Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ДОДАТКУ ДЛЯ ОБЛІКУ ВНЕСКІВ ТА ВИТРАТ НА ПРОВЕДЕННЯ КОЛЕКТИВНОГО ЗАХОДУ З ВИКОРИСТАННЯМ NEST.JS

DEVELOPMENT OF AN APPLICATION FOR ACCOUNTING OF CONTRIBUTIONS AND EXPENSES FOR HOLDING A COLLECTIVE EVENT USING NEST.JS

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗс-21
Руднік Дмитро В'ячеславович

Керівник:
Гулієва Наталія Михайлівна

Кваліфікаційну роботу
допущено до захисту
«8» 05 2024 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А. Лінник
«2» 02 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Рудніку Дмитру В'ячеславовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка додатку для обліку внесків та витрат на проведення колективного заходу з використанням Nest.js»
Керівник роботи: к.т.н., доцент Гулієва Н. М.

затверджені наказом закладу вищої освіти від «*30*» *грудня 2023 р.* №477/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи:

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.

5. Перелік графічного матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Гулієва Н.М.*		
Специфікація вимог до розробленої системи	*Гулієва Н.М.*		
Розробка об'єкта проектування	*Гулієва Н.М.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	*7,7%*		
Академічна доброчесність	*Ящук А. А.*		

7. Дата видачі завдання *«2» лютого 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітки
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*до 16.02.2024 р.*	
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*до 01.03.2024 р.*	
3	*Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання*	*до 15.03.2024 р.*	
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*до 29.03.2024 р.*	
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*до 26.04.2024 р.*	
6	*Тестування та налагодження об'єкта проектування*	*до 03.05.2024 р.*	
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*до 5.06.2024 р.*	

Здобувач вищої освіти _____ Ящук Д. В.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Гулієва Н.М.
(підпис) (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія

на кваліфікаційну роботу

Здобувач вищої освіти: Руднік Дмитро В'ячеславович

Тема: «*Розробка додатку для обліку внесків та витрат на проведення колективного заходу з використанням Nest.js»*

Коротка характеристика кваліфікаційної роботи. *Кваліфікаційна робота представляє собою добре структуровану, детально обґрунтовану та професійно виконану роботу, яка орієнтована на розробку додатку для обліку внесків та витрат на проведення колективного заходу з використанням Nest.js.*

Самостійні розробки і пропозиції автора *Розроблений веб-додаток є ефективним інструментом для обліку внесків та витрат на проведення колективних заходів, що має потенціал для подальшого розвитку та впровадження в реальних умовах. Розроблено користувацький інтерфейс, який забезпечує зручний доступ до функціональності додатку через веб-браузер. Забезпечено безпечну аутентифікацію та авторизацію користувачів.*

Недоліки. *Істотних недоліків в роботі не виявлено.*

Загальний висновок. *Робота виконана на високому фаховому рівні, має логічну та послідовну структуру, відповідає встановленим вимогам, може бути представлена до захисту екзаменаційній комісії та заслуговує позитивної оцінки.*

Рецензент: к.т.н., доцент каф КІтаКБ Бакунок Н.В.
(прізвище, ім'я, по-батькові, посада)

Рецензент кваліфікаційної роботи _____
(підпис)

«7» 06 2024 р.

Міністерство освіти і науки України
Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
Здобувач вищої освіти *Руднік Дмитро В'ячеславович*
група *ІПЗс-21*

Тема кваліфікаційної роботи: *Розробка додатку для обліку внесків та витрат на проведення колективного заходу з використанням Nest.js*

Керівник: Гулієва Наталія Михайлівна

Актуальність теми. Розробка веб-додатку для обліку внесків та витрат на проведення колективного заходу з використанням Nest.js є актуальною темою, що відповідає сучасним вимогам до фінансового менеджменту в організації подій. Це дозволить забезпечити зручний, ефективний та безпечний спосіб управління фінансовими ресурсами, що сприятиме успішній реалізації різноманітних заходів.

Об'єкт дослідження – процес організації та управління фінансами при проведенні колективних заходів.

Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи. Розроблений веб-додаток є ефективним інструментом для обліку внесків та витрат на проведення колективних заходів, що має потенціал для подальшого розвитку та впровадження в реальних умовах.

Зауваження та недоліки: істотних недоліків не виявлено.

Загальний висновок робота виконана на високому рівні та заслуговує позитивної оцінки, за умови успішного захисту.

Керівник _____ (Гулієва Н. М.)
(підпис) (прізвище, ім'я, по батькові)

«7» червня 2024 р.

АНОТАЦІЯ

Руднік Д. В. Розробка додатку для обліку внесків та витрат на проведення колективного заходу з використанням Nest.js. Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота бакалавра складається з вступу, трьох розділів, висновків, списку використаних джерел, додатків. У роботі розглядається розробка веб-додатку для обліку внесків та витрат на проведення колективних заходів з використанням фреймворку Nest.js. Проведено аналіз вимог, розроблено архітектуру системи, реалізовано основні модулі та проведено тестування додатку. Отриманий результат забезпечує зручний і безпечний інструмент для управління фінансами під час організації заходів.

Ключові слова: проектування, React, Node.js, Postgresql, клієнт-серверна архітектура додатку.

ABSTRACT

Rudnik D. V. Development of an Application for Accounting of Contributions and Expenses for Holding a Collective Event Using Nest.js. Manuscript.

Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2024.

The bachelor's qualification work consists of an introduction, three sections, conclusions, a list of used sources, and appendices. The paper considers the development of a web application for accounting of contributions and expenses for holding collective events using the Nest.js framework. An analysis of the requirements was carried out, the system architecture was developed, the main modules were implemented and the application was tested. The obtained result provides a convenient and safe tool for managing finances during the organization of events.

Keywords: design, React, Node.js, Postgresql, client-server application architecture.

ЗМІСТ

ВСТУП

У сучасному світі, де організація колективних заходів, таких як конференції, семінари, спортивні події, корпоративні зустрічі, весілля та інші події, стає дедалі популярнішою, виникає необхідність у ефективних засобах управління фінансами цих заходів. Успішна організація таких подій вимагає точного обліку внесків та витрат, що є критичним для забезпечення прозорості, підзвітності та уникнення фінансових втрат.

Традиційні методи обліку, такі як ведення записів у таблицях Excel або на папері, мають низку недоліків. Вони є трудомісткими, схильними до людських помилок, і не забезпечують достатнього рівня безпеки даних. Крім того, відсутність автоматизації ускладнює процес аналізу даних і генерації звітів.

З іншого боку, існують спеціалізовані платформи для організації заходів, але вони часто не мають достатньо гнучких функцій для обліку фінансів, або є занадто дорогими для використання невеликими організаціями чи приватними особами.

Використання сучасних технологій для розробки спеціалізованих веб-додатків може значно покращити процес управління фінансами під час проведення колективних заходів. Одним із перспективних рішень є використання Nest.js – прогресивного фреймворку для Node.js, який дозволяє створювати масштабовані та підтримувані серверні додатки. Nest.js використовує TypeScript, що підвищує надійність та зручність розробки, а також забезпечує високий рівень безпеки та продуктивності.

Таким чином, розробка веб-додатку для обліку внесків та витрат на проведення колективного заходу з використанням Nest.js є актуальною темою, що відповідає сучасним вимогам до фінансового менеджменту в організації подій. Це дозволить забезпечити зручний, ефективний та безпечний спосіб управління фінансовими ресурсами, що сприятиме успішній реалізації різноманітних заходів.

Таким чином, створення заявки на облік внесків та витрат на проведення колективного заходу є актуальним завданням та допоможе зробити організацію спільного заходу простішою та прозорішою.

Метою даної роботи є розробка веб-додатку для обліку внесків та витрат на проведення колективного заходу з використанням фреймворку Nest.js.

Об’єкт дослідження – процес організації та управління фінансами при проведенні колективних заходів.

Предмет дослідження – методи та засоби розробки веб-додатків для обліку внесків та витрат на проведення колективних заходів з використанням фреймворку Nest.js.

Завдання дослідження:

- здійснити аналіз вимог до системи;
- виконати проектування архітектури системи;
- здійснити вибір технологій;
- розробити серверну частину;
- розробити клієнтську частину.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз сучасного стану проблеми

Організація колективних заходів вимагає значних фінансових витрат. У таких випадках необхідно вести облік внесків і витрат. Організатори та кожен учасник заходу повинні знати, скільки грошей потрібно, скільки ще потрібно зібрати, скільки вніс кожен учасник, які витрати були зроблені та скільки грошей залишилося на рахунку. Однак вести такі записи вручну може бути досить складно і довго. Для полегшення цього процесу пропонується створити додаток для обліку внесків і витрат на колективний захід.

Існує багато програм для обліку доходів і витрат, які можна використовувати для організації колективного обліку. Нижче розглянуто три приклади.

Splitwise – чудовий додаток для тих, хто хоче тримати свої витрати під контролем. Це дозволяє користувачам створювати групи та ділитися витратами з іншими членами групи, що, у свою чергу, може допомогти зменшити фінансові витрати кожного учасника. Крім того, кожен учасник може додати свої витрати та вказати, кому вони належать. Ви можете додати всі витрати на їжу, оренду, проїзд тощо, і Splitwise автоматично підрахує, хто скільки винен або кому скільки винен. Крім того, Splitwise має багато додаткових функцій, таких як сповіщення про оплату, можливість додавати кілька валют, а також функції для керування кількома обліковими записами. Використовуючи Splitwise, можна бути впевненим, що фінанси будуть під контролем, і можна уникнути непорозумінь з друзями чи сусідами по квартирі та навіть скоротити свої витрати [1].

Toshl Finance – це програма для фінансового обліку, яка дозволяє користувачам зручно відстежувати свої витрати та доходи. Але це ще не все, що може запропонувати додаток [2].

Додатковою функцією є можливість створювати групи та розподіляти витрати між учасниками. Це особливо корисно, якщо ви плануєте груповий захід або спільну покупку.

Однак можливості програми не обмежуються лише цими функціями. Функція «рахунки» допоможе користувачам визначити, хто кому скільки грошей винен. Це дуже зручно, якщо ви спілкуєтеся з друзями або колегами і ділите витрати.

Крім того, додаток дозволяє створювати багато категорій витрат, щоб можна було точно відстежувати свої витрати.

Таким чином, використання Toshl Finance не тільки допоможе контролювати свої витрати, але й полегшить фінансове планування у житті.

Tricount – це зручна програма для відстеження витрат, яка допомагає не тільки відстежувати свої витрати, але й ділитися ними в групах. Це особливо корисно під час подорожі або організації спільного заходу.

На додаток до функції відстеження витрат, у додатку також є функція «балансу», яка показує, хто скільки винен або кому скільки винен. Це дозволяє уникнути непорозумінь і конфліктів в групі.

Більше того, додаток дозволяє створювати список покупок, що полегшує планування та організацію покупок у групі.

В цілому, Tricount – відмінний помічник в організації спільних заходів і поїздок, що дозволяє легко управляти витратами і уникати непорозумінь в групі.

Кожен з цих додатків може бути використано для організації колективного обліку доходів і витрат.

Слід зазначити, що кожен з цих додатків має свої недоліки:

– Splitwise. Додаток не пропонує можливості відстежувати витрати та доходи в різних валютах, що може бути проблематичним для користувачів, які проживають у різних країнах. Крім того, додаток не дозволяє відстежувати заборгованості між користувачами в режимі реального часу, що може призвести до непорозумінь;

– Toshl Finance. Додаток не завжди точно визначає категорії витрат, що може призвести до неточностей бухгалтерського обліку. Крім того, деякі користувачі можуть вважати його інтерфейс занадто складним і заплутаним;

– Tricount. Додаток не пропонує можливості створювати групи з більш ніж трьома учасниками, що може бути незручно для великих груп. Крім того, деяким користувачам його дизайн може здатися застарілим і незручним.

У таблиці 1.1 наводиться порівняльна характеристика розглянутих аналогів.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій	Splitwise	Toshl Finance	Tricount
Можливість створення груп	+	+	+
Можливість розподіляти витрати між членами групи	+	+	+
Сповіщення про оплату	+	–	–
Кожен учасник може додати витрати	+	+	+
Наявність категорій витрат	–	+	–
Перегляд звітів про заборгованість	+	+	+

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Для успішного виконання кваліфікаційної роботи необхідно вирішити низку завдань, які можна розділити на декілька основних етапів. Кожен етап містить конкретні завдання, спрямовані на досягнення загальної мети роботи:

– здійснити аналіз вимог до системи. Провести дослідження існуючих рішень для обліку внесків; визначити основні функціональні та нефункціональні вимоги до додатку; скласти список вимог до системи;

– виконати проектування архітектури системи. Розробити загальну архітектуру додатку, включаючи клієнтську і серверну частини; визначити структуру бази даних, необхідні таблиці та взаємозв’язки між ними; розробити модулі системи (користувачі, заходи, внески, витрати, звіти) та їх взаємодію;

– здійснити вибір технологій. Обґрунтувати вибір Nest.js як основного фреймворку для розробки серверної частини додатку; вибрати базу даних; вибрати додаткові інструменти для тестування;

– розробити серверну частину. Реалізувати модуль користувачів: реєстрація, аутентифікація, авторизація; реалізувати модуль заходів: створення, редагування, видалення заходів, управління учасниками та ролями; реалізувати модуль внесків та витрат: додавання, редагування, видалення внесків та витрат, підрахунок загальних сум та часток;

– розробити клієнтську частину. Розробити користувацький інтерфейс для взаємодії з системою: форми для введення даних, таблиці для відображення внесків та витрат, графіки та діаграми для звітів; забезпечити зручну навігацію та інтуїтивно зрозумілий дизайн інтерфейсу.

Виконання цих завдань дозволить створити повноцінний веб-додаток для обліку внесків та витрат на проведення колективного заходу з використанням Nest.js, що забезпечить зручний, ефективний та надійний спосіб управління фінансами для організаторів подій.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Метою кваліфікаційної роботи є розробка програми для обліку фінансів колективного заходу. Розроблена система повинна спростити відстеження та облік витрат і внесків на заході.

Система повинна відповідати таким вимогам:

– правильний облік фінансів;
– можливість створення подій;
– можливість додавати співробітників на заходи;
– можливість створення звітів по співробітникам і подіям;
– створення загального складу, де будуть зберігатися речі.

Система повинна виконувати такі завдання: забезпечити, щоб внески та витрати обліковувалися зручним для користувача способом.

Розглянемо на прикладі, як повинен працювати додаток.

Припустимо, колектив вирішив влаштувати пікнік на честь закінчення робочого року. Вони вибирали місце і призначали відповідальну особу. У розроблюваному додатку відповідальна особа створила захід і додала до нього співробітників. Після цього склав список необхідних речей, приблизні витрати на все. Додаток підрахував, скільки рублів потрібно скинутися на захід. Для кожного учасника заходу ця сума відображається на його балансі. Якщо учасник купує ще щось для пікніка, він може додати це до загального списку. Також у додатку є склад. Там зберігаються речі, які залишилися від минулих подій, або речі, які не закінчилися. Наприклад, намети, розкладні стільці, стіл.

Це допомагає побачити, що вже є в організації.

Завданнями цієї системи є:

– автоматизація процесу фінансового обліку під час колективного заходу;

– створення гнучких фінансових звітів про колективні заходи.

Функціонал системи:

– відстеження внеску кожного учасника. Щоб забезпечити належне фінансове управління заходом, важливо відстежувати внески, зроблені кожним учасником. Цього можна досягти, розробивши чітку та організовану систему відстеження платежів та забезпечивши обізнаність усіх учасників про статус своїх внесків;

– відстеження витрат на події. Відстеження витрат на події є важливим аспектом організації будь-якого заходу. Необхідно враховувати витрати на оренду приміщення, закупівлю необхідного обладнання та матеріалів, закупівлю їжі та напоїв для фуршету або вечері, а також оплату послуг професійних підрядників при необхідності. Важливо заздалегідь спланувати бюджет заходу, щоб уникнути непередбачених витрат і забезпечити успішну реалізацію;

– створення балансу по кожному співробітнику. Кожен співробітник повинен знати, скільки і де він повинен платити, він повинен знати про витрати на кожен захід, а також загальну суму за всі минулі і заплановані заходи. Якщо працівник залишився в боргу за подію або якщо він заплатив більше, ніж повинен, то він також повинен це побачити;

– створення звіту про витрати на захід. Звіт про витрати на захід допоможе в цілому побачити, скільки грошей знадобилося для заходу, які категорії витрат були. Також має бути загальний звіт за всіма видами діяльності;

– створення складу. У цій таблиці зберігаються речі, які вже є в команді, і які можна використовувати для організації заходів.

Дизайн додатків – це процес створення архітектури та дизайну програми, яка забезпечує її функціональність, надійність та зручність використання. Він включає розробку бізнес-логіки, вибір технологій, дизайн інтерфейсу користувача та багато іншого.

Функціональна модель IDEF0. IDEF0 – це методологія для створення функціональних моделей і графічна нотація, розроблена для формалізації та опису бізнес-процесів. Особливістю IDEF0 є акцент на ієрархії об’єктів. Вона

аналізує логічні зв’язки між завданнями, не зосереджуючись на часовій послідовності виконання робіт.

Функціональна модель IDEF0 складається з набору блоків, кожен з яких виступає у ролі «чорного ящика» з вхідними і вихідними даними, елементами управління та механізмами, деталізованими до потрібного рівня. Функції з’єднуються між собою стрілками та описами функціональних блоків. При цьому кожен вид стрілки або діяльності має своє значення. Дана модель дозволяє описати всі основні типи процесів, як адміністративних, так і організаційних.

Папки «Вхідні» – це вхідні дані, які встановлюють певне завдання.

Вихідний – висновок результату діяльності.

Control (зверху-вниз) – механізми управління (регламенти, інструкції тощо).

Механізми (від низу до верху) – те, що використовується для виконання необхідної роботи.

На рисунку 2.1 представлена функціональна модель застосування для обліку внесків і витрат колективного заходу.

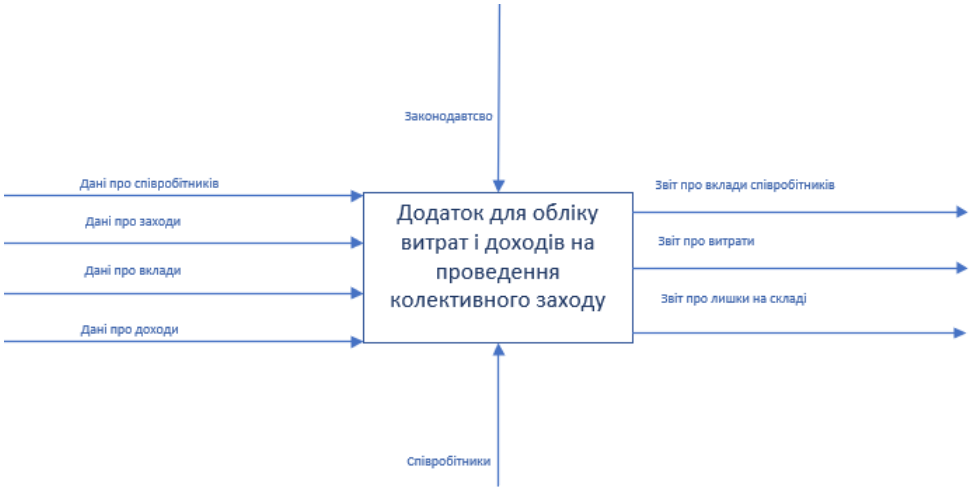

Рисунок 2.1 – Функціональна модель IDEF0

З рисунка видно, що в бізнес-процесі беруть участь співробітники. На вхід надходять такі дані: дані про співробітників, дані про події, дані про внески, дані

про витрати. В результаті отримуємо звіт про внески співробітників, звіт про витрати заходу, звіт про залишки на складі. Кожен співробітник зможе побачити, на який захід і скільки грошей він здав або потрібно здати. Звіт про захід включатиме дані про те, які категорії витрат потрібні та сума за кожну, а також кількість людей, які будуть присутні на заході. Звіт про залишки на складі міститиме інформацію про те, які товари залишилися від минулих подій.

Діаграма варіантів використання – це діаграма, яка описує, який функціонал розроблюваної програмної системи доступний кожній групі користувачів.

У розроблюваному додатку одна група користувачів – це співробітники. Системні функції, створені для цієї групи, показані на рисунку 2.2.

Рисунок 2.2 – Діаграма варіантів використання

Співробітники можуть користуватися всіма можливостями програми: додавати співробітників, створювати події, підключати співробітників до подій,

підключатися до складу, переглядати залишки на складі, додавати інформацію про внесені та витрачені на захід гроші, переглядати заборгованості.

Розглянемо карту інтерфейсу. Структура програми показана на рисунку 2.3.

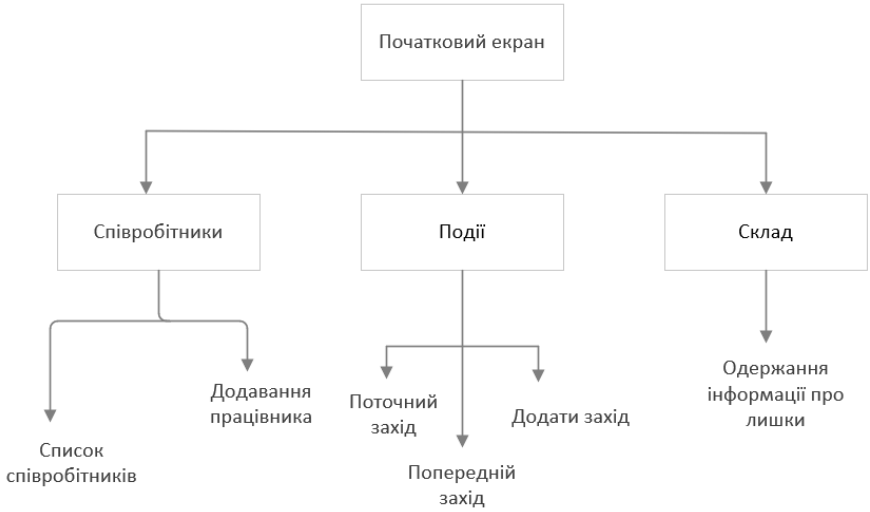

Рисунок 2.3 – Структура додатку

Прямокутниками позначені сторінки інтерфейсу, стрілками – переходи з однієї сторінки на іншу.

На головному екрані можна відкрити ще три екрани: «Співробітники», «Події», «Склад».

На екрані «Співробітники» користувач може переглянути список співробітників і додати нового співробітника в додаток.

На екрані «Події» користувач може переглянути вже створені події, названі в додатку як актуальні, переглянути історію минулих подій або створити нову подію, заповнивши відповідну форму.

Через екран «Склад» користувач може отримати інформацію про складські залишки. Це може бути як загальний звіт, так і інформація про конкретну подію.

У розроблюваному додатку було прийнято рішення використовувати клієнт-серверну архітектуру.

Клієнт-серверна архітектура додатків – це підхід до розробки додатків, при якому додаток ділиться на дві основні частини – клієнтську і серверну. Клієнтська частина – це інтерфейс користувача, який взаємодіє з користувачем, а бекенд обробляє запити клієнта та повертає результати.

Фронтенд програми – це інтерфейс користувача, який взаємодіє з користувачем і відображає дані, отримані з бекенда. Клієнтська частина може бути написана на різних мовах програмування і може працювати на різних платформах і пристроях, таких як веб-браузери, мобільні пристрої та десктопні додатки. На стороні клієнта можна використовувати різні технології, такі як HTML, CSS, JavaScript, React, Angular и інші.

Бекенд додатку – це програма, яка обробляє запити з боку клієнта, виконує необхідні операції та повертає результати. Бекенд може бути написаний на різних мовах програмування, таких як Node.js, Python, Ruby та інших. Бекенд може використовувати різноманітні технології, такі як бази даних, API, веб-сервери та інші інструменти для обробки запитів і повернення даних на фронтенд.

Клієнт-серверна архітектура програми дозволяє розробникам легко масштабувати та розширювати програму, оскільки клієнт та сервер можуть бути розроблені та розгорнуті незалежно один від одного. Фронтенд може бути розгорнутий на різних платформах і пристроях, а бекенд може бути розгорнутий на різних серверах в залежності від потреб програми.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Щоб розробити додаток, потрібно вибрати інструменти для написання клієнтської частини програми, серверної частини програми та бази даних.

Для початку розглянемо інструменти розробки баз даних.

MongoDB – це документоорієнтована база даних, яка дозволяє зберігати дані у вигляді документів BSON (Binary JSON). Вона відрізняється від реляційних баз даних тим, що не вимагає жорсткої схеми даних і може зберігати дані різних типів і структур. MongoDB також має високу масштабованість та продуктивність завдяки своїй здатності масштабувати та індексувати дані. Однак, MongoDB має обмежену підтримку транзакцій і вимагає певних знань і навичок для управління та налаштування [3].

Плюси використання MongoDB:

– гнучкість. MongoDB дозволяє зберігати дані в різних форматах і структурах, що робить їх дуже гнучкими та адаптованими до різних завдань;

– масштабованість. MongoDB легко масштабується по горизонталі, що дозволяє їй обробляти великі обсяги даних і високі навантаження;

– висока продуктивність. MongoDB забезпечує швидкий доступ до даних і швидкий запис і читання;

– проста у використанні мова запитів: MongoDB використовує просту та інтуїтивно зрозумілу мову запитів, що полегшує роботу з даними.

Мінуси використання MongoDB:

– обмежена підтримка транзакцій. MongoDB підтримує лише обмежений набір транзакційних операцій, що може бути проблемою для додатків, які вимагають високої узгодженості даних;

– складність адміністрування. MongoDB вимагає певних знань та навичок для управління та налаштування бази даних;

– обмежений набір інструментів для аналізу даних. MongoDB не надає такого широкого набору інструментів аналізу даних, як реляційні бази даних.

MySQL – це реляційна база даних, яка підтримує багато Функції та можливості для зберігання та керування даними. Він використовується для багатьох типів додатків, включаючи веб-додатки, системи управління контентом і так далі. MySQL є однією з найпопулярніших баз даних у світі з відкритим вихідним кодом [4].

Плюси використання MySQL:

– простота використання. MySQL надає просту та інтуїтивно зрозумілу мову запитів, що полегшує роботу з даними;

– висока продуктивність. MySQL забезпечує швидкий доступ до даних і швидкий запис і читання;

– широкий спектр інструментів для аналізу даних. MySQL надає різноманітні інструменти для аналізу та обробки даних.

Висока підтримка транзакцій: MySQL забезпечує високу узгодженість даних під час транзакцій.

Мінуси використання MySQL:

– обмежена масштабованість. MySQL може зіткнутися з проблемами масштабування при обробці великих обсягів даних;

– жорстка схема даних. MySQL вимагає суворої схеми даних, що може ускладнити роботу з даними;

– обмежені можливості зберігання даних. MySQL має обмеження на обсяг даних, які можуть зберігатися в одній таблиці.

PostgreSQL – це об’єктно-реляційна система управління базами даних (СУБД), яка надає широкий спектр функцій і можливостей для зберігання і управління даними. Він використовується для багатьох типів додатків, включаючи веб-додатки, системи управління контентом і так далі. PostgreSQL є однією з найпопулярніших баз даних у світі з відкритим вихідним кодом [5].

Плюси використання PostgreSQL:

– гнукість. PostgreSQL дозволяє зберігати дані в різних форматах і структурах, що робить їх дуже гнучкими та адаптованими до різних завдань;

– масштабованість. PostgreSQL легко масштабується по горизонталі та вертикалі, що дозволяє їй обробляти великі обсяги даних і високі навантаження;

– висока продуктивність. PostgreSQL забезпечує швидкий доступ до даних і швидкий запис і читання;

– високий рівень транзакційної підтримки. PostgreSQL забезпечує високу узгодженість даних під час транзакцій і підтримує багато рівнів ізоляції транзакцій;

– широкий спектр інструментів для аналізу даних. PostgreSQL надає різноманітні інструменти для аналізу та обробки даних;

– велика спільнота розробників. PostgreSQL має велику спільноту розробників і користувачів, що забезпечує її постійний розвиток і підтримку.

Мінуси використання PostgreSQL:

– складність адміністрування. PostgreSQL вимагає певних знань та навичок для управління та налаштування бази даних;

– обмежені можливості зберігання даних. PostgreSQL має обмеження на обсяг даних, які можуть зберігатися в одній таблиці;

– обмежена масштабованість. PostgreSQL може зіткнутися з проблемами масштабування під час обробки великих обсягів даних.

У таблиці 2.1 наведені порівняльні характеристики і поняття PostgreSQL і MongoDB.

Таблиця 2.1 – Порівняння баз даних SQL та NoSQL

Ознака	Бази даних SQL	NoSQL бази даних
Типи та структури даних	Інформація зберігається у взаємопов'язаних таблицях. Таблиця має набір стовпців, кожен з яких відповідає певному типу даних.	Дані зберігаються в документах, колекціях або графіках. Документ – це структурований контейнер для зберігання даних у форматі пар ключ-значення, де пари можуть мати різні типи даних. Колекція – це група документів, які пов’язані між собою. Граф – це множина вершин і зв’язків між ними.
Методи зберігання даних та продуктивність	Дані в таблицях структуровані за суворими правилами і можуть бути пов'язані за допомогою зовнішніх ключів.	Немає строгих правил організації даних, що робить більш гнучким зберігання різних типів даних і структур

Продовження таблиці 2.1

Ознака	Бази даних SQL	NoSQL бази даних
Надійність і стабільність	Транзакційна модель використовується для збереження цілісності даних і забезпечення властивостей ACID для відмовостійкості та надійності	Для підвищення надійності та відмовостійкості використовується розподілена архітектура
Гнучкість і масштабованість	Більш обмежені в гнучкості структури даних	Легко масштабуйтеся по горизонталі, додаючи нові сервери та розподіляючи навантаження між ними

Проаналізувавши засоби розробки баз даних, було прийнято рішення використовувати в цій роботі PostgreSQL.

Сучасна фронтенд-розробка – це складний процес, який включає в себе безліч технологій та інструментів. Вона охоплює не тільки створення дизайну і верстки сайту, але і його функціональне наповнення і оптимізацію.

Основними технологіями, які використовуються у front-end розробці, є HTML, CSS та JavaScript. Вони дозволяють створювати різні елементи інтерфейсу, визначати їх стилі, забезпечувати інтерактивність на сторінці.

Однак одного використання цих технологій недостатньо для створення сучасного додатку. Саме тут на допомогу приходять фреймворки та бібліотеки, такі як React, Vue.js та Angular. Вони дозволяють прискорити процес розробки, спростити створення складних інтерфейсів, забезпечити більш високу продуктивність.

Далі ми розглянемо інструменти розробки програми на стороні клієнта.

Angular – це фреймворк для розробки веб-додатків, створений Google і використовується багатьма розробниками по всьому світу. Він дозволяє швидко та ефективно створювати складні та масштабовані додатки завдяки своїй компонентній архітектурі та декларативному підходу до побудови інтерфейсу користувача.

Angular також надає різноманітні інструменти та бібліотеки для спрощення процесу розробки, такі як маршрутизація, форми, перевірка та тестування. Ці інструменти дозволяють створювати програми швидше та безпечніше.

Крім того, Angular має широку підтримку спільноти, що полегшує пошук рішень і допомогу при виникненні проблем. Це дозволяє розробникам швидше вирішувати проблеми та прискорювати процес розробки [6].

Плюси використання Angular:

– потужна функціональність. Angular надає широкий спектр інструментів і функцій для створення складних веб-додатків;

– висока продуктивність. Angular забезпечує швидкий доступ до даних і швидкий запис і читання;

– широкий спектр інструментів аналізу даних. Angular надає різноманітні інструменти аналізу та обробки даних;

– багатоплатформна підтримка. Angular можна використовувати для створення веб-додатків на різних платформах, включаючи мобільні та настільні комп’ютери;

– високий ступінь модульності. Angular дозволяє створювати та використовувати модулі, що робить його дуже гнучким та адаптивним до різних завдань.

Мінуси використання:

– високий поріг входу. Angular вимагає певних знань і навичок для його використання, що може ускладнити роботу з ним;

– обмежена масштабованість. Angular може зіткнутися з проблемами масштабування під час обробки великих обсягів даних.

Vue.js – це прогресивний фреймворк інтерфейсу користувача, який дозволяє створювати динамічні односторінкові програми (SPA). Цей фреймворк надає можливість повторного використання компонентів інтерфейсу, що спрощує розробку та економить час. Багато розробників вибирають Vue.js через його легку вагу і простоту використання. Крім того, Vue.js має широкий спектр опцій для масштабування проєктів та зручні інструменти для тестування коду. Загалом,

використання Vue.js дозволяє створювати якісні та професійні інтерфейси для користувачів [7].

Плюси використання Vue.js:

– легкий. Vue.js – це легкий і швидкий фреймворк, який дозволяє створювати швидкі та адаптивні веб-додатки;

– простота використання. Vue.js має простий та інтуїтивно зрозумілий синтаксис, що робить його доступним для розробників-початківців;

– гнучкість. Vue.js дозволяє гнучко налаштовувати та розширювати його функціональність за допомогою плагінів та додаткових бібліотек;

– швидка швидкість розробки. Vue.js спрощує процес розробки та дозволяє швидко створювати компоненти та інтерфейси;

– хороша підтримка документації. Vue.js має якісну документацію та велику спільноту розробників, що забезпечує її постійний розвиток та підтримку.

Мінуси використання Vue.js:

– обмежена масштабованість. Vue.js може зіткнутися з проблемами масштабування під час обробки великих обсягів даних;

– обмежена функціональність. Vue.js може не підійти для створення дуже складних додатків, оскільки він не має такої великої функціональності, як Angular.

React – це бібліотека інтерфейсу користувача, яка дозволяє створювати динамічні односторінкові додатки (SPA) та багатокомпонентні інтерфейси. React забезпечує швидкий доступ до даних і швидкий запис і читання. React також має високий ступінь модульності, що дозволяє створювати та використовувати компоненти, що робить його дуже гнучким та адаптованим до різних завдань.

Плюси використання React:

– висока продуктивність. React забезпечує швидкий доступ до даних і швидкий запис і читання;

– гнучкість. React дозволяє гнучко налаштовувати та розширювати його функціональність за допомогою плагінів та додаткових бібліотек;

– високий ступінь модульності. React дозволяє створювати та використовувати компоненти, що робить його дуже гнучким та адаптованим до різних завдань;

– широкий спектр інструментів для аналізу даних. React надає різноманітні інструменти для аналізу та обробки даних;

– багатоплатформна підтримка. React можна використовувати для створення веб-додатків на різних платформах, включаючи мобільні та десктопні.

Мінуси використання React:

– потрібно використовувати JSX. React використовує синтаксис JSX, який може бути незнайомим деяким розробникам;

– необхідність використання інструментів збірки. React вимагає використання інструментів збірки, таких як Webpack або Gulp;

– необхідність використання Redux. React часто використовує бібліотеку Redux для керування станом програми, що може ускладнити код програми.

Розглянувши вищеописані варіанти розробки клієнтської частини програми, було прийнято рішення використовувати React.

Сучасна бекенд-розробка включає в себе безліч технологій та інструментів, які дозволяють створювати високопродуктивні та масштабовані додатки. Деякі з найбільш популярних технологій включають: Node.js, Express.js, Nest.js. Однак одних технологій недостатньо для створення якісного бекенд-додатку. Також важливо враховувати вимоги до безпеки, масштабованості і продуктивності.

Node.js – це платформа для створення високопродуктивних JavaScript-додатки, які працюють на стороні сервера. Node.js використовує єдину потокову модель для обробки запитів, що робить її дуже масштабованою та швидкою. Node.js також надає широкий спектр інструментів і бібліотек для створення веб-додатків і API [8-9].

Плюси використання Node.js:

– висока продуктивність. Завдяки асинхронній моделі потокового передавання Node.js забезпечує високу продуктивність і швидкий доступ до даних;

– масштабованість. Node.js дозволяє масштабувати ваші програми в міру зростання навантаження;

– уніфікована модель потокового передавання. Node.js використовує єдину модель потокового передавання для обробки запитів, що робить її дуже масштабованою та швидкою;

– широкий спектр інструментів і бібліотек. Node.js надає широкий спектр інструментів і бібліотек для створення веб-додатків і API;

– низькі витрати на розробку. Node.js спрощує процес розробки та дозволяє швидко створювати веб-додатки та API.

Мінуси використання Node.js:

– обмежена підтримка старих веб-переглядачів. Node.js не підтримує старіші версії веб-переглядачів, що може бути проблемою для деяких користувачів;

– необхідність використання модулів для деяких функцій. Node.js не містить деяких функцій, які можуть знадобитися для створення програми, що вимагає використання модулів.

Express.js – це фреймворк Node.js, який спрощує створення веб-додатків та API. Express.js надає різноманітні інструменти та функції для створення швидких і масштабованих програм [10].

Плюси використання Express.js:

– швидка розробка додатків. Express.js спрощує процес розробки та дозволяє швидко створювати веб-додатки та API;

– гнучкість. Express.js дозволяє гнучко налаштовувати та розширювати його функціональність за допомогою плагінів та додаткових бібліотек;

– низькі витрати на розробку. Express.js спрощує процес розробки та дозволяє швидко створювати веб-додатки та API;

– широкий спектр інструментів і бібліотек. Express.js надає широкий спектр інструментів і бібліотек для створення веб-додатків і API.

Мінуси використання Express.js:

– необхідність використання інструментів збірки. Express.js вимагає використання інструментів збірки, таких як Webpack або Gulp;

– потреба в проміжному програмному забезпеченні. Деякі функції, такі як обробка даних форм або аутентифікація, вимагають використання проміжного програмного забезпечення, що може ускладнити код програми;

– необхідність використання сторонніх бібліотек. Деякі функції, такі як робота з базами даних, вимагають використання сторонніх бібліотек, що може ускладнити код програми.

Nest.js – це фреймворк Node.js, який дозволяє легко створювати масштабовані та легко тестовані веб-додатки. Nest.js використовує TypeScript і надає різноманітні інструменти та функції для створення швидких і масштабованих програм.

Плюси використання Nest.js:

– TypeScript. Nest.js використовує TypeScript, що полегшує розробку та покращує якість коду [11];

– масштабованість. Nest.js дозволяє створювати масштабовані додатки;

– простота тестування. Nest.js полегшує тестування вашого додатку за допомогою TypeScript та Dependency Injection;

– гнучкість. Nest.js дозволяє гнучко налаштовувати та розширювати його функціональність за допомогою плагінів та додаткових бібліотек;

– низькі витрати на розробку. Nest.js спрощує процес розробки та дозволяє швидко створювати веб-додатки та API.

Мінуси використання Nest.js [12]:

– необхідність використання TypeScript. Для використання Nest.js необхідно знати TypeScript, що може бути проблемою для деяких розробників;

– складність. Nest.js може бути складним завданням для нових користувачів через його широкі функції та інструменти.

Проаналізувавши описані вище інструменти, було прийнято рішення використовувати Nest.js в роботі.

РОЗДІЛ 3
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

3.1 Проектування бази даних

Для того, щоб розробити додаток для обліку внесків та витрат на колективний захід, було створено базу даних, наведену на рисунку 3.1.

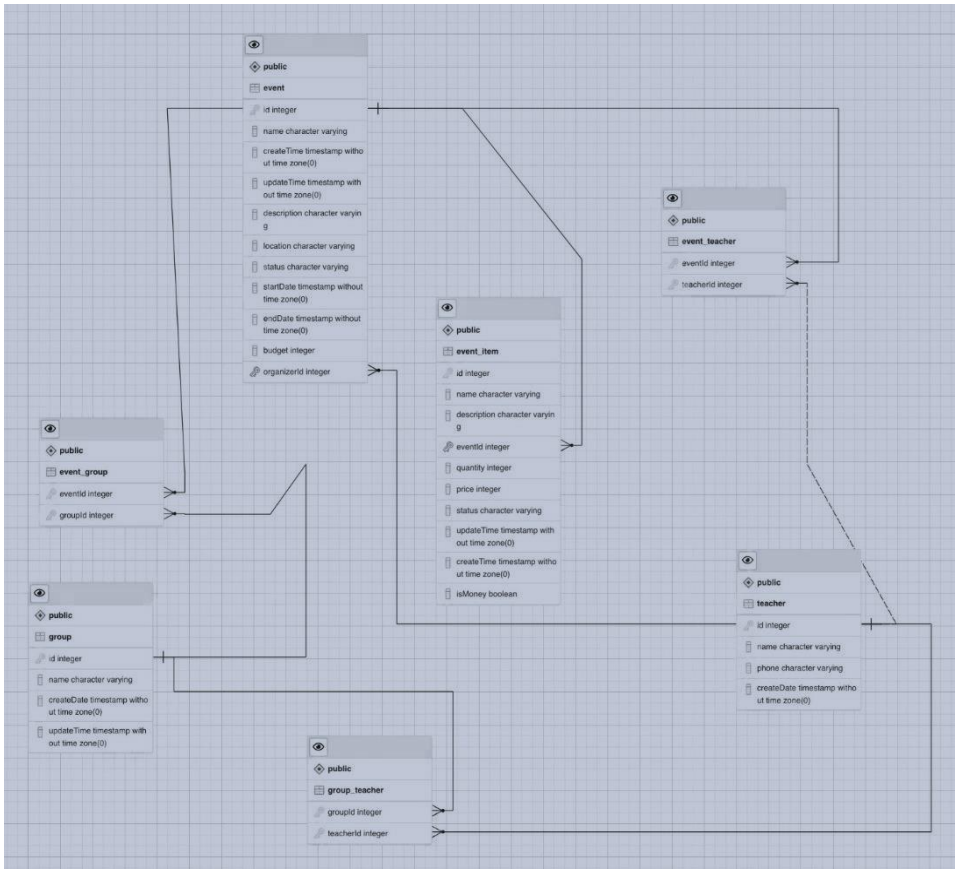

Рисунок 3.1 – База даних програми, що розробляється

База даних містить 4 основні таблиці: «Працівник», «Група», «Подія», «Склад (позиція для заходу)». Крім того, в базі даних є 3 додаткові таблиці, які з’єднують основні.

Таблиця «Подія» містить такі поля: назва події (текстовий тип даних), опис (тип текстових даних), місце події (тип текстових даних), статус (тип текстових даних), час і дата початку та закінчення події, бюджет (числовий тип даних). У полі «Бюджет» зберігається сума, необхідна для проведення події. Ця сума

розподіляється за кількістю учасників, так з’ясовується, скільки кожен співробітник повинен заплатити за цю подію.

Таблиця «Предмети для заходу» – це склад з усіма речами, які потрібні для поточних подій, а також предметами, які вже є в команді. Ця таблиця містить такі поля: ім’я (текстовий тип даних), опис (текстовий тип даних), ціна (числовий тип даних), кількість (числовий тип даних), статус (текстовий тип даних). Також є поле «гроші». Через це поле можна відзначити грошовий ресурс.

У таблиці «Працівник» створені такі поля: ім’я (текстовий тип даних), номер телефону (текстовий тип даних).

Таблиця «Група» має лише одне поле – ім’я (тип текстових даних).

Також є таблиці для зв’язування основних таблиць – групи та працівника, працівника та події, події та групи. Таблиці пов’язані за допомогою зв’язку «багато-до-багатьох».

Зв’язок «багато-до-багатьох» – це тип зв’язку, при якому кожен запис з однієї таблиці може бути пов’язаний з кількома записами з іншої таблиці, і навпаки. Це означає, що один запис може відповідати декільком записам з іншої таблиці, і навпаки – один запис з іншої таблиці може відповідати декільком записам з першої таблиці.

Для реалізації зв’язку «багато-до-багатьох» у базі даних зазвичай використовується проміжна таблиця, яка містить ключі обох таблиць, пов’язаних одна з одною. Ця таблиця дає змогу пов’язати кілька записів з однієї таблиці з кількома записами з іншої таблиці.

Таблиці «Події» та «Елементи подій» зв’язуються за допомогою зв’язку «один-до-багатьох».

Зв’язок «один-до-багатьох» – це тип зв’язку, при якому один запис в одній таблиці може бути пов’язаний із кількома записами в іншій таблиці, але кожен запис у другій таблиці може бути пов’язаний лише з одним записом у першій таблиці.

3.2 Серверна частина

Для розробки бекенду була використана чиста архітектура. Clean Architecture – це концепція розробки програмного забезпечення, яка пропонує підхід до організації коду та архітектури, щоб зробити додаток більш зрозумілим, гнучким і легко модифікованим.

Основна ідея чистої архітектури полягає в тому, що різні компоненти програми повинні бути відокремлені та ізольовані один від одного. Це дозволяє легко змінювати один компонент, не впливаючи на інші. У той же час чиста архітектура визначає правила, яких необхідно дотримуватися, щоб досягти такої ізоляції. Зокрема, це стосується того, як компоненти залежать один від одного, які дані передаються між ними і як вони обробляються.

У додатку, що розробляється, створено два рівні: домен та інфраструктура.

Доменний рівень – це центральний рівень програми, який містить бізнес-логіку та бізнес-правила. Цей рівень не залежить від інших прикладних рівнів і не залежить від фреймворків або бібліотек. Доменний рівень повинен бути написаний мовою, яка найбільш точно відображає бізнес-модель і бізнес-правила, які реалізує додаток. Цей рівень має бути легким для тестування, оскільки саме в ньому знаходиться основна логіка програми.

Рівень інфраструктури – це рівень, який дозволяє додатку взаємодіяти із зовнішнім світом. Сюди входять бази даних, мережеві служби, файлові системи та інші зовнішні компоненти. Інфраструктурний рівень містить адаптери, які пов’язують доменний рівень із зовнішнім світом. Адаптери – це класи, які дозволяють взаємодіяти із зовнішніми компонентами, але приховують деталі реалізації з доменного рівня.

Було створено три складові: захід, працівник та група. Кожен компонент має модель і контролер. Вони відокремлені один від одного, що дозволяє легко змінювати один компонент, не впливаючи на інший.

Модель – це компонент, який містить бізнес-логіку та бізнес-правила програми. Це може бути набір класів, структур даних, сервісів або інших компонентів, які відповідають за обробку даних і виконання бізнес-операцій. Модель не залежить від призначеного для користувача інтерфейсу або інших компонентів програми, що робить її легко портативною та тестованою.

Контролер – це компонент, який обробляє введені користувачем дані та керує взаємодією між інтерфейсом користувача та моделлю. Контролер може бути реалізований у вигляді класу, який приймає запити від UI, викликає відповідні методи моделі та повертає результати назад в UI. Контролер також може відповідати за обробку помилок і управління станом програми.

На рисунку 3.2 представлена модель сутності події.

Рисунок 3.2 – Модель сутності події

На лістингу 3.1 показано контролер сутності події.

Лістинг 3.1 – Контролер сутності події

```
import {Body, Controller, Delete, Get, Param, Post, Query} from '@nestjs/common';
import {ApiBody, ApiOkResponse, ApiQuery, ApiTags} from '@nestjs/swagger';
import {EventService} from '.. /.. /domain/services/EventService';
import {EventSearchDto} from '.. /.. /domain/dtos/EventSearchDto';
import {EventSchema} from '.. /schemas/EventSchema';
```

```
import {EventSaveDto} from '.. /.. /domain/dtos/EventSaveDto';
@Controller('/ event')
@ApiTags() export class EventController
 { constructor( private service:
 EventService,
    ) {
    }
    @Get()
    @ApiQuery({type: EventSearchDto})
    @ApiOkResponse({type: EventSchema, isArray: true})
 async get(@Query() dto: EventSearchDto) { return
 this.service.search(dto);
    }
    @Get('/:id')
    @ApiOkResponse({type: EventSchema, isArray: false})
 async getOne(@Param('id') id: number) { return
 this.service.findById(id);
    }
    @Post()
    @ApiBody({type: EventSaveDto}) @ApiOkResponse({type:
 EventSchema, isArray: false}) async create(
        @Body() dto: EventSaveDto,
    ) {
        return this.service.create(dto);
    }
    @Post('/:id')
    @ApiBody({type: EventSaveDto}) @ApiOkResponse({тип:
 EventSchema, isArray: false}) async update(
        @Body() dto: EventSaveDto,
        @Param('id') id: number,
    ) {
        return this.service.update(id, dto);
    }
    @Delete('/:id') async
 delete(@Param('id') id: number) { return
 this.service.remove(id);
          }
      }
```

Кінець лістингу 3.1

Контролер використовував запити для обробки введених користувачем даних і виконання бізнес-логіки. HTTP-запити для веб-додатку, як правило, поділяються на два типи: GET і POST.

GET-запити – це запити на отримання даних із сервера. Контролер може використовувати GET-запити для відображення даних на сторінці або для

отримання даних з бази даних чи інших джерел. GET-запити можуть містити параметри, які передаються контролеру для виконання певних операцій.

POST-запити – це запити на відправку даних на сервер. Контролер може використовувати POST-запити для створення, оновлення або видалення даних на сервері. POST-запити можуть містити дані форми або інші дані, які передаються контролеру для обробки.

У контролерах веб-додатків GET-запити зазвичай обробляються методами, які повертають представлення для відображення даних на сторінці. Наприклад, метод Index() контролера може бути використаний для відображення списку елементів у базі даних.

POST-запити в контролерах зазвичай обробляються методами, які змінюють дані на сервері. Наприклад, метод Create() контролера може бути використаний для створення нового елементу в базі даних на основі даних, надісланих користувачем з форми.

Сутності «подія», «працівник», «група» програмуються за допомогою методології впровадження залежностей.

Dependency Injection (DI) – це методологія програмування, яка дозволяє керувати залежностями між компонентами програми. Він надає спосіб створювати об’єкти, які залежать від інших об’єктів, без явного створення цих об’єктів всередині класу.

Суть DI полягає в тому, що залежності компонента передаються йому ззовні, наприклад, через конструктор або властивості. Це дозволяє створювати компоненти, які можна використовувати в різних ситуаціях і на різних платформах.

У Nest.js році Dependency Injection надає механізм для створення та вставки залежностей у компоненти. Компонентами можуть бути класи, служби, провайдери або контролери.

Залежності в Nest.js визначаються за допомогою провайдерів. Провайдери – це класи, які надають об’єкти або функції, які можуть бути вставлені в інші компоненти. Наприклад, постачальник може надати об’єкт бази даних або функцію для обробки файлів.

Щоб вставити залежність компонент, потрібно визначити провайдера для цієї залежності та вказати його в конструкторі або властивості компонента.

На рисунках 3.3 і 3.4 показаний основний модуль Dependency Injection програми.

Рисунок 3.3 – Основний модуль DI додатку

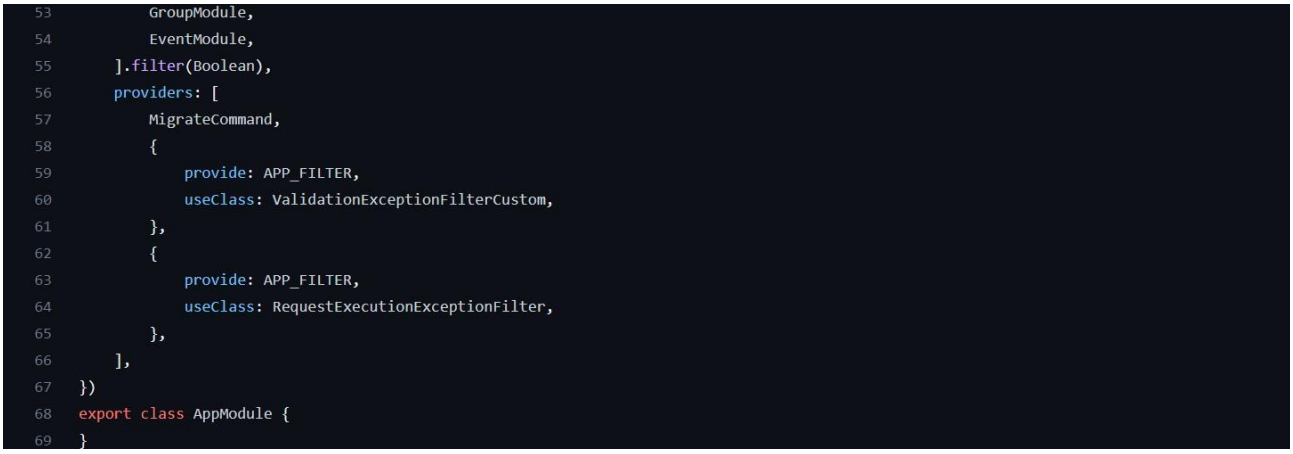

Рисунок 3.4 – Основний модуль DI додатку (продовження)

3.3 Клієнтська частина

Клієнтська частина написана за допомогою бібліотеки React. При розробці додатків на React використовуються компоненти, які можуть бути повторно використані і зібрані в ієрархічну структуру.

Архітектура React-додатку зазвичай будується поверх Flux або Redux. Ці архітектурні патерни забезпечують механізми для керування станом програми та зв’язком між компонентами.

Крім того, архітектура React може використовувати компоненти вищого порядку (HOC), контексти та хуки. КВП – це функції, які приймають компонент і повертають новий компонент з додатковими властивостями або функціональністю. Контекст дозволяє передавати дані між компонентами без необхідності пропускати їх через усі проміжні компоненти. Хуки надають механізми для додавання стану та функціональності функціональним компонентам.

При розробці клієнтської частини використовувалася бібліотека Tanstack Query. Tanstack Query – бібліотека для роботи із запитами даних на клієнті та сервері. У ньому використовується концепція декларативних запитів, що означає, що розробники можуть описати, які дані їм потрібні, а не як їх отримати.

Ось як працює Tanstack Query:

– описує, як запитувати дані за допомогою хука useQuery або інших хуків, які надає бібліотека;

– коли компонент, що містить хук useQuery, рендериться, Tanstack Query автоматично надсилає запит на сервер, використовуючи вказану URL-адресу та параметри;

– під час виконання запиту компонент може показувати індикатор завантаження або інший елемент інтерфейсу, який повідомляє користувачеві про те, що завантажуються дані;

– коли запит завершується, Tanstack Query зберігає отримані дані в кеші та викликає функцію зворотного виклику, яку розробник надав при визначенні запиту;

– якщо дані зміняться, Tanstack Query автоматично перевиконає запит і оновить кеш.

Tanstack Query також підтримує безліч інших функцій, таких як оптимістичні оновлення, скасування запитів, кешування тощо. Є можливість створювати події, групи, додавати працівників.

Клієнтська частина дозволяє додавати та зберігати групи працівників та робити внески до кожного члена групи. Усі внески розраховуються за загальною сумою на групу. Ці групи можуть бути прикріплені до подій, а окремі користувачі можуть бути додані до подій. В результаті роботи програма підсумує суму коштів, отриманих від груп і від окремих користувачів, і підрахує загальну суму заходу. Також є можливість отримати звіт про те, як використовувалися гроші та предмети. Гроші представлені у вигляді предмета.

На головній сторінці є дашборд з поточними та майбутніми подіями. Поточна діяльність визначається таким чином: кожна подія має дату початку та дату завершення. Якщо поточна дата припадає на цей часовий проміжок, подія вважається поточною. На рисунку 3.5 показаний головний екран програми.

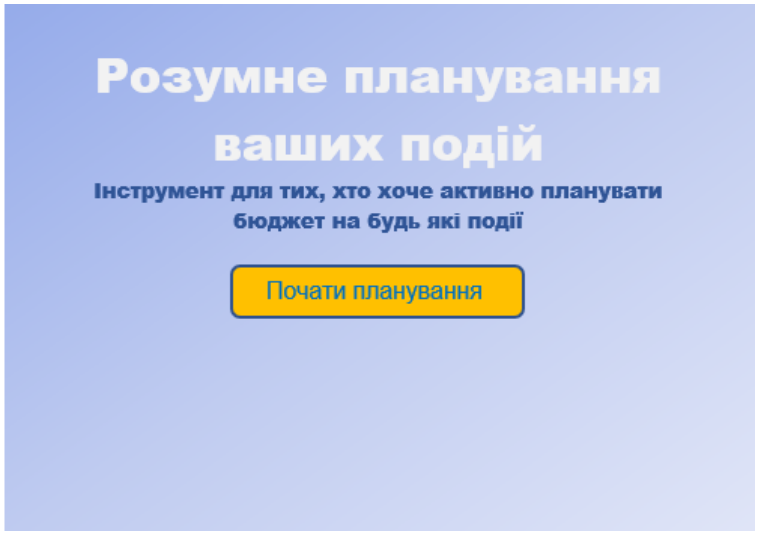

Рисунок 3.5 – Головний екран програми

На рисунку 3.6 показаний екран реєстрації.

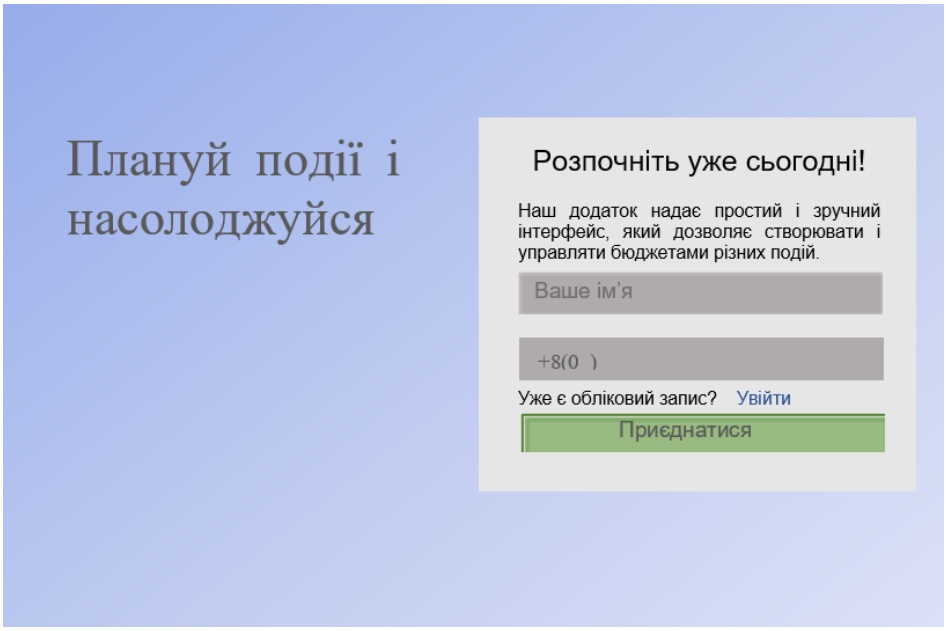

Рисунок 3.6 – Екран реєстрації

На рисунку 3.7 показані заплановані дії користувача.

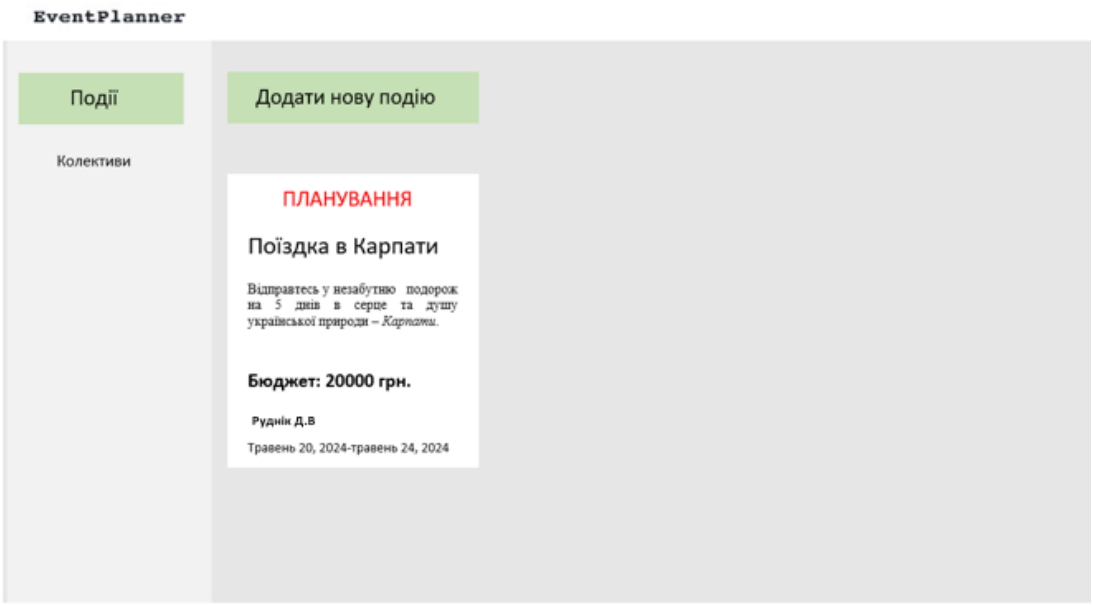

Рисунок 3.7 – Екран запланованих подій

На рисунку 3.8 показана форма додавання нової активності. Також є можливість внесення грошей в бюджет заходу.

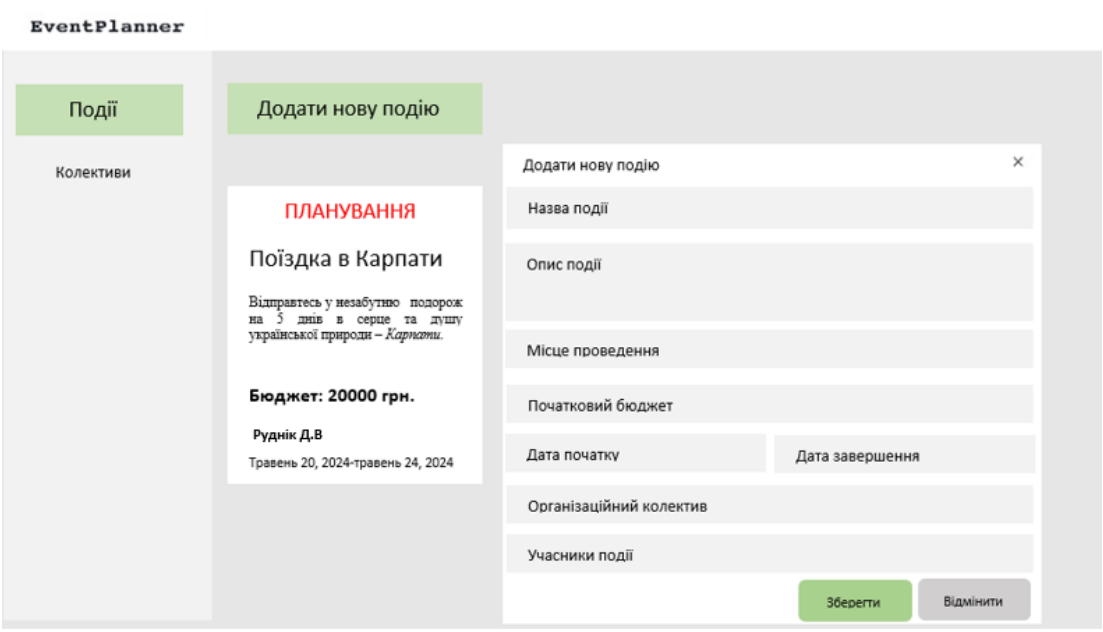

Рисунок 3.8 – Форма «Додати подію»

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи бакалавра поставлена мета була досягнута і поставлені завдання вирішені.

Проводився аналіз предметної області, в якій розглядалися аналогічні заявки. Розкрито їх переваги та недоліки. Також провівся аналіз вимог до системи і сформулювався функціонал.

Наступним етапом було проектування. Створено функціональну модель IDEF0, діаграму варіантів використання та карту інтерфейсу.

Також були проаналізовані різні інструменти розробки баз даних, фронтенди та бекенди. Розглянуто переваги та недоліки, на підставі яких визначено найбільш підходящі інструменти розробки: React ,Nest.js, бібліотека Tanstack Query.

Після цього реалізовано серверну частину додатку, включаючи модулі для управління користувачами, заходами, внесками та витратами, а також генерації фінансових звітів.

Розроблено користувацький інтерфейс, який забезпечує зручний доступ до функціональності додатку через веб-браузер. Забезпечено безпечну аутентифікацію та авторизацію користувачів.

У ході виконання кваліфікаційної роботи було розроблено веб-додаток для обліку внесків та витрат на проведення колективних заходів з використанням Nest.js. Додаток забезпечує:

– зручний і інтуїтивно зрозумілий інтерфейс для організаторів заходів та учасників;

– можливість управління користувачами та заходами з розширеними функціями для додавання, редагування та видалення даних;

– ефективний облік фінансів з автоматичним розрахунком часток витрат та внесків, що забезпечує прозорість та точність обліку;

– безпеку даних завдяки використанню сучасних методів аутентифікації та авторизації.

Розроблений додаток має великі можливості для подальшого розвитку та вдосконалення. Серед перспективних напрямків можна виділити наступні: розширення функціональності, розробка мобільних версій додатку для платформ iOS та Android, що забезпечить зручність доступу до системи з будь-якого пристрою, подальша оптимізація роботи системи для забезпечення стабільної роботи під великими навантаженнями.

Розроблений веб-додаток є ефективним інструментом для обліку внесків та витрат на проведення колективних заходів, що має потенціал для подальшого розвитку та впровадження в реальних умовах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Splitwise. URL: https://www.splitwise.com/ (дата звернення: 05.05.2024).
2. Toshl Finance. URL: https://www.google.com/search?client=firefox-b-d&q=Toshl+Finance+ (дата звернення: 05.05.2024).
3. Mongodb. URL: https://www.mongodb.com/ (дата звернення: 10.05.2024).
4. MySQL. URL: https://www.mysql.com/ (дата звернення: 10.05.2024).
5. Postgresq. URL: https://www.postgresql.org/ (дата звернення: 10.05.2024).
6. Angular. URL: https://angular.dev/ (дата звернення: 10.05.2024).
7. VueJs. URL: https://vuejs.org/ (дата звернення: 10.05.2024).
8. Nest.js. URL: https://foxminded.ua/nest-js/ (дата звернення: 20.05.2024).
9. Nest.js. URL: https://w3schoolsua.github.io/nodejs/index.html#gsc.tab=0 (дата звернення: 20.05.2024).
10. ExpressJs. URL: https://expressjs.com/uk/ (дата звернення: 20.05.2024).
11. Typescript. URL: https://www.typescriptlang.org/ (дата звернення: 20.05.2024).
12. NestJs. URL: https://nestjs.com/ (дата звернення: 20.05.2024).