



Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης  
Πολυτεχνική Σχολή  
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών  
Τομέας Ηλεκτρονικής & Υπολογιστών

---

Υλοποίηση εργαλείου πλήρους στοίβας σε περιβάλλον  
Kubernetes για την αυτοματοποίηση εφαρμογής φίλτρων σε  
μηνύματα με χρήση της τεχνολογίας διαμεσολάβησης  
μηνυμάτων

---

Διπλωματική Εργασία  
του  
Εμμανουήλ Ζήση-Μήλη  
[zemmanox@ece.auth.gr](mailto:zemmanox@ece.auth.gr)  
ΑΕΜ: 8053

**Επιβλέπων:** Συμεωνίδης Ανδρέας  
Αναπ. Καθηγητής Α.Π.Θ.  
**Συνεπιβλέποντες:** Εμμανουήλ Τσαρδούλιας  
Μεταδιδακτορικός Ερευνητής Α.Π.Θ.  
Κωνσταντίνος Παναγιώτου  
Υποψήφιος Διδάκτορας Α.Π.Θ.

26 Οκτωβρίου 2021

## Περίληψη

Η μετάβαση των τεχνολογιών του διαδικτύου προς αρχιτεκτονικές μικροϋπηρεσιών (microservices) και η ανάπτυξη του Διαδικτύου των Πραγμάτων (Internet of Things - IoT) συνέβαλλαν σημαντικά στην αύξηση των αναγκών για νέες μεθόδους αποδοτικής επικοινωνίας μεταξύ ανομοιογενών και διανεμημένων συστημάτων. Οι μεθοδολογίες διαμεσολάβησης μηνυμάτων (*brokered messaging*) λειτουργούν καλύτερα από τις τεχνολογίες / προσεγγίσεις REST (Representational State Transfer) και RPC (Remote Procedure Call) σε συστήματα επικοινωνίας παραγωγών-καταναλωτών (μηνυμάτων) όπου είναι επιθυμητή τόσο η μετάδοση μεγάλου όγκου δεδομένων σε υψηλούς ρυθμούς όπως και η απεμπλοκή των υποσυστημάτων των παραγωγών και των καταναλωτών.

Μια ελαφριά και αξιόπιστη τεχνολογία που προσφέρει τα πλεονεκτήματα της μεσολάβησης μηνυμάτων είναι το RabbitMQ. Με τη χρήση αυτής, μπορούν να χτιστούν πολύπλοκα και αποδοτικά συστήματα ειδικά σε συνθήκες ασύγχρονης επικοινωνίας, αναξιόπιστων δικτύων και σε περιβάλλοντα εφαρμογών μεγάλων δεδομένων (big data).

Η παρούσα διπλωματική εστιάζει στην ανάπτυξη ενός εργαλείου πλήρους στοίβας (full-stack), το οποίο κάνει χρήση της τεχνολογίας διαμεσολάβησης μηνυμάτων για την εφαρμογή φίλτρων στα διακινούμενα μηνύματα του συστήματος. Η αυτοματοποίηση αυτών των λειτουργιών μέσω του εργαλείου, καθιστά τα ωφέλη των εμπλεκόμενων τεχνολογιών προσβάσιμα από τους χρήστες, ανεξάρτητα από το βαθμό της εμπειρίας τους στις συγκεκριμένες τεχνολογίες. Η επικοινωνία των μηνυμάτων επιτελείται μέσω του διακομιστή Rabbitmq Server, ο οποίος εφαρμόζει την τεχνολογία διαμεσολάβησης μηνυμάτων.

Τέλος για τη διευκόλυνση της διαχείρισης ολόκληρου του συστήματος, αυτό ενσωματώθηκε στο πλαίσιο της τεχνολογίας Kubernetes, η οποία προσφέρει την αυτοματοποίηση της ενορχήστρωσης των κομματιών του συστήματος. Για τη σύσταση του περιβάλλοντος Kubernetes επιλέχθηκε η τεχνολογία minikube καθώς προσφέρει εύκολη και γρήγορη δημιουργία ενός περιβάλλοντος Kubernetes.

Η απόδοση του συστήματος ελέγχθηκε για διαφορετικές τιμές του φορτίου εισαγωγής μηνυμάτων και των εφαρμοζόμενων φίλτρων. Τα μεγέθη που μετρήθηκαν αφορούν την συχνότητα εισόδου μηνυμάτων, τη συχνότητα κατανάλωσης μηνυμάτων, τη συχνότητα καταγραφής μηνυμάτων στη Βάση Δεδομένων και τον αριθμό αποθηκευμένων δεδομένων στις ουρές του διαμεσολαβητή.

Από τα πειράματα εξάγεται το συμπέρασμα ότι είναι ιδιαίτερα σημαντική η επιλογή του κατάλληλου αριθμού εφαρμοζόμενων φίλτρων σύμφωνα με τους διαθέσιμους πόρους επεξεργαστικής ισχύος και μνήμης του συστήματος.

---

# Implementation of a full stack tool in Kubernetes environment to automate the application of filters on messages using message broker technology

---

## Abstract

The transition of internet technologies to microservice architectures and the development of the Internet of Things (IoT) have significantly increased the need for new methods of efficient communication between heterogeneous and distributed systems. Brokered messaging methodologies work better than REST (Representational State Transfer) and RPC (Remote Procedure Call) technologies / approaches in producer-consumer (messaging) communication systems where both high-throughput transmission of large volumes of data is desirable as well as the abstraction of producer and consumer subsystems.

A lightweight and reliable technology that offers the benefits of brokered messaging is RabbitMQ. By using it, complex and efficient systems can be built under conditions of asynchronous communication, unreliable networks and within big data application environments.

This dissertation focuses on the full-stack development of a tool, which uses brokered messaging technology to implement filters on the messaging of a system. The automation of these functions through the tool, makes the effects of the involved technologies accessible to the users, regardless of the degree of their experience in the specific technologies. Messaging is carried out via a Rabbitmq Server which implements the brokered messaging technology.

Finally, to facilitate the management of the entire system, this was set up in the context of Kubernetes, which offers the automated orchestration of the parts of the system. For the establishment of the Kubernetes environment, the minikube technology was chosen as it offers easy and fast creation of a Kubernetes environment.

System performance was tested for different values of message input load and number of applied filters. The measured parameters refer to the frequency of message entry, the frequency of message consumption, the frequency of message logging to the Database and the number of messages stored in the broker queues.

From the experiments it is concluded that it is particularly important to select the appropriate number of applied filters according to the available processing power and memory resources of the system.

Zisis-Milis Emmanouil  
Electrical & Computer Engineering Department,  
Aristotle University of Thessaloniki, Greece  
September 2021

## Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή κ. Ανδρέα Συμεωνίδη για την εμπιστοσύνη που μου έδειξε με την ανάθεση της συγκεκριμένης διπλωματικής και την ευκαιρία να ασχοληθώ με τόσο ενδιαφέροντα θέματα.

Η υποστήριξη και η βοήθεια από τους συνεπιβλέποντες Δρ. Εμμανουήλ Τσαρδούλια και Υπ. Δρ. Κωσταντίνο Παναγιώτου ήταν κρίσιμες. Οι συμβουλές και η καθοδήγησή σας έκαναν δυνατό αυτό το ταξίδι γνώσης και η ένθερμη αντιμετώπισή σας ελάφρυναν το βάρος του δύσκολου αυτού επιτεύγματος. Σας ευχαριστώ βαθιά για τη συνεισφορά σας.

Ένα ευχαριστώ είναι λίγο για να εκφράσει την ευγνωμοσύνη που νιώθω για τους γονείς μου Ξενοφών Ζήση και Ελένη Μήλη, για όλα τα εφόδια που μου έδωσαν απλόχερα όλα αυτά τα χρόνια. Η αγάπη, η στήριξη, η ενθάρρυνση και η εμπιστοσύνη τους με βοήθησαν και με βοηθούν ακόμα να φτάνω όλο και πιο μακριά. Ένα ευχαριστώ ωφείλω και στη γιαγιά μου Ειρήνη Μήλη, που με τον δικό της τρόπο με φρόντιζε καθημερινά και με στηρίζει ακόμα με όλες τις δυνάμεις της. Ευχαριστώ επίσης τον αδερφό μου Γιάννη Ζήση-Μήλη που βρίσκεται πάντα στο πλευρό μου και μου δείχνει την αγάπη του.

Για την συμπαράσταση και την κατανόηση που μου έδειξαν οι φίλοι μου Δημήτρης και Ρομαρίκ, καθ' όλη τη διάρκεια της διπλωματικής αλλά και γενικά, χρωστώ και σε αυτούς ένα μεγάλο ευχαριστώ.

Τέλος θα ήθελα να επεκτείνω τις ευχαριστίες μου σε όλους τους συγγενείς και φίλους, οι οποίοι ήταν ή είναι μέρος της ζωής μου και την χρωματίζουν μοναδικά.

# Περιεχόμενα

<b>1</b>	<b>Εισαγωγή</b>	<b>9</b>
1.1	Κίνητρο	9
1.2	Περιγραφή προβλήματος	10
1.3	Στόχοι της διπλωματικής	10
1.4	Διάρθρωση	11
1.5	Κώδικας και αρχεία της εργασίας	12
<b>2</b>	<b>Υπόβαθρο</b>	<b>13</b>
2.1	Τεχνολογίες αυτοματοποίησης λογισμικού	13
2.1.1	Λογισμικό Αυτοματοποίησης	13
2.1.2	Πολίτες Προγραμματιστές	13
2.1.3	Ανάπτυξη Οδηγούμενη από Μοντέλα	14
2.1.4	Ανάπτυξη Χαμηλού Κώδικα / Ανάπτυξη Χωρίς Κώδικα	14
2.2	Διαμεσολάβηση μηνυμάτων	14
2.2.1	Advanced Message Queuing Protocol	15
2.2.2	Επιβεβαιώσεις Λήψης (Acknowledgements)	15
2.2.3	Απόρριψη Μηνυμάτων (Negative Acknowledgements)	16
2.2.4	Κόμβοι Ανταλλαγής (Exchanges)	16
2.2.5	Ουρές (Queues)	17
2.2.6	Παραγωγοί (Producers)	17
2.2.7	Καταναλωτές (Consumers)	17
2.2.8	Συνδέσεις & Κανάλια (Connections & Channels)	18
2.2.9	Εικονικοί Διαμεσολαβητές (Virtual Hosts)	18
2.2.10	Αυθεντικοποίηση και Εξουσιοδότηση στο RabbitMQ (Authentication & Authorization)	18
2.3	Kubernetes	19
2.3.1	Δοχεία και δοχείοποίηση εφαρμογών (containers & containerized apps)	19
2.3.2	Κόμβοι (Nodes)	20
2.3.3	Minikube	20
2.3.4	Περιγραφές ανάπτυξης εφαρμογών (Deployments)	20
2.3.5	Μονάδες λειτουργικότητας (Pods)	21
2.3.6	Υπηρεσίες (Services)	21
2.3.7	Διεχείριση αποθήκευσης (Volumes & Persistent Volumes)	22
2.3.8	Διαχείριση δεδομένων ρυθμίσεων (ConfigMaps & Secrets)	22
2.4	Frontend	23
2.5	Backend	24

2.6	Βάση Δεδομένων (Database)	24
2.6.1	MongoDB	24
<b>3</b>	<b>Επισκόπηση Ερευνητικής Περιοχής</b>	<b>26</b>
3.1	Ανάπτυξη σε cloud υποδομές	26
3.2	Τεχνολογίες Διαμεσολάβησης Μηνυμάτων	26
3.3	Software automation και Model driven τεχνικές	28
<b>4</b>	<b>Υλοποίηση</b>	<b>29</b>
4.1	Αρχιτεκτονική συστήματος	29
4.2	Διάταξη της υλοποίησης και διαθέσιμοι πόροι	31
4.2.1	Διαθέσιμα μηχανήματα	31
4.2.2	Εκχώρηση πόρων επεξεργασίας και μνήμης	32
4.3	Παρουσίαση των οντοτήτων του συστήματος	33
4.3.1	Backend	33
4.3.2	Frontend	35
4.3.3	Βάση Δεδομένων	38
4.3.4	Διαμεσολαβητής μηνυμάτων - RMQ Server	40
4.3.5	Φίλτρα / Καταναλωτές - Consumers	43
4.4	Διασύνδεση οντοτήτων μέσω K8s	46
4.4.1	Backend	46
4.4.2	Frontend	47
4.4.3	Βάση Δεδομένων	48
4.4.4	Διαμεσολαβητής μηνυμάτων - RMQ Server	49
4.4.5	Φίλτρα / Καταναλωτές - Consumers	49
4.4.6	Διαγραμματική αναπαράσταση των K8s οντοτήτων	50
<b>5</b>	<b>Παρουσίαση του συστήματος - Σενάρια Χρήσης</b>	<b>51</b>
5.1	Εγγραφή χρήστη	51
5.2	Σύνδεση χρήστη	53
5.3	Αποσύνδεση χρήστη	55
5.4	Δημιουργία RabbitMQ Server	56
5.5	Διαγραφή RabbitMQ Server	57
5.6	Χρήση του γραφικού περιβάλλοντος διαχείρισης του RabbitMQ Server	58
5.7	Δημιουργία Φίλτρου - Καταναλωτή μηνυμάτων	60
5.8	Διαγραφή Φίλτρου - Καταναλωτή μηνυμάτων	62
5.9	Εμφάνιση καταγεγραμμένων μηνυμάτων στη ΒΔ από Φίλτρο	63
5.10	Ενεργοποίηση ροής στατιστικών στοιχείων του συστήματος	64
<b>6</b>	<b>Πειράματα &amp; Αποτελέσματα</b>	<b>67</b>
6.1	Σταθερές παράμετροι των πειραμάτων	68
6.2	Ακρίβεια μετρήσεων	68
6.3	Πρώτος κύκλος πειραμάτων	69
6.3.1	1 Καταναλωτής   Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο	69
6.3.2	2 Καταναλωτές   Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο	70
6.3.3	4 Καταναλωτές   Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο	71

6.3.4	6 Καταναλωτές   Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο . . . . .	72
6.3.5	8 Καταναλωτές   Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο . . . . .	73
6.3.6	10 Καταναλωτές   Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο . . . . .	74
6.4	Δεύτερος κύκλος πειραμάτων . . . . .	75
6.4.1	Συχνότητα εισόδου 120 μηνύματα το δευτερόλεπτο   Εύρος Καταναλωτών 1 - 10 . . . . .	76
6.4.2	Συχνότητα εισόδου 240 μηνύματα το δευτερόλεπτο   Εύρος Καταναλωτών 1 - 10 . . . . .	76
6.4.3	Συχνότητα εισόδου 300 μηνύματα το δευτερόλεπτο   Εύρος Καταναλωτών 1 - 10 . . . . .	77
6.4.4	Συχνότητα εισόδου 360 μηνύματα το δευτερόλεπτο   Εύρος Καταναλωτών 1 - 10 . . . . .	78
6.4.5	Συχνότητα εισόδου 420 μηνύματα το δευτερόλεπτο   Εύρος Καταναλωτών 1 - 10 . . . . .	79
6.4.6	Συχνότητα εισόδου 480 μηνύματα το δευτερόλεπτο   Εύρος Καταναλωτών 1 - 10 . . . . .	80
6.5	Συμπεράσματα . . . . .	81
6.5.1	Επιρροή του αριθμού Καταναλωτών στη συμπεριφορά του συστήματος . . . . .	81
6.5.2	Επιρροή της συχνότητας εισόδου μηνυμάτων στη συμπεριφορά του συστήματος . . . . .	82
<b>7</b>	<b>Μελλοντική εργασία</b>	<b>86</b>
7.1	Επεκτάσεις του συστήματος . . . . .	86
7.1.1	Αύξηση διαθέσιμων ενεργειών για μηνύματα ενδιαφέροντος . . . . .	86
7.1.2	Εμπλουτισμός δυνατοτήτων παρακολούθησης και διαχείρισης μηνυμάτων . . . . .	87
7.1.3	Μεγαλύτερη ποικιλία συνθηκών εκτέλεσης ενεργειών . . . . .	87
7.1.4	Παροχή προτύπων παραμετροποίησης Φίλτρων για συγκεκριμένα σενάρια χρήσης . . . . .	88
7.1.5	Ανάπτυξη μιας Domain Specific Language (DSL) για την εφαρμογή φίλτρων . . . . .	88
7.1.6	Υποστήριξη διαφορετικών ρόλων χρηστών με κατάλληλες δυνατότητες . . . . .	88
7.1.7	Αυτοματοποίηση της κλιμάκωσης του συστήματος . . . . .	88
7.2	Ασφάλεια - Security context . . . . .	89
7.2.1	Ενεργοποίηση TLS στις επικοινωνίες . . . . .	89
7.2.2	Κρυπτογράφηση ΒΔ . . . . .	89
7.2.3	Εφαρμογή κανόνων τείχους προστασίας (firewall) . . . . .	89
7.2.4	Εκτέλεση των διεργασιών ως απλοί χρήστες - όχι ως root μέσα στα containers . . . . .	89
7.3	Ανάπτυξη του συστήματος σε περιβάλλον εμπορικής χρήσης . . . . .	90
<b>Α΄</b>	<b>Ακρωνύμια και συντομογραφίες</b>	<b>91</b>

# Κατάλογος Σχημάτων

2.1	Παράδειγμα δρομολόγησης μηνυμάτων στο πρωτόκολλο AMQP . . . . .	15
4.1	Παρουσίαση της αρχιτεκτονικής του συστήματος με τη χρήση Διαγράμματος Τμημάτων . . . . .	31
4.2	Παράδειγμα δρομολόγησης μηνυμάτων μέσω Topic Exchanges . . . . .	42
4.3	Παραδείγματα υλοποιημένων Φίλτρων με παραμέτρους δρομολόγησης και καταγραφής μηνυμάτων . . . . .	45
4.4	Διάγραμμα Οντοτήτων του K8s cluster σε συνδυασμό με τις εσωτερικές και εξωτερικές διασυνδέσεις του, όπως αναπτύχθηκε στα διαθέσιμα μηχανήματα. . . . .	50
5.1	Φόρμα εγγραφής που περιγράφεται στο σενάριο χρήσης "Εγγραφή Χρήστη"	52
5.2	Συμπληρωμένη φόρμα εγγραφής με αναγνωρισμένα σφάλματα που περιγράφεται στην εναλλακτική ροή "Διαχείριση Εσφαλμένων Στοιχείων" του σεναρίου χρήσης "Εγγραφή Χρήστη" . . . . .	53
5.3	Φόρμα σύνδεσης που περιγράφεται στο σενάριο χρήσης "Σύνδεση Χρήστη"	54
5.4	Γραφικό περιβάλλον προστατευμένου περιεχομένου ("Authenticated Content") της πλατφόρμας. Περιγράφεται στο σενάριο χρήσης "Σύνδεση Χρήστη" . . . . .	54
5.5	Αποτυχία εύρεσης του ονόματος χρήστη στη ΒΔ που περιγράφεται στην εναλλακτική ροή "Διαχείριση Εσφαλμένων Στοιχείων" του σεναρίου χρήσης "Σύνδεση Χρήστη" . . . . .	55
5.6	Αναντιστοιχία δοσμένων διαπιστευτηρίων από τον χρήστη με την εγγραφή της ΒΔ που περιγράφεται στην εναλλακτική ροή "Διαχείριση Εσφαλμένων Στοιχείων" του σεναρίου χρήσης "Σύνδεση Χρήστη" . . . . .	55
5.7	Στιγμιότυπο της μπάρας πλοήγησης με το κουμπί αποσύνδεσης. Περιγράφεται στο σενάριο χρήσης "Αποσύνδεση Χρήστη" . . . . .	56
5.8	Γραφικό περιβάλλον της δημιουργίας Διαμεσολαβητή Μηνυμάτων. Περιγράφεται στο σενάριο χρήσης "Δημιουργία Διαμεσολαβητή Μηνυμάτων" .	57
5.9	Στιγμιότυπο της καρτέλας "Message Broker" του γραφικού περιβάλλοντος με ενεργό Διαμεσολαβητή Μηνυμάτων. Περιγράφεται στο σενάριο χρήσης "Δημιουργία Διαμεσολαβητή Μηνυμάτων" . . . . .	57
5.10	Φόρμα σύνδεσης του RabbitMQ Management UI. Περιγράφεται στο σενάριο χρήσης "Πρόσβαση στο Γραφικό Περιβάλλον Διαχείρισης του RabbitMQ Server" . . . . .	59
5.11	Στιγμιότυπο του γραφικού περιβάλλοντος διαχείρισης του RabbitMQ Server. Περιγράφεται στο σενάριο χρήσης "Πρόσβαση στο Γραφικό Περιβάλλον Διαχείρισης του RabbitMQ Server" . . . . .	59



5.12 Φόρμα δημιουργίας παραμετροποιημένου Φίλτρου που περιγράφεται στο σενάριο χρήσης "Δημιουργία Φίλτρου"	61
5.13 Μήνυμα σφάλματος στη φόρμα δημιουργίας Φίλτρου. Περιγράφεται στο σενάριο χρήσης "Δημιουργία Φίλτρου"	62
5.14 Παράθυρο μηνυμάτων Φίλτρου που κάλυπταν τις συνθήκες καταγραφής του Φίλτρου και αποθηκεύτηκαν στη ΒΔ. Περιγράφεται στο σενάριο χρήσης "Εμφάνιση καταγεγραμμένων μηνυμάτων στη ΒΔ από Φίλτρο"	64
5.15 Γραφικό περιβάλλον της καρτέλας "Overview" με ανενεργή τη μετάδοση μετρήσεων. Περιγράφεται στο σενάριο χρήσης "Ενεργοποίηση Μετάδοσης Μετρήσεων Συστήματος"	65
5.16 Γραφικό περιβάλλον της καρτέλας "Overview" με ενεργή τη μετάδοση μετρήσεων και καθορισμένες τιμές χρονισμού των μετρήσεων. Περιγράφεται στο σενάριο χρήσης "Ενεργοποίηση Μετάδοσης Μετρήσεων Συστήματος"	66
6.1 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς παράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)	70
6.2 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)	70
6.3 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς παράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)	71
6.4 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)	71
6.5 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς παράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)	72
6.6 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)	72
6.7 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς παράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)	73
6.8 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)	73
6.9 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς παράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)	74
6.10 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)	74
6.11 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς παράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)	75
6.12 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)	75
6.13 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς παράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)	76

6.14 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε) . . . . .	76
6.15 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς πα- ράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίω- ση Λήψης (γαλάζιο) . . . . .	77
6.16 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε) . . . . .	77
6.17 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς πα- ράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίω- ση Λήψης (γαλάζιο) . . . . .	78
6.18 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε) . . . . .	78
6.19 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς πα- ράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίω- ση Λήψης (γαλάζιο) . . . . .	79
6.20 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε) . . . . .	79
6.21 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς πα- ράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίω- ση Λήψης (γαλάζιο) . . . . .	80
6.22 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε) . . . . .	80
6.23 Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο) Μηνύματα έτοιμα προς πα- ράδοση (κίτρινο) Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίω- ση Λήψης (γαλάζιο) . . . . .	81
6.24 Ρυθμός άφιξης μηνυμάτων (κόκκινο) Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο) Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε) . . . . .	81
6.25 Σύγκριση συχνοτήτων εκκίνησης αστάθειας των πειραμάτων του πρώτου κύκλου . . . . .	82
6.26 Σύγκριση των μέσων τιμών της εισαγωγής μηνυμάτων, της κατανάλωσης μηνυμάτων και της καταγραφής μηνυμάτων στη ΒΔ για κάθε πείραμα της δεύτερης φάσης. Άφιξη μηνυμάτων (κόκκινο) Κατανάλωση μηνυμάτων (κίτρινο) Καταγραφή μηνυμάτων στη ΒΔ (μπλε) . . . . .	83
6.27 Σύγκριση των τυπικών αποκλίσεων της εισαγωγής μηνυμάτων, της κατα- νάλωσης μηνυμάτων και της καταγραφής μηνυμάτων στη ΒΔ για κάθε πείραμα της δεύτερης φάσης. Άφιξη μηνυμάτων (κόκκινο) Κατανάλωση μηνυμάτων (κίτρινο) Καταγραφή μηνυμάτων στη ΒΔ (μπλε) . . . . .	84

# Κεφάλαιο 1

## Εισαγωγή

### 1.1 Κίνητρο

Το κίνητρο για την πραγματοποίηση αυτής της διπλωματικής είναι η αυτοματοποίηση της διαδικασίας διαχείρισης μηνυμάτων ενός συστήματος και η διευκόλυνση χρήσης αυτών των λειτουργιών μέσω της διεπαφής του χρήστη.

Αναλυτικότερα, τα μηνύματα αυτά μπορεί να περιέχουν μια πληθώρα από μετρήσεις, συμβάντα και πληροφορίες. Επίσης μπορεί να αναφέρονται σε διαφορετικά υπομέρη ενός συστήματος και να μπορούν να κατηγοριοποιηθούν με περισσότερους από έναν τρόπους, βάσει πολλαπλών κριτηρίων. Για αυτό το λόγο, ένα τέτοιο σύστημα εμπλουτίζεται σημαντικά με τη συνεισφορά των λειτουργιών επισήμανσης και ομαδοποίησης μηνυμάτων σε λογικά θέματα (topics).

Μία ακόμα διευκόλυνση που προσφέρει το υπό εξέταση σύστημα, είναι ο διαχωρισμός των μηνυμάτων σε μηνύματα ενδιαφέροντος και μη, με τα επιθυμητά μηνύματα να αποθηκεύονται σε μια Βάση Δεδομένων (ΒΔ) αμέσως μετά τη λήψη τους. Αυτός ο διαχωρισμός γίνεται με τον απευθείας από τον χρήστη ορισμό των πληροφοριών ενδιαφέροντος που πρέπει να περιέχει κάποιο μήνυμα για να είναι επιθυμητή η αποθήκευσή του.

Προστιθέμενη αξία προσφέρει η παροχή πρόσβασης στους χρήστες σε εργαλεία διαχείρισης και παρακολούθησης της κατάστασης των μηνυμάτων τους. Μέσα από αυτά οι χρήστες μπορούν να βλέπουν τον ρυθμό με τον οποίο τα μηνύματα αποστέλλονται και λαμβάνονται, καθώς και το ποσοστό αυτών που καλύπτουν τις συνθήκες ενδιαφέροντος και καταλήγουν στη ΒΔ.

Τέλος η δημιουργία ολόκληρου του συστήματος με τη χρήση δοχειοποιημένων εφαρμογών μέσα σε περιβάλλον Kubernetes προσδίδει τρία βασικά προτερήματα.

Το πρώτο είναι ότι με τη χρήση των containers το σύστημα μπορεί να αναπτυχθεί σε διαφορετικές υποδομές με σχετική ευκολία, καθώς όλες οι βιβλιοθήκες και τα εργαλεία που απαιτούνται για τη λειτουργία του συστήματος, εμπεριέχονται στα containers.

Ακόμη η χρήση του Kubernetes προσδίδει μια επιπρόσθετη ανθεκτικότητα σε τυχαίες και μη προβλεπόμενες αποτυχίες των επιμέρους υποσυστημάτων της εφαρμογής.

Σαν τελευταίο πλεονέκτημα αναφέρεται η χαρακτηριστική ευκολία με την οποία μπορεί να γίνει οριζόντια κλιμάκωση του συστήματος μέσα σε ένα περιβάλλον Kubernetes. Σε απλοποιημένους όρους, σε συνθήκες υψηλού φόρτου μπορούν να αναπτυχθούν επιπλέον αντίγραφα της εφαρμογής για εξυπηρέτηση των αιτημάτων. Αυτό μπορεί να γίνει είτε με ρητές εντολές του διαχειριστή του συστήματος, είτε εφαρμόζοντας κάποια πολιτική που να υποδεικνύει τα όρια πάνω από τα οποία η κλιμάκωση θα συμβαίνει αυτόματα.

## 1.2 Περιγραφή προβλήματος

Ο ρυθμός διακίνησης μηνυμάτων σε μια σύγχρονη εφαρμογή μπορεί να έχει πολλές διακυμάνσεις ή και να φτάνει πολύ υψηλές τιμές (όπως συχνά συμβαίνει στον τομέα του IoT). Η δυσκολία της διαχείρισης ενός τέτοιου όγκου επικοινωνίας μέσω των γνωστών πρωτοκόλλων REST / RPC οδηγεί στην ανάγκη για χρήση ενός πρωτοκόλλου που να μπορεί να ανταπεξέλθει σε αυτές τις προκλήσεις [1] [2].

Τέτοιου είδους συστήματα πολλές φορές πρέπει να λειτουργήσουν κάτω από συνθήκες αναξιόπιστων δικτύων. Αυτό σημαίνει ότι οι δικτυακές συνδέσεις των εξαρτημάτων και των συσκευών μπορεί να διακόπτονται και να επιδιορθώνονται με τυχαία συχνότητα. Η χρήση πρωτοκόλλων σύγχρονης επικοινωνίας με αυτά τα δεδομένα μπορεί να γίνει ιδιαίτερα προβληματική [3].

Ένα επιπλέον πρόβλημα είναι η ετερογένεια που χαρακτηρίζει συνήθως αυτού του είδους τα συστήματα. Τα υποσυστήματα των παραγωγών των μηνυμάτων συνήθως έχουν αρκετά διαφορετική μορφή, τοπολογία και απαιτήσεις από τα υποσυστήματα των καταναλωτών των μηνυμάτων [4]. Η απεικονική λοιπόν αυτών των δύο συστατικών του συστήματος είναι κρίσιμη ώστε να διευκολύνεται και να απλοποιείται η λειτουργία του.

Τέλος, η χρήση της τεχνολογίας διαμεσολάβησης μηνυμάτων προϋποθέτει εξοικείωση με τα στοιχεία και τις μεθόδους από τα οποία αυτή απαρτίζεται. Αυτό σημαίνει ότι τα ωφέλη της τεχνολογίας διαμεσολάβησης μηνυμάτων είναι εκ πρώτης όψεως αξιοποιήσιμα μόνο από άτομα με βαθιά γνώση και εμπειρία σε αυτές τις τεχνικές. Όμως τα συστήματα στα οποία η τεχνολογία αυτή βρίσκει εφαρμογή, περιλαμβάνουν ομάδες χρηστών που χαρακτηρίζονται από διαφορετικά τεχνικά προσόντα. Έτσι κρίνεται σημαντικό να υπάρχουν εργαλεία τα οποία να καλύπτουν αυτή την έλλειψη τεχνογνωσίας πάνω στη διαμεσολάβηση μηνυμάτων από τους χρήστες.

## 1.3 Στόχοι της διπλωματικής

Ός συνολικός στόχος της διπλωματικής ορίστηκε η παροχή εύχρηστων εργαλείων εφαρμογής της τεχνολογίας διαμεσολάβησης μηνυμάτων στα συστήματα των χρηστών και η εφαρμογή Φίλτρων παρακολούθησης, κατανάλωσης και επεξεργασίας μηνυμάτων σε αυτά τα συστήματα.

Ειδικότερα στοχεύει στην υλοποίηση ενός εργαλείου εφαρμογής φίλτρων μηνυμάτων μέσω της τεχνολογίας διαμεσολάβησης μηνυμάτων, το οποίο δεν απαιτεί γνώσεις προγραμματισμού για τη χρήση του.

Στη συνέχεια εξετάζεται η ανάπτυξη του εν λόγω εργαλείου σε τοπικό σύστημα και πραγματοποιούνται έλεγχοι για την εξεύρεση των ορίων του συστήματος. Επίσης διερευνάται η επίδραση συγκεκριμένων παραμέτρων στην απόδοση του συστήματος.

Οι βασικές λειτουργικές απαιτήσεις του συστήματος που αναπτύχθηκε στα πλαίσια της διπλωματικής είναι οι εξής:

- ✓ Οι χρήστες πρέπει να μπορούν να εγγραφούν στο σύστημα και να διατηρούν στοιχεία για τα φίλτρα μηνυμάτων τους.
- ✓ Οι χρήστες πρέπει να μπορούν να δημιουργήσουν έναν RabbitMQ Server για τους ίδιους και να έχουν πρόσβαση στις υπηρεσίες παρακολούθησής του.
- ✓ Το σύστημα πρέπει να παρέχει στον κάθε χρήστη μια IP διεύθυνση στην οποία να μπορεί ο τελευταίος να δρομολογήσει μηνύματα προς τον RabbitMQ Server του.
- ✓ Οι χρήστες πρέπει να μπορούν να δημιουργήσουν φίλτρα μηνυμάτων, τα οποία να παρακολουθούν συγκεκριμένα θέματα (topics) του RabbitMQ Server και τα οποία να παραμένουν ενεργά μέχρι ο χρήστης να παύσει την λειτουργία τους.
- ✓ Οι χρήστες πρέπει να μπορούν να εφαρμόσουν συνθήκες διαφόρων μορφών στα Φίλτρα τους, κάτω από τις οποίες τα μηνύματα να θεωρούνται μηνύματα ενδιαφέροντος και να αποθηκεύονται σε μία κατάλληλη Βάση Δεδομένων.
- ✓ Το σύστημα πρέπει να παρέχει βασικά εργαλεία επισκόπησης στους χρήστες σχετικά με τα ενεργά φίλτρα τους και τα μηνύματα τα οποία αυτά τα φίλτρα επεξεργάζονται.
- ✓ Πρέπει να πραγματοποιηθεί δοχειοποίηση των επιμέρους κομματιών του συστήματος και το σύστημα να αναπτυχθεί μέσα σε περιβάλλον Kubernetes.

## 1.4 Διάρθρωση

Η διάρθρωση της παρούσας διπλωματικής εργασίας είναι η εξής:

- Κεφάλαιο 1: Εισαγωγή
- Κεφάλαιο 2: Υπόβαθρο [Το θεωρητικό και πρακτικό υπόβαθρο που πρέπει να γνωρίζει ο αναγνώστης για να κατανοήσει το κείμενο]
- Κεφάλαιο 3: Επισκόπηση Ερευνητικής Περιοχής
- Κεφάλαιο 4: Υλοποίηση [Περιγραφή της λειτουργικότητας του συστήματος που αναπτύχθηκε]
- Κεφάλαιο 5: Παρουσίαση του συστήματος [Παρουσίαση use cases σε συνδυασμό με στιγμιότυπα της διεπαφής του χρήστη]
- Κεφάλαιο 6: Πειράματα & Αποτελέσματα
- Κεφάλαιο 7: Μελλοντική εργασία

## 1.5 Κώδικας και αρχεία της εργασίας

Ο κώδικας του συστήματος που αναπτύχθηκε και τα αρχεία που χρησιμοποιήθηκαν είναι ελεύθερα προσβάσιμα στην τοποθεσία: <https://github.com/Provaldo/DiplomaThesis>.

Η τοποθεσία δημοσίευσης των containers που χρησιμοποιήθηκαν είναι η ακόλουθη: <https://hub.docker.com/u/provaldo>.

# Κεφάλαιο 2

## Υπόβαθρο

Παρακάτω θα παρουσιαστούν σύντομα βασικές έννοιες και εργαλεία που χρησιμοποιούνται στη συνέχεια της διπλωματικής. Αυτές οι πληροφορίες κρίνονται απαραίτητες για την κατανόηση της υπόλοιπης εργασίας.

### 2.1 Τεχνολογίες αυτοματοποίησης λογισμικού

#### 2.1.1 Λογισμικό Αυτοματοποίησης

Στη βάση του, το *Λογισμικό Αυτοματοποίησης (Automation Software)* αναφέρεται στη μετατροπή επαναλαμβανόμενων εργασιών σε αυτοματοποιημένες ενέργειες. Κάτι τέτοιο είναι πολύ χρήσιμο καθώς στη συνέχεια, μπορεί να υλοποιηθεί και η αυτοματοποίηση των διαφόρων διαδικασιών που αποτελούνται από αυτοματοποιημένες ενέργειες. Αυτό απλοποιεί πολλές επιχειρηματικές εργασίες και διευκολύνει τη διαχείρισή τους<sup>1</sup>.

Ο αυτοματισμός διαδικασιών επομένως οδηγεί σε αυξημένη παραγωγικότητα και αποδοτικότητα, καθώς οι τυποποιημένες διεργασίες αφήνονται στους υπολογιστές. Αυτό απελευθερώνει το ανθρώπινο δυναμικό να αφοσιώνεται σε πιο δημιουργικές εργασίες.

#### 2.1.2 Πολίτες Προγραμματιστές

Ο όρος *Πολίτης Προγραμματιστής (Citizen Developer)* αναφέρεται σε υπαλλήλους ή επιχειρηματικούς χρήστες με μικρή εμπειρία προγραμματισμού, οι οποίοι δημιουργούν λύσεις λογισμικού. Σε αυτή την κατηγορία μπορεί να ανήκουν και άτομα τα οποία έχουν τεχνολογική κατάρτιση, όμως σε κάποιο συγκεκριμένο τομέα προγραμματισμού δεν έχουν αντίστοιχη εμπειρία και γνώσεις. Σε αυτόν τον τομέα επομένως αναγνωρίζονται ως Πολίτες Προγραμματιστές<sup>2</sup>.

<sup>1</sup>Ορισμός της Αυτοματοποίησης Λογισμικού: <https://www.automationanywhere.com/rpa/automation-software>

<sup>2</sup>Ορισμός των Πολιτών Προγραμματιστών: <https://www.gartner.com/en/information-technology/glossary/citizen-developer>

### 2.1.3 Ανάπτυξη Οδηγούμενη από Μοντέλα

Ως *Ανάπτυξη Οδηγούμενη από Μοντέλα (Model Driven Development)* ορίζεται η μεθοδολογία ανάπτυξης λογισμικού που επιτρέπει στους χρήστες να δημιουργούν λύσεις λογισμικού μέσω απλοποιημένων αφαιρέσεων προκατασκευασμένων στοιχείων. Αποτελεί τη βασική αρχή Ανάπτυξης Χαμηλού Κώδικα η οποία εξηγείται στη συνέχεια.

### 2.1.4 Ανάπτυξη Χαμηλού Κώδικα / Ανάπτυξη Χωρίς Κώδικα

Η *Ανάπτυξη Χαμηλού Κώδικα (Low-code Development)* είναι μια οπτική προσέγγιση στην ανάπτυξη λογισμικού που επιτρέπει την ταχύτερη παράδοση εφαρμογών μέσω ελάχιστης χρήσης προγραμματισμού<sup>3</sup>. Στην ουσία χρησιμοποιούνται γραφικές διεπαφές χρήστη και δυνατότητες ενεργοποίησης και τοποθέτησης προκαθορισμένων στοιχείων ελαχιστοποιώντας την ανάγκη για παραδοσιακό προγραμματισμό.

Παρόμοια με την Ανάπτυξη Χαμηλού Κώδικα, μια πλατφόρμα Ανάπτυξης Χωρίς Κώδικα επιτρέπει στους χρήστες να δημιουργούν λογισμικό εφαρμογών μέσω διεπαφών μεταφοράς και απόθεσης στοιχείων αντί για προγραμματισμό.

Οι πλατφόρμες Ανάπτυξης Χαμηλού Κώδικα / Χωρίς Κώδικα καθιστούν δυνατή την ανάπτυξη εφαρμογών και λύσεων λογισμικού από Πολίτες Προγραμματιστές, ενώ διευκολύνουν και επιταχύνουν τη δουλειά επαγγελματιών προγραμματιστών.

Αυτές οι πλατφόρμες έχουν αυξηθεί σε δημοτικότητα, καθώς οι εταιρίες αντιμετωπίζουν τις τάσεις ενός ολοένα και πιο κινητού εργατικού δυναμικού και μιας περιορισμένης προσφοράς ικανών προγραμματιστών λογισμικού.

## 2.2 Διαμεσολάβηση μηνυμάτων

Η τεχνολογία γύρω από την οποία αναπτύχθηκε αυτή η διπλωματική είναι η Διαμεσολάβηση Μηνυμάτων (Brokered Messaging).

Ένας Διαμεσολαβητής Μηνυμάτων (Message Broker) τοποθετείται μεταξύ μιας υπηρεσίας παραγωγής μηνυμάτων και μιας υπηρεσίας κατανάλωσής τους. Τα μηνύματα αποστέλλονται στον Διαμεσολαβητή όπου αποθηκεύονται σε κατάλληλες δομές που ονομάζονται Ουρές (Queues) και στη συνέχεια δρομολογούνται στους συνδεδεμένους καταναλωτές ανάλογα με προκαθορισμένους κανόνες.

Ο Διαμεσολαβητής Μηνυμάτων που επιλέχθηκε προς χρήση της τεχνολογίας είναι το RabbitMQ<sup>4</sup>. Αποτελεί έναν ευρέως διαδεδομένο Διαμεσολαβητή Μηνυμάτων ανοιχτού κώδικα ο οποίος υποστηρίζει διάφορα πρωτόκολλα. Το RabbitMQ μπορεί να αναπτυχθεί σε διανεμημένα συστήματα για να ικανοποιήσει απαιτήσεις μεγάλης κλίμακας και υψηλής διαθεσιμότητας.

Παρακάτω θα παρουσιαστούν κάποιες βασικές έννοιες αυτού του Διαμεσολαβητή.

<sup>3</sup>Περιγραφή Ανάπτυξης Χαμηλού Κώδικα / Χωρίς Κώδικα: <https://www.ibm.com/cloud/learn/low-code>

<sup>4</sup>Επίσημη ιστοσελίδα του Διαμεσολαβητή RabbitMQ: <https://www.rabbitmq.com/>

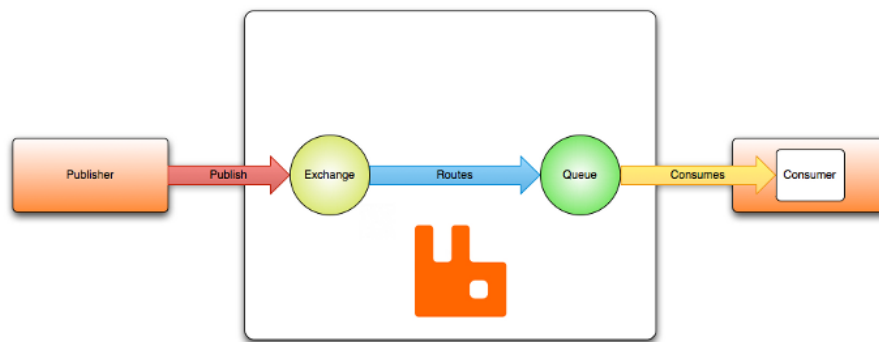


### 2.2.1 Advanced Message Queuing Protocol

Το AMQP είναι ένα πρωτόκολλο ανταλλαγής μηνυμάτων το οποίο επιτρέπει σε εφαρμογές την αποστολή και λήψη μηνυμάτων μέσω ενός Διαμεσολαβητή. Αποτελεί ένα πρωτόκολλο επιπέδου εφαρμογής που χρησιμοποιεί το Transmission Control Protocol (TCP) για να εγγυηθεί την αξιοπιστία της παράδοσης.

Η έκδοση του πρωτοκόλλου που χρησιμοποιήθηκε είναι η 0-9-1 σύμφωνα με την οποία η ανταλλαγή μηνυμάτων λειτουργεί ως εξής:

1. Τα μηνύματα δημοσιεύονται σε αρχικούς Κόμβους Ανταλλαγής (Exchanges) του Διαμεσολαβητή, η λειτουργία των οποίων συχνά παρομοιάζεται με ένα ταχυδρομικό κιβώτιο.
2. Στη συνέχεια οι αρχικοί κόμβοι διανέμουν αντίγραφα των μηνυμάτων σε Ουρές, σύμφωνα με κάποιους κανόνες (Bindings) στους οποίους ορίζεται η σύνδεση των Κόμβων με τις Ουρές.
3. Εντέλει ο Διαμεσολαβητής είτε παραδίδει τα μηνύματα στους Καταναλωτές που έχουν εγγραφεί σε Ουρές, είτε οι Καταναλωτές λαμβάνουν μηνύματα από Ουρές κατόπιν δικών τους αιτημάτων.



Σχήμα 2.1: Παράδειγμα δρομολόγησης μηνυμάτων στο πρωτόκολλο AMQP

### 2.2.2 Επιβεβαιώσεις Λήψης (Acknowledgements)

Καθώς τα δίκτυα μπορεί να είναι αναξιόπιστα, ή να συντρέξουν και άλλοι λόγοι για τους οποίους οι εφαρμογές ενδέχεται να αποτύχουν στην επεξεργασία των μηνυμάτων, στο πρωτόκολλο AMQP συμπεριλαμβάνεται μια μέθοδος αντιμετώπισης αυτού του προβλήματος: οι Επιβεβαιώσεις Λήψης μηνυμάτων.

Όταν αυτές είναι ενεργοποιημένες, ο Καταναλωτής στέλνει μια ειδοποίηση στον Διαμεσολαβητή Μηνυμάτων ότι το μήνυμα έχει ληφθεί και η επεξεργασία του έχει ολοκληρωθεί επιτυχώς. Αυτό μπορεί να συμβεί είτε αυτόματα κατά τη λήψη του μηνύματος από τον Καταναλωτή, είτε ρητά όταν τελειώσει η επεξεργασία του.

Στην περίπτωση που οι Επιβεβαιώσεις Λήψης μηνυμάτων είναι ενεργοποιημένες, τα μηνύματα αφαιρούνται από τις ουρές του Διαμεσολαβητή μόνο όταν αυτός δεχθεί την Επιβεβαίωση Λήψης τους από τον Καταναλωτή. Σε αντίθετη περίπτωση, αν παρέλθει ένα συγκεκριμένο χρονικό όριο χωρίς να δεχθεί Επιβεβαίωση Λήψης για κάποιο μήνυμα ο Διαμεσολαβητής, το δρομολογεί εκ νέου σε έναν Καταναλωτή.

### 2.2.3 Απόρριψη Μηνυμάτων (Negative Acknowledgements)

Υπάρχει η περίπτωση ένας Καταναλωτής να μην είναι σε θέση να επεξεργαστεί κάποιο μήνυμα που έχει λάβει από μια Ουρά. Τότε μπορεί να υποδείξει αυτή την κατάσταση στον Διαμεσολαβητή απορρίπτοντας το μήνυμα. Τα μηνύματα που απορρίπτονται, είτε θα ξαναεισαχθούν στην ουρά, είτε θα διαγραφούν πλήρως, σύμφωνα με την επιλογή του Καταναλωτή. Μεγάλη προσοχή απαιτείται όταν ξαναγίνεται εισαγωγή του μηνύματος στην ουρά ώστε να αποφεύγονται καταστάσεις ατέρμονων βρόγχων παράδοσης, απόρριψης και εισαγωγής του ίδιου μηνύματος στην Ουρά.

Ο Διαμεσολαβητής RabbitMQ προσφέρει μια υλοποίηση αυτής της μεθόδου: τις Επιβεβαιώσεις Απόρριψης (Negative Acknowledgements). Αυτές μπορούν να χρησιμοποιηθούν αντί των Επιβεβαιώσεων Λήψης, έτσι ώστε όλα τα μηνύματα να θεωρούνται καλώς ληφθέντα εκτός από αυτά για τα οποία έχει σταλεί Επιβεβαίωση Απόρριψης από τον Καταναλωτή στον Διαμεσολαβητή.

### 2.2.4 Κόμβοι Ανταλλαγής (Exchanges)

Οι Κόμβοι Ανταλλαγής αποτελούν τον πρώτο σταθμό στον οποίο στέλνονται τα μηνύματα. Αυτοί αναλαμβάνουν να τα προωθήσουν σε Ουρές όπως ορίζεται στους κανόνες που τους έχουν δοθεί κατά τη δημιουργία τους. Αυτοί οι κανόνες περιγράφουν με ποιες Ουρές συνδέεται ο κάθε Κόμβος Ανταλλαγής, τι τύπος Κόμβου Ανταλλαγής είναι και ποιο είναι το Κλειδί Σύνδεσης (Binding Key) ενός Κόμβου Ανταλλαγής με μια Ουρά. Κάθε σύνδεση ενός Κόμβου Ανταλλαγής με μια Ουρά χαρακτηρίζεται από ένα Κλειδί Σύνδεσης.

Στο πρωτόκολλο AMQP 0-9-1 ορίζονται 4 διαφορετικοί τύποι Κόμβων Ανταλλαγής:

- *Άμεσος κόμβος* (Direct Exchange): Ένας άμεσος Κόμβος Ανταλλαγής προωθεί ένα μήνυμα σε μια συνδεδεμένη Ουρά, μόνο αν το Κλειδί Δρομολόγησης του μηνύματος ταυτίζεται με το Κλειδί Σύνδεσης του Κόμβου Ανταλλαγής με την Ουρά.
- *Κόμβος ευρείας μετάδοσης* (Fanout Exchange): Όπως δηλώνει και το όνομά του, αυτός ο τύπος Κόμβου Ανταλλαγής προωθεί αντίγραφα των μηνυμάτων του σε κάθε Ουρά με την οποία είναι συνδεδεμένος αγνοώντας το Κλειδί Δρομολόγησης.
- *Κόμβος θεμάτων* (Topic Exchange): Στους Κόμβους Ανταλλαγής θεμάτων ορίζεται ένα Πρότυπο Δρομολόγησης σε κάθε σύνδεση του Κόμβου με μια Ουρά αντί για Κλειδί Σύνδεσης. Τα μηνύματα του Κόμβου συνεπώς προωθούνται σε μια Ουρά εάν το Κλειδί Δρομολόγησης των μηνυμάτων ταιριάζει με το Πρότυπο Δρομολόγησης που έχει οριστεί για τη συγκεκριμένη σύνδεση. Αυτό επιτρέπει τον έλεγχο πολλαπλών κριτηρίων κατά τη δρομολόγηση και για αυτό χρησιμοποιήθηκε αυτός ο τύπος Κόμβων Ανταλλαγής στο σύστημα που αναπτύχθηκε.
- *Κόμβος κεφαλίδων* (Headers Exchange): Σε αυτή την περίπτωση το Κλειδί Δρομολόγησης του μηνύματος αγνοείται και στη θέση του ελέγχονται οι παράμετροι κεφαλίδας του κάθε μηνύματος. Σε περίπτωση που ταιριάζουν με το Κλειδί Σύνδεσης μεταξύ ενός Κόμβου Ανταλλαγής κεφαλίδας και μιας Ουράς, τότε προωθούνται σε αυτήν.

Οι Κόμβοι Ανταλλαγής μπορεί να είναι μόνιμοι (Durable) ή παροδικοί (Transient). Οι μόνιμοι Κόμβοι Ανταλλαγής επιβιώνουν από μια επανεκκίνηση του Διαμεσολαβητή,

ενώ οι παροδικοί όχι (πρέπει να δηλωθούν ξανά όταν ο Διαμεσολαβητής επιστρέψει σε λειτουργία).

### **2.2.5 Ουρές (Queues)**

Οι Ουρές αποτελούν τη βασική δομή βραχυπρόθεσμης αποθήκευσης μηνυμάτων στον Διαμεσολαβητή, πριν αυτά παραδοθούν στους Καταναλωτές για επεξεργασία. Είναι δομές σειριακών δεδομένων που λειτουργούν με τη μέθοδο FIFO - First In First Out.

Για να χρησιμοποιηθεί μια Ουρά πρέπει να έχει προηγηθεί ο ορισμός της. Αν μια Ουρά δεν υπάρχει, ο ορισμός της θα οδηγήσει ταυτόχρονα και στη δημιουργία της, ενώ δε θα συμβεί τίποτα σε αντίθετη περίπτωση.

Αντίστοιχα με τους Κόμβους Ανταλλαγής, και οι Ουρές μπορούν να οριστούν ως μόνιμες (Durable) ή παροδικές (Transient). Στην πρώτη περίπτωση η περιγραφή και τα μεταδεδομένα της Ουράς αποθηκεύονται στον δίσκο, ενώ στη δεύτερη περίπτωση αποθηκεύονται στη μνήμη όταν είναι αυτό δυνατό. Όταν η ανθεκτικότητα σε επανεκκινήσεις του Διαμεσολαβητή είναι επιθυμητή, τότε πρέπει να ορίζονται εκτός από τις Ουρές και τα ίδια τα μηνύματα ως μόνιμα κατά τη δημοσίευσή τους στον Κόμβο Ανταλλαγής.

### **2.2.6 Παραγωγοί (Producers)**

Παραγωγοί είναι οι εφαρμογές οι οποίες συνδέονται σε έναν Διαμεσολαβητή Μηνυμάτων, αυθεντικοποιούνται, εγκαθιδρύουν μια σύνδεση μέσα στην οποία ανοίγουν ένα κανάλι και τέλος δημοσιεύουν τα παραγώμενα μηνύματά τους σε έναν Κόμβο Ανταλλαγής.

Οι σύνδεση του Παραγωγού με τον Διαμεσολαβητή είναι μακράς διάρκειας, δηλαδή εγκαθιδρύεται μια φορά και όλα τα μηνύματα που πρέπει να αποσταλούν στη διάρκεια ζωής του Παραγωγού στέλνονται μέσω αυτής της σύνδεσης.

Ένας Παραγωγός επιτρέπεται επίσης να λαμβάνει και μηνύματα από τον Διαμεσολαβητή, επομένως να λειτουργεί και ως Καταναλωτής.

### **2.2.7 Καταναλωτές (Consumers)**

Μια εφαρμογή μπορεί να λειτουργήσει ως Καταναλωτής στο πλαίσιο του Διαμεσολαβητή Μηνυμάτων με έναν από τους δύο ακόλουθους τρόπους:

1. Χρησιμοποιώντας τη συνιστώμενη μέθοδο της εγγραφής σε μια Ουρά. Με αυτό τον τρόπο, όποτε εισέρχονται νέα μηνύματα στην Ουρά, αυτά ωθούνται στην εφαρμογή - Καταναλωτή (Push API).
2. Ελέγχοντας επανειλημμένα (Polling) την Ουρά για την ύπαρξη νέων μηνυμάτων σε αυτήν (Pull API). Αυτή η μέθοδος όμως θεωρείται ιδιαίτερα αναποτελεσματική και στις περισσότερες περιπτώσεις θα πρέπει να αποφεύγεται.

Οι σύνδεση του Καταναλωτή με τον Διαμεσολαβητή είναι μακράς διάρκειας, δηλαδή εγκαθιδρύεται μια φορά και όλα τα μηνύματα που πρέπει να ληφθούν στη διάρκεια ζωής του Καταναλωτή λαμβάνονται μέσω αυτής της σύνδεσης.

Ένας Καταναλωτής επιτρέπεται επίσης να στέλνει και μηνύματα στον Διαμεσολαβητή, επομένως να λειτουργεί και ως Παραγωγός.

### **2.2.8 Συνδέσεις & Κανάλια (Connections & Channels)**

Οι συνδέσεις μεταξύ Παραγωγών / Καταναλωτών και Διαμεσολαβητή στο πρωτόκολλο AMQP 0-9-1 είναι ορισμένες έτσι ώστε να είναι μακράς διάρκειας. Έτσι συνιστάται η εδραίωση μιας σύνδεσης και η χρήση της για όλες της αποστολές και λήψεις μηνυμάτων από μια εφαρμογή, αντί της δημιουργίας μιας σύνδεσης για κάθε ανταλλαγή ενός μηνύματος. Οι συνδέσεις κάνουν χρήση αυθεντικοποίησης και μπορούν να προστατευτούν με τη χρήση του πρωτοκόλλου Transport Layer Security (TLS).

Ορισμένες εφαρμογές χρειάζονται πολλαπλές συνδέσεις με τον Διαμεσολαβητή. Ωστόσο, είναι ανεπιθύμητο να διατηρούνται πολλές συνδέσεις TCP ανοικτές ταυτόχρονα, διότι αυτό καταναλώνει πόρους συστήματος και καθιστά δυσκολότερη τη ρύθμιση του τείχους προστασίας (Firewall). Οι συνδέσεις AMQP 0-9-1 είναι πολυπλεγμένες με Κανάλια που μπορούν να θεωρηθούν ως “ελαφριές συνδέσεις που μοιράζονται μία μόνο σύνδεση TCP”.

Η επικοινωνία σε ένα συγκεκριμένο Κανάλι είναι εντελώς ξεχωριστή από την επικοινωνία σε άλλο Κανάλι. Επίσης ένα Κανάλι υπάρχει μόνο στο πλαίσιο μιας Σύνδεσης και ποτέ από μόνο του. Όταν κλείνει μια Σύνδεση, κλείνουν όλα τα Κανάλια σε αυτήν.

Για εφαρμογές που χρησιμοποιούν πολλαπλά νήματα / διαδικασίες για επεξεργασία, είναι πολύ συνηθισμένο να ανοίγουν ένα νέο Κανάλι ανά νήμα / διαδικασία και να μην μοιράζονται Κανάλια μεταξύ τους.

### **2.2.9 Εικονικοί Διαμεσολαβητές (Virtual Hosts)**

Για να καταστεί δυνατό για έναν διαμεσολαβητή να φιλοξενεί πολλαπλά απομονωμένα “περιβάλλοντα” (ομάδες χρηστών, Κόμβοι Ανταλλαγής, Ουρές και ούτω καθεξής), το AMQP 0-9-1 περιλαμβάνει την έννοια των Εικονικών Διαμεσολαβητών (vhosts). Αυτή η έννοια είναι παρόμοια με εικονικούς κεντρικούς υπολογιστές που χρησιμοποιούνται από πολλούς δημοφιλείς (web servers) και παρέχουν εντελώς απομονωμένα περιβάλλοντα στα οποία ζουν οντότητες AMQP. Οι χρήστες του πρωτοκόλλου καθορίζουν ποιον vhost θέλουν να χρησιμοποιήσουν κατά την εγκαθίδρυση της Σύνδεσης.

### **2.2.10 Αυθεντικοποίηση και Εξουσιοδότηση στο RabbitMQ (Authentication & Authorization)**

Αρχικά ένας σαφής διαχωρισμός των δύο αυτών εννοιών κρίνεται απαραίτητος:

- Η Αυθεντικοποίηση αναφέρεται στην αναγνώριση της ταυτότητας κάποιου χρήστη.
- Η Εξουσιοδότηση περιγράφει τα δικαιώματα και τις δυνατότητες που έχει κάποιος χρήστης πάνω σε συγκεκριμένους πόρους του συστήματος.

Κάθε ενέργεια στον Διαμεσολαβητή RabbitMQ γίνεται μετά από μια Σύνδεση με αυτόν. Για να συμβεί αυτό, κάποιος χρήστης παρέχει τα διαπιστευτήριά του και αιτείται τη σύνδεσή του με κάποιον Εικονικό Διαμεσολαβητή, για τον οποίο να έχει τα αντίστοιχα δικαιώματα.

Όταν ο Διαμεσολαβητής RabbitMQ εκκινεί για πρώτη φορά, αρχικοποιεί τη Βάση Δεδομένων (ΒΔ) του έτσι ώστε να υπάρχει ένας προκαθορισμένος Εικονικός Διαμεσολαβητής με όνομα `"/` και ένας προκαθορισμένος χρήστης με τα εξής διαπιστευτήρια: όνομα χρήστη `"guest"` και κωδικό `"guest"`. Αυτός ο χρήστης έχει πλήρη πρόσβαση στον Εικονικό Διαμεσολαβητή `"/`. Για λόγους ασφαλείας συνιστάται να παρέχεται ένας διαφορετικός χρήστης με νέα διαπιστευτήρια στη θέση του προκαθορισμένου κατά τη δημιουργία του Διαμεσολαβητή.

Η Αυθεντικοποίηση μπορεί να διεκπεραιωθεί είτε με τη χρήση ονόματος χρήστη/κωδικού, είτε με πιστοποιητικά τύπου X.509.

Με την ολοκλήρωση της Αυθεντικοποίησης, ο Διαμεσολαβητής ελέγχει αν ο επιβεβαιωμένος χρήστης έχει τα απαραίτητα δικαιώματα για να αποκτήσει πρόσβαση στον Εικονικό Διαμεσολαβητή τον οποίο προσδιόρισε στο αίτημα σύνδεσης. Στη συνέχεια για την εκτέλεση κάθε ενέργειας στον Διαμεσολαβητή ελέγχονται τα δικαιώματα του χρήστη που σχετίζονται με την ενέργεια και τους πόρους του συστήματος πάνω στους οποίους αυτή εκτελείται. Ένας πόρος μπορεί να είναι ένας Κόμβος Ανταλλαγής ή μια ουρά σε έναν Εικονικό Διαμεσολαβητή, ενώ τα δικαιώματα που ορίζονται στο RabbitMQ είναι τα εξής:

- `configure`: δημιουργία και διαγραφή πόρου ή μετατροπή της συμπεριφοράς του
- `write`: εγγραφή μηνυμάτων στον πόρο
- `read`: ανάγνωση μηνυμάτων από τον πόρο

## 2.3 Kubernetes

Συνεχίζοντας με τις τεχνολογίες που χρησιμοποιήθηκαν, ιδιαίτερη σημασία έχει το Kubernetes (K8s). Η λειτουργία του συνιστά την αυτοματοποίηση της ανάπτυξης, της αυξομειώσεως κλίμακας και της διαχείρισης δοχειοποιημένων εφαρμογών. Στην ουσία ομαδοποιεί δοχεία που αποτελούν μια εφαρμογή σε λογικές μονάδες για εύκολη διαχείριση και ανακάλυψη των υπηρεσιών τους. Το τελικό αποτέλεσμα είναι ένα σύμπλεγμα (Cluster) κόμβων και πόρων, οι οποίοι συντελούν την εφαρμογή.

Αυτό το πετυχαίνει συνδυάζοντας ένα πλήθος από υποσυστήματα τα οποία συνεργάζονται, προσφέροντας υπηρεσίες απαραίτητες για τη λειτουργία του K8s. Το επίπεδο ελέγχου (Control Plane) του K8s συντονίζει όλες τις δραστηριότητες του συμπλέγματος όπως προγραμματισμός εφαρμογών, διατήρηση της επιθυμητής κατάστασης εφαρμογών, κλιμάκωση εφαρμογών και εφαρμογή νέων ενημερώσεων.

Παρακάτω θα παρουσιαστούν βασικά στοιχεία και έννοιες του K8s, τα οποία θα συναντήσει ο αναγνώστης στη συνέχεια της διπλωματικής.

### 2.3.1 Δοχεία και δοχειοποίηση εφαρμογών (containers & containerized apps)

Με τη *δοχειοποίηση εφαρμογών* εννοείται η συσκευασία του κώδικα λογισμικού με τις βιβλιοθήκες του λειτουργικού συστήματος και τις εξαρτήσεις που απαιτούνται για την εκτέλεση του κώδικα για τη δημιουργία ενός μόνο ελαφρού εκτελέσιμου - που ονομάζεται *container* - το οποίο λειτουργεί με συνέπεια σε οποιαδήποτε υποδομή.

Η περιγραφή των βημάτων για τη συλλογή και εγκατάσταση των απαραίτητων κομματιών ενός container ονομάζεται εικόνα του container (container image).

Καθώς είναι πιο φορητά και αποδοτικά ως προς τους πόρους από τις εικονικές μηχανές (Virtual Machines - VMs), τα container έχουν γίνει οι κύριες υπολογιστικές μονάδες σύγχρονων εφαρμογών νέφους.

### 2.3.2 Κόμβοι (Nodes)

Ένας Κόμβος είναι ένας εικονικός ή ένας φυσικός υπολογιστής που χρησιμεύει ως μηχανή εργασίας σε ένα K8s cluster. Κάθε Κόμβος περιέχει ένα Kubelet, το οποίο είναι μια διεργασία υπεύθυνη για τη διαχείριση του Κόμβου και την επικοινωνία με το Επίπεδο Ελέγχου του K8s. Ο Κόμβος επίσης διαθέτει κάποιο εργαλείο για το χειρισμό λειτουργιών των containers, όπως το containerd ή το Docker. Ένα K8s cluster που δέχεται όλη την κίνηση μιας εφαρμογής πραγματικού περιβάλλοντος (Production Environment Application) πρέπει να αποτελείται από τουλάχιστον τρεις κόμβους.

Όταν αναπτύσσονται εφαρμογές στο K8s, δίνεται η οδηγία στο Επίπεδο Ελέγχου να ξεκινήσει τα Containers των εφαρμογών. Το Επίπεδο Ελέγχου προγραμματίζει τα Containers να λειτουργούν στους Κόμβους του Cluster. Οι Κόμβοι επικοινωνούν με το Επίπεδο Ελέγχου χρησιμοποιώντας το K8s API.

### 2.3.3 Minikube

Ένα K8s cluster μπορεί να αναπτυχθεί είτε σε φυσικές είτε σε εικονικές μηχανές. Για μια εισαγωγική ενασχόληση με την ανάπτυξη σε K8s, μπορεί να χρησιμοποιηθεί το Minikube<sup>5</sup>. Το Minikube είναι μια ελαφριά εφαρμογή K8s που αναπτύσσει ένα K8s cluster που περιέχει μόνο έναν κόμβο σε έναν τοπικό υπολογιστή. Αυτό γίνεται είτε με την ανάπτυξη του K8s cluster μέσα σε ένα VM είτε σε ένα Container. Το Minikube είναι διαθέσιμο για συστήματα Linux, macOS και Windows.

Η διασύνδεση της γραμμής εντολών του Minikube (Minikube CLI - Command Line Interface) παρέχει βασικές λειτουργίες εκκίνησης για την εκτέλεση εργασιών με ένα K8s cluster, συμπεριλαμβανομένης της έναρξης, διακοπής, εμφάνισης κατάστασης και διαγραφής του Cluster.

### 2.3.4 Περιγραφές ανάπτυξης εφαρμογών (Deployments)

Μόλις ολοκληρωθεί η εκκίνηση ενός K8s cluster, μπορούν να αναπτυχθούν εφαρμογές σε Container. Για να συμβεί αυτό, πρέπει να δημιουργηθεί μια περιγραφή των επιθυμητών ρυθμίσεων της εφαρμογής για το K8s περιβάλλον. Αυτή η περιγραφή ονομάζεται Deployment. Το Deployment δίνει οδηγίες στο K8s πώς να δημιουργήσει και να ενημερώσει οντότητες της εφαρμογής. Μόλις δημιουργηθεί ένα Deployment, το επίπεδο ελέγχου προγραμματίζει τις οντότητες εφαρμογών που περιλαμβάνονται σε αυτό το Deployment να εκτελούνται σε μεμονωμένους Κόμβους στο Cluster.

Αφού δημιουργηθούν οι οντότητες της εφαρμογής, ένας ελεγκτής των K8s Deployments παρακολουθεί συνεχώς αυτές τις οντότητες. Εάν ο κόμβος που φιλοξενεί μια

---

<sup>5</sup>Επίσημη ιστοσελίδα του Minikube: <https://minikube.sigs.k8s.io/docs/>

οντότητα βγει εκτός λειτουργίας ή διαγραφεί, ο ελεγκτής αντικαθιστά τη χαμένη οντότητα με μια αντίστοιχη σε έναν άλλο κόμβο του cluster. Αυτό παρέχει έναν μηχανισμό αυτοθεραπείας για την αντιμετώπιση βλαβών ή συντήρησης των μηχανημάτων.

Όταν δημιουργείται ένα Deployment, πρέπει να καθορίζονται οι εικόνες των εφαρμογών για τα Container και ο αριθμός των αντιγράφων των εφαρμογών που είναι επιθυμητό να δημιουργηθούν. Αυτές οι πληροφορίες μπορούν να αλλάχθούν αργότερα ενημερώνοντας το Deployment.

### 2.3.5 Μονάδες λειτουργικότητας (Pods)

Τα Pods είναι η μικρότερη αυτόνομη μονάδα στην πλατφόρμα K8s. Κάθε Pod αντιπροσωπεύει ένα μέρος του φόρτου εργασίας που εκτελείται στο Cluster. Περιέχει μια ομάδα από ένα ή περισσότερα Container εφαρμογών και μερικούς κοινούς πόρους για αυτά τα Container. Οι πόροι αυτοί μπορεί να περιλαμβάνουν μεταξύ άλλων:

- Κοινόχρηστους αποθηκευτικούς χώρους που ονομάζονται Volumes
- Υπηρεσίες δικτύωσης, όπως μια μοναδική διεύθυνση IP στο Cluster
- Πληροφορίες σχετικά με τον τρόπο εκτέλεσης κάθε Container, όπως η εικόνα του Container ή συγκεκριμένες θύρες προς χρήση

Το Pod συστεγάζει διαφορετικά Container της εφαρμογής που είναι σχετικά στενά συνδεδεμένα. Τα Container σε ένα Pod μοιράζονται μια διεύθυνση IP και το σύνολο των θυρών Ports. Επίσης προγραμματίζονται από το επίπεδο ελέγχου με τον ίδιο ακριβώς τρόπο ώστε να συγχρονίζονται και να εκτελούνται πάντα σε κοινό πλαίσιο στον ίδιο κόμβο.

Όταν δημιουργείται ένα K8s deployment, το επίπεδο ελέγχου δημιουργεί Pods με Containers μέσα τους (σε αντίθεση με τη δημιουργία απευθείας Containers). Αυτό σημαίνει ότι δεν υπάρχει η δυνατότητα να δημιουργηθεί απευθείας κάποιο Container, αντιθέτως πρέπει να έχει περιγραφεί ως μέλος ενός Pod. Κάθε Pod συνδέεται με τον κόμβο όπου έχει προγραμματιστεί και παραμένει εκεί μέχρι τον τερματισμό (σύμφωνα με την πολιτική επανεκκίνησης) ή τη διαγραφή. Σε περίπτωση αποτυχίας του κόμβου, τα ίδια Pods προγραμματίζονται σε άλλους διαθέσιμους κόμβους στο Cluster.

### 2.3.6 Υπηρεσίες (Services)

Μια Υπηρεσία στο K8s ομαδοποιεί ένα λογικό σύνολο από Pods και προσδιορίζει κάποιο καθορισμένο τρόπο με τον οποίο παρέχεται πρόσβαση σε αυτά.

Παρόλο που κάθε Pod έχει μια μοναδική διεύθυνση IP, αυτές οι διευθύνσεις δεν εκτίθενται εκτός του Cluster χωρίς την εφαρμογή μιας Υπηρεσίας. Οι Υπηρεσίες επιτρέπουν στις εφαρμογές να λαμβάνουν επισκεψιμότητα από τοποθεσίες εντός και εκτός του Cluster. Οι τρόποι με τους οποίους μια Υπηρεσία παρέχει πρόσβαση στα σχετιζόμενα με αυτήν Pods είναι οι εξής:

- *ClusterIP* (προεπιλογή) - Εκθέτει την Υπηρεσία σε μια εσωτερική Cluster IP. Αυτός ο τύπος καθιστά την Υπηρεσία προσβάσιμη μόνο από το Cluster.
- *NodePort* - Εκθέτει την Υπηρεσία στην ίδια θύρα κάθε επιλεγμένου κόμβου στο Cluster χρησιμοποιώντας τη μέθοδο Network Address Translation (NAT). Κάνει

μία Υπηρεσία προσβάσιμη εκτός του συμπλέγματος χρησιμοποιώντας τη διεύθυνση <NodeIP>:<NodePort>. Υπερσύνολο του ClusterIP.

- *LoadBalancer* - Δημιουργεί έναν εξωτερικό εξισορροπητή φορτίου στο τρέχον cloud (εάν υποστηρίζεται) και εκχωρεί μια σταθερή, εξωτερική IP στην Υπηρεσία. Υπερσύνολο του NodePort.
- *ExternalName* - Χαρτογραφεί την Υπηρεσία στα περιεχόμενα του πεδίου externalName (π.χ. foo.bar.example.com), επιστρέφοντας μια εγγραφή CNAME με την τιμή της. Δεν εγκαθιδρύεται κανενός είδους proxying.

Η χρησιμότητα των Υπηρεσιών αναδεικνύεται με τη δυνατότητα που προσφέρει στα Pods να καταστρέφονται και να επαναδημιουργούνται στο K8s χωρίς να επηρεάζεται η λειτουργία της εφαρμογής.

### 2.3.7 Διεχείριση αποθήκευσης (Volumes & Persistent Volumes)

Η διαχείριση αποθηκευτικού χώρου είναι ένα διαφορετικό πρόβλημα από τη διαχείριση υπολογιστικών πόρων.

Τα αρχεία στο δίσκο σε ένα Container είναι εφήμερα, γεγονός που παρουσιάζει δύο βασικά προβλήματα. Ένα πρόβλημα είναι η απώλεια αρχείων σε περίπτωση αποτυχίας ενός Container. Το Container επανεκκινείται αλλά με μηδενικά δεδομένα. Ένα δεύτερο πρόβλημα παρουσιάζεται κατά την κοινή χρήση αρχείων μεταξύ Container που λειτουργούν μαζί σε ένα Pod. Καθώς τα Container του ίδιου Pod μοιράζονται το σύστημα αρχείων, μπορεί να προκληθούν race conditions. Το K8s δίνει λύση σε αυτά τα προβλήματα με τη χρήση Τόμων Αποθήκευσης (Volumes).

Το K8s υποστηρίζει πολλούς τύπους Τόμων Αποθήκευσης. Ένα Pod μπορεί να χρησιμοποιήσει ταυτόχρονα διαφορετικούς τύπους Τόμων Αποθήκευσης. Οι τύποι εφήμερων Τόμων μοιράζονται τη διάρκεια ζωής ενός Pod, αλλά υπάρχουν και Τόμοι Διατήρησης (Persistent Volumes - PV) οι οποίοι διατηρούνται ανεξάρτητα από τη διάρκεια ζωής ενός Pod. Όταν ένα Pod παύει να υπάρχει, το K8s καταστρέφει τους εφήμερους Τόμους, όμως όχι και τα Persistent Volumes. Για οποιοδήποτε είδος Τόμου σε ένα δεδομένο Pod, τα δεδομένα διατηρούνται σε κάθε επανεκκίνηση κάποιου Container.

Για την χρήση των Persistent Volumes απαιτείται η καταχώρηση ενός αιτήματος κατανάλωσης αυτών των πόρων αποθήκευσης από κάποιον χρήστη. Αυτά τα αιτήματα κατανάλωσης αποθηκευτικού χώρου ονομάζονται Persistent Volume Claims (PVC). Τα PVCs μπορούν να ζητήσουν συγκεκριμένο μέγεθος αποθηκευτικού χώρου και τρόπους πρόσβασης σε αυτόν.

### 2.3.8 Διαχείριση δεδομένων ρυθμίσεων (ConfigMaps & Secrets)

Για τη διευκόλυνση της διαχείρισης δεδομένων ρυθμίσεων ενός συστήματος, το K8s προσφέρει δύο χρήσιμα εργαλεία: τα ConfigMaps και τα Secrets.

Τα ConfigMaps παρέχουν έναν τρόπο εισαγωγής δεδομένων ρυθμίσεων σε Pods. Είναι αντικείμενα που χρησιμοποιούνται για την αποθήκευση μη εμπιστευτικών δεδομένων σε ζεύγη κλειδιών-τιμών. Τα Pods μπορούν να καταναλώνουν ConfigMaps ως



μεταβλητές περιβάλλοντος, ορίσματα γραμμής εντολών ή ως αρχεία διαμόρφωσης σε έναν τόμο αποθήκευσης.

Τα ConfigMaps επιτρέπουν την αποσύνδεση των παραμέτρων που σχετίζονται με το περιβάλλον από τα Container Images, έτσι ώστε οι εφαρμογές να είναι εύκολα φορητές.

Δεν έχουν σχεδιαστεί για να περιέχουν μεγάλα κομμάτια δεδομένων. Τα δεδομένα που αποθηκεύονται σε ένα ConfigMap δεν μπορούν να υπερβαίνουν το 1 MB. Εάν πρέπει να αποθηκευτούν ρυθμίσεις που είναι μεγαλύτερες από αυτό το όριο, ίσως χρειαστεί να χρησιμοποιηθούν Τόμοι Αποθήκευσης ή μια ξεχωριστή βάση δεδομένων ή υπηρεσία αρχείων.

Επίσης δεν παρέχουν μυστικότητα ή κρυπτογράφηση. Εάν τα δεδομένα που πρέπει να αποθηκευτούν είναι εμπιστευτικά, συνιστάται η χρήση ενός Secret και όχι ενός ConfigMap ή η χρήση επιπλέον εργαλείων (τρίτων) για να διατηρηθεί η ιδιωτικότητα των δεδομένων.

Τα Secrets είναι παρόμοια με τα ConfigMaps αλλά προορίζονται ειδικά για τη αποθήκευση εμπιστευτικών δεδομένων.

Επειδή τα Secrets μπορούν να δημιουργηθούν ανεξάρτητα από τα Pods που τα χρησιμοποιούν, υπάρχει μικρότερος κίνδυνος να εκτίθεται το Secret (και τα δεδομένα του) κατά τη ροή εργασιών δημιουργίας, προβολής και επεξεργασίας Pods. Το K8s και οι εφαρμογές που εκτελούνται στο Cluster, μπορούν επίσης να λάβουν πρόσθετες προφυλάξεις με τα Secrets, όπως η αποφυγή εγγραφής εμπιστευτικών δεδομένων σε μη πτητικό αποθηκευτικό χώρο.

Τα Secrets αποθηκεύονται, από προεπιλογή, χωρίς κρυπτογράφηση στη ΒΔ του K8s API server (etcd). Οποιοσδήποτε έχει πρόσβαση στο API μπορεί να ανακτήσει ή να τροποποιήσει ένα Secret, και το ίδιο μπορεί να κάνει και οποιοσδήποτε έχει πρόσβαση στη ΒΔ etcd. Επιπλέον, οποιοσδήποτε έχει εξουσιοδότηση για τη δημιουργία Pods μπορεί να χρησιμοποιήσει αυτήν την πρόσβαση για να διαβάσει οποιοδήποτε Secret. Αυτό περιλαμβάνει την έμμεση πρόσβαση, όπως η εξουσιοδότηση δημιουργίας ενός Deployment (κάτι το οποίο με τη σειρά του θα δημιουργήσει Pods).

Για την ασφαλή χρήση των Secrets, τα ακόλουθα βήματα είναι κρίσιμα:

- Ενεργοποίηση της κρυπτογράφησης των Secrets κατά την αποθήκευσή τους στη ΒΔ etcd.
- Ενεργοποίηση ή διαμόρφωση κανόνων Role-based Access Control (RBAC) που να περιορίζουν την ανάγνωση δεδομένων στα Secrets (συμπεριλαμβανομένης της έμμεσης πρόσβασης).
- Όπου ενδείκνυται, να γίνεται περιορισμός μέσω RBAC ως προς το ποιοι χρήστες επιτρέπεται να δημιουργούν νέα Secrets ή να αντικαθιστούν υπάρχοντα.

## 2.4 Frontend

Ως *Frontend* ορίζεται το σύνολο των στοιχείων και γραφικών με τα οποία αλληλεπιδρά ο χρήστης ενός συστήματος, μιας εφαρμογής, μιας ιστοσελίδας και ούτω καθεξής.

Στο Frontend πραγματοποιείται η αναπαράσταση όλων των πληροφοριών και των λειτουργιών ενός συστήματος. Η εκτέλεση των λειτουργιών αυτών και η αποθήκευση και διαχείριση των αντίστοιχων δεδομένων συνήθως λαμβάνουν χώρα σε ένα άλλο σημείο του συστήματος όπως είναι το Backend.

## 2.5 Backend

Ο όρος *Backend* περιγράφει όλες τις λειτουργίες που πραγματοποιούνται στο παρασκήνιο ενός συστήματος. Το Backend είναι υπεύθυνο να οργανώνει και να συγχρονίζει τα μέρη ενός συστήματος τα οποία δεν είναι ορατά στον χρήστη. Η τυπική ροή διεργασιών υποδεικνύει ότι οι αλληλεπιδράσεις του χρήστη με το Frontend, εκκινεί λειτουργίες στο Backend οι οποίες παράγουν ή επεξεργάζονται δεδομένα από άλλα κομμάτια του συστήματος. Αφού ολοκληρωθούν οι παρασκηνιακές διεργασίες, τα αποτελέσματα οδηγούνται πίσω στο Frontend και παρουσιάζονται στον χρήστη.

## 2.6 Βάση Δεδομένων (Database)

Μια *Βάση Δεδομένων* είναι μια οργανωμένη συλλογή δομημένων πληροφοριών ή δεδομένων, που συνήθως αποθηκεύονται ηλεκτρονικά σε ένα σύστημα υπολογιστή<sup>6</sup>. Μια βάση δεδομένων ελέγχεται συνήθως από ένα σύστημα διαχείρισης βάσεων δεδομένων (Database Management System - DBMS). Μαζί, τα δεδομένα και το DBMS, μαζί με τις εφαρμογές που σχετίζονται με αυτά, αναφέρονται ως σύστημα βάσης δεδομένων, συχνά συντομευμένο σε *Βάση Δεδομένων*.

Τα δεδομένα στους πιο συνηθισμένους τύπους Βάσεων Δεδομένων που λειτουργούν σήμερα τυπικά διαμορφώνονται σε γραμμές και στήλες σε μια σειρά πινάκων για να κάνουν την επεξεργασία και την αναζήτηση δεδομένων αποτελεσματική. Τα δεδομένα μπορούν στη συνέχεια να έχουν εύκολη πρόσβαση, διαχείριση, τροποποίηση, ενημέρωση, έλεγχο και οργάνωση.

### 2.6.1 MongoDB

Το MongoDB είναι μια βάση δεδομένων NoSQL προσανατολισμένη σε έγγραφα που χρησιμοποιείται για αποθήκευση δεδομένων μεγάλου όγκου. Ο όρος *NoSQL* έχει τη σημασία ότι παρέχεται ένας μηχανισμός αποθήκευσης και ανάκτησης δεδομένων που διαμορφώνεται με άλλα μέσα, διαφορετικά από τις σχέσεις πίνακα που χρησιμοποιούνται στις παραδοσιακές σχεσιακές βάσεις δεδομένων. Συγκεκριμένα το MongoDB χρησιμοποιεί συλλογές και έγγραφα.

Μια εγγραφή στη MongoDB είναι ένα έγγραφο, το οποίο είναι μια δομή δεδομένων που αποτελείται από ζεύγη πεδίων και τιμών. Τα έγγραφα MongoDB είναι παρόμοια με τα αντικείμενα JSON (JavaScript Object Notation). Οι τιμές των πεδίων μπορεί να περιλαμβάνουν άλλα έγγραφα, πίνακες και πίνακες εγγράφων.

Η MongoDB αποθηκεύει έγγραφα σε συλλογές. Οι συλλογές είναι ανάλογες με τους πίνακες στις σχεσιακές βάσεις δεδομένων.

---

<sup>6</sup>Ορισμός της Βάσης Δεδομένων από την εταιρία Oracle: <https://www.oracle.com/database/what-is-database/>

Στη MongoDB βάση δεδομένων καλείται ένα φυσικό δοχείο για συλλογές. Κάθε βάση δεδομένων λαμβάνει το δικό της σύνολο αρχείων στο σύστημα αρχείων. Ένας διακομιστής MongoDB μπορεί να περιλαμβάνει πολλές βάσεις δεδομένων.

## Κεφάλαιο 3

# Επισκόπηση Ερευνητικής Περιοχής

Λόγω του πλήθους των ερευνητικών περιοχών οι οποίες σχετίζονται με αυτή τη διπλωματική εργασία, γίνεται μια επιμέρους ανάλυση του κάθε ερευνητικού τομέα, δίνοντας χρήσιμες πληροφορίες για την τρέχουσα κατάσταση.

### 3.1 Ανάπτυξη σε cloud υποδομές

Το cloud computing, αναφέρεται στην παροχή πρόσβασης σε υπολογιστικούς πόρους και στην κατανάλωση αυτών. Ο επίσημος ορισμός του από το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας των Η.Π.Α. είναι ο εξής: Το *Cloud Computing* είναι ένα μοντέλο που επιτρέπει την ανεξάρτητη της τοποθεσίας, βολική, κατά απαίτηση πρόσβαση μέσω δικτύου σε ένα κοινόχρηστο σύνολο παραμετροποιήσιμων υπολογιστικών πόρων (δίκτυα, εικονικές μηχανές, τόμοι αποθήκευσης, εφαρμογές και υπηρεσίες) που μπορούν να δεσμευτούν και να απελευθερωθούν με ελάχιστη προσπάθεια διαχείρισης ή αλληλεπίδραση του παρόχου υπηρεσιών [5].

Η τάση για μετάβαση σε τεχνολογίες και υποδομές Νέφους είναι εύλογη λόγω των πλεονεκτημάτων που προσφέρει αυτό το μοντέλο. Σύμφωνα με την έκθεση υιοθεσίας Νέφους του 2021 από την εταιρία Security Compass, το 50% του συνολικού φόρτου εργασίας των επιχειρήσεων σήμερα βρίσκεται σε δημόσιες εφαρμογές Νέφους [6] [7].

### 3.2 Τεχνολογίες Διαμεσολάβησης Μηνυμάτων

Σε ότι αφορά τις τεχνολογίες Διαμεσολάβησης Μηνυμάτων, ενδιαφέρον παρουσιάζει η σύγκριση μεταξύ των διαθέσιμων πρωτοκόλλων, οι δυνατότητές τους και οι διάφορες εφαρμογές στις οποίες το κάθε πρωτόκολλο εμφανίζει προτερήματα.

Τα αποτελέσματα έρευνας σύγκρισης πρωτοκόλλων του επιπέδου εφαρμογών στον τομέα του IoT, έδειξαν ότι το CoAP (Constrained Application Protocol) αποτελεί την πιο κατάλληλη επιλογή εάν μια εφαρμογή χρειάζεται λειτουργικότητα REST. Λόγω χαμηλής κατανάλωσης ενέργειας και μικρού μεγέθους κεφαλίδων μηνυμάτων το MQTT (Message Queuing Telemetry Transport Protocol) είναι το πιο δημοφιλές πρωτόκολλο στο IoT. Παρότι το AMQP αποτελεί ένα δυαδικό και ελαφρύ πρωτόκολλο, έχει λίγο

αυξημένο μέγεθος κεφαλίδων μηνύματος λόγω της υποστήριξης μεθόδων ασφαλείας, αξιοπιστίας και διαλειτουργικότητας και που προσφέρει. Τέλος το XMPP (Extensible Messaging Protocol) είναι το πιο βαρύ πρωτόκολλο. Απαιτεί μεγαλύτερο μέγεθος κεφαλίδων μηνυμάτων καθώς δεν είχε σχεδιαστεί αρχικά για το IoT. Το XMPP είναι καλύτερο σε εφαρμογές που υποστηρίζουν multi-threading, λόγω χαμηλότερης χρήσης του Διακομιστή [8].

Αντίστοιχη σύγκριση πραγματοποιήθηκε και μεταξύ των πρωτοκόλλων AMQP, MQTT, Kafka, ZeroMQ για τον προσδιορισμό της καταλληλότητάς τους σε περιβάλλοντα βιομηχανικών συστημάτων παραγωγής [9]. Ανάμεσα στα συμπεράσματα της σύγκρισης προκύπτει πως το AMQP τείνει να προσφέρει έναν καλό συμβιβασμό μεταξύ μεγιστης βιώσιμης απόδοσης και καθυστερήσεων. Επίσης αναγνωρίστηκε ότι έχει μακράν τις περισσότερες ενσωματωμένες λειτουργικότητες και τη μεγαλύτερη ευελιξία και ως εκ τούτου μπορεί να θεωρηθεί ως πολύ ισορροπημένη τεχνολογία.

Σε μία πιο ειδικευμένη σύγκριση μεταξύ του MQTT και του AMQP σε ασταθή και mobile δίκτυα [10], παρατηρήθηκε ότι τα δύο πρωτόκολλα εμφανίζουν παρόμοια συμπεριφορά, με το MQTT να έχει λίγο καλύτερο όριο ωφέλιμου φορτίου μηνύματος κυρίως λόγω των μικρότερων κεφαλίδων του. Το AMQP έχει κεφαλίδα σταθερού μεγέθους 8 bytes ενώ η κεφαλίδα του MQTT αποτελείται από 2 bytes μόνο. Συνεπώς η χρήση του AMQP συνίσταται για τη δημιουργία αξιόπιστων, επεκτάσιμων και προηγμένων δομών ανταλλαγής μηνυμάτων λόγω των χαρακτηριστικών ασφαλείας που παρέχει [11], ενώ το πρωτόκολλο MQTT ενδείκνυται για χρήση σε κόμβους ακμής των συστημάτων που λειτουργούν σε δίκτυα περιορισμένης και ασταθούς συνδεσιμότητας λόγω της καλύτερης ενεργειακής αποδοτικότητάς του [12].

Σχετικά με τις συγκεκριμένες υλοποιήσεις των Διαμεσολαβητών, από τη σύγκριση που πραγματοποιήθηκε μεταξύ δύο διαδεδομένων διαμεσολαβητών (του RabbitMQ και του ActiveMQ) [13], αποδείχθηκε ότι το ActiveMQ είναι ελαφρώς ταχύτερο στη δημοσίευση μηνυμάτων στον Διαμεσολαβητή. Αντιθέτως, τα ίδια τεστ έδειξαν ότι το RabbitMQ είναι κατά πολύ γρηγορότερο στην παράδοση μηνυμάτων από τον Διαμεσολαβητή στους Καταναλωτές. Παρατηρήθηκε επίσης ότι όσο περισσότεροι Παραγωγοί δημοσιεύουν μηνύματα στις Ουρές του Διαμεσολαβητή η απόδοση μειώνεται. Αυτή η συμπεριφορά ήταν κοινή και στους δύο Διαμεσολαβητές με το RabbitMQ να παρουσιάζει ελαφρώς μικρότερη επιδείνωση της απόδοσης.

Οι τεχνολογίες Διαμεσολάβησης Μηνυμάτων έχουν βρει εφαρμογή τόσο σε μικρά συστήματα όσο και σε συστήματα μεγάλης κλίμακας.

Μια μικρού μεγέθους εφαρμογή αυτής της τεχνολογίας παρουσιάστηκε στην εργασία [14], όπου μια Βάση Δεδομένων χρησιμοποιούνταν ως μέθοδος επικοινωνίας μεταξύ των οντοτήτων ενός συστήματος καταστήματος ηλεκτρονικού εμπορίου. Με την αντικατάσταση της ΒΔ από δύο Διαμεσολαβητές Μηνυμάτων (RabbitMQ και ActiveMQ), επιτεύχθηκε βελτίωση της απόδοσης κατά 17% στον χρόνο επεξεργασίας των τιμολογίων.

Ένα παράδειγμα μεγάλης κλίμακας είναι το έργο ENCOURAGE το οποίο στόχευε στην ανάπτυξη τεχνολογιών έξυπνου δικτύου ηλεκτρικής ενέργειας στην πλευρά των καταναλωτών [15]. Το πρωτόκολλο AMQP χρησιμοποιήθηκε με την υλοποίηση του RabbitMQ για τη διαχείριση των δεδομένων του δικτύου. Πάνω σε αυτή την τεχνολογία αναπτύχθηκαν λειτουργίες σχετικές με την απρόσκοπτη αλληλεπίδραση υφιστάμενων

συσκευών από διαφορετικούς προμηθευτές, τον τηλεχειρισμό συσκευών των χρηστών και τις ενεργειακές ανταλλαγές μεταξύ γειτόνων.

### 3.3 Software automation και Model driven τεχνικές

Η χρήση των προηγούμενων τεχνολογιών συνδυάζεται με τις αρχές της αυτοματοποίησης λογισμικού με ενδιαφέροντα αποτελέσματα. Με στόχο την αυτοματοποίηση της δημιουργίας και διαχείρισης RabbitMQ clusters και του πειραματικού ελέγχου τέτοιων συστημάτων, έχει προταθεί η εξής αρχιτεκτονική: Χρήση της τεχνολογίας Kubernetes και ανάπτυξη του συστήματος σε υπολογιστές μονής πλακέτας Raspberry pi, σε συνδυασμό με τους μηχανισμούς οριζόντιας κλιμάκωσης και αυτοθεραπείας που προσφέρει το Kubernetes [16].

Παρά τα πλεονεκτήματα που προσφέρουν όμως, οι τεχνικές Ανάπτυξης Λογισμικού Οδηγούμενης από Μοντέλα προσεγγίζονται με επιφυλακτικότητα καθώς σύμφωνα με κάποιες έρευνες, τα ωφέλη τους μπορεί να είναι σημαντικά σε ακαδημαϊκές πρακτικές και σε Πολίτες Προγραμματιστές μόνο [17] [18]. Παρόλα αυτά, σύμφωνα με πειράματα σε πραγματικές εργασίες στον τομέα της ανάπτυξης παιχνιδιών, οι Model driven τεχνικές βελτίωσαν την ορθότητα και την αποτελεσματικότητα των αποτελεσμάτων κατά 41% και 54% αντίστοιχα [19]. Τα αποτελέσματα αυτά έρχονται σε σύγκρουση με τους προηγούμενους ισχυρισμούς, καθώς η βελτίωση αυτή παρουσιάστηκε τόσο σε προγραμματιστές χωρίς εμπειρία (49% και 69%), όσο και σε προγραμματιστές με εμπειρία (39% και 54%).

Οι προβλέψεις της Gartner - μιας παγκόσμιας εταιρείας ερευνών με έδρα το Stamford, CT, ΗΠΑ - μιλούν για αύξηση κατά 23% στην παγκόσμια αγορά τεχνολογιών Ανάπτυξης Χαμηλού Κώδικα κατά το 2021 [20]. Επίσης στην ίδια αναφορά γίνεται γνωστό ότι η αυξημένη ζήτηση για προσαρμοσμένες λύσεις λογισμικού λόγω της ψηφιοποίησης των επιχειρήσεων έχει συντελέσει στην αύξηση των Πολιτών Προγραμματιστών (citizen developers) οι οποίοι καλούνται να υλοποιήσουν αυτές τις λύσεις. Η έρευνα της Gartner υποστηρίζει ότι κατά μέσο όρο το 41% των εργαζομένων εκτός των τμημάτων IT (Information Technology) προσαρμόζουν ή δημιουργούν λύσεις δεδομένων ή τεχνολογίας.

Μια ενδιαφέρουσα εφαρμογή της Ανάπτυξης Χαμηλού/Χωρίς Κώδικα, πραγματοποιήθηκε στο πλαίσιο ερευνών της Νέας Υόρκης και της Ουάσινγκτον για τα κρούσματα του COVID-19 [21]. Χρησιμοποιώντας μια πλατφόρμα Ανάπτυξης Λογισμικού Χαμηλού/Χωρίς κώδικα η οποία παρέχεται από την Unqork, μια νεοσύστατη εταιρεία με έδρα τη Νέα Υόρκη, δημιουργήθηκαν οι απαραίτητες εφαρμογές μέσα σε 3 ημέρες.

# Κεφάλαιο 4

## Υλοποίηση

Για την ικανοποίηση των στόχων της διπλωματικής και με γνώμονα τα κίνητρα της, σχεδιάστηκε η αρχιτεκτονική του συστήματος.

### 4.1 Αρχιτεκτονική συστήματος

Το αποτέλεσμα της σχεδίασης αυτής είναι μια πλατφόρμα, η οποία δίνει τη δυνατότητα στους χρήστες της να εγγράφονται, να αποκτούν πρόσβαση σε έναν προσωπικό Server Διαμεσολάβησης Μηνυμάτων, να δρομολογούν μηνύματα σε αυτόν, να εφαρμόζουν Φίλτρα παρακολούθησης των εισερχόμενων μηνυμάτων, να αποθηκεύουν τα μηνύματα ενδιαφέροντος σε μία ΒΔ και να βλέπουν την τρέχουσα κατάσταση των λειτουργιών που εκτελούνται στο σύστημά τους.

Όλες οι παραπάνω λειτουργίες μπορούν να πραγματοποιηθούν με ελάχιστες απαιτήσεις γνώσεων από τους χρήστες πάνω στις τεχνολογίες που χρησιμοποιήθηκαν. Επομένως το σύστημα κρίνεται κατάλληλο τόσο για έμπειρους στις χρησιμοποιούμενες τεχνολογίες χρήστες, όσο και για χρήστες που εμπίπτουν στην κατηγορία των Πολιτών Προγραμματιστών. Οι τελευταίοι έχουν τη δυνατότητα να χρησιμοποιήσουν με ευκολία τις δυνατότητες της τεχνολογίας Διαμεσολάβησης Μηνυμάτων, χωρίς να χρειαστεί να γράψουν κώδικα, σύμφωνα με τις αρχές της Ανάπτυξης Χαμηλού Κώδικα που χρησιμοποιήθηκαν για τη δημιουργία του συστήματος.

Στην πράξη ο μηχανισμός που έχει υλοποιηθεί, ορίζει ότι οι χρήστες εκκινούν λειτουργίες ή ζητούν συγκεκριμένες ενέργειες μέσω μιας φόρμας στη διεπαφή του χρήστη, οι οποίες προωθούνται και επεξεργάζονται από το Backend και εντέλει πυροδοτούν τις αντίστοιχες απαιτούμενες δράσεις στην υποδομή Νέφους στην οποία στεγάζεται το σύστημα.

**Διαμεσολάβηση Μηνυμάτων:** Η τεχνολογία που επιλέχθηκε για τον διαμεσολαβητή μηνυμάτων είναι το RabbitMQ. Αυτή η επιλογή έγινε επειδή το RabbitMQ είναι ένα έργο ανοιχτού κώδικα, ευρέως διαδεδομένο - άρα έχει μια μεγάλη ενεργή κοινότητα προς αντιμετώπιση προβλημάτων, υποστηρίζει διάφορα πρωτόκολλα, παρέχει πολλά εργαλεία σε μια μεγάλη ποικιλία γλωσσών προγραμματισμού και μπορεί να ικανοποιήσει απαιτήσεις μεγάλης κλίμακας και υψηλής διαθεσιμότητας.

**Περιβάλλον ανάπτυξης:** Για την ανάπτυξη όλου του απαραίτητου κώδικα της πλατφόρμας, χρησιμοποιήθηκε η γλώσσα προγραμματισμού Javascript στο περιβάλλον Node.js runtime environment. Η καταλληλότητα της Javascript για τον προγραμματισμό διεπαφών σε Web Browser και η ομοιογένεια που προσφέρει η χρήση μίας μοναδικής γλώσσας προγραμματισμού σε όλο το σύστημα, συντέλεσαν σε αυτήν την επιλογή.

**Ενορχήστρωση εφαρμογών:** Τέλος, ολόκληρο το σύστημα εντάχθηκε μέσα σε περιβάλλον Kubernetes. Έτσι, τα παραγόμενα επιμέρους κομμάτια του συστήματος μετατράπηκαν σε αυτόνομες containerized εφαρμογές. Η δομή της κάθε επιμέρους εφαρμογής και ο τρόπος με τον οποίο αυτή εντάσσεται στο K8s περιβάλλον περιγράφεται στο αντίστοιχο αρχείο τύπου yaml της εφαρμογής. Στο K8s ανατέθηκε η διαχείριση και ενορχήστρωση των containers αυτών, όπως επίσης και των επικοινωνιών μεταξύ τους.

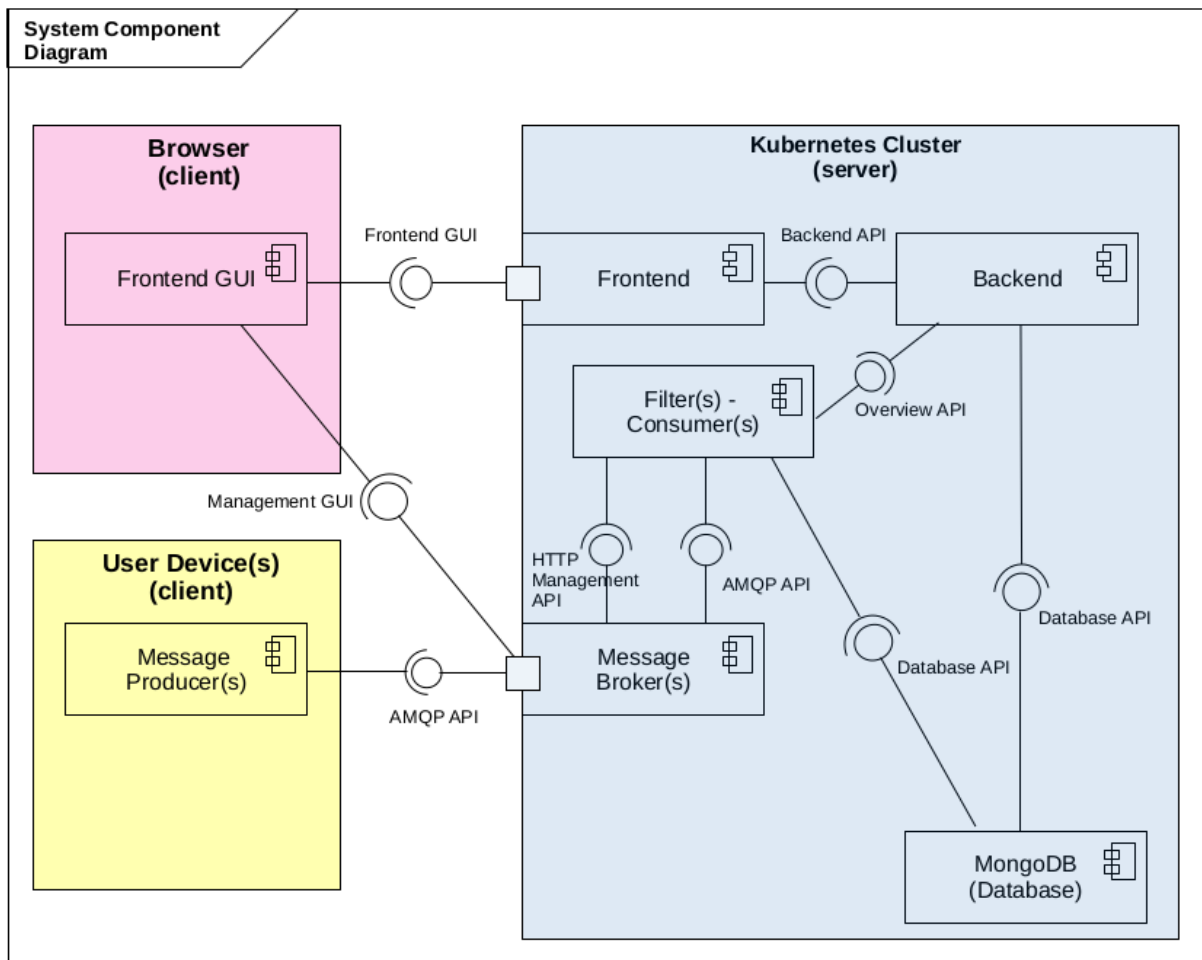
Η αρχιτεκτονική του συστήματος που θα παρουσιαστεί επομένως είναι μεταφέρσιμη και μπορεί να αναπτυχθεί σε οποιαδήποτε δομή που υποστηρίζει την ανάπτυξη K8s clusters, όπως είναι οι μεγαλύτερες εμπορικές υποδομές νέφους.

**Οντότητες της πλατφόρμας:** Για την υλοποίηση των προαναφερθέντων λειτουργιών, αναπτύχθηκαν οι παρακάτω λογικές οντότητες:

- Backend
- Frontend
- Βάση Δεδομένων
- Διαμεσολαβητής μηνυμάτων - RabbitMQ Server
- Φίλτρα - Καταναλωτές

Η τοπολογία του συστήματος και οι διασυνδέσεις μεταξύ αυτών των λογικών οντοτήτων παρουσιάζονται στο παρακάτω σχήμα 4.1. Η αναλυτική παρουσίαση της κάθε οντότητας πραγματοποιείται στην παράγραφο 4.3, ενώ οι σχέσεις μεταξύ τους αποτελούν το θέμα της παραγράφου 4.4.





Σχήμα 4.1: Παρουσίαση της αρχιτεκτονικής του συστήματος με τη χρήση Διαγράμματος Τμημάτων

## 4.2 Διάταξη της υλοποίησης και διαθέσιμοι πόροι

Λόγω διαθεσιμότητας πόρων, η ανάπτυξη του K8s cluster δεν πραγματοποιήθηκε σε κάποια cloud υποδομή. Αντίθετα χρησιμοποιήθηκε το εργαλείο Minikube το οποίο δημιουργεί ένα τοπικό K8s cluster ενός κόμβου στο μηχάνημα στο οποίο εκτελείται. Αυτό δεν αλλάζει τη δομή της υλοποίησης, παρά μόνο στον τρόπο παροχής πρόσβασης στο cluster από άλλα μηχανήματα όπως θα εξηγηθεί παρακάτω στις ενότητες 4.4.2, 4.4.4.

### 4.2.1 Διαθέσιμα μηχανήματα

**Κύριος υπολογιστής:** Ο κύριος υπολογιστής στον οποίο αναπτύχθηκε η πλατφόρμα προσδιορίζεται από τα παρακάτω τεχνικά χαρακτηριστικά:

- Λειτουργικό Σύστημα: Ubuntu 20.04.2 LTS 64bit
- Μνήμη RAM: 8GB (7.7GB διαθέσιμη) (2x4GB DDR3 1333 MHz (0,8 ns))
- Επεξεργαστής: Intel(R) Core(TM) i5-2450M CPU @ 2.50GHz
  - cores = 2

- threads = 4

**Δευτερεύοντα μηχανήματα:** Οι συσκευές παραγωγής μηνυμάτων προς το σύστημα προσομοιώθηκαν από δύο όμοια μηχανήματα με τα εξής τεχνικά χαρακτηριστικά:

- Λειτουργικό Σύστημα: Microsoft Windows 10 Home 64bit
- Μνήμη RAM: 8GB (5.9GB διαθέσιμη) @ 2400 MHz
- Επεξεργαστής: AMD Ryzen 5 3450U @ 2.10GHz
  - cores = 4
  - threads = 8

Το γεγονός ότι ο αριθμός των νημάτων που υποστηρίζουν τα μηχανήματα είναι μεγαλύτερος από τον αριθμό των πυρήνων οφείλεται στην τεχνολογία Hyper-Threading<sup>1</sup>.

#### 4.2.2 Εκχώρηση πόρων επεξεργασίας και μνήμης

Τη στιγμή της δημιουργίας ενός Pod μπορεί προαιρετικά να καθοριστεί το ποσό του κάθε πόρου που χρειάζεται το κάθε container εντός του Pod. Οι πιο συνηθισμένοι πόροι για καθορισμό είναι η CPU και η μνήμη (RAM)<sup>2</sup>.

Παρόλο που αυτή η λειτουργία είναι προαιρετική, συνίσταται να χρησιμοποιείται ώστε να αποφεύγονται περιπτώσεις εξάντλησης των πόρων ολόκληρου του συστήματος από την αλόγιστη χρήση πόρων από κάποιο container [22].

Τα όρια και τα αιτήματα για πόρους CPU μετρώνται σε μονάδες cpu. Ένας επεξεργαστής, στο K8s ισοδυναμεί με 1 vCPU/Core για παρόχους cloud και 1 υπερ-νήμα σε επεξεργαστές Intel. Επιτρέπονται τα κλασματικά αιτήματα. Η έκφραση 0.1 είναι ισοδύναμη με την έκφραση 100m, η οποία μπορεί να διαβαστεί ως "εκατό millicpu".

Τα όρια και τα αιτήματα για μνήμη μετρώνται σε byte. Η μνήμη μπορεί να εκφραστεί ως απλός ακέραιος ή ως αριθμός σταθερού σημείου χρησιμοποιώντας μία από αυτές τις καταλήξεις: E, P, T, G, M, k. Μπορεί επίσης να χρησιμοποιηθούν οι καταλήξεις που αντιπροσωπεύουν τις αντίστοιχες δυνάμεις του δύο: Ei, Pi, Ti, Gi, Mi, Ki.

**Διαθεσιμότητα πόρων στο Minikube:** Οι προκαθορισμένοι πόροι που ανατίθενται στο Minikube κατά την εκκίνηση του K8s cluster του είναι 2 CPUs και 2200 MB μνήμης. Στην υλοποίηση της παρούσας διπλωματικής το όριο της διαθέσιμης μνήμης αυξήθηκε στο τριπλάσιο, δηλαδή στα 6600 MB μνήμης, ενώ το όριο των διαθέσιμων CPUs αυξήθηκε σε 4 cpu.

<sup>1</sup><https://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html>

<sup>2</sup>Διαχείριση πόρων των container στο K8s: <https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/>

## 4.3 Παρουσίαση των οντοτήτων του συστήματος

Παρακάτω θα εξηγηθεί ο ρόλος της κάθε οντότητας, μαζί με τις λειτουργίες που υλοποιεί και τις λεπτομέρειες της υλοποίησής της.

### 4.3.1 Backend

Το Backend αποτελεί τον κύριο πυλώνα του συστήματος, καθώς είναι υπεύθυνο για τη διαχείριση όλων των δράσεων που πρέπει να πραγματοποιηθούν.

Η βασικότερή του λειτουργία είναι να δέχεται αιτήματα από τις άλλες εφαρμογές του συστήματος, να τα επεξεργάζεται, να εκτελεί τις απαιτούμενες ενέργειες και να στέλνει απαντήσεις στις εφαρμογές που τα δημιουργήσαν σχετικά με την εξέλιξη των αιτημάτων.

Η ανάπτυξη του Backend βασίστηκε στο Express.js<sup>3</sup> framework, το οποίο παρέχει ένα ισχυρό σύνολο δυνατοτήτων για εφαρμογές ιστού και εφαρμογές κινητών συσκευών. Μέσω αυτού διευκολύνεται η δρομολόγηση των αιτημάτων που δέχεται το Backend και η εφαρμογή εργαλείων και ενδιάμεσων λογισμικών (middlewares) στα αιτήματα πριν αρχίσει η κύρια επεξεργασία τους.

Το Backend προσφέρει μια Διεπαφή Προγραμματισμού Εφαρμογών - Application Programming Interface (API) για όλες τις λειτουργίες που μπορεί να πραγματοποιήσει.

Μέσω των Backend API, μπορούν να εξυπηρετηθούν αιτήματα σχετικά με τη δημιουργία / διαγραφή RMQ Servers, τη δημιουργία / διαγραφή φίλτρων - καταναλωτών μηνυμάτων, την αποστολή περιοδικών στατιστικών της λειτουργίας του συστήματος προς τη διεπαφή του χρήστη, τη σύνδεση / αποσύνδεση των χρηστών από το σύστημα, την εμφάνιση των καταγεγραμμένων μηνυμάτων από κάποιο Φίλτρο στη ΒΔ και την αυθεντικοποίηση των χρηστών.

**Εγγραφή:** Κατά την εγγραφή χρηστών στο σύστημα δημιουργείται ένα αίτημα από τη διεπαφή του χρήστη το οποίο περιλαμβάνει τα απαραίτητα στοιχεία εγγραφής. Το Backend αφού επιβεβαιώσει τη σωστή μορφή των στοιχείων αυτών, τα αποθηκεύει στη ΒΔ σε κατάλληλη δομή. (Όλες οι επικοινωνίες του Backend με τη ΒΔ γίνεται μέσω της βιβλιοθήκης Mongoose<sup>4</sup>.) Η λειτουργία αυτή παρουσιάζεται με τη μορφή σεναρίου χρήσης στην παράγραφο 5.1.

**Αυθεντικοποίηση:** Η αυθεντικοποίηση των χρηστών υλοποιήθηκε στο Backend με τη χρήση session cookies. Αυτή η μέθοδος λειτουργεί ως εξής:

1. Δημιουργείται ένα αίτημα αυθεντικοποίησης μέσω της διεπαφής του χρήστη το οποίο περιλαμβάνει τα διαπιστευτήριά του.
2. Το Backend δέχεται αυτό το αίτημα και ελέγχει αν στη ΒΔ υπάρχει αποθηκευμένος κάποιος χρήστης με τα συγκεκριμένα διαπιστευτήρια. (Η σύγκριση των

<sup>3</sup>Επίσημη ιστοσελίδα της βιβλιοθήκης Express.js: <https://expressjs.com/>

<sup>4</sup>Επίσημη ιστοσελίδα της βιβλιοθήκης Mongoose: <https://mongoosejs.com/>

κωδικών πρόσβασης γίνεται μέσω της βιβλιοθήκης `bcrypt.js`<sup>5</sup>.)

3. Εάν τα διαπιστευτήρια που δόθηκαν ήταν σωστά, τότε εισάγεται σε κατάλληλο πεδίο του session cookie ένα αναγνωριστικό του χρήστη. Σε αντίθετη περίπτωση το αίτημα απορρίπτεται.
4. Τέλος κάθε αίτημα εισερχόμενο στο Backend που απαιτεί αυθεντικοποίηση του χρήστη, πρώτα περνάει από μηχανισμό επιβεβαίωσης ότι το session cookie περιλαμβάνεται στο αίτημα και το αναγνωριστικό του χρήστη είναι έγκυρο. (Τα session cookies δημιουργούνται με χρήση της βιβλιοθήκης `node-client-sessions`<sup>6</sup>)

Το αντίστοιχο σενάριο χρήσης της παραπάνω λειτουργίας παρατίθεται στην παράγραφο 5.2.

**Διαχείριση RMQ Servers:** Για την επεξεργασία αιτημάτων δημιουργίας / διαγραφής RMQ Servers, επιστρατεύθηκε η βιβλιοθήκη `@kubernetes/client-node`<sup>7</sup>.

Όταν ένα τέτοιο αίτημα καταφτάνει στο Backend, επαληθεύονται τα στοιχεία του χρήστη και στη συνέχεια καλείται το K8s API μέσω του client που υλοποιεί η βιβλιοθήκη ώστε να αναπτυχθεί ένα νέο στιγμιότυπο της εφαρμογής του RMQ Server σε μορφή container στο περιβάλλον του K8s (ή να διαγραφεί ένα προϋπάρχον στιγμιότυπο στην αντίστοιχη περίπτωση).

Στη συνέχεια δημιουργούνται μέσω του ίδιου εργαλείου άλλα απαραίτητα στοιχεία για τη λειτουργία του διαμεσολαβητή μηνυμάτων (όπως είναι τα K8s Services που θα αναφερθούν παρακάτω) και τέλος ανανεώνεται η καταγραφή του χρήστη στη ΒΔ ώστε να περιλαμβάνει (ή να μην περιλαμβάνει σε περίπτωση διαγραφής) τα στοιχεία του διαμεσολαβητή.

Οι λειτουργίες αυτές αναλύονται περισσότερο στα σενάρια χρήσης των παραγράφων 5.4, 5.5.

**Διαχείριση Φίλτρων - Καταναλωτών μηνυμάτων:** Αντίστοιχα με την προηγούμενη λειτουργικότητα εξυπηρετούνται και τα αιτήματα δημιουργίας / διαγραφής φίλτρων - καταναλωτών μηνυμάτων. Επαληθεύονται τα στοιχεία του χρήστη και καλείται το ίδιο εργαλείο με σκοπό την ανάπτυξη ενός στιγμιότυπου της εφαρμογής Φίλτρου. Τέλος ενημερώνεται η καταγραφή του χρήστη στη ΒΔ για να περιλαμβάνει τα νέα δεδομένα.

Σενάρια χρήσης διαχείρισης Φίλτρων: 5.7, 5.8.

**Εμφάνιση μηνυμάτων Φίλτρου καταγεγραμμένα στη ΒΔ** Για την εξυπηρέτηση αυτών των αιτημάτων, το Backend εκτελεί τις απαραίτητες λειτουργίες αυθεντικοποίησης και στη συνέχεια δημιουργεί ένα αίτημα προς τη ΒΔ για να δεχτεί τα αποθηκευμένα μηνύματα που καταγράφηκαν από το συγκεκριμένο Φίλτρο για το οποίο ζητήθηκαν τα

<sup>5</sup>Σελίδα του `bcrypt.js` στην αποθήκη λογισμικών npm: <https://www.npmjs.com/package/bcryptjs>

<sup>6</sup>Σελίδα του πακέτου `client-sessions` στο github: <https://github.com/mozilla/node-client-sessions>.

<sup>7</sup>Σελίδα της βιβλιοθήκης `client-node` στην αποθήκη λογισμικών npm: <https://www.npmjs.com/package/@kubernetes/client-node>

δεδομένα. Αφού η ΒΔ απαντήσει με τα μηνύματα, το Backend τα προωθεί στη διεπαφή του χρήστη στο Frontend.

Η περιγραφή της λειτουργίας αυτής δίνεται στην παράγραφο [5.9](#) με τη μορφή σεναρίου χρήσης.

**Περιοδικά στατιστικά:** Το Backend μπορεί να προσφέρει την άμεση και περιοδική μετάδοση στατιστικών δεδομένων σχετικών με τη λειτουργία του συστήματος.

Για να γίνει αυτό, ξεκινάει μια επικοινωνία με ένα από τα στιγμιότυπα Φίλτρου - Καταναλωτή μηνυμάτων του χρήστη και του αναθέτει την ανάκτηση χρήσιμων δεδομένων από τον RMQ Server. Όταν το Φίλτρο - Καταναλωτής απαντήσει με τα δεδομένα, το Backend τα στέλνει πίσω στη διεπαφή του χρήστη.

Αυτή η ανάθεση γίνεται αφενός για την αποφυγή αποθήκευσης στο Backend των διαπιστευτηρίων του χρήστη που απαιτούνται για τη σύνδεση με τον RMQ Server και αφετέρου για την αποφόρτωση του Backend από πρόσθετη λειτουργικότητα.

Η περιοδική αποστολή των δεδομένων προς τη διεπαφή του χρήστη πραγματοποιείται με τη χρήση της μεθόδου Server Sent Events (SSE). Αυτή η μέθοδος περιγράφει μια ενσωματωμένη κλάση της Javascript μέσω της οποίας εγκαθιδρύεται μια σύνδεση ανάμεσα στη διεπαφή χρήστη και τον διακομιστή και επιτρέπεται η λήψη συμβάντων από αυτόν.

Αντίστοιχο σενάριο χρήσης: [5.10](#).

### **Βιβλιοθήκες / εργαλεία**

- express: 4.17.1 - Διαχείριση αιτημάτων API
- @kubernetes/client-node: 0.14.3 - Χρήση του K8s API
- bcryptjs: 2.4.3 - Κρυπτογράφηση διαπιστευτηρίων χρήστη
- client-sessions: 0.8.0 - Διαχείριση session cookies
- mongoose: 5.11.19 - Επικοινωνία με τη ΒΔ
- validator: 13.5.2 - Επαλήθευση σωστής μορφής εισαγόμενων δεδομένων από τον χρήστη
- cors: 2.8.5 - Επιτρέπει τη φόρτωση πόρων στο πρόγραμμα περιήγησης από προέλευση διαφορετική του διακομιστή του αιτήματος
- csurf: 1.11.0 - Προσφέρει προστασία ενάντια σε επιθέσεις τύπου Cross-Site Request Forgery (CSRF)
- helmet: 4.4.1 - Προσφέρει προστασία προσθέτοντας συγκεκριμένα HTTP Headers
- axios: 0.21.1 - Δημιουργία αιτημάτων HTTP

### **4.3.2 Frontend**

Ο ρόλος του Frontend είναι να υλοποιεί τη διεπαφή του χρήστη με το σύστημα (User Interface - UI). Αυτό σημαίνει την παρουσίαση ενός περιβάλλοντος φιλικού προς το

χρήστη, το οποίο να εμφανίζει με τρόπο κατανοητό τις πληροφορίες που του διατίθενται και να παρέχει εργαλεία αξιοποίησης όλων των λειτουργιών που προσφέρονται από το σύστημα.

Καθώς είναι το κομμάτι του συστήματος που εκτελείται στην πλευρά του χρήστη και συγκεκριμένα στον Browser του, δεν έχει απευθείας πρόσβαση σε καμία πληροφορία του συστήματος. Για οτιδήποτε χρειάζεται να παρουσιαστεί στον χρήστη, δημιουργείται ένα κατάλληλο αίτημα προς το Backend το οποίο αναλαμβάνει να του επιστρέψει τις αντίστοιχες πληροφορίες.

Η ανάπτυξη του Frontend βασίστηκε στη βιβλιοθήκη React<sup>8</sup>, η οποία διευκολύνει τη δημιουργία διαδραστικών διεπαφών χρήστη.

Οι δυνατότητες που προσφέρει το Frontend στον χρήστη είναι οι αντίστοιχες λειτουργίες που παρουσιάστηκαν στο Backend. Επιγραμματικά περιλαμβάνουν την εγγραφή και σύνδεση / αποσύνδεση χρηστών, τη δημιουργία / διαγραφή RMQ Servers, την πρόσβαση στο γραφικό περιβάλλον διαχείρισης των RMQ Servers, την εφαρμογή / διαγραφή Φίλτρων - Καταναλωτών μηνυμάτων, την εμφάνιση μηνυμάτων που έχουν καταγραφεί από κάποιο Φίλτρο στη ΒΔ και την παρουσίαση περιοδικών στατιστικών σχετικά με τη λειτουργία του συστήματος.

### **Δημιουργία / Διαγραφή διαμεσολαβητή μηνυμάτων**

Όπως αναφέρθηκε στην ανάλυση του Backend, κατά την εγγραφή του χρήστη στο σύστημα δημιουργείται ένας νέος RMQ Server για αυτόν. Ο χρήστης μπορεί από την καρτέλα Message Broker της διεπαφής να τον διαγράψει για διατήρηση πόρων σε περίπτωση που δεν επιθυμεί την διακίνηση μηνυμάτων άμεσα. Αφού προβεί στη διαγραφή του μπορεί πάλι να δημιουργήσει έναν άλλο RMQ Server οποιαδήποτε στιγμή από την ίδια καρτέλα.

Είναι σημαντικό να αναφερθεί ότι ο χρήστης δεν μπορεί να διαγράψει τον Διαμεσολαβητή Μηνυμάτων του αν υπάρχουν ενεργά Φίλτρα εκείνη τη στιγμή. Σε εκείνη την περίπτωση πρώτα πρέπει να διαγραφούν τα Φίλτρα και μετά ο Διαμεσολαβητής. Αυτό συμβαίνει γιατί αλλιώς τα ενεργά Φίλτρα θα προσπαθούσαν να συνδεθούν με έναν ανύπαρκτο Διαμεσολαβητή, κάτι το οποίο θα οδηγούσε σε άτακτη αποτυχία τους.

### **Εφαρμογή / Διαγραφή Φίλτρων - Καταναλωτών Μηνυμάτων**

Ο χρήστης μπορεί να δημιουργήσει ένα προσαρμόσιμο Φίλτρο της προτίμησής του, το οποίο θα δέχεται μηνύματα συγκεκριμένων θεμάτων και κάτω από συγκεκριμένες συνθήκες θα χαρακτηρίζει τα μηνύματα ως μηνύματα ενδιαφέροντος και θα τα αποθηκεύει στη ΒΔ.

Επομένως η διεπαφή χρήστη του επιτρέπει να ορίσει το όνομα της Ουράς στην οποία θα συνδέεται το Φίλτρο του. Ομοίως ορίζει το όνομα του Κόμβου Ανταλλαγής στο οποίο θα συνδεθεί η Ουρά αυτή. Προσδιορίζεται επίσης μέσω αντίστοιχου πεδίου το Κλειδί Σύνδεσης μεταξύ της ουράς και του exchange.

Στη συνέχεια ο χρήστης μπορεί να δώσει ένα πλήθος από συνθήκες, αν οποιαδήποτε από τις οποίες καλύπτεται από κάποιο μήνυμα, τότε αυτό κρίνεται ως σημαντικό και

---

<sup>8</sup>Επίσημη ιστοσελίδα της βιβλιοθήκης React: <https://reactjs.org/>

αποθηκεύεται από το Φίλτρο στη ΒΔ. Η κάθε συνθήκη αποτελείται από τρία ξεχωριστά μέρη :

α) το όνομα μιας μεταβλητής που πρέπει να βρίσκεται στα περιεχόμενα του μηνύματος

β) την τιμή της μεταβλητής αυτής και

γ) τον τελεστή που προσδιορίζει την σχέση που πρέπει να ισχύει μεταξύ της τιμής που δόθηκε από τον χρήστη και της τιμής που βρίσκεται στα περιεχόμενα του μηνύματος, έτσι ώστε αυτό να καταγραφεί στη ΒΔ. (Δυνατές τιμές του τελεστή είναι οι ακόλουθες: >, <, =, >=, <=, !=)

Πρακτικά ελέγχεται αν η τιμή της μεταβλητής είναι μικρότερη, μεγαλύτερη ή ίση με την τιμή ελέγχου που έχει δοθεί από τον χρήστη σε περίπτωση αριθμητικής μεταβλητής. Εάν η μεταβλητή αποτελείται από μια ακολουθία χαρακτήρων, η συνθήκη που εφαρμόζεται είναι ο έλεγχος της ταύτισης της ακολουθίας χαρακτήρων του μηνύματος με μια ακολουθία χαρακτήρων ελέγχου δοσμένη από τον χρήστη.

Κάθε ενεργό Φίλτρο - Καταναλωτής Μηνυμάτων μπορεί να διαγραφεί οποιαδήποτε στιγμή από την καρτέλα "Filters" της διεπαφής του χρήστη.

**Εμφάνιση μηνυμάτων Φίλτρου καταγεγραμμένα στη ΒΔ** Το Frontend παρέχει τη δυνατότητα στον χρήστη να ζητήσει την εμφάνιση των μηνυμάτων που έχουν καταγραφεί από κάποιο Φίλτρο στη ΒΔ.

Από την καρτέλα "Filters", ο χρήστης μπορεί να πατήσει το κουμπί "Show messages accepted by the Filter" (σχήμα 4.3) κάποιου ενεργού Φίλτρου και να εμφανιστεί ένα παράθυρο (σχήμα 5.14) στο οποίο παρουσιάζονται τα βασικά χαρακτηριστικά του Φίλτρου, μαζί με όλα τα μηνύματα τα οποία κάλυπταν τουλάχιστον μία από τις συνθήκες καταγραφής του Φίλτρου και αποθηκεύτηκαν στη ΒΔ.

### Παρουσίαση περιοδικών στατιστικών στοιχείων

Στην καρτέλα Overview δίνεται η δυνατότητα να εκκινηθεί ο μηχανισμός των Server Sent Events ώστε να παρουσιάζονται στατιστικά στοιχεία της πιο πρόσφατης εικόνας της διακίνησης μηνυμάτων στο σύστημα του χρήστη.

Το χρονικό παράθυρο για το οποίο θα αναζητηθούν τα στατιστικά είναι επιλέξιμο με τιμές από 1 λεπτό μέχρι και 24 ώρες. Επιλέξιμος είναι και ο χρόνος μεταξύ των δειγμάτων, με τιμές από 15 δευτερόλεπτα μέχρι και 1 ώρα<sup>9</sup>.

### Αξιοσημείωτα εργαλεία

**Redux:** Όσο αυξάνεται ο αριθμός των σελιδών και των κομματιών μιας διεπαφής χρήστη, τόσο μεγαλώνει και η πολυπλοκότητα διασύνδεσης όλων των μερών μεταξύ τους. Κάτι τέτοιο είναι ιδιαίτερα εμφανές όταν απαιτείται κάποιες πληροφορίες που παρήχθησαν σε ένα σημείο να παρουσιαστούν ή να χρησιμοποιηθούν κάπου αλλού.

<sup>9</sup>Μια σημαντική λεπτομέρεια είναι ότι για τη σωστή παραγωγή των στατιστικών στοιχείων, το χρονικό παράθυρο πρέπει να είναι τουλάχιστον 4 φορές μεγαλύτερο από το διάστημα δειγματοληψίας.



Το εργαλείο Redux λύνει αυτό ακριβώς το πρόβλημα. Καθιστά δυνατή τη διατήρηση σημαντικών δεδομένων σε ένα κεντρικό σημείο και προσφέρει μεθόδους για την πρόσβαση σε αυτά και την επεξεργασία τους.

**React-Router-Dom:** Η δρομολόγηση (Routing) είναι η ικανότητα εμφάνισης διαφορετικών σελίδων στον χρήστη. Αυτό σημαίνει ότι ο χρήστης μπορεί να μετακινηθεί μεταξύ διαφορετικών τμημάτων μιας εφαρμογής εισάγοντας μια διεύθυνση URL (Uniform Resource Locator) ή κάνοντας κλικ σε κάποιο στοιχείο.

Το React δεν παρέχει ενσωματωμένες δυνατότητες δρομολόγησης, κάτι το οποίο αντιμετωπίστηκε με τη χρήση της βιβλιοθήκης React-Router-Dom. Έτσι πήρε μορφή η τελική αρχιτεκτονική της διεπαφής του χρήστη, κατά την οποία η δρομολόγηση του χρήστη στις σελίδες που παρέχουν τα εργαλεία λειτουργικότητας, απαιτούν την αυθεντικοποίησή του με τη σύνδεσή του στο σύστημα.

### Βιβλιοθήκες / εργαλεία

- react: 17.0.1 - Δημιουργία διαδραστικών διεπαφών χρήστη
- react-dom: 17.0.1 - Πακέτο που λειτουργεί συμπληρωματικά με το React
- react-router-dom: 5.2.0 - Προσθήκη δυνατοτήτων δρομολόγησης
- redux: 4.0.5 - Κεντρική διαχείριση δεδομένων κατάστασης
- react-redux: 7.2.2 - Πακέτο που λειτουργεί συμπληρωματικά με το Redux
- redux-thunk: 2.3.0 - Πακέτο που λειτουργεί συμπληρωματικά με το Redux
- styled-components: 5.3.0 - Διευκόλυνση χρήσης Cascading Style Sheet (CSS) κώδικα σε γραφικά στοιχεία
- axios: 0.21.1 - Δημιουργία αιτημάτων HTTP
- classnames: 2.2.6 - Υπό όρους ένωση κλάσεων διαμόρφωσης στοιχείων

### 4.3.3 Βάση Δεδομένων

Η Βάση Δεδομένων είναι το σημείο που αποθηκεύονται όλα τα δεδομένα του συστήματος τα οποία πρέπει να διατηρούνται ανεξάρτητα από τη λειτουργία του.

Για την υλοποίησή της επιλέχθηκε η MongoDB<sup>10</sup>. Η MongoDB είναι μια ΒΔ εγγράφων που έχει σχεδιαστεί με κύριο στόχο την ευκολία ανάπτυξης και κλιμάκωσης.

Στην αρχιτεκτονική της πλατφόρμας που αναπτύχθηκε για αυτή τη διπλωματική, τα δεδομένα που αποθηκεύονται στη ΒΔ είναι δύο ειδών:

1. Δεδομένα χρηστών
2. Δεδομένα μηνυμάτων των χρηστών

---

<sup>10</sup>Επίσημη ιστοσελίδα του εγχειριδίου της MongoDB: <https://docs.mongodb.com/manual/>



## Δεδομένα χρηστών

Για κάθε χρήστη που εγγράφεται στο σύστημα, εισάγεται ένα νέο έγγραφο στη βάση δεδομένων "auth", στη συλλογή "users". Το κάθε έγγραφο περιλαμβάνει τα στοιχεία του χρήστη, στοιχεία σχετικά με τον προσωπικό του RabbitMQ Server αν υπάρχει, τα στοιχεία των ενεργών Φίλτρων - Καταναλωτών Μηνυμάτων αν υπάρχουν, και τέλος στατιστικά στοιχεία που περιγράφουν την πιο πρόσφατη εικόνα της διακίνησης μηνυμάτων μεταξύ του RMQ Server των Φίλτρων και της ΒΔ.

**Αυθεντικοποίηση:** Κατά τη διαδικασία της αυθεντικοποίησης, ο κωδικός που δίνει ο χρήστης στη σελίδα σύνδεσης του Frontend συγκρίνεται με το hash του κωδικού που είχε ορίσει κατά την εγγραφή του και το οποίο είναι αποθηκευμένο στη ΒΔ.

**Δημιουργία Φίλτρων - Καταναλωτών Μηνυμάτων:** Οι πληροφορίες που αποθηκεύονται για το κάθε ενεργό Φίλτρο του χρήστη είναι οι εξής:

- Το όνομά του
- Οι ετικέτες του
- Το όνομα του K8s Deployment του
- Το όνομα του K8s Namespace του
- Η χρονική στιγμή δημιουργίας του
- Το όνομα της Ουράς του RMQ Server στην οποία συνδέεται το Φίλτρο
- Το όνομα του Exchange το οποίο είναι συνδεδεμένο με την Ουρά
- Το Κλειδί Σύνδεσης της Ουράς με το Exchange
- Μία λίστα με τις συνθήκες κάτω από τις οποίες καταγράφεται κάποιο μήνυμα στη ΒΔ. Το κάθε στοιχείο αυτής της λίστας περιέχει:
  - Το όνομα μιας μεταβλητής που πρέπει να περιλαμβάνεται στα δεδομένα του μηνύματος
  - Τον τελεστή ο οποίος προσδιορίζει τη μαθηματική πράξη με βάση την οποία θα γίνει η σύγκριση της τιμής της μεταβλητής του μηνύματος και της τιμής που δόθηκε από τον χρήστη
  - Η τιμή με την οποία θα συγκριθεί το περιεχόμενο της μεταβλητής του μηνύματος

**Στατιστικά στοιχεία του συστήματος του χρήστη:** Κάθε φορά που ενεργοποιείται η παρουσίαση των στατιστικών του συστήματος από τη διεπαφή του χρήστη για κάποιο χρονικό παράθυρο, οι πληροφορίες συλλέγονται αποθηκεύονται στη ΒΔ.

Αυτές οι καταγραφές αποτελούνται από:

- Τον αριθμό των ενεργών Exchanges
- Τον αριθμό των ενεργών Ουρών
- Τον αριθμό των ενεργών Φίλτρων - Καταναλωτών

- Τον αριθμό των μηνυμάτων που εισάχθηκαν στον RMQ Server
- Τον αριθμό των μηνυμάτων που καταναλώθηκαν από τα Φίλτρα
- Τον αριθμό των μηνυμάτων που εισάχθηκαν στη ΒΔ
- Τον αριθμό των μηνυμάτων που απορρίφθηκαν από τα Exchanges ως μη-δρομολογήσιμα
- Τους ρυθμούς εισαγωγής, κατανάλωσης, αποθήκευσης και απόρριψης μηνυμάτων
- Τον τρέχον αριθμό μηνυμάτων στις Ουρές σε απόλυτο νούμερο και αριθμό bytes

### **Δεδομένα μηνυμάτων των χρηστών**

Κατά τη λειτουργία του συστήματος, όσα από τα μηνύματα που δρομολογούνται στα Φίλτρα, παρατηρείται ότι καλύπτουν τις συνθήκες που έχουν οριστεί από τον χρήστη, αποθηκεύονται στη ΒΔ.

Η καταγραφή κάθε τέτοιου μηνύματος, συνοδεύεται από κάποιες πρόσθετες πληροφορίες:

- Το όνομα του Φίλτρου που το επεξεργάστηκε
- Το όνομα του Exchange μέσω του οποίου εισάχθηκε στο σύστημα
- Το όνομα του Queue στο οποίο δρομολογήθηκε
- Τον σειριακό αριθμό επεξεργασίας του μηνύματος από το συγκεκριμένο Φίλτρο
- Το Κλειδί Σύνδεσης του Queue με το Exchange
- Το Κλειδί Δρομολόγησης του μηνύματος
- Τη χρονική στιγμή της δημιουργίας του μηνύματος
- Τη χρονική στιγμή της καταγραφής του μηνύματος στη ΒΔ
- Τη συνθήκη η οποία καλύφθηκε με αποτέλεσμα την καταγραφή του μηνύματος στη ΒΔ

### **4.3.4 Διαμεσολαβητής μηνυμάτων - RMQ Server**

Για την υλοποίηση του Διαμεσολαβητή μηνυμάτων επιλέχθηκε ο RMQ Server και συγκεκριμένα η εικόνα του container "rabbitmq:3-management-alpine".

Ο ρόλος του Διαμεσολαβητή είναι διττός. Η κύρια λειτουργία του είναι να υλοποιεί έναν μηχανισμό ο οποίος δέχεται μηνύματα από συσκευές εξωτερικές του συστήματος, τα αποθηκεύει προσωρινά και τα δρομολογεί κατάλληλα στις αντίστοιχες εφαρμογές προς κατανάλωση, βάσει προκαθορισμένων κανόνων.

Παράλληλα με τη διαχείριση μηνυμάτων, ο Διαμεσολαβητής που επιλέχθηκε περιλαμβάνει ένα plugin το οποίο παρέχει πρόσβαση σε υπηρεσίες παρουσίασης μιας λεπτομερούς εικόνας του συστήματος και της διακίνησης μηνυμάτων μέσα από αυτό. Ο χρήστης έχει τη δυνατότητα να εξερευνήσει μέσω ενός φιλικού γραφικού περιβάλλοντος διάφορες πτυχές του Διαμεσολαβητή και των εσωτερικών στοιχείων του, τα ενεργά

Φίλτρα που είναι συνδεδεμένα με αυτόν, ρυθμούς εισαγωγής και εξαγωγής μηνυμάτων από και προς το σύστημα και πολλά άλλα.

**Δρομολόγηση μηνυμάτων:** Η είσοδος του συστήματος είναι τα Exchanges του Διαμεσολαβητή. Εκείνα αναλαμβάνουν την ευθύνη να δρομολογήσουν τα εισερχόμενα μηνύματα στις Ουρές με τις οποίες είναι συνδεδεμένα. Μία Ουρά μπορεί να δημιουργηθεί όταν ο χρήστης εφαρμόσει ένα Φίλτρο - Καταναλωτή μηνυμάτων. Ο χρήστης ορίζει το όνομα της Ουράς από την οποία θα λαμβάνει μηνύματα το Φίλτρο και αν δεν υπάρχει ήδη Ουρά με αυτό το όνομα τότε αυτή δημιουργείται.

Ο χρήστης ορίζει επίσης με ποιο Exchange θέλει να συνδεθεί η Ουρά και παρέχει το κλειδί σύνδεσης της Ουράς με το Exchange. Αυτό σημαίνει ότι το Exchange δρομολογεί προς την Ουρά μόνο τα εισερχόμενα μηνύματα, το κλειδί δρομολόγησης των οποίων ταιριάζει με το κλειδί αυτής της σύνδεσης.

Υπάρχουν διαφορετικοί τύποι από Exchanges, τα οποία δρομολογούν μηνύματα με διαφορετικό τρόπο. Ο τύπος που χρησιμοποιήθηκε είναι τα Topic Exchanges καθώς προσφέρουν αυξημένη ευελιξία στη δρομολόγηση. Τα μηνύματα που εισέρχονται σε αυτόν τον τύπο από Exchanges πρέπει να έχουν κλειδιά δρομολόγησης συγκεκριμένης μορφής. Το κάθε κλειδί πρέπει να είναι ένα σύνολο από λέξεις χωρισμένες από τελείες. Ο αριθμός των λέξεων που μπορεί να χρησιμοποιηθεί περιορίζεται μόνο από το ανώτατο όριο των 255 bytes. Ένα παράδειγμα έγκυρου κλειδιού δρομολόγησης θα μπορούσε να είναι το "quick.orange.rabbit".

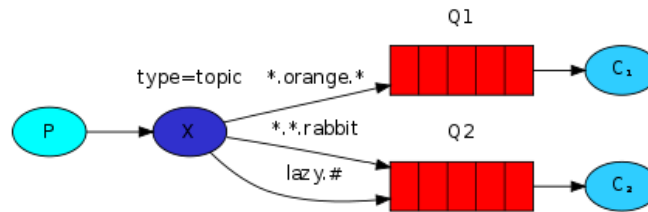
Τα κλειδιά σύνδεσης των Exchanges με τις Ουρές πρέπει να είναι της ίδιας μορφής. Για να προωθηθεί ένα μήνυμα από ένα Exchange σε μία Ουρά πρέπει το κλειδί σύνδεσής τους να ταιριάζει με το κλειδί δρομολόγησης του μηνύματος. Στα κλειδιά σύνδεσης μπορούν να χρησιμοποιηθούν δύο πρόσθετοι χαρακτήρες με ειδική σημασία:

- "\*" Ο αστερίσκος χρησιμοποιείται προς αντικατάσταση ακριβώς μίας λέξης
- "#" Η δέση χρησιμοποιείται προς αντικατάσταση καμίας ή περισσότερων λέξεων

Στο παράδειγμα του σχήματος 4.1 που ακολουθεί, στην Ουρά Q1 θα προωθηθούν όλα τα μηνύματα των οποίων το κλειδί δρομολόγησης αποτελείται από μια οποιαδήποτε λέξη, ακολουθούμενη από τη λέξη "orange", ακολουθούμενη από μια οποιαδήποτε λέξη.

Στην Ουρά Q2 θα προωθηθούν όλα τα μηνύματα των οποίων το κλειδί δρομολόγησης αποτελείται είτε από δύο οποιεσδήποτε λέξεις ακολουθούμενες από τη λέξη "rabbit", είτε από τη λέξη "lazy" ακολουθούμενη από αόριστο αριθμό από άλλες λέξεις.

Αναφορικά με το παράδειγμα που αναφέρθηκε, ένα μήνυμα με το κλειδί δρομολόγησης "quick.orange.rabbit" θα προωθούνταν και στις δύο ουρές στο παρακάτω σχήμα.



Σχήμα 4.2: Παράδειγμα δρομολόγησης μηνυμάτων μέσω Topic Exchanges

Ένα μήνυμα του οποίου το κλειδί δρομολόγησης δεν ταιριάζει με τη μορφή κανενός κλειδιού σύνδεσης, απορρίπτεται από το Exchange ως μη-δρομολογήσιμο.

Ένα μήνυμα του οποίου το κλειδί δρομολόγησης ταιριάζει με περισσότερα από ένα κλειδιά σύνδεσης, θα προωθηθεί σε όλες τις συνδέσεις αυτές. Στην περίπτωση που δύο ή περισσότερες συνδέσεις υπάρχουν μεταξύ ενός Exchange και μίας Ουράς και το κλειδί δρομολόγησης ενός μηνύματος ταιριάζει με περισσότερα από ένα κλειδιά των συνδέσεων αυτών, τότε το μήνυμα προωθείται μόνο μία φορά στη συγκεκριμένη ουρά.

**Υπηρεσίες παρακολούθησης και διαχείρισης του Διαμεσολαβητή:** Η πρόσβαση στο γραφικό περιβάλλον παρουσίασης του Διακομιστή επιτυγχάνεται από τον κάθε χρήστη με τα διαπιστευτήρια του λογαριασμού που δημιούργησε στην πλατφόρμα.

Σημαντικές δυνατότητες που προσφέρονται με την ενεργοποίηση του γραφικού περιβάλλοντος είναι οι ακόλουθες:

- Η δημιουργία, καταγραφή και διαγραφή στοιχείων του Διαμεσολαβητή όπως Exchanges, Queues και συνδέσεις
- Η παρακολούθηση του αριθμού των μηνυμάτων στις Ουρές, του ρυθμού εισαγωγής και εξαγωγής μηνυμάτων από κάθε στοιχείο του Διαμεσολαβητή και του ρυθμού χρήσης πόρων από τις συνδέσεις
- Παρακολούθηση χρήσης πόρων Διαμεσολαβητή: sockets και file descriptors, ανάλυση χρήσης μνήμης και διαθέσιμος χώρος στο δίσκο
- Αποστολή και λήψη μηνυμάτων (χρήσιμα σε περιβάλλοντα ανάπτυξης και για την αντιμετώπιση προβλημάτων)

Εκτός από τα παραπάνω θετικά σημεία, ο χρήστης πρέπει να λάβει υπόψιν του ότι η χρήση των υπηρεσιών αυτών ενέχει κάποια προβλήματα και περιορισμούς.

Συγκεκριμένα, το σύστημα παρακολούθησης μέρος του συστήματος που παρακολουθείται. Αυτό αρχικά σημαίνει πως εισάγονται στις μετρήσεις ορισμένες πρόσθετες καθυστερήσεις και αυξάνεται το φορτίο του συστήματος. Επομένως επιβαρύνεται η ακρίβεια των μετρήσεων.

Τα δεδομένα που αποθηκεύονται περιγράφουν την πιο πρόσφατη εικόνα του συστήματος μόνο και όχι τη συνολική λειτουργία του από την εκκίνησή του. Οι πληροφορίες οι οποίες παρουσιάζονται λοιπόν επεκτείνονται μέχρι και μερικές ώρες πριν την παρούσα στιγμή. Πληροφορίες για προηγούμενες ημέρες, εβδομάδες και μήνες δεν αποθηκεύονται.

Ο σχεδιασμός του εργαλείου δίνει έμφαση στην ευκολία χρήσης του και όχι στην βέλτιστη διαθεσιμότητα, επομένως μπορεί να παρατηρηθούν διακοπές και καθυστερήσεις στην παρουσίαση των δεδομένων.

#### 4.3.5 Φίλτρα / Καταναλωτές - Consumers

Τα Φίλτρα αποτελούν μονάδες εφαρμογών τις οποίες δημιουργεί ο κάθε χρήστης ανάλογα με τις προτιμήσεις του. Η λειτουργία τους εξαρτάται από παραμέτρους οι οποίες καθορίζονται από τον χρήστη κατά τη στιγμή της δημιουργίας τους. Στη συνέχεια αναπτύσσονται στο K8s περιβάλλον και επικοινωνούν με τις υπόλοιπες εφαρμογές ώστε να εκπληρώσουν τον σκοπό τους.

Ο σκοπός αυτός είναι η κατανάλωση και επεξεργασία μηνυμάτων από τον Διαμεσολαβητή. Αρχικά επιτυγχάνεται η σύνδεση με τον RMQ Server και δημιουργούνται τα απαραίτητα στοιχεία του Διαμεσολαβητή (Exchanges, Queues). Στη συνέχεια το Φίλτρο περιμένει την παράδοση μηνυμάτων από τον Διαμεσολαβητή όταν νέα μηνύματα οδηγηθούν στην Ουρά με την οποία είναι συνδεδεμένο. Τα περιεχόμενα των μηνυμάτων που θα παραδοθούν στο Φίλτρο ελέγχονται σε σχέση με τις συνθήκες καταγραφής που έχει ορίσει ο χρήστης. Στην περίπτωση που ικανοποιείται τουλάχιστον μία από τις συνθήκες αυτές, πραγματοποιείται μία σύνδεση με τη ΒΔ και το μήνυμα αποθηκεύεται στη συλλογή των μηνυμάτων στην προσωπική βάση δεδομένων του χρήστη.

Εκτός από τα παραπάνω, κάθε Φίλτρο έχει τη δυνατότητα συλλογής στατιστικών δεδομένων σχετικά με την πρόσφατη λειτουργία του Διαμεσολαβητή, χρησιμοποιώντας το HTTP API του RMQ Server. Αυτή η δυνατότητα χρησιμοποιείται όταν ενεργοποιηθεί η ζωντανή μετάδοση δεδομένων της λειτουργίας του συστήματος από τη διεπαφή του χρήστη.

**Παραδείγματα Φίλτρων:** Για να γίνει απόλυτα κατανοητός ο τρόπος λειτουργίας των Φίλτρων, παραθέτονται τα ακόλουθα παραδείγματα. Σε αυτά γίνεται λόγος τόσο για τη λειτουργία δρομολόγησης μηνυμάτων από τον Διαμεσολαβητή προς το Φίλτρο, όσο και για τον έλεγχο των συνθηκών καταγραφής μηνυμάτων στη ΒΔ.

Θεωρούμε τα Φίλτρα που χαρακτηρίζονται από τις παρακάτω παραμέτρους δρομολόγησης και καταγραφής μηνυμάτων:

- Φίλτρο 1:
  - Όνομα Κόμβου Ανταλλαγής: clima
  - Όνομα συνδεδεμένης Ουράς: clima
  - Κλειδί Σύνδεσης Exchange - Ουράς: #.clima.#
  - Συνθήκες καταγραφής μηνυμάτων:
    - \* temp > 25
    - \* humidity > 25
    - \* toxicity = critical
- Φίλτρο 2:
  - Όνομα Κόμβου Ανταλλαγής: department

- Όνομα συνδεδεμένης Ουράς: department
- Κλειδί Σύνδεσης Exchange - Ουράς: \*.department.#
- Συνθήκες καταγραφής μηνυμάτων:
  - \* department = human resources
- Φίλτρο 3:
  - Όνομα Κόμβου Ανταλλαγής: hospital
  - Όνομα συνδεδεμένης Ουράς: hospital
  - Κλειδί Σύνδεσης Exchange - Ουράς: illness.cost
  - Συνθήκες καταγραφής μηνυμάτων:
    - \* illness = serious
    - \* cost > 10000

Σύμφωνα με τα όσα αναφέρθηκαν στην προηγούμενη παράγραφο (4.3.4), το Φίλτρο 1 θα δεχτεί μηνύματα από την Ουρά "clima", τα οποία εισάχθηκαν στον Διαμεσολαβητή στο Exchange "clima", και τα οποία φέρουν Κλειδί Δρομολόγησης το οποίο περιλαμβάνει τη λέξη "clima". Το Φίλτρο 1 θα ελέγξει το περιεχόμενο των μηνυμάτων αυτών και θα καταγράψει στη ΒΔ όσα μηνύματα έχουν τιμή μεγαλύτερη του 25 στη μεταβλητή "temp", όσα έχουν τιμή μεγαλύτερη του 25 στη μεταβλητή "humidity" και όσα έχουν την τιμή "critical" στη μεταβλητή "toxicity".

Αντίστοιχα στο Φίλτρο 2 θα δρομολογηθούν μηνύματα του Κόμβου Ανταλλαγής "department" των οποίων το Κλειδί Δρομολόγησης αποτελείται από μια οποιαδήποτε λέξη, ακολουθούμενη από τη λέξη "department", ακολουθούμενη από 0 ή περισσότερες λέξεις. Από αυτά τα μηνύματα, το Φίλτρο 2 θα καταγράψει στη ΒΔ όσα μηνύματα περιέχουν την τιμή "human resources" στη μεταβλητή "department" στα περιεχόμενά τους.

Το Φίλτρο 3 θα δεχτεί μηνύματα του Κόμβου Ανταλλαγής "hospital" των οποίων το Κλειδί Δρομολόγησης αποτελείται αυστηρά από τις λέξεις "illness.cost" και μόνο. Όσα από αυτά τα μηνύματα περιέχουν την τιμή "serious" στη μεταβλητή "illness", ή τιμή μεγαλύτερη του 10000 στη μεταβλητή "cost", θα αποθηκευτούν στη ΒΔ.

Η ανάπτυξη των παραπάνω Φίλτρων φαίνεται υλοποιημένη στη γραφική διεπαφή του χρήστη στην εικόνα που ακολουθεί.

The screenshot shows the RabbitMQ Filters web interface. It has a top navigation bar with tabs: User Info, Overview, Message Broker, Filters (selected), and Test Producer. The main content area is divided into two columns. The left column is for creating a new filter, and the right column shows active filters.

**Apply a custom Filter:**

- Filter Name:
- Exchange Name:
- Queue Name:
- Binding Key:
- Log messages to the DB when the following condition is met:
  - Variable:
  - Operator:
  - Value:
- 
- Please provide your password, for the producer to be able to connect to the RabbitMQ server:
  - Password:
  -

**Active Filters:**

- Name: clima
  - Exchange name: clima
  - Queue name: clima
  - Binding key: #.clima.#
  - Logging conditions:
    - temp > 25
    - humidity > 25
    - toxicity = critical
  - Created at: 10/3/2021, 11:18:51 AM
  -
- Name: patients
  - Exchange name: hospital
  - Queue name: hospital
  - Binding key: illness.cost
  - Logging conditions:
    - illness = serious
    - cost > 10000
  - Created at: 10/3/2021, 12:14:26 PM
  -

Σχήμα 4.3: Παραδείγματα υλοποιημένων Φίλτρων με παραμέτρους δρομολόγησης και καταγραφής μηνυμάτων

Περισσότερες πληροφορίες και σενάρια χρήσης για τη λειτουργία των Φίλτρων δίνονται στην παράγραφο 5.7 της ενότητας “Παρουσίαση του Συστήματος”.

**Λεπτομέρειες της υλοποίησης:** Τόσο οι Ουρές, όσο και τα Exchanges που ορίζονται μέσω των Φίλτρων, παίρνουν τις ιδιότητες durable και autoDelete.

Η σημασία της πρώτης ιδιότητας είναι ότι τα στοιχεία αυτά θα διατηρηθούν ακόμα και μετά από μια επανεκκίνηση του Διαμεσολαβητή. Προφανώς κάτι τέτοιο απαιτείται, μιας και σε αυτά τα στοιχεία βασίζεται το μοντέλο της δρομολόγησης των μηνυμάτων.

Η δεύτερη ιδιότητα σημαίνει ότι όταν αποσυνδεθεί το τελευταίο Φίλτρο από μια Ουρά τότε η Ουρά θα διαγραφεί αυτόματα. Αντίστοιχα όταν αποσυνδεθεί η τελευταία Ουρά από ένα Exchange, τότε το Exchange θα διαγραφεί. Αυτό είναι σημαντικό ώστε να μη προκαλείται συνωστισμός αχρείαστων στοιχείων στον Διαμεσολαβητή.

Μια άλλη σημαντική λεπτομέρεια είναι ότι οι Ουρές ΔΕΝ ορίζονται ως αποκλειστικές (exclusive: false). Κατά αυτόν τον τρόπο, μπορούν να δημιουργηθούν πολλά Φίλτρα τα οποία να καταναλώνουν μηνύματα από την ίδια Ουρά όταν το φορτίο σε αυτήν αυξάνεται πέρα από τις δυνατότητες κατανάλωσης ενός Φίλτρου.

Επιπλέον πραγματοποιήθηκε δοκιμή χρήσης της οδηγίας channel.prefetch(1). Αυτή ορίζει ότι η Ουρά δε θα παραδίδει περισσότερα από ένα μηνύματα σε κάθε καταναλωτή κάθε στιγμή. Έτσι το κάθε επόμενο μήνυμα είτε παραδίδεται στον επόμενο ελεύθερο καταναλωτή, είτε γίνεται αναμονή της επεξεργασίας του προηγούμενου μηνύματος από τον πρώτο καταναλωτή πριν του παραδοθεί το επόμενο.

Η παραπάνω μέθοδος όμως αποδείχτηκε μέσω πειραμάτων ότι περιόριζε αρκετά την

απόδοση των Φίλτρων και την ταχύτητα κατανάλωσης μηνυμάτων από αυτά, επομένως και εγκαταλείφθηκε.

**Συλλογή δεδομένων πρόσφατης λειτουργίας του Διαμεσολαβητή:** Ο τρόπος με τον οποίο γίνεται χρήση αυτής της δυνατότητας είναι μέσω της βιβλιοθήκης `express.js`. Το Φίλτρο εκτός από Καταναλωτής μηνυμάτων, λειτουργεί παράλληλα και ως `server`. Όταν καταφτάσει ένα αίτημα από το Backend ζητώντας δεδομένα τότε το Φίλτρο το εξυπηρετεί.

Στο αίτημα περιλαμβάνονται οι παράμετροι που προσδιορίζουν τόσο το χρονικό παράθυρο για το οποίο ζητούνται δεδομένα, όσο και ο ρυθμός δειγματοληψίας. Το Φίλτρο συνδέεται στο HTTP API του `RMQ Server`, αποκτάει τα ζητούμενα δεδομένα και τα επιστρέφει στο Backend.

#### **Βιβλιοθήκες / εργαλεία:**

- `amqp-lib`: 0.8.0 - Επικοινωνία με τον `RMQ Server` και διαχείριση μηνυμάτων
- `express`: 4.17.1 - Διαχείριση αιτημάτων API
- `mongoose`: 5.13.3 - Επικοινωνία με τη ΒΔ
- `cors`: 2.8.5 - Επιτρέπει τη φόρτωση πόρων στο πρόγραμμα περιήγησης από προέλευση διαφορετική του διακομιστή του αιτήματος
- `node-fetch`: 2.6.1 - Δημιουργία αιτημάτων HTTP
- `body-parser`: 1.19.0 - Προσαρμογή των περιεχομένων ενός αιτήματος HTTP στο πεδίο `body` του αντικειμένου `req` του αιτήματος

## **4.4 Διασύνδεση οντοτήτων μέσω K8s**

Ο συνδυασμός όλων αυτών των οντοτήτων απαιτεί τον προσεκτικό καθορισμό των μέσων επικοινωνίας μεταξύ των `containers` τους. Ο τρόπος με τον οποίο αυτές οι υπηρεσίες επικοινωνίας γνωστοποιούνται και λαμβάνουν χώρα γίνεται κατανοητός στην ανάλυση που ακολουθεί.

Αφού παρουσιαστούν οι διασυνδέσεις μεταξύ των οντοτήτων, ακολουθεί η αναπαράστασή τους στο Διάγραμμα K8s Οντοτήτων [4.4](#).

### **4.4.1 Backend**

Το Backend καθώς θα πρέπει να επικοινωνεί με τη ΒΔ για τις λειτουργίες της αυθεντικοποίησης και της διαχείρισης των χρηστών, χρειάζεται να έχει γνώση της IP διεύθυνσης του `container` της ΒΔ. Προς ικανοποίηση αυτής της απαίτησης, δημιουργείται ένα K8s Service του τύπου `ClusterIP`, το οποίο παρέχει πρόσβαση στο Pod της ΒΔ. Η σύνδεση του Backend με αυτό το Service γίνεται μέσω του `yaml` εγγράφου περιγραφής του Backend. Σε αυτό το έγγραφο ορίζεται μία μεταβλητή περιβάλλοντος που έχει ως τιμή το DNS (Domain Name System) όνομα του Service που εκθέτει τη ΒΔ σε επικοινωνίες εντός του K8s cluster.



Επίσης καθώς μέσω του Backend θα πρέπει να δημιουργούνται και να διαγράφονται άλλα K8s components (όπως είναι τα Pods των Φίλτρων και τα Services που τα εκθέτουν), είναι απαραίτητο το Backend να έχει τα απαραίτητα δικαιώματα δημιουργίας και διαγραφής K8s αντικειμένων. Η διαδικασία απόδοσης δικαιωμάτων στο K8s περιγράφεται ως εξής:

Αρχικά δημιουργείται ένας λογαριασμός ServiceAccount για την εφαρμογή η οποία χρειάζεται τα δικαιώματα. Ο λογαριασμός αυτός πρέπει να συνδεθεί με κάποιον ρόλο δικαιωμάτων (K8s Role). Υπάρχουν συγκεκριμένοι ρόλοι που ομαδοποιούν λογικά συναφή δικαιώματα, όμως μπορεί να δημιουργηθεί και κάποιος προσαρμοσμένος ρόλος σε περίπτωση που τα δικαιώματα που απαιτούνται δεν καλύπτονται ή υπερκαλύπτονται από κάποιον έτοιμο. Η σύνδεση του ServiceAccount με τον ρόλο γίνεται μέσω ενός άλλου K8s αντικειμένου που λέγεται RoleBinding.

Αφού ολοκληρωθεί η παραπάνω διαδικασία και δημιουργηθούν τα απαραίτητα K8s αντικείμενα, τότε μπορεί να εισαχθεί το ServiceAccount στο yaml έγγραφο περιγραφής του Backend. Έτσι, όταν δημιουργηθεί ένα αίτημα από το Backend προς το σύστημα διαχείρισης του K8s, θα ελεγχθούν τα δικαιώματα που απαιτούνται για την εκτέλεσή του. Αν τα δικαιώματα αυτά περιλαμβάνονται στα δικαιώματα που είναι συνδεδεμένα με τον λογαριασμό του Backend, τότε θα επιτραπεί η εκτέλεση του αιτήματος από το K8s.

Στη συγκεκριμένη περίπτωση, τα δικαιώματα που αποδίδονται στο Backend είναι αυτά της καταγραφής, της δημιουργίας και της διαγραφής των K8s αντικειμένων Deployments, Services και Secrets.

Τέλος για να παρέχεται πρόσβαση στο Backend από άλλα Pods, δημιουργείται και ένα Service τύπου ClusterIP το οποίο εκθέτει το Backend εσωτερικά του K8s cluster.

#### 4.4.2 Frontend

**Nginx:** Τα παραγόμενα αρχεία της ανάπτυξης μιας εφαρμογής μέσω του React.js πρέπει να παραδίδονται στους χρήστης που ζητάνε πρόσβαση στην εφαρμογή μέσω αιτημάτων HTTP. Ταυτόχρονα, οι λειτουργίες του Frontend παράγουν HTTP αιτήματα τα οποία πρέπει να δρομολογηθούν στο Backend.

Τη διαχείριση και εξυπηρέτηση αυτών των αιτημάτων αναλαμβάνει ένας HTTP Proxy Server. Για την υλοποίηση αυτού του server επιλέχθηκε το Nginx. Λόγοι που οδήγησαν σε αυτή την επιλογή ήταν η ευρεία διάδοση του Nginx, η μακρά παρουσία του στον χώρο και η πληθώρα των λειτουργικών επιλογών που παρέχει. Το container του Frontend έτσι χτίστηκε πάνω στην Docker εικόνα "nginx:1.19-alpine".

Το Nginx συνεπώς εξυπηρετεί τις συνδέσεις των χρηστών προσφέροντάς τους τα αρχεία της εφαρμογής του Frontend. Παράλληλα διαχειρίζεται τα εσωτερικά παραγόμενα αιτήματα και προωθεί όσα από αυτά έχουν ως προορισμό το Backend στην κατάλληλη τοποθεσία.

**Frontend - Backend:** Για τη γνωστοποίηση της τοποθεσίας του Backend στο Frontend, ορίζεται μία μεταβλητή περιβάλλοντος στο yaml έγγραφο περιγραφής του Frontend. Η μεταβλητή αυτή έχει ως τιμή το DNS όνομα του Service που εκθέτει το

Backend σε επικοινωνίες εντός του K8s cluster.

**Εξωτερική πρόσβαση στο Frontend:** Το Frontend πρέπει να είναι προσβάσιμο από τοποθεσίες εκτός του K8s cluster ώστε να να επικοινωνούν με αυτό οι χρήστες μέσω των Browser τους. Αυτή η δυνατότητα πρόσβασης δίνεται με τη χρήση ενός K8s Service του τύπου LoadBalancer.

Στις τυπικές cloud υποδομές, παρέχονται υπηρεσίες LoadBalancers από τους διαχειριστές των cloud για αυτόν τον σκοπό. Οι LoadBalancers προσφέρουν μια δημόσια διεύθυνση IP προς χρήση, ενώ ταυτόχρονα εκτελούν λειτουργίες εξισορρόπησης φορτίου μεταξύ των στιγμιοτύπων των εφαρμογών τις οποίες εκθέτουν.

**Προσέγγιση παροχής εξωτερικής πρόσβασης στο Minikube:** Καθώς όμως επιλέχθηκε λόγω διαθεσιμότητας πόρων το Minikube για την ανάπτυξη του K8s cluster, αναλύεται παρακάτω η διαδικασία παροχής πρόσβασης σε μια εφαρμογή που βρίσκεται εντός του K8s cluster από τοποθεσία εκτός του cluster.

Στο περιβάλλον του Minikube ενώ υποστηρίζονται τα LoadBalancer Services, δεν έχουν την ίδια λειτουργικότητα. Αυτό που συμβαίνει αντίθετα είναι ότι το LoadBalancer Service εκτίθεται σε μία θύρα (Port) της διεύθυνσης IP του K8s Node που έχει δημιουργήσει το Minikube. Πρακτικά δηλαδή πρόκειται για ένα Service του τύπου NodePort. Η διαφοροποίηση είναι ότι το Minikube LoadBalancer Service μπορεί να λάβει μία επιπλέον εξωτερική διεύθυνση IP η οποία όμως θα είναι προσβάσιμη μόνο από το μηχάνημα στο οποίο έχει αναπτυχθεί το Minikube. Αυτό γίνεται με την εκτέλεση της εντολής "minikube tunnel".<sup>11</sup>

Για το λόγο αυτό, για να μπορέσει να έχει πρόσβαση κάποιος άλλος υπολογιστής του ίδιου δικτύου σε ένα LoadBalancer Service πρέπει να επικοινωνήσει με τη διεύθυνση <nodeIP>:<ServicePort>. Όμως η IP του K8s Node δεν είναι γνωστή στο router γιατί όπως αναφέρθηκε είναι host-only. Έτσι για να επιτευχθεί η επικοινωνία πρέπει να προστεθεί η πληροφορία στον δεύτερο υπολογιστή ότι τα αιτήματα με παραλήπτη το IP του Minikube θα πρέπει να έχουν ως gateway την τοπική διεύθυνση IP του μηχανήματος στο οποίο είναι ανεπτυγμένο το Minikube και να δρομολογηθούν εκεί.<sup>12</sup>

Αντίστοιχα στο πρώτο μηχάνημα πρέπει να επιτραπεί η προώθηση μηνυμάτων με τελικό παραλήπτη την τοπική διεύθυνση του K8s cluster, καθώς η προκαθορισμένη πολιτική είναι να απορρίπτονται όλα τα μηνύματα τα οποία δεν έχουν ως τελικό παραλήπτη την τοπική διεύθυνση του συγκεκριμένου μηχανήματος.<sup>13</sup>

#### 4.4.3 Βάση Δεδομένων

Κύρια λειτουργία της ΒΔ είναι η αποθήκευση και διατήρηση δεδομένων του συστήματος. Επομένως τα δεδομένα της πρέπει να αποθηκεύονται εκτός του εφήμερου συ-

<sup>11</sup>Η εντολή "minikube tunnel" λειτουργεί ως ένα process, δημιουργώντας ένα network route στο μηχάνημα (host) στο οποίο είναι ανεπτυγμένο το Minikube προς την υπηρεσία CIDR (Classless inter-domain routing) του K8s cluster χρησιμοποιώντας τη διεύθυνση IP του cluster ως gateway. Έτσι εκτίθεται κάθε εξωτερική IP απευθείας σε οποιοδήποτε πρόγραμμα εκτελείται στο λειτουργικό σύστημα του host.

<sup>12</sup>e.g.: route add 192.168.49.0 mask 255.255.255.0 192.168.1.154

<sup>13</sup>sudo sysctl net.ipv4.ip\_forward=1, sudo iptables -A FORWARD -j ACCEPT

στήματος αρχείων του container στο οποίο στεγάζεται. Αυτή η απαίτηση καλύπτεται με τη χρήση των Persistent Volumes που προσφέρει το K8s.

Πρώτα δημιουργείται ένα Persistent Volume (PV) και στη συνέχεια εκδίδεται μία αξίωση πάνω σε αυτόν τον τόμο αποθήκευσης μέσω ενός Persistent Volume Claim (PVC). Αυτή η αξίωση συνδέεται με το container της ΒΔ μέσω του yaml εγγράφου περιγραφής της ΒΔ.

Επίσης η ΒΔ αρχικοποιείται με ενεργοποιημένη την ελεγχόμενη πρόσβαση στις υπηρεσίες της μέσω αυθεντικοποίησης. Αυτό γίνεται με την εισαγωγή διαπιστευτηρίων επιπέδου διαχειριστή στην εφαρμογή της ΒΔ κατά τη δημιουργία της. Το έγγραφο περιγραφής της ΒΔ συνδέεται για αυτόν τον λόγο με τα περιεχόμενα ενός K8s Secret στο οποίο είναι αποθηκευμένα τα διαπιστευτήρια.

Για την παροχή πρόσβασης στην εφαρμογή της ΒΔ δημιουργείται ένα Service τύπου ClusterIP. Έτσι τα άλλα Pods εντός του K8s cluster μπορούν να επικοινωνούν με τη ΒΔ.

#### **4.4.4 Διαμεσολαβητής μηνυμάτων - RMQ Server**

Κατά την εγγραφή κάθε χρήστη, το Backend αναλαμβάνει να δημιουργήσει ένα Pod που υλοποιεί έναν RMQ Server μόνο για αυτόν τον χρήστη. Ταυτόχρονα αναπτύσσει δύο Services τα οποία προσφέρουν πρόσβαση στον Διαμεσολαβητή για διαφορετικούς λόγους το καθένα.

Δημιουργείται ένα ClusterIP Service που παρέχει πρόσβαση στο Port 5672 του Διαμεσολαβητή ώστε τα Φίλτρα να μπορούν να καταναλώνουν μηνύματα από αυτόν με χρήση του πρωτοκόλλου AMQP.

**Εξωτερική πρόσβαση στον Διαμεσολαβητή:** Επίσης αναπτύσσεται και ένα LoadBalancer Service που εκθέτει τα Ports 15672 και 5672 εκτός του K8s cluster. Το Port 15672 παρέχει πρόσβαση στον χρήστη στο γραφικό περιβάλλον διαχείρισης και εποπτείας του Διαμεσολαβητή, ενώ το Port 5672, δίνει τη δυνατότητα σε συσκευές ή εφαρμογές εκτός του cluster να παράγουν και να στέλνουν μηνύματα στον RMQ Server.

Όπως και στην περίπτωση του Frontend, η διαδικασία παροχής πρόσβασης σε αυτά τα Ports του Διαμεσολαβητή από τοποθεσία που βρίσκεται εκτός του cluster είναι η ίδια. Συνεπώς τα βήματα που ακολουθήθηκαν στην ενότητα του Frontend (4.4.2) επαρκούν και δεν απαιτείται κάποια πρόσθετη ενέργεια.

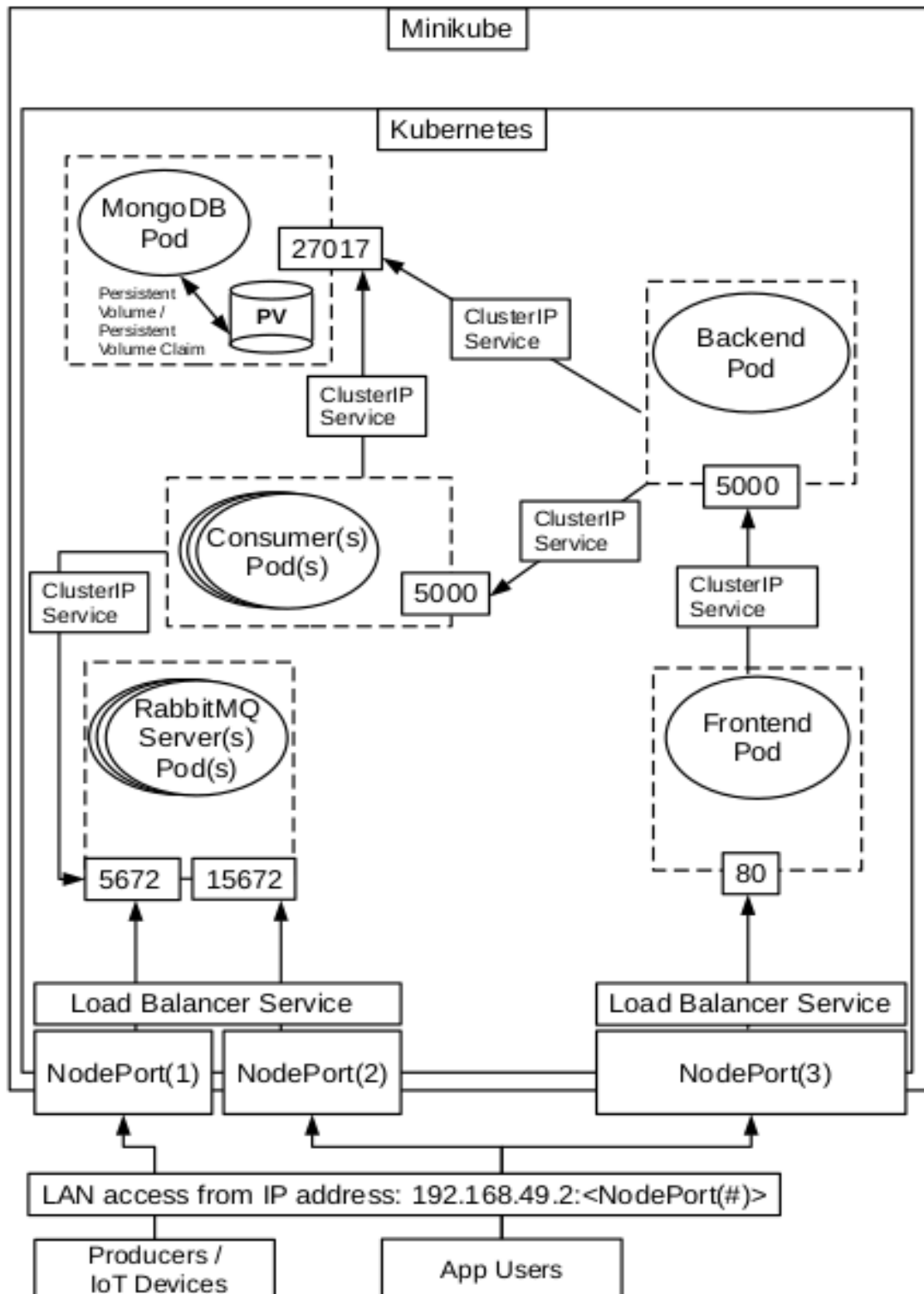
#### **4.4.5 Φίλτρα / Καταναλωτές - Consumers**

Η εκτέλεση των λειτουργιών του Φίλτρου απαιτεί πρόσβαση τόσο στον RMQ Server όσο και στη ΒΔ. Για αυτό δέχεται σαν μεταβλητές περιβάλλοντος τα διαπιστευτήρια του χρήστη, με τα οποία συνδέεται και στις δύο εφαρμογές που προαναφέρθηκαν.

Η τοποθεσία της ΒΔ και του RMQ Server εισάγονται επίσης με τη μορφή μεταβλητών περιβάλλοντος στο container του Φίλτρου κατά τη δημιουργία του από το Backend.

#### 4.4.6 Διαγραμματική αναπαράσταση των K8s οντοτήτων

Στο παρακάτω σχήμα συνοψίζονται οι διασυνδέσεις μεταξύ των K8s οντοτήτων, όπως αυτές μελετήθηκαν προηγουμένως.



Σχήμα 4.4: Διάγραμμα Οντοτήτων του K8s cluster σε συνδυασμό με τις εσωτερικές και εξωτερικές διασυνδέσεις του, όπως αναπτύχθηκε στα διαθέσιμα μηχανήματα.

## Κεφάλαιο 5

# Παρουσίαση του συστήματος - Σενάρια Χρήσης

Ακολουθεί η παρουσίαση των δυνατοτήτων που προσφέρει το σύστημα στους χρήστες μέσω αυτόνομων σεναρίων χρήσης. Η ανάλυση των σεναρίων χρήσης εμπλουτίζεται με την παρουσίαση των αντίστοιχων στιγμιотύπων της διεπαφής χρήστη (User Interface - UI) κατά την εκτέλεση του κάθε σεναρίου.

### 5.1 Εγγραφή χρήστη

Περιγράφεται η διαδικασία εγγραφής ενός νέου χρήστη στην πλατφόρμα :

**Όνομα:** Εγγραφή Χρήστη

**Χρήστης:** Χρήστης του συστήματος

**Στόχος:** Δημιουργία προσωπικού λογαριασμού του χρήστη

**Προϋποθέσεις:** Δεν υπάρχει συνδεδεμένος χρήστης στην πλατφόρμα κατά την εκκίνηση του σεναρίου χρήσης

**Περιγραφή Βασικής Ροής Εγγραφής Χρήστη :**

1. Ο χρήστης επιλέγει το κουμπί "Register" της γραμμής πλοήγησης στο πάνω μέρος της οθόνης.
2. Ο χρήστης συμπληρώνει τα στοιχεία του στη φόρμα εγγραφής που εμφανίζεται.
3. Ο χρήστης πατάει το κουμπί "Register User"
4. Το σύστημα εκτελεί έλεγχο των εισαγόμενων στοιχείων του χρήστη
5. Το σύστημα αποθηκεύει τα στοιχεία του χρήστη στη ΒΔ
6. Το σύστημα δημιουργεί έναν προσωπικό Διαμεσολαβητή Μηνυμάτων για τον χρήστη
7. Το σύστημα εμφανίζει στον χρήστη τη σελίδα Σύνδεσης Χρήστη (Σχήμα [5.3](#))

**Περιγραφή Εναλλακτικής Ροής Διαχείριση Εσφαλμένων Στοιχείων:**

1. Εάν στο βήμα 4. της Βασικής Ροής Εγγραφή Χρήστη το σύστημα αναγνωρίσει κάποιο από τα στοιχεία του χρήστη ως μη αποδεκτά (μη αποδεκτή μορφή ονόματος χρήστη, e-mail ή κωδικού πρόσβασης, αναντιστοιχία εισαχθέντων κωδικών πρόσβασης) το σύστημα εμφανίζει κατάλληλα μηνύματα λάθους κάτω από το αντίστοιχο πεδίο της φόρμας στο οποίο αναγνωρίστηκε το λάθος
2. Ο χρήστης διορθώνει το λάθος στοιχείο
3. Το σενάριο χρήσης συνεχίζει από το βήμα 3. της Βασικής Ροής Εγγραφή Χρήστη

**Παρουσίαση στιγμιотύπων της διεπαφής χρήστη για το συγκεκριμένο σενάριο χρήσης:**

Auth\_THMMY\_2020\_b-infrastructure [Register](#) [Login](#)

## Registration

[Register User](#)

Σχήμα 5.1: Φόρμα εγγραφής που περιγράφεται στο σενάριο χρήσης “Εγγραφή Χρήστη”

## Registration

❗

Email is invalid

❗

Password and Confirm Password must match

Σχήμα 5.2: Συμπληρωμένη φόρμα εγγραφής με αναγνωρισμένα σφάλματα που περιγράφεται στην εναλλακτική ροή “Διαχείριση Εσφαλμένων Στοιχείων” του σεναρίου χρήσης “Εγγραφή Χρήστη”

## 5.2 Σύνδεση χρήστη

Περιγράφεται η διαδικασία σύνδεσης ενός χρήστη στην πλατφόρμα:

**Όνομα:** Σύνδεση Χρήστη

**Χρήστης:** Χρήστης του συστήματος

**Στόχος:** Σύνδεση του χρήστη στην πλατφόρμα και πρόσβαση στις προστατευόμενες πληροφορίες του

**Προϋποθέσεις:**

- Δεν υπάρχει συνδεδεμένος χρήστης στην πλατφόρμα κατά την εκκίνηση του σεναρίου χρήσης
- Ο χρήστης έχει δημιουργήσει λογαριασμό στην πλατφόρμα (σενάριο χρήσης “Εγγραφή Χρήστη”) σε προηγούμενη χρονική στιγμή

**Περιγραφή Βασικής Ροής Σύνδεση Χρήστη:**

1. Ο χρήστης επιλέγει το κουμπί "Login" της γραμμής πλοήγησης στο πάνω μέρος της οθόνης.
2. Ο χρήστης συμπληρώνει τα στοιχεία του στη φόρμα σύνδεσης που εμφανίζεται.
3. Ο χρήστης πατάει το κουμπί "Login User"
4. Το σύστημα εκτελεί έλεγχο των εισαγόμενων στοιχείων του χρήστη
5. Το σύστημα δημιουργεί ένα session-cookie περιορισμένου χρόνου και το προσθέτει στην επικοινωνία του φυλλομετρητή με το σύστημα για την

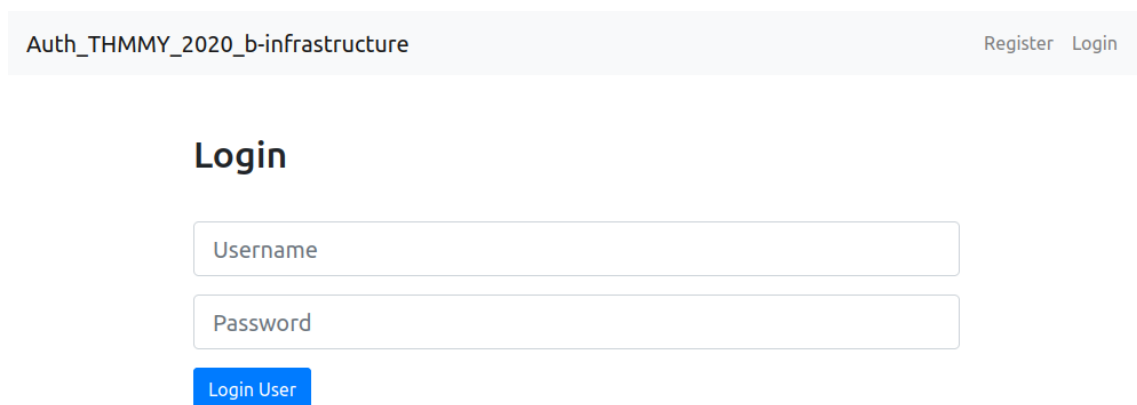
αυθεντικοποίηση των μελλοντικών αιτημάτων προς αυτό

6. Το σύστημα επιστρέφει στη διεπαφή χρήστη τις βασικές πληροφορίες του χρήστη που είναι αποθηκευμένες στη ΒΔ
7. Το σύστημα εμφανίζει στον χρήστη τη σελίδα προστατευμένου περιεχομένου ("Authenticated Content") (Σχήμα 5.4)

#### **Περιγραφή Εναλλακτικής Ροής Διαχείριση Εσφαλμένων Στοιχείων:**

1. Εάν στο βήμα 4. της Βασικής Ροής Σύνδεση Χρήστη το σύστημα δεν βρει αντίστοιχη εγγραφή στη ΒΔ για τα στοιχεία λογαριασμού που δόθηκαν από τον χρήστη, το σύστημα εμφανίζει κατάλληλα μηνύματα λάθους κάτω από το αντίστοιχο πεδίο της φόρμας στο οποίο αναγνωρίστηκε το λάθος
2. Ο χρήστης διορθώνει το λάθος στοιχείο
3. Το σενάριο χρήσης συνεχίζει από το βήμα 3. της Βασικής Ροής Σύνδεση Χρήστη

#### **Παρουσίαση στιγμιotypών της διεπαφής χρήστη για το συγκεκριμένο σενάριο χρήσης:**



Auth\_THMMY\_2020\_b-infrastructure [Register](#) [Login](#)

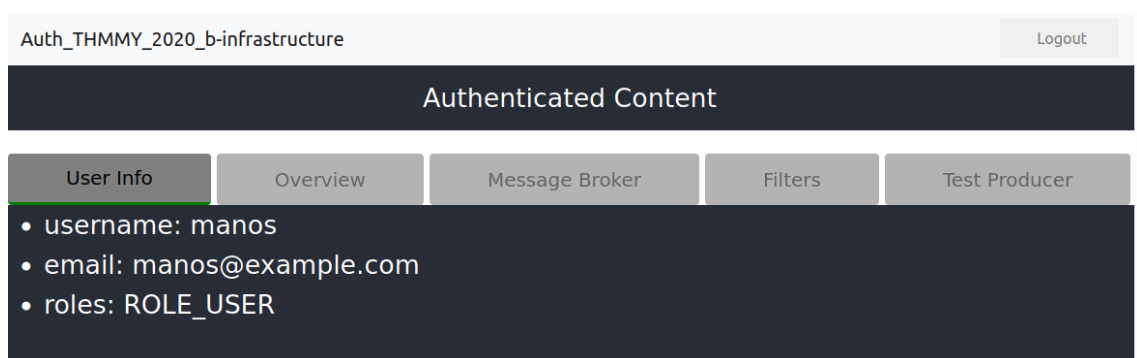
### Login

Username

Password

Login User

Σχήμα 5.3: Φόρμα σύνδεσης που περιγράφεται στο σενάριο χρήσης "Σύνδεση Χρήστη"



Auth\_THMMY\_2020\_b-infrastructure [Logout](#)

### Authenticated Content

User Info Overview Message Broker Filters Test Producer

- username: manos
- email: manos@example.com
- roles: ROLE\_USER

Σχήμα 5.4: Γραφικό περιβάλλον προστατευμένου περιεχομένου ("Authenticated Content") της πλατφόρμας. Περιγράφεται στο σενάριο χρήσης "Σύνδεση Χρήστη"



Auth\_THMMY\_2020\_b-infrastructureRegisterLogin

Login

ⓘ

User not found

Login User

Σχήμα 5.5: Αποτυχία εύρεσης του ονόματος χρήστη στη ΒΔ που περιγράφεται στην εναλλακτική ροή “Διαχείριση Εσφαλμένων Στοιχείων” του σεναρίου χρήσης “Σύνδεση Χρήστη”

Auth\_THMMY\_2020\_b-infrastructureRegisterLogin

Login

ⓘ

Incorrect Password

Login User

Σχήμα 5.6: Αναντιστοιχία δοσμένων διαπιστευτηρίων από τον χρήστη με την εγγραφή της ΒΔ που περιγράφεται στην εναλλακτική ροή “Διαχείριση Εσφαλμένων Στοιχείων” του σεναρίου χρήσης “Σύνδεση Χρήστη”

### 5.3 Αποσύνδεση χρήστη

Περιγράφεται η διαδικασία αποσύνδεσης ενός χρήστη στην πλατφόρμα :

**Όνομα:** Αποσύνδεση Χρήστη

**Χρήστης:** Χρήστης του συστήματος

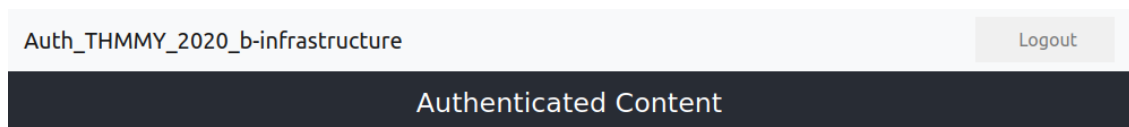
**Στόχος:** Αποσύνδεση του χρήστη από την πλατφόρμα

**Προϋποθέσεις:** Ο χρήστης έχει εκτελέσει τη διαδικασία σύνδεσης στην πλατφόρμα σε προηγούμενη χρονική στιγμή

### Περιγραφή Βασικής Ροής Αποσύνδεση Χρήστη :

1. Ο χρήστης επιλέγει το κουμπί "Logout" της γραμμής πλοήγησης στο πάνω μέρος της οθόνης.
2. Το σύστημα ακυρώνει την ισχύ του session-cookie και το διαγράφει από τον φυλλομετρητή του χρήστη
3. Το σύστημα εμφανίζει στον χρήστη τη σελίδα Σύνδεσης Χρήστη (Σχήμα 5.3)

### Παρουσίαση στιγμιοτύπων της διεπαφής χρήστη για το συγκεκριμένο σενάριο χρήσης :



Σχήμα 5.7: Στιγμιότυπο της μπάρας πλοήγησης με το κουμπί αποσύνδεσης. Περιγράφεται στο σενάριο χρήσης "Αποσύνδεση Χρήστη"

## 5.4 Δημιουργία RabbitMQ Server

Περιγράφεται η διαδικασία δημιουργίας ενός προσωπικού Διαμεσολαβητή Μηνυμάτων RabbitMQ Server για τον χρήστη :

**Όνομα :** Δημιουργία Διαμεσολαβητή Μηνυμάτων

**Χρήστης :** Χρήστης του συστήματος

**Στόχος :** Δημιουργία Διαμεσολαβητή Μηνυμάτων του χρήστη

#### Προϋποθέσεις :

- Ο χρήστης έχει εκτελέσει τη διαδικασία σύνδεσης στην πλατφόρμα σε προηγούμενη χρονική στιγμή
- Δεν υπάρχει οντότητα Διαμεσολαβητή Μηνυμάτων στο σύστημα για τον συγκεκριμένο χρήστη

### Περιγραφή Βασικής Ροής Δημιουργία Διαμεσολαβητή Μηνυμάτων :

1. Ο χρήστης πλοηγείται στην καρτέλα "Message Broker" του γραφικού περιβάλλοντος
2. Ο χρήστης εισάγει τον κωδικό του λογαριασμού του στο κενό πεδίο
3. Ο χρήστης πατάει το κουμπί "Create RabbitMQ Server"
4. Το σύστημα δημιουργεί μια οντότητα Διαμεσολαβητή Μηνυμάτων
5. Το σύστημα δημιουργεί τα απαραίτητα K8s στοιχεία για τη λειτουργία του Διαμεσολαβητή Μηνυμάτων (ClusterIP Service, LoadBalancer Service)
6. Το σύστημα αποθηκεύει τις πληροφορίες του Διαμεσολαβητή Μηνυμάτων στην εγγραφή του χρήστη στη ΒΔ

7. Το σύστημα επιστρέφει στη διεπαφή χρήστη τις πληροφορίες του Διαμεσολαβητή Μηνυμάτων
8. Η διεπαφή χρήστη παρουσιάζει τις πληροφορίες του Διαμεσολαβητή Μηνυμάτων στον χρήστη

**Παρουσίαση στιγμιοτύπων της διεπαφής χρήστη για το συγκεκριμένο σενάριο χρήσης:**

Auth\_THMMY\_2020\_b-infrastructure Logout

Authenticated Content

User Info Overview Message Broker Filters Test Producer

Request the creation of a personal Message Broker:  
Please provide your account password, for the filters  
to be able to connect to the Message Broker:

Password

Create RabbitMQ Server

Σχήμα 5.8: Γραφικό περιβάλλον της δημιουργίας Διαμεσολαβητή Μηνυμάτων. Περιγράφεται στο σενάριο χρήσης "Δημιουργία Διαμεσολαβητή Μηνυμάτων"

Auth\_THMMY\_2020\_b-infrastructure Logout

Authenticated Content

User Info Overview Message Broker Filters Test Producer

Existing Message Broker:  
Name: rabbitmq-manos  
Created on: Sat Sep 18 2021 17:45:53 GMT+0300 (Eastern European Summer Time)  
Message Broker can receive messages  
on the following address: 192.168.49.2:32352  
[Open RabbitMQ Management UI in a new Tab](#)  
Delete RabbitMQ Server

\*Message broker CANNOT be deleted without deleting all the active filters first.

Σχήμα 5.9: Στιγμιότυπο της καρτέλας "Message Broker" του γραφικού περιβάλλοντος με ενεργό Διαμεσολαβητή Μηνυμάτων. Περιγράφεται στο σενάριο χρήσης "Δημιουργία Διαμεσολαβητή Μηνυμάτων"

## 5.5 Διαγραφή RabbitMQ Server

Περιγράφεται η διαδικασία διαγραφής του προσωπικού Διαμεσολαβητή Μηνυμάτων RabbitMQ Server του χρήστη:

**Όνομα:** Διαγραφή Διαμεσολαβητή Μηνυμάτων

**Χρήστης:** Χρήστης του συστήματος

**Στόχος:** Διαγραφή Διαμεσολαβητή Μηνυμάτων του χρήστη

**Προϋποθέσεις:**

- Ο χρήστης έχει εκτελέσει τη διαδικασία σύνδεσης στην πλατφόρμα σε προηγούμενη χρονική στιγμή
- Υπάρχει οντότητα Διαμεσολαβητή Μηνυμάτων στο σύστημα για τον συγκεκριμένο χρήστη
- Δεν υπάρχουν ενεργά Φίλτρα - Καταναλωτές Μηνυμάτων στο σύστημα του χρήστη

**Περιγραφή Βασικής Ροής Διαγραφή Διαμεσολαβητή Μηνυμάτων:**

1. Ο χρήστης πλοηγείται στην καρτέλα "Message Broker" του γραφικού περιβάλλοντος (Σχήμα 5.9)
2. Ο χρήστης πατάει το κουμπί "Delete RabbitMQ Server"
3. Το σύστημα διαγράφει K8s στοιχεία που σχετίζονται με τη λειτουργία του Διαμεσολαβητή Μηνυμάτων (ClusterIP Service, LoadBalancer Service)
4. Το σύστημα διαγράφει τον Διαμεσολαβητή Μηνυμάτων του χρήστη
5. Το σύστημα διαγράφει τις πληροφορίες του Διαμεσολαβητή Μηνυμάτων από την εγγραφή του χρήστη στη ΒΔ
6. Το γραφικό περιβάλλον επιστρέφει στην κατάσταση του σχήματος 5.8

**Εξαίρεση Βασικής Ροής Διαγραφή Διαμεσολαβητή Μηνυμάτων:** Σε περίπτωση που υπάρχουν ενεργά Φίλτρα - Καταναλωτές Μηνυμάτων στο σύστημα του χρήστη, το κουμπί "Delete RabbitMQ Server" είναι απενεργοποιημένο και εμφανίζεται κατάλληλο μήνυμα στον χρήστη ότι η διαγραφή του Διαμεσολαβητή δεν επιτρέπεται μέχρι να διαγραφούν πρώτα όλα τα ενεργά Φίλτρα.

## 5.6 Χρήση του γραφικού περιβάλλοντος διαχείρισης του RabbitMQ Server

Περιγράφεται η διαδικασία χρήσης του γραφικού περιβάλλοντος διαχείρισης του RabbitMQ Server από τον χρήστη:

**Όνομα:** Πρόσβαση στο Γραφικό Περιβάλλον Διαχείρισης του RabbitMQ Server

**Χρήστης:** Χρήστης του συστήματος

**Στόχος:** Πρόσβαση στο Γραφικό Περιβάλλον Διαχείρισης του RabbitMQ Server από τον χρήστη

**Προϋποθέσεις:**

- Ο χρήστης έχει εκτελέσει τη διαδικασία σύνδεσης στην πλατφόρμα σε προηγούμενη χρονική στιγμή
- Υπάρχει οντότητα Διαμεσολαβητή Μηνυμάτων στο σύστημα για τον συγκεκριμένο χρήστη

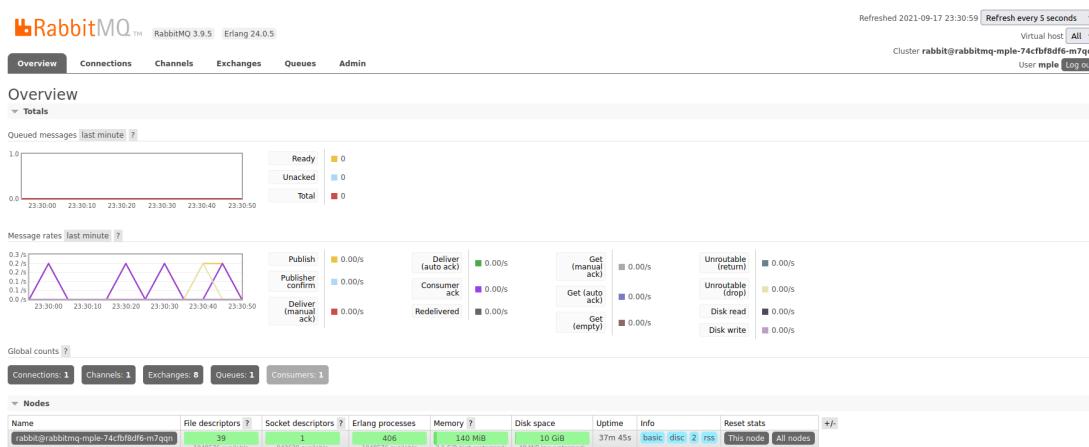
## Περιγραφή Βασικής Ροής Πρόσβαση στο Γραφικό Περιβάλλον Διαχείρισης του RabbitMQ Server:

1. Ο χρήστης πλοηγείται στην καρτέλα "Message Broker" του γραφικού περιβάλλοντος
2. Ο χρήστης πατάει πάνω στον σύνδεσμο "Open RabbitMQ Management UI in a new Tab" (Σχήμα 5.9)
3. Το σύστημα ανοίγει σε νέα καρτέλα το γραφικό περιβάλλον διαχείρισης του RabbitMQ Server
4. Ο χρήστης εισάγει τα στοιχεία του λογαριασμού του στη φόρμα σύνδεσης του RabbitMQ Management UI (Σχήμα 5.10)
5. Το γραφικό περιβάλλον διαχείρισης του RabbitMQ Server εμφανίζεται στον χρήστη (Σχήμα 5.11)

## Παρουσίαση στιγμιοτύπων της διεπαφής χρήστη για το συγκεκριμένο σενάριο χρήσης:



Σχήμα 5.10: Φόρμα σύνδεσης του RabbitMQ Management UI. Περιγράφεται στο σενάριο χρήσης "Πρόσβαση στο Γραφικό Περιβάλλον Διαχείρισης του RabbitMQ Server"



Σχήμα 5.11: Στιγμιότυπο του γραφικού περιβάλλοντος διαχείρισης του RabbitMQ Server. Περιγράφεται στο σενάριο χρήσης "Πρόσβαση στο Γραφικό Περιβάλλον Διαχείρισης του RabbitMQ Server"

## 5.7 Δημιουργία Φίλτρου - Καταναλωτή μηνυμάτων

Περιγράφεται η διαδικασία δημιουργίας ενός Φίλτρου - Καταναλωτή Μηνυμάτων στην πλατφόρμα :

**Όνομα:** Δημιουργία Φίλτρου

**Χρήστης:** Χρήστης του συστήματος

**Στόχος:** Δημιουργία παραμετροποιημένου Φίλτρου - Καταναλωτή Μηνυμάτων και εκκίνηση λειτουργίας του στο σύστημα του χρήστη

**Προϋποθέσεις:** Ο χρήστης έχει εκτελέσει τη διαδικασία σύνδεσης στην πλατφόρμα σε προηγούμενη χρονική στιγμή

### Περιγραφή Βασικής Ροής Δημιουργία Φίλτρου :

1. Ο χρήστης πλοηγείται στην καρτέλα "Filters" του γραφικού περιβάλλοντος
2. Ο χρήστης ορίζει τις επιθυμητές παραμέτρους του Φίλτρου (όνομα του Φίλτρου, όνομα του Exchange, όνομα της Ουράς, κλειδί σύνδεσης μεταξύ Exchange και Ουράς) στη φόρμα δημιουργίας Φίλτρου που εμφανίζεται
3. Ο χρήστης συμπληρώνει τα πεδία Variable, Operator, Value
4. Ο χρήστης καταγράφει τη συνθήκη καταγραφής μηνυμάτων πατώντας το κουμπί "Apply Logging Condition"
5. Τα βήματα 3. και 4. μπορούν να επαναληφθούν όσες φορές επιθυμεί ο χρήστης για την περίληψη όσων συνθηκών καταγραφής μηνυμάτων χρειάζονται
6. Ο χρήστης συμπληρώνει τον κωδικό του λογαριασμού του στο αντίστοιχο πεδίο
7. Ο χρήστης πατάει το κουμπί "Create Filter"
8. Το σύστημα εκτελεί έλεγχο των εισαγόμενων στοιχείων του χρήστη
9. Το σύστημα ελέγχει εάν υπάρχει ήδη ενεργό Φίλτρο με το ίδιο όνομα που δόθηκε από τον χρήστη
10. Το σύστημα δημιουργεί το παραμετροποιημένο Φίλτρο του χρήστη
11. Το σύστημα καταγράφει τις πληροφορίες του Φίλτρου στη ΒΔ στην εγγραφή του χρήστη
12. Το σύστημα επιστρέφει τις πληροφορίες του Φίλτρου στη διεπαφή του χρήστη
13. Το σύστημα εμφανίζει στον χρήστη τις πληροφορίες του Φίλτρου στην καρτέλα "Filters" (Σχήμα 5.3)

**Εξαίρεση Βασικής Ροής Δημιουργία Φίλτρου:** Σε περίπτωση που στο βήμα 9. υπάρχει ήδη ενεργό Φίλτρο με το ίδιο όνομα, τότε εμφανίζεται κατάλληλο μήνυμα σφάλματος στη διεπαφή του χρήστη (Σχήμα 5.13) και το σενάριο χρήσης συνεχίζει απ το βήμα 2.

**Παρουσίαση στιγμιotypών της διεπαφής χρήστη για το συγκεκριμένο σενάριο χρήσης:**

### Apply a custom Filter:

Log messages to the DB when the following condition is met:

▾

Apply Logging Condition

Please provide your password, for the producer to be able to connect to the RabbitMQ server:

Create Filter

Σχήμα 5.12: Φόρμα δημιουργίας παραμετροποιημένου Φίλτρου που περιγράφεται στο σενάριο χρήσης “Δημιουργία Φίλτρου”

Auth\_THMMY\_2020\_b-infrastructure Logout

Authenticated Content

User Info

Overview

Message Broker

Filters

Test Producer

Apply a custom Filter:

Filter Name ⓘ

Consumer name already exists

Exchange Name

Queue Name

Binding Key

Log messages to the DB when the following condition is met:

Variable

Operator

Value

Apply Logging Condition

Please provide your password, for the producer to be able to connect to the RabbitMQ server:

Password

Active Filters:

- Name: clima
  - Exchange name: clima
  - Queue name: clima
  - Binding key: #.clima.#
  - Logging conditions:
    - temp > 25
    - humidity > 25
    - toxicity = critical
  - Created at: 10/3/2021, 11:18:51 AM

Show messages accepted by the Filter

Delete Filter

Σχήμα 5.13: Μήνυμα σφάλματος στη φόρμα δημιουργίας Φίλτρου. Περιγράφεται στο σενάριο χρήσης "Δημιουργία Φίλτρου"

## 5.8 Διαγραφή Φίλτρου - Καταναλωτή μηνυμάτων

Περιγράφεται η διαδικασία διαγραφής ενός ενεργού Φίλτρου - Καταναλωτή Μηνυμάτων στην πλατφόρμα :

**Όνομα:** Διαγραφή Φίλτρου

**Χρήστης:** Χρήστης του συστήματος

**Στόχος:** Διαγραφή ενεργού Φίλτρου - Καταναλωτή Μηνυμάτων

**Προϋποθέσεις:**

- Ο χρήστης έχει εκτελέσει τη διαδικασία σύνδεσης στην πλατφόρμα σε προηγούμενη χρονική στιγμή
- Υπάρχει οντότητα Διαμεσολαβητή Μηνυμάτων στο σύστημα για τον συγκεκριμένο χρήστη
- Υπάρχει τουλάχιστον ένα ενεργό Φίλτρο - Καταναλωτής Μηνυμάτων στο σύστημα του χρήστη

**Περιγραφή Βασικής Ροής Διαγραφής Φίλτρου :**

1. Ο χρήστης πλοηγείται στην καρτέλα "Filters" του γραφικού περιβάλλοντος



2. Ο χρήστης πατάει το κουμπί "Delete Filter" που αντιστοιχεί στο Φίλτρο που επιθυμεί να διαγράψει όπως φαίνεται στο σχήμα 5.13
3. Το σύστημα διαγράφει το Φίλτρο του χρήστη
4. Το σύστημα διαγράφει τις πληροφορίες του Φίλτρου από την εγγραφή του χρήστη στη ΒΔ
5. Οι πληροφορίες του συγκεκριμένου Φίλτρου εξαφανίζονται από το γραφικό περιβάλλον της καρτέλας "Filters"

## 5.9 Εμφάνιση καταγεγραμμένων μηνυμάτων στη ΒΔ από Φίλτρο

Περιγράφεται η διαδικασία εμφάνισης των μηνυμάτων που κάλυπταν τις συνθήκες καταγραφής κάποιου Φίλτρου και καταγράφηκαν στη ΒΔ.

**Όνομα:** Εμφάνιση καταγεγραμμένων μηνυμάτων στη ΒΔ από Φίλτρο

**Χρήστης:** Χρήστης του συστήματος

**Στόχος:** Εμφάνιση όλων των μηνυμάτων τα οποία δρομολογήθηκαν σε συγκεκριμένο Φίλτρο και κατέληξαν να αποθηκευτούν στη ΒΔ καθώς κάλυπταν μία ή περισσότερες από τις συνθήκες καταγραφής μηνυμάτων του Φίλτρου.

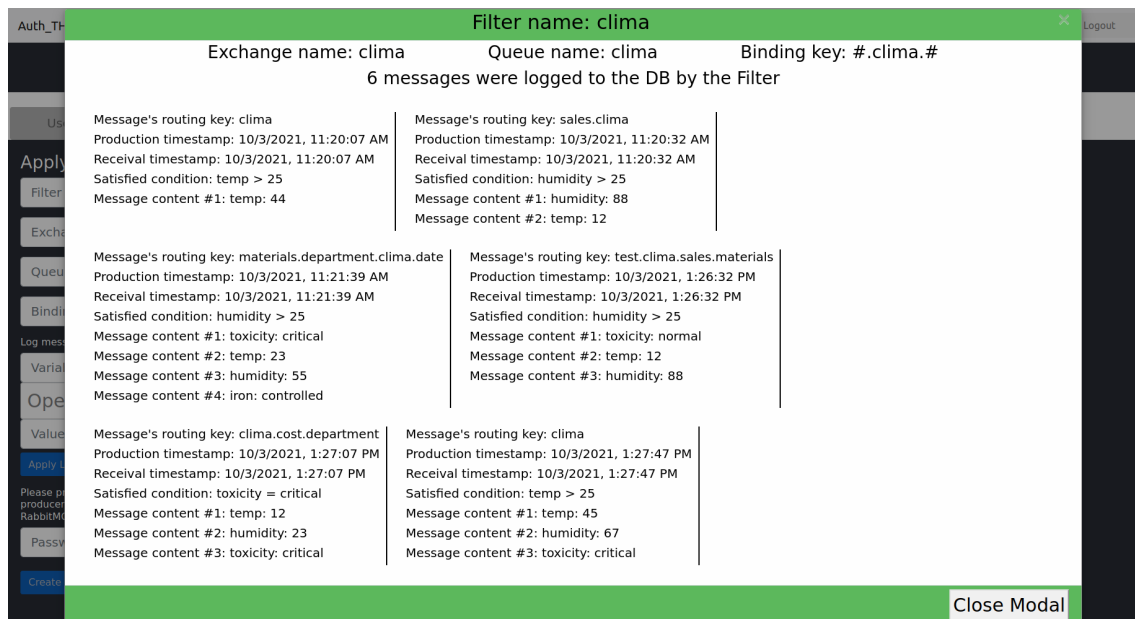
**Προϋποθέσεις:** Ο χρήστης έχει εκτελέσει τη διαδικασία σύνδεσης στην πλατφόρμα σε προηγούμενη χρονική στιγμή

**Περιγραφή Βασικής Ροής Εμφάνιση καταγεγραμμένων μηνυμάτων στη ΒΔ από Φίλτρο:**

1. Ο χρήστης πλοηγείται στην καρτέλα "Filters" του γραφικού περιβάλλοντος
2. Ο χρήστης πατάει το κουμπί "Show messages accepted by the Filter" που αντιστοιχεί σε κάποιο συγκεκριμένο Φίλτρο.
3. Το σύστημα δημιουργεί ένα αίτημα προς τη ΒΔ ζητώντας όλα τα μηνύματα που έχουν καταγραφεί από το συγκεκριμένο Φίλτρο.
4. Η ΒΔ επιστρέφει τα ζητούμενα μηνύματα στο σύστημα
5. Το σύστημα επιστρέφει τα ζητούμενα μηνύματα στη διεπαφή του χρήστη
6. Το σύστημα εμφανίζει στον χρήστη τα μηνύματα σε κατάλληλο αναδυόμενο παράθυρο (Σχήμα 5.14)

**Εξαίρεση Βασικής Ροής Εμφάνιση καταγεγραμμένων μηνυμάτων στη ΒΔ από Φίλτρο:** Σε περίπτωση που στο βήμα 4. δεν υπάρχουν μηνύματα αποθηκευμένα στη ΒΔ που να έχουν καταγραφεί από το συγκεκριμένο Φίλτρο, τότε εμφανίζεται κατάλληλο μήνυμα έλλειψης μηνυμάτων στο αναδυόμενο παράθυρο στη διεπαφή του χρήστη και το σενάριο χρήσης τερματίζει.

**Παρουσίαση στιγμιotypών της διεπαφής χρήστη για το συγκεκριμένο σενάριο χρήσης:**



Σχήμα 5.14: Παράθυρο μηνυμάτων Φίλτρου που κάλυπταν τις συνθήκες καταγραφής του Φίλτρου και αποθηκεύτηκαν στη ΒΔ. Περιγράφεται στο σενάριο χρήσης "Εμφάνιση καταγεγραμμένων μηνυμάτων στη ΒΔ από Φίλτρο"

## 5.10 Ενεργοποίηση ροής στατιστικών στοιχείων του συστήματος

Περιγράφεται η διαδικασία ενεργοποίησης της ζωντανής μετάδοσης στατιστικών στοιχείων για τη λειτουργία του συστήματος προς τον χρήστη :

**Όνομα:** Ενεργοποίηση Μετάδοσης Μετρήσεων Συστήματος

**Χρήστης:** Χρήστης του συστήματος

**Στόχος:** Ενεργοποίησης της ζωντανής μετάδοσης στατιστικών στοιχείων προς τον χρήστη

**Προϋποθέσεις:**

- Ο χρήστης έχει εκτελέσει τη διαδικασία σύνδεσης στην πλατφόρμα σε προηγούμενη χρονική στιγμή
- Υπάρχει οντότητα Διαμεσολαβητή Μηνυμάτων στο σύστημα για τον συγκεκριμένο χρήστη
- Υπάρχει τουλάχιστον ένα ενεργό Φίλτρο στο σύστημα του χρήστη

**Περιγραφή Βασικής Ροής Ενεργοποίηση Μετάδοσης Μετρήσεων Συστήματος:**

1. Ο χρήστης πλοηγείται στην καρτέλα "Overview" του γραφικού περιβάλλοντος
2. Ο χρήστης πατάει το κουμπί "Start data stream" (Σχήμα 5.15)
3. Το σύστημα εκτελεί μετρήσεις στα δεδομένα της ΒΔ για το προκαθορισμένο χρονικό παράθυρο 2 λεπτών με συχνότητα δειγματοληψίας 15 δευτερολέπτων

4. Το σύστημα καλεί το API του Διαμεσολαβητή Μηνυμάτων ζητώντας του τα αντίστοιχα δεδομένα
5. Το σύστημα επιστρέφει τα δεδομένα μετρήσεων που έχει συλλέξει στη διεπαφή του χρήστη
6. Τα βήματα 3., 4., 5. επαναλαμβάνονται με περιοδικότητα 5 δευτερολέπων μέχρι ο χρήστης να πατήσει το κουμπί "Stop data stream" (Σχήμα 5.16)

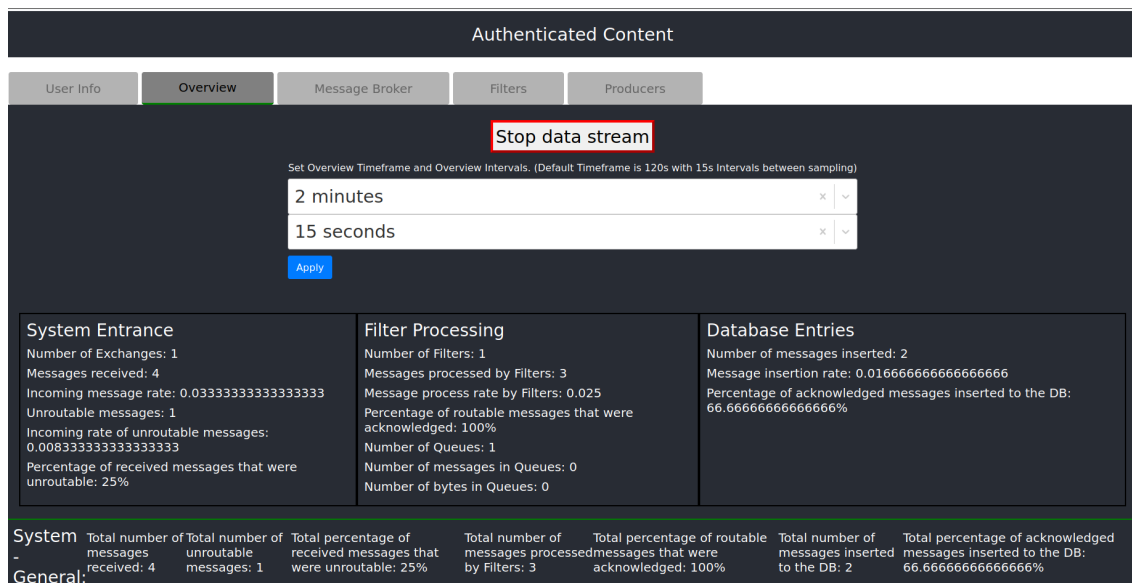
#### **Περιγραφή Εναλλακτικής Ροής Ορισμός Χρονικών Παραμέτρων Μετρήσεων:**

1. Αντί για το βήμα 2. της Βασικής Ροής Ενεργοποίηση Μετάδοσης Μετρήσεων Συστήματος ο χρήστης ορίζει τιμές για το μέγεθος του χρονικού παραθύρου που θα αφορούν οι μετρήσεις και για τη συχνότητα δειγματοληψίας
2. Ο χρήστης πατάει το κουμπί "Apply" (Σχήμα 5.16)
3. Το σενάριο χρήσης συνεχίζει με τις τιμές που έδωσε ο χρήστης από το βήμα 3. της Βασικής Ροής Ενεργοποίηση Μετάδοσης Μετρήσεων Συστήματος

#### **Παρουσίαση στιγμιotypών της διεπαφής χρήστη για το συγκεκριμένο σενάριο χρήσης:**

The screenshot displays the 'Overview' tab of a web application. At the top, the user is logged in as 'Auth\_THMMY\_2020\_b-infrastructure' with a 'Logout' button. Below the header, there are five tabs: 'User Info', 'Overview' (selected), 'Message Broker', 'Filters', and 'Producers'. The main content area features a 'Start data stream' button. Below this, a message states: 'Set Overview Timeframe and Overview Intervals. (Default Timeframe is 120s with 15s Intervals between sampling)'. There are two input fields: 'Set Timeframe of overview data to be gathered' and 'Set Intervals period between samples', both with dropdown arrows. An 'Apply' button is located below these fields. At the bottom of the main content area, there are three buttons: 'System Entrance', 'Filter Processing', and 'Database Entries'. The footer of the page reads 'System - General:'.

Σχήμα 5.15: Γραφικό περιβάλλον της καρτέλας "Overview" με ανενεργή τη μετάδοση μετρήσεων. Περιγράφεται στο σενάριο χρήσης "Ενεργοποίηση Μετάδοσης Μετρήσεων Συστήματος"



Σχήμα 5.16: Γραφικό περιβάλλον της καρτέλας "Overview" με ενεργή τη μετάδοση μετρήσεων και καθορισμένες τιμές χρονισμού των μετρήσεων. Περιγράφεται στο σενάριο χρήσης "Ενεργοποίηση Μετάδοσης Μετρήσεων Συστήματος"

## Κεφάλαιο 6

### Πειράματα & Αποτελέσματα

Τα όρια των δυνατοτήτων του συστήματος διερευνήθηκαν με την πραγματοποίηση δύο κύκλων πειραμάτων. Τα βασικά μεγέθη που μετρήθηκαν κατά τη διάρκεια εκτέλεσης των πειραμάτων ήταν τα εξής:

- Η συχνότητα εισαγωγής μηνυμάτων στο σύστημα. Δηλαδή η συχνότητα άφιξης μηνυμάτων στα Exchanges του RMQ Server
- Η συχνότητα κατανάλωσης μηνυμάτων από τα ενεργά Φίλτρα
- Η συχνότητα καταγραφής μηνυμάτων στη ΒΔ
- Ο αριθμός μηνυμάτων που ήταν αποθηκευμένος στις Ουρές του Διαμεσολαβητή κάθε στιγμή

Η συχνότητα εισαγωγής μηνυμάτων θεωρείται ότι είναι η είσοδος του συστήματος, ενώ οι συχνότητες κατανάλωσης μηνυμάτων, καταγραφής στη ΒΔ και ο αριθμός μηνυμάτων στις Ουρές περιγράφουν την απόκριση ή εξόδους του συστήματος.

Στον πρώτο κύκλο πειραμάτων ερευνήθηκε η απόκριση του συστήματος για αύξουσες και κατόπιν φθίνουσες τιμές της συχνότητας εισόδου του συστήματος με δεδομένο αριθμό ενεργών Φίλτρων.

Στον δεύτερο κύκλο πειραμάτων που αποτελεί την αντιστροφή του πρώτου, ερευνήθηκε η απόκριση του συστήματος με αυξανόμενους αριθμούς ενεργών Φίλτρων για κάθε δεδομένη συχνότητα εισόδου του συστήματος.

Οι μετρήσεις των μεγεθών του RabbitMQ Server πραγματοποιήθηκαν καλώντας το HTTP management API του RMQ Server και ζητώντας τις τιμές των μεγεθών για συγκεκριμένες στιγμές.

Οι μετρήσεις των μεγεθών της ΒΔ έγιναν ελέγχοντας τους ρυθμούς καταγραφής μηνυμάτων που αντιστοιχούσαν στις χρονικές στιγμές δειγματοληψίας των μετρήσεων του RabbitMQ Server.

Στα παρακάτω πειράματα παρουσιάζονται με τη σειρά τα εξής διαγράμματα:

1. Διάγραμμα αποθηκευμένων μηνυμάτων σε Ουρές
2. Διάγραμμα συχνότητας άφιξης μηνυμάτων - κατανάλωσης μηνυμάτων - καταγραφής μηνυμάτων στη ΒΔ

Στα διαγράμματα αποθηκευμένων μηνυμάτων σε Ουρές, με κόκκινο χρώμα παρουσιάζεται ο συνολικός αριθμός μηνυμάτων σε Ουρές, με γαλάζιο χρώμα ο αριθμός των μηνυμάτων που έχουν αποσταλεί σε καταναλωτές όμως δεν έχει καταγραφεί η επιβεβαίωση λήψης τους και με κίτρινο χρώμα ο αριθμός των μηνυμάτων που είναι έτοιμα προς επεξεργασία αλλά δεν έχουν παραδοθεί σε Καταναλωτές ακόμα.

Στα διαγράμματα συχνοτήτων άφιξης μηνυμάτων - κατανάλωσης μηνυμάτων - καταγραφής μηνυμάτων στη ΒΔ, με κόκκινο χρώμα παρουσιάζεται ο ρυθμός εισαγωγής μηνυμάτων στα Exchanges, με κίτρινο χρώμα ο ρυθμός επιβεβαίωσης κατανάλωσης μηνυμάτων από τα Φίλτρα - Καταναλωτές και με μπλε χρώμα ο ρυθμός εισαγωγής μηνυμάτων στη ΒΔ.

## **6.1 Σταθερές παράμετροι των πειραμάτων**

Για να μπορεί να γίνει σύγκριση και ερμηνεία των αποτελεσμάτων των πειραμάτων ήταν κρίσιμο αυτά να πραγματοποιηθούν σε ένα κοινό πλαίσιο. Αυτό σημαίνει πως θα έπρεπε οι συνθήκες εκτέλεσής τους να είναι ίδιες για όλα τα πειράματα και αλλαγές να γινόντουσαν μόνο στις μεταβλητές ελέγχου που ήταν η συχνότητα εισαγωγής μηνυμάτων και ο αριθμός των ενεργών Φίλτρων.

Σύμφωνα με τα παραπάνω έγινε η επιλογή των ακόλουθων παραμέτρων οι οποίες διατηρήθηκαν σταθερές σε όλα τα πειράματα :

- Τα μηνύματα είχαν όλα σταθερό μέγεθος 49 bytes
- Το ποσοστό των μηνυμάτων που ικανοποιούσαν τις δοσμένες από τον χρήστη συνθήκες και καταγράφονταν στη ΒΔ ήταν σταθερά 100%
- Το ποσοστό των μηνυμάτων των οποίων το κλειδί δρομολόγησης αντιστοιχούσε σε κάποιο κλειδί σύνδεσης Exchange - Ουράς ώστε αυτά να δρομολογούνται σε κάποιο Φίλτρο ήταν σταθερά 100%
- Η συχνότητα δειγματοληψίας για την καταγραφή των μεγεθών παρέμεινε σταθερή στο ένα δείγμα / 15 δευτερόλεπτα
- Όλα τα Φίλτρα - Καταναλωτές συνδέονται στην ίδια Ουρά και δέχονται μηνύματα από αυτήν

## **6.2 Ακρίβεια μετρήσεων**

Καθώς το όλο σύστημα είναι ασύγχρονο και οι μετρήσεις των μεγεθών συμβαίνουν ταυτόχρονα με τη λειτουργία, δεν εγγυώνται απόλυτα ακριβή αποτελέσματα. Σύμφωνα με αναλύσεις ανθρώπων που εργάζονται στην ανάπτυξη του RabbitMQ [23] ακόμα και σε περιπτώσεις συστημάτων με έναν Παραγωγό μηνυμάτων, έναν Καταναλωτή και μια Ουρά, ο συνολικός αριθμός εισερχόμενων μηνυμάτων μπορεί να μην ταυτίζεται με τον αριθμό των εξερχόμενων μηνυμάτων. Σε αυτό συνεισφέρει και το γεγονός ότι αν γίνουν επαναποστολές μηνυμάτων, αυτές προστίθενται στον αριθμό των εξερχόμενων μηνυμάτων, οδηγώντας σε διαφορετικούς ρυθμούς εισόδου και εξόδου.

Επίσης καθώς το σύστημα που παρακολουθούσε τα ζητούμενα μεγέθη ταυτίζεται με το σύστημα υπό παρακολούθηση, είναι λογικό να εισάγεται θόρυβος στην ακρίβεια

των μετρήσεων λόγω της καταγραφής των μεγεθών. Κάτι ακόμα που προσμετράται επηρεάζοντας την ακρίβεια των δεδομένων είναι το γεγονός ότι τα συστήματα διανομής και ανταλλαγής μηνυμάτων είναι εγγενώς ταυτόχρονα και ασταθή. Καθώς πραγματοποιείται η μέτρηση ενός μεγέθους, συμβαίνουν άλλες λειτουργίες ταυτόχρονα οι οποίες μπορεί να μην συντονίζονται με τη διαδικασία μετρήσεων.

Παρόλα αυτά οι μετρήσεις σε αρκετά μεγάλο χρονικό διάστημα θα δώσουν μια αρκετά καλή προσέγγιση. Με γνώμονα τα παραπάνω ερμηνεύτηκαν τα αποτελέσματα των πειραμάτων που πραγματοποιήθηκαν.

## 6.3 Πρώτος κύκλος πειραμάτων

Ο πρώτος κύκλος πειραμάτων αφορά την καταγραφή της απόκρισης του συστήματος για διάφορες συχνότητες εισόδου με δεδομένο αριθμό ενεργών Φίλτρων - Καταναλωτών Μηνυμάτων.

Η μέθοδος εκτέλεσης των πειραμάτων του πρώτου κύκλου ήταν η εξής:

Σε κάθε πείραμα αναπτυσσόταν ο καθορισμένος αριθμός Καταναλωτών και στη συνέχεια άρχιζε η αποστολή μηνυμάτων προς το σύστημα. Ο ρυθμός αποστολής μηνυμάτων ξεκινούσε από 0 μηνύματα το δευτερόλεπτο και κάθε 15 δευτερόλεπτα αυξανόταν κατά 60 μηνύματα το δευτερόλεπτο μέχρι να φτάσει τα 600. Στη συνέχεια ακολουθούσε η αντίστροφη μείωση του ρυθμού αποστολής μηνυμάτων με τους ίδιους χρόνους και το ίδιο βήμα μείωσης.

Το πείραμα επαναλήφθηκε για κάθε τιμή του αριθμού Φίλτρων - Καταναλωτών μηνυμάτων στο εύρος [1 έως 10]. Για συντομία θα παρουσιαστούν τα πειράματα με 1,2,4,6,8 και 10 Καταναλωτές.

### 6.3.1 1 Καταναλωτής | Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο

Στο Σχήμα 6.2 μπορεί να παρατηρηθεί από το διάγραμμα συχνοτήτων ότι ο Καταναλωτής ακολουθεί πιστά και διαχειρίζεται χωρίς πρόβλημα το φορτίο των εισερχόμενων μηνυμάτων μέχρι ο ρυθμός άφιξής τους να υπερβεί τα 540 μηνύματα / δευτερόλεπτο περίπου. Σε εκείνο το σημείο ο Καταναλωτής αρχίζει να μη μπορεί να διαχειριστεί το φορτίο των εισερχόμενων μηνυμάτων με αποτέλεσμα να συνωστίζονται αυτά στην Ουρά.

Κατά τη μείωση του ρυθμού εισαγωγής μηνυμάτων παρατηρείται ότι μόλις αυτός περάσει το όριο των 400 μηνυμάτων / δευτερόλεπτο, ο Καταναλωτής διαχειρίζεται και πάλι ικανοποιητικά τα εισερχόμενα μηνύματα.

Μια άλλη πληροφορία που δίνει το διάγραμμα αποθηκευμένων μηνυμάτων στις Ουρές είναι ότι τα μηνύματα που συνωστίζονται στις Ουρές είναι τα μηνύματα που έχουν παραδοθεί στον Καταναλωτή και για τα οποία δεν έχει καταφτάσει η Επιβεβαίωση Λήψης τους στον Διαμεσολαβητή. Η γραφική παράσταση των μηνυμάτων χωρίς Επιβεβαίωση Λήψης (γαλάζιο χρώμα) ταυτίζεται από τη γραφική παράσταση του συνολικού αριθμού μηνυμάτων στις Ουρές (κόκκινο χρώμα), για αυτό και δεν είναι εμφανής στο σχήμα καθώς κρύβεται πίσω από την τελευταία.

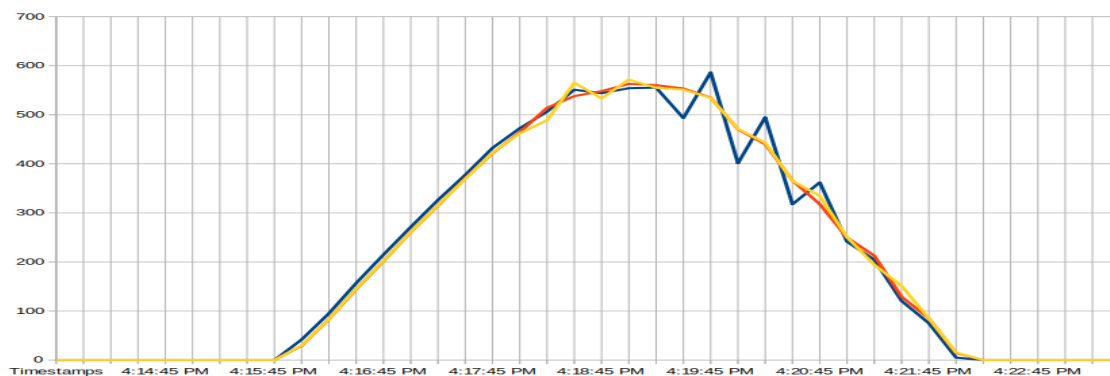
Τέλος, παρατηρούμε ότι η δυσκολία που έχει ο Καταναλωτής να διαχειριστεί το σύνολο των μηνυμάτων από τον ρυθμό των 500 μηνυμάτων / δευτερόλεπτο και πάνω αντικατοπτρίζεται και στον ρυθμό καταγραφών στη ΒΔ.



Σχήμα 6.1: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)

Μηνύματα έτοιμα προς παράδοση (κίτρινο)

Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)



Σχήμα 6.2: Ρυθμός άφιξης μηνυμάτων (κόκκινο)

Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)

Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

### 6.3.2 2 Καταναλωτές | Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο

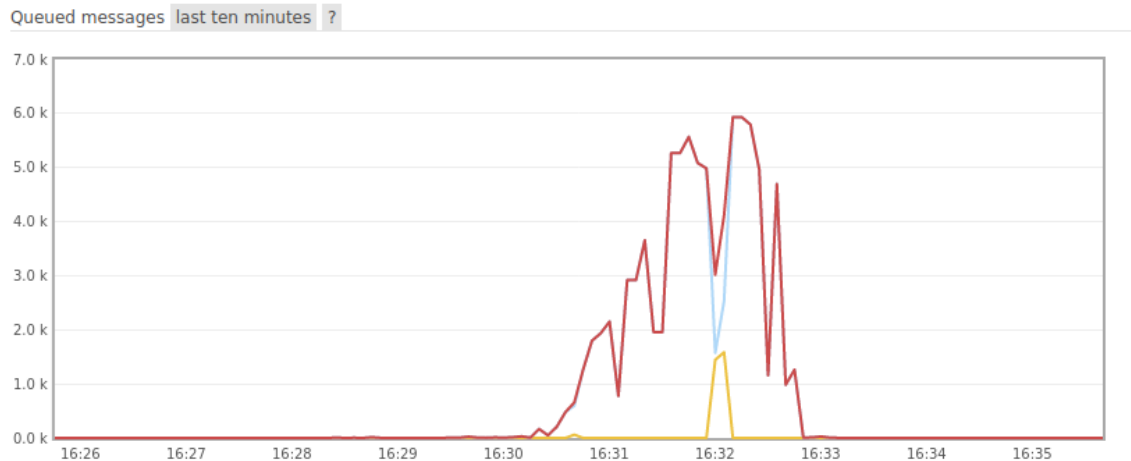
Στο Σχήμα 6.4 παρατηρείται ότι οι Καταναλωτές ακολουθούν πιστά το φορτίο των εισερχόμενων μηνυμάτων μέχρι τον ρυθμό άφιξης των 480 μηνυμάτων / δευτερόλεπτο περίπου.

Από τη συχνότητα των 500 μηνυμάτων / δευτερόλεπτο και ύστερα, ο ρυθμός κατανάλωσης μηνυμάτων δεν μπορεί να ακολουθήσει τον ρυθμό άφιξης μηνυμάτων, με συνέπεια να αρχίζουν να συνωστιζονται μηνύματα στις Ουρές. Αυτά παραδίδονται στον Καταναλωτή ο οποίος λειτουργεί στα όρια των δυνατοτήτων του προσπαθώντας να επε-



ξεργαστεί και τα μηνύματα που καταφθάνουν την κάθε στιγμή αλλά και τα μηνύματα που έχουν συνωστιστεί στο σύστημα.

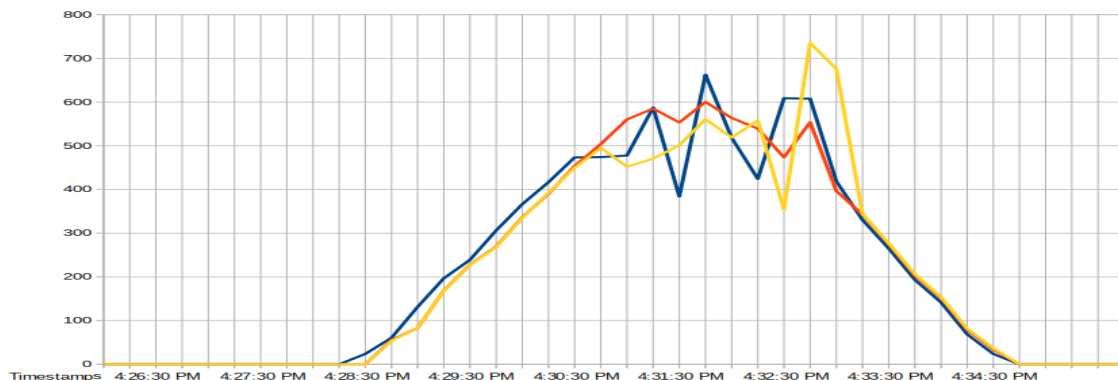
Κατά τη μείωση του ρυθμού εισαγωγής μηνυμάτων παρατηρείται ότι το σύστημα επιστρέφει σε μια σταθερή κατάσταση κάτω από το όριο των 400 μηνυμάτων / δευτερόλεπτο.



Σχήμα 6.3: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)

Μηνύματα έτοιμα προς παράδοση (κίτρινο)

Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)



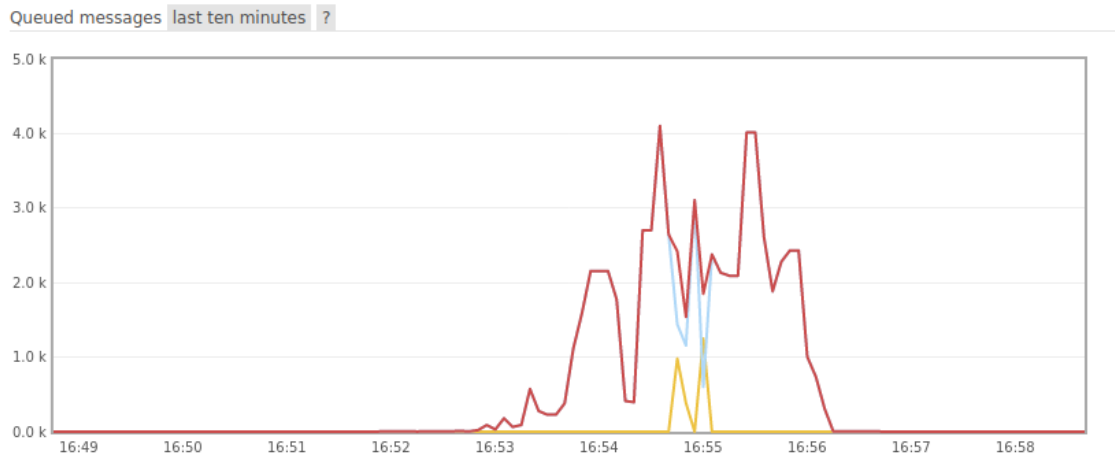
Σχήμα 6.4: Ρυθμός άφιξης μηνυμάτων (κόκκινο)

Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)

Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

### 6.3.3 4 Καταναλωτές | Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο

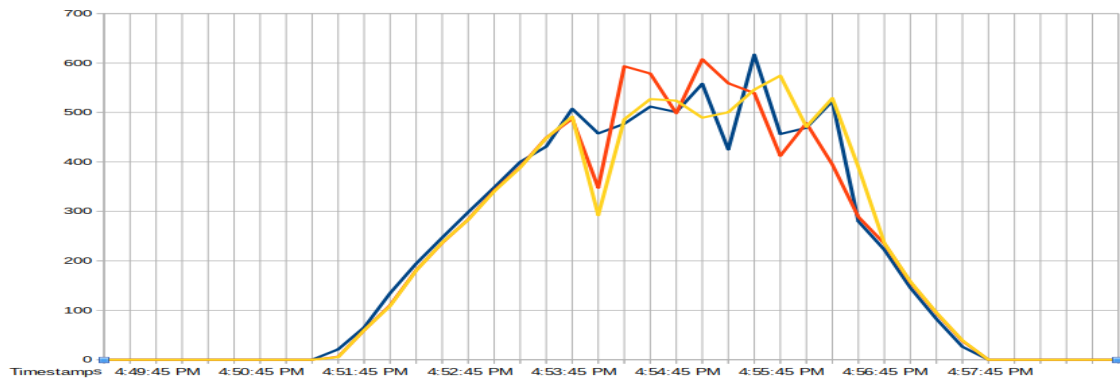
Παρόμοια συμπεριφορά με το προηγούμενο πείραμα παρουσιάζεται και εδώ. Οι 4 Καταναλωτές καλύπτουν το φορτίο των μηνυμάτων μέχρι ο ρυθμός άφιξής τους να ξεπεράσει τα 480 μηνύματα / δευτερόλεπτο. Το σύστημα τότε οδηγείται σε αστάθεια μέχρι ο ρυθμός αυτός να πέσει ξανά κάτω από τα 300 μηνύματα / δευτερόλεπτο.



Σχήμα 6.5: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)

Μηνύματα έτοιμα προς παράδοση (κίτρινο)

Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)



Σχήμα 6.6: Ρυθμός άφιξης μηνυμάτων (κόκκινο)

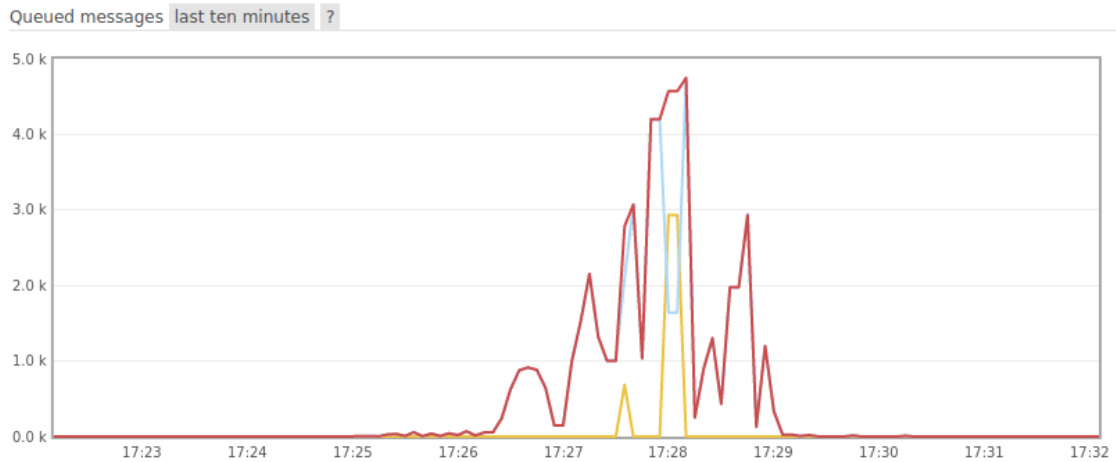
Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)

Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

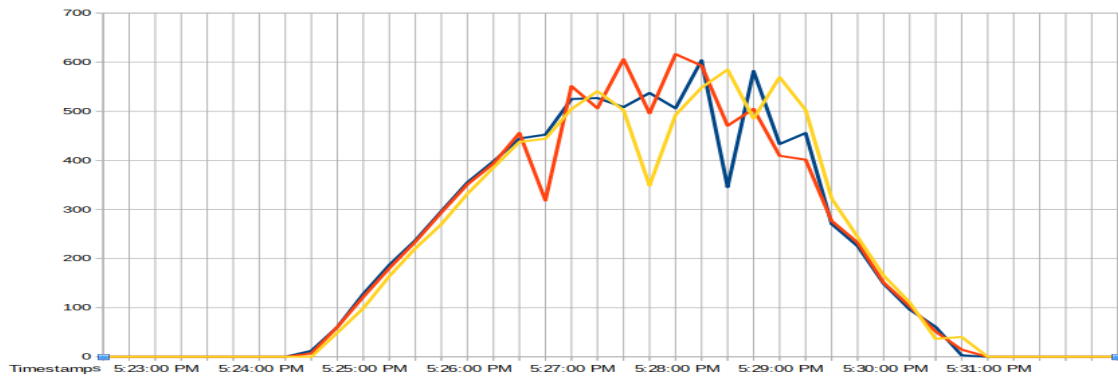
### 6.3.4 6 Καταναλωτές | Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο

Όπως φαίνεται στο σχήμα 6.6, η συχνότητα εισαγωγής μηνυμάτων στην οποία αρχίζει η αστάθεια του συστήματος υποχωρεί στα 450 μηνύματα / δευτερόλεπτο.

Η ισορροπία επιστρέφει στο σύστημα από τον ρυθμό των 400 μηνυμάτων / δευτερόλεπτο και κάτω.



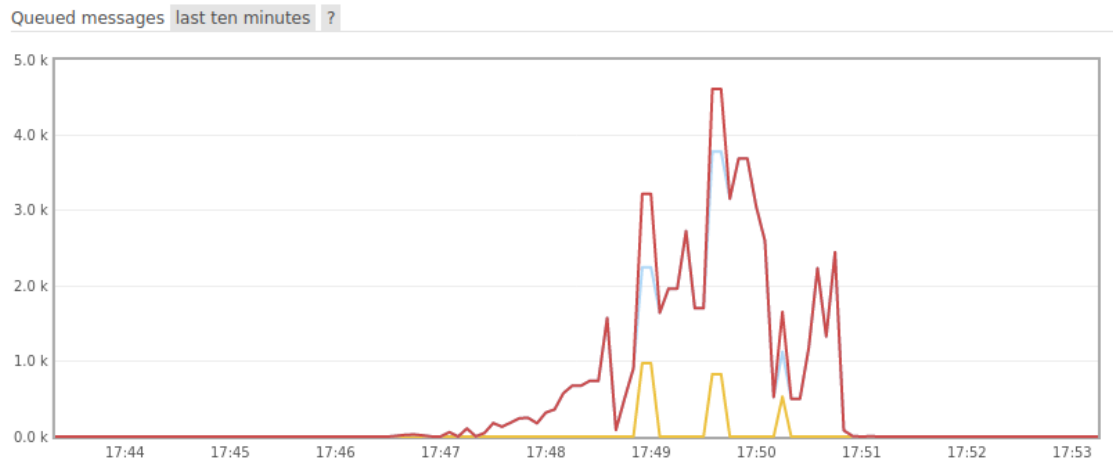
Σχήμα 6.7: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)  
 Μηνύματα έτοιμα προς παράδοση (κίτρινο)  
 Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)



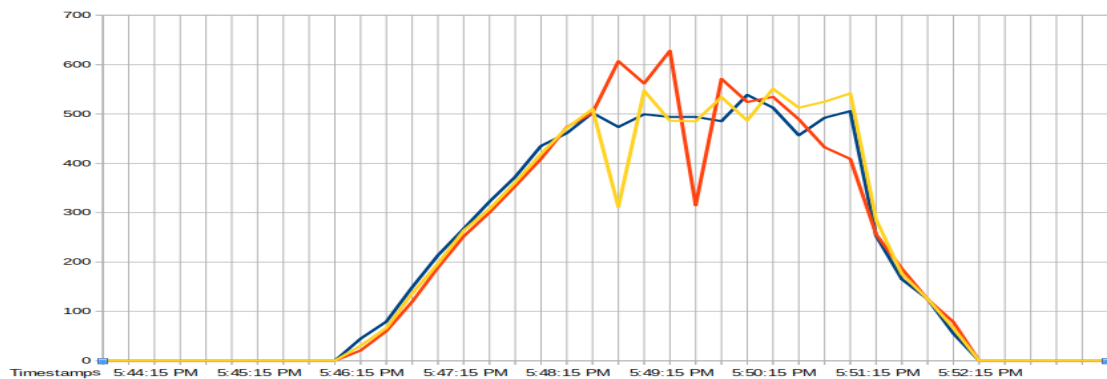
Σχήμα 6.8: Ρυθμός άφιξης μηνυμάτων (κόκκινο)  
 Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)  
 Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

### 6.3.5 8 Καταναλωτές | Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο

Η συμπεριφορά του συστήματος δεν αλλάζει σημαντικά με την προσθήκη άλλων δύο Καταναλωτών όπως φαίνεται στα παρακάτω διαγράμματα.



Σχήμα 6.9: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)  
 Μηνύματα έτοιμα προς παράδοση (κίτρινο)  
 Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)



Σχήμα 6.10: Ρυθμός άφιξης μηνυμάτων (κόκκινο)  
 Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)  
 Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

### 6.3.6 10 Καταναλωτές | Συχνότητες εισόδου 0 - 600 μηνύματα το δευτερόλεπτο

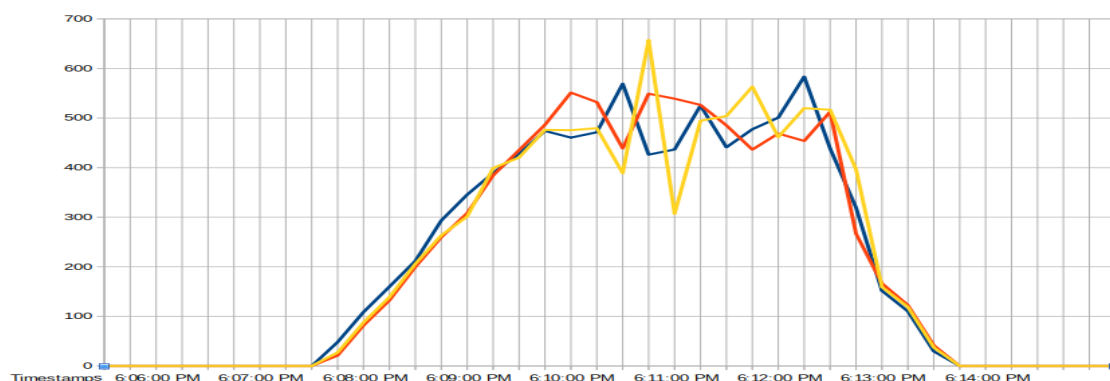
Αντίστοιχα με τα προηγούμενα το σύστημα πέφτει σε αστάθεια από τον ρυθμό των 450 μηνυμάτων / δευτερόλεπτο και άνω. Η ισορροπία επιστρέφει κάτω από τα 400 μηνύματα / δευτερόλεπτο.



Σχήμα 6.11: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)

Μηνύματα έτοιμα προς παράδοση (κίτρινο)

Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)



Σχήμα 6.12: Ρυθμός άφιξης μηνυμάτων (κόκκινο)

Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)

Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

## 6.4 Δεύτερος κύκλος πειραμάτων

Ο δεύτερος κύκλος πειραμάτων αφορά την καταγραφή της απόκρισης του συστήματος για αυξανόμενο αριθμό ενεργών Φίλτρων - Καταναλωτών με δεδομένο ρυθμό εισόδου μηνυμάτων στο σύστημα.

Η μέθοδος εκτέλεσης των πειραμάτων του δεύτερου κύκλου ήταν η εξής:

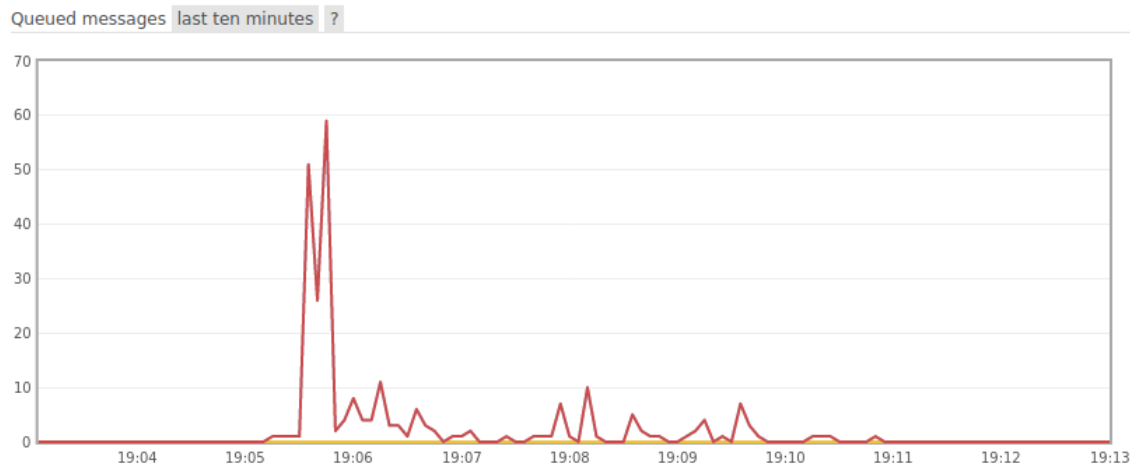
Κάθε πείραμα ξεκινούσε με 1 Καταναλωτή ανεπτυγμένο. Στη συνέχεια εφαρμοζόταν αποστολή μηνυμάτων συγκεκριμένου ρυθμού προς το σύστημα. Ο αριθμός των Καταναλωτών του συστήματος αυξανόταν κατά 1 κάθε 30 δευτερόλεπτα, μέχρι να φτάσει τους 10.

Πραγματοποιήθηκε επανάληψη του πειράματος για τις παρακάτω συχνότητες εισόδου μηνυμάτων στο σύστημα: 120, 240, 300, 360, 420 και 480 μηνυμάτων / δευτερόλεπτο.

### 6.4.1 Συχνότητα εισόδου 120 μηνύματα το δευτερόλεπτο | Εύρος Καταναλωτών 1 - 10

Ξεκινώντας τα πειράματα της δεύτερης φάσης βλέπουμε το σύστημα να ανταποκρίνεται ιδανικά στη συχνότητα εισόδου των 120 μηνυμάτων / δευτερόλεπτο. Η αύξηση των Καταναλωτών δεν είχε καμία επιρροή στη συμπεριφορά του συστήματος.

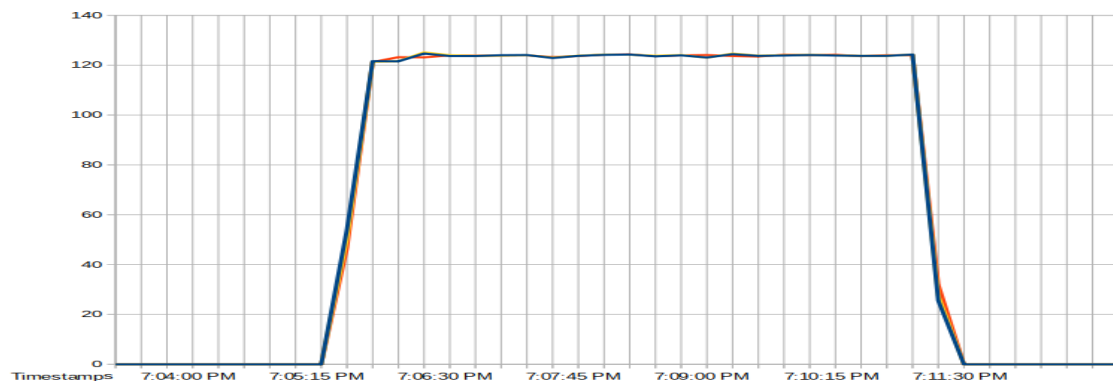
Επίσης ο αριθμός των συνωστιζόμενων μηνυμάτων στις Ουρές παρέμεινε ιδιαίτερα χαμηλός.



Σχήμα 6.13: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)

Μηνύματα έτοιμα προς παράδοση (κίτρινο)

Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)



Σχήμα 6.14: Ρυθμός άφιξης μηνυμάτων (κόκκινο)

Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)

Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

### 6.4.2 Συχνότητα εισόδου 240 μηνύματα το δευτερόλεπτο | Εύρος Καταναλωτών 1 - 10

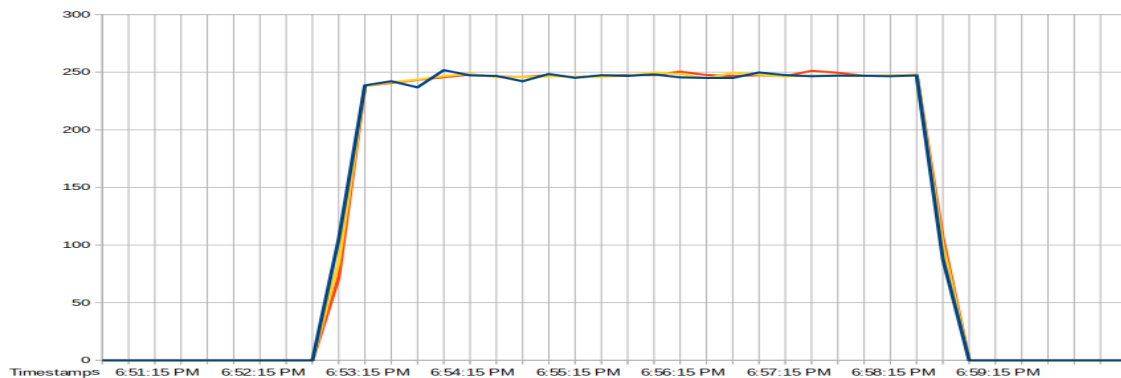
Το σύστημα παρέμεινε απόλυτα ισορροπημένο και στη δεύτερη εκτέλεση του πειράματος. Η μόνη διαφορά που παρατηρήθηκε ήταν η ελαφρώς μεγαλύτερη αναμονή μηνυμάτων στις Ουρές.



Σχήμα 6.15: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)

Μηνύματα έτοιμα προς παράδοση (κίτρινο)

Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)



Σχήμα 6.16: Ρυθμός άφιξης μηνυμάτων (κόκκινο)

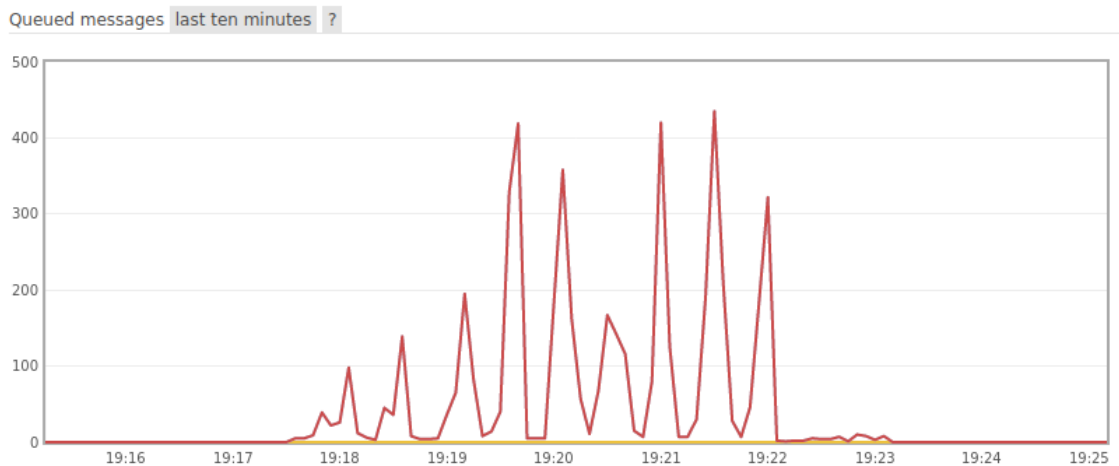
Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)

Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

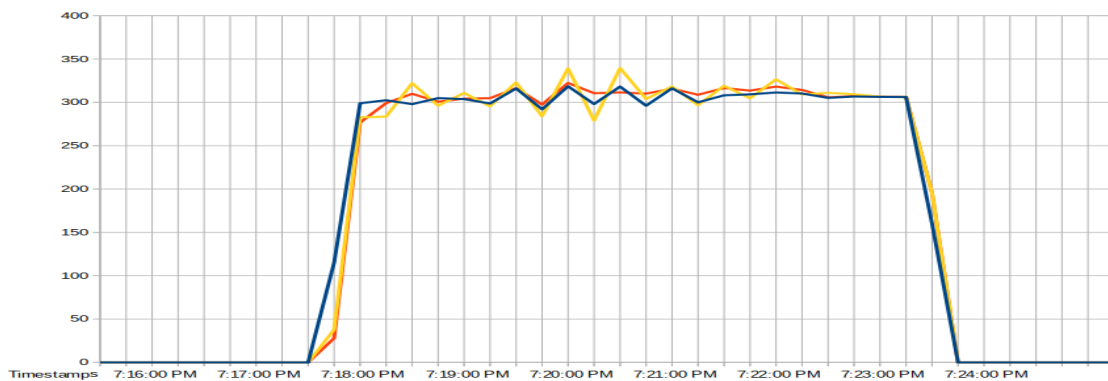
### 6.4.3 Συχνότητα εισόδου 300 μηνύματα το δευτερόλεπτο | Εύρος Καταναλωτών 1 - 10

Η συχνότητα των 300 μηνυμάτων / δευτερόλεπτο αποτελεί μια ενδιαφέρουσα τιμή καθώς αρχίζουν και φαίνονται οι επιπτώσεις της ενεργοποίησης του κάθε Καταναλωτή στο σύστημα. Συγκεκριμένα η κάθε έξαρση συσσωρευμένων μηνυμάτων στις Ουρές συμπίπτει χρονικά με τη δημιουργία κάποιου Καταναλωτή. Για αυτό υπάρχουν 9 ακμές, όσοι και οι Καταναλωτές οι οποίοι δημιουργήθηκαν μετά την εκκίνηση του πειράματος.

Αρχίζει να γίνεται ορατό δηλαδή το όριο των δυνατοτήτων του συστήματος δεδομένων των φυσικών πόρων του. Η εκκίνηση των Καταναλωτών καταναλώνει πόρους οι οποίοι δεσμεύονται και δεν μπορούν να χρησιμοποιηθούν στις άλλες διεργασίες που εκτελούνται παράλληλα. Για αυτό βλέπουμε και τον ρυθμό άφιξης μηνυμάτων να ξεφεύγει έστω ελάχιστα από τη σταθερότητα.



Σχήμα 6.17: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)  
 Μηνύματα έτοιμα προς παράδοση (κίτρινο)  
 Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)



Σχήμα 6.18: Ρυθμός άφιξης μηνυμάτων (κόκκινο)  
 Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)  
 Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

#### 6.4.4 Συχνότητα εισόδου 360 μηνύματα το δευτερόλεπτο | Εύρος Καταναλωτών 1 - 10

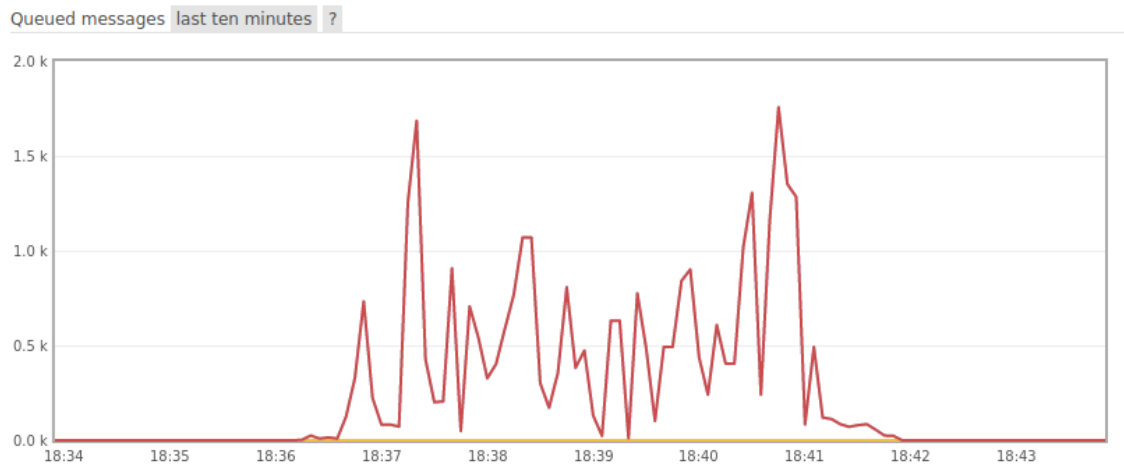
Σε αυτή τη συχνότητα εισόδου αρχίζει να γίνεται εμφανής η αδυναμία του συστήματος να συμβαδίσει με τον ρυθμό των εισερχόμενων μηνυμάτων. Έτσι εμφανίζονται μεγαλύτερες διακυμάνσεις στις στιγμιαίες τιμές των ρυθμών άφιξης κατανάλωσης και αποθήκευσης μηνυμάτων.

Πρακτικά το κάθε μέρος του συστήματος, στην προσπάθειά του να διαχειριστεί το φορτίο το οποίο του εφαρμόζεται, καταναλώνει πολλούς πόρους οι οποίοι λείπουν στιγμιαία από τα άλλα υποσυστήματα. Έτσι δημιουργείται αυτό το φαινόμενο των περιοδικών διακυμάνσεων που φαίνεται στο σχήμα 6.20.

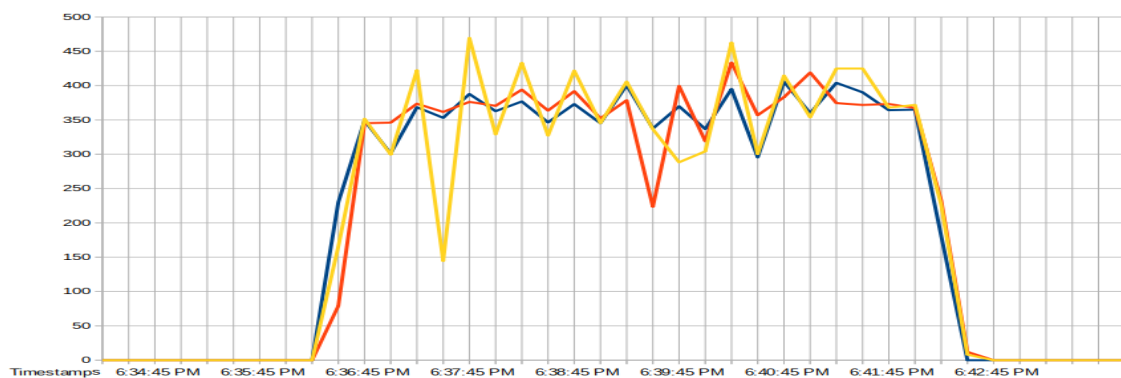
Η επιρροή της δημιουργίας του κάθε Καταναλωτή είναι ακόμα ορατή καθώς οι εντάσεις των αταξιών των γραφημάτων ταυτίζονται με τις χρονικές στιγμές δημιουργίας των Καταναλωτών.



Το πλήθος των συσσωρευμένων μηνυμάτων στις Ουρές αντίστοιχα έχει αυξηθεί σημαντικά σε σχέση με τα προηγούμενα πειράματα.



Σχήμα 6.19: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)  
Μηνύματα έτοιμα προς παράδοση (κίτρινο)  
Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)

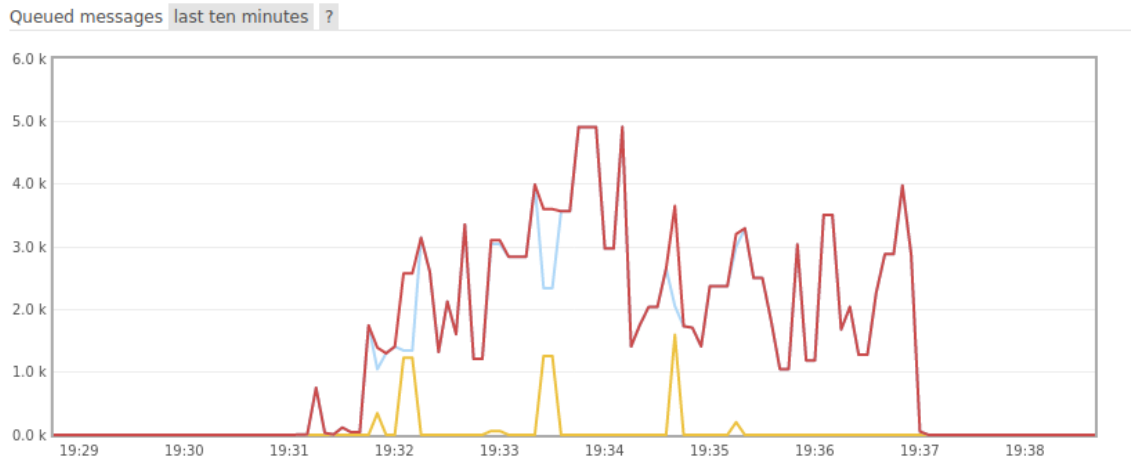


Σχήμα 6.20: Ρυθμός άφιξης μηνυμάτων (κόκκινο)  
Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)  
Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

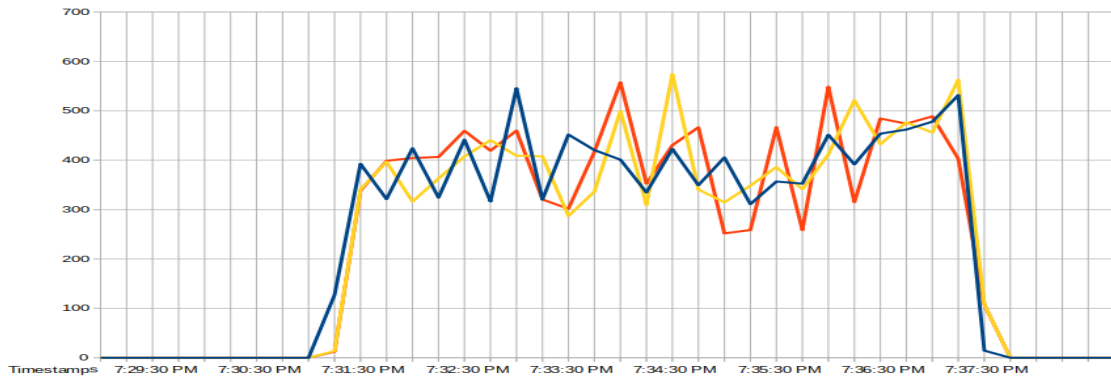
#### 6.4.5 Συχνότητα εισόδου 420 μηνύματα το δευτερόλεπτο | Εύρος Καταναλωτών 1 - 10

Παρατηρείται από τα παρακάτω διαγράμματα ότι πλέον το σύστημα φτάνει κοντά στην αστάθεια άσχετα με τον αριθμό των ενεργών Καταναλωτών και άσχετα με τις χρονικές στιγμές δημιουργίας τους.

Ο αριθμός των αποθηκευμένων μηνυμάτων στις Ουρές αυξάνεται και πάλι.



Σχήμα 6.21: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)  
Μηνύματα έτοιμα προς παράδοση (κίτρινο)  
Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)

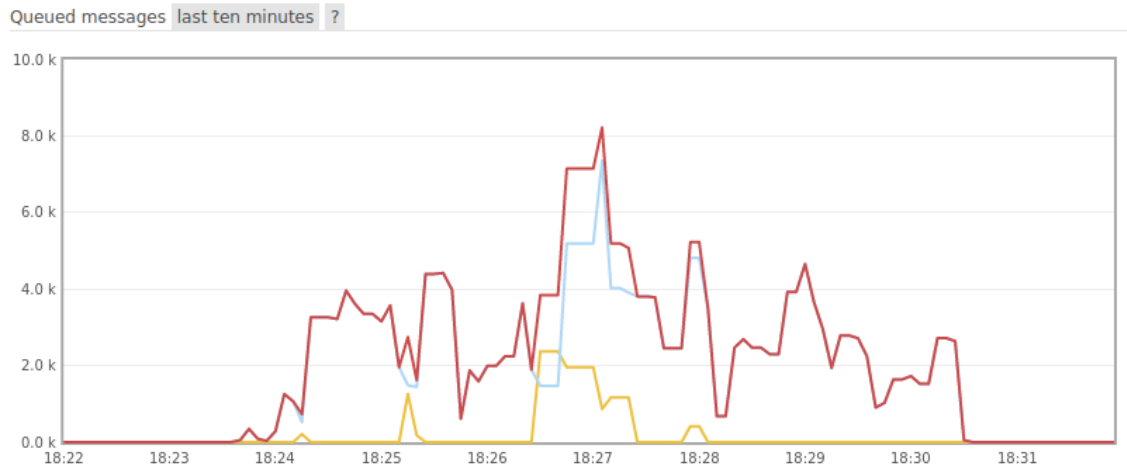


Σχήμα 6.22: Ρυθμός άφιξης μηνυμάτων (κόκκινο)  
Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)  
Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

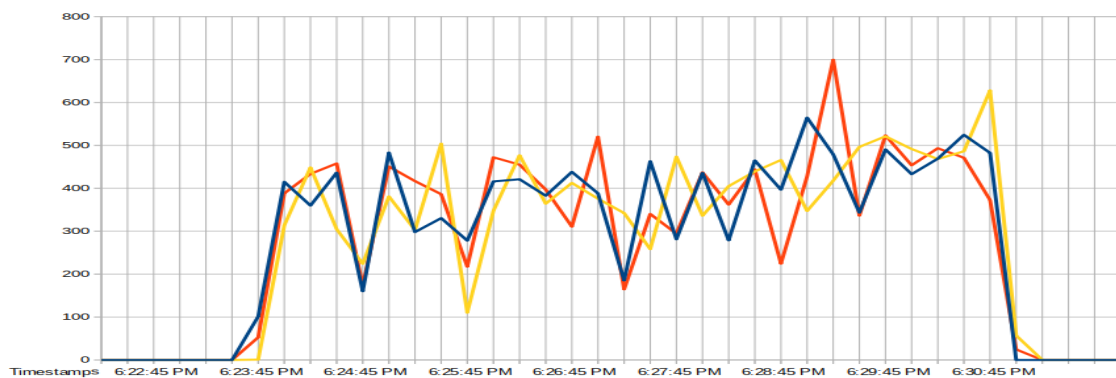
#### 6.4.6 Συχνότητα εισόδου 480 μηνύματα το δευτερόλεπτο | Εύρος Καταναλωτών 1 - 10

Το σύστημα μπορεί να χαρακτηριστεί ασταθές πλέον, καθώς οι διακυμάνσεις των ρυθμών άφιξης κατανάλωσης και αποθήκευσης μηνυμάτων γίνονται ιδιαίτερα βίαιες από την αρχή μέχρι το τέλος του πειράματος.

Παρατηρούμε ότι η συχνότητα των 480 μηνυμάτων / δευτερόλεπτο είναι η ίδια συχνότητα στην οποία παρατηρήθηκε στην πρώτη φάση πειραμάτων ότι το σύστημα οδηγείται σε αστάθεια. Αυτό επιβεβαιώνει τις προηγούμενες παρατηρήσεις μας. Αυτός είναι και ο λόγος για τον οποίο κρίθηκε ότι δεν ήταν απαραίτητη η συνέχιση των πειραμάτων σε υψηλότερες συχνότητες εισόδου, καθώς ήδη το σύστημα αδυνατεί να συμβαδίσει με το φορτίο που του επιβάλλεται σε καθεστώς ισορροπίας.



Σχήμα 6.23: Μηνύματα αποθηκευμένα σε Ουρές (κόκκινο)  
Μηνύματα έτοιμα προς παράδοση (κίτρινο)  
Μηνύματα για τα οποία δεν έχει καταγραφεί Επιβεβαίωση Λήψης (γαλάζιο)



Σχήμα 6.24: Ρυθμός άφιξης μηνυμάτων (κόκκινο)  
Ρυθμός κατανάλωσης μηνυμάτων (κίτρινο)  
Ρυθμός καταγραφής μηνυμάτων στη ΒΔ (μπλε)

## 6.5 Συμπεράσματα

### 6.5.1 Επιρροή του αριθμού Καταναλωτών στη συμπεριφορά του συστήματος

Εποπτεύοντας συνολικά τον πρώτο κύκλο πειραμάτων, είναι εμφανής μια σταδιακή επιδείνωση της συμπεριφοράς του συστήματος με κάθε νέα προσθήκη ενός Καταναλωτή.

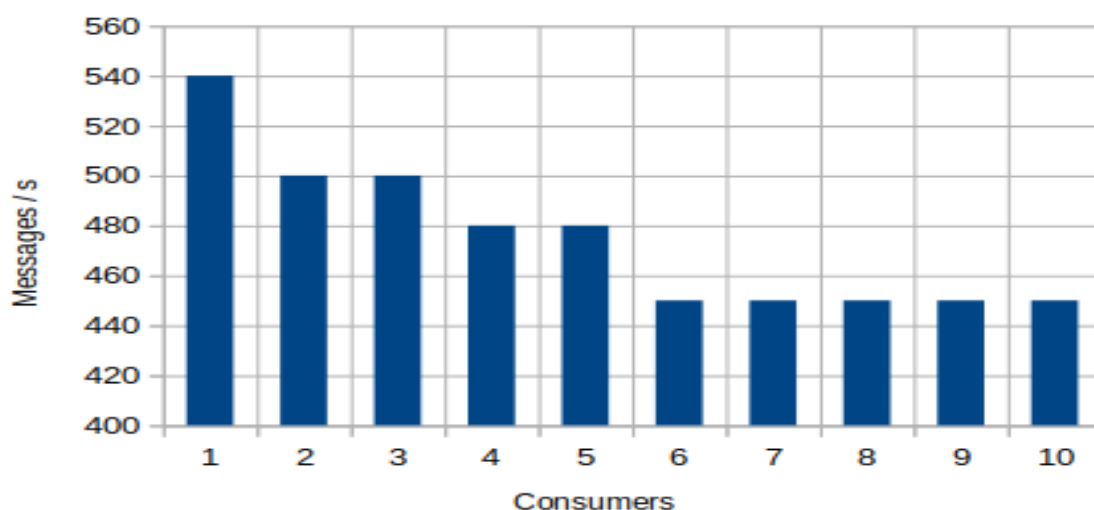
Η γραφική παράσταση των ρυθμών άφιξης, κατανάλωσης και αποθήκευσης μηνυμάτων παρουσίασε αρχικά μια ομαλή καμπύλη σαρώνοντας τις συχνότητες εισόδου από 0 έως 600 μηνύματα / δευτερόλεπτο. Με κάθε νέα προσθήκη Καταναλωτών παρατηρήθηκε ότι όλο και μεγαλύτερο κομμάτι στην κορυφή του διαγράμματος παρουσίαζε αυξανόμενες διακυμάνσεις.

Η τελική μορφή της συμπεριφοράς του συστήματος χαρακτηρίζεται από δύο περιοχές. Η πρώτη αναφέρεται στα δυο ομαλά κομμάτια στην αρχή και στο τέλος των διαγραμμάτων στις οποίες το φορτίο εξυπηρετείται ικανοποιητικά. Η δεύτερη αποτελεί την περιοχή στην οποία παρατηρείται τοπική αστάθεια με μεγάλες διακυμάνσεις των αποδόσεων του κάθε υποσυστήματος.

Στον παρακάτω πίνακα συνοψίζονται οι συχνότητες εισόδου στις οποίες αρχίζει η περιοχή αστάθειας για κάθε αριθμό Καταναλωτών:

Πίνακας 6.1: Σύγκριση συχνοτήτων εκκίνησης αστάθειας των πειραμάτων του πρώτου κύκλου

Αριθμός Καταναλωτών:	1	2	3	4	5	6	7	8	9	10
Συχνότητα αστάθειας (μηνύματα / δευτερόλεπτο):	540	500	500	480	480	450	450	450	450	450



Σχήμα 6.25: Σύγκριση συχνοτήτων εκκίνησης αστάθειας των πειραμάτων του πρώτου κύκλου

Από τα παραπάνω προκύπτει το συμπέρασμα ότι η βέλτιστη επιλογή στο συγκεκριμένο σύστημα είναι η χρήση ενός Καταναλωτή με τη χρήση όλων των διαθέσιμων πόρων από αυτόν. Η διαχείριση του φορτίου σε αυτή την περίπτωση είναι βέλτιστη, καθώς κάθε πρόσθετος Καταναλωτής εισάγει μια μικρή επιβάρυνση στο σύστημα.

### 6.5.2 Επιρροή της συχνότητας εισόδου μηνυμάτων στη συμπεριφορά του συστήματος

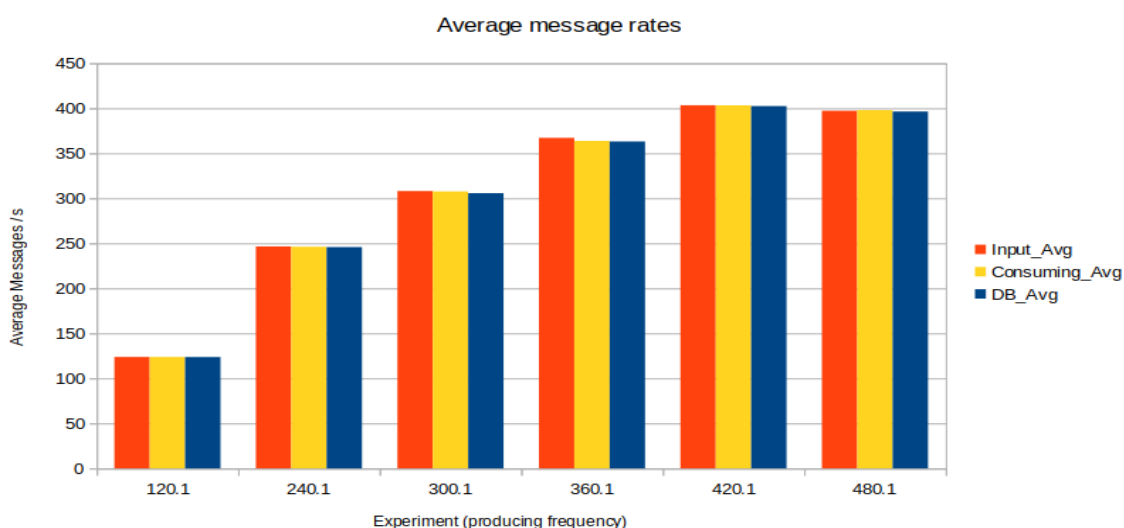
Κατά την εκτέλεση του δεύτερου κύκλου πειραμάτων έγινε ξεκάθαρο ότι ο κύριος παράγοντας της συμπεριφοράς του συστήματος είναι η συχνότητα εισόδου των μηνυμάτων.

Σε όλα τα πειράματα της δεύτερης φάσης, η προσθήκη Καταναλωτών δεν βελτιώνει την απόδοση του συστήματος, αντίθετα την επιβάρυνε σύντομα κατά τις χρονικές στιγμές δημιουργίας των Καταναλωτών. Αυτό συνέβαινε καθώς πόροι αναγκαίοι για την δρομολόγηση, επεξεργασία και αποθήκευση των μηνυμάτων δεσμευόντουσαν για να εκτελεστούν οι διαδικασίες εκκίνησης των Καταναλωτών.

Σημαντικό αποτέλεσμα αποτελεί η επιβεβαίωση των παρατηρήσεων του πρώτου κύκλου πειραμάτων σχετικά με την κρίσιμη συχνότητα εισόδου μηνυμάτων. Όπως παρατηρήθηκε οι συχνότητες εισόδου μέχρι και τα 420 μηνύματα / δευτερόλεπτο ήταν ως επί το πλείστον διαχειρίσιμες από το σύστημα. Από αυτή τη συχνότητα και πάνω το σύστημα παρουσίασε ασταθή συμπεριφορά.

Επομένως ως κρίσιμη συχνότητα εισαγωγής μηνυμάτων στο συγκεκριμένο σύστημα με τους δεδομένους πόρους αναγνωρίστηκαν τα 420 μηνύματα / δευτερόλεπτο, αναξάρτητα από τον αριθμό των Καταναλωτών.

Το παραπάνω συμπέρασμα επιβεβαιώνεται και από το διάγραμμα 6.26, στο οποίο είναι εμφανές ότι μετά τη συχνότητα των 420 μηνυμάτων / δευτερόλεπτο, η απόδοση του συστήματος χειροτερεύει.



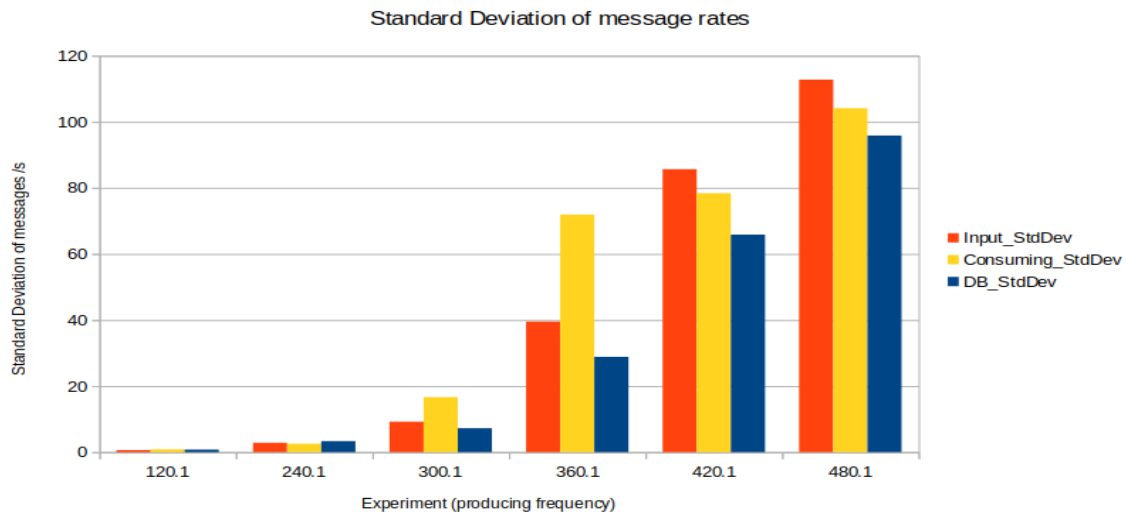
Σχήμα 6.26: Σύγκριση των μέσων τιμών της εισαγωγής μηνυμάτων, της κατανάλωσης μηνυμάτων και της καταγραφής μηνυμάτων στη ΒΔ για κάθε πείραμα της δεύτερης φάσης.

Άφιξη μηνυμάτων (κόκκινο)

Κατανάλωση μηνυμάτων (κίτρινο)

Καταγραφή μηνυμάτων στη ΒΔ (μπλε)

Την επιδείνωση της απόδοσης του συστήματος μπορεί να αντιληφθεί κανείς ακόμα καλύτερα παρατηρώντας το διάγραμμα 6.27, στο οποίο παρουσιάζονται οι τυπικές αποκλίσεις των συχνοτήτων εισαγωγής μηνυμάτων, κατανάλωσης μηνυμάτων και καταγραφής μηνυμάτων στη ΒΔ για κάθε πείραμα της δεύτερης φάσης.



Σχήμα 6.27: Σύγκριση των τυπικών αποκλίσεων της εισαγωγής μηνυμάτων, της κατανάλωσης μηνυμάτων και της καταγραφής μηνυμάτων στη ΒΔ για κάθε πείραμα της δεύτερης φάσης.

Άφιξη μηνυμάτων (κόκκινο)

Κατανάλωση μηνυμάτων (κίτρινο)

Καταγραφή μηνυμάτων στη ΒΔ (μπλε)

Τέλος χρήσιμη είναι και η σύγκριση της μέσης τιμής των δειγμάτων με την τυπική απόκλισή τους για κάθε πείραμα της δεύτερης φάσης που εκτελέστηκε. Η σύνοψη αυτή η οποία παρουσιάζεται στους τρεις πίνακες που ακολουθούν, μας δείχνει τη σταδιακή επιδείνωση της απόδοσης του κάθε μέρους του συστήματος ξεχωριστά.

Οι παρακάτω τιμές της τυπικής απόκλισης ως ποσοστά της μέσης τιμής μας δείχνουν πόσο πιο ασταθής γίνεται η απόδοση του συστήματος ανάλογα με τη συχνότητα δημοσίευσης μηνυμάτων από τους Παραγωγούς.

Πίνακας 6.2: Μέσες τιμές και τυπικές αποκλίσεις της συχνότητας εισαγωγής μηνυμάτων των δειγμάτων για κάθε πείραμα της δεύτερης φάσης.

Συχνότητα παραγωγής μηνυμάτων:	120	240	300	360	420	480
Μέση τιμή συχνότητας εισόδου:	123.696	246.472	308.030	367.093	403.333	397.245
Τυπική απόκλιση συχνότητας εισόδου:	0.622	2.814	9.202	39.508	85.655	112.734
Τυπική απόκλιση ως ποσοστό της μέσης τιμής:	0.50%	1.14%	2.98%	10.76%	21.23%	28.37%

Πίνακας 6.3: Μέσες τιμές και τυπικές αποκλίσεις της συχνότητας κατανάλωσης μηνυμάτων των δειγμάτων για κάθε πείραμα της δεύτερης φάσης.

Συχνότητα παραγωγής μηνυμάτων:	120	240	300	360	420	480
Μέση τιμή συχνότητας κατανάλωσης:	123.730	246.166	307.581	363.648	403.247	397.938
Τυπική απόκλιση συχνότητας κατανάλωσης:	0.838	2.560	16.634	71.910	78.374	104.110
Τυπική απόκλιση ως ποσοστό της μέσης τιμής:	0.67%	1.04%	5.40%	19.77%	19.43%	26.16%

Πίνακας 6.4: Μέσες τιμές και τυπικές αποκλίσεις της συχνότητας καταγραφής μηνυμάτων στη ΒΔ των δειγμάτων για κάθε πείραμα της δεύτερης φάσης.

Συχνότητα παραγωγής μηνυμάτων:	120	240	300	360	420	480
Μέση τιμή συχνότητας καταγραφής στη ΒΔ:	123.654	245.781	305.672	363.136	402.438	396.380
Τυπική απόκλιση συχνότητας καταγραφής στη ΒΔ:	0.766	3.298	7.223	28.832	65.835	95.842
Τυπική απόκλιση ως ποσοστό της μέσης τιμής:	0.61%	1.34%	2.36%	7.93%	16.35%	24.17%

# Κεφάλαιο 7

## Μελλοντική εργασία

### 7.1 Επεκτάσεις του συστήματος

Η πλατφόρμα που αναπτύχθηκε στην παρούσα διπλωματική μπορεί να βρει εφαρμογή σε ένα μεγάλο πλήθος πεδίων. Θεωρητικά οποιοδήποτε σύστημα συμπεριλαμβάνει τη διαχείριση και διακίνηση μεγάλου όγκου δεδομένων μηνυμάτων μπορεί να βελτιωθεί με τη χρήση των εργαλείων της πλατφόρμας αυτής. Από τις δυνατότητές της μπορούν να ωφεληθούν τόσο επαγγελματίες αλλά και ερευνητές διαφορετικών υποβάθρων.

Από τα παραπάνω γίνεται κατανοητό ότι λόγω της μεγάλης ποικιλίας συστημάτων στα οποία μπορεί να συνεισφέρει κάποιο θετικό πρόσημο η συγκεκριμένη πλατφόρμα, αντίστοιχα πολλές μπορεί να είναι και οι δυνητικές επεκτάσεις που μπορεί να δεχθεί για να υποστηρίξει αυτά τα διαφορετικά συστήματα.

Παρακάτω αναφέρονται μερικές μόνο επεκτάσεις του συστήματος που κρίνονται και ως οι πιο σημαντικές και οι πιο άμεσα χρηστικές.

#### 7.1.1 Αύξηση διαθέσιμων ενεργειών για μηνύματα ενδιαφέροντος

Στην τρέχουσα μορφή της πλατφόρμας που αναπτύχθηκε, διατίθεται μια μοναδική ενέργεια που μπορεί να εφαρμοστεί όταν κάποιο εισερχόμενο μήνυμα ικανοποιεί τις συνθήκες που έχει ορίσει ο χρήστης. Αυτή η ενέργεια είναι η αποθήκευση του μηνύματος στη ΒΔ.

Θα ήταν χρήσιμο να οριστούν περισσότερες διαθέσιμες ενέργειες. Κάποια παραδείγματα θα μπορούσαν να είναι τα ακόλουθα:

- Η δημιουργία μιας ειδοποίησης προς τον χρήστη στη διεπαφή του χρήστη
- Η αποστολή ενός email με παραμετροποιήσιμο παραλήπτη με περιεχόμενο τα δεδομένα του μηνύματος και τη συνθήκη που ικανοποιήθηκε
- Η δημιουργία ενός HTTP request προς κάποιο API ορισμένο από τον χρήστη με προαιρετικό περιεχόμενο τα δεδομένα του μηνύματος για περαιτέρω επεξεργασία
- Η κλιμάκωση και αποκλιμάκωση του ίδιου του συστήματος θα μπορούσε να προκαλείται από τη λήψη συγκεκριμένων προειδοποιητικών μηνυμάτων



### **7.1.2 Εμπλουτισμός δυνατοτήτων παρακολούθησης και διαχείρισης μηνυμάτων**

Χρήσιμο θα ήταν επίσης να υλοποιηθούν πρόσθετες λειτουργίες παρακολούθησης και διαχείρισης των μηνυμάτων που έχουν ήδη καταγραφεί από κάποιο Φίλτρο στη ΒΔ.

Κάποια παραδείγματα τέτοιων λειτουργιών αναφέρονται ενδεικτικά παρακάτω:

- Η παροχή δυνατότητας στον χρήστη να εξάγει τα καταγεγραμμένα μηνύματα της ΒΔ σε ένα αρχείο και να τα αποθηκεύει τοπικά
- Η υλοποίηση αναζήτησης εντός του παραθύρου παρουσίασης των καταγεγραμμένων μηνυμάτων των Φίλτρων προς περιορισμό των αποτελεσμάτων
- Η μεμονωμένη ή και ομαδική διαγραφή μηνυμάτων από τη ΒΔ
- Η αυτοματοποίηση διαγραφής μηνυμάτων μετά την πάροδο χρονικού διαστήματος επιλέξιμου από τον χρήστη
- Η ταξινόμηση των καταγεγραμμένων μηνυμάτων κατά αύξουσα ή φθίνουσα σειρά με βάση κάποια παράμετρο επιλέξιμη από τον χρήστη

### **7.1.3 Μεγαλύτερη ποικιλία συνθηκών εκτέλεσης ενεργειών**

Μια άλλη χρήσιμη προσθήκη στις δυνατότητες του συστήματος θα μπορούσε να είναι η υποστήριξη περισσότερων προτύπων συνθηκών εκτέλεσης ενεργειών.

Οι συνθήκες που υποστηρίζονται αυτή τη στιγμή έχουν τη μορφή μαθηματικών συγκρίσεων και ελέγχου ταύτισης ακολουθιών χαρακτήρων. Κατά αυτή τη μέθοδο, ελέγχονται οι μεταβλητές που βρίσκονται στα περιεχόμενα του μηνύματος. Η ενέργεια του Φίλτρου (αποθήκευση στη ΒΔ επί του παρόντος) ενεργοποιείται εάν βρεθεί συδυασμός μεταβλητής - τιμής που ικανοποιεί τη σχέση που έχει ορίσει ο χρήστης για αυτή τη μεταβλητή.

Πρακτικά ελέγχεται αν η τιμή της μεταβλητής είναι μικρότερη, μεγαλύτερη ή ίση με την τιμή ελέγχου που έχει δοθεί από τον χρήστη σε περίπτωση αριθμητικής μεταβλητής. Εάν η μεταβλητή αποτελείται από μια ακολουθία χαρακτήρων, η συνθήκη που εφαρμόζεται είναι ο έλεγχος της ταύτισης της ακολουθίας χαρακτήρων του μηνύματος με μια ακολουθία χαρακτήρων ελέγχου δοσμένη από τον χρήστη.

Εναλλακτικές μορφές συνθηκών ενεργοποίησης δράσεων θα μπορούσαν να είναι οι εξής:

- Η χρήση regular expressions με σκοπό την αναγνώριση συγκεκριμένων προτύπων στα περιεχόμενα κειμένου των μηνυμάτων
- Η ενεργοποίηση δράσεων με βάση το μέγεθος του μηνύματος
- Η ενεργοποίηση δράσεων με βάση τις παραμέτρους του μηνύματος που ορίζονται στο RabbitMQ (π.χ. timestamp, userId, deliveryMode, priority, headers)

#### **7.1.4 Παροχή προτύπων παραμετροποίησης Φίλτρων για συγκεκριμένα σενάρια χρήσης**

Στο πλαίσιο της μείωσης των απαραίτητων γνώσεων της τεχνολογίας Διαμεσολάβησης Μηνυμάτων που πρέπει να έχει κάποιος χρήστης του συστήματος, θα μπορούσαν να προστεθούν πρότυπα παραμετροποίησης Φίλτρων.

Το κάθε πρότυπο θα μπορούσε να αντιστοιχεί σε ένα σύνηθες σενάριο χρήσης και θα είχε προκαθορισμένες τιμές για τις παραμέτρους του Φίλτρου που θα υλοποιούσε. Έτσι ο χρήστης δε θα χρειάζεται να έχει γνώση των εσωτερικών στοιχείων του Διαμεσολαβητή όπως είναι τα Exchanges, οι Ουρές και τα κλειδιά δρομολόγησης. Αντίθετα θα μπορεί να εφαρμόζει Φίλτρα ορίζοντας τον ελάχιστο αριθμό παραμέτρων για την εκτέλεση ενός σεναρίου χρήσης.

Κάτι τέτοιο θα έκανε το σύστημα χρησιμοποιήσιμο από ακόμα μεγαλύτερη μερίδα του πληθυσμού καθώς οι γνωστικές απαιτήσεις θα μειώνονταν.

#### **7.1.5 Ανάπτυξη μιας Domain Specific Language (DSL) για την εφαρμογή φίλτρων**

Σε συνέχεια της προηγούμενης πρότασης θα μπορούσε να οριστεί μια Domain Specific Language με σκοπό την αυτοματοποίηση της παραπάνω διαδικασίας. Με μία DSL θα διευκολυνόταν η διαδικασία περιγραφής νέων σεναρίων χρήσης και η δημιουργία κατάλληλων Φίλτρων που να καλύπτουν τις ανάγκες τους.

#### **7.1.6 Υποστήριξη διαφορετικών ρόλων χρηστών με κατάλληλες δυνατότητες**

Όσο οι δυνατότητες και οι χρήστες του συστήματος αυξάνονται, τόσο γίνονται αναγκαίες λειτουργίες παρακολούθησης και ελέγχου του συστήματος. Επομένως κρίνεται συνετό να γίνει λόγος για ανάπτυξη δυνατοτήτων διαχείρισης των χρηστών και των πόρων του συστήματος. Προφανώς καθώς τέτοιες λειτουργίες δεν μπορούν να είναι προσβάσιμες από το σύνολο των απλών χρηστών, θα έπρεπε να εισαχθούν νέοι ρόλοι χρηστών στο σύστημα στους οποίους να ανατίθενται οι αντίστοιχες αρμοδιότητες και τα δικαιώματα αξιοποίησης των λειτουργιών διαχείρισης.

Η ανάπτυξη λειτουργιών καταγραφής δεδομένων και στατιστικών σχετικά με τη λειτουργία του συστήματος θεωρείται ότι θα ήταν επίσης απαραίτητη για την εκτέλεση των καθηκόντων των διαχειριστών του συστήματος.

#### **7.1.7 Αυτοματοποίηση της κλιμάκωσης του συστήματος**

Μία ισχυρή δυνατότητα που παρέχει η ανάπτυξη σε K8s περιβάλλον είναι η αυτοματοποίηση της κλιμάκωσης του συστήματος. Θα μπορούσε επομένως να υλοποιηθεί κατάλληλη δομή, έτσι ώστε σε συνθήκες υψηλού φορτίου, να γίνεται αυτόματη επέκταση του συστήματος.

Αυτή η επέκταση θα μπορούσε να αφορά την αύξηση των διαθέσιμων πόρων στο μηχανήμα στο οποίο εκτελείται το σύστημα, την επέκταση του συστήματος σε πρόσθετα μηχανήματα (Nodes), ή συνδυασμό και των δύο.

Η κλιμάκωση θα μπορούσε να εξαρτάται από μεγέθη όπως είναι η συχνότητα εισόδου μηνυμάτων στους Διαμεσολαβητές, ο αριθμός των ενεργών χρηστών την κάθε δεδομένη στιγμή και ο αριθμός των ενεργειών αποθήκευσης στη ΒΔ.

## **7.2 Ασφάλεια - Security context**

Πριν γίνει οποιοδήποτε σύστημα διαθέσιμο στους πραγματικούς χρήστες, πρέπει να δοθεί ιδιαίτερη σημασία στο πλαίσιο ασφαλείας.

### **7.2.1 Ενεργοποίηση TLS στις επικοινωνίες**

Η πρώτη και πιο σημαντική ενέργεια αυτού του τομέα θα ήταν η ενεργοποίηση πρωτοκόλλων ασφαλείας στις επικοινωνίες μεταξύ των μερών του συστήματος. Ένα τέτοιο πρωτόκολλο είναι το TLS μέσω του οποίου πραγματοποιείται κρυπτογράφηση των διακινούμενων δεδομένων.

Η κρυπτογράφηση μέσω TLS στηρίζεται στη χρήση ψηφιακών πιστοποιητικών. Μπορεί να αναπτυχθεί κατάλληλη δομή με σκοπό την αυτοματοποίηση της παραγωγής, διαχείρισης και ανανέωσης των πιστοποιητικών του συστήματος λοιπόν.

### **7.2.2 Κρυπτογράφηση ΒΔ**

Ένα άλλο μέτρο ασφαλείας που είναι ιδιαίτερα σημαντικό για δύο λόγους κυρίως είναι η κρυπτογράφηση της ΒΔ. Ο πρώτος και προφανής λόγος είναι η αποφυγή της διαρροής ευαίσθητων δεδομένων των χρηστών προς κάποιον επιτιθέμενο ο οποίος θα αποκτήσει με κάποιον τρόπο πρόσβαση στη ΒΔ.

Ο δεύτερος λόγος είναι ότι εκτός από τους επιτιθέμενους, ούτε οι διαχειριστές του συστήματος θα έπρεπε να έχουν πρόσβαση στα ευαίσθητα δεδομένα των χρηστών του. Έτσι είναι κρίσιμο τα δεδομένα στη ΒΔ να είναι κρυπτογραφημένα με το κλειδί του χρήστη ώστε να είναι προσβάσιμα αποκλειστικά από αυτόν.

### **7.2.3 Εφαρμογή κανόνων τείχους προστασίας (firewall)**

Με την εφαρμογή κανόνων τείχους προστασίας, μπορούν να καθοριστούν αυστηρά οι δυνατότητες επικοινωνίας του κάθε υποσυστήματος με τα υπόλοιπα. Με αυτόν τον τρόπο γίνεται πιο δύσκολη άλλη μια πιθανή τακτική επίθεσης προς το σύστημα.

### **7.2.4 Εκτέλεση των διεργασιών ως απλοί χρήστες - όχι ως root μέσα στα containers**

Τέλος αυτό ένα μέτρο το οποίο είναι χρήσιμο στην περίπτωση που κάποιος επιτιθέμενος αποκτήσει πρόσβαση σε κάποιο container. Η εκτέλεση των διεργασιών στο κάθε υποσύστημα με τα ελάχιστα απαιτούμενα δικαιώματα για την ολοκλήρωση των λειτουργιών του μπορεί να αποτρέψει την εισχώρηση του επιτιθέμενου βαθύτερα στο σύστημα.

### **7.3 Ανάπτυξη του συστήματος σε περιβάλλον εμπορικής χρήσης**

Η ανάπτυξη του συστήματος στο πλαίσιο αυτής της διπλωματικής εργασίας έγινε σε τοπικό περιβάλλον με τα μέσα που ήταν διαθέσιμα.

Τα πραγματικά ωφέλη του συστήματος όμως μπορούν να γίνουν ορατά με την ανάπτυξή του σε cloud υποδομές. Σε τέτοιες συνθήκες θα μπορούσαν ευκολότερα να υλοποιηθούν και οι δομές αυτοματοποίησης της κλιμάκωσης του συστήματος, όπου θα είχαν και περισσότερο νόημα.

# Παράρτημα Α΄

## Ακρωνύμια και συντομογραφίες

**LAN** Local Area Network

**REST** Representational State Transfer

**RPC** Remote Procedure Call

**AMQP** Advanced Message Queuing Protocol

**DB** Database

**TCP** Transmission Control Protocol

**TLS** Transport Layer Security

**K8s** Kubernetes

**VM** Virtual Machine

**CLI** Command Line Interface

**NAT** Network Address Translation

**UI** User Interface

**PV** Persistent Volume

**PVC** Persistent Volume Claim

**RBAC** Role-based Access Control

**API** Application Programming Interface

**RMQ** Rabbit MQ

**CoAP** Constrained Application Protocol

**MQTT** Message Queuing Telemetry Transport protocol

**XMPP** Extensible Messaging protocol

**CSRF** Cross-Site Request Forgery

**HTTP** Hyper-Text Transfer Protocol

**SSE** Server Sent Events

**URL** Uniform Resource Locator

**CSS** Cascading Style Sheet

**JSON** JavaScript Object Notation

**DNS** Domain Name System

**IP** Internet Protocol

**CIDR** Classless inter-domain routing

**PV** Persistent Volume

**PVC** Persistent Volume Claim

**IT** Information Technology

**ΒΔ** Βάση Δεδομένων

# Bibliography

- [1] J. L. Fernandes, I. C. Lopes, J. J. P. C. Rodrigues, and S. Ullah, "Performance evaluation of restful web services and amqp protocol," in *2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2013, pp. 810–815.
- [2] N. Murthy. REST, RPC, and brokered messaging. [Online]. Available: <https://medium.com/@natemurthy/rest-rpc-and-brokered-messaging-b775aeb0db3>
- [3] S. E. P. Hernández, E. L. Domínguez, and G. R. Gómez, "An efficient -causal distributed algorithm for synchronous cooperative systems in unreliable networks," vol. 14, no. 1, p. 14.
- [4] J. Yongguo, L. Qiang, Q. Changshuai, S. Jian, and L. Qianqian, "Message-oriented middleware: A review," in *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*, 2019, pp. 88–97.
- [5] P. Mell and T. Grance, "The nist definition of cloud computing," 2011-09-28 2011.
- [6] S. Compass. The 2021 state of cloud adoption. [Online]. Available: <https://resources.securitycompass.com/reports/2021-state-of-cloud-adoption>
- [7] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," in *2017 IEEE International Systems Engineering Symposium (ISSE)*. IEEE, pp. 1–7. [Online]. Available: <http://ieeexplore.ieee.org/document/8088251/>
- [8] H. Alneamy and Z. Alisa, "Survey on IoT application layer protocols," publisher: Unpublished. [Online]. Available: <http://rgdoi.net/10.13140/RG.2.2.11387.85283>
- [9] P. Sommer, F. Schellroth, M. Fischer, and J. Schlechtendahl, "Message-oriented middleware for industrial production systems," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, 2018, pp. 1217–1223.
- [10] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni, "A comparative evaluation of amqp and mqtt protocols over unstable and mobile networks," in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, 2015, pp. 931–936.
- [11] OASIS advanced message queuing protocol (AMQP) version 1.0, part 0: Overview. [Online]. Available: <https://docs.oasis-open.org/amqp/core/v1.0/amqp-core-overview-v1.0.html>

- [12] Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong, and Hongtaek Ju, "Correlation analysis of MQTT loss and delay according to QoS level," in *The International Conference on Information Networking 2013 (ICOIN)*. IEEE, pp. 714–717. [Online]. Available: <http://ieeexplore.ieee.org/document/6496715/>
- [13] V. M. Ionescu, "The analysis of the performance of rabbitmq and activemq," in *2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER)*, 2015, pp. 132–137.
- [14] T. Johansson, "Message-oriented middleware as a queue management solution to improve job handling within an e-commerce system," Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2018.
- [15] M. Albano, L. L. Ferreira, L. M. Pinho, and A. R. Alkhawaja, "Message-oriented middleware for smart grids," vol. 38, pp. 133–143. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0920548914000804>
- [16] I.-C. Donca, C. Corches, O. Stan, and L. Miclea, "Autoscaled RabbitMQ kubernetes cluster on single-board computers," in *2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*. IEEE, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/9129886/>
- [17] Y. Martínez, C. Cachero, and S. Meliá, "MDD vs. traditional software development: A practitioner's subjective perspective," vol. 55, no. 2, pp. 189–200. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0950584912001309>
- [18] T. Bailey, R. Vogel, and J. Mansell, "From code centric to model centric software engineering: Practices, implications and ROI," p. 38.
- [19] Domingo, J. Echeverría, Pastor, and C. Cetina, "Evaluating the benefits of model-driven development," in *Advanced Information Systems Engineering*, S. Dustdar, E. Yu, C. Salinesi, D. Rieu, and V. Pant, Eds. Springer International Publishing, pp. 353–367.
- [20] Gartner, Inc. Gartner forecasts worldwide low-code development technologies market to grow 23% in 2021. Gartner, Inc. [Online]. Available: <https://www.gartner.com/en/newsroom/press-releases/2021-02-15-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-23-percent-in-2021>
- [21] M. Woo, "The rise of no/low code software development—no experience needed?" vol. 6, no. 9, pp. 960–961. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7361109/>
- [22] Managing resources for containers. Section: docs. [Online]. Available: <https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/>
- [23] . Michael Klishin, Staff Software Engineer Pivotal/RabbitMQ. Interpretation of rmq message stats. [Online]. Available: <https://groups.google.com/g/rabbitmq-users/c/syUd9f499Ik/m/7OHMlkLiDgAJ>