# **Державний вищий навчальний заклад**

# **«Запорізький національний університет»**

# **Міністерства освіти і науки України**

# **Лабораторна робота №4**

# 

# **Виконала студентка**

# **гр. 8.1212:**

# **Зінченко А.О.**

# **м. Запоріжжя**

# **2022-2023 н.р.**

**Завдання**

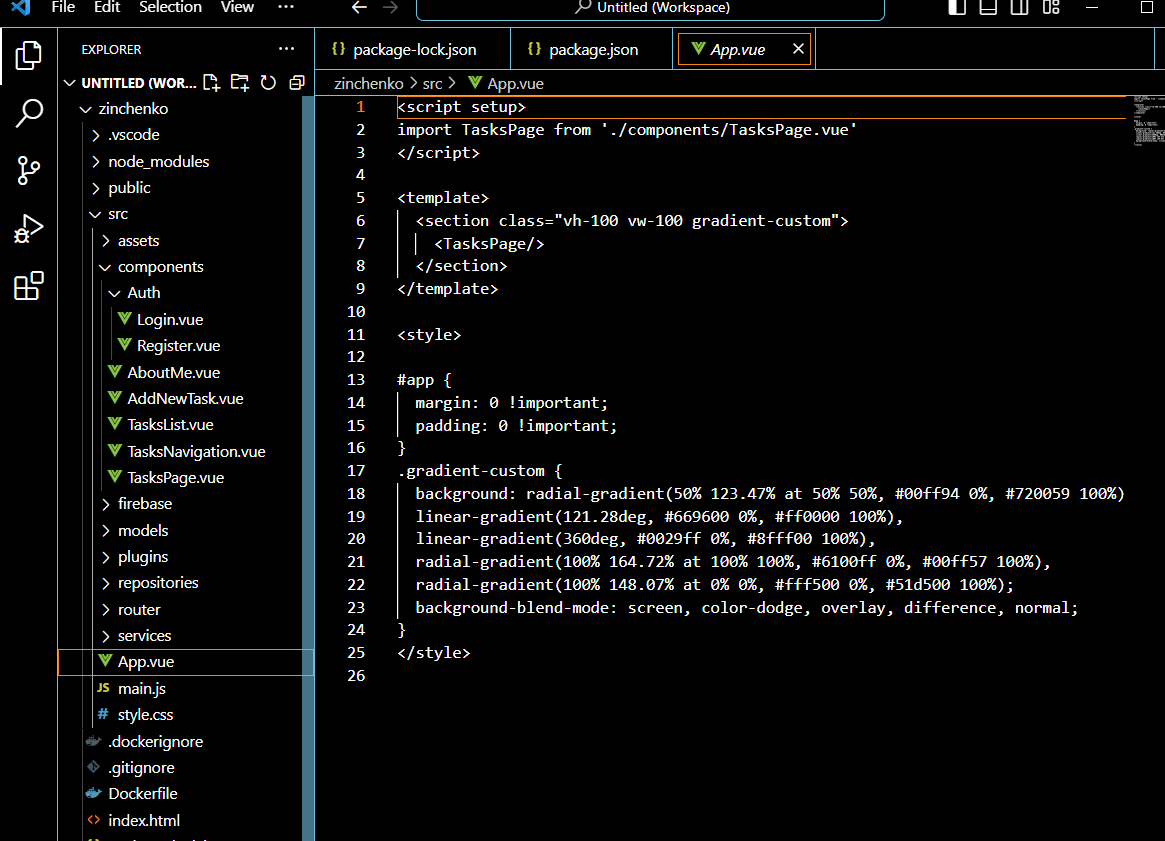
1. Проаналізувати[**статтю**](https://javascript.plainenglish.io/building-vue-enterprise-application-part-2-services-f7ec400190e7)**.**
2. Створити провайдер для сервісів створених у попередній лабораторній роботі.
3. Створити плагін для реестрації провайдеру.
4. Інтегрувати плагін та сервіс в компоненти створені в попередніх роботах.
5. Зберігти зміни в новому репозиторію з назвою mjsf-mag-lab4.
6. Надати посилання на репозиторій у якості звіту.

## **Хід роботи**

Стаття охоплює розробку фронтенд-додатку на базі Vue.js з використанням принципів чистої архітектури. Стаття розглядає створення сервісів, які відповідають за виконання основних бізнес-операцій. Основні моменти статті включають наступне:

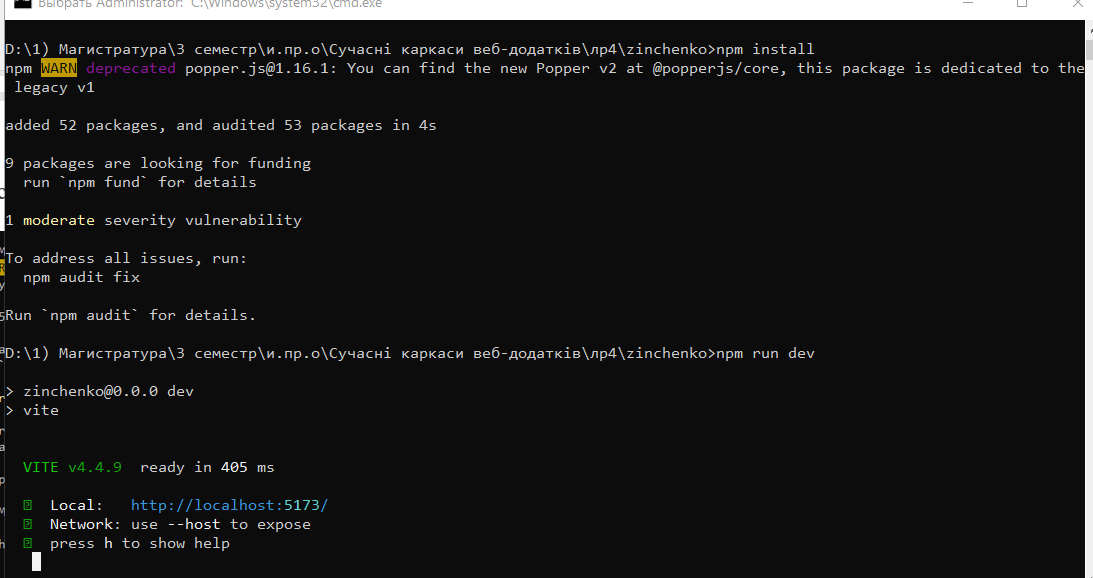
1. Стаття розглядає поняття сервісу в контексті Vue.js та чистої архітектури. Вона називає сервіс "чимось, що виконує основні бізнес-операції" та наголошує на тому, що сервіси не повинні мати залежностей від деталей, таких як фреймворки, UI тощо.
2. Стаття розпочинається з аналізу вимог для створення функціональності блогу, де потрібно реалізувати можлисть перегляду статей та залишення коментарів.
3. Автор започатковує розробку, використовуючи підхід розробки через тести. Він створює тести для функціональності сервісу перед написанням коду, що допомагає гарантувати правильну роботу.
4. Стаття розглядає структуру папок та файлів для організації сервісу та його тестів.
5. Автор обговорює важливість ін'єкції даних у сервіс та рекомендує внедрювати дані, замість жорсткого залежання від файлів чи інших джерел.
6. Стаття також розглядає паттерн сервісного провайдера, який допомагає інкапсулювати ініціалізацію сервісів та надавати їх іншим частинам додатку.

Стаття включає добре структурований код та приклади, які допомагають розібратися в концепціях та практичних аспектах розробки Vue.js додатків з використанням чистої архітектури. Ця стаття є частиною серії і може бути корисною для розробників, які цікавляться створенням більших та більш складних додатків з використанням Vue.js.

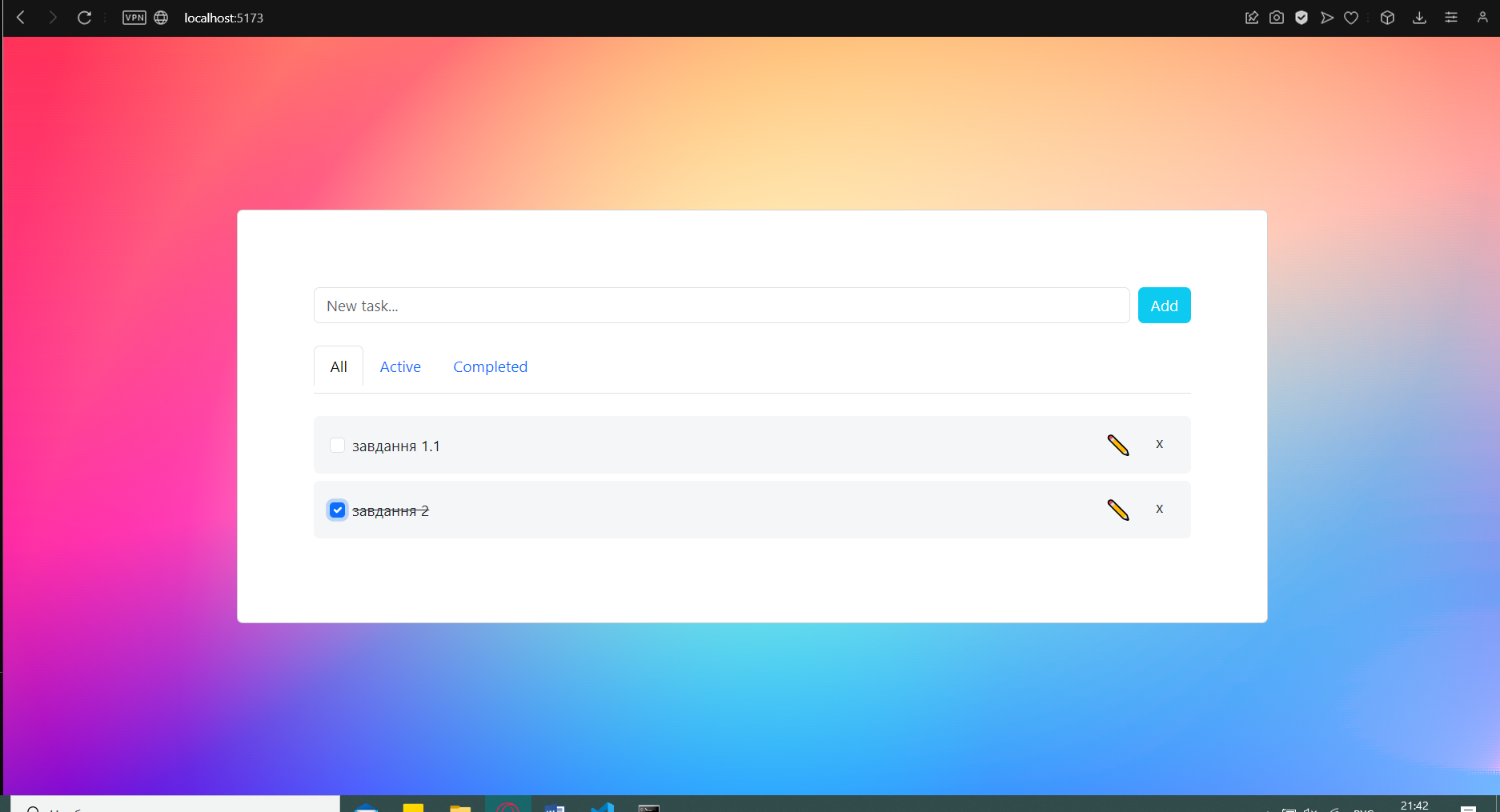


## **Результат**

Запуск:



Все працює.



## **Контрольні питання**

1. Що таке плагіни в контексті Vue.js 3 і як вони розширюють функціональність Vue?

Це спеціальні розширення, які дозволяють розширювати функціональність Vue.js. Вони використовуються для додавання глобальних або локальних функцій, директив, міксінів, компонентів та інших ресурсів до Vue.js додатку.

1. Як встановити плагін?

За допомогою методу app.use(plugin) на екземплярі Vue (app):

import { createApp } from 'vue';

import MyPlugin from './my-plugin';

const app = createApp(App);

app.use(MyPlugin);

1. Які два способи визначення плагіна підтримуються в Vue.js 3?

Об'єкт плагіна, який містить метод install. Цей метод буде викликаний при встановленні плагіна.

Функція викликається при встановленні плагіна і приймає два аргументи: app і будь-які інші параметри, які передаються при встановленні.

1. Які можливі сценарії використання плагінів Vue?

* Додавання глобальних компонентів, директив або міксінів.
* Розширення функціональності Vue і створення користувацьких методів та властивостей.
* Доступ до сторонніх бібліотек і сервісів.
* Спрощення роботи з HTTP запитами або локальним сховищем даних.

1. Як передати параметри конфігурації в плагін під час встановлення?

Функція:

app.use(MyPlugin, { option1: 'value1', option2: 'value2' });

Об'єкт конфігурації:

app.use(MyPlugin, { options: { option1: 'value1', option2: 'value2' } });

1. Як можна використовувати inject для надання функції або атрибута користувачам плагіна?

inject дозволяє віддавати значення або функції всередині компонентів, які використовують плагін. Наприклад, якщо плагін надає якусь функціональність, можна використовувати inject для передачі цієї функціональності компонентам, які використовують плагін.

Для прикладу, в плагіні:

const MyPlugin = { install(app) { app.config.globalProperties.$myFunction = someFunction; } };

В компоненті:

export default { created() { this.$myFunction(); // Викликаємо функцію, яку надав плагін } };

Це дозволяє компонентам легко взаємодіяти з функціональністю плагіна.