ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

Проект на тема:

Курсова работа по Виртуализация и облачни технологии “ВОТ”

Изготвил: *Милица Кирякова*

Научен ръководител: *Любомир Стоянов*

София

2023

1. Какво е “BOT”

“BOT” представлява просто приложение, написано на JavaScript, което създава HTTP сървър, който слуша на порт 8080 и връща отговор със съобщението

```
O))        O))    O))))       O))) O)))))))
 O))       O))   O))    O))       O))
  O))     O))  O))        O))     O))
   O))   O))   O))        O))     O))
    O)) O))    O))        O))     O))
     O))))       O))     O))       O))
      O))           O))))           O))
```

Когато е достъпен.

2. Използвани технологии

а) JavaScript

Приложението е написано изцяло на JavaScript, като е използван http модула на Node.js. http модулът осигурява функционалност за създаване на HTTP сървъри и обработка на входящи заявки. Кодът просто създава HTTP сървър, който слуша на порт 8080 и изпраща обикновен текстов отговор.

б) Docker

За контейнеризирането на приложението е използван Docker, с цел то да може да бъде качено в Google Cloud

в) Google Cloud Platform

Проектът е качен на GCP, където са използвани необходимите функционалности

3. Как работи кодът?

a) Javascript

const http = require('http');
Този ред импортва http модула, който е вграден модул в Node.js, използван за създаване на HTTP сървър.
const server = http.createServer((req, res) => {
Този ред създава HTTP сървър с помощта на метода createServer от http модула. Той приема функция за обратно извикване с два параметъра: req (заявка) и res (отговор). Функцията за обратно извикване се изпълнява всеки път, когато се направи заявка към сървъра.

```
  res.writeHead(200, {'Content-Type': 'text/plain'});
```

Този ред задава заглавката на HTTP отговора. Методът writeHead се използва за указване на кода на състоянието на отговора (200 означава „ОК“) и типа съдържание на отговора, който в този случай е зададен на 'text/plain'

```
res.write(`
O))        O))    O))))     O))) O)))))))
 O))       O))   O))    O))       O))
  O))     O))  O))        O))     O))
   O))   O))   O))        O))     O))
    O)) O))    O))        O))     O))
     O))))       O))     O))      O))
      O))          O))))          O))
```

`);
Този ред записва тялото на отговора. Той използва write метода, за да напише посочения текст като тяло на отговора.

  res.end();
});
Този ред завършва отговора. Той сигнализира на сървъра, че отговорът е завършен и трябва да бъде изпратен обратно на клиента.

server.listen(8080, () => {
  console.log('Server started!');
});
Този ред стартира сървъра и го кара да слуша на порт 8080. Listen методът приема номер на порт и незадължителна функция за обратно извикване. В този случай функцията за обратно извикване се използва за регистриране на съобщение в конзолата, когато сървърът стартира успешно.

б) Dockerfile

FROM node:14-alpine
Този ред задава основният image за Docker image-а. Той уточнява, че то трябва да бъде изградено върху изображението node:14-alpine.

WORKDIR /app
Този ред задава работната директория вътре в контейнера на /app. Това означава, че всички последващи команди ще бъдат изпълнени в тази директория.

COPY package*.json ./
Този ред копира файловете package.json от локалната директория в директорията /app вътре в контейнера. Тази стъпка се извършва

отделно от копирането на целия код на приложението, за да се използва механизмът за кеширане на Docker.

RUN npm install
Този ред изпълнява командата npm install вътре в контейнера. Тя инсталира зависимостите, посочени във файла package.json, в директорията /app/node_modules на контейнера.

COPY . .
Този ред копира всички файлове и директории от локалната директория в директорията /app вътре в контейнера.

EXPOSE 8080
Този ред информира Docker, че контейнерът ще слуша на порт 8080 по време на изпълнение.

CMD ["node", "app.js"]
Този ред указва командата, която трябва да се изпълни при стартиране на контейнера, а именно node app.js. Това ще стартира приложението Node.js, посочено във файла app.js в контейнера.