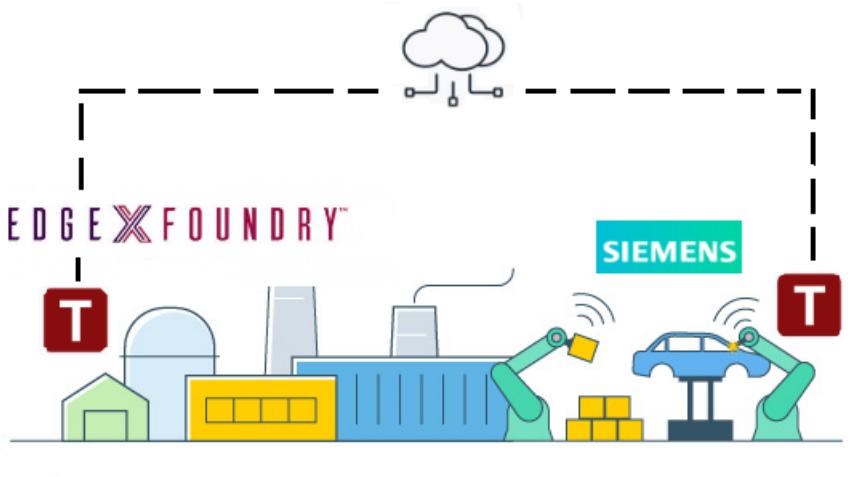
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Μετεγκατάσταση και Λειτουργία Μικροϋπηρεσιών σε Ετερογενείς Πλατφόρμες του Βιομηχανικού Διαδικτύου των Πραγμάτων»



Της φοιτήτριας
Μαρίνας Πατράλη
Αρ. Μητρώου: 164728

Επιβλέπων
Δρ. Περικλής Χατζημίσιος
Καθηγητής

24 Μαΐου 2023

Μετεγκατάσταση και Λειτουργία Μικροϋπηρεσιών σε Ετερογενείς Πλατφόρμες του Βιομηχανικού Διαδικτύου των Πραγμάτων
Κωδικός Δ.Ε. 23106
Μαρίνα Πατράλη
Περικλής Χατζημίσιος
Ημερομηνία ανάληψης Δ.Ε. 08/02/2023
Ημερομηνία περάτωσης Δ.Ε. 24/05/2023

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία της φοιτήτριας Μαρίνας Πατράλη που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Σε όλους όσους είναι πάντα δίπλα μου»

Πρόλογος

Η ιδέα της εκπόνησης του συγκεκριμένου θέματος ως διπλωματική εργασία δημιουργήθηκε κατά τη διάρκεια της πρακτικής μου άσκησης μέσω Erasmus στο Πανεπιστήμιο της Μπολόνια. Κατά τη διάρκεια συνεργασίας μου με την ερευνητική ομάδα του Πανεπιστημίου μου δόθηκε η δυνατότητα να γνωρίσω όλες τις τεχνολογίες που χρησιμοποιήθηκαν στην εργασία αλλά και όλο το πλαίσιο γύρω από το Βιομηχανικό Διαδίκτυο των Πραγμάτων και τις εφαρμογές του. Είχα την τιμή να συμμετάσχω στην ερευνητική ομάδα του Πανεπιστήμιου καθώς και ως συγγραφέας στην δημοσίευση του άρθρου με τίτλο ”Siemens and EdgeX IIoT Platforms: A Functional and Performance Evaluation” στο διεθνές συνέδριο IEEE International Conference on Communication (ICC 2023). Το θέμα αυτό παρουσιάζει ιδιαίτερο ενδιαφέρον όχι μόνο γιατί χρησιμοποιούνται εργαλεία και τεχνολογίες που έχουν υιοθετηθεί από τις εφαρμογές στη σημερινή βιομηχανία αλλά κυρίως γιατί ενισχύει την δυνατότητα της ευελιξίας και επεκτασιμότητας καθώς αποτελεί λύση σε προβλήματα που μπορούν να προκύψουν.

Περίληψη

Το Διαδίκτυο των Πραγμάτων (Internet of Things - IoT) είναι μια έννοια η οποία εντάσσεται ολοένα και περισσότερο σε όλους τους τομείς που σχετίζονται με την ζωή και την καθημερινότητα των ανθρώπων. Ένας από τους τομείς εφαρμογής που παρουσιάζει ιδιαίτερο ενδιαφέρον είναι αυτός της βιομηχανίας. Ειδικότερα με την εισαγωγή των αισθητήρων, των τεχνολογιών όπως την υπολογιστική στα άκρα του δικτύου, των έξυπνων μηχανών η παρακολούθηση των συστημάτων και των μηχανών πραγματοποιείται με μεγαλύτερη ακρίβεια και χρήσιμα συμπεράσματα. Έτσι λοιπόν, με την εφαρμογή του βιομηχανικού διαδικτύου των πραγμάτων εκτός από την αύξηση της παραγωγικότητας παρατηρείται ακόμα τόσο η εξέλιξη σε τεχνολογικό επίπεδο των υπηρεσιών και λειτουργιών αλλά και η βελτίωση της ποιότητας τους ενώ ταυτόχρονα δημιουργεί το έδαφος για μια διαφορετική πραγματικότητα για τον κόσμο της βιομηχανίας.

Παράλληλα η αρχιτεκτονική των μικροϋπηρεσιών αποτελεί πλέον επίσης ένα ιδιαίτερα σημαντικό κομμάτι όσον αφορά σε εφαρμογές οι οποίες σχετίζονται με το περιβάλλον της βιομηχανίας. Με τον τρόπο αυτό, πλέον κάθε εργασία μπορεί να πραγματοποιείται χρησιμοποιώντας μια μοναδική υπηρεσία, κάτι που δίνει ένα πλεονέκτημα στη βιομηχανία που αφορά όχι μόνο την εξοικονόμηση χρόνου και ενέργειας αλλά και την βελτίωση της επεκτασιμότητας και των ανεξάρτητων εγκαταστάσεων. Λαμβάνοντας υπ' όψιν τα παραπάνω, η δυνατότητα για ευελιξία στον τομέα της τοποθέτησης και ανατοποθέτησης των υπηρεσιών αυτών αποτελεί μια πρόκληση. Στα πλαίσια που περιγράφηκαν περιληπτικά, η παρούσα διπλωματική εργασία επικεντρώνεται κυρίως στην ταξινόμηση, ανάπτυξη και ανατοποθέτηση των υπηρεσιών με τον πιο αποτελεσματικό τρόπο σε διάφορα κατανεμημένα συστήματα και πλατφόρμες του περιβάλλοντος της βιομηχανίας.

«Migration and Operation of Microservices in Heterogeneous Platforms for the Industrial Internet of Things»

«Marina Patrali»

Abstract

The Internet of Things (IoT) is a concept that is increasingly integrated into all areas related to people's lives and everyday life. One of the very interesting application areas is industry. More specifically, with the introduction of sensors, technologies such as Edge Computing, and smart machines the monitoring of systems and machines is carried out with greater precision and useful conclusions. Thus, with the implementation of the Industrial Internet of Things (IIoT), in addition to the increase in productivity, there is still both the evolution at the technological level of services and functions and the improvement of their quality, while at the same time, it is generated a different reality for the industrial world.

At the same time, the architecture of microservices is getting more and more involved into applications connected to the industrial environment. In this way, each task can now be implemented by using a dedicated service, which benefits the industry not only in terms of time and energy consumption but also in the improvement of scalability and in independent eployments. Taking this into account, the flexibility of positioning and replacing these services becomes a challenging task. In this context, the current thesis is mainly focused on the placement, development, and migration of services by the most efficient way within different distributed environments and platforms in the industrial environment.

Ευχαριστίες

Θα ήθελα μέσα από την καρδιά μου να ευχαριστήσω όλους όσους συνέβαλλαν στην πορεία και υλοποίηση της διπλωματικής μου εργασίας. Πιο συγκεκριμένα θα ήθελα να ευχαριστήσω τον επιβλέποντα μου Καθηγητή κ. Περικλή Χατζημίσιο ο οποίος καθ' όλη τη διάρκεια της εργασίας συμμετείχε με παρατηρήσεις, διορθώσεις και χρήσιμες συμβουλές οι οποίες οδηγούσαν κάθε φορά στην ολοένα και μεγαλύτερη βελτίωση της εργασίας. Επιπλέον ένα μεγάλο ευχαριστώ οφείλω στο Πανεπιστήμιο της Μπολόνια και στους ανθρώπους που βρίσκονται στην ερευνητική ομάδα του Τμήματος "Computer and Information Science" καθώς με βοήθησαν τόσο με ενίσχυση της ιδέας του θέματος όσο και με την παροχή χρήσιμων εργαλείων και πρόσβασης σε εργαλεία τα οποία ήταν απαραίτητα για την υλοποίηση της εργασίας. Επίσης ένα μεγάλο ευχαριστώ συγκεκριμένα στον καθηγητή Luca Foscini του Πανεπιστημίου της Μπολόνια για όλες τις συμβουλές και ιδέες που μου παρείχε αλλά και στον διδάκτορα και ερευνητή Michele Solimando ο οποίος στήριζε επίσης την υλοποίηση της εργασίας μέχρι και την ολοκλήρωση της υλοποίησης της με συνεχείς βελτιώσεις και προτροπές που αφορούσαν στην εξέλιξη της εργασίας. Τέλος οφείλω και ένα μεγάλο ευχαριστώ στην οικογένεια μου και τους φίλους μου που με στήριξαν καθ' όλη την διάρκεια της εργασίας.

Περιεχόμενα

Κατάλογος Εικόνων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Γενικά

Ο κλάδος της βιομηχανίας αποτελεί ένα σημαντικό κομμάτι όσον αφορά στην παραγωγή και συνεπώς στην οικονομία και την ποιότητα ζωής των ανθρώπων. Είναι ένας κλάδος που έχει περάσει από διάφορα στάδια εξέλιξης προκαλώντας μεγάλες αλλαγές τόσο στις θέσεις εργασίας όσο και στον έλεγχο που έχει ο ανθρώπινος παράγοντας απέναντι στις μηχανές. Ο έλεγχος αυτός που πραγματοποιείται απέναντι στον τρόπο λειτουργίας και ανταλλαγής πληροφοριών μεταξύ των μηχανών λιγοστεύει ολοένα και περισσότερο καθώς γίνεται η εισαγωγή ολοένα και πιο σύγχρονων τεχνολογιών που δημιουργούν ένα περιβάλλον το οποίο περιλαμβάνει έξυπνες μηχανές που μπορούν να είναι σχεδόν ανεξάρτητες. Μια από τις σημαντικότερες εισαγωγές με στόχο την εξέλιξη της βιομηχανίας που συνέβαλλε στην αναβάθμιση της λειτουργίας και επικοινωνίας των συστημάτων των μηχανών αποτέλεσε αυτή του διαδικτύου των πραγμάτων. Συνεπώς μια δραματική εξέλιξη που συνδέει τον φυσικό με τον τεχνολογικό τομέα συνοδεύεται με ένα σύνολο από αλλαγές με σκοπό την αυτοματοποίηση των μηχανών, μια αλλαγή που χρειάζεται ιδιαίτερη παρακολούθηση.

Όσον αφορά λοιπόν τον έλεγχο που πραγματοποιείται από την μεριά του ανθρώπινου παράγοντα είναι σημαντικό να είναι ιδιαίτερα συγκεκριμένος και συνεχόμενος. Είναι επίσης απαραίτητη η παραγωγή χρήσιμων αποτελεσμάτων που σχετίζονται με την παραγωγικότητα, λειτουργία και κατάσταση όλων των μηχανών που χρησιμοποιούνται σε μια σύγχρονη βιομηχανία. Για να υλοποιηθεί η παραπάνω απαίτηση είναι απαραίτητο να υπάρχουν τεχνολογίες οι οποίες πλαισιώνουν το περιβάλλον με τέτοιο τρόπο ώστε να γίνεται συνεχής παρακολούθηση σε πραγματικό χρόνο. Μια τέτοια λειτουργία μπορεί να αποβεί καθοριστική για την αποφυγή υπολειτουργικότητας και προβλημάτων που σχετίζονται με την κατάσταση των μηχανών. Μπορεί επίσης να προβλέψει μελλοντικές βλάβες οι οποίες θα μπορούσαν να προκαλέσουν κάποια καταστροφή, για παράδειγμα σε ένα εργοστάσιο.

Όσον αφορά την παρακολούθηση αλλά και την οπτικοποίηση και ανάλυση των δεδομένων υπάρχει ένα σύνολο από πλατφόρμες που χρησιμοποιούνται συγκεκριμένα για τις εφαρμογές της βιομηχανίας. Οι πλατφόρμες αυτές δίνουν την δυνατότητα εκτέλεσης εφαρμογών δημιουργώντας συστήματα που μπορούν να έχουν άμεση επικοινωνία με τις συσκευές και τις μηχανές. Προσφέρουν επίσης ένα πολύ πιο φιλικό στον χρήστη περιβάλλον ώστε να μπορεί να έχει μια πιο καθαρή εικόνα της αναπαράστασης των δεδομένων. Τα συστήματα που δημιουργούνται πρέπει να παρουσιάζουν ιδιότητες όπως αυτή της ευελιξίας και της ανεξαρτησίας έτσι ώστε να μπορούν να προσαρμόζονται ανάλογα με τα σενάρια και τις απαιτήσεις της κάθε εφαρμογής καθώς αναφερόμαστε σε ένα περιβάλλον που αλλάζει συνεχώς και παρουσιάζει εξελίξεις.

Ένα ακόμα πολύ σημαντικό ζήτημα είναι η ευκολία στην εγκατάσταση, συντήρηση και διαχείριση όλων των εφαρμογών που συμβάλλουν στην μελέτη και λειτουργία των μηχανών. Οι εφαρμογές αυτές πρέπει να έχουν την λιγότερη δυνατή έκταση, να μπορούν να ελεγχθούν και διορθωθούν με ευκολία, να είναι διαχωρισμένες ώστε η λειτουργικότητα τους να μην εξαρτάται από την λειτουργικότητα άλλων εφαρμογών που εκτελούνται. Η δυνατότητα μια ενιαία εφαρμογή να χωριστεί σε μικρότερα κομμάτια όπου το κάθε ένα θα εκτελεί μια συγκεκριμένη λειτουργία δίνει το πλεονέκτημα ακόμα καλύτερης και γρηγορότερης επίλυσης προβλημάτων αλλά και την ευκολία προσαρμογής σε διαφορετικούς τύπους δομής

ενός συστήματος. Μία από τις μεγαλύτερες προκλήσεις είναι αυτή της κατάλληλης τοποθέτησης των υπηρεσιών σε ένα σύστημα. Ένα από τα ερωτήματα που προκύπτουν είναι εάν οι μικροϋπηρεσίες αυτές είναι δυνατόν να ανατοποθετηθούν σε διαφορετικά περιβάλλοντα και αν μια τέτοια συνθήκη μπορεί να επηρεάσει την λειτουργία του υπόλοιπου συστήματος.

Αξίζει να σημειωθεί πως συνδυάζοντας έναν αριθμό τεχνολογιών όπως της υπολογιστικής στα άκρα του δικτύου, των πλατφορμών που χρησιμοποιούνται στο βιομηχανικό διαδίκτυο και εκτελούνται στο επίπεδο αυτό αλλά και των μικροϋπηρεσιών μπορεί να εξυπηρετηθεί ο μεγαλύτερος αριθμός σεναρίων που εκτελούνται στον τομέα της βιομηχανίας. Στην εργασία αυτή εκτός από την προσομοίωση ενός σεναρίου που μπορεί να εφαρμοστεί στον τομέα της βιομηχανίας αντιμετωπίζεται και μια πρόκληση η οποία μπορεί να προκύψει και να οδηγήσει σε απώλειες δεδομένων και όχι μόνο. Είναι σημαντικό να υπάρχει ευελιξία και να διαβεβαιώνεται το γεγονός ότι ένα σύστημα θα λειτουργεί ομαλά ακόμα και αν υπάρξει κάποια βλάβη ή διακοπή μιας από τις υπηρεσίες που εκτελούνται.

Έτσι λοιπόν στο πρώτο κομμάτι της εργασίας παρουσιάζεται αναλυτικά όλο το θεωρητικό πλαίσιο το οποίο καλύπτει όλες τις τεχνολογίες που επιλέχθηκαν για την υλοποίηση του πρακτικού μέρους της εργασίας. Έχει γίνει επιλογή κυρίως να αναλυθούν οι τεχνολογίες και μεθοδολογίες που χρησιμοποιήθηκαν τονίζοντας την σημαντικότητα τους τα πλεονεκτήματα και τα μειονεκτήματα τους. Σε περιπτώσεις που κρίθηκε απαραίτητο αναλύονται και άλλες τεχνολογίες οι οποίες δεν έχουν χρησιμοποιηθεί στην εργασία όμως αποτελούν σημαντικό μέρος του γενικού πλαισίου και βοηθούν στο να γίνει πιο κατανοητό το περιεχόμενο αλλά και να μπορεί να πραγματοποιηθεί σύγκριση και αξιολόγηση των επιλογών που έγιναν.

Έχοντας ως βάση το θεωρητικό μέρος σχεδιάστηκε και υλοποιήθηκε μια προσομοίωση ενός ολοκληρωμένου συστήματος ανάγνωσης και αποστολής δεδομένων με την χρήση πρωτοκόλλου που χρησιμοποιείται ευρέως στις εφαρμογές του βιομηχανικού διαδικτύου των πραγμάτων. Έχει προσομοιωθεί και η πλευρά ενός αισθητήρα που αποστέλλει τα δεδομένα καταμέτρησης θερμοκρασίας από μια υποθετική μηχανή αλλά και η πλευρά της εφαρμογής που διαβάζει τα δεδομένα. Η εφαρμογή που πραγματοποιεί την ανάγνωση εκτελείται στο περιβάλλον μιας πλατφόρμας που χρησιμοποιείται στο βιομηχανικό διαδίκτυο. Το σενάριο επικεντρώνεται στην αντιμετώπιση βλάβης της πλατφόρμας από δύο πιθανά αίτια. Η λύση που προτείνεται, δοκιμάστηκε και εκτελείται επιτυχώς είναι η μετεγκατάσταση της υπηρεσίας στο περιβάλλον διαφορετικής πλατφόρμας. Η λύση αυτή αποδεικνύει τόσο την χρησιμότητα του χαρακτηριστικού της ευελιξίας στον σχεδιασμό των εφαρμογών όσο και την σημαντικότητα του χαρακτηριστικού της προσαρμοστικότητας σε διαφορετικά περιβάλλοντα. Όλα τα βήματα που αφορούν στο κομμάτι της υλοποίησης ελέγχονται και αξιολογούνται με την χρήση δοκιμών και παρουσίασης αποτελεσμάτων.

1.2 Στόχοι και σκοπός εργασίας

Το θέμα της παρούσας διπλωματικής εργασίας εντάσσεται στο περιβάλλον των εφαρμογών συστημάτων του διαδικτύου των πραγμάτων και λύσεων που εφαρμόζονται για την ενίσχυση του βιομηχανικού τομέα. Η παρούσα διπλωματική εργασία αποτελεί μια εκτενή και πρωτοποριακή διερεύνηση μιας συγκεκριμένης λύσης που αφορά την αποφυγή διακοπής του συστήματος σε περίπτωση βλάβης σε μια πλατφόρμα του βιομηχανικού διαδικτύου των πραγμάτων. Κύριο στόχο αποτελεί η επιτυχία της διαπίστωσης πως σε μια τέτοια περίπτωση είναι δυνατόν μια εφαρμογή να εγκατασταθεί εκ νέου σε μια νέα πλατφόρ-

μα, που αποτελεί τελείως διαφορετικό περιβάλλον από το αρχικό, χωρίς αυτό να επηρεάζει τόσο την λειτουργία που εκτελεί η αντίστοιχη εφαρμογή όσο και την λειτουργία ολόκληρου του υπόλοιπου συστήματος. Ακόμα ένας στόχος είναι να αποδειχθεί πόσο αποτελεσματική είναι η εφαρμογή της λογικής των μικροϋπηρεσιών και πόσο αυτή ενισχύει την ομαλή λειτουργία χάρης την ευελιξία που προσφέρει.

1.3 Παρουσίαση κεφαλαίων

Τα Κεφάλαια 2 και 3 αφορούν το θεωρητικό κομμάτι το οποίο πλαισιώνει τη διπλωματική εργασία. Πιο συγκεκριμένα γίνεται αναφορά στις διάφορες περιόδους εξέλιξης του τομέα της βιομηχανίας με μεγαλύτερη έμφαση στην τέταρτη βιομηχανική επανάσταση. Γίνεται επίσης αναφορά και ανάλυση στο διαδίκτυο των πραγμάτων που αποτελεί την εκκίνηση για τη θεματική περιοχή στην οποία εντάσσεται το σενάριο της πρακτικής υλοποίησης. Έπειτα αναλύονται η έννοια, οι εφαρμογές και όλα τα στοιχεία γύρω από την έννοια του βιομηχανικού διαδικτύου των πραγμάτων. Το τελευταίο μέρος του 2ου κεφαλαίου επικεντρώνεται στην ανάλυση των κυριότερων πρωτοκόλλων που χρησιμοποιούνται στις εφαρμογές του διαδικτύου των πραγμάτων στην βιομηχανία μέσα στα οποία συμπεριλαμβάνεται και το πρωτόκολλο που επιλέχθηκε στο σενάριο της υλοποίησης.

Το Κεφάλαιο 3 επικεντρώνεται στην παρουσίαση όλων των τεχνολογιών που σχετίζονται με το Βιομηχανικό Διαδίκτυο των Πραγμάτων και υλοποιήθηκαν στην εργασία. Έτσι ξεκινώντας με την επεξήγηση της έννοιας της υπολογιστικής στα άκρα του δικτύου αλλά και με την παράθεση των πλεονεκτημάτων που προσδίδει, γίνεται η παρουσίαση της εφαρμογής της συγκεκριμένα στις εφαρμογές τις βιομηχανίας. Έπειτα παρουσιάζεται η αρχιτεκτονική των μικροϋπηρεσιών σε συνδυασμό με όλα τα πλεονεκτήματα της σε σύγκριση με την μονολιθική αρχιτεκτονική η οποία πλέον φαίνεται να εφαρμόζεται σε ολοένα και λιγότερα σενάρια. Το επόμενο κομμάτι αφιερώνεται αποκλειστικά στην παρουσίαση των δύο πλατφορμών που χρησιμοποιήθηκαν για την υλοποίηση του σεναρίου ώστε να γίνει μια περιγραφή της αρχιτεκτονικής και των χαρακτηριστικών τους. Τέλος αναλύονται τα εργαλεία που χρησιμοποιήθηκαν τόσο στην υλοποίηση όσο και στο κομμάτι της αξιολόγησης και παραγωγής αποτελεσμάτων.

Το δεύτερο και εκτενέστερο μέρος της εργασίας που αποτελείται από το Κεφάλαιο 4 και το Κεφάλαιο 5, αφορά στην υλοποίηση ενός σεναρίου το οποίο εντάσσεται στο περιβάλλον της βιομηχανίας και του διαδικτύου των πραγμάτων. Συγκεκριμένα αφορά μια περίπτωση η οποία μπορεί να παρουσιαστεί και με πραγματικά δεδομένα σε μια βιομηχανία η οποία έχει επιλέξει να λειτουργεί χρησιμοποιώντας ειδικές πλατφόρμες. Στο Κεφάλαιο 4 αρχικά περιγράφονται αναλυτικά όλα τα στοιχεία που δομήθηκαν και χρησιμοποιήθηκαν για να ικανοποιήσουν τις ανάγκες του σεναρίου αλλά και το περιβάλλον εργασίας. Έπειτα γίνεται μια σύντομη κριτική όσον αφορά την επιλογή των πλατφορμών που χρησιμοποιήθηκαν για την υλοποίηση και το πόσο αυτές ικανοποιούν τις ανάγκες του σεναρίου. Στην συνέχεια περιγράφεται όλη η διαδικασία της υλοποίησης αναλύοντας όλα τα βήματα που ακολουθήθηκαν από την δημιουργία του σεναρίου έως το τελικό στάδιο της εκτέλεσης.

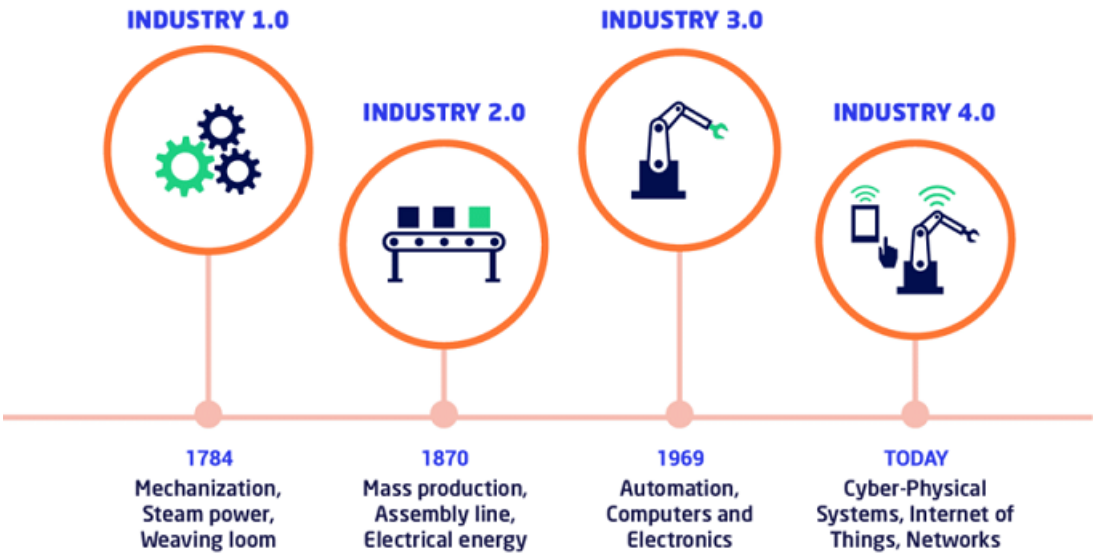
Στο Κεφάλαιο 5 γίνεται αξιολόγηση όλου του σεναρίου που υλοποιήθηκε έπειτα από μετρήσεις χρόνων των οποίων τα αποτελέσματα σχολιάζονται αναλυτικά. Στο τελευταίο κεφάλαιο διατυπώνονται όλα τα συμπεράσματα που προκύπτουν έπειτα από την διαδικασία της υλοποίησης και προτείνονται επιπλέον βελτιώσεις και προτάσεις για την ακόμα μεγαλύτερη εξέλιξη της εργασίας και της εφαρμογής της σε πραγματικά σενάρια.

Κεφάλαιο 2ο: Βιομηχανία 4.0 και βιομηχανικό διαδίκτυο

Η έννοια της βιομηχανίας αναφέρεται στο κομμάτι αυτό της οικονομίας το οποίο είναι υπεύθυνο για την παραγωγή υλικών και κατ' επέκταση υπηρεσιών [1]. Η διαδικασία της παραγωγής αυτής τείνει να γίνεται σχεδόν σε ολοκληρωτικό βαθμό αυτοματοποιημένη και να πραγματοποιείται όσο το δυνατόν αποκλειστικά με την χρήση μηχανών. Στο παρόν κεφάλαιο παρουσιάζεται η διαδικασία της εξέλιξης της φύσης των τεχνολογιών συγκεκριμένα στην περίοδο της τέταρτης βιομηχανικής επανάστασης, στην οποία και εντάσσεται το κλίμα της εργασίας.

2.1 Ιστορική αναδρομή

Στην πορεία της εισαγωγής των μηχανών στην διαδικασία της παραγωγής, οι διάφορες εμπλοκές της τεχνολογίας οδήγησαν στον διαχωρισμό των περιόδων σε βιομηχανικές επαναστάσεις [2]. Οι μεταβολές και προσθήκες στον τομέα της βιομηχανίας κυρίως οφείλονται στην όλο και μεγαλύτερη αύξηση της παραγωγής και των αναγκών.Στην πρώτη βιομηχανική επανάσταση όπου πλέον οι μηχανές εισάγονται στην διαδικασία της παραγωγής, κύριο χαρακτηριστικό αποτελεί η εντατικοποίηση της χρήσης νέων ενεργειακών πόρων όπως για παράδειγμα το νερό και ο ατμός. Έπειτα και από την μαζική παραγωγή η οποία εντάσσει πλέον την βιομηχανία στην περίοδο της Βιομηχανίας 2.0 (Industry 2.0) ακολουθεί η αυτοματοποίηση και η ψηφιοποίηση των μηχανισμών που αποτελούν χαρακτηριστικά που οδήγησαν στην περίοδο της Βιομηχανίας 3.0 (Industry 3.0) που φέρνει ακόμα μια μεγάλη εξέλιξη. Παρ' όλη την ανάπτυξη κατά τη διάρκεια όλων των παραπάνω περιόδων τόσο ο αριθμός της ζήτησης αγαθών όσο και η μείωση των διαθέσιμων πόρων συνεχώς είχαν αυξητική τάση. Σε αυτό το σημείο με την είσοδο εντελώς πρωτόγνωρων εννοιών για τη βιομηχανία όπως αυτή της μηχανικής μάθησης και του διαδικτύου των πραγμάτων περνάμε πλέον στην 4η βιομηχανική επανάσταση η οποία αποτελεί και το μεγαλύτερο ενδιαφέρον καθώς έρχεται να ικανοποιήσει έναν μεγάλο αριθμό αναγκών. Στην Εικόνα 2.1 παρουσιάζεται συνοπτικά η εξελικτική διαδικασία από την πρώτη έως την τέταρτη βιομηχανική επανάσταση.



Εικόνα 2.1: Βιομηχανική εξέλιξη [3]

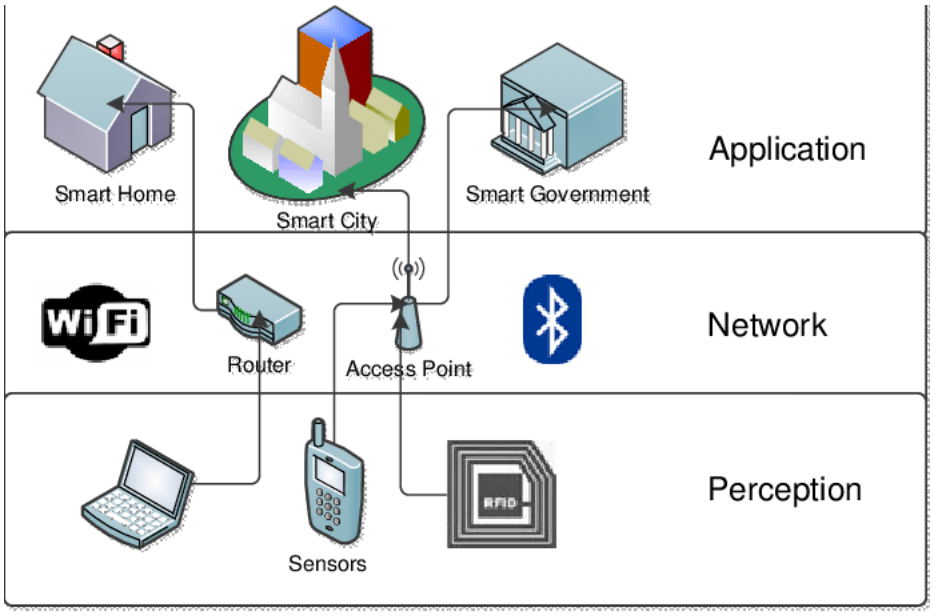
2.2 Βιομηχανία 4.0

Η ανάγκη που οδήγησε στην εξέλιξη που αφορά στην τέταρτη βιομηχανική επανάσταση είναι κυρίως η δημιουργία ενός περιβάλλοντος μέσα στο οποίο οι μηχανές πλέον θα αλληλεπιδρούν μεταξύ τους αλλά και με το περιβάλλον με έναν πλέον πιο ”έξυπνο” και αυτόνομο τρόπο [4, 5]. Η αλληλεπίδραση αυτή η οποία πραγματοποιείται με όλο και λιγότερη εμπλοκή του ανθρώπου, αφορά κυρίως στην ανταλλαγή δεδομένων, η οποία προσδίδει μια επιπλέον δυνατότητα στις μηχανές να μπορούν να βελτιώσουν την λειτουργία τους αυτόνομα αλλά και να μπορούν ακόμα και να προβλέψουν μελλοντικές βλάβες. Συνεπώς η παραπάνω εξέλιξη έρχεται να καλύψει σε ακόμα μεγαλύτερο βαθμό την ανάγκη της βιομηχανίας για ποσότητα και ποιότητα παραγωγής με μεγαλύτερη αυτοματοποίηση. Έτσι η έννοια της Βιομηχανίας 4.0 (Industry 4.0) μπορούμε να πούμε πως συνοψίζεται με τρία κεντρικά παραδείγματα [6]: το έξυπνο προϊόν, δηλαδή η δυνατότητα των προϊόντων συλλέγοντας πόρους αυτόνομα να μπορούν να οργανώσουν την διαδικασία της παραγωγής μέχρι τέλους, οι έξυπνες μηχανές, οι οποίες μετατρέπονται σε κυβερνοφυσικά συστήματα παραγωγής και καταργούν την μέχρι τώρα ιεραρχία στην διαχείριση των συστημάτων δημιουργώντας ένα περιβάλλον με μεγαλύτερη ευελιξία στην διαχείριση και τα νέα δίκτυα παραγωγής τα οποία μπορούν να οργανωθούν αυτόνομα. Τέλος η επαυξημένη διαχείριση, η οποία δεν έχει ως στόχο την ολοκληρωτική κατάργηση του ανθρώπινου παράγοντα αλλά την μετατροπή του σε μια πιο ευέλικτη οντότητα όπου μπορεί να διαχειριστεί ένα ευρύ φάσμα εργασιών στην διαδικασία της παραγωγής.

Η μετάβαση στην περίοδο αυτή ξεκίνησε στις αρχές του 21ου αιώνα [7]. Με αυτή την αλλαγή πλέον διαμορφώνεται ένα νέο περιβάλλον στο οποίο πραγματοποιείται η παραγωγή. Μερικές από τις μεταβολές που οδήγησαν στην εξέλιξη που περιγράφεται είναι η απαίτηση για ευελιξία τόσο του τρόπου λειτουργίας των συστημάτων όσο και στην αναβάθμιση των προϊόντων και η αποκέντρωση καθώς αυξάνεται η ανάγκη να λαμβάνονται πιο γρήγορα αποφάσεις [8]. Για να ικανοποιηθούν οι απαιτήσεις οι οποίες πλαισιώνουν τον ψηφιακό σχεδιασμό του περιβάλλοντος χρησιμοποιείται ένα νέο σύνολο από τεχνολογίες και εξοπλισμό [9]. Για παράδειγμα εισάγονται ελεγκτές και αισθητήρες οι οποίοι είναι υπεύθυνοι για τη συλλογή και αποστολή δεδομένων από τις μηχανές με σκοπό την παρακολούθηση, αυτόνομη βελτίωση και ”εκπαίδευση” τους. Επιπλέον εισαγωγές αποτελούν έννοιες όπως αυτή της ρομποτικής, της επαυξημένης πραγματικότητας, της μηχανικής και αυτόνομης μάθησης και της τεχνητής νοημοσύνης.

2.3 Διαδίκτυο των Πραγμάτων

Η αύξηση των συσκευών στο περιβάλλον της βιομηχανίας και η ανάγκη διαχείρισης και βελτίωσης τους οδηγεί σε ακόμα μια σημαντική μεταβολή στον τομέα της βιομηχανίας. Μια ακόμα πτυχή που αλλάζει λοιπόν, είναι ότι πλέον υπάρχει η δυνατότητα τόσο οι μηχανές όσο και ολόκληρα συστήματα να επικοινωνούν μεταξύ τους ανταλλάσσοντας πληροφορίες χάρης την συμβολή του διαδικτύου. Πιο συγκεκριμένα, για να είναι εφικτή η διασύνδεση των στοιχείων εμφανίζονται νέες τεχνολογίες επικοινωνίας με το Διαδίκτυο των Πραγμάτων (Internet of Things - IoT) να χρησιμοποιείται κατά κόρον στον τομέα της βιομηχανίας [10] αλλά και σε πολλούς άλλους τομείς όπως είναι αυτός της γεωργίας ή της υγείας. Αναφερόμαστε σε ένα δίκτυο στο οποίο μέσω αισθητήρων πραγματοποιείται αλληλεπίδραση μεταξύ του φυσικού και του ψηφιακού περιβάλλοντος [11]. Πιο συγκεκριμένα το σύνολο των συσκευών, μηχανών, κτηρίων, πρωτοκόλλων επικοινωνίας και λογισμικού δημιουργεί ένα δίκτυο στο οποίο τα στοιχεία που το συντελούν ανταλλάσσουν, αποθηκεύουν επεξεργάζονται και αναλύουν δεδομένα τα οποία είναι πολύτιμα για την ανάπτυξη ολόκληρου του συστήματος και της λειτουργίας του δικτύου.



Εικόνα 2.2: Βασική Αρχιτεκτονική του διαδικτύου των πραγμάτων με τρία επίπεδα [12]

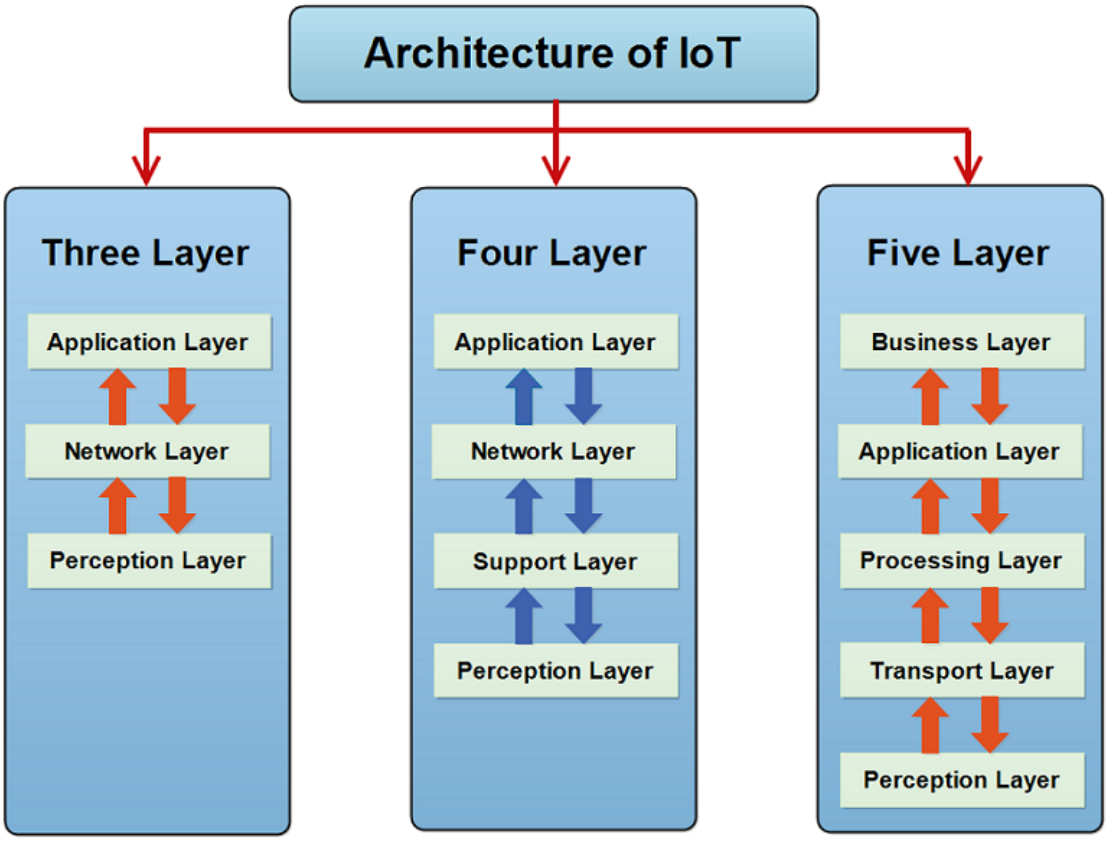
Αν και σε κάθε τομέα το διαδίκτυο των πραγμάτων εφαρμόζεται με διαφορετική μορφή υπάρχει μια βασική γραμμή στην οποία κινείται η διαμόρφωση της αρχιτεκτονικής του. Η αρχή πραγματοποιήθηκε με τα τρία κύρια επίπεδα τα οποία διαμορφώνουν την αρχιτεκτονική σε γενικό πλαίσιο [13]. Στην Εικόνα 2.2 γίνεται μια σύντομη απεικόνιση των επιπέδων αυτών μαζί με βασικά στοιχεία που τα συνοδεύουν. Στην βάση της ιεραρχίας βρίσκεται το επίπεδο της αντίληψης (perception) το οποίο συμπεριλαμβάνει όλα τα είδη των αισθητήρων και ενεργοποιητών. Οι αισθητήρες αυτοί μπορούν να χρησιμοποιηθούν σε συσκευές του διαδικτύου των πραγμάτων με σκοπό την συλλογή δεδομένων από το περιβάλλον και την αποστολή τους. Τα είδη τους ποικίλουν έτσι ώστε να μπορούν να πραγματοποιούνται μετρήσεις σε διάφορους τομείς και έτσι για παράδειγμα συμπεριλαμβάνονται αισθητήρες θερμοκρασίας που βοηθάνε στην μέτρηση και κατ' επέκταση τον έλεγχο της θερμοκρασίας, αισθητήρες πίεσης που συμβάλλουν στην μέτρηση έντασης ροής του αέρα ή των υγρών.

Το αμέσως επόμενο επίπεδο με την ονομασία επίπεδο δικτύου (network layer) είναι υπεύθυνο για την μεταφορά και επεξεργασία των δεδομένων που συλλέχθηκαν στα ανώτερα επίπεδα που ακολουθούν αλλά επίσης σε περίπτωση που επρόκειτο να σταλεί αντιστρόφως μια εντολή στο επίπεδο της αντίληψης. Το τελευταίο επίπεδο που βρίσκεται στην κορυφή της ιεραρχίας ή αλλιώς το επίπεδο εφαρμογής (application layer) αναφέρεται σε όλες τις εφαρμογές οι οποίες χρησιμοποιούν την τεχνολογία του διαδικτύου των πραγμάτων [13] και είναι υπεύθυνο για τον ορισμό της διεπαφής για τους χρήστες ώστε να είναι διαθέσιμη η πρόσβαση σε εφαρμογές του διαδικτύου των πραγμάτων. Γενικότερα έχει την ευθύνη παροχής υπηρεσιών, η οποία εξαρτάται από το σενάριο που υλοποιείται, στις εφαρμογές που υλοποιείται το διαδίκτυο των πραγμάτων.

Η παραπάνω μορφή αρχιτεκτονικής αποτελεί ένα βασικό και απλουστευμένο μοντέλο το οποίο δεν μπόρεσε να καλύψει σε μεγάλο βαθμό τις ανάγκες που παρουσιάζει η τεχνολογία αυτή. Συνεπώς στη συνέχεια προστέθηκαν επιπλέον επίπεδα για να εξυπηρετήσουν περισσότερες ανάγκες [13]. Μία από τις προσθήκες που αναγνωρίστηκε είναι αυτή του επιπέδου υποστήριξης (support layer) το οποίο τοποθετείται ανάμεσα στα επίπεδα δικτύου και perception. Επικεντρώνεται κυρίως σε θέματα ασφαλείας

και συγκεκριμένα σιγουρεύει πως όλα τα μηνύματα που εισέρχονται προέρχονται από εξουσιοδοτημένη πηγή. Ακόμα μια προσθήκη είναι αυτή του υπολογιστικού (computation) ή επεξεργασίας (processing) επίπεδου στο οποίο συλλέγονται όλα τα προηγούμενα εισερχόμενα δεδομένα από το επίπεδο δικτύου και πραγματοποιούνται αναλύσεις μέσω των οποίων προκύπτουν συμπεράσματα που οδηγούν στην λήψη μελλοντικών αποφάσεων. Ένα από τα κυριότερα στάδια των διαδικασιών επεξεργασίας δεδομένων είναι η συσσώρευση δεδομένων όπου πραγματοποιείται ταξινόμηση των εισερχόμενων δεδομένων ώστε στην πορεία να μπορούν να αποθηκευτούν με αποτελεσματικότερο τρόπο. Ακόμα το επίπεδο επιχείρησης (business) το οποίο αποτελεί μια επέκταση του επιπέδου εφαρμογής και έχοντας έναν ρόλο διαχείρισης του συστήματος, απαλλάσσει στον χρήστη της εφαρμογής του διαδικτύου των πραγμάτων από πιθανά προβλήματα.Τέλος μια από τις σημαντικότερες προσθήκες είναι αυτή του επιπέδου άκρων (edge layer) η οποία θα συζητηθεί εκτενώς στην αντίστοιχη ενότητα. Στην Εικόνα 2.3 παρουσιάζεται η πορεία προσθήκης επιπλέον επιπέδων στην αρχιτεκτονική του διαδικτύου των πραγμάτων.

Τόσο οι αρχιτεκτονικές όσο και οι αναφορές σε τεχνολογίες και πρωτόκολλα επικοινωνίας ποικίλουν από εφαρμογή σε εφαρμογή. Όλες οι εφαρμογές σε διάφορους τομείς παρουσιάζουν μια μεγάλη εξέλιξη χάρης στα πλεονεκτήματα που προσφέρει η υλοποίηση του διαδικτύου των πραγμάτων. Πλέον, υπάρχει μεγαλύτερος έλεγχος στην διαχείριση και λειτουργία των συστημάτων, υπάρχει αξιοσημείωτη μείωση στην κατανάλωση πόρων και ενέργειας και υπάρχει μεγαλύτερη ακρίβεια. Τα παραπάνω πλεονεκτήματα σε συνδυασμό με μεγάλη εξέλιξη και επιπλέον δυνατότητες εξηγούν την υλοποίηση του διαδικτύου των πραγμάτων ειδικά στον τομέα της βιομηχανίας στην οποία εντάσσεται πλέον ο όρος Βιομηχανικό Διαδίκτυο των Πραγμάτων (Industrial Internet of Things - IIoT).



Εικόνα 2.3: Προσθήκες στα επίπεδα της αρχιτεκτονικής του διαδικτύου των πραγμάτων [13]

2.4 Βιομηχανικό Διαδίκτυο των Πραγμάτων

Η ευελιξία όσον αφορά την ταχύτατη προσαρμογή στο περιβάλλον της βιομηχανίας και την κατανομή τον πόρων στο περιβάλλον αυτό,η εκτέλεση βέλτιστων διαδικασιών και η διασύνδεση των βιομηχανικών πόρων με σκοπό την επικοινωνία και την αλληλεπίδραση είναι μερικές από τις υπηρεσίες στις οποίες είναι προσανατολισμένη η υλοποίηση του βιομηχανικού διαδικτύου των πραγμάτων [14].

Με τον όρο IIoT, αναφερόμαστε σε όλες τις εφαρμογές του διαδικτύου των πραγμάτων που σχετίζονται με το βιομηχανικό περιβάλλον. Το διαδίκτυο των πραγμάτων με την εισαγωγή του στον τομέα της βιομηχανίας συμβάλλει στην δημιουργία ενός δικτύου το οποίο αποτελείται από βιομηχανικές συσκευές κατάλληλα εξοπλισμένες με ενεργοποιητές και αισθητήρες. Οι συσκευές αυτές συνδέονται και επικοινωνούν μεταξύ τους μέσω τεχνολογιών και πρωτοκόλλων επικοινωνίας με σκοπό την έξυπνη παραγωγή [15]. Μια από τις θετικές συνέπειες της ανάπτυξης αυτής είναι πως μηχανές και κεντρικοί υπολογιστές επικοινωνούν μέσω δικτύου με τα συστήματα υλικού και λογισμικού μέσω των οποίων πραγματοποιούνται οι έλεγχοι που σχετίζονται με τη λειτουργία ενός εργοστασίου. Όλη η πληροφορία που παράγεται μέσα από την επικοινωνία αυτή οφείλεται στην βελτιστοποίηση των βιομηχανικών διαδικασιών και του βιομηχανικού περιβάλλοντος. Επιπλέον η αυτοματοποίηση και ψηφιοποίηση των διαδικασιών οδηγεί σε μια κατάσταση στην οποία μειώνεται σε μεγάλο βαθμό ο αριθμός και το κόστος των διαδικασιών που εφαρμόζονται στην παραγωγή. Παρακάτω αναλύονται η αρχιτεκτονική του συγκεκριμένου είδους δικτύου, τα πρωτόκολλα που χρησιμοποιούνται για την ανταλλαγή δεδομένων και την επικοινωνία μεταξύ των στοιχείων του αλλά και ένα σύνολο αναφορών παραδειγμάτων και εφαρμογών στις οποίες εφαρμόζεται και αφορούν την βιομηχανία.

Η ύπαρξη του διαδικτύου των πραγμάτων στην βιομηχανία εντάσσει νέες έννοιες οι οποίες το χαρακτηρίζουν και μπορούν να θεωρηθούν και ως αρχές του [16]. Αρχικά τα δεδομένα που παράγονται από τις συσκευές πλέον έχουν έναν όγκο ο οποίος είναι απαραίτητο να διαχειριστεί ώστε να προκύψουν χρήσιμες πληροφορίες. Ένα παράδειγμα μπορούν να αποτελέσουν τα δεδομένα μιας μηχανής στην οποία υπάρχει τοποθετημένος αισθητήρας θερμοκρασίας που πραγματοποιεί μετρήσεις κάθε λεπτό. Οι εφαρμογές επεξεργασίας δεδομένων που χρησιμοποιούνται συνήθως σε αυτή την περίπτωση αντικαθιστώνται από hiveql και handoop τεχνικές. Η έννοια αυτή παρουσιάζεται ως έννοια των Δεδομένων Μεγάλου Όγκου (Big Data) και φέρνει ποικίλες προκλήσεις στον τομέα της ψηφιοποίησης, διαχείρισης, ανάλυσης αλλά και της αποθήκευσης των δεδομένων. Όσον αφορά την παραγωγή πολύτιμων αποτελεσμάτων, μπορεί να είναι δυνατόν να προκύψουν άμεσα αλλά από την άλλη μπορεί να απαιτείται ένα μεγάλο διάστημα ώστε το μέγεθος της πληροφορίας να θεωρείται επαρκές για την εξαγωγή συμπερασμάτων. Τα αποτελέσματα αυτά προσφέρουν τελικά μια επιπλέον γνώση στο σύστημα ή στον μηχανικό ανάλογα με την περίπτωση του σεναρίου .

Μια ακόμα έννοια είναι αυτή της διαρκούς συνδεσιμότητας. Καθώς οι συσκευές στο βιομηχανικό περιβάλλον διατηρούνται μονίμως συνδεδεμένες δίνεται η δυνατότητα για συνεχή παρακολούθηση της λειτουργίας τους σε πραγματικό χρόνο. Κάτι τέτοιο μπορεί να ωφελήσει βελτιώνοντας τόσο το λογισμικό όσο και το κατασκευαστικό κομμάτι. Η ροή παραγωγής και η παρακολούθηση των δεδομένων σε πραγματικό χρόνο ενισχύει σημαντικά την ασφάλεια καθώς για παράδειγμα μια καθυστέρηση σε μέτρηση ή μετρήσεις οι οποίες απέχουν μεταξύ τους ανά μεγάλα διαστήματα μπορούν να αποβούν ακόμα και καταστροφικές. Για την παροχή υπηρεσιών υπολογισμού υπεύθυνο είναι το υπολογιστικό νέφος (Cloud

Computing). Πιο συγκεκριμένα παρέχει την δυνατότητα που αφορά στην παρακολούθηση και ανάλυση όλων των αντικειμένων τα οποία συμπεριλαμβάνονται σε εφαρμογές του διαδικτύου των πραγμάτων γενικά αλλά και κυρίως στην βιομηχανία. Οι αισθητήρες έχουν την ικανότητα να αποθηκεύουν μονάχα τοπικά δεδομένα κάτι το οποίο γεννάει την ανάγκη για ένα περιβάλλον το οποίο θα μπορεί να διαχειριστεί δεδομένα που συλλέγονται από το σύνολο των στοιχείων του.

2.4.1 Πρωτόκολλα επικοινωνίας στο Βιομηχανικό Διαδίκτυο των Πραγμάτων

Η χρήση πρωτοκόλλων επικοινωνίας είναι απαραίτητη για την ομαλή αλληλεπίδραση μεταξύ των στοιχείων ενός δικτύου. Τα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται στις εφαρμογές που υλοποιείται το Βιομηχανικό Διαδίκτυο των Πραγμάτων ποικίλουν και χωρίζονται σε κατηγορίες που αφορούν τις Τεχνολογίες Πληροφορικής (IT), το IoT και την Τεχνολογία Λειτουργίας (Operational Technology - OT) [17]. Μπορεί επίσης να γίνει μια ταξινόμηση τους με βάση τα δίκτυα που εξυπηρετούν/υποστηρίζουν [18]. Για παράδειγμα το ασύρματο δίκτυο που συντελείται από τους αισθητήρες, περιλαμβάνει πρωτόκολλα όπως: WirelessHART, IEC 62591 (WirelessHART), ISA 100.11a και Zigbee. Στην επικοινωνία μηχανής προς μηχανή (Machine-To-Machine - M2M) συμπεριλαμβάνονται τα πρωτόκολλα CoAP, OPC-UA, DDS και Modbus. Έπειτα οι τεχνολογίες δικτύων ευρείας περιοχής χαμηλής ισχύος (LPWANs) αποτελούνται από πρωτόκολλα όπως τα NB-IoT, LoRa και LoRaWAN. Τα κυψελοειδή δίκτυα περιλαμβάνουν τα 5G/4G/3G/2G, LTE και WiMAX. Τέλος οι τεχνολογίες ασύρματων τοπικών δικτύων (WLANs) περιλαμβάνουν την οικογένεια πρωτοκόλλων IEEE 802.11 ενώ οι τεχνολογίες Ασύρματων Δικτύων Προσωπικής Περιοχής (WPANs) περιλαμβάνουν την οικογένεια πρωτοκόλλων IEEE 802.15.4.

Στο κεφάλαιο αυτό επιλέχθηκαν να αναλυθούν λεπτομερώς κυρίως το πρωτόκολλο MQTT το οποίο χρησιμοποιήθηκε και στο πρακτικό κομμάτι της εργασίας, το Modbus και OPC-UA που χρησιμοποιούνται κατά κόρον στις εφαρμογές στην βιομηχανία και το πρωτόκολλο HTTP που παρ' όλο που δεν ανήκει στην οικονένεια των πρωτοκόλλων αυτών και δεν χρησιμοποιείται συνήθως σε εφαρμογές του IIoT, επιλέγεται για να ενισχύσει το κομμάτι της παρακολούθησης δεδομένων. Στην εργασία το πρωτόκολλο HTTP χρησιμοποιήθηκε στο κομμάτι της υλοποίησης, κυρίως στο σημείο που πραγματοποιείται η αξιολόγηση και η αποστολή δεδομένων. Για τον λόγο αυτό αποτελεί μέρος της συγκεκριμένης ενότητας.

2.4.2 Το πρωτόκολλο επικοινωνίας MQTT

Σε ένα περιβάλλον όπως αυτό της βιομηχανίας στο οποίο εφαρμόζεται το διαδίκτυο των πραγμάτων, χρειάζεται να υπάρχει συνεχής επικοινωνία μεταξύ των συσκευών και μάλιστα με ένα πρωτόκολλο το οποίο λειτουργεί με ασύγχρονο τρόπο. Πιο συγκεκριμένα αν αναλογιστεί κανείς τον όγκο δεδομένων που ανταλλάσσονται καθημερινά θα ήταν αρκετά χρονοβόρο αλλά και δυσλειτουργικό αν για κάθε επικοινωνία χρησιμοποιούνταν πρωτόκολλα στα οποία για να ολοκληρωθεί ένα αίτημα αναμένεται πάντα απάντηση. Σε περιπτώσεις όπως αυτή της αποστολής δεδομένων από τους αισθητήρες και στη συνέχεια ανάγνωσης τους, δεν είναι απαραίτητη η αναμονή για απάντηση ώστε να σταλούν εκ νέου δεδομένα. Πολλές φορές μάλιστα η αναμονή αυτή αποφεύγεται ή μειώνεται έχοντας ως σκοπό την εξοικονόμηση ενέργειας, πόρων και φόρτου του συστήματος.

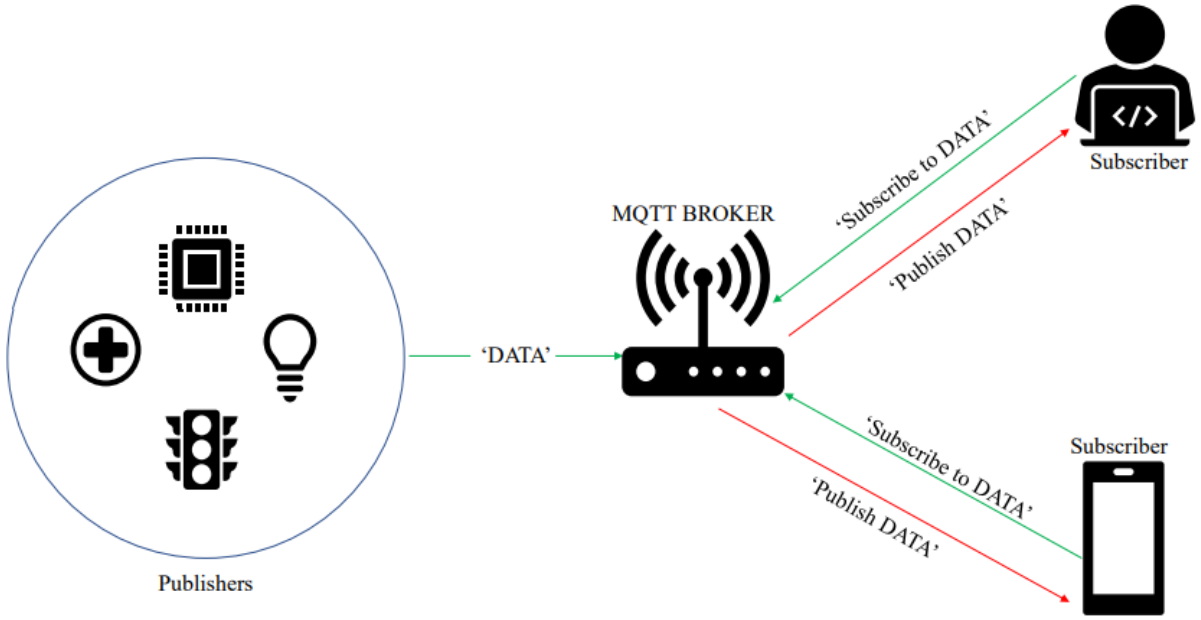
Αντιθέτως τα δεδομένα στέλνονται συνεχώς ανάλογα με τον τρόπο με τον οποίο έχει ρυθμιστεί να γίνο-

νται οι μετρήσεις και οι αποστολές. Επιπλέον υπάρχει η ανάγκη η αλληλεπίδραση να πραγματοποιείται με όσο το δυνατόν γρηγορότερους ρυθμούς και ακόμα η κάθε ξεχωριστή επικοινωνία να μην επιβαρύνει σε μεγάλο βαθμό το δίκτυο. Ακόμα, το χαρακτηριστικό της επεκτασιμότητας πρέπει να ικανοποιείται, εκτός από την αρχιτεκτονική με βάση την οποία σχεδιάζεται το σύστημα, και από τα πρωτόκολλα που εφαρμόζονται στην επικοινωνία έτσι ώστε να μπορούν να προστεθούν ή να αφαιρεθούν από το δίκτυο συσκευές χωρίς να υπάρχει κάποια συνέπεια. Έτσι μια από τις επιλογές που συνήθως πραγματοποιείται και η οποία εφαρμόστηκε και στην εκτέλεση του σεναρίου που περιγράφεται στην υλοποίηση, είναι το πρωτόκολλο Μεταφοράς Τηλεμετρίας στην Ουρά Μηνυμάτων (Message Queuing Telemetry Transport - MQTT).

Το MQTT αποτελεί ένα πρωτόκολλο επικοινωνίας που χρησιμοποιείται στην επικοινωνία μηχανής προς μηχανή και στα συστήματα στα οποία εφαρμόζεται το διαδίκτυο των πραγμάτων [19] καθώς εξυπηρετεί ένα μεγάλο μέρος των απαιτήσεων που παρουσιάζονται στο κομμάτι της επικοινωνίας μεταξύ των συσκευών. Αρχικά ανήκει στην κατηγορία των ελαφριών (lightweigh) πρωτοκόλλων τα οποία δίνουν την δυνατότητα στα στοιχεία ενός συστήματος να έχουν μια πιο άμεση επικοινωνία και εξαιρούν ένα μεγάλο μέρος πληροφορίας η οποία δεν είναι χρήσιμη ή χρησιμοποιούν μηχανισμούς ώστε να γίνεται συμπίεση των δεδομένων. Ένα από τα πιο χρήσιμα χαρακτηριστικά του είναι ότι προσφέρει στο σύστημα την δυνατότητα ασύγχρονης επικοινωνίας με αποτέλεσμα να υπάρχει μεγάλη ευελιξία όσον αφορά τα χρονικά πλαίσια μέσα στα οποία πραγματοποιείται μια ολοκληρωμένη ανταλλαγή μηνύματος.

Το MQTT εφαρμόζει την λογική του μοτίβου της δημοσίευσης - ανάγνωσης γνωστό και ως μοτίβο publisher subscribre (pub-sub). Για την εφαρμογή του αρχικά υπάρχει μια οντότητα που παρουσιάζει τα χαρακτηριστικά ενός εξυπηρετητή (server) και έχει την ονομασία broker. Ο broker λειτουργεί ως μεσολαβητής και μέσω αυτού πραγματοποιούνται όλες οι αναγνώσεις και εγγραφές δεδομένων. Όλα τα υπόλοιπα στοιχεία στο μοντέλο αυτό είναι πελάτες (clients) οι οποίοι ορίζονται ως publishers η subscribers ανάλογα με το εάν πρόκειται να δημοσιεύσουν ή να αναγνώσουν δεδομένα από τον broker αντίστοιχα. Ο αριθμός των πελατών σε μια τέτοια αρχιτεκτονική δεν είναι σταθερός ή καθορισμένος και συνεπώς είναι δυνατόν να αυξομειώνεται ανά πάσα στιγμή. Επίσης ένα ακόμα χαρακτηριστικό που δεν είναι σταθερό είναι ο ρόλος που έχει ο κάθε πελάτης μέσα στο δίκτυο.

Για παράδειγμα, μια συσκευή μπορεί να λειτουργεί για ένα διάστημα ως publisher δημοσιεύοντας δεδομένα που σχετίζονται με μετρήσεις που πραγματοποιούνται και σε μια άλλη στιγμή να χρειάζεται να διαβάσει κάποια τιμή ή εντολή και κατά συνέπεια να λειτουργήσει ως subscriber. Προϋπόθεσή για την δημιουργία οποιασδήποτε νέας επικοινωνίας αποτελεί η εγγραφή του κάθε πελάτη στον broker. Σε κάθε εγγραφή ο πελάτης δηλώνει με ποια ιδιότητα εγγράφεται και επίσης καθορίζει το περιβάλλον στο οποίο αναφέρονται τα δεδομένα με τα οποία πρόκειται να αλληλεπιδράσει. Το περιβάλλον αυτό ορίζεται από τους πελάτες που λειτουργούν με την ιδιότητα του publisher καθώς κατά την πρώτη αποστολή δεδομένων που πραγματοποιούν δημιουργούν ένα θέμα (topic). Έτσι όταν ένας subscriber θέλει να αναγνώσει συγκεκριμένα δεδομένα κατά την εγγραφή του στον broker θα δηλώσει το topic στο οποίο θέλει να έχει πρόσβαση έτσι ώστε να ξεκινήσει την ανάγνωση. Όλη η διαδικασία και μορφή που περιγράφηκε παραπάνω παρουσιάζεται συνοπτικά στην Εικόνα 2.4. Στην περίπτωση αυτή οι publishers είναι συσκευές IoT οι οποίες στέλνουν τις μετρήσεις στον broker που αφορούν το topic ”DATA” και οι subscribers είναι υπολογιστικές συσκευές και εφαρμογές οι οποίες εγγράφονται στο topic DATA και έπειτα από την πρόσβαση αυτή μπορούν να αναγνώσουν τα εξαγόμενα δεδομένα των μετρήσεων που τους αφορούν.



Εικόνα 2.4: Αρχιτεκτονική πρωτοκόλλου επικοινωνίας MQTT [20]

Κατά την εφαρμογή του MQTT μπορούν να οριστούν διάφοροι τύποι ποιότητας υπηρεσιών (Quality of Service - QoS) [21] έτσι ώστε να μπορούν να εξυπηρετηθούν πολλαπλοί τύποι αξιοπιστίας των μηνυμάτων που ανταλλάσσονται. Στο επίπεδο 0 η κάθε αποστολή μηνύματος πραγματοποιείται μόνο μια φορά και χωρίς να ελέγχεται για το εάν τα δεδομένα παραδόθηκαν επιτυχώς [22]. Έτσι κάθε μήνυμα που στέλνει ο publisher δεν επιστρέφει καμία πληροφορία που σχετίζεται με την παράδοση του. Μπορεί όμως να υπάρξουν χρονικά διαστήματα κατά τα οποία είτε ο subscriber δεν είναι διαθέσιμος να λάβει νέα δεδομένα ή δεν έχει εγγραφεί στο δίκτυο προτού ξεκινήσει η αποστολή είτε ο publisher να παρουσιάσει κάποια δυσλειτουργία [23]. Με την ποιότητα υπηρεσιών 0 στην περίπτωση αυτή δεν υπάρχει μεγάλη αξιοπιστία καθώς μπορεί να υπάρχει μεγάλη απώλεια μηνυμάτων χωρίς αυτό να γίνεται αντιληπτό. Έτσι αν για παράδειγμα υπάρχει μια ροή δεδομένων η οποία πρέπει να ληφθεί με συγκεκριμένη σειρά από τον παραλήπτη και κατά την διάρκεια της επικοινωνίας κάποια κομμάτια της χάνονται, είναι πολύ πιθανόν είτε να προκύψει λανθασμένο αποτέλεσμα ή ακόμα και να μην μπορεί να παραχθεί κάποια χρήσιμη πληροφορία. Από την άλλη πλευρά στην περίπτωση αυτή υπάρχει πολύ γρήγορη επικοινωνία και γίνεται ελάχιστη κατανάλωση ενέργειας καθώς οι διαδικασίες που εκτελούνται για την επίτευξη της επικοινωνίας είναι οι ελάχιστες δυνατές.

Στο αμέσως επόμενο επίπεδο με ποιότητα υπηρεσίας 1 προστίθεται ένα επιπλέον χαρακτηριστικό όσον αφορά την επιβεβαίωση παράδοσης. Σε αντίθεση με την ποιότητα υπηρεσίας 0, κάθε μήνυμα αποστέλλεται τουλάχιστον μια φορά και στη συνέχεια ανά διαστήματα ο αποστολέας του μηνύματος ελέγχει για την επιβεβαίωση παράδοσης από τον παραλήπτη. Η επιβεβαίωση αυτή στέλνεται με έναν έλεγχο της κατάστασης του μηνύματος που ονομάζεται PUBACK. Έτσι ο publsiher κάθε φορά που αποστέλλει ένα καινούριο μήνυμα περιμένει την επιβεβαίωση από τον subscriber [23]. Αν και η περίπτωση αυτή παρουσιάζει μια μεγαλύτερη αξιοπιστία, αν η τιμή της πληροφορίας που αφορά την κατάσταση του μηνύματος χαθεί ανεξάρτητα από το εάν το μήνυμα έχει παραδοθεί ή όχι ,τότε είναι πιθανόν ο publisher να θεωρή-

σει ότι η παράδοση απέτυχε και συνεπώς να στείλει το μήνυμα περισσότερες φορές μέχρι να λάβει την επιβεβαίωση. Σε αυτή την περίπτωση υπάρχει άσκοπη υπερφόρτωση του δικτύου καθώς ένα μήνυμα μπορεί να αποστέλλεται περισσότερες φορές από όσες είναι απαραίτητο.

Τέλος, στην τελευταία μέθοδο με ποιότητα υπηρεσίας 2, στην οποία εξασφαλίζεται πως το κάθε μήνυμα θα παραδοθεί επιτυχώς και ακριβώς μια φορά. Πιο συγκεκριμένα η διαδικασία της κάθε αποστολής αποτελείται αυτή τη φορά από δύο στάδια όπου το πρώτο είναι ακριβώς το ίδιο με την διαδικασία που ακολουθείται στο επίπεδο με ποιότητα υπηρεσίας 1. Αφού ολοκληρωθεί η διαδικασία και ο publisher λάβει την επιβεβαίωση από τον subscriber ότι το μήνυμα έχει παραδοθεί επιτυχώς στέλνει ένα επιπλέον μήνυμα με σκοπό την ενημέρωση πως η διαδικασία ήταν επιτυχής και συνεπώς τα δεδομένα είναι ολοκληρωμένα. Το μήνυμα επιβεβαίωσης ονομάζεται PUBREL και η απώλεια του έχει την ίδια επίπτωση με την απώλεια του PUBACK που περιγράφηκε στην ποιότητα υπηρεσίας 1. Έτσι, όσο πιο αυξημένο είναι το επίπεδο της ποιότητας υπηρεσίας που επιλέγεται τόσο πιο πολλά μηνύματα ανταλλάσσονται επιτυχώς και τόσο μεγαλύτερη καθυστέρηση παρουσιάζεται στην ολοκλήρωση της κάθε επικοινωνίας.

2.4.3 Το πρωτόκολλο Μεταφοράς Υπερκειμένου

Αν και είναι ένα πρωτόκολλο το οποίο χρησιμοποιείται κατά κόρον στον Παγκόσμιο Ιστό [24] παρατηρείται η εμφάνιση και η εφαρμογή του και στο περιβάλλον των υλοποιήσεων που σχετίζονται με το βιομηχανικό διαδίκτυο. Το Πρωτόκολλο Μεταφοράς Υπερκειμένου (HyperText Transfer Protocol - HTTP), αφορά στην επικοινωνία μεταξύ ενός εξυπηρετητή (server) με πελάτες (clients). Στον πυρήνα του το πρωτόκολλο αυτό δεν ορίζει ούτε διαχειρίζεται την κατάσταση των αιτημάτων (stateless) και έτσι κάθε αίτημα που αποστέλλεται από έναν πελάτη αποτελεί ανεξάρτητη οντότητα και το ιστορικό το οποίο σχετίζεται με αυτό δεν αποθηκεύεται. Βασίζεται στο μοντέλο αιτήματος-απάντησης (request-response) και έτσι έπειτα από κάθε αποστολή μηνύματος επιστρέφεται απάντηση η οποία περιλαμβάνει τον κωδικό που χαρακτηρίζει την κατάσταση του αιτήματος και μια σύντομη περιγραφή.

Όσον αφορά τα αιτήματα που μπορούν να πραγματοποιηθούν ο σκελετός τους αποτελείται από τρία βασικά μέρη τα οποία είναι: η μέθοδος που χρησιμοποιεί το αίτημα και αποσκοπεί σε μια συγκεκριμένη αλληλεπίδραση μεταξύ πελάτη και εξυπηρετητή, οι κεφαλίδες και το σώμα του αιτήματος. Αρχικά, το πλήθος μεθόδων που μπορεί να χρησιμοποιηθεί εξυπηρετεί ένα σύνολο λειτουργιών που σχετίζονται με μεταβολές των δεδομένων. Οι πιο συχνά χρησιμοποιούμενες μέθοδοι είναι η ανάκτηση δεδομένων που πραγματοποιούνται με τη μέθοδο GET, η αποστολή η δημοσίευση δεδομένων που πραγματοποιείται με τη χρήση της μεθόδου POST, η ολοκληρωτική ανανέωση και ενημέρωση ήδη υπαρχόντων δεδομένων που πραγματοποιείται με την μέθοδο PUT, η μερική μεταβολή και ενημέρωση δεδομένων που πραγματοποιείται με την μέθοδο PATCH και τέλος η διαγραφή εγγραφής ή εγγραφών δεδομένων από τον εξυπηρετητή με την μέθοδο DELETE.

Για να μπορέσουν να δοθούν περισσότερες πληροφορίες σχετικά με το αίτημα χρησιμοποιούνται οι κεφαλίδες που αποτελούν μεταδεδομένα μέσω των οποίων μπορεί να οριστεί για παράδειγμα η επιτρεπόμενη μορφή της απάντησης που μπορεί να λάβει το αίτημα, ο τύπος της αυθεντικοποίησης που πραγματοποιείται, η διεύθυνση του εξυπηρετητή. Τέλος όσον αφορά το σώμα του αιτήματος το οποίο είναι προαιρετικό αλλά εξίσου σημαντικό αποτελεί ένα σύνολο επιπλέον δεδομένων τα οποία συνήθως συντελούν την πληροφορία που προορίζεται να μεταδοθεί στον εξυπηρετητή. Μπορεί είτε να είναι κενό όπως για

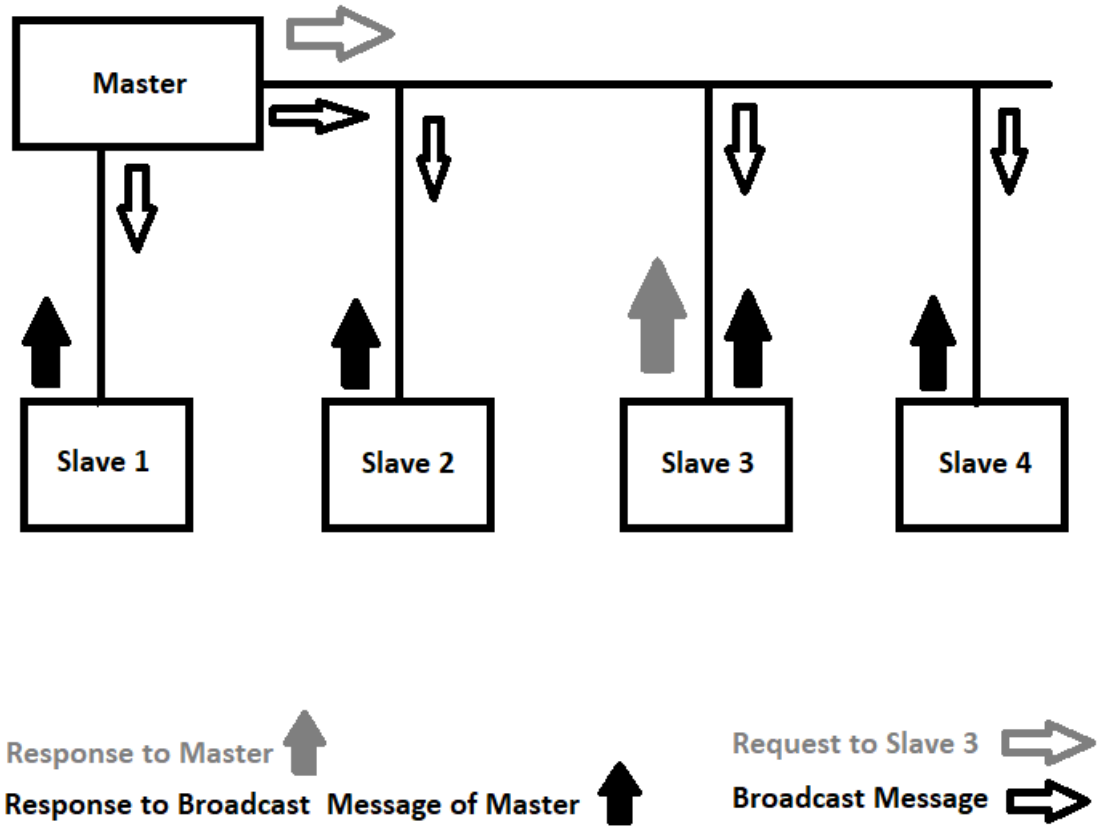
παράδειγμα σε περίπτωσης αιτήματος για απλή ανάγνωση δεδομένων, είτε να περιλαμβάνει ένα σύνολο πληροφοριών οι οποίες θα αποτελέσουν μια νέα προσθήκη ή τις τιμές μιας νέας μεταβολής.

Η απάντηση που αποστέλλεται από τον εξυπηρετητή αποτελείται από τρία μέρη [25]. Το πρώτο κομμάτι αφορά τον κωδικό που δηλώνει την κατάσταση του αιτήματος. Οι καταστάσεις αυτές χωρίζονται σε 5 κατηγορίες [26]. Το εύρος της πρώτης κατηγορίας περιλαμβάνει τους κωδικούς από 100 έως 199 και αποτελεί μια ενημέρωση από τη μεριά του εξυπηρετητή. Η δεύτερη κατηγορία περιλαμβάνει τους κωδικούς από 200 έως 299 και αποτελεί μια επιτυχημένη απάντηση αναφορικά με το αίτημα. Στην τρίτη κατηγορία από 300 έως 399 οι απαντήσεις σχετίζονται με την ανακατεύθυνση του αιτήματος ενώ στην τέταρτη (400-499) και πέμπτη κατηγορία (500-599) εντάσσονται οι κωδικοί που αναφέρονται σε σφάλμα από την μεριά του πελάτη ή του εξυπηρετητή αντίστοιχα. Το δεύτερο κομμάτι της απάντησης περιλαμβάνει όλες τις κεφαλίδες (headers) που αφορούν το αίτημα, ονομάζεται επικεφαλίδα απάντησης (response header) και μπορεί να περιέχει το περιεχόμενο και το μέγεθος του περιεχομένου ενός αιτήματος. Τέλος το τρίτο κομμάτι της απάντησης που είναι το σώμα απάντησης (response body) στο οποίο περιλαμβάνονται όλες οι πληροφορίες του αιτήματος στην μορφή που έχει επιλεχθεί με τις πιο συχνά χρησιμοποιούμενες να είναι οι JSON, XML και HTML.

Προκειμένου να επιτευχθεί η αλληλεπίδραση μεταξύ διαφορετικών στοιχείων και να μην υπάρχει μόνο η επικοινωνία που αφορά την διασύνδεση πελάτη-εξυπηρετητή ενσωματώθηκε η έννοια της μεταφοράς κατάστασης της αναπαράστασης (REpresentational State Transfer - REST) στο HTTP. Σε αυτή την περίπτωση γίνεται αντιστοίχηση των HTTP μεθόδων POST, GET, PUT, DELETE με τις εντολές δημιουργίας (create), ανάγνωσης (read), ανανέωσης (update) και διαγραφής (delete) δεδομένων αντίστοιχα και η παρουσίαση των δεδομένων γίνεται συνήθως με την μορφή JSON ή XML. Για να επιτευχθεί ο στόχος αυτός προστίθεται ένα επιπλέον κομμάτι στον σύνδεσμο του αιτήματος το οποίο δίνει πληροφορίες σχετικά με την πηγή των δεδομένων,με το ποια δεδομένα πρόκειται να επηρεαστούν και σε τι βαθμό.

2.4.4 Modbus

Στις εφαρμογές της βιομηχανίας ακόμα ένα πρωτόκολλο που χρησιμοποιείται για την μεταφορά δεδομένων μεταξύ συσκευών και εφαρμογών είναι το Modbus [27]. Λειτουργώντας στο επίπεδο εφαρμογών, το συγκεκριμένο πρωτόκολλο δημοσιεύτηκε επίσημα το 1979 παρέχοντας επικοινωνία πελάτη (slave) - εξυπηρετητή (master) προσανατολισμένη στα bytes σε δίκτυα πολλαπλών διαδρομών [28]. Τα δίκτυα που σχεδιάστηκαν με την χρήση του συγκεκριμένου πρωτοκόλλου θεωρείται πως παρουσιάζουν μερικές αδυναμίες σχετικά με θέματα που αφορούν την ασφάλεια. Αυτό οφείλεται κυρίως στο ότι κατά την περίοδο που ξεκίνησε εντατικά η εφαρμογή τους δεν υπήρχε η ανάγκη να ληφθεί υπόψιν ο συγκεκριμένος παράγοντας και κατά συνέπεια δεν θεωρήθηκε κρίσιμη η διαχείρισή του. Από την άλλη πλευρά, το κυριότερο πλεονέκτημα που προσφέρει αναφορικά με την χρήση του στις εφαρμογές του βιομηχανικού διαδικτύου των πραγμάτων είναι η άμεση και απευθείας επικοινωνία μεταξύ των μηχανών (επικοινωνία μηχανής προς μηχανή) [29]. Μπορεί ο μεγαλύτερος αριθμός των μηχανών να έχει ήδη ενσωματωμένα συστήματα που συμβάλλουν στην παρακολούθηση τους, όμως με τη χρήση ενός τέτοιου πρωτοκόλλου μπορούν να δημιουργηθούν δίκτυα ώστε να συλλέγεται μια πληθώρα από διαφορετικές πληροφορίες για τη μηχανή και επιπλέον τα δεδομένα αυτά να μπορούν να αποθηκεύονται σε ξεχωριστό περιβάλλον.



Εικόνα 2.5: Παράδειγμα αρχιτεκτονικής του πρωτοκόλλου Modbus

Οι συσκευές και οι εφαρμογές που εμπλέκονται στην επικοινωνία αυτή δεν έχουν την δυνατότητα να βρίσκονται σε διαφορετικούς σταθμούς και έτσι είναι υποχρεωτικό να ανήκουν στο ίδιο δίκτυο. Υπάρχουν δύο τύποι επικοινωνίας που μπορούν να υλοποιηθούν με τη χρήση του Modbus και παρουσιάζονται περιληπτικά στην Εικόνα 2.5. Στην μία περίπτωση υπάρχει ένας κόμβος πελάτη και ένας κόμβος εξυπηρετητή και αναφέρεται ως τύπος ερώτησης/απάντησης. Στην δεύτερη περίπτωση του τύπου της εκπομπής, υπάρχει ένας κόμβος εξυπηρετητή που στέλνει εντολές σε πολλαπλούς κόμβους πελατών. Και στις δύο παραπάνω περιπτώσεις, ακόμα και όταν μια συσκευή που παρακολουθείται έχει μια καινούρια μέτρηση η οποία θεωρείται κρίσιμη δεν μπορεί να ενημερώσει με πρωτοβουλία τον κύριο κόμβο του εξυπηρετητή ξεκινώντας από την μεριά της την επικοινωνία. Συνεπώς η έναρξη ενός μηνύματος και κατά συνέπεια η ενημέρωση του εξυπηρετητή για οποιαδήποτε πληροφορία έχει συλλεχθεί από τις συσκευές μπορεί να πραγματοποιηθεί μονάχα από τον εξυπηρετητή [30].

Η επικοινωνία που μπορεί να πραγματοποιηθεί μέσω του πρωτοκόλλου αυτού παρουσιάζει τρεις διαφορετικές λειτουργίες. Όταν η μετάδοση των δεδομένων πραγματοποιείται σε δυαδική μορφή και κάθε πληροφορία αποτελείται από οχτώ δυαδικά ψηφία γίνεται αναφορά στην λειτουργία Modbus Απομακρυσμένου Τερματικού Σταθμού (Remote Terminal Unit - RTU) [31]. Σε κάθε μήνυμα που αποστέλλεται συμπεριλαμβάνεται ένα σύνολο από πληροφορίες που ορίζουν βασικά στοιχεία της επικοινωνίας η οποία πραγματοποιείται μέσω διεπαφής ethernet. Αρχικά δηλώνεται η συσκευή στην οποία αποστέλλεται το μήνυμα συγκεκριμένα ορίζοντας τη διεύθυνσή της, ο τύπος του μηνύματος ή αλλιώς ο κωδικός λειτουργίας, ο έλεγχος σφαλμάτων και τέλος όλα τα δεδομένα που πρόκειται να σταλούν. Η λειτουργία αυτή

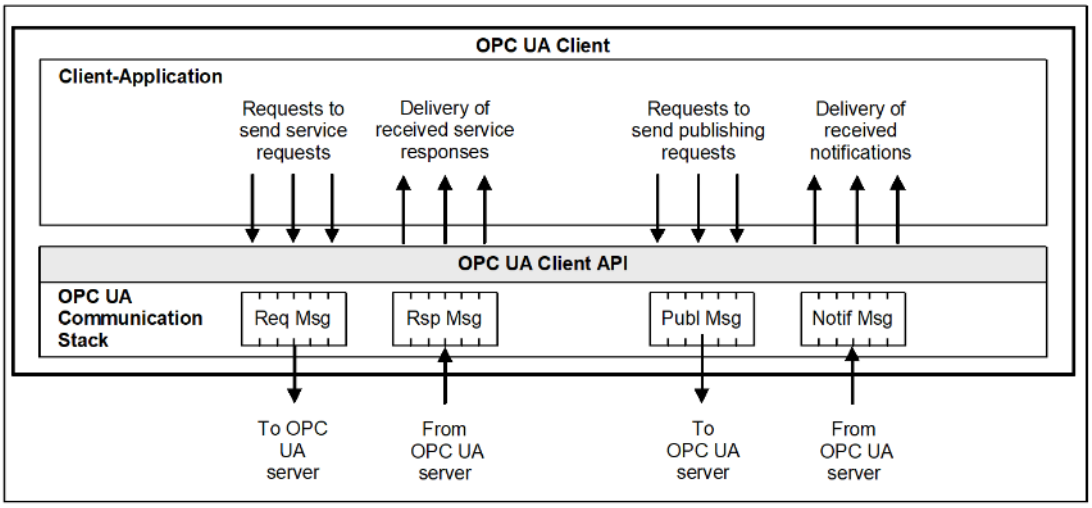
χρησιμοποιείται συχνά στις εφαρμογές του βιομηχανικού διαδικτύου κυρίως επειδή η επικοινωνία γίνεται γρηγορότερα, με μεγαλύτερη ευκολία και το δίκτυο είναι σχεδιαστικά πιο ευέλικτο [32]. Προτιμάται κυρίως να χρησιμοποιείται σε εφαρμογές στις οποίες δεν υπάρχουν σημαντικές απαιτήσεις για μεγάλο εύρος ζώνης αλλά και σε περιπτώσεις που οι ο θόρυβος και οι παρεμβολές είναι συχνά φαινόμενα.

Όπως το Modbus RTU έτσι και ο δεύτερος τύπος επικοινωνίας Modbus με τη χρήση του Πρωτοκόλλου Ελέγχου Μετάδοσης (Transmission Control Protocol - TCP) εκτελείται με τον ίδιο τρόπο, δηλαδή μέσω διεπαφής Ethernet [33]. Ένα μήνυμα που αποστέλλεται με την συγκεκριμένη λειτουργία αποτελείται από την κεφαλίδα της εφαρμογής του Modbus η οποία καταλαμβάνει επτά ψηφία, τον κωδικό λειτουργίας και τα δεδομένα. Στο κομμάτι της κεφαλίδας συμπεριλαμβάνονται πληροφορίες όπως ο κωδικός της μεταφοράς, ο κωδικός του πρωτοκόλλου, το μέγεθος του μηνύματος που αποστέλλεται και ο κωδικός του παραλήπτη. Όσον αφορά την δομή του εφαρμόζει το μοντέλο πελάτη εξυπηρετητή, κατά την επικοινωνία των οποίον το κάθε μήνυμα περνάει από τέσσερα στάδια της αποστολής από τον πελάτη, της επιβεβαίωσης από τον εξυπηρετητή, της αποστολής απάντησης από τον εξυπηρετητή και τελικά την επιβεβαίωση από την μεριά του πελάτη. Με την χρήση της συγκεκριμένης λειτουργίας συσκευές οι οποίες είναι κατασκευασμένες από διαφορετικό κατασκευαστή και ανήκουν σε διαφορετικές κατηγορίες μπορούν να συνδεθούν μεταξύ τους. Έτσι μειώνονται οι περιορισμοί που προκύπτουν πριν την ενσωμάτωση νέων συσκευών σε ένα δίκτυο και συνεπώς το σύστημα γίνεται πιο ευέλικτο με μόνη προϋπόθεσή όλες οι συσκευές να υποστηρίζουν το ίδιο πρωτόκολλο.

Ο τελευταίος τύπος λειτουργίας αντίστοιχα χρησιμοποιεί το δυαδικό σύστημα αναφορικά με την αναπαράσταση του μηνύματος όμως υστερεί στην κατανάλωση χώρου και χρόνου καθώς χρειάζεται διπλάσιους πόρους συγκριτικά με την προηγούμενη λειτουργία. Οι πληροφορίες που περιλαμβάνει το κάθε μήνυμα είναι ακριβώς οι ίδιες με αυτές που περιγράφηκαν και στην λειτουργία RTU και ανταλλάσσονται κατά τη διάρκεια της επικοινωνίας σε μορφή ASCII. Για κάθε πλαίσιο πληροφοριών χρησιμοποιούνται δύο χαρακτήρες και συνεπώς χρησιμοποιούνται δύο δυαδικά στοιχεία για να το αναπαραστήσουν. Το κυριότερο πλεονέκτημα του είναι πως χάρης σε αυτή την δεκαεξαδική αναπαράσταση που χρησιμοποιεί καθιστά ευκολότερη και πιο ξεκάθαρη την ανάγνωση.

2.4.5 OPC UA

Το πρότυπο Ανοιχτής Πλατφόρμας Επικοινωνιών Ενοποιημένης Αρχιτεκτονικής (Open Platform Comm unications Unified Architecture - OPC UA) πρωτοπαρουσιάστηκε από την OPC Foundations [34] και επικεντρώνεται στην επικοινωνία μηχανής προς μηχανή. Ενώ αρχικά αποτελούσε λύση κυρίως για τις εφαρμογές στη βιομηχανία που σχετίζονται με την αυτοματοποίηση στην πορεία η χρήση του διευρύνεται και σε άλλους τομείς. Αν και χρησιμοποιεί μια ιδιαίτερα περίπλοκη δομή λογισμικού ώστε να επιτυγχάνεται η επικοινωνία μεταξύ των στοιχείων της βιομηχανίας αναγνωρίζεται ως μια μεγάλη εξέλιξη συγκριτικά με τις προηγούμενες τεχνολογίες ανοιχτής πλατφόρμας που είχαν προταθεί. Μερικοί από τους στόχους της ανάπτυξης του αφορούν στην διευκόλυνση της ενσωμάτωσης μεταξύ των συστημάτων αλλά και στην διεύρυνση των δυνατοτήτων που αφορούν την επικοινωνία των συσκευών στο περιβάλλον της βιομηχανίας. Αποτελεί λοιπόν μια λύση στην εγκαθίδρυση μιας κοινής υποδομής έτσι ώστε να επιτυγχάνεται ο έλεγχος των βιομηχανικών διεργασιών αλλά και η ανταλλαγή πληροφοριών.



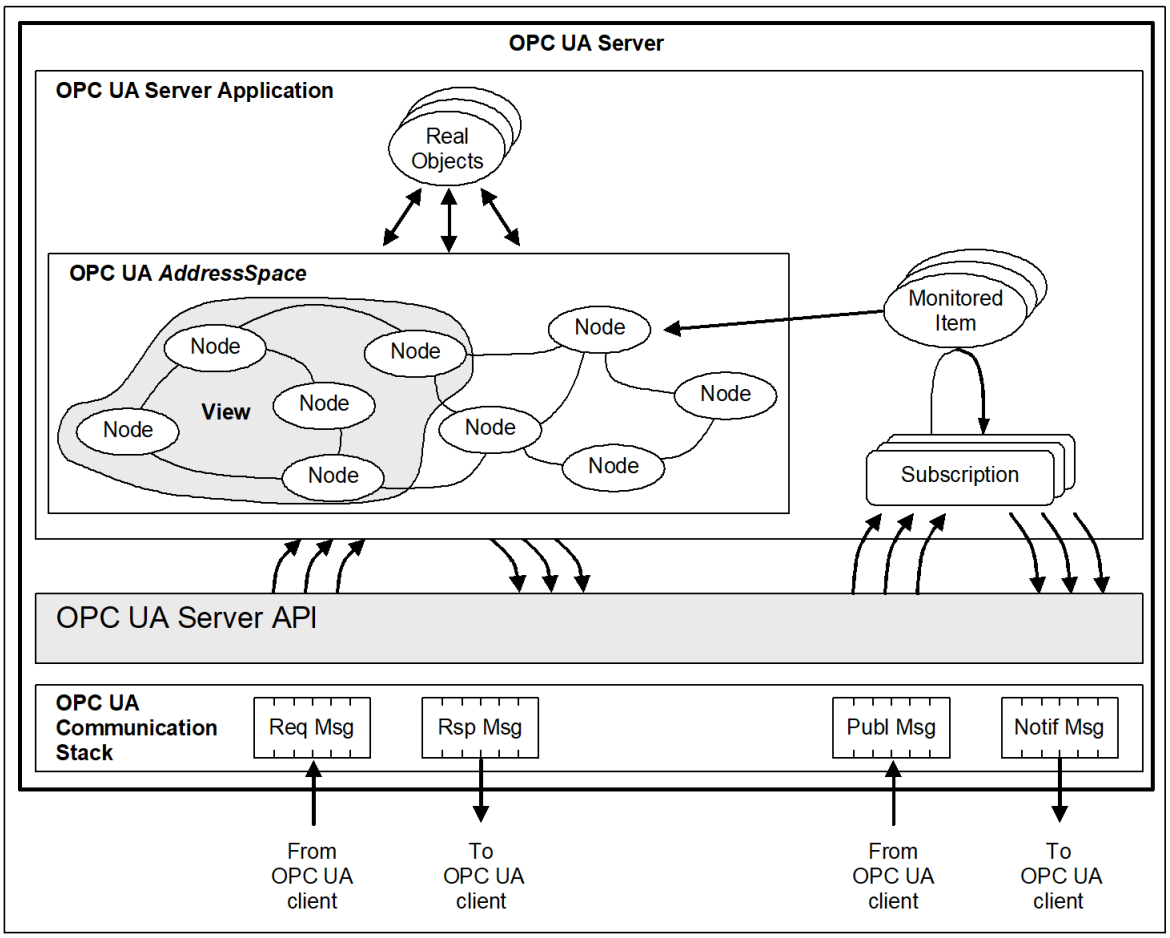
Εικόνα 2.6: Αρχιτεκτονική Πελάτη OPC-UA [35]

Το κομμάτι της αρχιτεκτονικής του OPC UA που αφορά τον πελάτη παρουσιάζεται στην Εικόνα 2.6. Η υλοποίηση του πελάτη με την χρήση της στοίβας επικοινωνίας είναι το μοντέλο που χρησιμοποιείται και η βάση στην οποία σχηματίζεται ο πελάτης στο OPC UA. Το κομμάτι της εφαρμογής του πελάτη είναι υπεύθυνο για την προώθηση των μηνυμάτων στον εξυπηρετητή. Η δημιουργία των μηνυμάτων αυτών μπορεί να βασίζεται στις προδιαγραφές της κάθε υπηρεσίας [34]. Μπορεί επίσης να ακολουθεί στην πιο απλουστευμένη μορφή της την διαδικασία της ανάγνωσης και εγγραφής όπου δίνεται η δυνατότητα για μεταβολή ενός η περισσοτέρων τιμών που βρίσκονται στους κόμβους του εξυπηρετητή. Μια πιο περίπλοκη εκδοχή είναι αυτή των εγγραφών που δημιουργούν το πλαίσιο για ανταλλαγή τιμών που αφορούν σε μεταβολές δεδομένων και των αντικειμένων παρακολούθησης, τα οποία δημιουργούνται από την πλευρά του πελάτη με στόχο τον έλεγχο των κόμβων του εξυπηρετητή.

Το κομμάτι της εφαρμογής του πελάτη επικοινωνεί με την μεριά του εξυπηρετητή χρησιμοποιώντας την διεπαφή προγραμματισμού εφαρμογών του πελάτη OPC UA. Ο πελάτης χρησιμοποιεί την διεπαφή αυτή με στόχο την αποστολή ή ανάγνωση δεδομένων που σχετίζονται με τις υπηρεσίες OPC UA. Αποτελεί ένα στρώμα μεσολάβησης ανάμεσα στην εφαρμογή του πελάτη και στην στοίβα επικοινωνίας καταφέρνοντας με αυτόν τον τρόπο από την μια τα δύο περιβάλλοντα να είναι απομονωμένα όμως από την άλλη να μπορούν να επικοινωνούν μεταξύ τους. Στη συνέχεια τα αιτήματα που έχουν πραγματοποιηθεί αφού περάσουν στην στοίβα επικοινωνίας θα μετατραπούν σε μηνύματα τα οποία πλέον μπορούν να αποσταλούν στο περιβάλλον του εξυπηρετητή. Ένας ακόμα ρόλος του στρώματος της στοίβας επικοινωνίας είναι να προωθεί αντίστοιχα στο περιβάλλον του πελάτη το σύνολο των μηνυμάτων αλλά και ειδοποιήσεων που τον αφορούν και τα οποία λαμβάνει από κατώτερα στρώματα επικοινωνίας. Ένα από τα πλεονεκτήματα που παρουσιάζει το στρώμα αυτό είναι πως δεν συνδέεται ή δεσμεύεται στενά με κάποια τεχνολογία και έτσι είναι πολύ ευέλικτο στην χρήση και αξιοποίηση νέων και διαφορετικών τεχνολογιών [34].

Αφού το μήνυμα φθάσει στην πλευρά του εξυπηρετητή ο στόχος είναι αφού αυτό αναλυθεί να επιστραφεί στην μεριά του πελάτη πάλι με την μεσολάβηση της διεπαφής OPC UA από τη μεριά του εξυπηρετητή η απάντηση που αντιστοιχεί. Αυτή τη φορά το μήνυμα λαμβάνεται στο κομμάτι της στοίβας του εξυπηρετητή. Ανάλογα με τον τρόπο που έχει υλοποιηθεί η πλευρά του εξυπηρετητή επιλέγει το αντικείμενο από το οποίο θα αντλήσει τις πληροφορίες που απαιτούνται. Πιο συγκεκριμένα ο εξυπηρετητής έχει τη δυνα-

τότητα να αλληλεπιδρά με ολόκληρα συστήματα, βάσεις δεδομένων έτσι ώστε να μπορέσει να ανακτήσει τις ανάλογες πληροφορίες ώστε να μπορέσει στην συνέχεια να ενημερώσει το κομμάτι των διευθύνσεων. Το κομμάτι των διευθύνσεων αποτελείται από ένα σύνολο κόμβων οι οποίοι μπορούν να είναι ίδιου ή διαφορετικού τύπου και συνδέονται μεταξύ τους χρησιμοποιώντας αναφορές [36]. Το κάθε στοιχείο ή κόμβος αποτελείται από ένα πλήθος γνωρισμάτων όπως συμβαίνει και στην περίπτωση των κλάσεων. Στην περίπτωση αυτή το κομμάτι της εφαρμογής είναι υπεύθυνο εκτός από την διαχείριση των μηνυμάτων που λαμβάνονται και για την οργάνωση της επικοινωνίας μεταξύ των στοιχείων της αρχιτεκτονικής. Όλη η δομή από την πλευρά του εξυπηρετητή απεικονίζεται παρακάτω στην Εικόνα 2.7.



Εικόνα 2.7: Αρχιτεκτονική Εξυπηρετητή OPC-UA [35]

Κεφάλαιο 3ο: Τεχνολογίες του βιομηχανικού διαδικτύου των πραγμάτων

Στο κεφάλαιο αυτό περιλαμβάνονται όλες οι τεχνολογίες που εφαρμόζονται στην υλοποίηση του διαδικτύου των πραγμάτων στην βιομηχανία και χρησιμοποιήθηκαν στο πρακτικό κομμάτι της εργασίας. Στο σύνολο τους όλα τα παρακάτω εργαλεία και κατηγορίες όχι μόνο μπορούν να πλαισιώσουν ένα μεγάλο εύρος εφαρμογών στο περιβάλλον της βιομηχανίας αλλά κυρίως μπορούν να εξελίξουν και να βελτιώσουν σενάρια που ήδη έχουν υλοποιηθεί. Κύριος στόχος του κεφαλαίου αυτού είναι να καλύψει όλο το θεωρητικό πλαίσιο γύρω από το σενάριο υλοποίησης της εργασίας αλλά και να υπογραμμίσει την σημαντικότητα της επιλογής ενός εργαλείου αναφορικά με την ποιότητα και την βελτιστοποίηση μιας εφαρμογής.

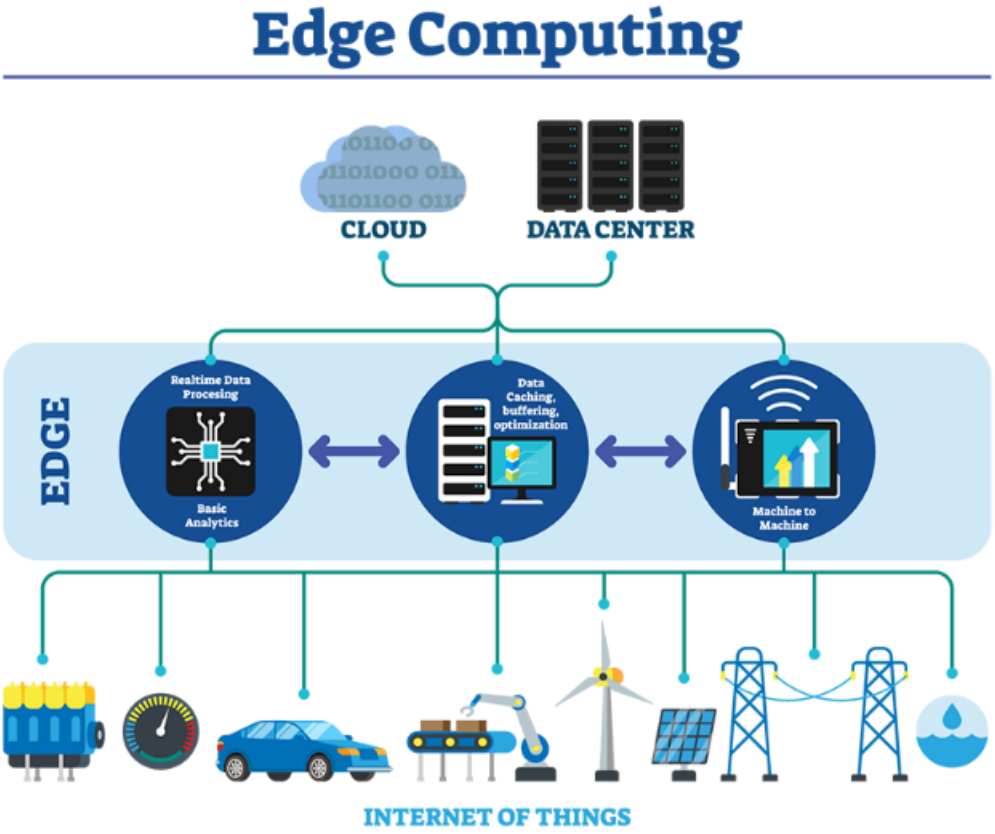
3.1 Υπολογιστική στα άκρα του δικτύου

Αν και η υπολογιστική νέφους έχει συμβάλλει σημαντικά στην εξέλιξη και αύξηση ποιότητας της λειτουργίας των συστημάτων και των τομέων της καθημερινότητας [37], παρουσιάζει μερικά μειονεκτήματα. Πιο συγκεκριμένα η ανάγκη για κάλυψη των απαιτήσεων που παρουσιάζουν οι εφαρμογές του διαδικτύου των πραγμάτων σε όλους τους τομείς δημιουργεί εκ νέου προκλήσεις και αναδιαμορφώσεις που αφορούν τις λύσεις που σχετίζονται με την εφαρμογή του Υπολογιστικού Νέφους (Cloud Computing) [38]. Με την δομή του υπολογιστικού νέφους όπου κύριο στοιχείο αποτελεί η κεντρική και κατανεμημένη διαχείριση όσον αφορά κυρίως τις ενέργειες που σχετίζονται με τους υπολογισμούς και την αποθήκευση δεδομένων, δημιουργούνται αρκετές δυσκολίες.

Συγκεκριμένα τα κύρια εμπόδια που προκύπτουν οφείλονται κατ' εξοχήν στην συνεχή αύξηση των συσκευών και των δεδομένων που παράγονται με μεγαλύτερη συχνότητα και σε μεγαλύτερη ποσότητα. Τα δεδομένα αυτά καθώς όλα πρέπει να επεξεργαστούν και να αποθηκευτούν με τον καταλληλότερο τρόπο δημιουργούν την ανάγκη για περισσότερους διαθέσιμους πόρους καθώς δεν είναι δυνατόν να είναι αποτελεσματικά διαχειρίσιμα από μια μόνο κεντρική τεχνολογία. Η παραπάνω συνθήκη γίνεται ακόμα πιο κρίσιμη όσον αφορά τις εφαρμογές του διαδικτύου των πραγμάτων γενικά αλλά και συγκεκριμένα στην βιομηχανία όπου η παρακολούθηση και παραγωγή αποτελεσμάτων σε πραγματικό χρόνο παρουσιάζει προϋπόθεση ομαλής λειτουργίας και ορθότερων συμπερασμάτων. Η απάντηση στις προκλήσεις αυτές δίνεται με την εφαρμογή της δομής της υπολογιστικής στα άκρα του δικτύου (Edge Computing).

3.1.1 Επεξήγηση έννοιας

Μέχρι πριν την εξέλιξη λειτουργίας της υπολογιστικής στα άκρα του δικτύου, το άκρο (Edge) ήταν υπεύθυνο ώστε οι συσκευές να μπορούν να συνδέονται σε αυτό, να στέλνουν δεδομένα και να δέχονται οδηγίες και εντολές που προέρχονται κυρίως από το Cloud ή από έναν κεντρικό εξυπηρετητή (server) τοποθετημένο σε κάποιο κέντρο δεδομένων (data center). Με την υπολογιστική άκρων αναφερόμαστε σε ένα σύνολο τεχνολογιών οι οποίες έχουν ως στόχο την αποτελεσματικότερη επεξεργασία και αποθήκευση δεδομένων [39]. Η λογική που ικανοποιεί τον στόχο αυτό, είναι οι διαδικασίες που αφορούν τα δεδομένα να βρίσκονται όσο το δυνατόν πιο κοντά στην πηγή που τα παράγει. Πιο συγκεκριμένα, ο υπολογισμός, η αποθήκευση δεδομένων και οι διαθέσιμοι πόροι μεταφέρονται στα άκρα του δικτύου σε κατάλληλες συσκευές άκρων (edge devices), δηλαδή στην περιοχή όπου οι τελικές συσκευές (end devices) όπως



Εικόνα 3.1: Υπολογισμός στα άκρα του δικτύου (Edge Computing) [40]

αισθητήρες και βιομηχανικά ρομπότ αποκτούν πρόσβαση με τα υπόλοιπα μέρη του δικτύου. Έτσι, το Edge Computing τελικά λειτουργεί ως ενδιάμεσος κόμβος μεταξύ των συσκευών-αισθητήρων και του υπολογιστικού νέφους όχι μόνο για την επικοινωνία αλλά και για την εξυπηρέτηση ενός σημαντικού μέρους του συνόλου των λειτουργιών. Με λίγα λόγια, τα δεδομένα παράγονται από τις συσκευές, περνάνε με τη σειρά τους στις συσκευές που βρίσκονται στα άκρα του δικτύου (edge devices) και έπειτα τα αποτελέσματα που προκύπτουν από την ανάλυση τους αποστέλλονται στο Cloud. Στην Εικόνα 3.1 απεικονίζεται η γενική αρχιτεκτονική ενός περιβάλλοντος που συμπεριλαμβάνει το Edge. Το πρώτο επίπεδο συντελείται από όλες τις συσκευές και μηχανές από τις οποίες παράγονται τα δεδομένα. Στη συνέχεια στο αμέσως επόμενο επίπεδο το οποίο επικοινωνεί απευθείας με το σύνολο των συσκευών βρίσκονται όλες οι τεχνολογίες που πλαισιώνουν το Edge Computing το οποίο λειτουργεί ως ενδιάμεσο επίπεδο πλέον μεταξύ των συσκευών και του υπολογιστικού νέφους και των κέντρων δεδομένων αντίστοιχα.

Η δομή της υπολογιστικής στα άκρα του δικτύου εξυπηρετεί σε μεγάλο βαθμό την έννοια της ετερογένειας [41]. Πιο συγκεκριμένα, παρέχεται ένα σύνολο από πλατφόρμες οι οποίες εκτελούνται στο άκρο του δικτύου και μέσω των οποίων όλες οι λειτουργίες που αφορούν το σύστημα μπορούν να εκτελεστούν με μεγαλύτερη ευκολία και να αντιμετωπίσουν διαφορετικού είδους περιβάλλοντα και απαιτήσεις. Επίσης γίνεται χρήση τεχνολογιών που αφορούν τόσο το υπολογιστικό όσο και το επικοινωνιακό κομμάτι και χρησιμοποιούνται από τα στοιχεία που συντελούν το Edge Computing. Ένα από τα σημαντικότερα στοιχεία είναι οι συσκευές άκρου (edge devices) οι οποίες εκτελούν όλες τις λειτουργίες που σχετίζονται με τα δεδομένα και περιγράφηκαν παραπάνω. Τέλος, η αρχιτεκτονική και ο τρόπος διαμόρφωσης συστημάτων στα οποία εφαρμόζεται εξαρτώνται από το σενάριο που πρόκειται να υλοποιηθεί και από

τις προκλήσεις που αυτό παρουσιάζει. Με τον τρόπο με τον οποίο υλοποιείται η έννοια της υπολογιστικής στο άκρο του δικτύου δημιουργείται μια πιο κατανεμημένη μορφή της διαχείρισης των δεδομένων η οποία παρουσιάζει μια σειρά πλεονεκτημάτων.

3.1.2 Πλεονεκτήματα της υπολογιστικής στα άκρα του δικτύου

Ένα από τα κυριότερα πλεονεκτήματα αφορά στην βελτίωση της αποδοτικότητας. Από την στιγμή που οι υπολογιστικές διαδικασίες πραγματοποιούνται κοντά στην πηγή, η αλληλεπίδραση γίνεται γρηγορότερη. Με τον τρόπο αυτόν, η αποστολή των τιμών που παράγονται γίνεται πλέον σε πραγματικό χρόνο και εφιστά δυνατόν να γίνει με μεγαλύτερη συχνότητα η παρακολούθηση ενός συστήματος [39]. Έτσι, εφόσον η ανάλυση και η αξιολόγηση των δεδομένων είναι πιο πλήρης τα συμπεράσματα που προκύπτουν είναι πιο σαφή και έχουν μεγαλύτερη αξία καθώς μπορεί να υπάρχει άμεση ανταπόκριση για την οποιαδήποτε αποφυγή σοβαρών επιπτώσεων. Για παράδειγμα, σε μια περίπτωση όπου γίνεται μέτρηση της πίεσης του αέρα σε έναν σωλήνα και εντοπιστεί από τους αισθητήρες μια τιμή που θεωρείται επικίνδυνη η απόφαση για την αποφυγή διαρροής μπορεί να παρθεί σε ένα χρονικό διάστημα που απέχει όσο το δυνατόν λιγότερο από τη στιγμή της μέτρησης.

Η αμεσότητα στην επικοινωνία συνδέεται επίσης με την ομαλότερη λειτουργία του δικτύου. Καθώς στα άκρα του δικτύου κατανέμεται ένα μεγάλο ποσοστό της κίνησης και μεταφοράς πληροφοριών η καθυστέρηση στο συνολικό δίκτυο μειώνεται και το εύρος ζώνης του δικτύου ελαχιστοποιείται. Κατά συνέπεια καθώς δεν είναι πλέον απαραίτητο όλα τα δεδομένα που παράγονται να περάσουν στο υπολογιστικό νέφος, το κόστος για μεγαλύτερο εύρος ζώνης και κατανάλωση ενέργειας μειώνεται δραματικά [39]. Από την άλλη πλευρά. αποτελέσματά της μείωσης καθυστέρησης στο δίκτυο είναι η παροχή καλύτερης ποιότητας υπηρεσιών στους χρήστες του συστήματος αλλά και η αποφυγή προβλημάτων που σχετίζονται με την διαχείριση πόρων ενός δικτύου που τελικά συμβάλλει στην μεγαλύτερη απόδοση του συστήματος.

Εκτός από τον διαμοιρασμό των πόρων του δικτύου, κατανέμεται επίσης και ο χώρος αποθήκευσης. Τα δεδομένα που συλλέγονται στα άκρα του δικτύου όχι μόνο επεξεργάζονται αλλά μπορούν επίσης και να δεσμεύονται ώστε να αποθηκεύονται προσωρινά. Το γεγονός αυτό προσδίδει μεγαλύτερη ευκολία στην οργάνωση και διαχείριση των δεδομένων καθώς αυτά είναι πλέον πιο ταξινομημένα. Επίσης, αποδεσμεύει πόρους που σχετίζονται με την αποθήκευση στο υπολογιστικό νέφος καθώς αποσαφηνίζει ποίες πληροφορίες δεν είναι χρήσιμες και δεν είναι απαραίτητο να μεταβούν στο υπολογιστικό νέφος η ακόμα και ποιες έχουν την μεγαλύτερη προτεραιότητα. Με αυτόν τον τρόπο ενισχύονται οι πιθανότητες για επεκτασιμότητα του συστήματος [38].

Τέλος σημαντική βελτίωση παρατηρείται ακόμα και στον τομέα της ασφάλειας. Στην περίπτωση της υπολογιστικής νέφους όπου όλες οι πληροφορίες μεταφέρονται στο υπολογιστικό νέφος, υπάρχει κίνδυνος να χαθούν ή ακόμα και να διαρρεύσουν σημαντικά δεδομένα και ευαίσθητες πληροφορίες [39]. Αντίθετα στην περίπτωση της υπολογιστικής των άκρων καθώς τα δεδομένα μένουν πολύ κοντά στο ίδιο δίκτυο ή μέσα στο ίδιο δίκτυο, οι πιθανότητες απώλειας ή διαρροής τους μειώνονται σημαντικά. Επιπλέον χάρης τους διαφορετικούς κόμβους που παρέχονται στο Edge προσδίδεται επιπρόσθετη απομόνωση των δεδομένων αλλά και αυξημένη ιδιωτικότητα. Έτσι, ακόμα και στην περίπτωση όπου ένας κόμβος δεχτεί επίθεση, οι συνέπειες που θα ακολουθήσουν αφορούν αποκλειστικά και μόνο τον συγκεκριμένο κόμβο χωρίς να επηρεάζονται ανεξέλεγκτα και τα υπόλοιπα στοιχεία.

3.1.3 Υπολογιστική άκρων και βιομηχανικό διαδίκτυο

Το περιβάλλον της βιομηχανίας ενισχύεται σημαντικά με την είσοδο των τεχνολογιών και των εξελίξεων που συμπεριλαμβάνονται στην έννοια της υπολογιστικής των άκρων [14]. Πιο συγκεκριμένα, η επίδοση της λειτουργίας του βιομηχανικού διαδικτύου μεγαλώνει καθώς ο αριθμός των στοιχείων που το συντελούν δεν είναι παράγοντας που επηρεάζει την ομαλή λειτουργία. Ακόμα, οι συσκευές και όλες οι οντότητες μπορούν να αντεπεξέλθουν ακόμα περισσότερο στις απαιτήσεις που παρουσιάζονται από τις εφαρμογές. Σχεδόν σε όλες τις εφαρμογές που αφορούν την βιομηχανία, απαιτείται συνεχής παρακολούθηση του συστήματος. Με την υπολογιστική στα άκρα του δικτύου κάτι τέτοιο επιτυγχάνεται με την εγκατάσταση συσκευών στο άκρο, οι οποίες είναι υπεύθυνες να διαχειριστούν και να προ-επεξεργαστούν διαφορετικής μορφής δεδομένα που προέρχονται από διαφορετικού τύπου μηχανές και τελικά να λάβουν αποφάσεις που αφορούν το αντίστοιχο σύστημα σε τοπικό επίπεδο. Οι αποφάσεις αυτές στέλνονται έπειτα στο υπολογιστικό νέφος με την χρήση ειδικών πρωτοκόλλων επικοινωνίας.

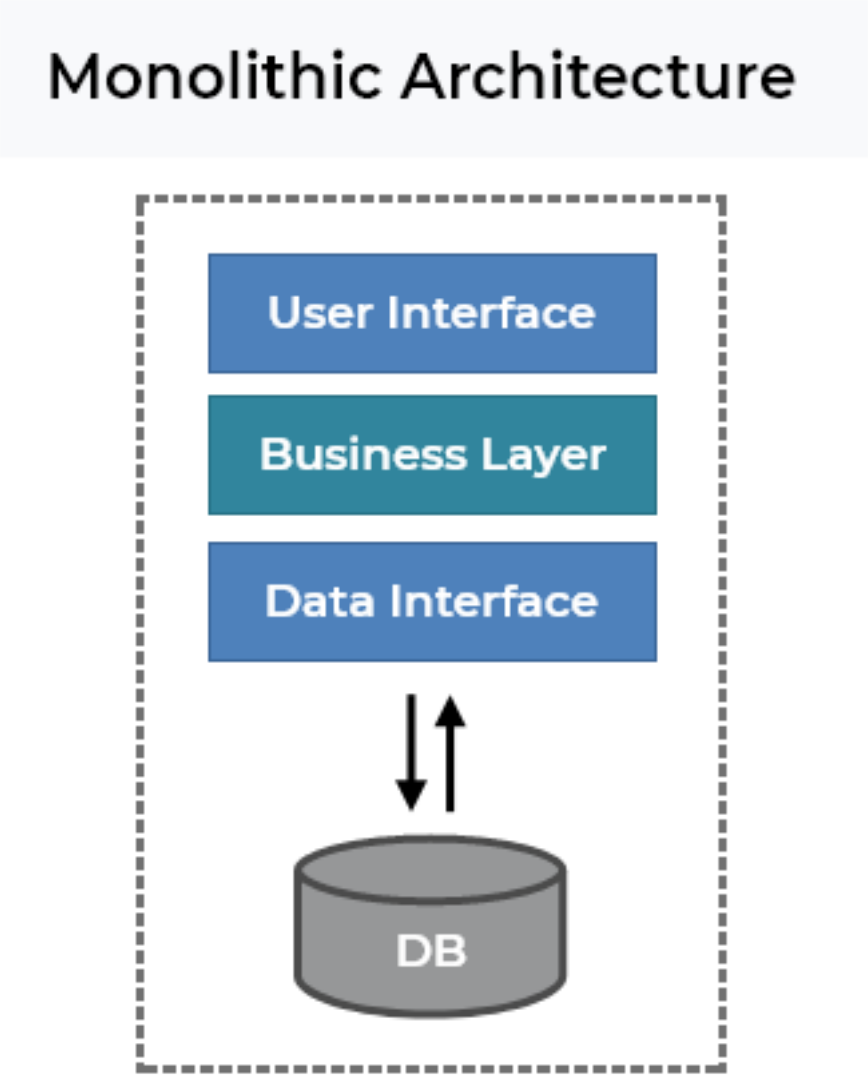
Επιπλέον στην βιομηχανία υπάρχει συχνά η απαίτηση τα συστήματα και οι εφαρμογές να ανανεώνονται ή ακόμα και να αναδιαμορφώνονται καθώς η αλληλεπίδραση των στοιχείων αλλά και ο συνολικός τρόπος λειτουργίας του συστήματος εξαρτάται άμεσα από το σενάριο της εφαρμογής που πρόκειται να υλοποιηθεί. Το πλεονέκτημα που προσφέρει η υπολογιστική άκρων σε αυτό το κομμάτι είναι πολύ σημαντικό, αφού εξασφαλίζει την δυνατότητα για επέκταση ή και ανανέωση του συστήματος ανά πάσα στιγμή και με μεγάλη ευκολία. Καθώς η δομή του δεν είναι στατική και υπάρχει μεγάλη κατανομή εργασιών και πόρων, η οποιαδήποτε νέα αναβάθμιση δεν απαιτεί ολοκληρωτική αναδόμηση του συστήματος ούτε αλλαγές που αφορούν το δομικό κομμάτι. Αντιθέτως όλες οι αλλαγές ανανεώνουν το σύστημα με την μορφή αναβαθμίσεων λογισμικού κάτι που επιτρέπει επίσης την μαζική αναβάθμιση του μεγαλύτερου μέρους των συσκευών από τις οποίες αποτελείται το δίκτυο. Συνοπτικά, με την συνδυαστική εφαρμογή της υπολογιστικής στο άκρο του δικτύου και του βιομηχανικού διαδικτύου, παρουσιάζεται μεγαλύτερη ευελιξία στην συχνότητα αναβάθμισης συστημάτων της βιομηχανίας.

3.2 Μικροϋπηρεσίες (Microservices)

Μια από τις πιο συχνές μεταβολές που εφαρμόζονται στον τομέα της τεχνολογίας είναι αυτή της μετάβασης από την κλασική μονολιθική αρχιτεκτονική σε αυτή των μικροϋπηρεσιών. Έτσι, όλο και περισσότερες εταιρείες και τομείς αποφασίζουν να αναδιαμορφώνουν την δομή των εφαρμογών τους ώστε να μπορέσουν να επωφεληθούν από τα πλεονεκτήματα που παρουσιάζονται αλλά και να εξελίσσονται τεχνολογικά. Με την μονολιθική αρχιτεκτονική που μέχρι τώρα επικρατούσε, η κάθε εφαρμογή εκτελεί το σύνολο των λειτουργιών που πρέπει να εφαρμοστούν ώστε να λειτουργήσει το σύστημα. Με τον τρόπο αυτό όμως το σύστημα δεν παρουσιάζει ευελιξία στην εξέλιξη του και επίσης λόγω των αυστηρά εξαρτώμενων σχέσεων που συνδέουν τις λειτουργίες της εφαρμογής είναι ιδιαίτερα ευάλωτο σε οποιοδήποτε πρόβλημα προκύψει. Αυτό συμβαίνει καθώς η βλάβη μιας μεθόδου που εκτελείται μπορεί να καταλήξει σε διακοπή και παράλυση της λειτουργίας ολόκληρης της εφαρμογής. Στο κεφάλαιο αυτό λοιπόν παρουσιάζονται αναλυτικά και τα δύο είδη αρχιτεκτονικών που πλαισιώνουν τα συστήματα με μεγαλύτερη βάση στην μετάβαση από την μονολιθική σχεδίαση στην υλοποίηση των μικροϋπηρεσιών και της σημασίας τους στην πιο ομαλή και οργανωμένη ενσωμάτωσή τους στην παραγωγή εφαρμογών.

3.2.1 Μονολιθική αρχιτεκτονική εφαρμογών

Σε μια εφαρμογή η οποία ακολουθεί την λογική της μονολιθικής αρχιτεκτονικής κατά την διάρκεια της υλοποίησης της συμπεριλαμβάνονται όλες οι λειτουργίες τις οποίες επρόκειτο να εκτελέσει. Όλη η λογική στην οποία συμπεριλαμβάνονται οι λειτουργίες αυτές, είναι γραμμένη σε μια συγκεκριμένη γλώσσα προγραμματισμού και για κάθε λειτουργία χρησιμοποιούνται διαφορετικά πακέτα τα οποία παρέχονται αντίστοιχα ανάλογα με το αποτέλεσμα που χρειάζεται να προκύψει. Θα μπορούσαμε να πούμε πως για την δημιουργία της παραπάνω λογικής δεν είναι απαραίτητο να υπάρχει κάποιο είδος αρχιτεκτονικής το οποίο να απαιτεί μεγάλη οργάνωση και σχεδίαση. Αυτό συμβαίνει καθώς τα στοιχεία δεν είναι διασκορπισμένα σε διαφορετικούς κόμβους και δεν υπάρχουν διαφορετικοί τρόποι με τους οποίους τα στοιχεία αυτά θα πρέπει να ταξινομηθούν και να καταταχθούν ώστε να μπορούν να επικοινωνούν μεταξύ τους και να ανταλλάσσουν πληροφορίες.



Εικόνα 3.2: Αρχιτεκτονική των μονολιθικών εφαρμογών [42]

Τυπικά μια μονολιθική εφαρμογή όπως παρουσιάζεται και στην Εικόνα 3.2 αποτελείται από τρία επίπεδα, της διεπαφής, της επιχειρησιακής λογικής, και της αλληλεπίδρασης και διεπαφής δεδομένων το οποίο επικοινωνεί με την βάση δεδομένων [43]. Στο κομμάτι της διεπαφής χρήστη συμπεριλαμβάνονται όλες οι λειτουργίες που σχετίζονται με την εικόνα και την εμφάνιση της εφαρμογής και την αλληλεπίδραση που θα έχει με τον χρήστη. Μερικοί από τους τύπους που σχετίζονται με την διεπαφή χρήστη είναι οι παρακάτω: γραφική διεπαφή χρήστη που αφορά όλα τα γραφικά στοιχεία που εμφανίζονται στην οθόνη του και τον βοηθούν να αλληλεπιδράσει με την εφαρμογή, διεπαφή γραμμής εντολών που δίνει την δυνατότητα αλληλεπίδρασης με την εφαρμογή μέσα από εντολές και ένα σύνολο από διεπαφές μέσω των οποίων γίνεται χρήση της φωνής, αφής και φυσικής γλώσσας. Στη συνέχεια το επίπεδο της επιχειρησιακής λογικής συμπεριλαμβάνει την συνολική κωδικοποίηση όλων των κανόνων πραγματικής λογικής αλλά και όλων των λειτουργιών οι οποίες ορίζουν τον τρόπο λειτουργίας της εφαρμογής αλλά και όλες τις διαδικασίες που σχετίζονται με τα δεδομένα. Αφού υπάρχουν όλες οι λειτουργίες ενσωματωμένες στο σώμα της εφαρμογής στο τέλος παράγεται ένα μεμονωμένο τελικό αρχείο. Τέλος το επίπεδο που βρίσκεται στο χαμηλότερο στρώμα, εκτελεί την απαραίτητη επικοινωνία με την βάση δεδομένων και όλες τις διαδικασίες αλληλεπίδρασης. Έτσι τελικά η εφαρμογή εκτελείται ως μια μεμονωμένη διαδικασία.

3.2.2 Πλεονεκτήματα μονολιθικής αρχιτεκτονικής

Μια μονολιθική εφαρμογή στα αρχικά στάδια ανάπτυξης της θεωρείται μια αρκετά απλή και γρήγορα εφαρμόσιμη διαδικασία. Κάτι τέτοιο προσδίδει ένα σημαντικό πλεονέκτημα σε μικρές εταιρείες ή ακόμα και ομάδες οι οποίες ως στόχο έχουν την ανάπτυξη μιας μικρής εφαρμογής που δεν έχει σκοπό να γίνει προϊόν για μεγάλες υλοποιήσεις που απαιτούν συχνή εξέλιξη. Επιπλέον, στο κομμάτι του ελέγχου μιας μονολιθικής εφαρμογής απαιτείται μια μεμονωμένη διαδικασία. Τέλος, το γεγονός ότι ολόκληρη η εφαρμογή βρίσκεται σε ένα μόνο μέρος προσφέρει εξοικονόμηση σε διαδικασίες που αφορούν την δομή και ανάπτυξη της. Τα παραπάνω πλεονεκτήματα παρατηρούνται κυρίως στα αρχικά στάδια μιας εφαρμογής αλλά και σε περιπτώσεις όπου η ομάδα η οποία είναι υπεύθυνη για την υλοποίηση της αποτελείται από μικρό αριθμό ατόμων, όλα τα μέλη της βρίσκονται στο ίδιο σημείο γεωγραφικά και έχουν πρόσβαση στο μέρος που είναι αποθηκευμένη και τρέχει η εφαρμογή και τέλος τόσο ο κώδικας όσο και η βάση του δεν είναι εκτενείς.

3.2.3 Μειονεκτήματα μονολιθικής αρχιτεκτονικής

Παρ' όλο που η αρχή των περισσότερων πρότζεκτ ξεκινάει εφαρμόζοντας την λογική της μονολιθικής αρχιτεκτονικής στην πορεία προκύπτουν αρκετά προβλήματα. Αρχικά στην πορεία το μέγεθος της κάθε εφαρμογής αυξάνεται ολοένα και περισσότερο καθώς αυξάνονται και οι απαιτήσεις τις οποίες πρέπει να ικανοποιήσει. Μια τέτοια ενέργεια γίνεται ολοένα και πιο δύσκολη καθώς η αλλαγή και προσθήκη χαρακτηριστικών μπορούν να επηρεάσουν ολόκληρη την εφαρμογή και να χρειάζεται να πραγματοποιηθούν διαδικασίες από την αρχή. Η δυσκολία που αποκτά η διαδικασία απλών μεταβολών στις λειτουργίες της εφαρμογής οδηγεί στον φόβο για μεγάλες νέες αλλαγές καθώς δεν υπάρχει μεγάλη επίγνωση για το πόσο και το πως θα επηρεαστεί το υπόλοιπο σύστημα. Η αύξηση του όγκου της εφαρμογής παρουσιάζει ακόμα δυο μειονεκτήματα. Συγκεκριμένα, η πολυπλοκότητα ανάμεσα στις αυστηρές σχέσεις που ενώνουν τα στοιχεία της εφαρμογής αυξάνεται και ο έλεγχος και η διαχείριση της ολοένα και δυσκολεύει. Σε περίπτωση που είναι αναγκαία μια διόρθωση συγκεκριμένης λειτουργίας που εκτελεί η εφαρμογή είναι

δυνατόν τόσο ο έλεγχος που εκτελείται για την εύρεση της όσο και η διαδικασία επίλυσης της απαιτούν κάθε φορά έλεγχο ολόκληρου του συστήματος. Επιπλέον, από τη στιγμή που χρησιμοποιήθηκε μια συγκεκριμένη τεχνολογία για την δημιουργία της εφαρμογής, τόσο ο εκσυγχρονισμός της με την εισαγωγή νέων τεχνολογιών όσο και η βελτίωση της δυσκολεύουν. Τέλος, όπως είναι εμφανές και από την βασική αρχή σχηματισμού της μια μονολιθική εφαρμογή χρησιμοποιεί μόνο μια βάση. Το χαρακτηριστικό αυτό δεν επιτρέπει στα δεδομένα να μπορούν να επεκταθούν σε περίπτωση που το απαιτούν οι συνθήκες.

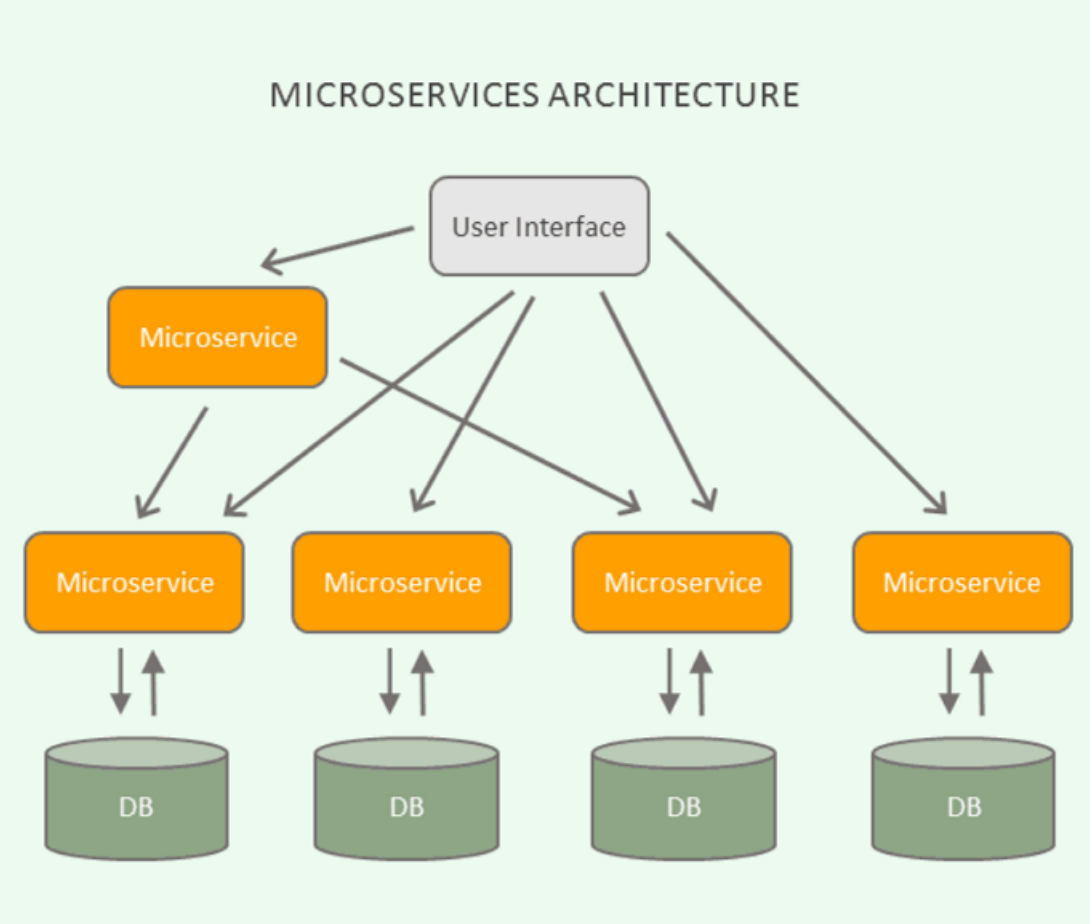
Συνεπώς, η συνεχής ανάπτυξη των συστημάτων και του αριθμού των λειτουργιών που πρέπει να εκτελεί ένα σύστημα οδηγούν στην ανάγκη για την ένταξη μιας νέας μορφής σχεδίασης και υλοποίησης η οποία θα καλύπτει πολλά από τα μειονεκτήματα που παρουσιάστηκαν. Επιπλέον, οι περισσότεροι τομείς όπως είναι αυτός της βιομηχανίας, απαιτούν συνεχώς αλλαγές και δεν έχουν το έδαφος το οποίο απαιτείται για να υπερισχύουν τα πλεονεκτήματα της μονολιθικής αρχιτεκτονικής. Έτσι περνάμε σε ένα νέο μοντέλο αρχιτεκτονικής το οποίο συμπληρώνει τα κενά που παρουσιάζει το μοντέλο που περιγράφηκε προηγουμένως και προσδίδει ένα σύνολο από πλεονεκτήματα τόσο στον τρόπο λειτουργίας όσο και στον τρόπο εξέλιξης των εφαρμογών.

3.2.4 Αρχιτεκτονική των Μικροϋπηρεσιών

Τα τελευταία χρόνια ο ρυθμός ζήτησης αγοράς νέων πιο εξελιγμένων εφαρμογών, η απαίτηση για συνεχή βελτίωση τους και ανακατασκευή τους αλλά και η απαίτηση για αλλαγή στον τρόπο με τον οποίο αυτές αναπτύσσονται οδηγεί στην ανάγκη για μια νέα μορφή αρχιτεκτονικής. Με τον όρο μικροϋπηρεσίες, ορίζεται ένα σύνολο ανεξάρτητων υπηρεσιών οι οποίες αλληλεπιδρούν και λειτουργούν συνολικά [44]. Η νέα μορφή αυτή παρουσιάζει διάφορες προκλήσεις ,τόσο στο τεχνικό κομμάτι όσο και στο κομμάτι της οργάνωσης, ανάλογα με το εάν η δημιουργία της ξεκινήσει από την αρχή η αν θα αποτελέσει στάδιο μετάβασης από μια ήδη υπάρχουσα εφαρμογή.

Δύο από τα κυριότερα στοιχεία που χαρακτηρίζουν τις μικροϋπηρεσίες είναι η αποκέντρωση και η αυτονομία [44]. Αποκέντρωση επιτυγχάνεται καθώς όλη η λογική της εφαρμογής χωρίζεται σε μικρά κομμάτια, όπου η κάθε υπηρεσία είναι ανεξάρτητη και αυτόνομη σε σχέση με τις υπόλοιπες και είναι υπεύθυνη για την εκτέλεση μιας λειτουργίας. Έχοντας ως βάση την λογική αυτή η κάθε υπηρεσία μπορεί να είναι απομονωμένη χωρίς να υπάρχει κεντρικός έλεγχος και διαχείριση από ένα κεντρικό μεμονωμένο στοιχείο. Αυτό συμβαίνει καθώς η κάθε υπηρεσία αποτελεί ένα πλήρες κομμάτι μιας ολοκληρωμένης λειτουργίας. Με λίγα λόγια η κάθε υπηρεσία διαχειρίζεται στο δικό της περιβάλλον ότι είναι απαραίτητο ώστε να λειτουργεί ομαλά χωρίς να εξαρτάται άμεσα ή έμμεσα η αποτελεσματική εκτέλεση της.

Όσον αφορά το χαρακτηριστικό της αυτονομίας, αυτό είναι άμεσα συνδεδεμένο με την επιλογή της κάθε ομάδας που είναι υπεύθυνη να εξελίξει την ανάλογη υπηρεσία για όλες τις αποφάσεις που επρόκειτο να ληφθούν και αφορούν την εξέλιξη του λογισμικού. Η κάθε υπηρεσία μπορεί να εξελίσσεται και να αναπτύσσεται με τον δικό της ρυθμό χωρίς να επηρεάζει το υπόλοιπο σύνολο. Η λογική της αρχιτεκτονικής των μικροϋπηρεσιών, όπως φαίνεται στην Εικόνα 3.3, έχει ως στόχο να διαχωρίσει όλη την λογική του συστήματος σε μικρότερα και απλούστερα στοιχεία τα οποία αλληλεπιδρούν μεταξύ τους κυρίως με ελαφριούς μηχανισμούς με την πιο συχνή επιλογή να αποτελεί αυτή των HTTP APIs. Κάθε υπηρεσία επίσης μπορεί να έχει ξεχωριστή βάση δεδομένων με την οποία μπορεί να ανταλλάσσει δεδομένα.



Εικόνα 3.3: Αρχιτεκτονική των μικροϋπηρεσιών [45]

3.2.5 Πλεονεκτήματα Μικροϋπηρεσιών

Ένα από τα μεγαλύτερα πλεονεκτήματα που προσφέρει η αρχιτεκτονική των μικροϋπηρεσιών είναι αυτό της κλιμάκωσης και αποκλιμάκωσης. Συγκεκριμένα, σε περίπτωση που κατά την διάρκεια της εξέλιξης ενός προγράμματος παρουσιαστεί η ανάγκη για την προσθήκη επιπλέον λειτουργιών ή ακόμα και επιπλέον πόρων αποθήκευσης υπάρχει η δυνατότητα να ενταχθούν στο σύστημα πάλι με την μορφή υπηρεσιών χωρίς να υπάρχει καμία επιρροή στο υπόλοιπο σύστημα και χωρίς να απαιτείται αναδιαμόρφωση του. Αν πάλι στην πορεία η ανάγκη των επιπλέον πόρων αναιρεθεί είναι δυνατόν η παραπάνω διαδικασία να πραγματοποιηθεί αντιστρόφως, δηλαδή οι πόροι και οι λειτουργίες να αφαιρεθούν ολοκληρωτικά πάλι χωρίς να διαταράσσεται το σύστημα και οι υπόλοιπες λειτουργίες του. Το μόνο χαρακτηριστικό που προστίθεται ή αφαιρείται αντίστοιχα στις περιπτώσεις αυτές είναι ο τρόπος επικοινωνίας με τις υπόλοιπες υπηρεσίες του προγράμματος ώστε να μπορεί να πραγματοποιείται αποτελεσματικά η μεταφορά δεδομένων μεταξύ τους. Τέλος εκτός από την δυνατότητα προσθαφαίρεσης των μικροϋπηρεσιών δίνεται η επιπλέον δυνατότητα επαναχρησιμοποίησης μιας μεμονωμένης και ολοκληρωμένης υπηρεσίας για κάποιο άλλο σκοπό σε περίπτωση που αυτή τον αντιπροσωπεύει.

Εκτός από την δυνατότητα επέκτασης ακόμα ένα πλεονέκτημα είναι η ευελιξία. Πρώτα απ' όλα η δυνατότητα αυτή δίνεται στο τεχνολογικό κομμάτι, όπου είναι πλέον εφικτό να χρησιμοποιείται για κάθε υπηρεσία διαφορετική γλώσσα προγραμματισμού και διαφορετικές τεχνολογίες υλοποίησης. Κάτι τέτοιο είναι ιδιαίτερα χρήσιμο καθώς μια λειτουργία μπορεί να είναι αποτελεσματικότερη αν υλοποιείται με μια

τεχνολογία ενώ κάποια άλλη να είναι πιο αργή η πιο δυσλειτουργική. Έτσι ανάλογα με τις απαιτήσεις της κάθε λειτουργίας αυτή διαμορφώνεται τεχνολογικά ανεξάρτητα από τις υπόλοιπες.

Ακόμα μια ευελιξία παρουσιάζεται στο κομμάτι της ανάπτυξης και της αποθήκευσης. Κάθε υπηρεσία μπορεί να υλοποιείται σε διαφορετική πλατφόρμα και σε διαφορετικό περιβάλλον από τις υπόλοιπες συνεχίζοντας να επικοινωνεί κανονικά. Αυτό το πλεονέκτημα παρουσιάζει πολλές ευκολίες στο πως μπορούν οι ομάδες να δρουν και αυτές ανεξάρτητα με διαφορετικές γνώσεις με στόχο την ανάπτυξη διαφορετικών υπηρεσιών. Για παράδειγμα με την είσοδο νέων μελών στην ομάδα, δεν είναι υποχρεωτική η εκπαίδευση τους σε όλη την εφαρμογή αλλά αντιθέτως μπορούν να τους ανατεθούν εργασίες που αφορούν μια μόνο υπηρεσία και έτσι η εκπαίδευση αλλά και η αρχή της παραγωγικότητας τους είναι γρηγορότερη καθώς χρειάζεται να εκπαιδευτούν σε πολύ μικρότερο υλικό. Το χαρακτηριστικό της ευελιξίας επίσης μπορεί να αποβεί σωτήριο σε περίπτωση κάποιου προβλήματος όπου η υπηρεσία έχει την δυνατότητα να αλλάζει περιβάλλον χωρίς επιπλέον προβλήματα.

3.2.6 Μειονεκτήματα των Μικροϋπηρεσιών

Παρ' όλο που τα χαρακτηριστικά των μικροϋπηρεσιών προσδίδουν σημαντικά οφέλη στην ανάπτυξη και διαμόρφωση των συστημάτων, είναι σημαντικό να αναφέρουμε και τα σημεία στα οποία υστερούν. Το πρώτο μειονέκτημα που παρουσιάζει η αρχιτεκτονική αυτή είναι η δυσκολία στην αρχική φάση της ανάπτυξης της. Από την στιγμή που η κάθε υλοποίηση της εφαρμογής παρουσιάζει διαφορετικές απαιτήσεις είναι αναγκαίο να βρεθεί ο καταλληλότερος και πιο αντιπροσωπευτικός τρόπος ταξινόμησης και επικοινωνίας μεταξύ των υπηρεσιών.

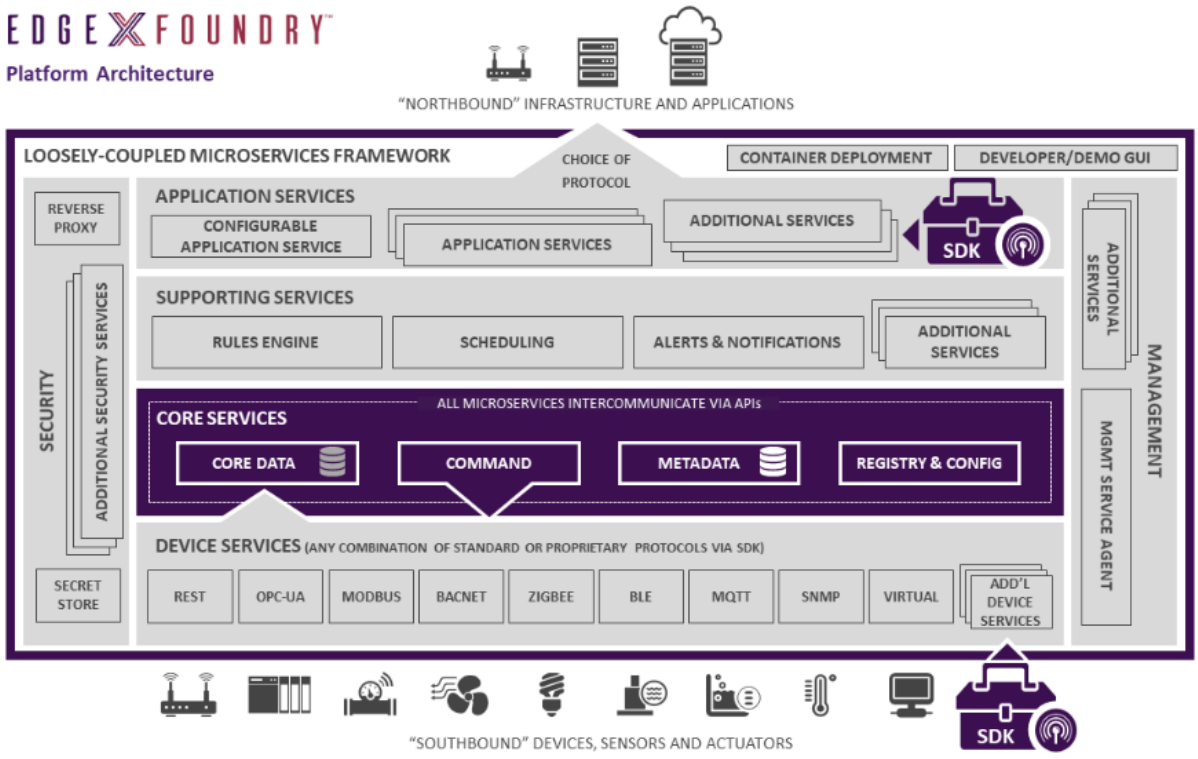
Μια τέτοια διαδικασία απαιτεί καλή οργάνωση και συνεπώς κατανάλωση χρόνου, επειδή στα αρχικά στάδια εμφανίζει αρκετά μεγάλη πολυπλοκότητα και χρειάζεται να γίνει με αρκετά μεγάλη ακρίβεια ώστε να εξασφαλίζεται πρώτον πως οι υπηρεσίες έχουν διαχωριστεί τελείως μεταξύ τους και δεύτερον ότι τα προβλήματα που θα προκύψουν κατά τη διάρκεια της εφαρμογής της αρχιτεκτονικής θα είναι όσο το δυνατόν λιγότερα. Χρειάζεται επίσης να οριστεί εξ' αρχής ένας καλός και άμεσος τρόπος επικοινωνίας μεταξύ τους και να εξασφαλίζεται πως κάθε υπηρεσία μπορεί να είναι προσβάσιμη για ανταλλαγή δεδομένων από όλα τα στοιχεία του συστήματος. Ακόμα ένα μειονέκτημα που μπορεί να παρουσιαστεί αφορά την αύξηση στην κίνηση μέσα στο δίκτυο. Καθώς ο τρόπος με βάση τον οποίο σχεδιάζεται η αρχιτεκτονική των μικροϋπηρεσιών απαιτεί συνεχή επικοινωνία μεταξύ τους κάτι τέτοιο μπορεί να αυξήσει αρκετά την κίνηση του δικτύου. Συνεπώς αν η σύνδεση μεταξύ τους δεν έχει σχεδιαστεί με κατάλληλο τρόπο είναι πιθανόν να εμφανίζεται μια επιπλέον αύξηση στην καθυστέρηση που αφορά στην λειτουργία του δικτύου.

Όπως έγινε αντιληπτό από την παρουσίαση των δύο αρχιτεκτονικών και οι δύο εμφανίζουν πλεονεκτήματα ανάλογα με τις απαιτήσεις με αυτά της μονολιθικής αρχιτεκτονικής να επικρατούν σε μικρά συστήματα που δεν χρειάζονται συνεχή ανάπτυξη και αναβάθμιση ,ενώ με αυτά των μικροϋπηρεσιών να επικρατούν σε εφαρμογές που αντιπροσωπεύουν την εποχή και τις απαιτήσεις του σήμερα καθώς απαιτούν μεγάλη ευελιξία, δυνατότητα εξέλιξης και μεγάλο αριθμό από λειτουργίες όπως είναι οι εφαρμογές στον τομέα της βιομηχανίας. Η αύξηση του αριθμού των μικροϋπηρεσιών που εφαρμόζονται για τη λειτουργία ενός συστήματος δημιουργεί την ανάγκη για χρήση εργαλείων τα οποία είναι κατάλληλα για την ανάπτυξη και διαχείριση και εκτέλεση των μικροϋπηρεσιών αυτών.

3.3 Πλατφόρμες του Βιομηχανικού Διαδικτύου των Πραγμάτων που εφαρμόστηκαν

3.3.1 Πλατφόρμα EdgeX Foundry

Η πλατφόρμα EdgeXFoundry [46] αποτελεί μια λύση ανοιχτού κώδικα, δημιουργημένη από το Linux Foundation, μια ομάδα που συμβάλλει στην δημιουργία ανοιχτού κώδικα και σήμερα υπάρχουν διαθέσιμες οχτώ διαφορετικές εκδόσεις. Με στόχο την διευκόλυνση στην επικοινωνία μεταξύ των στοιχείων του διαδικτύου των πραγμάτων και του περιβάλλοντος στο οποίο εφαρμόζονται, δημιουργεί ένα πρότυπο με βάση το οποίο θα λειτουργεί η επικοινωνία μεταξύ των συσκευών του διαδικτύου των πραγμάτων και του IT συστήματος εκτελώντας τον ρόλο ενός ενδιάμεσου επιπέδου. Όλα τα στοιχεία τα οποία συμπεριλαμβάνονται στην πλατφόρμα αυτή υφίστανται με την μορφή μικροϋπηρεσιών εκτελούνται σε διαφορετικές γλώσσες προγραμματισμού με την πλειονότητα τους να είναι σχεδιασμένες στην γλώσσα προγραμματισμού Go και εκτελούνται σε περιβάλλον container. Όσον αφορά την αποθήκευση δεδομένων στο άκρο του δικτύου χρησιμοποιείται η βάση δεδομένων Mongo DB. Η πλατφόρμα του EdgeX χρησιμοποιείται σε εφαρμογές του διαδικτύου των πραγμάτων στην βιομηχανία [47] και καθώς έχει την δυνατότητα να επικοινωνεί απευθείας με τις συσκευές - αισθητήρες σχεδιάστηκε κυρίως για να εκτελείται στο edge επίπεδο του δικτύου. Επιπλέον η συγκεκριμένη πλατφόρμα μπορεί να εκτελεστεί σε οποιοδήποτε από τα διαθέσιμα λειτουργικά συστήματα (Windows, Linux, MacOs..). Στην Εικόνα 3.4 παρουσιάζεται η αρχιτεκτονική της πλατφόρμας που συμπεριλαμβάνει τα επίπεδα και τα κύρια στοιχεία από τα οποία αποτελείται η πλατφόρμα.



Εικόνα 3.4: Αρχιτεκτονική της πλατφόρμας EdgeX Foundry [48]

Υπάρχουν τέσσερα επίπεδα που συντελούν το περιβάλλον του EdgeX, το επίπεδο των υπηρεσιών συσκευής (device services), το επίπεδο υπηρεσιών πυρήνα (core services), το επίπεδο υπηρεσιών υποστήριξης (support services) και τέλος το επίπεδο υπηρεσιών εφαρμογής(application services). Η αρχιτεκτονική των τεσσάρων αυτών επιπέδων συνδέει τα δύο μέρη του δικτύου που χωρίζονται σε δύο περιοχές, το νότιο όριο (south bound) το οποίο αποτελείται από διάφορα είδη IoT συσκευών και το βόρειο όριο (north bound) το οποίο αποτελείται από επιχειρησιακά συστήματα, εφαρμογές και τα συστήματα του υπολογιστικού νέφους.

Το ενδιάμεσο επίπεδο του συνδέει τις υπηρεσίες του επιπέδου των υπηρεσιών πυρήνα με τις συσκευές είναι το επίπεδο των υπηρεσιών συσκευής το οποίο αποτελείται από ένα σύνολο μικρο-υπηρεσιών οι οποίες χρησιμοποιώντας το καταλληλότερο πρωτόκολλο, που σχετίζεται με τη συσκευή που αποστέλλει δεδομένα, μεταφέρουν την πληροφορία στο επόμενο επίπεδο της πλατφόρμας. Για την λειτουργία αυτή η πλατφόρμα παρέχει ένα προϋπάρχον SDK ειδικά σχεδιασμένο για τη δημιουργία καινούριας υπηρεσίας συσκευής αλλά ακόμα παρέχει ένα σύνολο από προσχεδιασμένες υπηρεσίες διαφορετικού τύπου που μπορούν να εξυπηρετήσουν ένα μεγάλο εύρος σεναρίων.

Το επίπεδο των core services είναι από τα πιο σημαντικά επίπεδα καθώς σε αυτό περιλαμβάνονται σημαντικές πληροφορίες που αφορούν τις συσκευές που αποστέλλουν δεδομένα στο σύστημα, τα ίδια τα δεδομένα που διέρχονται από την πλατφόρμα αλλά και τις προσαρμογές που πρέπει να γίνουν στην πλατφόρμα ανάλογα με την περίπτωση αποστολής. Το επίπεδο αυτό συντελείται από τέσσερις κύριες υπηρεσίες. Η πρώτη και από τις πιο σημαντικές είναι η υπηρεσία core data η οποία είναι υπεύθυνη για την ανάγνωση δεδομένων που στέλνονται από οποιαδήποτε υπηρεσία συσκευής και για την διατήρηση των δεδομένων στο edge σε περιπτώσεις που είναι απαραίτητο. Έπειτα η υπηρεσία core-metadata περιλαμβάνει πληροφορίες που έχουν να κάνουν κυρίως με την διασύνδεση των device-service με τις συσκευές ποια είναι η σύνδεση μεταξύ τους και πώς ορίζεται η επικοινωνία τους. Επιπλέον η υπηρεσία εντολής (command) είναι απαραίτητη καθώς διαχειρίζεται το σύνολο των εντολών που λαμβάνονται από την βόρεια πλευρά από τα στοιχεία του συστήματος και απευθύνονται στις συσκευές της νότιας πλευράς, δηλαδή τις συσκευές. Τέλος η υπηρεσία εγγραφής και διαμόρφωσης παρέχει ένα σύνολο πληροφοριών που σχετίζονται με όλες τις υπηρεσίες που είναι συνδεδεμένες στο περιβάλλον της πλατφόρμας.

Στη συνέχεια ακολουθεί το επίπεδο των υποστηρικτικών του οποίου οι υπηρεσίες μπορούν και να παραληφθούν καθώς ο ρόλος τους είναι προαιρετικός για την λειτουργία της πλατφόρμας και του συστήματος. Αυτές οι υπηρεσίες είναι η μηχανή κανόνων (rules engine) που εκτελεί συνθήκες υπόθεσης στα δεδομένα των αισθητήρων και μπορεί να αντικατασταθεί από μια υπηρεσία ανάλυσης τους, η υπηρεσία του χρονοπρογραμματισμού η οποία μπορεί να ορίσει τη χρονική στιγμή κατά της οποία μια υπηρεσία θα ξεκινήσει ή θα σταματήσει την λειτουργία της και τέλος η υπηρεσία των προειδοποιήσεων και ειδοποιήσεων και αφορά την αποστολή ενημερωτικών πληροφοριών που σχετίζονται με την κατάσταση των υπηρεσιών της πλατφόρμας σε τρίτο πρόσωπο που παρακολουθεί την λειτουργία της.

Το τελευταίο επίπεδο αποτελείται από ένα σύνολο υπηρεσιών εφαρμογών είναι υπεύθυνο για όλες τις διαδικασίες ανάλυσης και επεξεργασίας των δεδομένων που λαμβάνει με σκοπό την εξαγωγή τους στην αντίθετη περιοχή του δικτύου. Υπάρχουν επίσης στα δύο ακριανά σημεία της αρχιτεκτονικής της πλατφόρμας δύο επιπλέον επίπεδα. Από την μια όσον αφορά το κομμάτι της ασφάλειας υπάρχουν δύο κύρια στοιχεία που το υποστηρίζουν και αυτά είναι μια βάση δεδομένων που περιέχει όλα τα ευαίσθητα δεδο-

μένα που σχετίζονται με την πλατφόρμα και μια θύρα API με την χρήση της οποίας ελέγχεται η κίνηση που αφορά την κατοχύρωση REST πόρων της πλατφόρμας. Τέλος, το επίπεδο της διαχείρισης παρέχει την δυνατότητα στον εξωτερικό χρήστη να μπορεί να ελέγχει την κατάσταση των υπηρεσιών και να την μεταβάλλει ορίζοντας πότε και ποιες υπηρεσίες εκτελούνται.

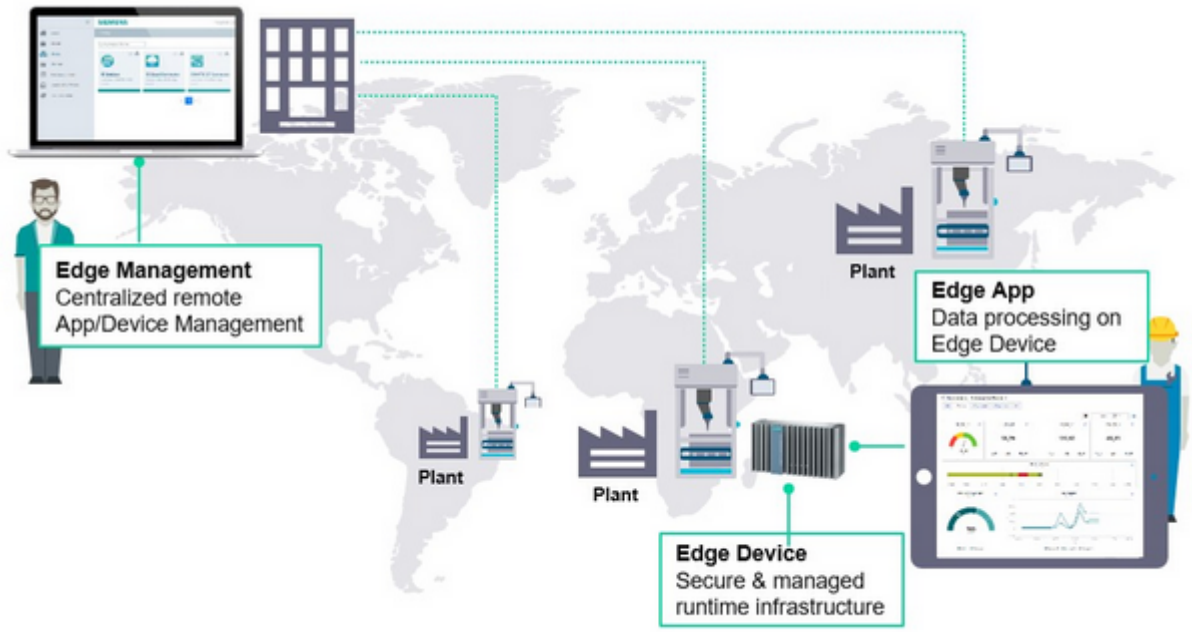
Η ροή των δεδομένων μεταξύ των επιπέδων της αρχιτεκτονικής που περιγράφηκαν προηγουμένως έχει ως εξής. Αρχικά μια συσκευή-αισθητήρας πραγματοποιεί μια μέτρηση και επιλέγει ένα από τα πρωτόκολλα που παρέχονται μέσω των device services στην υπηρεσία core-data στο δεύτερο επίπεδο όπου γίνεται και μια τοπική διατήρηση των δεδομένων που ελήφθησαν. Έπειτα η υπηρεσία core-data στο επίπεδο των υπηρεσιών εφαρμογών όπου πραγματοποιούνται διαδικασίες φιλτραρίσματος και απαραίτητων επεξεργασιών και έπειτα από εκεί με την επιλογή κατάλληλου πρωτοκόλλου περνάνε στο σύστημα. Η πλατφόρμα που παρουσιάστηκε αποτελεί όπως προαναφέρθηκε μια λύση ανοιχτού κώδικα. Στη συνέχεια παρουσιάζεται μια δεύτερη λύση για τα συστήματα που εφαρμόζουν το διαδίκτυο των πραγμάτων στο edge.

3.3.2 Πλατφόρμα SIEMENS

Η πλατφόρμα την οποία παρέχει η SIEMENS για την εξυπηρέτηση υπηρεσιών στις οποίες εφαρμόζεται το Βιομηχανικό Διαδίκτυο των Πραγμάτων αποτελεί μια από τις πιο συχνά χρησιμοποιούμενες λύσεις στον τομέα της διαχείρισης των λειτουργιών στον τομέα της βιομηχανίας. Η ονομασία της πλατφόρμας είναι Βιομηχανικό Άκρο (Industrial Edge) [49] , αναπτύσσεται στο άκρο του δικτύου ώστε να υπάρχει άμεση αλληλεπίδραση με τις συσκευές, αποτελεί ιδιόκτητη υπηρεσία και το περιβάλλον της περιλαμβάνει ένα σύνολο από διαφορετικά επίπεδα. Καθώς αποτελεί ιδιόκτητη υπηρεσία οι πληροφορίες που μπορούν να παρατεθούν σχετικά με την περιγραφή της και μερικά κομμάτια από το περιβάλλον της είναι περιορισμένες.

Στην Εικόνα 3.5 απεικονίζεται ένα μέρος των λειτουργιών του Industrial Edge. Πιο συγκεκριμένα δυνατότητες όπως παρακολούθηση των δεδομένων από απόσταση, παροχή συσκευών που λειτουργώντας με ασφάλεια συλλέγουν τα δεδομένα που παράγονται από τις βιομηχανικές μηχανές αλλά και εφαρμογές που εκτελούν την επεξεργασία και διαχείριση των δεδομένων που παράγονται από τις μηχανές που βρίσκονται στο άκρο του δικτύου. Η πλατφόρμα αυτή συμβάλλει στην αυτοματοποίηση και βελτιστοποίηση της βιομηχανίας μέσω των παραπάνω δυνατοτήτων που προσφέρει που αφορούν στην διαχείριση και επεξεργασία των δεδομένων που παράγονται σε ένα βιομηχανικό περιβάλλον. Καθώς όλες οι εφαρμογές εκτελούνται στα άκρα του δικτύου όλες οι διαδικασίες πραγματοποιούνται ταχύτερα και με μεγαλύτερη ακρίβεια. Οι λειτουργίες που προσφέρονται από το Industrial Edge πραγματοποιούνται χάρης τα τρία βασικά στοιχεία που το συντελούν: τον Βιομηχανικό Κόμβο Άκρου (Industrial Edge Hub), την Διαχείριση Βιομηχανικού Άκρου(Industrial Edge Management) και τον Βιομηχανικό Χρόνο Εκτέλεσης στις Βιομηχανικές Συσκευές (Industrial Edge Runtime on Industrial Devices).

Αρχικά υπάρχει το επίπεδο της Διαχείρισης (Management) το οποίο λειτουργεί ως μεσολαβητής σε κάθε διαδικασία εγκατάστασης μιας νέας υπηρεσίας στην πλατφόρμα και παρέχει μια αξιόπιστη και κεντρική διαχείριση τόσο των συσκευών που βρίσκονται στην πλατφόρμα όσο και των εφαρμογών της. Είναι επίσης υπεύθυνο όσον αφορά την οποιαδήποτε αναβάθμιση και ανανέωση εφαρμογών και υπηρεσιών της πλατφόρμας και είναι το περιβάλλον στο οποίο μπορεί να οριστούν όλοι οι μετασχηματισμοί.



Εικόνα 3.5: Λειτουργία του Industrial Edge [50]

Το επίπεδο διαχείρισης αποτελεί μοναδικό στοιχείο και είναι δυνατόν να αναπτυχθεί σε ένα βιομηχανικό περιβάλλον σε οποιοδήποτε σημείο του εύρους που παρέχεται από το επίπεδο IT. Το επίπεδο αυτό παρέχει μια επιπλέον μορφή ασφάλειας καθώς όλες οι διαδικασίες που αφορούν την λειτουργία της πλατφόρμας και την αλληλεπίδραση της με το βιομηχανικό περιβάλλον περνάνε και ελέγχονται από το Management. Το δεύτερο κύριο στοιχείο της πλατφόρμας αποτελεί το περιβάλλον των Βιομηχανικών Συσκευών. Σε κάθε βιομηχανική εγκατάσταση συσκευές άκρως συνδέονται απευθείας με τις μηχανές οι οποίες αποστέλλουν δεδομένα μέτρησης και συνεπώς το στοιχείο αυτό αντικατοπτρίζει μια πραγματική συσκευή η οποία παρουσιάζεται και εικονικά στο περιβάλλον της πλατφόρμας. Παρέχονται δύο κύριοι τύποι συσκευών, η εικονική και η φυσική και επιπλέον ο αριθμός των συσκευών που συνδέονται και αλληλεπιδρούν με το επίπεδο της διαχείρισης μπορεί να ποικίλει.

Όλες οι εφαρμογές που υλοποιούνται στην πλατφόρμα αυτή εκτελούνται με την μορφή container (docker containers). Στο περιβάλλον της πλατφόρμας υπάρχει επίσης μια λίστα από όλες τις εφαρμογές που υπάρχουν στην πλατφόρμα και ονομάζεται IE Catalog. Από την στιγμή που μια υπηρεσία παρουσιάζεται στην λίστα αυτή είναι διαθέσιμη για την εγκατάσταση της σε κάποια από τις συσκευές. Το IED παρέχει διαχείριση εφαρμογών και Security και User/Idetity Mgmt ως μέρος της υλοποίησης των υπηρεσιών πλατφόρμας μαζί με ένα σύστημα pub/sub βασισμένο στο MQTT, που ονομάζεται IE Databus, για τη διαχείριση δεδομένων και επιτυχημένη παράδοση τους.

Η πρώτη ενότητα υλοποιεί λειτουργίες για εφαρμογές διαχείρισης και διαμόρφωσης κόμβων, όπως η εκκίνηση/διακοπή εφαρμογών, παρακολούθηση της κατάστασης και γενικές ρυθμίσεις της συσκευής. Η ενότητα της ασφάλειας και διαχείρισης χρηστών/ταυτότητας ρυθμίζει την πρόσβαση έλεγχο με βάση τους ρόλους των χρηστών και αποφασίζει για τις λειτουργίες που επιτρέπεται να πραγματοποιηθούν όπως και το ποια οντότητα είναι εξουσιοδοτημένη να τις εκτελέσει. Η ασφάλεια στο IED της Siemens ρυθμίζεται

από πιστοποιητικά και την πρόσβαση χρήστη/κωδικού πρόσβασης. Το IED έχει τη δυνατότητα εκτέλεσης προσαρμοσμένων εφαρμογών και υπηρεσιών με την γενική ονομασία Edge App. Πρόκειται για εφαρμογή dockercompose που φορτώνεται στο IEM και στη συνέχεια αναπτύσσεται στο IED. Το αρχείο docker compose πρέπει να τηρεί ορισμένα κριτήρια προκειμένου να είναι συμβατό με το Industrial Edge Runtime. Η προσαρμοσμένη εφαρμογή Edge μπορεί να επικοινωνεί αυτόνομα ή με άλλες εφαρμογές της Siemens μέσω του IE Databus.

3.4 Εργαλεία που χρησιμοποιήθηκαν

3.4.1 Docker

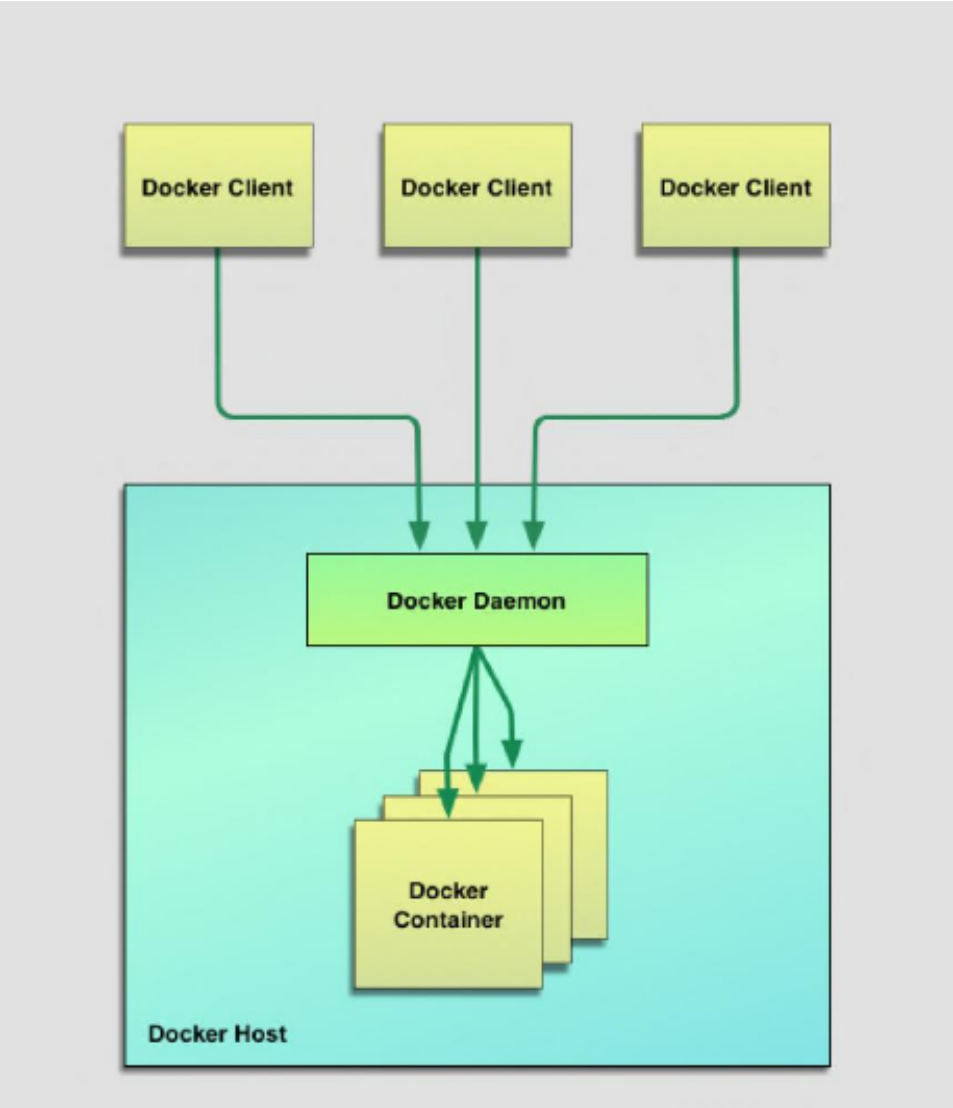
Είναι σημαντικό η κάθε εφαρμογή να μπορεί να αναπτύσσεται με ευελιξία χωρίς η λειτουργία της να εξαρτάται άμεσα από το περιβάλλον στο οποίο εκτελείται. Ειδικά στα πλαίσια εφαρμογής των μικροϋπηρεσιών, παρουσιάζεται η ανάγκη μια υπηρεσία να εκτελείται για την υλοποίηση διαφορετικών σεναρίων και σε πολλαπλά πλαίσια εργασίας και να μπορεί να μεταβάλλεται με μεγάλη ευκολία. Μια από τις αλλαγές που ενσωματώνουν όλο και περισσότερες επιχειρήσεις είναι η εκτέλεση των εφαρμογών στο πλήρες και απομονωμένο περιβάλλον των container (container).

Σε κάθε σύστημα εκτελείται ένα σύνολο εφαρμογών των οποίων η λειτουργία σχετίζεται άμεσα με τις απαιτήσεις (dependencies) και με τις βιβλιοθήκες που τις καθορίζουν. Η σύνδεση αυτή είναι πολύ συγκεκριμένη καθώς η κάθε εφαρμογή ή ακόμα και η κάθε αναβάθμιση της εξαρτάται από ειδική έκδοση των επιμέρους στοιχείων της. Μια τέτοια συνθήκη δυσκολεύει ιδιαίτερα τόσο την μετακίνηση της εφαρμογής από μηχανή σε μηχανή όσο και την ευελιξία που αφορά στις μελλοντικές μεταβολές και την συντήρηση της. Ένας container μπορεί να οριστεί ως ένα περιβάλλον απομονωμένο όπου εκτελείται μια εφαρμογή και το οποίο συμπεριλαμβάνει όλα τα προαπαιτούμενα στοιχεία της εκτός από το λειτουργικό σύστημα. Συγκεκριμένα ένας container (container) εκτελείται μέσω μιας μηχανής (container engine). Η διεργασία που αφορά έναν container είναι ολοκληρωτικά συνδεδεμένη με τον κύκλο ζωής (lifecycle) του. Πιο συγκεκριμένα η έναρξη του container σημαίνει έναρξη λειτουργίας της διεργασίας του και αντίστοιχα η έξοδος από την διεργασία του container σηματοδοτεί την διακοπή του. Παράλληλα κατά την εκτέλεση του εκτελούνται και άλλες διεργασίες στο περιβάλλον του.

Οι container όπως και οι εικονικές μηχανές είναι και οι δύο δημοφιλείς τεχνολογίες που χρησιμοποιούνται για την ανάπτυξη και την απομόνωση λογισμικού. Ωστόσο, έχουν διαφορετικές αρχιτεκτονικές και περιπτώσεις χρήσης. Μια εικονική μηχανή (Virtual Machine - VM) αποτελεί μια μορφή αφαίρεσης λογισμικού μιας φυσικής μηχανής, η οποία επιτρέπει την εκτέλεση πολλαπλών λειτουργικών συστημάτων σε μια μόνο φυσική μηχανή υποδοχής [51]. Κάθε VM έχει τους δικούς του εικονικοποιημένους πόρους υλικού, όπως CPU, μνήμη και αποθήκευση. Αυτό επιτρέπει την εκτέλεση πολλαπλών εφαρμογών σε διαφορετικά VMs, η καθεμία με το δικό της απομονωμένο περιβάλλον. Έτσι παρ' όλο που οι εικονικές μηχανές μοιράζονται τους πόρους του υλικού του συστήματος, οι εφαρμογές που εκτελούνται σε κάθε μια από αυτές δεν έχουν επίγνωση για τα στοιχεία και τις εφαρμογές που εκτελούνται στις υπόλοιπες εικονικές μηχανές. Τα VM απαιτούν ένα σύστημα ελέγχου παρακολούθησης(hypervisor) για τη διαχείριση των εικονικοποιημένων πόρων υλικού, ο οποίος προσθέτει ένα πρόσθετο επίπεδο επιβάρυνσης. Υπάρχουν όμως πολλές περιπτώσεις στις οποίες δεν είναι απαραίτητο για την απομονωμένη εκτέλεση μιας εφαρμογής να δημιουργείται ένα τόσο ολοκληρωμένο περιβάλλον το οποίο οδηγεί σε μεγάλη κατανά-

λωση πόρων, μεγαλύτερη καθυστέρηση αλλά και μεγάλη δυσκολία στην συντήρηση και ανάπτυξη των εφαρμογών.

Από την άλλη πλευρά, οι container αποτελούν μια ελαφριά εναλλακτική λύση για τα VM που παρέχουν εικονικοποίηση σε επίπεδο λειτουργικού συστήματος. Πιο συγκεκριμένα οι container επιτρέπουν την εκτέλεση πολλαπλών εφαρμογών σε μία μόνο περίπτωση λειτουργικού συστήματος, με κοινή χρήση του ίδιου πυρήνα και των ίδιων βιβλιοθηκών. Αυτό καθιστά τους καθιστά πιο αποδοτικούς ως προς τους πόρους σε σχέση με τα VM, καθώς δεν απαιτούν συστήματα ελέγχου παρακολούθησης και μπορούν να εκκινούνται και να σταματούν γρήγορα. Επιπλέον οι container χρησιμοποιούν ένα πολυεπίπεδο σύστημα αρχείων για τη συσκευασία των εφαρμογών και των εξαρτήσεων τους, καθιστώντας εύκολη την ανάπτυξη και τη διαχείρισή τους. Συνοπτικά, τα VM παρέχουν εικονικοποίηση σε επίπεδο υλικού, ενώ οι container παρέχουν εικονικοποίηση σε επίπεδο λειτουργικού συστήματος. Τα VM είναι καταλληλότερα για την εκτέλεση πολλαπλών λειτουργικών συστημάτων σε έναν μόνο κεντρικό υπολογιστή, ενώ οι container είναι ιδανικά για την ανάπτυξη και διαχείριση πολλαπλών εφαρμογών σε μία μόνο περίπτωση λειτουργικού συστήματος.



Εικόνα 3.6: Αρχιτεκτονική Docker [52]

Το Docker [53] αποτελεί ένα εργαλείο το οποίο χρησιμοποιείται με στόχο την διευκόλυνση της ανάπτυξης διαχείρισης και εκτέλεσης των εφαρμογών σε μορφή container. Παρέχεται ως εργαλείο ανοιχτού κώδικα και πρωτοεκδόθηκε από τους δημιουργούς του με τη χρήση της άδειας Apache 2.0 [52]. Συντελείται από τέσσερα βασικά στοιχεία όπως φαίνεται και στην Εικόνα 3.6 τα οποία καθορίζουν τις λειτουργίες που εκτελεί και παρουσιάζονται παρακάτω μαζί με τα υπόλοιπα χρήσιμα χαρακτηριστικά του.

Πελάτης-Εξυπηρετητής(Client-Server): Το Docker χρησιμοποιεί μια αρχιτεκτονική πελάτη εξυπηρετητή η οποία απεικονίζεται στην Εικόνα 3.6. Το πρώτο επίπεδο αφορά την επικοινωνία του πελάτη Docker με το αντίστοιχο Docker deamon ή με τον Docker εξυπηρετητή και έπειτα αντίστοιχα ο deamon ή ο εξυπηρετητής επικοινωνούν με τον container τον οποίο αφορά το αίτημα. Ο Docker deamon είναι υπεύθυνος για τη διαχείριση των container, των εικόνων, των δικτύων και των όγκων έννοιες που σχετίζονται με τον σχηματισμό και τις λειτουργίες που εκτελεί ο container. Ο πελάτης του Docker μπορεί να εγκατασταθεί σε διαφορετικά λειτουργικά συστήματα, όπως τα Windows, το macOS και το Linux, και παρέχει μια διεπαφή γραμμής εντολών (Command Line Interface - CLI) και μια γραφική διεπαφή χρήστη (Graphical User Interface - GUI) για την αλληλεπίδραση με το Docker deamon. Η επικοινωνία μεταξύ του πελάτη και του εξυπηρετητή πραγματοποιείται με την χρήση εντολών που παρέχει το Docker και ως λέξη κλειδί αλλά και έναρξη σύνταξης της οποιασδήποτε εντολής αποτελεί η λέξη ”docker”.

Εικόνα Docker (Docker Image): Μια εικόνα είναι ένα πρότυπο μόνο για ανάγνωση που περιέχει οδηγίες για τη δημιουργία ενός container. Περιλαμβάνει στοιχεία όπως τον κώδικα της εφαρμογής, τον χρόνο εκτέλεσης, τα εργαλεία συστήματος, τις βιβλιοθήκες και άλλες εξαρτήσεις που αποτελούν προϋπόθεση για την εκτέλεση της εφαρμογής. Οι εικόνες Docker βασίζονται σε μια πολυεπίπεδη αρχιτεκτονική, όπου κάθε επίπεδο αντιπροσωπεύει μια αλλαγή στην εικόνα. Κάθε στρώμα χτίζεται πάνω στο προηγούμενο επίπεδο και οι αλλαγές που γίνονται σε αυτό δεν επηρεάζουν τα υπόλοιπα. Η παραπάνω διαδικασία επιτρέπει την αποτελεσματική αποθήκευση και διανομή εικόνων, καθώς μόνο τα επίπεδα που έχουν αλλάξει χρειάζεται να ενημερωθούν. Όταν δημιουργείται ένας container Docker από μια εικόνα, ένα εγγράψιμο επίπεδο προστίθεται πάνω από τα στρώματα μόνο για ανάγνωση της εικόνας. Αυτό το στρώμα επιτρέπει την πραγματοποίηση αλλαγών στον container χωρίς να επηρεάζεται η υποκείμενη εικόνα. Για παράδειγμα, εάν μια εφαρμογή γράψει δεδομένα σε ένα αρχείο, τα δεδομένα θα αποθηκευτούν στο εγγράψιμο στρώμα αντί να τροποποιηθεί η υποκείμενη εικόνα. Συνολικά, οι εικόνες Docker αποτελούν βασικό συστατικό του οικοσυστήματος Docker, επιτρέποντας στους προγραμματιστές να δημιουργούν, να συσκευάζουν και να διανέμουν εφαρμογές εύκολα και αποτελεσματικά.

Docker container: Ο κάθε container, ένα στοιχείο το οποίο περιγράφηκε αναλυτικότερα παραπάνω, αποτελεί την εκτελέσιμη περίπτωση μιας εικόνας που είναι απομονωμένη από το σύστημα υποδοχής και όλους τους υπόλοιπους container. Παρ’ όλο που αποτελεί μια μορφή απομονωμένου συστήματος, είναι εφικτό και μάλιστα ιδιαίτερα χρήσιμο το γεγονός πως οι διάφοροι container έχουν την δυνατότητα να επικοινωνούν μεταξύ τους. Ως σύστημα παρουσιάζει ιδιαίτερη αυτονομία καθώς διαθέτει το δικό του σύνολο αρχείων, το δίκτυο στο οποίο έχει ρυθμιστεί να εντάσσεται και τον χώρο ονομάτων διεργασιών που εκτελούνται. Με τον τρόπο αυτό έχει την δυνατότητα να ξεκινήσει, να σταματήσει την εκτέλεση του αλλά και να διαγραφεί ανάλογα με τις ανάγκες του συστήματος και των εφαρμογών.

Μητρώο Docker (Docker Registry): Ένα μητρώο Docker είναι μια κεντρική τοποθεσία για την αποθήκευση και την κοινή χρήση εικόνων Docker. Είναι ουσιαστικά ένας διακομιστής που φιλοξενεί

αποθετήρια Docker, τα οποία περιέχουν εικόνες Docker. Ένα μητρώο επιτρέπει στους προγραμματιστές να μοιράζονται και να διανέμουν τις εικόνες Docker τους εύκολα και αποτελεσματικά, εντός του οργανισμού τους ή με την ευρύτερη κοινότητα. Το Docker παρέχει ένα προεπιλεγμένο δημόσιο μητρώο που ονομάζεται Docker Hub, το οποίο επιτρέπει στους προγραμματιστές να αποθηκεύουν και να μοιράζονται τις εικόνες Docker τους δημόσια ή ιδιωτικά. Το Docker Hub χρησιμοποιείται ευρέως και παρέχει ένα μεγάλο αποθετήριο προ-κατασκευασμένων εικόνων που μπορούν εύκολα να μεταφορτωθούν και να χρησιμοποιηθούν σε διάφορες εφαρμογές. Οι προγραμματιστές μπορούν επίσης να δημιουργήσουν τα δικά τους ιδιωτικά μητρώα Docker για την αποθήκευση των εικόνων τους, τα οποία μπορεί να είναι χρήσιμα σε περιπτώσεις όπου οι εικόνες περιέχουν ιδιόκτητες ή ευαίσθητες πληροφορίες που δεν θα πρέπει να μοιράζονται δημόσια. Η δημιουργία ενός ιδιωτικού μητρώου Docker απαιτεί τη λειτουργία ενός διακομιστή μητρώου στον οποίο μπορεί να έχει πρόσβαση η μηχανή Docker. Το μητρώο Docker υποστηρίζει διάφορους μηχανισμούς ελέγχου ταυτότητας και εξουσιοδότησης για να διασφαλίσει την ασφάλεια των εικόνων. Για παράδειγμα, το Docker Hub επιτρέπει στους χρήστες να συνδεθούν χρησιμοποιώντας το Docker ID τους και τα ιδιωτικά μητρώα Docker μπορούν να ρυθμιστούν ώστε να απαιτούν έλεγχο ταυτότητας πριν επιτρέψουν την πρόσβαση στις εικόνες.

Dockerfile:Μια εικόνα Docker είναι ένα αυτοτελές, ελαφρύ και εκτελέσιμο πακέτο που περιέχει όλα τα απαραίτητα αρχεία, βιβλιοθήκες και ρυθμίσεις που απαιτούνται για την εκτέλεση μιας συγκεκριμένης εφαρμογής. Πρόκειται ουσιαστικά για ένα στιγμιότυπο ενός συγκεκριμένου περιβάλλοντος που περιλαμβάνει την εφαρμογή και όλες τις εξαρτήσεις της, συμπεριλαμβανομένου του λειτουργικού συστήματος και του χρόνου εκτέλεσης, συγκεντρωμένα σε ένα ενιαίο πακέτο. Οι εικόνες Docker κατασκευάζονται από ένα σύνολο οδηγιών που ονομάζεται Dockerfile, το οποίο καθορίζει τον τρόπο κατασκευής της εικόνας. Οι εικόνες Docker έχουν σχεδιαστεί ώστε να είναι φορητές και μπορούν να μεταφέρονται εύκολα μεταξύ διαφορετικών συστημάτων και περιβαλλόντων, επιτρέποντας στους προγραμματιστές να κατασκευάζουν, να δοκιμάζουν και να αναπτύσσουν εφαρμογές πιο αποτελεσματικά. Συνολικά, οι εικόνες Docker αποτελούν βασικό δομικό στοιχείο του οικοσυστήματος Docker, επιτρέποντας στους προγραμματιστές να δημιουργούν και να διανέμουν εφαρμογές με τυποποιημένο, επαναλαμβανόμενο και κλιμακούμενο τρόπο.

Docker-compose: Το Docker Compose είναι ένα εργαλείο που χρησιμοποιείται για τη διαχείριση πολλαπλών εφαρμογών σε περιβάλλοντα Docker. Αποτελείται από ένα αρχείο YAML που περιέχει τις πληροφορίες για τις διαφορετικές υπηρεσίες και τις ρυθμίσεις που αφορούν το περιβάλλον εκτέλεσης τους. Το Docker Compose επιτρέπει στους προγραμματιστές να ορίζουν, να εκτελούν και να συντονίζουν πολλαπλές εφαρμογές σε ένα μόνο περιβάλλον. Με τη χρήση του Docker Compose, οι προγραμματιστές μπορούν να ορίσουν τις εξαρτήσεις των διαφορετικών εφαρμογών και να επιτρέψουν στο Docker να τις εκτελέσει με συντονισμένο τρόπο. Μια από τις πιο χρήσιμες λειτουργίες του είναι η δυνατότητα να ορίσει πολλαπλά περιβάλλοντα (όπως development, staging, production) και να τα διαχειρίζεται αυτόματα.

Στο αρχείο docker-compose περιλαμβάνονται οι ακόλουθες πληροφορίες: Ορισμός των υπηρεσιών, που περιγράφει τις εφαρμογές που θα εκτελεστούν και ποια εικόνα Docker θα χρησιμοποιηθεί για τη δημιουργία τους. Επίσης, περιγράφει τις παραμέτρους που χρειάζονται για την εκτέλεση της υπηρεσίας, όπως πόρτες (ports) και μεταβλητές περιβάλλοντος (environment variables). Περιλαμβάνεται επίσης ο ορισμός του δικτύου (network) που περιγράφει το δίκτυο που θα χρησιμοποιηθεί για τη σύνδεση των υπηρεσιών μεταξύ τους. Μπορεί να δημιουργηθεί ένα νέο δίκτυο ή να χρησιμοποιηθεί ένα υπάρχον

δίκτυο, ο ορισμός των όγκων (volumes) που περιγράφει τα δεδομένα που πρέπει να διατηρηθούν μεταξύ των εκτελέσεων της εφαρμογής. Μπορεί να χρησιμοποιηθεί για τη δημιουργία ενός κοινόχρηστου όγκου (shared volume) για τον διαμοιρασμό δεδομένων μεταξύ των container. Τέλος οι μεταβλητές περιβάλλοντος οι οποίες μπορούν να οριστούν για κάθε container, και μπορούν να χρησιμοποιηθούν για να ρυθμίσουν τον container κατά τη διάρκεια της εκτέλεσης του.

Η χρήση του Docker παρουσιάζει ένα σύνολο από πλεονεκτήματα στις εφαρμογές του. Αρχικά δίνεται η δυνατότητα της φορητότητας καθώς οι container του Docker είναι εξαιρετικά φορητοί και μπορούν να εκτελούνται σε οποιοδήποτε μηχάνημα που υποστηρίζει το Docker. Αυτό καθιστά εύκολη την ανάπτυξη εφαρμογών σε διαφορετικά περιβάλλοντα, όπως αυτό της ανάπτυξης της εφαρμογής, της δοκιμής της αλλά και στο περιβάλλον της παραγωγής. Ένα ακόμα πλεονέκτημα είναι αυτό της επεκτασιμότητας. Το Docker επιτρέπει την κλιμάκωση των εφαρμογών εκτελώντας πολλαπλές περιπτώσεις εκδόσεις και μορφές που έχουν ως βάση τον ίδιο container σε έναν μόνο εξυπηρετητή ή σε πολλαπλούς εξυπηρετητές. Αυτό ενισχύει κυρίως τη διανομή του φορτίου και στην εξασφάλιση υψηλής διαθεσιμότητας. Όσον αφορά την αποδοτικότητα που προσδίδει σε ένα σύστημα η εκτέλεση εφαρμογών σε container, παρατηρείται πολύ λιγότερη κατανάλωση πόρων σε σύγκριση με τις κλασικές εικονικές μηχανές. Η παραπάνω συνθήκη καθιστά δυνατή την εκτέλεση περισσότερων container και κατ' επέκταση εφαρμογών στον ίδιο κεντρικό υπολογιστή και βελτιώνει τη χρήση των πόρων. Όσον αφορά τα θέματα ασφαλείας, το Docker παρέχει διάφορα χαρακτηριστικά ασφαλείας, όπως απομόνωση χώρου ονομάτων, περιορισμό πόρων και έλεγχο πρόσβασης. Υποστηρίζει επίσης σάρωση εικόνας για τον εντοπισμό ευπαθειών και ζητημάτων συμμόρφωσης στην εικόνα του container.

3.4.2 Postman

Οι Διεπαφές Εφαρμογών Προγραμματισμού (Application Programming Interfaces - APIs) αποτελούν ένα σημαντικό κομμάτι της διασύνδεσης και επικοινωνίας μεταξύ των εφαρμογών αλλά και μεταξύ του χρήστη και των εφαρμογών. Ειδικότερα ορίζεται ένα πλαίσιο επιτρεπόμενων μεθόδων τις οποίες ο χρήστης ή η εφαρμογή μπορεί να χρησιμοποιήσει ώστε να αλληλεπιδράσει με ένα τελικό κόμβο ο οποίος αντίστοιχα για τον οποίο αντίστοιχα έχει καθοριστεί ένα σύνολο από μεθόδους. Η αλληλεπίδραση αυτή μπορεί να αφορά μια αλλαγή που αιτείται να πραγματοποιηθεί ή ακόμα και μια απλή ανάγνωση δεδομένων. Αρχικά με την εξέλιξη του διαδικτύου ο ρόλος του ήταν ιδιαίτερα σημαντικός καθώς εφαρμόστηκε στην διασύνδεση του μοντέλου πελάτη-εξυπηρετητή όπου στην περίπτωση αυτή από την μεριά του πελάτη εκτελούνται αιτήματα τα οποία απευθύνονται στον εξυπηρετητή ο οποίος με τη σειρά του στέλνει την απάντηση για το κάθε αίτημα που λαμβάνει μέσω της δυνατότητας αυτής της διεπαφής [54]. Η αύξηση του αριθμού των εφαρμογών αλλά και οι στενές αλληλεξαρτήσεις μεταξύ τους που οδηγούν στην συνεχή επικοινωνία, δημιουργούν την απαίτηση για συνεχή ανταλλαγή δεδομένων και έτσι οι διεπαφές εφαρμογών προγραμματισμού χρησιμοποιούνται και σε άλλα σενάρια.

Το Postman είναι ένα δημοφιλές εργαλείο ανάπτυξης και δοκιμής API που παρουσιάστηκε για πρώτη φορά το 2012 από τους Abhinav Asthana και Ankit Sobti. Έκτοτε έχει γίνει ένα κρίσιμο εργαλείο στο οπλοστάσιο των προγραμματιστών, χάρη στη φιλική προς το χρήστη διεπαφή του και το ευρύ φάσμα χαρακτηριστικών του. Στον πυρήνα του, το Postman είναι ένας πελάτης HTTP που επιτρέπει στους προγραμματιστές να στέλνουν αιτήματα σε API και να λαμβάνουν απαντήσεις. Λειτουργεί δηλαδή ως ένας μεσολαβητής μεταξύ του πραγματικού πελάτη και του εξυπηρετητή προσφέροντας δυνατότητες

διαμόρφωσης και καλύτερης διαχείρισης του κάθε αιτήματος. Ένα από τα βασικά πλεονεκτήματα του Postman είναι η ικανότητά του να στέλνει αιτήματα χρησιμοποιώντας διάφορες μεθόδους HTTP, όπως GET, POST, PUT, DELETE και PATCH. Οι προγραμματιστές μπορούν επίσης να προσθέτουν κεφαλίδες, παραμέτρους και έλεγχο ταυτότητας στα αιτήματά τους, διαμορφώνοντας εξ ολοκλήρου το κάθε αίτημα και καθιστώντας ευκολότερη τη δοκιμή και την αποσφαλμάτωση διαφορετικών σεναρίων.

Το Postman είναι επίσης εξαιρετικά προσαρμόσιμο και ευέλικτο ως εργαλείο, καθώς επιτρέπει τη δημιουργία και την αποθήκευση αιτημάτων για μελλοντική χρήση. Η λειτουργία αυτή είναι πολύ σημαντική καθώς αφενός μελλοντικά τα αιτήματα μπορούν να ξαναχρησιμοποιηθούν εξοικονομώντας τον χρόνο που χρειάζεται για την αναζήτηση πληροφοριών που συντελούν στην ανασύνταξή τους αλλά και αφετέρου δημιουργούν μια πιο ξεκάθαρη εικόνα για την ροή των αιτημάτων που χρησιμοποιούνται για την κάθε εφαρμογή. Τα αιτήματα που αποθηκεύονται, μπορούν να οργανωθούν σε συλλογές, καθιστώντας εύκολη τη διαχείριση και την κοινή χρήση API μεταξύ των μελών της ομάδας. Οι συλλογές μπορούν επίσης να εξαχθούν και να εισαχθούν σε διάφορες μορφές, όπως JSON και YAML, καθιστώντας εύκολη την κοινή χρήση με άλλους προγραμματιστές. Προσφέρει επίσης μια λειτουργία που ονομάζεται ”περιβάλλοντα”, η οποία επιτρέπει στους προγραμματιστές να εναλλάσσονται μεταξύ διαφορετικών περιβαλλόντων, όπως ανάπτυξη, δοκιμή και παραγωγή. Αυτό διευκολύνει τη δοκιμή των APIs σε διαφορετικά περιβάλλοντα και διασφαλίζει ότι τα APIs λειτουργούν σωστά σε διαφορετικά περιβάλλοντα.

Ένα άλλο σημαντικό πλεονέκτημα του Postman είναι η δυνατότητά του να αυτοματοποιεί την διαδικασία που αφορά στις δοκιμές. Οι προγραμματιστές μπορούν να δημιουργήσουν αυτοματοποιημένες δοκιμές για τα API τους, οι οποίες μπορούν να εκτελούνται αυτόματα και να εντοπίζουν πιθανά προβλήματα που μπορεί να προκύψουν. Το Postman προσφέρει επίσης οπτικές δοκιμές παλινδρόμησης, οι οποίες επιτρέπουν στους προγραμματιστές να συγκρίνουν οπτικά τις αποκρίσεις από διαφορετικές εκδόσεις API. Εκτός από τις δυνατότητες δοκιμών, το Postman ενσωματώνεται επίσης με άλλα εργαλεία και υπηρεσίες που χρησιμοποιούνται συνήθως στη διαδικασία ανάπτυξης. Το Postman διαθέτει επίσης μια ενεργή κοινότητα χρηστών που συμβάλλουν στην ανάπτυξή του με διάφορους τρόπους, όπως η δημιουργία πρόσθετων προγραμμάτων και η παροχή ανατροφοδότησης. Η ίδια η ομάδα του Postman παρέχει επίσης τακτικές ενημερώσεις και διορθώσεις σφαλμάτων, διασφαλίζοντας ότι το εργαλείο παραμένει σχετικό και ενημερωμένο.

Συμπερασματικά, το Postman είναι ένα απαραίτητο εργαλείο για τους προγραμματιστές που εργάζονται με APIs. Η φιλική προς το χρήστη διεπαφή του, οι επιλογές που προσφέρει για την απλοποίηση όλου του κύκλου ζωής των διεπαφών εφαρμογών προγραμματισμού αλλά και οι επιλογές προσαρμογής και το ευρύ φάσμα χαρακτηριστικών το καθιστούν απαραίτητο εργαλείο. Όλες οι παραπάνω ιδιότητες που αναφέρθηκαν αλλά και τα χαρακτηριστικά που περιγράφηκαν το κατέστησαν ένα πολύ χρήσιμο εργαλείο στην εργασία αυτή στο κομμάτι της υλοποίησης που περιγράφεται στο κεφάλαιο που ακολουθεί.

3.5 Επίλογος

Στο κεφάλαιο αυτό περιγράφηκαν τα κύρια εργαλεία που χρησιμοποιήθηκαν για την εκπόνηση της παρούσας εργασίας που αφορούν κυρίως στο πρακτικό κομμάτι της υλοποίησης. Το θεωρητικό πλαίσιο που δόθηκε είναι απαραίτητο να καλυφθεί ώστε να μπορεί να γίνει πιο εύκολη η κατανόηση όλων των βημάτων του τεχνικού μέρους που ακολουθεί.

Κεφάλαιο 4ο: Υλοποίηση Σεναρίου

Στο παρόν κεφάλαιο κύριος στόχος είναι η περιγραφή του πειράματος το οποίο αποφάσισα να εκτελέσω με στόχο την απόδειξη της πραγματοποίησης μετεγκατάστασης μιας υπηρεσίας σε περίπτωση απρόβλεπτου συμβάντος στην πλατφόρμα. Το πρώτο μέρος αφορά την διαδικασία δημιουργίας ενός σεναρίου δημοσίευσης-εγγραφής (publisher-subscriber) γραμμένο στη γλώσσα προγραμματισμού python. Η υπηρεσία που αποτελεί το κεντρικότερο στοιχείο του σεναρίου είναι ο subscriber ο οποίος και θέλουμε αρχικά να πραγματοποιήσει εγγραφή στην πλατφόρμα EdgeX Foundry. Πιο συγκεκριμένα περιγράφονται αναλυτικά όλα τα στάδια δημιουργίας του περιβάλλοντος, της υπηρεσίας αλλά και της διαδικασίας εγγραφής στην πλατφόρμα αποδεικνύοντας κάθε στάδιο εγγραφής-αποστολής και ανάγνωσης των δεδομένων.

Στη συνέχεια, το δεύτερο μέρος επικεντρώνεται κυρίως στην διαδικασία που ακολουθήθηκε ώστε να γίνει αποτελεσματικά η μετεγκατάσταση στην πλατφόρμα της Siemens περιγράφοντας όλα τα ενδιάμεσα στάδια αλλά και την τελική μορφή λειτουργίας του σεναρίου. Το σενάριο που πραγματοποιήθηκε αντιπροσωπεύει μια κατάσταση η οποία μπορεί να παρουσιαστεί και σε πραγματικό περιβάλλον καθώς αφορά μια απλή εφαρμογή publisher-subscriber με τη χρήση πρωτοκόλλου MQTT και οι πλατφόρμες που επιλέχθηκαν χρησιμοποιούνται ευρέως στον τομέα των μικρο-υπηρεσιών και του βιομηχανικού διαδικτύου των πραγμάτων. Όσον αφορά τους λόγους για τους οποίους μπορεί να ενεργοποιηθεί η διαδικασία της μετεγκατάστασης επιλέγονται δύο σενάρια που αφορούν προβλήματα σύνδεσης με την πλατφόρμα και υπερφόρτωση των πόρων της πλατφόρμας αντίστοιχα. Στόχος των δύο αυτών επιλογών είναι να καλυφθεί ένα ευρύ φάσμα σημαντικών προβλημάτων για τα οποία η διαδικασία της μετεγκατάστασης παρουσιάζει σημαντική λύση.

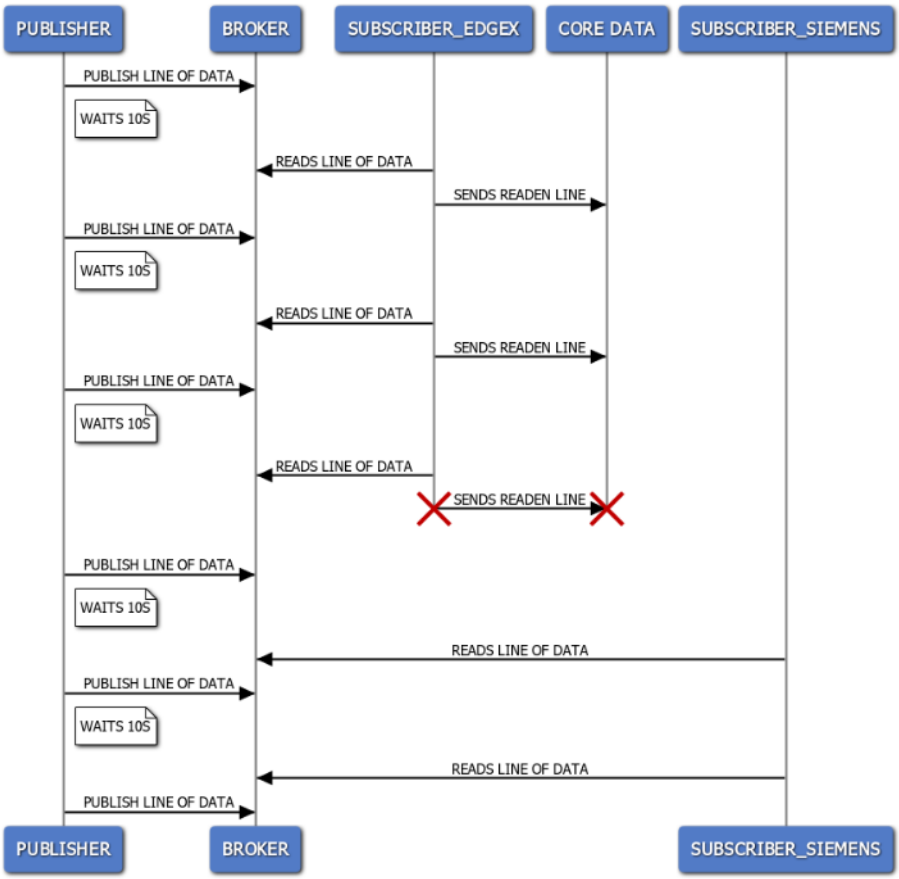
Έπειτα από την ανάλυση της υλοποίησης και των δύο σεναρίων, παρουσιάζεται το σύνολο των αποτελεσμάτων που προκύπτουν ύστερα από την εφαρμογή πειραμάτων τα οποία επικεντρώνονται κυρίως σε μετρήσεις χρονικών διαστημάτων και συγκεκριμένα, στον χρόνο που μεσολαβεί για την αντίστοιχη μετεγκατάσταση σε δύο διαφορετικά είδη συσκευών αλλά και του χρόνου τον οποίο χρειάζεται μια μικρο-υπηρεσία σε περίπτωση βλάβης να επαναλειτουργήσει στην ίδια πλατφόρμα. Στόχος των μετρήσεων αυτών είναι η λεπτομερής μελέτη των σεναρίων που υλοποιούνται και η εξαγωγή χρήσιμων συμπερασμάτων.

4.1 Περιγραφή περίπτωσης χρήσης: Μετεγκατάσταση λόγω απώλειας σύνδεσης

Στο σενάριο που επέλεξα να δημιουργήσω κύριο στοιχείο αποτελεί η υλοποίηση της αρχιτεκτονικής publisher-subscriber κυρίως καθώς στο περιβάλλον του βιομηχανικού διαδικτύου των πραγμάτων χρησιμοποιείται παρέχοντας λύσεις σε σενάρια και εφαρμογές όπου η ανταλλαγή των μηνυμάτων επιθυμούμε να πραγματοποιείται ασύγχρονα. Το σενάριο αποτελείται από τρεις μικροϋπηρεσίες: έναν συνδρομητή (broker), έναν publisher και έναν subscriber. Ως πρωτόκολλο επικοινωνίας μεταξύ των μικρο-υπηρεσιών επιλέχθηκε το MQTT κυρίως γιατί το υποστηρίζουν και οι δύο πλατφόρμες, είναι από τα κυρίαρχα πρωτόκολλα επικοινωνίας στον κλάδο του βιομηχανικού διαδικτύου των πραγμάτων αλλά και επειδή δίνει την δυνατότητα επεκτασιμότητας, ένα χαρακτηριστικό σημαντικό στην περίπτωση των μικροϋπηρεσιών αλλά και στην εφαρμογή όλων των συστημάτων.

Για την πρώτη περίπτωση, όπως παρατηρούμε στην Εικόνα 4.1, αρχικά ο subscriber βρίσκεται εγκατεστημένος στην πλατφόρμα EdgeX Foundry. Για όσο δεν παρουσιάζεται κάποιο πρόβλημα, η υπηρεσία συνεχίζει και τρέχει κανονικά στην πλατφόρμα εκτελώντας την λειτουργία για την οποία προορίζεται. Πιο συγκεκριμένα η λειτουργία αυτή αφορά στην ανάγνωση δεδομένων από τον συνδρομητή και έπειτα την αποστολή τους στην μικρο-υπηρεσία της πλατφόρμας EdgeX, core-data. Τα δεδομένα που διαβάζει ο subscriber έχουν σταλεί από μια δεύτερη υπηρεσία τον publisher. Ο publisher στην περίπτωση αυτή αποτελεί την προσομοίωση ενός αισθητήρα ο οποίος αποστέλλει δεδομένα θερμοκρασίας με έναν συγκεκριμένο χρόνο μεσολάβησης των 10 δευτερολέπτων ανά αποστολή.

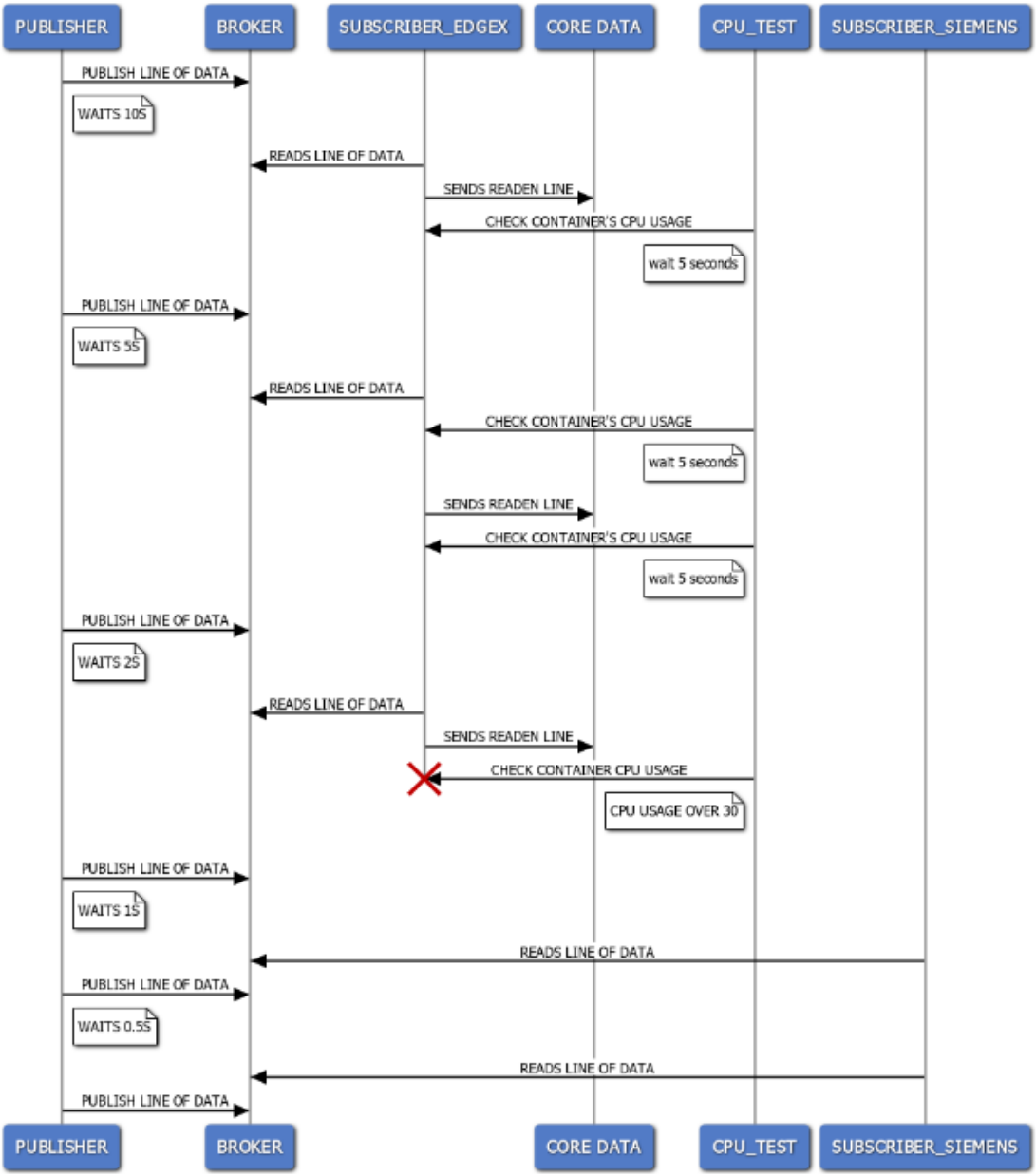
Τη χρονική στιγμή X προκαλείται μια βλάβη στην πλατφόρμα η οποία έχει ως αποτέλεσμα την υπολειτουργία της και κατ' επέκταση την απώλεια σύνδεσης και επικοινωνίας του subscriber με την υπηρεσία core-data. Στο σημείο αυτό ενεργοποιείται η αυτόματη διαδικασία που έχει δημιουργηθεί έτσι ώστε τελικά ο subscriber να εγκατασταθεί εκ νέου στην δεύτερη πλατφόρμα της SIEMENS με στόχο την ομαλή επαναλειτουργία του μοντέλου. Μόλις ολοκληρωθεί η διαδικασία της εγκατάστασης η υπηρεσία συνεχίζει κανονικά να εκτελεί τις λειτουργίες της και έτσι το ολικό σύστημα δεν επηρεάζεται καθόλου. Αντί να σταματήσει ο κύκλος της ανάγνωσης και αποστολής των δεδομένων το μοναδικό γεγονός που συμβαίνει στην περίπτωση βλάβης είναι μια μικρή περίοδος αναμονής που μεσολαβεί μέχρι την ολοκλήρωση της διαδικασία που αφορά την επανεγκατάσταση. Έτσι γίνεται αποφυγή της ολικής δυσλειτουργίας του συστήματος το οποίο σε κάποια άλλη περίπτωση θα μπορούσε να σταματήσει να δουλεύει αποτελεσματικά πλήρως χωρίς να επανέλθει σε σύντομο χρονικό διάστημα.



Εικόνα 4.1: Διάγραμμα ροής για περίπτωση χρήσης: Πρόβλημα σύνδεσης με core-data

4.2 Περιγραφή περίπτωσης χρήσης: Μετεγκατάσταση λόγω υπερφόρτωσης CPU της πλατφόρμας

Στην δεύτερη περίπτωση που παρουσιάζεται στην Εικόνα 4.2 αρχικά ο κύριος σκελετός της αρχιτεκτονικής και της ροής δεδομένων είναι ο ίδιος με προηγουμένως. Πιο συγκεκριμένα στο αρχικό στάδιο ο συνδρομητής εκτελείται κανονικά σε container στην εικονική ubuntu μηχανή και ξεκινάει την εκτέλεση του. Ο publisher τρέχει τοπικά στην τοπική μηχανή, εγγράφεται στον συνδρομητή και ξεκινάει να στέλνει δεδομένα στο θέμα ”tesi” με την διαφορά ότι αυτή την φορά η ανάγκη για πιο συχνή παρακολούθηση της θερμοκρασίας αυξάνεται και έτσι ανά διαστήματα μειώνεται ο ενδιάμεσος χρόνος κατά την αποστολή νέας μέτρησης. Ο subscriber εκτελείται σε μορφή container στην πλατφόρμα του EdgeX και ξεκινάει την ανάγνωση των μηνυμάτων στο ίδιο θέμα και κάνει προώθηση της κάθε γραμμής στην υπηρεσία core-data του EdgeX.



Εικόνα 4.2: Διάγραμμα ροής για περίπτωση χρήσης: Κατανάλωση Μνήμης

Ένα επιπλέον στοιχείο σε αυτή την περίπτωση είναι η υπηρεσία CPU_TEST η οποία ανά 5 δευτερόλεπτα ελέγχει την κατανάλωση επί τοις εκατό της Κεντρικής Μονάδας Επεξεργασίας (Central Process Unit - CPU) του container στον οποίο εκτελείται ο subscriber. Επιλέγω μια τυχαία χρονική στιγμή X, στην οποία μετά από τον έλεγχο που πραγματοποιείται παρατηρείται πως ο container έχει ξεπεράσει το 30% χρήσης της μνήμης. Την χρονική στιγμή αυτή ξεκινάει η μετεγκατάσταση του container στην πλατφόρμα της SIEMENS και παράλληλα σταματάει η λειτουργία του στην πλατφόρμα του EdgeX. Στόχος αυτής της δοκιμασίας είναι σε μια παρόμοια περίπτωση να υπάρχει δυνατότητα μετακίνησης της υπηρεσίας σε μια διαφορετική πλατφόρμα με περισσότερους πόρους και μεγαλύτερη επεξεργαστική ισχύ, ώστε να μην επιβαρύνεται το σύστημα και καταλήξει να λειτουργεί με μεγαλύτερη καθυστέρηση ή να υπερφορτωθεί και να σταματήσει να λειτουργεί ολοκληρωτικά.

Ακριβώς η ίδια διαδικασία και λογική της συγκεκριμένης περίπτωσης χρήσης είναι δυνατόν να υλοποιηθεί και με τον έλεγχο κατανάλωσης αποθηκευτικού χώρου που χρησιμοποιεί ο container της μικρουπηρεσίας του subscriber, μια μεταβλητή που επίσης είναι σημαντικό να ελεγχθεί. Για την περίπτωση αυτή μπορούμε να ελέγχουμε πότε ο container της υπηρεσίας ξεπερνάει ένα συγκεκριμένο επιτρεπτό όριο χρήσης αποθηκευτικού χώρου αλλάζοντας απλά την μεταβλητή που ελέγχεται. Τη στιγμή που το όριο αυτό ξεπεραστεί πραγματοποιείται η μετεγκατάσταση στην πλατφόρμα της SIEMENS. Στην περίπτωση αυτή το διάγραμμα ροής είναι ακριβώς το ίδιο με αυτό που παρουσιάζεται στην Εικόνα 4.2, με τη διαφορά ότι η υπηρεσία που ελέγχει ονομάζεται MEMORY_TEST και αντί για CPU OVER 30 έχουμε στην ετικέτα της συνθήκης MEMORY OVER 30%.

4.3 Περιβάλλον εργασίας

Στην παρακάτω υποενότητα παρουσιάζονται όλα τα στοιχεία τα οποία συντελούν το περιβάλλον εργασίας μας. Στο πρώτο μέρος περιγράφονται τα τρία μέρη της publisher-subscriber αρχιτεκτονικής όπως υλοποιήθηκαν για το συγκεκριμένο πείραμα. Παρατίθενται επίσης τα χαρακτηριστικά λογισμικού και υλικού από όλες τις συσκευές και υλικό που χρησιμοποιήθηκε. Τέλος όσον αφορά τις πλατφόρμες οι οποίες με τη σειρά τους αποτελούν ένα πολύ σημαντικό κομμάτι της υλοποίησης, εξηγούνται οι κύριοι λόγοι επιλογής τους.

4.3.1 Broker

Ο συνδρομητής (broker) που χρησιμοποιείται εκτελείται σε container του docker (docker container) που βασίζεται στην εικόνα (image) eclipse-mosquitto. Η συγκεκριμένη εικόνα καταλαμβάνει αποθηκευτικό χώρο 11.9MB και αποτελεί μια ανοιχτού κώδικα εφαρμογή εξυπηρετητή (server) που υλοποιεί τις εκδόσεις 5, 3.1.1, και 3.1 του MQTT πρωτοκόλλου. Ο container που τελικά τρέχει στην εικονική μηχανή καταλαμβάνει χώρο 8B στον δίσκο και 11.9MB εικονικό μέγεθος το οποίο περιλαμβάνει και την εικόνα. Η εικονική μηχανή που χρησιμοποιείται για την εκτέλεση του συνδρομητή είναι μια linux μηχανή ανάπτυξης η οποία χρησιμοποιείται χάρης το πανεπιστήμιο της Μπολόνια. Το λειτουργικό σύστημα της μηχανής είναι το Ubuntu και συγκεκριμένα η έκδοση Ubuntu 20.04.5 ενώ η IP διεύθυνση της είναι η 10.0.7.132. Όσον αφορά τα χαρακτηριστικά που σχετίζονται με το υλικό (hardware), αποτελείται από RAM 16GB, Κεντρική Μονάδα Επεξεργασίας με επεξεργαστή οχτώ πυρήνων και σκληρό δίσκο με 150 GB χωρητικότητα.

Με την εγκατάσταση της εικόνας δημιουργούνται τρεις φάκελοι, /mosquitto/config, που μπορεί να χρησιμοποιηθεί για την ρύθμιση παραμέτρων,/ mosquitto/data, για την μόνιμη αποθήκευση δεδομένων και τέλος /mosquitto/log για την διαχείριση των αρχείων καταγραφής. Για να μπορέσω να χρησιμοποιήσω τον συνδρομητή με την δική μου παραμετροποίηση αλλά και για να μπορώ να το διαχειρίζομαι ευκολότερα χρησιμοποιώ το πρότυπο (template) του αρχείου mosquitto.conf το οποίο βρίσκεται μέσα στον container του συνδρομητή και εφαρμόζω ορισμένες προσθήκες που εξυπηρετούν στην υλοποίηση. Για την δική μου περίπτωση χρειάστηκε να δηλώσω τα εξής χαρακτηριστικά [55]:

• **listener 1883**, με το οποίο καθορίζεται η συγκεκριμένη πόρτα 1883 στην οποία θα ακούει ο μεσολαβητής οποιαδήποτε εισερχόμενη σύνδεση δικτύου.

• **persistence true**, που δηλώνει ότι αποθηκεύονται τα δεδομένα εγγραφής, σύνδεσης και μηνυμάτων στον δίσκο κάθε φορά που κλείνει το mosquitto αλλά και περιοδικά ανάλογα με την τιμή που δηλώνεται στην μεταβλητή autosave_interval. Στο συγκεκριμένο σενάριο η τιμή αυτή δεν έχει οριστεί και συνεπώς τα δεδομένα αποθηκεύονται μόνο πριν γίνει έξοδος. Η αποθήκευση αυτή πραγματοποιείται σε μια βάση δεδομένων με όνομα mosquitto.db της οποίας η διαδρομή ορίζεται από την μεταβλητή persistence_location. Η τιμή που δηλώθηκε στη μεταβλητή αυτή είναι η προεπιλεγμένη, δηλαδή η κενή συμβολοσειρά και συνεπώς αναφερόμαστε στον τρέχον φάκελο. Έτσι κάθε φορά που θα γίνεται επανεκκίνηση του mosquitto όλα τα αποθηκευμένα δεδομένα ξαναφορτώνονται χωρίς καμία αλλοίωση.

• **allow_anonymous true**, καθώς δεν υπάρχουν άλλοι ακροατές δηλωμένοι στο αρχείο των παραμέτρων. Οι συνδέσεις επιτρέπονται μόνο από το τοπικό μηχάνημα. Επίσης εφόσον έχει δηλωθεί ως false η τιμή της μεταβλητής per_listener, η ρύθμιση αυτή ισχύει για όλους τους ακροατές.

Για να μπορέσω να χρησιμοποιήσω αυτό το αρχείο με τις προσθήκες των τιμών στις μεταβλητές που αναφέρθηκαν παραπάνω, το δηλώνω ως volume τη στιγμή που τρέχω τον container. Επιπλέον από προεπιλογή, ο container κληρονομεί τις ρυθμίσεις του Συστήματος Ονοματοδοσίας Διαδικτύου (Domain Name System - DNS) του κεντρικού υπολογιστή, όπως ορίζονται στο αρχείο διαμόρφωσης /etc/resolv.conf. Οι container που χρησιμοποιούν το προεπιλεγμένο δίκτυο bridge λαμβάνουν ένα αντίγραφο αυτού του αρχείου, ενώ οι container που χρησιμοποιούν προσαρμοσμένο δίκτυο χρησιμοποιούν τον ενσωματωμένο διακομιστή DNS του Docker, ο οποίος προωθεί τις εξωτερικές αναζητήσεις DNS στους διακομιστές DNS που έχουν διαμορφωθεί στον κεντρικό υπολογιστή. Ο συγκεκριμένος container τρέχει στο προεπιλεγμένο δίκτυο της εικονικής μηχανής με όνομα bridge του οποίου η subnet IP είναι η 172.17.0.0 και η gateway 172.17.0.1. Η τελική εντολή που εκτελέστηκε στο docker ώστε να τρέχει ο συνδρομητής είναι η παρακάτω:

docker run -d -it -p 1883:1883 -p 9001:9001 -v $PWD/mosquitto.conf:/mosquitto/config/mosquitto.conf eclipse-mosquitto.

Με την παραπάνω εντολή ο συνδρομητής τρέχει σε container στην εικονική ubuntu μηχανή με IP διεύθυνση 172.17.0.2 χρησιμοποιώντας την εικόνα eclipse-mosquitto. Επιπλέον, οι πόρτες 1883 και 9001 γίνονται διαθέσιμες (exposed) και δηλώνεται ως volume το διαμορφωμένο αρχείο των παραμέτρων.

4.3.2 Publisher

Ο publisher αποτελεί την αναπαράσταση μιας IoT συσκευής αισθητήρα (sensor) η οποία διαβάζει ένα αρχείο csv το οποίο είναι αποθηκευμένο στην τοπική μηχανή. Το αρχείο csv αποτελείται από δεδομένα θερμοκρασίας τα οποία αφορούν διάστημα ενός μήνα και προκύπτουν ύστερα από μετρήσεις ενός αισθητήρα θερμοκρασίας λαδιού. Η μορφή του αρχείου προέκυψε έπειτα από μετασχηματισμό του αυθεντικού αρχείου δεδομένου το οποίο υπάρχει στο github και είναι διαθέσιμο και δημόσιο. Ένα δείγμα από το αρχείο παρατίθεται παρακάτω και όπως είναι διακριτό αποτελείται από δύο στήλες, η μία συμπεριλαμβάνει την ακριβή ημερομηνία και ώρα και η δεύτερη στήλη την τιμή της εκάστοτε μέτρησης. Τέλος σε κάθε γραμμή οι δύο στήλες που αναφέρθηκαν παραπάνω χωρίζονται με κόμμα.

```
2017-10-27 21:44:19,120.1284408569336
2017-10-27 21:56:51,118.26673889160156
2017-10-27 22:08:16,116.5347671508789
2017-10-27 22:46:34,118.01374816894531
2017-10-27 23:07:20,109.9961166381836
2017-10-28 00:26:07,106.09107971191406
2017-10-28 00:54:11,105.38401794433594
2017-10-28 01:55:10,104.0347671508789
2017-10-28 02:35:04,115.12714385986328
2017-10-28 02:55:05,112.94759368896484
2017-10-28 03:10:16,107.75169372558594
2017-10-28 04:03:51,104.54722595214844
```

Η τοπική μηχανή στην οποία θα εκτελείται ο pubilsher χρησιμοποιεί το λειτουργικό σύστημα Windows 11 Home, έχει RAM με 16GB χωρητικότητα, σκληρό δίσκο SSD με 237GB χωρητικότητα και επεξεργαστή AMD Ryzen με έξι πυρήνες. Ο publisher δημιουργεί ένα θέμα στον συνδρομητή το οποίο αφορά το σενάριο και κάνει εγγραφή σε αυτόν με τα στοιχεία του ως νέος πελάτης (client). Αφού ολοκληρωθεί η διαδικασία αυτή με επιτυχία, ανά δέκα δευτερόλεπτα δημοσιεύει στον συνδρομητή στο topic που έχει δημιουργηθεί με όνομα tesi την κάθε γραμμή δεδομένων θερμοκρασίας. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την σύνταξη του publisher είναι η python και η βιβλιοθήκη paho.mqtt.client.

Παρακάτω είναι σημαντικό να γίνει η περιγραφή της σύνταξης αλλά και κατ' επέκταση των λειτουργιών που εκτελεί ο publisher. Αρχικά γίνεται η δήλωση της IP και της θύρας του συνδρομητή στον οποίο θα γίνονται τα δεδομένα publish. Έπειτα δηλώνονται το θέμα και ένα access token το οποίο μπορεί να χρησιμοποιηθεί προαιρετικά κατά την εγγραφή στον συνδρομητή ως ένας απλός τρόπος αυθεντικοποίησης. Στη συνέχεια δηλώνεται η μέθοδος on_publish(). Η μέθοδος αυτή καλείται όταν ένα μήνυμα που επρόκειτο να σταλεί χρησιμοποιώντας την κλήση publish() έχει ολοκληρώσει τη μετάδοση στον συνδρομητή. Για μηνύματα με επίπεδα QoS 1 και 2, αυτό σημαίνει ότι έχουν ολοκληρωθεί οι κατάλληλες χειραψίες(handshakes) ενώ για QoS 0, αυτό σημαίνει απλώς ότι το μήνυμα έφυγε από τον πελάτη. Χρησιμοποιούνται τρεις μεταβλητές (client,userdata και result). Η μεταβλητή result ταιριάζει με τη μεταβλητή που επιστρέφεται από την αντίστοιχη κλήση publish() για να επιτρέπεται η παρακολούθηση των εξερχόμενων μηνυμάτων. Αυτή η επανάκληση (callback) είναι σημαντική γιατί ακόμα κι αν η κλήση publish() επιστρέψει με επιτυχία, δεν σημαίνει πάντα ότι το μήνυμα έχει σταλεί.

Στην συνέχεια δημιουργείται ένα αντικείμενο τύπου client, πραγματοποιείται αντιστοίχηση με την συνάρτηση που δημιουργήθηκε πριν με την επανάκληση, ορίζω ως κωδικό το access token και εγκαθιδρύεται η σύνδεση με τον συνδρομητή. Τέλος δημιουργώ έναν ατέρμονο βρόγχο επανάληψης στον οποίο ανοίγω μία φορά το αρχείο με τα δεδομένα και για κάθε γραμμή του αρχείου αποθηκεύεται το περιεχόμενο της γραμμής στην μεταβλητή payload και γίνεται η δημοσίευση της στο θέμα tesi ορίζοντας qos = 1. Η δήλωση αυτή εξυπηρετεί στο γεγονός της επιβεβαίωσης πως το κάθε μήνυμα θα παραδοθεί τουλάχιστον μια φορά στον παραλήπτη αποτελεσματικά. Ο publisher αποθηκεύει το εκάστοτε μήνυμα μέχρι να λάβει ένα πακέτο PUBACK από τον συνδρομητή που επιβεβαιώνει την παραλαβή του μηνύματος. Η επανάληψή του βρόγχου και κατ' επέκταση η αποστολή του κάθε μηνύματος εκτελείται από τον publisher κάθε 10 δευτερόλεπτα.

4.3.3 Subscriber

Ο subscriber αποτελεί το κύριο κομμάτι της υλοποίησης ως υπηρεσία καθώς είναι αυτό το οποίο αργότερα θα γίνει επανεγκατάσταση (migration) και του οποίου η γενική δομή παρουσιάζεται παρακάτω. Οι λειτουργίες που εκτελεί είναι η εγγραφή στο θέμα που αφορά την εργασία (tesi), η ανάγνωση/εκτύπωση των ληφθέντων δεδομένων και η αποστολή αυτών στην αντίστοιχη πλατφόρμα. Το θέμα στο οποίο εγγράφεται είναι το ίδιο με εκείνο που έχει δημιουργηθεί προγενέστερα από τον publisher.

Η υλοποίηση της υπηρεσίας του subscriber αλλά και τα χαρακτηριστικά του περιγράφονται παρακάτω. Αρχικά η γλώσσα προγραμματισμού και αυτή τη φορά είναι η python και οι βιβλιοθήκες που δηλώνονται είναι οι paho.mqtt.client, requests, json, uuid, os και sys. Αυτή τη φορά δημιουργείται η συνάρτηση onmessage() με μεταβλητές client, userdata και message. Η συνάρτηση αυτή εκτυπώνει το ληφθέν μήνυμα όταν εισέρχεται, καθώς και το θέμα στο οποίο δημοσιεύτηκε. Η μεταβλητή client που δηλώνεται ως παράμετρος, αντιστοιχεί στο αντικείμενο τύπου client που δημιουργούμε για τον subscriber και η μεταβλητή message αντιστοιχεί στο ληφθέν μήνυμα. Μέσα στη συνάρτηση αυτή έχει δημιουργηθεί και μια μεταβλητή payload στην οποία αποθηκεύεται το μήνυμα που αναγνώστηκε σε μια ιδιαίτερη μορφή η οποία θα χρησιμοποιηθεί στην περίπτωση που η εφαρμογή τρέχει στην πλατφόρμα Edgex και είναι απαραίτητη η αποστολή τους στην πλατφόρμα. Η δομή της μεταβλητής αυτής θα εξηγηθεί στην συνέχεια.

Για να μπορέσει να γίνει η εγγραφή στον συνδρομητή δημιουργείται και σε αυτήν την περίπτωση ένα αντικείμενο τύπου Client με όνομα "SUBSCRIBER" και γίνεται σύνδεση της συνάρτησης on_message με την επανάκληση. Έπειτα γίνεται εγκαθίδρυση της σύνδεσης με τον συνδρομητή σύμφωνα με τις τιμές που έχουμε ορίσει στην αρχή (οι ίδιες όπως περιγράφηκαν στον publisher) και γίνεται εγγραφή στο θέμα tesi. Από τη στιγμή αυτή υπάρχει ένα διάστημα αναμονής καθώς ο subscriber πλέον στέλνει αιτήματα στον συνδρομητή με σκοπό να διαβάσει τα μηνύματα που θα φτάσουν από τον publisher. Για όσο διάστημα ο publisher δεν δημοσιεύει κάποια καινούρια εγγραφή δεν υπάρχει καμία αλληλεπίδραση. Αντιθέτως με τον ερχομό ενός νέου μηνύματος, ο subscriber μετά το αντίστοιχο αίτημα του θα ενημερωθεί για το συγκεκριμένο μήνυμα και θα μπορεί να το αναγνώσει. Είναι σημαντικό να αναφερθεί πως όλες οι επανακλήσεις βασίζονται στον βρόγχο του πελάτη και αυτό πρέπει επίσης να έχει ξεκινήσει χρησιμοποιώντας την μέθοδο loop_start() ή την loop_forever(). Στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε το loop_forever() το οποίο βρίσκεται μέσα σε έναν ατέρμονο βρόγχο while True καθώς θέλουμε συνεχώς ο subscriber να ελέγχει για πιθανά νέα μηνύματα.

Η υπηρεσία του subscriber εκτελείται και αυτή σε container του docker. Η επιλογή αυτή έγινε έτσι ώστε να υπάρχει η δυνατότητα εγκατάστασης της αργότερα στις πλατφόρμες του βιομηχανικού διαδικτύου των πραγμάτων καθώς όλες οι υπηρεσίες και εφαρμογές μέσα στις πλατφόρμες υφίστανται σε μορφή container. Για να επιτευχθεί ο στόχος αυτός αρχικά είναι απαραίτητη η δημιουργία ενός Dockerfile το οποίο αφορά αποκλειστικά την υπηρεσία αυτή και είναι αποθηκευμένο στον ίδιο φάκελο. Μέσα στο αρχείο αυτό δηλώνεται το python:3.8-slim-buster που είναι η εικόνα στην οποία βασίζεται ο subscriber και επιλέχθηκε κυρίως λόγω της γλώσσας προγραμματισμού που χρησιμοποιήθηκε. Επίσης γίνονται όλες οι απαραίτητες ενημερώσεις και εγκαταστάσεις πακέτων αλλά και των δηλωμένων απαιτήσεων του αρχείου requirements.txt έτσι ώστε το περιβάλλον να είναι ολοκληρωμένο και να περιλαμβάνει όλα τα στοιχεία που χρειάζεται η υπηρεσία. Το αρχείο requirements.txt περιέχει όλες τις βιβλιοθήκες που είναι υποχρεωτικό να εγκατασταθούν ώστε να τρέξει η υπηρεσία μέσα σε έναν container επιτυχώς. Τελική εγγραφή στο Dockerfile αποτελεί η εκτέλεση του αρχείου της εφαρμογής subscriber.py.

```
FROM python:3.8-slim-buster

RUN apt-get update
RUN apt-get install -y --no-install-recommends
RUN apt-get install -y iputils-ping
COPY . .
RUN pip install -r requirements.txt

CMD ["python","-u","subscriber.py"]
```

Τελευταία προϋπόθεση πριν την εκτέλεση του container αποτελεί η δημιουργία ενός docker compose αρχείου. Μέσα στο αρχείο αυτό υπάρχουν όλες οι πληροφορίες σχετικά με την εκτέλεση του container. Δηλώνεται το όνομα του container του subscriber, μεταβλητές περιβάλλοντος που θα εξηγηθούν και θα σχολιαστούν στην πορεία, το dockerfile που χρησιμοποιείται για την εκτέλεση του container και η θέση στην οποία βρίσκεται και τέλος όλα τα δίκτυα στα οποία θα ανήκει ο container. Αφού σχηματιστεί το compose θα πρέπει να εκτελεστεί η εντολή docker-compose build έτσι ώστε να δημιουργηθεί το ανανεωμένο με κάθε εκτέλεση της εικόνας που αφορά τον subscriber.

```
subscriber-tesi:

    container_name: edgex-subscriber-tesi
    depends_on:
    - consul
    - data
    environment:
      REGISTRY_HOST: edgex-core-consul
    build: ./subscriber_service/

networks:
      edgex-network: {}
```

Για να μπορέσει να σχηματιστεί και να τρέξει ο container χρησιμοποιείται η εντολή docker-compose up αφού έχει γίνει αλλαγή θέσης στον φάκελο του subscriber από το PowerShell. Το docker αναγνωρίζει με την εντολή αυτή πως υπάρχει το compose αρχείο στον φάκελο που βρισκόμαστε καθώς δεν έχουμε ορίσει συγκεκριμένη διαδρομή.

4.3.4 Σχεδιαστικές επιλογές των πλατφορμών του βιομηχανικού διαδικτύου των πραγμάτων

Η πρώτη πλατφόρμα που επιλέχθηκε για το σενάριο της υλοποίησης είναι η EdgeX Foundry. Η επιλογή αυτή έγινε κυρίως γιατί είναι μια πλατφόρμα ανοιχτού λογισμικού η οποία χρησιμοποιείται ευρέως στον τομέα του βιομηχανικού διαδικτύου των πραγμάτων. Αν και φαίνεται να αποτελεί ένα αρκετά αυστηρό πλαίσιο όσον αφορά τον τρόπο λειτουργίας και τις νέες προσθήκες, καθώς όλες οι καινούριες υπηρεσίες έχουν συγκεκριμένη δομή, έχει ενδιαφέρον να γίνει δοκιμή λειτουργίας και με μια υπηρεσία που έχει διαφορετικά χαρακτηριστικά. Επιπλέον τόσο η αρχιτεκτονική της πλατφόρμας όσο και τα επιμέρους στοιχεία της που περιγράφηκαν σε προηγούμενο κεφάλαιο αποδεικνύουν πως αποτελεί μια ολοκληρωμένη λύση για σενάρια όπως αυτό. Στο σενάριο μας όλα τα στοιχεία είναι κατανεμημένα σε διαφορετικά επίπεδα. Ο subscriber θέλουμε να βρίσκεται στο άκρο του δικτύου (edge) κάτι το οποίο επιτυγχάνεται καθώς και η πλατφόρμα λειτουργεί σε αυτό το επίπεδο. Τέλος ένα ακόμα χρήσιμο χαρακτηριστικό αποτελεί η αλληλεπίδραση μεταξύ των υπηρεσιών σε πραγματικό χρόνο αφού το κομμάτι της υλοποίησης θέλουμε να αντικατοπτρίζει όσον το δυνατόν καλύτερα ένα πραγματικό σενάριο.

Η δεύτερη επιλογή πλατφόρμας είναι αυτή της Siemens. Αναφερόμαστε πλέον σε ένα τελείως διαφορετικό περιβάλλον τόσο δομικά όσο και λειτουργικά. Αρχικά έχει ενδιαφέρον η σύγκριση με μια επίσης κυρίαρχη πλατφόρμα, αυτή τη φορά όμως στο κομμάτι των ιδιόκτητων υπηρεσιών, αφού είναι από τις πιο χρησιμοποιούμενες στην αγορά για σενάρια που αφορούν την Βιομηχανία 4.0. Δομικά η αρχιτεκτονική της είναι εντελώς διαφορετική σε σχέση με την προηγούμενη πλατφόρμα και επίσης αποτελείται από ένα επιπλέον επίπεδο το MANAGEMENT. Και αυτή η πλατφόρμα εκτελείται στα άκρα του δικτύου το οποίο είναι χρήσιμο για τις απαιτήσεις του σεναρίου καθώς μια από τις προϋποθέσεις που τίθενται είναι η υπηρεσία του subscriber να εκτελείται στο άκρο. Επιπλέον η συγκεκριμένη πλατφόρμα απαιτεί μια ιδιαίτερη διαδικασία στα αρχικά στάδια ώστε να μπορέσει να δεχθεί στην συνέχεια την εγκατάσταση μιας νέας υπηρεσίας. Παρέχει επίσης δύο διαφορετικά είδη edge συσκευών, χαρακτηριστικό που επιτρέπει την διερεύνηση δύο διαφορετικών περιπτώσεων. Επιπλέον το γεγονός ότι στην πλατφόρμα αυτή όλες οι μικροϋπηρεσίες εκτελούνται σε μορφή container είναι εξαιρετικά χρήσιμο καθώς η υπηρεσία που πρόκειται να εγκατασταθεί είναι εκτελείται μέσα σε container.

Συνεπώς και τα δύο περιβάλλοντα ικανοποιούν τις ανάγκες των σεναρίων και καθώς παρουσιάζουν σημαντικές διαφορές είναι δυνατόν να αποδειχθεί πως υπάρχει ευελιξία όσον αφορά την διαδικασία μιας μετεγκατάστασης ανεξάρτητα από την μορφή και τις διαφορές που παρουσιάζουν οι δύο πλατφόρμες.

4.4 Περιγραφή Υλοποίησης

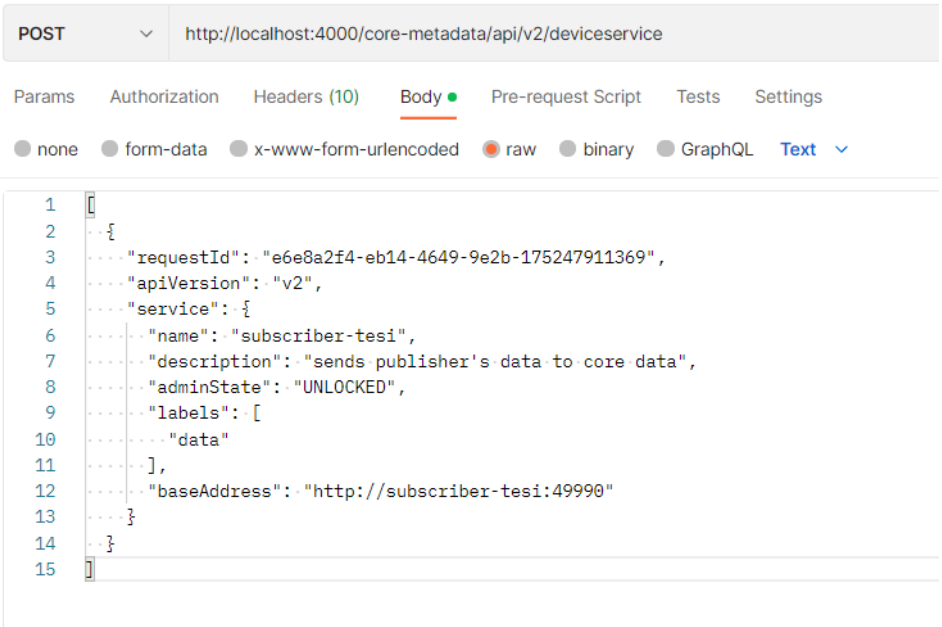
Σε αυτή την υποενότητα παρουσιάζονται και περιγράφονται όλα τα βήματα της υλοποίησης των σεναρίων που περιγράφηκαν θεωρητικά παραπάνω στην υποενότητα 8.2. Σε κάθε βήμα παρατίθενται επίσης εικόνες που επεξηγούνται και βοηθούν στην απεικόνιση των σταδίων. Η υλοποίηση του σεναρίου αναλύεται χρονικά με στιγμή εκκίνησης την προσομοιωμένη εγκατάσταση του subscriber στην πρώτη

πλατφόρμα EdgeX. Ακολουθεί η διαδικασία της μετεγκατάστασης και πως αυτή επιτεύχθηκε και στη συνέχεια περιγράφονται οι απαιτήσεις για την μετεγκατάσταση στην πλατφόρμα της SIEMENS και πως αυτή πραγματοποιείται τελικά αυτοματοποιημένα. Τέλος αναλύεται η λειτουργία του συστήματος που έχω δημιουργήσει στο περιβάλλον της νέας πλατφόρμας.

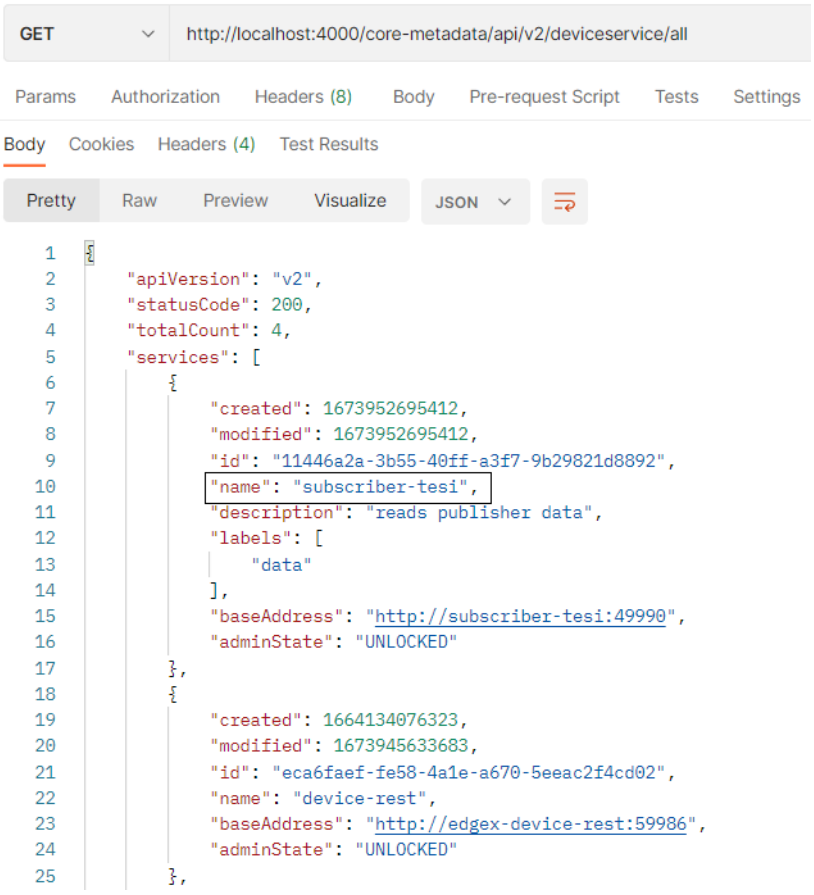
4.4.1 Περιγραφή προσομοιωμένης εγκατάστασης στην πλατφόρμα EdgeX

Στην πρώτη φάση της υλοποίησης ο subscriber εκτελείται μέσα σε container στην πλατφόρμα του EdgeX. Για να επιτευχθεί ο στόχος αυτός, είναι απαραίτητη η εγγραφή της υπηρεσίας στην πλατφόρμα. Ο subscriber σύμφωνα με την αρχιτεκτονική του EdgeX ανήκει στο πρώτο επίπεδο των device services καθώς θα είναι υπεύθυνος για την ανάγνωση των δεδομένων από την IoT συσκευή μας(publisher) και έπειτα για την αποστολή τους στα επόμενα επίπεδα της πλατφόρμας. Σε αυτό το επίπεδο οι μικρο-υπηρεσίες σύμφωνα με το EdgeX δημιουργούνται βάση ενός προϋπάρχοντος Κιτ Ανάπτυξης Λογισμικού (Software Development Kit - SDK).

Το συγκεκριμένο SDK παρέχει έναν ειδικό σκελετό για τον κώδικα και τις μεθόδους που αφορούν τις μικρο-υπηρεσίες συσκευών. Ωστόσο επειδή ο subscriber έχει διαφορετική δομή αποφάσισα να μην χρησιμοποιήσω το SDK που παρέχεται και να ακολουθήσω τις προϋποθέσεις που ορίζουν πότε ένα device service εγγράφεται και εκτελείται επιτυχώς στην πλατφόρμα. Πρώτη προϋπόθεσή για την εγγραφή στην πλατφόρμα είναι η δήλωση μέσω της υπηρεσίας core-metadata. Με αυτόν τον τρόπο γνωστοποιείται σε όλες τις υπηρεσίες της πλατφόρμας η νέα προσθήκη και έτσι γνωρίζουν την κατάσταση της και πιο συγκεκριμένα εάν εκτελείται ή βρίσκεται σε κατάσταση διακοπής. Στην Εικόνα 4.3 παρουσιάζεται το REST API αίτημα που στέλνεται στην υπηρεσία core-metadata και χρησιμοποιείται για την εγγραφή του subscriber. Αξίζει να σημειωθεί πως ολόκληρο το API παρουσιάζεται στην ιστοσελίδα [56]. Στο σώμα του αιτήματος περιλαμβάνονται πληροφορίες όπως το όνομα της νέας υπηρεσίας, το οποίο πρέπει να είναι και μοναδικό, η περιγραφή της αλλά και η διεύθυνση της.



Εικόνα 4.3: Εγγραφή υπηρεσίας subscriber στο core-metadata



Εικόνα 4.4: Λίστα των device services

Μπορούμε να διαπιστώσουμε πως η διαδικασία έχει ολοκληρωθεί επιτυχώς πραγματοποιώντας ένα ακόμα GET αίτημα στην υπηρεσία core-metadata **http://localhost:4000/core-metadata/api/v2/deviceservice/all**. Όπως φαίνεται στην Εικόνα 4.4 η οποία αποτελεί ένα μέρος στιγμιότυπου από την απάντηση του αιτήματος, η υπηρεσία μας πλέον υπάρχει στην λίστα των device-services.

Επόμενο βήμα αποτελεί η ένταξη του subscriber στο ίδιο δίκτυο με τις υπόλοιπες μικρο-υπηρεσίες. Κάτι τέτοιο είναι απαραίτητο έτσι ώστε να μπορεί να υπάρχει επικοινωνία με όλους τους container και να μπορεί να αλληλεπιδρά με την υπηρεσία core-data με REST API αιτήματα. Η υπηρεσία core-data είναι η πιο σημαντική καθώς είναι υπεύθυνη για την διαχείριση της ροής δεδομένων στο σύστημα και έτσι θεωρούμε πως εφόσον υπάρχει αναγνώριση και ορθή επικοινωνία με την μικροϋπηρεσία αυτή, ο subscriber επικοινωνεί επιτυχώς με το περιβάλλον του EdgeX. Η ένταξη στο δίκτυο δηλώνεται μέσω του docker-compose του subscriber. Με την εγκατάσταση της πλατφόρμας δημιουργείται το δίκτυο στο οποίο θα τρέχουν όλες οι υπηρεσίες και δηλώνεται στο compose αρχείο του EdgeX.

Το δίκτυο που προκύπτει τελικά με την εκτέλεση της πλατφόρμας συμπεριλαμβάνει και το όνομα του φακέλου και έτσι το τελικό όνομα είναι mp_tesi_edgex-network. Για να μπορέσει ο subscriber να τρέξει στο συγκεκριμένο δίκτυο αρκεί η παρακάτω δήλωση στο compose αρχείο του και έτσι μόλις δημιουργηθεί θα είναι ενταγμένος αυτόματα σε αυτό.

```
networks:
  edgex-network:
```

```
    driver: bridge

networks:
  edgex-network: {}
```

Για να το διαπιστώσω και να το αποδείξω υπάρχουν δύο τρόποι. Ο πρώτος είναι μέσω της εντολής **docker network inspect mp_tesi_edgex-network**, που παρέχεται από το docker και δίνει αναλυτικά όλες τις πληροφορίες σχετικά με το δίκτυο που αναγράφεται. Ένα από τα στοιχεία που μας ενδιαφέρει είναι όλοι οι container μαζί με τα χαρακτηριστικά τους οι οποίοι εντάσσονται στο αντίστοιχο δίκτυο. Ο δεύτερος τρόπος είναι προσθέτοντας έναν ακόμα container που αφορά την εφαρμογή portainer η οποία παρέχει την δυνατότητα διαχείρισης container με έναν πιο φιλικό τρόπο απέναντι στον χρήστη. Προσθέτω λοιπόν στο compose της πλατφόρμας και η υπηρεσία του portainer ο οποίος εκτελείται στην πόρτα 9000 και αποδεικνύει ότι όλες οι υπηρεσίες μας τρέχουν στο επιθυμητό δίκτυο. Στην Εικόνα 4.5 παρουσιάζεται ένα στιγμιότυπο από το περιβάλλον που παρέχει η υπηρεσία του portainer με το οποίο αποδεικνύεται πως ο subscriber βρίσκεται πλέον στο ίδιο δίκτυο με όλες τις υπόλοιπες μικρο-υπηρεσίες του EdgeX.

Name	mp_tesi_edgex-network
ID	2136d8d248ef92091846c335544b3cb
Driver	bridge
Scope	local
Attachable	true
Internal	false
IPV4 Subnet - 172.28.0.0/16	IPV4 Gateway - 172.28.0.1
IPV4 IP range -	IPV4 Excluded Ips

Access control

Ownership

Change ownership

Containers in network

Container Name	IPv4 Address
edgex-subscriber-tesi	172.28.0.15/16
edgex-app-rules-engine	172.28.0.5/16
edgex-ui-go	172.28.0.8/16
edgex-core-command	172.28.0.2/16
edgex-sys-mgmt-agent	172.28.0.7/16
edgex-core-data	172.28.0.10/16
edgex-core-metadata	172.28.0.13/16
edgex-kuiper	172.28.0.6/16
edgex-support-notifications	172.28.0.3/16
edgex-support-scheduler	172.28.0.12/16
edgex-redis	172.28.0.4/16
edgex-core-consul	172.28.0.11/16

Εικόνα 4.5: Εμφάνιση του subscriber στο δίκτυο του EdgeX

Αφού λοιπόν έχει γίνει η εγγραφή στην πλατφόρμα ελέγχεται πως είναι επιτυχής. Μια μέθοδος με την οποία αυτό μπορεί να αποδειχθεί είναι παρακολουθώντας αν τα δεδομένα μας φτάνουν ως το service core data, αν διαβάζονται από αυτό και εκτυπώνονται στην πλατφόρμα. Είναι ένας αξιόπιστος τρόπος καθώς αποτελεί μια από τις προϋποθέσεις για την εγγραφή νέας υπηρεσίας στην πλατφόρμα. Τα δεδομένα στέλνονται στην υπηρεσία core-data ως γεγονότα (events) με την μορφή REST APIS αιτημάτων. Αρχικά για τη δημιουργία της δομής των αιτημάτων και τον έλεγχο αποτελεσματικής λειτουργίας τους χρησιμοποίησα το εργαλείο POSTMAN το API του οποίου υπάρχει στην ιστοσελίδα [56]. Συγκεκριμένα τα δύο αιτήματα που χρησιμοποιήθηκαν είναι δύο αιτήματα ανάκτησης GET **http://127.0.0.1:59880/api/v2/event/device/name/publisher**, για να δω πόσα γεγονότα είναι καταγεγραμμένα και αφορούν τον publisher και GET **http://127.0.0.1:598 80/api/v2/ event/all**, για να δω πόσα και ποια είναι τα συνολικά γεγονότα που έχει λάβει το core data. Έτσι λοιπόν για να στέλνω κάθε δεδομένο που διαβάζω από τον subscriber στέλνω ένα POST αίτημα /event/profile-002/publisher/resource-002 με ακατέργαστα δεδομένα (raw-data) στο σώμα το οποίο δηλώνω μέσα στον κώδικα του subscriber.

Προτού βάλουμε σε λειτουργία όλα τα στοιχεία εκτελείται ένα προσαρμοσμένο (custom) script το οποίο είναι γραμμένο σε python. Το script αυτό είναι υπεύθυνο για τον συνεχή έλεγχο για το εάν είναι επιτυχής η επικοινωνία με την υπηρεσία core-data. Στο σώμα του κώδικα υπάρχουν δύο μέθοδοι, οι check_app_on_device και η deploy_on_siemens που αφορούν την μετεγκατάσταση. Δημιουργείται ένας ατέρμων βρόγχος επανάληψης στον οποίο ανά τρία δευτερόλεπτα στέλνεται στην υπηρεσία core-data ένα αίτημα **GET http://localhost:59880/api/v2/ping**. Για όσο το αίτημα αυτό επιστρέφει απάντηση που επιβεβαιώνει πως η επικοινωνία με την μικροϋπηρεσία core-data λειτουργεί ομαλά η ροή του σεναρίου συνεχίζει κανονικά. Τη στιγμή που χάνεται η επικοινωνία με το core-data την συμπεριλαμβάνουμε μέσα σε μια περιοχή εξαίρεσης για να μην σταματήσει το πρόγραμμα.

Έτσι λοιπόν η αρχική μορφή έχει σχηματιστεί, με τον publisher να τρέχει στην τοπική μηχανή, τον συνδρομητή ως container στην εικονική ubuntu μηχανή και τον subscriber ως container στην πλατφόρμα του EdgeX. Αρχικά ξεκινάω τον συνδρομητή, ο οποίος έπειτα από μερικά δευτερόλεπτα είναι ενεργός και περιμένει για νέες συνδέσεις. Επιλέγουμε μια χρονική στιγμή T στην οποία εκτελούμε τον publisher. Όπως είναι διακριτό στην Εικόνα 4.6, αφού έχει πραγματοποιηθεί επιτυχώς η εγγραφή του δημιουργεί το νέο θέμα και ξεκινάει την αποστολή των δεδομένων με την αποστολή των τριών πρώτων εγγραφών που συμπεριλαμβάνονται στο αρχείο CSV.

```
PS C:\Users\it164\Project_workspace\MP_tesi> & C:/Users/it164/AppDat
a/Local/Programs/Python/Python310/python.exe c:/Users/it164/Project_
workspace/MP_tesi/publisher.py

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

CONNECTION TO BROKER ON 10.0.7.132 SUCESSFULL

Sending data is :

2017-10-27 21:44:19,120.1284408569336

Check for the sending data on Subscriber

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

Sending data is :

2017-10-27 21:56:51,118.26673889160156

Check for the sending data on Subscriber

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

Sending data is :

2017-10-27 22:08:16,116.5347671508789

Check for the sending data on Subscriber

----------------------------------------
```

Εικόνα 4.6: Αποστολή 3 δεδομένων από τον publisher

Στη συνέχεια ενώ έχουμε θέσει ήδη την πλατφόρμα σε λειτουργία, δηλαδή όλες οι μικρο-υπηρεσίες που την συντελούν τρέχουν επιτυχώς, εκτελούμε και τον container του subscriber μέσω του docker. Σε αυτό το σημείο αφού και αυτός με τη σειρά του ολοκληρώσει με επιτυχία την εγγραφή του στον συνδρομητή ξεκινάει να διαβάζει μηνύματα όπως φαίνεται στην Εικόνα 4.7. Το στιγμιότυπο προκύπτει από την έξοδο της εφαρμογής του Docker Desktop και αφορά τον συγκεκριμένο container.

Για κάθε μήνυμα εκτυπώνεται η τιμή που αναγνώστηκε. Το μήνυμα αυτό με το που ληφθεί αποστέλλεται αυτόματα με τον τρόπο που περιγράφηκε παραπάνω στο core-data. Το αίτημα που αποστέλλεται βρίσκεται σε μια συνάρτηση η οποία ονομάζεται deploy_on_edgex, έχει ως παράμετρο το μήνυμα και βρίσκεται δηλωμένη στον κώδικα του subscriber. Η συνάρτηση αυτή εκτελεί την αποστολή των δεδομένων στην μικρο-υπηρεσία core-data μόνο στην περίπτωση που ο subscriber εκτελείται στο EdgeX. Συνεπώς, ενεργοποιείται μόνο κάθε φορά που η μεταβλητή περιβάλλοντος INDUSTRIAL_PLATFORM έχει την τιμή EDGEX. Αυτή η μεταβλητή ορίζεται με την τιμή EDGEX εξαρχής ως προεπιλογή, ελέγχεται με την κάθε λήψη μηνύματος και όσο η υπηρεσία τρέχει στην ίδια πλατφόρμα δεν αλλάζει τιμή. Έτσι τα δεδομένα μας φτάνουν στην πλατφόρμα με την μορφή που παρουσιάζεται στις Εικόνες 4.8 και 4.9. Το σώμα της απάντησης στο αίτημα που παρουσιάζεται αφορά αποκλειστικά την συσκευή publisher, μια ιδιότητα που διευκρινίζεται στο τέλος της διεύθυνσης του αιτήματος.

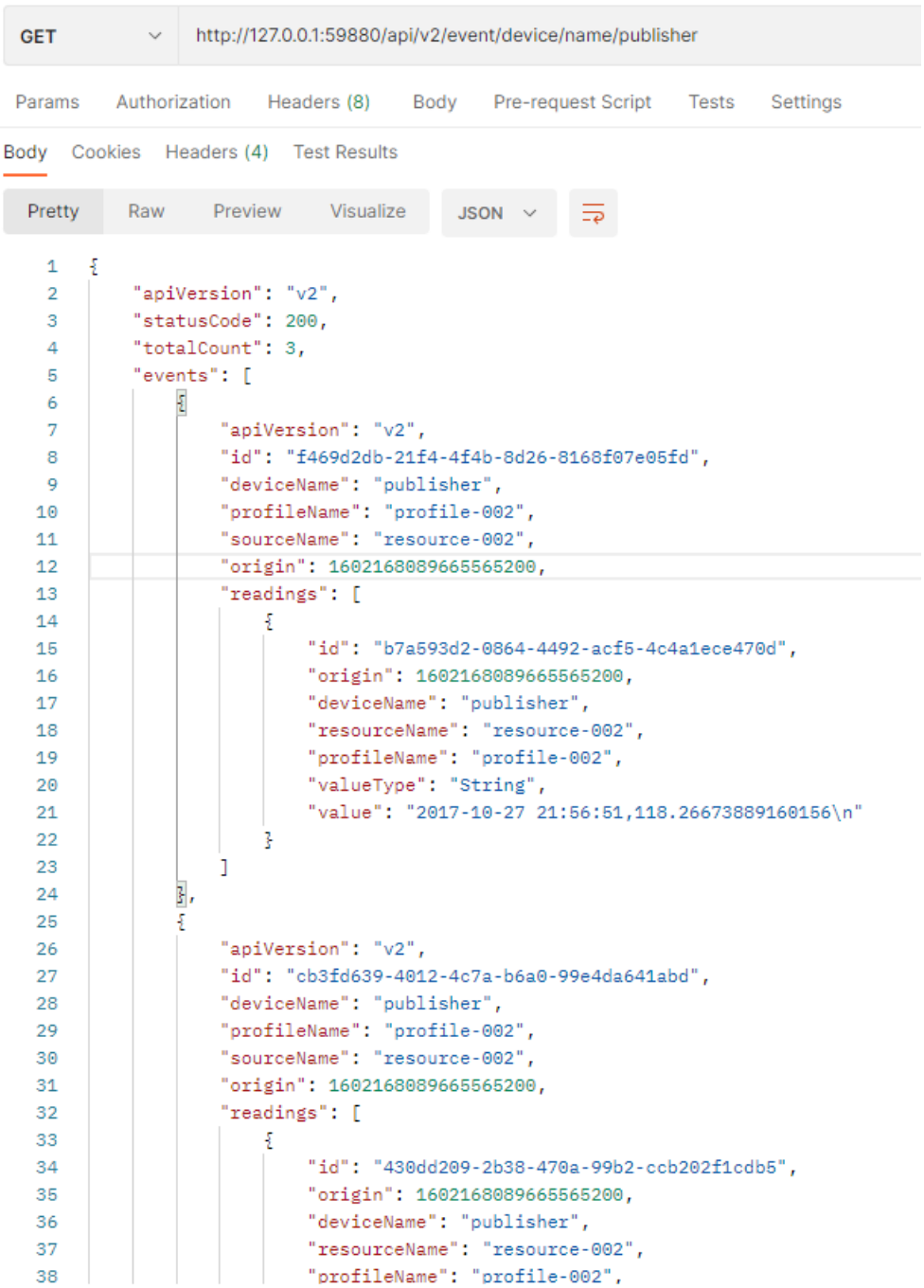
```
connecting to broker host broker_tesi
Subscribing to topic tesi
subscribing begins here
received data is : {2017-10-27 21:44:19,120.1284408569336
}
message topic= tesi
message qos= 0

{'apiVersion': 'v2', 'statusCode': 201, 'id': 'cf8dce82-9978-450a-bd2e-e88371db3548'}
received data is : {2017-10-27 21:56:51,118.26673889160156
}
message topic= tesi
message qos= 0

{'apiVersion': 'v2', 'statusCode': 201, 'id': '2afe00fc-7032-4b38-9078-0267c4988351'}
received data is : {2017-10-27 22:08:16,116.5347671508789
}
message topic= tesi
message qos= 0

{'apiVersion': 'v2', 'statusCode': 201, 'id': 'dc62fec3-8638-4046-a215-3264e331da08'}
received data is : {2017-10-27 22:46:34,118.01374816894531
}
```

Εικόνα 4.7: Ανάγνωση 3 δεδομένων από τον subscriber



Εικόνα 4.8: Παράδειγμα ανάγνωσης δεδομένων από την μικροϋπηρεσία core-data

```
                "valueType": "String",
                "value": "2017-10-27 22:08:16,116.5347671508789\n"
            }
        ]
    },
    {
        "apiVersion": "v2",
        "id": "9cefad98-b642-4f67-95e5-fdcc323a116c",
        "deviceName": "publisher",
        "profileName": "profile-002",
        "sourceName": "resource-002",
        "origin": 1602168089665565200,
        "readings": [
            {
                "id": "d148c4a0-00da-4f25-bb7a-ec5e7372968a",
                "origin": 1602168089665565200,
                "deviceName": "publisher",
                "resourceName": "resource-002",
                "profileName": "profile-002",
                "valueType": "String",
                "value": "2017-10-27 21:44:19,120.1284408569336\n"
            }
        ]
    }
]
}
```


Εικόνα 4.9: Παράδειγμα ανάγνωσης δεδομένων από την μικροϋπηρεσία core-data, μέρος δεύτερο

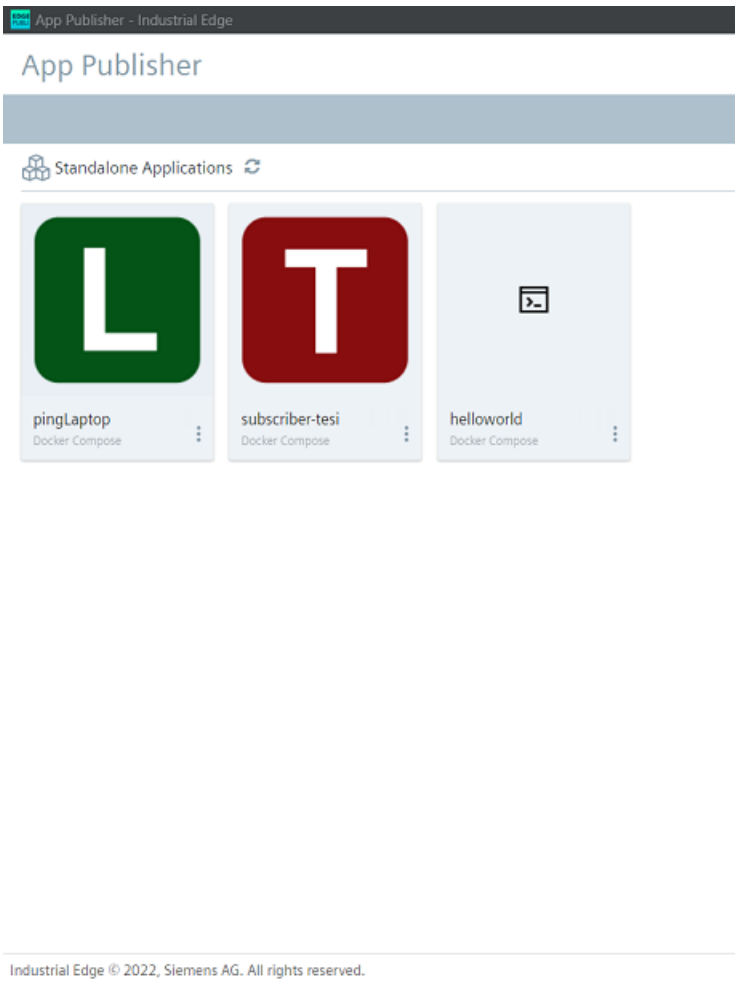
4.4.2 Μετεγκατάσταση στην πλατφόρμα SIEMENS

Ενώ η ροή του σεναρίου κυλάει με τον τρόπο αυτό, επιλέγουμε την χρονική στιγμή Υ στην οποία αποφασίζουμε να προκαλέσουμε μια υποθετική βλάβη στην πλατφόρμα. Η βλάβη αφορά την υπηρεσία core-data η οποία θεωρούμε πως για κάποιο λόγο σταματάει να λειτουργεί και ως συνέπεια χάνεται και η επικοινωνία του subscriber με αυτή. Ο λόγος για τον οποίο προέκυψε το παραπάνω συμβάν μπορεί να οφείλεται είτε σε κάποιο πρόβλημα σύνδεσης δικτύου που έχει ως αποτέλεσμα την απώλεια σύνδεσης της μικρο-υπηρεσίας. Ακόμα μπορεί να οφείλεται και σε κάποια δυσλειτουργία του container στον οποίο τρέχει το core-data κάτι το οποίο οδηγεί στην διακοπή της λειτουργίας του και κατά συνέπεια πάλι στην αδυναμία σύνδεσης με οποιοδήποτε στοιχείο προσπαθεί να αποστείλει δεδομένα. Για να δημιουργήσουμε την κατάσταση αυτή, σταματάμε τον container ο οποίος αφορά την μικρο-υπηρεσία των core-data μέσω του περιβάλλοντος που παρέχει το Docker Desktop. Στο σημείο αυτό, γίνεται αντιληπτή η βλάβη και ξεκινά αυτόματα η διαδικασία της επανεγκατάστασης. Η αναγνώριση γίνεται προτού πραγματοποιηθεί η τελευταία αποστολή στο core-data αφού κάθε φορά για να επιτραπεί η αποστολή γίνεται έλεγχος από το script που κάνει ping και επαληθεύει συνεχώς την ομαλή επικοινωνία με την πλατφόρμα. Στο σημείο αυτό λοιπόν που προκύπτει η βλάβη ενεργοποιείται η εκτέλεση των εντολών που βρίσκονται στο μέρος της εξαίρεσης.

Όταν η εξαίρεση αυτή ενεργοποιηθεί ακολουθούνται τα παρακάτω βήματα στα οποία περιγράφεται πιο ουσιαστικά ο τρόπος με τον οποίο πραγματοποιείται η μετεγκατάσταση. Αρχικά ανοίγει το αρχείο .env στο οποίο συμπεριλαμβάνεται η μεταβλητή περιβάλλοντος **INDUSTRIAL_PLATFORM** και αλλάζει την τιμή της από ”EDGEX” σε ”SIEMENS” και το αρχείο κλείνει. Ακολούθως γίνεται κατασκευή (build) του compose αρχείου του subscriber με την εντολή **docker-compose -f ./subscriber_service/docker-compose.yml build** ώστε να ενημερωθεί και στον container η τιμή της μεταβλητής. Στη συνέχεια κα-

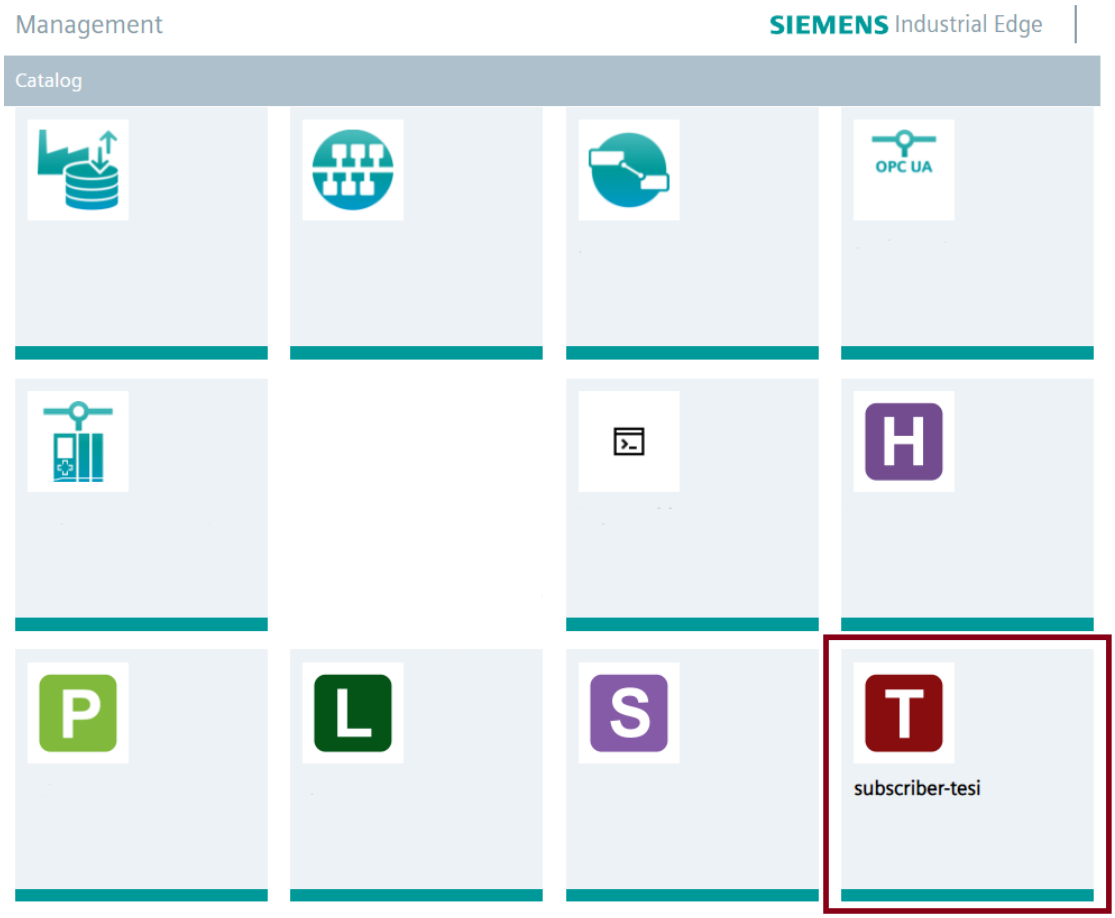
λείται η συνάρτηση που πραγματοποιεί την εγκατάσταση του ενημερωμένου container στην πλατφόρμα της Siemens και μια ακόμα συνάρτηση η οποία είναι υπεύθυνη για έλεγχο ο οποίος αφορά τον χρόνο μετεγκατάστασης και θα αναλυθεί στο κεφάλαιο των πειραμάτων. Η συνάρτηση που αφορά την εγκατάσταση στην πλατφόρμα της SIEMENS αποτελείται από ένα rest αίτημα POST για την είσοδο στο επίπεδό MANAGEMENT της πλατφόρμας. Αν το αίτημα αυτό είναι επιτυχές και συνεπώς η σύνδεση πραγματοποιήθηκε στέλνεται ακόμα ένα αίτημα POST στο middleware που χρησιμοποιήθηκε και το οποίο προωθεί το deploy στο MANAGEMENT. Το αίτημα αυτό περιλαμβάνει την διεύθυνση προορισμού και τις απαραίτητες κεφαλίδες (headers) που είναι το compose αρχείο του subscriber, το όνομα του container, η έκδοση του compose και η πλατφόρμα εγκατάστασης. Αφού ολοκληρωθεί η διαδικασία γίνεται διακοπή λειτουργίας του container στον οποίο τρέχει ο subscriber με την εντολή **docker stop edgex-subscriber-tesi** με στόχο την μείωση κατανάλωσης πόρων της Κεντρικής Μονάδας Επεξεργασίας.

Στην παρακάτω Εικόνα 4.10 παρουσιάζεται ο subscriber στην εγκατεστημένη πλέον μορφή του στο περιβάλλον του Industrial Edge Publisher με το όνομα να ταυτίζεται με αυτό του container. Σε αυτό το στάδιο η εφαρμογή ακόμα βρίσκεται απλά αποθηκευμένη εκτός σύνδεσης (offline) και έτσι είναι απαραίτητο να μεταβεί στο περιβάλλον της πλατφόρμας ώστε να είναι έτοιμη προς εγκατάσταση οποιαδήποτε στιγμή.



Εικόνα 4.10: Επιτυχημένη εγκατάσταση του subscriber στο περιβάλλον του Industrial Edge Publisher

Η διαδικασία της εισαγωγής του subscriber στον Industrial Edge Publisher αποτελεί προϋπόθεση για την εγκατάσταση στην πλατφόρμα SIEMENS σε οποιαδήποτε συσκευή. Το εργαλείο αυτό έχει σχηματιστεί με σκοπό να μετατρέπει τις εικόνες (images) που του παρέχονται από τον χρήστη σε industrial edge εφαρμογές οι οποίες μπορούν να είναι διαχειρίσιμες από τις συσκευές της SIEMENS και μπορούν να εγκατασταθούν και να τρέξουν στο περιβάλλον τους. Επιλέγοντας την εφαρμογή μου δημιουργώ καινούρια έκδοση αυτής 0.0.5 και ανεβάζω το compose.yml αρχείο του subscriber. Η πρώτη διαφορά είναι πως αντί για την εντολή ”build: .” που χρησιμοποίησα στο προηγούμενο compose και καθορίζει τη διαδρομή στην οποία βρίσκεται το Dockerfile τώρα γράφω image: subscriber_service_subscriber-tesi:latest καθώς το Edge Publisher τρέχει τοπικά και συνεπώς έχω στη διάθεση μου όλες τις εικόνες που είχα δημιουργήσει προηγουμένως. Αυτό βοηθάει στη συνέχεια καθώς με την ενημέρωση της εικόνας τοπικά μπορεί να ενημερωθεί η εικόνα που θα γίνει build στην πλατφόρμα ανά πάσα στιγμή. Η δεύτερη διαφορά είναι πως αφαιρούμε όλα τα δίκτυα που σχετίζονται με το Edgex. Αυτό συμβαίνει γιατί θέλουμε ο container που θα τρέχει στην πλατφόρμα της Siemens να είναι τελείως αποσυνδεδεμένος από το περιβάλλον του EdgeX. Έπειτα αφού μεταβώ σε κατάσταση σύνδεσης (online) στην πλατφόρμα, μπορώ να ανεβάσω την έκδοση που δημιούργησα στον κατάλογο (catalog) και έπειτα από εκεί να την εγκαταστήσω σε μία από τις συσκευές που παρέχει η πλατφόρμα.



Εικόνα 4.11: Επιτυχημένη εγκατάσταση του subscriber στον κατάλογο της SIEMENS

Η εγκατάσταση στην λίστα του καταλόγου που βρίσκεται στο περιβάλλον του Industrial Edge Management γίνεται από την εφαρμογή του Industrial Edge publisher. Αφού μεταβώ σε κατάσταση σύνδεσης μπορώ απευθείας να ανεβάσω την εικόνα η οποία πρώτα σχηματίζεται ανάλογα ώστε να έχει την μορφή διαχειρίσιμης εφαρμογής και έπειτα προστίθεται στην λίστα του καταλόγου. Αυτό το βήμα είναι σημαντικό καθώς το επίπεδο Management σε άλλη περίπτωση δεν θα μπορούσε να διαχειριστεί την εγκατάσταση της εφαρμογής του subscriber εφόσον δεν υπήρχε στο περιβάλλον και δεν θα μπορούσε να την αναγνωρίσει ως μορφή.

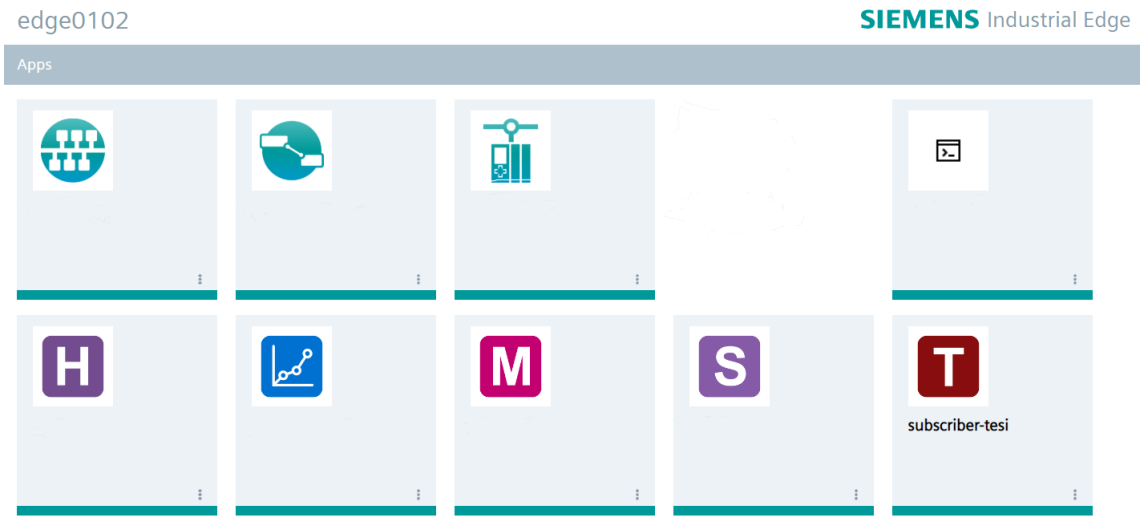
Στην Εικόνα 4.11 παρουσιάζεται πλέον η εφαρμογή μας εγκατεστημένη στον κατάλογο. Στην συγκεκριμένη απεικόνιση επίσης συμπεριλαμβάνονται και άλλες εφαρμογές οι οποίες προϋπήρχαν και δεν σχετίζονται με την εργασία. Για τον λόγο αυτό αλλά και για λόγους ιδιωτικότητας έχει γίνει απόκρυψη των ονομάτων. Εφόσον η εφαρμογή υπάρχει ήδη στο περιβάλλον αυτό την στιγμή που ενεργοποιείται η διαδικασία της μετεγκατάστασης, μπορούμε εκτελώντας μια σειρά εντολών που παρέχονται από το CLI της Siemens να εγκαταστήσουμε την εφαρμογή. Επιλέγουμε να την εγκαταστήσουμε για αρχή στην φυσική edge συσκευή. Η συσκευή αυτή είναι η πιο διαθέσιμη καθώς έχει μεγαλύτερο ελεύθερο αποθηκευτικό χώρο σε σύγκριση με τις υπόλοιπες δύο εικονικές συσκευές και έτσι δεν θα επιβαρύνουμε το σύστημα.

Το script το οποίο καλείται και εκτελείται με την κάθε ενεργοποίηση της μετεγκατάστασης περιλαμβάνει τις παρακάτω εντολές. Πρώτη εκτελείται η **iectl publisher edgemanagement application uploadtocata log -a $NAME -w $VERSION**, όπου στη μεταβλητή NAME έχει δηλωθεί το όνομα της πλέον edge εφαρμογής μας subscriber-tesi και στη μεταβλητή VERSION έχει δηλωθεί η έκδοση της ”0.0.1”. Με την εντολή αυτή πραγματοποιείται το ανέβασμα (upload) στον κατάλογο της πλατφόρμας ο οποίος περιλαμβάνει όλες της εφαρμογές οι οποίες είναι πλέον διαθέσιμες για εγκατάσταση σε κάποια συσκευή. Αμέσως μετά εκτελείται η εντολή **iectl portal batches submit-batch –appid $APP_ID –infoMap ”devices”:$DEVICE_ID –operation installApplication**. Με την εντολή αυτή ξεκινάει η εγκατάσταση της εφαρμογής μας στην επιθυμητή συσκευή στο περιβάλλον της πλατφόρμας. Η διαδικασία αυτή ολοκληρώνεται έπειτα από ένα χρονικό διάστημα το οποίο διερευνάται στο κεφάλαιο των τεστ. Αφού ολοκληρωθεί, το σενάριο συνεχίζει να εκτελείται κανονικά χωρίς καμία επιπλέον απώλεια και χωρίς να απαιτείται κάποιος άλλος μετασχηματισμός. Ο subscriber στη νέα πλατφόρμα πραγματοποιεί εκ νέου εγγραφή ως νέος πελάτης και συνεχίζει να διαβάζει κανονικά τα δεδομένα από τον publisher.

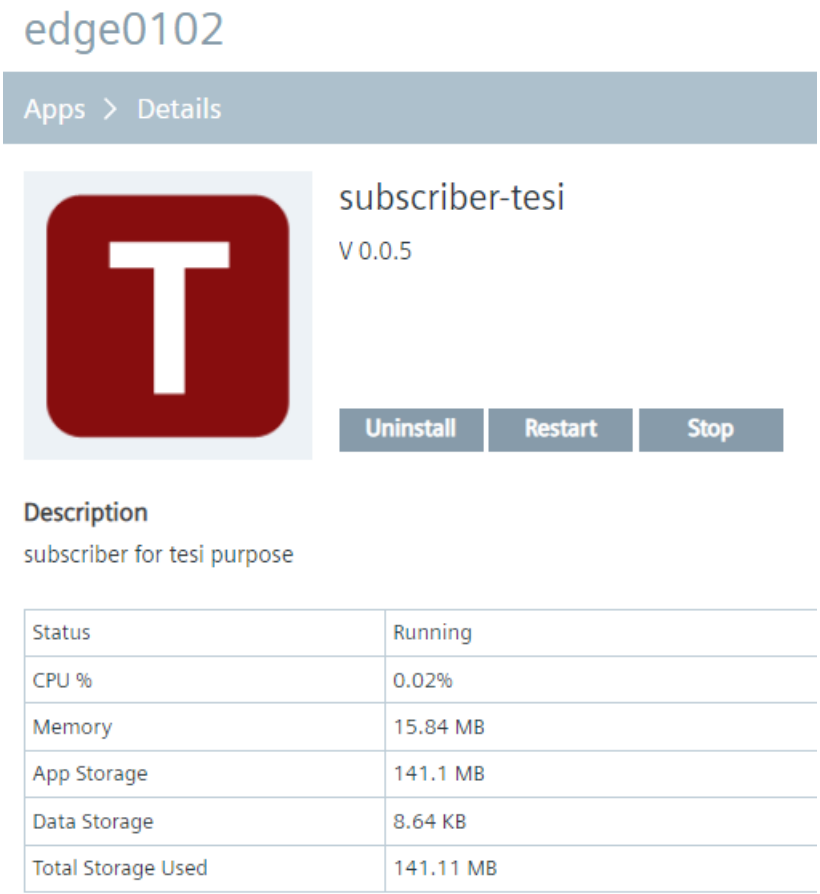
Το μοναδικό μειονέκτημα στην διαδικασία αυτή είναι ο χρόνος που μεσολαβεί μέχρι να ολοκληρωθεί η μετεγκατάσταση. Κατά τη διάρκεια του χρόνου αυτού τα άλλα δύο στοιχεία δεν σταματούν την λειτουργία τους όμως δεν γίνεται ανάγνωση των δεδομένων από τον subscriber μέχρις ότου αυτός ξεκινήσει ξανά την λειτουργία του από την πλατφόρμα της Siemens. Είναι επίσης σημαντικό να σημειωθεί πως από τη στιγμή που βρισκόμαστε σε αυτή την πλατφόρμα έχει σταματήσει η οποιαδήποτε προσπάθεια για αποστολή δεδομένων στο core-data καθώς πλέον η μεταβλητή περιβάλλοντος που έχει οριστεί να ελέγχεται έχει την τιμή SIEMENS και έτσι ο subscriber εκτελεί μόνο ανάγνωση δεδομένων.

Στην Εικόνα 4.12 παρουσιάζεται ένα στιγμιότυπο από την πλατφόρμα της Siemens στο οποίο φαίνεται πως η εφαρμογή έχει εγκατασταθεί με επιτυχία και βρίσκεται στην πλατφόρμα. Για λόγους ιδιωτικότητας έχουν αφαιρεθεί τα ονόματα από τις υπόλοιπες εφαρμογές που είναι εγκατεστημένες στην συσκευή. Από το συγκεκριμένο περιβάλλον μπορούμε να συμπεράνουμε πως πλέον ο subscriber βρίσκεται στην

λίστα των εφαρμογών στην συσκευή που επιλέξαμε. Εκτός από το παραπάνω συμπέρασμα μπορούμε επίσης να παρακολουθήσουμε σε ποια κατάσταση βρίσκεται ο container και να καταγράψουμε σημαντικές πληροφορίες σχετικά με αυτόν. Αφού έχει ενεργοποιηθεί η απομακρυσμένη πρόσβαση στην συσκευή, μπορούμε με την επιλογή info να δούμε όλες τις λεπτομέρειες σχετικά με τον container.



Εικόνα 4.12: Επιτυχημένη εγκατάσταση του subscriber στην φυσική συσκευή της SIEMENS



Εικόνα 4.13: Χαρακτηριστικά του container του subscriber στο περιβάλλον SIEMENS

Όπως φαίνεται και στην Εικόνα 4.13 ο container του subscriber είναι εγκατεστημένος και έχει τεθεί σε λειτουργία αυτόματα από την πλατφόρμα. Μια από τις χρήσιμες πληροφορίες που παρατηρούνται είναι το ποσοστό της Κεντρικής Μονάδας Επεξεργασίας που δεσμεύει ο container για να εκτελέσει τις λειτουργίες του. Η τιμή 0.02% μπορεί να αποτελέσει ένα αρχικό συμπέρασμα για το ότι η καινούρια προσθήκη στην πλατφόρμα δεν επηρεάζει το σύστημα. Αντίθετα ο συνολικός χώρος που καταλαμβάνει ο container και συγκεκριμένα 141.1MB είναι ένας αρκετά επιβαρυντικός όγκος όσον αφορά τους διαθέσιμους πόρους του αποθηκευτικού χώρου. Αυτό οφείλεται στην εικόνα στην οποία βασίστηκε η δημιουργία του container και μπορεί να βελτιωθεί καθώς ο επιπλέον αποθηκευτικός χώρος που χρησιμοποιεί είναι μονάχα 8.64kB που είναι μια φυσιολογική τιμή για έναν container.

Ο τρόπος με τον οποίο μπορεί να μελετήσει κάποιος την λειτουργία του container εφαρμογής που εκτελείται στην πλατφόρμα της SIEMENS είναι μέσω του αρχείου καταγραφής λειτουργιών. Στο σημείο αυτό μας δίνεται η δυνατότητα να κατεβάζουμε οποιαδήποτε χρονική στιγμή το αρχείο αυτό. Στο αρχείο καταγραφών (log file), συμπεριλαμβάνονται όλες οι πληροφορίες σχετικά με την stdout έξοδο όπως για παράδειγμα τα δεδομένα που ελήφθησαν και ακόμα ένα μήνυμα το οποίο επιβεβαιώνει πως ο container τρέχει πλέον στην πλατφόρμα της Siemens.

C: > Users > it164 > e89b2de7c2c3a78db5cf6cb8248cea086163d73733abb6259561427279beab61-json.log


```
1
2
3   {"log":"connecting to broker host 10.0.7.132\n","stream":"stdout",
4   "time":"2023-01-02T22:58:28.485273247Z"}
5   {"log":"Subscribing to topic tesi\n","stream":"stdout",
6   "time":"2023-01-02T22:58:28.489186112Z"}
7   {"log":"subscribing begins here\n","stream":"stdout",
8   "time":"2023-01-02T22:58:28.489210292Z"}
9
10
11  {"log":"received data is : 2017-10-28 05:11:40,100.96003723144531\n",
12  "stream":"stdout","time":"2023-01-02T22:58:29.611438776Z"}
13  {"log":"message topic= tesi\n","stream":"stdout",
14  "time":"2023-01-02T22:58:29.612644626Z"}
15  {"log":"message qos= 0\n","stream":"stdout",
16  "time":"2023-01-02T22:58:29.612776893Z"}
17  {"log":"The app is installed and running on siemens industrial edge platform\n",
18  "stream":"stdout","time":"2023-01-02T22:58:29.61301494Z"}
19
20  {"log":"received data is : 2017-10-28 05:41:10,101.98495483398438\n",
21  "stream":"stdout","time":"2023-01-02T22:58:39.598351848Z"}
22  {"log":"message topic= tesi\n","stream":"stdout",
23  "time":"2023-01-02T22:58:39.598444297Z"}
24  {"log":"message qos= 0\n","stream":"stdout",
25  "time":"2023-01-02T22:58:39.59846675Z"}
26  {"log":"The app is installed and running on siemens industrial edge platform\n",
27  "stream":"stdout","time":"2023-01-02T22:58:39.598761138Z"}
28
```


Εικόνα 4.14: Φύλλο καταγραφής (Log file) του subscriber στην πλατφόρμα SIEMENS μετά την μετεγκατάσταση

Παραπάνω στην Εικόνα 4.14 παρουσιάζεται ένα μέρος από το αρχείο καταγραφής για τον subscriber αφού έχει ξεκινήσει τη λειτουργία του στην πλατφόρμα της Siemens. Στο στιγμιότυπο περιλαμβάνεται καταγεγραμμένη η διαδικασία της νέας έγγραφης στον συνδρομητή αλλά και δύο αναγνώσεις δεδομένων. Με κάθε ανάγνωση εκτυπώνεται και το μήνυμα που έχω ορίσει να εμφανίζεται όταν ο container λειτουργεί στην πλατφόρμα της Siemens **"The app is installed and running on Siemens industrial edge"** .Αποδεικνύεται λοιπόν πως η διαδικασία συνεχίζεται κανονικά χωρίς κανένα πρόβλημα.

Όλη η παραπάνω διαδικασία πραγματοποιήθηκε με σκοπό να αποδειχθεί πως σε περίπτωση που παρουσιάζεται ένα πρόβλημα σε ένα μέρος του σύστημά μας δεν είναι απαραίτητο πως πρέπει να επηρεαστεί η λειτουργία και των υπόλοιπων στοιχείων του. Παρ' όλο που οι δύο πλατφόρμες είναι τελείως διαφορετικές σε αρκετά βασικά χαρακτηριστικά όπως είναι για παράδειγμα η αρχιτεκτονική, μπορούν να χρησιμοποιηθούν ως εναλλακτική λύση σε περίπτωση που προκύπτει ένα πρόβλημα σαν αυτό. Συνεπώς αποτελεί μια πολύ χρήσιμη λύση για τα συστήματα στο περιβάλλον του βιομηχανικού διαδικτύου των πραγμάτων και μπορεί να αποβεί και καταλυτική για την αποφυγή απωλειών με την προσθήκη βελτιώσεων.

4.5 Μετεγκατάσταση λόγω υπερφόρτωσης της Κεντρικής Μονάδας Επεξεργασίας

Σε αυτό το κομμάτι παρουσιάζεται το δεύτερο σενάριο, που αφορά την υπερφόρτωση της Κεντρικής Μονάδας Επεξεργασίας του συστήματος εξ' αιτίας του container στον οποίο εκτελείται ο subscriber. Μια τέτοια συνθήκη μπορεί να αποτελέσει σοβαρό λόγο για την μετεγκατάσταση μιας μικροϋπηρεσίας σε άλλη πλατφόρμα καθώς με τον τρόπο αυτό απελευθερώνονται οι πόροι, η πλατφόρμα δεν κινδυνεύει από πιθανή κατάρρευση ή διακοπή και η υπηρεσία εγκαθίσταται και συνεχίζει να εκτελείται κανονικά σε μια πλατφόρμα με μεγαλύτερη ανεκτικότητα και περισσότερους πόρους Κεντρικής Μονάδας Επεξεργασίας. Καθώς η διαδικασία την οποία εκτελώ που αφορά αποκλειστικά και μόνο τον τρόπο με τον οποίο πραγματοποιείται η μετεγκατάσταση από την μια πλατφόρμα στην άλλη είναι ακριβώς ο ίδιος με προηγουμένως και αναλύθηκε στην υποενότητα 8.3 για την αποφυγή επανάληψης περιγραφής της ίδιας διαδικασίας σε αυτή την υποενότητα, οι επαναλαμβανόμενες διαδικασίες απλώς να αναφέρονται και θα αναλύονται μόνο οι επιπλέον λειτουργίες που προστέθηκαν και οι μεταβολές που εφαρμόστηκαν ώστε να εκτελεστεί και αυτό το σενάριο.

Αρχικά δημιουργείται ένα επιπλέον πρόγραμμα με την ονομασία check_cpu το οποίο ακολουθεί τον σκελετό του προγράμματος ping_core_data. Στην περίπτωση αυτή, αλλάζει τόσο ο έλεγχος που πραγματοποιείται όσο και το στοιχείο το οποίο ελέγχεται. Ενώ στην προηγούμενη περίπτωση μας ενδιέφερε έχοντας ως στόχο την μικροϋπηρεσία core-data να ελέγχουμε τη διαθεσιμότητα της, αυτή την φορά ο στόχος είναι η μικροϋπηρεσία του subscriber. Πιο συγκεκριμένα ο έλεγχος αφορά τον container edgex_subscriber _tesi στον οποίο εκτελείται ο subscriber και ελέγχει το ποσοστό της Κεντρικής Μονάδας Επεξεργασίας που χρησιμοποιεί από τους διαθέσιμους πόρους του εξυπηρετητή (host). Το docker δίνει την δυνατότητα για την παρακολούθηση όλων των παραμέτρων ενός container. Συγκεκριμένα μέσω της εντολής docker stats αναγράφεται ένα σύνολο από πληροφορίες που αφορούν την κατανάλωση πόρων της Κεντρικής Μονάδας Επεξεργασίας του εξυπηρετητή που κάνει ο container σε ποσοστό επί τοις εκατό, την χρήση μνήμης του εξυπηρετητή ου κάνει ο container σε ποσοστό επί τοις εκατό, το όριο που έχει οριστεί για τη μνήμη που μπορεί να χρησιμοποιήσει ο συγκεκριμένος container, το σύνολο των δεδομένων που αφορούν διαδικασίες I/O που σχετίζονται με τον container και τέλος τον αριθμό των

διαδικασιών που δημιουργούνται και εκτελούνται από τον container.

Η πλήρης σύνταξη της εντολής που χρησιμοποιήθηκε για την συγκεκριμένη περίπτωση παρουσιάζεται παρακάτω και η έξοδος που παράγεται αντιπροσωπεύει την τιμή του ποσοστού επί τοις εκατό της Κεντρικής Μονάδας Επεξεργασίας που καταναλώνεται, τη συγκεκριμένη χρονική στιγμή καθώς δηλώνεται η παράμετρος –no-stream:

docker stats –no-stream edgex-subscriber-tesi –format {{.CPUPerc}}

Ο αλγόριθμος που χρησιμοποιείται εκτελείται μέσα σε έναν ατέρμων βρόγχο επανάληψης. Για όσο ο βρόγχος είναι ενεργός(while True) ελέγχεται η έξοδος της εντολής που αφορά το ποσοστό κατανάλωσης πόρων της Κεντρικής Μονάδας Επεξεργασίας. Η τιμή που επιστρέφεται έπειτα από την εκτέλεση της εντολής αποθηκεύεται σε μια μεταβλητή που ορίζουμε με όνομα cmd. Στη συνέχεια χρησιμοποιώντας την βιβλιοθήκη **subprocess** που παρέχεται από την γλώσσα προγραμματισμού python και συγκεκριμένα την μέθοδο check_output αποθηκεύουμε την έξοδο ως αποτέλεσμα το οποίο μετατρέπουμε σε μια μεταβλητή με περιεχόμενο μια συμβολοσειρά και κρατάμε τα 4 πρώτα στοιχεία, εξαιρώντας δηλαδή το σύμβολο του ποσοστού. Έπειτα πραγματοποιείται έλεγχος για το εάν η τιμή που παράγεται είναι μικρότερη από 30.0 και στην περίπτωση που η συνθήκη αυτή είναι αληθής εκτυπώνεται αντίστοιχο μήνυμα στο οποίο αναγράφεται η τιμή μέτρησης και ότι το ποσοστό που χρησιμοποιείται είναι κάτω του 30%. Σε κάθε επανάληψη υπάρχει ενδιάμεση αναμονή 5 δευτερολέπτων. Αν σε κάποιον έλεγχο η τιμή είναι μεγαλύτερη του 30% τότε ξεκινάει η διαδικασία της μετεγκατάστασης και εμφανίζεται στην οθόνη ανάλογο μήνυμα. Στην Εικόνα 4.15 παρουσιάζεται ένα στιγμιότυπο που περιλαμβάνει την έξοδο που αφορά και την περίπτωση στην οποία ο έλεγχος είναι αληθής και την περίπτωση όπου διαπιστώνεται ότι το όριο έχει ξεπεραστεί και ενεργοποιείται η μετεγκατάσταση στην πλατφόρμα της SIEMENS.

Η υπερφόρτωση την χρονική στιγμή Τ που παρατηρείται στην Εικόνα 4.15 επιτυγχάνεται χάρης την χρήση του εργαλείου γραμμής εντολών pumba [57] μέσω της εικόνας gaiaadm/pumba που είναι διαθέσιμο μέσω του docker-hub [58]. Το εργαλείο αυτό χρησιμοποιείται για stress test με στόχο έναν συγκεκριμένο container. Προσφέρει πολλές δυνατότητες μέσω εντολών μεταξύ των οποίων η διακοπή ή η ολική αφαίρεση container και προσομοίωση Δικτύου Ευρείας Περιοχής (WAN). Η εντολή που χρησιμοποιείται για την πυροδότηση του γεγονότος κατά το οποίο η Κεντρική Μονάδα Επεξεργασίας υπερφορτώνεται είναι η stress ορίζοντας συγκεκριμένη διάρκεια στα 10 δευτερόλεπτα κατά τα οποία ο container θα δέχεται τον επιπλέον φόρτο. Η πλήρης σύνταξη της εντολής παρουσιάζεται παρακάτω και στην ουσία αφού συνδεθεί με το docker deamon της τοπικής μηχανής, από το τερματικό της οποίας εκτελείται, προκαλεί μια κατάσταση χάους εκτελώντας τυχαίες λειτουργίες.Έτσι από την στιγμή που ξεκινάει να εκτελείται με το που η τιμή της κατανάλωσης πόρων της Κεντρικής Μονάδας Επεξεργασίας ξεπεράσει αυτήν που ορίζουμε (30%) ικανοποιείται η συνθήκη που έχουμε ορίσει και ξεκινάει η διαδικασία της μετεγκατάστασης.

docker run -v /var/run/docker.sock:/var/run/docker.sock gaiaadm/pumba stress edgex-subscriber-tesi -d 10s

```
PS C:\Users\it164\Project_workspace\MP_tesi> python .\check_cpu.py
CPU usage under 30% .. exact value measured :  0.04%

CPU usage under 30% .. exact value measured :  0.07%

CPU usage under 30% .. exact value measured :  0.04%

CPU usage under 30% .. exact value measured :  0.08%

CPU usage OVER 30% ! ! .. exact value measured :  394.71%
  STARTING MIGRATION PROCESS . .
```

Εικόνα 4.15: Ενεργοποίηση μετεγκατάστασης εξ' αιτίας υπερφόρτωσης της Κεντρικής Μονάδας Επεξεργασίας

Κεφάλαιο 5ο: Μετρήσεις και πειραματικά αποτελέσματα

Έχοντας ως στόχο την πιο ενδελεχή διερεύνηση του σεναρίου πραγματοποιήθηκε ένα σύνολο από μετρήσεις οι οποίες παρουσιάζονται σε αυτό το κεφάλαιο. Συνολικά παρουσιάζονται δύο είδη μετρήσεων που αφορούν τον συνολικό χρόνο της μετεγκατάστασης από την μια πλατφόρμα στην άλλη και τον συνολικό χρόνο ανάκαμψης μιας μικροϋπηρεσίας στην ίδια πλατφόρμα (downtime). Για κάθε διαδικασία που περιγράφεται, παρουσιάζονται και σχολιάζονται όλα τα αποτελέσματα που προέκυψαν έπειτα από τις μετρήσεις. Το περιβάλλον στο οποίο πραγματοποιήθηκαν οι μετρήσεις της πρώτης κατηγορίας συντελείται από τρία βασικά στοιχεία. Το πρώτο στοιχείο είναι το Industrial Management επίπεδο που παρέχει η πλατφόρμα της Siemens καθώς κάθε εφαρμογή προτού εγκατασταθεί σε μια από τις συσκευές περνάει πρώτα από το περιβάλλον αυτό. Τα υπόλοιπα δύο στοιχεία είναι οι δύο τύποι συσκευών που παρέχει η πλατφόρμα τις Siemens και στις οποίες μπορεί να εγκατασταθεί η εφαρμογή. Καθώς θέλουμε να δοκιμάσουμε την εγκατάσταση και στην εικονική αλλά και στη φυσική συσκευή είναι σημαντικό να αναφερθούν τα χαρακτηριστικά τους. Παρακάτω στην Εικόνα 5.1 παρατίθεται ο πίνακας στον οποίο συνοψίζονται όλα τα στοιχεία του περιβάλλοντος δοκιμών μαζί με τα χαρακτηριστικά τους.

	IEM	IED(Φυσική Συσκευή)	IED(Virtual Συσκευή)
Λειτουργικό Σύστημα	IEM O.S. 1.5.6	IED O.S. 1.5.0-21	IED O.S. 1.5.0-5e
Μνήμη RAM (GB)	12	8	16
CPU (πυρήνες)	4	4	6
Σκληρός δίσκος (GB)	200	200	120

Εικόνα 5.1: Χαρακτηριστικά στοιχείων πειράματος

5.1 Μετρήσεις χρόνου μετεγκατάστασης στην φυσική συσκευή

Μια από τις παραμέτρους που είναι σημαντικό να διερευνηθεί είναι ο χρόνος που διαρκεί η μετεγκατάσταση μέχρι και την ολοκλήρωση της εγκατάστασης στην νέα πλατφόρμα. Ο χρόνος αυτός ταυτίζεται με το χρονικό διάστημα το οποίο χρειάζεται μια εφαρμογή για να εγκατασταθεί σε μια συσκευή της Siemens όταν έχει εκτελεστεί σε προηγούμενο χρόνο η διαδικασία για την εγκατάσταση στον κατάλογο. Αν διερευνήσουμε τον χρόνο αυτό με διαφορετικές εκτελέσεις του σεναρίου, μπορούμε να έχουμε μια ξεκάθαρη εικόνα τόσο απέναντι στην ανταπόκριση από την πλατφόρμα όσο και για το ποιες πιθανές βελτιώσεις μπορούμε να εφαρμόσουμε. Γίνεται επίσης αντιληπτό για πόσο χρόνο παραμένει σε αδράνεια το σύστημα μέχρι να μπορέσει να επαναλειτουργήσει ομαλά. Το χρονικό διάστημα το οποίο θα μετρηθεί ξεκινάει από τη χρονική στιγμή που γίνεται αντιληπτή η βλάβη στην πλατφόρμα του Edgex και πιο συγκεκριμένα με την πρώτη αποτυχημένη προσπάθεια αποστολής ping από την εφαρμογή ping_core_data. Η μέτρηση ολοκληρώνεται τη στιγμή που η εφαρμογή του subscriber εντοπιστεί στη λίστα των εφαρμογών της συσκευής στην οποία εγκαταστάθηκε.

Η μέτρηση της χρονικής αυτής περιόδου γίνεται με τη βοήθεια της βιβλιοθήκης time της python. Ακριβώς

πριν ξεκινήσει η εγκατάσταση της εφαρμογής στη Siemens δηλώνεται η μεταβλητή time_migration _start η οποία παίρνει την τιμή time.time() που επιστρέφει σε δευτερόλεπτα την τρέχουσα χρονική στιγμή σε float μορφή. Αμέσως μετά ακολουθεί η συνάρτηση deploy_on_Siemens. Μόλις αυτή ολοκληρωθεί καλούμε την συνάρτηση check_app_on_device η οποία χρησιμοποιώντας το API που παρέχεται από την Siemens και αφορά την άμεση επικοινωνία με τις συσκευές χωρίς την μεσολάβηση του Management επιπέδου αρχικά πραγματοποιεί σύνδεση στην συσκευή που περιμένουμε να εγκατασταθεί η εφαρμογή.

Στην περίπτωση που η είσοδος πραγματοποιηθεί με επιτυχία στέλνεται ακόμα ένα αίτημα το οποίο επιστρέφει ώς απάντηση την λίστα των εφαρμογών που λειτουργούν στο περιβάλλον της συσκευής. Έπειτα ξεκινάει να εκτελείται ένας βρόγχος επανάληψης έως ότου η εφαρμογή του subscriber βρεθεί στη λίστα με τις εφαρμογές. Για να επιτευχθεί αυτό σε κάθε επανάληψη αναζητείται μέσα στη λίστα των εφαρμογών αυτή με το όνομα subscriber-tesi. Για όσο δεν εντοπίζεται ο subscriber γίνεται ανανέωση της λίστας έως ότου ολοκληρωθεί η εγκατάσταση και τελικά το όνομα του subscriber βρεθεί στη λίστα. Με την εύρεση του subscriber γίνεται έξοδος από την μέθοδο και δηλώνεται μια ακόμα μεταβλητή η time_migration_end με τιμή time.time(). Έτσι ο υπολογισμός του συνολικού χρόνου προκύπτει από την αφαίρεση του αρχικού χρόνου από τον τελικό και αποθηκεύεται σε μια νέα μεταβλητή η οποία εκτυπώνεται στο τέλος της κάθε εκτέλεσης του προγράμματος.

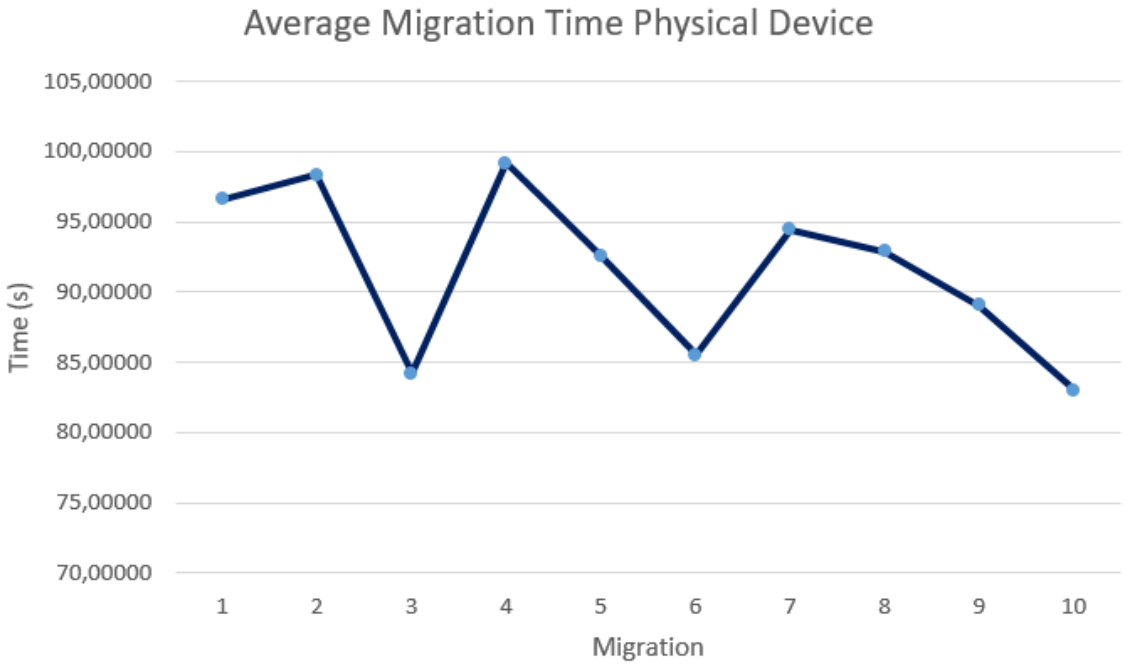
MIGRATION ΣΤΗΝ ΦΥΣΙΚΗ ΣΥΣΚΕΥΗ ΤΗΣ SIEMENS

RUNS	TIME(s)	TIME(s)	TIME(s)	AVERAGE
1	99,74820948	95,11759138	94,86914945	96,57832
2	94,81838298	107,6090531	92,63987422	98,35577
3	76,68870735	91,89852309	83,98064709	84,18929
4	106,92852	88,77983546	101,7660072	99,15812
5	76,84747076	100,9162629	99,97867513	92,58080
6	90,96127343	92,66176915	72,95966935	85,52757
7	103,5672114	82,96903849	96,80053687	94,44560
8	82,90988207	95,60154271	100,1448748	92,88543
9	88,00549841	95,14571333	83,97197461	89,04106
10	80,31075025	78,7302053	89,93342543	82,99146

Εικόνα 5.2: Μετρήσεις χρόνου μετεγκατάστασης στην φυσική συσκευή

Στην παραπάνω Εικόνα 5.2 παρουσιάζεται ο πίνακας ο οποίος διαμορφώθηκε με βάση τα αποτελέσματα που προέκυψαν έπειτα από ένα σύνολο μετρήσεων που πραγματοποιήθηκαν κατά τη διάρκεια του σεναρίου. Συγκεκριμένα κάθε στήλη συμπεριλαμβάνει δέκα επαναλήψεις εκτέλεσης της διαδικασίας της μετεγκατάστασης στην φυσική edge συσκευή της SIEMENS. Για κάθε επανάληψη γίνεται μέτρηση της χρονικής περιόδου και σημειώνεται η τελική τιμή. Οι παραπάνω εκτελέσεις πραγματοποιήθηκαν με ένα πρόσθετο χρονικό διάστημα που μεσολαβεί ανάμεσα σε αυτές με σκοπό να αποφευχθεί η υπερφόρτωση της πλατφόρμας το οποίο δεν συμπεριλαμβάνεται σε καμία μέτρηση. Η διαδικασία αυτή πραγματοποιήθηκε τρεις φορές για να προκύψει μεγαλύτερη εγκυρότητα στα αποτελέσματα. Από κάθε γραμμή των τριών διαφορετικών εκτελέσεων προκύπτει μια μέση τιμή του χρονικού διαστήματος για μια μετεγκατάσταση σε κάθε περίπτωση.

Στην Εικόνα 5.3 παρακάτω παρουσιάζεται η απεικόνιση των αποτελεσμάτων σε διάγραμμα που σχεδιάστηκε στο Excel. Ο άξονας X αποτελείται από τον αριθμό της εκτέλεσης την οποία πραγματοποιούμε, για παράδειγμα ο αριθμός 2 αναφέρεται στην τιμή που μετρήθηκε κατά την δεύτερη εκτέλεση μετεγκατάστασης. Στον άξονα Y παρατίθενται όλες οι τιμές των αποτελεσμάτων που συλλέχθηκαν από τις μέσες τιμές του χρόνου της μετεγκατάστασης που προέκυψαν παραπάνω και είναι σε μορφή δευτερολέπτων.



Εικόνα 5.3: Διάγραμμα μέσων τιμών μετρήσεων μετεγκατάστασης στην φυσική συσκευή της SIEMENS

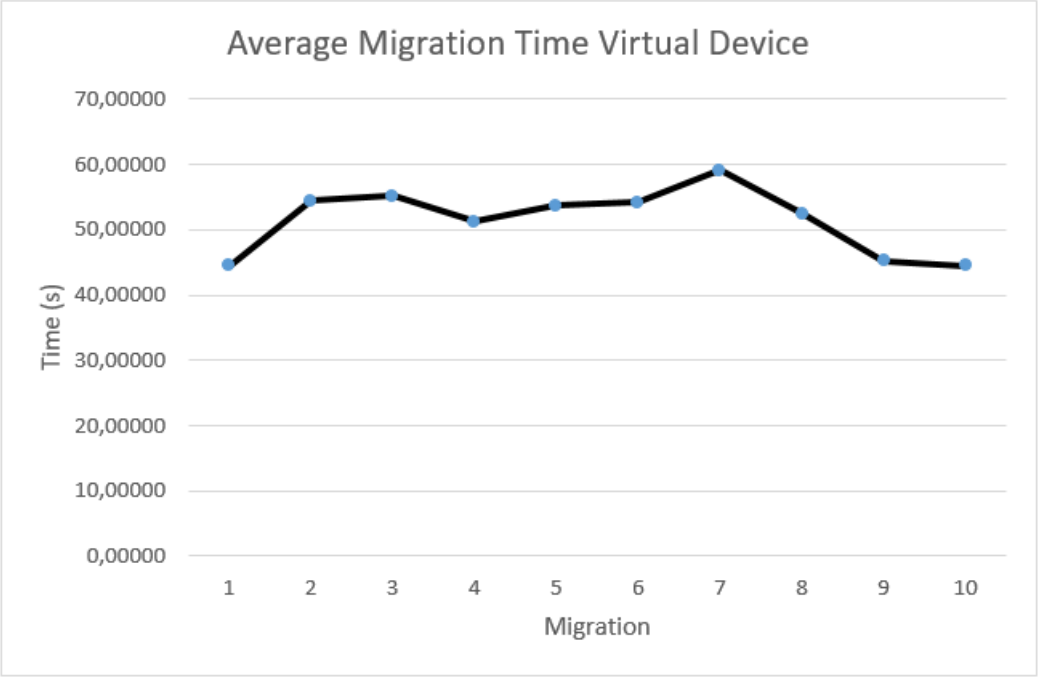
Τόσο από το διάγραμμα όσο και από τις μετρήσεις προκύπτουν συμπεράσματα σχετικά με την επιτυχία της διαδικασίας. Αρχικά όλες οι εκτελέσεις πραγματοποιήθηκαν χωρίς να υπάρξει κάποια η οποία ήταν ανεπιτυχής είτε δεν ολοκληρώθηκε μέχρι τέλος. Επιπλέον ο χρόνος ο οποίος χρειάζεται για να ολοκληρωθεί η διαδικασία δεν προκαλεί καθυστέρηση στην ολοκλήρωση της μετεγκατάστασης καθώς η μέγιστη διάρκεια είναι κάτω των 2 λεπτών. Επίσης δεν παρατηρείται καμία συστηματική αυξομείωσή στην χρονική τιμή ούτε κάποια συγκεκριμένη συμπεριφορά που ακολουθεί κάποιο μοτίβο, όπως για παράδειγμα μείωση στο μισό μετά από κάθε εκτέλεση. Τέλος οι διαφορές μεταξύ των εκτελέσεων δεν παρουσιάζουν σημαντική απόκλιση ώστε να μην μπορούμε να διακρίνουμε μια αρκετά αντιπροσωπευτική τιμή στο αποτέλεσμα και να μπορέσει να υπάρχει μια πρόβλεψη για το πόσο χρόνο διαρκεί η διαδικασία.

5.2 Μετρήσεις χρόνου μετεγκατάστασης στην εικονική συσκευή

MIGRATION ΣΤΗΝ VIRTUAL ΣΥΣΚΕΥΗ ΤΗΣ SIEMENS				
RUNS	**TIME(s)**	**TIME(s)**	**TIME(s)**	**AVERAGE**
1	42,13389659	44,56677723	46,9141221	44,53827
2	57,79045224	64,10824227	41,56438136	54,48769
3	45,32325172	64,89310384	55,39255166	55,20297
4	46,09286141	61,51545048	46,07640862	51,22824
5	54,7853086	52,35023689	54,04454112	53,72670
6	67,00131989	48,95294905	46,59078884	54,18169
7	48,98506069	68,5658114	59,54674411	59,03254
8	54,54666829	61,21615124	41,62887073	52,46390
9	43,32654524	48,77917838	43,45272589	45,18615
10	41,98961735	40,71421552	50,82842708	44,51075

Εικόνα 5.4: Μετρήσεις χρόνου μετεγκατάστασης στην εικονική συσκευή

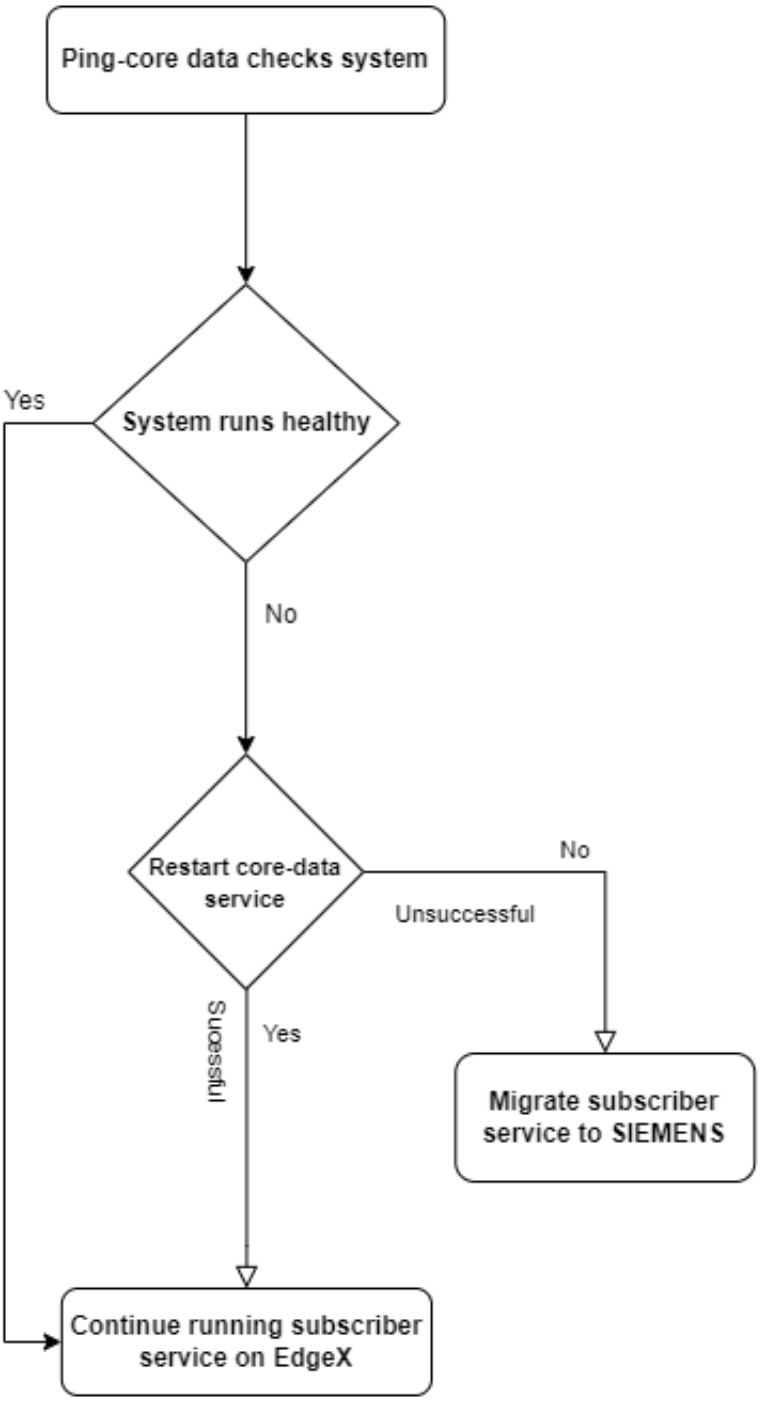
Η ίδια ακριβώς διαδικασία πραγματοποιήθηκε και στην εικονική συσκευή που παρέχεται από την SIEMENS. Είναι σημαντικό να σημειωθεί πως δεν άλλαξε κανένα άλλο στοιχείο εκτός από την IP διεύθυνση συσκευής στην οποία προορίζεται να γίνει η μετεγκατάσταση. Στόχος είναι να προκύψει μια πρώτη σύγκριση σχετικά με τα δύο είδη συσκευών της πλατφόρμας. Πιο συγκεκριμένα μπορεί να διερευνηθεί αν ο χρόνος μπορεί να αποτελέσει μια μεταβλητή που μπορεί να ληφθεί υπ' όψιν όσον αφορά την επιλογή συσκευής. Η Εικόνα 5.4 αποτελεί μια απεικόνιση όλων των μετρήσεων που πραγματοποιήθηκαν και αφορούν την εικονική συσκευή της Siemens. Όπως είναι εμφανές το χρονικό διάστημα είναι αισθητά μειωμένο σε σχέση με την προηγούμενη περίπτωση. Συγκεκριμένα ο χρόνος που χρειάζεται για τη μετεγκατάσταση σε αυτή τη συσκευή είναι σχεδόν ο μισός συγκριτικά με την φυσική συσκευή. Η μέγιστη τιμή που παρατηρείται δεν ξεπερνάει το ένα λεπτό και η μέση τιμή μιας μετεγκατάστασης υπολογίζεται από 44 έως 55 δευτερόλεπτα. Και σε αυτή την περίπτωση δεν υπάρχει καμία απώλεια, δηλαδή όλες οι μετεγκαταστάσεις ολοκληρώνονται μέχρι τέλος με επιτυχία. Συνεπώς συγκρίνοντας τους χρόνους που απαιτούνται και αφορούν την κάθε συσκευή καταλήγουμε στο συμπέρασμα πως η μετεγκατάσταση στην εικονική μηχανή είναι πιο αποτελεσματική αν λάβουμε υπ' όψιν μας στην περίπτωση που προτεραιότητα αποτελεί η μετεγκατάσταση θέλουμε να πραγματοποιηθεί όσο πιο σύντομα γίνεται.



Εικόνα 5.5: Διάγραμμα μέσων τιμών μετρήσεων μετεγκατάστασης στην εικονική συσκευή της SIEMENS

Το διάγραμμα που παρουσιάζεται στην Εικόνα 5.5, σχηματίζεται σε κάθε περίπτωση από τη μέση τιμή των μετρήσεων παρουσιάζει μια ακόμα πιο ομαλή συμπεριφορά σε σχέση με αυτήν που παρατηρείται από την φυσική συσκευή. Πιο συγκεκριμένα δεν υπάρχουν διακυμάνσεις μεγαλύτερες των 20 δευτερολέπτων μεταξύ των διαφορετικών τιμών. Επιπλέον ο τρόπος σχηματισμού του είναι ακριβώς ο ίδιος με το προηγούμενο διάγραμμα και οι άξονες αντικατοπτρίζουν τις ίδιες μεταβλητές.

5.3 Μετρήσεις χρόνου επαναφοράς (downtime) στην πλατφόρμα EdgeX Foundry



Εικόνα 5.6: Διάγραμμα ροής: Απόφαση για μετεγκατάσταση του subscriber

Αφού διερευνήθηκαν οι δύο παραπάνω περιπτώσεις που αφορούν την μετεγκατάσταση στα δύο είδη συσκευών, παρουσιάζει ενδιαφέρον η διερεύνηση του αν αυτή η διαδικασία τελικά είναι απαραίτητη πάντα σε περίπτωση βλάβης ή μπορεί να παραληφθεί σε κάποιες περιπτώσεις. Για παράδειγμα, στην περίπτωση όπου η διακοπή σύνδεσης με την υπηρεσία core-data οφείλεται σε μια απλή διακοπή λειτουργίας του container, το πρόβλημα δυσλειτουργίας της πλατφόρμας και κατ' επέκταση της λειτουργίας του subscriber είναι δυνατόν να λυθεί με την επανεκκίνησή του container στον οποίο εκτελείται η υπηρεσία core-data αποφεύγοντας την διαδικασία της μετεγκατάστασης.

Όπως παρουσιάζεται στο διάγραμμα ροής της Εικόνας 5.6, η πρώτη κατάσταση που αποτελεί και την εκκίνηση είναι ο συνεχής έλεγχος που πραγματοποιείται από την υπηρεσία ping_core_data εκτελώντας ping στην υπηρεσία code-data. Σε κάθε ping ελέγχεται αν το σύστημα λειτουργεί ομαλά. Αν το ping είναι επιτυχές και συνεπώς το σύστημα δεν παρουσιάζει πρόβλημα δικτύου που αφορά την υπηρεσία core-data, ο subscriber συνεχίζει να εκτελείται κανονικά στην πλατφόρμα του EdgeX. Αν όμως το ping είναι ανεπιτυχές ακολουθεί η επόμενη διαδικασία ελέγχου που αφορά την δοκιμή για επανεκκίνηση της υπηρεσίας core-data. Στην περίπτωση αυτή γίνεται επανεκκίνηση του container και αν η επανεκκίνηση είναι επιτυχής και διαπιστωθεί πως η λειτουργία του συστήματος συνεχίζει κανονικά ο subscriber συνεχίζει να εκτελείται στην πλατφόρμα του EdgeX. Στην άλλη περίπτωση όπου η επανεκκίνηση είναι ανεπιτυχής και άρα συνεχίζει να υπάρχει πρόβλημα στην πλατφόρμα, πραγματοποιείται κανονικά η διαδικασία μετεγκατάστασης στην πλατφόρμα της SIEMENS. Η αναφορά της πιθανότητας αυτής έχει μεγάλη αξία καθώς όπως αποδείχθηκε προηγουμένως η διαδικασία της μετεγκατάστασης διαρκεί περίπου ένα λεπτό. Έτσι είναι χρήσιμο να διερευνήσουμε και να συγκρίνουμε αυτόν τον χρόνο με την διάρκεια της ανάκαμψης του συστήματος στην περίπτωση επανεκκίνησης.

Μια παράμετρος λοιπόν που αξίζει να συγκριθεί με τα προηγούμενα πειράματα και συνδέεται με την περίπτωση που περιγράφηκε είναι αυτή του χρόνου επαναφοράς μιας υπηρεσίας στην πλατφόρμα του Edgex. Συγκεκριμένα, αφορά το χρονικό διάστημα το οποίο ξεκινάει από την διακοπή μιας υπηρεσίας έως τη στιγμή που αυτή θα επανέλθει και θα επαναλειτουργεί κανονικά. Για την περίπτωση του σεναρίου η χρονική στιγμή εκκίνησης είναι το σημείο όπου γίνεται αντιληπτό ότι ο container στον οποίο εκτελείται η υπηρεσία core-data έχει σταματήσει να λειτουργεί και η λήξη είναι η στιγμή όπου η υπηρεσία θα επανέλθει και θα εμφανίζεται ξανά στην λίστα με τους container που εκτελούνται.

Οι δοκιμές που εκτελέστηκαν με σκοπό την μέτρηση του χρονικού διαστήματος ανάκτησης μικροϋπηρεσίας στην πλατφόρμα EdgeX πραγματοποιήθηκαν μέσω ενός προγράμματος το οποίο είναι γραμμένο στην γλώσσα προγραμματισμού python και το σχεδίασα συγκεκριμένα για αυτή την λειτουργία. Αρχικά δηλώνονται δύο μεταβλητές, όπου η πρώτη αφορά την διεύθυνση της μικροϋπηρεσίας core-data με την ονομασία CORE_PING με την οποία θα αναφέρεται στην συνέχεια του κειμένου. Στην δεύτερη μεταβλητή δηλώνεται ως osCheck και ως τιμή της αποθηκεύεται σε μορφή συμβολοσειράς η εντολή του docker η οποία εμφανίζει από την λίστα των container που εκτελούνται τον container με όνομα edgex-core-data.

Έπειτα ξεκινάει ένας ατέρμων βρόγχος επανάληψης με την μορφή while(True). Στο σώμα του βρόγχου στέλνονται συνεχώς ping REST αιτήματα στην μικροϋπηρεσία core-data. Για όσο διάστημα τα αιτήματα αυτά είναι επιτυχή η διαδικασία επαναλαμβάνεται και τα αιτήματα συνεχίζουν να αποστέλλονται κανονικά. Επιλέγουμε μια τυχαία χρονική στιγμή στην οποία σταματάμε τον container edgex-core-data χειροκίνητα από το περιβάλλον του Docker Desktop. Στο σημείο αυτό από την στιγμή που ο container

σταματήσει να εκτελείται το ping αποτυγχάνει και έτσι εκτελείται η εξαίρεση όπου πρώτο βήμα αποτελεί η αποθήκευση της τρέχουσας χρονικής στιγμής στην μεταβλητή downtime_start_point ως σημείο εκκίνησης της διαδικασίας. Την ίδια στιγμή εκτελείται η εντολή του docker docker-compose start data, η οποία ενεργοποιεί την επανεκκίνηση της υπηρεσίας data που αφορά τον container edgex-core-data. Στη συνέχεια επανειλημμένα για όσο το αποτέλεσμα που επιστρέφει η εντολή της osCheck είναι κενό, δηλαδή ο container edgex-core-data δεν βρίσκεται στη λίστα με τους container που εκτελούνται μεσολαβεί ένα διάστημα αναμονής στο οποίο ελέγχεται συνεχόμενα το αποτέλεσμα της εντολής αυτής. Μόλις η εντολή επιστρέψει αποτέλεσμα ο βρόγχος επανάληψης διακόπτεται και η χρονική στιγμή αυτή αποθηκεύεται ως στιγμή λήξης καθώς ο container πλέον βρίσκεται πάλι στη λίστα και εκτελείται κανονικά. Τέλος ο συνολικός χρόνος που προκύπτει από την αφαίρεση της στιγμής εκκίνησης από τη στιγμή λήξης είναι ο χρόνος που χρειάστηκε για να ολοκληρωθεί η διαδικασία της επαναφοράς.

Ο πίνακας που παρουσιάζεται στην Εικόνα 5.7 περιλαμβάνει όλα τα αποτελέσματα που προέκυψαν έπειτα από τις μετρήσεις. Συγκεκριμένα εκτελέστηκαν συνολικά 50 δοκιμές, οι οποίες χωρίστηκαν σε 5 στήλες όπου η κάθε μία περιλαμβάνει δέκα μετρήσεις που πραγματοποιήθηκαν. Η τελευταία γραμμή της κάθε στήλης αποτελεί τον μέσο όρο των μετρήσεων της αντίστοιχης στήλης. Επιπλέον δεν υπάρχει κάποιο συγκεκριμένο χρονικό διάστημα που μεσολαβεί ανάμεσα σε κάθε εκτέλεση καθώς η κάθε μια εκτελείται με τυχαίο σημείο διακοπής λειτουργίας του container.

DOWNTIME 1	DOWNTIME 2	DOWNTIME 3	DOWNTIME 4	DOWNTIME 5
3,4669830799	3,6129045486	4,5193095207	3,5652487278	3,4918797016
3,4578104019	3,3913805485	3,7824447155	3,4960949421	3,5586957932
3,3542945385	3,4304761887	3,5906310081	3,5382435322	3,5093903542
3,4418714046	3,6540434361	3,4654901028	3,5584199429	3,5870282650
3,3234014511	3,5372481346	3,5303847790	3,6117227077	4,2681841850
3,3094382286	3,4157645702	3,4493088722	3,4811663628	3,4420926571
3,3974359035	3,4763126373	3,9614830017	3,6304037571	3,4686124325
4,2339692116	3,3503944874	3,4448232651	3,4819116592	3,6108145714
3,0865845680	3,6302230358	3,3762915134	3,1794700623	3,6135497093
3,6302254200	3,3273394108	3,6030833721	4,1611003876	3,4476253986
3,4702014208	***3,4826086998***	***3,6723250151***	***3,5703782082***	***3,5997873068***

Εικόνα 5.7: Μετρήσεις χρόνου επαναφοράς στην πλατφόρμα EdgeX

Η πιο σημαντική παρατήρηση που μπορεί να γίνει έπειτα από τις παραπάνω μετρήσεις είναι ο χρόνος που απαιτείται για την ανάκτηση λειτουργίας της υπηρεσίας που είχε υποστεί βλάβη. Το διάστημα που μεσολαβεί μέχρις ότου μια υπηρεσία να επανέλθει σε λειτουργία έπειτα από βλάβη είναι συνεχώς μικρότερο των 5 δευτερολέπτων. Συγκριτικά με την διαδικασία της μετεγκατάστασης χρονικά είναι αποτελεσματικότερη διαδικασία καθώς μέχρι ο subscriber να ξεκινήσει να στέλνει επιτυχώς νέα δεδομένα στην πλατφόρμα του EdgeX δεν υπάρχει μεγάλη απώλεια εισερχόμενων δεδομένων από τον publisher. Πάραυτα η συνθήκη αυτή μπορεί να ισχύει μόνο στην περίπτωση που μπορέσει να γίνει επιτυχής επανεκκίνηση της μικροϋπηρεσίας core-data. Σε κάθε άλλη περίπτωση η διαδικασία της μετεγκατάστασης είναι αποτελεσματικότερη τόσο χρονικά όσο και από την άποψη της απώλειας δεδομένων συγκριτικά με την ολοκληρωτική διακοπή του συστήματος.

5.4 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκαν όλες οι πειραματικές μετρήσεις που σχετίζονται με το περιβάλλον των υλοποιήσεων που πραγματοποιήθηκαν, σε συνδυασμό με απεικόνιση διαγραμμάτων μέσω των οποίων μπορούν να προκύψουν ουσιαστικά συμπεράσματα.

Κεφάλαιο 6ο: Συμπεράσματα και μελλοντικές βελτιώσεις

Μετά από την θεωρητική ανάλυση όλων των τεχνολογιών που χρησιμοποιήθηκαν αλλά και την υλοποίηση του πρακτικού μέρους, παρακάτω ακολουθούν χρήσιμα συμπεράσματα και παρατηρήσεις που έχουν προκύψει. Στο τέλος του κεφαλαίου παρατίθενται προτάσεις που αφορούν σε πιθανές προσθήκες ή μεταβολές της παρούσας εργασίας με σκοπό τη βελτίωση της. Τέλος, σχολιάζονται και προτείνονται επιπλέον λειτουργίες οι οποίες θα μπορούσαν να υλοποιηθούν ώστε η μεθοδολογία και τα αποτελέσματά της εργασίας να ενσωματωθούν πλήρως σε ένα πραγματικό σενάριο.

6.1 Ανάλυση συμπερασμάτων

Ο τρόπος με τον οποίο οι υπηρεσίες είναι τοποθετημένες σε ένα σύστημα είναι ιδιαίτερα σημαντικός. Στην προσομοίωση που παρουσιάστηκε η κύρια εφαρμογή του subscriber επιλέχθηκε να εκτελείται σε πλατφόρμες που βρίσκονται στο edge. Η επικοινωνία ήταν άμεση αλλά και ακριβής καθώς αναγνώστηκαν όλα τα δεδομένα τα οποία είχαν αποσταλεί χωρίς να υπάρχει καμία απώλεια. Ακόμα οι ελάχιστοι αλλά υπαρκτοί πόροι που καταναλώνει η εφαρμογή δεν επιβαρύνουν σε κανέναν βαθμό το υπόλοιπο σύστημα καθώς δεν είναι όλες οι υπηρεσίες τοποθετημένες στο ίδιο περιβάλλον. Στην περίπτωση που η υπηρεσία δεν είχε εκτελεστεί στο επίπεδο του edge η αναγνώριση του σφάλματος στο περιβάλλον της θα καθυστερούσε, θα είχαμε μεγάλη απώλεια δεδομένων στο διάστημα που μεσολαβεί από την βλάβη μέχρι και την αντιμετώπισή της αλλά και θα χρειαζόταν να τοποθετηθεί σε ένα διαφορετικό επίπεδο καταναλώνοντας επιπλέον πόρους.

Γίνονται επίσης αντιληπτά όλα τα πλεονεκτήματα που προσφέρει η οργάνωση ενός συστήματος με βάση την αρχιτεκτονική των μικρουπηρεσιών. Είναι σημαντικό να σημειωθεί πως η διαδικασία της μετεγκατάστασης πραγματοποιήθηκε χωρίς να επηρεαστεί κανένα άλλο στοιχείο του συστήματος. Πιο συγκεκριμένα, δεν μεταβλήθηκε ούτε η δομή αλλά ούτε και το πλήθος των στοιχείων που ήδη υπήρχαν συνολικά. Επιπλέον, το σφάλμα που δημιουργήθηκε στην πλατφόρμα και αφορούσε μόνο το περιβάλλον της υπηρεσίας του subscriber έγινε αντιληπτό στιγμιαία και μπόρεσε να αντιμετωπιστεί τοπικά. Στην περίπτωση που όλο το σενάριο ήταν δομημένο σε μία εφαρμογή, όλες οι υπηρεσίες θα έπρεπε να εκτελούνται στην ίδια πλατφόρμα και συνεπώς η βλάβη θα επηρέαζε όλο το σύστημα το οποίο θα παρέλυε χωρίς να γίνεται αντιληπτό το σημείο από το οποίο προήλθε. Ακόμα και στην περίπτωση όπου θα γινόταν μετεγκατάσταση ολόκληρου του συστήματος ο απαιτούμενος χρόνος θα ήταν πολύ μεγαλύτερος συγκριτικά με αυτόν που χρειάστηκε ώστε να μετακινηθεί μονάχα μια υπηρεσία.

Αποδείχθηκε επίσης πόσο σημαντικό είναι να υπάρχει η δυνατότητα της ευελιξίας και πόσα οφέλη μπορεί να προσδώσει στη βελτίωση της λειτουργίας ενός συστήματος. Ο στόχος αυτός επιτευχθεί καθώς η μετεγκατάσταση πραγματοποιήθηκε επιτυχώς σε δύο διαφορετικές πλατφόρμες παρ' όλο που παρουσιάζουν διαφορετικά χαρακτηριστικά. Εκτός από τις διαφορετικές πλατφόρμες η ευελιξία επιλογής συσκευής τοποθέτησης κρίνεται απαραίτητη ακόμα και στο περιβάλλον της ίδιας της πλατφόρμας. Όπως προέκυψε έπειτα από τις μετρήσεις που πραγματοποιήθηκαν στο Κεφάλαιο 5, ο χρόνος που χρειάζεται μια μετεγκατάσταση της υπηρεσίας στην εικονική συσκευή είναι πολύ λιγότερος σε σύγκριση με την φυσική συσκευή. Χάριν στην ευελιξία που προσδίδει ο τρόπος με τον οποίο αναπτύχθηκε η εφαρμογή, παρέχεται η δυνατότητα επιλογής της συσκευής ανάλογα με το ποια αποτελεί την πιο γρήγορη λύση.

Ένα ακόμα κομμάτι που έχει ενδιαφέρον να σχολιαστεί είναι αυτό που προκύπτει έπειτα από την σύγκριση του απαιτουμένου χρόνου μετεγκατάστασης και επαναλειτουργίας μιας υπηρεσίας. Με τις μετρήσεις που πραγματοποιήθηκαν στο Κεφάλαιο 5 διαφαίνεται ότι ο μέσος χρόνος που χρειάστηκε για να ολοκληρωθεί η μετεγκατάσταση της εφαρμογής του subscriber ήταν πολύ περισσότερος συγκριτικά με το μέσο χρόνο που χρειάστηκε για να μπορέσει η υπηρεσία που παρουσίασε βλάβη να επαναλειτουργήσει χωρίς να χρειαστεί να πραγματοποιηθεί η μετεγκατάσταση. Πράγματι σε σενάρια όπου η βλάβη που προκλήθηκε αφορά για παράδειγμα μια απώλεια σύνδεσης που οφείλεται σε κάποια προσωρινή δυσλειτουργία του συστήματος, είναι πολύ πιο αποτελεσματικό χρονικά να γίνει προσπάθεια για την επανεκτέλεση της υπηρεσίας που παρουσίασε την αδυναμία επικοινωνίας. Αντιθέτως, στην περίπτωση στην οποία το πρόβλημα που προκύπτει αφορά μεγαλύτερο μέρος της πλατφόρμας, ή στην περίπτωση που δεν υπάρχει βεβαιότητα για το ποια είναι η πηγή του προβλήματος, η λύση της μετεγκατάστασης είναι πολύ πιο αποτελεσματική καθώς εξοικονομείται όλος ο χρόνος που χρειάζεται για την διερεύνηση και επίλυση του προβλήματος.

6.2 Παρατηρήσεις και προτάσεις για μελλοντική βελτίωση

Κάθε υλοποίηση και έρευνα μπορεί να εξελιχθεί με νέες προσθήκες ή και μεταβολές. Ακόμα και η συγκεκριμένη υλοποίηση πέρασε από διάφορα στάδια έως ότου ολοκληρωθεί, με κάθε βήμα να είναι πιο ολοκληρωμένο και βελτιωμένο από το προηγούμενο. Μια πρώτη μεταβολή που θα παρουσίαζε ιδιαίτερο ενδιαφέρον θα ήταν να χρησιμοποιηθούν για το ίδιο σενάριο διαφορετικά πρωτόκολλα που χρησιμοποιούνται στο βιομηχανικό διαδίκτυο των πραγμάτων. Με τον τρόπο αυτόν θα μπορούσε να γίνει μια σύγκριση όσον αφορά την αποτελεσματικότητα του κάθε πρωτοκόλλου αλλά και το τι διαφορές θα μπορούσαν να προκύψουν κατά την υλοποίηση τους, καθώς το καθ' ένα παρουσιάζει διαφορετικά χαρακτηριστικά. Θα είχε επίσης ενδιαφέρον να διερευνηθεί πόσο και με ποιόν τρόπο θα μπορούσε να επηρεάσει η επιλογή πρωτοκόλλου στην διαδικασία της μετεγκατάστασης αλλά και στην επιλογή του τρόπου επικοινωνίας με τις υπόλοιπες υπηρεσίες του συστήματος.

Ακόμα μια προσθήκη που παρουσιάζει ενδιαφέρον είναι η σύγκριση μεγαλύτερου αριθμού πλατφορμών. Μπορεί οι πλατφόρμες που επιλέχθηκαν να είναι από τις πιο συχνά χρησιμοποιούμενες στις εφαρμογές του βιομηχανικού διαδικτύου όμως δεν είναι οι μοναδικές. Μια πιο ολοκληρωμένη σύγκριση θα μπορούσε να προκύψει στην περίπτωση που η μετεγκατάσταση πραγματοποιείται σε διάφορες πλατφόρμες συγκρίνοντας τον χρόνο που απαιτείται αλλά και μελετώντας αν η διαδικασία αυτή είναι εφικτή σε όλες. Σε αυτό το σημείο θα μπορούσε επίσης να μελετηθεί ένα σενάριο στο οποίο το αρχικό περιβάλλον λειτουργίας της υπηρεσίας δεν είναι το edge. Έπειτα από την μετεγκατάσταση στο edge θα παρουσίαζε ιδιαίτερο ενδιαφέρον να συγκριθεί η λειτουργία, αποτελεσματικότητα και συμπεριφορά της υπηρεσίας στα δύο αυτά περιβάλλοντα.

Τέλος, μια ακόμα προσθήκη θα μπορούσε να είναι αυτή της δοκιμής περισσότερων σεναρίων βλαβών. Στην περίπτωση αυτή μπορεί να γίνει διακριτό σε ποιες περιπτώσεις αποτελεί πλεονέκτημα η μετεγκατάσταση μιας υπηρεσίας και σε ποιες είναι προτιμότερο να εφαρμοστεί μια διαφορετική λύση. Επιπλέον, θα ήταν ιδιαίτερα σημαντικό να διερευνηθούν τα θέματα ασφάλειας που αφορούν τόσο την διαδικασία μετακίνησης της υπηρεσίας όσο και την επικοινωνία που πραγματοποιείται μεταξύ των στοιχείων του υπόλοιπου συστήματος.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] I. G. Itisha Sharma and D. D. Kiran, "Industry 5.0 And Smart Cities: A Futuristic Approach," *European Journal of Molecular and Clinical Medicine*, vol. 07, no. 08, pp. 2750–2756, 2020.

[2] H.-G. K. Heiner Lasi, P. Fettke, T. Feld, and M. Hoffmann, "Industry 4.0," *Business and Information Systems Engineering*, vol. 06, p. 239–242, 2014.

[3] Image was retreived from : https://www.researchgate.net/figure/From-Industry-10-to-40_fig1_346901544.

[4] L. S. Alejandro GermánFrank and N. FabiánAyalac, "Industry 4.0 technologies: Implementation patterns in manufacturing companies," *International Journal of Production Economics*, vol. 210, pp. 15–26, 2019.

[5] P. A. Saurabh Vaidya and S. Bhosle, "Industry 4.0 – A Glimpse," *IProcedia Manufacturing*, vol. 20, pp. 233–238, 2018.

[6] G. Aceto, V. Persico, and A. Pescapé, "A survey on information and communication technologies for industry 4.0: State-of-the-art, taxonomies, perspectives, and challenges," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 4, pp. 3467–3501, 2019.

[7] N. H. N. A. Mohd Aiman Kamarul Bahrin, Mohd Fauzi Othman and M. F. Talib, "Industry 4.0: A review on industrial automation and robotic," *Jurnal Teknologi*, vol. 78, pp. 6–13, 2016.

[8] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl, "Industry 4.0 – an introduction in the phenomenon," *IFAC-PapersOnLine*, vol. 49, no. 25, pp. 8–12, 2016. 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016.

[9] M. Ghobakhloo, "Industry 4.0, digitization, and opportunities for sustainability," *Journal of Cleaner Production*, vol. 252, p. 119869, 2020.

[10] P. Marcon, F. Zezulka, I. Vesely, Z. Szabo, Z. Roubal, O. Sajdl, E. Gescheidtova, and P. Dohnal, "Communication technology for industry 4.0," in *2017 Progress In Electromagnetics Research Symposium - Spring (PIERS)*, pp. 1694–1697, 2017.

[11] M. A. Jabraeil Jamali, B. Bahrami, A. Heidari, P. Allahverdizadeh, and F. Norouzi, *IoT Architecture*, pp. 9–31. Cham: Springer International Publishing, 2020.

[12] T. Yousuf, R. Mahmoud, F. Aloul, and I. Zualkernan, "Internet of things (iot) security: Current status, challenges and countermeasures," *International Journal for Information Security Research*, vol. 5, pp. 608–616, 12 2015.

[13] M. Burhan, R. A. Rehman, B.-S. Kim, and B. Khan, "Iot elements, layered architectures and security issues: A comprehensive survey," *Sensors*, vol. 18, pp. 27–96, 08 2018.

[14] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: Architecture, advances and challenges," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[15] D. G. Pivoto, L. F. de Almeida, R. da Rosa Righi, J. J. Rodrigues, A. B. Lugli, and A. M. Alberti, "Cyber-physical systems architectures for industrial internet of things applications in industry 4.0: A literature review," *Journal of Manufacturing Systems*, vol. 58, pp. 176–192, 2021.

[16] A. Sari, A. Lekidis, and I. Butun, *Industrial Networks and IIoT: Now and Future Trends*, pp. 3–55. Cham: Springer International Publishing, 2020.

[17] S. Figueroa-Lorenzo, J. Añorga, and S. Arrizabalaga, "A survey of iiot protocols: A measure of vulnerability risk analysis based on cvss," *ACM Comput. Surv.*, vol. 53, apr 2020.

[18] Q. Zhao, *Presents the Technology, Protocols, and New Innovations in Industrial Internet of Things (IIoT)*, pp. 39–56. Cham: Springer International Publishing, 2020. Editors : "Kanagachidambaresan, G. R. and Anand, R. and Balasubramanian, E. and Mahima, V.".

[19] G. Hillar, *MQTT Essentials - A Lightweight IoT Protocol*. Packt Publishing, 2017.

[20] A. Thantharate, C. Beard, and P. Kankariya, "Coap and mqtt based models to deliver software and security updates to iot devices over the air," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1065–1070, 2019.

[21] M. Collina, G. E. Corazza, and A. Vanelli-Coralli, "Introducing the qest broker: Scaling the iot by bridging mqtt and rest," in *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, pp. 36–41, 2012.

[22] S. Lee, H. Kim, D.-k. Hong, and H. Ju, "Correlation analysis of mqtt loss and delay according to qos level," in *The International Conference on Information Networking 2013 (ICOIN)*, pp. 714–717, 2013.

[23] J. Toldinas, B. Lozinskis, E. Baranauskas, and A. Dobrovolskis, "Mqtt quality of service versus energy consumption," in *2019 23rd International Conference Electronics*, pp. 1–4, 2019.

[24] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration," *ACM Comput. Surv.*, vol. 51, jan 2019.

[25] X. Huang, A. Zhou, P. Jia, L. Liu, and L. Liu, "Fuzzing the android applications with http/https network data," *IEEE Access*, vol. 7, pp. 59951–59962, 2019.

[26] M. S. David Gourley, Brian Totty, A. Aggarwal, and S. Reddy, *HTTP: The Definitive Guide*. O'Reilly Media, Inc., 2002.

[27] S. Jaloudi, "Communication protocols of an industrial internet of things environment: A comparative study," *Future Internet*, vol. 11, no. 3, 2019.

[28] I. N. Fovino, A. Carcano, M. Masera, and A. Trombetta, "Design and implementation of a secure modbus protocol," in *Critical Infrastructure Protection III* (C. Palmer and S. Shenoi, eds.), (Berlin, Heidelberg), pp. 83–96, Springer Berlin Heidelberg, 2009.

[29] P. Mathur, *Overview of IoT and IIoT*, pp. 19–43. Berkeley, CA: Apress, 2020.

[30] G. Thomas, "Introduction to the modbus protocol," *The Extension*, vol. 9, no. 4, pp. 1–4, 2008.

[31] Y. Fang, X. Han, and B. Han, "Research and implementation of collision detection based on modbus protocol.," *Journal of Engineering Science & Technology Review*, vol. 6, no. 1, 2013.

[32] D. Pliatsios, P. Sarigiannidis, T. Liatifis, K. Rompolos, and I. Siniosoglou, "A novel and interactive industrial control system honeypot for critical smart grid infrastructure," in *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–6, 2019.

[33] S. Tamboli, M. Rawale, R. Thoraiet, and S. Agashe, "Implementation of modbus rtu and modbus tcp communication using siemens s7-1200 plc for batch process," in *2015 international conference on smart technologies and management for computing, communication, controls, energy and materials (ICSTM)*, pp. 258–263, IEEE, 2015.

[34] S. Cavalieri and F. Chiacchio, "Analysis of opc ua performances," *Computer Standards & Interfaces*, vol. 36, no. 1, pp. 165–177, 2013.

[35] Image was retreived from : https://reference.opcfoundation.org/Core/Part1/v104/docs/6.2.

[36] C. von Arnim, S. Friedl, A. Lechler, and A. Verl, "Automated opc ua address space generation from existing data structures," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, pp. 959–964, 2019.

[37] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[38] I. Sittón-Candanedo, R. S. Alonso, S. Rodríguez-González, J. A. García Coria, and F. De La Prieta, "Edge computing architectures in industry 4.0: A general survey and comparison," in *14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019)* (F. Martínez Álvarez, A. Troncoso Lora, J. A. Sáez Muñoz, H. Quintián, and E. Corchado, eds.), (Cham), pp. 121–131, Springer International Publishing, 2020.

[39] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020.

[40] "Real-life use cases for edge computing." `https://innovationatwork.ieee.org/real-life-edge-computing-use-cases/`, 2023. Accessed: 2023-01-18.

[41] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.

[42] Image was retreived from : https://henriquesd.medium.com/monolithic-microservices-architecture-239e8799d3e1.

[43] M. Kalske, N. Mäkitalo, and T. Mikkonen, "Challenges when moving from monolith to microservice architecture," in *Current Trends in Web Engineering* (I. Garrigós and M. Wimmer, eds.), (Cham), pp. 32–47, Springer International Publishing, 2018.

[44] M. M. Irakli Nadareishvili, Ronnie Mitra and M. Amundsen, *Microservice Architecture: Aligning Principles, Practices, and Culture*. O'Reilly Media, Inc., 2016.

[45] Image was retreived from : https://www.datarobot.com/blog/introduction-to-microservices/.

[46] EdgeX Foundry The Open Source Edge Platform : https://www.edgexfoundry.org/.

[47] J. Liang, F. Liu, S. Li, and Z. Cai, "A comparative research on open source edge computing systems," in *Artificial Intelligence and Security* (X. Sun, Z. Pan, and E. Bertino, eds.), (Cham), pp. 157–170, Springer International Publishing, 2019.

[48] Image was retreived from the official EdgeX Foundry Documentation page: https://docs.edgexfoundry.org/2.1/.

[49] `https://www.siemens.com/global/en/products/automation/topic-areas/industrial-edge.html`.

[50] "Industrial edge documentation." `https://docs.eu1.edge.siemens.cloud`, 2023. Accessed: April 26, 2023.

[51] T. Kämäräinen, Y. Shan, M. Siekkinen, and A. Ylä-Jääski, "Virtual machines vs. containers in cloud gaming systems," in *2015 International Workshop on Network and Systems Support for Games (NetGames)*, pp. 1–6, 2015.

[52] J. Turnbull, *The Docker Book: Containerization Is the New Virtualization*. James Turnbull, 2014.

[53] Docker Official Docs : https://docs.docker.com/.

[54] D. Westerveld, *API Testing and Development with Postman: A practical guide to creating, testing, and managing APIs for automated software testing*. Packt Publishing Ltd, 2021.

[55] mosquitto.conf man page : https://mosquitto.org/man/mosquitto-conf-5.html.

[56] EdgeXFoundry core-data API : https://reference.opcfoundation.org/Core/Part1/v104/docs/6.2.

[57] Pumba: chaos testing tool for Docker : https://github.com/alexei-led/pumba.

[58] Pumba gaiaadm image : https://hub.docker.com/r/gaiaadm/pumba/.