МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Навчально-науковий інститут електричної інженерії
та інформаційних технологій

КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ЗВІТ

З ЛАБОРАТОРНИХ РОБІТ

З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

«Frontend-розробка»

Виконав студент групи КН-23-1

Полинько Ігор Миколайович

Перевірив старший викладач кафедри АІС Притчин О. С.

КРЕМЕНЧУК 2025

Лабораторна робота № 5

Тема: Робота з вебзапитами та API в JavaScript та React

Мета: Познайомитись і отримати головні навички роботи з відправкою вебзапитів, обробкою відповіді від сервера та API в JavaScript та React

Виконання завдання лабораторної роботи:

Знайдіть будь-який безкоштовний API без авторизації. Створіть просту html-сторінку для виводу інформації з цього API, з кнопкою для її отримання. Напишіть код, що отримуватиме дані з обраного API через fetch з обробкою помилок і виведіть частину отриманої інформації на сторінку при її відкритті і частину при натисканні на кнопку. Для цих двох частин мають бути свої запити. Мінімум один для отримання інформаціїї при відкитті і один для отримання даних по кнопці.

Перепишіть цей код на axios, закоментувавшм варіант з fetch.

Переробіть свою сторінку на проєкт React. Виконайте почергово знову варіанти з fetch та axios. Для виконання запитів по кнопці достатньо відправляти їх в обробнику події. Для запитів, що відправляються при відкритті сторінки в React використовується хук useEffect з порожнім масивом залежностей. Код запиту написаний в ньому виконається один раз при старті.

Не забудьте використати state компонента для зберігання отриманих з API даних.

Fetch:

```
import React, { useState, useEffect } from "react";

function App() {
  const [ageOnLoad, setAgeOnLoad] = useState(null);
  const [ageOnClick, setAgeOnClick] = useState(null);
  const [nameInput, setNameInput] = useState("");
  const [error, setError] = useState(null);

  useEffect(() => {
    fetch("https://api.agify.io/?name=ihor")
      .then((res) => {
        if (!res.ok) throw new Error(`Помилка: ${res.status}`);
        return res.json();
      })
```

```
      .then((data) => setAgeOnLoad(data))
      .catch((err) => setError(err.message));
  }, []);

  const handleClick = () => {
    if (!nameInput) return;
    fetch(`https://api.agify.io/?name=${nameInput}`)
      .then((res) => {
        if (!res.ok) throw new Error(`Помилка: ${res.status}`);
        return res.json();
      })
      .then((data) => setAgeOnClick(data))
      .catch((err) => setError(err.message));
  };

  return (
    <div
      style={{
        display: "flex",
        flexDirection: "column",
        alignItems: "center"
      }}
    >
      <h1>Робота з API (Fetch)</h1>

      <h2>Дані при завантаженні сторінки:</h2>
      {ageOnLoad ? (
        <p>{`Ім'я: ${ageOnLoad.name}, вік: ${ageOnLoad.age}`}</p>
      ) : (
        <p>Завантаження...</p>
      )}

      <h2>Введіть ім'я для перевірки:</h2>
      <input
        type="text"
        value={nameInput}
        onChange={(e) => setNameInput(e.target.value)}
        placeholder="Введіть ім'я"
      />
      <button onClick={handleClick}>Отримати дані</button>

      <h2>Дані по кнопці:</h2>
      {ageOnClick ? (
        <p>{`Ім'я: ${ageOnClick.name}, вік: ${ageOnClick.age}`}</p>
      ) : (
        <p>Натисніть кнопку</p>
      )}

      {error && <p style={{ color: "red" }}>{error}</p>}
    </div>
  );
}
```

```
export default App;
```

Axios:

```
import React, { useState, useEffect } from "react";
import axios from "axios";

function App() {
  const [ageOnLoad, setAgeOnLoad] = useState(null);
  const [ageOnClick, setAgeOnClick] = useState(null);
  const [nameInput, setNameInput] = useState("");
  const [error, setError] = useState(null);

  useEffect(() => {
    const fetchOnLoad = async () => {
      try {
        const res = await axios.get("https://api.agify.io/?name=ihor");
        setAgeOnLoad(res.data);
      } catch (err) {
        setError(err.message);
      }
    };

    fetchOnLoad();
  }, []);

  const handleClick = async () => {
    if (!nameInput) return;

    try {
      const res = await axios.get(`https://api.agify.io/?name=${nameInput}`);
      setAgeOnClick(res.data);
    } catch (err) {
      setError(err.message);
    }
  };

  return (
    <div style={{ display: "flex", flexDirection: "column", alignItems: "center"
}}>
      <h1>Робота з API (Axios)</h1>

      <h2>Дані при завантаженні сторінки:</h2>
      {ageOnLoad ? (
        <p>{`Ім'я: ${ageOnLoad.name}, вік: ${ageOnLoad.age}`}</p>
      ) : (
        <p>Завантаження...</p>
      )}

      <h2>Введіть ім'я для перевірки:</h2>
```

```
      <input
        type="text"
        value={nameInput}
        onChange={(e) => setNameInput(e.target.value)}
        placeholder="Введіть ім'я"
      />
      <button onClick={handleClick}>Отримати дані</button>

      <h2>Дані по кнопці:</h2>
      {ageOnClick ? (
        <p>{`Ім'я: ${ageOnClick.name}, вік: ${ageOnClick.age}`}</p>
      ) : (
        <p>Натисніть кнопку</p>
      )}

      {error && <p style={{ color: "red" }}>{error}</p>}
    </div>
  );
}

export default App;
```

Результат:

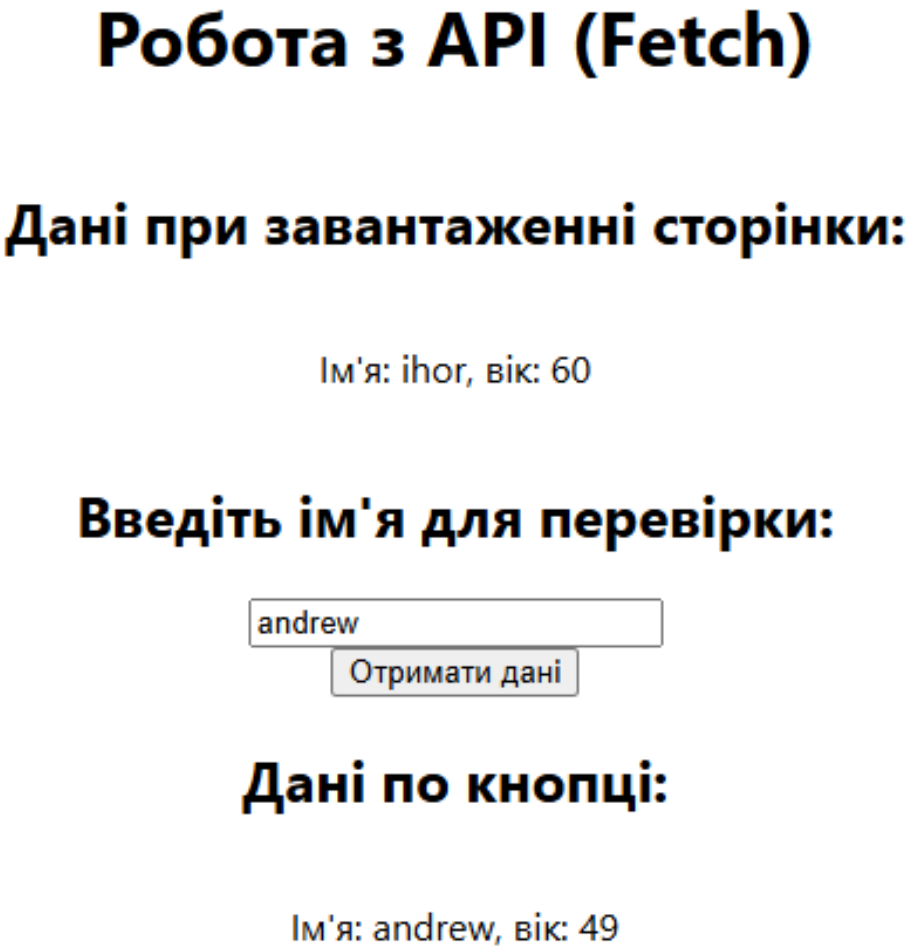

Рисунок 5.1 – Робота Fetch

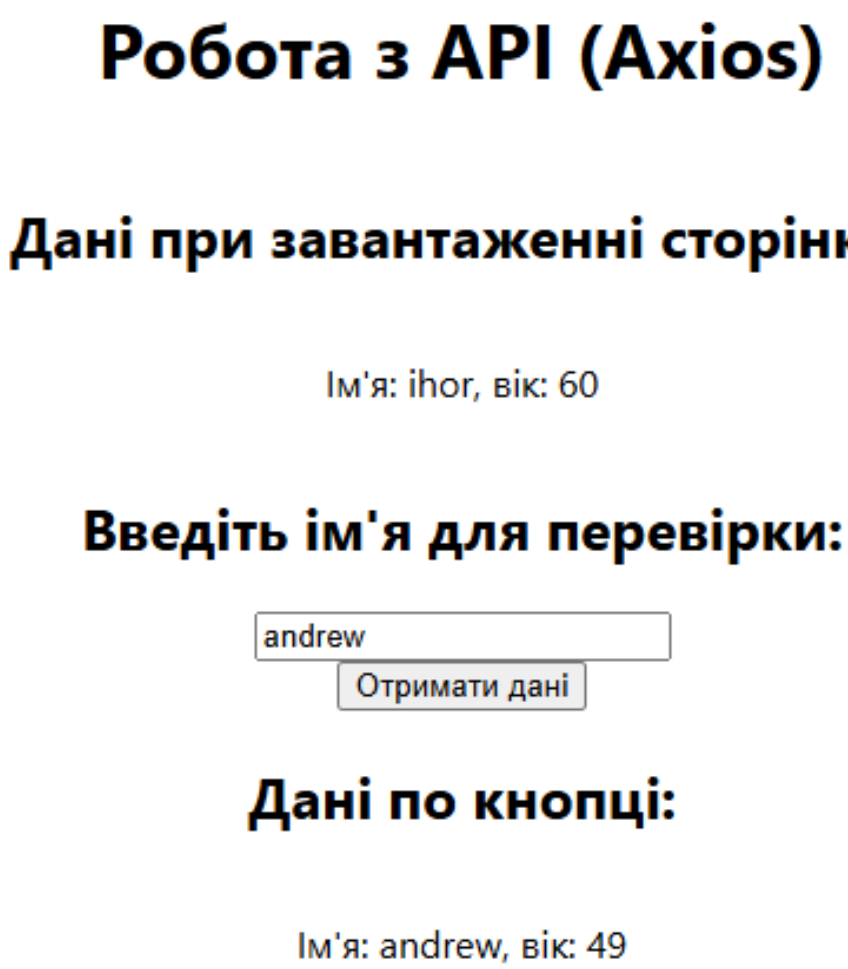

Рисунок 5.2 – Робота Axios

Висновок: на цій лабораторній роботі ми познайомились і отримали головні навички роботи з відправкою вебзапитів, обробкою відповіді від сервера та API в JavaScript та React.

Контрольні питання:

1. Функція JSON.parse()

JSON.parse() використовується для перетворення рядка у форматі JSON на відповідний JavaScript-об'єкт.

Приклад:

```
const jsonString = '{"name":"Ihor","age":22}';
const obj = JSON.parse(jsonString);
console.log(obj.name);
```

2. Функція JSON.stringify()

JSON.stringify() перетворює JavaScript-об'єкт у рядок JSON, який можна передавати через HTTP-запити або зберігати як текст.

Приклад:

```
const obj = { name: "Ihor", age: 22 };
const jsonString = JSON.stringify(obj);
```

3. Отримання даних з response при Fetch

При використанні fetch() функція повертає Promise, що містить об’єкт Response.

Для отримання даних у форматі JSON потрібно викликати метод .json(), який також повертає Promise:

```
fetch("https://api.example.com/data")
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error(error));
```

4. Випадки відхилення Promise у Fetch

– Promise відхиляється тільки при мережевих помилках, наприклад:

– Відсутнє інтернет-з’єднання.

– Сервер не відповідає.

– Проблеми з CORS.

– HTTP-статуси помилок (404, 500) не відхиляють Promise, тому їх необхідно перевіряти вручну через response.ok або response.status.

5. Основні переваги Axios над Fetch

1) Автоматичний парсинг JSON – дані доступні через res.data.

2) Автоматична обробка HTTP-помилок – Promise відхиляється при статусах 4xx і 5xx.

3) Простий синтаксис для різних методів HTTP (GET, POST, PUT, DELETE).

4) Можливість легко додавати заголовки, таймаути та авторизацію.

5) Працює як у браузері, так і в Node.js.

6) Підтримка перехоплювачів запитів і відповідей, що зручно для централізованого логування або обробки токенів.

6. Відсутність параметру method у методах Axios

Методи axios.get(), axios.post(), axios.put(), axios.delete() автоматично визначають HTTP-метод, тому додатковий параметр method не потрібен. Приклад явного використання:

```
axios({
  method: "POST",
  url: "https://api.example.com/data",
  data: { name: "Ihor" }
});
```