

ЗВІТ
до лабораторної роботи № 2
на тему:
" Пілотний проєкт інформаційної системи."

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

Мета роботи: Створення пілоотної версії інформаційної системи з отриманням даних через JSON-сервлет та відображенням їх на фронт-енд з використанням Vue.js.

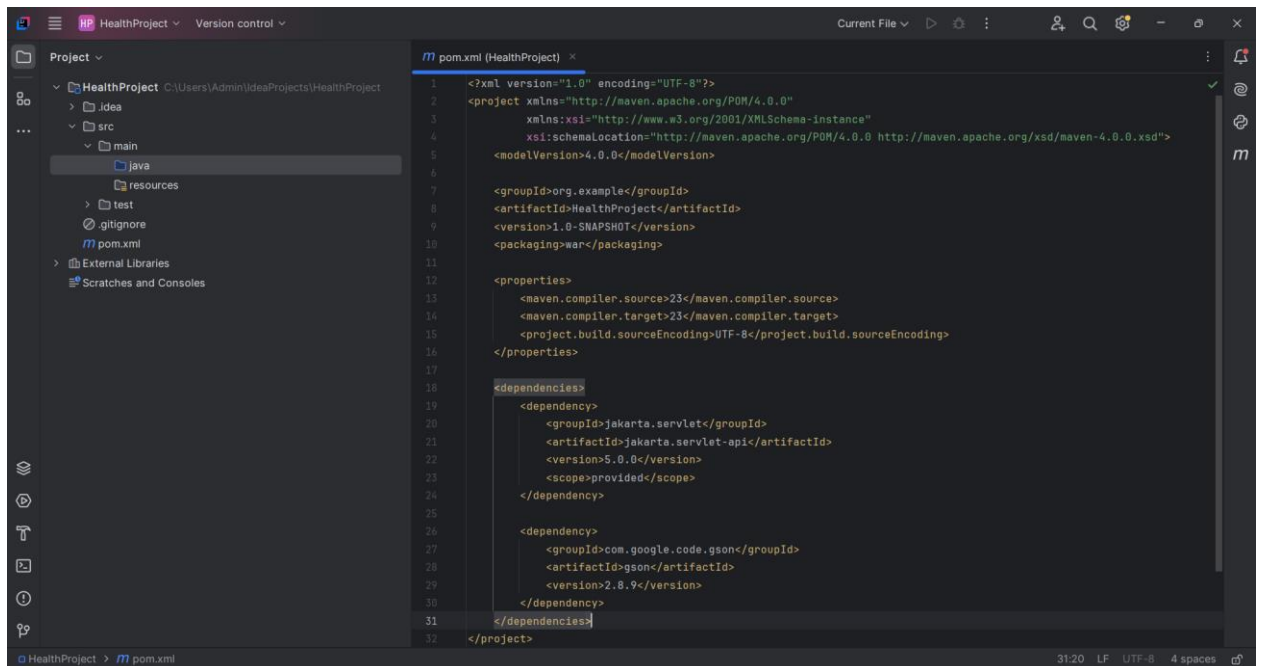
Здоров'я: фітнес-трекери

Хід роботи:

Створення проєкту:

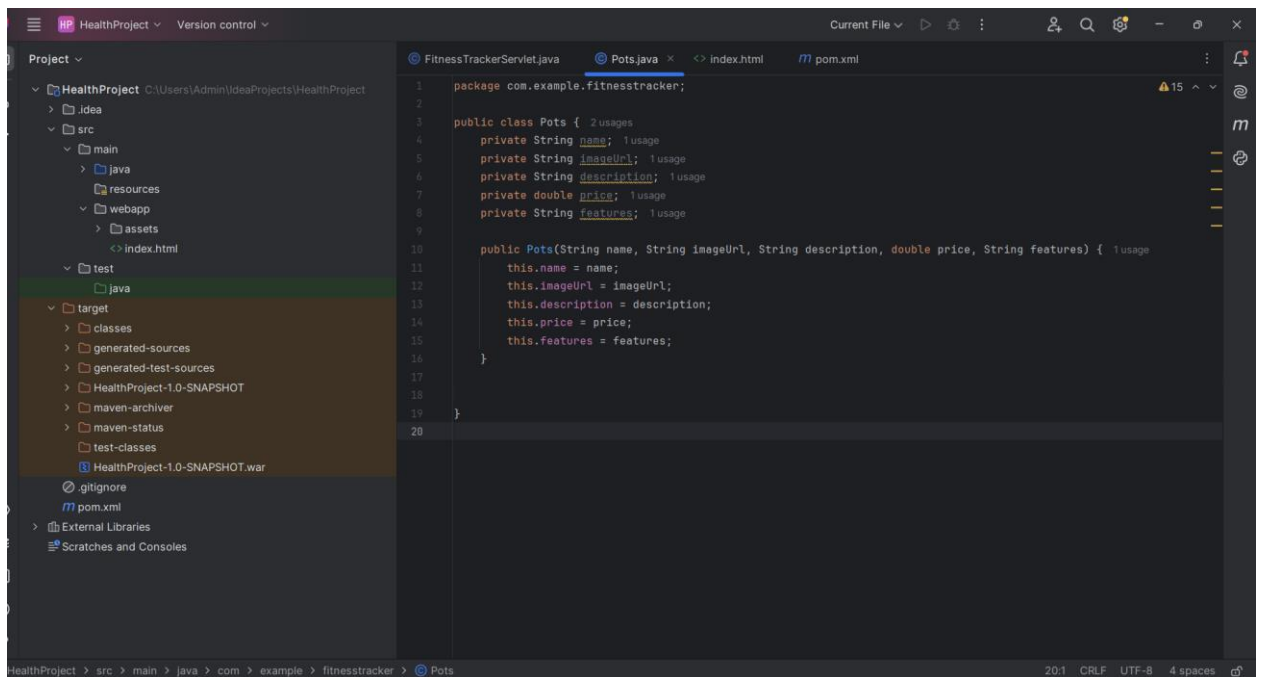
1. Відкрити IntelliJ IDEA Community Edition.
2. Створити новий Maven-проєкт із підтримкою Java Servlet API (додати в файл `pom.xml` ті ж залежності що і в попередній лабораторній роботі + `war`).
3. Додати залежність для Gson у `pom.xml`:

```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.8.9</version>
</dependency>
```

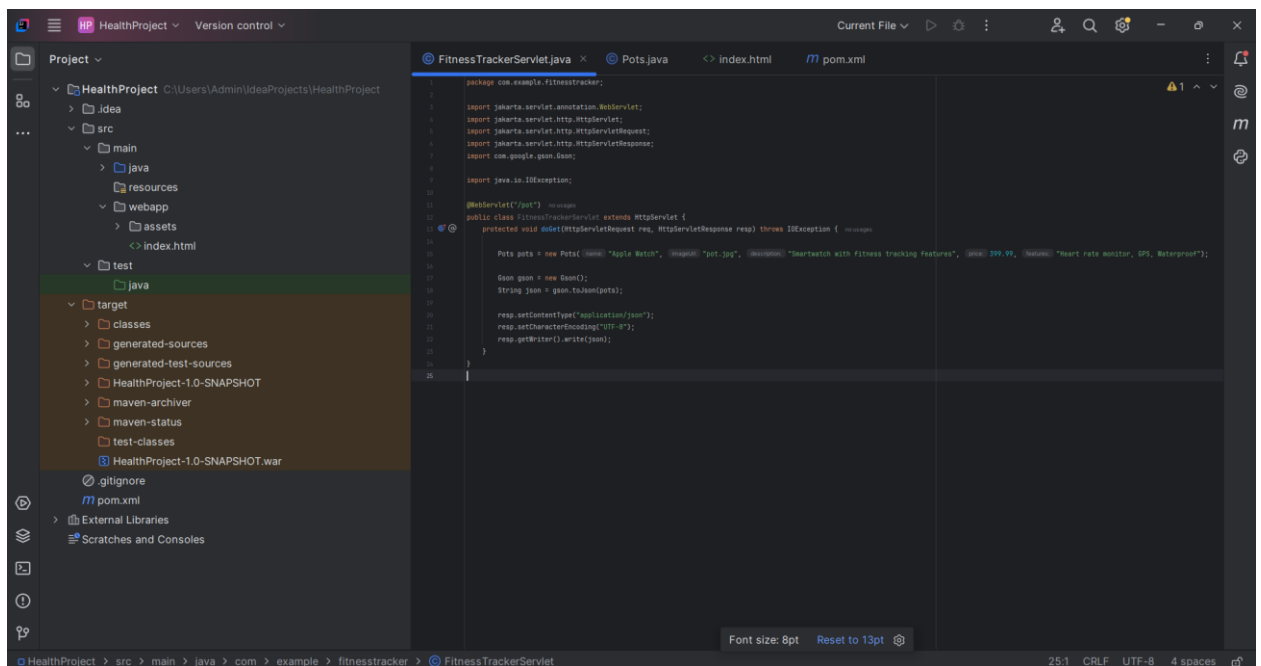


Розробка Back-end:

1. Створити клас для збереження даних про об'єкти інформаційної системи відповідно до обраного варіанту (наприклад, назва, зображення, опис, інші характеристики).



2. Створити сервлет, що обробляє GET-запити та повертає JSON-відповідь.

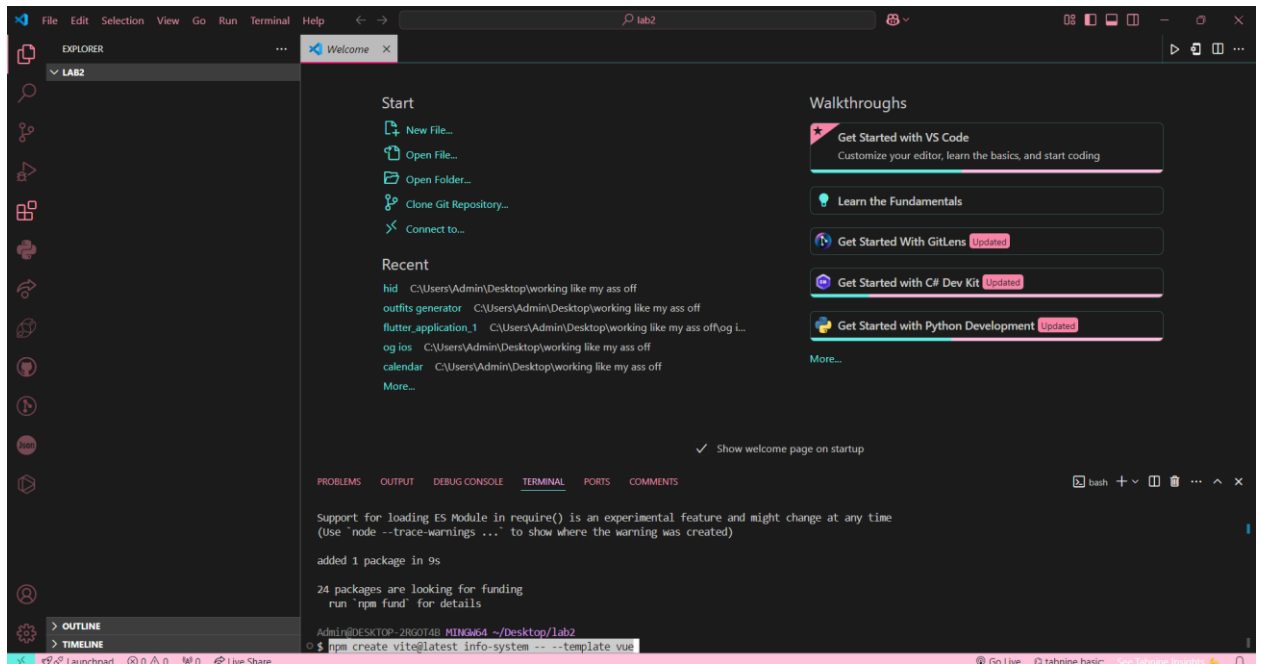


Розробка Front-end:

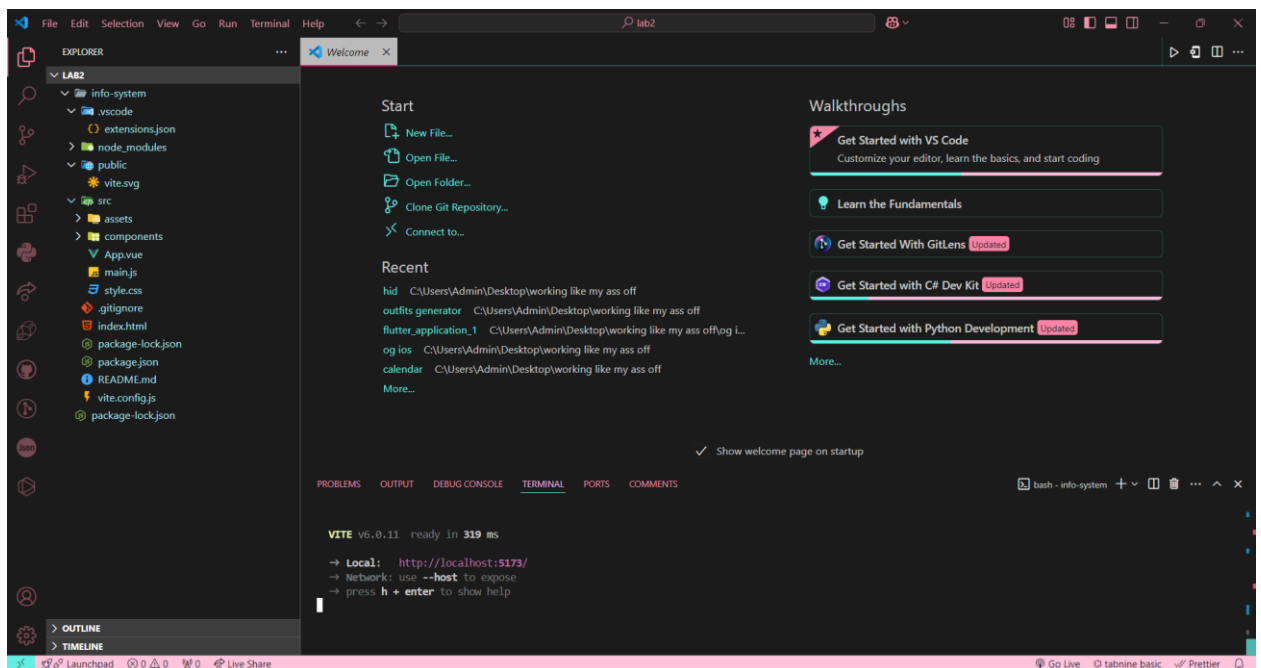
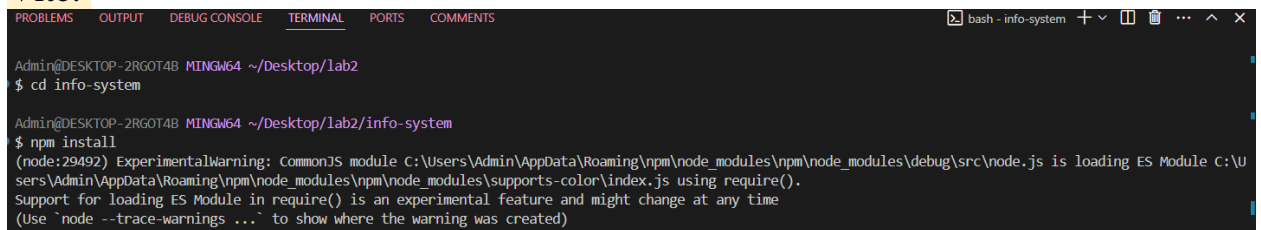
1. Встановити Node.js та npm (якщо ще не встановлено).
2. Створити проєкт за допомогою Vite та Vue.js. Виконайте команду для створення нового проєкту за допомогою Vite:

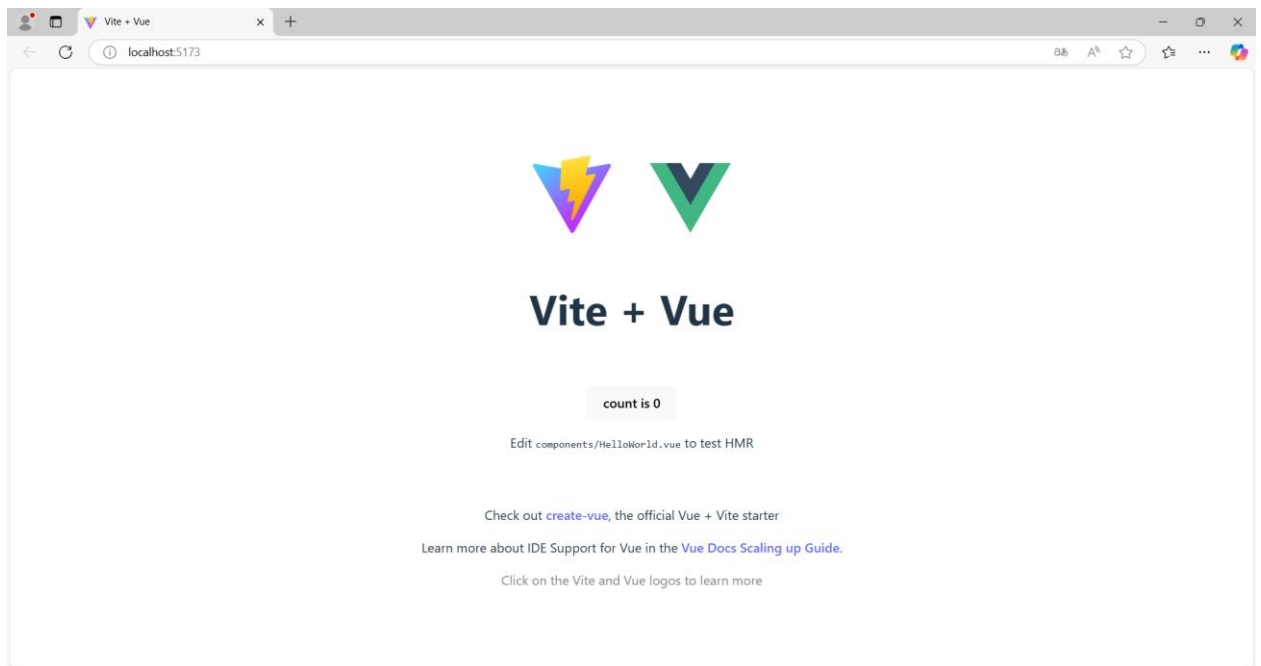
`npm create vite@latest info-system -- --template vue`

Тут *info-system* — це назва вашого проєкту.

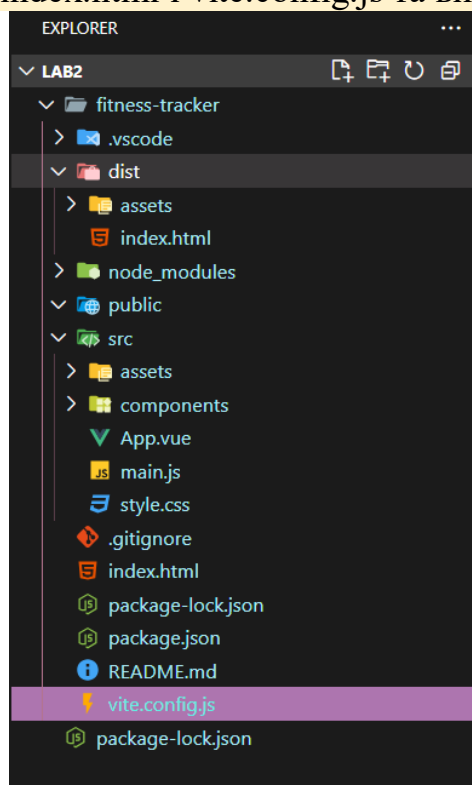


3. Виберіть фреймворк Vue, потім JavaScript.
4. Запустіть команди які відображаються в терміналі. Таким чином можна створити зразок фронт-енд проекту за допомогою Vite.

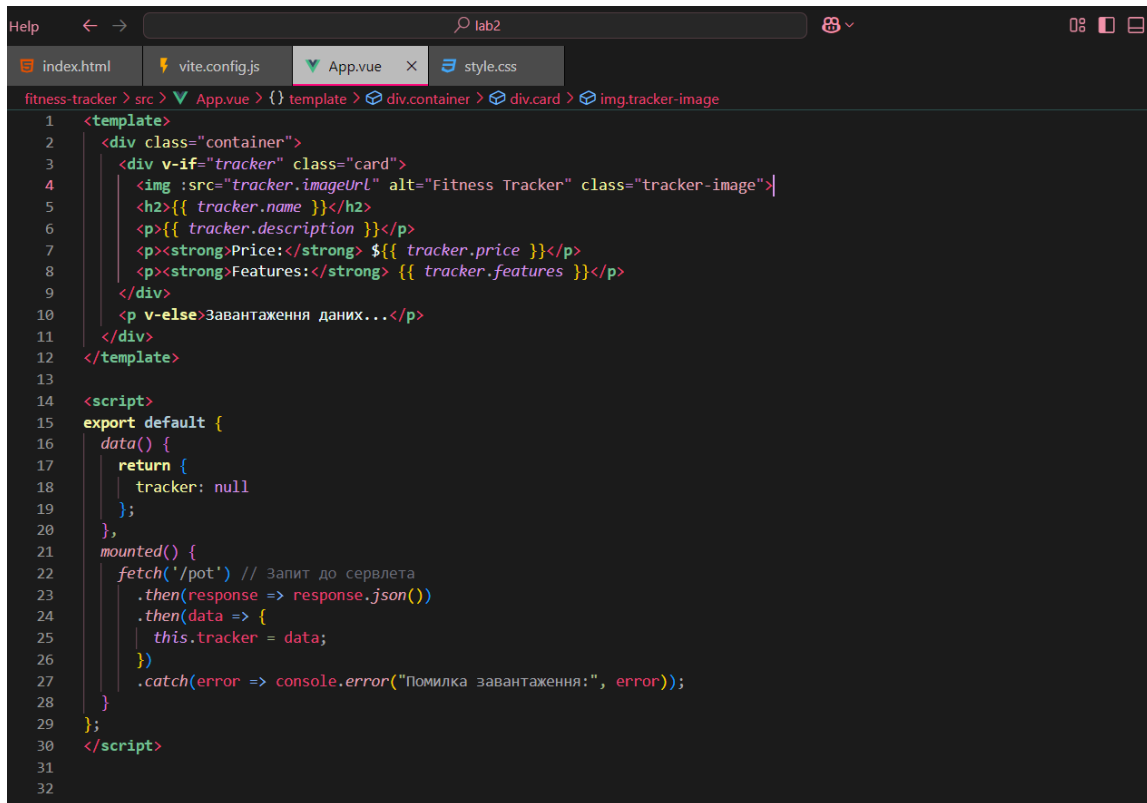




5. Після запуску команд в терміналі потрібно змінити App.vue, main.js, index.html і vite.config.js та видалити непотрібні файли



6. Розробити компонент в App.vue для отримання та відображення даних через HTTP-запит.



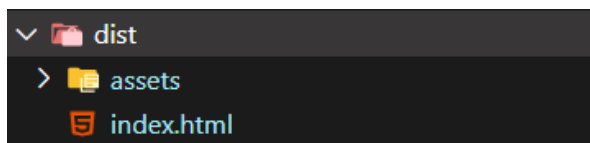
```
1 <template>
2   <div class="container">
3     <div v-if="tracker" class="card">
4       
5       <h2>{{ tracker.name }}</h2>
6       <p>{{ tracker.description }}</p>
7       <p><strong>Price:</strong> ${{ tracker.price }}</p>
8       <p><strong>Features:</strong> {{ tracker.features }}</p>
9     </div>
10    <p v-else>Завантаження даних...</p>
11  </div>
12 </template>
13
14 <script>
15 export default {
16   data() {
17     return {
18       tracker: null
19     };
20   },
21   mounted() {
22     fetch('/pot') // Запит до сервлета
23       .then(response => response.json())
24       .then(data => {
25         this.tracker = data;
26       })
27       .catch(error => console.error("Помилка завантаження:", error));
28   }
29 };
30 </script>
31
32
```

Інтеграція Front-end і Back-end:

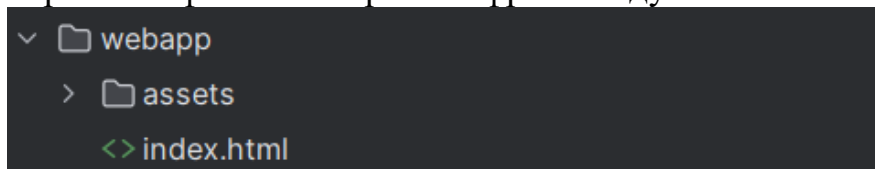
1. Налаштувати проксі-сервер у Vite для роботи з сервлетом.
2. Виконайте збірку проєкту командою

```
npm run build
```

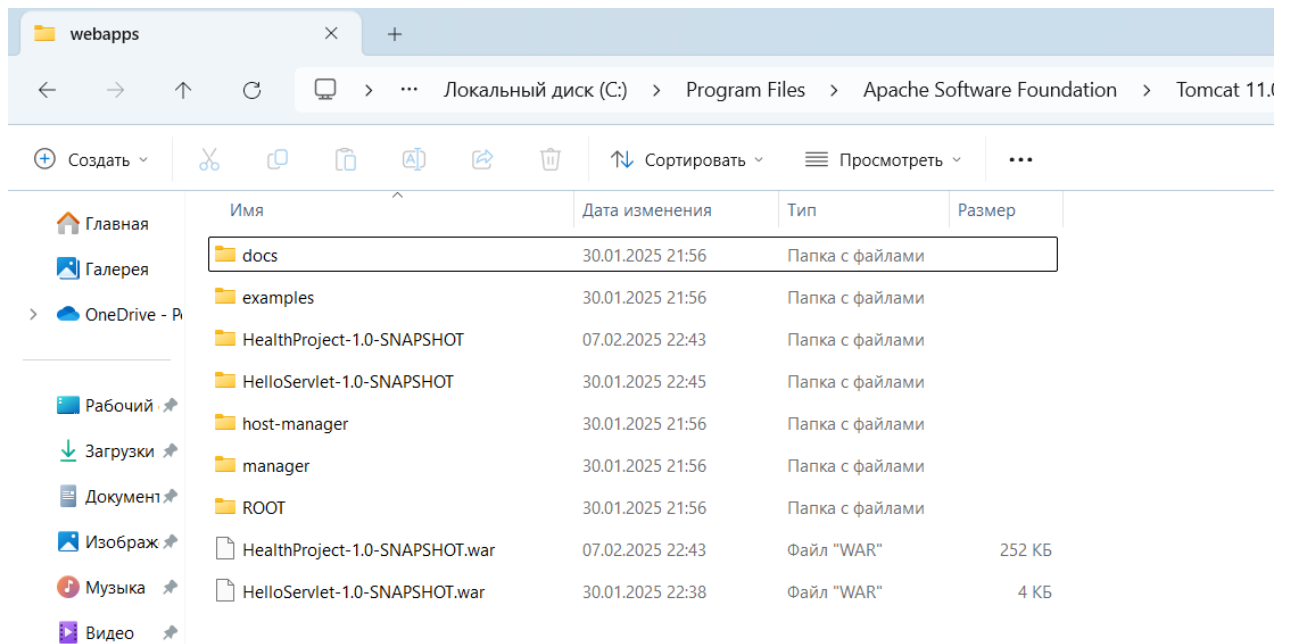
Це перемістить файли до папки dist для інтеграції з сервлетом.



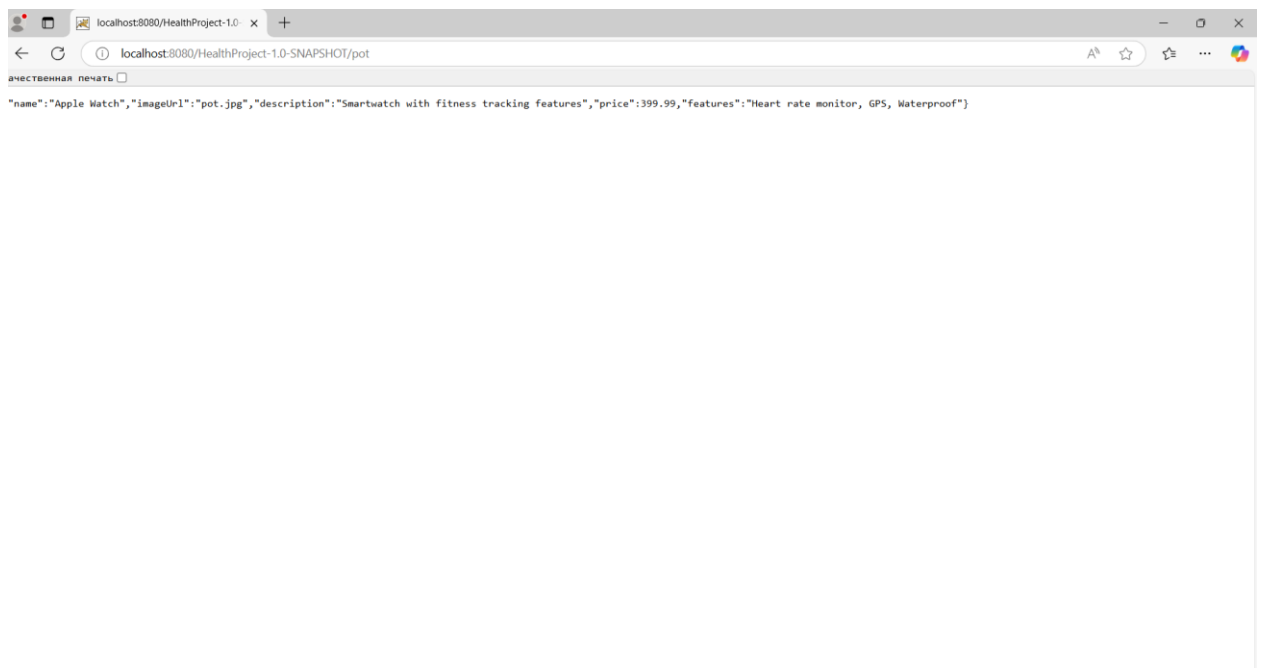
3. Перевірити коректність обміну даними між Vue та сервлетом.
4. У Java-проєкті створити директорию src/main/webapp куди перенести файли із зібраного фронт-енду з папки dist.

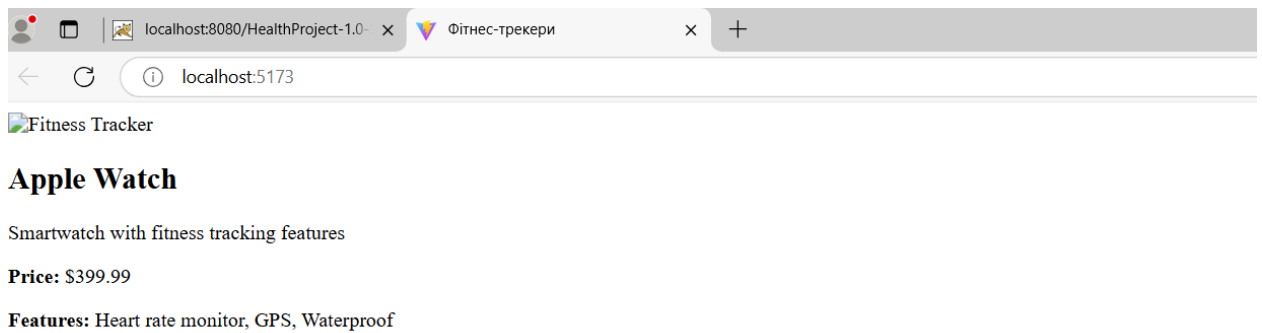


5. Зібрати Java-проєкт за допомогою Maven.
6. Скопіювати зібраний .war файл із Java-проєкту до папки webapps у директорії Tomcat.



7. Перейти до папки `bin` у директорії Tomcat.
8. Запустити сервер за допомогою команди `startup.bat` (для Windows) або `./startup.sh` (для Linux/macOS).
9. Переконайся, що сервлет доступний за відповідним URL (localhost:8080/HealthProject-1.0-SNAPSHOT/pot)





Запитання для самоперевірки:

1. Які функції виконує сервлет у проєкті?

Сервлет у проєкті виконує функцію обробки HTTP-запитів (наприклад, GET-запитів) від клієнта (фронт-енду). Він отримує дані, обробляє їх (наприклад, звертається до бази даних або іншого джерела), серіалізує їх у формат JSON та повертає відповідь клієнту.

2. Як відбувається обмін даними між фронт-ендом і бек-ендом?

Обмін даними відбувається через HTTP-запити. Фронт-енд (Vue.js) відправляє запит до сервлету (back-end), який обробляє запит і повертає дані у форматі JSON. Vue.js отримує ці дані та відображає їх на сторінці.

3. Що таке JSON і чому він використовується для передачі даних?

JSON (JavaScript Object Notation) — це текстовий формат для зберігання та передачі структурованих даних. Він використовується через свою простоту, легкість читання для людей та машин, а також через те, що він легко інтегрується з JavaScript, що робить його ідеальним для веб-додатків.

4. Як налаштувати Vue для інтеграції з Java-сервлетами?

Для інтеграції Vue з Java-сервлетами потрібно:

1. Налаштувати проксі-сервер у Vite для перенаправлення запитів до сервлету.

2. Використовувати бібліотеку axios або вбудований fetch у Vue для відправки HTTP-запитів до сервлету.
3. Обробляти відповіді від сервлету та відображати дані у компонентах Vue.

5. Як працює бібліотека Gson для серіалізації об'єктів у JSON?

Бібліотека Gson дозволяє конвертувати Java-об'єкти у JSON та навпаки. Вона автоматично серіалізує поля об'єкта у JSON-рядок і десеріалізує JSON-рядок у Java-об'єкт.

6. Які параметри можна задати для опису вашого об'єкта?

Параметри об'єкта залежать від вимог проєкту. Наприклад, для фітнес-трекера це можуть бути:

1. Назва трекера.
2. Опис.
3. Зображення.
4. Характеристики (вага, розмір, колір тощо).
5. Дані про здоров'я (пульс, кількість кроків, спалених калорій).

7. Як обробляти помилки при отриманні даних у Vue?

Помилки можна обробляти за допомогою блоку catch у axios або fetch. Наприклад:

```
axios.get('/api/data')
  .then(response => {
    // Обробка успішної відповіді
  })
  .catch(error => {
    console.error('Помилка при отриманні даних:', error);
  });
```

Також можна використовувати глобальні обробники помилок або показувати повідомлення користувачу.

Висновок

У ході виконання лабораторної роботи я створила пілотну версію інформаційної системи для відстеження даних про здоров'я за допомогою фітнес-трекера. Розробка включала створення back-end на основі Java Servlet

для обробки запитів та повернення даних у форматі JSON, а також front-end на базі Vue.js для відображення цих даних.

Основні етапи роботи включали:

1. Налаштування Maven-проєкту з підтримкою Java Servlet API та бібліотеки Gson.
2. Розробку сервлету для обробки GET-запитів.
3. Створення Vue.js-додатку за допомогою Vite.
4. Інтеграцію фронт-енду та бек-енду через проксі-сервер.
5. Збірку та розгортання проєкту на сервері Tomcat.

В результаті було досягнуто коректного обміну даними між клієнтом та сервером, що підтверджує працездатність створеної системи. Робота дозволила закріпити навички роботи з Java Servlet, Vue.js, JSON та інтеграції front-end і back-end.