

# Weather Application

Mitocaru Irina | 343C1



## Cuprins

Introducere .....	4
Componente (containere) .....	4
Modul de comunicare .....	4
Tehnologii folosite .....	5
Angular .....	5
Descriere.....	5
Locație Github .....	5
Locație Dockerhub .....	5
Dockerfile .....	5
Java .....	5
Descriere.....	5
Locație Github .....	6
Locație Dockerhub .....	6
Dockerfile .....	6
Python .....	6
Descriere.....	6
Locație Github .....	6
Locație Dockerhub .....	6
Dockerfile .....	6
Elasticsearch .....	7
Descriere.....	7
Locație Github .....	7
Locație Dockerhub .....	7
Dockerfile .....	7
Prometheus .....	7
Descriere.....	7
Locație Github .....	7
Locație Dockerhub .....	7
Dockerfile .....	8
Grafana.....	8
Descriere.....	8
Locație Github .....	8
Locație Dockerhub .....	8
Dockerfile .....	8
Stiva de servicii (docker-compose) .....	8
Rezultate .....	10



## Introducere

Aplicația dockerizată are ca scop vizualizarea vremii pentru o locație anume (sau mai multe). Ea se poate folosi atât ca și guest, cât și ca utilizator, în cazul din urmă având anumite beneficii (de exemplu, salvarea și ștergerea locațiilor favorite). Aplicația dispune de 5 tehnologii (containere) descrise în capitolul 2 și 4. Deși este suficient un singur server de Backend pentru a atinge scopul aplicației, se folosește un al doilea server, cel de Python, pentru ca funcționalitățile să poată fi extinse cu ușurință. De exemplu, se poate introduce un model antrenat care să recomande noi locații din lume pentru care să se vizualizeze vremea pe baza preferințelor utilizatorului.

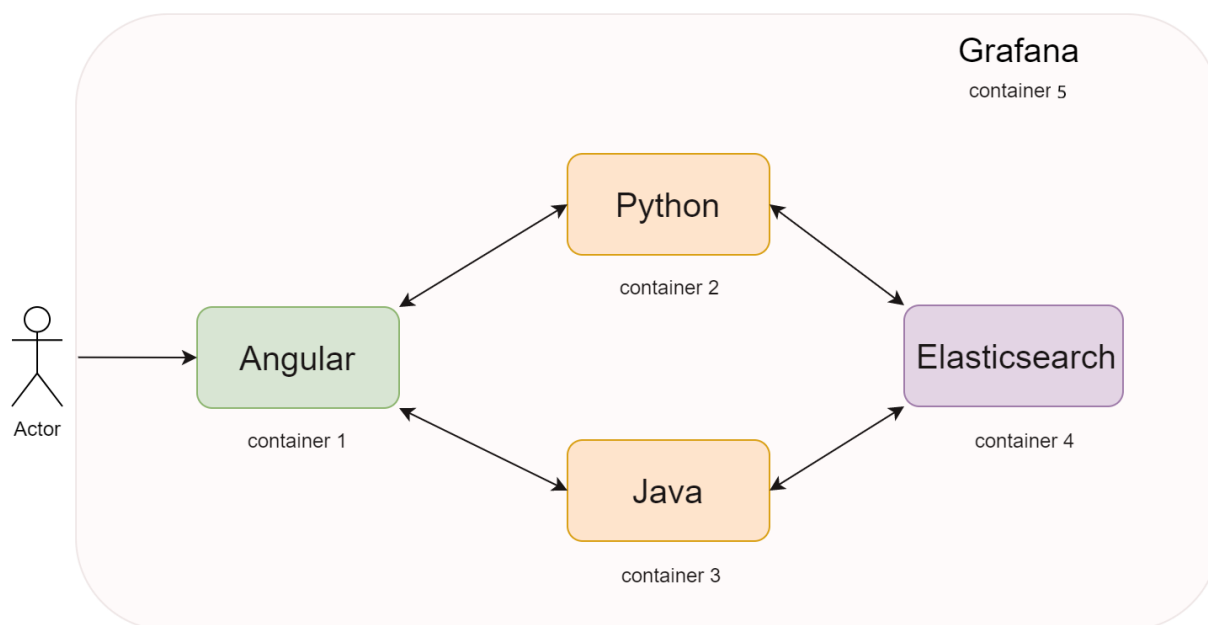
## Componente (containere)

Aplicația folosește 6 containere:

- 1 container pentru Frontend
- 2 containere pentru Backend
- 1 container pentru stocarea persistentă a datelor
- 1 container pentru vizualizarea a diferite metrici
- 1 container pentru monitorizare

## Modul de comunicare

Utilizatorul interacționează cu interfața grafică oferită de Angular. Folosind componentele expuse de Angular, acesta poate comunica atât cu serverul de Java, cât și cu cel Python prin servicii REST. Cele două servere de Backend nu interacționează direct între ele. În schimb, ele comunică direct cu baza de date distribuită Elasticsearch, iar între Angular și Elasticsearch nu există legătură directă. Toată aplicația (CPU, memorie folosită etc) este monitorizată folosind Grafana. Aceste caracteristici sunt redată și în figura de mai jos.



## Tehnologii folosite

Weather Application folosește următoarele tehnologii:

- *Angular* - pentru Frontend
- *Java* cu server Grizzly - pentru Backend
- *Python* cu server Flask - pentru Backend
- *Elasticsearch* - pentru stocarea datelor
- *Grafana* - pentru vizualizarea metricilor
- *Prometheus* – pentru monitorizarea aplicației

### Angular

#### Descriere

Angular este un framework pentru aplicații Web și Mobile. Oferind o structură organizată a proiectului bazată pe componente, este o alegere ideală pentru aplicația curentă. Angular oferă toată interfața grafică (pagina de autentificare, pagina principală, locațiile, opțiunile etc).

#### Locație Github

- <https://github.com/IrinaM09/DockerApp/blob/master/angular>

#### Locație Dockerhub

- [docker pull neileen/dockerhub\\_app:latest](#)

#### Dockerfile

```
# base image
FROM node:12.13.0

# set working directory
WORKDIR /app-angular

# cache app dependencies (needed by npm install)
COPY /angular/package.json ./package.json

# install
RUN npm install
RUN npm install -g @angular/cli@8.3.25
RUN npm install --save-dev @angular-devkit/build-angular

# add app
COPY /angular ./

# listen on port
EXPOSE 4200

# start app
CMD ng serve --host 0.0.0.0
```

### Java

#### Descriere

Aplicația folosește Maven Web Application scris în limbajul de programare Java, un server Grizzly și Javax pentru apeluri REST.

## Locație Github

- <https://github.com/IrinaM09/DockerApp/blob/master/java>

## Locație Dockerhub

- [docker pull neileen/dockerhub\\_app:java-latest](#)

## Dockerfile

```
# fetch basic image
FROM maven:3.6.1-jdk-8

# working directory
WORKDIR /app-java

# add the POM file
COPY /java/pom.xml ./

# install dependencies
RUN mvn install

# rest of the project
COPY /java/src ./src
RUN mvn package

# local application port
EXPOSE 8000

# run app
CMD ["mvn", "clean"]
CMD ["mvn", "install"]
CMD ["mvn", "exec:java"]
```

## Python

### Descriere

Python este folosit atât pentru a interacționa ușor cu baza de date Elasticsearch, cât și pentru a oferi posibilitatea de extindere a aplicației pe viitor. Se folosește serverul Flask care oferă și servicii REST.

## Locație Github

- <https://github.com/IrinaM09/DockerApp/blob/master/python>

## Locație Dockerhub

- [docker pull neileen/dockerhub\\_app:python-latest](#)

## Dockerfile

```
# base image
FROM python:3.7

# set the application directory
WORKDIR /app-python

# Install requirements.txt
ADD /python/requirements.txt ./requirements.txt
RUN pip install -r requirements.txt

# add app
```

```
COPY /python/main.py ./
COPY /python ./

# listen on port
EXPOSE 5002

# start app
CMD python main.py
```

## Elasticsearch

### Descriere

Elasticsearch este o bază de date distribuită care funcționează pe bază de indexare, fiind foarte rapidă. Este folosită pentru persistarea datelor despre utilizatori.

### Locație Github

- <https://github.com/IrinaM09/DockerApp/blob/master/elasticsearch>

### Locație Dockerhub

- [docker.elastic.co/elasticsearch/elasticsearch:7.6.1](https://hub.docker.com/r/docker.elastic.co/elasticsearch/elasticsearch:7.6.1)

### Dockerfile

```
# service name
es:
  image: docker.elastic.co/elasticsearch/elasticsearch:7.6.1
  ports:
    - "9200:9200"
    - "9300:9300"
  environment:
    - discovery.type=single-node
  networks:
    - elastic
...
# custom networks
networks:
  elastic:
    driver: bridge
```

## Prometheus

Prometheus este un o platformă folosită la monitorizarea sistemelor și a bazelor de date. În cazul nostru ea va fi folosită la monitorizarea containerelor Docker.

### Descriere

### Locație Github

- <https://github.com/IrinaM09/DockerApp/blob/master/prometheus>

### Locație Dockerhub

- [prom/prometheus:latest](https://hub.docker.com/r/prom/prometheus:latest)

## Dockerfile

```
# service name
image: prom/prometheus:latest
ports:
  - "9090:9090"
restart: unless-stopped
volumes:
  - type: bind
    source: E:\Docker\DockerApp\prometheus
    target: /prometheus
command:
  - '--config.file=/prometheus/prometheus.yml'
networks:
  - elastic
...
# custom networks
networks:
  elastic:
    driver: bridge
```

## Grafana

### Descriere

Grafana este o platformă open-source de afișare a diferite metrice, oferind diferite sisteme sau baze de date ca sursă. În cazul nostru ea e folosită la vizualizarea metricilor pentru containere oferite de Prometheus.

### Locație Github

- <https://github.com/IrinaM09/DockerApp/blob/master/grafana>

### Locație Dockerhub

- [grafana/grafana](https://hub.docker.com/r/grafana/grafana)

## Dockerfile

```
# service name
grafana:
  image: grafana/grafana
  ports:
    - "3000:3000"
...
# custom networks
networks:
  elastic:
    driver: bridge
```

## Stiva de servicii (docker-compose)

```
version: "3.7"
services:
  # container 1
  es:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.6.1
    ports:
      - "9200:9200"
      - "9300:9300"
    environment:
      - discovery.type=single-node
```



```
    networks:
      - elastic

# container 2
java-app:
  image: neileen/dockerhub_app:java-latest
  ports:
    - "8080:8080"
  depends_on:
    - es
  networks:
    - elastic

# container 3
angular-app:
  image: neileen/dockerhub_app:latest
  ports:
    - "4200:4200"
  networks:
    - elastic
  depends_on:
    - es

# container 4
python-app:
  image: neileen/dockerhub_app:python-latest
  ports:
    - "5002:5002"
  depends_on:
    - es
  networks:
    - elastic

# container 5
prometheus:
  image: prom/prometheus:latest
  ports:
    - "9090:9090"
  restart: unless-stopped
  volumes:
    - type: bind
      source: E:\Docker\DockerApp\prometheus
      target: /prometheus
  command:
    - '--config.file=/prometheus/prometheus.yml'
  networks:
    - elastic

# container 6
grafana:
  image: grafana/grafana
  ports:
    - "3000:3000"

networks:
  elastic:
    driver: bridge
```

# Rezultate

