
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

Διπλωματική εργασία

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Πανεπιστήμιο Δυτικής Αττικής

**Χρήστος Γιακουμεττής, Ph.D**

**Ιούλιος 2021**

# Σχεδιασμός και ανάπτυξη πλατφόρμας αποστολής μαζικών και εξατομικευμένων ειδοποιήσεων, δεδομένων και μηνυμάτων (web push notifications) σε συσκευές διασυνδεμένες στο διαδίκτυο.

**Επιβλέπων καθηγητής: Δρ. Γεώργιος Μιαούλης**

**Δήλωση Συγγραφέα Διπλωματικής Εργασίας**

Ο κάτωθι υπογεγραμμένος Χρήστος Γιακουμεττής του Παναγιώτη, με αριθμό μητρώου 18390281 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

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

Ο Δηλών

Χρήστος Γιακουμεττής

Athanasios Voulodimos
Digitally signed by Athanasios Voulodimos
Date: 2021.07.28 14:23:58 +03'00'

Georgios Bardis
Digitally signed by Georgios Bardis
Date: 2021.07.29 15:17:10 +03'00'

Georgios Miaoulis
Digitally signed by Georgios Miaoulis
DN: c=GR, l=Athens, o=University of West Attica, ou=Class B - Private Key created and stored in software CSP, sn=Miaoulis, givenName=Georgios, serialNumber=3179983822, cn=Georgios Miaoulis, email=gmiaoul@uniwa.gr
Date: 2021.07.28 06:12:26 +03'00'

# Ευχαριστίες

Αυτή η εργασία αποτελεί τη διπλωματική μου εργασία για το πανεπιστήμιο Δυτικής Αττικής στο Προπτυχιακό Πρόγραμμα Σπουδών Μηχανικών Πληροφορικής και Υπολογιστών.

Το αντικείμενο που πραγματεύεται η παρούσα διπλωματική εργασία έχει τίτλο «Σχεδιασμός και ανάπτυξη πλατφόρμας αποστολής μαζικών και εξατομικευμένων ειδοποιήσεων, δεδομένων και μηνυμάτων (web push notifications) σε συσκευές διασυνδεμένες στο διαδίκτυο».

Θα ήθελα αρχικά να ευχαριστήσω τον υπεύθυνο καθηγητή για την βοήθεια και την υποστήριξη του όπως επίσης και όλους όσους με υποστήριξαν στην προσπάθεια αυτή.

Πιο συγκεκριμένα

Θα ήθελα να ευχαριστήσω τον Δρ. Γεώργιο Μιαούλη, Καθηγητή στο Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών, του Πανεπιστημίου Δυτικής Αττικής για την υποστήριξη και βοήθεια του στην εργασία αυτή.

Επιπροσθέτως, θα ήθελα να ευχαριστήσω τη σύζυγο μου Γαρυφαλλιά Αγουρίδη και τα παιδιά μας, Στέλλα και Παναγιώτη για τη κατανόηση και συμπαράσταση που έδειξαν κατά τη διάρκεια της εκπόνησης αυτής της διπλωματικής εργασίας.

# Περιεχόμενα

## Πίνακας περιεχομένων

# Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται την ανάλυση απαιτήσεων, το σχεδιασμό και ανάπτυξη ενός πληροφοριακού συστήματος μαζικής αποστολής προωθημένων μηνυμάτων (web push notifications). Σχεδιάστηκε ένα ολοκληρωμένο πληροφοριακό σύστημα αποστολής μαζικών προωθημένων μηνυμάτων (web push notifications). Το όλο σύστημα σχεδιάστηκε με γνώμονα την ευκολία χρήσης και εγκατάστασης από την πλευρά των διαχειριστών ιστοτόπων. Παράλληλα δόθηκε έμφαση στο σχεδιασμό προκειμένου να προκύψει ένα ολοκληρωμένο σύστημα που να μπορεί κλιμακωθεί εύκολα και να εκτελείται σε πολλαπλούς εξυπηρετητές σε μελλοντική επέκταση του έργου.

Το όλο έργο αποτελείται από τρεις λειτουργικές μονάδες. Το πρώτο μέρος του συστήματος που αποτελεί και τον πυρήνα του συστήματος αποτελείται από ένα ιδιωτικό API, το οποίο υλοποιήθηκε σε περιβάλλον εργασίας Slim 4.0. Παράλληλα με το API σχεδιάστηκε και ένας service worker και οι απαραίτητες συνοδευτικές συναρτήσεις του (SDK), προκειμένου να μπορεί να εκτελεστεί η υπηρεσία σε έναν ιστότοπο και να μπορεί να λάβει ο τελικός χρήστης ένα προωθημένο μήνυμα. Τέλος αναπτύχθηκε η βασική δομή ενός περιβάλλοντος αποστολής προωθημένων μηνυμάτων που απευθύνεται σε διαχειριστές ιστοτόπων σε περιβάλλον εργασίας Slim 4.0. Και οι τρεις επιμέρους λειτουργικές μονάδες αναπτύχθηκαν χρησιμοποιώντας το ιδεατό περιβάλλον υπηρεσιών σε containers του Docker.

**Λέξεις κλειδιά:** web push notification, distributed application, service worker, push API, API

# Abstract

This work is focusing on the analysis, design and implementation of a web push notification system. During this work a new web push notifications information system was designed. The whole system was designed having in mind the usability and the user easy experience of using it or integrated it to any third part website. At the same time, this work was focused an architecture and system design in order to be able to be scaled up and run on multiple servers in a future extension of the work.

The whole project consists of three functional components. The first part of the system, which is the main core of the whole system, is been consisted by a private Application Programming Interface (API), which was implemented in a Slim 4.0 framework. Along with the API, a service worker and its necessary functions (sdk) were designed, so that the service can be executed on any third-party website and the end user can receive a push notification message. The whole platform, was developed under docker virtual services within containers. Finally, the basic structure of the last function component of the proposed platform was implemented on Slim 4.0 framework too. The last is dealing with an interface of platform that allow web administrators to use and sent massive push notification messages to all their subscribers.

**Keywords:** web push notification, distributed application, service worker, push API, API

# Λίστα φωτογραφιών

# Λίστα πινάκων

# 1 Εισαγωγή

Τα τελευταία δυο χρόνια έχουν γίνει πολλές αλλαγές στον τρόπο που επικοινωνούμε και εργαζόμαστε. Η πανδημία του covid-19 επέβαλε νέους τρόπους επικοινωνίας. Αρκετές εργασίες που γίνονταν ή απαιτούσε παλιότερα με τη φυσική παρουσία των εμπλεκόμενων ατόμων, γίνεται πλέον με ηλεκτρονικό τρόπο και απομακρυσμένα με ασφάλεια, μέσω διαδικτύου.

Η νέα αυτή σύγχρονη πραγματικότητα από την πλευρά των παρόχων ψηφιακών υπηρεσιών δημιουργεί ευκαιρίες ανάπτυξης νέων ψηφιακών υπηρεσιών και προϊόντων, ενώ από την πλευρά του καταναλωτή, δίνει τη δυνατότητα να αναζητήσει και να βρει νέες ηλεκτρονικές υπηρεσίες που ταιριάζουν στα μέτρα.

Τόσο οι ίδιοι οι πάροχοι υπηρεσιών και προϊόντων όσο και οι ίδιοι οι καταναλωτές ψάχνουν και χρησιμοποιούν διάφορα μέσα ενημέρωσης και αναζήτησης πληροφοριών προκειμένου να βοηθηθούν είτε να προωθήσουν, είτε να αναζητήσουν προϊόντα και υπηρεσίες προς κατανάλωση. Το μέσο επικοινωνίας που χρησιμοποιείται είναι το διαδίκτυο και πιο συγκεκριμένα μια διάφορα εργαλεία προώθησης και επικοινωνίας.

## 1.1 Εισαγωγή

Το πιο διαδεδομένο μέσω επικοινωνίας αλλά και το αρχαιότερο είναι το ηλεκτρονικό ταχυδρομείο που χρονολογείται από την πρώτη περίοδο λειτουργίας του διαδικτύου [10]. Όλοι γνωρίζουμε και χρησιμοποιούμε το ηλεκτρονικό ταχυδρομείο. Αποτελεί πλέον αναπόσπαστο κομμάτι της καθημερινότητας μας και αναγκαίο εργαλείο. Καθημερινά δεχόμαστε επαγγελματικά αλλά και διαφημιστικά/προωθητικά μηνύματα ηλεκτρονικού ταχυδρομείου. Αποτελεί έναν ασύγχρονο τρόπο επικοινωνίας.

Παράλληλα με το μηνύματα ηλεκτρονικού ταχυδρομείου, έκανε την εμφάνιση της μια νέα τεχνολογία αποστολής μηνυμάτων μέσω του πρωτοκόλλου προωθημένης επικοινωνίας διαδικτύου (web push protocol). Πρόκειται για προωθημένα μηνύματα ειδοποιήσεων μέσω διαδικτύου (web push notifications) που επιτρέπουν την αποστολή μηνυμάτων μικρού μήκους σε διασυνδεμένες συσκευές μέσω διαδικτύου. Τα προωθημένα αυτά μηνύματα μπορούν να περιέχουν δεδομένα ή κείμενο συνοδευόμενο από μια φωτογραφία και ένα σύνδεσμο. Δεν απαιτείται ειδικό λογισμικό για την λήψη τους και δεν εσωκλείουν προσωπικά δεδομένα. Η επικοινωνία επιτυγχάνεται μέσω του

περιηγητή της εκάστοτε συσκευής. Το δυσκολότερο μέρος στα προωθημένα μηνύματα ειδοποιήσεων είναι η διαχείριση και αποστολή των νέων μηνυμάτων στους αποδέκτες ή ακόμα και η εξατομικευμένη αποστολή μηνύματος σε ένα αποδέκτη.

Η παρούσα εργασία έρχεται να παρουσιάσει μια αρχιτεκτονική και υλοποίηση ενός πληροφοριακού συστήματος που θα διαχειρίζεται εγγεγραμμένους αποδέκτες, θα συνθέτει προωθημένα μηνύματα και θα τα προωθεί σε ομάδες ή μεμονωμένα άτομα, παρέχοντας εξατομικευμένη ενημέρωση και αποστολή δεδομένων.

## 1.2 Αντικείμενο διπλωματικής

Η παρούσα διπλωματική εργασία πραγματεύεται το σχεδιασμό, ανάλυση και ανάπτυξη ενός νέου πληροφοριακού συστήματος διαχείρισης εγγεγραμμένων συσκευών (περιηγητών ανά συσκευή για την ακρίβεια) σε υπηρεσία αποστολής προωθημένων μηνυμάτων, παράλληλα με τη σύνθεση και αποστολή νεών μηνυμάτων στις εγγεγραμμένες αυτές συσκευές (περιηγητές). Ιδιαίτερη έμφαση θα δοθεί στο σχεδιασμό του συστήματος προκειμένου να μπορεί να κλιμακωθεί και να εξυπηρετεί από πολλαπλά σημεία σε μελλοντική επέκταση του συστήματος.

## 1.3 Οργάνωση υλικού διπλωματικής

Στα αρχικά κεφάλαια θα παρουσιαστεί ο τρόπος λειτουργίας του πρωτοκόλλου προώθησης μηνυμάτων μέσω διαδικτύου (web push). Στο κεφάλαιο 2 θα παρουσιαστούν τα βασικά στοιχεία των συστημάτων αποστολής προωθημένων μηνυμάτων. Στη συνέχεια, στο κεφάλαιο 3, θα παρουσιαστούν οι σημαντικότερες σχετικές υλοποιήσεις και αντίστοιχες τεχνικές αποστολής μαζικών και εξατομικευμένων μηνυμάτων μέσω διαδικτύου. Στο κεφάλαιο 4, θα παρουσιαστεί η συμβολή αλλά και οι στόχοι της παρούσας διπλωματικής εργασίας. Από το κεφάλαιο 5 και έπειτα παρουσιάζεται η προτεινόμενη πλατφόρμα.

Πιο συγκεκριμένα στο κεφάλαιο 5 παρουσιάζεται η αρχιτεκτονική και η λειτουργία του συστήματος. Συνεχίζοντας, στο κεφάλαιο 6 παρουσιάζεται το περιβάλλον εργασίας και απαραίτητες υπηρεσίες που απαιτούνται από κάθε λειτουργική μονάδα του πληροφοριακού συστήματος. Στο κεφάλαιο 7, η ίδια η υλοποίηση, αναλύοντας την κάθε λειτουργική μονάδα του συστήματος. Εν κατακλείδι στο τελευταίο κεφάλαιο, παρουσιάζονται γίνεται λόγος για τα συμπεράσματα και τους στόχους που εκπληρώθηκαν σε αυτή τη διπλωματική εργασία

Τέλος, η εργασία συνοδεύεται με τρία παραρτήματα. Κάθε παράρτημα περιέχει τα αρχεία ρυθμίσεων και διαμόρφωσης του ιδεατού περιβάλλοντος ανάπτυξης των λειτουργικών μονάδων του πληροφοριακού συστήματος. Ο ίδιος ο πηγαίος κώδικας του API και του περιβάλλοντος διαχείρισης μηνυμάτων δε μπορεί να παρατεθεί στο παράρτημα καθώς είναι εκτενής. Από την άλλη πλευρά ο πηγαίος κώδικας του service worker και των συνοδευτικών συναρτήσεων είναι μικρός και παρατίθεται στο παράτημα Β.

# 2 Θεωρητικό υπόβαθρο

## 2.1 Μηχανισμός αποστολής προωθημένων μηνυμάτων/ειδοποιήσεων

Ο μηχανισμός αποστολής προωθημένων μηνυμάτων συνήθως χρησιμοποιείται για την αποστολή μαζικών μηνυμάτων σε ανώνυμους αποδέκτες ή σε ειδικές περιπτώσεις σε εξατομικευμένα μηνύματα ή ειδοποιήσεις. Το ίδιο το πρωτόκολλο αποστολής προωθημένων μηνυμάτων δεν μπορεί να φιλοξενήσει προσωπικά στοιχεία χρηστών, για το λόγο αυτό η αποστολή εξατομικευμένων μηνυμάτων επιτυγχάνεται μόνο με ειδικά διαμορφωμένες υλοποιήσεις όπου τηρούνται και επιπλέον προσωπικά στοιχεία. Σε αυτή την περίπτωση δύναται το σύστημα αποστολής να γνωρίζει σε ποιον αποδέκτη θα να σταλεί το μήνυμα βάσει κριτηρίων επιλογής.

Για την αποστολή των μηνυμάτων/ειδοποιήσεων απαιτούνται τρεις διαφορετικές λειτουργικές μονάδες προκειμένου να δημιουργηθεί, να αποσταλεί και τέλος να παραληφθεί και να εμφανιστεί ή να αξιοποιηθεί το μήνυμα από την πλευρά του τελικού αποδέκτη. Σε αντίθεση με το παραδοσιακό ηλεκτρονική ταχυδρομείο οι τρεις αυτές λειτουργικές μονάδες συνήθως είναι διακριτές με προκαθορισμένους ρόλους.

Η πρώτη λειτουργική μονάδα έχει το ρόλο της σύνθεσης ενός προωθημένου μηνύματος/ειδοποίησης. Το ρόλο αυτό, συνήθως, τον έχει ένας εξυπηρετητής μια υπηρεσίας ή μια κεντρική πλατφόρμας αποστολής μαζικών μηνυμάτων.

Στη συνέχεια, τη σκυτάλη αναλαμβάνει το οικοσύστημα του λειτουργικού συστήματος ή του περιηγητή προκειμένου να προωθήσει το μήνυμα στους σωστούς αποδέκτες.

Τέλος το ρόλο του αποδέκτη αναλαμβάνει κάποιος περιηγητής για λογαριασμό ενός ιστότοπου ή κάποια εφαρμογή. Η πλευρά του αποδέχτη είναι υπεύθυνη για τη διαχείριση του ληφθέντος μηνύματος. Δύναται, οι ληφθέντες πληροφορίες και δεδομένα να αξιοποιηθούν από την εφαρμογή ή τον ιστότοπο ή να εμφανιστούν σαν αναδυόμενο παράθυρο ειδοποίησης εντός του λειτουργικού συστήματος. Στην περίπτωση του περιηγητή η λήψη, διαχείριση αλλά και η εμφάνιση των προωθημένων μηνυμάτων ως ειδοποίηση γίνεται μέσω ενός javascript που εκτελείται συνεχώς και ονομάζεται service worker. Στην περίπτωση που το ρόλο του αποδέκτη τον έχει μια εφαρμογή, η ίδια είναι υπεύθυνη για την διαχείριση των ληφθέντων δεδομένων.

Αξίζει να σημειώσουμε ότι η επιλογή των προωθημένων μηνυμάτων που εσωκλείουν δεδομένα είναι μονόδρομος, για την επικοινωνία μεταξύ ενός εξυπηρετητή και μιας εφαρμογής που εκτελείται σε φορητές συσκευές. Σε αυτή την περίπτωση μπορεί ο εξυπηρετητής να εκκινήσει επικοινωνία με

μια εφαρμογή χωρίς να γνωρίζει αν η συσκευή είναι συνδεδεμένη εκείνη τη στιγμή στο διαδίκτυο ή την ip διεύθυνση της της εν λόγω συσκευής. Ο παραλήπτης-εφαρμογή θα λάβει το μήνυμα και τα δεδομένα που εμπεριέχει, μόλις συνδεθεί στο διαδίκτυο. Στην ακόλουθη παράγραφο παρουσιάζεται συνοπτικά ο τρόπος λειτουργίας ενός τυπικού συστήματος αποστολής προωθημένων μηνυμάτων

## 2.2 Αρχιτεκτονική τυπικού πληροφοριακού συστήματος αποστολής προωθημένων μηνυμάτων

Ένα τυπικό απλό πληροφοριακό σύστημα αποστολής προωθημένων μηνυμάτων αποτελείται από τέσσερις λειτουργικές μονάδες με διακριτό ρόλο η κάθε μια. Η πρώτη λειτουργικό μονάδα έχει το ρόλο του εξυπηρετητή και είναι υπεύθυνη για τη διαχείριση των εγγεγραμμένων συσκευών και επομένως των αποδεκτών των μηνυμάτων. Η δεύτερη λειτουργική μονάδα αφορά την ίδια την αποστολή του μηνύματος. Η Τρίτη λειτουργική μονάδα είναι υπεύθυνη για τη σύνταξη των προωθημένων μηνυμάτων αλλά και την επιλογή των αποδεκτών στους οποίους θα σταλεί το προωθημένο μήνυμα.

Τέλος η τελευταία λειτουργική μονάδα αφορά το λογισμικό από την πλευρά του τελικού χρήστη που είναι υπεύθυνο για την προβολή του μηνύματος και την όποια απαραίτητη αλληλεπίδραση του χρήστη με την πλατφόρμα προωθημένων μηνυμάτων μπορεί να γίνει.

Τέλος η τελευταία λειτουργική μονάδα είναι υπεύθυνη για τη σύνταξη

**Εξυπηρετητής διαχείρισης εγγεγραμμένων συσκευών**

Η λειτουργική αυτή μονάδα είναι υπεύθυνη για τη διατήρηση των εγγεγραμμένων συσκευών σε λίστες προκειμένου να χρησιμοποιηθούν αργότερα ως αποδέκτες των μηνυμάτων. Η δυο βασικές λειτουργίες που εκτελούν είναι η εγγραφή και απεγγραφή μιας συσκευής. Παράλληλα με αυτές τις δυο διαδικασίες δίνεται να μπορεί να οργανώσει τις εγγεγραμμένες συσκευές σε διακριτές ομάδες/ λίστες προκειμένου να σταλθούν ομαδοποιημένα και στοχευμένα μηνύματα στους αποδέκτες.

### 2.2.1 Λειτουργική μονάδα αποστολής προωθημένων μηνυμάτων

Η λειτουργική μονάδα αυτή είναι υπεύθυνη για την αποστολή του ίδιου του μηνύματος σε έναν αποδέκτη ή πολλούς αποδέκτες. Ο ρόλος αυτής της μονάδας δύναται να τον έχουν ένα από τα παρακάτω συστήματα, όπου και το καθένα απευθύνεται και στο δικό του κλειστό οικοσύστημα εφαρμογών και δύναται να είναι ένα από τα ακόλουθα [12].

- Google Cloud Messaging (GCM) / Firebase Cloud Messaging (FCM)
- Apple Push Notification service (APNs)

- Windows Push Notification Services (WNS)
- Mozilla Push Service
- Amazon Device Messaging (ADM) / Amazon Simple Notification Service (SNS)

Η λειτουργική μονάδα αποστολής προωθημένων μηνυμάτων δέχεται σαν ορίσματα τον μήνυμα τον αποστολέα όπως ακριβώς γίνεται και με το μήνυμα ηλεκτρονικής αλληλογραφίας. Κάθε μήνυμα δύναται να έχει ένα τίτλο, ένα εικονίδιο, ένα σύντομο κείμενο (κυρίως σώμα μηνύματος), μια συνοδευόμενη εικόνα και ένα υπερσύνδεσμο. Σε αυτό το σημείο πρέπει να αναφέρουμε ότι ο κάθε παραλήπτης αναφέρεται καταχρηστικά στη βιβλιογραφία αλλά και στα τεχνικά εγχειρίδια ως device_id (αναγνωριστικός αριθμός συσκευής) καθώς αντιπροσωπεύει μόνο την εν λόγω εφαρμογή ή τον περιηγητή έχει κάνει το αίτημα εγγραφής στην υπηρεσία. Αυτό σημαίνει ότι σε έναν ηλεκτρονικό υπολογιστή ή ένα φορητή συσκευή που έχουμε δυο διαφορετικούς περιηγητές τότε για το ίδιο φυσικό μηχάνημα (υπολογιστή) θα έχουμε δυο διαφορετικά device ids (αναγνωστικά νούμερα συσκευής).

### 2.2.2 Συντάκτης προωθημένων μηνυμάτων και επιλογή αποδεκτών

Ο συντάκτης προωθημένων μηνυμάτων κάνει αυτό ακριβώς που αναφέρει ο τίτλος του. Αφορά το μέρος λογισμικού που είναι υπεύθυνο για τη συγγραφή του προωθημένου μηνύματος, το χρόνο αποστολής του, το χρόνο ζωής του – επιτρεπτός χρόνος παράδοσης – εάν παρέλθει αυτό το χρονικό ορόσημο το μήνυμα χάνεται και δεν πάει στον αποδέκτη του. Τέλος σημαντική λειτουργία αυτής της λειτουργική μονάδας δίνοντας προστιθέμενη αξία στην υπηρεσία είναι αν υπάρχει δυνατότητα συνδυασμού και συνυπολογισμού επιπλέον ανώνυμης πληροφορίας για κάθε χρήστη εγγεγραμμένης συσκευής προκειμένου να επιλεγεί από μια λίστα βάσει κριτηρίων προκειμένου να σταλθούν στοχευμένα και προσωποιημένα προωθημένα μηνύματα.

### 2.2.3 Λογισμικό τελικού χρήστη (περιηγητής)

Το ρόλο αυτό έχει συνήθως ο περιηγητής όπου αναλαμβάνει να λάβει και να προβάλει τα εισερχόμενα προωθημένα μηνύματα στον τελικό χρήστη. Στην περίπτωση που το λογισμικό λήψης του μηνύματος είναι κάποια εφαρμογή, εναπόκειται στην τελευταία το πως θα χειριστεί το μήνυμα και τα δεδομένα του, αν θα το εμφανίσει κάποιο μήνυμα στον τελικό χρήστη ή θα αξιοποιήσει τα δεδομένα με άλλο τρόπο. Στην περίπτωση του περιηγητή το λόγο της διαχείρισης του προωθημένων μηνυμάτων αναλαμβάνει ένα τμήμα κώδικα γραμμένο σε javascript που έχει τη δυνατότητα να λειτουργεί ως υπηρεσία εντός του λειτουργικού συστήματος και για λογαριασμό συγκεκριμένου ιστότοπου και ονομάζεται service worker.

## 2.3 Πρωτόκολλο προωθημένων μηνυμάτων (web push protocol)

Το πρωτόκολλο προωθημένων μηνυμάτων (web push protocol) περιγράφει τον τρόπο επικοινωνίας μεταξύ ενός παρόχου και μιας υπηρεσίας προωθημένων μηνυμάτων. Το εν λόγω πρωτόκολλο βασίζεται στο W3C Push API μια πρώτη έκδοση του οποίου παρουσιάστηκε για πρώτη φορά τον Οκτώβριο του 2012 [5], [9]. Με την πάροδο του χρόνου και την παρουσίαση και του service worker το 2014, το πρωτόκολλο web push εξελίχθηκε μέχρι που έφτασε στη σημερινή του μορφή, όπου και αποτελεί ένα πρωτόκολλο σύγχρονής μετάδοσης δεδομένων [6].

## 2.4 Προωθημένα μηνύματα (push notification)

Τα προωθημένα μηνύματα διαδικτύου είναι δεδομένα και πληροφορίες που μεταδίδονται στις συσκευές των χρηστών και δύναται να εμφανίζονται σαν αναδυόμενα παράθυρα στο στις συσκευές των χρηστών. Τα προωθημένα μηνύματα διακρίνονται σε δυο κατηγορίες. Αυτά που εμφανίζονται στο χρήστη σαν αναδυόμενα παράθυρα εσωκλείοντας τα δεδομένα και την πληροφορία και σε αυτά τα μηνύματα που λαμβάνονται από τον τελικό χρήστη και πρέπει η αρμόδια εφαρμογή να τα λάβει και να τα διαχειριστεί. Μπορεί να εμφανιστούν στην οθόνη σαν αναδυόμενο παράθυρο ή να συλλεχθούν τα δεδομένα χωρίς να γίνει περαιτέρω ενημέρωση του τελικού χρήστη.

Ειδικότερα για τις φορητές συσκευές, ο ίδιος μηχανισμός μπορεί να χρησιμοποιηθεί για την αποστολή δεδομένων και πληροφοριών από κάποιο κεντρικό σημείο προς τους χρήστες της εκάστοτε εφαρμογής προκειμένου να επιτευχθεί μεταφορά δεδομένων προς τον τελικό χρήστη. Σε αυτή την περίπτωση η έναρξη / αίτημα αποστολής δεδομένων μπορεί να ξεκινήσει από ένα κεντρικό σημείο προς τη συσκευή του τελικού χρήστη.

Γενικά τα προωθημένα αυτά μηνύματα απαιτούν ειδική μεταχείριση από την πλευρά του αποδέκτη προκειμένου να εμφανιστούν σαν αναδυόμενα μηνύματα στις συσκευές των χρηστών (υπολογιστές, ταμπλέτες, κινητά) ή να αξιοποιηθούν από τον παραλήπτη με άλλο τρόπο. Σημαντικό πλεονέκτημα των προωθημένων μηνυμάτων είναι ότι μπορούν να ληφθούν ακόμα και σε χρονικές στιγμές που η αρμόδια εφαρμογή, εάν υπάρχει, που διαχειρίζεται τα μηνύματα αυτά δεν είναι ενεργεί στη συσκευή του χρήστη.

### 2.4.1 Χαρακτηριστικά – δομή προωθημένου μηνύματος

Η δομή και η μηχανογράφηση ενός προωθημένου μηνύματος όπως και τα μηνύματα ηλεκτρονικής αλληλογραφίας έχουν μια σειρά από υποχρεωτικά και από προαιρετικά πεδία. Στη συνέχεια αυτής της παραγράφου θα παρουσιαστούν μόνο το κυριότερα πεδία ενός προωθημένου

μηνύματος που συναντάμε στα συνηθέστερα οικοσυστήματα. Υπάρχουν κάποια πεδία και λειτουργίες που είχε σχεδιασμένες μόνο για φορητές συσκευές. Τα πεδία ενός προωθημένου μηνύματος παρουσιάζονται τον ακόλουθο Πίνακας 1.

| **Πεδίο / δεδομένα** | **Περιγραφή** | **Συσκευές** |
|---|---|---|
| **Title** | Τίτλος του μηνύματος | όλες |
| **Body** | το κυρίως κείμενου του μηνύματος | όλες |
| **Notification Icon** | συνοδευτικό εικονίδιο μηνύματος | όλες |
| **Notification Image** | φωτογραφία μηνύματος | όλες |
| **Action Buttons** | Κουπιά ανάδρασης κάτω από το σώμα του μηνύματος | όλες |
| **Language Direction** | κατεύθυνση κειμένου | όλες |
| **Tag** | Ομαδοποίηση μηνυμάτων με tags | όλες |
| **Renotify** | Ειδοποίηση χρήση με δόνηση και ήχο σε περίπτωση λήψης μηνύματος που ανήκει σε ίδια ομάδα (έχει χαρακτηριστεί με το ίδιο tag) με ήδη προηγούμενο ληφθέν μήνυμα. | όλες |
| **Silent** | Εάν χρησιμοποιηθεί δε θα αναπαραχθεί ήχος ή δόνηση ούτε θα ενεργοποιηθεί η οθόνη | φορητές συσκευές |
| **Require Interaction** | Εάν ενεργοποιηθεί απαιτεί την αλληλεπίδραση του χρήστη για να κλείσει | φορητές συσκευές |
| **Vibration** | Ρυθμίζει τη δόνηση που θα αναπαραχθεί με τη λήψη του | φορητές συσκευές |
| **Timestamp** | εμφανίζει τη χρονική διαφορά της στιγμή της λήψης και μιας άλλης χρονικής στιγμής αναφοράς (υπολειπόμενος χρόνος ή χρόνος που παρήλθε από κάποιο συμβάν-χρονικό σημείο αναφοράς) | όλες |
| **Sound** | αναπαραγωγή ήχου | φορητές συσκευές |
| **URL** | υπερσύνδεσμος που θα ανοίξει με το πάτημα στον τίτλο/σώμα/φωτογραφία του μηνύματος | όλες |
| **requireInteraction** | στην περίπτωση των φορητών συσκευών παραμένει η ειδοποίηση ενεργεί μέχρι να λάβει την ανάδραση του χρήστη | φορητές συσκευές |

| | | |
|---|---|---|
| **Custom Data** | συνοδευτικά δεδομένα οποιασδήποτε μορφής για αξιοποίηση από τον αποδέκτη | όλες |

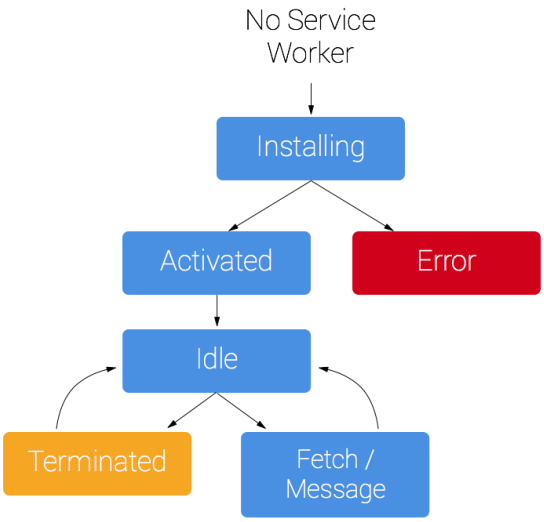
Πίνακας 1 – πεδία / δομή ενός προωθημένου μηνύματος

## 2.5 Service Worker

Ο service worker είναι κώδικας script γραμμένος σε javascript οποίος εγκαθίσταται στον εκάστοτε περιηγητή. Ο service worker παρουσιάστηκε για πρώτη φορά το 2014, είναι πρόσφατη μορφή κώδικα script που μπορεί να εκτελείται στις τελευταίες εκδόσεις των γνωστών περιηγητών [7]. Πρώτος το περιηγητής που το υιοθέτησε ήταν ο Google Chrome (έκδοση για υπολογιστές) το 3$^{ο}$ τετράμηνο του 2015. Στη συνέχεια ακολούθησαν και οι υπόλοιποι γνωστοί περιηγητές. Ο service worker εγκαθίσταται στον περιηγητή και μπορεί να διαχειριστεί το περιεχόμενο ενός ιστότοπου από τον οποίο και έχει προέλθει (εγκατασταθεί) και με τη βοήθεια κάποιων συμβάντων που μπορούν να τον αφυπνίσουν.

Ο service worker μπορεί θεωρητικά να εκτελεί οποιαδήποτε διεργασία πάνω στο περιεχόμενο του ιστότοπου που τον αφορά, όπως να τροποποιεί τις εικόνες ενός ιστότοπου και να τις βελτιστοποιεί προτού εμφανιστούν στον περιηγητή και στον τελικό χρήστη. Ο service worker είναι το μέσο που χρησιμοποιείται από έναν περιηγητή προκείμενου να λάβει τη συγκατάθεση του χρήστη ότι επιθυμεί να εγγραφεί στην υπηρεσία προωθημένων μηνυμάτων, την επιλογή του για να απεγγραφεί, προβολή των εισερχόμενων προωθημένων μηνυμάτων ως ειδοποίηση κ.α.

Μια βασική προϋπόθεση για να εγκατασταθεί ένας service worker και αν εκτελεστεί είναι να διαθέτει ένα έγκυρο πιστοποιητικό ασφαλείας και να κάνει χρήση πρωτοκόλλου https ο ιστότοπος από τον οποίο έχει προέλθει. Στην ακόλουθη Εικόνα 1 φαίνονται οι διαφορετικές καταστάσεις που μπορεί να επέλθει ένας service worker από τη στιγμή της έναρξης εγκατάστασης του στον περιηγητή [8].




Εικόνα 1 – καταστάσεις service worker

Ένας service worker για να εγκατασταθεί ή να απεγκατασταθεί θα πρέπει να γίνει σχετικό αίτημα προς τον περιηγητή που μπορεί να γίνει από κώδικα που εσωκλείει ο ίδιος ο service worker. Παράλληλα με κώδικα script η απενεργοποίηση εντός service worker μπορεί να πραγματοποιηθεί και μέσα από τις ρυθμίσεις του περιηγητή. Στην τελευταία περίπτωση ο τερματισμός του service worker είναι προσωρινός καθώς θα επενεγκτασταθεί στον εν λόγω περιηγητή με την επαναφότωση μιας σελίδας του ίδιου ιστόποπου εφόσον ο κώδικας του service worker συνεχίζει να υπαγορεύει την εγκατάσταση του και υπάρχει και η συγκατάθεση του χρήστη.

Για τη λήψη των σχετικών προωθημένων μηνυμάτων ο service worker χρησιμοποιεί το Push API πρωτόκολλο, η πρώτη έκδοση του οποίου παρουσιάστηκε τον Οκτώβριο του 2012 [5].

Όσο ο service worker βρίσκεται στην κατάσταση idle μπορεί να εκτελείται στο παρασκήνιο του λειτουργικού συστήματος ακόμα και αν έχουμε φύγει από τον ιστότοπο που χρησιμοποιεί service worker ή ακόμα και αν έχουμε κλείσει τον περιηγητή αλλά συνεχίζει να εκτελείται σαν διαδικασία στο παρασκήνιο του λειτουργικού συστήματος.

Ο μηχανισμός του service worker αν και αριθμεί μερικά χρόνια ύπαρξης δεν έχει ακόμα υιοθετηθεί και χρησιμοποιηθεί ευρέως από πολλές υλοποιήσεις. Περιορίζεται κυρίως σε υπηρεσίες προωθημένων μηνυμάτων με μεμονωμένες εξαιρέσεις όπως σε υπηρεσία βελτιστοποίησης εικόνων για διαδικτυακή χρήση.

## 2.6 Πως λειτουργεί ο μηχανισμός από την πλευρά του τελικού χρήστη προωθημένων μηνυμάτων

Για να μπορέσει κάποιος να εγγραφεί σε μια υπηρεσία λήψης προωθημένων μηνυμάτων σε έναν ιστότοπο θα πρέπει να χρησιμοποιεί έναν περιηγητή που να μπορεί να τρέξει και να υποστηρίζει την εκτέλεση service worker, πράγμα που συμβαίνει με όλους τους σημερινούς σύγχρονους περιηγητές. Παράλληλα και πολύ σημαντικό είναι να είναι συμβατός με το πρωτόκολλο προωθημένων μηνυμάτων όπως επίσης συμβαίνει με όλους τους σύγχρονους περιηγητές. Για τις ανάγκες της παρούσας εργασίας υλοποιήθηκαν οι ακόλουθες τρεις διαδικασίες από την πλευρά του service worker

- Δ1. Εγγραφή χρήστη στην υπηρεσία
- Δ2. Προβολή εισερχόμενου μηνύματος
- Δ3. Απεγγραφή περιηγητή από την υπηρεσία

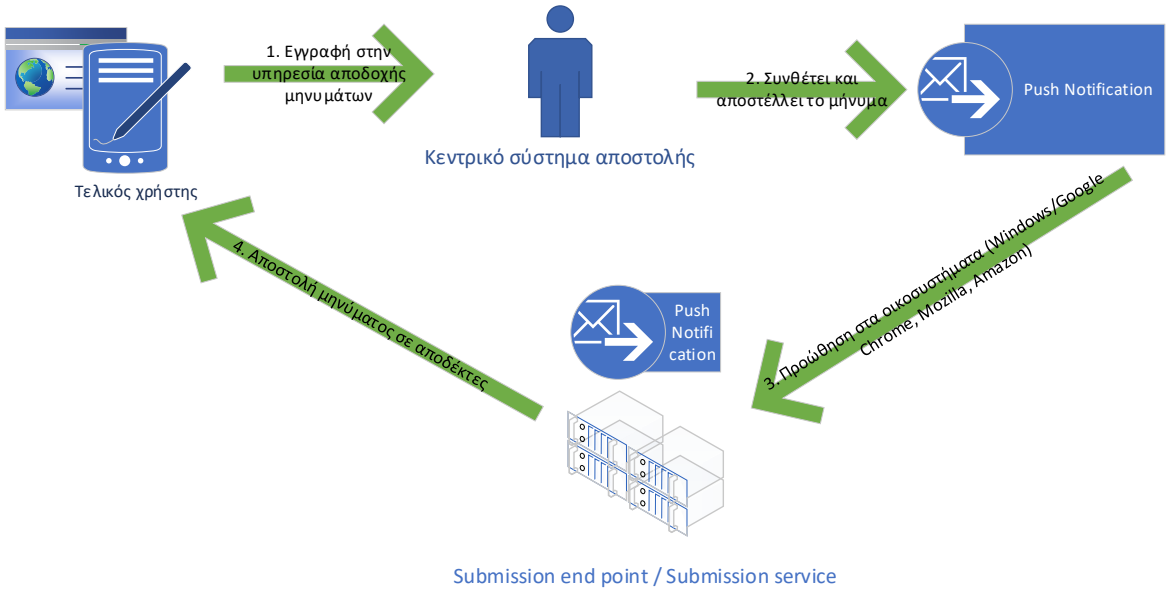
Ένας δυνητικός χρήστης στης υπηρεσίας μόλις μπει για πρώτη φορά στον εν λόγω ιστότοπο θα ειδοποιηθεί εάν θέλει να εγγραφεί στη σχετική υπηρεσία (διαδικασία Δ1. Εγγραφή χρήστη στην υπηρεσία). Παράλληλα θα εγκατασταθεί και ο service worker στον περιηγητή του χρήστη. Με την αποδοχή του χρήστη θα αποσταλεί ένα αίτημα εγγραφής στον εξυπηρετητή διαχείρισης των εγγεγραμμένων συσκευών με το μοναδικό αναγνωριστικό αριθμό του περιηγητή (καταχρηστικά αναφέρεται σαν device id).

Σε δεύτερο χρόνο, όταν σταλεί κάποιο μήνυμα από τη μονάδα αποστολής προωθημένων μηνυμάτων προς τη συγκεκριμένη συσκευή εκτελείται η διαδικασία Δ2. Προβολή εισερχόμενου μηνύματος. Αναλαμβάνει τη διαχείριση του εισερχόμενου μηνύματος ο service worker ο οποίος πρέπει να εμπεριέχει κώδικα για να πυροδοτείται όταν λάβει το σχετικό νέο μήνυμα και με τη σειρά του το εμφανίζει σαν αναδυόμενο παράθυρο ενημέρωσης στον τελικό χρήστη.

Αντίστοιχα η διαδικασία της απεγραφής (διαδικασία Δ3. Απεγγραφή περιηγητή από την υπηρεσία) μπορεί να γίνει είτε κατευθείαν μέσα από τον περιηγητή είτε μέσω του service worker όπου θα επικοινωνήσει με τη σειρά με τη μονάδα διαχείρισης εγγεγραμμένων συσκευών και θα αιτηθεί την απεγγραφή του περιηγητή.

Αναλυτικότερα θα παρουσιαστούν οι εμπλεκόμενες διαδικασίες του πληροφοριακού συστήματος στο κεφάλαιο 5 Αρχιτεκτονική πλατφόρμας.

Στη συνέχεια παρουσιάζεται μια συνοπτική ροή των βημάτων που εκτελούνται στον κύκλο ζωής ενός προωθημένου μηνύματος.




Εικόνα 2 – Ροή βημάτων αποστολής προωθημένου μηνύματος

Στην Εικόνα 2 παρουσιάζονται τα βασικά βήματα προκειμένου να δημιουργηθεί και να αποσταλεί στους αποδέκτες του ένα προωθημένο μήνυμα.

Αρχικά οι χρήστες εγγράφονται μέσω των περιηγητών τους στην υπηρεσία αποδεχόμενοι την λήψη προωθημένων μηνυμάτων. Σε δεύτερο χρόνο και με δικιά του πρωτοβουλία το κεντρικό σύστημα συνθέτει και αποστέλλει ένα προωθημένο μήνυμα. Το μήνυμα μέσω των οικοσυστημάτων των λειτουργικών συστημάτων και των περιηγητών φτάνει τους αποδέκτες του.

# 3 Σχετικές υλοποιήσεις

## 3.1 Σχετικές τεχνολογίες

Η τεχνολογία των προωθημένων μηνυμάτων είναι σχετικά μια νέα υλοποίηση. Βασίζεται στο Push API που χρονίζει περίπου 10 έτη. Τα προωθημένα μηνύματα έχουν περιορισμένη χρήση στους ιστοτόπους, ενώ χρησιμοποιούνται κατά κύριο λόγο ως μέσω επικοινωνίας μεταξύ εξυπηρετητών και εφαρμογών για φορητές συσκευές. Στο κεφάλαιο αυτό θα αναφέρουν οι κυριότερες παρεμφερείς τεχνικές υλοποιήσεις

### 3.1.1 Ηλεκτρονικό ταχυδρομείο

Το πρώτο από όλα και πιο συνηθισμένο εργαλείο αποτελεί το ηλεκτρονικό ταχυδρομείο η πρώτη έκδοση του οποίου χρονολογείται από τις αρχές του 1970 [10]. Αποτελεί τον πιο διαδεδομένο τρόπο αποστολής ηλεκτρονικών μηνυμάτων ενώ αποτελεί και αναπόσπαστό εργαλείο της καθημερινότητας μας πλέον. Δύναται να αποσταλεί κείμενο, πολυμεσικό υλικό και γενικά κάθε είδους τύπου αρχείο μέσω ηλεκτρονική αλληλογραφία. Τα λογισμικά που υποστηρίζουν την λήψη ενός μηνύματος συνήθως υποστηρίζουν και τη σύνθεση και αποστολής ενός νέου μηνύματος. Για τη χρήση της υπηρεσία απαιτούνται δυο εξυπηρετητές. Ένας εξυπηρετητής για την αποστολή του μηνύματος και ένας εξυπηρετητής για τη λήψη του μηνύματος. Ο τελευταίος διατηρεί και το ηλεκτρονικό γραμματοκιβώτιο του χρήστη

### 3.1.2 Υπηρεσίες αποστολής προωθημένων μηνυμάτων (Push notifications services) κατόχους οικοσυστημάτων

Κάθε οικοσύστημα λειτουργικού συστήματος ή περιηγητή έχει αναπτύξει σχετικές υπηρεσίες που βασίζεται στην βασική έκδοση του Push API πρωτοκόλλου προκειμένου να μπορεί να προωθήσει τα σχετικά μηνύματα στους εγγεγραμμένους χρήστες του οικοσυστήματος του. Στις παρακάτω παραγράφους αναφέρονται τα σημαντικότερα εξ αυτών.

- Firebase Cloud Messaging service (FCM)
- Apple push notification service
- Windows push notification service
- Mozilla Push Service
- Amazon cloud messaging service

Τα πρώτα δυο έχουν δημιουργηθεί από τη Google και το Firebase Cloud Messaging έχει αντικαταστήσει το Google Cloud Messaging. Απευθύνεται στους κατόχους φορητών συσκευών με λειτουργικό σύστημα Android και στους χρήστες που χρησιμοποιούν τον περιηγητή Google Chrome.

Η δεύτερη λύση απευθύνεται στο οικοσύστημα της Apple ενώ τα υπόλοιπα απευθύνεται στο εκάστοτε οικοσύστημα της κάθε εταιρείας (λειτουργικό σύστημα και περιηγητές). Τέλος η υλοποίηση της Microsoft και αντίστοιχα της Amazon απευθύνεται στα αντίστοιχα λειτουργικά συστήματα και περιηγητές του εκάστοτε οικοσυστήματος.

Κάθε ένα από τα παραπάνω συστήματα αποτελούν προϊόν εμπορικής πλατφόρμας όπου συγκεντρώνει μεταξύ άλλων και λοιπές υπηρεσίες προστιθέμενης αξίας όπου διατίθενται επί πληρωμή στους συνδρομητές της πλατφόρμας. Παράλληλα με τις επί πληρωμή υπηρεσίες διαθέτουν και μικρά πακέτα υπηρεσιών που διατίθενται δωρεάν.

Κάθε ένα από τα παραπάνω συστήματα χρησιμοποιεί τη βασική έκδοση του πρωτόκολλου προωθημένων μηνυμάτων (web push protocol) το οποίο ανεξάρτητο από το λειτουργικό συστήματα και εξυπηρετητή. Αυτό σημαίνει ότι μπορεί να χρησιμοποιηθεί ένα από τα παραπάνω συστήματα προκειμένου να μπορέσει να στείλει κάποιος ένα προωθημένο μήνυμα σε κάποιον περιηγητή ή εφαρμογή οποιουδήποτε άλλου οικοσυστήματος εφόσον είναι συμβατή με το γενικό πρωτόκολλο προωθημένων μηνυμάτων όπως είναι και οι γνωστοί περιηγητές (internet explorer, google chrome, mozila firefox, safari κ.α.) [1], [2]. Σε αντίθετη περίπτωση θα πρέπει να επιλεγεί η αντίστοιχη επιλογή για το οικοσύστημα που ανήκει η εν λόγω εφαρμογή.

### 3.1.3 Πρωτόκολλο ανάκτησης πληροφορίας (Pull protocol)

Σε αντίθεση με το πρωτόκολλο προωθημένων μηνυμάτων έρχεται το πρωτόκολλο Pull protocol. Πρόκειται για τον βασικό τρόπο επικοινωνίας. Για να γίνει κατανοητό αυτό το πρωτόκολλο επικοινωνίας θα παραθέσουμε ένα παράδειγμα όπου ο ένας ενδιαφερόμενος έχει υποβάλει ένα αίτημα προς διεκπεραίωση που όμως θα εξυπηρετηθεί ετεροχρονισμένα ή θα χρονίσει να ολοκληρωθεί και δε γνωρίζουμε το χρόνο ολοκλήρωσης του. Σύμφωνα με αυτό το μοντέλο επικοινωνίας ο αιτών ζητά να ανακτήσει (Pull) την πληροφορία που τον ενδιαφέρει και να πάρει τα αποτελέσματα. Η εκκίνηση της επικοινωνίας γίνεται πάντα με πρωτοβουλία του αιτούντα και μπορεί να γίνει πολλαπλές φορές, αφού δεν είναι σε θέση να γνωρίζει αν το αρχικό αίτημα του έχει διεκπεραιωθεί. Η επικοινωνία του αιτούνται γίνεται με βάση τα γνωστές τεχνικές ανάκτησης (pull) πληροφορίας (post/get) [4]. Σε κάποιο αίτημα του αιτούνται θα λάβει θετική απάντηση και τα αποτελέσματα του αιτήματος του

## 3.2 Άλλες εμπορικές υπηρεσίες υλοποίησης προωθημένων μηνυμάτων

Μεταξύ άλλων συστημάτων, οι πλέον διαδεδομένες πλατφόρμες αποστολής μαζικών προωθημένων μηνυμάτων σήμερα διατίθενται από τις ακόλουθες τρεις υλοποιήσεις. Μπορείτε για εμπορικές εφαρμογές που μπορεί ο διαχειριστής ενός ιστοτόπου να στείλει μαζικά προωθημένα μηνύματα σε όλους τους αποδέκτες-χρήστες του ιστοτόπου του.

- Onesignal
- Omnisend
- Blaze

Η τελευταία υλοποίηση του Blaze έχει τη δυνατότητα να δημιουργεί ομάδες αποδεκτών και αποστολής μηνυμάτων σε συγκεκριμένους χρήστες δημιουργώντας επιμέρους ομάδες με βάση κάποια κριτήρια που συνοδεύουν το χρήστη. Τέλος, η τελευταία διαθέτει και μηχανισμό διαθέτει και μηχανισμός αποστολής μαζικών μηνυμάτων ηλεκτρονικού ταχυδρομείου σε πιστοποιημένους χρήστες του εκάστοτε ιστότοπου.

## 3.3 Άλλες υλοποιήσεις με χρήση service worker

Κλείνοντας τη σχετικές υλοποιήσεις αξίζει να σημειώσουμε άλλες υλοποιήσεις που χρησιμοποιήσαν τη τεχνική του service worker προκειμένου να παραδώσουν άλλες υπηρεσίες διαφορετικές υπηρεσίες από τα προωθημένα μηνύματα. Μια τέτοια υλοποίηση αποτελεί και η υπηρεσία του IMGHaste όπου μέσω ενός service worker προσφέρει καινοτόμες υπηρεσίες εξατομικευμένης βελτιστοποίησης εικόνων για χρήση διαδικτύου.

### 3.3.1 IMGHaste - Υπηρεσία εξατομικευμένης βελτιστοποίησης εικόνων διαδικτύου

Πρόκειται για μια υπηρεσία βελτιστοποίησης εικόνων μέσω service worker. Η υπηρεσία χρησιμοποιεί έναν service worker προκειμένου να παρακολουθεί όλα τα αιτήματα που κάνει ένας περιηγητής για να ανακτήσει αρχεία φωτογραφιών, αρχείων μορφοποίησης (css), αρχεία κώδικα (javascript) και γραμματοσειρών. Προτού ανακτήσει τα συγκεκριμένα αρχεία τα αποστέλλει στην υπηρεσία του IMGHaste προς βελτιστοποίηση. Αφού βελτιστοποιηθεί το εκάστοτε αρχείο τοποθετείται σε διεθνές Content Delivery Network (CDN) και στη συνέχεια επιστρέφει η υπηρεσία πίσω το βελτιστοποιημένο αρχείο. Ο service worker αναλαμβάνει να εμφανίσει ή να χρησιμοποιήσει το βελτιστοποιημένο αρχείων αντί των αρχικών αρχείων. Ο service worker στέλνει επιπλέον πληροφορίες που είναι διαθέσιμος μόνο κατά την περίοδο του rendering των φωτογραφιών όπως τους συμβατούς τύπους εικόνων του περιηγητή, την ανάλυση της οθόνη, το Device Pixel Ratio και την

ταχύτητα σύνδεσης του χρήστη προκειμένου να προσφέρει πλήρως εξατομικευμένη και βελτιστοποιημένη υπηρεσία.

# 4 Συμβολή, ανάγκη και στόχος διπλωματικής

## 4.1 Κίνητρο

Η παροχή υπηρεσιών και προϊόντων μέσω διαδικτύου επιβάλει τη συνέχει ενημέρωση και επικαιροποίηση των σχετικών πληροφοριών και δεδομένων τόσο από την πλευρά του παρόχου της εκάστοτε υπηρεσίας ή προϊόντος όσο και από την πλευρά του καταναλωτή που οφείλει και πρέπει να είναι ενήμερος για τις εξελίξεις. Η επικοινωνία μέσω διαδικτύου μπορεί να πραγματοποιηθεί με μέσα που χρησιμοποιούν τεχνολογίες σύγχρονης επικοινωνίας όπως είναι οι τηλεδιασκέψεις και οι ζωντανές συνομιλίες. Μέσα σύγχρονης επικοινωνίας προσφέρουν την καλύτερη δυνατή επικοινωνία καθώς δύναται να αντικαταστήσουν επαρκώς ή σε μεγάλο βαθμό τη δια ζώσης φυσική επικοινωνία μεταξύ ατόμων. Ο περιορισμός της σύγχρονης επικοινωνίας είναι ότι ο αριθμός των συμμετεχόντων περιορίζεται σε μερικά μόνο άτομα λόγω της φύσης της επικοινωνίας.

Παράλληλα με τη σύγχρονη επικοινωνία υπάρχουν και τρόποι ασύγχρονης μαζικής επικοινωνίας με πρώτο και καλύτερο το ηλεκτρονικό ταχυδρομείο που εμφανίστηκε και χρησιμοποιείται κατά κόρον από την πρώτη περίοδο λειτουργίας του διαδικτύου [10]. Αποτελεί το πλέον διαδεδομένα πρότυπο αποστολής προωθητικών και μη μηνυμάτων μεταξύ διασυνδεδεμένων χρηστών διαδικτύου. Είναι ένα μέσο που προσδιορίζει μια προσωπική ταυτότητα και ύπαρξη στο χώρο του διαδικτύου για κάθε συνδεδεμένο χρήστη. Θεωρείται ένα από τα στοιχεία επικοινωνίας και ταυτοποίησης χρηστών διαδικτύου. Αποτελεί το μόνο ευρέως διαδεδομένο ασύγχρονο τρόπο επικοινωνίας μέσω διαδικτύου. Μια διεύθυνση ηλεκτρονικής αλληλογραφίας, καθώς συνοδεύει και ταυτοποιεί ή προσδίδει κάποια ιδιότητα σε ένα φυσικό πρόσωπο, έχει σαν αποτέλεσμα μια διεύθυνση ηλεκτρονικής αλληλογραφίας να είναι για πολλά χρόνια ενεργή. Με τη συνεχόμενη χρήση της διεύθυνσης αλληλογραφίας, ως μέσο επικοινωνίας και ταυτοποίησης στις διάφορες υπηρεσίες διαδικτύου, αυξάνεται συνεχόμενα ο αριθμός των μηνυμάτων που δύναται να δέχεται κάποιος ως εισερχόμενη αλληλογραφία. Καθημερινά ενδέχεται να βομβαρδίζεται κάποιος από πολλαπλά μηνύματα και να αποτελούν είναι επί το πλείστων ενοχλητική αλληλογραφία, δημιουργώντας «θόρυβο» στην εισερχόμενη αλληλογραφία  Εν όψει του κανονισμού GDPR του 2016, περί προστασίας προσωπικών δεδομένων το πρόβλημα μετριάστηκε κάπως, αλλά εξακολουθεί να παραμένει.

Μια εναλλακτική μορφή πάλι ασύγχρονης επικοινωνίας από κάποιον πάροχο προς το ευρύ κοινό για μαζική αποστολή μηνυμάτων, αλλά με περιορισμό στο μέγεθος της μεταδιδόμενης πληροφορίας έρχεται να δώσει το πρωτόκολλο των προωθημένων μηνυμάτων. Σε αντίθεση με τα

μηνύματα ηλεκτρονικού ταχυδρομείου στα προωθημένα μηνύματα η επικοινωνία είναι μονόδρομη και μόνο από ένα κεντρικό σημείο προς τους/τον τελικό καταναλωτή. Το μεγάλο πλεονέκτημα της τεχνολογίας αυτής είναι ότι ο τελικός καταναλωτής εφόσον δώσει τη συγκατάθεση του για τη λήψη των μηνυμάτων αυτών, δε διαμοιράζει προσωπικές πληροφορίες κατά τη διάρκεια της εγγραφής του στην υπηρεσία. Παράλληλα, δύναται ανά πάσα στιγμή να διακόψει τη λήψη των προωθημένων μηνυμάτων από πολλαπλά σημεία και επίπεδα ελέγχου. Τα προωθημένα μηνύματα μπορούν να διακοπούν από

- το ίδιο το λειτουργικό σύστημα της διασυνδεδεμένης συσκευής (σε καθολικό επίπεδο λειτουργικού συστήματος)
- από την εν λόγω εφαρμογή / περιηγητή (σε καθολικό επίπεδο εφαρμογής)
- σε επίπεδο εφαρμογής για ειδικά για συγκεκριμένο ιστότοπο
- δυνατότητα απεγγραφής απευθείας από την εν λόγω υπηρεσία αποστολής προωθημένων μηνυμάτων

Η τελευταία δυνατότητα απεγγραφής από την ίδια την υπηρεσία αποστολής προωθημένων μηνυμάτων, αν και τεχνικά είναι εφικτή δε μπορεί να πραγματοποιηθεί εύκολα καθώς το ίδιο το μήνυμα δε φέρει πληροφορίες, ούτε κάποιο σύνδεσμο με δυνατότητα απεγγραφής από την εν λόγω υπηρεσία.

Παράλληλα με τον έλεγχο λήψης προωθημένων μηνυμάτων που μπορεί να γίνει από πολλαπλά επίπεδα τα προωθημένα μηνύματα έχουν συγκεκριμένη διάρκεια ζωής. Αν παρέλθει το χρονικό ορόσημο της ζωής τους, τότε χάνονται και ο αποδέκτης δε τα λαμβάνει. Τέτοιου είδους προωθημένα μηνύματα έχουν νόημα ύπαρξης σε περιπτώσεις που ζητείται η αλληλεπίδραση του χρήση εντός κάποιου χρονικού παραθύρου, ενώ από εκεί και πέρα δε θέλουμε ή δεν έχει νόημα να λάβουμε την ανατροφοδότηση του χρήστη.

Όπως γίνεται εύκολα κατανοητό τα προωθημένα μηνύματα δεν έρχονται να αντικαταστήσουν τα μηνύματα ηλεκτρονικού ταχυδρομείου αλλά εν μέρη κάποια μηνύματα αυτών προκειμένου να αποφορτιστεί το ηλεκτρονικό ταχυδρομείο. Τα προωθημένα μηνύματα φέρουν πληροφορία που δεν αποθηκεύεται κάπου, και χάνεται αφού, αναγνωστεί, χωρίς να καταλαμβάνει κάποιο χώρο σε κάποια υπηρεσία.

Μια πλατφόρμα διαχείρισης και προώθησης προωθημένων μηνυμάτων θα προσέφερε καλύτερη χρονική διαχείριση των μηνυμάτων, θα επιτυγχάνονταν μεγαλύτερα ποσοστά επιτυχημένης ανατροφοδότησης από τους τελικούς χρήστης καθώς θα ανταποκρίνονταν μόνο όσοι ήθελαν και εκτός του επιτρεπτού χρονικού ορίου που έχει νόημα να καταγραφή η ανατροφοδότηση

του χρήστη. Σημαντικό πλεονέκτημα αυτής της τεχνολογίας είναι ότι δε συνδέεται με προσωπικά δεδομένα των χρηστών αλλά με το αναγνωριστικό αριθμό της εκάστοτε εφαρμογής ή περιηγητή. Η εγγραφή σε μια υπηρεσία προωθημένων μηνυμάτων δε συνοδεύει το χρήστη «εφόρου ζωής» όπως η διεύθυνση ηλεκτρονικής αλληλογραφίας. Πράγμα που σημαίνει ότι αν κάποιος αλλάξει περιηγητή ή συσκευή αυτόματα σημαίνει ότι παύει να δέχεται προωθημένα μηνύματα καθώς η νέα εφαρμογή ή περιηγητής δεν θα είναι εγγεγραμμένος στην υπηρεσία.

## 4.2 Λειτουργίες προτεινόμενου πληροφοριακού συστήματος διαχείρισης και αποστολής προωθημένων μηνυμάτων διαδικτύου

Έχοντας υπόψη όλα τα παραπώ κρίνεται σκόπιμος ο σχεδιασμός ενός νέου πληροφοριακού συστήματος διαχείρισης και αποστολής προωθημένων μηνυμάτων σε έναν ή πολλαπλούς αποδέκτες με σκοπό αφετέρου να αποφορτιστούν οι λογαριασμού ηλεκτρονικού ταχυδρομείου και αφετέρου να μόνο επίκαιρη ενημέρωση στον τελικό χρήστη, με αυτοδιαγραφόμενα προωθημένα μικρού μήκους μηνύματα. Τα μηνύματα αυτά δε θα αποθηκεύονται κάπου, θα διαγράφονται αμέσως μετά την ανάγνωση τους και δε χρήζει κάποια ενέργεια διαχείρισης παλιότερων ληφθέντων μηνυμάτων.

Επίσης λαμβάνοντας υπόψη το μεγάλο αριθμό ταυτόχρονων προωθημένων μηνυμάτων που δύναται να στέλνονται θα προταθεί μια αρχιτεκτονική τόσο σε επίπεδο βάσης δεδομένων αλλά και σε αρχιτεκτονική του όλου συστήματος με στόχο να μπορεί να εκτελεστεί κατανεμημένα σε πολλαπλούς εξυπηρετητές. Το προτεινόμενο σύστημα θα βασίζεται σε μια δομή που θα μπορεί δυναμικά να επεκτείνεται να και να περιλαμβάνει νέους «κόμβους» εξυπηρετητές.

Πιο συγκεκριμένα το προτεινόμενο πληροφοριακό σύστημα διαχείρισης και αποστολής προωθημένων μηνυμάτων μαζικής ή εξατομικευμένης αποστολής μέσω διαδικτύου θα έχει τις ακόλουθες σχεδιαστικές απαιτήσεις

- Να μπορεί να εκτελεί στο μικρότερο δυνατό χρονικό διάστημα τα αιτήματα που δέχεται.
- Να χρησιμοποιεί τη βάση δεδομένων όσο το δυνατόν γίνεται λιγότερο.
- Να κρατιούνται οι συνδέσεις στη βάσεις δεδομένων ενεργές για το μικρότερο δυνατό χρονικό διάστημα.
- Να ληφθεί υπόψη η απαίτηση για μελλοντική δυνατότητα δυναμικής προσθήκης δυναμικά επιπλέον εξυπηρετητών μαζικής αποστολής προωθημένων μηνυμάτων στο προτεινόμενο σύστημα.

- Να μπορεί να χρησιμοποιηθεί από τον τελικό διαχειριστή ιστοτόπων άμεσα και εύκολα χωρίς τη συγγραφή κώδικα και χωρίς καμία ουσιαστική τροποποίηση του πηγαίου κώδικα του ιστοτόπου.
- Να μπορεί να υποστηρίζει και να στέλνει προωθημένα μηνύματα σε πολλούς ιστότοπους.
- Να μπορεί να στείλει μαζικά ή εξατομικευμένα μηνύματα μαζικά σε εγγεγραμμένους αποδέκτες του ιδίου και μόνο ιστότοπου.
- Να μπορεί να αξιοποιεί το πρωτόκολλο προωθημένων μηνυμάτων για να είναι συμβατή με όλους τους νέους περιηγητές ανεξαρτήτου λειτουργικού συστήματος (αρκεί τις ελάχιστες απαιτήσεις λειτουργίας της προτεινόμενης πλατφόρμας).
- Τέλος το προτεινόμενο πληροφοριακό σύστημα να περιλαμβάνει όλες τις λειτουργικές μονάδες που απαιτούνται προκειμένου να σταλεί και να παρουσιαστεί ένα προωθημένο μήνυμα στον τελικό χρήστη, πλην τη οικοσύστημα που θα στείλει το προωθημένο μήνυμα. Κατά περίπτωση θα επιλέγεται το κατάλληλο οικοσύστημα για την αποστολή των παρωθημένων μηνυμάτων στους αποδέκτες μέσω της βασικής έκδοσης του πρωτοκόλλου προωθημένων μηνυμάτων (web push protocol).

Παράλληλα με τις σχεδιαστικές απαιτήσεις το προτεινόμενο σύστημα θα πρέπει να έχει και τις ακόλουθες λειτουργικές απαιτήσεις.

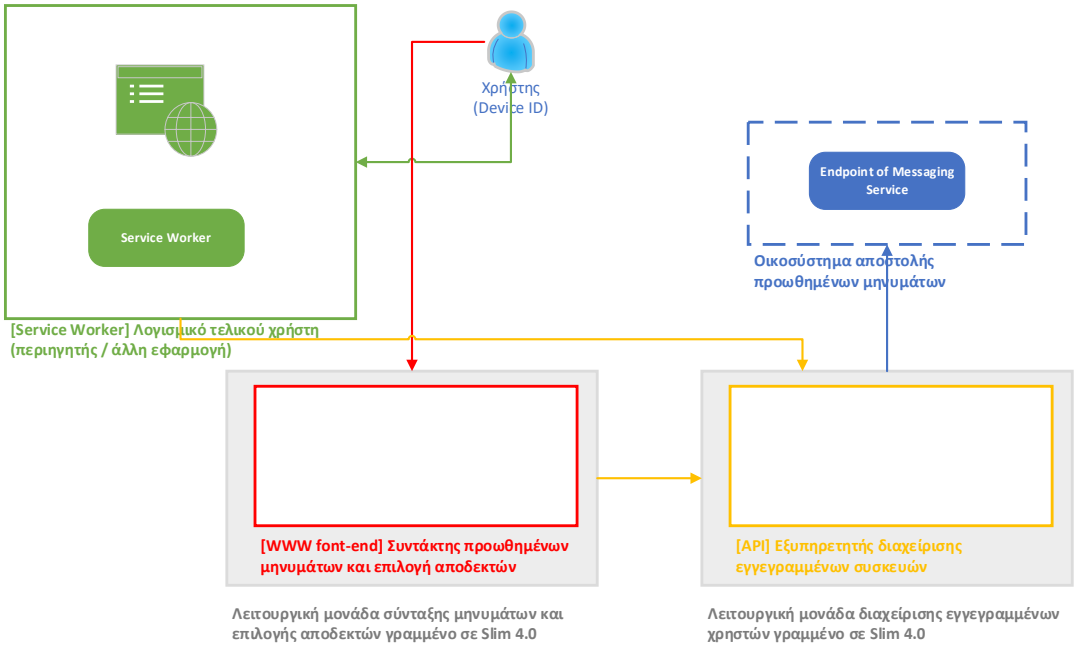
- Ο τελικός χρήστης θα μπορεί να εγγράφεται εύκολα την υπηρεσία ενημέρωσης μέσω προωθημένων μηνυμάτων χωρίς να διαμοιράζει καμιά προσωπική πληροφορία που θα τον αντιπροσωπεύει.
- Η πλατφόρμα θα έχει χρήστες διαχειριστές έναν ανά ιστότοπο
- Θα μπορούν να αποσταλούν μηνύματα που περιέχουν ένα σύντομο τίτλο, μια σύντομή περιγραφή, ένα εικονίδιο, μια εικόνα και έναν υπερσύνδεσμο ανάδρασης από την πλευρά του τελικού χρήστη
- Να μπορούν να στείλουν προωθημένα μηνύματα που να απαιτούν την ανάδραση και ανατροφοδότηση του χρήση.

## 4.3 Το νέο πληροφοριακό σύστημα διαχείρισης και αποστολής προωθημένων μηνυμάτων διαδικτύου

Ένα ολοκληρωμένο πληροφοριακό σύστημα ικανό να διαχειριστεί προωθημένα μηνύματα και εγγεγραμμένους χρήστες (περιηγητές/εφαρμογές) πρέπει να περιέχει 3 λειτουργικές μονάδες όπως

αυτές αναφέρθηκαν και στην παράγραφο 2.1. Για την ίδια αποστολή των προωθημένων μηνυμάτων θα χρησιμοποιηθούν τα κατάλληλα οικοσυστήματα που χρειάζεται ανάλογα με το περιβάλλον χρήσης του εκάστοτε τελικού χρήστη και αποδέκτη των προωθημένων μηνυμάτων / ειδοποιήσεων.

Το λειτουργικό διάγραμμα του προτεινόμενου πληροφοριακού συστήματος αποτυπώνεται στην ακόλουθη Εικόνα 3.




Εικόνα 3 – συνοπτικό λειτουργικό διάγραμμα ΠΣ αποστολής προωθημένων μηνυμάτων

Κάθε λειτουργική μονάδα έχει αποτυπωθεί με διαφορετικό χρώμα. Ο χρήστης έρχεται σε επαφή μόνο με την εφαρμογή / περιηγητή που έχει στη συσκευή του. Στην πλευρά του τελικού χρήστη θα εκτελείται ο service worker ο οποίος θα είναι υπεύθυνος για την εκτέλεση όλων των διαδικασιών που απαιτούνται από την πλευρά του τελικού χρήστη προκειμένου να μπορέσει να λειτουργήσει η πλατφόρμα προωθημένων μηνυμάτων. Η λειτουργική μονάδα που σχετίζεται με την σύνταξη και προώθηση μηνυμάτων όπως επίσης και η λειτουργική μονάδα που διαχειρίζεται τους εγγεγραμμένους χρήστες θα υλοποιηθεί με php χρησιμοποιώντας το framework Slim 4.0. Η πρώτη εκ των δύο θα περιέχει μια γραφική διεπαφή του χρήστη-διαχειριστή των μηνυμάτων. Η δεύτερη μονάδα θα αναπτυχθεί επίσης σε Slim 4.0 και θα αποτελεί το Application Programming Interface (API) της πλατφόρμας που θα έχει άμεση πρόσβαση στη βάση δεδομένων του συστήματος, στη διαχείριση των δεδομένων και στην επικοινωνία με οικοσυστήματα αποστολής προωθημένων μηνυμάτων. Για λόγους ευκολίας ανάπτυξης και γρήγορής κατανομής (κλωνοποίηση υπηρεσιών) και εγκατάστασης του προτεινόμενου πληροφοριακού συστήματος το όλο έργο θα αναπτυχθεί και θα φιλοξενηθεί σε περιβάλλον των ιδεατών υπηρεσιών μέσω containers υπηρεσιών χρησιμοποιώντας Docker.

Τέλος σαν λειτουργική μονάδα που θα είναι αρμόδια για τη λήψη και προβολή των προωθημένων μηνυμάτων θα εκτελεί τις απαραίτητες διεργασίες γραμμένες σε έναν service worker.

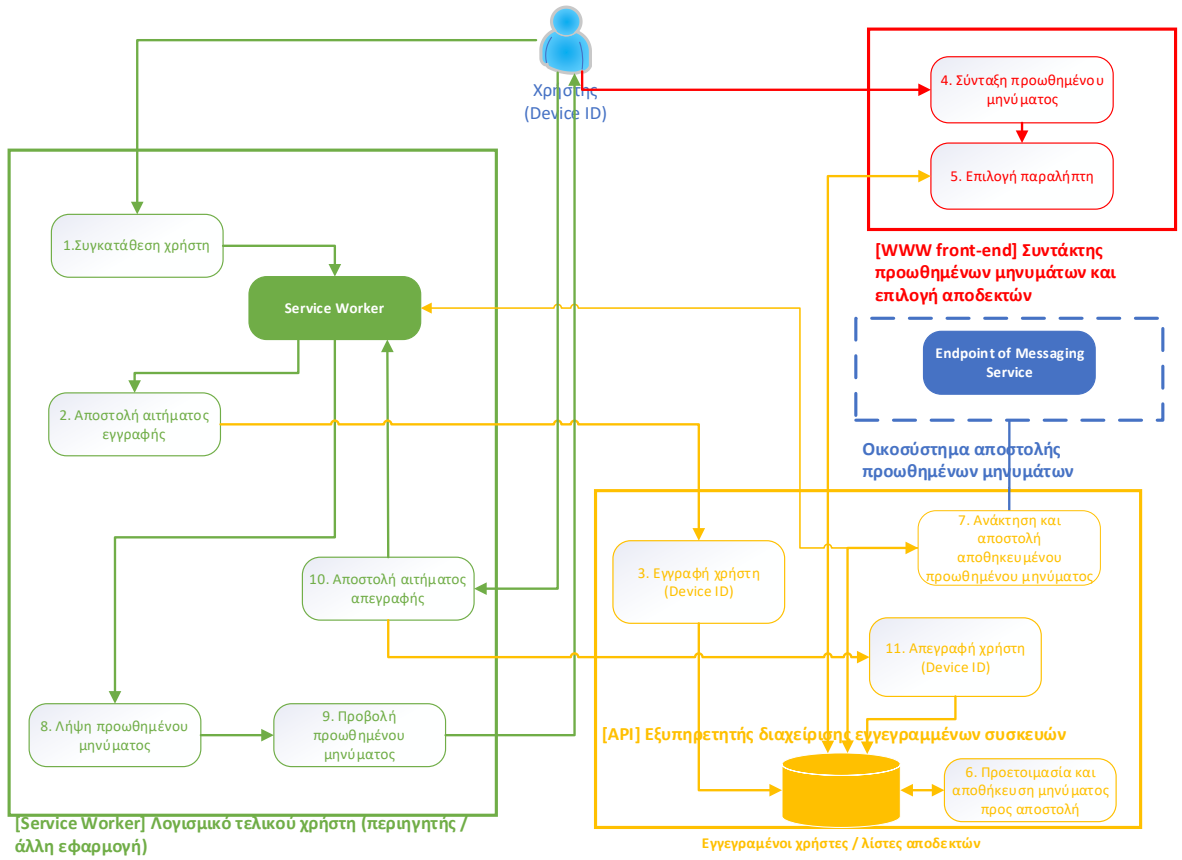
Αναλυτικότερα, θα παρουσιαστεί η κάθε λειτουργική μονάδα του πληροφοριακού συστήματος ξεχωριστά σε επόμενο κεφάλαιο της παρούσας εργασίας.

# 5 Αρχιτεκτονική πλατφόρμας

## 5.1 Γενική αρχιτεκτονική πλατφόρμας προωθημένων μηνυμάτων

Σε αυτό το κεφάλαιο θα γίνει αν αναλυτική παρουσίαση της αρχιτεκτονικής και των επιμέρους λειτουργικών μονάδων της πλατφόρμας προωθημένων μηνυμάτων που πραγματεύεται η παρούσα εργασία.

Το όλο πληροφοριακό σύστημα αποτελείται από τρεις λειτουργικές μονάδες. Το λειτουργικό κομμάτι του συστήματος που είχε χαρακτηριστεί με διακεκομμένη γραμμή στο ακόλουθο διάγραμμα (Εικόνα 4) αφορά μια εξωτερική οντότητα του συστήματος. Πρόκειται για το σημείο επαφής και αποστολής ων προωθημένων μηνυμάτων του εκάστοτε οικοσυστήματος.




Εικόνα 4 – αρχιτεκτονική ΠΣ προωθημένων μηνυμάτων

### 5.1.1 Λειτουργική μονάδα διαχείρισης εγγεγραμμένων συσκευών (χρηστών)

Αυτή η λειτουργική μονάδα είναι υπεύθυνη για τη διατήρηση των εγγεγραμμένων χρηστών και έγκυρων συνδρομητών της υπηρεσίας προωθημένων μηνυμάτων σε μια βάση δεδομένων

PostgreSQL. Ο ρόλος της μονάδας αυτή θα είναι να παρέχει τις πρόσβαση στη βάση δεδομένων του πληροφοριακού συστήματος. Αυτή η λειτουργική μονάδα θα αποτελεί το Application Programming Interface (API) του όλου πληροφοριακού συστήματος. Παράλληλα με τους εγγεγραμμένου χρήστες η λειτουργική μονάδα αυτή είναι υπεύθυνη για τη διαχωρισμό των εγγεγραμμένων χρηστών ανά ιστότοπο και δυνητικά σε μελλοντική επέκταση του έργου σε επιμέρους ομάδες χρηστών προκειμένου να σταλούν εξατομικευμένα προωθημένα μηνύματα [3]. Τέλος θα λαμβάνει το αποθηκευμένο μήνυμα από τη μονάδα σύνταξης μηνυμάτων και θα το προωθεί στο κατά περίπτωση οικοσύστημα που ταιριάζει με τον εκάστοτε αποδέκτη του μηνύματος.

Κάθε χρήστης θα αναπαρίσταται από μια εγγραφή που θα περιέχει το οικοσύστημα που θα πρέπει να χρησιμοποιηθεί προκειμένου να γίνει η αποστολή του μηνύματος, το αναγνωριστικό αριθμό της συσκευής που αιτήθηκε να εγγραφή στην υπηρεσία αποστολής μηνυμάτων και hash -- επαλήθευσης και επιβεβαίωσης της εγγραφής του χρήστη και την κωδικοποίηση των μηνυμάτων που θα αποσταλούν. Η δομή ενός χρήστη-συσκευής παρουσιάζεται στον ακόλουθο Πίνακας 2.

| Πεδίο | Περιγραφή |
|---|---|
| subscriver id | αναγνωριστικός αριθμός (uuid) χρήστη – δεν αποστέλλεται και δε χρησιμοποιείται – χρησιμοποιείται για τις ανάγκες της βάσης δεδομένων |
| endpoint | σημείο επικοινωνίας για την αποστολή ενός μηνύματος |
| p256dh (authorisation_key) | αναγνωριστικό συσκευής |
| auth (authToken) | hash επιβεβαίωσης και επαλήθευσης εγγραφής συσκευής |
| contentEncoding | Κωδικοποίηση μηνύματος |

Πίνακας 2 – δομή ενός χρήστη-συσκευή (αποδέκτη μηνυμάτων)

### 5.1.2 Λειτουργική μονάδα σύνταξης προωθημένων μηνυμάτων και επιλογής αποδεκτών

Η μονάδα αυτή του πληροφοριακού συστήματος παρέχει μια διεπαφή στους διαχειριστές του εκάστοτε ιστότοπου προκείμενου να συντάξουν και να στείλουν τα προωθημένα μηνύματα στους εγγεγραμμένους αποδέκτες της υπηρεσίας του ιστότοπου τους.

Η σύνταξη του μηνύματος εμπεριέχει τα ακόλουθα πεδία

| Πεδίο | Υποχρεωτικό / προαιρετικό |
|---|---|
| Τίτλος μηνύματος | **υποχρεωτικό** |
| Σώμα μηνύματος | **υποχρεωτικό** |
| εικονίδιο ειδοποίησης | προαιρετικό |
| συνοδευόμενη εικόνα μηνύματος | προαιρετικό |

| | |
|---|---|
| συνοδευόμενος υπερσύνδεσμος | προαιρετικό |
| Κουπιά ανάδρασης | προαιρετικό |
| Λίστα παραληπτών | **υποχρεωτικό** |

Πίνακας 3 – πεδία που εμπεριέχονται στη σύνταξη νέου προωθημένου μηνύματος

Η λειτουργική μονάδα αυτή θα είναι υπεύθυνη για την παροχή της διαθέσιμης λίστας εγγεγραμμένων αποδεκτών του εκάστοτε ιστότοπου.

### 5.1.3 Λειτουργική μονάδα αποστολής προωθημένων μηνυμάτων

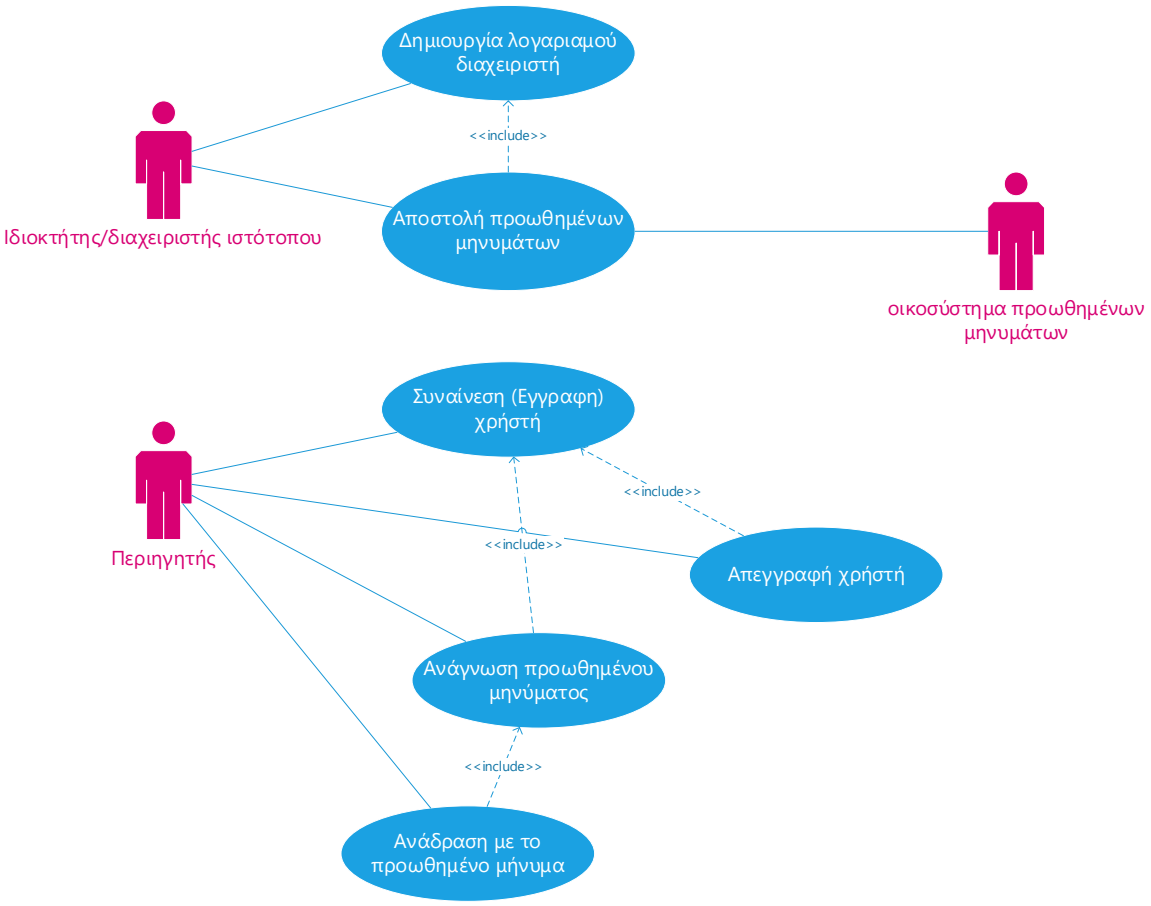
Για την αποστολή των προωθημένων μηνυμάτων γίνεται επικοινωνία με το εκάστοτε οικοσύστημα που απαιτείται κατά περίπτωση ανάλογα με το περιβάλλον χρήσης του αποδέκτη. Η επικοινωνία αυτή πραγματοποιείται με τη χρήση μιας βιβλιοθήκης ανοιχτού και ελεύθερου κώδικα minishlink/web-push. Η εν λόγω βιβλιοθήκη χρησιμοποιεί τη βασική έκδοση του πρωτοκόλλου προωθημένων μηνυμάτων (Web Push Protocol). Ανάλογα με το λειτουργικό σύστημα αλλά τον περιηγητή του έκαστο χρήστη, χρησιμοποιείται κάθε φορά το αντίστοιχο endpoint (σημείο επικοινωνίας με το εν λόγω οικοσύστημα) προκειμένου να εξασφαλιστεί η σωστή αποστολή των προωθημένων μηνυμάτων.

### 5.1.4 Λειτουργική μονάδα διεκπεραίωσης αιτημάτων τελικού χρήστη και προβολής εισερχόμενων προωθημένων μηνυμάτων

Η τελευταία λειτουργική μονάδα αφορά το κόμματι λογισμικού είναι υπεύθυνο για τη διαχείριση όλης της υπηρεσίας στην πλευρά του τελικού χρήστη. Το όλο λογισμικό που απαιτείται από για τη διεκπεραίωση όλων των εμπλεκόμενων λειτουργιών και διαδικασιών πραγματοποιείται από το service worker. Ο τελευταίος είναι υπεύθυνος να μεταφέρει τα αιτήματα του τελικού χρήστη προς κεντρική υπηρεσία (εγγραφή / απεγγραφή). Τέλος ο service worker και κατ’ επέκταση όλη η λειτουργική μονάδα είναι υπεύθυνος για τη λήψη και προβολή εισερχόμενων προωθημένων μηνυμάτων σαν αναδυόμενες ειδοποιήσεις στη συσκευή του τελικού χρήστη.

### 5.1.5 Αλληλεπίδραση συστήματος με περιβάλλον

Το προτεινόμενο σύστημα αλληλοεπιδρά με δυο ειδών χρήστες ενώ επικοινωνεί και με τρίτα συστήματα προκειμένου να στείλει τα προωθημένα μηνύματα. Στο ακόλουθο Use case διάγραμμα (Εικόνα 5) παρουσιάζεται σε υψηλό επίπεδο η αλληλεπίδραση των χρηστών του συστήματος με τον έξω κόσμο. Στο συγκεκριμένο διάγραμμα δε λαμβάνεται υπόψη η τεχνική αρχιτεκτονική του συστήματος. Αποτελεί απλά μια αφηρημένη απεικόνιση των λειτουργικών του όλου συστήματος.




Εικόνα 5 – use case πλατφόρμας

## 5.2 Βασικές διαδικασίες πληροφοριακού συστήματος προωθημένων μηνυμάτων χρηστών συστήματος

Το όλο πληροφοριακό σύστημα ενσωματώνει τις ακόλουθες διαδικασίες. Κάποιες από αυτές πραγματοποιούνται από τον service worker, κάποιες από το api και η σύνταξη του μηνύματος πραγματοποιείται από την υποδειγματική φόρμα σύνταξης μηνυμάτων (frontend)

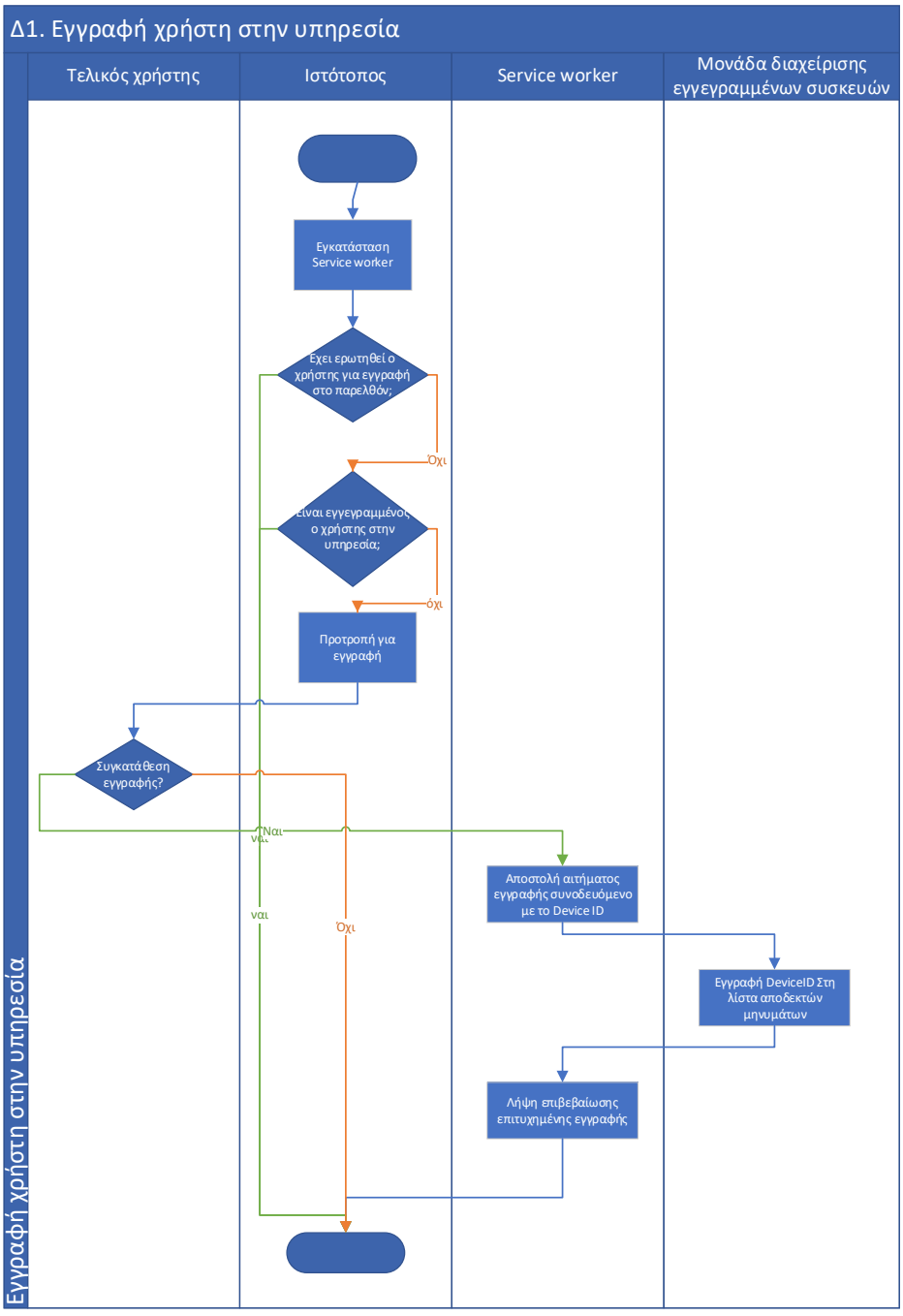
- Δ1. Εγγραφή (τελικού) χρήστη στην υπηρεσία [service worker / api]
- Δ2. Λήψη και προβολή εισερχόμενου μηνύματος [service worker]
- Δ3. Απεγγραφή περιηγητή από την υπηρεσία [service worker / api]
- Δ4. Εγγραφή διαχειριστή ιστοτόπου στην υπηρεσία προωθημένων μηνυμάτων [api]
- Δ5. Σύνδεση και ταυτοποίηση διαχειριστή ιστοτόπου στην υπηρεσία προωθημένων μηνυμάτων [api]
- Δ6. Προσθήκη ιστότοπου στην υπηρεσία [api]
- Δ7. Σύνταξη προωθημένου μηνύματος και επιλογή αποδεκτών [www-frontend]

- Δ8. Αποστολή προωθημένου μηνύματος σε ένα αποδέκτη [api]

Πέρα από τις παραπάνω διαδικασίες υπάρχουν και άλλες διαδικασίες που εκτελούνται από το API και θα αναφερθούν και να θα αναλυθούν στο αντίστοιχο κεφάλαιο.

### 5.2.1 Δ1. Εγγραφή χρήστη στην υπηρεσία [service worker / api]

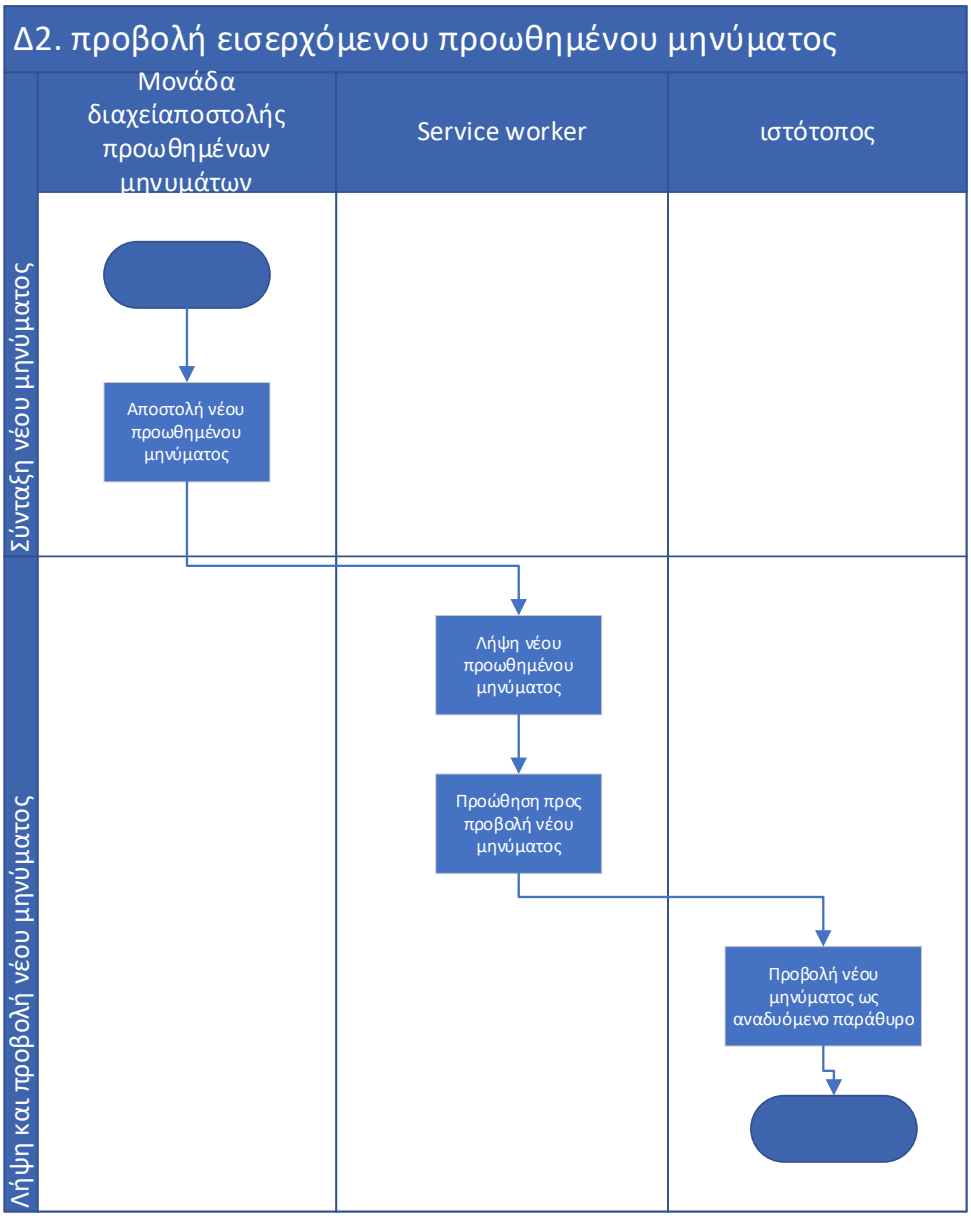
Το λειτουργικό διάγραμμα της διαδικασίας Δ1 Εγγραφή χρήστη στην υπηρεσία παρουσιάζεται στην Εικόνα 6.




Εικόνα 6 - διάγραμμα ροής διαδικασίας Δ1. Εγγραφή χρήστη στην υπηρεσία

### 5.2.2 Δ2. Λήψη και προβολή εισερχόμενου μηνύματος [service worker]

Το λογικό διάγραμμα της προαναφερόμενης διαδικασίας απεικονίζεται στην Εικόνα 7.

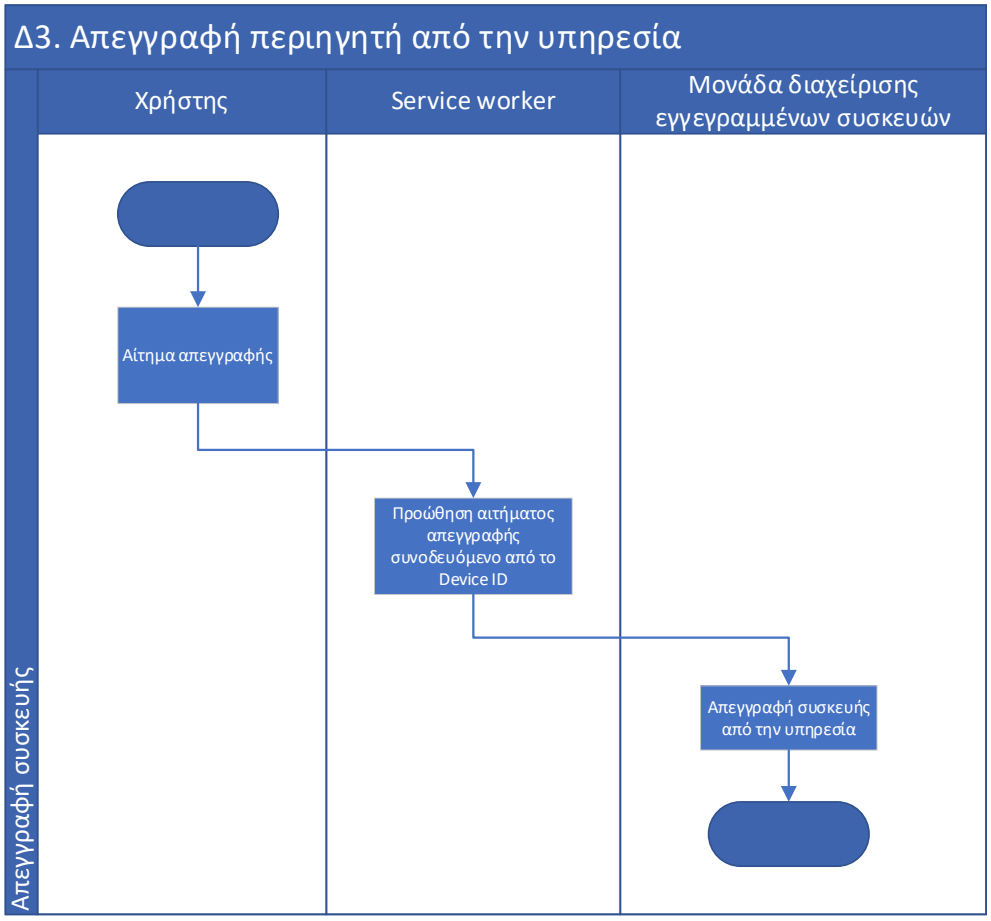



Εικόνα 7 - διάγραμμα ροής διαδικασίας Δ2. Αποστολή και προβολή νέου εισερχόμενου μηνύματος

### 5.2.3 Δ3. Απεγγραφή περιηγητή από την υπηρεσία [service worker / api]

Η διαδικασία της απεγγραφής μιας εφαρμογής/ περιηγητή από την υπηρεσία προωθημένων μηνυμάτων είναι εφικτή μέσω του service worker. Ο service worker αναλαμβάνει να προωθήσει το σχετικό αίτημα απεγγραφής του χρήστη στη μονάδα διαχείρισης εγγεγραμμένων χρηστών. Θεωρητικά είναι εφικτό να πραγματοποιηθεί ένα τέτοιο αίτημα, αλλά στην πράξη ένας χρήστης δε πραγματοποιεί απεγγραφή από τη σχετική υπηρεσία μέσα από τον περιηγητή ή που χρησιμοποιεί, καθώς συνηθίζεται να δίνεται η δυνατότητα στο τελικό χρήστη να απεγγραφή από την υπηρεσία

προωθημένων μηνυμάτων μέσω από ιστότοπο. Το αντίστοιχο διάγραμμα απεικονίζεται στην Εικόνα 8.




Εικόνα 8 - διάγραμμα ροής διαδικασίας Δ3. Απεγγραφή περιηγητή από την υπηρεσία

### 5.2.4 Δ4. Εγγραφή διαχειριστή ιστοτόπου στην υπηρεσία προωθημένων μηνυμάτων [api]

Η διαδικασία εγγραφής (sign up) ενός νέου διαχειριστή ιστοτόπου στην πλατφόρμα γίνεται μια κλήση τύπου post στο API του συστήματος. Η όλη διαδικασία, είναι μια διαδικασία ενός βήματος. Δέχεται σε μορφή json τα δεδομένα του χρήστη (email, κωδικό πρόσβασης) και επιστρέφει σε μορφή json μια εγγραφή ενός νέου χρήστη.

### 5.2.5 Δ5. Σύνδεση και ταυτοποίηση διαχειριστή ιστοτόπου στην υπηρεσία προωθημένων μηνυμάτων [api]

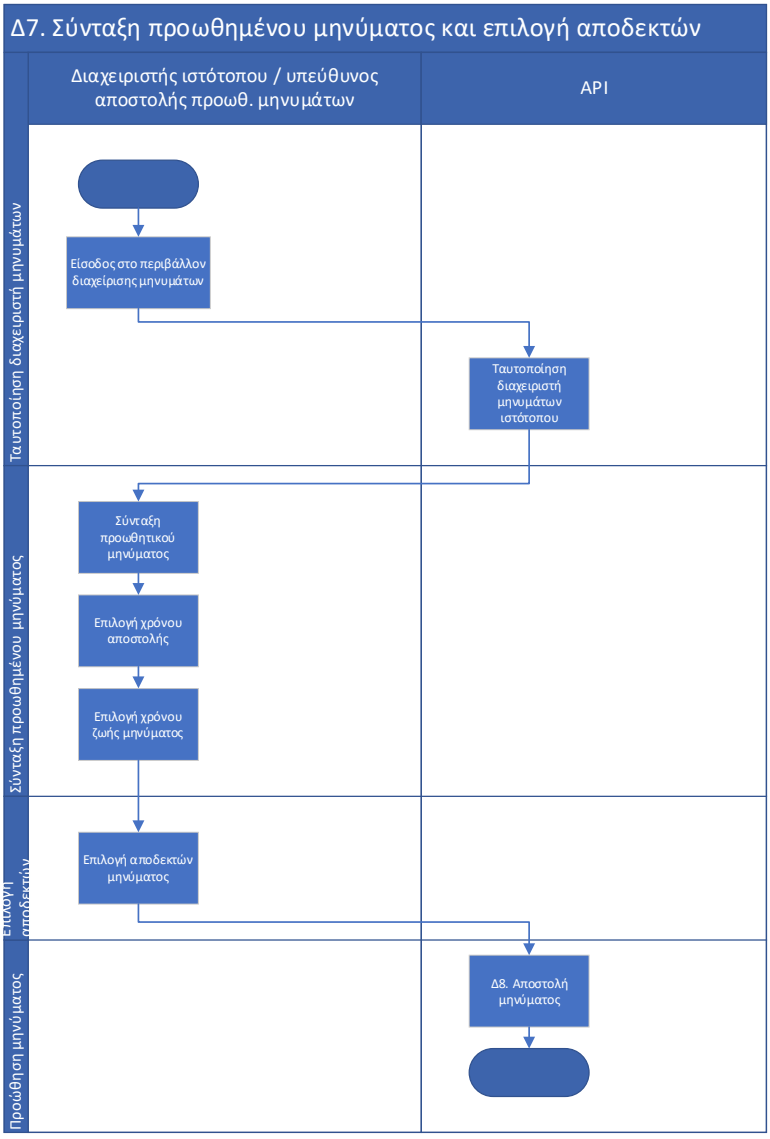
Η διαδικασία εγγραφής (sign in) ενός διαχειριστή ιστοτόπου στην πλατφόρμα γίνεται μια κλήση τύπου post στο API του συστήματος. Η όλη διαδικασία, και σε αυτή την περίπτωση είναι μια διαδικασία ενός βήματος. Δέχεται σε μορφή json τα δεδομένα του χρήστη (email, κωδικό πρόσβασης) και επιστρέφει σε μορφή json μια εγγραφή ενός νέου χρήστη.

### 5.2.6 Δ6. Προσθήκη ιστότοπου στην υπηρεσία [api]

Ένας διαχειριστή δύναται να έχει πολλούς ιστότοπους. Δύναται να μπορεί να προσθέσει πολλαπλούς ιστότοπους στο λογαριασμό του στην πλατφόρμα. Η κλήση αυτής της εντολής μέσω api μπορεί μόνο να επιτευχθεί μόνο εάν ο χρήστης έχει επιτυχώς ταυτοποιηθεί από τη διαδικασία Δ5 ή Δ5. Και σε αυτή την περίπτωση τα δεδομένα εισόδου και εξόδου είναι σε μορφή json. Δέχεται σαν είσοδο ένα url ενός ιστοπου και επιστρέφει μια json δομή που εκφράζει έναν ιστότοπο.

### 5.2.7 Δ7. Σύνταξη προωθημένου μηνύματος και επιλογή αποδεκτών [www-frontend]

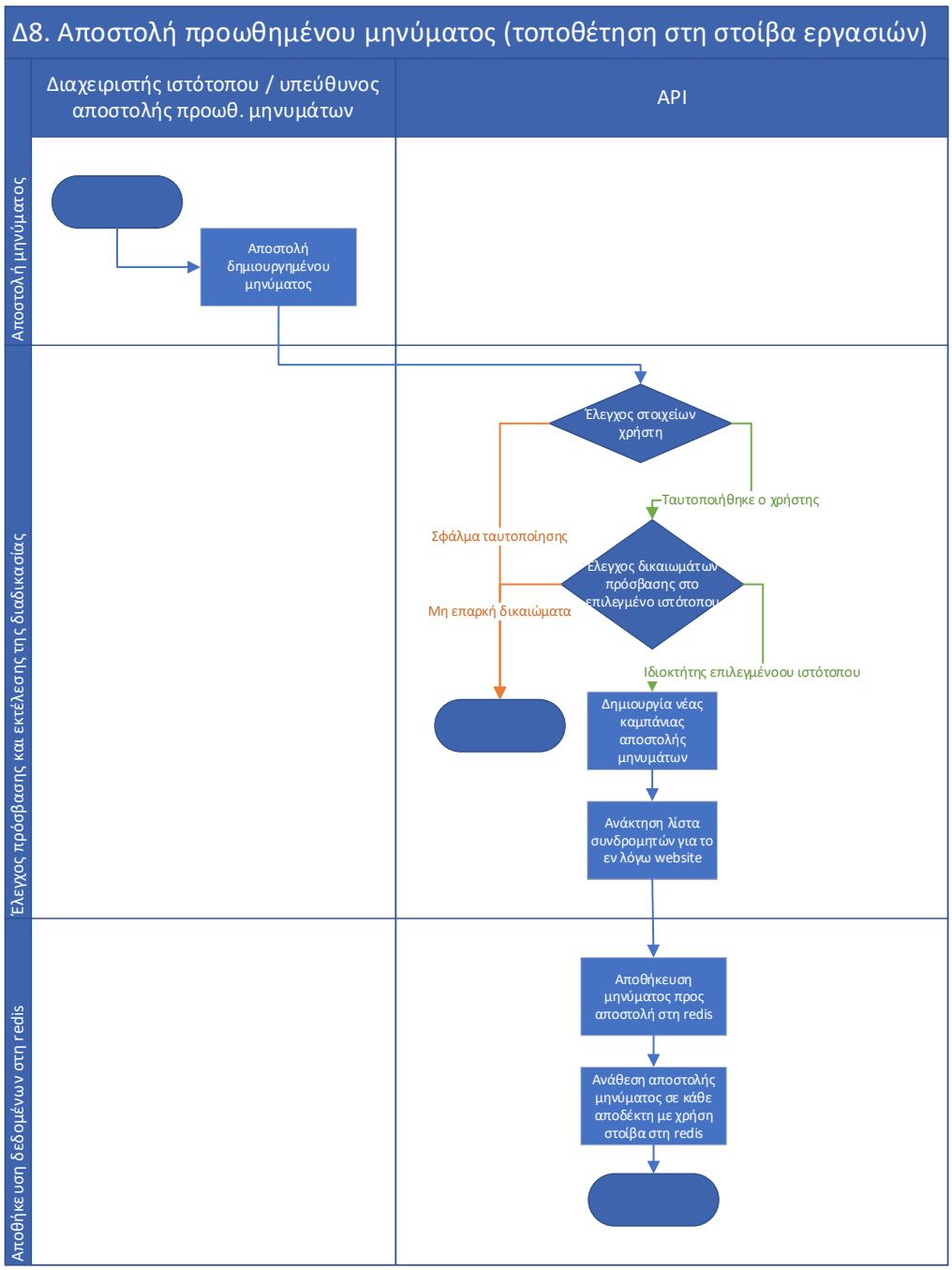
Η σύνταξη του προωθημένου μηνύματος λαμβάνει χώρα σε αυτή τη διαδικασία. Τέλος επιλέγονται οι αποδέκτες και το μήνυμα προωθείται με όλη τη συνοδευόμενη πληροφορία στην μονάδα αποστολής μηνυμάτων. Στην ακόλουθη Εικόνα 9 παρουσιάζεται διαγραμματικά η ροή των βημάτων αυτής της διαδικασίας.




Εικόνα 9 – διάγραμμα ροής διαδικασίας Δ4. Σύνταξη προωθημένου μηνύματος και επιλογή αποδεκτών

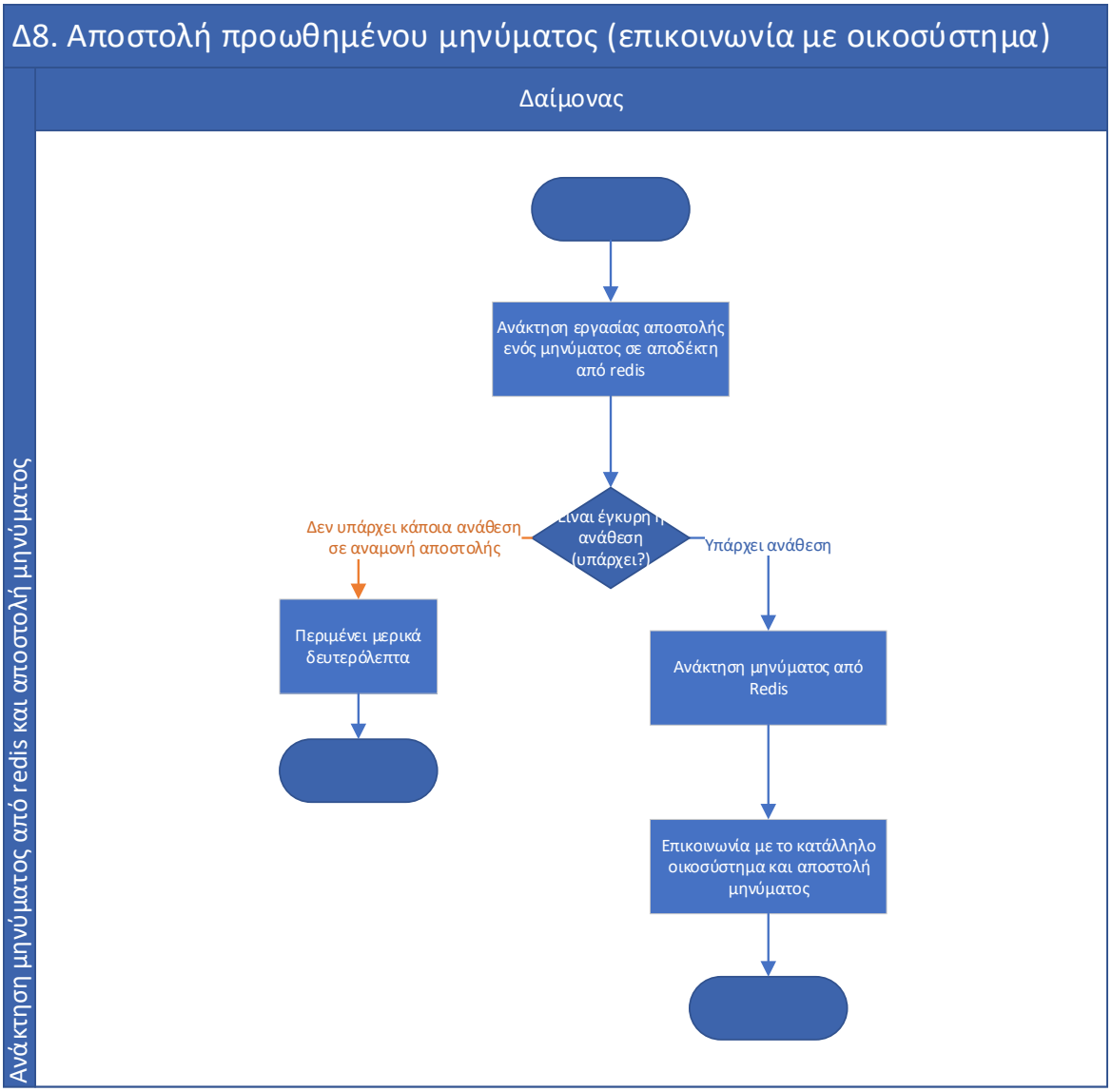
### 5.2.8 Δ8. Αποστολή προωθημένου μηνύματος σε ένα αποδέκτη [api]

Η διαδικασία της αποστολής ενός προωθημένου μηνύματος πραγματοποιείται ασύγχρονα με τη βοήθεια ενός «δαίμονα» ή μιας χρονοπρογραμματισμένης διαδικασίας. Η όλη διαδικασία χωρίζεται σε δυο φάσεις. Στην πρώτη φάση δημιουργειται η ανάθεση της αποστολής μηνυμάτων ανά παραλήπτη. Στην τελευταία φάση και βήμα της διαδικασίας γίνεται επικοινωνία στο κατάλληλο οικοστύστημα προκειμένου να προωθηθεί το μήνυμα στον κατάλληλο παραλήπτη. Αναλυτικά παρουσιάζεται στα ακόλουθα διαγράμματα οι φάσεις της διαδικασίας αποστολής των μηνυμάτων.




Εικόνα 10 – λειτουργικό διάγραμμα ανάθεσης εργασιών αποστολής μηνυμάτων

Στη συνέχεια παρουσιάζεται το λογικό διάγραμμα του «δαίμονα» που είναι υπεύθυνο για την αποστολή ενός μηνύματος από τη στοίβα των αναθέσεων αποστολής μηνυμάτων. Η ανάκτηση των πληροφοριών γίνεται από μια στοίβα με λειτουργία FIFO που έχει υλοποιηθεί από τη redis.




Εικόνα 11 – λειτουργικό διάγραμμα δαίμονα αποστολής μηνυμάτων

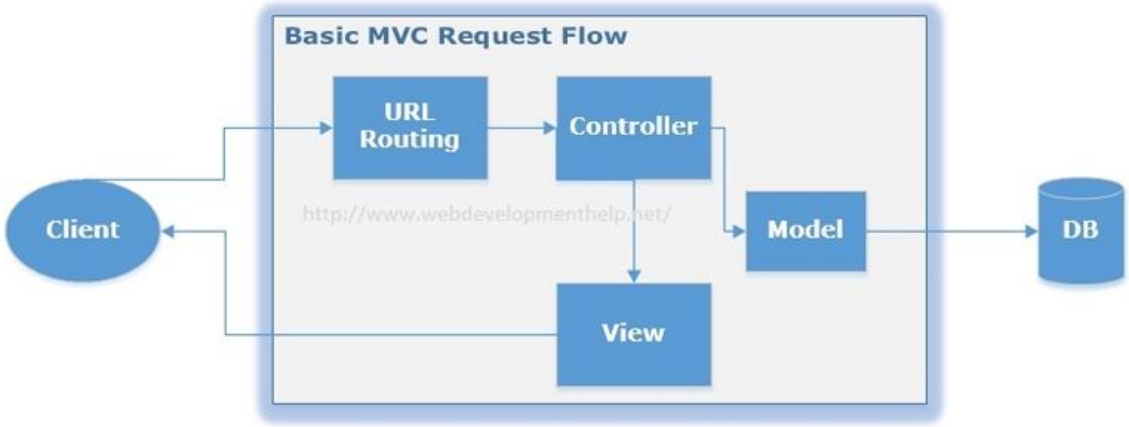
Το παραπάνω λογικό διάγραμμα παρουσιάζει τη λειτουργίας ενός script που θα λειτουργεί έως «δαίμονας» και θα εκτελείται συνεχώς. Μόλις ολοκληρωθεί θα εκτελείται πάλι από την αρχή.

## 5.3 Τεχνική αρχιτεκτονική της πλατφόρμας (Slim 4.0)

Η αρχιτεκτονική της υλοποίησης βασίστηκε στο περιβάλλον εργασίας Slim 4.0, όπου είναι και η καρδιά του συστήματος αποστολής προωθημένων μηνυμάτων. Για τις ανάγκες ανάπτυξης του API του πληροφοριακού συστήματος επιλέχθηκε το framework Slim που είναι γραμμένο σε PHP. Το Slim

όπως και το Laravel είναι διαδεδομένα frameworks ανάπτυξης λογισμικού σε γλώσσα PHP. Το Slim είναι μικρό και πιο ελαφρύ framework σε σχέση με το Laravel, ενώ παράλληλα αποτελεί μια εύκολη και γρήγορη επιλογή για ανάπτυξη API σε γλώσσα PHP. Η τρέχουμε σταθερή έκδοση του οποίου είναι η έκδοση 4.0. Η αρχική εγκατάσταση του slim 4.0 γίνεται εύκολα και γρήγορα μέσω αποθετηρίου κώδικα και το εργαλείο εγκατάστασης και ενημέρωσης πακέτων/βιβλιοθηκών λογισμικού το composer.

Η αρχιτεκτονική του όλου συστήματος είναι σύμφωνα με το Model View Control (MVC). Προσφέρει σύγχρονες και δομημένες προδιαγραφές ανάπτυξης λογισμικού σε γλώσσα PHP. Επιλέχθηκε το περιβάλλον ανάπτυξης Slim καθώς προσφέρει ένα γρήγορο και δομημένο τρόπο ανάπτυξης διαδικτυακών APIs. Η τυπική δομή της δομής του Slim παρουσιάζεται στην ακόλουθη Εικόνα 12 .

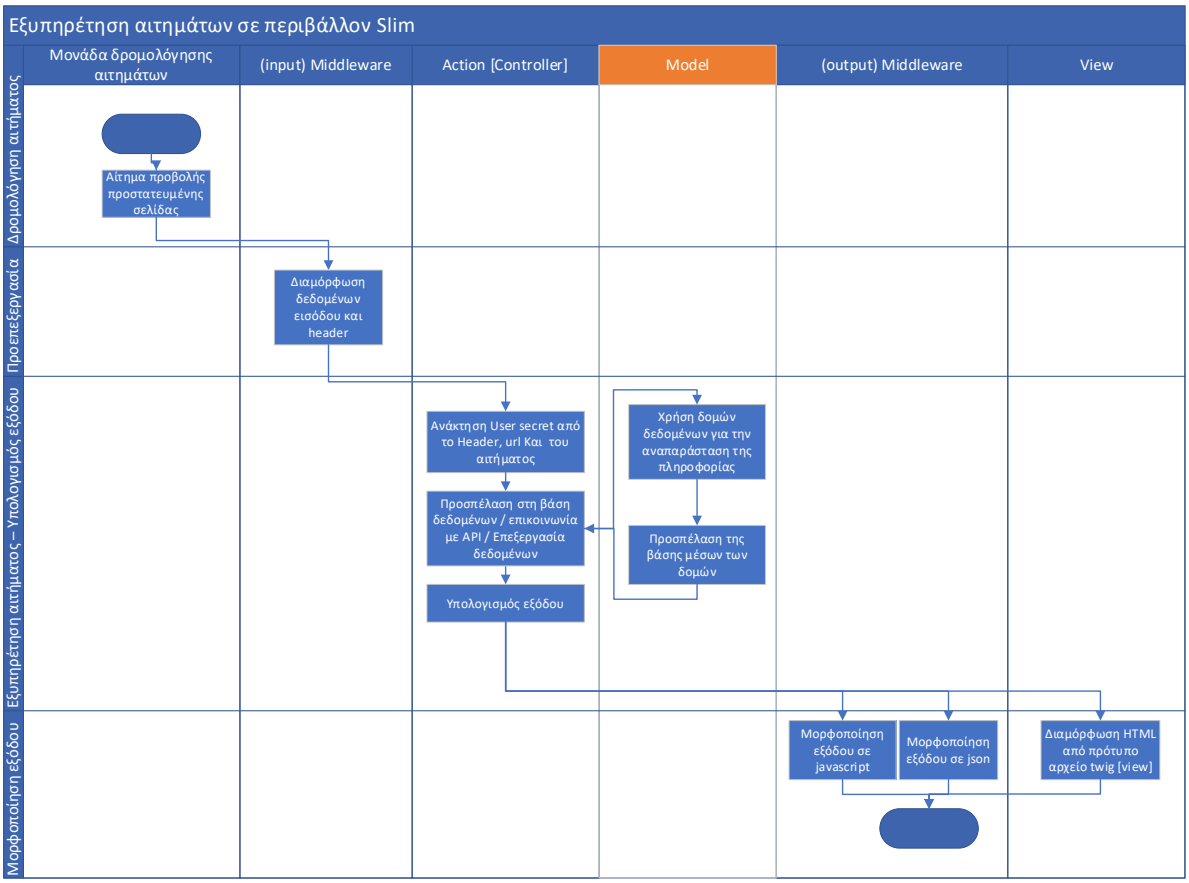



Εικόνα 12 – μοντέλο Slim MVC

Τόσο το ίδιο το API αλλά και το περιβάλλον διαχείρισης είναι γραμμένα σε Slim 4.0. Κάθε σελίδα ακολουθεί ή αίτημα στο API ακολουθεί μια ροή βημάτων προκειμένου να εξυπηρετηθεί και να επιτρέψει μια απάντηση ή μια σελίδα. Το βασικό λειτουργικό διάγραμμα του Slim 4.0 παρουσιάζεται στο ακόλουθο λειτουργικό διάγραμμα (Εικόνα 13).

Αρχικά όλα τα αιτήματα δρομολογούνται από τη μονάδα δρομολόγησης εισερχόμενων αιτημάτων. Αν είναι απαραίτητο τα δεδομένα μπορεί να υποστούν επεξεργασία από κάποιο middleware επεξεργασίας δεδομένων εισόδου. Στη συνέχεια γίνεται η επεξεργασία των δεδομένων / εξυπηρέτηση αιτήματος. Αν κρίνεται απαραίτητο τα αποτελέσματα διαμορφώνονται κατάλληλα με middleware εξόδου. Στην περίπτωση που η έξοδος είναι html σελίδα τότε συνηθίζεται να χρησιμοποιείται μια scripting γλώσσα για th διαμόρφωση του view. Στην προκειμένη περίπτωση έχει χρησιμοποιηθεί η twig.

Η προσπέλαση των δεδομένων δύναται να γίνει απευθείας από τον controller ή από ένα πιο διακριτό επίπεδο (model), όπου τα δεδομένα θα αναπαρίστανται σε δομές δεδομένων και θα περιλαμβάνουν διαδικασίες που θα επικαιροποιούν τα φυσικά αρχεία των δεδομένων (π.χ. βάση δεδομένων) κάθε φορά που τροποποιούνται οι πληροφορίες στις δομές αυτές από το ίδιο το Slim.




Εικόνα 13 – γενικό λειτουργικό διάγραμμα Slim 4.0

# 6 Διαμόρφωση περιβάλλοντος ανάπτυξης εφαρμογής

## 6.1 Εργαλεία ανάπτυξης που χρησιμοποιήθηκαν για την ανάπτυξη της πλατφόρμας

### 6.1.1 Git

Για τη διαχείριση των εκδόσεων αλλά και των επεκτάσεων του project χρησιμοποιήθηκε το αποθετήριο κώδικα λογισμικού bitbucket. Για να μπορούμε να ελέγξουμε τις εκδόσεις και τα διάφορα χαρακτηριστικά που αναπτύχθηκαν και θα αναπτυχθούν στη ζωή του έργου αυτού. Ο έλεγχος και η χρήση που αποθετηρίου έγινε με το εργαλείο Git.

### 6.1.2 Composer

Πρόκειται για ένα εργαλείο μεταφόρτωσης και ενσωμάτωσης κώδικα στο project από διαθέσιμα πακέτα ανοικτού κώδικα. Χρήσιμοι σύνδεσμοι για την αναζήτηση και ενσωμάτωση πακέτων στο project είναι ο ακόλουθος σύνδεσμος https://packagist.org.

Στο συγκεκριμένο έργο έχει χρησιμοποιηθεί το εργαλείο composer. Η χρήση του οποίου είναι απαραίτητα όταν προσθέτουμε κάποια βιβλιοθήκη ή τροποποιούμε με οποιοδήποτε τρόπο (προσθήκη, μετονομασία ή διαγραφή) τις κλάσεις του συστήματος που σχεδιάσουμε.

Το εργαλείο αυτό είναι υπεύθυνο στο να δημιουργήσει ένα αρχείο που θα εσωκλείει όλες τις σχετικές βιβλιοθήκες του έργου με σκοπό να μπορούν να κληθούν και να χρησιμοποιηθούν αργότερα εντός του πηγαίου κώδικα. Για το λόγο αυτό φτιάχνει ένα αρχείο ***autoload*** κάθε φορά που καλείται με τις απαραίτητες βιβλιοθήκες. Στη συνέχεια η ενσωμάτωση μονάχα του ***autoload*** τον πηγαίο κώδικα έχει σαν αποτέλεσμα την ενσωμάτωση όλων των απαραίτητων βιβλιοθηκών στον πηγαίο κώδικα.

## 6.2 Διαμόρφωση containers λογισμικού σε περιβάλλον Docker

Για τις ανάγκες ανάπτυξης, εύκολης μεταφοράς αλλά και δυνατότητα γρήγορης και εύκολης κλιμάκωση των υπηρεσιών του παρόντος πληροφοριακού συστήματος όλες οι συνιστώσες λογισμικού εκτελούνται κεντρικά σε εξυπηρετητή αναπτύχθηκαν χρησιμοποιώντας ιδεατές υπηρεσίες σε περιβάλλον docker. Με αυτό τον τρόπο δύναται να εξομοιωθεί ακριβώς το περιβάλλον πραγματικής λειτουργίας (περιβάλλον λειτουργικού συστήματος, εκδόσεις λογισμικού υπηρεσιών,

μεταβλητές συστήματος κ.α.) και να γίνει χρήση ακριβώς της ίδια έκδοση οποιαδήποτε συνιστώσας λογισμικού, που θα χρησιμοποιηθεί, από το συστήματος παραγωγής, στο σύστημα ανάπτυξης.

Ένα άλλο σημαντικό πλεονέκτημα των τεχνικής ανάπτυξης λογισμικού κάνοντας χρήση του Docker είναι ότι κάθε συνιστώσα λογισμικού έχει το δικό της ανεξάρτητο περιβάλλον λειτουργίας χωρίς να αλληλοεπιδρά και να επηρεάζεται από άλλες υπηρεσίες, ρυθμίσεις και γενικά από το περιβάλλον λειτουργίας του συστήματος που χρησιμοποιείται για την ανάπτυξη.

## 6.3 Περιβάλλον λειτουργίας του API του πληροφοριακού συστήματος από τη μονάδα διαχείρισης εγγεγραμμένων χρηστών και προώθησης μηνυμάτων στη μονάδα αποστολής

Η μονάδα διαχείρισης εγγεγραμμένων χρηστών και προώθησης μηνυμάτων στη μονάδα αποστολής αποτελεί το API του εν λόγω συστήματος. Έχει πρόσβαση στη βάση δεδομένων και ενσωματώνει βασικές λειτουργίες όπως εγγραφή και απεγγραφή χρηστών, πολιτικές προσωρινής αποθήκευσης (caching policies), διάθεση service worker, προώθηση μηνυμάτων στη μονάδα αποστολής στο τελικό χρήστη και τέλος κάθε είδους προσπέλαση στη βάση δεδομένων του συστήματος.

Το συγκεκριμένο μέρος του πληροφοριακού συστήματος υλοποιήθηκε με Slim 4.0 και απαιτεί για να λειτουργήσει τα ακόλουθα επιμέρους στοιχεία και υπηρεσίες

- Nginx (web server)
- PHP 8.0
- PostgreSQL 9.6

Οι προαναφερόμενες υπηρεσίες ρυθμίστηκαν στα πλαίσια της ανάπτυξης του παρόντος έργου να λειτουργούν σε τρεις διαφορετικούς docker containers, ένας για κάθε μια υπηρεσία. Παράλληλα με τις ρυθμίσεις των ανωτέρων υπηρεσιών αρχικοποιείται και το σχήμα της βάσης που απαιτείται για να λειτουργήσει το API. Το εννοιολογικό μοντέλο δεδομένων της βάσης παρουσιάζεται στην επόμενη ενότητα. Ο κώδικας και τα script που εκτελούνται στην δημιουργία και ρύθμιση των containers παρατίθενται στο Παράτημα Α.

## 6.4 Περιβάλλον λειτουργίας του διαχειριστικού εργαλείου αποστολής προωθημένων μηνυμάτων

Το περιβάλλον λειτουργίας του διαχειριστικού εργαλείου αποστολής προωθημένων μηνυμάτων ενσωματώνει τη λειτουργική μονάδα σύνθεσης νέων μηνυμάτων. Η όλη επικοινωνία με τις υπόλοιπες λειτουργικές μονάδες γίνεται μέσω API. Για τις ανάγκες της παρούσας εργασίας περιλαμβάνει μόνο τις απαραίτητες διαδικασίες που απαιτούνται προκειμένου να αναδειχτεί η καλή λειτουργίας της πλατφόρμας και να αποσταλούν προωθημένα μηνύματα.

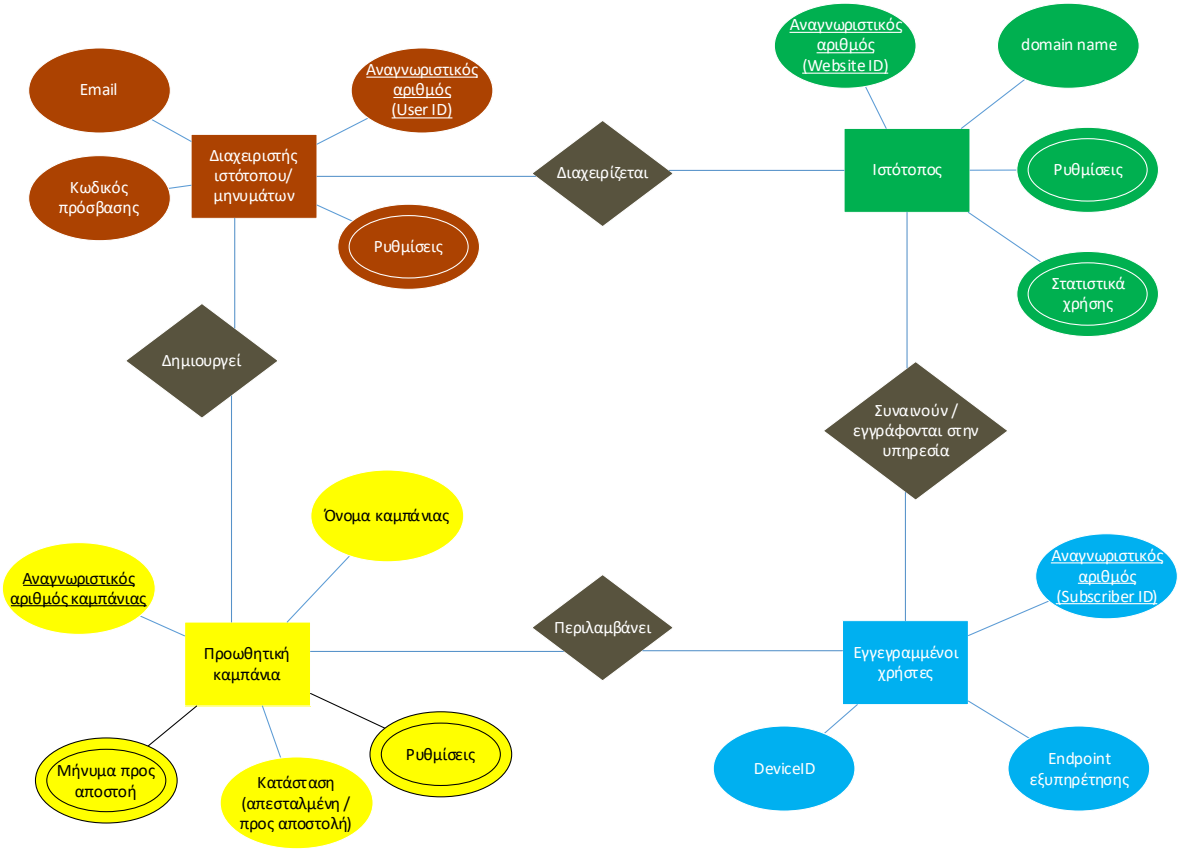
Για να μπορέσουν να εκτελεστούν όλες οι παραπάνω διεργασίες, γίνεται επικοινωνία και αποστολή αιτημάτων στο API του συστήματος. Το συγκεκριμένο μέρος του πληροφοριακού συστήματος λειτουργεί σε ιδεατά docker containers και υλοποιήθηκε με Slim 4.0 κατά αναλογία και αντιστοιχία με το API

- Nginx (web server)
- PHP 8.0

Οι προαναφερόμενες υπηρεσίες ρυθμίστηκαν να λειτουργούν σε δυο διαφορετικούς docker containers, ένας για κάθε μια υπηρεσία. Ο κώδικας και τα script που εκτελούνται στην δημιουργία και ρύθμιση των containers παρατίθενται στο Παράτημα Γ.

## 6.5 Βάση δεδομένων PostgreSQL για τη λειτουργική μονάδα που θα έχει το ρόλο του API του πληροφοριακού συστήματος.

Για τις ανάγκες του API σχεδιάστηκε ένα σχήμα βάσης δεδομένων σε PostgreSQL. Επιλέχθηκε η postgreSQL καθώς είναι αρκετά γρηγορότερη από την αντίστοιχη mariaDB ή MySQL. Το κύριο πλεονέκτημα της PostgreSQL σε αντίθεση με τις δυο προηγούμενες είναι ότι παρέχει εν γένει υποστήριξη σε προηγμένες και πολύπλοκες συναρτήσεις διαχείρισης json και jsonb (binary-indexed json) τύπων δεδομένων, με αποτέλεσμα να την καθιστά ιδανική λύση για ανάπτυξη διαδικτυακών υπηρεσιών. Στο ακόλουθο εννοιολογικό βάσης δεδομένων της Εικόνα 14 παρουσιάζονται οι τέσσερεις εννοιολογικές οντότητες που απαιτούνται για να λειτουργήσει το API.




Εικόνα 14 – εννοιολογικό μοντέλο βάσης δεδομένων PostgreSQL

Διακρίνουμε τέσσερις διαφορετικές εννοιολογικές οντότητες στο σχήμα της βάσης. Κάθε ενδιαφερόμενος ιδιοκτήτης ιστότοπου που επιθυμεί να εγγραφεί και να χρησιμοποιήσει την υπηρεσία αποστολής προωθημένων μηνυμάτων στους επισκέπτες του ιστότοπου του, αναφέρεται σαν διαχειριστής ιστότοπου/υπεύθυνος αποστολής μηνυμάτων και δύναται να χρησιμοποιήσει την υπηρεσία σε πολλαπλούς ιστότοπους που μπορεί να διαθέτει. Ενώ κάθε ιστότοπος έχει τους δικούς του επισκέπτες και επομένως εγγεγραμμένους χρήστες στην υπηρεσία προωθημένων μηνυμάτων. Τέλος για κάθε ιστότοπο μπορούμε να πραγματοποιήσουμε πολλές διαφορετικές αποστολές (καμπάνιες αποστολής προωθημένων μηνυμάτων).

# 7 Υλοποίηση προτεινόμενου συστήματος

Η υλοποίηση του προτεινόμενου συστήματος απαρτίζεται από τρεις επιμέρους λειτουργικές μονάδες όπως αναλύθηκαν στην παράγραφο 5.1. Ο σχεδιασμός και ο τρόπος που διεκπεραιώνονται οι διαδικασίες έχει σχεδιασμένος με τέτοιο τρόπο έτσι ώστε να μπορεί το σύστημα σε μελλοντική επέκταση του να ενσωματώνει δυναμικά επιπλέον εξυπηρετητές αποστολής προωθημένων μηνυμάτων. Το μεγαλύτερο μέρος λειτουργίας του διεκπεραιώνεται του συστήματος πραγματοποιείται, μέσω του API που σχεδιάστηκε για αυτό το λόγω. Το API υποβοηθάτε από μια διαδικασία που θα εκτελείται επαναλαμβανόμενα σαν δαίμονας ή σαν χρονοπρογραμματισμένη εργασία και η οποία θα αποστέλλει τα προωθημένα μηνύματα.

Παράλληλα με το API την λειτουργία του συστήματος συμπληρώνει ο service worker που αναλαμβάνει να μεσολαβήσει ανάμεσα στο API και στον τελικό χρήστη και αποδέκτη των μηνυμάτων, όπως επίσης και την προβολή των ληφθέντων προωθημένων μηνυμάτων.

Τέλος το τελευταίο κομμάτι του συστήματος αποστολής προωθημένων μηνυμάτων συμπληρώνει μια φόρμα ελεγχόμενης πρόσβασης όπου μπορεί ο εκάστοτε διαχειριστής ενός ιστοτόπου να εγγραφεί, να καταχωρήσει τα στοιχεία του και τέλος να συνθέσει ένα νέο προωθημένο μήνυμα και να το προωθήσει στους χρήστες του ιστότοπου του.

## 7.1 Υλοποίηση Application Programming Interface (API)

### 7.1.1 Επισκόπηση API

Η καρδιά του συστήματος αποτελείται από το Application Programming Interface (API). Το API έχει το ρόλο του διαχειριστή των εγγεγραμμένων συσκευών, τη διαχείριση των διαχειριστών ιστοτόπου και των ιστοσελίδων τους και τέλος την προώθηση των προς αποστολή μηνυμάτων στα αντίστοιχα οικοσυστήματα αποστολής προωθημένων μηνυμάτων.

Το API περιλαμβάνει τις κλήσεις όπως φαίνονται στον ακόλουθο Πίνακας 4.

| **Κατηγορία** | **κλήση** | **endpoint** | **Διαδικασία** |
|---|---|---|---|
| users | register | [POST] /v1/user/register | Δ4 |
| | login | [POST] /v1/user/register | Δ5 |
| | validate | [POST] /v1/user/validate | |
| devices | subscribe | [POST] /v1/device/subscribe | Δ1 |

| | | | |
|---|---|---|---|
| | unsubscribe | [DELETE] /v1/device/unsubscribe | Δ3 |
| notification | push | [POST] /v1/notification/push | |
| service worker | get service worker | [GET] /v1/{website_id}/service-worker.js | |
| | get SDK | [GET] /v1/{website_id}/sdk.js | |
| campaign | create campaign | [POST] /v1/campaign/{website} | Δ8 |
| websites | add website | [POST] /v1/websites | Δ6 |

Πίνακας 4 – λίστα κλήσεων API

Οι κλήσεις του API όπως παρουσιάζονται στον παραπάνω πίνακα αποτυπώνουν τις βασικές λειτουργίες της πλατφόρμα αλλά ταυτόχρονα αρκούν για να υποστηρίξουν ένα ολοκληρωμένο σύστημα αποστολής προωθημένων μηνυμάτων.

Δυο εκ των παραπάνω κλήσεων του API επιστρέφουν δεδομένα τύπου application/javascript. Πρόκειται για τις κλήσεις που σχετίζονται με τον service worker και τις συνοδευτικές συναρτήσεις που απαιτούνται για να λειτουργήσει ο μηχανισμός εγγραφής/απεγγραφής ενός χρήστη όπως επίσης και λήψης ενός προωθημένου μηνύματος.

Για να μπορέσει ένας διαχειριστής να ενσωματώσει την υπηρεσία αποστολής και λήψης προωθημένων μηνυμάτων στον ιστότοπο του, θα πρέπει να ενσωματώσει δυο αρχεία κατάλληλα διαμορφωμένα με κώδικα javascript. Για λόγους απλούστευσης αλλά και ευκολίας εγκατάστασης αυτής της πλατφόρμας σε έναν ιστότοπο, τα δυο απαραίτητα αυτά αρχεία javascript γεννιούνται αυτόματα και κατάλληλα διαμορφωμένα από το API του συστήματος. Με αυτό τον τρόπο η εγκατάσταση της υπηρεσίας αποστολής προωθημένων μηνυμάτων σε έναν ιστοτοπο απαιτεί μόνο την προσθήκη μιας γραμμής κώδικα στην κεφαλή κάθε σελίδας(header) που να κάνει μια κλήση προς το API για να γεννήσει και να ενσωματώσει τα σχετικά αρχεία javascript.

### 7.1.2 Διαδικασία σχεδιασμού και υλοποίησης API

#### 7.1.2.1 Διαμόρφωση περιβάλλον και εγκατάσταση Slim 4.0

Αρχικά δημιουργήθηκε ένα ιδεατό περιβάλλον εργασίας κάνοντας χρήστη του docker και δημιουργώντας τέσσερις containers

| **Docker container** | **Περιγραφή** |
|---|---|
| api_php_fpm_1 | Παρέχει ένα ιδεατό αντίγραφο της PHP 8.0 |
| api_postgres_1 | Παρέχει ένα ιδεατό αντίγραφο της PostgreSQL 9.6 |
| api_redis_1 | Παρέχει ένα ιδεατό αντίγραφο της redis |
| api_ngnix_1 | Παρέχει ένα ιδεατό αντίγραφο του εξυπηρετητή Nginx |

Πίνακας 5 - λίστα ιδεατών υπηρεσιών για τις ανάγκες του API

Η χρήση των παραπάνω containers παρέχουν όλες τις απαραίτητες βασικές προϋποθέσεις για να λειτουργήσει η νέα πλατφόρμα.

#### 7.1.2.2 Εγκατάσταση απαραίτητων βιβλιοθηκών και επεκτάσεων της PHP

Η προσθήκη κάθε πακέτου/βιβλιοθήκη λογισμικού δύναται να προστίθεται χειροκίνητα και μεμονωμένα με τη χρήση του εργαλείου composer. Αντί αυτού δημιουργήθηκε ένα αρχείο composer.json όπου προσπελαύνεται αυτόματα από το εργαλείο composer και εγκαθιστά αυτόματα όλες τις σχετικές και απαραίτητες βιβλιοθήκες ενώ ταυτόχρονα ελέγχει αν ικανοποιούνται οι ελάχιστες απαιτήσεις λειτουργίας λογισμικού για να εγκατασταθούν και να λειτουργήσουν. Ο πλήρης κώδικας δημιουργίας και διαμόρφωσης των προαναφερόμενων docker containers παρατίθεται στο Παράτημα Α.

#### 7.1.2.3 Δημιουργία σχήματος βάσης δεδομένων για τις ανάγκες του API

Για τις ανάγκες του API σχεδιάστηκε και υλοποιήθηκε ένα σχήμα βάσης γραμμένο σε PostgreSQL με γνώμονα της ταχύτητα ανάκτησης της πληροφορίας και της ελαχιστοποίησης του χρόνου προσπέλασης στο σχήμα της βάσης. Το εννοιολογικό μοντέλο της βάσης δεδομένων έχει ήδη παρουσιαστεί σε προηγούμενη ενότητα ενώ ο πλήρης κώδικας δημιουργίας του σχήματος της βάσης δεδομένων εκτελείται με την αρχικοποίηση του container api_postgres_1. Ο πλήρης κώδικας σχεδιασμού της βάσης αποτελείται από δυο αρχεία για λόγους ευκολίας διαχείρισης. Η σειρά εκτέλεσης των αρχείων ανάκτησης και σχεδιασμού της βάσης δεδομένων γίνεται από τον docker container με αλφαβητική σειρά. Πρώτα εκτελείται το αρχείο που έχει πρόθεμα «1.» όπου και ενεργοποιείται μια επέκταση της PostgreSQL που δεν είναι ενεργοποιημένη εξ αρχής και στη συνέχεια το αρχείο με το πρόθεμα «2.», όπου περιλαμβάνει το ουσιαστικό σχήμα της βάσης δεδομένων. Στο Παράτημα Α παρατίθεται ο πλήρης κώδικας των προαναφερόμενων δυο αυτών αρχείων.

#### 7.1.2.4 Δημιουργία κλειδιών χρήσης υπηρεσίας push notifications

Προκειμένου να σχεδιαστεί ένα σύστημα που θα μπορεί να στέλνει προωθημένα μηνύματα χρειάζεται ενός είδους εξουσιοδότησης, ταυτοποίησης και κρυπτογράφησης ταυτόχρονα, σύμφωνα με το πρωτόκολλο χρήσης των προωθημένων μηνυμάτων διαδικτύου (web push). Η ταυτοποίηση και εξουσιοδότηση αποστολής προωθημένων μηνυμάτων γίνεται με τη χρήση του Voluntary Application Server Identification for Web Push (VAPID) protocol [11]. Το VAPID αποτελείται από ένα ζευγάρι ιδιωτικού και δημόσιου κλειδιού κρυπτογράφησης των μηνυμάτων. Παράλληλα το ζευγάρι αυτό των κλειδιών έχει και ρόλο εξουσιοδότησης και ταυτοποίησης της πλατφόρμας (αποστολέα) αποστολής προωθημένων μηνυμάτων στα προαναφερόμενα οικοσυστήματα. Για τη δημιουργία των κλειδιών

αυτών θα χρησιμοποιηθεί μια βιβλιοθήκη γραμμένη σε Node.js. Η εγκατάσταση και η χρήση της οποίας παρουσιάζεται στον ακόλουθο Πίνακας 6.

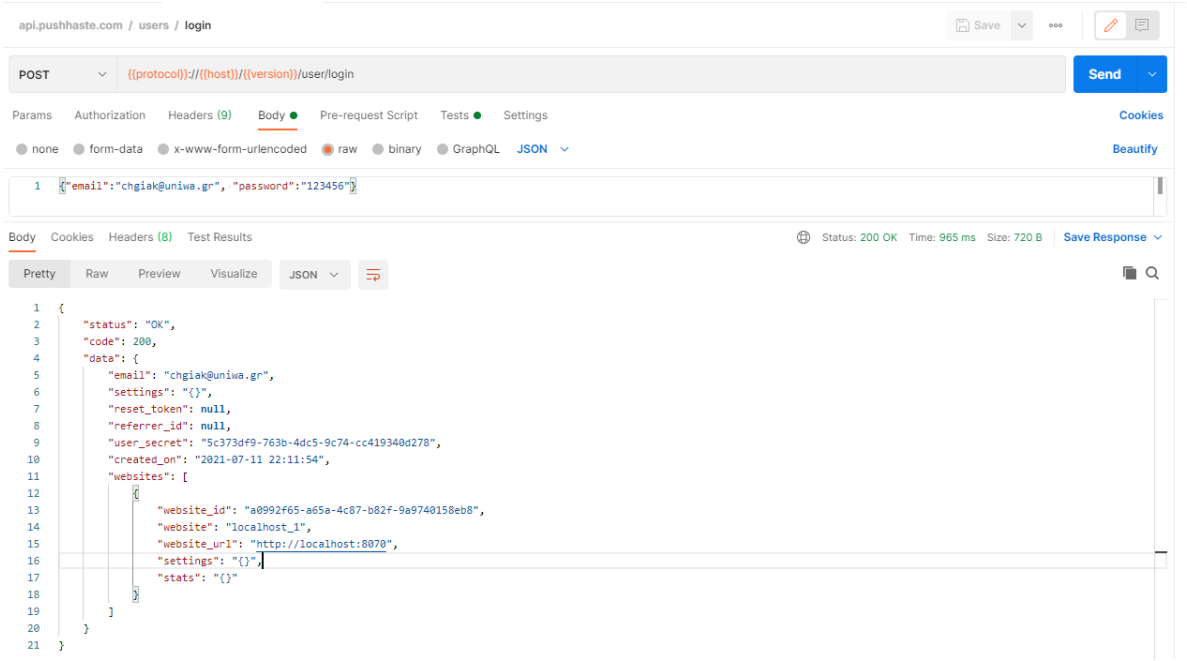
```
npm install -g web-push


web-push generate-vapid-keys


=====================================

Public Key:
BPoOtviwDDlaapToFvisTMNeOSG7eEJRKotOWnzAa0gtuMn_JuBm7yxWJvgCYjuFti0cWtJL3oXkSdSZ3
zh6Vt0

Private Key:
6aWyYz-5_OkDP6nxmgexR7UrWocs-PM8Qug91FChpIk

=====================================
```

Πίνακας 6 – γέννηση VAPID

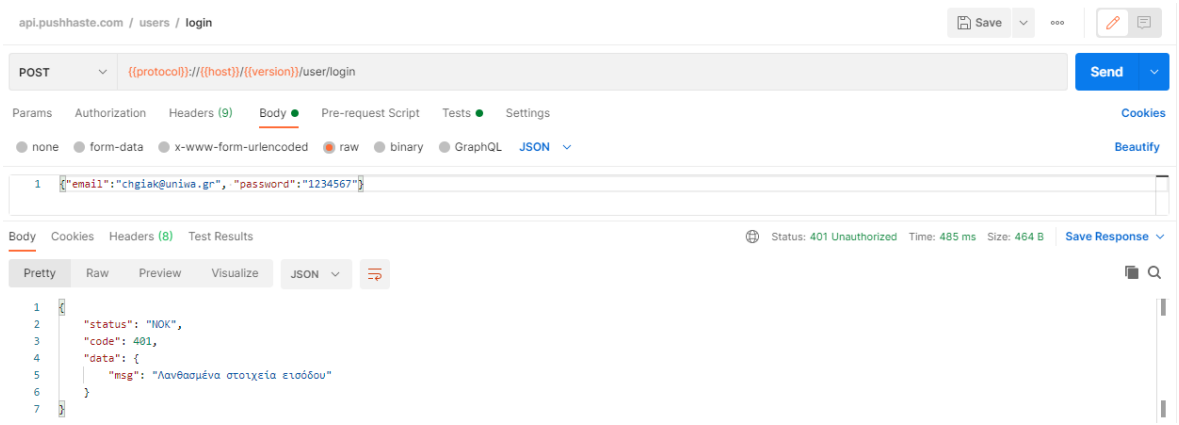
Το ιδιωτικό και το δημόσιο κλειδί χρησιμοποιείται για να κρυπτογραφηθούν και να αποκρυπτογραφούν αντίστοιχα τα μηνύματα προς αποστολή. Το ιδιωτικό κλειδί θα χρησιμοποιηθεί κατά την αποστολή των μηνυμάτων ενώ το δημόσιο κλειδί θα χρησιμοποιηθεί από τον service worker για να αποκρυπτογραφήσει το μήνυμα και να το παρουσιάσει στον τελικό χρήστη.

### 7.1.3 Δομή δεδομένων API

Το API περιλαμβάνει κλήσεις χωρισμένες σε πέντε κατηγορίες με βάση τη διαδικασία που εκτελούν. Όλες οι κλήσεις εκτός αυτές που αφορούν τον service worker επιστρέφουν δεδομένα τύπου json (application/json) έχουν την ακόλουθη δομή. Στην περίπτωση του service worker επιστέφει δεδομένα τύπου κειμένου javascript (application/javascript). Κάθε απάντηση τύπου json έχει τρία κλειδιά. Το κλειδί status που μπορεί να είναι OK ή N(ot)OK και υποδηλώνει ότι ολοκληρώθηκε επιτυχώς το αίτημα. Το κλειδί code περιέχει το ένα κωδικό σφάλματος αντίστοιχο με του πρωτοκόλλου http για λόγους ευκολίας αναγνώρισης. Τέλος το πεδίο data περιέχει τα δεδομένα προς απάντηση στο σχετικό αίτημα κατά περίπτωση. Στις ακόλουθη Εικόνα 15 και Εικόνα 16 παρουσιάζονται δυο περιπτώσεις επιτυχημένης και αποτυχημένης εκτέλεσης αιτήματος σύνδεσης ενός χρήστη διαχειριστή μέσω του API. Πρόκειται για την κλήση /v1/user/register με δεδομένα εισόδου ένα json που περιέχει ένα σωστό ζευγάρι email και password την πρώτη φορά ενώ ένα λανθασμένο τη δεύτερη φορά.



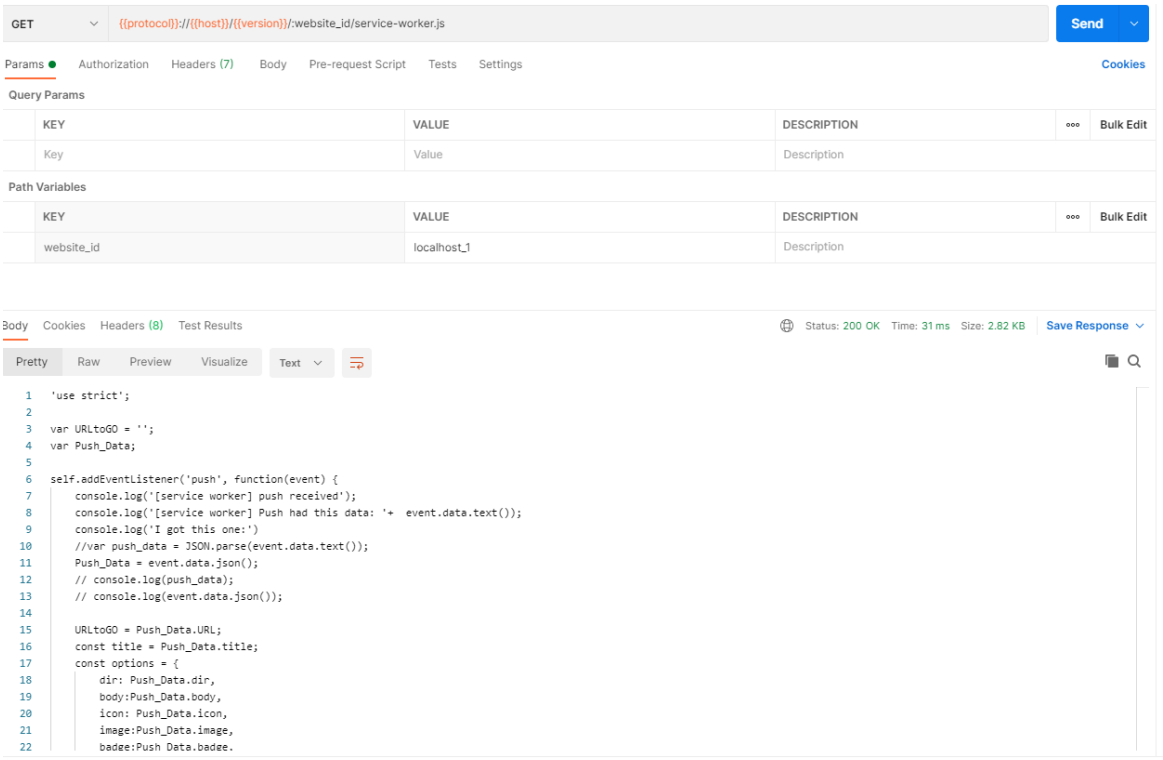

Εικόνα 15 – απάντηση επιτυχημένης εκτέλεσης αιτήματος σύνδεσης στο API



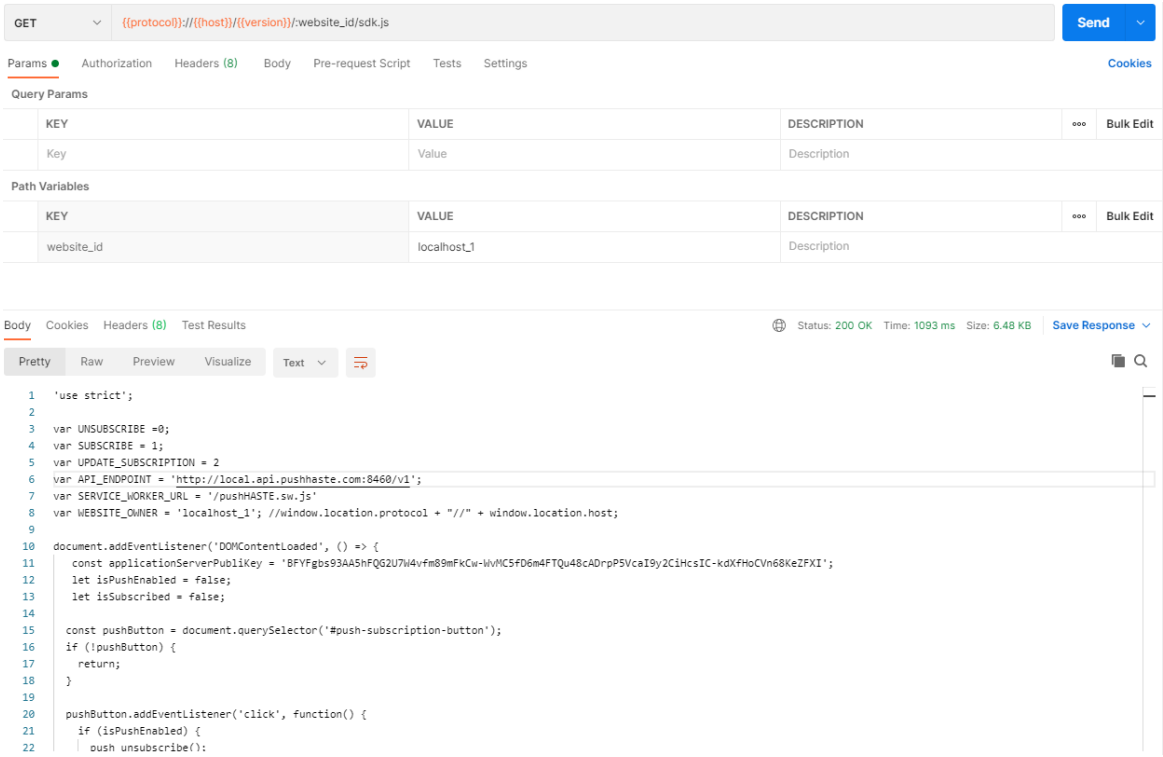

Εικόνα 16 – απάντηση αποτυχημένης εκτέλεσης αιτήματος σύνδεσης στο API

Όπως προαναφέρθηκε, εκτός από τα αιτήματα προς το API τύπου json υπάρχουν και δυο αιτήματα που επιστρέφουν δεδομένα τύπου application/javascript. Ο λόγος γίνεται για τον service worker και τις συνοδευτικές συναρτήσεις που απαιτούνται για να λειτουργήσει ο μηχανισμός εγγραφής/απεγγραφής ενός χρήστη όπως επίσης και λήψης ενός προωθημένου μηνύματος. Και τα δυο αυτά αιτήματα είναι τύπου GET για να μπορούν αν εκτελεστούν απλά με κλήση του ονόματος τους από οποιοδήποτε σημείο και δέχονται σαν όρισμα εντός του URL τους ένα λεκτικό αναγνωριστικό, ξεχωριστό για κάθε ιστότοπο. Το σύνολο των συνοδευτικών συναρτήσεων περιλαμβάνονται σε ένα αρχείο sdk.js που γεννιέται από τον API. Παραδείγματα χρήσης των δυο προαναφερόμενων κλήσεων παρουσιάζονται αντίστοιχα στην Εικόνα 17 και στην Εικόνα 18. Στην Εικόνα 17 παρουσιάζεται το αποτέλεσμα της κλήσης /v1/localhost_1/service-worker.js όπου localhost_1 είναι το αναγνωριστικό πεδίο κλειδί ενός ιστότοπου. Το αποτέλεσμα είναι τύπου

κειμένου και πιο συγκεκριμένα είναι application/Javascript. Αντίστοιχης λογικής είναι και η κλήση στο /v1/localhost_1/sdk.js όπου επιστρέφει κώδικα javascript κατάλληλα διαμορφωμένο για να διαχειριστεί ο ιστότοπος τον service worker.




Εικόνα 17 – ανάκτηση service worker από API




Εικόνα 18 – ανάκτηση συνοδευτικών συναρτήσεων (sdk) κατάλληλα διαμορφωμένο από τον API

### 7.1.4 Ταυτοποίηση χρηστών API

Η ταυτοποίηση επιτυγχάνεται με τη χρήση ενός επιπλέον header στα πακέτα http του πρωτοκόλλου tcp. Κάθε κλήση εγγραφής (/v1/users/register) ή σύνδεσης (/v1/users/login) επιστρέφει ένα κλειδί στο json που ονομάζεται user_secret όπως φαίνεται στην Εικόνα 15 και στην Εικόνα 16 αντίστοιχα. Στη συνέχεια η επισύναψη αυτή του κλειδιού στις κεφαλίδες των πακέτων http σε κάθε αίτημα προς το API ταυτοποιεί το χρήστη του API και επιτρέπει την εκτέλεση της κάθε διαδικασίας του API.

Αξίζει να σημειώσουμε ότι για λόγους ασφαλείας δεν επιστρέφεται το user_id του χρήστη αλλά ένα μοναδικό αναγνωριστικό (user_secret) που ο χρόνος ζωής μπορεί να αλλάξει ανάλογα τις ανάγκες του API.

Για λόγους ασφαλείας δε συνιστάται να εμφανίζεται το user_id ή το μοναδικό αναγνωριστικό πεδίο που συνδέεται ή αντιπροσωπεύει το πρωτεύον κλειδί κάποιας εγγραφής στη βάση, καθώς αν εκτεθούν αυτά τα δεδομένα ενδέχεται η κακόβουλη χρήση του API να αποκαλύψει ευπάθειες του συστήματος. Αντί αυτού χρησιμοποιείται το user_secret.

Αν το API είναι δημόσιο το user_secret δύναται να αλλάζει κάθε φορά που συνδέεται ένας χρήστης, υπολογίζοντας το με βάσει κάποια μαθηματική συνάρτηση ή με τυχαίο τρόπο. Παράλληλα δύναται ακόμα και να «πεθαίνει» (παύει να υπάρχει το user_secret) μετά από κάποιο χρονικό περιθώριο, πράγμα που σημαίνει ότι θα απαιτείται εκ νέου ταυτοτποίηση χρήστη.

Στην περίπτωση του συγκεκριμένου έργου οι αρχικές ανάγκες σε επίπεδο ασφαλείας είναι χαμηλότερες καθώς πρόκειται για ιδιωτικής και αποκλειστικής χρήσης API το ίδιο το πληροφοριακό σύστημα προκειμένου να δημιουργήσει και να διαχειριστεί τα προωθημένα μηνύματα. Παρόλα αυτά το user_secret αλλάζει κάθε φορά που συνδέεται ο χρήστης μέσω του API με τυχαίο τρόπο.

Επιπροσθέτως, για λόγους ασφαλείας έχει επιλεγεί όλα τα δεδομένα του συστήματος που να εκπροσωπούνται από Universally Unique IDentidier (UUID), έκδοσης 4. Αυτό σημαίνει ότι όλα τα πρωτεύον κλειδιά στο φυσικό επίπεδο της βάσης δεδομένων εκφράζονται με αλφαριθμητικά hashes μη σειριακά, καθιστώντας με αυτό τον τρόπο αδύνατη την εύρεση του προηγούμενη ή επόμενο hash (άρα και αντίστοιχης εγγραφής) σε περίπτωση που κάποιος μπορέσει να βρει ένα uuid, όπως αυτό του λογαριασμού του.

Τέλος αξίζει να σημειωθεί ότι, καθώς πρόκειται για ιδιωτικό API τα δεδομένα που παρουσιάζονται στην παρούσα τεκμηρίωση δεν είναι ορατά πότε στον τελικό χρήστη αλλά είναι μόνο προσβάσιμα και διαχειρίσιμα μόνο από το ίδιο το σύστημα που κάνει χρήση του API.

### 7.1.5 Αποστολή προωθημένων μηνυμάτων

Ενδιαφέρον παρουσιάζει ο μηχανισμός αποστολής προωθημένων μηνυμάτων που αναπτύχθηκε στο API. Η ανάθεση αποστολής ενός προωθημένου μηνύματος αρχική γίνεται με την κλήση /v1/campaign/{website_id} που είναι τύπου [POST]. Η κλήση αυτή επιτρέπεται να πραγματοποιηθεί μόνο από ταυτοποιημένους χρήστες του API. Η ταυτοποίηση γίνεται μέσω του πεδίου token που ενσωματώνεται στην κεφαλίδα κάθε αιτήματος αποστολής μηνυμάτων. Εντός του URL ενσωματώνεται και το λεκτικό-αναγνωριστικό του εκάστοτε ιστότοπου, π.χ. /v1/campaign/localhost_1.

Σε αυτή την περίπτωση αφού ελεγχθεί ότι ο χρήστης είναι ταυτοποιημένος ελέγχεται αν έχει αν του ανήκει ο εν λόγω ιστότοπος (localhost_1). Αν δε του ανήκει, το API του επιστρέφει το ανάλογο σφάλμα.

Σε περίπτωση που περάσουν όλοι οι έλεγχοι ελέγχου πρόσβασης για το χρήστη επιτυχώς, ανακτώνται από τη βάση δεδομένων όλοι οι πληροφορίες και αποθηκεύονται στη redis. Το μήνυμα αποθηκεύεται σε ένα κλειδί στη redis ενώ οι αποδέκτες σε μια στοίβα στη redis.

Μια διαδικασία που εκτελείται συνέχεια αναλαμβάνει να ανακτήσει από τη redis το μήνυμα και έναν αποδέκτη τη φορά από τη στοίβα και να το προωθήσει προς αποστολή στο κατάλληλο οικοσύστημα. Η διαδικασία αυτή θα λειτουργεί ως δαίμονας και θα εκτελείται συνεχώς. Όταν τερματίζεται θα ξεκινάει πάλι. Η επικοινωνία με το κατάλληλο οικοστύστημα γίνεται με τη βοήθεια της βιβλιοθήκης ανοιχτού και ελεύθερου κώδικα minishlink/web-push-php.

Στο API φτιάχτηκαν δυο κλήσεις για αποστολή προωθημένων μηνυμάτων. Αρχικά φτιάχτηκε μια κλήση [POST] /v1/nitification/push. Η κλήση αυτή στοχεύει στην αποστολή ενός προωθημένου μηνύματος σε συγκεκριμένο αποδέκτη. Δέχεται σαν όρισμα τον αποδέκτη και το προωθημένο μήνυμα με τα αντίστοιχα πεδία του.


```
{"endpoint":"https://fcm.googleapis.com/fcm/send/eFOdchKv5zE:APA91bGRIdVovv1HaPmJUvt2OoaHHRFLxyLq4LdA8BzAm0MWauBxKHgWxH4JAe9iBKbCVaOfrWHL9zojPE-V3BaT7glYLIE_efkjHzVEcCkrS0t4jlri6z9GsgR-rQcYzIhohZ7remLo","expirationTime":null,"keys":{"p256dh":"BMq2XA+tR7yFs0hp7l88pSymsrY8pA59F7nxhQobNKWiF5E14zLXg8cW8aidNE2hgTyZbltWjaDEofNBZNM+kV4=","auth":"61KxYtxTxNkw/rln/nqxCQ=="},"contentEncoding":"aes128gcm", "payload":{
    "title":"Θα θέλατε να εμβολιαστείτε κατά του COVID-19;",
    "dir": "auto",
    "body":"Ο εμβολιασμός κατά της COVID-19 μειώνει τον κίνδυνο να νοσήσετε με COVID-19. Προστατεύει εσάς και τους οικείους σας.",
    "icon": "https://www.durham.ca/en/health-and-wellness/resources/Images/Pop-Up-Icon.png",
    "image":"https://www.naftemporiki.gr/fu/p/1612868/638/399/0x000000000187c219/2/koronoios-embolio.jpg",
```

```
    "badge":"https://www.durham.ca/en/health-and-wellness/resources/Images/Pop-Up-
Icon.png",
    "tag":null,
    "vibrate": [500,110,500,110,450,110,200,110,170,40,450,110,200,110,170,40,500],
    "timestamp": null,
    "renotify": true,
    "actions": [
      {
        "action": "vaccine-yes",
        "title": "Ναι",
        "icon": "https://www.shareicon.net/data/256x256/2016/08/20/817720_check_395x512.
png",
        "url":"http://localhost:8070/vaccine/yes"
      },
      {
        "action": "vaccine-no",
        "title": "Όχι",
        "icon": "https://findicons.com/files/icons/1008/quiet/256/no.png",
        "url": "http://localhost:8070/vaccine/no"
      },
      {
        "action": "vaccine-maybe",
        "title": "Δε ξέρω",
        "icon": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQL_dWZpFDZWp8X-
5eueVpodAbNCK_cndRDgw&usqp=CAU",
        "url": "http://localhost:8070/vaccine/maybe"
      }
      ],
    "silent": false,
    "URL": "http://localhost:8070/vaccine",
    "requireInteraction": true,
    "sticky": true,
    "notificationCloseEvent": false,
    "data": {}
    }
 }
```


Πίνακας 7 – αίτημα αποστολής μηνύματος σε ένα αποδέκτη

Στον παραπάνω Πίνακας 7 παρουσιάζεται το αίτημα στον API για αποστολή ενός μηνύματος σε έναν αποδέκτη. Ενώ στην ακόλουθη Εικόνα 19 παρουσιάζεται το προωθημένο μήνυμα του παραπάνω αιτήματος όπως λήφθηκε από τον αποδέκτη.

Στην συνέχεια υλοποίησης του έργου φτιάχτηκε και ένα δεύτερο αίτημα όπου δημιουργείται μια καμπάνια με την κλήση [POST] /v1/campaign/{website_id} όπου αποθηκεύεται το αίτημα στη βάση δεδομένων και για κάθε αποδέκτη του ιστοτόπου που εκφράζεται από την παράμετρο {website_id} προστίθενται σε μια στοίβα ανάθεση μιας εργασίας για την αποστολή του συγκεκριμένου μηνύματος. Η στοίβα υλοποιείται με τη Redis. Ένα αρχείο php θα καλείται συνέχεια

σαν “δαίμονας» και κάθε φορά που εκτελείται ανακτά και εκτελεί μια τέτοια ανάθεση (αποστολή) από τη λίστα, εκτελώντας τον ίδιο κώδικα με το κώδικα της κλήσης /v1/notification/push.


Εικόνα 19 – παράδειγμα προωθημένου μηνύματος

## 7.2 Service Worker και συνοδευτικό SDK

Το μέρος του λογισμικού που θα τρέχει σε κάθε ιστότοπο και θα είναι υπεύθυνο για την εγγραφή επισκεπτών του εν λόγω ιστότοπου στην υπηρεσία, όπως επίσης και την εμφάνιση των σχετικών μηνυμάτων ως αναδυόμενες ειδοποιήσεις είναι ο service worker. Αναλυτική περιγραφή του service worker έγινε στην παράγραφο 2.5.

Σύμφωνα με την υλοποίηση του παρόντος έργου μέσω του service worker και των συνοδευτικών javascript συναρτήσεων (sdk.js) εκτελούνται οι ακόλουθες διαδικασίες

- Δ1. Εγγραφή χρήστη (περιηγητή) στην υπηρεσία
- Δ2. Λήψη και προβολή εισερχόμενου μηνύματος ως αναδυόμενη ειδοποίηση
- Δ3.Απεγγραφή περιηγητή από την υπηρεσία

### 7.2.1 Συνοδευτικές συναρτήσεις του εκτός του Service Worker (SDK.js)

Αρχικά μέσω του sdk.js ο service worker εγκαθίσταται στον περιηγητή. Η εγκατάσταση του service worker απαιτεί ο ιστότοπος που θα εκτελεστεί να λειτουργεί κάτω από το πρωτόκολλο https. Θα πρέπει να διαθέτει ένα έγκυρο πιστοποιητικό ασφαλείας. Εξαίρεση αποτελεί ο ιστότοπος που τρέχει τοπικά σε έναν εξυπηρετητή κάτω από το hostname localhost. Για την εγκατάσταση του service worker απαιτείται η συγκατάθεση του χρήστη, όπου δίνεται με έκκριση σε αναδυόμενο παράθυρο. Ο κώδικας του sdj.js που είναι υπεύθυνος για την εγγραφή του χρήστη είναι ο ακόλουθος όπως φαίνεται στον Πίνακας 8.

```
navigator.serviceWorker.register(SERVICE_WORKER_URL)
    .then(function(SWRegistration) {
         console.log('[SW] Service worker has been registered', SWRegistration);
      subscribeUser(UPDATE_SUBSCRIPTION);
      //push_updateSubscription();
    })
    .catch(function(error) {
        console.error('[SW] Service worker registration failed', error);
    }
);
```

Πίνακας 8 – κώδικας εγκατάστασης service worker

Αφού εγκατασταθεί ο service worker καλείται μια συνάρτηση (subscribeUser(SUBSCRIBE{=1}) για να εγγράψει τον περιηγητή στην υπηρεσία. Το sdk.js περιέχει δυο συναρτήσεις που ανάλογα με το όρισμα που δέχονται εγγράφουν, επικαιροποιούν ή απεγγράφουν το χρήστη. Η κατάσταση μια συνδρομής μπορεί να είναι οι ακόλουθες όπως φαίνεται στον ακόλουθο Πίνακας 9.

| **Σταθερά** | **τιμή** |
|---|---|
| *UNSUBSCRIBE* | 0 |
| *SUBSCRIBE* | 1 |
| *UPDATE_SUBSCRIPTION* | 2 |

Πίνακας 9 – καταστάσεις εγγραφής (συνδρομής) περιηγητή

Για να εγγραφεί ένας χρήστης θα πρέπει να εκτελεστεί η συνάρτηση subscribeUser() με όρισμα *SUBSCRIBE* (=1). Η συνάρτηση subscribeUser() αφού ελέγξει εάν έχει εγκατασταθεί ο service worker σωστά και υπάρχει η συγκατάθεση του χρήστη καλεί με τη σειρά της την subscribe_through_api() με το ίδιο όρισμα όπως και η subscribeUser.

| | |
|---|---|
| subscribeUser(*UNSUBSCRIBE*) | απεγγράφει ένα χρήστη από την υπηρεσία |
| subscribeUser(*SUBSCRIBE*) | εγγράφει ένα χρήστη στην υπηρεσία |

| | |
|---|---|
| subscribeUser(*UPDATE_SUBSCRIPTION*) | επικαιροποιηεί (εγγράφει) ένα χρήστη στην υπηρεσία |

Πίνακας 10 – εγγραφή / απεγγραφή χρήστη

Η τελευταία ανάλογα με την τιμή του ορίσματος [0,1,2] επικοινωνεί με το API σύμφωνα με τη διαδικασία Δ1 και εγγράφει, ενημερώνει ή απεγγράφει το χρήστη από την υπηρεσία. Ο κώδικας της subscribe_through_api() παρουσιάζεται στον Πίνακας 11.

```
function subscribe_through_api(subscription, action = SUBSCRIBE) {
  if (subscription) {
    const p256dh = subscription.getKey('p256dh');
    const authorisation_key = subscription.getKey('auth');
    const key = subscription.getKey('p256dh');
    const token = subscription.getKey('auth');


    switch (action) {
      case SUBSCRIBE:
      case UPDATE_SUBSCRIPTION:
      default:
        var _api_url = "/device/subscribe";
        var _method = 'post';
        break;
      case UNSUBSCRIBE:
        var _api_url = "/device/unsubscribe";
        var _method = 'delete';
    }
    fetch(API_ENDPOINT + _api_url, {
      method: _method,
      headers: new Headers({'Content-Type': 'application/json'}),
      body: JSON.stringify({
        endpoint: subscription.endpoint,
        //p256dh: p256dh ? btoa(String.fromCharCode.apply(null, new
Uint8Array(subscription.getKey(p256dh)))) : null,
        //authorisation_key: authorisation_key ?
btoa(String.fromCharCode.apply(null, new
Uint8Array(subscription.getKey(authorisation_key)))) : null
        authorisation_key: key ? btoa(String.fromCharCode.apply(null, new
Uint8Array(key))) : null,
        authToken: token ? btoa(String.fromCharCode.apply(null, new
Uint8Array(token))) : null,
        website: WEBSITE_OWNER
      })
    }).then(function(response) {
      return response.text();
    }).then(function(response) {
      console.log(response);
    }).catch(function(error) {
      console.log(error);
    })
  }
}
```

Πίνακας 11 – συνάρτηση επικοινωνίας μας το API

### 7.2.2 Service Worker

Το σημαντικότερο κομμάτι για του συστήματος είναι ο service worker πάνω στο οποίο βασίστηκε και σχεδιάστηκε όλη η υπηρεσία των προωθημένων μηνυμάτων. Ο ίδιος ο service worker είναι σε θέση να διαχειριστεί το περιεχόμενο αλλά και τους πόρους (εικόνες, javascript, css αρχεία ενός ιστότοπου).

Στην προκειμένη περίπτωση παρακολουθεί συνεχώς ένα συμβάν που πυροδοτείται όταν λάβει μέσω του πρωτοκόλλου Push API του http ένα μήνυμα.

```
self.addEventListener('push', function(event) {
    console.log('[service worker] push received');
    console.log('[service worker] Push had this data: '+  event.data.text());
    Push_Data = event.data.json();

    URLtoGO = Push_Data.URL;
    const title = Push_Data.title;
    const options = {
        dir: Push_Data.dir,
        body:Push_Data.body,
        icon: Push_Data.icon,
        image:Push_Data.image,
        badge:Push_Data.badge,
        tag: Push_Data.tag,
        vibrate: Push_Data.vibrate,
        timestamp: null,
        renotify: Push_Data.renotify,
        actions: Push_Data.actions,
        silent: Push_Data.silent,
        requireInteraction: Push_Data.requireInteraction,
        sticky: Push_Data.sticky
    }
    event.waitUntil(self.registration.showNotification(title, options));
});
```

Πίνακας 12 - συμβάν που πυροδοτείται όταν ληφθεί ένα προωθημένο μήνυμα

Τέλος για να ολοκληρωθεί η λειτουργικότητα ενός προωθημένου μηνύματος πρέπει να είναι σε θέση να ανοίξει κάποιο ή κάποιους συνδέσμους ανάλογα με τις επιλογές ανάδρασης που έχει το σχετικό προωθημένο μήνυμα. Για το άνοιγμα ενός url με ανάδραση του προωθημένου μηνύματος χρειάζεται να γίνει πυροδότηση του συμβάντος onnotificationclick(), ο κώδικας του οποίου βρίσκεται και αυτός μέσα στο service worker. Στον ακόλουθο πίνακα φαίνεται ο κώδικας του προαναφερόμενης συνάρτησης. Το συμβάν αυτό πυροδοτείται κάθε φορά που γίνεται κλικ πάνω στην αναδυόμενη ειδοποίηση. Θα πρέπει να εξεταστεί αν πατήθηκε κάποιος κουμπί ή αν πατήθηκε η ειδοποίηση σε κάποιο άλλο σημείο.

Αξίζει να σημειωθεί ότι οι προδιαγραφές των προωθημένων μηνυμάτων δεν ενσωματώνουν άνοιγμα κάποιες διεύθυνσης URL με το πάτημα ενός κουμπιού στο αναδυόμενο παράθυρο. Η επιλογή του κουμπιού υπάρχει για να δώσει τη δυνατότητα στον ιστότοπο από όπου έχει προέλθει

το μήνυμα να πυροδοτήσει κάποια συνάρτηση (κώδικα) γραμμένη σε javascript προκειμένου να εκτελέσει κάποια ενέργεια. Η εν λόγω συνάρτηση θα πρέπει να είναι μέρος του service worker και στα πλαίσια αυτής της εργασία είναι αδύνατο να ενσωματωθεί κώδικα στον service worker από τη που ο service worker γεννιέται αυτόματα από το API. Τέτοιου είδους υλοποίηση θα απαιτούσε να γραφεί για κάθε ιστότοπο ξεχωριστός service worker που να ενσωματώνει όλες τις ενέργειες που θα θέλαμε να κάνει το κάθε κουμπί. Ενδεχομένως να ήταν πολύπλοκος κώδικας και μακροσκελής. Σε κάθε περίπτωση είναι εκτός του προσανατολισμού και του στόχου του όλου έργου.

Για το λόγω αυτό περάστηκαν μέσα από τα προωθημένα μηνύματα επιπλέον πληροφορίες για να συμπεριληφθεί και ένα URL ανακατεύθυνσης για κάθε κουμπί. Στη συνέχεια με ειδική διαχείριση του συμβάντος κλικ στην αναδυόμενη ειδοποίηση ελέγχεται αρχικά αν έχει πατηθεί κάποιο πλήκτρο και ανοίγει ο αντίστοιχος σύνδεσμος, εναλλακτικά ανοίγει ο σύνδεσμος που συνοδεύει το προωθημένο μήνυμα και που ενεργοποιείται όταν λάβει χώρα κλικ σε οποιοδήποτε σημείο πάνω στην ειδοποίηση (εικονίδιο φωτογραφία, τίτλο κείμενο). Ο κώδικας της διαχείρισης του συμβάντος κλικ στον αναδυόμενο παράθυρο παρουσιάζεται στον Πίνακας 13.

```
self.onnotificationclick = function(event) {
    let URLtoVisit = URLtoGO;


    //έλεγχος αν πατηθεί κουμπί ανάδρασης
    Push_Data.actions.forEach(function (action, index) {
        if (action.action == event.action)
            URLtoVisit = action.url;
    });

    const urlToOpen = new URL(URLtoVisit, self.location.origin).href;

    const promiseChain = clients.matchAll({
        type: 'window',
        includeUncontrolled: true
    })
        .then((windowClients) => {
            let matchingClient = null;

            //έλεγχος εάν ο σύνδεσμος που προκειται να επισκεφτούμε είνια ήδη
ανοικτός σε κάποια
            //καρτέλα του περιηγητή
            for (let i = 0; i < windowClients.length; i++) {
                const windowClient = windowClients[i];
                if (windowClient.url === urlToOpen) {
                    matchingClient = windowClient;
                    break;
                }
            }

            //αν υπάρχει ήδη ο ιστότοπος ανοιχτός τότε έρχεται στο προσκήνιο
            if (matchingClient) {
                return matchingClient.focus();
            } else {
```

```
                //αν δεν υπάρχει ανοιχτός ο ιστότοπος ανοίγει σε νέα καρτέλα
                return clients.openWindow(urlToOpen);
            }
        });

    event.waitUntil(promiseChain);
};
```

Πίνακας 13 – κώδικας ελέγχου του κλικ στην αναδυόμενη ειδοποίηση

**Συνολικά όλος ο κώδικας του sdk.js και του serviceworker.js παρουσιάζονται στο Κώδικας αποστολής προωθημένου μηνύματος – cli script «δαίμονας»**

```php
<?php

use \Minishlink\WebPush\WebPush as webpush;
use \Minishlink\WebPush\Subscription as subscription;

error_reporting(E_ERROR | E_PARSE);
sleep(1);
require_once(__DIR__ . '/../../vendor/autoload.php');


if (file_exists(__DIR__ . '/../../.env')) {
    //$dotenv = Dotenv\Dotenv::createMutable(__DIR__ . '/../../');
    $dotenv = Dotenv\Dotenv::createUnsafeMutable(__DIR__ . '/../../');
    $dotenv->load();
}
$settings = require_once(__DIR__ . '/../../config/settings.php');


$redis = new Predis\Client("{$settings['redis']['dsn']}");

$auth['VAPID'] = ['subject'=>$settings['VAPID']['subject'],
'publicKey'=>$settings['VAPID']['publicKey'],
'privateKey'=>$settings['VAPID']['privateKey']];
$webPush = new \Minishlink\WebPush\WebPush($auth);


$key = "NotificationsQueue";


//ανάκτηση αποδέκτη από στοίβα redis
$subscription_data = $redis->rpop($key);

if (!$subscription_data) {
    echo "sleeping";
    sleep(3);
}
else {
    $subscription_data = @unserialize($subscription_data);

    $website_id = $subscription_data['website_id'];
    unset ($subscription_data['website_id']);

    //ανάκτηση μηνύματος από redis
    $campaign_key = "setCampaignFromWebSiteID:{$website_id}";
    $campaign_data = $redis->get($campaign_key);
    $campaign_data = json_decode($campaign_data);

    $subscription = Subscription::create($subscription_data);
```

```
    $payload = $campaign_data->message;

    $report = $webPush->sendOneNotification(
        $subscription,
        $payload
    );

    $endpoint = $report->getRequest()->getUri()->__toString();

    $reply =  ["endpoint" => $endpoint, "subscriber"=>$report->getReason()];

    if ($report->isSuccess()) {
        $reply['msg'] = "[v] Message sent successfully for subscription.";
    } else {
        $reply['msg'] = "x] Message failed to sent for subscription.";
    }

    echo json_encode($reply, JSON_UNESCAPED_UNICODE);
}
```

Παράτημα Β.

## 7.3 Πλατφόρμα διαχείρισης και αποστολής μηνυμάτων [front-end]

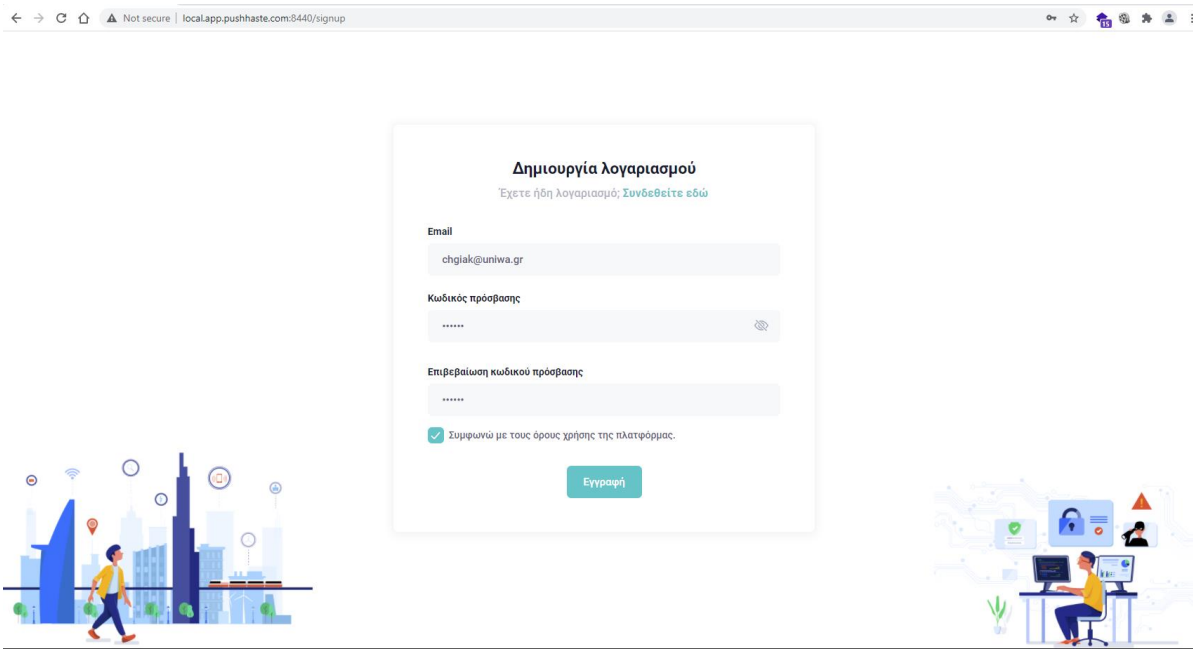
Για τις ανάγκες της παρούσας εργασίας υλοποιήθηκε ένα απλό διαχειριστικό περιβάλλον αποστολής προωθημένων μηνυμάτων σε Slim 4.0. Διαμορφώθηκαν σε περιβάλλον docker δυο ιδεατές υπηρεσίες app_nginx_1 και app_php-fpm_1. Η πρώτη ιδεατή υπηρεσία αφορά τον εξυπηρετητή των ιστοσελίδων. Ρυθμίστηκε να δουλεύει στην πόρτα 8440 ενώ η δεύτερη ιδεατή υπηρεσία αφορά την PHP 8.0 που θα δουλεύει συνδυαστικά με τον εξυπηρετητή.

Στη συνέχεια φτιάχτηκε ένα νέο έργο (Project με όνομα APP_pushHASTE όπου και θα περιλαμβάνει το εν λόγω απλό διαχειριστικό εργαλείο. Το διαχειριστικό εργαλείο επικοινωνεί με το API και μπορεί να πραγματοποιήσει 4 διεργασίες.

- Δημιουργία ενός νέου λογαριασμού
- Ταυτοποίηση χρήστη μέσω του λογαριασμού του
- Προσθήκη/εγγραφή ιστότοπων στην υπηρεσία προωθημένων μηνυμάτων
- Σύνθεση νέου προωθημένου μηνύματος.

### 7.3.1 Δημιουργία ενός νέου λογαριασμού

Για τη δημιουργία του λογαριασμού κατασκευάστηκε το view των σελίδων σε scripting γλώσσα twig. Σε επίπεδο περιηγητή ελέγχονται τα τιμές της φόρμας εγγραφής και στη συνέχεια στέλνεται ένα αίτημα προς το API με το email και το κωδικό πρόσβασης για να δημιουργηθεί ο νέος χρήστης. Η φόρμα εγγραφής έχει φτιαχτεί με τα ελάχιστα δυνατά πεδία που απαιτούνται προκειμένου να μπορεί κάποιος να γραφτεί συμπληρώνοντας μόνο ένα email και ένα κωδικό πρόσβασης, όπως ακριβώς γίνεται και με στη φόρμα εισόδου της επόμενης παραγράφου. Στην ακόλουθη Εικόνα 20 παρουσιάζεται η φόρμα εγγραφής στο διαχειριστικό περιβάλλον προωθημένων μηνυμάτων.



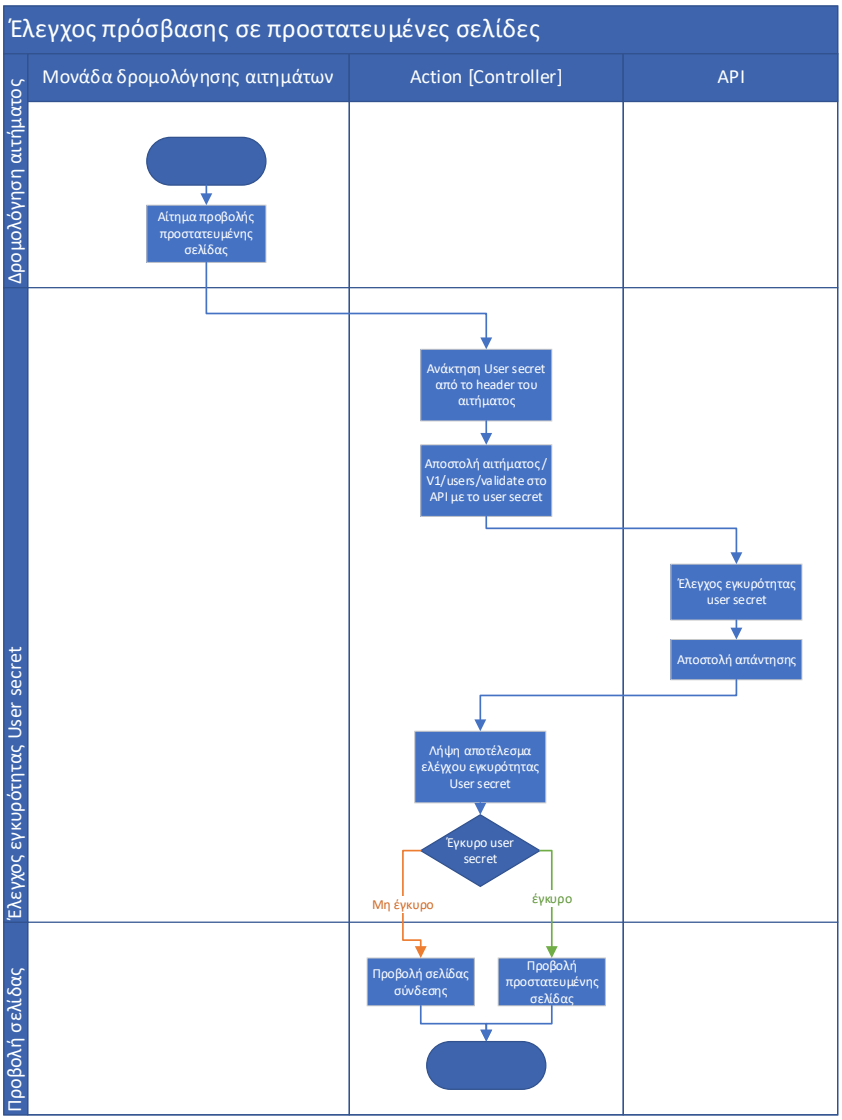

Εικόνα 20 - σελίδα εγγραφής την πλατφόρμα αποστολής μηνυμάτων

Αφού γίνει το αίτημα στο API, επιτυχώς, επιστρέφει πίσω σε μορφή json ένα token που θα χρησιμοποιηθεί στην πορεία ως αναγνωριστικό αλφαριθμητικό (user secret) για ταυτοποίησης του χρήστη (ο τρόπος λειτουργίας του API έχει αναλυθεί στην παράγραφο 7.1). Σε περίπτωση που τα στοιχεία είσοδού της φόρμα (username/password) υπάρχουν στο σύστημα τότε γίνεται ταυτοποίηση σύνδεση του χρήστη, πραγματοποιείται σύνδεση αντί εγγραφής και το API επιστρέφει πίσω το αντίστοιχο αναγνωριστικό αλφαριθμητικό (user secret).

### 7.3.2 Ταυτοποίηση χρήστη μέσω του λογαριασμού του

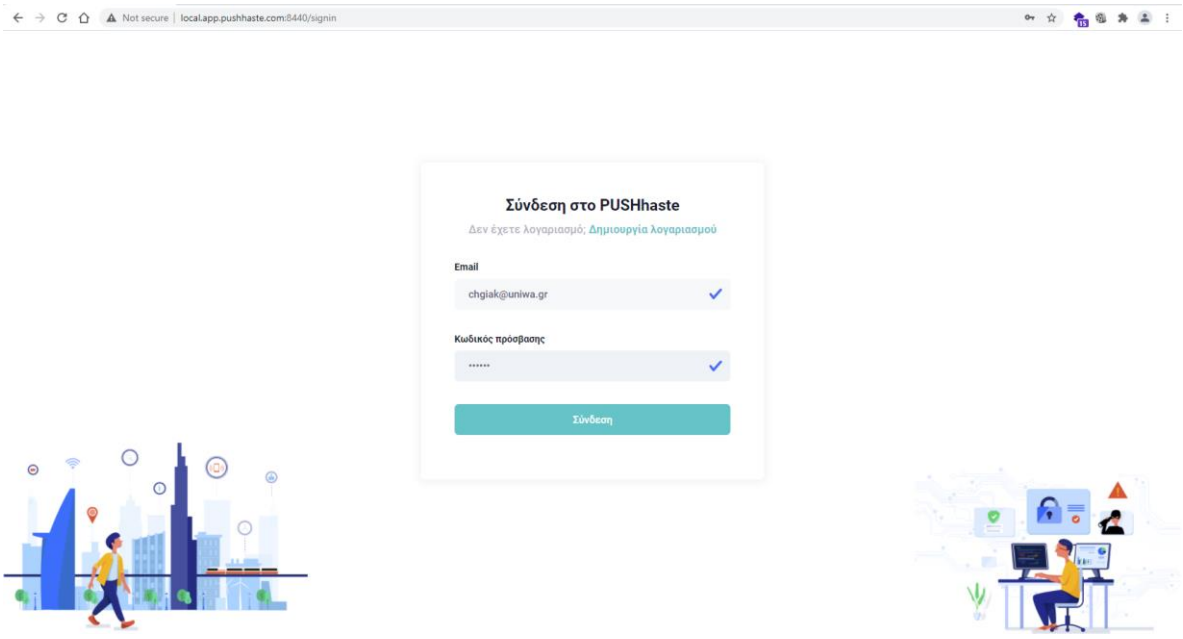
Για κάθε σελίδα του περιβάλλοντος διαχείρισης θα πρέπει να μπορεί να καλέσει το API και να επιβεβαίωση την ορθότητα των στοιχείων ταυτοποίησης του χρήστη μέσω του hash user secret, προκειμένου να επιτρέψει την πρόσβαση σε προστατευμένες σελίδες.

Στην προκειμένη περίπτωση η προστατευμένη σελίδα του διαχειριστικού περιβάλλοντος είναι μια και πρόκειται για τη σελίδα σύνθεσης προωθημένων μηνυμάτων. Το λογικό διάγραμμα λειτουργίας ελέγχου και προβολής προστατευμένων σελίδων απεικονίζεται στην ακόλουθη Εικόνα 21.




Εικόνα 21 - έλεγχος πρόσβασης σε προστατευμένες σελίδες

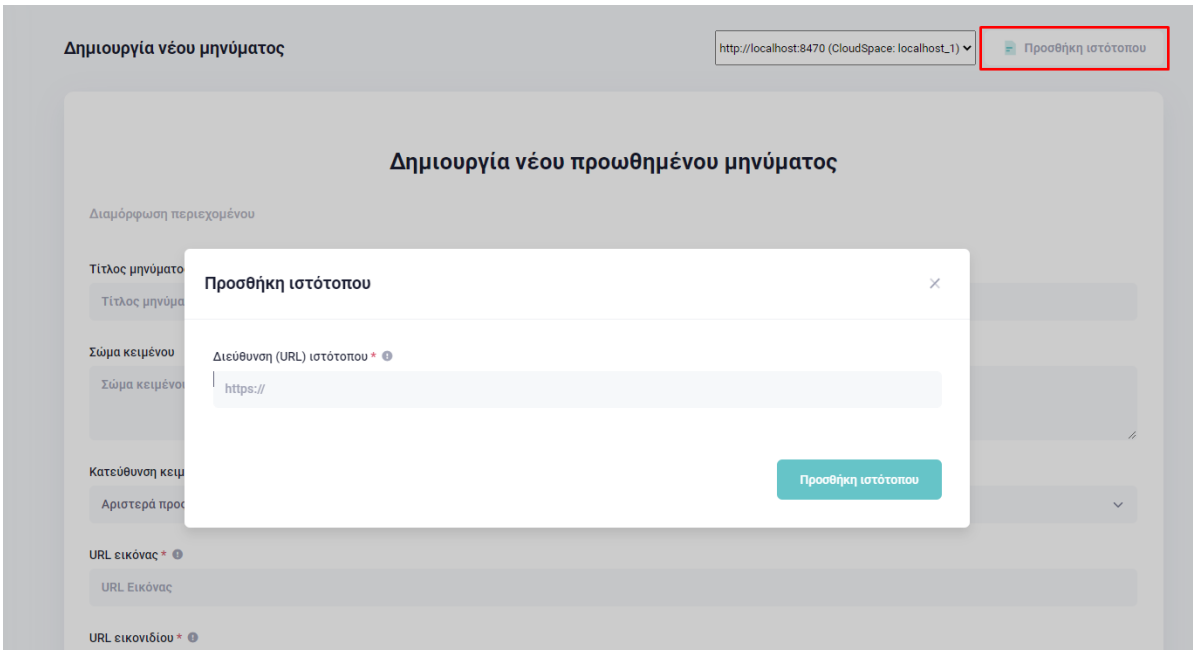
Βλέποντας τη διαδικασία στην πράξη η φόρμα σύνδεσης στην πλατφόρμα διαμορφώνεται όπως η Εικόνα 22.




Εικόνα 22 – σελίδα σύνδεσης στην πλατφόρμα

### 7.3.3 Προσθήκη/εγγραφή ιστότοπων στην υπηρεσία προωθημένων μηνυμάτων

Αφού συνδεθεί επιτυχώς ένας χρήστης στο περιβάλλον αποστολή προωθημένων μηνυμάτων, θα πρέπει αρχικά να προσθέσει έναν ιστότοπο.



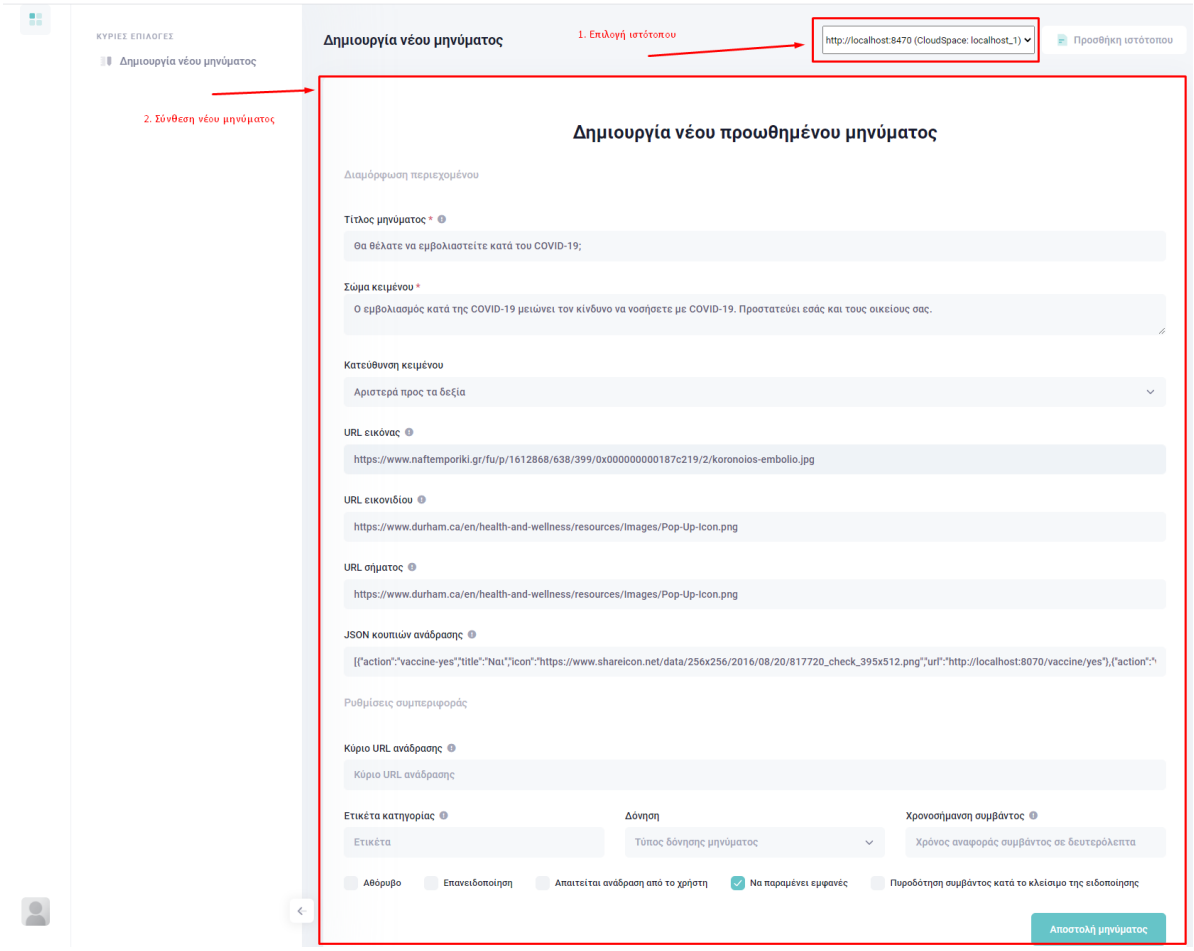

Εικόνα 23 – προσθήκη ιστότοπου

Για την προσθήκη ενός νέου ιστότοπου θα πρέπει γίνει κλήση στο API στο /v1/websites δίνοντας σαν δεδομένα μέσα σε json το URL του νέου ιστότοπου. Η επιτυχημένη προσθήκη του ιστότοπου θα μας επιτρέψει να αποστείλουμε μαζικά προωθημένα μηνύματα στους χρήστες του εν λόγω ιστοτόπου.

### 7.3.4 Σύνθεση νέου προωθημένου μηνύματος.

Αφού προστεθεί ένας ιστότοπος επιτυχώς, δύναται να σταλούν προωθημένα μηνύματα στους επισκέπτες του εν λόγω ιστότοπου που έχουν εγγραφεί στην υπηρεσία. Η αποστολή των μηνυμάτων γίνεται σε όλους του εγγεγραμμένους χρήστες του εκάστοτε ιστότοπου μέσα από μια επαναληπτική διαδικασία που λειτουργεί ως «δαίμονας». Σε κάθε εκτέλεση του στέλνει και ένα μήνυμα σε έναν παραλήπτη.

Για τη δημιουργία ενός νέου μηνύματος θα πρέπει να επιλεγεί ο σωστός ιστότοπος από το μενού του διαχειριστικού περιβάλλοντος και στη συνέχεια να συμπληρωθεί η φόρμα φαίνεται και στην ακόλουθη Εικόνα 24.




Εικόνα 24 – επιλογή ιστότοπου και συμπλήρωση φόρμας σύνθεσης νέου μηνύματος.

Με την αποστολή της παραπάνω φόρμας στέλνεται ένα αίτημα στον API στον ακόλουθο σύνδεσμο /v1/campaign/{website_id}. Όπως website_id είναι το CloudSpace του ιστότοπου. Για τις ανάγκες της παρούσας εργασίας έχει προστεθεί ο ιστότοπος http://localhost:8470 και έχει CloudSpace localhost_1. Το αίτημα που θα σταλεί από τη συμπληρωμένη φόρμα της Εικόνα 24 είναι το ακόλουθο στη διεύθυνση /v1/campaign/localhost_1.

```
{"endpoint":"https://fcm.googleapis.com/fcm/send/eFOdchKv5zE:APA91bGRIdVovv1HaPmJUvt2OoaHHRFLxyLq4LdA8BzAm0MWauBxKHgWxH4JAe9iBKbCVaOfrWHL9zojPE-V3BaT7glYLIE_efkjHzVEcCkrS0t4jlri6z9GsgR-rQcYzIhohZ7remLo","expirationTime":null,"keys":{"p256dh":"BMq2XA+tR7yFs0hp7l88pSymsrY8pA59F7nxhQobNKWiF5E14zLXg8cW8aidNE2hgTyZbltWjaDEofNBZNM+kV4=","auth":"61KxYtxTxNkw/rln/nqxCQ=="},"contentEncoding":"aes128gcm", "payload":{
    "title":"Θα θέλατε να εμβολιαστείτε κατά του COVID-19;",
    "dir": "auto",
    "body":"Ο εμβολιασμός κατά της COVID-19 μειώνει τον κίνδυνο να νοσήσετε με COVID-19. Προστατεύει εσάς και τους οικείους σας.",
    "icon": "https://www.durham.ca/en/health-and-wellness/resources/Images/Pop-Up-Icon.png",
    "image":"https://www.naftemporiki.gr/fu/p/1612868/638/399/0x000000000187c219/2/koronoios-embolio.jpg",
    "badge":"https://www.durham.ca/en/health-and-wellness/resources/Images/Pop-Up-Icon.png",
    "tag":null,
    "vibrate": [500,110,500,110,450,110,200,110,170,40,450,110,200,110,170,40,500],
    "timestamp": null,
    "renotify": true,
    "actions": [
      {
        "action": "vaccine-yes",
        "title": "Ναι",
        "icon": "https://www.shareicon.net/data/256x256/2016/08/20/817720_check_395x512.png",
        "url":"http://localhost:8070/vaccine/yes"
      },
      {
        "action": "vaccine-no",
        "title": "Όχι",
        "icon": "https://findicons.com/files/icons/1008/quiet/256/no.png",
        "url": "http://localhost:8070/vaccine/no"
      },
      {
        "action": "vaccine-maybe",
        "title": "Δε ξέρω",
        "icon": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQL_dWZpFDZWp8X-5eueVpodAbNCK_cndRDgw&usqp=CAU",
        "url": "http://localhost:8070/vaccine/maybe"
      }
      ],
    "silent": false,
    "URL": "http://localhost:8070/vaccine",
    "persistent": true,
    "requireInteraction": true,
    "sticky": true,
    "notificationCloseEvent": false,
    "data": {}
    }
 }
```

Πίνακας 14 – αίτημα στο API για αποστολή νέου μηνύματος στο localhost_1

## 7.4 Ενσωμάτωσης πλατφόρμας σε τρίτους ιστοτόπους

Η ενσωμάτωση της πλατφόρμας αποστολής προωθημένων μηνυμάτων είναι πολύ εύκολη. Ό όλος σχεδιασμός από την αρχή του έργου έγινε με γνώμονα την εύκολη εγκατάσταση του συστήματος σε τρίτους ιστότοπους. Για να μπορέσει κάποιος αν χρησιμοποιήσει την πλατφόρμα θα πρέπει να ικανοποιούνται οι ελάχιστες απαιτήσεις του συστήματος.

Εφόσον ικανοποιούνται οι ελάχιστές λειτουργικές απαιτήσεις όπως παρουσιάζονται στην ακόλουθη παράγραφο. Θα πρέπει να ακολουθηθούν να ενσωματωθεί το SDK.js της πλατφόρμας και ο αντίστοιχος service worker.

### 7.4.1 Ελάχιστες απαιτήσεις λειτουργίας του συστήματος

Η πλατφόρμα έχει σχεδιαστεί για να μπορεί να λειτουργεί χωρίς πρόβλημα σε όλα τα λειτουργικά συστήματα και σε όλους τους περιηγητές. Οι μόνες λειτουργικές απαιτήσεις του συστήματος προωθημένων μηνυμάτων είναι να μπορεί να λειτουργήσει το ίδιο το πρωτόκολλο προωθημένων μηνυμάτων. Για το σκοπό αυτό θα πρέπει

- Ο ιστότοπος που επιθυμεί να ενσωματώσει την υπηρεσία να διαθέτει πιστοποιητικό ασφαλείας και να λειτουργεί με το πρωτόκολλο https.
- Να μη διαθέτει άλλο εγκατεστημένο service worker ο ιστότοπο που θα εγγραφεί στην υπηρεσία.

Από την άλλη πλευρά ο τελικός χρήστης που θα είναι αποδέκτης των προωθημένων μηνυμάτων

- θα πρέπει να διαθέτει έναν περιηγητή που να μπορεί να εκτελέσει έναν service worker. Όλοι οι σύγχρονοι περιηγητές έχουν τη δυνατότητα να εκτελέσουν service workers σήμερα.

Σχεδόν όλοι οι σημερινοί εμπορικοί και όχι μόνο ιστότοποι διαθέτουν πιστοποιητικό ασφαλείας και λειτουργούν κάτω από το πρωτόκολλο https. Το μεγάλο πρόβλημα εντοπίζεται στο γεγονός ότι μπορεί να υπάρξει μόνο ένας service worker σε έναν ιστότοπο, οπότε η ύπαρξη υφιστάμενου service worker αποτρέπει την εγκατάσταση δεύτερου. Καθώς είναι μια νέα σχετικά τεχνολογία το θέμα του συνδυασμού πολλαπλών service worker με διαφορετικά συμβάντα πυροδότησης δεν έχει ακόμα λυθεί και όλοι οι σημερινοί περιηγητές μπορούν να εκτελούν μονάχα έναν service worker.

Αξίζει να σημειώσουμε ότι ειδικά για την περίπτωση του ιστότοπου http://localhost οι παραπάνω λειτουργικές απαιτήσεις δεν είναι απαραίτητο να ικανοποιούνται. Ο ιστοτόπος δεν έχει πιστοποιητικό ασφαλείας και δε λειτουργεί κάτω από το https πρωτόκολλο.

### 7.4.2 Βήματα ενσωμάτωσης και ενεργοποίησης πλατφόρμας

Αρχικά για να μπορέσει κάποιος να χρησιμοποιήσει την υπηρεσία θα πρέπει να εγγραφεί στο διαχειριστικό περιβάλλον του συστήματος, όπως περιγράφτηκε σε προηγούμενη παράγραφο. Έπειτα θα πρέπει να προσθέσει τη διεύθυνση του ιστότοπου του στο διαχειριστικό περιβάλλον.

Στη συνέχεια θα πρέπει να γίνουν δυο αλλαγές στον ιστότοπο του.

1. Να προστεθεί μια γραμμή κώδικα στην κεφαλίδα του ιστότοπου του, σε κάθε σελίδα όπως φαίνεται στον Πίνακας 15.
2. Να προστεθεί ένα αρχείο, με όνομα pushHASTE.sw.js στον αρχικό φάκελο του ιστότοπου (/pushHASTE.sw.js) ή όπως αλλιώς υπαγορεύεται μέσα στον πηγαίο κώδικα του sdk.js, με μια γραμμή κώδικα όπως φαίνεται στον Πίνακας 16.

Όπου **{{website_id}}** είναι το μοναδικό λεκτικό αναγνωριστικό κλειδί του ιστότοπου που πρόκειται να ενσωματώσει την υπηρεσία προωθημένων μηνυμάτων. Η πληροφορία του **{{website_id}}** είναι προσβάσιμη στο διαχειριστικό περιβάλλον της πλατφόρμας.

| |
|---|
| <script type="text/javascript" src="http://local.api.pushhaste.com:8460/v1/**{{website_id}}**/sdk.js"></script> |
| Πίνακας 15 - κλήση sdk.js |

| |
|---|
| self.importScripts('http://local.api.pushhaste.com:8460/v1/**{{website_id}}**/service-worker.js'); |
| Πίνακας 16 - αρχείο pushHASTE.sw.js |

Θα πρέπει να σημειωθεί ότι αυτή τη στιγμή για λόγους ανάπτυξης της πλατφόρμας το API του συστήματος λειτουργεί στη διεύθυνση http://local.api.pushhaste.com:8460. Σε πραγματικές συνθήκες θα πρέπει να αντικατασταθεί το κομμάτι αυτό της διεύθυνσης με το πραγματικό URL της υπηρεσίας API.

# 8 Συμπεράσματα

## 8.1 Γενικά συμπεράσματα

Στα πλαίσια της παρούσας διπλωματικής εργασίας αναλύθηκαν οι απαιτήσεις σχεδιασμού μιας νέας πλατφόρμας αποστολής προωθημένων μηνυμάτων. Σχεδιάστηκε και υλοποιήθηκε ένα νέο πληροφοριακό σύστημα ικανό να αποστείλει μαζικά προωθημένα μηνύματα σε επισκέπτες ενός ιστότοπου. Λαμβάνοντας υπόψη ότι μια υπηρεσία αποστολής προωθημένων μηνυμάτων θα πρέπει να μπορεί να αποστείλει δεκάδες χιλιάδες μηνύματα ημερησίως σχεδιάστηκε ένα μοντέλο βάσης δεδομένων αλλά και όλη η υπηρεσία για να μπορεί να κλιμακωθεί και να εκτελεστεί παράλληλα σε πολλαπλά μηχανήματα. Ο συγχρονισμός των δεδομένων σε μελλοντική επέκταση της δουλειάς αυτής είναι να μπορεί να πραγματοποιηθεί μέσω της redis.

Στο παρόν έργο σχεδιάστηκε ένα ιδιωτικό API ικανό να διαχειριστεί τις απαιτήσεις ενός συστήματος διαχείρισης προωθημένων μηνυμάτων πολλαπλών χρηστών και πολλαπλών ιστοτόπων. Οι κλήσεις στο API περιλαμβάνουν ελεύθερες κλήσεις από όλους αλλά και κάποιες που απαιτούν την ταυτοποίηση του χρήστη για να εκτελεστούν.

Τέλος το API σχεδιάστηκε με τέτοιο τρόπο ώστε όλο το κομμάτι που χρειάζεται να ενσωματωθεί στην πλευρά του χρήστη να γεννιέται αυτόματα από το API. Με αυτό τον τρόπο τα βήματα ενσωμάτωσης της υπηρεσίας περιορίζονται σε δυο γραμμές κώδικα ανά ιστότοπο.

Παράλληλα με το API σχεδιάστηκε και ένας service worker και οι απαραίτητες συνοδευτικές συναρτήσεις του (sdk). Στο τέλος της διαδικασίας ανάπτυξης τους, ενσωματώθηκαν στο API για να μπορούν να παράγονται αυτόματα και να ενσωματώνονται εύκολα στους ιστότοπους.

Τέλος για λόγους επίδειξης και μόνο αναπτύχθηκε η βασική δομή ενός περιβάλλοντος αποστολής προωθημένων μηνυμάτων που απευθύνεται σε διαχειριστές ιστοτόπων. Κάθε διαχειριστής δύναται να εισάγει πολλαπλούς ιστότοπους και στη συνέχεια αφού επιλέξει έναν ιστότοπο να μπορεί να στείλει ένα προωθημένο μήνυμα σε όλους τους εγγεγραμμένους χρήστες του ιστοτόπου στην εν λόγω υπηρεσία.

Το όλο έργο ολοκληρώθηκε επιτυχώς και όλοι οι αρχικοί στόχοι που είχαν τεθεί διαπεραιώθηκαν. Επιπροσθέτως, αποτελεί το εναρκτήριο λάκτισμα για την ανάπτυξη μιας ολοκληρωμένης υπηρεσίας αποστολής και διαχείρισης προωθημένων μηνυμάτων.

## 8.2 Περιορισμοί και δυσκολίες

Κατά τη διάρκεια σχεδιασμού και ανάπτυξης του έργου που πραγματεύεται η παρούσα διπλωματική, προέκυψαν διάφορά σχεδιαστικά προβλήματα αλλά και τεχνικά θέματά.

### 8.2.1 Σχεδιαστικοί περιορισμοί αρχιτεκτονικής

Αρχικά θα έπρεπε να σχεδιαστεί ένα σύστημα που να μπορεί να φιλοξενήσει χιλιάδες προωθημένα μηνύματα, καθώς ο αριθμός των εγγεγραμμένων χρηστών σε μια υπηρεσία προωθημένων μηνυμάτων τείνουν να είναι πολλαπλάσιοι σε σχέση με τους αντίστοιχους εγγεγραμμένους χρήστες λήψης ενημερωτικών μηνυμάτων μέσω ηλεκτρονικού ταχυδρομείου. Για το λόγω αυτό έπρεπε να γίνει κατάλληλός σχεδιασμό στη βάση δεδομένων προκειμένου να μειωθεί στο ελάχιστο οι φορές αλλά και ο χρόνος προσπέλασης στη βάση δεδομένων.

Ο σχεδιασμός της βάσης δεδομένων έγινε με τέτοιο τρόπο έτσι ώστε να μπορεί να ανακτήσει κάποιος τα απαραίτητα δεδομένα για κάθε αίτημα του API από τη βάση δεδομένων χωρίς να κάνει χρήση JOIN στα ερωτήματα στη βάση δεδομένων. Επιπροσθέτως χρησιμοποιήθηκε η υπηρεσία redis προκειμένου να φορτώνονται όλα στη ram μέσω της redis και να αποφεύγονται οι αναζητήσεις στη βάση δεδομένων.

Ο σχεδιασμός αυτός έγινε με γνώμονα ότι η υπολογιστή ισχύ μιας υπηρεσίας σε επίπεδο php/webserver (εξυπηρετητές) μπορεί εύκολα να πολλαπλασιαστεί, ενώ από την άλλη πλευρά μια κατανεμημένη βάση δεδομένων PostgreSQL είναι απαιτεί πολύ μεγάλο κόστος συντήρησης.

Επέκταση της συγκεκριμένης αρχιτεκτονικής θα συζητηθεί πιο κάτω στην παράγραφό που σχετίζεται με τις μελλοντικές επεκτάσεις της πλατφόρμας.

### 8.2.2 Τεχνικοί περιορισμοί

Ένα από τα σημαντικότερα προβλήματα που έχει το συγκεκριμένο πρωτόκολλο προωθημένων μηνυμάτων είναι η απαίτηση ύπαρξης ενός μόνο service worker. Επιπροσθέτως αξίζει να σημειωθεί και η τεχνική δυσκολία που αντιμετωπίστηκε στο να πραγματοποιηθούν αιτήματα από τρίτους ιστότοπους προς το API της πλατφόρμας καθώς για λόγους ασφαλείας δεν επιτρέπεται να γίνει σε επίπεδο javascript από τους περιηγητές.

#### 8.2.2.1 Μοναδικότητα service worker

Η τεχνολογία του service worker είναι σχετικά πρόσφατη. Δεν έχει ευρέως διαδοθεί και έχει επί τω πλείστων χρησιμοποιηθεί για την διαχείριση και υποστήριξη υπηρεσίας προωθημένων μηνυμάτων, με εξαίρεση ειδικές υλοποιήσεις όπως η βελτιστοποίηση εικόνων διαδικτύου.

Παρόλο που ένας service worker αιτείται πρόσβαση σε συγκεκριμένους πόρους του ιστότοπου ή λειτουργίες που σχετίζονται με το κώδικα του και το περιεχόμενο του, τεχνικά είναι αδύνατη η εκτέλεση δυο διαφορετικών service workers στον ίδιο ιστότοπο ακόμα και αν αιτείται ο κάθε ένας πρόσβαση δε διαφορετικούς πόρους. Οι τρέχουσες εκδόσεις των περιηγητών δεν κάνουν ακόμα διαχείριση των service workers και των σχετικών αιτημάτων τους για να μπορούν να συνδυαστούν και να εγκατασταθούν παραπάνω από ένας σε έναν ιστότοπο.

#### 8.2.2.2 Κλήση API από javascript τρίτων ιστότοπων

Ένα πρόβλημα που προέκυψε στη πορεία υλοποίησης του έργου ήταν να μπορεί ένας τρίτος ιστότοπος να καλέσει το API που της πλατφόρμας των μηνυμάτων. Για λόγους ασφαλείας όλοι οι περιηγητές επιτρέπουν κλήσεις σε AJAX (απευθείας κλήσεις αιτημάτων από javascript) μόνο σε αιτήματα που είναι κάτω από το ίδιο domain name.

Σε περίπτωση που κάποιος επιθυμεί να εκτελέσει ένα αίτημα σε άλλο σύστημα τότε αυτό μπορεί να γίνει σε επίπεδο server side γλώσσας (php / jsp / asp). Η javascript που εκτελείται στον περιηγητή του επισκέπτη για λόγους ασφαλείας απαγορεύεται να εκτελέσει αιτήματα προς «ξένα συστήματα».

Αρχικά το πρόβλημα εμφανίστηκε από το αίτημα εγγραφής και απεγγραφής συσκευών με τα σχετικά αιτήματα προς το API. Για την αντιμετώπιση του προβλήματος έπρεπε να τροποποιηθούν οι κεφαλίδες των πακέτων http στα εισερχόμενα αιτήματα για να μπορέσει να παρακαμφθεί ο σχετικός έλεγχος ασφαλείας. Προστέθηκαν κάποιες επιπλέον κεφαλίδες στα πακέτα των αιτημάτων http για να επιτρέπεται η εκτέλεση του API από οποιοδήποτε αιτούνται ακόμα και από ξένα συστήματα σε επίπεδο javascript.

## 8.3 Στόχοι που διεκπεραιώθηκαν

Κατά τη διάρκεια εκπόνησης της συγκεκριμένης διπλωματικής αναλύθηκαν απαιτήσεις , σχεδιάστηκε και υλοποιήθηκε μια πλατφόρμα διαχείρισης προωθημένων μηνυμάτων. Στα πλαίσια της διπλωματικής εργασίας σχεδιάστηκε ένα σύστημα που αποτελείται από τρεις λειτουργικές μονάδες. Ο κύριος στόχος της διπλωματικής ήταν να σχεδιαστεί ένα ολοκληρωμένο πληροφοριακό σύστημα αποστολής και λήψης προωθημένων μηνυμάτων.

Στα πλαίσια της παρούσας εργασίας σε επίπεδο αρχιτεκτονικής ικανοποιήθηκαν οι αρχικοί στόχοι του έργου και πιο συγκεκριμένα ολοκληρώθηκε επιτυχώς

- Ο σχεδιασμός ενός σχήματος βάσης δεδομένων με γνώμονα τη μορφή των δεδομένων που χρειάζεται να ανακτηθούν ανά αίτημα. Με αποτέλεσμα να μειωθεί σημαντικό ο χρόνος ανάκτησης της πληροφορίας και ο χρόνος προσπέλασης στη βάση δεδομένων.
- Χρησιμοποιήθηκε η υπηρεσία redis και η προσπέλαση στη βάση δεδομένων περιορίστηκε στο ελάχιστο δυνατό. Ενώ η ίδια η redis αποτελεί και ο τρόπος που χρησιμοποιήθηκε επιτρέπει τη μελλοντική κλιμάκωση της υπηρεσίας σε παράλληλα πολλαπλούς εξυπηρετητές, κάνοντας χρήση συγχρονισμένων redis.
- Ο τρόπος υλοποίησης του API και η αυτόματη γέννηση των σχετικών συναρτήσεων κατάλληλα διαμορφωμένες επιτρέπει την εύκολη ενσωμάτωση της πλατφόρμας με την προσθήκη δυο γραμμών κώδικα
- Ένα script που εκτελείτε συνεχώς σε μορφή «δαίμονα» και είναι σε θέση να στέλνει σε όλους τους αποδέκτες ενός ιστότοπου μαζικά ένα προωθημένο μήνυμα.
- Ένα ολοκληρωμένο σύστημα εγγραφής συσκευών και μαζικής αποστολής προωθημένων μηνυμάτων σε επισκέπτες ενός ιστότοπου. Η συμβατότητα του όλου συστήματος είναι 100% των γνωστών περιηγητών και το 96% όλων των περιηγητών ανεξαρτήτου εκδόσεων [8].
- Μηχανισμός αποστολής μαζικών προωθημένων μηνυμάτων. Επιλέγεται επιτυχώς το κατάλληλο οικοσύστημα ανάλογα με τον περιηγητή του επισκέπτη και τέλος στέλνεται το μήνυμα στον αποδέκτη.

Επιπροσθέτως σε λειτουργικό επίπεδο το πληροφοριακό σύστημα του σχεδιάστηκε φέρει επιτυχώς τις ακόλουθες λειτουργίες.

- Πολλαπλοί ιστότοποι μπορούν να εγγραφούν στην υπηρεσία, με απλή δημιουργία ενός λογαριασμού διαχειριστή στην πλατφόρμα.
- Ο τελικός χρήστης μπορεί να εγγράφεται εύκολα την υπηρεσία ενημέρωσης μέσω προωθημένων μηνυμάτων χωρίς να διαμοιράζει καμιά προσωπική πληροφορία που τον αντιπροσωπεύει, απλά δίνοντας την έγκριση του σε ένα αναδυόμενο παράθυρο.
- Κάθε προωθημένο μήνυμα δε μεταφέρει απλά μια πληροφορία με κείμενο και φωτογραφία αλλά δίνει τη δυνατότητα στο σχεδιαστεί του μηνύματος να ορίζει και κουμπιά ανάδρασης του τελικού χρήστη, όπου κάθε κουμπί οδηγεί σε διαφορετικό url.
- Από την πλευρά των διαχειριστών ιστότοπων, η πλατφόρμα έχει χρήστες διαχειριστές έναν ανά ιστότοπο

## 8.4 Μελλοντικές επεκτάσεις

Η παρούσα εργασία παρουσιάζει μια πλατφόρμα αποστολής προωθημένων μηνυμάτων. Ενσωματώνει τις βασικές λειτουργίες αποστολής μιας τέτοιας πλατφόρμας. Οι μελλοντικές επεκτάσεις που μπορούν να γίνουν είναι πολλαπλές, τόσο στη βελτίωση των υφιστάμενων λειτουργικών αλλά και προφανώς στην επέκταση των δυνατοτήτων της πλατφόρμας. Οι βελτιώσεις που θα μπορούν να γίνουν για να αποτελέσει ένα ολοκληρωμένο σύστημα και να μπορεί να σταθεί σαν προϊόν είναι οι ακόλουθες

Αρχικά καλό θα ήταν να υπάρχει ιστορικό απεσταλμένων μηνυμάτων. Όπως επίσης και να φτιαχτεί μηχανισμός που να διατηρεί στατιστικά. Μια ολοκληρωμένη υπηρεσία θα πρέπει να καταμετρά τα push notifications που προβλήθηκαν (όσων έχουν φωτογραφία).

Παράλληλα με τα στατιστικά προβολής θα πρέπει να μπορεί να καταγράφει τα urls που ανοίγουν από το προωθημένο μήνυμα. Προέκταση του, αποτελεί και η εσωτερική καταγραφή των αποκρίσεων του χρήστη σε περίπτωση που υπάρχει ανάδραση με κουμπιά εντός του μηνύματος.

Μια άλλη σημαντική επέκταση του συστήματος θα είναι η εκτενής χρήση της redis. Η ανάκτηση των δεδομένων στη redis να γίνεται εκτός του api και του διαχειριστικού περιβάλλοντος από εξωτερικό κώδικα που θα αναλαμβάνει να προσπελαύνει τη βάση δεδομένων ανά τακτά χρονικά διαστήματα και να ενημερώνει τη redis. Με αυτό τον τρόπο θα αποκοπεί από τη βάση δεδομένων το API και σε μεγάλο βαθμό το διαχειριστικό περιβάλλον της πλατφόρμας. Σε αυτή την περίπτωση το σύστημα θα μπορεί να κατανεμηθεί σε πολλαπλούς εξυπηρετητές και δυναμική θα διαμορφώνεται ανάλογα με τις ανάγκες της υπηρεσίας. Η redis μπορεί να τρέχει και να συγχρονίζεται σε πολλαπλούς εξυπηρετητές καθώς τρέχει μόνιμα στη ram και ο συνολικός όγκος όλων δεδομένων που περιέχει δεν ξεπερνούν τα μερικά MB για χιλιάδες δυνητικούς χρήστες της πλατφόρμας. Ο όγκος όλων των ανακτημένων πληροφοριών από τη βάση δεδομένων ανέρχεται σε μερικά byte ή έστω μερικά kilobyte ανά χρήστη.

Τέλος θα μπορούσε να συνδυαστεί με άλλον service worker παρέχοντας για πρώτη φορά έναν service worker που να παρέχει πολλαπλές διαφορετικές υπηρεσίες, π.χ. λήψη προωθημένων μηνυμάτων και βελτιστοποίηση εικόνων σε ένα ενιαίο σύστημα.

# 9 Αναφορές


[1]. Matt Gaunt, “Service Workers: an Introduction”, https://developers.google.com/web/fundamentals/primers/service-workers, 2018

[2]. Matt Gaunt, Joseph Medley, “Web Push Interoperability Wins”, https://developers.google.com/web/updates/2016/07/web-push-interop-wins.

[3]. A. Kumar and S. Johari, "Push notification as a business enhancement technique for e-commerce," 2015 Third International Conference on Image Information Processing (ICIIP), 2015, pp. 450-454, doi: 10.1109/ICIIP.2015.7414815.

[4]. N. Li, Y. Du and G. Chen, "Survey of Cloud Messaging Push Notification Service," 2013 International Conference on Information Science and Cloud Computing Companion, 2013, pp. 273-279, doi: 10.1109/ISCC-C.2013.132.

[5]. Push API, https://www.w3.org/TR/2012/WD-push-api-20121018/, 2012

[6]. Push technology, https://en.wikipedia.org/wiki/Push_technology

[7]. Service Worker compatibly, https://caniuse.com/?search=service%20worker

[8]. Service Workers, https://caniuse.com/?search=service%20workers

[9]. Van Ouwerkerk, M., Thomson, M., Sullivan, B., and E. Fullea, "W3C Push API", ED push-api, https://w3c.github.io/push-api/, 2016.

[10]. Email, https://en.wikipedia.org/wiki/Email

[11]. Introduction to Push Notifications, Google Developers, https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications#identifying_your_service_with_vapid_auth

[12]. I. Warren, A. Meads, S. Srirama, T. Weerasinghe and C. Paniagua, "Push Notification Mechanisms for Pervasive Smartphone Applications," in IEEE Pervasive Computing, vol. 13, no. 2, pp. 61-71, Apr.-June. 2014, doi: 10.1109/MPRV.2014.34.

# Παραρτήματα

# 10 Παράτημα Α

Πηγαίος κώδικας και ρυθμίσεις API πλατφόρμας. Πρόκειται για τη λειτουργική μονάδα διαχείρισης εγγεγραμμένων χρηστών και προώθησης μηνυμάτων προς αποστολή.

**Εγκατάσταση και ρύθμιση των πακέτων βιβλιοθηκών composer.json**

```
{
        "name": "pushhaste/api",
  "description": "api of pushhaste",
  "type": "project",
  "version": "0.1.0",
  "require": {
          "php": "^7 || ^8",
                "ext-curl": "*",
                "ext-pdo": "*",
                "ext-json": "*",
                "ext-openssl": "*",
                "ext-mbstring": "*",

                "monolog/monolog": "^2",
                "predis/predis": "^v1",
                "php-di/php-di": "^6",
                "slim/psr7": "^1",
                "slim/slim": "^4.7",
                "minishlink/web-push": "v6.0.5",
                "vlucas/phpdotenv": "^v5.3"
        },
  "require-dev": {
    "phpunit/phpunit": "^8.4"
  },
        "autoload": {
                "psr-4": {
                        "PUSHhaste_API\\": "src/"
                }
        },
        "autoload-dev": {
                "psr-4": {
                        "App\\Test\\": "tests/"
                }
        },
```

```
    "config": {
        "process-timeout": 0,
        "sort-packages": true
    }
}
```

**Κώδικας διαμόρφωσης docker container του API (docker-compser.yml)**

```
version: "3.8"
services:

  redis:
    image: redis:alpine

  postgres:
    image: postgres:9.6-alpine
    working_dir: /application
    volumes:
      - ./docker-entrypoint-initdb.d:/docker-entrypoint-initdb.d
    ports:
      - "8766:5432"
    environment:
      - POSTGRES_HOST=${POSTGRES_HOST}
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
      - POSTGRES_DB=${POSTGRES_DB}
      - POSTGRES_PORT=${POSTGRES_PORT}



  php-fpm:
    build: docker/php-fpm
    working_dir: /application
    depends_on:
      - postgres
      - redis
    volumes:
      - .:/application
      - ./docker/php-fpm/php-ini-overrides.ini:/etc/php/8.0/fpm/conf.d/99-overrides.ini
    environment:
      - POSTGRES_DRIVER=${POSTGRES_DRIVER}
      - POSTGRES_HOST=${POSTGRES_HOST}
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
```

```
    - POSTGRES_DB=${POSTGRES_DB}
    - POSTGRES_PORT=${POSTGRES_PORT}
    - REDIS_HOST=redis
    - REDIS_SCHEME=tcp
    - REDIS_PASSWORD=
    - REDIS_PORT=6379
    -       VAPID_PUPLIC_KEY=BFYFgbs93AA5hFQG2U7W4vfm89mFkCw-WvMC5fD6m4FTQu48cADrpP5VcaI9y2CiHcsIC-
kdXfHoCVn68KeZFXI
    - VAPID_SUBJECT=https://www.imghaste.com
    - VAPID_PRIVATE_KEY=kEKkjuXebm2X_0w7KzxSc8LWQ47JLxuCZugqvmRJbeI

 webserver:
  image: nginx:alpine
  working_dir: /application
  depends_on:
   - php-fpm
  volumes:
   - .:/application
   - ./docker/nginx/nginx.conf:/etc/nginx/conf.d/default.conf
  ports:
   - "8460:80"
```

**Ειδικές ρυθμίσεις nginx (nginx.conf)**

```
server {
  listen 80 default;
            server_name local.api.pushhaste.com;

  client_max_body_size 8M;

  access_log /var/log/nginx/application.access.log;



  root /application/public;
  index index.php;

  if (!-e $request_filename) {
    rewrite ^.*$ /index.php last;
  }

  location ~ \.php$ {
    fastcgi_pass php-fpm:9000;
    fastcgi_index index.php;
```

```
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PHP_VALUE "error_log=/var/log/nginx/application_php_errors.log";
        fastcgi_buffers 16 16k;
        fastcgi_buffer_size 32k;
        include fastcgi_params;
    }
}
```

### Ειδικές ρυθμίσεις nginx (Dockerfile)

```
FROM phpdockerio/php80-fpm:latest
WORKDIR "/application"

# Fix debconf warnings upon build
ARG DEBIAN_FRONTEND=noninteractive

# Install selected extensions and other stuff
RUN apt-get update \
    && apt-get -y --no-install-recommends install  php8.0-pgsql php-redis \
    && apt-get clean; rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* /usr/share/doc/*
```

### Ειδικές ρυθμίσεις php (php-ini-overrides.ini)

```
upload_max_filesize = 100M
post_max_size = 8M
```

### Ειδικές ρυθμίσεις php (Dockerfile)

```
FROM phpdockerio/php80-fpm:latest
WORKDIR "/application"

# Fix debconf warnings upon build
ARG DEBIAN_FRONTEND=noninteractive

# Install selected extensions and other stuff
RUN apt-get update \
    && apt-get -y --no-install-recommends install  php8.0-pgsql  php8.0-gmp php-redis \
    && apt-get clean; rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* /usr/share/doc/*

# Install GMP
RUN apt-get update; \
    && apt-get -y --no-install-recommends install \
      libgmp-dev \
    && apt-get clean; \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* /usr/share/doc/*
```

```
RUN docker-php-ext-configure gmp
RUN docker-php-ext-install gmp

# Install git
RUN apt-get update \
   && apt-get -y install git nodejs npm \
            && npm install forever -g \
   && apt-get clean; rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* /usr/share/doc/*
```

**Αρχικοποίηση βάσης δεδομένων**

```
CREATE extension IF NOT EXISTS "uuid-ossp";
```

```
DROP TABLE IF EXISTS "public"."users";
CREATE TABLE "public"."users" (
  "user_id" uuid NOT NULL DEFAULT uuid_generate_v4(),
  "email" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
  "password" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
  "user_secret" uuid NOT NULL,
  "status" int4 NOT NULL DEFAULT 0,
  "settings" jsonb DEFAULT '{}'::jsonb,
  "presets" jsonb DEFAULT '[]'::jsonb,
  "created_on" timestamp(0) DEFAULT now(),
  "reset_token" varchar(255) COLLATE "pg_catalog"."default",
  "deleted_on" timestamp(0),
  "referrer_id" uuid
)
;



-- ----------------------------
-- Indexes structure for table users
-- ----------------------------
CREATE INDEX "email_idx" ON "public"."users" USING btree (
  "email" COLLATE "pg_catalog"."default" "pg_catalog"."text_ops" ASC NULLS LAST
);
CREATE INDEX "user_id_idx" ON "public"."users" USING btree (
  "user_id" "pg_catalog"."uuid_ops" ASC NULLS LAST
);


-- ----------------------------
-- Uniques structure for table users
```

```
-- ----------------------------
ALTER TABLE "public"."users" ADD CONSTRAINT "email_uqidx" UNIQUE ("email");


-- ----------------------------
-- Primary Key structure for table users
-- ----------------------------
ALTER TABLE "public"."users" ADD CONSTRAINT "users_pkey" PRIMARY KEY ("user_id");



-- ----------------------------
-- Table structure for websites
-- ----------------------------
DROP TABLE IF EXISTS "public"."websites";
CREATE TABLE "public"."websites" (
  "website_id" uuid NOT NULL DEFAULT uuid_generate_v4(),
  "user_id" uuid NOT NULL,
  "website" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
  "website_url" varchar(255) COLLATE "pg_catalog"."default" NOT NULL,
  "settings" jsonb DEFAULT '{}'::jsonb,
  "stats" jsonb DEFAULT '{}'::jsonb,
  "created_on" timestamp(0) NOT NULL DEFAULT now(),
  "updated_on" timestamp(0) NOT NULL DEFAULT now(),
  "deleted_on" timestamp(0)
);

-- ----------------------------
-- Indexes structure for table websites
-- ----------------------------
CREATE INDEX "website_user_id_idx" ON "public"."websites" USING btree (
  "user_id" "pg_catalog"."uuid_ops" ASC NULLS LAST
);

-- ----------------------------
-- Uniques structure for table websites
-- ----------------------------
ALTER TABLE "public"."websites" ADD CONSTRAINT "websites_website_uq_idx" UNIQUE ("website");

-- ----------------------------
-- Primary Key structure for table websites
-- ----------------------------
ALTER TABLE "public"."websites" ADD CONSTRAINT "websites_pkey" PRIMARY KEY ("website_id");
```

```
-- ----------------------------
-- Table structure for subscribers
-- ----------------------------
DROP TABLE IF EXISTS "public"."subscribers";
CREATE TABLE "public"."subscribers" (
  "subscriber_id" uuid NOT NULL DEFAULT uuid_generate_v4(),
  "endpoint" TEXT not null,
  "authorisation_key" TEXT not null,
  "authToken" TEXT not null,
  "website_id" uuid NOT NULL,
  "created_on" timestamp NOT NULL DEFAULT NOW()
);

-- ----------------------------
-- Indexes structure for table subscribers
-- ----------------------------
CREATE INDEX "subscribers_website_id_idx" ON "public"."subscribers" USING btree (
  "website_id" "pg_catalog"."uuid_ops" ASC NULLS LAST
);

-- ----------------------------
-- Uniques structure for table subscribers
-- ----------------------------
ALTER TABLE "public"."subscribers" ADD CONSTRAINT "subscribers_authorisation_key_authToken_key" UNIQUE ("authorisation_key", "authToken", "website_id");

-- ----------------------------
-- Primary Key structure for table subscribers
-- ----------------------------
ALTER TABLE "public"."subscribers" ADD CONSTRAINT "subscribers_pkey" PRIMARY KEY ("subscriber_id");

-- ----------------------------
-- Table structure for campaigns
-- ----------------------------
DROP TABLE IF EXISTS "public"."campaigns";
CREATE TABLE "public"."campaigns" (
  "campaign_id" uuid NOT NULL DEFAULT uuid_generate_v4(),
  "user_id" uuid NOT NULL,
  "data" jsonb NOT NULL,
  "status" int4 NOT NULL DEFAULT 0,
  "is_deleted" bool NOT NULL DEFAULT false,
```

```
  "is_paused" bool NOT NULL DEFAULT false,
  "created_on" TIMESTAMP NOT NULL DEFAULT NOW(),
  "message" jsonb NOT NULL,
  "website_id" uuid NOT NULL
);


-- ----------------------------
-- Indexes structure for table campaigns
-- ----------------------------
CREATE INDEX "campaigns_user_id_idx" ON "public"."campaigns" USING btree (
  "user_id" "pg_catalog"."uuid_ops" ASC NULLS LAST
);


-- ----------------------------
-- Primary Key structure for table campaigns
-- ----------------------------
ALTER TABLE "public"."campaigns" ADD CONSTRAINT "campaigns_pkey" PRIMARY KEY ("campaign_id");
```

**Κώδικας αποστολής προωθημένου μηνύματος – cli script «δαίμονας»**

```php
<?php

use \Minishlink\WebPush\WebPush as webpush;
use \Minishlink\WebPush\Subscription as subscription;

error_reporting(E_ERROR | E_PARSE);
sleep(1);
require_once(__DIR__ . '/../../vendor/autoload.php');


if (file_exists(__DIR__ . '/../../.env')) {
    //$dotenv = Dotenv\Dotenv::createMutable(__DIR__ . '/../../');
    $dotenv = Dotenv\Dotenv::createUnsafeMutable(__DIR__ . '/../../');
    $dotenv->load();
}
$settings = require_once(__DIR__ . '/../../config/settings.php');


$redis = new Predis\Client("{$settings['redis']['dsn']}");

$auth['VAPID'] = ['subject'=>$settings['VAPID']['subject'],
'publicKey'=>$settings['VAPID']['publicKey'],
'privateKey'=>$settings['VAPID']['privateKey']];
$webPush = new \Minishlink\WebPush\WebPush($auth);


$key = "NotificationsQueue";


//ανάκτηση αποδέκτη από στοίβα redis
$subscription_data = $redis->rpop($key);

if (!$subscription_data) {
    echo "sleeping";
    sleep(3);
}
```

```php
else {
    $subscription_data = @unserialize($subscription_data);

    $website_id = $subscription_data['website_id'];
    unset ($subscription_data['website_id']);

    //ανάκτηση μηνύματος από redis
    $campaign_key = "setCampaignFromWebSiteID:{$website_id}";
    $campaign_data = $redis->get($campaign_key);
    $campaign_data = json_decode($campaign_data);

    $subscription = Subscription::create($subscription_data);

    $payload = $campaign_data->message;

    $report = $webPush->sendOneNotification(
        $subscription,
        $payload
    );

    $endpoint = $report->getRequest()->getUri()->__toString();

    $reply =  ["endpoint" => $endpoint, "subscriber"=>$report->getReason()];

    if ($report->isSuccess()) {
        $reply['msg'] = "[v] Message sent successfully for subscription.";
    } else {
        $reply['msg'] = "x] Message failed to sent for subscription.";
    }

    echo json_encode($reply, JSON_UNESCAPED_UNICODE);
}
```

# 11 Παράτημα Β

Πηγαίος κώδικας και service worker και υποστηριζόμενων συναρτήσεων.

```
'use strict';

var URLtoGO = '';
var Push_Data;

self.addEventListener('push', function(event) {
    console.log('[service worker] push received');
    console.log('[service worker] Push had this data: '+  event.data.text());
    //console.log('I got this one:')
    //var push_data = JSON.parse(event.data.text());
    Push_Data = event.data.json();
    // console.log(push_data);
    // console.log(event.data.json());

    URLtoGO = Push_Data.URL;
    const title = Push_Data.title;
    const options = {
        dir: Push_Data.dir,
        body:Push_Data.body,
        icon: Push_Data.icon,
        image:Push_Data.image,
        badge:Push_Data.badge,
        tag:Push_Data.tag,
        vibrate: Push_Data.vibrate,
        timestamp: null,
        renotify: Push_Data.renotify,
        actions: Push_Data.actions,
        silent: Push_Data.silent,
        requireInteraction: Push_Data.requireInteraction,
        sticky: Push_Data.sticky,
        notificationCloseEvent: Push_Data.notificationCloseEvent,
        data: Push_Data.data
    }
    event.waitUntil(self.registration.showNotification(title, options));
});


self.onnotificationclick = function(event) {
    let URLtoVisit = URLtoGO;


    //έλεγχος αν πατηθεί κουμπί ανάδρασης
    Push_Data.actions.forEach(function (action, index) {
        if (action.action == event.action)
            URLtoVisit = action.url;
    });

    const urlToOpen = new URL(URLtoVisit, self.location.origin).href;

    const promiseChain = clients.matchAll({
        type: 'window',
        includeUncontrolled: true
    })
        .then((windowClients) => {
            let matchingClient = null;

            //έλεγχος εάν ο σύνδεσμος που προκειται να επισκεφτούμε είνια ήδη ανοικτός
```

```
σε κάποια
            //καρτέλα του περιηγητή
            for (let i = 0; i < windowClients.length; i++) {
                const windowClient = windowClients[i];
                if (windowClient.url === urlToOpen) {
                    matchingClient = windowClient;
                    break;
                }
            }

            //αν υπάρχει ήδη ο ιστότοπος ανοιχτός τότε έρχεται στο προσκήνιο
            if (matchingClient) {
                return matchingClient.focus();
            } else {
                //αν δεν υπάρχει ανοιχτός ο ιστότοπος ανοίγει σε νέα καρτέλα
                return clients.openWindow(urlToOpen);
            }
        });

    event.waitUntil(promiseChain);
};
```

Πηγαίος κώδικας service worker (serviceworker.js)

```
'use strict';

var UNSUBSCRIBE =0;
var SUBSCRIBE = 1;
var UPDATE_SUBSCRIPTION = 2
var API_ENDPOINT = '{{API_ENDPOINT}}';
var SERVICE_WORKER_URL = '{{SERVICE_WORKER_URL}}'
var WEBSITE_OWNER = '{{WEBSITE_ID}}'; //window.location.protocol + "//" +
window.location.host;

document.addEventListener('DOMContentLoaded', () => {
   const applicationServerPubliKey = '{{APPLICATION_PUBLIC_KEY}}';
   let isPushEnabled = false;
   let isSubscribed = false;

  const pushButton = document.querySelector('#push-subscription-button');
  // if (!pushButton) {
  //   return;
  // }

  // pushButton.addEventListener('click', function() {
  //   if (isPushEnabled) {
  //     subscribeUser(UNSUBSCRIBE);
  //   } else {
  //     subscribeUser();
  //   }
  // });

  if (!('serviceWorker' in navigator)) {
    console.warn('Service workers are not supported by this browser');
    return;
  }

  if (!('PushManager' in window)) {
    console.warn('Push notifications are not supported by this browser');
    return;
```

```
  }

  if (!('showNotification' in ServiceWorkerRegistration.prototype)) {
    console.warn('Notifications are not supported by this browser');
    return;
  }

  // Check the current Notification permission.
  // If its denied, the button should appears as such, until the user changes the
permission manually
  if (Notification.permission === 'denied') {
    console.warn('Notifications are denied by the user');
    return;
  }

  navigator.serviceWorker.register(SERVICE_WORKER_URL)
      .then(function(SWRegistration) {
          console.log('[SW] pushHASTE Service worker has been registered'); //,
SWRegistration);
          subscribeUser(UPDATE_SUBSCRIPTION);
      })
      .catch(function(error) {
          console.error('[SW] Service worker registration failed', error);
      }
  );



  function subscribeUser(action = SUBSCRIBE) {
      return checkNotificationPermission()
        .then(() => navigator.serviceWorker.ready)
        .then(serviceWorkerRegistration =>
            serviceWorkerRegistration.pushManager.subscribe({userVisibleOnly: true,
applicationServerKey: urlBase64ToUint8Array(applicationServerPubliKey),
            })
        )
        .then(subscription => {
          return subscribe_through_api(subscription, action);
        })
        .catch(e => {
          if (Notification.permission === 'denied') {
            console.warn('Notifications are denied by the user.');
          } else {
            console.error('Impossible to subscribe to push notifications', e);
          }
        });
  }


  function subscribe_through_api(subscription, action = SUBSCRIBE) {
    if (subscription) {
      const p256dh = subscription.getKey('p256dh');
      const authorisation_key = subscription.getKey('auth');
      const key = subscription.getKey('p256dh');
      const token = subscription.getKey('auth');


      switch (action) {
        case SUBSCRIBE:
        case UPDATE_SUBSCRIPTION:
        default:
          var _api_url = "/device/subscribe";
          var _method = 'post';
          isPushEnabled = true;
          //document.getElementById('push-subscription-button').innerText  = 'Διαφραφή
από την υπηρεσία';
```

```
          break;
        case UNSUBSCRIBE:
          var _api_url = "/device/unsubscribe";
          var _method = 'delete';
          isPushEnabled = false;
          //document.getElementById('push-subscription-button').innerText  = 'Εγγραφή
στην υπηρεσία';
      }
      fetch(API_ENDPOINT + _api_url, {
        method: _method,
        headers: new Headers({'Content-Type': 'application/json'}),
        body: JSON.stringify({
          endpoint: subscription.endpoint,
          //p256dh: p256dh ? btoa(String.fromCharCode.apply(null, new
Uint8Array(subscription.getKey(p256dh)))) : null,
          //authorisation_key: authorisation_key ? btoa(String.fromCharCode.apply(null,
new Uint8Array(subscription.getKey(authorisation_key)))) : null
          authorisation_key: key ? btoa(String.fromCharCode.apply(null, new
Uint8Array(key))) : null,
          authToken: token ? btoa(String.fromCharCode.apply(null, new
Uint8Array(token))) : null,
          website: WEBSITE_OWNER
        })
      }).then(function(response) {
        return response.text();
      }).catch(function(error) {
        console.log(error);
      })
    }
  }


  function urlBase64ToUint8Array(base64String) {
    const padding = '='.repeat((4 - (base64String.length % 4)) % 4);
    const base64 = (base64String + padding).replace(/\-/g, '+').replace(/_/g, '/');

    const rawData = window.atob(base64);
    const outputArray = new Uint8Array(rawData.length);

    for (let i = 0; i < rawData.length; ++i) {
      outputArray[i] = rawData.charCodeAt(i);
    }
    return outputArray;
  }

  function checkNotificationPermission() {
    return new Promise((resolve, reject) => {
      if (Notification.permission === 'denied') {
        return reject(new Error('Push messages are blocked.'));
      }

      if (Notification.permission === 'granted') {
        return resolve();
      }

      if (Notification.permission === 'default') {
        return Notification.requestPermission().then(result => {
          if (result !== 'granted') {
            reject(new Error('Bad permission result'));
          } else {
            resolve();
          }
        });
      }

      return reject(new Error('Unknown permission'));
```

```
    });
  }
  //
  //
  // const sendPushButton = document.querySelector('#send-push-button');
  // if (!sendPushButton) {
  //   return;
  // }
  //
  //
  // sendPushButton.addEventListener('click', () =>
  //   navigator.serviceWorker.ready
  //     .then(serviceWorkerRegistration =>
serviceWorkerRegistration.pushManager.getSubscription())
  //     .then(subscription => {
  //       if (!subscription) {
  //         alert('Please enable push notifications');
  //         return;
  //       }
  //
  //       const contentEncoding = (PushManager.supportedContentEncodings ||
['aesgcm'])[0];
  //       const jsonSubscription = subscription.toJSON();
  //       fetch(API_ENDPOINT + '/notification/push', {
  //         method: 'POST',
  //         body: JSON.stringify(Object.assign(jsonSubscription, { contentEncoding })),
  //       });
  //     })
  // );

  //subscribeUser();
});
```

Πηγαίος κώδικας συνοδευτικών συναρτήσεων (sdk.js)

# 12 Παράτημα Γ

Πηγαίος κώδικας και ρυθμίσεις πλατφόρμα αποστολής μηνυμάτων. Πρόκειται για τη λειτουργική μονάδα σύνδεσης νέων προωθημένων μηνυμάτων προς αποστολή από τους διαχειριστές ιστοτόπων.

**Εγκατάσταση και ρύθμιση των πακέτων βιβλιοθηκών composer.json**

```
{
        "name": "pushhaste/app",
  "description": "app of pushhaste",
  "type": "project",
  "version": "0.1.0",
  "require": {
          "php": "^7 || ^8",
                "ext-curl": "*",
                "ext-pdo": "*",
                "ext-json": "*",
                "ext-openssl": "*",
                "ext-mbstring": "*",

    "guzzlehttp/guzzle": "^6.5",
                "monolog/monolog": "^2",
                "predis/predis": "^v1",
                "php-di/php-di": "^6",
                "slim/psr7": "^1",
                "slim/slim": "^4.7",
                "minishlink/web-push": "v6.0.5",
                "vlucas/phpdotenv": "^v5.3",
    "slim/twig-view": "3.0.0-beta"
        },
  "require-dev": {
    "phpunit/phpunit": "^8.4"
  },
        "autoload": {
                "psr-4": {
                        "PUSHhaste_APP\\": "src/"
                }
        },
        "autoload-dev": {
```

```
                "psr-4": {
                        "App\\Test\\": "tests/"
                }
        },
    "config": {
        "process-timeout": 0,
        "sort-packages": true
    }
}
```

**Κώδικας διαμόρφωσης docker container της πλατφόρμας αποστολής μηνυμάτων (docker-composer.yml)**

```
version: "3.8"
services:

  redis:
    image: redis:alpine

  php-fpm:
    build: docker/php-fpm
    working_dir: /application
    depends_on:
      - redis
    volumes:
      - .:/application
      - ./docker/php-fpm/php-ini-overrides.ini:/etc/php/8.0/fpm/conf.d/99-overrides.ini
    environment:
      - REDIS_HOST=redis
      - REDIS_SCHEME=tcp
      - REDIS_PASSWORD=
      - REDIS_PORT=6379

  webserver:
    image: nginx:alpine
    working_dir: /application
    depends_on:
      - php-fpm
    volumes:
      - .:/application
      - ./docker/nginx/nginx.conf:/etc/nginx/conf.d/default.conf
    ports:
      - "8440:80"
```

```
  extra_hosts:
   - "local.api.pushhaste.com:192.168.1.6"
```

**Ειδικές ρυθμίσεις nginx πλατφόρμας αποστολής μηνυμάτων (nginx.conf)**

```
server {
  listen 80 default;
        server_name local.app.pushhaste.com;

  client_max_body_size 8M;

  access_log /var/log/nginx/application.access.log;



  root /application/public;
  index index.php;

  if (!-e $request_filename) {
    rewrite ^.*$ /index.php last;
  }

  location ~ \.php$ {
    fastcgi_pass php-fpm:9000;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param PHP_VALUE "error_log=/var/log/nginx/application_php_errors.log";
    fastcgi_buffers 16 16k;
    fastcgi_buffer_size 32k;
    include fastcgi_params;
  }
}
```

**Ειδικές ρυθμίσεις nginx πλατφόρμας αποστολής μηνυμάτων (Dockerfile)**

```
FROM phpdockerio/php80-fpm:latest
WORKDIR "/application"

# Fix debconf warnings upon build
ARG DEBIAN_FRONTEND=noninteractive

# Install selected extensions and other stuff
RUN apt-get update \
  && apt-get -y --no-install-recommends install php-redis \
  && apt-get clean; rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* /usr/share/doc/*
```

**Ειδικές ρυθμίσεις php για την πλατφόρμα αποστολής μηνυμάτων (php-ini-overrides.ini)**

```
upload_max_filesize = 100M
post_max_size = 8M
```

**Ειδικές ρυθμίσεις php για την πλατφόρμα αποστολής μηνυμάτων (Dockerfile)**

```
FROM phpdockerio/php80-fpm:latest
WORKDIR "/application"

# Fix debconf warnings upon build
ARG DEBIAN_FRONTEND=noninteractive

# Install selected extensions and other stuff
RUN apt-get update \
  && apt-get -y --no-install-recommends install php-redis iputils-ping vim \
  && apt-get clean; rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* /usr/share/doc/*



# Install git
RUN apt-get update \
  && apt-get -y install git nodejs npm \
        && npm install forever -g \
  && apt-get clean; rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* /usr/share/doc/*
```