POLITECHNIKA
LUBELSKA
WYDZIAŁ ELEKTROTECHNIKI
I INFORMATYKI

Programowanie Full-Stack w Chmurze Obliczeniowej
LABORATORIUM

Zadanie 1

Adrian Domański

Zadanie 1.
Proszę napisać program serwera (dowolny język programowania), który realizować będzie następującą funkcjonalność:

a. po uruchomieniu kontenera, serwer pozostawia w logach informację o dacie uruchomienia, imieniu i nazwisku autora serwera (imię i nazwisko studenta) oraz porcie TCP, na którym serwer nasłuchuje na zgłoszenia klienta.

Na potrzeby tego zadania serwer został napisany w frameworku Javy – Spring Boot.

```java
@SpringBootApplication
public class OblokiApplication {

    private static final Logger logger = LoggerFactory.getLogger(OblokiApplication.class);
    public static void main(String[] args) {
        ConfigurableApplicationContext context = SpringApplication.run(OblokiApplication.class, args);
        logger.info("Data uruchomienia: {}", java.time.LocalDateTime.now());
        logger.info("Autor serwera: {} {}", "Adrian", "Domanski");
        if(context.getEnvironment().getProperty( key: "server.port", Integer.class) != null)
        {
            int serverPort = context.getEnvironment().getProperty( key: "server.port", Integer.class);
            logger.info("Port TCP: {}", serverPort);
        }

    }

}
```

```
C:\Users\User\Downloads\obloki>docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED              STATUS                            PORTS                    NAMES
dda058becacd   obloki-0.0.1-snapshot:done   "java -jar /app.jar"   About a minute ago   Up 4 seconds (health: starting)   0.0.0.0:8080->8080/tcp   angry_pasteur

C:\Users\User\Downloads\obloki>docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED              STATUS                    PORTS                    NAMES
dda058becacd   obloki-0.0.1-snapshot:done   "java -jar /app.jar"   About a minute ago   Up 51 seconds (healthy)   0.0.0.0:8080->8080/tcp   angry_pasteur

C:\Users\User\Downloads\obloki>docker logs dda058becacd
```

```
2023-06-16 19:12:57.246  INFO 1 --- [           main] com.example.obloki.OblokiApplication     : Data uruchomienia: 2023-06-16T19:12:57.246853
2023-06-16 19:12:57.248  INFO 1 --- [           main] com.example.obloki.OblokiApplication     : Autor serwera: Adrian Domanski
2023-06-16 19:12:57.248  INFO 1 --- [           main] com.example.obloki.OblokiApplication     : Port TCP: 8080
2023-06-16 19:13:03.318  INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
```

b. na podstawie adresu IP klienta łączącego się z serwerem, w przeglądarce powinna zostać wyświetlona strona informująca o adresie IP klienta i na podstawie tego adresu IP, o dacie i godzinie w jego strefie czasowej.

Tutaj na chwilę zatrzymamy się. Potrzebujemy adresu IP klienta, żeby na jego podstawie wyłapać informacje o dacie i godzinie w jego strefie czasowej. Żeby uzyskać owe informacje skorzystałem z API apistack (https://ipstack.com/), które oferuje w darmowej wersji 100 użyć.
Jednakże teraz trzeba zająć się problemem, iż wszystko dzieje się lokalnie, więc początkowo chcąc pobrać ip (HttpServletRequest request) została zwrócona wartość „0:0:0:0:0:0:0:1" z czego ten adres IP jest standardowym adresem IPv6 dla lokalnego hosta i odpowiada adresowi loopback w IPv4 (127.0.0.1). Jednakże nie interesuje nas adres lokalny.

Korzystając z komendy „nslookup myip.opendns.com resolver2.opendns.com"

Światła dziennego mogą ujrzeć takie informacje:

```
C:\Users\User\Downloads\obloki>nslookup myip.opendns.com resolver2.opendns.com
Server:  dns.opendns.com
Address:  [redacted]

Non-authoritative answer:
Name:    myip.opendns.com
Address:  [redacted]
```

Z czego to ten adres pod non-authoritative answer będzie nam potrzebny, aby móc połączyć się z API apistack. Żeby w nieinwazyjny sposób uzyskać adres IP możemy skorzystać ze strony: http://checkip.amazonaws.com/, który wyświetla nasz publiczny adres IP.

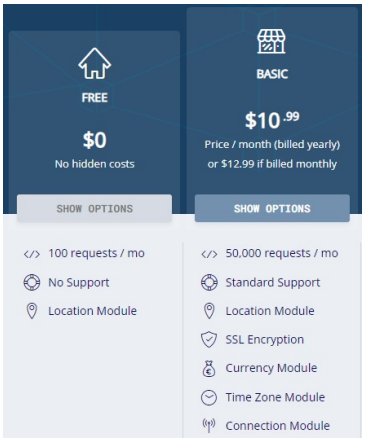
```java
private String getIpAddressF() {
    try {
        URL url = new URL( spec: "http://checkip.amazonaws.com");
        BufferedReader reader = new BufferedReader(new InputStreamReader(url.openStream()));
        String ipAddress = reader.readLine().trim();

        if (!ipAddress.isEmpty()) {
            return ipAddress;
        } else {
            System.out.println("BRAK");
        }

    } catch (IOException e) {
        e.printStackTrace();
    }

    return "-1";
}
```

Teraz możemy już przejść do wyłapania informacji, które oferuje nam nasze API apistack. Niestety w darmowej wersji nie otrzymujemy informacji o czasie:



Ale można uzyskać informacje o kontynencie i kraju i na podstawie tych informacji będę wybierał informacje na temat godziny i daty.

Funkcja zwracająca informacje z API:

```
private String getIpInfoF(String ipAddress) {
    OkHttpClient client = new OkHttpClient();
    String url = "http://api.ipstack.com/" + ipAddress + "?access_key=f341a578e5a0
    Request request = new Request.Builder()
            .url(url)
            .build();
    try (Response response = client.newCall(request).execute()) {
        ResponseBody responseBody = response.body();
        if (responseBody != null) {
            return responseBody.string();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    return "ERROR";
}
```

Funkcje dzięki którym można zwrócić informacje o strefie czasowej:

```
private static String getTimeZoneFromIpInfo(String ipInfo) {
    Map<String, String> timeZoneMap = createTimeZoneMap();

    //Wyobrębniamy potrzebne informacje oferowane przez API
    String[] parts = ipInfo.split( regex: "\"continent_code\":");
    if (parts.length > 1) {
        String continentCode = parts[1].trim().replace( target: "\"", replacement: "").replace( target: ",", replacement: "");

        parts = ipInfo.split( regex: "\"country_code\":");
        if (parts.length > 1) {
            String countryCode = parts[1].trim().replace( target: "\"", replacement: "").replace( target: ",", replacement: "");
            String timeZoneId = timeZoneMap.get(continentCode + countryCode);
            if (timeZoneId != null) {
                return timeZoneId;
            }
        }
    }

    return ZoneId.systemDefault().getId(); // Ustaw domyślną strefę czasową
}
private static Map<String, String> createTimeZoneMap() {
    Map<String, String> timeZoneMap = new HashMap<>();
    // Dodaj tutaj mapowanie kontynentu i kraju na odpowiednią strefę czasową
    // Przykład: timeZoneMap.put("EUPL", "Europe/Warsaw");
    return timeZoneMap;
}
```

I później wykorzystać to w funkcji zwracającej informacje o czasie:

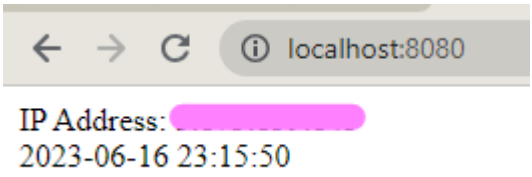
```
private String getTime(String timeZoneIP) {
    LocalDateTime localDateTime = LocalDateTime.now(ZoneId.of(timeZoneIP));
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
    return localDateTime.format(formatter);

}
```

Co później wyświetlamy na stronie:

```
@GetMapping("/")
public String getInfo() {
    ClientService clientService = new ClientService();
    return "IP Address: " + clientService.getIpAddress() + "<br>" + clientService.getClientTime();

}
```



Zadanie 2.

Opracować plik Dockerfile, który pozwoli na zbudowanie obrazu kontenera realizującego funkcjonalność opisaną w punkcie 1. Przy ocenie brane będzie sposób opracowania tego pliku (wieloetapowe budowanie obrazu, ewentualne wykorzystanie warstwy scratch, optymalizacja pod kątem funkcjonowania cache-a w procesie budowania, optymalizacja pod kątem zawartości i ilości warstw, healthcheck itd). Dockerfile powinien również zawierać informację o autorze tego pliku (ponownie imię oraz nazwisko studenta).

```
FROM openjdk:11
# Ustawienie autorstwa
LABEL author="Adrian Domanski"
VOLUME /tmp
# Nasluchiwanie kontenera na porcie 8080
EXPOSE 8080
# Monitorowanie zdrowia kontenera
HEALTHCHECK --interval=30s --timeout=3s \
  CMD curl -f http://localhost:8080/health || exit 1
# Sciezka do pliku JAR aplikacji
ARG JAR_FILE=target/obloki-0.0.1-SNAPSHOT.jar
ADD ${JAR_FILE} app.jar
# Uruchomienie aplikacji
ENTRYPOINT ["java","-jar","/app.jar"]
```

```
C:\Users\User\Downloads\obloki>docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED              STATUS                            PORTS                    NAMES
dda058becacd   obloki-0.0.1-snapshot:done   "java -jar /app.jar"   About a minute ago   Up 4 seconds (health: starting)   0.0.0.0:8080->8080/tcp   angry_pasteur

C:\Users\User\Downloads\obloki>docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED              STATUS                    PORTS                    NAMES
dda058becacd   obloki-0.0.1-snapshot:done   "java -jar /app.jar"   About a minute ago   Up 51 seconds (healthy)   0.0.0.0:8080->8080/tcp   angry_pasteur
```

Zadanie 3.
Należy podać polecenia niezbędne do:

a. zbudowania opracowanego obrazu kontenera,

```
C:\Users\User\Downloads\obloki>docker build -t obloki-0.0.1-snapshot:ptasiemleczko .
```

b. uruchomienia kontenera na podstawie zbudowanego obrazu,

```
C:\Users\User\Downloads\obloki>docker run -p 8080:8080 obloki-0.0.1-snapshot:ptasiemleczko

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.12)

2023-06-16 19:03:18.951  INFO 1 --- [           main] com.example.obloki.OblokiApplication     : Starting OblokiApplication v0.0.1-SNAPSHOT using Java 11.0.16 on 4d2f04372e82 with PID 1 (/app.jar started by root in /)
2023-06-16 19:03:18.959  INFO 1 --- [           main] com.example.obloki.OblokiApplication     : No active profile set, falling back to 1 default profile: "default"
2023-06-16 19:03:20.286  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2023-06-16 19:03:20.301  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-06-16 19:03:20.302  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.75]
```

c. sposobu uzyskania informacji, które wygenerował serwer w trakcie uruchamiana kontenera (patrz: punkt 1a),

```
C:\Users\User\Downloads\obloki>docker ps
CONTAINER ID   IMAGE                                COM
4d2f04372e82   obloki-0.0.1-snapshot:ptasiemleczko   "ja

C:\Users\User\Downloads\obloki>docker logs 4d2f04372e82
```

d. sprawdzenia, ile warstw posiada zbudowany obraz.

```
C:\Users\User\Downloads\obloki>docker history obloki-0.0.1-snapshot:ptasiemleczko
IMAGE          CREATED         CREATED BY                                      SIZE      COMMENT
41d9d8ee741b   2 hours ago     ENTRYPOINT ["java" "-jar" "/app.jar"]           0B        buildkit.dockerfile.v0
<missing>      2 hours ago     ADD target/obloki-0.0.1-SNAPSHOT.jar app.jar…   18.2MB    buildkit.dockerfile.v0
<missing>      2 hours ago     ARG JAR_FILE=target/obloki-0.0.1-SNAPSHOT.jar   0B        buildkit.dockerfile.v0
<missing>      2 hours ago     HEALTHCHECK &{["CMD-SHELL" "curl -f http://l…   0B        buildkit.dockerfile.v0
<missing>      2 hours ago     EXPOSE map[8080/tcp:{}]                         0B        buildkit.dockerfile.v0
<missing>      2 hours ago     VOLUME [/tmp]                                   0B        buildkit.dockerfile.v0
<missing>      2 hours ago     LABEL author=Adrian Domanski                    0B        buildkit.dockerfile.v0
```