МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра дискретного аналізу та інтелектуальних систем
(повна назва кафедри)

Кваліфікаційна (бакалаврська) робота

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ БЛАГОДІЙНОГО ПРОЄКТУ

Виконав: студент групи ПМі-43с
спеціальності
122 «Комп'ютерні науки»
(шифр і назва спеціальності)

Вавpикович М. І.
(підпис) (прізвище та ініціали)

Керівник доц. Позднякова І. В.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

2024

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладної математики та інформатики
Кафедра Дискретного аналізу та інтелектуальних систем
Спеціальність 122 «Комп'ютерні науки»
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____

" 30" серпня 2023 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ (БАКАЛАВРСЬКУ) РОБОТУ СТУДЕНТУ

Вавриковичу Михайлу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи
Розробка веб-застосунку для благодійного проєкту

керівник роботи Позднякова Інна Володимирівна, кандидат фіз.-мат. наук, доцент ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від **" 27 "вересня 2023 року № 24**

2. Строк подання студентом роботи 14.06.2024р.

3. Вихідні дані до роботи

JavaScript, Python, NextJS, NestJS, PostgreSQL, Google Maps API

4. Зміст кваліфікаційної (бакалаврської) роботи (перелік питань, які потрібно розробити)

Розробка веб-застосунку

Створення алгоритму оптимізації маршрутів доставки подарунків

Перевірка веб-застосунку на коректність роботи

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання **30 серпня 2023 р.**

Керівник роботи _____ доц. Позднякова І. В.____
(підпис) (прізвище та ініціали)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної (бакалаврської) роботи	Строк виконання етапів роботи	Примітка
1	Затвердження теми дипломної роботи	Вересень	Виконано
2	Розробка веб-застосунку	Листопад	Виконано
3	Перевірка веб-застосунку на коректність роботи	Лютий	Виконано
4	Написання дипломної роботи	Травень	Виконано

Студент _____ Вавpикович М. І.
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Розробка веб-застосунку для благодійного проєкту є надзвичайно актуальною у сучасному світі. Благодійні організації відіграють важливу роль у наданні допомоги соціально незахищеним верствам населення, зокрема дітям. Традиційні методи організації благодійних заходів часто потребують великих ресурсів і значного часу на координацію. Використання сучасних технологій може значно полегшити ці процеси, зробивши їх більш ефективними і зручними. Веб-застосунки дозволяють автоматизувати багато аспектів благодійної діяльності, таких як збір інформації, планування маршрутів доставки, облік отримувачів допомоги, ведення звітності, впорядкування різних списків тощо. Це забезпечує більш точне і швидке виконання благодійних заходів, зменшує адміністративні витрати і покращує координацію між учасниками проєкту.

Актуальність. Цей веб-застосунок буде використовуватись благодійним проєктом “Миколай про тебе не забуде” у місті Тернопіль, що займається збором і розвезенням близько 1 тис подарунків дітям із неблагополучних сімей на свято Миколая. Організатори стикнулися із проблемою оптимізувати упорядкування списків дітей та створення маршрутів розвезення подарунків, тому цей застосунок буде використовуватися на практиці та має змогу в перспективі реалізовуватися багаторічно. Він зможе пришвидшити роботу проєкту та дасть можливість організаторам масштабуватися.

Мета роботи. Метою цієї роботи є створення веб-застосунку, який забезпечить ефективну організацію та проведення благодійних заходів у рамках конкретного проєкту. Застосунок має забезпечити оптимізацію маршрутів доставки подарунків, спрощення обліку отримувачів допомоги та підвищення загальної ефективності роботи проєкту.

Завдання роботи. Розробити веб-застосунок, який забезпечить ефективну організацію та проведення благодійних заходів у рамках конкретного проєкту. Це включає створення бази даних для зберігання інформації про дітей, їхні адреси та стан доставки подарунків, а також розробку адміністративної панелі

для введення та оновлення даних, забезпечення безпеки та цілісності збереженої інформації.

Також потрібно створити алгоритм оптимізації маршрутів доставки подарунків. Цей алгоритм має враховувати географічне положення адрес та інші релевантні фактори. Основною метою є забезпечення ефективного використання ресурсів та мінімізація часу доставки.

Перевірити веб-застосунок на коректність роботи. Це завдання включає тестування всіх функціональних компонентів веб-застосунку у реальних умовах. Необхідно провести тестування бази даних, алгоритму оптимізації маршрутів і адміністративної панелі з використанням реальних даних конкретного благодійного проєкту. Важливо перевірити, чи всі частини системи працюють належним чином, чи немає помилок або збоїв у роботі, а також чи відповідають результати роботи веб-застосунку очікуваним параметрам. За результатами тестування слід провести аналіз і внести необхідні корективи для покращення роботи застосунку.

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

На сьогоднішній день існує багато інструментів, які можуть бути використані для окремих аспектів організації благодійних заходів. Наприклад, для ведення обліку можна використовувати Google Sheets [1]. Це дозволяє зберігати інформацію про дітей, їхні адреси та стан доставки подарунків у зручному форматі таблиці з можливістю спільного доступу та редагування. Також можна використовувати Airtable [2], що забезпечує гнучкий та інтерактивний підхід до управління базами даних завдяки зручному інтерфейсу та можливостям автоматизації. Ще однією альтернативою є Notion [3], який теж пропонує універсальні інструменти для організації даних.

Для побудови маршрутів доставки подарунків можуть бути використані сервіси, такі як Google Maps [4] або Circuit [5]. Google Maps пропонує функціонал для планування маршрутів з урахуванням реальних умов на дорогах, таких як затори та перекриття. Circuit є спеціалізованим сервісом для оптимізації маршрутів, який дозволяє враховувати безліч факторів для забезпечення ефективної доставки. Також можна використовувати MapQuest [6], який надає інструменти для планування маршрутів та визначення оптимальних шляхів.

Проте, незважаючи на наявність цих інструментів, на ринку немає комплексного рішення, яке б одночасно покривало всі необхідні вимоги для організації роботи конкретного проєкту. Існуючі сервіси можуть забезпечити окремі функції, такі як ведення обліку або оптимізація маршрутів, але не пропонують інтегрованого підходу, який би поєднував усі аспекти роботи в одному зручному інтерфейсі. Тому виникає потреба у розробці спеціалізованого веб-застосунку, який об’єднує всі необхідні функції для ефективної організації та проведення благодійних заходів.

РОЗДІЛ 2. ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

2.1 JavaScript

JavaScript [7] є мовою програмування, що виконується на стороні клієнта і дозволяє створювати інтерактивні та динамічні веб-сторінки. Ця мова використовується для маніпулювання елементами HTML-документа [8], зміни їхніх властивостей, додавання анімацій і реакції на дії користувачів, такі як кліки або введення даних. JavaScript дозволяє виконувати асинхронні запити до серверу, що забезпечує оновлення даних без перезавантаження сторінки. Завдяки підтримці у всіх сучасних браузерах, JavaScript є невід'ємною частиною сучасної веб-розробки і основою для багатьох фреймворків і бібліотек.

2.2 Node.js

Node.js [9] є середовищем виконання JavaScript, яке працює на сервері. Вона базується на рушії V8 [10] від Google Chrome [11] і дозволяє виконувати JavaScript-код поза браузером. Node.js відрізняється подієво-орієнтованою, неблокуючою архітектурою, що робить її особливо ефективною для розробки масштабованих мережевих застосунків, таких як веб-сервери та API [12]. Середовище Node.js забезпечує високу продуктивність і швидкодію завдяки асинхронній обробці подій, що дозволяє обслуговувати велику кількість одночасних з'єднань. Node.js має багатий набір вбудованих модулів і величезну екосистему бібліотек, доступних через npm (Node Package Manager) [13], що спрощує розробку і прискорює впровадження різноманітних функцій у веб-застосунки.

2.3 TypeScript

TypeScript [14] є надбудовою над JavaScript, яка додає статичну типізацію до мови програмування. Це дозволяє виявляти помилки на етапі написання коду, що сприяє зменшенню кількості багів і підвищенню якості програмного забезпечення. TypeScript зберігає всі можливості JavaScript, оскільки компілюється у звичайний JavaScript, який може виконуватися у будь-якому браузері чи серверному середовищі. Підтримка новітніх функцій JavaScript у TypeScript дозволяє використовувати сучасні можливості мови, незалежно від

того, чи підтримує їх поточна версія JavaScript. Це робить TypeScript популярним вибором для створення масштабованих і надійних веб-застосунків.

2.4 React

React [15] є бібліотекою JavaScript для побудови користувацьких інтерфейсів, розробленою для створення динамічних і інтерактивних веб-застосунків. Вона дозволяє розробникам створювати компоненти, які можна повторно використовувати в різних частинах застосунку, що значно спрощує процес розробки та підтримки коду. Основною особливістю React є використання віртуального DOM [16], що дозволяє ефективно оновлювати та рендерити компоненти, забезпечуючи високу продуктивність навіть при складних змінах у структурі сторінки. React також підтримує JSX [17], розширення синтаксису JavaScript, що дозволяє писати код, який виглядає як HTML, що робить його більш інтуїтивно зрозумілим і зручним для розробки користувацьких інтерфейсів.

2.5 NextJS

Next.js [18] є фреймворком для розробки веб-застосунків на основі React, який забезпечує рендеринг на стороні сервера та генерацію статичних сайтів. Він дозволяє створювати високопродуктивні веб-застосунки з мінімальними зусиллями. Основні можливості Next.js включають автоматичний розподіл коду, маршрутизацію на основі файлів, підтримку статичного та серверного рендерингу, а також інтеграцію з API для динамічного отримання даних. Використання Next.js спрощує налаштування і управління сучасними веб-застосунками, надаючи вбудовані засоби для покращення продуктивності і зручності розробки.

2.6 RESTful API

RESTful API (Representational State Transfer Application Programming Interface) [19] є архітектурним стилем для побудови веб-сервісів, який використовує стандартні HTTP-методи [20] для виконання операцій над

ресурсами. Ключовими принципами REST є використання уніфікованих інтерфейсів, відокремлення клієнта і сервера, відсутність збереження стану між запитами та можливість кешування. RESTful API дозволяє взаємодіяти з ресурсами через чітко визначені ендпоінти, використовуючи методи GET для отримання даних, POST для створення нових ресурсів, PUT або PATCH для оновлення існуючих та DELETE для видалення. Завдяки простоті, гнучкості та сумісності з різними клієнтами і серверами, RESTful API є популярним підходом для створення масштабованих і ефективних веб-сервісів.

2.7 NestJS

NestJS [21] є фреймворком для розробки серверних застосунків, побудованим на основі Node.js. Він використовує TypeScript як основну мову програмування і надихається архітектурними принципами, що забезпечують модульність, масштабованість і легкість у підтримці коду. NestJS пропонує потужні вбудовані засоби для створення RESTful API, підтримуючи такі концепції, як залежності, модулі, контролери і сервіси. Завдяки використанню об'єктно-орієнтованого підходу та впровадженню декораторів, NestJS забезпечує чисту і зрозумілу архітектуру, що сприяє швидкому розробленню та легкому розширенню застосунків. Фреймворк також підтримує роботу з базами даних, що робить його універсальним рішенням для створення серверних додатків будь-якої складності.

2.8 PostgreSQL

PostgreSQL [22] є потужною системою управління реляційними базами даних з відкритим вихідним кодом. Вона відома своєю надійністю, стабільністю і відповідністю стандартам SQL[23]. PostgreSQL підтримує складні запити, транзакції, зовнішні ключі, тригери і процедури. Система забезпечує високу продуктивність завдяки можливостям паралельного виконання запитів, індексування та оптимізації запитів. PostgreSQL також підтримує розширення, що дозволяє додавати нові функціональні можливості без необхідності змінювати основний код. Завдяки своїй гнучкості і потужним інструментам для

роботи з даними, PostgreSQL є популярним вибором для багатьох критично важливих застосунків.

2.9 Prisma

Prisma [24] є ORM (Object-Relational Mapping) інструментом для TypeScript і JavaScript, який забезпечує спрощений та типізований доступ до баз даних. Він дозволяє взаємодіяти з базою даних через зручний та інтуїтивно зрозумілий API, генеруючи типізовані запити, що зменшує кількість помилок на етапі розробки. Однією з основних функцій Prisma є Prisma Client, яка забезпечує типізований і автоматично згенерований клієнт для взаємодії з базою даних. Завдяки своїй інтеграції з TypeScript та підтримці сучасних функцій баз даних, Prisma забезпечує ефективний і надійний спосіб роботи з PostgreSQL у сучасних веб-застосунках.

2.10 Python

Python [25] є високорівневою мовою програмування загального призначення, яка відома своєю простотою та читабельністю коду. Завдяки своєму зрозумілому синтаксису та великій стандартній бібліотеці, Python підходить для різних видів розробки, включаючи веб-застосунки, автоматизацію систем та наукові дослідження. Python підтримує кілька парадигм програмування, таких як об'єктно-орієнтоване, процедурне та функціональне програмування. Він широко використовується для аналізу даних та машинного навчання завдяки наявності спеціалізованих інструментів та бібліотек. Завдяки активній спільноті та постійним оновленням, Python залишається одним з найпопулярніших і затребуваних мов програмування у світі.

2.11 scikit-learn

scikit-learn [26] є бібліотекою машинного навчання для мови програмування Python, яка забезпечує простий і ефективний інструментарій для аналізу даних. Вона пропонує широкий спектр алгоритмів машинного навчання, включаючи методи класифікації, регресії, кластеризації та зниження

розмірності. scikit-learn використовує інтуїтивно зрозумілий API, що дозволяє легко застосовувати складні моделі до реальних даних. Завдяки своїй простоті у використанні та потужності, scikit-learn є важливим інструментом для дослідників та інженерів, які працюють у галузі машинного навчання і аналізу даних.

2.12 Vercel Functions

Vercel Functions [27] є функціями без сервера, що інтегруються з платформою Vercel [28] для розробки та розгортання веб-застосунків. Вони дозволяють виконувати серверну логіку без необхідності управління інфраструктурою серверів. Vercel Functions підтримують написання коду на JavaScript, TypeScript, Python та інших мовах, забезпечуючи гнучкість у виборі інструментів для розробки. Вони автоматично масштабуються залежно від навантаження, забезпечуючи високу продуктивність і доступність. Завдяки простій інтеграції з фронтенд-застосунками, Vercel Functions є ефективним рішенням для реалізації API, обробки вебхуків та інших серверних завдань у сучасних веб-застосунках.

2.13 Google Maps API

Google Maps API [29] є набором інструментів для інтеграції картографічних сервісів Google Maps у веб-застосунки. Він дозволяє додавати інтерактивні карти, маршрути, місця та інші географічні дані на веб-сторінки. API підтримує різні функціональні можливості, такі як відображення маркерів, створення поліліній та полігонів, а також надання інформації про місця. Крім того, Google Maps API пропонує сервіси для геокодування, обчислення відстаней та побудови маршрутів. Використовуючи Google Maps API, розробники можуть створювати потужні і зручні для користувачів картографічні рішення, які легко інтегруються з іншими веб-технологіями.

2.14 Docker

Docker [30] є платформою для контейнеризації, яка дозволяє розробникам створювати, розгортати і запускати застосунки в ізольованих середовищах, відомих як контейнери. Контейнери забезпечують узгодженість роботи застосунків, незалежно від середовища, в якому вони виконуються, що спрощує процес розробки, тестування і розгортання. Docker використовує образи, які містять все необхідне для запуску застосунку, включаючи код, системні інструменти, бібліотеки та налаштування.

Контейнеризація за допомогою Docker забезпечує швидке розгортання та масштабування веб-застосунків, а також покращує ефективність використання ресурсів. Кожен контейнер працює в ізоляції, що підвищує безпеку та стабільність системи.

Використання Docker у веб-застосунку забезпечує узгодженість середовищ розробки, тестування та продакшену, що зменшує ризик виникнення проблем при перенесенні застосунку між різними етапами розробки.

РОЗДІЛ 3. ОПИС АРХІТЕКТУРИ

3.1 Клієнт-серверна архітектура

Веб-застосунок побудований на основі клієнт-серверної архітектури [31], що є однією з найбільш поширених моделей для розробки веб-застосунків. Вона передбачає розподіл функцій між клієнтською частиною та серверною частиною. Цей підхід дозволяє створювати ефективні та масштабовані системи, які легко підтримувати і розширювати.

Основна ідея клієнт-серверної архітектури полягає у тому, що клієнт надсилає запити до сервера, який обробляє ці запити, виконує необхідні операції з даними та повертає відповідь клієнту. Клієнтська частина відповідає за відображення інтерфейсу користувача і взаємодію з ним, тоді як серверна частина виконує всю бізнес-логіку, обробляє дані і управляє з'єднанням з базою даних.

Ключовими аспектами клієнт-серверної архітектури є:

1. Розподіл обов'язків: Клієнт і сервер мають чітко розподілені обов'язки. Клієнт займається відображенням інтерфейсу та взаємодією з користувачем, тоді як сервер обробляє запити та управляє даними.
2. Масштабованість: Клієнт-серверна архітектура дозволяє легко масштабувати систему. Сервер може обробляти запити від великої кількості клієнтів одночасно, а клієнтська частина може працювати незалежно на різних пристроях.
3. Гнучкість: Компоненти клієнтської та серверної частини можуть бути розроблені, оновлені та замінені незалежно один від одного, що забезпечує гнучкість в розробці та підтримці системи.
4. Безпека: Серверна частина забезпечує централізоване управління безпекою даних, аутентифікацією та авторизацією користувачів, що підвищує загальний рівень безпеки системи.

3.2 Зберігання даних

Для забезпечення ефективного зберігання і управління даними в веб-застосунку використовується реляційна база даних PostgreSQL. Ця система управління базами даних забезпечує високу надійність, масштабованість і продуктивність, що є критично важливим для зберігання великої кількості даних про користувачів, адреси, дітей, маршрути та конфігурації маршрутів. PostgreSQL підтримує складні запити, транзакції та індексацію, що дозволяє швидко і точно отримувати необхідну інформацію та ефективно управляти нею.

3.2.1 Опис таблиць та їх атрибутів

База даних складається з наступних таблиць:

Таблиця User

Ця таблиця зберігає інформацію про користувачів системи.

- id: Унікальний ідентифікатор користувача.
- firstName: Ім'я користувача.
- lastName: Прізвище користувача.

- phone: Телефон користувача.
- email: Електронна пошта користувача.
- password: Пароль користувача.
- role: Роль користувача.
- isActive: Статус активності користувача.
- deactivationReason: Причина деактивації.
- createdAt: Дата створення запису.
- updatedAt: Дата останнього оновлення запису.

Таблиця Address

Ця таблиця зберігає інформацію про адреси.

- id: Унікальний ідентифікатор адреси.
- city: Місто.
- street: Вулиця.
- streetNumber: Номер будинку.
- flatNumber: Номер квартири.
- latitude: Географічна широта.
- longitude: Географічна довгота.
- createdAt: Дата створення запису.
- updatedAt: Дата останнього оновлення запису.

Таблиця Child

Ця таблиця зберігає інформацію про дітей.

- id: Унікальний ідентифікатор дитини.
- firstName: Ім'я дитини.
- lastName: Прізвище дитини.
- birthYear: Рік народження.
- gender: Стать.
- phone: Телефон дитини.
- notes: Примітки.
- needStatus: Статус потреби.

- isActive: Статус активності.
- deactivationReason: Причина деактивації.
- addressId: Ідентифікатор адреси.
- createdAt: Дата створення запису.
- updatedAt: Дата останнього оновлення запису.

Таблиця Route

Ця таблиця зберігає інформацію про маршрути доставки подарунків.

- id: Унікальний ідентифікатор маршруту.
- year: Рік маршруту.
- number: Номер маршруту.
- routeConfigId: Ідентифікатор конфігурації маршруту.
- createdAt: Дата створення запису.
- updatedAt: Дата останнього оновлення запису.

Таблиця RouteConfig

Ця таблиця зберігає конфігурації маршрутів.

- id: Унікальний ідентифікатор конфігурації маршруту.
- year: Рік конфігурації.
- isEditable: Статус редагування.
- isCreated: Статус створення.
- isCompleted: Статус завершення.
- createdAt: Дата створення запису.
- updatedAt: Дата останнього оновлення запису.

Таблиця RouteAdress

Ця таблиця забезпечує зв'язок між маршрутами і адресами.

- routeId: Ідентифікатор маршруту.
- addressId: Ідентифікатор адреси.
- createdAt: Дата створення запису.
- updatedAt: Дата останнього оновлення запису.

3.3 Серверна частина

3.3.1 Загальний опис структури

Серверна частина веб-застосунку реалізована з використанням фреймворку NestJS, який забезпечує модульну архітектуру та легку інтеграцію з іншими бібліотеками та сервісами. NestJS побудований на основі Node.js та використовує TypeScript.

Основні компоненти серверної частини включають модулі, контролери та сервіси, які взаємодіють між собою, забезпечуючи виконання бізнес-логіки, обробку запитів і відповідей, а також взаємодію з базою даних.

- Модулі. Модулі є основними блоками NestJS-застосунку. Вони організовують код у логічні групи і дозволяють структурувати застосунок за функціональними областями. Кожен модуль може містити контролери, сервіси та інші провайдери.
- Контролери. Контролери обробляють HTTP-запити, що надходять від клієнта, і повертають відповідні відповіді. Вони відповідають за маршрутизацію запитів до відповідних сервісів і обробку результатів.
- Сервіси. Сервіси містять бізнес-логіку застосунку і взаємодіють з базою даних або іншими зовнішніми системами. Вони використовуються контролерами для виконання основних операцій, таких як створення, читання, оновлення або видалення даних.

3.3.2 Комунікація з базою даних

Для комунікації з базою даних використовується Prisma – сучасний ORM (Object-Relational Mapping), який спрощує роботу з базами даних і забезпечує зручний інтерфейс для виконання запитів. Prisma дозволяє ефективно взаємодіяти з базою даних PostgreSQL, використовуючи сучасні практики та інструменти. Налаштування Prisma включає конфігурацію підключення до бази даних та визначення моделей даних. Використовуючи Prisma Client, можна виконувати складні запити з фільтрами, сортуванням і пагінацією, що

забезпечує високу гнучкість у роботі з даними. Це значно спрощує розробку бізнес-логіки та взаємодію з базою даних, дозволяючи зосередитися на основних завданнях застосунку.

3.3.3 Документування

Документування є критично важливим етапом у розробці серверної частини застосунку, оскільки воно забезпечує зручний доступ до інформації про API для розробників, тестувальників та інших зацікавлених сторін. У цьому проєкті для документування використовується Swagger [32], інтегрований за допомогою NestJS.

NestJS надає зручні інструменти для автоматичного генерування документації API у форматі Swagger. Це дозволяє створювати зрозумілі та детальні описи всіх кінцевих точок API, включаючи параметри запитів, тіла запитів та відповіді, а також можливі коди статусу. Документація, створена за допомогою Swagger, не лише полегшує процес розробки, але й значно сприяє подальшому тестуванню.

3.3.4 Контейнеризація

Використання Docker для контейнеризації застосунку на NestJS та бази даних PostgreSQL дозволяє досягти високої продуктивності та надійності. Застосунок у контейнері NestJS містить увесь необхідний код та залежності, що робить його легко переносимим між різними середовищами. Аналогічно, контейнер PostgreSQL забезпечує стабільну роботу бази даних із мінімальними зусиллями на налаштування. Це зменшує ризики, пов'язані з конфліктами версій та залежностей, та сприяє більш ефективному використанню ресурсів системи.

Контейнеризація за допомогою Docker є важливим кроком у забезпеченні ефективності та надійності роботи веб-застосунку. Вона забезпечує стабільність, передбачуваність та легкість у розгортанні та масштабуванні, що є критично важливим для успішної реалізації проєкту

3.4 Безсерверні функції

Використання безсерверних функцій у цьому проєкті дозволяє ефективно інтегрувати Python-код із основним застосунком, не потребуючи розгортання окремого серверного середовища для виконання невеликих частин коду. Для реалізації безсерверних функцій використовується платформа Vercel Functions, яка забезпечує зберігання та виконання коду на Python.

Однією з основних причин використання безсерверних функцій є необхідність застосування бібліотеки scikit-learn для обробки даних. Оскільки ці функції займають лише невелику частину загального обсягу коду, немає сенсу піднімати окремий проєкт на Python. Замість цього, Python-код зберігається та виконується за допомогою Vercel Functions, що значно спрощує процес управління та розгортання.

Застосунок на NestJS звертається до безсерверних функцій через HTTP-запити, що забезпечує зручну та ефективну взаємодію між компонентами системи. Це забезпечує високу гнучкість та масштабованість, дозволяючи швидко впроваджувати нові функціональні можливості без значних затрат на інфраструктуру.

3.5 Клієнтська частина

3.5.1 Загальний опис структури

Клієнтська частина застосунку розроблена з використанням фреймворку Next.js, який забезпечує гнучкість та високу продуктивність для створення сучасних веб-застосунків. Структура клієнтської частини побудована таким чином, щоб забезпечити легкість у розробці та підтримці коду. Основні компоненти застосунку організовані у відповідні папки, що забезпечує зрозумілість і логічну послідовність. У цьому проєкті використовується директорія app, де кожен файл автоматично перетворюється на окремий маршрут. Це значно спрощує управління маршрутами у застосунку, забезпечуючи плавний і швидкий перехід між сторінками.

Клієнтська частина, розроблена на базі Next.js, забезпечує високу продуктивність, зручність у використанні та легкість у підтримці, що є важливими аспектами для успішної реалізації та експлуатації веб-застосунку.

3.5.2 Комунікація із серверною частиною

Для забезпечення ефективної взаємодії між клієнтською та серверною частинами застосунку, використовується автогенерація запитів на основі документації серверної частини. Це дозволяє автоматично створювати необхідні запити до API, базуючись на детальній документації, що значно спрощує процес розробки та знижує ймовірність помилок.

Документація серверної частини, створена за допомогою Swagger, містить усю необхідну інформацію про кінцеві точки API, включаючи методи запитів, параметри, які вони приймають, та типи даних, що повертаються у відповідях.

РОЗДІЛ 4. ОПИС ІНТЕРФЕЙСУ ТА КОРИСТУВАЦЬКИХ МОЖЛИВОСТЕЙ

4.1 Авторизація адміністраторів

При запуску застосунку, буде відображена сторінка авторизації. Тут адміністратор може увійти в свій облікові запис, використовуючи імейл та наданий йому пароль. Після цього потрібно нажати кнопку ‘Увійти’ для відправлення запиту на сервер. В разі успішного проходження цього кроку, адміністратор входить у застосунок. Автентифікація забезпечує безпеку та обмежує доступ до приватної інформації

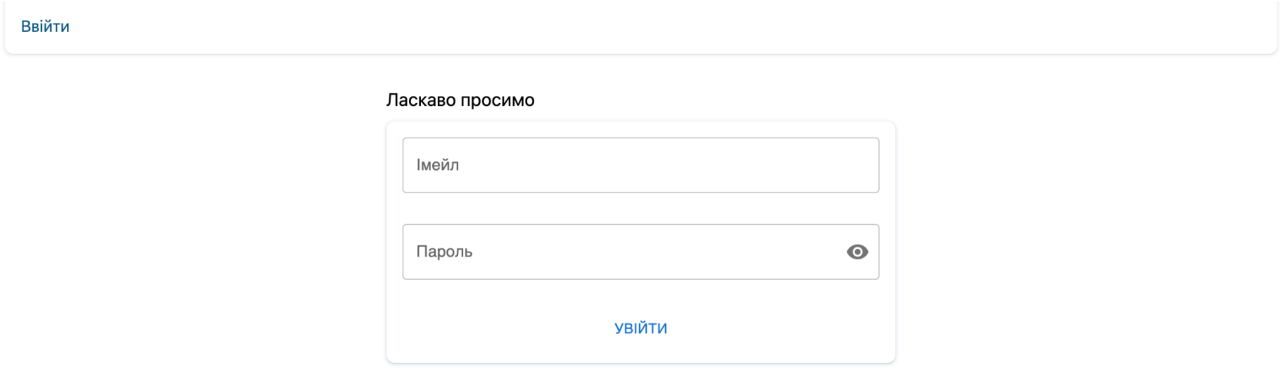

Рисунок 4.1 – Сторінка авторизації

4.2 Управління адміністраторами

4.2.1 Таблиця адміністраторів

На цій сторінці знаходиться таблиця, яка відображає всіх адміністраторів у застосунку з їхніми основними даними, такими як ім'я та прізвище, тощо.

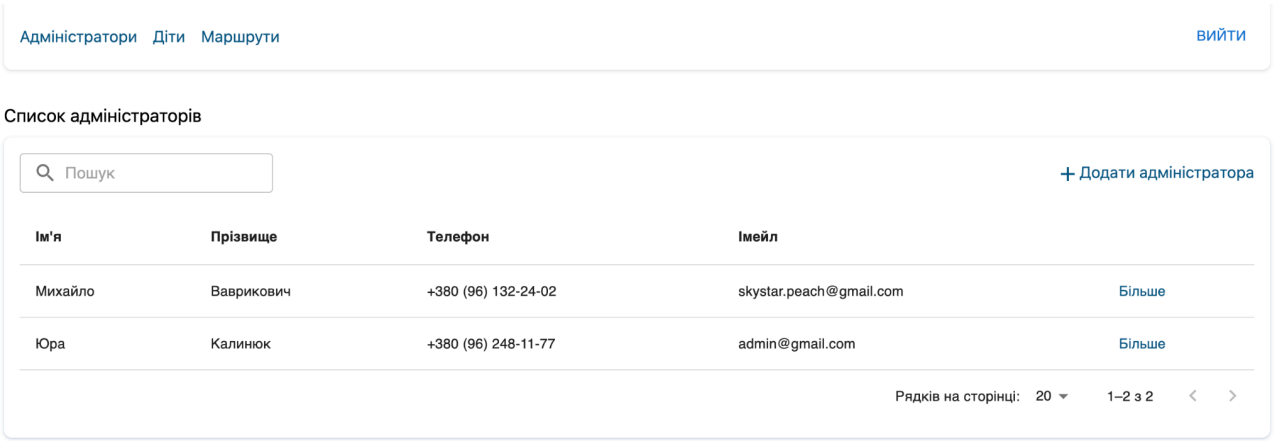

Рисунок 4.2.1 – Таблиця адміністраторів

4.2.2 Пошук та сортування

Над таблицею знаходиться поле пошуку за даними адміністратора. Також при натисканні на відповідний стовпчик у таблиці відбувається сортування за цим значенням.

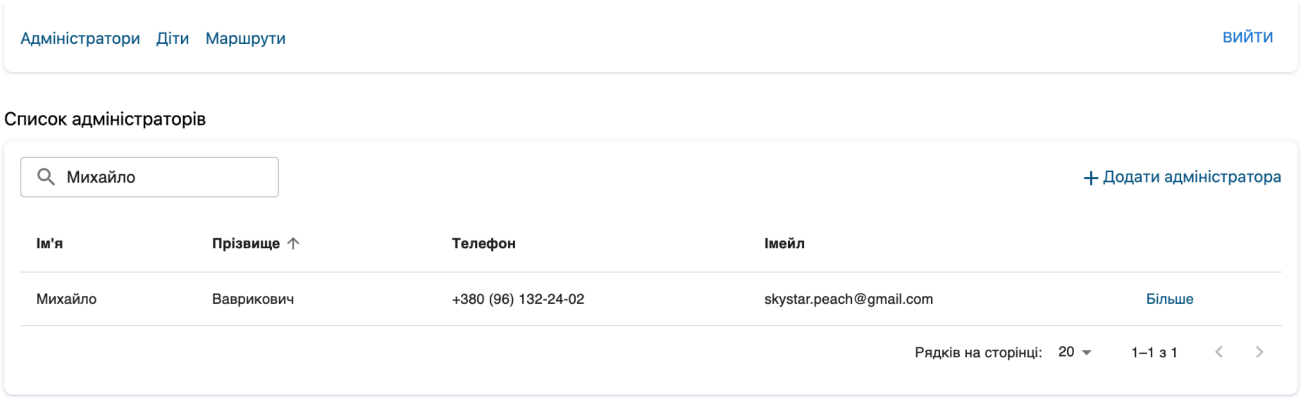

Рисунок 4.2.2 – Пошук та сортування адміністраторів

4.2.3 Інформація про конкретного адміністратора

В рядку таблиці присутня кнопка ‘Більше’, яка перенаправляє на сторінку з більш детальною інформацією про конкретного адміністратора.

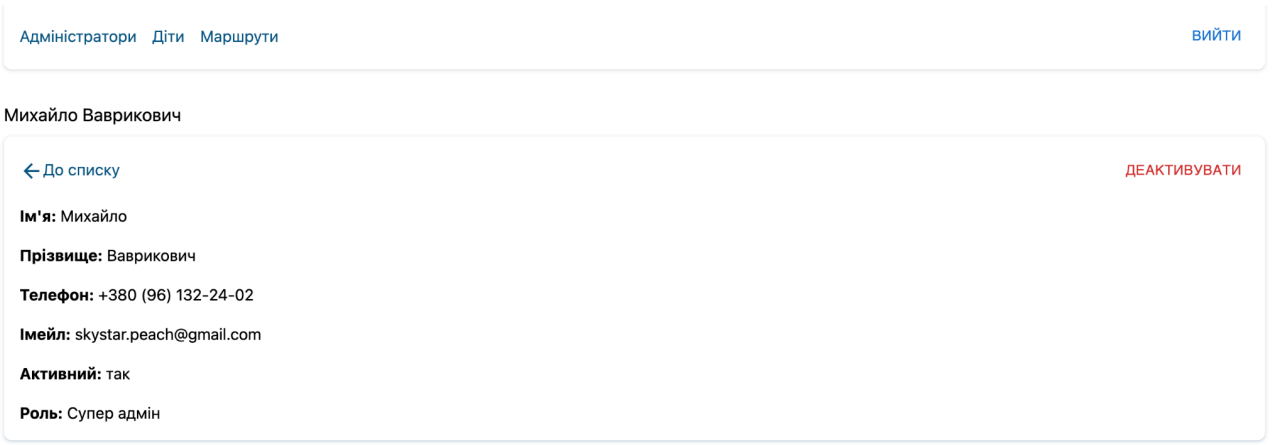

Рисунок 4.2.3 – Інформація про конкретного адміністратора

4.2.4 Деактивація

Також на даній сторінці з інформацією присутня кнопка ‘Деактивувати’, яка дозволяє деактивувати адміністратора, після чого він не зможе автентифікуватися в системі і відповідно виконувати в ній якісь дії. Ця функція доступна лише адміністратору в якого є роль ‘супер адміністратор’.

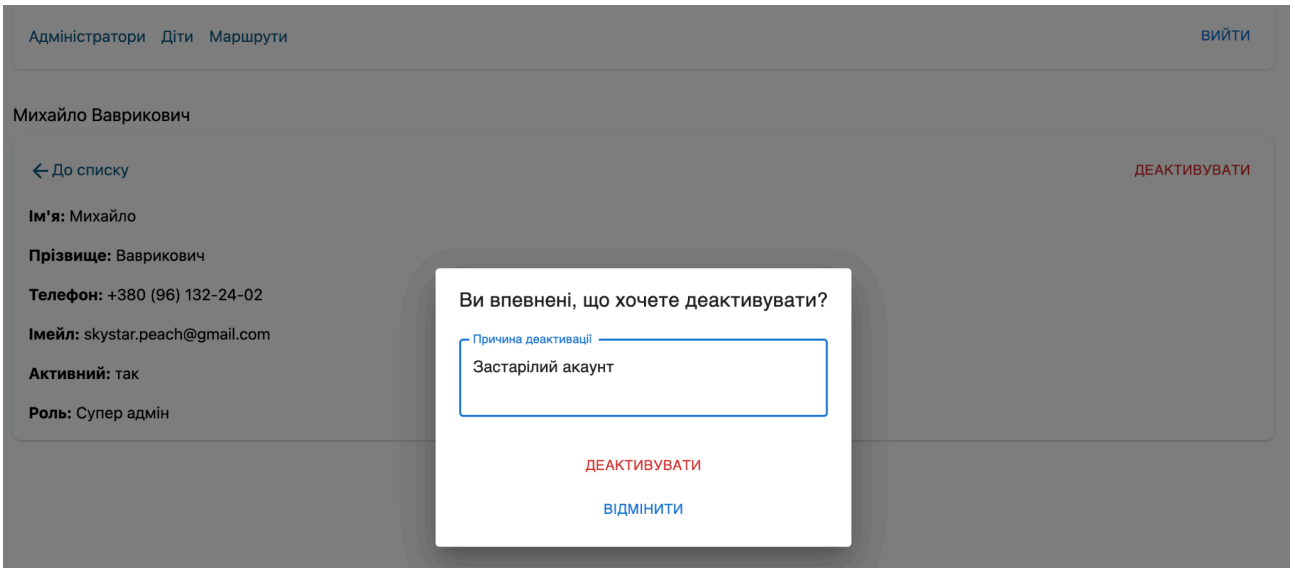

Рисунок 4.2.4 – Вікно деактивації адміністратора

4.2.5 Активація

Також є можливість активувати раніше деактивованого адміністратора. Ця функція доступна лише адміністратору в якого є роль ‘супер адміністратор’.

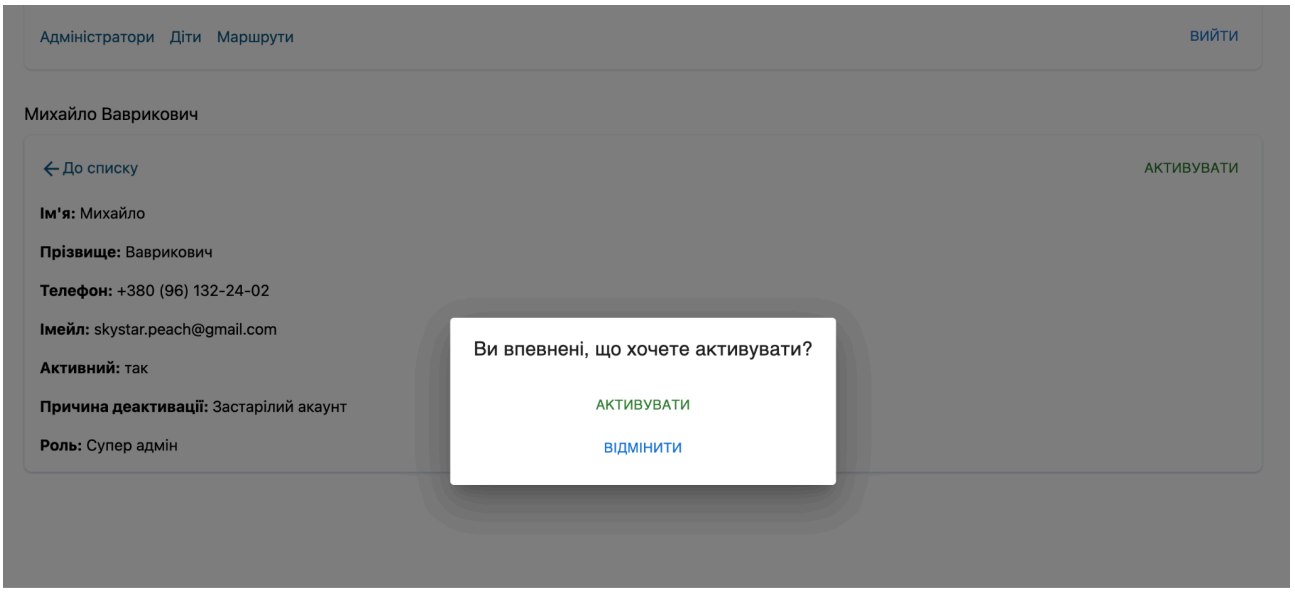

Рисунок 4.2.5 – Вікно активації адміністратора

4.2.6 Додавання адміністратора в систему

В застосунку присутній інтерфейс, який дозволяє створювати нових адміністраторів в системі. Після створення адміністратора він появиться в таблиці. Ця функція доступна лише адміністратору в якого є роль ‘супер адміністратор’.

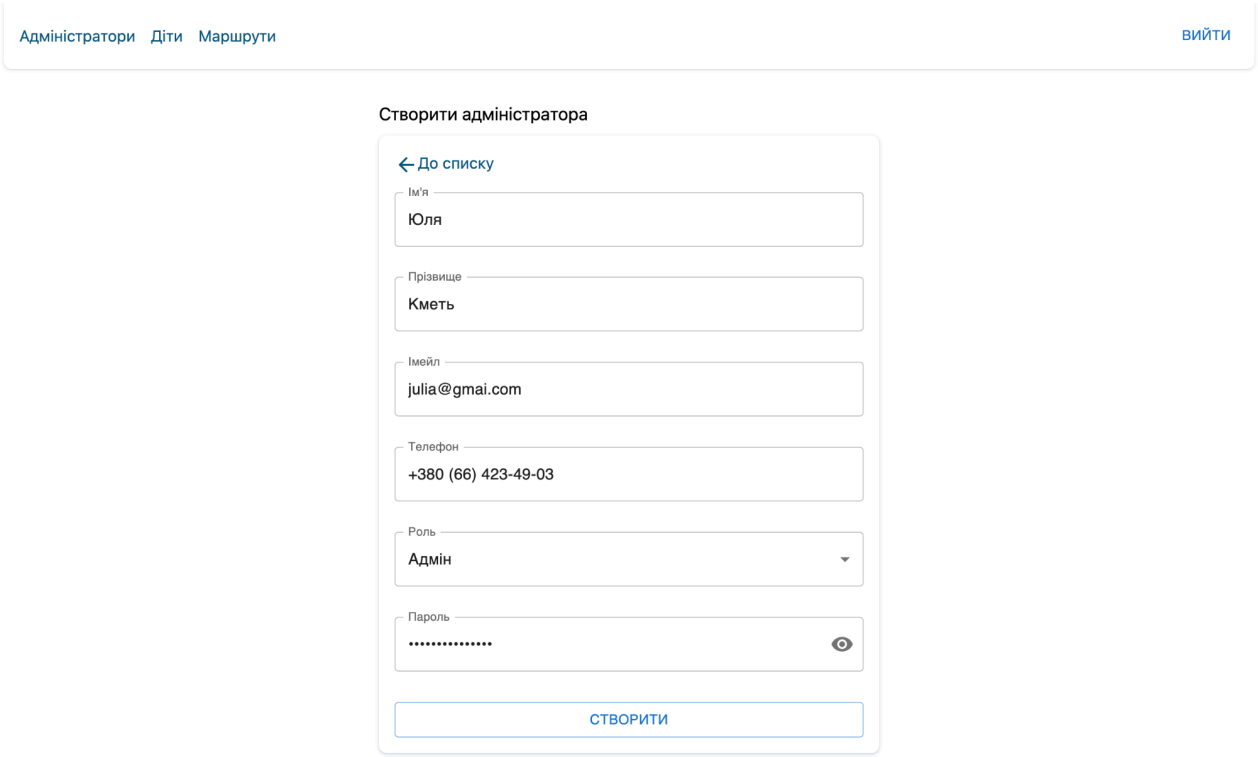

Рисунок 4.2.5 – Інтерфейс додавання адміністратора в систему

4.3 Адміністрування дітей

4.3.1 Таблиця дітей

На цій сторінці знаходиться таблиця, яка відображає усіх дітей доданих в базу даних з їхніми основними даними, такими як ім'я та прізвище, тощо.

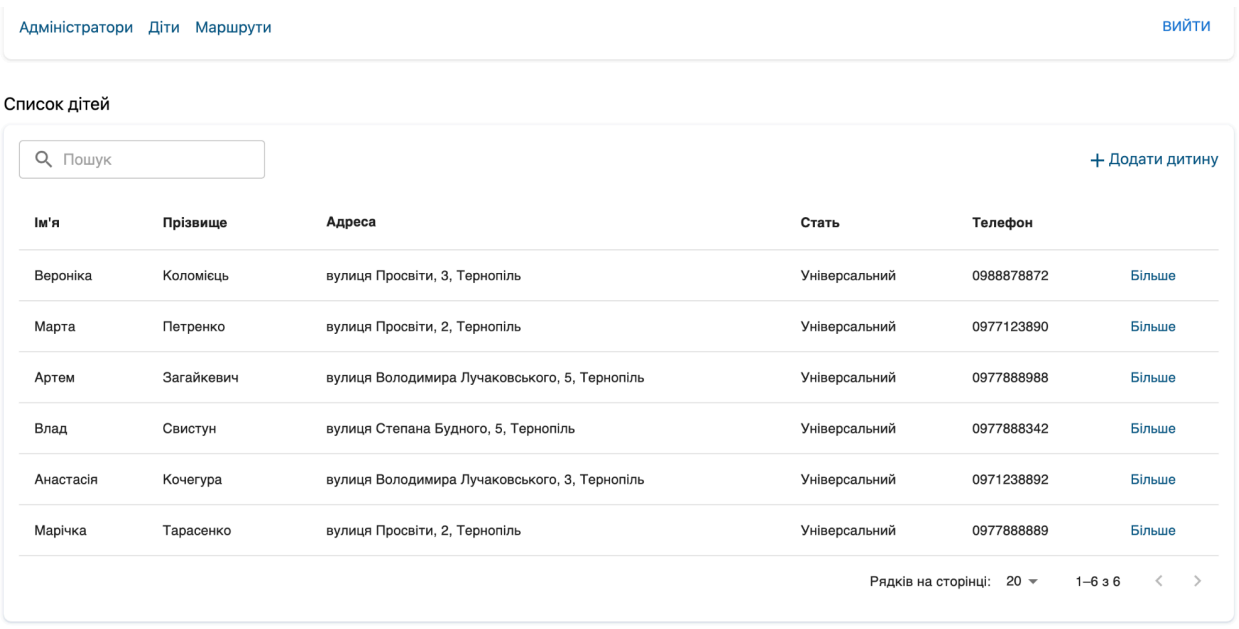

Рисунок 4.2.1 – Таблиця дітей

4.3.2 Пошук та сортування

Над таблицею знаходиться поле пошуку за даними дитини. Також при натисканні на відповідний стовпчик у таблиці відбувається сортування за цим значенням.

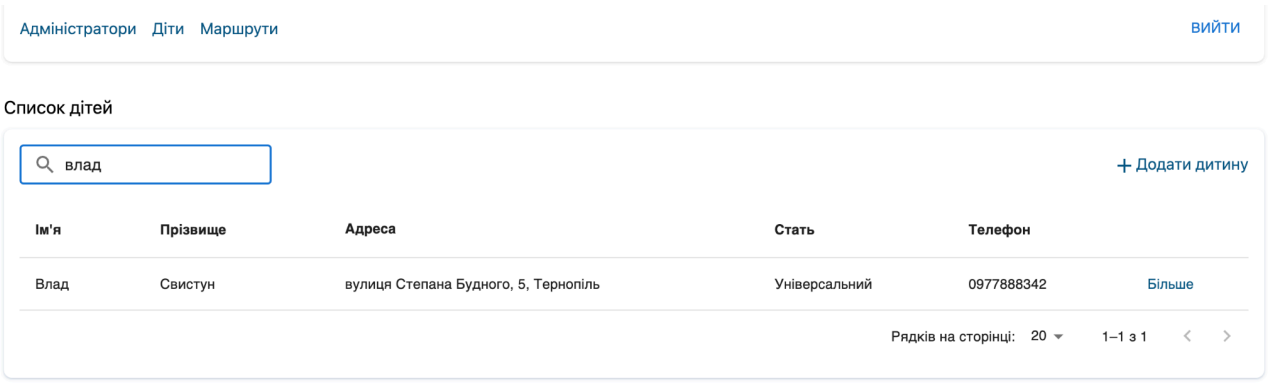

Рисунок 4.2.2 – Пошук та сортування інформації про дітей

4.3.3 Інформація про конкретну дитину

В рядку таблиці присутня кнопка ‘Більше’, яка перенаправляє на сторінку з більш детальною інформацією про конкретну дитину.

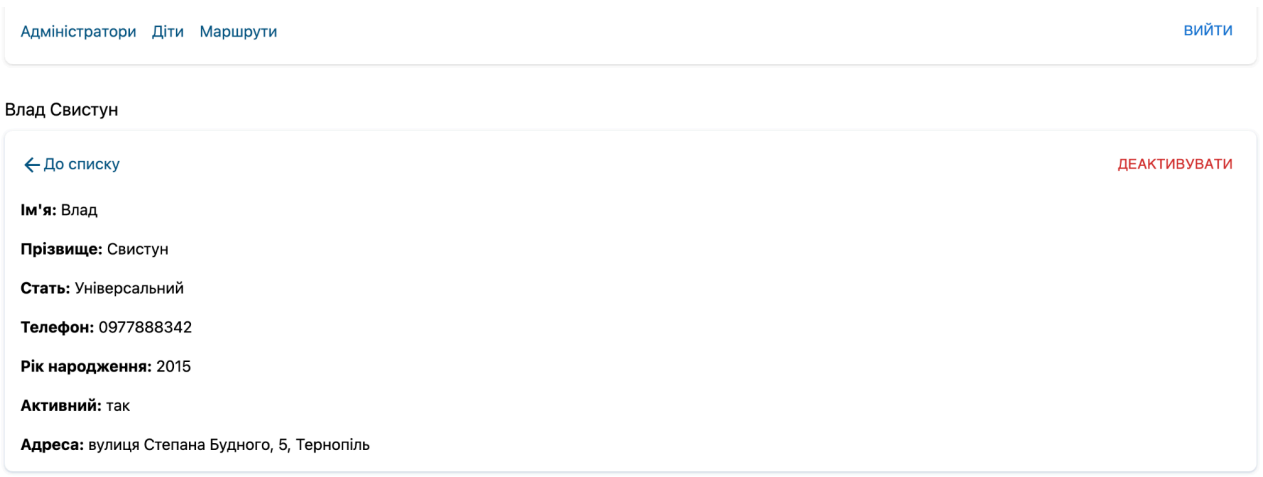

Рисунок 4.2.3 – Інформація про конкретну дитину

4.3.4 Деактивація

Також на даній сторінці з інформацією присутня кнопка ‘Деактивувати’, яка дозволяє деактивувати дитину, після чого вона не буде потрапляти у новостворені маршрути. Ця функція доступна лише адміністратору в якого є роль ‘супер адміністратор’.

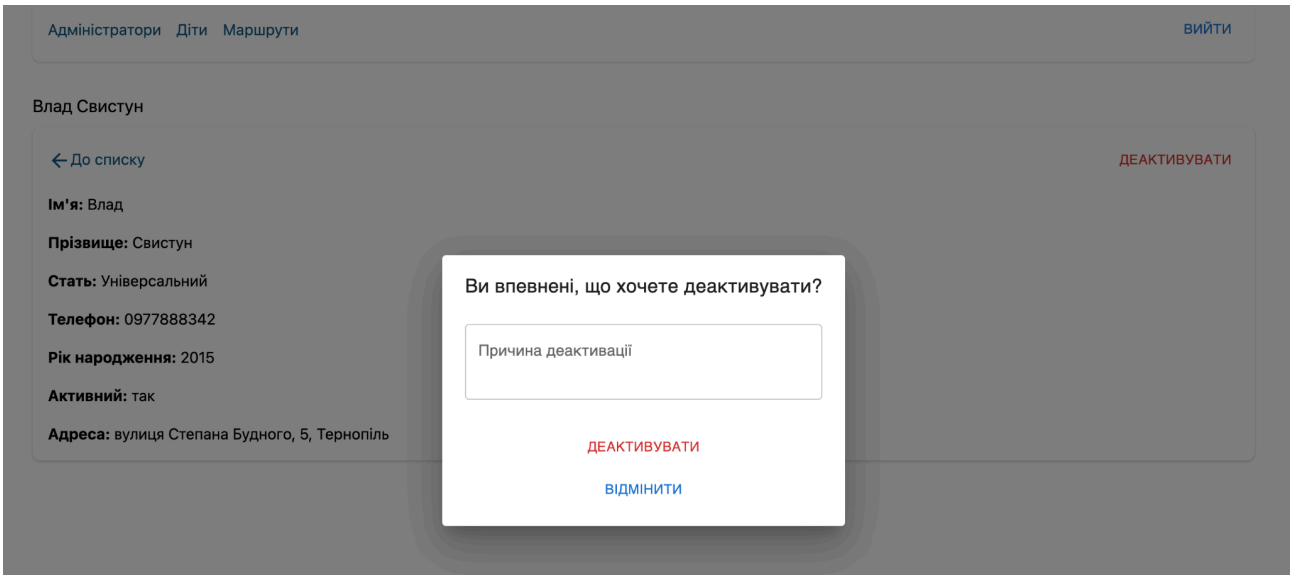

Рисунок 4.2.4 – Вікно деактивації дитини

4.3.5 Активація

Також є можливість активувати раніше деактивованого дитину. Ця функція доступна лише адміністратору в якого є роль ‘супер адміністратор’.

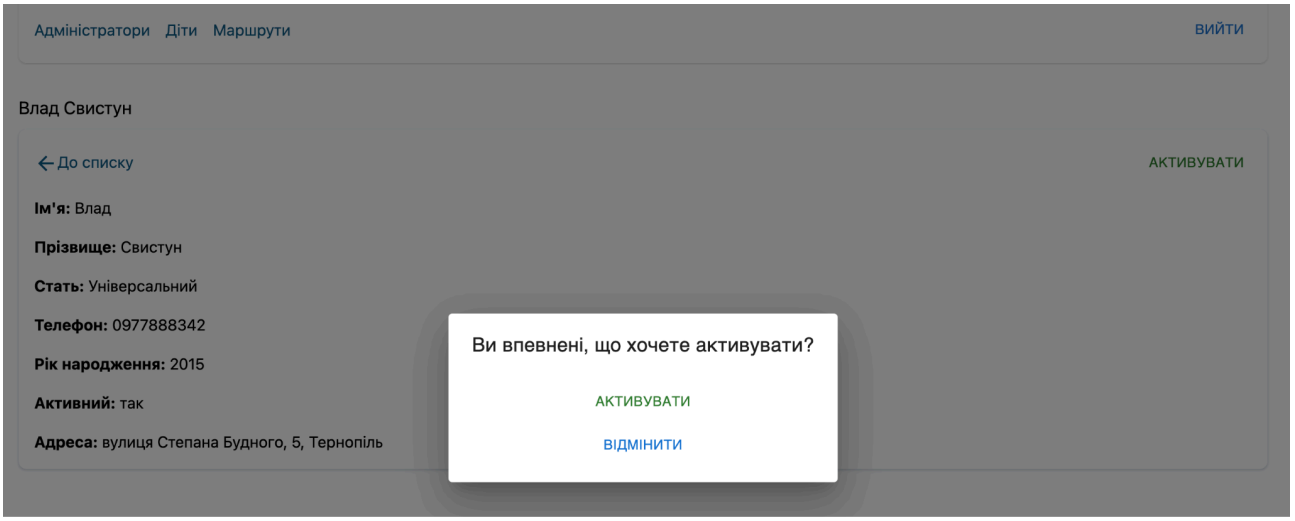

Рисунок 4.2.5 – Вікно активації дитини

4.3.6 Додавання дитини в базу даних

В застосунку присутній інтерфейс, який дозволяє додавати нових дітей в базу даних. Після створення дитина появиться в таблиці. Ця функція доступна лише адміністратору в якого є роль ‘супер адміністратор’.

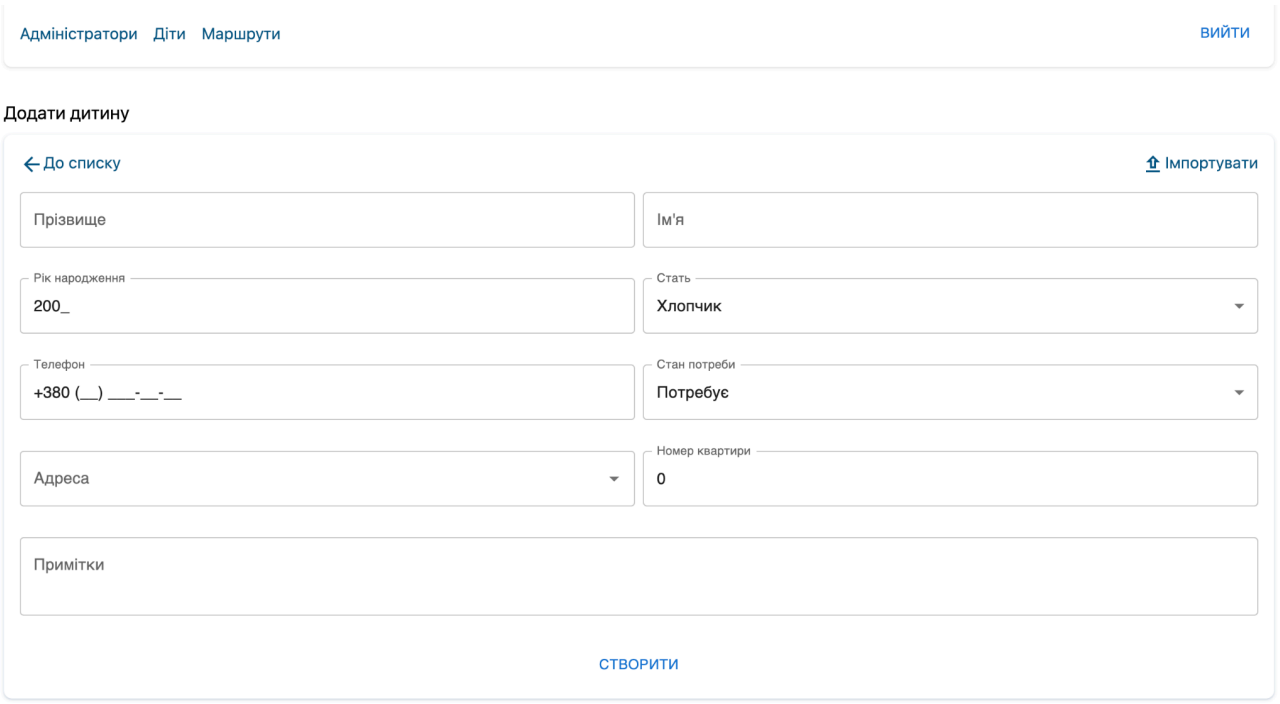

Рисунок 4.3.6.1 – Інтерфейс додавання дитини в базу даних

Також, якщо потрібно імпортувати дітей із .csv файлу, то при натисканні на кнопку ‘Імпортувати’, з'являється відповідний інтерфейс

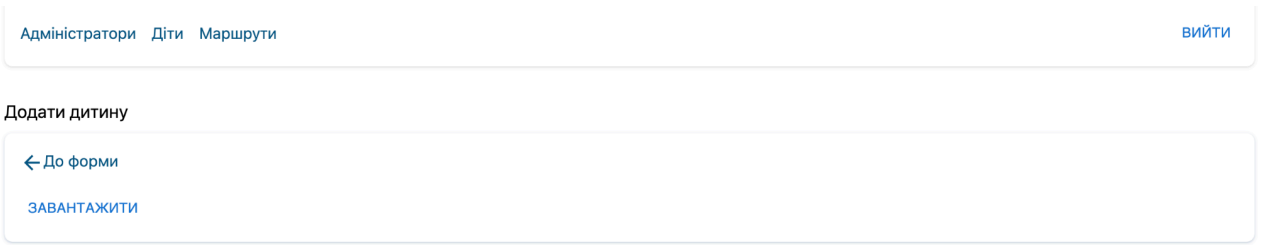

Рисунок 4.3.6.2 – Інтерфейс імпорту дітей із .csv файлу

Після натискання на кнопку ‘Завантажити’ можна обрати бажаний файл, після чого в інтерактивному режимі імпортувати дітей. Система автоматично визначає чи така дитина вже є в базі даних або чи інформація про неї містить помилки.

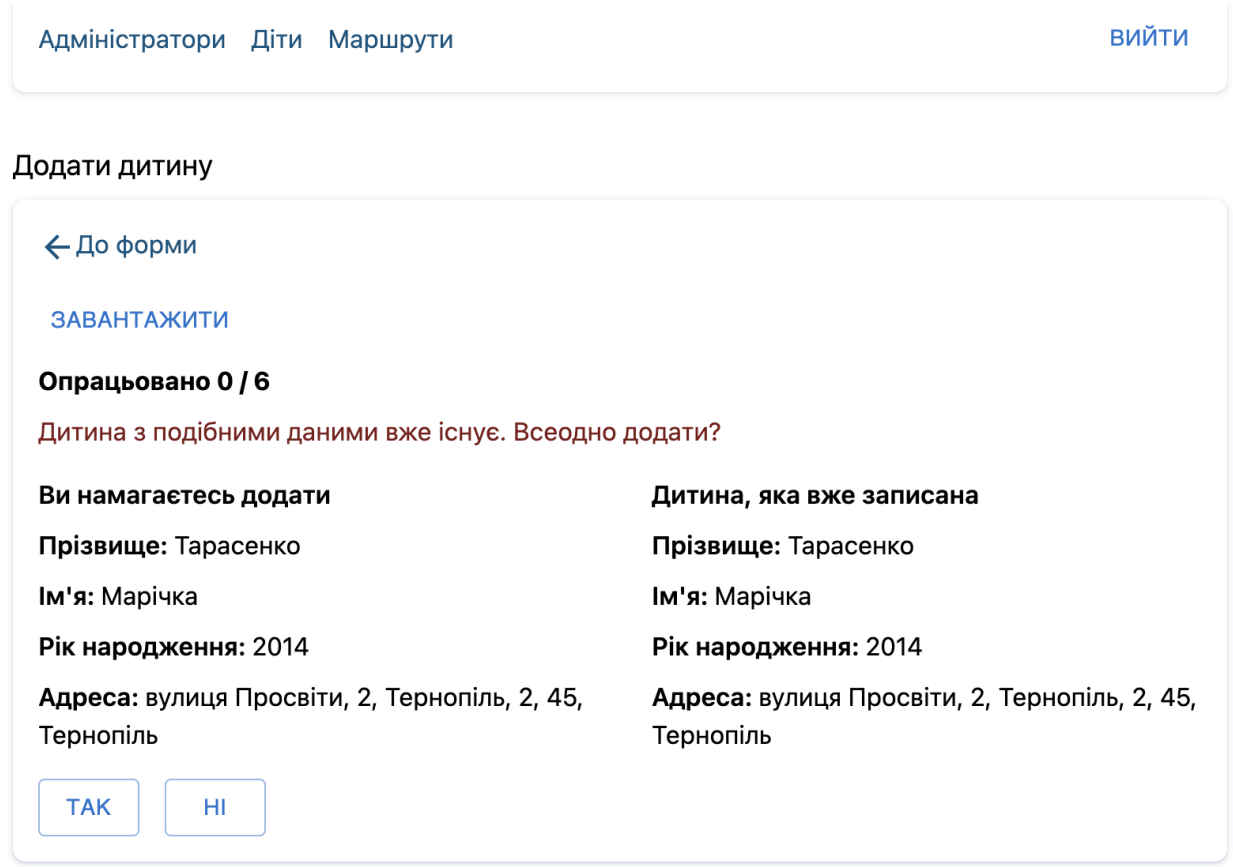

Рисунок 4.3.6.3 – Інтерактивний імпорт

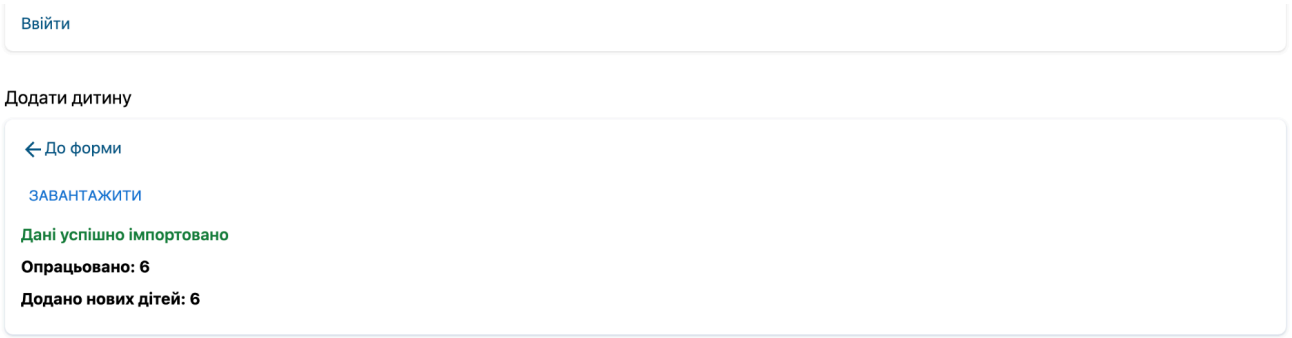

Рисунок 4.3.6.4 – Результати імпорту

4.4 Адміністрування маршрутів

4.4.1 Таблиця маршрутів

На цій сторінці знаходиться таблиця, яка відображає усі групи створених маршрутів.

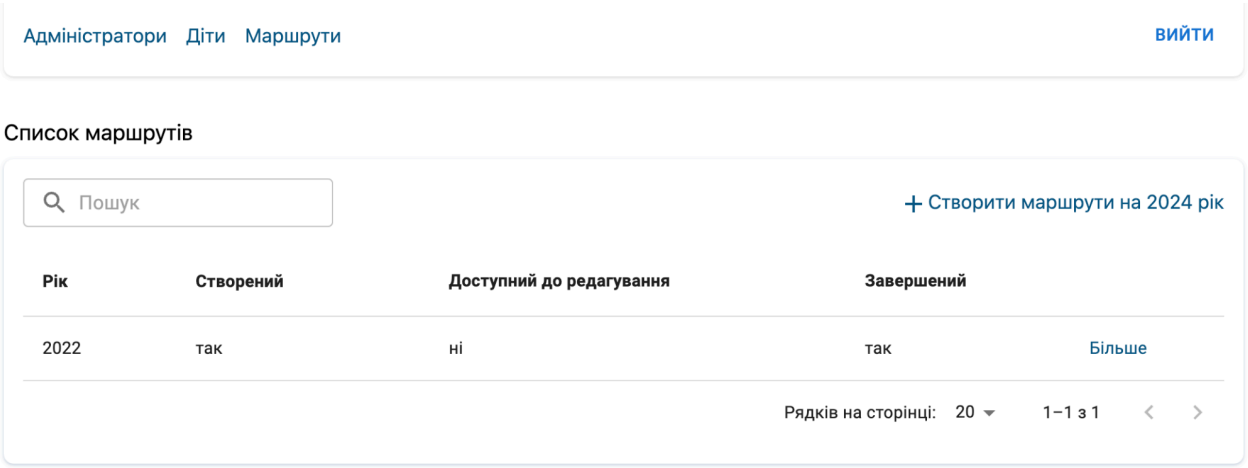

Рисунок 4.4.1 – Таблиця маршрутів

4.4.2 Пошук та сортування

Над таблицею знаходиться поле пошуку за даними маршруту. Також при натисканні на відповідний стовпчик у таблиці відбувається сортування за цим значенням.

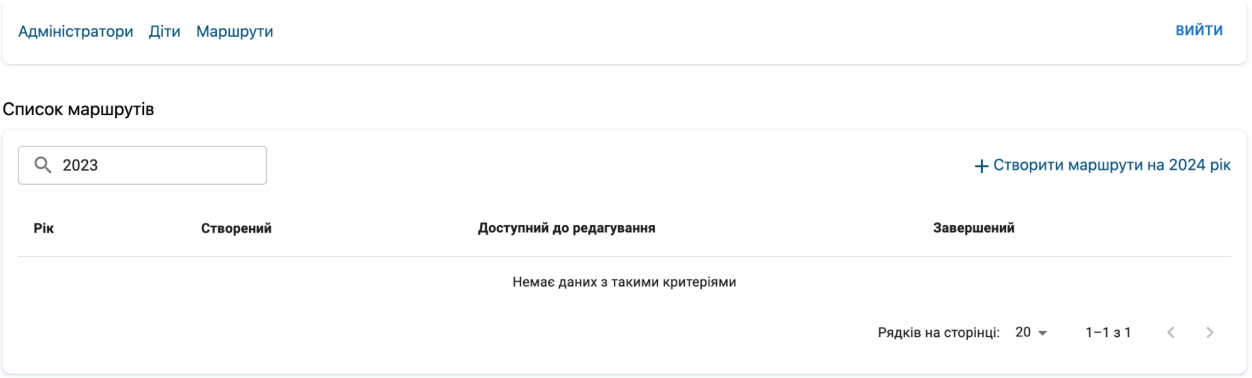

Рисунок 4.4.2 – Пошук та сортування груп маршрутів

4.4.3 Інформація про конкретний маршрут

В рядку таблиці присутня кнопка ‘Більше’, яка перенаправляє на сторінку з більш детальною інформацією про конкретний маршрут. Тут можна фільтрувати маршрути за роками та їх номером, а також скопіювати URL адресу маршруту для Google Maps

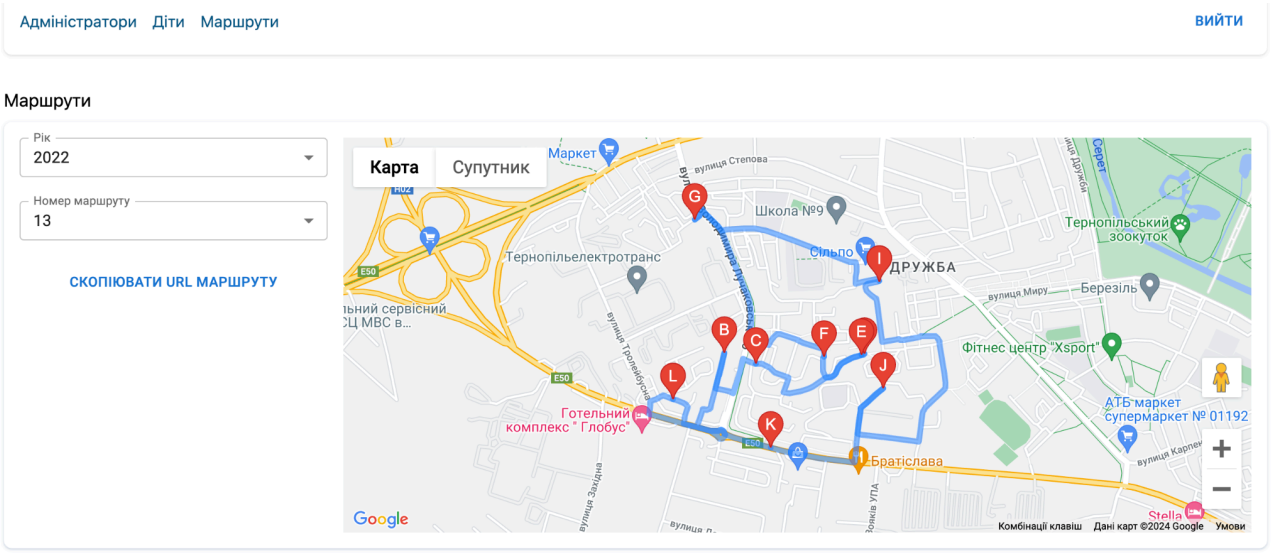

Рисунок 4.4.3 – Інформація про конкретну дитину

4.4.4 Створення маршрутів

В застосунку присутній інтерфейс, який дозволяє створювати маршрути на поточний рік. Ця функція доступна лише адміністратору в якого є роль ‘супер адміністратор’.

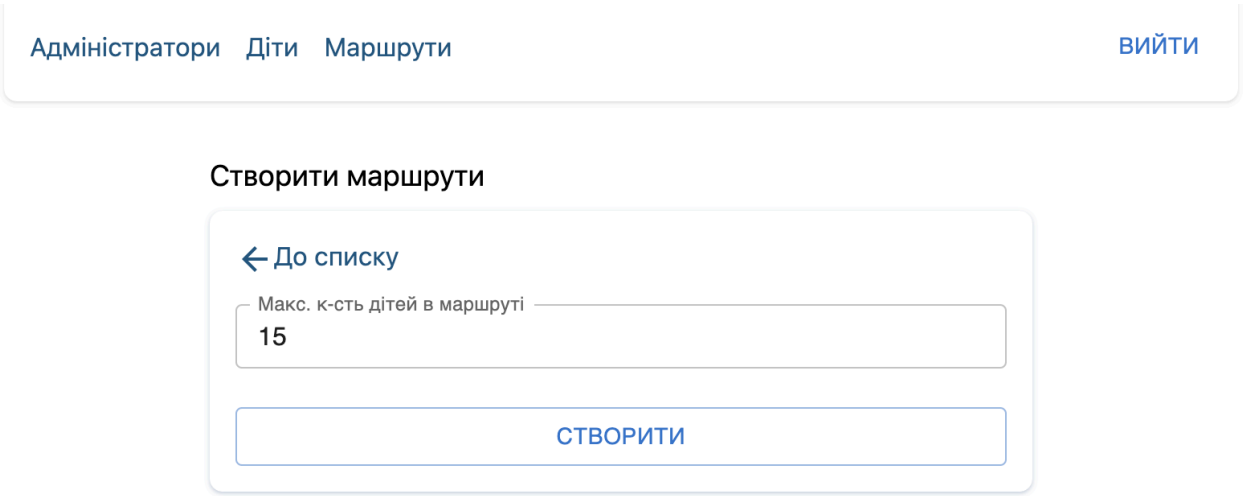

Рисунок 4.4.4 – Інтерфейс створення маршрутів

ВИСНОВКИ

Було розроблено веб-застосунок, який забезпечує ефективну організацію та проведення благодійних заходів у рамках конкретного проєкту. Це включало в себе створення бази даних для зберігання інформації про дітей, їхні адреси та стан доставки подарунків, а також розробку адміністративної панелі для введення та оновлення даних, забезпечення безпеки та цілісності збереженої інформації.

Було створено алгоритм оптимізації маршрутів доставки подарунків. Цей алгоритм враховує географічне положення адрес та інші релевантні фактори.

Веб-застосунок перевірено на коректність роботи. Це завдання включало тестування всіх функціональних компонентів веб-застосунку у реальних умовах. Було проведено тестування бази даних, алгоритму оптимізації маршрутів і адміністративної панелі з використанням реальних даних конкретного благодійного проєкту. Перевірено, чи всі частини системи працюють належним чином, чи немає помилок або збоїв у роботі, а також чи відповідають результати роботи веб-застосунку очікуваним параметрам. За результатами тестування проведено аналіз і внесено необхідні корективи для покращення роботи застосунку.

СПИСОК ВИКОРИСТАНОЇ ДЖЕРЕЛ

1. Google Таблиці: онлайн-редактор електронних таблиць
   URL: https://www.google.com/intl/uk_ua/sheets/about/ (дата звернення 10.03.2024)
2. Airtable: The platform to build next-gen apps
   URL: https://www.airtable.com/ (дата звернення 10.03.2024)
3. Notion: Your connected workspace for wiki, docs & projects
   https://www.notion.so/ (дата звернення 10.03.2024)
4. Google Maps
   URL: https://en.wikipedia.org/wiki/Google_Maps (дата звернення 10.03.2024)
5. Circuit – We build software that delivers
   URL: https://getcircuit.com/ (дата звернення 10.03.2024)
6. Official MapQuest - Maps, Driving Directions, Live Traffic
   URL: https://www.mapquest.com/ (дата звернення 10.03.2024)
7. JavaScript
   URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення 10.03.2024)
8. HTML basics
   URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics (дата звернення 10.03.2024)
9. Node.js — Run JavaScript Everywhere
   URL: https://nodejs.org/en (дата звернення 10.03.2024)
10. V8 JavaScript engine
    URL: https://v8.dev/ (дата звернення 10.03.2024)
11. Google Chrome - The Fast & Secure Web Browser Built to be Yours
    URL: https://www.google.co.uk/chrome/ (дата звернення 10.03.2024)
12. What is an API?
    URL: https://aws.amazon.com/what-is/api/ (дата звернення 10.03.2024)

13.NPM

URL: https://www.npmjs.com/ (дата звернення 10.03.2024)

14. TypeScript: JavaScript With Syntax For Types

URL: https://www.typescriptlang.org/ (дата звернення 10.03.2024)

15. React

URL: https://react.dev/ (дата звернення 10.03.2024)

16. Introduction to the DOM

URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction (дата звернення 10.03.2024)

17. React JSX

URL: https://www.w3schools.com/react/react_jsx.asp (дата звернення 10.03.2024)

18. The React Framework for the Web

URL: https://nextjs.org/ (дата звернення 10.03.2024)

19. What is RESTful API?

URL: https://aws.amazon.com/what-is/restful-api/ (дата звернення 10.03.2024)

20. HTTP request methods

URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods (дата звернення 10.03.2024)

21. NestJS - A progressive Node.js framework

URL: https://nestjs.com/ (дата звернення 10.03.2024)

22. PostgreSQL: The World's Most Advanced Open Source Relational Database

URL: https://www.postgresql.org/ (дата звернення 10.03.2024)

23. SQL

URL: https://en.wikipedia.org/wiki/SQL (дата звернення 10.03.2024)

24. Prisma | Simplify working and interacting with databases

URL: https://www.prisma.io/ (дата звернення 10.03.2024)

25. Welcome to Python.org

URL: https://www.python.org/ (дата звернення 10.03.2024)

26. scikit-learn: Machine Learning in Python
URL: https://scikit-learn.org/stable/ (дата звернення 10.03.2024)
27. Vercel Functions
URL: https://vercel.com/docs/functions (дата звернення 10.03.2024)
28. Vercel Documentation
URL: https://vercel.com/docs (дата звернення 10.03.2024)
29. Maps Platform - Google for Developers
URL: https://developers.google.com/maps (дата звернення 10.03.2024)
30. Docker: Accelerated Container Application Development
URL: https://www.docker.com/ (дата звернення 10.03.2024)
31. Client–server model
URL: https://en.wikipedia.org/wiki/Client%E2%80%93server_model (дата звернення 10.03.2024)
32. Swagger: API Documentation & Design Tools for Teams
URL: https://swagger.io/ (дата звернення 10.03.2024)

Додаток А. Посилання на GitHub сховища

Серверна частина - https://github.com/mykhailo-vavr/nicholas-crm-backend

Клієнтська частина - https://github.com/mykhailo-vavr/nicholas-crm-frontend

Безсерверні функції - https://github.com/mykhailo-vavr/nicholas-crm-functions