

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΕΝΟΠΟΙΗΜΕΝΗ ΠΡΟΣΒΑΣΗ ΣΕ ΣΥΣΤΗΜΑΤΑ ΑΠΟΘΗΚΕΥΣΗΣ ΝΕΦΟΥΣ

Γιώτης Κωνσταντίνος

Διπλωματική Εργασία

Επιβλέπων: Π. Τσαπάρας

Ιωάννινα, Αύγουστος 2020



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

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

Ευχαριστίες

Σε αυτό το σημείο, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Π. Τσαπάρα για την πολύτιμή του βοήθεια και καθοδήγηση καθ'όλη την διάρκεια εκπόνησης της διπλωματικής μου εργασίας, καθώς και για την υπέροχη ιδέα του θέματος της διπλωματικής αυτής εργασίας.

Αύγουστος 2020

Γιώτης Κωνσταντίνος

Περίληψη

Τα τελευταία χρόνια όλο και περισσότεροι χρήστες του διαδικτύου χρησιμοποιούν τα μέσα αποθήκευσης νέφους για την αποθήκευση αρχείων στο νέφος. Οι λόγοι που ωθούν τους χρήστες σε μία τέτοια επιλογή, σχετίζονται με την ευκολία πρόσβασης στα αρχεία τους από οποιαδήποτε συσκευή στην οποία βρίσκονται συνδεδεμένοι, καθώς και για την ασφάλεια που παρέχουν αυτά τα μέσα σε σχέση με τις κοινές συσκευές αποθήκευσης που μπορούν ανά πάσα στιγμή να πάψουν να λειτουργούν. Τα μέσα αποθήκευσης νέφους, όμως, παρέχουν περιορισμένο δωρεάν ελεύθερο χώρο χωρίς χρηματικές δεσμεύσεις. Ο σκοπός της παρούσας διπλωματικής είναι να αντιμετωπίσουμε αυτό το πρόβλημα μέσω της υλοποίησης μιας εφαρμογής διαδικτύου, που θα ενοποιεί τα διαφορετικά μέσα αποθήκευσης νέφους σε μία ενιαία γραφική διεπαφή, χρησιμοποιώντας πολλαπλούς λογαριασμούς με δωρεάν ελεύθερο χώρο. Ο τελικός στόχος είναι τελικά η παροχή μεγαλύτερου ελεύθερου χώρου για αποθήκευση αρχείων στο νέφος, χωρίς να χρειάζεται να επενδύουμε επιπλέον χρόνο για την διαχείριση των πολλαπλών διαφορετικών λογαριασμών.

Λέξεις Κλειδιά: Μέσα αποθήκευσης νέφους, Ενοποίηση, Διαδικτυακή εφαρμογή

Abstract

Over the last few years, more and more Internet users are starting to use the cloud drive services to store their files in the cloud. The reasons that drive users to this selection, are related to the ease of access to their files from many different devices to which they are connected as well as the security provided by these cloud drives, in comparison with the common storage devices that may stop working at any given time. However, these cloud drive services, only provide a limited amount of free storage space without subscriptions. The purpose of this thesis is to face this problem through the implementation of a web application that merges the different cloud drive services in one single user interface, by using multiple user accounts with free storage space. This implementation will provide us with additional storage space for storing files in the cloud, without investing additional time in the management of the multiple different user accounts we own.

Keywords: Cloud storage services, Merging, Web application

Πίνακας περιεχομένων

Κεφάλαιο 1. Εισαγωγή.....	1
1.1 Αντικείμενο της διπλωματικής.....	1
1.2 Οργάνωση του τόμου.....	2
Κεφάλαιο 2. Περιγραφή Θέματος.....	4
2.1 Στόχος της εργασίας.....	4
2.2 Ανάλυση απαιτήσεων	4
2.3 Περιγραφή των απαιτήσεων με Use Cases	6
2.4 Παρόμοιες εφαρμογές.....	15
Κεφάλαιο 3. Τεχνολογικό Υπόβαθρο	17
3.1 Εισαγωγή.....	17
3.2 Ορολογία.....	17
3.3 Η στοίβα MERN.....	18
3.4 Το framework Node.js.....	20
3.5 Το framework Express.js	21
3.6 Η βιβλιοθήκη ReactJS	22
3.7 Η βάση δεδομένων MongoDB	23
3.8 Η αρχιτεκτονική REST και το REST-ful API [7][8]	26
Κεφάλαιο 4. Σχεδίαση & Υλοποίηση.....	30
4.1 Σχεδίαση και αρχιτεκτονική λογισμικού.....	30
4.1.1 Αρχιτεκτονική της εφαρμογής [9].....	30
4.1.2 Σύστημα αυθεντικοποίησης χρήστη.....	33
4.1.3 Επικοινωνία με τα μέσα αποθήκευσης νέφους	34
4.1.4 Εικονικό σύστημα αποθήκευσης νέφους.....	37
4.1.5 Ανάλυση σχήματος βάσης δεδομένων.....	49
4.2 Ασφάλεια.....	53

Κεφάλαιο 5. Τεχνικές Λεπτομέρειες	55
5.1 Επεκτασιμότητα του λογισμικού	55
5.2 Πληροφορίες εγκατάστασης.....	58
5.3 Επιπλέον βιβλιοθήκες	58
5.4 Οδηγός του API της εφαρμογής.....	60
Κεφάλαιο 6. Γραφική διεπαφή της εφαρμογής	69
6.1 Παρουσίαση του εικονικού συστήματος αποθήκευσης νέφους.....	69
6.1.1 Σελίδες της εφαρμογής.....	69
6.1.2 Εκτέλεση λειτουργιών.....	71
6.2 Επιπλέον λειτουργικότητα: Η “All-In-One” Διεπαφή.....	77
Κεφάλαιο 7. Επίλογος.....	79
7.1 Σύνοψη	79
7.2 Μελλοντικές επεκτάσεις	80

Κεφάλαιο 1. Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Τα τελευταία χρόνια, τα μέσα αποθήκευσης δεδομένων νέφους αποκτούνε όλο και περισσότερη δημοφιλία. Πάρα πολλοί χρήστες προτιμούν την αποθήκευση αρχείων στο νέφος αντί των κοινών συσκευών αποθήκευσης (π.χ. HDDs, USBs, CDs), για λόγους ευκολίας πρόσβασης των αρχείων από πολλαπλές συσκευές καθώς και για λόγους ασφάλειας. Παρόλη την αυξανόμενη δημοφιλία τους, η χρήση των μέσων αποθήκευσης νέφους από μερικούς χρήστες, συνοδεύεται από κάποια προβλήματα.

Το πιο σημαντικό πρόβλημα, είναι η παροχή περιορισμένου ελεύθερου χώρου για δωρεάν αποθήκευση αρχείων. Οι περισσότεροι χρήστες δεν είναι διατεθειμένοι να δεσμευτούν με χρηματικές συνδρομές, με αποτέλεσμα να δημιουργούν πολλαπλούς διαφορετικούς λογαριασμούς για επιπλέον δωρεάν ελεύθερο χώρο. Το πρόβλημα αυτό που μόλις αναφέραμε, δημιουργεί ακόμη ένα πρόβλημα το οποίο σχετίζεται με την διαχείριση των πολλαπλών αυτών λογαριασμών. Οι πολλαπλοί αυτοί λογαριασμοί, δημιουργούν μία σύγχυση στον χρήστη εφόσον θα πρέπει να κάνει συχνά εναλλαγές στους επιμέρους λογαριασμούς του για την πρόσβαση των αρχείων του, γεγονός που οδηγεί σε ένα αρκετά μεγάλο κόστος διαχείρισης αυτών των λογαριασμών.

Το αντικείμενο της διπλωματικής εργασίας, σχετίζεται με την ανάπτυξη μιας διαδικτυακής εφαρμογής που θα χρησιμοποιεί τα διαφορετικά μέσα αποθήκευσης νέφους κάτω από μία κοινή και ενοποιημένη γραφική διεπαφή με σκοπό να λυθούν τα προβλήματα που αναφέραμε. Πιο συγκεκριμένα, θα παρέχεται στον χρήστη η δυνατότητα να προσθέσει ένα από τα τρία πιο γνωστά μέσα αποθήκευσης νέφους, “Google Drive”, “OneDrive” και “Dropbox” αρχικά και να διαχειρίζεται τον χώρο που αυτά

που προσφέρουν σαν να είναι ένας ενιαίος χώρος, αποκρύπτοντας του όλη την λογική της υλοποίησης.

Συνοπτικά, μέσω της εφαρμογής, ο χρήστης θα μπορεί να συνδέει πολλαπλούς λογαριασμούς μέσω αποθήκευσης νέφους και να έχει ως ελεύθερο χώρο το άθροισμα των επιμέρους αυτών λογαριασμών, χρησιμοποιώντας το σαν να είναι ένα ενιαίο μέσο αποθήκευσης. Κατά την διάρκεια ανεβάσματος ενός αρχείου, η εφαρμογή θα αποφασίζει στο παρασκήνιο σε ποιο από τα συνδεδεμένα μέσα αποθήκευσης νέφους θα αποθηκευτεί το αρχείο, αποκρύπτοντας εξ' ολοκλήρου αυτή την διαδικασία από τον χρήστη.

Μέσω αυτού του λεγόμενου «εικονικού συστήματος νέφους» που συνθέτει η εφαρμογή, ο χρήστης θα έχει την δυνατότητα να εκτελεί ορισμένες λειτουργίες διαχείρισης των αρχείων που ανέβασε όπως κατέβασμα, μετονομασία, διαγραφή και διαμοιρασμό. Επίσης ο χρήστης θα μπορεί να δημιουργεί «εικονικούς» φακέλους οι οποίοι θα περιέχουν αρχεία διαφορετικών μέσων αποθήκευσης. Επιπλέον, θα μπορεί να διαμοιράζεται αυτούς τους φακέλους με άλλους εγγεγραμμένους -στην εφαρμογή μας- χρήστες με σκοπό την κοινοχρησία των περιεχομένων τους. Τέλος, θα παρέχεται υποδομή για την προσθήκη νέων μέσων αποθήκευσης νέφους που ενδεχομένως θα αποκτήσουν δημοφιλία τα χρόνια που θα ακολουθήσουν.

1.2 Οργάνωση του τόμου

Το γραπτό μέρος της διπλωματικής εργασίας, αποτελείται από έξι συνολικά κεφάλαια, όπου το κάθε ένα καλύπτει ένα ανεξάρτητο θεματικό μέρος.

Στο πρώτο κεφάλαιο, αναφέρουμε συνοπτικά το θέμα της διπλωματικής και τους λόγους που μας ώθησαν στην εκπόνηση της.

Στο δεύτερο κεφάλαιο, αναλύουμε τους στόχους που θέσαμε πριν την έναρξη εκπόνησης της διπλωματικής εργασίας και αναλύουμε τις απαιτήσεις που θα πρέπει να πληροί η υλοποίηση της.

Στο τρίτο κεφάλαιο εξηγούμε αρχικά κάποιους χρήσιμους όρους, που είναι απαραίτητοι για την κατανόηση των κεφαλαίων που ακολουθούν. Στην συνέχεια, αναλύουμε εκτενώς τα εργαλεία που χρησιμοποιήσαμε για την ανάπτυξη της διαδικτυακής μας εφαρμογής, παρέχοντας σχήματα και σύντομα κομμάτια κώδικα για την κατανόηση τους καθώς εξηγούμε επίσης και τους λόγους που μας ώθησαν στην επιλογή τους.

Το τέταρτο κεφάλαιο, αφορά τον τρόπο που σχεδιάσαμε και υλοποιήσαμε την εφαρμογή. Εξηγούμε αρχικά την αρχιτεκτονική στην οποία βασιστήκαμε για την υλοποίηση της και στην συνέχεια αναλύουμε την λογική πίσω από κάθε λειτουργία και δυνατότητα της εφαρμογής. Έπειτα, αναφέρουμε τον τρόπο που καθιστά την εφαρμογή επεκτάσιμη και κάνουμε μία αναφορά στις τεχνικές ασφαλείας που χρησιμοποιήσαμε. Κλείνοντας το τέταρτο κεφάλαιο, εξηγούμε τον τρόπο εκκίνησης της εφαρμογής καθώς και τις επιπλέον βοηθητικές βιβλιοθήκες που χρησιμοποιήσαμε.

Στο πέμπτο κεφάλαιο, κάνουμε μία σύντομη παρουσίαση των λειτουργιών της εφαρμογής. Εξηγούμε αρχικά τις σελίδες που απαρτίζουν την εφαρμογή μας, και στην συνέχεια εξηγούμε με στιγμιότυπα πώς εκτελούνται οι λειτουργίες αυτές.

Στο έκτο κεφάλαιο, συνοψίζουμε τον στόχο και την επίτευξη της διπλωματικής εργασίας και παρέχουμε μερικές προτάσεις για μελλοντικές επεκτάσεις που θα μπορούσαν να υλοποιηθούν στην παρούσα εφαρμογή.

Κεφάλαιο 2.

Περιγραφή

Θέματος

2.1 Στόχος της εργασίας

Είναι γνωστό, ότι ο δωρεάν ελεύθερος χώρος που παρέχουν τα μέσα αποθήκευσης νέφους είναι περιορισμένος. Η ανάπτυξη αυτής της διαδικτυακής εφαρμογής σκοπεύει να λύσει αυτό το πρόβλημα, παρέχοντας επιπλέον ελεύθερο χώρο για αποθήκευση αρχείων στο νέφος, μέσω της προσθήκης και ενοποίησης πολλαπλών τέτοιων μέσων χρησιμοποιώντας απλώς διαφορετικούς λογαριασμούς χρήστη. Για την επίτευξη αυτού του στόχου, επιλέξαμε τις πλέον σύγχρονες τεχνολογίες διαδικτύου όσον αφορά την ανάπτυξη της εφαρμογής καθώς και εκσυγχρονισμένες τεχνικές σχεδιασμού της γραφικής διεπαφής για την ομαλή εμπειρία χρήστη.

2.2 Ανάλυση απαιτήσεων

Σε αυτή την ενότητα θα αναλύσουμε τις απαιτήσεις που θα πρέπει να πληροί η διαδικτυακή μας εφαρμογή όσον αφορά τις λειτουργίες που θα προσφέρει, την ομαλή εμπειρία του χρήστη, καθώς και το κομμάτι της ασφάλειας. Σε αυτή την ενότητα θα αναλύσουμε αυτές τις απαιτήσεις.

Λειτουργικές απαιτήσεις

Η διαδικτυακή μας εφαρμογή θα πρέπει:

1. Να προσφέρει στον χρήστη την δυνατότητα να κάνει εγγραφή στην εφαρμογή χρησιμοποιώντας την διεύθυνση ηλεκτρονικού ταχυδρομείου έτσι ώστε να συνδέεται και να αποσυνδέεται όποτε το επιθυμεί.

2. Να προσφέρει στον χρήστη την δυνατότητα να προσθέτει και να αφαιρεί όσα μέσα αποθήκευσης νέφους επιθυμεί, χρησιμοποιώντας διαφορετικούς λογαριασμούς με αποτέλεσμα να έχει ελεύθερο χώρο για αποθήκευση αρχείων όσο είναι και το άθροισμα των επιμέρους μέσων αποθήκευσης νέφους.
3. Να υποστηρίζει την σύνδεση των πιο γνωστών μέσων αποθήκευσης νέφους όπως το “Google Drive”, το “OneDrive” και το “Dropbox” αρχικά.
4. Να προσφέρει στον χρήστη την δυνατότητα να βλέπει, να ανεβάζει και να κατεβάζει τα αρχεία των συνδεδεμένων μέσων αποθήκευσης νέφους. Κατά την διαδικασία του ανεβάσματος θα πρέπει η εφαρμογή να βρίσκει το κατάλληλο μέσο αποθήκευσης νέφους στο οποίο θα σωθεί τελικά το αρχείο, αναλόγως τον ελεύθερο χώρο χωρίς να εμπλέκεται ο χρήστης σε αυτή την διαδικασία.
5. Να προσφέρει στον χρήστη την δυνατότητα να δημιουργεί «εικονικούς» φακέλους, οι οποίοι θα περιέχουν αρχεία από οποιοδήποτε συνδεδεμένο μέσο αποθήκευσης νέφους. Θα πρέπει δηλαδή αυτοί οι φάκελοι να μπορούν να περιέχουν αρχεία τα οποία ανήκουν είτε στο “Google Drive”, είτε στο “OneDrive”, είτε στο “Dropbox”.
6. Να προσφέρει στον χρήστη την δυνατότητα να διαμοιράζεται τους εικονικούς φακέλους με άλλους χρήστες που βρίσκονται εγγεγραμμένοι στην εφαρμογή. Σε αυτούς τους φακέλους, θα μπορούν να ανεβάζουν, να κατεβάζουν και να ανοίγουν τα αρχεία όλοι οι χρήστες που συμμετέχουν στον διαμοιρασμό.

Απαιτήσεις ασφάλειας

Η διαδικτυακή μας εφαρμογή, θα πρέπει επίσης να προστατεύει σε ικανοποιητικό βαθμό τα στοιχεία του χρήστη μέσω τεχνικών κρυπτογραφίας και κατακερματισμού για την ασφαλή αποθήκευση των προσωπικών και ευαίσθητων πληροφοριών στην βάση δεδομένων. Πιο συγκεκριμένα, σε περίπτωση παραβίασης της βάσης δεδομένων από κακόβουλους χρήστες, τα στοιχεία που πρόκειται να διαρρεύσουν θα πρέπει να μην μπορούν να χρησιμοποιηθούν για την υποκλοπή δεδομένων.

Επιπλέον, η μεταφορά των δεδομένων μεταξύ πελάτη και εξυπηρετητή, θα πρέπει γίνεται μέσω ασφαλούς καναλιού έτσι ώστε τυχόν ενδιάμεσοι κακόβουλοι χρήστες να μην μπορούν να υποκλέψουν ευαίσθητες πληροφορίες.

Απαιτήσεις εμπειρίας χρήστη

Μία ακόμη σημαντική απαίτηση, είναι η ομαλή εμπειρία του χρήστη κατά την χρήση της εφαρμογής. Ο χρήστης δε θα πρέπει να συναντά δυσκολία ως προς την εκτέλεση των υποστηριζόμενων λειτουργιών της εφαρμογής, καθώς θα πρέπει επίσης να έχει πλήρη επίγνωση της κατάστασης της εφαρμογής κατά την ολοκλήρωση ή αποτυχία μίας λειτουργίας μέσω εμφάνισης διαδραστικών μηνυμάτων.

Επιπλέον, όλη η λογική της ενοποίησης των διαφορετικών μέσων αποθήκευσης νέφους, θα πρέπει να είναι πλήρως αποκρυμμένη από τον χρήστη. Παρόλο που στο παρασκήνιο θα γίνεται επικοινωνία με τα διαφορετικά μέσα αποθήκευσης νέφους, ο χρήστης θα πρέπει να έχει την αίσθηση ότι διαχειρίζεται μόνο ένα μοναδικό και ενοποιημένο μέσο αποθήκευσης νέφους.

2.3 Περιγραφή των απαιτήσεων με Use Cases

Σε αυτή την ενότητα, θα περιγράψουμε αναλυτικά τις λειτουργικές απαιτήσεις και τα σενάρια χρήσης της εφαρμογής με την μορφή των Use Cases.

ID: 1
Use Case: Δημιουργία λογαριασμού
Actors: Χρήστης, Φυλλομετρητής
Preconditions: -
Flow of Events: <ol style="list-style-type: none">1. Το Use Case ξεκινάει όταν ο χρήστης επισκεφθεί την εφαρμογή2. Ο χρήστης εισάγει την διεύθυνση ηλεκτρονικού ταχυδρομείου και έναν κωδικό πρόσβασης στο πλαίσιο εγγραφής που εμφανίζεται στον φυλλομετρητή

<p>3. Αν η διεύθυνση ηλεκτρονικού ταχυδρομείου είναι έγκυρη ως προς την μορφή της και δεν αντιστοιχεί ήδη σε κάποιον εγγεγραμμένο χρήστη</p> <p>3.1 Η εφαρμογή εμφανίζει μήνυμα επιτυχίας δημιουργίας του λογαριασμού “Sign up successful! You can now log in”</p> <p>4. Αν η διεύθυνση ηλεκτρονικού ταχυδρομείου δεν είναι έγκυρη ως προς την μορφή της ή αν αντιστοιχεί σε κάποιον ήδη υπάρχον χρήστη</p> <p>4.1 Η εφαρμογή εμφανίζει μήνυμα σφάλματος “Please include ‘@’ in the email address”</p> <p>5. Αν η διεύθυνση ηλεκτρονικού ταχυδρομείου αντιστοιχεί σε κάποιον ήδη υπάρχον χρήστη</p> <p>4.1 Η εφαρμογή εμφανίζει μήνυμα σφάλματος “This email is already in use”</p>
<p>Postconditions: Ο φυλλομετρητής στέλνει τα στοιχεία του χρήστη στον εξυπηρετητή ο εξυπηρετητής τα σώζει στην βάση δεδομένων</p>

UD: 2
Use Case: Σύνδεση στην εφαρμογή
Actors: Χρήστης
<p>Preconditions:</p> <p>1. Ο χρήστης έχει ήδη δημιουργήσει έναν λογαριασμό</p>
<p>Flow of Events:</p> <p>1. Το Use Case ξεκινάει όταν ο χρήστης επισκεφθεί την εφαρμογή</p> <p>2. Ο χρήστης εισάγει την διεύθυνση ηλεκτρονικού ταχυδρομείου και τον κωδικό πρόσβασης στο πλαίσιο σύνδεσης που εμφανίζεται στον φυλλομετρητή</p> <p>3. Αν ο συνδιασμός διεύθυνσης ηλεκτρονικού ταχυδρομείου και κωδικού είναι σωστός</p> <p>3.1 Η εφαρμογή ανακατευθύνει τον χρήστη στην κεντρική σελίδα της εφαρμογής</p> <p>4. Αν ο συνδιασμός διεύθυνσης ηλεκτρονικού ταχυδρομείου δεν είναι σωστός</p> <p>3.1 Η εφαρμογή εμφανίζει στον χρήστη μήνυμα σφάλματος “Username/Password combination is not correct”</p>
Postconditions: -

UD: 3
Use Case: Απούνδεση από την εφαρμογή
Actors: Χρήστης
Preconditions: <ol style="list-style-type: none"> 1. Ο χρήστης έχει συνδεθεί στην εφαρμογή
Flow of Events: <ol style="list-style-type: none"> 1. Το Use Case ξεκινάει όταν ο χρήστης ανοίξει το μενού χρήστη και επιλέγει “Log out” 2. Η εφαρμογή ανακατευθύνει τον χρήστη στην αρχική σελίδα σύνδεσης
Postconditions: <ol style="list-style-type: none"> 1. Ο χρήστης αποσυνδέθηκε από την εφαρμογή

ID: 4
Use Case: Προσθήκη μέσου αποθήκευσης νέφους
Actors: Χρήστης, Μέσο αποθήκευσης νέφους, φυλλομετρητής
Preconditions: <ol style="list-style-type: none"> 1. Ο χρήστης έχει συνδεθεί
Flow of Events: <ol style="list-style-type: none"> 1. Το Use Case ξεκινάει όταν ο χρήστης ανοίγει το μενού και επιλέγει την ένδειξη “Add a Drive” 2. Η εφαρμογή εμφανίζει αναδυόμενο παράθυρο με τις επιλογές “Google Drive, Dropbox, One Drive” 3. Ο χρήστης επιλέγει μία από τις 3 επιλογές 4. Ο φυλλομετρητής ανακατευθύνει τον χρήστη στην διεύθυνση εξουσιοδότησης του μέσου αποθήκευσης νέφους που επέλεξε 5. Ο χρήστης εισάγει τα στοιχεία του μέσου αποθήκευσης νέφους 6. Ο φυλλομετρητής ανακατευθύνει τον χρήστη πίσω στην εφαρμογή
Postconditions: <ol style="list-style-type: none"> 1. Ο χρήστης έχει συνδέσει το μέσο αποθήκευσης νέφους

2. Η εφαρμογή εμφανίζει στον χρήστη τον ελεύθερο χώρο του μέσου αποθήκευσης νέφους που σύνδεσε που του παρέχει

ID: 5
Use Case: Ανέβασμα αρχείου
Actors: Χρήστης
Preconditions: <ol style="list-style-type: none">1. Ο χρήστης έχει συνδέσει τουλάχιστον ένα μέσο αποθήκευσης νέφους
Flow of Events: <ol style="list-style-type: none">1. Το Use Case ξεκινάει όταν ο χρήστης σέρνει ένα αρχείο ή κάνει κλικ στο κουμπί ανεβάσματος αρχείου από το μενού που βρίσκεται στο κάτω δεξί μέρος της εφαρμογής2. Η εφαρμογή ψάχνει να βρεί ποιο από τα συνδεδεμένα μέσα αποθήκευσης νέφους έχει τον μεγαλύτερο ελεύθερο χώρο3. Η εφαρμογή ανεβάζει το αρχείο στο κατάλληλο μέσο αποθήκευσης νέφους
Postconditions: <ol style="list-style-type: none">1. Ο χρήστης έχει ανεβάσει το αρχείο στην εφαρμογή2. Ο χρήστης μπορεί να διαχειριστεί το αρχείο (μετονομασία, διαγραφή, διαμοιρασμός)

ID: 6
Use Case: Μετονομασία αρχείου
Actors: Χρήστης
Preconditions: <ol style="list-style-type: none">1. Ο χρήστης έχει ανεβάσει τουλάχιστον ένα αρχείο
Flow of Events: <ol style="list-style-type: none">1. Το Use Case ξεκινάει όταν ο χρήστης ανοίξει το μενού του αρχείου και επιλέξει «Rename».2. Η εφαρμογή εμφανίζει αναδυόμενο παράθυρο στο οποίο ο χρήστης εισάγει το νέο όνομα3. Αν ο χρήστης επιλέξει από το αναδυόμενο παράθυρο την επιλογή “Rename”<ol style="list-style-type: none">3.1 Η εφαρμογή μετονομάζει το αρχείο

<p>4. Αν ο χρήστης επιλέξει από το αναδυόμενο παράθυρο την επιλογή “Cancel”</p> <p>4.1 Η εφαρμογή κλείνει το αναδυόμενο παράθυρο και η μετονομασία ακυρώνεται</p>
<p>Postconditions:</p> <ol style="list-style-type: none"> 1. Το αρχείο μετονομάζεται από την εφαρμογή 2. Το αρχείο δεν μετονομάζεται (στην περίπτωση ακύρωσης)

ID: 7
Use Case: Διαγραφή αρχείου
Actors: Χρήστης
<p>Preconditions:</p> <ol style="list-style-type: none"> 1. Ο χρήστης έχει ανεβάσει τουλάχιστον ένα αρχείο
<p>Flow of Events:</p> <ol style="list-style-type: none"> 1. Το Use Case ξεκινάει όταν ο χρήστης ανοίξει το μενού του αρχείου και επιλέξει «Delete». 2. Η εφαρμογή εμφανίζει αναδυόμενο παράθυρο με μήνυμα “Are you sure you want to delete the file?” 3. Αν ο χρήστης επιλέξει από το αναδυόμενο παράθυρο την επιλογή “Delete” <ol style="list-style-type: none"> 3.1 Η εφαρμογή διαγράφει το αρχείο 4. Αν ο χρήστης επιλέξει από το αναδυόμενο παράθυρο την επιλογή “Cancel” <ol style="list-style-type: none"> 4.1 Η εφαρμογή κλείνει το αναδυόμενο παράθυρο και η διαγραφή ακυρώνεται
<p>Postconditions:</p> <ol style="list-style-type: none"> 1. Το αρχείο διαγράφεται από την εφαρμογή 2. Το αρχείο δεν διαγράφεται (στην περίπτωση ακύρωσης)

ID: 8
Use Case: Διαμοιρασμός αρχείου
Actors: Χρήστης
<p>Preconditions:</p> <ol style="list-style-type: none"> 1. Ο χρήστης έχει ανεβάσει τουλάχιστον ένα αρχείο
Flow of Events:

<ol style="list-style-type: none"> 1. Το Use Case ξεκινάει όταν ο χρήστης ανοίξει το μενού του αρχείου και επιλέξει «Share». 2. Η εφαρμογή εμφανίζει ένα πράσινο εικονίδιο με την ένδειξη συνδέσμου δίπλα από το αρχείο 3. Αν ο χρήστης κάνει κλικ επάνω σε αυτό το εικονίδιο 3.1 Ο σύνδεσμος διαμοιρασμού αντιγράφεται
Postconditions: <ol style="list-style-type: none"> 1. Το αρχείο μπορεί να διαμοιραστεί

ID: 9
Use Case: Αφαίρεση διαμοιρασμού αρχείου
Actors: Χρήστης
Preconditions: <ol style="list-style-type: none"> 1. Ο χρήστης έχει ανεβάσει τουλάχιστον ένα αρχείο 2. Ο χρήστης έχει διαμοιραστεί το αρχείο
Flow of Events: <ol style="list-style-type: none"> 1. Το Use Case ξεκινάει όταν ο χρήστης ανοίξει το μενού του αρχείου και επιλέξει «Unshare». 2. Η εφαρμογή σταματάει να εμφανίζει το πράσινο εικονίδιο με την ένδειξη συνδέσμου δίπλα από το αρχείο
Postconditions: <ol style="list-style-type: none"> 1. Το αρχείο σταματάει να είναι δημόσιο

ID: 10
Use Case: Δημιουργία εικονικού φακέλου
Actors: Χρήστης
Preconditions: -
Flow of Events:

<ol style="list-style-type: none"> 1. Το Use Case ξεκινάει όταν ο χρήστης κάνει κλικ στο μενού στο κάτω δεξί μέρος της εφαρμογής με την ένδειξη «+» και κάνει κλικ στο εικονίδιο δημιουργίας φακέλου 2. Η εφαρμογή δημιουργεί τον εικονικό φάκελο
Postconditions: <ol style="list-style-type: none"> 1. Η εφαρμογή εμφανίζει ειδοποίηση επιτυχίας “Virtual folder created” 2. Ο χρήστης δημιούργησε έναν εικονικό φάκελο

ID: 11
Use Case: Διαμοιρασμός εικονικού φακέλου
Actors: Χρήστης
Preconditions: <ol style="list-style-type: none"> 1. Ο χρήστης έχει δημιουργήσει τουλάχιστον έναν εικονικό φάκελο
Flow of Events: <ol style="list-style-type: none"> 1. Το Use Case ξεκινάει όταν ο χρήστης ανοίξει το μενού του εικονικού φακέλου και επιλέξει «Share». 2. Η εφαρμογή εμφανίζει αναδυόμενο παράθυρο όπου ο χρήστης εισάγει την διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη με τον οποίο θέλει να διαμοιραστεί τον εικονικό φάκελο 3. Αν ο χρήστης επιλέξει από το αναδυόμενο παράθυρο την επιλογή “Share” <ol style="list-style-type: none"> 3.1 Η εφαρμογή εμφανίζει στον χρήστη μήνυμα επιτυχίας του διαμοιρασμού 4. Αν ο χρήστης επιλέξει από το αναδυόμενο παράθυρο την επιλογή “Cancel” <ol style="list-style-type: none"> 4.1 Η εφαρμογή κλείνει το αναδυόμενο παράθυρο και ακυρώνει τον διαμοιρασμό
Postconditions: <ol style="list-style-type: none"> 1. Ο χρήστης διαμοιράστηκε τον εικονικό φάκελο 2. Ο χρήστης δεν διαμοιράστηκε τον εικονικό φάκελο (στην περίπτωση ακύρωσης)

ID: 12
Use Case: Αφαίρεση διαμοιρασμού εικονικού φακέλου

Actors: Χρήστης
Preconditions: <ol style="list-style-type: none"> 1. Ο χρήστης έχει δημιουργήσει τουλάχιστον έναν εικονικό φάκελο 2. Ο χρήστης έχει διαμοιραστεί τον εικονικό φάκελο
Flow of Events: <ol style="list-style-type: none"> 1. Το Use Case ξεκινάει όταν ο χρήστης ανοίξει το μενού του εικονικού φακέλου και επιλέξει «Unshare». 2. Η εφαρμογή εμφανίζει αναδυόμενο παράθυρο με όλες τις διευθύνσεις ηλεκτρονικού ταχυδρομείου των μελών που συμμετέχουν στον διαμοιρασμό 3. Αν ο χρήστης κάνει κλικ μία από τις διευθύνσεις ηλεκτρονικού ταχυδρομείου <ol style="list-style-type: none"> 3.1 Η εφαρμογή αφαιρεί τον χρήστη από τον διαμοιρασμό 4. Αν ο χρήστης επιλέξει από το αναδυόμενο παράθυρο την επιλογή “Cancel” <ol style="list-style-type: none"> 4.1 Η εφαρμογή κλείνει το αναδυόμενο παράθυρο
Postconditions: <ol style="list-style-type: none"> 1. Ο χρήστης αφαίρεσε κάποιον άλλο χρήστη από τον διαμοιρασμό 2. Ο χρήστης δεν αφαίρεσε κανέναν χρήστη από τον διαμοιρασμό (στην περίπτωση ακύρωσης)

ID: 13
Use Case: Μετονομασία εικονικού φακέλου
Actors: Χρήστης
Preconditions: <ol style="list-style-type: none"> 1. Ο χρήστης έχει δημιουργήσει τουλάχιστον έναν εικονικό φάκελο
Flow of Events: <ol style="list-style-type: none"> 2. Το Use Case ξεκινάει όταν ο χρήστης ανοίξει το μενού του εικονικού φακέλου και επιλέξει «Rename». 3. Η εφαρμογή εμφανίζει αναδυόμενο παράθυρο στο οποίο ο χρήστης εισάγει το νέο όνομα 4. Αν ο χρήστης επιλέξει από το αναδυόμενο παράθυρο την επιλογή “Rename” <ol style="list-style-type: none"> 4.1 Η εφαρμογή μετονομάζει τον εικονικό φάκελο 5. Αν ο χρήστης επιλέξει από το αναδυόμενο παράθυρο την επιλογή “Cancel”

5.1 Η εφαρμογή κλείνει το αναδυόμενο παράθυρο και η μετονομασία ακυρώνεται
Postconditions: <ol style="list-style-type: none"> 1. Το αρχείο μετονομάζεται από την εφαρμογή 2. Το αρχείο δεν μετονομάζεται (στην περίπτωση ακύρωσης)

ID: 14
Use Case: Διαγραφή εικονικού φακέλου
Actors: Χρήστης
Preconditions: <ol style="list-style-type: none"> 1. Ο χρήστης έχει δημιουργήσει τουλάχιστον έναν εικονικό φάκελο
Flow of Events: <ol style="list-style-type: none"> 1. Το Use Case ξεκινάει όταν ο χρήστης ανοίξει το μενού του αρχείου και επιλέξει «Delete». 2. Η εφαρμογή εμφανίζει αναδυόμενο παράθυρο με μήνυμα “Are you sure you want to delete the file ? “ 3. Αν ο χρήστης επιλέξει από το αναδυόμενο παράθυρο την επιλογή “Delete” <ol style="list-style-type: none"> 3.1 Η εφαρμογή διαγράφει τον εικονικό φάκελο καθώς και τα αρχεία που εμπεριέχονται 4. Αν ο χρήστης επιλέξει από το αναδυόμενο παράθυρο την επιλογή “Cancel” <ol style="list-style-type: none"> 4.1 Η εφαρμογή κλείνει το αναδυόμενο παράθυρο και η διαγραφή ακυρώνεται
Postconditions: <ol style="list-style-type: none"> 1. Ο εικονικός φάκελος και τα περιεχόμενα του διαγράφονται από την εφαρμογή 2. Η εφαρμογή δεν διαγράφει τίποτα (στην περίπτωση ακύρωσης)

ID: 15
Use Case: Άνοιγμα περιηγητή αρχείων των μέσων αποθήκευσης νέφους
Actors: Χρήστης
Preconditions: <ol style="list-style-type: none"> 1. Ο χρήστης έχει συνδέσει τουλάχιστον ένα μέσο αποθήκευσης νέφους

Flow of Events:

2. Το Use Case ξεκινάει όταν ο χρήστης ανοίξει το μενού χρήστη και κάνει κλικ στην επιλογή “Switch to all in one drive”.
3. Η εφαρμογή εμφανίζει τα αρχεία των μέσων αποθήκευσης νέφους που έχει συνδέσει ο χρήστης όπως ακριβώς αυτά βρίσκονται αποθηκευμένα στα μέσα αυτά

Postconditions:

1. Ο χρήστης μπορεί οποιαδήποτε στιγμή να μεταβεί στην σελίδα του εικονικού μέσου αποθήκευσης νέφους ανοίγοντας πάλι το μενού χρήστη και κάνοντας κλικ στην ένδειξη “Switch to virtual drive”

2.4 Παρόμοιες εφαρμογές

Σε αυτή την ενότητα θα αναλύσουμε μερικές εφαρμογές που έχουν παρόμοια λειτουργικότητα με την δικιά μας εφαρμογή. Αυτό δε σημαίνει όμως ότι οι εφαρμογές που αναλύονται παρακάτω είναι ακριβώς ίδιες με την δικιά μας.

MultCloud

Η εφαρμογή MultCloud αποτελεί μία εφαρμογή που προσφέρει μία ενοποιημένη γραφική διεπαφή για διάφορα μέσα αποθήκευσης νέφους που μπορεί να προσθέσει ο χρήστης. Επιπλέον προσφέρει και άλλες υπηρεσίες όπως μετακίνηση αρχείων μεταξύ των συνδεδεμένων μέσων αποθήκευσης νέφους. Παρόλο που υπάρχει δωρεάν πλάνο για έναν περιορισμένο χώρο αποθήκευσης, αποτελεί μία κατά κύριο λόγο συνδρομητική εφαρμογή.

oCloud

Η εφαρμογή oCloud αποτελεί ακόμη μία λύση ενοποιημένου μέσου αποθήκευσης νέφους. Ο χρήστης σε αυτή την εφαρμογή μπορεί να συνδέει τα διάφορα μέσα αποθήκευσης νέφους που διαθέτει, και να τα προσπελάζει μέσα από μία κοινή γραφική διεπαφή. Αποτελεί επίσης μία συνδρομητική εφαρμογή, όμως δεν υπάρχει κανένας περιορισμός στην σύνδεση και προσπέλαση επιμέρους μέσων αποθήκευσης νέφους ακόμη και χωρίς συνδρομή.

Koofr

Η εφαρμογή Koofr υποστηρίζει την σύνδεση των τριών μέσων αποθήκευσης νέφους που υποστηρίζει και η εφαρμογή μας (Google Drive, Dropbox, OneDrive). Επιπλέον, η εφαρμογή αυτή προσφέρει και δικό της χώρο για αποθήκευση αρχείων στον νέφος. Αποτελεί επίσης μία συνδρομητική υπηρεσία, μόνο όμως για την αύξηση του χώρου που προσφέρει. Οι μη-συνδρομητές χρήστες έχουν δωρεάν 2GB ελεύθερου χώρου καθώς και την δυνατότητα να συνδέσουν τα μέσα αποθήκευσης νέφους που αναφέραμε και να προσπελάσουν τα αρχεία τους.

Κύριες διαφορές με την εφαρμογή μας

Οι υπηρεσίες που αναφέραμε, παρόλο που μοιάζουν με την δικιά μας εφαρμογή, προσφέρουν απλώς μία γραφική διεπαφή όπου ο χρήστης μπορεί να προσπελάζει τα αρχεία του κάθε μέσου αποθήκευσης νέφους και να εκτελεί λειτουργίες σε αυτά. Οι λειτουργίες που δεν παρέχονται από αυτές τις εφαρμογές και τις καθιστούν ουσιαστικά διαφορετικές από την εφαρμογή μας, σχετίζονται με το εικονικό μέσο αποθήκευσης νέφους που υλοποιούμε (ενότητα **4.1.4**) και είναι οι εξής:

- Αυτόματη κατανομή αρχείων στα επιμέρους μέσα αποθήκευσης νέφους χωρίς την απόφαση του χρήστη.
- Υποστήριξη εικονικών φακέλων που «φιλοξενούν» αρχεία μεταξύ διαφορετικών μέσων αποθήκευσης νέφους
- Αναδιάταξη αρχείων σε περίπτωση που υπάρξει εξωτερικός κατακερματισμός στο εικονικό μέσο αποθήκευσης νέφους (υπο-ενότητα **4.1.4.4**)

Σύμφωνα με τα παραπάνω, η εφαρμογή μας προσφέρει μία πραγματική αίσθηση ενός ενοποιημένου μέσου αποθήκευσης νέφους και όχι μία απλή γραφική διεπαφή που δίνει πρόσβαση στα αρχεία των συνδεδεμένων μέσων αποθήκευσης νέφους, γεγονός που την καθιστά διαφορετική από τις υπόλοιπες.

Κεφάλαιο 3.

Τεχνολογικό

Υπόβαθρο

3.1 Εισαγωγή

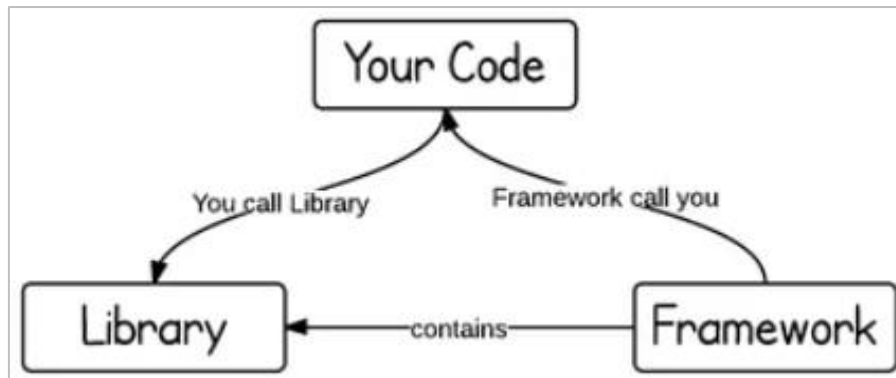
Στο κεφάλαιο αυτό, θα περιγράψουμε αναλυτικά όλες τις τεχνολογίες που χρειάστηκαν για την υλοποίηση της εφαρμογής τόσο στην μεριά του πελάτη (frontend) όσο και στην μεριά του εξυπηρετητή (backend), αφού πρώτα εξηγήσουμε κάποιους χρήσιμους όρους που θα χρειαστούν για την πλήρη κατανόηση των κεφαλαίων και ενοτήτων που θα ακολουθήσουν.

3.2 Ορολογία

Frontend: Με τον όρο frontend, αναφερόμαστε στα κομμάτια κώδικα που υλοποιούν την γραφική διεπαφή (UI) την οποία βλέπει ο χρήστης.

Backend: Με τον όρο backend, αναφερόμαστε στα κομμάτια κώδικα που υλοποιούν τον εξυπηρετητή (server).

Framework: Με τον όρο framework, εννοούμε μία «αφαίρεση» η οποία περιέχει έναν συγκεκριμένο τρόπο και κανόνες, σύμφωνα με τους οποίους μπορεί να αναπτυχθεί μία εφαρμογή. Ο όρος αυτός, δεν θα πρέπει να μπερδεύεται με τον όρο της βιβλιοθήκης (library). Σε μία βιβλιοθήκη, εμείς αποφασίζουμε πότε και πού θα χρειαστεί να την χρησιμοποιήσουμε. Αντιθέτως, το framework χρησιμοποιεί τον κώδικα που γράφουμε αν αυτός ακολουθεί τους κανόνες που αυτό καθορίζει.



Σχήμα 3.2.1: Σχηματική απεικόνιση της κύριας διαφοράς *framework*-βιβλιοθήκης. [\[1\]](#)

API: Ο όρος API είναι το ακρώνυμο του όρου Application Programming Interface. Το API αποτελεί ένα μέσο που προσφέρει σε κάποιον χρήστη πρόσβαση στους πόρους μιας εφαρμογής, όπως π.χ. πληροφορίες για μία οντότητα που βρίσκεται αποθηκευμένη στην βάση δεδομένων της εφαρμογής αυτής.

Full-stack: Με τον όρο *full-stack*, αναφερόμαστε σε ένα πλήρες πρότζεκτ που αποτελείται από τα τρία κύρια στοιχεία που συνθέτουν μια διαδικτυακή εφαρμογή: **frontend**, **backend** και βάση δεδομένων. Αυτά τα τρία στοιχεία, λειτουργούν ανεξάρτητα μεταξύ τους, που όμως συνεργάζονται μεταξύ τους μέσω ενός **API**.

Cross-platform: Ο όρος “*cross-platform*” χρησιμοποιείται όταν κάποια οντότητα (π.χ. μία τεχνολογία), υποστηρίζεται και μπορεί να χρησιμοποιηθεί από ένα πλήθος πλατφόρμων.

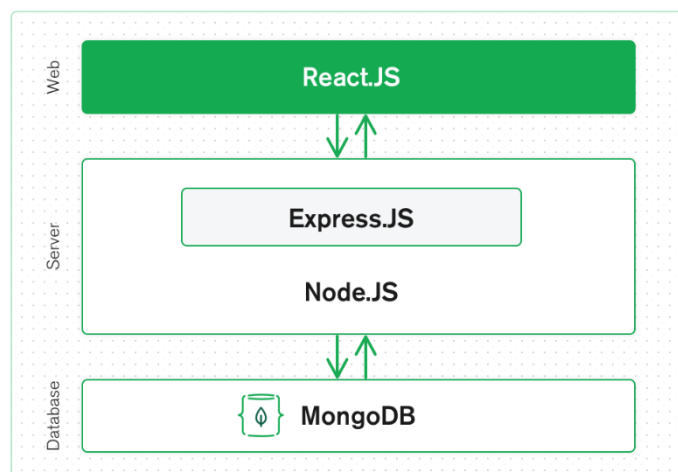
3.3 Η στοίβα MERN

Με τον όρο «στοίβα», στον κόσμο της ανάπτυξης διαδικτυακού λογισμικού, αναφερόμαστε σε οποιονδήποτε συνδυασμό γλωσσών προγραμματισμού και τεχνολογιών ή συνδυασμού προϊόντων λογισμικού, με σκοπό την ανάπτυξη μιας *full-stack* εφαρμογής.

Στην δικιά μας περίπτωση, κάναμε την επιλογή της λεγόμενης «στοίβας MERN». Η στοίβα MERN, ακολουθεί την παραδοσιακή τριεπίπεδη αρχιτεκτονική [2] και κάθε γράμμα της λέξης “MERN” αντιστοιχεί σε μία τεχνολογία διαδικτύου. Οι τεχνολογίες αυτές, αντιστοιχίζονται βάσει το επίπεδο που ανήκουν και αναλύονται ως εξής:

- **Backend:** Node.js, Express.js
- **Frontend:** ReactJS
- **Βάση Δεδομένων:** MongoDB

Όλες οι τεχνολογίες που αναφέρθηκαν, περιέχουν κάποια κοινά χαρακτηριστικά, τα οποία αποτελούν τον κυριότερο λόγο που επιλέξαμε τις τεχνολογίες που μόλις αναφέραμε. Ένα από αυτά, είναι η χρήση κοινής γλώσσας προγραμματισμού, μεταξύ frontend και backend καθώς το Node.js, το Express.js και η ReactJS, χρησιμοποιούν την γλώσσα προγραμματισμού Javascript. Επίσης, η βάση δεδομένων MongoDB, λειτουργεί αρμονικά με το Node.js, καθιστώντας εύκολη την αποθήκευση, διαχείριση και αναπαράσταση των δεδομένων σε κάθε ένα από τα τρία επίπεδα της εφαρμογής, εφόσον η ροή των δεδομένων είναι εύκολη χρησιμοποιώντας παντού την μορφή JSON. Αυτό σημαίνει ότι κερδίζουμε αρκετό χρόνο τόσο στην ανάπτυξη της εφαρμογής, όσο και στην εκσφαλμάτωση της [3].



Σχήμα 3.3.1: Σχηματική απεικόνιση της τριεπίπεδης αρχιτεκτονικής, της στοίβας MERN. [2]

3.4 Το framework Node.js

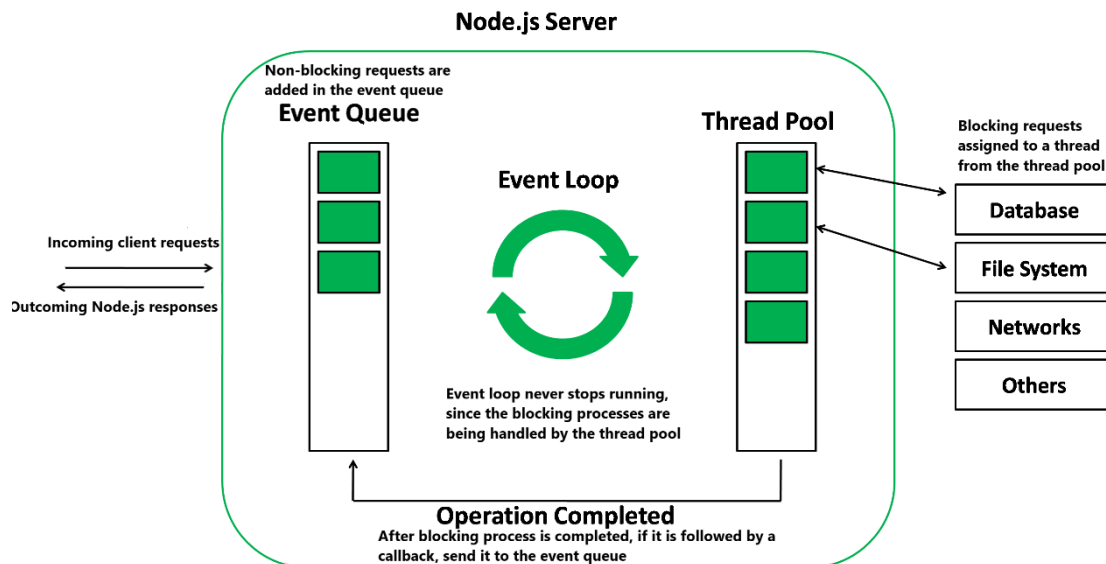
Το Node.js είναι ένα ανοιχτού λογισμικού framework, το οποίο μας επιτρέπει να τρέχουμε κώδικα Javascript έξω από το περιβάλλον ενός browser. Αυτή η δυνατότητα, είναι μία από τις βασικότερες τις εφαρμογής μας εφόσον στο μέρος του εξυπηρετητή (backend) θα χρειαστεί να τρέξουμε κώδικα Javascript. Πριν την εμφάνιση του Node.js, η εκτέλεση κώδικα Javascript ήταν εφικτή μόνο εντός του περιβάλλοντος ενός browser. Το Node.js, μας επιτρέπει την χρήση Javascript στο backend καθώς και την κοινοχρησία του κώδικα μεταξύ frontend και backend καθιστώντας το ως την, πλέον, ιδανική επιλογή εφόσον δεν χρειάζεται να χρησιμοποιούμε δύο ή περισσότερες διαφορετικές γλώσσες προγραμματισμού μεταξύ frontend και backend.

Ως προς την αρχιτεκτονική, το Node.js ακολουθεί μία “event-driven” αρχιτεκτονική που επιτρέπει ασύγχρονες λειτουργίες εισόδου/εξόδου (I/O). Αυτή η αρχιτεκτονική, στοχεύει στην βελτιστοποίηση και την επεκτασιμότητα των εφαρμογών διαδικτύου που απαιτούν πολλές λειτουργίες εισόδου/εξόδου καθώς και σε εφαρμογές πραγματικού χρόνου όπως εφαρμογές επικοινωνίας καθώς και διαδικτυακά παιχνίδια που τρέχουν σε browser.

Η “event-driven” αρχιτεκτονική, επιτρέπει την ανάπτυξη γρήγορων διαδικτυακών server οι οποίοι δέχονται πολλαπλά αιτήματα ταυτόχρονα, χωρίς όμως την χρήση πολλαπλών νημάτων, εκτός του κύριου νήματος το οποίο τρέχει το λεγόμενο “**event loop**” το οποίο ακολουθεί μία “non-blocking” προσέγγιση. Πιο συγκεκριμένα, όταν ο server δέχεται μία αίτηση, την προσθέτει σε μία ουρά ονόματι “event queue” και στην συνέχεια ακολουθούν δύο σενάρια:

1. Αν η αίτηση αποτελεί μία non-blocking διεργασία, το Node.js θα επεξεργαστεί αυτή την αίτηση, θα ετοιμάσει μία απάντηση και θα την στείλει πίσω στον πελάτη καθώς επίσης θα αφαιρεθεί από την ουρά.
2. Αν η αίτηση αποτελεί μία blocking διεργασία, δηλαδή μία διεργασία εισόδου/εξόδου, το Node.js αναθέτει αυτή την διεργασία σε μία ομάδα νημάτων που είναι ανεξάρτητα του κυρίως προγράμματος με αποτέλεσμα το “**event-loop**” να μην σταματάει να διαχειρίζεται άλλα αιτήματα που εισέρχονται στην ουρά

“event-queue”. Όταν η διεργασία εισόδου/εξόδου ολοκληρωθεί, ο κώδικας που πρέπει να ακολουθήσει μετά από αυτή την διεργασία (ο οποίος ονομάζεται “callback”) προστίθεται στην ουρά, επεξεργάζεται από το Node.js και αποστέλλεται στον πελάτη η απάντηση.



Σχήμα 3.4.1: Σχηματική απεικόνιση της “event-driven” αρχιτεκτονικής.[4]

3.5 Το framework Express.js

Το Express.js είναι επίσης ένα ανοιχτού λογισμικού framework που τρέχει πάνω στο Node.js και χρησιμοποιείται για την σχεδίαση APIs διαδικτυακών εφαρμογών καθώς και αποτελεί το “de facto” server framework για το Node.js. Ο λόγος που χρησιμοποιούμε το Express.js στην παρούσα διπλωματική, είναι για την επίτευξη επικοινωνίας μεταξύ πελάτη και εξυπηρετητή μέσω της σχεδίασης και υλοποίησης ενός API. Πιο συγκεκριμένα το Express.js προσφέρει μηχανισμούς για μία σειρά λειτουργιών όπως οι εξής:

- Δημιουργία χειριστών (handlers) για HTTP αιτήματα διαφορετικών διαδρομών (routes).

- Δημιουργία HTML «όψεων» (views) που δημιουργούνται δυναμικά ανάλογα με τα δεδομένα που χρειάζονται να αναπαρασταθούν, με σκοπό την αποστολή τους στον πελάτη (browser).
- Διαχείριση κοινών λειτουργιών όπως την ανάθεση συγκεκριμένης θύρας (port), που θα χρησιμοποιηθεί από την εφαρμογή του server.
- Δυνατότητα προσθήκης ενδιάμεσης διαχείρισης των HTTP αιτημάτων (middleware) σε οποιοδήποτε σημείο εντός της ροής ενός αιτήματος.

Σε γενικές γραμμές, το Express.js αποτελεί ένα μινιμαλιστικό framework, παρόλα αυτά, υπάρχουν πολλές συμβατές βιβλιοθήκες που διευθετούν και λύνουν κάθε είδους προβλήματα στην ανάπτυξη μιας διαδικτυακής εφαρμογής. Υπάρχουν δηλαδή βιβλιοθήκες που διαχειρίζονται τα “cookies”, τις συνεδρίες (sessions), την σύνδεση και αποσύνδεση ενός χρήστη σε μία διαδικτυακή εφαρμογή, την μεταφορά παραμέτρων μέσω του URL, την ασφαλή μεταφορά δεδομένων μέσω HTTP αιτημάτων κ.ο.κ. Όλες αυτές οι βιβλιοθήκες συντηρούνται από την κοινότητα προγραμματιστών που αναπτύσσουν το Express.js, επομένως η συμβατότητα τους είναι πάντα άρτια και χωρίς προβλήματα.

3.6 Η βιβλιοθήκη ReactJS

Η ReactJS, αποτελεί μία ανοιχτού-λογισμικού βιβλιοθήκη της γλώσσας προγραμματισμού Javascript και χρησιμοποιείται για την ανάπτυξη διαδικτυακών διεπαφών χρήστη (user interfaces). Αναπτύχθηκε από την Facebook (από την οποία συντηρείται ακόμη και σήμερα) με σκοπό την γρήγορη και αποτελεσματική ανάπτυξη διαδραστικών διεπαφών διαδικτύου. Αποτελεί την πλέον πιο δημοφιλή βιβλιοθήκη για την δημιουργία διεπαφών διαδικτύου και σε αυτό συνέβαλαν μερικά χαρακτηριστικά της, όπως:

- Εύκολη επαναχρησιμοποίηση του κώδικα με την χρήση των λεγόμενων “components”. καθώς και η ευκολία δημιουργίας διεπαφών «μονής σελίδας» (single page applications).

- Ευκολία στην δημιουργία διεπαφών «μονής σελίδας», των οποίων το περιεχόμενο αλλάζει δυναμικά, χωρίς να χρειάζεται συνεχής ανανέωση της σελίδας σε έναν browser.
- Δυνατότητα χρήσης της γλώσσας προγραμματισμού Typescript που αποτελεί μία προέκταση της γλώσσας Javascript, που προσθέτει τύπους μεταβλητών, κάτι το οποίο δεν υφίσταται στην Javascript.
- Επαναχρησιμοποίηση κώδικα μεταξύ εφαρμογών διαδικτύου και εφαρμογών κινητών συσκευών, χάρη στο συγγενικό framework “React Native” που χρησιμοποιείται για την ανάπτυξη εφαρμογών στις πλατφόρμες Android, Apple και Windows.

Η ReactJS συνδιάζει την εγγραφή Javascript κώδικα για το λογικό της κομμάτι όπως π.χ. scripts και HTML κώδικα για την δημιουργία της σελίδας καθώς και την χρήση της CSS (Cascading Style Sheets) για ο στυλάρισμα της HTML.

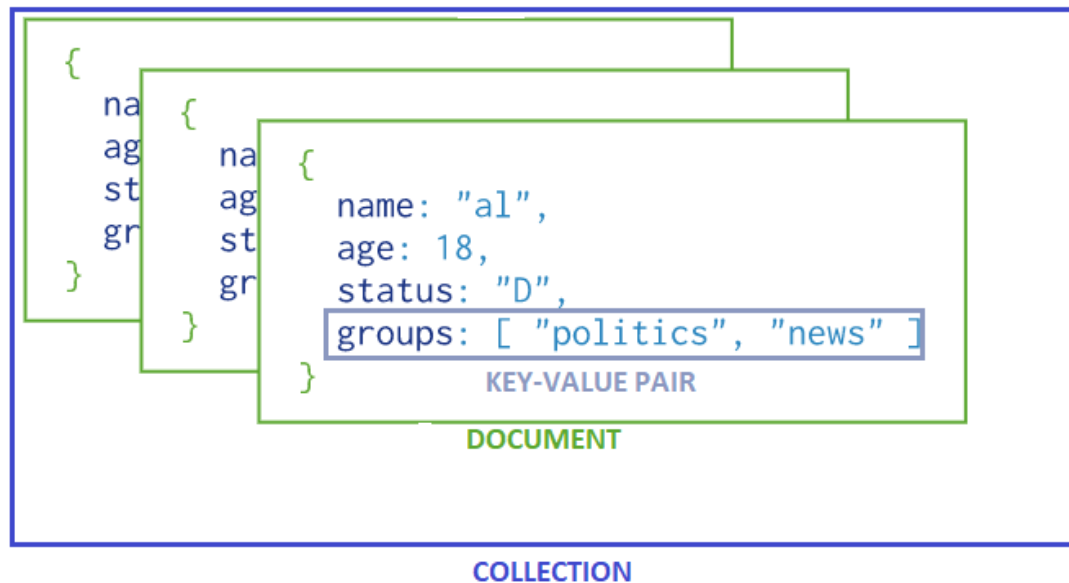
3.7 Η βάση δεδομένων MongoDB

Η MongoDB, αποτελεί μία γενικής χρήσης cross-platform (όσον αφορά τις υποστηριζόμενες γλώσσες προγραμματισμού), μη σχεσιακή βάση δεδομένων. Βασίζεται στην αποθήκευση των δεδομένων ως έγγραφα (document-based) και όχι ως πίνακες με εγγραφές όπως στις παραδοσιακές σχεσιακές βάσεις δεδομένων (π.χ. SQL). Γι’ αυτόν το λόγο, χαρακτηρίζεται και ως μία “No-SQL” βάση δεδομένων.

Αποθήκευση των δεδομένων

Η αποθήκευση των δεδομένων, γίνεται σε μορφή JSON. Αυτό, την καθιστά πολύ ευέλικτη διότι αυτός είναι ο τρόπος που σκεφτόμαστε τα δεδομένα φυσιολογικά. Μία οντότητα δηλαδή, να περιέχει κάποια χαρακτηριστικά. Κάθε οντότητα στην βάση MongoDB, ονομάζεται «έγγραφο» (document) και μία ομάδα εγγράφων με όμοια χαρακτηριστικά, συνθέτουν μία «συλλογή» (collection). Κάθε έγγραφο, αντιστοιχεί σε ένα JSON αντικείμενο που περιέχει ζεύγη κλειδιού-τιμής (key-value pairs). Κάθε κλειδί, έχει όνομα

που έχει προκαθοριστεί κατά την δήλωση του εγγράφου και η τιμή του θα πρέπει να έχει τύπο που έχει επίσης προκαθοριστεί. Οι υποστηριζόμενοι τύποι των τιμών μπορούν να είναι οι εξής: Αριθμός, Αλφαριθμητικό, Πίνακας, Αντικείμενο τύπου JSON.



Σχήμα 3.7.1: Η μορφή μιας συλλογής στην βάση MongoDB.[\[5\]](#)

Κριτήρια επιλογής, χρήσης της MongoDB

Ένα από τα βασικότερα ερωτήματα, είναι το πότε είναι ιδανικό να χρησιμοποιήσουμε μία μη σχεσιακή βάση δεδομένων όπως η MongoDB. Μερικοί από τους λόγους που ωθούν εταιρείες και προγραμματιστές να χρησιμοποιήσουν την MongoDB είναι οι εξής:

- Όταν τα δεδομένα που πρόκειται να διαχειριστούμε δεν περιέχουν συσχετίσεις, μεταξύ τους, επομένως δεν υπάρχει λόγος χρήσης SQL βάσης δεδομένων εφόσον οι κυριότερες της δυνατότητες, δεν πρόκειται να αξιοποιηθούν.
- Όταν δεν μπορεί να προκαθοριστεί ένα «σχήμα» (schema) σε σχεσιακή βάση, εξαιτίας της πολυπλοκότητας που έχουν τα δεδομένα.
- Όταν η εφαρμογή χρησιμοποιεί δεδομένα σε JSON μορφή, επομένως η χρήση μιας βάσης δεδομένων όπως η MongoDB θα εκφέρει ευκολίες στην διαχείριση των δεδομένων.
- Όταν δεν είμαστε σίγουροι αν τα δεδομένα θα αλλάξουν κατά την διάρκεια «ζωής» της εφαρμογής μας. Σε μία σχεσιακή βάση δεδομένων, αν τα δεδομένα

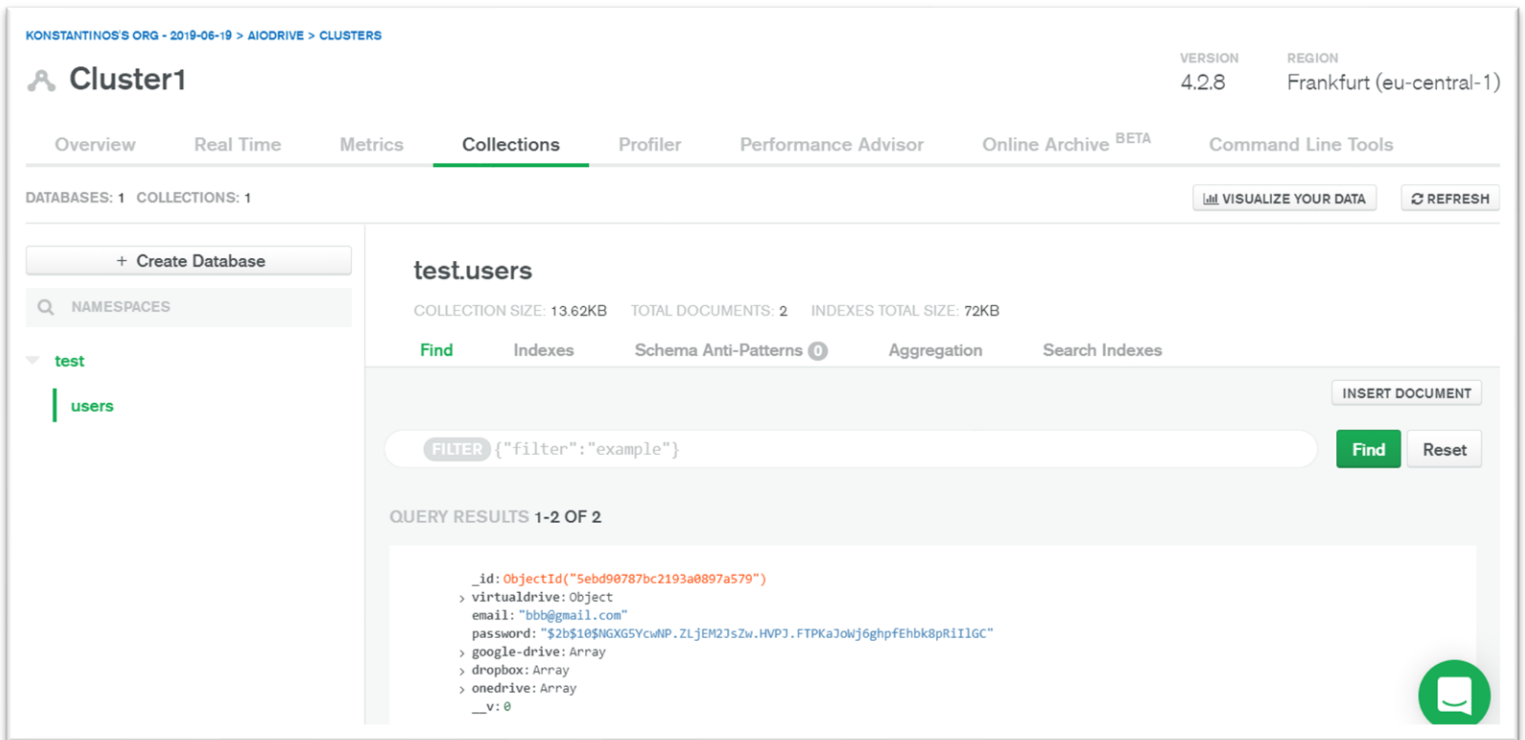
αλλάξουν, χρειάζεται ολική ανασύνταξη του «σχήματος» της βάσης. Αντιθέτως, σε μία μη σχεσιακή βάση, αυτό γίνεται με ευκολία, λόγω της ευελιξίας στην προσθήκη και αφαίρεση νέων χαρακτηριστικών χάρη στην JSON μορφή των δεδομένων.

MongoDB Atlas[\[6\]](#)

Η MongoDB Atlas, είναι μία καινούρια database-as-a-service (DBaaS) πλατφόρμα της MongoDB, που προσφέρει στους χρήστες της, πλήρη διαχείριση της βάσης δεδομένων μέσω μιας online γραφικής διεπαφής. Αυτό, σημαίνει πως οι χρήστες αυτής της υπηρεσίας, δεν χρειάζεται να διαχειρίζονται τα δεδομένα τους μέσω του δικού τους συστήματος. Πιο συγκεκριμένα, αυτά που προσφέρει η εν λόγω υπηρεσία και την καθιστά χρήσιμη, είναι τα εξής:

- Αποθήκευση όλων των δεδομένων στο «νέφος» (cloud) αναλαμβάνοντας επίσης το κομμάτι της ασφάλειας.
- Παρακολούθηση των συνδέσεων στην βάση, έχοντας με αυτόν τον τρόπο τον πλήρη έλεγχο για το ποιος έχει πρόσβαση.
- Ενσωματωμένο σύστημα που κάνει αυτόματη δημιουργία αντιγράφων ασφαλείας (backups), εξαλείφοντας έτσι τον κίνδυνο απώλειας δεδομένων.
- Μετρικές και γραφήματα σχετικά με την χρήση της βάσης, τα αιτήματα που γίνονται, την κίνηση που υπάρχει στην υπηρεσία, καθώς και την επεξεργαστική ισχύ που χρειάζονται τα απομακρυσμένα συστήματα στο «νέφος», παρέχοντας έτσι μια σφαιρική εικόνα για την κατάσταση της βάσης δεδομένων.
- Αυτόματη αύξηση της επεξεργαστικής ισχύς που χρειάζεται ανάλογα με τον φόρτο που υπάρχει στην βάση δεδομένων.

Όλα τα παραπάνω, καθιστούν την πλατφόρμα MongoDB Atlas ως μία πολύ καλή επιλογή, εφόσον μας γλυτώνουν από πολλά ζητήματα που χρειάζονται ενδεχομένως πολύ χρόνο για να υλοποιηθούν.



Σχήμα 3.7.2: Το interface της πλατφόρμας MongoDB Atlas.

3.8 Η αρχιτεκτονική REST και το REST-ful API [\[7\]\[8\]](#)

Η αρχιτεκτονική REST (Representational State Transfer), χρησιμοποιείται για την επίτευξη επικοινωνίας μεταξύ δύο μέσων με την χρήση του πρωτοκόλλου HTTP. Περιέχει ένα σύνολο από κανόνες, που χρησιμοποιούνται με ακριβώς τον ίδιο τρόπο ανεξαρτήτως λειτουργικού συστήματος και γλώσσας προγραμματισμού, σε κάθε σύστημα. Στην παρούσα διπλωματική, θα χρησιμοποιήσουμε τους κανόνες αυτής της αρχιτεκτονικής για να σχεδιάσουμε το API.

Η λογική της αρχιτεκτονικής αυτής, χρησιμοποιείται για την σχεδίαση APIs, γνωστά και ως “RESTful APIs” στον κόσμο της ανάπτυξης εφαρμογών διαδικτύου, με σκοπό την επίτευξη επικοινωνίας μεταξύ πελάτη (client) και εξυπηρετητή (server). Η επικοινωνία αυτή πραγματοποιείται μέσω αιτημάτων που γίνονται από την μεριά του πελάτη τα οποία συνοδεύονται από μία απάντηση από την μεριά του εξυπηρετητή. Οι εφαρμογές διαδικτύου που ακολουθούν και τηρούν τους κανόνες της αρχιτεκτονικής REST ονομάζονται “RESTful εφαρμογές διαδικτύου”. Στην συνέχεια, θα κάνουμε μία πιο αναλυτική εξήγηση αυτών που αναφέρθηκαν.

Ανατομία ενός αιτήματος

Είναι σημαντικό να αναφέρουμε, ότι κάθε αίτημα που γίνεται από κάποιον πελάτη, αποτελείται από τέσσερα επιμέρους κομμάτια:

1. Τελικό σημείο (endpoint)
2. Μέθοδος (method):
3. Κεφαλίδες (headers):
4. Κορμός (body):

Τελικό σημείο (endpoint)

Το **endpoint** αποτελεί το αρχικό σημείο ενός API στο οποίο γίνονται τα αιτήματα. Περιέχει δύο κομμάτια διεύθυνσης, ένα αμεταβλητό (ονομάζεται **root-endpoint**) και ένα μεταβλητό (το οποίο ονομάζεται **path**). Το **root-endpoint** αποτελεί το αρχικό σημείο ενός API, ενώ το **path** σχετίζεται με τον πόρο που θα χρειαστεί να κάνουμε κάποιου είδους αιτήματος.

Για παράδειγμα, η υπηρεσία **Github** προσφέρει πρόσβαση σε διάφορους πόρους της μέσω του **root-endpoint** της, <https://api.github.com> το οποίο μπορεί να συνοδευτεί από ένα **path** που σχετίζεται με την πρόσβαση στους πόρους των εγγεγραμμένων μελών της, το `/users/<username>/repos`. Συνολικά, κάνοντας το κατάλληλο - όσον αφορά το είδος του αιτήματος - αίτημα στο ολοκληρωμένο **endpoint** <https://api.github.com/users/<username>/repos>, μπορούμε να έχουμε πρόσβαση σε όλα τα αποθετήρια (repositories) ενός εγγεγραμμένου χρήστη,

Μέθοδος (method)

Η μέθοδος σε ένα αίτημα, αναφέρεται στο είδος που έχει το αίτημα αυτό. Τα πιο γνωστά είδη αιτημάτων που θα χρησιμοποιήσουμε, είναι τέσσερα και αναφέρονται στο **τι** ενέργεια θα πραγματοποιήσουν σε κάποιον πόρο:

1. **GET**: Λήψη ενός πόρου από τον server.
2. **POST**: Προσθήκη ενός νέου πόρου στον server.
3. **DELETE**: Αφαίρεση ενός πόρου από τον server.
4. **PUT/PATCH**: Τροποποίηση της τιμής ενός πόρου στον server.

Κεφαλίδες (headers)

Οι κεφαλίδες, χρησιμοποιούνται για την παροχή πληροφορίας προς τον εξυπηρετητή αλλά και προς τον πελάτη. Αποτελούνται από ένα ζεύγος κλειδιού-τιμής. Το κλειδί καθορίζει τον τίτλο της κεφαλίδας, και η τιμή το είδος της κεφαλίδας.

Οι πιο σημαντικές κεφαλίδες για την ανάγκη της διπλωματικής μας, είναι δύο:

1. **Authorization Header:** Χρησιμοποιείται για την αποστολή ενός **token** αυθεντικοποίησης, έτσι ώστε να μπορεί ο server να ελέγξει αν θα πρέπει να δώσει πρόσβαση στον πελάτη, στον πόρο που ζητά. Διατυπώνεται ως εξής:

```
"Authorization: Bearer <token>"
```

2. **Content-Type Header:** Χρησιμοποιείται για την δήλωση στον server, σχετικά με το είδος που θα έχει ο **κορμός** που πρόκειται να στείλουμε στον server ως δεδομένα. Μερικοί τύποι είναι οι εξής: "application/json", "text/plain", "text/html" κ.α. και διατυπώνεται ως εξής:

```
"Content-Type: <type>"
```

Κορμός (body)

Ο κορμός, περιέχει ουσιαστικά τα δεδομένα που πρόκειται να στείλουμε. Τα δεδομένα θα πρέπει να είναι σε μορφή string πριν γίνει η αποστολή τους στον server, καθώς θα πρέπει επίσης το συντακτικό τους να ακολουθεί κάποιους κανόνες σχετικά με το είδος τους, όπως αναφέρθηκε και παραπάνω(π.χ. μορφές JSON, plain text, HTML κ.ο.κ).

```
1  curl -X POST \
2
3  https://my-awesome-site.com/my-path \
4      root-endpoint path
5  -H 'Content-Type: application/json' \
6
7  -H 'Authorization: Bearer eyJhbGciOi...' \
8      token
9  -d '{"foo": "bar"}'
    stringified json value
    body
```

headers

Σχήμα 3.8.1: Συντακτικό ενός αιτήματος *POST*, χρησιμοποιώντας την εντολή τερματικού “curl” για να σταλεί το αίτημα.

Κεφάλαιο 4.

Σχεδίαση &

Υλοποίηση

4.1 Σχεδίαση και αρχιτεκτονική λογισμικού

Σε αυτό το κεφάλαιο θα περιγράψουμε αναλυτικά την αρχιτεκτονική δομή και σχεδίαση της διαδικτυακής εφαρμογής που υλοποιήσαμε. Αρχικά, θα περιγράψουμε την διαδικασία της αυθεντικοποίησης που χρησιμοποιείται για την είσοδο των χρηστών στο σύστημα. Στην συνέχεια, θα εξηγήσουμε τον τρόπο που γίνεται η επικοινωνία μεταξύ πελάτη και εξυπηρετητή μέσω του API που υλοποιήσαμε. Τέλος, θα εξηγήσουμε τον τρόπο που συνδέεται η εφαρμογή με τα συστήματα αποθήκευσης νέφους (Google Drive, Dropbox, OneDrive) σε συνδιασμό με τους αλγόριθμους που χρησιμοποιήσαμε για τον σκοπό αυτό.

4.1.1 Αρχιτεκτονική της εφαρμογής [\[9\]](#)

Η διαδικτυακή εφαρμογή που υλοποιήσαμε, βασίζεται στην τριεπίπεδη αρχιτεκτονική (Σχήμα 4.1.1.1). Αυτό σημαίνει ότι υλοποιούνται τρία επίπεδα ανεξάρτητα μεταξύ τους που το καθένα από αυτά είναι υπεύθυνο για διαφορετικές λειτουργίες. Παρόλο που είναι ανεξάρτητα, τα τρία αυτά επίπεδα επικοινωνούν μεταξύ τους είτε άμεσα, είτε έμμεσα.

Επίπεδο Παρουσίασης (Presentation Tier)

Το πρώτο επίπεδο, ονομάζεται «επίπεδο παρουσίασης» (presentation tier) και είναι υπεύθυνο για την αλληλεπίδραση του χρήστη με την εφαρμογή μέσω της γραφικής διεπαφής που προβάλλεται στον περιηγητή του χρήστη. Σε αυτό το επίπεδο χρησιμοποιήσαμε τη βιβλιοθήκη ReactJS για να υλοποιήσουμε την γραφική διεπαφή.

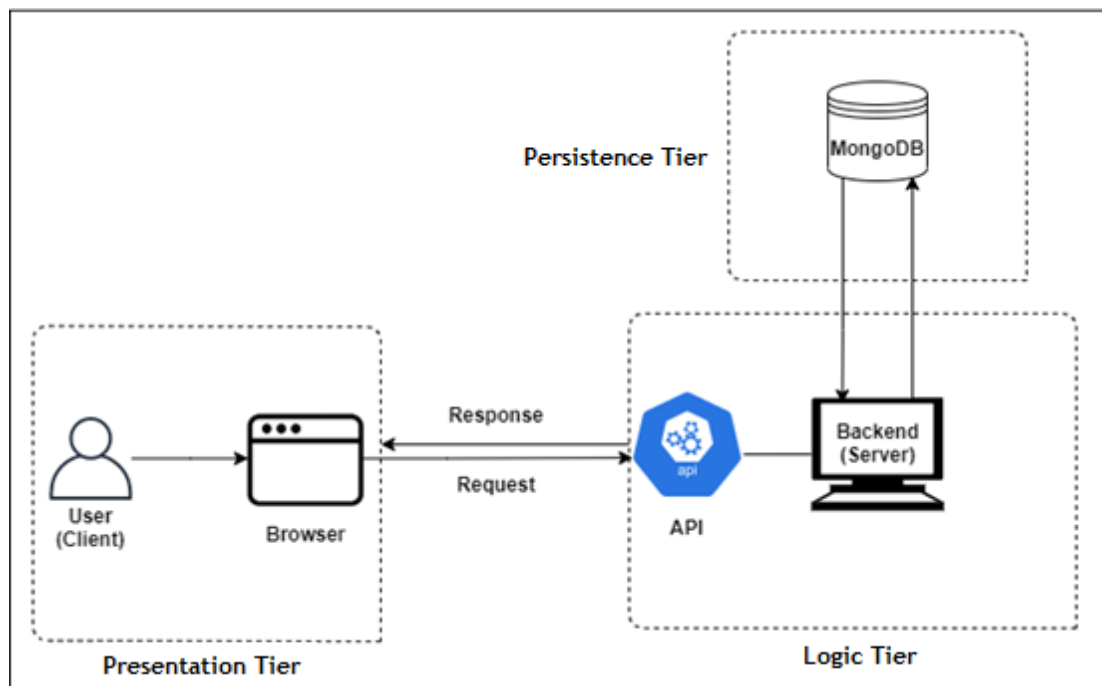
Λογικό Επίπεδο (Logic Tier)

Το δεύτερο επίπεδο, ονομάζεται «λογικό επίπεδο» (logic tier), εφόσον σε αυτό το επίπεδο υλοποιείται όλη η λογική και οι λειτουργίες της εφαρμογής. Στο επίπεδο αυτό, υλοποιήσαμε τον εξυπηρετητή(server), ο οποίος είναι αρμόδιος για την επικοινωνία του

πρώτου επιπέδου με το δεύτερο επίπεδο. Σε αυτή την αρμοδιότητα, συμβάλλει η λειτουργία του API, μέσω του οποίου γίνεται ξεκάθαρο ως προς τί λειτουργία πρέπει να εκτελέσει ο εξυπηρετητής, ανάλογα με το τί πόρους ζητάει ο πελάτης(client). Στο «επίπεδο εφαρμογής», υλοποιείται επίσης η επικοινωνία με τη βάση δεδομένων. Για την υλοποίηση του εξυπηρετητή, χρησιμοποιείται το framework Node.JS καθώς για την υλοποίηση του API, χρησιμοποιείται το framework Express.js

Επίπεδο Διατήρησης (Persistence Tier)

Το τρίτο – και τελευταίο – επίπεδο, ονομάζεται «επίπεδο διατήρησης». Σε αυτό το επίπεδο, αποθηκεύονται τα δεδομένα με την χρήση της βάσης δεδομένων MongoDB. Το τρίτο επίπεδο, επικοινωνεί με το δεύτερο επίπεδο, με σκοπό να έχει έμμεση πρόσβαση ο πελάτης στα δεδομένα που ζητάει.



Σχήμα 4.1.1.1: Σχηματική απεικόνιση της τριεπίπεδης αρχιτεκτονικής της διαδικτυακής εφαρμογής.

4.1.1.1 Δομή αρχείων

Σε αυτή την υπο-ενότητα θα περιγράψουμε την δομή των αρχείων της εφαρμογής. Γενικά, η εφαρμογή όσον αφορά τα αρχεία, απαρτίζεται από δύο κύρια τμήματα, το τμήμα του πελάτη (frontend) και το τμήμα του εξυπηρετητή (server) όπου το καθένα είναι μία ανεξάρτητη εφαρμογή και εμπεριέχεται σε έναν ξεχωριστό φάκελο.

Ο φάκελος του πελάτη ονομάζεται "client" και περιέχει τον κώδικα υλοποίησης της γραφικής διεπαφής. Σε αυτόν τον φάκελο περιέχονται αρκετοί υπο-φακέλοι όπου περιέχουν απαιτούμενα αρχεία (dependencies) τα οποία δημιουργήθηκαν αυτόματα κατά την δημιουργία της κενής εφαρμογής, καθώς και τρεις βασικοί υπο-φακέλοι που περιέχουν τον δικό μας κώδικα. Οι υπο-φακέλοι είναι οι εξής:

- **"components"**: Περιέχει επαναχρησιμοποιήσιμα κομμάτια κώδικα τα οποία χρησιμοποιούνται δυναμικά σε πολλά μέρη των σελίδων στον φάκελο **"pages"**
- **"pages"**: Περιέχει την υλοποίηση των τριών βασικών σελίδων της εφαρμογής:
 - **"LandingPage"**: Η αρχική σελίδα της εφαρμογής
 - **"DrivePage"**: Η κύρια σελίδα της εφαρμογής
 - **"ErrorPage"**: Η σελίδα σφάλματος
- **"utilities"**: Περιέχει βοηθητικές συναρτήσεις όπως για παράδειγμα την λογική της αποστολής ενός αιτήματος από τον πελάτη προς τον εξυπηρετητή, συναρτήσεις επεξεργασίας αλφαριθμητικών (parsers), σταθερές, κ.ο.κ.

Ο φάκελος του εξυπηρετητή ονομάζεται "api" και περιέχει τον κώδικα που χρειάζεται για να εκκινηθεί ο εξυπηρετητής, να διαχειρίζεται τα αιτήματα από τον πελάτη, να συνδέει τα μέσα αποθήκευσης νέφους, να γίνεται η επικοινωνία με τα μέσα αποθήκευσης νέφους καθώς και η επικοινωνία με την βάση δεδομένων. Οι τρεις βασικοί υποφάκελοι είναι οι εξής:

- **"routes"**: Περιέχει τόσα αρχεία όσα και τα τελικά σημεία (endpoints) του API. Κάθε αρχείο δηλαδή, «τρέχει» όταν γίνει αίτημα προς τον εξυπηρετητή ίδιο με το όνομα του αρχείου αυτού.
- **"drive-handlers"**: Περιέχει τα αρχεία στα οποία υλοποιείται η λογική σύνδεσης με τα μέσα αποθήκευσης νέφους "Google Drive", "Dropbox", "OneDrive".
- **"utilities"**: Περιέχει τα αρχεία βοηθητικών συναρτήσεων καθώς και ένα βασικό αρχείο στο οποίο υλοποιούνται οι μέθοδοι μέσω των οποίων γίνεται επικοινωνία με την βάση δεδομένων

4.1.2 Σύστημα αυθεντικοποίησης χρήστη

Μίας από τις βασικότερες λειτουργίες της εφαρμογής μας, είναι η εγγραφή και η σύνδεση ενός χρήστη στο σύστημα, καθώς και η διατήρηση της σύνδεσης του μέχρι να αποσυνδεθεί. Για την ανάγκη αυτή, χρειάστηκε η συνεργασία όλων των επιμέρους τμημάτων της εφαρμογής, δηλαδή του frontend, του backend και της βάσης δεδομένων.

Εγγραφή χρήστη στο σύστημα

Κατά την επίσκεψη ενός χρήστη στην εφαρμογή, του ζητείται η εγγραφή του στο σύστημα. Αφότου ο χρήστης εισάγει έγκυρη – ως προς την μορφή – διεύθυνση ηλεκτρονικού ταχυδρομείου και έναν προσωπικό κωδικό πρόσβασης, ακολουθεί η εξής ροή γεγονότων:

1. Αποστολή της διεύθυνσης ηλεκτρονικού ταχυδρομείου και κωδικού πρόσβασης στον εξυπηρετητή μέσω αιτήματος στο δικό μας API.
2. Κατακερματισμός του κωδικού πρόσβασης (hashing).
3. Δημιουργία μιας **μοναδικής** οντότητας χρήστη με τα στοιχεία του ως πεδία, καθώς και πρόσθετα πεδία που εξυπηρετούν άλλες λειτουργίες που θα εξηγηθούν αργότερα.
4. Αποθήκευση της οντότητας στην βάση δεδομένων.
5. Αποστολή μηνύματος από το backend και εμφάνιση του στο frontend σχετικά με το αν πέτυχε η δημιουργία χρήστη.

```
_id: ObjectId("5ebd90787bc2193a0897a579")
> virtualdrive: Object
  email: "bbb@gmail.com"
  password: "$2b$10$NGXG5YcwNP.ZLjEM2JsZw.HVPJ.FTPKaJowj6ghpfEhbk8pRi1lGC"
> google-drive: Array
> dropbox: Array
> onedrive: Array
__v: 0
```

Σχήμα 4.1.2.1: Η οντότητα ενός χρήστη όπως αποθηκεύεται στην βάση δεδομένων MongoDB.

Σύνδεση χρήστη στο σύστημα

Όταν ο χρήστης εισάγει τα στοιχεία του στην φόρμα για σύνδεση και αφότου αυτά είναι έγκυρα όπως προς την μορφή, ακολουθεί η εξής ροή γεγονότων:

1. Αποστολή της διεύθυνσης ηλεκτρονικού ταχυδρομείου και κωδικού πρόσβασης στον εξυπηρετητή μέσω αιτήματος στο δικό μας API.
2. Κατακερματισμός του κωδικού πρόσβασης (hashing).
3. Έλεγχος στην βάση δεδομένων για το αν η διεύθυνση ηλεκτρονικού ταχυδρομείου και ο κατακερματισμένος κωδικός πρόσβασης αντιστοιχούν σε κάποιον χρήστη.
4. Αποστολή μηνύματος από το backend σχετικά με το αν ο έλεγχος ήταν επιτυχής.

Αν ήταν επιτυχής τότε:

- 4.1 Γίνεται ανακατεύθυνση στην βασική σελίδα της εφαρμογής.
- 4.2 Αποθηκεύεται “cookie” στον περιηγητή (browser) με κρυπτογραφημένα τα στοιχεία του χρήστη, έτσι ώστε σε περίπτωση που γίνει ανανέωση στην σελίδα ή κλείσιμο της σελίδας, ο χρήστης να μην χρειαστεί να εισάγει τα στοιχεία του εκ νέου.

Αν δεν ήταν επιτυχής:

- 4.3 Εμφανίζεται μήνυμα ότι τα στοιχεία που εισήχθησαν δεν αντιστοιχούν σε κάποιον χρήστη.

Αποσύνδεση χρήστη από το σύστημα

Για την αποσύνδεση του χρήστη από το σύστημα, ακολουθεί η εξής ροή γεγονότων:

1. Διαγραφή του “cookie” που περιέχει τα στοιχεία του χρήστη από τον περιηγητή και είναι αποθηκευμένο στον περιηγητή.
2. Ανακατεύθυνση του χρήστη στην αρχική σελίδα.

Εφόσον το cookie παύει να υπάρχει, αν ο χρήστης προσπαθήσει να μπει στην σελίδα που βλέπει τα αρχεία του, εμφανίζεται μήνυμα σφάλματος.

4.1.3 Επικοινωνία με τα μέσα αποθήκευσης νέφους

Για να πετύχουμε την πρόσβαση στα αρχεία που βρίσκονται στα απομακρυσμένα μέσα αποθήκευσης νέφους, θα χρειαστεί ο εξυπηρετητής μας να μπορεί να επικοινωνεί με αυτά τα μέσα, μέσω των APIs που αυτά προσφέρουν. Επομένως, η εφαρμογή θα πρέπει

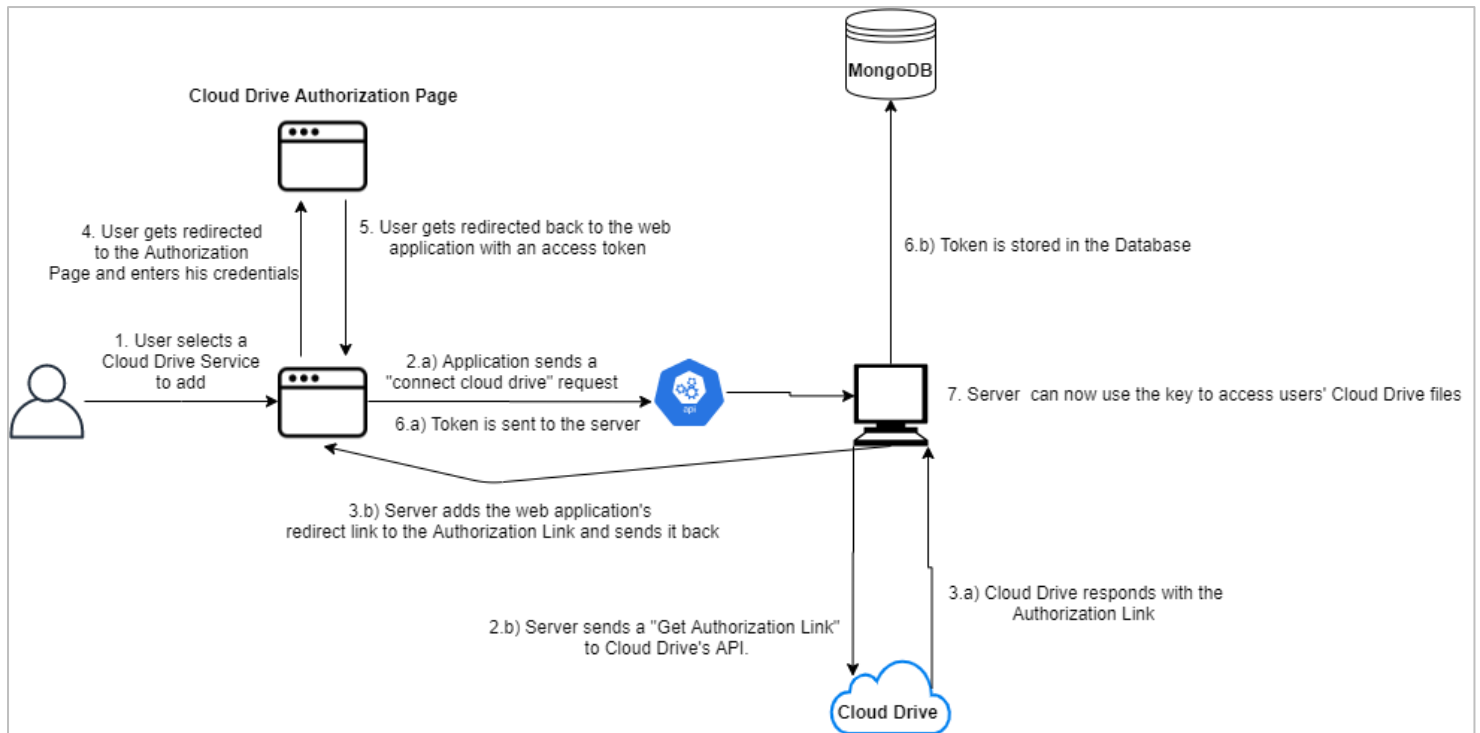
για κάθε χρήστη ξεχωριστά, να κρατάει πληροφορία σχετικά με τα μέσα αποθήκευσης νέφους που ο πελάτης έχει συνδέσει με την εφαρμογή και με την χρήση αυτής της πληροφορίας να μπορεί να έχει πρόσβαση στα απομακρυσμένα του αρχεία. Η ροή γεγονότων της λογικής που ακολουθήσαμε για την επίτευξη αυτής της λειτουργίας αναλύεται ως εξής:

1. Ο χρήστης επιλέγει μία από τις 3 υποστηριζόμενες υπηρεσίες αποθήκευσης νέφους (Google Drive, Dropbox, OneDrive) μέσω της γραφικής διεπαφής της εφαρμογής μας.
2. α) Η εφαρμογή μας κάνει κατάλληλο αίτημα στο API που έχουμε υλοποιήσει.
β) Στη συνέχεια ο εξυπηρετητής κάνει αίτημα στο API του μέσου αποθήκευσης νέφους για την λήψη της σελίδας στην οποία θα πρέπει να εισάγει τα στοιχεία του για να πάρει πρόσβαση η εφαρμογή στα δεδομένα του χρήστη.
3. α) Το μέσο αποθήκευσης νέφους αποστέλνει την διεύθυνση της σελίδας. β) Ο εξυπηρετητής, αφότου λάβει το URL της σελίδας, τοποθετεί στο URL ένα πεδίο με τη διεύθυνση ανακατεύθυνσης πίσω στην εφαρμογή μας και το αποστέλλει μέσω απάντησης στον φυλλομετρητή του πελάτη.
4. Αφότου η εφαρμογή μας λάβει το URL, τον ανακατευθύνει στη σελίδα της υπηρεσίας αποθήκευσης νέφους για να εισάγει τα στοιχεία του, έτσι ώστε να μπορεί να γίνει η πρόσβαση στα δεδομένα του.
5. Αφότου ο χρήστης εισάγει τα στοιχεία του, η υπηρεσία αποθήκευσης νέφους ανακατευθύνει τον χρήστη πίσω στην εφαρμογή μας (μέσω του URL που προστέθηκε στο τρίτο βήμα) μαζί με ένα κλειδί που δίνει πρόσβαση στα αρχεία του.
6. α) Το κλειδί αυτό στέλνεται στον εξυπηρετητή. β) Στην συνέχεια αποθηκεύεται στο «έγγραφο» στην βάση δεδομένων, που αντιστοιχεί στον τρέχοντα χρήστη, από τον εξυπηρετητή.
7. Στην συνέχεια ο εξυπηρετητής, χρησιμοποιώντας το κλειδί, στέλνει αίτημα στο API του μέσου αποθήκευσης νέφους για δημιουργία ενός νέου φακέλου. Στον φάκελο αυτό, θα αποθηκεύονται τα αρχεία που θα ανεβάζει ο χρήστης στο

εικονικό σύστημα νέφους (περισσότερα για το εικονικό σύστημα νέφους, θα αναλυθούν στην επόμενη ενότητα 4.1.4).

8. Ο χρήστης ζητάει πρόσβαση στα απομακρυσμένα αρχεία του μέσω αιτήματος στο API που υλοποιήσαμε.
9. Ο εξυπηρετητής, βλέπει μέσω του αιτήματος από ποιον χρήστη προήλθε το αίτημα και σε ποιο μέσο αποθήκευσης νέφους απευθύνεται, επικοινωνεί με την βάση δεδομένων και βρίσκει το «έγγραφο» που αντιστοιχεί στον χρήστη αυτόν. Από το «έγγραφο» αυτό, παίρνει το κλειδί που αντιστοιχεί στο μέσο αποθήκευσης νέφους.
10. Τέλος, ο εξυπηρετητής χρησιμοποιώντας το κλειδί αυτό, κάνει αίτημα στο API του μέσου αποθήκευσης νέφους, παίρνει πρόσβαση στα αρχεία και επιστρέφει μέσω απάντησης την πληροφορία για αυτά τα αρχεία στον φυλλομετρητή.
11. Η πληροφορία αυτή προβάλλεται στην γραφική διεπαφή και ο χρήστης μπορεί να διαχειριστεί τα αρχεία του.

Στο σχήμα **4.1.3.1** απεικονίζεται η διαδικασία που αντιστοιχεί στα παραπάνω πρώτα επτά βήματα, η οποία ονομάζεται “**oAuth2 Authorization Flow**” και αποτελεί τον πιο συνηθισμένο και απλό τρόπο για την σύνδεση εφαρμογών διαδικτύου με τα APIs άλλων εξωτερικών εφαρμογών με σκοπό την πρόσβαση σε προσωπικούς, απομακρυσμένους πόρους.



Σχήμα 4.1.3.1: Σχηματική απεικόνιση του “oAuth2 Authorization Flow” που υλοποιεί η εφαρμογή.

4.1.4 Εικονικό σύστημα αποθήκευσης νέφους

Η βασικότερη λειτουργία της διαδικτυακής εφαρμογής μας, είναι η δυνατότητα ύπαρξης ενός εικονικού και ενοποιημένου μέσου αποθήκευσης αρχείων στο οποίο θα υπάρχουν αρχεία από όλα τα μέσα αποθήκευσης νέφους που έχει συνδέσει ο χρήστης. Πιο συγκεκριμένα, το εικονικό αυτό σύστημα προσφέρει τις εξής δυνατότητες:

- Δυνατότητα ο χρήστης να ανεβάζει αρχεία, χωρίς να χρειαστεί να επιλέξει σε ποιο συνδεδεμένο μέσο αποθήκευσης νέφους θα σώζονται τα αρχεία. Την απόφαση για το κατάλληλο μέσο αποθήκευσης ανάλογα με τον υπολειπόμενο χώρο, θα την παίρνει η εφαρμογή.
- Δημιουργία εικονικών φακέλων που θα περιέχουν αρχεία από διαφορετικά μέσα αποθήκευσης νέφους (π.χ. ένας εικονικός φάκελος θα μπορεί να περιέχει αρχεία του Google Drive, του Dropbox και του OneDrive ταυτόχρονα).
- Διαμοιρασμός εικονικών φακέλων μεταξύ εγγεγραμμένων – στην εφαρμογή μας – χρηστών.

- Δυνατότητα να μπορούν πολλαπλοί χρήστες να διαχειρίζονται τα αρχεία ενός διαμοιρασμένου εικονικού φακέλου.

Οι τεχνικές που εφαρμόσαμε για την επίτευξη των παραπάνω λειτουργιών θα εξηγηθούν αναλυτικά στις υποενότητες που ακολουθούν.

4.1.4.1 Λειτουργίες διαχείρισης αρχείων

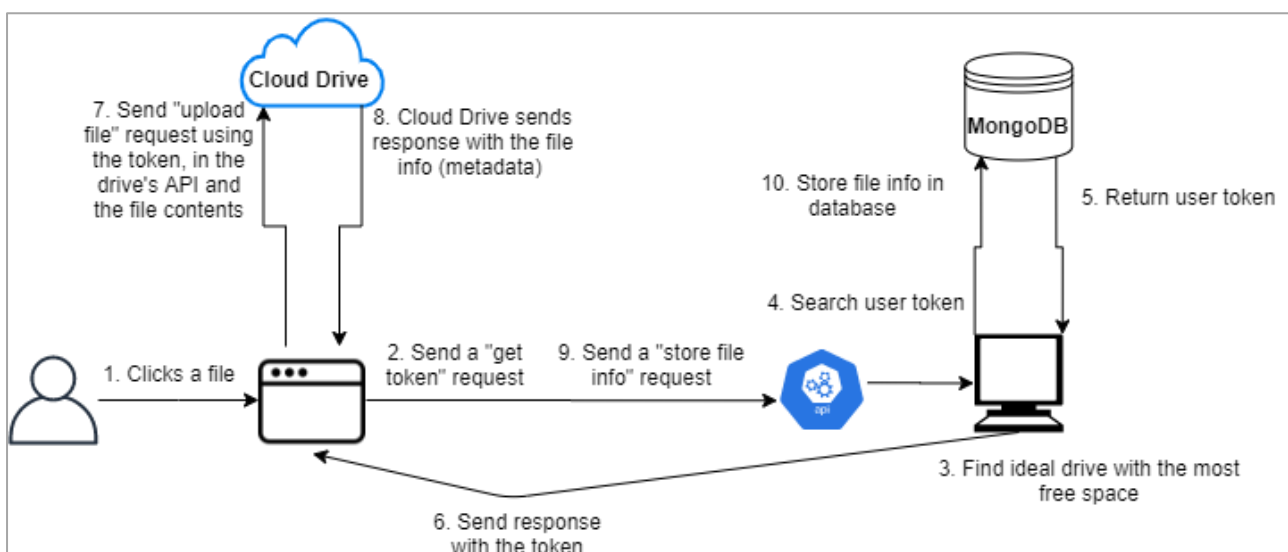
Σε αυτή την υποενότητα, θα εξηγήσουμε πως ακριβώς υλοποιήθηκαν οι διάφορες λειτουργίες του εικονικού συστήματος αποθήκευσης, που σχετίζονται με την διαχείριση των προσωπικών αρχείων ενός χρήστη.

Ανέβασμα αρχείου (Σχήμα 4.1.4.1.1)

Όταν ο χρήστης πρόκειται να ανεβάσει ένα αρχείο στο εικονικό σύστημα αποθήκευσης νέφους, ακολουθεί η εξής ροή γεγονότων:

1. Ο χρήστης ανοίγει τον περιηγητή αρχείων και στην συνέχεια επιλέγει ένα αρχείο για ανέβασμα (ή σέρνει το αρχείο στην εφαρμογή).
2. Αποστέλλεται από τον φυλλομετρητή ένα αίτημα στο API μας. Το αίτημα αυτό αποσκοπεί στην εύρεση του ιδανικού μέσου αποθήκευσης, δεδομένου ότι ο χρήστης έχει συνδέσει με την εφαρμογή περισσότερα από ένα μέσα αποθήκευσης νέφους.
3. Ο εξυπηρετητής αφότου λάβει το αίτημα από τον φυλλομετρητή, στέλνει κατάλληλα αιτήματα σε όλα τα μέσα αποθήκευσης νέφους που έχει συνδέσει ο χρήστης και μέσω απαντήσεων μαθαίνει τους ελεύθερους χώρους που αυτά έχουν. Στην συνέχεια, επιλέγει το μέσο με τον περισσότερο ελεύθερο χώρο ως το ιδανικότερο μέσο.
4. Στη συνέχεια, αφότου βρήκε το ιδανικότερο μέσο αποθήκευσης νέφους, επικοινωνεί με την βάση δεδομένων με σκοπό να βρεί το κλειδί που αντιστοιχεί στο ιδανικότερο αυτό μέσο.

5. Η βάση δεδομένων επιστρέφει το κλειδί του ιδανικότερου μέσου, στον εξυπηρετητή.
6. Ο εξυπηρετητής, στέλνει μέσω απάντησης στον φυλλομετρητή του χρήστη, το κλειδί.
7. Ο φυλλομετρητής, εφόσον είναι ο μόνος που έχει πληροφορία για το αρχείο που πρόκειται να ανέβει, εκτελεί απευθείας αίτημα στο API του μέσου αποθήκευσης νέφους, χρησιμοποιώντας το κλειδί που έστειλε ως απάντηση ο εξυπηρετητής.
8. Το API του μέσου αποθήκευσης νέφους, στέλνει με απάντηση την πληροφορία για το αν το ανέβασμα του αρχείου ήταν επιτυχές, καθώς και την πληροφορία του αρχείου αυτού.
9. Ο φυλλομετρητής στέλνει αίτημα στο API μας με την πληροφορία του αρχείου.
10. Ο εξυπηρετητής, αφού λάβει την πληροφορία του αρχείου, την μετασχηματίζει κατάλληλα έτσι ώστε ανεξαρτήτα από το μέσο αποθήκευσης νέφους που ανήκει το αρχείο αυτό, η πληροφορία να είναι στην ίδια ακριβώς μορφή (Η μορφή αυτή θα εξηγηθεί στην ενότητα 4.1.5 όπου θα αναλύσουμε εκτενώς το σχήμα της βάσης δεδομένων) και επικοινωνεί με την βάση για να καταχωρήσει την εγγραφή με την πληροφορία του αρχείου στο έγγραφο του πελάτη που ανέβασε το αρχείο.

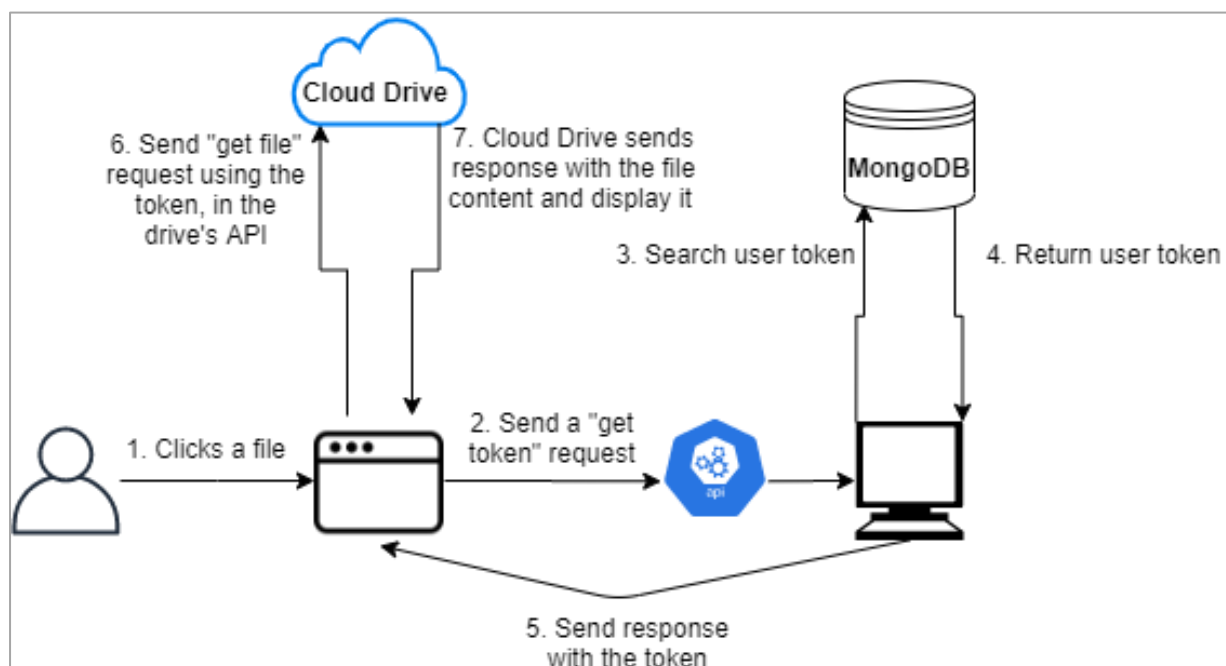


Σχήμα 4.1.4.1.1: Τα βήματα ανεβάσματος ενός αρχείου στο εικονικό σύστημα νέφους

Κατέβασμα αρχείου (Σχήμα 4.1.4.1.2)

Όταν ο χρήστης πρόκειται να κατεβάσει ένα αρχείο από το εικονικό σύστημα αποθήκευσης νέφους, ακολουθεί η εξής ροή γεγονότων:

1. Αρχικά ο χρήστης επιλέγει ένα αρχείο το οποίο επιθυμεί να κατεβάσει.
2. Αποστέλλεται από τον φυλλομετρητή ένα αίτημα στο API με σκοπό ο εξυπηρετητής να βρει το κλειδί του μέσου αποθήκευσης νέφους στο οποίο ανήκει το αρχείο προς κατέβασμα
3. Ο εξυπηρετητής, επικοινωνεί με την βάση δεδομένων, και εκτελεί αναζήτηση για το κλειδί μέσου αποθήκευσης νέφους στο οποίο ανήκει το αρχείο προς κατέβασμα.
4. Η βάση δεδομένων, επιστρέφει το κλειδί στον εξυπηρετητή.
5. Στην συνέχεια, ο εξυπηρετητής αποστέλλει το κλειδί μέσω απάντησης στον φυλλομετρητή.
6. Εφόσον το αρχείο θα πρέπει να κατέβει στον προσωπικό υπολογιστή του πελάτη, ο φυλλομετρητής θα χρησιμοποιήσει το κλειδί για να στείλει απευθείας αίτημα στο API του μέσου αποθήκευσης νέφους.
7. Το API του μέσου αποθήκευσης νέφους, αποστέλλει με απάντηση το αρχείο σε δυαδική μορφή και η εφαρμογή στον φυλλομετρητή, συνθέτει το αρχείο στην κατάλληλη μορφή με βάσει την προέκταση του αρχείου και το σώζει τοπικά στο σύστημα του χρήστη



Σχήμα 4.1.4.1.2: Τα βήματα κατεβάσματος ενός αρχείου στο εικονικό σύστημα νέφους

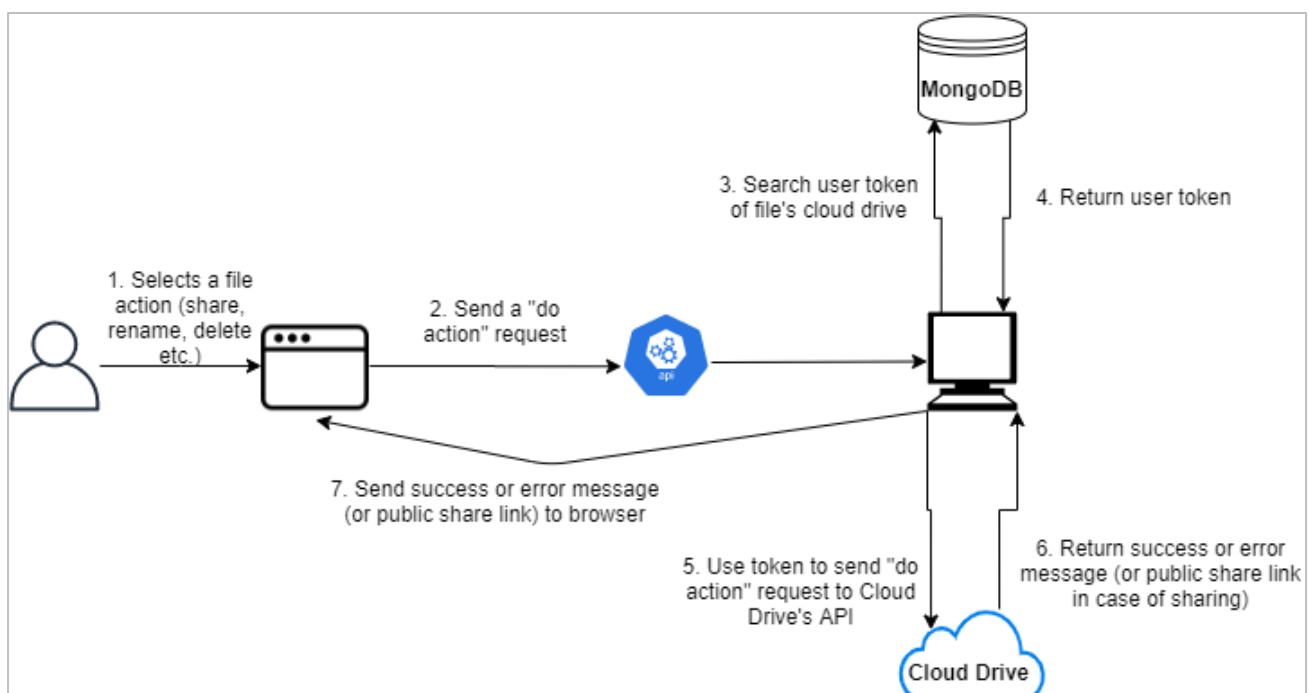
Διαγραφή/Μετονομασία/Διαμοιρασμός αρχείου (Σχήμα 4.1.4.1.3)

Οι λειτουργίες της διαγραφής, μετονομασίας και δημόσιου διαμοιρασμού ενός αρχείου ακολουθούν παρόμοια ροή γεγονότων:

1. Ο χρήστης επιλέγει να εκτελέσει μία από τις λειτουργίες σε ένα αρχείο.
2. Στη συνέχεια, αποστέλλεται αίτημα στο API μας με το αναγνωριστικό αρχείου (file id) καθώς και πληροφορία για το μέσο αποθήκευσης νέφους στο οποίο ανήκει αυτό το αρχείο (στην περίπτωση της μετονομασίας, αποστέλλεται επίσης και το νέο όνομα του αρχείου).
3. Ο εξυπηρετητής, αφού λάβει το αίτημα, επικοινωνεί με την βάση δεδομένων και κάνει αναζήτηση για το κλειδί του μέσου αποθήκευσης νέφους στο οποίο ανήκει το αρχείο.
4. Η βάση δεδομένων επιστρέφει το κλειδί στον εξυπηρετητή.

5. Ο εξυπηρετητής, χρησιμοποιώντας το κλειδί αυτό, αποστέλλει κατάλληλο αίτημα στο API του μέσου αποθήκευσης νέφους για να ολοκληρωθεί η ζητούμενη λειτουργία.
6. Το API του μέσου αποθήκευσης νέφους, επιστρέφει με απάντηση ένα μήνυμα επιτυχίας ή αποτυχίας (ή τον δημόσιο σύνδεσμο πρόσβασης αρχείου στην περίπτωση του διαμοιρασμού)*.
7. Τέλος, αποστέλλεται στον φυλλομετρητή μέσω απάντησης επίσης η πληροφορία για το αν η ζητούμενη λειτουργία πέτυχε ή απέτυχε (στην περίπτωση του διαμοιρασμού αποστέλλεται ο δημόσιος σύνδεσμος πρόσβασης του αρχείου).

*Στην περίπτωση αιτήματος διαγραφής αρχείου, διαγράφονται επίσης οι πληροφορίες του από την βάση καθώς και το ίδιο το αρχείο από το αντίστοιχο μέσο αποθήκευσης νέφους που ανήκει.



Σχήμα 4.1.4.1.3: Τα βήματα ενεργειών (μετονομασία, διαγραφή, διαμοιρασμός) ενός αρχείου στο εικονικό σύστημα νέφους

4.1.4.2 Υλοποίηση εικονικών φακέλων

Όπως αναφέραμε και στην αρχή του κεφαλαίου, μία από τις απαιτούμενες λειτουργίες της διαδικτυακής εφαρμογής μας, είναι η δυνατότητα να μπορούμε να δημιουργούμε

εικονικούς φακέλους. Οι φάκελοι αυτοί, θα μπορούν να περιέχουν αρχεία από όλα τα υποστηριζόμενα μέσα αποθήκευσης νέφους (Google Drive, Dropbox, OneDrive).

Η λογική είναι, ότι ένας τέτοιος φάκελος θα αποτελεί απλώς μία εγγραφή στην βάση δεδομένων και θα περιέχει ένα αναγνωριστικό. Επομένως, κάθε φορά που ανεβαίνει κάποιο αρχείο στο εικονικό σύστημα νέφους και μέσα σε έναν εικονικό φάκελο θα ακολουθείται πάντα η εξής ροή γεγονότων:

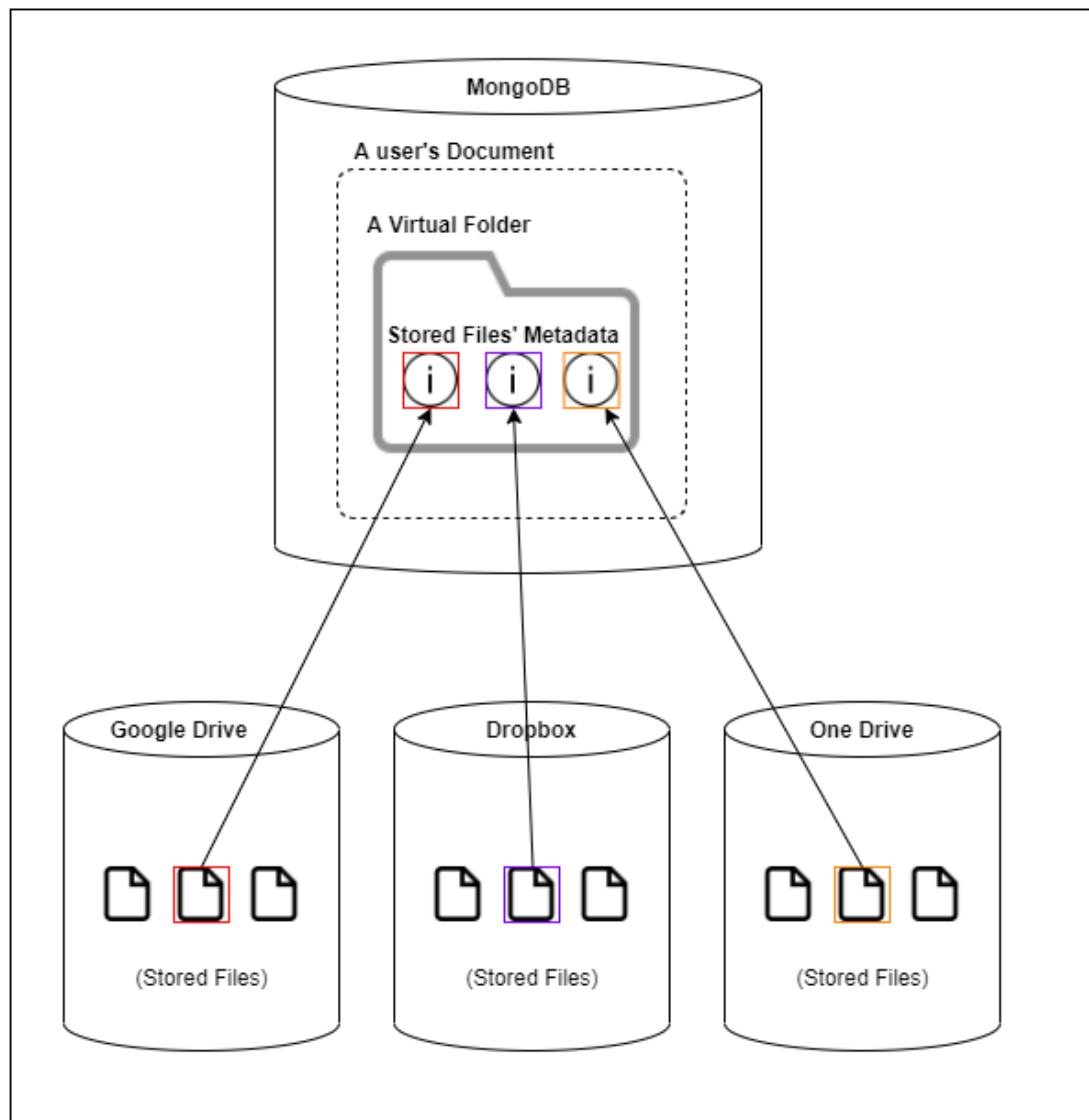
1. Το αρχείο πρώτα θα ανέβει στο μέσο αποθήκευσης νέφους (Google Drive, Dropbox, OneDrive) με την διαδικασία ανεβάσματος που αναφέραμε στην υποενότητα 4.1.4.1.
2. Στην συνέχεια, στην εγγραφή που δημιουργήθηκε για αυτό το αρχείο στην βάση δεδομένων, θα προσθέσουμε ένα πεδίο **“virtualParentId”** το οποίο θα περιέχει το αναγνωριστικό του εικονικού φακέλου που δημιουργήσαμε.

Σύμφωνα με αυτή την λογική, κάθε φορά που ο χρήστης ανοίγει έναν εικονικό φάκελο, ο φυλλομετρητής στέλνει κατάλληλο αίτημα στο API συμπεριλαμβάνοντας το αναγνωριστικό του εικονικού αυτού φακέλου. Ο εξυπηρετητής στην συνέχεια, προσπελαύνει όλες τις εγγραφές των αρχείων και των φακέλων που υπάρχουν στο έγγραφο του τρέχοντα χρήστη στην βάση και ελέγχει ποιες από αυτές έχουν το πεδίο **“virtualParentId”** ίδιο με το αναγνωριστικό εικονικού φακέλου που στάλθηκε στον εξυπηρετητή. Τέλος, ο εξυπηρετητής συλλέγει αυτές εγγραφές αρχείων και τις στέλνει με απάντηση στον φυλλομετρητή του πελάτη.

Είναι απαραίτητο επίσης να αναφέρουμε ότι κατά την διαγραφή ενός φακέλου, ακολουθεί παρόμοια λογική με την διαγραφή ενός αρχείου. Δηλαδή, γίνεται αρχικά διαγραφή των πληροφοριών αρχείων που περιέχει ο εικονικός φάκελος στην βάση δεδομένων και στην συνέχεια διαγράφονται τα αρχεία από τα αντίστοιχα μέσα αποθήκευσης νέφους που ανήκουν. Τέλος, αφαιρείται και η εγγραφή του εικονικού φακέλου από την βάση δεδομένων.

Στο σχήμα 4.1.4.2.1 γίνεται μια οπτική απεικόνιση της λογικής που εξηγήσαμε. Τα αρχεία που είναι χρωματισμένα, ανέβηκαν από έναν χρήστη σε έναν εικονικό φάκελο. Στην

πραγματικότητα όμως, τα αρχεία είναι ανεβασμένα στα μέσα αποθήκευσης νέφους. Μόνο οι πληροφορίες για τα αρχεία αυτά (metadata) είναι ουσιαστικά σωσμένες στην βάση δεδομένων και απλώς έχουν κοινό **“virtualParentId”**, με αποτέλεσμα ο χρήστης να βλέπει μέσω της γραφικής διεπαφής ότι τα αρχεία αυτά εμπεριέχονται στον ίδιο εικονικό φάκελο.



Σχήμα 4.1.4.2.1: Σχηματική απεικόνιση της λογικής του εικονικού φακέλου.

4.1.4.3 Διαμοιρασμός εικονικών φακέλων

Εφόσον εξηγήσαμε αναλυτικά την λογική πίσω από τους εικονικούς φακέλους, σε αυτή την υποενότητα θα εξηγήσουμε μία ακόμη από τις απαιτούμενες λειτουργίες της εφαρμογής μας που είναι ο διαμοιρασμός των εικονικών αυτών φακέλων. Η λογική που ακολουθήσαμε για αυτή την λειτουργία είναι απλή. Κάθε φορά που ο ένας χρήστης

επιλέγει να μοιραστεί έναν εικονικό φάκελο με κάποιον άλλο χρήστη εισάγοντας την διεύθυνση ηλεκτρονικού ταχυδρομείου του (με την οποία έκανε εγγραφή στην εφαρμογή), ο φυλλομετρητής στέλνει αίτημα στο API που περιέχει αυτή την διεύθυνση καθώς και το αναγνωριστικό του εικονικού φακέλου. Στην συνέχεια, στην εγγραφή αυτού του φακέλου στην βάση δεδομένων, προσθέτει στο πεδίο **“sharedWithEmails”** την διεύθυνση αυτή. Με αυτό τον τρόπο, ο εικονικός φάκελος περιέχει πλέον την πληροφορία για τους χρήστες από τους οποίους θα πρέπει να είναι προσβάσιμος.

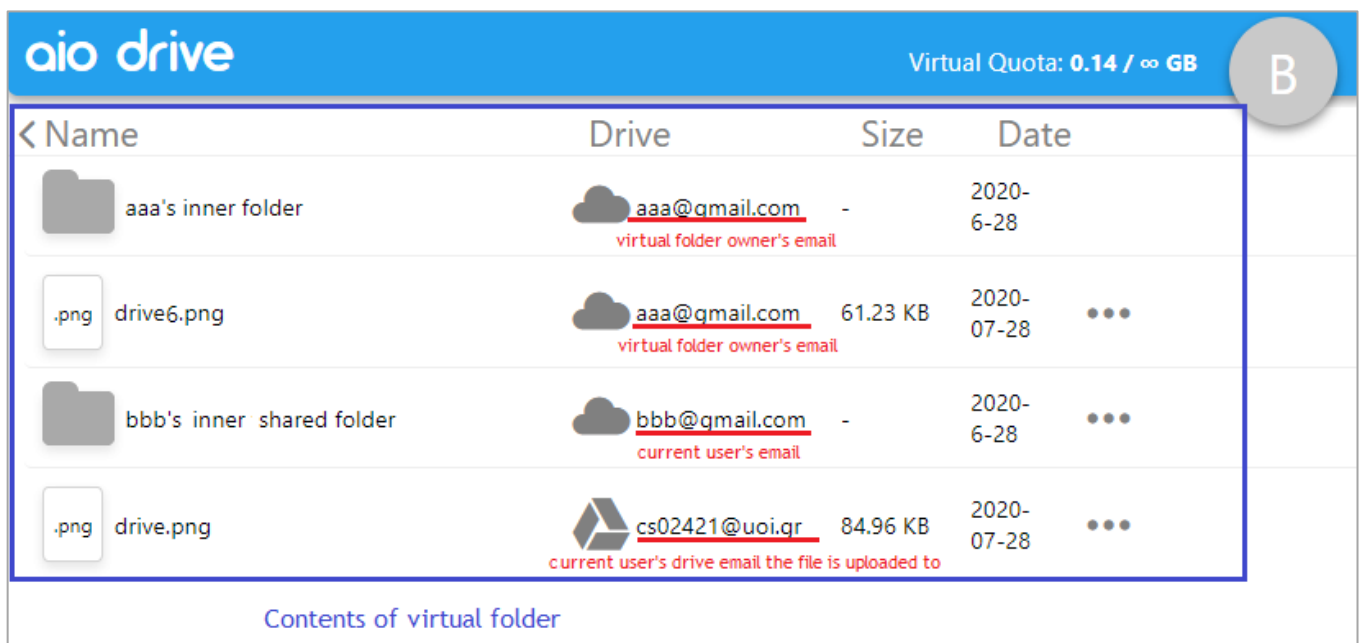
Είναι αναγκαίο επίσης να αναφέρουμε, ότι κάθε χρήστης έχει από προεπιλογή έναν μη-διαγράψιμο εικονικό φάκελο στο εικονικό σύστημα νέφους με όνομα **“Shared with me”**. Σε αυτόν τον φάκελο θα εμφανίζονται όλοι οι εικονικοί φάκελοι που έχουν μοιραστεί άλλοι χρήστες με αυτόν. Επομένως όταν ένας χρήστης με διεύθυνση ηλεκτρονικού ταχυδρομείου **«X»**, ανοίξει αυτόν τον φάκελο, στέλνεται αίτημα στο API μας και στην συνέχεια γίνεται αναζήτηση στην βάση δεδομένων για φακέλους άλλων χρηστών που περιέχουν στο πεδίο **“sharedWithEmails”** την διεύθυνση ηλεκτρονικού ταχυδρομείου **«X»**. Οι πληροφορίες αυτών των φακέλων στέλνονται στην συνέχεια στην γραφική διεπαφή του χρήστη μέσω απάντησης και έτσι επιτυγχάνεται το πρώτο βήμα του διαμοιρασμού.

Η ίδια σχεδόν λογική ακολουθεί και στην περίπτωση που ο χρήστης ανοίξει έναν διαμοιρασμένο εικονικό φάκελο. Σε αυτή την περίπτωση, στέλνεται από τον φυλλομετρητή αίτημα στο API μας, με σκοπό να βρεθούν όλα τα αρχεία χρηστών που συμμετέχουν στον διαμοιρασμένο αυτό φάκελο, δηλαδή όλα τα αρχεία και οι φάκελοι που έχουν πεδίο **“virtualParent”** ίσο με το αναγνωριστικό του διαμοιρασμένου φακέλου και να σταλούν στον φυλλομετρητή του χρήστη.

Τέλος, θα πρέπει να αναφέρουμε ότι όταν ένας χρήστης επιλέξει να ανοίξει ένα αρχείο ενός άλλου χρήστη που συμμετέχει στον διαμοιρασμό, το άνοιγμα αυτού του αρχείου θα γίνει μέσω λήψης ενός δημόσιου συνδέσμου του μέσου αποθήκευσης νέφους σε μία νέα καρτέλα. Αυτό γίνεται διότι για να έχουμε απευθείας πρόσβαση σε ένα αρχείο, θα χρειαστεί απευθείας κλήση από τον φυλλομετρητή στο API του μέσου αποθήκευσης νέφους, διαδικασία η οποία προϋποθέτει την χρήση κλειδιού για την αποστολή των αιτημάτων και αυτό δεν είναι καθόλου ασφαλές. Σαν χρήστες, δεν θα πρέπει να χρησιμοποιούμε τα κλειδιά άλλων χρηστών για να έχουμε πρόσβαση στα αρχεία τους. Στο σχήμα **4.1.4.3.2** φαίνεται σχηματικά πως γίνεται το άνοιγμα ενός αρχείου που ανήκει

στον τρέχον χρήστη και στο σχήμα 4.1.4.3.3 πως γίνεται το άνοιγμα ενός αρχείου που ανήκει στον χρήστη που συμμετέχει στον διαμοιρασμό του εικονικού φακέλου.

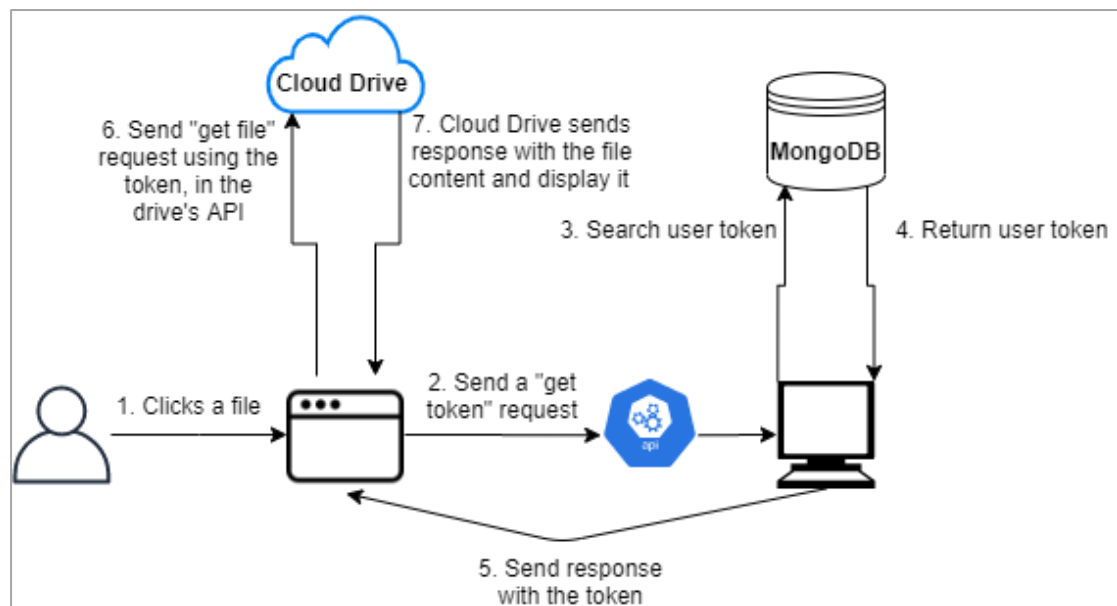
Στο σχήμα 4.1.4.3.1, εμφανίζονται τα περιεχόμενα ενός εικονικού φακέλου που μοιράστηκε ο χρήστης “aaa@gmail.com” με τον-τρέχοντα συνδεδεμένο – χρήστη “bbb@gmail.com”. Ο φάκελος αυτός όπως φαίνεται παρακάτω, περιέχει τόσο αρχεία του ενός χρήστη όσο και του άλλου χρήστη.



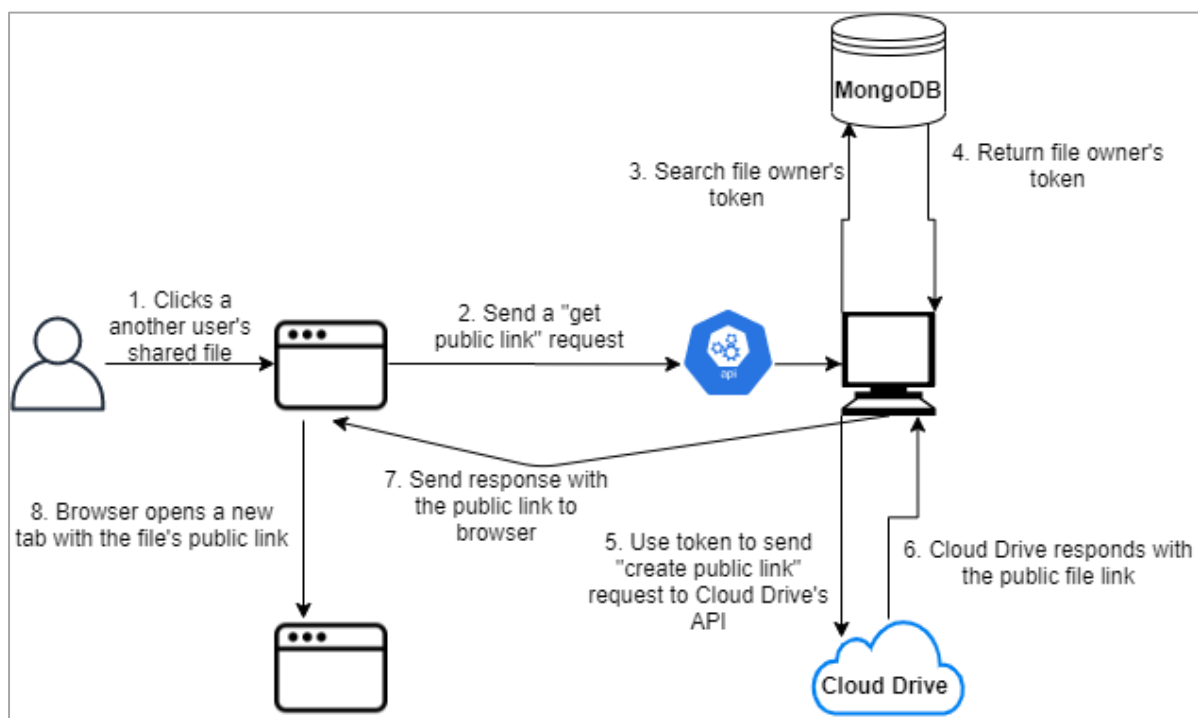
Name	Drive	Size	Date
aaa's inner folder	aaa@gmail.com virtual folder owner's email	-	2020-6-28
drive6.png	aaa@gmail.com virtual folder owner's email	61.23 KB	2020-07-28
bbb's inner shared folder	bbb@gmail.com current user's email	-	2020-6-28
drive.png	cs02421@uoi.gr current user's drive email the file is uploaded to	84.96 KB	2020-07-28

Contents of virtual folder

Σχήμα 4.1.4.3.1: Στιγμιότυπο της γραφικής διεπαφής στο εσωτερικό ενός διαμοιρασμένου εικονικού φακέλου



Σχήμα 4.1.4.3.2: Βήματα ανοίγματος αρχείου (του τρέχοντος χρήστη) που συμμετέχει σε διαμοιρασμένο εικονικό φάκελο



Σχήμα 4.1.4.3.3: Βήματα ανοίγματος αρχείου από τον τρέχοντα χρήστη, σε αρχείο του χρήστη που συμμετέχει στον διαμοιρασμένο εικονικό φάκελο

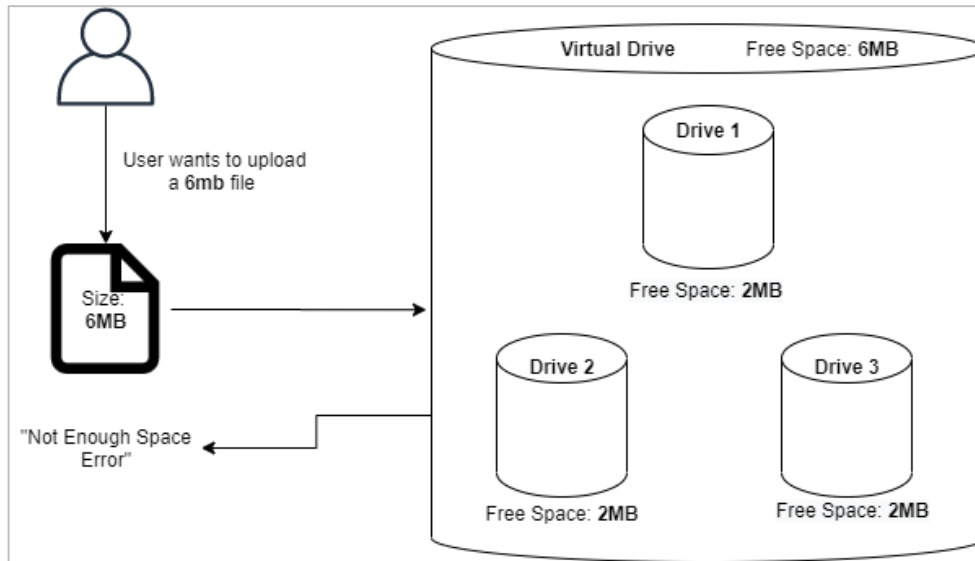
4.1.4.4 Αναδιάταξη αρχείων

Ένα από τα προβλήματα στα οποία μπορεί να οδηγηθεί η εφαρμογή είναι το σενάριο του εξωτερικού κατακερματισμού κατά την διαδικασία ανεβάσματος ενός αρχείου. Στο παρακάτω σχήμα 4.1.4.4.1 απεικονίζεται αυτό το σενάριο όπου ο χρήστης επιθυμεί να ανεβάσει ένα αρχείο εφόσον ο συνολικός ελεύθερος χώρος είναι μεγαλύτερος από το μέγεθος του αρχείου. Το πρόβλημα εδώ είναι ότι παρόλο που αθροιστικά υπάρχει αρκετός χώρος για να ανέβει το αρχείο, κανένα από τα συνδεδεμένα μέσα αποθήκευσης νέφους δεν έχουν αρκετό χώρο για να «φιλοξενήσουν» το αρχείο.

Για την προσεγγιστική λύση αυτού του προβλήματος, εφαρμόσαμε έναν αλγόριθμο αναδιάταξης των αρχείων. Τα βήματα του αλγορίθμου είναι τα εξής:

1. Επέλεξε ένα αρχείο “**x**” για ανέβασμα.
2. Βρες το μέσο αποθήκευσης νέφους “**max**” με τον περισσότερο ελεύθερο χώρο
3. Κάνε λήψη των μεγεθών των αρχείων του μέσου “**max**”.
4. Για κάθε άλλο μέσο αποθήκευσης νέφους “**other**” (δηλ. εκτός του “**max**”):
 - 4.1. Για κάθε αρχείο του “**max**”:
 - 4.1.1. Αν το αρχείο χωράει στο μέσο “**other**”:
 - 4.1.1.1. Μάρκαρε το για μετακίνηση στο μέσο “**other**” και αύξησε τον ελεύθερο χώρο του μέσου “**max**”.
 - 4.1.1.2. Αν το αρχείο “**x**” πλέον χωράει στο μέσο “**max**”:
 - 4.1.1.2.1. Κάνε τις μετακινήσεις των αρχείων που μαρκαρίστηκαν στο βήμα 4.1.2 και επέστρεψε. (Το αρχείο μπορεί πλέον να ανέβει στο μέσο “**max**”).
 - 4.1.1.3. Αλλιώς αν το αρχείο “**x**” δεν χωράει στο μέσο “**max**”, συνέχισε.
 - 4.1.2. Αλλιώς αν το αρχείο δεν χωράει στο μέσο “**other**” δοκίμασε το επόμενο αρχείο.

Με την εφαρμογή του παραπάνω αλγορίθμου γίνεται μία προσπάθεια αναδιάταξης των αρχείων με σκοπό να υπάρξει αρκετός ελεύθερος χώρος για να ανέβει τελικά το αρχείο. Ο παραπάνω αλγόριθμος δεν λειτουργεί πάντα εφόσον θα πρέπει να υπάρξει ένα σενάριο όπου τα αρχεία που εμπεριέχονται στα επιμέρους μέσα θα μπορούν να μετακινηθούν αρμονικά μεταξύ των μέσων με αποτέλεσμα να υπάρξει αρκετός ελεύθερος χώρος. Τα μεγέθη των αρχείων δεν είναι προβλέψιμα, επομένως για το πρόβλημα αυτό δεν μπορεί να υπάρξει κάποιος αλγόριθμος που να το λύνει πλήρως παρά μόνο προσεγγιστικά.



Σχήμα 4.1.4.4.1: Σχηματική απεικόνιση του προβλήματος του εξωτερικού κατακερματισμού κατά το ανέβασμα ενός αρχείου

4.1.5 Ανάλυση σχήματος βάσης δεδομένων

Σε αυτή την υποενότητα, θα εξηγήσουμε αναλυτικά το σχήμα της βάσης δεδομένων που ακολουθήσαμε για την υποστήριξη όλων των λειτουργιών που αναφέραμε στις προηγούμενες ενότητες. Όπως αναφέραμε στην ενότητα 4.1.2, κατά την εγγραφή ενός χρήστη στην βάση δεδομένων δημιουργείται ένα έγγραφο που αντιπροσωπεύει την οντότητα ενός χρήστη και αποθηκεύεται σε μία συλλογή από όμοιες τέτοιες οντότητες.

Το έγγραφο περιέχει ένα σύνολο πληροφοριών σχετικά με:

- Τα στοιχεία σύνδεσης του χρήστη.
- Τα κλειδιά των μέσων αποθήκευσης νέφους που έχει συνδέσει στην εφαρμογή.
- Τους εικονικούς φακέλους που έχει δημιουργήσει/μοιραστεί.
- Τα αρχεία που έχει ανεβάσει στο εικονικό σύστημα αποθήκευσης νέφους.

Η δομή του εγγράφου είναι τύπου JSON επομένως τα πεδία που περιέχει μπορούν να είναι οποιουδήποτε από τους συνηθισμένους τύπους (boolean, number, string, array, JSON κ.ο.κ.). Στην συνέχεια θα κάνουμε επεξήγηση όλων των πεδίων χωρίζοντας τα σε δύο κατηγορίες. Η πρώτη κατηγορία θα αφορά τα πεδία που σχετίζονται με τις προσωπικές πληροφορίες του χρήστη και η δεύτερη κατηγορία θα αφορά τα πεδία που σχετίζονται με το εικονικό σύστημα νέφους όπως τα αρχεία και οι φάκελοι που το συνθέτουν.

Πεδία προσωπικών πληροφοριών χρήστη

ID (string): Μοναδική ακολουθία χαρακτήρων που προσδιορίζουν αποκλειστικά και μόνο έναν μοναδικό χρήστη.

Email (string): Η διεύθυνση ηλεκτρονικού ταχυδρομείου με την οποία ο χρήστης έκανε εγγραφή στην εφαρμογή.

Password (string): Ο κατακερματισμένος κωδικός πρόσβασης του χρήστη που εισήγαγε κατά την εγγραφή του στην εφαρμογή.

Πεδία "Google-drive" – "Dropbox" – "OneDrive" (String Array):

Τα πεδία αυτά είναι το καθένα ένας πίνακας που περιέχει όλες τις πληροφορίες για τα μέσα αποθήκευσης νέφους που έχουμε συνδέσει με την εφαρμογή. Κάθε οντότητα του πίνακα αντιστοιχεί σε μία JSON τύπου εγγραφή με τα εξής υπο-πεδία:

- **aiofolder_id (String):** Είναι το αναγνωριστικό του φακέλου του μέσου αποθήκευσης νέφους, στον οποίο σώζονται τα αρχεία που ανεβαίνουν στο εικονικό σύστημα νέφους
- **email (String):** Η διεύθυνση ηλεκτρονικού ταχυδρομείου που αντιστοιχεί στο συνδεδεμένο μέσο αποθήκευσης νέφους.
- **token (String):** Το κρυπτογραφημένο κλειδί του μέσου αποθήκευσης νέφους, με την χρήση του οποίου γίνονται οι κλήσεις στο API του μέσου αποθήκευσης νέφους για την πρόσβαση στα προσωπικά αρχεία.

Πεδία σχετικά με το εικονικό σύστημα αποθήκευσης νέφους

Virtualdrive (JSON): Είναι το κύριο πεδίο που περιέχει όλες τις πληροφορίες που αφορούν το εικονικό σύστημα αποθήκευσης νέφους. Έχει μορφή JSON και περιέχει τα εξής υποπεδία:

- **Files (JSON Array):** Ένας πίνακας με οντότητες τύπου JSON όπου η κάθε μία αντιπροσωπεύει ένα αρχείο που είναι αποθηκευμένο στα μέσα αποθήκευσης νέφους. Η οντότητα αυτή έχει τα δικά της πεδία:
 - **ID (String):** Μοναδικό αναγνωριστικό του αρχείου.
 - **Name (String):** Όνομα αρχείου.
 - **CreatedTime (String):** Ημερομηνία που ανέβηκε το αρχείο στο αντίστοιχο συνδεδεμένο μέσο αποθήκευσης νέφους.
 - **Type (String):** Έχει την τιμή "file" όταν αφορά τα αρχεία και χρησιμοποιείται για να γίνεται ο διαχωρισμός μεταξύ αρχείου – φακέλου στην γραφική διεπαφή.

- **Size (Number):** Το μέγεθος του αρχείου σε bytes.
- **MimeType (String):** Το όνομα του τύπου του αρχείου (π.χ. “image/jpeg”, “text/plain”) όπως προσδιορίζεται από το μέσο αποθήκευσης νέφους.
- **Email (String):** Η διεύθυνση ηλεκτρονικού ταχυδρομείου του μέσου αποθήκευσης νέφους στο οποίο ανήκει
- **Drive (String):** Το είδος του μέσου αποθήκευσης νέφους. Οι πιθανές τιμές του θα είναι “google-drive”, “dropbox” ή “onedrive”.
- **Extension (String):** Η προέκταση του αρχείου που καθορίζει το είδος του (π.χ. “.txt”, “.pdf”)
- **VirtualParent (String):** Το αναγνωριστικό του εικονικού φακέλου στον οποίο έχει ανεβεί το αρχείο.
- **Folders (JSON Array):** Ένας πίνακας με οντότητες τύπου JSON όπου η κάθε μία αντιπροσωπεύει έναν εικονικό φάκελο που είναι αποθηκευμένος στο εικονικό σύστημα νέφους. Τα πεδία είναι όλα ίδια με τα πεδία των αρχείων, με τις εξής όμως διαφορές στα παρακάτω πεδία:
 - **Type (String):** Έχει την τιμή “folder” αντί της τιμής “file” εφόσον αναφέρεται σε εικονικό φάκελο.
 - **Drive (String):** Έχει την τιμή “aio-drive” (το όνομα δηλαδή του εικονικού συστήματος νέφους) αντί των μέσων αποθήκευσης νέφους “google-drive”, “dropbox” και “onedrive”.
 - **Extension (String):** Έχει την τιμή “folder” εφόσον ένας εικονικός φάκελος δεν έχει προέκταση.

sharedFolders (JSON Array): Το πεδίο αυτό, περιέχει πληροφορίες για τους εικονικούς φακέλους που έχει μοιραστεί ο χρήστης. Ο λόγος που κρατάμε αυτή την πληροφορία ξεχωριστά είναι για την μείωση του κόστους αναζήτησης κατά την αναζήτηση των διαμοιρασμένων εικονικών φακέλων. Περιέχει δύο υποπεδία:

- **virtualFolderId (String):** Το αναγνωριστικό του εικονικού φακέλου που είναι διαμοιρασμένος.
- **sharedwithEmail (String Array):** Ένας πίνακας που περιέχει τις διευθύνσεις ηλεκτρονικού ταχυδρομείου με τους οποίους έχει μοιραστεί ο χρήστης τον εικονικό φάκελο.

Στο παρακάτω σχήμα **4.1.5.1** φαίνεται η εγγραφή ενός χρήστη όπως ακριβώς εμφανίζεται στην βάση δεδομένων. Πιο συγκεκριμένα εμφανίζονται οι εγγραφές για

αρχεία που έχει ανεβάσει στο εικονικό σύστημα νέφους (πορτοκαλί περίγραμμα), έναν εικονικό φάκελο που έχει δημιουργήσει (κόκκινο περίγραμμα), η εγγραφή για έναν εικονικό φάκελο που έχει μοιραστεί (μωβ περίγραμμα) καθώς και η εγγραφή του μέσου αποθήκευσης νέφους Google Drive που έχει συνδέσει με την εφαρμογή (μπλε περίγραμμα)

```
_id: ObjectId("5f12fc21b4ce9d1d5854e85e")
email: "aaa@gmail.com"
password: "$2b$10$ZxNlZqaYzaoL3BISxogGwOLLmwpW3poEpaFZvr9YD1gYpK5pbR530"
virtualdrive: Object
  folders: Array
    0: Object
      id: ObjectId("5f12faf1b4ce9d1d5854e85d")
      name: "Shared with me"
      parents: Array
      createdAt: "2020-6-18"
      type: "folder"
      email: "aaa@gmail.com"
      drive: "aio-drive"
      extension: "folder"
      path: ""
      virtualParent: "root"
    1: Object
  files: Array
    0: Object
      kind: "drive#file"
      id: "1n9IyCLY5xEiC21IbTQEf5vAhcRzw3YP5"
      name: "\pokeball.png"
      mimeType: "image/jpeg"
      size: 62702
      extension: ".png"
      type: "file"
      email: "kon.yiotis@gmail.com"
      drive: "google-drive"
      createdAt: "2020-07-28"
      virtualParent: "5f1fd894d884af3b2c012017"
    1: Object
    2: Object
    3: Object
  sharedFolders: Array
    0: Object
      _id: ObjectId("5f1fd8ded884af3b2c012019")
      virtualFolderId: "5f1fd894d884af3b2c012017"
      sharedWithEmail: "bbb@gmail.com"
  google-drive: Array
    0: Object
      aiofolder_id: "1LYKtJj8XTw7ufIr1fesC-_gVx0tSoN-t"
      _id: ObjectId("5f1fd700d884af3b2c012013")
      email: "kon.yiotis@gmail.com"
      token: "{\"iv\":\"0954d640ed0b7255b1810d76eb4dcee1\",\"encryptedData\":\"60b9dceeb754..."
  dropbox: Array
  onedrive: Array
```

Virtual Folder

Info about a shared folder

Connected Google Drive Storage Info

File that belongs to a virtual folder and is uploaded to Google Drive

Σχήμα 4.1.5.1: Αναλυτική απεικόνιση της εγγραφής ενός χρήστη στην βάση δεδομένων MongoDB.

4.2 Ασφάλεια

Ένα από τα σημαντικότερα ζητήματα στις διαδικτυακές εφαρμογές είναι η ασφάλεια. Με την εφαρμογή μέτρων ασφαλείας, επιτυγχάνεται η αποτροπή της κακόβουλης χρήσης των λειτουργιών μιας εφαρμογής. Παρόλο που η ασφάλεια αποτελεί ένα πολύ σύνθετο κλάδο των τεχνολογιών διαδικτύου και χρειάζεται αρκετή εξειδίκευση, υπάρχουν μερικά μέτρα ασφαλείας που είναι εύκολο να εφαρμοστούν. Σε αυτή την ενότητα, θα περιγράψουμε μερικές από τις κοινές τεχνικές που εφαρμόσαμε για να λύσουμε μερικά από τα θέματα ασφαλείας που μπορεί να προκύψουν.

Κατακερματισμός κωδικών πρόσβασης

Όπως αναφέραμε και στο κεφάλαιο 4.1.2, οι κωδικοί πρόσβασης των χρηστών που εγγράφονται στην διαδικτυακή εφαρμογή μας, κατακερματίζονται με την βοήθεια βιβλιοθήκης. Αυτό σημαίνει ότι στην βάση δεδομένων δεν υπάρχουν οι κανονικοί κωδικοί πρόσβασης, παρά μόνο οι κατακερματισμένες μορφές τους. Αυτό αποτρέπει την διαρροή ευαίσθητων πληροφοριών σε περίπτωση που γίνει παραβίαση της βάσης δεδομένων.

Κρυπτογραφία των κλειδιών των μέσων αποθήκευσης νέφους

Όταν ένας χρήστης συνδέει κάποιο μέσο αποθήκευσης νέφους με την εφαρμογή μας, αποθηκεύεται ένα κλειδί στην βάση δεδομένων μέσω του οποίου γίνονται οι κλήσεις στο API του μέσου. Τα κλειδιά αυτά, κρυπτογραφούνται με την βοήθεια βιβλιοθήκης κρυπτογράφησης (περισσότερα στο κεφάλαιο 5.3) πριν αποθηκευτούν στην βάση δεδομένων, και κατά την πρόσβαση τους από το API, αποκρυπτογραφούνται για να επανέλθουν στην κανονική τους μορφή έτσι ώστε να μπορέσουν να γίνουν τα αιτήματα. Σε περίπτωση που δεν εφαρμοζόταν η τεχνική της κρυπτογραφίας, αν κάποιος κακόβουλος χρήστης έπαιρνε πρόσβαση στην βάση δεδομένων, θα μπορούσε να χρησιμοποιήσει τα κλειδιά αυτά και να πάρει πρόσβαση σε όλα τα δεδομένα των χρηστών στα μέσα αποθήκευσης νέφους που είναι συνδεδεμένα με την εφαρμογή μας.

Μεταφορά “cookies” μόνο μέσω ασφαλών καναλιών

Τα “cookies” που βρίσκονται σωσμένα στον περιηγητή κατά την διάρκεια χρήσης της εφαρμογής μας, μεταφέρονται από τον φυλλομετρητή στον εξυπηρετητή, με σκοπό να γίνεται έλεγχος για την προέλευση των αιτημάτων. Επομένως, κάθε φορά που γίνεται ένα αίτημα στο API μας από έναν χρήστη, τα “cookies” περιέχουν πληροφορίες αυτού του χρήστη όπως την διεύθυνση ηλεκτρονικού ταχυδρομείου του. Αν κατά την διάρκεια μετάδοσης των αιτημάτων, μπορέσει κάποιος ενδιάμεσος χρήστης να «ακούσει» το αίτημα, θα έχει πρόσβαση στις πληροφορίες που περιλαμβάνει το “cookie”. Στην δικιά μας περίπτωση, εφόσον μεταφέρεται μόνο η διεύθυνση ηλεκτρονικού ταχυδρομείου δεν υπάρχει μεγάλος κίνδυνος. Υπάρχει όμως η περίπτωση σε μελλοντική εξέλιξη της εφαρμογής να μεταφέρονται επίσης επιπλέον ευαίσθητα στοιχεία. Επομένως, με την βοήθεια της βιβλιοθήκης “jsonwebtoken”, προσδιορίζουμε στα “cookies” να μεταφέρονται μόνο μέσω ασφαλή HTTPS πρωτοκόλλου αποτρέποντας την μετάδοση τους μέσω απλών, μη ασφαλών HTTP καναλιών.

Αποτροπή πρόσβασης σε ξένους πόρους

Μέσω κακόβουλων τεχνικών, υπάρχουν περιπτώσεις όπου χρήστες τροποποιούν κακόβουλα τα αιτήματα με σκοπό να αποκτήσουν πρόσβαση σε πόρους που δεν τους ανήκουν. Για την αποτροπή τέτοιων κακόβουλων ενεργειών, κάθε φορά που γίνεται ένα αίτημα στο API μας, περνάει πρώτα μέσα από μία ενδιάμεση συνάρτηση ελέγχου (middleware) (βλ. κεφάλαιο 3.5) με την οποία γίνεται αντιστοίχιση της ταυτότητας του χρήστη που έκανε το αίτημα, με την ταυτότητα του χρήστη του οποίου ζητάει πρόσβαση στους πόρους του. Σε περίπτωση δηλαδή που ένας χρήστης «Α» ζητήσει πρόσβαση σε πόρους του χρήστη «Β», το αίτημα θα απορριφθεί.

Κεφάλαιο 5.

Τεχνικές

Λεπτομέρειες

5.1 Επεκτασιμότητα του λογισμικού

Μία από τις απαιτήσεις που μπορεί να προκύψουν κατά την λειτουργία, ανάπτυξη και συντήρηση ενός λογισμικού, είναι η προσθήκη νέων λειτουργιών που θα ενισχύσουν τις δυνατότητες του. Πολλές φορές, η προσθήκη μιας νέας λειτουργίας μπορεί να είναι αρκετά όμοια με κάποιες από τις ήδη υλοποιημένες λειτουργίες. Για αυτόν τον λόγο θα πρέπει κατά την ανάπτυξη του λογισμικού να σκεφτόμαστε αν υπάρχει το ενδεχόμενο στο μέλλον να προστεθεί μία τέτοια λειτουργία και να δομούμε τον κώδικα του με κατάλληλο τρόπο έτσι ώστε να είναι εύκολα επεκτάσιμος.

Στην δικιά μας διαδικτυακή εφαρμογή, μία πιθανή επέκταση που μπορεί να γίνει είναι η προσθήκη νέων μέσων αποθήκευσης νέφους πέρα από τα τρία βασικά που υποστηρίζονται αυτή τη στιγμή. Η προσθήκη ενός νέου μέσου διευκολύνεται αρκετά εφόσον για κάθε μέσο αποθήκευσης νέφους έχουμε ένα αντίστοιχο αρχείο στο οποίο υλοποιούνται ακριβώς οι ίδιες μέθοδοι. Πιο συγκεκριμένα στον κατάλογο “api/src” του πηγαίου κώδικα υπάρχουν οι εξής κλάσεις-αρχεία:

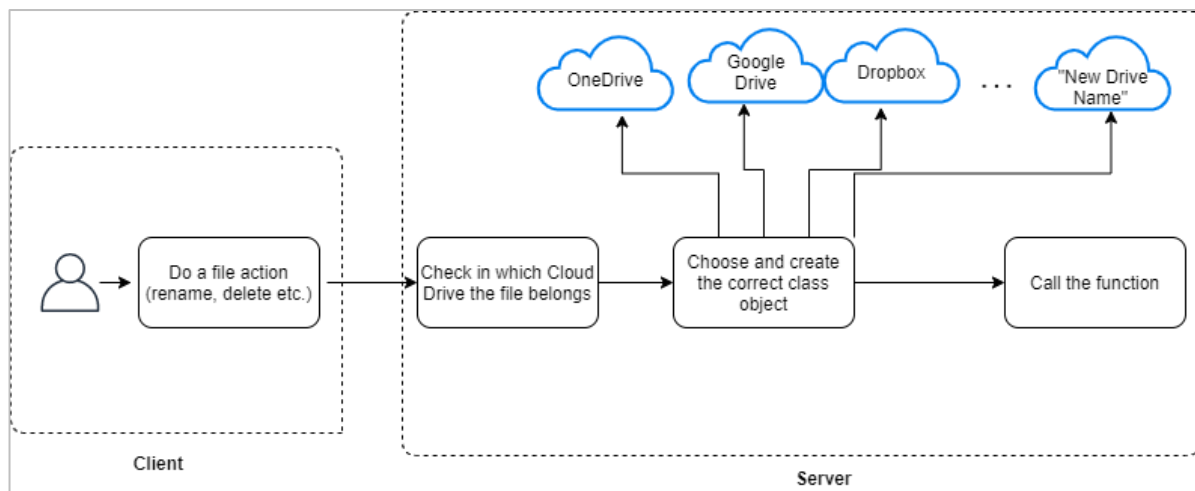
- **DropboxHandler.js:** Περιέχει όλες τις κλήσεις στο API του Dropbox
- **GoogleDriveHandler.js:** Περιέχει όλες τις κλήσεις στο API του Google Drive
- **OneDriveHandler.js:** Περιέχει όλες τις κλήσεις στο API του One Drive

Σε όλα αυτά τα αρχεία, υλοποιούνται οι εξής μέθοδοι:

- **initialize():** Περιέχει οτιδήποτε χρειάζεται αρχικοποίηση και πρέπει να καλείται πάντα πρώτη.
- **authorize():** Περιέχει την λήψη για την λήψη του συνδέσμου που πρέπει να σταλεί στον φυλλομετρητή για την ανακατεύθυνση του χρήστη στην σελίδα του μέσου αποθήκευσης νέφους.

- **generateToken()**: Περιέχει την κλήση για την λήψη του κλειδιού μέσω του οποίου θα μπορούμε να κάνουμε τις κλήσεις στο API του μέσου αποθήκευσης νέφους για την πρόσβαση στα προσωπικά αρχεία.
- **getDriveInfo()**: Περιέχει τη κλήση για την λήψη πληροφοριών όπως η διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη και τον υπολειπόμενο ελεύθερο χώρο στο μέσο αποθήκευσης νέφους.
- **getFiles()**: Περιέχει την κλήση για την λήψη όλων των πληροφοριών των αρχείων που είναι αποθηκευμένα στο μέσο αποθήκευσης νέφους.
- **renameFile()**: Περιέχει την κλήση για την μετονομασία ενός αρχείου, σωσμένου στο μέσο αποθήκευσης νέφους.
- **deleteFile()**: Περιέχει την κλήση για την διαγραφή ενός αρχείου, σωσμένου στο μέσο αποθήκευσης νέφους.
- **downloadFile()**: Περιέχει την κλήση για το κατέβασμα ενός αρχείου στον εξυπηρετητή. Χρησιμοποιείται μόνο κατά την διαδικασία της ανασυγκρότησης αρχείων (υπό-ενότητα 4.1.4.4).
- **uploadFile()**: Περιέχει την κλήση για το ανέβασμα ενός αρχείου στον εξυπηρετητή. Χρησιμοποιείται μόνο κατά την διαδικασία της ανασυγκρότησης αρχείων (υπό-ενότητα 4.1.4.4).
- **getSharedUrl()**: Περιέχει την κλήση για την λήψη του δημόσιου συνδέσμου κοινοποίησης ενός αρχείου, σωσμένου στο μέσο αποθήκευσης νέφους.
- **disableSharing()**: Περιέχει την κλήση για την διακοπή λειτουργίας του δημόσιου συνδέσμου που δημιουργήθηκε από την συνάρτηση “**getSharedUrl()**”.

Επομένως, αναλόγως το μέσο αποθήκευσης νέφους στο οποίο είναι σωσμένο ένα αρχείο και θέλουμε να εκτελέσουμε κάποια λειτουργία, δημιουργούμε το κατάλληλο αντικείμενο των κλάσεων που αναφέραμε παραπάνω και καλούμε τις παραπάνω συναρτήσεις. Άρα για την προσθήκη ενός καινούριου μέσου αποθήκευσης νέφους, το μόνο που χρειάζεται είναι η δημιουργία ενός καινούριου αρχείου “<drivename>**Handler.js**” στον κατάλογο “api/src” του πηγαίου κώδικα και η υλοποίηση των παραπάνω μεθόδων. Στο σχήμα **4.2.1** φαίνεται και η λογική αυτή που μόλις εξηγήσαμε.



Σχήμα 4.2.1: Σχηματική απεικόνιση της επεκτάσιμης λογικής που εφαρμόσαμε για την προσπέλαση αρχείων σε διαφορετικά μέσα αποθήκευσης νέφους.

5.2 Πληροφορίες εγκατάστασης

Εγκατάσταση - εκκίνηση της εφαρμογής

Για να μπορέσουμε να εκκινήσουμε τοπικά τη διαδικτυακή εφαρμογή μας, θα πρέπει αρχικά να εγκαταστήσουμε το Node.js και πιο συγκεκριμένα στην έκδοση 12.10.0 καθώς είναι το μόνο πρόγραμμα που απαιτείται για την εκτέλεση της.

Στην συνέχεια, εφόσον η εφαρμογή αποτελείται από δύο λειτουργικά μέρη, το μέρος της γραφικής διεπαφής και το μέρος του εξυπηρετητή, θα πρέπει να γίνει εκκίνηση και των δύο υπο-εφαρμογών ξεχωριστά. Πιο συγκεκριμένα, μέσω τερματικού πρέπει να κάνουμε τα εξής:

1. Πλοήγηση στον φάκελο “api” και εκτέλεση της εντολής “npm start” για την εκκίνηση του εξυπηρετητή.
2. Πλοήγηση στον φάκελο “client” και εκτέλεση της εντολής “npm start” για την εκκίνηση της γραφικής διεπαφής. Αυτομάτως, θα ανοίξει και ο περιηγητής όπου θα εμφανίζεται η γραφική διεπαφή με διεύθυνση <http://localhost:3000>.

Εκτελώντας τις παραπάνω εντολές τερματικού, ο χρήστης μπορεί πλέον να δει την γραφική διεπαφή της εφαρμογής μας μέσω του τοπικού συνδέσμου <http://localhost:3000>.

5.3 Επιπλέον βιβλιοθήκες

Σε αυτή την ενότητα θα κάνουμε μία σύντομη αναφορά των διαφόρων εξωτερικών βιβλιοθηκών που μας χρειάστηκαν για την υλοποίηση των λειτουργιών της εφαρμογής.

Διαχείριση των μέσων αποθήκευσης νέφους

Για την σύνδεση των μέσων αποθήκευσης νέφους “Google Drive” και “Dropbox” και για την διαχείριση των αρχείων τους, χρησιμοποιήσαμε στον εξυπηρετητή τις βιβλιοθήκες “googleapis” και “dropbox-v2-api” που προσφέρουν βοηθητικές μεθόδους για την χρήση των API τους. Για το “OneDrive” δεν υπήρχε αντίστοιχη βιβλιοθήκη, και η υλοποίηση της επικοινωνίας με το API του έγινε αποκλειστικά από εμάς.

Κατακερματισμός

Για τον κατακερματισμό των κωδικών πρόσβασης, χρησιμοποιήσαμε την βιβλιοθήκη **“bcrypt”**. Η βιβλιοθήκη αυτή, κατακερματίζει τους κωδικούς πρόσβασης με σκοπό οι πραγματικοί κωδικοί πρόσβασης, να μην είναι εμφανείς στην βάση δεδομένων, παρά μόνο οι κατακερματισμένες εκδόσεις τους. Αυτό μας αποτρέπει να έχουμε γνώση των κωδικών πρόσβασης των χρηστών που είναι εγγεγραμμένοι στο σύστημα, για λόγους ασφαλείας. Ο αλγόριθμος κατακερματισμού που χρησιμοποιεί στο παρασκήνιο αυτή η βιβλιοθήκη βασίζεται σε παραλλαγή του αλγόριθμου **“Blowfish block cipher”**. Ο λόγος που χρησιμοποιούμε αυτή την μέθοδο είναι για την ευελξία που προσφέρει αυτός ο αλγόριθμος διότι με την πάροδο των χρόνων και με την εξέλιξη της ταχύτητας των υπολογιστών, μπορούμε να προσαρμόσουμε τον αλγόριθμο θέτοντας πόσο ακριβή είναι η συνάρτηση κατακερματισμού (adaptive hash algorithm)[\[10\]](#)

Κρυπτογράφηση

Τα κλειδιά των μέσων αποθήκευσης νέφους, κρυπτογραφούνται πριν την αποθήκευση τους στην βάση δεδομένων με τον αλγόριθμο κρυπτογράφησης **“AES-256 ”** με την βοήθεια της βιβλιοθήκης **“crypto”**. Χρησιμοποιούμε κρυπτογράφηση και όχι κατακερματισμό για να μπορούμε να αποκρυπτογραφούμε το κλειδί και να παίρνουμε την αρχική τιμή του. Το κλειδί κρυπτογράφησης, αποθηκεύεται σε μεταβλητή περιβάλλοντος στον εξυπηρετητή.

Διαχείριση βάσης δεδομένων MongoDB

Για την διαχείριση της βάσης δεδομένων MongoDB Atlas, χρησιμοποιήσαμε την βιβλιοθήκη **“mongoose”**. Η βιβλιοθήκη αυτή, μας δίνει πρόσβαση σε όλες τις λειτουργίες της βάσης δεδομένων, σχετικά με την διαχείριση των δεδομένων που είναι αποθηκευμένα σε αυτή.

Διαχείριση των “cookies”

Για την δημιουργία και την διαγραφή του κρυπτογραφημένου **“cookie”**, χρησιμοποιείται η βιβλιοθήκη **“jsonwebtoken”** η οποία μέσω των «απαντήσεων» (responses) στα

αιτήματα (requests) διαχειρίζεται τα cookies που βρίσκονται στον περιηγητή του πελάτη.

Εμπειρία χρήστη

Για την εμφάνιση διαδραστικών μηνυμάτων στην γραφική διεπαφή του χρήστη, χρησιμοποιήσαμε μία βιβλιοθήκη ανοιχτού λογισμικού **“react-toastify”**. Η συγκεκριμένη βιβλιοθήκη, καθιστά εύκολη την προσθήκη αναδυόμενων ειδοποιήσεων στην εφαρμογή μας που σχετίζονται με σφάλματα ή επιτυχίες των ενεργειών που κάνουμε.

5.4 Οδηγός του API της εφαρμογής

Σε αυτή την ενότητα, θα γίνει ανάλυση του API που σχεδιάσαμε με σκοπό να γίνεται η επικοινωνία του πελάτη με τον εξυπηρετητή. Θα εξηγήσουμε σε ποιες περιπτώσεις και γιατί χρησιμοποιείται η κάθε κλήση στα ανάλογα τελικά σημεία (endpoints) του API

Σε αυτή την ενότητα, θα γίνει ανάλυση του API που σχεδιάσαμε με σκοπό να γίνεται η επικοινωνία του πελάτη με τον εξυπηρετητή. Θα εξηγήσουμε σε ποιες περιπτώσεις και γιατί χρησιμοποιείται η κάθε κλήση στα ανάλογα τελικά σημεία (endpoints) του API, καθώς τα δεδομένα που στέλνονται μέσω του κορμού του αιτήματος και τα δεδομένα που λαμβάνουμε μέσω της απάντησης από τον εξυπηρετητή.

Ανάλυση των τελικών σημείων

Όνομα μονοπατιού (path name)	/register
Τύπος αιτήματος	POST
Δεδομένα που περιέχονται	Διεύθυνση ηλεκτρονικού ταχυδρομείου, κωδικός πρόσβασης του χρήστη
Δεδομένα που επιστρέφονται	Μήνυμα για το αν η εγγραφή στην εφαρμογή ήταν επιτυχής.
Περιγραφή	Το αίτημα αυτό στέλνεται όταν ο χρήστης κάνει εγγραφή στην εφαρμογή μας. Στέλνονται τα στοιχεία του στον εξυπηρετητή και ο εξυπηρετητής

	αποφασίζει αν η εγγραφή του χρήστη είναι εφικτή. Σε περίπτωση που η διεύθυνση ηλεκτρονικού ταχυδρομείου υπάρχει ήδη καταχωρημένη στην βάση δεδομένων, η εγγραφή του χρήστη δεν μπορεί να γίνει.
--	---

Όνομα μονοπατιού (path name)	/login
Τύπος αιτήματος	POST
Δεδομένα που περιέχονται	Διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη.
Δεδομένα που επιστρέφονται	Cookie με τα κρυπτογραφημένα στοιχεία του χρήστη.
Περιγραφή	Με το που κάνει σύνδεση ο χρήστης στην εφαρμογή, στέλνεται αυτό το αίτημα με σκοπό να γίνει ταυτοποίηση του χρήστη και να επιστραφεί το cookie που θα του επιτρέψει να μένει συνδεδεμένος στην εφαρμογή

Όνομα μονοπατιού (path name)	/logout
Τύπος αιτήματος	POST
Δεδομένα που περιέχονται	Το cookie με τα κρυπτογραφημένα στοιχεία του χρήστη.
Δεδομένα που επιστρέφονται	Εντολή προς τον φυλλομετρητή για διαγραφή του cookie.
Περιγραφή	Όταν ένας χρήστης αποσυνδεθεί από την εφαρμογή, στέλνεται αυτό το αίτημα προς τον εξυπηρετητή με σκοπό να επιστραφεί η εντολή για την διαγραφή

	του cookie και να γίνει η ανακατεύθυνση στην αρχική σελίδα σύνδεσης.
--	--

Όνομα μονοπατιού (path name)	/checkToken
Τύπος αιτήματος	GET
Δεδομένα που περιέχονται	Το cookie με τα κρυπτογραφημένα στοιχεία του χρήστη.
Δεδομένα που επιστρέφονται	Μήνυμα για το αν τα στοιχεία του cookie αντιστοιχούν σε κάποιον χρήστη.
Περιγραφή	Αν ο χρήστης έχει συνδεθεί προηγουμένως στην εφαρμογή και αποφασίσει να κλείσει την σελίδα του περιηγητή και να ξαναμπει μετά από κάποιο χρονικό διάστημα ή να κάνει ανανέωση της σελίδας, στέλνεται αυτόματα το παραπάνω αίτημα για να ελεγχτεί αν ο παρών χρήστης είναι συνδεδεμένος με βάση το cookie που στάλθηκε. Αν δεν υπήρχε αυτή η κλήση, ο χρήστης σε κάθε ανανέωση της σελίδας η στο κλείσιμο της θα πρέπει να ξανακάνει σύνδεση στην εφαρμογή.

Όνομα μονοπατιού (path name)	/connectDrive/<drive name>
Τύπος αιτήματος	GET
Δεδομένα που περιέχονται	Μεταβλητή <drive name> που ισούται με "google-drive", "dropbox" ή "onedrive".
Δεδομένα που επιστρέφονται	Ο σύνδεσμος της σελίδας του μέσου αποθήκευσης νέφους για εισαγωγή στοιχείων και σύνδεση με την εφαρμογή.

Περιγραφή	Όταν ο χρήστης επιλέξει από την γραφική διεπαφή να συνδέσει ένα μέσο αποθήκευσης νέφους, στέλνεται αίτημα στον εξυπηρετητή με το όνομα του νέφους και δημιουργείται και στέλνεται στον φυλλομετρητή ο ανάλογος σύνδεσμος που ανακατευθύνεται ο χρήστης για να εισάγει τα στοιχεία του.
------------------	--

Όνομα μονοπατιού (path name)	/authenticateDrive
Τύπος αιτήματος	POST
Δεδομένα που περιέχονται	Ο κωδικός που επιστράφηκε στον χρήστη μετά την εισαγωγή των στοιχείων στην σελίδα του μέσου αποθήκευσης νέφους και το όνομα του μέσου αποθήκευσης νέφους.
Δεδομένα που επιστρέφονται	Μήνυμα επιτυχίας ή μήνυμα αποτυχίας.
Περιγραφή	Όταν ο χρήστης ανακατευθυνθεί στον σύνδεσμο που επιστράφηκε από το αίτημα στο τελικό σημείο "/connectDrive" και εισάγει τα στοιχεία του, ανακατευθύνεται πάλι στην εφαρμογή μαζί με έναν κωδικό. Ο κωδικός αυτός στέλνεται μέσω αιτήματος στο μονοπάτι "/authenticateDrive" στον εξυπηρετητή και χρησιμοποιείται για την λήψη του κλειδιού του μέσου αποθήκευσης νέφους που χρησιμοποιείται για να γίνονται οι κλήσεις στο API του για την λήψη πληροφοριών των αρχείων.

Όνομα μονοπατιού (path name)	/getAccessToken
Τύπος αιτήματος	GET

Δεδομένα που περιέχονται	Η διεύθυνση ηλεκτρονικού ταχυδρομείου του μέσου αποθήκευσης νέφους, το όνομα του μέσου αποθήκευσης νέφους.
Δεδομένα που επιστρέφονται	Το κλειδί του μέσου αποθήκευσης νέφους που αντιστοιχεί στην διεύθυνση ηλεκτρονικού ταχυδρομείου που στάλθηκε.
Περιγραφή	Το παρόν αίτημα, χρησιμοποιείται όταν ο χρήστης θέλει να κατεβάσει ή να ανεβάσει ένα αρχείο στο μέσο αποθήκευσης νέφους. Όπως αναφέραμε και στην υποενότητα 4.1.4.1 τα αιτήματα για ανέβασμα και κατέβασμα ενός αρχείου, γίνονται απευθείας από τον φυλλομετρητή, επομένως χρειάζονται το κλειδί του μέσου αποθήκευσης νέφους για αυτόν τον σκοπό.

Όνομα μονοπατιών (path names)	/renameFile, /deleteFile, /shareFile
Τύπος αιτήματος	GET
Δεδομένα που περιέχονται	Αναγνωριστικό του αρχείου, διεύθυνση ηλεκτρονικού ταχυδρομείου του μέσου αποθήκευσης νέφους που ανήκει το αρχείο, όνομα του μέσου αποθήκευσης νέφους και στην περίπτωση αιτήματος "/renameFile" περιέχεται επίσης το νέο όνομα του αρχείου.
Δεδομένα που επιστρέφονται	Μήνυμα για το αν το αίτημα πέτυχε ή απέτυχε.
Περιγραφή	Τα παραπάνω αιτήματα εφόσον αφορά την τροποποίηση ενός αρχείου, ακολουθούν παρόμοια λογική. Καλούνται όταν ο χρήστης επιλέγει να κάνει μία από τις ακόλουθες αλλαγές σε ένα αρχείο:

	<ul style="list-style-type: none"> • Μετονομασία • Διαγραφή • Δημιουργία δημόσιου συνδέσμου πρόσβασης
--	--

Όνομα μονοπατιού (path name)	/getFiles
Τύπος αιτήματος	POST
Δεδομένα που περιέχονται	Αναγνωριστικό εικονικού φακέλου, διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη που έχει δημιουργήσει τον εικονικό φάκελό, ακριβές μονοπάτι του φακέλου.
Δεδομένα που επιστρέφονται	Οι εγγραφές των αρχείων που περιέχονται στον εικονικό φάκελο.
Περιγραφή	Το παραπάνω αίτημα, γίνεται όταν ο χρήστης επιλέγει να ανοίξει έναν εικονικό φάκελο. Όταν ο εξυπηρετητής λάβει το αίτημα, ψάχνει στην βάση δεδομένων, βρίσκει ποια αρχεία ανήκουν στον εικονικό φάκελο και επιστρέφει τις πληροφορίες τους στον φυλλομετρητή.

Όνομα μονοπατιού (path name)	/createFolder
Τύπος αιτήματος	POST
Δεδομένα που περιέχονται	Αναγνωριστικό του εικονικού φακέλου-πατέρα στον οποίο ενδεχομένως να περιέχεται ο νέος φάκελος που πρόκειται να δημιουργηθεί.
Δεδομένα που επιστρέφονται	Πληροφορίες του εικονικού φακέλου που δημιουργήθηκε.
Περιγραφή	Όταν ο χρήστης επιλέγει να δημιουργήσει έναν εικονικό φάκελο, στέλνεται το παρόν

	αίτημα. Όταν ο εξυπηρετητής λάβει το αίτημα, προσθέτει μία εγγραφή με τις πληροφορίες, στον πίνακα των εικονικών φακέλων του χρήστη, στην βάση δεδομένων. Στην συνέχεια επιστρέφει τις πληροφορίες αυτές με αποτέλεσμα να εμφανιστεί στην γραφική διεπαφή ο εικονικός αυτός φάκελος.
--	--

Όνομα μονοπατιού (path name)	/uploadVirtual
Τύπος αιτήματος	POST
Δεδομένα που περιέχονται	JSON αντικείμενο με τις πληροφορίες του αρχείου που ανέβασε ο χρήστης σε εικονικό φάκελο.
Δεδομένα που επιστρέφονται	JSON αντικείμενο με τις πληροφορίες του αρχείου που ανέβασε ο χρήστης σε εικονικό φάκελο.
Περιγραφή	Όταν ο χρήστης ανεβάσει ένα αρχείο σε εικονικό φάκελο μέσω της διεπαφής, στέλνεται το εξής αίτημα διότι οι πληροφορίες του αρχείου αυτού πρέπει να σωθούν στην βάση για να μπορεί αργότερα να προσπελάσει τα περιεχόμενα του εικονικού φακέλου.

Όνομα μονοπατιού (path name)	/getVirtualQuota
Τύπος αιτήματος	GET
Δεδομένα που περιέχονται	Το cookie με τα κρυπτογραφημένα στοιχεία του συνδεδεμένου χρήστη.
Δεδομένα που επιστρέφονται	Ο συνολικός ελεύθερος χώρος του εικονικού συστήματος νέφους.

Περιγραφή	Κάθε φορά που ο χρήστης συνδέεται στην εφαρμογή, στέλνεται το παραπάνω αίτημα στον εξυπηρετητή. Ο εξυπηρετητής μέσω του cookie, βλέπει από ποιον συνδεδεμένο χρήστη προήλθε το αίτημα και στην συνέχεια αναζητά στην βάση δεδομένων τον χρήστη αυτόν, χρησιμοποιεί τα κλειδιά του και στέλνει αιτήματα σε όλα τα συνδεδεμένα μέσα αποθήκευσης νέφους για να βρεί τον ελεύθερο τους χώρο. Αφού λάβει πληροφορίες για τους ελεύθερους χώρους, τους προσθέτει και τους στέλνει μέσω απάντησης στον φυλλομετρητή για να μπορεί ο χρήστης να βλέπει πόσο ελεύθερο χώρο έχει αθροιστικά, στο εικονικό σύστημα νέφους.
------------------	---

Όνομα μονοπατιού (path name)	/getIdealDrive
Τύπος αιτήματος	GET
Δεδομένα που περιέχονται	Το cookie με τα κρυπτογραφημένα στοιχεία του συνδεδεμένου χρήστη.
Δεδομένα που επιστρέφονται	Το κλειδί του μέσου αποθήκευσης νέφους με τον πιο πολύ ελεύθερο χώρο
Περιγραφή	Κάθε φορά που ο χρήστης επιλέγει ένα αρχείο για ανέβασμα, στέλνεται το παραπάνω αίτημα στον εξυπηρετητή. Ο εξυπηρετητής μέσω του cookie, βλέπει από ποιον συνδεδεμένο χρήστη προήλθε το αίτημα και στην συνέχεια αναζητά στην βάση δεδομένων τον χρήστη αυτόν, χρησιμοποιεί τα κλειδιά του και στέλνει αιτήματα σε όλα τα συνδεδεμένα μέσα

	<p>αποθήκευσης νέφους για να βρεί τον ελεύθερο τους χώρο. Αφού λάβει πληροφορίες για τους ελεύθερους χώρους, βρίσκει το μέσο με τον περισσότερο ελεύθερο χώρο και στέλνει μέσω απάντησης το κλειδί του στον φυλλομετρητή έτσι ώστε να ανέβει το αρχείο σε αυτό το μέσο αποθήκευσης νέφους.</p>
--	--

Κεφάλαιο 6.

Γραφική διεπαφή

της εφαρμογής

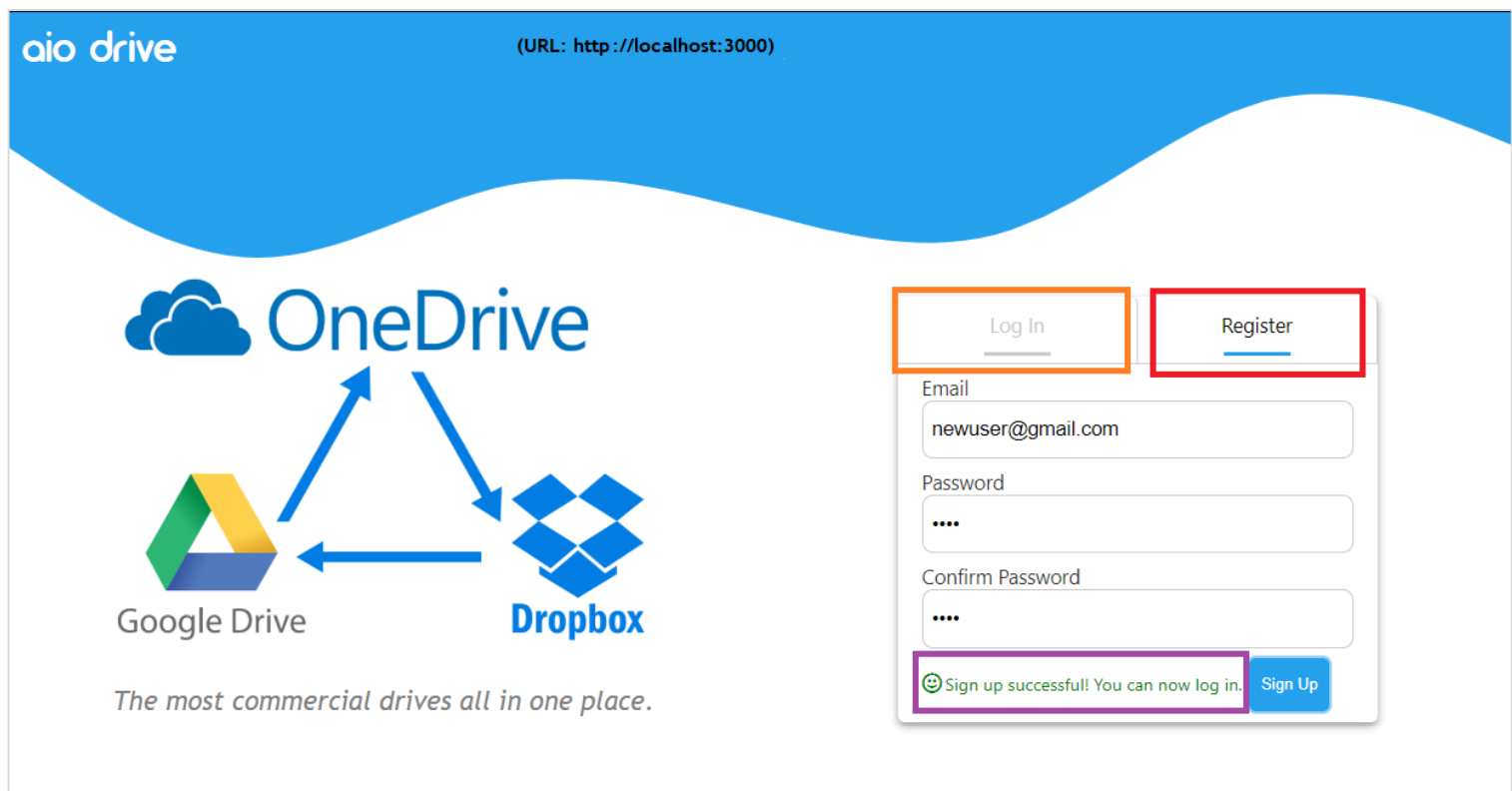
6.1 Παρουσίαση του εικονικού συστήματος αποθήκευσης νέφους

Σε αυτό το κεφάλαιο, θα γίνει μία βήμα προς βήμα παρουσίαση της γραφική διεπαφής της εφαρμογής μας, με σκοπό να μπορέσει ένας χρήστης να εξοικειωθεί με τις λειτουργίες που προσφέρει. Θα αναλύσουμε αρχικά τις δύο ξεχωριστές σελίδες της εφαρμογής και στην συνέχεια θα δείξουμε τις λειτουργίες που σχετίζονται με την σύνδεση μέσω των αποθήκευσης νέφους και τις λειτουργίες των αρχείων.

6.1.1 Σελίδες της εφαρμογής

Σελίδα εγγραφής - σύνδεσης χρήστη (Σχήμα 5.1.1.1)

Ο χρήστης, αφού εκκινήσει την εφαρμογή τοπικά και επισκεπτεί τον σύνδεσμο <http://localhost:3000>, θα πρέπει αρχικά να δημιουργήσει έναν λογαριασμό. Αφότου κάνει κλικ στην επιφάνεια που σημειώνεται με κόκκινο περίγραμμα μπορεί πλέον να εισάγει τα στοιχεία του. Αν εισάγει έγκυρη – ως προς την μορφή – διεύθυνση και έναν κωδικό πρόσβασης, θα εμφανιστεί ένα μήνυμα επιτυχίας (μωβ περίγραμμα). Στην συνέχεια, κάνοντας κλικ στην πορτοκαλί επιφάνεια, και πατώντας το κουμπί “Log In” ανακατευθύνεται στην βασική σελίδα της εφαρμογής.



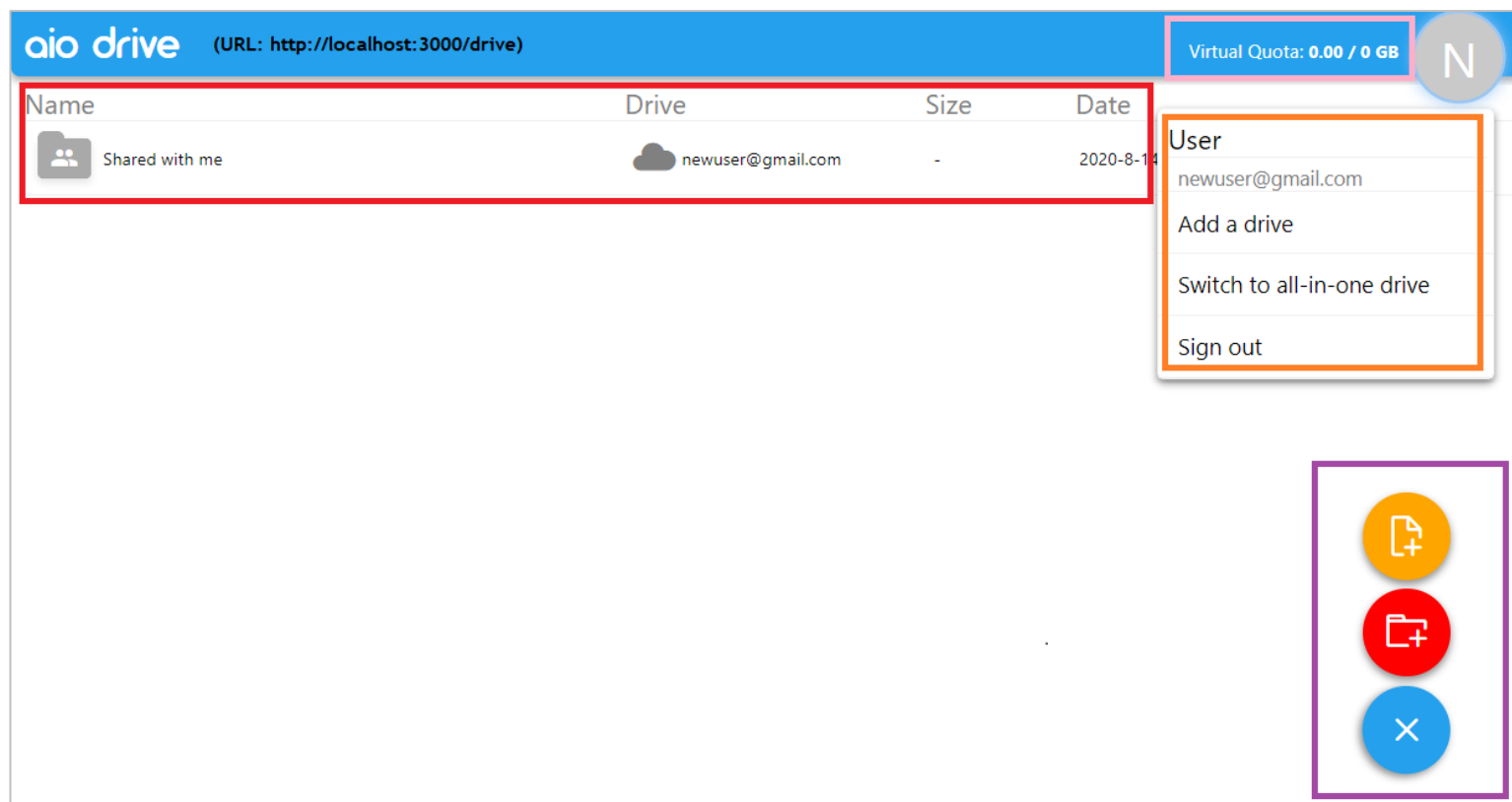
Σχήμα 5.1.1.1: Σελίδα εγγραφής-σύνδεσης της εφαρμογής.

Κύρια σελίδα εικονικού συστήματος νέφους (Σχήμα 5.1.1.2)

Κατά την επιτυχή σύνδεση στην εφαρμογή μας, ο χρήστης ανακατευθύνεται στην κύρια σελίδα του εικονικού συστήματος νέφους με σύνδεσμο <http://localhost:3000/drive>. Σε αυτήν την σελίδα αρχικά εμφανίζεται ο συνολικός ελεύθερος χώρος των συνδεδεμένων μέσων αποθήκευσης νέφους (ροζ περίγραμμα). Πατώντας κλικ στην στρόγγυλη επιφάνεια (που αντιστοιχεί στο πρώτο γράμμα της διεύθυνσης ηλεκτρονικού ταχυδρομείου του χρήστη) ανοίγει ένα αναδυόμενο παράθυρο – μενού (πορτοκαλί περίγραμμα). Μέσα από αυτό το μενού, ο χρήστης μπορεί να συνδέσει ένα ή περισσότερα μέσα αποθήκευσης νέφους πατώντας στην επιλογή “Add a drive”, να αποσυνδεθεί πατώντας στην επιλογή “Sign out” καθώς και να μεταβεί στην “All-in-one” διεπαφή (η οποία θα εξηγηθεί στην επόμενη ενότητα 5.2). πατώντας την ένδειξη “Switch to all-in-one drive”.

Στο μωβ περίγραμμα, εμφανίζονται 3 κουμπιά. Πατώντας το πορτοκαλί κουμπί ή σέρνοντας ένα αρχείο στην σελίδα, ο χρήστης μπορεί να ανεβάσει ένα αρχείο καθώς και να φτιάξει έναν εικονικό φάκελο πατώντας το κόκκινο κουμπί. Τέλος με το μπλε κουμπί κλείνει το μενού με τα κουμπιά.

Στο κόκκινο περίγραμμα εμφανίζεται ο φάκελος “Shared with me”. Μέσα σε αυτόν τον φάκελο θα εμφανίζονται άλλοι εικονικοί φάκελοι που άλλοι χρήστες θα διαμοιραστούν με τον τρέχον χρήστη μελλοντικά.



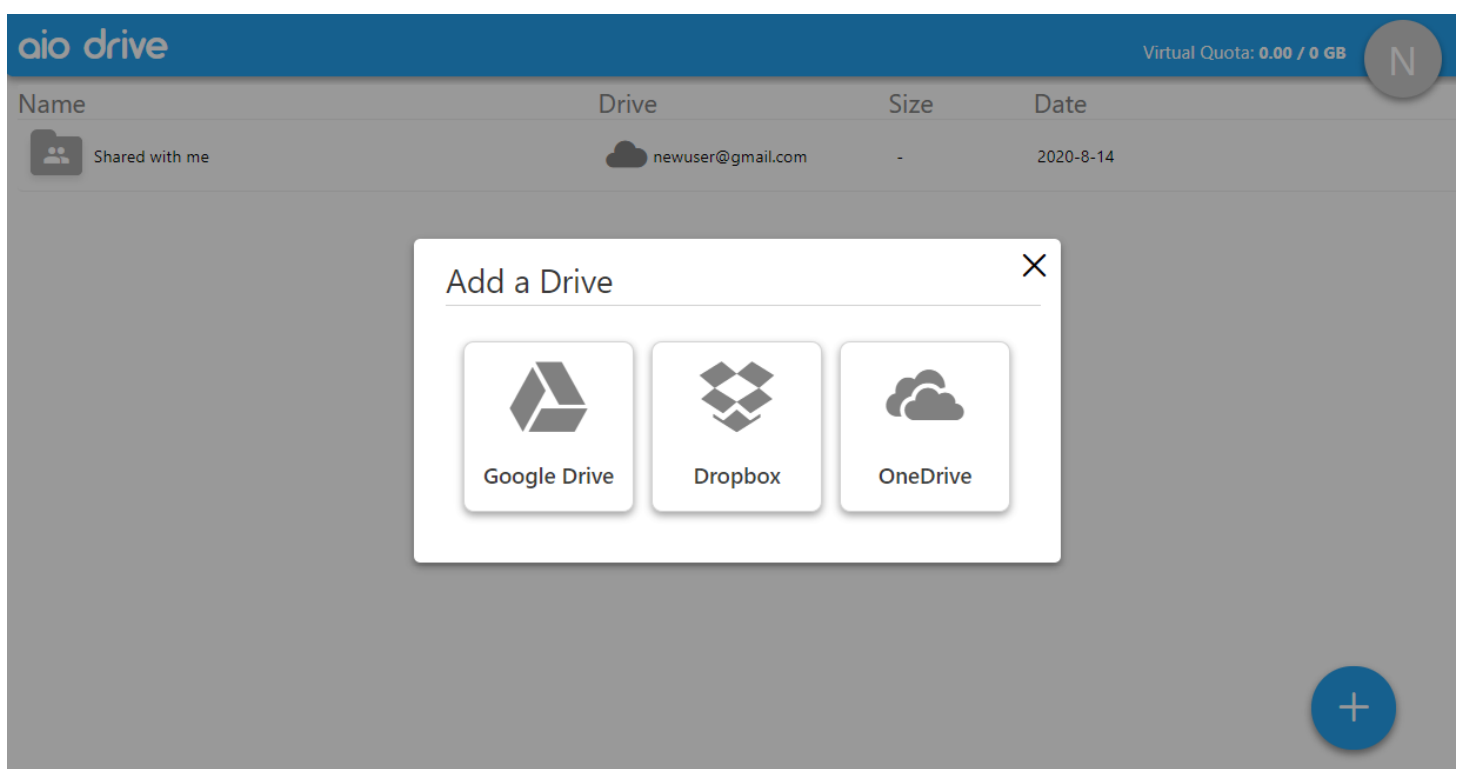
Σχήμα 5.1.1.2: Βασική σελίδα του εικονικού συστήματος νέφους.

6.1.2 Εκτέλεση λειτουργιών

Σύνδεση ενός μέσου αποθήκευσης νέφους

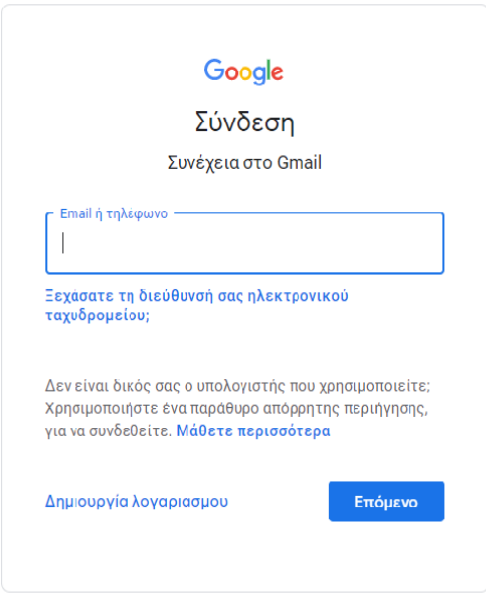
Όταν ο χρήστης κάνει κλικ στο μενού την επιλογή “Add a drive”, εμφανίζεται ένα αναδυόμενο παράθυρο με τρία κουμπιά όπου κάθε ένα από αυτά αντιστοιχεί σε ένα από τα μέσα αποθήκευσης νέφους που υποστηρίζει η εφαρμογή μας (Σχήμα 5.1.2.1). Επιλέγοντας ένα από αυτά τα μέσα, η εφαρμογή ανακατευθύνει τον χρήστη στην σελίδα του μέσου όπου θα πρέπει να εισάγει τα στοιχεία του μέσου αποθήκευσης νέφους για να γίνει η σύνδεση (Σχήμα 5.1.2.2). Αφότου εισάγει τα στοιχεία του επιτυχώς, το μέσο αποθήκευσης νέφους ανακατευθύνει τον χρήστη πίσω στην εφαρμογή μας και η σύνδεση του μέσου είναι πλέον επιτυχής. Αυτό μπορεί να φανεί από την ένδειξη “Virtual

Quota” που θα εμφανίζει πλέον τον ελεύθερο χώρο του μέσου που μόλις προσθέσαμε. Στην προκειμένη περίπτωση προσθέσαμε ένα μέσο αποθήκευσης νέφους “Google Drive” που μας προσφέρει 15GB ελεύθερου χώρου για το εικονικό μας σύστημα αποθήκευσης (Σχήμα 5.1.2.3).

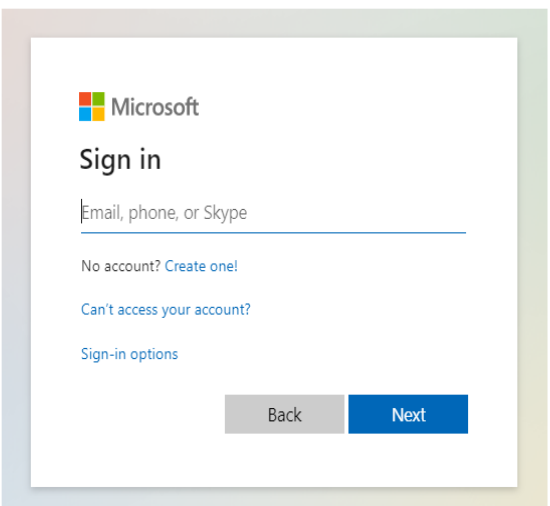


Σχήμα 5.1.2.1: Το μενού επιλογής μέσου αποθήκευσης νέφους προς σύνδεση με την εφαρμογή μας

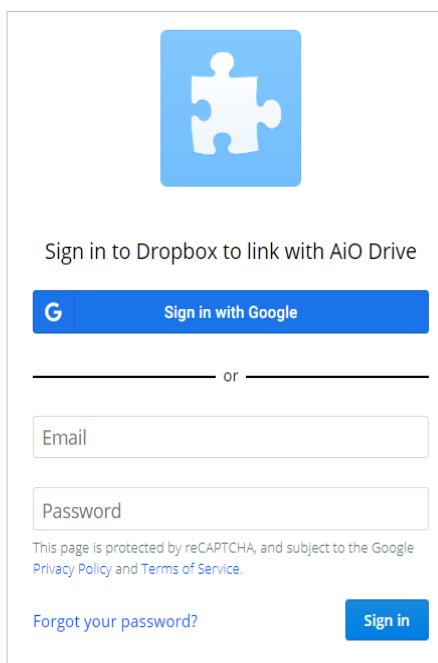
(URL: <https://accounts.google.com/ServiceLogin/identifier>)



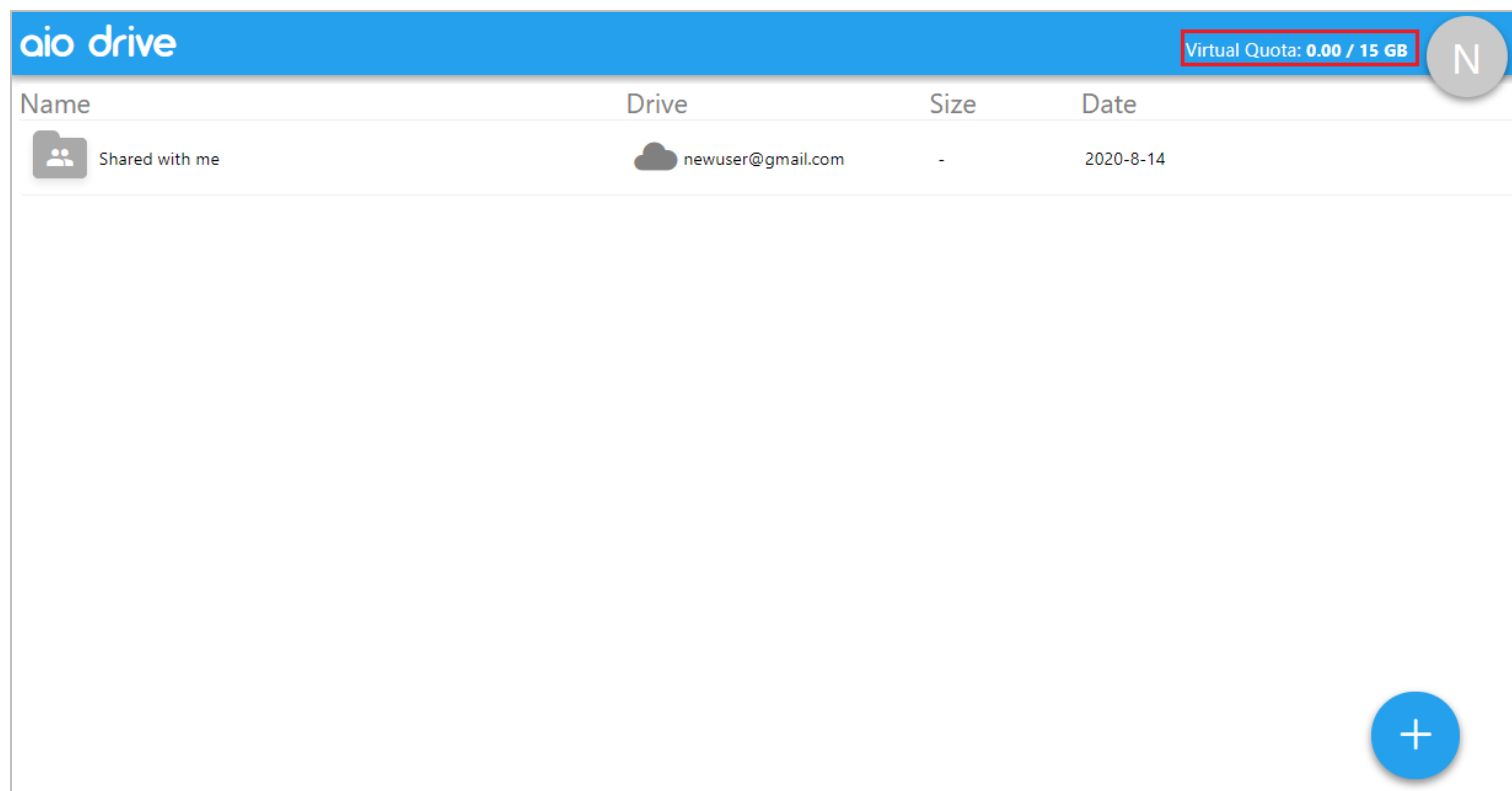
(URL: <https://login.microsoftonline.com/common/oauth2/v2.0/authorize>)





(URL: <https://www.dropbox.com/1/oauth2/authorize>)



Σχήμα 5.1.2.2: Οι σελίδες ανακατεύθυνσης για σύνδεση των μέσων αποθήκευσης νέφους με την σειρά “Google Drive”, “OneDrive”, “Dropbox”.



Name	Drive	Size	Date
 Shared with me	 newuser@gmail.com	-	2020-8-14

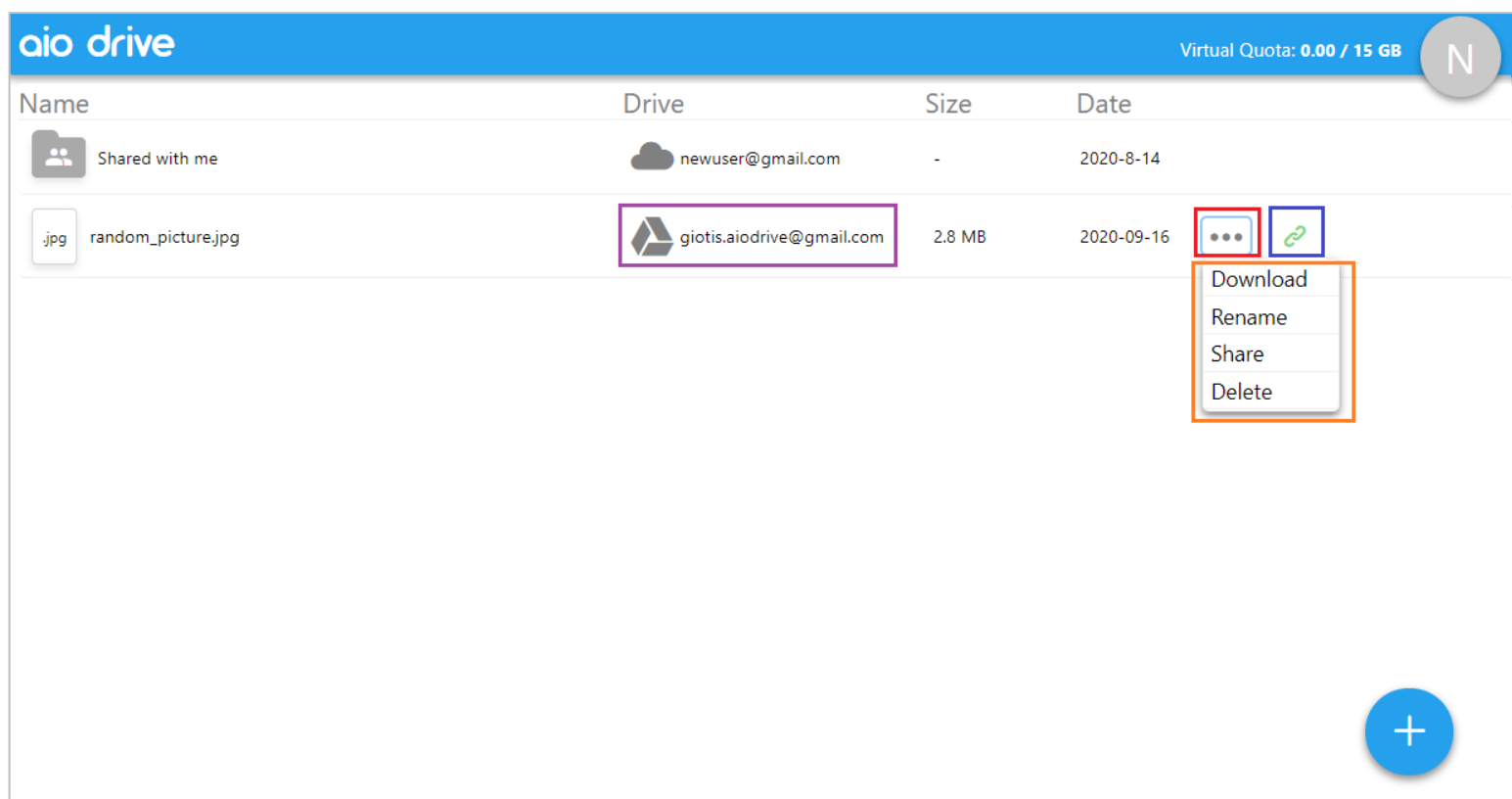
Σχήμα 5.1.2.3: Η κύρια σελίδα της εφαρμογής μας που ανακατευθύνεται ο χρήστης μετά την προσθήκη ενός μέσου αποθήκευσης νέφους με 15GB ελεύθερου χώρου (εμφανές με κόκκινο περίγραμμα).

Διαχείριση αρχείων (Σχήμα 5.1.2.4)

Μετά την σύνδεση μέσω αποθήκευσης νέφους, ο χρήστης μπορεί πλέον να ανεβάσει αρχεία. Αφού ανεβάσει επιτυχώς ένα αρχείο και πατήσει την επιλογή μενού (κόκκινο περίγραμμα), ανοίγει το μενού (πορτοκαλί περίγραμμα) για να εκτελέσει τις λειτουργίες διαγραφής, μετονομασίας, διαμοιρασμού μέσω δημοσίου συνδέσμου, και κατεβάσματος του αρχείου.

Στην περίπτωση διαμοιρασμού, ο χρήστης μπορεί να κάνει κλικ στην επιφάνεια του εικονιδίου συνδέσμου (μπλε περίγραμμα) για να γίνει η αντιγραφή του δημόσιου συνδέσμου. Επίσης, αφού ανέβει το αρχείο, μπορεί ο χρήστης να δει σε ποιο από τα μέσα αποθήκευσης νέφους ανέβηκε το αρχείο (μωβ περίγραμμα).

Ο χρήστης μπορεί επίσης να ανοίξει το αρχείο κάνοντας κλικ πάνω στην επιφάνεια του ονόματος αρχείου και αν ο τύπος αρχείου υποστηρίζεται από την περιηγητή, θα ανοίξει κανονικά στην οθόνη (π.χ. εικόνα, βίντεο, αρχείου ήχου, PDF κ.α.).

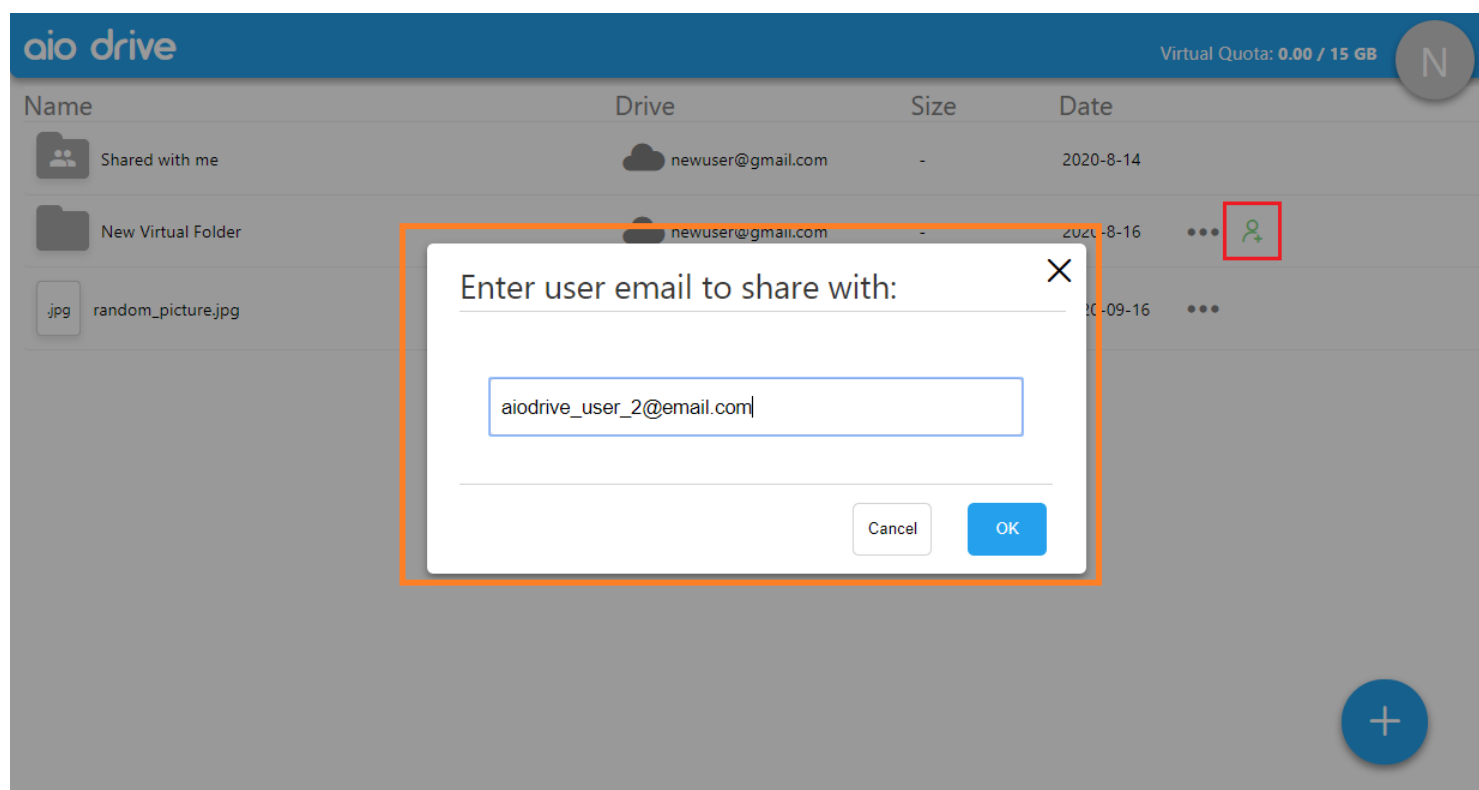


Σχήμα 5.1.2.4: Στιγμιότυπο ενός ανεβασμένου αρχείου, μαζί με το μενού των λειτουργιών διαχείρισης του αρχείου που υποστηρίζονται.

Διαχείριση εικονικών φακέλων (Σχήμα 5.1.2.5 – Σχήμα 5.1.2.6)

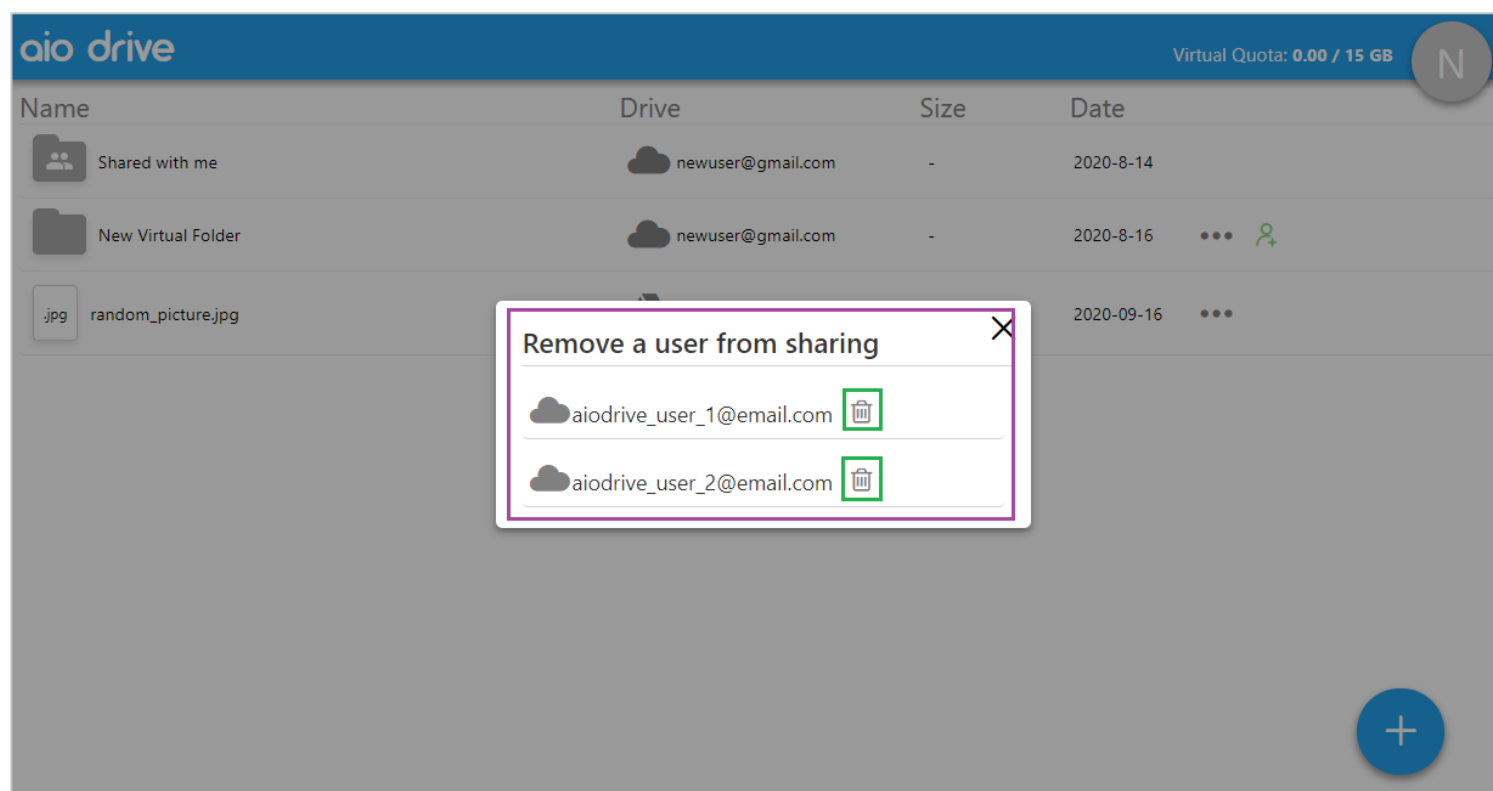
Μετά την δημιουργία ενός εικονικού φακέλου, ο χρήστης μπορεί επίσης να εκτελέσει κάποιες λειτουργίες διαχείρισης εικονικών φακέλων όπως μετονομασία, διαγραφή και διαμοιρασμό με άλλους – εγγεγραμμένους στην εφαρμογή – χρήστες με τον ίδιο τρόπο που γίνεται και στα αρχεία.

Όταν ο χρήστης κάνει κλικ στην επιλογή “Share” ανοίγει ένα αναδυόμενο παράθυρο στο οποίο ο χρήστης καλείται να εισάγει την διεύθυνση ηλεκτρονικού ταχυδρομείου του εγγεγραμμένου χρήστη με τον οποίο επιθυμεί να διαμοιραστεί τον φάκελο του (πορτοκαλί περίγραμμα). Με το που προστεθεί τουλάχιστον ένας χρήστης στον διαμοιρασμό, εμφανίζεται το εικονίδιο διαμοιρασμένου χρήστη (κόκκινο περίγραμμα) για να φαίνεται ότι ο εικονικός φάκελος έχει διαμοιραστεί. Πατώντας κλικ πάνω σε αυτό το εικονίδιο, ο χρήστης μπορεί να προσθέσει επιπλέον χρήστες στην διαμοιρασμό.



Σχήμα 5.1.2.5: Αναδυόμενο παράθυρο για προσθήκη χρήστη στον διαμοιρασμό του εικονικού φακέλου.

Τέλος, για την αφαίρεση χρηστών από τον διαμοιρασμό, ο χρήστης μπορεί να κάνει κλικ ξανά στο μενού του φακέλου όπου πλέον η επιλογή “Share” έχει πλέον μετατραπεί σε επιλογή “Unshare”. Πατώντας λοιπόν αυτή την επιλογή, εμφανίζεται αναδυόμενο παράθυρο (μωβ περίγραμμα) με την λίστα των χρηστών που συμμετέχουν στον διαμοιρασμό, με κουμπί αφαίρεσης δίπλα από τις διευθύνσεις ηλεκτρονικού ταχυδρομείου τους (πράσινο περίγραμμα).



Σχήμα 5.1.2.6: Αναδυόμενο παράθυρο για αφαίρεση χρήστη από τον διαμοιρασμό του εικονικού φακέλου.

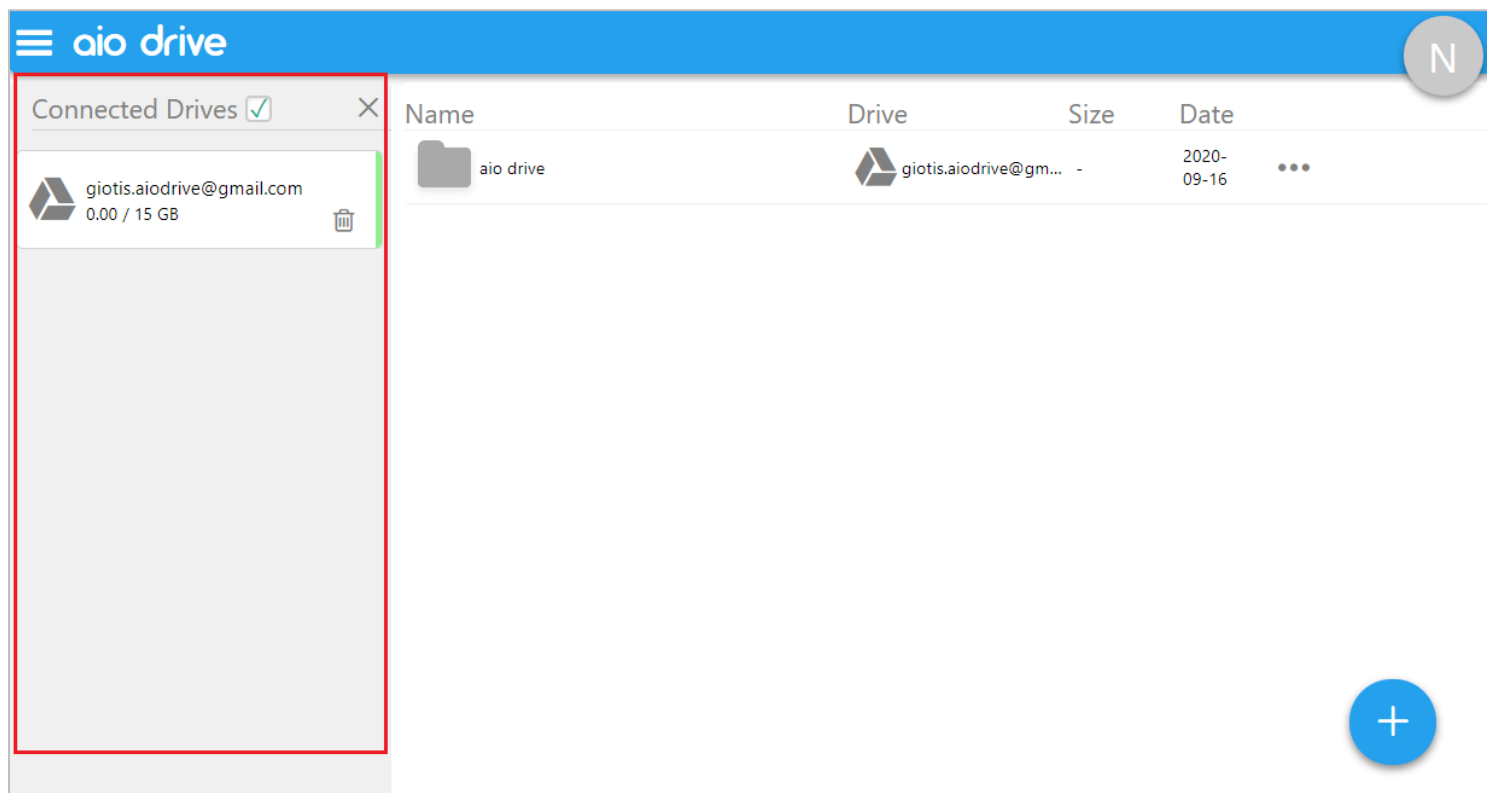
6.2 Επιπλέον λειτουργικότητα: Η “All-In-One”

Διεπαφή

Μία επιπλέον λειτουργία που προσθέσαμε εκτός των απαιτήσεων της διπλωματικής εργασίας, είναι η δυνατότητα ο χρήστης να μπορεί να διαχειρίζεται όλα τα συνδεδεμένα μέσα αποθήκευσης νέφους πάνω σε μία ενιαία γραφική διεπαφή. Αυτή η λειτουργία διαφέρει από το εικονικό σύστημα νέφους και το μόνο που προσφέρει, είναι η απευθείας πρόσβαση και διαχείριση των αρχείων των συνδεδεμένων μέσων αποθήκευσης νέφους. Ο χρήστης δηλαδή, βλέπει και διαχειρίζεται τα αρχεία του όπως ακριβώς εμφανίζονται αν έμπαινε στην σελίδα του μέσου αποθήκευσης νέφους.

Ανοίγοντας λοιπόν ο χρήστης το μενού χρήστη και κάνοντας κλικ στην επιλογή “Switch to all-in-one drive” ο χρήστης μεταφέρεται σε αυτή την “all-in-one” διεπαφή όπως φαίνεται στο σχήμα 5.2.1. Στο κόκκινο περίγραμμα εμφανίζονται τα μέσα αποθήκευσης νέφους που ο χρήστης έχει προσθέσει και κάνοντας κλικ πάνω σε αυτά, λειτουργούν ως φίλτρα εμφάνισης. Αυτό σημαίνει ότι αν ο χρήστης έχει προσθέσει παραπάνω από ένα μέσο αποθήκευσης νέφους και θέλει να διαχειριστεί το ένα από αυτά, κάνοντας κλικ στο άλλο, παύουν να εμφανίζονται τα αρχεία του μέσου αυτού στην γραφική διεπαφή. Επίσης, κάνοντας κλικ στο εικονίδιο δίπλα από την διεύθυνση ηλεκτρονικού ταχυδρομείου του συνδεδεμένου μέσου, ο χρήστης μπορεί να αφαιρέσει αυτό το μέσο από την εφαρμογή.

Όσον αφορά τις λειτουργίες, ισχύουν οι ίδιες λειτουργίες με το εικονικό σύστημα αποθήκευσης δηλαδή δημιουργία «φυσικών» φακέλων, ανέβασμα, κατέβασμα, μετονομασία και διαγραφή φακέλων-αρχείων. Κατά το ανέβασμα αρχείου και κατά την δημιουργία «φυσικού» φακέλου, εμφανίζεται αναδυόμενο παράθυρο για να επιλέξει ο χρήστης σε ποιο μέσο αποθήκευσης νέφους επιθυμεί να εκτελέσει την λειτουργία αυτή.



Σχήμα 5.2.1: Η "All-in-one" διεπαφή.

Κεφάλαιο 7.

Επίλογος

7.1 Σύνοψη

Ολοκληρώνοντας την διπλωματική εργασία, καταφέραμε να έχουμε μία πλήρως λειτουργική εφαρμογή που πετυχαίνει τους στόχους και πληροί τις απαιτήσεις που θέσαμε.

Συνοψίζοντας, καταφέραμε να υλοποιήσουμε μία εφαρμογή, μέσω της οποίας θα μπορούν οι χρήστες να δημιουργούν έναν λογαριασμό και στην συνέχεια να συνδέουν τα μέσα αποθήκευσης νέφους που αυτοί επιθυμούν με σκοπό να έχουν μεγαλύτερο ελεύθερο χώρο για αποθήκευση αρχείων. Η εφαρμογή, αποκρύπτει επιτυχώς την υλοποίηση της διαχείρισης των συνδεδεμένων μέσων αποθήκευσης νέφους παρέχοντας στον χρήστη την αίσθηση ότι όλα τα αρχεία του ανήκουν σε ένα κοινό μέσο αποθήκευσης νέφους.

Σημαντικό ρόλο, έπαιξε και το σχεδιαστικό κομμάτι της εφαρμογής, παρέχοντας έτσι στον χρήστη μία ευχάριστη εμπειρία χρήσης, έχοντας γνώση ανά πάσα στιγμή για την επιτυχία ή την αποτυχία των λειτουργιών που εκτελεί μέσω διαδραστικών ειδοποιήσεων.

Επιπλέον, η εφαρμογή υποστηρίζει την προσθήκη νέων μέσων αποθήκευσης νέφους χάρη στην σχεδίαση που κάναμε, έτσι ώστε αν μελλοντικά χρειαστεί η προσθήκη ενός νέου μέσου, αυτό να είναι εφικτό με ευκολία.

Τέλος, μέσω μερικών κοινών τεχνικών ασφάλειας όπως ο κατακερματισμός κωδικών πρόσβασης και η κρυπτογραφία των κλειδιών πρόσβασης των μέσων, καταφέραμε να εφαρμόσουμε την ασφαλή αποθήκευση των ευαίσθητων στοιχείων των χρηστών σε περίπτωση κακόβουλης πρόσβασης στην βάση δεδομένων, αποτρέποντας έτσι την διαρροή τους.

7.2 Μελλοντικές επεκτάσεις

Προσθήκη υποστήριξης νέων μέσων αποθήκευσης νέφους

Όπως αναφέραμε και στην ενότητα 4.2, μία από τις χρήσιμες μελλοντικές επεκτάσεις που θα μπορούσαν να προστεθούν στην υπάρχουσα υλοποίηση, είναι η προσθήκη υποστήριξης σύνδεσης νέων μέσων αποθήκευσης νέφους.

Τα τελευταία χρόνια, εμφανίζονται όλο και περισσότερα μέσα αποθήκευσης νέφους που αποκτούνε δημοφιλή σε γρήγορους ρυθμούς, επομένως δεν μπορούμε να είμαστε σίγουροι αν σε λίγο καιρό από τώρα, τα δημοφιλέστερα μέσα θα παραμείνουν τα ίδια. Σε κάθε περίπτωση, το κομμάτι προσθήκης νέου μέσου αποθήκευσης νέφους είναι επεκτάσιμο οπότε δεν θα υπάρξει δυσκολία στο να γίνουν νέες προσθήκες. Μερικά από τα μέσα αποθήκευσης νέφους που ενδεχομένως να γίνουν σε μικρό χρονικό διάστημα αρκετά περισσότερο δημοφιλή, είναι τα εξής:

- Amazon Cloud Drive
- Box Cloud Storage
- pCloud Cloud Storage

Και στα τρία αυτά μέσα, υπάρχει ανοιχτό API για την διαχείριση των αρχείων τους, επομένως, χρησιμοποιώντας τα API τους η σύνδεση με την εφαρμογή μας θα είναι σίγουρα εφικτή.

Βελτίωση των μέτρων ασφάλειας

Τα μέτρα ασφάλειας μπορούν να βελτιωθούν κατά πολύ περαιτέρω. Θα χρειαστεί στο μέλλον να βρεθεί λύση στο πρόβλημα κακόβουλης πρόσβασης χρήστη στο περιβάλλον του εξυπηρετητή. Έχοντας ένας κακόβουλος χρήστης πρόσβαση στον εξυπηρετητή, εκθέτεται το κλειδί κρυπτογράφησης γεγονός που έχει ως αποτέλεσμα την αποκρυπτογράφηση των κλειδιών των μέσων αποθήκευσης νέφους, στην βάση. Επιπλέον, θα πρέπει να διαπιστωθεί η λειτουργία της ασφαλούς μεταφοράς των “cookies” μέσω ασφαλών καναλιών με την χρήση SSL πιστοποιητικών.

Βελτίωση του αλγόριθμου αναδιάταξης αρχείων

Ο αλγόριθμος αναδιάταξης αρχείων που αναφέραμε στην υπο-ενότητα 4.1.4.4 μπορεί να βελτιωθεί έτσι ώστε η αναδιάταξη αρχείων να είναι πιο αποδοτική.

Βιβλιογραφία

- [1] The difference between a library and a framework.
(<https://www.programcreek.com/2011/09/what-is-the-difference-between-a-java-library-and-a-framework/>)
- [2] MERN stack explained by MongoDB .
(<https://www.mongodb.com/mern-stack>)
- [3] Most popular technology stack to choose from full stack.
(<https://el.bccrwp.org/compare/most-popular-technology-stack-to-choose-from-full-stack-vs-mean-stack-vs-mern-stack-in-2019-6bafbb/>)
- [4] User “AritraSen”. Node.js Event Loop explanation, 13 February 2020.
(<https://www.geeksforgeeks.org/node-js-event-loop/>)
- [5] MongoDB Databases and Collections overview.
(<https://docs.mongodb.com/manual/core/databases-and-collections/>)
- [6] Nick Parsons. MongoDB Atlas technical overview benefits, 19 January 2017.
(medium.com/@nparsons08/mongodb-atlas-technical-overview-benefits-9e4cff27a75e)
- [7] Gougousis Alexandros. Introduction to the REST Architecture.
(<http://www.users.dpem.tuc.gr/gougousis/rest/>)

- [8] Zell Liew. Understanding and using REST APIs, 17 January 2018.(<https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/>)
- [9] Armado Fox, David Patterson (Μετάφραση/Επιστημονική Επιμέλεια: Στέργιος Αναστασιάδης), ΤΕΧΝΟΛΟΓΙΑ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ ΩΣ ΥΠΗΡΕΣΙΑ, Εκδόσεις Κλειδάριθμος, σελίδα 78, 2017
- [10] Coda Hale, How To Safely Store A Password, 31 January 2010 (<https://codahale.com/how-to-safely-store-a-password/>)