ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	Λογισμικό διαμοίρασης αρχείων μέσω cloud Cloud application for sharing files
Ονοματεπώνυμο Φοιτητή	Κλαούντιο Ζντράβα
Πατρώνυμο	Αγκρόν
Αριθμός Μητρώου	Π/18223
Επιβλέπων	Ευάγγελος Σακκόπουλος, Αναπληρωτής Καθηγητής

Ημερομηνία Παράδοσης Σεπτέμβριος 2024

Copyright©

Επιτελική Σύνοψη

Η παρούσα διπλωματική εργασία παρουσιάζει την ανάπτυξη μιας διαδικτυακής εφαρμογής διαμοιρασμού αρχείων και διαχείρισης φακέλων, η οποία υλοποιήθηκε με τη χρήση της Java και του Spring Framework. Η εφαρμογή στοχεύει στη διευκόλυνση της ασφαλούς και αποτελεσματικής κοινής χρήσης και αποθήκευσης αρχείων με τη χρήση τεχνολογιών νέφους, επιτρέποντας στους χρήστες να ανεβάζουν, να κατεβάζουν και να μοιράζονται αρχεία μέσω του διαδικτύου. Το backend αναπτύσσεται σε Java, ενώ το frontend κατασκευάζεται με χρήση HTML, CSS και JavaScript, ενώ η εφαρμογή είναι προσβάσιμη μέσω ενός διακομιστή ιστού χρησιμοποιώντας ένα καθορισμένο URI.

Το σύστημα ενσωματώνει βασικές αρχές ασφαλείας, όπως η εμπιστευτικότητα, η ακεραιότητα και η διαθεσιμότητα, ώστε να διασφαλίζεται ότι τα δεδομένα παραμένουν προστατευμένα, ακριβή και εύκολα προσβάσιμα σε εξουσιοδοτημένους χρήστες. Ο σχεδιασμός είναι εμπνευσμένος από το Dropbox, αν και το παρόν έργο υλοποιεί ένα απλουστευμένο υποσύνολο των λειτουργιών του. Η εφαρμογή επιτρέπει στους χρήστες να δημιουργούν φακέλους, να μεταφορτώνουν και να προβάλλουν αρχεία, να διαχειρίζονται κοινόχρηστους πόρους και να ελέγχουν την πρόσβαση άλλων χρηστών. Επίσης έχει δοθεί έμφαση στην υλοποιήση των αλγορίθμων που επιτρέπουν το αποδοτικό ανέβασμα των αρχείων καθώς και τη αποθήκευση τους.

Στην τρέχουσα επανάληψή της, η εφαρμογή εκτελείται τοπικά χρησιμοποιώντας έναν ενσωματωμένο διακομιστή Tomcat. Οι μελλοντικές εκδόσεις θα περιλαμβάνουν τις απαραίτητες ρυθμίσεις για την ανάπτυξη σε ένα πλήρες περιβάλλον διακομιστή. Η διπλωματική εργασία συζητά επίσης πρόσθετα πιθανά χαρακτηριστικά και βελτιώσεις για τη βελτίωση της λειτουργικότητας της εφαρμογής.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Κεφάλαιο 1ο

1 Εισαγωγή

Η παρούσα διατριβή αφορά την ανάπτυξη ενός λογισμικού διαμοίρασης και φακέλων υλοποιημένο με χρήση της γλώσσας προγραμματισμού Java και του Spring Framework. Η εφαρμογή θα αποτελείται απο ένα web application το οποίο θα χρησιμοποιεί ως backend γλώσσα προγραμματισμού την Java ενώ το frontend χρησιμοποιεί τις τεχνολογίες HTML/CSS και JavaScript. Η εφαρμογή θα είναι εγκατεστημένη σε κάποιον server και θα είναι προσβάσιμη μέσω του παρακάτω URI :
http://{server-uri}:{port-number}/data-collections/collections.
Πρός το παρόν η εφαρμογή τρέχει μόνο στον localhost και χρησιμοποιεί έναν embedded tomcat για να δέχεται http requests. Για να "τρέχει" σε οποιονδήποτε server θα χρειαστεί να αλλάξουν μερικά configurations , κάτι το οποίο μπορεί να γίνει σε επόμενο release.

Η εφαρμογή είναι σχεδιασμένη με τρόπο τέτοιο ώστε να ενσωματώνει τις τρείς παρακάτω ένοιες που αποτελούν κανόνες για ένα ασφαλές λογισμικό. Οι κανόνες είναι οι εξής:

- **Εμπιστευτικότητα (confidentiality):** Είναι έννοια στενά συνδεμένη με την ιδιωτικότητα (privacy) και τη μυστικότητα (secrecy). Αφορά τη μη αποκάλυψη των ευαίσθητων πληροφοριών σε χρήστες που δεν έχουν την κατάλληλη εξουσιοδότηση.
- **Ακεραιότητα (integrity):** Αφορά τη δυνατότητα τροποποιήσεων (προσθήκες, διαγραφές και μεταβολές) των πληροφοριών. Μόνο σε κατάλληλα εξουσιοδοτημένους χρήστες πρέπει το σύστημα να επιτρέπει τέτοιου είδους ενέργειες. Έτσι διαφυλάσσεται η ακρίβεια και η πληρότητα των περιεχομένων ενός πληροφοριακού συστήματος.
- **Διαθεσιμότητα (availability):** Αφορά τη δυνατότητα άμεσης πρόσβασης στις πληροφορίες, στις υπηρεσίες και γενικότερα σε όλους τους πόρους πληροφορικής τεχνολογίας (IT resources) όταν ζητούνται, χωρίς αδικαιολόγητες καθυστερήσεις.

Ο σχεδιασμός της εφαρμογής καθώς και οι λειτουργίες που θα εμπεριέχονται σε αυτήν, έχει γίνει με γνώμονα το γνωστό σε όλους σύστημα διαμοίρασης αρχείων *Dropbox*. Λόγω του ότι το Dropbox είναι αρκετά πολύπλοκο σύστημα, στην εργασία αύτην θα σχεδιαστούν και υλοποιηθούν μερικές μόνο απο το πλήθος των δυνατοτήτων που προσφέρει. Στην συνέχεια θα δοθεί ένας σχεδιασμός και μια περιγραφή για τήν χρήση της εφαρμογής καθώς και τα απαιτούμενα βήματα που χρειάζεται να εκτελέσει ο χρήστης ώστε να επιτελέσει τον στόχο του. Τέλος, θα συζητηθούν επιπλέον λειτουργίες που θα μπορούσαν να θεωρηθούν χρήσιμες για την εφαρμογή μας.

Η εφαρμογή, όπως αναφέραμε, θα είναι μια web εφαρμογή μέσω της οπόιας ο χρήστης θα μπορεί να διαχειρίζεται αρχεία στο cloud μέσω του internet.
Πιο συγκεκριμένα οι δυνατότητες που θα προσφέρει θα είναι οι εξής:

- Εγγραφή του χρήστη.
- Σύνδεση/Αποσύνδεση απο την εφαρμογή.
- Δημιουργία φακέλου.
- Ανέβασμα αρχείου.
- Κατέβασμα αρχείου.
- Προβολή αρχείου στον browser.
- Διαμόιρσμα φακέλου/αρχείου με άλλους χρήστες.
- Διαγραφή φακέλου/αρχείου από όλους τους χρήστες που έχουν πρόσβαση σε αυτό.
- Εμφάνιση φακέλων/αρχείων που ανήκουν στον χρήστη.
- Εμφάνιση φακέλων/αρχείων που διαμοιράστηκαν στον χρήστη.
- Εμφάνιση φακέλων/αρχείων που ο χρήστης έχει διαμοιράσει σε άλλους χρήστες.
- Αφαίρεση πρόσβασης διαμοιρασμένου αρχείου σε τρίτους χρήστες.

1.1 Περιγραφή του υπό μελέτη προβλήματος

Το πρόβλημα που αντιμετωπίζεται σε αυτή τη διατριβή περιστρέφεται γύρω από την ανάγκη για μια ασφαλή, αποτελεσματική και προσιτή μέθοδο κοινής χρήσης και διαχείρισης αρχείων σε περιβάλλον που βασίζεται σε σύννεφο. Οι χρήστες απαιτούν μια πλατφόρμα όπου μπορούν εύκολα να ανταλλάσσουν δεδομένα, να αποθηκεύουν αρχεία και να ελέγχουν την πρόσβαση σε κοινόχρηστους πόρους. Ωστόσο, η διασφάλιση ότι μια τέτοια πλατφόρμα πληροί βασικές απαιτήσεις ασφαλείας —όπως η αποτροπή μη εξουσιοδοτημένης πρόσβασης σε ευαίσθητες πληροφορίες, η διατήρηση της ακρίβειας των δεδομένων και η διασφάλιση της διαθεσιμότητας— μπορεί να είναι πρόκληση.

1.2 Σκοπός και στόχοι της εργασίας

Σκοπός της εφαρμογής είναι να μπορεί έυκολα και με ασφάλεια, να παρέχει την δυνατότητα στους χρήστες να μπορούν να διαμοιράζονται αρχεία μεταξύ τους με σκοπό την συνεργασία για κάποιον σκοπό ή απλώς την εύκολη μεταφορά πληροφορίας. Επίσης η εφαρμογή στοχεύει στην παροχή αποθηκευτικού χώρου στους χρήστες χρησιμοποιώντας την τεχνολογιά cloud.

1.3 Βασικοί ορισμοί

Tomcat: Το Apache Tomcat είναι ένας διακομιστής ιστού ανοιχτού κώδικα και κοντέινερ servlet που αναπτύχθηκε από το Ίδρυμα Λογισμικού Apache. Εφαρμόζει τεχνολογίες Java Servlet, JavaServer Pages (JSP) και WebSocket, επιτρέποντας σε εφαρμογές web που βασίζονται σε Java να εκτελούνται σε περιβάλλον διακομιστή. Χρησιμοποιείται ευρέως για την ανάπτυξη εφαρμογών ιστού Java.

Server: Διακομιστής είναι ένα σύστημα υπολογιστή ή λογισμικό που παρέχει υπηρεσίες, πόρους ή δεδομένα σε άλλες συσκευές, γνωστές ως πελάτες, μέσω δικτύου. Οι διακομιστές μπορούν να φιλοξενούν ιστότοπους, να διαχειρίζονται βάσεις δεδομένων, να αποθηκεύουν αρχεία ή να χειρίζονται αιτήματα από υπολογιστές-πελάτες ή εφαρμογές web.

Web application: Μια εφαρμογή Ιστού είναι μια εφαρμογή λογισμικού που εκτελείται σε διακομιστή Ιστού και είναι προσβάσιμη μέσω ενός προγράμματος περιήγησης Ιστού μέσω ενός δικτύου, συνήθως στο Διαδίκτυο. Σε αντίθεση με τις παραδοσιακές εφαρμογές επιτραπέζιου υπολογιστή, οι εφαρμογές web δεν χρειάζεται να εγκατασταθούν στη συσκευή του χρήστη και μπορούν να χρησιμοποιηθούν σε πολλές πλατφόρμες.

Java: Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού υψηλού επιπέδου, σχεδιασμένη για ευελιξία, φορητότητα και επεκτασιμότητα. Χρησιμοποιείται ευρέως για την κατασκευή διαδικτυακών εφαρμογών, εφαρμογών για κινητές συσκευές, εταιρικών συστημάτων και πολλά άλλα. Οι εφαρμογές Java μπορούν να εκτελεστούν σε οποιοδήποτε σύστημα που διαθέτει Java Virtual Machine (JVM), καθιστώντας το μια γλώσσα "γράψτε μια φορά, εκτελέστε οπουδήποτε".

Spring Framework: Το Spring Framework είναι ένα ολοκληρωμένο πλαίσιο ανοιχτού κώδικα για τη δημιουργία εφαρμογών Java. Παρέχει υποστήριξη υποδομής για διάφορες πτυχές των εταιρικών εφαρμογών, όπως η ένεση εξάρτησης, η πρόσβαση δεδομένων και η διαχείριση συναλλαγών, ενώ απλοποιεί την ανάπτυξη διαδικτυακών εφαρμογών, API και μικροϋπηρεσιών που βασίζονται σε Java.

HTML/CSS:

Η HTML (HyperText Markup Language) είναι η τυπική γλώσσα σήμανσης που χρησιμοποιείται για τη δημιουργία και τη δομή περιεχομένου στον Ιστό. Καθορίζει τα στοιχεία που συνθέτουν μια ιστοσελίδα, όπως επικεφαλίδες, παραγράφους, συνδέσμους και εικόνες.

Το CSS (Cascading Style Sheets) είναι μια γλώσσα φύλλου στυλ που χρησιμοποιείται για την περιγραφή της παρουσίασης ενός εγγράφου γραμμένου σε HTML ή XML. Το CSS ελέγχει τη διάταξη, το σχεδιασμό και την οπτική μορφοποίηση των ιστοσελίδων, συμπεριλαμβανομένων των χρωμάτων, των γραμματοσειρών και των διαστημάτων.

Cloud: Στους υπολογιστές, το "σύννεφο" αναφέρεται σε ένα δίκτυο απομακρυσμένων διακομιστών που φιλοξενούνται στο Διαδίκτυο που αποθηκεύουν, διαχειρίζονται και επεξεργάζονται δεδομένα. Το cloud computing επιτρέπει στους χρήστες να έχουν πρόσβαση σε δεδομένα και εφαρμογές από οπουδήποτε, χωρίς να χρειάζονται φυσικούς διακομιστές ή αποθήκευση στα τοπικά τους μηχανήματα. Το cloud παρέχει επεκτασιμότητα, ευελιξία και οικονομική αποδοτικότητα για επιχειρήσεις και ιδιώτες.

1.3.1 Μη λειτουργικές ανάγκες

Για να μπορέσει ο χρήστης να στήσει επιτυχώς της εφαρμογή στον υπολογιστή του, θα πρέπει να έχει εγκατεστημένη την Java καθώς και τον web server ονόματι Tomcat. Ύστερα θα χρειαστεί να προσθέσει το αρχείο με κατάληξη war, που είναι και το εκτελέσιμο του κώδικα, μέσα στον φάκελο webapps μέσα στον μονοπάτι που βρίσκεται εγκατεστημένος ο Tomcat. Ο Tomcat κάνει detect το εκτελέσιμο και ύστερα το κάνει deploy.

Μόλις αναπτυχθεί, η διαδικτυακή εφαρμογή θα είναι προσβάσιμη μέσω του προγράμματος περιήγησης στη διεύθυνση http://{server-uri}:{port}/{app-name}, όπου {app-name} είναι το όνομα του αρχείου WAR.

1.4 Παραδοτέα της εργασίας

Στα παραδοτέα της εργασίας περιλαμβάνονται τα εξής:

1. Το γραπτό της πτυχιακής ερασίας που περιλαμβάνει την ανάλυση και τον σχεδιασμό της λύσης καθώς και την επεξήγηση της υλοποίησης του κώδικα.
2. Το εκτελέσιμο αρχείο με κατάληξη war.
3. Την βάση δεδομένων για την αποθήκευση των αρχείων και των χρηστών.

1.5 Δομή της εργασίας

Η εργασία παρουσιάζεται με μια μορφή απο πάνω προς τα κάτω. Δηλαδή, πρώτα γίνεται μια παρουσιάση της ανάλυσης του προβλήματος και τον σχεδιασμό μιας επίλυσης με την βοήθεια σχεδιαγραμμάτων για να είναι και πιο κατανοητό στον χρήστη και ύστερα παρουσιάζεται η υλοποίηση της επίλυσης μέσω της γλώσσας προγραμματισμού Java και των απαραίτητων τεχνολογιών.

Κεφάλαιο 2ο

2 Ανάλυση και σχεδίαση

2.1 Σύγκριση εφαρμογής με υπάρχοντα συστήματα

Παρακάτω παρουσιάζονται υπάρχοντα συστήματα διαμοίρασης αρχείων μέσω cloud όπως και οι λειτουργίες που αυτά περιέχουν. Επιπλέον πραγματοποιείται σύγκριση μεταξύ της εφαρμογής που υλοποιήθηκε και των συστημάτων αυτών.

	Πτυχιακή εργασία	Dropbox	Google drive	OneDrive
Sharing Files While Controlling Access	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ
Assigning Tasks to Employees With Comments	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ
Sorting Your Files by Size	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ
Cloud storage	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ
Doing Advanced Search	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ
Share with people outside your organisation	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ
Drag and drop files	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ

Use the mobile app to scan straight to your drive	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ
Starred folders and files	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ
View any type of file	ΝΑΙ μονο για pdf, csv, txt, jpg, png, xlsx	ΝΑΙ	ΝΑΙ	ΝΑΙ
Backup files and folders	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ
Sync	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ
Find everything you're sharing	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ
Restore files and folders	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ

2.2 Use case diagram

Στα παρακάτω διαγράμματα απεικονίζονται οι καταστάσεις χρήσης της εφαρμογής απο τον χρήστη.
Στο πρώτο διάγραμμα παρουσιάζεται η σελίδα σύνδεσης/εγγραφής του χρήστη στην εφαρμογή. Ο χρήστης συνδέεται με χρήση του email του καθώς και τον κωδικό του. Σε περίπτωση που δεν είναι εγγεγραμμένος έχει την επιλογή να κάνει εγγραφή πατώντας το κουμπί Register, το οποίο βρίσκεται κάτω στήν σελίδα Login, και αφού μεταφερθεί στη νέα σελίδα του ζητείται να συμπλήρώσει τα στοιχεία του. Κατόπιν επιτυχής εγγραφής, μεταφέρεται στην σελίδα σύνδεσης για να εισάγει τα στοιχεία του. Με την επιτυχή του σύνδεση μεταφέρεται στο Dashboard όπου μπορεί να κάνει χρήση των λειτουργιών της εφαρμογής που θα συζητηθούν παρακάτω.

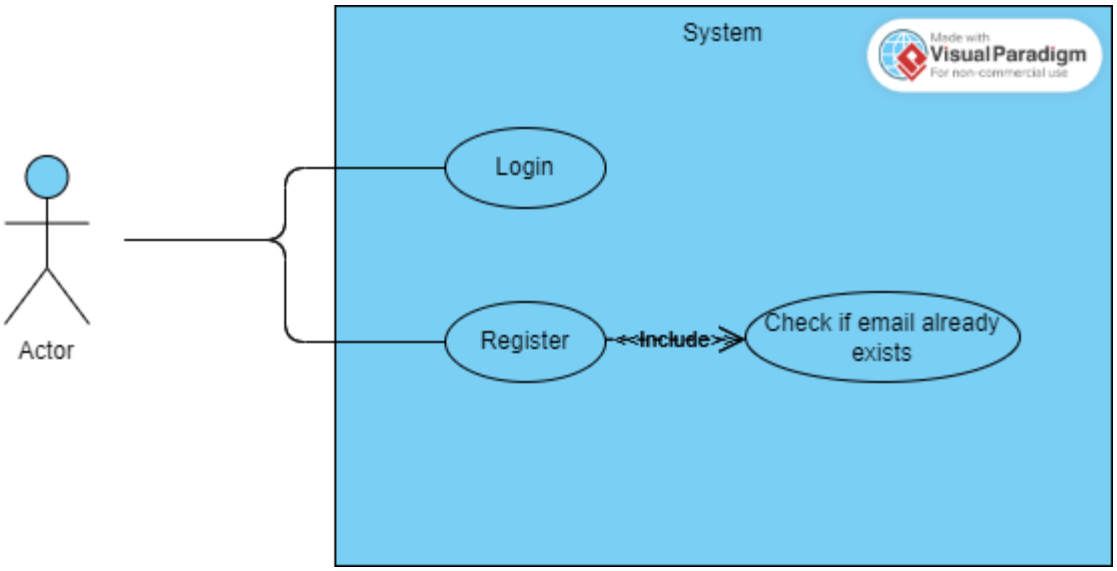


Figure 1 Use case diagram - Σύστημα σύνδεσης

Πρωτέυον μονοπάτι:

1. Ο επισκέπτης κάνει σύνδεση ή εγγραφή στο σύστημα.
2. Με την επιτυχή σύνδεση ή εγγραφή του επισκέπτη, πλέον σαν χρήστης συνδέεται στο διαχειριστικό του σύστημα της εφαρμογής.
3. Η περίπτωση χρήσης τελειώνει με επιτυχία.

Εναλλακτικό μονοπάτι:

Αποτυχία σύνδεσης στο σύστημα

a. Ο επισκέπτης προσπαθεί να κάνει σύνδεση με λάθος στοιχεία σύνδεσης ή ο χρήστης προσπαθεί να κάνει εγγραφή με email που υπάρχει ήδη στο σύστημα.
b. Εκτυπώνεται μήνυμα αποτυχίας στον χρήστη και φορτώνεται νέα σελίδα για να εισάγει τα σωστά στοιχεία του.
c. Η περίπτωση χρήσης τελειώνει με αποτυχία.

Το δεύτερο διάγραμμα αφορά την βασική αλληλεπίδραση του χρήστη με το σύστημα. Ο χρήστης αφού συνδεθεί επιτυχώς στο σύστημα έχει μεταξύ άλλων τις εξής επιλογές:

- Δημιουργία φακέλου.
- Ανέβασμα αρχείου.
- Κατέβασμα αρχείου.
- Προβολή αρχείου στον browser.

- Διαμόιρσμα φακέλου/αρχείου με άλλους χρήστες.
- Διαγραφή φακέλου/αρχείου.
- Εμφάνιση φακέλων/αρχείων που ανήκουν στον χρήστη.
- Εμφάνιση φακέλων/αρχείων που διαμοιράστηκαν στον χρήστη.
- Εμφάνιση φακέλων/αρχείων που ο χρήστης έχει διαμοιράσει σε άλλους χρήστες.
- Αφαίρεση πρόσβασης διαμοιρασμένου αρχείου σε τρίτους χρήστες.

Ο φάκελος που δημιουργεί ο χρήστης καθώς και το αρχείο που ανεβάζει, αποθηκέυονται στην βάση με κάποιο πρωτεύον κλειδί που είναι μοναδικό για το κάθε αρχείο. Ο τρόπος με τον οποίο δημιουργείται το id του αρχείου πραγματοποιείται σύμφωνα με τήν παρακάτω μέθοδο.

id = hash(concat(username, parent_foldername, filename/foldername))

Δηλαδή δύο αρχεία με το ίδιο όνομα μπορούν να υπάρξουν στην βάση δεδομένων μονο εάν ανέβηκαν απο διαφορετικό χρήστη ή ανήκουν σε διαφορετικό φάκελο. Στην περίπτωση που ο χρήστης προσπαθήσει να ανεβάσει αρχείο ή φάκελο που υπάρχει ήδη στον υπάρχων κατάλογο προκύπτουν τα εξής σενάρια:

- Εάν είναι φάκελος, εκτυπώνουμε μήνυμα λάθους και η διαδικασία αποτυγχάνει.
- Εάν είναι αρχείο, ανεβάζουμε μόνο τα μέρη του αρχείου που έχουν γίνει updated.

Αφου διαμοιράσει το αρχείο ή τον φάκελο σε τρίτους χρήστες, αυτόματα στέλνεται ένα email στον ιδιοκτήτη του φακέλου/αρχείου για να εγκρίνει ή να απορρίψει το αίτημα. Σημειώνεται πως το mail αποστέλλεται μόνον σε περίπτωση που το αρχείο που διαμοιράζεται δεν ανήκει στον χρήστη που το διαμοιράζει. Με αυτόν τον τρόπο επιτυγχάνεται ασφάλεια στο σύστημα, καθώς ο ιδιοκτήτης του αρχείου θα πρέπει να είναι ενήμερος για το ποιός έχει πρόσβαση στο αρχείο/φάκελο του. Εφόσον αποδεχτεί το αίτημα ο κάτοχος του αρχείου/φακέλου, εμφανίζεται αυτόματα το αρχείο/φάκελος στον παραλήπτη. Διαφορετικά διαγράφεται η προσωρινή εγγραφή απο την βάση δεδομένων και ο χρήστης δεν ενημερώνεται ούτε αποκτά ποτέ πρόσβαση στον φάκελο/αρχείο.

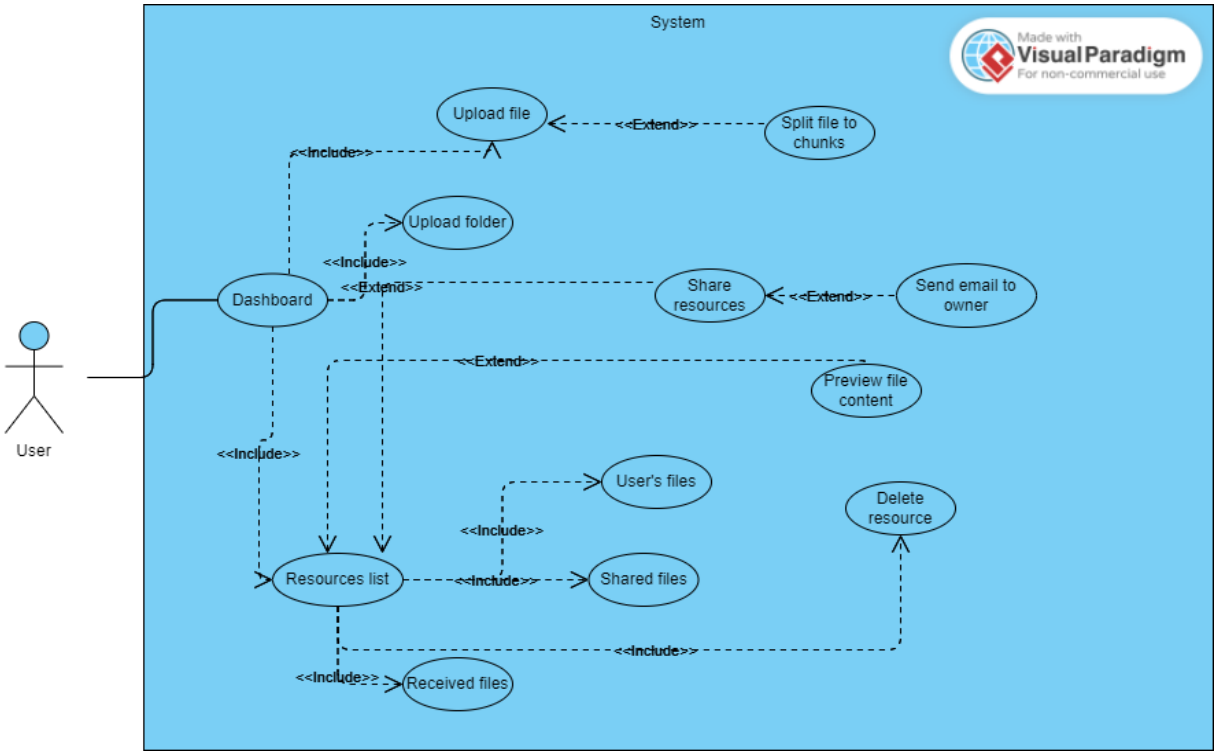


Figure 2 Use case diagram - Σύστημα διαχειριστικού

Πρωτέυον μονοπάτι:

1. Με την επιτυχή σύνδεση ή εγγραφή του, πλέον σαν χρήστης του συστήματος ανακατευθύνεται στο διαχειριστηκό του, όπου μπορεί να δει τα αρχεία και τους φακέλους που έχει πρόσβαση.
2. Απο την λίστα με τα resources του, μπορεί να διαμοιράσει κάποιο αρχείο ή φακελο είτε εισάγωντας τα emails των χρηστών που επιθυμεί είτε επιλέγοντας να το διαμοιράσει με όλους όσους έχουν το link. Αυτόματα στέλνεται email στον κάτοχο του resource για να κάνει approve.
3. Επίσης υπάρχουν οι επιλογές create folder και upload file, όπου μετά την δημιουργία του ή την μεταφόρτωση του αρχείου εμφανίζεται αμέσως στη λίστα με τα resources.
4. Έχουμε και τις λειτουργίες delete, preview και download. Με την εντολή delete διαγράφουμε το αρχείο ή τον φάκελο με όλα τα resources που περιέχει, οι επιλογές preview και download αφορούν μόνο αρχεία και δίνουν την δυνατότητα

στον χρήστη να δει το περιεχόμενο του αρχείου ή να το κατεβάσει στον υπολογιστή του αντίστοιχα.

5. Επίσης έχουμε την επιλογή revoke sharing όπου ο χρήστης ανακαλεί το διαμοίρασμα του resource που έχει κάνει.
6. Τέλος ο χρήστης μπορεί να δεί στο διαχειριστικό του εκρεμμή αιτήματα πρός αποδοχή.
7. Η περίπτωση χρήσης τελείώνει με επιτυχία.

Εναλλακτικά μονοπάτια:

1. Ο χρήστης κάνει copy το link του resource και ενώ το στέλνει σε τρίτους χρήστες, δεν έχει πατήσει το κουμπί share ώστε να καταχωρηθεί το αίτημα οπότε και οι χρήστες δεν έχουν πρόσβαση.
2. Ο κάτοχος του resource απορρίπτει την πρόσβαση σε κάποιο αρχείο/φάκελο του.
3. Ο χρήστης επιθυμεί να ανεβάσει αρχείο σε μορφή που δεν υποστηρίζεται.
4. Δημιουργία φακέλου με όνομα που υπάρχει ήδη στον τρέχων κατάλογο.
5. Η περίπτωση χρήσης τελειώνει με αποτυχία.

2.3 Swimlane activity diagram

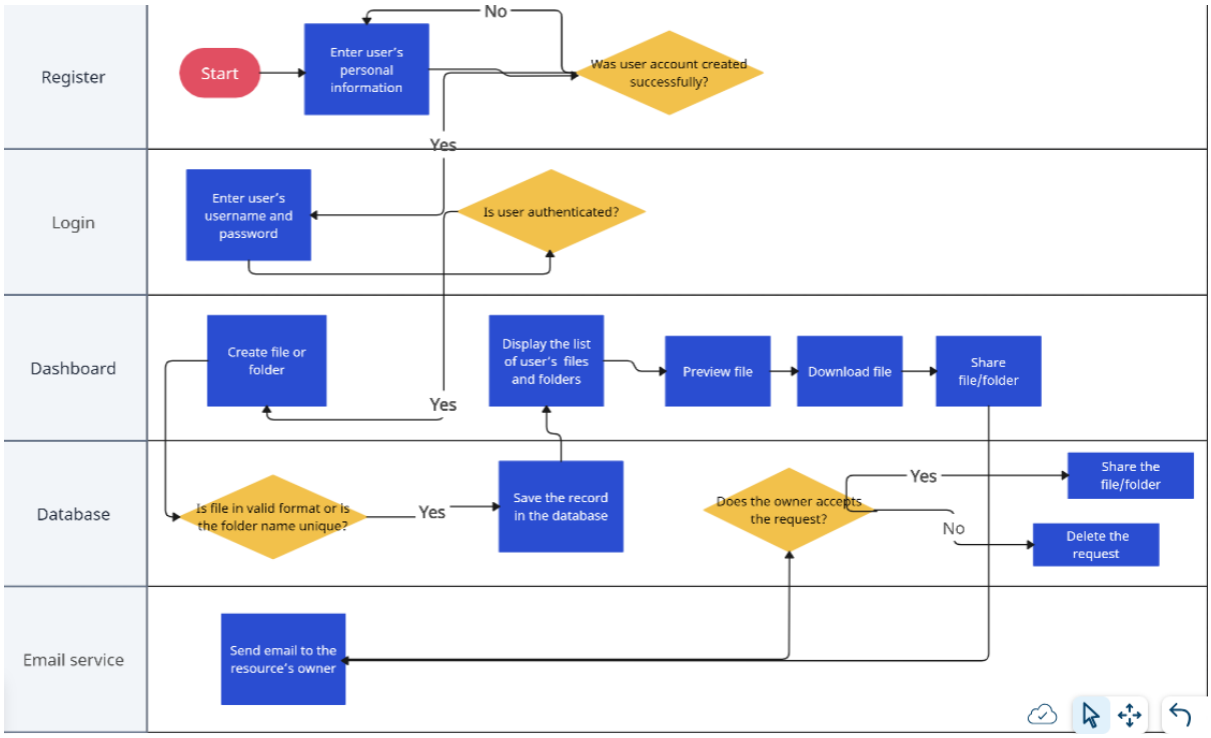


Figure 3 Activity diagram - Επικοινωνία μεταξύ των components του συστήματος

2.4 Sequence diagram

Στο σχήμα παρουσιάζεται όλη η διαδικασία των μεθόδων που χρησιμοποιούνται για να υποστηρίξουν τις λειτουργίες του συστήματος. Ο επισκέπτης της εφαρμογής αρχικά ανακατεύθυνεται στο να κάνει login. Απο εκεί επιλέγει να φτιάξει κάποιο account μέσω κατάλληλου συνδέσμου. Αφού εγγραφεί επιτυχώς, ανακατεύθυνεται στην αρχική σελίδα που του ζητείται να συνδεθεί με τα email και password που εισήγαγε προηγουμένως στο σύστημα. Κατόπιν επιτυχής σύνδεσης ανακατεύθυνεται στο Dashboard όπου έχει την επιλογή είτε να ανεβάσει κάποιο αρχείο απο τον υπολογιστή του είτε να δημιουργήσει κάποιον φάκελο. Εφόσον επιλέξει να δημιουργήσει καινούριο φάκελο, ανακατεύθυνεται στο εσωτερικό του φακέλου απο όπου μπορεί είτε να δημιουργήσει κάποιον υποφάκελο είτε να ανεβάσει κάποιο αρχείο. Εφόσον επιλέξει να ανεβάσει κάποιο αρχείο, τότε το ίδιο το όνομα του αρχείου θα εμφανίζεται στην αρχικη του οθόνη. Στην αρχική του οθόνη λοιπον εμφανίζονται όλα τα αρχεία και οι φάκελοι που έχει πρόσβαση. Μέσω της λίστας των resources ο χρήστης επιλέγει κάποιο αρχείο να διαμοιράσει και στην συνέχεια εμφανίζεται παράθυρο για να εισάγει τα emails των χρηστών οι οποίοι επιθυμεί να έχουν πρόσβαση στο αρχείο/φάκελο. Στην συνέχεια δημιουργείται προσωρινές εγγραφές στην βάση δεδομένων με εκρεμμή αιτήματα, μία για κάθε παραλήπτη. Ύστερα στέλνεται email στον κάτοχο του αρχείου/φακέλου ώστε να επισκεφθεί το Dashboard και να αποδεχτεί τα αιτήματα. Αφού

αποδεχτεί ο κάτοχος του αρχείου/φάκελου τα αιτήματα, οι παραλήπτες μπορούν πλέον να έχουν πρόσβαση σε εκείνα μέσω της σελίδας του Dashboard, αφού εμφανίζονται στη λίστα με τα υπόλοιπα resources.

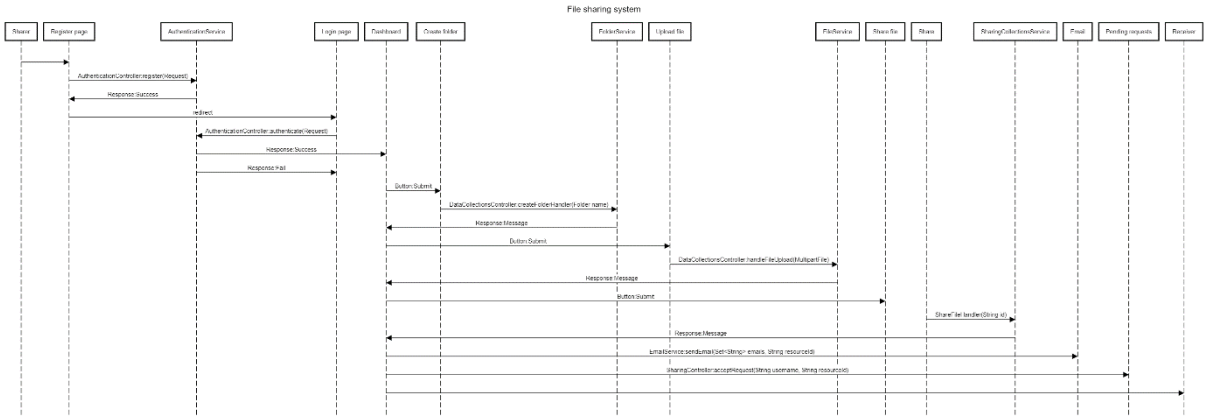


Figure 4 Sequence diagram εφαρμογής

Κεφάλαιο 3ο

3 Παρουσίαση εκτέλεσης της εφαρμογής

3.1 Παρουσίαση

Με το που εισάγει την διεύθυνση http://localhost:8082/data-collections/collections ο χρήστης στον φυλλομετρητή του, εμφανίζεται η πρακάτω σελίδα που του ζητείται να κάνει login.

Figure 5 Login page

Αν πατήσει το κουμπι Sign up μεταφέρεται σε μιά άλλη σελίδα οπου του ζητείται να συμπληρώσει τα στοιχεία του για να κάνει εγγραφή.

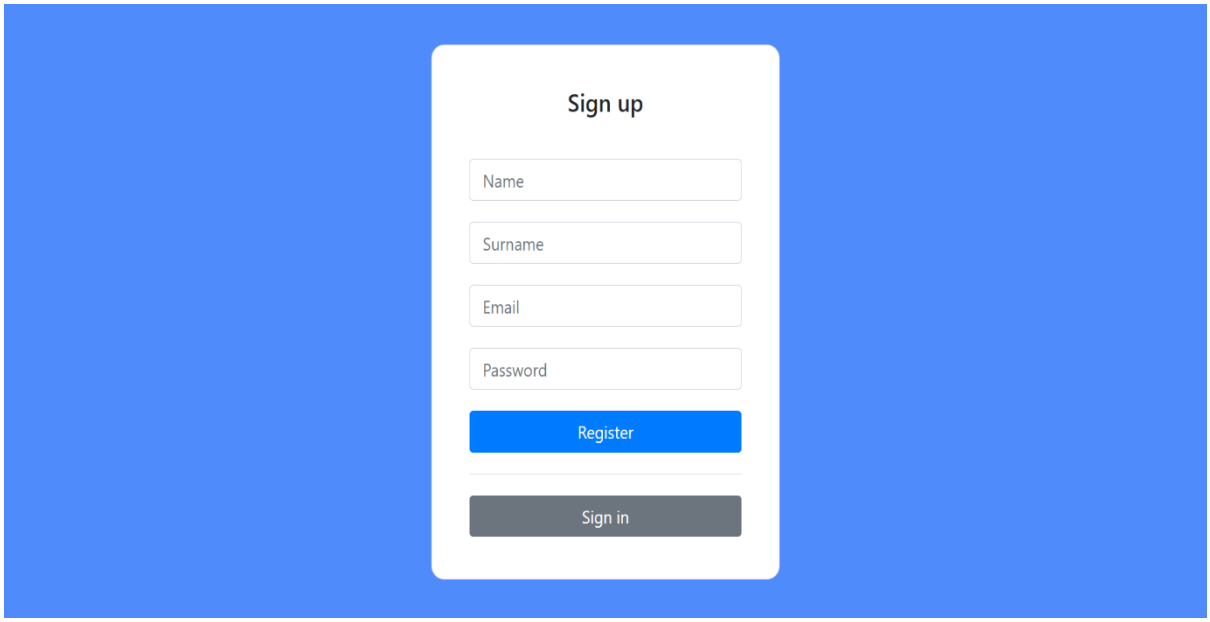


Figure 6 Register page

Αφού συμπληρώσει τα στοιχεία του και πατήσει το κουμπί Register αν όλα πάνε καλά η εγγραφή πραγματοποιείται με επιτυχία και ο χρήστης ανακατευθύνεται στην προηγούμενη σελίδα για να συμπληρώσει το email και password, που εισήγαγε προηγουμένως, ωστε να μπορέσει να συνδεθεί στο Dashboard.

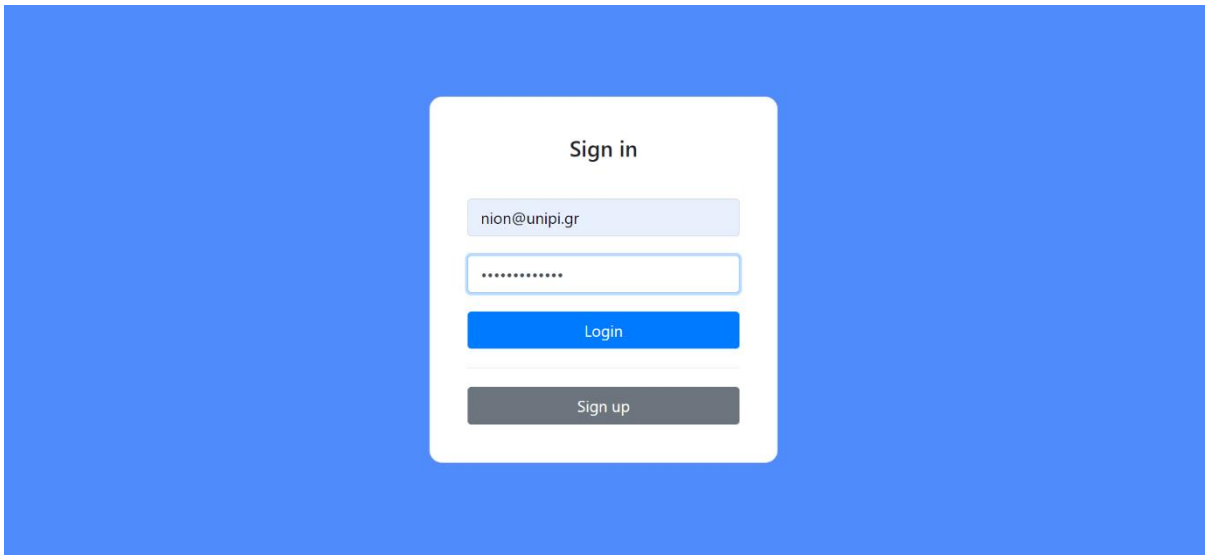


Figure 7 Login page

Εάν κατά την διάρκεια της σύνδεσης ο χρήστης δώσει λάθος στοιχεία, εμφανίζεται κατάλληλο μήνυμα επάνω στην φόρμα “Bad Credentials”.

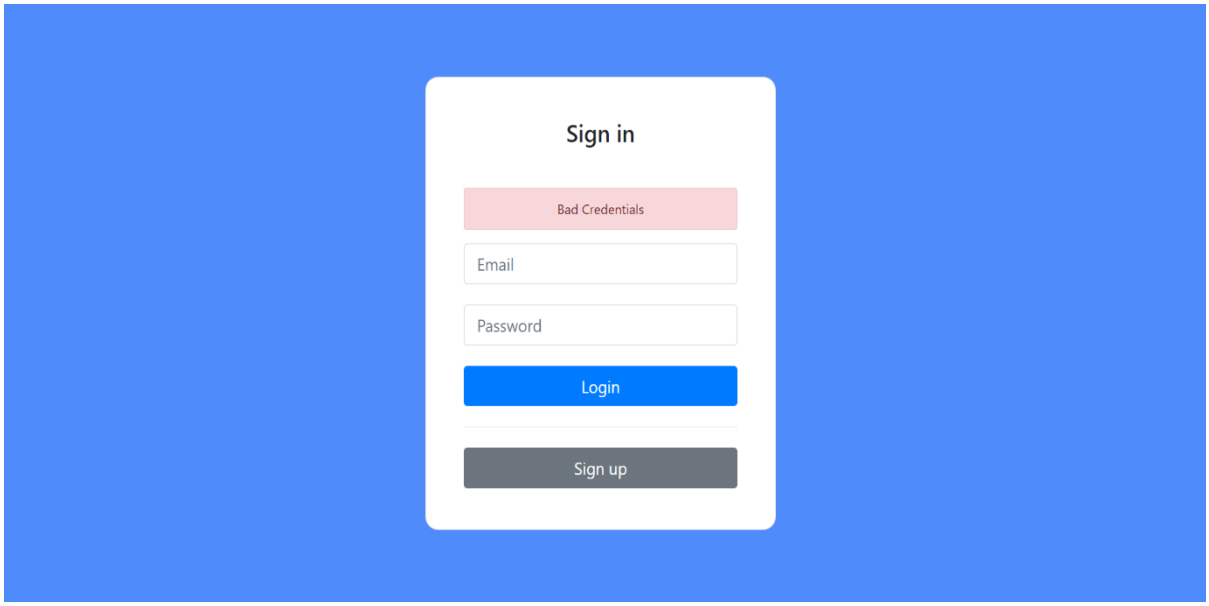


Figure 8 Login page - Bad credentials

Με το που εισάγει τα στοιχεία του και είναι έγκυρα, ο χρήστης μεταφέρεται στην σελίδα του Dashboard όπως φαίνεται παρακάτω.

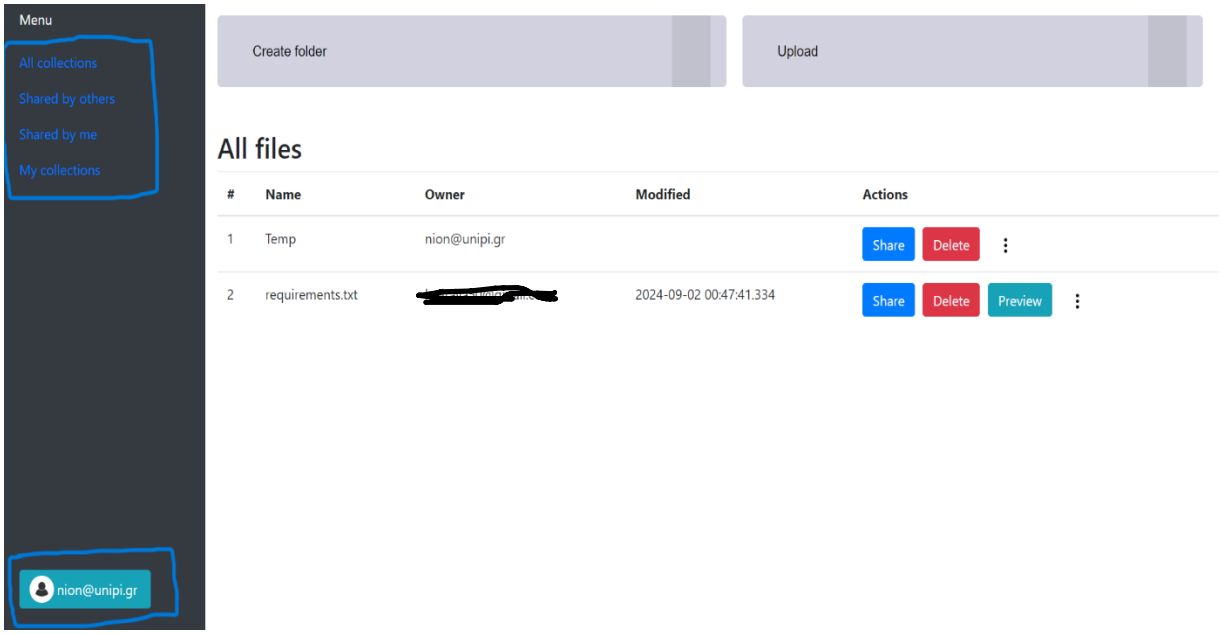


Figure 9 Dashboard

Στην σελίδα παρατηρούμε πως κάτω αριστερά εμφανίζεται ένα κουμπί με το email του χρήστη, ενώ πάνω αριστερά εμφανίζονται οι επιλογές με τις οποίες μπορεί να διαλέξει να αναζητήσει τα αρχεία ή τους φακελούς που έχει πρόσβαση. Πιο συγκεκριμένα πατώντας το link All collections, εμφανίζονται όλα τα αρχεία/φάκελοι που ο χρήστης έχει πρόσβαση. Πατώντας το link Shared by others, εμφανίζονται μόνο τα αρχεία/ φάκελοι τα οποία έχουν διαμοιραστει στον χρήστη. Αν πατήσει το link Shared by me εμφανίζονται μόνον τα αρχεία/φάκελοι τα οποία έχει διαμοιράσει ο χρήστης σε τρίτους. Τέλος έαν πατήσει το link My collections θα εμφανισθούν μόνο τα αρχεία ή οι φάκελοι που ανήκουν στον χρήστη. Παρακάτω παρουσιάζονται screenshots με τις παραπάνω επιλογές και την αντίστοιχη σειρά.

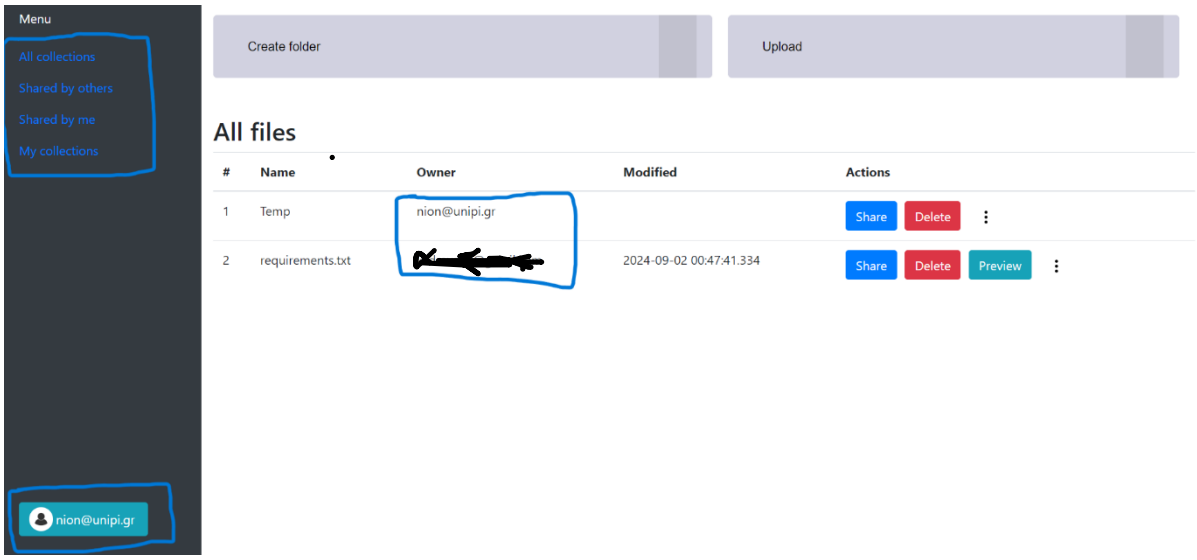


Figure 10 Dashboard

Παρατηρούμε πως έχουμε αρχεία απο διαφορετικους owners. Ο ένας είναι ο χρήστης που συνδέθηκε nion@unipi.gr και ο άλλος είναι ο k@gmail.com. Αυτό σημαίνει πως ο φάκελος ανήκει στον πρώτο χρήστη ενώ το αρχείο requirements.txt έχει διαμοιραστεί στον χρήστη.

Παρακάτω εμφανίζεται μόνο το αρχείο requirements.txt που έχει διαμοιραστεί στον χρήστη.

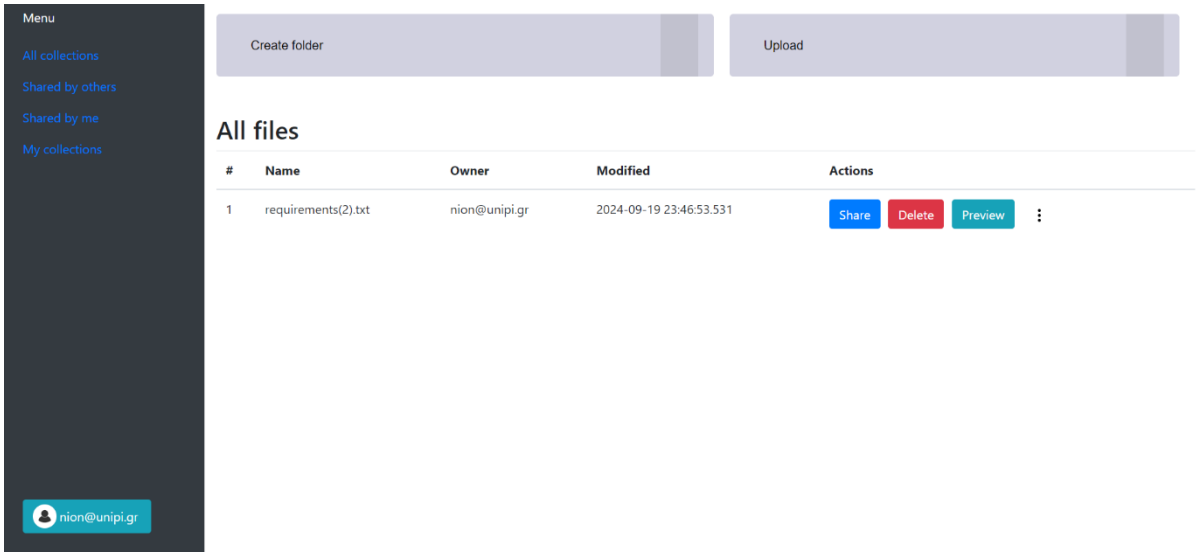


Figure 11 Dashboard - Received resources

Παρακάτω δεν εμφανίζεται κανένας φάκελος ή αρχείο, αφού ο χρήστης δεν έχει διαμοιράσει κάποιο σε τρίτους.

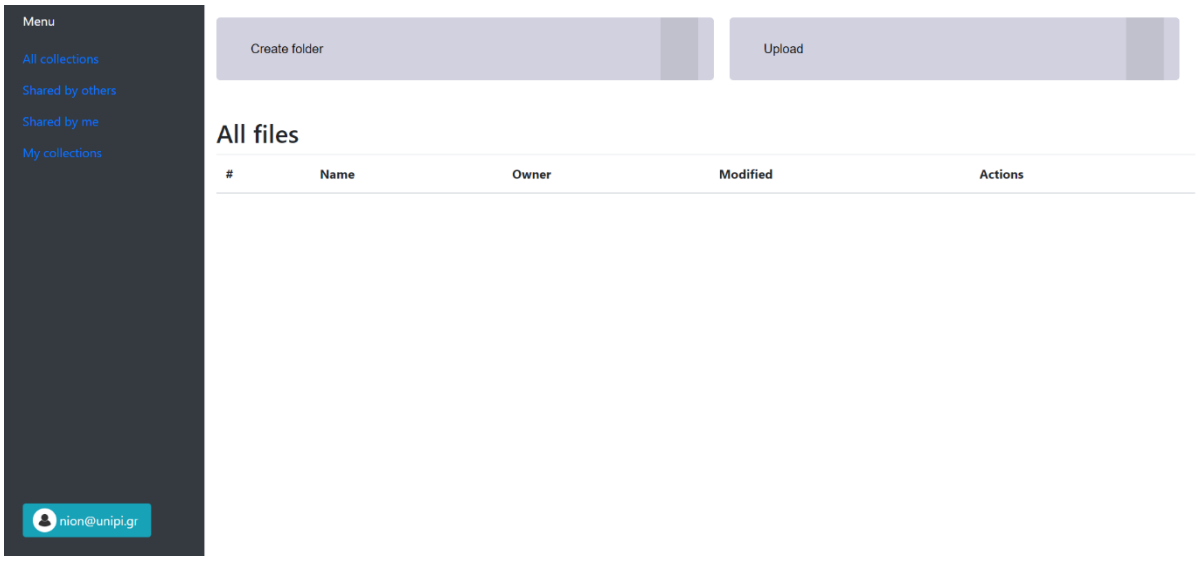


Figure 12 Dashboard - Shared resources

Τέλος εμφανίζονται τα αρχεία/φάκελοι τα οποία τα έχει δημιουργήσει ο χρήστης.

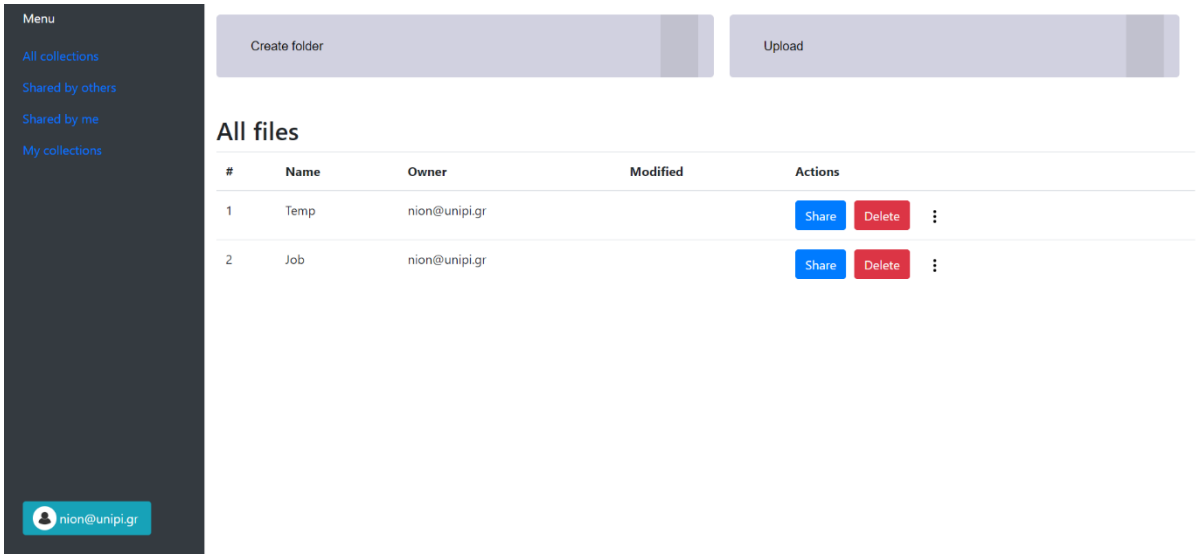


Figure 13 Dashboard - User's resources

Αν πατήσει το κουμπί Create folder, ο χρήστης μπορεί να δημιουργήσει καινούριο φάκελο εισάγοντας το όνομα που επιθυμεί, να το ονομάσει, μέσω του παραθύρου που εμφανίζεται όπως φαίνεται και στις παρακάτω εικόνες.

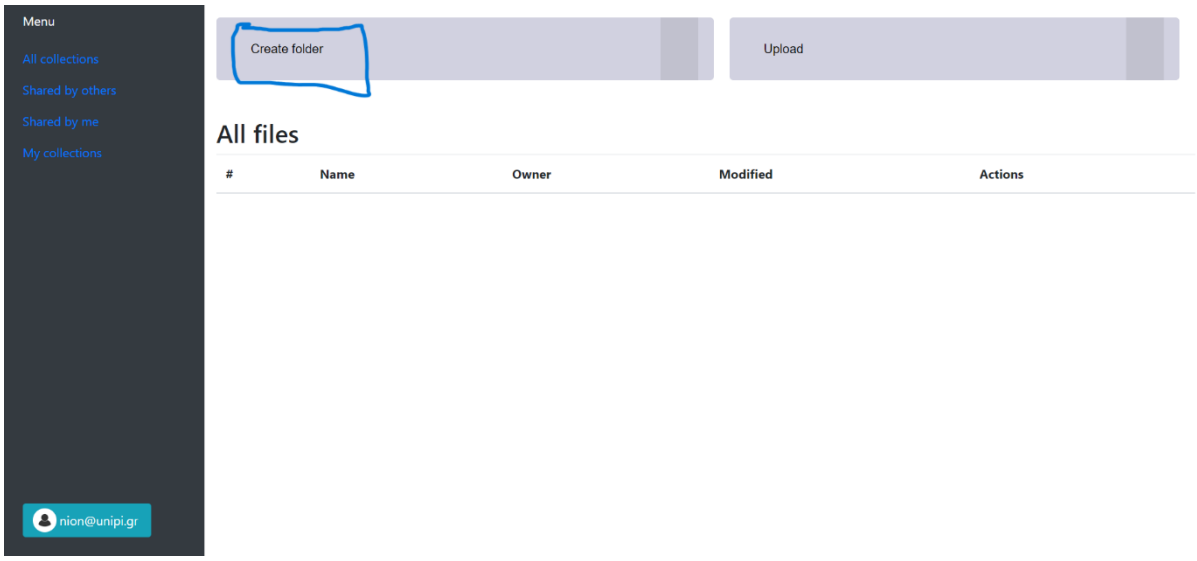


Figure 14 Dashboard - Create folder

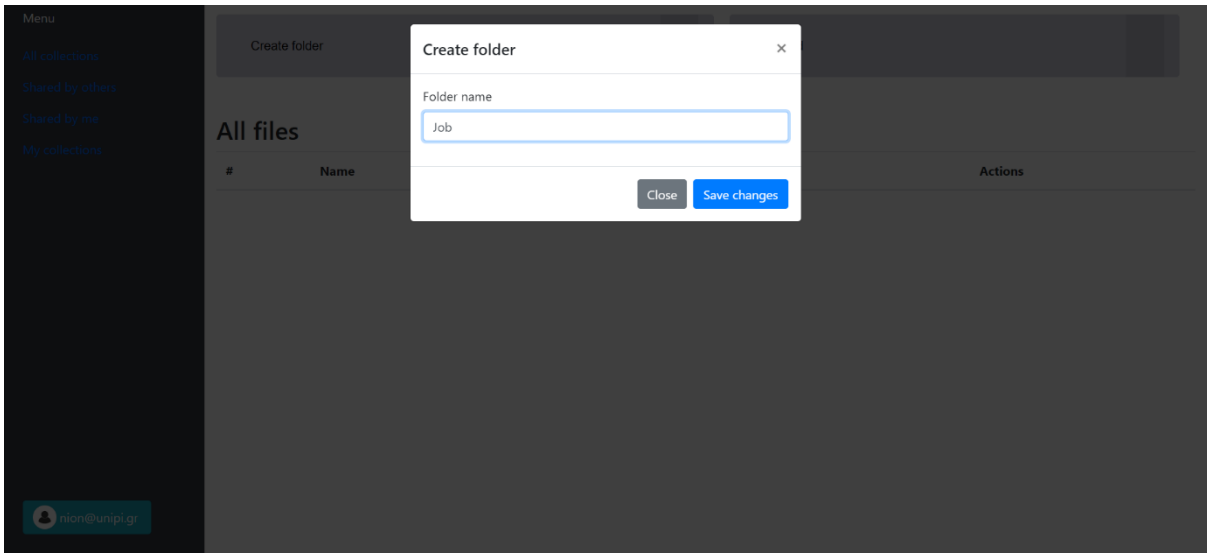


Figure 15 Dashboard - Create folder

Το αποτέλεσμα φαίνεται στην παρακάτω εικόνα.

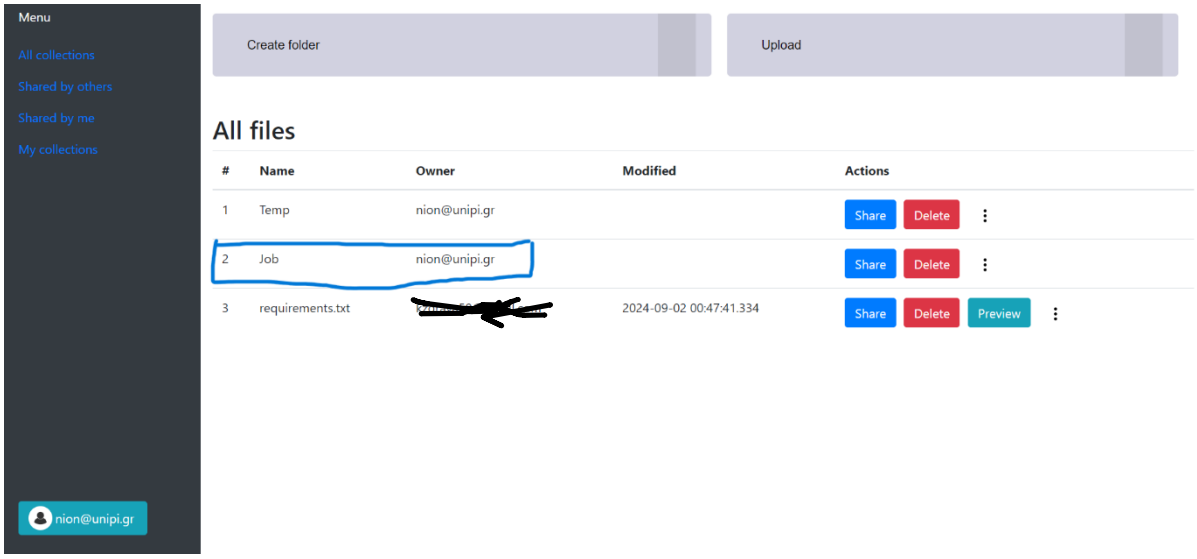


Figure 16 Dashboard

Πατώντας το κουμπί Upload ο χρήστης μπορεί να ανεβάσει μόνο αρχεία που έχουν μια απο τις εξής καταλήξεις: pdf, csv, txt, jpg, png, docx, jpeg. Το παράθυρο που θα εμφανισθεί δηλαδη θα περιέχει μονο τα αρχεία του υπολογιστή που έχουν τις παραπάνω καταλήξεις.

Αν πατήσει το κουμπί που βρίσκεται κάτω αριστερά, θα εμφανισθούν δύο επιλογές Sign out και Requests. Πατώντας πάνω στο Sign out ο χρήστης αποσυνδέεται απο την εφαρμογή, καταστρέφεται το session του και μεταφέρεται στην σελίδα του Login.

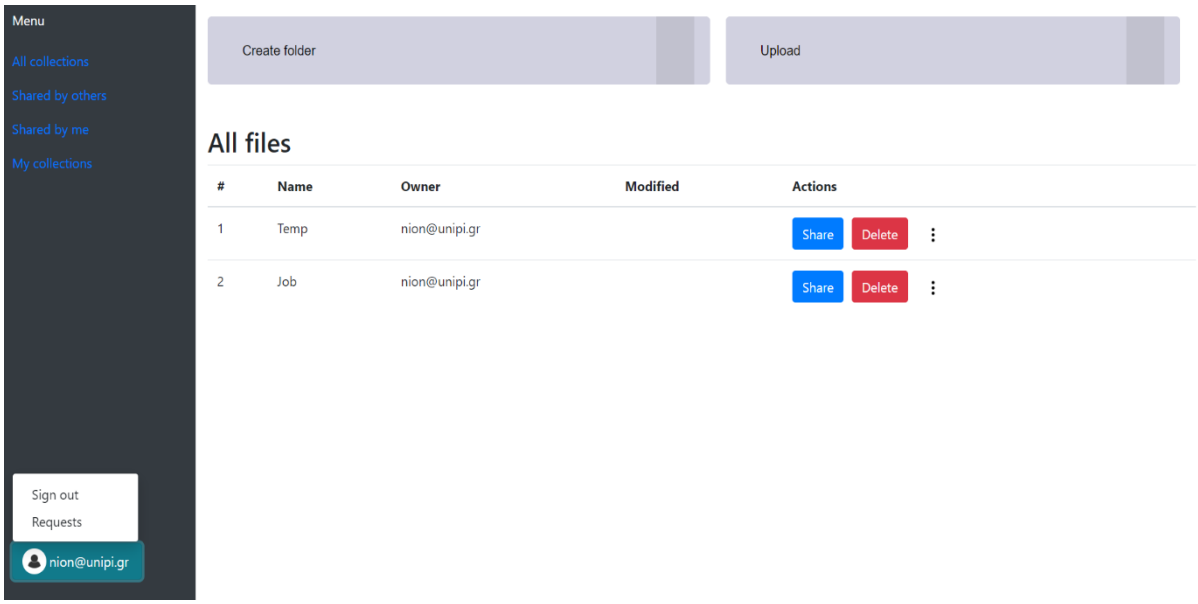


Figure 17 Dashboard

Αν πατήσει το κουμπί Requests, εμφανίζεται ένα παράθυρο με τα εκρεμμή requests. Δηλαδή αιτήματα απο άλλους χρήστες όπου ζητούν απο τον κάτοχο του resource να κάνει approve το διαμοίρασμα του αρχείου/ φακέλου σε τρίτους χρήστες. Στην παράκατω εικόνα δεν υπάρχει πρός το παρόν κάποιο τέτοιο μήνυμα οπότε το παράθυρο είναι κενό.

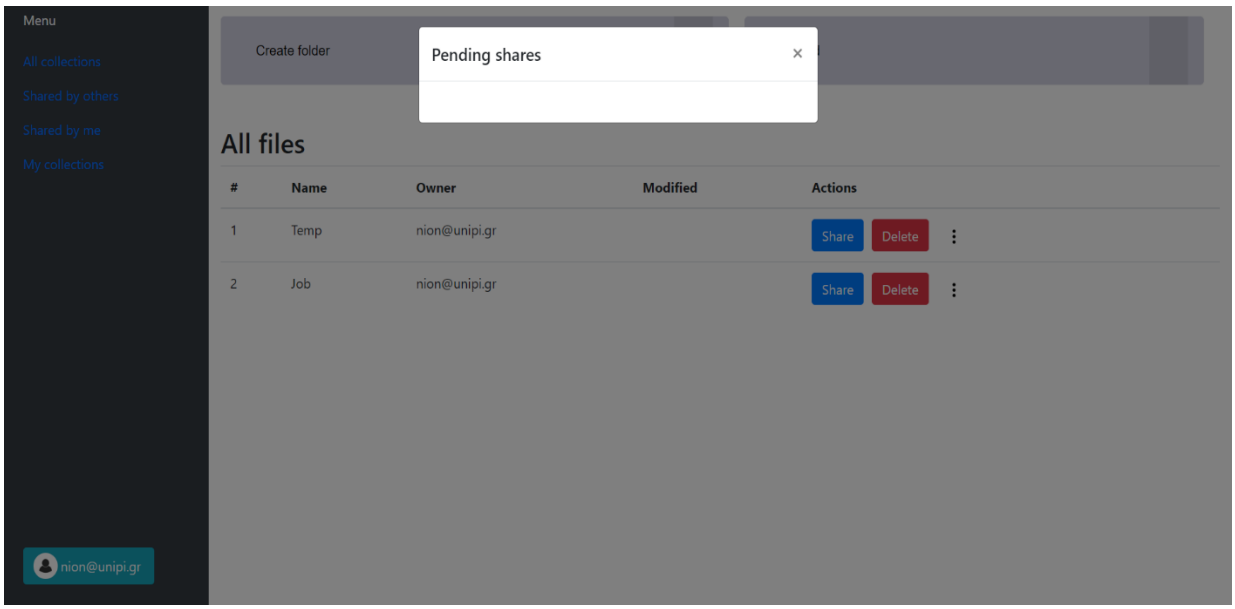


Figure 18 Dashboard - Pending requests

Πατώντας το κουμπί Delete δίπλα απο το αντίστοιχο αρχείο/φάκελο και κάτω απο την στήλη Actions, ο χρήστης μπορεί να το διαγράψει όπως φαίνεται στις παρακάτω εικόνες.

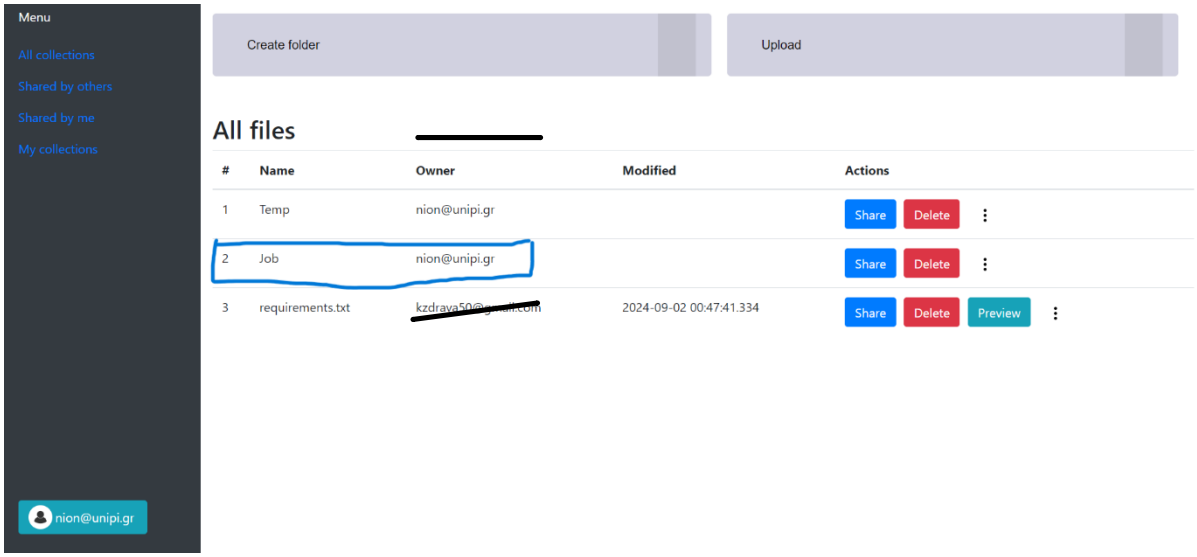
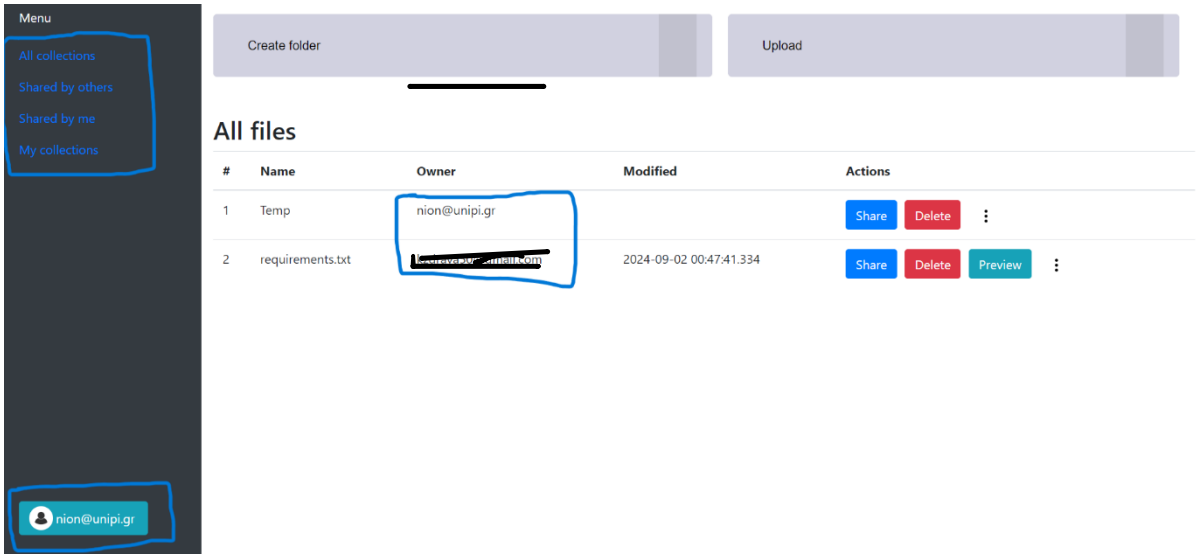


Figure 19 Dashboard



Πατώντας το κουμπί Preview ο χρήστης μπορεί να “δεί” το περιεχόμενο του αρχείου πάνω στην εφαρμογή, χωρίς δηλαδή να χρειαστεί να το κατεβάσει. Παρακάτω υπάρχει screenshot για το αρχείο με όνομα requirements.txt.

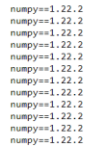


Figure 20 Dashboard - Page view

Αν πατήσει το κουμπί Share ο χρήστης, θα εμφανισθεί ένα παράθυρο με ένα πεδίο εισαγωγής και τέσσερα κουμπιά. Στο πεδίο εισαγώγης ο χρήστης εισάγει το email του χρήστη στον οποίο επιθυμεί να διαμοιράσει το αρχείο. Σε ότι αφορά τα κουμπία υπάρχουν τα εξής: Copy link, Share with all, Share, Add new user.

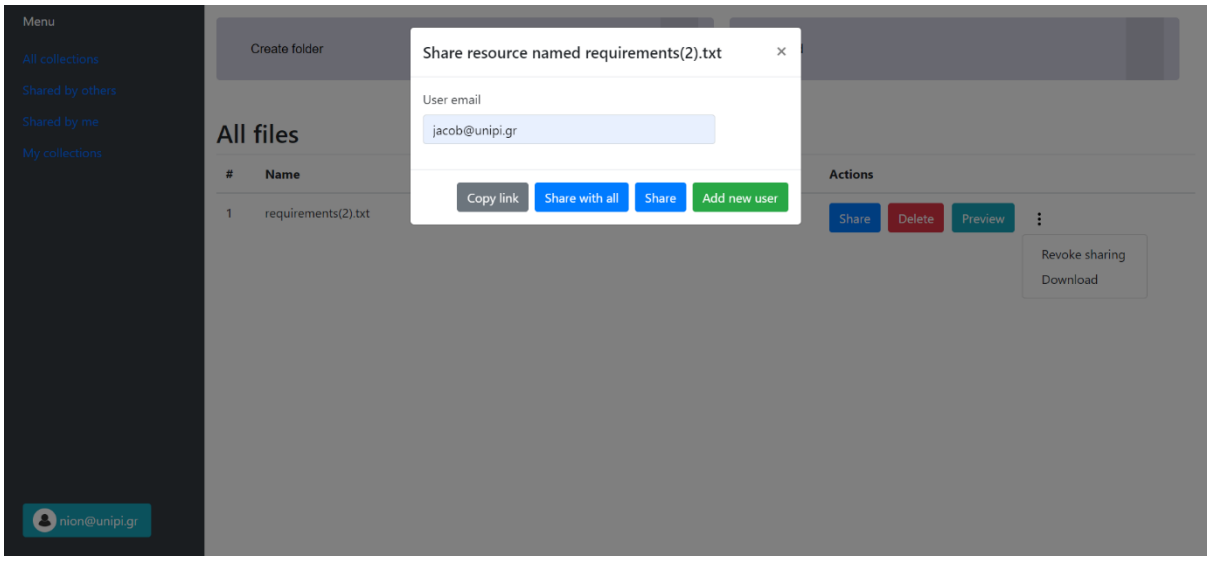


Figure 21 Dashboard - Share resource modal

Πατώντας το κουμπί Copy link ο χρήστης κάνει copy στο click board το link του αρχείου/φακέλου, το οποίο ύστερα μπορεί να το στείλει σε όποιον χρήστη επιθυμεί. Πατώντας το κουμπί Share with all ο χρήστης διαμοιράζει το αρχείο/φάκελο σε όλους. Με το κουμπί Share η διαμοίραση γίνεται μόνο στα emails που έχουν εισαχθεί στη φόρμα. Τέλος πατώντας το κουμπί Add new user προστίθεται νέο πεδίο εισαγώγης email στο παράθυρο, όπου δίνει την δυνατότητα στον χρήστης να εισάγει και άλλα άτομα στα οποία θέλει να διαμοιράσει το αρχείο/φάκελο. Σημειώνεται πως για να μπορέσουν άλλα άτομα να έχουν πρόσβαση στο αρχείο/φάκελο μέσω του link θα πρέπει ο χρήστης πρώτα να έχει πατήσει είτε το κουμπί Share with all είτε να έχει συμπληρώσει το email των χρηστών στους οποίους θέλει να το διαμοιράσει και έπειτα να πατήσει το κουμπί Share. Διαφορετικά θα εμφανισθεί το παρακάτω μήνυμα στην οθόνη:

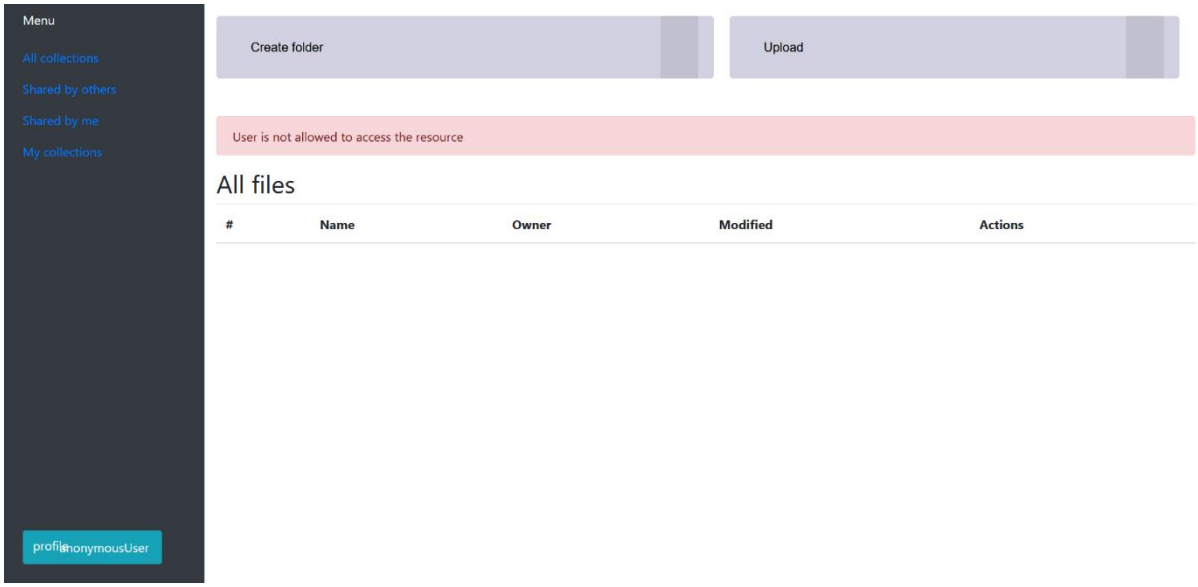


Figure 22 Dashboard - Resource forbidden

Αν πατήσει ο χρήστης τις τρείς τελίτσες δίπλα απο κάποιο αρχείο/φάκελο και ύστερα την επιλογή Revoke sharing, τότε εμφανίζεται ένα παράθυρο με όλους τους χρήστες που έχει διαμοιράσει το αρχείο/φάκελο και που ο κάτοχος του έχει κάνει approve.

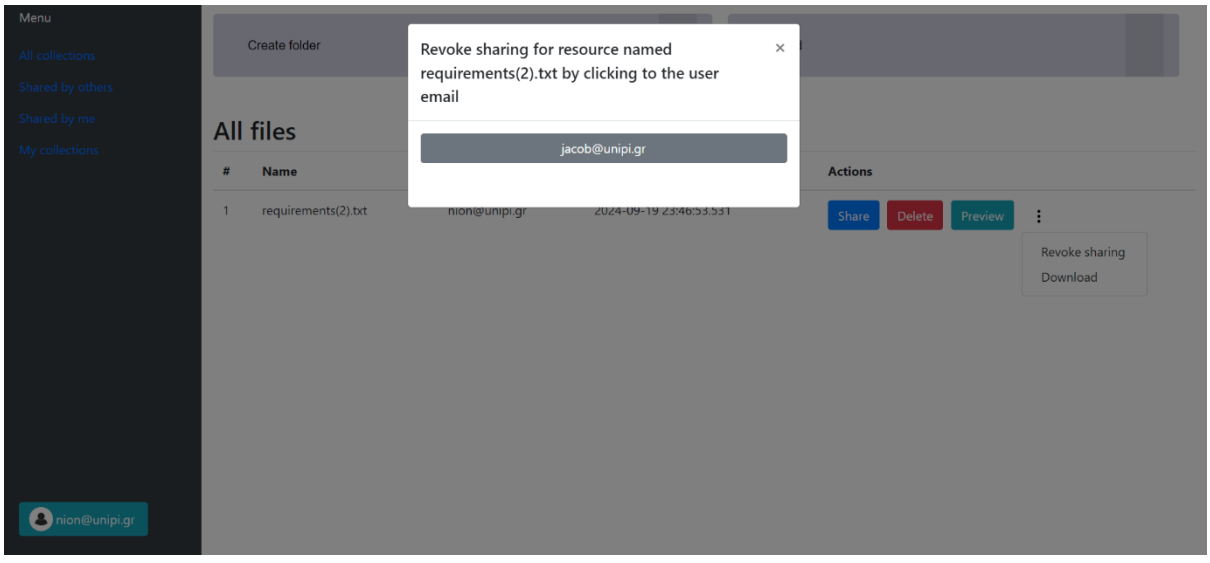


Figure 23 Dashboard - Revoke sharing

Πατώντας λοιπον στο όνομα jacob@unipi.gr "παίρνουμε" πίσω την πρόσβαση που είχαμε δώσει στον χρήστη, συνεπώς δεν θα μπόρει πλέον να έχει πρόσβαση στο αρχείο/φάκελο.

Τέλος με την επιλογή Download μπορεί να κατεβάσει το αρχείο και τοπικά στον υπολογιστή του.

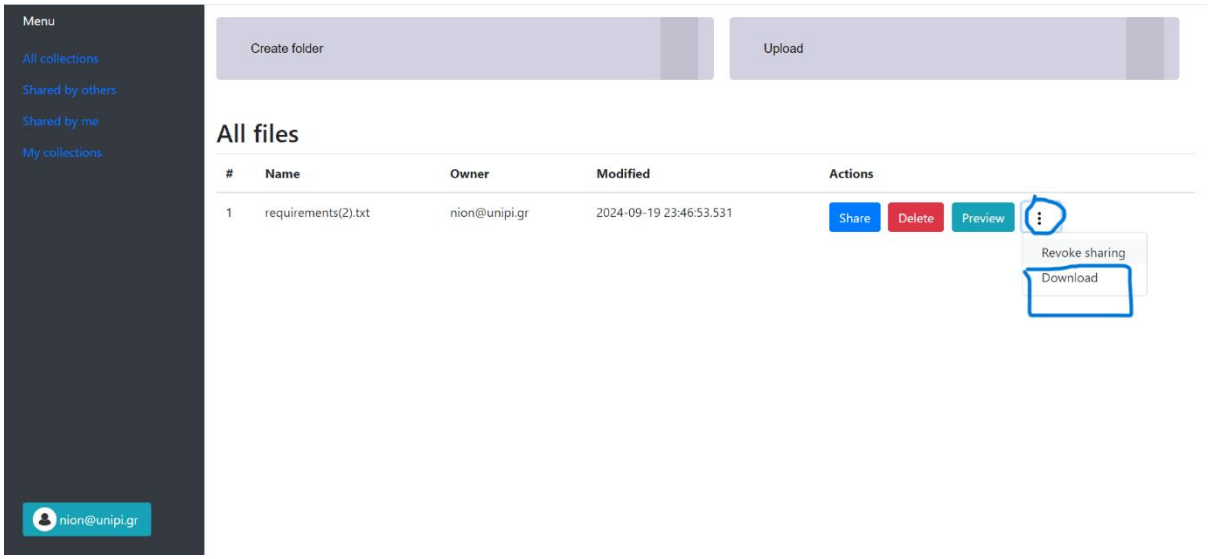


Figure 24 Dashboard - Download action

Κεφάλαιο 4ο

4 Υλοποίηση

Η εφαρμογή υλοποιήθηκε χρησιμοποιώντας ο κύρια γλώσσα προγραμματισμού την Java, το Spring framework, HTML/CSS για την εμφάνιση των σελίδων, την τεχνολογία hibernate για την αναπαράσταση των κλάσεων σε πίνακες στην βάση δεδομένων και το bootsrap framework που έχει μερικά έτοιμα templates όπως πίνακες, κουμπιά, φόρμες κ.α. Για την αποθήκευση των δεδομένων χρησιμοποίηθηκε η βασή δεδομένων MySQL.
Πιο συγκεκριμένα, η εγαρμογή είναι μια web εφαρμογή η οποία εκτελείται σε κάποιον server ο οποίος δέχεται http requests εκτελεί κάποιες λειτουργίες και επιστρέφει ένα response. Ο server που χρησιμοποιήθηκε για τον σχεδιασμό και το testing της εφαρμογής είναι ο Tomcat embedded server του Spring boot. Απο εκεί και πέρα, η εφαρμογή μπορεί εύκολα να "τρέξει" και σε οποιονδήποτε άλλον server μέσα στον tomcat με κάποιες μικροαλλαγές στα configurations.
Η εφαρμογή υλοποιεί την αρχιτεκτονική MVC (Model View Controller) που είναι μέρος του Spring Web. Ως models αναπαριστουμε τις κλάσεις που θέλουμε να αποθηκεύσουμε στην βάση δεδομένων με την μορφή πινάκων, οι Controllers είναι κλάσεις υπεύθυνες για να αποδέχονται τα αιτήματα του χρήστη και να τα επεξεργάζονται σύμφωνα με το business logic και τα Views είναι το user interface, για το οποίο χρησιμοποιήθηκε η HTML/CSS μαζί με την Thymeleaf, η οποία είναι μια τεχνολογία που μπορεί να φορτώνει δυναμικά τα δεδομένα του χρήστη (π.χ ονόματα αρχείων, ονόματα χρηστών κ.α) σε στατικές σελίδες.
Παρακάτω θα αναλυθούν περιγραφικά οι σημαντικότεροι Controllers, οι ανάγκες που καλύπτουν καθώς και components με τα οποία "επικοινωνούν" για να φέρουν εις πέρας τα αιτήματα των χρηστών. Επίσης θα παρουσιασθούν οι πίνακες που υπάρχουν για να αποθηκέυουν τα δεδομένα των χρηστών καθώς και με ποιόν τρόπο αυτά τα δεδομένα παρουσιάζονται στον χρήστη. Τέλος αναλύεται ο τρόπος που έχει σχεδιαστεί η ασφάλεια του συστήματος, για να αποτρέπει κακόβουλους χρήστες από το να έχουν πρόσβαση σε αρχεία που δεν είναι προσβάσιμα.

4.1 Controllers

Οι Controllers είναι το κυρίως μέρος της εφαρμογής. Είναι υπεύθυνοι να δέχονται τα αιτήματα των χρηστών, να τα επεξεργάζονται και να επιστρέφουν μιά απάντηση στον χρήστη. Χρησιμοποιούν το πρωτόκολλο HTTP και τις μεθόδος POST και GET μεταξύ άλλων για να δημιουργήσουν χρήστες ή αρχεία/φακέλους και να επιστρέψουν στον χρήστη τα αρχεία ή τους φακέλους που επιθυμεί.

Οι βασικότεροι Controllers της εφαρμογής είναι οι εξής:

- AuthenticationsController
- DataCollectionsController
- SharingController

4.1.1 AuthenticationController

Είναι υπεύθυνος για την αυθεντικοποίηση των χρηστών. Πιο συγκεκριμένα, όταν ο χρήστης προσπαθεί να συνδεθεί στην εφαρμογή ή να κάνει εγγραφή, μέσω της φόρμας, στέλνεται ένα Request object με τα στοιχεία του στον controller. Ο controller ύστερα μεταβιβάζει την λογική της αυθεντικοποιήσης στον AuthenticationService ο οποίος με την σειρά του ελέγχει εάν τα στοιχεία είναι έγκυρα. Για την εγγραφή του χρήστη γίνεται πρώτα έλεγχος εάν το username που εισήγαγε ο χρήστης υπάρχει ήδη στην βάση δεδομένων. Αν ναι, επιστρέφει κατάλληλο μήνυμα και απορρίπτει την εγγραφή διαφορετικά με χρήστη του UserRepository αποθηκέυει την εγγραφή στην βάση. Σημειώνεται πως ο κωδικός αποθηκεέυται σε hashed μορφή και όχι σε plain text.

Σε ότι αφορά την αυθεντικοποίηση του χρήστη, ο Controller δέχεται το request του χρήστη με το username & password που εισήγαγε στην φόρμα και τα μεταβιβάζει στον AuthenticationService ο οποίος με την σειρά του χρησιμοποιεί τον AuthenticationManager για να ελέγξει εάν υπάρχει τέτοιος χρήστης στην βάση. Αν εντοπιστεί με επιτυχία τέτοια εγγραφή, δηλαδή εαν ο χρήστης αυθεντικοποιηθεί επιτυχώς τότε το Spring Security αποθηκέυει την κατάσταση του χρήστη στο SecurityContextHolder.

```
    private final AuthenticationService service;

    no usages  klaudiozdrava
    @GetMapping("/login")
    public String getLoginPage() { return "login"; }

    no usages  klaudiozdrava
    @GetMapping("/register")
    public String getRegisterPage() { return "register"; }

    no usages  klaudiozdrava
    @PostMapping("/register")
    public String register(@ModelAttribute RegisterRequest registerRequest) {
        ResponseEntity<String> response = service.register(registerRequest);
        if(response.getStatusCode().is2xxSuccessful()) {
            return "redirect:/login";
        }else {
            return "redirect:/register?error";
        }
    }

    no usages  klaudiozdrava
    @PostMapping("/authenticate")
    public String authenticate(@ModelAttribute AuthenticateRequest authenticateRequest) {
        ResponseEntity<String> response = service.authenticate(authenticateRequest);
        if(response.getStatusCode().is2xxSuccessful()) {
            return "redirect:/login?success";
        }else {
            return "redirect:/login";
        }
    }
```

Figure 25 Class AuthenticationController

Τα annotations @GetMapping επιστρέφουν τις HTML σελίδες για να συνδεθεί ή να κάνει εγγραφή ο χρήστης και τα annotations @PostMapping περιέχουν τον κωδικά που εκετλείται μόλις ο χρήστης συμπληρώσει τα στοιχεία της φόρμας και πατήσει το κουμπί Submit.

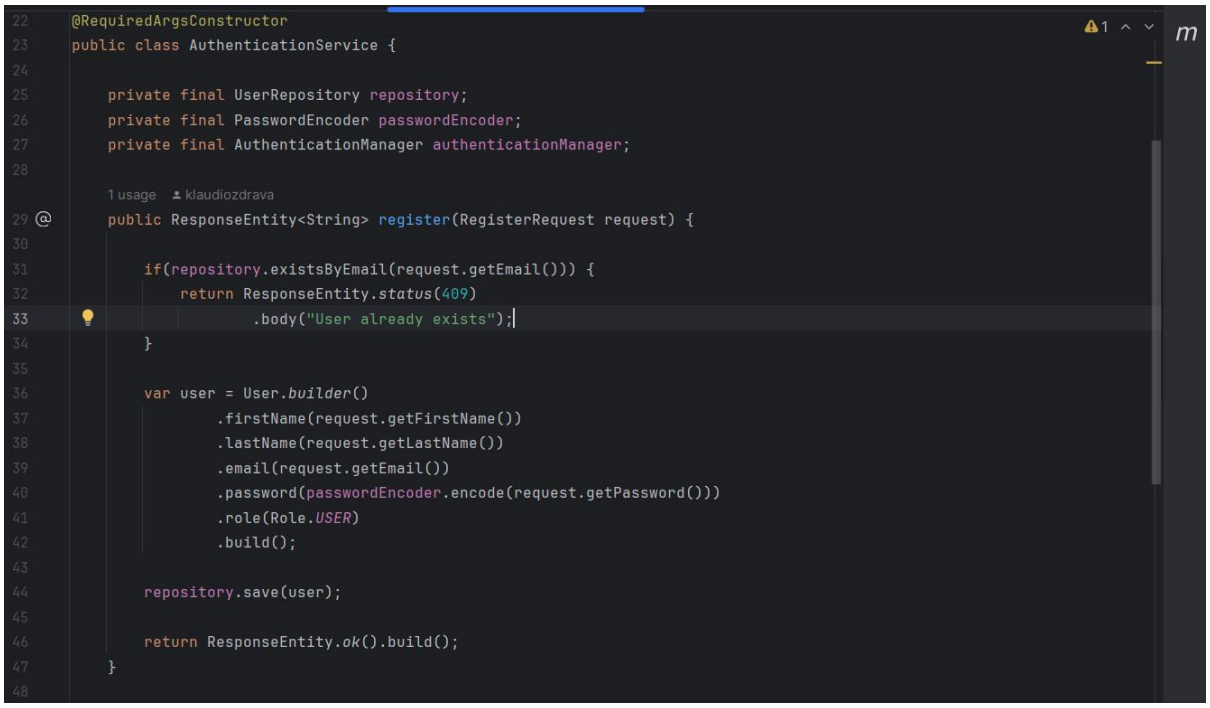


Figure 26 Class AuthenticationService

Figure 27 AuthenticationService - authenticate

4.1.2 DataCollectionsController

Ο σγκεκριμένος controller αποτελεί την core λογική της εφαρμογής. Είναι το backend του Dashboards στην ουσία και είναι υπεύθυνος ωστε να επεξεργάζεται requests που έχουν να κάνουν με της επεξεργασία ή την εμφάνιση των αρχείων/ φακέλων. Περιέχει τα κατάλληλα endpoints για να διαγράψουμε κάποιο αρχείο/φάκελο, να εμφανίσουμε το περιεχόμενο του κ.α. Χρησιμοποιεί τέσσερα διαφορετικα services για να επεξεργαστει αιτήματα που αφορούν φακέλους, αρχεία, γενικά resources και διαμοιρασμούς. Τα Services είναι τα εξής: FolderService, FileService, ResourceService και ShareService. Ο FolderService είναι υπέυθυνος να διαχερίζεται ενέργειες του χρήστη που αφορούν τους φακέλους όπως την δημιουργία τους και την διαγραφή τους. Ο ResourceService είναι υπεύθυνος να διαχειρίζεται τα resources του χρήστη. Με τον όρο resources εννοούμε, τους φακέλους και τα αρχεία. Καθε resource μπορεί να έχει υποφακέλους και αρχεία που ανήκουν σε αυτό και κάθε φάκελος και αρχείο είναι ένα resource. Τέλος ο ShareService χρησιμοποείται για να αναζητήσει resources που έχουν διαμοιραστεί είτε απο τον χρήστη σε τρίτους χρήστες είτε απο τρίτους στον χρήστη. Παρακάτω παρουσιάζονται μερικές απο τις λειτουργίες του controller και θα γίνει μια περιγραφική επεξήγηση του κώδικα.

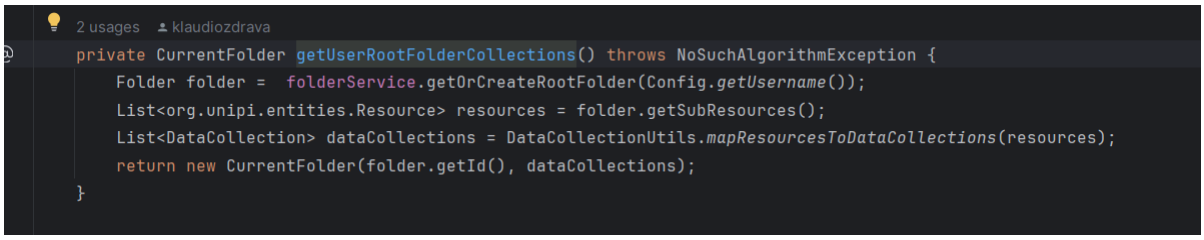


Figure 28 DataCollectionsController - getRootFolder

Η παραπάνω μέθοδος δημιουργεί ένα root folder για τον χρήστη, την πρώτη φορά που συνδέεται στην εφαρμογή, τις υπόλοιπες φορές επιστρέφει τον φάκελο αυτόν και τα resources που βρίσκονται μέσα στον φάκελο. Ο root folder υπάρχει για να μπορέσει να υπάρξει μια δομή στους φακέλους και τα αρχεία, όπως πχ στο λειτουργικό σύστημα Linux.

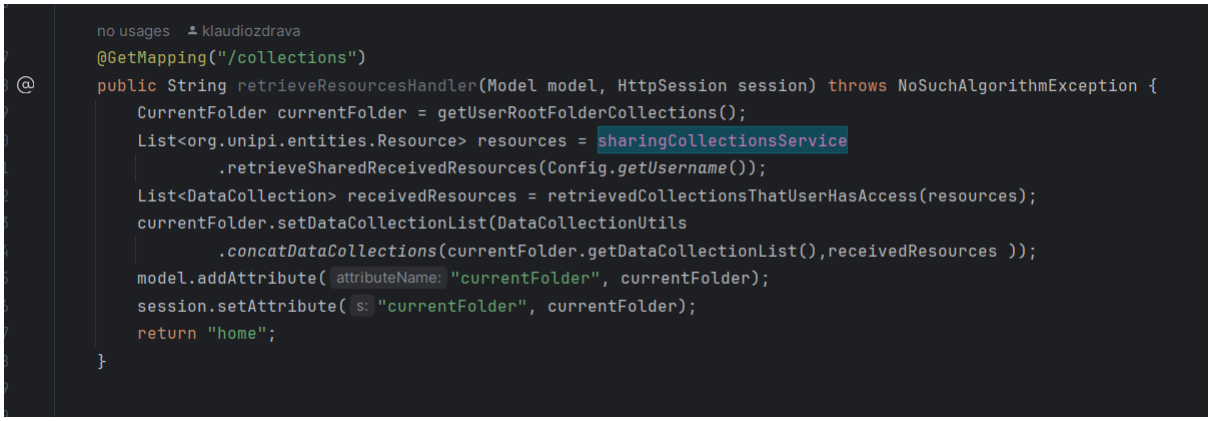


Figure 29 DataCollectionsController – retrieveResourcesHandler

Στην παραπάνω εικόνα βρίσκεται η λογική του endpoint για όλα τα αρχεία/φακέλους που έχει πρόσβαση ο χρήστης. Περιγραφικά, αναζητεί όλα τα αρχεία/φακέλους που του έχουν διαμοιραστεί και όλα τα αρχεία τα οποία ο ίδιος έχει δημιουργήσει. Με βάση όλα αυτά δημιουργεί έναν CurrentFolder object το οποίο κρατάει το state του τρέχοντος φακέλου. Δηλαδή στην ουσία είναι σαν τον φάκελο στον υπολογιστή μας που περιέχει τους υποφακέλους και τα αρχεία εσωτερικά. Με αυτόν τον τρόπο γνωρίζουμε εάν δημιουργήσουμε νέο φάκελο ή ανεβάσουμε κάποιο αρχείο, σε ποιόν φάκελο θα ανήκει. Επίσης μας βοηθάει να εμφανίσουμε την λίστα μόνο των resources του τρέχοντος φακέλου.

```
no usages  klaudiozdrava
@GetMapping("/collections/{id}")
public String getCollectionsOfFolder(@PathVariable String id, Model model, HttpSession session) {
    Folder folder = folderService.getFolderFromDB(id);
    CurrentFolder currentFolder;
    if(Objects.isNull(folder)) {
         currentFolder = new CurrentFolder(id,
                Collections.emptyList());
    }else{
        List<org.unipi.entities.Resource> resources = folder.getSubResources();
        List<DataCollection> dataCollections =  DataCollectionUtils.mapResourcesToDataCollections(resources);
        currentFolder = new CurrentFolder(id,
                dataCollections);
    }
    model.addAttribute( attributeName: "currentFolder", currentFolder);
    session.setAttribute( s: "currentFolder", currentFolder);
    return "home";
}
```

Figure 30 DataCollectionsController - getCollectionsFolder

Στην παραπάνω εικόνα, επιστρέφουμε μόνο τα resources (αρχεία/υποφάκελοι) που ανήκουν σε κάποιο συγκεκριμένο φάκελο με βάση το id του. Το annotation @PathVariable παίρνει την τιμή του {id} απο το endpoint που λειτουργεί σαν placeholder και την εισάγει στην τιμή της μεταβλητής String id. Απο εκεί και έπειτα ο κώδικας με χρήση των services αναζητάει στην βάση δεδομένων τον συγκεκριμένω φάκελο και φορτώνει τα subresources.

```
no usages  klaudiozdrava
@PostMapping("/new-file/{parentFolderId}")
public String handleFileUpload(@RequestParam("files") MultipartFile file, @PathVariable String parentFolderId)  {
    final FileDto fileDto;
    try {
         fileDto = FileDto
                .builder()
                .name(file.getOriginalFilename())
                .fileContent(file.getBytes())
                 .size(file.getSize())
                .build();

        Folder folder = folderService.getFolderFromDB(parentFolderId);
        fileService.updateFile(fileDto, folder);
    }catch (IOException e) {
        log.error("File with name {} is not correct", file.getName());
    } catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
    return "redirect:/data-collections/collections/" + parentFolderId;
}
```

Figure 31 DataCollectionsController - handleFileUpload

Η παραπάνω μέθοδος δέχεται ένα αρχείο σε μορφή MultipartFile απο την σελίδα του Dashboard και το id του φακελου στον οποίο θα ανήκει. Ύστερα αναζητεί τον συγκεκριμένο φάκελο απο την βάση δεδομένων και εισχωρεί καινούρια εγγραφή στην βάση με το καινούριο αρχείο και τον φάκελο που αυτό ανήκει.

```
1 usage  klaudiozdrava
@Transactional
public void updateFile(FileDto fileDto, Folder folder) throws NoSuchAlgorithmException {

    CustomFile file = createFileObject(fileDto, folder);
    List<Chunk> chunks = splitFileToChunks(fileDto.getFileContent());
    file.setChunks(chunks);

    if(fileDoesNotExist(file)) {
        chunks.forEach(chunk -> chunk.setFile(file));
        fileRepository.save(file);
    }
    updateOnlyNewChunks(chunks, file.getId());

}
```

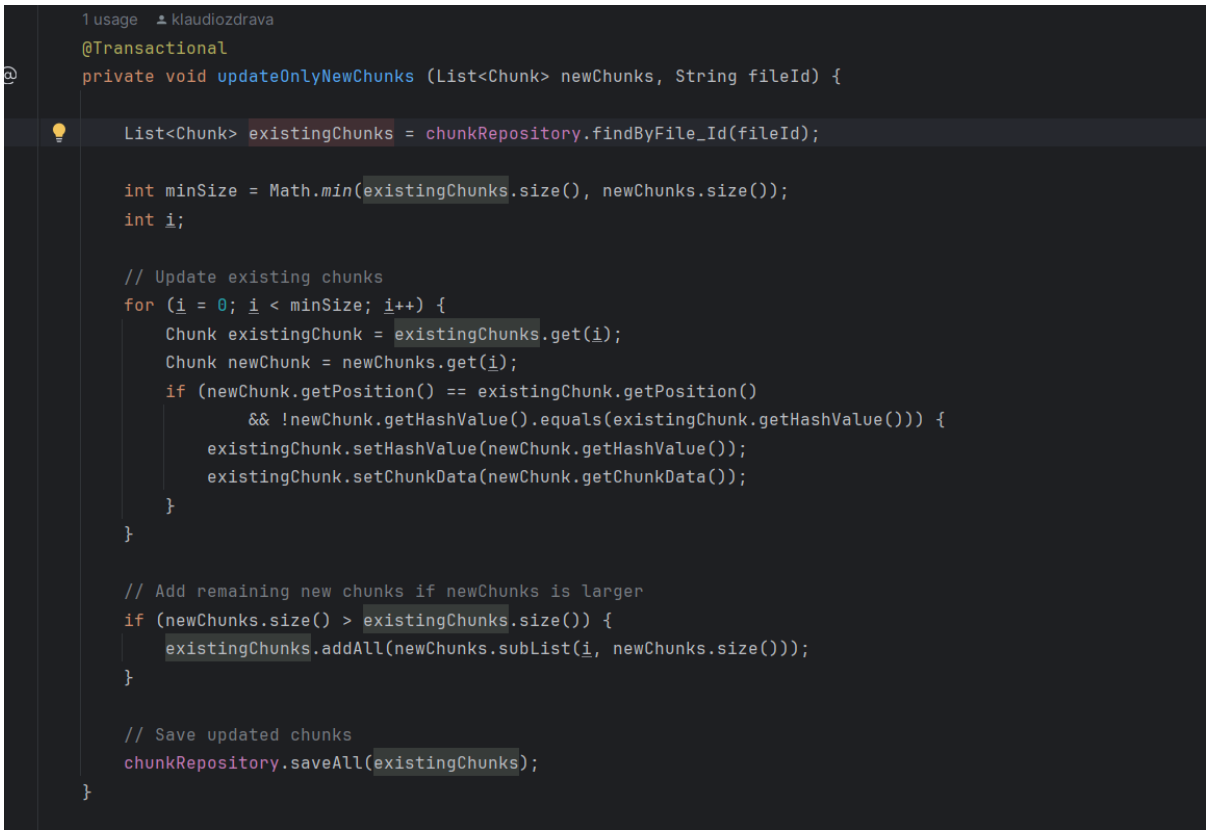
Figure 32 FileService - updateFile

Η μέθοδος updateFile αρχικά κάνει split το αρχείο σε chunks και να ελέγξει έαν υπάρχει ήδη αρχείο με το ίδιο όνομα στον ίδιο φάκελο στην βάση δεδομένων. Αν δεν υπάρχει τέτοιο ανεβάζει όλα τα chunks στην βάση καθώς και το ίδιο το αρχείο. Σε διαφορετική περίπτωση κάνει update μονο εκείνα τα μέρη του αρχείου τα οποία είχαν υποστεί αλλαγές.

```
1 usage   klaudiozdrava
private List<Chunk> splitFileToChunks(byte[] fileBytes)  {

    List<byte[]> listOfBytes = fileChunkSplitter
            .splitFileToChunks(fileBytes, BATCH_SIZE_TO_BYTES);

    List<Chunk> chunks = new ArrayList<>();

    for(int i = 0; i< listOfBytes.size(); i++) {
        byte[] byteArray = listOfBytes.get(i);
        log.info("Split size {}", byteArray.length);
        Chunk chunk ;
        try {
            chunk = Chunk.builder()
                    .chunkData(byteArray)
                    .hashValue(hashGenerator
                            .calculateHash(byteArray, HASH_ALGORITHM ))
                    .position(i)
                    .build();
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
        chunks.add(chunk);
    }
    return chunks;

}
```

Figure 33 FileService - splitFileToChunks

Η μέθοδος σπάει το περιεχόμενο του αρχείο σε μέρη απο bytes συγκεκριμένου μεγέθους (chunks) και ύστερα για κάθε τέτοιο chunk υπολογίζει μια hash τιμή και την αποθηκέυει στην βάση δεδομένων, μαζί με την θέση του chunk στο αρχείο καθώς και το ιδιο το περιεχόμενο του σε μορφή πινάκων απο bytes. Η θέση του chunk μαζί με την hash τιμή μας βοηθάνε να εντοπίσουμε γρήγορα και έυκολα σε ποιά σημεία το αρχείο υπέστη αλλαγές, όποτε να κάνουμε update μόνο εκείνα τα parts.



```
1 usage  klaudiozdrava
@Transactional
private void updateOnlyNewChunks (List<Chunk> newChunks, String fileId) {

    List<Chunk> existingChunks = chunkRepository.findByFile_Id(fileId);

    int minSize = Math.min(existingChunks.size(), newChunks.size());
    int i;

    // Update existing chunks
    for (i = 0; i < minSize; i++) {
        Chunk existingChunk = existingChunks.get(i);
        Chunk newChunk = newChunks.get(i);
        if (newChunk.getPosition() == existingChunk.getPosition()
                && !newChunk.getHashValue().equals(existingChunk.getHashValue())) {
            existingChunk.setHashValue(newChunk.getHashValue());
            existingChunk.setChunkData(newChunk.getChunkData());
        }
    }

    // Add remaining new chunks if newChunks is larger
    if (newChunks.size() > existingChunks.size()) {
        existingChunks.addAll(newChunks.subList(i, newChunks.size()));
    }

    // Save updated chunks
    chunkRepository.saveAll(existingChunks);
}
```

Figure 34 FileService - updateOnlyNewChunks

Η παραπάνω μέθοδος είναι υπεύθυνη να κάνει update τον πίνακα με τα chunks στην βάση δεδομένων. Συγκρίνει για διαφορές μεταξύ του ήδη υπάρχων αρχείου με το αρχείο που εισήγαγε ο χρήστης, εάν υπάρχουν διαφορές κάνει override τα chunks με τα καινούρια. Υπενθυμίζεται πως η σύγρκιση γίνεται μόνον μεταξύ του καινούριου αρχείου και αρχείου στη βάση δεδομένων που έχει το ίδιο όνομα.

4.1.3 SharingController

Ο συγκεκριμένος controller είναι υπεύθυνος να δέχεται και να επεξεργάζεται αιτήματα του χρήστη που αφορούν τον διαμοιρασμό των φακέλων/αρχείων. Πιο συγκεκριμένα, οι λειτουργίες που υλοποιεί είναι μεταξύ άλλων: Διαμοίραση αρχείου/φακέλου, κατάργήση πρόσβασης σε κάποιο resource, αποστολή email στον κάτοχο του resource, αναζήτηση όλων των εκρεμμών resources και αποδοχή ή απόρριψη πρόσβασης άδειας κάποιου resource απο τον κάτοχο.

Παρακάτω θα δοθεί μια μικρή περιγραφή για μερικές απο τις λειτουργίες.

```
no usages  klaudiozdrava
@GetMapping("/accept/{id}/{user}")
public String acceptResourceHandler(@PathVariable String id) {
    sharingCollectionsService.updateSharingResourceStatus(id);
    return "redirect:/data-collections/collections";
}

no usages  klaudiozdrava
@GetMapping("/reject/{id}/{receiver}")
public String rejectResourceHandler(@PathVariable String id, @PathVariable String receiver) {
    sharingCollectionsService.rejectSharingRequestForResource(id, receiver);
    return "redirect:/data-collections/collections";
}
```

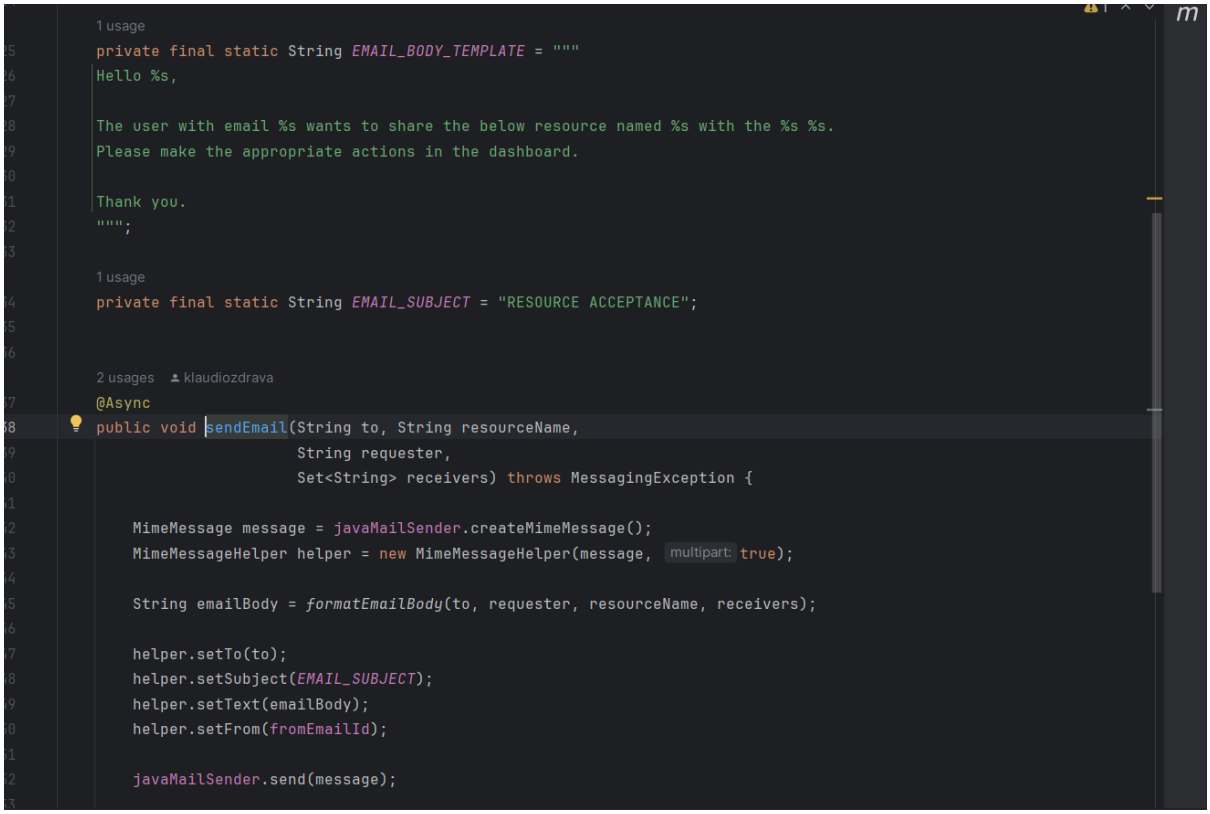
Figure 35 SharingController

Τα παραπάνω δύο endpoints είναι υπέυθυνα για την αποδοχή ή την απόρριψη του αιτήματος διαμοίρασης κάποιου resource, που έλαβε ο κάτοχος του με παραλήπτες τρίτους. Στην ουσία κάνουν είτε update την στήλη status της εγγραφής του αιτήματος του συγκεκριμένου resource απο pending σε accepted είτε διαγράφεται η εγγραφή.

```
no usages  klaudiozdrava
@PostMapping("/share/{id}")
public String shareFileHandler(@RequestParam("email") Set<String> emails,
                               @PathVariable String id, RedirectAttributes redirectAttributes) throws Messag

    Resource resource = resourceService.getResourceById(id);
    if (Objects.isNull(resource)) {
        redirectAttributes.addFlashAttribute( attributeName: "statusCode", HttpStatus.BAD_REQUEST.value());
        redirectAttributes.addFlashAttribute( attributeName: "statusMessage", attributeValue: "Resource with id: " + id + '
    } else if(emails.isEmpty()) {
        redirectAttributes.addFlashAttribute( attributeName: "statusCode", HttpStatus.BAD_REQUEST.value());
        redirectAttributes.addFlashAttribute( attributeName: "statusMessage", attributeValue: "You have to enter at least o
    }
    else {
       ResponseEntity<String> response = sharingCollectionsService
               .shareResource(resource, emails, Config.getUsername());
       if(response.getStatusCode().is2xxSuccessful() && !resource.getOwner().equals(Config.getUsername())) {
           emailService.sendEmail(resource.getOwner(), resource.getName(), Config.getUsername(), emails);
       }
        redirectAttributes.addFlashAttribute( attributeName: "statusCode", response.getStatusCode().value());
        redirectAttributes.addFlashAttribute( attributeName: "statusMessage",response.getBody());
    }
    return "redirect:/data-collections/collections";
}
```

Figure 36 SharingController - shareFileHandler

Η παραπάνω μέθοδος είναι υπεύθυνη για το διαμοιρασμό του resource. Δέχεται σαν input τα emails των χρηστών που αποτελούν τους παραλήπτες και το id του resource. Ύστερα δημιουργεί καινούρια εγγραφή στην βάση δεδομένων με τα συγκεκριμένα στοιχεία. Σε αυτήν την κατάσταση ο τελικός παραλήπτης του αρχείου, δεν μπορεί ακόμα να έχει πρόσβαση στο αρχείο/φάκελο. Η κατάσταση της εγγραφής είναι ακόμη σε εκρεμμότητα και θα πρέπει να γίνει approve απο τον κάτοχο του πρώτα. Στην συνέχεια στέλνεται ενα email στον κάτοχο του αρχείου, ωστε να τον ενημερώσει πώς κάποιος χρήστης θέλει να διαμοιράσει το αρχείο/φάκελο οπότε θα χρειαστεί να το κάνει approve η reject μέσω του Dashboard.



```
1 usage
private final static String EMAIL_BODY_TEMPLATE = """
Hello %s,

The user with email %s wants to share the below resource named %s with the %s %s.
Please make the appropriate actions in the dashboard.

Thank you.
""";

1 usage
private final static String EMAIL_SUBJECT = "RESOURCE ACCEPTANCE";


2 usages  klaudiozdrava
@Async
public void sendEmail(String to, String resourceName,
                      String requester,
                      Set<String> receivers) throws MessagingException {

    MimeMessage message = javaMailSender.createMimeMessage();
    MimeMessageHelper helper = new MimeMessageHelper(message, multipart: true);

    String emailBody = formatEmailBody(to, requester, resourceName, receivers);

    helper.setTo(to);
    helper.setSubject(EMAIL_SUBJECT);
    helper.setText(emailBody);
    helper.setFrom(fromEmailId);

    javaMailSender.send(message);
```

Figure 37 EmailService - sendEmail

```
no usages  klaudiozdrava
@GetMapping("/get-pending-shares")
public ResponseEntity<List<PendingShare>> getPendingShares() {
    List<Share> shares = sharingCollectionsService.findOwnerPendingShares(Config.getUsername());

    List<PendingShare> pendingShares = shares
            .stream().map(share -> new PendingShare(share.getResource().getId(),
                    share.getResource().getName(), share.getResourceReceiver())).toList();
    pendingShares.forEach(pendingShare -> log.info("Share {}", pendingShare.getResourceName()));

    return ResponseEntity.ok().body(pendingShares);
}
```

Figure 38 DataCollectionsController - getPendingshares

Το παραπάνω endpoint, επιστρέφει στον χρήστη τα εκρεμμή αιτήματα. Δηλαδή, τις εγγραφές οι οποίες έχουν status pending.

4.2 Models

Με τον όρο models στο μοντέλο MVC αναπαριστούμε τα δεδομένα της εφαρμογής, όπως τα αρχεία, τους χρήστες, τους φακέλους, τα resources, τα chunks και τέλος τα shares. Στην ούσια με χρήση της Hibernate, και της ORM τεχνικής μπορουμε να μετατρέψουμε κλάσεις σε πίνακες και να τρέξουμε έυκολα queries χωρίς να χρειαστει να γράψουμε SQL κώδικα. Κάθε πεδίο της κλάσης αντιστοιχεί σε μια στήλη του πίνακα της SQL και με χρήση του @Repository annotation επεξεργαζόμαστε τους πίνακες μέσω της Java και του Spring. Παρακάτω θα παρουσιασθούν οι βασίκες κλάσεις και τα πεδία τους.

User

```
5 usages  klaudiozdrava
@Entity
@Data
@Table(name = "users", uniqueConstraints = @UniqueConstraint(columnNames = {"email"}))
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long Id;

    private String firstName;

    private String lastName;

    private String email;

    private String password;

    @Enumerated(EnumType.STRING)
    private Role role;
```

Figure 39 User Entity

Το annotation @Table αντιστοιχεί την συγκεκριμένη κλαση στον πίνακα users και το uniqueConstraints ειναι κάτι σαν το candidate key στην MySQL. Τα πεδίο Id είναι το pk του πίνακα και είναι auto increment. Τα υπόλοιπα πεδία αφορούν το όνομα του χρήστη, το επώνυμο, το email του, τον κωδικό του και τον ρόλο του (User, Admin). Σημειώνεται πως η κλάση User κάνει extend την UserDetails που χρησιμοποιεί ο AuthenticationManager για να αυθεντικοποιήσει τον χρήστη.

4.2.1 Resource

```
11 usages   2 inheritors
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "resources")
@Inheritance(strategy = InheritanceType.JOINED)
public class Resource {
    @Id
    @Column(name = "id")
    protected String id;

    @Column(name = "name")
    protected String name;

    @Column(name = "owner")
    protected String owner;

    @Enumerated(EnumType.STRING)
    protected ResourceEnum resource;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "parent_folder_id", referencedColumnName = "id")
    protected Folder parentFolder;

    @OneToMany(mappedBy = "resource", cascade = CascadeType.ALL, orphanRemoval = true, fetch = FetchType.LAZY)
    private List<Share> shares;

```

Figure 40 Resource Entity

Η κλαση Resource αποτελεί το αρχείο και τον φάκελο, δηλαδή κάθε αρχείο ή φάκελος κάνουν extend την συγκεκριμένη κλαση. Το πεδίο name αποτελει το όνομα του αρχείου, το πεδίο owner τον ιδιοκτήτη, το πεδίο resource τον τύπο του resource δηλαδή αν είναι αρχείο ή φάκελος και το πεδίο parentFolder τον φάκελο στον οποίο ανήκει. Το annotation @Inheritance έχει τον ρόλο του να αντιστοιχίσει τις κλάσεις που βρίσκονται σε κάποια ιεραρχία γονέα-πεδιού σε διαφορετικούς πίνακες οι οποιοί συνδέονται με χρήση foreign key.

4.2.2 Folder

Figure 41 Folder Entity

Η κλάση Folder αποθηκέυει στοιχεία για τους φακέλους και το πεδίο subResources τα αρχεία/φακέλους που ανήκουν στον συγκεκριμένο φάκελο.

4.2.3 CustomFile

Figure 42 CustomFile Entity

Η κλάση CustomFile χρησιμοποιείται για να αποθηκέυει τα αρχεία των χρήστων. Το πεδίο lastModified χρησιμοποείται για να αποθηκέυσουμε την ημερομηνία της τελεύταίας έκδοσης του αρχείου. Το πεδίο size χρησιμοποιείται για να αποθηκέυσουμε το μέγεθους του αρχείου σε bytes. Και το πεδίο chunk για τα μέρη του αρχείου.

4.2.4 Chunk

```
10 usages
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Builder
@Table(name = "chunks")
public class Chunk {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "hashed_value")
    private String hashValue;

    @Lob
    @Column(name = "chunk_data",columnDefinition = "LONGBLOB")
    private byte[] chunkData;

    @Column(name = "position")
    private int position;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "parent_file_id", referencedColumnName = "id")
    private CustomFile file;
}

Chunk                                                    12:19  CRLF  UTF-8  4 spaces
```

Figure 43 Chunk Entity

Η κλάση chunk αναπαριστά κάποιο μέρος του αρχείου. Το κάθε αρχείο όπως έχουμε αναφέρει χωρίζεται σε chunks που το καθένα έχει ίσο μέγεθος και αποτελεί ένος μέρος του. Πχ μπορεί να είναι μόνο η πρώτη γραμμή ή μπορεί να είναι οι πρώτες γραμμές του αρχείου. Τα πεδία του είναι το hashedValue που είναι μια hash τιμή η οποία υπολογίζεται απο το περιεχόμενο του αρχείου, το chunkData που αποτελεί έναν πίνακα απο bytes και αποθηκέυει το ίδιο το περιεχόμενο του chunk. Τέλος το position είναι η θέση του chunk στο αρχείο. Αρχίζοντας από τον δείκτη 0.

4.2.5 Share

```java
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "shares")
public class Share {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Column(name = "resource_sharer")
    private String resourceSharer;

    @Column(name = "resource_receiver")
    private String resourceReceiver;

    @Enumerated(EnumType.STRING)
    private Status status;


    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "resource_id", referencedColumnName = "id")
    private Resource resource;
}
```

Figure 44 Share Entity

Παραπάνω έχουμε την κλάση Share που αντιπροσωπεύει το διαμοίρασμα των αρχείων/φακέλων. Κάθε αίτηση για διαμοιρασμό αποθηκέυεται ως καινούρια εγγραφή με αρχίκη τιμή status pending. Το πεδίο resourceSharer αποθηκέυει τον χρήστη που αιτήθηκε τον διαμοιρασμό, το πεδίο resourceReceiver τον παραλήπτη και το status παίρνει τιμές PENDING (εκρεμμή η αποδοχή απο τον κάτοχο του αρχείου/φακέλου) και ACCEPTED (το αίτημα έγινε approved).

4.3 Security

Για την αυθεντικοποίηση του χρήστη και τον περιορισμό πρόσβασης του σε resources που δεν του επιτρέπεται, καθώς και γενικότερα την ασφάλεια του συστήματος χρησιμοποιήθηκε το Spring Security. Πιο συγκεκριμένα, τα objects που είναι υπεύθυνα για την αυθεντικοποίηση του χρήστη είναι τα **UserDetailsService**, **AuthenticationProvider**, **AuthenticationManager**, **PasswordEncoder** και για την πρόσβαση στην εφαρμογή το **SecurityFilterChain**. Παρακάτω αναλύεται ο σκοπός του κάθε object καθώς και πώς δουλεύει.

4.3.1 UserDetailsService

Σκοπός: Ορίζει μια υπηρεσία που είναι υπεύθυνη για τη φόρτωση δεδομένων που αφορούν συγκεκριμένους χρήστες (όπως το email και ο κωδικός πρόσβασης) από τη βάση δεδομένων.
Πώς λειτουργεί: Ανακτά έναν χρήστη με βάση το email του από το αποθετήριο. Εάν ο χρήστης δεν βρεθεί, εκπέμπεται μια εξαίρεση UsernameNotFoundException. Αυτή η υπηρεσία θα χρησιμοποιηθεί αργότερα για την αυθεντικοποίηση των χρηστών.

4.3.2 AuthenticationProvider

Σκοπός: Αυτό το φασόλι είναι υπεύθυνο για τον πραγματικό μηχανισμό ελέγχου ταυτότητας.
Πώς λειτουργεί:

- DaoAuthenticationProvider χρησιμοποιεί την υπηρεσία UserDetailsService για να αναζητήσει πληροφορίες του χρήστη (email και κωδικό πρόσβασης).
- Χρησιμοποιεί επίσης το PasswordEncoder (BCrypt) για να ελέγξει αν ο κωδικός πρόσβασης ταιριάζει με τον αποθηκευμένο (κωδικοποιημένο) κωδικό πρόσβασης.
- Μετά τον έλεγχο ταυτότητας, ο πάροχος μεταβιβάζει τα στοιχεία του χρήστη για περαιτέρω χειρισμό.

4.3.3 AuthenticationManager

Σκοπός: Ο AuthenticationManager είναι μια κεντρική διεπαφή για τη διαχείριση του ελέγχου ταυτότητας στο Spring Security.
Πώς λειτουργεί: Αναθέτει τον έλεγχο ταυτότητας στον AuthenticationProvider. Αυτό το bean επιτρέπει την προσαρμογή και τη διαχείριση της διαδικασίας ελέγχου ταυτότητας.

4.3.4 PasswordEncoder

Σκοπός: Κωδικοποιεί τους κωδικούς πρόσβασης χρησιμοποιώντας τον αλγόριθμο κατακερματισμού BCrypt για να διασφαλίσει την ασφαλή αποθήκευση των κωδικών πρόσβασης στη βάση δεδομένων.
Πώς λειτουργεί: Όταν οι χρήστες εγγράφονται, οι κωδικοί πρόσβασής τους κατακερματίζονται χρησιμοποιώντας τον BCrypt πριν αποθηκευτούν. Κατά τη σύνδεση, ο αποθηκευμένος κατακερματισμένος κωδικός πρόσβασης συγκρίνεται με τον παρεχόμενο κωδικό πρόσβασης, διασφαλίζοντας την ασφάλεια.

4.3.5 SecurityFilterChain

Η αλυσίδα SecurityFilterChain διασφαλίζει ότι ορισμένα endpoints είναι δημόσια και άλλα απαιτούν έλεγχο ταυτότητας. Καθορίζει επίσης τη συμπεριφορά σύνδεσης και αποσύνδεσης.

Συνεπώς για την ταυτοποίηση του χρήστη, αναζητούμε το email, που εισήαγαγε στην φόρμα, αν υπάρχει στην βάση δεδομένων και αν το βρούμε , τοτε με χρήση του AuthenticationManager object ελέγχουμε αν και ο κωδικός είναι έγκυρος.

4.4 Βάση Δεδομένων

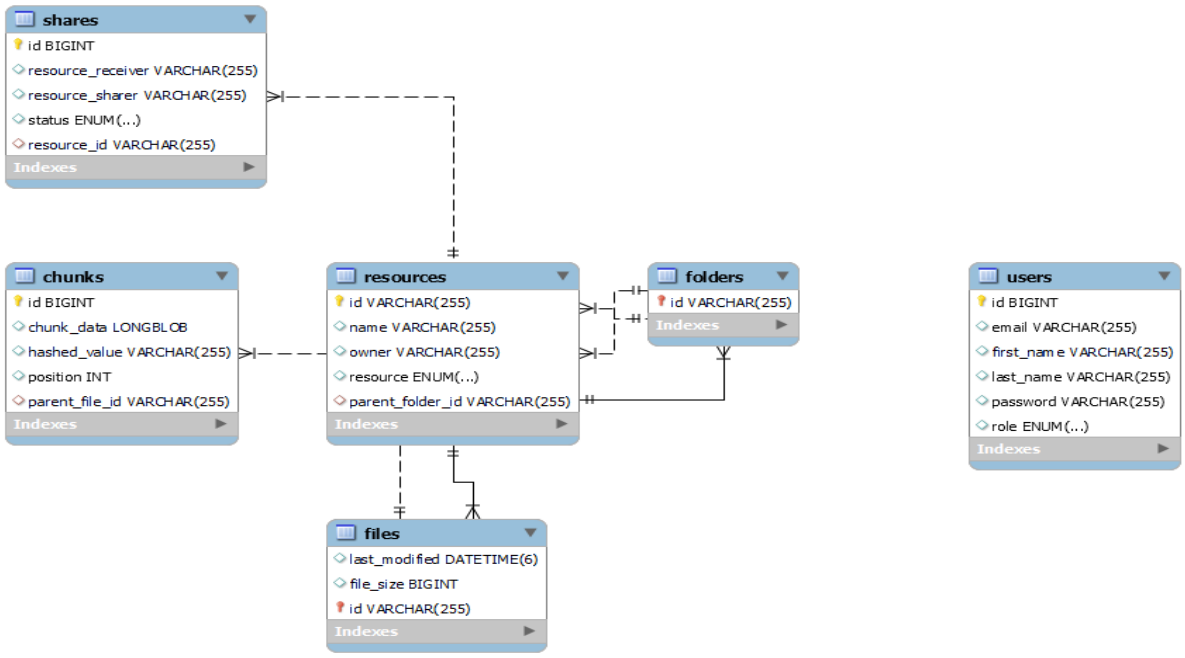


Figure 45 Σχέσεις των πινάκων της βάσης δεδομένων

Κεφάλαιο 5ο

5 Συμπεράσματα

Συνοπτικά η εφαρμογή είναι μια web εφαρμογή που κάνει χρήση του http πρωτοκόλλου για να δέχεται αιτήματα απο τον χρήστη και σκοποό έχει να δώσει την δυνατότητα στους χρήστες της να αποθηκέυουν αρχεία και φακέλους εξωτερικά της προσωπικής τους συσκεύης και να μπορούν να διαμοιράζονται αρχεία έυκολα και γρήγορα.

Ο σχεδιασμός της εφαρμογής έχει υλοποιηθεί με τρόπο τέτοιο ώστε να μπορούν να προστεθούν επιπλέον χαρακτηριστικά και λειτουργίες στο μέλλον. Οι λειτουργίες που θα μπορύσαν να έχουν κάποια χρησιμοτήτα να υπάρχουν είναι οι εξής:

5.1 Ιστορικό αρχείων

Θα μπορύσε να υπάρχει η επιλογή να εμφανίζονται δίαφορες εκδόσεις κάποιου αρχείου ανάλογα με τις αλλαγές που έχουν γίνει. Δηλαδή να κρατάμε κάποιο ιστορικό του αρχείου με όλες τις ανανεωμένες εκδόσεις του περιεχόμένου του αρχείου. Η συγκεκριμένη λειτουργία δεν είναι δύσκολο να υλοποιηθεί, καθώς ο τρόπος που έχει σχεδιαστεί η βάση δεδομένων και η αποθήκευση των αρχείων καθίστα εύκολο την αποθήκευση ιστορικού των αρχείων. Πιο συγκεκριμένα, τα αρχεία όπως αναφέραμε, αποθηκέυονται σε δύο πίνακες με τον έναν να κρατά στοιχεία όπως το όνομα, το μέγεθος κτλπ και τον άλλον να κρατάει το περιεχόμενο σε batches. Συνεπώς δημιουργώντας έναν άλλο πίνακα, μπορούμε να αποθηκέυσουμε τα δίαφορα batches της κάθε θέσης τους στο αρχείο, σύμφωνα με την εκάστοτε έκδοση. Αν για το αρχείο δηλαδή, αλλάζει μόνο ας πόυμε η πρώτη γραμμή του τότε μπορύμε να κρατήσουμε τα υπόλοιπα batches ως έχουν, αλλά για το πρώτο batch να έχουμε σε έναν άλλο πίνακα τις δύο του εκδόσεις μαζί με τις ημερομήνιες των αλλαγών. Με αυτόν τον τρόπο δεν αποθηκέυουμε διπλότυπες εγγραφές οπότε γλιτώνουμε αποθηκεύτικο χώρο. Τέλος για να εμφανίσουμε, για πάράδειγμα, το ιστορικό ενός συγκεκριμένου αρχείου ανάζητουμε τα batches τους και τις θέσεις τους στο αρχείο όπως γίνεται και τώρα και για κάθε εκδόση, με ένα inner join πανω στην θέση του αρχείου επιλέγουμε τα δίαφορα batches και δημιουργούμε διπλότυπα του αρχείου.

5.2 Desktop εφαρμογή

Θα ήταν επίσης χρήσιμο για την εφαρμογή, να υπάρχει ένα desktop application το οποίο θα κάνει track τα αρχεία του χρήστη τοπικά στον υπολογιστή και αν εντοπίσει κάποια αλλαγή σε κάποιο αρχείο τοπικά τότε κάνει update και το αρχείο στον server. Το ίδιο θα συμβεί και έαν υπάρχει αλλαγή για το συγκεκριμένο αρχείο στο web application, τότε θα γίνεται και update το αντίστοιχο στο τοπικό υπολογιστή. Με αυτόν τον τρόπο συγχρονίζουμε τις αλλαγές που γίνονται σε κάποιο αρχείο απο εμάς ή τρίτους. Για να λειτουργεί σωστά η εφαρμογή ο χρήστης διαλέγει τον φάκελο που θέλει να κάνει track και γίνονται update μόνο όσα αρχεία υπάρχουν και στο web server.

5.3 Αναζήτηση αρχείων/φακέλων

Ενδιαφέρουσα προσθήκη θα ήταν και η επιλογη αναζήτησης αρχείου/φακελου μέσω του ονόματος τους. Θα υπήρχε δηλαδή ενα πεδίο εισαγωγής αλφαρηθμητικών όπου ο χρήστης θα γράφει το όνομα του αρχείου/φακέλου και με το που βρεί κάποιο αρχείο/φάκελο που κάνει match θα το εμφανίσει.

5.4 Ταξινόμηση αρχείων/φακέλων

Τέλος, θα μπορούσε να υπάρχει η επιλογή ταξινόμησης αρχείων/φακέλων ώστε να εμφανίζονται με βάση την ημερομηνία δημιουργίας τους ή ανανεώσης τους.

Βιβλιογραφικές Πηγές

1 Dropbox/Google drive system design : https://w84iit.wordpress.com/2020/04/19/system-design-dropbox/

2 Design dropbox : https://astikanand.github.io/techblogs/high-level-system-design/design-dropbox

3 Dropbox system design: https://systemdesignschool.io/problems/dropbox/solution

4 Spring Boot in Action, Craig Walls