

Факултет за информатички науки и компјутерско инженерство

Континуирана интеграција и испорака

Проект: Hotels App

Изработил:

Стефан Тосев, 213091

Датум:

23.09.2024г.

Професори:

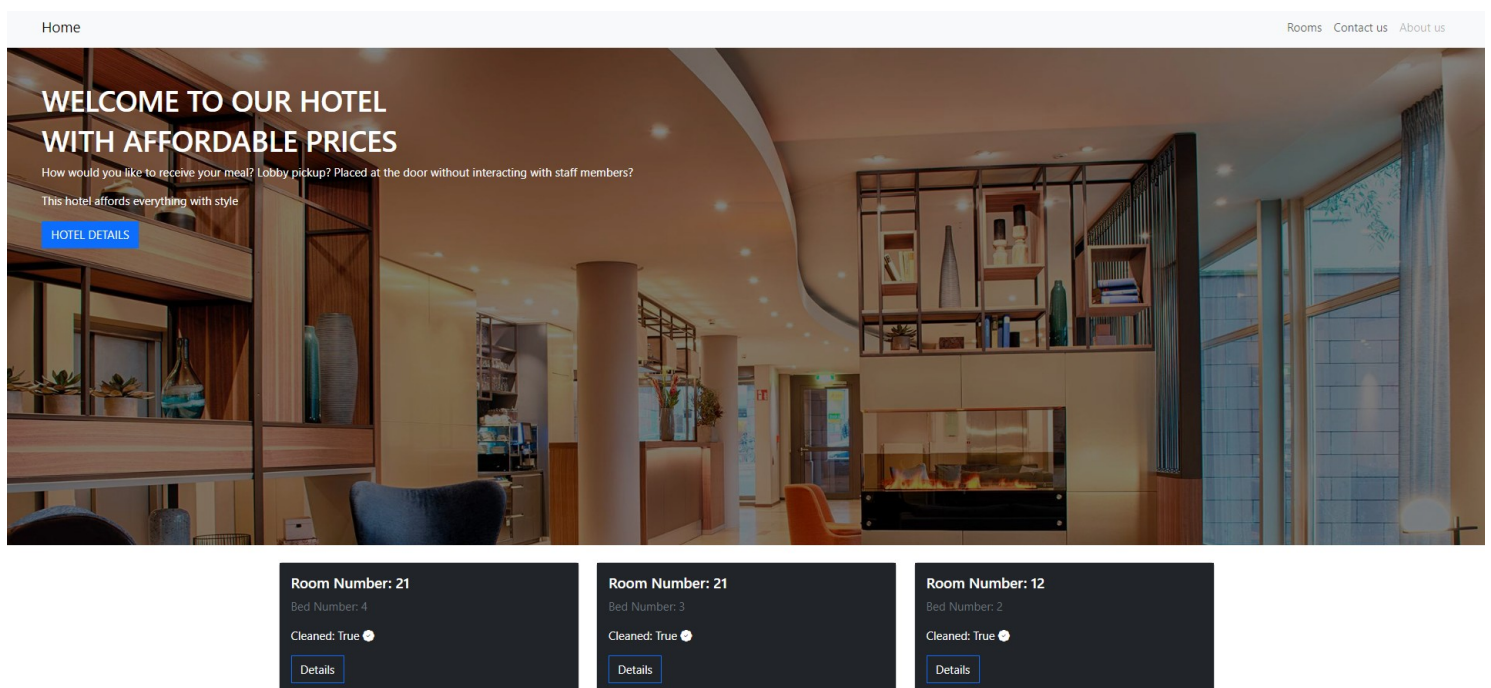
проф. д-р. Милош Јовановиќ

проф. д-р. Панче Рибарски

1. Вовед

Овој проект вклучува развој на апликација за Хотел каде што може да се резервираат соби и да се прави преглед на соби. Изградена е со помош на Django и база на податоци со помош на PostgreSQL.

Контејнеризирање на апликацијата е направено со помош на Docker и нејзина испорака преку Kubernetes. Целта е да се обезбеди скалабилна апликација со автоматска иницијализација на шемата на базата на податоци и континуирана интеграција и испорака.



Слика1. Home page на апликацијата

Линкови:

https://github.com/stefantosev/Hotel_App_Devops
(Github)

https://hub.docker.com/repository/docker/walkorion/hotels_kol-web/general
(Dockerhub)



Code:

Start:

End:


Id picture:

IsReserved: ☐

Room:

Employee:

Слика2. Форма/Страна за резервација



Room Number: 23

Bed Number: 4

Cleaned: True

Слика3. Преглед на детали на собите

2. Докеризација

Овој `Dockerfile` е скрипта која го дефинира како да се креира Docker image за Django апликацијата.

```
1 FROM python:3.12
2
3 ENV PYTHONDONTWRITEBYTECODE 1
4 ENV PYTHONUNBUFFERED 1
5
6 WORKDIR /Hotels
7
8 COPY requirements.txt /Hotels/
9
10 RUN pip install --upgrade pip
11 RUN pip install -r requirements.txt
12
13 COPY . /Hotels/
14
15 EXPOSE 8080
16
17 CMD ["sh", "-c", "python manage.py migrate && python manage.py runserver 0.0.0.0:8080"]
```

Слика3. Dockerfile

Се користи официјалниот `python:3.12` image, кој доаѓа со Python 3.12 претходно инсталиран. Потоа се поставува работната папка во контејнерот која се вика `/Hotels`. Сите наредни команди ќе се извршуваат во оваа директорија.

Следно се копира датотеката `requirements.txt` (што ги содржи потребните Python пакети) од локалната машина во директоријата `/Hotels` во контејнерот.

Се ажурира `pip` (Python пакет менаџерот) на најновата верзија во контејнерот. `RUN pip install -r requirements.txt`
Потоа со овој ред ги инсталира сите Python пакети наведени во `requirements.txt`.

Со командата `EXPOSE 8080` вој ред укажува дека контејнерот ќе ја користи портата 8080 за комуникација со надворешниот свет (со надворешни корисници или други контејнери).

Прво се извршува `python manage.py migrate`, што ги извршува миграциите на базата на податоци (ако има промени во моделите на Django). Потоа `python manage.py runserver 0.0.0.0:8080` го стартува Django серверот на адресата `0.0.0.0` (што го прави достапен на сите мрежни интерфејси) и на порта 8080.

Овој `docker-compose.yml` фајл дефинира како да се оркестрираат повеќе Docker контејнери за Django апликација која користи PostgreSQL база на податоци.

```
version: '3'

services:
  db:
    image: postgres:16
    volumes:
      - postgres_data:/var/lib/postgresql/data
    environment:
      POSTGRES_DB: ${DATABASE_NAME}
      POSTGRES_USER: ${DATABASE_USER}
      POSTGRES_PASSWORD: ${DATABASE_PASSWORD}
    ports:
      - "5432:5432"

  web:
    build: .
    command: >
      sh -c "python manage.py migrate &&
        python manage.py createsuperuser --noinput --username ${DJANGO_SUPERUSER_USERNAME} --email ${DJANGO_SUPERUSER_EMAIL}
        python manage.py runserver 0.0.0.0:8000"
    volumes:
      - .:/Hotels
    ports:
      - "8000:8000"
    depends_on:
      - db
    env_file:
      - .env

volumes:
  postgres_data:
```

Слика4. Docker compose

3. Github Actions pipeline

```
name: CI

on:
  push:
    branches:
      - master

jobs:
  check-required-files:
    name: Check Required Files
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Check for required files
        run: |
          required_files=("Dockerfile" "docker-compose.yaml" "requirements.txt")
          for file in "${required_files[@]"; do
            if [ ! -f "$file" ]; then
              echo "$file is missing"
              exit 1
            fi
          done
          echo "All required files are present"

  check-yaml-syntax:
    name: Check YAML Syntax
    runs-on: ubuntu-latest
    needs: check-required-files

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Check YAML syntax
        run: find . -name '*.yaml' -exec yamllint {} +
```

```
build-and-push:
  name: Build and Push Docker Image to Docker Hub
  runs-on: ubuntu-latest
  needs: check-yaml-syntax

  steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Login to Docker Hub
      uses: docker/login-action@v2
      with:
        username: ${ secrets.DOCKERHUB_USERNAME }}
        password: ${ secrets.DOCKERHUB_ACCESS_TOKEN }}

    - name: Build and Push to Docker Hub
      uses: docker/build-push-action@v4
      with:
        context: .
        push: true
        tags: ${ secrets.DOCKERHUB_USERNAME }}/hotels_kol-web:latest

    - name: Verify Docker Image
      run: docker pull ${ secrets.DOCKERHUB_USERNAME }}/hotels_kol-web:latest
```

Слика5. CI pipeline

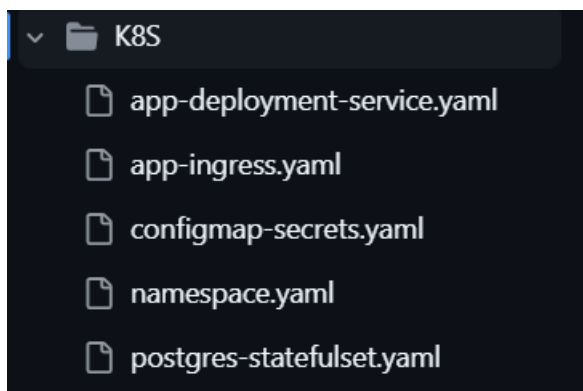
Овој CI pipeline дефинира три основни чекори за континуирана интеграција и испорака (CI/CD) со GitHub Actions.

Jobs: Проверка на YAML синтаксата, Проверка на потребните фајлови, Градење и испраќање на Docker слика во Docker Hub.

Сите чекори се автоматизирани и се извршуваат секогаш кога ќе има нов push на master гранката.

4. Kubernetes

Подоле е прикажан фолдерот со yaml фајловите/манифестети потребни за узвршување на deployment на апликацијата. Прикажани се namespace.yaml и app-ingress.yaml.



```
1  apiVersion: v1
2  kind: Namespace
3  metadata:
4    name: hotel-app
```

```
1  apiVersion: traefik.containo.us/v1alpha1
2  kind: Middleware
3  metadata:
4    name: my-middleware
5    namespace: hotel-app
6  spec:
7    headers:
8      customRequestHeaders:
9        X-Custom-Header: "MyValue"
10 ---
11 apiVersion: networking.k8s.io/v1
12 kind: Ingress
13 metadata:
14   name: myapp-ingress
15   namespace: hotel-app
16   annotations:
17     traefik.ingress.kubernetes.io/router.middlewares: hotel-app-my-middleware@kubernetescrd
18     traefik.ingress.kubernetes.io/router.entrypoints: web
19 spec:
20   rules:
21   - host: hotels.local
22     http:
23       paths:
24       - path: /
25         pathType: Prefix
26         backend:
27           service:
28             name: myapp
29             port:
30               number: 80
```