Звіт з лабораторної роботи № 1  
студента групи 8.1212   
Кішкіна Олександра Івановича

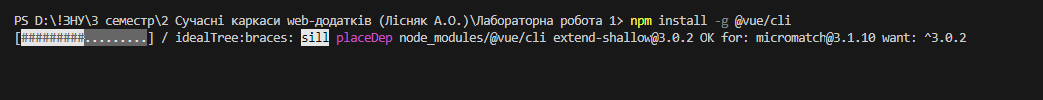
1. **Виконати налаштування робочого середовища (встановити останню стабільну версию сервера NodeJs та менеджера залежностей npm).**

Виконали.

1. **За допомогою утиліти vue-cli створити елементарний додаток з назвою "ваше прізвище англійскою"**

Для створення елементарного додатку з назвою "ваше прізвище англійською" за допомогою Vue CLI, потрібно встановити Vue CLI, для цього робимо такі кроки:

Крок 1: Встановити Node.js. Встановити Vue CLI, виконавши наступну команду в командному рядку: npm install -g @vue/cli



Крок 2: Створення нового проекту Після встановлення Vue CLI ви можете створити новий проект за допомогою наступної команди:

vue create Kishkin

Крок 3: Запуск локального сервера розробки Після створення проекту переходимо до його каталогу за допомогою команди:

cd Kishkin

Тепер запускаємо локальний сервер розробки за допомогою команди:

npm install

npm run dev

Ця команда запустить сервер розробки, і ви побачите URL для доступу до вашого додатку в браузері <http://localhost:8080/>

Крок 4: Тестування додатку Відкрили браузер та перейшли за URL, який вивела попередня команда. Побачили створений Vue.js додаток з назвою " Kishkin ". Відредагували та розширювати додаток, змінюючи файли у каталозі проекту.

1. Відкрили файл HelloWorld.vue у редакторі коду.
2. Знайшли шаблон компонента (<template>) і додали необхідну інформацію про себе.:

html

<template>

<div class="hello">

<h1>{{ message }}</h1>

<p>Спеціальність: Спеціальність: 121 Інженерія програмного забезпечення</p>

<p>Курс: 2</p>

<p>Група: 8.1212</p>

<p>ПІБ: Кішкін О.І. </p>

</div>

</template>

1. Зберегли файл після внесення змін.
2. Для перегляду зміни, перезапускаємо локальний сервер розробки, командою

Після запуску сервера, відкрили браузер і перейшли за URL додатку. Ви побачите компонент HelloWorld і в ньому буде відображатися інформація.

1. Протестувати роботу створеного додатку засобами vue-cli.
2. Створили файли Dockerfile та .dockerignore
3. Розмістили проект у системі контролю версій (github.com, bitbucket.org) з іменем ропозиторію mjsf-mag
4. Відправили посилання на репозиторій у якості звіту виконаної роботи. <https://github.com/KishkinA/lab1>

**Контрольні питання**

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

Перед тим як розпочати роботу з Vue.js, слід виконати декілька підготовчих кроків:

1. **Встановлення Node.js та npm**: Vue.js вимагає наявності Node.js і пакетного менеджера npm (Node Package Manager) на вашому комп'ютері. Ви можете завантажити та встановити Node.js з [офіційного сайту Node.js](https://nodejs.org/).
2. **Встановлення Vue CLI (командний рядок інструменту для Vue.js)**: Vue CLI - це офіційний інструмент командного рядку для створення та керування Vue.js проектами. Ви можете встановити його глобально за допомогою npm:

bash

 npm install -g @vue/cli

 **Створення нового Vue.js проекту**: Після встановлення Vue CLI ви можете створити новий проект за допомогою команди vue create:

bash

 vue create my-vue-project

Де my-vue-project - назва вашого нового проекту.

 **Вибір конфігурації проекту**: Під час створення проекту Vue CLI запитає вас про конфігурацію. Ви можете вибрати базовий проект або проект з додатковими опціями, такими як роутер (Vue Router) або станове керування (Vuex).

 **Запуск локального сервера розробки**: Після створення проекту, ви можете запустити локальний сервер розробки, щоб переглядати та тестувати свій додаток в режимі реального часу. Виконайте наступну команду у каталозі проекту:

bash

1. npm run serve
2. **Навчання Vue.js**: Ознайомтесь з документацією Vue.js, офіційним посібником та прикладами для кращого розуміння основ та концепцій Vue.js. Vue.js має добру документацію, яка допоможе вам швидко вивчити цей фреймворк.

Після виконання цих підготовчих кроків, ви будете готові розпочати розробку веб-додатків з використанням Vue.js.

* **Що таке Vue CLI і як його встановити глобально на комп'ютері?**

Vue CLI (Command Line Interface) - це офіційний інструмент командного рядку для розробки Vue.js додатків. Він надає широкий спектр можливостей, включаючи створення нових проектів, розробку, тестування і пакування додатків. Vue CLI також пропонує багато корисних плагінів і інструментів для спрощення розробки Vue.js додатків.

Для встановлення Vue CLI глобально на комп'ютері, слід виконати наступні кроки:

1. Відкрийте командний рядок або термінал на вашому комп'ютері.
2. Виконайте команду для встановлення Vue CLI за допомогою npm (Node Package Manager):

bash

 npm install -g @vue/cli

Ця команда встановить Vue CLI глобально на вашому комп'ютері. Опція -g означає, що пакет буде встановлений глобально і буде доступний в будь-якому каталозі.

 Після успішного встановлення Vue CLI ви можете перевірити, чи він встановлений, виконавши команду:

bash

vue --version

1. Ця команда покаже версію Vue CLI, якщо встановлено правильно.

Тепер Vue CLI готовий до використання на вашому комп'ютері. Ви можете створити нові Vue.js проекти, керувати налаштуваннями проектів і використовувати всі переваги цього інструмента для розробки веб-додатків на Vue.js.

* **Як створити новий Vue.js проект за допомогою Vue CLI?**

Для створення нового Vue.js проекту за допомогою Vue CLI слід виконати наступні кроки:

1. Відкрийте командний рядок або термінал на вашому комп'ютері.
2. Перейдіть в каталог, в якому ви хочете створити новий Vue.js проект. Ви можете використовувати команду cd для цього. Наприклад:

bash

 cd ваша-папка-проекту

 Виконайте команду vue create, де <назва-проекту> - це ім'я вашого нового проекту. Вам також буде запропоновано вибрати конфігурацію для вашого проекту:

bash

vue create <назва-проекту>

Наприклад:

bash

 vue create my-vue-project

 Після запуску команди vue create, Vue CLI запитає вас про конфігурацію проекту. Виберіть варіант, який відповідає вашим потребам, або використовуйте значення за замовчуванням. Наприклад, ви можете вибрати базовий проект або проект з роутером (Vue Router) і/або становим керуванням (Vuex).

 Зачекайте, доки Vue CLI створить новий проект і встановить всі залежності.

 Після завершення процесу створення проекту, перейдіть в каталог проекту за допомогою команди cd:

bash

cd <назва-проекту>

Наприклад:

bash

 cd my-vue-project

 Тепер ви можете запустити локальний сервер розробки для перегляду вашого нового Vue.js проекту:

bash

1. npm run serve
2. Відкрийте веб-браузер і перейдіть за адресою http://localhost:8080/, щоб побачити свій новий Vue.js додаток у дії.

Тепер у вас є новий Vue.js проект, готовий для розробки. Ви можете редагувати файл src/App.vue та інші файли в каталозі проекту, щоб створити свій власний веб-додаток на Vue.js.

* **Як запустити створений Vue.js додаток для перевірки його роботи?**

Для запуску створеного Vue.js додатку і перевірки його роботи використовується локальний сервер розробки. Ви можете запустити додаток за допомогою Vue CLI. Ось кроки для запуску Vue.js додатку:

1. Відкрийте командний рядок або термінал.
2. Перейдіть в каталог проекту Vue.js, де знаходиться ваш створений проект. Використовуйте команду cd, щоб перейти до каталогу:

bash

cd шлях-до-каталогу-проекту

 Після переходу в каталог проекту виконайте команду npm run serve, яка запустить локальний сервер розробки:

bash

npm run serve

или npm run dev

1. Після виконання цієї команди, Vue CLI повідомить вас про те, на якому порту запущений сервер. Зазвичай це буде http://localhost:8080/.
2. Відкрийте веб-браузер та перейдіть за адресою http://localhost:8080/, щоб побачити свій Vue.js додаток у дії.

Тепер ви можете перевірити та тестувати свій Vue.js додаток на локальному сервері розробки. Ви можете вносити зміни в свій код, і сервер автоматично оновлюватиме відображення на веб-сторінці у реальному часі.

* **Що таке Docker і для чого він використовується в розробці програмного забезпечення?**

Docker - це платформа для контейнеризації програмного забезпечення. Вона дозволяє упаковувати програмне забезпечення та його залежності в контейнери, які можна легко переносити, розгортати і запускати на будь-якому комп'ютері або сервері. Контейнери Docker вирішують проблеми, пов'язані з різними середовищами розробки, розгортанням та масштабуванням програмного забезпечення. Ось декілька ключових аспектів Docker і його використання в розробці програмного забезпечення:

1. **Ізольованість середовища**: Docker контейнери ізолюють програмне забезпечення та його залежності, що дозволяє уникнути конфліктів між різними версіями бібліотек і компонентів. Ви можете впакувати ваш додаток і всі його залежності в контейнер, забезпечуючи однаковість середовища як на вашому робочому столі, так і на сервері.
2. **Переносність**: Docker контейнери є переносними і можуть бути виконані на будь-якій системі, яка підтримує Docker, незалежно від оперативної системи або сервера. Це спрощує розгортання програмного забезпечення на різних платформах.
3. **Легкість розгортання**: Ви можете легко розгортати контейнери Docker на серверах або в хмарних середовищах. Все, що вам потрібно, - це Docker-сервер та контейнери, які містять ваше програмне забезпечення. Docker надає інструменти для автоматизації розгортання.
4. **Швидкість розгортання**: Docker контейнери можуть бути розгорнуті дуже швидко, оскільки вони містять в собі все необхідне для роботи додатка. Це особливо корисно для мікросервісної архітектури, де кожен сервіс може бути упакований в окремий контейнер.
5. **Масштабованість**: Docker дозволяє легко масштабувати програмне забезпечення за допомогою контейнерів. Ви можете запускати багато екземплярів контейнера для обробки великого навантаження.
6. **Версіонування**: Docker контейнери можуть бути версіоновані, що дозволяє зберігати різні версії вашого додатка та забезпечувати можливість відкату до попередніх версій.
7. **Спрощений розробка та тестування**: Docker полегшує розробку та тестування, оскільки ви можете створити контейнери, які включають ваше програмне забезпечення та всі залежності. Це дозволяє швидко створювати

* **Які основні етапи створення Dockerfile для розгортання Vue.js додатку?**

Створення Dockerfile для розгортання Vue.js додатку включає в себе кілька основних етапів. Dockerfile - це текстовий файл, який містить інструкції для побудови Docker контейнера. Ось основні етапи створення Dockerfile для розгортання Vue.js додатку:

1. **Вибір базового образу**: Перший крок - вибрати базовий образ для вашого контейнера. Зазвичай ви можете використовувати офіційний образ Node.js або Nginx для статичних файлів. Наприклад:

Dockerfile

 FROM node:14

Цей приклад використовує офіційний образ Node.js версії 14.

 **Створення робочого каталогу**: Створіть робочий каталог в контейнері, де буде зберігатися ваш Vue.js додаток. Використовуйте інструкцію WORKDIR:

Dockerfile

 WORKDIR /usr/src/app

 **Копіювання файлів**: Скопіюйте файли вашого Vue.js додатку (наприклад, файли збудованого додатку) до контейнера за допомогою інструкції COPY:

Dockerfile

 COPY . .

Ця команда копіює всі файли з поточного каталогу вашого проекту в робочий каталог контейнера.

 **Встановлення залежностей (якщо потрібно)**: Якщо ваш Vue.js додаток вимагає додаткових залежностей, таких як npm пакети, встановіть їх за допомогою команди RUN. Наприклад:

Dockerfile

 RUN npm install

 **Збірка додатку (якщо потрібно)**: Якщо ви маєте збудувати ваш Vue.js додаток всередині контейнера, використовуйте команду RUN для запуску збірки. Наприклад:

Dockerfile

 RUN npm run build

 **Налаштування команди запуску**: Вкажіть команду, яку Docker має виконати при запуску контейнера. Наприклад:

Dockerfile

 CMD ["npm", "run", "start"]

 **Відкриття необхідних портів**: Якщо ваш додаток використовує певні порти, відкрийте їх у контейнері за допомогою інструкції EXPOSE. Наприклад:

Dockerfile

 EXPOSE 8080

 **Побудова контейнера**: Збережіть цей Dockerfile в каталозі вашого проекту і використовуйте команду docker build, щоб побудувати Docker контейнер:

bash

 docker build -t назва-контейнера .

Вкажіть назву контейнера та поточний каталог . як контекст для збірки.

 **Запуск контейнера**: Після побудови контейнера ви можете запустити його за допомогою команди docker run:

bash

1. docker run -p 8080:8080 -d назва-контейнера
2. З цією командою ваш Vue.js додаток буде запущений в Docker контейнері і буде доступний за адресою http://localhost:8080 на вашому комп'ютері.

Це загальний опис створення Dockerfile для розгортання Vue.js додатку. Ви можете налаштувати його додатково, враховуючи особливості вашого проекту і вимоги розгортання.

* **Які файли можна включити до файлу .dockerignore, і для чого це робиться?**

Файл .dockerignore використовується для визначення файлів та каталогів, які повинні бути виключені з контексту збірки Docker контейнера під час використання команди docker build. Це важливо для того, щоб зменшити розмір та обсяг контейнера, а також забезпечити безпечніше та ефективніше розгортання.

Ось деякі типові файли та каталоги, які можна включити до файлу .dockerignore, і для чого це робиться:

1. **node\_modules**: Каталог node\_modules містить всі залежності вашого проекту (наприклад, npm пакети), і включення його до контейнера зазвичай непотрібно, оскільки залежності можуть бути встановлені всередині контейера під час збірки.
2. **dist**: Каталог dist зазвичай містить збудовані файли вашого Vue.js додатку. Включення його до контейнера може займати зайвий простір, оскільки ви можете збудувати додаток всередині контейнера.
3. **.git**: Каталог .git містить інформацію про систему контролю версій Git, і включення його до контейнера зазвичай непотрібно.
4. **.dockerignore самого себе**: Додаючи .dockerignore до самого себе, ви забезпечуєте те, що цей файл не буде включений до контейнера.
5. **Зайві файли та каталоги**: Ви можете включити до .dockerignore будь-які файли або каталоги, які ви вважаєте зайвими або небезпечними для включення в контейнер.

Використання файлу .dockerignore допомагає зменшити обсяг контейнера, що полегшує його розгортання та зменшує час на збірку. Він також забезпечує безпеку, оскільки ви можете контролювати, які файли не потрапляють в контейнер і не використовуються в розгортанні.

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

Для завантаження вашого проекту до репозиторію на GitHub або Bitbucket слід виконати такі кроки:

### На GitHub:

1. **Створення репозиторію на GitHub**:
   * Увійдіть до свого облікового запису на GitHub.
   * Натисніть на плюс (+) у верхньому правому куті та виберіть "New repository."
   * Надайте репозиторію назву, опис та інші налаштування за бажанням.
   * Натисніть "Create repository."
2. **З'єднання локального репозиторію з віддаленим репозиторієм GitHub**:
   * В командному рядку перейдіть до каталогу вашого проекту.
   * Виконайте наступні команди:

bash

* + git init
  + git add .
  + git commit -m "Перший коміт"
  + git branch -M main
  + git remote add origin URL-репозиторію-GitHub
  + git push -u origin main

### На Bitbucket:

1. **Створення репозиторію на Bitbucket**:
   * Увійдіть до свого облікового запису на Bitbucket.
   * Натисніть на "+" у верхньому правому куті та виберіть "Repository."
   * Надайте репозиторію назву, опис та інші налаштування за бажанням.
   * Натисніть "Create repository."
2. **З'єднання локального репозиторію з віддаленим репозиторієм Bitbucket**:
   * В командному рядку перейдіть до каталогу вашого проекту.
   * Виконайте наступні команди:

bash

* + git init
  + git add .
  + git commit -m "Перший коміт"
  + git branch -M main
  + git remote add origin URL-репозиторію-Bitbucket
  + git push -u origin main

Зверніть увагу, що вам потрібно буде замінити "URL-репозиторію-GitHub" або "URL-репозиторію-Bitbucket" на фактичний URL вашого репозиторію на GitHub або Bitbucket.

Після виконання цих кроків ваш проект буде завантажений до віддаленого репозиторію, і ви зможете спільно працювати над ним з іншими розробниками або зберігати копії проекту в безпечному місці в мережі.

* **Які основні переваги використання системи контролю версій для розробки програмного забезпечення?**

Використання системи контролю версій (СКВ) для розробки програмного забезпечення має багато переваг, які полегшують роботу розробників та покращують управління проектами. Ось основні переваги використання СКВ:

1. **Історія змін**: СКВ веде історію всіх змін у вашому програмному коді. Це означає, що ви завжди можете переглядати, коли і які зміни внесені до коду. Історія змін дозволяє вам відстежувати, хто, коли і навіщо вніс зміни.
2. **Спільна робота**: Використання СКВ полегшує спільну роботу розробників над проектом. Кожен розробник може створювати свою гілку для роботи над певною функцією або виправленням помилок, а потім об'єднувати їх з іншими гілками.
3. **Відокремлення середовища розробки**: Розробники можуть працювати над своїми завданнями в ізольованих гілках, що дозволяє уникнути конфліктів між змінами і забезпечує вищу продуктивність.
4. **Резервне копіювання і відновлення**: СКВ дозволяє створювати резервні копії проекту та відновлювати попередні версії коду. Це корисно для відновлення випадково видалених або пошкоджених файлів.
5. **Синхронізація та розгортання**: СКВ дозволяє легко синхронізувати код між різними серверами і розгортати програмне забезпечення на різних середовищах без зайвого зусилля.
6. **Аналіз помилок та виправлення дефектів**: З історією змін ви можете встановлювати, коли і як помилка вперше з'явилася, що полегшує виявлення та виправлення дефектів.
7. **Відстеження версій та релізів**: Ви можете легко визначати версії вашого програмного забезпечення та створювати релізи, які дозволяють вам визначати, яка версія програми була випущена.
8. **Автоматизація тестування та розгортання**: СКВ інтегруються з іншими інструментами розробки, такими як CI/CD системи, що дозволяє автоматизувати тестування та розгортання програмного забезпечення.
9. **Можливість працювати офлайн**: Ви можете працювати над проектом навіть тоді, коли немає з'єднання з мережею, оскільки історія змін з