Piotr Łach gl 2.2/3

Proszę napisać applikację, dzięki której można będzie:

- podać na stronie www określoną wartość całkowitą dodatnią K,
- wyświetlić wyliczoną na jej podstawie wartość K-tego elementu ciągu Fibonacciego
-

W sprawozdaniu proszę podać procedurę postępowania i opracowane pliki, które realizują zadania przedstawione na slajdzie 18 (ETAP NR1).

Sprawozdanie proszę wykonać w postaci repozytorium na GitHub a na moodle jedynie wgrać plik z linkiem do tego repozytorium.

Zawartość aplikacji, plik App.js:

```
import logo from './logo.svg';
import './App.css';
import React,{useState} from  'react';
import { Component } from 'react';


function App() {


const [fPos, setFPos] = useState('');
const [fWynik, setFWynik] = useState('');

function calculateFib(){
    var a=1;
    var b=1;
    if(fPos<1)setFWynik('Podano liczbę niedodatnią');
    if(fPos==1||fPos==2)setFWynik('1');
    if(fPos>2){
        var wynik=0;
        for(var i=2;i<fPos;++i){
            wynik=a+b;
            a=b;
            b=wynik;
        }
        setFWynik(wynik);
    }

}
```

Testy do aplikacji: App.test.js:

```
import { render, screen } from '@testing-library/react';
import App from './App';
```

```
test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/Podaj numer pozycji/i);
  expect(linkElement).toBeInTheDocument();
});
```

Plik dockerfile deweloperski, Dockerfile.dev3:

```
FROM node:alpine

WORKDIR '/app'
ENV NODE_OPTIONS=--openssl-legacy-provider
COPY package.json .
RUN npm install

RUN chown -R node:node /app

CMD ["npm", "run", "start"]
```

Plik dockerfile produkcyjny, Dockerfile:

```
FROM node:alpine
ENV NODE_OPTIONS=--openssl-legacy-provider
WORKDIR '/app'
COPY package.json .
RUN npm install
COPY . .
RUN npm run build

FROM nginx
EXPOSE 80
COPY --from=0 /app/build /usr/share/nginx/html
```

Plik docker-compose do testowania wykorzystania wolumenów do śledzenia wpływu zmian na funkcjonowanie usługi, docker-compose_old.yml:

```
version: '3'
services:
  web:
    stdin_open: true
    environment:
      - CHOKIDAR_USEPOLLING=true
    build:
      context: .
      dockerfile: Dockerfile.dev3
    ports:
```

```
      - '3000:3000'
    volumes:
      - /app/node_modules
      - .:/app
```

Plik docker-compose do testowania rozwoju usługi równolegle z testami, docker-compose2.yml:

```
version: '3'
services:
  web:
    stdin_open: true
    environment:
      - CHOKIDAR_USEPOLLING=true
    build:
      context: .
      dockerfile: Dockerfile.dev3
    ports:
      - '3000:3000'
    volumes:
      - /app/node_modules
      - .:/app

  test:
    stdin_open: true
    environment:
      - CHOKIDAR_USEPOLLING=true
    build:
      context: .
      dockerfile: Dockerfile.dev3
    volumes:
      - /app/node_modules
      - .:/app
    command: ["npm", "run", "test"]
```

Plik docker-compose do wersji “produkcyjnej”, docker-compose.yml:

```
version: '3'
services:
  web:
    stdin_open: true
    environment:
      - CHOKIDAR_USEPOLLING=true
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "80:80"
```

Link do repozytorium: link, wersja do skopiowanie: https://github.com/IrayOblivion/flowDocker

Należy pobrać repozytorium, potem w terminalu wejść do folderu frontend.

Następnie w celu uruchomienia kontenera do testowania zmian w kodzie z wolumenami:

DOCKER_BUILDKIT=1 docker compose -f docker-compose_old.yml up --build

Sprawdzenie działania wejście na localhost:3000

Uruchomienie kontenra do rozwoju usługi równolegle z testami.

W trakcie spokojnie można zmieniać zawartość kodu w pliku App.js

DOCKER_BUILDKIT=1 docker compose -f docker-compose2.yml up --build

Sprawdzenie działania wejście na localhost:3000

Uruchomienie kontenera z wersją produkcyjną:

DOCKER_BUILDKIT=1 docker compose -f docker-compose.yml up --build

Sprawdzenie działania wejście na localhost:80