ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка веб-інтерфейсу з використанням сучасних фреймворків React, Angular та Vue.js»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

	Ілля АНДРІЄНКО
(підпис)	*Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-41
Ілля АНДРІЄНКО
Ім'я, ПРІЗВИЩЕ
Керівник: Борис КОНДРАТЮК
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент:
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Андрієнко Ілля Миколайович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка веб-інтерфейсу з використанням сучасних фреймворків React, Angular та Vue.js
керівник кваліфікаційної роботи Борис КОНДРАТЮК ,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024р. № 36
2. Строк подання кваліфікаційної роботи «3» червня 2024р.
3. Вихідні дані до кваліфікаційної роботи: _____
_____
_____
_____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Питання огляду та порівняння сучасних фреймворків для веб-розробки.
2. Питання розробки, тестування та розгортання на сервері REST API.
3. Питання розробки CRUD додатків з використанням React, Angular та Vue.
5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «27» лютого 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	28.02-22.03.2024	
2	Обробка матеріалу	23.03-05.04.2024	
3	Розробка теоретичної частини	06.04-10.04.2024	
4	Розробка практичної частини	11.04-30.04.2024	
5	Дослідження проблемних питань	01.05-03.05.2024	
6	Висновки за результатами аналізу	04.05-08.05.2024	
7	Розробка презентації	09.05-11.05.2024	
8	Передзахист	14.05-17.05.2024	
9	Здача в деканат	25.05-03.06.2024	

Здобувач вищої освіти _____ Ілля АНДРІЄНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Борис КОНДРАТЮК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 51 стор., 32 рис., 1 табл., 30 джерел.

Мета роботи – дослідження та порівняння сучасних фреймворків для розробки веб-інтерфейсів, таких як React, Angular та Vue.js, а також розробка практичних рекомендацій щодо їх ефективного використання для створення інтерактивних та високопродуктивних веб-додатків.

Об'єкт дослідження – розробка веб-інтерфейсів з використанням сучасних фреймворків, зокрема React, Angular та Vue.js.

Предмет дослідження – фреймворки React, Angular та Vue.js, їх основні концепції, архітектура, переваги та недоліки, а також методи розробки, тестування та розгортання веб-додатків на їх основі.

Короткий зміст роботи: У сучасному світі веб-розробка відіграє ключову роль у створенні інтуїтивних та ефективних веб-додатків. Вибір правильного інструментарію є важливим аспектом, що впливає на успішність проєкту. Серед популярних інструментів для розробки веб-інтерфейсів виділяються сучасні JavaScript-фреймворки React, Angular та Vue.js.

Результати цього дослідження можуть бути корисними для розробників, які стоять перед вибором інструментів для реалізації своїх проєктів, а також для тих, хто прагне поглибити свої знання у сфері веб-розробки. Висновки, зроблені на основі проведеного аналізу, підтверджують, що належне використання сучасних фреймворків може значно покращити ефективність і якість веб-додатків.

КЛЮЧОВІ СЛОВА: ANGULAR, REACT, VUE, POSTGRESQL, NODE.JS, EXPRESS.JS, REST API, CRUD, POSTMAN.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасні фреймворки веб-розробки, такі як React, Angular та Vue, відіграють вирішальну роль у створенні динамічних та інтерактивних веб-додатків. Вибір відповідного фреймворку може суттєво вплинути на результативність розробки, продуктивність додатку та зручність підтримки коду. Аналіз особливостей та порівняння цих фреймворків є дуже актуальним для веб-розробників, які прагнуть оптимізувати свої процеси розробки.

Мета і завдання дослідження. Метою цього дослідження є детальний огляд та порівняння сучасних фреймворків для веб-розробки - React, Angular та Vue.

Основними завданнями є:

- вивчити історію та шляхи розвитку React, Angular та Vue;
- проаналізувати ключові концепції та архітектуру фреймворків;
- визначити переваги та недоліки кожного фреймворку;
- розробити, протестувати та розгорнути REST API на сервері;
- реалізувати CRUD-операції за допомогою React, Angular та Vue.

Об’єкт дослідження. Об'єктом дослідження є сучасні фреймворки для веб-розробки: React, Angular та Vue. Ці фреймворки допомагають створювати динамічні веб-додатки, спрощуючи процес розробки інтерактивних користувацьких інтерфейсів.

Предмет дослідження. Предметом дослідження є методи та підходи до розробки, тестування та розгортання веб-додатків з використанням фреймворків React, Angular та Vue. Особливу увагу приділено реалізації CRUD-операцій у кожному з цих фреймворків.

Методи дослідження. У дослідженні використовуються методи порівняльного аналізу, огляд літератури, експериментальні методи розробки та тестування програмного забезпечення, а також методи системного аналізу для оцінки ефективності фреймворків.

Теоретична, методична та практична значущість отриманих

результатів. Теоретичне значення полягає в розширенні розуміння сучасних фреймворків веб-розробки та їх архітектури. Методологічне значення полягає в розробці методології порівняння фреймворків. Практичне значення полягає у наданні рекомендацій для веб-розробників щодо вибору правильного фреймворку для різних задач.

Структура роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1. ОГЛЯД ТА ПОРІВНЯННЯ СУЧАСНИХ ФРЕЙМВОРКІВ ДЛЯ ВЕБ-РОЗРОБКИ

1.1 Історія та розвиток React, Angular та Vue

Спочатку давайте познайомимося з фреймворком React. Розроблений компанією Facebook у 2013 році, React швидко став однією з найбільш затребуваних JavaScript-бібліотек для створення користувацьких інтерфейсів і з тих пір заслужив широке визнання серед розробників.

React є незамінним компонентом в управлінні шарами представлення в додатках, діючи як «V» в MVC для забезпечення ефективного рендерингу. Замість того, щоб розглядати користувацькі інтерфейси як єдине ціле, React заохочує розробників розбивати складні користувацькі інтерфейси на дискретні багаторазові компоненти, які слугують будівельними блоками всього інтерфейсу. Завдяки такому підходу ReactJS поєднує швидкість та ефективність JavaScript з більш продуктивними методами маніпулювання DOM, що дозволяє пришвидшити рендеринг та створювати високоінтерактивні веб-додатки [1].

Давайте розглянемо його історію. У 2011 році інженери Facebook зіткнулися з труднощами при масштабуванні свого коду. Користувацькі інтерфейси ставали дедалі складнішими та важчими в підтримці, що вимагало ефективного рішення, яке могло б оновлювати інтерфейс без необхідності повного оновлення сторінки [1].

Джордан Волке (Jordan Walke), інженер Facebook, заснував React як спробу розробити інструмент опису інтерфейсу за допомогою компонентів. Кожен компонент представляв собою частину інтерфейсу, що спрощувало розробку та підтримку [2].

Вперше React дебютував на американській конференції JSConf у травні 2013 року. Концепція React була створена як відповідь на проблеми, з якими зіткнулися інженери Facebook при розробці динамічних користувацьких інтерфейсів, і її

основною метою було підвищення продуктивності та простоти розробки за рахунок використання компонентного підходу та віртуальних DOM. Хоча спочатку React використовувався лише для проектів з веб-розробки, він швидко завоював популярність у спільноті розробників завдяки своїм інноваційним можливостям управління інтерфейсом користувача [2].

У 2015 році Facebook випустив React Native, щоб поширити принципи та інструменти React на розробку мобільних додатків на платформах iOS та Android. Замість HTML-компонентів, React Native використовує нативні компоненти мобільних платформ для своєї архітектури компонентів, що дозволяє розробникам створювати крос-платформні мобільні додатки з нативною продуктивністю та користувацьким досвідом, використовуючи при цьому досвід та знання React.

React 15 був випущений у 2016 році з суттєвими покращеннями продуктивності та розміру бібліотеки, такими як оптимізоване управління оновленням DOM, що зменшило кількість операцій, необхідних для оновлення сторінок, і ще більше підвищило загальну продуктивність. Крім того, було покращено підтримку компонентів, що використовують стани, а також впроваджено нові можливості рендерингу для більш ефективної оптимізації процесів розробки.

У 2017 році вийшло важливе оновлення React 16 («Fiber»), яке забезпечило значну реструктуризацію внутрішньої архітектури з нуля. Fiber дозволив більш гнучко планувати та виконувати завдання рендерингу, щоб покращити швидкість відгуку інтерфейсів у великих та складних додатках, особливо там, де могли виникати помилки в компонентах. Крім того, React 16 включив нові функції, такі як Error Boundaries для обробки помилок у компонентах, а також підтримку фрагментів, що дозволяє групувати декілька елементів без додаткових обгорток у DOM-дереві.

React 17 був випущений як частина релізу 2020 року і зосереджений на покращенні інтеграції з іншими бібліотеками та фреймворками, а також на спрощенні оновлення версій React у великих проектах. Не впроваджуючи жодних значних нових функцій для розробників, React 17, тим не менш, слугував для

підтримки сумісності між існуючим кодом та новими версіями, а також для полегшення плавного переходу на них. Він дозволив декільком версіям React співіснувати одночасно на одній сторінці, що значно спростило поступове оновлення для додатків.

На сьогоднішній день, React 18 – це остання версія, яка вже доступна і все ще пропонує різноманітні інновації, покликані підвищити продуктивність та полегшити розробку. Одним з таких нововведень стало автоматичне розділення коду, яке підвищило швидкість завантаження додатку, зменшивши час до першого рендерингу. Крім того, було покращено підтримку рендерингу на стороні сервера, включно з можливостями потокового та інкрементного рендерингу, що ще більше підвищило продуктивність та зручність роботи з великими додатками. Крім того, було оптимізовано обробку станів компонентів, а також додано нові API для керування асинхронними завданнями.

Другий фреймворк, який ми будемо розглядати це Angular. AngularJS - це відомий фреймворк з відкритим вихідним кодом, який спрощує веб-розробку, дозволяючи створювати інтерактивні односторінкові додатки (SPA). На відміну від традиційних веб-сайтів, які завантажують нові сторінки щоразу, коли користувач натискає на посилання, SPA пропонують покращений користувацький досвід, оновлюючи вміст безпосередньо на поточній сторінці.

AngularJS допомагає досягти цього, перетворюючи статичний HTML на динамічний контент, який реагує на дії користувача. Такі функції, як двостороннє зв'язування даних та ін'єкція залежностей, спрощують розробку, а регулярні оновлення гарантують, що ваші веб-додатки залишатимуться актуальними та продуктивними [3].

Його історія бере свій початок у 2009 році, коли Міško Хевері (Misko Hevery) та Адам Абронс (Adam Abrons), інженери Google, розробили фреймворк, відомий як AngularJS, і офіційно випустили його у 2010 році (див. рис. 1.1) [3].

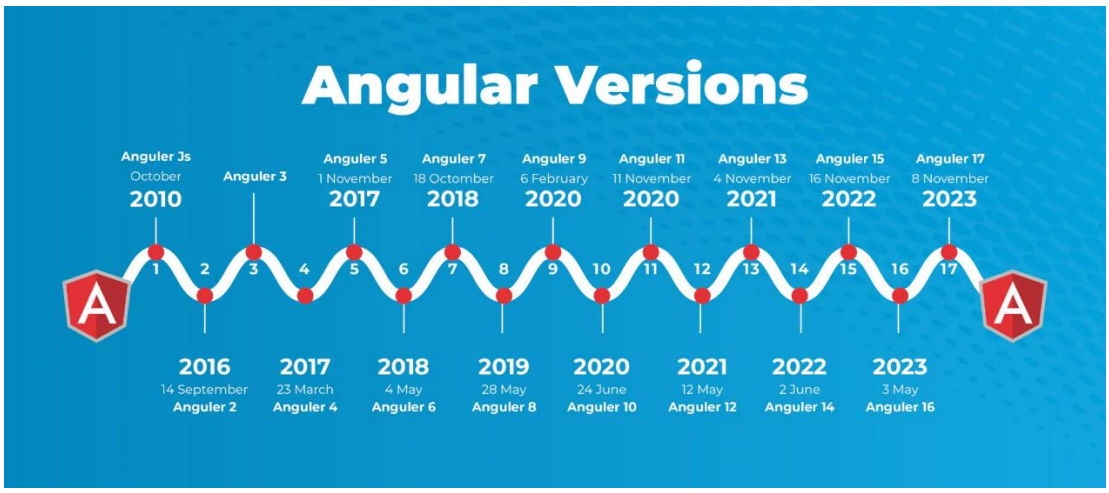


Рисунок 1.1 - Версії фреймворка Angular та рік їх випуску

З часом виявилося, що архітектура AngularJS мала певні обмеження, які ускладнювали створення великих і складних додатків. У зв'язку з цим команда Angular прийняла рішення розробити нову версію фреймворку, яка б відповідала сучасним вимогам веб-розробки.

У вересні 2016 року було представлено Angular 2, що представляє собою повністю переписаний з нуля фреймворк. Заснована на принципах модульності та використанні TypeScript як мови розробки, ця нова версія значно підвищила продуктивність, гнучкість та легкість розробки[4].

У березні 2017 року (версія 3 не була включена, щоб уникнути плутанини з іншими бібліотеками), Angular 4 був офіційно представлений з численними поліпшеннями в плані продуктивності, розміру додатків і новими можливостями для роботи з анімацією і формами [4].

Angular 5, випущений у листопаді 2017 року, продовжив покращення продуктивності, досягнуті в попередніх версіях, і забезпечив кращу підтримку прогресивних веб-додатків (PWA). Тим часом, у травні 2018 року вийшов Angular Elements, який уможливив безперешкодну інтеграцію з іншими фреймворками та бібліотеками, а також інтерфейс командного рядка для спрощення процесів розробки [5].

Angular 7 та 8, випущені відповідно у жовтні 2018 та травні 2019 року, запропонували додаткові покращення продуктивності, функції маршрутизації та

підтримку стандартів веб-розробки. Особливо слід відзначити підтримку динамічного імпорту модулів та рушій рендерингу Ivy - дві функції, які стали основними в Angular 9 [5].

Починаючи з Angular 9, випущеного в лютому 2020 року, рушій рендерингу Ivy став стандартним рушієм рендерингу для всіх додатків. Ця зміна призвела до значного підвищення продуктивності, зменшення розміру пакету, спрощення процесу налагодження та покращення сумісності з бібліотеками також стало результатом цього переходу.

Наступні версії, Angular 10 (червень 2020), 11 (листопад 2020) та 12 (травень 2021), забезпечили значне покращення стабільності, продуктивності та підтримки нових функцій. Особливо це стосується підтримки TypeScript 4.2, можливостей локалізації та покращення роботи з CLI [6].

Angular 13, випущений у листопаді 2021 року, включав оновлення для поліпшення підтримки нових версій TypeScript і Angular CLI, а також функції для роботи з формами і управління станом програми.

Angular 14, випущений у червні 2022 року, надав нові можливості для створення автономних компонентів, покращив роботу з формами та розширив підтримку Angular CLI. До листопада 2022 року в ньому було впроваджено покращення для маршрутів, можливості анімації та підвищення продуктивності - все це було ще більше покращено з виходом 15-ї версії.

Випущений у червні 2023 року, Angular 16 зосередився на подальшому підвищенні продуктивності, зменшенні розмірів пакунків та нових інструментах для розробників.

Angular 17 вийшов у листопаді 2023 року з кількома помітними покращеннями для сумісності з бібліотеками та фреймворками, а також функціями для моніторингу стану додатків.

Angular 18, випущений у травні 2024 року, представив автоматичне розділення коду, покращену підтримку рендерингу на стороні сервера та нові API для управління асинхронними завданнями. У ньому також з'явилися оновлення, призначені для покращення роботи з інтерфейсом командного рядка, а також

підтримка останніх версій TypeScript.

Останнім фреймворком буде Vue. Vue.js може похвалитися своєю назвою, що він наповнений безліччю функцій, але насправді він надає лише найнеобхідніше, що очікується від будь-якого JavaScript-фреймворку.

Vue.js є ідеальним фреймворком для спрощення життя у цьому складному світі технологій та досягнень, який, здається, ніколи не закінчиться [7].

Додатки на Vue.js дозволяють розробникам починати з малого і створювати більші та кращі, тоді як інший фреймворк зробив би складність опцією за замовчуванням [7].

Vue.js - це JavaScript-фреймворк для створення користувацьких інтерфейсів (UI) та односторінкових додатків (SPA). Як і найкращі з них, Vue.js має відкритий вихідний код. Він використовує архітектурний патерн «model-view-viewmodel» (MVVM) [8].

VueJS створив Еван Ю (Evan You). Він працював у Google над проектами на AngularJS. Він витягнув з AngularJS частини, які нам дуже сподобалися, і створив щось дійсно легке. Перший вихідний код був випущений в липні 2013 року, а VueJs був випущений в лютому 2014 року [8].

Vue.js (1.x) пропонував розробникам ключові функції, які робили його привабливим а саме:

– декларативний рендеринг надає можливість визначати вигляд інтерфейсу за допомогою шаблонів;

– архітектура на основі компонентів підтримує створення багаторазових компонентів для побудови складних інтерфейсів;

– реактивність забезпечує автоматичне оновлення при зміні даних;

– двостороння прив'язка даних спрощує синхронізацію даних між моделями та поданнями.

Vue.js 2.0 був офіційно представлений у вересні 2016 року, пропонуючи значні оновлення та функції, які полегшили життя такі як [7]:

– віртуальний DOM. Для покращення продуктивності рендерингу та зменшення кількості прямих маніпуляцій з DOM;

– покращена система компонентів. Ця система дозволяє швидше створювати складні компоненти з гнучкими інтерфейсами;

– розширена реактивність. Покращення механізму реактивності забезпечують більшу гнучкість і продуктивність.

Та оновлений у вересні 2020 року, Vue.js 3.0 був офіційно представлений і мав різні покращення та можливості [8]:

– composition API. Інноваційний метод розробки компонентів з більшою гнучкістю та можливістю повторного використання логіки;

– покращена продуктивність. Оптимізація для зменшення розміру бібліотеки та прискорення швидкості рендерингу;

– підтримка TypeScript. Покращена підтримка TypeScript забезпечує більш плавну інтеграцію з сучасними інструментами розробки.

React, Angular та Vue - три найвідоміші JavaScript-фреймворки, які сьогодні використовуються для веб-розробки, а їхня історія створення демонструє різноманітність підходів до створення сучасних користувацьких інтерфейсів.

Кожен фреймворк пропонує власні інновації та підходи до розробки веб-додатків, залишаючись при цьому популярним і широко застосовуваним у різних галузях.

1.2 Основні концепції та архітектура фреймворків

React побудований на наборі ключових концепцій та архітектур, таких як (див. рис. 1.2) [9]:

– компонентний підхід;

– JSX;

– односпрямований потік даних;

– віртуальний DOM (Virtual DOM);

– стан та пропси (State та Props);

– життєвий цикл компонента;
– контекст (Context);
– хуки (Hooks).

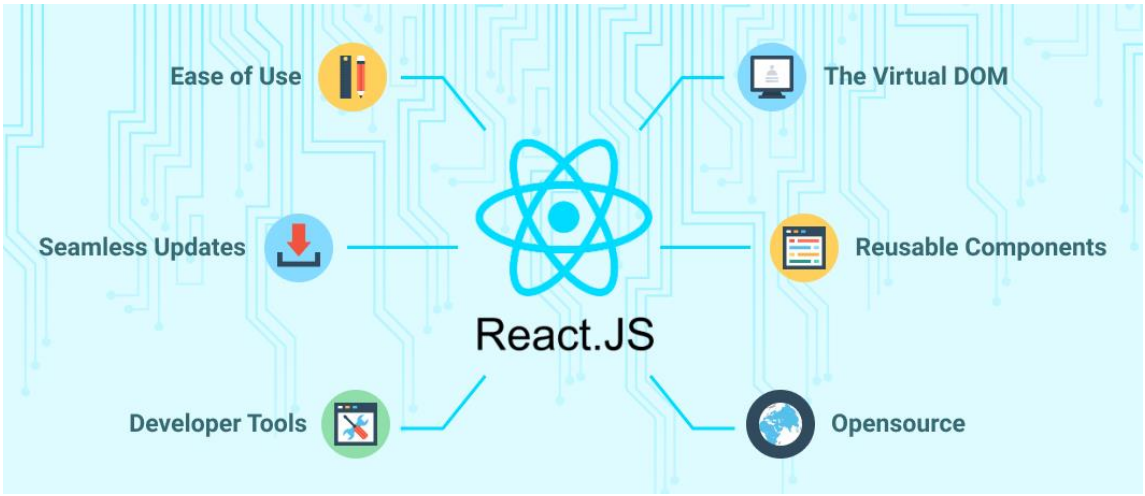


Рисунок 1.2 - Основні концепції та архітектура React.JS

Фундаментом React-додатків – є компоненти. Ці частини коду можуть мати власний стан та логіку, що визначає їхню функціональність або безстатусність (stateless) чи клас (stateful).

Функціональні компоненти можна визначити як функції, що отримують властивості (вхідні дані) в якості аргументів і повертають React-елементи, які описують, що має з'явитися на екрані.

Класи, що розширюють React.Component, компоненти можуть мати внутрішній стан і методи життєвого циклу, які дозволяють їм функціонувати як завгодно.

JSX (JavaScript XML) - це розширення файлів JavaScript, яке дозволяє користувачам писати код, що нагадує HTML, більш інтуїтивно, ніж звичайний HTML, що робить створення та редагування елементів інтерфейсу користувача простішим.

Односторонній потік даних дозволяє передавати дані від батьківських компонентів до дочірніх через властивості, створюючи передбачувану поведінку програми та спрощуючи управління станами.

React використовує віртуальну DOM як механізм для ефективного оновлення

користувацьких інтерфейсів. Замість того, щоб безпосередньо маніпулювати реальним DOM, React використовує віртуальний DOM - полегшену копію реального DOM. Коли стан окремого компонента змінюється, React створює нове віртуальне DOM-дерево і порівнює його з попередньою копією, перш ніж оновити реальний DOM лише з необхідними змінами.

Властивості (Props) - це незмінні об'єкти, що передаються між батьківськими та дочірніми компонентами для визначення того, як компонент має виглядати та поводитися, тоді як стани (State) відносяться до будь-яких змінних даних, які можуть зустрічатися всередині самого компонента, і коли вони змінюються, це змушує перерендерити цей компонент.

React надає методи життєвого циклу, які дозволяють розробникам виконувати дії на різних етапах життєвого циклу компонента.

Основні методи життєвого циклу включають [9]:

– componentDidMount(). Спрацьовує після початкового рендерингу компонента;

– componentDidUpdate(prevProps, prevState). Спрацьовує, коли компонент було оновлено або змінено будь-яким чином;

– componentWillUnmount(). Викликається перед видаленням з DOM.

Контекст (Context) дозволяє легко передавати дані між компонентами без необхідності вручну передавати властивості на кожному рівні. Це особливо корисно для глобальної інформації, такої як теми або автентифікація користувачів.

Хуки (Hooks) - це функції, які дозволяють використовувати стан та інші можливості React у функціональних компонентах.

Основні хуки включають [9]:

– useState. Для додавання стану до функціональних компонентів;

– useEffect. Для виконання побічних ефектів у функціональних компонентах;

– useContext. Використання контексту, коли це доречно у функціональних компонентах.

React - незамінна бібліотека для створення сучасних веб-додатків. Її

компонентна архітектура, використання JSX, віртуального DOM та одностороннього потоку даних роблять розробку інтерфейсів зручною та ефективною, а подальше спрощення та розширення роботи з функціональними компонентами за допомогою хуків зробило React одним з найбільш затребуваних інструментів веб-розробки.

Наступний фреймворк концепції та архітектуру якого буде розглядатися – Angular. Ось основні концепції та архітектура Angular [10]:

а) основні концепції Angular;

1) компоненти (Components);

2) шаблони (Templates);

3) модулі (Modules);

4) директиви (Directives);

5) сервіси (Services) та впровадження залежностей (Dependency Injection);

6) зв'язування даних (Data Binding);

б) архітектура Angular (див. рис. 1.3);

1) модульність;

2) компонентно-орієнтована архітектура;

3) система маршрутизації (Routing);

4) RxJS та реактивне програмування.

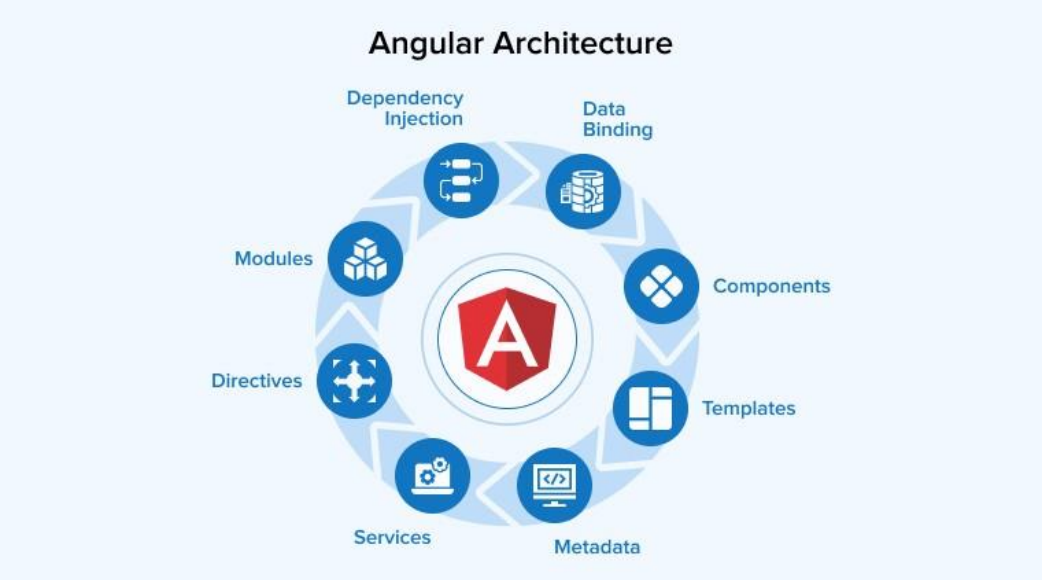


Рисунок 1.3 – Архітектура фреймворка Angular

Компоненти є основою Angular-додатків. Кожен компонент містить HTML-шаблон, логіку TypeScript і стилі CSS. Вони організовані в дерево компонентів додатку, щоб представити його структуру та взаємодію.

Шаблони визначають вигляд і структуру компонентів, використовуючи Angular прив'язки і директиви для прив'язки даних, умовного рендерингу, циклів і обробки подій.

Angular-додаток складається з одного або декількох модулів. Кожен з них має файл NgModule, який описує компоненти, директиви та сервіси, що входять до складу цього модуля. Модуль AppModule є головним.

Сервіси надають спільну функціональність, яку можна використовувати в різних частинах програми. Використовуючи ін'єкцію залежностей, Angular забезпечує створення та розповсюдження екземплярів сервісів.

Angular підтримує кілька форм зв'язування даних [10]:

- одностороннє зв'язування;
- двостороннє зв'язування;
- зв'язування властивостей;
- зв'язування подій.

Angular-додаток будується на основі модулів, які дозволяють розробникам розділити функціонал на керовані блоки для полегшення обслуговування та розширення свого додатку.

Кожен компонент користувацького інтерфейсу реалізований як окремий компонент, і ці вкладені компоненти дозволяють формувати складні ієрархії користувацького інтерфейсу.

DI (Dependency Injection) є ключовим аспектом Angular, що спрощує управління залежностями між різними частинами програми. Angular автоматично вбудовує залежності в компоненти та сервіси.

Маршрутизація дозволяє розробникам визначати шляхи, що відповідають компонентам у додатку, і полегшує навігацію між різними частинами.

Бібліотека RxJS інтегрується з Angular для ефективної обробки асинхронних

операцій за допомогою Observables. Це забезпечує ефективне управління даними та подіями в додатку.

Це лише основи, але вони дають уявлення про те, як Angular організовує та реалізує ключові концепції, необхідні для створення сучасних веб-додатків.

І останнім звісно ж буде Vue. Основні концепції та архітектура Vue в себе включають [11]:

а) основні концепції Vue (див. рис. 1.4);

1) компоненти (Components);
2) шаблони (Templates);
3) директиви (Directives);
4) реактивність (Reactivity);
5) властивості та події (Props and Events);
6) композиційний API (Composition API);

б) архітектура Vue;

1) односторінкові додатки (Single Page Applications, SPA);
2) стан управління (State Management);
3) шаблонна архітектура (Template Architecture);
4) плагіни та модулі (Plugins and Modules).

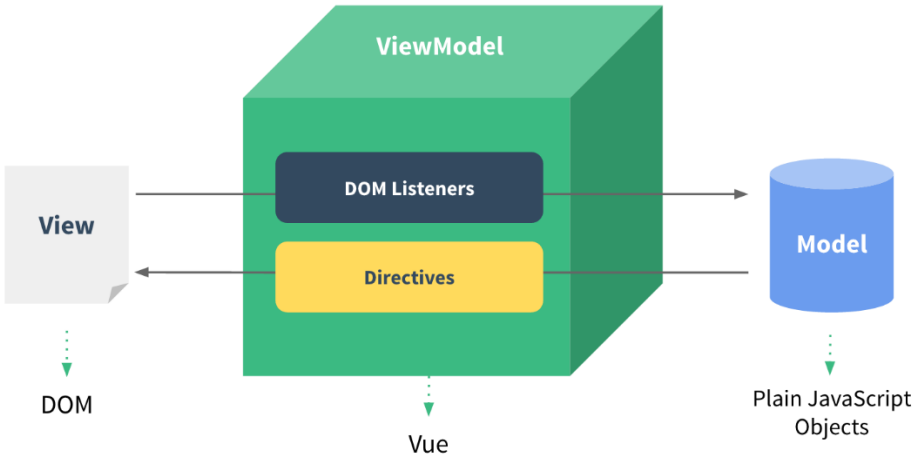


Рисунок 1.4 - Основна концепція фреймворка Vue

Базовими елементами додатків Vue – є компоненти. Кожен з них оснащений

власним шаблоном, сценарієм і таблицею стилів; крім того, їх можна вкладати і використовувати багаторазово для більшої універсальності і повторного використання.

Шаблони включають HTML з вбудованими директивами Vue для з'єднання елементів DOM з даними компонентів.

Директиви надають спеціальні атрибути для додавання логіки до шаблонів. Доступні директиви включають v-if, v-else, v-for, v-show, v-bind і v-model [11].

Vue використовує реактивну систему даних, яка автоматично оновлює DOM при зміні стану даних.

Властивості дозволяють батьківським компонентам передавати дані дочірнім компонентам, а дочірні компоненти можуть створювати події, щоб сповіщати батьківські про зміни.

Альтернативний метод організації та використання логічних компонентів за допомогою функцій замість опцій.

Vue Router може допомогти організувати маршрутизацію в додатку і дозволяє швидко і ефективно створювати багатосторінкові додатки з компонентами.

Vuex використовується для управління станами у великих додатках, забезпечуючи централізоване сховище станів програми. Це забезпечує ефективне та масштабоване адміністрування станів.

Компоненти поділяються на шаблони (HTML), логіку (JavaScript) і стилі (CSS) для полегшення розробки та обслуговування.

Vue пропонує систему плагінів для розширення своїх можливостей. Це полегшує використання та інтеграцію сторонніх бібліотек.

Vue є гнучким і потужним фреймворком для розробки сучасних веб-додатків, завдяки цим концепціям та архітектурним елементам, які складають його основу.

Фреймворки Angular, React та Vue втілюють основні концепції та архітектурні підходи в сучасній веб-розробці. Кожен фреймворк має свої переваги, які роблять їх придатними для певних типів проектів.

1.3 Переваги та недоліки фреймворків

React зарекомендував себе як ключова бібліотека для створення динамічних та адаптивних веб-додатків з наступних причин [12]:

- архітектура на основі компонентів;
- покращена кастомізація в розробці;
- перспективний вибір для розробників;
- редизайн для управління станом.

Традиційні JavaScript-додатки часто мають проблеми з управлінням станом при масштабуванні. React, однак, пропонує складні, незалежно підтримувані та багаторазові компоненти, що дозволяють розробникам оновлювати частини веб-сторінки, не впливаючи на інші - для забезпечення вільного зв'язку та спільної функціональності.

Універсальність React сяє, коли справа доходить до створення додатків на замовлення, пристосованих до конкретних бізнес-потреб. Зокрема, його компонентна архітектура дозволяє без проблем збирати складні структури в додатки.

Перспективність React є однією з найвагоміших переваг, які він пропонує розробникам. Гнучка архітектура React задовольняє поточні потреби веб-розробки, а також легко адаптується до нових технологій, які визначатимуть розвиток індустрії в найближчому майбутньому.

Зокрема, машинне навчання робить значні кроки у веб-розробці, а світовий ринок машинного навчання вже оцінюється у 21 мільярд доларів у 2022 році, що підкреслює важливість перспективності React та його здатності гармонійно поєднуватися з такими досягненнями [12].

Одним з найяскравіших прикладів є TensorFlow.js, бібліотека ML, що використовується для розпізнавання зображень та шаблонів. Аналогічно, React дозволяє інтегрувати чат-ботів на основі ML і навіть функції рекомендацій. Крім

того, WebAssembly може бути корисною підмогою, дозволяючи ML-додаткам, написаним на Rust, Python або C++, існувати в нативному додатку.

В односторінкових додатках, також відомих як SPA, де декілька компонентів знаходяться на одній сторінці, управління станами та міжкомпонентною взаємодією може швидко стати складним завданням - саме тут Redux для React стає в нагоді.

Невід'ємна частина React, він слугує «менеджером» для забезпечення послідовного та точного потоку даних між компонентами, що централізує управління станами та сприяє незалежності компонентів, значно покращуючи стабільність даних та користувацький досвід.

Хоча React надає безліч переваг для розробників різного рівня кваліфікації, він не позбавлений відповідних недоліків, серед яких можна виділити наступні [12]:

- складні концепції та просунуті патерни;
- складність інтеграції з іншими технологіями;
- бар'єр для не-JavaScript розробників;
- не повноцінний фреймворк;
- роздуття коду;
- зниження продуктивності на застарілих пристроях та слабких мережах.

React впроваджує декілька складних концепцій та патернів, які спочатку можуть бути незрозумілими для початківців. Розуміння JSX, компонентів, пропсів, управління станами, методів життєвого циклу та хуків вимагає ґрунтовного знання основ JavaScript.

React часто використовується разом з іншими інструментами та технологіями - такими як Redux, React Router та різноманітним проміжним програмним забезпеченням - і розуміння того, як інтегрувати ці технології з React, може бути складним для новачків.

Значна залежність React від JavaScript може бути бар'єром для розробників, які не володіють JavaScript. Хоча JavaScript є універсальною та широко використовуваною мовою, розробникам з різним досвідом програмування може

бути складно адаптуватися до парадигм JavaScript та способу його використання в React.

React в першу чергу працює з частиною MVC «View», також відомою як архітектура Model View Controller. Для «Моделі» та «Контролера» потрібні додаткові бібліотеки, що в кінцевому підсумку може призвести до менш структурованого та потенційно більш хаотичного коду у порівнянні з повнофункціональними фреймворками, такими як Angular[12].

React.js, що характеризується значними вимогами до бібліотек та залежностей, сумнозвісний своїми роздутими додатками. Це часто проявляється у довшому часі завантаження, особливо у складних проектах. Структура фреймворку, що значною мірою покладається на віртуальну пам'ять DOM, вимагає завантаження цілих бібліотек навіть для незначних функцій, що значно збільшує цифровий слід програми та знижує її ефективність.

Продуктивність React.js-додатків має тенденцію до погіршення на старому обладнанні та в районах з поганим інтернет-зв'язком. Це в першу чергу пов'язано з клієнтською моделлю рендерингу фреймворку та інтенсивною обробкою JavaScript. Ці фактори можуть викликати затримки у рендерингу інтерактивних елементів, що особливо помітно на пристроях з обмеженими обчислювальними можливостями або в середовищах з обмеженою пропускною здатністю - і це негативно впливає на користувацький досвід.

Як і будь-який інший фреймворк, Angular має свої переваги та недоліки. Ось деякі з переваг його використання [13]:

– Angular є структурованим фреймворком, який сприяє чітко визначеній та організованій методології веб-розробки;

– двостороння архітектура зв'язування даних в Angular полегшує автоматичну синхронізацію між моделлю та дисплеєм;

– Angular Command Line Interface (CLI) - це комплексний інструмент, який спрощує процес ініціалізації, розробки та тестування проекту;

– Angular розробляється з використанням TypeScript, який є статично типізованим розширенням JavaScript.

Програмна система дотримується архітектурного шаблону Model-View-Controller (MVC), який виступає за чіткий розподіл обов'язків. Це полегшує розробникам можливість ефективно структурувати та підтримувати свій код, що призводить до створення більш впорядкованих та адаптованих програм.

Полегшена автоматична синхронізація між моделлю та дисплеєм означає, що модифікації, внесені в модель, негайно відображаються у вікні, і навпаки, зводить до мінімуму необхідність людських маніпуляцій з об'єктною моделлю документа (DOM).

Angular Command Line Interface (CLI) створює стандартизований шаблон коду та пропонує ряд інструкцій для створення компонентів, модулів, сервісів та інших елементів, що підвищує ефективність роботи розробників та зменшує їхнє робоче навантаження.

TypeScript полегшує виявлення помилок протягом усього процесу розробки та пропонує розширену підтримку інструментів, а отже, спрощує обслуговування та масштабування проектів.

Однак є й деякі мінуси, які слід враховувати при використанні Angular [13]:

- крива навчання;
- менше можливостей для SEO;
- міграція.

Фреймворк складається з декількох складних тем, які є фундаментальними для розробки. Синтаксис також складний у порівнянні з іншими фреймворками. Отже, Angular має круту криву навчання.

Індексація Angular-додатків у пошукових системах є складною, оскільки Angular має дуже обмежені можливості з точки зору SEO.

Міграція вашого існуючого додатку з іншого технологічного фреймворку на Angular є складним завданням. Навіть оновлення зі старої версії до останньої версії є складним при використанні Angular.

Переходимо до Vue.js. Він має кілька переваг, які роблять його цінним доповненням до вашого технологічного стеку [14]:

- простота у використанні;

– чудові офіційні бібліотеки;

– швидка візуалізація;

– простий у вивченні.

Vue.js є поступово адаптованим, що дозволяє розробникам починати з основ і ускладнювати за потреби. Ядро бібліотеки базується на фундаментальних технологіях веб-розробки - CSS, HTML та JavaScript.

Vue.js надає офіційні бібліотеки, такі як Vue Router та Vuex, для маршрутизації та управління станом, що спрощує загальні проблеми розробки.

Легкий розмір пакунка Vue.js (лише 21 КБ) та його віртуальна DOM роблять його швидшим за багатьох конкурентів. Віртуальна DOM ефективно синхронізує зміни, забезпечуючи оптимальну продуктивність.

Vue.js вимагає мінімальних базових знань про бібліотеки та варіації JavaScript, що робить його доступним для розробників.

Мінуси, які можна виділити в цьому доволі сучасному фреймворці [14]:

– занадто гнучкий;

– занадто обмежений;

– занадто новий.

Як не дивно, найпоширеніша скарга на Vue.js - це те, що він занадто гнучкий, або, принаймні гнучкіший, ніж потрібно.

Оскільки Vue.js дає розробникам можливість почати все з нуля, безпосереднім плюсом є більша гнучкість у впровадженні нових функцій.

Але, з іншого боку, це є недоліком, що все може ускладнитися, оскільки помилки та невідповідності починають розвиватися з великими проектами.

Незважаючи на те, що екосистема Vue.js досить широка, включаючи офіційні бібліотеки та спільноти, при порівнянні Vue.js з React, або з AngularJS. Можна помітити, що Vue.js просто не має такої кількості плагінів і компонентів, як інші фреймворки подібного роду. Без сумніву, це є головним недоліком Vue.js.

З багатьох причин сучасність Vue.js не є такою вигідною, як це може здатися.

Vue.js також був розроблений китайськими авторами, тому багато пов'язаних з ним проектів важко зрозуміти англомовним користувачам. І пройде якийсь час,

перш ніж знайдеться достатньо небайдужих людей, у яких буде змога долучитися до повного перекладу.

Всю порівняльну характеристику я висвітлив у таблиці нижче (див. табл. 1.1).

Таблиця 1.1 – Порівняння фреймворків між собою

Характеристика	Назва фреймворків		
	Фреймворк React	Фреймворк Angular	Фреймворк Vue
Мова програмування	JavaScript (JSX).	TypeScript.	JavaScript (ES6+), TypeScript.
Основна концепція	Компонентний підхід, віртуальний DOM.	MVC (Model-View-Controller), двосторонній зв'язок.	Компонентний підхід, реактивність.
Складність вивчення	Середня.	Висока.	Низька.
Управління станом	Redux, Context API.	Вбудоване рішення (ngRx).	Vuex.
Шаблони	JSX.	Angular Templates.	HTML з директивами Vue.
Маршрутизація	React Router.	Вбудований (Angular Router).	Vue Router.
Підтримка мобільних додатків	React Native.	NativeScript, Ionic.	Weex
Підтримка серверного рендерингу	Next.js.	Angular Universal.	Nuxt.js.
Інструменти розробки	Create React App, Next.js.	Angular CLI.	Vue CLI.

Фреймворк Angular використовується досвідченими розробниками для створення додатків з розгалуженою структурою, і цей фреймворк найкраще працює, коли мова йде про створення прогресивних веб-додатків та гібридних мобільних додатків[15].

React-розробники використовують React для створення динамічних веб-сайтів з бездоганною функціональністю. Він також допомагає у створенні нативних мобільних додатків (з використанням фреймворку react native)[15].

Vue поєднує в собі найкращі риси Angular та React. Він допомагає створювати односторінкові додатки [15].

Це доводить, що ці три JavaScript фреймворки творять магію при створенні користувацьких інтерфейсів і пропонують найкращі рішення для надійної веб-розробки. Але який з них обрати для вашого наступного проекту, залежить від середовища розробки програмного забезпечення, гнучкості команди та типу проекту, над яким вона працює.

РОЗДІЛ 2. РОЗРОБКА, ТЕСТУВАННЯ ТА РОЗГОРТАННЯ НА СЕРВЕРІ REST API

2.1 Вибір технологій для серверної частини

Вибір технологій для бекенду веб-розробки має вирішальне значення для її загальної продуктивності, масштабованості та обслуговування. У цьому розділі ми зосередимося на чотирьох відомих технологіях бекенда - Node.js, Express.js, PM2 та PostgreSQL.

Node JS - це серверна платформа, яка дозволяє розробникам створювати високопродуктивні додатки за допомогою JavaScript. Node JS використовує модель вводу/виводу, керовану подіями, що не блокується, що робить її ідеальною для створення швидких і масштабованих додатків. За допомогою Node JS розробники можуть створювати серверні додатки, веб-розробки і навіть мобільні додатки, використовуючи такі фреймворки, як Express, Koa та Sails [16].

Однією з найбільших переваг використання Node JS для розробки корпоративних додатків є його масштабованість. Node JS призначений для обробки великої кількості одночасних з'єднань з мінімальними накладними витратами. Це робить його ідеальним для створення додатків, які повинні обробляти великий обсяг трафіку (див. рис. 2.1).

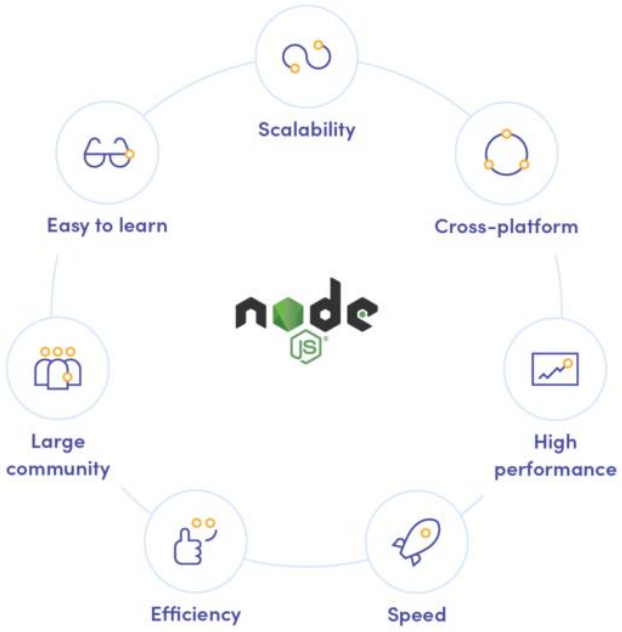


Рисунок 2.1 - Переваги використання Node JS

Node JS також відомий своєю швидкістю. Він використовує JavaScript-рушій V8, який є тим самим рушієм, що використовується в Google Chrome. Це означає, що Node JS може виконувати JavaScript-код швидше, ніж інші платформи. Крім того, Node JS використовує модель вводу/виводу, керовану подіями, що не блокує, що дозволяє йому швидко обробляти запити [16].

Ця швидкість особливо важлива для корпоративних додатків, де кожна секунда на рахунку. Використовуючи Node JS, розробники можуть створювати додатки, які швидко реагують на запити, забезпечуючи кращий користувацький досвід.

Ефективність - це ще одна причина, чому Node JS є чудовим вибором для вашого корпоративного проекту. Node JS дозволяє розробникам писати як клієнтський, так і серверний код на JavaScript, а це означає, що ви можете повторно використовувати код в обох частинах вашого додатку. Це може допомогти скоротити час розробки і зробити вашу кодову базу більш зручною для підтримки.

Крім того, Node JS має велику екосистему модулів і бібліотек, які можуть допомогти прискорити час розробки. Ці модулі та бібліотеки можна легко встановити за допомогою npm, менеджера пакетів Node JS.

Node JS також є кросплатформенним, що означає, що ви можете запускати

його на будь-якій операційній системі. Це робить його ідеальним для створення додатків, які повинні працювати на декількох платформах. Node JS також сумісний з багатьма іншими технологіями, що полегшує інтеграцію з іншими системами. Дізнайтеся більше про переваги кросплатформних додатків тут [16].

Node JS побудований на основі JavaScript-рушія V8, який відомий своєю високою продуктивністю. Це дозволяє Node JS легко обробляти велику кількість паралельних з'єднань і запитів, що робить його ідеальним для створення високопродуктивних веб-додатків.

Express.js - це швидкий, гнучкий і мінімалістичний веб-фреймворк для Node.js. Це фактично інструмент, який спрощує створення веб-додатків та API за допомогою JavaScript на стороні сервера. Express має відкритий вихідний код, який розробляється та підтримується фондом Node.js [17].

Express.js пропонує потужний набір функцій, які підвищують вашу продуктивність і спрощують створення веб-додатків. Він полегшує організацію функціональності вашого додатку за допомогою проміжного програмного забезпечення та маршрутизації. Він додає корисні утиліти до HTTP-об'єктів Node і полегшує рендеринг динамічних HTTP-об'єктів.

«PM2» - це менеджер процесів, який допоможе керувати та підтримувати вашу програму онлайн 24/7 [18].

СУБД «PostgreSQL» як рішення для керування базами даних пропонує унікальне поєднання переваг, які задовольняють широкий спектр потреб в управлінні даними. Однією з головних причин вибору PostgreSQL є її виняткова підтримка розширених типів даних і складна функціональність. Сюди входить вбудована підтримка JSON, геометричних даних і користувацьких типів, що забезпечує гнучке та ефективне зберігання і пошук даних, які можуть адаптуватися до найскладніших і найрізноманітніших моделей даних. Здатність PostgreSQL легко обробляти складні запити, транзакції та об'ємні операції зберігання даних робить її ідеальним рішенням для додатків, які потребують детальної аналітики, обробки в режимі реального часу та високої цілісності даних [19].

Ще однією причиною використання PostgreSQL є її сильний акцент на

розширюваність і відповідність стандартам. Вона розроблена з можливістю налаштування, що дозволяє користувачам розширювати базу даних власними функціями, операторами та типами даних. У поєднанні з відкритим вихідним кодом, PostgreSQL може розвиватися разом з потребами вашого проекту, сприяючи інноваціям і гарантуючи, що ваша система баз даних залишається перспективною.

Вибір відповідних технологій для бекенда веб-додатків є життєво важливим рішенням, яке може мати значний вплив на продуктивність, масштабованість та ремонтопридатність веб-додатків. Модель вводу/виводу Node.js, що керується подіями та не блокується, забезпечує ефективну платформу для одночасної роботи з декількома паралельними з'єднаннями. Express.js пропонує додатковий рівень надійного фреймворку з мінімалістичними, але потужними функціями для швидкого та ефективного створення веб- та мобільних додатків. PM2, який дбає про безвідказну роботу додатка. PostgreSQL, багатофункціональна реляційна база даних, забезпечує цілісність, надійність та високу продуктивність даних.

Поєднання Node.js, Express.js, PM2 та PostgreSQL для ефективного задоволення цих вимог дозволяє розробникам створювати сучасні серверні рішення, пристосовані до сучасного динамічного веб-середовища.

2.2 Проектування та реалізація REST API

Перед тим як проектувати та реалізовувати REST API. Потрібно розібратися що це таке.

RESTful API - це архітектурний підхід до розробки інтерфейсу прикладного програмування, який використовує HTTP-запити для доступу до даних і їх використання (див. рис. 2.2) [20].

Команди RESTful API:

– «GET». Команда GET дозволяє отримати ресурс;

– «PUT». Дозволяє змінити стан або оновити ресурс, який може бути

об'єктом, файлом чи блоком;

– «POST». Дозволяє створити ресурс;

– «DELETE». Дозволяє видалити ресурс.

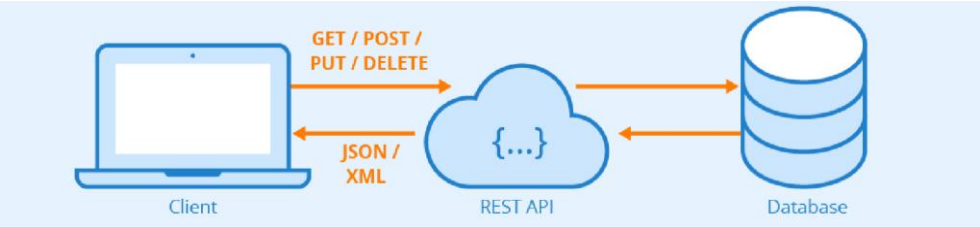


Рисунок 2.2 – Приклад роботи REST API

API - це код, який дозволяє двом програмам взаємодіяти одна з одною. Дизайн API визначає правильний спосіб для розробника написати програму або клієнта, який використовує API для запиту ресурсів від іншої програми або сервера. API стали життєво важливим механізмом для забезпечення комунікації між програмним забезпеченням [20].

RESTful API також називають RESTful веб-сервісами та REST API. Вони засновані на передачі стану представлення, архітектурному стилі та підході до комунікацій, який часто використовується при розробці веб-сервісів. Цей підхід також може полегшити комунікацію між іншими типами додатків.

Технології REST зазвичай надають перевагу перед іншими подібними технологіями. Це пов'язано з тим, що REST використовує менше ресурсів мережі, що робить її більш ефективною у використанні інтернету.

REST API буде розташований та реалізований на віртуальній машині VMware Workstation Pro, а в якості операційної системи буде виступати Ubuntu Server. Вибір VMware обумовлений його гарним дизайном та необхідним функціоналом, хоча він досить дорогий, що може бути досить суттєвою проблемою для деяких користувачів.

Мій вибір пав на Ubuntu Server. Оскільки, у порівнянні з іншими операційними системами, Ubuntu Server відзначається своєю стабільністю, безпекою та універсальністю. Вона сумісна з широким спектром апаратного та програмного забезпечення, що робить її ідеальним вибором як для бізнесу, так і для

приватних осіб. Незалежно від того, чи ви запускаєте невеликий веб-сайт, чи простеньке REST API, Ubuntu Server має інструменти та функції, необхідні для досягнення цієї мети (див. рис. 2.3) [21].

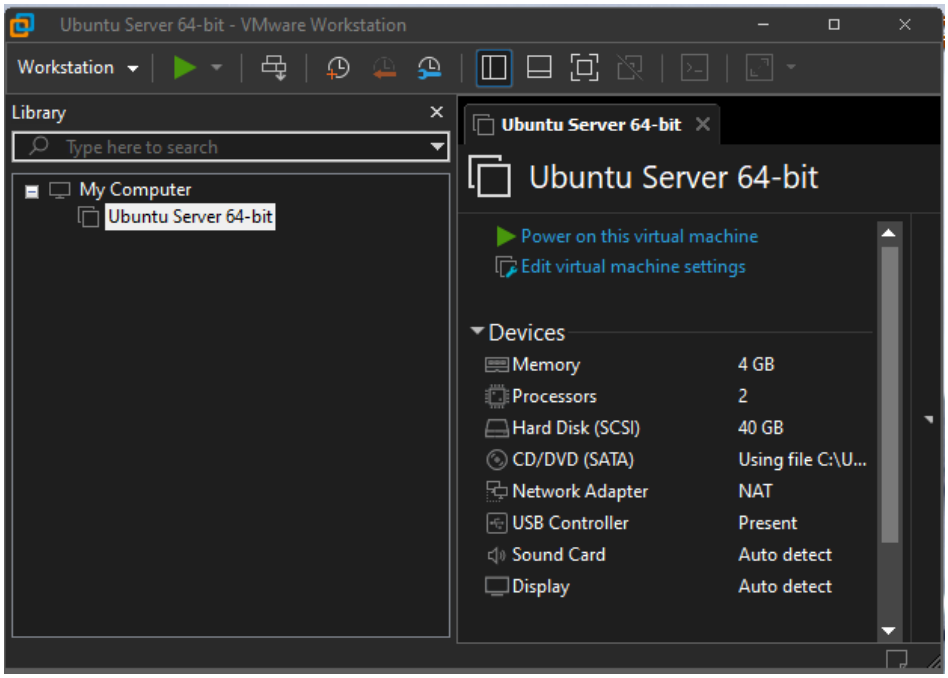


Рисунок 2.3 – Налаштована віртуальна машина з ОС'ою Ubuntu Server

В якості середовища виконання будемо використовувати поєднання Node.js, PostgreSQL та PM2, як фреймворк, який буде створювати сам сервер – Express.js. Тому спочатку встановлюємо Node.js, PM2 та базу даних PostgreSQL.

Після успішного встановлення та ініціалізації. Перед тим як реалізовувати саме REST API нам потрібно на віртуальній машині в брандмауері відкрити порт для нього. Оскільки, заздалегідь був обраний порт 8080, то потрібно його відкрити за допомогою команды «sudo ufw allow 8080/tcp», та за допомогою іншої команди «sudo ufw status», можна переглянути всі дозволені порти. Тепер веб-програми зможуть звертатися до REST API через цей порт (див рис. 2.4).

```
dialekt@crud-server:~$ sudo ufw status
[sudo] password for dialekt:
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
8080/tcp (v6)              ALLOW       Anywhere (v6)
```

Рисунок 2.4 – Перевірка стану портів у брандмауері

Далі встановлюємо PostgreSQL на нашу віртуальну машину та налаштовуємо його. Створюємо базу даних під назвою «diploma_postgres». Потім створюємо таблицю «student» з полями:

– «id»;
– «name»;
– «surname»;
– «gender»;
– «birthdate»;
– «specialty»;
– «university».

Тож, щоб перевірити, що все працює як треба, підключаємся до бази даних та перевіряємо за допомогою двох SQL команд «\dt» та «SELECT * FROM student», те що таблиця створена та в ній є вказані раніше поля (див. рис. 2.5).

```
diploma_postgres=# \dt;
         List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | student | table | postgres
(1 row)

diploma_postgres=# select * from student;
 id | name |  surname   | gender | birthdate  | specialty | university
----+------+------------+--------+------------+-----------+------------
  1 | Vika | Prokopenko | female | 2002-03-12 | 125       | KPI
  2 | Vika | Prokopenko | female | 2002-03-12 | 125       | KPI
(2 rows)
```

Рисунок 2.5 – Перевірка створеної таблиці та її полів

База даних налаштована, можна переходити до організацію самого REST API.

Спочатку створюємо структуру нашого додатка (див. рис. 2.6).

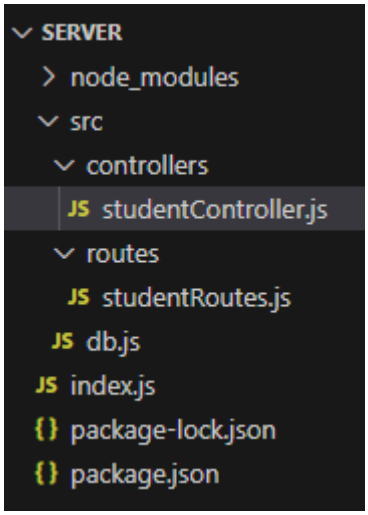


Рисунок 2.6 – Структура REST API

Ініціалізуємо наш додаток за допомого команди «npm init», ця команда створює необхідні файли та папки.

Наступний етап - встановлення необхідних залежностей:

– «Axios»;

– «cors»;

– «pg».

Ці залежності покращують процес розробки додатка. За допомогою них полегшується реалізація CRUD операцій та підключення до бази даних.

Далі створюємо головний файл «index.js» в ньому буде знаходитися тіло API.

В папці «src» будуть знаходитися:

– основна логіка додатка;

– налаштування для бази даних;

– маршрути.

Файл під назвою «studentController.js» вміщує в собі всю основну логіку додатка, а саме всі CRUD операції зі студентами:

– отримання даних про конкретного студента;

– отримання даних про всіх наявних студентів;

– додавання студента з введеними даними;

- оновлення даних конкретного студента;
- видалення даних конкретного студента;
- видалення даних усіх студентів.

При операціях з конкретним студентом необхідно вказувати в параметрах запиту його ідентифікатор (id). Наприклад, ми обрали метод передачі даних – GET, то ми повинні в посиланні вказати ідентифікатор існуючого студента, інакше сервер відправить помилка.

Інший файл з назвою «studentRoutes.js» імпортує всю логіку з переднього файлу та визначає увесь набір маршрутів для CRUD операцій.

Третій файл «db.js» налаштовує підключення до бази даних PostgreSQL. Тут визначається така конфігурація:

- «user». Ім'я користувача бази даних;
- «password». Пароль користувача бази даних;
- «host». Хост, де знаходиться база даних;
- «port». Порт, на якому працює PostgreSQL;
- «database». Ім'я бази даних.

І останній файл «index.js» в нього імпортується всі маршрути з логікою. Тут вказується порт на якому буде працювати сервер. І це є виконавчим файлом, який запуская додаток.

Останній етап це запуск та збереження в автозавантаження операційної системи нашого додатка за допомогою PM2. Перша команда «pm2 start index.js» ініціює виконання додатка. Друга «pm2 save» зберігає поточний стан процесів у спеціальний файл конфігурації «dump.pm2». І остання команда «pm2 startup systemd» налаштовує PM2 на автоматичний запуск при завантаженні системи. Вона генерує і виконує необхідні команди для інтеграції з системним менеджером запуску, в нашому випадку «systemd».

Отже, всі ці етапи дозволили успішно реалізувати додаток/сервер REST API. Використання Ubuntu Server в якості операційної системи забезпечило стабільність і безпеку середовища, а застосування таких технологій, як Node.js, Express.js, PostgreSQL та PM2, сприяло створенню ефективного, масштабованого і надійного

додатка.

2.3 Тестування REST API за допомогою програми Postman

Правильна робота та виявлення потенційних помилок мають вирішальне значення при тестуванні нашого REST API. Для виконання поставленого завдання, я буду використовувати Postman. Postman - це «швейцарський ніж» інструментів API, що дозволяє вам проектувати, створювати, тестувати, документувати і контролювати сервіси в одному місці.. Додаток «Postman» спрощує кожен крок життєвого циклу API та оптимізує співпрацю, щоб ви могли створювати кращі API швидше (див. рис. 2.7) [22].

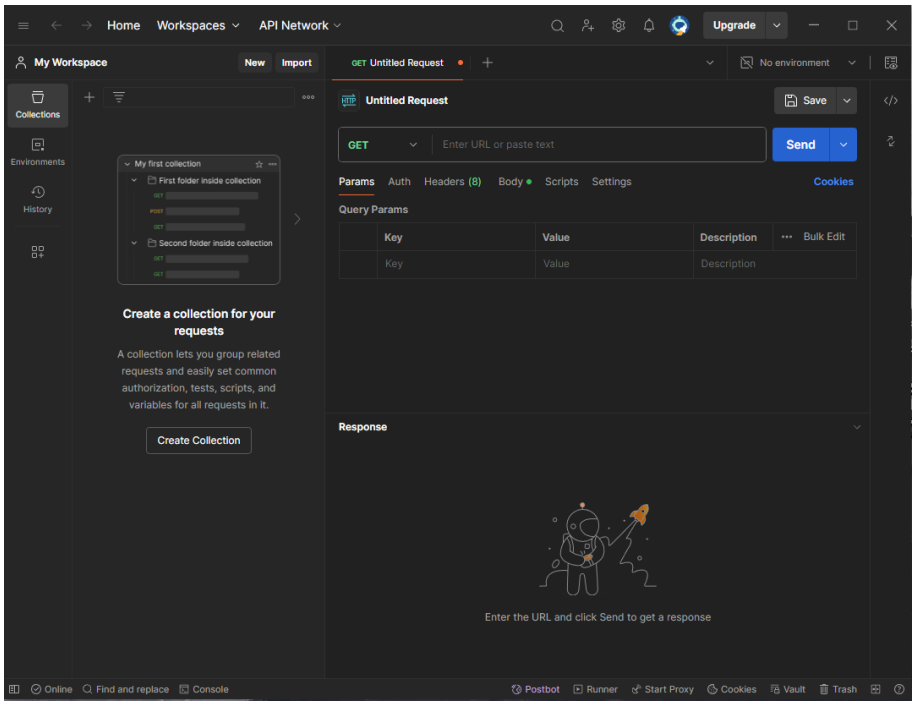


Рисунок 2.7 – Програма для тестування REST API – Postman

Перед тим як користуватися Postman. Потрібно визначити на якому порті та яка адреса була присвоєна нашому RESTful API додатку. Щоб визначити ip-адресу потрібно після запуску віртуальної машини подивитися в термінал, де буде написано, що за допомогою протоколу DHCP була привласнена ip-адреса:

«192.168.111.128». А порт ми визначили в конфігурації додатка, а саме: «8080».

Щоб перевірити, чи працює API правильно, будуть надсилатися різні HTTP-запити до додатка з різними методами.

Першим запитом буде отримання усіх існуючих студентів. Метод має назву «GET», та посилання буде виглядати ось так: «http://192.168.111.128:8080/api/students» (див. рис. 2.8).

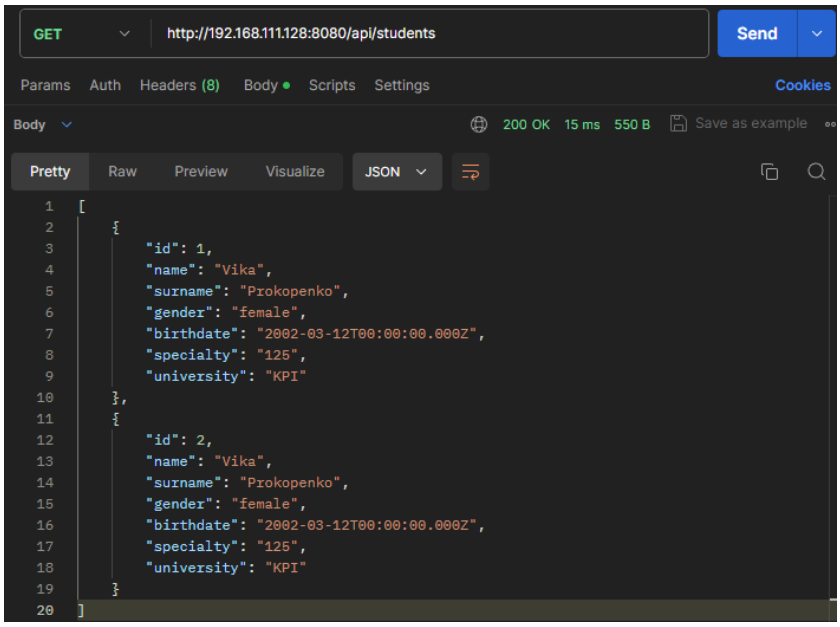


Рисунок 2.8 – Результат тестування першого метода: «GET»

Наступний буде схожий до першого, тут у відмінності від попереднього отримуються дані тільки про одного студента, запит має такий вигляд: «http://192.168.111.128:8080/api/student/1». Тут додаємо параметр «1», який вказує на конкретний ідентифікатор студента в базі даних (див. рис. 2.9).

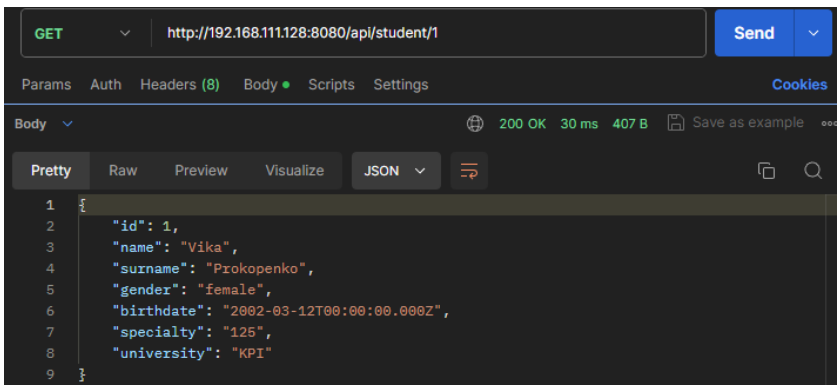


Рисунок 2.9 – Результат тестування наступного метода «GET»

Другий метод має назву «POST», запит має такий вигляд: «http://192.168.111.128:8080/api/student», цей метод дозволяє додати нового студента, для того щоб його додати, в тілі (body) запита потрібно вказати та заповнити поля, а саме:

– ім’я (name);
– прізвище (surname);
– стать (gender);
– дата народження (birthdate);
– спеціальність (specialty);
– університет (university).

Після надіслання запита, у відповідь отримуємо доданого студента з привласненим ідентифікатором (див. рис. 2.10).

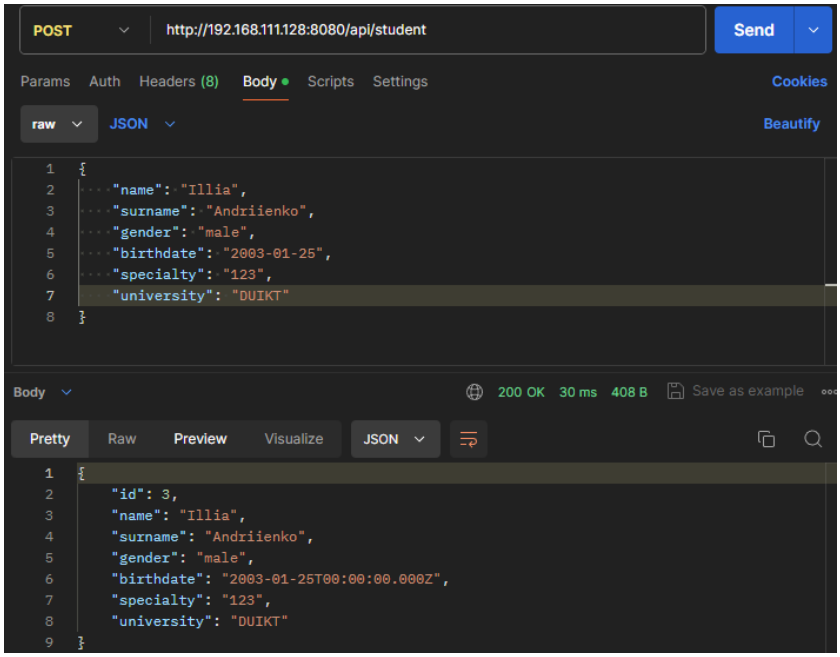


Рисунок 2.10 – Результат тестування метода «POST»

Третій метод «PUT», дозволяє нам оновити дані вказаного студента за його ідентифікатором, посилання таке: «http://192.168.111.128:8080/api/student/2». В цьому випадку, так само вказуємо та заповнюємо необхідні поля в тілі запита та вказуємо в параметрах ідентифікацію (id) студента, дані котрого ми хочемо оновити. Як результат отримуємо оновленні дані студента (див. рис. 2.11).

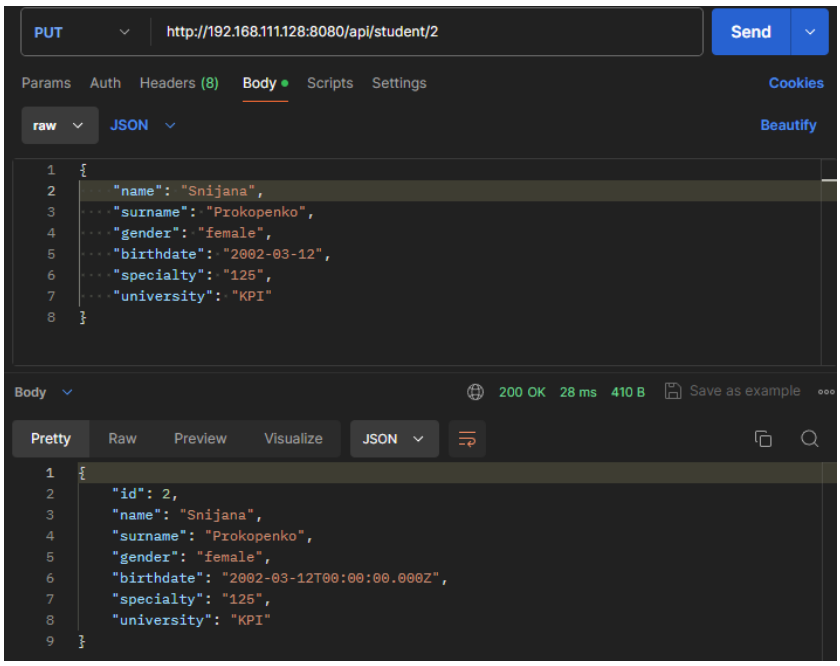


Рисунок 2.11 – Результат тестування метода «PUT»

І останній метод «DELETE», дає змогу, в нашому випадку, видалити або одного студента, або всіх разом.

Перший запит з таким виглядом: «http://192.168.111.128:8080/api/student/1», видаляє студента з вказаним ідентифікатором. У відповідь отримуємо повідомлення: «Студент з ідентифікатором 1 успішно видалений» (див. рис. 2.12).

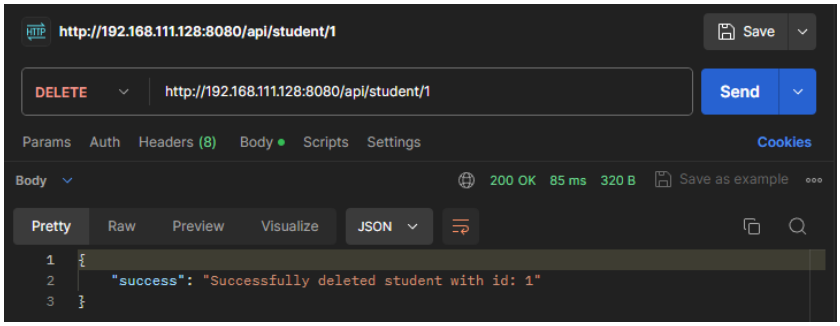


Рисунок 2.12 – Результат тестування метода «DELETE» з першим запитом

Другий запит з трохи іншим виглядом: «http://192.168.111.128:8080/api/students», видаляє усіх наявних студентів в базі даних. У відповідь отримуємо повідомлення: «Всі студенти успішно видалені» (див. рис. 2.13).

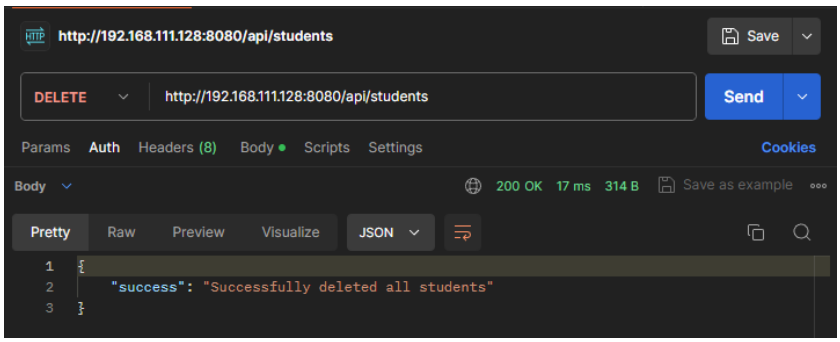


Рисунок 2.13 – Результат тестування метода «DELETE» з другим запитом

API-тестування стає все більш популярним. За допомогою API-тестування можна передбачити, як система відреагує на реального користувача, запустити одні й ті ж тести з різними наборами вхідних даних, виконати будь-які допоміжні дії для створення тестових даних і ситуацій, і таким чином прискорити тестування і зробити його більш якісним.

Таким чином, використання тестувального додатку Postman для тестування CRUD методів REST API забезпечує надійність, зручність та ефективність процесу тестування та налагодження створених додатків.

РОЗДІЛ 3. РОЗРОБКА CRUD ДОДАТКІВ З ВИКОРИСТАННЯМ REACT, ANGULAR ТА VUE

3.1 Налаштування робочого середовища

Перед тим як розробляти нові веб-застосунки, потрібно налаштувати робоче середовище. В цьому підрозділі, будуть налаштовані корневі папки для фреймворків та їх середовище виконання.

Організація React-проєкту з добре спланованою структурою папок дуже важлива для читабельності, масштабованості та супроводжуваності. Чітка структура допомагає ефективно керувати кодом, конфігураціями, модулями та іншими ресурсами [23].

Щоб створити свій перший React-додаток, потрібно відкрити термінал та ввести команду «npx create-react-app crud-react», де «crud-react» є назвою проекту. Після створення проекту були внесенні нові файли та папки. (див. рис. 3.1).

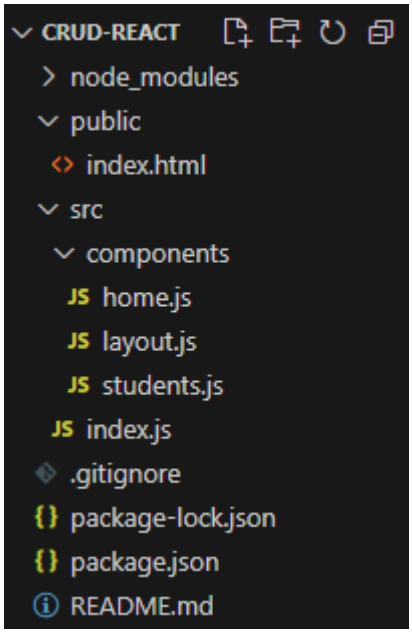


Рисунок 3.1 – Структура React-проєкту

В каталозі «src» знаходяться основні компоненти та виконавчі файли додатка. В каталозі «public» знаходиться статичний HTML-файл.

Та файл «package.json» містить інформацію про проєкт та додаткові пакети.

Після організації структури. Потрібно встановити та налаштувати додаткові пакети. В проєкті будуть використовуватися декілька допоміжних пакетів, а саме:

– «Axios». Цей пакет використовується для здійснення HTTP-запитів;

– «React-router-dom». Використовується для маршрутизації в React-застосунках;

– «Moment». Використовується для полегшення роботи з датами в JavaScript.

Щоб встановити ці всі залежності, використовуємо команду «npm install axios react-router-dom moment».

Створення добре організованої структури папок є важливим для підтримки React-проєкту. Це підвищує читабельність, ремонтопридатність та масштабованість додатку. Прийнявши добре організовану структуру, це дає можливість з легкістю керувати кодом.

Наступне середовище, яке буде організоване це Angular.

В принципі, гарна архітектура проєкту не покращить продуктивність додатку, не змусить його працювати швидше або краще. Однак за допомогою найкращих практик Angular-архітектури можна швидко перейти до вихідних файлів і зрозуміти, де все зберігається. Це допоможе полегшити налагодження і звести до мінімуму зусилля розробника або новачків, які блукають туди-сюди в пошуках потрібного файлу [24].

Для початку потрібно встановити Angular CLI. Angular CLI - це інструмент інтерфейсу командного рядка, який дозволяє створювати, розробляти, тестувати, розгортати та підтримувати Angular-додатки безпосередньо з командної оболонки [25].

Щоб встановити Angular CLI потрібно виконати команду «npm install -g @angular/cli». Після успішного встановлення, ініціалізуємо новий Angular-проєкт, виконавши команду «ng new crud-angular» (див. рис. 3.2).

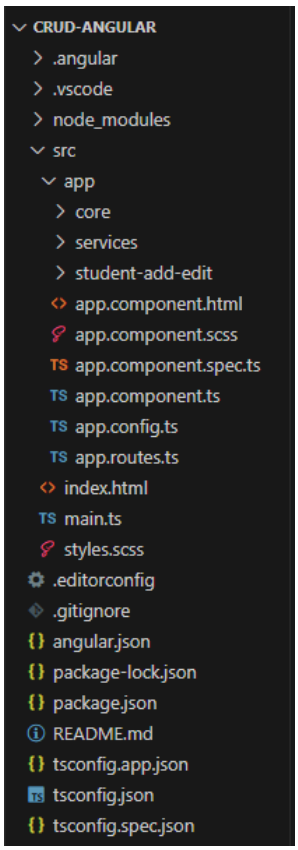


Рисунок 3.2 – Структура Angular-проєкту

Каталог «src», це місце, де розташовується весь вихідний код нашого додатку. Коли створюється Angular-додаток, за замовчуванням Angular CLI створює кілька файлів і каталогів у каталозі «src». Цей каталог також повинен містить кожен «Модуль», компонент, сервіс і пов'язаний з ним вихідний код [25].

Каталог «app» функціонує як коренева папка програми, яка слугує, як «app» модуль програми.

Angular не є винятком, тут також встановлюємо деякі додаткові пакети, а саме:

– «Angular Material». Angular Material використовується для створення інтерфейсів користувача з використанням «Material Design»;

– «Service http». Використовується для здійснення HTTP-запитів.

Отже, вибір відповідної структури проєкту, яка підходить для конкретного додатку, залежить від вимог проєкту. Обговорення архітектури Angular-проєкту та допоможуть створити масштабовані та підтримувані додатки, а також читабельність коду. Це допоможе новому розробнику швидко приєднатися до

команди і ввійти в курс справи. Крім того, додаток буде добре взаємодіяти з іншими Angular-залежностями та сторонніми компонентами.

І останнє робоче середовище, яке буде організовуватися – Vue.

Коли справа доходить до створення масштабованого проєкту, хочеться, щоб все в ньому було максимально зручним для обслуговування та розширення.

Насамперед, потрібно створити наш проєкт. За допомогою Vue CLI це можна легко реалізувати, за своєю функціональністю дуже схожий на Angular CLI. Тому командою «npm install -g @vue/cli», легко встановлюємо, та наступною командою «vue create crud-vue» швидко створюємо проект.

Далі необхідно встановити додаткові залежності. В цьому Vue-проєкті, я вирішив спробувати використовувати Bootstrap. Bootstrap - це безкоштовний фреймворк з відкритим вихідним кодом для створення веб-сайтів та веб-додатків. Розроблений для адаптивної розробки веб-сайтів, орієнтованих на мобільні пристрої, Bootstrap надає колекцію синтаксису для створення шаблонів [26].

Командою «npm install bootstrap moment axios vue-router» встановив все необхідне і можна переходити до організації структури.

Структура проєкту Vue складається з декількох ключових директорій та файлів (див. рис. 3.3).

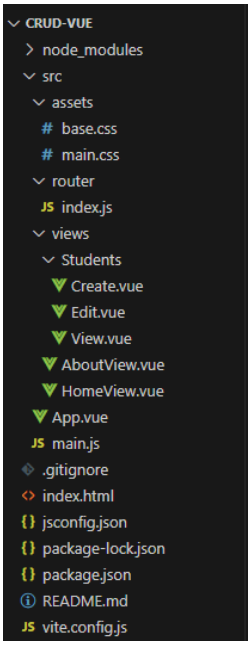


Рисунок – 3.3 Структура Vue-проєкту

Головною директорією проєкту виступає «src». Тут містяться всі основні файли додатка. Ця директорія складається з:

– каталога «assets». В нашому випадку цей каталог вміщує в собі 2 файла, які містять базові та специфічні стилі для додатка;

– каталога «router». Містить файл, що налаштовує маршрутизацію для додатка;

– каталога «views». Ця директорія містить компоненти, відповідальні за різні вигляди для сторінок додатка;

– файла «App.vue». Він є головним компонентом додатка, який включає в себе шаблони, «router-view /» та інші важливі елементи;

– файла «main.js». Вхідна точка додатка. Цей файл ініціалізує додаток Vue, імпортує головний компонент «App.vue», налаштовує маршрутизатор та монтує додаток до HTML елемента з ідентифікатором «#app».

Така організація Vue-проєкта є типовою структурою, створеною за допомогою Vue CLI. Вона забезпечує організацію коду, розподіляючи різні аспекти додатка по відповідним каталогам і файлам. Це допомагає зберігати код чистим і легким для підтримки та розширення.

3.2 Реалізація CRUD операцій

Першим середовищем в якому буде реалізація CRUD операцій – React. Перед тим як ці операції реалізувати потрібно створити React-компоненти в яких вони будуть діяти.

Тому створюємо 3 React-компонента [28]:

– компонент «Students» - відповідає за відображення списку студентів та форми для додавання/редагування студента;

– компонент «StudentList» — відповідає за відображання списка студентів та надає можливість видалення і редагування записів за допомоги

додаткових кнопок;

– компонент «StudentForm» — відповідає за форму для створення нового або редагування існуючого студента.

Першою реалізованою операцією буде «Read» (читання даних), для цього була створена функція «getStudents» у компоненті «StudentList».

Функція «getStudents» використовується для отримання списку всіх студентів з бази даних. Спочатку виконується запит до REST API за URL «http://192.168.111.128:8080/api/students» з методом «GET». Потім, отримані дані записуються в стан «students» за допомогою «setStudents» (див. рис 3.4).

```
const getStudents = useCallback(async () => {
  const requestOptions = {
    url: "http://192.168.111.128:8080/api/students",
    method: "get",
    responseType: "json",
  };

  try {
    const response = await axios(requestOptions);
    setStudents(response.data);
  } catch (error) {
    console.error(error);
  }
}, []);
```

Рисунок 3.4 – Реалізація функції «getStudents»

Операція «Create», реалізована через функцію «handleSubmit» у компоненті «StudentForm». Її ціллю є створення нового студента. Якщо властивості «props.student.id» не існує, то формується запит для створення нового студента. Запит виконується до REST API за посилання «http://192.168.111.128:8080/api/student» з методом «POST». Тілом запиту є дані з форми (див. рис. 3.5).

```
if (!props.student.id) {
  const requestOptions = {
    url: `http://192.168.111.128:8080/api/student`,
    method: "post",
    headers: { "Content-Type": "application/json" },
    responseType: "json",
    data: student,
  };

  try {
    await axios(requestOptions);
    props.showList();
  } catch (error) {
    console.error(error);
  }
```

Рисунок 3.5 – Перша частина реалізація функції «handleSubmit»

Операція «Update» для оновлення даних студентів, так само реалізована через функцію «handleSubmit» у тому ж самому компоненті. Мета її оновити вже існуючого студента. Реалізація є така, якщо властивість «props.student.id» існує, то формується новий запит для оновлення студента. Запит формується з посилання «http://192.168.111.128:8080/api/student/${props.student.id}», де властивість «props.student.id» вказує на ідентифікатор обраного студента, обраний метод «PUT». Так само, тіло запиту є дані з форми.

```
if (props.student.id) {
  const requestOptions = {
    url: `http://192.168.111.128:8080/api/student/${props.student.id}`,
    method: "put",
    headers: { "Content-Type": "application/json" },
    responseType: "json",
    data: student,
  };

  try {
    await axios(requestOptions);
    props.showList();
  } catch (error) {
    console.error(error);
  }
```

Рисунок 3.6 – Друга частина реалізація функції «handleSubmit»

Остання операція «Delete», яка призначена для видалення студента, реалізована за допомогою функції «deleteStudent» у компоненті «StudentList». Ця функція формує запит до API за посиланням

«http://192.168.111.128:8080/api/student/${id}», ідентифікатором студента є «id», з методом «DELETE». Після успішного видалення, викликається функція «getStudents» та оновлюється компонент.

```
async function deleteStudent(id) {
  const requestOptions = {
    url: `http://192.168.111.128:8080/api/student/${id}`,
    method: "delete",
    responseType: "json",
  };

  try {
    await axios(requestOptions);
    getStudents();
  } catch (error) {
    console.error(error);
  }
}
```

Рисунок 3.7 – Реалізація функції «deleteStudent»

Таким чином, цей код реалізує повний набір CRUD операцій для управління даними про студентів, забезпечуючи створення, читання, оновлення та видалення записів через взаємодію з REST API.

Наступним середовищем буде слугувати Angular.

Спочатку, створили два компонента (AppComponent та StudentAddEditComponent) та один сервіс-файл (StudentService). Сервіс-файл відповідає за надсилання до Rest API усіх потрібних запитів [29].

Першої операцією в середовищі Angular буде «Read». У компоненті «AppComponent» викликаємо метод «getStudentsList» з відповідним методом сервісу (метод «GET»). Отримані дані вставляються в «MatTableDataSource» для відображення в таблиці (див. рис. 3.8).

```
getStudentsList() {
  this._studentService.getStudentsList().subscribe({
    next: (res) => {
      this.dataSource = new MatTableDataSource(res);
      this.dataSource.sort = this.sort;
      this.dataSource.paginator = this.paginator;
    },
    error: console.error,
  });
}
```

Рисунок 3.8 - Реалізація метода «getStudentsList» в «AppComponent»

Наступною операцією буде «Create». Виклик відбувається у компоненті «StudentAddEditComponent». Коли відкривається форма, перевіряється наявність в ній даних, якщо їх немає то викликається метод «addStudent» з сервіс-файлу. В результаті вдалого створення студента, висвічується повідомлення та закривається модальне вікно (див. рис. 3.9).

```
else {
this._studentService.addStudent(this.studentForm.value).subscribe({
  next: (val: any) => {
    this._coreService.openSnackBar('Student added successfully');
    this._dialogRef.close(true);
  },
  error: console.error,
});
```

Рисунок 3.9 - Реалізація метода «addStudent» в «StudentAddEditComponent»

Третьою операцією є «Update». Операція схожа до попередньої, так само відкривається форма, але якщо дані в формі є, то викливається метод «updStudent». І як результат, оновлюються дані студента, також висвічується повідомлення і врешті-решт закривається модальне вікно (див. рис. 3.10).

```
if (this.data) {
  this._studentService
    .updStudent(this.data.id, this.studentForm.value)
    .subscribe({
      next: (val: any) => {
        this._coreService.openSnackBar(
          'Student info updated successfully'
        );
        this._dialogRef.close(true);
      },
      error: console.error,
    });
```

Рисунок 3.10 – Реалізація метода «updStudent» в «StudentAddEditComponent»

І заключною операцією в середовищі Angular є «Delete». В результаті виклику метода «deleteStudent» в компоненті «AppComponent». Успішно видаляється обраний студент, знову ж таки висвічується повідомлення та оновлюється таблиця зі списком студентів (див. рис. 3.11).

```
deleteStudent(id: number) {
  this._studentService.deleteStudent(id).subscribe({
    next: (res) => {
      this._coreService.openSnackBar(`Student deleted with id ${id}`, 'done')
      this.getStudentsList();
    },
    error: console.error,
  });
}
```

Рисунок 3.11 – Реалізація метода «deleteStudent» в «AppComponent»

Отже, ці методи в компонентах, написані в середовищі Angular, реалізують CRUD-операції для керування даними студентів, забезпечуючи можливість створення, читання, оновлення та видалення записів через взаємодію з RESTful API.

Останнім середовищем, де будуть реалізовані CRUD операції, буде сучасний веб-фреймворк Vue.

Спочатку як і в попередніх фреймворках створюємо 3 нових компонента [30]:

– «students». Використовується для відображення списку студентів;

– «studentCreate». Використовується для створення нового запису студента;

– «studentEdit». Використовується для оновлення даних існуючого студента.

Щоб реалізувати першу CRUD операцію, а саме створення (Create) нового студента. В компоненті «studentCreate» викликаємо метод «saveStudent», який створює запит до сервера, на основі введених в формі даних, які прив'язані до модуля «student», та створює новий запис в базі даних (див. рис. 3.12).

```
async saveStudent() {
  const requestOptions = {
    url: "http://192.168.111.128:8080/api/student",
    method: "post",
    headers: { "Content-Type": "application/json" },
    responseType: "json",
    data: this.model.student,
  };

  try {
    await axios(requestOptions);

    this.model.student = {
      name: "",
      surname: "",
      birthdate: "",
      gender: "",
      specialty: "",
      university: "",
    };
  } catch (error) {
    console.error(error);
  }
},
```

Рисунок 3.12 – Реалізація метода «saveStudent» в «studentCreate»

Для реалізації «Read» операції, будем використовувати компонент «students» та метод «getStudents» ініціалізований у ньому. Так само надсилається HTTP-запит до API з методом «GET», і як результат отримуємо список всіх існуючих в базі студентів. Після цього, отримані дані відобразяться у таблиці (див. рис. 3.13).

```
async getStudents() {
  const requestOptions = {
    url: "http://192.168.111.128:8080/api/students",
    method: "get",
    responseType: "json",
  };

  try {
    const res = await axios(requestOptions);

    this.students = res.data;

    this.students.forEach((student) => {
      student.birthdate = moment(student.birthdate).format("DD.MM.YYYY");
    });
  } catch (error) {
    console.error(error);
  }
},
```

Рисунок 3.13 – Реалізація метода «getStudents» в «students»

Реалізація операції «Update» в Vue, дещо відрізняється від інших фреймворків. Всі наступні методи цієї операції реалізовані в компоненті «studentEdit».

Метод «getStudentData» слугує для отримання даних студента, з використанням метода «GET», і вони вносяться у форму.

```
async getStudentData(studentId) {
  const requestOptions = {
    url: `http://192.168.111.128:8080/api/student/${studentId}`,
    method: "get",
    responseType: "json",
  };

  try {
    const res = await axios(requestOptions);

    this.model.student = res.data;
    this.model.student.birthdate = moment(res.data.birthdate).format(
      "YYYY-MM-DD"
    );
  } catch (error) {
    console.error(error);
  }
},
```

Рисунок 3.14 – Реалізація метода «getStudentData» в «studentEdit»

Потім, коли в формі змінилися всі потрібні поля, викликається інший метод «updateStudent» з методом «PUT». І вже він дає запит до бази даних та всі потрібні поля оновлюються (див. рис. 3.15).

```
async updateStudent() {
  const requestOptions = {
    url: `http://192.168.111.128:8080/api/student/${this.studentId}`,
    method: "put",
    headers: { "Content-Type": "application/json" },
    responseType: "json",
    data: this.model.student,
  };

  try {
    await axios(requestOptions);
  } catch (error) {
    console.error(error);
  }
},
```

Рисунок 3.14 – Реалізація метода «updateStudent» в «studentEdit»

І заключною операцією є - видалення (Delete). Ця операція виконується в компоненті «students», і виконується методом «deleteStudent». Метод нічим не відрізняються від реалізованих раніше в інших середовищах. Він так само виконує «DELETE» запит до RESTful API та видаляє конкретний запис про обраного студента.

Таким чином, були реалізовані всі CRUD-операції в різних виконавчих середовищах. Хоча загальна мета цих операцій однакова, підходи до їх реалізації дещо відрізняються між різними веб-фреймворками. Кожен з них має свої унікальні особливості та переваги, що впливає на спосіб роботи з даними та інтеграцію з API.

ВИСНОВКИ

В результаті проведеного дослідження та практичної реалізації різноманітних аспектів веб-розробки з використанням сучасних фреймворків вдалося досягнути кількох важливих результатів. Ці результати сприяють глибшому усвідомленню переваг і недоліків кожного фреймворку та формують міцний фундамент для обґрунтованого вибору в майбутніх проектах.

Здійснено детальний аналіз історії та шляхів розвитку трьох популярних фреймворків - React, Angular та Vue. Ці фреймворки стали основою для сучасного веб-програмування завдяки своїм унікальним функціям та можливостям. Зокрема, було розглянуто динаміку розвитку кожного фреймворку в часі, показано, як вони адаптувалися до потреб ринку та побажань розробників.

Основні принципи та архітектура кожного фреймворку були ґрунтовно досліджені, що надало уявлення про їхню внутрішню структуру та принципи роботи. Було виявлено, що React з його компонентно-орієнтованим підходом і використанням віртуального DOM, Angular з його потужною модульною архітектурою і двостороннім зв'язуванням даних, а також Vue з його гнучкістю і доступністю у використанні, пропонують різні підходи до вирішення схожих завдань.

Було проведено оцінку переваг та недоліків кожного з фреймворків. Зокрема, React вирізняється високою продуктивністю та великою спільнотою, але вимагає більшої налаштованості. Angular пропонує комплексний набір інструментів «з коробки», але може бути складнішим для початківців. Vue відзначається легкістю та доступністю у вивченні, але має меншу спільноту порівняно з React та Angular.

Розглянуто вибір технологій для серверної частини. Було визначено, що такі технології, як Node.js та Express.js підходять для створення ефективного та масштабованого REST API, пропонуючи високу продуктивність та гнучкість.

Проектування та реалізація REST API були виконані з урахуванням кращих стандартів, що дозволило отримати комфортний та ефективний інтерфейс для

взаємодії з клієнтськими додатками. Тестування REST API за допомогою Postman допомогло виявити та виправити потенційні помилки на етапі розробки.

Реалізовано повний комплекс CRUD-операцій з використанням усіх трьох фреймворків. Було наглядно продемонстровано, як різні шляхи реалізації одних і тих самих функцій можуть впливати на простоту розробки та супроводу додатків у різних середовищах.

Після всіх проведених досліджень, як на мене, найкраще середовище розробки – React. Він привабив мене з кількох причин:

- JSX синтаксис;
- ефективне управління станом;
- використання хуків (hooks);
- зручний спосіб встановлення.

Ці фактори роблять розробку CRUD-операцій у React надзвичайно зручним та результативним досвідом.

Отже, реалізація CRUD-додатків з використанням React, Angular та Vue засвідчили, що кожен з цих фреймворків має свої унікальні переваги і може бути обраний, виходячи з особливостей конкретного проєкту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Herbert D. What is React.js? Uses, examples, & more. *HubSpot Blog | Marketing,Sales,* Agency, and Customer Success Content. URL: https://blog.hubspot.com/website/react-js (дата звернення: 06.04.2024).

2. Deshpande C. What is React? [easily explained]. Simplilearn.com. URL: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs (дата звернення: 06.04.2024).

3. Kanade V. What is AngularJS? Meaning, working, benefits and challenges. Spiceworks. URL: https://www.spiceworks.com/tech/devops/articles/what-is-angular-js/ (дата звернення: 06.04.2024).

4. niku123. Introduction to AngularJS - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/introduction-to-angularjs/ (дата звернення: 07.04.2024).

5. Pros and cons of Angular development framework. AltexSoft. URL: https://www.altexsoft.com/blog/the-good-and-the-bad-of-angular-development/ (дата звернення: 07.04.2024).

6. Angular vs AngularJS: Which Framework is Better for Enterprise Applications. Monarch Innovation Private Limited. URL: https://www.monarch-innovation.com/angular-vs-angularjs (дата звернення: 07.04.2024).

7. Kugell A. In-Depth look at VueJs: pros, cons, and real-life applications in 2024 - trio. Trio. URL: https://trio.dev/why-use-vue-js/ (дата звернення: 07.04.2024).

8. MadhushaPrasad. Vue.js history. Medium. URL: https://madushaprasad21.medium.com/vue-js-history-1a6b8567198f (дата звернення: 07.04.2024).

9. Buna S. All the fundamental React.js concepts, jammed into this one article. freeCodeCamp.org. URL: https://www.freecodecamp.org/news/all-the-fundamental-react-js-concepts-jammed-into-this-single-medium-article-c83f9b53eac2/ (дата звернення: 08.04.2024).

10. Dabhade R. Angular – basic to advance – every concept explained! – part 1. Medium. URL: https://medium.com/@Rushabh_/angular-basic-to-advance-every-concept-explained-part-1-e8f8692e04b3 (дата звернення: 08.04.2024).

11. Explaining the key concepts in Vue.js. Web Reference - Code Documentation, Tutorials, and Demos. URL: https://webreference.com/javascript/frameworks/vue/ (дата звернення: 09.04.2024).

12. T. Williams A. The pros and cons of using React today. The New Stack. URL: https://thenewstack.io/the-pros-and-cons-of-using-react-today/ (дата звернення: 09.04.2024).

13. Pros and cons of Angular development. Agiliway - Custom Software Development Company. URL: https://agiliway.com/pros-and-cons-of-angular-development/ (дата звернення 09.04.2024).

14. Johnson P. What is Vue.js? The pros and cons of Vue.js in 2023. Digital Marketing Agency in USA. URL: https://foreignerds.com/what-is-vue-js-the-pros-and-cons-of-vue-js-in-2023/ (дата звернення: 10.04.2024).

15. Shah V. Angular vs React vs Vue: which one to choose - TatvaSoft blog. TatvaSoft Blog. URL: https://www.tatvasoft.com/blog/angular-vs-react-vs-vue/ (дата звернення: 11.04.2024).

16. Miquido. Where & Why to use Node JS in 2023 - Miquido Blog. Miquido. URL: https://www.miquido.com/blog/why-use-node-js/ (дата звернення: 11.04.2024).

17. souravsharma098. Express.js tutorial - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/express-js/ (дата звернення: 11.04.2024).

18. PM2 - home. PM2 - Home. URL: https://pm2.keymetrics.io/ (дата звернення: 13.04.2024).

19. Scott P. What is PostgreSQL? Everything you need to know. Percona Database Performance Blog. URL: https://www.percona.com/blog/what-is-postgresql-used-for/ (дата звернення: 13.04.2024).

20. Андрій Денисенко. Вступ до REST API – RESTful вебсервіси. robot_dreams. URL: https://robotdreams.cc/uk/blog/466-vstup-do-rest-api-restful-vebservisi (дата звернення: 14.04.2024).

21. Ubuntu Server vs. other OS: why it's the best choice. Technical Tips and Guides. URL: https://mangohost.net/blog/ubuntu-vs-other-operating-systems-better-choice-for-a-server/ (дата звернення: 15.04.2024).

22. What is Postman? Postman API platform. Postman API Platform. URL: https://www.postman.com/product/what-is-postman/ (дата звернення: 16.04.2024).

23. souravsharma098. Folder structure for a React JS project. GeeksforGeeks. URL: https://www.geeksforgeeks.org/folder-structure-for-a-react-js-project/ (дата звернення: 17.04.2024).

24. Varahade G. Angular best practices: tips for project structure and organization | thinkitive. Software Development Company | Thinkitive. URL: https://www.thinkitive.com/blog/angular-best-practices-tips-for-project-structure-and-organization/ (дата звернення: 18.04.2024).

25. Angular. Angular CLI. URL: https://angular.dev/tools/cli (дата звернення: 19.04.2024).

26. Zola A. What is a Bootstrap and how does it work?. Techtarget. URL: https://www.techtarget.com/whatis/definition/bootstrap (дата звернення: 20.04.2024).

27. Nortey S. How to structure folders in your Vue application. Medium. URL: https://simeonnortey.medium.com/how-to-structure-folders-in-your-vue-application-ea3934d56380 (дата звернення: 25.04.2024).

28. Lawal K. Ngrok blog: react CRUD: an introductory guide. ngrok | Unified Application Delivery Platform for Developers. URL: https://ngrok.com/blog-post/react-crud-example (дата звернення: 26.04.2024).

29. Periyaiah S. How to build a CRUD app in Angular. Syncfusion. URL: https://www.syncfusion.com/blogs/post/build-a-crud-app-in-angular (дата звернення: 30.04.2024).

30. chintanonweb. Creating a simple CRUD application with Vue.js. DEV Community. URL: https://dev.to/chintanonweb/creating-a-simple-crud-application-with-vuejs-5apm (дата звернення: 30.04.2024).

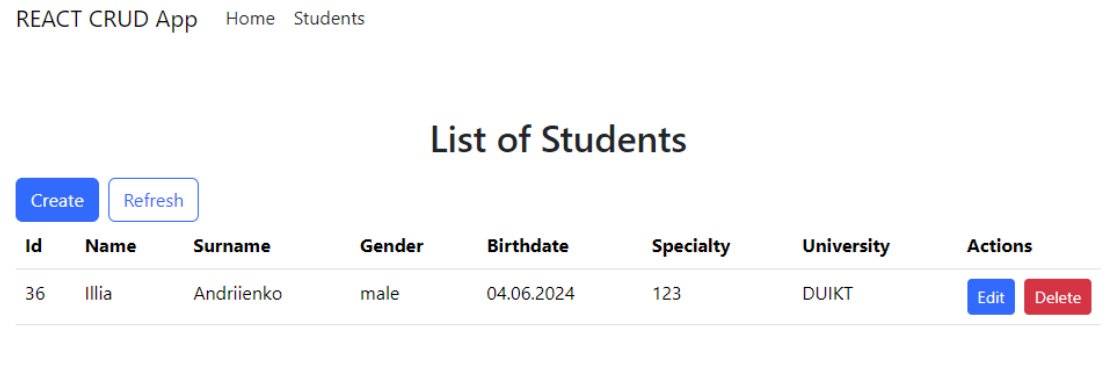
Додаток А

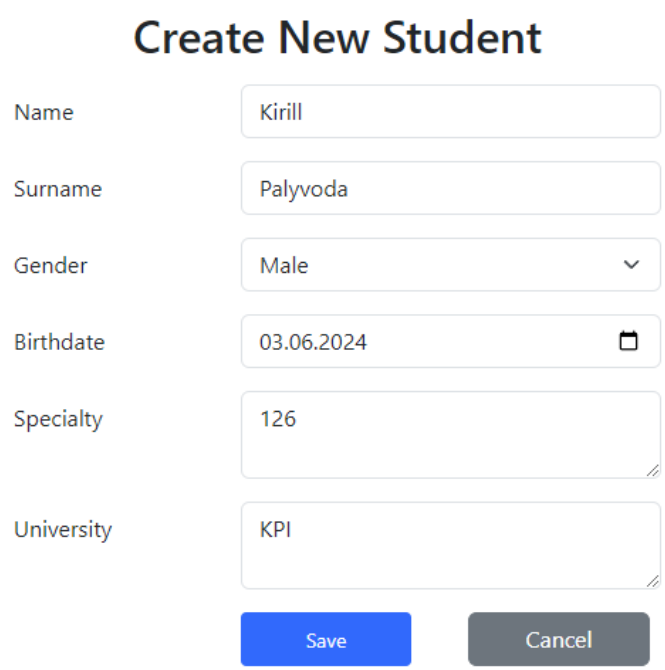
```
import express from “express”;
import cors from “cors”;
import studentRouter from “./src/routes/studentRoutes.js”;

const PORT = 8080;

const app = express();

app.use(express.json());
app.use(cors());
app.use(“/api,” studentRouter);

app.listen(PORT, () => {
        console.log(`Server started on port ${PORT}`)
});



import pg from “pg”;

const { Pool } = pg;

const dbConfig = {
        user: “postgres”,
        password: “postgres”,
        host: “localhost”,
        port: 5432,
        database: “diploma_postgres”,
};

export default new Pool(dbConfig);
```

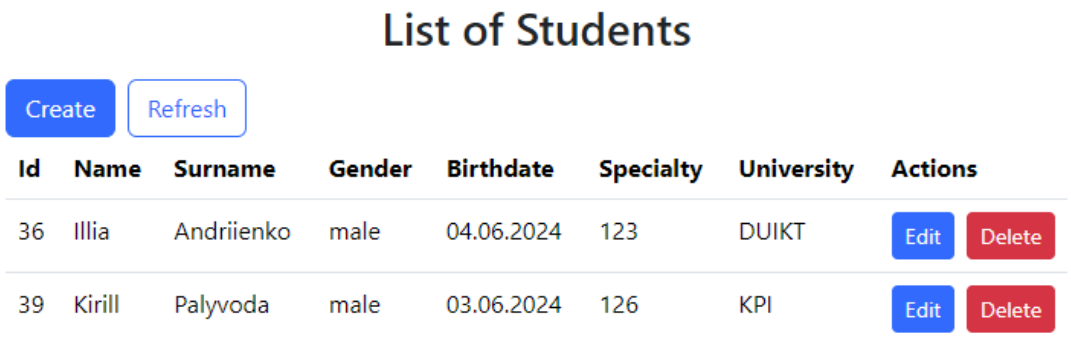
```
import { Router } from “express”;
import StudentController from “../controllers/studentController.js”;

const router = new Router();

router.get(“/student/:id”, StudentController.getStudent);
router.get(“/students”, StudentController.getStudents);
router.post(“/student”, StudentController.createStudent);
router.put(“/student/:id”, StudentController.updateStudent);
router.delete(“/student/:id”, StudentController.deleteOneStudent);
router.delete(“/students”, StudentController.deleteStudents);

export default router;
```

```
import db from “../db.js”;

class StudentController {
        async getStudent(req, res){ … }
        async getStudents (_, res){ … }
        async createStudent (req, res){ … }
        async updateStudent (req, res){ … }
        async deleteOneStudent (req, res){ … }
        async deleteStudents (_, res){ … }
}

export default new StudentController();
```

Додаток Б


REACT CRUD App
Home
Students
List of Students
Create
Refresh
Id
Name
Surname
Gender
Birthdate
Specialty
University
Actions
36
Illia
Andriienko
male
04.06.2024
123
DUIKT
Edit
Delete
© 2024 - REACT CRUD App


Create New Student
Name
Kirill
Surname
Palyvoda
Gender
Male
Birthdate
03.06.2024
Specialty
126
University
KPI
Save
Cancel


List of Students
Create
Refresh
Id
Name
Surname
Gender
Birthdate
Specialty
University
Actions
36
Illia
Andriienko
male
04.06.2024
123
DUIKT
Edit
Delete
39
Kirill
Palyvoda
male
03.06.2024
126
KPI
Edit
Delete

Edit student

Id 39

Name Tamila

Surname Palyvoda

Gender Female

Birthdate 03.06.2024

Specialty 126

University KPI

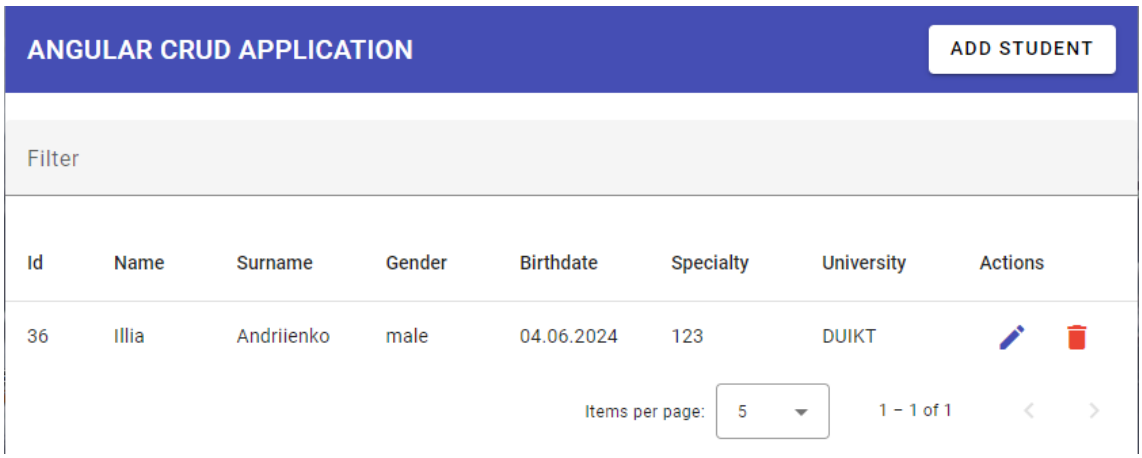
Save Cancel

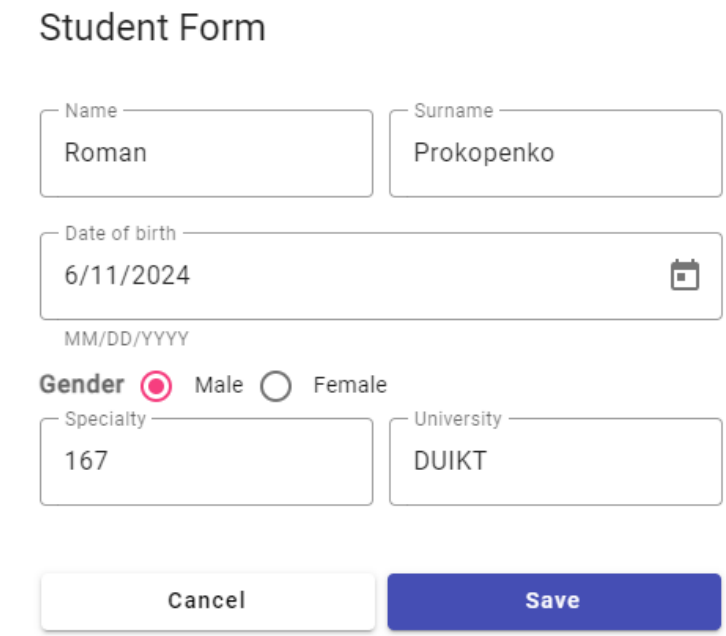
List of Students

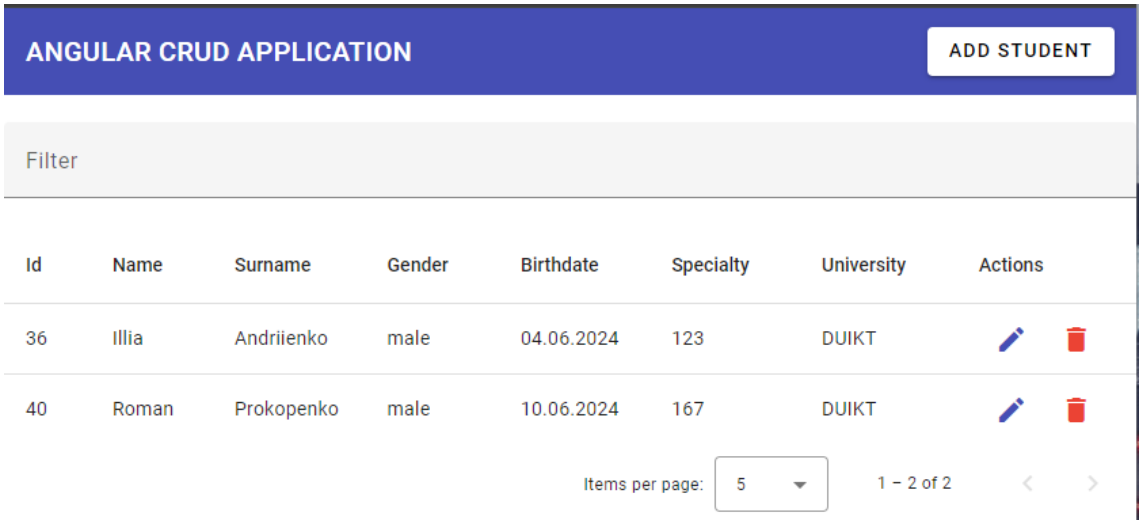
Create Refresh

Id	Name	Surname	Gender	Birthdate	Specialty	University	Actions
36	Illia	Andriienko	male	04.06.2024	123	DUIKT	Edit Delete
39	Tamila	Palyvoda	female	03.06.2024	126	KPI	Edit Delete

Додаток В


ANGULAR CRUD APPLICATION
ADD STUDENT
Filter
Id
Name
Surname
Gender
Birthdate
Specialty
University
Actions
36
Illia
Andriienko
male
04.06.2024
123
DUIKT
Items per page:
5
1 – 1 of 1


Student Form
Name
Roman
Surname
Prokopenko
Date of birth
6/11/2024
MM/DD/YYYY
Gender
Male
Female
Specialty
167
University
DUIKT
Cancel
Save


ANGULAR CRUD APPLICATION
ADD STUDENT
Filter
Id
Name
Surname
Gender
Birthdate
Specialty
University
Actions
36
Illia
Andriienko
male
04.06.2024
123
DUIKT
40
Roman
Prokopenko
male
10.06.2024
167
DUIKT
Items per page:
5
1 – 2 of 2

Student Form

Name: Olena

Surname: Prokopenko

Date of birth: 6/1/2024

MM/DD/YYYY

Gender: Male / Female

Specialty: 145

University: DUIKT

Cancel | Update

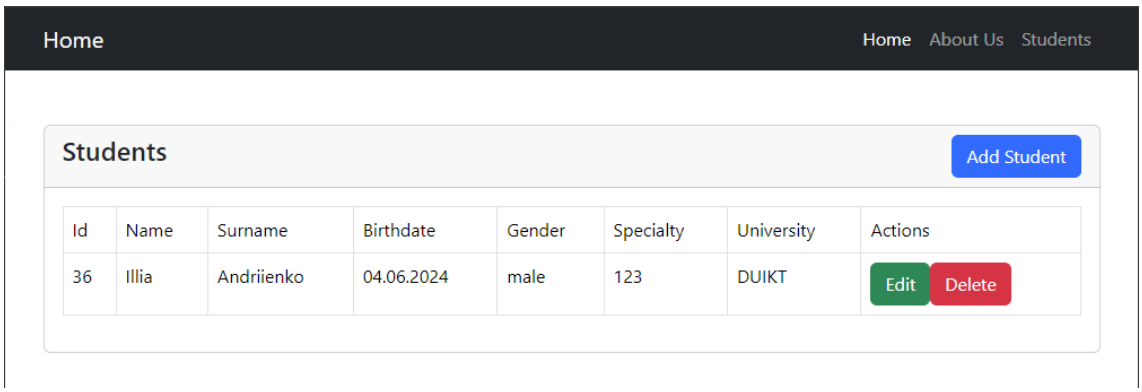
ANGULAR CRUD APPLICATION — ADD STUDENT

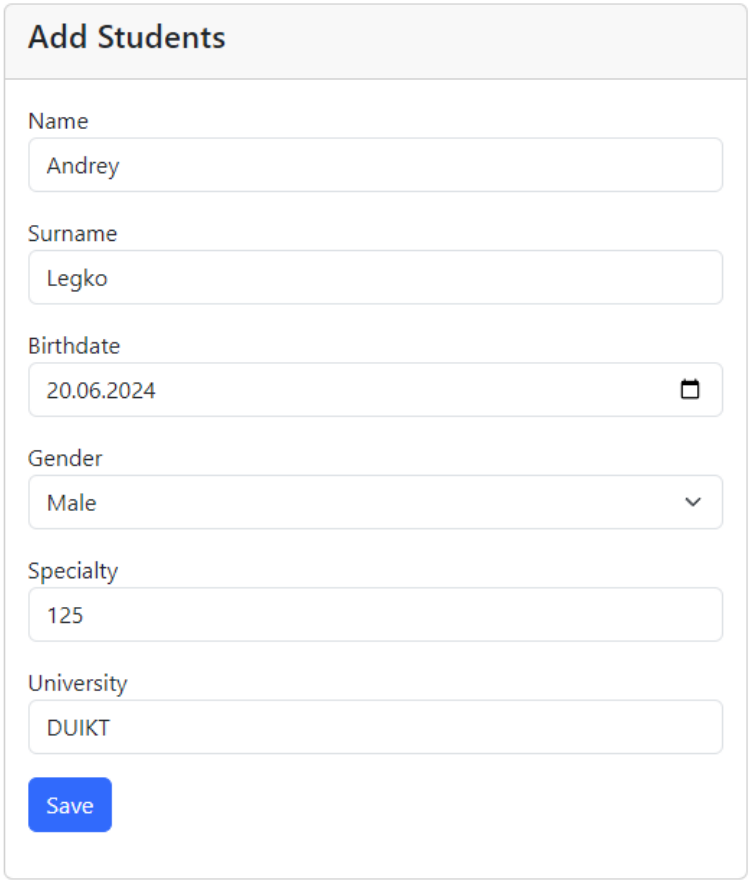
Filter

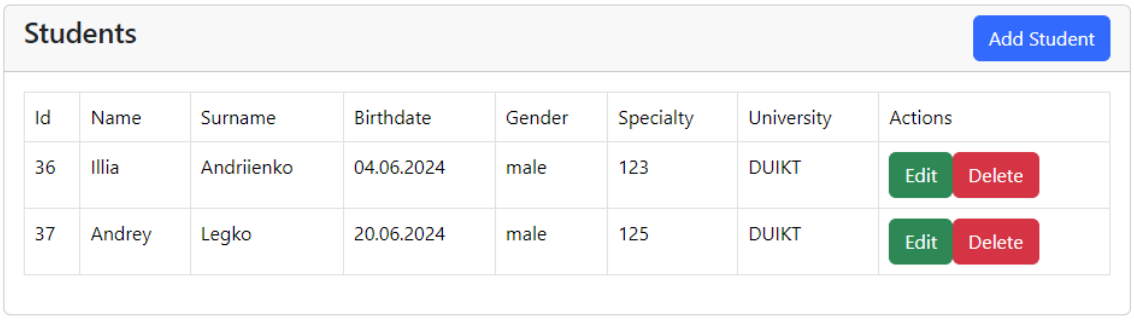
Id	Name	Surname	Gender	Birthdate	Specialty	University	Actions
36	Illia	Andriienko	male	04.06.2024	123	DUIKT	
40	Olena	Prokopenko	female	01.06.2024	145	DUIKT	

Items per page: 5 — 1 – 2 of 2

Додаток Г


Home
Home
About Us
Students
Students
Add Student
Id
Name
Surname
Birthdate
Gender
Specialty
University
Actions
36
Illia
Andriienko
04.06.2024
male
123
DUIKT
Edit
Delete


Add Students
Name
Andrey
Surname
Legko
Birthdate
20.06.2024
Gender
Male
Specialty
125
University
DUIKT
Save


Students
Add Student
Id
Name
Surname
Birthdate
Gender
Specialty
University
Actions
36
Illia
Andriienko
04.06.2024
male
123
DUIKT
Edit
Delete
37
Andrey
Legko
20.06.2024
male
125
DUIKT
Edit
Delete

Edit Students

Name

Irina

Surname

Legko

Birthdate

20.06.2024

Gender

Female

Specialty

125

University

DUIKT

Update

Students

Add Student

Id	Name	Surname	Birthdate	Gender	Specialty	University	Actions
36	Illia	Andriienko	04.06.2024	male	123	DUIKT	Edit Delete
37	Irina	Legko	20.06.2024	female	125	DUIKT	Edit Delete

Додаток Д

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій
Кафедра Комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

на тему: «Розробка веб-інтерфейсу з використанням сучасних фреймворків React, Angular та Vue.js»

Виконав студент групи КІД-41
Андрієнко Ілля Миколайович

Керівник кваліфікаційної роботи :
Кондратюк Борис Олександрович ,
асистент кафедри КІ

Київ – 2024

Мета роботи – дослідження та порівняння сучасних фреймворків для розробки веб-інтерфейсів, таких як React, Angular та Vue.js, а також розробка практичних рекомендацій щодо їх ефективного використання для створення інтерактивних та високопродуктивних веб-додатків.

Об'єкт дослідження – розробка веб-інтерфейсів з використанням сучасних фреймворків, зокрема React, Angular та Vue.js.

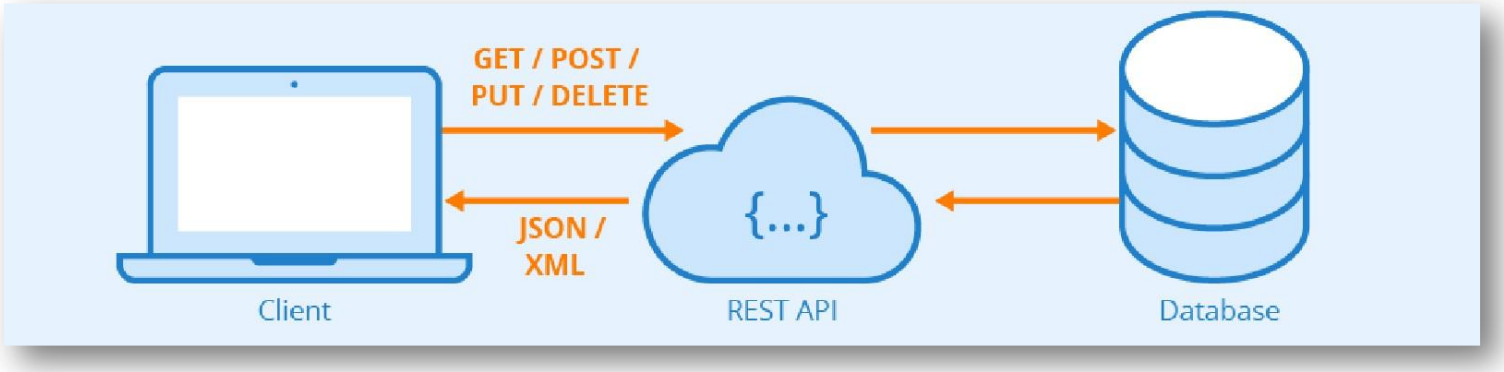
Предмет дослідження – фреймворки React, Angular та Vue.js, їх основні концепції, архітектура, переваги та недоліки, а також методи розробки, тестування та розгортання веб-додатків на їх основі.

Введення

- Сучасні фреймворки веб-розробки, такі як React, Angular та Vue, відіграють вирішальну роль у створенні динамічних та інтерактивних веб-додатків.
- Вибір відповідного фреймворку може суттєво вплинути на результативність розробки, продуктивність додатку та зручність підтримки коду. Аналіз особливостей та порівняння цих фреймворків є дуже актуальним для веб-розробників, які прагнуть обрати конкретну бібліотеку під свої задачі та оптимізувати свої процеси розробки.

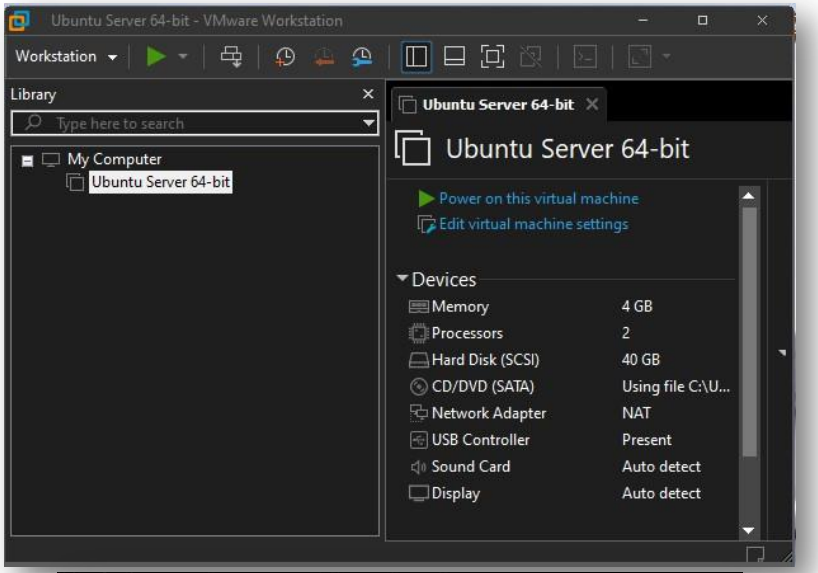
Що таке REST API?

- RESTful API — це архітектура інтерфейсу прикладних програм (API), що використовує HTTP-запити для доступу до ресурсів та їхнього використання.
- Такими HTTP-запитами можуть бути: GET, PUT, POST і DELETE.



Налаштування ОС на віртуальній машині

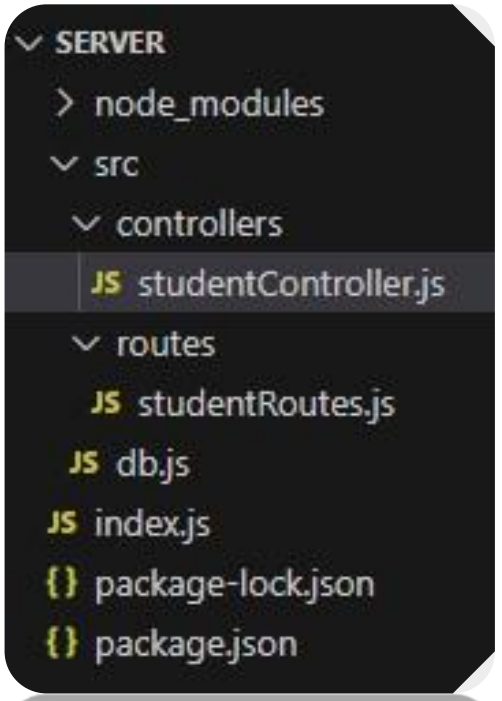
- Операційна система, яка буде встановлена та налаштована на віртуальній машині - це Ubuntu Server. Ubuntu Server відзначається своєю стабільністю, безпекою та універсальністю.
- Під час встановлення Ubuntu Server'а, за допомогою протоколу DHCP, був призначений ip-адрес, який в подальшому буде використовувати API.
- Далі встановлюємо СУБД PostgreSQL. Створюємо базу даних «diploma_postgres» та таблицю «student».
- Наступним етапом це відкриття в брандмауері порта «8080», на якому буде працювати додаток.



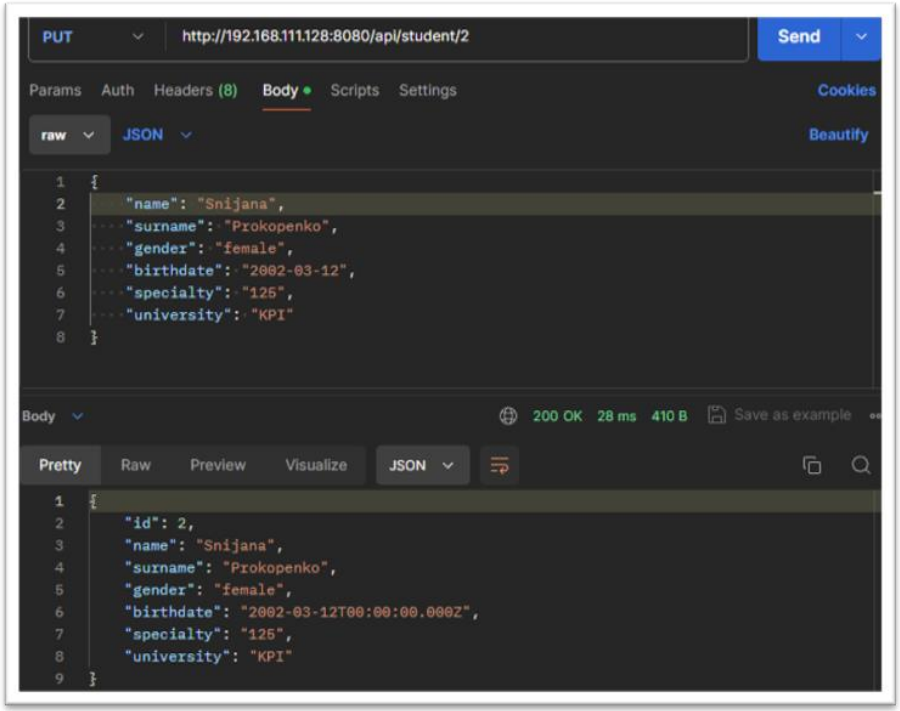

```
dialekt@crud-server:~$ sudo ufw status
[sudo] password for dialekt:
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
8080/tcp (v6)              ALLOW       Anywhere (v6)
```

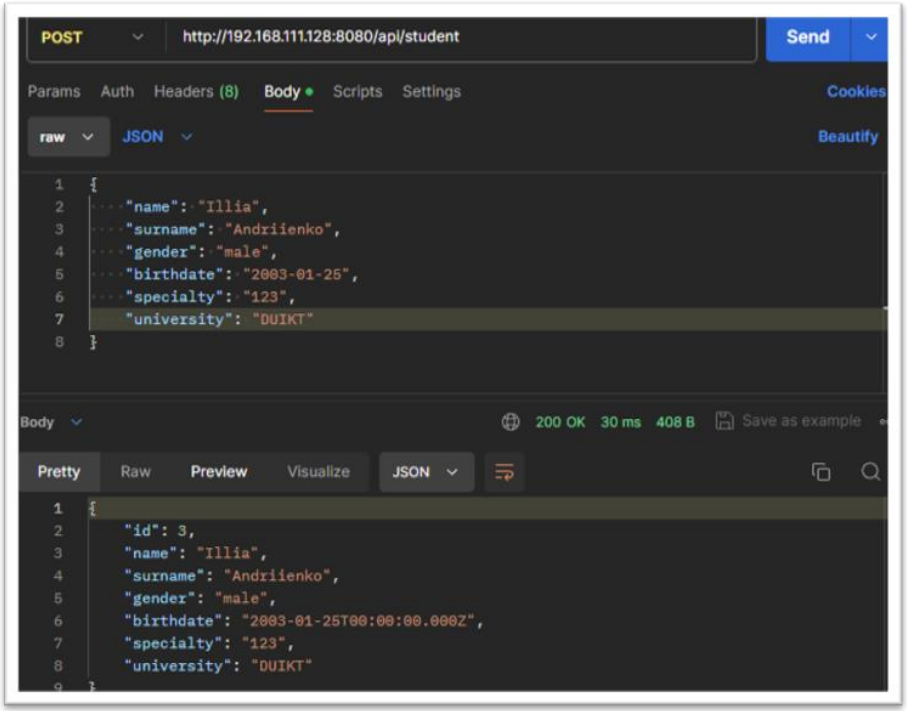
Реалізація REST API на Node.js, Express.js та PM2

- Node JS - це серверна платформа, яка дозволяє розробникам створювати високопродуктивні додатки за допомогою JavaScript.
- Express.js - це швидкий, гнучкий і мінімалістичний веб-фреймворк для Node.js.
- PM2 - це менеджер процесів, який допоможе керувати та підтримувати програму онлайн 24/7.
- Вся логіка, маршрутизація та підключення до PostgreSQL реалізована в каталозі «src» в окремих файлах. Файл «index.js», який є головним виконуючим файлом, відповідає за запуск сервера, в ньому вказується конфігурація REST-додатка.
- Останнім етапом, за допомогою додаткового пакету PM2 організовується безперебійна робота REST API.

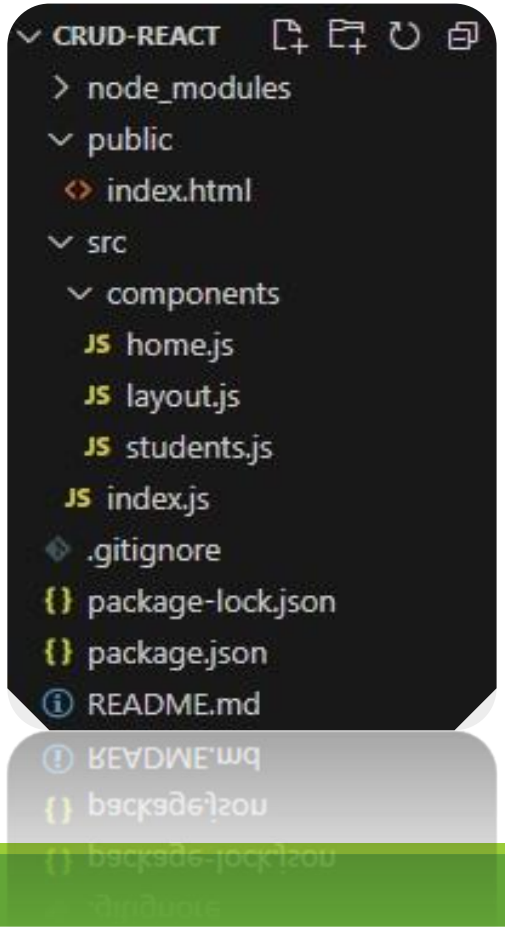


Тестування роботиREST API за допомогою Postman



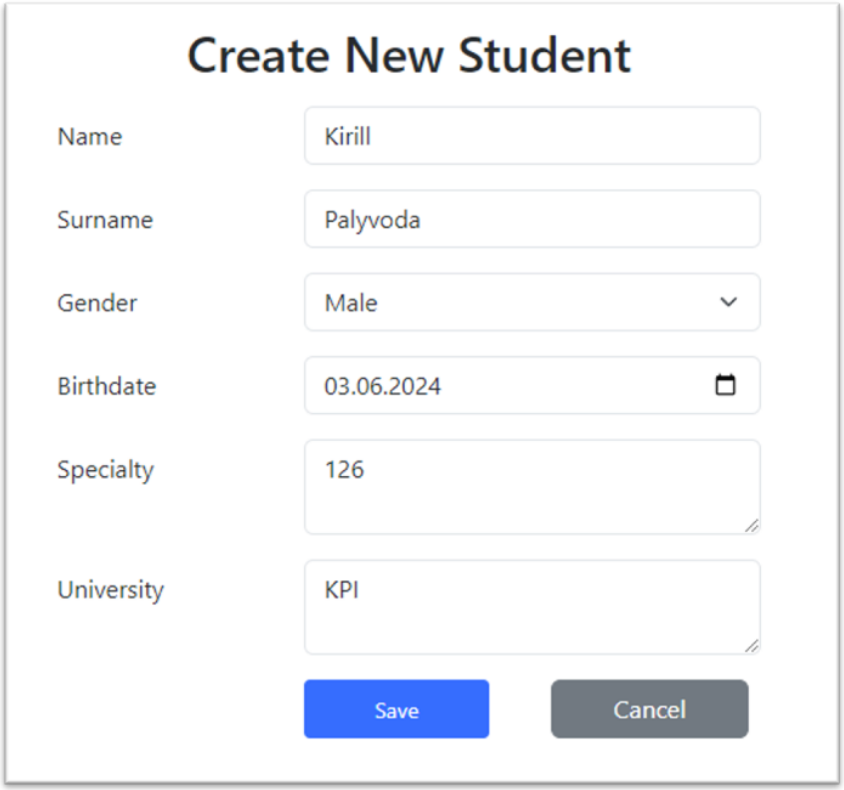


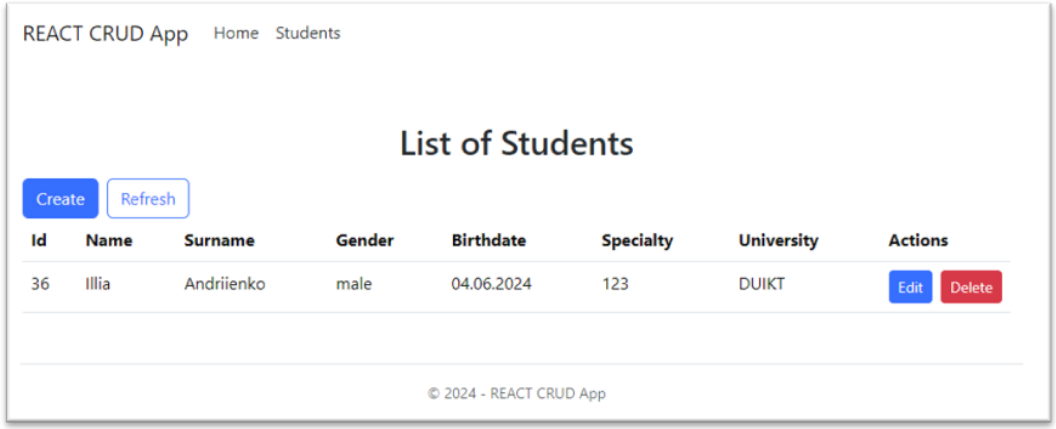
Веб-фреймворк React. Структура React-проєкта та його реалізація

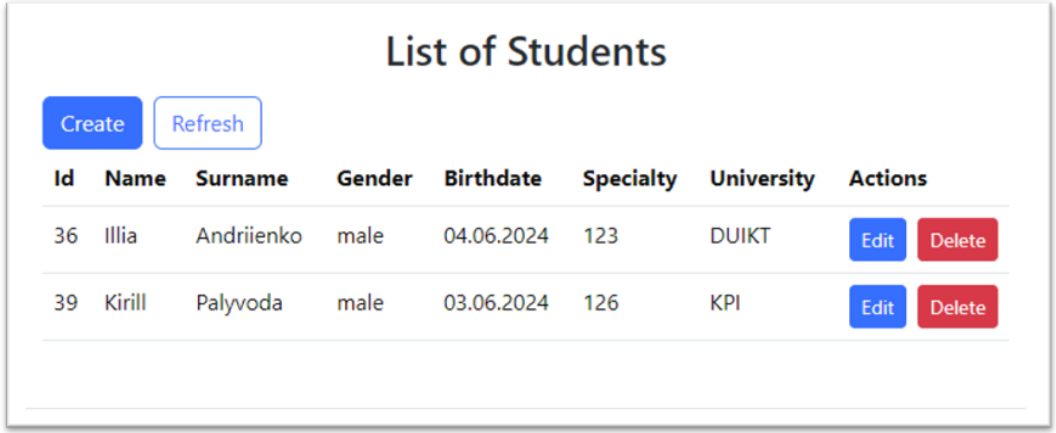


- React JS — це відкритий JavaScript-фреймворк, а точніше, бібліотека JavaScript, яка використовується для розробки інтерфейсів користувача.
- Структура React-проєкту складається з каталогів «src» та «public», де знаходяться основні компоненти та виконавчі файли React-додатка.
- Реалізований на 3-х React-компонентах і в яких ініціалізовані методи для реалізації CRUD-операцій.

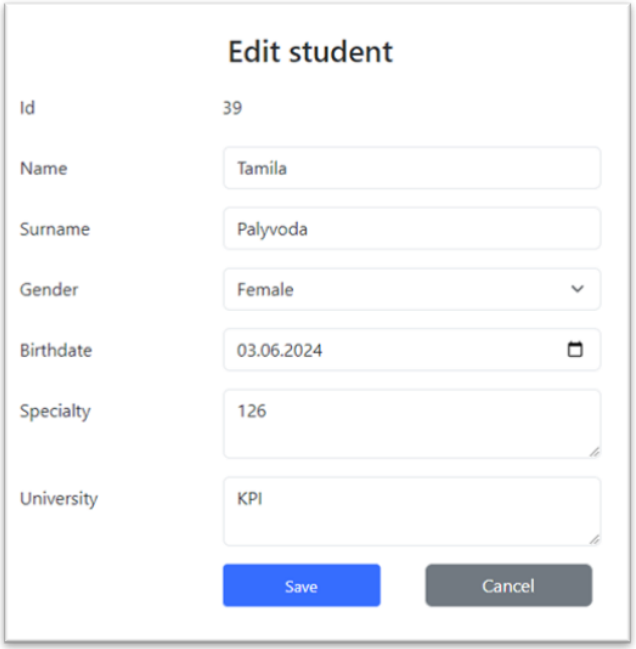
Реалізація CRUD-операцій React додатка

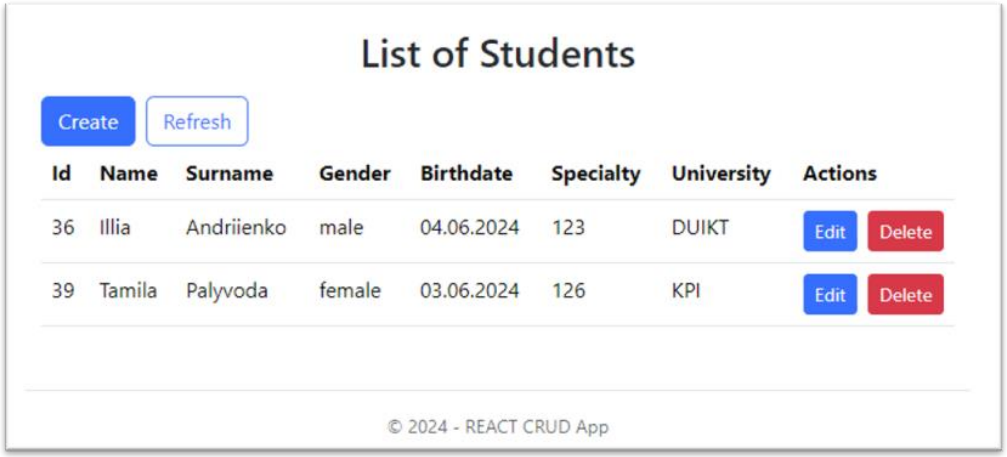






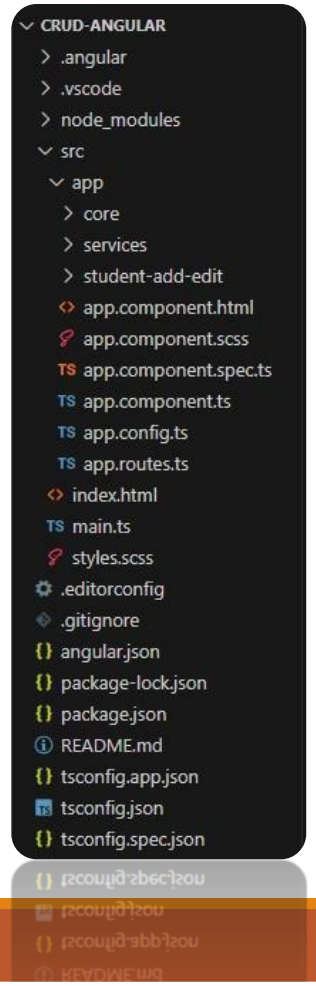
Реалізація CRUD-операцій React додатка



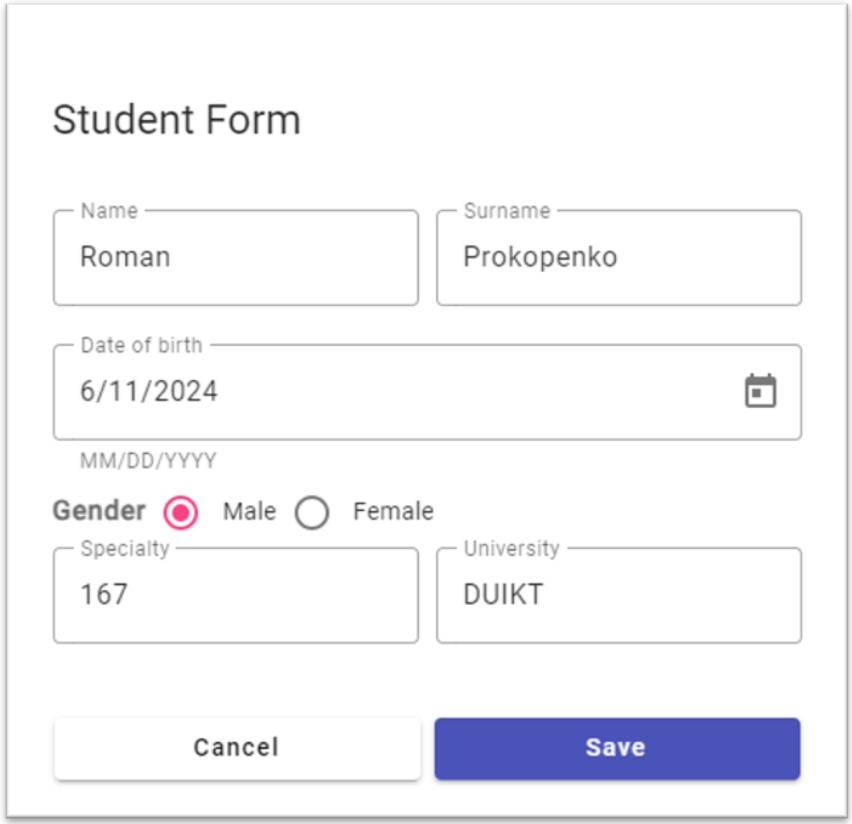


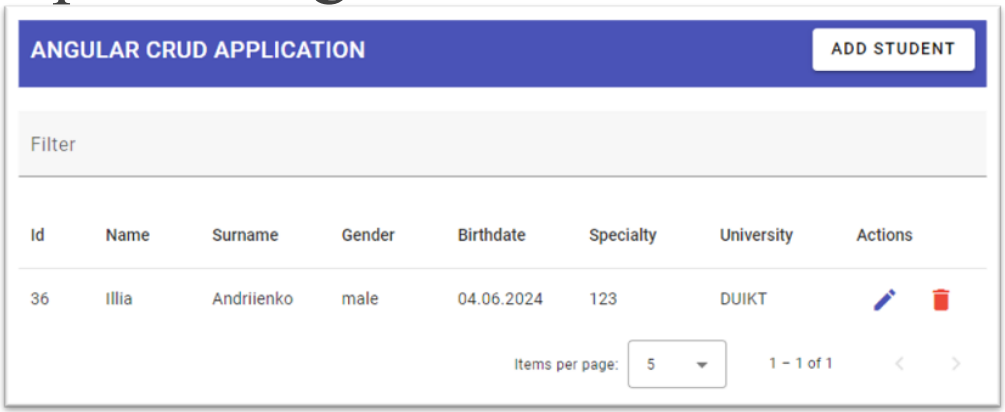
Веб-фреймворк Angular. Структура Angular-проєкта та його реалізація

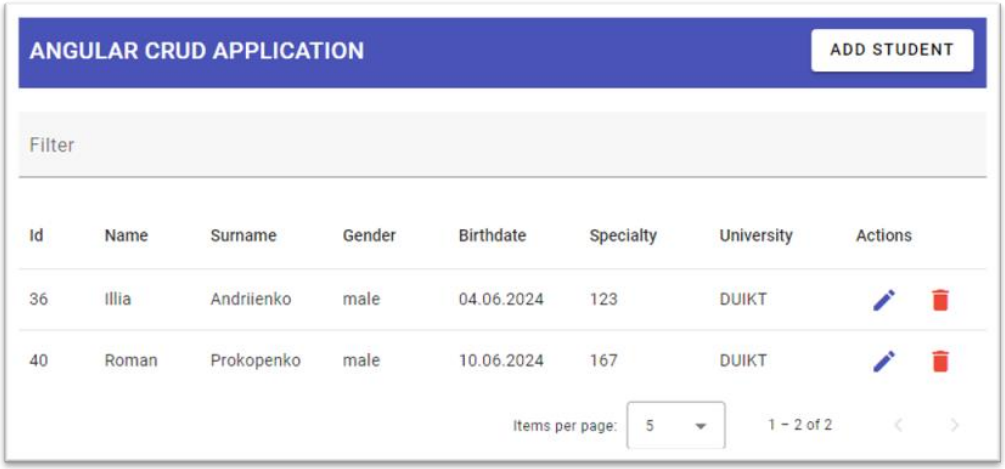
- Angular – це веб-фреймворк, розроблений компанією Google, який використовується для створення динамічних односторінкових додатків.
- Здебільшого структура складається тільки з одного каталога «src», це місце, де розташовується весь вихідний код нашого додатку. Цей каталог повинен містить кожен «Модуль», компонент, сервіс і пов'язаний з ним вихідний код.
- Реалізований на 2-х Angular-компонентах та одному сервіс-файлі. Сервіс-файл вміщує в собі логіку спілкування з Rest API. А компоненти вже цю логіку використовують .



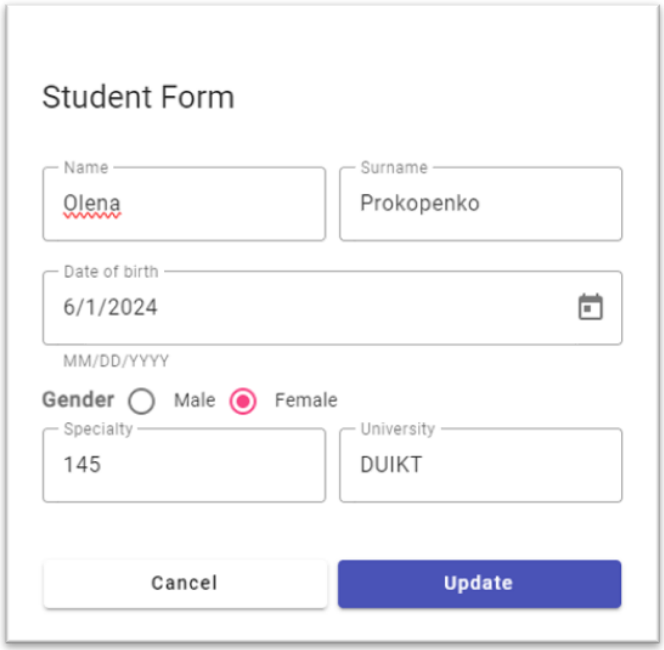
Реалізація CRUD-операцій Angular додатка

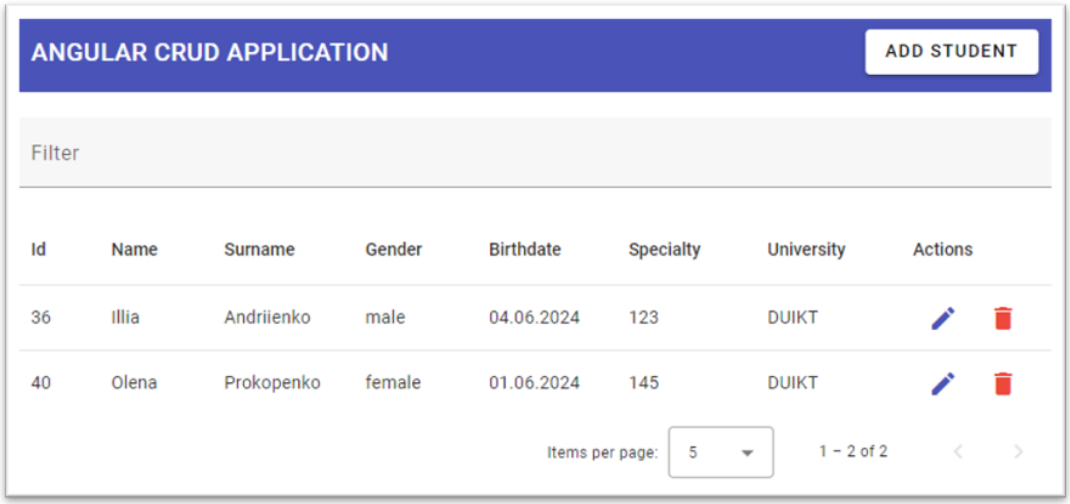




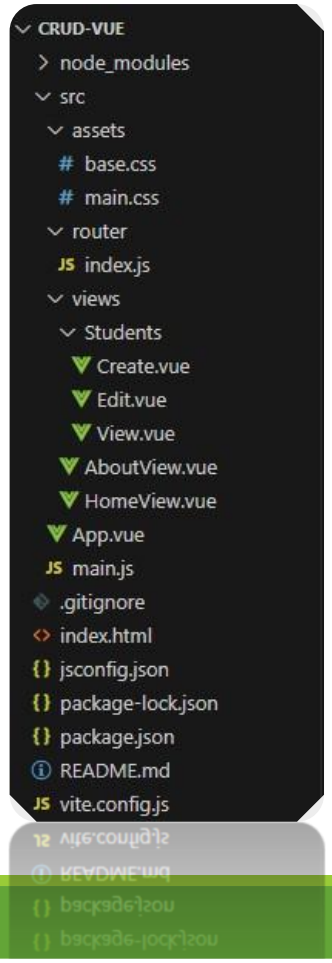


Реалізація CRUD-операцій Angular додатка



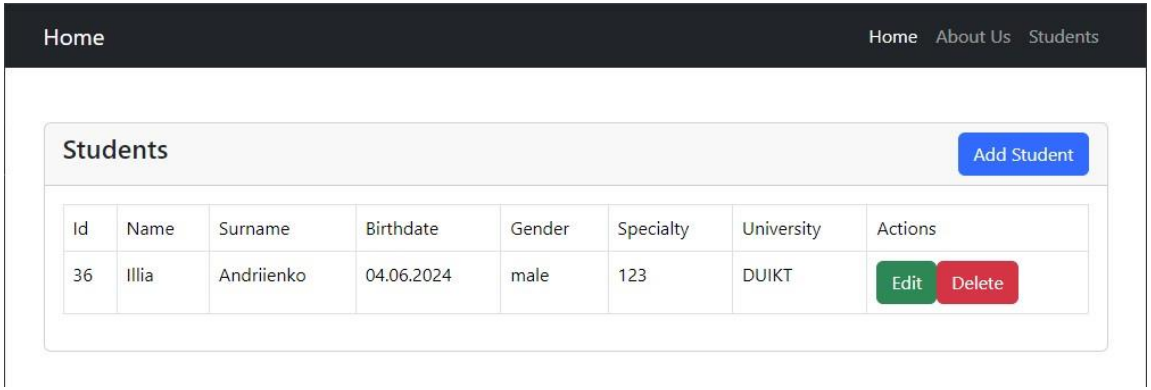


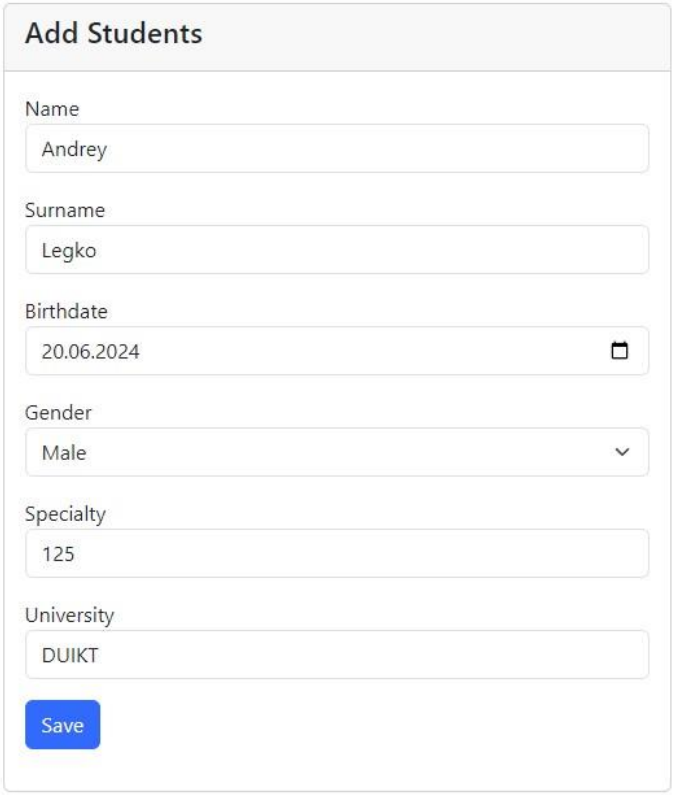
Веб-фреймворк Vue. Структура Vue-проєкта та його реалізація



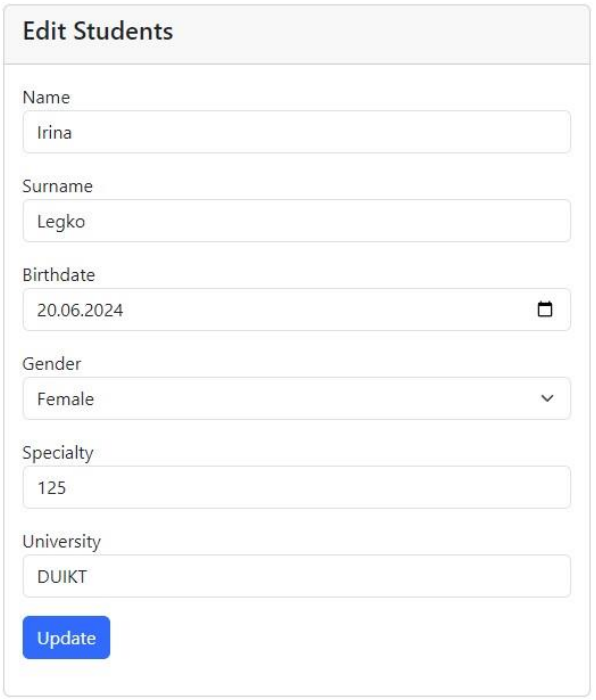
- Vue.js — це прогресивний фреймворк для JavaScript, який використовується для створення веб-інтерфейсів і односторінкових програм.
- Головною директорією проєкту виступає «src». Тут містяться всі основні файли додатка.
- Також створюються 3 компоненти. Кожен з яких відповідає за конкретну дію, наприклад відображення списку студентів. І кожен компонент має методи в яких він звертається до бази даних за допомогою REST API.

Реалізація CRUD-операцій Vue додатка





Реалізація CRUD-операцій Vue додатка



Висновки

- В результаті проведеного дослідження та практичної реалізації різноманітних аспектів веб-розробки з використанням сучасних фреймворків вдалося досягнути кількох важливих результатів. Ці результати сприяють глибшому усвідомленню переваг і недоліків кожного фреймворку та формують міцний фундамент для обґрунтованого вибору в майбутніх проектах.
- Було проведено оцінку переваг та недоліків кожного з фреймворків. Зокрема, React вирізняється високою продуктивністю та великою спільнотою, але вимагає більшої налаштованості. Angular пропонує комплексний набір інструментів «з коробки», але може бути складнішим для початківців. Vue відзначається легкістю та доступністю у вивченні, але має меншу спільноту порівняно з React та Angular.

Продовження висновків

- Реалізація повного комплексу CRUD-операцій з використанням усіх трьох фреймворків. Наглядно показало, як різні шляхи реалізації одних і тих самих функцій можуть впливати на простоту розробки та супроводу додатків у різних середовищах.
- Після всіх проведених досліджень, як на мене, найкраще середовище розробки – React. Він привабив мене з кількох причин:
  1) JSX синтаксис;
  2) ефективне управління станом;
  3) використання хуків (hooks);
  4) зручний спосіб встановлення.
- Ці фактори роблять розробку CRUD-операцій у React надзвичайно зручною та результативною.