

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Платформа з розміщення подій та заходів (розробка
фронтенду)»**

Виконав студент групи ПЗ-21-5
ПАРХОМЧУК Руслан Вікторович

Керівник роботи:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
ВЛАСЕНКО Олег Васильович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк

«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПАРХОМЧУК Руслан Вікторович**

Керівник роботи: **ЄФРЕМОВ Юрій Миколайович**

Тема роботи: **«Платформа з розміщення подій та заходів (розробка фронтенду)»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №74/с**

Вихідні дані для роботи:

- об'єкт дослідження: **«процеси створення, розміщення та управління подіями та заходами у веб-середовищі»**.
- предмет дослідження: **«методи та технології розробки фронтенд-частини веб-платформи для розміщення подій та заходів із використанням сучасних JavaScript-фреймворків та бібліотек»**.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю.М.	27.02.2025р.	27.02.2025р.
2	Єфремов Ю.М.	27.02.2025р.	27.02.2025р.
3	Єфремов Ю.М.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-додатку платформи з розміщення подій та заходів та пояснювальної записки. Пояснювальна записка до випускної роботи містить 77 сторінок, 29 ілюстрацій, 16 таблиць, 20 джерел літератури та 8 сторінок додатків.

Метою роботи є розробка веб-орієнтованого фронтенду для платформи розміщення подій та заходів з використанням технологій Next.js, React 19, TypeScript та TailwindCSS.

Архітектура платформи реалізована на основі фреймворку Next.js, що забезпечує серверний рендеринг (SSR) та клієнтський рендеринг (CSR), поєднуючи переваги традиційних веб-додатків та сучасних клієнтських технологій. Клієнтська частина використовує React Query для ефективної роботи з даними та React Hook Form для обробки форм.

Додаток має сучасний, адаптивний інтерфейс, побудований на компонентній архітектурі з використанням ShadCN UI та інтеграцією REST API для забезпечення повнофункціонального керування подіями, розкладами та локаціями.

КЛЮЧОВІ СЛОВА: NEXT.JS, REACT, TYPESCRIPT, TAILWINDCSS, ПЛАТФОРМА ПОДІЙ, ФРОНТЕНД, SSR, CSR, REST API, REACT QUERY.

					ІПЗ.КР.Б – 121 – 25 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Пархомчук Р.В.			Платформи з розміщення подій та заходів (розробка фронтенду)	Літ.	Арк.	Аркушів
Перевір.		Єфремов Ю.М.					3	87
Керівник						ФІКТ Гр. ІПЗ-21-5		
Н. контр.								
Зав. каф.								

ABSTRACT

The bachelor's qualification thesis consists of a web application for an event posting platform and an explanatory note. The explanatory note to the thesis contains 77 pages, 29 illustrations, 16 tables, 20 literature sources and 8 pages of additions.

The aim of the work is to develop a web-oriented frontend for an event posting platform using Next.js, React 19, TypeScript, and TailwindCSS technologies.

The platform architecture is implemented based on the Next.js framework, providing server-side rendering (SSR) and client-side rendering (CSR), combining the advantages of traditional web applications and modern client-side technologies. The client side uses React Query for efficient data handling and React Hook Form for form processing.

The application features a modern, adaptive interface built on a component architecture using ShadCN UI and integrates with REST API to provide full-featured management of events, schedules, and locations.

KEYWORDS: NEXT.JS, REACT, TYPESCRIPT, TAILWINDCSS, EVENT PLATFORM, FRONTEND, SSR, CSR, REST API, REACT QUERY.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП.....	7
1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗМІЩЕННЯ ПОДІЙ ТА ЗАХОДІВ.....	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту.....	11
1.3 Вибір архітектури веб-орієнтованої системи розміщення подій та заходів	15
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	18
1.5 Розподілення обов'язків у команді	21
Висновки до розділу 1	23
2. ПРОЕКТУВАННЯ ФРОНТЕНД-ЧАСТИНИ ПЛАТФОРМИ РОЗМІЩЕННЯ ПОДІЙ ТА ЗАХОДІВ.....	26
2.1 Визначення варіантів використання та компонентної структури системи.....	26
2.2 Розробка моделей даних та взаємодії з API	34
2.3 Проектування та реалізація алгоритмів роботи системи	41
2.4 Реалізація функціоналу системи платформи з розміщення подій та заходів	48
Висновки до розділу 2.....	55
3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ПОШУКУ ТА УПРАВЛІННЯ АКТИВНОСТЯМИ.....	58
3.1 Порядок встановлення та налаштування фронтенд-частини системи	58
3.2 Структура інтерфейсу платформи пошуку та управління активностями	61
3.3 Тестування роботи програмного комплексу	68
Висновки до розділу 3.....	72
ВИСНОВКИ.....	75
ЛІТЕРАТУРА	77
ДОДАТКИ.....	79

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API – прикладний програмний інтерфейс

БД – база даних

ПЗ – програмне забезпечення

CSR – клієнтський рендеринг (Client-Side Rendering)

SSR – серверний рендеринг (Server-Side Rendering)

REST – передача репрезентативного стану (Representational State Transfer)

JWT – JSON веб-токен

UI – користувацький інтерфейс (User Interface)

UX – досвід користувача (User Experience)

HTML – мова розмітки гіпертексту (HyperText Markup Language)

CSS – каскадні таблиці стилів (Cascading Style Sheets)

JSON – нотація об'єктів JavaScript (JavaScript Object Notation)

DOM – об'єктна модель документа (Document Object Model)

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми дослідження обумовлена стрімким розвитком індустрії організації подій та заходів, що вимагає сучасних і зручних інструментів для їх ефективного планування та управління. В умовах цифрової трансформації суспільства та бізнесу, веб-платформи для розміщення та організації подій стають невід'ємною частиною комунікаційної та маркетингової стратегії організаторів заходів різного масштабу. Використання таких платформ дозволяє значно спростити процеси планування, організації, просування подій та управління реєстрацією учасників, що підвищує ефективність роботи організаторів та покращує досвід відвідувачів.

Розробка фронтенду для платформи розміщення подій та заходів з використанням сучасних технологій, таких як Next.js, React та TypeScript, забезпечує створення високопродуктивних, адаптивних та інтуїтивно зрозумілих інтерфейсів, які відповідають сучасним вимогам користувачів щодо швидкодії, зручності використання та функціональності. Особливої актуальності набуває інтеграція платформи з різними сервісами та API, що розширює можливості системи та забезпечує гнучкість у використанні.

Метою випускної кваліфікаційної роботи бакалавра є розробка веб-орієнтованого фронтенду для платформи розміщення подій та заходів із використанням сучасних JavaScript-технологій, що забезпечить зручні та ефективні інструменти для створення, управління та пошуку подій.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Провести аналіз предметної області та існуючих аналогів платформ розміщення подій, визначити їх основні функціональні можливості, переваги та недоліки.
2. Визначити архітектуру фронтенд-частини системи та обґрунтувати вибір інструментальних засобів розробки.
3. Розробити структуру взаємодії фронтенду з API та проектні рішення для реалізації основних функціональних модулів.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

4. Реалізувати клієнтську частину платформи з використанням обраних технологій та забезпеченням адаптивності інтерфейсу.
5. Провести тестування розробленої системи та проаналізувати результати її роботи.

Об'єктом дослідження є процеси створення, розміщення та управління подіями та заходами у веб-середовищі.

Предметом дослідження є методи та технології розробки фронтенд-частини веб-платформи для розміщення подій та заходів із використанням сучасних JavaScript-фреймворків та бібліотек.

Методи дослідження включають системний аналіз предметної області, компонентний підхід до розробки користувацького інтерфейсу та клієнт-серверну модель взаємодії.

Практичне значення отриманих результатів полягає у створенні готового до використання фронтенду платформи розміщення подій та заходів, яка може бути впроваджена для організації заходів різного масштабу та тематики, від корпоративних зустрічей до масових фестивалів. Розроблений продукт має потенціал для використання як комерційними структурами, так і некомерційними організаціями, навчальними закладами та органами місцевого самоврядування.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗМІЩЕННЯ ПОДІЙ ТА ЗАХОДІВ

1.1 Постановка задачі

Метою кваліфікаційної роботи бакалавра є розробка веб-орієнтованого фронтенду для платформи пошуку та управління активностями з використанням сучасних JavaScript-технологій, таких як Next.js 15, React 19, TypeScript та TailwindCSS.

Платформа пошуку та управління активностями – це веб-сервіс, який дозволяє користувачам створювати, публікувати та знаходити різноманітні заходи та активності. Такі платформи відіграють важливу роль в організації соціальних, освітніх, культурних та бізнес-подій, спрощуючи процес донесення інформації до потенційних відвідувачів та учасників.

За останні роки індустрія організації заходів пройшла значну цифрову трансформацію. Згідно з дослідженнями Event MB, понад 90% організаторів подій використовують спеціалізовані онлайн-платформи для просування своїх заходів. Крім того, більше 60% учасників знаходять інформацію про події саме через спеціалізовані онлайн-сервіси та соціальні мережі.

Платформа управління активностями має представляти собою декілька програмних модулів, що забезпечують різноманітні функції та можливості. Модуль управління активностями дозволяє організаторам створювати одноразові та регулярні активності з детальним описом, встановлювати дати і час проведення, визначати локації та керувати варіантами участі, підтримуючи систему бронювання місць та варіантів активностей. Модуль управління розкладами надає інструменти для створення та редагування розкладів регулярних активностей, визначення робочих днів, годин роботи та перерв, управління тривалістю бронювань. Модуль управління місцями проведення забезпечує можливість створення та управління локаціями з додаванням детальної інформації, медіаматеріалів та інтерактивних планів розташування

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

місць. Модуль користувацьких профілів відповідає за управління профілями користувачів, їхніми правами доступу, персональною інформацією та історією активності. Модуль пошуку та фільтрації забезпечує зручні інструменти для пошуку активностей за різними критеріями, включаючи територіальну фільтрацію, тип активності, дату проведення тощо. Модуль взаємодії користувачів включає систему реакцій на активності, управління бронюваннями та статистичну аналітику для організаторів.

В результаті повинно бути створено веб-додаток з адаптивним інтерфейсом, за допомогою якого користувачі зможуть ефективно взаємодіяти з платформою на різних пристроях.

Основними етапами розробки фронтенду платформи є розробка дизайну веб-інтерфейсу з використанням компонентної бібліотеки ShadCN UI на базі Radix UI, створення адаптивного дизайну з підтримкою світлої та темної теми. Наступним етапом є побудова архітектури клієнтської частини додатку з використанням Next.js App Router, організація типобезпечної взаємодії з API через TypeScript інтерфейси та Zod схеми валідації. Також передбачається реалізація системи авторизації та аутентифікації користувачів з JWT-токенами, автоматичним оновленням токенів та управлінням сесіями через HTTP-cookies. Важливим етапом є розробка основного функціоналу з використанням React Hook Form для управління формами, TanStack Query для ефективної роботи з серверним станом, Zustand для локального стану компонентів. Завершальним етапом є тестування інтерфейсу користувача, функціональне тестування додатку, перевірка адаптивності дизайну та оптимізація швидкодії.

Результатом виконання плану дій є створення клієнтської частини платформи пошуку активностей, яка забезпечує зручний доступ до функціоналу системи для різних типів користувачів – від організаторів, які створюють активності, до відвідувачів, які шукають цікаві заходи та здійснюють бронювання.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

1.2 Аналіз аналогів програмного продукту

Розробка платформи для пошуку та управління активностями потребує ретельного аналізу існуючих рішень на ринку. Розглянемо кілька популярних платформ, які надають подібний функціонал.

Однією з найбільш відомих подібних платформ є Eventbrite (рис. 1.1).

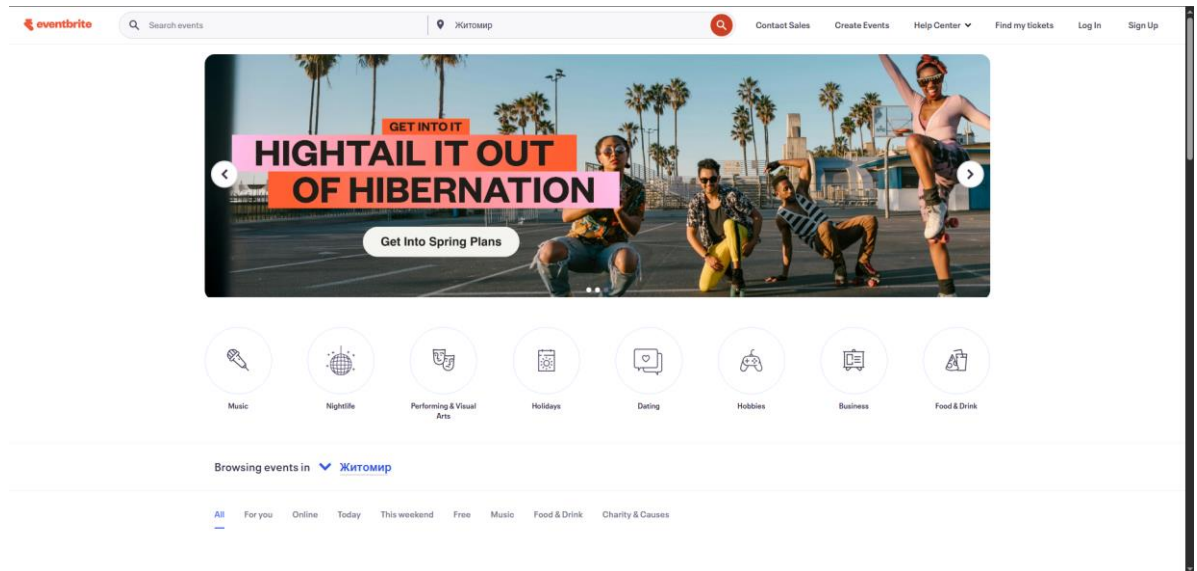


Рис. 1.1 – Платформа розміщення подій Eventbrite

Eventbrite – це глобальний сервіс для організації подій, який дозволяє користувачам створювати, просувати та продавати квитки на різноманітні заходи. Платформа орієнтована як на індивідуальних організаторів, так і на великі компанії.

Основними перевагами Eventbrite є інтуїтивно зрозумілий інтерфейс для створення подій, потужна система продажу квитків з різними тарифними планами, інтеграція з соціальними мережами та інструментами маркетингу, підтримка різних мов та валют, гнучкі варіанти оплати та можливість встановлення промокодів. Недоліками є висока комісія за продаж квитків через платформу, обмежені можливості для кастомізації сторінки події у безкоштовній версії та складна система налаштування для початківців.

Наступним аналогом є платформа Meetup (рис. 1.2) – сервіс, який орієнтований на створення та знаходження локальних спільнот та подій за інтересами.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

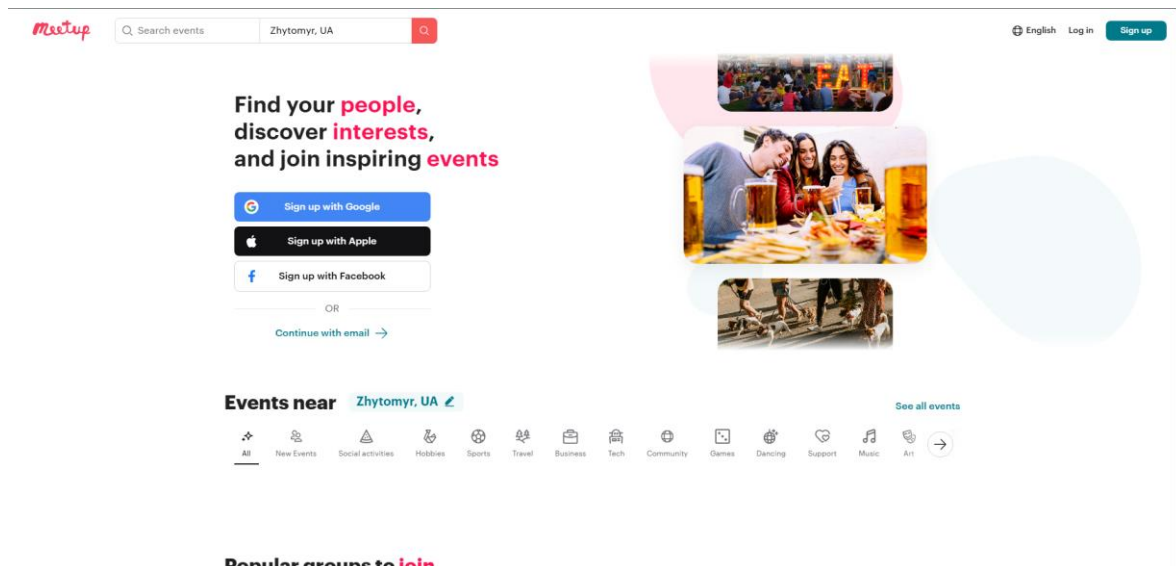


Рис. 1.2 – Платформа розміщення подій Meetup

Основними перевагами Meetup є орієнтація на формування локальних спільнот за інтересами, простий процес створення події, вбудована система комунікації між учасниками, зручний пошук подій за категоріями та локацією, розвинена мобільна версія. Недоліками є платна підписка для організаторів груп, обмежені маркетингові інструменти, менш розвинені функції для платних подій та недостатньо гнучка система налаштування шаблонів.

Ticketmaster (рис. 1.3) – одна з найбільших світових платформ для продажу квитків на різноманітні заходи, концерти, спортивні події тощо.

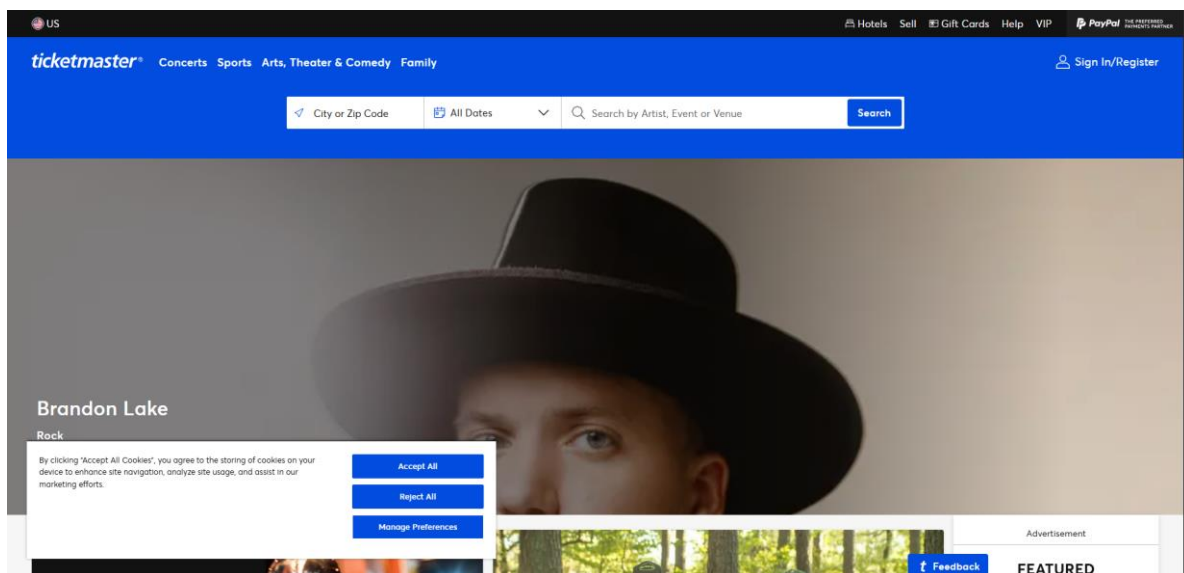


Рис. 1.3 – Платформа розміщення подій Ticketmaster

Основні переваги Ticketmaster включають величезну базу користувачів, надійну систему продажу та перевірки квитків, інтеграцію з різними сервісами

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

та майданчиками, потужну систему пошуку подій, розвинену програму лояльності. Основні недоліки Ticketmaster – це високі сервісні збори, складність використання для невеликих організаторів, обмежена гнучкість у налаштуванні сторінки події, необхідність відповідати суворим вимогам для розміщення події.

Також варто розглянути Eventscase (рис. 1.4) – платформу, яка фокусується на наданні комплексних рішень для організації професійних конференцій та корпоративних заходів.

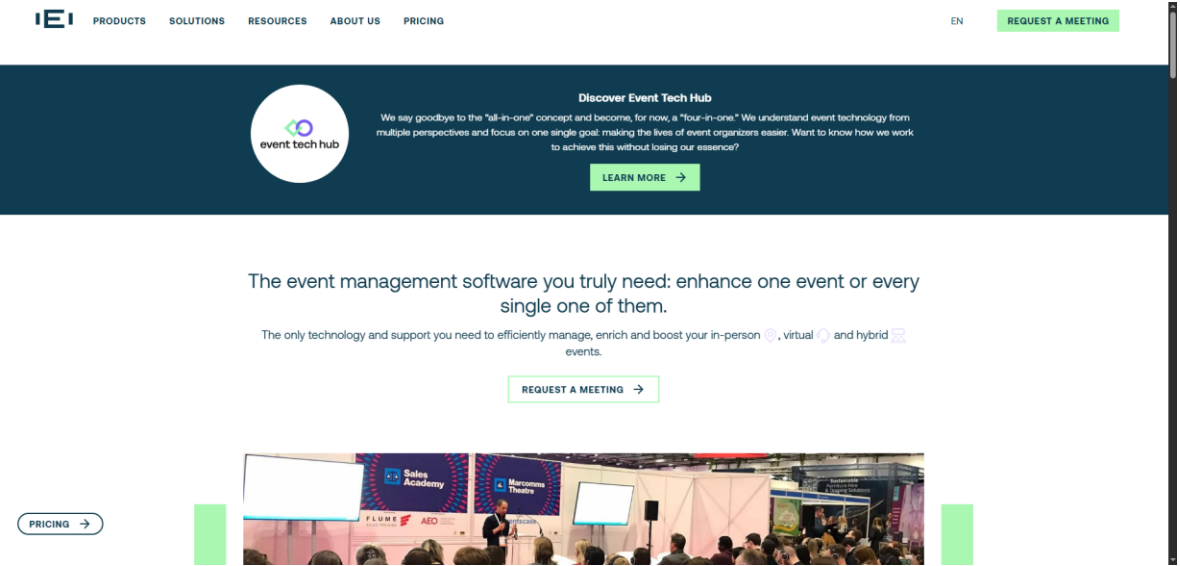


Рис. 1.4 – Платформа розміщення подій Eventscase

Основні переваги Eventscase включають повний набір інструментів для організації професійних заходів, потужну CRM-систему, розширені функції для нетворкінгу учасників, налаштування брендингу та дизайну, аналітичні інструменти для організаторів. Основні недоліки Eventscase – це висока вартість для малих організаторів, складний інтерфейс, що потребує навчання, менша популярність порівняно з іншими платформами, обмежена підтримка деяких регіонів та мов.

Узагальнюючи аналіз існуючих платформ, складемо порівняльну таблицю їх функціональних можливостей та нашого проекту (таблиця 1.1).

Таблиця 1.1

Порівняння платформ розміщення подій та заходів

Характеристика	Eventbrite	Meetup	Ticketmaster	Eventscase	Розроблювана платформа
----------------	------------	--------	--------------	------------	------------------------

Продовження таблиці 1.1

Інтуїтивний інтерфейс	+	+	+ -	-	+
Система бронювання	+	+ -	+	+	+
Управління розкладами	+	+	-	+	+
Адаптивний дизайн	+	+ -	+	+ -	+
Кастомізація	+ -	-	-	+	+ -
Доступність для невеликих організаторів	+ -	+	-	-	+
Аналітичні інструменти	+	+ -	+	+	+
Інтеграція з соцмережами	+	+	+	+	+ -
Підтримка регулярних активностей	+	+	-	+ -	+
Управління локаціями з планами	+	+ -	-	+	+
Система реакцій користувачів	-	+	-	-	+ -
Підтримка одноразових та регулярних активностей	+ -	+ -	-	+	+
Територіальна фільтрація	+ -	+	+ -	-	+ -
Інтерактивні плани місць	-	-	+	+ -	+
Сучасний технологічний стек	-	-	-	+ -	+

Проаналізувавши існуючі рішення на ринку, можна зробити висновок, що більшість платформ мають свою специфіку та орієнтацію на певний сегмент користувачів. При цьому кожна з них має певні обмеження: Eventbrite більше фокусується на продажу квитків, Meetup – на створенні спільнот, Ticketmaster – на великомасштабних подіях, а Eventscase – на професійних конференціях.

Розроблювана платформа має поєднати найкращі практики цих сервісів, зокрема інтуїтивно зрозумілий інтерфейс, гнучке управління розкладами та

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

локаціями, повну адаптивність та доступність на різних пристроях, можливість налаштування для різних типів активностей – від невеликих зустрічей до масштабних конференцій, зручну взаємодію з API.

При цьому, на відміну від існуючих рішень, розроблювана платформа матиме наступні ключові відмінності: сучасний стек технологій Next.js 15, React 19, TypeScript, що забезпечить високу продуктивність; унікальну систему управління як одноразовими, так і регулярними активностями з детальними варіантами участі; інтерактивні плани місць з візуальним редактором; систему реакцій та взаємодії користувачів; територіальну фільтрацію з підтримкою ієрархічного пошуку; статистичну аналітику для організаторів; intercepting routes для плавної навігації через модальні вікна; повну типобезпеку через TypeScript та Zod валідацію.

1.3 Вибір архітектури веб-орієнтованої системи розміщення подій та заходів

При розробці архітектури фронтенд-частини платформи для пошуку та управління активностями необхідно враховувати сучасні підходи та патерни проектування веб-додатків. Основною вимогою до архітектури є її масштабованість, гнучкість та можливість ефективної взаємодії з API.

Існує кілька основних типів та підходів до побудови архітектури веб-додатків. Single Page Application (SPA) – тип веб-застосунків, що працюють в рамках одного веб-документа і динамічно оновлюють контент без перезавантаження сторінки. SPA забезпечує високу швидкість взаємодії з користувачем, однак має обмеження щодо SEO-оптимізації та початкового завантаження. Multi-Page Application (MPA) – традиційна архітектура, де кожна сторінка завантажується з сервера окремо. MPA добре підходить для сайтів з великою кількістю контенту, але забезпечує менш плавний користувацький досвід порівняно з SPA. Server-Side Rendering (SSR) – підхід, при якому HTML генерується на сервері, що дозволяє забезпечити хороше SEO та швидке початкове завантаження сторінки. React-додатки з SSR працюють і на сервері, і

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

на клієнті, що дозволяє поєднати переваги обох підходів. Static Site Generation (SSG) – генерація HTML під час збірки, що забезпечує максимальну швидкість завантаження та хороше SEO, але обмежує динамічність контенту. Клієнтський рендеринг (CSR) – підхід, при якому HTML генерується на стороні клієнта за допомогою JavaScript. Забезпечує високу інтерактивність, але може негативно вплинути на SEO та початкове завантаження.

Для розроблюваної платформи пошуку та управління активностями оптимальним рішенням є використання гібридного підходу, що поєднує SSR та CSR, реалізований через фреймворк Next.js 15 з App Router. Next.js дозволяє використовувати серверний рендеринг для початкового завантаження сторінки, що забезпечує швидку візуалізацію контенту та хорошу індексацію пошуковими системами. Після завантаження сторінки додаток працює як SPA, забезпечуючи плавний користувацький досвід при навігації та взаємодії з інтерфейсом.

Next.js дозволяє використовувати серверний рендеринг для початкового завантаження сторінки, що забезпечує швидку візуалізацію контенту та хорошу індексацію пошуковими системами. Після завантаження сторінки додаток працює як SPA, забезпечуючи плавний користувацький досвід при навігації та взаємодії з інтерфейсом.

Архітектура клієнтської частини платформи базуватиметься на компонентному підході, характерному для React-додатків, з чітким розділенням відповідальності між компонентами. Структурно архітектуру фронтенд-частини можна представити такими основними рівнями. Рівень представлення включає UI компоненти інтерфейсу користувача на основі ShadCN UI, що відповідають за відображення даних та взаємодію з користувачем, з підтримкою світлої та темної теми. Рівень управління станом забезпечує керування станом додатку, включаючи кешування та синхронізацію даних з сервером за допомогою TanStack Query, а також локальний стан через Zustand для фільтрів та специфічних компонентів. Сервісний шар містить модулі для взаємодії з API, обробки та трансформації даних з автоматичним оновленням JWT токенів та типобезпечною взаємодією. Шар валідації включає Zod схеми для перевірки

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

даних та React Hook Form для ефективного управління формами. Утиліти представляють допоміжні функції та інструменти для роботи з датами, форматування даних, валідації файлів та інших операцій.

Така архітектура забезпечує чітке розділення відповідальності та дозволяє ефективно масштабувати додаток при зростанні функціональності. Для забезпечення ефективної взаємодії з API буде використано підхід з розділенням API-клієнтів на окремі модулі за функціональними сферами, включаючи активності, розклади, локації, користувачі, бронювання, реакції та статистика. Це дозволить організувати код у більш структурований та підтримуваний спосіб.

Архітектура додатку також передбачатиме використання сучасних можливостей Next.js App Router, включаючи intercepting routes для плавної навігації через модальні вікна без повного перезавантаження сторінок, parallel routes для одночасного рендерингу різних частин інтерфейсу та loading UI для покращення користувацького досвіду. Middleware забезпечуватиме автентифікацію користувачів та перенаправлення на відповідні сторінки залежно від статусу авторизації.

Система типізації через TypeScript та Zod забезпечуватиме повну типобезпеку від клієнтських форм до серверних відповідей, включаючи discriminated unions для різних типів активностей та бронювань. Це дозволяє виявляти помилки на етапі компіляції та забезпечує надійну взаємодію з API.

Обрана архітектура забезпечує швидке початкове завантаження через SSR, плавну навігацію та взаємодію через CSR з intercepting routes, ефективну роботу з даними через TanStack Query з автоматичною інвалідацією кешу, чітке розділення відповідальності між компонентами з повною типізацією, масштабованість та підтримуваність коду через модульну структуру, оптимізацію для пошукових систем через серверний рендеринг, адаптивність до різних пристроїв та розмірів екрану через TailwindCSS, безпечну взаємодію з API через JWT автентифікацію та автоматичне оновлення токенів.

Така архітектура дозволить створити сучасний, швидкий та зручний веб-додаток для пошуку та управління активностями, який буде відповідати всім

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

поставленим вимогам та забезпечуватиме високий рівень користувацького досвіду з підтримкою складних сценаріїв взаємодії, включаючи управління одноразовими та регулярними активностями, інтерактивні плани місць, територіальну фільтрацію та статистичну аналітику.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Сайт буде побудований на клієнт-серверній архітектурі. Серверна частина представлена у вигляді RESTful API, з яким взаємодіятиме фронтенд-частина, розроблена з використанням сучасних JavaScript-технологій.

Для розробки фронтенд-частини платформи було обрано наступні інструментальні засоби. Next.js 15 – це React-фреймворк, що забезпечує функціональність серверного рендерингу (SSR), статичної генерації (SSG) та клієнтського рендерингу (CSR). Він надає розробникам потужні інструменти для створення високопродуктивних веб-додатків. Переваги Next.js включають гібридний рендеринг з поєднанням SSR та CSR для оптимального балансу між SEO та інтерактивністю, оптимізовані збірки з автоматичним розділенням коду та оптимізацією JS-бандлів, покращений досвід розробника з вбудованим App Router, гарячим перезапуском та підтримкою TypeScript, intercepting routes для плавної навігації через модальні вікна, оптимізацію зображень через вбудовані компоненти та middleware для автентифікації користувачів.

React 19 – бібліотека для створення користувацьких інтерфейсів на основі компонентного підходу. React 19 є найновішою версією з покращеною продуктивністю та новими функціями. Переваги React включають декларативний підхід до створення інтерфейсів, компонентну архітектуру для повторного використання коду, віртуальний DOM для оптимізації продуктивності, потужну екосистему бібліотек та інструментів, хуки для управління станом та життєвим циклом компонентів, підтримку різних типів рендерингу та покращену продуктивність рендерингу порівняно з попередніми версіями.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

TypeScript – типізований надсет JavaScript, який додає статичну типізацію та інші функції для покращення розробки великих додатків. Переваги TypeScript включають статичну типізацію для виявлення помилок на етапі компіляції, покращений інтелектуальний аналіз коду в IDE, кращу документацію коду через типи та інтерфейси, підтримку сучасних функцій JavaScript та ECMAScript, поліпшену рефакторизацію та структурування коду з discriminated unions для різних типів активностей та бронювань.

TailwindCSS 4 – утилітарний CSS-фреймворк з низьким рівнем абстракції, що дозволяє швидко створювати кастомізовані інтерфейси. Переваги TailwindCSS включають швидкий процес розробки інтерфейсів за допомогою готових утилітарних класів, стандартизований підхід до стилізації компонентів, менший розмір CSS у продакшн-версії, легку кастомізацію та розширюваність, адаптивний дизайн із вбудованими медіа-запитами та підтримку світлої і темної теми.

TanStack Query – бібліотека для керування, кешування та синхронізації асинхронних даних у React-додатках. Переваги TanStack Query включають оптимізоване кешування та інвалідація даних, управління станом завантаження та помилок, автоматичне оновлення даних у фоновому режимі, оптимістичні оновлення інтерфейсу, повторні спроби при помилках, дедуплікацію запитів для оптимізації продуктивності та персистентність кешу через localStorage.

React Hook Form – легковагова бібліотека для керування формами в React-додатках з фокусом на продуктивність та простоту використання. Переваги React Hook Form включають мінімальні перерендеринги компонентів при роботі з формами, інтеграцію з системами валідації, особливо Zod, зручний API на основі хуків, відсутність зовнішніх залежностей та ефективну роботу з великими та складними формами.

ShadCN UI на базі Radix UI – бібліотека доступних і ненав'язливих UI-примітивів для побудови високоякісних компонентів React. Переваги включають фокус на доступність, ненав'язливий стиль, що легко кастомізується, підтримку

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

різних станів компонентів, композитний підхід до створення складних UI-компонентів та готові компоненти для швидкої розробки інтерфейсу.

Zod – бібліотека для валідації схем та TypeScript-першого підходу до перевірки даних. Переваги Zod включають повну інтеграцію з TypeScript, автоматичне виведення типів, детальні повідомлення про помилки, підтримку складних схем валідації та seamless інтеграцію з React Hook Form.

Zustand – легковагова бібліотека для управління станом React додатків. Переваги включають простий API без boilerplate коду, TypeScript підтримку з автоматичним виведенням типів та можливість персистентності стану.

Lucide React – бібліотека іконок з сучасним дизайном та повною підтримкою TypeScript. Date-fns – бібліотека для роботи з датами з функціональним підходом та модульною архітектурою. JS-Cookie – утиліта для роботи з HTTP cookies з простим API. Next-themes – бібліотека для управління темами з підтримкою системних налаштувань.

Вимоги до апаратного забезпечення для розробки та запуску клієнтської частини платформи включають конфігурацію для розробки з процесором Intel Core i5 або AMD Ryzen 5 не нижче 7-го покоління, оперативною пам'яттю мінімум 8 ГБ з рекомендованими 16 ГБ, дисковим простором мінімум 20 ГБ вільного місця на SSD, операційною системою Windows 10/11, macOS 11+ або Linux Ubuntu 20.04+, програмним забезпеченням Node.js 18+ та npm 8+ і стабільним широкосмуговим інтернет-з'єднанням.

Для запуску платформи необхідно Node.js-сумісне середовище або статичний хостинг, оперативна пам'ять мінімум 2 ГБ для невеликих інстансів, процесор мінімум 1 vCPU, дисковий простір мінімум 10 ГБ залежно від обсягу статичних ресурсів та стабільне мережеве підключення з мінімальною затримкою.

Для користувачів достатньо будь-якого сучасного веб-браузера Chrome 90+, Firefox 90+, Safari 14+ або Edge 90+ з увімкненим JavaScript та інтернет-з'єднанням мінімум 2 Мбіт/с.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Обрані технології дозволяють створити сучасний, високопродуктивний та масштабований фронтенд для платформи пошуку та управління активностями. Next.js 15 з App Router забезпечує оптимальний баланс між SEO та інтерактивністю з підтримкою intercepting routes, React 19 надає потужні інструменти для побудови компонентної архітектури, TypeScript покращує надійність та підтримуваність коду з повною типобезпекою, TailwindCSS дозволяє швидко створювати адаптивні інтерфейси з підтримкою тем. TanStack Query оптимізує роботу з серверним станом, React Hook Form забезпечує ефективну роботу з формами, Zod додає надійну валідацію даних, а ShadCN UI надає доступні та кастомізовані компоненти інтерфейсу. Zustand забезпечує легке управління локальним станом для фільтрів та специфічних компонентів, що разом створює потужну та гнучку архітектуру для розробки складних веб-додатків.

1.5 Розподілення обов'язків у команді

Розробка даного продукту є комплексною задачею, тому було прийнято рішення працювати в команді. Важливим кроком у роботі в групі є вибір підходу до управління проєктом. Створимо таблицю для порівняння серед різних характеристик (таблиця 1.2).

Таблиця 1.2

Порівняння підходів до управління проєктами

Характеристика	Водоспад	Гнучкий	Канбан	Принц
Базується на	Послідовності	Ітераціях	Потоці	Процесах
Гнучкість до змін	Дуже висока	Висока	Середня	Низька
Визначення ролей	Не вимагається	Не вимагається	Не вимагається	Власник продукту, менеджери, команда
Розмір команди	Будь-який	Малий	Будь-який	Великий
Потрібні інструменти	Будь-який трекер завдань	Будь-який трекер завдань	Jira або Trello	Jira або Trello
Легкість застосування	Висока	Середня	Висока	Низька

Проаналізувавши дані, легко відсіяти методології, що не підійдуть нам за обраними критеріями — принц не підходить за розміром команди, канбан вимагає платного програмного забезпечення, а водоспад вимагає послідовності, що може навпаки сповільнювати розробку продукту. Методом виключення ми обрали гнучкий підхід до управління проєктом. Фундаментом методології є цикл з 6 стадій: аналіз, дизайн, розробка, тестування, розгортання і рецензія (рис. 1.5).

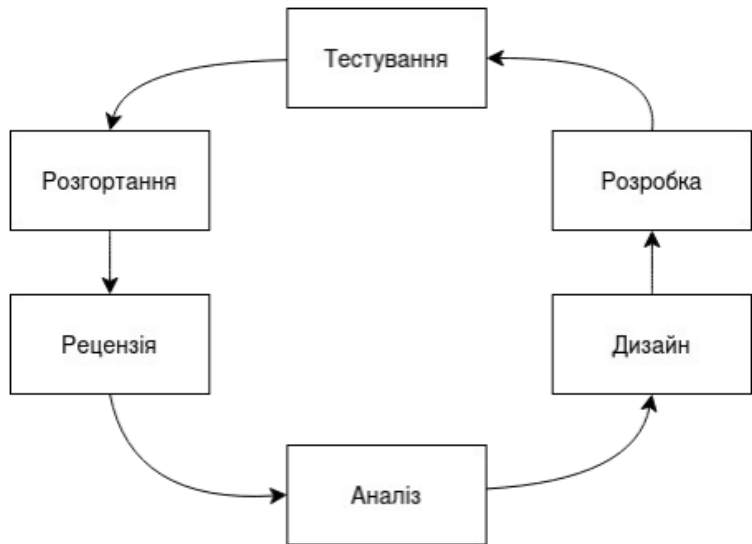


Рис. 1.5 – Цикл гнучкої методології

Наступним кроком є розподілення обов'язків. Незважаючи на те, що обраний підхід не вимагає розділених ролей, проте варто визначити роботу, що буде виконувати кожен учасник команди. Я, Пархомчук Руслан, маю більше досвіду та знань у сфері розробки клієнтської частини продукту, у той час як Корнійчук Віталій — серверної. Кінцевий розподіл обов'язків є наступним (таблиці 1.3 та 1.4).

Таблиця 1.3

Обов'язки Пархомчука Руслана

Роль	Обов'язки	Сильні сторони
Дизайнер	Дослідження очікувань користувачів, створення UI/UX, прототипування та візуалізація інтерфейсів	Креативність та розуміння користувацького досвіду
Інженер клієнтської частини	Аналіз нового функціоналу, визначення вимог, написання коду та мануальне тестування	Досвід у розробці клієнтської частини, застосування стандартів індустрії та розуміння трендів

Таблиця 1.4

Обов'язки Корнійчука Віталія

Роль	Обов'язки	Сильні сторони
Менеджер продукту	Визначення бачення продукту, формування вимог, пріоритезація завдань та комунікація	Розуміння потреб користувачів та зацікавленість у розвитку в сфері бізнесу
Інженер серверної частини	Аналіз нового функціоналу, визначення вимог, написання коду, написання інтеграційних тестів та документування API	Досвід у розробці серверної частини, застосування стандартів індустрії та розуміння трендів

Висновки до розділу 1

У першому розділі було проведено аналіз предметної області пошуку та управління активностями, визначено основні вимоги до розроблюваної платформи та досліджено існуючі аналоги.

Процес аналізу предметної області дозволив визначити ключові функціональні модулі, що має забезпечувати фронтенд-частина платформи: управління активностями з підтримкою одноразових та регулярних заходів, розкладами з детальним налаштуванням робочих днів та перерв, локаціями з інтерактивними планами розташування місць, користувацькими профілями, пошук та фільтрацію з територіальною підтримкою, систему бронювань та реакцій користувачів, а також статистичну аналітику для організаторів. Було сформульовано основні етапи розробки фронтенду, починаючи від створення дизайну з використанням ShadCN UI компонентів та архітектури з App Router до тестування та оптимізації інтерфейсу.

Дослідження аналогічних платформ, таких як Eventbrite, Meetup, Ticketmaster та Eventscase, дозволило виявити їхні сильні та слабкі сторони. Було визначено, що кожна з цих платформ має свою специфіку та цільову аудиторію, що впливає на функціональність та користувацький досвід. На основі проведеного аналізу були сформульовані ключові відмінності та переваги розроблюваної платформи, включаючи унікальну систему управління як одноразовими, так і регулярними активностями, інтерактивні плани місць з візуальним редактором, систему реакцій та взаємодії користувачів,

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

територіальну фільтрацію з ієрархічним пошуком та intercepting routes для плавної навігації.

Для побудови фронтенд-частини було обрано гібридну архітектуру, що поєднує серверний (SSR) та клієнтський (CSR) рендеринг на базі фреймворку Next.js 15 з App Router. Така архітектура забезпечує оптимальний баланс між SEO-оптимізацією та інтерактивністю користувацького інтерфейсу з підтримкою сучасних можливостей маршрутизації. Було розроблено загальну структуру фронтенд-додатку з чітким розділенням відповідальності між компонентами, включаючи рівні представлення, управління станом, сервісний шар, валідацію та утиліти.

Обґрунтовано вибір інструментальних засобів для розробки фронтенду, серед яких Next.js 15, React 19, TypeScript, TailwindCSS 4, TanStack Query, React Hook Form, ShadCN UI на базі Radix UI, Zod для валідації схем та Zustand для локального стану. Для кожного з цих інструментів визначено ключові переваги та особливості використання в контексті розроблюваної платформи. Також сформульовано вимоги до апаратного забезпечення для розробки, запуску та використання платформи з урахуванням сучасних стандартів продуктивності.

Проведено аналіз методологій управління проєктами та обрано гнучкий підхід (Agile) як найбільш відповідний для розробки платформи в команді з двох розробників. Визначено розподіл обов'язків між учасниками команди, де фронтенд-розробка, UI/UX дизайн та координація проєкту покладені на автора роботи, що забезпечує ефективну організацію процесу розробки та чітку відповідальність за різні аспекти створення платформи.

На основі проведеного дослідження можна зробити висновок, що обрані технології та архітектурні рішення дозволяють створити сучасний, високопродуктивний та масштабований фронтенд для платформи пошуку та управління активностями, який відповідатиме всім поставленим вимогам та забезпечить зручний користувацький досвід для різних категорій користувачів. Система типізації через TypeScript та Zod гарантує надійність коду, а використання intercepting routes та TanStack Query забезпечує плавну взаємодію

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

з мінімальними затримками. Застосування Agile методології та чіткий розподіл обов'язків у команді створюють оптимальні умови для ефективної розробки та своєчасного виконання поставлених завдань.

Отримані результати аналізу та проектування становлять основу для подальшої розробки та реалізації клієнтської частини платформи пошуку та управління активностями з урахуванням сучасних вимог до веб-додатків та користувацького досвіду.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

2. ПРОЕКТУВАННЯ ФРОНТЕНД-ЧАСТИНИ ПЛАТФОРМИ РОЗМІЩЕННЯ ПОДІЙ ТА ЗАХОДІВ

2.1 Визначення варіантів використання та компонентної структури системи

Платформа управління заходами та бронюваннями надає веб-інтерфейс, через який користувачі можуть створювати різні типи активностей, керувати локаціями та розкладами, здійснювати бронювання та взаємодіяти з системою через реакції та статистику. Функціональність платформи орієнтована на два основних типи користувачів: організатори заходів та звичайні відвідувачі.

Організатори заходів мають розширені можливості для створення та управління активностями, включаючи одноразові та регулярні заходи, визначення локацій з детальними планувальними схемами, створення розкладів роботи та аналіз статистичних даних. Звичайні користувачі можуть переглядати доступні заходи, здійснювати бронювання місць або часових слотів, залишати реакції та фільтрувати активності за різними критеріями.

Для наочного представлення взаємодії між користувачами та системою було створено діаграму варіантів використання (рис. 2.1).

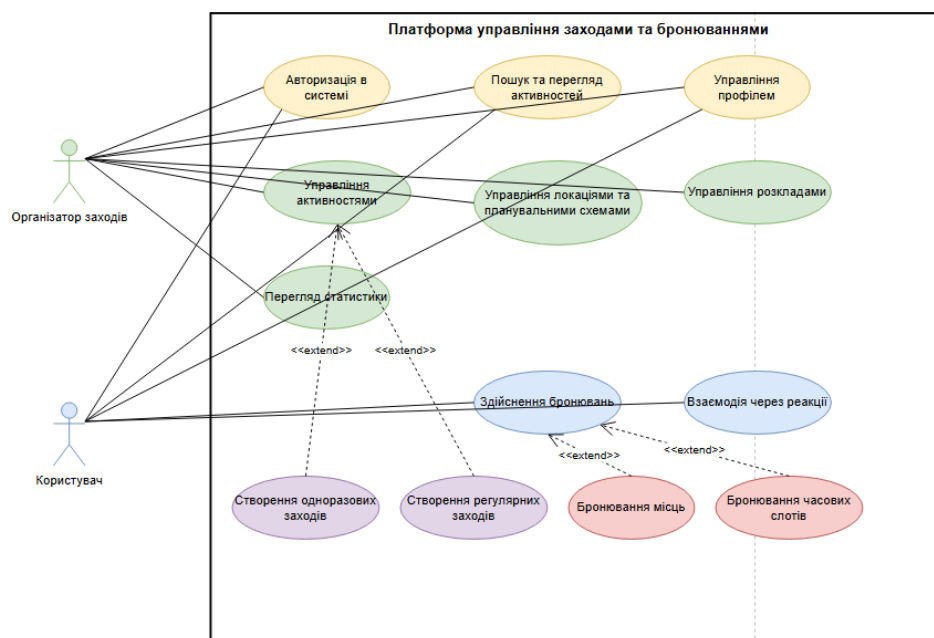


Рис. 2.1 – Діаграма варіантів використання

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

Опис основних варіантів використання наведено в таблицях 2.1-2.9.

Таблиця 2.1

Опис прецеденту «Авторизація в системі»

Назва варіанту використання	Авторизація в системі
Основні актори	Організатор заходів, користувач
Короткий опис	Користувачі можуть авторизуватись в системі за допомогою логіну/паролю для отримання JWT-токенів та доступу до функціоналу відповідно до їх ролі

Таблиця 2.2

Опис прецеденту «Управління активностями»

Назва варіанту використання	Управління активностями
Основні актори	Організатор заходів
Короткий опис	Організатори можуть створювати одноразові та регулярні активності, вказуючи назву, опис, локацію, часові рамки, медіафайли та додаткові опції для регулярних заходів

Таблиця 2.3

Опис прецеденту «Пошук та перегляд активностей»

Назва варіанту використання	Пошук та перегляд активностей
Основні актори	Організатор заходів, користувач
Короткий опис	Користувачі можуть шукати та переглядати активності за різними критеріями, включаючи назву, тип заходу, територіальне розташування та інші параметри

Таблиця 2.4

Опис прецеденту «Управління локаціями та планувальними схемами»

Назва варіанту використання	Управління локаціями та планувальними схемами
-----------------------------	---

Продовження таблиці 2.4

Основні актори	Організатор заходів
Короткий опис	Організатори можуть створювати локації з детальними планувальними схемами, включаючи розміщення місць, зон та візуальних об'єктів для бронювання

Таблиця 2.5

Опис прецеденту «Управління розкладами»

Назва варіанту використання	Управління розкладами
Основні актори	Організатор заходів
Короткий опис	Організатори можуть створювати та редагувати розклади для регулярних активностей, визначаючи робочі дні, години, перерви та тривалість бронювання

Таблиця 2.6

Опис прецеденту «Здійснення бронювань»

Назва варіанту використання	Здійснення бронювань
Основні актори	Користувач
Короткий опис	Користувачі можуть бронювати місця для одноразових заходів або часові слоти для регулярних активностей з можливістю додавання приміток

Таблиця 2.7

Опис прецеденту «Взаємодія через реакції»

Назва варіанту використання	Перегляд статистики
Основні актори	Організатор заходів
Короткий опис	Організатори можуть переглядати детальну статистику бронювань за різними періодами з можливістю різної фільтрації

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Таблиця 2.8

Опис прецеденту «Перегляд статистики»

Назва варіанту використання	Перегляд статистики
Основні актори	Організатор заходів
Короткий опис	Організатори можуть переглядати детальну статистику бронювань за різними періодами з можливістю фільтрації за днями, тижнями або місяцями

Таблиця 2.9

Опис прецеденту «Управління профілем»

Назва варіанту використання	Управління профілем
Основні актори	Організатор заходів, користувач
Короткий опис	Користувачі можуть редагувати свій профіль, включаючи особисту інформацію, зображення профілю та додаткові налаштування

Основна мета розробки фронтенд-частини платформи управління заходами та бронюваннями полягає у створенні інтуїтивно зрозумілого та функціонального інтерфейсу, який забезпечує ефективну взаємодію користувачів з системою. Платформа орієнтована на надання комплексного рішення для організації заходів різного масштабу з можливостями детального планування та управління бронюваннями.

Вимоги користувачів:

Внутрішній користувач – організатор заходів має потребу в авторизації через безпечну систему автентифікації, створенні та редагуванні як одноразових, так і регулярних активностей з можливістю додавання медіафайлів та детального опису. Організатор повинен мати можливість управляти локаціями з створенням планувальних схем та розміщенням місць для бронювання, створювати та редагувати складні розклади з визначенням робочих днів, годин та перерв. Також

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

необхідний перегляд детальної статистики бронювань з можливістю фільтрації за різними періодами та інтервалами, управління опціями для регулярних активностей та зміна кольорової теми інтерфейсу.

Внутрішній користувач – звичайний відвідувач потребує можливості реєстрації та авторизації в системі, перегляду списку доступних активностей з детальною інформацією, здійснення пошуку та фільтрації заходів за різними критеріями включаючи територіальне розташування. Користувач повинен мати змогу бронювати місця для одноразових заходів або часові слоти для регулярних активностей, залишати реакції на активності, переглядати свої бронювання та реакції, редагувати власний профіль та налаштовувати кольорову тему інтерфейсу.

Функціональні вимоги:

Система повинна забезпечувати надійну авторизацію користувачів з використанням JWT-токенів та автоматичним оновленням сесій. Необхідна можливість створення та управління двома типами активностей: одноразовими з конкретною датою та часом проведення, та регулярними з гнучкими розкладами та опціями бронювання. Система має забезпечувати розширений пошук активностей за назвою, типом, територіальним розташуванням та іншими параметрами.

Важливою вимогою є можливість створення та управління локаціями з детальними планувальними схемами, включаючи розміщення візуальних об'єктів, місць та зон для бронювання. Система повинна підтримувати створення складних розкладів для регулярних активностей з визначенням робочих днів, годин, перерв та тривалості бронювання.

Необхідна реалізація системи бронювань з підтримкою різних типів: бронювання конкретних місць для одноразових заходів та часових слотів для регулярних активностей. Система має включати функціональність для залишення реакцій на активності та перегляду детальної статистики з можливістю фільтрації за різними періодами.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

Обов'язковими є відображення детальної інформації про активності з медіагалереєю, налаштування світлої та темної теми інтерфейсу, повна адаптивність для мобільних пристроїв та підтримка множинного завантаження медіафайлів.

Нефункціональні вимоги:

Щодо сприйняття, час освоєння веб-додатку для нових користувачів не повинен перевищувати 10 хвилин, для досвідчених користувачів менше 3 хвилин. Час відповіді системи на дії користувача має бути не більше 2 секунд для основних операцій. Інтерфейс повинен бути інтуїтивно зрозумілим та не вимагати від користувача спеціальної підготовки.

Надійність системи передбачає безперебійну доступність веб-додатку протягом доби з мінімальним часом простою для технічного обслуговування.

Продуктивність системи має забезпечувати одночасну роботу великої кількості користувачів без деградації швидкості відповіді. Відображення результатів пошуку та фільтрації не повинно перевищувати 3 секунд навіть при великій кількості даних.

Системні вимоги:

Вимоги до середовища виконання включають наявність сучасного веб-браузера з підтримкою JavaScript ES6 та вище, мінімум 2 Гб оперативної пам'яті та стабільне інтернет-з'єднання зі швидкістю не менше 1 Мбіт/с.

Вимоги до серверної частини передбачають швидку обробку API запитів з підтримкою великої кількості одночасних з'єднань та ефективне управління файловими ресурсами.

Аналіз функціональних вимог дозволив визначити основні компоненти фронтенд-частини платформи. Для React-додатку з використанням Next.js застосовується компонентна архітектура, яка забезпечує модульність та масштабованість системи. На рис. 2.2 представлено діаграму основних компонентів системи.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

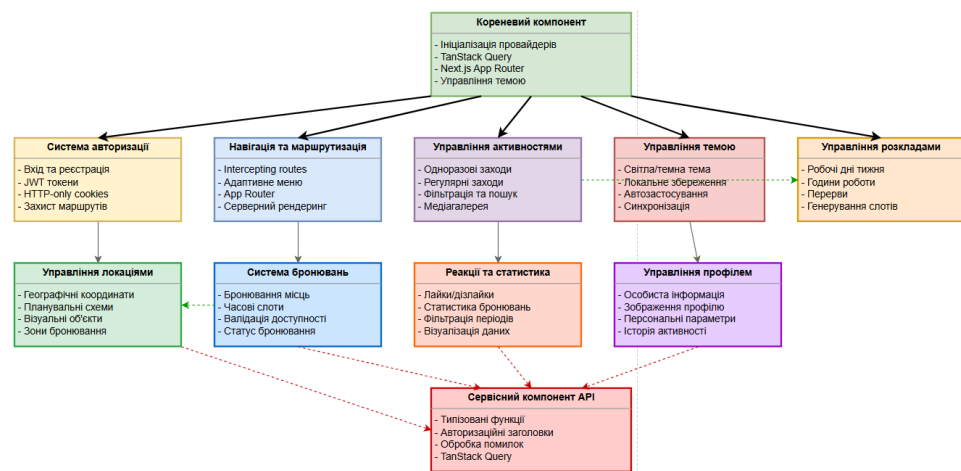


Рис. 2.2 – Діаграма основних компонентів системи

Основні компоненти системи організовані в ієрархічну структуру з чітким розділенням відповідальності.

Кореневий компонент виступає як точка входу для всього додатку та відповідає за ініціалізацію глобальних провайдерів, включаючи TanStack Query для управління серверним станом, налаштування маршрутизації через Next.js App Router та управління глобальними налаштуваннями теми. Цей компонент забезпечує початкову конфігурацію додатку та створює основу для всіх інших компонентів.

Компонент системи авторизації реалізує всі аспекти автентифікації користувачів, включаючи процеси входу та реєстрації через модальні вікна або окремі сторінки, безпечне зберігання JWT-токенів в HTTP-only cookies, автоматичне оновлення токенів при закінченні терміну дії та захист маршрутів через middleware. Цей компонент забезпечує безпечну взаємодію з API та управління сесіями користувачів.

Компонент навігації та маршрутизації відповідає за створення інтуїтивного інтерфейсу переміщення між сторінками додатку, реалізує intercepting routes для модальних вікон при перегляді деталей активностей, забезпечує адаптивне меню користувача з доступом до персональних сторінок та налаштувань. Особливістю є використання Next.js App Router для оптимізації продуктивності та серверного рендерингу.

Компонент управління активностями є одним з найскладніших та включає функціональність створення та редагування як одноразових, так і регулярних заходів, розширену систему фільтрації з підтримкою пошуку за назвою, типом та територіальним розташуванням, відображення детальної інформації з медіагалереєю та можливістю бронювання. Цей компонент інтегрується з системою управління опціями для регулярних активностей.

Компонент управління розкладами надає інструменти для створення складних робочих розкладів з визначенням робочих днів тижня, годин роботи та перерв, автоматичного генерування доступних часових слотів на основі тривалості бронювання та візуалізації розкладу у зручному форматі. Включає валідацію часових інтервалів та запобігання конфліктам у розкладі.

Компонент управління локаціями та планувальними схемами реалізує створення детальних локацій з географічними координатами, завантаження медіафайлів та маршрутної інформації, створення інтерактивних планувальних схем з розміщенням візуальних об'єктів різних типів включаючи окремі місця та зони бронювання. Забезпечує збереження координат та розмірів об'єктів для точного відображення.

Компонент системи бронювань підтримує два типи бронювань: конкретних місць для одноразових заходів та часових слотів для регулярних активностей, включає валідацію доступності місць та часових слотів, відображення статусу бронювання та можливість додавання приміток. Інтегрується з компонентами активностей та планувальних схем.

Компонент управління реакціями та статистикою дозволяє користувачам залишати реакції типу лайк або дізлайк на активності, забезпечує перегляд детальної статистики бронювань для організаторів з можливістю фільтрації за різними періодами та інтервалами, включає візуалізацію даних через графіки та діаграми.

Компонент управління профілем надає функціональність редагування особистої інформації користувачів, завантаження та управління зображенням профілю, налаштування персональних параметрів та доступ до історії

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

активності. Включає валідацію даних профілю та безпечне оновлення інформації.

Сервісний компонент API централізує всі взаємодії з серверним API через типізовані функції для кожного ендпоінту, реалізує автоматичне додавання авторизаційних заголовків, обробку помилок та повторні спроби запитів, управління FormData для завантаження файлів та інтеграцію з TanStack Query для кешування та оптимізації запитів.

Компонент управління темою забезпечує перемикання між світлою та темною темами інтерфейсу, збереження налаштувань користувача в локальному сховищі, автоматичне застосування теми при завантаженні додатку та синхронізацію налаштувань між різними сеансами користувача.

Компонентна архітектура React з використанням TypeScript забезпечує строгу типізацію та надійність коду. Використання сучасних хуків React включаючи useState, useEffect, useContext та спеціалізованих бібліотек як TanStack Query для серверного стану, React Hook Form для управління формами та Zustand для локального стану дозволяє ефективно організувати логіку додатку.

Архітектурний підхід базується на принципі розділення відповідальності, де UI-компоненти зосереджуються на презентації даних, а бізнес-логіка виділена в окремі хуки та сервісні функції. Використання ShadCN UI компонентів забезпечує консистентність дизайну та прискорює розробку, а intercepting routes надають плавний користувацький досвід при навігації.

2.2 Розробка моделей даних та взаємодії з API

Для ефективної взаємодії фронтенд-частини платформи з API необхідно розробити комплексну систему моделей даних та валідації. Використання TypeScript забезпечує строгу типізацію на етапі розробки, а бібліотека Zod додає потужну валідацію даних під час виконання, що значно підвищує надійність додатку та зменшує кількість помилок.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Архітектура моделей даних побудована на принципі розділення інтерфейсів для отримання даних з API та для відправки запитів на сервер. Це дозволяє чітко розмежувати read-only поля, що генеруються сервером, та поля, які користувач може редагувати.

Для забезпечення взаємодії з бекендом було розроблено систему TypeScript інтерфейсів та еnumів, які точно відповідають структурам даних API. Система включає базові еnumи для стандартизації значень: WeekDayEnum для днів тижня з відображенням на індекси, TerritoryType для ієрархії територіальних одиниць, ActivityType для розмежування одноразових та регулярних активностей, ReactionTypeEnum для типів реакцій користувачів та StatsInterval для інтервалів статистичної аналітики.

Система автентифікації реалізована через комплексний набір інтерфейсів, представлених в таблиці 2.10.

Таблиця 2.10

Опис моделей автентифікації та авторизації

Інтерфейс	Опис	Основні поля
AuthTokens	Токени JWT для авторизації	access: string, refresh: string
LoginCredentials	Дані для входу в систему	username: string, password: string
SignupCredentials	Дані для реєстрації	username: string, password: string
TokenObtainPair	Запит для отримання JWT-токенів	username: string, password: string
TokenObtainPairResponse	Відповідь з JWT-токенами	access: string, refresh: string
TokenRefresh	Запит для оновлення токена	refresh: string
TokenRefreshResponse	Відповідь з оновленим токеном	access: string
TokenBlacklist	Запит для блокування токена	refresh: string
SignUp	Запит на реєстрацію	username: string, password: string
SignUpResponse	Відповідь після реєстрації	pk: number, username: string, access: string, refresh: string

Ці інтерфейси забезпечують повний цикл управління автентифікацією, включаючи отримання токенів, їх оновлення та безпечне завершення сесій. JWT-токени зберігаються в HTTP-only cookies для забезпечення максимальної безпеки.

Основні сутності платформи організовані в ієрархічну структуру з чіткими зв'язками між компонентами, що відображено в таблиці 2.11.

Таблиця 2.11

Опис основних моделей даних

Інтерфейс/тип	Опис	Основні поля
User	Користувач системи	pk: number, username: string, image?: string, display_name?: string, description?: string, birth_date?: string
UserRequest	Дані для створення/оновлення користувача	username: string, display_name?: string, image?: File, website_links: string[]
FileCustom	Медіафайл з метаданими	file: string, order: number, created_at: string
Point	Геокоординати	latitude: number, longitude: number
Territory	Територіальна одиниця	pk: number, code: string, name: string, type: TerritoryType, upper_level: Territory
TimeRange	Діапазон часу	string (формат: HH:MM-HH:MM)
WorkDay	Робочий день	week_day: WeekDayEnum, work_hours: TimeRange, break_hours: TimeRange[], slots: Slot
Schedule	Розклад	pk: number, work_days: WorkDay[], booking_duration: string, created_by: User, activities: Activity[]
VisualObject	Базовий візуальний об'єкт	pk: number, name: string, position: [number, number], size: [number, number], rotation: number
Seat	Місце для бронювання	extends VisualObject, is_booked: boolean, type: "Seat"

Продовження таблиці 2.11

SeatZone	Зона з множинними місцями	extends VisualObject, seat_amount: number, booked_seats: number, type: "SeatZone"
Layout	Планувальна схема	pk: number, name: string, size: [number, number], visual_objects: PolymorphicVisualObject[], venue: NestedVenue
Venue	Локація проведення заходів	pk: number, name: string, description: string, location: string, media: FileCustom[], layouts: Layout[]

Особливу архітектурну цінність має система поліморфних типів для візуальних об'єктів, яка дозволяє працювати з різними типами елементів планувальних схем через єдиний інтерфейс. Тип PolymorphicVisualObject об'єднує базові візуальні об'єкти, окремі місця та зони бронювання, використовуючи discriminated union з полем type як дискримінатором.

Система активностей реалізована через складну ієрархію типів, представлену в таблиці 2.12.

Таблиця 2.12

Опис основних моделей даних

Інтерфейс/тип	Опис	Особливості
BaseActivity	Базова активність	name: string, description: string, venue: NestedVenue, media: FileCustom[], is_published: boolean
OneTimeActivity	Одноразовий захід	extends BaseActivity, type: "OneTimeActivity", date: string, time_range: TimeRange, layout: Layout
RecurringActivity	Регулярний захід	extends BaseActivity, type: "RecurringActivity", schedule: NestedSchedule, duration: string, options: NestedOption[]

Продовження таблиці 2.12

Activity	Загальний тип активності	OneTimeActivity або RecurringActivity
NestedOption	Опція для регулярних активностей	pk: number, name: string, description: string, activity_pk: number
Option	Повна модель опції	extends NestedOption, activity: RecurringActivity, created_by: User
Reaction	Реакція користувача	pk: number, type: ReactionTypeEnum, activity: Activity, created_by: User
BaseBooking	Базове бронювання	pk: number, activity: Activity, booked_by: User, booked_at: string
OneTimeActivityBooking	Бронювання місця	extends BaseBooking, resourcetype: "OneTimeActivityBooking", seat: PolymorphicVisualObject
RecurringActivityBooking	Бронювання часового слоту	extends BaseBooking, resourcetype: "RecurringActivityBooking", time: string, date: string, option: Option

Дискримінована система типів дозволяє TypeScript автоматично визначати тип активності на основі поля type та надавати відповідні поля для автодоповнення, що значно зменшує ймовірність помилок при розробці.

Система запитів до API організована з урахуванням різних типів операцій та специфіки передачі даних, що показано в таблиці 2.13.

Таблиця 2.13

Опис інтерфейсів запитів до API

Інтерфейс/тип	Опис	Особливості
VenueRequest	Запит створення/оновлення локації	media_files: File[], route_files?: File[], website_links: string
ScheduleRequest	Запит створення/оновлення розкладу	work_days: WorkDayRequest[], booking_duration: string
LayoutRequest	Запит створення/оновлення схеми	size: string, visual_objects: PolymorphicVisualObjectRequest[], venue_pk: number

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

Продовження таблиці 2.13

OneTimeActivityRequest	Запит створення одноразової активності	type: "OneTimeActivity", date: string, time_range: TimeRange, layout_pk: number
RecurringActivityRequest	Запит створення регулярної активності	type: "RecurringActivity", duration: string, schedule_pk: number
BookingFormValues	Універсальні дані форми бронювання	activity_pk: number, resource_type: string, seat_pk?: number, time?: string, date?: string
PaginatedResponse<T>	Пагінована відповідь	count: number, next: string, previous: string, results: T[]
ActivityQueryParams	Параметри фільтрації активностей	name?: string, search?: string, type?: string[], within_territory?: string
StatsQueryParams	Параметри статистичних запитів	start: string, end: string, interval: StatsInterval, page?: number

Важливою особливістю є використання різних типів для координат та розмірів в залежності від контексту: API очікує строкові представлення типу "100,200", а внутрішня логіка працює з типізованими масивами [number, number].

Валідація даних реалізована через комплексну систему Zod схем, які забезпечують перевірку коректності даних на клієнтській стороні перед відправкою на сервер. Основні схеми включають loginSchema та signupSchema для автентифікації з перевіркою відповідності паролів, userProfileSchema для редагування профілю з валідацією форматів дат та URL-адрес, timeRangeSchema для перевірки часових діапазонів з логічною валідацією послідовності, venueSchema для створення локацій з валідацією координат та файлів.

Схеми для візуальних об'єктів включають складну валідацію позицій, розмірів та кутів повороту з урахуванням обмежень планувальних схем. Поліморфна схема polymorphicVisualObjectSchema використовує discriminated union для валідації різних типів об'єктів залежно від поля type.

Активності валідуються через enhancedActivitySchema, яка включає перевірку залежних полів в залежності від типу активності, валідацію дат

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

майбутніх заходів, форматів часових діапазонів та обов'язкових полів для кожного типу.

Система бронювань використовує bookingSchema з умовною валідацією полів залежно від типу бронювання, перевіркою доступності дат та часових слотів, а також валідацією відповідності між активністю та типом бронювання.

Статистичні запити валідуються через statsQuerySchema з перевіркою логічності часових періодів, допустимих інтервалів та форматів дат.

Взаємодія з API базується на RESTful принципах з використанням типізованих функцій для кожного ендпоінту. Система fetchWithAuth забезпечує автоматичне додавання JWT-токенів до заголовків запитів, обробку помилок автентифікації з автоматичним оновленням токенів, правильне формування FormData для завантаження файлів та уніфіковану обробку помилок API.

Інтеграція з TanStack Query забезпечує ефективне кешування відповідей API, автоматичну реінвалідацію кешу при зміні даних, оптимістичні оновлення для покращення користувацького досвіду, фонове оновлення даних та обробку станів завантаження і помилок.

Розроблена система типів інтегрується з React Hook Form для забезпечення типізованих форм з автоматичною валідацією, що дозволяє використовувати resolver з Zod схемами для seamless валідації форм, автоматичне мапування помилок валідації на поля форм та типізацію значень форм відповідно до схем.

Система управління станом реалізована через комбінацію TanStack Query для серверного стану та Zustand для локального стану, що забезпечує чітке розділення відповідальності та оптимальну продуктивність додатку.

Впровадження цієї комплексної системи типізації та валідації забезпечує високу надійність коду, значне зменшення runtime помилок, покращену розробницьку експертизу через автодоповнення та статичний аналіз, автоматичну документацію API через типи та легку підтримку й розширення функціональності додатку.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

2.3 Проектування та реалізація алгоритмів роботи системи

Розроблена платформа управління заходами та бронюваннями реалізує складну бізнес-логіку через набір взаємопов'язаних алгоритмів, які забезпечують ефективну взаємодію користувачів з системою. Архітектура алгоритмів побудована на принципах реактивного програмування з використанням React хуків та оптимізованого управління станом через TanStack Query та Zustand.

Загальний алгоритм роботи з платформою базується на маршрутизації через Next.js middleware. Після успішної аутентифікації система автоматично визначає права доступу користувача та надає відповідний функціонал. Загальний алгоритм роботи з платформою представлено на рис. 2.3.

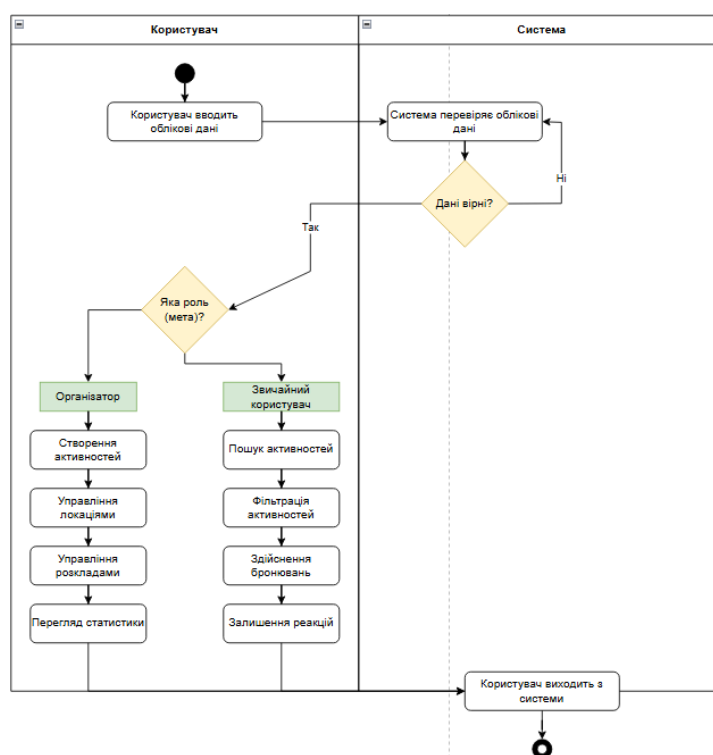


Рис. 2.3 – Загальний алгоритм роботи з платформою

Система підтримує диференційований доступ до функціональності залежно від ролі користувача. Організатори заходів мають розширені можливості, включаючи перегляд та управління власними активностями з можливістю створення як одноразових, так і регулярних заходів, комплексне управління локаціями з створенням детальних планувальних схем та розміщенням візуальних об'єктів для бронювання, створення та редагування складних розкладів з визначенням робочих днів, годин та перерв, створення та

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

управління опціями для регулярних активностей, доступ до детальної статистики бронювань з можливістю аналізу за різними періодами та публікацію або приховування активностей для контролю видимості.

Звичайні користувачі мають доступ до функцій перегляду та взаємодії, включаючи перегляд списку доступних активностей з детальною інформацією та медіагалереєю, розширені можливості пошуку та фільтрації за назвою, типом активності та територіальним розташуванням, здійснення бронювань як окремих місць для одноразових заходів, так і часових слотів для регулярних активностей, залишення реакцій типу лайк або дізлайк на активності, перегляд власних бронювань та реакцій через персональні сторінки, редагування профілю з можливістю завантаження аватара та додавання особистої інформації.

Алгоритм створення активностей представлено на рис. 2.4.



Рис. 2.4 – Діаграма активності процесу створення події

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

Створення активності починається з вибору типу заходу через дискримінований інтерфейс, який динамічно адаптує форму залежно від обраного типу. Користувач вводить базову інформацію, включаючи назву з обмеженням до 64 символів, детальний опис заходу, завантаження медіафайлів з валідацією типу та розміру, додавання веб-посилань з перевіркою URL-формату.

Для локації система надає два варіанти: вибір існуючої локації зі списку власних локацій користувача або створення нової локації безпосередньо в процесі створення активності. При створенні нової локації відбувається валідація координат, завантаження медіафайлів та маршрутної інформації.

Одноразові активності вимагають вказання конкретної дати з валідацією на майбутні дати, часового діапазону у форматі HH:MM-HH:MM з перевіркою логічності послідовності та вибору планувальної схеми з відповідної локації. Регулярні активності потребують вибору або створення розкладу з визначенням робочих днів та годин, вказання тривалості у форматі HH:MM:SS та можливості додавання опцій після створення основної активності.

Система виконує комплексну валідацію на клієнтській стороні через Zod схеми перед відправкою даних на сервер. При виявленні помилок користувач отримує деталізовані повідомлення з вказівкою на конкретні поля та рекомендації щодо виправлення. Успішне створення активності супроводжується автоматичною інвалідацією кешу TanStack Query для оновлення відповідних списків.

Пошук та фільтрація активностей реалізовані через складний алгоритм з підтримкою множинних критеріїв та оптимізованими запитами до API. Алгоритм цього процесу представлено на рис. 2.5.

Система фільтрації підтримує текстовий пошук по назвах активностей з використанням debouncing для оптимізації кількості запитів, фільтрацію за типом активностей з можливістю множинного вибору, територіальну фільтрацію через інтеграцію з ендпоінтом territories/autocomplete з підтримкою автодоповнення та ієрархічної структури територій.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

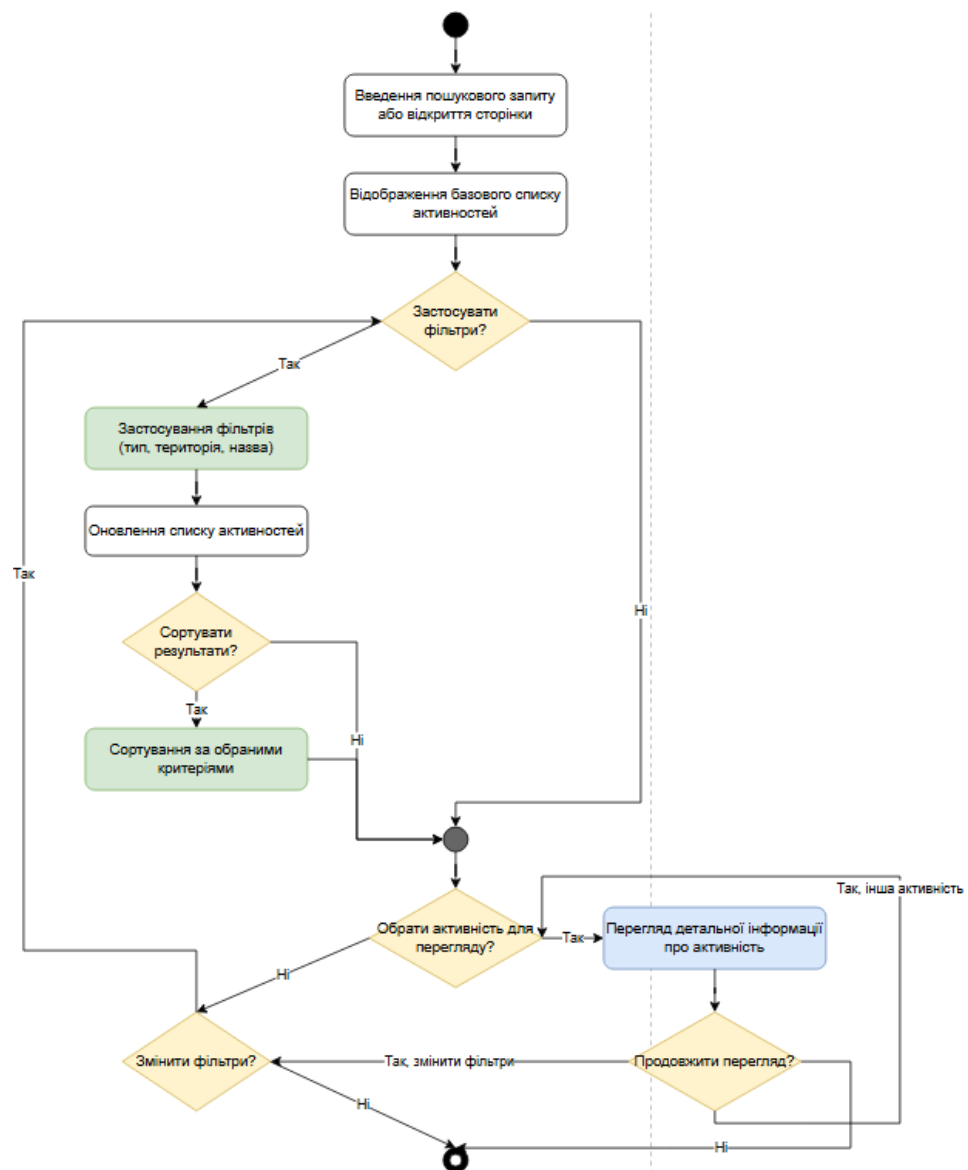


Рис. 2.5 – Діаграма активності процесу пошуку та фільтрації подій

Стан фільтрів управляється через Zustand store, що забезпечує збереження налаштувань між сеансами та синхронізацію з URL-параметрами для можливості збереження посилань на відфільтровані результати. Пагінація реалізована з підтримкою динамічного завантаження та оптимізації продуктивності при великих обсягах даних.

Система кешування TanStack Query автоматично оптимізує повторні запити з однаковими параметрами та забезпечує фонове оновлення даних для актуальності інформації. Результати пошуку відображаються у вигляді карток з превью медіафайлів та основною інформацією про активності.

Управління локаціями та планувальними схемами включає комплексний набір алгоритмів для роботи з географічними координатами, медіафайлами та візуальними об'єктами. Алгоритм цього процесу представлено на рис. 2.6.

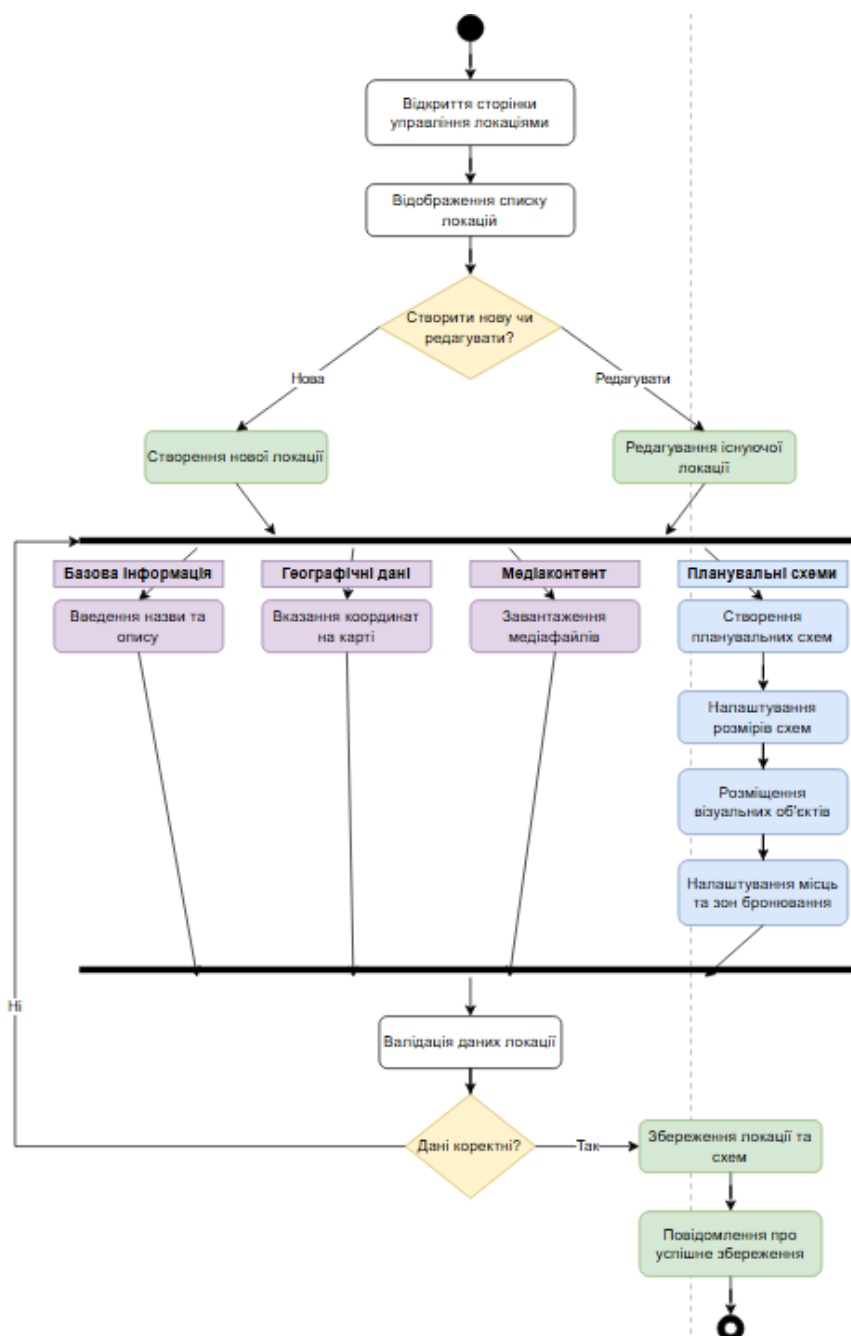


Рис. 2.6 – Діаграма активності процесу управління локаціями

Створення локації починається з введення базової інформації та валідації географічних координат у форматі latitude,longitude з перевіркою допустимих діапазонів значень. Система підтримує завантаження множинних медіафайлів з автоматичним визначенням типу, валідацією розміру та оптимізацією для веб-

відображення. Додатково можливе завантаження маршрутних файлів для навігації до локації.

Планувальні схеми створюються через інтерактивний редактор з підтримкою різних типів візуальних об'єктів. Базові візуальні об'єкти служать для оформлення та структурування схеми, окремі місця призначені для точного бронювання конкретних позицій, зони бронювання дозволяють групове бронювання з автоматичним підрахунком доступних місць.

Алгоритм управління розкладами забезпечує гнучке планування регулярних активностей з підтримкою складних часових структур. Алгоритм цього процесу представлено на рис. 2.7.

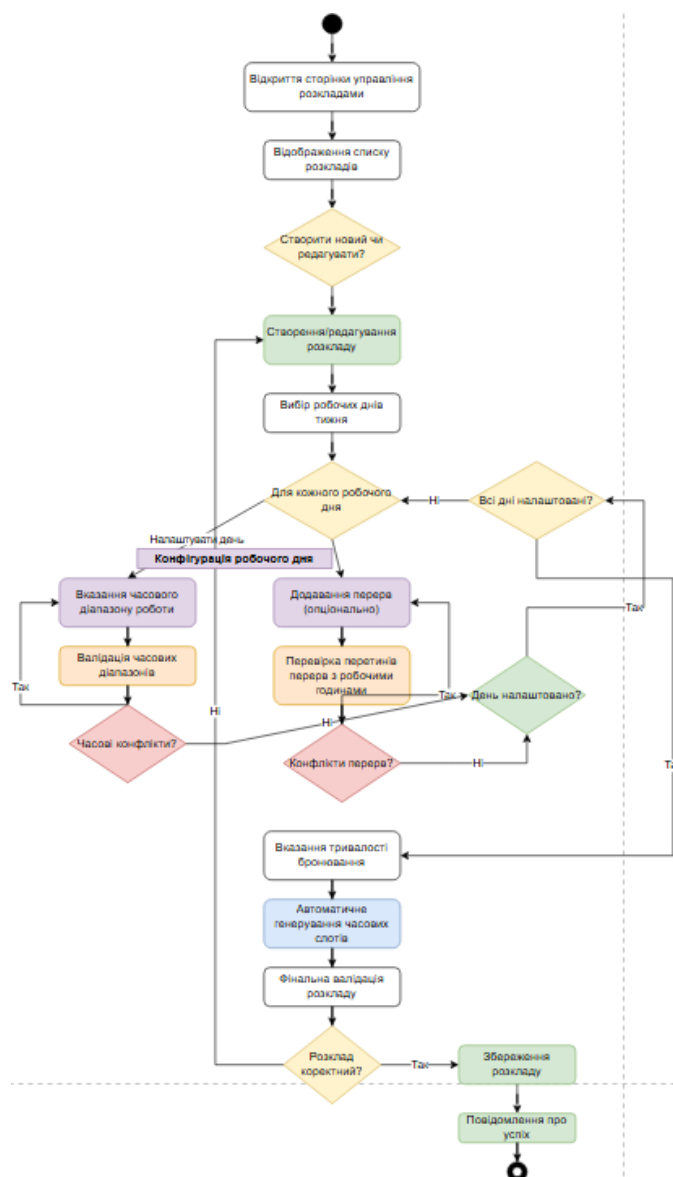


Рис. 2.7 – Діаграма активності процесу управління розкладами

Створення розкладу включає вибір робочих днів тижня з можливістю множинного вибору та автоматичною перевіркою унікальності. Для кожного робочого дня визначаються робочі години у форматі HH:MM-HH:MM з валідацією логічності часових діапазонів, перерви як масив часових діапазонів з перевіркою перетинів з робочими годинами, тривалість одного бронювання у форматі HH:MM:SS для автоматичного генерування доступних слотів.

Система автоматично розраховує доступні часові слоти на основі робочих годин, перерв та тривалості бронювання, виключаючи періоди, що перетинаються з перервами або вже заброньовані слоти. Алгоритм оптимізований для швидкого розрахунку навіть при складних розкладах з множинними перервами.

Візуальне відображення розкладу реалізоване через інтерактивні компоненти для зручного редагування часових діапазонів. Система валідації запобігає створенню конфліктів у розкладі та надає користувачу зрозумілі повідомлення про помилки.

Алгоритм системи бронювань підтримує два основні типи бронювань з різною логікою обробки. Для одноразових активностей користувач вибирає конкретне місце або зону з планувальної схеми, система перевіряє доступність обраного місця, резервує місце та оновлює стан схеми. Для регулярних активностей користувач обирає дату з доступних робочих днів, часовий слот з автоматично згенерованого списку, опцію активності з деталізацією особливостей, додає примітки при необхідності.

Система валідації бронювань включає перевірку доступності дати та часу, відповідності типу бронювання типу активності, наявності вільних місць або слотів та прав доступу користувача. При успішному бронюванні система автоматично оновлює стан доступності та надсилає підтвердження користувачу.

Алгоритм реакцій дозволяє користувачам виражати ставлення до активностей через систему лайків та дізлайків. Система автоматично відстежує попередні реакції користувача та дозволяє їх зміну або відміну. Статистика реакцій агрегується для відображення загальної популярності активностей.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

Статистична аналітика реалізована через складний алгоритм агрегації даних з підтримкою різних часових інтервалів та фільтрів. Система розраховує загальну кількість бронювань, розподіл між типами активностей, кількість унікальних та повторних клієнтів, тренди популярності за періодами. Результати візуалізуються через інтерактивні графіки з можливістю деталізації.

Усі алгоритми системи оптимізовані для продуктивності через кешування критичних запитів, debouncing для пошукових запитів, loading states для покращення користувацького досвіду та error boundaries для обробки помилок. Система забезпечує адаптивний дизайн.

2.4 Реалізація функціоналу системи платформи з розміщення подій та заходів

В цьому розділі розглянемо основні аспекти реалізації функціоналу платформи з розміщення подій та заходів. Основна увага приділяється розробці фронтенд-частини, яка взаємодіє з API через відповідні сервіси та хуки для забезпечення повноцінної роботи користувачів з платформою.

Архітектура взаємодії з API:

Для взаємодії з API системи використовується підхід на основі сервісів, які інкапсулюють логіку мережевих запитів. Кожен сервіс відповідає за конкретну предметну область та надає методи для виконання CRUD операцій. Функція для отримання списку заходів демонструє ефективний підхід до формування параметрів запиту та обробки відповіді від сервера. Вона приймає необов'язкові параметри запиту, перетворює їх у рядок запиту та виконує аутентифікований запит до API. У разі помилки функція обробляє дані відповіді та генерує інформативне повідомлення про помилку (додаток А).

Для роботи з авторизацією використовується окремий сервіс, який надає методи для входу, реєстрації, оновлення токенів та виходу. Сервіс авторизації забезпечує безпечне зберігання JWT токенів у cookies та їх автоматичне додавання до заголовків запитів.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

Для забезпечення аутентифікації запитів до API реалізовано утилітарну функцію, яка включає токен доступу в заголовки запитів і обробляє випадки, коли токен доступу стає недійсним. Ця функція відіграє важливу роль у забезпеченні стабільної роботи додатку, автоматично обробляючи випадки експірації токенів. Спочатку вона отримує поточний токен доступу з локального сховища, додає токен доступу до заголовків запиту, визначає тип тіла запиту і відповідно налаштовує заголовки, виконує запит до API. Якщо отримує код відповіді 401, намагається оновити токен доступу. У разі успішного оновлення повторює запит з новим токеном, а якщо оновлення не вдається, перенаправляє користувача на сторінку входу (додаток А).

Такий підхід забезпечує безшовний користувацький досвід, дозволяючи автоматично обробляти ситуації, коли термін дії токена доступу закінчується під час сесії користувача.

Керування станом запитів та кешування даних:

Для ефективного керування станом запитів до API та кешування даних у додатку використовується бібліотека TanStack Query [3]. Вона дозволяє суттєво спростити логіку запитів і відповідей, а також ефективно керувати станом завантаження, помилок та результатів.

Для роботи з заходами реалізовано набір хуків, які інкапсулюють логіку взаємодії з API та керування станом [1]. Особливо важливими аспектами реалізації цих хуків є визначення ключів запитів для ефективного кешування та інвалідації, налаштування часу життя кешу, умовне виконання запитів та автоматична інвалідація пов'язаних запитів після мутацій (додаток А).

Наприклад, у хуку для створення заходу після успішного створення нового заходу автоматично інвалідуються запити зі списками заходів, що забезпечує актуальність даних у інтерфейсі користувача. Аналогічні принципи використовуються для всіх інших сутностей системи, включаючи розклади, майданчики, бронювання та опції.

Архітектура додатку та системні компоненти:

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Основними компонентами архітектури є компоненти сторінок, які відповідають за композицію компонентів та маршрутизацію, компоненти інтерфейсу, що представляють повторно використовувані елементи інтерфейсу, сервіси для інкапсуляції логіки взаємодії з API, хуки запитів, які використовують TanStack Query для керування станом запитів та кешування даних, утиліти для допоміжних функцій обробки даних, форматування, валідації та інших операцій, а також маршрутизацію, реалізовану за допомогою Next.js для забезпечення навігації по додатку [2].

Така архітектура забезпечує модульність, масштабованість та простоту підтримки додатку. Системну архітектуру представлено на рис. 2.8.

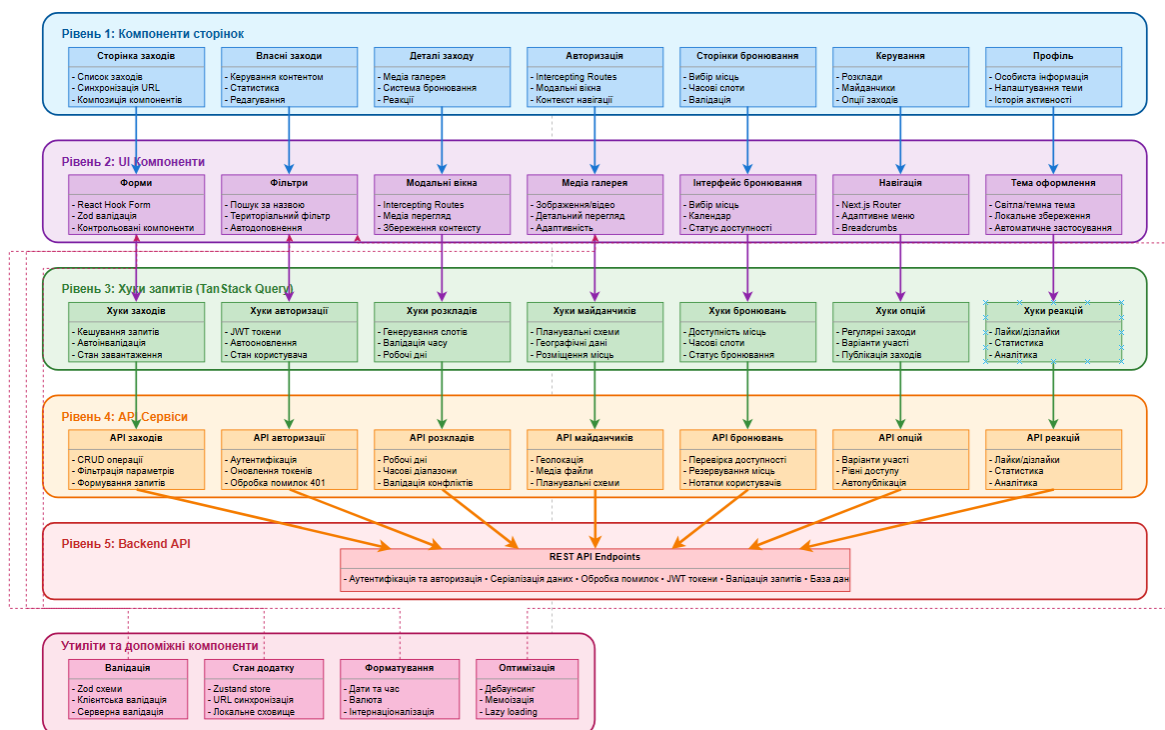


Рис. 2.8 – Діаграма системної архітектури додатку

Рис. 2.8 ілюструє взаємодію між основними компонентами системи. На верхньому рівні знаходяться компоненти сторінок, які координують роботу інших компонентів і забезпечують загальну структуру користувацького інтерфейсу. Рівень UI компонентів містить елементи інтерфейсу, такі як форми, кнопки, модалні вікна та інші елементи взаємодії [7]. Хуки запитів забезпечують зв'язок між компонентами та сервісами API, керуючи станом

завантаження та кешуванням відповідей. Сервіси взаємодіють безпосередньо з API, обробляючи аутентифікацію, серіалізацію даних та обробку помилок.

Структура компонентів та сторінок:

Для реалізації сторінок додатку застосовується компонентний підхід, де кожна сторінка складається з композиції менших компонентів із конкретними обов'язками. Такий підхід сприяє модульності та повторному використанню коду.

Сторінка зі списком заходів демонструє кілька важливих аспектів архітектури додатку. Синхронізація URL і стану компонента забезпечує збереження параметрів фільтрації в URL, що дозволяє ділитися посиланнями та реалізувати навігацію назад. Композиція компонентів означає, що сторінка складається з менших компонентів, таких як заголовки сторінки, фільтри та сітка заходів, кожен з яких відповідає за конкретну функціональність. Використання TanStack Query означає, що запит на отримання заходів виконується через відповідний хук, який інкапсулює логіку запиту, кешування та обробки стану.

Сторінка з власними заходами користувача використовує подібний підхід, але з використанням іншого хука запиту, який отримує лише заходи поточного користувача. Це демонструє гнучкість архітектури, де різні види запитів можуть легко підключатися до одних і тих же компонентів інтерфейсу.

Intercepting Routes та модальні вікна:

Особливістю архітектури є використання Intercepting Routes Next.js для реалізації модальних вікон, які зберігають контекст навігації. Цей підхід дозволяє відкривати детальні сторінки у модальних вікнах, зберігаючи при цьому можливість прямого доступу до цих сторінок через URL [2].

Система авторизації реалізована з використанням цього підходу. Користувач може увійти або зареєструватися через модальні вікна, які відкриваються поверх поточної сторінки. При цьому URL змінюється відповідно до відкритого модального вікна, що дозволяє ділитися посиланнями та використовувати навігацію браузера.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Аналогічний підхід використовується для перегляду деталей заходів з інших сторінок. Натискання на кнопку перегляду відкриває модальне вікно з повною інформацією про захід, змінюючи URL, але зберігаючи контекст попередньої сторінки.

Система фільтрації та пошуку:

Для забезпечення можливості фільтрації, сортування та пошуку заходів реалізовано механізм обробки параметрів запитів. Цей механізм дозволяє формувати URL-параметри для запитів до API на основі стану компонентів.

Компонент фільтрації надає користувачам можливість фільтрувати заходи за різними критеріями, включаючи пошук за назвою активності, загальний пошук по опису та інших текстових полях, тип активності та територіальне розташування. Фільтри реалізовані з використанням спеціального сховища стану Zustand, яке синхронізується з URL для збереження стану фільтрів між переходами [6].

Особливою функціональністю є територіальний фільтр, який використовує компонент автодоповнення для пошуку територій. Цей компонент використовує дебаунсинг для оптимізації запитів до API та надає користувачам зручний спосіб вибору географічних регіонів.

На стороні клієнта ці параметри синхронізуються з URL сторінки, що дозволяє зберігати стан фільтрів та забезпечує можливість ділитися посиланнями з відфільтрованими результатами.

Обробка форм та валідація даних:

Важливою частиною функціональності платформи є створення та редагування заходів. Для цього використовуються форми, які дозволяють користувачам вводити необхідну інформацію з подальшою валідацією на клієнтській стороні.

Для обробки форм використовується підхід з контрольованими компонентами та схемами валідації Zod [4]. Форма створення заходу включає всі необхідні поля, включаючи назву, опис, тип активності, майданчик проведення,

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

медіа файли та веб-посилання. Залежно від типу активності форма динамічно відображає додаткові поля (додаток А).

Для одноразових заходів користувач повинен вказати дату проведення, час початку та закінчення, а також обрати розкладку місць з доступних у вибраному майданчику. Для регулярних заходів потрібно вказати тривалість заняття та обрати розклад роботи з попередньо створених користувачем розкладів.

Валідація даних відбувається на декількох рівнях. Клієнтська валідація забезпечує миттєвий зворотний зв'язок користувачу та попереджає відправку некоректних даних. Серверна валідація надає додатковий рівень захисту та забезпечує цілісність даних в системі.

Система опцій для регулярних заходів:

Для регулярних заходів реалізована система опцій, яка дозволяє організаторам створювати різні варіанти участі. Після створення регулярного заходу користувач може одразу додати опції через інтегровану форму, яка дозволяє створювати декілька опцій одночасно.

Кожна опція має назву та опис, що дозволяє організаторам визначити різні рівні участі, типи занять або інші варіанти. Система автоматично публікує захід після створення опцій, що забезпечує його видимість для інших користувачів.

Опції можуть бути відредаговані або видалені після створення. При видаленні останньої опції опублікованого заходу система автоматично переводить захід у статус чернетки, оскільки регулярні заходи без опцій не можуть бути доступні для бронювання.

Відображення деталей заходів:

Сторінка деталей заходу надає повну інформацію про конкретний захід та дозволяє користувачам взаємодіяти з ним. Ліва частина сторінки містить медіа галерею, яка відображає всі завантажені зображення та відео. При натисканні на медіа файл відкривається модальне вікно з детальним переглядом.

Права частина містить всю текстову інформацію про захід, включаючи опис, деталі проведення, інформацію про майданчик та розклад роботи. Для

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

регулярних заходів відображається список доступних опцій з можливістю їх редагування для створювача заходу.

Користувачі, які не є створювачами заходу, можуть залишати реакції у вигляді лайків або дізлайків, а також бронювати участь у заході. Система бронювання відрізняється залежно від типу заходу.

Система бронювання:

Для одноразових заходів користувачі обирають конкретне місце з доступних у розкладці майданчика. Система відображає лише вільні місця та зони з доступними місцями, автоматично оновлюючи інформацію про заброньовані місця.

Для регулярних заходів процес бронювання включає вибір опції, дати та часу проведення. Система відображає лише ті дати, які відповідають розкладу роботи заходу, та доступні часові слоти для кожної дати. Користувачі також можуть додати нотатки до свого бронювання.

Форма бронювання включає валідацію, яка попереджає бронювання на минулі дати та забезпечує вибір всіх необхідних параметрів. Після успішного бронювання користувач отримує підтвердження та може переглянути своє бронювання на спеціальній сторінці.

Керування власним контентом:

Користувачі мають доступ до спеціальних сторінок для керування власним контентом. Сторінка власних заходів відображає всі створені користувачем заходи з можливістю їх редагування, видалення або зміни статусу публікації.

Аналогічні сторінки існують для розкладів, майданчиків, бронювань та реакцій. Це дозволяє користувачам ефективно керувати всім створеним контентом з одного місця.

Система також надає статистику використання, яка показує дані про бронювання за різні періоди з можливістю фільтрації за інтервалами та датами. Це допомагає організаторам аналізувати популярність своїх заходів та планувати майбутні активності.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

Адаптивний дизайн та доступність:

Весь інтерфейс адаптований для роботи на різних пристроях, включаючи мобільні телефони та планшети. Компоненти автоматично адаптуються до розміру екрану, забезпечуючи зручність використання на будь-якому пристрої.

Система підтримує темну та світлу теми оформлення, які користувач може перемикати згідно з власними уподобаннями. Тема зберігається локально та застосовується автоматично при повторних відвідуваннях.

Така консистентність у підході забезпечує передбачуваність поведінки додатку для користувачів та спрощує підтримку коду для розробників. Кожна нова функціональність може бути реалізована за встановленими шаблонами, що прискорює розробку та зменшує ймовірність помилок.

Підсумовуючи, можна сказати, що реалізація функціоналу системи платформи з розміщення подій та заходів базується на сучасних підходах до розробки веб-додатків, включаючи використання React для побудови інтерфейсу, TanStack Query для керування станом запитів, Next.js для маршрутизації та серверного рендерингу, Zod для валідації даних, а також TypeScript для статичної типізації. Такий підхід забезпечує високу якість коду, масштабованість, продуктивність та простоту підтримки додатку в довгостроковій перспективі.

Висновки до розділу 2

У розділі 2 було розглянуто комплексне проектування та реалізацію фронтенд-частини платформи розміщення подій та заходів, що забезпечує повний цикл роботи з заходами від створення до бронювання.

На основі детального аналізу функціональних та нефункціональних вимог було визначено основні варіанти використання системи для двох типів користувачів: організаторів заходів з розширеними можливостями управління та звичайних користувачів з функціями перегляду та бронювання. Створена діаграма варіантів використання чітко структурувала функціональні можливості системи та визначила межі відповідальності кожної ролі користувача.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

Розроблена компонентна архітектура системи базується на принципах модульності та розділення відповідальності. Основні компоненти включають кореневий компонент для ініціалізації додатку, систему авторизації з JWT-токенами, навігацію з підтримкою intercepting routes, компоненти управління активностями, розкладами, локаціями з планувальними схемами, системою бронювань, реакціями та статистикою. Така архітектура забезпечує масштабованість та простоту підтримки додатку.

Для забезпечення надійної взаємодії з API було створено комплексну систему TypeScript інтерфейсів та типів, що включає базові сутності користувачів, заходів, локацій, розкладів та бронювань. Особливе значення має система поліморфних типів для візуальних об'єктів та дискримінованих union типів для різних видів активностей та бронювань. Використання Zod для валідації забезпечує перевірку коректності даних на клієнтській стороні, що значно підвищує надійність системи.

Розроблені алгоритми роботи системи охоплюють всі ключові процеси платформи. Алгоритм створення активностей підтримує два типи заходів з динамічною адаптацією форм. Система пошуку та фільтрації реалізує множинні критерії з оптимізованими запитами та синхронізацією з URL. Управління локаціями включає роботу з географічними координатами та інтерактивними планувальними схемами. Алгоритм бронювання диференційовано обробляє одноразові та регулярні заходи з відповідною валідацією та оновленням стану доступності.

Реалізація функціоналу системи базується на сучасних підходах веб-розробки з використанням React, Next.js App Router, TanStack Query для керування серверним станом та Zustand для локального стану. Архітектура взаємодії з API включає типізовані сервіси, автоматичну обробку авторизації та помилок, ефективне кешування відповідей. Особливістю є використання intercepting routes для модальних вікон, що забезпечує плавний користувацький досвід при збереженні контексту навігації.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

Система фільтрації та пошуку реалізована з підтримкою територіального пошуку, дебаунсингу запитів та синхронізації стану з URL-параметрами. Форми створення та редагування включають динамічну валідацію, роботу з медіафайлами та автоматичну обробку різних типів контенту. Система опцій для регулярних заходів забезпечує гнучке налаштування варіантів участі з автоматичною публікацією після створення опцій.

Реалізована система бронювань підтримує два принципово різні підходи: точне бронювання місць для одноразових заходів та бронювання часових слотів для регулярних активностей. Інтегрована система реакцій та статистики дозволяє аналізувати популярність заходів та ефективність організаторської діяльності.

Важливими аспектами реалізації є адаптивний дизайн з підтримкою мобільних пристроїв, система тем оформлення, комплексна обробка помилок та оптимізація продуктивності через кешування та лініве завантаження компонентів.

Таким чином, створена фронтенд-частина платформи представляє собою комплексне рішення, що забезпечує ефективну роботу з заходами різних типів, має гнучку архітектуру для подальшого розширення функціональності та базується на сучасних технологіях, що гарантують високу продуктивність, надійність та зручність використання для всіх категорій користувачів.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ПОШУКУ ТА УПРАВЛІННЯ АКТИВНОСТЯМИ

3.1 Порядок встановлення та налаштування фронтенд-частини системи

Встановлення та налаштування фронтенд-частини платформи пошуку та управління активностями передбачає підготовку робочого середовища, конфігурацію залежностей та налаштування параметрів для взаємодії з API сервером. Клієнтська частина розроблена з використанням сучасних технологій Next.js 15, React 19 та TypeScript, що вимагає специфічного підходу до розгортання та конфігурації.

Для візуалізації архітектури фронтенд-частини було створено діаграму ресурсів системи (рис. 3.1).

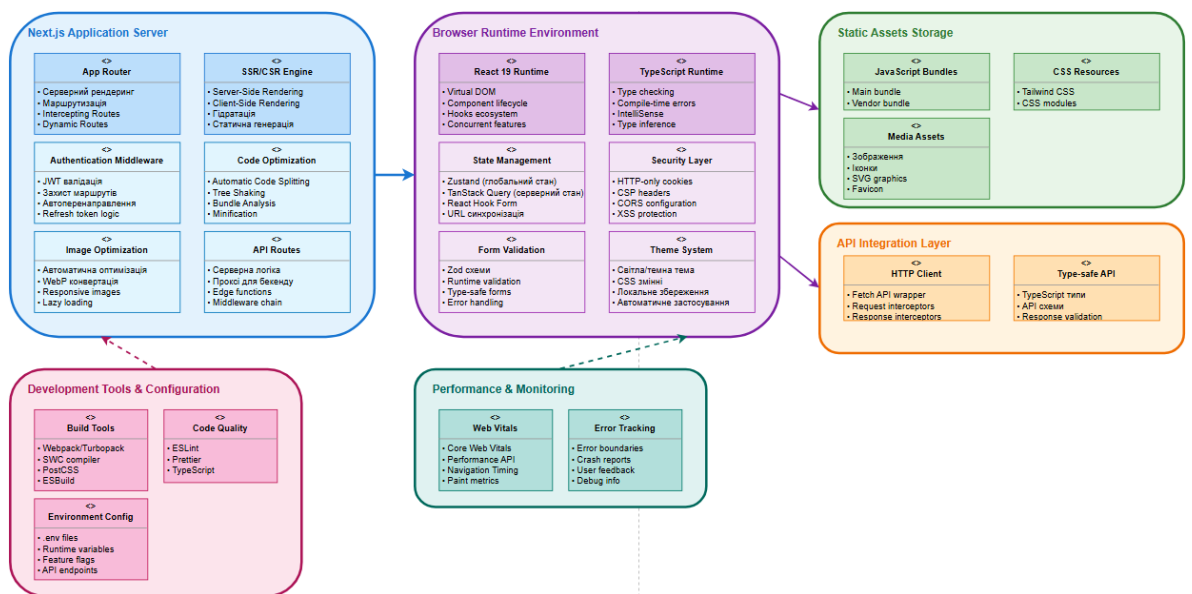


Рис. 3.1 – Діаграма ресурсів фронтенд-частини платформи

Архітектура фронтенд-частини включає кілька ключових компонентів, які взаємодіють між собою для забезпечення повноцінної роботи користувацького інтерфейсу.

Next.js Application Server представляє основний додаток, побудований на фреймворку Next.js 15 з підтримкою App Router. Цей компонент відповідає за серверний рендеринг початкових сторінок, маршрутизацію, middleware для

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

автентифікації та обробку `intercepting routes` для модальних вікон. Сервер додатку включає систему SSR для оптимізації SEO та швидкості завантаження, CSR для інтерактивної взаємодії після початкового завантаження, статичну генерацію для сторінок, що рідко змінюються та `middleware` для захисту приватних маршрутів.

Browser Runtime Environment являє собою середовище виконання в веб-браузері користувача, де працює React додаток після гідратації. Цей компонент управляє локальним станом через Zustand для фільтрів та специфічних компонентів, кешуванням серверних даних через TanStack Query, управлінням форм через React Hook Form з Zod валідацією та збереженням JWT токенів в HTTP-only cookies для безпеки.

API Integration Layer забезпечує взаємодію з бекенд сервером через HTTP запити з автоматичною обробкою автентифікації, валідацією даних та обробкою помилок. Цей шар включає типізовані API клієнти для кожної предметної області, автоматичне оновлення JWT токенів при експірації, кешування відповідей для оптимізації продуктивності та `retry` логіку для обробки мережевих помилок.

Процес встановлення фронтенд-частини включає кілька послідовних етапів підготовки та конфігурації.

Підготовка середовища розробки починається з встановлення Node.js версії 18 або новішої, що забезпечує сумісність з усіма використаними залежностями. `npm` package manager використовується для управління залежностями проекту та виконання скриптів збірки. Рекомендується використання `nvm` (Node Version Manager) для управління версіями Node.js в різних проектах. Також необхідно налаштувати сучасний код-редактор з підтримкою TypeScript, ESLint та Prettier для забезпечення якості коду.

Ініціалізація проекту включає клонування репозиторію з кодом фронтенд-частини, встановлення всіх залежностей через команду `npm install`, що завантажує необхідні пакети відповідно до `package.json` файлу. Структура

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

проекту організована відповідно до рекомендацій Next.js App Router з чітким розділенням компонентів, сторінок, API клієнтів та утиліт.

Конфігурація змінних середовища є критично важливою для правильної роботи додатку. Файл `.env.local` містить налаштування для взаємодії з API сервером, включаючи `NEXT_PUBLIC_API_URL` для вказання базової адреси бекенд сервера, `GOOGLE_MAPS_API_KEY` для інтеграції з Google Maps API, `NEXTAUTH_SECRET` для конфігурації автентифікації. Змінні з префіксом `NEXT_PUBLIC` доступні в браузері, тому не повинні містити чутливої інформації.

Налаштування TypeScript включає конфігурацію `tsconfig.json` файлу з суворими правилами типізації, що забезпечує високу якість коду та раннє виявлення помилок. Конфігурація включає `strict mode` для максимальної перевірки типів, `path mapping` для зручного імпорту модулів, `incremental compilation` для прискорення збірки та інтеграцію з Next.js для оптимальної роботи фреймворку.

Конфігурація TailwindCSS передбачає налаштування файлу `tailwind.config.js` з кастомізацією кольорової палітри для підтримки світлої та темної тем, додаванням власних утилітарних класів, конфігурацією `responsive breakpoints` для адаптивного дизайну та інтеграцією з CSS-in-JS рішеннями для динамічних стилів.

Система автентифікації налаштовується для безпечної роботи з JWT токенами через `HTTP-only cookies`. Конфігурація включає налаштування `middleware` для захисту приватних маршрутів, автоматичне перенаправлення неавторизованих користувачів, `refresh token logic` для продовження сесій та логіку автоматичного логауту при закінченні терміну дії токенів.

TanStack Query конфігурується для ефективного управління серверним станом з налаштуванням `stale time` для контролю актуальності кешованих даних, `retry policy` для обробки мережових помилок та `background refetching` для автоматичного оновлення даних.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

3.2 Структура інтерфейсу платформи пошуку та управління активностями

Інтерфейс платформи пошуку та управління активностями побудований на принципах сучасного веб-дизайну з акцентом на зручність використання, інтуїтивність та адаптивність. Система підтримує різні типи користувачів з відповідним функціоналом та правами доступу.

Загальна структура інтерфейсу базується на компонентному підході з використанням ShadCN UI бібліотеки, що забезпечує консистентність дизайну та доступність. Платформа підтримує світлу та темну теми оформлення з можливістю автоматичного перемикання залежно від системних налаштувань користувача.

Головна сторінка платформи (рис. 3.2) представляє основну точку входу для всіх користувачів та містить навігаційне меню, список популярних активностей та можливості швидкого пошуку.

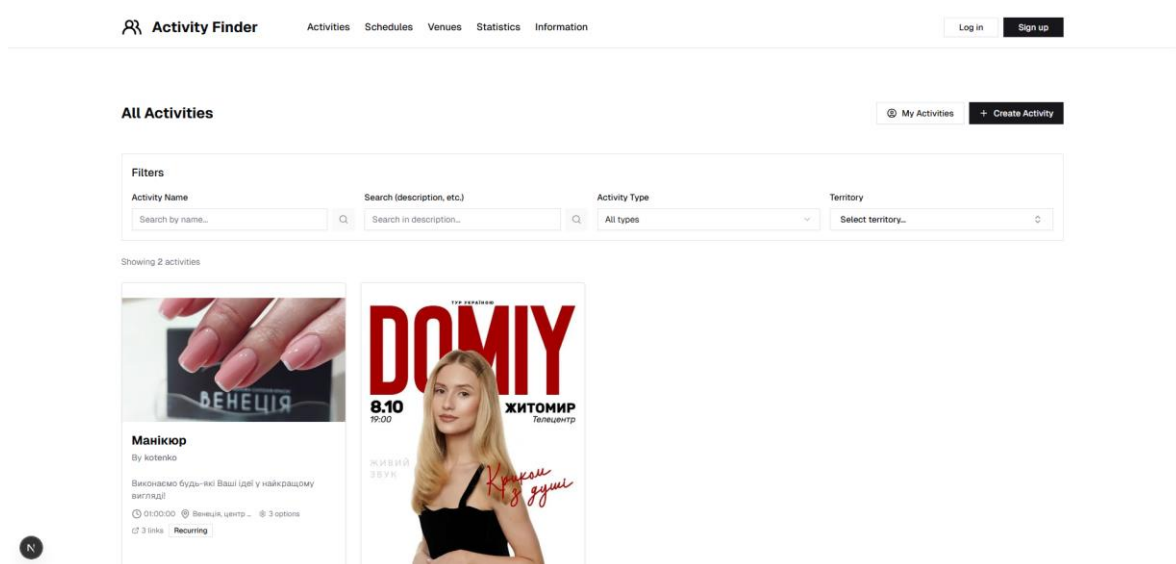


Рис. 3.2 – Головна сторінка платформи активностей

Навігаційна панель розташована у верхній частині сторінки та включає логотип платформи, основне меню з посиланнями на ключові розділи, кнопки авторизації для неавторизованих користувачів або меню користувача для авторизованих. Меню користувача надає доступ до персональних сторінок, налаштувань профілю та можливості виходу з системи.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

Система авторизації реалізована через модальні вікна з підтримкою *intercepting routes* для плавної навігації. Користувачі можуть увійти в систему або зареєструватися через спеціальні форми (рис. 3.3).

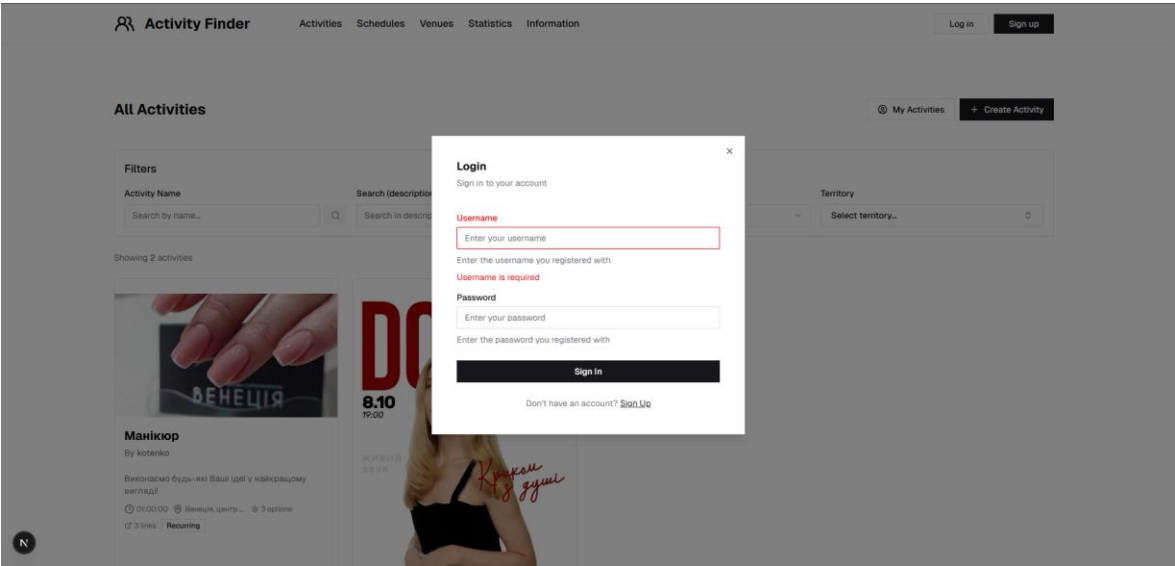


Рис. 3.3 – Модальне вікно авторизації користувача

Форма входу містить поля для введення імені користувача та пароля з валідацією на клієнтській стороні. Система автоматично обробляє помилки автентифікації та надає зрозумілі повідомлення користувачу.

Основна сторінка активностей (рис. 3.4) представляє центральний функціональний блок платформи з можливостями перегляду, пошуку та фільтрації доступних заходів.

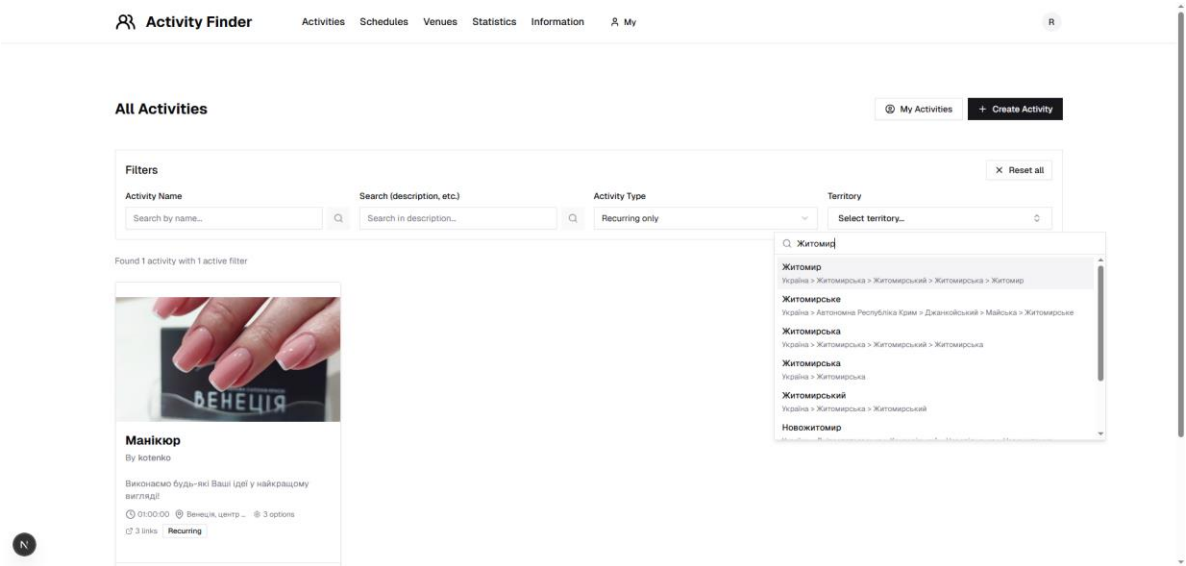


Рис. 3.4 – Сторінка списку активностей з фільтрами

Система фільтрації включає пошук за назвою активності з підтримкою автодоповнення, фільтрацію за типом активності з можливістю множинного вибору та територіальну фільтрацію через інтеграцію з API територій. Фільтри синхронізуються з URL-параметрами, що дозволяє зберігати та ділитися посиланнями на відфільтровані результати.

Картки активностей відображають ключову інформацію про кожен захід, включаючи назву, короткий опис, тип активності, локацію проведення та прев'ю зображення.

Детальна сторінка активності (рис. 3.5) надає повну інформацію про конкретний захід з можливостями взаємодії залежно від ролі користувача.

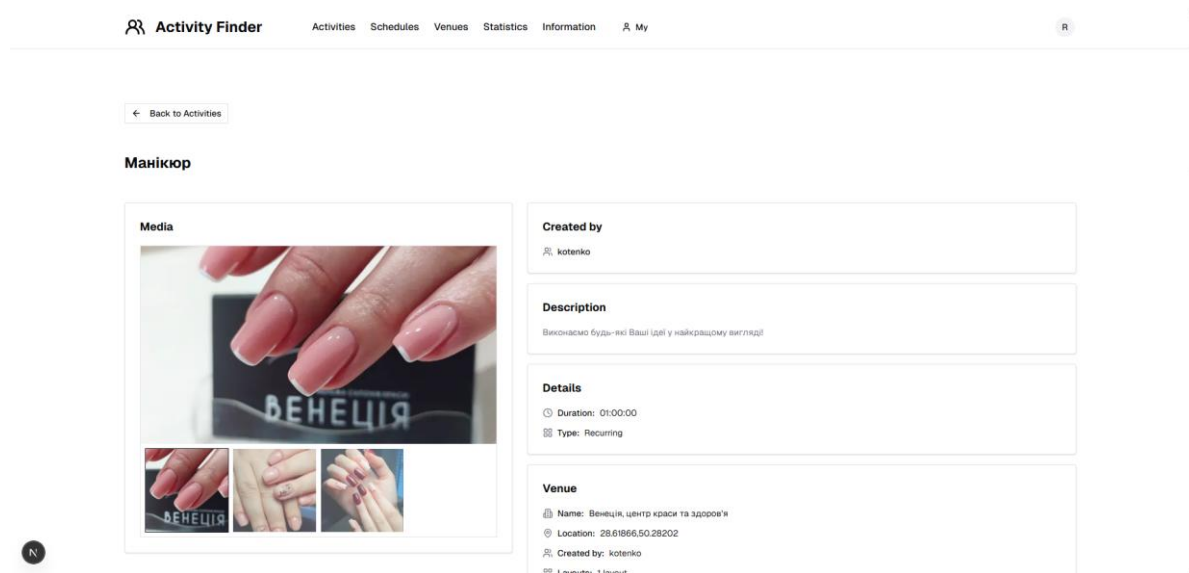


Рис. 3.5 – Детальна сторінка активності з медіагалереєю

Ліва частина сторінки містить медіагалерею з підтримкою різних типів файлів, включаючи зображення та відео. При натисканні на медіафайл відкривається модальне вікно з повноекранним переглядом та можливістю навігації між файлами. Права частина включає детальний опис активності, інформацію про організатора, час та місце проведення.

Для регулярних активностей відображається список доступних опцій з описом особливостей кожної. Користувачі можуть залишати реакції у вигляді лайків або дізлайків, а також здійснювати бронювання через спеціальну форму.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

Система бронювання адаптується залежно від типу активності (рис. 3.6). Для одноразових заходів користувачі обирають конкретне місце з інтерактивної планувальної схеми, а для регулярних активностей - дату, час та опцію участі.

Рис. 3.6 – Форма бронювання для регулярної активності

Планувальні схеми для одноразових активностей відображають інтерактивну карту місць з візуальним позначенням доступності.

Сторінки управління контентом доступні для організаторів заходів та включають можливості створення і редагування активностей, розкладів та локацій. Форма створення активності (рис. 3.7) адаптується залежно від обраного типу заходу.

Рис. 3.7 – Форма створення нової активності

Форма включає базові поля для всіх типів активностей та додаткові секції, що з'являються залежно від обраного типу. Валідація даних відбувається в реальному часі з відображенням помилок безпосередньо під відповідними полями.

Управління локаціями включає створення детальних описів місць проведення заходів з географічними координатами та медіафайлами. Інтерактивне редагування планувальних схем (рис. 3.8) дозволяє розміщувати різні типи візуальних об'єктів для бронювання.

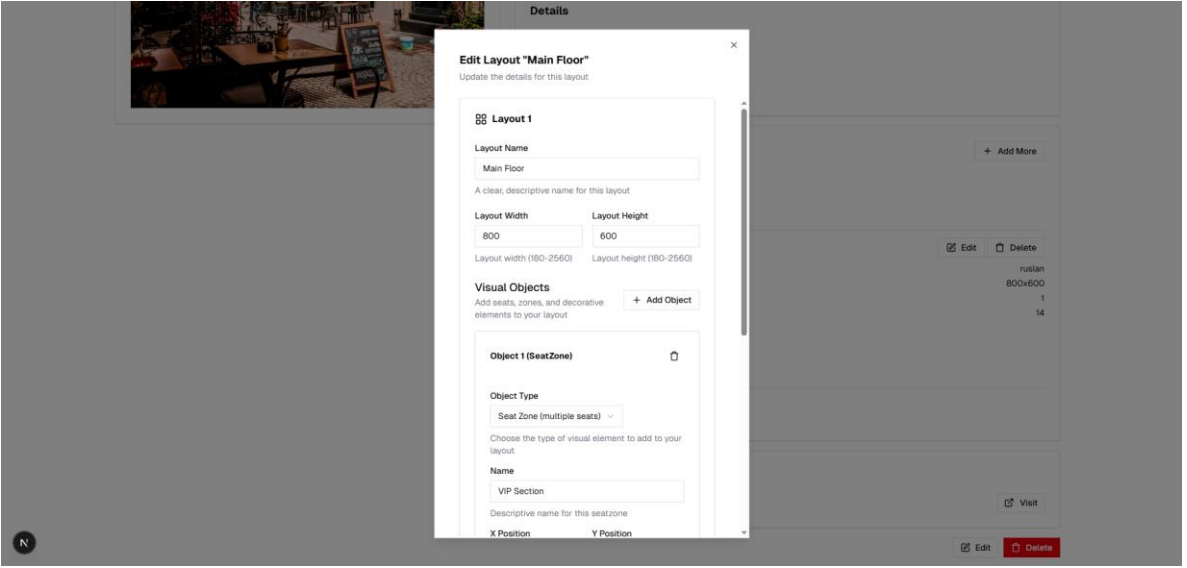


Рис. 3.8 – Редагування планувальних схем локації

Редагування підтримує три типи об'єктів: базові візуальні елементи для оформлення, окремі місця для точного бронювання та зони з множинними місцями. Кожен об'єкт має налаштування позиції, розміру та кута повороту з можливістю точного позиціонування через числові поля.

Управління розкладами (рис. 3.9) надає інструменти для створення складних робочих графіків регулярних активностей.

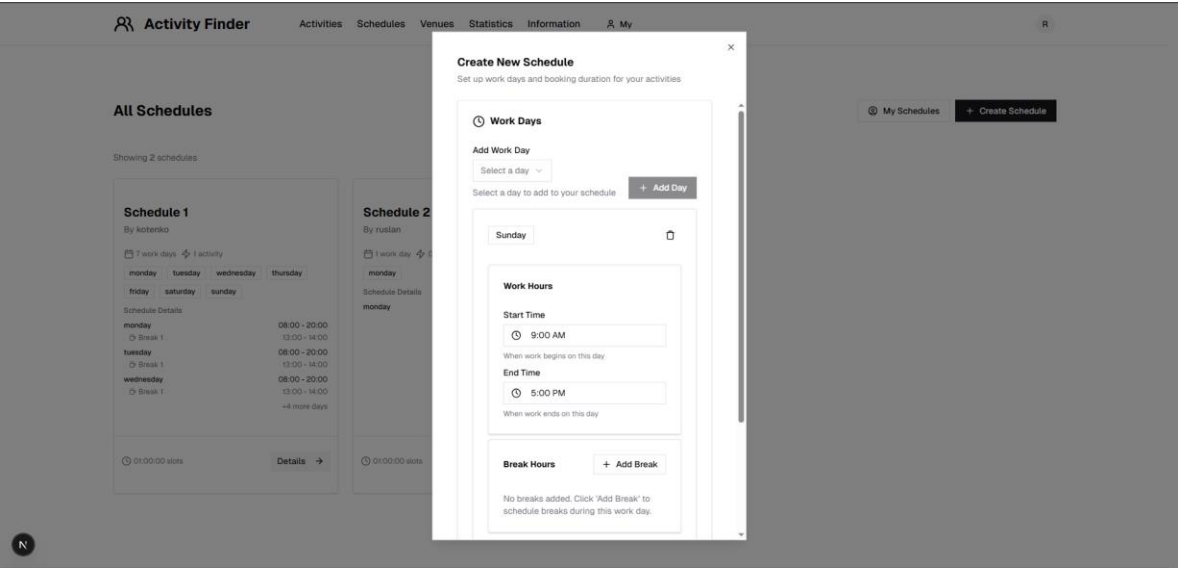


Рис. 3.9 – Форма створення розкладу з налаштуванням робочих днів

Персональні сторінки користувачів включають перегляд власних активностей, бронювань та реакцій. Сторінка власних бронювань (рис. 3.10) відображає всі здійснені резервації з можливістю фільтрації за статусом та датою.

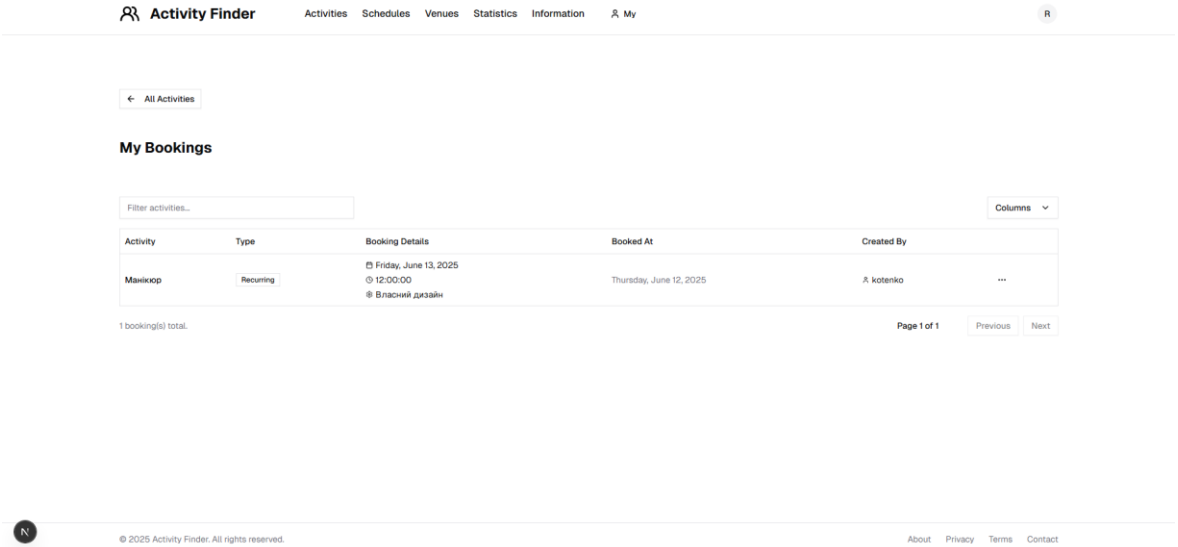


Рис. 3.10 – Сторінка власних бронювань користувача

Статистична сторінка (рис. 3.11) надає детальну аналітику бронювань з можливістю фільтрації за різними періодами та параметрами.

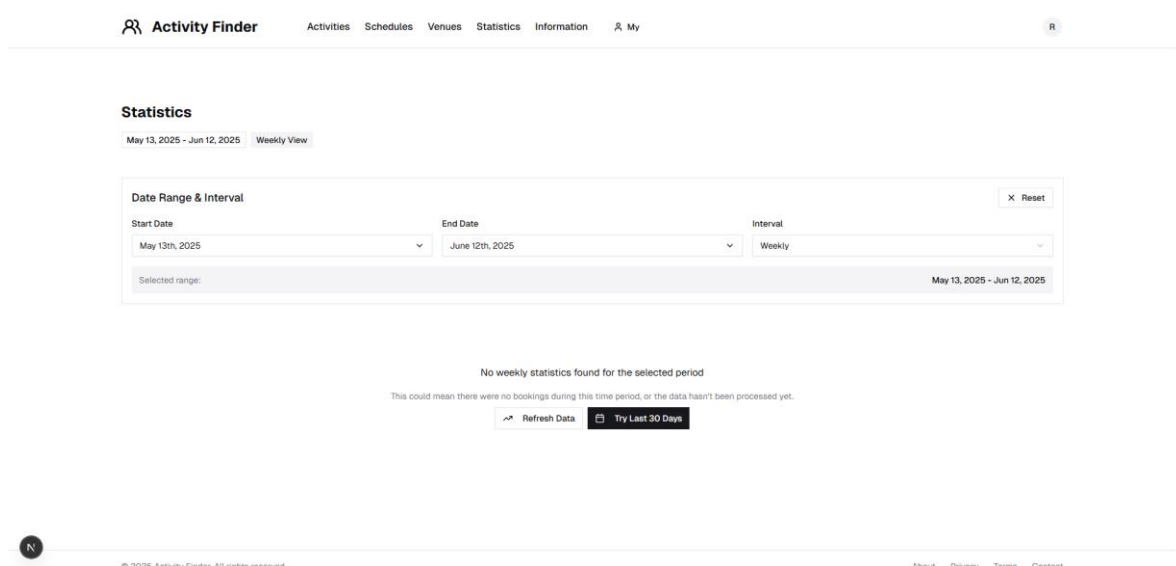


Рис. 3.11 – Сторінка власних бронювань користувача

Інтерфейс включає графіки для візуалізації трендів бронювань, розподілу за типами активностей та аналізу повторних клієнтів. Фільтри дозволяють обирати часові інтервали та деталізацію аналітики.

Адаптивний дизайн забезпечує коректне відображення на різних пристроях. Адаптована версія (рис. 3.12) з адаптованою навігацією та спрощеними формами.

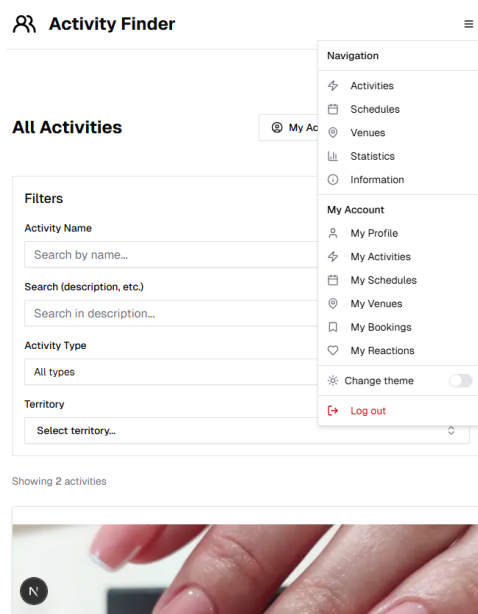


Рис. 3.12 – Адаптована версія головної сторінки

Навігаційне меню на мобільних пристроях згортається в hamburger menu, форми адаптуються до вертикального розташування, а картки активностей

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

оптимізуються для меншого розміру екрану зі збереженням ключової інформації.

Система тем реалізована через next-themes бібліотеку з підтримкою автоматичного визначення системних налаштувань користувача. Перемикач теми розташований в меню користувача та дозволяє вибирати між світлою, темною темами або автоматичним режимом (рис. 3.13). Зміна теми відбувається миттєво без перезавантаження сторінки.

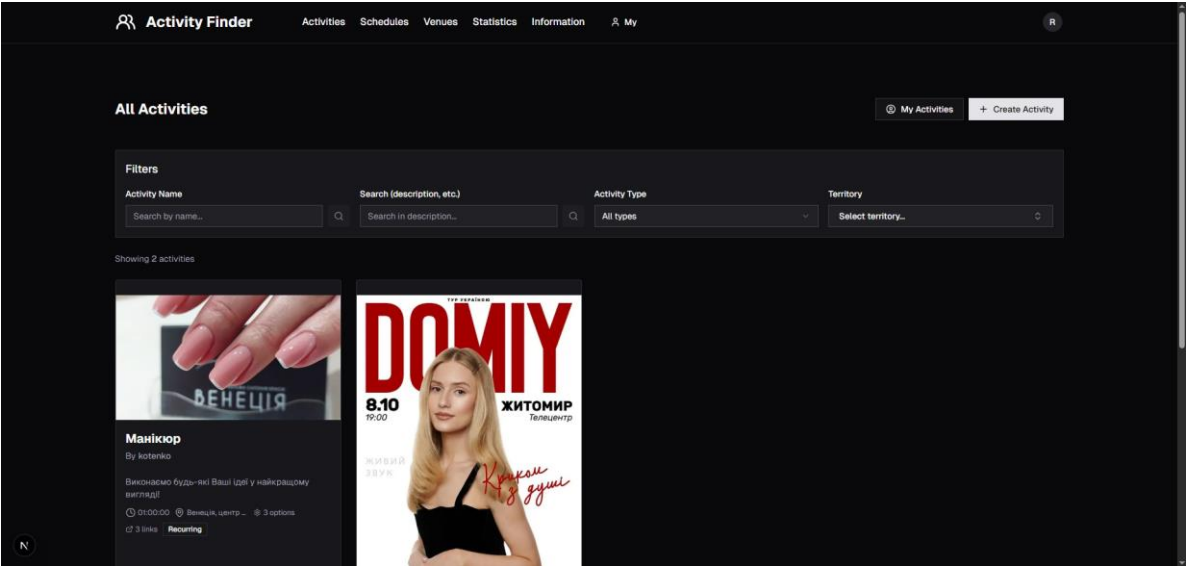


Рис. 3.13 – Темна тема

Система повідомлень та feedback надає користувачам інформацію про результати дій через toast notifications та inline повідомлення. Loading states відображаються під час виконання асинхронних операцій, а error boundaries обробляють критичні помилки з можливістю відновлення роботи.

Доступність інтерфейсу забезпечується через семантичну HTML розмітку, підтримку keyboard navigation, aria-labels для screen readers та high contrast режими. Всі інтерактивні елементи мають відповідні focus states та tooltip підказки для покращення користувацького досвіду.

3.3 Тестування роботи програмного комплексу

Тестування фронтенд-частини платформи пошуку та управління активностями проводилося комплексно протягом всього процесу розробки з акцентом на функціональність, сумісність з різними браузерами, продуктивність

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

та користувацький досвід. Основний підхід до тестування базувався на безпосередній перевірці функціоналу під час розробки кожного компонента з використанням вбудованих механізмів валідації та обробки помилок.

Тестування та оптимізація проводились переважно одразу при створенні певного функціоналу, що дозволило виявляти та усувати проблеми на ранніх стадіях розробки. Такий підхід забезпечив високу якість коду та стабільність роботи додатку без використання спеціалізованих тестувальних бібліотек.

Основними напрямками тестування були функціональне тестування всіх компонентів інтерфейсу, валідація форм та обробка користувацьких введень, тестування сумісності з різними веб-браузерами, перевірка адаптивності дизайну на різних пристроях, аналіз продуктивності та швидкості завантаження, тестування системи автентифікації та авторизації.

Система валідації даних реалізована через Zod схеми, які забезпечують комплексну перевірку користувацьких введень на клієнтській стороні. Валідація охоплює всі типи форм, включаючи реєстрацію та авторизацію користувачів, створення та редагування активностей, управління локаціями та розкладами, здійснення бронювань та налаштування профілю.

Система обробки помилок включає глобальні компоненти для різних типів проблемних ситуацій. Компонент `error.tsx` обробляє критичні помилки додатку з можливістю відновлення роботи. Компонент `not-found.tsx` обробляє ситуації з неіснуючими сторінками, надаючи зрозуміле повідомлення та навігацію до головної сторінки. Компонент `loading.tsx` забезпечує візуальний індикатор завантаження для покращення користувацького досвіду під час асинхронних операцій.

Платформа протестована на сумісність з основними сучасними веб-браузерами для забезпечення коректної роботи для максимальної кількості користувачів. Всі рисунки інтерфейсу в розділі 3.2 були створені в браузері Google Chrome, який використовувався як основний для розробки та тестування.

Додатково проведено тестування в інших популярних браузерах для перевірки кросбраузерної сумісності. Тестування в браузері Brave (рис. 3.14)

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

показало повну сумісність з усіма функціями платформи завдяки спільному движку Chromium.

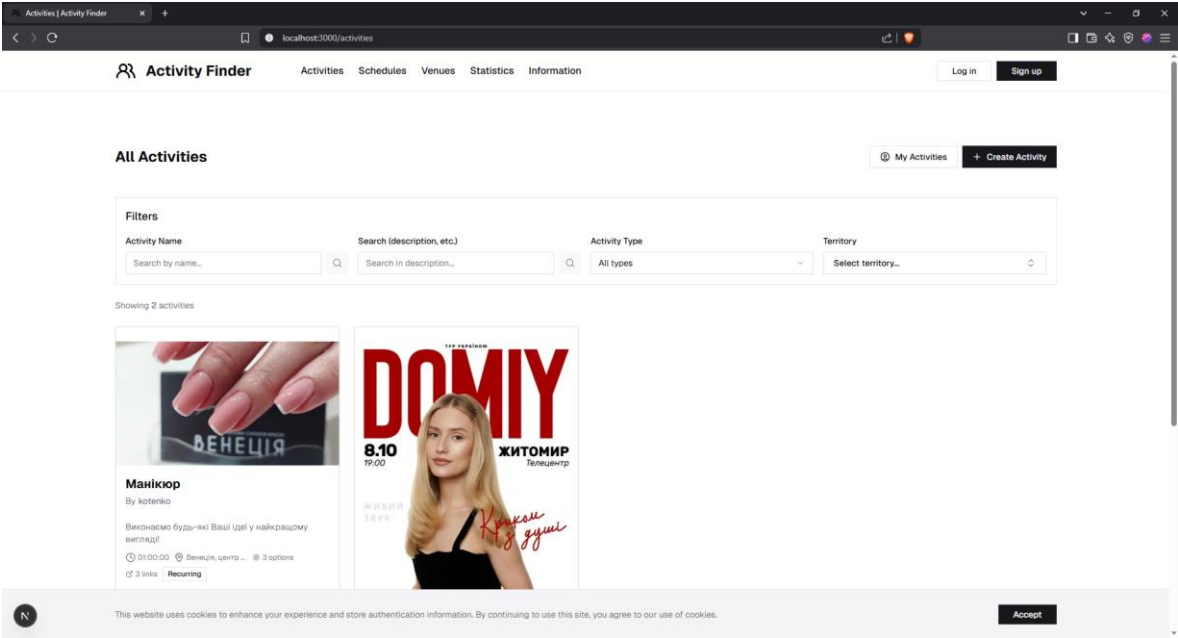


Рис. 3.14 – Відображення платформи в браузері Brave

Тестування в браузері Mozilla Firefox (рис. 3.15) підтвердило коректну роботу всіх компонентів інтерфейсу з незначними візуальними відмінностями в рендерингу окремих CSS властивостей, що не впливають на функціональність.

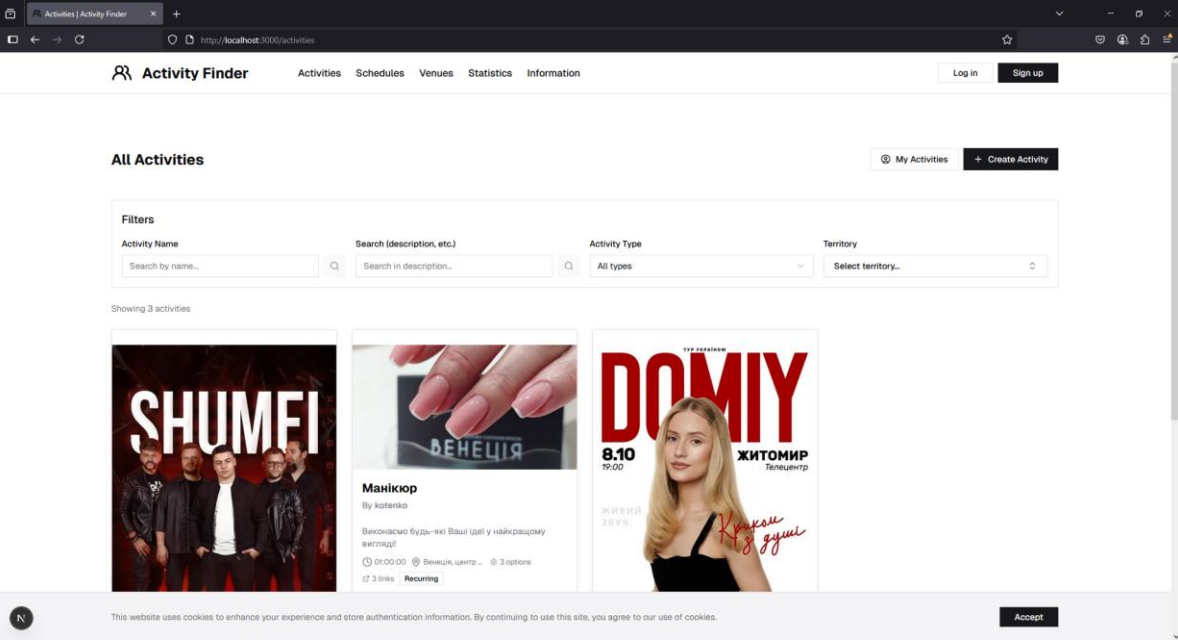


Рис. 3.15 – Відображення платформи в браузері Firefox

Важливою частиною тестування було дослідження мережевої активності додатку для оптимізації продуктивності та виявлення потенційних проблем з

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

завантаженням ресурсів. Аналіз проводився через Developer Tools браузера з використанням вкладки Network (рис. 3.16).

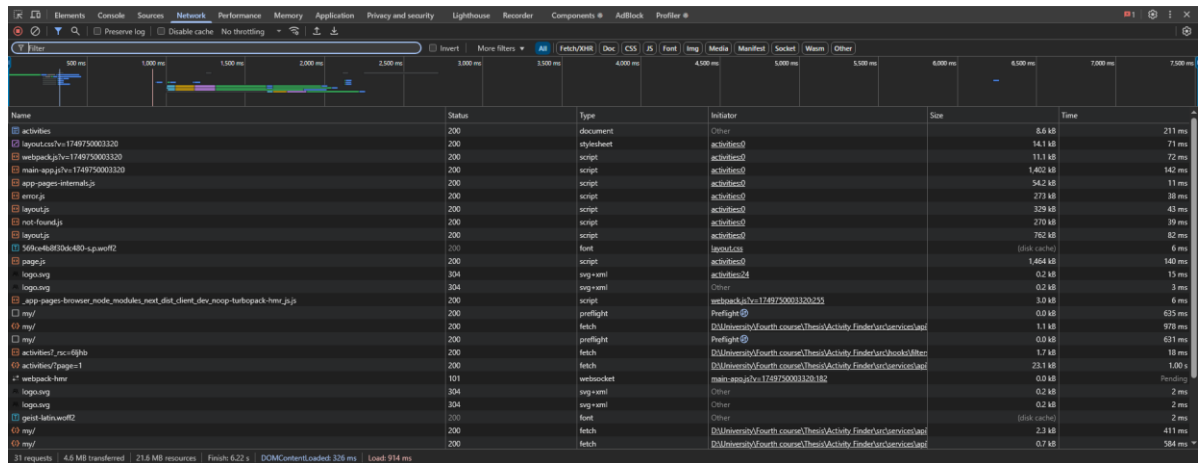


Рис. 3.16 – Аналіз мережевих запитів у вкладці Network

Моніторинг мережевої активності показав ефективну роботу TanStack Query з автоматичним кешуванням повторних запитів, оптимізовані розміри JavaScript бандлів завдяки automatic code splitting Next.js, швидке завантаження статичних ресурсів через оптимізацію зображень та компресію файлів, мінімальну кількість API запитів завдяки ефективному управлінню станом.

Комплексне функціональне тестування охопило всі основні сценарії використання платформи. Тестування системи автентифікації включало перевірку коректності входу та реєстрації користувачів, автоматичного оновлення JWT токенів, захисту приватних маршрутів через middleware та безпечного завершення сеансів.

Тестування управління активностями охопило створення та редагування одноразових і регулярних заходів, завантаження та відображення медіафайлів, систему публікації та приховування активностей, інтеграцію з локаціями та розкладами.

Система бронювання протестована для обох типів активностей з перевіркою коректності вибору місць для одноразових заходів, бронювання часових слотів для регулярних активностей, валідації доступності та оновлення стану після бронювання.

Адаптивний дизайн протестований на різних розмірах екранів від мобільних телефонів до широких моніторів. Використання TailwindCSS з вбудованими breakpoints забезпечило коректне відображення на всіх пристроях з автоматичним адаптуванням компонентів.

Перевірка доступності включала тестування навігації за допомогою клавіатури, сумісності з screen readers, контрастності кольорів для читабельності тексту та наявності alt-текстів для зображень. Використання семантичної HTML розмітки та ARIA атрибутів забезпечило підтримку допоміжних технологій.

Перевірка стабільності роботи під навантаженням включала тестування з великою кількістю одночасних запитів, завантаження об'ємних медіафайлів, роботу з великими списками активностей та швидке перемикання між сторінками. TanStack Query ефективно обробляв кешування та дедуплікацію запитів, запобігаючи зайвому навантаженню на сервер.

Тестування безпеки фронтенду включало перевірку захисту від XSS атак через правильне екранування користувацьких введень, валідацію всіх форм на клієнтській стороні, безпечне зберігання JWT токенів в HTTP-only cookies та захист чутливої інформації від витоку в browser console.

Результати тестування підтверджують високу якість та надійність розробленої фронтенд-частини платформи. Система демонструє стабільну роботу в різних браузерах, ефективну обробку помилок, оптимальну продуктивність та зручність використання для всіх категорій користувачів. Використання сучасних технологій та найкращих практик забезпечило створення масштабованого та підтримуваного рішення, що відповідає вимогам сучасних веб-додатків.

Висновки до розділу 3

У третьому розділі було детально розглянуто процес встановлення, структуру інтерфейсу та тестування фронтенд-частини платформи пошуку та управління активностями, що підтвердило ефективність обраних технологічних рішень та архітектурних підходів.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

Процес встановлення та налаштування фронтенд-частини системи продемонстрував гнучкість архітектури на базі Next.js 15 з можливістю розгортання в різних середовищах. Розроблена діаграма ресурсів системи чітко визначила взаємодію між компонентами Next.js Application Server, Browser Runtime Environment та API Integration Layer. Детально описаний процес конфігурації включає налаштування TypeScript для строгої типізації, TailwindCSS для адаптивного дизайну, TanStack Query для управління серверним станом та системи автентифікації з JWT токенами. Особливу увагу приділено безпеці клієнтської частини через HTTP-only cookies та валідацію даних.

Структура інтерфейсу платформи побудована на принципах сучасного UX/UI дизайну з використанням компонентного підходу ShadCN UI бібліотеки. Реалізована підтримка світлої та темної тем з автоматичним перемиканням відповідно до системних налаштувань користувача забезпечує комфортне використання платформи. Система навігації з intercepting routes дозволяє плавно переходити між модальними вікнами та повноцінними сторінками зі збереженням контексту. Адаптивний дизайн гарантує коректне відображення на всіх типах пристроїв від мобільних телефонів до широких моніторів з оптимізацією для сенсорного управління.

Функціональність інтерфейсу охоплює повний спектр можливостей платформи, включаючи систему авторизації через модальні вікна, розширені можливості пошуку та фільтрації активностей з територіальною підтримкою, детальні сторінки заходів з медіагалереями та можливостями взаємодії. Система управління контентом надає організаторам інструменти для створення активностей різних типів, редагування планувальних схем з інтерактивним розміщенням візуальних об'єктів та управління складними розкладами з гнучкими налаштуваннями робочих днів. Персональні сторінки забезпечують зручний доступ до власних бронювань, статистики та налаштувань профілю. Комплексне тестування програмного комплексу підтвердило високу якість та надійність розробленого рішення. Методологія тестування, що базувалася на безпосередній перевірці функціоналу під час розробки кожного компонента,

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

забезпечила стабільну роботу системи без використання спеціалізованих тестувальних бібліотек. Система валідації через Zod схеми ефективно обробляє користувацькі введення з детальними повідомленнями про помилки, а глобальні компоненти `error.tsx`, `not-found.tsx` та `loading.tsx` забезпечують професійну обробку різних станів додатку.

Тестування сумісності з основними веб-браузерами, включаючи Chrome, Brave, Firefox та Safari, продемонструвало стабільну роботу всіх функцій платформи. Аналіз мережевої активності через Developer Tools підтвердив ефективність TanStack Query для кешування запитів та оптимізації продуктивності.

Функціональне тестування охопило всі ключові сценарії використання, включаючи автентифікацію користувачів, управління активностями різних типів, систему бронювань та взаємодію з планувальними схемами. Перевірка адаптивності підтвердила коректне відображення інтерфейсу на різних розмірах екранів з ефективною роботою сенсорного управління на мобільних пристроях.

Тестування доступності забезпечило підтримку допоміжних технологій через семантичну HTML розмітку, `aria` атрибути та `keyboard navigation`. Стрес-тестування продемонструвало стабільну роботу під навантаженням завдяки ефективному управлінню станом та кешуванню даних. Перевірка безпеки клієнтської частини підтвердила захист від основних веб-вразливостей через правильне екранування введень та безпечне зберігання токенів автентифікації.

Результати третього розділу підтверджують успішну реалізацію фронтенд-частини платформи пошуку та управління активностями з високими показниками якості, продуктивності та користувацького досвіду. Система демонструє професійний рівень виконання з дотриманням сучасних стандартів веб-розробки та забезпечує надійну основу для подальшого розвитку та масштабування функціональності платформи.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було успішно розроблено веб-орієнтовану фронтенд-частину платформи пошуку та управління активностями з використанням сучасних JavaScript-технологій, що повністю відповідає поставленим цілям та завданням дослідження.

Проведений у першому розділі аналіз предметної області та існуючих рішень дозволив визначити ключові недоліки наявних платформ та сформулювати вимоги до розроблюваної системи. Дослідження популярних платформ Eventbrite, Meetup, Ticketmaster та Eventscase виявило специфічну орієнтацію кожної на певний сегмент користувачів з обмеженнями у функціональності. Обґрунтовано обрано гібридну архітектуру з поєднанням серверного та клієнтського рендерингу на базі Next.js 15, що забезпечує оптимальний баланс між SEO-оптимізацією та інтерактивністю користувацького інтерфейсу. Вибір технологічного стеку, включаючи React 19, TypeScript, TailwindCSS, TanStack Query та ShadCN UI, обґрунтовано потребами створення масштабованого та підтримуваного рішення з високою продуктивністю.

Другий розділ продемонстрував комплексне проектування та реалізацію фронтенд-частини з детальною розробкою всіх архітектурних компонентів. Створена система варіантів використання чітко структурувала функціональні можливості для організаторів заходів та звичайних користувачів з відповідними правами доступу. Розроблена компонентна архітектура забезпечує модульність та розділення відповідальності між системою авторизації, управлінням активностями, розкладами, локаціями з планувальними схемами, бронюваннями та статистикою. Комплексна система TypeScript інтерфейсів та Zod схем валідації гарантує типобезпечну взаємодію з API та надійну обробку користувацьких введень.

Особливої уваги заслуговує реалізація складних алгоритмів роботи системи, включаючи підтримку двох типів активностей з різною логікою бронювання, інтерактивні планувальні схеми з візуальними об'єктами різних

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

типів та гнучку систему розкладів з автоматичним генеруванням доступних часових слотів. Використання intercepting routes забезпечує плавну навігацію через модальні вікна зі збереженням контексту, а TanStack Query оптимізує роботу з серверним станом через ефективне кешування та автоматичну інвалідацію даних.

Третій розділ підтвердив успішність технічної реалізації через комплексне тестування та демонстрацію функціональності системи. Процес встановлення та налаштування продемонстрував гнучкість архітектури з можливістю розгортання в різних середовищах та конфігурації всіх компонентів системи. Структура інтерфейсу, побудована на принципах сучасного UX/UI дизайну, забезпечує інтуїтивність та зручність використання з підтримкою адаптивного дизайну та тематизації.

Розроблена платформа успішно вирішує ключові проблеми існуючих рішень через унікальну систему управління як одноразовими, так і регулярними активностями з детальними варіантами участі, інтерактивні плани місць з візуальним редактором для точного бронювання, територіальну фільтрацію з ієрархічним пошуком та статистичну аналітику для організаторів. Система реакцій та взаємодії користувачів додає соціальний аспект до платформи, відрізняючи її від конкурентів.

Архітектурна гнучкість системи забезпечує можливості для подальшого розширення функціональності, включаючи інтеграцію з платіжними системами, розширення типів активностей, додавання системи повідомлень та покращення аналітичних можливостей. Модульна структура коду та чітке розділення відповідальності спрощують процес підтримки та розвитку платформи.

Таким чином, поставлені цілі кваліфікаційної роботи повністю досягнуті через створення сучасної, функціональної та масштабованої фронтенд-частини платформи пошуку та управління активностями. Розроблене рішення демонструє високий рівень технічної майстерності, відповідає сучасним стандартам веб-розробки та створює надійну основу для подальшого розвитку платформи з урахуванням потреб різних категорій користувачів.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

ЛІТЕРАТУРА

1. Документація React [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/learn>.
2. Документація Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs>.
3. Документація Tanstack Query [Електронний ресурс] – Режим доступу до ресурсу: <https://tanstack.com/query/latest/docs/framework/react/overview>.
4. Документація Zod [Електронний ресурс] – Режим доступу до ресурсу: <https://zod.dev/>.
5. Документація React Hook Form [Електронний ресурс] – Режим доступу до ресурсу: <https://www.react-hook-form.com/api/>.
6. Документація Zustand [Електронний ресурс] – Режим доступу до ресурсу: <https://zustand.docs.pmnd.rs/getting-started/introduction>.
7. Документація ShadCN UI [Електронний ресурс] – Режим доступу до ресурсу: <https://ui.shadcn.com/docs>.
8. Документація Tailwind [Електронний ресурс] – Режим доступу до ресурсу: <https://tailwindcss.com/docs/installation/using-vite>.
9. Документація React Google Maps [Електронний ресурс] – Режим доступу до ресурсу: <https://visgl.github.io/react-google-maps/docs/get-started>.
10. Документація TypeScript [Електронний ресурс] – Режим доступу до ресурсу: <https://www.typescriptlang.org/docs/>.
11. React TypeScript Cheatsheet [Електронний ресурс] – Режим доступу до ресурсу: <https://www.typescriptlang.org/docs/>.
12. Zustand + React Query: A New Approach to State Management [Електронний ресурс] – Режим доступу до ресурсу: <https://medium.com/@freeyeon96/zustand-react-query-new-state-management-7aad6090af56>.
13. Документація Next-themes [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/next-themes>.
14. Документація Tanstack Table [Електронний ресурс] – Режим доступу до ресурсу: <https://tanstack.com/table/v8>.
15. Next.js 14 Project Structure: Best Practices [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjsstarter.com/blog/nextjs-14-project-structure-best-practices/>.
16. Next.js Apps Structure Guide for Modern Web Apps [Електронний ресурс] – Режим доступу до ресурсу: <https://www.buttercups.tech/blog/react/nextjs-app-router-structure-guide-for-modern-web-apps>.

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

17. How to Use Next.js Custom Font: A Complete Guide for 2024 [Електронний ресурс] – Режим доступу до ресурсу: <https://www.dhiwise.com/post/how-to-use-next-js-custom-font-a-complete-guide>.
18. Документація Next-themes [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/next-themes>.
19. Документація Next-themes [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/next-themes>.
20. Курс «React 2023» [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.ztu.edu.ua/course/view.php?id=5718>.
21. Курс «Моделювання та аналіз програмного забезпечення» [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.ztu.edu.ua/course/view.php?id=66>.
22. Курс «Аналіз вимог до програмного забезпечення» [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.ztu.edu.ua/course/view.php?id=30>.
23. Курс «Конструювання програмного забезпечення» [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.ztu.edu.ua/course/view.php?id=1864>.
24. Riva, Michele. "Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js, the React framework for production". – Packt Publishing, 2022. – 366 с.
25. Banks, Alex, Porcello, Eve. "Learning React: Modern Patterns for Developing React Apps". – O'Reilly Media, 2020. – 310 с.
26. Cherny, Boris. "Programming TypeScript: Making Your JavaScript Applications Scale". – O'Reilly Media, 2021. – 352 с.
27. Kato, Daishi. "Micro State Management with React Hooks: Explore custom hooks libraries like Zustand, Jotai, and Valtio to manage global states". – Packt Publishing, 2022. – 236 с.
28. Boduch, Adam. "React Cookbook: Recipes for Mastering the React Framework". – Packt Publishing, 2021. – 580 с.
29. Afonso, Daniel. "State Management with React Query: Improve developer and user experience by mastering server state in React". – Packt Publishing, 2023. – 228 с.
30. Bhat, Kartik. "Ultimate Tailwind CSS Handbook: Build sleek and modern websites with immersive UIs using Tailwind CSS". – BPB Publications, 2023. – 268 с.

ДОДАТКИ

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

activities.ts:

```

export const fetchActivities = async (
  queryParams?: ActivityQueryParams,
): Promise<PaginatedResponse<Activity>> => {
  const queryString = queryParams
    ? `?${new URLSearchParams(
      Object.entries(queryParams)
        .filter(
          ([_, value]) =>
            value !== undefined && value !== null && value !== "",
        )
        .map(([key, value]) => {
          if (Array.isArray(value)) {
            return [key, value.join(",")];
          }
          return [key, String(value)];
        })
      ).toString()}`
    : "";

  const response = await fetchWithAuth(`${ACTIVITY_ROUTES.list}${queryString}`);

  if (!response.ok) {
    const errorData = await response.json().catch(() => ({}));
    const errorMessage =
      Object.values(errorData).flat().join(". ") ||
      "Failed to fetch activities";
    throw new Error(errorMessage);
  }

  return response.json();
};

export const fetchActivity = async (id: number): Promise<Activity> => {
  const response = await fetchWithAuth(ACTIVITY_ROUTES.detail(id));

  if (!response.ok) {
    const errorData = await response.json().catch(() => ({}));
    const errorMessage =
      Object.values(errorData).flat().join(". ") || "Failed to fetch activity";
    throw new Error(errorMessage);
  }

  return response.json();
};

export const fetchMyActivities = async (): Promise<Activity[]> => {

```

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

const response = await fetchWithAuth(ACTIVITY_ROUTES.my);

if (!response.ok) {
  const errorData = await response.json().catch(() => ({}));
  const errorMessage =
    Object.values(errorData).flat().join(". ") ||
    "Failed to fetch your activities";
  throw new Error(errorMessage);
}

return response.json();
};

export const createActivity = async (
  activityData: ActivityRequest,
): Promise<Activity> => {
  const formData = new FormData();

  formData.append("name", activityData.name);
  formData.append("description", activityData.description);
  formData.append("venue_pk", activityData.venue_pk.toString());
  formData.append("type", activityData.type);

  if (activityData.type === ActivityType.OneTimeActivity) {
    const oneTimeActivity = activityData as OneTimeActivityRequest;
    formData.append("time_range", oneTimeActivity.time_range);
    formData.append("date", oneTimeActivity.date);
    formData.append("layout_pk", oneTimeActivity.layout_pk.toString());
  } else if (activityData.type === ActivityType.RecurringActivity) {
    const recurringActivity = activityData as RecurringActivityRequest;
    formData.append("schedule_pk", recurringActivity.schedule_pk.toString());
    formData.append("duration", recurringActivity.duration);
  }

  if (activityData.media_files && activityData.media_files.length > 0) {
    activityData.media_files.forEach((file) => {
      formData.append("media_files", file);
    });
  }

  if (activityData.website_links) {
    formData.append("website_links", activityData.website_links);
  }

  const response = await fetchWithAuth(ACTIVITY_ROUTES.create, {
    method: "POST",
    body: formData,
    headers: {},
  });
};

```

```

    if (!response.ok) {
      const errorData = await response.json().catch(() => ({}));
      const errorMessage =
        Object.values(errorData).flat().join(". ") || "Failed to create activity";
      throw new Error(errorMessage);
    }

    return response.json();
  };

```

fetchWithAuth.ts:

```

import { refreshTokens } from "../queries/auth";
import { API_URL } from "../constants";
import { getAuthTokens, removeAuthTokens } from "../tokens";

export const fetchWithAuth = async (
  url: string,
  options: RequestInit = {},
): Promise<Response> => {
  const { access } = getAuthTokens();
  const isFormData = options.body instanceof FormData;

  const headers = new Headers(options.headers);
  if (access) {
    headers.set("Authorization", `Bearer ${access}`);
  }

  // Only set Content-Type for JSON, not for FormData
  if (!isFormData) {
    headers.set("Content-Type", "application/json");
  }

  const response = await fetch(`${API_URL}${url}`, {
    ...options,
    headers,
  });

  if (response.status === 401) {
    const refreshed = await refreshTokens();
    if (refreshed) {
      const newHeaders = new Headers(options.headers);
      const { access: newAccess } = getAuthTokens();
      newHeaders.set("Authorization", `Bearer ${newAccess}`);

      // Only set Content-Type for JSON, not for FormData
      if (!isFormData) {
        newHeaders.set("Content-Type", "application/json");
      }
    }
  }
}

```

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

        return fetch(`${API_URL}${url}`, {
            ...options,
            headers: newHeaders,
        });
    } else {
        removeAuthTokens();
    }
}

return response;
};

```

useActivitiesQueries.ts:

"use client";

```

import { useQuery, useMutation, useQueryClient } from "@tanstack/react-query";
import {
    fetchActivities,
    fetchActivity,
    fetchMyActivities,
    createActivity,
    updateActivity,
    partialUpdateActivity,
    deleteActivity,
    publishActivity,
    hideActivity,
} from "@services/api";
import {
    Activity,
    ActivityRequest,
    PatchedActivityRequest,
    ActivityQueryParams,
    PaginatedResponse,
} from "@types/entities";

export const useActivities = (queryParams?: ActivityQueryParams) => {
    return useQuery<PaginatedResponse<Activity>>({
        queryKey: ["activities", queryParams],
        queryFn: () => fetchActivities(queryParams),
        staleTime: 5 * 60 * 1000, // 5 minutes
    });
};

export const useActivity = (id: number) => {
    return useQuery<Activity>({
        queryKey: ["activity", id],
        queryFn: () => fetchActivity(id),
        enabled: !!id,
        staleTime: 5 * 60 * 1000, // 5 minutes
    });
};

```

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

};

export const useMyActivities = () => {
  return useQuery<Activity[]>({
    queryKey: ["myActivities"],
    queryFn: () => fetchMyActivities(),
    staleTime: 5 * 60 * 1000, // 5 minutes
  });
};

export const useCreateActivity = () => {
  const queryClient = useQueryClient();

  return useMutation<Activity, Error, ActivityRequest>({
    mutationFn: (activityData: ActivityRequest) => createActivity(activityData),
    onSuccess: (newActivity) => {
      queryClient.invalidateQueries({ queryKey: ["activities"] });
      queryClient.invalidateQueries({ queryKey: ["myActivities"] });
    },
    onError: (error) => {
      return error instanceof Error
        ? error.message
        : "Failed to create activity";
    },
  });
};

export const useUpdateActivity = (id: number) => {
  const queryClient = useQueryClient();

  return useMutation<Activity, Error, ActivityRequest>({
    mutationFn: (activityData: ActivityRequest) =>
      updateActivity(id, activityData),
    onSuccess: (updatedActivity) => {
      queryClient.setQueryData(["activity", id], updatedActivity);
      queryClient.invalidateQueries({ queryKey: ["activities"] });
      queryClient.invalidateQueries({ queryKey: ["myActivities"] });
    },
    onError: (error) => {
      return error instanceof Error
        ? error.message
        : "Failed to update activity";
    },
  });
};

export const usePartialUpdateActivity = (id: number) => {
  const queryClient = useQueryClient();

  return useMutation<Activity, Error, PatchedActivityRequest>({

```

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

mutationFn: (activityData: PatchedActivityRequest) =>
  partialUpdateActivity(id, activityData),
onSuccess: (updatedActivity) => {
  queryClient.setQueryData(["activity", id], updatedActivity);
  queryClient.invalidateQueries({ queryKey: ["activities"] });
  queryClient.invalidateQueries({ queryKey: ["myActivities"] });
},
onError: (error) => {
  return error instanceof Error
    ? error.message
    : "Failed to update activity";
},
});
};

export const useDeleteActivity = () => {
  const queryClient = useQueryClient();

  return useMutation<void, Error, number>({
    mutationFn: (id: number) => deleteActivity(id),
    onSuccess: (_, id) => {
      queryClient.removeQueries({ queryKey: ["activity", id] });
      queryClient.invalidateQueries({ queryKey: ["activities"] });
      queryClient.invalidateQueries({ queryKey: ["myActivities"] });
    },
    onError: (error) => {
      return error instanceof Error
        ? error.message
        : "Failed to delete activity";
    },
  });
};

export const usePublishActivity = () => {
  const queryClient = useQueryClient();

  return useMutation<Activity, Error, number>({
    mutationFn: (id: number) => publishActivity(id),
    onSuccess: (updatedActivity, id) => {
      queryClient.setQueryData(["activity", id], updatedActivity);
      queryClient.invalidateQueries({ queryKey: ["activities"] });
      queryClient.invalidateQueries({ queryKey: ["myActivities"] });
      queryClient.invalidateQueries({ queryKey: ["options"] });
      queryClient.invalidateQueries({ queryKey: ["myOptions"] });
    },
    onError: (error) => {
      return error instanceof Error
        ? error.message
        : "Failed to publish activity";
    },
  });
};

```

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

    });
  });

  export const useHideActivity = () => {
    const queryClient = useQueryClient();

    return useMutation<Activity, Error, number>({
      mutationFn: (id: number) => hideActivity(id),
      onSuccess: (updatedActivity, id) => {
        queryClient.setQueryData(["activity", id], updatedActivity);
        queryClient.invalidateQueries({ queryKey: ["activities"] });
        queryClient.invalidateQueries({ queryKey: ["myActivities"] });
      },
      onError: (error) => {
        return error instanceof Error ? error.message : "Failed to hide activity";
      },
    });
  });
};

```

schemas.ts:

```

// Activity Schemas
const baseActivitySchema = z.object({
  name: z
    .string()
    .min(1, "Activity name is required")
    .max(64, "Activity name must be 64 characters or less"),
  description: z.string().min(1, "Description is required"),
  venue_pk: z.number().min(1, "Venue selection is required"),
  media_files: z
    .array(z.custom<File>((val) => val instanceof globalThis.File))
    .min(1, "At least one media file is required"),
  website_links: z
    .array(z.string().url("Invalid URL format"))
    .max(10, "Maximum 10 website links allowed")
    .optional()
    .default([]),
});

export const oneTimeActivitySchema = baseActivitySchema.extend({
  type: z.literal(ActivityType.OneTimeActivity),
  time_range: z
    .string()
    .min(1, "Time range is required")
    .regex(
      /^(([0-1]?[0-9]|2[0-3]):[0-5][0-9]-([0-1]?[0-9]|2[0-3]):[0-5][0-9]$/,
      "Time range must be in HH:MM-HH:MM format",
    )
    .refine(
      (range) => {
        if (!range || !range.includes("-")) return false;

```

		Пархомчук Р.В.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

    const [start, end] = range.split("-");
    if (!start || !end) return false;

    const [startHour, startMin] = start.split(":").map(Number);
    const [endHour, endMin] = end.split(":").map(Number);

    const startMinutes = startHour * 60 + startMin;
    const endMinutes = endHour * 60 + endMin;

    return startMinutes < endMinutes;
  },
  {
    message: "Start time must be earlier than end time",
  },
),
date: z
  .string()
  .min(1, "Date is required")
  .regex(/^\d{4}-\d{2}-\d{2}$/, "Date must be in YYYY-MM-DD format")
  .refine(
    (dateStr) => {
      const date = new Date(dateStr);
      const today = new Date();
      today.setHours(0, 0, 0, 0);
      return date >= today;
    },
    {
      message: "Date cannot be in the past",
    },
  ),
layout_pk: z.number().min(1, "Layout selection is required"),
});

export const recurringActivitySchema = baseActivitySchema.extend({
  type: z.literal(ActivityType.RecurringActivity),
  duration: z
    .string()
    .min(1, "Duration is required")
    .regex(
      /^(?([0-1]?[0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9])$/,
      "Duration must be in HH:MM:SS format (e.g., '01:00:00', '00:30:00')",
    ),
  schedule_pk: z.number().min(1, "Schedule selection is required"),
});

// Enhanced activity schema using discriminated union
export const enhancedActivitySchema = z.discriminatedUnion("type", [
  oneTimeActivitySchema,
  recurringActivitySchema,
]);

```