1η Εργασία για το Μάθημα

Υπολογιστικό Νέφος και Ομίχλη ΠΛΗ513

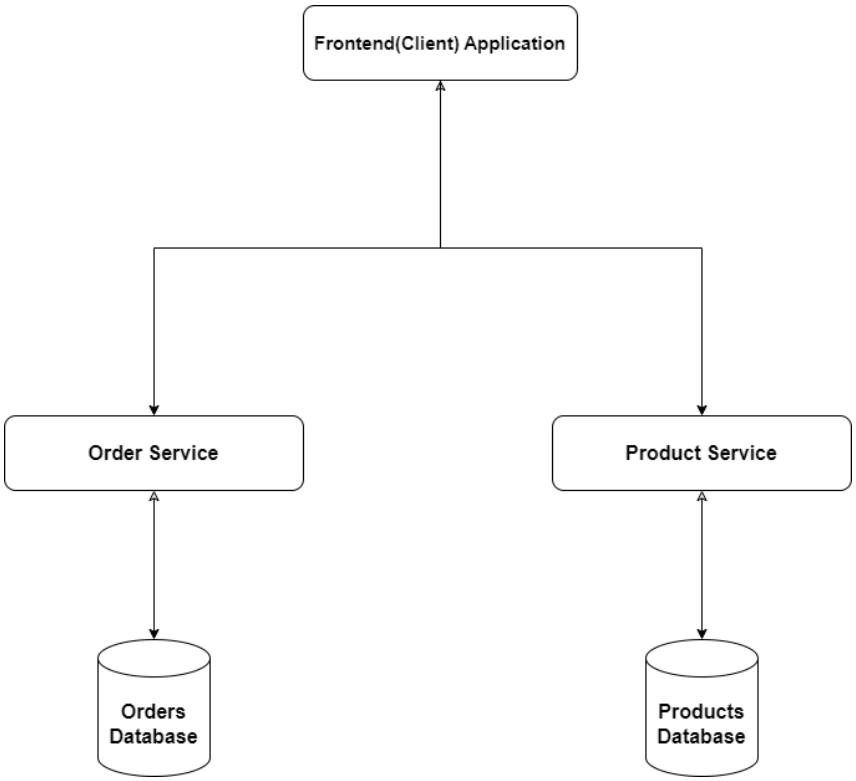
Παράδοση - 13 Νοεμβρίου

Email επικοινωνίας: alazidis@tuc.gr

Για την εργασία του μαθήματος, θα δημιουργήσετε μια εφαρμογή E-Shop, δηλαδή ένα ηλεκτρονικό κατάστημα. **Για τη δημιουργία του είστε ελεύθεροι να χρησιμοποιήσετε ότι γλώσσα προγραμματισμού θέλετε (και frameworks) και όποια βάση δεδομένων θέλετε**. Αυτή η εφαρμογή θα πρέπει να διαχωρίζεται σε δύο μέρη:

1. **Frontend**:
    - Είναι η "πρόσοψη" της εφαρμογής, δηλαδή το μέρος που βλέπουν και αλληλεπιδρούν οι χρήστες (πελάτες) του E-Shop.
    - Περιλαμβάνει το γραφικό περιβάλλον (UI), όπου οι χρήστες θα μπορούν να κάνουν διάφορες ενέργειες, όπως να πλοηγηθούν στα προϊόντα, να τα προσθέσουν στο καλάθι αγορών, να πραγματοποιήσουν αγορές κλπ.
    - Το Frontend μπορεί να υλοποιηθεί με οποιαδήποτε τεχνολογία για ιστοσελίδες, όπως **HTML, CSS, JavaScript** και κάποιο framework όπως React, Angular ή Vue.js.
    - **Είναι ξεχωριστή εφαρμογή από το backend και επικοινωνεί μαζί του μέσω αιτήσεων HTTP (API calls)**.
2. **Backend**:
    - Είναι η λογική της εφαρμογής, δηλαδή το μέρος που χειρίζεται τα δεδομένα και συνδέεται με τη βάση δεδομένων.
    - Περιλαμβάνει τις λειτουργίες του API, το οποίο δέχεται αιτήματα από το Frontend και επιστρέφει τα απαραίτητα δεδομένα (π.χ., λίστα προϊόντων, προσθήκη προϊόντων στο καλάθι, διαχείριση παραγγελιών).
    - Το Backend μπορεί να είναι υλοποιημένο με οποιαδήποτε γλώσσα προγραμματισμού (π.χ. Node.js, Java, Python) και μπορείτε να χρησιμοποιήσετε οποιαδήποτε βάση δεδομένων επιθυμείτε (π.χ., MySQL, PostgreSQL, MongoDB).
    - Είναι επίσης υπεύθυνο για τη διασφάλιση ότι τα δεδομένα αποθηκεύονται και ενημερώνονται σωστά στις βάσεις δεδομένων.

Η αρχιτεκτονική του eshop φαίνεται παρακάτω.



Στη περίπτωση της άσκησης θα υπάρχουν 2 Backend υπηρεσίες μία για τα προϊόντα και μία για τις παραγγελίες. Περιγράφονται η εφαρμογή και τα services της εικόνας. Όλα τα Services μπορούνε να επεκταθούνε με βάση τις προτιμήσεις σας. Επίσης όλα τα services (υπηρεσίες) θα πρέπει να είναι σε docker containers:

Frontend

Είναι το service που θα πρέπει να φτιάξετε πρώτο και σε αυτό απεικονίζεται όλη η εφαρμογή. Δεν είστε υποχρεωμένοι να χρησιμοποιήσετε κάποιο σύστημα ταυτοποίησης (Σύνδεσης ή Εγγραφής) χρήστη, αλλά ο καθένας θα μπορεί να εκτελέσει όλες τις παρακάτω ενέργειες. Οι σελίδες που θα πρέπει να φτιάξετε είναι:

1. Τα Προϊόντα μου - Εδώ ο χρήστης μπορεί να:
    i. Προσθέσει Προϊόντα (Τίτλο προϊόντος, εικόνα, τιμή και ποσότητα)
    ii. Ανανεώσει τη τιμή και τη ποσότητα ενός προϊόντος
    iii. Διαγράψει ένα προϊόν
2. Προϊόντα - Εδώ θα φαίνονται όλα τα διαθέσιμα προϊόντα που μπορεί να αγοράσει ο χρήστης που εισέρχεται στη σελίδα. 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. Όλα τα προϊόντα υπάρχουν στη βάση δεδομένων που συνδέεται με το **Product Service**.

3. Παραγγελίες - Εδώ θα φαίνονται όλες οι παραγγελίες του χρήστη. Ο χρήστης απλά μπορεί να δει τις παραγγελίες με το σύνολο των προϊόντων. Όλες οι παραγγελίες είναι αποθηκευμένες στη βάση δεδομένων που συνδέεται με το **Order Service.**
4. 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 **Order service**.

Αυτές οι επιλογές θα βρίσκονται στο menu και θα είναι ορατές.

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.) και θα αποθηκεύει όλες τις πληροφορίες των προϊόντων.

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)

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 (Αυτό μπορείτε να το τροποποιήσετε ή και να προσθέσετε περισσότερα Routes):

METHOD	Routes	Description
POST	/products	Δημιουργία προϊόντος.
GET	/products	Επιστροφή όλων των Προϊόντων. Την επιλογή αυτή μπορείτε να τη χρησιμοποιήσετε στη σελίδα Προϊόντα.
GET	/products/:id και /products/:title	Το **id** είναι ο μοναδικός αριθμός κάθε προϊόντος. Επιστρέφει ένα ή περισσότερα προϊόντα με το συγκεκριμένο id. Για **title** επιστρέφονται όλα τα προϊόντα με το συγκεκριμένο όνομα.
PUT	/product/:id	Ενημέρωση στοιχείων ενός συγκεκριμένου προϊόντος με βάση το id.

DELETE	/product/:id	Διαγραφή ενός προϊόντος με βάση το id.

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.). Σκοπός αυτού του Service είναι όλες οι παραγγελίες ενός χρήστη που θέλει να αγοράσει ένα προϊόν 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)
3. “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)
4. **Total_price** (Πόσο κοστίζει συνολικά η αγορά του χρήστη)
5. **Status** (Το status θα έχει τιμές είτε *Pending*, είτε *Success*, είτε *Reject*. Η αρχική τιμή του status θα είναι *Pending*)

Ένα παράδειγμα του API των orders περιγράφεται παρακάτω (Αυτό μπορείτε να το τροποποιήσετε ή και να προσθέσετε περισσότερα Routes):

METHOD	Routes	Description
POST	/orders	Δημιουργία παραγγελίας από έναν customer.
GET	/orders	Επιστροφή όλων των παραγγελιών του χρήστη.
GET	/orders/:username	Αυτό αφορά μόνο τη περίπτωση χρήσης συστήματος ταυτοποίησης. Επιστροφή όλων των παραγγελιών ενός customer.

Βάσεις δεδομένων

Στο project θα υπάρχουν 2 διαφορετικές βάσεις δεδομένων. Μπορείτε να επιλέξετε όποια εσείς θέλετε π.χ. MySQL, PostgreSQL, MongoDB κλ.π. Σας επισυνάπτω, ένα παράδειγμα **docker-compose.yml** αρχείου με ένα docker container μιας PostgreSQL βάση δεδομένων και ένα PgAdmin (γραφικό περιβάλλον για διαχείρηση της PostgreSQL). Επειδή το **Order Service** και το **Product Service** έχουν το καθένα τη δική του βάση δεδομένων. Εσείς θα πρέπει να έχετε **1 docker-compose.yml αρχείο**, με δύο PostgreSQL containers, ένα container για το Frontend, ένα container για το Order Service και ένα container για το Product Service.

```yaml
version: "3.8"

services:
  order_db:
    image: postgres
    container_name: order_db
    restart: always
    expose:
      - 5555
    ports:
      - "5555:5432"
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: admin
      POSTGRES_DB: order_db
    volumes:
      - .data/order_db:/var/lib/postgresql/data




  pgadmin:
    image: dpage/pgadmin4
    container_name: pgadmin4_container
    restart: always
    ports:
      - "8888:80"
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@admin.com
      PGADMIN_DEFAULT_PASSWORD: admin
```

```
    volumes:
      - pgadmin-data:/var/lib/pgadmin

volumes:
  order_db:
  pgadmin-data:

networks:
  default:
    driver: bridge
```

Παραδοτέα

Ένα zip αρχείο με όλο το κώδικα και μια αναφορά με το τι κάνατε και τι όχι και τι χρεάζεται να κάνω για να τρέξω την εφαρμογή. Όλη η εφαρμογή θα πρέπει να τρέχει μέσα από το **docker-compose.yml** αρχείο, όπου εκεί θα υπάρχουν όλες οι υπηρεσίες σε docker containers.