυλιγμένο” σε ένα στοιχείο μετάβασης εισάγεται ή αφαιρείται, συμβαίνει αυτό:

1. Το Vue θα μυρίσει αυτόματα εάν το στοιχείο προορισμού έχει εφαρμοστεί μεταβάσεις CSS ή κινούμενα σχέδια. Εάν συμβεί αυτό, οι κλάσεις μετάβασης CSS θα προστεθούν/αφαιρούνται σε κατάλληλους χρόνους.
2. Εάν το στοιχείο μετάβασης παρείχε αγκίστρια JavaScript, αυτά τα άγκιστρα θα κληθούν σε κατάλληλους χρονισμούς.
3. Εάν δεν εντοπιστούν μεταβάσεις/κινούμενα σχέδια CSS και δεν παρέχονται άγκιστρα JavaScript, οι λειτουργίες DOM για εισαγωγή ή/και αφαίρεση θα εκτελεστούν αμέσως στο επόμενο πλαίσιο.

3.15 Apptainer

Το Apptainer είναι μια πλατφόρμα περιέκτη ανοιχτού κώδικα σχεδιασμένη να είναι απλή, γρήγορη και ασφαλής.[23] Πολλές πλατφόρμες περιεκτών είναι διαθέσιμες, αλλά το Apptainer έχει σχεδιαστεί με γνώμονα την ευκολία στη χρήση σε κοινόχρηστα συστήματα και σε περιβάλλοντα υπολογιστών υψηλής απόδοσης (HPC).

Χαρακτηριστικά:

- Μια αμετάβλητη μορφή εικόνας περιέκτη ενός αρχείου, που υποστηρίζει κρυπτογραφικές υπογραφές και κρυπτογράφηση.
- Ενσωμάτωση πάνω από απομόνωση από προεπιλογή. Χρησιμοποιήστε εύκολα GPU, δίκτυα υψηλής ταχύτητας, παράλληλα συστήματα αρχείων σε ένα σύμπλεγμα ή διακομιστή.
- Κινητικότητα υπολογιστών. Η μορφή περιέκτη SIF ενός αρχείου είναι εύκολη στη μεταφορά και κοινή χρήση.
- Ένα απλό, αποτελεσματικό μοντέλο ασφάλειας. Ο ίδιος χρήστης εντός ενός περιέκτη όπως και εκτός δεν μπορεί να αποκτήσει επιπλέον δικαιώματα στο κεντρικό σύστημα από προεπιλογή.

Το Apptainer ήταν παλαιότερα γνωστό ως Singularity και τώρα αποτελεί μέρος του Linux. Παρέχει περιβάλλοντα με περιέκτες. Ένα χαρακτηριστικό παράδειγμα είναι το Docker, το οποίο είναι επίσης λογισμικό περιεκτών. Περιέκτης λοιπόν είναι ένα συσκευασμένο λειτουργικό σύστημα με εγκατεστημένο λογισμικό που είναι «απομονωμένο» από το κύριο λειτουργικό μας σύστημα. Συνήθως τα VM παραπονιούνται για αργή λειτουργία και το κάνουν επειδή είναι ένα πλήρες αντίγραφο όλων όσων εκτελούνται σε έναν εικονικό υπολογιστή με αποκλειστικούς πόρους (RAM, πυρήνες CPU, HDD) που είναι λιγότεροι από το βασικό σύστημα. Οι περιέκτες διαφέρουν εδώ επειδή εκτελούν μόνο το υποσύνολο του συστήματος που χρειάζεται, πράγμα που συνήθως σημαίνει όχι τόσες πολλές διεργασίες ή ένα GUI (σκεφτείτε ένα λειτουργικό σύστημα χωρίς κεφάλι). Επίσης, υπάρχει μικρότερο σημείο συμφόρησης επειδή δεν αφιερώνονται πόροι σε αυτό, αντίθετα, μοιράζονται δυναμικά με το λειτουργικό σύστημα. Επομένως, εάν ο περιέκτης χρησιμοποιεί πολλούς υπολογιστικούς κύκλους ενώ το λειτουργικό σύστημα είναι σε αδράνεια, μπορεί να χρειαστεί περισσότερος χρόνος.

4. Σχεδιασμός

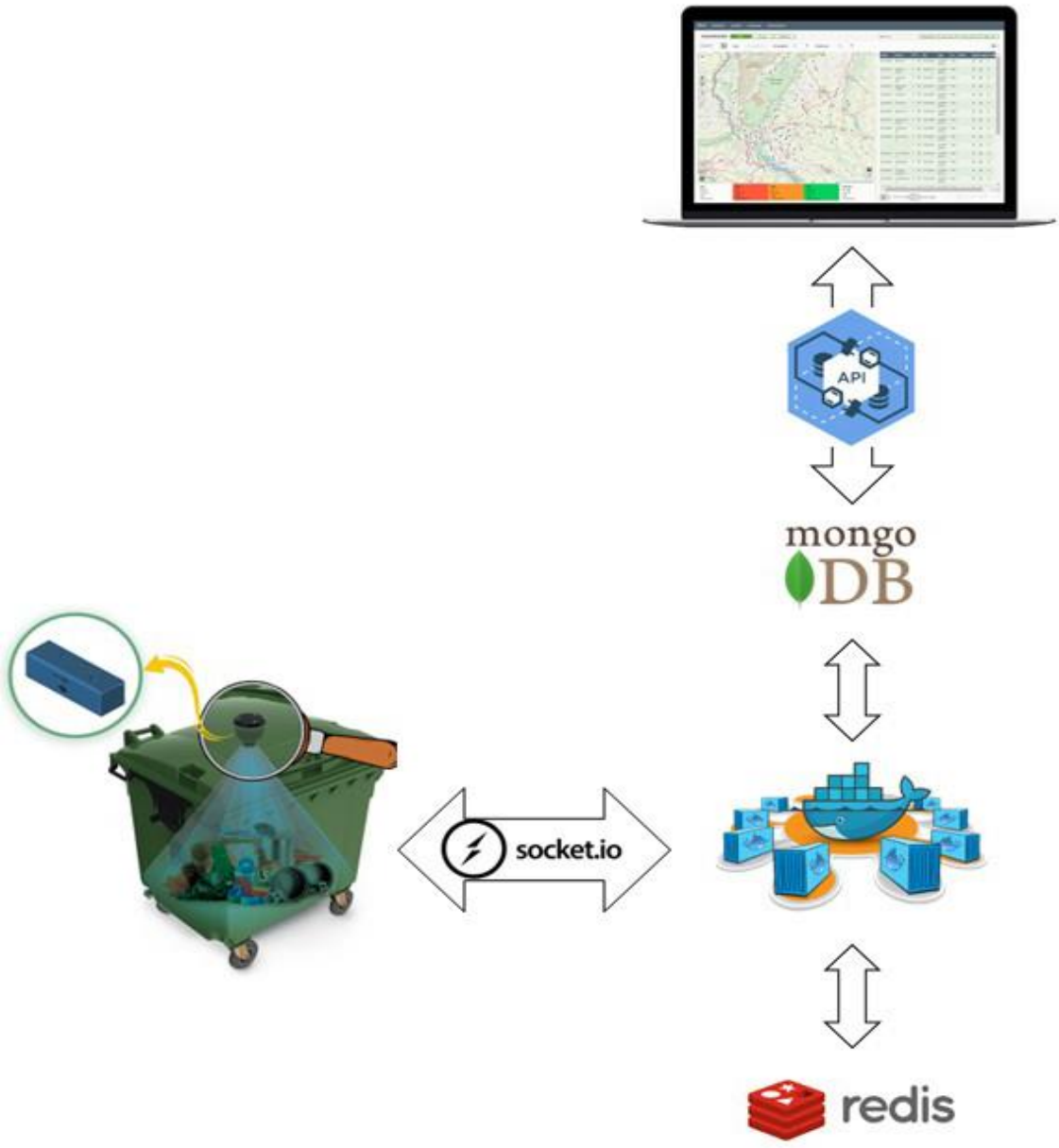
4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιαστεί ο σχεδιασμός της διπλωματικής για εφαρμογή όλων των παραπάνω τεχνολογιών σε «Έξυπνους Κάδους Απορριμμάτων» (Smart Waste Bin) με την χρήση αισθητήρων διαφόρων ειδών που θα μας ενημερώνουν και θα μας ειδοποιούν για κάθε συμβάν ή αλλαγή κατάστασης που θα αφορά τις λειτουργίες των τριών αισθητήρων που αναλύονται στο κεφάλαιο 4.2. Επίσης θα παρουσιαστεί η έρευνα που διεξήχθη και αφορά την λειτουργικότητα των αισθητήρων που παρατίθενται στον πίνακα «Άλλοι αισθητήρες και λειτουργίες» «Πίνακας 4.1».

Ένα ολοκληρωμένο σύστημα έξυπνων κάδων περιλαμβάνει τα παρακάτω συστατικά και λειτουργίες:

- Κάδους που έχουν ενσωματωμένο μικροελεγκτή με αισθητήρες, ενεργοποιητές και μικρό λειτουργικό σύστημα. Στον μικροελεγκτή, μέσω περιέκτη, εκτελείται η μικροϋπηρεσία που συλλέγει τις τιμές των αισθητήρων και τις αποστέλλει μέσω WebSocket πρωτοκόλλου στον περιέκτη που εξυπηρετεί τον κεντρικό κάδο.
- Κεντρικό κάδο στον οποίο μέσω περιέκτη εκτελείται η μικροϋπηρεσία που λαμβάνει τις τιμές των αισθητήρων. Οι τιμές αποθηκεύονται σε μία γρήγορη προσωρινή βάση δεδομένων μνήμης redis και σε μία μακράς διαρκείας βάση δεδομένων εγγράφων mongoDB. Και οι δύο βάσεις βρίσκονται μέσα σε περιβάλλον περιέκτη docker που διατίθεται δωρεάν από το DockerHub. Το πρόγραμμα επεξεργάζεται το είδος της πληροφορίας που λαμβάνει και σε περίπτωση που η τιμή του αισθητήρα φωτιάς υποδεικνύει πυρκαγιά τότε αποστέλλει εντολή εκκίνησης του ενεργοποιητή πυρόσβεσης στον προβληματικό κάδο. Η επικοινωνία για να είναι άμεση γίνεται μέσω WebSocket πρωτοκόλλου προς τον περιέκτη που εξυπηρετεί τον συγκεκριμένο κάδο από τον οποίο έλαβε την πληροφορία ο κεντρικός κάδος.
- Διεπαφή προγραμματισμού εφαρμογών (API) που υλοποιείται σε περιέκτη η οποία έχει πλήρεις δυνατότητες CRUD δηλαδή δημιουργίας – ανάγνωσης – ενημέρωσης και διαγραφής τιμών κάδων - αισθητήρων στην βάση δεδομένων mongoDB. Το API μπορεί να φιλοξενηθεί σε οποιαδήποτε cloud ή fog υποδομή υποστηρίζει χρήση περιεκτών.
- Web εφαρμογή μέσω περιέκτη η οποία χρησιμοποιώντας το API προσφέρει στον χρήστη ένα γραφικό περιβάλλον εύκολης διαχείρισης των τιμών αισθητήρων των κάδων. Η

εφαρμογή βασισμένη στη βιβλιοθήκη Vue.js προσφέρει όμορφο και λειτουργικό περιβάλλον εμφάνισης και επεξεργασίας των δεδομένων (Σχήμα 4.1).



Σχήμα 4.1: Απεικόνιση κυκλώματος έξυπνου κάδου

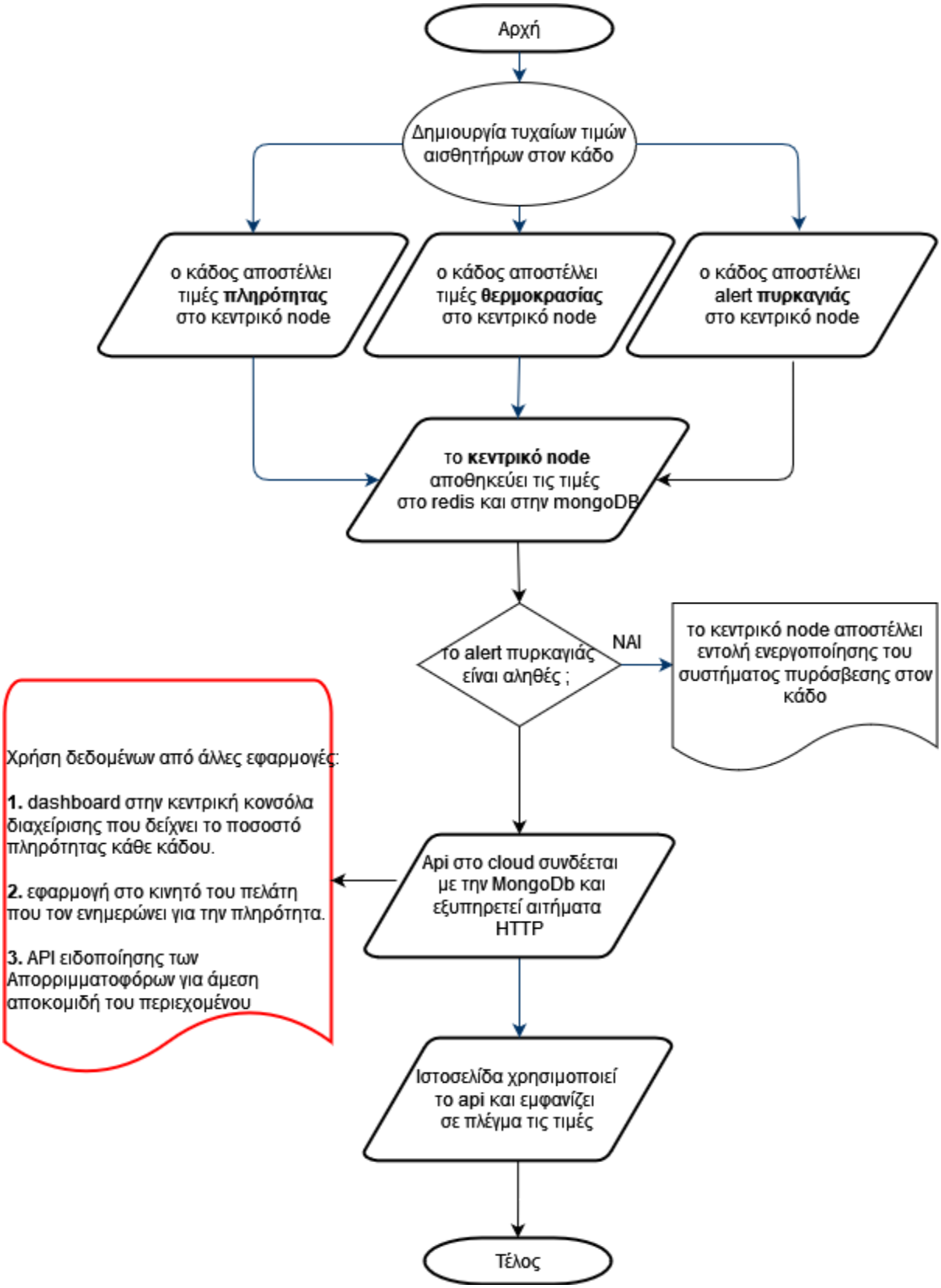
Η λογική αλληλουχία των ενεργειών που πρέπει να διεκπαιρεώσει η εφαρμογή αποτυπώνεται στο λογικό διάγραμμα στου σχήματος (Σχήμα 4.2).

Αφότου επιτευχθεί η επικοινωνία του περιφερειακού κάδου μέσω socket με τον κεντρικό κάδο ξεκινάει η εκτέλεση των αλγορίθμων που παράγουν τυχαίες τιμές αισθητήρων στον περιφερειακό κάδο. Για τον αισθητήρα πληρότηρας παράγεται κάθε τρία δευτερόλεπτα μία ακέραια τιμή μεταξύ των ορίων 0% και 10% που προστίθεται στην προηγούμενη τιμή. Για τον αισθητήρα φωτιάς παράγεται κάθε ένα δευτερόλεπτο μία δυαδική τιμή που σε τυχαίο ποσοστό 1% είναι αληθής και σε πσοστό 99% είναι ψευδής. Για τον αισθητήρα θερμοκρασίας παράγεται κάθε τρία δευτερόλεπτα μία τιμή μεταξύ των ορίων -10C° και +10C° που προστίθεται στην προηγούμενη τιμή. Οι τρεις τιμές που δημιουργήθηκαν αποστέλλονται μέσω του ανοικτού καναλιού Socket στην εφαρμογή του κεντρικού κάδου - node.

Το κεντρικό node ελέγχει το κανάλι και όταν λάβει τιμές τις αποθηκεύει στην βάση δεδομένων μνήμης Redis και στην βάση δεδομένων εγγράφων MongoDB με σήμανση για την Ημερομηνία και Ώρα του συμβάντος και το όνομα του κάδου. Παράλληλα ελέγχει αν η τιμή του αισθητήρα φωτιάς είναι αληθής και τότε παράγει εντολή ενεργοποίησης του συστήματος αυτόματης πυρόσβεσης και την αποστέλλει μέσω του ίδιου καναλιού στον περιφερειακό κάδο.

Διεπαφή προγραμματισμού εφαρμογών (API) συνδέεται με την βάση δεδομένων mongoDB με πλήρη δικαιώματα διαχείρισης.

Web εφαρμογή χρησιμοποιεί το API μέσω αιτημάτων Http για να αποκτήσει πρόσβαση στα δεδομένα της MongoDB προσφέρωντας στον χρήστη ένα γραφικό περιβάλλον εύκολης διαχείρισης των τιμών αισθητήρων των κάδων.



Σχήμα 4.2: Cloud Computing

4.2 Ανάλυση λειτουργιών αισθητήρων

4.2.1 Αισθητήρας Πυρκαγιάς - Fire Sensor

Ο αισθητήρας εκδήλωσης πυρκαγιάς στον κάδο είναι για την ενημέρωση του κέντρου αλλά και της ενεργοποίησης κάποιου συστήματος πυρόσβεσης ή ξεκλειδώματος σε περίπτωση που υποστηρίζεται από τον εξοπλισμό του κάδου.

Η ανταλλαγή μηνυμάτων με την εφαρμογή δεν έχει κάποια συχνότητα αλλά με την ενεργοποίηση του αισθητήρα, άμεσα αποστέλλεται ειδοποίηση για τις περεταίρω ενέργειες. Η σειρά των ενεργειών αποφασίζεται από τον διαχειριστή του συστήματος σε συνεργασία με τον εξοπλισμό που διατίθεται αλλά και την οργάνωση και σχεδίαση του συστήματος. Μια τυπική σειρά ενεργειών είναι

- Ενεργοποίηση αυτόματου συστήματος πυρόσβεσης,
- Ειδοποίηση Πυροσβεστικής Υπηρεσίας,
- Ειδοποίηση Εταιρείας – Δήμου,
- Ενημέρωση χρήστη (όσων είναι πλησίον)
- Ενεργοποίηση απεμπλοκής του καλύμματος του κάδου για την περίπτωση που δεν έχει σύστημα πυρόσβεσης ή η ενέργεια απέτυχε.

Ο τρόπος υλοποίησης των ενεργειών (Actions) στην εφαρμογή αναλύεται ως εξης:
Έναρξη με αποστολή εντολής στο node του κάδου για ενεργοποίηση actuator συστήματος αυτόματης πυρόσβεσης του κάδου και ενημέρωση της εφαρμογής ειδοποίησης πυρκαγιών της Πυροσβεστικής Υπηρεσίας. Στη συνέχεια ενημέρωση της κεντρικής εφαρμογής ελέγχου της εταιρείας – Δήμου και αποστολή notification στην mobile εφαρμογή του χρήστη για την πυρκαγιά και το σημείο του κάδου για τη μη χρήση του, ή για όποια βοηθητική παρέμβαση μπορεί να γίνει. Ενημέρωση του status του κάδου στην mobile εφαρμογή, αποστολή sms στην κινητή συσκευή του χρήστη και αποστολή εντολής στο node του κάδου για εκκίνηση ενεργοποιητή (actuator) κλειδώματος του κάδου ή ξεκλειδώματος για παρέμβαση από εξωτερικούς παράγοντες.

4.2.2 Αισθητήρας Πληρότητας - Ultrasonic sensor – Laser

Ο αισθητήρας ελέγχου πληρότητας του κάδου είναι για την ενημέρωση του κέντρου σχετικά με την διαθεσιμότητα του κάθε κάδου προς χρήση, για τον έλεγχο του σε περίπτωση πλήρωσης (π.χ. να κλειδώνεται και να μην δέχεται άλλα απορρίμματα) αλλά και να μπαίνει σε δρομολόγηση η περισυλλογή του από κάποιο απορριμματοφόρο. Οι τιμές πληρότητας του κάδου αποστέλλονται στην εφαρμογή περιοδικά κάθε 10 λεπτά οπού γίνεται καταγραφή σε βάση δεδομένων και ανάλυση για εξαγωγή χρήσιμων συμπερασμάτων - ενεργειών και δράσεων. Ακολουθεί ενημέρωση του χρήστη για την κατάσταση πληρότητας του κάδου που χρησιμοποιείται από αυτόν συνήθως για μη διαθεσιμότητα, ενεργοποιείται ο μηχανισμός κλειδώματος του κάδου και ενημέρωση των απορριμματοφόρων για περισυλλογή του.
Η σειρά των ενεργειών αποφασίζεται από τον διαχειριστή του συστήματος βάση της οργάνωσης και σχεδίασης του συστήματος.
Αναλύοντας λοιπόν τον τρόπο υλοποίησης του Action στην εφαρμογή και με την έναρξη της, η εφαρμογή εμφανίζει όλους τους κάδους και τα ποσοστά πληρότητας. Στη Web εφαρμογή εμφανίζονται γραφικές παραστάσεις ποσοστών πληρότητας σε συσχέτιση με τον χρόνο, γίνεται αποστολή notification στην mobile εφαρμογή του χρήστη, ενημέρωση του status του κάδου και αποστολή sms στην κινητή συσκευή του χρήστη.

- Αποστολή εντολής στο node του κάδου για ενεργοποίηση actuator κλειδώματος του κάδου
- Ενεργοποίηση ένδειξης led λυχνίας κλειστού κάδου (κόκκινο).
- Ενημέρωση του status του κάδου στην εφαρμογή του απορριμματοφόρου
- Ενημέρωση της κεντρικής εφαρμογής ελέγχου της εταιρείας διαχείρισης – Δήμου
- Ενημέρωση του απορριμματοφόρου από την κεντρική εφαρμογή ελέγχου της εταιρείας διαχείρισης – Δήμου.

Με λίγα λόγια, το προτεινόμενο σύστημα συλλογής απορριμμάτων βασίζεται σε δεδομένα στάθμης απορριμμάτων από κάδους απορριμμάτων σε μια μητροπολιτική περιοχή. Τα δεδομένα που συλλέγονται από τους αισθητήρες αποστέλλονται μέσω του Διαδικτύου σε έναν διακομιστή όπου αποθηκεύονται και υποβάλλονται σε επεξεργασία. Τα δεδομένα που συλλέγονται χρησιμοποιούνται στη συνέχεια για την παρακολούθηση και τη βελτιστοποίηση της καθημερινής επιλογής των κάδων απορριμμάτων που θα συλλέγονται, υπολογίζοντας ανάλογα τις διαδρομές.

4.2.3 Αισθητήρας Θερμοκρασίας - Temperature sensor

Ο αισθητήρας ενημέρωσης θερμοκρασίας του εσωτερικού του κάδου, είναι για την ενημέρωση του κέντρου για τον κάθε κάδο που είναι προς χρήση, σε περίπτωση ακραίων θερμοκρασιών να ενεργοποιούνται κάποια πρωτόκολλα, (π.χ. να κλειδώνεται και να μην δέχεται άλλα απορρίμματα) και να μπαίνει σε δρομολόγηση η περισυλλογή του από κάποιο απορριμματοφόρο. Οι τιμές από τον κάδο αποστέλλονται περιοδικά κάθε 30`` και με την επισήμανση του alert πολύ υψηλής θερμοκρασίας να ενεργοποιείται ο μηχανισμός αυτόματης πυρόσβεσης ακόμα και πριν την σήμανσή από τον αισθητήρας πυρκαγιάς.

Πίνακας 4.1: Άλλοι αισθητήρες και λειτουργίες

	Humidity sensor	**RFID tag**	**NFC tag**
Trigger	Τιμές υγρασίας του κάδου	Ανίχνευση συσκευής χρήστη	Ανίχνευση συσκευής χρήστη
Time	Περιοδικά κάθε 30``	Την στιγμή ενεργοποίησης του sensor	Την στιγμή ενεργοποίησης του sensor
Action	Έλεγχος πληρότητας σε υγρά ή υγρασία.	Έλεγχος - ταυτοποίηση χρήστη για την χρήση του κάδου	Έλεγχος - ταυτοποίηση χρήστη για την χρήση του κάδου
Λειτουργικότητα	Ενημέρωση για πλημμύρισμα κάδου.	Ταυτοποίηση χρήστη για χρήση	Ταυτοποίηση χρήστη για χρήση

Περιγραφή Λειτουργικότητας	Άνοιγμα βαλβίδας αποστράγγισης για την μη πρόκληση ζημιάς στο απορριμματοφόρο με την εισροή υγρών.	Προϋποθέτει την κατοχή κάρτας πιστοποιημένου χρήστη (διάθεση από Δήμο αναλόγως την περιοχή διαμονής)	Προϋποθέτει την κατοχή κινητού με NFC και εισαγωγής σε αυτό λογαριασμού δοθέντος απο Δήμο.

	Weight-meter	**GPS**	**GAS detector**	**Accelerometer**
Trigger	Τιμές βάρους του περιεχομένου στον κάδο	Τιμή σημείου GPS του κάδου = σταθερή	τιμή μεθανίου ή CO2 > όριο	Τιμή επιτάχυνσης ή ταχύτητας κίνησης κάδου
Time	Περιοδικά κάθε 10 λεπτά	Περιοδικά κάθε 30``	Περιοδικά κάθε 10 λεπτά	Περιοδικά κάθε 3 λεπτά
Action	Καταγραφή σε βάση δεδομένων και ανάλυση για εξαγωγή χρήσιμων συμπερασμάτων - ενεργειών και δράσεων.	Καταγραφή σε βάση δεδομένων και ανάλυση για εξαγωγή χρήσιμων συμπερασμάτων. Αν έχουμε αλλαγή τιμών και πιστοποιημένο όχημα τότε ο κάδος είναι σε κίνηση - πλύσιμο και πορεία προς νέα θέση.	Alert - Ενεργοποίηση μηχανισμού απορρόφησης του μεθανίου και αποθήκευσης του σε δοχεία	Alert - Ενεργοποίηση του κέντρου για εύρεση του κάδου σε περίπτωση μετακίνησής του.
Τρόπος υλοποίησης Action	1. Ενημέρωση του status του κάδου στην εφαρμογή του απορριμματοφόρου 2. Ενημέρωση της κεντρικής εφαρμογής ελέγχου της εταιρείας διαχείρισης - Δήμου 3. Ενημέρωση του απορριμματοφόρου από την κεντρική εφαρμογή ελέγχου της εταιρείας διαχείρισης - Δήμου	1. Web εφαρμογή που εμφανίζει όλους τους κάδους, αν βρίσκονται σε κίνηση ή όχι και σε ποιο σημείο γεωγραφικά βρίσκεται ο καθένας. 2. Web εφαρμογή που εμφανίζει ποιοι κάδοι έχουν καθαριστεί και πότε. 3. Web εφαρμογή που δημιουργεί πλάνο - διαδρομή καθαρισμού κάδων και ενημερώνει τα πλυστικά οχήματα.		Ενημέρωση της κεντρικής εφαρμογής ελέγχου της εταιρείας για λόγους απώλειας

Λειτουργικότητα	Πληρότητα λόγω υπερβολικού βάρους το οποίο δεν συμβαδίζει με τον όγκο των απορριμμάτων και μπορεί να προκληθεί ζημιά από αντικείμενα που έχουν τοποθετηθεί είτε στον ίδιο τον κάδο είτε στο απορριμματοφόρο που θα τα συλλέξει (πχ μπάζα).	Περίπτωση αλλαγής θέσης λόγω πλύσης		Ενημέρωση για απομάκρυνση κάδου.
Περιγραφή Λειτουργικότητας		Το όχημα περισυλλέγει τον άδειο κάδο #1 από το σημείο #1. Το όχημα κατευθύνεται προς τον επόμενο κάδο #2 που βρίσκεται στο σημείο #2. Στην πορεία πλένει τον κάδο #1. Όταν φτάσει στο σημείο #2 περισυλλέγει τον άδειο κάδο #2 και στην θέση του τοποθετεί τον καθαρό κάδο #1.Το όχημα κατευθύνεται προς τον επόμενο κάδο #3 που βρίσκεται στο σημείο #3.		

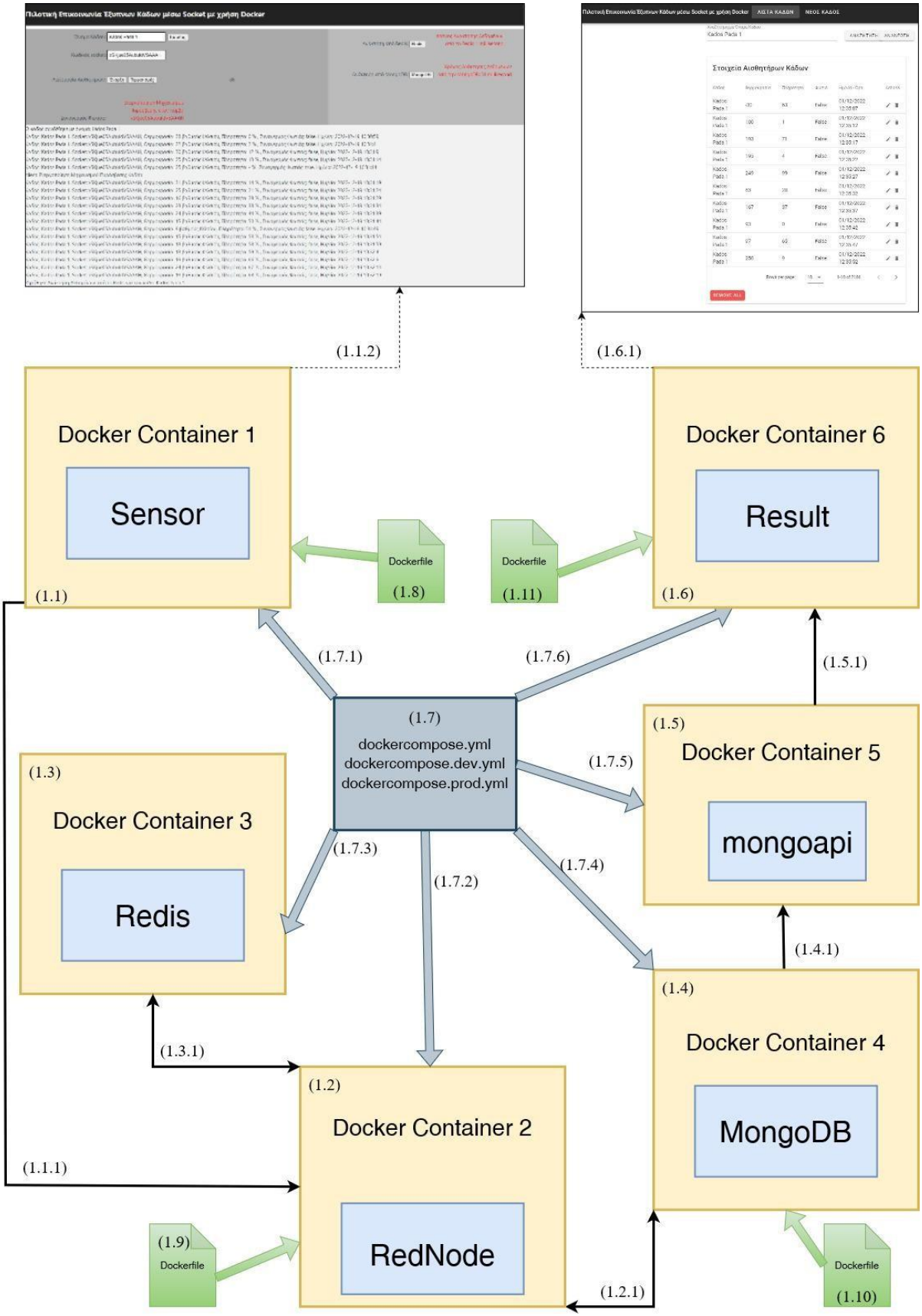
5. Υλοποίηση

5.1 Εισαγωγή

Η υλοποίηση αποτελείται από δύο προγράμματα παραγωγής και εμφάνισης δεδομένων (sensor και result), δύο βάσεις δεδομένων στη μνήμη και σε αποθηκευτικό χώρο (redis και mongodb), ένα πρόγραμμα διαχείρισης της επικοινωνίας και της μεταφοράς δεδομένων προς και από τις βάσεις δεδομένων και λήψης αποφάσεων (RedNode), και ένα πρόγραμμα application programming interface για την διαχείριση και διοχέτευση των δεδομένων της βάσης MongoDB (mongoapi).

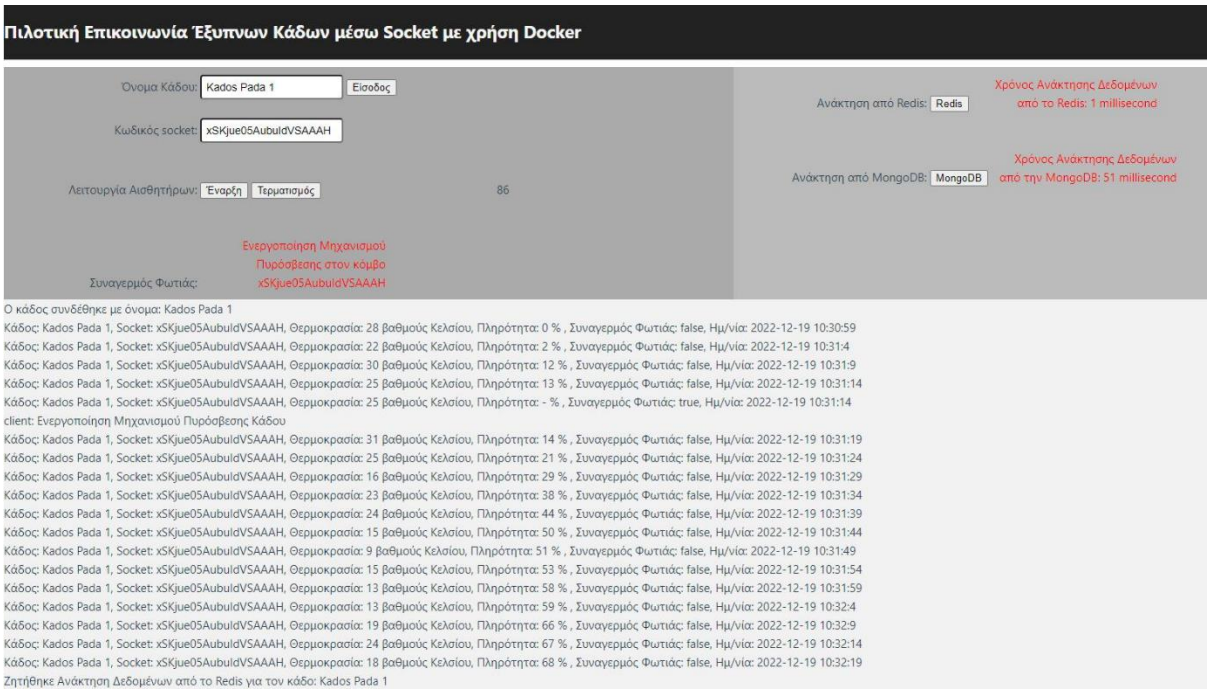
Τα τέσσερα προγράμματα και οι δύο βάσεις δεδομένων εσωκλείονται σε έξι ξεχωριστούς Docker περιέκτες (containers), με χρήση των αρχείων παραμετροποίησης Dockerfile και Docker Compose και επικοινωνούν μεταξύ τους με τρόπο που θα αναλύσουμε παρακάτω.

Το αριθμητικό σχήμα που ακολουθεί (π.χ. (1.1), (1.2), (3.1) κ.λ.π.), αντιστοιχεί στην αρίθμηση που υπάρχει στο σχήμα (Σχήμα 5.1).



Σχήμα 5.1: Σχηματικό διάγραμμα των τμημάτων της εφαρμογής

(1.1) “Docker Container 1”: ο περιέκτης εκτελεί το πρόγραμμα διακομιστή σε περιβάλλον Node.js με όνομα «sensor» που επικοινωνεί με τον browser πάνω αριστερά.(Σχήμα 5.1)



Σχήμα 5.2: Ιστοσελίδα διαχείρισης της λειτουργίας των αισθητήρων των κάδων της εφαρμογής πιλοτικής επικοινωνίας έξυπνων κάδων

Αριστερό τμήμα σελίδας: Κάθε φορά που συμπληρώνουμε το πεδίο «Όνομα Κάδου» και πατάμε το πλήκτρο «Είσοδος» ξεκινάει μία socket επικοινωνία με τον διακομιστή Node.js που εκτελείται στο «Docker Container 2» με όνομα RedNode η οποία έχει ένα μοναδικό κωδικό που μέσω του ανοιχτού διαύλου επιστρέφει στο «Docker Container 1» και εμφανίζεται στο πεδίο «Κωδικός socket». Για να ενεργοποιηθεί η λειτουργία των αισθητήρων του κάδου πρέπει να πατήσουμε το κουμπί «Έναρξη». Με το πάτημα του ενεργοποιείται μηχανισμός παραγωγής και αποστολής τιμών από τους εικονικούς αισθητήρες προς τον διακομιστή RedNode. Ο μηχανισμός αποτελείται από πέντε μέρη. Το πρώτο παράγει κάθε πέντε δευτερόλεπτα τυχαίες τιμές θερμοκρασίας μεταξύ των ορίων -10 C και +10 C τις οποίες στην συνέχεια τις προσθέτει στην προηγούμενη τιμή θερμοκρασίας σχηματίζοντας την τελική τιμή. Το δεύτερο παράγει κάθε πέντε δευτερόλεπτα τυχαίες τιμές ποσοστού πληρότητας μεταξύ των ορίων 0% και +10%, τις οποίες στην συνέχεια τις προσθέτει στην προηγούμενη τιμή ποσοστού πληρότητας και μηδενίζει την τιμή όταν ξεπεραστεί το όριο του 90%. Το τρίτο παράγει κάθε ένα δευτερόλεπτο τυχαίες δυικές τιμές συναγερμού φωτιάς μεταξύ των αληθής ή ψευδής σε

ποσοστό 98% ή 2% αντίστοιχα. Το τέταρτο αποστέλλει κάθε πέντε δευτερόλεπτα τα δεδομένα Όνομα κάδου, Κωδικός Socket, Ποσοστό Πληρότητας, Τιμή Θερμοκρασίας και τρέχουσα Ημερομηνία – Ώρα στην κεντρική εφαρμογή RedNode. Το πέμπτο ενεργοποιεί λειτουργία χρονομετρητή που καταγράφει και εμφανίζει σε δευτερόλεπτα το χρόνο που εκτελούνται οι άνωθεν λειτουργίες έως να πατηθεί το πλήκτρο «Τερματισμός».

Δεξί τμήμα σελίδας: Πατώντας το πλήκτρο «Redis» αποστέλλεται στο rednode μέσω socket ένα αίτημα ανάκτησης των δεδομένων που υπάρχουν αποθηκευμένα στο Redis και αφορούν τον κάδο που αναφέρεται στο πεδίο «Όνομα Κάδου». Ο χρόνος ανάκτησης των δεδομένων χρονομετρείται milliseconds και η τιμή του επιστρέφεται στο πρόγραμμα sensor και εμφανίζεται στην οθόνη. Πατώντας το πλήκτρο «MongoDB» αποστέλλεται στο rednode μέσω socket ένα αίτημα ανάκτησης των δεδομένων που υπάρχουν αποθηκευμένα στην βάση Mongodb και αφορούν τον κάδο που αναφέρεται στο πεδίο «Όνομα Κάδου». Ο χρόνος ανάκτησης των δεδομένων χρονομετρείται σε milliseconds και η τιμή του επιστρέφεται στο πρόγραμμα sensor και εμφανίζεται στην οθόνη.

(1.8) “Dockerfile” — ορίζει και ελέγχει την διαδικασία του διακομιστή Node.js που περιέχεται στο “Docker Container 1”.

(1.7, 1.7.1, 1.7.2, 1.7.3, 1.7.4, 1.7.5, 1.7.6) “docker-compose.yml”, “docker-compose.dev.yml”: τα Docker Compose αρχεία παραμετροποίησης ορίζουν και ελέγχουν τα “Docker Container 1”, “Docker Container 2”, “Docker Container 3”, “Docker Container 4”, “Docker Container 5”, “Docker Container 6”. Το “Docker Container 1” εκτελεί την Node.js εφαρμογή διακομιστή “sensor”. Το “Docker Container 2” εκτελεί την Node.js εφαρμογή διακομιστή “RedNode”. Το “Docker Container 3” εκτελεί το instance της Redis βάσης δεδομένων. Το “Docker Container 4” εκτελεί την το instance της MongoDb βάσης δεδομένων. Το “Docker Container 5” εκτελεί την Node.js εφαρμογή API “mongoapi”. Το “Docker Container 6” εκτελεί την Node.js εφαρμογή “result”.

(1.1.1, 1.5.1) Το Docker Compose εγκαθιστά δύο δίκτυα επικοινωνιών μεταξύ των περιεκτών Docker. Το πρώτο δίκτυο «front-tier επιτρέπει στο “Docker Container 1” να επικοινωνεί αμφίδρομα με το “Docker Container 2”, και να ανταλλάσσει δεδομένα μεταξύ τους σχετικά με τις τιμές των αισθητήρων και τις χρονομετρημένες επιδόσεις των διαδικασιών αποθήκευσης και ανάκτησης δεδομένων από τις βάσεις. Επίσης επιτρέπει στο “Docker

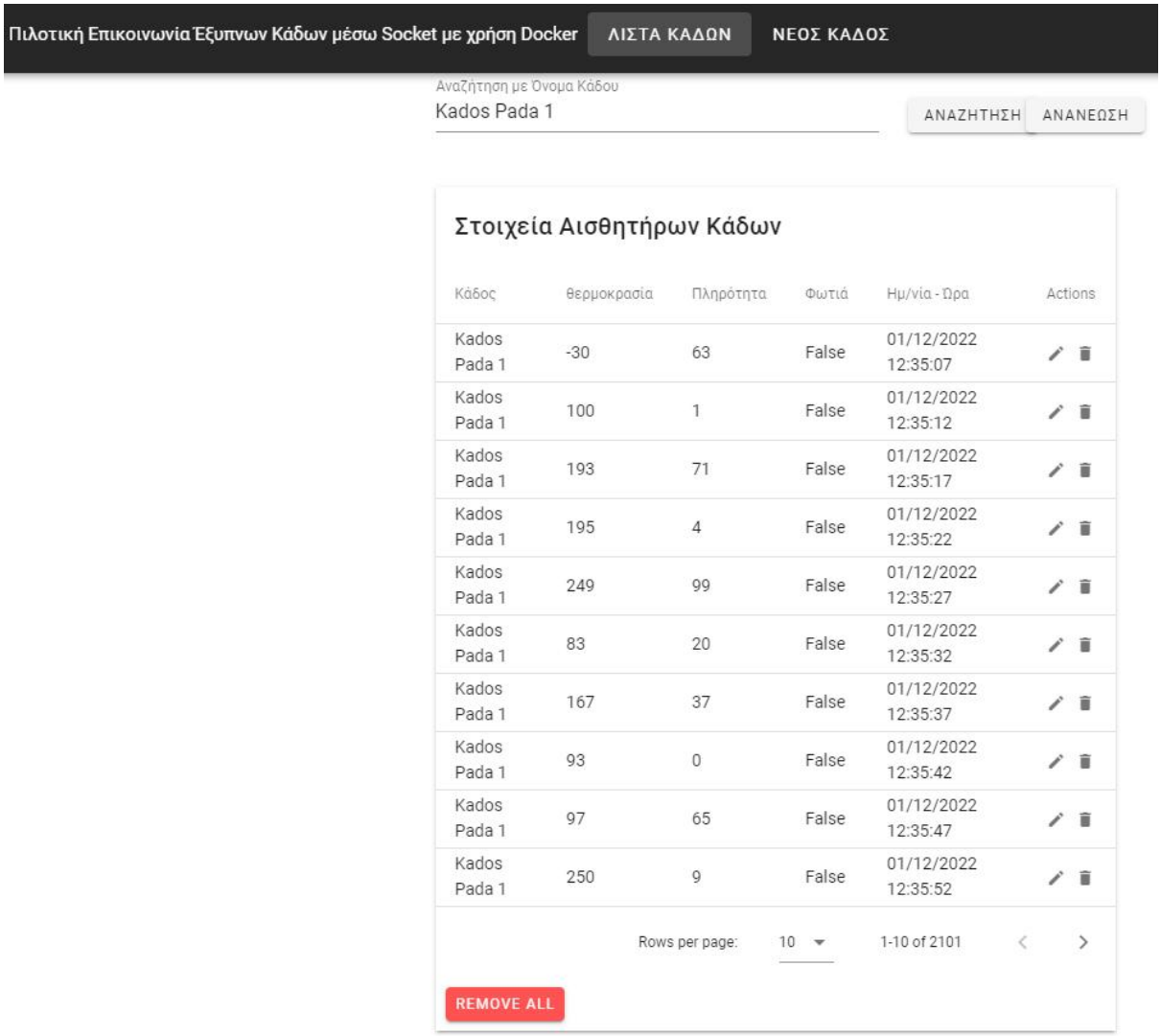
Container 6” να επικοινωνεί με το “Docker Container 5” και χρησιμοποιώντας τις λειτουργίες που του παρέχει το api να εμφανίζει και να διαχειρίζεται τα δεδομένα της MongoDB.

(1.1.2) Το Docker Compose αντιστοιχίζει την τοπική πόρτα 4000 της υποδομής που φιλοξενεί το docker στην πόρτα 5000 του “Docker Container 1”. Η πόρτα 5000 είναι η πόρτα στην οποία ο διακομιστής Node.js “sensor” ακούει και ανταποκρίνεται στις εντολές socket που αποστέλλονται από τον περιηγητή. Η διεύθυνση «http://localhost:4000/» είναι αυτή που μας εμφανίζει την σελίδα demo της εφαρμογής στον περιηγητή ιστοσελίδων.

(1.3) “Docker Container 3”: ο περιέκτης εκτελεί το στιγμιότυπο της Redis βάσης δεδομένων στο οποίο αποθηκεύονται τα στοιχεία των αισθητήρων του κάδου που παράγονται από τον περιηγητή στη διεύθυνση «http://localhost:4000/». Τα στοιχεία αποθηκεύονται από την Node.js διαδικασία διακομιστή «rednode» στο set που φέρει κάθε φορά το όνομα του εκάστοτε κάδου π.χ. “Kados Pada1”. Είναι προσπελάσιμα στη μνήμη του συστήματος με συνέπεια μεγάλη ταχύτητα ανάγνωσης και εγγραφής αλλά όταν συμβεί επανεκκίνηση της διαδικασίας σβήνουν τα παλαιά και αποθηκεύονται προσωρινά μόνο τα νέα παραγώμενα.

(1.4) “Docker Container 4”: ο περιέκτης εκτελεί το στιγμιότυπο της MongoDB βάσης δεδομένων στο οποίο αποθηκεύονται τα στοιχεία των αισθητήρων του κάδου που παράγονται από τον περιηγητή στη διεύθυνση «http://localhost:4000/». Τα στοιχεία αποθηκεύονται από την Node.js διαδικασία διακομιστή «rednode» στο “set” (σύνολο) που φέρει κάθε φορά το όνομα του εκάστοτε κάδου π.χ. “Kados Pada1”. Η αποθήκευση γίνεται σε αρχείο σε αποθηκευτικό μέσο του συστήματος με συνέπεια μικρότερη ταχύτητα ανάγνωσης και εγγραφής από το Redis αλλά διαγραφή γίνεται μόνο όταν θελήσει ο διαχειριστής του συστήματος.

(1.5) “Docker Container 5”: ο περιέκτης εκτελεί το πρόγραμμα διακομιστή σε περιβάλλον Node.js με όνομα «mongoapi» που έχει τον ρόλο της διεπαφής προγραμματισμού εφαρμογών που ορίζει και διατυπώνει τις λειτουργίες CRUD (Create, Read, Update, Delete) που παρέχει στο πρόγραμμα διακομιστή με όνομα «result» (Docker Container 6) για την επικοινωνία του με την βάση MongoDB (Docker Container 4).



Σχήμα 5.3: Ιστοσελίδα παρουσίασης στοιχείων αισθητήρων των κάδων της εφαρμογής πιλοτικής επικοινωνίας έξυπνων κάδων

(1.6) “Docker Container 6”: ο περιέκτης εκτελεί το πρόγραμμα διακομιστή σε περιβάλλον Node.js με όνομα «result» που επικοινωνεί με τον browser πάνω δεξιά. Εκεί μας απεικονίζει σε δομή δεδομένων τύπου grid τα στοιχεία των αισθητήρων των Κάδων «Θερμοκρασία», «Πληρότητα», «Φωτιά» και «Ημ/νία-Ώρα». Υπάρχει η δυνατότητα να γίνει μεταβολή των στοιχείων ή διαγραφή. Η εμφάνιση γίνεται σε ομάδες των δέκα εγγραφών ανά σελίδα με δυνατότητα αλλαγής σε λιγότερες ή περισσότερες ή όλες. Επίσης υπάρχει η δυνατότητα του φιλτραρίσματος των εγγραφών με φίλτρο το όνομα του κάδου.

(1.6.1) Το Docker Compose αντιστοιχίζει την τοπική πόρτα 8001 της υποδομής που φιλοξενεί το docker στην πόρτα 8001 του “Docker Container 6”. Η διεύθυνση

«http://localhost:8001/wastebins» είναι αυτή στην οποία εμφανίζεται η σελίδα με τις τιμές των αισθητήρων ανά κάδο στον περιηγητή ιστοσελίδων.

(1.11) “Dockerfile” — ορίζει και ελέγχει την διαδικασία του διακομιστή Node.js που περιέχεται στο “Docker Container 6”.

5.2 Ανάλυση αρχείων Docker Compose

5.2.1 Docker-compose.yml

Η έκδοση του docker-compose που χρησιμοποιείται είναι η έκδοση 3 όπως ορίζει το κλειδί version. Το κλειδί services ορίζει ποια θα είναι τα services της εφαρμογής που θα αποτελούν ξεχωριστούς περιέκτες (containers). Το service “sensor” θα χτιστεί στον φάκελο “sensor” του περιέκτη, θα εκτελείται στην πόρτα 5000, είναι εξαρτώμενο από το service “rednode” και ανήκει στα δίκτυα “front-tier” και “back-tier”. Το service “ rednode ” θα χτιστεί στον φάκελο

```yaml
version: "3"
services:
  sensor:
    build: ./sensor
    environment:
      - PORT=5000
    depends_on:
      - rednode
    networks:
      - front-tier
      - back-tier

  rednode:
    build: ./rednode
    environment:
      - PORT=5001
    depends_on:
      - redis
      - mongo
    networks:
      - front-tier
      - back-tier

  mongoapi:
    build: ./mongoapi
    environment:
      - PORT=8080
    depends_on:
      - mongo
    networks:
      - front-tier
      - back-tier

  result:
    build: ./result
    environment:
      - PORT=8001
    networks:
      - front-tier
```

Σχήμα 5.4: Εντολές αρχείου docker-compose.yml για τα microservices sensor, rednode, mongoapi και result

“ rednode ” του περιέκτη, θα εκτελείται στην πόρτα 5001, είναι εξαρτώμενο από τα services “redis” και “mongo” και ανήκει στα δίκτυα “front-tier” και “back-tier”. Το service “ mongoapi ” θα χτιστεί στον φάκελο “mongoapi” του περιέκτη, θα εκτελείται στην πόρτα 8080, είναι εξαρτώμενο από το service “mongo” και ανήκει στα δίκτυα “front-tier” και “back-tier”. Το service “ result ” θα χτιστεί στον φάκελο “result” του περιέκτη, θα εκτελείται στην πόρτα 8001, είναι εξαρτώμενο από το service “mongo” και ανήκει στο δίκτυο “front-tier”. (Σχήμα 5.4).

Το service mongo χρησιμοποιεί το επίσημο image “mongo” από το Docker Hub της mongoDB, ορίζει τιμές για τις μεταβλητές “χρήστη” και “κωδικός” του περιβάλλοντος της mongoDB, ορίζει τον χώρο αποθήκευσης με όνομα“mongo-db” στον φάκελο “data/db” του host, και ανήκει στο δίκτυο “back-tier”. Το service “redis” χρησιμοποιεί το επίσημο image “redis” από το Docker Hub της redis και ανήκει και αυτό στο δίκτυο “back-tier”.Με την εντολή volumes δηλώνεται ότι τον αποθηκευτικό χώρο “mongo-db” θα μπορούν να τον χρησιμοποιήσουν όλοι οι περιέκτες (containers). Με την εντολή networks δηλώνεται ότι τα εικονικά δίκτυα που μπορούν να χρησιμοποιήσουν οι περιέκτες (containers) είναι τα “ front-tier ” και “ back-tier ”.(Σχήμα 5.5).

docker-compose.yml

```
41    mongo:
42      image: mongo
43      environment:
44        - MONGO_INITDB_ROOT_USERNAME=sanjeev
45        - MONGO_INITDB_ROOT_PASSWORD=mypassword
46      volumes:
47        - mongo-db:/data/db
48      networks:
49        - back-tier
50
51    redis:
52      image: redis
53      networks:
54        - back-tier
55
56  volumes:
57    mongo-db:
58
59  networks:
60    front-tier:
61    back-tier:
```

Σχήμα 5.5: Εντολές αρχείου docker-compose.yml για τα microservices mongo και redis και τον ορισμό

5.2.2 Docker-compose.dev.yml

Το docker-compose.dev.yml συμπληρώνει το αρχείο docker-compose.yml ορίζοντας παραμέτρους που εκτελούνται στο περιβάλλον ανάπτυξης. Κάθε υπηρεσία δηλώνεται ως διαφορετικό κλειδί πρώτου επιπέδου κάτω από το γενικό κλειδί services.

Η πρώτη υπηρεσία είναι η sensor (Σχήμα 5.6). Το κλειδί context ορίζει που βρίσκεται το αρχείο Dockerfile, είτε είναι η διαδρομή ενός φακέλου είτε το url για ένα αποθετήριο git. Στη περίπτωση μας είναι η σχετική διαδρομή αναφορικά με την θέση που βρίσκεται το Compose αρχείο που αναλύουμε. Το Dockerfile για την υπηρεσία sensor βρίσκεται στον φάκελο sensor. Το κλειδί args ορίζει την τιμή “developent” της παραμέτρου NODE_ENV που είναι μία μεταβλητή περιβάλλοντος που χρησιμοποιείται μόνο κατά το χτίσιμο του περιέκτη και σηματοδοτεί στο Dockerfile ότι η λειτουργία είναι δοκιμαστικού περιβάλλοντος. Το κλειδί ports ορίζει ότι οι πόρτες 3000 και 5000 του περιέκτη θα αντιστοιχηθούν στις πόρτες 3000 και 4000 του οικοδεσπότη (host). Το κλειδί volumes δημιουργεί ένα docker volume και αντιστοιχίζει τον φάκελο sensor του οικοδεσπότη (host) στον φάκελο app του περιέκτη (container). Επίσης δημιουργεί ένα δεύτερο docker volume για τον φάκελο /app/sensor/node_modules του περιέκτη (container). Το κλειδί environment δηλώνει μεταβλητές συστήματος (environment variables) που αφορούν το περιβάλλον λειτουργίας καθώς και τα στοιχεία εισόδου στην βάση δεδομένων της mongoDB που χρησιμοποιεί η εφαρμογή. Το κλειδί command εκτελεί την εντολή “npm run dev” η οποία εκτελεί το script dev” που βρίσκεται μέσα στο αρχείο package.json. Το script “dev” εκτελεί την εντολή “ nodemon -L index.js” που εκκινεί την εφαρμογή του node.js “sensor”.

Η δεύτερη υπηρεσία είναι η rednode (Σχήμα 5.6). Το κλειδί context ορίζει που βρίσκεται το αρχείο Dockerfile, είτε είναι η διαδρομή ενός φακέλου είτε το url για ένα αποθετήριο git. Στη περίπτωση μας είναι η σχετική διαδρομή αναφορικά με την θέση που βρίσκεται το Compose αρχείο που αναλύουμε. Το Dockerfile για την υπηρεσία rednode βρίσκεται στον φάκελο rednode. Το κλειδί args ορίζει την τιμή “developent” της παραμέτρου NODE_ENV που είναι μία μεταβλητή περιβάλλοντος που χρησιμοποιείται μόνο κατά το χτίσιμο του περιέκτη και σηματοδοτεί στο Dockerfile ότι η λειτουργία είναι δοκιμαστικού περιβάλλοντος. Το κλειδί ports ορίζει ότι οι πόρτες 3001 και 5001 του περιέκτη θα αντιστοιχηθούν στις πόρτες 3001 και 4001 του οικοδεσπότη (host). Το κλειδί volumes δημιουργεί ένα docker volume και αντιστοιχίζει τον φάκελο rednode του οικοδεσπότη (host) στον φάκελο app του περιέκτη

(container). Επίσης δημιουργεί ένα δεύτερο docker volume για τον φάκελο /app/rednode/node_modules του περιέκτη (container). Το κλειδί environment δηλώνει μεταβλητές συστήματος (environment variables) που αφορούν το περιβάλλον λειτουργίας καθώς και τα στοιχεία εισόδου στην βάση δεδομένων της mongoDB που χρησιμοποιεί η εφαρμογή τα οποία είναι δηλωμένα σε αυτό το σημείο. Το κλειδί command εκτελεί την εντολή “npm run dev” η οποία εκτελεί το script “dev” που βρίσκεται μέσα στο αρχείο package.json. Το script “dev” εκτελεί την εντολή “ nodemon -L server.js” που εκκινεί την εφαρμογή του node.js “rednode”.

docker-compose.dev.yml

```yaml
version: "3"
services:
  sensor:
    build:
      context: ./sensor
      args:
        NODE_ENV: development
    ports:
      - "3000:3000"
      - "4000:5000"
    volumes:
      - ./sensor:/app
      - /app/sensor/node_modules
    environment:
      - NODE_ENV=development
      - MONGO_USER=sanjeev
      - MONGO_PASSWORD=mypassword
      - SESSION_SECRET=secret
    command: npm run dev
  rednode:
    build:
      context: ./rednode
      args:
        NODE_ENV: development
    ports:
      - "3001:3001"
      - "4001:5001"
    volumes:
      - ./rednode:/app
      - /app/rednode/node_modules
    environment:
      - NODE_ENV=development
      - MONGO_USER=sanjeev
      - MONGO_PASSWORD=mypassword
      - SESSION_SECRET=secret
    command: npm run dev

```

Σχήμα 5.6: Εντολές αρχείου docker-compose.dev.yml για τις υπηρεσίες sensor και rednode.

Η τρίτη υπηρεσία είναι η mongoapi (Σχήμα 5.7). Το κλειδί context ορίζει που βρίσκεται το αρχείο Dockerfile, είτε είναι η διαδρομή ενός φακέλου είτε το url για ένα αποθετήριο git. Στη περίπτωση μας είναι η σχετική διαδρομή αναφορικά με την θέση που βρίσκεται το Compose αρχείο που αναλύουμε. Το Dockerfile για την υπηρεσία mongoapi βρίσκεται στον φάκελο mongoapi. Το κλειδί args ορίζει την τιμή “developent” της παραμέτρου NODE_ENV που είναι μία μεταβλητή περιβάλλοντος που χρησιμοποιείται μόνο κατά το χτίσιμο του περιέκτη και σηματοδοτεί στο Dockerfile ότι η λειτουργία είναι δοκιμαστικού περιβάλλοντος. Το κλειδί ports ορίζει ότι η πόρτα 8080 του περιέκτη (container) θα αντιστοιχηθεί στην πόρτα 8080 του

οικοδεσπότη (host). Το κλειδί volumes δημιουργεί ένα docker volume και αντιστοιχίζει τον φάκελο mongoapi του οικοδεσπότη (host) στον φάκελο app του περιέκτη (container). Επίσης δημιουργεί ένα δεύτερο docker volume για τον φάκελο /app/mongoapi/node_modules του περιέκτη (container). Το κλειδί environment δηλώνει μεταβλητές συστήματος (environment variables) που αφορούν το περιβάλλον λειτουργίας καθώς και τα στοιχεία εισόδου στην βάση δεδομένων της mongoDB που χρησιμοποιεί η εφαρμογή τα οποία είναι δηλωμένα σε αυτό το σημείο. Το κλειδί command εκτελεί την εντολή “npm run dev” η οποία εκτελεί το script “dev” που βρίσκεται μέσα στο αρχείο package.json. Το script “dev” εκτελεί την εντολή “ nodemon -L server.js” που εκκινεί την εφαρμογή του node.js “mongoapi”.

Η τέταρτη υπηρεσία είναι η result (Σχήμα 5.7). Το κλειδί context ορίζει που βρίσκεται το αρχείο Dockerfile, είτε είναι η διαδρομή ενός φακέλου είτε το url για ένα αποθετήριο git. Στη περίπτωση μας είναι η σχετική διαδρομή αναφορικά με την θέση που βρίσκεται το Compose αρχείο που αναλύουμε. Το Dockerfile για την υπηρεσία result βρίσκεται στον φάκελο result. Το κλειδί args ορίζει την τιμή “developent” της παραμέτρου NODE_ENV που είναι μία μεταβλητή περιβάλλοντος που χρησιμοποιείται μόνο κατά το χτίσιμο του περιέκτη και σηματοδοτεί στο Dockerfile ότι η λειτουργία είναι δοκιμαστικού περιβάλλοντος. Το κλειδί ports ορίζει ότι η πόρτα 8001 του περιέκτη (container) θα αντιστοιχηθεί στην πόρτα 8001 του οικοδεσπότη (host). Το κλειδί volumes δημιουργεί ένα docker volume και αντιστοιχίζει τον φάκελο result του οικοδεσπότη (host) στον φάκελο app του περιέκτη (container). Επίσης δημιουργεί ένα δεύτερο docker volume για τον φάκελο /app/result/node_modules του περιέκτη (container). Το κλειδί environment δηλώνει μεταβλητές που αφορούν το περιβάλλον λειτουργίας καθώς και τα στοιχεία εισόδου στην βάση δεδομένων της mongoDB που χρησιμοποιεί η εφαρμογή τα οποία είναι δηλωμένα σε αυτό το σημείο. Το κλειδί command εκτελεί την εντολή “npm run dev” η οποία εκτελεί το script “serve” που βρίσκεται μέσα στο αρχείο package.json. Το script “serve” εκτελεί την εντολή “vue-cli-service serve” που εκκινεί την εφαρμογή του node.js “result”.

Η πέμπτη υπηρεσία είναι η mogo (Σχήμα 5.7). Το κλειδί environment δηλώνει μεταβλητές που αφορούν τα στοιχεία εισόδου στην βάση δεδομένων της mongoDB που παίρνουν τιμή με δήλωση που γίνεται σε αυτό το σημείο.

docker-compose.dev.yml

```yaml
mongoapi:
  build:
    context: ./mongoapi
    args:
      NODE_ENV: development
  ports:
    - "8080:8080"
  volumes:
    - ./mongoapi:/app
    - /app/mongoapi/node_modules
  environment:
    - NODE_ENV=development
    - MONGO_USER=sanjeev
    - MONGO_PASSWORD=mypassword
    - SESSION_SECRET=secret
  command: npm run dev

result:
  build:
    context: ./result
    args:
      NODE_ENV: development
  ports:
    - "8001:8001"
  volumes:
    - ./result:/app
    - /app/result/node_modules
  environment:
    - NODE_ENV=development
    - MONGO_USER=sanjeev
    - MONGO_PASSWORD=mypassword
    - SESSION_SECRET=secret
  command: npm run serve

mongo:
  environment:
    - MONGO_INITDB_ROOT_USERNAME=sanjeev
    - MONGO_INITDB_ROOT_PASSWORD=mypassword
```

Σχήμα 5.7: Εντολές αρχείου docker-compose.dev.yml για τις υπηρεσίες mongoapi, result και mongo

5.2.3 Docker-compose.prod.yml

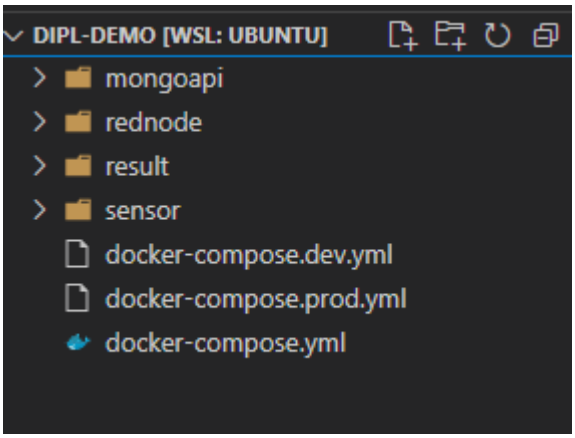
Το docker-compose.prod.yml συμπληρώνει το αρχείο docker-compose.yml ορίζοντας παραμέτρους που εκτελούνται στο περιβάλλον λειτουργίας (Σχήμα 5.8). Οι βασικές διαφορές που έχει με το περιβάλλον ανάπτυξης είναι δύο. Η πρώτη είναι η χρησιμοποίηση του reverse proxy Nginx στην πόρτα 80 σε περιέκτη και οικοδεσπότη και η δεύτερη η χρήση μεταβλητών συστήματος (environment variables) στα στοιχεία του διαχειριστή της βάσης δεδομένων mongoDB ούτως ώστε να μην δημοσιευτούν σε κοινό αποθετήριο τύπου git και γίνουν ευρέως γνωστά.

```yaml
# docker-compose.prod.yml
version: "3"
services:
  nginx:
    ports:
      - "80:80"
  demo-app:
    build:
      context: .
      args:
        NODE_ENV: production
    environment:
      - NODE_ENV=development
      - MONGO_USER=${MONGO_USER}
      - MONGO_PASSWORD=${MONGO_PASSWORD}
      - SESSION_SECRET=${SESSION_SECRET}
    command: node index.js
  mongo:
    environment:
      - MONGO_INITDB_ROOT_USERNAME=${MONGO_INITDB_ROOT_USERNAME}
      - MONGO_INITDB_ROOT_PASSWORD=${MONGO_INITDB_ROOT_PASSWORD}
```

Σχήμα 5.8: Εντολές αρχείου docker-compose.prod.yml

5.3 Δομή κώδικα microservices

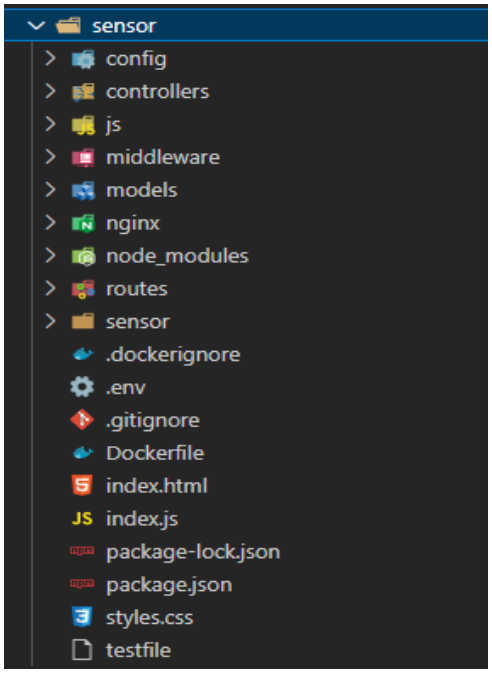
Το πρώτο επίπεδο της εφαρμογής “dipl-demo” περιλαμβάνει τους φακέλους των υπηρεσιών και τα αρχεία του docker compose που οργανώνουν τους περιέκτες (containers) και δημιουργούν όλη την αρχιτεκτονική των επιμέρους συστατικών της εφαρμογής (Σχήμα 5.9).



Σχήμα 5.9: Δομή αρχείων – φακέλων

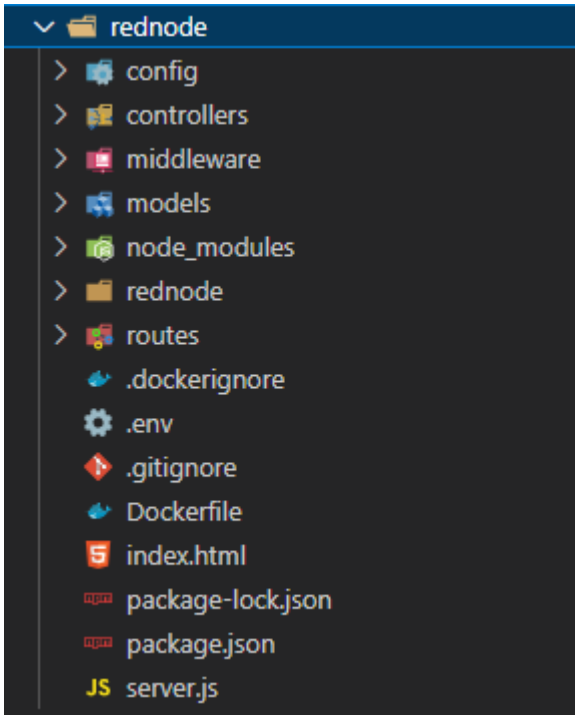
Τα περιεχόμενα του φακέλου “sensor”, που περιέχει των πηγαίο κώδικα για την εφαρμογή Node.js “sensor” εμφανίζονται στο σχήμα (Σχήμα 5.10). Το αρχείο “Dockerfile” δημιουργεί τον περιέκτη (container) της Node.js εφαρμογής. Το αρχείο “.dockerignore” περιέχει λίστα με

τα αρχεία που δεν θα αντιγραφούν μέσα στον περιέκτη. Το αρχείο “.gitignore” περιέχει λίστα με τα αρχεία που δεν θα μεταφερθούν στο αποθετήριο κώδικα. Το αρχείο “package.json” περιέχει την παραμετροποίηση του node.js και το script εκκίνησης "dev": "nodemon -L index.js". Το αρχείο “index.js” περιέχει τον πηγαίο κώδικα που θα εκτελεστεί πρώτος.



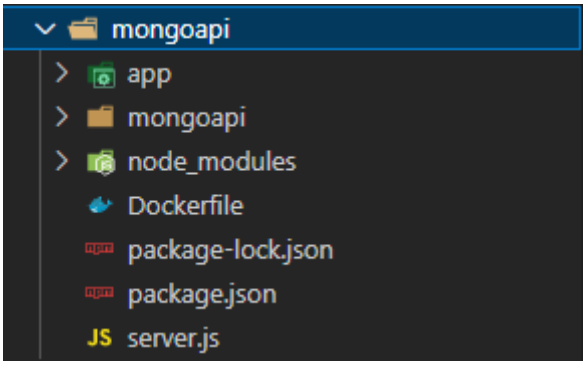
Σχήμα 5.10: Δομή φακέλων - αρχείων της υπηρεσίας sensor

Τα περιεχόμενα του φακέλου “rednode”, που περιέχει των πηγαίο κώδικα για την εφαρμογή Node.js “rednode” εμφανίζονται στο σχήμα (Σχήμα 5.11). Το αρχείο “Dockerfile” δημιουργεί τον περιέκτη (container) της Node.js εφαρμογής. Το αρχείο “.dockerignore” περιέχει λίστα με τα αρχεία που δεν θα αντιγραφούν μέσα στον περιέκτη. Το αρχείο “.gitignore” περιέχει λίστα με τα αρχεία που δεν θα μεταφερθούν στο αποθετήριο κώδικα. Το αρχείο “package.json” περιέχει την παραμετροποίηση του node.js και το script εκκίνησης "dev": "nodemon -L server.js". Το αρχείο “server.js” περιέχει τον πηγαίο κώδικα που θα εκτελεστεί πρώτος.



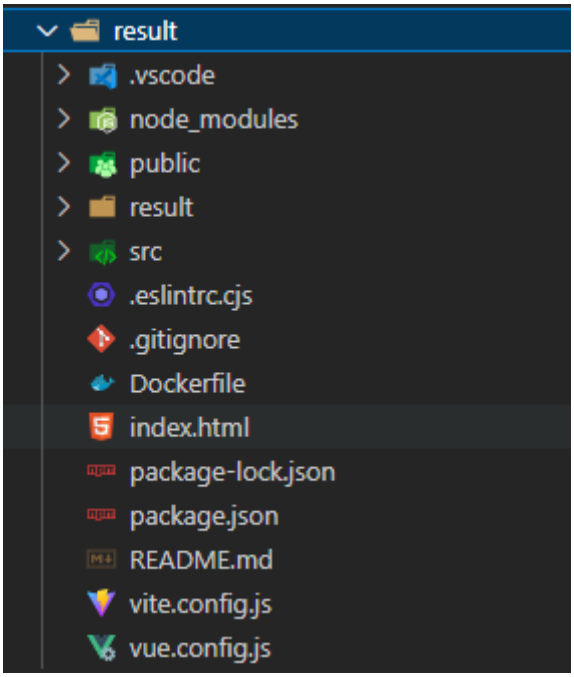
Σχήμα 5.11: Δομή φακέλων - αρχείων της υπηρεσίας rednode

Τα περιεχόμενα του φακέλου “mongoapi”, που περιέχει των πηγαίο κώδικα για την εφαρμογή Node.js “mongoapi” εμφανίζονται στο σχήμα (Σχήμα 5.12). Το αρχείο “Dockerfile” δημιουργεί τον περιέκτη (container) της Node.js εφαρμογής. Το αρχείο “package.json” περιέχει την παραμετροποίηση του node.js και το script εκκίνησης "start": "node server.js",. Το αρχείο “server.js” περιέχει τον πηγαίο κώδικα που θα εκτελεστεί πρώτος.



Σχήμα 5.12: Δομή φακέλων - αρχείων της υπηρεσίας mongoapi

Τα περιεχόμενα του φακέλου “result”, που περιέχει των πηγαίο κώδικα για την εφαρμογή Node.js “result” εμφανίζονται στο σχήμα (Σχήμα 5.13). Το αρχείο “Dockerfile” δημιουργεί τον περιέκτη (container) της Node.js εφαρμογής. Το αρχείο “.dockerignore” περιέχει λίστα με τα αρχεία που δεν θα αντιγραφούν μέσα στον περιέκτη. Το αρχείο “.gitignore” περιέχει λίστα με τα αρχεία που δεν θα μεταφερθούν στο αποθετήριο κώδικα. Το αρχείο “package.json” περιέχει την παραμετροποίηση του node.js και το script εκκίνησης "serve": "vue-cli-service serve". Το αρχείο “vue.config.js” περιέχει τον πηγαίο κώδικα που θα εκκινήσει ένα τοπικό εξυπηρετητή ιστοσελίδων (web server).



Σχήμα 5.13: Δομή φακέλων - αρχείων της υπηρεσίας result

5.4 Ανάλυση κώδικα “sensor” microservice

5.4.1 Εισαγωγή

Η υπηρεσία sensor αποτελείται από δύο μέρη, μία ιστοσελίδα και έναν εξυπηρετητή ιστού. Η ιστοσελίδα περιέχει κουμπιά ελέγχου για παραγωγή εικονικών τιμών και πεδία απεικόνισης τιμών χρόνου και γεγονότων. Ο εξυπηρετητής ιστού παράγει εικονικές τιμές αισθητήρων φωτιάς ανά τυχαία χρονικά διαστήματα, τιμές αισθητήρων πυρκαγιάς και πληρότητας ανά τακτά χρονικά διαστήματα, και τις αποστέλλει μέσω διαύλου επικοινωνίας socket στην υπηρεσία rednode. Επίσης λαμβάνει πληροφορίες εντολής πυρόσβεσης και τιμές χρόνου ανάκτησης δεδομένων από τις βάσεις δεδομένων redis και mongoDB μέσω της rednode υπηρεσίας.

5.4.2 Ανάλυση αρχείου Dockerfile

Το αρχείο “Dockerfile” (Σχήμα 5.14) δημιουργεί τον περιέκτη (container) της Node.js εφαρμογής λαμβάνοντας από το Docker Hub την “node:15” εικόνα, εγκαθιστώντας το Node.js στον περιέκτη, αντιγράφοντας στον περιέκτη τα αρχεία του πηγαίου κώδικα, εγκαθιστώντας το npm, και μετά εκκινεί τον Node.js εξυπηρετητή ιστού (web server) της εφαρμογής.

sensor > Dockerfile > ...

```
1   FROM node:15
2   WORKDIR /app
3   COPY package.json .
4   #RUN npm install --only=production
5   ARG NODE_ENV
6   RUN if [ "$NODE_ENV" = "development" ]; \
7           then npm install ; \
8           else npm install --only=production; \
9           fi
10
11  COPY . ./
12  ENV PORT 5000
13  EXPOSE $PORT
14  CMD ["node", "index.js"]
```

Σχήμα 5.14: Αρχείο Dockerfile της υπηρεσίας sensor

5.4.3 Ανάλυση αρχείου client.js

Το αρχείο client.js περιέχει τον κώδικα που διαχειρίζεται την επικοινωνία μεταξύ της ιστοσελίδας και του κεντρικού κόμβου rednode.Στην αρχή ορίζεται το url του rednode με ειδικές παραμέτρους cors (cross-origin resource sharing) που επιτρέπει την ασφαλή ανταλλαγή δεδομένων μέσω socket μεταξύ κόμβων που βρίσκονται σε διαφορετικά domain. Εκεί δηλώνεται η IP διεύθυνση και η πόρτα του άλλου κόμβου με τον οποίο θα γίνει η επικοινωνία. Μετά ανοίγει το κανάλι socket μεταξύ sensor και rednode. (σχήμα 5.15).

```
sensor > js > JS client.js > randomNum
1   // Ορισμός της HTTP διεύθυνσης του red node με το οποίο θα γίνει η Socket επικοινωνία
2   const url = `http://localhost:4001`;
3   //const url = `http://dipl-demo_rednode_1:4001`;
4   const corsSettings = `{
5     withCredentials: true,
6     extraHeaders: {
7       "my-custom-header": "socket.io"
8     }
9   }`;
10
11  // Άνοιγμα του καναλιού Socket με το red node
12  const socket = io(url, {
13    withCredentials: true,
14    extraHeaders: {
15      "Access-Control-Allow-Origin": "socket.io"
16    }
17  });
```

Σχήμα 5.15: Client.js: Ορισμός της HTTP διεύθυνσης του red node με το οποίο θα γίνει η Socket επικοινωνία και άνοιγμα του καναλιού Socket.

Από τη στιγμή που έχει ανοιχτεί το κανάλι, έχει εγκαθιδρυθεί αμφίδρομη επικοινωνία μεταξύ sensor και rednode. Για να γίνει εφικτή η διαχείριση των μηνυμάτων που αποστέλλονται έχουν αναπτυχθεί event listeners που ανάλογα τον τίτλο του μηνύματος που θα λάβουν εκτελούν τον αντίστοιχο κώδικα που περικλείουν.(Σχήμα 5.16) Το μήνυμα “WastebinTimeFromRedis” εμπεριέχει την παράμετρο “redisTime” που εμφανίζεται στην ιστοσελίδα του sensor και είναι ο χρόνος ανάκτησης δεδομένων από το Redis που διαβιβάστηκε από το Red Node. Αντίστοιχα το μήνυμα “WastebinTimeFromMongoDB” εμπεριέχει την παράμετρο “mongoDBTime ” που εμφανίζεται στην ιστοσελίδα του sensor και είναι ο χρόνος ανάκτησης δεδομένων από την MongoDB που διαβιβάστηκε από το Red Node

sensor > js > JS client.js > ...

```
// ********** Socket Event Listeners ΑΡΧΗ ***************
// Οι παρακάτω listeners παρακολουθούν τα μηνύματα που έρχονται μέσω socket
// και ανάλογα τον τίτλο υλοποιούν την αντίστοιχη λειτουργικότητα

// Όταν συνδέεται ένας άλλος κάδος
socket.on("new-login", (name) => {
  console.log("client: Νέα Σύνδεση Κάδου ", name);
});

socket.on("joined-room", () => {
  console.log("sensor successfully joined");
});

// Χρόνος ανάκτησης δεδομένων από το Redis που μας έρχεται από το Red Node
socket.on("WastebinTimeFromRedis", (redisTime) => {
  log1 = "client: Χρόνος Ανάκτησης Δεδομένων από το Redis: " + redisTime + " millisecond";
  console.log(log1);
  $fromRedisLabel.style.color = "#FF0000"
  $fromRedisLabel.innerHTML = "Χρόνος Ανάκτησης Δεδομένων από το Redis:  " + redisTime + " millisecond";
});

// Χρόνος ανάκτησης δεδομένων από την MongoDB που μας έρχεται από το Red Node
socket.on("WastebinTimeFromMongoDB", (mongoDBTime) => {
  log1 = "client: Χρόνος Ανάκτησης Δεδομένων από την MongoDB: " + mongoDBTime + " millisecond";
  console.log(log1);
  $fromMongoDBLabel.style.color = "#FF0000"
  $fromMongoDBLabel.innerHTML = "Χρόνος Ανάκτησης Δεδομένων από την MongoDB:  " + mongoDBTime + " millisecond";
});
```

Σχήμα 5.16: Socket Event Listeners

Ο listener “FireAlertAction” ενεργοποιείται όταν το RedNode αποστείλει το αντίστοιχο μήνυμα και εξομοιώνει την λειτουργία ενός actuator (ενεργοποιητή) που θα έδινε εντολή σε ένα κύκλωμα πυρόσβεσης της πυρκαγιάς μέσα στον κάδο. Ο listener “FireAlertBroadcast” είναι για μελλοντική χρήση και ενεργοποιείται όταν κάποιο άλλο node – κάδος αποστείλει το αντίστοιχο μήνυμα πυρόσβεσης που εξομοιώνει την λειτουργία ενός actuator (ενεργοποιητή) που θα έδινε εντολή σε ένα κύκλωμα πυρόσβεσης της πυρκαγιάς μέσα στον κάδο. (Σχήμα 5.17).

sensor > js > JS client.js > ...

```

// Ειδοποίηση του actuator για Πυρόσβεση του κάδου που μας έρχεται από το Red Node
socket.on("FireAlertAction", (socketid) => {
  console.log("client: Ενεργοποίηση Μηχανισμού Πυρόσβεσης Κάδου ", socketid);
  $FireLabel.style.color = "#FF0000"
  $FireLabel.innerHTML = "Ενεργοποίηση Μηχανισμού Πυρόσβεσης στον κόμβο " + socketid;
  // $stopWastebinBtn.click();
});

// Ειδοποίηση του actuator για Πυρόσβεση του κάδου που μας έρχεται από άλλους κάδους
socket.on("FireAlertBroadcast", (name) => {
  console.log("client: Ενεργοποιήθηκε ο Μηχανισμός Πυρόσβεσης στον Κάδο : ", name);
  $FireLabel.style.color = "#008000"
  $FireLabel.innerHTML = "Ενεργοποιήθηκε ο Μηχανισμός Πυρόσβεσης στον Κάδο : " + name;
});
```

Σχήμα 5.17: Socket Event Listeners που αφορούν actuators

Η συνάρτηση “randomNum” υπολογίζει και επιστρέφει έναν τυχαίο ακέραιο αριθμό μεταξύ των δύο ορίων που λαμβάνει ως παραμέτρους. Η συνάρτηση “startTimer” δημιουργεί ένα χρονομετρητή και την τιμή του την προωθεί στην ετικέτα Timer της ιστοσελίδας. Επίσης σταματάει την δημιουργία τιμών όταν πατηθεί το κουμπί “stopWastebinBtn” (Σχήμα 5.18).

```
sensor > js > JS client.js > ...
113  // Δημιουργία τυχαίου ακέραιου μεταξύ δύο ορίων
114  const randomNum = (min, max) => {
115    var num = Math.floor((Math.random() * (max - min + 1) + min));
116    return num;
117  };
118
119  // Δημιουργία χρονομετρητή και Ενημέρωση ετικέτας Timer
120  function startTimer() {
121    let timePassed = 0;
122    var timerInterval = setInterval(() => {
123
124      // The amount of time passed increments by one
125      timePassed = timePassed += 1;
126
127      // The timer label is updated
128      $TimerLabel.innerHTML = timePassed;
129    }, 1000);
130
131    // Διακοπή λειτουργίας timer
132    $stopWastebinBtn.addEventListener("click", () => {
133      clearInterval(timerInterval);
134    });
135  }
136
```

Σχήμα 5.18: Δημιουργία τυχαίου ακεραίου και χρονομετρητή

Ο event listener στη γραμμή 141 ενεργοποιείται με το πάτημα του κουμπιού “loginBtn” και αποστέλλει μέσω socket με το κλειδί “login” το όνομα του κάδου στον κεντρικό κόμβο rednode. Ο event listener στη γραμμή 151 ενεργοποιείται με το πάτημα του κουμπιού “startWastebinBtn” και εκκινεί τον χρονομετρητή, εκτελεί την συνάρτηση “sendWastebinToRedNode” που αποστέλλει κάθε 5 δευτερόλεπτα τιμές πληρότητας και θερμοκρασίας και κάθε 1 δευτερόλεπτο τιμές πυρκαγιάς στο rednode. Ο event listener στη γραμμή 165 ενεργοποιείται με το πάτημα του κουμπιού “ stopWastebinBtn ” διακόπτει τις λειτουργίες του χρονομετρητή και της δημιουργίας των εικονικών τιμών των αισθητήρων (Σχήμα 5.19).

sensor > js > JS client.js > ...

```
139
140   // Σύνδεση κάδου μέσω socket
141   $loginBtn.addEventListener("click", () => {
142     log1 = "Ο κάδος συνδέθηκε με όνομα: " + $wastebinNameInput.value
143     console.log(log1);
144     socket.emit("login", $wastebinNameInput.value);
145     $recipientInput.value = socket.id;
146     $FireLabel.style.color = "#000000"
147     $FireLabel.innerHTML = "Αναμονή ...";
148   });
149
150   // Ενεργοποίηση διαδικασίας δημιουργίας τιμών αισθητήρων και αποστολή τους στο Red Node
151   $startWastebinBtn.addEventListener("click", () => {
152     startTimer();  // Εκκίνηση χρονομετρητή
153
154     // Κάθε 5 δευτερόλεπτα αποστολή τιμών πληρότητας και θερμοκρασίας στο Red Node
155     var si = setInterval(function A() {
156       return sendWastebinToRedNode(false);
157     }, 5000);
158
159     // Κάθε 1 δευτερόλεπτο αποστολή τιμής True or False για πυρκαγιά στο Red Node
160     var siFire = setInterval(function A() {
161       return sendWastebinToRedNode(true);
162     }, 1000);
163
164     // Διακοπή χρονομετρητή και δημιουργίας τιμών αισθητήρων
165     $stopWastebinBtn.addEventListener("click", () => {
166       clearInterval(si);
167       clearInterval(siFire);
168     });
169   });
```

Σχήμα 5.19:Σύνδεση κάδου sensor με κεντρικό κάδο rednode και ενεργοποίηση - απενεργοποίηση διαδικασίας δημιουργίας τιμών αισθητήρων.

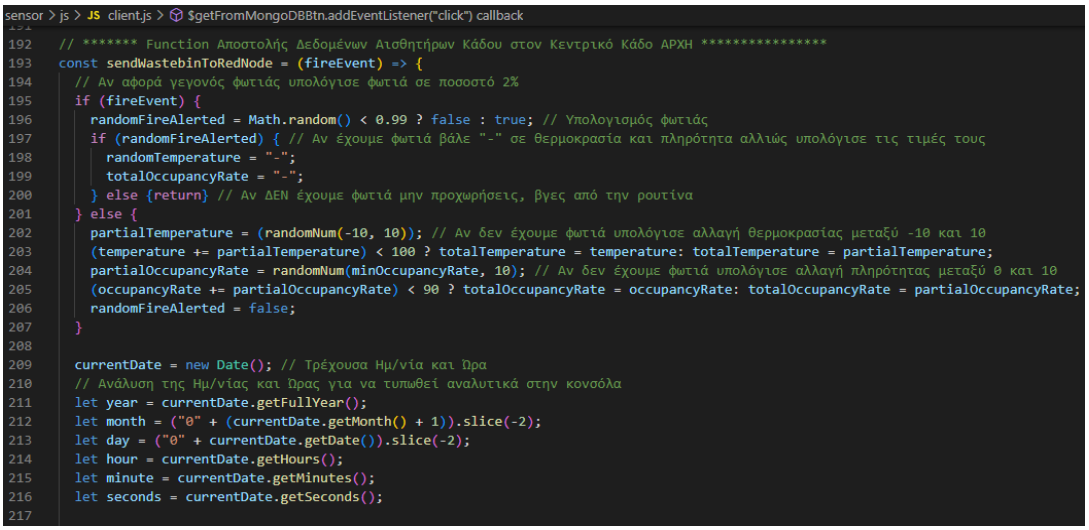
Ο event listener στη γραμμή 172 ενεργοποιείται με το πάτημα του κουμπιού “getFromRedisBtn” και αποστέλλει μέσω socket με το κλειδί “getWastebinFromRedis” το όνομα του κάδου στον κεντρικό κόμβο rednode ζητώντας να του επιστραφεί ο χρόνος ανάκτησης των δεδομένων του κάδου από την βάση “redis”. Επίσης αρχικοποιεί το χρώμα και την τιμή της ετικέτας εμφάνισης του χρόνου. Ο event listener στη γραμμή 182 ενεργοποιείται με το πάτημα του κουμπιού “getFromMongoDBBtn” και αποστέλλει μέσω socket με το κλειδί “getWastebinFromMongoDB” το όνομα του κάδου στον κεντρικό κόμβο rednode ζητώντας να του επιστραφεί ο χρόνος ανάκτησης των δεδομένων του κάδου από την βάση “mongo”. Επίσης αρχικοποιεί το χρώμα και την τιμή της ετικέτας εμφάνισης του χρόνου. (Σχήμα 5.20).

sensor > js > JS client.js > ...

```
171  // Αποστολή αιτήματος ανάκτησης δεδομένων από το Redis στο Red Node
172  $getFromRedisBtn.addEventListener("click", () => {
173    log1 = "Ζητήθηκε Ανάκτηση Δεδομένων από το Redis για τον κάδο: " + $wastebinNameInput.value
174    console.log(log1);
175    $fromRedisLabel.style.color = "#000000"
176    $fromRedisLabel.innerHTML = "Χρόνος ...";
177    socket.emit("getWastebinFromRedis", $wastebinNameInput.value);
178    $recipientInput.value = socket.id;
179  });
180
181  // Αποστολή αιτήματος ανάκτησης δεδομένων από την MongoDB στο Red Node
182  $getFromMongoDBBtn.addEventListener("click", () => {
183    log1 = "Ζητήθηκε Ανάκτηση Δεδομένων από την MongoDB για τον κάδο: " + $wastebinNameInput.value
184    console.log(log1);
185    $fromMongoDBLabel.style.color = "#000000"
186    $fromMongoDBLabel.innerHTML = "Χρόνος ...";
187    socket.emit("getWastebinFromMongoDB", $wastebinNameInput.value);
188    $recipientInput.value = socket.id;
189  });
190  // *************** Event Listeners ΤΕΛΟΣ ****************
```

Σχήμα 5.20: Αποστολή αιτήματος ανάκτησης δεδομένων από το Redis και από την MongoDB προς το Rednode

Η συνάρτηση “sendWastebinToRedNode” δέχεται ως παράμετρο το είδος του γεγονότος, αν αφορά φωτιά ή κάτι άλλο. Σε περίπτωση φωτιάς δημιουργεί τυχαίο γεγονός πυρκαγιάς σε ποσοστό 2% και ταυτόχρονα μηδενίζει τις άλλες τιμές θερμοκρασίας και πληρότητας. Σε περίπτωση που δεν καλείται για υπολογισμό φωτιάς, υπολογίζει τυχαία θερμοκρασία μεταξύ -10 και +10 βαθμών Κελσίου και την προσθέτει στην προηγούμενη τιμή θερμοκρασίας και υπολογίζει τιμή πληρότητας μεταξύ 0 % και 10 % και την προσθέτει στην προηγούμενη τιμή πληρότητας. Η συνάρτηση λαμβάνει την τρέχουσα ημερομηνία και ώρα και τις μετατρέπει σε χρόνια, μήνες, ημέρες, ώρες, λεπτά και δευτερόλεπτα για να εμφανιστούν και να τυπωθούν με ευανάγνωστο τρόπο στην κονσόλα και την ιστοσελίδα (Σχήμα 5.21).



```
sensor > js > JS client.js > $getFromMongoDBBtn.addEventListener("click") callback
// ******* Function Αποστολής Δεδομένων Αισθητήρων Κάδου στον Κεντρικό Κάδο ΑΡΧΗ ***************
const sendWastebinToRedNode = (fireEvent) => {
  // Αν αφορά γεγονός φωτιάς υπολόγισε φωτιά σε ποσοστό 2%
  if (fireEvent) {
    randomFireAlerted = Math.random() < 0.99 ? false : true; // Υπολογισμός φωτιάς
    if (randomFireAlerted) { // Αν έχουμε φωτιά βάλε "-" σε θερμοκρασία και πληρότητα αλλιώς υπολόγισε τις τιμές τους
      randomTemperature = "-";
      totalOccupancyRate = "-";
    } else {return} // Αν ΔΕΝ έχουμε φωτιά μην προχωρήσεις, βγες από την ρουτίνα
  } else {
    partialTemperature = (randomNum(-10, 10)); // Αν δεν έχουμε φωτιά υπολόγισε αλλαγή θερμοκρασίας μεταξύ -10 και 10
    (temperature += partialTemperature) < 100 ? totalTemperature = temperature: totalTemperature = partialTemperature;
    partialOccupancyRate = randomNum(minOccupancyRate, 10); // Αν δεν έχουμε φωτιά υπολόγισε αλλαγή πληρότητας μεταξύ 0 και 10
    (occupancyRate += partialOccupancyRate) < 90 ? totalOccupancyRate = occupancyRate: totalOccupancyRate = partialOccupancyRate;
    randomFireAlerted = false;
  }

  currentDate = new Date(); // Τρέχουσα Ημ/νία και Ώρα
  // Ανάλυση της Ημ/νίας και Ώρας για να τυπωθεί αναλυτικά στην κονσόλα
  let year = currentDate.getFullYear();
  let month = ("0" + (currentDate.getMonth() + 1)).slice(-2);
  let day = ("0" + currentDate.getDate()).slice(-2);
  let hour = currentDate.getHours();
  let minute = currentDate.getMinutes();
  let seconds = currentDate.getSeconds();
```

Σχήμα 5.21:Συνάρτηση Δημιουργίας και Αποστολής Δεδομένων Φωτιάς και Θερμοκρασίας Κάδου στον Κεντρικό Κάδο rednode

Όταν το ποσοστό πληρότητας φτάσει ή ξεπεράσει το 90 % ξεκινάει πάλι από την αρχή τους υπολογισμούς και τυπώνει ότι ο κάδος είναι πλήρης. Τυπώνει τις τιμές των αισθητήρων του κάδου και της ημερομηνίας δημιουργίας και τις αποστέλλει μέσω socket με το κλειδί “sendWastebin-to” στον κεντρικό κόμβο rednode (Σχήμα 5.22).

```
sensor > js > JS client.js > sendWastebinToRedNode
    if (occupancyRate >= 90) { // Αν η πληρότητα είναι >= 90% τύπωσε οτι ο Κάδος είναι ΠΛΗΡΗΣ
      occupancyRate = partialOccupancyRate;
      log1 = "Ενημέρωση του red node από τον κάδο " +
             $wastebinNameInput.value +
             " μέσω του socket " +
             $recipientInput.value +
             " οτι ο Κάδος είναι ΠΛΗΡΗΣ " +
             " ΗΜ/ΝΙΑ " +
             year + "-" + month + "-" + day + " " + hour + ":" + minute + ":" + seconds;
      console.log(log1);
    }
    // Εκτύπωση των τιμών του Κάδου και της Ημ/νίας
    log1 = "Κάδος: " +
           $wastebinNameInput.value +
           ",   Socket: " + $recipientInput.value +
           ",   Θερμοκρασία: " + totalTemperature + " βαθμούς Κελσίου" +
           ",   Πληρότητα: " + totalOccupancyRate + " % " +
           ",   Συναγερμός Φωτιάς: " + randomFireAlerted +
           ",   Ημ/νία: " + year + "-" + month + "-" + day + " " + hour + ":" + minute + ":" + seconds;
    console.log(log1);
    // Αποστολή τιμών μέσω socket στο Red Node
    socket.emit("sendWastebin-to", {
      wastebinName: $wastebinNameInput.value,
      wastebinIp: $recipientInput.value,
      temperature: totalTemperature,
      occupancyRate: totalOccupancyRate,
      fireAlerted: randomFireAlerted,
      createdAt: currentDate,
    });
  }
  // ******* Function Αποστολής Δεδομένων Κάδου στον Κεντρικό Κάδο ΤΕΛΟΣ ***************
```

Σχήμα 5.22:Συνάρτηση Δημιουργίας και Αποστολής Δεδομένων Πληρότητας Κάδου στον Κεντρικό Κάδο rednode

5.5 Ανάλυση κώδικα “rednode” microservice

5.5.1 Εισαγωγή

Η υπηρεσία rednode αποτελείται από έναν εξυπηρετητή ιστού που έχει τον ρόλο του κεντρικού κόμβου ανταλλαγής και αποθήκευσης πληροφοριών και λήψης αποφάσεων. Η ανταλλαγή πληροφοριών γίνεται μέσω διαύλου επικοινωνίας socket για άμεση ανταπόκριση και η αποθήκευση των δεδομένων χρησιμοποιεί την βάση δεδομένων μνήμης redis και την non sequential βάση δεδομένων mongoDB.

5.5.2 Ανάλυση αρχείου Dockerfile

Το αρχείο “Dockerfile” (Σχήμα 5.23) δημιουργεί τον περιέκτη (container) της Node.js εφαρμογής λαμβάνοντας από το Docker Hub την “node:15” εικόνα, εγκαθιστώντας το Node.js στον περιέκτη, αντιγράφοντας στον περιέκτη τα αρχεία του πηγαίου κώδικα, εγκαθιστώντας το npm, και μετά εκκινεί τον Node.js εξυπηρετητή ιστού (web server) της εφαρμογής.

rednode > Dockerfile > ...

```
1   FROM node:15
2   WORKDIR /app
3   COPY package.json .
4   ARG NODE_ENV
5   RUN if [ "$NODE_ENV" = "development" ]; \
6           then npm install; \
7           else npm install --only=production; \
8           fi
9
10  COPY . ./
11  ENV PORT 5001
12  EXPOSE $PORT
13  CMD ["node", "index.js"]
```

Σχήμα 5.23: Αρχείο Dockerfile της υπηρεσίας rednode

5.5.3 Ανάλυση αρχείου Server.js

Το αρχείο server.js περιέχει τον κώδικα που διαχειρίζεται την επικοινωνία μεταξύ της ιστοσελίδας – κάδου, του κεντρικού κόμβου rednode και των βάσεων δεδομένων redis και mongoDB και διαχειρίζεται την λήψη αποφάσεων ανάλογα τις πληροφορίες που δέχεται.

Στην αρχή ορίζονται οι βιβλιοθήκες που θα χρησιμοποιηθούν και δημιουργείται http αντικείμενο διακομιστή και socket αντικείμενο μέσω του οποίου θα πραγματοποιηθούν οι επικοινωνίες. Στον κατασκευαστή του socket δηλώνεται η χρήση της βιβλιοθήκης cors (cross-origin resource sharing) που επιτρέπει την ασφαλή ανταλλαγή δεδομένων μέσω socket μεταξύ κόμβων που βρίσκονται σε διαφορετικά domain. Εκεί δηλώνεται η IP διεύθυνση και η πόρτα του άλλου κόμβου με τον οποίο θα γίνει η επικοινωνία (σχήμα 5.24).

rednode > JS server.js > ...

```
54  // ***********  ΑΡΧΗ ****************
55  // *********** Δημιουργία Http Server με χρήση του app
56  // ***********και socket με χρήση του socket.io
57  const http = require("http");
58  const server = http.createServer(app);
59  const { Server } = require("socket.io");
60  const { promisify } = require("util");
61  const { traceDeprecation } = require("process");
62  const { threadId } = require("worker_threads");
63
64  // Δημιουργία socket object "io" με παραμέτρους σύνδεσης cors
65  const io = new Server(server, {
66      // Δήλωση στο cors για λόγους ασφαλείας,  IP και πόρτας του client
67      cors: {
68          origin: "http://localhost:4000",
69          credentials: true
70        }
71  });
72  // *********** Δημιουργία Http Server με χρήση του app
73  // ***********και socket με χρήση του socket.io
74  // ***********  ΤΕΛΟΣ ****************
75
```

Σχήμα 5.24: Δημιουργία Http Server και socket object "io" με παραμέτρους

Η συνάρτηση “connectWithRetry” χρησιμοποιώντας το url “mongoURL”, που περιέχει τα στοιχεία εισόδου στη βάση, επαναλαμβάνει τις προσπάθειες έως ότου επιτύχει την σύνδεση με την βάση δεδομένων “mongoDB” (Σχήμα 5.25).

```
rednode > JS server.js > ...
76  // Ορισμός URL για την mongoDB
77  const mongoURL = `mongodb://${MONGO_USER}:${MONGO_PASSWORD}@${MONGO_IP}:${MONGO_PORT}/?authSource=admin`
78
79  // Ρουτίνα σύνδεσης με την mongoDB με επανάληψη αν αποτύχει κάθε 5 δευτερόλεπτα
80  const connectWithRetry = () => {
81      mongoose
82          .connect(mongoURL, {
83              useNewUrlParser: true,
84              useUnifiedTopology: true
85          })
86          .then(() => console.log("succesfully connected to MongoDB"))
87          .catch((e) => {
88              console.log("connection to MongoDB failed!!",e);
89              setTimeout(connectWithRetry, 5000);
90          });
91  }
92  // Σύνδεση με την mongoDB
93  connectWithRetry();
```

Σχήμα 5.25:Σύνδεση με την mongoDB

Από τη στιγμή που έχει ανοιχτεί το socket κανάλι, έχει εγκαθιδρυθεί αμφίδρομη επικοινωνία μεταξύ sensor και rednode. Για να γίνει εφικτή η διαχείριση των μηνυμάτων που αποστέλλονται έχουν αναπτυχθεί event listeners που ανάλογα τον τίτλο του μηνύματος που θα λάβουν εκτελούν τον αντίστοιχο κώδικα που περικλείουν.(Σχήμα 5.26) Όταν ληφθεί μήνυμα με κλειδί “login” αποστέλλεται σε όλους μήνυμα με κλειδί “new-login” που περιέχει το όνομα και τον μοναδικό κωδικό της socket διασύνδεσης. Όταν ληφθεί μήνυμα με κλειδί “sendWastebin-to”, σημαίνει ότι έχουν παραληφθεί τα δεδομένα του κάδου και το πρόγραμμα ελέγχει την παράμετρο “ params.fireAlerted” και αν είναι αληθής που σημαίνει ότι υπάρχει συναγερμός πυρκαγιάς στέλνει socket μήνυμα πίσω στον κάδο με κλειδί “ FireAlertAction” που δίνει εντολή πυρόσβεσης. Στη συνέχεια καλούνται οι συναρτήσεις που θα αποθηκεύσουν τα δεδομένα του κάδου (όνομα κάδου, socket id, θερμοκρασία, πληρότητα, πυρκαγιά, ημερομηνία και ώρα μετρήσεων) στις βάσεις δεδομένων redis και mongoDB.

```
rednode > JS server.js > io.on("connection") callback
131
132   // S O C K E T
133   // Run when client connects
134   io.on("connection", (socket) => {
135       console.log("Συνδέθηκε στο Red Node");
136       // Όταν έρθει μήνυμα "login"
137       socket.on("login", async (name) => {
138         console.log("Συνδέθηκε ο κάδος ", name, " με socket id ", socket.id);
139         // γίνεται broadcast σε όλους ως "new-login"
140         socket.broadcast.emit("new-login", name);
141       });
142       // Όταν έρθει μήνυμα με τα δεδομένα του κάδου
143       socket.on("sendWastebin-to", async(params) => {
144           // έλεγχος αν έχει ενεργοποιηθείτο fireAlert
145           //  και αποστολή στο sensor μηνύματος για FireAlert Action
146           if  (params.fireAlerted == true) {
147               io.to(socket.id).emit("FireAlertAction", socket.id);
148           }
149           // Αποθήκευση δεδομένων κάδου στο Redis και στην MongoDB
150           saveWastebinToRedis(params);
151           saveWastebinToMongoDB(params);
152       });
```

Σχήμα 5.26: Socket Event Listeners για τη σύνδεση και υποδοχή δεδομένων από τον κάδο

Όταν ληφθεί μήνυμα με κλειδί “getWastebinFromRedis”, σημαίνει ότι έχουν ζητηθεί τα δεδομένα του κάδου από το Redis και το πρόγραμμα ξεκινάει να καταγράφει τον χρόνο ανάκτησης τους και καλεί την συνάρτηση “loadWastebinFromRedis” που θα ανακτήσει τα δεδομένα του κάδου (όνομα κάδου, socket id, θερμοκρασία, πληρότητα, πυρκαγιά, ημερομηνία και ώρα μετρήσεων) από την βάση δεδομένων redis. Μετά την επιτυχή ανάκτηση αποστέλλεται ο χρόνος ανάκτησης στον κάδο μέσω του διαύλου socket με κλειδί “ WastebinTimeFromRedis” (Σχήμα 5.27).

Όταν ληφθεί μήνυμα με κλειδί “getWastebinFromMongoDB”, σημαίνει ότι έχουν ζητηθεί τα δεδομένα του κάδου από την mongoDB και το πρόγραμμα ξεκινάει να καταγράφει τον χρόνο ανάκτησης τους και καλεί την συνάρτηση “ loadWastebinFromMongoDB” που θα ανακτήσει τα δεδομένα του κάδου (όνομα κάδου, socket id, θερμοκρασία, πληρότητα, πυρκαγιά, ημερομηνία και ώρα μετρήσεων) από την βάση δεδομένων mongoDB. Μετά την επιτυχή ανάκτηση αποστέλλεται ο χρόνος ανάκτησης στον κάδο μέσω του διαύλου socket με κλειδί “ WastebinTimeFromMongoDB” (Σχήμα 5.27).

```
rednode > JS server.js > io.on("connection") callback
152     });
153     // Όταν έρθει μήνυμα για ανάκτηση δεδομένων από το Redis
154     socket.on("getWastebinFromRedis", async(params) => {
155         console.log("Ανάκτηση Δεδομένων από τον Κάδο: ", params);
156           // Καταγραφή χρόνου ανάκτησης δεδομένων
157           var redisTime = await loadWastebinFromRedis(params);
158           if  (!redisTime) {
159             redisTime = 0;
160           }
161         console.log("redisTime: ", redisTime);
162         // και αποστολή στο sensor μηνύματος με τον χρόνο ανάκτησης
163         io.to(socket.id).emit("WastebinTimeFromRedis", redisTime);
164     })
165     // Όταν έρθει μήνυμα για ανάκτηση δεδομένων από το MongoDB
166     socket.on("getWastebinFromMongoDB", async(params) => {
167         console.log("Ανάκτηση Δεδομένων από τον Κάδο: ", params);
168           // Καταγραφή χρόνου ανάκτησης δεδομένων
169           var mongoDBTime = await loadWastebinFromMongoDB(params);
170           if  (mongoDBTime) {
171             console.log("mongoDBTime: ", mongoDBTime);
172             // και αποστολή στο sensor μηνύματος με τον χρόνο ανάκτησης
173             io.to(socket.id).emit("WastebinTimeFromMongoDB", mongoDBTime);
174           }
175     })
176 });
```

Σχήμα 5.27: Socket Event Listeners για ανάκτηση δεδομένων και χρονομέτρηση της από τις βάσεις Redis και MongoDB

Η συνάρτηση “saveWastebinToRedis” λειτουργεί ασύγχρονα ανακτώντας τα προηγούμενα μηνύματα από την βάση και προσθέτοντας το νέο, μετατρέποντας τα μηνύματα από μορφή json σε μορφή συμβολοσειράς (string) και αποθηκεύοντας τα στην βάση δεδομένων μνήμης redis (Σχήμα 5.28).

```
rednode > JS server.js > saveWastebinToRedis
178 // Ρουτίνα αποθήκευσης δεδομένων κάδου (μέσα από το μήνυμα του socket) στο Redis
179 const saveWastebinToRedis = async(message) =>{
180     try {
181         console.log("saveWastebinToRedis: ", message);
182         const { wastebinName } = message;
183         console.log("const { key } = message: ", wastebinName);
184         console.log("message: ", message);
185
186         const data = await redisGetAsync(wastebinName);
187
188         // the first message in the chat room
189         if (!data) {
190             console.log("redisClient.set(wastebinName");
191             const json = [];
192             json.push(message);
193             return redisClient.set(wastebinName, JSON.stringify(json));
194         }
195
196         const json = JSON.parse(data);
197         //console.log("JSON.parse(data)", json);
198         json.push(message);
199
200         redisClient.set(wastebinName, JSON.stringify(json));
201         //console.log("redisClient.set: ", JSON.stringify(json));
202
203     } catch (error) {
204         console.error(error);
205     }
206 };
```

Σχήμα 5.28: Συνάρτηση αποθήκευσης δεδομένων στο Redis

Η συνάρτηση “loadWastebinFromRedis” λειτουργεί ασύγχρονα ανακτώντας τα προηγούμενα μηνύματα από την βάση και υπολογίζοντας τον χρόνο ανάκτησης τον επιστρέφει σε μορφή datetime (Σχήμα 5.29).

```
rednode > JS server.js > ...
207
208   // Συνάρτηση ανάκτησης δεδομένων κάδου από το Redis και χρονομέτρηση της
209   const loadWastebinFromRedis = async(message) =>{
210       try {
211           console.log("loadWastebinFromRedis: ", message);
212           console.log("message: ", message);
213
214           const startTime = Date.now();
215           // Do your operations
216           const data = await redisGetAsync(message);
217           const endTime = Date.now();
218           if (data) {
219               const timeTaken = endTime - startTime;
220               console.log("return data");
221               console.log(startTime);
222               console.log(endTime);
223               console.log(`REDIS: Call to retrieve Wastebin took ${timeTaken} milliseconds.`);
224               return timeTaken;
225           }
226       } catch (error) {
227           console.error(error);
228       }
229   };
```

Σχήμα 5.29:Συνάρτηση ανάκτησης δεδομένων από το Redis

Η συνάρτηση “saveWastebinToMongoDB” λειτουργεί ασύγχρονα αποσυνθέτοντας το μήνυμα στα συστατικά του που αποτελούν τα δεδομένα του κάδου και αποθηκεύοντας τα στην βάση mongoDB χρησιμοποιώντας την βιβλιοθήκη “mongoose” (Σχήμα 5.30).

```
rednode > JS server.js > saveWastebinToMongoDB > newWastebin
230   // Ρουτίνα αποθήκευσης δεδομένων κάδου (μέσα από το μήνυμα του socket) στην MongoDB
231   const saveWastebinToMongoDB = async(message) =>{
232       try {
233           const { wastebinName } = message;
234           const { wastebinIp } = message;
235           const { temperature } = message;
236           const { occupancyRate } = message;
237           const { fireAlerted } = message;
238           const { createdAt } = message;
239
240           const db = mongoose.connection;
241
242           console.log("MONGODB newWastebin");
243
244           // Bind connection to error event (to get notification of connection errors)
245           db.on("error", console.error.bind(console, "MongoDB connection error:"));
246           const newWastebin = new Wastebin({
247               "wastebinName": wastebinName,
248               "wastebinIp": wastebinIp,
249               "temperature": temperature,
250               "occupancyRate": occupancyRate,
251               "fireAlerted": fireAlerted,
252               "createdAt": createdAt
253           });
254
255           newWastebin.save((err) => {
256               if (err) {
257                   console.error(err);
258                   return handleError(err);
259               } else
260                   console.log("Η εγγραφή αποθηκεύτηκε επιτυχώς στο collection Wastebin της mongoDb");
261           });
262       } catch (tryerror) {
263           console.error(tryerror);
264       }
```

Σχήμα 5.30:Ρουτίνα αποθήκευσης δεδομένων στη MongoDB

Η συνάρτηση “loadWastebinFromMongoDB” λειτουργεί ασύγχρονα ανακτώντας τα προηγούμενα μηνύματα από την βάση mongoDB και υπολογίζοντας τον χρόνο ανάκτησης τον επιστρέφει σε μορφή datetime (Σχήμα 5.31).

```
rednode > JS server.js > ...
266   // Συνάρτηση ανάκτησης δεδομένων κάδου από την MongoDB και χρονομέτρηση της
267   const loadWastebinFromMongoDB = async(message) =>{
268       try {
269           const db = mongoose.connection;
270           console.log("MONGODB loadWastebinFromMongoDB");
271
272           // Bind connection to error event (to get notification of connection errors)
273           db.on("error", console.error.bind(console, "MongoDB connection error:"));
274
275           if(typeof message !== 'undefined'){
276               const startTime = Date.now();
277               const myWastebin = await  Wastebin.find({ wastebinName: message}).exec();
278               const endTime = Date.now();
279               console.log("Η εγγραφή ανακτήθηκε επιτυχώς από το collection Wastebin της mongoDb");
280               const timeTaken = endTime - startTime;
281               console.log(startTime);
282               console.log(endTime);
283               console.log(`MONGODB: Call to retrieve Wastebin took ${timeTaken} milliseconds.`);
284               return timeTaken;
285           }
286       } catch (tryerror) {
287           console.error(tryerror);
288       }
289   }
```

Σχήμα 5.31: Ρουτίνα ανάκτησης δεδομένων από την MongoDB

Εκτέλεση του διακομιστή ιστού στην IP διεύθυνση του τοπικού μηχανήματος (0.0.0.0) και στην πόρτα που ορίζεται από τις παραμέτρους (Σχήμα 5.32).

```
rednode > JS server.js > ...
290
291   // Εκτέλεση Web server στην localhost IP
292   server.listen(port, '0.0.0.0', () =>
293       console.log(`Red Node listening on port ${port}`)
294   );
```

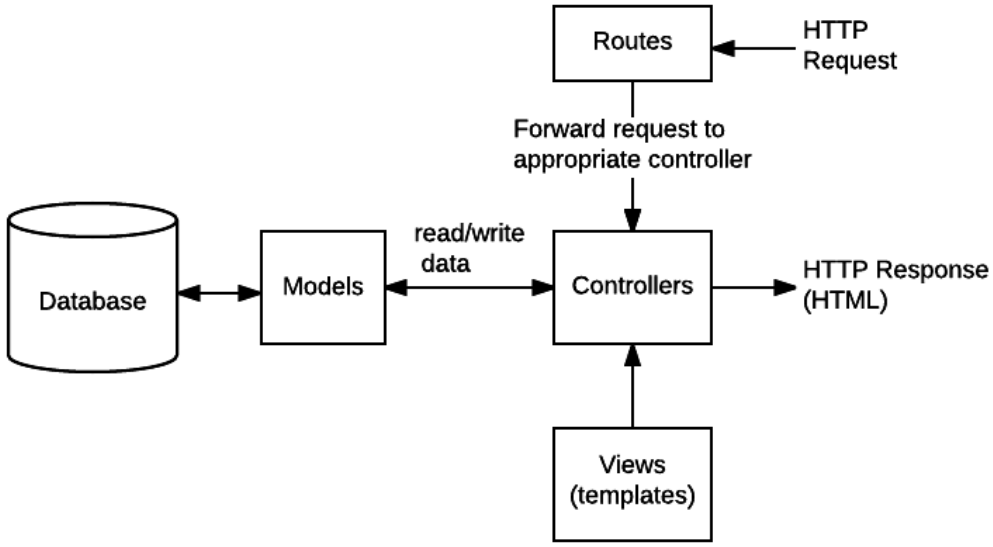
Σχήμα 5.32: Εκτέλεση Web Server Red Node

5.6 Ανάλυση κώδικα “mongoapi” microservice

5.6.1 Εισαγωγή

Η υπηρεσία mongoapi παρέχει τη απαραίτητη διασύνδεση και τις εντολές ελέγχου και διαχείρισης της βάσης δεδομένων mongoDB της εφαρμογής προς τις υπηρεσίες που θα χρειαστούν πρόσβαση στις τιμές των αισθητήρων όπως η ιστοσελίδα της υπηρεσίας result.

Έχει χρησιμοποιηθεί η αρχιτεκτονική MVC (Model – View - Controller) [24] (Σχήμα 5.33). Το αντικείμενο “Model” χρησιμοποιείται για να ορισθεί το σχήμα της βάσης, δηλαδή τα πεδία που θα αποθηκεύονται σε κάθε έγγραφο μαζί με τις εξ ορισμού τιμές και τους ελέγχους απαιτήσεων. Το αντικείμενο “Routes” χρησιμοποιείται για να προωθεί τα αιτήματα (και οποιαδήποτε άλλη πληροφορία περιλαμβάνεται στο αιτηθέν URL) προς τις κατάλληλες συναρτήσεις του “Controller”. Το αντικείμενο “Controller” για την ανάκτηση των αιτούμενων δεδομένων αλληλοεπιδρά με το αντικείμενο “Model” και επιστρέφει HTTP response με το αποτέλεσμα της αίτησης. Το αντικείμενο “Views (templates)” δεν χρησιμοποιείται στο “mongoapi” γιατί υλοποιείται στον περιέκτη “result”.



Σχήμα 5.33: Διάγραμμα αρχιτεκτονικής MVC (Model – View - Controller)

5.6.2 Ανάλυση αρχείου Dockerfile

Το αρχείο “Dockerfile” (Σχήμα 5.34) δημιουργεί τον περιέκτη (container) της Node.js εφαρμογής λαμβάνοντας από το Docker Hub την “node:15” εικόνα, εγκαθιστώντας το Node.js στον περιέκτη, αντιγράφοντας στον περιέκτη τα αρχεία του πηγαίου κώδικα, εγκαθιστώντας το npm, και μετά εκκινεί τον Node.js εξυπηρετητή ιστού (web server) της εφαρμογής.

mongoapi > Dockerfile > ...

```
1   FROM node:15
2   WORKDIR /app
3   COPY package.json .
4   #RUN npm install --only=production
5   ARG NODE_ENV
6   RUN if [ "$NODE_ENV" = "development" ]; \
7           then npm install; \
8           else npm install --only=production; \
9           fi
10
11  COPY . ./
12  ENV PORT 5000
13  EXPOSE $PORT
14  CMD ["node", "server.js"]
```

Σχήμα 5.34: Αρχείο Dockerfile της υπηρεσίας mongoapi

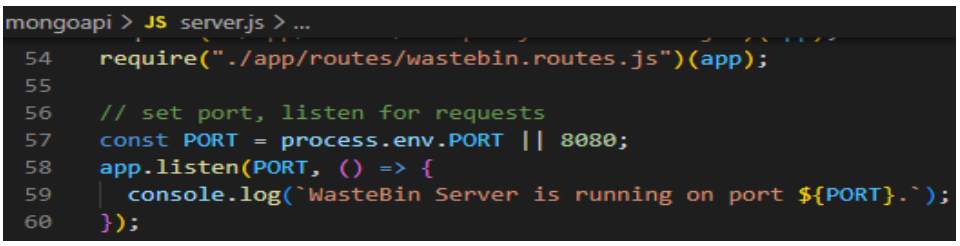
5.6.3 Ανάλυση αρχείου Server.js

Το αρχείο server.js περιέχει τον κώδικα που διαχειρίζεται την σύνδεση με τον περιέκτη “mongodb” που αποτελεί την βάση δεδομένων mongoDB.

Στην αρχή δηλώνεται το αρχείο “db.config” που περιέχει τις μεταβλητές “MONGO_USER”, “MONGO_PASSWORD”, “MONGO_IP”, “MONGO_PORT”, που χρησιμοποιούνται στο url της σύνδεσης με τη βάση. Στη συνέχεια δηλώνεται το αρχείο “app/models” που περιέχει την βιβλιοθήκη mongoose που χρησιμοποιείται για την δημιουργία του σχήματος της βάσης δεδομένων. Μέσω της mongoose επιτυγχάνεται η σύνδεση με την mongoDB και η διαχείριση της. Η εφαρμογή μέχρι να είναι επιτυχής η σύνδεση προσπαθεί επαναληπτικά να συνδεθεί κάθε 5 δευτερόλεπτα. (σχήμα 5.35).

Σχήμα 5.35: Σύνδεση με MongoDb INSIDE container

Η εφαρμογή ορίζει το αρχείο “wastebin.routes.js” που περιέχει τα routes που περιγράφουν πως τα URIs (Uniform Resource Identifiers) (Ενιαία Αναγνωριστικά Πόρων) της εφαρμογής ανταποκρίνονται στα αιτήματα των πελατών. Τέλος ορίζει την πόρτα στην οποία θα εκτελείται ο διακομιστής και η οποία θα «ακούει» και θα παραλαμβάνει τα αιτήματα (Σχήμα 5.36).



Σχήμα 5.36: Σύνδεση με αρχείο routes και εκτέλεση διακομιστή web

Το αρχείο “Wastebin.routes.js” έχει δημιουργηθεί για να περιέχει τις δρομολογήσεις που αφορούν την ενότητα “wastebin” έτσι ώστε να αποφύγουμε την συνάθροιση τους στο αρχείο “server.js”. Οι δρομολογήσει αφορούν αιτήματα βασικών λειτουργιών CRUD (Create Read Update Delete) (Δημιουργία Ανάγνωση Ενημέρωση Διαγραφή) που θα εφαρμοστούν στις τιμές των αισθητήρων του κάθε κάδου. Το κάθε αίτημα καλεί διαφορετική συνάρτηση – μέθοδο του αντικειμένου “controller” (Σχήμα 5.37).

Σχήμα 5.37: Routes methods που χρησιμοποιούνται για την εξυπηρέτηση των HTTP requests

Το αρχείο “Wastebin.model.js” έχει δημιουργηθεί για να περιέχει το σχήμα της βάσης δεδομένων “Wastebin” στο οποίο ορίζονται τα πεδία κάθε πίνακα ή εγγράφου μαζί με τις απαιτήσεις ελέγχου και τις αρχικές τιμές. Η μέθοδος “toJSON” μετατρέπει το εξορισμού πεδίο “_id” της mongoDB σε “id” για διευκόλυνση στην περαιτέρω χρήση του (Σχήμα 5.38).

Σχήμα 5.38: Database Model για την δημιουργία και διαχείριση της βάσης δεδομένων των αισθητήρων του κάδου wastebin στην mongoDB

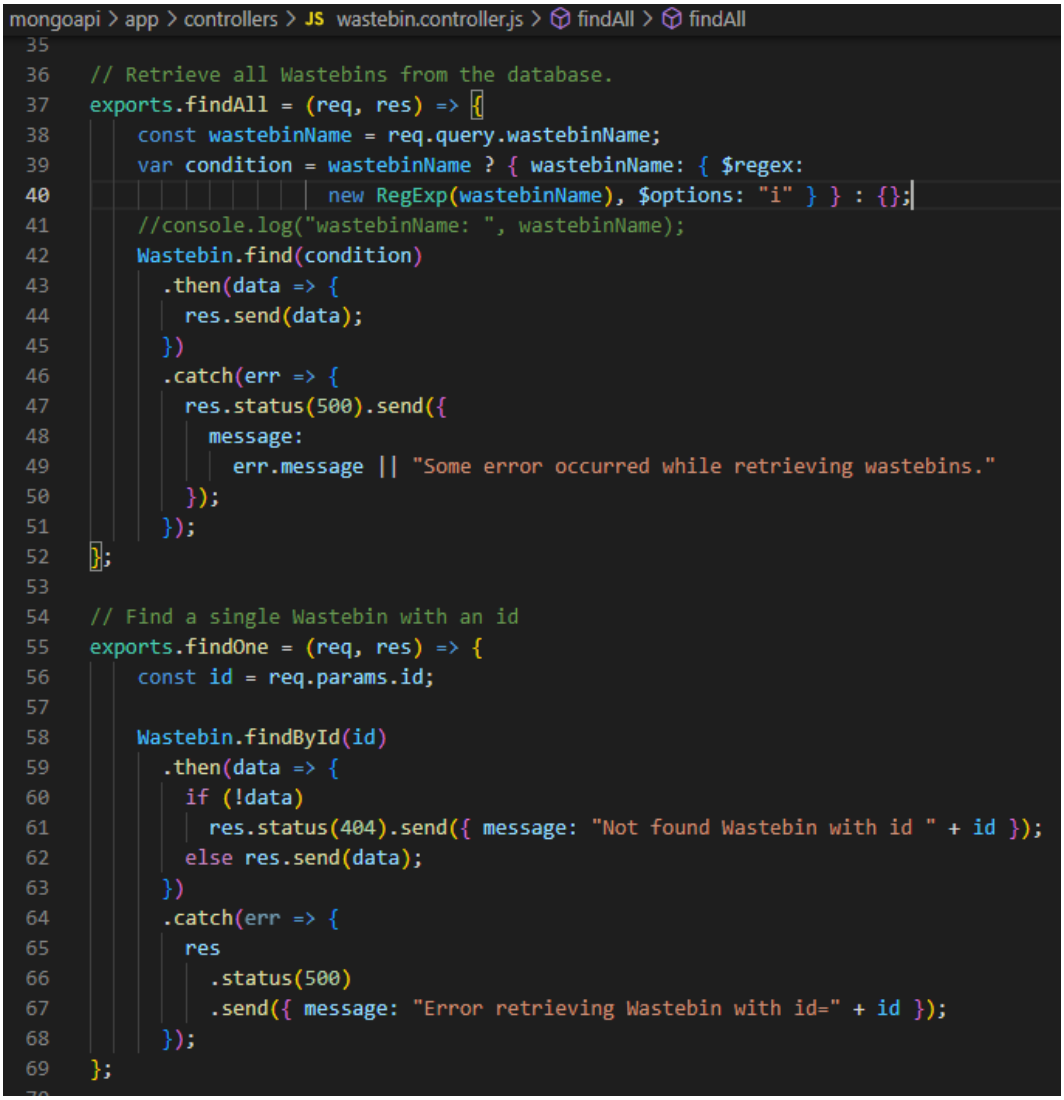
Το αρχείο “Wastebin.controller.js” έχει δημιουργηθεί για να περιέχει τις μεθόδους που υλοποιούν τα αιτήματα CRUD (Create Read Update Delete) προς το αντικείμενο της βάσης δεδομένων “Wastebin” που μας το παρέχει το “wastebin.model”. Η πρώτη μέθοδος “create” δημιουργεί ένα νέο αντικείμενο “wastebin” και αφού αναθέσει τιμές στα πεδία του από τις αντίστοιχες μεταβλητές του αιτήματος το αποθηκεύει στην mongoDB (Σχήμα 5.39).

mongoapi > app > controllers > JS wastebin.controller.js > findAll

```
const db = require("../models");
const Wastebin = db.wastebins;

// Create and Save a new Wastebin
exports.create = (req, res) => {
  // Validate request
  if (!req.body.wastebinName) {
    res.status(400).send({ message: "wastebinName can not be empty!" });
    return;
  }

  // Create a Wastebin
  const wastebin = new Wastebin({
    wastebinName: req.body.wastebinName,
    wastebinIp: req.body.wastebinIp,
    temperature: req.body.temperature,
    occupancyRate: req.body.occupancyRate,
    createdAt: req.body.createdAt,
    fireAlerted: req.body.fireAlerted ? req.body.fireAlerted : false
  });

  // Save Wastebin in the database
  wastebin
    .save(wastebin)
    .then(data => {
      res.send(data);
    })
    .catch(err => {
      res.status(500).send({
        message:
          err.message || "Some error occurred while creating the Wastebin."
      });
    });
};

```

Σχήμα 5.39: Μέθοδος create του controller “wastebin.controller”

Η μέθοδος “findAll” επιστρέφει τα αντικείμενα όλων των κάδων ή αυτών που ταιριάζουν βάση ονόματος κάδου, μερικού ή ολικού, που περιέχουν τις τιμές όλων των πεδίων. Η μέθοδος “findOne” επιστρέφει το αντικείμενο ενός συγκεκριμένου κάδου βάση κωδικού “id” κάδου, που περιέχει τις τιμές όλων των πεδίων. (Σχήμα 5.40).



Σχήμα 5.40: Μέθοδοι findAll και findOne του controller “wastebin.controller”

Η μέθοδος “update” χρησιμοποιώντας την μέθοδο “findByIdAndUpdate” επιστρέφει το αντικείμενο του κάδου με το συγκεκριμένο id αφού πρώτα ενημερώσει τα πεδία του με νέες τιμές. Η μέθοδος “delete” χρησιμοποιώντας την μέθοδο “findByIdAndRemove” επιστρέφει το αντικείμενο του κάδου με το συγκεκριμένο id και το σβήνει από την βάση δεδομένων (Σχήμα 5.41).

mongoapi > app > controllers > JS wastebin.controller.js > ...

```
71  // Update a Wastebin by the id in the request
72  exports.update = (req, res) => {
73      if (!req.body) {
74          return res.status(400).send({
75            message: "Data to update can not be empty!"
76          });
77        }
78
79        const id = req.params.id;
80
81        Wastebin.findByIdAndUpdate(id, req.body, { useFindAndModify: false })
82          .then(data => {
83            if (!data) {
84              res.status(404).send({
85                message: `Cannot update Wastebin with id=${id}. Maybe Wastebin was not found!`
86              });
87            } else res.send({ message: "Wastebin was updated successfully." });
88          })
89          .catch(err => {
90            res.status(500).send({
91              message: "Error updating Wastebin with id=" + id
92            });
93          });
94  };
95
96  // Delete a Wastebin with the specified id in the request
97  exports.delete = (req, res) => {
98      const id = req.params.id;
99
100     Wastebin.findByIdAndRemove(id)
101       .then(data => {
102         if (!data) {
103           res.status(404).send({
104             message: `Cannot delete Wastebin with id=${id}. Maybe Wastebin was not found!`
105           });
106         } else {
107           res.send({
108             message: "Wastebin was deleted successfully!"
109           });
110         }
111       })
112       .catch(err => {
113         res.status(500).send({
114           message: "Could not delete Wastebin with id=" + id
115         });
116       });
117 };
```

Σχήμα 5.41: Μέθοδοι update και findByIdAndRemove του controller “wastebin.controller

5.7 Ανάλυση κώδικα “result” microservice

5.7.1 Εισαγωγή

Η υπηρεσία result αποτελείται από μία ιστοσελίδα που κάνοντας χρήση της υπηρεσίας mongoapi απεικονίζει και διαχειρίζεται τα δεδομένα των αισθητήρων των κάδων σε ένα λειτουργικό περιβάλλον που έχει δημιουργηθεί με το JavaScript πλαίσιο Vue.js.

5.7.2 Ανάλυση αρχείου Dockerfile

Το αρχείο “Dockerfile” (Σχήμα 5.42) δημιουργεί τον περιέκτη (container) της Node.js εφαρμογής λαμβάνοντας από το Docker Hub την “node:15” εικόνα, εγκαθιστώντας το Node.js στον περιέκτη, αντιγράφοντας στον περιέκτη τα αρχεία του πηγαίου κώδικα, εγκαθιστώντας το npm έκδοση 9.1.3, και μετά εκκινεί τον Node.js εξυπηρετητή ιστού (web server) της εφαρμογής.

result > Dockerfile > ...

```
1   FROM node:15
2   WORKDIR /app
3   COPY package.json .
4   #RUN npm install --only=production
5   ARG NODE_ENV
6   RUN if [ "$NODE_ENV" = "development" ]; \
7           then npm install -g npm@9.1.3; \
8           else npm install --only=production; \
9           fi
10
11  COPY . ./
12  ENV PORT 8081
13  EXPOSE $PORT
14  CMD ["node", "index.js"]
```

Σχήμα 5.42: Αρχείο Dockerfile της υπηρεσίας result

Το αρχείο “index.html” αποτελεί την αρχική ιστοσελίδα η οποία δεν περιέχει εντολές εμφάνισης κάποιων πεδίων ή ετικετών αλλά ορίζει το αρχικό script “main.js” και το κεντρικό τμήμα “div” με την ονομασία “app” που χρησιμοποιείται ως το κέλυφος για τον περαιτέρω κώδικα (Σχήμα 5.43).

result > index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <meta charset="UTF-8" />
5       <link rel="icon" href="/favicon.ico" />
6       <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7       <title>Wastebin Center</title>
8     </head>
9     <body>
10      <div id="app">
11
12      </div>
13      <script type="module" src="/src/main.js" defer></script>
14    </body>
15  </html>
```

Σχήμα 5.43: Αρχείο index.html

Το αρχείο app.vue αποτελεί το πρότυπο για την δημιουργία της γραμμής μενού και της κεντρικής περιοχής της αρχικής ιστοσελίδας συνδυάζοντας HTML, CSS και JavaScript. Το “v-app” της βιβλιοθήκης vuetify είναι το βασικό στοιχείο που περικλείει όλα τα υπόλοιπα στοιχεία. Η γραμμή μενού δημιουργείται με το στοιχείο “v-app-bar” ενώ το στοιχείο “v-main” αλλάζει δυναμικά μέγεθος ανάλογα με την δομή των στοιχείων εμφάνισης. Μέσα στο “v-main” δηλώνεται το στοιχείο “router-view” που εμφανίζει το συστατικό ή την μήτρα που ορίζει η συγκεκριμένη διαδρομή (route) (Σχήμα 5.44).

result > src > App.vue > {} template > v-app > v-app-bar

```vue
<template>
  <v-app>
    <v-app-bar app dark>
      <div class="d-flex align-center mr-2">
        Πιλοτική Επικοινωνία Έξυπνων Κάδων μέσω Socket με χρήση Docker
      </div>


      <v-btn to="/wastebins" text>
        ΛΙΣΤΑ ΚΑΔΩΝ
      </v-btn>

      <v-btn to="/addWastebins" text>
        ΝΕΟΣ ΚΑΔΟΣ
      </v-btn>
    </v-app-bar>

    <v-main>
      <router-view />
    </v-main>
  </v-app>
</template>

<script>
export default {
  name: "app",
};
</script>
```

Σχήμα 5.44: Αρχείο App.vue

Το αρχείο “main.js” είναι το σημείο εισόδου της εφαρμογής που αρχικοποιεί το κεντρικό στοιχείο της ιστοσελίδας. Επίσης είναι υπεύθυνο για την σύνδεση plugins όπως και στοιχείων άλλων κατασκευαστών σαν το “vuetify” (Σχήμα 5.45)

result > src > JS main.js > ...

```js
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import vuetify from './plugins/vuetify'
import './assets/main.css'

Vue.config.productionTip = false;

new Vue({
  router,
  vuetify,
  render: (h) => h(App)
}).$mount('#app')
```

Σχήμα 5.45: Αρχείο Main.js

Οι σύγχρονες εφαρμογές μονής ιστοσελίδας (single-page), όπως είναι οι εφαρμογές που χρησιμοποιούν “Vue”, μπορούν να πλοηγούνται από σελίδα σε σελίδα χωρίς να αιτούνται στον διακομιστή. Το αρχείο “router.js” χρησιμοποιεί την βιβλιοθήκη “vue-ruter” για την πλοήγηση από σελίδα σε σελίδα χρησιμοποιώντας τα routes που έχουν οριστεί στο αντικείμενο “Router” (Σχήμα 5.46).

result > src > JS router.js > ...

```
import Vue from "vue";
import Router from "vue-router";

Vue.use(Router);

export default new Router({
  mode: "history",
  routes: [
    {
      path: "/",
      alias: "/wastebins",
      name: "wastebins",
      component: () => import("./components/WastebinsList")
    },
    {
      path: "/wastebins/:id",
      name: "wastebin-details",
      component: () => import("./components/Wastebin")
    },
    {
      path: "/addWastebins",
      name: "wastebins-new",
      component: () => import("./components/AddWastebin")
    },
  ]
});
```

Σχήμα 5.46: Αρχείο Router.js

Το “Axios” είναι μία βιβλιοθήκη HTTP πελάτη που χρησιμοποιεί υποσχέσεις (promises) και εκτελείται τόσο στον πελάτη όσο και στον διακομιστή. Είναι ιδανική για ανάκτηση δεδομένων κατά την απόδοση από την πλευρά του διακομιστή. Στο αρχείο “Http-common.js” δημιουργείται μία κοινή βάση στιγμιότυπου (instance) που επιτρέπει την διαμοίραση μιας κοινής διεύθυνσης URL σε όλες τις κλήσεις προς αυτό το στιγμιότυπο καθώς όλες αφορούν έναν συγκεκριμένο διακομιστή (Σχήμα 5.47).

result > src > JS http-common.js > ...

```
import axios from "axios";

export default axios.create({
  //baseURL: "http://localhost:4001/api/v2",
  baseURL: "http://localhost:8080/api",
  headers: {
    "Content-type": "application/json"
  }
});
```

Σχήμα 5.47: Αρχείο Http-common.js

Το αρχείο “WastebinDataService.js” περιλαμβάνει – παρέχει τις μεθόδους που θα χρησιμοποιήσουν τα “.vue” αρχεία για να εκτελέσουν HTTP εντολές όπως “http.get()”, “http.post”, “http.put”, “http.delete” (Σχήμα 5.48).

Σχήμα 5.48: Αρχείο WastebinDataService

Το αρχείο “WastebinsList.vue” αποτελείται από δύο μέρη, το πρότυπο (template) και το “script”. Το πρότυπο δημιουργεί το κυρίως σώμα της ιστοσελίδας στην οποία εμφανίζονται ένα πεδίο αναζήτησης, ένα κουμπί ανανέωσης και η λίστα των κάδων με τις τιμές των αισθητήρων τους και την δυνατότητα επεξεργασίας ή διαγραφής των τιμών ανά γραμμή ή καθολικά (Σχήμα 5.49).

result > src > components > WastebinsList.vue > {} template

```vue
<template>
  <v-row align="center" class="list px-3 mx-auto">
    <v-col cols="12" md="8">
      <v-text-field v-model="title" label="Αναζήτηση με Όνομα Κάδου"></v-text-field>
    </v-col>

    <v-col cols="12" md="2">
      <v-btn small @click="searchTitle">
        ΑΝΑΖΗΤΗΣΗ
      </v-btn>
    </v-col>

    <v-col cols="12" md="2">
      <v-btn small @click="refreshPage">
        ΑΝΑΝΕΩΣΗ
      </v-btn>
    </v-col>

    <v-col cols="12" sm="12">
      <v-card class="mx-auto" tile>
        <v-card-title>Στοιχεία Αισθητήρων Κάδων</v-card-title>

        <v-data-table
          :headers="headers"
          :items="wastebins"
          enable-pagination
          :hide-default-footer="false"
        >
          <template v-slot:[`item.actions`]="{ item }">
            <v-icon small class="mr-2" @click="editWastebin(item.id)">mdi-pencil</v-icon>
            <v-icon small @click="deleteWastebin(item.id)">mdi-delete</v-icon>
          </template>
        </v-data-table>

        <v-card-actions v-if="wastebins.length > 0">
          <v-btn small color="error" @click="removeAllWastebins">
            Remove All
          </v-btn>
        </v-card-actions>
      </v-card>
    </v-col>
  </v-row>
</template>
```

Σχήμα 5.49: Αρχείο WastebinsList.vue - Template

Στο script ορίζονται οι επικεφαλίδες της λίστας και καθορίζονται ιδιότητες τους όπως η στοίχιση και η ταξινόμηση ανά κολόνα. Επίσης αναπτύσσονται οι μέθοδοι “retrieveWastebins” και “removeAllWastebins” που καλούν τις μεθόδους “WastebinDataService.getAll” και “WastebinDataService.deleteAll” αντίστοιχα από το αρχείο “WastebinDataService.js” (Σχήμα 5.50).

result > src > components > WastebinsList.vue > {} script

```
<script>
import WastebinDataService from "../services/WastebinDataService";
import moment from 'moment';
export default {
  name: "wastebins-list",
  data() {
    return {
      wastebins: [],
      title: "",
      headers: [
        { text: "Κάδος", align: "start", sortable: true, value: "wastebinName" },
        { text: "θερμοκρασία", value: "temperature", sortable: false },
        { text: "Πληρότητα", value: "occupancyRate", sortable: false },
        { text: "Φωτιά", value: "fireAlert", sortable: false },
        { text: "Ημ/νία - Ώρα", value: "createdAt", sortable: true },
        { text: "Actions", value: "actions", sortable: false },
      ],
    };
  },
  methods: {
    retrieveWastebins() {
      WastebinDataService.getAll()
        .then((response) => {
          this.wastebins = response.data.map(this.getDisplayWastebin);
          console.log(response.data);
        })
        .catch((e) => {
          console.log(e);
        });
    },

    refreshList() {
      this.retrieveWastebins();
    },

    removeAllWastebins() {
      WastebinDataService.deleteAll()
        .then((response) => {
          console.log(response.data);
          this.refreshList();
        })
        .catch((e) => {
          console.log(e);
        });
    },

```

Σχήμα 5.50: Αρχείο WastebinList.vue -Script (ορισμός τίτλων grid, μέθοδοι retrieveWastebins, refreshList, removeAllWastebins).

Η μέθοδος “searchTitle” καλεί την μέθοδο “WastebinDataService.findByTitle” που επιστρέφει τα στοιχεία των κάδων που συμφωνούν με την περιγραφή που δίνει ο χρήστης στο πεδίο αναζήτησης. Η μέθοδος “editWastebin(id)” καλεί την διαδρομή “route” με όνομα “wastebin-details” που κάνει εισαγωγή το πρότυπο σελίδας “Wastebin.view” έτσι ώστε να εμφανιστεί νέο παράθυρο με τα στοιχεία του κάδου με το συγκεκριμένο id προς αλλαγή στοιχείων. Η μέθοδος “deleteWastebin(id)” καλεί την μέθοδο WastebinDataService.delete(id) η οποία σβήνει την εγγραφή του κάδου με το συγκεκριμένο id. Η μέθοδος “getDisplayWastebin(wastebin)” με όρισμα το αντικείμενο του κάδου, επιστρέφει τις τιμές

των πεδίων του κάδου και τις αναθέτει σε μεταβλητές κόβοντας τους επιπλέον χαρακτήρες μετά τους 30 πρώτους για να χωρέσουν στο πλέγμα εμφάνισης (Σχήμα 5.51).

Σχήμα 5.51: Αρχείο WastebinList.vue vue – Script (μέθοδοι searchTitle, refreshPage, deleteWastebin, getDisplayWastebin).

Το αρχείο “Wastebin.vue” αποτελείται από δύο μέρη, το πρότυπο (template) και το “script”. Το πρότυπο δημιουργεί το κυρίως σώμα της ιστοσελίδας στην οποία εμφανίζεται η φόρμα καταχώρισης στοιχείων ενός κάδου στην οποία υπάρχει έλεγχος για τα πεδία που απαγορεύεται αν είναι κενά (Σχήμα 5.52).

Σχήμα 5.52: Wastebin.view Πρότυπο (template)

Στο script η μέθοδος “getWastebin(id)” καλεί την μέθοδο “WastebinDataService.get(id)” που επιστρέφει τα στοιχεία του κάδου με το συγκεκριμένο id. Η μέθοδος “ updateWastebin” καλεί την μέθοδο “WastebinDataService.update” που ενημερώνει τα στοιχεία του συγκεκριμένου κάδου. Η μέθοδος “deleteWastebin” καλεί την μέθοδο WastebinDataService.delete(this.currentWastebin.id) η οποία σβήνει την εγγραφή του κάδου με το συγκεκριμένο id (Σχήμα 5.53).

result > src > components > Wastebin.vue > {} script

```
<script>
import WastebinDataService from "../services/WastebinDataService";

export default {
  name: "wastebin",
  data() {
    return {
      currentWastebin: null,
      message: "",
    };
  },
  methods: {
    getWastebin(id) {
      WastebinDataService.get(id)
        .then((response) => {
          this.currentWastebin = response.data;
          console.log(response.data);
        })
        .catch((e) => {
          console.log(e);
        });
    },

    updateWastebin() {
      WastebinDataService.update(this.currentWastebin.id, this.currentWastebin)
        .then((response) => {
          console.log(response.data);
          this.message = "The wastebin was updated successfully!";
        })
        .catch((e) => {
          console.log(e);
        });
    },

    deleteWastebin() {
      WastebinDataService.delete(this.currentWastebin.id)
        .then((response) => {
          console.log(response.data);
          this.$router.push({ name: "wastebins" });
        })
        .catch((e) => {
          console.log(e);
        });
    },
```

Σχήμα 5.53: Wastebin.view script – getWastebin, updateWastebin, deleteWastebin.

Η μέθοδος “updateFireAlerted(status)” καλεί την μέθοδο “WastebinDataService.update()” που ενημερώνει τα στοιχεία του κάδου με το status του συναγερμού πυρκαγιάς. Η μέθοδος “mounted” καλείται την στιγμή που το Vue καταστήσει τα εικονικά στοιχεία DOM ορατά στο αληθινό DOM.(Σχήμα 5.54).

result > src > components > Wastebin.vue > {} script > default > methods > delet

```
128       updateFireAlerted(status) {
129         var data = {
130           id: this.currentWastebin.id,
131           wastebinName: this.currentWastebin.wastebinName,
132           wastebinIp: this.currentWastebin.wastebinIp,
133           temperature: this.currentWastebin.temperature,
134           occupancyRate: this.currentWastebin.occupancyRate,
135           fireAlerted: status,
136           createdAt: this.currentWastebin.createdAt,
137         };
138
139         WastebinDataService.update(this.currentWastebin.id, data)
140           .then((response) => {
141             this.currentWastebin.fireAlerted = status;
142             console.log(response.data);
143           })
144           .catch((e) => {
145             console.log(e);
146           });
147       },
148     },
149     mounted() {
150       this.message = "";
151       this.getWastebin(this.$route.params.id);
152     },
153   };
154   </script>
```

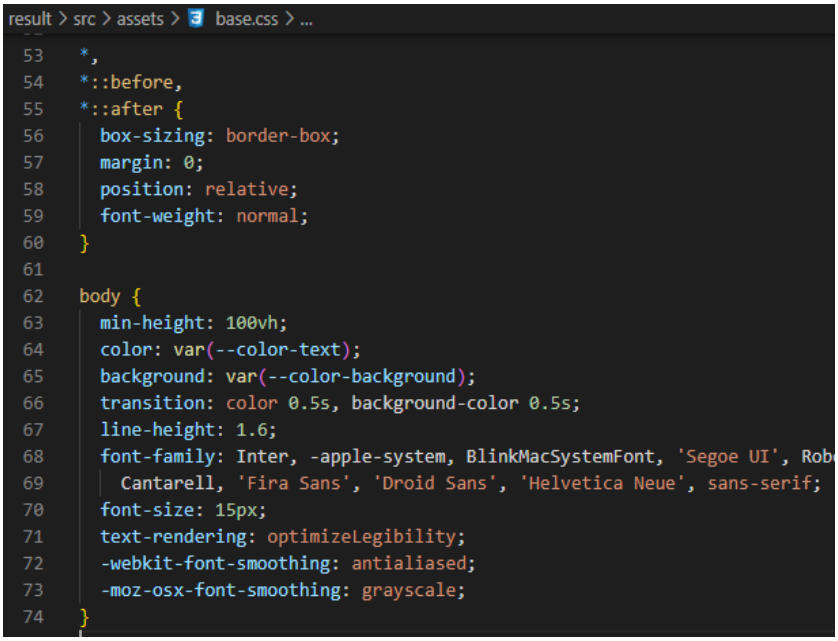
Σχήμα 5.54: Wastebin.view script – updateFireAlerted, mounted.

Το base.css είναι το βασικό αρχείο css στο οποίο ορίζονται λεκτικά και μεταβλητές για τα χρώματα και τους χρωματισμούς των αντικειμένων του DOM (Σχήμα 5.55) καθώς και άλλες παράμετροι που αφορούν το σώμα της html σελίδας όπως ύψος γραμμών, είδος γραμματοσειράς, μέγεθος γραμματοσειράς, χρώμα, θέση (Σχήμα 5.56).

result > src > assets > base.css > ...

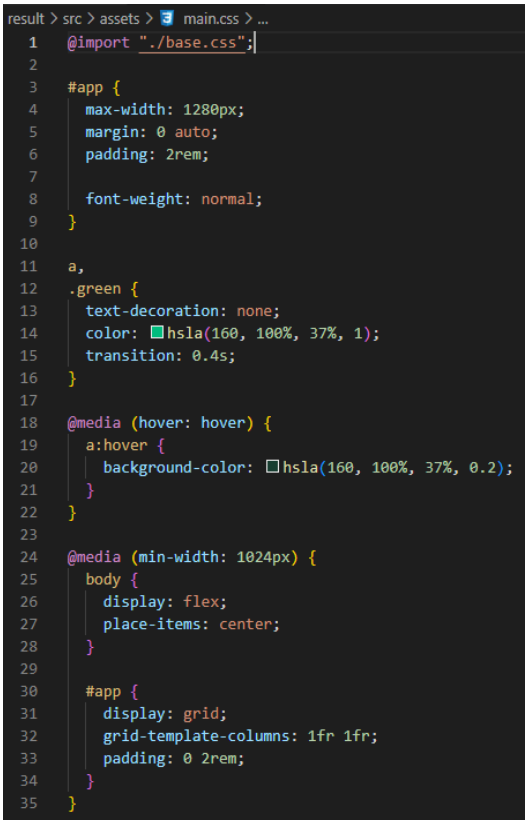
```
1   /* color palette from <https://github.com/vuejs/theme> */
2   :root {
3     --vt-c-white: #ffffff;
4     --vt-c-white-soft: #f8f8f8;
5     --vt-c-white-mute: #f2f2f2;
6
7     --vt-c-black: #181818;
8     --vt-c-black-soft: #222222;
9     --vt-c-black-mute: #282828;
10
11    --vt-c-indigo: #2c3e50;
12
13    --vt-c-divider-light-1: rgba(60, 60, 60, 0.29);
14    --vt-c-divider-light-2: rgba(60, 60, 60, 0.12);
15    --vt-c-divider-dadarkrk-1: rgba(84, 84, 84, 0.65);
16    --vt-c-divider-dark-2: rgba(84, 84, 84, 0.48);
17
18    --vt-c-text-light-1: var(--vt-c-indigo);
19    --vt-c-text-light-2: rgba(60, 60, 60, 0.66);
20    --vt-c-text-dark-1: var(--vt-c-white);
21    --vt-c-text-dark-2: rgba(235, 235, 235, 0.64);
22  }
```

Σχήμα 5.55: base.css Ορισμός χρωματικής παλέτας και μεταβλητών για τον χρωματισμό των αντικειμένων του DOM



Σχήμα 5.56: base.css – παράμετροι για το σώμα της html σελίδας

Το main.css είναι το κεντρικό αρχείο css που χρησιμοποιεί ως βάση το base.css. Σε αυτό ορίζονται τιμές για διαστήματα, στοιχίσεις, χρώματα σε ποιο ειδικά στοιχεία του DOM αντικειμένου (Σχήμα 5.57).



Σχήμα 5.57: mainl.css – Παράμετροι για ειδικά στοιχεία του DOM

5.8 Οικοδόμηση, εκτέλεση και τερματισμός των υπηρεσιών της εφαρμογής.

Η οικοδόμηση της εφαρμογής υλοποιείται με το Docker Compose εκτελώντας μέσα από τον φάκελο “dipl-demo” την εντολή “docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d –build” από την γραμμή εντολών του Ubuntu (Σχήμα 5.58). Η παράμετρος -s επιτρέπει την εκτέλεση ενναλακτικών αρχείων “docker-compose.yml”. Η εντολή “up” δημιουργεί και εκκινεί τους περιέκτες. Η παράμετρος -d υποχρεώνει τους περιέκτες να εκκινήσουν και να εκτελούνται στο παρασκήνιο. Η εντολή “--build” οικοδομεί ή ανοικοδομεί τις εικόνες που θα δημιουργήσουν στην συνέχεια τους περιέκτες.

```
user@39746W122064:~/dipl-demo$ docker-compose -f docker-compose.yml
-f docker-compose.dev.yml up -d --build
```

Σχήμα 5.58: Εντολή οικοδόμησης και εκκίνησης των περιεκτών της εφαρμογής

Η επιτυχής οικοδόμηση της εφαρμογής επιβεβαιώνεται με τα αποτελέσματα που εμφανίζονται στην γραμμή εντολών του Ubuntu (Σχήμα 5.59).

```
Step 9/9 : CMD ["node", "index.js"]
 ---> Using cache
 ---> 1d1af25795d7
Successfully built 1d1af25795d7
Successfully tagged dipl-demo_sensor:latest
Starting dipl-demo_result_1 ... done
Starting dipl-demo_redis_1  ... done
Starting dipl-demo_mongo_1  ... done
Starting dipl-demo_rednode_1  ... done
Starting dipl-demo_mongoapi_1 ... done
Starting dipl-demo_sensor_1   ... done
user@39746W122064:~/dipl-demo$
```

Σχήμα 5.59: Επιτυχής οικοδόμηση και εκκίνηση των περιεκτών της εφαρμογής

Ο έλεγχος επιτυχούς εκτέλεσης χωρίς την εμφάνιση λαθών επιτυγχάνεται με έλεγχο των περιεχομένων του αρχείου log καθενός από τα στιγμιότυπα των περιεκτών που εκτελούνται. Το αρχείο καταγραφής του στιγμιότυπου “dipl-demo_sensor_1” εμφανίζεται με την εντολή

“docker logs dipl-demo_sensor_1” και δεν δείχνει λάθη αλλά την αναμονή του “sensor” διακομιστή στην πόρτα 5000 (Σχήμα 5.60).

```
Starting dipl-demo_sensor_1   ... done
user@39746W122064:~/dipl-demo$ docker logs dipl-demo_sensor_1

> docker@1.0.0 dev
> nodemon -L index.js

[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Sensor listening on port 5000
```

Σχήμα 5.60: Αποτελέσματα log “docker logs dipl-demo_sensor_1”

Το αρχείο καταγραφής του στιγμιότυπου “dipl-demo_rednode_1” εμφανίζεται με την εντολή “docker logs dipl-demo_rednode_1” και δεν δείχνει λάθη αλλά την αναμονή του “rednode” διακομιστή στην πόρτα 5001 και την επιτυχή σύνδεση στην βάση δεδομένων MongoDB (Σχήμα 5.61).

```
MONGODB newWastebin
Η εγγραφή αποθηκεύτηκε επιτυχώς στο collection Wastebin της mongoDb
Συνδέθηκε στο Red Node

> docker@1.0.0 dev
> nodemon -L server.js

[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Red Node listening on port 5001
succesfully connected to MongoDB
```

Σχήμα 5.61: Αποτελέσματα log “docker logs dipl-demo_rednode_1”

Το αρχείο καταγραφής του στιγμιότυπου “dipl-demo_result_1” εμφανίζεται με την εντολή “docker logs dipl-demo_result_1” και δεν δείχνει λάθη αλλά μας ενημερώνει ότι το App εκτελείται στην διεύθυνση διακομιστή “http://localhost:8001/” και με ποιο τρόπο θα έχουμε πρόσβαση στο εσωτερικό του περιέκτη έτσι ώστε να εκτελέσουμε το “Vue CLI” (Σχήμα 5.62).

```
 DONE  Compiled successfully in 10962ms7:57:39 AM
<s> [webpack.Progress] 100%


  App running at:
  - Local:   http://localhost:8001/

  It seems you are running Vue CLI inside a container.
  Access the dev server via http://localhost:<your container's exte
rnal mapped port>/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

Σχήμα 5.62: Αποτελέσματα log “docker logs dipl-demo_result_1”

Ο τερματισμός της εφαρμογής υλοποιείται με το Docker Compose εκτελώντας μέσα από τον φάκελο “dipl-demo” την εντολή “docker-compose -f docker-compose.yml -f docker-compose.dev.yml down” από την γραμμή εντολών του Ubuntu (Σχήμα 5.63). Η εντολή “down” τερματίζει και καταστρέφει τους περιέκτες όπως και τα εικονικά δίκτυα που έχουν δημιουργηθεί μέσω του docker (Σχήμα 5.64).

```
user@39746W122064:~/dipl-demo$ docker-compose -f docker-compose.yml
 -f docker-compose.dev.yml down
```

Σχήμα 5.63: Εντολή τερματισμού εκτέλεσης των περιεκτών της εφαρμογής

```
user@39746W122064:~/dipl-demo$ docker-compose -f docker-compose.yml
 -f docker-compose.dev.yml down
Stopping dipl-demo_mongoapi_1 ... done
Stopping dipl-demo_sensor_1   ... done
Stopping dipl-demo_rednode_1  ... done
Stopping dipl-demo_mongo_1    ... done
Stopping dipl-demo_result_1   ... done
Stopping dipl-demo_redis_1    ... done
Removing dipl-demo_mongoapi_1 ... done
Removing dipl-demo_sensor_1   ... done
Removing dipl-demo_rednode_1  ... done
Removing dipl-demo_mongo_1    ... done
Removing dipl-demo_result_1   ... done
Removing dipl-demo_redis_1    ... done
Removing network dipl-demo_back-tier
Removing network dipl-demo_front-tier
```

Σχήμα 5.64: Αποτελέσματα τερματισμού εκτέλεσης των περιεκτών της εφαρμογής

6. Συμπεράσματα – Επόμενο Βήμα

6.1 Συμπεράσματα

Η δημιουργία εφαρμογής επικοινωνίας έξυπνων κάδων είναι ένα δύσκολο εγχείρημα γιατί συνδυάζει πολλές διαφορετικές τεχνολογίες πληροφορικής που πρέπει να παντρευτούν αποτελεσματικά με επιτυχία. Οι τεχνολογίες αυτές χωρίζονται σε τέσσερις μεγάλες οικογένειες, το Διαδίκτυο των Αντικειμένων - Internet of Things (IoT), το Υπολογιστικό Νέφος – Cloud Computing, την Περιεκτοποίηση – Containerization και τις Μικροϋπηρεσίες – Microservices.

Η χρήση τεχνολογίας εικονικοποίησης σε επίπεδο λογισμικού Docker μπορεί να παράγει με επιτυχία μικροϋπηρεσίες που είναι χαλαρά συνδεδεμένες μεταξύ τους αλλά συνεργάζονται άψογα. Οι Docker μικροϋπηρεσίες προσφέρουν εύκολη δημιουργία και διασύνδεση μέσω Docker compose και εύκολη διαχείριση και δυναμική κλιμάκωση μέσω Docker Swarm. Η δυνατότητα παράλληλης ελεύθερης χρήσης του αποθετηρίου εικόνων docker “Docker Hub” και του αποθετηρίου κώδικα “GitHub” προσδίδει στις εφαρμογές τον ευκολότερο τρόπο δημιουργίας, διαχείρισης και παράδοσης στον τελικό χρήστη – προγραμματιστή.

Επιτυγχάνεται η μέγιστη καθυστέρηση στην αμφίδρομη ανταλλαγή και αποθήκευση - ανάκτηση τιμών αισθητήρων να είναι στα λίγα ms μέσω ενός βελτιστοποιημένου συστήματος ανταλλαγής δεδομένων Socket.io με χρήση της βάσης δεδομένων στη μνήμη Redis και της βάσης δεδομένων τύπου εγγράφου MongoDB.

Παράγονται αποτελέσματα που αποδεικνύουν ότι η εκμετάλλευση των πληροφοριών των κάδων σε μεγάλη κλίμακα μέσω της χρήσης υπηρεσιών του υπολογιστικού νέφους όπως η Επιχειρηματική Ευφυία και η Ανάλυση Δεδομένων μεγάλης κλίμακας θα ενισχύσει την ποιότητα των υπηρεσιών διαχείρισης απορριμμάτων σε μία έξυπνη πόλη. Με αυτό τον τρόπο η πληροφορία που παράγεται από έναν αισθητήρα που βρίσκεται στο εσωτερικό ενός κάδου συλλογής απορριμμάτων είναι δυνατό να αξιοποιηθεί με τρόπο γρήγορο και έξυπνο και να δώσει στον πάροχο των υπηρεσιών την δυνατότητα να προβεί σε αποτελεσματικές ενέργειες και στον τελικό πελάτη τις υπηρεσίες που χρειάζεται για να διευκολύνει την ζωή του σε ποιοτικό και οικονομικό επίπεδο.

6.2 Επόμενο Βήμα:

Το επόμενο βήμα στην συνέχιση των ευρύτερων στόχων της διπλωματικής αποτελείται από βελτιώσεις σε όλους τους τομείς από την χρήση μικροελεγκτών έως την δημιουργία εφαρμογών νέφους.

Το κύκλωμα παραγωγής τυχαίων τιμών αισθητήρων μπορεί να αντικατασταθεί από ένα πραγματικό κύκλωμα αισθητήρων που θα αποτελούν μέρος ενός μικροελεγκτή με την δυνατότητα τοποθέτησης σε κάδο απορριμμάτων. Ο μικροελεγκτής θα εκτελεί τον περιέκτη της προώθησης των τιμών μέσω του socket καναλιού ή σε περίπτωση αδυναμίας θα χρησιμοποιεί το λειτουργικό του για να επιτύχει τέτοια επικοινωνία.

Η επέκταση της εφαρμογής σε πολλούς απομακρυσμένους κάδους προϋποθέτει την εκτέλεση πολλών περιεκτών ταυτόχρονα στο ίδιο docker υποδίκτυο. Άρα θα πρέπει να δημιουργηθεί υποδομή ενορχήστρωσης και να προγραμματιστούν υπηρεσίες με εργαλεία Docker Swarm ή Kubernetes που θα διαχειρίζονται και θα κλιμακώνουν επαρκώς τους περιέκτες των περιφερειακών κάδων. Επίσης η χρήση της πλατφόρμας Apptainer θα προσδώσει εύκολη διαχείριση των χρηστών και των δικαιωμάτων τους σε ένα πολυχρηστικό σύστημα περιεκτών παρέχοντας την απαραίτητη ασφάλεια σε περιβάλλοντα υψηλών υπολογιστικών επιδόσεων.

Στον τομέα του υπολογιστικού νέφους μπορούν να δημιουργηθούν εφαρμογές τεχνητής νοημοσύνης που θα υπολογίζουν βέλτιστες διαδρομές για την αποκομιδή των απορριμμάτων από τα απορριμματοφόρα βάσει του ποσοστού πληρότητας. Αντίστοιχα εφαρμογές επιχειρηματικής ευφυίας θα υπαγορεύουν αλλαγές στην χωροταξική διαμόρφωση – τοποθέτηση των κάδων ανάλογα με το ποσοστό χρήσης τους.

Μπορούν να δημιουργηθούν εφαρμογές ενημέρωσης των υπεύθυνων υπηρεσιών όπως ο Δήμος ή η Πυροσβεστική σε περίπτωση υψηλής θερμοκρασίας ή φωτιάς. Είναι εφικτή η δημιουργία εφαρμογών για κινητές συσκευές που θα χρησιμοποιούν οι πολίτες προς ενημέρωση τους για τις συνθήκες λειτουργίας και την πληρότητα των κάδων. Αν ο κάδος είναι πλήρης θα ειδοποιείται ο πολίτης να μην τον χρησιμοποιεί και θα του υποδεικνύονται οι επόμενοι πιο κοντινοί μέσω στίγματος σε χάρτη. Την ίδια πληροφόρηση θα έχει και στην περίπτωση που ο κάδος έχει πάρει φωτιά ή έχει πολύ υψηλή θερμοκρασία.

Το σύνολο των υπηρεσιών – εφαρμογών που θα δημιουργηθεί μπορεί να ενταχθεί σε ένα γενικότερο πλάνο διαχείρισης της έξυπνης πόλης μέσω τεχνολογίας blockchain.

7. ΠΑΡΑΡΤΗΜΑ

7.1 Πίνακες εντολών

Πίνακας 1: Εντολές Docker με περιγραφή

Εντολή	Περιγραφή
sudo /etc/init.d/docker status	
sudo /etc/init.d/docker start	
sudo /etc/init.d/docker stop	
docker exec -it dipl-demo_redis_1 redis-cli	
docker run nginx	Run – start a container
docker ps	ps – list containers
docker ps -a	ps – list containers All
docker stop silly_sammet	STOP – stop a container
docker rm silly_sammet	Rm – Remove a container
docker images	images – List images
docker rmi nginx	rmi – Remove images
docker pull nginx	Pull – download an image
docker run ubuntu sleep 5	Append a command
docker exec distracted_mcclintock cat /etc/hosts	Exec – execute a command
docker run –d kodekloud/simple-webapp	Run – detach
docker attach a043d	Run – attach
docker run redis:4.0	Run – tag
docker run –i kodekloud/simple-prompt-docker	RUN - Input
docker run –it kodekloud/simple-prompt-docker	RUN - Input Terminal
docker run –p 80:5000 kodekloud/simple-webapp	Run – PORT mapping
docker run –p 80:5000 kodekloud/simple-webapp	80:Host 5000:Docker
docker run –v /opt/datadir:/var/lib/mysql mysql	RUN – Volume mapping
docker inspect blissful_hopper	Inspect Container

docker logs blissful_hopper	Container Logs
docker run -e APP_COLOR=green simple-webapp-color	ENV Variables in Docker
docker inspect blissful_hopper	Inspect Environment Variable
Dockerfile	How to create my own image?
FROM Ubuntu	1. OS - Ubuntu
RUN apt-get update	2. Update apt repo
RUN apt-get install python	3. Install dependencies using apt
RUN pip install flask RUN pip install flask-mysql	4. Install Python dependencies using pip
COPY . /opt/source-code	5. Copy source code to /opt folder
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run	6. Run the web server using “flask” command
docker build Dockerfile –t mmumshad/my-custom-app	Create Image
docker push mmumshad/my-custom-app	Docker Registry
Default networks	
docker run ubuntu	Bridge
docker run Ubuntu –-network=none	none
docker run Ubuntu --network=host	host
docker network create \ –-driver bridge \ -–subnet 182.18.0.0/16 custom-isolated-network	User-defined network
docker network ls	Network List
docker inspect blissful_hopper	Inspect Network
File system	
/var/lib/docker	
/aufs	
/containers	
/image	
/volumes	
Docker compose	

docker-compose -f <config-filename> build	Use the command to build containers and the applications that will be running in each container
docker-compose -f <config-filename> up	Let's launch both "test-webapp" and "redis" services, as described in config file "docker-compose.yml", using the command
Εκτέλεση Docker	
docker exec -it node-server /bin/sh	
docker exec -it a8756127bff5 sh	We are using the docker exec -it command that allows us to connect to a running container while the -it option allows us to capture the stdin/stdout of that container. Then we specify the CONTAINER ID a8756127bff5 obtained from the docker ps command above, followed by the shell (sh) that we want to launch as we enter the container

Πίνακας **2:** Εντολές Linux με αποτέλεσμα και περιγραφή

Εντολή	Αποτέλεσμα	Περιγραφή
[~]$ echo Hello	Hello [~]$	Τυπώνει στην οθόνη το argument που γράφουμε μετά το echo
[~]$ type echo	echo is a shell built-in [~]$	Δείχνει αν η εντολή είναι εσωτερική ή εξωτερική
[~]$ type mv	mv is hashed (/bin/mv) [~]$	Δείχνει αν η εντολή είναι εσωτερική ή εξωτερική
[~]$ pwd	/home/michael	pwd (present working directory)
[~]$ ls	Asia Europe Africa America	Ls (List contents)
[~]$ mkdir Asia		mkdir (make a new directory)
[~]$ mkdir Europe Africa America		mkdir (multiple directories)

[~]$ cd Asia	[~/Asia]$	cd (change directory)
[~/Asia]$ cd ..	[~]$	
[~/Asia]$ cd	[~]$	
[~/Asia]$ cd /home/michael	[~]$	
[~]$ mv /home/michael/Europe/Morocco /home/michael/Africa/		mv (Move file or directory) - Absolute path
[~]$ mv Europe/Morocco Africa/		mv (Move file or directory) - Relative path
[~]$ mv Asia/India/Munbai Asia/India/Mumbai		Rename directory
[~]$ cp Asia/India/Mumbai/City.txt Africa/Egypt/Cairo		cp (Copy file)
[~]$ rm Europe/UK/London/Tottenham.txt		rm (Remove file or directory)
[~]$ cp –r Europe/UK Europe/UnitedKingdom		cp -r (Copy directory)
[~]$ cat Asia/India/Mumbai/City.txt	Mumbai	Concatenate (Βλέπουμε τα περιεχόμενα του αρχείου)
[~]$ cat > Africa/Egypt/Cairo/City.txt Cairo		Concatenate (redirect) (Προσθέτουμε περιεχόμενα στο αρχείο) ctrl d Βγαίνουμε από την καταχώριση

[~]$ more new_file.txt		[Space] - scrolls the display, one screenful of data at a time [Enter] - scrolls the display one line [b] - scrolls the display backwards one screenful of data [/] – search text
[~]$ less new_file.txt		[Up Arrow] - scrolls up the display one line [Down Arrow] – scrolls down the display one line [/] – search text
[~]$ ls -l		ls –l (long list)
[~]$ ls -a		ls –a (list all files including hidden)
[~]$ ls -lt		ls –lt (long list files in order created)
[~]$ ls -ltr		ls –ltr (long list files in the reverse order created)
Using Command Line to Get Help		
[~]$ whatis date	date (1) - print or set the system date and time2	Using Command Line to Get Help
[~]$ man date		Using Command Line to Get Help
[~]$ date --help		
[~]$ apropos modpr		Εμφανίζει όλες τις εντολές που έχουν τη λέξη modpr
[~]$ echo $HOME		Μας εμφανίζει το HOME directory
Bash Shell Features		
[~]$ ls Docu + TAB	ls Documents	Bash Auto-Completion using TAB
[~]$ alias dt=date		Alias
[~]$ history		Command History

Bash Environment Variables		
[~]$ echo $SHELL		Εμφανίζει την τιμή της $SHELL
[~]$ env		Εμφανίζει όλες τις Environment Variables
[~]$ export OFFICE=caleston	~/.profile or ~/.pam_environment	Δίνει τιμή στην ev OFFICE και την αποθηκεύει
[~]$ echo $PATH	/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin	Εμφανίζει την τιμή της $PATH
[~]$ which obs-studio	/opt/obs/bin/obs-studio	Εμφανίζει το directory στο οποίο είναι η εντολή obs-studio
Bash Prompt		
[~]$ PS1=“ubuntu-server:”	ubuntu-server:	Αλλάζει το prompt σε "ubuntu-server:"
ubuntu-server: PS1="[\d \t \u@\h:\w] $ "	[Thu Mar 12 22:12:54 bob@caleston:~] $	Αλλάζει το prompt σε Ημέρα Μήνα Ημ/νία Ώρα Χρήστης H/Y
		\d : the date in “Weekday Month Date” format (e.g., “Tue May 26”)
		\e : an ASCII escape character (033)
		\h : the hostname HQDN
		\H : the complete hostname
		\n : newline
		\r : carriage return
		\s : the name of the shell
		\t : the current time in 24-hour HH:MM:SS format
		\T : the current time in 12-hour HH:MM:SS format

		\@ : the current time in 12-hour am/pm format
		\A : the current time in 24-hour HH:MM format
		\u : the username of the current user
		\w : the current working directory, with $HOME abbreviated with a tilde
		\W : the basename of the current working directory, with $HOME abbreviated
		with a tilde
		\$: if the effective UID is 0, a #, otherwise a $
Kernel Versions		
curl --location --request GET 'http://localhost:8084'		
curl --location --request GET 'http://localhost:8085/fire'		

Πίνακας 3: VS Code

Εντολή	Αποτέλεσμα
Debug Node.js Javascript and TypeScript apps with VSCode Like A Pro!	https://www.youtube.com/watch?v=adSEgWVlp7Q
npm init -y	
npm i -D @types/express typescript ts-node	
npm i express	
touch test.js	

node test.js	
curl http://localhost:4000	
node --inspect test.js	
node --inspect-brk test.js	

Πίνακας 4: Node-Docker

Εντολή	Περιγραφή	Αποτέλεσμα
Debug Node.js Javascript and TypeScript apps with VSCode Like A Pro!	https://www.youtube.com/watch?v=adSEgWVlp7Q	Δημιουργία express app
npm init -y	Πρώτα θα δημιουργήσουμε την εφαρμογή με express και μετά θα αναπαράγουμε τα βήματα στο Docker	Δημιουργία αρχείου package.json
npm install express		Εγκατάσταση express
		Δημιουργία αρχείου index.js
npm install nodemon --save-dev		
sudo /etc/init.d/docker status		
explorer.exe .		
sudo /etc/init.d/docker status		
https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-20-04		
sudo/etc/init.d/docker start	Παίρνουμε από το Docker Hub το image που υπάρχει για nodejs και πάνω σε αυτό θα προσθέσουμε τα dependencies και τον κώδικα μας για να	Δημιουργούμε το αρχείο Dockerfile που περιέχει τις παραμέτρους για την δημιουργία του image

	δημιουργήσουμε ένα τελικό docker image δικο μας	
docker build -t node-app-image		
docker run -v $(pwd):/app --name node-app -p 4000:3000 -d node-app-image	Εκτέλεση Docker run	
docker exec -it node-app bash		
docker logs node-app		
docker run -v $(pwd):/app -v /app/node_modules --name node-app -p 4000:3000 -d node-app-image		
docker run -v $(pwd):/app:ro -v /app/node_modules --name node-app -p 4000:3000 -d node-app-image		
touch newfile		
docker run -v $(pwd):/app:ro -v /app/node_modules --name node-app --env PORT=5000 -p 4000:5000 -d node-app-image		
printenv		
docker run -v $(pwd):/app:ro -v /app/node_modules --name node-app --env-file ./.env -p 4000:5000 -d node-app-image		
docker volume ls		
docker volume prune		

docker system prune -f		
docker volume rm node-app		
docker rm node-app -fv		
	Εκτέλεση Docker Compose	
docker-compose up -d		
docker-compose down -v		
docker-compose down -v --build		
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d	Εκτέλεση Docker Compose με διπλό αρχείο configuration docker-compose.yml και docker-compose.dev.yml ή docker-compose.prod.yml	
docker-compose -f docker-compose.yml -f docker-compose.dev.yml down -v		
docker-compose -f docker-compose.yml -f docker-compose.prod.yml up -d --build		
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d --build -V	αν κάνεις αλλαγή στο compose ή npm install something μπορείς να τρέξεις πάλι το up χωρίς να τρέξεις πρώτα το down. Το -V χρειάζεται για να κάνει πάλι τις συνδέσεις με τις νέες εγκαταστάσεις στο volumes	
mongo -u "sanjeev" -p "mypassword"	Εισαγωγή στο container της Mongodb	
db		

use mydb		
show dbs		
db.books.insert({"name": "harry potter"})		
docker exec -it node-docker_mongo_1 mongo -u "sanjeev" -p "mypassword"		
db.books.find()		
	Αν η Mongodb θα φιλοξενείται στο περιβάλλον του Docker μπορούμε να χρησιμοποιούμε το όνομα της από το Docker-compose αλλιώς αν είναι σε άλλο περιβάλλον π.χ. AWS θα πρέπει να περνάμε την IP της σε Environment Variable	Δημιουργούμε το αρχείο config.js στο οποίο δηλώνουμε μεταβλητές - παραμέτρους της mongodb για την IP, Port, UserName, Password στις οποίες δίνουμε τιμές από Enviroment Variables
npm install mongoose		
docker logs node-docker_node-app_1	Όταν σηκώνεται η μηχανή του Docker δεν ξέρουμε ποιό container θα δημιουργήσει πρώτο , το application ή την Mongodb ?	
docker inspect node-docker_mongo_1	Αν σηκωθεί πρώτα το application τότε η εφαρμογή θα κρασάρει.	
docker network ls	Για να σιγουρευτούμε οτι θα σηκωθεί πρώτα η mongodb προσθέτουμε στο docker-compose την ιδιότητα depends_on. Αυτό όμως δεν μας εξασφαλίζει οτι θα έχει ολοκληρωθεί το iniitalize της	

	database. Για να το πετύχουμε δημιουργούμε στο index.js function connectWithRetry	
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d node-app		
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d --no-deps node-app	Φορτώνουμε μόνο το service που μας ενδιαφέρει node-app απόφευγοντας τα depedencies	
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d --no-deps mongo		
npm install bcryptjs	Η bcrypt είναι βιβλιοθήκη που χρησιμοποιούμε για να κάνουμε encrypt το password πριν το αποθηκεύσουμε	
npm install -g npm		
	Χρησιμοποιούμε το Postman για να τεστάρουμε με Requests kai Responces τι δεδομένα βλέπουμε	
{		
"title": "my first post",	Αν δεν δηλώσουμε στο index.js ένα middleware όπως το express τότε δεν λειτουργούν σωστά τα post - get	
"body": "body of first post"		

}		
{		
"title": "my 4th post",		
"body": "body of 4th post"		
}		
	Εισάγουμε το redis container για να υλοποιήσουμε διαδικασία user signon	
npm install redis connect-redis express-session	Εισάγουμε τα express sessions για να υλοποιήσουμε διαδικασία αποθήκευσης και ελέγχου του user status.	
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d -V	Τα express sessions χρησιμοποιούν cookies για να αποθηκεύσουν τα στοιχεία του session	
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d --scale node-app=2		
docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d --build --scale node-app=2	Το express middlware είναι στην ουσία ένα function που τρέχει πριν τον controller και ελέγχει αν υπάρχει στο session authorized user	Λεπτομέρειες για την συμπεριφορά του Express σε συνάρτηση με το Nginx βρίσκουμε στο Express behind proxies
	Cors allows front-end to run in one domain and back-end to run in another domain	
git init		

git add --all		Στο production δεν μπορούμε να έχουμε στο docker-compose.prod.yml το username , password της mongodb γιατί μπορεί καταλάθος να τα ανεβάσουμε στο git
git config --global user.name "itheodorop"		Γιαυτό τα περνάμε σε enviromnt variables του μηχανήματος που τρέχει την εφαρμογή
git commit -m "first commit"		
git branch -M main		
git remote add origin https://github.com/itheodorop/node-docker.git		
git push -u origin main	To https://github.com/itheodorop/node-docker.git	
	* [new branch] main -> main	
export SESSION_SECRET="hello"	Branch 'main' set up to track remote branch 'main' from 'origin'.	
printenv		
printenv SESSION_SECRET		
git add --all		
git commit -m "env changes"		
git push		

git add --all		
mkdir app		
cd app		
git clone https://github.com/itheodorop/node-docker .		
docker-compose -f docker-compose.yml -f docker-compose.prod.yml up -d		
	Docker Swarm	
docker info		
docker swarm init		
docker service --help		
docker stack --help		
docker node ls		
docker stack ls		
docker swarm leave -f	force	
docker exec -it e0c061a5700bfa400f8f24b redis-cli		

Πίνακας 5: Εγκαταστάσεις

Τίτλος	Link - Command
SwarmLab Documentation	

Πηγαίνουμε στο git του swarmlab και βρίσκουμε την παράγραφο Build	https://git.swarmlab.io:3000/docs/Documentation
Σε Linux λειτουργικό δημιουργούμε directory swarmlab	mkdir swarmlab
Αντιγράφουμε την εντολή git clone ... και την εκτελούμε σε CLI Linux μέσα στο directory swarmlab	git clone https://git.swarmlab.io:3000/docs/Documentation.git
Μπαίνουμε στο directory που δημιουργήθηκε Documentation	cd Documentation
Εκτελούμε την εντολή ./build.sh	./build.sh
Μας εμφανίζει μία λίστα με παραμέτρους με πράσινο χρώμα που μπορούμε να εκτελέσουμε μετά την εντολή ./build.sh	./build.sh poc-datacollector
Στην περίπτωση προβλήματος: ERROR: Get https://hub.swarmlab.io:5480/v2/: x509: certificate signed by unknown authority εκτελούμε την εντολή	./0-get-certs.sh
Με την εκτέλεση δημιουργείται ένας web server που μας σερβίρει μία ιστοσελίδα με οδηγίες. Στην οθόνη του CLI ανάμεσα στις εντολές εμφανίζονται και 2 links. Το 1ο link είναι για εκτέλεση από τον host και το 2ο για εκτέλεση από το container.	http://127.0.0.1:8080 http://172.17.0.4:8080
Με Control + left click στο link του CLI μας ανοίγει αυτόματα την σελίδα στον browser	http://172.17.0.4:8080/swarmlab_poc-datacollector/docs/index.html
Ακολουθούμε τις οδηγίες και τα menus - links της ιστοσελίδας	
Atom	

add our official package repository to your system by running the following commands:	wget -qO - https://packagecloud.io/AtomEditor/atom/gpgkey | sudo apt-key add -
	sudo sh -c 'echo "deb [arch=amd64] https://packagecloud.io/AtomEditor/atom/any/ any main" > /etc/apt/sources.list.d/atom.list'
	sudo apt-get update
Install Atom	sudo apt-get install atom
Atom run	atom
Node.JS	https://github.com/nodesource/distributions/blob/master/README.md#rpminstall
Node.js v17.x:	# Using Ubuntu curl -fsSL https://deb.nodesource.com/setup_17.x | sudo -E bash - sudo apt-get install -y nodejs

Πίνακας 6: WSL Εγκαταστάσεις

WSL Installation	
wsl --install	You can now install everything you need to run Windows Subsystem for Linux (WSL) by entering this command in an administrator PowerShell or Windows Command Prompt and then restarting your machine.
Manual installation steps for older versions of WSL	
Step 1 - Enable the Windows Subsystem for Linux	

Open PowerShell as Administrator (Start menu > PowerShell > right-click > Run as Administrator) and enter this command:	
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart	You must first enable the "Windows Subsystem for Linux" optional feature before installing any Linux distributions on Windows.
Step 2 - Check requirements for running WSL 2	
To check your version and build number, select Windows logo key + R, type winver, select OK.	If you are running Windows 10 version 1903 or 1909, open "Settings" from your Windows menu, navigate to "Update & Security" and select "Check for Updates". Your Build number must be 18362.1049+ or 18363.1049+, with the minor build # over .1049.
Step 3 - Enable Virtual Machine feature	
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart	Restart your machine to complete the WSL install and update to WSL 2.
Step 4 - Download the Linux kernel update package	
Download the latest package:	WSL2 Linux kernel update package for x64 machines
Run the update package downloaded in the previous step. (Double-click to run - you will be prompted for elevated permissions, select ‘yes’ to approve this installation.)	
Step 5 - Set WSL 2 as your default version	wsl --set-default-version 2
Step 6 - Install your Linux distribution of choice	
Open the Microsoft Store and select your favorite Linux distribution.	

Invoke-WebRequest -Uri https://aka.ms/wslubuntu2004 -OutFile Ubuntu.appx -UseBasicParsing	If you prefer, you can also download your preferred distribution(s) via the command line, you can use PowerShell with the Invoke-WebRequest cmdlet. For example, to download Ubuntu 20.04:
curl.exe -L -o ubuntu-2004.appx https://aka.ms/wslubuntu2004	You also have the option to use the curl command-line utility for downloading. To download Ubuntu 20.04 with curl:
Add-AppxPackage .\app_name.appx	Once the distribution has been downloaded, navigate to the folder containing the download and run the following command in that directory, where app-name is the name of the Linux distribution .appx file.
	Once the Appx package has finished downloading, you can start running the new distribution by double-clicking the appx file. (The command wsl -l will not show that the distribution is installed until this step is complete).

Πίνακας 7: WSL Διαχείριση

If you forgot the password for your Linux distribution:	
Open PowerShell and enter the root of your default WSL distribution using the command: wsl -u root	
If you need to update the forgotten password on a distribution that is not your default, use the command: wsl -d Debian -u root, replacing Debian with the name of your targeted distribution.	

Once your WSL distribution has been opened at the root level inside PowerShell, you can use this command to update your password: passwd <username> where <username> is the username of the account in the distribution whose password you've forgotten.	
You will be prompted to enter a new UNIX password and then confirm that password. Once you're told that the password has updated successfully, close WSL inside of PowerShell using the command: exit.	
Terminate a Linux distro on WSL	
To terminate a Linux distro on WSL running on Windows 11 or Windows 10, use these steps:	
Open Start.	
Search for Command Prompt (or PowerShell), right-click the top result and select the Run as administrator option.	
Type the following command to view all running WSL distros and press Enter:	
wsl --list --verbose	
Type the following command to shut down the Linux distro on Windows 11 or Windows 10 and press Enter:	
wsl -t DISTRO-NAME	

(Optional) Type the following command to confirm the distro is no longer running and press Enter:	
wsl --list --verbose	
Terminate all Linux distros on WSL	
To shut down all the WSL distros running on Windows 10 (or 11), use these steps:	
Open Start.	
Search for Command Prompt (or PowerShell), right-click the top result and select the Run as administrator option.	
Type the following command to view all running WSL distros and press Enter:	
wsl --list --verbose	
Type the following command to shut down the Linux distributions on Windows 11 or 10 and press Enter:	
wsl --shutdown	
Set up your Linux username and password	
Install Node.js on Windows Subsystem for Linux (WSL2)	

Install nvm, node.js, and npm	
Install cURL (a tool used for downloading content from the internet in the command-line) with:	sudo apt-get install curl
Install nvm, with:	curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
To verify installation, enter: command -v nvm ...this should return 'nvm', if you receive 'command not found' or no response at all, close your current terminal, reopen it, and try again.	command -v nvm
List which versions of Node are currently installed (should be none at this point):	nvm ls
$ ip route	
default via 172.24.64.1 dev eth0	
172.24.64.0/20 dev eth0 proto kernel scope link src 172.24.66.230	
$ cat /etc/resolv.conf	
# This file was automatically generated by WSL. To stop automatic generation of this file, add the following entry to /etc/wsl.conf:	
# [network]	
# generateResolvConf = false	
nameserver 172.24.64.1	
$ dig +noall +answer microsoft.com	
microsoft.com. 0 IN A 13.77.161.179	
microsoft.com. 0 IN A 40.76.4.15	
microsoft.com. 0 IN A 40.112.72.205	
microsoft.com. 0 IN A 40.113.200.201	
microsoft.com. 0 IN A 104.215.148.63	

$ curl -4sv -m5 https://microsoft.com/	
* Trying 13.77.161.179...	
* TCP_NODELAY set	
* After 2498ms connect time, move on!	
* connect to 13.77.161.179 port 443 failed: Connection timed out	
* Trying 40.76.4.15...	
* TCP_NODELAY set	
* After 1249ms connect time, move on!	
* connect to 40.76.4.15 port 443 failed: Connection timed out	
* Trying 40.112.72.205...	
* TCP_NODELAY set	
* After 623ms connect time, move on!	
* connect to 40.112.72.205 port 443 failed: Connection timed out	
* Trying 40.113.200.201...	
* TCP_NODELAY set	
* After 311ms connect time, move on!	
* connect to 40.113.200.201 port 443 failed: Connection timed out	
* Trying 104.215.148.63...	
* TCP_NODELAY set	
* After 155ms connect time, move on!	
* connect to 104.215.148.63 port 443 failed: Connection timed out	
* Failed to connect to microsoft.com port 443: Connection timed out	
* Closing connection 0	

Πίνακας 8: MongoDB

Install MongoDB on WSL	
To install MongoDB (version 5.0) on WSL (Ubuntu 20.04):	
Open your WSL terminal (ie. Ubuntu) and go to your home directory:	cd ~
Update your Ubuntu packages:	sudo apt update
Import the public key used by the MongoDB package management system:	wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add -
Create a list file for MongoDB:	echo "deb [arch=amd64,arm64] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/5.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list
Reload local package database:	sudo apt-get update
Install MongoDB packages:	sudo apt-get install -y mongodb-org
Confirm installation and get the version number:	mongod --version
Make a directory to store data:	mkdir -p ~/data/db
Run a Mongo instance:	sudo mongod --dbpath ~/data/db
Check to see that your MongoDB instance is running with:	ps -e | grep 'mongod'
To exit the MongoDB Shell, use the shortcut keys:	Ctrl + C
Tip	
Installing MongoDB may require slightly different steps depending on the Linux distribution being used	

for installation. See the MongoDB installation tutorials. Also note that MongoDB installation may differ depending on the version # that you are aiming to install. Use the version drop-down list in the top-left corner of the MongoDB documentation to select the version that aligns with your goal.	
MongoDB init system differences	
In the example above we ran MongoDB directly. Other tutorials may start MongoDB using the operating system's built-in init system. You might see the command sudo systemctl status mongodb used in tutorials or articles. Currently WSL does not have support for systemd (a service management system in Linux).	
You shouldn't notice a difference, but if a tutorial recommends using sudo systemctl, instead use: sudo /etc/init.d/. For example, sudo systemctl status docker, for WSL would be sudo /etc/init.d/docker status ...or you can also use sudo service docker status.	sudo systemctl
Παράδειγμα :	sudo systemctl status docker
Add the init script to start MongoDB as a service	
The installation instructions above install a version of MongoDB that doesn't include a script automatically in /etc/init.d/. If you would like to use the service commands, you can download the init.d script for mongodb from this source, place that manually as a file at this path: /etc/init.d/mongodb and then you can start Mongo as a service using sudo service mongodb start.	

Download the init.d script for MongoDB:	curl https://raw.githubusercontent.com/mongodb/mongo/master/debian/init.d | sudo tee /etc/init.d/mongodb >/dev/null
Assign that script executable permissions:	sudo chmod +x /etc/init.d/mongodb
Now you can use MongoDB service commands:	
sudo service mongodb status for checking the status of your database. You should see a [Fail] response if no database is running.	sudo service mongodb status
sudo service mongodb start to start running your database. You should see a [Ok] response.	sudo service mongodb start
sudo service mongodb stop to stop running your database.	sudo service mongodb stop
Verify that you are connected to the database server with the diagnostic command: mongo --eval 'db.runCommand({ connectionStatus: 1 })' This will output the current database version, the server address and port, and the output of the status command. A value of 1 for the "ok" field in the response indicates that the server is working.	mongo --eval 'db.runCommand({ connectionStatus: 1 })'
Note	
MongoDB has several default parameters, including storing data in /data/db and running on port 27017. Also, mongod is the daemon (host process for the database) and mongo is the command-line shell that connects to a specific instance of mongod.	

VS Code supports working with MongoDB databases via the Azure CosmosDB extension, you can create, manage and query MongoDB databases from within VS Code. To learn more, visit the VS Code docs: Working with MongoDB.	

Πίνακας 9: Redis

Τίτλος	Link - Command	Αποτέλεσμα
How to install redis server on ubuntu?		
Automatic installation		
Update Ubuntu Package List	sudo apt-get update	
Upgrade	sudo apt-get upgrade	
Install Redis Server	sudo apt-get install redis-server	
Backup configuration file	sudo cp /etc/redis/redis.conf /etc/redis/redis.conf.default	
Manual Installation		
# update apt package	sudo apt-get update	sudo apt-get update
	sudo apt-get install build-essential tcl	sudo apt-get install build-essential tcl
# change dir to /tmp	cd /tmp	cd /tmp
# download redis stable tar.gz	curl -O https://download.redis.io/releases/redis-stable.tar.gz	curl -O https://download.redis.io/releases/redis-2.8.1.tar.gz
# unpack redis package	tar xzvf redis-stable.tar.gz	tar xzvf redis-2.8.1.tar.gz
# build and install	cd redis-stable	cd redis-2.8.1
	make	make

	make test	make test
	sudo make install	sudo make install
Run CLI for Redis	redis-cli	redis-cli
Test Connection	ping	ping
Test Echo	ECHO "Hello World"	ECHO "Hello World"
Close Connection	Quit	Quit
How to uninstall redis server on ubuntu?		
Redis Uninstall		
If you have install redis server and want to know how to remove redis server completely from your ubuntu server try following commands one by one:		
# if you use apt-get to install redis then use	sudo apt-get purge --auto-remove redis-server	
# if you compiled redis manually then follow the steps below to remove it completely from linux/ubuntu	sudo service redis_version stop	
# Now delete everything related to Redis server from /usr/local/bin/	sudo rm /usr/local/bin/redis-*	
# Now delete Redis Configuration files directory and it’s content.	sudo rm -r /etc/redis/	
# Delete existing Redis log files.	sudo rm /var/log/redis_*	
# Delete existing Redis data directory and it’s content.	sudo rm -r /var/lib/redis/	
# Delete existing Redis server init scripts	sudo rm /etc/init.d/redis_*	

# Remove existing Redis PID files (Only if exists)	sudo rm /var/run/redis_*	
How to configure a redis server?		
# create a redis directory in /etc	$ sudo mkdir /etc/redis	$ sudo mkdir /etc/redis
# copy configuration files	$ sudo cp /tmp/redis-stable/redis.conf /etc/redis	$ sudo cp /tmp/redis-2.8.1/redis.conf /etc/redis
# open and edit redis.conf	$ sudo nano /etc/redis/redis.conf	$ sudo nano /etc/redis/redis.conf
# search for "supervised" in /etc/redis/redis.conf and replace that text with following	supervised systemd	supervised systemd
# search for "dir" in /etc/redis/redis.conf and replace that text with following	dir /var/lib/redis	dir /var/lib/redis
how to keep redis server running?		
# create a redis service file	$ sudo nano /etc/systemd/system/redis.service	
# paste following contents to open file and save the file once contents are pasted	[Unit] Description=Redis Server After=network.target [Service] User=redis Group=redis ExecStart=/usr/local/bin/redis-server /etc/redis/redis.conf ExecStop=/usr/local/bin/redis-cli shutdown Restart=always [Install] WantedBy=multi-user.target	

How to start/stop redis service?		
# to start redis service	$ sudo service redis start	
# to stop redis service	$ sudo service redis stop	
# to check redis service status	$ sudo service redis status	
Enable Redis to Start at Boot		
Once you have created a redis service you might want to run following command so that even if your ubuntu server stops for any reason and restarted you will have redis server running.	sudo systemctl enable redis	
Create the Redis User, Group and Directories		
		# create redis user and group
		$ sudo adduser --system --group --no-create-home redis
		# make redis home dir
		$ sudo mkdir /var/lib/redis
		# update file permissions
		$ sudo chown redis:redis /var/lib/redis
		# prevent regular users
		$ sudo chmod 770 /var/lib/redis

How to test if redis server is installed correctly?		# switch to redis cli mode
		$ redis-cli
		# send ping request
		127.0.0.1:6379> ping
		# you will see following output
		PONG
		# exit the server
		127.0.0.1:6379> exit
How to test using script if redis server is running ?	redis.sh	#!/bin/sh set -eo pipefail host="$(hostname -i \|\| echo '127.0.0.1')" if ping="$(redis-cli -h "$host" ping)" && ["$ping" = 'PONG']; then exit 0 fi exit 1
		#!/bin/sh :Executes the script using the Bourne shell or a compatible shell, with path /bin/sh set -eo pipefail : The set -e option instructs bash to immediately exit if any command [1] has a non-

		zero exit status. The -o pipefail setting prevents errors in a pipeline from being masked. If any command in a pipeline fails, that return code will be used as the return code of the whole pipeline. By default, the pipeline's return code is that of the last command even if it succeeds. $(hostname -i || echo '127.0.0.1'): Επιστρέφει την IP του host ή το 127.0.0.1 if ping= : Αν εκτελείται το redis κανονικά (στο ping απαντάει με PONG) τότε θα μας επιστρέψει την τιμή 0 αλλιώς την τιμή 1
Commands		
Ορισμός πεδίου - τιμής	SET foo 100	OK
	GET foo	"100"
	SET bar "Hello World"	OK
	GET bar	"Hello World"
Increment foo	INCR foo	101
Decrement foo	DECR foo	100
Ύπαρξη πεδίου	EXISTS foo	1
	EXISTS foo1	0

Σβήσιμο πεδίου	DEL bar	
	GET bar	nil
Σβήσιμο Όλων	FLUSHALL	
	SET server:name someserver	
	SET server:port 8000	
	SET greeting "Hello World"	
Λήξη σε 50 δευτερόλεπτα του Key - value	EXPIRE greeting 50	
Δείχνει τον χρόνο που απομένει	TTL greeting	
	GET greeting	nil
Ταυτόχρονος ορισμός key - value και expiration	SETEX greeting 30 "Hello World"	
Ακύρωση του timer και διατήρηση του key - value	PERSIST greeting	
Ορισμός πολλών κλειδιών με μία εντολή	MSET key1 "Hello" key2 "World"	
Προσθήκη στην τιμή ενός κλειδιού	APPEND key1 " World"	
Αλλαγή ονόματος κλειδιού	RENAME key1 greeting	
Προσθήκη τιμών σε Λίστα από αριστερά	LPUSH people "Brad"	(integer) 1
Προσθήκη τιμών σε Λίστα από αριστερά	LPUSH people "Jen"	(integer) 2
Προσθήκη τιμών σε Λίστα από αριστερά	LPUSH people "Tom"	(integer) 3
Εμφάνιση τιμών λίστας από τη 1η θέση αριστερά (0) έως το τέλος (-1)	LRANGE people 0 -1	1) "Tom" 2) "Jen" 3) "Brad"

Προσθήκη τιμών σε Λίστα από δεξιά	RPUSH people Harry	(integer) 4
Εμφάνιση τιμών λίστας από τη 1η θέση αριστερά (0) έως το τέλος (-1)	LRANGE people 0 -1	1) "Tom" 2) "Jen" 3) "Brad" 4) "Harry"
Πλήθος κλειδιών Λίστας	LLEN people	(integer) 4
Αφαίρεση τιμής από αριστερά της Λίστας	LPOP people	"Tom"
Αφαίρεση τιμής από δεξιά της Λίστας	RPOP people	"Harry"
Προσθήκη τιμής στη μέση της Λίστας πριν την τιμή "Brad"	LINSERT people BEFORE "Brad" "Tom"	
Προσθήκη τιμής σε set	SADD cars "Ford"	(integer) 1
Προσθήκη τιμής σε set	SADD cars "Honda"	(integer) 1
Προσθήκη τιμής σε set	SADD cars "BMW"	(integer) 1
Εύρεση τιμής σε set (Υπάρχει)	SISMEMBER cars "Ford"	(integer) 1
Εύρεση τιμής σε set (Δεν Υπάρχει)	SISMEMBER cars "Chevy"	(integer) 0
Εμφάνιση όλων των members του set	SMEMBERS cars	1) "Honda" 2) "Ford" 3) "BMW"
Πλήθος στοιχείων set	SCARD cars	(integer) 3
Μετακίνηση ενός στοιχείου σε άλλο set (mycars)	SMOVE cars mycars "Ford"	(integer) 1
Διαγραφή στοιχείου "BMW"	SREM cars "BMW"	(integer) 1
Προσθήκη τιμής και στοιχείου ταξινόμησης σε ταξινομημένη λίστα	ZADD users 1981 "Brad Traversy"	(integer) 1

Προσθήκη τιμής και στοιχείου ταξινόμησης σε ταξινομημένη λίστα	ZADD users 1975 "John Doe"	(integer) 1
Προσθήκη τιμής και στοιχείου ταξινόμησης σε ταξινομημένη λίστα	ZADD users 1990 "Mike Smith"	(integer) 1
Εύρεση της θέσης του "Mike Smith" στη λίστα	ZRANK users "Mike Smith"	(integer) 3
Εύρεση της θέσης του "John Doe" στη λίστα	ZRANK users "John Doe"	(integer) 0
Εμφάνιση όλων των members του sorted set	ZRANGE users 0 -1	1) "John Doe"
		2) "Brad Traversy"
		3) "Kate Rogers"
		4) "Mike Smith"
Αύξηση της τιμής του κλειδιού ταξινόμησης κατά 10	INCRBY users 10 "John Doe"	1985
Ορισμός κλειδιού και τιμής σε hash	HSET user:brad name "Brad Traversy"	(integer) 1
Ορισμός κλειδιού και τιμής σε hash	HSET user:brad email "brad@gmail.com"	(integer) 1
Ανάκτηση τιμής συγκεκριμένου κλειδιού σε hash	HGET user:brad name	Brad Traversy
Ανάκτηση όλων των κλειδιών και των τιμών τους για το hash user:brad	HGETALL user:brad	1) "name" 2) "Brad Traversy" 3) "email" 4) "brad@gmail.com"
Ορισμός πολλαπλών κλειδιών και τιμών σε hash	HMSET user:john name "John Doe" email "jdoe@yahoo.com" age "25"	OK

Ανάκτηση όλων των κλειδιών και των τιμών τους για το hash user:john	HMGETALL user:john	1) "name" 2) "John Doe" 3) "email" 4) "jdoe@yahoo.com" 5) "age" 6) "25"
Ανάκτηση όλων των κλειδιών για το hash user:john	HKEYS user:john	1) "name" 2) "email" 3) "age"
Ανάκτηση όλων των τιμών για το hash user:john	HVALS user:john	1) "John Doe" 2) "jdoe@yahoo.com" 3) "25"
Αύξηση τιμής ιδιότητας age κατά 1	HINCRBY user:john age 1	(integer) 26
Διαγραφή ιδιότητας age	HDEL user:john age	(integer) 1
Ανάκτηση όλων των κλειδιών και των τιμών τους για το hash user:john	HGETALL user:john	1) "name" 2) "John Doe" 3) "email" 4) "jdoe@yahoo.com"
Εμφάνιση του πλήθους των εγγραφών του hash	HLEN user:john	(integer) 2
redis - nodejs application redusers	https://github.com/bradtraversy/redusers	
	mkdir redusers	
	cd redusers	
	npm init	package name: (redusers) version: (1.0.0) description: Simple user managment app entry point: (index.js)

		app.js test command: git repository: keywords: author: Stelios Karagiannakis license: (ISC)
	npm install express body-parser redis method-override express-handlebars --save	
Δημιουργία αρχείου app.js	File -> New File -> Save As	
Node Redis: Examples	https://github.com/redis/node-redis/tree/master/examples	
Setup	$ git clone https://github.com/redis/node-redis.git	
	$ cd node-redis	
	$ npm install -ws && npm run build	
	$ cd examples	
	$ npm install	

7.2 ΠΗΓΕΣ - ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Waste Management https://www.sciencedirect.com/topics/earth-and-planetary-sciences/waste-management

[2] Internet of things https://en.wikipedia.org/wiki/Internet_of_things

[3] Trend Watch: Internet of Things (IoT) https://www.techtarget.com/trend-watch-internet-of-things-iot/

[4] What is IoT**?** https://www.sap.com/insights/what-is-iot-internet-of-things.html

[5] Internet of things https://mynewtechnologies.com/internet-of-things

[6] Cloud computing https://en.wikipedia.org/wiki/Cloud_computing

[7] Cloud Computing Architecture https://eduinpro.com/blog/cloud-computing-architecture

[8] Containerization https://www.ibm.com/au-en/cloud/learn/containerization

[9] Containers at AWS https://aws.amazon.com/containers/

[10] Docker containers Sample Clauses https://www.lawinsider.com/clause/docker-containers

[11] Linux: An Open Source Operating System https://www.systranbox.com/linux-an-open-source-operating-system/

[12] Node.js® https://nodejs.org/en/

[13] What is Express.js https://www.codecademy.com/article/what-is-express-js

[14] What is web socket and how it is different from the HTTP https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/

[15] Docker (software) https://en.wikipedia.org/wiki/Docker_(software)

[16] Swarm mode overview https://docs.docker.com/engine/swarm/

[17] What is Kubernetes https://cloud.google.com/learn/what-is-kubernetes

[18] Nomad https://github.com/hashicorp/nomad

[19] What is MongoDB https://www.mongodb.com/why-use-mongodb

[20] What is Postman https://www.postman.com/product/what-is-postman/

[21] What is NGINX https://www.nginx.com/resources/glossary/nginx/

[22] What is Vue https://vuejs.org/guide/introduction.html

[23] Apptainer http://apptainer.org/

[24] Express: Routes and controllers https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes

[25] From Intelligent to Smart Cities, Mark Deakin - Husam Al Waer, 2012

[26] Cloud cost-optimization simulator https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/cloud-cost-optimization-simulator

7.3 ΠΗΓΕΣ ΚΩΔΙΚΑ

[27] http://docs.swarmlab.io/SwarmLab-HowTos/courses/main.adoc.html

[28] https://git.swarmlab.io:3000/docs/Documentation

[29] https://kodekloud.com/courses/docker-for-the-absolute-beginner/

[30] https://kodekloud.com/courses/docker-swarm-services-stacks-hands-on/

[31] https://kodekloud.com/learning-path-linux/

[32] https://kodekloud.com/courses/kubernetes-for-the-absolute-beginners-hands-on/

[33] https://medium.com/@marcel_katz/node-js-server-and-redis-instance-deployed-in-docker-containers-communicating-with-each-other-1b6ab32a511

[34] https://pureinfotech.com/uninstall-wsl2-windows-10/

[35] https://www.digitalocean.com/community/tutorials/how-to-build-a-node-js-application-with-docker

[36] https://www.digitalocean.com/community/tutorials/how-to-install-mongodb-on-ubuntu-20-04

[37] https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04

[38] https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-20-04

[39] https://www.digitalocean.com/community/tutorials/containerizing-a-node-js-application-for-development-with-docker-compose

[40] http://www.philipermish.com/blog/docker-example-with-nginx-node-redis-mongodb-and-jekyll/

[41] https://learn.microsoft.com/en-us/windows/wsl/tutorials/wsl-vscode

[42] https://raft.github.io/

[43] https://nodejs.org/en/docs/guides/nodejs-docker-webapp/

[44] https://iabhishek.dev/post/building-a-multimedia-chat-app-using-expres-socketio-redis-and-docker-part-1

[45] https://www.arbazsiddiqui.me/debugging-nodejs-inside-docker/

[46] https://medium.com/geekculture/nodejs-console-logs-in-docker-containers-hidden-no-more-d04bcfe1dc5c

[47] https://blog.risingstack.com/how-to-debug-a-node-js-app-in-a-docker-container/

[48] https://dev.to/sw360cab/scaling-websockets-in-the-cloud-part-1-from-socket-io-and-redis-to-a-distributed-architecture-with-docker-and-kubernetes-17n3

[49] https://vuejs.org/guide/introduction.html#api-styles

[50] https://docs.microsoft.com/en-us/windows/wsl/setup/environment#set-up-your-linux-username-and-password

[51] https://docs.microsoft.com/en-us/windows/wsl/basic-commands#set-wsl-version-to-1-or-2

[52] https://www.youtube.com/watch?v=ZKEqqIO7n-k(1)

Learn Socket.io In 30 Minutes – YouTube

[53] https://www.youtube.com/watch?v=t4eFjyJWmqQ(1)

WSL and VS Code: code on Linux on a Windows laptop 💻 – YouTube

[54] https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/development_environment

[55] https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction#introducing_express

[56] https://code.visualstudio.com/docs/containers/quickstart-node

[57] https://github.com/devat-youtuber/docker-nodejs-express-mongodb-react-redis-nginx

[58] https://www.digitalocean.com/community/tutorials/how-to-secure-a-containerized-node-js-application-with-nginx-let-s-encrypt-and-docker-compose?fbclid=IwAR1gU2v8wkZkIqHSQLK5S5-NXbqsomJyY1JuX-8Oz51mcd2xba4Y0YVIxj4

[59] https://www.digitalocean.com/community/tutorials/how-to-integrate-mongodb-with-your-node-application

[60] https://github.com/Rammohan-bitzop/login-app/blob/master/app.js

[61] https://www.youtube.com/watch?v=gm_L69NHuHM

Docker + Node.js/express tutorial: Building dev/prod workflow with docker and Node.js – YouTube

[62] https://github.com/socketio/socket.io-redis-adapter

[63] https://ixorasolution.com/blog/design-a-mern-stack-application-with-multiple-docker-containers

[64] https://github.com/bezkoder/vuetify-data-table-example

[65] https://www.bezkoder.com/vuetify-data-table-example/

[66] https://www.bezkoder.com/node-express-mongodb-crud-rest-api/

[67] https://www.freecodecamp.org/news/how-to-get-a-docker-container-ip-address-explained-with-examples/

[68] https://www.youtube.com/watch?v=qGwaOeBM-X0&t=138s

(1) Connecting Redis and Mongo through Node – YouTube

[69] https://stackoverflow.com/questions/16877968/call-a-server-side-method-on-a-resource-in-a-restful-way

7.4 ΠΗΓΕΣ ΕΙΚΟΝΩΝ

[70] Παρουσίαση - Εικονικοποίηση - Περιεκτοποίηση Σελίδα 10: https://www.freecodecamp.org/news/docker-vs-vm-key-differences-you-should-know

[71] Παρουσίαση –Μικροϋπηρεσίες Σελίδα 11: https://www.mend.io/resources/blog/microservices-architecture/

[72] Παρουσίαση -Μικροϋπηρεσίες Σελίδα 11: https://middleware.io/blog/microservices-architecture-docker/

[73] Παρουσίαση – Socket.io Σελίδα 24: https://www.fusioncharts.com/blog/visualize-real-time-data-socket-io-charts/

[74] Παρουσίαση – Ευχαριστούμε Σελίδα 30: https://www.esferize.com/en/keys-to-understanding-the-importance-of-smart-cities/