

Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра автоматики та управління в технічних системах

# Front-End-програмування

## Лабораторна робота 2

Варіант №21

Виконав:

студент групи ІА-72

Чорнолоз Денис

Дата здачі

Захищено з балом

Перевірив:

викладач кафедри АУТС

Жереб К. А.

Київ 2021

## Завдання

Необхідно встановити та налаштувати інструментальні засоби для front-end розробки. Результатом виконання даної лабораторної роботи є docker-образ з встановленими інструментальними засобами, а також звіт про використані технології та можливості. У docker-образі можна також додати компоненти, що відповідають за backend.

## Посилання на проект

[https://github.com/Velns/FrontEnd\\_Lab2/tree/master](https://github.com/Velns/FrontEnd_Lab2/tree/master)

## Хід роботи

У ході даної лабораторної роботи було створено docker образ, який містить налаштування для розгортання створеного у лабораторній роботі 1 додатку.

Docker являє собою стандартизований пакетний програмний застосунок для створення та розгортання програмних комплексів.

В межах нашого проекту він слугує для розгортання середовища та запуску контейнеру в якому знаходиться розроблений сайт.

Першим кроком необхідно налаштувати майбутній контейнер.

Створюємо пустий файл з іменем «Dockerfile» та заповнюємо його налаштуваннями, де вказуємо початковий образ, задаємо змінні, вказуємо робочу директорію, задаємо початкові команди, передаємо файли проекту всередину контейнеру та задаємо порт.

```
FrontEnd_Lab1.github.io-main > Dockerfile > ...
1  FROM node:14
2  ENV NODE_ENV=development
3
4  # Create app directory
5  WORKDIR /usr/src/app
6
7  # Install app dependencies
8  # A wildcard is used to ensure both package.json AND package-lock.json are copied
9  # where available (npm@5+)
10 COPY package*.json ./
11
12 RUN npm install
13 # If you are building your code for production
14 # RUN npm ci --only=production
15
16 # Bundle app source
17 COPY . .
18
19 EXPOSE 8085
20
21 CMD [ "npm", "start" ]
```

Рисунок 1 – Файл Dockerfile

Наступним кроком необхідно створити файл docker-compose.yml, в якому задаються пов'язані служби.

```

FrontEnd_Lab1.github.io-main > 🐙 docker-compose.yml
1  version: '3.4'
2
3  services:
4    usergallery:
5      image: denyshornoloz/frent-end-lab2
6      build:
7        context: .
8        dockerfile: ./Dockerfile
9      environment:
10     NODE_ENV: development
11     ports:
12     - 9090:8085
13

```

Рисунок 2 – Файл docker-compose.yml

Далі створимо файл package.json. У цьому файлі описується наш додаток та вказуються його зв'язки.

```

FrontEnd_Lab1.github.io-main > {} package.json > ...
1  {
2    "name": "user-gallery",
3    "version": "1.0.0",
4    "description": "own project",
5    "main": "server.js",
6    ▶ Debug
7    "scripts": {
8      "test": "echo \"Error: no test specified\" && exit 1",
9      "start": "node server.js"
10   },
11   "author": "Denys Chornoloz",
12   "license": "MIT",
13   "dependencies": {
14     "bootstrap": "^5.0.0-beta1",
15     "connect": "^3.7.0",
16     "serve-static": "^1.14.1"
17   }
18 }

```

Рисунок 3 – Файл package.json

Після слід створити файл server.js записи в якому визначають наш веб додаток.

```
FrontEnd_Lab1.github.io-main > JS server.js > ...
1  |var connect = require('connect');
2  |var serveStatic = require('serve-static');
3  |const port = 8085;//8080
4
5  |connect()
6  |  .use(serveStatic("./app"))
7  |  .listen(port, () => console.log(`Server running on ${port}...`));
```

Рисунок 3 – Файл server.js

В результаті маємо директорію представлену на рисунку 5.

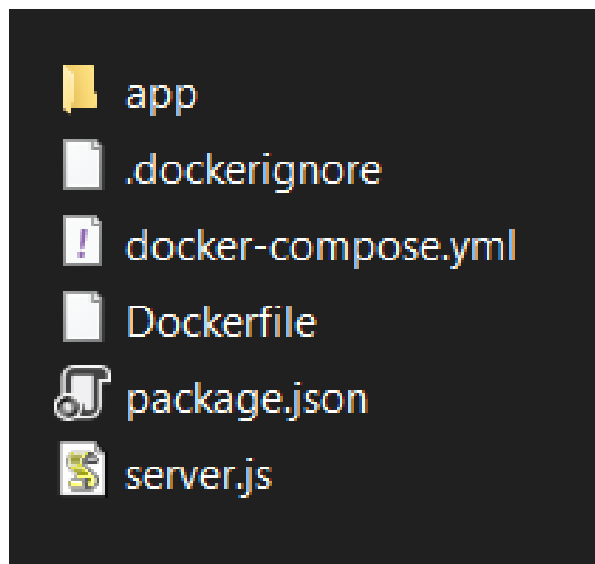


Рисунок 5 – Директорія з підготовленим до розгортання додатком

На наступному етапі виконується побудова контейнеру.

Для цього необхідно:

1. Запустити термінал
2. Перейти в директорію з проектом
3. виконати команду docker-compose up

Результатом описаних вище маніпуляцій виступатиме повідомлення про успішні побудову та запуск контейнеру (рисунок 6).

```
Cmder
Stopping frontend_lab1githubio-main_usergallery_1 ... done

C:\Users\Laiks\Desktop\FrontEnd_Lab1.github.io-main (user-gallery@1.0.0)
λ docker-compose up --build
Building usergallery
Step 1/8 : FROM node:14
----> 7bef16bb2cf1
Step 2/8 : ENV NODE_ENV=development
----> Using cache
----> e72ae57fe6de
Step 3/8 : WORKDIR /usr/src/app
----> Using cache
----> 63aaed95808f
Step 4/8 : COPY package*.json ./
----> b2b5481b4dbf
Step 5/8 : RUN npm install
----> Running in c42f163716d8
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN bootstrap@5.0.0-beta2 requires a peer of @popperjs/core@^2.6.0 but none is installed. You m
ust install peer dependencies yourself.
npm WARN user-gallery@1.0.0 No repository field.
npm WARN user-gallery@1.0.0 No license field.

added 26 packages from 20 contributors and audited 26 packages in 1.553s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities

Removing intermediate container c42f163716d8
----> 202f6af657da
Step 6/8 : COPY . .
----> ef274460b36a
Step 7/8 : EXPOSE 8085
----> Running in 32f86a05a632
Removing intermediate container 32f86a05a632
----> c38a63184fd2
Step 8/8 : CMD [ "npm", "start" ]
----> Running in 561ae402c905
Removing intermediate container 561ae402c905
----> 14f762e2ffa7

Successfully built 14f762e2ffa7
Successfully tagged denyshornoloz/frent-end-lab2:latest
Recreating frontend_lab1githubio-main_usergallery_1 ... done
Attaching to frontend_lab1githubio-main_usergallery_1
usergallery_1 |
usergallery_1 | > user-gallery@1.0.0 start /usr/src/app
usergallery_1 | > node server.js
usergallery_1 |
usergallery_1 | Server running on 8085...
```

Рисунок 6 – Запущенный docker контейнер

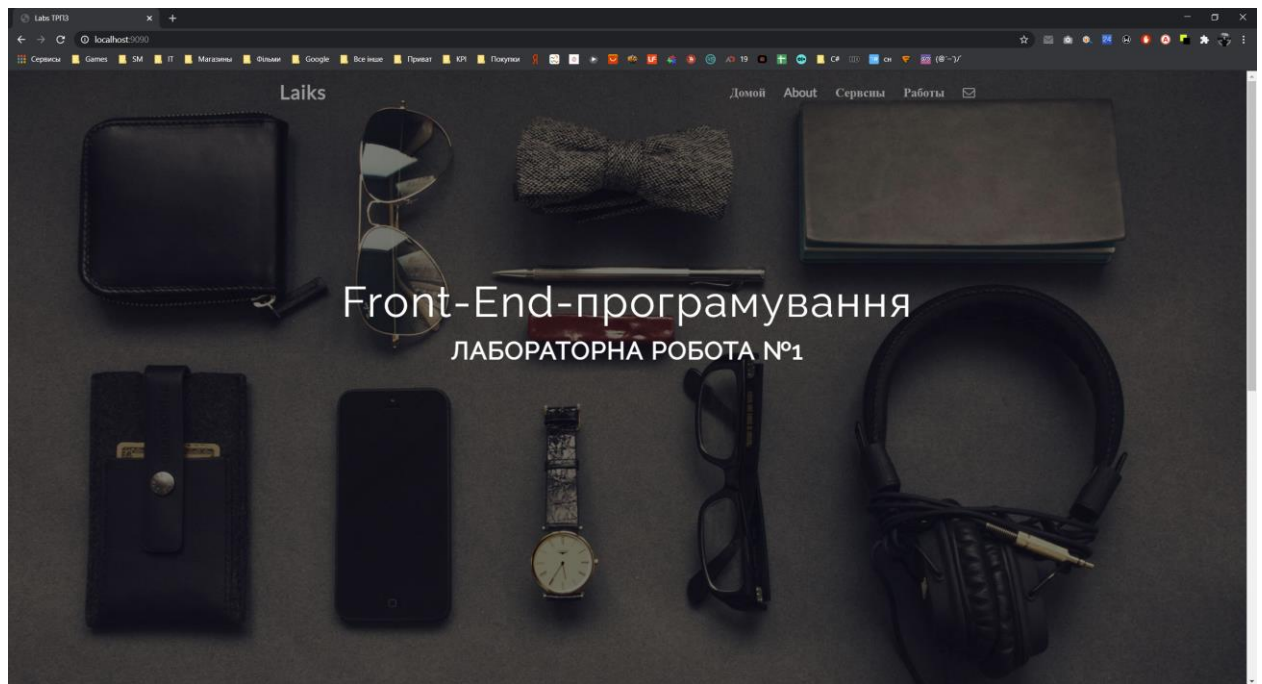


Рисунок 7 – Результат роботи