ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Εφαρμογή Ηλεκτρονικών Δημοπρασιών στον Παγκόσμιο Ιστό

(για το μάθημα “Τεχνολογίες Εφαρμογών Διαδικτύου)

Μέλη Ομάδας:
Λινάρδος Αλέξανδρος 1115201600093
Μήλιος Αντώνιος 1115201600100

Διδάσκων:
Δρ. Γ. Χαμόδρακας

Ακαδ. Έτος 2021-2022

Περιεχόμενα

Εισαγωγή

Σκοπός Εργασίας

Στόχος αυτής της εργασίας είναι η ανάπτυξη εφαρμογής ηλεκτρονικών δημοπρασιών. Η εφαρμογή αυτή υλοποιήθηκε βάσει της αρχιτεκτονικής web browser / web server και κάθε χρήστης έχει πρόσβαση σε αυτή μέσω του φυλλομετρητή παγκόσμιου ιστού.

Στην εφαρμογή υπάρχουν 4 ρόλοι: Διαχειριστής Εφαρμογής / Πωλητής / Προσφοροδότης (bidder) / Επισκέπτης. Κάθε άτομο που εγγράφεται σε αυτή την

εφαρμογή μπορεί να θέσει με το ρόλο του Πωλητή ένα ή περισσότερα αντικείμενα σε δημοπρασία ή με το ρόλο του Προσφοροδότη να πλοηγηθεί / να αναζητήσει αντικείμενα που έχουν θέσει σε δημοπρασία άλλοι χρήστες προκειμένου να υποβάλλει προσφορές. Οι χρήστες που εισέρχονται στην υπηρεσία με το ρόλο του επισκέπτη μπορούν μόνο να πλοηγηθούν / να αναζητήσουν αντικείμενα που έχουν τεθεί σε δημοπρασία χωρίς να μπορούν να υποβάλλουν προσφορές.

Επιλογές Υλοποίησης

Το νωτιαίο άκρο της εφαρμογής (Backend) έχει υλοποιηθεί με **Django version 4.1.1** και κατ' επέκταση με την προγραμματιστική διεπαφή **Django REST framework**. Το μετωπιαίο άκρο της εφαρμογής έχει υλοποιηθεί με **React Javascript Web framework**.

Created superuser:
dimitris
1234

Dependencies and Build

Dependencies

Python/Django:

- django
- djangorestframework
- djangorestframework-simplejwt
- pyOpenSSL
- pillow
- django-cors-headers
- django-extensions
- django-extensions Werkzeug
- django-sslserver
- sklearn
- numpy
- beautifulsoup4
- lxml

React/Node.js:

- axios
- jwt-decode

- react-axios
- react-bootstrap
- react-countdown
- react-dom
- react-redux
- react-router-bootstrap
- react-router-dom
- redux
- redux-devtools-extension
- redux-thunk
- react-select
- crypto-js
- date-fns –save
- file-saver –save

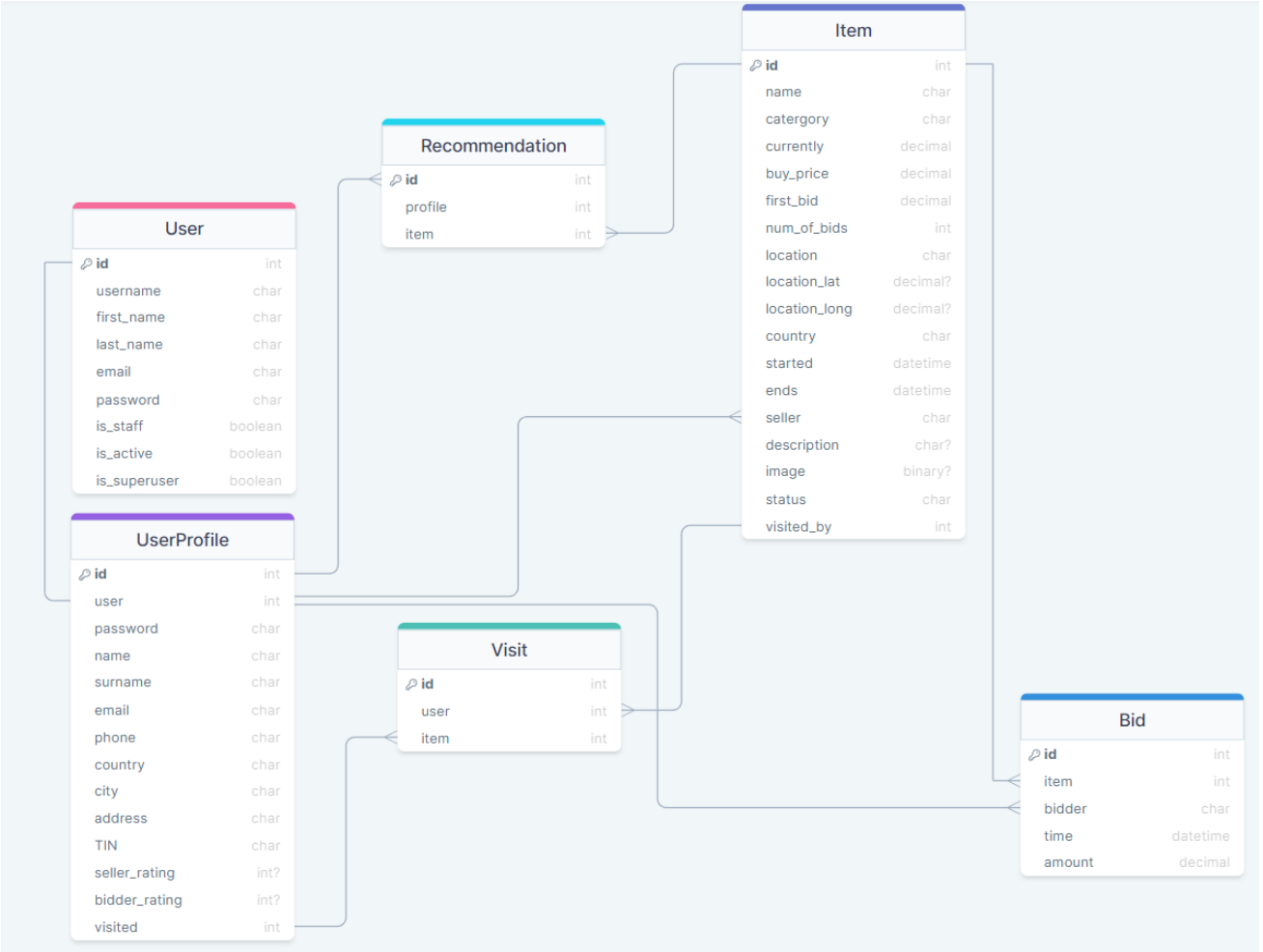
Build

Για την εκκίνηση του backend της εφαρμογής οδηγηθείτε εντός του directory backend και εκτελέστε :
python manage.py runserver_plus --cert-file cert.pem --key-file key.pem

Για την εκκίνηση του frontend της εφαρμογής οδηγηθείτε εντός του directory frontend και εκτελέστε:
npm start

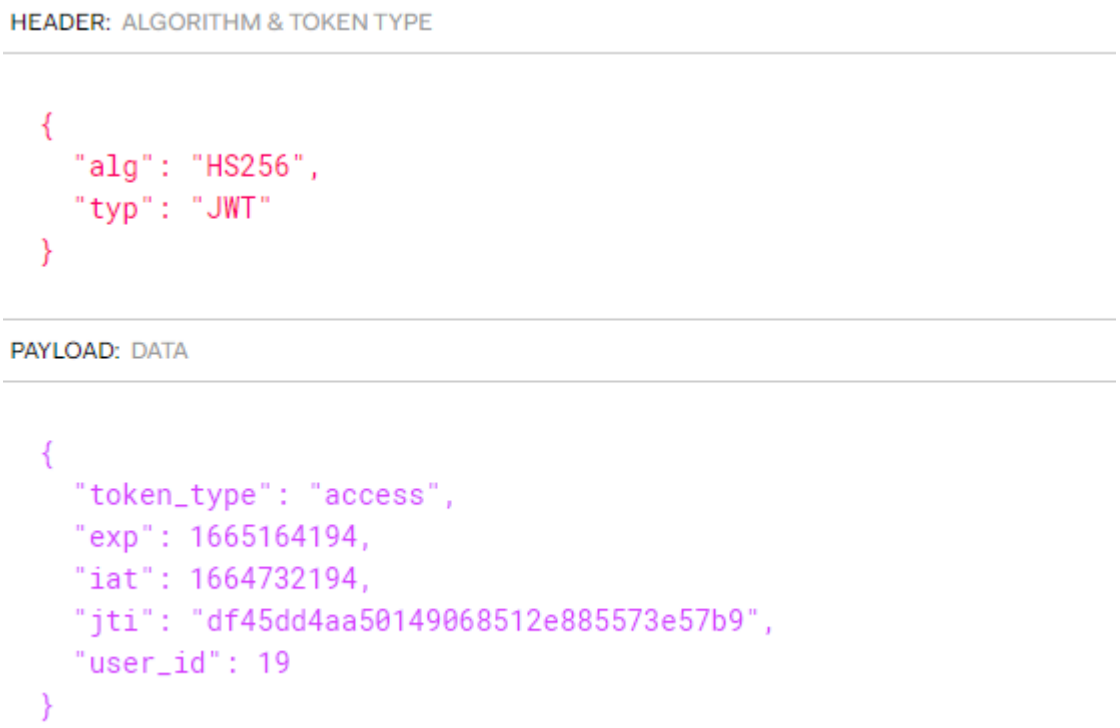
Παρουσίαση Βάσης Δεδομένων

Ο σχεδιασμός της Βάσης Δεδομένων μας παρουσιάζεται από το παρακάτω σχεσιακό διάγραμμα:


Item
id int
name char
catergory char
currently decimal
buy_price decimal
first_bid decimal
num_of_bids int
location char
location_lat decimal?
location_long decimal?
country char
started datetime
ends datetime
seller char
description char?
image binary?
status char
visited_by int
Recommendation
id int
profile int
item int
User
id int
username char
first_name char
last_name char
email char
password char
is_staff boolean
is_active boolean
is_superuser boolean
UserProfile
id int
user int
password char
name char
surname char
email char
phone char
country char
city char
address char
TIN char
seller_rating int?
bidder_rating int?
visited int
Visit
id int
user int
item int
Bid
id int
item int
bidder char
time datetime
amount decimal

Ασφάλεια Συναλλαγών Ταυτοποίησης

Για την πιο ασφαλή ταυτοποίηση των χρηστών της εφαρμογής έχουν χρησιμοποιηθεί JSON Web Tokens με τη παρακάτω μορφή:



Επιπλέον, οι συναλλαγές κρυπτογραφούνται μέσω του πρωτοκόλλου SSL/TLS χάρη στο εργαλίο mkcert (https://github.com/FiloSottile/mkcert). Τέλος, ο κωδικός του χρήστη κρυπτογραφείται με MD5 και στη συνέχεια με Base64 από το frontend έτσι ώστε να μην εμφανίζεται αυτούσιος στο payload του request. Φτάνοντας στο backend η Django αναλαμβάνει να κάνει αυτόματα και την default κρυπτογράφηση κατά την αποθήκευση στη βάση δεδομένων.

Παρουσίαση Εφαρμογής

Ο χρήστης που δεν έχει συνδεθεί σε λογαριασμό, κατευθύνεται στο welcome page. Έχει την επιλογή να συνεχίσει ως guest. Σε αυτή τη περίπτωση έχει μόνο τη δυνατότητα να πλοηγηθεί στις ενεργές δημοπρασίες και να παρακολουθήσει τις προσφορές καθώς και να αναζητήσει περιεχόμενο. Οποιαδήποτε άλλη υπηρεσία θα τον παραπέμψει να συνδεθεί πρώτα. Ο εγγεγραμμένος χρήστης μετά τη σύνδεσή του ανακατευθύνεται στην αρχική σελίδα όπου βλέπει

1. ένα carousel με τα hot items (τις δημοπρασίες που έχουν τα μεγαλύτερα number_of_bids
2. αντικείμενα ενεργών δημοπρασιών σε σελιδοποιημένη μορφή με σελιδοποίηση 4.
3. αν υπάρχουν, εως 4 αντικείμενα που του έχουν προταθεί σύμφωνα με το recommendation system που έχουμε υλοποιήσει. (ο αλγοριθμος των recommendation τρεχει κάθε φορά που τρέχουμε εντολή manage.py όπως πχ κατα την εκινηση του backend server)

Πατώντας πάνω σε κάποιο αντικείμενο θα δει τις λεπτομέρειες της δημοπρασίας όπου ανάλογα με τη δημοπρασία μπορεί να

1. υποβάλει την 1η προσφορα (starting bid)
2. να υποβάλει προσφορά με ελεγχόμενο input χρησιμοποιώντας το σχετικό slider.
3. να αγοράσει επιτόπου το αντικείμενο αλλάζωντας την κατάστασή του σε Concluded

Μια concluded δημοπρασία δεν εμφανίζεται πλεον στην πλοήγηση ουτε κατα την αναζητηση. Και εμφανίζεται στη σελίδα Won Auctions του νικητή.

Matrix Factorization

Γενικά

Για την παρουσίαση προτεινόμενων δημοπρασιών βάση του ιστορικού κάθε χρήστη χρησιμοποιείται αλγόριθμος Matrix Factorization, ο οποίος έχει υλοποιηθεί εκ του μηδενός και όπως παρουσιάζεται στις διαφάνειες του μαθήματος. Ο πίνακας X περιλαμβάνει τιμές από 0 έως και 2 όπου:
0 = ο εκάστοτε χρήστης δεν έχει επισκεφθεί ποτέ την αντίστοιχη δημοπρασία
1 = ο εκάστοτε χρήστης έχει επισκεφθεί την αντίστοιχη δημοπρασία
2 = ο εκάστοτε χρήστης έχει υποβάλει προσφορά στην αντίστοιχη δημοπρασία

Μετά την εκτέλεση του ο αλγόριθμος παράγει 4 προτεινόμενες δημοπρασίες για κάθε χρήστη κατά φθίνουσα σειρά της τιμής που τους ανατέθηκε κατά την εκτέλεση. Ο αλγόριθμος εκτελείται αυτόματα κάθε φορά κατά την εκκίνηση του server.

Fine-tuning

Το fine-tuning των παραμέτρων K και learning rate του αλγορίθμου ήταν σχετικά δύσκολο καθώς ο αλγόριθμος επηρεάζεται αρκετά από το πλήθος των δεδομένων. Οι τιμές τις οποίες καταλήξαμε να αφήσουμε στον κώδικα είναι K=2 και learning_rate = 0.0002. Με αυτές τις τιμές θεωρήσαμε ότι η ακρίβεια του αλγορίθμου ήταν αποδεκτή και επίσης ο αλγόριθμος τελείωνε την εκτέλεση του σε έναν αποδεκτό χρόνο καθώς τα δεδομένα δεν είχαν μεγάλη διαστατικότητα. Σε κάθε περίπτωση ο αλγόριθμος εκτυπώνει το τελικό του RMSE καθώς και το τελικό MAE, επομένως παρατηρώντας τις τιμές και κάνοντας αρκετές δοκιμές ίσως θα μπορούσαμε να πετύχουμε και κάτι καλύτερο.

Επίλογος

Μία από τις δυσκολίες που συναντούσαμε τακτικά κατά την υλοποίηση της εργασίας ήταν κατά την ενοποίηση των branches μας στο Git και γενικότερα του συγχρονισμού του έργου μας. Λόγω της φύσης του project έπρεπε ο καθένας μας να δουλεύει με διαφορετικές βάσεις δεδομένων και άλλα ασυγχρόνιστα αρχεία καθώς σε πολλά αρχεία του project αλλαγές και conflicts μεταξύ των εκδόσεων μας δεν μας επέτρεπαν τον συγχρονισμό τους.
Όσον αφορά το κομμάτι του Matrix Factorization εν τέλει δεν δυσκολευτήκαμε τόσο να υλοποιήσουμε τον ίδιο τον αλγόριθμο (όπως περιμέναμε) αλλά περισσότερο μας προκάλεσε το σχεδιαστικό κομμάτι της συλλογή και αποθήκευσης των απαραίτητων δεδομένων στα οποία εφαρμόζεται ο αλγόριθμος.