Τεχνολογίες Εφαρμογών Διαδικτύου

Προγραμματιστική Εργασία 2024 : Εφαρμογή Επαγγελματικής Δικτύωσης

Μάγου Ειρηναίος 1115201700208

Περιεχόμενα

Εισαγωγή

Η εργασία έχει ώς στόχο την ανάπτυξη εφαρμογής επαγγελματικής δικτύωσης (τύπου LinkedIn) , στην οποία οι χρήστες θα έχουν πρόσβαση μέσω σύγχρονου φυλλομετρητή παγκόσμιου ιστού. Πρώτα περιγράφεται η δομή της εργασίας , οι γλώσσες προγραμματισμού , καθώς και τα εργαλεία που χρησιμοποιούνται για να τρέχουν και να επικοινωνούν μεταξύ τους οι διακομιστές των δύο άκρων. Έπειτα εξηγείται η λογική και η υλοποίηση του κώδικα των δύο άκρων για κάθε ζητούμενο της εργασίας και δίνονται παραδοχές για συγκεκριμένες σχεδιαστικές αποφάσεις. Τέλος γίνεται μια σύνοψη της εργασίας

Εργαλεία και Δομή

Για την υλοποίηση της εργασίας δημιουργήθηκαν σε περιβάλλον unix με τη βοήθεια WSL(Windows substyme for Linux) βάση δεδομένων με SQLite3 και το νωτιαίο άκρο της εφαρμογής με FastAPI της python. Σε περιβάλλον Windows το μετωπιαίο άκρο με τη βοήθεια ReactJS.

Νωτιαίο Άκρο

Η επιλογή του FastAPI έγινε λόγω της εύκολης χρήσης και κατανόησης της γλώσσας , του καλού documentation , του UI που προσφέρει (Swagger) και των υψηλών επιδόσεων. Το UVICORN είναι μια υλοποίηση διακομιστή ιστού ASGI (Asynchronous Server Gateway Interface) προσαρμοσμένη για python και εξαιρετικά χρήσιμο με FastAPI. Έγινε εγκατάσταση των βιβλιοθηκών fastapi και uvicorn.

```
pip install fastapi
pip install uvicorn
```

Για να τεθεί ο διακομιστής σε λειτουργία , εκτελείται η παρακάτω εντολή

```
uvicron main:app --reload
```

- main : το όνομα του αρχείου Python όπου η ASGI εφαρμογή ορίζεται(χωρίς το main.py).
- app : η μεταβλητή της instance ASGI εφαρμογής.
- reload : ο διακομιστής ενημερώνεται κάθε φορα που ανιχνέυει αλλαγές στα πηγαία αρχεία που συμπεριλαβάνονται στην εφαρμογή και προσφέρει περισσότερες λειτουργίες εντοπισμού σφαλμάτων.

Για να δουλέψει σωστά η εντολή , πρέπει να περιγοιηθείς στο directory που περιέχει το αρχείο main.py το οποίο πρέπει σίγουρα να περιέχει

```
    from fastapi import FASTAPI
    app = FASTAPI()
```

Ο κατάλογος για το νωτιαίο άκρο περιέχει τα αρχεία :

- main.py
- dependencies.py
- core.db (βάση δεδομένων)

Καθως και τους ακόλουθους καταλόγους :

κατάλογος	**purpose**
classes	Κλάσεις για συναρτήσεις και αντικείμενα
models	Μοντέλα JSON για επικοινωνία με μετωπιαίο άκρο
queries	Ερωτήματα SQL για επικοινωνία με τη βάση
validation	Validatiion JWT
media	Αποθήκευση των άρθρων και αγγελιών
pictures	Εικόνες Προφίλ των χτηστών
recommendations	Το σύστηαμ συστάσεων
### Μετωπιαίο Άκρο	
Η ReactJS προσφέρει έναν κατανοητό και οργανωμένο τρόπο γραφής UI με τη βοήθεια επαναχρησιμοποιούμενων components.Επίσης περιέχει extensions για την χρήση HTML στα ίδια αρχεία.	
Έγινε εγκατάσταση node.js από τη σελίδα τους , το οποίο έρχεται με Node Package Manager(npm)	
Μετά δημιουργείται ο κατάλογος με τα απαραίτητα για την εκκίνηση ενός project με τις εντολές:	

Δημιουργία :

```
npx create-react-app my-react-app
```

```
cd my-react-app
```

Εκκίνηση :

```
npm run dev
```

Η εντολή εκκίνησης ψάχνει το index.html αρχείο που υπάρχει στον εκάστοτε κατάλογο και διαβάζει τα που οδηγονυ στο entry point της εφαρμογης.

```
<body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
</body>
```

Ο κατάλογος src για το μετωπιαίο άκρο περιέχει τα αρχεία :

- App.js
- contextProvider.js (για διαμοιρασμό states σε όλα τα component)
- main.js
- myButton.jsx , Pictures.jsx , dropzone.jsx , debounce.jsx (components)

Καθώς και τους καταλόγους

Βάση Δεδομένων

SQLite3 μια απλή βάση κατάλληλη για μικρα project. Υπάρχει ένα αρχείο test.sql στο νωτιαίο άκρο της εφαρμογής στο οποίο υπάρχουν SQL queries για τη δημιουργία και διαχείριση των πινάκων της βάσης. Οι πίνακες που χρειάστηκα είναι οι παρακάτω : --users και personal_info και user_picture για τους χρήστες. --connections για τις συνδέσεις των χρηστών --chats και messages για τις συζητήσεις --article_categories , uploaded_articles , article_comments , article_interests και interactions για τα άρθρα και την αλληλεπίδραση με τους χρήστες. --uploaded_ads και ad_interest για τις αγγελίες.

Υλοποίηση Ζητουμενων

Βασικά ReactJS :

Έχει χρησιμοποιηθεί *context provider* για τον σωστό διαμοιρασμό του JWT σαν state σε διάφορα components , αφού αυτό έρθει από το νωτιαίο άρθρο .Το αρχείο *index.jsx* είναι υπέυθυνο για την πλοήγηση ανάμεσα στις σελίδες(components). Με την βοήθεια του *react-router-dom* ,δημιουργούνται τα επιθυμητά routes.Στο ίδιο αρχείο χρησιμοποιείται και ένα *layout* για τον διαμοιρασμό της μπάρας πλοήγησης στις σελίδες. Για επικονωνία με endpoints που έχουν λειτουργία POST χρησιμοποείται συχνα formData που πρόκειται για ένα API που παρέχει έναν τρόπο κατασκευής ζευγών κλειδιού/τιμής ακριβώς για τέτοιες λειτουργίες.Γίνεται χρήση *react hooks* για τη δημιουργία και διαχείριση διαφόρων states.Τα states σε React εφαρμογές είναι άκρως σημαντικά αφού είναι τρόπος να διατηρηθούν δεδομένα μεταξύ render των components , αλλά και για το ίδιο το κάλεσμα των re-render. Στα components που θεωρείται απαραίτητο υπάρχουν *states* για error και *messages* που έχουν να κάνουν με την επικοινωνία μεταξύ των δύο άκρων. Όταν υπάρξει κάποιο error να γίνεται γνωστό αυτό και επίσης να ενημερώνεται ο χρήστης μέσω μηνύματος.

Επαναχρησιμοποιούμενα Components

Στον κατάλογο *myreuse* υπάρχουν components που χρησιμοποιούνται σε διάφορα σημεία της εφαρμογής.

- ***Navbar.jsx*** :Με conditional rendering δείχνει τις κατάλληλες επιλογες , ανάλογα με το αν ο χρήστης είναι συνδεδεμένος. Κάνοντας κλικ στις επιλογές , γίνονται οι πλοηγήσεις με την βοήθεια του .

- ***debounce.jsx*** :Είναι μια συνάρτηση που χρησιμοποίει την τεχνική της αποσύνδεσης για τον περιορισμό του ρυθμού εκτέλεσης μιας συνάρτησης , που ενεργοποιεί γεγονότα συχνα(π.χ onChange σε inputs)
- ***myButton.jsx*** : Custom κουμπί που δέχεται διάφρα ορίσματα ούτως ώστε να μπορεί να εξυπηρετεί λειτουργίες ανάλογα με τις ανάγκες του κάθε component που το χρησιμοποίει.
- ***drop_text*** : Με κλικ εμφανίζει πλαίσιο εισαγωγής κειμένου.
- ***dropzone.jsx*** : Δημιουργεί ένα drag 'n' drop zone με την βοήθεια 'react-dropzone' ,για να ανεβάζει αρχεία ο χρήστης.

Έχω επιλέξει να εμφανίζω τις επιλογές του χρήστη κάνοντας κλικ στο Avatar στην μπάρα πλοήγησης , παρά σε μια λίστα στα αριστερά της οθόνης.

1 & 2

Ένας χρήστης που δεν έχει JWT κατά την πλοήγηση του στην κεντρική('/') η σελίδα που βλέπει είναι το login . Από την μπάρα πλοήγησης του δίνεται η επιλογή Εγγραφής στην εφαρμογή.

- **Login** :
- - *SignIn.jsx* : Επιστρέφει μια HTML φόρμα για τη σύνδεση του χρήστη. Σε περίπτωση επιτυχούς υποβολής , επιστρέφεται ένα JWT , το οποίο και φυλάσσεται τοπικά ,ενημερώνει το αντίστιχο state του context provider και ο χρήστης μεταβιβάζεται στην αρχική σελίδα.Αλλιώς παρουσιάζεται το μήνυμα λάθους που ελήφθη από το νωτιαίο άκρο.
- -Απ'την μεριά του νωτιαίο χρησιμοποιείται το endpoint */token* και η κλάσση *SignIn()* του αρχείου *signIn.py*. Λαμβάνει την φόρμα και τρέχει queries στη βάση για το email και έπειτα για τον κωδικό. Στην περίπτωση επιτυχίας δημιουργεί ένα JWT και το επιστρέφει στον χρήστη.
- **SignUp** :
- *-SignUp.jsx*: Επιστρέφει μια HTML φόρμα από την οποία ο χρήστης μπορεί να εγγραφεί. Γίνονται οι απαραίτητοι ελέγχοι στη φόρμα και σε περίπτωση επιτύχιας , επιστρέφεται ένα JWT , το οποίο και φυλάσσεται τοπικά ,ενημερώνει το αντίστιχο state του context provider και ο χρήστης μεταβιβάζεται στην επόμενη σελίδα για να ανεβάσει τα προσωπικά του στοιχεία.
- *-insert_pictures.jsx*:Επιστρέφει μια είσοδο αρχείου και όταν γίνει η επιλογή του χρήστη.
- -Απ'την μεριά του νωτιαίο χρησιμοποιείται το endpoint */create_user/* και η κλάση *NewUser()* του αρχείου *new_user.py*. Λαμβάνει την φόρμα και τρέχει queries στη βάση , πρώτα για έλεγχο αν υπάρχει ήδη το email(αν υπάρχει ενημερώνει το μετωπιαίο άκρο) και έπειτα για τη δημιουργία του χρήστη. Παίρνει την τελευταία είσοδο από τη βάση(τον καινούργιο χρήστη) και δημιουργεί ένα JWT και το στέλνει σαν απάντηση.

3 & 4

Ο Admin έχει τη δυνατότητα να κάνει ακριβώς όσα ζητάει η εκφώνηση. Κατά την είσοδο του , εμφανίζεται μια λίστα με το όνομα και διεύθυνση ηλ.ταχυδρομίου των χρηστών . Επιλέγοντας ένα χρήστη , μεταφέρεται στη σελίδα του όπου μπορεί να εξαγάγει τα δεδομένα στις μορφές JSON και XML .Τα δεδομένα του χρήστη είναι όλα ,ανεξαρτήτου απόκρυψης.Οι κατάλογοι με το όνομα admin και στα δύο άκρα συνεργάζονται για τις λειτουργίες.

5

Αρχική Σελίδα

--***WelcomePage.jsx*** : Επιστρέφει τα άρθρα σε χρονική σειρά και τα παρουσιάζει σε 3 στήλες , χρήστη-συνδέσεις , ενδιαφέρον από συνδέσεις χρήστη και από σο σύστημα συστάσεων. ***Για τα άρθρα*** : Υπάρχουν πέντε αρχεία στον κατάλογο *articles*. --***article_fun.jsx*** : Περιέχει συναρτήσεις που αφορούν το πλέγμα με τα άρθρα ArticleGrid() , τα άρθρα που βρίσκονται στα κελιά του πλέγματος GridElem() , την προβολή ολόκληρου το άρθρου ArticleView() και τέλος τη συνάρτηση που ευθύνεται για το ενδιαφέρον του χρήστη σε ένα άρθρο Interest().Για το σχόλιο άρθρου χρησιμποιείται από το myreuse το drop_text.jsx. --Η αρχική σελίδα κάνει render το ConnectedArticles()(αρχείο article_grid). Αυτό το component επικοινωνεί με το endpoint */view_articles* και κάνει με τη σειρά του render τα άρθρα με τη χρήση της ArticleGrid().Έπειτα η ArticleGrid() κάνει render το κάθε άρθρο με τη βοήθεια της GridElem(). Ο χρήστης κάνοντας κλικ σε ένα άρθρο , μπορεί να μεταβεί στο component FullArticle() και να διαβάσει ολόκληρο το άρθρο.Το component αυτό επικοινωνεί με το endpoint */full_article* .Το MediaViewer() , αναγνωρίζει τον τύπο του ληφθέν πολυμέσο και το κάνει render.Ο UploadArticle() επιστρέφει μια HTML φορμα για εισαγωγή και ανέβασμα του άρθρου. Υπάρχει επιλογή προβολής των άρθρων του χρήστη για επεξεργασία τους μέσω του manage_article.jsx. Στο νωτιαίο άρθρο υπάρχει η κλάση Articles() που αναλαμβάνει όλες τις παραπάνω λειτουργίες.

6

Δίκτυο

Στον κατάλογο Network υπάρχουν πέντε αρχεία με την ίδια λογική υλοποιήσης που έχουν τα άρθρα.Στο user_profile.jsx είναι τα components που δημιουργούν το πλέγμα UserGrid() , τα κελιά στο πλέγμα ProfileGrid() , και ολόκληρη τη σελίδα του χρήστη UserProfile().Το Network() component φέρνει τις συνδέσεις του χρήστη και αναλαμβάνει την αναζήτηση χρηστών. Κάνει render πλαίσιο εισαγωγης κειμένου(SearchBar() από search_conn.jsx)και με conditional rendering δείχνει τις συνδέσεις του χρήστη ή τα αποτελέσματα αναζήτησης.Στο νωτιαίο άκρο οι κλάσεις connections.py και users.py περιέχουν τις κατάλληλες λειτουργίες για την υλοποίηση του δικτύου.

7

Αγγελίες

Ίδια λογική με τα άρθρα , χωρίς αρχείο για πολυμέσα. Οι συναρτήσεις στους καταλόγους ads και στα δύο άκρα , έχουν τις ίδιες λειτουργίες με τα άρθρα.

8

Συζητήσεις

Ανάλογα με τον τρόπο πλοήγησης προς τις συζητήσεις , γίνεται render και η κατάλληλη συζητηση. Κάνοντας κλικ στην καρτέλα συζητήσεις , τότε εμφανίζεται η τελευταία συζήτηση , ενώ πηγαίνοντας από ένα προγιλ κάποιου χρήστη , εμφανίζεται η συζήτηση με τον χρήστη αυτό. Η λειτουργία στο νωτιαίο άκρο γίνεται με τη χρήση websocket και βρίσκεται στην κλάση communications.py. Το fastAPI έχει πολύ καλή καθοδήγηση για τη δημιουργία και χρήση websocket.Στο μετωπιαίο άκρο υπάρχει ο κατάλογος chats που αναλαμβάνει την προβολή συζητήσεων(sidebar) , την ανταλλαγή μηνυμάτων(window) , την πραγματική επικοινωνία μέσω websocket(communication). Αυτό το κομμάτι ήταν το πιο δύσκολο , κυρίως στο μετωπιαίο άκρο λόγω του διαμοιρασμού πολλών πληροφοριών μεταξύ των component για την ωμαλή λειτουργία των συζητήσεων.

9

Ειδοποιήσεις

Στον κατάλογο account_settings υπάρχει το αρχείο Notifications.jsx που αναλαμβάνει να φέρει και να προβάλει τυχόν υπάρχουσες ειδοποιήσεις(αιτήματα και αλληλεπίδραση σε άρθρα από άλλους χρήστες)

10 & 11

Ρυθμίσεις και Προσωπικά Στοιχεία

Τα δύο αυτά ερωτήματα τα έχω συγχωνέυσει σαν "Ρυθμίσεις Λογαριασμού" και ο χρήστης μπορεί από την ίδια σελίδα να κάνει τις ζητούμενες λειτουργίες. Τα αρχεία αυτα βρίσκονται στον κατάλογο account_settings. Στο νωτιαίο άκρο γίνεται χρήση της κλάσσης personal_info.py αλλά changes.py.

12

Σύστημα συστάσεων

Στον κατάλογο recommednations βρίσκεται ένα αρχείο fill_database.py ,το οποίο με την βοήθεια sqlalchemy δημιουργεί τα μοντέλα(πίνακες) της βάσης και έπειτα τους γεμίζει με τυχαίο τρόπο. Η σύνδεση που κάνω μεταξύ των άρθρων είναι ανάλογα με τον τύπο/κατηγορία τους. Στη συνάρτηση populate_db ,ημιουργώ 7 λίστες , όσες και οι κατηγορίες. Για κάθε χρήστη επιλέγω τυχαία ένα αριθμό κατηγοριών που θα του "αρέσουν" και τυχαία επιλέγω λίστες/κατηγορίες που θα συμπεριλαμβάνεται ανάλογα με τον αριθμό κατηγοριών. Τα άρθρα μόλις τυχαία επιλέξουν την κατηγορία τους , ο χρήστης επιλέγεται από τη συγκεκριμένη λίστα. Το ίδιο συμβαίνει και με τα Σχόλια και Ενδιαφέρον. Μόλις επιλεγεί το άρθρο , κάποιοι χρήστες που είναι στη λίστα ενδιαφέροντος επιλέγονται τυχαία για να κάνουν interact μς το άρθρο.

- Το αρχείο train.py καλείται μετά το γέμισμα της βάσης και η εκπαίδευση γίνεται με Stochastic Gradient Descent :
- Υπάρχουν δύο βοηθητικές συναρτήσεις : η compute_mse_loss που είναι για τον υπολογισμο του Mean Squarred Error κατά τη διαδικασία της εκπαίδευσης και η compute_metrics που δίνει τις μετρικές μετά την εκπαίδευση.
- Φέρνει τα κατάλληλα δεδομένα από τη βαση (interactions ,articles και users) και δημιουργείται ο "pivot table" μεταξύ χρηστών και άρθρων.
- Ο πίνακας χωρίζεται σε user_matrix και article_matrix. ο category_matrix είναι για τις κατηγορίες των άρθρων.
- Ακολουθεί το loop της εκπαίδευσης: Το μοντέλο προβλέπει τις τιμές αλληλεπίδρασης ως το dot product των λανθανόντων παραγόντων χρήστη και άρθρου, με μια πρόσθετη συνεισφορά από την κατηγορία του άρθρου. Έπειτα υπολογίζει το error και με gradient descent ενημερώνονται κατάλληλα οι πίνακες. Τέλος υπολογίζεται το MSE.
- Μετά την εκπαίδευση, δημιουργείται ένας πίνακας αλληλεπίδρασης με πολλαπλασιασμό των πινάκων χρήστη και αντικειμένου και για κάθε χρήστη, τα άρθρα με τα οποία δεν έχει αλληλεπιδράσει κατατάσσονται με βάση τις προβλεπόμενες τιμές αλληλεπίδρασης και τα 5 κορυφαία άρθρα επιλέγονται ως συστάσεις.
- Υπολογίζονται οι μετρικές και αποθηκεύονται οι πίνακες για να χρησιμοποιηθούν για σύσταση , όταν ένας χρήστης συνδεθεί στην εφαρμογή.

Για την εφαρμογή

- Ο κωδικός για όλους τους χρήστες είναι: passWord123
- Τα στοιχεία του admin είναι : email : admin@mail.com password : @dmin1234
  Τρέχοντας το αρχείο profile_pics που βρίσκεται στον κατάλογο recommendations , θα γεμίσει ο κατάλαγος που είναι για τις φωτογραφίες των χρηστών.
-
  - Βιβλιοθήκες ** :
- JavaScript : react-rputer-dom , mui-material
- python : numpy ,sqlite3 ,asyncio