

Job Finder

ΕΡΓΑΣΙΑ ΤΕΔΙ

Καλοκαίρι 2024

Μέλη ομάδας

Δημήτριος Χρυσός - 1115202100275

Αναστάσιος Μουμουλίδης - 1115202100108

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ	2
2. ΓΕΝΙΚΕΣ ΠΑΡΑΔΟΧΕΣ	4
3. ΒΑΣΙΚΑ ΣΧΗΜΑΤΑ	6
4. API CALLS	8
5. COMPONENTS	11
6. ΕΠΙΛΟΓΟΣ	20

ΕΙΣΑΓΩΓΗ

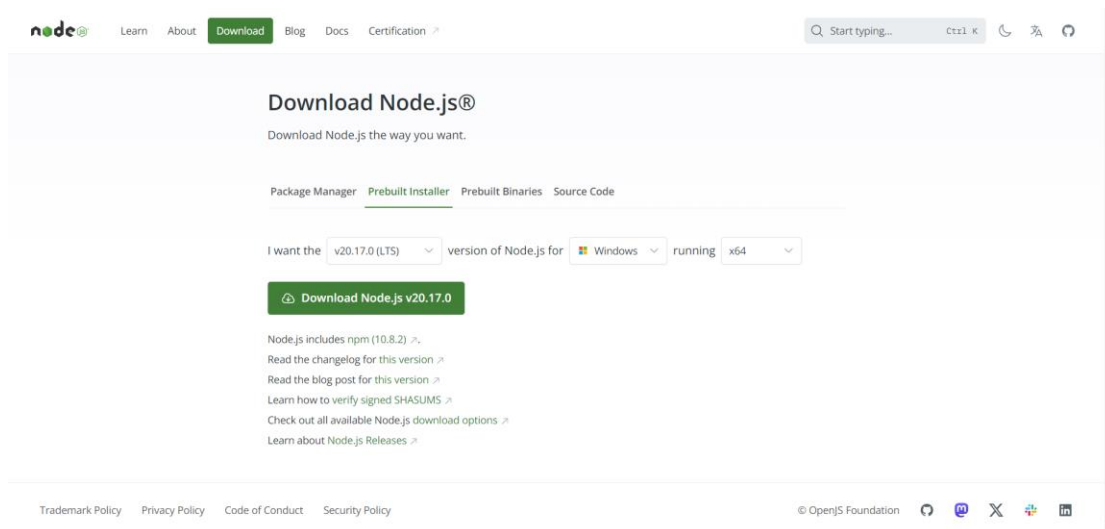
Καλησπέρα σας! Είμαστε οι φοιτητές 3ου έτους Δημήτριος Χρυσός και Αναστάσιος Μουμουλίδης και αυτή είναι η αναφορά της εργασίας μας στο πλαίσιο του μαθήματος Τεχνολογίες Εφαρμογών Διαδικτύου.

Σε αυτή την αναφορά θα αναλύσουμε τις σχεδιαστικές επιλογές και τον συνολικό τρόπο σκέψης μας, καθώς και θα δώσουμε μια γενική εικόνα του κώδικα των επιμέρους στοιχείων που αποτελούν την εφαρμογή μας.

Ας ξεκινήσουμε από τα βασικά. Για την υλοποίηση της εφαρμογής μας, το framework που χρησιμοποιήσαμε είναι το Next.js (με χρήση Node.js στο backend) σε συνδυασμό με JavaScript και Tailwind CSS. Για τη βάση δεδομένων μας, χρησιμοποιήσαμε το εργαλείο MongoDB Atlas.

Πάμε τώρα στις οδηγίες εγκατάστασης και εκτέλεσης της εφαρμογής. Αρχικά, απαιτείται η εγκατάσταση του Node.js.

Για την εγκατάσταση του Node.js συνιστάται να χρησιμοποιηθεί κάποιος prebuilt installer που βρίσκεται στο παρακάτω link: <https://nodejs.org/en/download/prebuilt-installer>



Αφού κατεβάσετε τον installer, τον τρέχετε και ακολουθείται τις οδηγίες που σας δίνονται στο παράθυρο της εφαρμογής. Μόλις ολοκληρώσετε την εγκατάσταση, είστε έτοιμοι!

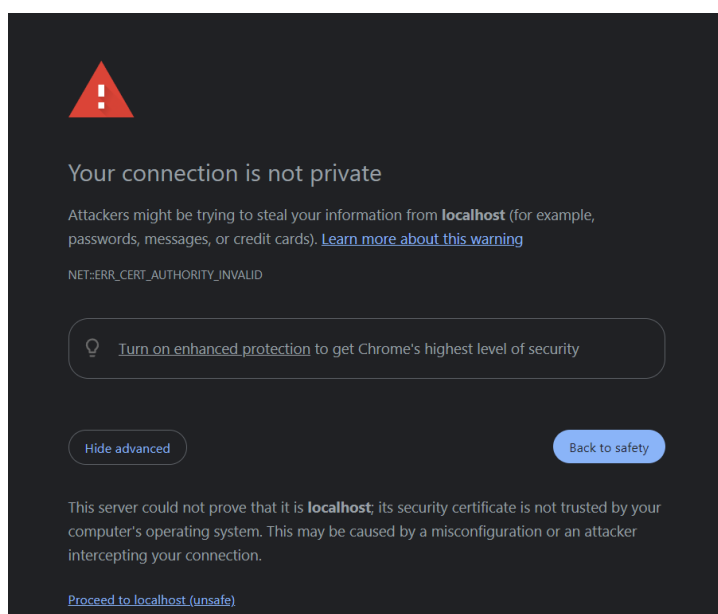
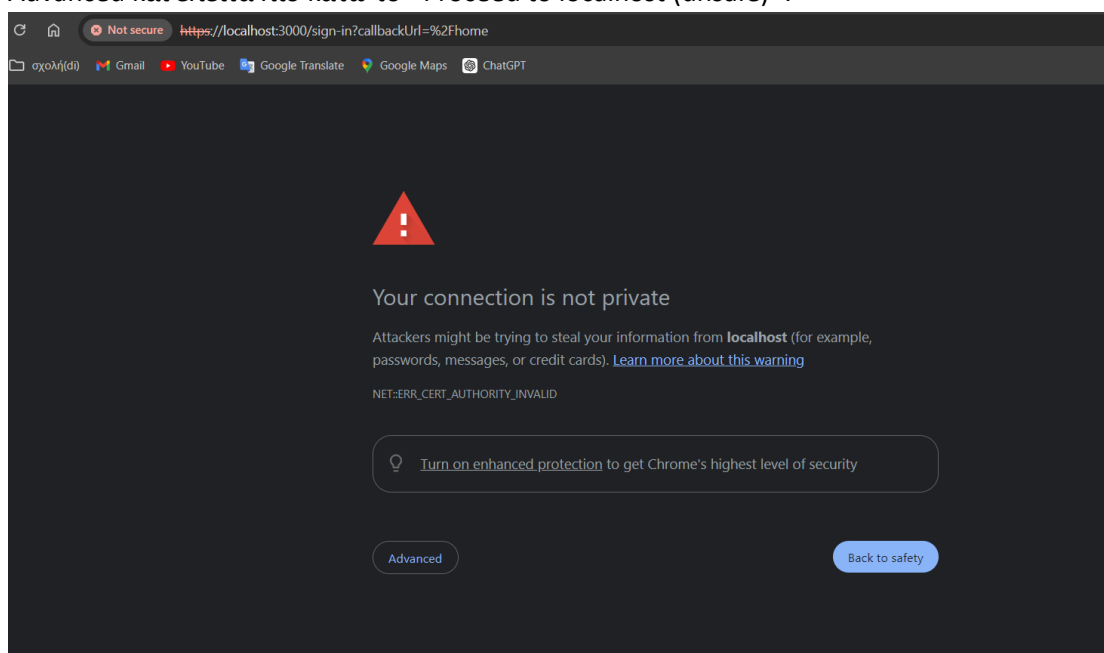
Το μόνο που μένει τώρα, είναι να κάνετε decompress τον φάκελο της εργασίας και να τον ανοίξετε με κάποιο IDE, π.χ. το VS-Code. Έπειτα, σε κάποιο terminal, αρκεί να

πληκτρολογήσετε την εντολή ***npm run dev*** και η εφαρμογή θα ξεκινήσει να τρέχει, στη διεύθυνση <https://localhost:3000>

Στα επόμενα κεφάλαια ακολουθούν κάποιες γενικές παραδοχές που κάναμε κατά την εκτέλεση και δημιουργία της εργασίας, καθώς και μία ανάλυση των API calls και επιμέρους στοιχείων που χρησιμοποιήσαμε.

ΓΕΝΙΚΕΣ ΠΑΡΑΔΟΧΕΣ

1. Όλα τα αρχεία (εικόνα, ήχος, βίντεο) αποθηκεύονται τοπικά στο directory του project μας και όχι σε κάποιο εξωτερικό server.
2. Για τη κρυπτογράφηση των HTTP αιτήσεων μέσω πρωτοκόλλου SSL/TLS, δημιουργήσαμε Self-Signed Certificate το οποίο διαχειρίζεται το αρχείο server.js μαζί με το httpsAgent.js (για τα API calls) στο αρχικό directory. Λόγο της επιλογής του συγκεκριμένου certificate ο browser βγάζει το παρακάτω warning που εύκολα όμως μπορεί να αγνοηθεί πατώντας κάτω αριστερά το κουμπί Advanced και έπειτα πιο κάτω το «Proceed to localhost (unsafe)».



3. Τα routes της εφαρμογής μας υλοποιούνται με αρκετά απλοϊκό τρόπο λόγω της ευκολίας που προσφέρει το Next.js. Έτσι τα αρχεία «page.jsx» των sub-

directories του directory «app», αντιστοιχούν σε κάθε route της εφαρμογής μας.

4. Μέσα στα προαναφερθέντα αρχεία «page.jsx» κυρίως καλούνται τα αντίστοιχα components που πραγματοποιούν και εμφανίζουν τις λειτουργίες του κάθε route.
5. Το «page.jsx» αρχείο μέσα στο βασικό «app» directory, αποτελεί την αρχική σελίδα της εφαρμογής πριν την είσοδο του χρήστη σε αυτή.
6. Το «layout.jsx» αρχείο μέσα στο βασικό «app» directory, διαμορφώνει κατάλληλα κάθε σελίδα της εφαρμογής ώστε να περιέχεται το navigation bar και να εξασφαλίζει το authentication των χρηστών.
7. Για την υλοποίηση της λειτουργίας του Admin role, το user schema περιέχει μία Boolean μεταβλητή «admin» με default τιμή false για όλους τους users.
8. Ο Admin, δημιουργείται στην εφαρμογή τη πρώτη φορά που κάποιος προσπαθήσει να μπει με τα στοιχεία του σε αυτή και ύστερα υπάρχει εκεί για κάθε νέα είσοδο. Τα στοιχεία του είναι:
 - a. email: admin@u.com
 - b. password: admin123
9. Για τη δημιουργία μίας δικής μας βάσης, δημιουργήσαμε το script «populateDB.mjs», αυτό στο τέλος του καλεί την εντολή:

```
// Populate the database with 1000 users
populateDB(1000);
```

Με διαφορετικό αριθμό ως argument μπορεί κάποιος να βάλει περισσότερους ή λιγότερους χρήστες. Το script αυτό είναι υπεύθυνο για τη δημιουργία ενός ορισμένου αριθμού χρηστών που περιέχουν posts/listings/likes/comments και άλλα. Για να τρέξει κανείς αυτό το script, χρειάζεται απλά να πατήσει την εντολή «node populated.mjs» σε terminal του αρχικού directory.

10. Τα matrix factorization αρχίζουν πρώτη φορά από το αρχείο server.js του αρχικού directory και από εκείνη τη στιγμή και έπειτα ανανεώνονται κάθε 30 λεπτά. Το factorization σαν διαδικασία γίνεται σε worker threads του Node.js.
11. Το factorized matrix για τα listings και τα posts, αποθηκεύεται σε μικρότερα κομμάτια (chunks). Αυτή η επιλογή έγινε λόγω της δυσκολίας που υπήρξε στο ανέβασμα (περιορισμό της πλατφόρμας στο μέγεθος κάθε document) και κατέβασμα των matrix από τη βάση δεδομένων λόγω του μεγάλου μεγέθους τους. Έτσι η επιλογή των post και των listing που εμφανίζονται στους χρήστες στα αντίστοιχα pages είναι αρκετά πιο αποδοτική.
12. Το αρχείο «middleware.js» χρησιμοποιείται για να περιορίσουμε τη πρόσβαση στις περισσότερες σελίδες, μόνο σε authenticated χρήστες.
13. Έχουμε δημιουργήσει ένα example account πέρα από τα 1000 του populated.mjs, το οποίο έχει δημιουργήσει τρία post, ένα με εικόνα, ένα με βίντεο και ένα με ήχο. Τα στοιχεία του είναι: email: “ex@gmail.com” και password: “123”.

ΒΑΣΙΚΑ ΣΧΗΜΑΤΑ

Πάμε τώρα στα βασικά σχήματα που χρησιμοποιήσαμε για την υλοποίηση της εφαρμογής.

1. User schema

Το συγκεκριμένο σχήμα αναπαριστά τον χρήστη στον χώρο της εφαρμογής. Περιέχει όλες τις βασικές πληροφορίες που αφορούν τον χρήστη (όνομα, επίθετο, email, τηλέφωνο, κωδικό κλπ), πληροφορίες σχετικά με το επάγγελμά του (θέση εργασίας, skills, εκπαίδευση κλπ) καθώς και άλλες πληροφορίες σχετικά με τη δραστηριότητά του στην εφαρμογή (όπως πχ σε ποια post έχει κάνει like, comment κλπ).

2. Notification schema

Πρόκειται για ένα βοηθητικό σχήμα, ώστε να στέλνονται σωστά τα notifications. Περιέχει το id του χρήστη-παραλήπτη, το id του post το οποίο αφορά, καθώς και μια περιγραφή.

3. Post schema

Αυτό το σχήμα ορίζει πλήρως τα άρθρα/post της εφαρμογής. Περιέχει βασικές πληροφορίες για το post (id συντάκτη, κείμενο, αρχείο, likes κλπ).

4. Comment schema

Μοιάζει πολύ με το σχήμα του post, αλλά είναι αρκετά πιο απλοϊκό. Περιέχει μόνο user id και text.

5. Listing schema

Παρόμοιο με το post schema, αλλά προσαρμοσμένο για τις αγγελίες. Π.χ. αντί για ένα text segment, έχει δύο ξεχωριστά τμήματα κειμένου: ένα για το job position κι ένα για την περιγραφή του.

6. Application schema

Πανομοιότυπο με το comment schema, προσαρμοσμένο για τις αγγελίες.

7. Message schema

Περιλαμβάνει τα id του παραλήπτη, του αποστολέα και του chat στο οποίο εμπεριέχεται, καθώς και το κείμενο που περιέχει.

8. Chat schema

Περιλαμβάνει τα IDs των συμμετεχόντων στη συνομιλία, καθώς κι ένα array από τα μέχρι τώρα απεσταλμένα μηνύματα στο συγκεκριμένο chat.

9. MatrixPosts schema

Χρησιμοποιείται για την αποθήκευση κάθε μέρους (chunk) του factorized matrix των post.

10. MatrixListings schema

Χρησιμοποιείται για την αποθήκευση κάθε μέρους (chunk) του factorized matrix των post.

API CALLS

Αφού τελειώσαμε με τα σχήματα (που βρίσκονται στον φάκελο */models*), πάμε στα API calls που χρησιμοποιούμε (στον φάκελο */api*).

/auth/[...nextauth]/route.js: Υπεύθυνο για το authentication ενός χρήστη στην εφαρμογή.

/chats/route.js: Η POST δημιουργεί μια νέα συνομιλία μεταξύ δύο χρηστών και η GET ανακτά όλες τις συνομιλίες ενός χρήστη.

/chats/chat-exists/route.js: Η GET ελέγχει αν υπάρχει ένα συγκεκριμένο chat στη βάση δεδομένων.

/chats/get-chat-by-id/route.js: Η GET ανακτά από τη βάση δεδομένων το chat με το συγκεκριμένο id που δίνεται σαν παράμετρος.

/chats/messages/route.js: Η POST προσθέτει ένα νέο μήνυμα σε μια συνομιλία και η GET επιστρέφει όλα τα μηνύματα μιας συνομιλίας.

/chats/update-last-chat/[id]/route.js: Η PUT κάνει update τον user ώστε να εμφανίζεται σαν τελευταία συνομιλία όντως η τελευταία συνομιλία του.

/connections/add-remove-connection/route.js: Η POST κάνει remove ή add μία σύνδεση μεταξύ δύο χρηστών.

/connections/are-connected/route.js: Η POST ελέγχει αν δύο χρήστες είναι συνδεδεμένοι.

/connections/get-all-connections/[id]/route.js: Η GET επιστρέφει όλες τις συνδέσεις ενός χρήστη με συγκεκριμένο id.

/connections/requests/route.js: Η POST στέλνει ή διαγράφει ένα αίτημα σύνδεσης από έναν χρήστη σε κάποιον άλλον.

/connections/requests/request-exists/route.js: Η POST ελέγχει αν υπάρχει ήδη στη βάση δεδομένων αίτημα σύνδεσης από έναν χρήστη σε έναν άλλον.

/connections/requests/[id]/route.js: Η GET επιστρέφει όλα τα αιτήματα σύνδεσης του χρήστη με συγκεκριμένο id.

/download-user/[id]/route.js: Η GET επιστρέφει τα δεδομένα ενός χρήστη με συγκεκριμένο id (χρησιμοποιεί για τον admin).

/files/[id]/route.js: Η POST ανεβάζει ένα αρχείο στη βάση δεδομένων και η DELETE το διαγράφει από αυτή.

/listing/route.js: Η POST δημιουργεί μία νέα αγγελία και η GET ανακτά όλες τις αγγελίες ενός συγκεκριμένου χρήστη.

/listing/[id]/route.js: Η GET επιστρέφει μία συγκεκριμένη αγγελία.

/listing/application/route.js: Η POST προσθέτει μία αίτηση σε μία αγγελία.

/listing/get-all-listings/route.js: Η GET επιστρέφει όλες τις αγγελίες που βρίσκονται στη βάση δεδομένων.

/listing/get-application-for-listing/route.js: Η GET επιστρέφει όλες τις αιτήσεις κάτω από μια συγκεκριμένη αγγελία.

/listing/get-listings-from-ids/route.js: Η GET επιστρέφει όλες τις αγγελίες που έχουν δημοσιεύσει οι χρήστες με τα ids που δίνονται σαν παράμετροι.

/listing/views/route.js: Η POST προσθέτει μία προβολή σε μια συγκεκριμένη αγγελία και η GET επιστρέφει τις συνολικές προβολές μιας αγγελίας.

/listingFiles/route.js: Η POST ανεβάζει στον server ένα αρχείο που σχετίζεται με τις αγγελίες.

/matrix-factorization-listings/matrix-exists/route.js: Η GET ελέγχει αν υπάρχει ήδη matrix για τις αγγελίες, και αν υπάρχει, το επιστρέφει.

/matrix-factorization-listings/remove-all-chunks/route.js: Η DELETE διαγράφει όλα τα κομμάτια (chunks), του matrix από τη βάση.

/matrix-factorization-listings/return-userRow-postRow/route.js: Η GET επιστρέφει, αν υπάρχουν, τη πρώτη γραμμή του πρώτου chunk που περιέχει τη λίστα με τα listing id του matrix και τη γραμμή του χρήστη στο Matrix.

/matrix-factorization-listings/save-array/route.js: Η POST δημιουργεί ένα κομμάτι (chunk) του factorized matrix.

/matrix-factorization-posts: Έχει ίδια ακριβώς λειτουργικότητα με αυτό για τις αγγελίες, απλώς αυτό ασχολείται με τα άρθρα.

/notifications/[id]/route.js: Η GET ανακτά όλες τις ειδοποιήσεις που υπάρχουν στη βάση δεδομένων και αντιστοιχούν σε έναν συγκεκριμένο χρήστη και η DELETE διαγράφει μία συγκεκριμένη ειδοποίηση από τη βάση δεδομένων.

/password/[id]/route.js: Η PUT αλλάζει τον κωδικό ενός συγκεκριμένου χρήστη.

/post: Πανομοιότυπες λειτουργίες με το /listing απλά προσαρμοσμένες για άρθρα.

/postFiles: Πανομοιότυπες λειτουργίες με το /listingFiles, απλά για άρθρα.

/profile/route.js: Η POST δημιουργεί έναν νέο χρήστη, η GET επιστρέφει όλους τους χρήστες στη βάση δεδομένων και η DELETE διαγράφει έναν συγκεκριμένο χρήστη, καθώς και όλα τα άρθρα, τις αγγελίες και τα αρχεία του.

/profile/[id]/route.js: Η PUT κάνει update τις πληροφορίες ενός συγκεκριμένου χρήστη και η GET επιστρέφει έναν συγκεκριμένο χρήστη.

/profile/public-info/route.js: Η POST προσθέτει ή αφαιρεί κάποια πληροφορία από τις δημόσιες πληροφορίες ενός χρήστη και η GET επιστρέφει όλες τις δημόσιες πληροφορίες ενός χρήστη.

/profile/info/[id]/route.js: Η PUT κάνει update τις επαγγελματικές πληροφορίες ενός χρήστη και η GET επιστρέφει έναν συγκεκριμένο χρήστη.

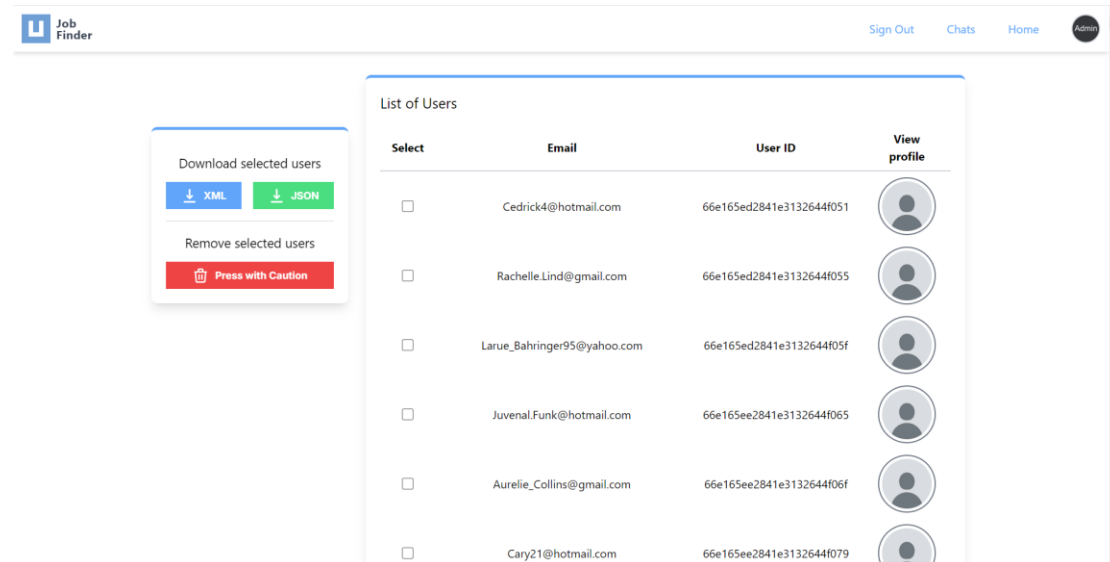
/search/route.js: Η GET ψάχνει και επιστρέφει τους χρήστες των οποίων το όνομα ταιριάζει με το κείμενο αναζήτησης.

/userExists/route.js: Η POST ελέγχει αν υπάρχει ένας συγκεκριμένος χρήστης στη βάση δεδομένων, κι αν υπάρχει, τον επιστρέφει.

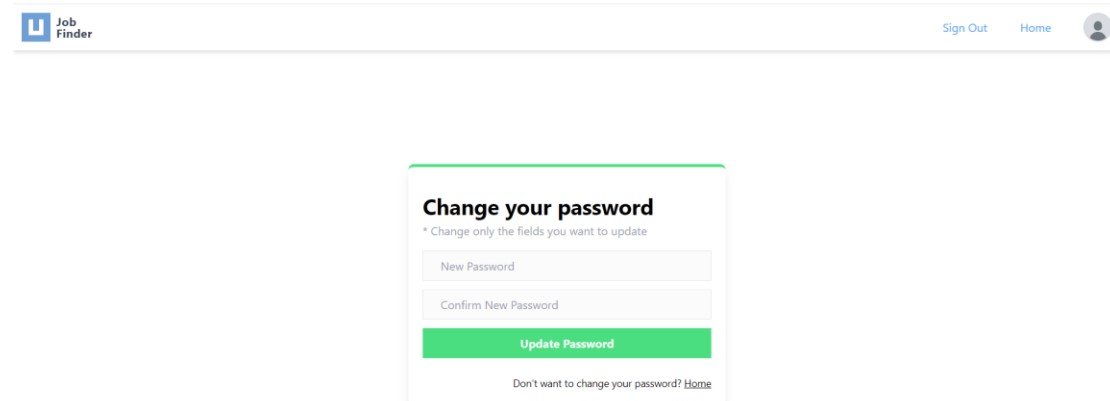
COMPONENTS

Πάμε τώρα στα επιμέρους components (στον φάκελο `/components`):

AdminHomePage.jsx: Περιέχει τον κώδικα για την εμφάνιση και λειτουργία της αρχικής σελίδας, που βλέπει όταν εισέρχεται στην εφαρμογή κάποιος από τους διαχειριστές.

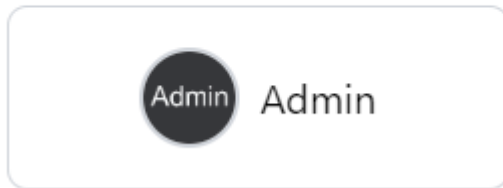


ChangePassword.jsx: Περιέχει τον κώδικα της φόρμας αλλαγής κωδικού ενός χρήστη.

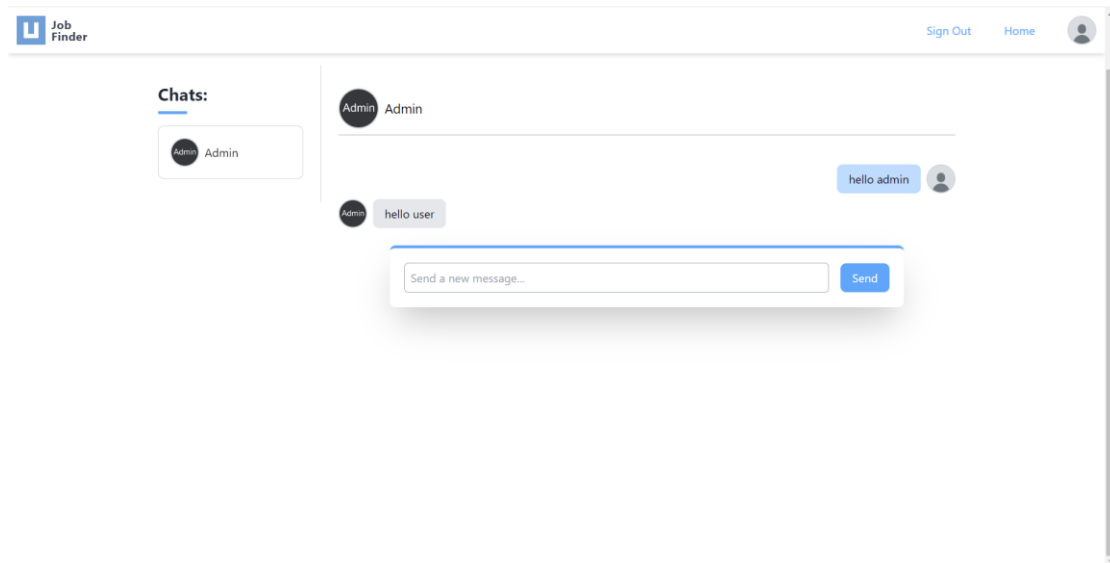


ChatCard.jsx: Αυτό το αρχείο είναι υπεύθυνο για την εμφάνιση ενός συγκεκριμένου chat στη σελίδα των συνομιλιών ενός χρήστη.

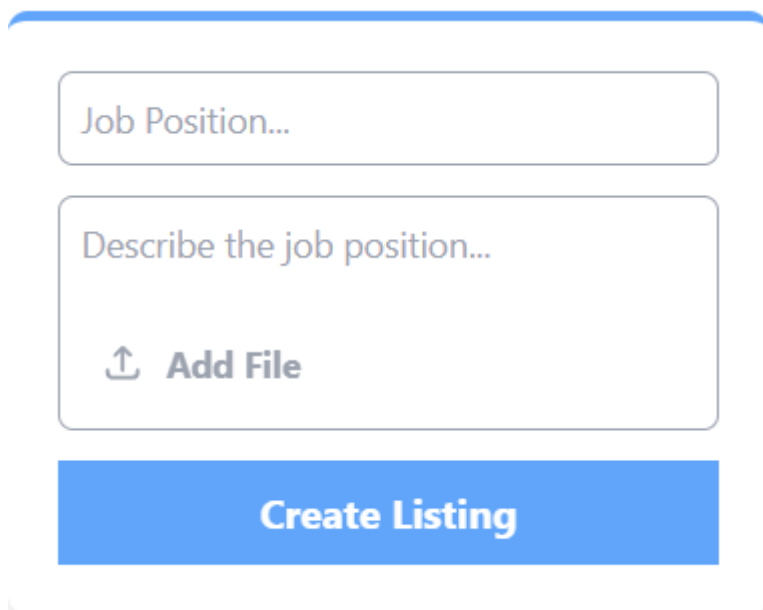
Chats:



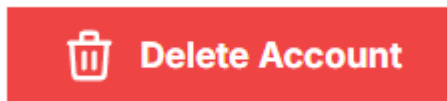
ConversationsPage.jsx: Περιέχει τον κώδικα της σελίδας των Συζητήσεων κάθε χρήστη.



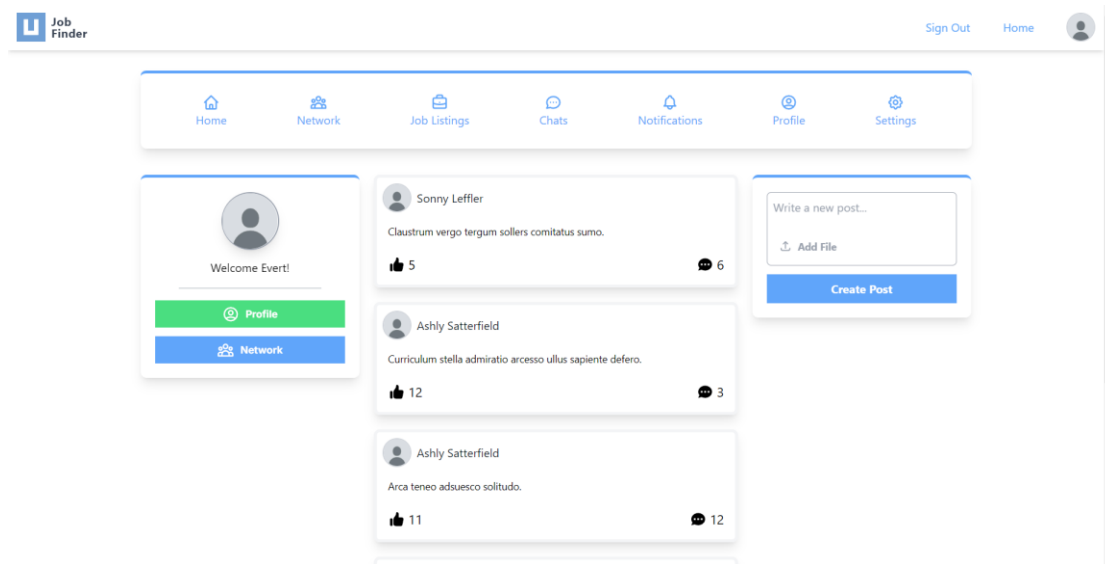
CreateListing.jsx: Υπεύθυνο για την εμφάνιση ενός πλαισίου δεξιά στη σελίδα των Αγγελιών, μέσω του οποίου ένας χρήστης μπορεί να δημιουργήσει μία νέα αγγελία.



DeleteAccountBtn.jsx: Το συγκεκριμένο στοιχείο υλοποιεί τη λειτουργία και εμφάνιση ενός κουμπιού, υπεύθυνο για τη διαγραφή του λογαριασμού ενός χρήστη.

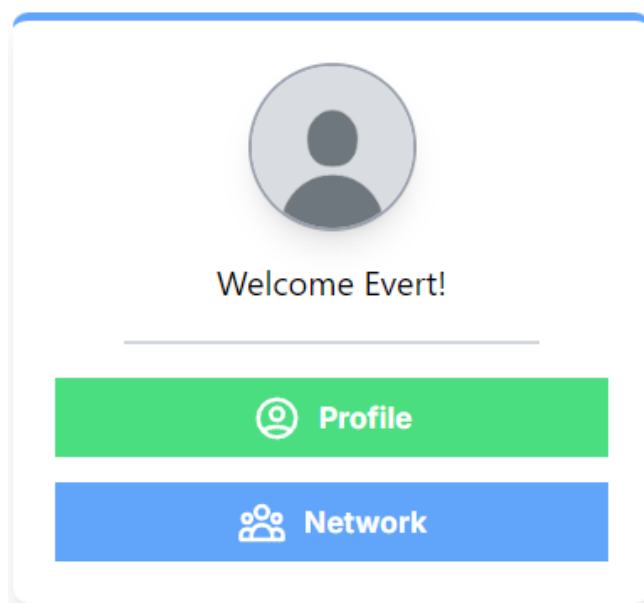


HomePage.jsx: Περιέχει των κώδικα της αρχικής σελίδας που βλέπει ένας επαγγελματίας όταν εισέρχεται στην εφαρμογή.

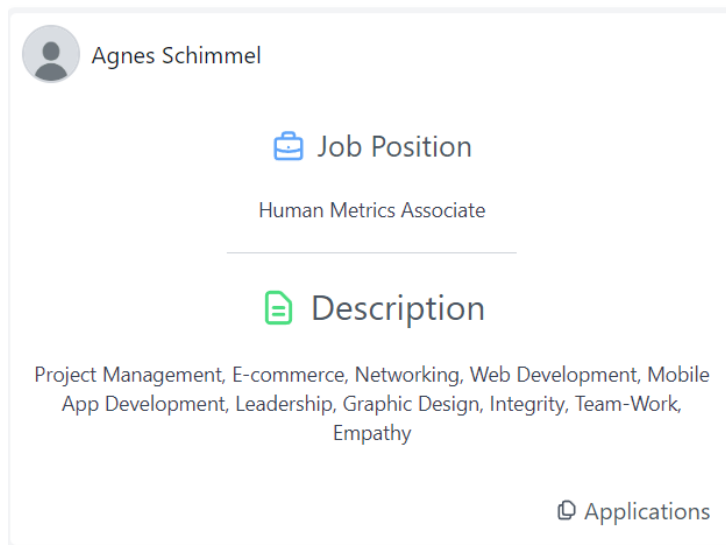


InitAdminUser.jsx: Δημιουργεί by default έναν admin user όταν προσπαθήσει κάποιος να κάνει login με τα στοιχεία αυτού.

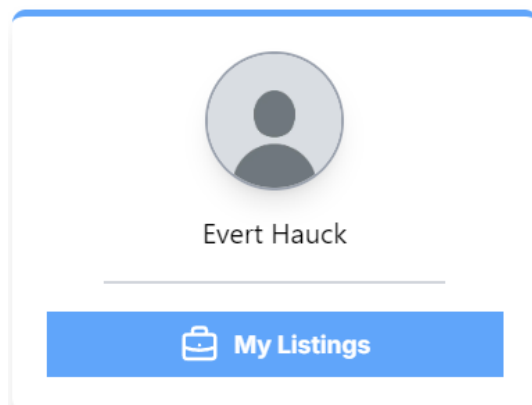
LeftHomePage.jsx: Υπεύθυνο για την εμφάνιση ενός μενού στο αριστερό μέρος της αρχικής σελίδας, μέσω του οποίου ο χρήστης μπορεί να πλοηγηθεί στο προφίλ ή στο δίκτυό του.



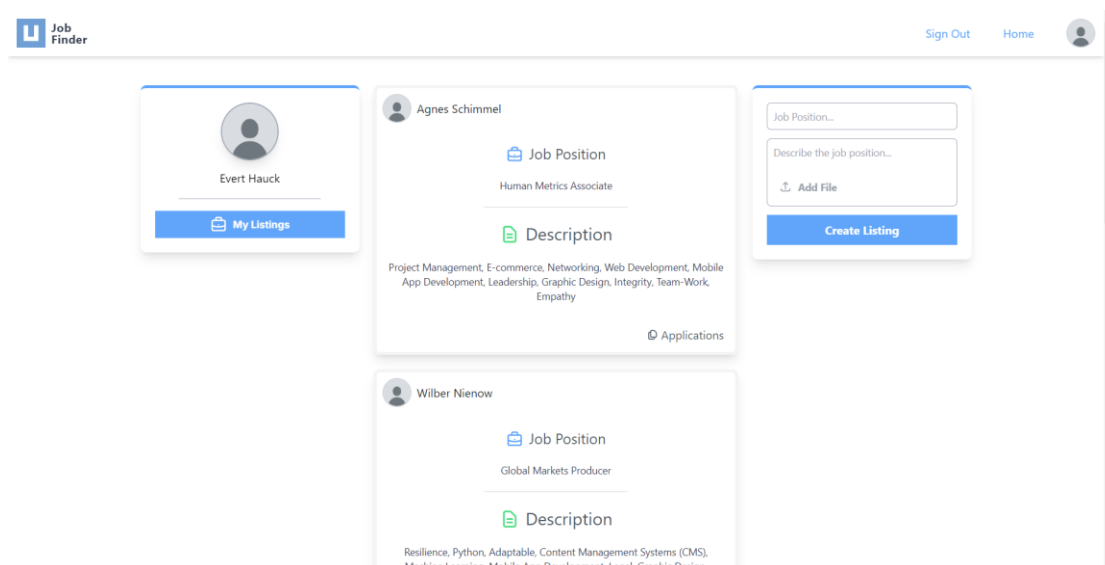
ListingCard.jsx: Υπεύθυνο για την εμφάνιση μιας συγκεκριμένης αγγελίας.



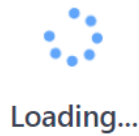
ListingsMenu.jsx: Υπεύθυνο για την εμφάνιση ενός μενού στο αριστερό μέρος της σελίδας των αγγελιών, μέσω του οποίου ο χρήστης μπορεί να δει τις αγγελίες που έχει αναρτήσει ο ίδιος, καθώς και τις αιτήσεις άλλων επαγγελματιών σε αυτές.



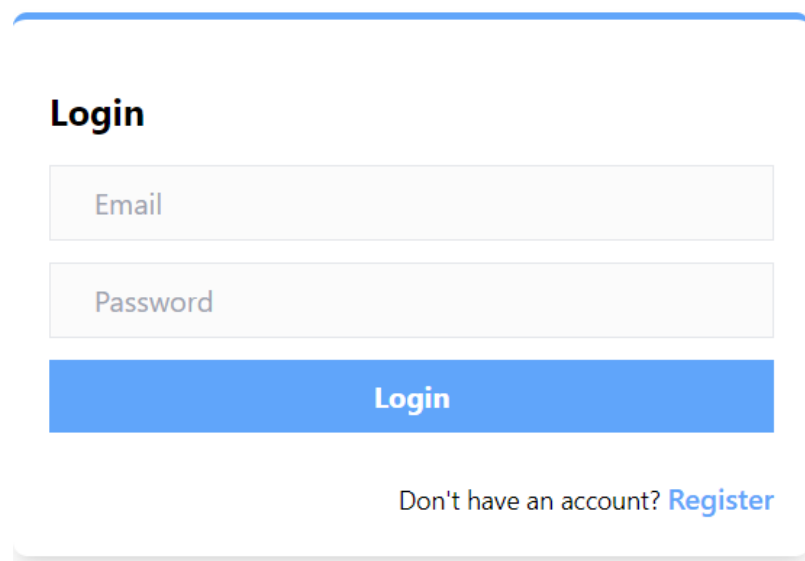
ListingsPage.jsx: Περιέχει τον κώδικα της σελίδας των αγγελιών του χρήστη.



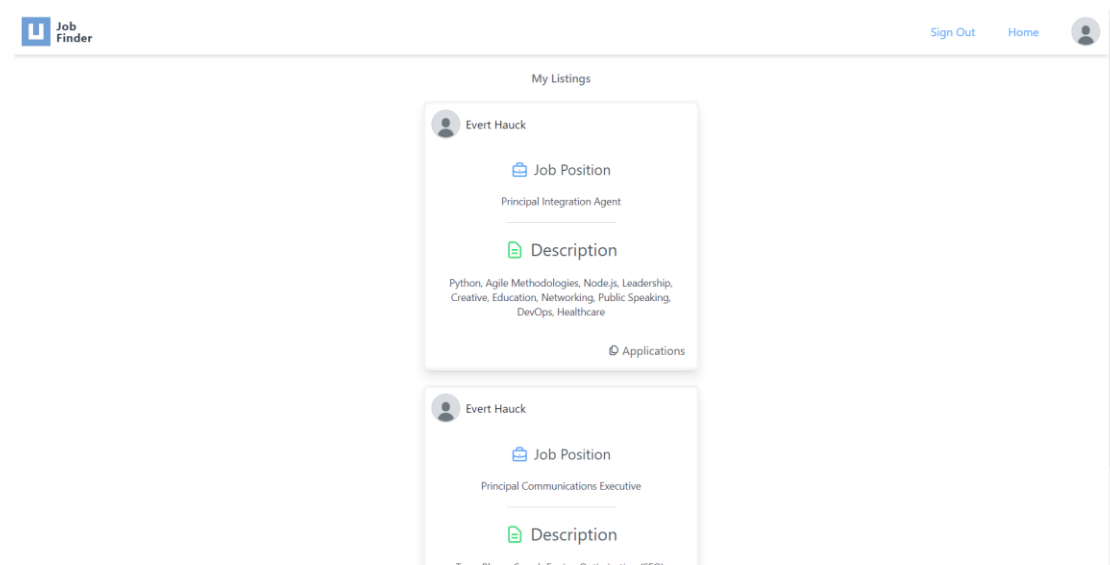
Loading.jsx: Υπεύθυνο για το front-end κομμάτι που εμφανίζεται στον χρήστη όταν η εφαρμογή φορτώνει κάποια δεδομένα.



LoginForm.jsx: Περιέχει τον κώδικα της φόρμας που συμπληρώνει ο χρήστης και την υλοποίηση την λειτουργείας εισόδου χρήστη στην εφαρμογή.

A login form UI mockup. It features a white card with a blue border. At the top left is the title "Login" in bold. Below it are two input fields: "Email" and "Password". A blue "Login" button is positioned below the password field. At the bottom right, there is a link that says "Don't have an account? Register".

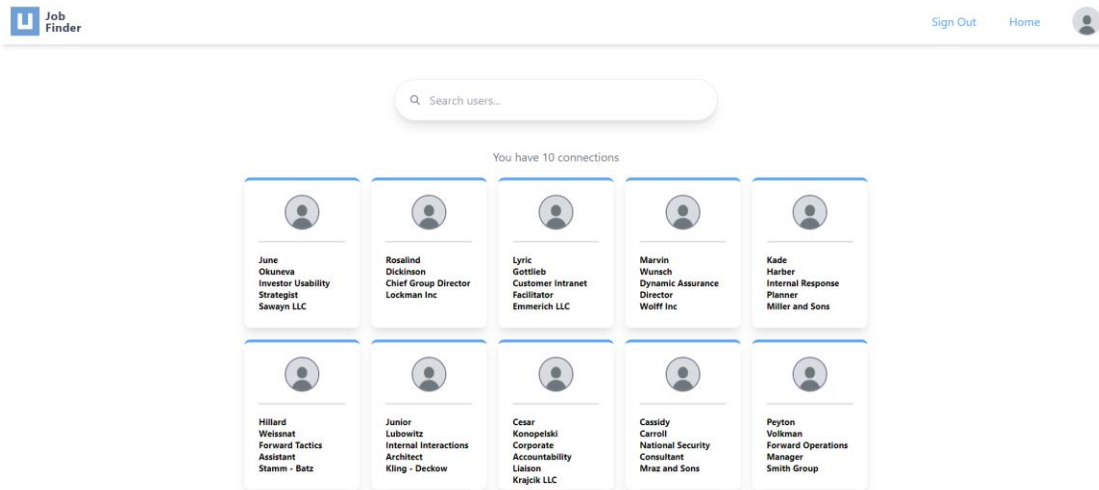
MyListings.jsx: Περιέχει τον κώδικα της σελίδας στην οποία ο χρήστης βλέπει τις αγγελίες του.



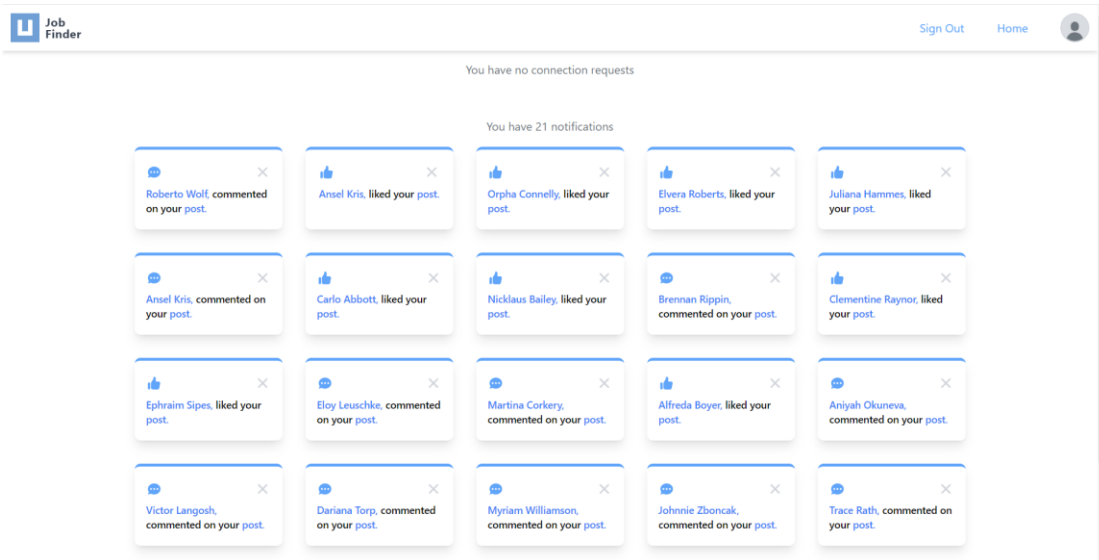
Nav.jsx: Περιέχει τον κώδικα της μπάρας πλοήγησης που υπάρχει στην κορυφή κάθε σελίδας.



NetworkPage.jsx: Περιέχει τον κώδικα που είναι υπεύθυνος για τη σελίδα του δικτύου του χρήστη.



NotificationsPage.jsx: Περιέχει τον κώδικα που είναι υπεύθυνος για τη σελίδα των Ειδοποιήσεων του χρήστη.



PersonalInfoForm.jsx: Περιέχει τον κώδικα της φόρμας μέσω της οποίας ο χρήστης μπορεί να επεξεργαστεί τις προσωπικές και επαγγελματικές του πληροφορίες.

Edit Personal Info

* Check only the fields you want to make available to the public
* Change only the fields you want to update

Job Position
 ☐

Employment Agency
 ☐

Experience
 ☐

Education
 ☐

Skills
 ☐

[Update Profile](#)

PostCard.jsx: Υπεύθυνο για την εμφάνιση ενός συγκεκριμένου άρθρου.

Ashly Satterfield

Curriculum stella admiratio arcesso ullus sapiente defero.

12 3

ProfileView.jsx: Υπεύθυνο για την προβολή ενός συγκεκριμένου προφίλ.

Job Finder [Sign Out](#) [Home](#)

Name: **Sonny**
Surname: **Leffler**
Email: **May.Wiegand@hotmail.com**
Phone Number: **1-393-575-7594 x74294**

[Start Chat](#) [Add Connection](#)

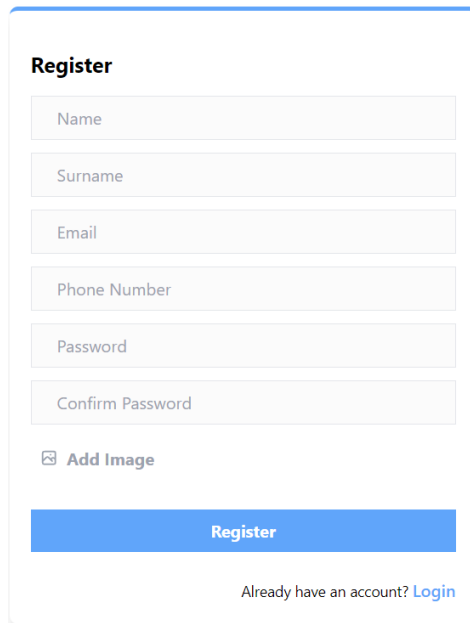
Sonny Leffler
Clastrum vergo tergum sollers comitatus sumo.
 5 6

Sonny Leffler
Vacuus volubilis blanditiis maiores adamo cras distinctio uberrime tempora.
 6 7

Sonny Leffler
Solitudo dolor tredecim cursus desidero desino trepide.
 8 8

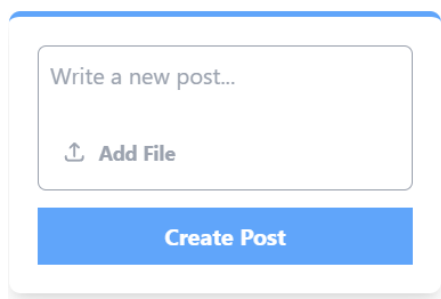
Providers.js: Αρχείο υπεύθυνο για την εξασφάλιση του authentication για την εφαρμογή.

RegisterForm.jsx: Περιέχει τον κώδικα της φόρμας που συμπληρώνει ο χρήστης χρήστης και την υλοποίηση την λειτουργείας για να πραγματοποιήσει εγγραφή στην εφαρμογή.



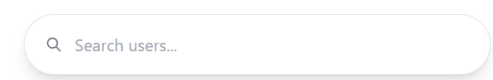
The image shows a 'Register' form with a title 'Register' at the top. Below the title are six input fields: 'Name', 'Surname', 'Email', 'Phone Number', 'Password', and 'Confirm Password'. Below these fields is a link 'Add Image' with a small icon. At the bottom of the form is a blue button labeled 'Register'. Below the button is a link 'Already have an account? Login'.

RightHomePage.jsx: Υπεύθυνο για την εμφάνιση ενός πλαισίου δεξιά στην αρχική σελίδα, μέσω του οποίου ένας χρήστης μπορεί να δημιουργήσει ένα νέο άρθρο.



The image shows a 'Create Post' form. It has a text area with the placeholder text 'Write a new post...'. Below the text area is a link 'Add File' with a small icon. At the bottom of the form is a blue button labeled 'Create Post'.

SearchBar.jsx: Υπεύθυνο για τη δημιουργία μίας μπάρας αναζήτησης χρηστών στη βάση δεδομένων.



The image shows a search bar with a magnifying glass icon and the placeholder text 'Search users...'.

Settings.jsx: Περιέχει τον κώδικα της σελίδας των Ρυθμίσεων ενός χρήστη.



Settings

* Change only the fields you want to update

Evert

Hauck

Aurelie_Collins@gmail.com

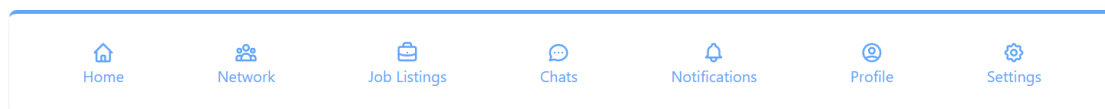
1-861-756-8232 x21804

☐ Update Image

Update Profile

You want to change your password? [Change your password](#)

TopHomeBar.jsx: Μία μπάρα που βρίσκεται στην κορυφή της αρχικής σελίδας, μέσω της οποίας ο χρήστης μπορεί να πλοηγηθεί σε κάποια από τις υπόλοιπες σελίδες, κάνοντας κλικ στο αντίστοιχο εικονίδιο.



Μέσα στον φάκελο /app , εκτός από τον φάκελο /api υπάρχουν κι άλλοι φάκελοι, οι φάκελοι των σελίδων (π.χ. conversations, home, listings, network κλπ). Κάθε ένας από αυτούς τους φακέλους περιέχει ένα μόνο αρχείο page.jsx και χρησιμοποιώντας τον κατάλληλο συνδυασμό από API calls και components δημιουργεί την αντίστοιχη σελίδα που βλέπει ο χρήστης στην εφαρμογή.

ΕΠΙΛΟΓΟΣ

Αυτή ήταν η αναφορά μας για την εργασία μας στο πλαίσιο του μαθήματος Τεχνολογίες Εφαρμογών Διαδικτύου. Υπήρχαν κάποιες δυσκολίες που καθυστέρησαν τον αρχικό μας χρονοπρογραμματισμό για την υλοποίηση της εργασίας, όπως η κρυπτογράφηση των https αιτήσεων και η εξοικείωση μας με τις τεχνολογίες που χρησιμοποιήσαμε τις οποίες δεν είχαμε χρησιμοποιήσει στο παρελθόν. Ωστόσο θεωρούμε πως έχουμε καλύψει πλήρως όλες τις απαιτήσεις της εκφώνησης (συμπεριλαμβανομένου το BONUS) και η εφαρμογή μας λειτουργεί με τον επιθυμητό τρόπο.

Ελπίζουμε να τα καλύψαμε όλα, και σας ευχαριστούμε πολύ για τον χρόνο σας!

Οι συντάκτες της εργασίας,

Δημήτριος Χρυσός και Αναστάσιος Μουμουλίδης