"Ss. Cyril and Methodius" University in Skopje
FACULTY OF COMPUTER
SCIENCE AND ENGINEERING

MOVIE WATCHLIST APP

Завршен проект по предметот Континуирана интеграција и испорака

Docker, Kubernetes, Github Actions & AWS

Ментор:

д-р Панче Рибарски

м-р Стефан Андонов

Изработила:

Бојана Андонова 221225

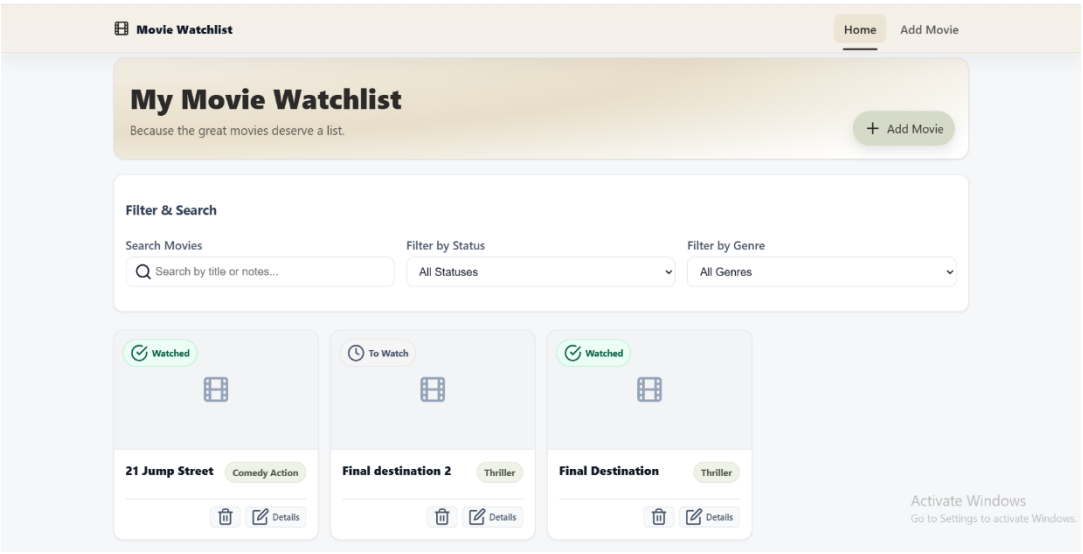
Скопје, 2025

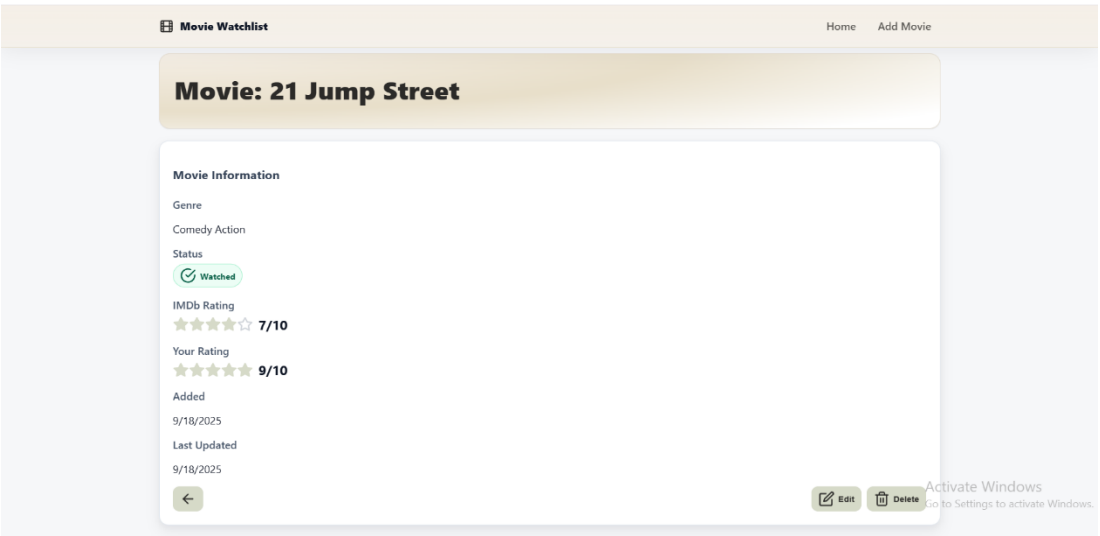
Содржина

Вовед

Movie Watchlist App е веб-апликација за управување со листа на филмови што сакате да ги гледате или веќе сте ги гледале. Апликацијата е развиена како финален проект за предметот Континуирана интеграција и испорака. Овозможува да додавате филмови во вашата колекција, да ги организирате според статус (за гледање/гледано), да давате лични рејтинзи и белешки, да пребарувате и филтрирате според жанр или наслов, и да проследувате статистики за вашата колекција. Апликацијата е изработена со React/TypeScript за frontend, FastAPI/Python за backend, MongoDB за база на податоци.

За потребите на овој предмет, соодветно се извршени сите барања кои подетално се опишани подолу.

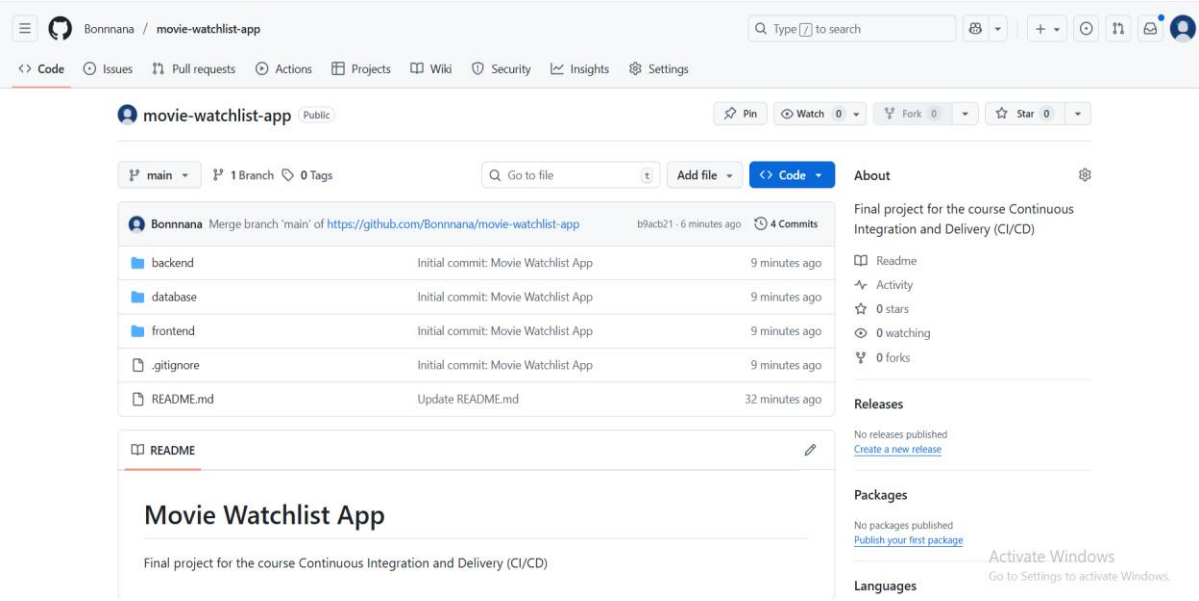




1. Github Репозиториум

Апликацијата е поставена на јавен Github репозиториум достапен на следниот линк:

https://github.com/Bonnnana/movie-watchlist-app.git



2. Докеризација

Во рамките на гит репозиториумот, поставени се два Dockerfile фајлови, еден за frontend и еден за backend. За базата ја користам официјалната MongoDB Docker слика од Docker Hub.

```dockerfile
FROM node:20-alpine as build

WORKDIR /app

COPY package*.json ./

RUN npm ci

COPY . ./

RUN npm run build

FROM nginx:alpine

COPY --from=build /app/dist /usr/share/nginx/html

COPY nginx.conf /etc/nginx/nginx.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

```dockerfile
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python", "main.py"]
```

3. Оркестрација со Docker Compose

Во docker-compose.yml фајл-от за оркестрација на базата на податоци и апликацијата имам дефинирано три сервиси:

mongodb: користи последна верзија од MongoDB и ги сетира потребните environment променливи како и volumes за базата, односно mongodb_data за персистентност и init-mongo.js за иницијализација.

backend: користејќи го Dockerfile-от се build-нува апликацијата и сетира соодветна околина со environment променливи и порти (5000:5000). Дополнително се означува дека апликацијата backend зависи од mongodb, односно за да биде стартувана апликацијата, прво ни е потребен сервисот mongodb за базата на податоци

frontend: исто така користејќи го Dockerfile-от се build-нува апликацијата и сетира соодветна околина со environment променливи и порти (3000:80). Дополнително се означува дека апликацијата frontend зависи од backend сервисот.

Сите сервиси се поврзани преку дефинираната movie-watchlist-network мрежа за комуникација помеѓу контејнерите.

```yaml
version: '3.8'

services:
  mongodb:
    image: mongo:latest
    container_name: movie-watchlist-mongodb
    ports:
      - "27017:27017"
    volumes:
      - mongodb_data:/data/db
    networks:
      - movie-watchlist-network

  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    container_name: movie-watchlist-backend
    ports:
      - "5000:5000"
    environment:
      - DATABASE_URL=${DATABASE_URL}
      - FRONTEND_URL=${FRONTEND_URL}
```

```
    depends_on:
      - mongodb
    networks:
      - movie-watchlist-network

  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    container_name: movie-watchlist-frontend
    ports:
      - "3000:80"
    environment:
      - VITE_API_URL=${VITE_API_URL}
    depends_on:
      - backend
    networks:
      - movie-watchlist-network

volumes:
  mongodb_data:

networks:
  movie-watchlist-network:
    driver: bridge
```

```
(.venv) PS C:\Users\Bojana\Desktop\Proekti\movie-watchlist-app> docker compose up -d
time="2025-09-16T23:37:48+02:00" level=warning msg="C:\\Users\\Bojana\\Desktop\\Proekti\\movie-watchlist-app\\docker-compose.yml: the attribute `version`
is obsolete, it will be ignored, please remove it to avoid potential confusitime="2025-09-16T23:37:48+02:00" level=warning msg="C:\\Users\\Bojana\\Desktop
\\Proekti\\movie-watchlist-app\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
"
 ✓ Container movie-watchlist-mongodb                 Started                                                                                0.7s
 ✓ Container movie-watchlist-backend                 Started                                                                                0.9s
 ✓ Container movie-watchlist-frontend                Started                                                                                1.0s
```

```
(.venv) PS C:\Users\Bojana\Desktop\Proekti\movie-watchlist-app> docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS                      NAMES
2e4906f86cbc   movie-watchlist-app-frontend   "/docker-entrypoint.…"   11 minutes ago   Up 11 minutes   0.0.0.0:3000->80/tcp       movie-watchlist-fronten
d
b72bb0d69fae   movie-watchlist-app-backend    "python main.py"         11 minutes ago   Up 11 minutes   0.0.0.0:5000->5000/tcp     movie-watchlist-backend
a70ec2c84744   mongo:latest                   "docker-entrypoint.s…"   11 minutes ago   Up 11 minutes   0.0.0.0:27017->27017/tcp   movie-watchlist-mongodb
```

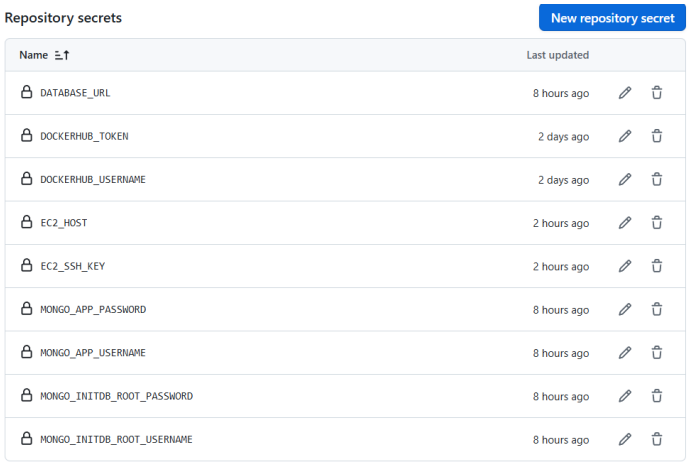
4. Github Actions и автоматски Deploy

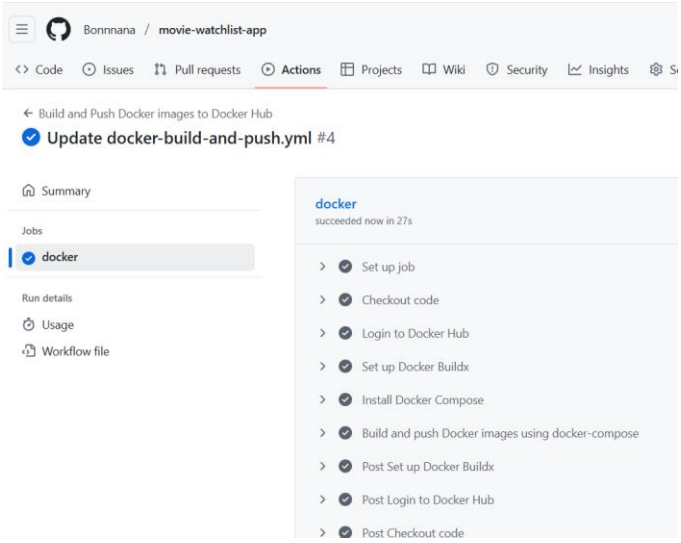
За делот на Continuous Integration користам Github Actions со цел да се автоматизира процесот за build и deploy на Docker Images. Со тоа, секогаш кога одредена промена е направена на Git репозиториумот, GitHub Actions го започнува **docker-build-and-push.yml** workflow-от (pipeline-от) за да се направи нова Docker Image и истата да се стави на DockerHub, така што секоја нова верзија на кодот автоматски се транспарентира во нов

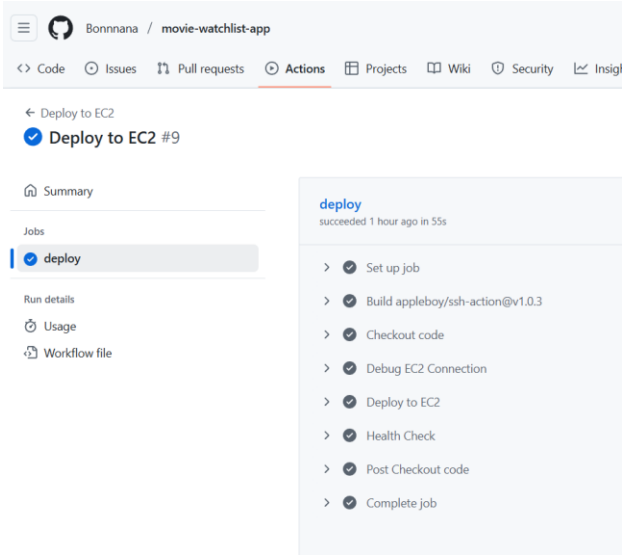
Docker Image, подготвен за користење при локален развој, тестирање или деплојмент во Kubernetes.

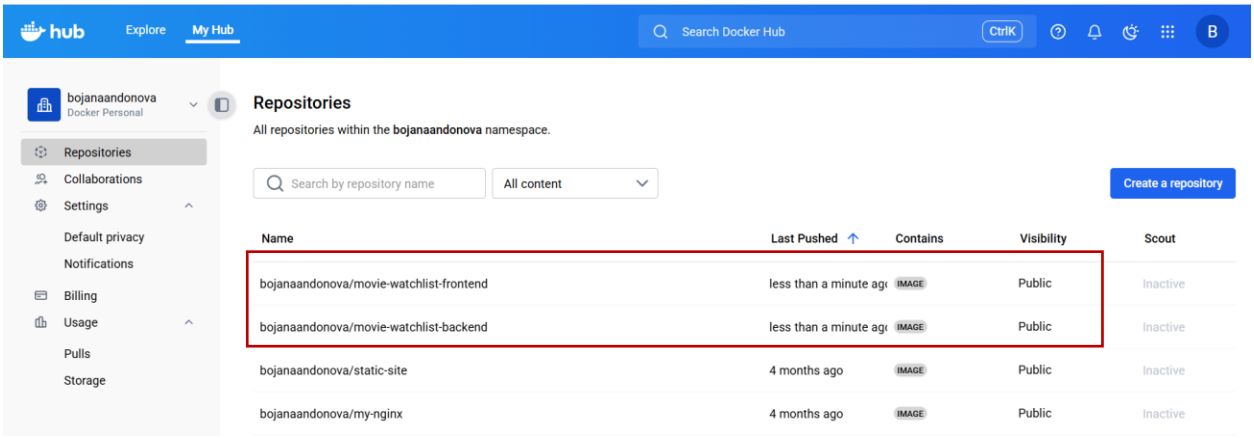
По успешното завршување на build процесот, се активира deploy.yml workflow-от кој автоматски се поврзува на AWS EC2 инстанцата преку SSH и ја деплојнува апликацијата со pull на најновиот код од Git репозиториумот, рестартирање на контејнерите со docker-compose, и извршување на health check за да се потврди дека апликацијата работи правилно во продукциското окружување.

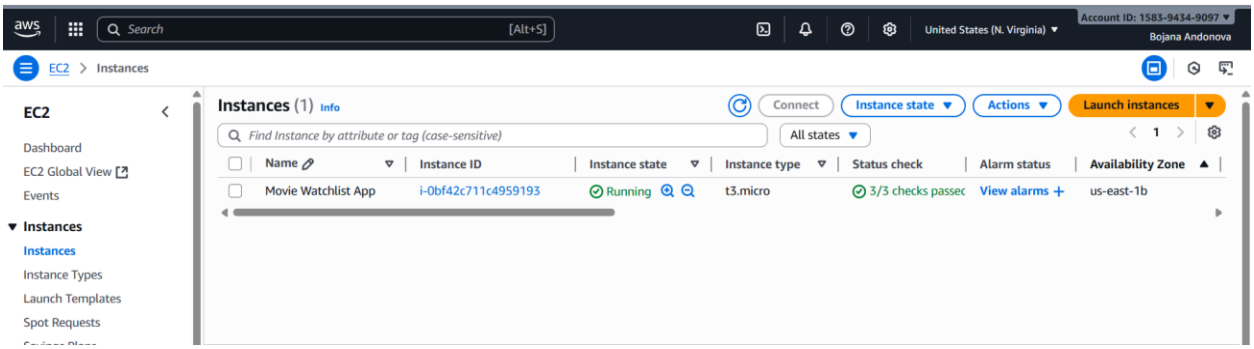
Дополнително, за сето ова да работи, во делот Secrets додадени се потребните креденцијали и променливи. Ова овозможува истите да не се хардкодирани и јавно достапни.

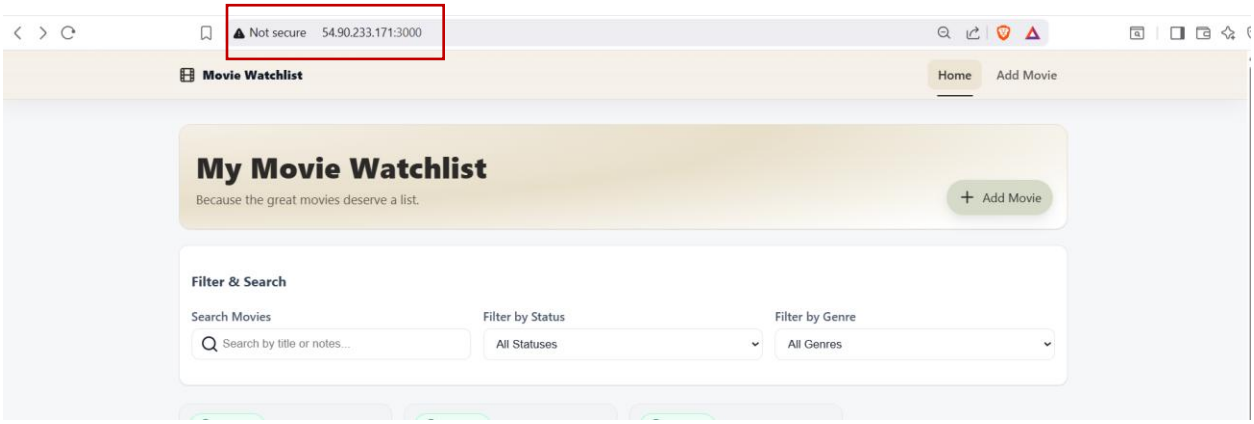












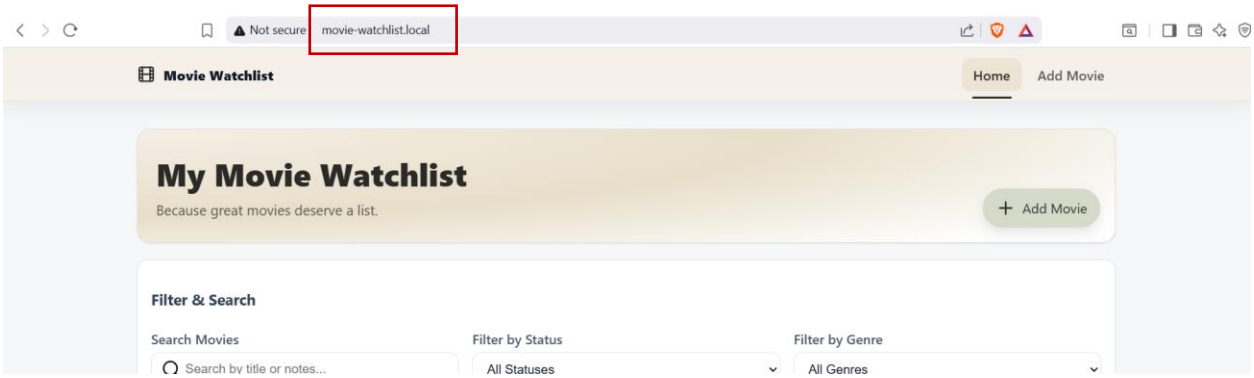
Апликацијата сега е јавно достапна на http://54.90.233.171:3000/

5. Kubernetes

За овој дел најпрво креирав посебен namespace movie-watchlist со namespace.yml file. За базата на податоци дефинирав secrets.yml file каде што се дефинира secret за лозинката на MongoDB базата на податоци, како и Stateful Set што овозможува стабилна мрежа и перзистирање на податоците што ги имаме за апликацијата, менаџира со подови од image од самиот MongoDB и ни овозможува scaling на подовите. Перзистирањето на податоците е овозможено со Persistent Volume Claim і креирање на Persistent Volume од тој claim со што податоците се чуваат и после уништување на подот и се достапни после креирање на нов под. Понатаму, за backend апликацијата дефинирав backend/deployment.yml што содржи Deployment и backend/service.yml што содржи Service за FastAPI апликацијата. Преку Deployment овозможуваме креирање и менаџирање на повеќе исти Pods со што се обезбедува Scaling, Self-Healing, Rollouts и Rollbacks, а преку Service овозможуваме пристап до самите подови. Слично, за frontend делот дефинирав frontend/deployment.yml и frontend/service.yml за React апликацијата. Последно, креирав nginx-proxy Ingress што овозможува пристап до нашата апликација од надвор и го користи претходно дефинираните сервиси за frontend и backend деловите од апликацијата. Ова овозможува пристап до апликацијата преку адресата: http://movie-watchlist.local/

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: movie-watchlist
  labels:
    name: movie-watchlist
```

```yaml
spec:
  ingressClassName: nginx
  rules:
  - host: movie-watchlist.local
    http:
      paths:
      - path: /api
        pathType: Prefix
        backend:
          service:
            name: movie-watchlist-backend
            port:
              number: 5000
      - path: /
        pathType: Prefix
        backend:
          service:
            name: movie-watchlist-frontend
            port:
              number: 80
```



Во configmap.yml file се дефинирани конфигурациони податоци за апликацијата како што се database URL, API endpoints и други environment variables кои се потребни за правилно функционирање на апликацијата

Дополнително, креирав и kustomization.yml file кој служи како централна точка за управување со сите Kubernetes ресурси во апликацијата. Овој фајл овозможува лесно деплојнување на целата апликација со една команда kubectl apply -k . наместо да се применуваат сите YAML фајлови посебно