**ЗВІТ**

**до лабораторної роботи № 3**  
на тему:  
**"Розширення функціональності інформаційної системи."**

**Виконала:**  
студентка групи **МІТ-21**  
**Йовхимищ Діана**

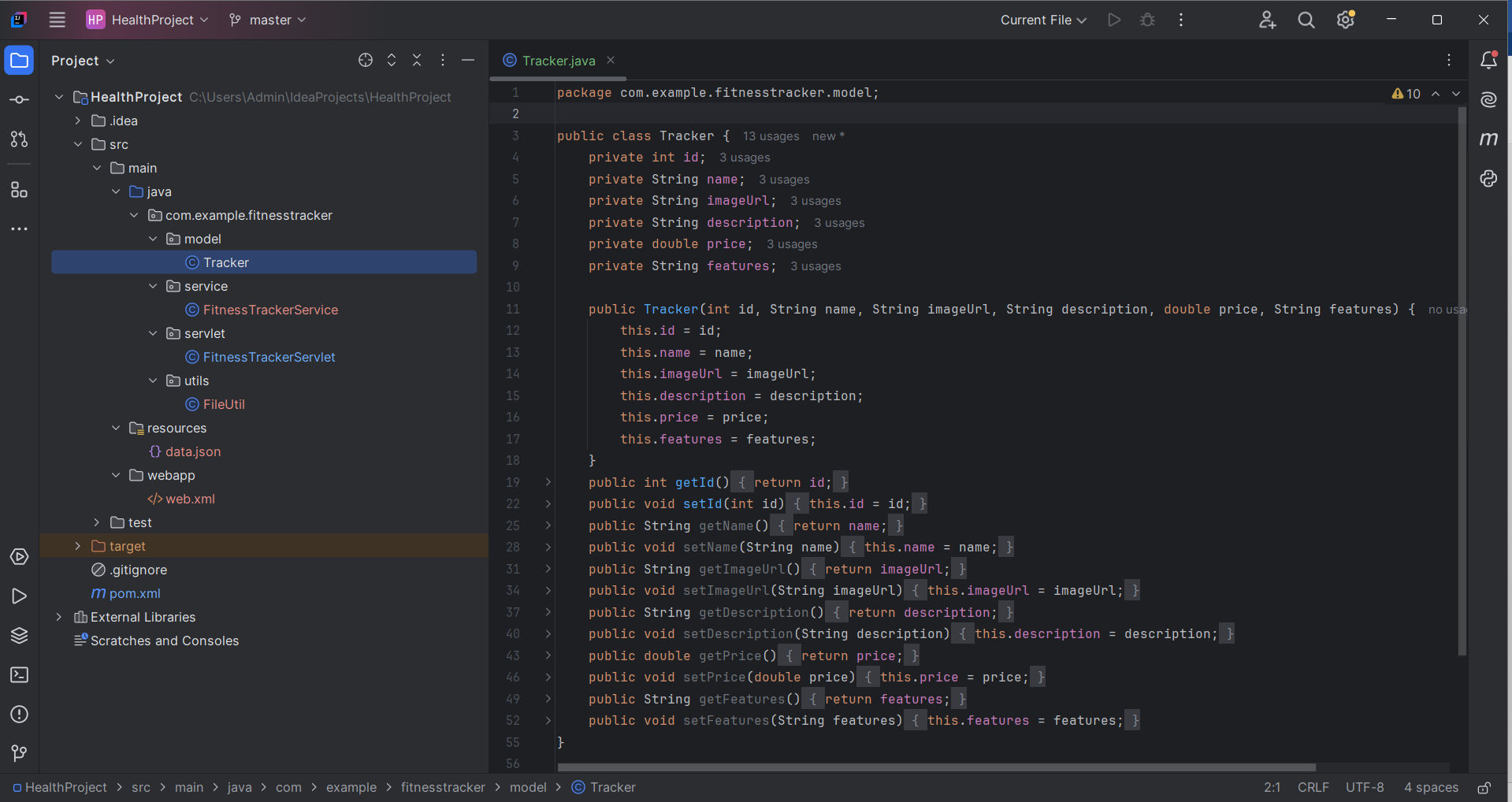
**Мета роботи:** Удосконалення інформаційної системи шляхом реалізації повного набору CRUD-операцій та збереження інформації у файл.

**Хід роботи:**

1. **Удосконалення Back-end**
2. Створення моделі даних:

Додати клас Tracker для опису об'єкта з унікальним id.

Додати геттери та сеттери для всіх полів.

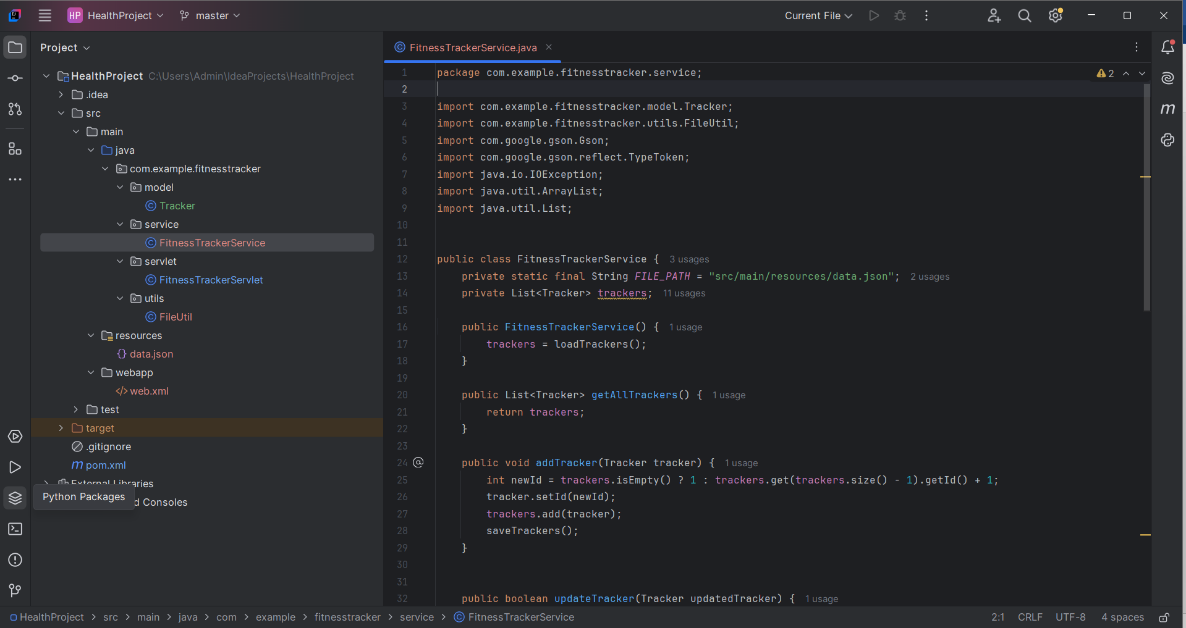


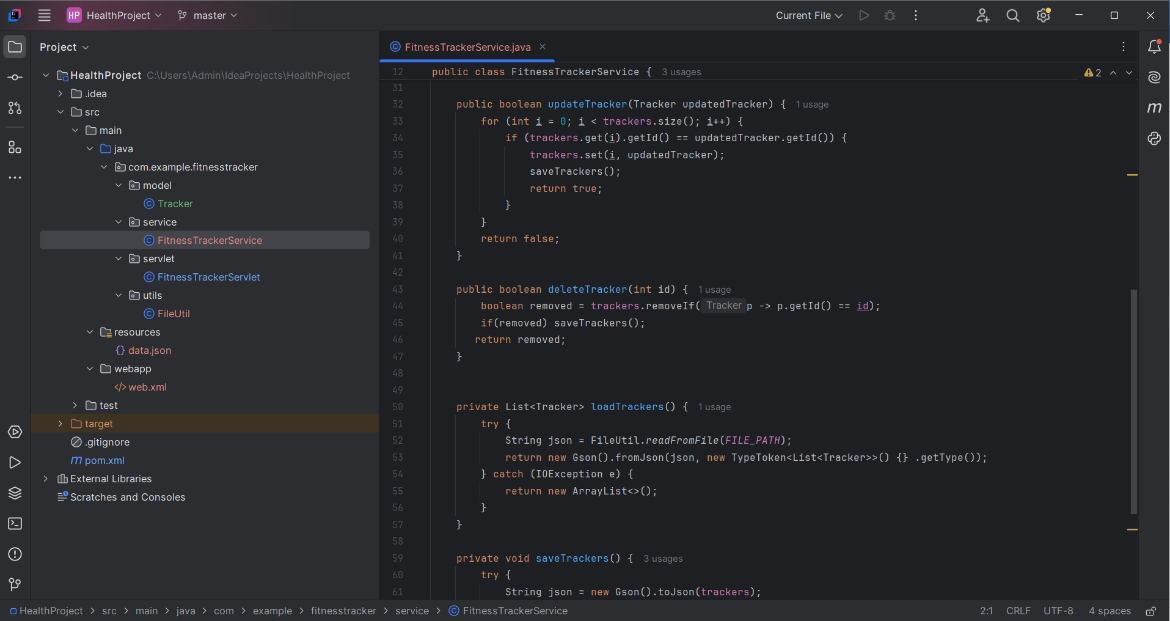
1. Створення сервісу для роботи з даними:

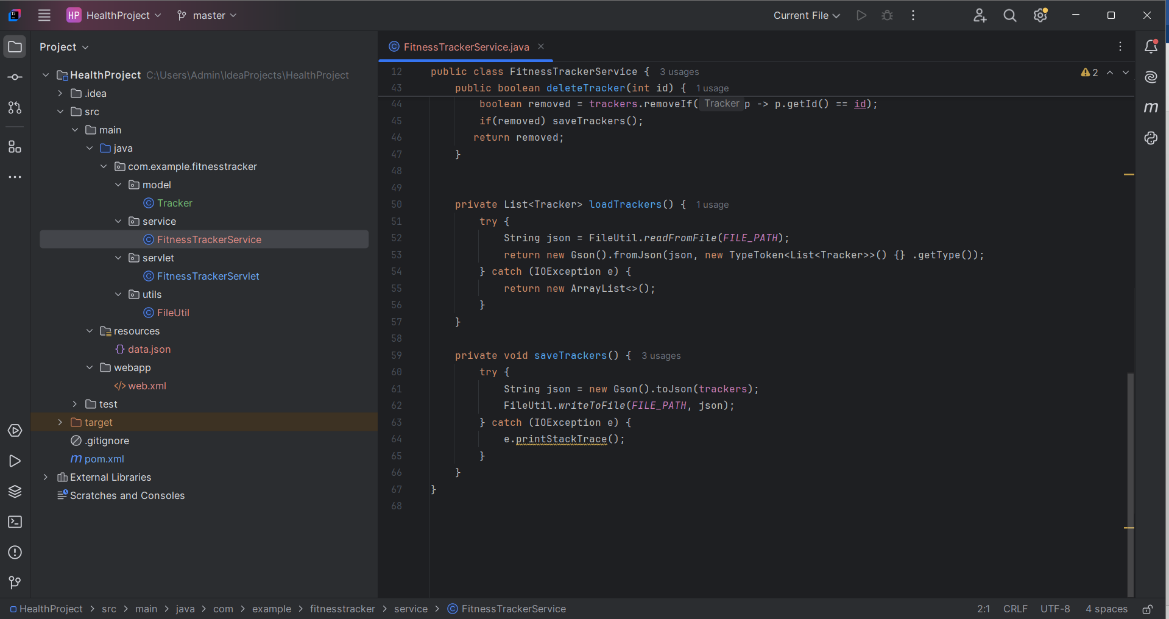
Реалізувати FitnessTrackerService.java, що буде:

Зчитувати та записувати дані у файл (data.json).

Додавати, оновлювати, видаляти та отримувати список об’єктів.







1. Створення CRUD-сервлета:

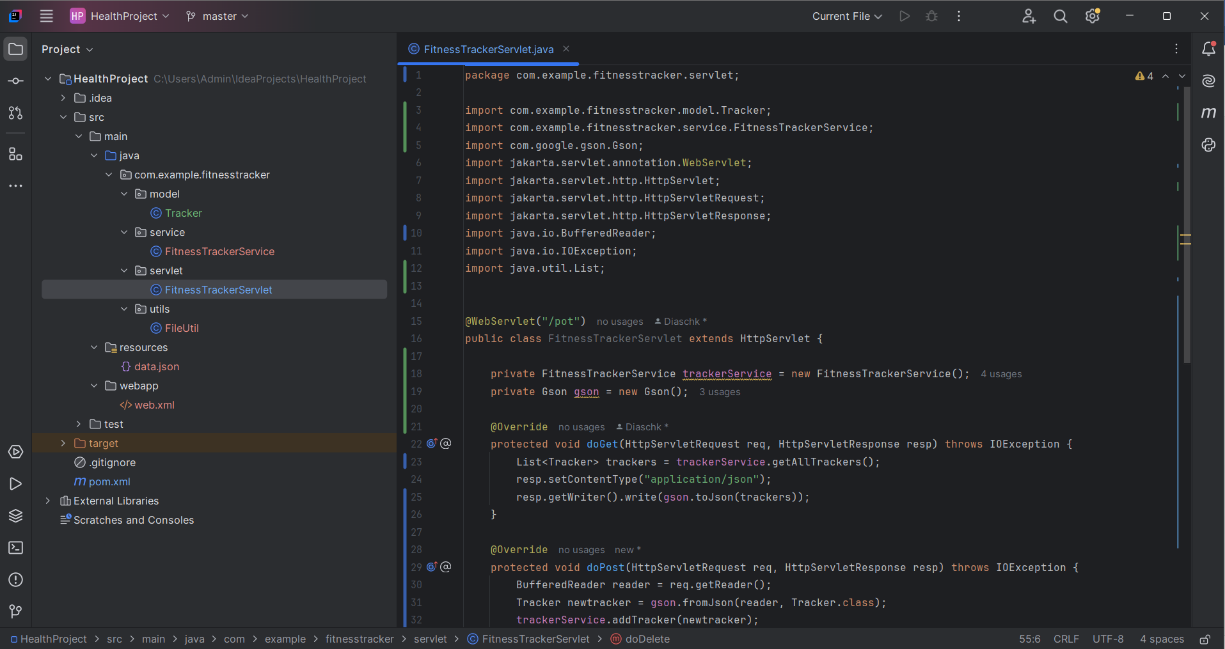
FitnessTrackerServlet.java повинен:

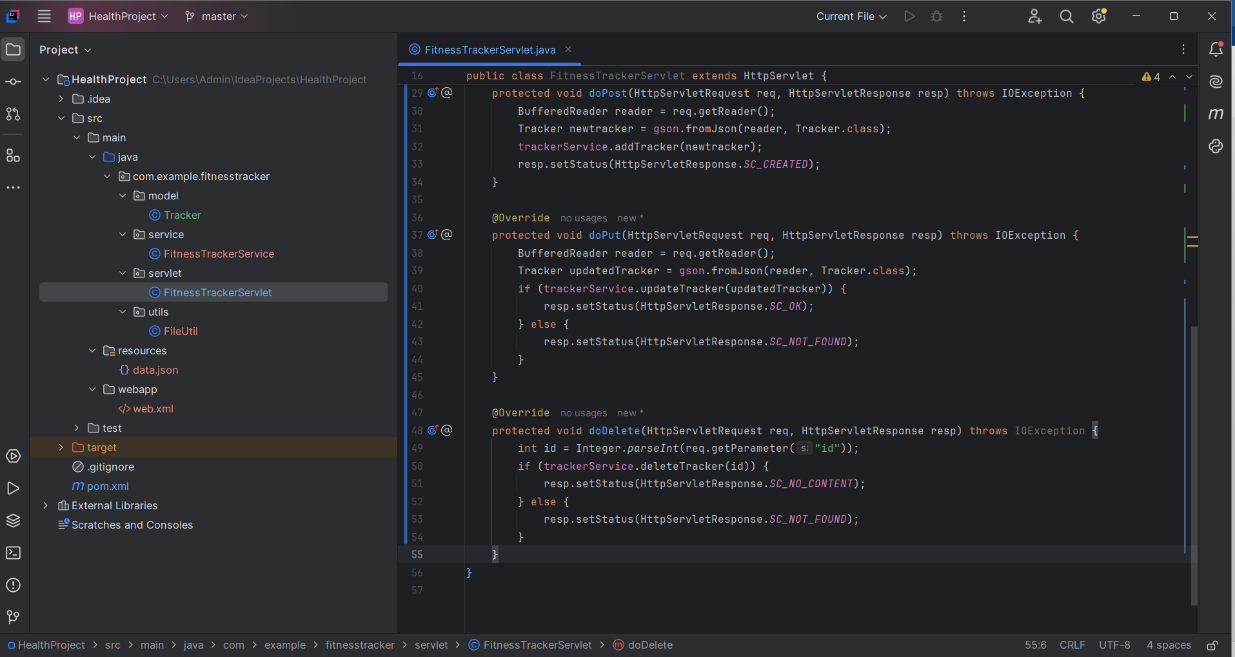
Обробляти GET-запити (отримання списку фітнес-трекерів).

Обробляти POST-запити (створення нового фітнес-трекеру).

Обробляти PUT-запити (оновлення даних фітнес-трекера).

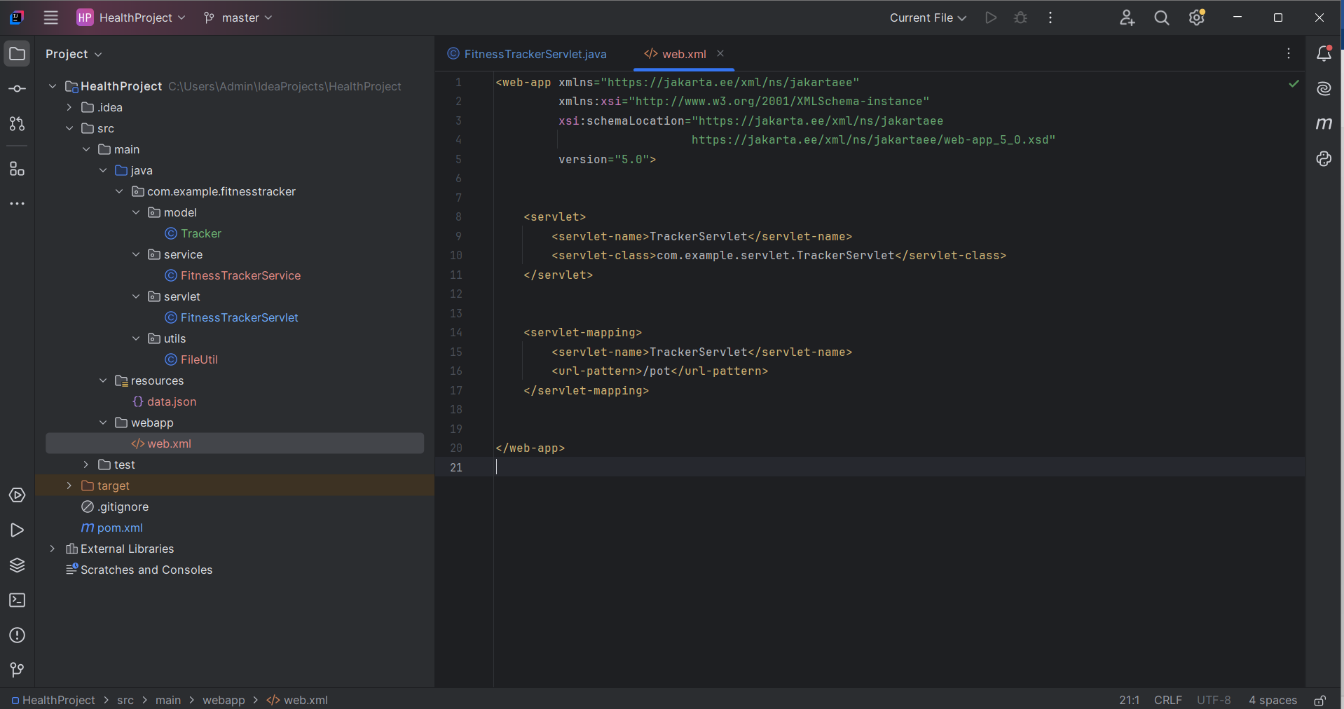
Обробляти DELETE-запити (видалення фітнес-трекера за id).





1. Налаштування web.xml або анотацій:

Додати анотацію @WebServlet("/pot") або описати у web.xml.



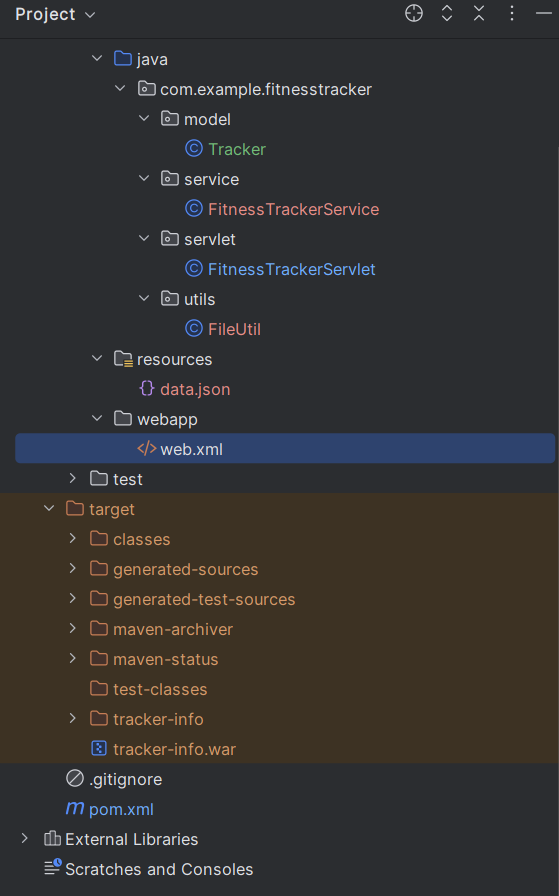
1. Перекомпілювати проєкт та розгорнути його на Tomcat

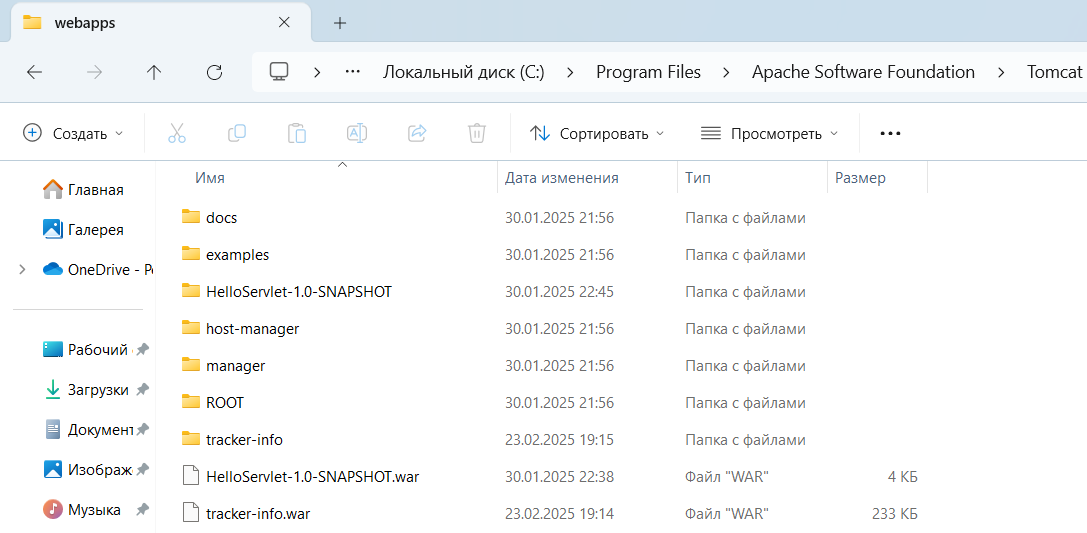
Виконати mvn clean package.

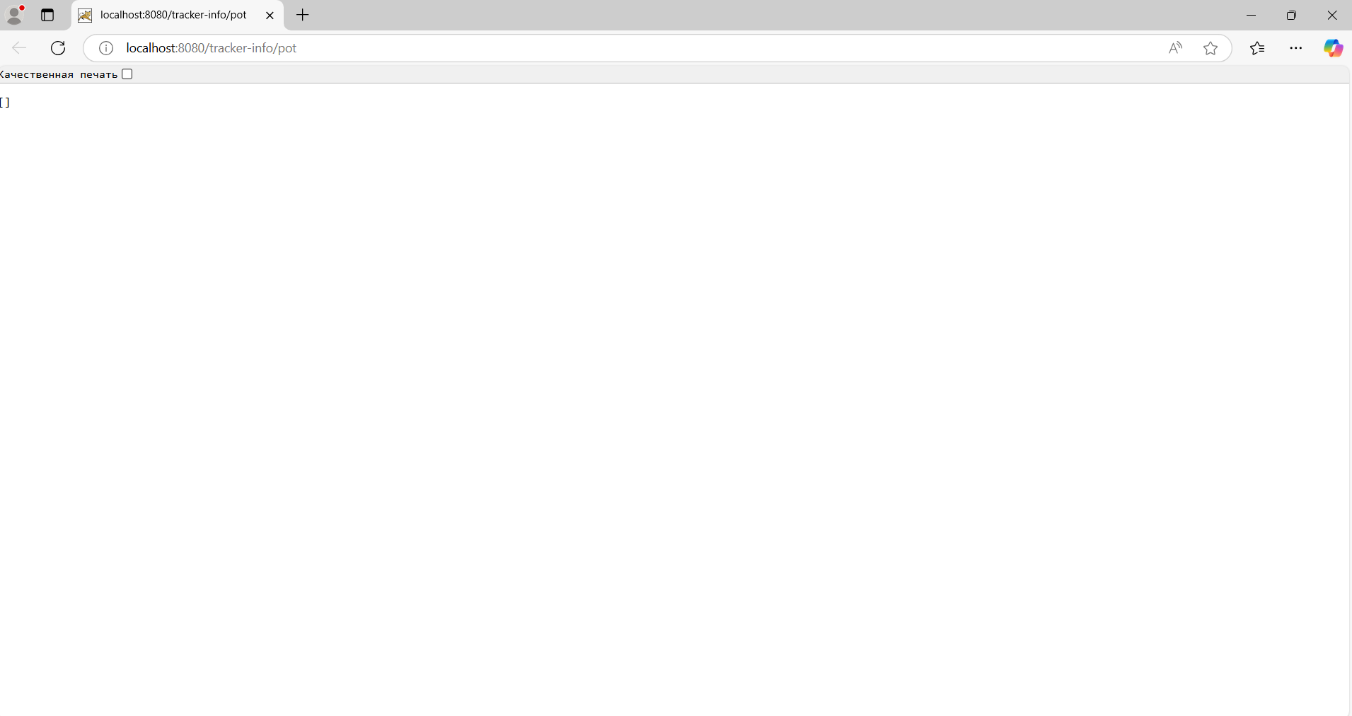
Переконатися, що .war файл з'явився у target/.

Скопіювати .war до папки webapps Tomcat та перезапустити сервер.

Перевірити у браузері <http://localhost:8080/tracker-info/pot>.







1. **Завершення розробки Front-end**
2. Налаштування середовища розробки

Відкрити VS Code та проєкт Vue.js.

Виконати npm install, щоб встановити залежності.

1. Розширення UI для підтримки CRUD-операцій

Додати кнопки "Додати", "Редагувати", "Видалити".

Додати форми введення для введення даних про об’єкти.

1. Взаємодія з бекендом

Використати fetch() для запитів до /pot.

Переконатися, що Vue.js отримує дані та відображає їх у списку.

1. Налаштування vite.config.js для проксі

Додати:  
 server: {

  proxy: {

    '/pot': {

      target: 'http://localhost:8080/назва\_проєкту',

      changeOrigin: true,

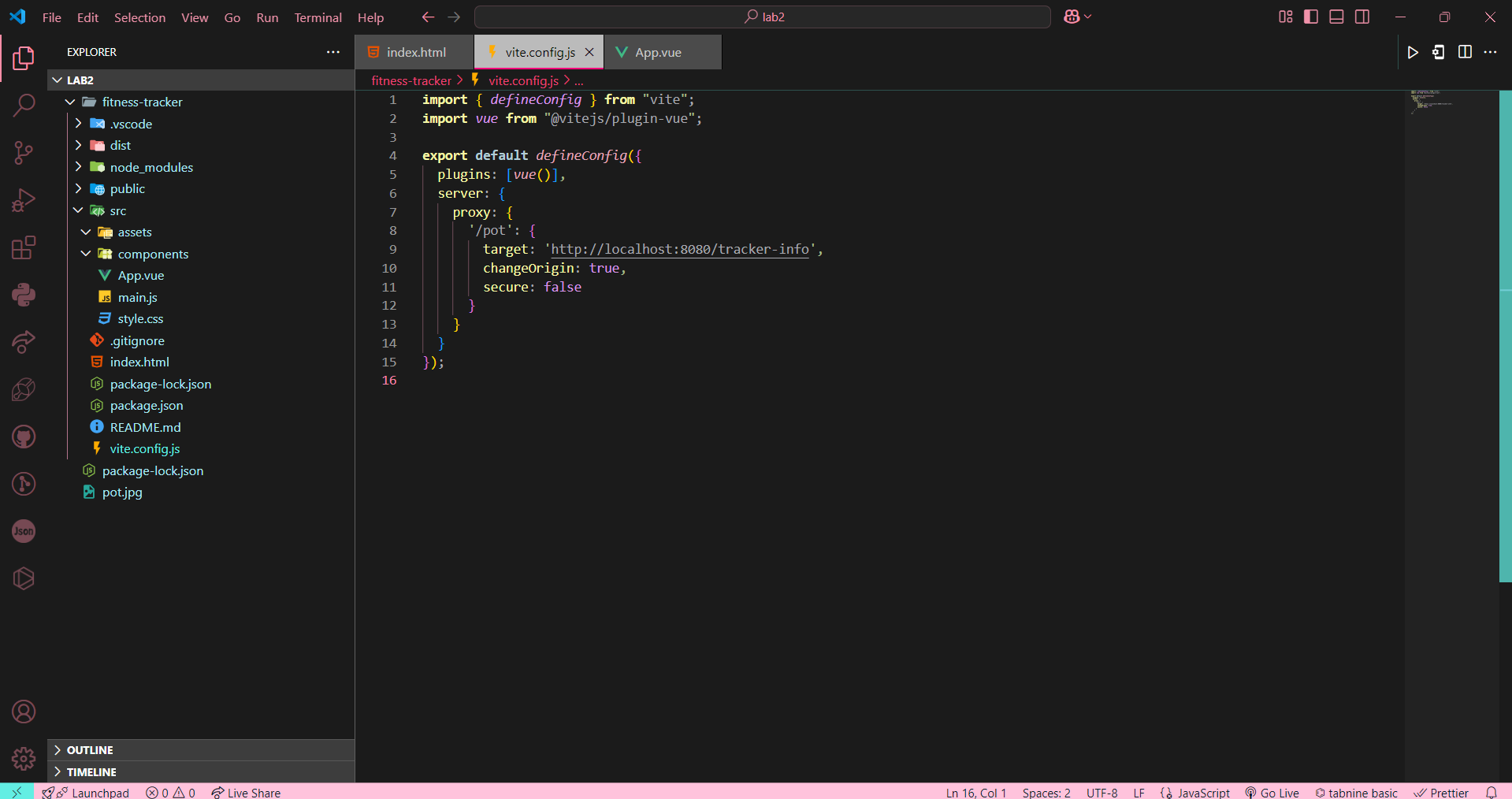
      secure: false

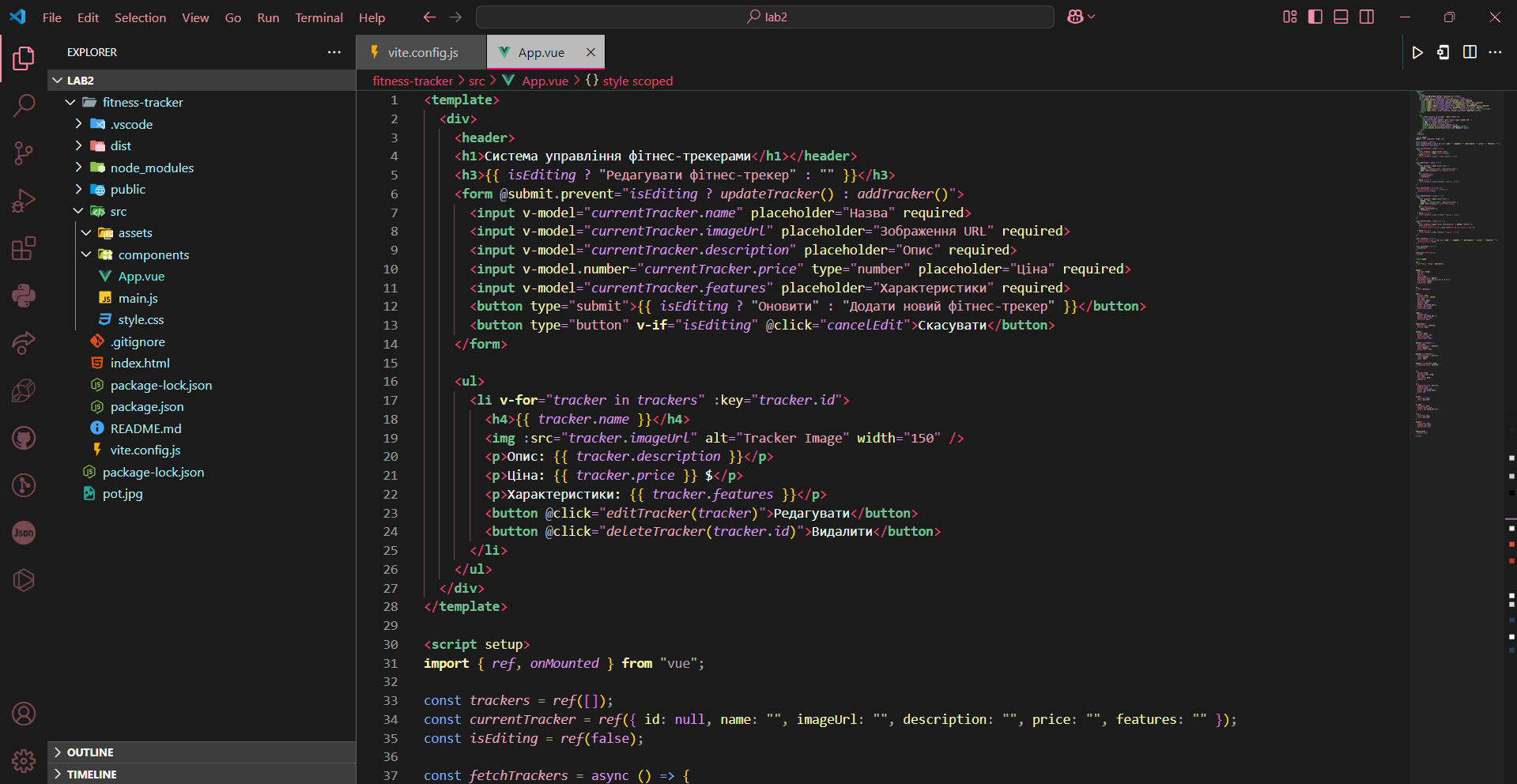
    }

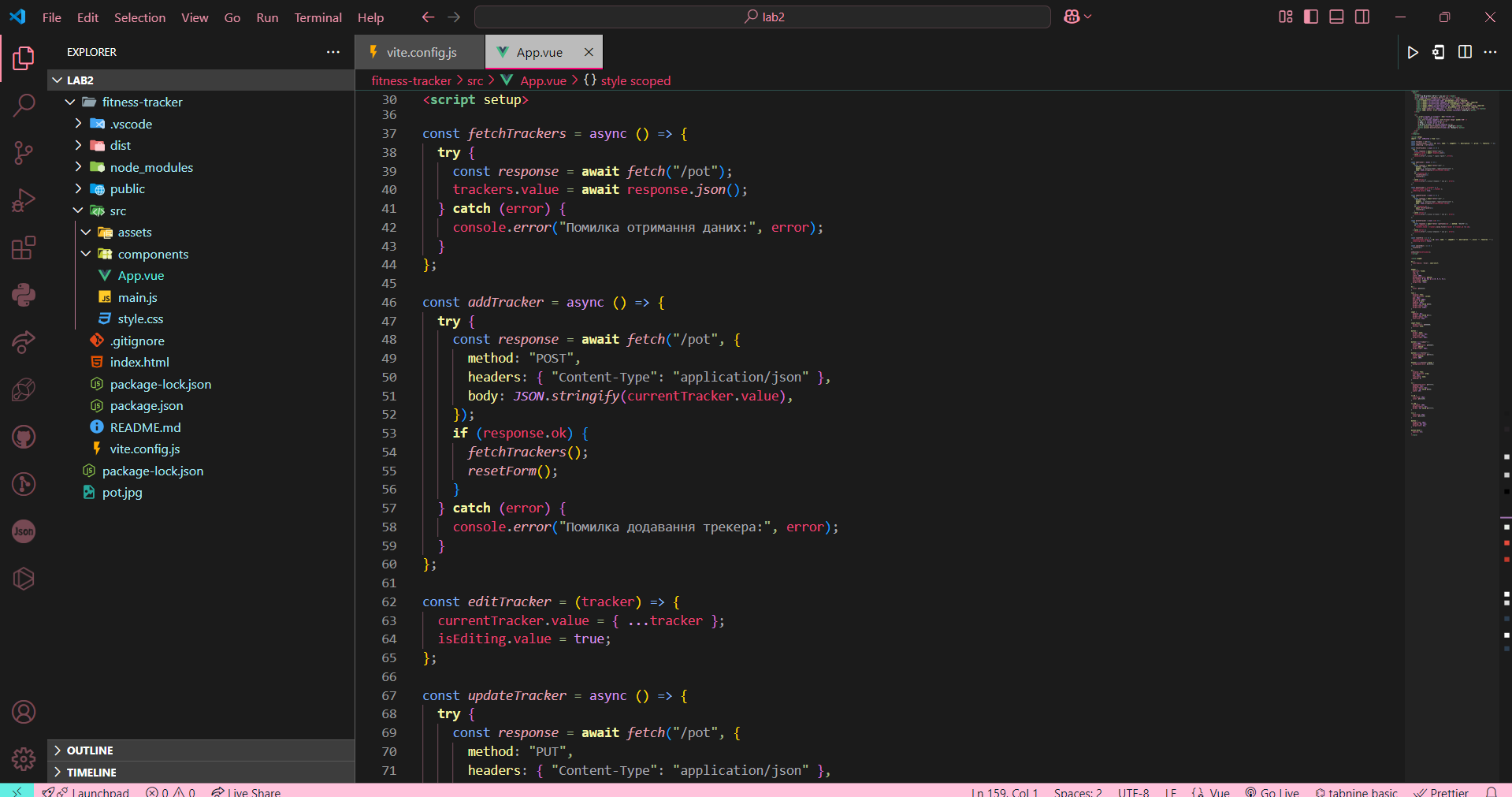
  }

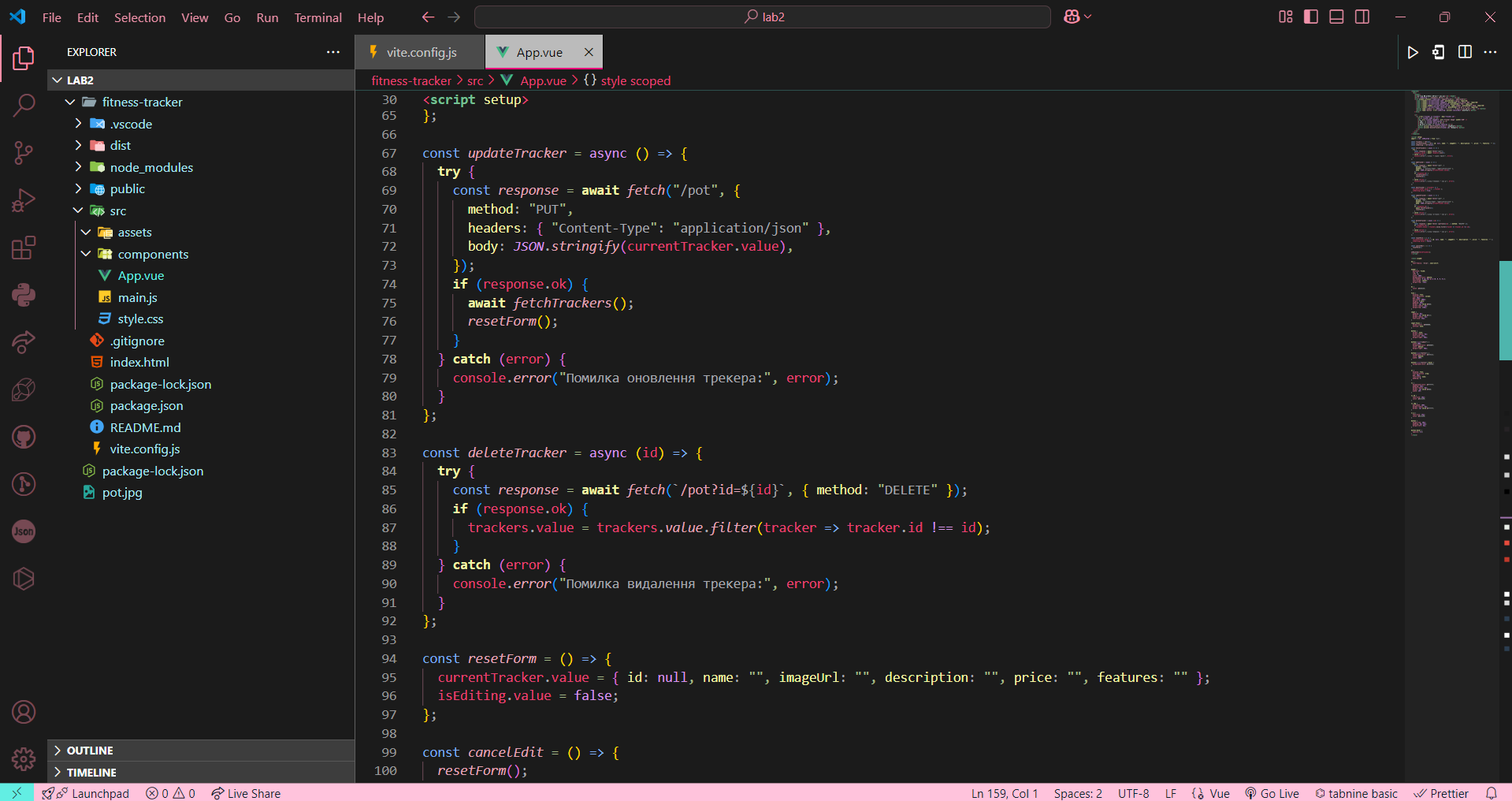
}

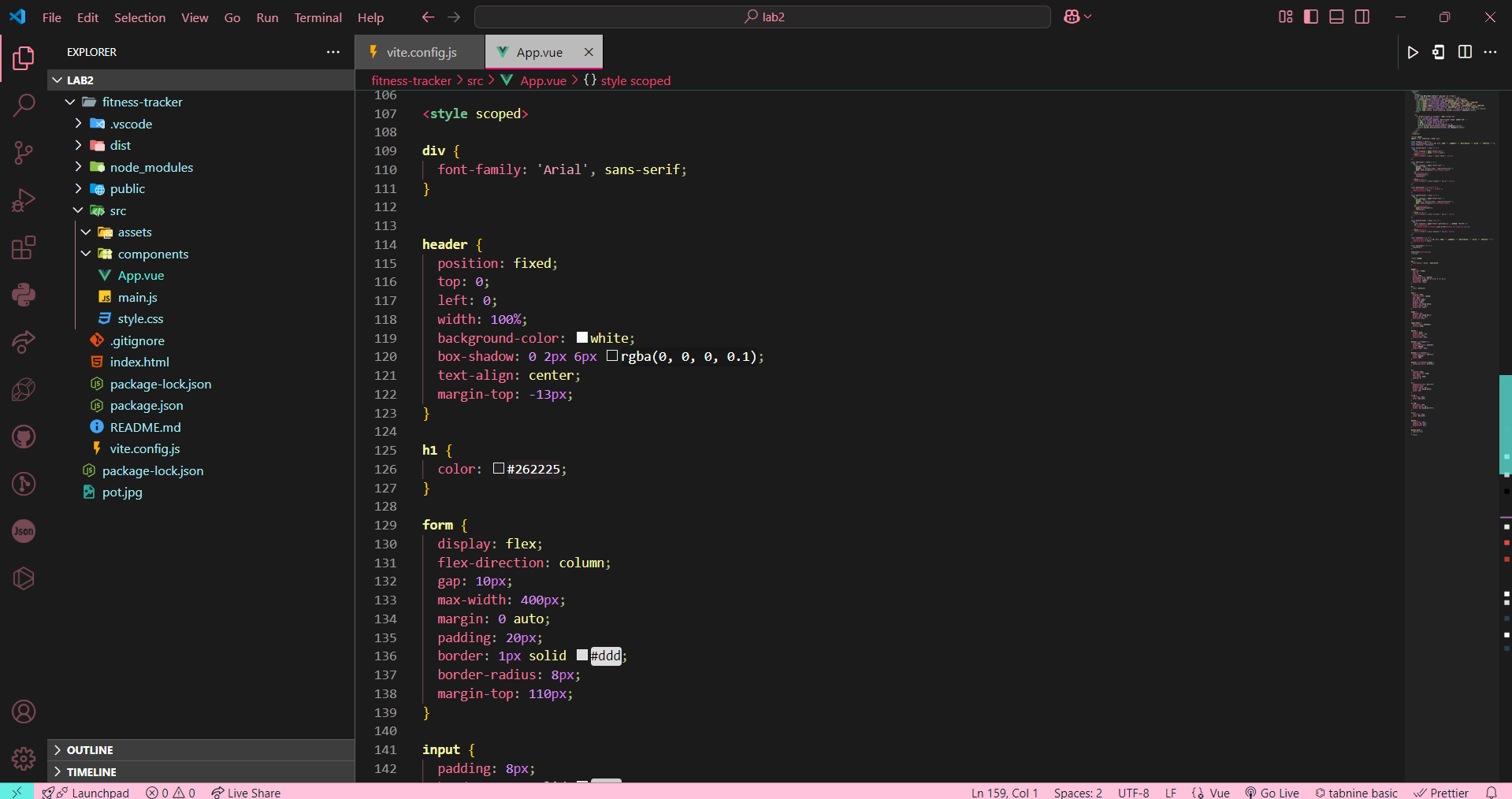
5. Перезапустити Vue (npm run dev).

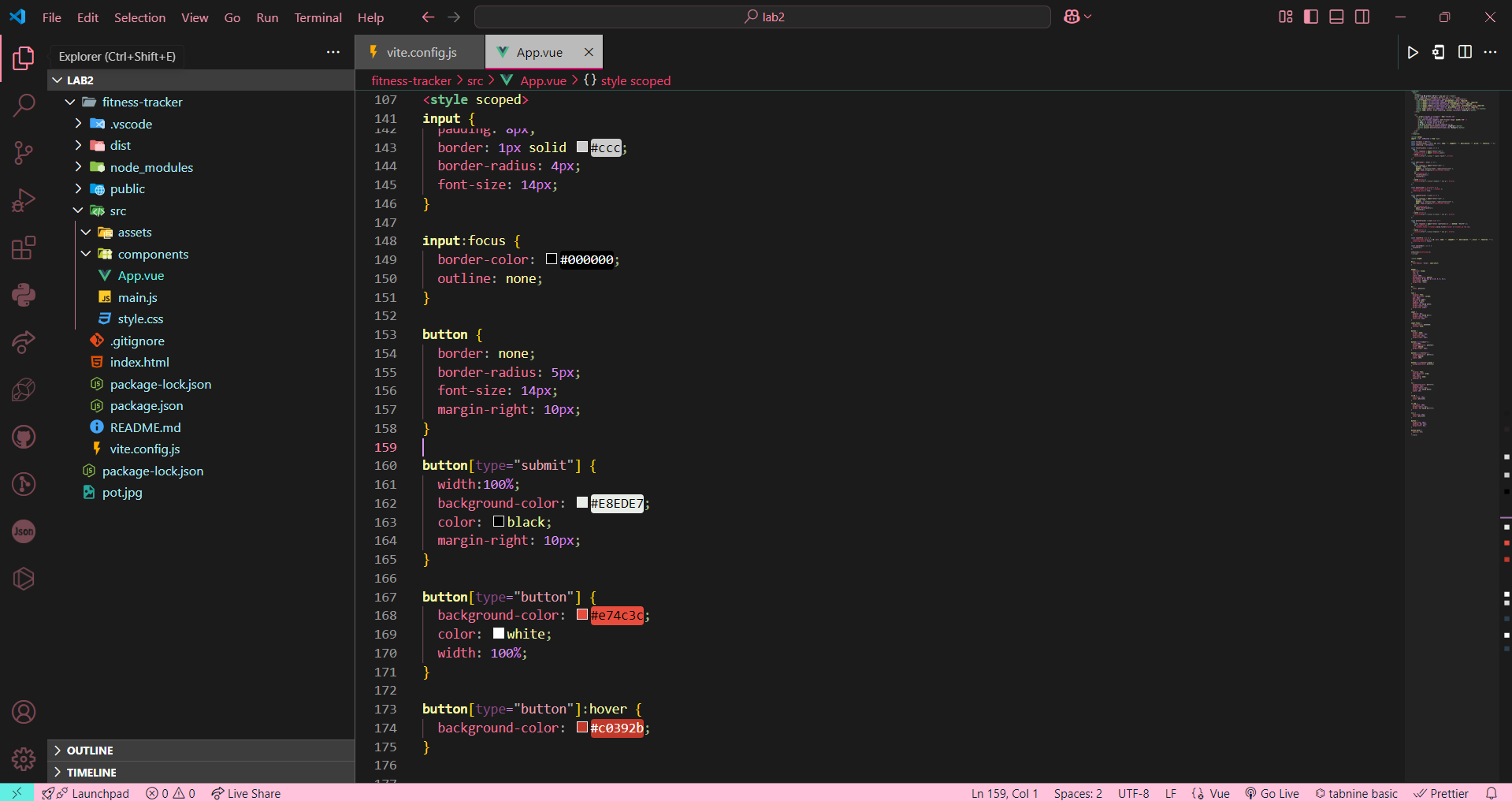


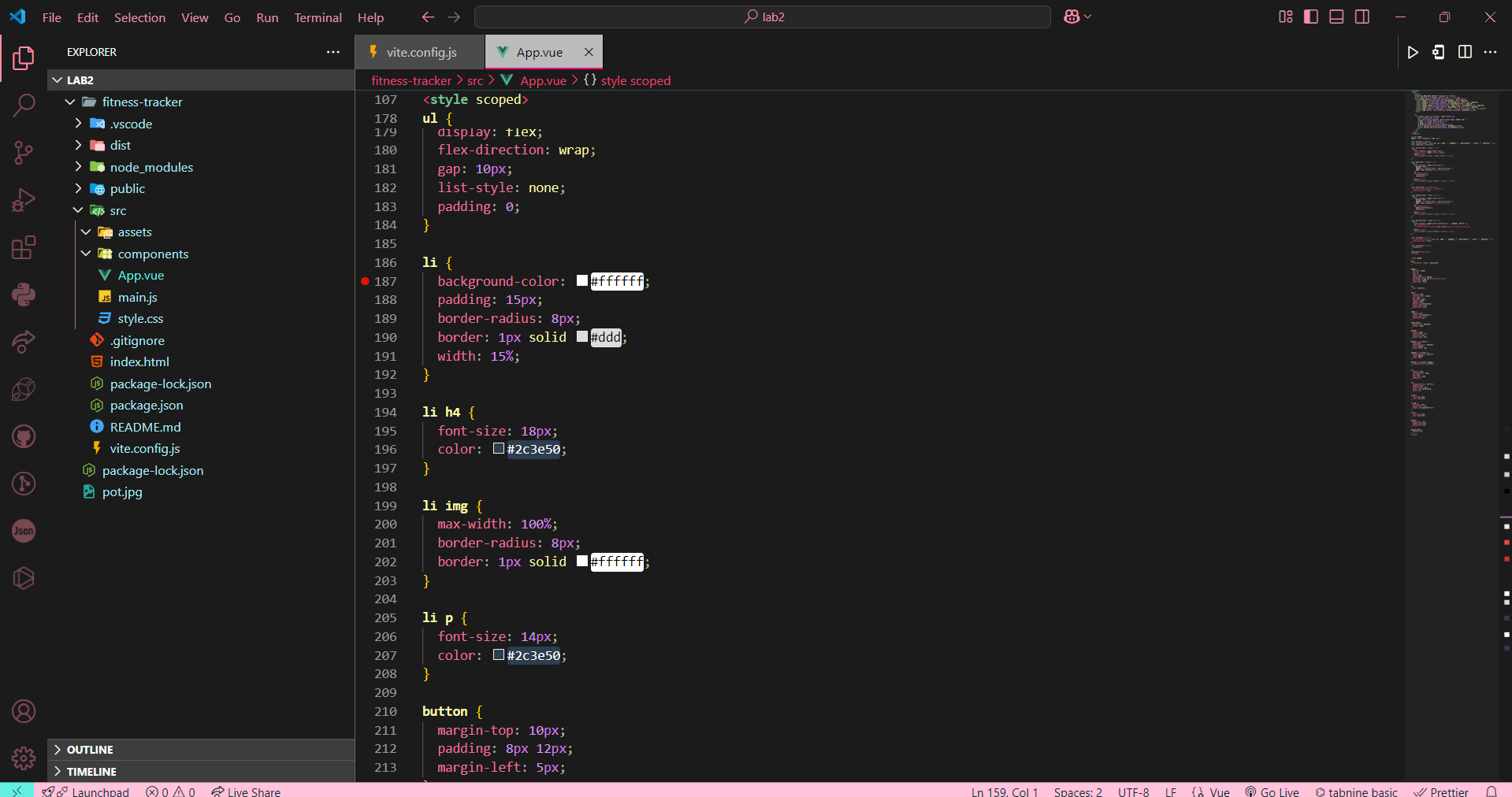


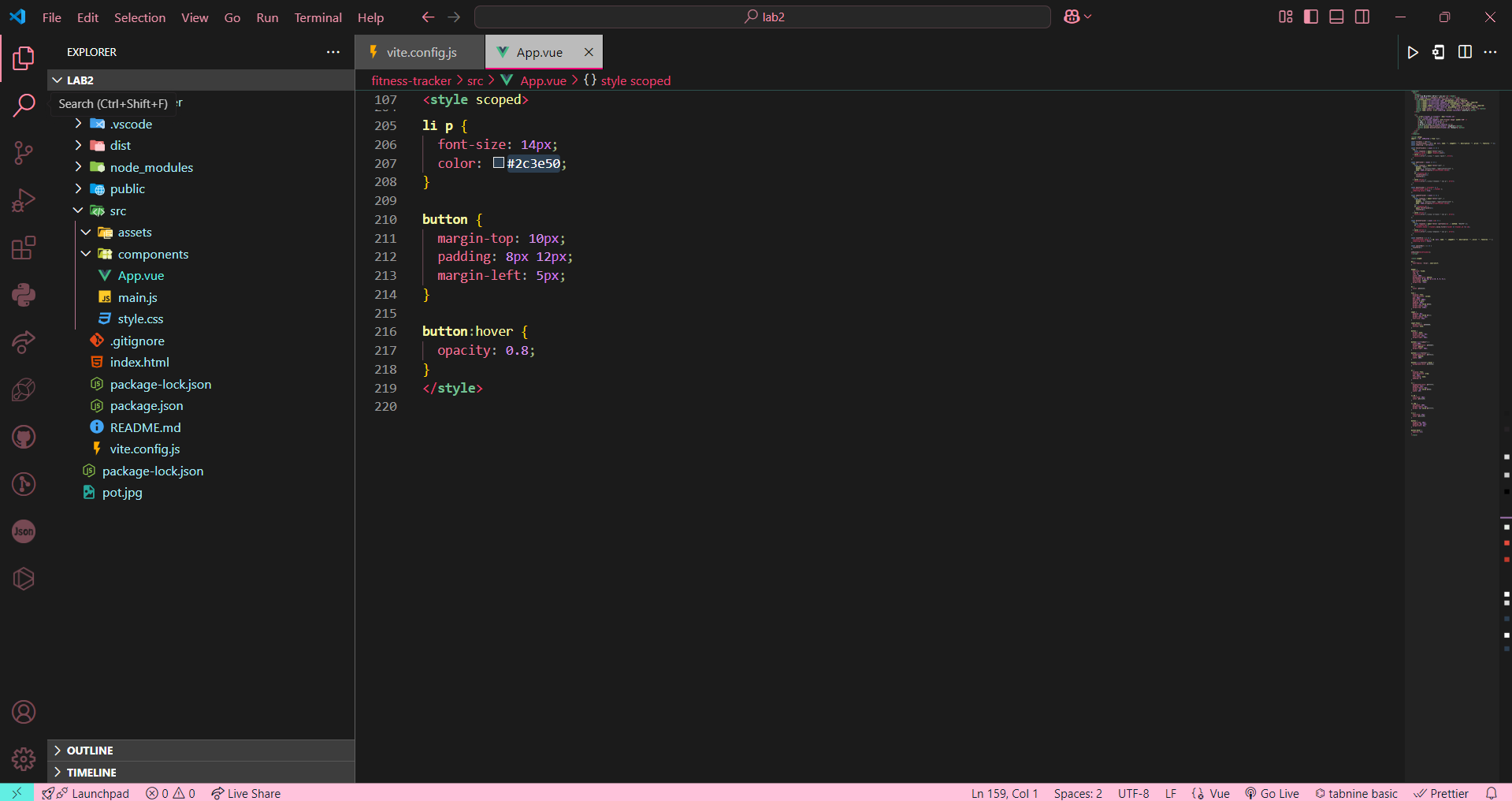


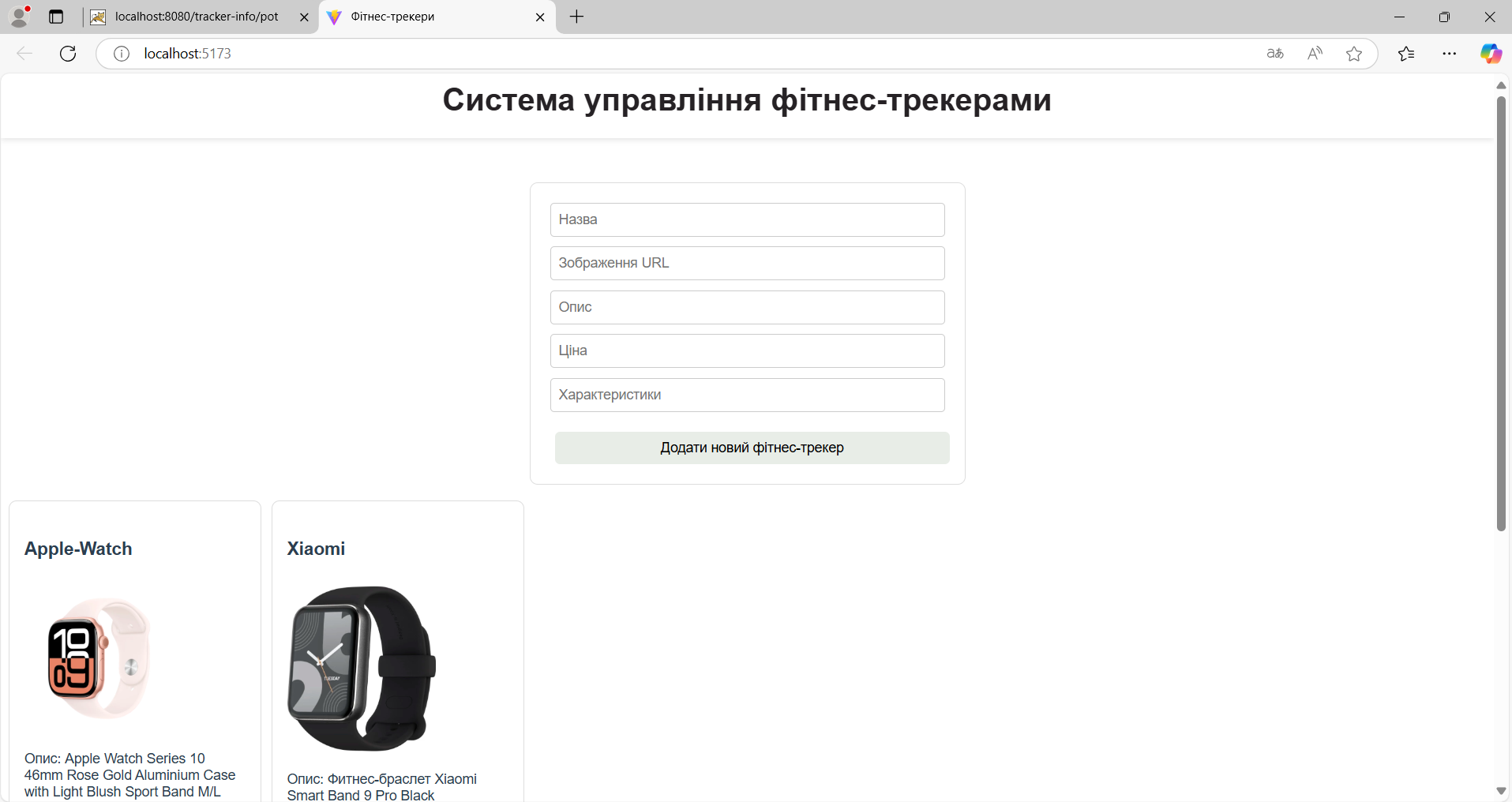


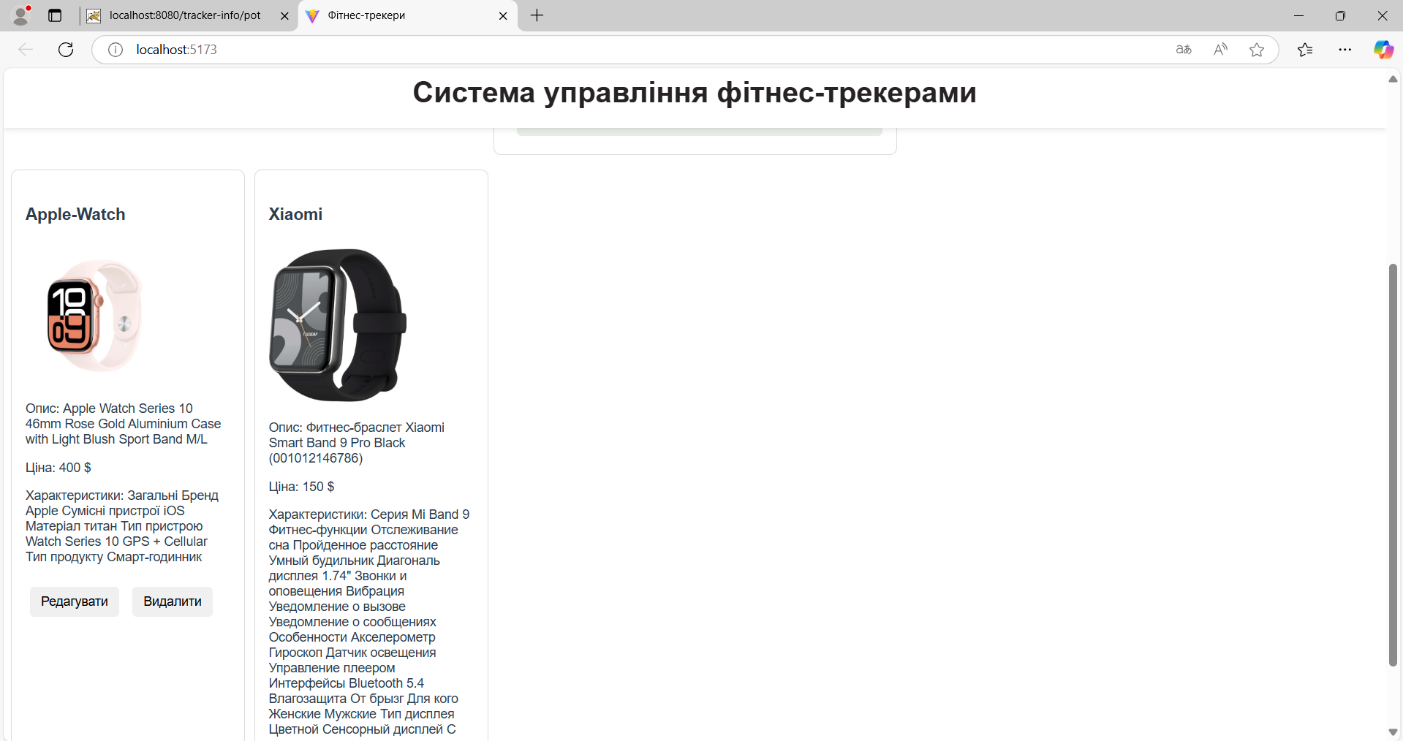


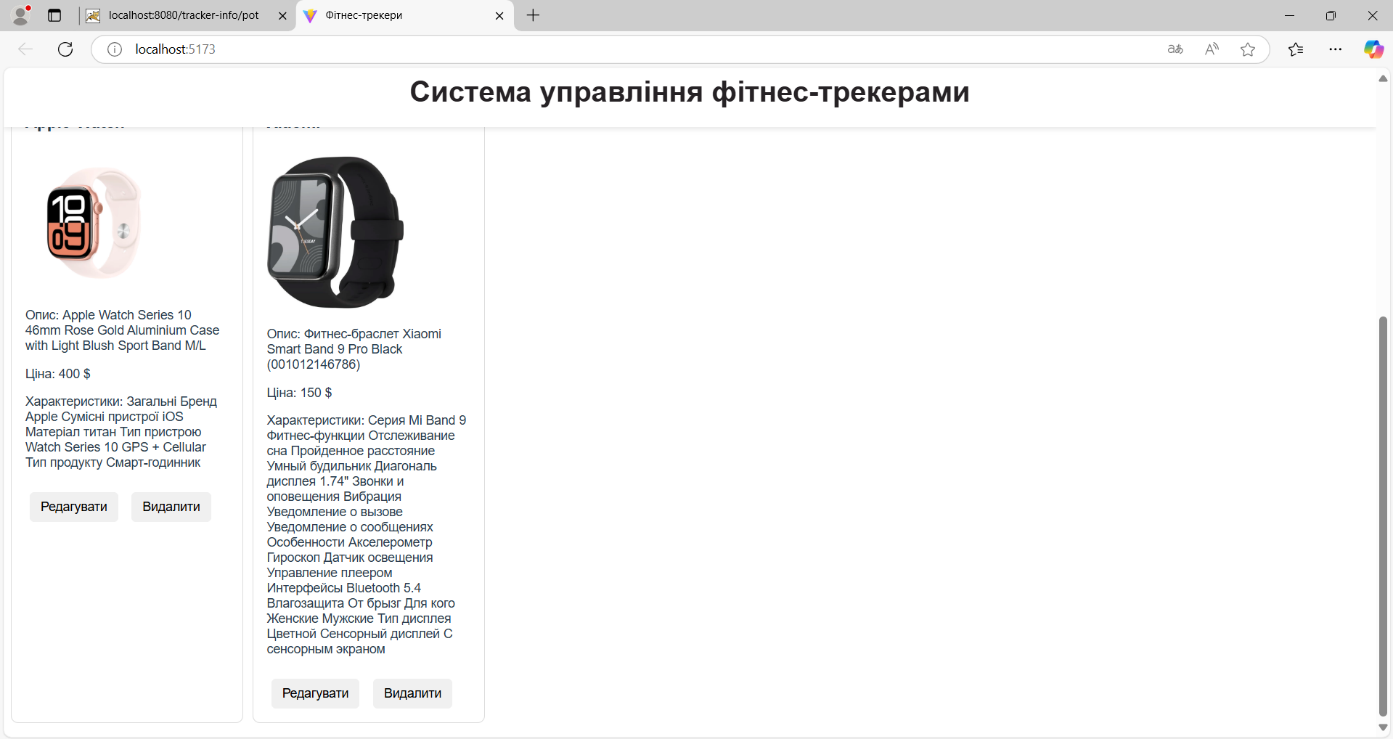


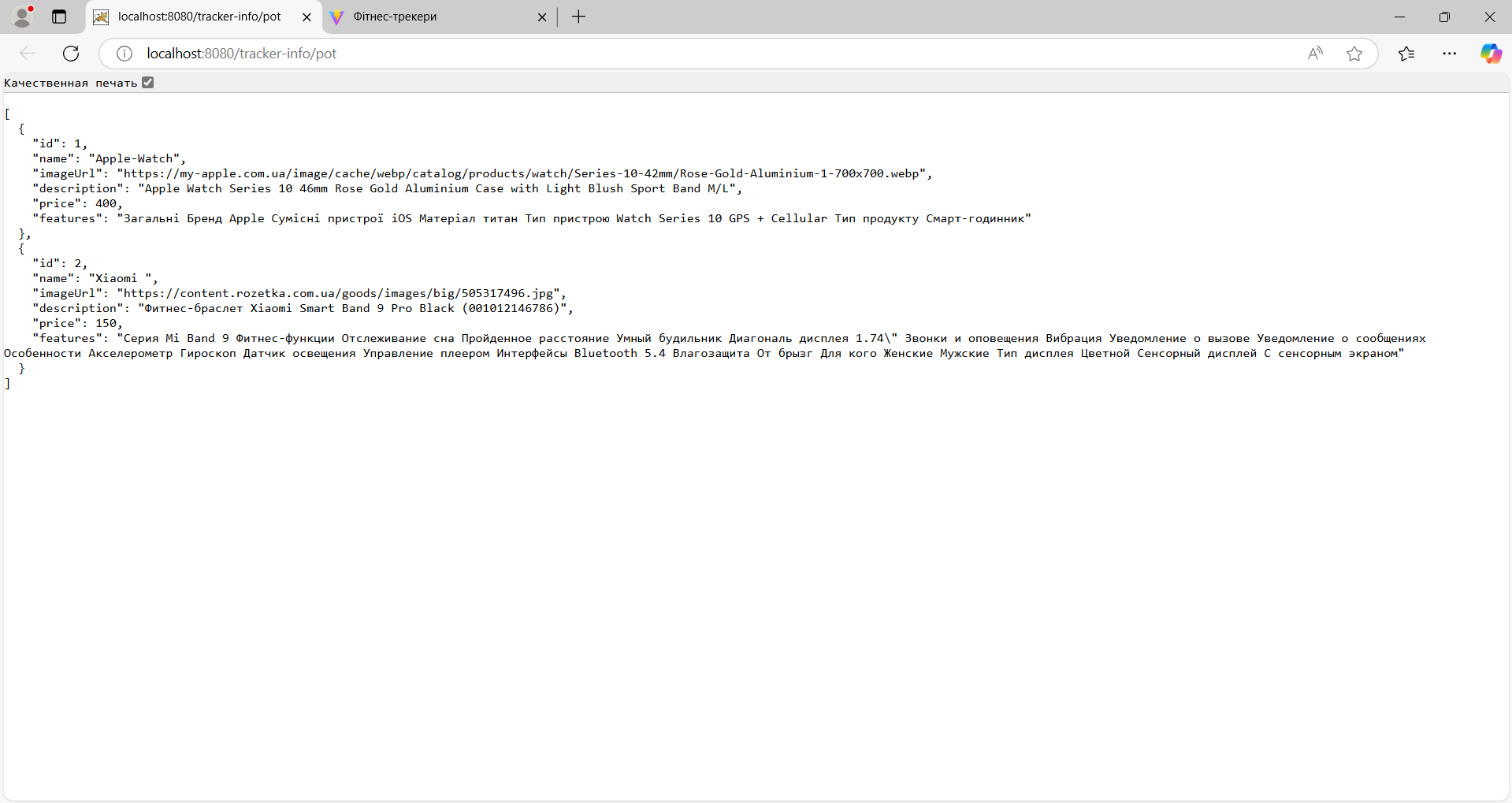
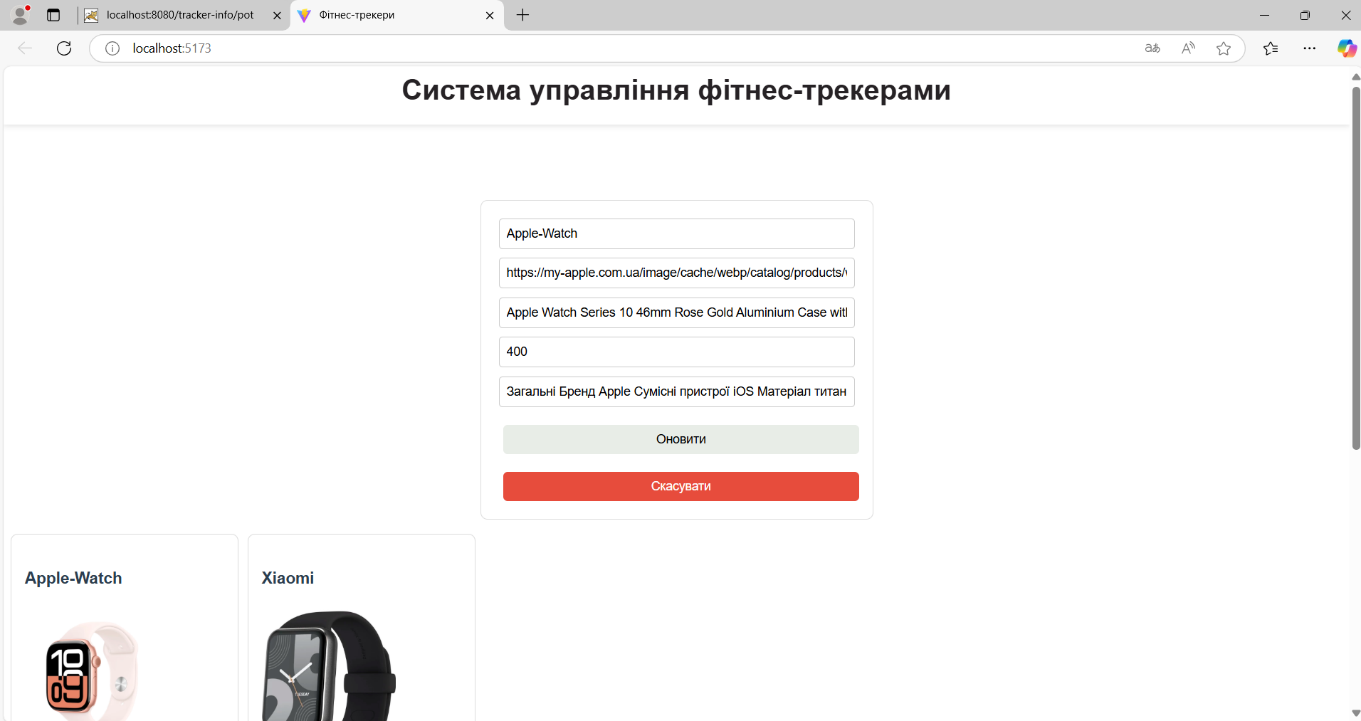












1. **Інтеграція Front-end і Back-end**
2. Перевірка всіх CRUD-операцій

Додати об’єкт через форму (POST-запит).

Редагувати об’єкт та перевірити оновлення (PUT-запит).

Видалити об’єкт та переконатися, що запис видаляється у файлі (DELETE-запит).

Використати DevTools (вкладка Network) для перевірки запитів.

1. Тестування та виправлення помилок

Переконатися, що всі операції працюють без помилок.

Перевірити логи Tomcat на наявність помилок (logs/catalina.out).

**Запитання для самоперевірки:**

1. Як реалізовані CRUD-операції у бекенд-частині проєкту?

**CRUD-операції у бекенді** реалізовані через сервлет FitnessTrackerServlet, який обробляє HTTP-запити:

* 1. GET — отримання списку об’єктів.
  2. POST — додавання нового об’єкта (зчитування JSON з тіла запиту, збереження у файл).
  3. PUT — оновлення існуючого об’єкта (пошук за id, заміна даних).
  4. DELETE — видалення об’єкта за id (фільтрація списку).  
     Сервіс FitnessTrackerService відповідає за логіку роботи з файлом data.json за допомогою бібліотеки Jackson.

1. Як відбувається збереження даних у файл?

**Збереження даних у файл** відбувається через серіалізацію колекції об’єктів у JSON. При кожній зміні даних (додавання, оновлення, видалення) сервіс перезаписує файл.

1. Як працює оновлення даних у фронтенді без перезавантаження сторінки?

**Оновлення без перезавантаження** реалізоване через асинхронні запити fetch() у Vue.js. Після успішного запиту фронтенд оновлює локальний стан (масив об’єктів), що автоматично відображається у інтерфейсі через реактивність Vue.

1. Як обробляються помилки при взаємодії між фронтендом і бекендом?
   1. На бекенді: сервлети повертають HTTP-коди (наприклад, 404 якщо об’єкт не знайдено).
   2. На фронтенді: try/catch у методах Vue.js з виведенням повідомлень користувачу.
   3. Використання DevTools для аналізу відповідей сервера.
2. Як забезпечити коректну роботу DELETE-запиту у сервлетах?

Сервлет отримує id з параметрів URL (наприклад, /pot?id=123), після чого сервіс видаляє об’єкт зі списку та оновлює файл.

1. Як правильно працювати з Git-гілками під час розробки?

Необхідно створювати окремі гілки для нових функцій, комітити зі зрозумілими назвами та мерджити гілки через pull request з рев’ю коду.

**Висновок**

Під час лабораторної роботи я розширила функціональність інформаційної системи, реалізувавши повний цикл CRUD-операцій. На бекенді створила сервіс для роботи з JSON-файлом та сервлет для обробки HTTP-запитів. На фронтенді інтегрувала Vue.js з асинхронними запитами, що забезпечило динамічне оновлення інтерфейсу без перезавантаження сторінки. Отриманий результат підтверджує ефективність обраного підходу та технологій, що можуть бути розширені для подальшого розвитку системи.