Υπηρεσίες στο Υπολογιστικό Νέφος και την Ομίχλη

Αναφορά πρώτης εργασίας

Κωνσταντίνος Μυλωνάς

7 December 2022

Εισαγωγή

Στην πρώτη εργασία του μαθήματος κληθήκαμε να δημιουργήσουμε μια web εφαρμογή (CRUD operations). Ασχοληθήκαμε τόσο με το frontend οσο και με το backend κομμάτι και ετσι έχουμε πλεον μια πλήρη εικόνα του τι περιλαμβάνει ο σχεδιασμός και η υλοποίηση μιας απλής εφαρμογής με CRUD operations. Η εφαρμογή προσομοιάζει ένα site όπως το γνωστό skroutz οπου πωλητές μπορούν να ανεβάζουν τα προϊόντα τους, απλοί χρήστες να τα αγοράζουν και φυσικα οι Admins που διαχειρίζονται τους χρήστες.

Τεχνολογίες που χρησιμοποιήθηκαν

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν οι εξής τεχνολογίες:

- ReactJS για το frontend κομμάτι
- MaterialUI (reusable styled components for React)
- NodeJS για το backend κομμάτι
- ExpressJS
- MongoDB
- Docker

Ανάλυση

Έχουν υλοποιηθεί όλα τα ζητούμενα της άσκησης.

Έχει γίνει η εξής θεώρηση: Μόνο οι users μπορούν να μπουν στη σελίδα με όλα τα προϊόντα. Οι sellers ή ο admin για παράδειγμα δεν μπορούν να δουν τα προϊόντα που είναι διαθέσιμα. Οι sellers μπορούν στην αντίστοιχη σελίδα να δουν μόνο τα προϊόντα τους και να τα επεξεργαστούν.

Error Handle & Data Validation

Γενικά γίνεται data validation στις περισσότερες περιπτώσεις με χρήση του node package Joi. Ειδικά στο register ελέγχεται η ορθότητα των δεδομένων (π.χ. αν στο πεδίο e-mail εχει η τιμή ειναι τύπου name@email.com). Data validation εχουμε τοσο στον client οσο και στον server. Βέβαια όσο προχωρούσε η εφαρμογή για λόγους απλότητας παραλείφθηκε το data validation σε ορισμένες περιπτώσεις θεωρώντας πως θα γίνει “σωστή χρήση της εφαρμογής”. Σε περίπτωση πάντως που γίνει κάποιο λάθος το οποίο δεν “προλαβαίνω” με data validation τότε αυτό ανάγεται σε unexpected error και γινεται handle σε εναν γενικό handler που εχω ορίσει σε μια middleware function στον server. Εκεί γίνονται handle και πραγματικά unexpected errors οπως π.χ. αν πέσει η βάση. Σε τέτοια περίπτωση μετά από το timeout εμφανίζεται στον client ειδοποίηση για unexpected error και γίνεται revert του view στο προηγούμενο state. Το timeout εχει τεθεί στα 4 sec αντι των default 30 ωστε να μπορεί αν γίνει πιο γρήγορα προσομοίωση τέτοιων erros σε περίπτωση που θέλετε να τα επιβεβαιώσετε

React reusable components

Το React δινει τη δυνατότητα για χρήση reusable components. Για παράδειγμα το component που χρησιμοποιείται για να δείξουμε στον User τα Products είναι ιδιο με το component που χρησιμοποιείται για να δει ο Seller τα δικα του Products. Ενα αρχείο είναι “Product.jsx” στο οποίο γίνεται conditional rendering ανάλογα με το αν πρόκειται για user ή για seller και ετσι αλλάζουν τα κουμπιά (user-> Add to Cart, seller-> Edit, Remove)

Cart

Σχετικά με τη σελίδα του Cart, είναι προσβάσιμη μόνο απο τον εκαστοτε user. Στη βάση είναι αποθηκευμένο καθε cart με references στον user-owner και στα products που εχει προσθέσει. Επειδή χρησμιοποιήθηκε monogdb χρειάστηκε λίγο παραπάνω δουλειά για να πετύχουμε consistency στα δεδομένα μιας και απο πλευράς της βάσης δεν υπάρχει ελεγχος για το αν ενα reference πχ σε κάποιον user ανταποκρίνεται στη πραγματικότητα. Μπορεί αυτός ο user να μην υπάρχει καν. Έτσι φροντίσα ώστε όταν διαγράφεται ένας user να γίνεται αντίστοιχα διαγραφή και του cart του, οταν διαγράφεται ενα προϊόν να διαγράφεται και απο ολα τα carts στα οποία εχει προστεθεί και το ιδιο οταν διαγράφεται ενας seller που σημαίνει οτι διαγράφονται και τα products του.

Στη μεριά του client ένα προϊόν μπορεί να προστεθεί πολλές φορές στο καλάθι και αυξάνεται η ποσότητα (quantity) στο καλάθι όπως σε ολες τις συγρονες εφαρμογές.

Υπάρχει ειδικό κουμπί για διαγραφή όλων των προϊόντων καθώς και + - buttons για αυξηση της ποσότητας η μείωση χωρίς να χρειαστει να φύγει απο το cart και να παει στη σελίδα products. Προφανώς αν πατηθεί - σε product με quantity = 1 τότε το product διαγράφεται απο το καλάθι.

Η συνολική τιμή του καλαθιού (bonus) υπολογίζεται κάτω δεξιά.

Οδηγίες για να τρέξει η εφαρμογή

Για λόγους ευκολίας η εφαρμογή είναι dockerzied. Παρέχονται τόσο Dockerfiles οσο και docker-compose.yml όποτε θεωρητικά με ένα:

docker-compose up —build στο directory οπου είναι το docker-compose.yml θα πρεπει να πανε ολα καλα.

Έχει δημιουργηθεί και volume ωστε αν κλεισει η εφαρμογή και ξανα ανοίξει να έχουμε persistence. Στην πρώτη φορα που θα τρέξει η εφαρμογή γίνεται mongoDB migration και αρχικοποιείται η βάση ωστε να υπάρχει ενας admin. Μπορεί να πάρει λίγη ωρα γιατι πρέπει να γίνουν install ολα τα dependencies. Κανετε login σε αυτόν τον λογαριασμό για να “αποδεχτείτε” άλλους users που θα κάνετε register.

Admin credentials:

E-mail: admin@gmail.com

Pass: admin

URL: localhost:3000