МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБСАЙТУ ФІТНЕС-ЦЕНТРІВ З ВИКОРИСТАННЯМ VUE.JS»**

Виконав: студент 4 курсу, групи 6.1219-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Є.О. Глухов
(ініціали та прізвище)

Керівник доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.пед.н. Пшенична О.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Глухову Єгору Олеговичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебсайту фітнес-центрів з використанням Vue.js

керівник роботи Панасенко Євген Валерійович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка вебсайту фітнес-центрів з використанням Vue.js.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

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

7. Дата видачі завдання 07.02.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	14.02.2023	
2.	Збір вихідних даних.	07.03.2023	
3.	Обробка методичних та теоретичних	21.03.2023	
	джерел.		
4.	Розробка першого та другого розділу.	18.04.2023	
5.	Розробка третього розділу.	09.05.2023	
6.	Оформлення та нормоконтроль	28.05.2023	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	22.06.2023	

Студент _____ Є.О. Глухов
(підпис) (ініціали та прізвище)

Керівник роботи _____ Є.В. Панасенко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка вебсайту фітнес-центрів з використанням Vue.js»: 72 с., 20 рис., 19 табл., 18 джерел, 2 додатки.

БІБЛІОТЕКА, ВЕБІНТЕРФЕЙС, СИСТЕМА, HTML, CSS, JAVASCRIPT, SINGLE PAGE, VUEJS.

Об’єкт дослідження – фреймворк Vue.js, бібліотеки для роботи з ним, взаємодія користувача з інтерфейсом системи.

Мета роботи: розробка вебінтерфейсу для мережі фітнес-центрів засобами Vue.js.

Методи дослідження – метод збору аналізу вимог до програмного забезпечення, методи проєктування та конструювання програмного забезпечення.

У кваліфікаційній роботі приведені основні теоретичні підходи та засоби розробки програмного забезпечення. На основі фреймворку Vue.js розроблено вебінтерфейс системи керування фітнес-центорм. У роботі розроблено функціонал, необхідний для керування фітнес-центром з використанням мови програмування Javascript.

SUMMARY

Master’s qualifying paper «Development of a Website for a Fitness Centers Using Vue.js»: 72 pages, 20 figures, 19 tables, 18 references, 2 supplements.

HTML, CSS, JAVASCRIPT, LIBRARY, SINGLE PAGE, SYSTEM, VUEJS, WEBINTERFACE.

The object of the study is Vue.js framework, libraries for working with it, interfacing with the system interface.

The aim of the study is developmenting of a web interface for a network of fitness centers using Vue.js.

The method of research is a method of gathering software requirements analysis, software design and construction methods.

The qualification paper presents the main theoretical approaches and means of software development. The web interface of the fitness center management system was developed based on the Vue.js framework. In the work, the functionality necessary for managing a fitness center using the Javascript programming language has been developed.

ЗМІСТ

ВСТУП

Кваліфікаційна робота присвячена розробці вебінтерфейсу системи для мережі фітнес-центрів з використанням Vue.js. У сучасному світі фітнес-індустрія має значний розмах і швидкий розвиток. Для оптимізації роботи фітнес-центрів та покращення користувацького досвіду важливо мати потужну та ефективну систему управління.

У роботі розглядаються основні принципи та методи розробки вебінтерфейсу з використанням Vue.js – прогресивного фреймворку JavaScript для побудови користувацьких інтерфейсів. Досліджуються основні можливості фреймворку та його переваги у контексті розробки вебінтерфейсу системи для мережі фітнес-центрів.

Метою кваліфікаційної роботи є розробка інтерфейсу системи для будь-якого фітнес-центру, щоб замінити паперову документацію та спростити роботу з даними.

1 ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ РОЗРОБКИ

1.1 Огляд засобів веброзробки

Javascript є динамічною, об'єктно-орієнтованою прототипною мовою програмування, яка в основному використовується для створення сценаріїв на вебсторінках. Вона надає можливість взаємодіяти з користувачем на боці клієнта (у пристроях кінцевого користувача), керувати браузером, асинхронно обмінюватися даними з сервером та змінювати структуру та зовнішній вигляд вебсторінки [1].

JavaScript можна визначити як скриптову мову програмування з динамічною типізацією, яка класифікується як прототипна (як підмножина об'єктно-орієнтованої мови). Крім прототипної парадигми, JavaScript також частково підтримує інші парадигми програмування, такі як імперативна і частково функціональна. Він також володіє деякими характеристиками архітектури, такими як динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування та можливість використання функцій як об'єктів першого класу.

TypeScript є мовою програмування, яка базується на JavaScript і має статичну типізацію. Ця мова була розроблена в 2012 році у компанії Microsoft під керівництвом Андреса Гейлсберга. TypeScript дозволяє оголошувати типи для змінних та інших структур даних, таких як рядки або булеві значення, і здійснює перевірку правильності цих значень. У випадку JavaScript, де типизація не є строгою, така можливість відсутня [2].

NPM (Node Package Manager) – це менеджер пакетів для JavaScript, який використовується для установки, управління та розповсюдження пакетів, модулів та залежностей в проєктах, побудованих з використанням Node.js. npm дозволяє розробникам легко встановлювати сторонні бібліотеки та модулі, а також керувати версіями залежностей у своїх проєктах. Він також надає доступ

до різноманітних інструментів та команд для роботи з пакетами, таких як оновлення, видалення, пошук, публікація та керування версіями. npm є вбудованим інструментом установки Node.js і є одним з найпоширеніших інструментів для розробки проєктів JavaScript та Node.js [3].

Axios – це бібліотека JavaScript, яка використовується для здійснення HTTP-запитів з браузера або з серверної сторони (Node.js). Вона дозволяє взаємодіяти з API та виконувати асинхронні запити для отримання даних з сервера або відправлення даних на сервер [4].

Наведемо основні переваги Axios.

Простота використання. Axios надає простий та зрозумілий API, що дозволяє легко виконувати HTTP-запити. Він пропонує методи, такі як axios.get(), axios.post(), axios.put(), axios.delete() і т.д., які дозволяють зробити різні типи запитів з легкістю.

Підтримка обіцянок (Promises). Axios використовує обіцянки, що дозволяє зручно обробляти асинхронні запити та їхні результати. Ви можете використовувати .then() та .catch() для обробки успішних та помилкових відповідей сервера.

Підтримка перехоплення запитів та відповідей. Axios дозволяє перехоплювати та обробляти запити та відповіді перед їх відправленням або після отримання. Це корисно для авторизації, додавання заголовків, обробки помилок та інших маніпуляцій з даними.

Підтримка відміни запитів. Axios надає можливість відміняти запити у разі потреби. Це особливо корисно, коли користувач переключається на іншу сторінку або відправляє новий запит, і вам потрібно відмінити попередній запит.

Широкий спектр функціональності. Axios підтримує багато функцій, таких як встановлення заголовків, відправлення файлів, перехоплення прогресу завантаження та багато іншого. Він також добре інтегрується з іншими бібліотеками та фреймворками, такими як Vue.js та React.

Ці переваги роблять Axios популярним вибором для виконання HTTP-запитів в JavaScript-проєктах, незалежно від того, чи ви працюєте на клієнтській

стороні, чи на сервері.

HTML є стандартизованою мовою розмітки документів, призначеною для перегляду вебсторінок у браузері. Веббраузери отримують HTML документи від сервера за допомогою протоколів HTTP/HTTPS або відкривають їх з локального диска, після чого інтерпретують код і відображають його інтерфейс на екрані монітора [5].

Елементи HTML виступають як будівельні блоки сторінок HTML. Завдяки HTML конструкціям, зображення та інші об'єкти, такі як інтерактивні форми, можуть бути вбудовані у відображену сторінку. HTML надає засоби для створення структурованих документів, за допомогою яких можна позначити структурну семантику тексту, таку як заголовки, абзаци, списки, посилання, цитати та інші елементи.

CSS представляє собою спеціальну мову стилів, яка використовується для опису зовнішнього вигляду вебсторінок. Власне сторінки самі за себе написані за допомогою мов розмітки даних. CSS є ключовою технологією у світовій павутині, разом з HTML та JavaScript. Зазвичай CSS використовується для візуальної презентації сторінок, які написані на HTML або XHTML, але формат CSS може застосовуватися й до інших типів XML-документів [6].

Vexip UI є компонентною бібліотекою користувацького інтерфейсу (UI) для розробки вебдодатків. Вона забезпечує широкий набір готових компонентів, які можна використовувати для швидкої і простої побудови сучасних інтерфейсів користувача [7].

Vexip UI пропонує різноманітні компоненти, такі як кнопки, форми, таблиці, модальні вікна, панелі, меню та багато інших. Ці компоненти мають зручний і чистий дизайн, який добре підходить для розробки професійного вигляду вебдодатків.

Однією з основних переваг Vexip UI є його висока налаштованість та гнучкість. Розробники можуть легко налаштовувати вигляд та поведінку компонентів, використовуючи різні параметри та опції.

Vue.js є JavaScript-фреймворком, який використовує шаблон MVVM

(Model-View-ViewModel) для створення інтерфейсів користувача на основі моделей даних, забезпечуючи реактивне зв'язування даних [8].

У Vue.js використовується синтаксис шаблонів, що базується на HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue. Всі шаблони Vue є валідними HTML і можуть бути розпарсені браузерами та парсерами HTML. Усередині Vue компілюються шаблони в рендерингові функції віртуального DOM. Завдяки реактивній системі, Vue здатний розумно визначати, які компоненти потребують ре-рендерингу та застосовувати мінімальну кількість маніпуляцій з DOM, коли стан застосунку змінюється.

Однією з найвиразніших особливостей Vue є його ненав'язлива реактивна система. Моделі просто є плоскими об'єктами JavaScript. Це робить керування станами дуже простим та інтуїтивним. Vue надає оптимізований ре-рендеринг «з коробки», не потребуючи додаткових зусиль. Кожен компонент відстежує свої реактивні залежності під час рендерингу, що дозволяє системі точно визначити, коли потрібен ре-рендеринг та які компоненти потрібно оновити.

Vue-Cal, або Vue Calendar (див. рис. 1.1), є компонентом або бібліотекою для Vue.js, яка надає можливість легко і зручно створювати та відображати календарні компоненти в вебдодатках. Vue-Cal забезпечує розширені функціональні можливості для відображення подій, зустрічей, завдань та інших елементів на календарі, а також взаємодії з ними, такими як додавання, редагування та видалення подій. Бібліотека має гнучку конфігурацію, що дозволяє налаштувати вигляд та поведінку календаря відповідно до вимог та потреб проєкту. Vue-Cal є популярним рішенням для розробки календарних функціональностей в Vue.js-додатках [9].

Pinia – це становий управлінський пакет для Vue.js, який надає простий та ефективний спосіб керування станом додатків. Він є альтернативою до популярного станового менеджера Vuex і пропонує декларативний підхід до роботи зі станом додатків [10].

Pinia працює на основі концепції станових сховищ (stores), які

представляють окремі частини стану додатку. Кожне станове сховище містить дані, методи для їх зміни та обробки, а також спеціальні функції для взаємодії з іншими сховищами.

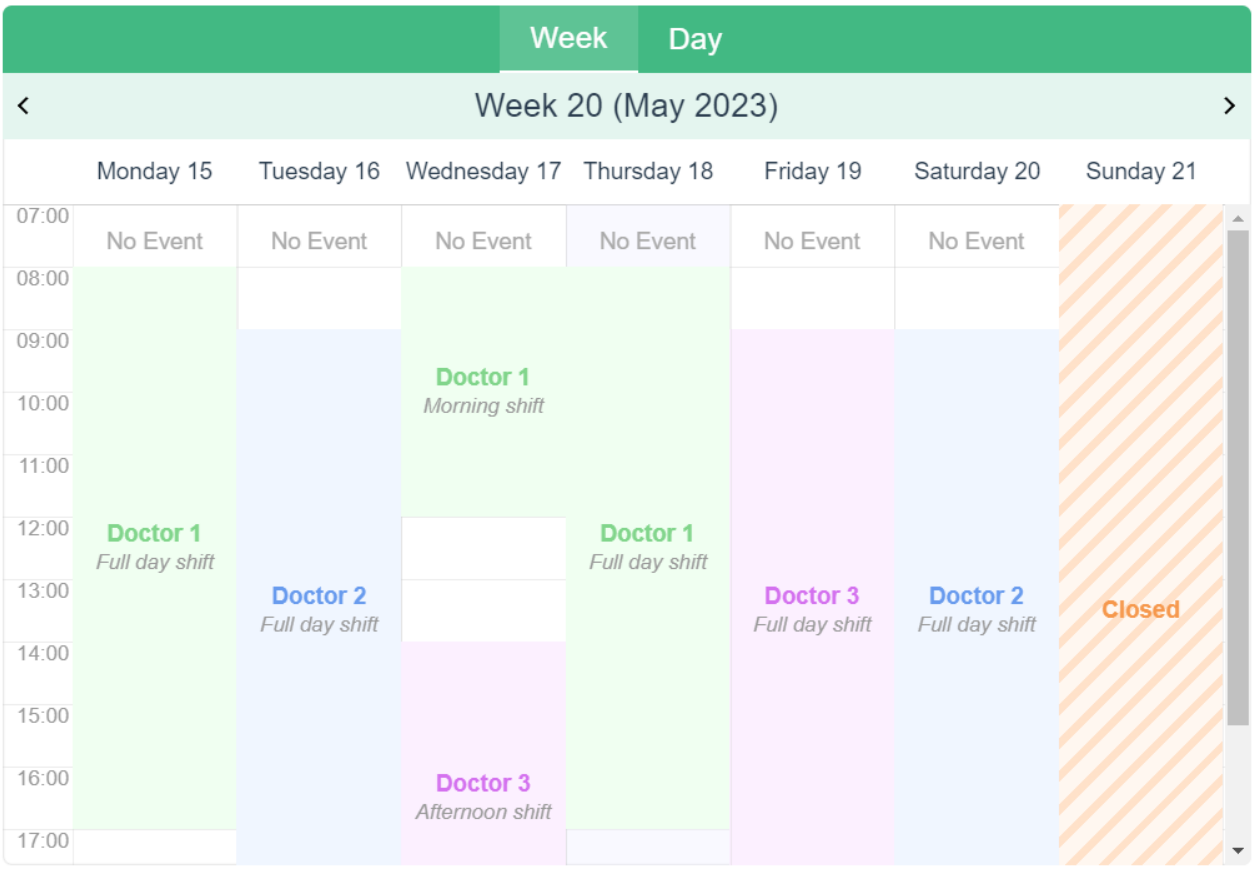

Рисунок 1.1 – Приклад вигляду розкладу Vue-Cal

Один з ключових аспектів Pinia – це типізація. Він інтегрується з TypeScript і надає можливість типізувати стан, методи та змінні в сховищах, що дозволяє знизити кількість помилок та полегшує розробку. Pinia також підтримує реактивну систему Vue, що дозволяє автоматично оновлювати компоненти, коли стан змінюється.

Pinia і Vuex є обидва становим управлінням для Vue.js, але вони мають деякі відмінності у своїй функціональності та підходах.

Архітектура. Vuex базується на концепції централізованого сховища (centralized store), де всі дані додатка зберігаються в одному сторі. Pinia, з іншого боку, використовує розподілену архітектуру (distributed architecture), де кожен

модуль має своє власне сховище. Це дозволяє кращу організацію та розширюваність коду.

Типізація. Pinia підтримує статичну типізацію з використанням TypeScript, що дозволяє виявляти помилки типів на етапі розробки. Vuex, у свою чергу, не має вбудованої підтримки статичної типізації, але може бути використаний з TypeScript за допомогою додаткових налаштувань.

Масштабованість. Pinia пропонує більш гнучку масштабованість, оскільки дозволяє створювати декілька сховищ та модулів залежно від потреб проєкту. Vuex, натомість, працює з одним централізованим сховищем, що може бути складніше масштабувати для великих проєктів.

Синтаксис. Синтаксис і спосіб використання таких функцій як getters, mutations і actions в Pinia та Vuex трохи відрізняються. Pinia використовує прямий доступ до сховища та має простий та прямолінійний синтаксис. Vuex використовує «контекст» для доступу до модулів та має більш складну структуру коду.

Розмір. Pinia має менший розмір (вагу) порівняно з Vuex, що може бути важливим фактором для додатків з обмеженими обсягами.

Вибір між Pinia та Vuex залежить від певних потреб, розміру проєкту та особистих вподобань. Pinia може бути більш підходящим для проєктів з більшою розміром та комплексністю, якщо потрібна статична типізація та більша гнучкість масштабування. Vuex є добрим вибором для менших проєктів або тих, які не вимагають складної архітектури.

Vite – це швидкий і легкий інструмент для розробки вебдодатків, оснований на сучасних вебтехнологіях, таких як ES modules, HMR (гаряча заміна модулів) і багатопоточна обробка. Він спроєктований для прискорення процесу розробки та покращення продуктивності розробників [11].

Наведемо основні переваги Vite.

Швидкість завантаження. Vite використовує ES modules для завантаження модулів у браузері, що дозволяє швидко завантажувати код без необхідності його компіляції або бандлінгу перед запуском. Це дозволяє

зменшити час очікування під час розробки і підвищити продуктивність.

Гаряча заміна модулів (HMR). Vite підтримує гарячу заміну модулів, що дозволяє вам відразу бачити зміни в коді без перезавантаження сторінки. Це забезпечує швидкий розкрут модулів під час розробки та спрощує відладку.

Простота налаштування. Vite має просту конфігурацію та структуру проєкту. Ви можете швидко налаштувати свої вебдодатки без необхідності в складних налаштуваннях.

Розширюваність. Vite є розширюваним за допомогою плагінів, які дозволяють вам додавати додаткову функціональність до вашого проєкту. Ви можете використовувати готові плагіни або створити власні.

Підтримка Vue і React. Vite надає вбудовану підтримку для розробки вебдодатків на Vue.js та React. Це означає, що ви можете легко створювати проєкти на основі цих фреймворків з використанням всіх переваг, які надає Vite.

Загалом, Vite пропонує швидку та ефективну розробку вебдодатків, забезпечуючи переваги швидкості завантаження, гарячої заміни модулів та простоти налаштування. Він є особливо підходящим для проєктів, що використовують сучасні вебтехнології та фреймворки.

Model-View-ViewModel (MVVM) є шаблоном проєктування, який використовується при архітектурному проєктуванні додатків (див. рис. 1.2). Вперше він був представлений Джоном Госсманом у 2005 році як модифікація шаблону Presentation Model. MVVM спрямований на сучасні платформи розробки, такі як Windows Presentation Foundation та Silverlight від компанії Microsoft [12].

MVVM спрощує відокремлення розробки графічного інтерфейсу від розробки бізнес-логіки (бек-енд логіки), яка відома як модель. Відокремлення представлення від моделі є головною метою. Модель-представлення відповідає за трансформацію даних для їх підтримки та використання. З цієї точки зору, модель-представлення більше схожа на модель, ніж на представлення, і вона обробляє більшість, якщо не всю, логіку відображення даних. Модель-представлення також може використовувати патерн медіатор, що організовує

доступ до бек-енд логіки за допомогою набору правил використання, які підтримуються представленням.

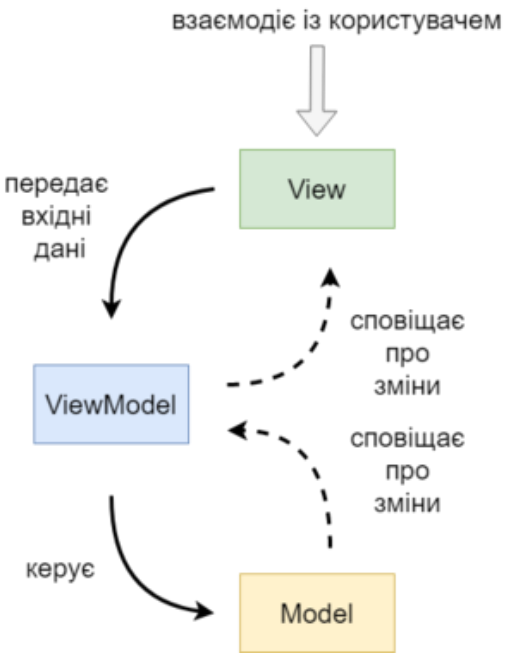

Рисунок 1.2 – Шаблон MVVM

Створення зручних і привабливих інтерфейсів вимагає інших навичок, ніж просто розробка програмного коду, і саме для цього був створений MVVM. Він дозволяє розподілити роботу між дизайнерами і програмістами, що є складним завданням, коли Java-розробник намагається побудувати графічний інтерфейс у Swing або розробник на Visual C++ створює користувацький інтерфейс у MFC. Розробники володіють різними навичками, і створення зручних та привабливих інтерфейсів вимагає спеціальних талантів, які можуть бути відмінними від навичок, якими володіють розробники програмного коду.

Система керування версіями (СКВ, англ. source code management, SCM) – це програмний інструмент, який дозволяє керувати версіями різних типів інформації, таких як початковий код програми, скрипти, вебсторінки, вебсайти, 3D-моделі, текстові документи та інше. СКВ забезпечує можливість працювати над проєктами в групі, не перешкоджаючи один одному [13].

Системи керування версіями широко використовуються при розробці програмного забезпечення для відстеження, документування та контролю над

поступовими змінами в електронних документах, таких як вихідний код програм, креслення, електронні моделі та інші документи, над якими працюють декілька людей одночасно.

Кожна версія в СКВ має унікальний ідентифікатор (цифра або літера), і зміни, які вносяться до документу, фіксуються. Зазвичай зберігаються також дані про автора змін та час їх внесення. Інструменти керування версіями часто включаються до складу інтегрованих середовищ розробки.

Таким чином, система керування версіями є невід'ємною частиною процесу розробки програмного забезпечення, яка допомагає контролювати зміни, спільно працювати над проєктами та зберігати історію різних версій документів.

1.2 Опис предметної області

Предметна область системи мережі фітнес-центрів охоплює всі аспекти, пов'язані з управлінням та функціонуванням фітнес-центрів. Вона включає різні елементи, процеси та взаємодії, які відбуваються у цьому середовищі.

Філія є однією з ключових складових предметної області. Філії можуть мати різні місцезнаходження, такі як тренажерні зали, де надаються різноманітні послуги, такі як групові та індивідуальні тренування або SPA-процедури.

Іншою важливою сутністю є Абонемент, який визначає доступ клієнтів до послуг, наданих фітнес-центром. Існують різні типи абонементів, такі як стандартний (з обмеженою кількістю відвідувань протягом певного періоду), безлімітний (без обмежень у кількості відвідувань протягом певного періоду) і free way (з обмеженою кількістю відвідувань, але з доступом до всіх видів послуг). Таким чином, абонементи можуть мати різні умови, обмеження та терміни дії.

Клієнти представляють особи, які скористалися послугами фітнес-центру. Вони можуть мати різні персональні дані, такі як ім'я, прізвище, номер телефону

та дата народження.

Робітники є фахівцями, які забезпечують індивідуальні консультації та проводять тренування клієнтам або надають SPA-послуги. Тренери можуть мати різні спеціалізації та сертифікації.

Розклади занять визначають час і місце проведення тренувань, групових занять та інших подій у фітнес-центрі.

Події можуть бути регулярними або нерегулярними і включати різні види активностей. Контроль платежів є також важливим аспектом.

Платежі включають оплату абонементів та інших витрат, пов'язаних з функціонуванням фітнес-центру. Вони можуть бути здійснені різними способами оплати, такими як готівка, кредитні картки або електронні платіжні системи.

Для власника бізнесу важливо аналізувати статистику різних показників для прийняття рішень щодо розвитку свого підприємства. Збір та аналіз даних про продажі, відвідуваність, популярність послуг та інші показники є необхідними для ефективного управління філіями. Звіти та аналітика допомагають зрозуміти стан бізнесу, приймати рішення та планувати подальший розвиток.

1.3 Порівняльний аналіз фреймворків

На даний момент існують три найпопулярніші фронтенд фреймоврки: Vue, React [14] та Angular [15]. Вони мають різні особливості та підходи до розробки, тому, щоб обрати найоптимальніший фреймворк для моєї кваліфікаційної роботи, мною було проведено порівняльний аналіз за декількома ключовими аспектами:

а) навчання та вступ до розробки:

1) Vue: вважається досить легким для вивчення та має просту синтаксичну структуру, вимагає менше знань попередньої розробки;

2) React: вимагає вміння працювати з JavaScript та JSX (розширення синтаксису JavaScript), має стриману API-документацію, що може потребувати додаткового зусилля для вивчення;
3) Angular: вимагає глибокого розуміння TypeScript та розробки з використанням паттернів, таких як Dependency Injection, має великий обсяг документації та концепцій, що можуть становити виклик для новачків;

б) розмір та продуктивність:
1) Vue: має дуже маленький розмір, що дозволяє швидко завантажувати вебсторінки, має хорошу продуктивність, але може мати обмеження при роботі з великими додатками;
2) React: має компактний розмір і добру продуктивність, використовує Virtual DOM, що забезпечує ефективне оновлення компонентів;
3) Angular: має більший розмір, що може призвести до довших часів завантаження, однак, зазвичай має добру продуктивність завдяки підхопленню змін;

в) екосистема та розширення:
1) Vue: має активну та швидкозростаючу екосистему, але менше засновників, багато готових розширень та компонентів, таких як Vue Router, Vuex;
2) React: має велику та зрілу екосистему, багато розширень та компонентів, таких як React Router, Redux;
3) Angular: має повноцінну екосистему, яка включає у себе багато офіційних розширень та модулів, Angular CLI допомагає з легкістю створювати та налаштовувати проєкти;

г) компонентна архітектура та реактивність:
1) Vue: основна концепція – це компонентна архітектура, пропонує реактивність через використання прослуховувачів подій та реактивних властивостей;
2) React: використовує компонентну архітектуру з JSX, забезпечує

реактивність за допомогою Virtual DOM та зміни стану компонентів;

3) Angular: використовує шаблони HTML для побудови компонентів, забезпечує реактивність через спеціальну бібліотеку RxJS та впровадження шаблонів;

д) спільнота та підтримка:

1) Vue: має активну та зростаючу спільноту розробників, отримує швидке оновлення та вирішення проблем;
2) React: має одну з найбільших спільнот розробників та широку підтримку, часті оновлення та вирішення проблем;
3) Angular: має велику спільноту розробників та активну підтримку від Google, часті оновлення та документація.

Зробивши порівняльний аналіз, я прийшов до висновку, що фреймворк Vue ідеально підходить для проєкту кваліфікаційної роботи через високу продуктивність, малий розмір, просту синтаксичну структуру.

1.4 Висновки до розділу 1

Отже, у цьому розділі було проведено огляд таких засобів веброзробки: мова програмування Javascript, мова програмування, що базується на Javscript – Typescript, бібліотека Javascript – Axios, мова розмітки HTML, мова стилів CSS, компонента бібліотека користувацького інтерфейсу Vexip UI, фреймворк Vue.js, бібліотека Vue.js – Vue-Cal, становий управліньский пакет для Vue.js – Pinia, інструмент розробки вебдодатків Vite, шаблон проєктування MVVM, система керування версіями SCM. Також була більш детально описана предметна область кваліфікаційної роботи, та наведено порівнювальний аналіз трьох популярних Javascript фреймворків – Vue, React та Angular.

2 ПРОЄКТУВАННЯ СИСТЕМИ ФІТНЕС ЗАЛІВ

2.1 Загальна логіка системи

Система дозволяє вести управління мережею фітнес-залів. Серверна складова надає простий програмний інтерфейс (API), а клієнтська складова забезпечує інтерфейс користувача та взаємодію з API.

Для використання функціоналу користувачу слід авторизуватись в системі, за допомогою логіну та паролю, також бажано вказати філію, де знаходиться користувач. Користувача повинен зареєструвати інший користувач, що має дозвіл на його створення.

Після реєстрації користувача, йому надається певна роль, як користувача, або менеджера системи, або робітника, або адміністратором системи. Відповідно до ролі користувача у них будуть різні функціональні та дизайнерські особливості.

2.2 Проєктування системи

Процес проєктування програмного забезпечення (ПЗ) включає визначення архітектури, компонентів, інтерфейсів та інших характеристик системи та кінцевого результату. У сфері проєктування інформаційних систем широко використовується уніфікована мова візуального моделювання (UML), яка дозволяє візуалізувати, специфікувати, конструювати та документувати програмні системи.

Проєктування ПЗ є важливим процесом, що передує розробці системи, оскільки дозволяє виявити характеристики та можливі слабкі місця майбутньої системи. На цьому етапі можна оцінити різні варіанти архітектури, враховуючи як системи загалом, так і мобільні додатки. Також можна оцінити варіанти

взаємодії користувача з системою та внести зміни до функціональних особливостей системи ще до початку розробки.

Розробка різних діаграм взаємодії компонентів у системі, потоків даних та взаємодії користувача з додатками і самою системою дозволяє оцінити обсяг робіт, знайти розробників або команду для реалізації процесів створення та підтримки майбутнього продукту.

Крім того, важливо пам'ятати, що діаграми є зручним інструментом для відображення та передачі інформації розробникам, аналітикам, тестувальникам та іншим зацікавленим сторонам, пов'язаним з системою.

2.2.1 Етап концептуального проєктування

На даному етапі можна проаналізувати різні варіанти взаємодії, такі як взаємодія користувача з системою, користувач-користувач і користувач-застосунок. Зі списку варіантів взаємодії стає зрозумілим, що користувач є основним об'єктом дослідження на даному етапі. Для відображення всіх можливих дій та процесів, пов'язаних з користувачем, була розроблена діаграма варіантів використання (діаграма прецедентів) (рис. 2.1).

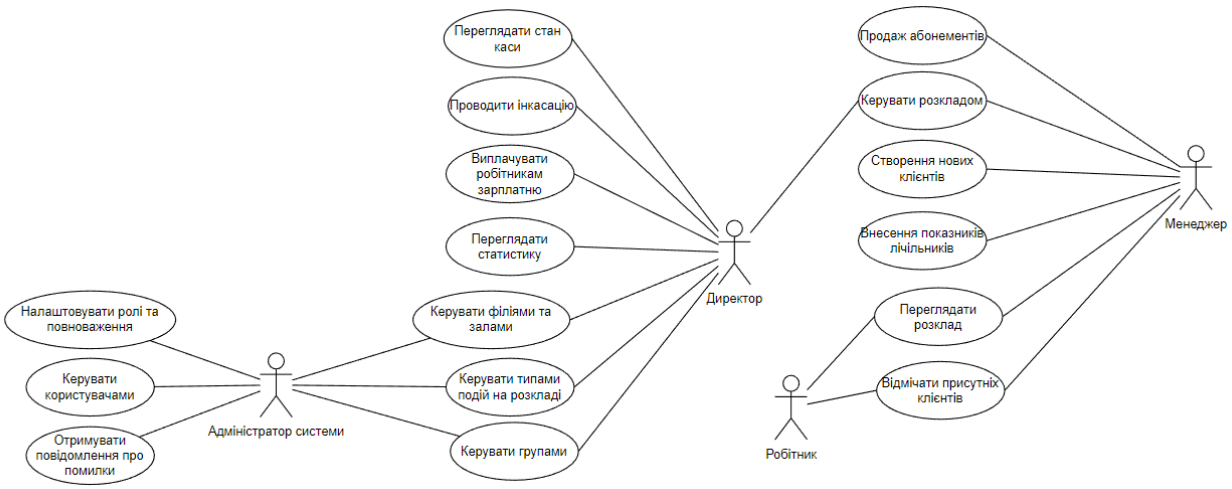

Рисунок 2.1 – Діаграма прецедентів

Ця діаграма відображає відношення між акторами (учасниками системи) та прецедентами (діями або функціональними вимогами) у системі.

Хоча сам варіант використання може містити багато деталей про кожну конкретну можливість, діаграма варіантів використання надає загальне уявлення про систему. Завдяки своєї спрощеній природі, діаграми варіантів використання можуть бути ефективним засобом комунікації зі зацікавленими сторонами. Вони намагаються імітувати реальний світ та дати зацікавленій стороні візуальне уявлення про те, як працюватиме розроблена система.

Розглянемо деякі основні варіанти використання майбутньої системи.

Прецедент «Створення нових клієнтів».

Призначення: даний варіант дозволяє менеджеру додавати у систему нових клієнтів, щоб можна було зафіксувати продажу абонементу та подальші дії клієнта у фітнес-центрі.

Основний потік подій: прецедент починає виконуватися, коли менеджер на головній сторінці у модальному вікні «Продаж абонементу» відкриває модальне вікно «Швидке створення клієнта» та вводить дані і реєструє нового клієнта. Також можливий інший варіант основного потоку подій, коли менеджер створює нового клієнта з сторінки «Створення користувача».

Передумова: менеджер повинен бути авторизований у системі та мати дозвіл на створення клієнтів.

Прецедент «Виплачувати робітникам зарплатню».

Призначення: даний варіант дозволяє фіксувати виплати працівникам за певний проміжок часу.

Основний потік подій: прецедент починає виконуватися, коли директор на сторінці «Надходження працівникам» обирає певного робітника та створює виплату за певний проміжок часу (зазвичай за один місяць).

Передумова: директор повинен бути авторизований у системі та у певного робітника повинне бути мінімум одне надходження.

Прецедент «Проводити інкасацію».

Призначення: фіксувати надходження та відходження у фітнес-центрі за

певними критеріями.

Основний потік подій: прецедент починає виконуватися, коли директор на сторінці «Платежі» обирає певних або усіх операторів (користувачів, котрі створювали певні дописи у системі, що пов’язані з грошовим оборотом) та фіксує грошовий оборот за певний проміжок часу.

Передумова: директор повинен бути авторизований у системі та у певних операторів має бути мінімум одне надходження/відходження.

Прецедент «Керування розкладом».

Призначення: створювати, редагувати або видаляти події на розкладі.

Основний потік подій: директор або менеджер, якщо у нього є дозвіл на створення/редагування/видалення подій на розкладі, можуть на головній сторінці, обрав певний зал, робити певні маніпуляції з подіями, тобто створювати кастомні або групові події, переносити подію на іншу дати, змінювати її тривалість, також можна видаляти події з розкладу.

Альтернативний потік подій: директор може створити групу зі своїм дефолтним розкладом, тоді події цієї групи автоматично з’являться на розкладі. Також менеджер може продати клієнту абонемент на індивідуальні заняття, тоді події теж автоматично з’являться на розкладі.

Передумова: користувач повинен бути авторизований у системі та мати дозвіл на створення/редагування/видалення подій.

Прецедент «Керування групами».

Призначення: створювати, редагувати або видаляти групи у системі.

Основний потік подій: директор та адміністратор системи можуть на сторінці «Групи» переглядати інформацію про кожну групу, створювати та редагувати групу, змінювати дефолтний розклад, видаляти групу.

Передумова: користувач повинен бути авторизований у системі та мати дозвіл на створення/редагування/видалення груп.

Прецедент «Продаж абонементів».

Призначення: фіксувати продаж абонементу певному клієнту.

Основний потік подій: менеджер на головній сторінці у модальному вікні

«Продаж абонементу» обирає клієнта, кому треба продати абонемент, та встановлює критерії цього абонемента, а саме:

– тип сервісу, котрий може бути груповими, індивідуальними заняттями, тощо;
– тип абонементу (стандартний, безлімітний, free way);
– кількість відвідувань (за умови, що не обрано безлімітний тип абонементу);
– типи сервісів, до яких може ходити клієнт за абонементом (не потрібно вказувати тільки у випадку придбання free way абонементу);
– працівники, до яких може ходити клієнт за абонементом (не потрібно вказувати тільки у випадку придбання free way абонементу);
– ціна, котра залежить від обраної кількості відвідувань.

Передумови: менеджер повинен бути авторизований до системи та мати дані, за якими треба продати абонемент.

На цей прецедент було розроблено діаграму послідовності (рис. 2.2).

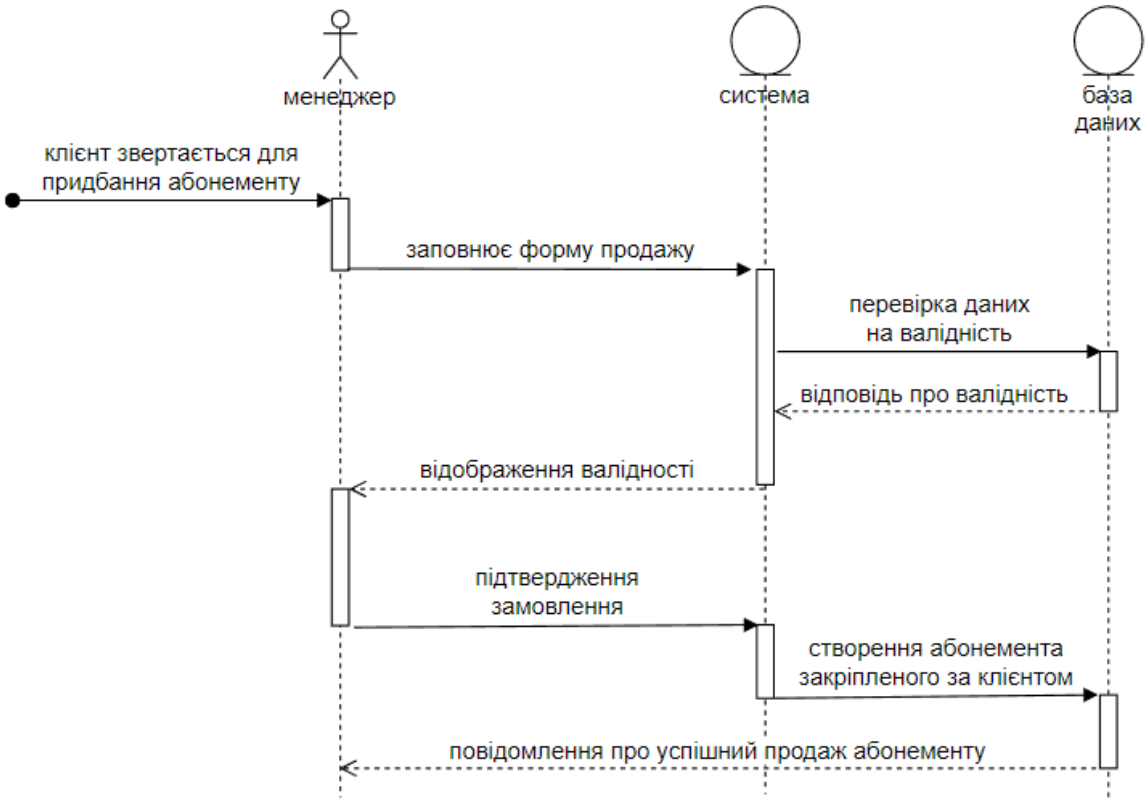

Рисунок 2.2 – Діаграма послідовності прецеденту «Продаж абонементів»

На цій діаграмі можна побачити, що клієнт звертається до менеджеру з бажанням купити абонемент до цього центру, менеджер заповнює необхідні дані, котрі відсилаються на сервер для перевірки на коректність, у випадку коректності заповнених даних менеджеру потрібно підтвердити продаж абонементу та вказати тип оплати, після цього на сервері створюється абонемент, що закріплюється за клієнтом та на екран виводиться повідомлення про успішний продаж абонементу.

Прецедент «Відмічати присутніх клієнтів».

Призначення: фіксувати присутність певних клієнтів до певних занять.

Основний потік подій: менеджер або працівник можуть відмітити присутність/відсутність певних клієнтів до певних занять. Це, в першу чергу, впливає на кількість залишившихся занять клієнта (за умови, що було придбано не безлімітний тип абонементу), також це впливає на надходження працівнику, бо працівник отримує певний відсоток за відвідування заняття за певним абонементом.

Передумови: користувач системи повинен бути авторизований до системи та мати дозвіл на відмічання присутності/відсутності клієнтів.

Також, на цей прецедент було побудовано діаграму послідовності (див. рис. 2.3).

На цій діаграмі можна побачити, що коли клієнт відвідує заняття, менеджер (або працівник) відкриває модальне вікно з детальною інформацією про заняття, в момент цього на сервер посилається запит на отримання цієї інформації, потім, коли інформація отримана та відображена на екрані, менеджер нажимає на кнопку відмітки клієнта, котрого треба відмітити, відсилається запит на сервер, де створюється запис у базі даних про цю відмітку та повертається повідомлення про успішне створення відмітки. Також, після успішного повідомлення на сервері створюється нарахування працівнику заняття, до котрого прийшов клієнт.

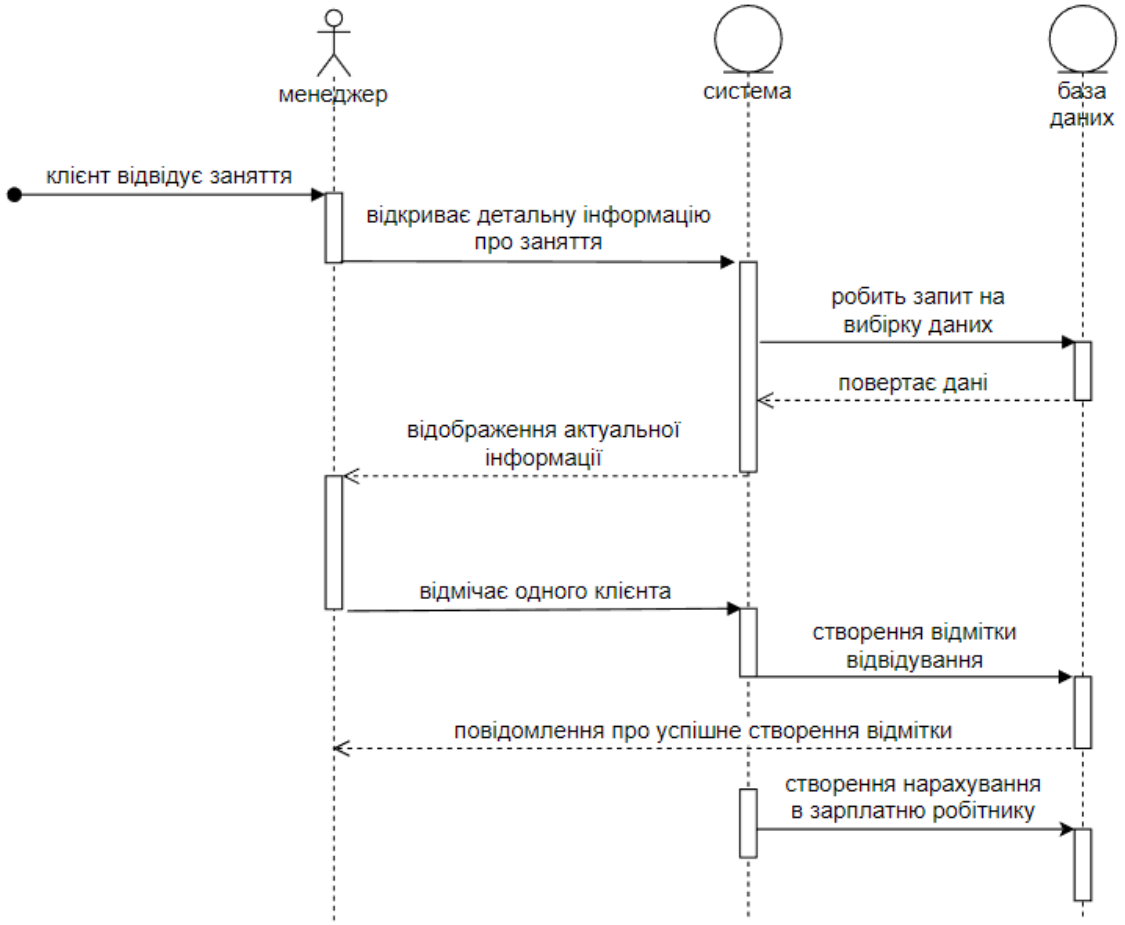

Рисунок 2.3 – Діаграма послідовності «Процес відмітки відвідування»

2.2.2 Етап логічного проєктування

Діаграма діяльності (activity diagram) в UML та SysML представляє собою візуальне відображення графу діяльностей. Граф діяльностей є певним варіантом графу станів скінченного автомату, де вершини представляють конкретні дії, а переходи відбуваються після завершення дій [16].

Дія (action) є основною одиницею визначення поведінки в специфікації. Дія отримує вхідні сигнали і перетворює їх на вихідні сигнали. Одна або обидві з цих множин можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Так само, виконання діяльності відповідає виконанню окремої діяльності, включаючи виконання всіх дій, що містяться в цій діяльності. Кожна дія в діяльності може виконуватись один або більше разів під час виконання

діяльності. Дії можуть отримувати дані, здійснювати їх перетворення і тестування, а деякі дії можуть вимагати певної послідовності. Специфікація діяльності на вищих рівнях може дозволяти виконання кількох (логічних) потоків та містити механізми синхронізації для забезпечення виконання дій у правильному порядку.

На етапі логічного проєктування було розроблено діаграму діяльності (див. рис. 2.4). На діаграмі відображено процес авторизації користувача до системи від моменту входу до отримання від серверу повідомлення про успішну чи невдалу спробу авторизації.

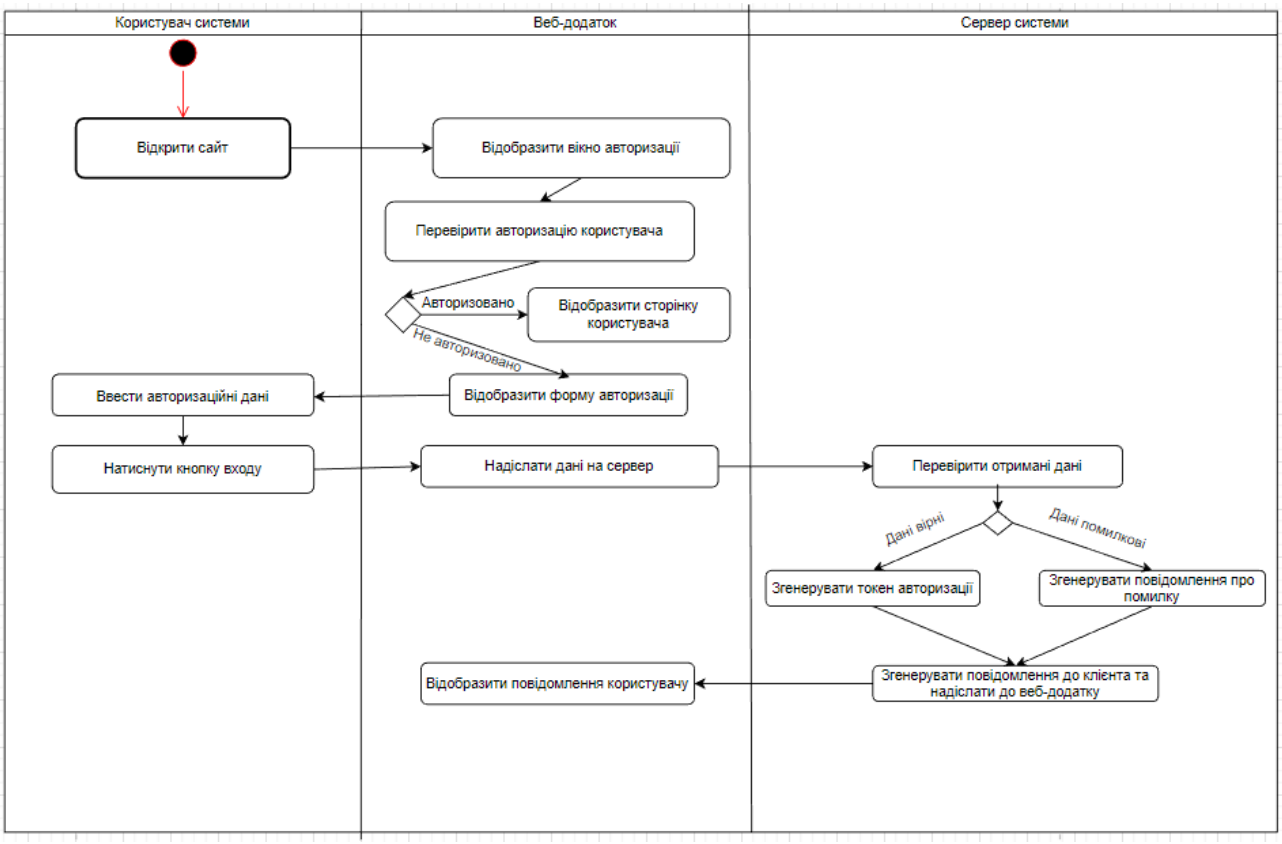

Рисунок 2.4 – Діаграма діяльності

2.2.3 Етап фізичного проєктування

Діаграма розгортання, використовувана в UML, є візуальним зображенням обчислювальних вузлів, компонентів та об'єктів, які виконуються на цих вузлах.

Кожен компонент відображає реалізацію певних кодових одиниць в програмі. На діаграмі розгортання не відображаються компоненти, які не мають реалізації в робочому середовищі; замість цього такі компоненти можуть бути показані на діаграмах компонентів. Діаграма розгортання демонструє робочі екземпляри компонентів, тоді як діаграма компонентів відображає зв'язки між типами компонентів [17].

Отже, процес взаємодії полягає в тому, що клієнтська програма ініціює запит до сервера за допомогою протоколу HTTP. При отриманні запиту, вебсервер обробляє його та взаємодіє з інтерфейсом бази даних для отримання необхідної інформації. Після обробки сервер повертає результат клієнту у форматі JSON.

Щоб отримати загальне розуміння роботи системи та взаємодії з клієнтською програмою, ми можемо скласти діаграму розгортання (див. рис. 2.5).

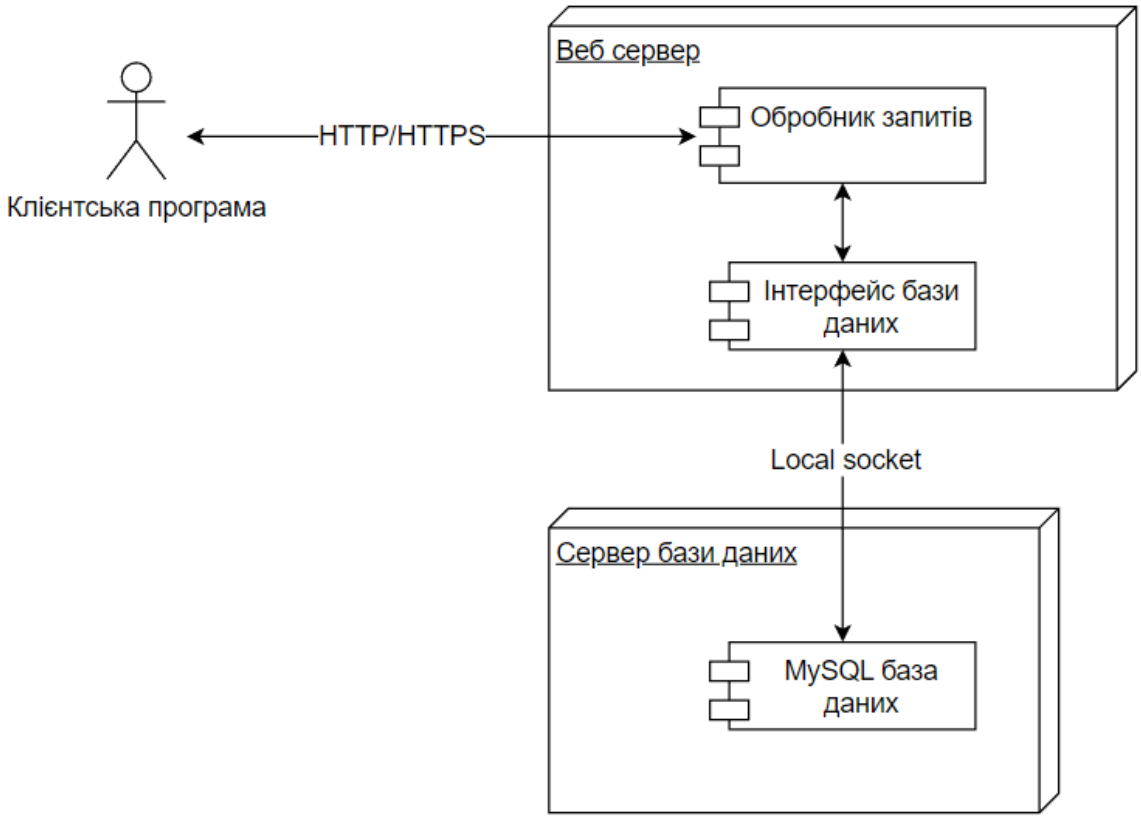

Рисунок 2.5 – Діаграма розгортання

2.2.4 Проєктування структури даних

На цьому етапі було проведено проєктування загальної структури інформаційної моделі даних для системи управління фітнес-центром. Структура моделі даних була представлена у формі ER-діаграми (див. рис. А.1), яка включає опис таблиць у базі даних, полів у цих таблицях та їх характеристики.

Модель «сутність-зв'язок» (ER-модель) є формою моделювання даних, що дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель є мета-моделлю даних, що служить для опису моделей даних незалежно від конкретного програмного забезпечення. Вона є універсальним інструментом для представлення даних і може бути використана для породження інших моделей даних, таких як ієрархічна, мережева, реляційна і об'єктна моделі [18].

Модель сутність-зв'язок відображає систематичний процес опису та визначення предметної області. Вона візуалізує дані у вигляді компонентів, відомих як сутності, які взаємодіють між собою за допомогою зв'язків, виражаючи залежності і вимоги. Наприклад, одна будівля може містити кілька квартир, але кожна квартира може бути прив'язана тільки до однієї будівлі. Сутності можуть мати атрибути, що описують їх властивості. Графічні діаграми, що відображають сутності, атрибути і зв'язки, називаються діаграмами сутність-зв'язок.

ER-модель часто реалізується у вигляді баз даних. У реляційній базі даних, де дані зберігаються у вигляді таблиць, кожний рядок таблиці представляє один екземпляр сутності. Деякі поля в таблицях вказують на індекси в інших таблицях, використовуючи покажчики, які фізично реалізують зв'язки між сутностями.

Розглянемо детальніше сутності-зв’язки інформаційної моделі системи.

Сутність User – це користувач системи, який може мати певні ролі та, відповідно до них, дозволи на певні дії у системі (див. табл. 2.1).

Таблиця 2.1 – Користувачі

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор користувача
name	VARCHAR (255)		Логін користувача
email	VARCHAR (255)		Електрона пошта користувача
email_verified_at	TIMESTAMP		Дата підтвердження електронної пошти користувача
first_name	VARCHAR (255)		Ім’я користувача
last_name	VARCHAR (255)		Прізвище користувача
patron_name	VARCHAR (255)		По-батькові користувача
phone	VARCHAR (255)		Телефон користувача
birth_date	DATE		Дата народження користувача
password	VARCHAR (255)		Hash-пароль користувача
remember_token	VARCHAR (100)		Токен авторизації
created_at	TIMESTAMP		Дата створення користувача
updated_at	TIMESTAMP		Дата останнього редагування

Продовження табл. 2.1

Назва поля	Тип даних	Властивість	Опис
			користувача
deleted_at	TIMESTAMP		Дата видалення користувача
branch_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор філії, до якої належить користувач

Сутність UserSetting – це певне налаштування користувача системи. В цій системі користувач може налаштувати наступні властивості:

– мову системи (українська або англійська);
– тему вебдодатку (темна або світла);
– стан бокового меню (згорнутий чи розгорнутий).

Сутність Permission – це дозвіл на будь-яку дію у системі (табл. 2.2).

Таблиця 2.2 – Дозволи

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор дозволу
name	VARCHAR (255)		Назва дозволу
created_at	TIMESTAMP		Дата створення дозволу
updated_at	TIMESTAMP		Дата останнього редагування дозволу

Сутність Role – це сукупність певних дозволів у системі для користувачів (див. табл. 2.3).

За замовчуванням, у системі існують такі ролі:

- Super Administrator – користувач, який має дозвіл на усі дії в системі;
- директор – голова фітнес-центру, який має дозвіл на усі дії в системі в межах свого фітнес-центру;
- адміністратор системи – користувач, який керує іншими користувачами, отримує повідомлення про помилки, керує типами подій на розкладі, налаштовує ролі та повноваження для користувачів свого фітнес-центру, керує групами;
- менеджер – користувач, який керує розкладом занять, продає абонементи, створює нових клієнтів, вносить показники лічильників, відмічає присутніх клієнтів;
- працівник – користувач, який може відмічати присутніх клієнтів на своїх подіях та дивитися певну статистику про себе;
- клієнт – користувач, який не має доступу до системи.

Таблиця 2.3 – Ролі

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор ролі
name	VARCHAR (255)		Назва ролі
created_at	TIMESTAMP		Дата створення ролі
updated_at	TIMESTAMP		Дата останнього редагування ролі

Відношення permission_role – це зв’язок між дозволами та ролями. У одної ролі може бути декілька дозволів, та один дозвол може бути у декількох ролів.

Відношення role_user – це зв’язок між ролями та користувачами. Одна роль може бути у декількох користувачів, та у одного користувача може бути декілька ролей.

Сутність Branch – це філія певного фітнес-центру, де проходять заняття (табл. 2.4).

Таблиця 2.4 – Філії

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор філії
name	VARCHAR (255)		Назва філії
address	VARCHAR (255)		Адреса філії
created_at	TIMESTAMP		Дата створення філії
updated_at	TIMESTAMP		Дата останнього редагування філії
deleted_at	TIMESTAMP		Дата видалення філії

Відношення branch_user – це зв’язок між філіями та користувачами. За одним користувачем може бути закріплена тільки одна філія, в той час як за однією філією може бути закріплено декілька користувачів.

Сутність Room – це зал певної філії, де безпосередньо відбуваються заняття (табл. 2.5).

Таблиця 2.5 – Зали

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор залу
name	VARCHAR (255)		Назва залу
branch_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор філії, за якої

Продовження табл. 2.5

Назва поля	Тип даних	Властивість	Опис
			закріплено зал
created_at	TIMESTAMP		Дата створення залу
updated_at	TIMESTAMP		Дата останнього редагування залу
deleted_at	TIMESTAMP		Дата видалення залу

Відношення branch_room – це зв’язок між філіями та залами. Один зал може бути закріплений тільки за однією філією, в той час як за однією філією може бути закріплено декілька залів.

Сутність Meter – це лічильник (будь то газ, електроенергія, вода, тощо), який закріплений за певним залом (табл. 2.6). У одного залу може бути декілька лічильників.

Таблиця 2.6 – Лічильники

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор лічильника
name	VARCHAR (255)		Назва лічильника
room_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор залу, за яким закріплено лічильник
created_at	TIMESTAMP		Дата створення лічильника
updated_at	TIMESTAMP		Дата останнього

Продовження табл. 2.6

Назва поля	Тип даних	Властивість	Опис
			редагування лічильника
deleted_at	TIMESTAMP		Дата видалення лічильника

Сутність MeterReading – це показник певного лічильника (табл. 2.7).

Таблиця 2.7 – Показники лічильників

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор показника лічильника
value	BIGINT		Значення показника лічильника
image	VARCHAR (255)		Фотографія показника лічильника
meter_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор лічильника, за яким закріплено показник
user_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор користувача, котрий вніс показник лічильника

Продовження табл. 2.7

Назва поля	Тип даних	Властивість	Опис
created_at	TIMESTAMP		Дата створення показника лічильника
updated_at	TIMESTAMP		Дата останнього редагування показника лічильника
deleted_at	TIMESTAMP		Дата видалення показника лічильника

Сутність ServiceType – це тип сервісу, який об’єднує певні сервіси за певними критеріями (табл. 2.8).

Таблиця 2.8 – Типи сервісу

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор типу сервісу
name	VARCHAR (255)		Назва типу сервісу
background_color	VARCHAR (7)		Колір фону, який використовується при відображенні подій цього типу сервісу на розкладі
text_color	VARCHAR (7)		Колір тексту,

Продовження табл. 2.8

Назва поля	Тип даних	Властивість	Опис
			який використовується при відображенні подій цього типу сервісу на розкладі
created_at	TIMESTAMP		Дата створення лічильника
updated_at	TIMESTAMP		Дата останнього редагування лічильника
deleted_at	TIMESTAMP		Дата видалення лічильника

Сутність Service – це сервіс, за яким проходять певні події (табл. 2.9).

Таблиця 2.9 – Сервіси

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор сервісу
name	VARCHAR (255)		Назва сервісу
created_at	TIMESTAMP		Дата створення сервісу
updated_at	TIMESTAMP		Дата останнього редагування сервісу
deleted_at	TIMESTAMP		Дата видалення сервісу

Відношення service_service_type – це зв’язок між сервісом та типом сервісу. Один сервіс може бути закріплено за декількома типами сервісів, так само як і за одним типом сервісу може бути закріплено декілька сервісів.

Сутність Group – це група, до якої закріплено певного працівника та певний сервіс (табл. 2.10).

Таблиця 2.10 – Групи

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор групи
worker_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор працівника, якого закріплено до групи
service_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор сервісу, який закріплено до групи
created_at	TIMESTAMP		Дата створення групи
updated_at	TIMESTAMP		Дата останнього редагування групи
deleted_at	TIMESTAMP		Дата видалення групи

Сутність GroupSchedule – це розклад події, закріпленої за певною групою (табл. 2.11).

Таблиця 2.11 – Розклад подій

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор розкладу події
group_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор групи, за якою закріплено розклад події
room_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор залу, за котрим закріплено розклад події
day_number	SMALLINT unsigned		День тижня події (від 1 до 7, де 1 – це понеділок, 7 - неділя)
start	TIME		Час початку події
end	TIME		Час кінця події
created_at	TIMESTAMP		Дата створення показника лічильника
updated_at	TIMESTAMP		Дата останнього редагування показника лічильника

Сутність SubscriptionType – це тип абонементу, за яким може ходити клієнт (див. табл. 2.12). За замовчуванням, у системі розроблено три типи абонементів:

– стандартний;

– безлімітний (абонемент не має обмежень щодо кількості відвідуваних

занять);

– free way (за цим абонементом можна ходити на будь-які групи).

Таблиця 2.12 – Типи абонементів

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор типу абонементу
name	VARCHAR (255)		Назва тип абонементу
internal_class	VARCHAR (255)		Клас тип абонементу (Standard, Unlimited або FreeWay)
created_at	TIMESTAMP		Дата створення типу абонементу
updated_at	TIMESTAMP		Дата останнього редагування типу абонементу

Сутність SubscriptionPricingThreshold – це ціновий поріг, закріплений за кожним типом абонементу, виконує функцію ціноутворення, в залежності від обраної кількості занять абонементу (табл. 2.13).

Таблиця 2.13 – Цінові пороги

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор цінового порогу
subscription_type_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор

Продовження табл. 2.13

Назва поля	Тип даних	Властивість	Опис
			типу абонемента, за яким закріплено ціновий поріг
items_quantity	SMALLINT unsigned		Кількість занять цінового порогу
price_per_item	INT		Ціна кожного заняття цінового порогу
created_at	TIMESTAMP		Дата створення цінового порогу
updated_at	TIMESTAMP		Дата останнього редагування цінового порогу
deleted_at	TIMESTAMP		Дата видалення цінового порогу

Сутність Subscription – це абонемент, який дозволяє клієнту відвідувати заняття за певними характеристиками (табл. 2.14).

Таблиця 2.14 – Абонементи

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор абонементу
key	VARCHAR (20)		Ключ абонементу,

Продовження табл. 2.14

Назва поля	Тип даних	Властивість	Опис
			утворений від дати та часу створення абонементу
subscription_type_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор типу абонементу, закріплений за абонементом
service_type_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор типу сервісу, за яким закріплено абонемент
user_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор клієнта, котрий придбав абонемент
user_balance_payment_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор оплати клієнтом за абонемент
price	BIGINT		Ціна за абонемент
activated_at	DATETIME		Дата актівації абонементу
expired_at	DATETIME		Дата деактивації абонементу

Кінець табл. 2.14

Назва поля	Тип даних	Властивість	Опис
allowed_visits_count	SMALLINT		Кількість дозволених відвідувань
remained_visits_count	SMALLINT		Кількість залишившихся відвідувань
extended_at	DATETIME		Дата продовження термін дії абонемента
extended_by_user_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор користувача, що продовжив термін дії абонемента
frozen_at	DATETIME		Дата заморозки абонементу
created_at	TIMESTAMP		Дата створення абонементу
updated_at	TIMESTAMP		Дата останнього редагування абонементу
deleted_at	TIMESTAMP		Дата видалення абонементу

Відношення subscription_worker – це зв’язок між абонементом та працівником. За одним абонементом може бути закріплено декілька працівників, так само, як і один працівник може бути закріплений за декількома

абонементами.

Відношення subscription_service – це зв’язок між абонементом та сервісом. За одним абонементом може бути закріплено декілька сервісів, так само, як і один сервіс може бути закріплений за декількома абонементами.

Відношення group_customer – це зв’язок між групою, клієнтом та його абонементом. За однією групою може бути закріплено декілька клієнтів з їх абонементами, так само, як абонемент клієнта може бути закріплений до декількох груп.

Сутність Individual – це індивідуальна подія, котру може відвідати лише один клієнт (табл. 2.15).

Таблиця 2.15 – Індивідуальні події

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор індивідуальної події
worker_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор працівника, закріпленого за індивідуальною подією
customer_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор клієнта, закріпленого за індивідуальною подією
subscription_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор абонементу, закріпленого за індивідуальною

Продовження табл. 2.15

Назва поля	Тип даних	Властивість	Опис
			подією
created_at	TIMESTAMP		Дата створення індивідуального івенту
updated_at	TIMESTAMP		Дата останнього редагування індивідуального івенту
deleted_at	TIMESTAMP		Дата видалення індивідуального івенту

Сутність ScheduleEvent – це подія на розкладі, що закріплена за певним часом та залом (табл. 2.16). Вона може бути декількох видів:

– групова подія (за неї закріплена певна група);
– індивідуальна подія;
– кастомна подія (її може створити будь-який користувач, котрий має на це дозвіл).

Таблиця 2.16 – Події на розкладі

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор події на розкладі
title	VARCHAR (255)		Назва події на розкладі
description	VARCHAR (255)		Опис події на

Продовження табл. 2.16

Назва поля	Тип даних	Властивість	Опис
			розкладі
start	DATETIME		Дата початку події на розкладі
end	DATETIME		Дата кінця події на розкладі
room_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор залу, за котрим закріплена подія на розкладі
service_type_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор типу сервісу, який закріплено за подією на розкладі
service_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор сервісу, який закріплено за подією на розкладі
worker_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор працівника, закріпленого за подією на розкладі
eventable_type	VARCHAR (255)		Тип події на

Кінець табл. 2.16

Назва поля	Тип даних	Властивість	Опис
			розкладі (груповий, індивідуальний або кастомний)
eventable_id	BIGINT unsigned		Ідентифікатор типу події
created_by_user_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор користувача, що створив подію на розкладі
subscription_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор абонементу, закріпленого за подією на розкладі
created_at	TIMESTAMP		Дата створення події на розкладі
updated_at	TIMESTAMP		Дата останнього редагування події на розкладі
deleted_at	TIMESTAMP		Дата видалення події на розкладі

Сутність VisitMark – це відвідування клієнтом певної події за певним абонементом (табл. 2.17).

Таблиця 2.17 – Відвідування подій

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор індивідуальної події
schedule_event_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор події на розкладі, що відвідав клієнт
created_by_user_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор користувача, котрий відмітив клієнта
subscription_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор абонементу, за яким було відвідувано подію
created_at	TIMESTAMP		Дата створення відвідування події
updated_at	TIMESTAMP		Дата останнього редагування відвідування події
deleted_at	TIMESTAMP		Дата видалення відвідування події

Сутність UserBalancePaymentMethod – це тип оплати, котрим клієнт розраховується за абонемент (див. табл. 2.18). За замовчуванням, тип оплати може бути картою та готівкою.

Таблиця 2.18 – Типи оплати

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор типа оплати
name	VARCHAR (255)		Назва
created_at	TIMESTAMP		Дата створення типу оплати
updated_at	TIMESTAMP		Дата останнього редагування типу оплати
deleted_at	TIMESTAMP		Дата видалення типу оплати

Сутність UserBalancePayment – це оплата за певний абонемент з балансу клієнта, який він поповнює перед купівлею цього абонементу (табл. 2.19).

Таблиця 2.19 – Оплати з балансу

Назва поля	Тип даних	Властивість	Опис
id	BIGINT unsigned	Primary key (PK), AutoIncrement	Ідентифікатор оплати з балансу
amount	BIGINT		Сума оплати з балансу
description	VARCHAR (255)		Опис оплати з балансу
user_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор клієнта, що зробив оплату з балансу

Продовження табл. 2.19

Назва поля	Тип даних	Властивість	Опис
user_balance_payment_method_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор типу оплати, котрим було проведено оплату з балансу
payment_id	BIGINT unsigned	Foreign key (FK)	Ідентифікатор оплати
created_at	TIMESTAMP		Дата створення оплати з балансу
updated_at	TIMESTAMP		Дата останнього редагування оплати з балансу
deleted_at	TIMESTAMP		Дата видалення оплати з балансу

3 РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ФІТНЕС ЦЕНТРОМ

3.1 Реалізація основної сторінки системи Dashboard

Dashboard (рис. 3.1) є основною сторінкою цієї системи тому що саме у ній відбувається більшість взаємодій користувача з даними.

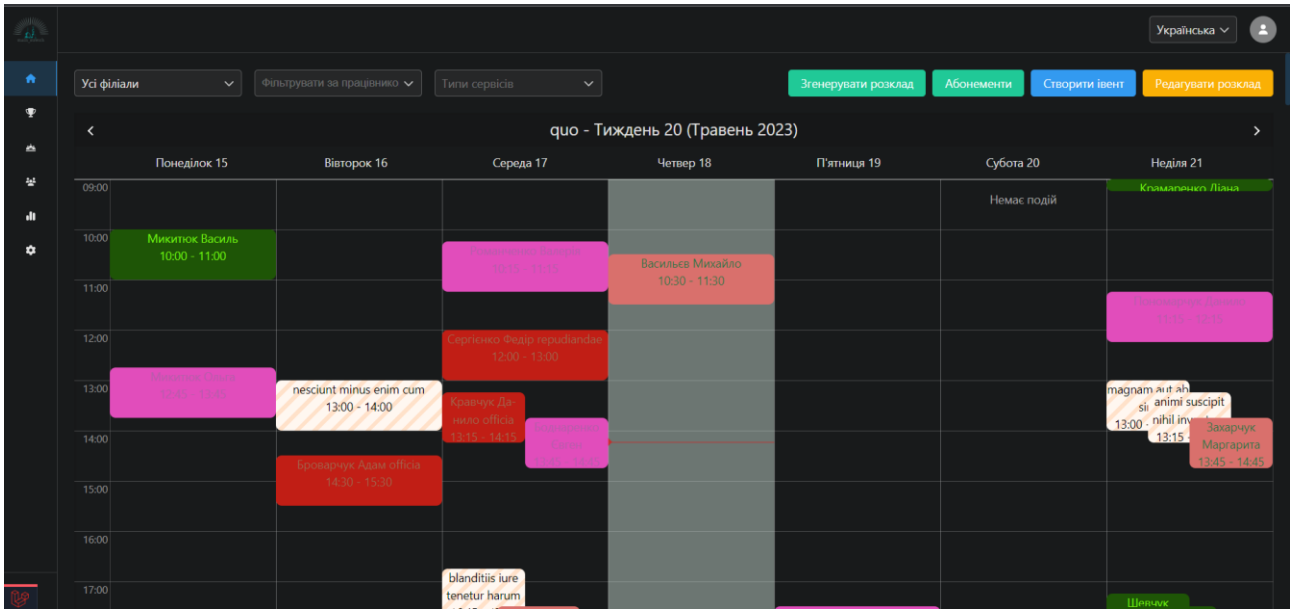

Рисунок 3.1 – Зовнішній вигляд сторінки Dashboard

Основні функції, які доступні користувачеві на сторінці Dashboard:

– перегляд, фільтрація, редагування та генерування розкладу;
– перегляд, редагування певного івенту, продаж абонементів до певного івента, створення групових та кастомних івентів;
– перегляд, редагування та продаж певного абонементу.

Програмну реалізацію основного функціоналу головної сторінки Dashboard продемонстровано в додатку Б.

3.1.1 Взаємодія користувача з розкладом

На розкладі можна побачити, в якому залі коли відбуваються ті чи інші івенти. Івент – це групова, індивідуальна або кастомна подія, яка закріплена за певним залом та працівником (у випадку якщо подія не кастомна).

Для зручності користуванням розкладу, є можливість фільтрувати його за певними критеріями, а саме:

– фільтрація за філіалом;
– фільтрація за працівником;
– фільтрація за типами сервісів (груповий, індивідуальний або кастомний);
– фільтрація за тижнем.

Розглянемо рисунок 3.2. На розкладі можна пересувати івент на іншу дату (в рамках обраного тижня та залу), змінювати тривалість івента та видаляти його з розкладу.

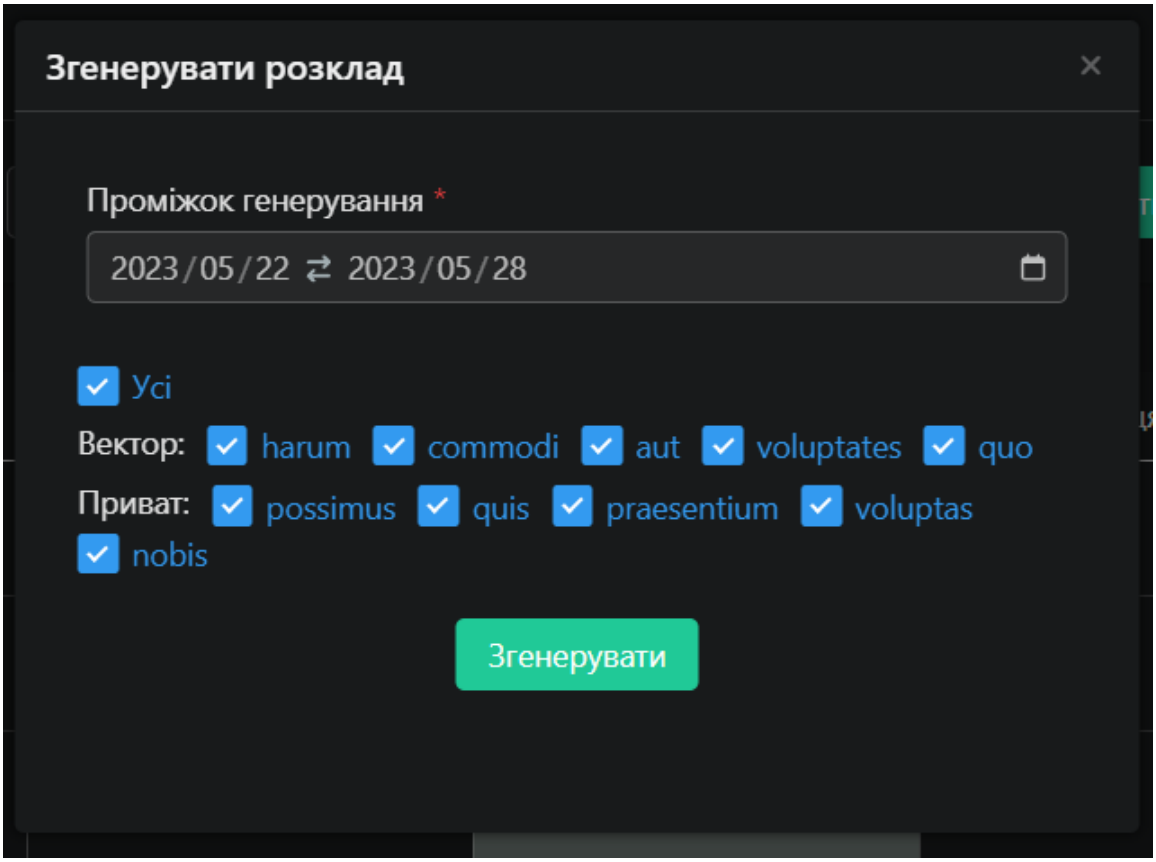

Рисунок 3.2 – Модальне вікно «Генерування розкладу»

Ще можна відмітити функціонал генерування розкладу, тобто користувач, у якого є дозвіл на цю дію у системі, може автоматично розставити групові івенти, в залежності від створених груп та їх дефолтних розкладів у системі, обравши певні зали та проміжок днів, куди треба розставити події.

3.1.2 Взаємодія користувача з івентами

Кожен івент на розкладі містить в собі певну інформацію, яку можна подивитись в модалці «Інформація про івент» (рис. 3.3).

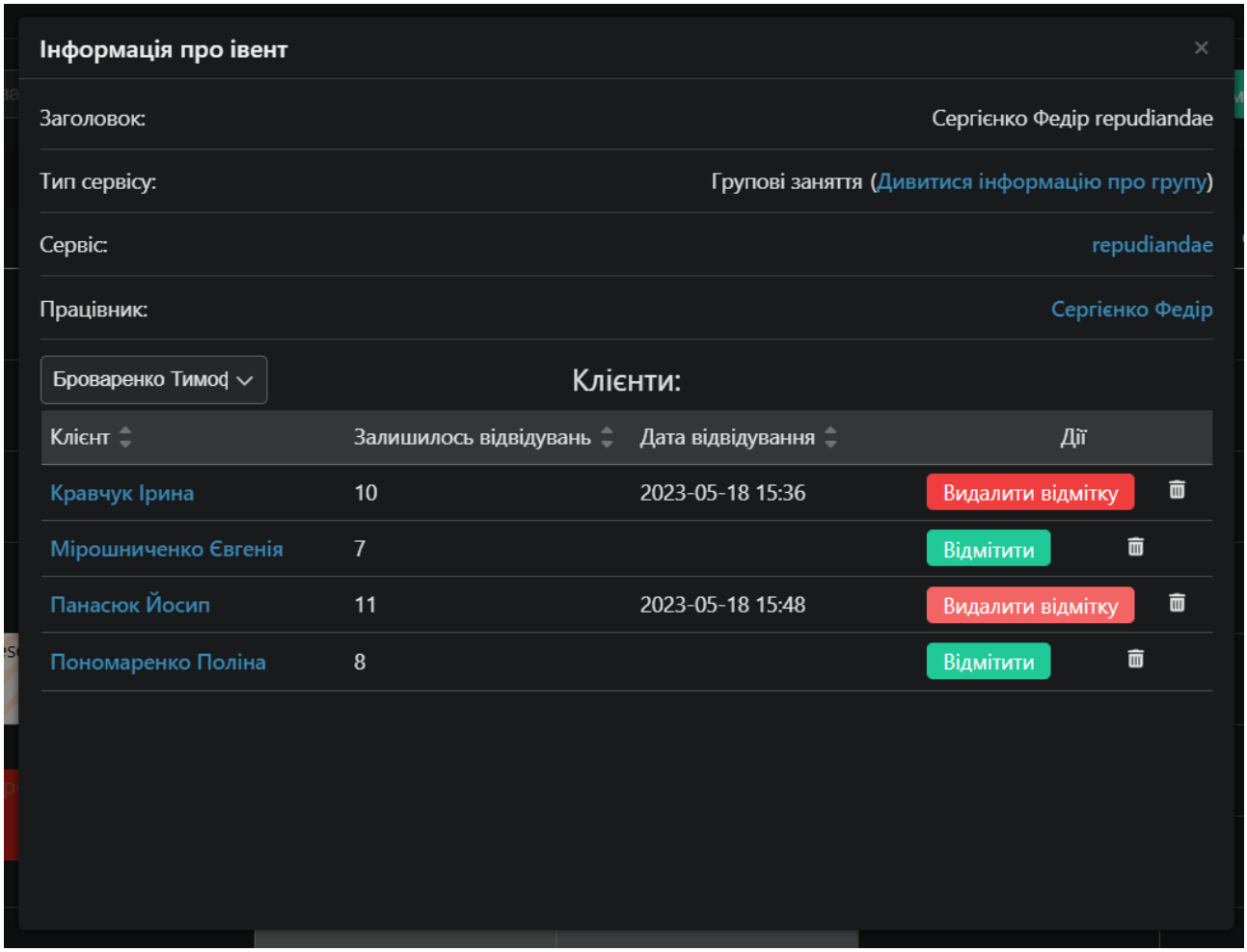

Рисунок 3.3 – Модальне вікно «Інформація про івент»

В залежності від типу події (групова, індивідуальна або кастомна), можна отримати різну інформацію. В основному це інформація про сервіс, про працівника, опис даного івенту. Але, наприклад, у групової події, на відміну від інших, є список закріплених за нею клієнтів, у якому можна подивитись скільки залишилось занять у кожного клієнта, можливість відмітити або відмінити відвідування клієнтів до цього івенту, також можна додавати та видаляти клієнтів зі списку.

Якщо у користувача є можливість у системі редагувати івенти, то він може за потребою змінювати певні дані івентів (рис. 3.4).

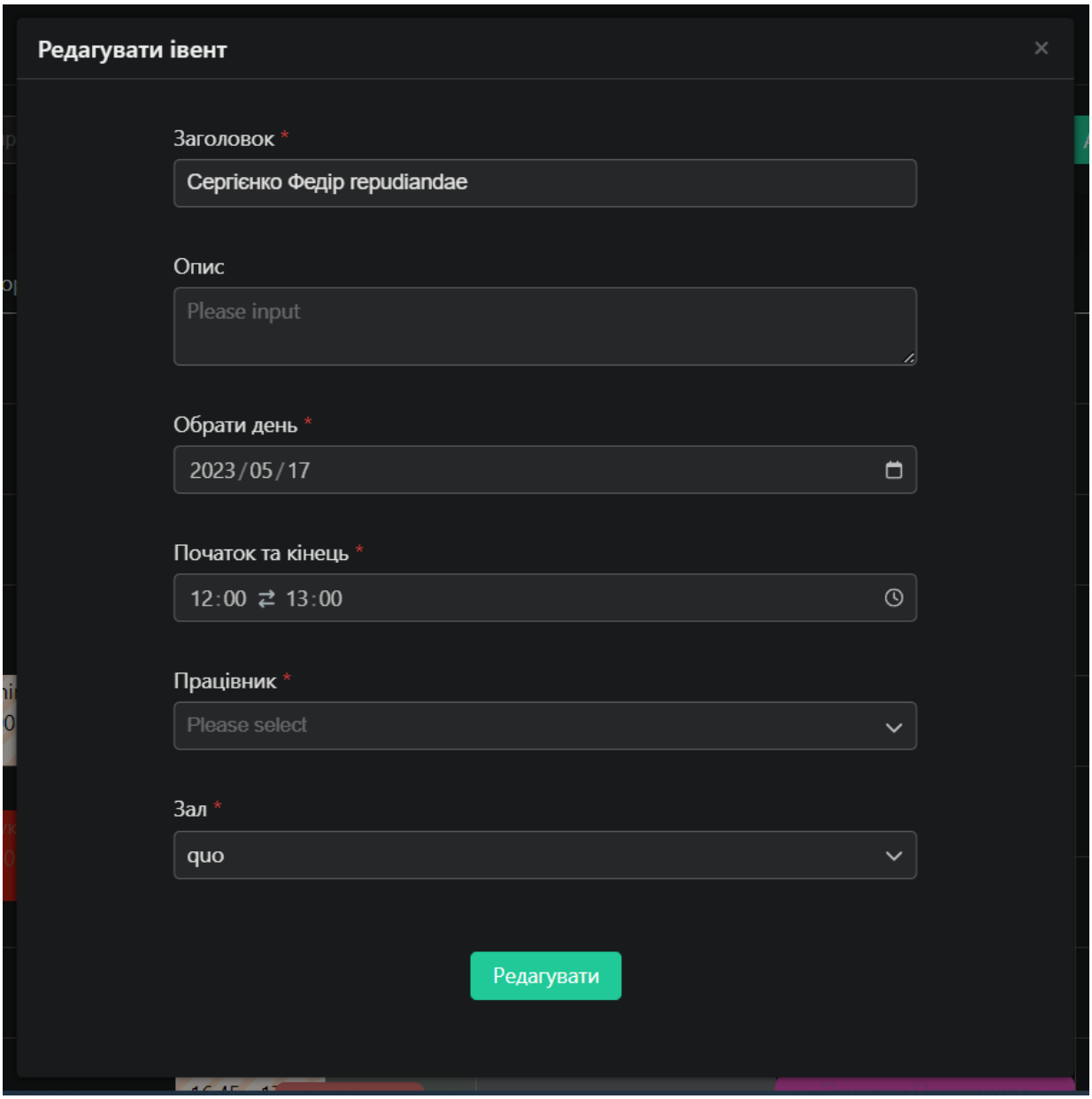

Рисунок 3.4 – Модальне вікно «Редагування івенту»

Ще була реалізована можливість створення кастомних та групових івентів власноруч. Якщо для створення кастомного івенту потрібно лише натиснути на кнопку відкриття модалки «Створення кастомного івенту» (рис. 3.5), то, щоб створити групову подію, потрібно обрати групу з розкладу та натиснути «Створити груповий івент» для відкриття модалки «Створення групового івенту» (див. рис. 3.6).

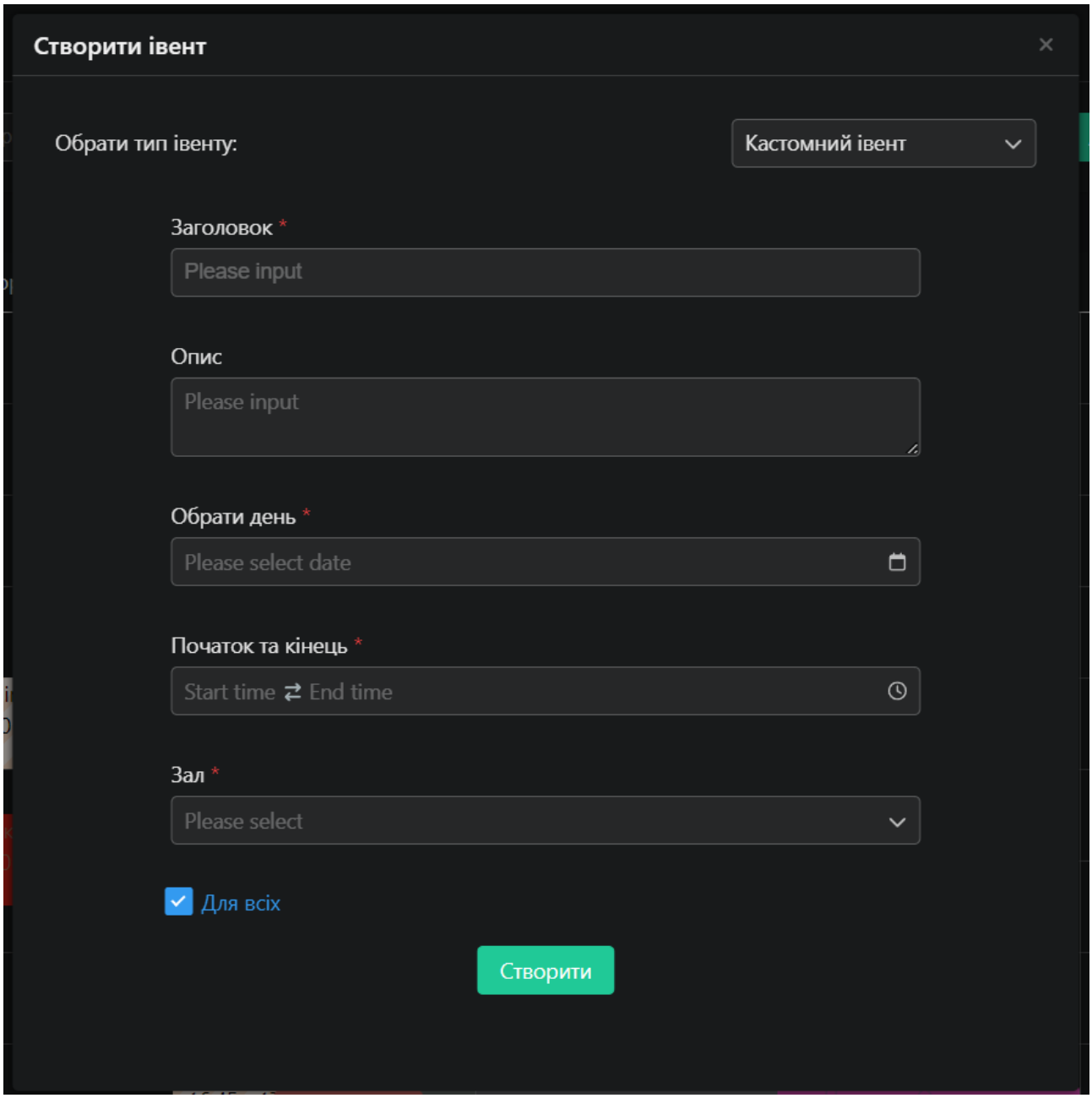

Рисунок 3.5 – Модальне вікно «Створення кастомного івенту»

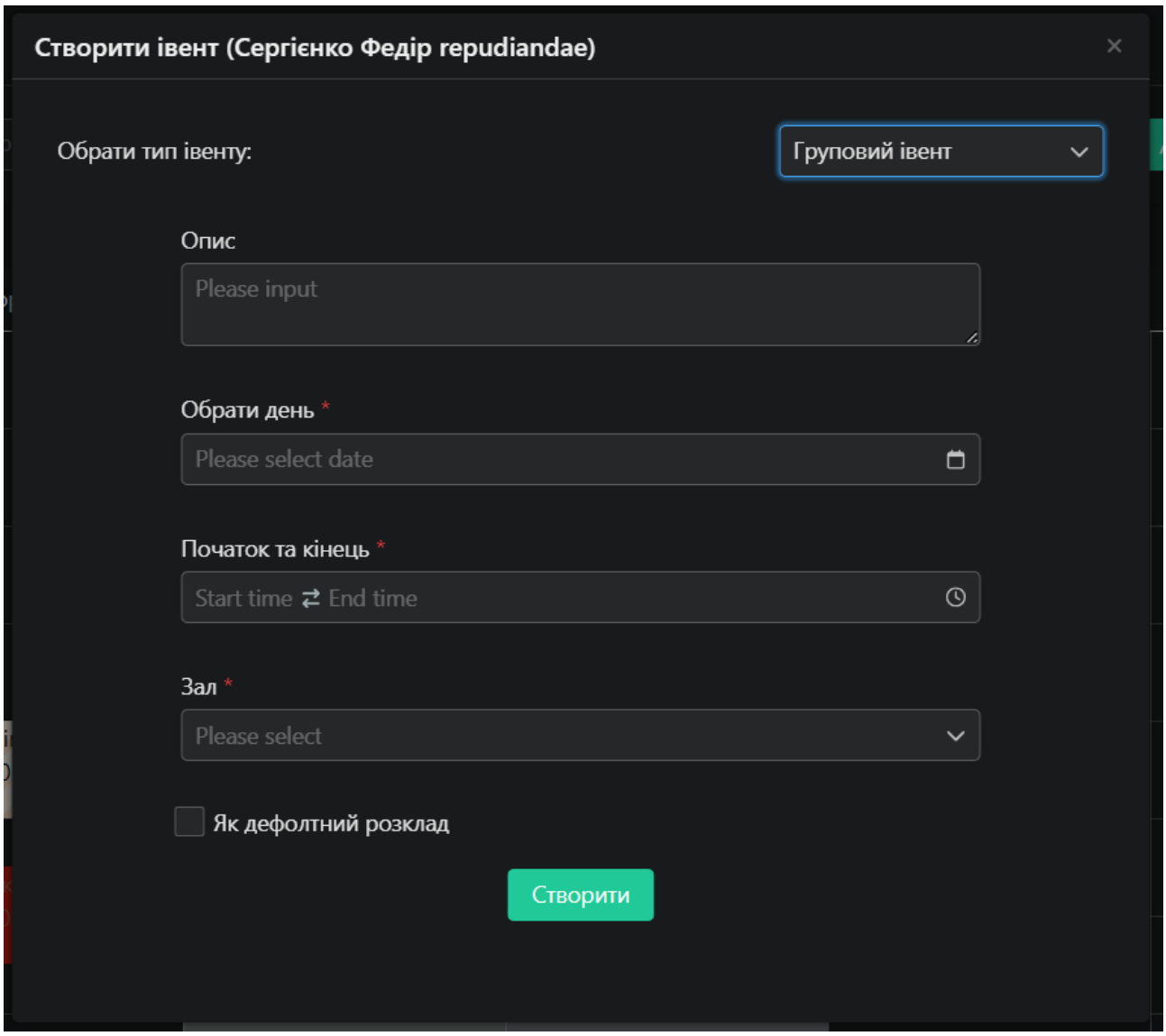

Рисунок 3.6 – Модальне вікно «Створення групового івенту»

3.1.3 Взаємодія користувача з абонементами

На сторінці Dashboard користувач має можливість фіксувати продажі абонементів. Для цього була розроблена окрема модалка «Продаж абонементів» (див. рис. 3.7), до якої можна потрапити або через натискання на кнопку «Абонементи» або переходячи з групового івента на розкладі (у такому випадку, в модалці підставляться дані про цю групу).

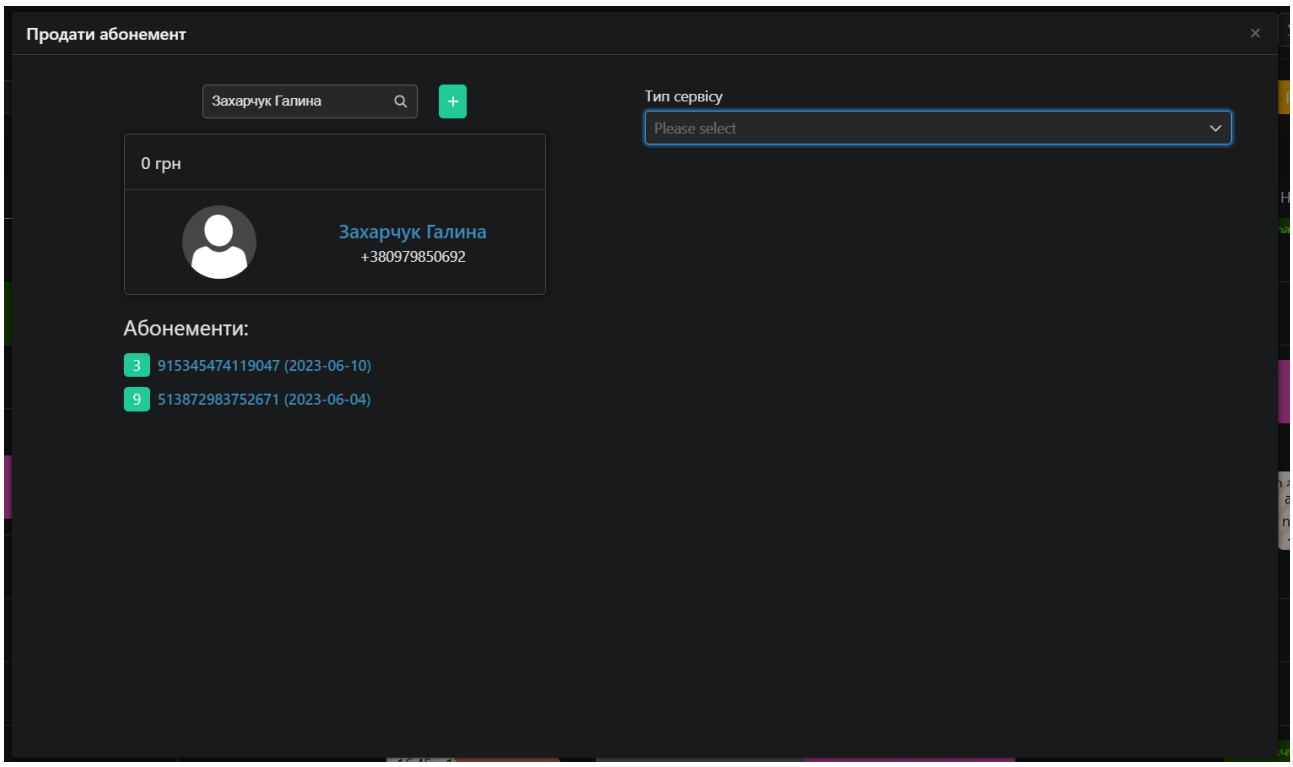

Рисунок 3.7 – Модальне вікно «Продаж абонементу»

Для того, щоб продати абонемент, треба:

1) обрати користувача (або створити в іншій модалці (див. рис. 3.8), після чого створений клієнт автоматично підставиться в основну модалку);
2) обрати тип сервісу (груповий чи індивідуальний);
3) обрати тип абонементу (стандартний, безлімітний чи free way);
4) в залежності від обраного типу абонементу, потрібно буде обрати кількість відвідувань (у випадку стандартного та free way типів абонементу), сервіс(и) та працівника(ів) (якщо обрано стандарт чи безліміт), та ціну абонемента (завжди);
5) якщо було обрано абонемент на індивідуальні заняття, то перед продажем можна встановити розклад івентів (див. рис. 3.9);
6) підтвердити продаж, обравши тип оплати (бронювання, карткою, готівкою чи мультиоплатою (див. рис. 3.10).

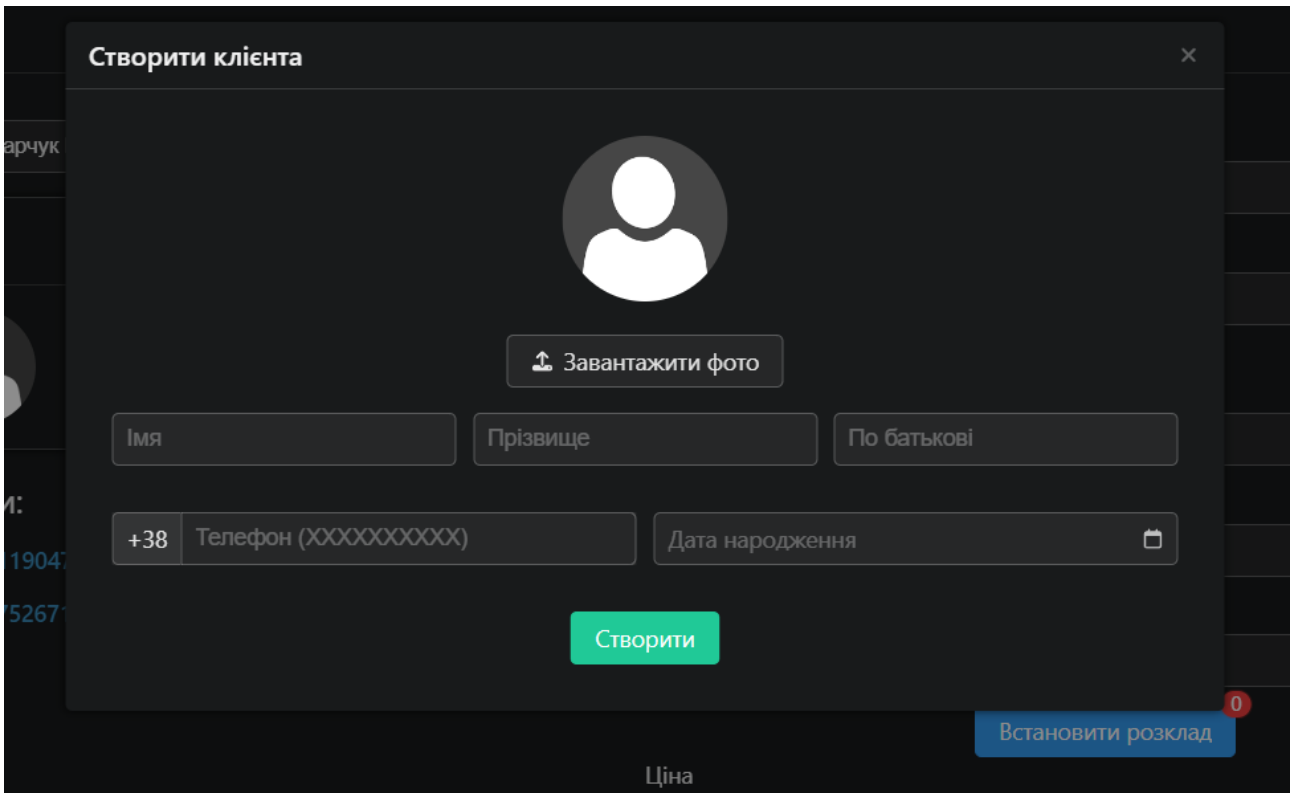

Рисунок 3.8 – Модальне вікно «Швидке створювання клієнта»

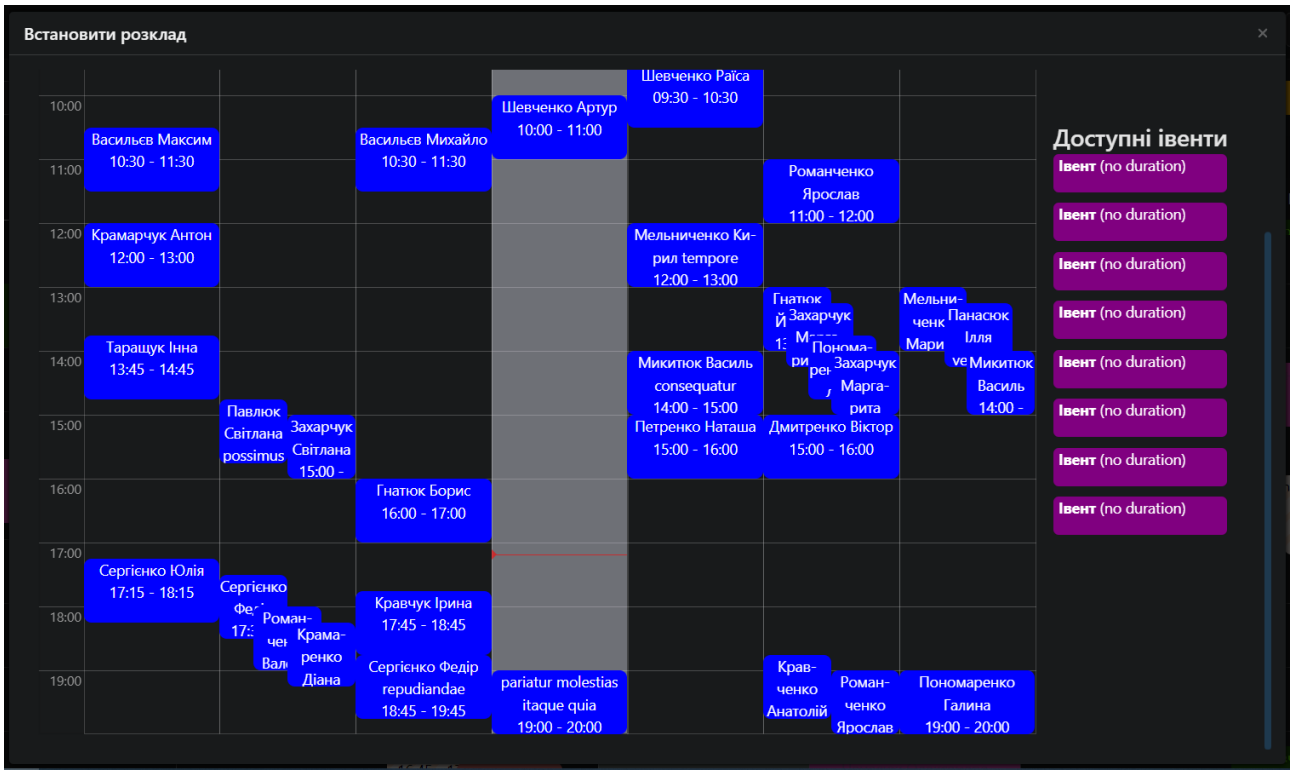

Рисунок 3.9 – Модальне вікно «Встановлення розкладу»

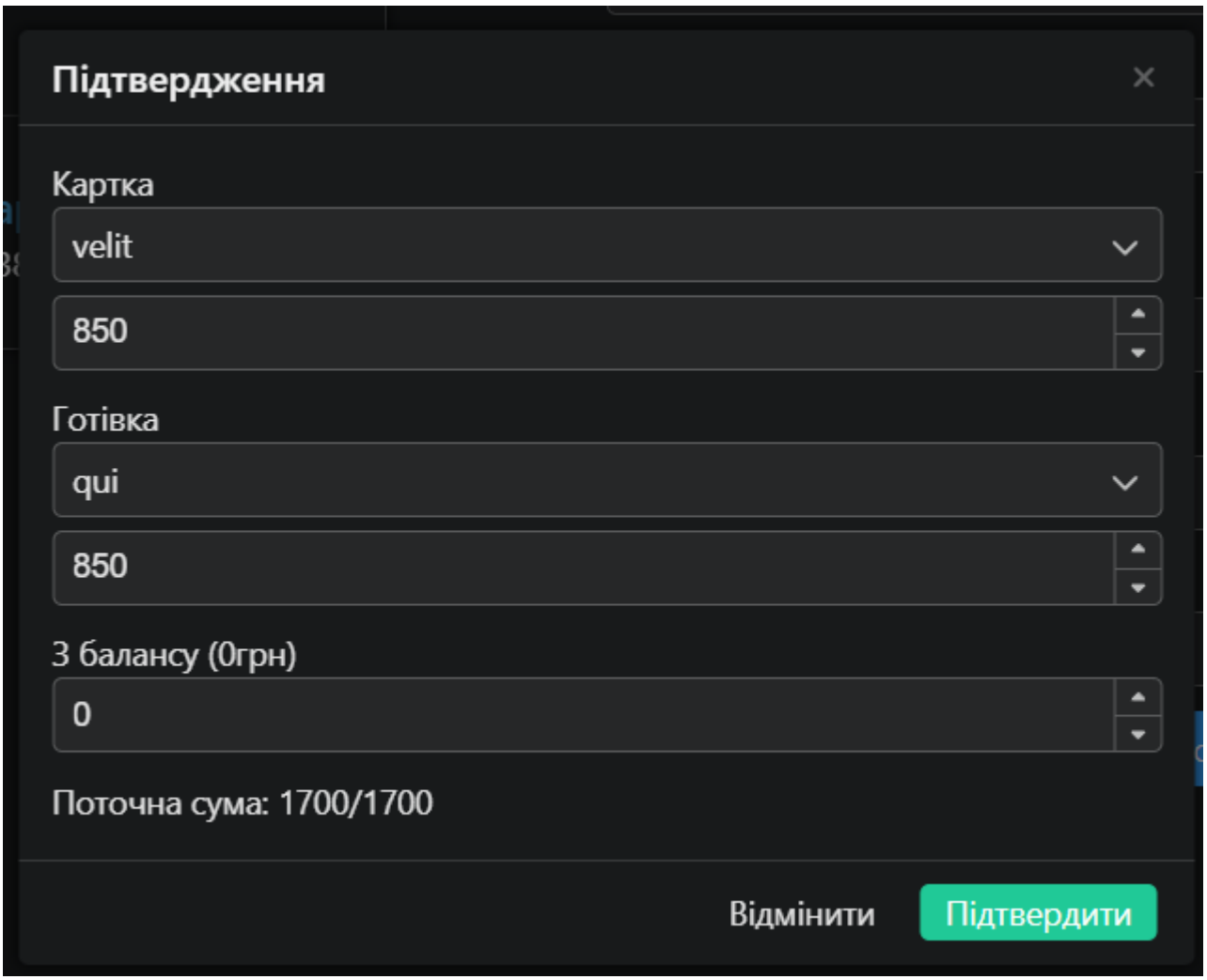

Рисунок 3.10 – Модальне вікно «Підтвердження продажу абонемента» на прикладі мультіоплати

Після продажу абонемента, у користувача є можливість переглядати детальну інформацію про цей абонемент (див. рис. 3.11), довстановлювати індивідуальні івенти та, за потребою, продовжувати термін закінчення абонемента на 7 днів.

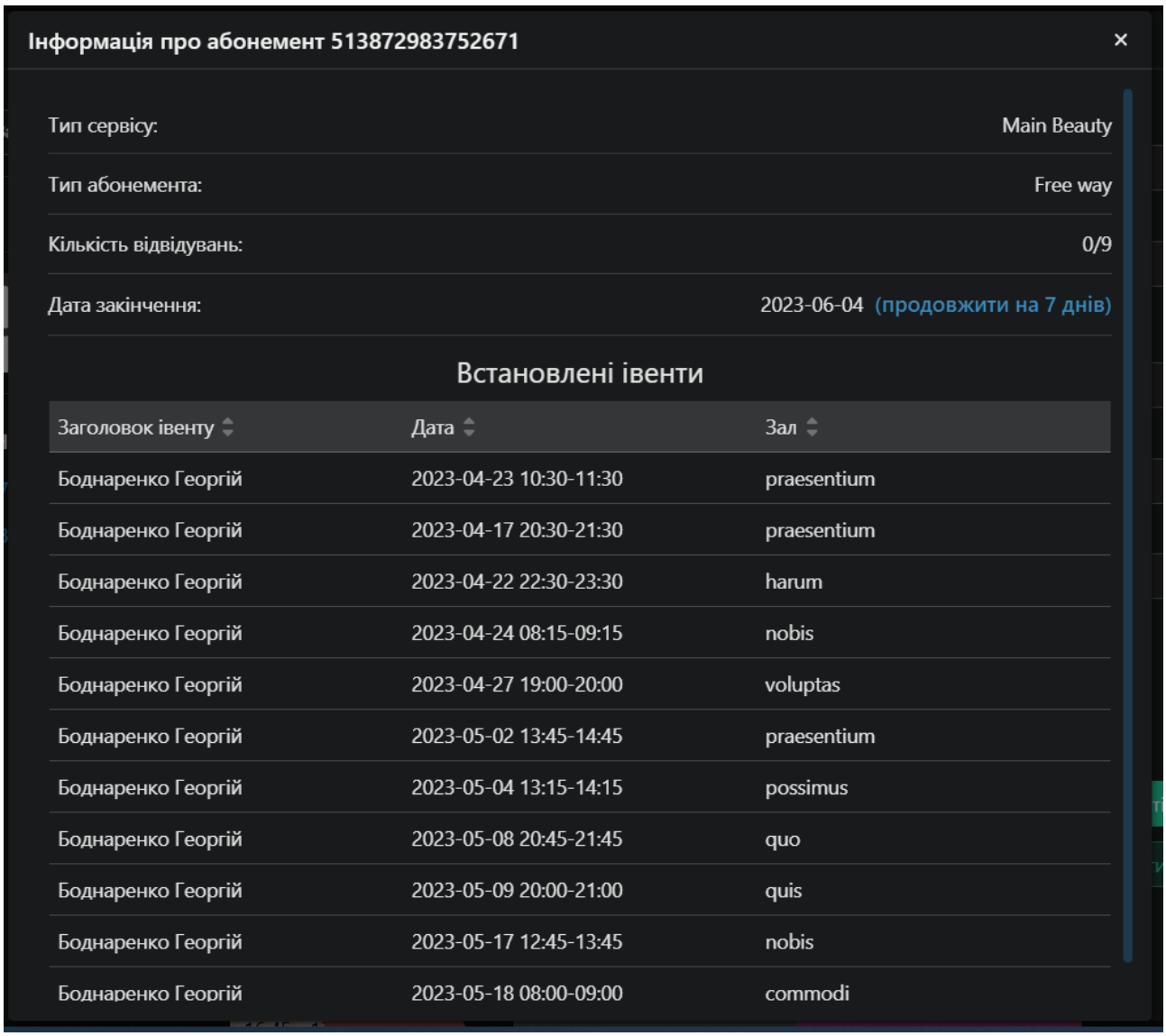

Рисунок 3.11 – Модальне вікно «Перегляд інформації про абонемент»

3.2 Висновки до розділу 3

Отже, було описано та продемонстровано взаємодію користувачей з розкладом, івентами та абонементами, продемонстрований результат розробки функціоналу продажу абонементів засобами фреймворку Vue.js, мовою програмування Javascript, мовою розмітки HTML та мовою стилів CSS.

ВИСНОВКИ

В роботі наведено короткий огляд сучасних засобів веб-розробки, зокрема мова програмування Javascript та її фреймворк Vue.js, становий управлінський пакет Pinia, розглянуто патерн MVVM. Описано предметну область системи управління фінтес-центрами, включаючи всі сутності, їх атрибути та зв’язки між цими сутностями. Розроблено ER-модель та її програмну реалізацію з використанням фреймворку Vue.js. Також були продемонстровані основні операції на головній сторінці Dashboard.

Цей фреймворк був спеціально розроблений для легкої масштабованості та адаптації будь-якого веб-додатка. Крім того, варто відзначити, що додаток, побудований на Vue.js, має реактивну природу, що дозволяє йому гнучко адаптуватись до змін шляхом декларативного підходу.

Таким чином, Vue.js може бути використаний в практично будь-якому веб-проєкті, оскільки він надає необхідний набір функціональності вже після свого встановлення. За допомогою Vue.js можна створювати сайти, розробляти веб-додаткові інтерфейси, взаємодіяти з сервером, реагувати на зміни, що відбуваються під час взаємодії користувача з системою, і навіть оновлювати контент сторінки без перезавантаження всієї сторінки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Javascript. URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата звернення: 03.03.2023).
2. Typescript. URL: https://highload.today/typescript (дата звернення: 04.03.2023).
3. NPM. URL: https://uk.wikipedia.org/wiki/Npm (дата звернення: 14.03.2023).
4. Axios docs. URL: https://axios-http.com/uk/docs/intro (дата звернення: 04.03.2023).
5. HTML. URL: http://htmlbook.ru/html (дата звернення: 12.02.2023).
6. CSS. URL: https://developer.mozilla.org/ru/docs/Web/CSS (дата звернення: 16.02.2023).
7. Vexip docs. URL: https://www.vexipui.com/en-US/guides/setup (дата звернення: 05.03.2023).
8. Vue.js. URL: https://github.com/vuejs/vue (дата звернення: 04.03.2023).
9. Vue-Cal. URL: https://github.com/vuejs/vue (дата звернення: 07.03.2023).
10. Pinia. URL: https://pinia.vuejs.org/ (дата звернення: 08.03.2023).
11. Vite docs. URL: https://vite-docs-ru.vercel.app/guide/ (дата звернення: 09.03.2023).
12. MVVM. URL: https://uk.wikipedia.org/wiki/Model-View-ViewModel (дата звернення: 13.03.2023).
13. Вступ – Про систему контролю версій. URL: https://git-scm.com/book/uk/v2/Вступ-Про-систему-контролю-версій (дата звернення: 01.04.2023).
14. React docs. URL: https://ru.legacy.reactjs.org/docs/getting-started.html (дата звернення: 01.04.2023).
15. Angular docs. URL: https://angular.io/docs (дата звернення: 01.04.2023).
16. Діаграма діяльності. URL: https://uk.wikipedia.org/wiki/Діаграма_діяльності (дата звернення: 08.04.2023).
17. Діаграма розгортання. URL:

https://uk.wikipedia.org/wiki/Діаграма_розгортання (дата звернення: 08.04.2023).

18. Модель «сутність-зв’язок». URL: https://uk.wikipedia.org/wiki/Модель_сутність_зв’язок (дата звернення: 08.04.2023).

ДОДАТОК А

ER-діаграма

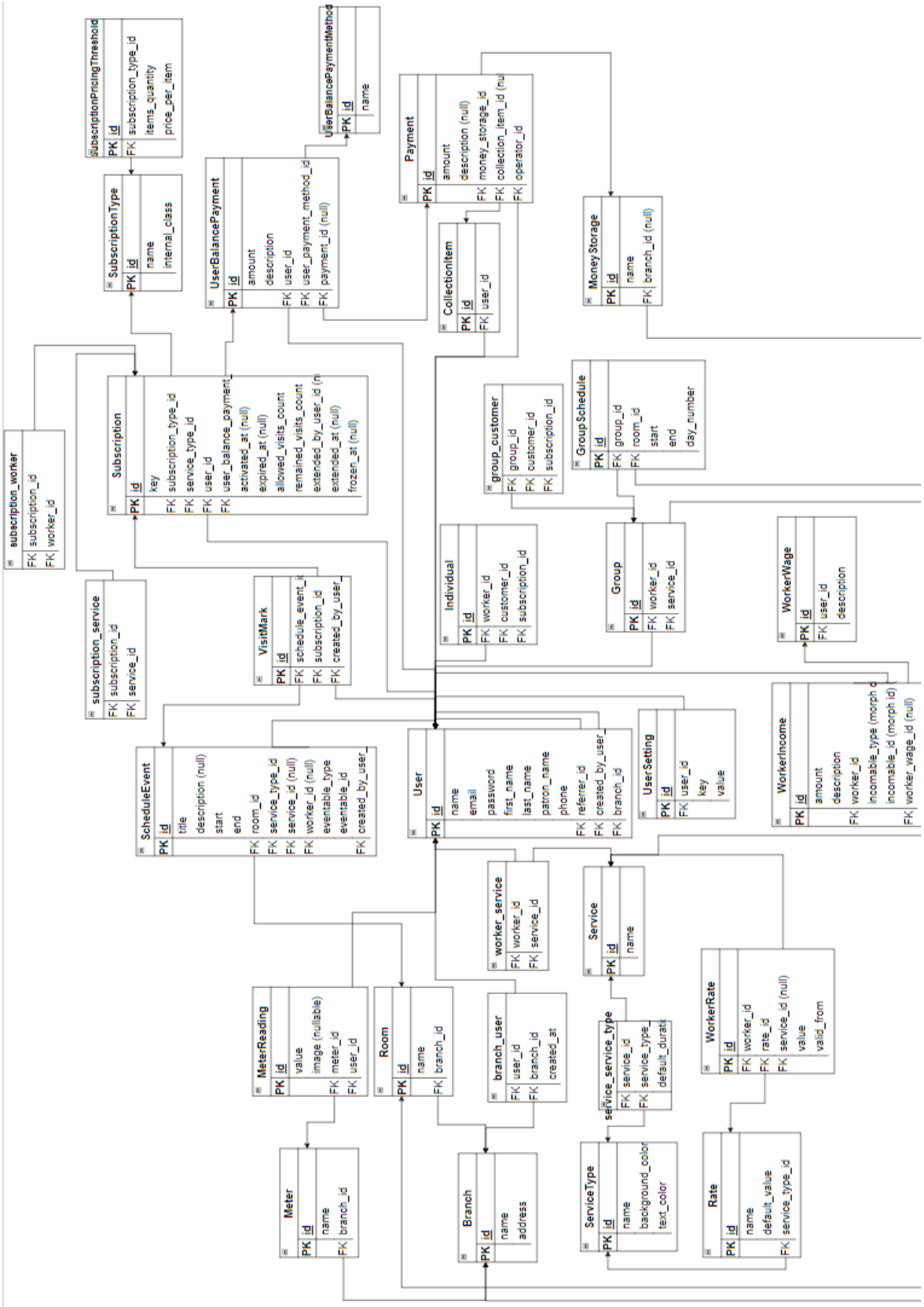

Рисунок А.1 – ER-діаграма

ДОДАТОК Б

Програмна реалізація головної сторінки системи Dashboard

У цьому додатку продемонстровано програмний код наступних функціоналів:

– генерування розкладу (дод. Б.1);

– продаж абонементу (дод. Б.2);

– створення івентів (дод. Б.3).

Б.1 Функції генерування розкладу

```
function generateSchedule(){
   generateScheduleFormModel.from = generateScheduleFormModel.from_to[0]
   generateScheduleFormModel.to = generateScheduleFormModel.from_to[1]
   schedulesEventsStore.generateSchedule(generateScheduleFormModel).then(res =>
{
      Notice.config({ placement: 'bottom-right' })
      Notice.success(t('Успішно!'))
      isActiveGenerateScheduleModal.value = false
      generateScheduleFormModel.from = ''
      generateScheduleFormModel.from_to = ''
      generateScheduleFormModel.room_ids = []
      generateScheduleFormModel.to = ''
      getSchedules()
   })
}

function getSchedules(without_effects:boolean=false){
   let exclude = without_effects ? {
```

```
        exclude_request: true,
        exclude_response: true
    } : {}

    let filters = {
        ...schedules_filters.value,
        ...exclude
    }

    schedulesEventsStore.getScheduleEvents({params: filters}).then(res => {
        schedules.value = res.data?.data
        schedules.value.forEach(room => {
            room.schedule_events.forEach(item => {
                item.resizable = isAllowScheduleEdit.value
                item.draggable = isAllowScheduleEdit.value
                item.deletable = isAllowScheduleEdit.value


            })
        })
    })
}
```

Б.2 Функції продажу абонементу

```
function sellSubscription(){
    if (isAllowConfirmSellSubscription()){
        let subscriptions_to_sell = JSON.parse(JSON.stringify(subscription_cart));

        subscriptions_to_sell.subscriptions.forEach(subscription => {
            if(typeof subscription.service_ids == 'number'){        // convert id to array of
```

```
id
            subscription.service_ids = [subscription.service_ids]
        }

        if(typeof subscription.worker_ids == 'number'){     // convert id to array of id
            subscription.worker_ids = [subscription.worker_ids]
        }

        if(subscription.events.length){
            subscription.events.forEach(event => {
                event.start = getFormatDate(event.start, true)
                event.end = getFormatDate(event.end, true)
            })
        }
    })

    subscriptionsStore.createSubscription(subscriptions_to_sell).then(res => {
        Notice.config({ placement: 'bottom-right' })
        Notice.success(t('Успішно!'))

        res.data.data.forEach(item => {
            Notice.info({ title: `№ ${t('Абонемент')} ${item.key}`, color: true,
duration: 0, closable: true })
        })

        resetSubscriptionDetails()
        setUserInSubscriptionSellModal(customer_in_subscription_sell_modal.value.
info.id)

        isActiveSellSubscriptionConfirmModal.value = false
```

```
        })
    }
}

function isAllowConfirmSellSubscription():boolean{
    if(subscription_cart.payment_type == 'single' &&
!subscription_cart.payment_parts[0].money_storage_id){
        Notice.config({ placement: 'bottom-right' })
        Notice.error(t('Встановіть сховище грошей!'))
        return false
    }

    if(subscription_cart.payment_type == 'multiple'){
        let is_return = false

        if(!subscription_cart.payment_parts[userBalancePaymentMethodsStore.UserBal
ancePaymentMethods.Card - 1].money_storage_id &&
subscription_cart.payment_parts[userBalancePaymentMethodsStore.UserBalancePay
mentMethods.Card - 1].amount){
            subscription_cart.payment_parts[userBalancePaymentMethodsStore.UserBala
ncePaymentMethods.Card - 1].moneyStorageSelectState = 'error'
            Notice.config({ placement: 'bottom-right' })
            Notice.error(t('Встановіть карту!'))
            is_return = true
        }

        if(!subscription_cart.payment_parts[userBalancePaymentMethodsStore.UserBal
ancePaymentMethods.Cash - 1].money_storage_id &&
subscription_cart.payment_parts[userBalancePaymentMethodsStore.UserBalancePay
mentMethods.Cash - 1].amount){
```

```
            subscription_cart.payment_parts[userBalancePaymentMethodsStore.UserBala
ncePaymentMethods.Cash - 1].moneyStorageSelectState = 'error'
            Notice.config({ placement: 'bottom-right' })
            Notice.error(t('Встановіть готівкове сховище!'))
            is_return = true
        }

        let paymentsSum = 0
        subscription_cart.payment_parts.forEach(item => {
            paymentsSum += item.amount
        })

        if(paymentsSum != subscription_cart.total_price){
            Notice.config({ placement: 'bottom-right' })
            Notice.error(t('Сума не збігається!'))
            is_return = true
        }

        if(is_return) return false
    }

    return true
}
```

Б.3 Функція створення івентів

```
function createScheduleEvent(event_type){
    if(event_type == 'custom'){
        createCustomScheduleEventModel.start =
```

```
createCustomScheduleEventModel.day + ' ' +
createCustomScheduleEventModel.start_end[0]
      createCustomScheduleEventModel.end =
createCustomScheduleEventModel.day + ' ' +
createCustomScheduleEventModel.start_end[1]


      schedulesEventsStore.createScheduleEvent(createCustomScheduleEventModel).
then(res => {
         Notice.config({ placement: 'bottom-right' })
         Notice.success(t('Успішно!'))
         let refreshCreateCustomScheduleEventModel = {
            title: '',
            description: '',
            start: '',
            end: '',
            start_end: '',
            room_id: '',
            day: '',
            is_public: true
         }
         Object.assign(createCustomScheduleEventModel,
refreshCreateCustomScheduleEventModel)
         getSchedules()
         isActiveCreateScheduleEventModal.value = false
      })
   }
   else if(event_type == 'group'){
      createGroupScheduleEventModel.start = createGroupScheduleEventModel.day
+ ' ' + createGroupScheduleEventModel.start_end[0]
      createGroupScheduleEventModel.end = createGroupScheduleEventModel.day +
```

```
' ' + createGroupScheduleEventModel.start_end[1]
        createGroupScheduleEventModel.group_id = current_event.value.eventable_id

        schedulesEventsStore.createGroupScheduleEvent(createGroupScheduleEventM
odel).then(res => {
            Notice.config({ placement: 'bottom-right' })
            Notice.success(t('Успішно!'))
            let refreshCreateGroupScheduleEventModel = {
                description: '',
                day: '',
                start: '',
                end: '',
                start_end: '',
                room_id: '',
                group_id: '',
                as_default_schedule_event: false
            }
            Object.assign(createGroupScheduleEventModel,
refreshCreateGroupScheduleEventModel)
            getSchedules()
            isActiveCreateScheduleEventModal.value = false
        })
    }

}
```