

БАКАЛАВРСЬКА РОБОТА

БР. ІІ - 94.00.00.000 ІЗ

Група ІІ-22-4

Грищук Олег

2026

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Гришук Олег Олексійович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка власного додатку планування на базі Vue.js

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня Гришук О.О.
(підпис, ініціали та прізвище здобувача)

Науковий керівник Царева Олександра Степанівна, асистент
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківський національний технічний університет нафти і газу
Інститут, факультет інформаційних технологій
Кафедра інженерії програмного забезпечення
Ступінь вищої освіти бакалавр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедрою

ІІЗ

доцент.

В.В. Бандура

“ ” 2026 р.

ЗАВДАННЯ НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Грищук Олег Олексійович

(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) "Розробка власного додатку планування на базі Vue.js"

керівник проекту (роботи) Царева Олександра Степанівна, асистент

затверджені наказом вищого навчального закладу від “ 21 ” квітня 2026 р. № 179/7

2. Строк подання студентом проекту (роботи) 10 червня 2026 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження перелічених практик

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Аналіз вимог до програмного забезпечення

2. Аналіз вимог і постановка задачі

3. Проектування системи

4. Реалізація проекту

5. Тестування та впровадження

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1 Розподіл завдань фреймворків (рис. 1.1, ст. 12)

2 Історія Angular, React та Vue на GitHub (рис. 2.2, ст. 27)

4. Використання умовного значення для налаштування publicPath (рис. 2.5, ст. 35)

4 Інтерфейс календаря програми нагадування (рис. 3.1, ст. 51)

5 Деталі дня (рис. 3.3, ст. 52)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

2. Дата видачі завдання 2026 р.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обґрунтування теми роботи	17.01.2026	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	21.02.2026	виконано
3	Побудова моделі або алгоритму власного рішення	01.03.2026	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	21.04.2026	виконано
5	Оформлення пояснювальної записки бакалаврської роботи	02.05.2026	виконано

Студент _____ Гришук О.О.
(підпис)

Керівник роботи _____ Царева О.С.
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 75 сторінки, 24 рисунки, 2 таблиці, список використаних джерел із 40 найменувань.

Метою роботи є дослідження принципів розробки сучасних веб-додатків на базі Vue.js і підходів до побудови односторінкових застосунків.

Об'єкт дослідження: веб-додатки планування та організації задач, побудовані на сучасних JavaScript-фреймворках.

Предмет дослідження: методи, технології та інструменти розробки веб-додатків на базі Vue.js, маршрутизація та обробка користувацьких даних.

Результати дослідження: проведено аналіз сучасних JavaScript-фреймворків, досліджено особливості роботи Vue.js, реалізовано веб-додаток управління даними та інтерактивного користувацького інтерфейсу.

У першому розділі розглянуто теоретичні основи розробки односторінкових веб-додатків, принципи роботи JavaScript-фреймворків та механізми двостороннього зв'язування даних у Vue.js.

У другому розділі досліджено методи та інструменти реалізації додатку планування, проведено порівняльний аналіз сучасних фреймворків та розглянуто використання Vue Router для реалізації навігації у веб-додатку.

У третьому розділі розглянуто процес розробки веб-додатку планування, методи збору та аналізу даних користувача, а також оцінювання ефективності та результатів роботи системи.

Висновок: використання Vue.js дозволяє створювати сучасні, масштабовані та інтерактивні веб-додатки з високою продуктивністю та зручним користувацьким інтерфейсом. Реалізований додаток демонструє переваги компонентного підходу і сучасних веб-технологій.

КЛЮЧОВІ СЛОВА: VUE.JS, JAVASCRIPT, ВЕБ-ДОДАТОК, ОДНОСТОРІНКОВИЙ ДОДАТОК, SPA, VUE ROUTER, ПЛАНУВАННЯ ЗАДАЧ, FRONTEND-РОЗРОБКА, ІНТЕРАКТИВНИЙ ІНТЕРФЕЙС, КОМПОНЕНТНИЙ ПІДХІД.

ANNOTATION

Bachelor's thesis of 75 pages, 24 figures, 2 tables, and a reference list containing 40 sources.

The aim of the work is to study the principles of developing modern web applications based on Vue.js and approaches to building single-page applications.

Research object: planning and task management web applications built on modern JavaScript frameworks.

Research subject: methods, technologies, and tools for developing web applications based on Vue.js, routing, and user data processing.

Research results: an analysis of modern JavaScript frameworks was conducted, the features of Vue.js were studied, and a planning web application with data management, and an interactive user interface was implemented.

The first section describes the theoretical foundations of single-page web application development, the principles of JavaScript frameworks, and the mechanisms of two-way data binding in Vue.js.

The second section analyzes methods and tools for implementing the planning application, provides a comparative analysis of modern frameworks, and examines the use of Vue Router for navigation in the web application.

The third section covers the development process of the planning web application, methods of collecting and analyzing user data, and evaluation of the system's efficiency and results.

Conclusion: the use of Vue.js makes it possible to create modern, scalable, and interactive web applications with high performance and a user-friendly interface. The implemented planning application provides the advantages of the component-based approach and modern web technologies.

KEY WORDS: VUE.JS, JAVASCRIPT, WEB APPLICATION, SINGLE-PAGE APPLICATION, SPA, VUE ROUTER, TASK PLANNING, FRONTEND DEVELOPMENT, INTERACTIVE INTERFACE, COMPONENT-BASED APPROACH.

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБ-ДОДАТКІВ НА БАЗІ VUE.JS	10
1.1 Основи розробки односторінкових веб-додатків	10
1.2 JavaScript-бібліотеки та фреймворки для створення веб-додатків	18
1.3 Принцип двостороннього зв'язування даних у Vue.js	23
1.4 Висновки по розділу	27
РОЗДІЛ 2. МЕТОДИ ТА ІНСТРУМЕНТИ РЕАЛІЗАЦІЇ ДОДАТКУ ПЛАНУВАННЯ	28
2.1 Порівняльний аналіз сучасних JavaScript-фреймворків	28
2.2 Концепція та функціональні можливості додатку планування	31
2.3 Реалізація маршрутизації та навігації за допомогою Vue Router	44
2.4 Висновки по розділу	54
РОЗДІЛ 3. РОЗРОБКА, ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ВЕБ-ДОДАТКУ	55
3.1 Реалізація веб-додатку планування засобами Vue.js	55
3.2 Збір, обробка та аналіз даних користувача	60
3.3 Оцінка ефективності, результати роботи та обмеження системи	67
3.4 Висновки по розділу	75
ВИСНОВКИ	76
СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА	78

					БР.ІІ –94.00.00.000 ІІЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Розробка власного додатку планування на базі Vue.js Пояснювальна записка	Лім.	Арк.	Акрушіє
Розроб.		Гришук О.О.						
Перевір.		Царева О.С.					7	
Реценз.		Лютак І.З.				ІФНТУНГ ІІ-22-4		
Н. Контр.		Піх М.М.						
Затверд.		Бандура В. В.						

ВСТУП

У сучасному світі цифрові технології відіграють важливу роль у повсякденному житті людини, забезпечуючи автоматизацію багатьох процесів та підвищення ефективності організації часу. Зростання обсягів інформації, необхідність керування великою кількістю завдань і потреба в швидкому доступі до даних формують попит на сучасні веб-додатки для планування та управління задачами. Особливої популярності набувають односторінкові веб-додатки (SPA), які забезпечують високу швидкодію, інтерактивність та зручність користувацького інтерфейсу. У зв'язку з цим актуальним є використання сучасних JavaScript-фреймворків, зокрема Vue.js, який дозволяє створювати масштабовані, гнучкі та продуктивні веб-застосунки.

Vue.js є одним із найпоширеніших frontend-фреймворків завдяки простоті використання, компонентному підходу, підтримці двостороннього зв'язування даних та високій продуктивності. Використання Vue.js дозволяє розробникам створювати інтерактивні користувацькі інтерфейси, ефективно організовувати структуру додатку та реалізовувати функціональність без значного ускладнення архітектури програмного забезпечення. Сучасні додатки планування повинні забезпечувати можливість створення та редагування задач, управління подіями, зручну навігацію між сторінками та швидку взаємодію з користувачем, що робить використання Vue.js доцільним і перспективним.

Актуальність теми зумовлена необхідністю створення ефективного веб-додатку планування, який забезпечуватиме зручне управління задачами, високу швидкість роботи та сучасний користувацький інтерфейс. Існуючі рішення часто мають надлишкову функціональність, складний інтерфейс або недостатню гнучкість для персоналізації. Використання Vue.js дозволяє реалізувати легкий, продуктивний та масштабований веб-додаток із підтримкою сучасних підходів до frontend-розробки. У сучасних умовах розвитку веб-технологій та зростання потреб користувачів у цифрових інструментах організації часу створення власного додатку планування є актуальним напрямом дослідження.

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

Метою роботи є розробка власного веб-додатку планування на базі Vue.js, який забезпечує управління задачами, зручну навігацію між сторінками, обробку даних користувача та інтерактивну взаємодію з інтерфейсом.

Завданнями дослідження є аналіз предметної області та сучасних підходів до розробки веб-додатків, дослідження JavaScript-фреймворків і бібліотек, аналіз принципів роботи Vue.js та двостороннього зв'язування даних, порівняння сучасних frontend-фреймворків, проектування архітектури додатку планування, реалізація навігації за допомогою Vue Router, створення основних функціональних модулів системи, тестування додатку та оцінювання ефективності його роботи.

Об'єктом дослідження є процеси розробки сучасних веб-додатків для планування та організації задач із використанням frontend-технологій.

Предметом дослідження є методи, технології та інструменти створення веб-додатків на базі Vue.js, включаючи компонентний підхід, маршрутизацію, управління станом і механізми взаємодії користувача з інтерфейсом.

Методи дослідження включають аналіз наукових джерел і технічної документації для вивчення сучасних підходів до frontend-розробки, порівняльний аналіз JavaScript-фреймворків, моделювання структури веб-додатку, експериментальний метод реалізації та тестування програмного забезпечення, а також аналіз результатів роботи системи.

Розроблений веб-додаток передбачатиме можливість створення, редагування та видалення задач, організацію подій, зручну навігацію між сторінками та інтерактивну взаємодію з користувачем. Особлива увага приділятиметься продуктивності, масштабованості, адаптивності інтерфейсу та зручності використання. Очікується, що реалізований додаток забезпечить ефективне управління задачами та демонструватиме переваги використання Vue.js для створення сучасних веб-застосунків.

Бакалаврська робота містить 75 сторінки, 24 рисунки, 2 таблиці, 3 розділи, список використаних джерел із 40 найменувань.

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБ-ДОДАТКІВ НА БАЗІ VUE.JS

1.1 Основи розробки односторінкових веб-додатків

Протягом останніх семи років JavaScript щороку вважався найуживанішою мовою програмування. У 2019 році статистика показала, що 67,8% розробників використовують JavaScript як одну з мов програмування. Крім того, 95,2% вебсайтів у світі, приблизно 1,52 мільярда вебсайтів, написані на JavaScript. JavaScript, разом із мовою гіпертекстової розмітки (HTML) та каскадними таблицями стилів (CSS), є основними мовами веб-сайту або веб-застосунку. У той час як HTML та CSS відповідають за стиль та структуру сайту, JavaScript дозволяє розробникам створювати функції та поведінку для сторінки. Ця мова працює з моделлю об'єктів документа (скорочено DOM), яка визначає, що робити, коли користувач виконує певні дії у вашому застосунку. Люди часто думають про JavaScript як про мову інтерфейсу. Тим не менш, ми можемо використовувати JavaScript для написання серверної частини застосунку через середовище Nodejs, яке використовує JavaScript і також дуже популярне.

Більше того, завдяки JavaScript також доступні переходи прокручування та переміщення об'єктів. Через різноманітність функціональних можливостей, які пропонує JavaScript, багато сучасних розробників браузерів намагаються допомогти своїм браузерам запускати JavaScript якомога швидше, щоб покращити взаємодію з користувачем [1]. У статті Кайли Браун з Codecademy розповідається про те, як JavaScript здобув свою популярність, перераховані ці фактори, які, «він перетворив веб-браузери на платформу додатків», виділені жирним шрифтом. Однак є ще один момент, про який я раніше не згадував, і який, на мою думку, є найважливішим фактором, що робить написання JavaScript не тільки комфортнішим, але й доступнішим та менш трудомістким. Це тому, що JavaScript має багато фреймворків та бібліотек, які надзвичайно допомагають розробникам покращити свій стиль кодування.

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

React, Vue.js та Angular є одними з найпопулярніших фреймворків JavaScript на даний момент. Хоча React та Angular розробляються та підтримуються відповідно Facebook та Google, фреймворк Vue.js був створений Еваном Ю та його командою. Хоча немає великої компанії, яка б підтримувала Vue.js, цей фреймворк все ще привертає значну увагу спільноти веб-розробників [2]. Як і на GitHub, Vue.js перевершив React, найвідоміший фреймворк на даний момент, за кількістю зірок, отриманих у його репозиторії. Завдяки цьому ми можемо сказати, що хоча більшість компаній все ще використовують React (рисунок 1.1), люди звертають свою увагу на Vue.js, і в майбутньому він може стати більш популярним.

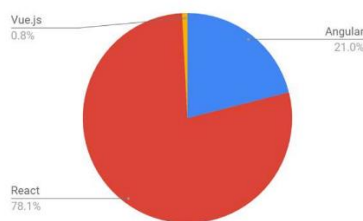


Рисунок 1.1 - Розподіл завдань фреймворків

Щоб розпочати дослідження, ставиться кілька питань, які допоможуть створити орієнтири для роботи. Метою роботи є допомога новим розробникам у розумінні деяких фундаментальних аспектів Vue.js, фреймворку JavaScript, та їхньої користі для процесу розробки. Щоб досягти цієї мети, вирішує вказати на деякі ключові особливості фреймворку та пояснює, як ці особливості стають перевагами для процесу розробки.

Для цього робота має на меті відповісти на ці два основні питання:

- Які основні особливості фреймворку Vue.js та для чого вони використовуються?
- Як ці функції покращують процес розробки?

«Різниця між дедукцією та індукцією» стверджується, що дослідження – це інструмент, який допомагає підтвердити концепцію, що, у свою чергу, сприяє пізнанню світу. Але щоб люди повірили в дослідження, потрібен

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

певний підхід до міркування, щоб переконати їх [3]. Два найпоширеніші підходи до дослідження – це дедукція та індукція, і вони обидва протилежні один одному. Дедуктивне міркування зазвичай починається з гіпотези, потім переходить до спостереження і, нарешті, підтверджує теорію. З іншого боку, «Індуктивне міркування зазвичай використовує дослідницькі питання, щоб звузити обсяг дослідження». На рисунку 1.2 нижче також описано різницю між підходами. Обидва ці підходи дуже корисні по-своєму. Теза стосується доведення того, як фреймворк Vue.js позитивно впливає на процес розробки, надаючи перевірені реалізації та порівнюючи їх між собою. Тому дедуктивний підхід є більш придатним для цього дослідження.

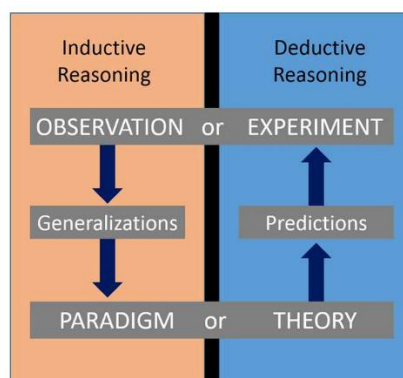


Рисунок 1.2 - Індуктивні та дедуктивні міркування

Для завершення дослідження збір та аналіз даних є вирішальною частиною, і його не повинно бракувати в жодному дослідженні, оскільки їх відсутність може призвести до неточних висновків, що, у свою чергу, робить дослідження недійсним [4]. Тому це дослідження потребує іншого підходу до збору та аналізу даних. Якісний та кількісний підходи добре відомі для роботи з даними; кожен має свої цілі, методи та необхідний для отримання різних видів знань. Кількісний підхід використовує числа, графіки та статистику для аналізу або встановлення теорій. Опитування, експериментальні дослідження спостережливі дослідження та контент-аналіз – це способи, якими дослідники отримують дані для кількісного підходу, тобто слова, аналізи та символи.

Якісний та кількісний підходи

	Якісні (Qualitative)	Кількісні (Quantitative)
Концептуальні	Спрямовані на розуміння людської поведінки з точки зору інформанта	Спрямовані на виявлення фактів про соціальні явища
	Припускають динамічну та узгоджену реальність	Припускають фіксовану та вимірювану реальність
Методологічні	Дані збираються шляхом спостереження за учасниками та інтерв'ю	Дані збираються шляхом вимірювання об'єктів
	Дані аналізуються за темами на основі описів інформантів	Дані аналізуються шляхом числових порівнянь та статистичних висновків
	Звіт про дані надається мовою інформанта	Звіт про дані надається за допомогою статистичного аналізу

Веб-застосунок та веб-сайт:

Метою роботи є впровадження Vue.js у процес розробки веб-застосунків. Тому нам потрібно зрозуміти, що таке веб-застосунок і чим він відрізняється від звичайного веб-сайту? Веб-сайт надає клієнтам інформацію, яка сьогодні в основному використовується компаніями для маркетингових цілей. Такі веб-сайти, як Вікіпедія, газети, статті та блоги, є чудовими прикладами веб-сайтів. Вміст цих сайтів зазвичай залишається незмінним щоразу, коли ми їх відвідуємо.

На відміну від веб-сайту, веб-застосунок – це застосунок, який працює в інтернет-браузері. Застосунок вимагає взаємодії з клієнтами, тому метою застосунку є не лише відображення інформації. Веб-застосунок може надавати широкий спектр послуг, таких як комунікація, розваги, комерція та багато іншого. Деякі популярні веб-застосунки – це YouTube, Facebook, Gmail [5]. Робота присвячена впровадженню Vue.js у процес розробки веб-застосунку, щоб побачити переваги, які він може принести.

Фронтенд-розробка

Загальний веб-застосунок складається з чотирьох основних компонентів: фронтенду, дизайну інтерфейсу/UX, бекенду та бази даних. Фронтенд-розробка – це процес обробки візуальної перспективи для користувача (Steward, 2019). Здебільшого все, що бачить користувач, обробляється в процесі фронтенд-

розробки. Для веб-розробки фронтенд використовує HTML, CSS та JavaScript. Популярність цього тріо змушує більшість сучасних браузерів, таких як Chrome, Mozilla та Microsoft Edge, якомога швидше адаптувати їх, щоб вони могли надати своїм клієнтам найновіші функції, що надаються цими мовами. Як зазначалося в блозі, HTML, CSS та JavaScript – це два з десяти обов’язкових навичок для фронтенд-розробників. Нижче наведено повний список найважливіших знань, необхідних у блозі: HTML/CSS, JavaScript/Jquery, фреймворк CSS та JavaScript, Попередня обробка CSS, контроль версій/Git, адаптивний дизайн, тестування/налагодження, інструменти розробника браузера, інструменти для створення та автоматизації / веб-продуктивність, командний рядок

Загалом, ці навички та знання потрібні для більшості посад front-end розробників. Кожна частина front-end розробки вимагає певної навички з цього списку. Наприклад, адаптивний дизайн є критично важливим, якщо розробник вирішив, що його веб-сайт або веб-додаток працюватиме як на настільних комп’ютерах, так і на мобільних пристроях [6]. Основною темою роботи є Vue.js, відомий JavaScript-фреймворк, який, як ви бачите, посідає третє місце в цьому списку. Інші навички можуть бути згадані в роботі, але вона зосереджена лише на обговоренні цієї теми.

Дизайн інтерфейсу користувача/інтерфейсу користувача

Дизайн користувацького інтерфейсу (UI), який часто плутають з дизайном користувацького досвіду (також відомим як UX), – це процес визначення того, як виглядатиме ваш веб-сайт, а також як відбуватимуться всі взаємодії. UI зосереджується на візуальному представленні, від вибору колірних схем до типографіки; його цілі – забезпечити гарний зовнішній вигляд продукту. Дизайн UI також відповідає за створення адаптивності, ефективності та доступності веб-сайту.

На відміну від інтерфейсу користувача, UX, з іншого боку, в першу чергу працює над покращенням вражень клієнта та, як правило, спрямовує користувача до дій, яких компанія хоче від нього вжити, досягаючи кінцевої мети компанії. Водночас, він також підвищує задоволеність клієнтів. UX-дизайн визначає, яким

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

стане дизайн інтерфейсу користувача продукту, оскільки інтерфейс користувача є частиною UX. Користувацький досвід більше не стосується візуального аспекту, а стосується всього, що покращує враження, задоволення та задоволеність клієнта. У таблиці нижче наведено деякі ключові відмінності між дизайном інтерфейсу користувача та дизайном користувацького досвіду. Цей рисунок допомагає розробникам зрозуміти ідею UI/UX дизайну веб-застосунку, що є необхідним для більшості веб-застосунків. У роботі немає подальшого дослідження чи обговорення цієї теми.

UI Design	UX Design
There are lot of common things between UI and UX design, while there are some key points also.	
1. UI is focused on looks and design.	1. UX is focused on functionality.
2. UI designer to focus on the more visual elements.	2. UX designer is concerned with the conceptual aspects.
3. UI makes interfaces beautiful.	3. UX makes interfaces useful.
4. UI includes an artistic component.	4. UX focuses more on the functionality and purpose of the design.
5. They work generally on graphic design, brand design, web design, layouts	5. They work generally on wireframes, prototypes, research, functional analyst.

Рисунок 1.3 - Дизайн інтерфейсу користувача/користувацької інтерфейсу

Розробка бек-енду

На відміну від фронтенд-розробки, бекенд-розробка вказує на серверну частину програми. Бекенд складається з кодів, які взаємодіють між браузером клієнта та базою даних. Докладніше, це включає оновлення, керування та моніторинг функцій і даних для програми. Більшість із цих речей не може бачити клієнт, тому це називається бекендом.

У той час як фронтенд-розробка використовує HTML та CSS як основні мови, бекенд-розробники повинні ознайомитися з такими мовами програмування, як Java, PHP, Ruby on Rails, .NET або Python [7]. Зазвичай розробникам потрібно використовувати лише одну або дві мови одночасно, залежно від технологій, які використовує їхня компанія, тому належне вивчення всіх цих мов не є необхідним. JavaScript також можна використовувати для написання бекенд-розробки для додатків за допомогою Node.js, середовища виконання JavaScript.

Node.js є одним з найбільш успішні мови програмування для серверної частини, і багато великих компаній, таких як NASA, IBM, Microsoft, PayPal, Walmart, Uber, LinkedIn, Netflix, використовують їх.



Рисунок 1.4 - Розробники фронтенду та бекенду

База даних

База даних – це сукупність даних, які систематично зберігаються. Ця система дозволяє запитувати, оновлювати, маніпулювати та керувати даними дуже ефективним та логічним способом. Існує багато типів баз даних, таких як реляційні бази даних, бази даних No SQL, об'єктно-орієнтовані бази даних. Кожен тип бази даних корисний у своїй галузі [8]. Як і бази даних NoSQL, також відомі як нереляційні бази даних, вони можуть зберігати та керувати даними без структури або напівструктурованими даними. Цей тип бази даних широко використовується у веб-індустрії, особливо коли кількість веб-додатків зростає, а самі додатки стають складнішими. Використано базу даних NoSQL, MongoDB, якщо бути конкретним, для створення бази даних для досліджуваного випадку застосування, який представлений у наступному розділі.

Як вже згадувалося, JavaScript, безперечно, є найпопулярнішою мовою програмування в наш час, особливо коли йдеться про веб-індустрію. Тоді, коли ця мова програмування була недоступна, в Інтернеті були лише повністю статичні веб-сайти, які використовували лише HTML та CSS. І Netscape змінив це, винайшовши, як ми знаємо сьогодні, найпоширенішу мову програмування, Javascript. Спочатку поява JavaScript не була повним успіхом. Але зрештою JavaScript став настільки популярним, що знайти іншу мову для його заміни дуже

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

важко, якщо не неможливо. Роль JavaScript полягала в тому, щоб зробити веб-сайт повнофункціональним додатком. Він дозволяє створювати інтерактивний та динамічний веб-сайт, логіка якого може виконуватися з боку клієнта, чого ніколи раніше не траплялося. JavaScript також отримав значну підтримку від великих технологічних компаній, таких як Facebook або Google, оскільки ці компанії створили React та Angular, два найпопулярніші фреймворки JavaScript, для подальшого розширення своїх можливостей.

1.2 JavaScript-бібліотеки та фреймворки для створення веб-додатків

Визначення JavaScript може створити враження, що JavaScript є відповідною мовою програмування для розробників. Але насправді створити веб-додаток з нуля за допомогою Vanilla JavaScript дуже складно та займає набагато більше часу [9]. Саме тут стають у пригоді бібліотеки та фреймворки JavaScript. Ці бібліотеки та фреймворки є шаблонами для розробників, які допомагають зробити процес розробки набагато ефективнішим. Зокрема, це колекції попередньо написаного коду скрипту JavaS, який може виконувати певні завдання з кращою продуктивністю та надійністю. Особливо, якщо ви новачок, оскільки більшість бібліотек та фреймворків написані досвідченими та професійними розробниками. Крім того, вам не потрібно писати ці рядки коду, що пришвидшує процес розробки. Наприклад, JQuery була популярною бібліотекою JavaScript, яка дуже зручно кодує за допомогою JavaScript, оскільки ця бібліотека надає багато однорядкових методів, які відповідають багатьом рядкам коду, якщо їх написати лише на JavaScript.

Фреймворки та бібліотеки JavaScript не обмежилися лише забезпеченням зручності та спрощення написання коду [10]. Деякі фреймворки JavaScript навіть змінюють спосіб, яким розробники створюють свої проекти. До них належать React, Angular, Vue.js – тріо фреймворків, які наразі є лідерами в цій галузі.

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

```

1 <template>
2   <p>{{ greeting }} World!</p>
3 </template>
4
5 <script>
6   module.exports = {
7     data: function () {
8       return {
9         greeting: 'Hello'
10      }
11    }
12  }
13 </script>
14
15 <style scoped>
16   p {
17     font-size: 2em;
18     text-align: center;
19   }
20 </style>

```

Рисунок 1.5 - Шаблон однофайлового компонента Vue.js

Vue.js вперше був випущений у 2014 році Еваном Ю, який працював у Google, використовуючи Angular, перш ніж зосередитися на розробці власного фреймворку. Версія, що використовується в роботі, - Vue.js 2.0 і вище, що означає, що робота може бути неточною для інших версій до Vue.js 2.0 і починаючи з 3.0 (яка вийде пізніше). Згідно з офіційним веб-сайтом Vue.js, вони визначають Vue.js як прогресивний фреймворк, що використовується для розробки користувацьких інтерфейсів, що належать до фронтенд-розробки. Vue.js - це компонентний фреймворк, за допомогою якого розробники можуть створювати власні елементи, які розробники можуть повторно використовувати протягом усього проєкту [11]. Ця функція не є ексклюзивною для Vue.js, оскільки React та Angular також мають свою функцію компонентів. Але для Vue.js компонент може включати HTML, CSS та JavaScript разом, що утворює так званий однофайловий компонент. В однофайловому компоненті HTML-код пишеться всередині тегу <template>, тоді як JavaS-скрипт та CSS-код належать до тегів <script> та <style> відповідно. На рисунку 1.6 нижче показано, як має виглядати однофайловий компонент або файл .vue.

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

```

1  <script>
2  import ComponentA from './ComponentA'
3  import ComponentB from './ComponentB'
4
5  export default {
6    name: "MyComponent",
7    components: { ComponentA, ComponentB },
8    data() {
9      return {
10        name: '',
11        age: '',
12      }
13    },
14    methods: {
15      getName() {
16        return this.name
17      },
18      setAge(age) {
19        this.age = age
20      }
21    },
22    created() {
23      this.name = 'Nguyen'
24      this.age = 22
25    }
26  }
27 </script>

```

Рисунок 1.6 - Приклад опцій компонентів

Тепер давайте детальніше розглянемо деякі цікаві функції, які роблять Vue.js блискучим та популярним на даний момент. Варіанти компонентів HTML та CSS можуть бути знайомі більшості розробників, але те, що створює відмінності між фреймворками, - це їхня компонентна структура. Компонент - це багаторазово використовуваний користувацький елемент зі своїми методами, логікою та стилем. Компоненти можуть взаємодіяти один з одним, передаючи дані (props) або прослуховуючи події інших компонентів, а потім запускаючи певні дії. Погляньте на рисунок 1.6 це приклад, написаний для демонстрації вмісту всередині тегу `<script>` окремого файлового компонента.

Спочатку в цьому компоненті імпортуються та використовуються інші компоненти [12]. Після цього виконується експорт JavaScript-об'єкта за замовчуванням, який є самим компонентом. Усередині цього об'єкта розробники оголошують усі дані, DOM, життєвий цикл, ресурси та інші опції компонента. На рисунку 1.6 показано деякі основні та найчастіше використовувані опції. Деякі опції можуть бути дуже простими, наприклад, "name" для оголошення назви компонента та "data" для оголошення даних компонента. Опція "Components"

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

використовується для оголошення імпортованих компонентів, які використовуються в цьому компоненті. Опція "Methods" використовується для оголошення методів компонента. Остання опція в цьому прикладі - "created", що є перехоплювачем життєвого циклу. Коли компонент досягає стадії "created" свого життєвого циклу, він автоматично викликає цей метод. Існують інші перехоплювачі життєвого циклу, а також інші опції компонента. Повний список опцій можна знайти на офіційній сторінці API Vue.js або пізніше в розділі.

Повторна активність

Всередині тегу <script> є функція даних, за допомогою якої розробники можуть визначати дані для компонента. Об'єкт, що повертається цією функцією, стає реактивними ресурсами компонента, що означає, що щоразу, коли властивість цього об'єкта змінюється, компонент запускає повторний рендеринг та оновлення представлення. Іншими словами, коли розробники оголошують дані всередині об'єкта даних, компонент переглядає кожну властивість об'єкта та перетворює їх на сетери та геттери. Кожен компонент має свого спостерігача для відстеження змін даних за допомогою сетерів та геттерів. Якщо виявляється зміна, спостерігач запускає повторний рендеринг, який оновлює представлення користувача останніми даними. На рисунку 1.7, наданому офіційним веб-сайтом Vue.js нижче, показано, як працює процес реактивності в компоненті Vue.js. Ця функція реактивності дозволяє зв'язування даних простішим та контрольованішим [13]. У блозі, написаному AltexSoft, консалтинговою компанією з технологій, це згадувалося як перевага Vue.js, класної функції, яку навіть ви успадкували від фреймворку Angular та застосували у своєму продукті.

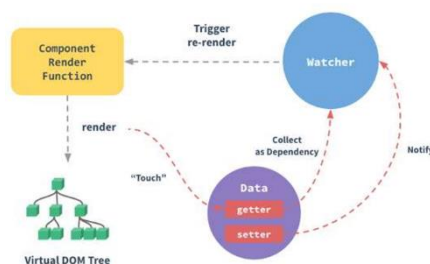


Рисунок 1.7 - Процес реактивності в компоненті Vue.js

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3 Принцип двостороннього зв'язування даних у Vue.js

Як зазначалося вище, двостороннє зв'язування даних є переконливою особливістю фреймворку Vue.js [14]. Воно виявилось дуже корисним для зв'язування даних у вхідних даних форми за допомогою директиви v-model. На рисунку 1.8 нижче показано приклад двостороннього зв'язування даних у формі з використанням директиви v-model.



Рисунок 1.8 - Приклад двостороннього зв'язування даних

Директива v-model пов'язує все, що в ній записано, з відповідною властивістю об'єкта "data". Це означає, що щоразу, коли ми записуємо щось усередині вхідних даних, властивість "message" об'єкта "data" автоматично оновлюється до рівня цього значення, і навпаки. Цей процес робить керування станом у компонентах набагато комфортнішим та менш трудомістким. У наступних розділах роботи детально пояснюється, як це працює. У цьому прикладі ми також зіткнулися з новою концепцією - синтаксисом "Mustache" або подвійними фігурними дужками.

Слово "message" всередині цих дужок замінюється значенням відповідної властивості в об'єкті "data". Таким чином, щоразу, коли значення властивості "message" змінюється, {{ message }} також замінюється цим значенням. На рисунку 1.8 слово під вхідними даними показує його значення, яке є результатом цього процесу. Процес реактивності можна пояснити так. Директива V-model елемента введення дозволяє відповідній властивості в об'єкті "data" оновлюватися тим самим значенням введення щоразу, коли користувачі вводять щось у

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

поле введення. Спостерігаючи за зміною властивості в об'єкті "data", спостерігач компонента потім запускає повторний рендеринг, щоб оновити представлення найновішими даними. Саме тому користувачі завжди бачать з прикладу, що вони ввели у поле введення в абзаці нижче. Vue.js сам по собі може забезпечити багато вражаючих функцій, але розробники можуть зробити набагато більше з іншими бібліотеками Vue.js. Ці бібліотеки включають Vuex, Vue Router, Vue Server Renderer та багато інших бібліотек, але в роботі представлені лише Vuex та Vue Router.

Vuex

Vuex, як описано на його офіційному сайті, - це шаблон керування станами та бібліотека для застосунку Vue.js. Він створює центральне сховище, до якого може отримати доступ будь-який компонент застосунку. Щоб забезпечити краще керування станом та безпеку, розробники можуть створювати правила та методи, які змінюють стани сховища Vuex [15]. Цей процес гарантує, що дані змінюються лише контрольованим чином та за заданим шаблоном, щоб уникнути конфліктів даних та небажаної поведінки. Зазвичай компоненти Vue.js передають дані іншим компонентам через властивості (props). Але коли застосунок зростає, це може стати занадто складним та важким для обслуговування, а також займає більше часу. Vuex може допомогти розробникам у цій ситуації. Vuex складається з трьох основних частин: стану, представлення та дій. Стан – це дані програми, які служать джерелом істини. Представлення – це місце, де ми можемо бачити дані зі стану, а також може бути місцем, де ми можемо маніпулювати станом. Дії – це можливі способи, створені розробниками для примусового впровадження передбачуваних змін стану. Дії зазвичай складаються з двох кроків: відправлення дії та здійснення мутації. Дія використовується для виконання довільних асинхронних операцій, які можуть бути викликом API до сервера в бекенді програми для отримання, оновлення або видалення даних. Після завершення операції метод дії вирішує затвердити мутацію, яка відповідає відповідям сервера. Мутації – це місце, де змінюються стани Vuex. Діаграма на рисунку 1.9 нижче демонструє процес роботи Vuex у програмах Vue.js.

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

Ще однією чудовою бібліотекою, створеною для застосунку Vue.js, є Vue Router. Vue Router - це бібліотека маршрутизаторів, створена переважно для застосунків Vue.js.

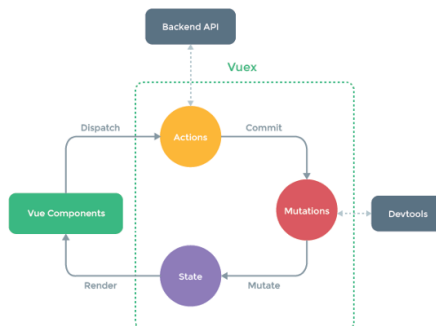


Рисунок 1.9 - Процес Vuex у діаграмі застосунку Vue.js. Vue Router

Оскільки Vue.js не має вбудованої функції маршрутизації, потрібна інша бібліотека. Ця бібліотека дозволяє розробникам налаштовувати URL-адреси своїх застосунків, а також контролювати безпеку навігації. Наприклад, розробники можуть перевірити, чи користувач увійшов у систему, щоб дозволити йому перейти до URL-адреси панелі інструментів застосунку, що в іншому випадку заборонено.

На офіційному сайті Vue Router перелічені такі функції, доступні в їхньому продукті:

- Вкладене відображення маршрутів/видів
- Модульна, компонентна конфігурація маршрутизатора
- Параметри маршруту, запит, шаблони підстановки
- Переглянути ефекти переходу
- Точне керування навігацією
- Посилання з автоматично активними CSS-класами
- Режим історії HTML5 або режим хешування
- Налаштовувана поведінка прокручування

Отже, Vue.js є одним із найзручніших та найефективніших сучасних JavaScript-фреймворків для розробки вебзастосунків. Його основними

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

перевагами є простота вивчення, зрозумілий синтаксис, висока продуктивність та гнучкість у використанні. Завдяки компонентному підходу та реактивній системі оновлення даних Vue.js дозволяє створювати швидкі, масштабовані та зручні користувацькі інтерфейси.

У порівнянні з Angular та React, Vue.js поєднує простоту інтеграції та достатню функціональність для реалізації як невеликих, так і середніх за складністю проектів [16]. Невисокий поріг входження робить його особливо привабливим для початківців, а сучасна екосистема та підтримка спільноти забезпечують широкі можливості для подальшого розвитку застосунків.

Використання Vue.js у розробці систем планування дозволяє створювати інтерактивні SPA-додатки з високою швидкодією, адаптивним дизайном та зручною взаємодією користувача з інтерфейсом. Саме тому Vue.js є доцільним вибором для створення сучасних вебзастосунків, орієнтованих на ефективність, простоту підтримки та якісний користувацький досвід.

1.4 Висновки по розділу

У першому розділі було розглянуто теоретичні основи розробки веб-додатків на базі Vue.js. Проаналізовано принципи створення односторінкових застосунків, особливості сучасних JavaScript-фреймворків та механізми двостороннього зв'язування даних. Визначено переваги використання Vue.js для розробки інтерактивних веб-додатків, зокрема простоту інтеграції, компонентний підхід та високу продуктивність. Отримані результати створюють теоретичну основу для подальшого проектування та реалізації додатку планування.

.

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. МЕТОДИ ТА ІНСТРУМЕНТИ РЕАЛІЗАЦІЇ ДОДАТКУ ПЛАНУВАННЯ

2.1 Порівняльний аналіз сучасних JavaScript-фреймворків

Порівняно з найбільшими конкурентами в сучасному просторі JavaScript-фреймворків, Angular та React (див. рис.2.1), Vue, як кажуть, має чудову загальну продуктивність завдяки своєму керованому розміру, чудовій документації, найнижчій початковій кривій навчання та найпростішій інтеграції з існуючими проектами завдяки своєму поступово адаптованому набору функцій.

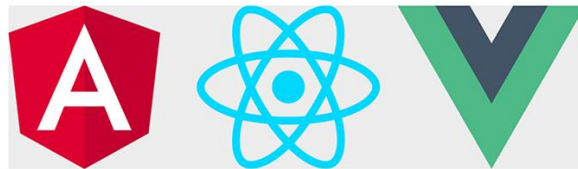


Рисунок 2.1 - Фреймворки JavaScript Angular, React та Vue

Це зроблено за задумом, оскільки Vue використовує прості шаблони на основі HTML разом із декларативним рендерингом для вираження свого інтерфейсу користувача, а це означає, що будь-який дійсний HTML також є дійсним шаблоном Vue. React, з іншого боку, використовує JSX, розширення синтаксису, подібне до XML, у функціях рендерингу для досягнення того ж самого, що вводить новий синтаксис у рівняння [17]. Те саме стосується і порівняння Vue з Angular, оскільки, на відміну від Angular, використання TypeScript з Vue все ще є необов'язковим, незважаючи на те, що Vue 3 написаний на TypeScript. Це дає Vue невелику стратегічну перевагу над його прямими конкурентами, оскільки його використання не вимагає додаткового впровадження синтаксису. (Порівняння з іншими фреймворками - див. також)

Продуктивність

Vue також рекламується як легкий фреймворк з точки зору його розміру,

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

що, безумовно, вірно, враховуючи, що вага фреймворку становить приблизно 80 КБ, тоді як React та Angular мають більший розмір, приблизно 100 КБ та 500 КБ відповідно [18]. Різниця в розмірі також означає, що Vue має найкоротший середній час запуску та найменше виділення пам'яті з усіх трьох, за ним слідує React, тоді як Angular може бути майже вдвічі повільнішим при запуску та споживати приблизно вдвічі менше пам'яті.

Щодо загальної продуктивності, Vue та React зазвичай можуть перевершити Angular у застосунках, де потрібне постійне повторне відображення вигляду, оскільки вони обидва можуть використовувати свою віртуальну DOM, тоді як Angular повинен модифікувати та оновлювати реальну DOM. Однак, Angular, як правило, є найшвидшим з трьох в операціях, які потребують маніпулювання реальною DOM, завдяки більш ефективному зв'язуванню даних.

Популярність

Оцінити популярність JavaScript-фреймворків може бути складно, оскільки не існує єдиної чіткої метрики для визначення фактичної популярності фреймворку, але інтерпретація сукупних результатів тенденцій завантаження npm-пакетів, опитувань розробників Stack Overflow, зірок GitHub та списків відкритих вакансій може дати нам досить гарне уявлення про популярність певного фреймворку [19].

За допомогою npm-stat.com, веб-сайту, який може створювати діаграми завантажень для npm-пакетів, можна визначити, що завантаження здійснені між груднем 2018 року та у грудні 2020 року показники завантажень для всіх трьох вибраних фреймворків зростають: React лідирує з понад 664 мільйонами завантажень, за ним йде Vue з понад 134 мільйонами завантажень, а Angular посідає третє місце з майже 48 мільйонами завантажень.

Згідно з опитуванням розробників Stack Overflow, Vue та React є найулюбленішими та найбажанішими веб-фреймворками, тоді як Angular посідає четверте місце за популярністю та третє місце за популярністю серед дванадцяти веб-фреймворків, перелічених в опитуванні (Результати опитування розробників Stack Overflow).

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

Кількість зірок, наданих проєкту на GitHub, може бути використана як єдиний показник, намагаючись визначити популярність фреймворку. Це також перше порівняння, де Vue може випередити як React, так і Angular, причому Vue має понад 177 000 зірок, тоді як React та Angular мають приблизно 161 000 та 69 000 зірок відповідно. (Репозиторій GitHub для Angular 0; репозиторій GitHub для React; репозиторій GitHub для Vue 2)

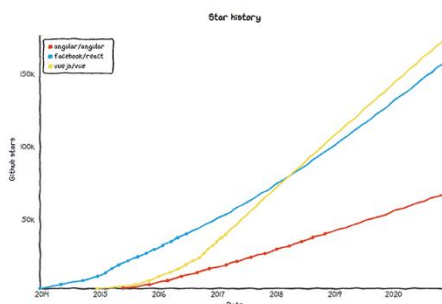


Рисунок 2.2 - Історія Angular, React та Vue на GitHub

Vue також є найновішим фреймворком з трьох, а це означає, що він набирає популярності серед розробників швидше, ніж React та Angular протягом свого існування (див. рис. 2.2), навіть з урахуванням зірок GitHub для AngularJS, проекту-попередника Angular (репозиторій GitHub для AngularJS). Однак слід зазначити, що кількість зірок для Vue базується на Vue 2, а не на Vue 3, яка є останньою версією, що має приблизно 20 000 зірок на момент написання. Це пояснюється тим, що Vue 3 є повністю переписаним кодом Vue на TypeScript, а це означає, що для збереження старої простої кодової бази Vue 2 на основі JavaScript знадобився новий репозиторій. (репозиторій GitHub для Vue 3)

Порівнюючи відкриті пропозиції роботи між цими трьома фреймворками, стає очевидним, що, незважаючи на очевидну популярність Vue серед розробників, React та Angular мають набагато більший реальний попит на роботу, ніж Vue. Причина цього явища, ймовірно, пов'язана з тим, що Vue все ще є відносно новим фреймворком порівняно з React та Angular, які мали кілька років більше, щоб закріпити свій статус та базу користувачів у менш конкурентному середовищі, або з тим фактом, що Vue не має великого та відомого прихильника, який

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

фінансує його постійний розвиток, як Facebook та Google у випадку React та Angular відповідно.

Порівняльний аналіз сучасних JavaScript-фреймворків Angular, React та Vue.js є актуальним питанням у сфері веброзробки, оскільки саме ці технології сьогодні найчастіше використовуються для створення сучасних клієнтських веб-застосунків [20]. Кожен із цих інструментів має власну архітектуру, підхід до розробки, переваги та сфери застосування, що впливає на вибір технології залежно від вимог проєкту та досвіду команди розробників.

Angular є повноцінним фреймворком, розробленим компанією Google. Він надає готову структуру для побудови великих і складних застосунків, включаючи вбудовані інструменти для маршрутизації, роботи з HTTP-запитами, формами та тестуванням. Angular використовує мову TypeScript та підтримує двостороннє зв'язування даних, що полегшує синхронізацію між інтерфейсом і логікою програми. Основною перевагою Angular є висока структурованість та придатність для корпоративних систем, однак через велику кількість вбудованих можливостей цей фреймворк має складніший процес навчання та вищі вимоги до продуктивності.

React, створений компанією Meta, технічно є не фреймворком, а бібліотекою для побудови користувацьких інтерфейсів. React базується на компонентному підході та використовує Virtual DOM для швидкого оновлення сторінок і покращення продуктивності. Однією з головних переваг React є гнучкість, адже розробники можуть самостійно обирати додаткові бібліотеки для маршрутизації, управління станом та інших функцій. React має найбільшу спільноту серед frontend-технологій, значну кількість готових рішень та високий попит на ринку праці. Недоліком є необхідність самостійного налаштування багатьох інструментів, що може ускладнювати розробку великих проєктів.

Vue.js є прогресивним JavaScript-фреймворком, створеним Еваном Ю. Він поєднує простоту Angular та гнучкість React, забезпечуючи зручний синтаксис і невисокий поріг входження для нових розробників. Vue.js використовує реактивну систему оновлення даних та компонентний підхід, що дозволяє швидко

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

створювати інтерактивні інтерфейси. Основними перевагами Vue є легкість інтеграції, хороша продуктивність та простота підтримки коду. Цей фреймворк часто застосовується для невеликих і середніх проєктів, а також стартапів, де важливими є швидкість розробки та гнучкість архітектури.

Порівнюючи продуктивність цих технологій, можна зазначити, що React та Vue.js демонструють високу швидкість роботи завдяки використанню Virtual DOM, тоді як Angular через складнішу архітектуру потребує додаткової оптимізації для великих за стосунків [21]. З точки зору навчання Vue.js вважається найпростішим для початківців, React має середній рівень складності, а Angular - найвищий через використання TypeScript, RxJS та складнішої структури проєкту.

Також важливим критерієм є популярність та підтримка спільноти. React залишається найпоширенішою frontend-технологією у світі та має найбільший ринок праці, Angular активно використовується у корпоративному секторі, а Vue.js швидко набирає популярність серед стартапів та невеликих команд розробників.

Отже, вибір між Angular, React та Vue.js залежить від масштабу проєкту, вимог до продуктивності, досвіду команди та необхідного рівня гнучкості. Angular доцільно використовувати для великих корпоративних систем, React - для масштабованих і динамічних вебзастосунків із широкими можливостями кастомізації, а Vue.js - для швидкої та зручної розробки сучасних інтерфейсів із мінімальними витратами часу на навчання.

2.2 Концепція та функціональні можливості додатку планування

Середовище виконання JavaScript з відкритим кодом, яке дозволяє виконувати код поза браузером (репозиторій GitHub для Node.js). Node.js постачається в комплекті з менеджером пакетів npm, який можна використовувати для керування локальними залежностями проєкту, а також глобальними інструментами JavaScript, такими як Vue CLI. Для використання навчального інструменту потрібне встановлення Node.js версії 8.9 або вище, оскільки він є залежністю Vue

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

CLI, тоді як для найкращої сумісності рекомендується оновлення до останньої версії LTS.

Альтернативний менеджер пакетів, який можна використовувати замість npm. Yarn пропонує кілька переконливих функцій та більш лаконічний синтаксис порівняно з npm. За допомогою Yarn пакети можна встановлювати через локальне кешування, що забезпечує швидшу установку та можливість встановлювати кешовані пакети офлайн. Однак, використання як npm, так і Yarn в одному проекті слід уникати через вирішення проблем узгодженості, спричинених несинхронізованими файлами блокування [22]. Хоча встановлення Yarn не є обов'язковим для використання навчального інструменту або інтерфейсу командного рядка Vue, його рекомендується використовувати головним чином через його більш лаконічний синтаксис порівняно з npm.

Код Visual Studio. Безкоштовний редактор вихідного коду, створений Microsoft для всіх основних операційних систем, включаючи Windows, Linux та MacOS (див. рис. 2.3). Він підтримує такі функції, як підсвічування синтаксису, інтелектуальне автодоповнення коду, налагодження та вбудований Git. Він також дозволяє використовувати користувацькі комбінації клавіш, персоналізувати інтерфейс користувача за допомогою тем та додавати функціональність за допомогою розширень.

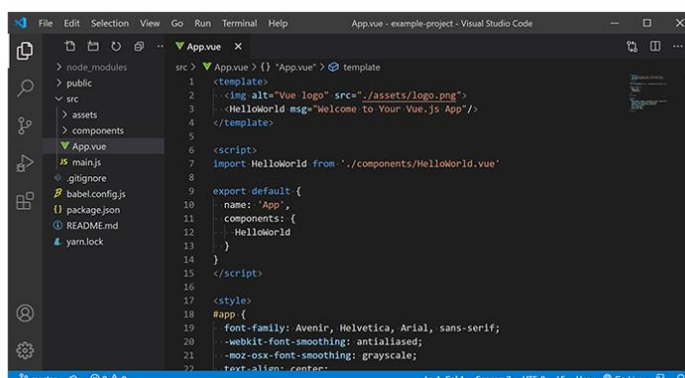


Рисунок 2.3 - Код Microsoft Visual Studio

Згідно з опитуванням розробників Stack Overflow 2019 року, Visual Studio Code є найпопулярнішим інструментом середовища розробки не лише для веб-

					БР.ІП - 94.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		31

розробки, але й для SRE та DevOps, тоді як 0,4% розробників мобільних пристроїв все ще надають перевагу Android Studio (результати опитування розробників Stack Overflow). Хоча встановлення Visual Studio Code не є обов'язковим для використання навчального інструменту або інтерфейсу командного рядка Vue, його рекомендується, оскільки його потужні функції та покращення якості життя можуть значно покращити загальний DX порівняно з більш традиційним редактором вихідного коду.

Vetur. Безкоштовне розширення Visual Studio Code, створене Pine Wu, яке пропонує підтримку Vue SFC з такими функціями, як підсвічування синтаксису, інтелектуальне автодоповнення коду, налагодження та перевірка помилок. Хоча встановлення Vetur не є обов'язковим для використання навчального інструменту або Vue CLI, рекомендується використовувати Visual Studio Code як редактор вихідного коду, оскільки без нього не можна забезпечити підтримку Vue SFC, що негативно впливає на загальний DX.

Інтерфейс командного рядка Vue. Vue CLI – це стандартний інструментарій для розробки на Vue, який постачається з готовою підтримкою Babel, TypeScript, PWA, Vue Router, Vuex, CSS Pre-Processors та ESLint, а також підтримує модульне та наскрізне тестування. Vue CLI також має додатковий графічний інтерфейс, який можна використовувати для створення та керування проектами.

Для використання Vue CLI потрібне встановлення глобального пакета через менеджер пакетів, оскільки це дозволить викликати Vue CLI через командний рядок. Vue CLI можна встановити глобально за допомогою npm, використовуючи команду `npm install -g @vue/cli`, або за допомогою Yarn, використовуючи команду `yarn global add @vue/cli`.

Новий проект Vue можна ініціалізувати, викликавши CLI з командного рядка Node.js або з вбудованого терміналу у Visual Studio Code за допомогою команди `vue create project`, де параметр `project` – це назва проекту, який буде створено. Після введення команди користувачеві буде запропоновано вибрати пресет. Пресет за замовчуванням включає Babel та ESLint, але користувач також

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

може вибрати додаткові функції вручну, вибравши опцію « *Вибрати функції вручну* ». Останній рекомендується використовувати під час розробки складніших застосунків для продакшену, тоді як пресет за замовчуванням можна використовувати для швидкого прототипування [23]. Доступні для вибору функції включають підтримку Babel, TypeScript, PWA, Vue Router, Vuex, препроцесорів, ESLint, а також модульного та наскрізного тестування.

Ручний вибір функцій також генерує нове запит на збереження вибраних функцій як пресетів. Збережені пресети зберігаються у JSON-файлі під назвою `.vuerc`, який можна знайти в домашньому каталозі користувача, розташованому в `%userprofile%\.vuerc` у Windows та `~/.vuerc` у системах на базі Linux. Інші можливі запити під час створення проекту включають запит про бажаний менеджер пакетів, який потрібно вказати, якщо у користувача встановлено як `npm`, так і `Yarn`, а також запит на використання дзеркала реєстру Taobao `npm` для швидшого встановлення залежностей. Усі збережені пресети можна змінити, безпосередньо модифікуючи файл `.vuerc`.

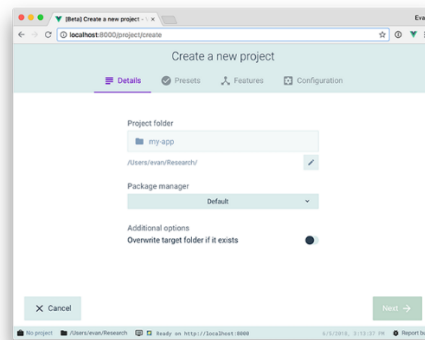


Рисунок 2.4 - Створення нового проекту за допомогою графічного інтерфейсу Vue CLI,

Або ж новий проект Vue можна створити за допомогою команди `vue ui`, яка відкриє графічний інтерфейс користувача за адресою `localhost:8000` у браузері, а потім вибере опцію « Створити новий проект тут » на вкладці «Створити ». Графічний інтерфейс користувача проведе вас через процес ініціалізації проекту (див. рис. 2.4).

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

Службу CLI можна використовувати для попереднього перегляду розробницької збірки або для створення робочої збірки проекту. За замовчуванням використання команди *npm run serve* або *yarn serve*, під час використання npm або Yarn відповідно, відбуде попередній перегляд проекту за адресою *localhost:8080* у браузері. Робочу збірку проекту можна створити за допомогою команди *npm run build* або *yarn build*, яка створить мініфіковану робочу збірку проекту в папці *dist*, що знаходиться в кореневому каталозі проекту.

Додатковий режим *Modern Mode* можна використовувати під час створення робочої збірки, додавши прапорець *-modern* в кінці відповідної команди. Це створить дві окремі збірки проекту, одна з яких орієнтована на сучасні браузери з підтримкою специфікації ECMAScript 2015, а інша - це поліфілована версія, орієнтована на застарілі браузери. Це дозволяє обслуговувати ефективнішу збірку на основі ECMAScript 2015 для сучасних браузерів, зберігаючи при цьому можливість обслуговувати менш ефективну застарілу збірку для старіших браузерів, замість того, щоб обслуговувати одну неоптимізовану збірку, яка підходить для всіх випадків використання.

Продукційна збірка, створена CLI-службою, призначена для обслуговування HTTP-сервером, а це означає, що безпосередній доступ до файлу *index.html* проекту через схему URI файлу не може бути використаний для попереднього перегляду продакшн-збірки. Для попереднього перегляду продакшн-збірки локально можна використовувати статичний файловий сервер на базі Node.js, такий як *serve*. Цього можна досягти за допомогою команди *npm install -g serve* або *yarn global add serve*, яка встановить *serve* як глобальний пакет, а потім за допомогою команди *serve -s dist* для попереднього перегляду продакшн-збірки. Використання додаткового прапорця *-s* вирішує проблеми маршрутизації, що поширені під час спроби попереднього перегляду деяких збірок на базі Vue Router.

Залежно від робочого середовища, може знадобитися створення файлу *vue.config.js* у кореневому каталозі проекту та налаштування параметра *publicPath*. За замовчуванням значення *publicPath* дорівнює *'/'*, це означає, що

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

проект потрібно розгорнути в корені домену (наприклад, <https://www.exampledomain.com/>) і не працюватиме, якщо його розгорнути на підшляху (наприклад, <https://www.exampledomain.com/example-sub-path/>). Якщо проект потрібно розгорнути на підшляху, `publicPath` потрібно налаштувати відповідно до цього підшляху. Також можна зробити `publicPath` відносним значенням, встановивши його значення на `'/'`, що дозволяє розгортати проект за будь-яким публічним шляхом. Оскільки значення `publicPath` також враховується під час розробки, у більшості випадків бажано використовувати умовне значення, щоб дозволити різні значення для розробки та виробництва, на рисунку 2.5.

```
module.exports = {
  publicPath: process.env.NODE_ENV === 'production'
    ? '/example-sub-path'
    : '/',
};
```

Рисунок 2.5 Використання умовного значення для налаштування `publicPath`

Vue Loader - це завантажувач веб-пакетів, який дозволяє використовувати формат SFC та інші завантажувачі всередині SFC, що дозволяє використовувати багато додаткових функцій, таких як компіляція коду, лінтування та гаряче перезавантаження під час розробки. Усі проекти, створені за допомогою Vue CLI, постачаються з базовою попередньою конфігурацією Vue Loader, а також усіма додатковими завантажувачами, необхідними для функцій, вибраних під час процесу створення проекту (Початок роботи `nda`).

Vue SFC - це файли з розширенням `.vue`, які складаються з блоку шаблону для опису інтерфейсу користувача за допомогою HTML, блоку скрипта для додавання логіки за допомогою JavaScript та блоку стилю для стилізації компонента за допомогою CSS (див. рисунок 2.6). SFC також підтримують обмежений CSS, підсвічування синтаксису та модулі `webpack`, які дозволяють використовувати оператори `import` та `@import` у блоках скрипта та стилю відповідно (композиція одного файлу).

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

Коментарі можна використовувати в усіх трьох блоках відповідного синтаксису коментарів мови, тоді як коментарі верхнього рівня відповідають умовам коментарів HTML (Vue Single-File Component Specification).

```
<template>
  <p class='example-class'>{{ exampleProperty }} World!</p>
</template>

<script>
export default {
  data() {
    return {
      exampleProperty: 'Hello',
    };
  },
};
</script>

<style scoped>
.example-class { example-property: example-value }
</style>
```

Рисунок 2.6 - Приклад Hello World з однофайловим компонентом

Блок шаблону на основі HTML дозволяє декларативно прив'язати відрендерений DOM до базових даних блоку скрипта, що, у свою чергу, дозволяє Vue інтелектуально визначати мінімальну кількість компонентів для повторного відображення та маніпуляцій з DOM, які потрібно застосувати для врахування зміни стану. Інтерполяція тексту з синтаксисом "вусів", що позначається {{}}, є найпростішою формою зазначеного прив'язування даних і може використовуватися для динамічного встановлення тексту в блоці шаблону відповідно до відповідної властивості даних блоку скрипта. SFC може містити лише один блок шаблону одночасно (Специфікація однофайлових компонентів Vue, nd).

Блок скрипта на основі JavaScript використовується для додавання об'єкта options, який містить інформацію про дані та логіку (наприклад, властивості та методи даних) компонента. Властивість data об'єкта options – це функція, яка повертає об'єкт, який потім викликається, коли стан змінюється, і на який посиляються, коли представлення повторно відтворюється. SFC може містити лише один блок скрипта одночасно [24].

Блок стилю на основі CSS використовується для стилізації елементів у блоці шаблону та може бути інкапсульований у область видимості компонента

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

за допомогою CSS з певною областю видимості. SFC може містити кілька блоків стилю одночасно з локальними та глобальними оголошеннями CSS.

```
<style lang='scss' scoped>
  $example-variable: example-value;
  .example-class { example-property: $example-variable }
</style>
```

Рисунок 2.7 - Використання препроцесорів у блоці стилю SFC

Vue Loader дозволяє використовувати препроцесори CSS, такі як Sass, у блоці стилів SFC. Щоб використовувати препроцесор, потрібно встановити відповідний завантажувач через npm або Yarn, а також встановити атрибут lang блоку стилів. (див. Рис. 2.7) Якщо під час інтерактивних підказок процесу створення проєкту вибрано опцію використання препроцесора, ці конфігурації обробляються автоматично Vue CLI без необхідності додаткового налаштування.

```
module.exports = {
  css: {
    loaderOptions: {
      sass: {
        prependData: '$example-variable: example-value',
      },
      scss: {
        prependData: '$example-variable: example-value;',
      },
    },
  },
};
```

Рисунок 2.8 - Спільне використання змінних шляхом прямої передачі опцій завантажувачу

Під час використання препроцесора можна спільно використовувати змінні та файли між усіма SFC за допомогою відповідного параметра завантажувача, замість того, щоб явно імпортувати їх у кожен файл. Цього можна досягти, створивши необов'язковий файл vue.config.js у кореневому каталозі проєкту та передавши відповідний параметр (наприклад, prependData with Sass) завантажувачу в css.loaderOptions. Під час використання Sass слід зазначити, що хоча завантажувач налаштовано на розбір як Sass, так і SCSS, переданий параметр

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

завантажувача все одно повинен дотримуватися синтаксичних домовленостей відповідного синтаксису. (Див. Рисунок 2.8)

Vue Loader дозволяє використовувати атрибут `scoped` в блоці стилю SFC, що обмежить область дії CSS, оголошеного в блоці стилю, цим компонентом. Це запобігає витоку оголошень стилю батьківського компонента в дочірні компоненти. Хоча використання окремих блоків стилю для локального та глобального CSS можливе, кореневий вузол дочірнього компонента також може бути змінений з блоку стилю батьківського компонента з атрибутом `scoped` за допомогою глибокого селектора `>>>`, `/deep /` або `::v-deep`, причому два останні є псевдонімами першого, призначеними для використання з препроцесорами, які не можуть належним чином розібрати селектор `>>>` (див. рис. 2.9).

```
<style scoped>
  >>> .example-class { example-property: example-value }
  /* OR */
  /deep/ .example-class { example-property: example-value }
  /* OR */
  ::v-deep .example-class { example-property: example-value }
</style>
```

Рисунок 2.9 - Використання глибокого селектора в межах блоку стилю SFC з обмеженою областю видимості

Vue використовує директиви для декларативного зв'язування логіки з елементами DOM. Ці директиви проявляються як префіксовані атрибути HTML, які можна розбити на вигляд `<element prefix-directiveID[:argument][='expression']></element>`, де параметр префікса завжди є `v`, параметр `directiveID` визначає тип дії, яку повинна виконувати директива, параметр `argument` визначає, наприклад, тип події або атрибута, на який елемент повинен реагувати або з яким повинен бути пов'язаний відповідно, тоді як параметр `expression` посилається на дані або логіку (наприклад, властивість даних або метод), що використовуються для обробки директиви під час виконання. Параметри, вкладені в `[]`, є або необов'язковими, або ситуативними залежно від заданої директиви.

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

Директиви v-if, v-else та v-else-if можна використовувати для умовного відображення елементів, які потрібно відобразити лише в певних ситуаціях. Цього можна досягти, передаючи вираз директиві v-if , що призводить до відображення або ігнорування елемента під час компіляції, коли вираз є істинним або хибним відповідно.

Елемент з директивою v-if може бути одразу після елемента з директивою v-else або v-else-if , або якщо присутні елементи з обома директивами, елемент з директивою v-else-if повинен передувати елементу з директивою v-else (див. рис. 2.10). Директива v-else не очікує виразу та може бути використана для рендерингу альтернативного елемента, коли директива v-if попереднього елемента визначається як неправдива, тоді як директива v-else-if робить те саме, але очікує, що власний вираз виявиться правдивим для рендерингу.

```
<template>
  <p v-if='exampleProperty === exampleValue'></p>
  <p v-else-if='exampleProperty === exampleValue'></p>
  <p v-else></p>
</template>

<script>
export default {
  data() {
    return {
      exampleProperty: exampleValue,
    };
  },
};
</script>
```

Рисунок 2.10 - Умовне рендеринг елементів за допомогою директиви v-if

Директива v-bind може використовуватися для динамічного зв'язування атрибутів з елементами та властивостей (props) з дочірніми компонентами. Директива очікує аргумент, що представляє атрибут (наприклад, src або key) або властивість (prop) та вираз, що представляє значення зазначеного атрибута або властивості (attribute). (див. рис. 2.11) Як варіант, можна передати об'єкт атрибутів до директиви, що дозволяє пропустити частину аргументу директиви. Класи HTML також можна динамічно пов'язувати з атрибутом class , використовуючи синтаксис на основі аргументів та передаючи директиві об'єкт, що містить пари

властивостей класу та даних, при цьому істинність властивості даних визначає наявність класу. Директива `v-bind` також має скорочення, яке можна використовувати для пропускання ключового слова `v-bind`, але не можна використовувати під час прив'язки об'єкта атрибутів через синтаксичну несумісність [25].

```
<template>
  <img :src='example.src' :alt='example.alt' :class='{ exampleClass: exampleProperty }' />
</template>

<script>
import exampleSrc from '@assets/example.png';

export default {
  data() {
    return {
      example: { src: exampleSrc, alt: 'exampleAlt' },
      exampleProperty: exampleValue,
    };
  },
};
</script>

<style scoped>
.exampleClass { example-property: example-value }
</style>
```

Рисунок 2.11 - Прив'язка атрибутів до елементів за допомогою директиви `v-bind`

Директиву `v-for` можна використовувати для ітерації по елементах масиву або властивостях об'єкта та їх відображення. Директива очікує вираз у формі `example` у `examples`, де параметр `example` служить псевдонімом для поточної ітерації, параметр `in` служить роздільником, а параметр `examples` - це ітерований об'єкт. (див. рис. 2.12) Необов'язковий параметр індексу у формі (приклад, `index`) у прикладах може бути використаний для доступу до індексу ітерації, тоді якщо-межувач `in` `de` може бути замінений роздільником, щоб відповідати оператору `for...of` JavaScript.

Під час використання директиви `v-for` унікальний атрибут *ключа* має бути пов'язаний з кожним прямим дочірнім елементом елемента, для якого він використовується, щоб допомогти алгоритму віртуального DOM Vue мінімізувати переміщення елемента під час повторного рендерингу, тоді як використання директиви `v-if` для того самого елемента слід уникати через її вищий пріоритет, що призводить до неможливості доступу до змінних з області дії директиви `v-for`.

Директива `v-on` може бути використана для приєднання слухача подій до

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

елемента, що дозволяє виконання JavaScript у відповідь на дію користувача. Директива очікує аргумент, що представляє тип події (наприклад, клік або натискання клавіші), та вираз, що представляє логіку, яку має викликати ця подія (наприклад, метод або вбудований оператор). (див. рис. 2.13)

```
<template>
  <p v-for='example in examples' :key='example.id'>{{ example.property }}</p>
</template>

<script>
export default {
  data() {
    return {
      examples: [
        { id: exampleValue, property: exampleValue },
        { id: exampleValue, property: exampleValue },
      ],
    };
  },
};
</script>
```

Рисунок 2.12 - Ітеративне відображення елементів масиву за допомогою директиви v-for

Директива v-on підтримує додаткові модифікатори подій, які дозволяють автоматично викликати методи інтерфейсу подій (наприклад, preventDefault та stopPropagation) під час спрацьовування події шляхом додавання відповідного постфікса (наприклад, .prevent та .stop) до директиви.

```
<template>
  <button @click.stop='exampleMethod' id='example-id'></button>
</template>

<script>
export default {
  methods: {
    exampleMethod(event) {
      console.log(event.target.id);
    },
  },
};
</script>
```

Рисунок 2.13 - Обробка подій за допомогою директиви v-on

Також підтримуються модифікатори клавіш для подій клавіатури, які

дозволяють прослуховувати певні клавіші шляхом перетворення відповідного значення клавіші (наприклад, Enter, PageDown) у формат kebab-case (наприклад, .enter, .page-down) та додавання перетвореного значення як постфікса до директиви. Директива v-on також має скорочений символ @, який можна використовувати для пропускання ключового слова v-on та символу ; перед аргументом, що забезпечує стислий синтаксис.

2.3 Реалізація маршрутизації та навігації за допомогою Vue Router

Офіційна реалізація маршрутизатора, спеціально розроблена для Vue, яка забезпечує динамічну маршрутизацію на основі компонентів та різні режими історії для різних випадків використання [26]. Оскільки програми, створені за допомогою Vue CLI складаються з компонентів, використання Vue Router для зіставлення компонентів з певними маршрутами є логічним та потужним способом підходу до навігації у Vue SPA

```
import { createRouter, createWebHashHistory } from 'vue-router';

export default createRouter({
  history: createWebHashHistory(),
  routes: [
    {
      path: '/',
      component: () => import('../views/ExampleComponent'),
    },
    {
      path: '/example-sub-path',
      component: () => import('../views/ExampleComponent'),
    },
  ],
});
```

Рисунок 2.14 Створення екземпляра Vue Router

Щоб використовувати Vue Router, потрібно створити екземпляр маршрутизатора та вставити його в кореневий компонент проекту, що дозволить отримати доступ до маршрутизатора з усіх компонентів, які є дочірніми компонентами цього компонента. Екземпляр маршрутизатора можна створити за допомогою функції createRouter та передачі об'єкта options, що містить функцію history

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

та масив `routes`, які використовуються для визначення режиму історії та конфігурацій маршруту відповідно (див. рис. 2.14). Якщо під час інтерактивних підказок процесу створення проекту вибрано опцію використання Vue Router, екземпляр маршрутизатора створюється автоматично за допомогою Vue CLI та може бути знайдений у `src/router/index.js`.

Кореневий компонент проекту можна вказати за допомогою функції `createApp` та передачі компонента (наприклад, `App.vue`) як опції, після чого можна одразу ж здійснити ін'єкцію маршрутизатора за допомогою функції `use` та передачі створеного маршрутизатора як опції (див. рис. 2.15).

```
import { createApp } from 'vue';
import App from '@/App.vue';
import router from '@/router';

createApp(App).use(router).mount('#app');
```

Рисунок 2.15 - Впровадження екземпляра Vue Router

Якщо під час інтерактивних підказок процесу створення проекту вибрано опцію використання Vue Router, маршрутизатор автоматично ін'єктується до кореневого компонента проекту за допомогою Vue CLI, а конфігурацію можна знайти в `src/main.js`.

Vue Router підтримує різні режими історії, а саме: режим хешування та режим HTML, які мають свої переваги та недоліки. Режим хешування, створений за допомогою функції `createWebHashHistory`, використовує символ `#` перед шляхом (наприклад, <https://www.exampledomain.com/#/example-sub-path>), що дозволяє передавати маршрут у веб-застосунку або веб-сайті без надсилання на сервер. Це тимчасовий шлях, який негативно впливає на SEO, але дозволяє обійти необхідні конфігурації на стороні сервера (Різні режими історії та інше). Режим HTML, створений за допомогою функції `createWebHistory`, дозволяє використовувати звичайну SEO-дружню URL-адресу (наприклад, <https://www.exampledomain.com/example-sub-path>), але вимагає додаткової конфігурації на стороні сервера, щоб запобігти

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

помилкам 404 під час спроби доступу до певних маршрутів веб-застосунку або веб-сайту безпосередньо (Різні режими історії та інше).

```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.html$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.html [L]
</IfModule>
```

Рисунок 2.16 - Конфігурація резервного маршруту Apache catch-all

Цього можна досягти, реалізувавши універсальний резервний маршрут до сервера, який обслуговуватиме файл `index.html` проекту у разі виникнення помилки 404 (див. рис. 2.16). Однак це не ідеальне рішення, оскільки сервер не буде більше повідомляти про помилки 404 взагалі, якщо певний маршрут не існує. Щоб вирішити цю проблему, слід реалізувати універсальний маршрут у формі `{path: '/:pathMatch(*)', component: () => import('../views/ExampleComponent')}` для обслуговування спеціальної веб-сторінки 404 (різні режими історії та інші).

Крім того, якщо проект потрібно розгорнути на підшляху (наприклад, <https://www.exampledomain.com/example-sub-path/>), а не в корені домену, базова URL-адреса конфігурації сервера (наприклад, `RewriteBase /` (при використанні Apache) має відображати значення опції `publicPath` файлу `vue.config.js` (наприклад, `RewriteBase /example-sub-path/` при використанні Apache) та має бути передано як параметр функції, що використовується для створення режиму історії. `router-link` можна використовувати для створення посилань, які дозволяють змінювати URL-адресу веб-сторінки без її перезавантаження, використовуючи атрибут `to` для визначення шляху, тоді як користувацький компонент `router-view` можна використовувати для відображення компонента, що відповідає зазначеному шляху (див. рис. 2.17).

					БР.ІП - 94.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		44

```

<template>
  <button @click='exampleMethod'></button>
  <router-link to='/example-sub-path'></router-link>
  <router-view />
</template>

<script>
export default {
  methods: {
    exampleMethod() {
      this.$router.push('/example-sub-path');
    },
  },
};
</script>

```

Рисунок 2.17 - Використання Vue Router для переходу до іншого шляху

До екземпляра маршрутизатора можна отримати доступ через метод `this.$route`, а потім використовувати його разом із JavaScript `push` для переходу до іншого шляху, тоді як до динамічних властивостей поточного шляху можна отримати доступ, повернувши параметр через обчислювану властивість за допомогою `this.$route.params`.

Офіційний шаблон керування станом, подібний до Flux, спеціально розроблений для Vue, який забезпечує однаковий стан для всіх компонентів у застосунку [27]. Vuex сприяє односторонньому потоку даних, що забезпечує передбачувані зміни станів та спрощує зв'язок між компонентами, що призводить до більш зручного обслуговування коду в проєктах середнього та великого розміру, але за рахунок більшої кількості концепцій для вивчення та шаблонного коду для написання.

Щоб використовувати Vuex, потрібно створити екземпляр сховища та вставити його в кореневий компонент проєкту, що дозволить отримати доступ до стану сховища з усіх компонентів, які є дочірніми компонентами цього компонента. Екземпляр сховища можна створити за допомогою функції `createStore` та передачі об'єкта `options`, що містить об'єкт стану, об'єкт `getters`, об'єкт `mutations` та об'єкт `actions`, які використовуються для зберігання стану, зберігання обчисленого стану, синхронної мутації стану та асинхронної мутації стану відповідно (див. рис. 2.18). Якщо під час інтерактивних підказок процесу створення проєкту вибрано опцію використання Vuex, екземпляр сховища

створюється автоматично за допомогою Vue CLI та може бути знайдений у `src/store/index.js`.

Щоб використовувати Vuex, потрібно створити екземпляр сховища та вставити його в кореневий компонент проекту, що дозволить отримати доступ до стану сховища з усіх компонентів, які є дочірніми компонентами цього компонента. Екземпляр сховища можна створити за допомогою функції `createStore` та передачі об'єкта `options`, що містить об'єкт стану, об'єкт `getters`, об'єкт `mutations` та об'єкт `actions`, які використовуються для зберігання стану, зберігання обчисленого стану, синхронної мутації стану та асинхронної мутації стану відповідно (див. рис. 2.18). Якщо під час інтерактивних підказок процесу створення проекту вибрано опцію використання Vuex, екземпляр сховища створюється автоматично за допомогою Vue CLI та може бути знайдений у `src/store/index.js`

```
import { createStore } from 'vuex';

export default createStore({
  state: {
    exampleProperty: exampleValue,
  },
  getters: {
    exampleGetter(state) {
      return state.exampleProperty + exampleValue;
    },
  },
  mutations: {
    exampleMutation(state, payload) {
      state.exampleProperty = payload.exampleProperty;
    },
  },
  actions: {
    exampleAction(context) {
      context.commit('exampleMutation');
    },
  },
});
```

Рисунок 2.18 - Створення екземпляра сховища Vuex

Кореневий компонент проекту можна вказати за допомогою функції `createApp` та передачі компонента (наприклад, `App.vue`) як опції, після чого можна одразу ж виконати ін'єкцію сховища за допомогою функції `use` та передачі створеного сховища як опції (див. рис. 2.19). Якщо під час інтерактивних підказок процесу створення проекту вибрано опцію використання Vuex, сховище

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

автоматично ін'єктується до кореневого компонента проекту за допомогою інтерфейсу командного рядка Vue, а конфігурацію можна знайти в `src/main.js`.

До екземпляра сховища можна отримати доступ з компонента, повернувши стан або властивість `getter` через обчислювану властивість з `this.$store.state` та `this.$store.getters` відповідно, тоді як фіксація мутації або відправлення дії може бути досягнута за допомогою методу з `this.$store.commit` та `this.$store.dispatch` відповідно.

```
import { createApp } from 'vue';
import App from '@/App.vue';
import store from '@/store';

createApp(App).use(store).mount('#app');
```

Рисунок 2.19 - Впровадження екземпляра сховища Vuex

`mapState` та `mapGetters` можна використовувати для зіставлення стану сховища або властивостей геттерів з локальними обчислюваними властивостями компонента, наприклад, `this.$store.state.exampleProperty` можна зіставити з `this.exampleProperty` (див. рис. 2.20), тоді як допоміжні функції `mapMutations` та `mapActions` можна використовувати для зіставлення мутацій та дій з локальними методами компонента `mut`, наприклад, `this.$store.commit('exampleMutation')` можна зіставити з `this.exampleMutation()`, що значно зменшує кількість шаблонних елементів, коли потрібно кілька властивостей сховища в одному компоненті.

Розділ «Омега» слугує завершальним розділом навчального інструменту, і його головна мета - запропонувати обґрунтовані рекомендації щодо того, як користувач може розширити свої знання про Vue за межі концепцій, що розглядаються в навчальному інструменті. Це концепції, які не є необхідними для стандартної розробки на Vue, але можуть бути корисними в певних ситуаціях. У цьому розділі розглядаються наступні концепції.

Розширення мови JavaScript з відкритим кодом, яке додає статичні визначення типів і дозволяє виявляти та виправляти помилки ще до запуску коду. Type

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

Script - це надмножина JavaScript, яка транскompілюється в JavaScript компілятором TypeScript або Babel. Це означає, що весь валідний код JavaScript також є валідним TypeScript, що полегшує вивчення мови з попереднім досвідом роботи з першим, оскільки ігнорування синтаксичних домовленостей TypeScript не запобігає виконанню транскompільованого коду JavaScript, а натомість призводить до помилок перевірки типів, які можна виправити пізніше, щоб зберегти цілісність коду.

```
<template>
  <p>{{ exampleComputed }}</p>
</template>

<script>
import { mapState } from 'vuex';

export default {
  computed: {
    exampleComputed() {
      return this.exampleProperty;
    },
    ...mapState([
      'exampleProperty',
    ]),
  },
};
</script>
```

Рисунок 2.20 - Використання допоміжної функції mapState у Vuex

Згідно з опитуванням розробників Stack Overflow 2020 року, TypeScript є другою за популярністю та однією з найбажаніших мов програмування 2020 року, що свідчить про те, що TypeScript успішно зарекомендував себе як одна з провідних мов програмування для розробки веб- додатків. Як згадувалося в розділі 2.2, Vue 3 - це повністю переписана кодова база Vue на TypeScript, що дозволяє Vue забезпечити покращену підтримку TypeScript у майбутньому. Це означає, що хоча використання TypeScript з Vue все ще є необов'язковим, тепер це простіше та виправданіше, ніж будь-коли раніше [28].

Vue зазвичай використовується для розробки клієнтських веб-застосунків та веб-сайтів, створених з компонентів, які маніпулюють DOM у браузері, але те саме можна зробити на стороні сервера, відтворюючи компоненти в HTML, а потім надсилаючи їх до браузера. Порівняно з типовим SPA, переваги SSR полягають у кращому SEO, оскільки пошукові системи бачать повністю відображену

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

веб-сторінку, та швидшому часі створення контенту, оскільки веб-сторінці, відображеній за допомогою SSR, не потрібно чекати на завантаження JavaScript завантажуються та виконуються [29]. Однак SSR має деякі недоліки, включаючи обмеження розробки з певними зовнішніми бібліотеками, складніші вимоги до розгортання та більше навантаження на стороні сервера, оскільки рендеринг цілого веб-застосунку або веб-сайту потребує більше ресурсів процесора, ніж обслуговування статичних файлів SPA.

Крім того, налаштування належним чином налаштованої веб-програми або веб-сайту з візуалізацією на сервері може бути значно складнішим, ніж налаштування звичайної клієнтської SPA через кілька вимог до розгортання. Quasar Framework (згодом Quasar) – це фреймворк вищого рівня, заснований на екосистемі Vue, який має на меті оптимізувати розробку веб-програм та веб-сайтів з візуалізацією на сервері Vue. (Серверна рендерингова архітектура – й). Інші базові технології Quasar включають Capacitor, Cordova, Electron, Node.js та webpack, але досвід роботи з цими технологіями не є обов'язковим, оскільки всі вони інтегровані та налаштовані за замовчуванням (Чому Quasar? – й). Це означає, що людина з досвідом роботи з Vue на стороні клієнта, отриманим за допомогою навчального інструменту, може негайно почати вивчати Quasar, не вивчаючи жодних інших додаткових технологій, що робить Quasar привабливим та виправданим варіантом у майбутньому [30].

2.4 Висновки по розділу

У другому розділі досліджено методи та інструменти реалізації додатку планування засобами Vue.js. Проведено порівняльний аналіз сучасних JavaScript-фреймворків, визначено функціональні можливості системи та обґрунтовано вибір технологій для її реалізації. Особливу увагу приділено маршрутизації та навігації за допомогою Vue Router, що забезпечує зручну організацію структури веб-додатку. У результаті сформовано архітектурний підхід та набір інструментів для ефективної реалізації системи планування.

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. РОЗРОБКА, ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ВЕБ-ДОДАТКУ

3.1 Реалізація веб-додатку планування засобами Vue.js

У цьому підрозділі представлено це веб-застосунок. Спочатку представлено вступ до застосунку, а також його інтерфейси та функції. Після цього показано деякі інструменти, які використовувалися для створення дизайну та самого проєкту.

Розробка власного додатку планування на базі Vue.js передбачає створення сучасного вебзастосунку для організації задач, подій та управління часом. Основною метою такого проєкту є забезпечення користувачам зручного інструменту для планування щоденних справ, встановлення дедлайнів, контролю виконання завдань і підвищення особистої або командної продуктивності. Завдяки використанню Vue.js додаток може мати швидкий та інтерактивний інтерфейс, який оновлюється без перезавантаження сторінки, що значно покращує користувацький досвід.

Система може включати функції створення, редагування та видалення задач, сортування за пріоритетом або датою, а також можливість перегляду розкладу у форматі календаря. Для зручності користувачів можуть реалізовуватись нагадування про важливі події та дедлайни, а також авторизація для збереження персональних даних і синхронізації інформації між різними пристроями. Додатково застосунок може надавати статистику продуктивності, що дозволить аналізувати ефективність виконання завдань та планування часу.

Для реалізації frontend-частини зазвичай використовується Vue.js разом із такими інструментами, як Vue Router для навігації між сторінками та Pinia для управління станом додатку. Серверна частина може бути створена за допомогою Node.js і Express.js, а для зберігання даних використовуватись MongoDB або PostgreSQL. Такий технологічний стек забезпечує високу продуктивність, масштабованість та можливість подальшого розвитку системи.

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

Розробка подібного додатку є актуальною, оскільки цифрові інструменти планування дедалі більше використовуються у навчанні, бізнесі та повсякденному житті. Створення власного рішення дозволяє адаптувати функціонал під конкретні потреби користувачів, реалізувати унікальний дизайн та інтегрувати додаткові можливості, наприклад синхронізацію з Google Calendar або підтримку PWA для роботи в офлайн-режимі.

Застосунок для нагадування

Цей підрозділ зосереджений на демонстрації веб-застосунку. Застосунок розроблено та розроблено для шкільного проєкту. Також використовуються різні інструменти, доступні безкоштовно в Інтернеті, для підвищення ефективності процесу розробки. Застосунок працює на Vue.js та інших бібліотеках і фреймворках, таких як Vuex, Vue Router, Buefy, Webpack [31]. Це застосунок-нагадування, за допомогою якого користувачі можуть створювати та керувати планами або подіями у своєму повсякденному житті. Знімок екрана інтерфейсу календаря можна знайти нижче.

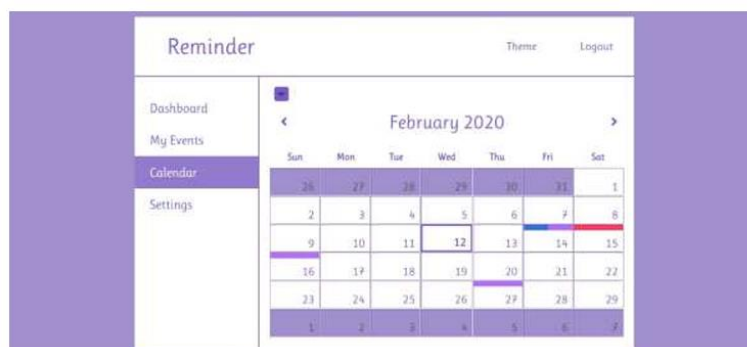


Рисунок 3.1 -Інтерфейс календаря програми нагадування

Інтерфейс календаря – це місце, де користувачі можуть знаходити створені ними події або плани у реальному календарі з його категоріями (кольоровими смужками), що відображаються на денній сітці події. Додаток також має фільтр, який допомагає користувачам фільтрувати події та плани для кращого керування та налаштування особистих уподобань.

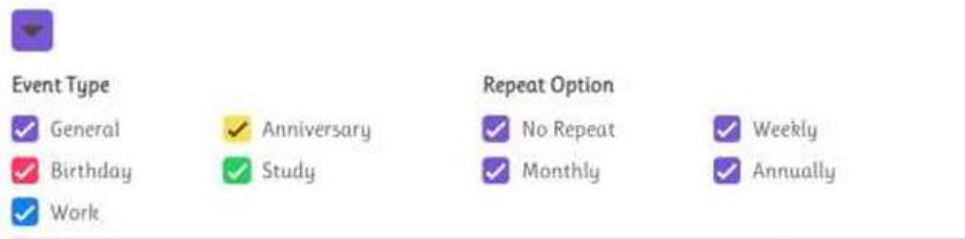


Рисунок 3.2 - Параметри фільтра календаря

Як показано на рисунку 3.2, ми бачимо, що події також мають свій тип та опцію повторення. Ці типи подій не можна змінити або налаштувати, але планується реалізувати цю функцію в пізніших версіях у майбутньому. Кожен тип події має свій колір, тому користувачі можуть легко розрізняти події, не шукаючи їх деталей. Поруч із типами подій розташовані опції повторення.

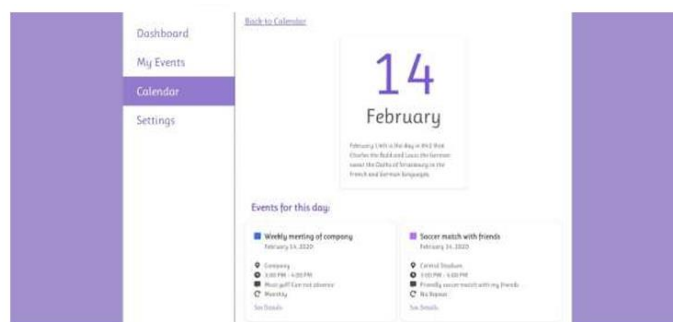


Рисунок 3.3 - Деталі дня

Ці опції стають у пригоді, коли користувачі хочуть, щоб їхні події автоматично повторювалися через однаковий період часу, наприклад, щотижня, щомісяця або щорічно. Щоденні події також доступні, але вони не відображаються в календарі, оскільки це зайве та викликає багато відволікаючих факторів, оскільки ці події з'являються щодня в календарі. Тому щоденні події відображаються в іншому місці програми, що згадується далі в цьому підрозділі. Ці доповнення додано для покращення відчуття та комфорту користувачів [32].

У режимі перегляду календаря користувачі можуть переглядати будь-які деталі та події дня. На рисунку 3.3 показано приклад інтерфейсу з деталями дня.

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

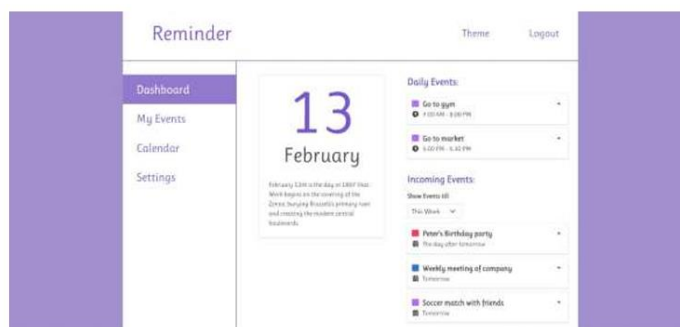


Рисунок 3.4 - Інтерфейс панелі інструментів програми «Нагадування»

На інформаційній панелі користувачі можуть бачити загальну інформацію про свої події, плани та цікавий факт дня. Щоденні події відображаються у верхньому правому куті інформаційної панелі. На інформаційній панелі також відображається список вхідних подій. Користувачі можуть вибрати, через який проміжок часу вони хочуть бачити свої події. Ці опції можуть відображати події цього тижня, наступного тижня, цього місяця або наступного місяця. Повторювані події можуть з'являтися багато разів протягом вибраного періоду, наприклад, щотижневі події, але список показує лише найближчу з цих подій. Кожна подія в цих списках є картою події, яку можна розгорнути, щоб показати більше деталей події, або закрити, щоб отримати кращий огляд списку [33]. Користувачі можуть бачити ці картки подій у багатьох місцях програми.

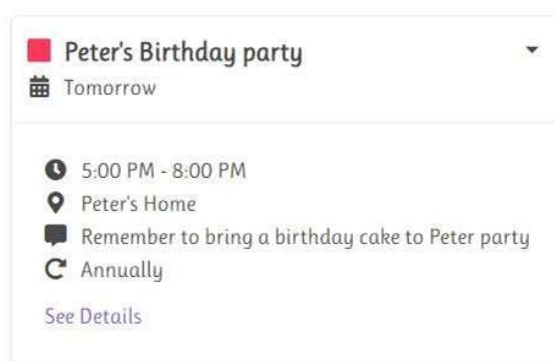


Рисунок 3.5 - Розгорнута картка події

На рисунку 3.5 показано, як виглядає розгорнута картка події. З розгорнутого вигляду картки користувачі можуть отримати доступ до сторінки з деталями події, щоб переглянути всі деталі події та змінити або видалити її.

					БР.ІП - 94.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		53

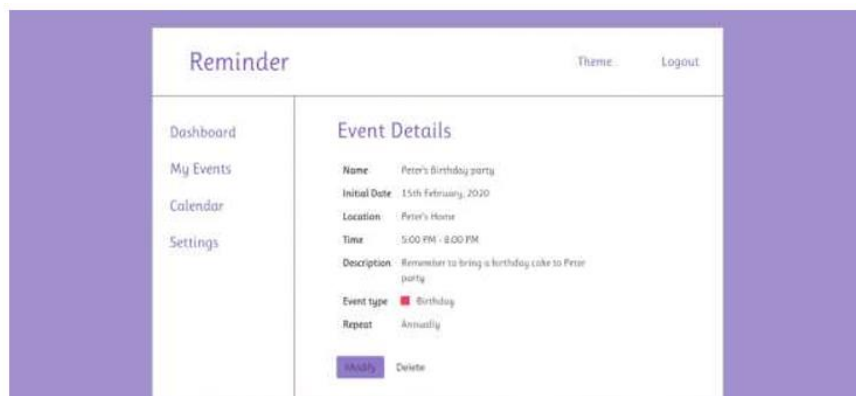


Рисунок 3.6 - Приклад деталей події

Список подій та форма:

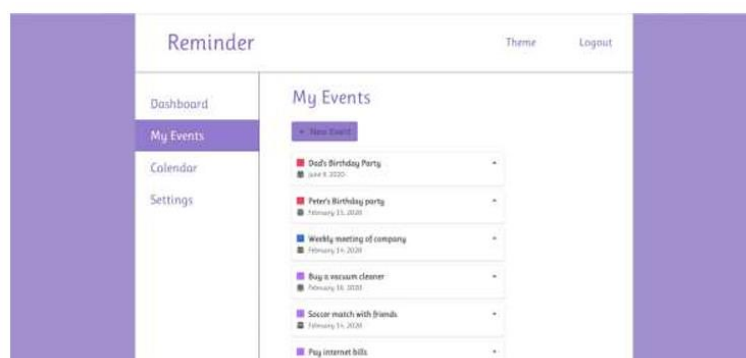


Рисунок 3.7 - Інтерфейс списків подій

Інтерфейс списку подій надає користувачам повний список подій, відсортованих за датою створення. Кожна подія тут також є карткою події, як-от та, що на панелі інструментів. Звідси користувачі можуть створювати нові події, перейшовши до іншого інтерфейсу – форми події. На рисунку 3.8 нижче показано, як виглядає форма події.

У формі події користувачі можуть заповнити інформацію про свої події та вибрати дату, час, варіант повторення та тип події. Поля місця проведення та опису можна залишити порожніми, але назва обов'язкова для створення події. Інші поля використовують значення за замовчуванням, якщо користувачі їх не змінюють [34].

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

Рисунок 3.8 - Інтерфейс форми події

Під час редагування подій користувачі також використовують цю форму, але назви та методи обробки форми відрізняються.

Налаштування користувача та програми

Спочатку користувачам потрібно створити обліковий запис, щоб користуватися цією програмою. Після входу в систему користувачі можуть надати іншу особисту інформацію, таку як ім'я та прізвище, дата народження та електронна пошта. Ця інформація не є необхідною, оскільки наразі вона не потрібна. Але планується розширити програму іншими функціями, за допомогою яких користувачі зможуть взаємодіяти один з одним. Ці функції можуть включати створення групи користувачів або створення спільних подій чи планів.

Це лише плани, і немає жодної гарантії, що вони будуть доступні в майбутньому. Ця програма нагадування також пропонує два інші налаштування. Перше - встановити перший день тижня. Це може бути понеділок або неділя, залежно від уподобань користувача. Це налаштування впливає на календар як у засобах вибору дати, так і в календарі подій програми. Інше налаштування схоже на перше, яке використовується для встановлення формату часу між 12-годинним та 24-годинним форматом. На рисунку 3.9 нижче показано інтерфейс налаштувань програми.

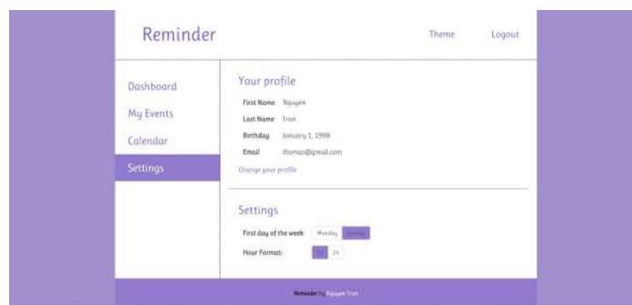


Рисунок 3.9 - Інтерфейс налаштувань користувача та програми

Це огляд деяких основних функцій програми. У наступному підрозділі демонструється деякі інструменти та допоміжні програми, які допомогли йому розробити та спроектувати програму.

Інші інструменти та допоміжні програми

Draw.io - це платформа, яка використовується для створення діаграм, схем, моделей та багато іншого. Використано цю програму для створення UML-моделі для бази даних програми [35]. Ця платформа є безкоштовною у використанні та має багато доступних шаблонів, які допомагають у процесі проектування структури.

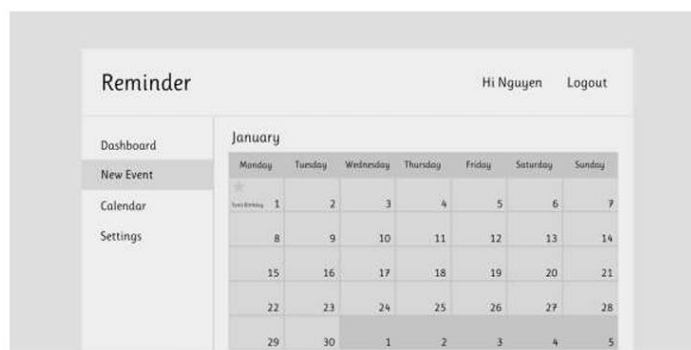


Рисунок 3.10 - Прототип інтерфейсу календаря, розроблений на Figma

Figma - це веб-додаток, відомий створенням інтерфейсів. Ця платформа працює на базі браузера, але також може бути настільним додатком у Windows та Mac OS. Figma дозволяє співпрацю, що означає, що дизайнери можуть працювати разом над створенням одного й того ж продукту. Нижче наведено чернетку інтерфейсу календаря, створеного за допомогою Figma для макета додатку.

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

Clockify - це безкоштовний трекер часу, який працює у браузері. Цей продукт також доступний на багатьох настільних комп'ютерах та мобільних пристроях. Використаноцей веб-застосунок для відстеження та ведення обліку того, скільки годин він витратив на розробку застосунку. Загальний час наразі становить близько 150 годин.

3.2 Збір, обробка та аналіз даних користувача

Для подальшого просування в дослідженні, для вивчення та пояснення досліджуваного випадку використовується технічна або кодова перспектива. Процес можна зрозуміти таким чином. Деякі особливості фреймворків Vue.js, які, як доводить це дослідження, принесуть певні переваги процесу розробки, представлені на прикладах коду з тематичного дослідження [36]. Після цього в роботі показано подібну реалізацію, яка дає той самий ефект, але тільки написана на Vanilla JavaScript або чистому JavaScript без використання фреймворків. Потім проводиться порівняння між прикладами, щоб вказати на можливі переваги використання фреймворку Vue.js у проекті. Але іноді, коли приклади реалізації неможливо навести через великий обсяг коду, або приклад був би надто складним чи невідповідним для мети дослідження чи рівня кваліфікації роботи, натомість наводяться статті та блоги для підтримки аргументації дослідження. Процеси збору та аналізу даних виконуються одночасно, щоб забезпечити більш прямі та точні порівняння та результати.

Компонент Vue.js проти його відсутності

Як було пояснено, архітектура компонентів є домінуючою рисою фреймворку Vue.js, особливо компонентів з одним файлом, де вони включають усі три фундаментальні елементи веб-розробки (HTML, CSS та JavaScript). Погляньте на приклад нижче; це код компонента **CalendarLayout** програми Reminder. Компонент можна імпортувати та використовувати так само, як і будь-який HTML-елемент, наприклад, **<CalendarTable>**.

```

1 <template>
2   <div>
3     <div v-if="showCalendar">
4       <CalendarFilter
5         v-on:input-event-types="inputEventTypes"
6         v-on:input-repeat-option="inputRepeatOption"
7       ></CalendarFilter>
8       <MonthPicker :month="month" :year="year" v-on:change-month-and-year="setMonthAndYear"></MonthPicker>
9       <CalendarTable
10        :month="month"
11        :year="year"
12        v-on:select-day="selectDay"
13        :selectedEventTypes="selectedEventTypes"
14        :selectedRepeatOptions="selectedRepeatOptions"
15      ></CalendarTable>
16    </div>
17    <DayDetail v-else :selectDay="selectDay" v-on:show-calendar="showCalendar = !showCalendar"></DayDetail>
18  </div>
19 </template>
20
21 <script>
22   import CalendarFilter from './calendarFilter/CalendarFilter'
23   import MonthPicker from './monthPicker/MonthPicker'
24   import CalendarTable from './calendarTable/CalendarTable'
25   import DayDetail from './dayDetails/DayDetails'

```

Рисунок 3.11 - Використання компонентів та зв'язок

Компоненти приносять багато зручностей та економлять величезну кількість часу розробникам, оскільки, зрештою, ці фреймворки є продуктами багатьох великих компаній, таких як Facebook, Google або колишніх співробітників Google. Логіка створення цих компонентів далеко виходить за межі знань. Тому в роботі не обговорюється, чому використання компонента Vue.js краще, ніж побудова структури компонентів за допомогою простого JavaScript. Натомість, порівняння між використанням компонентів та їх невикористанням є більш доцільним підходом. Порівняння вже не стосується використання компонента Vue.js проти побудови та використання компонента Vanilla JavaScript. Але принаймні розробники економлять багато часу, не створюючи структуру компонентів самостійно, а використовуючи її з фреймворку або бібліотеки.

Отже, як використання компонентів може принести переваги в процес розробки? Як пояснювалося в розділі теорії, компоненти – це елементи, що виготовляються на замовлення, які можна використовувати повторно. Розглянемо для прикладу картки подій, ці картки з'являються багато разів у застосунку, на багатьох сторінках.

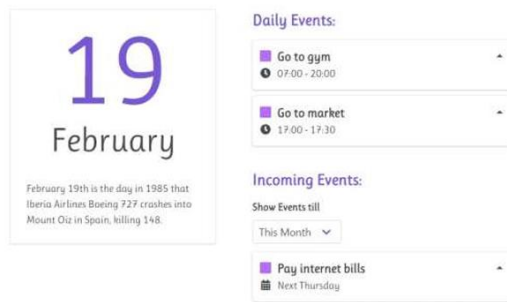


Рисунок 3.12 - Картка події відображається багато разів на інформаційній панелі

Ці картки подій майже ідентичні за структурним стилем та функціональністю, за винятком даних кожної картки. Усі інші функції карток однакові, оскільки вони використовують один компонент для рендерингу. Якщо ці картки подій додаються вручну в нашому застосунку, то щоразу, коли розробники хочуть щось змінити в картках подій, їм доводиться застосовувати це до всіх інших карток подій застосунку. Цей повторюваний процес може бути дуже неприємним і споживає величезну кількість часу. Більше того, у процесі розробки зміни є вирішальною частиною та часто трапляються для досягнення кінцевого продукту або бажаного результату. Це означає, що використання компонентів допомагає розробникам заощаджувати багато часу на виконання тієї ж роботи понаднормово та заохочує їх не виконувати виснажливу та трудомістку роботу [37].

Відображення списку

```

1 <template>
2   <div class="dailyEventsContainer">
3     <div class="label">Daily Events:</div>
4     <div v-if="dailyEvents.length > 0" class="eventsContainer">
5       <DailyEventCard v-for="(event) in dailyEvents" :key="event.id" :event="event"></DailyEventCard>
6     </div>
7     <p v-else class="secondaryTextColor">You have no Daily Event yet</p>
8   </div>
9 </template>

```

Рисунок 3.13 - Компонент списку щоденних подій

Як ми бачимо зі списку щоденних подій на панелі інструментів, відображається більше однієї картки, але з коду ми бачимо лише один елемент

<DailyEventCard> . Причина цього полягає в тому, що елемент

БР.ІП - 94.00.00.000 ПЗ

Змн.	Арк.	№ докум.	Підпис	Дата

р.р.к.

використовує певну директиву з Vue.js, а саме **v-for** . Ця директива дозволяє відображення списку, тобто вона приймає масив даних, зокрема "dailyEvents", і відтворює компонент **<DailyEventCard>** для кожного елемента масиву. Ця директива допомагає створювати динамічне відображення лише за допомогою HTML-коду. З іншого боку, створення динамічної функціональності, як-от ця операція, вимагає втручання JavaScript, якщо не використовуються жодні фреймворки.

```

1  <div id="eventsContainer"></div>
2
3  <script>
4  var events = ['Event 1', 'Event 2']
5
6  document.addEventListener('DOMContentLoaded', () => {
7    eventContainer = document.getElementById('eventsContainer')
8    events.forEach(event => {
9      var eventCard = document.createElement('p')
10     eventCard.innerText = event
11     eventContainer.appendChild(eventCard)
12   })
13 })
14 </script>

```

Рисунок 3.14 - Рендеринг списку за допомогою простого JavaScript

Візьмемо, наприклад, наведену вище реалізацію. Ця реалізація використовує масив подій для створення абзацу для кожного елемента масиву. Тексти абзаців є вмістом елемента в масиві. Цей приклад - це просто реалізація того, як використовувати чистий JavaScript для створення списку. Результат наведено нижче.



Рисунок 3.15 - Результат візуалізації списку

Результат дуже скромний порівняно з картою події, представленою в попередньому розділі. Незважаючи на простоту результату, обсяг необхідного коду досить великий і його важче читати [38]. Крім того, цей приклад містить лише прості HTML-абзаци та не має реалізованих інших стилів чи взаємодій.

Щоб вручну відрендерити повну картку події зі стилем та функціями, потрібно багато роботи та часу. Розгляньте інший приклад реалізації нижче.

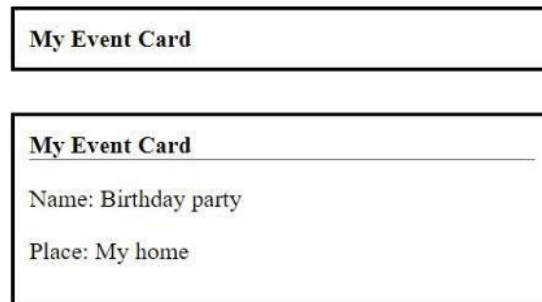


Рисунок 3.16 - Приклад реалізації картки події у закритому та відкритому стані

Існує два способи створити цей ефект. Перший – за допомогою звичайного HTML (можна використовувати як компоненти Vue.js), а інший – за допомогою чистого JavaScript.

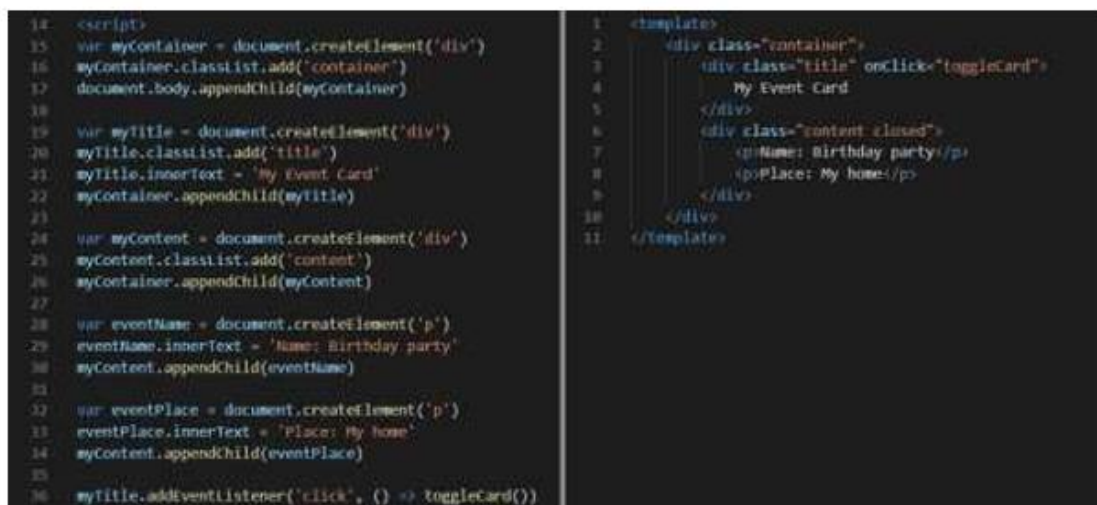


Рисунок 3.17 - Порівняння реалізації на JavaScript (ліворуч) та HTML (праворуч)

Як видно на рисунку 3.17, відмінності досить суттєві, оскільки реалізація HTML займає лише 11 рядків коду, а JavaScript - 22 рядки. Це вдвічі більше роботи, не кажучи вже про те, що рядок коду реалізації JavaScript довший і його складніше візуалізувати, ніж у HTML [39]. Вважається, що використання HTML для виконання динамічних операцій, таких як рендеринг списків, є кращим вибором. Крім того, деякі специфічні процеси (рендеринг карток подій з масивів), подібні до цього, вимагають динамічного рендерингу з JavaScript, але фреймворк Vue.js дозволяє рендеринг з HTML. Ці переваги стають більш помітними, коли додаток масштабується та стає складнішим.

Реактивність

Як видно з наведеного вище прикладу, використання Vanilla JavaScript для рендерингу будь-чого в DOM є дуже складним. Але на цьому справа не закінчується; розглянемо реактивність. Реактивність допомагає оновлювати представлення щоразу, коли дані змінюються. На рисунку 3.18 нижче наведено простий приклад лічильника.



Рисунок 3.18 - Приклад простого механізму лічильника 3

Пропущено CSS-код, оскільки він однаковий для обох реалізацій. На перший погляд, ці дві реалізації не мають суттєвих відмінностей. Обидві реалізації мають однакову кількість рядків коду та виглядають досить схожими одна на одну, особливо оскільки обидва мають однакову структуру HTML та дві функції з однаковим призначенням: метод **add** та **reset**. Але у реалізації Vanilla JavaScript потрібна інша функція, **updateItemNumberView**, яка оновлює представлення. І щоразу, коли програма викликає метод **add** або **reset**, цей метод потрібно викликати знову, щоб переконатися, що представлення завжди відображає найновіші дані [40].



Рисунок 3.19 - Порівняння реалізації на Vanilla JavaScript (ліворуч) та Vue.js (праворуч)

Тим часом, у реалізації Vue.js номер елемента зберігається в об'єкті даних, що робить його реактивним ресурсом компонента. Щоразу, коли номер елемента змінюється (за допомогою інших операцій або методів), представлення автоматично оновлює їх. Щоб відобразити дані в додатку, розробники можуть використовувати прив'язку даних (v-модель) у формах або використовувати синхронізацію вусів для відображення будь-яких даних в об'єкті даних. Ця операція гарантує, що представлення завжди відображає найновіші дані без їх ручного встановлення для кожних даних. Процес оновлення представлення за допомогою чистого JavaScript у наведеній вище реалізації може здаватися простим і зрозумілим. Але уявіть, що ці дані з'являються у представленні багато разів, і є багато фрагментів даних, про які потрібно подбати, а не лише одні дані, як у цьому прикладі. Саме тоді реактивність Vue.js стає переконливою функцією для розробників, щоб заощадити час і не турбуватися про цю неприємну та виснажливу проблему. У статті, написаній під назвою "Найглибша причина існування сучасних JavaScript-фреймворків" згадується безліч вагомих причин. Ці причини включають компонентну архітектуру, активну та велику спільноту, залежності від сторонніх розробників, розширення браузера та придатність для створення SPA (односторінкових додатків). Але також зазначається, що це не найважливіші

					БР.ІП - 94.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		63

причини. Натомість, головне - це проста, але потужна та зручна функціональність, синхронізація інтерфейсу користувача зі станом. Іншими словами, це те, чого розробники можуть досягти за допомогою **Reactivity** фреймворку Vue.js.

3.3 Оцінка ефективності, результати роботи та обмеження системи

У цьому підрозділі робота зосереджена на висновках та відповідях на дослідницькі питання. Після цього в роботі представлено достовірність дослідження, валідність, обмеження та пропозиції для майбутніх досліджень для завершення висновків роботи. Початок цього розділу полягає в перегляді та висновках дослідницьких питань.

Відповіді на дослідницькі питання:

З початку роботи впливають два дослідницькі питання. Ці питання визначають тему та напрямок дослідження. Ці два питання:

- Які основні особливості фреймворку Vue.js та для чого вони використовуються?
- Як ці функції покращують процес розробки?

На основі досліджень, «Компонентно-орієнтована архітектура», «Рендеринг списків» (або динамічний рендеринг з компонентів) та **реактивність** – це три з багатьох функцій, що надаються фреймворком Vue.js. Ці вибрані функції також є відповіддю на перше питання. На основі досліджень та оцінок у попередніх розділах, робота намагається вказати на позитивний вплив, створений цими функціональними можливостями, який представлений у наступному абзаці.

По-перше, приклади показують, що Vue.js зменшує обсяг коду, необхідного для певних завдань або операцій. Крім того, цей фреймворк також запобігає розробникам виконувати повторювані та виснажливі завдання за допомогою компонентів. Ці переваги призводять до скорочення часу розробки, а також підтримують високий моральний дух розробників. По-друге, завдяки цим реалізаціям, Vue.js забезпечує кращу візуалізацію коду, структурування та читабельність завдяки функціям **на основі компонентів та відображення списків**. Ці

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

ефекти можна перетворити на покращення керованості та ефективності проекту, що, якщо все зробити правильно, також може скоротити час розробки. Нарешті, Vue.js дозволяє розробникам легко виконувати певні завдання, що дуже важко зробити без допомоги фреймворку, особливо для нових розробників. Використання Vue.js **Reactivity** є прикладом; це допомагає процесу розробки різними способами. Це включає забезпечення менших витрат часу, кращої продуктивності з мінімальним обсягом коду, вирішення найскладнішої проблеми у веб-розробці (синхронізація даних). Всі ці переваги також стають більш значними, коли додаток починає масштабуватися. Зазвичай, у деяких конкретних ситуаціях ці три функції допомагають процесу розробки, створюючи кращу керованість та масштабованість проектів, а також скорочуючи час розробки та надаючи кращі рішення для конкретних складних проблем у багатьох проектах. Цей висновок є відповіддю на друге питання.

Надійність та валідація досліджень

У роботі не лише демонструються деякі теорії з використанням достовірних джерел, пов'язаних з темою дослідження, але й наводяться приклади реалізацій та статті для підтвердження ідей дослідження. Незважаючи на це, у роботі робиться спроба покращити надійність та валідність деякими іншими способами.

По-перше, під час представлення функції також демонструється та пояснюється багато прикладів реалізації. Усі приклади створюються та тестуються перед тим, як використовувати їх для аналізу. Тестування може бути проведене з використанням точного коду, щоб гарантувати надійність та валідність прикладів.

По-друге, основним предметом цього дослідження є досліджуваний кейс. Проект застосування можна знайти за цим посиланням: <https://github.com/tom2lua/reminder>. Весь застосунок можна знайти в цьому репозиторії. Бекенд застосунку не включено, оскільки він не є предметом роботи. Для тестування розробницьку версію можна запустити з цього репозиторію або за іншим посиланням: <https://amazing-shirley-0738ea.netlify.com>. Це посилання містить робочу версію проекту, яка зібрана на основі проекту. До програми можна отримати доступ

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

безпосередньо за цим другим посиланням без необхідності налаштування проєкту.

Обмеження дослідження

Версія фреймворку Vue.js, яка використовувалася як у досліджуваному випадку, так і в прикладах реалізації, - це 2.0. Це обмеження може призвести до відмінностей результатів у попередніх або пізніших версіях фреймворку Vue.js. Ще один момент, який варто зазначити, - це складність та розмір розроблюваного застосунку. Фреймворк виявляється значущим лише в тому випадку, якщо застосунок має середню або високу складність та розмір. Малі та прості проєкти або застосунки можуть не отримати користі від цих функцій. З іншого боку, їх використання може мати деякі недоліки, такі як дотримання додаткових принципів та правил фреймворку для забезпечення найкращої продуктивності. Ці речі варто враховувати, розпочинаючи новий проєкт, щоб передбачити кращий результат.

Пропозиції для майбутніх досліджень

Насамкінець, наведено деякі пропозиції щодо майбутніх досліджень та розробок, натхненних цією роботою. Досліджень інших функцій Vue.js може бути достатнім, оскільки це дослідження працює лише з трьома основними функціями. Іншими чудовими темами можуть бути зосередження на глибокий аналіз цих функцій для кращого розуміння та оптимізації їх використання. На момент написання роботи скоро вийде Vue.js версії 3. Це нове оновлення також може стати предметом для майбутніх досліджень.

Високий попит на веб-додатки в наш час призводить до зростання потреби в багатьох нових розробниках. У роботі розробникам представлені фундаментальні теорії необхідних компонентів веб-додатку. Інші важливі визначення - це мова програмування JavaScript та її бібліотеки й фреймворки. Оскільки робота зосереджена на вивченні фронтенду додатку, тема дослідження зміщується на Vue.js, один з найвідоміших фреймворків JavaScript для фронтенд-розробки.

Для кращого розуміння фреймворку в роботі наведено приклад дослідження веб-застосунку на базі Vue.js. Загалом, приклад дослідження являє собою веб-застосунок-нагадування, де користувачі можуть створювати плани, які

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

допоможуть їм нагадувати в майбутньому.

Після опису випадку представлено та пояснено три основні особливості фреймворку Vue.js для збору та аналізу даних. Цей процес полягає у порівнянні прикладів реалізації з використанням Vue.js та без нього, щоб знайти відмінності в підходах.

3.4 Висновки по розділу

У третьому розділі було реалізовано веб-додаток планування на базі Vue.js, виконано тестування його функціональних можливостей та проведено оцінювання ефективності роботи системи. Розглянуто процес збору та аналізу даних користувача, а також оцінено продуктивність і зручність використання додатку. Результати дослідження підтвердили доцільність використання сучасних вебтехнологій для створення інтерактивних систем планування. Визначено основні переваги реалізованого рішення, а також окреслено можливі напрями подальшого вдосконалення функціональності та оптимізації системи.

					БР.ІП - 94.00.00.000 ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання бакалаврської роботи було розроблено веб-додаток планування на базі Vue.js, який є комплексним програмним рішенням для організації задач, управління подіями та підвищення ефективності роботи користувача. У роботі проаналізовано сучасні підходи до створення односторінкових веб-додатків, досліджено можливості JavaScript-фреймворків та реалізовано функціональний додаток із використанням компонентного підходу, маршрутизації та інтерактивного користувацького інтерфейсу. Процес розробки охопив аналіз вимог, проектування архітектури, реалізацію функціональних модулів, тестування та оцінювання ефективності програмного забезпечення, що дозволило створити сучасний і зручний веб-застосунок.

На початковому етапі було проведено аналіз предметної області та визначено основні функціональні й нефункціональні вимоги до системи. До функціональних вимог було віднесено створення, редагування, перегляд і видалення задач, підтримку навігації між сторінками та забезпечення інтерактивної взаємодії користувача з інтерфейсом. Нефункціональні вимоги охоплювали продуктивність, масштабованість, адаптивність інтерфейсу та зручність використання. Аналіз існуючих рішень дозволив визначити переваги сучасних SPA-додатків і обґрунтувати вибір Vue.js як основної технології реалізації системи.

У процесі проектування було розроблено архітектуру веб-додатку з використанням компонентного підходу, що забезпечило модульність та гнучкість структури системи. Особливу увагу приділено реалізації механізмів двостороннього зв'язування даних, які дозволяють забезпечити автоматичне оновлення інтерфейсу при зміні даних користувача. Для організації навігації між сторінками було використано Vue Router, що дало можливість реалізувати швидкий перехід між функціональними модулями без перезавантаження сторінки. У роботі також досліджено переваги використання сучасних frontend-технологій для створення інтерактивних веб-застосунків.

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

Реалізація веб-додатку включала створення користувацького інтерфейсу, налаштування компонентів Vue.js, обробку подій користувача та організацію збереження й обробки даних. Для розробки використовувалися Visual Studio Code, JavaScript, HTML та CSS, що забезпечило ефективну взаємодію між компонентами системи та зручність підтримки програмного коду. Значну увагу було приділено адаптивності інтерфейсу та забезпеченню комфортної роботи користувача на різних типах пристроїв.

У межах роботи було проведено тестування програмного забезпечення, включаючи перевірку функціональності, стабільності роботи інтерфейсу та коректності маршрутизації. Під час тестування було виявлено та усунуто потенційні недоліки, пов'язані з оновленням даних і взаємодією між компонентами. Оцінювання ефективності показало, що використання Vue.js забезпечує високу швидкодію додатку, спрощує підтримку програмного коду та дозволяє масштабувати систему відповідно до потреб користувачів.

Загалом, розроблений додаток відповідає поставленим цілям та забезпечує ефективне управління задачами й організацію роботи користувача. Перевагами обраного підходу є простота використання Vue.js, компонентна архітектура, підтримка двостороннього зв'язування даних та можливість створення інтерактивного користувацького інтерфейсу. Водночас існують певні обмеження, пов'язані з необхідністю подальшого вдосконалення системи безпеки, оптимізації продуктивності при роботі з великими обсягами даних та інтеграції із зовнішніми сервісами.

У майбутньому веб-додаток може бути розширений функціями синхронізації даних із хмарними сервісами, підтримкою авторизації користувачів, інтеграцією календарів, систем сповіщень та мобільної версії застосунку. Це дозволить зробити систему більш функціональною, гнучкою та зручною для використання в різних сферах діяльності.

					БР.ІІІ - 94.00.00.000 ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Vue.js Documentation. (2025). vuejs.org.
<https://vuejs.org/guide/introduction.html>

2. Vue Router Documentation. (2025). router.vuejs.org.
<https://router.vuejs.org/>

3. Vuex Documentation. (2025). vuex.vuejs.org. <https://vuex.vuejs.org/>

4. Vite Documentation. (2025). vitejs.dev. <https://vitejs.dev/guide/>

5. JavaScript Guide. (2025). developer.mozilla.org.
<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>

6. HTML Living Standard. (2025). html.spec.whatwg.org.
<https://html.spec.whatwg.org/>

7. CSS Documentation. (2025). developer.mozilla.org.
<https://developer.mozilla.org/en-US/docs/Web/CSS>

8. Bootstrap Documentation. (2025). getbootstrap.com.
<https://getbootstrap.com/docs/>

9. Sass Basics. (2024). sass-lang.com. <https://sass-lang.com/guide>

10. Node.js Documentation. (2025). nodejs.org. <https://nodejs.org/en/docs>

11. npm Documentation. (2025). docs.npmjs.com. <https://docs.npmjs.com/>

12. Visual Studio Code Documentation. (2025). code.visualstudio.com.
<https://code.visualstudio.com/docs>

13. Cypress End-to-End Testing. (2025). docs.cypress.io.
<https://docs.cypress.io/guides/overview/why-cypress>

14. Selenium Documentation. (2025). selenium.dev.
<https://www.selenium.dev/documentation/>

15. Postman API Platform Documentation. (2025). learning.postman.com.
<https://learning.postman.com/docs/>

16. OWASP Top Ten Security Risks. (2024). owasp.org.
<https://owasp.org/www-project-top-ten/>

17. Martin, R. C. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design. pearson.com. <https://www.pearson.com/store/p/clean-architecture-a-craftmans-guide-to-software-structure-and-design/P100001465201>
18. Fowler, M. (2002). Patterns of Enterprise Application Architecture. martinowler.com. <https://martinfowler.com/eaCatalog/>
19. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. pearson.com. <https://www.pearson.com/store/p/design-patterns-elements-of-reusable-object-oriented/P100000252058>
20. Freeman, E., & Robson, E. (2021). Head First Design Patterns. oreilly.com. <https://www.oreilly.com/library/view/head-first-design/9781492077992/>
21. Flanagan, D. (2020). JavaScript: The Definitive Guide. oreilly.com. <https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/>
22. Duckett, J. (2014). JavaScript and JQuery: Interactive Front-End Web Development. wiley.com. <https://www.wiley.com/en-us/JavaScript+and+jQuery%3A+Interactive+Front+End+Web+Development-p-9781118531648>
23. ECMAScript Language Specification. (2025). ecma-international.org. <https://tc39.es/ecma262/>
24. Google Developers. (2025). Web Performance Optimization. developers.google.com. <https://developers.google.com/web/fundamentals/performance>
25. Mozilla Developer Network. (2025). Single-page applications. developer.mozilla.org. <https://developer.mozilla.org/en-US/docs/Glossary/SPA>
26. Vue Mastery. (2025). Vue.js Tutorials and Courses. vuemastery.com. <https://www.vuemastery.com/>
27. Vue School. (2025). Vue.js Learning Platform. vueschool.io. <https://vueschool.io/>
28. Netlify Documentation. (2025). docs.netlify.com. <https://docs.netlify.com/>

					БР.ІІІ - 94.00.00.000 ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		71

29. Vercel Documentation. (2025). vercel.com. <https://vercel.com/docs>
30. Firebase Documentation. (2025). firebase.google.com. <https://firebase.google.com/docs>
31. MongoDB Documentation. (2025). mongodb.com. <https://www.mongodb.com/docs/>
32. Axios HTTP Client Documentation. (2025). axios-http.com. <https://axios-http.com/docs/intro>
33. Lodash Documentation. (2025). lodash.com. <https://lodash.com/docs/>
34. Nielsen, J. (2020). Usability Engineering Principles for User Interfaces. nngroup.com. <https://www.nngroup.com/articles/usability-101-introduction-to-usability/>
35. Google Material Design Guidelines. (2025). material.io. <https://m3.material.io/>
36. Kumar, S., & Sharma, R. (2023). Modern frontend architectures for scalable web applications. Journal of Software Engineering Research and Development, 11(2), 67–82. <https://doi.org/10.1186/s40411-023-00189-9>
37. Wang, L., & Zhang, Y. (2024). Performance optimization techniques for single-page applications. ACM Transactions on Web, 18(4), 1–19. <https://doi.org/10.1145/3621458>
38. Chen, J., & Li, X. (2024). User interface optimization in modern JavaScript applications. Journal of Web Engineering, 23(3), 201–220. <https://doi.org/10.13052/jwe1540-9589.2334>
39. Hassan, M., & Khan, A. (2023). Scalable frontend architectures using component-based frameworks. IEEE Software, 40(5), 45–53. <https://doi.org/10.1109/MS.2023.3267812>
40. Taylor, M., & Brown, A. (2025). Security and performance practices for modern web applications. IEEE Security & Privacy, 23(1), 45–53. <https://doi.org/10.1109/MSEC.2024.347890>

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: " Розробка власного додатку планування на базі Vue.js "

Обсяг пояснювальної записки: 70 аркушів

Дата закінчення бакалаврської роботи 10 червня 2026р.

Підпис студента _____