Εφαρμογή Επαγγελματικής Δικτύωσης

Χρήστος Καφρίτσας (sdi1800072) Φώτιος Παπαγεωργάκης (sdi1800145)

Contents

Εισαγωγή

Ο στόχος της εργασίας ήταν η ανάπτυξη μίας εφαρμογής επαγγελματικής δικτύωσης στην οποία οι χρήστες θα έχουν πρόσβαση μέσω ενός browser. Μέσα από αυτή την παρουσίαση θα δούμε κάποιες λεπτομέριες και δυσκολίες που αντιμετωπίσαμε με την εργασία όσον αφορά το front end κομμάτι αλλά και το back end. Δεν έχουμε κάνει το bonus κομμάτι της εργασίας.

Για να τρέξει η βάση απαιτείται να υπάρχει μια βάση **betterlinkedin** με **όνομα χρήστη tedi** και **κωδικό tedi2021**. Το hibernate δημιουργεί όλους τους απαραίτητους πίνακες αυτόματα και τρέχει στη διεύθυνση **https://localhost:8084**.

Για να τρέξει το frontend χρειάζεται να εκτελεστεί η εντολή **npm update**, ώστε να γίνει λήψη των βιβλιοθηκών που χρειάζεται και μετά **npm start** ώστε να τρέξει το frontend στο http://localhost:3000.

Front End

Το front end κομμάτι έγινε με την χρήση της react. Η επιλογή αυτή έγινε είναι εύκολη στην χρήση και έχει την δυνατότητα να δημιουργήσει γρήγορες ιστοσελίδες, των οποίων η ανάπτυξη και η συντήρηση είναι εύκολη.

Έγινε χρήση boostrap, η οποία βοήθησε πολύ στην ομορφιά της ιστοσελίδας.

Ο χρήστης όταν πάει στην ιστοσελίδα θα οδηγηθεί στο login page. Από εκεί θα μπορέσει να κάνει login ή να πατήσει ένα link για να κάνει sign up στην σελίδα.

Αφού κάνει login στην περίπτωση που είναι admin θα πλοηγηθεί στην σελίδα του admin όπου θα μπορέσει να δει τα στοιχειά των χρηστών και να τα κάνει export σε XML ή JSON.

Στην περίπτωση που δεν είναι admin, θα πάει στο κύριο μέρος την σελίδας.

Στο πάνω μέρος της σελίδας υπάρχει ένα navigation bar, το οποίο μεταφέρει τον χρήστη στις διάφορες καρτέλες της ιστοσελίδας όπως οι ειδοποιήσεις και οι ρυθμίσεις.

Στην αρχική σελίδα μπορεί να δει τα post των φίλων του καθώς και τα δικά του. Σε αυτά τα post μπορεί να κάνει like ή να κάνει κάποιο σχόλιο. Μπορεί επίσης να δημιουργήσει καινούριο post το οποίο μπορεί να περιέχει κείμενο, εικόνα, βίντεο ή ήχο.

Στην καρτέλα δίκτυο, ο χρήστης μπορεί να αναζητήσει κάποιον χρήστη και μπορεί να δει σε μία λίστα του φίλους του.

Στην καρτέλα αγγελίες ο χρήστης μπορεί να δει τις αγγελίες στις οποίες έχει τις απαραίτητες ικανότητες. Σε αυτές τις αγγελίες μπορεί να κάνει αίτηση. Μπορεί επίσης να δημιουργήσει μία καινούρια αγγελία και να δει τις αγγελίες που έχει δημιουργήσει.

Στην καρτέλα συζητήσεις ο χρήστης μπορεί να στείλει και να λάβει μυνήματα. Τα μυνήματα καθώς και οι επαφές ανανεώνονται κάθε κάποιο μικρό χρονικό διάστημα.

Στην καρτέλα ειδοποιήσεις ο χρήστης μπορεί να δει τα αιτήματα σύνδεσης που έχουν κάνει οι άλλοι χρήστες και να δει τα likes και σχόλια που έχουν κάνει οι άλλοι χρήστες στα post του.

Στην καρτέλα προσωπικά στοιχεία ο χρήστης μπορεί να δει τα προσωπικά του στοιχεία και να επιλέξει ποια από αυτά θα είναι ιδιωτικά και ποια δημόσια.

Στην καρτέλα ρυθμίσεις ο χρήστης μπορεί να αλλάξει το email του και τον κωδικό του.

BackEnd

Το backend κομμάτι έγινε με τη χρήση του Java SpringBoot και για τη βάση δεδομένων χρησιμοποιήθηκε το MySql.

Κατά την πρώτη εκτέλεση του κώδικα του backend(άδεια βάση δεδομένων), προστίθεται ένας χρήστης με δικαιώματα *admin*, ο οποίος ως πρώτος χρήστης στην βάση, θα έχει το id 1.

Χρησιμοποιώ το Spring Security για την διασφάλιση του backend μέρους του site, το οποίο σημαίνει ότι η πρόσβαση σε όλα τα endpoints εκτός του **/perform_login** και του **/api/v1/registration**, επιτρέπεται μόνο σε εξουσιοδοτημένους χρήστες. Η εξουσιοδότηση γίνεται μέσω ενός JWT token το οποίο δίνεται στο χρήστη μετά από ένα επιτυχημένο login request και το οποίο ο χρήστης έπειτα έχει σε κάθε request του προς το server στο header *Authorization* με περιεχόμενο *"Bearer JWTToken"*. Το login και το registration γίνονται με ένα POST request στην κατάλληλη διεύθυνση και το σώμα του είναι σε μορφή form data. Στο login επιστρέφω ένα JSON με το JWTToken και το ένα String που είναι "ADMIN" ή "USER", ανάλογα με το ρόλο του χρήστη. Στο registration, η φωτογραφία σώζεται τοπικά στο φάκελο images, μέσα σε ένα υποφάκελο με όνομα το id του χρήστη.

Αν ο χρήστης είναι admin, υπάρχει το **/api/v1/users** που επιστρέφει ένα JSON με τα απαραίτητα στοιχεία για όλους τους χρήστες.

Αν ο χρήστης θέλει τα δικά του στοιχεία, χρειάζεται ένα GET στο **/api/v1/user**, ενώ αν θέλει τα στοιχεία κάποιου άλλου χρήστη, τότε με ένα GET στο **/api/v1/user/{id}**, όπου id το id του χρήστη. Επίσης, αν θέλει το δικό του id κάνει ένα GET στο **/api/v1/id**, ενώ αν θέλει το email του, κάνει ένα GET στο **/api/v1/email**. Επιπλέον, για το δικό του όνομα χρειάζεται ένα GET στο **/api/v1/name** και για το όνομα κάποιου άλλου χρήστη, ένα GET στο **/api/v1/name/{id}**, όπου id το id του άλλου χρήστη.

Για να ενημερώσει το email του, στέλνει ένα PUT request στο **/api/v1/email** και στο σώμα απλά ένα String και ομοίως για να ενημερώσει τον κωδικό του με ένα PUT στο **/api/v1/password**.

Η ενημέρωση της εμπειρίας, της εκπαίδευσης και των δυνατοτήτων του γίνεται μέσω ενός PUT request στα **/api/v1/experience**, **/api/v1/education**, **/api/v1/skills** αντίστοιχα, με ένα JSON στο σώμα του με το κείμενο και το αν είναι ορατό σε άλλους.

Ο χρήστης στέλνει αίτημα φιλίας σε άλλο χρήστη με ένα PUT στο **/api/v1/friendRequest/{receiverid}**, όπου receiverid το id του λήπτη και για να απαντήσει σε ένα αίτημα του χρήστη με id senderid, στέλνει ένα PUT στο **/api/v1/friendRequestResponse/{senderid}** και στο σώμα ένα wrapper JSON με ένα boolean για την απάντηση στο request. Μπορεί επίσης να δει τα αιτήματα που έχει στείλει και λάβει με ένα GET στα **/api/v1/friendRequestsSent** και **/api/v1/friendRequestsReceived** αντίστοιχα. Έχει επιπλέον τη δυνατότητα να δει όλους τους φίλους του σε λίστα με GET στο **/api/v1/friends** και να ελέγξει αν είναι φίλος με τον χρήστη με id, με GET στο **/api/v1/checkFriend/{id}**.

Για να πάρει τη λίστα των επαφών του, κάνει ένα GET request στο **/api/v1/contacts**. Η αποστολή μηνύματος σε χρήστη με τον οποίο είναι επαφή και έχει id receiverId γίνεται μέσω ενός PUT στο **api/v1/sendMessage/{receiverId}**, που έχει στο σώμα ένα wrapper JSON για το κείμενο του μηνύματος και για να πάρει όλη τη συνομιλία με τον χρήστη με receiverId στέλνει ένα GET στο **api/v1/messages/{receiverId}**. Έχει και τη δυνατότητα να πάρει τα μηνύματα με τον τελευταίο χρήστη που έχει επικοινωνήσει με GET στο **api/v1/messages** και να πάρει το id της τελευταίας επαφής με την οποία επικοινώνησε με ένα GET στο **/api/v1/lastContactId**.

Υπάρχει η δυνατότητα να ανεβάσει ένα post, το οποίο μπορεί να έχει κείμενο και φωτογραφία ή ηχητικό ή βίντεο(μπορεί και μόνο κείμενο ή μόνο έναν τύπο media), στέλνοντας ένα POST request στο **/api/v1/post** με τα δεδομένα σε μορφή form data. Ο χρήστης μπορεί να πάρει μια λίστα από τα posts που έχει κάνει ο ίδιος με ένα GET στο **/api/v1/myPosts** και μία λίστα από τα δικά του posts, των φίλων του και αυτάμε τα οποία έχουν αλληλεπιδράσει οι φίλοι του σε χρονική σειρά από το πιο πρόσφατο με ένα GET στο **/api/v1/posts**. Ο χρήστης σχολιάζει σε ένα post με id post_id στέλνοντας ένα PUT request στο **/api/v1/comment/{post_id}** με ένα wrapper JSON ενός String στο σώμα του και μπορεί να πάρει τα σχόλια ενός post με ένα GET στο **/api/v1/comment/{post_id}**. Ομοίως κάνει

like σε ένα post με id post_id στέλνοντας ένα PUT request στο **/api/v1/like/{post_id}** και μπορεί να πάρει λίστα των likes που έχουν γίνει σε αυτό με ένα GET στο **/api/v1/postLikes/{post_id}**.

Ένα GET στο **/api/v1/notifications** θα επιστρέψει τις ειδοποιήσεις του χρήστη για comments ή likes άλλων χρηστών σε post του.

Ο χρήστης μπορεί να ανεβάσει μία αγγελία με ένα POST στο **/api/v1/advert** με ένα JSON στο σώμα του με τις πληροφορίες της αγγελίας και να κάνει αίτηση σε μία αγγελία με id advert_id στέλνοντας ένα PUT στο **/api/v1/apply/{advert_id}**. Επίσης, μπορεί να πάρει μια λίστα από τις αγγελίες του με ένα GET στο **/api/v1/myAdverts** και μια λίστα από αγγελίες άλλων στις οποίες έχει *τουλάχιστον* μία από τις απαιτούμενες δυνατότητες με ένα GET στο **/api/v1/adverts**.

Τέλος, έχει τη δυνατότητα να ψάξει χρήστες με βάση το όνομα τους με ένα POST στο **/api/v1/search**, που επιστρέφει λίστα από JSON χρηστών. Η αναζήτηση είναι **case insensitive**, αναγνωρίζει ακόμα και το να έχεις ένα substring του ονόματος και είναι *SQL inject safe*, καθώς δεν χρησιμοποιείται string concatenation για την είσοδο του χρήστη στο SQL query, αλλά είναι parameterized.

Επίλογος

Ένα από τα δύσκολα κομμάτια της εργασία ήταν το refresh κάποιον δεδομένων καθώς αλλάζουν στο backend. Αυτό το λύσαμε κάνοντας fetch κάποια δεδομένα κάθε 10 δευτερόλεπτα.

Ένα ακόμα ήταν η αλλάγη του email στις ρυθμίσεις. Υπήρχαν κάποια τεχικά θέματα στην αλλαγή του email στο backend και για να το λύσουμε ο χρήστης γίνεται logged out όταν αλλάζει το email του.

Ένα ακόμα πρόβλημα που μας δυσκόλεψε ήταν το να εισάγουμε το SSL certificate στους διάφορους browser. Για την όλη διαδικασία ακολουθήσαμε τις οδηγίες που δώθηκαν και στο μάθημα, αλλά δεν άφηνε να εισάγω το certificate στα authorities(σε firefox και chromium) γιατί δεν ήταν root certificate ενώ το είχα προσθέσει στον κατάλληλο κατάλογο των Linux. Αυτό που βρήκα είναι ότι οι browsers δεν χρησιμοποιούν το certificate store των συστημάτων, αλλά δικά τους. Η λύση που βρήκα ήταν ένα bash script, το οποίο με την εντολή certutil τροποποιεί τα certificate stores cert8.db(Firefox) και cert9.db(Chromium & Chrome), αφού τα εντοπίσει.

Επίσης, είχαμε ένα θέμα με το να έχει πρόσβαση το frontend στις φωτογραφίες και τα media που σώζονται τοπικά από το backend. Η λύση που βρήκαμε ήταν να σώζουμε τα αρχεία μέσα στο φάκελο images, να προσθέσουμε ένα resource handler που θα επέτρεπε να έχει πρόσβαση εκεί και να μην χρειάζεται authentication γι'αυτά τα requests. Συνεπώς, τα media ήταν διαθέσιμα στο link **/https://localhost:8084/images/{mediaName}**, όπου mediaName το όνομα του αρχείου.