МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ  
  
  
  
  
  
  
  
  
  
Дисципліна «Сучасні каркаси web-додатків»

Лабораторна робота № 1

Виконала: студентка гр. 8.1213  
 Абдуллаєва Севда   
 Самандар Кизи

Перевірив: Лебідь Максим   
 Володимирович

Запоріжжя  
2024

**Тема**

Розгортання простого Vue.js додатку з використанням Docker.

**Мета**

Оволодіти основними кроками створення, налаштування та розгортання Vue.js додатку, а також ознайомитися з використанням контейнеризації за допомогою Docker для легкого розповсюдження та збереження середовища розробки.

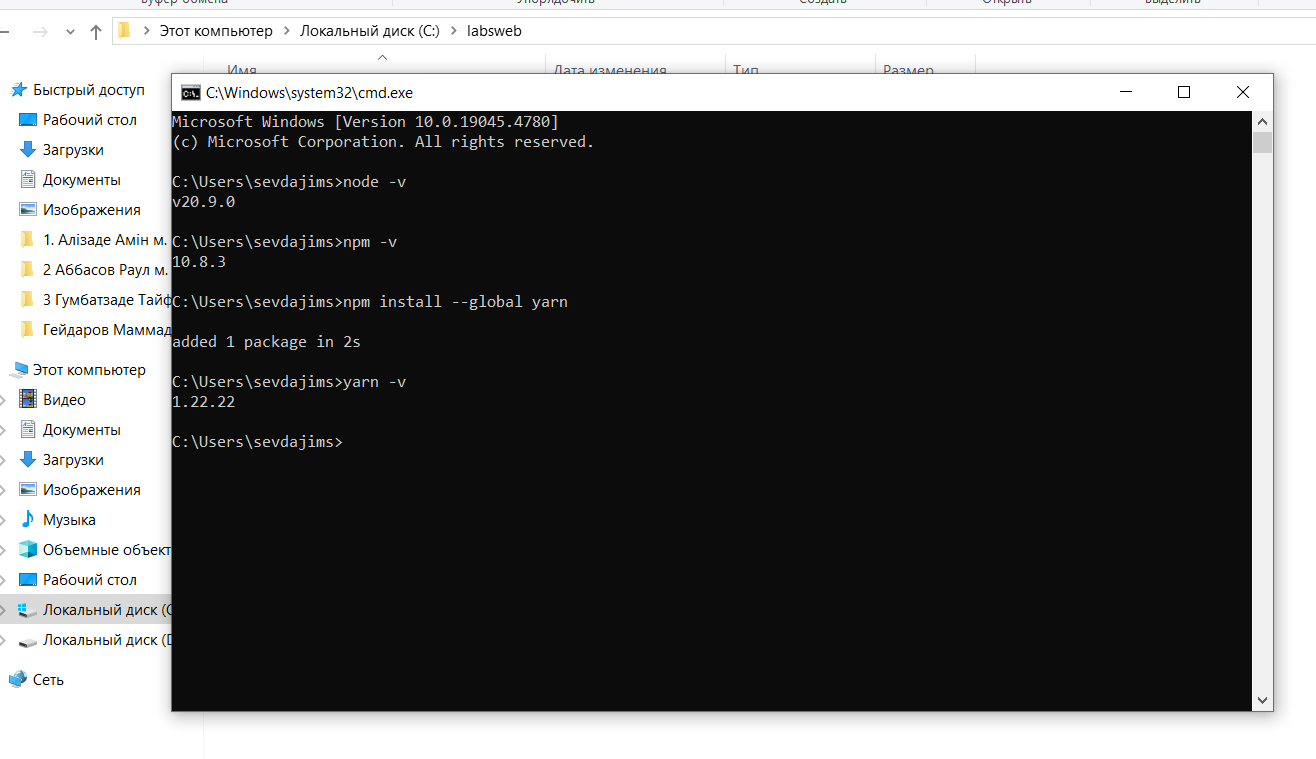
**Теоретичні відомості**

Vue.js - прогресивний фреймворк для створення веб-інтерфейсів, що спрощує розробку веб-додатків та дозволяє ефективно керувати їхнім станом і компонентами.

**Завдання**

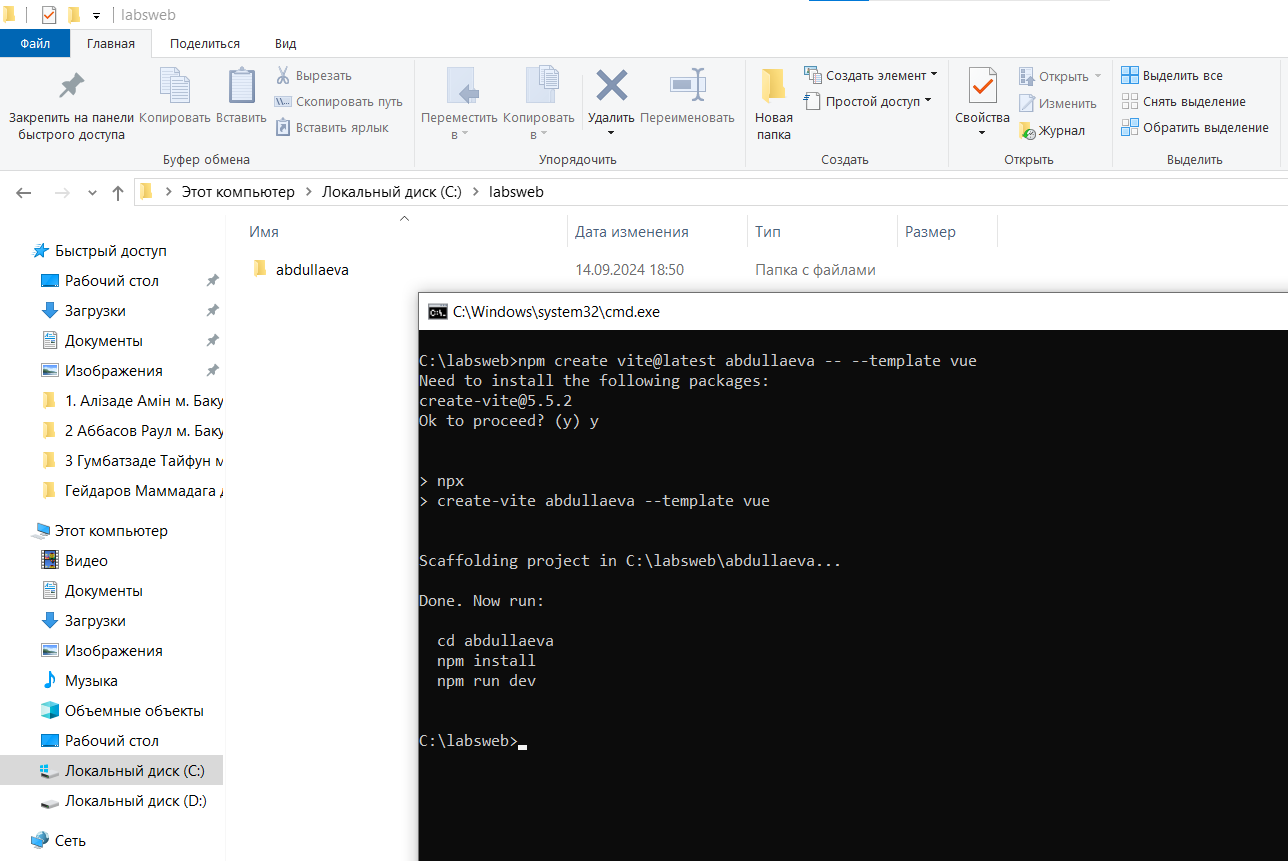
1. Ознайомтеся з документацією Vue.js, щоб зрозуміти основи роботи з фреймворком. Документацію можна знайти на офіційному сайті Vue.js.

2. Встановлення Node.js та npm, yarn на вашому комп'ютері.



3. Створіть новий проект Vue.js за допомогою Vite

npm create vite@latest <ваше-прізвище-англійською> -- --template vue

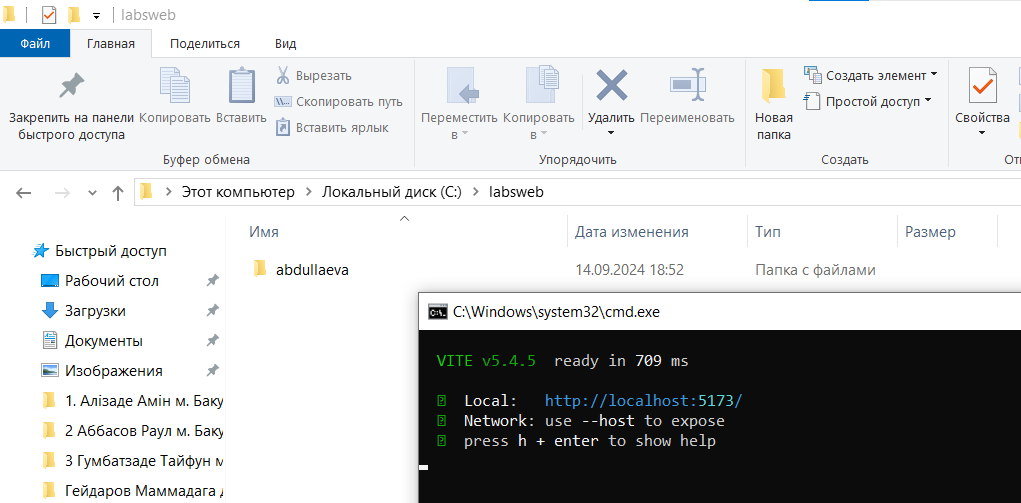


4. Запустіть проект та переконайтеся, що він працює коректно.

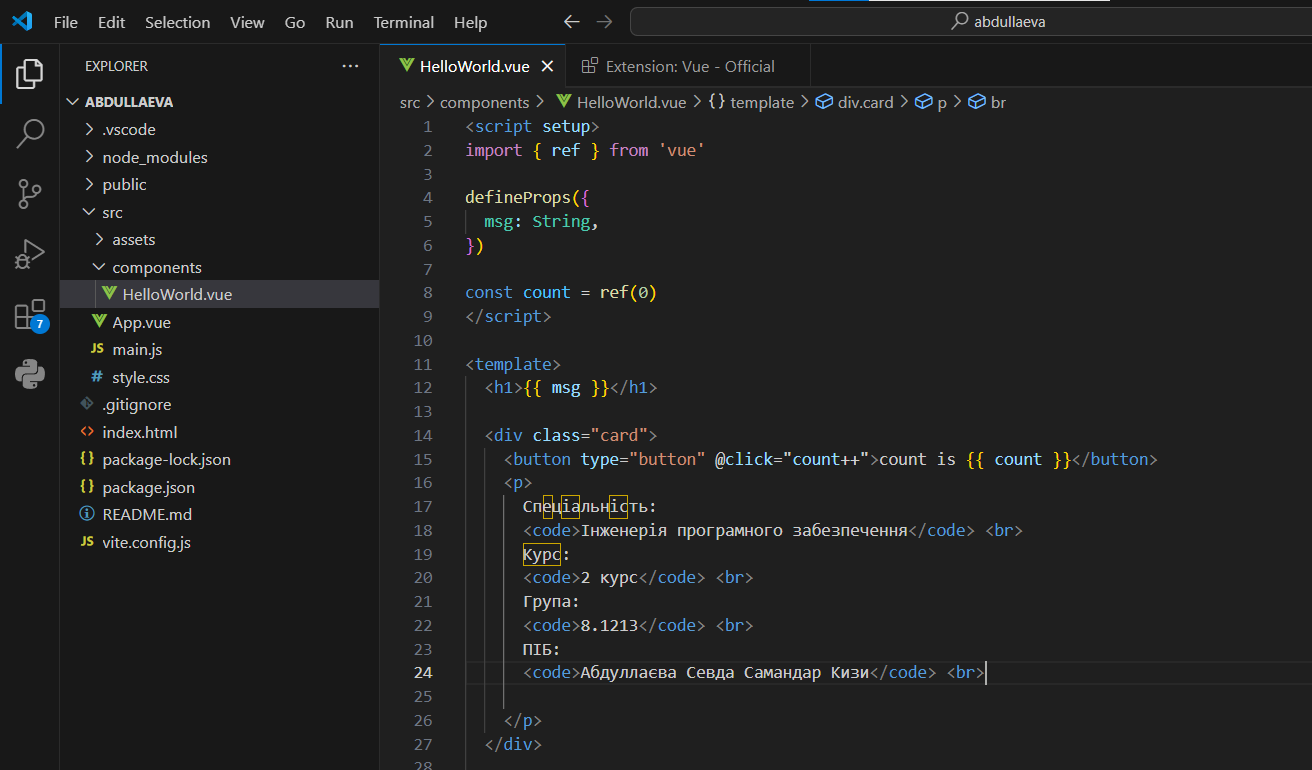
cd <ваше-прізвище-англійською>

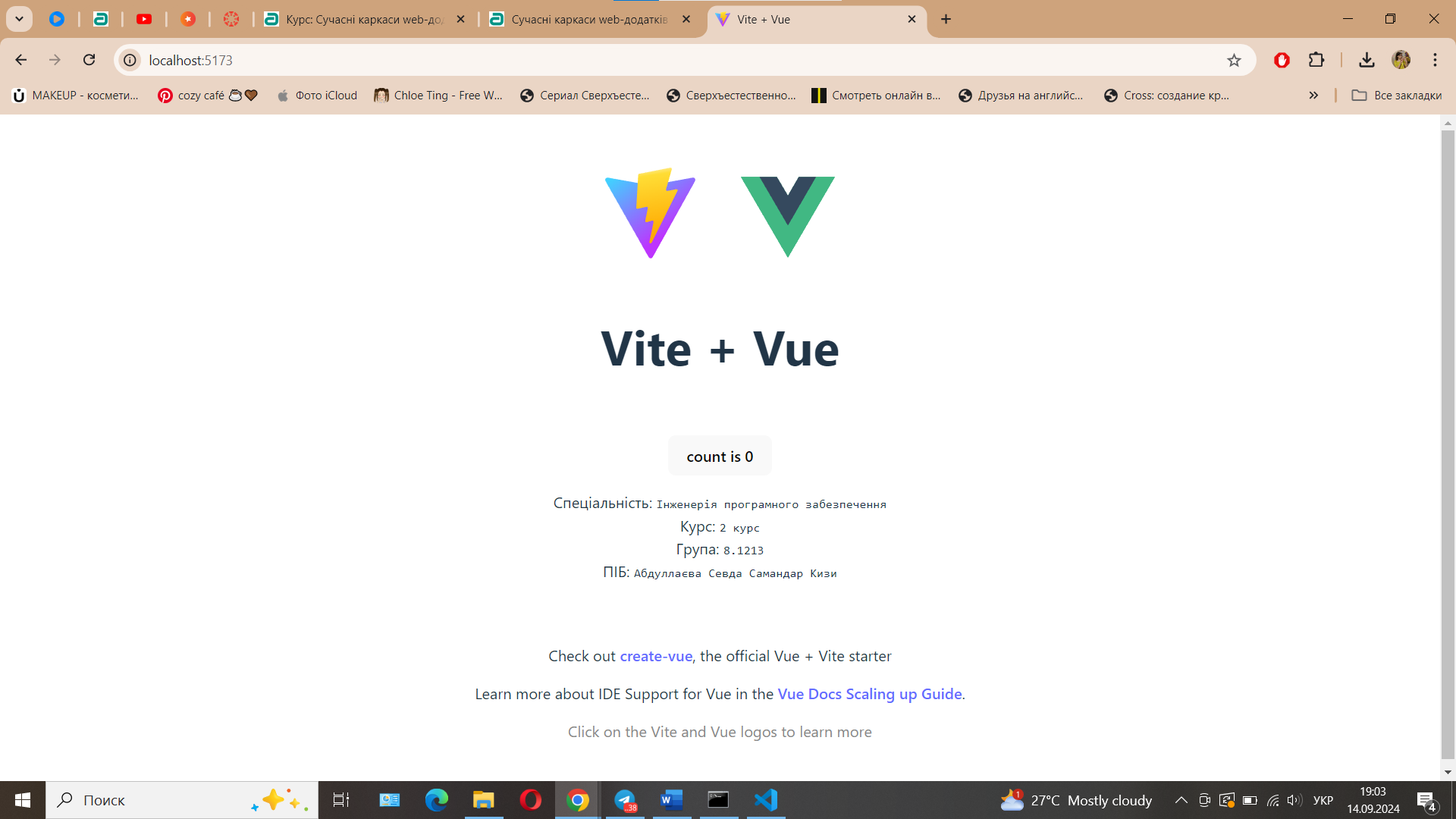
npm install

npm run dev



5. Знайдіть компонент HelloWorld.vue та додайте інформацію про себе (Спеціальність, курс, група, ПІБ).





6. Створіть Dockerfile для збірки та запуску Vue.js додатку.

# Build stage

FROM node:latest as build-stage

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

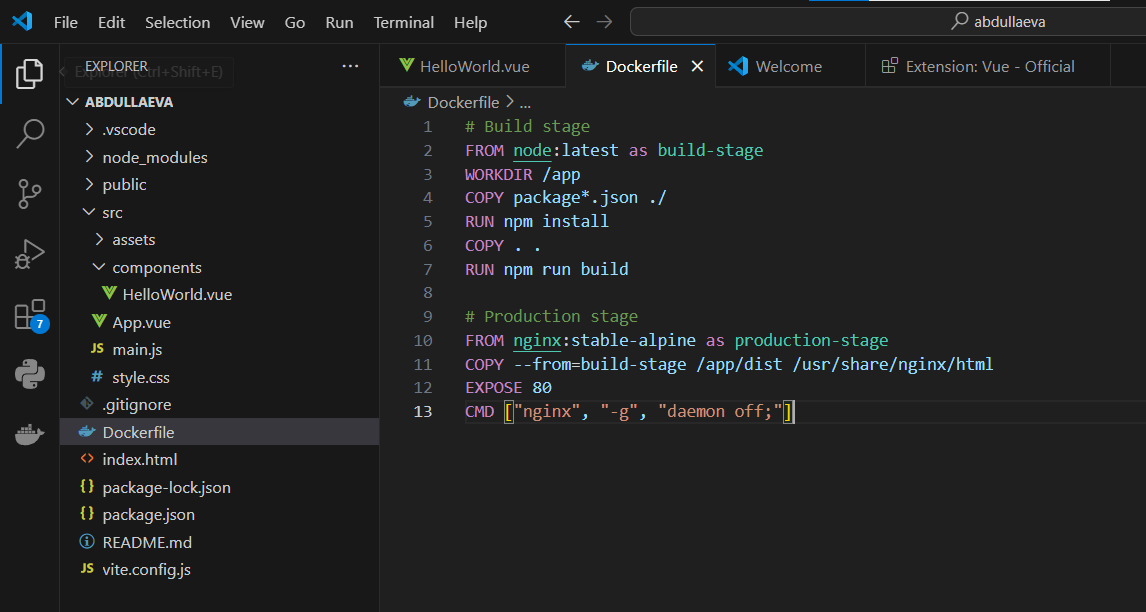
# Production stage

FROM nginx:stable-alpine as production-stage

COPY --from=build-stage /app/dist /usr/share/nginx/html

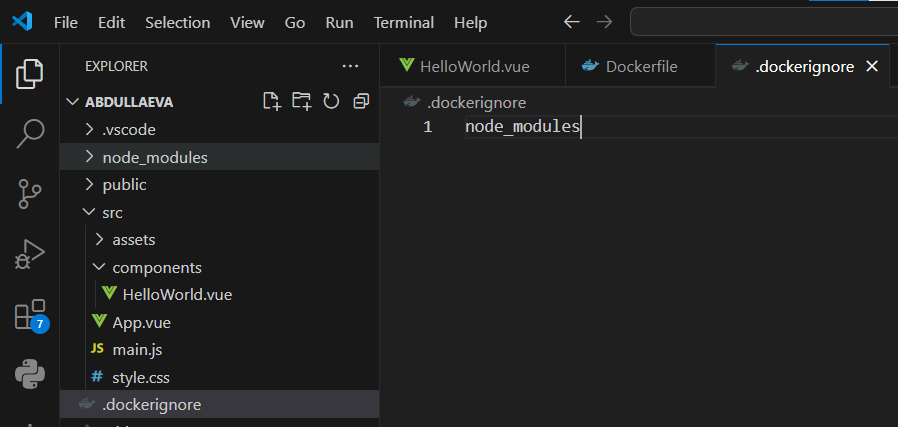
EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

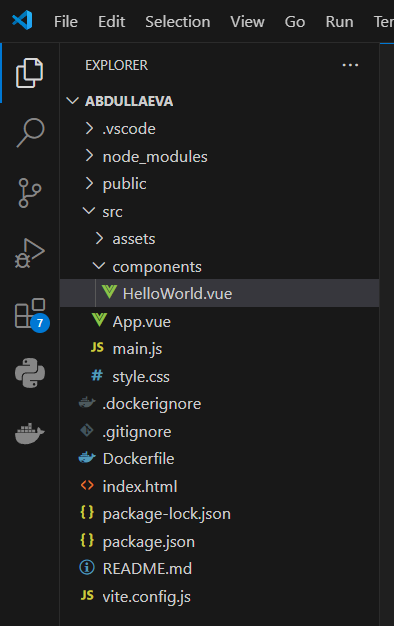


7. Додайте .dockerignore для виключення непотрібних файлів:

node\_modules



8. Збережіть Dockerfile та .dockerignore у кореневій папці проекту.



9. Додайте до package.json наступні команди:

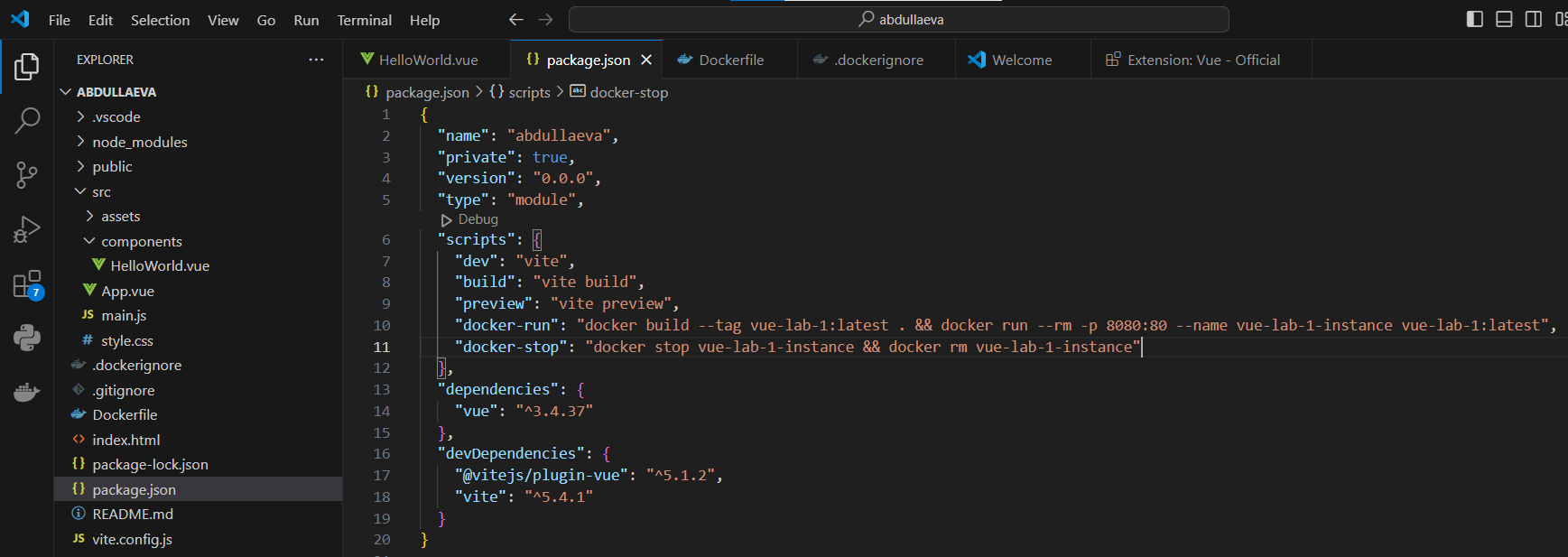
"scripts": {

...

"docker-run": "docker build --tag vue-lab-1:latest . && docker run --rm -p 8080:80 --name vue-lab-1-instance vue-lab-1:latest",

"docker-stop": "docker stop vue-lab-1-instance && docker rm vue-lab-1-instance",

}



10. Розмістіть ваш проект в директорії lab1 публічномого репозиторію GitHub/GitLab/Bitbucket з назвою <vendor>/mjsf-mag та додайте посилання на репозиторій у звіт виконавши наступні команди:

git init

git add .

git commit -m "Vue.js lab 1"

git branch -M main

git remote add origin <посилання-на-репозиторій>

git push -u origin main

**Посилання на директорію:** <https://github.com/SevdaAbdullaeva/Modern-frames-of-web-applications.git>

**Контрольні питання**

**1. Які кроки потрібно виконати перед початком роботи з Vue.js?**

Перед тим, як почати працювати з Vue.js, потрібно виконати кілька підготовчих кроків. Спочатку слід встановити Node.js та npm. Це можна зробити, завантаживши останню версію Node.js з офіційного сайту, що також включає npm. Після установки перевірте, чи все працює, запустивши команди node -v та npm -v у терміналі. Далі необхідно встановити Vue CLI, використовуючи команду npm install -g @vue/cli. Перевірте, чи Vue CLI встановлено правильно, запустивши команду vue --version.

Наступним кроком є створення нового проекту Vue.js. Для цього використовуйте команду vue create my-vue-project, де my-vue-project — це назва вашого проекту. Вас запросять вибрати налаштування для проекту. Після створення проекту перейдіть до його директорії командою cd my-vue-project.

Запустіть розробницький сервер, ввівши npm run serve. Це дозволить вам переглянути ваш проект у браузері за адресою http://localhost:8080. Ознайомтесь із документацією Vue.js, щоб краще зрозуміти основи фреймворку. Якщо плануєте використовувати системи контролю версій, налаштуйте Git для вашого проекту. Також може бути корисно встановити редактор коду з підтримкою Vue.js, наприклад, Visual Studio Code.

**2. Що таке Vue CLI і як його встановити?**

Vue CLI (Command Line Interface) — це офіційний інструмент командного рядка для створення та керування проектами на базі Vue.js. Він спрощує налаштування нового проекту, управління його конфігураціями та додавання різних функціональностей, таких як плагіни або налаштування розробницького середовища.

Щоб встановити Vue CLI, спочатку переконайтеся, що у вас встановлений Node.js і npm. Якщо це вже зроблено, відкрийте термінал або командний рядок і виконайте наступну команду:

npm install -g @vue/cli

Ця команда встановлює Vue CLI глобально на вашій системі, що дозволяє вам використовувати його з будь-якого місця в командному рядку. Після завершення встановлення ви можете перевірити, чи Vue CLI успішно встановлено, запустивши команду:

vue --version

Ця команда покаже версію Vue CLI, яка була встановлена, підтверджуючи, що установка пройшла успішно. Тепер ви готові використовувати Vue CLI для створення нових проектів і керування ними.

**3. Як створити новий Vue.js проект?**

Щоб створити новий Vue.js проект, вам слід скористатися Vue CLI. Після того, як ви встановили Vue CLI, ви можете створити новий проект за допомогою простих команд у терміналі. Ось як це зробити:

Відкрийте термінал або командний рядок.

Виконайте команду:

vue create my-vue-project

Замініть my-vue-project на бажане ім’я вашого проекту.

Після виконання команди вас буде запитано вибрати налаштування для вашого нового проекту. Ви можете вибрати один із стандартних шаблонів або створити власну конфігурацію, вибравши опцію "Manually select features". Це дозволяє вам налаштувати плагіни, підтримку TypeScript, Linter, і інші функції.

Після вибору налаштувань Vue CLI створить новий проект і завантажить усі необхідні залежності.

Після завершення процесу створення перейдіть до директорії вашого проекту:

cd my-vue-project

Запустіть розробницький сервер, щоб побачити ваш проект у браузері:

npm run serve

Відкрийте браузер і перейдіть за адресою http://localhost:8080, щоб переглянути ваш новий Vue.js додаток.

Це дозволить вам створити базову структуру проекту, налаштувати середовище розробки та почати роботу над вашим Vue.js додатком.

**4. Що таке Docker і для чого він використовується?**

Docker — це платформа для автоматизації розробки, доставки та запуску програм у контейнерах. Контейнери дозволяють упакувати програму разом з усіма її залежностями та конфігураціями в один зручний для переносу пакет. Це забезпечує однорідність середовища розробки та продуктивності на різних системах.

Основні переваги Docker:

* Ізоляція: Контейнери забезпечують ізоляцію програм, що дозволяє уникнути конфліктів між різними версіями бібліотек та залежностей.
* Портативність: Завдяки тому, що контейнери включають всі необхідні залежності, програми можуть бути запущені на будь-якому сервері, який підтримує Docker, без потреби в додатковій конфігурації.
* Ефективність: Контейнери використовують ресурси ефективніше ніж віртуальні машини, оскільки вони поділяють один і той же хост-ядро і не потребують окремих операційних систем.
* Швидкість: Створення та запуск контейнерів зазвичай відбувається швидше ніж запуск віртуальних машин, оскільки контейнери мають меншу накладну витрату на ресурсах.
* Консистентність: Docker дозволяє забезпечити однакові середовища для розробки, тестування та продакшн, що допомагає уникнути проблем, які можуть виникнути через різницю в середовищах.
* Масштабованість: Docker легко інтегрується з інструментами оркестрації, такими як Kubernetes, що дозволяє ефективно масштабувати та управляти великими розподіленими системами.

Docker складається з кількох основних компонентів:

* Docker Engine: Серверна частина, що управляє контейнерами.
* Docker Image: Шаблон для створення контейнерів. Образи є основними компонентами, які містять код, бібліотеки, інструменти та середовище, необхідне для роботи програми.
* Docker Container: Запущений екземпляр образу. Контейнери є легковажними та ізольованими середовищами, в яких програми можуть виконуватись.
* Docker Hub: Сервіс для зберігання та розповсюдження Docker образів.
* Docker спрощує процес розробки та розгортання програм, роблячи його більш передбачуваним і керованим.

**5. Як створити Dockerfile для Vue.js додатку?**

Щоб створити Dockerfile для Vue.js додатку, вам потрібно виконати кілька простих кроків. Спочатку створіть файл з ім'ям Dockerfile у кореневій директорії вашого Vue.js проекту.

В Dockerfile спочатку задайте базовий образ, який буде використовуватися для зборки додатку. Зазвичай для цього використовують образ node. Вкажіть робочу директорію всередині контейнера, скопіюйте файли package.json та package-lock.json, щоб встановити залежності за допомогою npm install. Потім скопіюйте весь код вашого додатку та запустіть команду npm run build, яка створить зібрані файли вашого Vue.js додатку.

Наступним етапом є створення образу для продакшн середовища. Для цього використовується образ nginx, який буде обслуговувати статичні файли. Скопіюйте зібрані файли з попереднього етапу до директорії Nginx, що відповідає за статичні файли, а саме /usr/share/nginx/html. Відкрийте порт 80 для доступу до сервісу та вкажіть команду для запуску Nginx у фоновому режимі.

Ваш Dockerfile виглядатиме наступним чином:

# Build stage

FROM node:latest as build-stage

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

# Production stage

FROM nginx:stable-alpine as production-stage

COPY --from=build-stage /app/dist /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

Цей Dockerfile розділяє процес на два етапи: спочатку створюється зібраний додаток, а потім він розгортається у продакшн середовищі за допомогою Nginx. Це забезпечує ефективне використання ресурсів і спрощує розгортання вашого Vue.js додатку.

**6. Які файли потрібно включати до .dockerignore?**

Файл .dockerignore використовується для вказання файлів і директорій, які потрібно виключити з контексту побудови Docker образу. Це допомагає уникнути копіювання непотрібних або чутливих файлів у Docker образ, що може зменшити розмір образу та прискорити процес побудови.

Ось які файли та директорії зазвичай включають до .dockerignore:

* node\_modules: Ця директорія містить залежності, які вже будуть встановлені всередині контейнера, тому її не потрібно копіювати.
* .git: Директорія .git містить інформацію про версійний контроль і не потрібна всередині контейнера.
* .env: Файли конфігураційного середовища можуть містити чутливу інформацію і не повинні бути включені до образу.
* .DS\_Store: Цей файл створюється на MacOS для зберігання метаданих директорій і не потрібен у контейнері.
* dist або build: Якщо ви вже створюєте зібрані файли всередині контейнера, не потрібно включати ці директорії з локальної розробки.
* logs або інші директорії з логами: Логи зазвичай не потрібні в контейнері, тому їх краще не включати.

Dockerfile: Зазвичай Dockerfile не потрібно включати в образ, оскільки він використовується тільки для побудови образу.

Ось приклад .dockerignore файлу для Vue.js проекту:

node\_modules

.git

.env

.DS\_Store

dist

logs

Цей файл забезпечує, що непотрібні або чутливі дані не потрапляють в Docker образ, що допомагає уникнути зайвого розміру образу та підвищує безпеку.

**7. Як завантажити проект до репозиторію на GitHub або Bitbucket?**

Щоб завантажити проект до репозиторію на GitHub або Bitbucket, спочатку створіть новий репозиторій на вибраній платформі. На GitHub або Bitbucket увійдіть у свій акаунт і створіть новий репозиторій, вказавши назву і налаштування (публічний або приватний).

Після цього відкрийте термінал або командний рядок і перейдіть до директорії вашого проекту. Ініціалізуйте локальний репозиторій Git, виконавши команду git init. Потім додайте всі файли до стадії коміту за допомогою git add . і створіть перший коміт командою git commit -m "Initial commit".

Далі додайте URL вашого нового репозиторію як віддалений репозиторій. Для GitHub це буде команда git remote add origin [URL репозиторію], а для Bitbucket відповідно. Після цього запуште ваш код до віддаленого репозиторію за допомогою git push -u origin master. Це завантажить ваш проект до репозиторію, і ви зможете продовжувати працювати з ним через Git.

**8. Які переваги систем контролю версій у розробці**?

Системи контролю версій пропонують численні переваги в процесі розробки програмного забезпечення. По-перше, вони забезпечують відстеження змін у коді, що дозволяє розробникам бачити, хто і коли вносив які зміни. Це важливо для розуміння еволюції проекту і допомагає при виявленні помилок.

По-друге, системи контролю версій дозволяють працювати над різними гілками проекту одночасно. Це означає, що розробники можуть створювати нові функціональності або виправляти помилки в окремих гілках, не заважаючи основній версії проекту. Після завершення роботи, зміни можуть бути об'єднані з основною гілкою, що спрощує управління функціональністю та виправленням помилок.

По-третє, системи контролю версій забезпечують можливість повернення до попередніх версій коду. Якщо нові зміни спричинили проблеми, можна легко повернутися до стабільної версії без втрати даних. Це значно знижує ризик втрати важливої інформації.

По-четверте, вони підтримують співпрацю між кількома розробниками. Завдяки системам контролю версій, кілька людей можуть одночасно працювати над проектом, і зміни кожного розробника будуть інтегровані в основний проект без конфліктів.

По-п'яте, системи контролю версій дозволяють зберігати історію змін, що є корисним для документації та аудиту. Це також допомагає при аналізі історії проекту і розумінні, які рішення були прийняті на різних етапах розробки.

Взагалі, системи контролю версій є невід'ємною частиною сучасного процесу розробки програмного забезпечення, забезпечуючи організацію, безпеку і ефективність управління кодом.