Εργασία για το Μάθημα

Υπηρεσίες στο Υπολογιστικό Νέφος και την Ομίχλη

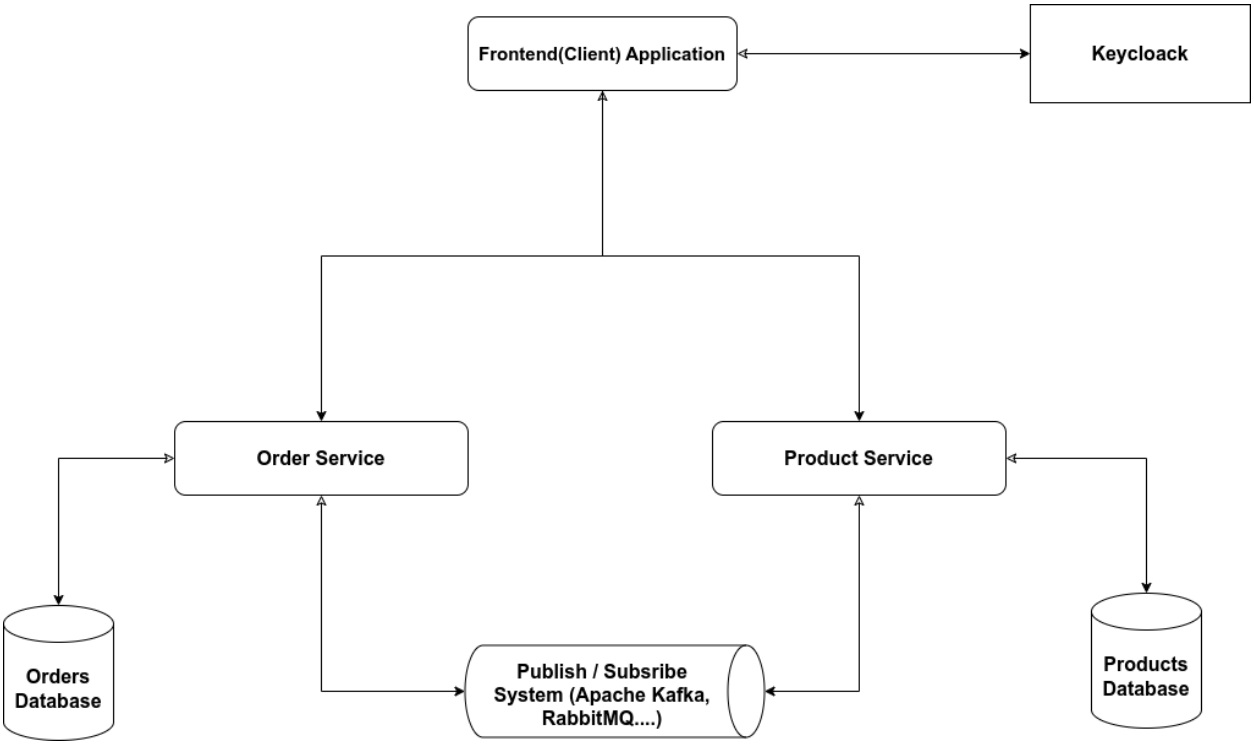
ΠΛΗ513

Παράδοση 15 Ιανουαρίου 2024

Email επικοινωνίας: alazidis@tuc.gr

Για την εργασία του μαθήματος θα δημιουργήσετε μια εφαρμογή E-Shop. **Για τη δημιουργία του είστε ελεύθεροι να χρησιμοποιήσετε ότι γλώσσα προγραμματισμού θέλετε (και frameworks), ότι βάση δεδομένων θέλετε και όποιο publish-subscribe σύστημα θέλετε.** Όλο το project θα πρέπει να τρέχει σε Docker και σε ένα Virtual Machine του Google Cloud Platform (GCP). Θα δημιουργήσετε ένα docker-compose.yml αρχείο με ότι αυτό χρειάζεται να έχει μέσα για να τρέξει η εφαρμογή. Σαν παραδοτέο, θα στείλετε όλο το κώδικά σας με το docker-compose αρχείο όπου με μια εντολή θα μπορώ να τρέξω την εφαρμογή σας. Επίσης μπορεί να ζητηθεί με ανακοίνωση κατά τη παράδοση της άσκησης να τρέξετε το Virtual Machine όπου τρέχουν οι εφαρμογές σας για να ελέγξω από εκεί τις εργασίες.

Η αρχιτεκτονική του eshop φαίνεται παρακάτω.



Περιγράφονται η εφαρμογή και τα services της εικόνας:

1. **Keycloack**: Το Keycloack είναι ένα service που χρησιμοποιείται για Authentication και Authorization. Οπότε, θα το χρησιμοποιεί το Frontend για να κάνει ένας χρήστης Σύνδεση και Εγγραφή. Θα τρέξετε το Keycloack σε ένα Docker Container μαζί με μια βάση δεδομένων (MySQL, PostgreSQL κλπ.) της επιλογής σας. Έπειτα, θα δημιουργήσετε τους εξής ρόλους:
    1. seller (θα είναι οι πωλητές προϊόντων, που θα μπορούν να εισάγουν προϊόντα στο eshop)

2. customer (θα είναι οι πελάτες που θα μπορούν να δούν και να παραγγείλουν τα προϊόντα)

Εσείς θα είστε ο admin.
Πληροφορίες για το Keycloack θα βρείτε στους παρακάτω συνδέσμους:

- https://www.appsdeveloperblog.com/keycloak-starting-standalone-server/
- https://www.appsdeveloperblog.com/keycloak-creating-a-new-user/
- https://wjw465150.gitbooks.io/keycloak-documentation/content/getting_started/index.html

2. **Frontend**: Είναι το service που θα πρέπει να φτιάξετε πρώτο και σε αυτό απεικονίζεται όλη η εφαρμογή.

Ένας χρήστης με του που προσπαθήσει να μπει στην εφαρμογή θα τον ανακατευθύνει σε μια σελίδα όπου θα γίνεται η Σύνδεση και η Εγγραφή. Ο χρήστης αν δεν συνδεθεί δεν θα πρέπει να μπορεί να δει τις σελίδες που θα περιγράψουμε παρακάτω.

**** Για τη σύνδεση και την εγγραφή ενός χρήστη στην εφαρμογή μας θα πρέπει να χρησιμοποιήσετε το rest api του Keycloack ****

Ο χρήστης κατά την εγγραφή θα δίνει το email που θα πρέπει να είναι μοναδικό, username, ένα password και θα επιλέξει αν θέλει να είναι **seller** ή **customer**. Από το frontend κατά την εγγραφή θα στέλνεται ένα request στο Keycloack για την εγγραφή του χρήστη. Αν η εγγραφή γίνει με επιτυχία η εφαρμογή θα πρέπει να εμφανίζει είτε ένα μήνυμα επιτυχίας, είτε να ανακατευθύνει το χρήστης στη σελίδα **Προϊόντα** που θα περιγραφεί παρακάτω.

Η σύνδεση του χρήστη θα γίνεται πάλι μέσω του keycloack και θα δίνονται το username και το password. Με τη σύνδεσή του ο χρήστης θα ανακατευθύνεται στη σελίδα *Προϊόντα* αν είναι customer ή *Τα Προϊόντα Μου* αν είναι seller. Θα πρέπει να έχετε ένα menu και ο κάθε χρήστης εκεί θα βλέπει τα ονόματα των σελίδων που του επιτρέπεται να δει και τέρμα δεξιά στο menu θα εμφανίζεται το username του χρήστη και θα του δίνεται και η επιλογή **Αποσύνδεσης**. Αν κάποιος προσπαθήσει να μπει σε κάποια σελίδα χωρίς να κάνει σύνδεση ή που δεν έχει πρόσβαση θα ανακατευθύνεται στη σελίδα Σύνδεσης.

Οι επιπλέον σελίδες που θα πρέπει να φτιάξετε είναι:

1. Προϊόντα - Εδώ θα φαίνονται όλα τα διαθέσιμα προϊόντα που μπορεί να αγοράσει ένας customer. Ο χρήστης θα μπορεί να αναζητήσει προϊόντα με βάση το όνομα ή όποια άλλη πληροφορία θεωρείτε εσείς χρήσιμη. Σε κάθε προϊόν θα φαίνεται το Όνομά του, μια εικόνα για το τι αφορά και η τιμή του.
2. Παραγγελίες - Εδώ θα φαίνονται όλες οι παραγγελίες του χρήστη. Ο χρήστης απλά μπορεί να δει τις παραγγελίες με το σύνολο των προϊόντων.

3. Καλάθι - Εδώ θα είναι όλα τα προϊόντα που ο χρήστης έχει επιλέξει να αγοράσει. Ο χρήστης μπορεί να αφαιρέσει κάποιο προϊόν ή να αγοράσει το ίδιο προϊόν πάνω από μία φορά. Τα προϊόντα που βρίσκονται στο καλάθι θα αποθηκέυονται σε ένα Session, LocalStorage ή Cookie και μόλις ο χρήστης ολοκληρώσει τη παραγγελία τότε θα στέλνονται στο service Order.

Αυτές οι επιλογές θα βρίσκονται στο menu και θα είναι ορατές μόνο από τους χρήστες με ρόλο **customer**.

Οι χρήστες με ρόλο **seller** αντίστοιχα θα βλέπουν τις σελίδες.

a. Τα Προϊόντα Μου - Όπου ένας seller μπορεί να:
    1) Προσθέσει Προϊόντα (Τίτλο προϊόντος, εικόνα, τιμή και ποσότητα)
    2) Ανανεώσει τη τιμή και τη ποσότητα ενός προϊόντος
    3) Διαγράψει ένα προϊόν

3. **Product Service**: Είναι το service όπου θα φτιάξετε το API για τα προϊόντα. Το Service αυτό θα συνδέεται με μία βάση δεδομένων της επιλογής σας (MySQL, PostgreSQL, MongoDB κλπ.). Αν ένας χρήστης προσπαθήσει να στείλει ένα request στο API θα ελέγχεται αρχικά αν έχει πρόσβαση και αν είναι συνδεδεμένος σε αυτό και στη συνέχεια θα του επιτρέπεται η ενέργεια. Αλλιώς θα επιστρέφεται ένα μήννυμα ότι ο χρήστης δεν έχει πρόσβαση στο συγκεκριμένο Service.

Η βάση δεδομένων που θα επιλέξετε θα αποθηκεύει τα εξής στοιχεία:

1. **Id** (Ένας αριθμός που θα είναι μοναδικός για κάθε προϊόν)
2. **Title** (Όνομα του προϊόντος)
3. **Img** (Εικόνα του προϊόντος)
4. **Price** (Τιμή του προϊόντος)
5. **Quantity** (Ο αριθμός των προϊόντων που είναι διαθέσιμος)
6. **User_username** (Το username του seller που δημιούργησε το προϊόν)

Είναι στην επιλογή σας να ονομάσετε όπως θέλετε τα παραπάνω στοιχεία και να προσθέσετε περισσότερα στοιχεία για κάθε προϊόν στη βάση (Δεν θα έχει σημασία στο βαθμό).

Ένα παράδειγμα του API των products περιγράφεται παρακάτω (Το API είναι ενδεικτικό και δεν είναι απαραίτητα χρηστικό).

METHOD	Routes	Description
POST	/products	Δημιουργία προϊόντος μόνο αν ο χρήστης είναι seller. (Ένα προϊόν τη φορά)
GET	/products	Επιστροφή όλων των Προϊόντων. Την επιλογή αυτή μπορείτε να τη

		χρησιμοποιήσετε στη σελίδα Προϊόντα. Ώστε οι customer να βλέπουν όλα τα προϊόντα που είναι διαθέσιμα.
GET	/products/:id και /products/:name /products/:username	Το **id** είναι ο μοναδικός αριθμός κάθε προϊόντος. Επιστρέφει ένα ή περισσότερα προϊόντα με το συγκεκριμένο id. Για **name** επιστρέφονται όλα τα προϊόντα με το συγκεκριμένο όνομα. Για username, επιστρέφονται όλα τα προϊόντα του χρήστη με αυτό το username.
PUT	/product/:id	Ενημέρωση στοιχείων ενός συγκεκριμένου προϊόντος με βάση το id (για seller)
DELETE	/product/:id	Διαγραφή ενός προϊόντος με βάση το id (για seller).

4. **Order Service**: Είναι το service όπου θα φτιάξετε το API για τις παραγγελίες. Το Service αυτό θα συνδέεται με μία βάση δεδομένων της επιλογής σας (MySQL, PostgreSQL, MongoDB κλπ.). Αν ένας χρήστης προσπαθήσει να στείλει ένα request στο API θα ελέγχεται αρχικά αν έχει πρόσβαση και αν είναι συνδεδεμένος σε αυτό και στη συνέχεια θα του επιτρέπεται η ενέργεια. Αλλιώς θα επιστρέφεται ένα μήνυμα ότι ο χρήστης δεν έχει πρόσβαση στο συγκεκριμένο Service. Σκοπός αυτού του Service είναι όλες οι παραγγελίες ενός customer να αποθηκεύονται σε ξεχωριστή βάση δεδομένων. Όταν γίνει μια παραγγελία αυτή θα εισέρχεται σε αυτό το service.

Η βάση δεδομένων που θα επιλέξετε θα αποθηκεύει τα εξής στοιχεία:

1. **Id** (Ένας αριθμός που θα είναι μοναδικός για κάθε παραγγελία)
2. **Products**: [
   {
   “title_1”, (Όνομα προϊόντος)
   “amount”, (Πόσες φορές πρόσθεσε ο χρήστης το προϊόν με τίτλο title_1 στο καλάθι του)
   “product_id_1” (id προϊόντος)
   },{
   “title_2”, (Όνομα προϊόντος)
   “amount”, (Πόσες φορές πρόσθεσε ο χρήστης το προϊόν με τίτλο title_1 στο καλάθι του)
   “product_id_2” (id προϊόντος)
   },…]
   (Σε ένα array από objects θα αποθηκεύσετε όλα τα ονόματα και το id των προϊόντων που αγόρασε ο χρήστης)
3. **Total_price** (Πόσο κοστίζει συνολικά η αγορά του χρήστη)
4. **Status** (Το status θα έχει τιμές είτε *Pending*, είτε *Success*, είτε *Reject*. Η αρχική τιμή του status θα είναι *Pending*)

5. **User_username** (Το όνομα του customer που έκανε την αγορά)

Ένα παράδειγμα του API των orders περιγράφεται παρακάτω (Το API είναι ενδεικτικό και δεν είναι απαραίτητα χρηστικό).

METHOD	Routes	Description
POST	/orders	Δημιουργία παραγγελίας από έναν customer.
GET	/orders/:username	Επιστροφή όλων των παραγγελιών ενός customer.

5. **Publish-Subscribe**: Το Publish-Subscribe σύστημα κάνει την επικοινωνία μεταξύ του *Product Service* και του *Order Service*. Μπορείτε να χρησιμοποιήσετε όποιο Publish-Subscribe σύστημα επιθυμείτε (Apache Kafka, RabbitMQ, MQTT κλπ.). Ας πούμε ότι ο customer με username John κάνει κάνει μια παραγγελία για ένα προϊόν με τίτλο «Nike Airmax». Μόλις ο John πατήσει να γίνει η παραγγελία για τα «Nike Airmax», αυτή θα σταλεί στο Order Service. Το Order Service θα ελέγξει αν ο John είναι customer και αν ναι τότε η παραγγελία του θα αποθηκευτεί στη βάση δεδομένων. Η παραγγελία του John έχει Status="Pending". Αφού η παραγγελία αποθηκευτεί στη βάση δεδομένων, στη συνέχεια στέλνεται στο Publish-Subscribe σύστημα ένα object με τα εξής στοιχεία της παραγγελίας:

   {
   Id (Το id της παραγγελίας),
   Products: [{
   product_id (id προϊόντος),
   amount (ποσότητα προϊόντος)
   }]
   }

Το Product Service με τη σειρά του θα διαβάσει από το Publish-Subscribe σύστημα τη παραπάνω παραγγελία και θα βρει το προϊόν στη βάση δεδομένων του. Θα δεί αν τον αριθμό του Quantity (Ποσότητα προϊόντος) και θα τον μειώσει με βάση τον αριθμό amount, του προϊόντος. Στη περίπτωσή μας θα ενημερώσει τη ποσότητα για το προϊόν «Nike Airmax». Οπότε αν το Quantity για το Nike Airmax είναι 10 και το amount 2 τότε το νέο Quantity για το «Nike Airmax» θα είναι 8. Η βάση δεδομένων που συνδέεται με τα products θα ενημερώσει το Quantity για το συγκεκριμένο προϊόν και θα στείλει στο Publish-Subscribe σύστημα ένα μήνυμα (object) "Success" με το id της παραγγελίας. Αν το Quantity μετά την αφαίρεση με το amount είναι αρνητικός αριθμός τότε το Product Service δεν θα ενημερώσει τη βάση του, αλλά αντίθετα θα στείλει στο Publish-Subscribe σύστημα ένα μήνυμα "Reject" με το id της παραγγελίας.

Π.χ. {

Id (Id παραγγελίας),
Status: “Success” ή “Reject”
}

Το Order Service με τη σειρά του θα διαβάσει το μήνυμα από το publish-subscribe σύστημα και στη συνέχεια θα ενημερώσει το Status της παραγγελίας με το συγκεκριμένο id.

Όπως αναφέρθηκε στην αρχή θα παραδώσετε όλο το κώδικα με το docker compose file όπου με volumes που θα φτιάξετε θα είναι αποθηκευμένα τα ήδη αποθηκευμένα αρχεία των βάσεων δεδομένων. Όλη η εργασία σας θα πρέπει να τρέχει με Docker και στο Virtual Machine του GCP. Μαζί με το κώδικα στείλτε και μια αναφορά με το τι κάνατε και τι όχι (Μισή σελίδα).

Σαν **Bonus** μπορεί κάποιος να τρέξει την εργασία σε GKE (Google Kubernetes Engine) οπού θα στείλετε τα επιπλέον yaml αρχεία και η εργασία θα ελεγχθεί με την ip του GKE.