



Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Τεχνολογίες Εφαρμογών Διαδικτύου

ΕΡΓΑΣΙΑ ΣΕΠΤΕΜΒΡΙΟΥ

ΣΕΠΤΕΜΒΡΙΟΣ 2022

ΟΜΑΔΑ 41

Στέτσικας Ελευθέριος - 1115201700150

Μπέρος Δημήτριος - 1115201600269

Μηλίτσης Πλάτων - 1115201700087

ΠΕΡΙΕΧΟΜΕΝΑ

Εισαγωγικό Κεφάλαιο	2
Οδηγίες εγκατάστασης	3
Οδηγίες εκτέλεσης	3
Https	4
Front End	4
Welcome Page	4
Register Page	5
Home Page	6
Announcement	6
Categories	7
Recommendations	7
NavBar	8
AccountModal	8
Messages	8
Floating Button	9
SearchBar	10
Category Page(s)	10
MyBids, MyProducts, Products, Categories	11
Error, Approval	11
Admin Page	12
Item Page	13
BackEnd	14
Βάση	14
Spring Boot	15
Security	15
Controllers	15
Μνήμη και ταχύτητα	16
Επίλογος	17

Εισαγωγικό Κεφάλαιο

Η εφαρμογή μας ονομάζεται hit-it και υλοποιήθηκε με React στο Front End κομμάτι και Spring Boot στο Back End με Java17, Gradle για compile και MySql για τη βάση. Σκοπός της εργασίας ήταν να υλοποιήσουμε μια διεπαφή για δημοπρασίες, στην οποία χρήστες έχουν την δυνατότητα να υποβάλουν νέα προϊόντα για προσφορά, να υποβάλουν προσφορές σε υπάρχοντα ή απλώς να πλοηγηθούν σε αυτά. Δώσαμε έμφαση στις επιμέρους προγραμματιστικές λειτουργίες και όχι τόσο στο γραφικό περιβάλλον. Ξεκινήσαμε την εργασία ακολουθώντας το μοντέλο του skroutz για την αρχική σελίδα τουλάχιστον, γιατί μας φάνηκε αρκετά βοηθητικό προς τον χρήστη για μια τέτοια εφαρμογή, και συνεχίσαμε προσθέτοντας ότι ακόμα χρειάζεται για το μπόνους ερώτημα και το dataset.

Οδηγίες εγκατάστασης

- 1) `CREATE USER 'lst'@'localhost' IDENTIFIED BY 'Password123!';`
- 2) `GRANT CREATE, ALTER, DROP, INSERT, UPDATE, DELETE, SELECT, REFERENCES, RELOAD on *.* TO 'lst'@'localhost' WITH GRANT OPTION;`
- 3) `cd backend/hit-it/src/main/resources/data`
- 4) `mysql -u lst -p`
- 5) `Password123!`
- 6) `source hit-it.sql;`
- 7) `exit`
- 8) `./gradlew bootRun`
- 9) `cd ../../../../frontend/hit-it`
- 10) `npm install --legacy-peer-deps`

Προαπαιτούμενα είναι το nodejs και mysql. Μπορείτε να αλλάξετε ότι χρειάζεται στο `application.properties` με δικό σας χρήστη. Το πρώτο `bootRun` θα κρατήσει περίπου 70' επειδή χρησιμοποιήσαμε όλο το dataset. Τα επόμενα λίγα δευτερόλεπτα μέχρι να βρεθούν τα αρχικά `recommendations`. Την εργασία την στέλνουμε χωρίς `node_modules`, αλλά μόνο με τα `package.json` για να χρειαστεί μόνο ένα `npm install --legacy-peer-deps` για το Front End κομμάτι.

Οδηγίες εκτέλεσης

- 1) `cd backend/hit-it && ./gradlew bootRun`
- 2) `cd frontend/hit-it && HTTPS=true npm start`

Είναι απαραίτητο να τρέχει το Back End, γιατί αλλιώς θα γίνεται ανακατεύθυνση σε σελίδα λάθους, όταν απαιτείται επικοινωνία.

Έχουμε κάποιους default χρήστες με insert into στη βάση: admin, Lst, dberos, Ourt, PlasPlas, Felarxos και User7, User8 που είναι not accepted ακόμα από admin. Όλοι έχουν κωδικό 1234.

Https

Δεν δημιουργήσαμε κάποιο certificate για την ασφάλεια των προσωπικών μας δεδομένων. Θεωρούμαι ότι για τα πλαίσια της εργασίας το HTTPS=true είναι ικανοποιητικό. Η εργασία στον δρομολογητή Firefox ζητάει την έγκριση του χρήστη γιατί δεν φαίνεται πολύ έμπιστο, η εργασία είναι έμπιστη, οπότε ο χρήστης ενθαρρύνεται να συνεχίσει κανονικά χωρίς φόβο.

Front End

Welcome Page

Στην σελίδα καλωσορίσματος ο χρήστης έχει τη δυνατότητα να χρησιμοποιήσει τα αρχικά του για να κάνει τη σύνδεση, να κάνει ανακατεύθυνση σε σελίδα εγγραφής ή να συνεχίσει σαν επισκέπτης. Τους επισκέπτες δεν τους κρατάμε κάπου στη βάση οπότε έχουμε χρησιμοποιήσει στην υπόλοιπη εφαρμογή απλά fetch χωρίς auth για απλή προβολή αντικειμένων ή χρηστών που έχουν υποβάλει δημοπρασίες σε αυτά, και απλά πλοηγούνται στην εφαρμογή.

Για το login γίνεται POST σε endpoint του login με το μοναδικό username του καθενός και τον κωδικό που έχει υποβάλει. Εκεί δίνουμε roles ACCEPTED για όλους μαζί με ADMIN για τον admin που τα κρατάμε στο localStorage. Εκεί κρατάμε και τα access και refresh tokens και ένα πίνακα για το μπόνους. Για τα απαγορευμένα request έχουμε φτιάξει ένα hook το useAxiosPrivate με interceptors για να ανανεώνεται το access token πριν αποτύχει το αίτημα. Από εκεί και ύστερα, αν είναι σωστά τα αρχικά, ο admin θα κάνει ανακατεύθυνση στο admin panel και από εκεί έχει τη δυνατότητα να πάει στην αρχική, ενώ οι υπόλοιποι στην αρχική.

Register Page

Για το register είναι προαπαιτούμενα το μοναδικό username και email σε σωστή μορφή, όνομα, επώνυμο, ΑΦΜ, και κωδικός, ενώ προαιρετικά τηλέφωνο και διεύθυνση. Τα username και email πρέπει να είναι μοναδικά στη βάση μας, και πρέπει να ακολουθούνται από τα υπόλοιπα υποχρεωτικά πεδία. Υπάρχει αρκετός έλεγχος για αυτά. Η βάση μας ξέρει τους κωδικούς κρυπτογραφημένους σε μορφή πχ \$2a\$10\$vXTQrYwJmne7y8uonQgrK0QqayK670EQkn/M9xzc6cALR7votIXQ0 αντί 1234. Δε θεωρήσαμε σημαντικό ναβάλουμε κάποιο ελάχιστο αριθμό χαρακτήρων στον κωδικό ούτε regex για τη διεύθυνση η τηλέφωνο, όχι ότι θα ήταν δύσκολο κάτι τέτοιο.

hit-it

Δημιουργία Λογαριασμού

Όνομα*
User ✓

Επώνυμο*
9 ✓

Όνομα Χρήστη*
User9 ✓

Email*
user9@hit-it.com ✓

ΑΦΜ*
Εισάγετε το ΑΦΜ σας
Ο ΑΦΜ είναι υποχρεωτικός Κωδικός Πρόσβασης* ✓

Επιβεβαίωση Κωδικού*
-
Οι Κωδικοί Πρόσβασης πρέπει να ταιριάζουν ✓

Τηλέφωνο
Εισάγετε το Τηλέφωνό σας

Διεύθυνση
Εισάγετε τη Διεύθυνσή σας

Εγγραφή

Σύνδεση

Home Page

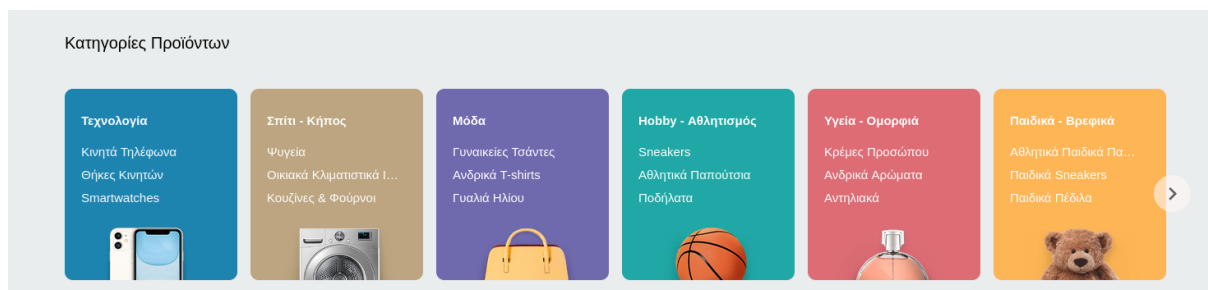
Η αρχική μας σελίδα αποτελείται από επιμέρους components.

Announcement

Μια απλή ενημέρωση προς τον χρήστη για τις ενεργές δημοπρασίες για να μπορεί να κάνει ανακατεύθυνση σε σελίδα με όλα τα Products, λόγω του μπόνους.

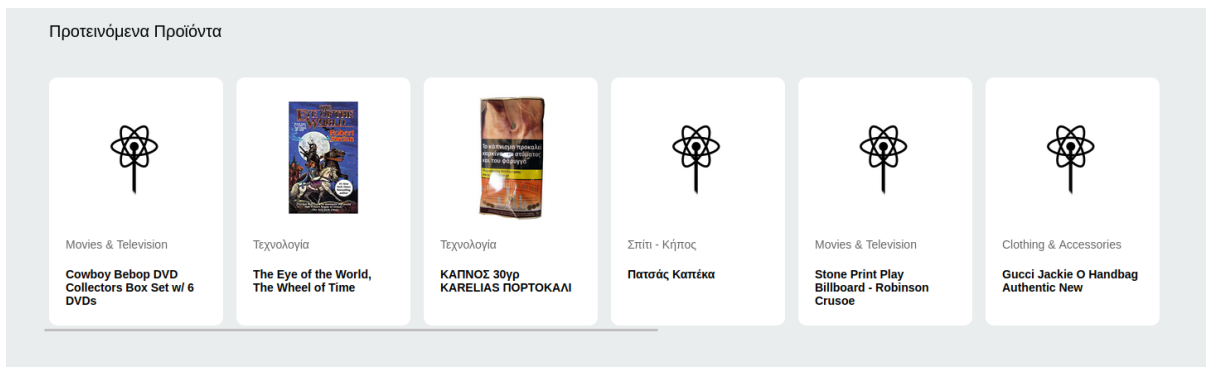
Categories

Οι default κατηγορίες μας που τις είχαμε περάσει σε ένα αρχείο, εκτός των insert into στη βάση, για να μπορούν να διαβαστούν πέραν των νέων από το dataset. Όλες τις κατηγορίες έχει τη δυνατότητα να τις βρει ο χρήστης από το `p` ακριβώς από πάνω. Έχει τη μορφή slider με τις 8 μας κατηγορίες και από εκεί γίνεται ανακατεύθυνση στη σελίδα της κάθε κατηγορίας. Hard Code κυρίως για να εξαφανίζονται τα βελάκια και να είναι ελάχιστα responsive.



Recommendations

Τα δημοφιλή προϊόντα τα βρίσκουμε όταν τρέχουμε το Back End για τους χρήστες που υπάρχουν εκείνη τη στιγμή. Όταν γίνεται νέα εγγραφή πρέπει να ξανατρεχτεί το `./gradlew bootRun`. Χρήστες που έχουν κάνει bids βρίσκουν τα δημοφιλή ανάλογα τα bids τους μεταξύ των άλλων χρηστών. Επισκέπτες και χρήστες που δεν έχουν κάνει bids, βρίσκουν τις δημοπρασίες ανάλογα των πιο δημοφιλέστερων προϊόντων και ανάλογα όσων επισκέπτονται, που τα κρατάμε στο localStorage.



NavBar

Έχουμε 2 navbars στην εργασία, ένα για το home και άλλο για τις υπόλοιπες σελίδες, επειδή στο home δε θέλαμε να βάλουμε το search στο navbar αλλά σε component από κάτω με μεγαλύτερο logo. Αν δεν είναι εγγεγραμμένος ο χρήστης θα φαίνονται επιλογές Σύνδεση και Εγγραφή. Αν είναι εγγεγραμμένος θα φαίνεται το Account Modal του με το username από κάτω. Δεν αλλάζουν και πολλά πράγματα στον κώδικα των δύο navbars.

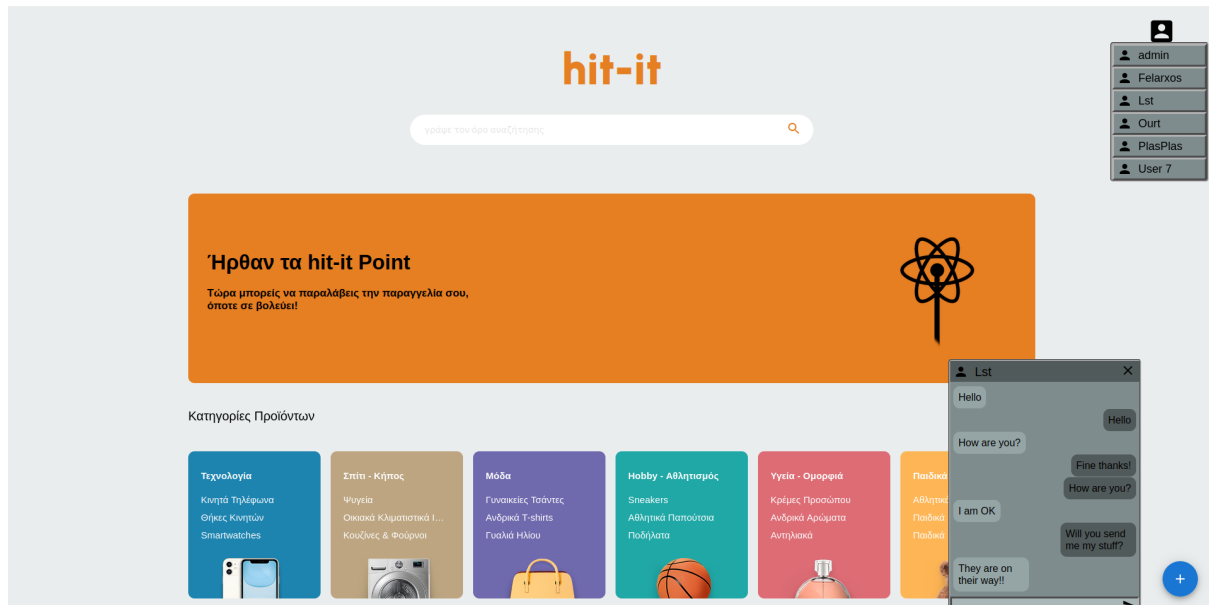
AccountModal

Private routes ή και axiosPrivate για ότι χρειάζεται. Ο χρήστης έχει την επιλογή να επιλέξει ανάμεσα στη σελίδα των προϊόντων του που έχει θέσει προς δημοπρασία, τις δημοπρασίες που έχει υποβάλει προσφορές, τα μηνύματά του και αποσύνδεση που καθαρίζεται το localStorage και γίνεται ανακατεύθυνση στο welcome.

Messages


Όταν γίνει η πώληση, ο χρήστης έχει τη δυνατότητα να προσθέσει επαφή τον πωλητή. Όλες οι επαφές αναπτύσσονται όταν επιλέξει την επιλογή των μηνυμάτων του και από εκεί μπορεί να ανοίξει ταυτόχρονα πολλές συνομιλίες, μέχρι 5 να φαίνονται και οι υπόλοιπες σε slide γιατί δε δίνουμε επιλογή minimize, μόνο close. Χρειάστηκε useContext με provider στα 2 navbar. Το modal κλείνει με κλικ απ έξω, μέσω ενός hook αλλά θέλαμε τα μηνύματα να μην κλείνουν. Οπότε σε ένα container του navbar ανοίγουν οι συνομιλίες, με fixed position στο τέλος και από εκεί επικοινωνούν τα υπόλοιπα components του messenger, αντί να πετάγεται μέσα από το modal, που μας έκλεινε. Είναι υλοποιημένες οι περισσότερες λειτουργίες με πολλά


useEffect στη σειρά, χωρίς infinite loops, όταν γίνεται το 1 async, τρέχει το επόμενο με && και το αλλαγμένο dependency.



Floating Button

Είναι το μπλε κουμπί που φαίνεται, στους εγγεγραμμένους χρήστες, όχι μόνο στο home αλλά σε όλες τις σελίδες. Από εκεί ο εγγεγραμμένος χρήστης έχει τη δυνατότητα να προσθέσει ένα νέο αντικείμενο προς προσφορά. Χρησιμοποιούμε axiosPrivate post στα απαγορευμένα endpoint και error checking για λάθη στη φόρμα. Επειδή ήταν ήδη αρκετά μεγάλη και κουραστική για τον χρήστη η φόρμα με τα απαιτούμενα πεδία, δε δίνουμε επιλογή να επιλέξει πότε θα λήξει η δημοπρασία, αλλά κάνουμε το post με το τωρινό date μέχρι για 1 χρόνο. Η φόρμα ανοίγει σε pop up και κλείνει με επιλογή απ' έξω.





Πούλα το Προϊόν σου!

Όνομα Προϊόντος*

 ✓

Περιγραφή Προϊόντος*

 ✓

Κατηγορία Προϊόντος*

Επιλέξτε μία ή περισσότερες κατηγορίες

Τεχνολογία

✓

Έναρξη Δημοπρασίας*

 ✓

Λήξη Δημοπρασίας*

 ✓

Περιοχή Κατοικίας*

Εισάγετε την περιοχή κατοικίας σας
 Η περιοχή κατοικίας είναι υποχρεωτική

Χώρα Διανομής*

Greece

✓

Γεωγραφικό Πλάτος

Εισάγετε το γεωγραφικό πλάτος της κατοικίας σας

Γεωγραφικό Μήκος

Εισάγετε το γεωγραφικό μήκος της κατοικίας σας

Εικόνα Προϊόντος

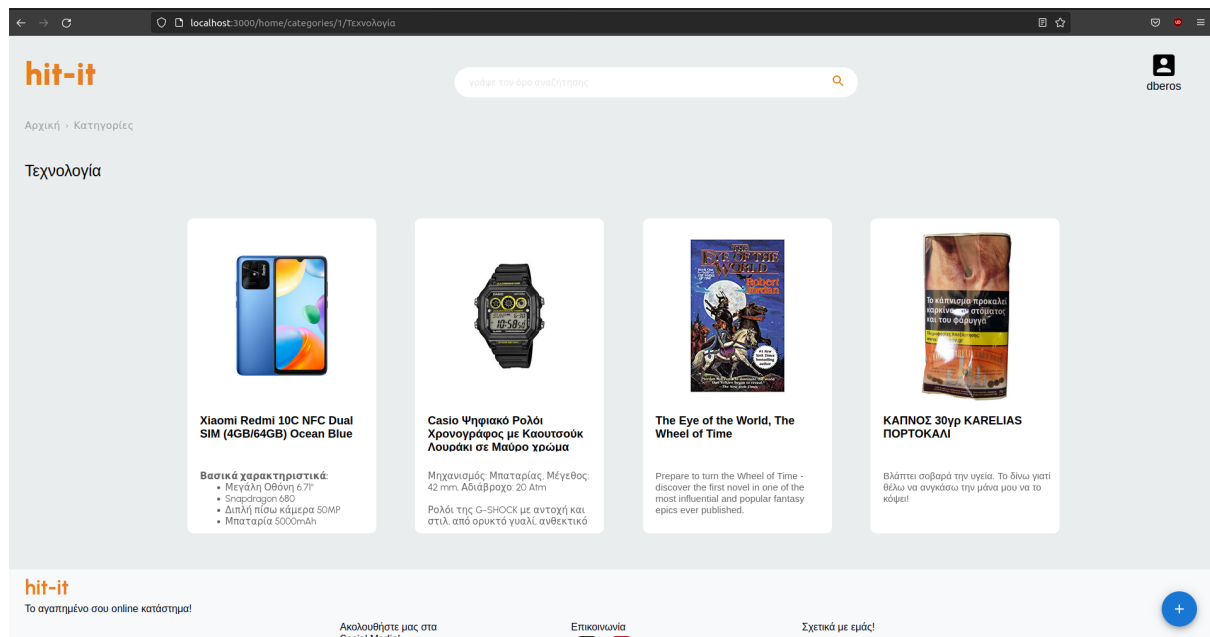
Εισάγετε το link της εικόνας του προϊόντος σας

SearchBar

Δίνουμε επιλογή στην χρήστη να μπορεί να αναζητήσει μόνο τις 8 κατηγορίες μας. Αυτό γιατί αν βάζαμε όλα τα αντικείμενα του dataset, θα καθυστερούσε πολύ, γιατί όταν υλοποιήσαμε το all products page μας για το μπόνους, με 2 fetch κάνει μισό λεπτό να φορτώσει ή και crashare!

Category Page(s)

Category Page για κάθε μία από τις 8 κατηγορίες μας και 1 Category Page για όλες τις κατηγορίες, λόγω του dataset. Το μεγάλο category page δείχνει όλες τις κατηγορίες που υπάρχουν στη βάση, ενώ το Category Page είναι για να φαίνονται τα δικά μας προϊόντα. Όλα είναι σε μορφή grid με tiles και ανάλογα το category page, γίνεται ανακατεύθυνση είτε στη σελίδα προϊόντος, είτε στην επιμέρους σελίδα κατηγορίας και από εκεί στη σελίδα προϊόντος. Ο χρήστης μπορεί να τα βρει είτε επιλέγοντας κάποια από τις κατηγορίες του slider είτε το `p` από πάνω από το slider.



MyBids, MyProducts, Products, Categories

Αρκετά όμοιες υλοποιήσεις με το category και μεταξύ τους, θα χρειάζονταν έτσι κι αλλιώς αυτά τα αρχεία, όχι να δημιουργούνταν δυναμικά τα paths όπως των categories και products. Fetch στο κατάλληλο endpoint και προβολή σε tiles μέσα σε grid.

Error, Approval

Σελίδα λάθους αν γίνει επικοινωνία με το Back End και δεν τρέχει ο server και σελίδα που ενημερώνει τον χρήστη μετά το register ότι αναμένεται έγκριση από τον διαχειριστή.

Admin Page

Welcome

Themistoklis

NON-ACCEPTED-USERS

ACCEPTED-USERS

HIT-IT


<input type="checkbox"/>	Username	First Name	Last Name	Telephone	E-mail	Address	Tin
<input type="checkbox"/>	User 7	Useros	User	231200000	felo@gmail.com	Ελβετίας 21. Αγία...	443
<input type="checkbox"/>	User 8	Useria	Useriou	231300000	felo@gmail.com	Αλωφώρος ΕΒρυκ...	442

ACCEPT

DELETE


Όταν ο διαχειριστής βάλει τα στοιχεία του κατάλληλα (admin, 1234) καθοδηγείται στο admin page. Εκεί μπορεί να δει τους χρήστες που αναμένουν αποδοχή, καθώς και να διαγράψει χρήστες ή να επιστρέψει στην εφαρμογή σαν απλός χρήστης. Στο admin page με χρήση κατάλληλου routing δεν μπορεί να εισέλθει άλλος χρήστης, πέρα από τον admin.

Item Page


Αρχική · Τεχνολογία

γράψε τον όρο αναζήτησης



admin

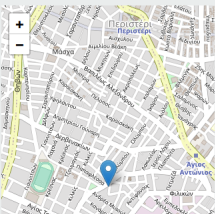


Xiaomi Redmi 10C NFC Dual SIM (4GB/64GB) Ocean Blue

Βασικά χαρακτηριστικά:

- Μεγάλη Οθόνη 6.71"
- Snapdragon 680
- Διπλή πίσω κάμερα 50MP
- Μπαταρία 5000mAh
- Γρήγορη φόρτιση 18W
- Θύρα 3.5mm για ακουστικά
- FM ραδιόφωνο
- Προστασία οθόνης με Corning Gorilla Glass

 **Πουλήθηκε:**  **dberos** : 700€



Όνομα Χρήστη	Πόντοι Δημοτικότητας	Χτυπητό Ποσό	Πρα Κρούσης
dberos	190	700€	2022-08-23 17:00
Felarios	250	500€	2022-08-23 16:55

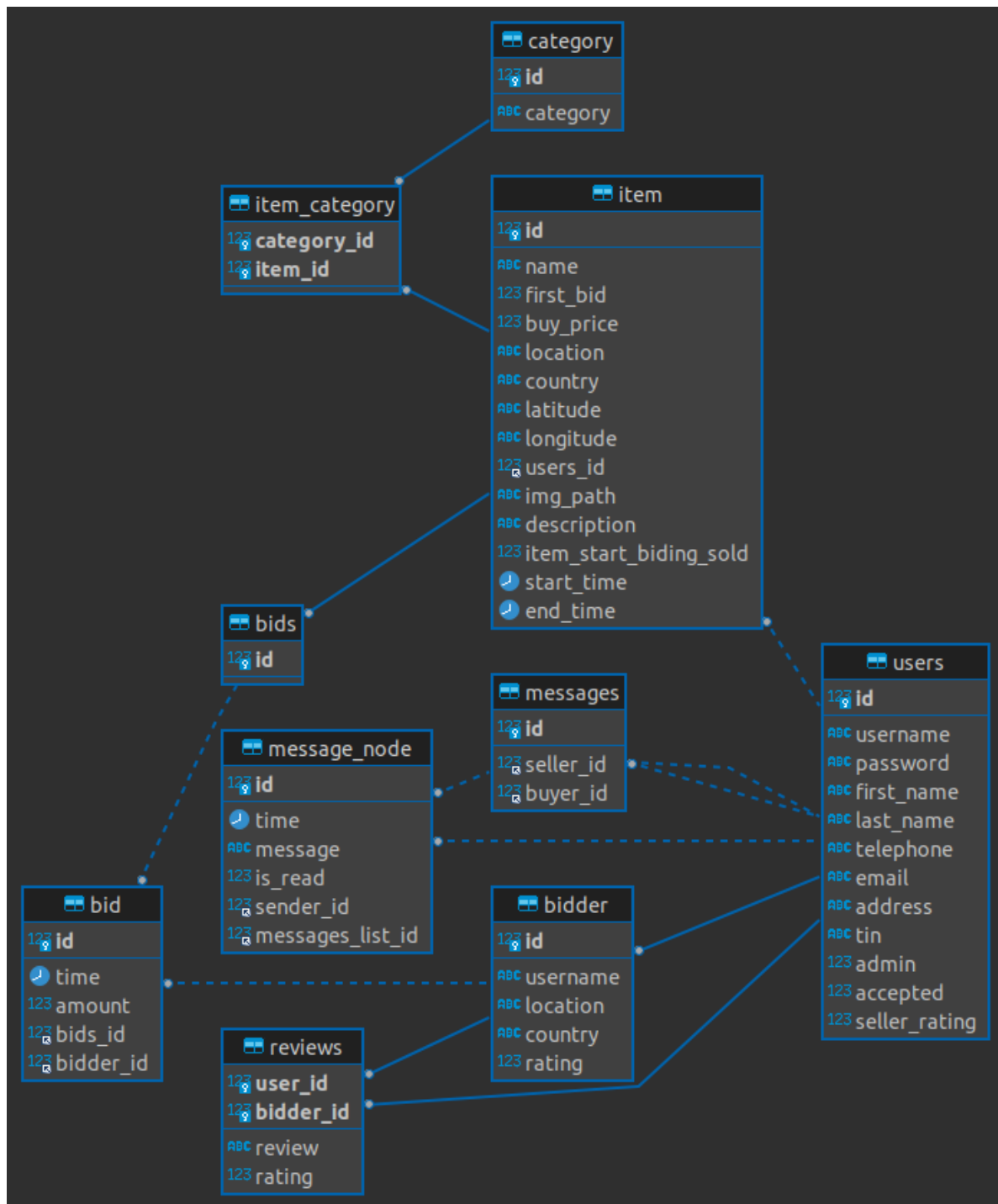
Χτύπα το με μάγκα!

Ένας χρήστης μπορεί να πλοηγηθεί και στη σελίδα του κάθε αντικειμένου. Από εκεί μπορεί να κάνει μια καινούργια δημοπρασία. Αν δεν είναι **Bidder** η εφαρμογή τον προτρέπει να γίνει. Ακόμα ανάλογα τον χρήστη εμφανίζονται διαφορετικές λειτουργίες. Αν ο χρήστης είναι αυτός που ξεκίνησε την δημοπρασία, μπορεί να αλλάξει τα χαρακτηριστικά του αντικειμένου (*κουμπί μολυβιού*) και να ξεκινήσει τη δημοπρασία (*κουμπί ευρώ*). Ο διαχειριστής μπορεί να κατεβάσει ένα αντικείμενο ως .xml (*κουμπί βελάκι προς τα κάτω*), γίνεται ανακατεύθυνση σε controller του back σε νέο tab. Στο τέλος της δημοπρασίας, στον αγοραστή και στον πωλητή εμφανίζονται δύο ακόμα λειτουργίες, να ξεκινήσουν να συνομιλούν (*κουμπί συννεφάκι*) μεταξύ τους και να στείλουν πόντους ο ένας στον άλλον (*κουμπί καρδούλας*).

BackEnd

Βάση

Η βάση υλοποιήθηκε σε mysql και μπορεί να περιγραφεί από το παρακάτω σχήμα



Spring Boot

Στην εργασία επιλέξαμε τα εργαλεία Spring Boot και Gradle για να μπορέσει να υπάρξει εύκολη εγκατάσταση των dependencies που χρησιμοποιήσαμε (jpa-hibernate, lombok, Spring Security κ.α), καθώς και η επικοινωνία Front-Back-Βάση να είναι ομαλή. Η λογική της δημιουργίας των αρχείων κώδικα ήταν να δημιουργήσουμε επίπεδα επικοινωνίας (controllers - services - repositories) και τα models αποτελούν τις αντικειμενοστραφείς κλάσεις των οντοτήτων που υπάρχουν στη βάση.

Security

Για την ασφάλεια της βάσης χρησιμοποιήθηκε το Spring Security. Οι χρήστες χωρίζονται σε USER, ACCEPTED, ADMIN. Τα endpoints που ανατίθενται σε controllers, υπάρχει έλεγχος για το ποιοι χρήστες, ανάλογα με το ρόλο τους, έχουν πρόσβαση. Τέλος, οι κωδικοί κρυπτογραφούνται με BCrypt Encoder δίνοντας μία μυστική λέξη. Δεν στέλνονται ποτέ πουθενά από το Back ούτε κρυπτογραφημένοι, ούτε κανονικά, και αποθηκεύονται στη βάση πάντα κρυπτογραφημένοι.

Controllers

Οι controllers δημιουργήθηκαν με τη λογική της REST επικοινωνίας και ταυτόχρονα με τις ανάγκες για καλύτερη εξυπηρέτηση του front. Κάποια endpoints δεν χρησιμοποιήθηκαν στην εργασία, τελικά. Η υλοποίησή τους είναι πολύ απλή και η ονομασίες τους είναι αρκετά καθοδηγητικές.

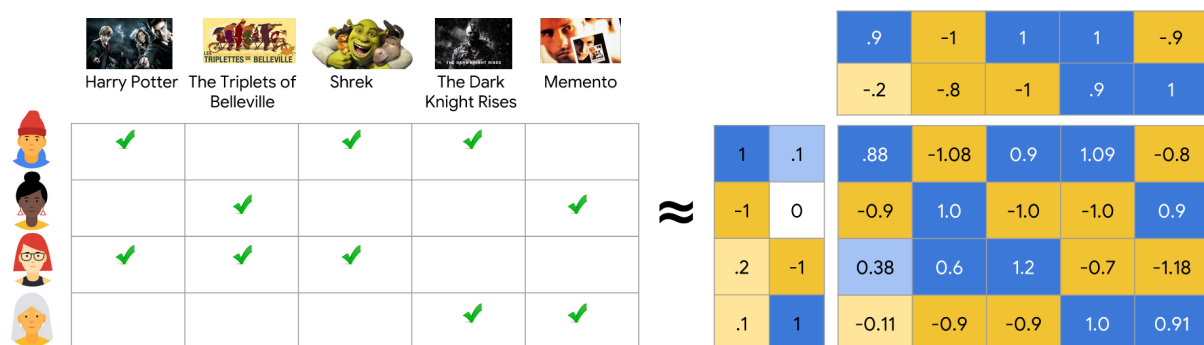
Συνηθώς:

- Get: Για να εκτελεστεί ένα απλό query στη βάση
- Post: Να δημιουργηθεί ένα καινούργιο αντικείμενο
- Put: Να αλλάξει ένα αντικείμενο κάποια στοιχεία
- Delete: Να διαγραφεί οριστικά από τη βάση

Bonus

Στο bonus υλοποιήθηκε ο αλγόριθμος matrix factorization από μνήμη για την εύρεση παρόμοιων χρηστών και από εκεί προτείνονται τα πιο δημοφιλή αντικείμενα. Αρχικά πριν ξεκινήσει η εφαρμογή δημιουργείται ένας πίνακας, και υπολογίζεται το πόσο μοιάζει ένας χρήστης με έναν άλλο χρήστη με βάση τη λογική των όμοιων

διανυσμάτων και το συνημίτονο τους. Αν δεν δημιουργηθεί ο παραπάνω πίνακας η εφαρμογή δεν προτείνει κανένα αντικείμενο. Ο πίνακας δημιουργείται με την έναρξη της εφαρμογής. Έπειτα όταν ζητηθεί από το back να προτείνει αντικείμενα, τότε βρίσκει τον πίνακα με την υπολογισμένη ομοιότητα των άλλων διανυσμάτων και επιλέγει τους 20 χρήστες που μοιάζουν περισσότερο στο χρήστη που θα γίνει η πρόταση. Έπειτα αφαιρούνται τα αντικείμενα τα οποία ο χρήστης έχει ήδη κάνει δημοπρασίες σ' αυτά και βρίσκονται τα 10 πιο δημοφιλή αντικείμενα με βάση το άθροισμα των δημοπρασιών. Θεωρούμε ότι σε ένα αντικείμενο ανεξαρτήτως των δημοπρασιών που έχει κάνει ο χρήστης, θα του δοθεί η μονάδα 1, αλλιώς αν δεν έχει κάνει 0.



Μνήμη και ταχύτητα

Κατά τη διάρκεια της υλοποίησης του Bonus, το heap της Java έσκαγε, οπότε αρχίσαμε να μειώνουμε την μνήμη που ζητάγαμε από την Java καθώς και αυξήσαμε το heap της κατά 4096 Mb.

Για να τρέξει το bonus πρέπει να φορτωθούν τα αρχεία στο φάκελο ebay-data, δυστυχώς είχαμε στο νου μας την υλοποίηση File Visitor - Parser με threads αλλά δεν μας βγήκε λόγω χρόνου. Ο χρόνος για να γίνει αυτό και για τα 30 αρχεία την πρώτη φορά, είναι κοντά στην 1 ώρα και 10 λεπτά. Μπορείτε είτε να διαγράψετε μερικά, είτε να περιμένετε, πάντως η εφαρμογή δοκιμάστηκε και με τις 13k εγγραφές.

Επίλογος

Υλοποιήσαμε μια προγραμματιστική διεπαφή υπηρεσιών REST API και προσπαθήσαμε να κάνουμε το γραφικό περιβάλλον όσο το δυνατόν μπορούσαμε καλύτερα.

Ο χρήστης έχει τη δυνατότητα να κάνει εγγραφή και σύνδεση ή να συνεχίσει σαν επισκέπτης στην εφαρμογή. Από εκεί και ύστερα μπορεί να υποβάλει νέα προϊόντα προς προσφορά ή να υποβάλει προσφορές σε υπάρχοντα, αν είναι εγγεγραμμένος, αλλιώς μόνο να αναζητήσει προϊόντα. Μπορεί επίσης να επικοινωνήσει με τον πωλητή αν αγοράσει το προϊόν του.

Δυσκολίες που συναντήσαμε αφορούν κυρίως το Front End κομμάτι, επειδή ασχοληθήκαμε πρώτη φορά με αυτό ή / και το Back End. Δε δώσαμε τόση μεγάλη έμφαση στο γραφικό περιβάλλον, αλλά θέλαμε να είναι ελάχιστα responsive και τα περισσότερα components τα υλοποιήσαμε από την αρχή τουλάχιστον 2 φορές. Μας δυσκόλεψε αρκετά το categories slider και το λύσαμε με `hardcode`, όλες οι λειτουργίες των μηνυμάτων να γίνονται όπως πρέπει και το Product Page να γίνονται επίσης οι λειτουργίες όπως πρέπει και στη σωστή σειρά. Λοιπές δυσκολίες αφορούν την επικοινωνία με το Back End και τα `asynchronous calls` όταν έπρεπε να γίνονται πολλά μαζί στη σειρά, να γεμίζουμε state arrays χωρίς infinite loops ώστε να μπορούμε να τα προβάλουμε.