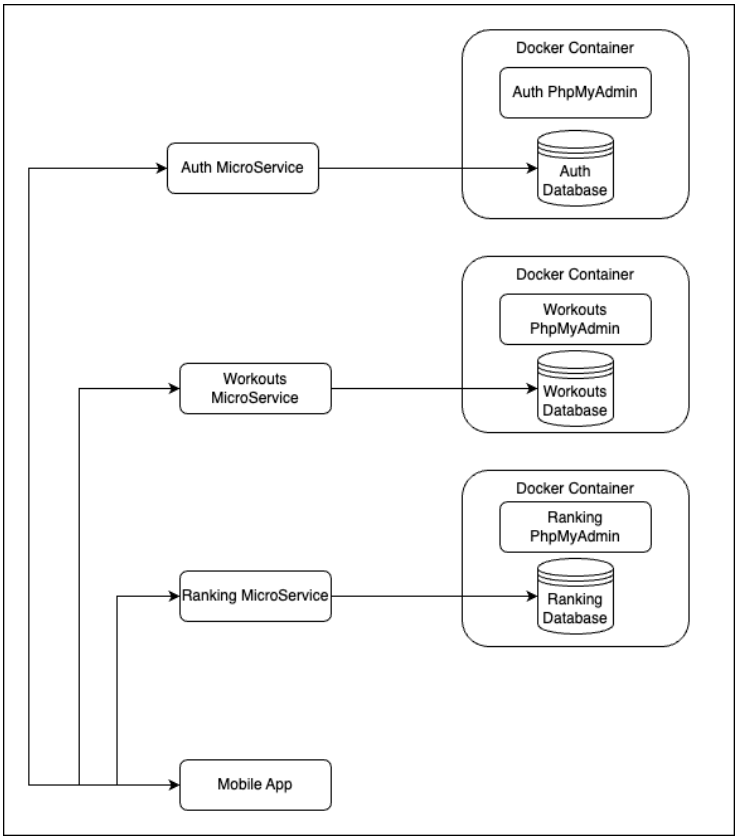
Documentație proiect Cloud Computing

Balica Adrian-Claudiu

Măncilă Doru-Bogdan

Gr. 506

Aplicația realizată sub numele de QuickTrain este o aplicație mobile, nativă iOS în domeniul sportului. Mai exact aceasta este o aplicație de workout ce oferă posibilitatea utilizatorilor să facă sport de acasă cu ajutorul unui sistem de gamification. Există o multitudine de exerciții sub forma de videoclip-uri (integrate prin intermediul Youtube) iar pentru ca utilizatorii să câștige experiență și puncte aceștia trebuie sa vizualizeze cel puțin 80% din antrenament (ulterior având în plan și adăugarea validării finalizării unui exercițiu prin intermediul brățărilor de fitness / smartwatch-urilor). De asemenea, pentru a debloca exerciții mai complexe aceștia trebuie sa acumuleze experiență (vizualizând exerciții mai ușoare) pentru a debloca antrenamentele cu dificultate mai mare.



Aplicația mobile folosește ca și tehnologii principale Swift și RxSwift, interfața grafică fiind realizata prin intermediul framework-ului UIKit. Arhitectura folosita este una de tip MVVM care încorporează design pattern-ul Coordinator pentru navigarea între ecrane. Partea de back-end folosește ca și tehnologie principală Node.js cu Nest.js ca framework peste acesta. Ca bază de date este folosită una de tip MySQL, mai exact MariaDB cu versiunea 10.3. Componenta de back-end comunică cu baza de date prin intermediul unui ORM numit TypeOrm. Aplicația este construită pe trei microservicii, cel de auth care se ocupă de partea de user și informații despre aceștia (login,

register, userInfo), microserviciul de workouts (care se ocupă de adăugare, editare, ștergere antrenamente) și serviciul de ranking care rulează sub forma unui cronjob și realizează automat un clasament lunar al jucătorilor bazat pe punctajele realizate de aceștia în luna respectivă. La finalul fiecărei luni se realizează un nou clasament si se resetează punctele. Serviciul de ranking comunica cu serviciul de user prin http request-uri, toate cele trei servicii rulând pe containere separate în docker și având fiecare propriul mediu de rulare. Lucrând efectiv ca trei aplicații separate în cazul în care un microserviciu se oprește celelalte încă funcționează (nu vor mai funcționa părțile care au nevoie de legături cu celelalte servicii dar funcțiile si funcționalitățile independente vor fi in continuare utilizabile).

docker-compose.dev.yml

```yaml
version: "10.3"
services:
  mysql:
    image: mariadb
    container_name: auth
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: auth
      MYSQL_DATABASE: 'auth'
      MYSQL_USER: 'auth'
      MYSQL_PASSWORD: 'auth'
    volumes:
      - $PWD/mysql-data:/var/lib/mysql
    ports:
      - "3306:3306"
    networks:
      - default
  phpmyadmin:
    image: phpmyadmin/phpmyadmin:5.0.1
    container_name: workout_phpmyadmin2
    restart: always
    environment:
      PMA_HOST: mysql
    ports:
      - "8080:80"
```

Pentru fiecare microserviciu avem o baza de date independentă și un container în care aceasta rulează definit prin fișierul yml. De asemenea, acolo avem pentru fiecare db integrat și un soft pentru gestionarea și vizualizarea bazelor de date (php my admin). Fiecare rulează pe porturi diferite acestea fiind rulate pe local prin comanda docker-compose:

- auth: aplicația rulează pe portul 3500, baza de date de pe docker pe 3306:3306 iar php myadmin aferent bazei de date pe 8080.

- workout: aplicația rulează pe protul 3300, baza de date de pe docker pe 3307:3306 iar php myadmin aferent bazei de date pe 8081.
- clasamentJobs: aplicația rulează pe protul 3600, baza de date de pe docker pe 3308:3306 iar php myadmin aferent bazei de date pe 8082.