# Τεχνολογίες Εφαρμογών Διαδικτύου

**Υποχρεωτική Εργασία Μαθήματος Εαρινού Εξαμήνου**

**2019-2020**

*Βαργιάμης Μιχαήλ - Α.Μ 1115201300018*

# Περιεχόμενα


1 Εισαγωγή 3

2 Οδηγίες εγκατάστασης 4

3 Frond-end 5

4 Back-end 6

5 Διάφορα 8

# 1 Εισαγωγή

Η εργασία έγινε ατομικά, σύμφωνα με τις προδιαγραφές της εκφώνησης, χωρίς την υλοποίηση του bonus ερωτήματος. Η ανάπτυξη της εφαρμογής έγινε με το IntelliJ IDEA Ultimate Edition. Για το front-end χρησιμοποιηθηκε Angular 9, ενώ για το back-end Java Spring Boot όπως και στις διαλέξεις του μαθήματος. Ως βάση της εφαρμογής χρησιμοποιήθηκε κώδικας από *αυτό* και *αυτό* το tutorial. Το όνομα της εφαρμογής είναι **getaroom**.

# 2 Οδηγίες εγκατάστασης

Αρχικά γίνεται αποσυμπίεση του "getaroom.zip". Οι φάκελοι front-end-getaroom και back-end-getaroom μπορούν να ανοιχθούν με το IntelliJ IDEA Ultimate ως project, καθώς και να τρέξουν από εκεί.

Η εφαρμογή τρέχει πάνω απο MySQL Server. Αρχικά, για την δημιουργία της βάσης πρέπει να εκτελεστούν οι παρακάτω εντολές:

```
mysql> create database getaroom;
mysql> create user 'springuser'@'%' identified by 'ThePassword';
mysql> grant all on db_example.* to 'springuser'@'%';
```

Στη συνέχεια το back-end ξεκινάει με την εντολή

```
mvnw spring-boot:run
```

στο root projcect directory. Οι συνδέσεις γίνονται μέσω https στο port 8443.

Αμέσως μετά, και πριν την έναρξη του front-end, πρέπει να εκτελεστούν σε MySQL terminal οι παρακάτω εντολές προκειμένου να εισαχθούν στη βάση οι 3 ρόλοι χρήστη.

```
mysql> insert into roles(name) values('ROLE_ADMIN');
mysql> insert into roles(name) values('ROLE_TENANT');
mysql> insert into roles(name) values('ROLE_HOST');
```

Για να αποκτήσει ένας χρήστης δικαιώματα διαχειριστή (admin) πρέπει πρώτα να γίνει κανονικά εγγραφή χρήστη μέσω front-end και στη συνέχεια, να εκτελεστεί σε MySQL terminal η εντολή

```
mysql> insert into user_roles(user_id, role_id) values(x, 1);
```

όπου x, το user id του χρήστη (το οποίο μπορεί να γίνει γνωστό διαβάζοντας την εγγραφή του χρήστη στον πίνακα user).

Στη συνέχεια, το front-end ξεκινάει με την εντολή

```
ng serve
```

στο root projcect directory. Οι συνδέσεις γίνονται στο port 4200. Επίσης, επειδή χρησιμοποιείται Angular Material, πιθανόν να χρειαστεί και η εντολή

```
ng add @angular/material
```

Να σημειωθεί ότι το certificate του back-end για την υποστήριξη του https βρίσκεται στο src/main/resources/keystore, με όνομα getaroom.p12.

# 3 Frond-end

Ένας μη συνδεδεμένος χρήστης μπορεί να κάνει αναζήτηση κατοικοιών απο την αρχική σελίδα (home component), να δει λίστα όλων των δωματίων (room component), καθώς και των λεπτομερειών (room-details component) και των φωτογραφιών (view-photos component) κάθε δωματίου. Επίσης μπορεί να κάνει εγγραφή νέου χρήστη (register component) ή να συνδεθεί εάν έχει ήδη λογαριασμό (login component). Ένας χρήστης με τον ρόλο του ενοικιαστή μπορεί **επιπλέον** να δει λίστα των κρατήσεων του (user-reservations component), να κάνει κρατήσεις (reservation component) και να επεξεργαστεί τα στοιχεία του προφίλ του (profile component). Ένας χρήστης με τον ρόλο του οικοδεσπότη μπορεί **επιπλέον** να προσθέσει ενα νέο δωμάτιο (add-a-room component), να δεί λίστα με τα δωμάτια που διαθέτει ο ίδιος προς ενοικίαση (my-rooms component), να επεξεργαστεί τις πληροφορίες των δωματίων του (edit-room-info component), να προσθέσει φωτογραφίες στα δωμάτια του (add-photos component) καθώς και να διαγράψει δωμάτια που έχει προσθέσει ο ίδιος (my-rooms component). Ο διαχειριστής της εφαρμογής μπορεί **επιπλέον** να δει λίστα με όλους τους χρήστες (users component), να δει λίστα με όλους τους χρήστες στους οποίους εκρεμμέι έγκριση για να αποκτήσουν ρόλο οικοδεσπότη ,να δει λεπτομέρειες για το προφίλ κάθε χρήστη (user-details component), καθώς να διαγράψει ένα χρήστη ή και να εγκρίνει το αίτημα οικοδεσπότη (user-details component).

Η διαχείριση του JWT γίνεται μέσω του token-storage service. Αφού ο χρήστης δώσει επιτυχώς το username και το password του, το JWT που επιστρέφεται ως απάντηση αποθηκέυεται στη συνεδρία μέσω του service αυτού. Μέσω του auth interceptor (στον φάκελο helpers), το JWT προσθέτεται στις κεφαλίδες κάθε HTTP αίτησης πριν αυτή σταλεί στο back-end.

# 4 Back-end

Η επικοινωνία με το back-end γίνεται ως εξής:

| **BACK-END API** | | | |
|---|---|---|---|
| **Method** | **URL** | **Request Body** | **Response on Success** |
| POST | /api/auth/signin | "username": ,"password": | JWT |
| POST | /api/auth/signup | "username": ,"password", "name": ,"surname": , "email": ,"phone": , "role": ["tenant"] or ["tenant", "host"] | Creates new user and returns success message |
| POST | /api/upload | file={room photo} &roomId={room id photo belongs to} **(in post-data format)** | Success message and url of saved file |
| GET | /files/room /{roomId} | | URL and name of all room photos |
| GET | /files/{roomId} /{filename} | | File contents |
| GET | /api/host-request | | All host requests pending |
| POST | /api/host-request /{hostRequestId} | | Approves host request with specific id |
| POST | /api/reservations | "user": {"id": }, "room": {"id": }, "price": , "dateFrom": , "dateTo": | All Reservation details |
| GET | /api/reservations /user/{userId} | | Reservations made by user userId |
| GET | /api/reservations /room/{roomId} | | Reservations made for room roomId |
| GET | /api/rooms | | All rooms |
| POST | /api/rooms | "owner": {"id": }, "title": ,"country": , "city": ,"area": , "address": ,"numBeds": , "description": , "pricePerDay": , "mainPhotoUrl": | Creates new room and returns room details |
| GET | /api/rooms /{roomId} | | Details of specific room |

| | | | |
|---|---|---|---|
| PUT | /api/rooms /{roomId} | ”owner”: {”id”: }, ”title”: ,”country”: , ”city”: ,”area”: , ”address”: ,”numBeds”: , ”description”: , ”pricePerDay”: , ”mainPhotoUrl”: | Success message |
| DELETE | /api/rooms /{roomId} | | Deletes specific and returns success message |
| GET | /api/rooms /{roomId}/add-main-photo | ”url”: | Adds photo at url as roomId’s main photo |
| POST | /api/search | ”country”: , ”city”: (optional), ”area”: (optional), ”dateFrom”: , ”dateTo”: , ”numBeds”: | Rooms satisfying search terms |
| GET | /api/users | | All users details |
| GET | /api/users /{userId} | | User userId details |
| PUT | /api/users /{userId} | ”username”: ,”password”, ”name”: ,”surname”: , ”email”: ,”phone”: | Updates userId’s details |
| DELETE | /api/users /{userId} | | Deletes userId user and returns success message |

Όλα τα δεδομένα στο σώμα μίας αίτησης είναι σε JSON format, εκτός αν υπάρχει διαφορετική διευκρίνιση. Όλες οι ημερομηνίες είναι σε μορφή YYYY-MM-DD.

Για κάποιες διεπαφές όπως π.χ. GET /api/users/{userId} εκτελούνται επιτυχώς οι ενέργειες μόνο για τους χρήστες στους οποίους ανήκει ο λογαριασμός. Κάποιες άλλες διεπαφές όπως DELETE /api/users/{userId} απαιτούν δικαιώματα διαχειριστή.

# 5 Διάφορα

Κάποιες λειτουργικότητες που αναφέρονται στην εκφώνηση δεν υλοποιήθηκαν λόγω περιορισμένου χρόνου. Γενικά δεν είχα προηγούμενη εμπειρία με Web development και σε συνδυασμό με το οτι επέλεξα να κάνω ατομικά την εργασία απαίτησε αρκετό χρόνο μέχρι να εξοικειωθώ με τις έννοιες και τα εργαλεία, και έκανε αρκετά πιο χρονοβόρα την ανάπτυξη της εφαρμογής. Συγκεκριμένα χρειάστηκα περίπου 2 μήνες καθημερινής ενασχόλησης προκειμένου να φτάσω σε αυτό το σημείο την εργασία. Παρόλα αυτά, θεωρώ πως αποκόμισα χρήσιμα skills στον τομέα αυτό.

Ο κώδικας του project υπάρχει και στο github.com χωρίς να έχει γίνει κάποιο commit μετά την ημερομηνία λήξης της προθεσμίας της άσκησης. Σε περίπτωση που χρειαστεί για οποιονδήποτε λόγο, επικοινωνήστε μαζί μου έτσι ώστε να κάνω τα repositories public και να σας στείλω link σε αυτά.