

**Ανάπτυξη Εφαρμογής Ιστού για Ενοποιημένη
Πρόσβαση σε Πολλαπλές Ψηφιακές
Υπηρεσίες με Χρήση API**

Παπαδιάς Κωνσταντίνος

A.M. 2614

Διπλωματική Εργασία

Επιβλέπων: Αναστασιάδης Στέργιος

Ιωάννινα, Οκτώβριος, 2022



**ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA**

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα μου, τον κύριο Στέργιο Αναστασιάδη, για την ελευθερία που μου έδωσε να δημιουργήσω αυτό που πραγματικά ήθελα, καθώς και την εμπιστοσύνη που μου έδειξε, για την υλοποίηση της εφαρμογής. Επίσης, θα ήθελα να τον ευχαριστήσω για την καθοδήγηση που μου έδωσε ώστε να καταλάβω πως πρέπει να γίνεται η σχεδίαση μιας εφαρμογής, και η συγγραφή ενός σωστού κειμένου διπλωματικής θέσης.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου, για την υποστήριξη που μου έδωσε όλα αυτά τα χρόνια.

Οκτώβριος 2022

Παπαδιάς Κωνσταντίνος

Περίληψη

Σκοπός της παρούσας διπλωματικής είναι η ανάπτυξη μιας διαδικτυακής εφαρμογής, η οποία θα προσφέρει στο χρήστη ενοποιημένη πρόσβαση, μέσω της γραφικής διεπαφής της, σε 3 διαφορετικούς λογαριασμούς που χρησιμοποιεί για να επικοινωνεί καθημερινά. Συγκεκριμένα, ο χρήστης θα μπορεί να συνδέσει στην εφαρμογή 2 διαφορετικούς Gmail λογαριασμούς, καθώς και 1 Microsoft Teams λογαριασμό, έτσι ώστε να μπορεί, στη συνέχεια, να δημιουργεί καινούριες συνομιλίες, να διαβάζει τα μηνύματα από τους 3 διαφορετικούς λογαριασμούς του, να οργανώνει τα εισερχόμενα μηνύματά του σε νήματα και να τα επισημαίνει ως σημαντικά, ή αδιάβαστα, στην περίπτωση των Gmail λογαριασμών, καθώς και να κάνει αποστολή νέων μηνυμάτων, κάθε ένα από τα οποία θα προέρχεται από τον κατάλληλο λογαριασμό του.

Λέξεις Κλειδιά: Διαδικτυακή Εφαρμογή, Ενοποιημένη Πρόσβαση

Abstract

The purpose of this thesis is the development of a web application, which will offer the user unified access, through its graphical interface, to 3 different accounts that he uses to communicate daily. Specifically, the user will be able to connect to the application 2 different Gmail accounts, as well as 1 Microsoft Teams account, so that he can then create new conversations, read messages from his 3 different accounts, organize his incoming messages into threads and mark them as important, or unread, in the case of the Gmail accounts, as well as send new messages, each one coming from the appropriate account.

Keywords: Web Application, Unified Access

Πίνακας Περιεχομένων

Κεφάλαιο 1. Εισαγωγή.....	1
1.1 Κίνητρο.....	1
1.2 Δυνατότητες της εφαρμογής που θα υλοποιήσουμε.....	1
1.3 Δομή της Εργασίας.....	2
Κεφάλαιο 2. Υπόβαθρο	3
2.1 Η Τριεπίπεδη αρχιτεκτονική.....	3
2.2 Το μοντέλο Πελάτη-Διακομιστή	4
2.3 Εφαρμογή μονής σελίδας	5
2.4 Διεπαφή Προγραμματισμού Εφαρμογών	5
2.4.1 Η αρχιτεκτονική <i>REST</i>	5
2.5 Η αρχιτεκτονική <i>MVC</i>	6
2.6 Βιβλιοθήκες και πλαίσια λογισμικού.....	6
2.7 Μοντέλο Αντικειμένου Εγγράφου	7
2.8 Τεχνολογίες Επιπέδου Παρουσίασης	7
2.8.1 Η βιβλιοθήκη <i>React</i>	7
2.9 Τεχνολογίες Επιπέδου Εφαρμογής	8
2.9.1 Το Περιβάλλον Χρόνου Εκτέλεσης <i>Node.js</i>	8
2.9.2 Το πλαίσιο <i>Express.js</i>	9
2.10 Τεχνολογίες Επιπέδου Αποθήκευσης	9
2.10.1 Το σύστημα διαχείρισης βάσεων δεδομένων <i>MongoDB</i>	10
2.11 Στοιίβα <i>MERN</i>	10
2.11.1 Μεταφορά δεδομένων μεταξύ των τριών επιπέδων.....	11
2.12 Gmail API	11
2.13 Microsoft Graph API.....	12

2.14 Κουπόνια πρόσβασης και ανανέωσης.....	12
2.15 Το πρωτόκολλο εξουσιοδότησης OAuth 2.0.....	12
2.16 HTTP Cookie.....	13
Κεφάλαιο 3. Σχεδίαση	14
3.1 Απαιτήσεις της εφαρμογής.....	14
3.2 Αυθεντικοποίηση χρήστη.....	15
3.3 Εξουσιοδότηση εφαρμογής για πρόσβαση στους λογαριασμούς του χρήστη	15
3.4 Ιστορίες χρήστη της εφαρμογής	16
3.5 Αποθήκευση δεδομένων	18
3.6 Ασφάλεια.....	19
3.7 Αρχιτεκτονική της εφαρμογής.....	22
Κεφάλαιο 4. Υλοποίηση	23
4.1 Δομή του κώδικα.....	23
4.1.1 Ο φάκελος <i>server</i>	23
4.1.2 Ο φάκελος <i>client</i>	24
4.2 Τα συστατικά της εφαρμογής	25
4.2.1 Συστατικά του διακομιστή.....	26
4.2.2 Συστατικά του πελάτη	34
4.3 Οι βιβλιοθήκες που θα χρησιμοποιήσουμε	35
4.3.1 Βιβλιοθήκες που θα χρησιμοποιήσουμε στο <i>backend</i>	35
4.3.2 Βιβλιοθήκες που θα χρησιμοποιήσουμε στο <i>backend</i>	37
4.4 Αποστολή αιτήματος στα διαδικτυακά APIs των παρόχων.....	37
4.4.1 Αυτόματη ανανέωση των <i>access tokens</i>	38
4.5 Αποθήκευση δεδομένων στη βάση	39
4.6 Ασφάλεια.....	41
4.6.1 Συνεδρίες διακομιστή	41
4.6.2 Κρυπτογράφηση και κατακερματισμός ευαίσθητων δεδομένων.....	42
4.7 Βελτίωση απόδοσης του διακομιστή	43
4.8 Σχεδιασμός API του backend διακομιστή εφαρμογής	44
4.8.1 <i>Endpoints</i> για την αυθεντικοποίηση του χρήστη	44
4.8.2 <i>Endpoints</i> για την υλοποίηση του πρωτοκόλλου OAuth 2.0	45

4.8.3 Endpoints για την διαχείριση των λογαριασμών του χρήστη	46
4.9 Επικοινωνία frontend-backend	48
4.9.1 Εγγραφή ενός καινούριου χρήστη στην εφαρμογή	48
4.9.2 Έλεγχος πρόσβασης στη Home σελίδα ως συνδεδεμένος χρήστης.....	49
4.9.3 Σύνδεση ενός εγγεγραμμένου χρήστη στην εφαρμογή.....	49
4.9.4 Σύνδεση Gmail, ή Microsoft Teams λογαριασμού στην εφαρμογή	50
4.9.5 Ανάκτηση εισερχόμενων μηνυμάτων των λογαριασμών του χρήστη.....	50
4.9.6 Ανάκτηση συνημμένου μηνύματος από Gmail λογαριασμό.....	52
4.9.7 Επισημάνση νήματος από Gmail λογαριασμό ως διαβασμένο, ή σημαντικό.....	53
4.9.8 Αποστολή μηνύματος από Gmail λογαριασμό.....	54
4.9.9 Δημιουργία καινούριου chat για το Microsoft Teams λογαριασμό.....	54
4.9.10 Αποστολή μηνύματος σε chat του Microsoft Teams λογαριασμού.....	55
4.9.11 Αποστολή συνημμένου σε chat του Microsoft Teams λογαριασμού	56
4.9.12 Αποσύνδεση ενός συνδεδεμένου χρήστη από την εφαρμογή	57
4.10 Πρόσβαση στα web APIs.....	57
4.10.1 Πρόσβαση στο Gmail API.....	57
4.10.2 Πρόσβαση στο Microsoft Graph API.....	59
Κεφάλαιο 5. Η γραφική διεπαφή της εφαρμογής.....	61
5.1 Οι βασικές σελίδες της εφαρμογής.....	61
5.2 Εγγραφή και σύνδεση χρήστη στην εφαρμογή	63
5.3 Σύνδεση Gmail, ή Microsoft Teams λογαριασμού στην εφαρμογή.....	64
5.4 Πρόσβαση στους λογαριασμούς μέσω της Home σελίδας.....	65
5.5 Οι Gmail λογαριασμοί.....	66
5.5.1 Προβολή των μηνυμάτων.....	66
5.5.2 Αποστολή καινούριου μηνύματος ή μηνύματος απάντησης σε νήμα	68
5.6 Ο Microsoft Teams λογαριασμός.....	69
5.6.1 Προβολή των chats	69
5.6.2 Δημιουργία καινούριου chat	69
5.6.3 Αποστολή μηνύματος σε chat.....	69
5.7 Αποσύνδεση χρήστη από την εφαρμογή	70
5.8 Το λογότυπο της εφαρμογής	70

Κεφάλαιο 6. Αξιολόγηση Απόδοσης	71
6.1 Πειραματικό Περιβάλλον.....	71
6.2 Τρόπος αξιολόγησης της απόδοσης και μετρική.....	71
6.3 Εργαλείο αξιολόγησης	72
6.4 Οι λειτουργίες της εφαρμογής που θα αξιολογήσουμε	72
6.5 Δημιουργία των tests	73
6.6 Αξιολογήσεις λειτουργιών της εφαρμογής	73
6.6.1 Ταυτόχρονη ανάκτηση των μηνυμάτων, των 3 λογαριασμών του χρήστη.....	73
6.6.2 Αποστολή μηνύματος, από Gmail λογαριασμό του χρήστη	74
6.6.3 Ανέβασμα αρχείου στο OneDrive του χρήστη, και δημιουργία δικαιώματος ανάγνωσης για το χρήστη για τον οποίο προορίζεται.....	75
6.6.4 Αποστολή μηνύματος από το Microsoft Teams λογαριασμό του χρήστη	76
6.7 Συμπεράσματα.....	77
Κεφάλαιο 7. Επίλογος.....	79
7.1 Σύνοψη και συμπεράσματα.....	79
7.2 Μελλοντικές επεκτάσεις	79
Παράρτημα Α. Web APIs reference	80
A.1 Τα endpoints των διαδικτυακών APIs που αξιοποιήσαμε.....	80
A.1.1 Gmail API endpoints.....	80
A.1.2 Microsoft Graph API endpoints	83
Βιβλιογραφία.....	87

Κεφάλαιο 1. Εισαγωγή

1.1 Κίνητρο

Στις μέρες μας, ο μέσος χρήστης χρησιμοποιεί πολλές διαφορετικές εφαρμογές, για επικοινωνία. Συνήθως, θα έχει δημιουργήσει διαφορετικούς λογαριασμούς σε κάθε μία από αυτές, ώστε να μπορεί να έχει πρόσβαση στις υπηρεσίες τους. Αυτό είναι κάτι που τον αναγκάζει να μεταφέρεται από τη μία εφαρμογή, ή υπηρεσία, στην άλλη, και από τον ένα λογαριασμό στον άλλο, καθώς και να θυμάται τα διαπιστευτήρια του, για κάθε μία από τις υπηρεσίες αυτές. Αυτό δεν είναι καθόλου πρακτικό και μπορεί πολύ εύκολα να του προκαλέσει μπέρδεμα.

Στην παρούσα διπλωματική θα αναπτύξουμε μια διαδικτυακή εφαρμογή η οποία θα αντιμετωπίζει, σε κάποιο βαθμό, αυτό το πρόβλημα, συνδυάζοντας υπηρεσίες μερικών πολύ γνωστών και πολυχρησιμοποιημένων εφαρμογών επικοινωνίας, σε μία εφαρμογή.

1.2 Δυνατότητες της εφαρμογής που θα υλοποιήσουμε

Στην παρούσα διπλωματικής, θα δώσουμε τη δυνατότητα στον χρήστη, μέσω της εφαρμογής μας, να έχει ενιαία πρόσβαση σε 3 διαφορετικούς λογαριασμούς του. Πιο συγκεκριμένα, ο χρήστης θα μπορεί να συνδέσει 2 Gmail λογαριασμούς, και 1 Microsoft Teams λογαριασμό, στην εφαρμογή, ώστε να χρησιμοποιεί τη γραφική διεπαφή της για να διαβάζει, και να στέλνει, email από τους Gmail λογαριασμούς του, ή μηνύματα από το λογαριασμό του στο Microsoft Teams, να επισημαίνει Gmail νήματα ως σημαντικά ή αδιάβαστα (ώστε να έχει τα μηνύματά του οργανωμένα σε νήματα, ανά κατηγορία), καθώς και να δημιουργεί καινούρια νήματα στους Gmail, ή συνομιλίες στο Microsoft Teams λογαριασμό. Τα απεσταλμένα μηνύματα και emails θα προέρχονται από τον

εκάστοτε λογαριασμό, επιτρέποντας στο χρήστη να χρησιμοποιεί μόνο την εφαρμογή μας στην καθημερινότητά του, και να μη χρειάζεται να περιηγείται μεταξύ 2 διαφορετικών εφαρμογών, και 3 διαφορετικών λογαριασμών.

1.3 Δομή της Εργασίας

Η διπλωματική εργασία αποτελείται από ακόμα 6 κεφάλαια (εκτός του παρόντος κεφαλαίου), και 1 παράρτημα.

Συγκεκριμένα το 2^ο κεφάλαιο αφορά το τεχνολογικό υπόβαθρο που χρειάζεται ο αναγνώστης, ώστε να μπορέσει να παρακολουθήσει σωστά το κείμενο. Γίνεται ανάλυση των τεχνολογιών που επιλέχθηκαν, καθώς και τα κριτήρια για την επιλογή τους. Επίσης, γίνεται αναφορά στα διάφορα διαδικτυακά πρωτόκολλα, και τις αρχιτεκτονικές τις οποίες θα πρέπει ο αναγνώστης να γνωρίζει, καθώς και στα διαδικτυακά APIs των παρόχων που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.

Το 3^ο κεφάλαιο αφορά τη σχεδίαση της εφαρμογής. Αναλύονται τα σχεδιαστικά χαρακτηριστικά που τέθηκαν, και ο σκοπός τους, καθώς και η αρχιτεκτονική της εφαρμογής.

Το 4^ο κεφάλαιο αφορά την υλοποίηση της εφαρμογής. Αναλύεται ο σχεδιασμός του API του back-end, η αποθήκευση δεδομένων στη βάση, η υλοποίηση της ασφάλειας, η υλοποίηση των υπηρεσιών που παρέχει η εφαρμογή, καθώς και η αλληλεπίδραση των συστατικών της (δηλαδή του frontend και το backend).

Στο 5^ο κεφάλαιο γίνεται επίδειξη εκτέλεσης της εφαρμογής, ώστε να γίνει πιο εύκολα κατανοητή η ορθή χρήση της και οι δυνατότητές της, καθώς και να αναδειχθεί η γραφική διεπαφή της.

Στο 6^ο κεφάλαιο γίνεται πειραματική αξιολόγηση της απόδοσης της εφαρμογής (κυρίως του backend/διακομιστή) ώστε να αποδειχθεί η ταχύτητα και ποιότητά της, αλλά και να παρατηρηθούν πιθανές βελτιώσεις για το μέλλον.

Στο 7^ο κεφάλαιο γίνεται μία αποτίμηση της συνολικής εμπειρίας του χρήστη από την εφαρμογή, αναφέρονται μερικές πιθανές μελλοντικές επεκτάσεις και ο τρόπος με τον οποίο αυτές μπορούν να πραγματοποιηθούν.

Τέλος, στο παράρτημα Α αναφέρονται, αναλυτικά, οι υπηρεσίες των διαδικτυακών APIs των παρόχων (Gmail API, Microsoft Graph API), που χρησιμοποιήθηκαν για να υλοποιηθεί η εφαρμογή, καθώς και ο τρόπος με τον οποίο μπορεί κανείς, να τις χρησιμοποιήσει.

Κεφάλαιο 2. Υπόβαθρο

2.1 Η Τριεπίπεδη αρχιτεκτονική

Η υλοποίηση της εφαρμογής μας βασίζεται στην τριεπίπεδη αρχιτεκτονική. Η τριεπίπεδη αρχιτεκτονική αποτελείται από 3 επίπεδα, το επίπεδο παρουσίασης (presentation tier), το επίπεδο λογικής (logic tier), και το επίπεδο διατήρησης (persistence tier) [1]. Το επίπεδο παρουσίασης αφορά την γραφική διεπαφή στον φυλλομετρητή του χρήστη, η οποία έχει ως σκοπό να του δώσει τη δυνατότητα διαδραστικής αλληλεπίδρασης με την εφαρμογή. Αυτό, επίσης, λέγεται και frontend κομμάτι της εφαρμογής. Το επίπεδο λογικής αφορά το λογικό κομμάτι της εφαρμογής, την επεξεργασία των δεδομένων, και την μεταφορά τους μεταξύ του επιπέδου παρουσίασης και του επιπέδου αποθήκευσης, και βρίσκεται σε έναν ή παραπάνω διακομιστές ιστού στο διαδίκτυο. Αυτό καλείται, επίσης, και backend της εφαρμογής. Τέλος, το επίπεδο αποθήκευσης αφορά την αποθήκευση και ανάκτηση των δεδομένων με χρήση κάποιας βάσης δεδομένων, και ενός διακομιστή βάσεων δεδομένων, καθώς και τη μεταφορά τους στο επίπεδο εφαρμογής, ώστε να γίνει η επεξεργασία τους, και στη συνέχεια η μεταφορά τους στο επίπεδο παρουσίασης, για να εμφανιστούν στον χρήστη.

Η αρχιτεκτονική αυτή έχει καλό διαχωρισμό μεταξύ των τριών επιπέδων της, καθώς, κάθε επίπεδο έχει τη δική του υποδομή [2]. Δηλαδή, κάθε επίπεδο είναι χτισμένο με τις δικές του τεχνολογίες και μπορεί να λειτουργήσει ανεξάρτητα από τα υπόλοιπα. Για παράδειγμα, μπορεί (και είναι πολύ σύνηθες) η βάση, ο διακομιστής του backend και ο περιηγητής (browser) να βρίσκονται σε 3 διαφορετικά φυσικά μηχανήματα. Αυτό κάνει εφικτή την ανάπτυξη κάθε επιπέδου ανεξάρτητα από τα υπόλοιπα, με αποτέλεσμα να μπορεί να γίνει πιο εύκολη αποσφαλμάτωση κατά τη διάρκεια ανάπτυξης (και το testing) της εφαρμογής, αφού το κάθε σφάλμα θα μπορεί να αντιμετωπιστεί «τοπικά» εντός του επιπέδου στο οποίο ανήκει.

Προφανώς, ο διαχωρισμός των τριών επιπέδων «καθαρίζει» τον κώδικα, κάνοντάς τον πιο ευανάγνωστο, αλλά και μειώνει τη σύζευξη (coupling) μεταξύ των διαφορετικών κομματιών κώδικα, συνεπώς, κάνοντάς τον πιο εύκολα επεκτάσιμο, στο μέλλον. Η επεκτασιμότητα αφορά και η απόδοση της εφαρμογής, καθώς, το κάθε επίπεδο μπορεί να κλιμακωθεί ξεχωριστά. Για παράδειγμα για να αυξηθεί η απόδοση της εφαρμογής μπορούν να αυξηθεί ο αριθμός των διακομιστών εφαρμογής.

Επίσης, τα δεδομένα για να μεταφερθούν μεταξύ του επιπέδου παρουσίας (ή πελάτη) και του επιπέδου δεδομένων (δηλαδή της βάσης) περνούν πάντα από το επίπεδο εφαρμογής, με αποτέλεσμα να αυξάνεται η ακεραιότητα τους, καθώς φιλτράρονται από τον διακομιστή εφαρμογής. Το γεγονός ότι ο πελάτης δεν έχει άμεση πρόσβαση στη βάση αυξάνει την ασφάλεια της εφαρμογής, μιας και ο διακομιστής εφαρμογής μπορεί να λειτουργήσει ως εσωτερικό τείχος προστασίας για να αποτρέψει διάφορα exploits (για παράδειγμα ένθεση SQL).

Τέλος, μπορεί να γίνει καλύτερος διαμοιρασμός διεργασιών και πόρων, από τον διακομιστή εφαρμογής, ο οποίος μπορεί να αποφασίσει που θα τρέξει ποια διεργασία (πχ εξωτερικές υπηρεσίες, APIs, μικροπηρεσίες...) και ποιον διακομιστή βάσεων δεδομένων (στην περίπτωση που υπάρχει πρόσβαση σε περισσότερους από ένα) θα χρησιμοποιήσει για αποθήκευση δεδομένων.

2.2 Το μοντέλο Πελάτη-Διακομιστή

Η εφαρμογή μας ακολουθεί το μοντέλο πελάτη-διακομιστή για επικοινωνία μεταξύ της γραφικής διεπαφής και του back-end. Το μοντέλο πελάτη-διακομιστή [3] είναι ένα μοντέλο που ακολουθείται συχνά σε κατανεμημένες εφαρμογές στο διαδίκτυο. Μοιράζει τις εργασίες των εφαρμογών μεταξύ οντοτήτων που παράγουν πόρους ή υπηρεσίες (οι οποίοι καλούνται διακομιστές, ή servers) και οντοτήτων που τις καταναλώνουν (οι οποίοι καλούνται πελάτες ή clients). Οι διακομιστές είναι συνεχώς ενεργοί και αναμένουν για αιτήματα. Οι πελάτες, με τη σειρά τους, στέλνουν αιτήματα στους διακομιστές, οι οποίοι απαντούν με αποκρίσεις. Οι πελάτες και οι διακομιστές μπορούν να βρίσκονται στο ίδιο μηχάνημα, είτε σε διαφορετικά μηχανήματα και η επικοινωνία τους να γίνεται μέσω δικτύων. Για παράδειγμα, οι περιηγητές ιστού των διαφορετικών χρηστών αποτελούν τους πελάτες, ενώ το μηχάνημα στο οποίο είναι εγκατεστημένος ο backend κώδικας μίας διαδικτυακής εφαρμογής αποτελεί τον διακομιστή (που μπορεί να αποτελείται και από μικρότερους διακομιστές).

2.3 Εφαρμογή μονής σελίδας

Το front-end κομμάτι της εφαρμογής μας αποτελείται από μία εφαρμογή μονής σελίδας. Εφαρμογή μονής σελίδας (single-page application ή SPA) [4] ονομάζεται μια εφαρμογή που φορτώνει ένα μοναδικό έγγραφο html, και, για να εμφανίσει διαφορετικό περιεχόμενο στο χρήστη, ανανεώνει το έγγραφο αυτό, για παράδειγμα αφού ανακτήσει δεδομένα από κάποιο διακομιστή ή API με κάποιο HTTP αίτημα. Αυτό απαλείφει την ανάγκη για φόρτωση πολλών διαφορετικών ιστοσελίδων από τον διακομιστή και δημιουργεί, έτσι, μια πιο δυναμική εμπειρία για το χρήστη, αλλά και συμβάλει θετικά στην απόδοση της εφαρμογής, αφού μειώνονται οι κλήσεις δικτύου.

2.4 Διεπαφή Προγραμματισμού Εφαρμογών

Το back-end κομμάτι της εφαρμογής μας αποτελείται από ένα διαδικτυακό διακομιστή. Ο διακομιστής έχει μία διεπαφή προγραμματισμού εφαρμογών, ώστε να μπορεί το front-end να αλληλεπιδράσει μαζί του. Διεπαφή Προγραμματισμού Εφαρμογών (Application programming interface, ή API) [5] καλείται ένα «συμβόλαιο» (contract) μεταξύ του παρόχου κάποιας πληροφορίας και του χρήστη που θέλει να την ανακτήσει. Στην πράξη, το API είναι μία διεπαφή που επιτρέπει σε ένα πελάτη να ζητήσει σε κάποιο διακομιστή να υλοποιήσει κάποια λειτουργία, στέλνοντάς του, ή και λαμβάνοντας πίσω κάποια δεδομένα. Το API δίνει τη δυνατότητα σε ένα διακομιστή να μπορεί να υλοποιεί λειτουργίες και να παραδίδει δεδομένα με ασφάλεια και έλεγχο στο ποιος έχει πρόσβαση στα δεδομένα αυτά, και τις υπηρεσίες.

2.4.1 Η αρχιτεκτονική REST

Τα διαδικτυακά APIs μπορούν να υλοποιηθούν με διάφορες αρχιτεκτονικές. Η αρχιτεκτονική που ακολουθεί το API του διαδικτυακού διακομιστή της εφαρμογής μας είναι η Μεταφορά Κατάστασης Αναπαράστασης (Representational state transfer, ή REST) [6]. Όταν γίνεται ένα HTTP αίτημα σε ένα REST (ή RESTful) API, το API επιστρέφει μια απεικόνιση της κατάστασης του πόρου τον οποίο αφορά το αίτημα, σε αυτόν που το έστειλε. Το είδος της HTTP μεθόδου φανερώνει τη λειτουργία που θα υλοποιηθεί στους πόρους του API, και οι βασικές λειτουργίες που υλοποιούνται είναι η δημιουργία, η ανάγνωση, η ενημέρωση, και η διαγραφή ενός πόρου. Η μεταφορά δεδομένων σε ένα REST API γίνεται, συνήθως, με JSON, καθώς η JSON μορφή είναι ανεξάρτητη των διαφόρων γλωσσών προγραμματισμού, αλλά και εύκολα αναγνώσιμη από ανθρώπους.

2.5 Η αρχιτεκτονική MVC

Για να οργανώσουμε τον κώδικα της εφαρμογής μας καλύτερα, χρησιμοποιήσαμε την αρχιτεκτονική Μοντέλο-Προβολή-Ελεγκτής (Model-View-Controller, ή MVC). Η αρχιτεκτονική Μοντέλο-Προβολή-Ελεγκτής [7] είναι ένα αρχιτεκτονικό μοτίβο που ακολουθείται πολύ συχνά σε εφαρμογές οποιουδήποτε τύπου. Ο λόγος είναι διότι προβάλλει καλό διαχωρισμό «προβλημάτων» (“separation of concerns”) μεταξύ της διαχείρισης των δεδομένων, της λογικής της εφαρμογής, και της προβολής των δεδομένων, σε μια εφαρμογή, κάνοντας ευκολότερο τον εντοπισμό σφαλμάτων, καθώς και τον κώδικα της εφαρμογής πιο εύκολα επεκτάσιμο, στο μέλλον. Στην αρχιτεκτονική MVC το μοντέλο αφορά την διαχείριση των δεδομένων, η προβολή την προβολή τους στο χρήστη, και ο ελεγκτής τη λογική της εφαρμογής. Στην εφαρμογή μας, θα θεωρήσουμε ως προβολές τις 2 βασικές σελίδες του frontend.

2.6 Βιβλιοθήκες και πλαίσια λογισμικού

Σε αυτό το σημείο, πρέπει να αναφέρουμε δύο σημαντικούς τύπους τεχνολογιών που θα χρειαστούμε. Τις βιβλιοθήκες, και τα πλαίσια λογισμικού. Μία βιβλιοθήκη λογισμικού είναι ένα σύνολο από συναρτήσεις, και γενικότερα, κώδικας γραμμένος από κάποιον ή κάποιους τρίτους προγραμματιστές, με σκοπό την αποδοτική επίλυση υποπροβλημάτων σε μια εφαρμογή. Επιτρέπει στον προγραμματιστή να την προσθέσει ολόκληρη, ή συγκεκριμένες δυνατότητές της στον υπάρχοντα κώδικα που έχει αναπτύξει. Αντίθετα, ένα πλαίσιο λογισμικού είναι ένα θεμέλιο, ή καλύτερα, ένας σκελετός κώδικα, που ακολουθεί συγκεκριμένη αρχιτεκτονική, και επιτρέπει στον προγραμματιστή να το χρησιμοποιήσει, συμπληρώνοντας και αναπτύσσοντας τον κώδικά του με βάση την αρχιτεκτονική αυτή, αλλά και συγκεκριμένους κανόνες που ορίζει.

Παρότι και οι δύο τύποι τεχνολογιών έχουν ως βασικό στόχο την παροχή βοήθειας στον προγραμματιστή, ώστε να μπορεί να υλοποιήσει την εφαρμογή του αποδοτικότερα και συντομότερα (μειώνοντας τόσο την πιθανότητα σφαλμάτων, όσο το μέγεθος, τον χρόνο, και το κόστος ανάπτυξης), η διαφορά τους έγκειται στον τρόπο με τον οποίο επεμβαίνουν στην αρχιτεκτονική μιας εφαρμογής. Συγκεκριμένα η βιβλιοθήκη επεμβαίνει ελάχιστα σε αυτή, σε αντίθεση με το framework, που, συνήθως, επεμβαίνει σε μεγάλο βαθμό, καθώς παρέχει τα θεμέλια πάνω στα οποία μπορεί να χτιστεί η εφαρμογή [8].

Στην επόμενη υποενότητα θα αναφερθούν οι τεχνολογίες ανά επίπεδο της αρχιτεκτονικής της εφαρμογής.

2.7 Μοντέλο Αντικειμένου Εγγράφου

Το μοντέλο αντικειμένου εγγράφου (Document Object Model [9], ή DOM), αποτελεί ένα τρόπο απεικόνισης ενός HTML document, δηλαδή μιας διαδικτυακής σελίδας. Χρησιμοποιώντας τη διεπαφή του DOM ο προγραμματιστής μπορεί να επεμβεί στη σελίδα, αλλάζοντας τα στοιχεία της, το στυλ της, καθώς και τα δεδομένα της.

2.8 Τεχνολογίες Επιπέδου Παρουσίασης

Στο επίπεδο παρουσίασης της τριεπίπεδης αρχιτεκτονικής αντιστοιχεί ο φυλλομετρητής, που προβάλλει ιστοσελίδες στο χρήστη. Η επιλογή μας για τη γλώσσα προγραμματισμού εδώ θα είναι η JavaScript. Όλοι οι γνωστοί, μοντέρνοι φυλλομετρητές έχουν καλή υποστήριξη για τη διερμηνεία της JavaScript (για την εκτέλεση κώδικα, και λογικής, στα στοιχεία μιας ιστοσελίδας), καθώς και για την γλώσσα σήμανσης HTML και την γλώσσα CSS (για μορφοποίηση της).

Ωστόσο, για να μπορέσουμε να αναπτύξουμε μία σύγχρονη, διαδραστική, αποδοτική και επεκτάσιμη γραφική διεπαφή θα χρειαστούμε και κάποιο από τα μοντέρνα frontend διαδικτυακά πλαίσια που βασίζονται στη JavaScript. Η επιλογή μας θα είναι η React [10].

2.8.1 Η βιβλιοθήκη React

Η React, παρότι, τεχνικά, είναι μια βιβλιοθήκη, ανταγωνίζεται με τα πιο γνωστά πλαίσια JavaScript, για αυτό και, αρκετές φορές, βρίσκεται (λανθασμένα) στη συζήτηση για το καλύτερο frontend πλαίσιο. Είναι μια frontend βιβλιοθήκη ανοιχτού κώδικα, η οποία αναπτύσσεται από την Meta (πρώην Facebook), και έχει ως στόχο τη δημιουργία διαδραστικών γραφικών διεπαφών, αποτελούμενων από επαναχρησιμοποιήσιμα UI συστατικά.

Οι λόγοι που μας οδηγούν στην επιλογή της React για ανάπτυξη του frontend/πελάτη της εφαρμογής μας είναι αρκετοί. Αρχικά, η React ακολουθεί ένα προγραμματιστικό μοντέλο δημιουργίας επαναχρησιμοποιήσιμων UI συστατικών, κάτι που ευνοεί την συγγραφή «καθαρού» (δηλαδή καλογραμμένου και εύκολα κατανοητού κώδικα), και μας αποτρέπει από το να επαναλαμβάνουμε πολλές φορές τον ίδιο κώδικα, κάτι που μας επιτρέπει να αναπτύξουμε ένα frontend που θα είναι εύκολα επεκτάσιμο στο μέλλον.

Επίσης, η React αποτελεί μία πολύ αποδοτική βιβλιοθήκη, καθώς χρησιμοποιεί την ιδέα του εικονικού DOM, μιας εικονικής απεικόνισης του DOM, πάνω στο οποίο αποθηκεύει τις αλλαγές που ο προγραμματιστής έχει ορίσει για το DOM, και ενημερώνει μόνο αυτές τις αλλαγές στο DOM, αποφεύγοντας την ενημέρωση ολόκληρου του DOM.

2.9 Τεχνολογίες Επιπέδου Εφαρμογής

Στο επίπεδο εφαρμογής της εφαρμογής μας θα χρειαστούμε ένα διαδικτυακό διακομιστή καθώς και κάποια γλώσσα προγραμματισμού αυτού του διακομιστή, και, σχεδόν σίγουρα, κάποιο μοντέρνο backend διαδικτυακό πλαίσιο. Η ανάγκη χρήσης του backend διαδικτυακού πλαισίου προκύπτει από το μέγεθος της εφαρμογής που θέλουμε να αναπτύξουμε, καθώς, χρησιμοποιώντας μία απλή γλώσσα προγραμματισμού, η πολυπλοκότητα ανάπτυξης μιας ευμεγέθους διαδικτυακής εφαρμογής είναι πολύ μεγάλη. Θα πρέπει να αντιμετωπίσουμε της δρομολόγησης (routing) των http αιτημάτων στους κατάλληλους ελεγκτές/ χειριστές αιτημάτων, τη δημιουργία http αιτημάτων και το parsing των εισερχομένων, την αλληλεπίδραση με τη βάση δεδομένων και πολλά ακόμα... Αυτό είναι κάτι που δεν θα θέλαμε, καθώς θα αναγκαζόμασταν να εστιάσουμε σε αρχιτεκτονικές και διαδικτυακά πρωτόκολλα, όχι στην ίδια τη λογική της εφαρμογής που θέλουμε να υλοποιήσουμε, και, πιθανότατα, θα την υλοποιούσαμε με λιγότερη ασφάλεια (πιθανά κενά ασφαλείας), χειρότερη απόδοση, και αρκετά σφάλματα που θα προέκυπταν στην πορεία.

2.9.1 Το Περιβάλλον Χρόνου Εκτέλεσης Node.js

Η επιλογής μας για backend γλώσσα προγραμματισμού θα είναι ξανά η JavaScript, με το Node.js περιβάλλον χρόνου εκτέλεσης [11]. Το Node.js είναι ένα ασύγχρονο περιβάλλον χρόνου εκτέλεσης το οποίο είναι φτιαγμένο ειδικά για να υποστηρίζει την δημιουργία διαδικτυακών διακομιστών.

Το μοντέλο που ακολουθεί θεωρείται ασύγχρονο καθώς υποστηρίζει τις non-blocking I/O κλήσεις, δηλαδή κλήσεις μεθόδων I/O, οι οποίες δεν μπλοκάρουν τη συνέχιση των επόμενων εντολών κώδικα, αλλά καλούν κάποια callback μέθοδο όταν υπολογιστεί αυτό που επιστρέφουν.

Παρότι το Node.js, σε εφαρμογές όπου απαιτείται σημαντικός υπολογισμός, από τη μεριά του διακομιστή, έχει κάποιο χρόνο καθυστέρησης, λόγω της φύσης της JavaScript, η οποία δεν υποστηρίζει πολυνηματικότητα (multithreading), η εφαρμογή μας δεν απαιτεί κάτι τέτοιο. Μάλιστα, η κύρια απαίτηση της εφαρμογής μας είναι οι ασύγχρονες κλήσεις API, και κλήσεις στον διακομιστή βάσεων δεδομένων μας (για τον

οποίο θα μιλήσουμε στη συνέχεια). Κάτι για το οποίο οι non-blocking κλήσεις δικτύου και βάσης είναι ιδανικές.

Επίσης, ένας ακόμα λόγος για την επιλογή του Node.js είναι η χρήση της JavaScript ως γλώσσας προγραμματισμού του περιβάλλοντος χρόνου εκτέλεσης. Έχοντας τη δυνατότητα για κοινή χρήση της JavaScript, τόσο στον πελάτη, όσο και στο διακομιστή, μπορούμε να έχουμε μεταφορά δεδομένων σε μορφή JSON, πολύ εύκολα και αποδοτικά, λόγω της εγγενούς (native) υποστήριξης της JavaScript για μετατροπή των JavaScript αντικειμένων από και προς JSON. Ένα ακόμα θετικό στοιχείο από την κοινή χρήση της JavaScript σε frontend και backend είναι η συγγραφή καλύτερου κώδικα, καθώς δεν θα χρειάζεται η χρήση διαφορετικής γλώσσας στο backend, κάτι που, σίγουρα, θα επηρέαζε το προγραμματιστικό στυλ σε κάποιο από τα δύο αρνητικά, αν αναλογιστούμε τις διαφορετικές γλωσσικές συμβάσεις, αλλά και δομές δεδομένων που θα είχαν οι δύο γλώσσες.

2.9.2 Το πλαίσιο Express.js

Το πλαίσιο που θα χρησιμοποιήσουμε στο backend της εφαρμογής μας είναι το Express.js. Το Express.js [12] εστιάζει στην απόδοση, παρέχοντας μόνο τα απαραίτητα χαρακτηριστικά για την ανάπτυξη μιας σύγχρονης διαδικτυακής εφαρμογής, χωρίς να μειώνει το ρυθμό εξυπηρέτησης αιτημάτων του Node.js.

Επίσης, είναι αρκετά δημοφιλές και χρησιμοποιημένο, είναι, δηλαδή, ένα «ώριμο» πλαίσιο. Υπάρχουν αρκετά πακέτα για να λύσουν σχεδόν κάθε πρόβλημα που θα χρειαστεί να αντιμετωπίσουμε κατά την ανάπτυξη της εφαρμογής μας, κάτι το οποίο είναι αρκετά σημαντικό κατά τη δημιουργία μιας καινούριας εφαρμογής, καθώς, η αδυναμία επίλυσης συγκεκριμένων προβλημάτων που δεν μπορούν να προβλεφθούν κατά την σχεδίαση της εφαρμογής θα μπορούσε να αποτελέσει μελλοντικό αδιέξοδο για την επιτυχή ανάπτυξή της.

2.10 Τεχνολογίες Επιπέδου Αποθήκευσης

Για το επίπεδο αποθήκευσης, της αρχιτεκτονικής που θα ακολουθήσουμε, θα χρειαστούμε κάποιο σύστημα διαχείρισης βάσεων δεδομένων, ώστε να μπορούμε να αποθηκεύουμε και να ανακτούμε τα δεδομένα της εφαρμογής μας. Η καλύτερη επιλογή που είχαμε στη διάθεσή μας ήταν η MongoDB.

2.10.1 Το σύστημα διαχείρισης βάσεων δεδομένων MongoDB

Η MongoDB [13] αποτελεί ένα μη σχεσιακό DBMS (NoSQL). Τα δεδομένα αποθηκεύονται σε μορφή εγγράφων (documents), τα οποία έχουν μορφή JSON. Συγκριτικά με τα σχεσιακά DBMSs, αυτό μας δίνει δύο βασικά πλεονεκτήματα:

1. Τα έγγραφα που αποθηκεύονται στη βάση μεταφράζονται άμεσα σε JavaScript αντικείμενα. Αυτό έχει ως συνέπεια να αποφεύγεται το κόστος και η δυσκολία μετατροπής των δεδομένων από και προς JavaScript αντικείμενα, αλλά και JSON.
2. Τα έγγραφα έχουν δυναμικό σχήμα (schema). Αυτό μας επιτρέπει να μπορούμε εύκολα να αλλάξουμε τη δομή των δεδομένων στη βάση, ώστε να μπορούμε να υποστηρίξουμε αλλαγές και έξτρα λειτουργικότητα (στο μέλλον) στην εφαρμογή μας.
3. Οι βάσεις μπορούν να κλιμακωθούν οριζόντια (δηλαδή κατανεμημένα) μέσω της μεθόδου «τεμαχισμού» (sharding) που η MongoDB χρησιμοποιεί για να κατανέμει δεδομένα σε πολλά διαφορετικά συστήματα.

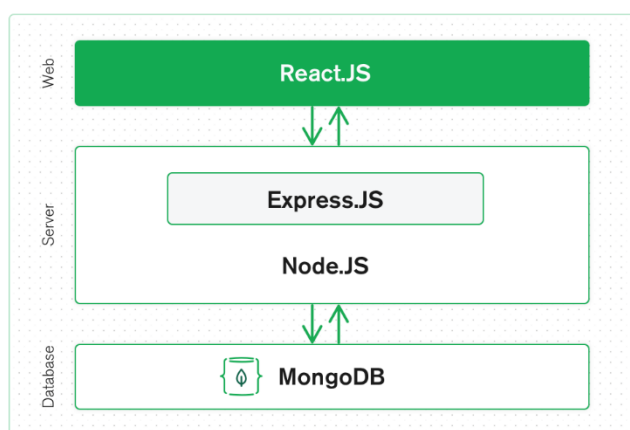
Επίσης υποστηρίζει υψηλή απόδοση. Μπορεί να απαντήσει ταχύτατα σε ερωτήματα, καθώς δεν χρειάζονται σχεσιακές ενώσεις (τα οποία κοστίζουν σε απόδοση) για να γίνει η απάντηση του ερωτήματος. Λόγω των ενσωματωμένων αντικειμένων ως τιμές, μειώνεται η ανάγκη για πολλές διαφορετικές ανακτήσεις δεδομένων από τη βάση (καθώς με ένα ερώτημα ανακτώνται πολλά δεδομένα σε μορφή ενσωματωμένων αντικειμένων). Τέλος, η δυνατότητα για ύπαρξη ευρετηρίων στα ενσωματωμένων αντικειμένων αυξάνει ακόμα περισσότερο την απόδοσή της.

2.11 Στοιίβα MERN

Στοιίβα λογισμικού (software stack), ορίζεται ως ο συνδυασμός από τεχνολογίες που μπορούν να αποτελέσουν μία λύση για την ανάπτυξη μίας εφαρμογής τόσο στο frontend, όσο και στο backend. Η κοινή χρήση της βιβλιοθήκης React, του Node.js περιβάλλοντος χρόνου εκτέλεσης, του Express.js πλαισίου, και της βάσης MongoDB, για τη σχεδίαση μιας τριεπίπεδης αρχιτεκτονικής, καλείται στοιίβα MERN [14] (MERN stack) από τα αρχικά των τεχνολογιών που την απαρτίζουν. Η στοιίβα MERN αποτελείται από τη βιβλιοθήκη React στο frontend, το πλαίσιο Express.js που τρέχει πάνω σε κάποιο Node.js διακομιστή στο επίπεδο εφαρμογής, και τον MongoDB διακομιστή βάσεων δεδομένων στο επίπεδο αποθήκευσης.

2.11.1 Μεταφορά δεδομένων μεταξύ των τριών επιπέδων

Τα δεδομένα στην εφαρμογή μας θα μεταφέρονται μεταξύ των επιπέδων σε μορφή JSON. Η React θα στέλνει HTTP αιτήματα (με δεδομένα στη μορφή JSON, σε περίπτωση ύπαρξης ωφέλιμου φορτίου στο αίτημα) στον Express διαδικτυακό διακομιστή, ο οποίος, με τη σειρά του, θα κάνει κλήσεις στον MongoDB διακομιστή βάσεων δεδομένων, για ανάκτηση και αποθήκευση δεδομένων, από και προς τη βάση.



Σχήμα 2.1 Σχηματική απεικόνιση της αλληλεπίδρασης των τριών επιπέδων στη στοίβα MERN.

Η χρήση της JavaScript, ως γλώσσα ανάπτυξης τόσο του επιπέδου παρουσίασης, όσο και του επιπέδου λογικής, μας επιτρέπει να έχουμε φυσική υποστήριξη για JSON και στα δύο επίπεδα, συνεπώς καλύτερη απόδοση για μετατροπή σε JSON από JavaScript αντικείμενα, και το αντίθετο. Επίσης, στο επίπεδο αποθήκευσης η MongoDB αποθηκεύει τα δεδομένα σε μορφή JSON (τεχνικά τα αποθηκεύει σε μορφή BSON, μια δυαδική έκδοση της JSON μορφής), συνεπώς, έχουμε φυσική υποστήριξη για JSON, και στα τρία tiers, και μπορούμε να δουλέψουμε μόνο με JSON δεδομένα σε ολόκληρη την εφαρμογή μας.

2.12 Gmail API

Για να μπορέσει η εφαρμογή μας να διαβάζει, και να στέλνει, μηνύματα, από, και προς, το Gmail λογαριασμό του χρήστη, θα πρέπει να χρησιμοποιήσουμε το Gmail API [15]. Το Gmail API είναι ένα RESTful διαδικτυακό API, το οποίο έχει υλοποιήσει η

Google, και μπορεί να χρησιμοποιηθεί για πρόσβαση στο Gmail γραμματοκιβώτιο του χρήστη, καθώς και για αποστολή email.

2.13 Microsoft Graph API

Για να μπορέσει η εφαρμογή μας να έχει πρόσβαση στα chats του χρήστη, καθώς και να δημιουργεί καινούρια, αλλά και να στέλνει μηνύματα, σε αυτά, θα πρέπει να χρησιμοποιήσουμε το Microsoft Graph API [16]. Το Microsoft Graph API είναι, και αυτό, ένα RESTful διαδικτυακό API, το οποίο έχει υλοποιήσει η Microsoft για πρόσβαση στα chats του Microsoft Teams λογαριασμού του χρήστη, καθώς και αποστολή μηνυμάτων, μεταξύ άλλων υπηρεσιών νέφους που παρέχει.

2.14 Κουπόνια πρόσβασης και ανανέωσης

Τα κουπόνια πρόσβασης (access tokens) [17] είναι συμβολοσειρές που δηλώνουν κάποια χαρακτηριστικά πρόσβασης. Σε πρωτόκολλα εξουσιοδότησης εκδίδονται σε τρίτους πελάτες (για παράδειγμα μία εφαρμογή) από ένα διακομιστή εξουσιοδότησης με την έγκριση του ιδιοκτήτη του πόρου. Στη συνέχεια, ο πελάτης χρησιμοποιεί το κουπόνι πρόσβασης για να αποκτήσει πρόσβαση στους προστατευόμενους πόρους που φιλοξενεί ο διακομιστής πόρων. Στο πρωτόκολλο εξουσιοδότησης OAuth 2.0 (το αναφέρουμε στην επόμενη υποενότητα), εκτός από τα κουπόνια πρόσβασης υπάρχουν, προαιρετικά, και κουπόνια ανανέωσης (refresh tokens), τα οποία χρησιμοποιούνται για να ανακτηθούν καινούρια κουπόνια πρόσβασης, όταν τα παλιά λήξουν, καθώς τα κουπόνια πρόσβασης στο πρωτόκολλο αυτό έχουν μικρή διάρκεια ζωής, για λόγους ασφαλείας.

2.15 Το πρωτόκολλο εξουσιοδότησης OAuth 2.0

Για να έχει πρόσβαση στους λογαριασμούς του χρήστη, εκ μέρους του, η εφαρμογή μας χρησιμοποιεί το πρωτόκολλο εξουσιοδότησης OAuth 2.0 [18]. Το πρωτόκολλο αυτό, επιτρέπει στο χρήστη να εξουσιοδοτήσει την εφαρμογή μας, ώστε αυτή να αποκτήσει περιορισμένη πρόσβαση στο λογαριασμό του, μέσω κάποιας HTTP υπηρεσίας (για παράδειγμα Gmail API, Microsoft Graph API). Οι οντότητες που συμμετέχουν σε αυτό είναι οι εξής:

- **Ιδιοκτήτης του πόρου (resource owner):** ο χρήστης που εξουσιοδοτεί μια εφαρμογή να έχει πρόσβαση στο λογαριασμό του

- **Πελάτης (client):** η εφαρμογή που θέλει να αποκτήσει πρόσβαση στο λογαριασμό του χρήστη
- **Διακομιστής πόρων (resource server):** ο διακομιστής που φιλοξενεί τους προστατευόμενους λογαριασμούς χρηστών
- **Διακομιστής εξουσιοδότησης (authorization server):** ο διακομιστής που επαληθεύει την ταυτότητα του χρήστη και στη συνέχεια εκδίδει κουπόνια πρόσβασης στην εφαρμογή

Το πρωτόκολλο υποστηρίζει διαφορετικές ροές εξουσιοδότησης για διαφορετικούς τύπους εφαρμογών. Η εφαρμογή μας ακολουθεί τη ροή της εξουσιοδότησης μέσω κωδικού εξουσιοδότησης (authorization code grant), τα βήματα της οποίας είναι τα εξής:

1. Ο χρήστης λαμβάνει έναν σύνδεσμο κωδικού εξουσιοδότησης
2. Ο χρήστης εξουσιοδοτεί την εφαρμογή για πρόσβαση στους λογαριασμούς του
3. Η εφαρμογή λαμβάνει τον κωδικό εξουσιοδότησης από τον διακομιστή εξουσιοδότησης
4. Η εφαρμογή χρησιμοποιεί τον κωδικό εξουσιοδότησης για να λάβει **κουπόνι πρόσβασης**, και, στην περίπτωση που ζητηθεί, **κουπόνι ανανέωσης**, αλλά και **πληροφορίες για το προφίλ του χρήστη**, από το διακομιστή εξουσιοδότησης
5. Η εφαρμογή λαμβάνει το κουπόνι πρόσβασης και μπορεί να το χρησιμοποιήσει για πρόσβαση στο λογαριασμό του χρήστη

2.16 HTTP Cookie

Το HTTP cookie είναι μία μακριά συμβολοσειρά (μέχρι 4KB) που μπορεί να στείλει ένας διακομιστής στον περιηγητή ιστού ενός χρήστη. Ο περιηγητής ιστού μπορεί να αποθηκεύσει το cookie και να το στείλει πίσω στον ίδιο διακομιστή σε μεταγενέστερα αιτήματα. Η εφαρμογή μας, για να δημιουργεί συνεδρίες με συνδεδεμένους χρήστες, ώστε να παραμένουν συνδεδεμένοι μέχρι να κάνουν αποσύνδεση από την εφαρμογή, αποθηκεύει το αναγνωριστικό της ενεργής συνεδρίας του χρήστη ως cookie στον υπολογιστή του [19].

Κεφάλαιο 3. Σχεδίαση

3.1 Απαιτήσεις της εφαρμογής

Στο εισαγωγικό κεφάλαιο, είχαμε αναφέρει τους στόχους τους οποίους θέλουμε να εκπληρώσει η εφαρμογή μας. Οι στόχοι αυτοί μπορούν να διακριθούν σε λειτουργικές, και μη λειτουργικές απαιτήσεις. Πιο συγκεκριμένα, οι λειτουργικές απαιτήσεις που έχουμε θέσει για την εφαρμογή μας είναι οι εξής:

1. Ο χρήστης πρέπει να μπορεί να συνδεθεί στην εφαρμογή με ένα μοναδικό όνομα χρήστη και ένα συνθηματικό, ώστε μόνο αυτός να έχει πρόσβαση στα δεδομένα του.
2. Ο χρήστης πρέπει να μπορεί να εξουσιοδοτήσει την εφαρμογή ώστε να έχει δυνατότητα πρόσβασης στα δεδομένα των λογαριασμών του (Gmail, Microsoft Teams).
3. Ο χρήστης θα πρέπει να μπορεί να ανακτά τα εισερχόμενα μηνύματα των λογαριασμών του, μέσω της εφαρμογής.
4. Ο χρήστης θα πρέπει να μπορεί να στείλει μηνύματα, μέσω της εφαρμογής, που να προέρχονται από τους αντίστοιχους λογαριασμούς του.
5. Ο χρήστης θα πρέπει να μπορεί να οργανώνει τα μηνύματα ηλεκτρονικού ταχυδρομείου του σε νήματα, και να τα επισημαίνει ως σημαντικά, ή αδιάβαστα, ώστε να τα διαχειρίζεται καλύτερα.

Αντίστοιχα, οι μη λειτουργικές απαιτήσεις που έχουμε θέσει είναι οι εξής:

1. Καλή απόδοση, ώστε ο χρήστης να μπορεί να ανακτά γρήγορα τα εισερχόμενα μηνύματα των λογαριασμών του, αλλά και να απαντά γρήγορα σε μηνύματα.
2. Ασφάλεια και καλή άμυνα σε επιθέσεις, ώστε να εξασφαλιστεί η εμπιστευτικότητα, και ακεραιότητα των δεδομένων του χρήστη.

3. Όμορφη και σύγχρονη γραφική διεπαφή, καθώς και διαδραστικό frontend, για να προσφέρει μία άρτια εμπειρία στο χρήστη.

Στην υποενότητα **3.4** θα αναφέρουμε τις λειτουργικές απαιτήσεις της εφαρμογής πιο αναλυτικά, όσον αφορά την αλληλεπίδραση του χρήστη με τη γραφική διεπαφή της εφαρμογής.

3.2 Αυθεντικοποίηση χρήστη

Το γεγονός ότι θέλουμε η εφαρμογή μας να υποστηρίζει την πρόσβαση σε προσωπικά δεδομένα του κάθε χρήστη, μέσω της πρόσβασης στους προσωπικούς Gmail και Microsoft Teams λογαριασμούς του, μας αναγκάζει να έχουμε κάποιο τρόπο αυθεντικοποίησης του κάθε χρήστη, ώστε να μπορούμε να του δίνουμε ελεγχόμενη πρόσβαση στα δεδομένα που του ανήκουν, και μόνο αυτά. Έτσι, κάθε χρήστης θα μπορεί να έχει πρόσβαση μόνο στα δικά του δεδομένα, και σε κανενός άλλου.

Προφανώς, για να μπορούμε να αυθεντικοποιούμε ένα χρήστη, και να τον συνδέουμε στην εφαρμογή, θα πρέπει πρώτα ο χρήστης αυτός να είναι εγγεγραμμένος στην εφαρμογή, ώστε να γνωρίζουμε τα στοιχεία του, τα οποία θα μας έχει δώσει κατά την εγγραφή του. Συγκεκριμένα, θα του επιτρέψουμε την εγγραφή στην εφαρμογή επιλέγοντας ένα μοναδικό όνομα χρήστη (το οποίο θα λειτουργεί ως μοναδικό αναγνωριστικό του χρήστη), καθώς και ένα προσωπικό συνθηματικό. Το συνθηματικό θα μας επιβεβαιώνει ότι ο χρήστης είναι αυτός που ισχυρίζεται, αφού μόνο ο ίδιος ο χρήστης θα γνωρίζει το προσωπικό του συνθηματικό.

Έχοντας τα στοιχεία του χρήστη αποθηκευμένα στη βάση μας, μετά την εγγραφή του, μπορούμε να τα συγκρίνουμε, κάθε φορά, με τα στοιχεία που θα μας δίνει ένας χρήστης που θα θέλει να χρησιμοποιήσει την εφαρμογή, κατά τη σύνδεσή του, και να τον συνδέουμε στην εφαρμογή, εάν αυτά είναι σωστά, ή να του απαγορεύουμε τη σύνδεση, εάν είναι λανθασμένα.

3.3 Εξουσιοδότηση εφαρμογής για πρόσβαση στους λογαριασμούς του χρήστη

Για να μπορεί η εφαρμογή μας να διαχειρίζεται τους λογαριασμούς ενός χρήστη, ο χρήστης θα πρέπει να την εξουσιοδοτήσει. Η εξουσιοδότηση του χρήστη θα δοθεί στον

πάροχο της κατάλληλης υπηρεσίας/λογαριασμού (Google, Microsoft), έτσι ώστε ο πάροχος με τη σειρά του να δώσει πρόσβαση στην εφαρμογή στις υπηρεσίες που παρέχουν τα διαδικτυακά APIs που έχει αναπτύξει, μέσω κουπονιών πρόσβασης. Όλη αυτή η διαδικασία θα γίνει μέσω του πρωτοκόλλου OAuth 2.0, μεταξύ εφαρμογής, χρήστη, και του εκάστοτε παρόχου.

3.4 Ιστορίες χρήστη της εφαρμογής

Ιστορίες χρήστη, στην ανάπτυξη λογισμικού, καλούνται τα χαρακτηριστικά ενός συστήματος, στη μορφή φυσικής γλώσσας. Κάθε ιστορία χρήστη έχει τη μορφή “Ως χρήστης, θέλω να μπορώ να ..., έτσι ώστε να ...”. Σε αυτή την υποενότητα θα αναφέρουμε τα χαρακτηριστικά της εφαρμογής, με τη μορφή ιστοριών χρήστη, ώστε να αναδείξουμε τα αιτήματα που θα μπορεί να κάνει ένας χρήστης στην εφαρμογή μας. Οι ιστορίες χρήστη είναι οι ακόλουθες:

1. Γενικές ιστορίες χρήστη:

- 1.1 Ως χρήστης, θέλω να μπορώ να δημιουργήσω ένα καινούριο λογαριασμό στην εφαρμογή εισάγοντας τα στοιχεία μου στη φόρμα της σελίδας εγγραφής και κάνοντας υποβολή. Η εφαρμογή πρέπει να με ανακατευθύνει στη Home σελίδα.
- 1.2 Ως χρήστης, θέλω να μπορώ να συνδεθώ στην εφαρμογή, εισάγοντας τα στοιχεία μου στη φόρμα της σελίδας σύνδεσης και κάνοντας υποβολή. Η εφαρμογή πρέπει να με ανακατευθύνει στη Home σελίδα.
- 1.3 Ως χρήστης, θέλω να μπορώ να συνδέσω τους δύο Gmail, και το Microsoft Teams λογαριασμό μου στην εφαρμογή. Στη Home σελίδα της εφαρμογής θα πρέπει να υπάρχουν τρία tabs, ένα για κάθε λογαριασμό, από όπου θα έχω πρόσβαση σε όλες τις λειτουργίες που αφορούν το λογαριασμό αυτό. Θα πρέπει πάνω από τα tabs να υπάρχουν τα κουμπιά Gmail #1, Gmail #2, MsTeams, και πατώντας τα να ενεργοποιείται το κατάλληλο tab, για πρόσβαση στον κατάλληλο λογαριασμό.
- 1.4 Ως χρήστης θέλω να μπορώ να αποσυνδεθώ από την εφαρμογή, πατώντας το κουμπί αποσύνδεσης από το navbar της Home σελίδας. Η εφαρμογή πρέπει να με ανακατευθύνει στη σελίδα σύνδεσης.

2. Ιστορίες χρήστη σχετικές με τους Gmail λογαριασμούς:

- 2.1 Ως χρήστης, θέλω να μπορώ να ανακτήσω τα εισερχόμενα μηνύματα των 2 Gmail λογαριασμών μου, από τη Home σελίδα της εφαρμογής και το κατάλληλο tab. Τα μηνύματα θα πρέπει να είναι χωρισμένα σε δύο λίστες, αδιάβαστα και σημαντικά, καθώς και να είναι οργανωμένα σε νήματα συνομιλιών, το ένα κάτω από το άλλο.
- 2.2 Ως χρήστης, θέλω να μπορώ να προβάλλω όλα τα μηνύματα ενός νήματος από κάποιο Gmail λογαριασμό πατώντας πάνω σε αυτό. Το νήμα θα πρέπει να ανοίγει και να εμφανίζει μία λίστα με όλα τα μηνύματα της συνομιλίας.
- 2.3 Ως χρήστης, θέλω να μπορώ να στείλω ένα μήνυμα σε κάποιο ήδη υπάρχων νήμα, από κάποιο Gmail λογαριασμό. Ανοίγοντας το νήμα, θα πρέπει μετά το πιο πρόσφατο μήνυμα να υπάρχει μία φόρμα που μου θα μου επιτρέψει να το κάνω αυτό. Επίσης θα πρέπει να υπάρχουν τα κατάλληλα κουμπιά για αποστολή συνημμένου, διαγραφή συνημμένου, καθώς και διαγραφή του μηνύματος πριν σταλεί.
- 2.4 Ως χρήστης, θέλω να μπορώ να ανακτήσω ένα συνημμένο κάποιου μηνύματος από κάποιο Gmail λογαριασμό μου μέσω της γραφικής διεπαφής της εφαρμογής. Τα συνημμένα ενός μηνύματος θα πρέπει να βρίσκονται σε μία λίστα μέσα στο μήνυμα, και πατώντας πάνω σε κάποιο από αυτά, το συνημμένο θα πρέπει να κατέβει στο λειτουργικό μου σύστημα.
- 2.5 Ως χρήστης, θέλω να μπορώ να επισημάνω ένα νήμα μηνυμάτων από κάποιο Gmail λογαριασμό μου ως διαβασμένο, ή σημαντικό. Όταν ανοίγω ένα αδιάβαστο νήμα, το νήμα θα πρέπει να χαρακτηρίζεται ως διαβασμένο. Όταν πατάω πάνω στο αστεράκι ενός νήματος θα πρέπει το νήμα να χαρακτηρίζεται ως σημαντικό και το αστεράκι του να γίνεται επιλεγμένο, ενώ εάν το νήμα ήταν ήδη σημαντικό, θα πρέπει να χαρακτηρίζεται ως μη σημαντικό, και το αστεράκι να μην είναι πλέον επιλεγμένο.
- 2.6 Ως χρήστης, θέλω να μπορώ να προβάλλω ένα μήνυμα σε μορφή απλού κειμένου, html, html με εισαγωγικά απάντησης (reply quotes), ή να μπορώ να το ελαχιστοποιήσω, ώστε να μπορώ να το δω όπως με βολεύει καλύτερα. Θα πρέπει κάθε μήνυμα να έχει τα κατάλληλα κουμπιά ώστε να γίνεται αυτό.
- 2.7 Ως χρήστης, θέλω να μπορώ να κάνω αποστολή ενός καινούριου μηνύματος από το Gmail λογαριασμό μου. Θα πρέπει να υπάρχει ένα κουμπί Compose και πατώντας το να εμφανίζεται κατάλληλη φόρμα αποστολής καινούριου μηνύματος. Επίσης θα πρέπει να υπάρχουν τα κατάλληλα κουμπιά για

αποστολή συνημμένου, διαγραφή συνημμένου, καθώς και διαγραφή του μηνύματος πριν σταλεί.

3. Ιστορίες χρήστη σχετικές με το Microsoft Teams λογαριασμό:

- 3.1 Ως χρήστης, θέλω να μπορώ να έχω ανακτήσω όλα τα chats από το Microsoft Teams λογαριασμό μου από τη Home σελίδα της εφαρμογής, και το κατάλληλο tab. Θα πρέπει αριστερά να υπάρχει μία λίστα με τα μέλη που συμμετέχουν σε κάθε chat, και δεξιά να υπάρχει μία λίστα από μηνύματα του επιλεγμένου chat. Τα μηνύματα ενός chat θα πρέπει να προβάλλονται στο δεξί μέρος όταν πατάω πάνω στο κατάλληλο chat από την αριστερή λίστα.
- 3.2 Ως χρήστης, θέλω να μπορώ να δημιουργήσω ένα καινούριο chat για το Microsoft Teams λογαριασμό μου. Θα πρέπει να υπάρχει ένα input πεδίο που να μου επιτρέπει να εισάγω τη διεύθυνση email του χρήστη με τον οποίο θέλω να δημιουργήσω καινούρια συνομιλία, και πατώντας enter η συνομιλία να δημιουργείται και να επιλέγεται από τη λίστα με τα chats.
- 3.3 Ως χρήστης, θέλω να μπορώ να κάνω αποστολή ενός καινούριου μηνύματος σε chat του Microsoft Teams λογαριασμού μου. Έχοντας επιλεγμένο το chat, θα πρέπει κάτω από τα μηνύματά του να υπάρχει μία φόρμα που να μου το επιτρέπει αυτό.
- 3.4 Ως χρήστης, θέλω να μπορώ να κάνω αποστολή ενός συνημμένου σε κάποιο chat του Microsoft Teams λογαριασμού μου. Κάτω από τη φόρμα μηνύματος του επιλεγμένου chat θα πρέπει να υπάρχει κουμπί που να μου το επιτρέπει.

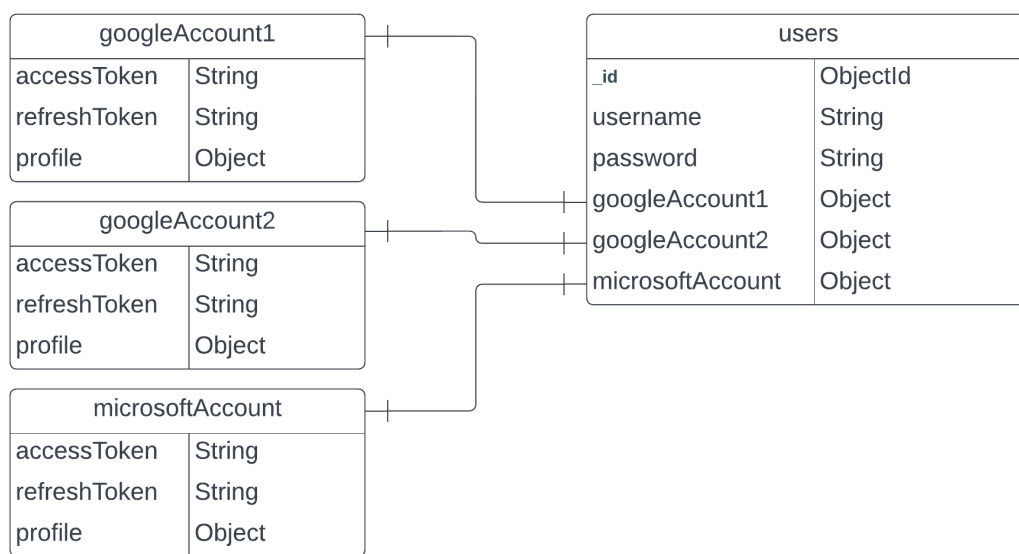
3.5 Αποθήκευση δεδομένων

Η βασική οντότητα της εφαρμογής μας είναι ο χρήστης. Κάθε χρήστης θα μπορεί να έχει έως 3 λογαριασμούς του, ταυτόχρονα, συνδεδεμένους στην εφαρμογή μας (2 Gmail λογαριασμούς, και 1 Microsoft Teams λογαριασμό). Κάθε ένας από τους 3, αυτούς, λογαριασμούς του χρήστη μπορεί να θεωρηθεί ως μια ξεχωριστή οντότητα, για την οποία η εφαρμογή μας θα πρέπει να έχει πρόσβαση στα δεδομένα του. Επίσης, κάθε χρήστης θα πρέπει να είναι και ο μοναδικός που έχει πρόσβαση στα δεδομένα του.

Η εφαρμογή μας, κατά τη ροή εξουσιοδότησης του πρωτοκόλλου OAuth 2.0, για κάθε λογαριασμό του χρήστη, θα ζητήσει από το διακομιστή εξουσιοδότησης του εκάστοτε παρόχου refresh tokens, καθώς και πληροφορίες για το προφίλ του χρήστη.

Συνεπώς, για κάθε χρήστη θα χρειαστεί να αποθηκεύσουμε το όνομα και τον κωδικό του στην εφαρμογή, καθώς και για κάθε λογαριασμό του τα access, και refresh tokens, ώστε να μπορούμε να έχουμε πρόσβαση σε αυτόν, καθώς και τα στοιχεία από το προφίλ του, όπως τη διεύθυνση email του, το όνομά του κ.τ.λ..

Κάθε έγγραφο της συλλογής αυτής θα έχει ενσωματωμένα τα στοιχεία των 3 λογαριασμών του χρήστη που περιγράψαμε, συγκεκριμένα τα πεδία τους θα ονομάζονται googleAccount1, googleAccount2, και microsoftAccount, ενώ οι τιμές τους θα είναι τύπου Object, δηλαδή θα περιέχουν και αυτά με τη σειρά τους ζεύγη κλειδιών-τιμών, που θα είναι τα στοιχεία που αναφέραμε από κάθε λογαριασμό.



Σχήμα 3.1 Διάγραμμα σχέσεων οντοτήτων για τη βάση δεδομένων που θα χρησιμοποιήσουμε στην εφαρμογή μας. Περιγράφει τη βασική οντότητα της εφαρμογής που είναι ο χρήστης, και τη σχέση του που είναι 1-1 με τους 3 λογαριασμούς του. Η συλλογή που θα αποθηκεύει την πληροφορία για κάθε χρήστη θα ονομάζεται `users`.

3.6 Ασφάλεια

Σημαντικό ζήτημα σε κάθε εφαρμογή αποτελεί η ασφάλεια των δεδομένων των χρηστών, συνεπώς η εφαρμογή μας δεν θα μπορούσε να αποτελεί εξαίρεση. Η εφαρμογή μας θα διαχειρίζεται δεδομένα των χρηστών από 3^{ες} εφαρμογές, καθώς και θα έχει πρόσβαση σε υπηρεσίες τους, για αυτό πρέπει να διασφαλίσουμε ότι δεν θα υπάρχουν κενά ασφαλείας που μπορούν να εκμεταλλευτούν οι επιτιθέμενοι για να αποκτήσουν πρόσβαση στα δεδομένα αυτά.

Αρχικά, η εφαρμογή μας θα έχει διαφορετικούς χρήστες, κάθε ένας από τους οποίους θα πρέπει να έχει πρόσβαση μόνο στα δικά του δεδομένα. Για να το πετύχουμε

αυτό, έχουμε να αποφασίσουμε μεταξύ 2 κύριων μεθόδων, τις συνεδρίες διακομιστή που βασίζονται σε cookie (cookie based, server side sessions), και τα JSON web tokens [20]. Ανάμεσα σε αυτές τις 2 επιλογές, θα επιλέξουμε την 1^η, δηλαδή τις συνεδρίες διακομιστή. Παρότι τα tokens είναι μια καλή επιλογή, καθώς ελαχιστοποιούν τις φορές που ο χρήστης χρειάζεται να εισάγει τα διαπιστευτήριά του, επειδή η πληροφορία που χρειάζεται για να αυθεντικοποιηθεί ο χρήστης είναι αποθηκευμένη στον πελάτη (και όχι στο διακομιστή, σε αντίθεση με τις συνεδρίες), η εφαρμογή μας θα αποτελείται μόνο από ένα backend διακομιστή, συνεπώς είναι κάτι που δε χρειαζόμαστε. Επίσης, οι συνεδρίες διακομιστή μας επιτρέπουν καλύτερο έλεγχο στην πρόσβαση του χρήστη στο διακομιστή/API, καθώς μπορούμε να τερματίσουμε τη συνεδρία του χρήστη εάν παρατηρήσουμε κάτι ύποπτο, κάτι το οποίο δε μπορούμε να κάνουμε με τα tokens, που είναι αποθηκευμένα στον πελάτη.

Έχοντας αποφασίσει τον τρόπο, με τον οποίο οι χρήστες θα έχουν πρόσβαση στην εφαρμογή μας, ήρθε η ώρα να βρούμε κάποια μέθοδο ώστε να έχουμε ασφαλή αποθήκευση των ευαίσθητων στοιχείων των χρηστών στη βάση. Το 1^ο ευαίσθητο στοιχείο που θα χρειαστεί να αποθηκεύσουμε στη βάση, για κάθε χρήστη, είναι το συνθηματικό που θα εισάγει κατά την εγγραφή του στην εφαρμογή. Το συνθηματικό ενός χρήστη είναι ένα ευαίσθητο στοιχείο, που χρειάζεται να γνωρίζει μόνο ο ίδιος ο χρήστης, καθώς χρειάζεται μόνο για την αυθεντικοποίησή του, ώστε να έχει πρόσβαση στις υπηρεσίες της εφαρμογής. Το γεγονός αυτό, μας οδηγεί στο να χρησιμοποιήσουμε κάποια μέθοδο κατακερματισμού (hashing), πριν αποθηκεύσουμε τον κωδικό στη βάση. Οι συναρτήσεις κατακερματισμού (hashing functions) έχουν την ιδιότητα να μπορούν να κωδικοποιήσουν ένα κείμενο εισόδου, με τέτοιο τρόπο, ώστε η αποκωδικοποίησή του να είναι, πρακτικά, αδύνατη, ακριβώς, δηλαδή, αυτό που θέλουμε για τους κωδικούς των χρηστών της εφαρμογής μας. Συγκεκριμένα, η συνάρτηση κατακερματισμού που θα χρησιμοποιήσουμε, είναι η bcrypt [21]. Η bcrypt, είναι μία συνάρτηση κατακερματισμού, βασισμένη στον αλγόριθμο Blowfish, και είναι μία από τις κύριες συναρτήσεις κατακερματισμού που χρησιμοποιούνται για κατακερματισμό συνθηματικού. Οι λόγοι που θα την χρησιμοποιήσουμε είναι ότι περιέχει άμυνα ενάντια σε rainbow table attacks, δηλαδή επιθέσεις που έχουν υπολογισμένα hashes για πάρα πολλά συνθηματικά, και τα χρησιμοποιούν για σύγκριση με το hash του χρήστη στη βάση, καθώς στο κείμενο εισόδου του προστίθεται salt. Το salt είναι ένας τυχαία υπολογισμένος αριθμός, έτσι ώστε 2 ίδιοι κωδικοί με διαφορετικό salt, να παράγουν διαφορετικό hash. Επίσης, είναι προσαρμόσιμη, κάτι που την κάνει ανθεκτική σε επιθέσεις brute force, καθώς, μια προσαρμόσιμη συνάρτηση

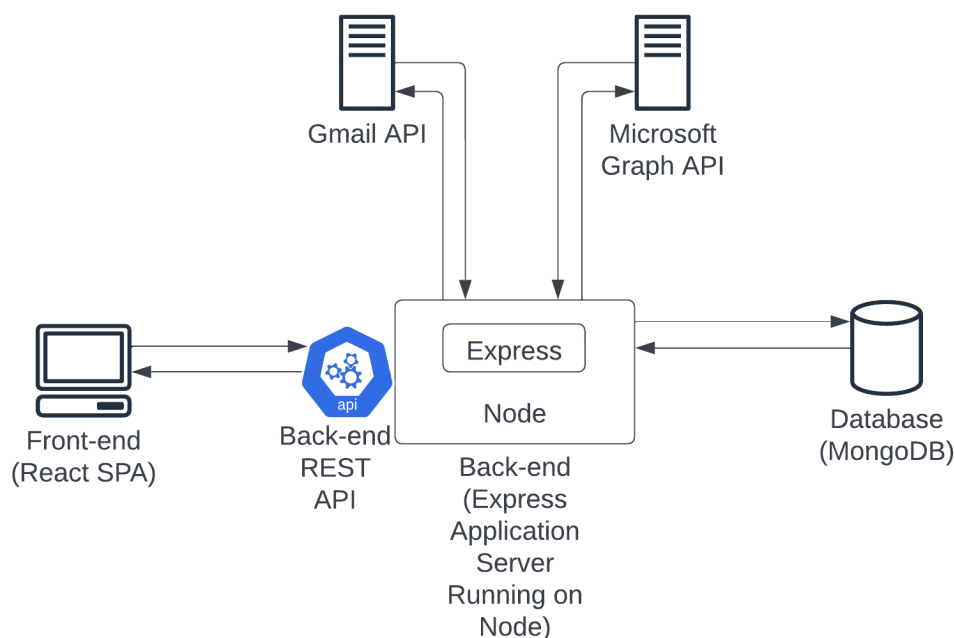
κατεκερματισμού χρειάζεται περισσότερη ώρα για να εκτελεστεί, και να παράγει hash εξόδου.

Το 2^ο ευαίσθητο στοιχείο, που θα χρειαστεί να αποθηκεύσουμε στη βάση, είναι τα access, και refresh tokens, τα οποία θα χρειαστεί η εφαρμογή μας, ώστε να έχει πρόσβαση στα 2 διαδικτυακά APIs που θα χρησιμοποιήσουμε. Κατά την εκτέλεση του πρωτοκόλλου OAuth 2.0, τα 2 διαδικτυακά APIs (Gmail API, Microsoft Graph API) θα δημιουργήσουν access, και refresh tokens, τα οποία θα δώσουν στην εφαρμογή μας, και τα οποία θα πρέπει να κρυπτογραφήσουμε, πριν τα αποθηκεύσουμε στη βάση, έτσι ώστε να τα αποκρυπτογραφούμε μόνο όταν χρειάζεται να στείλουμε κάποιο αίτημα στα διαδικτυακά APIs. Ο αλγόριθμος, που θα χρησιμοποιήσουμε για αυτή τη δουλειά, είναι ο AES-256-CBC. Ο αλγόριθμος AES (Advanced Encryption Standard) [22] είναι από τους πιο δυνατούς, αν όχι ο πιο δυνατός, αλγόριθμος κρυπτογράφησης (είναι πρότυπο της ομοσπονδιακής κυβέρνησης των ΗΠΑ, για κρυπτογράφηση ηλεκτρονικών δεδομένων, από τις 26 Μαΐου του 2002), και, μέχρι σήμερα, δεν έχουν βρεθεί πρακτικές επιθέσεις εναντίων του. Έχει 3 εκδοχές, με διαφορετικά μεγέθη κλειδιών, 128, 192 και 256 bits, με τα 256 bits να αποτελούν το μέγεθος κλειδιού που δίνει το πιο ασφαλές αποτέλεσμα, ενώ το μέγεθος κειμένου εισόδου του είναι 128 bits. Για να μπορέσουμε να κρυπτογραφήσουμε τα OAuth tokens που μπορούν να έχουν μέγεθος μεγαλύτερο από 128 bits, θα ακολουθήσουμε το CBC (Cipher block chaining) mode κρυπτογράφησης [23]. Το CBC mode μας επιτρέπει για 2 ίδια κομμάτια (block) κειμένου που κρυπτογραφούμε να έχουμε διαφορετική έξοδο, καθώς η έξοδος ενός κομματιού εξαρτάται από την έξοδο του προηγούμενου κομματιού. Το αρχικό κομμάτι του κειμένου κρυπτογραφείται μαζί με ένα διάνυσμα αρχικοποίησης (Initialization Vector, ή IV), δηλαδή ένα τυχαίο διάνυσμα (πρέπει να είναι τυχαίο ώστε να μειώνεται η πιθανότητα 2 ίδια κομμάτια να δίνουν πάντα ίδια έξοδο, μετά την κρυπτογράφηση), και, έτσι, οι πιθανότητες αποκρυπτογράφησης μειώνονται δραματικά.

Χρησιμοποιώντας τις συνεδρίες διακομιστή, η εφαρμογή μας θα παρέχει διασφάλιση εμπιστευτικότητας για το λογαριασμό του χρήστη. Μόνο ένας χρήστης που γνωρίζει τον κωδικό του θα έχει πρόσβαση στο λογαριασμό του. Αυτό περιλαμβάνει, προφανώς, εμπιστευτικότητα και για τα δεδομένα των συνδεδεμένων λογαριασμών του στην εφαρμογή. Επίσης, κατακερματίζοντας, ή κρυπτογραφώντας τα ευαίσθητα δεδομένα, η εφαρμογή μας θα παρέχει διασφάλιση ότι σε περίπτωση παραβίασης της βάσης δεδομένων, ο επιτιθέμενος δεν θα έχει πρόσβαση ούτε στο λογαριασμό του χρήστη, ούτε και σε κάποιον από τους λογαριασμούς που είχε συνδέσει στην εφαρμογή, καθώς ο κωδικός και τα access tokens του θα είναι κρυπτογραφημένα.

3.7 Αρχιτεκτονική της εφαρμογής

Όπως αναφέραμε και στο κεφάλαιο του υπόβαθρου, το frontend της εφαρμογής μας, θα είναι μία εφαρμογή μονής σελίδας σε React. Η εφαρμογή αυτή θα δέχεται αιτήματα από το χρήστη (μέσω της γραφικής διεπαφής), και για να τα ικανοποιήσει θα στέλνει αιτήματα στο backend διακομιστή εφαρμογής, μέσω του RESTful API του. Ο backend διακομιστής εφαρμογής, με τη σειρά του, θα ικανοποιεί τα αιτήματα του frontend, επικοινωνώντας, όταν χρειάζεται, με τα διαδικτυακά APIs των παρόχων (δηλαδή το Gmail API, και το Microsoft Graph API), και θα αποκρίνεται, τελικά στο frontend, το οποίο θα ενημερώνει το DOM της σελίδας του, αναλόγως την περίπτωση. Ο διακομιστής εφαρμογής θα επικοινωνεί, επίσης με τη βάση δεδομένων για να ανακτά και να αποθηκεύει δεδομένα σε κάθε λειτουργία.



Σχήμα 3.2 Σε αυτό το σχήμα απεικονίζεται η αρχιτεκτονική της εφαρμογής μας. Το front-end θα στέλνει αιτήματα στο back-end, μέσω ενός RESTful API. Το back-end, με τη σειρά του, θα επικοινωνεί με τα διαδικτυακά APIs των παρόχων, ικανοποιώντας τα αιτήματα του χρήστη, καθώς και με τη βάση, για ανάκτηση και αποθήκευση δεδομένων.

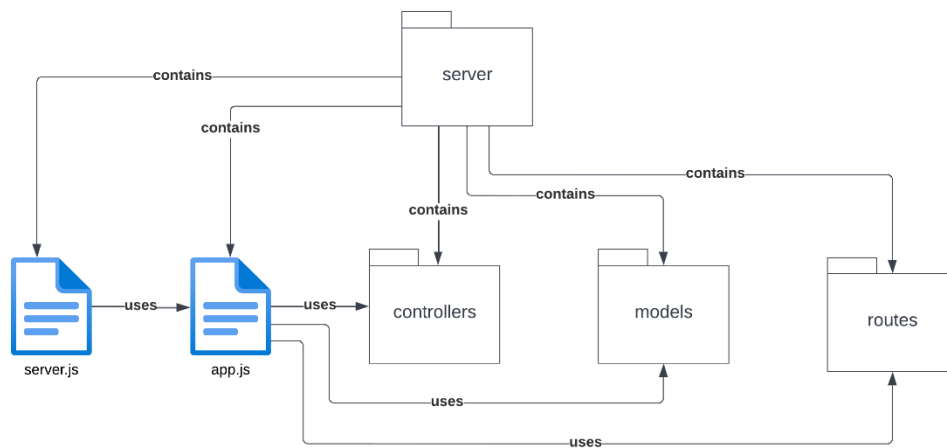
Κεφάλαιο 4. Υλοποίηση

4.1 Δομή του κώδικα

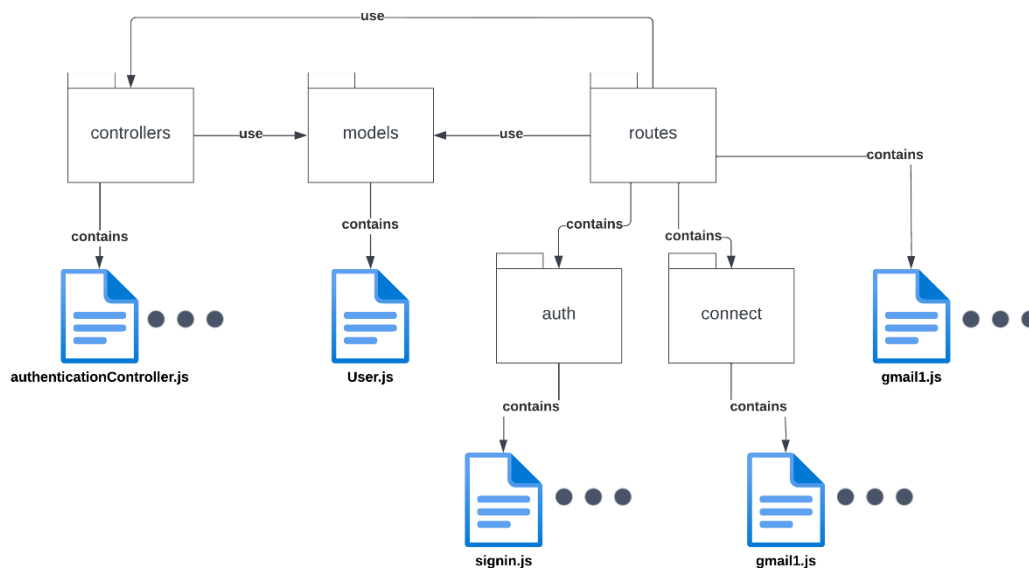
Σε αυτή την υποενότητα θα αναλύσουμε τη δομή που κώδικα της εφαρμογής. Τα βασικά υποσυστήματα της εφαρμογής θα είναι δύο, ο πελάτης, και ο διακομιστής. Ο πελάτης είναι το frontend, ενώ ο διακομιστής είναι ο διακομιστής εφαρμογής του backend. Ο κώδικας του πελάτη θα βρίσκεται στο φάκελο client, ενώ ο κώδικας του διακομιστή θα βρίσκεται στο φάκελο server.

4.1.1 Ο φάκελος server

Ο φάκελος server θα αποτελείται από το αρχείο server.js που θα περιέχει τον κώδικα για την εκκίνηση του backend διακομιστή εφαρμογής. Το αρχείο server.js θα χρησιμοποιεί τη λογική του διακομιστή εφαρμογής που θα βρίσκεται στο αρχείο app.js. Ο λόγος που αυτά τα δύο αρχεία δεν είναι ένα, είναι για καλύτερο διαχωρισμό λογικής, που επιτρέπει testing των endpoints του διακομιστή εφαρμογής. Το αρχείο app.js θα χρησιμοποιεί ελεγκτές μοντέλα και δρομολογητές, από τα αρχεία controllers, models, και routes, αντίστοιχα. Οι δρομολογητές θα δρομολογούν εισερχόμενα αιτήματα στις κατάλληλες συναρτήσεις του κατάλληλου ελεγκτή, ενώ οι ελεγκτές θα χρησιμοποιούν τα μοντέλα για ανάκτηση και αποθήκευση δεδομένων.



Σχήμα 4.1 Η δομή του φακέλου server.

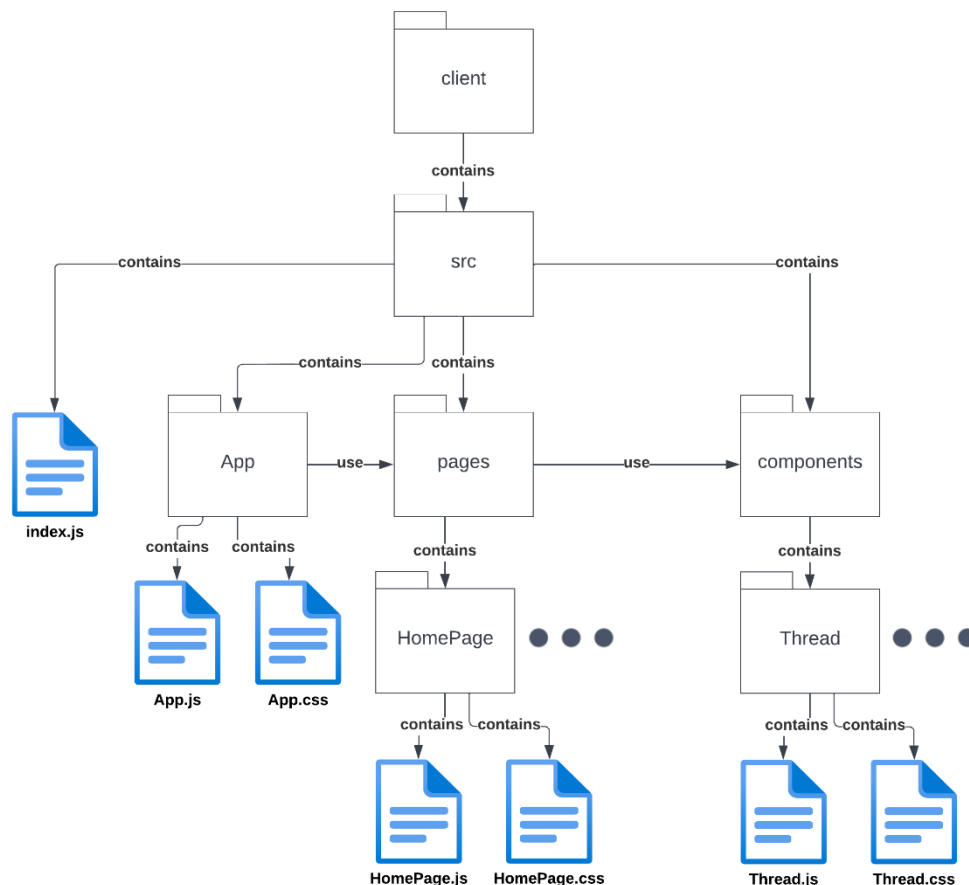


Σχήμα 4.2 Η δομή των συστατικών του φακέλου server.

4.1.2 Ο φάκελος client

Ο φάκελος client θα αποτελείται από το φάκελο src ο οποίος θα έχει τον κώδικα του πελάτη. Ο φάκελος src, με τη σειρά του, θα περιέχει το αρχείο index.js που θα δημιουργεί τη React εφαρμογή μονής σελίδας, καθώς και τους φακέλους App, pages, και components. Ο φάκελος App θα είναι το βασικό component της εφαρμογής, και θα χρησιμοποιείται από το αρχείο index.js. Κάθε σελίδα θα είναι subcomponent του App

component. Ο φάκελος App, όπως και όλες οι σελίδες και τα components, θα αποτελείται από δύο αρχεία. Το αρχείο App.js που θα περιέχει τον κώδικα του component, καθώς και το αρχείο App.css, που θα περιέχει τον κώδικα css του component. Ο κώδικας κάθε component θα είναι με αυτή τη μορφή. Ο φάκελος pages θα περιέχει τις σελίδες του frontend που θα είναι και αυτές components, ενώ θα αποτελούνται και από άλλα components, τα οποία θα βρίσκονται στο φάκελο components.

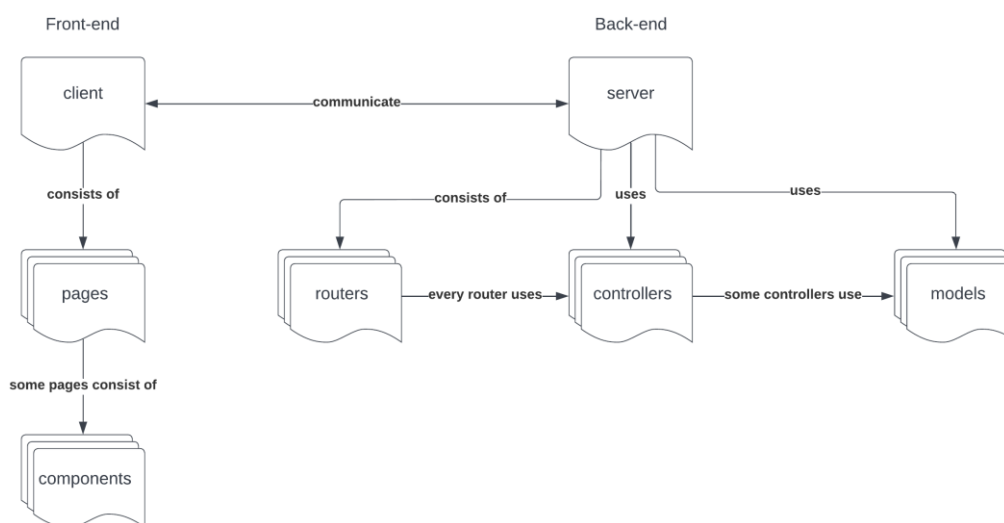


Σχήμα 4.3 Η δομή του φάκελου client.

4.2 Τα συστατικά της εφαρμογής

Η εφαρμογή μας θα αποτελείται από 2 βασικές οντότητες που θα αλληλεπιδρούν μεταξύ τους, τον πελάτη και τον διακομιστή. Ο πελάτης θα είναι το frontend της εφαρμογής, και ο διακομιστής το backend. Ο πελάτης θα λαμβάνει αιτήματα από το χρήστη, και θα επικοινωνεί με το διακομιστή, για να τα ικανοποιήσει. Πιο συγκεκριμένα, οι σελίδες και τα components θα στέλνουν HTTP αιτήματα στο backend, και οι

δρομολογητές θα τα δρομολογούν στις κατάλληλες συναρτήσεις των ελεγκτών. Οι ελεγκτές, με τη σειρά τους, θα χρησιμοποιούν, μεταξύ άλλων, τα μοντέλα (του χρήστη συγκεκριμένα), για να ικανοποιήσουν τα αιτήματα και θα αποκρίνονται στα components, τα οποία θα ανανεώνουν την κατάστασή τους, για να ενημερώσουν το χρήστη. Ο κώδικας τους θα βρίσκεται στους φακέλους client, και server, της εφαρμογής.



Σχήμα 4.4 Τα συστατικά της εφαρμογής. Αριστερά έχουμε τα συστατικά του front-end, ενώ δεξιά τα συστατικά του back-end, καθώς και το βελάκι στη μέση που δείχνει την αμφίδρομη επικοινωνία τους μέσω του REST API του back-end.

4.2.1 Συστατικά του διακομιστή

Ο διακομιστής της εφαρμογής θα αποτελείται από ελεγκτές, μοντέλα, και δρομολογητές. Τη στιγμή που θα φτάνει κάποιο εισερχόμενο αίτημα στο διακομιστή, ο κατάλληλος δρομολογητής (αναλόγως το μονοπάτι στο οποίο έγινε το αίτημα) θα αναλαμβάνει να το δρομολογήσει στην κατάλληλη συνάρτηση του ελεγκτή που θα είναι υπεύθυνος για τον πόρο τον οποίο αφορά το αίτημα. Εάν η λειτουργία που πρέπει να εκτελεστεί είναι πολύ απλή, θα μπορεί να δρομολογήσει το αίτημα σε μια απλή callback συνάρτηση που θα ορίζεται στο ίδιο αρχείο που ορίζεται ο δρομολογητής. Για αποθήκευση και ανάκτηση δεδομένων, προς, και από, τη βάση, θα χρησιμοποιούνται τα μοντέλα από τους ελεγκτές που θα θέλουν να τη χρησιμοποιήσουν.

Παρακάτω φαίνονται οι συσχετίσεις των συστατικών του backend σε 3 πίνακες, ένα πίνακα για κάθε κατηγορία. Κάθε πίνακας περιέχει 4 στήλες, την κατηγορία του

συστατικού, μια περιγραφή για το τι αφορά, τι βιβλιοθήκες και άλλα συστατικά χρησιμοποιεί, και από ποια συστατικά χρησιμοποιείται. Οι βιβλιοθήκες είναι χαρακτηρισμένες με το (lib), ενώ app είναι το αρχείο με τον κώδικα για τις ρυθμίσεις του διακομιστή. Για τις βιβλιοθήκες αναφερόμαστε αναλυτικά στην υποενότητα 4.3.

Μοντέλο	Περιγραφή	Χρησιμοποιεί	Χρησιμοποιείται από
User	Ανάκτηση και αποθήκευση δεδομένων	mongoose (lib)	app, authenticationController, authorizationController, signupController

Πίνακας 4.1 Συσχετίσεις των μοντέλων του backend.

Ελεγκτής	Περιγραφή	Χρησιμοποιεί	Χρησιμοποιείται από
authenticationController	Αυθεντικοποίηση του χρήστη	bcrypt (lib), User	app, signinRouter, connectGmail1Router, connectGmail2Router connectMsTeamsRouter, gmail1Router, gmail2Router, msTeamsRouter
authorizationController	Εξουσιοδότηση της εφαρμογής για πρόσβαση στους λογαριασμούς του χρήστη	User, encryptionController	app
encryptionController	Κρυπτογράφηση και αποκρυπτογράφηση δεδομένων	crypto (lib)	gmailController, msTeamsController
gmailController	Διαχείριση των Gmail λογαριασμών του χρήστη	axios (lib), nodemailer (lib), parseMessage (lib),	gmail1Router, gmail2Router

		replyParser (lib), encryptionController, async-mutex (lib)	
msTeamsController	Διαχείριση του MsTeams λογαριασμού του χρήστη	axios (lib), qs (lib), encryptionController	msTeamsRouter
signupController	Εγγραφή καινούριου χρήστη	bcrypt (lib), User	signupRouter

Πίνακας 4.2 Συσχετίσεις των ελεγκτών του backend.

Δρομολογητής	Περιγραφή	Χρησιμοποιεί	Χρησιμοποιείται από
authenticatedRouter	Απάντηση εάν ο χρήστης έχει ενεργή συνεδρία στο backend		app
signinRouter	Σύνδεση χρήστη	passport (lib), authenticationController	app
signoutRouter	Αποσύνδεση χρήστη		app
signupRouter	Εγγραφή χρήστη	signupController	app
connectGmail1Router, connectGmail2Router, connectMsTeamsRouter	Εξουσιοδότηση εφαρμογής για τους λογαριασμούς του χρήστη	passport (lib), authenticationController	app
gmail1Router, gmail2Router	Λειτουργίες στους Gmail λογαριασμούς του χρήστη	authenticationController, gmailController	app

msTeamsRouter	Λειτουργίες στο MsTeams λογαριασμό του χρήστη	authenticationController, msTeamsController	app
---------------	---	---	-----

Πίνακας 4.3 Συσχετίσεις των δρομολογητών του backend.

Στη συνέχεια θα αναλύσουμε κάθε ένα από αυτά τα συστατικά που αναφέραμε, ανά κατηγορία.

1. Μοντέλα:

1.1. **User** – Το μοντέλο του χρήστη, για αποθήκευση και ανάκτηση δεδομένων στη συλλογή users στη βάση (ανάλυση για το “σχήμα” της βάσης βρίσκεται στην υποενότητα 4.4).

2. Ελεγκτές:

2.1. **authenticationController** – Ελεγκτής σχετικός με την ταυτοποίηση του χρήστη. Οι συναρτήσεις που παρέχει στους δρομολογητές είναι οι εξής:

2.1.1. **userIsAuthenticated()** – Συνάρτηση που επιστρέφει true εάν ο χρήστης που κάνει το αίτημα έχει ενεργή συνεδρία στο διακομιστή, ή false, σε αντίθετη περίπτωση. Χρησιμοποιείται για να αποτρέψει την πρόσβαση ενός μη συνδεδεμένου χρήστη σε προστατευόμενα μονοπάτια.

2.1.2. **authenticateUser()** – Συνάρτηση που χρησιμοποιείται σε συνδυασμό με τη βιβλιοθήκη Passport.js για να αυθεντικοποιήσει ένα χρήστη.

2.2. **signupController** – Ελεγκτής σχετικός με την εγγραφή ενός καινούριου χρήστη. Η συνάρτηση που παρέχει για τον κατάλληλο δρομολογητή είναι η **registerUser()**, η οποία δημιουργεί καινούριο λογαριασμό για το χρήστη, αποθηκεύει τα στοιχεία του στη βάση, δημιουργεί καινούρια συνεδρία στο διακομιστή, και επιστρέφει μήνυμα επιτυχίας ή αποτυχίας.

2.3. **authorizationController** – Ελεγκτής σχετικός με την εξουσιοδότηση της εφαρμογής να διαχειρίζεται τους λογαριασμούς του χρήστη, εκ μέρους του. Οι συναρτήσεις που παρέχει είναι οι **authorizeAppForGmail1()**, **authorizeAppForGmail2()**, και **authorizeAppForMsTeams()**, και

χρησιμοποιούνται από τη βιβλιοθήκη Passport.js για να αποθηκεύσουν τα access, και refresh, tokens, καθώς και στοιχεία για το προφίλ του χρήστη, που θα επιστραφούν από το διακομιστή εξουσιοδότησης, κατά την υλοποίηση του πρωτοκόλλου OAuth 2.0. Πριν αποθηκεύσουν τα tokens τα κρυπτογραφούν.

2.4. **encryptionController** – Ελεγκτής σχετικός με την κρυπτογράφηση ευαίσθητων στοιχείων. Οι συναρτήσεις που παρέχει είναι οι **encrypt()**, και **decrypt()**, και χρησιμοποιούνται για κρυπτογράφηση των tokens πριν αποθηκευτούν στη βάση, και αποκρυπτογράφηση τους όταν ανακτηθούν από αυτή, και πριν σταλούν ως μέρος κάποιου αιτήματος στα διαδικτυακά APIs των παρόχων, αντίστοιχα.

2.5. **gmailController** – Ελεγκτής σχετικός με τη διαχείριση των Gmail λογαριασμών του χρήστη. Οι συναρτήσεις που παρέχει για τον κατάλληλο δρομολογητή είναι οι εξής:

2.5.1. **getGmailInbox()** – Συνάρτηση η οποία ανακτά τα εισερχόμενα και τα μηνύματα που έχουν χαρακτηριστεί ως σημαντικά, οργανωμένα σε νήματα, και τα επιστρέφει ως απόκριση.

2.5.2. **modifyThread()** – Συνάρτηση η οποία επισημαίνει ένα νήμα ως αδιάβαστο, ή σημαντικό, αναλόγως το φορτίο του αιτήματος, και αποκρίνεται με μήνυμα επιτυχίας ή σφάλματος.

2.5.3. **getMessageAttachment()** – Συνάρτηση η οποία ανακτά ένα συνημμένο που περιεχόταν σε κάποιο μήνυμα του Gmail λογαριασμού του χρήστη, και το επιστρέφει ως απόκριση.

2.5.4. **getUnreadEmailsFromGmail()** – Συνάρτηση η οποία ανακτά όλα τα αδιάβαστα μηνύματα του Gmail λογαριασμού του χρήστη, και τα επιστρέφει.

2.5.5. **sendEmail()** – Συνάρτηση η οποία στέλνει ένα καινούριο μήνυμα (το οποίο περιέχεται στο ωφέλιμο φορτίο του αιτήματος), εκ μέρους του χρήστη, και αποκρίνεται με μήνυμα επιτυχίας, ή σφάλματος.

2.5.6. **markMessageAsRead()** – Συνάρτηση η οποία επισημαίνει ένα μήνυμα ως αδιάβαστο, και αποκρίνεται με μήνυμα επιτυχίας, ή σφάλματος.

2.6. **msTeamsController** – Ελεγκτής σχετικός με τη διαχείριση του Microsoft Teams λογαριασμού του χρήστη. Οι συναρτήσεις που παρέχει για τον κατάλληλο δρομολογητή είναι οι εξής:

2.6.1. **getChats()** – Συνάρτηση η οποία ανακτά όλα τα μηνύματα από τις συνομιλίες του χρήστη, οργανωμένα ανά chat, και τα επιστρέφει ως απόκριση.

2.6.2. **sendChatMessage()** – Συνάρτηση η οποία κάνει αποστολή του μηνύματος που ορίζεται στο ωφέλιμο φορτίο του αιτήματος. Αποκρίνεται με μήνυμα επιτυχίας ή σφάλματος.

2.6.3. **uploadFileToOneDrive()** – Συνάρτηση η οποία ανεβάζει το συνημμένο αρχείο, που ορίζεται στο ωφέλιμο φορτίο του αιτήματος, στο OneDrive του χρήστη, και δίνει δικαιώματα ανάγνωσης στο λογαριασμό για τον οποίο προορίζεται το συνημμένο. Αποκρίνεται με μήνυμα επιτυχίας ή σφάλματος.

2.6.4. **createNewChat()** – Συνάρτηση η οποία δημιουργεί ένα καινούριο chat μεταξύ του χρήστη και ενός άλλου χρήστη που ορίζεται στο ωφέλιμο φορτίο του αιτήματος. Αποκρίνεται με μήνυμα επιτυχίας ή σφάλματος.

3. Δρομολογητές:

3.1. **authenticatedRouter** – Δρομολογεί τα GET αιτήματα που έρχονται στο μονοπάτι `"/api/auth/authenticated"` του διακομιστή, σε μία callback συνάρτηση που επιστρέφει τη μεταβλητή `authenticated`, αναλόγως εάν ο χρήστης είναι συνδεδεμένος (έχει ενεργή συνεδρία στο διακομιστή), ή όχι.

3.2. **signinRouter** – Δρομολογεί τα POST αιτήματα που έρχονται στο μονοπάτι `"/api/auth/signin"` του διακομιστή, στη συνάρτηση `authenticate()` της βιβλιοθήκης `Passport.js`, η οποία χρησιμοποιεί τη συνάρτηση `authenticateUser()` του `authenticationController` για να αυθεντικοποιήσει το χρήστη, με βάση το όνομα χρήστη και το συνθηματικό που έχει στείλει με το αίτημά του το frontend. Η συνάρτηση, με τη σειρά της, κάνει ανακατεύθυνση του αιτήματος στο μονοπάτι `"/api/auth/signin/success"`, ή στο μονοπάτι `"/api/auth/signin/failure"`, αναλόγως την επιτυχία, ή αποτυχία αυθεντικοποίησης του χρήστη. Στα 2 αυτά μονοπάτια υπάρχουν callback συναρτήσεις που αποκρίνονται κατάλληλα εάν ο χρήστης ταυτοποιήθηκε επιτυχώς, και δημιουργήθηκε συνεδρία στο διακομιστή, ή όχι.

3.3. **signoutRouter** - Δρομολογεί τα GET αιτήματα που έρχονται στο μονοπάτι `"/api/auth/signout"` του διακομιστή, σε μία callback συνάρτηση που τερματίζει τη συνεδρία του χρήστη στο διακομιστή και τον ανακατευθύνει στη Login σελίδα του frontend.

3.4. **signupRouter** - Δρομολογεί τα POST αιτήματα που έρχονται στο μονοπάτι `"/api/auth/signup"` του διακομιστή, στη συνάρτηση `registerUser()` του `signupController`.

3.5. **connectGmail1Router, connectGmail2Router, connectMsTeamsRouter** - Δρομολογούν τα GET αιτήματα που έρχονται στο μονοπάτι `"/api/connect/{account}/"` (όπου `account` είναι ίσο με 1 από τα strings `"gmail1"`, `"gmail2"`, `"msTeams"`, αναλόγως το λογαριασμό του χρήστη για τον οποίο απευθύνεται το αίτημα) του διακομιστή, στη συνάρτηση `authorize()` της βιβλιοθήκης `Passport.js`, η οποία ανακατευθύνει το χρήστη στον διακομιστή εξουσιοδότησης του παρόχου, ώστε να αυθεντικοποιηθεί, και να εξουσιοδοτήσει την εφαρμογή να διαχειρίζεται το λογαριασμό του, εκ μέρους του. Ο διακομιστής εξουσιοδότησης στη συνέχεια θα ανακατευθύνει το χρήστη στο μονοπάτι `"/api/connect/{account}/callback"`, όπου θα υπάρχει μία callback συνάρτηση που θα αποθηκεύσει τα `access`, και `refresh`, `tokens`, για πρόσβαση της εφαρμογής στο λογαριασμό του χρήστη, καθώς και δεδομένα από το προφίλ του χρήστη που θα επιστρέψει ο διακομιστής εξουσιοδότησης, στη βάση. Έπειτα θα ανακατευθύνει το χρήστη στη Home σελίδα του frontend.

3.6. **gmail1Router, gmail2Router** - Δρομολογούν τις αιτήσεις που έρχονται σε κάποιο endpoint των πόρων `gmail1`, `gmail2`, αντίστοιχα. Για κάθε έναν από τους 2 πόρους τα endpoints είναι 6, όπως αναφέραμε και στην προηγούμενη υποενότητα (όπου `gmailAccount` κάποιο από τα strings `"gmail1"`, `"gmail2"`):

3.6.1. **"/api/{gmailAccount}/threads"** - Οι δρομολογητές δρομολογούν τα GET αιτήματα που έρχονται σε αυτό το μονοπάτι στη συνάρτηση `getGmailInbox()` του `gmailController`.

3.6.2.

"/api/{gmailAccount}/messages/:messageId/attachments/:attachmentId" - Οι δρομολογητές δρομολογούν τα GET αιτήματα που έρχονται σε

αυτό το μονοπάτι στη συνάρτηση `getMessageAttachment()` του `gmailController`.

3.6.3. **“/api/{gmailAccount}/threads/:threadId/modify”** - Οι δρομολογητές δρομολογούν τα PATCH αιτήματα που έρχονται σε αυτό το μονοπάτι στη συνάρτηση `modifyThread()` του `gmailController`.

3.6.4. **“/api/{gmailAccount}/messages”** (GET αιτήματα) - Οι δρομολογητές δρομολογούν τα GET αιτήματα που έρχονται σε αυτό το μονοπάτι στη συνάρτηση `getUnreadEmailsFromGmail()` του `gmailController`. Αυτό το endpoint αντικαταστάθηκε από το endpoint με τα νήματα, και σταμάτησε να χρησιμοποιείται από το frontend στην τελική υλοποίηση.

3.6.5. **“/api/{gmailAccount}/messages”** (POST αιτήματα) - Οι δρομολογητές δρομολογούν τα POST αιτήματα που έρχονται σε αυτό το μονοπάτι στη συνάρτηση `sendEmail()` του `gmailController`.

3.6.6. **“/api/{gmailAccount}/messages/:messageId/markAsRead”** - Οι δρομολογητές δρομολογούν τα PATCH αιτήματα που έρχονται σε αυτό το μονοπάτι στη συνάρτηση `markMessageAsRead()` του `gmailController`. Αυτό το endpoint αντικαταστάθηκε από το `modify` endpoint, και σταμάτησε να χρησιμοποιείται από το frontend στην τελική υλοποίηση.

3.7. **msTeamsRouter** - Δρομολογεί τις αιτήσεις που έρχονται σε κάποιο endpoint του πόρου `msTeams`. Για τον πόρο αυτό, τα endpoints είναι 4, όπως αναφέραμε και στην προηγούμενη υποενότητα:

3.7.1. **“/api/msTeams/chats”** (GET αιτήματα) – Ο δρομολογητής δρομολογεί τα GET αιτήματα που έρχονται σε αυτό το μονοπάτι στη συνάρτηση `getChats()` του `msTeamsController`.

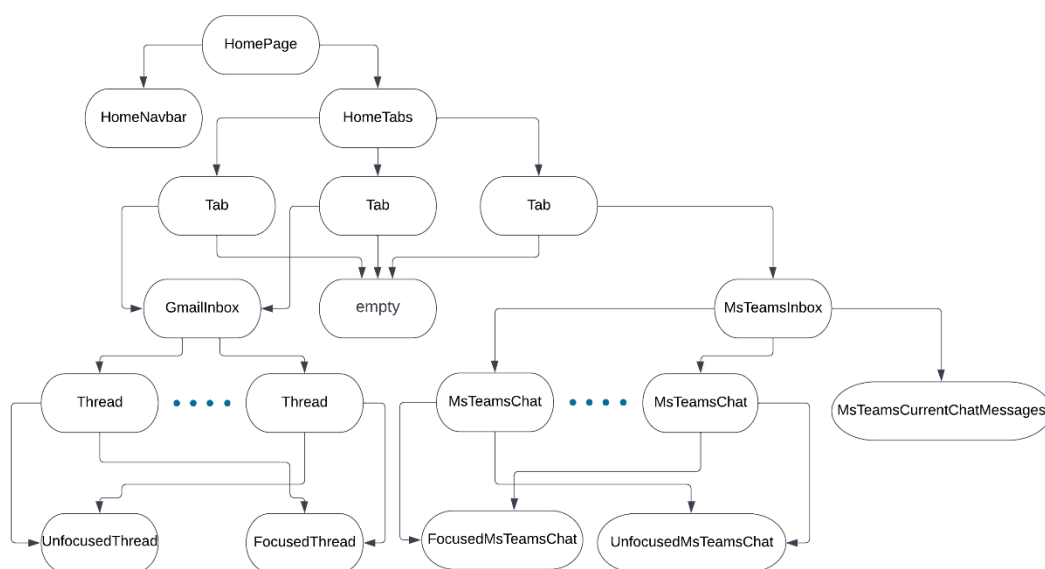
3.7.2. **“/api/msTeams/chats”** (POST αιτήματα) – Ο δρομολογητής δρομολογεί τα POST αιτήματα που έρχονται σε αυτό το μονοπάτι στη συνάρτηση `createNewChat()` του `msTeamsController`.

3.7.3. **“/api/msTeams/chats/:chatId”** - Ο δρομολογητής δρομολογεί τα POST αιτήματα που έρχονται σε αυτό το μονοπάτι στη συνάρτηση `sendChatMessage()` του `msTeamsController`.

3.7.4. **“/api/msTeams/drive”** - Ο δρομολογητής δρομολογεί τα POST αιτήματα που έρχονται σε αυτό το μονοπάτι στη συνάρτηση `uploadFileToOneDrive()` του `msTeamsController`.

4.2.2 Συστατικά του πελάτη

Ο πελάτης της εφαρμογής θα αποτελείται από 2 βασικές σελίδες, τη **Home** σελίδα, και τη **Login** σελίδα. Η **Home** σελίδα θα αποτελείται από μικρότερα React components, συγκεκριμένα από το **HomeNavbar**, και το **HomeTabs** component. Το **HomeTabs** component θα αποτελείται από 3 **Tab** components, ένα **Tab** για κάθε λογαριασμό του χρήστη. Τα **Tab** components μπορεί να περιέχουν είτε ένα **GmailInbox** component, είτε ένα **MsTeamsInbox** component, ή να είναι κενά (εάν ο χρήστης δεν έχει συνδέσει κάποιο λογαριασμό ακόμα). Το **GmailInbox** component θα περιέχει μία λίστα από **Thread** components τα οποία θα αποτελούν τα Gmail νήματα του λογαριασμού του χρήστη, ενώ κάθε **Thread** component θα αποτελείται από ένα **UnfocusedThread**, ή ένα **FocusedThread** component, αναλόγως εάν ο χρήστης το έχει ανοίξει για να διαβάσει τα μηνύματα, ή να στείλει καινούριο μήνυμα στο νήμα αυτό, ή όχι. Το **MsTeamsInbox** component θα περιέχει μία λίστα από **MsTeamsChat** components, τα οποία θα είναι τα chats του χρήστη, καθώς και ένα **MsTeamsCurrentChatMessages** component το οποίο θα εμφανίζει όλα τα μηνύματα του επιλεγμένου chat από τη λίστα, καθώς και φόρμα ώστε ο χρήστης να μπορεί να στείλει καινούριο μήνυμα. Κάθε **MsTeamsChat** θα περιέχει ένα **FocusedMsTeamsChat**, ή ένα **UnfocusedMsTeamsChat**, αναλόγως εάν το chat είναι επιλεγμένο, για προβολή των μηνυμάτων του, ή όχι.



Σχήμα 4.5 Γραφική αναπαράσταση του *HomePage* component. Κάθε component αποτελείται από τη σειρά του από κανένα, ένα, ή περισσότερα άλλα components.

4.3 Οι βιβλιοθήκες που θα χρησιμοποιήσουμε

Σε αυτή την υποενότητα θα αναφέρουμε τις βιβλιοθήκες που θα χρησιμοποιήσουμε, τόσο στο frontend, όσο και στο backend.

4.3.1 Βιβλιοθήκες που θα χρησιμοποιήσουμε στο backend

Οι βιβλιοθήκες που θα χρησιμοποιήσουμε στο backend της εφαρμογής είναι οι εξής:

async-mutex	Χρησιμοποιείται από τον gmailController για την αποφυγή race condition κατά την ανανέωση access token. Προσφέρει ένα mutex αντικείμενο που έχει τη μέθοδο runExclusive η οποία επιτρέπει την ταυτόχρονη πρόσβαση στην κρίσιμη περιοχή, από μόνο ένα νήμα κάθε φορά.
axios	HTTP client βιβλιοθήκη που χρησιμοποιείται από τον gmailController και τον msTeamsController για κλήσεις στα διαδικτυακά APIs των παρόχων.
bcrypt	Χρησιμοποιείται από το signupController για κατακερματισμό του συνθηματικού του χρήστη με τη συνάρτηση hash, καθώς και από τον authenticationController, για σύγκριση του κατακερματισμένου κωδικού του χρήστη με το hash που δημιουργήθηκε από αυτόν που έδωσε για να συνδεθεί στην εφαρμογή, με τη συνάρτηση compare.
compression	Χρησιμοποιείται από το διακομιστή εφαρμογής για συμπίεση των αποκρίσεων του.
cors	Χρησιμοποιείται από το διακομιστή εφαρμογής για να επιτρέπεται στο frontend να κάνει κλήσεις και να παίρνει επιτυχώς αποκρίσεις. Αυτή η βιβλιοθήκη χρειαζόταν κυρίως για την ανάπτυξη της εφαρμογής, όπου το frontend και το backend έτρεχαν σε διαφορετικές θύρες στο localhost
dotenv	Χρησιμοποιείται από το διακομιστή εφαρμογής για τη δυνατότητα διαβάσματος μεταβλητών περιβάλλοντος.
gmail-api-parse-message	Χρησιμοποιείται από τον gmailController για parsing του

	μηνύματος από απόκριση του Gmail API, με τη συνάρτηση <code>parseMessage</code> .
mongoose	Object document mapper που χρησιμοποιείται από το μοντέλο του χρήστη για ανάκτηση και αποθήκευση δεδομένων.
node-email-reply-parser	Χρησιμοποιείται από τον <code>gmailController</code> για εξαγωγή του ορατού κειμένου (χωρίς περιττές πληροφορίες), από κάποιο μήνυμα ηλεκτρονικού ταχυδρομείου, με τη συνάρτηση <code>getVisibleText</code> .
nodemailer	Χρησιμοποιείται από τον <code>gmailController</code> για αποστολή μηνύματος ηλεκτρονικού ταχυδρομείου, μέσω του πρωτοκόλλου SMTP.
passport	Χρησιμοποιείται από το διακομιστή εφαρμογής για αυθεντικοποίηση του χρήστη, καθώς και εξουσιοδότηση της εφαρμογής για πρόσβαση στους λογαριασμούς του χρήστη. Λειτουργεί σε συνδυασμό με τις βιβλιοθήκες <code>passport-local</code> , <code>passport-google-oauth20</code> , <code>passport-microsoft</code> .
passport-local	Χρησιμοποιείται, σε συνδυασμό με τη βιβλιοθήκη <code>passport</code> , για αυθεντικοποίηση κάποιου χρήστη, ενώ εμείς δίνουμε τον τρόπο αυθεντικοποίησης.
passport-google-oauth20	Χρησιμοποιείται από το διακομιστή εφαρμογής, σε συνδυασμό με τη βιβλιοθήκη <code>passport</code> , για εξουσιοδότηση της εφαρμογής για πρόσβαση στους Gmail λογαριασμούς του χρήστη.
passport-microsoft	Χρησιμοποιείται από το διακομιστή εφαρμογής, σε συνδυασμό με τη βιβλιοθήκη <code>passport</code> , για εξουσιοδότηση της εφαρμογής για πρόσβαση στο Microsoft Teams του χρήστη.
express-session	Χρησιμοποιείται από το διακομιστή εφαρμογής, σε συνδυασμό με τη βιβλιοθήκη <code>passport</code> , για δημιουργία συνεδρίας για κάποιο χρήστη.
qs	Χρησιμοποιείται από το <code>msTeamsController</code> για δημιουργία query string.

4.3.2 Βιβλιοθήκες που θα χρησιμοποιήσουμε στο backend

Οι βιβλιοθήκες που θα χρησιμοποιήσουμε στο frontend της εφαρμογής είναι οι εξής:

axios	Η ίδια HTTP client βιβλιοθήκη που χρησιμοποιήσαμε και στο backend για κοινό τρόπο αποστολής HTTP αιτημάτων. Χρησιμοποιείται από τα components App, HomePage, SigninPage, FocusedThread, GmailInbox, MsTeamsInbox, και UnfocusedThread.
dompurify	Βιβλιοθήκη που χρησιμοποιείται από το FocusedThread component για φιλτράρισμα στο html σώμα ενός μηνύματος ηλεκτρονικού ταχυδρομείου, και αποφυγή επιθέσεων cross site scripting.
planer	Χρησιμοποιείται από το FocusedThread component, για αφαίρεση εισαγωγικών απάντησης από μηνύματα ηλεκτρονικού ταχυδρομείου.
react-router-dom	Χρησιμοποιείται από το App component για δυνατότητα προβολής διαφορετικής σελίδας στο root μονοπάτι της εφαρμογής.
react-toastify	Βιβλιοθήκη για προβολή ενημερωτικών μηνυμάτων στο χρήστη κατά την εκτέλεση λειτουργιών, σε μορφή διαδραστικών toasts. Χρησιμοποιείται από τα components App, HomePage, SigninPage, FocusedThread, GmailInbox, MsTeamsInbox, και UnfocusedThread.

Πίνακας 4.5 Οι βιβλιοθήκες που χρησιμοποιήσαμε στο frontend της εφαρμογής.

4.4 Αποστολή αιτήματος στα διαδικτυακά APIs των παρόχων

Για να μπορέσει η εφαρμογή μας να κάνει ένα αίτημα σε κάποιο από τα διαδικτυακά APIs των παρόχων (Gmail, Microsoft Graph), εκ μέρους του χρήστη, θα χρειαστεί, αρχικά, να έχει αποκτήσει access, και refresh tokens, έχοντας λάβει τη συγκατάθεση του χρήστη για πρόσβαση στους λογαριασμούς του, κατά την υλοποίηση της εξουσιοδότησης OAuth 2.0, μεταξύ χρήστη, εφαρμογής, και του διαδικτυακού API του εκάστοτε παρόχου (θα περιγράψουμε πως θα υλοποιηθεί αυτό στην υποενότητα

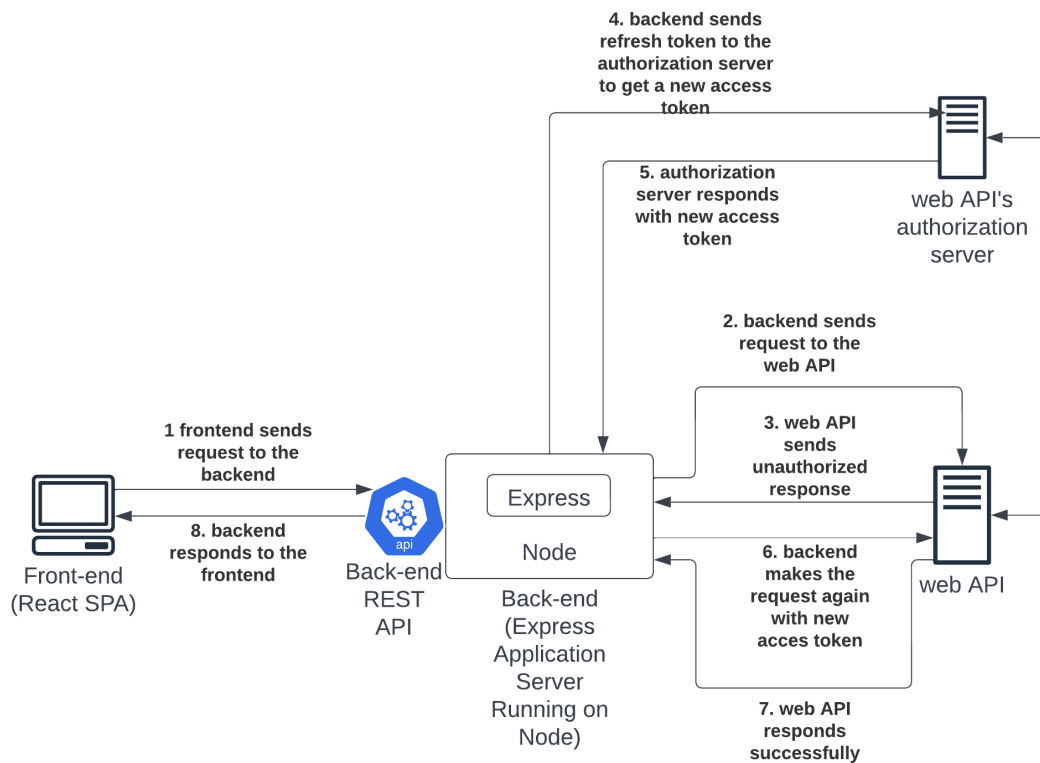
της επικοινωνίας frontend-backend]). Τα access tokens θα αποστέλλονται με τις κεφαλίδες του HTTP αιτήματος, και έτσι το διαδικτυακό API θα αποκρίνεται επιτυχώς. Για αποστολή HTTP αιτήματος θα χρησιμοποιήσουμε τη βιβλιοθήκη **axios**, η οποία είναι φτιαγμένη για να υποστηρίζει αποκλειστικά το πρωτόκολλο HTTP. Έχει δικλείδες ασφαλείας σε συγκεκριμένες επιθέσεις, υποστηρίζει ασύγχρονες HTTP κλήσεις, και γενικότερα προσφέρει πολύ λειτουργικότητα έτοιμη.

4.4.1 Αυτόματη ανανέωση των access tokens

Τα access tokens έχουν μικρή διάρκεια ζωής, και πρέπει να ανανεώνονται τακτικά. Για να μπορεί η ανανέωσή τους να γίνει χωρίς την παρουσία του χρήστη, θα πρέπει να έχουμε ζητήσει πρόσβαση εκτός σύνδεσης (offline access) από το κάθε διαδικτυακό API, ώστε να μας έχουν επιστραφεί refresh tokens, τα οποία θα χρησιμοποιήσουμε για να αποκτήσουμε καινούρια access tokens, χωρίς ο χρήστης να χρειαστεί να ξαναδώσει τη συγκατάθεσή του. Η διαδικασία που θα ακολουθήσουμε, εφόσον έχουμε τα access tokens, για να στείλουμε ένα αίτημα, είναι η εξής:

1. Το frontend/πελάτης της εφαρμογής μας θα λάβει από τη γραφική διεπαφή το αίτημα του χρήστη, και θα στείλει αίτημα στο backend, για υλοποίησή του.
2. Το backend/διακομιστής της εφαρμογής θα λάβει το αίτημα του frontend, και θα αποστείλει, με τη σειρά του, αίτημα στο κατάλληλο διαδικτυακό API, για να υλοποιήσει το αίτημα του frontend. Στο HTTP αίτημα που θα στείλει θα βάλει στις κεφαλίδες το access token που χρειάζεται για να ικανοποιηθεί το αίτημα από το διαδικτυακό API.
3. Εάν το διαδικτυακό API αποκριθεί με σφάλμα μη εξουσιοδοτημένου αιτήματος, σημαίνει ότι το access token έχει λήξει. Το backend της εφαρμογής θα στείλει ένα αίτημα, με το refresh token, σε κάποιο endpoint του διαδικτυακού API για το οποίο προοριζόταν το αίτημα, για ανανέωση του access token.
4. Το διαδικτυακό API θα αποκριθεί με ανανεωμένο access token.
5. Το backend θα αποστείλει, ξανά, το αίτημα που πήρε απόκριση μη εξουσιοδοτημένου αιτήματος. Εάν προκύψει κάποιο πρόβλημα εξουσιοδότησης ξανά, το backend θα αποστείλει κατάλληλη απόκριση στο frontend, ώστε το frontend να ενημερώσει το χρήστη για να ξαναδώσει την εξουσιοδότησή του.
6. Το frontend, λαμβάνοντας επιτυχή απόκριση από το backend θα ενημερώσει το χρήστη για την επιτυχία ή μη του αιτήματός του, και θα

του εμφανίσει τα κατάλληλα δεδομένα, ανανεώνοντας κομμάτια της σελίδας (μιας και το frontend είναι εφαρμογή μονής σελίδας).



Σχήμα 4.6 Τα βήματα της ικανοποίησης ενός αιτήματος του frontend, από το backend, εάν το access token για το λογαριασμό του χρήστη έχει λήξει.

4.5 Αποθήκευση δεδομένων στη βάση

Όπως αναφέραμε στο κεφάλαιο της σχεδίασης, η εφαρμογή μας θα χρειαστεί να αποθηκεύει πληροφορίες για κάθε χρήστη, ώστε να μπορεί να τον ταυτοποιεί, αλλά και να διαχειρίζεται τους λογαριασμούς του. Το γεγονός ότι θα αξιοποιήσουμε τα Gmail, και Microsoft Graph διαδικτυακά APIs μας επιτρέπει να εκμεταλλευτούμε τον αποθηκευτικό χώρο και την υπολογιστική ισχύ των 2 παρόχων (Google και Microsoft), ώστε τα μόνα δεδομένα που θα χρειάζεται να αποθηκεύουμε για κάθε χρήστη να είναι τα tokens (τύπου access, ή refresh) που θα μας επιτρέπουν την διαχείριση για κάθε λογαριασμό του. Για κάθε λογαριασμό ενός χρήστη, θα αποθηκεύουμε, επίσης, τα στοιχεία από το προφίλ του, τα οποία θα μας επιστρέφει ο διακομιστής εξουσιοδότησης του παρόχου, μαζί με τα tokens, κατά την εκτέλεση του πρωτοκόλλου OAuth 2.0 μεταξύ της εφαρμογής, του παρόχου, και του χρήστη, και τα οποία θα μας φανούν χρήσιμα.

Συνεπώς, τα στοιχεία που θα αποθηκεύσουμε για κάθε χρήστη θα είναι τα εξής:

1. id (primary key και index που χρησιμοποιεί η MongoDB για κάθε document)
2. username
3. password
4. Για κάθε έναν από τους 3 λογαριασμούς του (Gmail, Microsoft Teams):
 - i. access token
 - ii. refresh token
 - iii. profile (τα στοιχεία που θα μας επιστρέψει ο authorization server για τον εκάστοτε λογαριασμό του χρήστη)

Για αποθήκευση και ανάκτηση δεδομένων θα χρησιμοποιήσουμε τη βιβλιοθήκη **mongoose**, η οποία προσφέρει μας δίνει τη δυνατότητα για αντιστοίχιση αντικειμένου σε έγγραφο, δηλαδή να δημιουργήσουμε ένα αντικείμενο χρήστη με το οποίο με έτοιμες συναρτήσεις της βιβλιοθήκης θα ανακτούμε και θα αποθηκεύουμε δεδομένα στη βάση. Επίσης, η βιβλιοθήκη mongoose δίνει τη δυνατότητα να ορίσουμε σχήμα για τις συλλογές στη βάση μας.

```
const mongoose = require("mongoose");

const Schema = mongoose.Schema;
const userSchema = new Schema({
  username: {type: String, required: true},
  password: {type: String, required: true},
  googleAccount1: {
    accessToken: {type: String},
    refreshToken: {type: String},
    profile: {type: Object}
  },
  googleAccount2: {
    accessToken: {type: String},
    refreshToken: {type: String},
    profile: {type: Object}
  },
  microsoftAccount: {
    accessToken: {type: String},
    refreshToken: {type: String},
    profile: {type: Object}
  }
}, {minimize: false});
const User = mongoose.model("User", userSchema);

module.exports = User;
```

Σχήμα 4.70 ορισμός του μοντέλου του χρήστη. Στο σχήμα βλέπουμε τον τύπο κάθε πεδίου των εγγράφων της.

```

_id: ObjectId('63472c519de7ccaa9b91e70f')
username: "user2@bestmail.com"
password: "$2b$10$kNVEajSpYRnDdHcU9jd.gep.CzW7zZOaiuN0QB6AEkSXIVfsXNVEC"
> googleAccount1: Object
> googleAccount2: Object
v microsoftAccount: Object
  accessToken: "d8b2e5175cc0c7a61f4ee8a3c9011bf436c7874137450fa39e71183507020e8a8a93f7..."
  refreshToken: "3f9a729f60d72040c3abbe2d0fd22bb5c11170d0045e75f72153588f7412e0801bd46f..."
  v profile: Object
    provider: "microsoft"
    v name: Object
      familyName: null
      givenName: null
      id: "c15a5607-5aee-4941-8a70-45044e320a8c"
      displayName: "KONSTANTINOS PAPADIAS"
    v emails: Array
      v 0: Object
        type: "work"
        value: "cs02614@uoi.gr"
      _raw: '{"@odata.context":"https://graph.microsoft.com/v1.0/$metadata#users/$e..."
    v _json: Object
      @odata.context: "https://graph.microsoft.com/v1.0/$metadata#users/$entity"
      v businessPhones: Array
      displayName: "KONSTANTINOS PAPADIAS"
      givenName: null
      jobTitle: null
      mail: "cs02614@uoi.gr"
      mobilePhone: null
      officeLocation: null
      preferredLanguage: "el-GR"
      surname: null
      userPrincipalName: "cs02614@uoi.gr"
      id: "c15a5607-5aee-4941-8a70-45044e320a8c"
    __v: 0

```

Σχήμα 4.8 Αποθηκευμένο έγγραφο για το χρήστη με όνομα χρήστη “user2@bestmail.com” (λογαριασμός για τεστ της εφαρμογής). Παρατηρούμε το “σχήμα” της συλλογής, όπου κάθε χρήστης έχει ένα όνομα χρήστη, ένα συνθηματικό, καθώς και 3 λογαριασμούς σχετιζόμενους με το λογαριασμό του στην εφαρμογή μας. Στο σχήμα παρατηρούμε και τα δεδομένα που έχουν αποθηκευτεί για το Microsoft Teams λογαριασμό του χρήστη, δηλαδή ένα access, και ένα refresh token, καθώς και πληροφορίες για το προφίλ του χρήστη, ενώ τα δεδομένα για τον 1ο, και το 2ο Gmail λογαριασμό είναι ελαχιστοποιημένα.

4.6 Ασφάλεια

4.6.1 Συνεδρίες διακομιστή

Για να συνδέσουμε ένα χρήστη στην εφαρμογή θα χρησιμοποιήσουμε τη μέθοδο των συνεδριών διακομιστή, όπως αναφέραμε στο κεφάλαιο της σχεδίασης. Τα βήματα που ακολουθούνται, σε αυτή τη μέθοδο, μεταξύ πελάτη και διακομιστή, είναι τα εξής:

1. Ο χρήστης αυθεντικοποιείται από τον διακομιστή εισάγοντας τα διαπιστευτήρια του.

2. Ο διακομιστής δημιουργεί μία συνεδρία, την αποθηκεύει στη βάση του, και στη συνέχεια στέλνει το αναγνωριστικό συνεδρίας στο χρήστη, μέσω του cookie.
3. Ο χρήστης έχει, πλέον, εγκαθιδρύσει μία συνεδρία στο διακομιστή. Μπορεί να κάνει αιτήματα στο διακομιστή, στέλνοντας το cookie (που περιέχει το αναγνωριστικό συνεδρίας), και ο διακομιστής, με τη σειρά του, να ψάχνει για τη συνεδρία στη βάση που την αποθήκευσε, με βάση το αναγνωριστικό συνεδρίας από το cookie. Εάν το αναγνωριστικό συνεδρίας υπάρχει στη βάση με τις συνεδρίες, τότε ο διακομιστής αποκρίνεται επιτυχώς στον χρήστη/πελάτη.

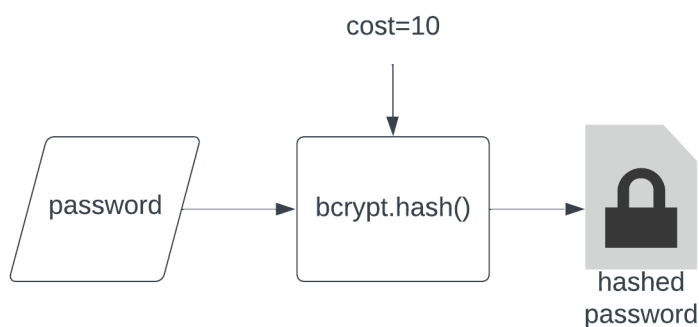
Για την αυθεντικοποίηση του χρήστη, και τη δημιουργία συνεδρίας θα χρησιμοποιήσουμε τη βιβλιοθήκη **Passport.js**, στην οποία θα χρειαστεί να δώσουμε την απαραίτητη συνάρτηση αυθεντικοποίησης για το χρήστη.

4.6.2 Κρυπτογράφηση και κατακερματισμός ευαίσθητων δεδομένων

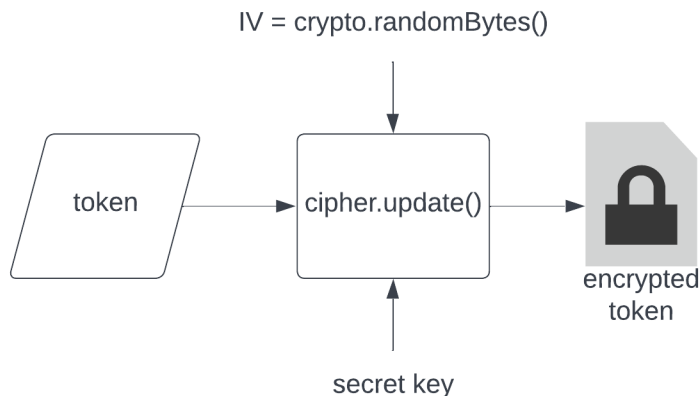
Για να κατακερματίσουμε τον κωδικό του χρήστη, πριν τον αποθηκεύσουμε στη βάση, θα χρησιμοποιήσουμε, όπως αναφέραμε στο κεφάλαιο της σχεδίασης, τη συνάρτηση κατακερματισμού `bcrypt`. Για το λόγο αυτό θα χρησιμοποιήσουμε τη βιβλιοθήκη **bcrypt**. Η συνάρτηση `bcrypt` είναι προσαρμόσιμη και μπορούμε να επιλέξουμε εμείς το υπολογιστικό κόστος δημιουργίας hash εξόδου. Θα περάσουμε κάθε κωδικό από τη συνάρτηση, η οποία θα τρέξει για υπολογιστικό κόστος 10, δηλαδή 2^{10} επαναλήψεις κατακερματισμού. Επιλέγοντας να τρέχουμε τη συνάρτηση για υπολογιστικό κόστος 10 έχουμε έναν αρκετά αργό χρόνο υπολογισμού της εξόδου της συνάρτησης, ώστε να επιτρέπει στους χρήστες της εφαρμογής μας να αλλάξουν τους κωδικούς τους, σε περίπτωση που η βάση εκτεθεί σε επιτιθέμενους, που θα επιχειρήσουν κάποια επίθεση brute force. Επίσης, επιλέγοντας υπολογιστικό κόστος 10 ο χρόνος υπολογισμού της εξόδου είναι αρκετά σύντομος, ώστε ένας χρήστης να μη χρειάζεται να περιμένει πολύ ώρα για να εγγραφεί, ή να συνδεθεί στην εφαρμογή.

Για να κρυπτογραφούμε, και να αποκρυπτογραφούμε, τα `access`, και `refresh`, `tokens`, κάθε φορά που θα τα σώζουμε στη βάση, ή θα τα ανακτάμε από αυτή, αντίστοιχα, θα χρησιμοποιήσουμε τον αλγόριθμο AES-256-CBC. Κάθε είδους token θα κρυπτογραφείται με ένα μυστικό κλειδί, και ένα τυχαίο διάνυσμα αρχικοποίησης, πριν αποθηκευτεί στη βάση, και θα αποκρυπτογραφείται κάθε φορά που θα ανακτάται από αυτή για να σταλεί ως μέρος κάποιου αιτήματος στα διαδικτυακά APIs (Gmail,

Microsoft Graph). Αυτό θα μας επιτρέψει να παρέχουμε ασφάλεια στο χρήστη, ότι οι λογαριασμοί του θα μείνουν απαραβίαστοι, σε περίπτωση παραβίασης της βάσης, αφού ο χρόνος για να αποκρυπτογραφηθούν τα tokens είναι σίγουρα αρκετός, για να δημιουργήσουμε καινούρια και να ακυρώσουμε τα παλιά. Η κρυπτογράφηση θα γίνει με τη χρήση του **crypto** module του Node.js. Το crypto υποστηρίζει πολλές κρυπτογραφικές συναρτήσεις, τη δημιουργία διανυσμάτων αρχικοποίησης, και πολλά ακόμα που σχετίζονται με την κρυπτογραφία και την ασφάλεια μιας εφαρμογής (και προφανώς τον αλγόριθμο AES).



Σχήμα 4.9 Κατακερματισμός του συνθηματικού του χρήστη με τη βιβλιοθήκη *bcrypt*.



Σχήμα 4.10 Κρυπτογράφηση token με χρήση του *crypto* module.

4.7 Βελτίωση απόδοσης του διακομιστή

Για να βελτιώσουμε την απόδοση του backend διακομιστή μας θα χρησιμοποιήσουμε 2 βασικές τεχνικές. Η 1^η είναι η συμπίεση των αποκρίσεων σε αιτήματα, από το διακομιστή. Για αυτό το σκοπό θα χρησιμοποιήσουμε τη βιβλιοθήκη **compression**, επιλέγοντας όλες οι αποκρίσεις να συμπιέζονται. Η 2^η είναι η ασύγχρονη αποστολή αιτημάτων στα διαδικτυακά APIs, όπου αυτό είναι εφικτό. Η βελτίωση απόδοσης, στην περίπτωση που τα αιτήματα που ικανοποιούνται ασύγχρονα είναι

πολλά, μπορεί να είναι σημαντική. Για παράδειγμα η ανάκτηση των μηνυμάτων από κάποιο λογαριασμό του χρήστη, σε οποιοδήποτε από τα 2 APIs, απαιτεί 1 αίτημα για να ανακτηθεί μία λίστα με τα αναγνωριστικά των μηνυμάτων, και μετά, για 1 αίτημα για το σώμα (δηλαδή το περιεχόμενο) κάθε μηνύματος, με βάση το αναγνωριστικό του. Εύκολα μπορούμε να καταλάβουμε πόσο σημαντική βελτίωση απόδοσης έχουμε όταν, έχοντας ανακτήσει την αρχική λίστα με τα αναγνωριστικά των μηνυμάτων κάνουμε ασύγχρονα τα αιτήματα για ανάκτηση κάθε μηνύματος, και δε χρειάζεται να τα κάνουμε σειριακά, κάτι που θα ανάγκαζε το διακομιστή να περιμένει όσο αναμένει την απόκριση κάποιου αιτήματος.

4.8 Σχεδιασμός API του backend διακομιστή εφαρμογής

Όπως αναφέραμε και στο 2^ο κεφάλαιο, το backend της εφαρμογής μας θα αποτελείται από ένα διακομιστή εφαρμογής. Ο διακομιστής εφαρμογής θα προσφέρει ένα API το οποίο θα ακολουθεί τη REST αρχιτεκτονική, με πόρους που θα αντιπροσωπεύουν τα δεδομένα των λογαριασμών του χρήστη. Το API θα διαχειρίζεται, επίσης, την αυθεντικοποίηση του χρήστη, καθώς και την εξουσιοδότηση της εφαρμογής να διαχειρίζεται τα δεδομένα των λογαριασμών του χρήστη. Εκτός από τα μονοπάτια `"/api/auth/authenticated"`, `"/api/auth/signin"`, και `"/api/auth/signup"`, όλα τα υπόλοιπα μονοπάτια είναι προστατευόμενα, δηλαδή εάν ο χρήστης που κάνει αίτηση σε αυτά δεν έχει ενεργή συνεδρία στο backend διακομιστή, θα ανακατευθυνθεί στη Login σελίδα της εφαρμογής.

4.8.1 Endpoints για την αυθεντικοποίηση του χρήστη

Η βασική διαδρομή (root route) του API θα είναι το `"/api"`. Τα endpoints στα οποία θα γίνεται ο έλεγχος εάν ο χρήστης είναι συνδεδεμένος, η εγγραφή, η σύνδεση, καθώς και η αποσύνδεση ενός χρήστη θα βρίσκονται κάτω από το μονοπάτι `"/api/auth"`. Τα αιτήματα στα οποία θα αποκρίνεται το API θα είναι τα εξής:

Για JSON απόκριση η οποία θα ενημερώνει το frontend εάν ο χρήστης έχει αυθεντικοποιηθεί (δηλαδή εάν το αίτημα που γίνεται σε αυτό το μονοπάτι περιέχει το id ενεργής συνεδρίας στο διακομιστή),	<i>GET /api/auth/authenticated</i>
---	------------------------------------

ώστε να του δοθεί πρόσβαση στην Home σελίδα της εφαρμογής (από το frontend)	
Για αυθεντικοποίηση του χρήστη και είτε 1) επιτυχή δημιουργία συνεδρίας διακομιστή σε συνδυασμό με μία JSON απόκριση που θα ενημερώνει για την επιτυχή αυθεντικοποίηση, ή 2) αποτυχία αυθεντικοποίησης σε συνδυασμό με μία JSON απόκριση που θα ενημερώνει για το λόγο της αποτυχίας (π.χ. λανθασμένη είσοδος κωδικού)	<i>POST /api/auth/signin</i> <i>Ωφέλιμο φορτίο (JSON): username, password</i>
Για τερματισμό της συνεδρίας διακομιστή και αποσύνδεση από την εφαρμογή (ανακατεύθυνση στη Login σελίδα)	<i>GET /api/auth/signout</i>
Για εγγραφή ενός καινούριου χρήστη και JSON απόκριση με μήνυμα επιτυχίας, ή του λόγου της αποτυχίας	<i>POST /api/auth/signup</i> <i>Ωφέλιμο φορτίο (JSON): username, password</i>

Πίνακας 4.6 Endpoints για την αυθεντικοποίηση του χρήστη

4.8.2 Endpoints για την υλοποίηση του πρωτοκόλλου OAuth 2.0

Τα endpoints στα οποία θα εκτελείται το πρωτόκολλο OAuth 2.0, μεταξύ εφαρμογής, χρήστη, και παρόχου (δηλαδή Google ή Microsoft), για την εξουσιοδότηση της εφαρμογής από το χρήστη να διαχειρίζεται τους λογαριασμούς του, θα βρίσκονται κάτω από το μονοπάτι “/api/connect”. Ας τα δούμε λίγο πιο αναλυτικά:

Για εξουσιοδότηση της εφαρμογής από το χρήστη να διαχειρίζεται τον 1 ^ο Gmail λογαριασμό του	<i>GET /api/connect/gmail1</i>
Για εξουσιοδότηση της εφαρμογής από το χρήστη να διαχειρίζεται τον 2 ^ο Gmail λογαριασμό του	<i>GET /api/connect/gmail2</i>
Για εξουσιοδότηση της εφαρμογής από το χρήστη να διαχειρίζεται τον Microsoft Teams λογαριασμό του	<i>GET /api/connect/msTeams</i>

Πίνακας 4.7 Endpoints για την υλοποίηση του πρωτοκόλλου OAuth 2.0

4.8.3 Endpoints για την διαχείριση των λογαριασμών του χρήστη

Οι πόροι του API θα αφορούν τα νήματα, τα μηνύματα, και τα συνημμένα από τους Gmail λογαριασμούς του χρήστη, καθώς και τα chats του Microsoft Teams λογαριασμού, και του drive του Microsoft λογαριασμού του χρήστη, αντίστοιχα. Τα βασικά μονοπάτια κάτω από τα οποία θα βρίσκονται τα endpoints είναι 3, τα “/api/gmail1”, “/api/gmail2” (αφορούν τα endpoints για τους 2 Gmail λογαριασμούς του χρήστη), και “/api/msTeams” (αφορούν τα endpoints για το Microsoft Teams λογαριασμό του χρήστη).

Endpoints για τη διαχείριση των Gmail λογαριασμών του χρήστη

Για λειτουργίες στα δεδομένα των 2 Gmail λογαριασμών του χρήστη (όπου {gmailAccount} συμβολίζουμε το Gmail λογαριασμό του χρήστη, δηλαδή “gmail1” για τον 1ο Gmail λογαριασμό, ή “gmail2” για τον 2ο Gmail λογαριασμό):

Για ανάκτηση όλων των αδιάβαστων μηνυμάτων από τον Gmail λογαριασμό του χρήστη	<i>GET /api/{gmailAccount}/messages</i>
Για δημιουργία και αποστολή ενός καινούριου μηνύματος που προέρχεται από τον Gmail λογαριασμό του χρήστη (δηλαδή με αποστολέα το Gmail λογαριασμό του χρήστη)	<i>POST /api/{gmailAccount}/messages</i> <i>Ωφέλιμο φορτίο (JSON): message</i>
Για επισήμανση ενός μηνύματος από τα εισερχόμενα μηνύματα ενός Gmail λογαριασμού του χρήστη, ως διαβασμένο	<i>PATCH</i> <i>/api/{gmailAccount}/messages/:messageId/markAsRead</i> <i>Ωφέλιμο φορτίο (JSON): {}</i>

Πίνακας 4.8 Endpoints για τη διαχείριση των Gmail λογαριασμών του χρήστη

Κατά τη διάρκεια ανάπτυξης της εφαρμογής παρατηρούμε ότι η οργάνωση των μηνυμάτων του χρήστη είναι καλύτερη με την χρήση των νημάτων που προσφέρει το

Gmail API (άλλωστε στο Gmail τα εισερχόμενα μηνύματα του χρήστη είναι οργανωμένα σε νήματα), αλλά και για λόγους απόδοσης (θα εξηγήσουμε στη συνέχεια) και βελτίωσης της λειτουργικότητας της εφαρμογής (δυνατότητα επισήμανσης νήματος ως αδιάβαστου ή σημαντικού) προσθέτουμε τα ακόλουθα endpoints:

Για ανάκτηση όλων των νημάτων που περιέχουν αδιάβαστα, αλλά και σημαντικά μηνύματα από το Gmail λογαριασμό του χρήστη	<i>GET /api/{gmailAccount}/threads</i>
Για ανάκτηση ενός συνημμένου κάποιου μηνύματος	<i>GET /api/{gmailAccount}/messages/:messageId/attachments/:attachmentId</i>
Για επισήμανση ενός νήματος ως αδιάβαστου ή σημαντικού	<i>PATCH /api/{gmailAccount}/threads/:threadId/modify</i> <i>Ωφέλιμο φορτίο (JSON): ο τύπος της ετικέτας που θέλουμε να προστεθεί, ή να αφαιρεθεί από το νήμα</i>

Πίνακας 4.9 Endpoints για τη διαχείριση των Gmail λογαριασμών του χρήστη (συνέχεια)

Endpoints για τη διαχείριση του Microsoft Teams λογαριασμού του χρήστη

Για λειτουργίες στα δεδομένα του Microsoft Teams λογαριασμού του χρήστη:

Για ανάκτηση όλων των chats (με τα μηνύματά τους), στα οποία ο χρήστης είναι μέλος, μαζί με όλα τα μέλη που συμμετέχουν	<i>GET /api/msTeams/chats</i>
Για δημιουργία ενός καινούριου chat μεταξύ του συνδεδεμένου χρήστη και ενός άλλου χρήστη	<i>POST /api/msTeams/chats</i> <i>Ωφέλιμο φορτίο (JSON): η διεύθυνση email του χρήστη με τον οποίο ο συνδεδεμένος χρήστης θέλει να δημιουργήσει καινούριο chat</i>
Για αποστολή ενός καινούριου μηνύματος σε κάποιο chat, στο οποίο ο χρήστης είναι μέλος	<i>POST /api/msTeams/chats/:chatId</i> <i>Ωφέλιμο φορτίο (JSON): το καινούριο μήνυμα</i>

Για ανέβασμα ενός αρχείου στο OneDrive του χρήστη (κάτι που χρειαζόμαστε ώστε να μπορεί στη συνέχεια ο χρήστης να στείλει συνημμένα σε κάποιο chat, καθώς το απαιτεί το Microsoft Graph API)	<i>POST /api/msTeams/drive</i> <i>Ωφέλιμο φορτίο (binary stream): το αρχείο σε μορφή data url</i>
--	--

Πίνακας 4.10 Endpoints για τη διαχείριση του Microsoft Teams λογαριασμού του χρήστη

4.9 Επικοινωνία frontend-backend

Σε αυτή την υποενότητα θα περιγράψουμε τον τρόπο που το frontend θα επικοινωνεί με το backend, για να πετύχουμε τη λειτουργικότητα που ψάχνουμε. Όλες οι αιτήσεις που κάνει ο πελάτης/frontend γίνονται προς το διακομιστή/backend, ενώ ο διακομιστής κάνει αιτήσεις και αυτός, με τη σειρά του, στα διαδικτυακά APIs (Gmail, Microsoft Graph), για να ικανοποιήσει τα αιτήματα του πελάτη. Έπειτα αποκρίνεται στον πελάτη, και ο πελάτης ενημερώνει το χρήστη.

Σημαντικό είναι, επίσης, ότι στο **Παράρτημα Α** έχουμε πολύ εκτενή αναφορά στις υπηρεσίες των διαδικτυακών APIs των παρόχων (API reference), που θα χρησιμοποιήσουμε. Διαβάζοντάς το, ο αναγνώστης μπορεί να καταλάβει όλες τις λεπτομέρειες για τις υπηρεσίες αυτές, καθώς και πως να τις χρησιμοποιεί, και τι επιστρέφουν.

4.9.1 Εγγραφή ενός καινούριου χρήστη στην εφαρμογή

Για να μπορέσει ένας χρήστης να χρησιμοποιήσει τις υπηρεσίες που μπορεί να του παρέχει η εφαρμογή μας, θα πρέπει, αρχικά, να δημιουργήσει ένα καινούριο λογαριασμό. Αυτό θα μπορεί να το κάνει από την αρχική (Login) σελίδα της εφαρμογής, με την επιλογή “Create a new account”. Με αυτή την επιλογή θα εμφανίζεται μία καινούρια φόρμα στο χρήστη, που θα τον προτρέπει να εισάγει ένα μοναδικό όνομα χρήστη, και ένα συνθηματικό, ενημερώνοντάς τον ότι τα δεδομένα του θα είναι ασφαλή στην εφαρμογή μας. Ο χρήστης, αφού εισάγει σωστά τα στοιχεία του, θα πρέπει να πατήσει το κουμπί “Submit” ώστε να ξεκινήσει η εγγραφή του στην εφαρμογή. Αφού ο χρήστης κάνει το submit, ο πελάτης θα κάνει ένα HTTP POST αίτημα στο backend, στο endpoint “/api/auth/signup”, με JSON φορτίο τα διαπιστευτήρια του χρήστη (όνομα χρήστη, συνθηματικό). Ο διακομιστής, με τη σειρά του, θα ελέγχει ότι το όνομα χρήστη δεν έχει καταχωρηθεί ήδη στη βάση, και στη συνέχεια, το συνθηματικό του χρήστη θα περνάει μέσα από μία συνάρτηση κατακερματισμού (συγκεκριμένα τη bcrypt), ώστε να κρυπτογραφείται, και θα δημιουργείται ένα καινούριο έγγραφο, για τον καινούριο χρήστη, το οποίο θα αποθηκεύεται στη βάση. Το έγγραφο θα έχει τα πεδία username

και password, καθώς και τα πεδία googleAccount1, googleAccount2, microsoftAccount (τα οποία θα περιέχουν αντικείμενα με κενά πεδία). Έπειτα, θα γίνεται σύνδεση του καινούριου χρήστη στην εφαρμογή, και θα επιστρέφεται μήνυμα επιτυχίας, μέσω της HTTP απόκρισης από το διακομιστή/API στο frontend/πελάτη. Τέλος, ο πελάτης αφού λάβει το επιτυχές μήνυμα, μέσω της απόκρισης, θα εμφανίζει στο χρήστη την κύρια σελίδα (Home page) της εφαρμογής, από την οποία θα μπορεί, πλέον, να χρησιμοποιήσει την εφαρμογή.

4.9.2 Έλεγχος πρόσβασης στη Home σελίδα ως συνδεδεμένος χρήστης

Η εφαρμογή μας, για να μπορεί να ελέγξει ποιος έχει πρόσβαση στη Home σελίδα θα ρωτάει, μέσω HTTP αιτήματος, το backend εάν ο χρήστης που στέλνει το αίτημα για πρόσβαση σε αυτή, έχει ενεργή συνεδρία στο backend. Το backend θα αποκρίνεται με τη μεταβλητή authenticated, ως JSON ωφέλιμο φορτίο, και, εάν η μεταβλητή έχει τιμή true (ο χρήστης που κάνει το αίτημα για πρόσβαση στη Home σελίδα έχει ενεργή συνεδρία), το frontend θα εμφανίζει στο χρήστη τη Home σελίδα. Εάν η μεταβλητή έχει τιμή false το frontend θα εμφανίζει στο χρήστη τη Login σελίδα.

4.9.3 Σύνδεση ενός εγγεγραμμένου χρήστη στην εφαρμογή

Ένας χρήστης, ο οποίος έχει κάνει, ήδη, εγγραφή, και έχει δημιουργήσει λογαριασμό στην εφαρμογή μας, για να μπορέσει να έχει πρόσβαση ξανά στην εφαρμογή (αφού αποσυνδεθεί) θα πρέπει να ξανακάνει σύνδεση. Αυτό θα γίνεται από την σελίδα Login, με τρόπο αντίστοιχο με την εγγραφή ενός καινούριου χρήστη. Ο χρήστης θα εισάγει σε μία φόρμα το email και τον κωδικό του, και στη συνέχεια θα πιέζει το κουμπί “Submit”. Αφού ο χρήστης πιέσει το κουμπί, θα γίνεται μία HTTP κλήση, με ωφέλιμο φορτίο τα στοιχεία του χρήστη από τον πελάτη στο διακομιστή, και ο διακομιστής θα αυθεντικοποιεί το χρήστη με βάση το email και τον κωδικό που είχε εισάγει κατά την εγγραφή του. Σε περίπτωση λανθασμένης εισόδου email ή κωδικού ο διακομιστής θα επιστρέφει JSON απόκριση με το μήνυμα λάθους, και ο πελάτης θα ενημερώνει, αναλόγως, το χρήστη. Σε περίπτωση επιτυχημένης αυθεντικοποίησης, ο διακομιστής θα δημιουργεί μία καινούρια συνεδρία για το χρήστη, και θα επιστρέφει μήνυμα επιτυχίας στο JSON φορτίο της HTTP απόκρισης. Ο πελάτης, με τη σειρά του, θα ανακατευθύνει το χρήστη, ξανά, στο βασικό URL, και η εφαρμογή θα εμφανίζει στο χρήστη τη Home σελίδα, καθώς θα βλέπει, μέσω της μεταβλητής authenticated, ότι ο χρήστης έχει ενεργή συνεδρία στο backend.

4.9.4 Σύνδεση Gmail, ή Microsoft Teams λογαριασμού στην εφαρμογή

Αφού ο χρήστης συνδεθεί επιτυχώς στην εφαρμογή, μπορεί να συνδέσει τους λογαριασμούς του, ώστε να τους διαχειρίζεται μέσω της εφαρμογής. Ο χρήστης θα πρέπει να επιλέξει την κατάλληλη επιλογή (“Connect Gmail account #1”, “Connect Gmail account #2”, ή “Connect MsTeams account”), από το navbar της Home σελίδας, και στη συνέχεια θα ξεκινήσει η διαδικασία της εξουσιοδότησης της εφαρμογής, για να διαχειρίζεται το λογαριασμό του χρήστη, μέσω του πρωτοκόλλου OAuth 2.0, μεταξύ του χρήστη, του backend διακομιστή, και του παρόχου της υπηρεσίας/λογαριασμού:

1. Ο πελάτης θα κάνει ένα GET αίτημα στο κατάλληλο path του API `“/api/connect/{account}”` (όπου {account} ένα από τα strings “gmail1”, “gmail2”, ή “msTeams”).
2. Ο διακομιστής, αφού ελέγξει ότι ο χρήστης είναι συνδεδεμένος (αν δεν είναι συνδεδεμένος θα τον ανακατευθύνει στη Login σελίδα της εφαρμογής), θα τον ανακατευθύνει στον διακομιστή εξουσιοδότησης του παρόχου, ώστε να αυθεντικοποιηθεί, και να εξουσιοδοτήσει τον backend διακομιστή να διαχειριστεί το λογαριασμό του.
3. Ο διακομιστής εξουσιοδότησης του παρόχου θα αποκριθεί με κάποιο κωδικό εξουσιοδότησης.
4. Ο backend διακομιστής θα ανταλλάξει τον κωδικό εξουσιοδότησης με ένα access token, ένα refresh token, καθώς και πληροφορίες για το προφίλ του χρήστη.
5. Ο backend διακομιστής θα αποθηκεύσει τα 2 tokens, και τις πληροφορίες για το προφίλ του χρήστη, και στη συνέχεια θα ανακατευθύνει το χρήστη στη Home σελίδα της εφαρμογής.

4.9.5 Ανάκτηση εισερχόμενων μηνυμάτων των λογαριασμών του χρήστη

Αφού ο χρήστης συνδεθεί επιτυχώς στην εφαρμογή, θα του εμφανιστεί η Home σελίδα, από την οποία θα έχει πρόσβαση στους λογαριασμούς του. Η σελίδα κατά την εμφάνισή της θα κάνει 3 HTTP αιτήματα στον διαδικτυακό διακομιστή του backend, κάθε μία από αυτές για να ανακτήσει τα δεδομένα από τα εισερχόμενα μηνύματα κάθε λογαριασμού του χρήστη.

Για να ανακτηθούν τα νήματα που έχουν χαρακτηριστεί ως αδιάβαστα, ή σημαντικά, από κάθε έναν από τους Gmail λογαριασμούς του χρήστη, θα ακολουθηθούν τα εξής βήματα (όπου {gmailAccount} είναι το “gmail1”, ή το “gmail2”, αντίστοιχα):

1. Ο πελάτης θα κάνει GET αίτημα στο endpoint “/api/{gmailAccount} /threads” του API.
2. Ο διακομιστής με τη σειρά του:
 - i) Θα ελέγξει στη βάση εάν ο χρήστης έχει συνδέσει το λογαριασμό του. Εάν δεν το έχει κάνει θα επιστρέψει μία JSON απόκριση που θα ενημερώνει τον πελάτη.
 - ii) Αν ο χρήστης έχει συνδέσει το λογαριασμό του, θα κάνει 2 GET αιτήματα, ένα για να ανακτήσει τα ids των νημάτων που έχουν επισημανθεί ως αδιάβαστα, και ένα για τα ids των νημάτων που έχουν επισημανθεί ως σημαντικά. Τα αιτήματα θα είναι στο μονοπάτι “/users/{userId}/threads” του Gmail API και θα έχουν από μία παράμετρο ερωτήματος, η οποία θα ζητά μόνο τα αδιάβαστα, ή μόνο τα σημαντικά νήματα, αντίστοιχα. Αυτό θα γίνεται με την παράμετρο ερωτήματος “q” και τιμή ένα από τα strings “is:unread” (για τα αδιάβαστα νήματα), “is:starred” (για τα σημαντικά νήματα).
 - iii) Το Gmail API θα αποκριθεί με μία λίστα από νήματα, χωρίς τα μηνύματά τους, για κάθε αίτημα. Ο διακομιστής, για κάθε μία από τις 2 λίστες, θα χρησιμοποιήσει το id κάθε νήματος ώστε να το ανακτήσει, κάνοντας GET αίτημα για κάθε νήμα στο μονοπάτι “/users/{userId}/threads/{id}” του Gmail API, και θα τα βάλει σε 2 λίστες.
 - iv) Αφού ανακτήσει τα αδιάβαστα, και τα σημαντικά νήματα, ο διακομιστής θα στείλει μία JSON απόκριση στον πελάτη, με το λογαριασμό του χρήστη (διεύθυνση email), το όνομά του στο λογαριασμό, μία λίστα με τα αδιάβαστα, και μία λίστα με τα σημαντικά νήματα.
 - v) Αν κάτι πάει στραβά (για παράδειγμα κάποιο πρόβλημα δικτύου, ή ένα ληγμένο refresh token) ο διακομιστής θα αποκριθεί με HTTP error status code 500 και το λόγο της αποτυχίας.
3. Ο πελάτης, αφού λάβει την απόκριση από το διακομιστή, θα ανανεώσει τα αντίστοιχα Gmail tabs της Home σελίδας με τα δεδομένα από τους 2 λογαριασμούς, ταξινομημένα ανά λογαριασμό.

Για να ανακτηθούν τα chats του χρήστη στο Microsoft Teams θα ακολουθηθούν τα εξής βήματα μεταξύ πελάτη-διακομιστή (ή frontend-backend):

1. Ο πελάτης θα κάνει ένα GET αίτημα στο μονοπάτι `"/api/msTeams/chats"` του API.
2. Ο διακομιστής με τη σειρά του:
 - i) Θα ελέγξει, στη βάση, εάν ο χρήστης έχει συνδέσει το Microsoft Teams λογαριασμό του, και εάν δεν τον έχει συνδέσει θα ενημερώσει τον πελάτη.
 - ii) Εάν ο χρήστης έχει συνδέσει το λογαριασμό του, θα κάνει 1 GET αίτημα στο μονοπάτι `"/users/{userId}/chats"` του Microsoft Graph API, με την παράμετρο ερωτήματος `"$expand=members"`, ώστε το API να επιστρέψει και τα μέλη που συμμετέχουν στο chat.
 - iii) Για κάθε ένα από τα chats που θα επιστραφούν, ο διακομιστής θα κάνει από ένα GET αίτημα στο μονοπάτι `"/users/{userId}/chats/{chatId}/messages"`, ώστε να ανακτήσει όλα τα μηνύματα του chat (το API επιστρέφει τα 50 τελευταία μηνύματα).
 - iv) Ο διακομιστής θα επιστρέψει μία JSON απόκριση με το λογαριασμό (διεύθυνση email), το όνομα του χρήστη στο λογαριασμό, καθώς και μία λίστα με όλα τα chats του χρήστη, κάθε ένα με τα μηνυματά του, και τα μέλη του.
 - v) Αν κάτι δεν πάει καλά (για παράδειγμα σφάλμα δικτύου, ή ληγμένο refresh token), ο διακομιστής θα επιστρέψει μία JSON απόκριση με error status code 500 και το λόγο του σφάλματος.
3. Ο πελάτης θα λάβει την απόκριση και θα εμφανίσει τα chats του χρήστη στο MsTeams tab της Home σελίδας.

4.9.6 Ανάκτηση συνημμένου μηνύματος από Gmail λογαριασμό

Για να ανακτηθεί ένα συνημμένο που έχει σταλεί μαζί με ένα μήνυμα (το οποίο ο χρήστης έχει εστιάσει) στο Gmail λογαριασμό του χρήστη, ο χρήστης θα επιλέξει το συνημμένο, και στη συνέχεια:

1. Ο πελάτης θα κάνει ένα GET αίτημα στο μονοπάτι `"/api/{gmailAccount}/messages/:messageId/attachments/:attachmentId"` (όπου {gmailAccount} είναι ένα από τα strings "gmail1", "gmail2") του API.
2. Ο διακομιστής θα κάνει ένα GET αίτημα στο μονοπάτι `"/users/{userId}/messages/{messageId}/attachments/{id}"` του Gmail API, για να ανακτήσει το συνημμένο. Έπειτα, θα στείλει μία JSON απόκριση στον πελάτη με το συνημμένο. Αν κάτι πάει στραβά τα στείλει JSON απόκριση με μήνυμα που θα ενημερώνει για το λόγο αποτυχίας.

3. Ο πελάτης, αφού λάβει το συνημμένο, θα δημιουργήσει ένα σύνδεσμο για λήψη και θα το ενεργοποιήσει ώστε το συνημμένο να «κατέβει» και να αποθηκευτεί στο σύστημα αρχείων του χρήστη, και ο χρήστης θα μπορεί να το ανοίξει από εκεί.

4.9.7 Επισήμανση νήματος από Gmail λογαριασμό ως διαβασμένο, ή σημαντικό

Για να επισημανθεί ένα νήμα από κάποιο Gmail λογαριασμό του χρήστη ως αδιάβαστο, ο χρήστης θα πρέπει να ανοίξει το μήνυμα από το αντίστοιχο Gmail tab. Για να επισημανθεί ένα νήμα ως σημαντικό, εάν το «αστεράκι» (star icon) ενός μη εστιασμένου νήματος δεν είναι ήδη επιλεγμένο, ο χρήστης θα πρέπει να το επιλέξει, ενώ για να ακυρωθεί η επισήμανσή του (δηλαδή να μη θεωρείται, πλέον, σημαντικό) ο χρήστης θα πρέπει να ξαναεπιλέξει το «αστεράκι», ώστε αυτό να μην είναι, πλέον, επιλεγμένο. Και στις 3 περιπτώσεις, η ακολουθία βημάτων μεταξύ πελάτη-διακομιστή θα είναι η εξής:

1. Ο πελάτης θα κάνει ένα PATCH αίτημα στο μονοπάτι (όπου {gmailAccount} ισούται με κάποιο από τα strings “gmail1”, ή “gmail2”) “/api/{gmailAccount}/threads/:threadId/modify” με JSON φορτίο το labelId, και την πράξη που θέλουμε να γίνει με αυτό για το νήμα. Οι περιπτώσεις είναι 3:
 - i. { removeLabelIds: [“UNREAD”] } – για επισήμανση νήματος ως διαβασμένου
 - ii. { addLabelIds: [“STARRED”] } – για επισήμανση νήματος ως σημαντικού
 - iii. { removeLabelIds: [“STARRED”] } – για επισήμανση νήματος ως μη σημαντικού
2. Ο διακομιστής θα κάνει ένα POST αίτημα στο μονοπάτι “/users/{userId}/threads/{id}/modify” του Gmail API, με JSON φορτίο το JSON φορτίο που έστειλε ο πελάτης.
3. Αφού λάβει απόκριση από το Gmail API, ο διακομιστής θα αποκριθεί στον πελάτη με JSON απόκριση με το μήνυμα επιτυχίας, ή με το λόγο αποτυχίας του αιτήματος.
4. Ο πελάτης θα ενημερώσει το εστιασμένο νήμα της γραφικής διεπαφής, ή, εάν κάτι πήγε στραβά, θα ενημερώσει το χρήστη με κάποιο toast.

4.9.8 Αποστολή μηνύματος από Gmail λογαριασμό

Υπάρχουν 2 περιπτώσεις αποστολής ενός καινούριου μηνύματος από κάποιον από τους 2 Gmail λογαριασμούς του χρήστη, αποστολή ενός καινούριου μηνύματος, ή αποστολή ενός μηνύματος-απάντησης σε προηγούμενο μήνυμα. Και στις 2 περιπτώσεις, ο χρήστης θα πρέπει, αρχικά, να επιλέξει την κατάλληλη επιλογή (Compose (από το Gmail tab) για την 1η περίπτωση, ή Reply (από το τέλος ενός εστιασμένου νήματος) για τη 2η). Έπειτα, θα του εμφανιστεί μία φόρμα δημιουργίας μηνύματος, στην οποία ο χρήστης θα πρέπει να συμπληρώσει τη διεύθυνση email στην οποία θέλει να αποσταλεί το μήνυμα (στην περίπτωση που το μήνυμα είναι απάντηση, δεν χρειάζεται αυτό το βήμα, καθώς η διεύθυνση είναι γνωστή, και είναι αυτή του τελευταίου μηνύματος), καθώς και να συμπληρώσει το περιεχόμενο του μηνύματος, και τα συνημμένα τα οποία θέλει να στείλει. Τέλος, με την επιλογή Send, ή Reply θα ξεκινήσει η επικοινωνία μεταξύ πελάτη-διακομιστή για την αποστολή του μηνύματος:

1. Ο πελάτης θα κάνει ένα POST αίτημα στο μονοπάτι (όπου {gmailAccount} είναι ένα από τα strings “gmail1”, ή “gmail2”) “/api/{gmailAccount}/messages”, με JSON φορτίο το μήνυμα (δηλαδή αποστολέα, παραλήπτη, περιεχόμενο, και συνημμένα).
2. Ο διακομιστής θα αποστείλει το μήνυμα χρησιμοποιώντας τη βιβλιοθήκη Nodemailer, η οποία θα χρησιμοποιήσει το πρωτόκολλο SMTP (Simple Mail Transfer Protocol) για να ολοκληρώσει επιτυχώς την αποστολή.
3. Ο διακομιστής θα αποκριθεί με μία JSON απόκριση επιτυχίας (στην περίπτωση που όλα πάνε καλά), ή με το λόγο της αποτυχίας (σε περίπτωση που υπάρξει κάποιο πρόβλημα).
4. Ο πελάτης, αφού λάβει την απόκριση του διακομιστή, θα ενημερώσει το χρήστη για την επιτυχία ή αποτυχία αποστολής του καινούριου μηνύματος.

4.9.9 Δημιουργία καινούριου chat για το Microsoft Teams λογαριασμό

Για να δημιουργηθεί ένα καινούριο chat μεταξύ Microsoft Teams λογαριασμού του χρήστη, και ενός άλλου Microsoft Teams λογαριασμού, ο χρήστης, θα πρέπει να επιλέξει το Create a new chat input field, να πληκτρολογήσει τη διεύθυνση email ενός Microsoft Teams χρήστη, και στη συνέχεια να πατήσει enter. Έπειτα:

1. Ο πελάτης θα κάνει ένα POST αίτημα στο μονοπάτι `"/api/msTeams/chats"` του API, με JSON φορτίο τη διεύθυνση email του χρήστη με τον οποίο ο χρήστης θέλει να δημιουργήσει το καινούριο chat.
2. Ο διακομιστής, με τη σειρά του, θα κάνει, αρχικά, ένα GET αίτημα στο μονοπάτι `"/users/{userId}"` του Microsoft Graph API, όπου {userId} θα είναι η διεύθυνση email του χρήστη με τον οποίο ο χρήστης θέλει να δημιουργήσει το καινούριο chat, ώστε να ανακτήσει το όνομα του χρήστη στο λογαριασμό.
3. Ο διακομιστής θα κάνει ένα POST αίτημα στο μονοπάτι `"/chats"` του Microsoft Graph API, με JSON φορτίο που θα περιέχει τις 2 διευθύνσεις email (του χρήστη της εφαρμογής, και του άλλου χρήστη) και την οδηγία να δημιουργηθεί chat τύπου `" oneOnOne "` (όπως το αποκαλεί το Microsoft Graph API).
4. Ο διακομιστής θα αποκριθεί στον πελάτη με JSON απόκριση με μήνυμα που περιέχει τα μέλη και μία κενή λίστα μηνυμάτων του καινούριου chat που δημιουργήθηκε, σε περίπτωση επιτυχίας, ή με JSON απόκριση που θα περιέχει το λόγο της αποτυχίας.
5. Ο πελάτης, αφού λάβει την απόκριση του διακομιστή θα ενημερώσει το MsTeams tab με το καινούριο chat (ώστε ο χρήστης να μπορεί να το χρησιμοποιήσει, χωρίς να ανανεώσει τη σελίδα), ή θα ενημερώσει το χρήστη για το λόγο αποτυχίας της δημιουργίας του καινούριου chat.

4.9.10 Αποστολή μηνύματος σε chat του Microsoft Teams λογαριασμού

Για να στείλει ένα καινούριο μήνυμα σε κάποιο chat του Microsoft Teams λογαριασμού του, ο χρήστης θα πρέπει να επιλέξει το Type a new message πεδίο input, να πληκτρολογήσει το μήνυμά του, και να πατήσει το enter (για αποστολή συνημμένου θα μιλήσουμε στη συνέχεια, καθώς η διαδικασία είναι λίγο περίπλοκη από άποψης backend-διακομιστή). Έπειτα:

1. Ο πελάτης θα κάνει ένα POST αίτημα στο μονοπάτι `"/api/msTeams/chats/:chatId"` του API, με JSON φορτίο το μήνυμα.
2. Ο διακομιστής θα κάνει ένα POST αίτημα στο μονοπάτι `"/chats/{chatId}/messages"` (όπου {chatId} η παράμετρος μονοπατιού του αιτήματος) του Microsoft Graph API, με JSON φορτίο το μήνυμα.

3. Ο διακομιστής θα αποκριθεί στον πελάτη με JSON απόκριση που θα περιέχει το φορτίο απόκρισης του Microsoft Graph API endpoint που κάλεσε, σε περίπτωση επιτυχίας, ή το λόγο που υπήρξε κάποιο σφάλμα, σε περίπτωση αποτυχίας.
4. Ο πελάτης θα ενημερώσει το εστιασμένο chat με το τελευταίο μήνυμα, σε περίπτωση επιτυχής αποστολής μηνύματος, ή θα ενημερώσει το χρήστη για το λόγο αποτυχίας, σε περίπτωση αποτυχίας.

4.9.11 Αποστολή συνημμένου σε chat του Microsoft Teams λογαριασμού

Κατά την αποστολή ενός μηνύματος, είναι πιθανό ο χρήστης να θέλει να στείλει και κάποιο συνημμένο, μαζί με το κείμενο ενός μηνύματος, ή και μόνο του, σε κάποιο chat. Για να το κάνει αυτό θα πρέπει πρώτα να ανέβει το συνημμένο στον προσωπικό χώρο αποθήκευσης του χρήστη στον οποίο θα σταλεί το συνημμένο, στο OneDrive, και, έπειτα, να δώσει δικαιώματα ανάγνωσης στο χρήστη για τον οποίο αυτό προορίζεται. Ο χρήστης θα πρέπει να επιλέξει το εικονίδιο συνημμένου, κάτω από το εστιασμένο chat, όπου θα του εμφανιστεί ένα HTML input τύπου αρχείου για να επιλέξει κάποιο αρχείο. Αφού ο χρήστης επιλέξει το αρχείο, η αλληλεπίδραση μεταξύ πελάτη-διακομιστή θα είναι η εξής:

1. Ο πελάτης θα κάνει ένα POST αίτημα στο μονοπάτι `"/api/msTeams/drive"` του API, με JSON φορτίο το περιεχόμενο του αρχείου.
2. Ο διακομιστής θα κάνει, αρχικά, 1 PUT αίτημα στο μονοπάτι (όπου {filename} το όνομα του αρχείου) `"/me/drive/items/root:{fileName}/content"` του Microsoft Graph API, με φορτίο το περιεχόμενο του αρχείου, ώστε το αρχείο να ανέβει στο βασικό φάκελο (root folder) του συνδεδεμένου χρήστη της εφαρμογής στο OneDrive.
3. Ο διακομιστής θα κάνει, στη συνέχεια, ένα POST αίτημα στο μονοπάτι (όπου {itemId} είναι το αναγνωριστικό του αρχείου που ανέβηκε στο βασικό φάκελο του χρήστη στο OneDrive) `"/me/drive/items/{itemId}/invite"` του Microsoft Graph API, με JSON φορτίο που θα περιέχει τη διεύθυνση email του χρήστη στον οποίο ο συνδεδεμένος χρήστης της εφαρμογής θέλει να δώσει πρόσβαση στο

συνημμένο αρχείο, καθώς και τον τρόπο πρόσβασης, ο οποίος θα είναι το δικαίωμα ανάγνωσης του αρχείου.

4. Εάν όλα πάνε καλά, ο διακομιστής θα αποκριθεί στον πελάτη με το περιεχόμενο του αρχείου που μόλις ανέβηκε στο OneDrive του χρήστη, αλλιώς, θα αποκριθεί με μία JSON απόκριση που θα περιέχει το λόγο αποτυχίας.
5. Ο πελάτης, αφού λάβει την απόκριση, αν κάτι πήγε στραβά θα ενημερώσει το χρήστη με toast, αλλιώς θα ενημερώσει τη λίστα με συνημμένα που είναι έτοιμα για να αποσταλούν όταν ο χρήστης πατήσει enter στο εστιασμένο chat για να στείλει καινούριο μήνυμα. Ο πελάτης θα ενημερώσει, επίσης τη γραφική διεπαφή, ώστε να φαίνονται τα συνημμένα που είναι έτοιμα για αποστολή στο εστιασμένο chat.

4.9.12 Αποσύνδεση ενός συνδεδεμένου χρήστη από την εφαρμογή

Για να αποσυνδεθεί ένας χρήστης, θα πρέπει να πιέσει το Logout κουμπί, το οποίο βρίσκεται τέρμα δεξιά, στο τέλος του navbar της Home σελίδας, και στη συνέχεια:

1. Ο πελάτης θα κάνει ένα GET αίτημα στο μονοπάτι `"/api/auth/signout"` του API.
2. Ο διακομιστής θα τερματίσει τη συνεδρία του χρήστη και θα τον ανακατευθύνει στη Login σελίδα της εφαρμογής.

4.10 Πρόσβαση στα web APIs

Για να μπορέσουμε να έχουμε πρόσβαση, και να χρησιμοποιήσουμε τις υπηρεσίες των 2 διαδικτυακών APIs (δηλαδή του Gmail API, και του Microsoft Graph API), θα χρειαστεί να κάνουμε κάποια βήματα, ώστε να μπορεί, και στις 2 περιπτώσεις, να εκτελεστεί με επιτυχία το πρωτόκολλο OAuth 2.0, μεταξύ του χρήστη, του εκάστοτε διαδικτυακού API, και της εφαρμογής, κάτι που θα μας επιτρέψει να αποκτήσουμε να αποκτήσουμε access, και refresh tokens. Σε αυτή την υποενότητα, θα περιγράψουμε τον τρόπο με τον οποίο θα το κάνουμε αυτό, για κάθε ένα από τα 2 διαδικτυακά APIs.

4.10.1 Πρόσβαση στο Gmail API

Το Gmail API απαιτεί μία σειρά από 4 βήματα, ώστε να αποκτήσουμε πρόσβαση σε αυτό. Οι παρακάτω υποενότητες περιγράφουν τα βήματά, αυτά, με τη σειρά.

Καταχώρηση εφαρμογής ως Google cloud project

Αρχικά, επισκεπτόμαστε το Google cloud console (το url του είναι το "https://console.cloud.google.com/"), και επιλέγουμε από το αναπτυσσόμενο μενού IAM & Admin > Create a Project. Δίνουμε το όνομα Uni-Portal στο project μας, και επιλέγουμε CREATE. Το καινούριο Google cloud project μας έχει δημιουργηθεί.

Ενεργοποίηση Gmail API

Στη συνέχεια, πρέπει να ενεργοποιήσουμε το Gmail API, από το project μας. Έχοντας επιλεγμένο το καινούριο Google cloud project, που μόλις δημιουργήσαμε (δηλαδή το Uni-Portal), επιλέγουμε από το αναπτυσσόμενο μενού APIs & Services > Library. Στο πεδίο αναζήτησης πληκτρολογούμε Gmail API, αφού εμφανιστεί το επιλέγουμε, και μετά επιλέγουμε ENABLE. Το Gmail API έχει ενεργοποιηθεί για το Google cloud project μας.

Διαμόρφωση οθόνης συγκατάθεσης Google OAuth

Σε αυτό το σημείο, ήρθε η ώρα να ρυθμίσουμε την οθόνη συγκατάθεσης Google OAuth, που θα εμφανίζεται στο χρήστη, ώστε να δώσει τα στοιχεία του και να αυθεντικοποιηθεί, κατά τη ροή του πρωτοκόλλου OAuth 2.0. Από το αναπτυσσόμενο μενού, επιλέγουμε APIs & Services > OAuth consent screen. Επιλέγουμε External User Type (ώστε να μπορεί να χρησιμοποιήσει την εφαρμογή μας οποιοσδήποτε χρήστης έχει Gmail λογαριασμό), και στη συνέχεια CREATE. Στο βήμα OAuth consent screen συμπληρώνουμε τη φόρμα, με το όνομα της εφαρμογής (Uni-Portal), καθώς και κάποιο email για επικοινωνία από τη Google, και επιλέγουμε SAVE AND CONTINUE. Στη συνέχεια, στο βήμα Scopes, επιλέγουμε ADD OR REMOVE SCOPES, και διαλέγουμε το scope "https://mail.google.com/", το οποίο είναι περιοριστικό, καθώς μας δίνει πλήρη πρόσβαση στο λογαριασμό του χρήστη. Συγκεκριμένα, με αυτό το scope η εφαρμογή μας μπορεί να διαβάσει όλα τα μηνύματα, να δημιουργήσει καινούρια μηνύματα, καθώς και να διαγράψει μηνύματα, από το λογαριασμό του χρήστη, και, συνεπώς, τη δυνατότητα να δημιουργήσουμε, στην εφαρμογή μας, τη λειτουργικότητα που θέλουμε. Έπειτα, στο βήμα Test users, προσθέτουμε τις διευθύνσεις email τις οποίες θέλουμε να χρησιμοποιήσουν την εφαρμογή μας κατά τη διάρκεια του development (πριν γίνει το publish), με την επιλογή ADD USERS και επιλέγουμε SAVE AND CONTINUE. Τέλος, επιλέγουμε BACK TO DASHBOARD.

Δημιουργία διαπιστευτηρίων πρόσβασης

Αφού έχουμε κάνει διαμορφώσει τη οθόνης συγκατάθεσης Google OAuth, αυτό που μένει να κάνουμε, είναι να δημιουργήσουμε διαπιστευτήρια πρόσβασης για την εφαρμογή μας. Από το αναπτυσσόμενο μενού, επιλέγουμε APIs & Services > Credentials. Έπειτα, επιλέγουμε Create Credentials > OAuth client ID, και στη συνέχεια Application type > Web application. Στα Authorized redirect URIs επιλέγουμε ADD URI, και εισάγουμε τα URIs “http://localhost:3001/api/connect/gmail1/callback”, “http://localhost:3001/api/connect/gmail2/callback”, τα οποία θα είναι τα endpoints του διακομιστή/API μας, κατά τη φάση της ανάπτυξης της εφαρμογής, όπου η Google θα επιστρέφει τα access, refresh tokens αφού ο χρήστης αυθεντικοποιηθεί από τη Google, και εξουσιοδοτήσει την εφαρμογή μας να έχει τα scopes που ζητήσαμε, για τον κάθε λογαριασμό του. Έπειτα, κάνουμε CREATE, αποθηκεύουμε το client id, και το client secret, που δημιούργησε η Google για την εφαρμογή μας, και επιλέγουμε OK. Πλέον, μπορούμε να χρησιμοποιήσουμε τα client id, secret, ώστε να κάνουμε εξουσιοδότηση OAuth 2.0 και να αποκτήσουμε access, και refresh tokens (στα κατάλληλα endpoints), για να έχουμε πρόσβαση στο Gmail API.

4.10.2 Πρόσβαση στο Microsoft Graph API

Το Microsoft Graph API, για να μπορέσουμε να έχουμε πρόσβαση στις υπηρεσίες του, απαιτεί, αρχικά, την καταχώρηση της εφαρμογής μας στο Azure portal, ώστε να μπορούμε στη συνέχεια να αποκτήσουμε access tokens και να τα χρησιμοποιήσουμε για να κάνουμε κλήσεις στο API.

Καταχώρηση εφαρμογής στο Azure portal και δημιουργία credentials

Αρχικά, επισκεπτόμαστε το Azure portal, και στο πλαίσιο αναζήτησης εισάγουμε App registrations. Επιλέγουμε το App registrations, από τα αποτελέσματα της αναζήτησης, και στη συνέχεια New registration. Συμπληρώνουμε τη φόρμα, βάζοντας το όνομα της εφαρμογής (Uni-Portal) στο πεδίο Name, ως Supported account types επιλέγουμε Accounts in any organizational directory (Any Azure AD directory - Multitenant), και στην επιλογή Redirect URI (optional), επιλέγουμε Web ως πλατφόρμα, και εισάγουμε το URI ανακατεύθυνσης “http://localhost:3001/api/connect/msTeams/callback”, στο οποίο θα επιστρέφει η Microsoft τα access, και refresh tokens, αφού αυθεντικοποιήσει το χρήστη και λάβει τη συγκατάθεση του για την εξουσιοδότηση της εφαρμογής να έχει πρόσβαση στο Microsoft Teams λογαριασμό του. Έπειτα, επιλέγουμε Register, και αποθηκεύουμε το

client id που δημιουργήθηκε από την καταχώρηση του Uni-Portal App. Τώρα μπορούμε να προσθέσουμε ένα client secret. Έχοντας επιλεγμένη την εφαρμογή μας, από το App registrations, επιλέγουμε Manage > Certificates & secrets > New client secret > Add, και αποθηκεύουμε το secret που δημιουργήθηκε. Πλέον, μπορούμε να χρησιμοποιήσουμε τα client id, secret, ώστε να εκτελέσουμε το πρωτόκολλο OAuth 2.0 και να αποκτήσουμε τα access, refresh tokens στο callback url που έχουμε ορίσει.

Κεφάλαιο 5. Η γραφική διεπαφή της εφαρμογής

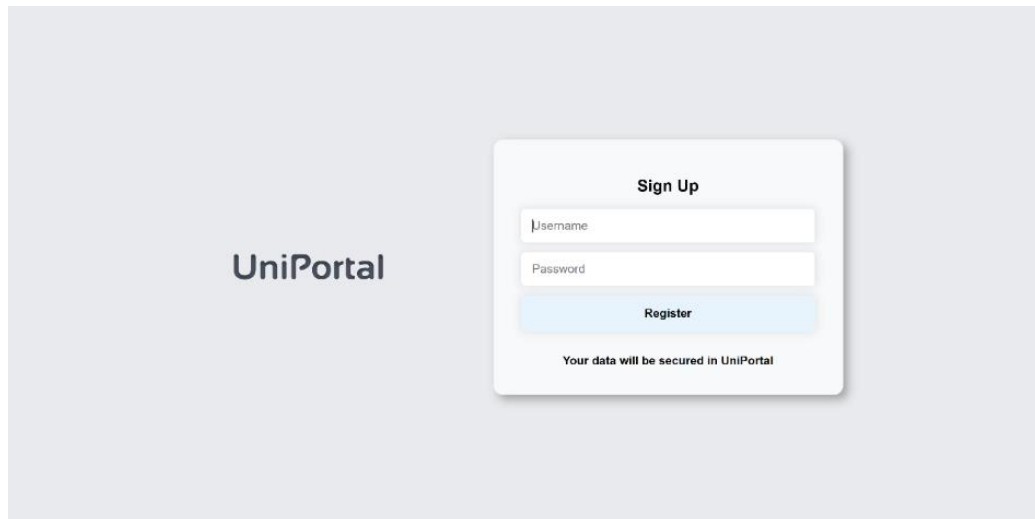
5.1 Οι βασικές σελίδες της εφαρμογής

Το frontend της εφαρμογής μας αποτελείται από ένα React SPA. Το React SPA, με τη σειρά του, αποτελείται από 2 βασικά components, τα οποία μπορούμε να θεωρήσουμε ως σελίδες. Τα components, αυτά, ονομάζονται Login και Home. Η Login σελίδα δίνει τη δυνατότητα στο χρήστη για εγγραφή και σύνδεση στην εφαρμογή, ενώ η Home σελίδα θα περιέχει την υπόλοιπη λειτουργικότητα.

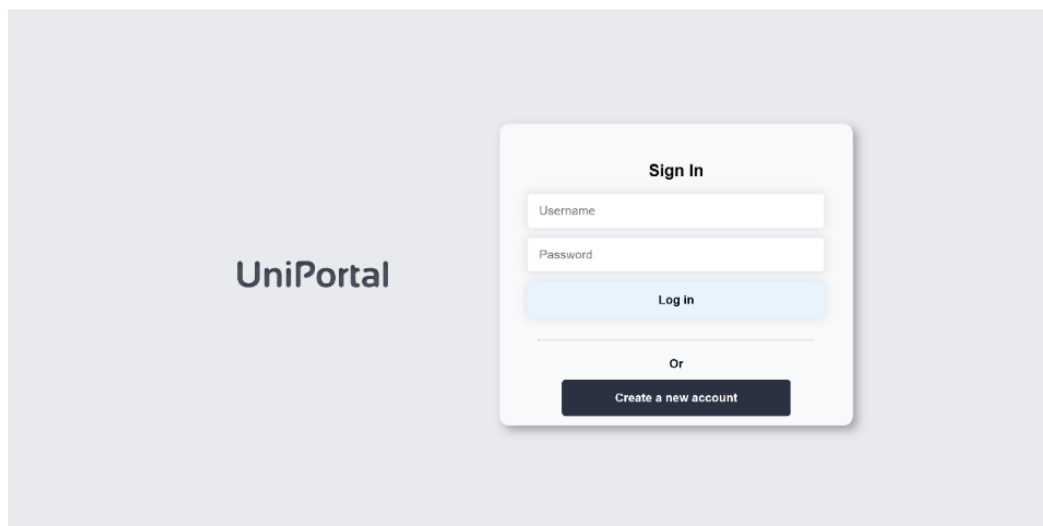
Καθώς το frontend μας είναι SPA, κάθε μία από τις σελίδες εμφανίζεται στο ίδιο μονοπάτι (συγκεκριμένα στο βασικό, ή root, μονοπάτι της εφαρμογής), αναλόγως εάν ο χρήστης είναι συνδεδεμένος. Δηλαδή εάν ο χρήστης δεν είναι συνδεδεμένος στο μονοπάτι “/” εμφανίζεται η Login σελίδα, ενώ αν είναι συνδεδεμένος εμφανίζεται η Home σελίδα. Για να αποφασίσει το frontend ποια από τις 2 σελίδες/components θα εμφανίσει χρησιμοποιεί το endpoint “/api/auth/authenticated” του API του backend διαδικτυακού διακομιστή, το οποίο δημιουργήσαμε για αυτόν ακριβώς το σκοπό. Το frontend SPA, κάθε φορά θα που ο χρήστης περιηγείται στο βασικό μονοπάτι της εφαρμογής, κάνει ένα HTTP GET αίτημα σε αυτό το endpoint, και ο backend διακομιστής αποκρίνεται με μία JSON απόκριση που το ενημερώνει εάν έχει δημιουργηθεί συνεδρία για το χρήστη, ή όχι. Έτσι, το frontend μπορεί να εμφανίσει στο χρήστη την κατάλληλη σελίδα.

Η Login σελίδα περιέχει 2 φόρμες, ώστε να μπορεί ο χρήστης να κάνει εγγραφή, και σύνδεση, στην εφαρμογή. Η Home σελίδα, από την άλλη, περιέχει τα μηνύματα του χρήστη, από τους 2 Gmail λογαριασμούς, αλλά και τον Microsoft Teams λογαριασμό του, καθώς και διάφορες φόρμες ώστε να μπορεί να στέλνει μηνύματα, να δημιουργεί καινούρια chats/νήματα (chats στο Microsoft Teams, νήματα στο Gmail), να επισημαίνει

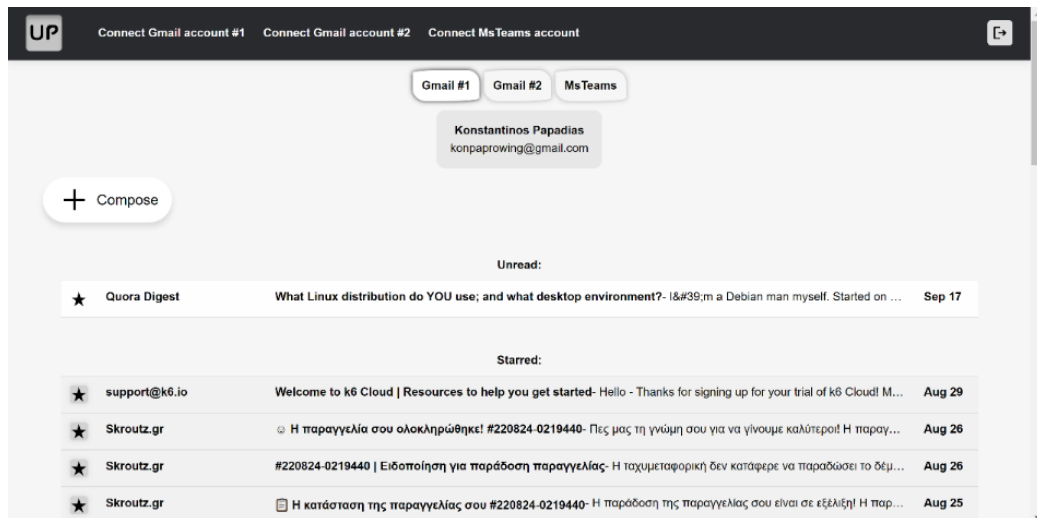
συνομιλίες ως σημαντικές ή διαβασμένες στο Gmail κ.τ.λ.. Επίσης, η Home σελίδα δίνει τη δυνατότητα στο χρήστη να συνδέσει (δηλαδή να εξουσιοδοτήσει, μέσω του πρωτοκόλλου OAuth 2.0 την εφαρμογή, ώστε να διαχειρίζεται τους λογαριασμούς του), και να αποσυνδεθεί από την εφαρμογή.

The image shows a web page for UniPortal with a 'Sign Up' form. The form is centered on a light gray background. To the left of the form is the 'UniPortal' logo. The form has a title 'Sign Up', two input fields for 'Username' and 'Password', a blue 'Register' button, and a small text line at the bottom stating 'Your data will be secured in UniPortal'.

Εικόνα 5.1 Η Login σελίδα, με ενεργό το Register component.

The image shows a web page for UniPortal with a 'Sign In' form. The form is centered on a light gray background. To the left of the form is the 'UniPortal' logo. The form has a title 'Sign In', two input fields for 'Username' and 'Password', a blue 'Log in' button, a horizontal line, the word 'Or', and a dark blue 'Create a new account' button.

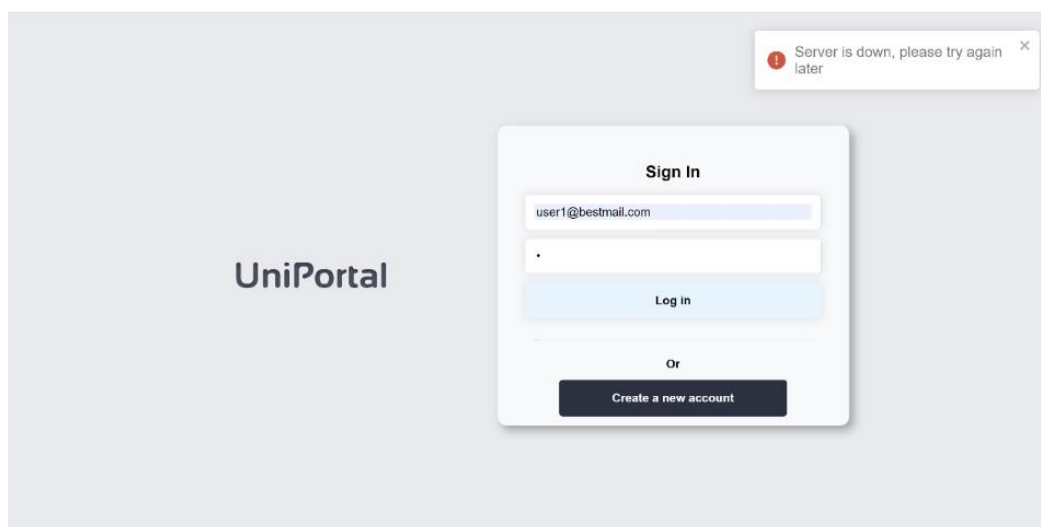
Εικόνα 5.2 Η Login σελίδα, με ενεργό το Login component.



Εικόνα 5.3 Η Home σελίδα.

5.2 Εγγραφή και σύνδεση χρήστη στην εφαρμογή

Η εγγραφή ενός καινούριου χρήστη, καθώς και η σύνδεση ενός ήδη εγγεγραμμένου χρήστη, είναι αρκετά απλή. Ο χρήστης εισάγει το όνομα χρήστη του, καθώς και το συνθηματικό του στη φόρμα. Έπειτα επιλέγει την κατάλληλη επιλογή (Register, ή Log in) και, εάν τα στοιχεία είναι σωστά, γίνεται η σύνδεση, ή εγγραφή και σύνδεση του χρήστη, και τελικά εμφανίζεται η Home σελίδα/component. Εάν τα στοιχεία του είναι λάθος, ή ο διακομιστής (δηλαδή το backend μας) είναι εκτός λειτουργίας και δεν αποκρίνεται, εμφανίζεται toast με το κατάλληλο μήνυμα στο χρήστη.



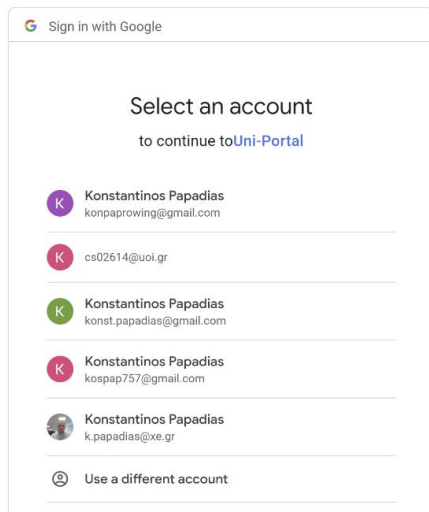
Εικόνα 5.4 Μήνυμα ότι ο διακομιστής είναι εκτός λειτουργίας κατά τη διαδικασία της σύνδεσης.

5.3 Σύνδεση Gmail, ή Microsoft Teams λογαριασμού στην εφαρμογή

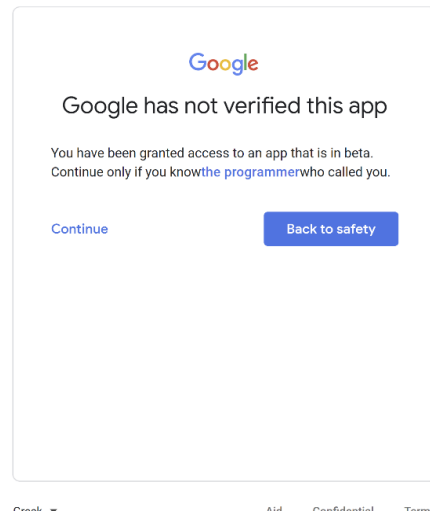
Έχοντας συνδεθεί, ο χρήστης βρίσκεται στη Home σελίδα. Από εκεί μπορεί να επιλέξει από το Home navbar μία από τις 3 επιλογές: “Connect Gmail account #1”, “Connect Gmail account #2”, “ Connect MsTeams account”, ώστε να ξεκινήσει η διαδικασία της εξουσιοδότησης OAuth 2.0. Και στις 2 περιπτώσεις τα βήματα που ακολουθούνται είναι τα ίδια, όσον αφορά το χρήστη, και είναι τα εξής:

1. Η εφαρμογή μας ανακατευθύνει το χρήστη σε κάποιον διακομιστή αυθεντικοποίησης, ώστε να ταυτοποιηθεί από τον κατάλληλο πάροχο.
2. Ο χρήστης ταυτοποιείται από τον πάροχο.
3. Ο πάροχος ενημερώνει το χρήστη για τις άδειες που ζητάει η εφαρμογή μας για το λογαριασμό του, και τον ρωτά εάν θέλει να κάνει την εξουσιοδότηση.
4. Ο χρήστης απαντάει θετικά, και η εφαρμογή μας αποκτά access tokens για το λογαριασμό του χρήστη, ανακατευθύνοντάς τον στη Home σελίδα.

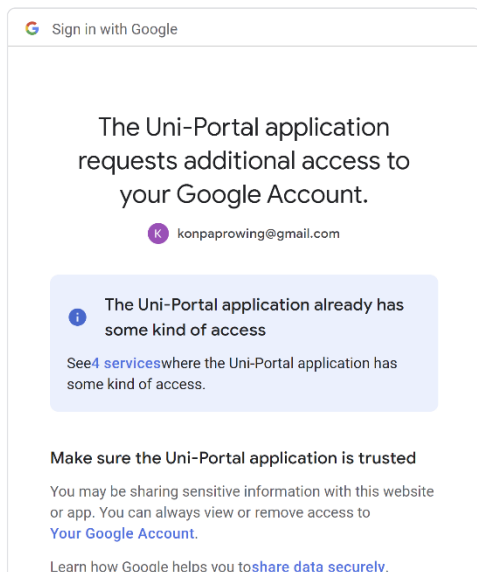
Στη συνέχεια ακολουθούν στιγμιότυπα από τη διαδικασία για ένα Gmail, καθώς και ένα Microsoft Teams λογαριασμό του χρήστη.



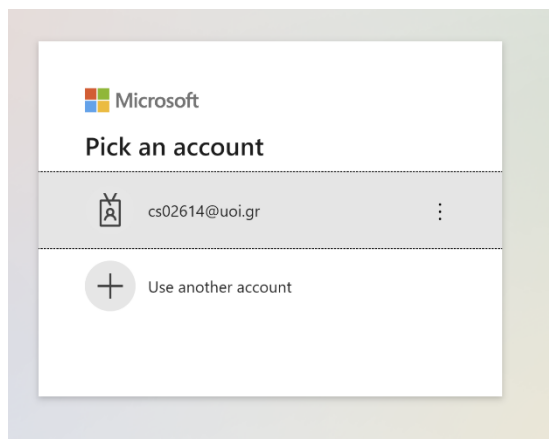
Εικόνα 5.5 Επιλογή λογαριασμού για τον οποίο θα γίνει η εξουσιοδότηση. Εμφανίζεται στον Google Chrome περιηγητή ιστού, για τους λογαριασμούς που ο χρήστης έχει συνδεθεί στο παρελθόν.



Εικόνα 5.6 Μήνυμα που εμφανίζεται από την Google, κατά την περίοδο που η εφαρμογή είναι στο στάδιο ανάπτυξης, και δεν έχει βγει στην παραγωγή.



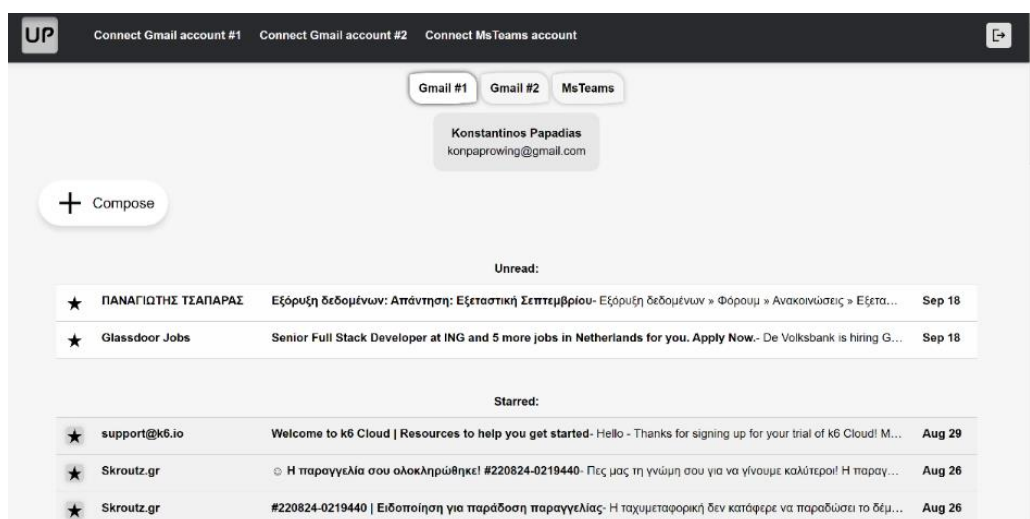
Εικόνα 5.7 Μήνυμα ενημέρωσης στο χρήστη, για τις άδειες που ζητάει η εφαρμογή μας, για το λογαριασμό του, από τη Google. Εμφανίζεται αφού ο χρήστης έχει ταυτοποιηθεί μέσω του email και του κωδικού του, από τη Google.



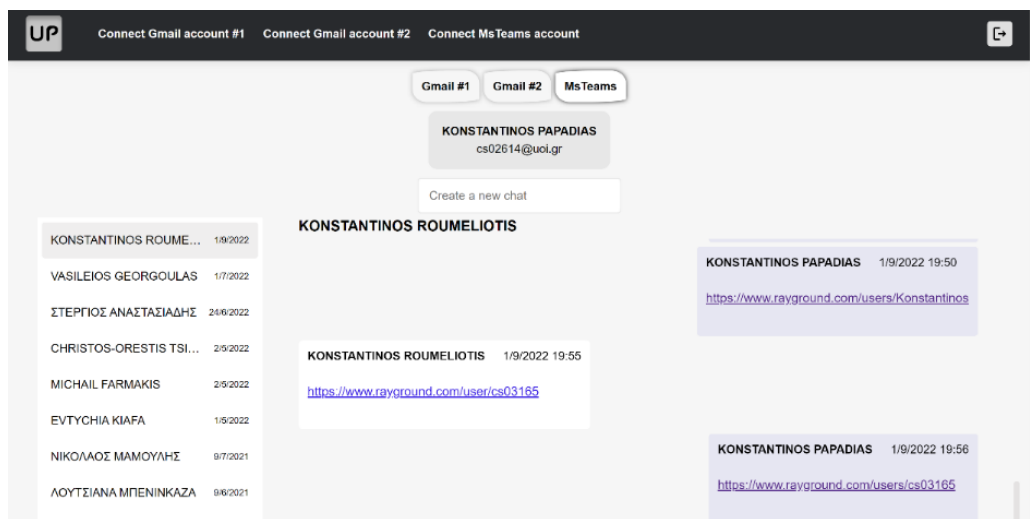
Εικόνα 5.8 Προτροπή για ταυτοποίηση σε κάποιο Microsoft Teams λογαριασμό. Αφού ο χρήστης ταυτοποιηθεί, η υπόλοιπη διαδικασία γίνεται αυτόματα, και ο χρήστης ανακατευθύνεται, τελικά, στη Home σελίδα.

5.4 Πρόσβαση στους λογαριασμούς μέσω της Home σελίδας

Έχοντας συνδέσει τους λογαριασμούς του, ο χρήστης μπορεί να περιηγηθεί σε καθένα από αυτούς, από τη Home σελίδα. Επιλέγοντας το κατάλληλο από τα 3 tabs ("Gmail #1", "Gmail #2", "MsTeams"), θα του εμφανιστούν τα μηνύματα του κατάλληλου λογαριασμού, καθώς και τα εργαλεία για να τον διαχειριστεί. Δηλαδή για να περιηγηθεί από τον 1 λογαριασμό στον άλλο, ο χρήστης αρκεί να επιλέξει άλλο tab.



Εικόνα 5.9 Πρόσβαση στον 1ο Gmail λογαριασμό του χρήστη, επιλέγοντας το tab "Gmail #1".

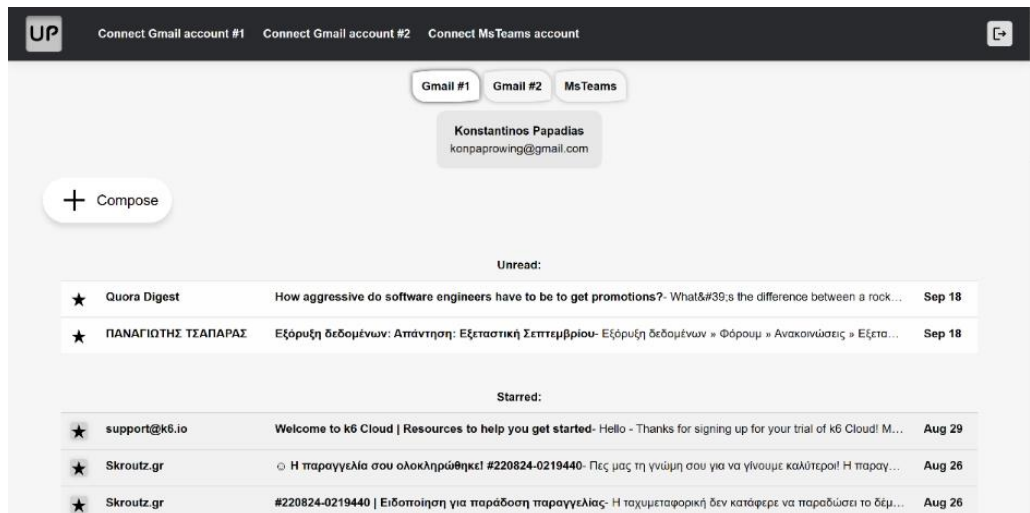


Εικόνα 5.10 Πρόσβαση στο Microsoft Teams λογαριασμό του χρήστη μέσω του tab "MsTeams".

5.5 Οι Gmail λογαριασμοί

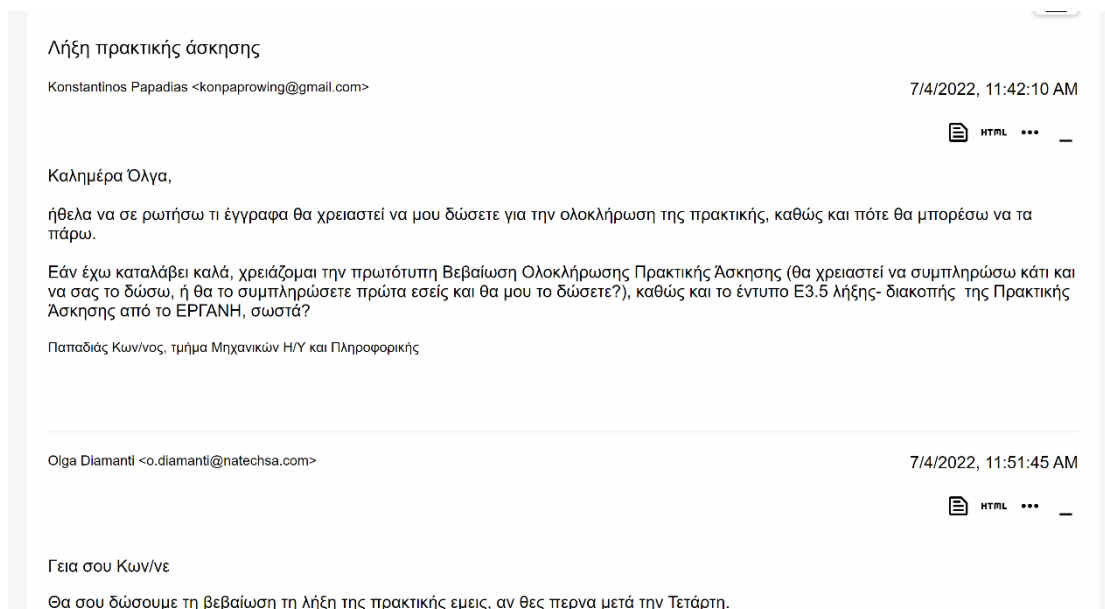
5.5.1 Προβολή των μηνυμάτων

Επιλέγοντας κάποιο από τα 2 Gmail tabs της Home σελίδας, ο χρήστης βλέπει τα νήματα μηνυμάτων, τα οποία περιέχουν κάποιο αδιάβαστο μήνυμα, ή έχουν χαρακτηριστεί ως σημαντικά. Είναι σε μορφή λίστας και όταν είναι μη εστιασμένα περιέχουν πληροφορίες για τη συνομιλία όπως εάν είναι αδιάβαστη, ή σημαντική, από ποιόν προέρχονται τα μηνύματα, ένα απόσπασμα του μηνύματος, την ημερομηνία του τελευταίου μηνύματος κ.τ.λ.. Για να επισημανθεί ένα νήμα ως σημαντικό, ο χρήστης αρκεί να επιλέξει το star εικονίδιο (ώστε να είναι σημαντικό), ή να το κάνει μη επιλεγμένο (ώστε να μη θεωρείται, πλέον, σημαντικό).



Εικόνα 5.11 Παράδειγμα Gmail inbox. Τα άσπρα, μη εστιασμένα, νήματα περιέχουν αδιάβαστο μήνυμα, ενώ τα μη εστιασμένα νήματα που έχουν επιλεγμένο το αστεράκι έχουν επισημανθεί ως σημαντικά.

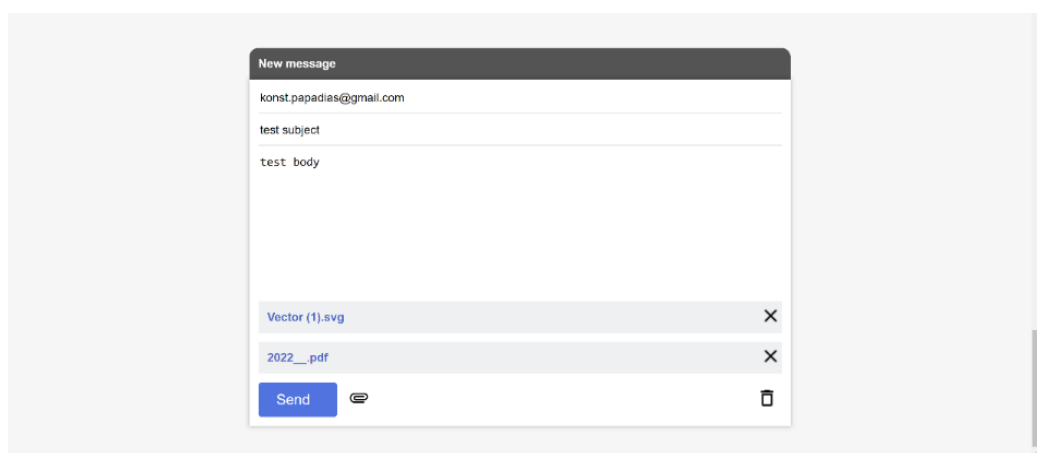
Επιλέγοντας κάποιο από τα νήματα, το νήμα “ανοίγει”, και επισημαίνεται ως διαβασμένο. Ο χρήστης μπορεί να δει όλα τα μηνύματά του, σε σειρά από το πιο πρόσφατο στο πιο παλιό. Ο χρήστης μπορεί να επιλέξει με ποιον τρόπο θέλει να δει κάθε μήνυμα από τα 4 εικονίδια πάνω δεξιά από κάθε μήνυμα. Οι επιλογές είναι (από αριστερά προς τα δεξιά): απλό κείμενο, html χωρίς όλα τα προηγούμενα μηνύματα, html με τα προηγούμενα μηνύματα, ή ελαχιστοποίηση του περιεχομένου.



Εικόνα 5.12 Παράδειγμα εστιασμένου νήματος. Ο χρήστης μπορεί να το ξανακάνει μη επιλεγμένο με το εικονίδιο της ελαχιστοποίησης πάνω δεξιά.

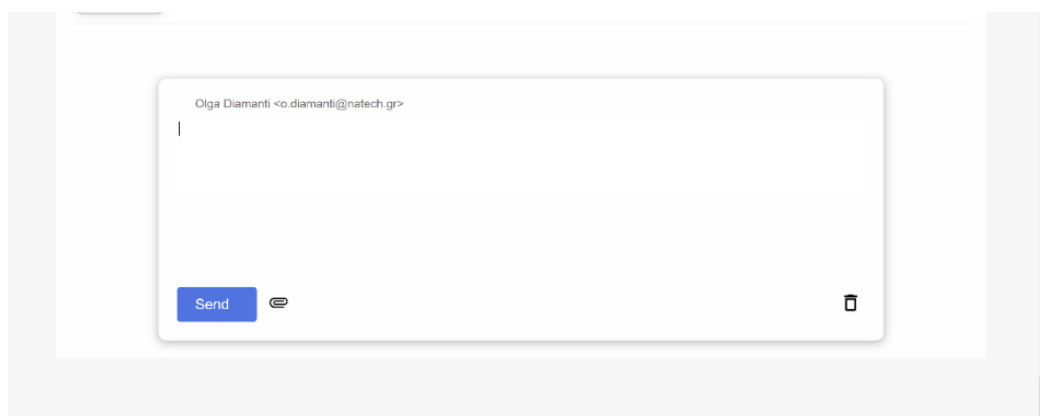
5.5.2 Αποστολή καινούριου μηνύματος ή μηνύματος απάντησης σε νήμα

Από το Google λογαριασμό του ο χρήστης μπορεί να κάνει αποστολή ενός καινούριου μηνύματος πατώντας στο κουμπί “Compose”. Η σελίδα θα κάνει κύλιση προς τα κάτω, στο τέλος των νημάτων, όπου θα εμφανιστεί μια φόρμα στο χρήστη που θα του επιτρέπει να εισάγει κείμενο, να επιλέξει συνημμένα, και, είτε να στείλει το μήνυμα, είτε να το διαγράψει. Κατά την αποστολή του μηνύματος θα υπάρξουν toasts που θα ενημερώνουν το χρήστη για την επιτυχία, ή μη, παράδοσης του μηνύματος.



Εικόνα 5.13 Παράδειγμα αποστολής καινούριου μηνύματος, με 2 συνημμένα.

Για αποστολή ενός μηνύματος απάντησης ο χρήστης μπορεί να πατήσει το κουμπί “Reply” που βρίσκεται κάτω από το πιο πρόσφατο μήνυμα ενός νήματος. Θα του εμφανιστεί μία φόρμα παρόμοια με αυτή της αποστολής καινούριου μηνύματος, από την οποία θα μπορεί να αποστείλει το μήνυμα.

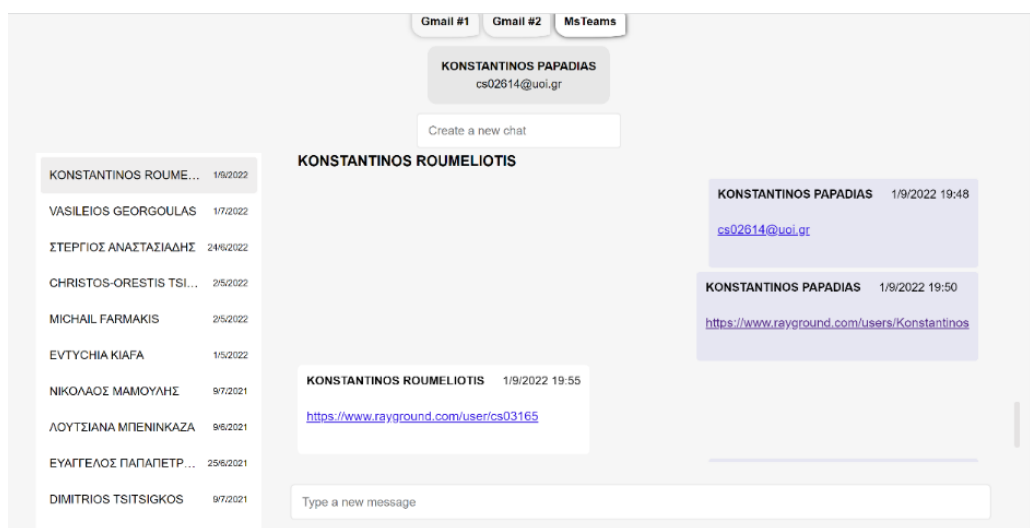


Εικόνα 5.14 Παράδειγμα φόρμας αποστολής μηνύματος απάντησης σε ήδη υπάρχων νήμα.

5.6 Ο Microsoft Teams λογαριασμός

5.6.1 Προβολή των chats

Επιλέγοντας το MsTeams tab από τη Home σελίδα, ο χρήστης έχει πρόσβαση στο Microsoft Teams λογαριασμό του. Στα αριστερά εμφανίζονται ως λίστα τα chats, με πρώτο το chat στο οποίο υπήρξε πιο πρόσφατα μήνυμα, ενώ δεξιά εμφανίζονται τα μηνύματα του επιλεγμένου chat (διαβάζονται από κάτω προς τα πάνω). Για αλλαγή chat, ο χρήστης αρκεί να επιλέξει από τη λίστα άλλο chat, ώστε το χρώμα του να αλλάξει σε γκρι.



Εικόνα 5.15 Παράδειγμα εμφάνισης των chats του χρήστη για το Microsoft Teams λογαριασμό του. Το γκρι chat είναι το επιλεγμένο, και για να μεταβεί σε παλαιότερα μηνύματα, ο χρήστης κάνει κύλιση προς τα πάνω.

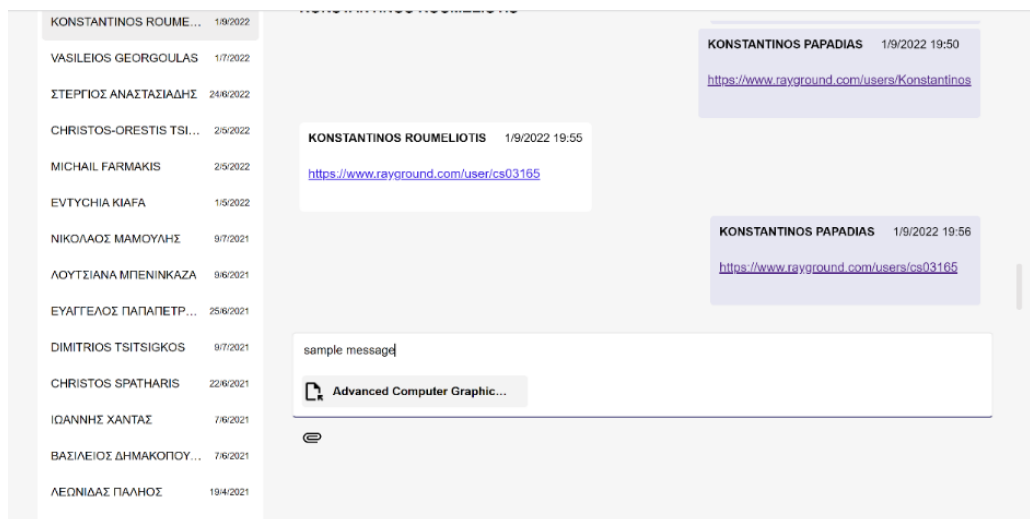
5.6.2 Δημιουργία καινούριου chat

Για να δημιουργήσει ένα καινούριο chat, ο χρήστης αρκεί να εισάγει τη διεύθυνση email ενός χρήστη στο πεδίο “Create new chat” και να πατήσει Enter.

5.6.3 Αποστολή μηνύματος σε chat

Έχοντας επιλεγμένο το chat στο οποίο θέλει να στείλει μήνυμα, ο χρήστης αρκεί να εισάγει το μήνυμα που θέλει στο πεδίο “Type a new message” (κάτω από τα μηνύματα του επιλεγμένου chat), και να πατήσει Enter. Εάν θέλει να στείλει και κάποιο συνημμένο, θα πρέπει να επιλέξει το εικονίδιο του συνημμένου (κάτω από το πεδίο “Type a new message”), και έπειτα να επιλέξει το συνημμένο που θέλει. Το συνημμένο θα ανέβει στο OneDrive του χρήστη, ενώ θα δοθεί και άδεια ανάγνωσης στο χρήστη στον οποίο ετοιμάζεται να σταλεί το μήνυμα. Για όλα αυτά υπάρχει toast που

ενημερώνει το χρήστη. Έπειτα το συνημμένο φαίνεται στο πεδίο που ο χρήστης έχει εισάγει το μήνυμά του, κάτω από το μήνυμα. Πατώντας Enter, ο χρήστης μπορεί να αποστείλει το μήνυμα.



Εικόνα 5.16 Παράδειγμα μηνύματος που περιέχει συνημμένο. Πριν εμφανιστεί το συνημμένο κάτω από το μήνυμα, θα έχει ανέβει στο OneDrive του χρήστη, ενώ θα έχει δοθεί άδεια ανάγνωσης για αυτό, στο χρήστη για τον οποίο προορίζεται.

5.7 Αποσύνδεση χρήστη από την εφαρμογή

Για να αποσυνδεθεί ένας χρήστης, αρκεί να πιάσει το εικονίδιο αποσύνδεσης, πάνω δεξιά στο πανbar της Home σελίδας. Θα τερματιστεί η συνεδρία του και θα ανακατευθυνθεί στη Login σελίδα.

5.8 Το λογότυπο της εφαρμογής

Για να δημιουργήσουμε το λογότυπο της εφαρμογής χρησιμοποιήσαμε το σχεδιαστικό εργαλείο Figma, το οποίο χρησιμοποιείται από επαγγελματίες UI/UX designers. Τα λογότυπα είναι 2, ένα με τα αρχικά της εφαρμογής, και 1 μεγαλύτερο που περιέχει όλο το όνομά της.



Εικόνα 5.17 Το πλήρες λογότυπο της εφαρμογής.



Εικόνα 5.18 Το λογότυπο με τα αρχικά της εφαρμογής.

Κεφάλαιο 6. Αξιολόγηση Απόδοσης

6.1 Πειραματικό Περιβάλλον

Για να αξιολογήσουμε την απόδοση της εφαρμογής μας, θα πρέπει, αρχικά, να αναφέρουμε το υπολογιστικό περιβάλλον, πάνω στο οποίο θα εκτελέσουμε τα πειράματά μας. Όλα τα πειράματα θα εκτελεστούν σε ένα laptop Dell XPS 13 9310 με τα ακόλουθα χαρακτηριστικά:

CPU	Intel® Core™ i7-1185G7 @ 3.00 GHz
Πυρήνες CPU	4
Λογικοί Επεξεργαστές CPU	8
RAM	16 GB
Λειτουργικό	Windows 10 Home
Έκδοση Λειτουργικού	21H2

Πίνακας 6.1 Τα χαρακτηριστικά του υπολογιστή, στον οποίο κάναμε τα πειράματα για την αξιολόγηση απόδοσης της εφαρμογής.

6.2 Τρόπος αξιολόγησης της απόδοσης και μετρική

Ο τρόπος με τον οποίο θα αξιολογήσουμε την απόδοση της εφαρμογής μας θα είναι το τεστ χρόνου απόκρισης (response time testing). Το τεστ χρόνου απόκρισης είναι ένα είδος test, που έχει ως σκοπό να αξιολογήσει την ταχύτητα με την οποία μια εφαρμογή ικανοποιεί τα αιτήματα των χρηστών. Ο χρόνος απόκρισης ξεκινά από τη στιγμή που ο χρήστης κάνει ένα αίτημα στην εφαρμογή, και τελειώνει τη στιγμή που το

αίτημα ικανοποιείται. Χρησιμοποιείται όταν θέλουμε να αξιολογήσουμε εάν η εφαρμογή μας ικανοποιεί συγκεκριμένα κριτήρια απόδοσης.

Η μετρική που θα χρησιμοποιήσουμε για την αξιολόγηση θα είναι ο μέσος χρόνος απόκρισης, ώστε να δούμε πόσο γρήγορα η εφαρμογή μας ικανοποιεί συγκεκριμένα αιτήματα, κατά μέσο όρο, κατά την ύπαρξη διαφορετικού φόρτου στο σύστημα.

6.3 Εργαλείο αξιολόγησης

Το εργαλείο που θα μας βοηθήσει στην αξιολόγηση, είναι το k6. Είναι ένα εργαλείο ανοιχτού κώδικα, από την Grafana Labs, φτιαγμένο ειδικά για τεστ φόρτου (load testing). Δουλεύει προσομοιώνοντας ένα περιβάλλον πολλών χρηστών που στέλνουν, παράλληλα, αιτήματα στην εφαρμογή. Μπορεί να τρέξει από τερματικό, και, αφού τα tests τελειώσουν, προβάλλει τα αποτελέσματα στο stdout, μεταξύ άλλων.

Η Grafana Labs προσφέρει, επίσης, ένα Google Chrome plugin, με το οποίο μπορούμε να προσομοιώσουμε τα βήματα που θα ακολουθούσε ένας χρήστης μέσω του περιηγητή ιστού, καταγράφοντας τα αιτήματα που θα στείλει στην εφαρμογή μας, σε ένα πραγματικό σενάριο χρήσης, ώστε να τα χρησιμοποιήσουμε, στη συνέχεια, για το τεστ χρόνου απόκρισης.

6.4 Οι λειτουργίες της εφαρμογής που θα αξιολογήσουμε

Αποφασίσαμε να μην τεστάρουμε τη δημιουργία λογαριασμού, καθώς αυτό γίνεται μόνο την 1^η φορά, από έναν καινούριο χρήστη. Για τον ίδιο λόγο αποφασίσαμε να μην τεστάρουμε την εξουσιοδότηση της εφαρμογής για να διαχειρίζεται τους λογαριασμούς του χρήστη. Επίσης, δεν έχει και τόσο νόημα να τεστάρουμε τη σύνδεση και την αποσύνδεση ενός χρήστη στην εφαρμογή. Συνεπώς, οι λειτουργίες που αποφασίσαμε να αξιολογήσουμε είναι οι εξής:

1. Ταυτόχρονη ανάκτηση των μηνυμάτων, των 3 λογαριασμών του χρήστη.
2. Αποστολή μηνύματος, από Gmail λογαριασμό του χρήστη.
3. Ανέβασμα αρχείου στο OneDrive του χρήστη, και δημιουργία δικαιώματος ανάγνωσης για το χρήστη για τον οποίο προορίζεται.
4. Αποστολή μηνύματος από το Microsoft Teams λογαριασμό του χρήστη.

6.5 Δημιουργία των tests

Για κάθε λειτουργία που θα αξιολογήσουμε, θα εκτελέσουμε 3 διαφορετικά τεστ. Σε όλα τα τεστ, εκτός από το ανέβασμα αρχείου στο OneDrive, ο αριθμός των εικονικών χρηστών θα είναι διαφορετικός, ώστε να δούμε πως αλλάζει ο χρόνος απόκρισης όσο αυξάνονται οι χρήστες που στέλνουν παράλληλα αιτήματα. Για το τεστ της αποθήκευσης αρχείου στο OneDrive και της δημιουργίας δικαιωμάτων ανάγνωσης για τον παραλήπτη, η μεταβλητή μεταξύ των 3 τεστ θα είναι το μέγεθος του αρχείου που ανεβάνει, ενώ ο χρήστης θα είναι ένας, για κάθε πείραμα. Κάθε ένα από τα 3 τεστ θα το εκτελέσουμε από 5 φορές, ώστε να βγάλουμε το μέσο όρο χρόνου απόκρισης (average response time), καθώς αναλόγως το φόρτο του δικτύου, ή του λειτουργικού, μπορεί να υπάρξουν διαφορές στο χρόνο απόκρισης μεταξύ των αιτημάτων. Έτσι θα έχουμε πιο αξιόπιστα αποτελέσματα.

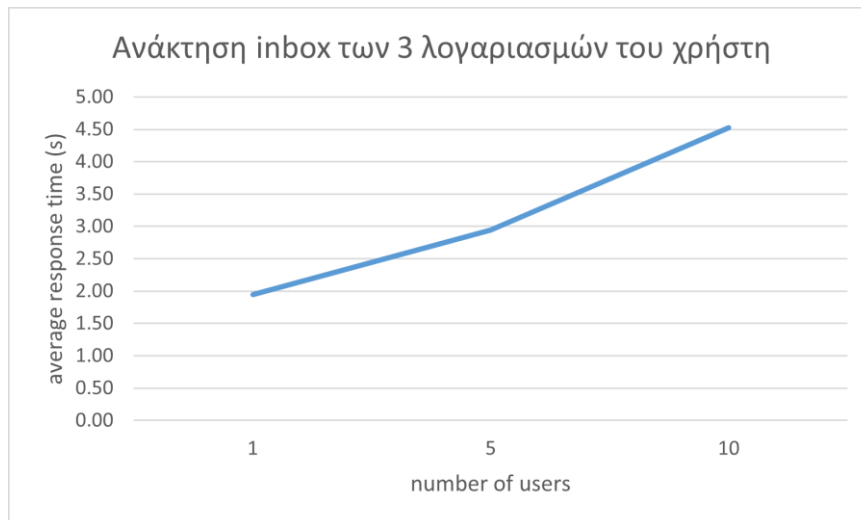
Αξίζει να σημειωθεί ότι οι 3 λογαριασμοί που θα χρησιμοποιηθούν για τα τεστ είναι 3 λογαριασμοί που χρησιμοποιούνται καθημερινά, και έχουν πολλά δεδομένα (νήματα, chats, εισερχόμενα και εξερχόμενα μηνύματα κ.τ.λ.), συνεπώς τα πειράματα προσομοιώνουν πραγματικές συνθήκες λειτουργιών σε Gmail και Microsoft Teams λογαριασμούς.

6.6 Αξιολογήσεις λειτουργιών της εφαρμογής

6.6.1 Ταυτόχρονη ανάκτηση των μηνυμάτων, των 3 λογαριασμών του χρήστη

Στο σχήμα 6.1 παρουσιάζονται τα αποτελέσματα των χρονομετρήσεων που κάναμε για τη λειτουργία της ανάκτησης των μηνυμάτων, των 3 λογαριασμών του χρήστη. Εκτελέσαμε 3 πειράματα, κάθε πείραμα για 5 φορές, και κρατήσαμε το μέσο χρόνο απόκρισης. Στο 1^ο πείραμα ο χρήστης που έστειλε αίτημα για ανάκτηση των μηνυμάτων των λογαριασμών του ήταν ένας, ενώ στα άλλα 2 πειράματα οι χρήστες που έστειλαν παράλληλα αιτήματα ήταν 5, και 10, αντίστοιχα.

Όπως ήταν αναμενόμενο, καθώς αυξανόταν ο αριθμός των χρηστών που έστειλαν παράλληλα αιτήματα, τόσο αυξανόταν ο μέσος χρόνος απόκρισης της εφαρμογής. Συγκεκριμένα, στο 1^ο πείραμα ο μέσος χρόνος, απόκρισης ήταν 1.94 δευτερόλεπτα, στο 2^ο πείραμα 2.94 δευτερόλεπτα, και στο 3^ο 4.53 δευτερόλεπτα.

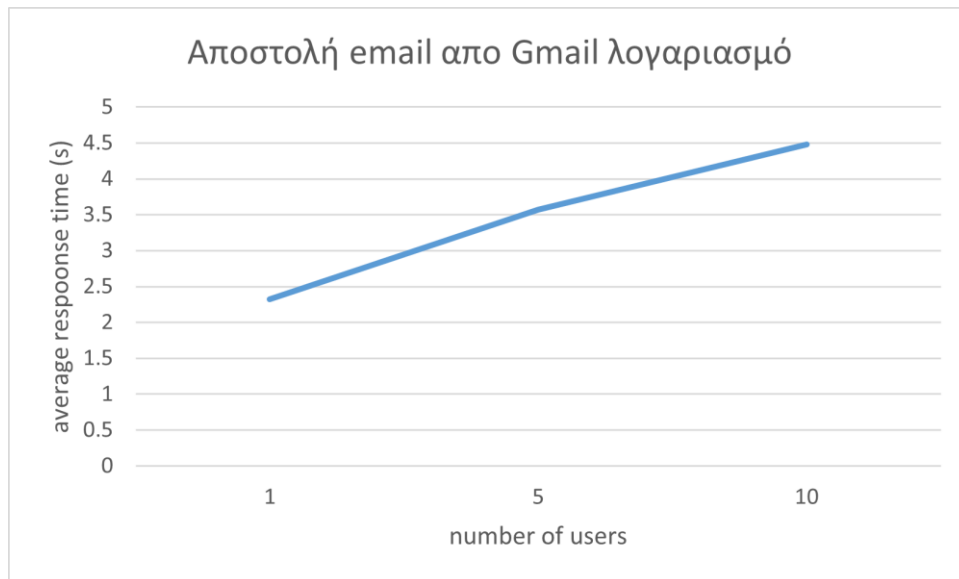


Σχήμα 6.1: Ο μέσος χρόνος απόκρισης, σε δευτερόλεπτα, για τη λειτουργία της ανάκτησης όλων των emails, και μηνυμάτων, των 3 λογαριασμών του χρήστη. Τα πειράματα που εκτελέσαμε αφορούν 1, 5, και 10 χρήστες, να στέλνουν παράλληλα αιτήματα, αντίστοιχα. Στον άξονα x έχουμε τον αριθμό των χρηστών που έστειλαν παράλληλα αιτήματα, για κάθε πείραμα, ενώ στον άξονα y έχουμε το μέσο χρόνο απόκρισης/ικανοποίησης των αιτημάτων, σε δευτερόλεπτα.

6.6.2 Αποστολή μηνύματος, από Gmail λογαριασμό του χρήστη

Στο σχήμα 6.2 παρουσιάζονται τα αποτελέσματα των πειραμάτων που κάναμε για τη λειτουργία της αποστολής ενός μηνύματος, από κάποιον από τους 2 Gmail λογαριασμούς του χρήστη. Εκτελέσαμε 3 πειράματα, κάθε πείραμα για 5 φορές, και από κάθε πείραμα κρατήσαμε το μέσο χρόνο απόκρισης, σε δευτερόλεπτα. Στο 1^ο πείραμα είχαμε 1 χρήστη που έστειλε αίτημα στην εφαρμογή, ενώ στα άλλα 2, οι χρήστες ήταν 5, και 10, αντίστοιχα.

Όπως παρατηρούμε από το σχήμα, ο μέσος χρόνος απόκρισης ανεβαίνει, σχεδόν γραμμικά, όσο αυξάνεται ο αριθμός των χρηστών, και συνεπώς των αιτημάτων. Αυτό είναι και το αναμενόμενο αποτέλεσμα. Συγκεκριμένα οι μετρήσεις έδωσαν για το 1^ο πείραμα μέσο χρόνο απόκρισης 2.32 δευτερόλεπτα, για το 2^ο πείραμα 3.57 δευτερόλεπτα, και για το 3^ο πείραμα 4.48 δευτερόλεπτα.

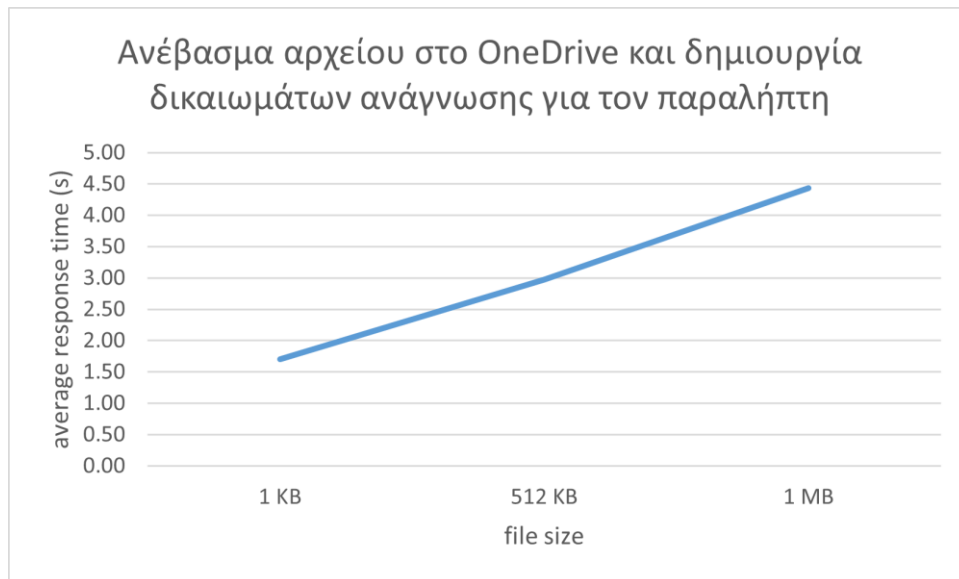


Σχήμα 6.2 Ο μέσος χρόνος απόκρισης, σε δευτερόλεπτα, για τη λειτουργία της αποστολής ενός μηνύματος, από κάποιο από τους Gmail λογαριασμούς του χρήστη. Τα πειράματα που εκτελέσαμε αφορούν 1, 5, και 10 χρήστες, να στέλνουν παράλληλα αιτήματα, αντίστοιχα. Στον άξονα x έχουμε τον αριθμό των χρηστών που έστειλαν παράλληλα αιτήματα, για κάθε πείραμα, ενώ στον άξονα y έχουμε το μέσο χρόνο απόκρισης/ικανοποίησης των αιτημάτων, σε δευτερόλεπτα.

6.6.3 Ανέβασμα αρχείου στο OneDrive του χρήστη, και δημιουργία δικαιώματος ανάγνωσης για το χρήστη για τον οποίο προορίζεται

Στο σχήμα 6.3 παρουσιάζονται τα αποτελέσματα των πειραμάτων που κάναμε για τη λειτουργία του ανεβάσματος ενός αρχείου στο OneDrive του χρήστη, και της δημιουργίας δικαιωμάτων ανάγνωσης για κάποιο λογαριασμό (ώστε να μπορεί στη συνέχεια να το μοιραστεί, μέσω μηνύματος στο Microsoft Teams). Εκτελέσαμε 3 πειράματα, κάθε πείραμα για 5 φορές, και από κάθε πείραμα κρατήσαμε το μέσο χρόνο απόκρισης, σε δευτερόλεπτα. Αυτή τη φορά, και στα 3 πειράματα είχαμε 1 χρήστη να στέλνει αίτημα στην εφαρμογή. Αυτό που άλλαζε ήταν το μέγεθος του αρχείου που θα ανέβαινε στο OneDrive. Στο 1^ο πείραμα το μέγεθος του αρχείου ήταν 1KB, ενώ στα άλλα 2, το μέγεθος ήταν 512KB, και 1MB, αντίστοιχα.

Όπως παρατηρούμε από το σχήμα, ο μέσος χρόνος απόκρισης ανεβαίνει, σχεδόν γραμμικά, όσο αυξάνεται το μέγεθος του αρχείου που ανεβαίνει στο OneDrive του χρήστη. Αυτό είναι και το αναμενόμενο αποτέλεσμα. Συγκεκριμένα οι μετρήσεις έδωσαν για το 1^ο πείραμα μέσο χρόνο απόκρισης 1.70 δευτερόλεπτα, για το 2^ο πείραμα 2.98 δευτερόλεπτα, και για το 3^ο πείραμα 4.43 δευτερόλεπτα.

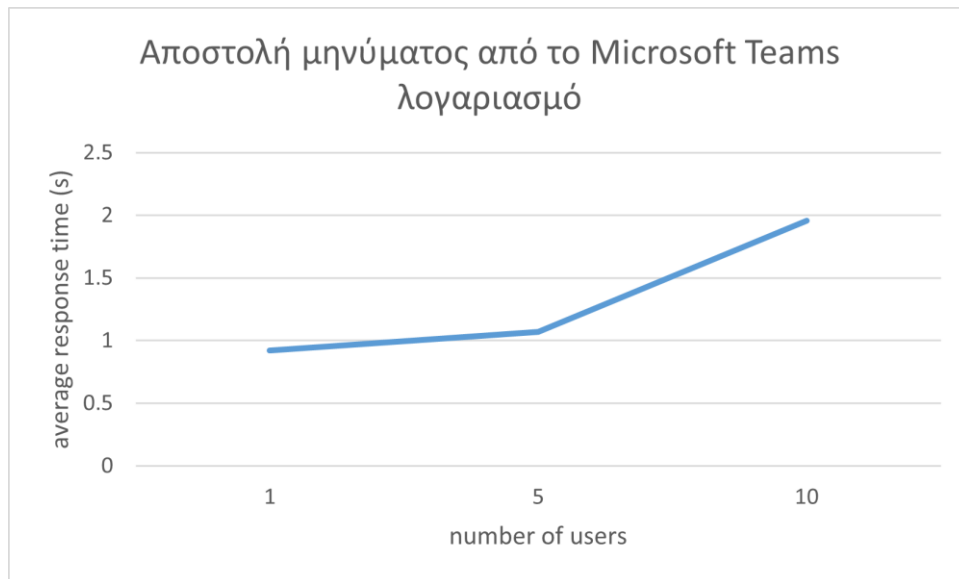


Σχήμα 6.3 Ο μέσος χρόνος απόκρισης, σε δευτερόλεπτα, για τη λειτουργία του ανεβάσματος ενός αρχείου στο OneDrive του χρήστη, και της δημιουργίας δικαιωμάτων ανάγνωσης για κάποιο λογαριασμό. Τα πειράματα που εκτελέσαμε αφορούν 1 χρήστη, να ανεβάζει αρχείο μεγέθους 1KB, 512KB, και 1MB, αντίστοιχα. Στον άξονα x έχουμε το μέγεθος του αρχείου, για κάθε πείραμα, ενώ στον άξονα y έχουμε το μέσο χρόνο απόκρισης/ικανοποίησης των αιτημάτων, σε δευτερόλεπτα.

6.6.4 Αποστολή μηνύματος από το Microsoft Teams λογαριασμό του χρήστη

Στο σχήμα 6.4 παρουσιάζονται τα αποτελέσματα των πειραμάτων που κάναμε για τη λειτουργία της αποστολής ενός μηνύματος, από το Microsoft Teams λογαριασμό του χρήστη. Εκτελέσαμε 3 πειράματα, κάθε πείραμα για 5 φορές, και από κάθε πείραμα κρατήσαμε το μέσο χρόνο απόκρισης, σε δευτερόλεπτα. Στο 1^ο πείραμα είχαμε 1 χρήστη που έστειλε αίτημα στην εφαρμογή, ενώ στα άλλα 2, οι χρήστες ήταν 5, και 10, αντίστοιχα.

Όπως παρατηρούμε από το σχήμα, ο μέσος χρόνος απόκρισης ανεβαίνει, όσο αυξάνεται ο αριθμός των χρηστών, και συνεπώς των αιτημάτων. Αυτό είναι και το αναμενόμενο αποτέλεσμα. Συγκεκριμένα οι μετρήσεις έδωσαν για το 1^ο πείραμα μέσο χρόνο απόκρισης 0.92 δευτερόλεπτα, για το 2^ο πείραμα 1.07 δευτερόλεπτα, και για το 3^ο πείραμα 1.96 δευτερόλεπτα.



Σχήμα 6.4 Ο μέσος χρόνος απόκρισης, σε δευτερόλεπτα, για τη λειτουργία της αποστολής ενός μηνύματος, από το Microsoft Teams λογαριασμό του χρήστη. Τα πειράματα που εκτελέσαμε αφορούν 1, 5, και 10 χρήστες, να στέλνουν παράλληλα αιτήματα, αντίστοιχα. Στον άξονα x έχουμε τον αριθμό των χρηστών που έστειλαν παράλληλα αιτήματα, για κάθε πείραμα, ενώ στον άξονα y έχουμε το μέσο χρόνο απόκρισης/ικανοποίησης των αιτημάτων, σε δευτερόλεπτα.

6.7 Συμπεράσματα

Από τα 4 πειράματα που κάναμε συμπεραίνουμε ότι η εφαρμογή μας έχει έναν αξιοπρεπή χρόνο απόκρισης για όλες τις λειτουργίες, όσο οι χρήστες που στέλνουν αιτήματα για την ίδια λειτουργία δεν είναι πάνω από 10, την ίδια χρονική στιγμή.

Αυτό βέβαια δεν θα έπρεπε να μας προβληματίζει, καθώς η εφαρμογή ήταν σε στάδιο ανάπτυξης, και όχι σε στάδιο παραγωγής, κατά τη διάρκεια αξιολόγησης της απόδοσής της. Εφόσον η εφαρμογή περάσει στο στάδιο παραγωγής, η απόδοσή της θα βελτιωθεί σημαντικά, καθώς τα πλαίσια λογισμικού θα κάνουν βελτιστοποιήσεις (παραλείποντας προειδοποιήσεις, κάνοντας caching σε στατικά αρχεία κ.τ.λ.) στο build τους.

Επίσης, ένας παράγοντας που προκαλούσε σημαντική αύξηση στο χρόνο απόκρισης, καθώς τα αιτήματα αυξανόταν, είναι το quota των διαδικτυακών APIs που χρησιμοποιήσαμε. Εφόσον η καταχωρημένη εφαρμογή μας (δηλαδή το backend) βρισκόταν σε στάδιο ανάπτυξης, γινόταν πολύ πιο εύκολα throttling (όρος που χρησιμοποιείται για να περιγράψει τη διαδικασία περιορισμού των αιτημάτων που μπορεί να υποβάλλει ένας πελάτης σε μια δεδομένη λειτουργία, για συγκεκριμένο χρονικό διάστημα, από ένα διακομιστή). Αυτό είναι κάτι που θα αλλάξει με το που η

εφαρμογή περάσει από το στάδιο ανάπτυξης, στους 2 παρόχους υπολογιστικής νέφους (Google Cloud, Microsoft Azure).

Τέλος, ένας σημαντικός τρόπος βελτίωσης της απόδοσης της εφαρμογής, είναι η εγκατάσταση (deployment) του build παραγωγής της στο νέφος (π.χ. AWS). Έτσι θα μπορούμε να κάνουμε αυτόματη κλιμάκωση (autoscaling), caching, να χρησιμοποιήσουμε κάποιο εξισορροπητή φορτίου (load balancer), κάποιο αντίστροφο πληρεξούσιο (reverse proxy), καλύτερη υπολογιστική ισχύ (πυρήνες, μνήμη).

Κεφάλαιο 7. Επίλογος

7.1 Σύνοψη και συμπεράσματα

Έχοντας στα χέρια μας την τελική εφαρμογή, μπορούμε εύκολα να διαπιστώσουμε ότι έχουν επιτευχθεί όλοι οι αρχικοί στόχοι που είχαμε θέσει. Ο χρήστης μπορεί να συνδέσει τους λογαριασμούς του στην εφαρμογή και να έχει πρόσβαση σε αυτούς από την ίδια γραφική διεπαφή. Η απόδοσή της είναι αρκετά καλή, ώστε να είναι προτιμότερο για το χρήστη να χρησιμοποιεί αυτή, αντί να εναλλάσσει 2 διαφορετικές εφαρμογές, και 3 διαφορετικούς λογαριασμούς. Επίσης, η γραφική διεπαφή της προσφέρει μία πολύ καλή εμπειρία χρήστη, καθώς έχει μοντελοποιηθεί με βάση το γραφικό περιβάλλον του Gmail και του Microsoft Teams. Έτσι, ο χρήστης γνωρίζει πως να υλοποιήσει τις λειτουργίες που θέλει, χωρίς να χρειάζεται να “μάθει” να χρησιμοποιεί την εφαρμογή.

7.2 Μελλοντικές επεκτάσεις

Η εφαρμογή αυτή υποστηρίζει μια σημαντική ιδέα, την ιδέα της κοινής γραφικής διεπαφής για πολλούς λογαριασμούς εφαρμογών. Στο μέλλον θα ήταν πολύ καλή ιδέα να προστεθεί υποστήριξη για λογαριασμούς εφαρμογών όπως είναι το Facebook, το Discord, και το Twitter, μεταξύ άλλων. Επίσης, θα μπορούσε να προστεθεί υποστήριξη και για άλλους email clients, όπως το Outlook, το Yahoo, κ.τ.λ..

Παράρτημα Α. Web APIs reference

A.1 Τα endpoints των διαδικτυακών APIs που αξιοποιήσαμε

Σε αυτό το παράρτημα περιγράφονται αναλυτικά τα endpoints των διαδικτυακών APIs που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής μας.

A.1.1 Gmail API endpoints

Για να μπορέσει η εφαρμογή μας να διαβάζει, και να στέλνει, μηνύματα, από, και προς, το Gmail λογαριασμό του χρήστη, θα πρέπει να χρησιμοποιήσουμε το Gmail API [15]. Το Gmail API είναι ένα RESTful διαδικτυακό API, το οποίο έχει υλοποιήσει η Google, και μπορεί να χρησιμοποιηθεί για πρόσβαση στο Gmail γραμματοκιβώτιο του χρήστη, καθώς και για αποστολή email. Το entry point του API είναι το “https://gmail.googleapis.com/gmail/v1”. Καθώς δεν θα κάνουμε απευθείας αιτήματα στο Gmail API, για να στείλουμε ένα καινούριο email (για αυτή τη δουλειά θα χρησιμοποιήσουμε το πακέτο nodemailer, όπως θα δούμε στο κεφάλαιο της υλοποίησης), τα αιτήματα, τα οποία θα μας προσφέρουν τη λειτουργικότητα που ψάχνουμε, είναι τα ακόλουθα:

Περιγραφή	Για ανάκτηση όλων των αδιάβαστων μηνυμάτων από το Gmail λογαριασμό του χρήστη
Μονοπάτι	/users/{userId}/messages
HTTP μέθοδος	GET
Παράμετροι ερωτήματος	q=is:unread

Πίνακας A.1 HTTP αίτημα για ανάκτηση όλων των αδιάβαστων μηνυμάτων από το Gmail λογαριασμό του χρήστη.

Περιγραφή	Για ανάκτηση ενός αδιάβαστου μηνύματος από το Gmail λογαριασμό του χρήστη, με βάση το id του
Μονοπάτι	/users/{userId}/messages/{id}
HTTP μέθοδος	GET

Πίνακας A.2 HTTP αίτημα για ανάκτηση ενός αδιάβαστου μηνύματος από το Gmail λογαριασμό του χρήστη.

Περιγραφή	Για ανάκτηση ενός συνημμένου, κάποιου αδιάβαστου μηνύματος, από το Gmail λογαριασμό του χρήστη, με βάση το id του
Μονοπάτι	/users/{userId}/messages/{messageId}/attachments/{id}
HTTP μέθοδος	GET

Πίνακας A.3 HTTP αίτημα για ανάκτηση ενός συνημμένου, από ένα μήνυμα από το Gmail λογαριασμό του χρήστη.

Περιγραφή	Για επισήμανση ενός μηνύματος, από το Gmail λογαριασμό του χρήστη, ως διαβασμένου
Μονοπάτι	/users/{userId}/messages/{id}/modify
HTTP μέθοδος	POST
Ωφέλιμο φορτίο (JSON)	{ "addLabelIds": [], "removeLabelIds": ["UNREAD"] }

Πίνακας A.4 HTTP αίτημα για επισήμανση ενός μηνύματος, από το Gmail λογαριασμό του χρήστη, ως διαβασμένου.

A.1.1.1 Βελτιωμένη οργάνωση του γραμματοκιβώτιου του χρήστη

Κατά τη διάρκεια ανάπτυξης της εφαρμογής, παρότι αρχικά η εφαρμογή λειτουργούσε με μηνύματα από το λογαριασμό του χρήστη παρατηρούμε ότι το Gmail API δίνει τη επιλογή ανάκτησης των μηνυμάτων του γραμματοκιβώτιου, ομαδοποιημένα σε νήματα, κάτι που μας δίνει τη δυνατότητα για εύκολη οργάνωση των μηνυμάτων σε συνομιλίες, και, συνεπώς, μία καλύτερη εμπειρία χρήστη. Τα endpoints που θα αξιοποιήσουμε είναι τα εξής:

Περιγραφή	Για ανάκτηση των ids όλων των νημάτων, από το Gmail λογαριασμό του χρήστη, με βάση κάποιο label id
Μονοπάτι	/users/{userId}/threads
HTTP μέθοδος	GET
Παράμετροι ερωτήματος	q=is:unread (για ανάκτηση των νημάτων που περιέχουν αδιάβαστα μηνύματα), q=is:starred (για ανάκτηση των νημάτων που περιέχουν μηνύματα που έχουν επισημανθεί ως σημαντικά)

Πίνακας Α.5 HTTP αίτημα για ανάκτηση των αναγνωριστικών όλων των νημάτων, από το Gmail λογαριασμό του χρήστη.

Περιγραφή	Για ανάκτηση ενός νήματος, από το Gmail λογαριασμό του χρήστη
Μονοπάτι	/users/{userId}/threads/{id}
HTTP μέθοδος	GET

Πίνακας Α.6 HTTP αίτημα για ανάκτηση ενός νήματος, από το Gmail λογαριασμό του χρήστη.

Περιγραφή	Για επισήμανση, ή κατάργηση επισήμανσης, ενός νήματος, από το Gmail λογαριασμό του χρήστη, ως διαβασμένου, ή ως σημαντικού
Μονοπάτι	/users/{userId}/threads/{id}/modify
HTTP μέθοδος	POST
Ωφέλιμο φορτίο (JSON, 1 από τις 4 επιλογές, αναλόγως το use case)	<p>{"addLabelIds": ["UNREAD"]} (για χαρακτηρισμό ενός νήματος ως αδιάβαστου),</p> <p>{"removeLabelIds": ["UNREAD"]} (για χαρακτηρισμό ενός νήματος ως διαβασμένου),</p> <p>{"addLabelIds": ["STARRED"]} (για χαρακτηρισμό ενός νήματος ως σημαντικού),</p> <p>{"removeLabelIds": ["STARRED"]} (για χαρακτηρισμό ενός νήματος ως μη σημαντικού)</p>

A.1.1.2 Κεφαλίδες αυθεντικοποίησης

Κάθε αίτημα για να ικανοποιηθεί, και να μην πάρει μήνυμα σφάλματος μη εξουσιοδότησης, από το API, θα πρέπει στις κεφαλίδες του να περιέχει την κεφαλίδα: `"Authorization: Bearer {accessToken}"`. Τα scopes που θα μας επιτρέψουν να πάρουμε access tokens που θα μας επιτρέψουν την επιτυχή απόκριση από τα endpoints που αναφέραμε είναι τα `"https://www.googleapis.com/auth/userinfo.email"` (για να ανακτήσουμε τη διεύθυνση του χρήστη στο Gmail), `"https://mail.google.com/"` (για να μπορούμε να διαβάζουμε και να στέλνουμε μηνύματα από το λογαριασμό του χρήστη). Συνεπώς θα πρέπει να τα ζητήσουμε κατά διάρκεια του πρωτοκόλλου OAuth 2.0 μεταξύ Google, εφαρμογής, και χρήστη. Επίσης, θα πρέπει να ζητήσουμε offline access, ώστε να λαμβάνουμε refresh tokens, τα οποία θα μας επιτρέψουν να τα χρησιμοποιούμε ώστε να αποκτήμε καινούρια access tokens, όταν αυτά γίνονται ληγμένα. Έτσι, δε θα χρειάζεται κάθε φορά που το access token, για το λογαριασμό ενός χρήστη, γίνεται ληγμένο, αυτός να εισάγει, ξανά, τα διαπιστευτήριά του (δηλαδή να ξαναγίνεται η διαδικασία αυθεντικοποίησής του, καθώς και η διαδικασία εξουσιοδότησης της εφαρμογής για πρόσβαση στους Gmail λογαριασμούς του χρήστη).

A.1.2 Microsoft Graph API endpoints

Για να μπορεί η εφαρμογή μας να έχει πρόσβαση στα chats του χρήστη, καθώς και να δημιουργεί καινούρια, αλλά και να στέλνει μηνύματα, σε αυτά, θα πρέπει να χρησιμοποιήσουμε το Microsoft Graph API. Το Microsoft Graph API είναι, και αυτό, ένα RESTful διαδικτυακό API, που έχει υλοποιήσει η Microsoft, το entry point του είναι το `"https://graph.microsoft.com/v1.0"`, και τα αιτήματα που θα χρειαστεί να του κάνουμε, είναι τα ακόλουθα:

Περιγραφή	Για ανάκτηση όλων των chats (χωρίς τα μηνύματά τους), στα οποία ο χρήστης είναι μέλος, μαζί με όλα τα μέλη που συμμετέχουν
Μονοπάτι	<code>/users/{user-id}/chats</code>
HTTP μέθοδος	GET
Παράμετροι ερωτήματος	<code>\$expand=members</code>

Πίνακας A.8 HTTP αίτημα για ανάκτηση των αναγνωριστικών όλων των chats, για το Microsoft Teams λογαριασμό του χρήστη, στα οποία ο χρήστης είναι μέλος, μαζί με όλα τα μέλη που συμμετέχουν.

Περιγραφή	Για ανάκτηση όλων των μηνυμάτων ενός chat, με βάση το id του
Μονοπάτι	/users/{user-id}/chats/{chatid}/messages
HTTP μέθοδος	GET

Πίνακας A.9 HTTP αίτημα για ανάκτηση όλων των μηνυμάτων ενός chat, για το Microsoft Teams λογαριασμό του χρήστη.

Περιγραφή	Για αποστολή ενός καινούριου μηνύματος σε κάποιο chat, στο οποίο ο χρήστης είναι μέλος
Μονοπάτι	/chats/{chat-id}/messages
HTTP μέθοδος	POST
Ωφέλιμο φορτίο (JSON)	chatMessage (Microsoft Graph resource)

Πίνακας A.10 HTTP αίτημα για αποστολή ενός καινούριου μηνύματος σε κάποιο chat στο Microsoft Teams, στο οποίο ο χρήστης είναι μέλος.

Περιγραφή	Για ανέβασμα ενός αρχείου στο OneDrive του χρήστη (κάτι που χρειαζόμαστε ώστε να μπορεί στη συνέχεια ο χρήστης να στείλει συνημμένα σε κάποιο chat, καθώς το απαιτεί το Microsoft Graph API)
Μονοπάτι	/me/drive/items/{item-id}/content
HTTP μέθοδος	PUT
Ωφέλιμο φορτίο (binary stream)	file contents

Πίνακας A.11 HTTP αίτημα για ανέβασμα ενός αρχείου στο OneDrive του χρήστη.

Περιγραφή	Για να δώσουμε δικαιώματα ανάγνωσης σε κάποιον χρήστη (ώστε να μπορεί να δει ένα αρχείο/συνημμένο που του έχουμε στείλει στο chat, πρέπει να του δώσουμε δικαιώματα ανάγνωσης για το αρχείο/συνημμένο αυτό, στο OneDrive)
Μονοπάτι	/me/drive/items/{item-id}/invite

HTTP μέθοδος	POST
Ωφέλιμο φορτίο (JSON)	<pre>{ "requireSignIn": true, "sendInvitation": false, "roles": ["read"], "recipients": [{ "email": {emailToGivePermissionsTo} }] }</pre>

Πίνακας A.12 HTTP αίτημα για δημιουργία δικαιωμάτων ανάγνωσης σε κάποιον χρήστη, για κάποιο αρχείο στο OneDrive του χρήστη της εφαρμογής.

Περιγραφή	Για να ανακτήσουμε το όνομα χρήστη ενός χρήστη με βάση τη διεύθυνση email του, καθώς και να πιστοποιήσουμε ότι ο χρήστης αυτός υπάρχει (αυτό το χρειαζόμαστε ώστε να μπορούμε να δημιουργήσουμε ένα καινούριο chat με βάση τη διεύθυνση email ενός χρήστη)
Μονοπάτι	/users/{id}
HTTP μέθοδος	GET

Πίνακας A.13 HTTP αίτημα για ανάκτηση του ονόματος χρήστη ενός χρήστη του Microsoft Teams, με βάση τη διεύθυνση email του.

Περιγραφή	Για να δημιουργήσουμε ένα καινούριο chat μεταξύ 2 χρηστών
Μονοπάτι	/chats
HTTP μέθοδος	POST
Ωφέλιμο φορτίο (JSON, πρακτικά η πληροφορία που στέλνουμε είναι ότι το chat θέλουμε να είναι one on one, καθώς και οι διευθύνσεις	<pre>{ "chatType": "oneOnOne", "members": [{ "@odata.type": "#microsoft.graph.aadUserConversationMember", </pre>

email των 2 χρηστών, από τους οποίους ο ένας είναι ο χρήστης της εφαρμογής, για τους οποίους θέλουμε να δημιουργηθεί το chat)	<pre> "roles": ["owner"], "user@odata.bind": `https://graph.microsoft.com/v1.0/users('{userId1}')` }, { "@odata.type": "#microsoft.graph.aadUserConversationMember", "roles": ["owner"], "user@odata.bind": `https://graph.microsoft.com/v1.0/users('{userId2 }')` }} } </pre>
--	--

Πίνακας A.14 HTTP αίτημα για δημιουργία ενός καινούριου chat μεταξύ 2 χρηστών στο Microsoft Teams.

Για να έχουμε πρόσβαση στις υπηρεσίες αυτές, θα χρειαστεί, κατά τη διάρκεια του πρωτοκόλλου εξουσιοδότησης OAuth 2.0, μεταξύ της Microsoft, της εφαρμογής, και του χρήστη, να ζητήσουμε τα εξής scopes: "offline_access", "user.read", "Chat.ReadWrite", "Files.ReadWrite.All", "User.ReadBasic.All".

Βιβλιογραφία

- [1] [Ηλεκτρονικό]. Available: <https://www.ibm.com/cloud/learn/three-tier-architecture>.
- [2] [Ηλεκτρονικό]. Available: <https://www.geeksforgeeks.org/advantages-and-disadvantages-of-three-tier-architecture-in-dbms/>.
- [3] [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Client%E2%80%93server_model.
- [4] [Ηλεκτρονικό]. Available: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>.
- [5] [Ηλεκτρονικό]. Available: <https://www.redhat.com/en/topics/api/what-is-a-rest-api#whats-an-api>.
- [6] [Ηλεκτρονικό]. Available: <https://www.redhat.com/en/topics/api/what-is-a-rest-api#rest>.
- [7] [Ηλεκτρονικό]. Available: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>.
- [8] [Ηλεκτρονικό]. Available: <https://www.geeksforgeeks.org/software-framework-vs-library/>.
- [9] [Ηλεκτρονικό]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction.
- [10] [Ηλεκτρονικό]. Available: <https://reactjs.org/>.
- [11] [Ηλεκτρονικό]. Available: <https://nodejs.org/en/about/>.
- [12] [Ηλεκτρονικό]. Available: <https://expressjs.com/>.

- [13] [Ηλεκτρονικό]. Available: <https://www.mongodb.com/>.
- [14] [Ηλεκτρονικό]. Available: <https://www.mongodb.com/mern-stack>.
- [15] [Ηλεκτρονικό]. Available: <https://developers.google.com/gmail/api>.
- [16] [Ηλεκτρονικό]. Available: <https://docs.microsoft.com/en-us/graph/use-the-api>.
- [17] [Ηλεκτρονικό]. Available: <https://auth0.com/docs/authenticate/protocols/oauth>.
- [18] [Ηλεκτρονικό]. Available: <https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2>.
- [19] [Ηλεκτρονικό]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies>.
- [20] [Ηλεκτρονικό]. Available: <https://www.geeksforgeeks.org/session-vs-token-based-authentication/>.
- [21] [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Bcrypt#Comparison_to_other_password_hashing_algorithms.
- [22] [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard.
- [23] [Ηλεκτρονικό]. Available: [https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher_block_chaining_\(CBC\)](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher_block_chaining_(CBC)).