

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТУКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ
ЕНЕРГЕТИКИ

кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

«До захисту допущено»
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА
«__» _____ 2024 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Цифрові технології в енергетиці»
зі спеціальності 122 «Комп'ютерні науки»

на тему: Веб-застосунок для організації діяльності спортивної школи

Виконала:
студентка IV курсу, групи ТР-01

_____ ХОЛОДНИЦЬКА Аріна Юріївна _____
(ПРІЗВИЩЕ, ім'я, по батькові) (підпис)

Керівник: *доцент каф. цифрових технологій в енергетиці*
_____ *доцент, к.т.н., Кублій Лариса Іванівна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент: _____
_____ (посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Нормоконтролер: _____ *Беспала О.М.* _____
(підпис)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів без
відповідних посилань.

Студентка _____
(підпис)

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТУКРАЇНИ

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ
ЕНЕРГЕТИКИ**

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти — перший (бакалаврський)

спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Цифрові технології в енергетиці»

ЗАТВЕРДЖУЮ

Завідувач кафедри ЦТЕ

Наталія АУШЕВА

«__» _____ 202__р.

ЗАВДАННЯ

на дипломну роботу студентці

ХОЛОДНИЦЬКІЙ Аріні Юріївні

(прізвище, ім'я, по батькові)

1. Тема роботи **«Веб-застосунок для організації діяльності спортивної школи»**

керівник роботи **Кублій Лариса Іванівна, к.т.н., доцент**

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ 202__р. № _____

2. Термін подання роботи студенткою **07.06.2024 року**

3. Вихідні дані до роботи мова програмування TypeScript, мова розмітки гіпертексту, каскадні таблиці стилів, фреймворки Angular та NestJS, система керування базами даних PostgreSQL, інструмент документування API Swagger, середовище розробки WebStorm.

4. Перелік питань, які потрібно розробити:

1) проаналізувати сучасні підходи, аналоги та інструменти, які використовують для організації діяльності спортивних закладів;

2) визначити архітектуру веб-застосунку для організації діяльності спортивної школи;

3) створити веб-застосунок «Go/ΛFit», спрямований на оптимізацію і полегшення процесів організації та керування спортивною школою.

5. Орієнтовний перелік графічного (ілюстративного) матеріалу діаграма прецедентів, ER-модель бази даних, логічна схема бази даних, інтерфейси сторінок і модальних вікон веб-застосунку, класи серверної частини застосунку, документування API.

6. Дата видачі завдання «15» вересня 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі	17-20.04.24	
3.	Розробка архітектури та загальної структури системи	21-25.04.24	
4.	Розробка частин окремих підсистем	25.04-02.05.24	
5.	Програмна реалізація системи	03-07.05.24	
6.	Оформлення пояснювальної записки	08-12.05.24	
7.	Захист програмного продукту	13-17.05.24	
8.	Передзахист	03-06.06.24	
9.	Подання готової роботи на кафедру	07.06.2024	
10.	Захист	17-21.06.2024	

Студентка

(підпис)

Аріна ХОЛОДНИЦЬКА

(ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Лариса КУБЛІЙ

(ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломну роботу виконано на 50 сторінках. Робота містить 35 ілюстрацій, 20 джерел у переліку посилань, 2 додатки.

Мета роботи — проектування і створення веб-застосунку для організації діяльності спортивної школи.

Підходи і засоби розробки: архітектурний стиль REST, шаблон проектування MVC, фреймворк NestJS для серверної частини застосунку, фреймворк Angular для клієнтської частини застосунку, система керування базами даних PostgreSQL, платформа адміністрування і керування базами даних PostgreSQL pgAdmin, інструмент тестування й документування API Swagger.

Ключові слова: ефективне керування спортивним закладом, веб-застосунок, TypeScript, NestJS, Angular, PostgreSQL.

ANNOTATION

The thesis is made on 50 pages, contains 35 illustrations, 2 appendixes, 20 sources in the list of references.

The purpose of the work is to develop and create a web application for organising the activities of a sports school.

Development approaches and tools: architectural style REST, design pattern MVC, NestJS framework for the server side of the application, Angular framework for the client side of the application, PostgreSQL database management system, administration and management platform for the PostgreSQL database pgAdmin, API testing and documentation tool Swagger.

Keywords: effective management of a sports establishment, web application, TypeScript, NestJS, Angular, PostgreSQL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ	8
ВСТУП.....	10
1 ЗАДАЧА РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ДІЯЛЬНОСТІ СПОРТИВНОЇ ШКОЛИ	12
1.1 Постановка задачі.....	12
1.2 Інструменти і підходи для оптимізації процесів керування спортивними зкладами.....	13
2 ПРОЄКТУВАННЯ ВЕБ-ЗАСТОСУНКУ «GO\FIT»	15
2.1 Визначення потреб користувачів	15
2.1.1 Системна роль «Атлет».....	15
2.1.2 Системна роль «Тренер».....	16
2.1.3 Системна роль «Адміністратор».....	16
2.2 Функціональні можливості веб-застосунку	17
2.3 Вибір засобів розробки	18
2.4 Концептуальна модель бази даних: опис сутностей та їхніх взаємозв'язків	19
3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ	23
3.1. Розробка клієнтської частини застосунку.....	23
3.1.1 Структура головного модуля	23
3.1.2 Навігація між сторінками застосунку «Go\Fit»	25
3.2 Розробка серверної частини веб-застосунку.....	27
3.2.1 Головний модуль	27
3.2.2 Каталог контролерів	28
3.2.3 Каталог сервісів	29
3.2.4 Використання технології ORM для взаємодії з базою даних.....	31
3.2.5 Каталог сутностей	32
3.2.6 Перехоплювач відповідей	33
3.2.7 Декоратори завантаження файлів	34
3.2.8 Документування API	36

4 РОБОТА КОРИСТУВАЧА З ВЕБ-ЗАСТОСУНКОМ	38
4.1 Сторінка входу.....	38
4.2 Головна сторінка	39
4.3 Сторінка «Розклад»	40
4.4 Діалогове вікно створення / редагування подій у розкладі.....	42
4.5 Сторінка «Fit-Gifts»	43
4.6 Діалогове вікно «Кошик».....	44
4.7 Сторінка перегляду активності користувача	45
4.8 Сторінка «Тренери»	46
4.9 Сторінка «Відвідуваність».....	47
4.10 Сторінка «Користувачі».....	48
ВИСНОВКИ	49
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	51
ДОДАТОК А	53
ДОДАТОК Б	62

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ACID (Atomicity, Consistency, Isolation, Durability) — набір властивостей, які гарантують надійну роботу транзакцій у базі даних: атомарність, узгодженість, ізолюваність, довговічність.

API (Application Programming Interface) — набір інструкцій, які дають можливість кільком програмам спілкуватись між собою та обмінюватись даними.

CSS (Cascading Style Sheets) — каскадні таблиці стилів, які застосовують для візуального форматування документа в мовах розмітки.

CRUD (Create, Read, Update, Delete) — аббревіатура, яка позначає чотири основні операції керування даними: створення, читання, редагування, видалення.

DOM (Document Object Model) — специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами.

DTO (Data Transfer Object) — об'єкт, який використовується для передачі даних з одного місця програми в інше.

HTML (HyperText Markup Language) — стандартизована мова розмітки документів для перегляду веб-сторінок у браузері.

HTTP (HyperText Transfer Protocol) — протокол передачі даних, на основі якого працює всесвітня павутина.

HTTP Endpoint — кінцева точка (адреса) веб-сервісу чи додатку, яку можна викликати за допомогою HTTP-протоколу.

MIME-тип (Multipurpose Internet Mail Extension) — специфікація для передачі мережею файлів різного типу: зображень, музики, текстів, відео, архівів тощо.

MVC (Model-View-Controller) — архітектурний шаблон, основною ідеєю якого є розділення програмного забезпечення на три окремі складові: модель, представлення і контролер.

ORM (Object Relational Mapping) — техніка програмування, яка дає можливість використовувати об'єктно-орієнтовану модель для роботи з реляційними базами даних.

REST (Representational State Transfer) — стиль архітектури для розподілених систем, що використовується для створення веб-сервісів.

SPA (Single Page Application) — тип веб-додатку, в якому відображення даних відбувається в межах однієї сторінки.

Angular — фреймворк, який використовується для створення динамічних односторінкових веб-додатків.

NestJS — фреймворк для створення серверних додатків на Node.js.

Node.js — програмне середовище, яке надає можливість запускати програми написані на JavaScript.

PostgreSQL — система керування реляційними базами даних.

Swagger — інструмент для документування та тестування API.

TypeORM — ORM для TypeScript та JavaScript, який дає змогу працювати з реляційними базами даних

pgAdmin — платформа для адміністрування та керування СКБД PostgreSQL.

ВСТУП

У сучасному світі розвиток фізичних і ментальних здібностей стає не лише запорукою успіху, але й важливим елементом самореалізації. Спорт відіграє невід’ємну роль у формуванні здорового способу життя і розвитку особистості, допомагаючи зберегти енергію та зосередженість у потоці щоденних викликів.

У контексті роботи термін «спортивна школа» охоплює різноманітні заклади, такі як спортивні центри, клуби, студії та інші, спрямовані на надання послуг з розвитку, підтримки та вдосконалення фізичних здібностей для людей різного віку — від дошкільного до дорослого.

Метою роботи є проєктування й розробка веб-застосунку, спрямованого на вдосконалення процесів організації та керування спортивним закладом.

Завданням роботи є:

- 1) проаналізувати сучасні підходи, аналоги та інструменти, які використовують для організації діяльності спортивних закладів;
- 2) визначити архітектуру веб-застосунку для організації діяльності спортивної школи;
- 3) створити веб-застосунок, спрямований на оптимізацію і полегшення процесів організації та керування спортивною школою.

Практичне значення роботи полягає в тому, що спроектований веб-застосунок може використовуватися спортивними закладами для підвищення ефективності їхньої діяльності й покращення взаємодії між адміністрацією, тренерами і відвідувачами-спортсменами.

Результати роботи апробовано на XXI Міжнародній науково-практичній конференції молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики», м. Київ, 23-26 квітня 2024 року [1].

Дипломна записка складається з чотирьох розділів. У першому розділі описано задачу розробки веб-застосунку для організації діяльності спортивної школи, а також інструменти та підходи, спрямовані на оптимізацію процесів

керування спортивними закладами. Другий розділ містить опис проєктування веб-застосунку «Go/∧Fit», включаючи визначення потреб користувачів, функціональні можливості, засоби розробки та концептуальну модель бази даних. У третьому розділі розглянуто програмну реалізацію клієнтської та серверної частин веб-застосунку. Четвертий розділ демонструє способи взаємодії користувачів різних системних ролей з веб-застосунком.

1 ЗАДАЧА РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ДІЯЛЬНОСТІ СПОРТИВНОЇ ШКОЛИ

В Україні існує велика кількість спортивних закладів, більшість яких для взаємодії з відвідувачами-спортсменами обмежується використанням інформаційних сайтів, соціальних мереж, месенджерів тощо. Ці засоби зручні для розповсюдження інформації, проте вони не забезпечують повноцінного керування та ефективної взаємодії з аудиторією. У зв'язку з цим виникає потреба в розробці веб-застосунку, спрямованого на оптимізацію процесів керування у сфері спортивного навчання.

1.1 Постановка задачі

Розробка веб-застосунку націлена на розв'язання таких задач:

- керування розкладом тренувань;
- ведення обліку відвідуваності на заняттях;
- інформування спортсменами про відсутність на заняттях;
- створення спортсменами запитів на відпрацювання пропущених занять;
- забезпечення зручного доступу до інформації (графік тренувань, оголошення, відомості про тренерів тощо);
- відображення статистики відвідуваності;
- моніторинг використання тренувального пакета (абонементу) спортсмена-відвідувача;
- стимулювання спортсменів до досягнення нових висот через бонусну систему «Fit-Gifts», яка передбачає накопичення віртуальних валютних одиниць та їхній обмін на спортивні товари або переваги в межах закладу / застосунку.

1.2 Інструменти і підходи для оптимізації процесів керування спортивними закладами

Можна виокремити два підходи, спрямовані на полегшення процесів керування та організації роботи спортивних закладів:

Використання спеціалізованих інструментів. Деякі спортивні установи обирають спеціалізовані платформи, наприклад, Glofox [2]. Цей інструмент призначений для керування різними типами спортивних закладів, такими як фітнес-студії, тренажерні зали, бойові клуби тощо. Платформа пропонує широкий спектр функцій для ефективного керування діяльністю спортивного закладу (рисунок 1.1).

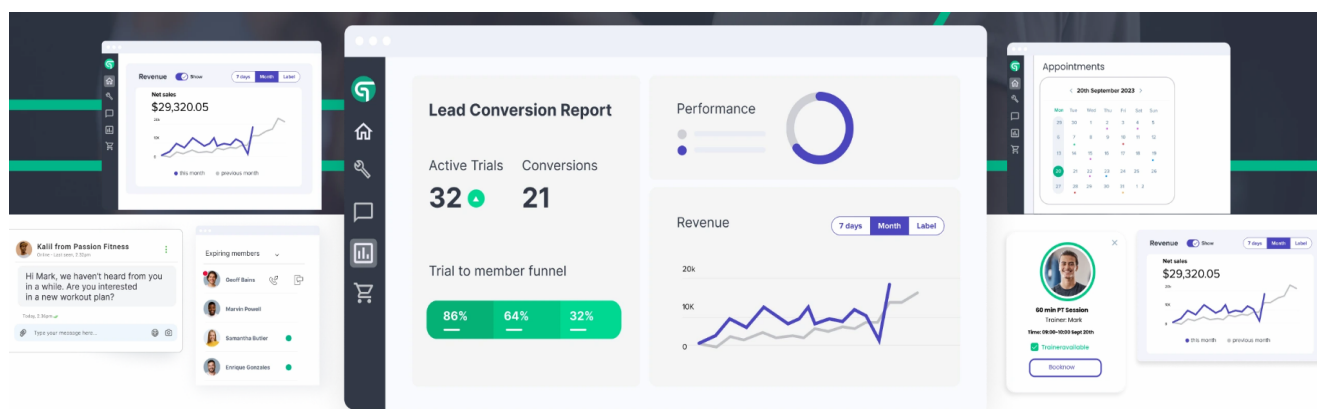


Рисунок 1.1 — Інтерфейси інтегрованої платформи «Glofox» [2]

Основні функціональні можливості платформи «Glofox»:

- керування розкладом занять;
- бронювання записів на тренування;
- ведення фінансової звітності;
- інтеграція з платіжними системами для оплати тренувань і послуг;
- сповіщення про нові події та акції.

Недоліки платформи «Glofox»:

- використання платформи є платним, визначається індивідуальними потребами спортивного закладу;

– обмежена адаптація до спеціальних функції або особливих вимог спортивного закладу / бізнесу.

Використання власних програмних засобів. Підхід спрямовано на задоволення конкретних потреб та особливостей діяльності закладу. Наприклад, школа акробатики і спортивної гімнастики «Jump Up» [3] використовує власний програмний засіб, який відповідає унікальним вимогам та особливостям закладу (рисунок 1.2).

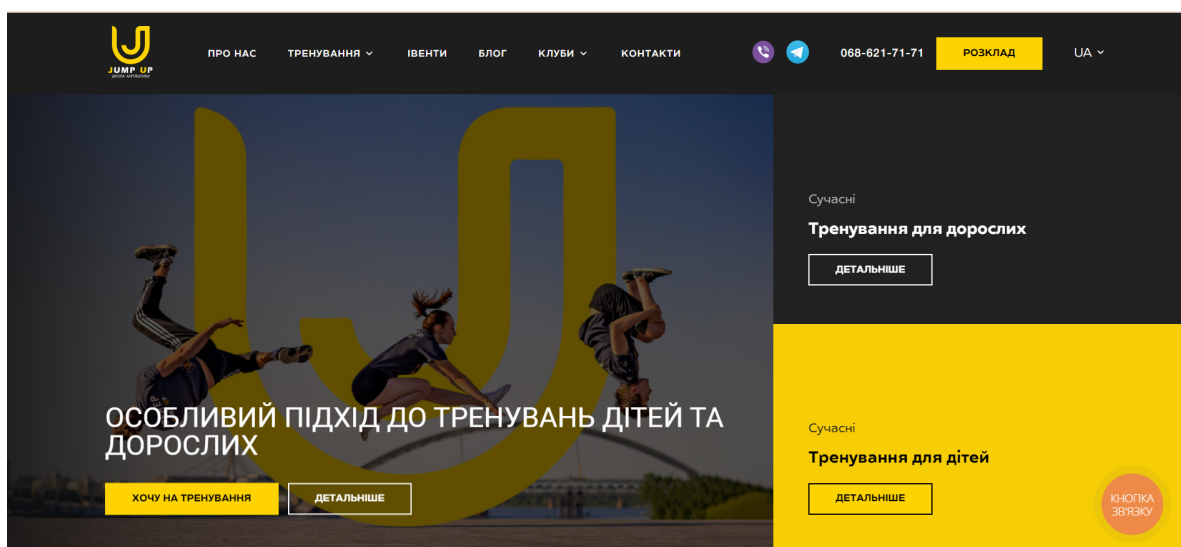


Рисунок 1.2 — Веб-ресурс школи акробатики «Jump Up» [3]

Основні функціональні можливості веб-ресурсу «Jump Up»:

- перегляд видів діяльності школи;
- перегляд розкладу занять за конкретними напрямками;
- бронювання запису на тренування;
- купівля абонементу через інтеграцію з платіжними сервісами;
- замовлення зворотного дзвінка.

Незважаючи на наявність готових інструментів, розробка власного програмного засобу є доцільною, якщо необхідно врахувати унікальні й специфічні потреби закладу та уникнути плати за надлишкові функції, оскільки безкоштовних аналогів, призначених для керування спортивним закладом, виявлено не було.

2 ПРОЄКТУВАННЯ ВЕБ-ЗАСТОСУНКУ «GO\FIT»

Веб-застосунок для організації діяльності спортивної школи «Go\Fit» передбачає розділення функціоналу за системними ролями: «Атлет», «Тренер», «Адміністратор».

Залежно від визначеної системної ролі, в застосунку надаватимуться дозволи або застосовуватимуться обмеження щодо використання конкретних функцій.

2.1 Визначення потреб користувачів

Потреби користувачів веб-застосунку для організації діяльності спортивної школи визначаються з урахуванням системних ролей та обов'язків, які вони повинні виконувати.

2.1.1 Системна роль «Атлет»

Атлети використовують систему з метою отримання необхідної інформації для підвищення ефективності своєї тренувальної діяльності. Основні потреби атлетів включають:

- перегляд розкладу занять;
- перегляд відомостей про тренерів;
- перегляд каталогу подарункових товарів спортивного закладу;
- перегляд власних досягнень у межах спортивного закладу;
- перегляд новин / оголошень спортивного закладу;
- моніторинг використання власного абонементу;
- моніторинг статистики власної відвідуваності;
- інформування про відсутність на занятті;
- створення запитів на відпрацювання пропущених занять;

- отримання сповіщень про зміни в розкладі, оголошення від адміністрації, відхилення або погодження запитів на відпрацювання пропущених занять;
- обмін накопичених бонусів на товари або переваги спортивного закладу;
- оцінка бонусних товарів.

2.1.2 Системна роль «Тренер»

Тренери відповідають за організацію і проведення занять. Основні потреби тренерів включають:

- перегляд розкладу занять;
- перегляд інформації про атлетів (лише тих, які займаються в закріплених за тренером групах);
- перегляд каталогу подарункових товарів спортивного закладу;
- перегляд новин / оголошень спортивного закладу;
- ведення обліку відвідуваності (лише за групами, закріпленими за тренером);
- зміна статусів запитів атлетів на відпрацювання пропущених занять (лише тих, які займаються в закріплених за тренером групах);
- отримання сповіщень про зміни в розкладі, оголошення від адміністрації, відсутність атлетів на заняттях, появу нових запитів на відпрацювання пропущених занять атлетами (які займаються в групах, закріплених за тренером);
- обмін накопичених бонусів на товари або переваги спортивного закладу;
- оцінка бонусних товарів;
- ознайомлення з правилами спортивного закладу.

2.1.3 Системна роль «Адміністратор»

Адміністратори відповідають за підтримку та функціонування системи. Основні обов'язки адміністраторів такі:

- реєстрація користувачів у системі;
- формування тренувальних груп;
- створення / видалення тренувальних пакетів;

- модифікація розкладу занять;
- ведення обліку відвідуваності;
- зміна статусів запитів атлетів на відпрацювання пропущених занять;
- додавання бонусних валютних одиниць користувачам;
- оновлення інформації про тренерів та атлетів;
- оновлення інформації в розділі оголошень;
- оновлення каталогу подарункових товарів закладу.

2.2 Функціональні можливості веб-застосунку

Основні функції веб-застосунку концептуально подано у вигляді діаграми «Use Case» на рисунку 2.1.



Рисунок 2.1 — Діаграма прецедентів веб-застосунку

Актори на схемі відповідають визначеним системним ролям: «Атлет», «Тренер», «Адміністратор». Діаграма забезпечує чітке розуміння ролей користувачів та можливі сценарії використання веб-застосунку.

2.3 Вибір засобів розробки

Для розробки веб-застосунку, спрямованого на організацію діяльності спортивної школи, обрано архітектурний стиль REST [4], який передбачає розділення функціоналу між клієнтською та серверною частинами. Клієнтська частина відповідає за відображення інтерфейсу користувача, в той час як серверна обробляє логіку бізнес-процесів, виконує запити до бази даних і здійснює обмін інформацією з клієнтом.

Для реалізації клієнтської частини обрано Angular-фреймворк [5], який використовує мову програмування TypeScript, HTML для створення структури веб-сторінок та CSS для стилізації. Фреймворк надає можливість систематизувати код у вигляді компонентів, що значно покращує читабельність коду, спрощує підтримку і подальший розвиток проєкту.

Серверну частину застосунку реалізовано з використанням фреймворку NestJS [6], який вважається доцільним для створення надійних і масштабованих серверних додатків у середовищі Node.js. Обраний інструмент поєднує в собі концепції об'єктно-орієнтованого програмування (ООП), функціонального програмування (ФП) і функціонально-реактивного програмування (ФРП).

Варто зазначити, що фреймворки NestJS і Angular синтаксично та архітектурно схожі, обидва використовують мову програмування TypeScript. Використання цих фреймворків у поєднанні надає можливість спростити архітектуру веб-застосунку в цілому і перевикористовувати код.

Як систему керування базами даних використано PostgreSQL [7], що має вбудований механізм транзакцій [8]. Взаємодію між PostgreSQL і NestJS реалізовано за допомогою бібліотеки TypeORM, яка базується на технології об'єктно-реляційного відображення (ORM) [9].

2.4 Концептуальна модель бази даних: опис сутностей та їхніх взаємозв'язків

Концептуальна модель бази даних веб-застосунку «Go/Fit» зображено на рисунку 2.2 у вигляді ER-моделі [10], основними елементами якої є:

- об'єкти (сутності);
- атрибути об'єктів;
- зв'язки між об'єктами.

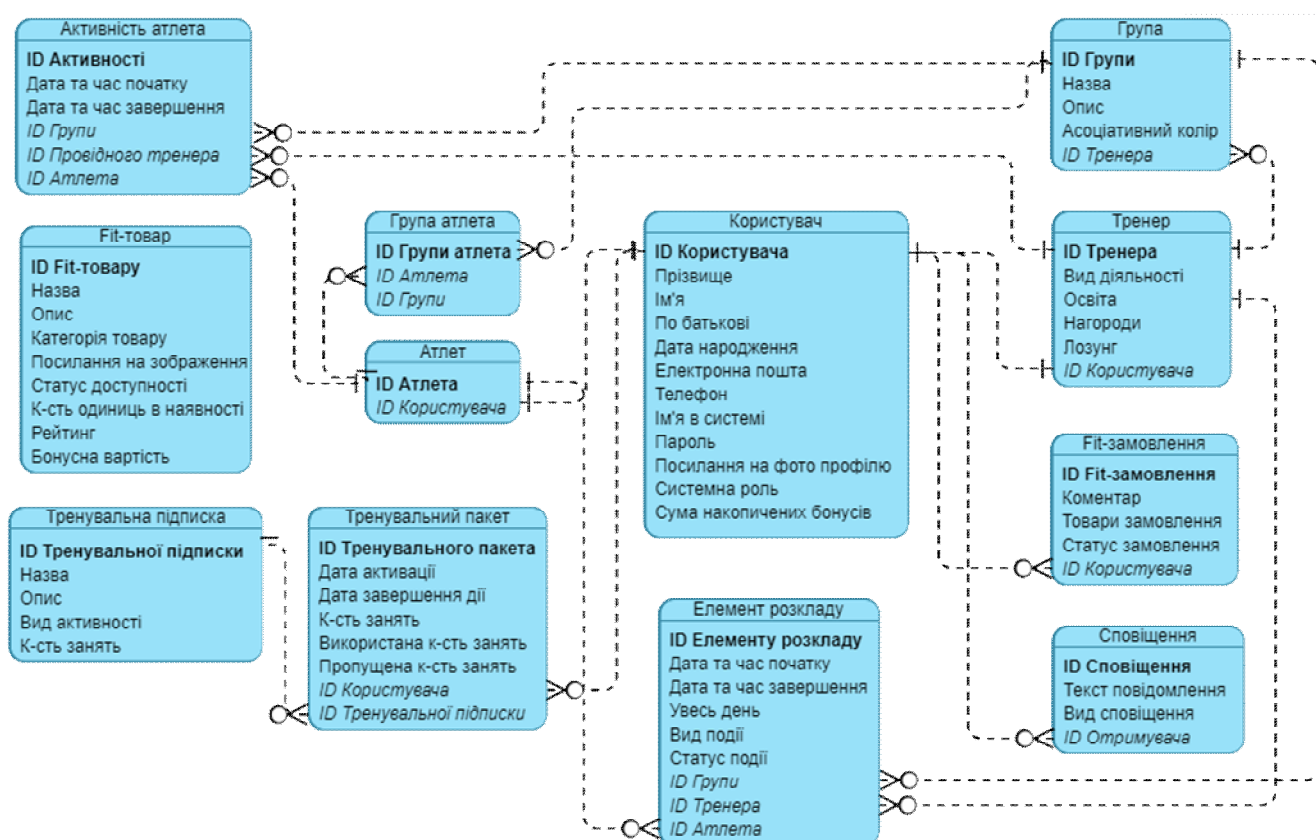


Рисунок 2.2 — ER-модель бази даних для веб-застосунку «Go/Fit»

У базі даних «go-a-fit» сутностями є (в дужках поряд з назвою сутності зазначено атрибути):

- «Користувач» (ID користувача, прізвище, ім'я, по батькові, дата народження, електронна адреса, телефон, ім'я в системі, пароль, посилання на фото профілю, системна роль, сума накопичених бонусів);
- «Атлет» (ID атлета, ID користувача);

- «Тренер» (ID тренера, вид діяльності, освіта, нагороди, лозунг, ID користувача);
- «Група» (ID групи, назва, опис, асоціативний колір, ID тренера):
- «Група атлета» (ID групи атлета, ID атлета, ID групи);
- «Активність атлета» (ID активності атлета, дата і час початку, дата і час закінчення, ID групи, ID провідного тренера, ID атлета);
- «Елемент розкладу» (ID елемента розкладу, дата і час початку, дата і час закінчення, відмітка про планування події на весь день, вид події, статус події, ID групи, ID тренера, ID атлета);
- «Тренувальна підписка» (ID тренувальної підписки, назва, опис, вид активності, доступна кількість занять);
- «Тренувальний пакет» (ID тренувального пакета, дата активації, дата закінчення терміну дії, доступна кількість занять, використана кількість занять, пропущена кількість занять, ID користувача, ID тренувальної підписки);
- «Fit-товар» (ID Fit-товару, назва, опис, категорія товару, посилання на зображення, статус доступності, кількість наявних одиниць, рейтинг, бонусна вартість);
- «Fit-замовлення» (ID Fit-замовлення, коментар, замовлені товари, статус замовлення, ID користувача);
- «Сповіщення» (ID сповіщення, повідомлення, вид сповіщення, ID отримувача).

Атрибути «ID користувача», «ID атлета», «ID тренера», «ID групи», «ID групи атлета», «ID активності атлета», «ID елемента розкладу», «ID тренувальної підписки», «ID тренувального пакета», «ID Fit-товару», «ID Fit-замовлення» та «ID сповіщення» є первинними ключами, оскільки гарантують унікальність кожної відповідної сутності.

У застосунку передбачено, що атлет може відвідувати кілька категорій занять, тобто бути учасником кількох груп. Для реалізації цього функціоналу створено таблицю «Група атлета», яка поєднує ідентифікатори атлета і групи (використано тип зв'язку «один-до-багатьох»).

Атлет і тренер є користувачами системи. Для поєднання таблиць «Атлет» — «Користувач», «Тренер» — «Користувач» використано типи зв'язків «один-до-одного».

Атлет має тренувальні підписки (іншими словами — абонементи), термін дії яких обмежений. При купівлі абонементу для атлета формується індивідуальний тренувальний пакет. Таблиці «Тренувальний пакет» і «Тренувальна підписка» поєднано зв'язком «багато-до-одного», таблиці «Користувач» і «Тренувальний пакет» поєднано зв'язком «один-до-багатьох».

За кожною групою закріплено тренера, відповідно, таблиці «Група» і «Тренер» поєднані за допомогою зв'язку «багато-до-одного».

У застосунку передбачено можливість ведення обліку відвідуваності, з цією метою створено таблицю «Активність атлета». Атлет може відпрацьовувати пропущене заняття не лише в межах закріплених за ним груп. У випадках непередбачуваних обставин, тренер однієї групи може бути замінений тренером іншої групи. Таблиця «Активність атлета» враховує описані сценарії, поєднуючи ідентифікатор групи, ідентифікатор провідного тренера та ідентифікатор атлета зв'язками «багато-до-одного» з відповідними сутностями.

Атлети можуть відвідувати і планувати не лише групові заняття, а також індивідуальні. Таблиця «Елемент розкладу» пов'язана зв'язком «багато-до-одного» з таблицею «Тренер». У випадку групового заняття таблиця додатково поєднується із таблицею «Група», у випадку індивідуального заняття — поєднується з таблицею «Атлет» (тип зв'язку «багато-до-одного»).

Користувачі можуть створювати замовлення спортивних товарів в обмін на накопичені бонусні кошти. Таблиці «Fit-замовлення» і «Користувач» поєднані зв'язком «багато-до-одного».

Користувачі повинні мати можливість отримувати сповіщення. Для таблиць «Користувач» і «Сповіщення» встановлено тип зв'язку «один-до-багатьох».

На основі описаної концептуальної моделі розроблено логічну схему БД, яка містить 12 таблиць і 15 відношень (рисунок 2.3).

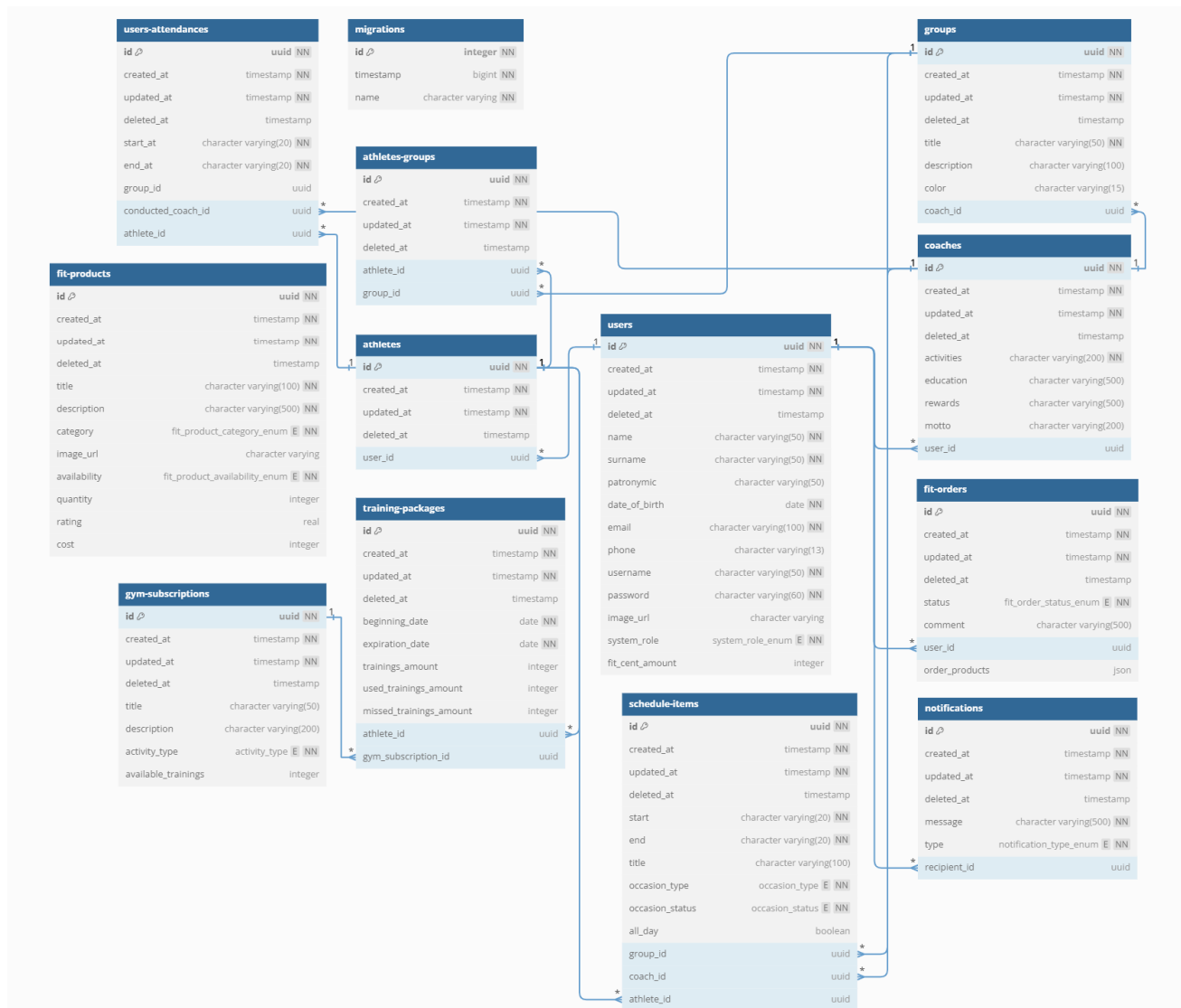


Рисунок 2.3 — Логічна схема бази даних «go-a-fit»

Логічна схема відображає структуру даних, необхідну для підтримки й функціонування веб-застосунку. Схема визначає, як дані будуть використовуватися і як вони пов'язанні між собою для виконання певних операцій і завдань у веб-застосунку.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Як зазначалося в підрозділі 2.3, для розробки веб-застосунку «Go/Fit» було використано ряд технологій і фреймворків. У цьому розділі розглянуто більш детально обидві частини застосунку — клієнтську і серверну, описано структури головних модулів, загальну архітектуру, особливості й підходи до розробки.

3.1. Розробка клієнтської частини застосунку

Розробка клієнтської частини застосунку «Go/Fit» здійснена з використанням фреймворку Angular, який ґрунтується на модульній архітектурі. Фреймворк надає широкий набір інструментів для створення динамічних та інтерактивних веб-застосунків. Основними будівельними блоками Angular-застосунку є компоненти, сервіси, декоратори, перехоплювачі запитів (interceptors), вартіві (guards), директиви (directives) та канали (pipes).

3.1.1 Структура головного модуля

Кожен проєкт з використанням Angular-фреймворку повинен містити принаймні один кореневий модуль, традиційно іменований як «AppModule» [11].

Модуль в Angular — це клас, який анотовано декоратором «@NgModule» з такими базовими конфігураційними властивостями:

- declarations — класи представлень, які належать модулю (компоненти, директиви, канали);
- exports — класи, які потрібно експортувати для використання в інших модулях;
- imports — інші модулі / автономні елементи, класи яких використовуються в поточному модулі;
- providers — класи сервісів, доступні для використання в межах модуля;
- bootstrap — вказівка на кореневий компонент застосунку.

Структуру кореневого модуля клієнтської частини застосунку «Go\Fit» подано на рисунку 3.1.

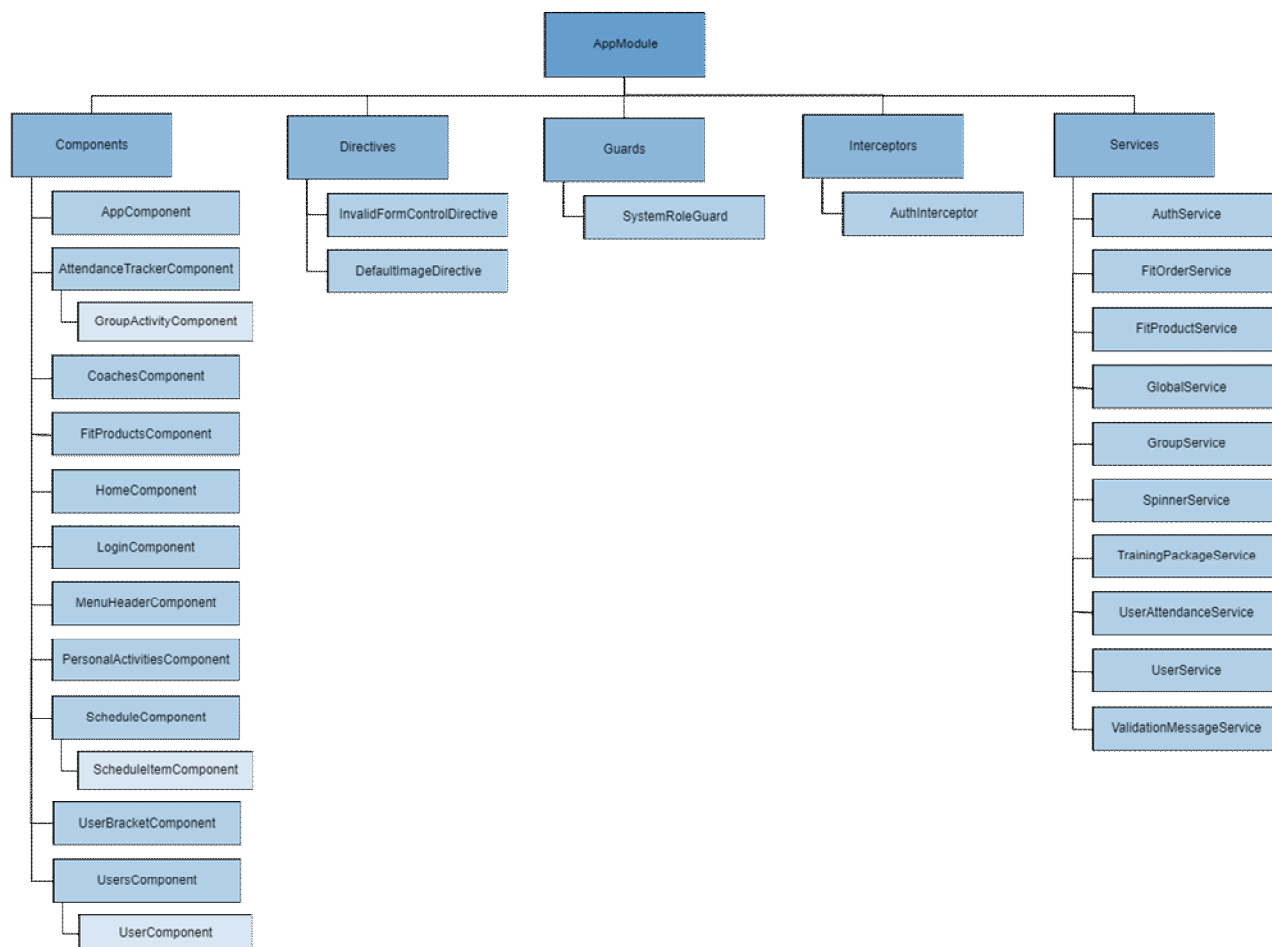


Рисунок 3.1 — Структура головного модуля клієнтської частини застосунку

Кореневий модуль («AppModule») застосунку «Go\Fit» декларує компоненти («AppComponent», «HomeComponent», «MenuHeaderComponent», «ScheduleComponent», «ScheduleItemFormComponent», «LoginComponent», «UsersComponent», «UserComponent», «CoachesComponent», «PersonalActivitiesComponent», «AttendanceTrackerComponent», «GroupActivityComponent», «FitProductsComponent», «UserBracketComponent»); імпортує інші модулі, класи яких використовуються складовими частинами модуля, автономні директиви («DefaultImageDirective» та «InvalidFormControlDirective»); реєструє сервіси, перехоплювач (interceptor) автентифікації та вартового (guard) системних ролей. Вміст файлу «app.module.ts» наведено в додатку А.

3.1.2 Навігація між сторінками застосунку «Go\Fit»

Фреймворк Angular призначений для розробки односторінкових застосунків (SPA) [12]. Фреймворк використовує компоненти для побудови інтерфейсу. Кожен компонент може містити в собі інші компоненти, відповідно, всі сторінки застосунку є компонентами.

Поняття маршрутизації в Angular (Routing) [13] — це перехід від одного подання (шаблону компонента) до іншого, залежно від вказаного шляху (URL). Для користувача процес навігації в застосунку має вигляд звичайного переходу за посиланнями, проте відмінність полягає в тому, що замість повного перезавантаження сторінки або завантаження додаткової, Angular Router здійснює оновлення інтерфейсу шляхом динамічної зміни компонента для відображення.

Відповідність між шляхом маршруту і компонентом відображення проілюстровано рисунку 3.2.



Рисунок 3.2 — Відповідність шляхів маршрутів до компонентів відображення

Оскільки користувачі, залежно від визначених системних ролей, можуть мати обмежений доступ до сторінок, було розроблено клас захисту маршрутів, відомий як механізм «Route Guard» [14]. Реалізація класу та приклад його застосування наведено на рисунках 3.3 і 3.4.

```
@Injectable()
export class SystemRoleGuard implements CanActivate {
  constructor(
    private readonly router: Router,
    private readonly userService: UserService,
  ) { }

  public canActivate(route: ActivatedRouteSnapshot): Observable<boolean> | boolean {
    const user = this.userService.user;
    const requiredRoles: SystemRole[] = route.data.roles;
    const isUserHasRequiredRoles = requiredRoles.some((role: SystemRole) => user?.systemRole === role);

    if (isUserHasRequiredRoles) {
      return true;
    } else {
      this.router.navigate([ '/home' ]);
      return false;
    }
  }
}
```

Рисунок 3.3 — Реалізація класу захистів маршрутів

```
{ path: 'user/new', component: UserComponent, canActivate: [SystemRoleGuard], data: { roles: [SystemRole.Admin] } },
```

Рисунок 3.4 — Приклад використання вартового (guard) системних ролей для обмеження доступу до сторінок

Клас захисту маршрутів забезпечує контроль доступу до різних сторінок веб-застосунку. У випадку невідповідності правилам доступу клас захисту маршрутів перенаправляє користувача на вказану в конфігураційних налаштуваннях сторінку (наприклад, на головну сторінку, якщо користувач не має потрібних прав доступу, проте є автентифікованим у системі). Такий підхід зменшує ризик несанкціонованого доступу до ресурсів і сприяє надійності веб-застосунку.

3.2 Розробка серверної частини веб-застосунку

Фреймворк NestJS використовує модульну архітектуру. Проєкт серверної частини веб-застосунку складається з:

- головного модуля (AppModule);
- каталога контролерів;
- каталога сервісів;
- каталога сутностей;
- об'єктів передачі даних (DTO) [15];
- міграцій;
- перераховуваних типів (ENUMs);
- перехоплювача відповідей;
- фільтра перевірки формату файлів, які завантажуються на сервер;
- декораторів завантаження файлів;
- конфігураційних файлів.

3.2.1 Головний модуль

Головний модуль (або кореневий модуль) є центральним елементом, з якого фреймворк NestJS будує застосунок. Модуль відповідає за налаштування і координацію всіх інших модулів, контролерів і сервісів додатку.

В головний файл (AppModule) імпортовано:

- конфігураційний модуль NestJS (ConfigModule) для налаштувань глобальних параметрів застосунку;
- модуль для інтеграції з Google Drive API (GoogleDriveModule);
- модуль для інтеграції з базою даних (TypeOrmModule);
- модуль для реалізації користувацьких репозиторіїв (TypeOrmExModule);
- модуль для роботи з JSON Web Tokens (JwtModule).

У головному модулі зареєстровано контролери, сервіси і користувацькі репозиторії. Лістинг файлу подано в додатку А.

3.2.2 Каталог контролерів

Контролери в NestJS призначені для обробки вхідних HTTP-запитів і повернення відповідей. Кожен контролер може містити більше одного маршруту. Кожен маршрут може виконувати різні дії [6].

Серверна частина застосунку «Go\Fit» налічує 13 контролерів, які здійснюють обробку запитів певних типів:

- ApplicationController — загальних запитів до застосунку;
- AuthController — запитів, пов'язаних з автентифікацією користувачів;
- CoachesController — запитів, пов'язаних з тренерами;
- AthletesController — запитів, пов'язаних з атлетами;
- FitOrdersController — запитів, пов'язаних із замовленнями Fit-товарів;
- FitProductsController — запитів, пов'язаних з Fit-товарами;
- GroupsController — запитів, пов'язаних з тренувальними групами;
- NotificationsController — запитів, пов'язаних зі сповіщеннями;
- ScheduleItemsController — запитів, пов'язаних з розкладом тренувань;
- TrainingPackagesController — запитів, пов'язаних з абонементом користувачів;
- UsersAttendancesController — запитів, пов'язаних з відвідуваністю користувачів;
- UsersController — запитів, пов'язаних з користувачами;
- GymSubscriptionsController — запитів, пов'язаних з тренувальними підписками.

Приклад створеного контролера класу «ScheduleItem» (елемент розкладу) для операцій пошуку, створення, оновлення і видалення (CRUD) продемонстровано на рисунку 3.5. Декоратор «@ApiTags('schedule-items')» використано для документування і тестування API за допомогою інструмента Swagger [16]. Декоратор «@Controller('api/schedule-items')» визначає шлях, за яким будуть доступні методи контролера. Декоратори «@Get()», «@Post()», «@Put(':id')», «@Delete(':id')» визначають тип HTTP-запиту. Параметр «:id» вказує на те, що в маршруті очікується ідентифікатор елемента розкладу.

```

@ApiTags({ tags: 'schedule-items' })
@Controller({ prefix: 'api/schedule-items' })
export class ScheduleItemsController {
    constructor(private readonly scheduleItemService: ScheduleItemService) {}

    @Get()
    public async findAll(): Promise<ScheduleItem[]> {
        return await this.scheduleItemService.findAll();
    }

    @Post()
    public async create(
        @Body() scheduleItemCreateDto: ScheduleItemUpsertDto,
    ): Promise<ScheduleItem> {
        return await this.scheduleItemService.create(scheduleItemCreateDto);
    }

    @Put({ path: ':id' })
    public async updateById(
        @Param({ property: 'id' }) id: string,
        @Body() scheduleItemUpdateDto: ScheduleItemUpsertDto,
    ): Promise<ScheduleItem> {
        return await this.scheduleItemService.updateById(id, scheduleItemUpdateDto);
    }

    @Delete({ path: ':id' })
    public async deleteById(
        @Param({ property: 'id' }) id: string,
    ): Promise<void> {
        return await this.scheduleItemService.deleteById(id);
    }
}

```

Рисунок 3.5 — Реалізація контролера для сутності «ScheduleItem»

Впровадження залежності (Dependency Injection) [17] сервісу «ScheduleItemService» до класу контролера («ScheduleItemController») реалізовано через конструктор контролера. Це дає можливість отримувати доступ до методів сервісу для обробки бізнес-логіки і виконання операцій з базою даних.

3.2.3 Каталог сервісів

Для серверної частини застосунку «Go/Fit» реалізовано такі класи сервісів: AppService, AthleteService, CoachService, UserService, FitOrderService, FitProductService, GroupService, NotificationService, ScheduleItemService, UserAttendanceService, TrainingPackageService, GymSubscriptionService. Зазначені сервіси впроваджено у відповідні контролери.

Як було згадано в підпункті 3.2.2, сервіси відповідають за реалізацію бізнес-логіки і взаємодію з базою даних веб-застосунку «Go\Fit». Лістинг коду класу «ScheduleItemService» подано в додатку А.

У методах створення, оновлення і видалення записів з таблиці «ScheduleItem» використано механізм транзакцій.

Транзакція — логічний блок, який поєднує одну чи більше операцій в базі даних і дає можливість підтвердити або відхилити результати виконання всіх операцій у блоці. Використання механізму транзакцій є важливим аспектом розробки веб-систем, особливо в контексті операцій, які вимагають надійності й цілісності даних. Впровадження механізму транзакцій у шаблоні проектування «UnitOfWork» наведено на рисунку 3.6.

```
@Injectable()
export class UnitOfWorkService {
  constructor(
    private readonly dataSource: DataSource,
    public readonly userRepository: UserRepository,
    public readonly athletesGroupRepository: AthletesGroupRepository,
    public readonly athleteRepository: AthleteRepository,
    public readonly coachRepository: CoachRepository,
    public readonly fitOrderRepository: FitOrderRepository,
    public readonly fitProductRepository: FitProductRepository,
    public readonly groupRepository: GroupRepository,
    public readonly scheduleItemRepository: ScheduleItemRepository,
    public readonly notificationRepository: NotificationRepository,
    public readonly userAttendanceRepository: UserAttendanceRepository,
    public readonly gymSubscriptionRepository: GymSubscriptionRepository,
    public readonly trainingPackageRepository: TrainingPackageRepository,
  ) {}

  public async doWork<T>(work: () => T): Promise<T> {
    const queryRunner = this.dataSource.createQueryRunner();

    await queryRunner.connect();
    await queryRunner.startTransaction();

    try {
      const result = await work();
      await queryRunner.commitTransaction();

      return result;
    } catch (error) {
      await queryRunner.rollbackTransaction();
      Logger.error(error);
      throw error;
    } finally {
      await queryRunner.release();
    }
  }
}
```

Рисунок 3.6 — Реалізація сервісу «UnitOfWork», який використовує механізм транзакцій

Для застосунку «Go\Fit» використання транзакцій є критично важливим не лише для операцій модифікації розкладу, а ще й запису / скасування присутності відвідувачів на заняттях, обміну бонусів на заохочувальні презенти, редагування персональних даних користувачів тощо.

3.2.4 Використання технології ORM для взаємодії з базою даних

Технологія об'єктно-реляційного відображення (ORM) надає можливість взаємодіяти з базою даних через об'єктно-орієнтований підхід. Замість прямого написання SQL-запитів, робота з базою даних відбувається шляхом її подання у вигляді об'єктів і класів.

Інтеграцію між сервером NestJS та БД PostgreSQL здійснено за допомогою пакету «@nestjs/typeorm». З цією метою в головний модуль серверної частини веб-застосунку імпортовано модуль «TypeOrmModule» і задано властивості конфігурації (рисунок 3.7).

```
export const typeOrmOptions = {
  type: 'postgres',
  url: process.env.POSTGRES_URL,
  host: process.env.POSTGRES_HOST || 'localhost',
  port: process.env.POSTGRES_PORT || 5432,
  password: process.env.POSTGRES_PASSWORD || 'admin',
  username: process.env.POSTGRES_USERNAME || 'postgres',
  entities: [`${__dirname}/../entities/*.entity{.ts,.js}`],
  migrations: [`${__dirname}/../migrations/*.ts`],
  migrationsRun: true,
  database: process.env.POSTGRES_DATABASE || 'go-a-fit',
  synchronize: false,
  logging: true,
} as TypeOrmModuleOptions;
```

Рисунок 3.7 — Конфігурація підключення БД «go-a-fit» до серверної частини застосунку

Властивості конфігурації включають тип, назву, номер порту, ідентифікатор доступу користувача до бази даних.

3.2.5 Каталог сутностей

Сутності в NestJS є класами, які відповідають таблицям у базі даних і використовуються для взаємодії з ними через ORM. На серверній частині застосунку «Go\Fit» реалізовано такі класи сутностей: «Athlete», «AthleteGroup», «BaseEntity», «Coach», «FitOrder», «FitProduct», «Group», «GymSubscription», «Notification», «ScheduleItem», «TrainingPackage», «User», «UserAttendance». Приклад реалізації сутності «ScheduleItem» для таблиці «schedule-items» (елементи розкладу) наведено на рисунку 3.8.

```
@Entity( options: { name: 'schedule-items' })
export class ScheduleItem extends BaseEntity {
  @Column( options: { name: 'start', type: 'varchar' })
  public start: string;

  @Column( options: { name: 'end', type: 'varchar' })
  public end: string;

  @Column( options: { name: 'title', type: 'varchar', length: 100 })
  public title: string;

  @Column( options: { name: 'occasion_type', type: 'enum', enum: OccasionType })
  public occasionType: OccasionType;

  @Column( options: { name: 'occasion_status', type: 'enum', enum: OccasionStatus })
  public occasionStatus: OccasionStatus;

  @Column( options: { name: 'all_day', type: 'boolean', default: false })
  public isAllDay: boolean;

  @ManyToOne( typeFunctionOrTarget: () => Group, inverseSide: (group : Group ) => group.scheduleItems )
  @JoinColumn( options: { name: 'group_id' })
  public group: Group;

  @ManyToOne( typeFunctionOrTarget: () => Coach, inverseSide: (coach : Coach ) => coach.scheduleItems )
  @JoinColumn( options: { name: 'coach_id' })
  public coach: Coach;

  @ManyToOne( typeFunctionOrTarget: () => Athlete, inverseSide: (athlete : Athlete ) => athlete.scheduleItems )
  @JoinColumn( options: { name: 'athlete_id' })
  public athlete: Athlete;
}
```

Рисунок 3.8 — Реалізація класу сутності для таблиці «schedule-items»

Декоратор «@Entity» необхідний для позначення сутності в NestJS.

Декоратор «@Column» вказує, що властивість є колонкою в таблиці.

Декоратор «@ManyToOne» забезпечує встановлення зв'язків «багато-до-одного» з вказаними сутностями в конфігураційних параметрах («Group», «Coach», «Athlete») через відповідні колонки зв'язку («group_id», «coach_id», «athlete_id»).

Сутність «ScheduleItem» успадковує клас «BaseEntity», який містить поля: «id», «createdAt», «updatedAt» та «deletedAt» (рисунок 3.9).

```
export abstract class BaseEntity {
  @PrimaryGeneratedColumn( strategy: 'uuid')
  public id: string;

  @CreateDateColumn( options: { name: 'created_at' })
  public createdAt: Date;

  @UpdateDateColumn( options: { name: 'updated_at' })
  public updatedAt: Date;

  @DeleteDateColumn( options: { name: 'deleted_at' })
  public deletedAt: Date;
}
```

Рисунок 3.9 — Реалізація базового класу сутностей

Спадкування від базового класу реалізовано для кожної сутності застосунку. Основна мета наслідування класу «BaseEntity» полягає в уникненні дублювання коду і забезпеченні єдності в структурі додатку.

3.2.6 Перехоплювач відповідей

Перехоплювачі в NestJS (interceptors) — це класи, які надають можливість перехоплювати вхідні й вихідні HTTP-запити для впровадження власної / спеціальної логіки, наприклад, обробки помилок чи модифікації структури запиту [18].

Для серверної частини застосунку «Go^Fit» створено клас «ResponseInterceptor», метою якого є стандартизація формату відповіді та обробка помилок (лістинг коду наведено в додатку А).

Клас «ResponseInterceptor» реалізує інтерфейс «NestInterceptor» і містить такі методи:

- intercept (перехоплює запит);
- responseHandler (обробляє успішні відповіді, формуючи стандартизований об'єкт);
- errorHandler (обробляє помилки, формуючи стандартизований об'єкт з інформацією про помилку).

3.2.7 Декоратори завантаження файлів

Користувацькі декоратори в NestJS — це функції, які надають можливість змінити або доповнити поведінку класів, методів чи властивостей. Декоратори можуть бути застосовані до контролерів, сервісів або будь-яких інших частин NestJS-застосунку [19].

Веб-додаток «Go/Find» передбачає завантаження зображень користувачів і Find-товарів. Для спрощення процесу завантаження файлів через HTTP-запити реалізовано користувацькі декоратори «ApiFile» та «ApiImageFile».

Декоратор «ApiFile» призначений для завантаження будь-яких файлів (рисунок 3.10).

```
export function ApiFile(
  fieldName: string = 'file',
  requiredFieldKeys: string[] = [],
  extraProperties: object = {},
  localOptions?: MulterOptions,
): <TFunction extends Function, Y>(target: (object | TFunction), propertyKey?: (string | symbol), descriptor?: TypedPropertyDescriptor<Y>) => void {
  return applyDecorators(
    UseInterceptors(FileInterceptor(fieldName, localOptions)),
    ApiConsumes( mimeTypes: 'multipart/form-data'),
    ApiBody( options: {
      schema: {
        type: 'object',
        required: requiredFieldKeys ? requiredFieldKeys : [],
        properties: {
          ...extraProperties,
          [fieldName]: { type: 'string', format: 'binary' },
        },
      },
    }),
  );
}
```

Рисунок 3.10 — Реалізація функції-декоратора «ApiFile»

Функція приймає такі параметри:

- назву параметра, в якому буде передано ім'я файлу в запиті («fieldName»);
- перелік ключів обов'язкових полів запиту («requiredFieldKeys»);
- додаткові властивості конфігурації («extraProperties»);
- опції завантаження файлу («localOptions»).

Декоратор «ApiImageFile» призначений виключно для завантаження зображень. Функція здійснює виклик «ApiFile» з аргументом фільтрації MIME-типу [20] «image» (рисунок 3.11).

```
export function ApiImageFile(
  fileName: string = 'image',
  requiredFieldKeys: string[],
  extraProperties: object = {},
): <TFunction extends Function, Y>(target: (object | TFunction), propertyKey?: (string | symbol), descriptor?: TypedPropertyDescriptor<Y>) => void {
  return ApiFile(fileName, requiredFieldKeys, extraProperties, localOptions: {
    fileFilter: fileMimeTypeFilter( mimetypes: 'image'),
  });
}
```

Рисунок 3.11 — Реалізація функції-декоратора «ApiImageFile»

Застосування декоратора «ApiImageFile» до кінцевої точки (endpoint) POST «/api/user/avatar-upload» подано на рисунку 3.12.

```
@Post( path: 'avatar-upload')
@ApiImageFile( fileName: 'image', requiredFieldKeys: ['userId'], extraProperties: { userId: { type: 'string' } })
public async uploadFile(
  @UploadedFile() file: Express.Multer.File,
  @Req() request: Request,
): Promise<string> {
  return await this.userService.uploadAvatar(request.body['userId'], file);
}
```

Рисунок 3.12 — Застосування декоратора «ApiImageFile» до методу завантаження фото профілю користувача

Використання користувацьких декораторів завантаження файлів забезпечує стандартизований процес їхньої обробки на серверній частині застосунку.

3.2.8 Документування API

Документування API є важливою складовою в розробці програмного забезпечення, оскільки сприяє коректній інтеграції з іншими сервісами й системам, полегшує розуміння процесів і знижує ризик виникнення помилок.

Головними принципами документування API є:

- опис методів (призначення і функціональність);
- опис шляхів з вказівкою HTTP-типів запитів (GET, POST, PUT, DELETE);
- опис параметрів запитів (повний перелік з позначенням типів даних та обов'язковості кожного з параметрів);
- опис формату відповідей (можливі значення або помилки, які повертаються методом).

Одним із сучасних підходів до написання API-документації є автоматична генерація документа за допомогою спеціальних інструментів, прикладом якого є Swagger [16].

Перевагою використання інструмента Swagger є те, що документовані фрагменти коду завжди залишаються в актуальному / оновленому стані, незалежно від внесених змін у вихідний код. Ще однією зручною опцією, яку надає Swagger, є можливість тестування реалізованих методів (endpoints) безпосередньо з браузера.

Для налаштування з'єднання Swagger із застосунком «Go^Fit» було встановлено пакет залежності «@nestjs/swagger». У головний файл серверної частини застосунку було додано конфігураційні дані, подані на рисунку 3.13.

```
const config = new DocumentBuilder()
  .setTitle('GoAFit')
  .setDescription('The GoAFit API description')
  .setVersion('1.0')
  .addTag({ name: 'go_a_fit' })
  .build();
const document = SwaggerModule.createDocument(app, config);

SwaggerModule.setup({ path: 'api', app, document});
```

Рисунок 3.13 — Конфігурація інструмента Swagger на серверній частині застосунку

Серед базових властивостей конфігурації — назва, опис, тег, версія API, а також шлях, за яким документація буде доступною в браузері.

Приклад документування методу, призначеного для завантаження фото профілю користувача (кінцева точка «/api/users/avatar-upload») подано на рисунку 3.14.

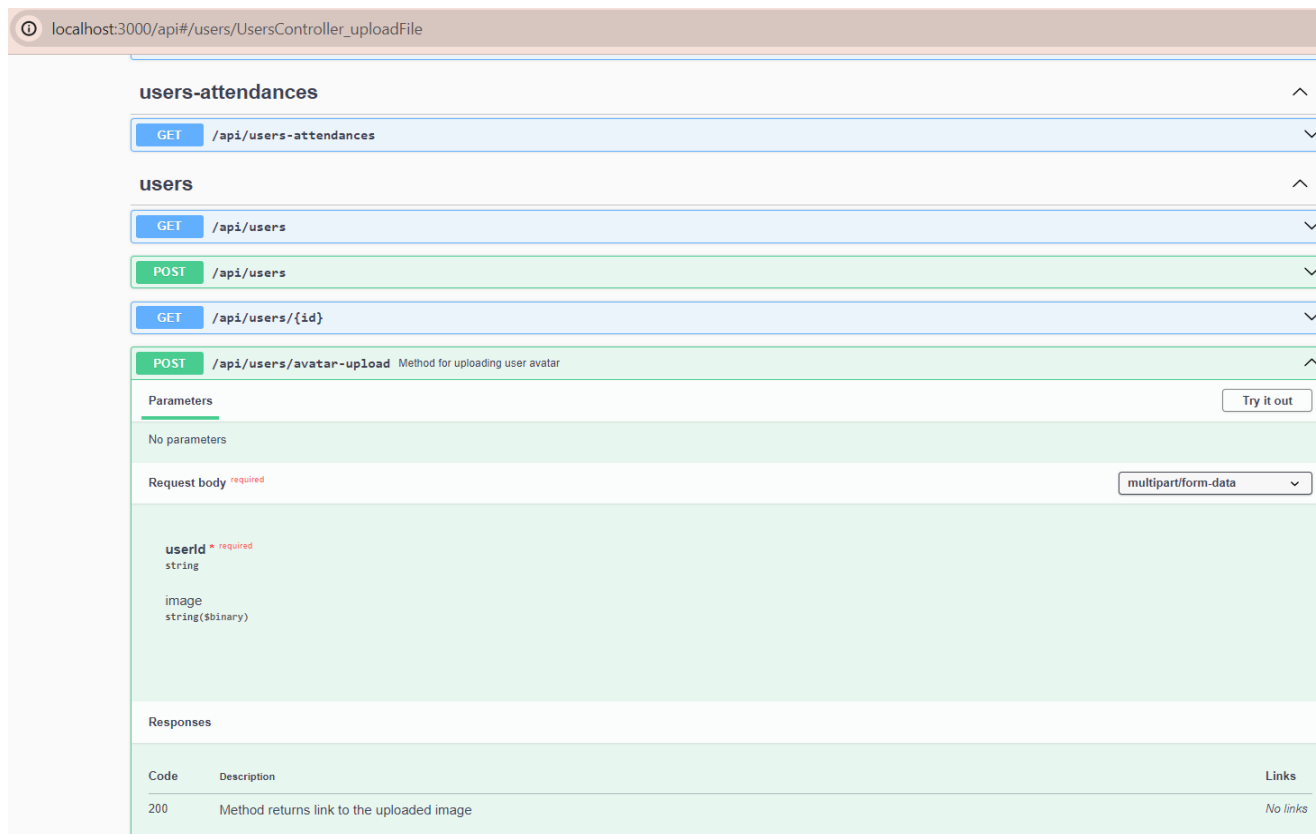


Рисунок 3.14 — Документована кінцева точка (endpoint)
«/api/users/avatar-upload» за допомогою інструмента Swagger

Використання інструмента Swagger для документування серверної частини застосунку «Go\Fit» надає актуальну інформацію щодо доступних кінцевих точок (endpoints) та їхніх параметрів.

4 РОБОТА КОРИСТУВАЧА З ВЕБ-ЗАСТОСУНКОМ

У розділі описано основні елементи інтерфейсу та етапи взаємодії користувача з веб-застосунком «Go\Fit».

Компонентами, які забезпечують користувачеві доступ до базових функцій веб-застосунку є:

- сторінка входу;
- головна сторінка;
- сторінка «Розклад»;
- діалогове вікно створення / редагування подій у розкладі;
- сторінка бонусних товарів «Fit-Gifts»;
- діалогове вікно «Кошик»;
- сторінка перегляду активності користувача;
- сторінка «Тренери»;
- сторінка «Відвідуваність»;
- сторінка «Користувачі».

4.1 Сторінка входу

На сторінці розміщено форму для введення системного імені користувача і пароля.

Макет сторінки передбачає додаткові опції, які стосуються збереження облікових даних для автоматичного заповнення форми при наступних входах, відновлення пароля і звернення до адміністрації з метою створення облікового запису, оскільки самотійна реєстрація не доступна користувачам. Інтерфейс сторінки входу зображено на рисунку 4.1.

Після заповнення форми з обліковими даними, користувач натискає кнопку «Увійти» (відправка форми також здійснюється при натисненні клавіші «Enter»).

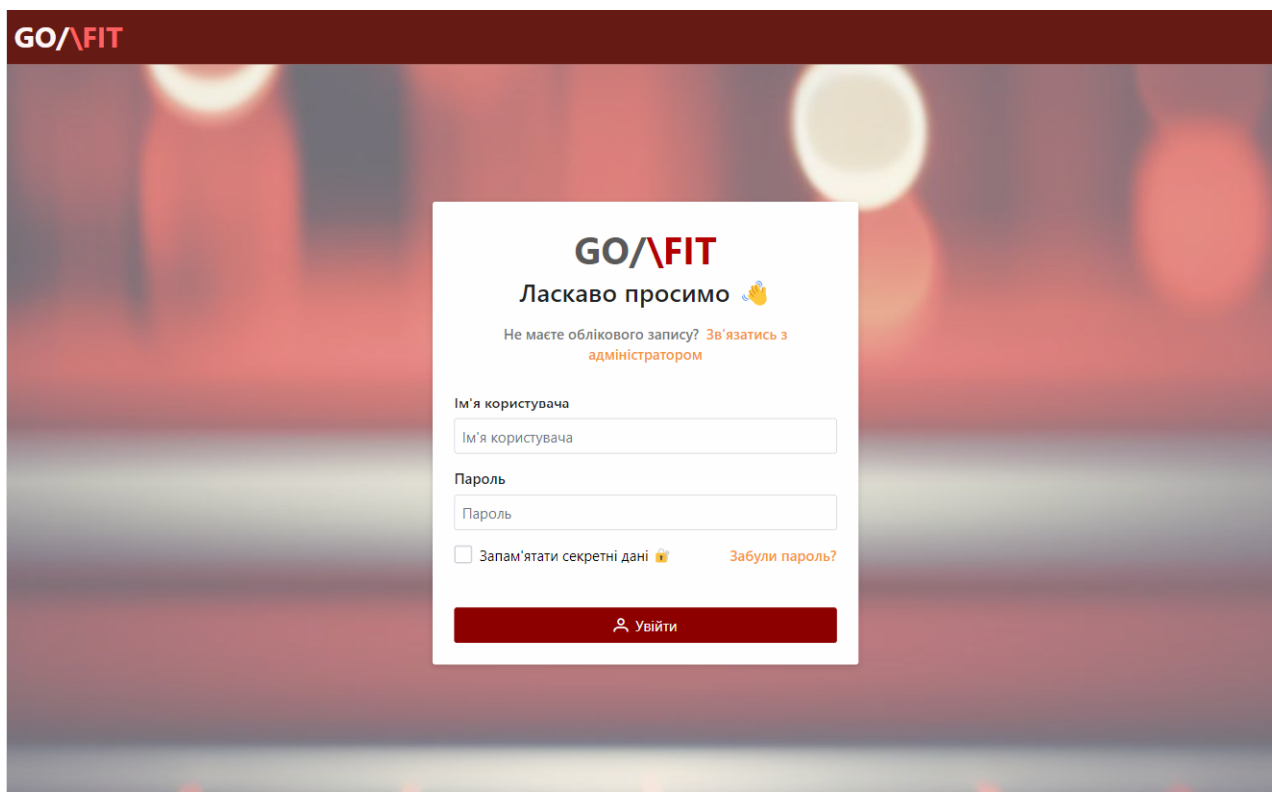


Рисунок 4.1 — Інтерфейс сторінки входу користувача

У випадку успішного проходження процесу автентифікації (перевірки справжності особи, яка намагається увійти до облікового запису), користувач автоматично переадресовується на головну сторінку застосунку.

4.2 Головна сторінка

На головній сторінці відображається персональна інформація авторизованого в системі користувача (ПІБ, дата народження, електронна адреса, телефон, фото профілю), сума накопичених користувачем бонусів, метрики щодо використання активного тренувального пакета (абонементу). Вищевказані дані отримуються з бази даних, за винятком товарів, які користувач додав до кошика (використано веб-сховище, яке надає можливість зберігати інформацію без часових обмежень, навіть після перезавантаження сторінки чи перевідкриття браузера). Інтерфейс головної сторінки подано на рисунку 4.2.

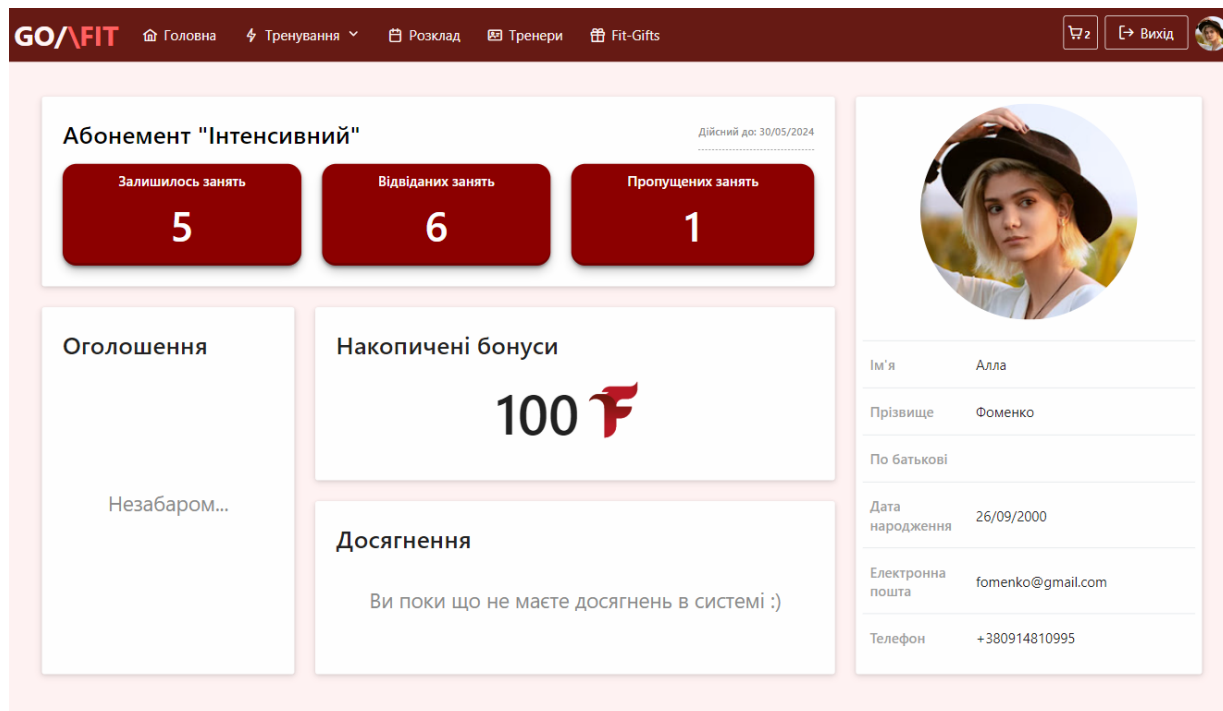


Рисунок 4.2 — Інтерфейс головної сторінки веб-застосунку «Go\Fit»

У верхній частині екрана розміщено меню, яке дає можливість здійснювати навігацію між сторінками застосунку. Кнопка з піктограмою кошика відкриває діалогове вікно з обраними користувачем спортивними товарами. Кнопка «Вихід» скасовує автентифікацію користувача в системі. У такому режимі доступ до перегляду сторінок обмежений, користувач автоматично переадресовується на сторінку «Вхід».

4.3 Сторінка «Розклад»

Для сторінки реалізовано два режими перегляду «Місяць» і «Тиждень».

Кожен елемент розкладу містить в описі дату й час початку і закінчення події, назву і тип («Групове заняття», «Індивідуальне заняття», «Вихідний», «Змагання», «Інше»).

Кожен елемент розкладу позначено асоціативним кольором, який може відповідати, наприклад, кольору групи. Сьогоднішній день виділено на календарі блідо-рожевим кольором. Інтерфейси сторінки «Розклад» в обох режимах перегляду зображено на рисунках 4.3 і 4.4.

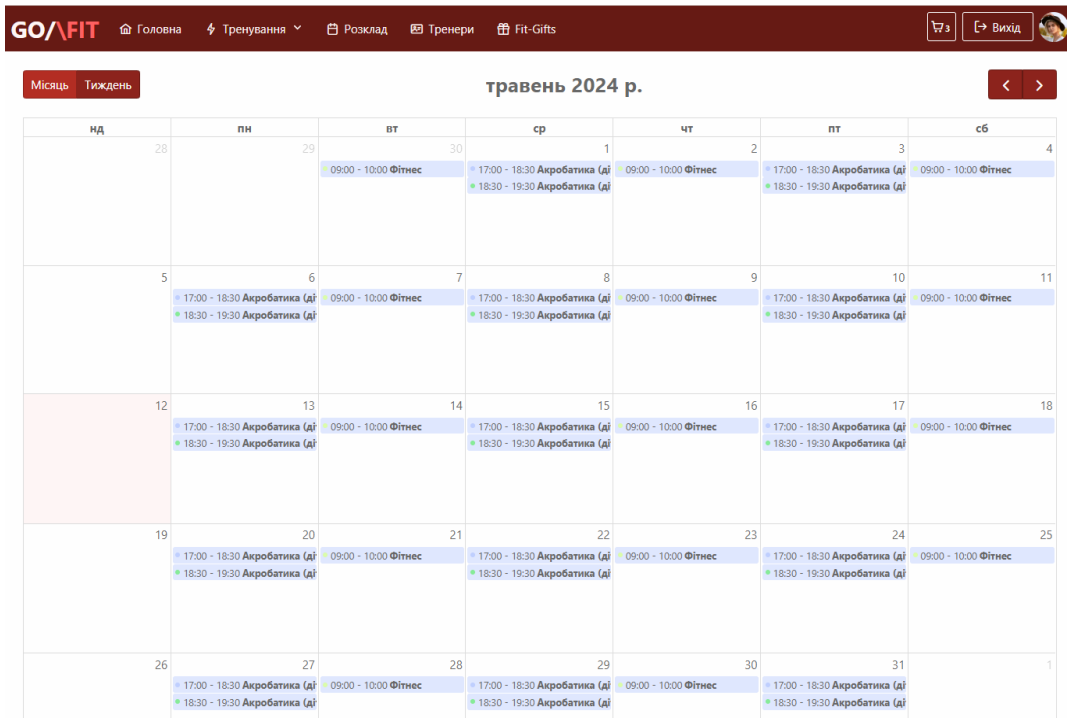


Рисунок 4.3 — Інтерфейс сторінки «Розклад» в режимі перегляду «Місяць»

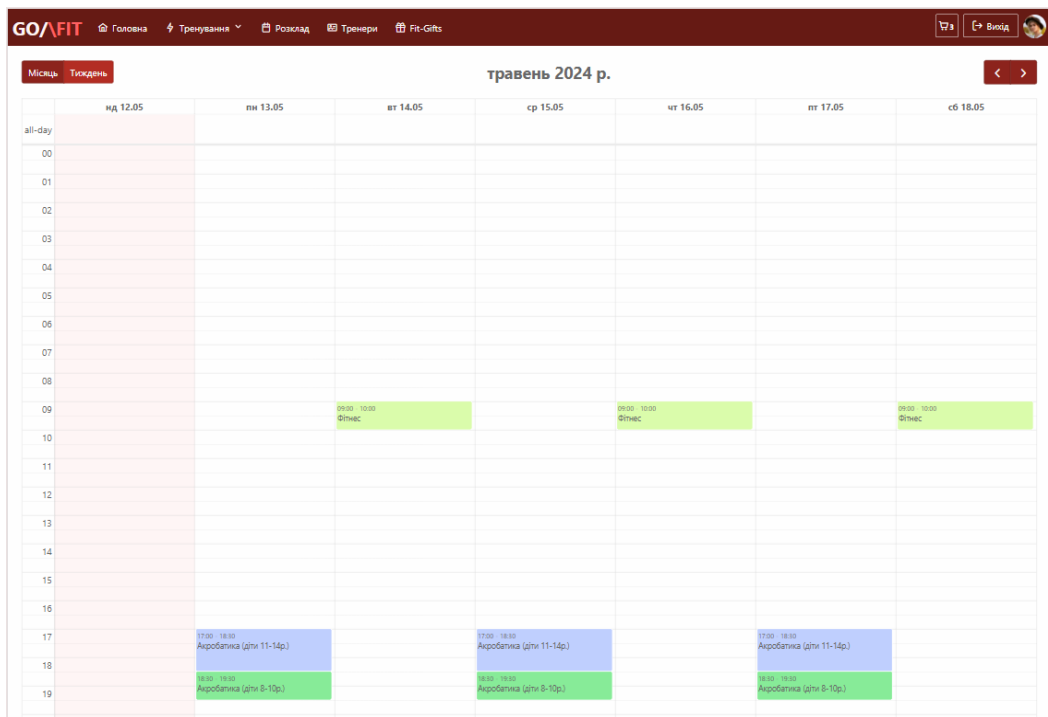


Рисунок 4.4 — Інтерфейс сторінки «Розклад» в режимі перегляду «Тиждень»

Модифікувати події розкладу або створювати нові можуть лише користувачі із системною роллю «Адміністратор». Функція редагування або створення події в розкладі стає доступною при натисканні на існуючий елемент розкладу або комірку дня / часу відповідно.

4.4 Діалогове вікно створення / редагування подій у розкладі

Функції створення і редагування подій у розкладі є доступними лише користувачам із системною роллю «Адміністратор». Інтерфейс діалогового вікна подано на рисунку 4.5.

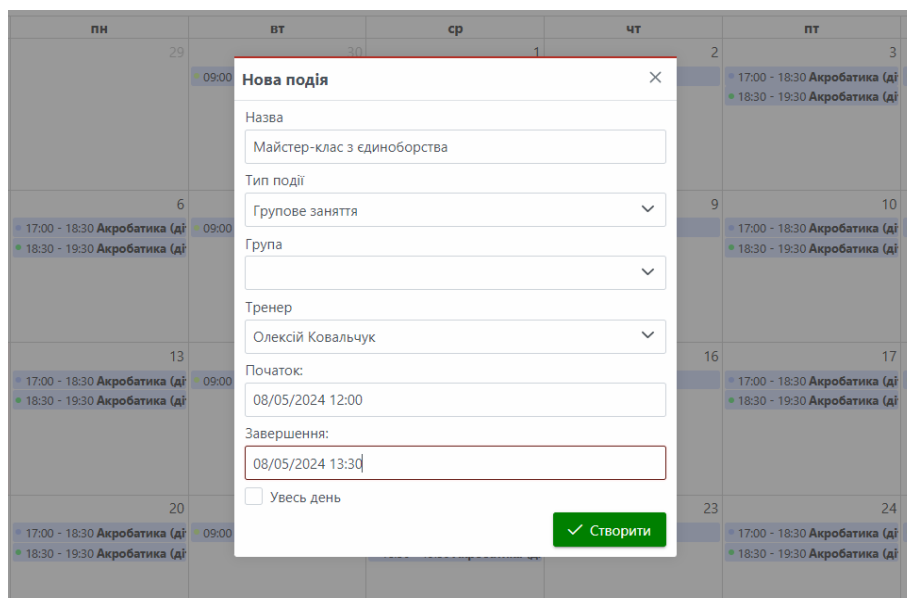


Рисунок 4.5 — Діалогове вікно створення події в розкладі

Відображення елементів у формі створення / редагування події розкладу є динамічним (рисунок 4.6).

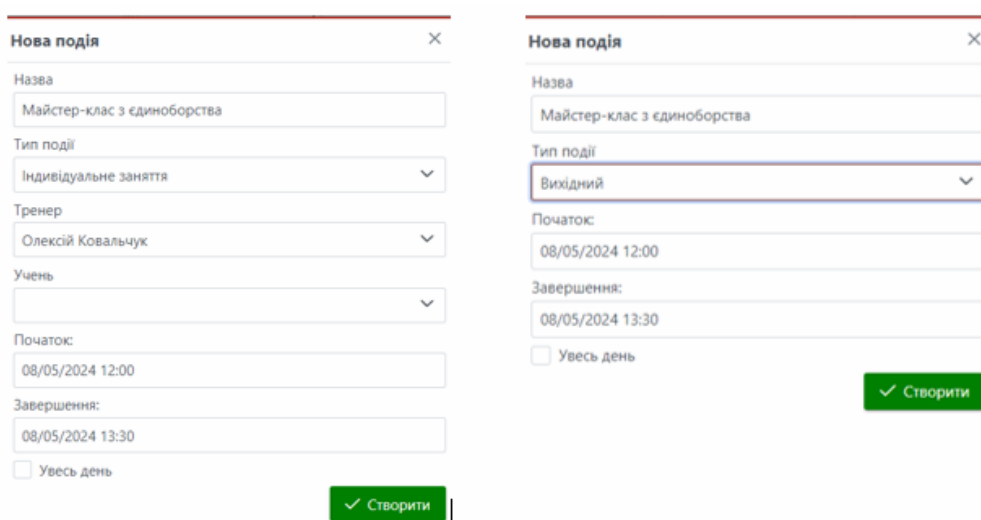


Рисунок 4.6 — Динамічне відображення елементів форми в залежності від обраного типу події

Наприклад, при виборі значення «Індивідуальне заняття» (селектор «Тип події») — поле «Група» замінюється вибором значення в полі «Учень»; при виборі значення «Вихідний» (селектор «Тип події») — поля «Тренер», «Група», «Учень» приховуються.

4.5 Сторінка «Fit-Gifts»

Для сторінки бонусних товарів «Fit-Gifts» реалізовано два режими перегляду: «Сітка» і «Список». Інтерфейс сторінки «Fit-Gifts» в обох режимах перегляду зображено на рисунках 4.7 і 4.8.

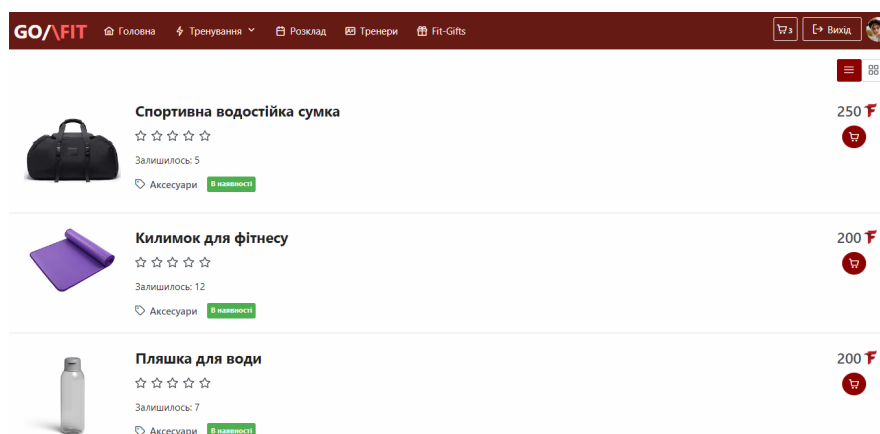


Рисунок 4.7 — Інтерфейс сторінки «Fit-Gifts» в режимі перегляду «Список»

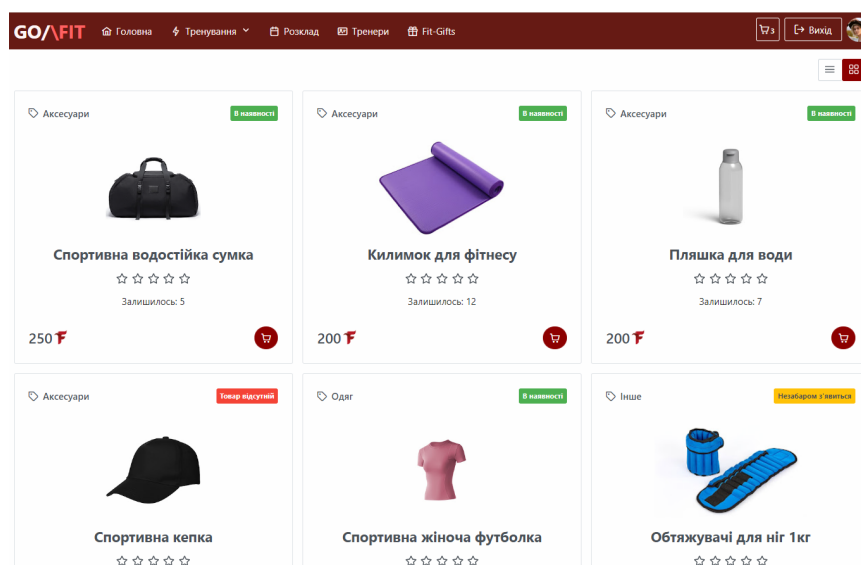


Рисунок 4.8 — Інтерфейс сторінки «Fit-Gifts» в режимі перегляду «Плитка»

Кожна позиція (товар) характеризується статусом доступності, залежно від якого кнопка «Додати до кошика» перебуває в активному або вимкненому стані.

4.6 Діалогове вікно «Кошик»

Вікно «Кошик» призначене для відображення товарів, які користувач обрав для обміну за накопиченими віртуальними валютними одиницями («Fit-Coins»). У вікні можна переглянути назви товарів, їхню вартість (у віртуальній валюті) і кількість. Загальна вартість замовлення автоматично перераховується при змінах кількості одиниць товару. Інтерфейс діалогового вікна зображено на рисунку 4.9.

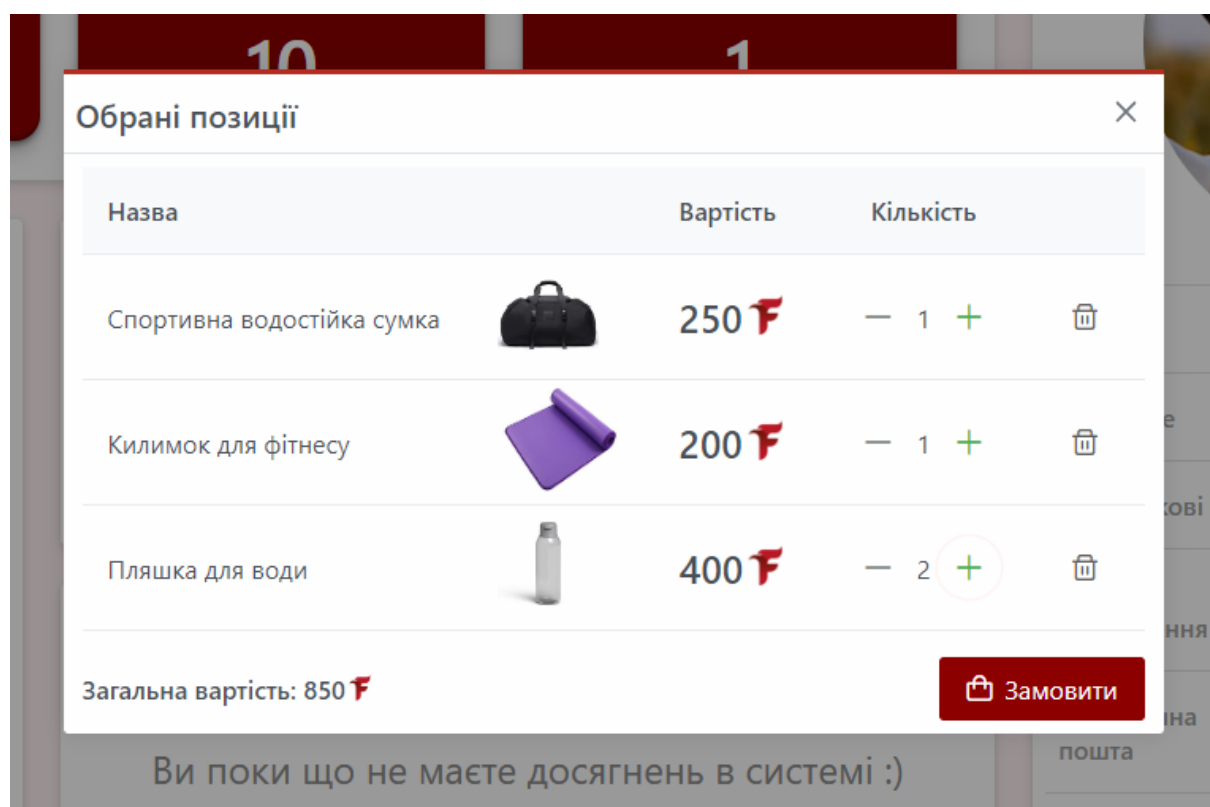


Рисунок 4.9 — Діалогове вікно з доданими користувачем товарами до кошика

При додаванні товару до кошика для користувача існує обмеження щодо наявності й доступної кількості одиниць. У випадку недостатньої суми бонусних коштів, система блокує можливість створення замовлення та інформує користувача про це у вигляді повідомлення, яке спливає (рисунок 4.10).

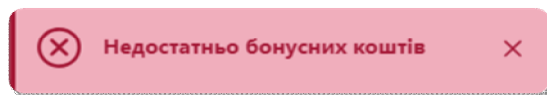


Рисунок 4.10 — Приклад інформування у вигляді повідомлення, яке спливає

Застосування повідомлень, що спливають, підвищує зручність користувачів, забезпечуючи оперативний зворотний зв'язок та сприяє позитивному користувацькому досвіду.

4.7 Сторінка перегляду активності користувача

Сторінка доступна для відображення користувачам із системною роллю «Атлет». У рамках дипломної роботи реалізовано демонстраційну версію сторінки. Дані на діаграмі є статичними, проте інтерактивними. При наведенні курсора на стовпчик гістограми з'являється підказка з кількістю відвіданих або пропущених занять за конкретним місяцем. Інтерфейс сторінки представлено на рисунку 4.11.

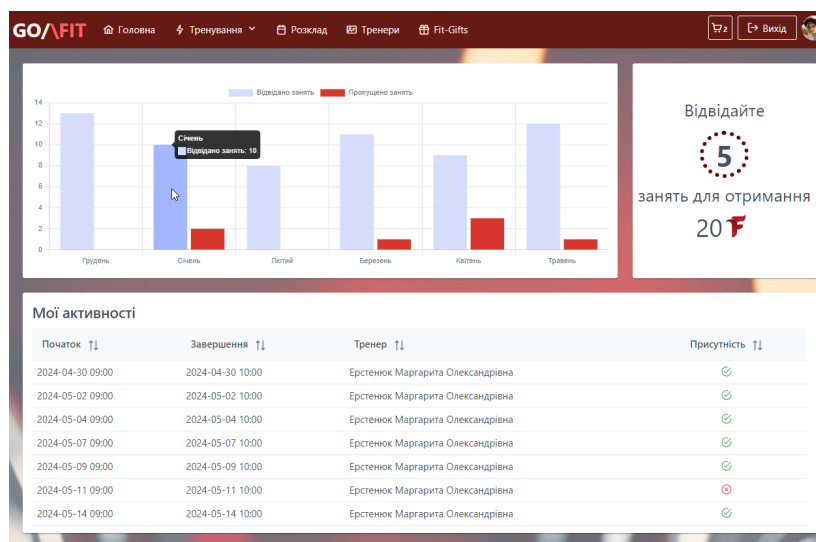


Рисунок 4.11 — Інтерфейс сторінки активності користувача з системною роллю «Атлет»

Користувач має змогу приховати відображення рядів даних, натиснувши на відповідний надпис у легенді (рисунок 4.12).

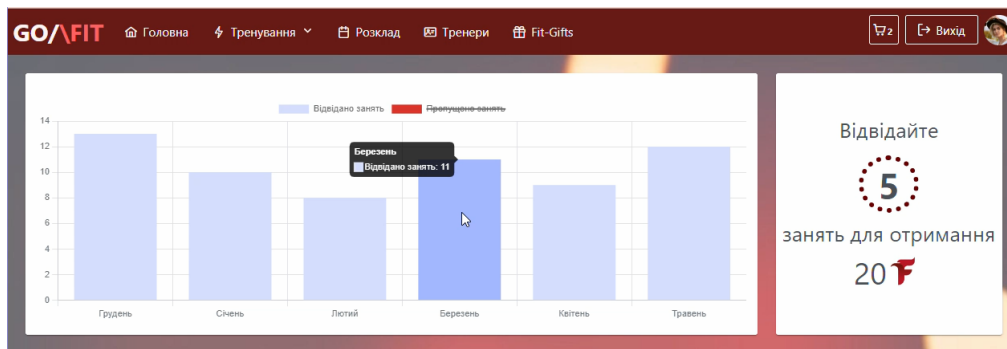


Рисунок 4.12 — Інтерактивна форма відображення даних на гістограмі активності користувача (приховування ряду даних «Пропущені заняття»)

Нижче від стовпчикової діаграми розміщено таблицю, яка містить інформацію про тренування, заплановані для атлета, і про ті, які вже відбулися. У таблиці, яка містить дані про час / дату початку й закінчення занять, ПІБ тренера, відмітку про присутність атлета на тренуванні, існує можливість сортування.

4.8 Сторінка «Тренери»

Сторінка призначена для відображення відомостей про тренерів, включаючи напрям спортивної діяльності, освіту, досягнення і гасло. Інтерфейс сторінки подано на рисунку 4.13.

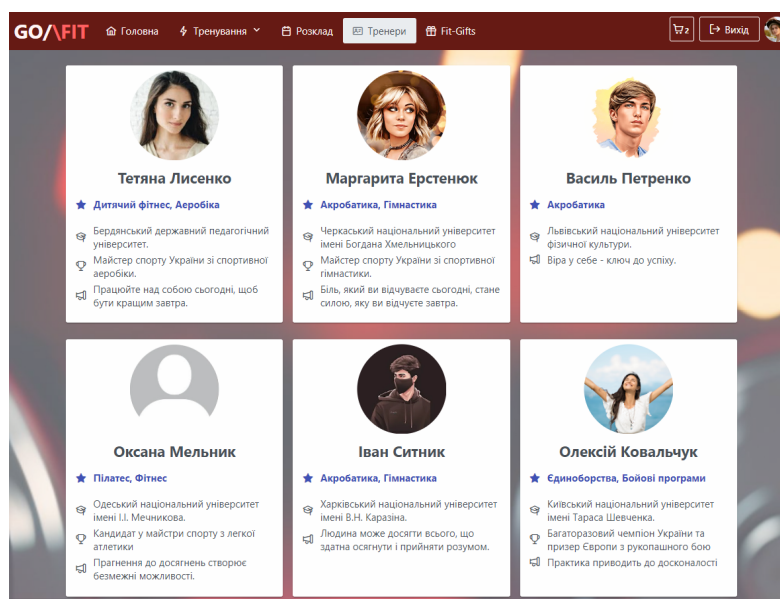


Рисунок 4.13 — Інтерфейс сторінки «Тренери»

4.9 Сторінка «Відвідуваність»

Сторінка ведення обліку відвідуваності доступна користувачам із системними ролями «Адміністратор» і «Тренер». Інтерфейс сторінки зображено на рисунку 4.14.

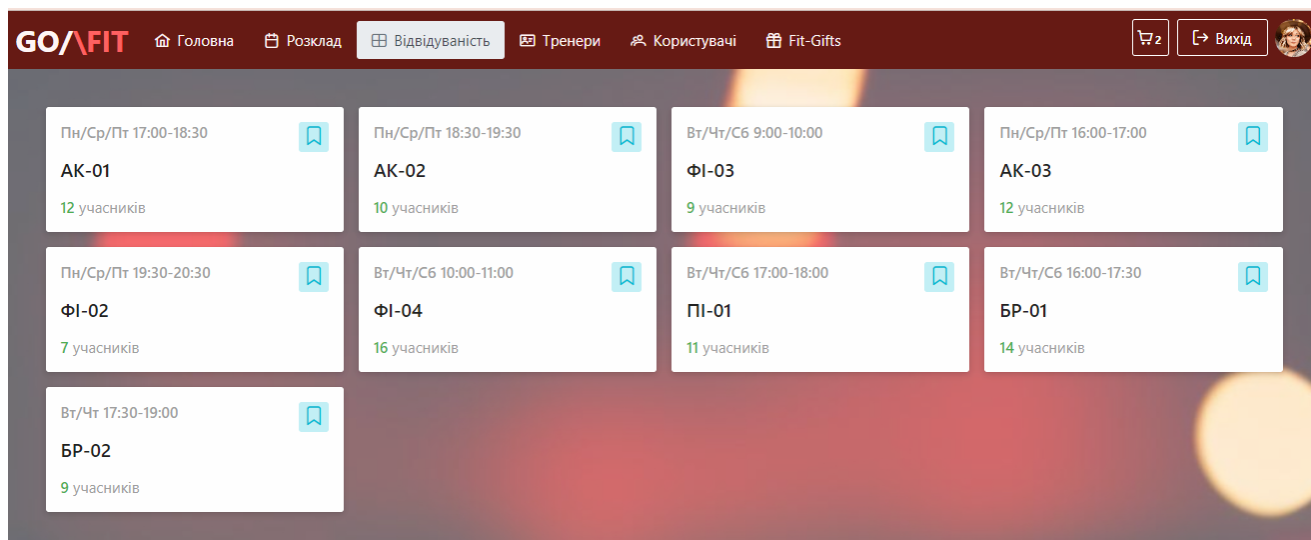


Рисунок 4.14 — Інтерфейс сторінки «Відвідуваність» для користувачів із системними ролями «Адміністратор» і «Тренер»

При натисканні на блок групи відбувається перехід до таблиці з відмітками про присутність / відсутність атлетів обраної групи на заняттях (рисунок 4.15).

ПІБ	02/05	04/05	07/05	09/05	11/05	14/05	16/05	18/05	21/05	23/05	25/05	28/05	30/05
Усенко І.	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Фоменко А.	☒	☒	☒	☒	☒	☒	☐	☐	☐	☐	☐	☐	☐
Гнатюк М.	☐	☒	☒	☒	☐	☒	☐	☐	☐	☐	☐	☐	☐
Козлов В.	☒	☐	☒	☒	☒	☒	☐	☐	☐	☐	☐	☐	☐
Морозова О.	☒	☒	☒	☒	☒	☒	☐	☐	☐	☐	☐	☐	☐
Петренко М.	☐	☒	☒	☒	☒	☒	☐	☐	☐	☐	☐	☐	☐
Кравчук А.	☒	☒	☐	☒	☒	☒	☐	☐	☐	☐	☐	☐	☐
Гриневич В.	☒	☒	☒	☒	☒	☒	☐	☐	☐	☐	☐	☐	☐

Рисунок 4.15 — Інтерфейс сторінки ведення обліку відвідуваності за конкретною групою та поточним місяцем

Адміністратори здатні вести облік відвідуваності для всіх груп системи. Тренери ведуть облік присутності / відсутності лише в групах, що закріплені за ними. Групи, які не призначені конкретному тренеру, не відображатимуться на його екрані.

4.10 Сторінка «Користувачі»

Сторінка доступна користувачам із системною роллю «Адміністратор». Інтерфейс сторінки представлено на рисунку 4.16.

☐	Прізвище ↑↓	Ім'я ↑↓	По батькові ↑↓	Телефон	Системна роль ↓↑	
☐	Лисенко	Тетяна		+380951234567	Тренер	
☐	Петренко	Василь		+380991234567	Тренер	
☐	Мельник	Оксана		+380686302468	Тренер	
☐	Ситник	Іван		+380977654321	Тренер	
☐	Ковальчук	Олексій		+380505050505	Тренер	
☐	Мазур	Антон		+380936172490	Атлет	
☐	Шевченко	Галина		+380934145347	Атлет	
☐	Гончар	Надія		+380934457453	Атлет	
☐	Павленко	Олег		+380739548457	Атлет	
☐	Усенко	Іванна		+380736376433	Атлет	
☐	Фоменко	Алла		+380914810995	Атлет	
☐	Гнатюк	Марина		+380931234567	Атлет	

Рисунок 4.16 — Інтерфейс сторінки «Користувачі»

У таблиці є можливості сортування і пошуку за атрибутами «Прізвище», «Ім'я», «Системна роль».

Інтерфейс сторінки передбачає можливість виконання CRUD-операцій для керування даними про користувачів.

ВИСНОВКИ

При виконанні дипломної роботи реалізовано такі завдання:

1) проаналізовано сучасні підходи та інструменти, які використовують для організації діяльності спортивних закладів;

2) спроєктовано архітектуру веб-застосунку для організації діяльності спортивної школи, що включає концептуальну модель бази даних, засоби і підходи до розробки;

3) розроблено веб-застосунок «Go\Fit», який реалізує такі функції за категоріями користувачів:

- Адміністратор — керування даними про користувачів, модифікація подій у розкладі, ведення обліку відвідуваності на заняттях;

- Тренер — перегляд розкладу занять, перегляд інформації про атлетів, перегляд каталогу подарункових товарів закладу, ведення обліку відвідуваності на заняттях (за закріпленими групами), обмін накопичених бонусів на подарункові товари / переваги спортивного закладу;

- Атлет — перегляд розкладу занять, перегляд відомостей про тренерів, перегляд каталогу подарункових товарів закладу; моніторинг статистики власної відвідуваності, моніторинг використання власного абонементу, обмін накопичених бонусів на подарункові товари / переваги спортивного закладу.

Веб-застосунок «Go\Fit» має потенціал для подальшого розвитку й розширення функціональних можливостей, серед яких:

- інтеграція з платіжними системами для онлайн-купівлі абонементів (тренувальних пакетів користувачів);

- інтеграція з поштовими серверами й месенджерами для отримання сповіщень про оновлення або новини закладу;

- інтеграція з системами аналізу даних: впровадження системи оцінок і відгуків, впровадження чату для комунікації між тренером та атлетом, впровадження модуля персональних рекордів / досягнень для відстеження прогресу спортсменів і встановлення нових цілей.

Результати роботи пройшли апробацію на XXI Міжнародній науково-практичній конференції молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики», м. Київ, 23-26 квітня 2024 року [1] (Додаток Б).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Холодницька А.Ю., Кублій Л.І. Веб-застосунок для організації діяльності спортивної школи. *Сучасні проблеми наукового забезпечення енергетики. У 2-х т.: Матеріали XXI Міжнародної науково-практичної конференції молодих вчених і студентів, м. Київ, 23-26 квітня 2024 року. Київ: КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2024. Т. 2. С. 289-290.*
2. Game-changing gym management software. URL: <https://www.glofox.com/gym-management-software/> (дата звернення: 12.05.2024).
3. Школа акробатики Jump Up. URL: <https://jump-up.com.ua/> (дата звернення: 12.05.2024).
4. Архітектура клієнт-сервер. URL: <https://www.geeksforgeeks.org/client-server-model/> (дата звернення: 12.05.2024).
5. Документація фреймворку Angular. URL: <https://angular.io/> (дата звернення: 12.05.2024).
6. Документація фреймворку NestJS. URL: <https://docs.nestjs.com/> (дата звернення: 12.05.2024).
7. Документація СКБД PostgreSQL. URL: <https://www.postgresql.org/> (дата звернення: 12.05.2024).
8. Транзакції в PostgreSQL. URL: <https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-transaction/> (дата звернення: 12.05.2024).
9. Документація ORM TypeORM. URL: <https://typeorm.io/> (дата звернення: 12.05.2024).
10. The Entity-Relationship Model — Toward a Unified View of Data / Peter PinShan Chen — Cambridge: Massachusetts Institute of Technology, 1976 — ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, Pages 9-36.
11. Angular architecture. The basic building blocks of Angular applications. URL: <https://v2.angular.io/docs/ts/latest/guide/architecture.html#:~:text=An%20Angular%20module%2C%20whether%20a,about%20decorators%20on%20the%20web> (дата звернення: 25.05.2024).

12. Angular Single Page Applications (SPA). URL: <https://blog.angular-university.io/why-a-single-page-application-what-are-the-benefits-what-is-a-spa/> (дата звернення: 25.05.2024).

13. Використання Angular маршрутів в односторінкових додатках. URL: <https://angular.io/guide/router-tutorial> (дата звернення: 12.05.2024).

14. Route Guards in Angular. URL: <https://raghuvardhankaranam.medium.com/route-guards-in-angular-c2c01fe6167b> (дата звернення: 12.05.2024).

15. DTO in JavaScript. URL: <https://flancer32.com/dto-in-javascript-3274a3063919> (дата звернення: 21.05.2024).

16. Документація інструмента документування та тестування API Swagger. URL: <https://swagger.io/> (дата звернення: 21.05.2024).

17. A Guide on Dependency Injection in NestJS. URL: <https://www.digitalocean.com/community/tutorials/a-guide-on-dependency-injection-in-nestjs> (дата звернення: 21.05.2024).

18. NestJS interceptors: Guide and use cases. URL: <https://blog.logrocket.com/nestjs-interceptors-guide-use-cases/> (дата звернення: 23.05.2024).

19. Custom Decorators in NestJS: A Practical Guide. URL: <https://peeyushjss.medium.com/custom-decorators-in-nestjs-a-practical-guide-2db9bc2d9db4> (дата звернення: 23.05.2024).

20. MIME-типи. URL: <https://html-css.co.ua/html/mime-type/> (дата звернення: 23.05.2024).

ДОДАТОК А

Фрагменти коду основних складових застосунку

«Веб-застосунок для організації діяльності спортивної школи»

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ІАТЕ_ЦТЕ_ТР_01

Мова програмування — TypeScript

Аркушів 8

Вміст файлу головного модуля клієнтської частини застосунку

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { MenuHeaderComponent } from './components/menu-header/menu-header.component';
import { MenubarModule } from 'primeng/menubar';
import { InputTextModule } from 'primeng/inputtext';
import { PanelMenuModule } from 'primeng/panelmenu';
import { StyleClassModule } from 'primeng/styleclass';
import { FullCalendarModule } from '@fullcalendar/angular';
import { ScheduleComponent } from './components/schedule/schedule.component';
import { HTTP_INTERCEPTORS, HttpClientModule } from '@angular/common/http';
import { ScheduleItemFormComponent } from './components/schedule/schedule-item-form/schedule-item-form.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { CheckboxModule } from 'primeng/checkbox';
import { ButtonModule } from 'primeng/button';
import { DialogModule } from 'primeng/dialog';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { InputMaskModule } from 'primeng/inputmask';
import { DropdownModule } from 'primeng/dropdown';
import { GlobalService } from './services/global.service';
import { ToastModule } from 'primeng/toast';
import { MessageService } from 'primeng/api';
import { LoginComponent } from './components/login/login.component';
import { RippleModule } from 'primeng/ripple';
import { ProgressSpinnerModule } from 'primeng/progressspinner';
import { SpinnerService } from './services/spinner.service';
import { UserComponent } from './components/users/user/user.component';
import { CalendarModule } from 'primeng/calendar';
import { InputGroupAddonModule } from 'primeng/inputgroupaddon';
import { InputGroupModule } from 'primeng/inputgroup';
import { DividerModule } from 'primeng/divider';
import { UsersComponent } from './components/users/users.component';
import { ToolbarModule } from 'primeng/toolbar';
import { TableModule } from 'primeng/table';
import { ConfirmDialogModule } from 'primeng/confirmdialog';
import { TagModule } from 'primeng/tag';
import { InvalidFormControlDirective } from './directives/invalid-form-control.directive';
import { ImageModule } from 'primeng/image';
import { FileUploadModule } from 'primeng/fileupload';
import { HomeComponent } from './components/home/home.component';
import { AuthInterceptor } from './interceptors/auth.interceptor';
import { AuthService } from './services/auth.service';
import { UserService } from './services/user.service';
import { ValidationMessageService } from './services/validation.service';
import { SystemRoleGuard } from './guards/system-role.guard';
import { AttendanceTrackerComponent } from './components/attendance-tracker/attendance-tracker.component';
import { GroupService } from './services/group.service';
import { FitProductsComponent } from './components/fit-products/fit-products.component';

```

```

import { DataViewModule } from 'primeng/dataview';
import { FitProductService } from './seviles/fit-product.service';
import { RatingModule } from 'primeng/rating';
import { AvatarModule } from 'primeng/avatar';
import { ChipModule } from 'primeng/chip';
import { TrainingPackageService } from './seviles/training-package.service';
import { UserBracketComponent } from './components/user-bracket/user-bracket.component';
import { BadgeModule } from 'primeng/badge';
import { FitOrderService } from './seviles/fit-order.service';
import { DialogService } from 'primeng/dynamicdialog';
import { CoachesComponent } from './components/coaches/coaches.component';
import { CardModule } from 'primeng/card';
import { PersonalActivitiesComponent } from './components/personal-activities/personal-activities.component';
import { ChartModule } from 'primeng/chart';
import { GroupActivityComponent } from './components/attendance-tracker/group-activity/group-activity.component';
import { UserAttendanceService } from './seviles/user-attendance.service';
import { DefaultImageDirective } from './directives/default-image.directive';

```

```

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    MenuHeaderComponent,
    ScheduleComponent,
    ScheduleItemFormComponent,
    LoginComponent,
    UsersComponent,
    UserComponent,
    CoachesComponent,
    PersonalActivitiesComponent,
    AttendanceTrackerComponent,
    GroupActivityComponent,
    FitProductsComponent,
    UserBracketComponent,
  ],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    CommonModule,
    HttpClientModule,
    AppRoutingModule,
    MenubarModule,
    InputTextModule,
    PanelMenuModule,
    StyleClassModule,
    FullCalendarModule,
    FormsModule,
    ReactiveFormsModule,
    CheckboxModule,
    ButtonModule,
    DialogModule,
    InputMaskModule,

```

```

    DropdownModule,
    ToastModule,
    RippleModule,
    ProgressSpinnerModule,
    CalendarModule,
    InputGroupAddonModule,
    InputGroupModule,
    DividerModule,
    ToolbarModule,
    TableModule,
    ConfirmDialogModule,
    TagModule,
    ImageModule,
    FileUploadModule,
    DataViewModule,
    RatingModule,
    AvatarModule,
    ChipModule,
    BadgeModule,
    CardModule,
    ChartModule,
    DefaultImageDirective,
    InvalidFormControlDirective,
  ],
  providers: [
    GlobalService,
    MessageService,
    DialogService,
    SpinnerService,
    AuthService,
    UserService,
    UserAttendanceService,
    TrainingPackageService,
    GroupService,
    FitProductService,
    FitOrderService,
    ValidationMessageService,
    SystemRoleGuard,
    { provide: HTTP_INTERCEPTORS, useClass: AuthInterceptor, multi: true },
  ],
  bootstrap: [AppComponent],
})
export class AppModule { }

```

Вміст файлу головного модуля серверної частини застосунку

```

import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { FitOrdersController } from './controllers/fit-orders.controller';
import { FitProductsController } from './controllers/fit-products.controller';
import { GroupsController } from './controllers/groups.controller';

```

```

import { NotificationsController } from './controllers/notifications.controller';
import { ScheduleItemsController } from './controllers/schedule-items.controller';
import { TrainingPackagesController } from './controllers/training-packages.controller';
import { UsersAttendancesController } from './controllers/users-attendances.controller';
import { UsersController } from './controllers/users.controller';
import { TypeOrmModule } from '@nestjs/typeorm';
import config, { GOOGLE_API_FOLDER_ID } from './configs/config';
import { TypeOrmExModule } from './typeorm-ex/typeorm-ex.module';
import { AthleteRepository } from './repositories/athlete.repository';
import { CoachRepository } from './repositories/coach.repository';
import { FitOrderRepository } from './repositories/fit-order.repository';
import { FitProductRepository } from './repositories/fit-product.repository';
import { GroupRepository } from './repositories/group.repository';
import { NotificationRepository } from './repositories/notification.repository';
import { ScheduleItemRepository } from './repositories/schedule-item.repository';
import { UserAttendanceRepository } from './repositories/user-attendance.repository';
import { TrainingPackageRepository } from './repositories/training-package.repository';
import { AthleteService } from './services/athlete.service';
import { CoachService } from './services/coach.service';
import { FitOrderService } from './services/fit-order.service';
import { FitProductService } from './services/fit-product.service';
import { GroupService } from './services/group.service';
import { NotificationService } from './services/notification.service';
import { ScheduleItemService } from './services/schedule-item.service';
import { UserAttendanceService } from './services/user-attendance.service';
import { TrainingPackageService } from './services/training-package.service';
import { UnitOfWorkService } from './services/unit-of-work.service';
import { UserRepository } from './repositories/user.repository';
import { CoachesController } from './controllers/coaches.controller';
import { AthletesController } from './controllers/athletes.controller';
import { UserService } from './services/user.service';
import { JwtModule } from '@nestjs/jwt';
import { GoogleDriveModule } from 'nestjs-googledrive-upload';
import { AuthController } from './auth/auth.controller';
import { AthletesGroupRepository } from './repositories/athletes-group.repository';
import { GymSubscriptionService } from './services/gym-subscription.service';
import { GymSubscriptionRepository } from './repositories/gym-subscription.repository';
import { GymSubscriptionsController } from './controllers/gym-subscriptions.controller';
import { ConfigModule } from '@nestjs/config';

```

```

const controllers = [
  AppController,
  AuthController,
  CoachesController,
  AthletesController,
  FitOrdersController,
  FitProductsController,
  GroupsController,
  NotificationsController,
  ScheduleItemsController,
  TrainingPackagesController,
  UsersAttendancesController,
  UsersController,
  GymSubscriptionsController,

```

```
];
```

```
const services = [
  AppService,
  UnitOfWorkService,
  AthleteService,
  CoachService,
  UserService,
  FitOrderService,
  FitProductService,
  GroupService,
  NotificationService,
  ScheduleItemService,
  UserAttendanceService,
  TrainingPackageService,
  GymSubscriptionService,
];
```

```
const repositories = [
  UserRepository,
  AthletesGroupRepository,
  AthleteRepository,
  CoachRepository,
  FitOrderRepository,
  FitProductRepository,
  GroupRepository,
  NotificationRepository,
  ScheduleItemRepository,
  UserAttendanceRepository,
  TrainingPackageRepository,
  GymSubscriptionRepository,
];
```

```
@Module({
  imports: [
    ConfigModule.forRoot({ isGlobal: true }),
    GoogleDriveModule.register(
      config.google,
      GOOGLE_API_FOLDER_ID,
    ),
    TypeOrmModule.forRoot(config.db),
    TypeOrmExModule.forCustomRepository(repositories),
    JwtModule.register({
      secret: 'secret',
      signOptions: { expiresIn: '1d' },
    }),
  ],
  controllers: [...controllers],
  providers: [...services],
})
export class AppModule { }
```

Вміст файлу «schedule-item.service.ts» серверної частини застосунку

```
import { Injectable } from '@nestjs/common';
import { BaseService } from '../base.service';
import { UnitOfWorkService } from '../unit-of-work.service';
import { ScheduleItem } from '../entities/schedule-item.entity';
import { ScheduleItemUpsertDto } from '../dtos/schedule-item/schedule-item-upsert.dto';
import { OccasionType } from '../enums/occasion-type';
import { Group } from '../entities/group.entity';
import { Coach } from '../entities/coach.entity';
import { Athlete } from '../entities/athlete.entity';
import { DateService } from '../date.service';

@Injectable()
export class ScheduleItemService extends BaseService {
  constructor(protected readonly unitOfWork: UnitOfWorkService) {
    super(unitOfWork);
  }

  public async findAll(): Promise<ScheduleItem[]> {
    return await this.unitOfWork.scheduleItemRepository.find({
      relations: { group: true, coach: true, athlete: true },
    });
  }

  public async create(scheduleItem: ScheduleItemUpsertDto): Promise<ScheduleItem> {
    const work = async () => {
      const scheduleItemToCreate: ScheduleItem = {
        start: DateService.formatToString(scheduleItem.start),
        end: DateService.formatToString(scheduleItem.end),
        title: scheduleItem.title,
        occasionType: scheduleItem.occasionType,
        isAllDay: scheduleItem.isAllDay,
      } as ScheduleItem;

      if (scheduleItem.occasionType === OccasionType.GroupTraining) {
        scheduleItemToCreate.group = { id: scheduleItem.groupId } as Group;
        scheduleItemToCreate.coach = { id: scheduleItem.coachId } as Coach;
      } else if (scheduleItem.occasionType === OccasionType.PersonalTraining) {
        scheduleItemToCreate.athlete = { id: scheduleItem.athleteId } as Athlete;
        scheduleItemToCreate.coach = { id: scheduleItem.coachId } as Coach;
      }

      return await this.unitOfWork.scheduleItemRepository.save(scheduleItemToCreate);
    };

    return await this.unitOfWork.doWork(work);
  }

  public async updateById(id: string, scheduleItem: ScheduleItemUpsertDto): Promise<ScheduleItem> {
    const work = async () => {
      const scheduleItemToUpdate = await this.unitOfWork.scheduleItemRepository.findById(id);
      if (!scheduleItemToUpdate) return;
    };
  }
}
```

```

scheduleItemToUpdate.start = DateService.formatToString(scheduleItem.start);
scheduleItemToUpdate.end = DateService.formatToString(scheduleItem.end);
scheduleItemToUpdate.title = scheduleItem.title;
scheduleItemToUpdate.occasionType = scheduleItem.occasionType;
scheduleItemToUpdate.isAllDay = scheduleItem.isAllDay;

if (scheduleItem.occasionType === OccasionType.GroupTraining) {
  scheduleItemToUpdate.group = { id: scheduleItem.groupId } as Group;
  scheduleItemToUpdate.coach = { id: scheduleItem.coachId } as Coach;
  scheduleItemToUpdate.athlete = null;
} else if (scheduleItem.occasionType === OccasionType.PersonalTraining) {
  scheduleItemToUpdate.athlete = { id: scheduleItem.athleteId } as Athlete;
  scheduleItemToUpdate.coach = { id: scheduleItem.coachId } as Coach;
  scheduleItemToUpdate.group = null;
} else {
  scheduleItemToUpdate.athlete = null;
  scheduleItemToUpdate.group = null;
  scheduleItemToUpdate.coach = null;
}

return await this.unitOfWork.scheduleItemRepository.save(scheduleItemToUpdate);
};

return await this.unitOfWork.doWork(work);
}

public async deleteById(id: string): Promise<void> {
  const work = async () => {
    const scheduleItem = await this.unitOfWork.scheduleItemRepository.findById(id);
    if (!scheduleItem) return;

    await this.unitOfWork.scheduleItemRepository.softRemove(scheduleItem);
  };

  return await this.unitOfWork.doWork(work);
}
}

```

Вміст файлу «response.interceptor.ts» серверної частини застосунку

```

import { Injectable, NestInterceptor, ExecutionContext, CallHandler, HttpException, HttpStatus }
from '@nestjs/common';
import { Observable, throwError } from 'rxjs';
import { catchError, map } from 'rxjs/operators';

export interface Response<T> {
  status: ResponseStatus,
  statusCode: number;
  message: string;
  result: T;
}

```

```

export enum ResponseStatus {
  Success = 'success',
  Error = 'error',
}

@Inject()
export class ResponseInterceptor implements NestInterceptor {
  public intercept(context: ExecutionContext, next: CallHandler): Observable<unknown> {
    return next.handle().pipe(map((res: unknown) => this.responseHandler(res, context)),
      catchError((err: HttpException) => throwError(() => this.errorHandler(err, context))));
  }

  public errorHandler(exception: HttpException, context: ExecutionContext): void {
    const ctx = context.switchToHttp();
    const response = ctx.getResponse();
    const status = exception instanceof HttpException ? exception.getStatus() :
    HttpStatus.INTERNAL_SERVER_ERROR;

    response.status(status).json({
      status: ResponseStatus.Error,
      statusCode: status,
      message: exception.message,
      result: exception,
    });
  }

  public responseHandler(res: any, context: ExecutionContext): Response<any> {
    const ctx = context.switchToHttp();
    const response = ctx.getResponse();
    const statusCode = response.statusCode;

    return {
      status: ResponseStatus.Success,
      statusCode,
      result: res,
      message: res?.message,
    };
  }
}

```

ДОДАТОК Б

Апробація

Матеріали XXI Міжнародної науково-практичної конференції молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики»

«Веб-застосунок для організації діяльності спортивної школи»

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ІАТЕ_ЦТЕ_ТР_01

Аркушів 5

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XXI Міжнародної
науково-практичної конференції
молодих вчених і студентів
м. Київ, 23–26 квітня 2024 року

ТОМ 2



Київ- 2024

Сервіс пошуку закладів харчування за смаковими вподобаннями користувача.	274
<i>ГУРИН І.А., бакалаврант</i>	
<i>Керівник - доц., к.т.н. Тихоход В.О.</i>	
Система обліку й аналізу ефективності роботи підприємства.	276
<i>ДЕМИДОВИЧ Д.П., бакалаврант</i>	
<i>Керівник - доц., к.т.н. Кублій Л.І.</i>	
Обмін інформацією системи Odoo з месенджером Telegram.	278
<i>ЗЕЛІНСЬКИЙ В.Я., бакалаврант</i>	
<i>Керівник - проф., д.т.н. Шушур О.М.</i>	
Система генерації мультимедійного контенту з використанням штучного інтелекту.	280
<i>КОНДРАШІН О.М., бакалаврант</i>	
<i>Керівник - доц., к.т.н. Тихоход В.О.</i>	
Веб-тренажер для багатокористувацької онлайн-гри Mobile Legends.	282
<i>КОРДЯК Р.М., бакалаврант</i>	
<i>Керівник - доц., к.т.н. Кублій Л.І.</i>	
Розробка лабораторного стенду для дослідження застосування методів ідентифікації пристроїв інтернету речей на базі архітектури цифрових об'єктів.	284
<i>КОРЧАН В.М., аспірант</i>	
<i>Керівник - проф., к.т.н. Морозова І.В.</i>	
Застосування великих мовних моделей для забезпечення якості освітньої діяльності кафедри.	286
<i>СУГУЛОВ Є.С., бакалаврант</i>	
<i>Керівник - доц., к.т.н. Коваль О.В.</i>	
Веб-застосунок для організації діяльності спортивної школи.	289
<i>ХОЛОДНИЦЬКА А.Ю., бакалаврант</i>	
<i>Керівник - доц., к.т.н. Кублій Л.І.</i>	
Веб-додаток обліку товарів в складському приміщені.	291
<i>ХОРОШИХ О.Л., бакалаврант</i>	
<i>Керівник - доц., к.е.н. Сегеда І.В.</i>	
Розробка сервісу веб застосунку забезпечення якості освітньої діяльності кафедри з використанням мовних моделей на базі апаратних засобів кафедри.	293
<i>ЧУКІН С.В., бакалаврант</i>	
<i>Керівник - асист. Голець В.О.</i>	
Розв'язання гравітаційної задачі N тіл за допомогою швидкого методу мультиполів.	295
<i>ЮРИК Д.М., бакалаврант</i>	
<i>Керівник - доц., к.т.н. Лабжинський В.А.</i>	
Засоби генерації текстових математичних задач шкільної програми з використанням Windows form.	298
<i>ЯКИМЕНКО Д.В., бакалаврант</i>	
<i>Керівник - доц., к.т.н. Кублій Л.І.</i>	
Веб-система пошуку та рейтингування закладів навчальних закладів.	300
<i>ЯЦЕНКО М.І., бакалаврант</i>	
<i>Керівник - доц., к.т.н. Тихоход В.О.</i>	

УДК 004.42:796.06

¹ Бакалаврант 4 курсу Холодницька А.Ю.

¹ Доц., к.т.н. Кублій Л.І.

<https://scholar.google.com.ua/citations?user=rXA1eMIAAAAJ&hl=uk>

¹ КПП ім. Ігоря Сікорського

ВЕБ-ЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ ДІЯЛЬНОСТІ СПОРТИВНОЇ ШКОЛИ

Постановка проблеми та її актуальність. У сучасному світі розвиток фізичних здібностей стає не лише запорукою успіху, але й важливим елементом самореалізації. Спорт відіграє важливу роль у формуванні здорового способу життя і розвитку особистості, допомагаючи зберегти енергію та зосередженість у потоці щоденних викликів. Послуги, спрямовані на розвиток та вдосконалення фізичних здібностей людей різного віку, від дошкільного до дорослого, надають спеціалізовані заклади — спортивні школи (у контексті даної роботи з поняттям “спортивна школа” ототожнено спортивні гуртки, секції, студії, центри, клуби тощо). В Україні існує багато спортивних закладів, переважна більшість яких для взаємодії з відвідувачами-спортсменами обмежується інформаційними сайтами, соціальними мережами і месенджерами. Ці засоби не забезпечують повноцінного керування та ефективної взаємодії з аудиторією. Таким чином, актуальною є розробка засобів організації діяльності спортивних шкіл.

Аналіз останніх досліджень. Один із сучасних аналогів такого інструменту — платформа Glofox [1], яка надає широкий функціонал для спортивних закладів, зокрема, керування розкладом занять, бронювання записів на тренування, ведення фінансової звітності, аналітику відвідуваності, а також можливість отримувати сповіщення про нові події та акції. Платформа є платною, вартість її використання визначається відповідно до індивідуальних потреб спортивного закладу. Деякі спортивні установи мають власні програмні засоби, орієнтовані на покращення процесів організації. Прикладом такого закладу є “Школа акробатики Jump Up” [2]. У режимі онлайн користувачі веб-сайту можуть ознайомитися з видами діяльності школи, переглянути графіки занять за напрямками, придбати абонемент, а також замовити дзвінок зворотного зв'язку.

Формулювання мети. Метою дослідження є розробка веб-застосунку, спрямованого на оптимізацію процесів керування у сфері спортивного навчання.

Основна частина. На основі існуючих рішень і з врахуванням потреб користувачів розроблений веб-застосунок розв'язує такі завдання:

- надає можливість керувати графіком тренувань — тренери та адміністратори можуть швидко й ефективно керувати розкладом занять, моніторити використання абонементів, а також вести облік відвідувань чи відсутностей на тренуваннях;

- надає можливість відвідувачам відпрацьовувати пропущені заняття завдяки можливості повідомляти про свою відсутність на занятті (не пізніше, ніж встановлена кількість годин/хвилин, визначена адміністрацією спортивного закладу в налаштуваннях веб-додатку). Своєчасне інформування про відсутність дає можливість відвідувачеві відпрацювати пропущене заняття (можливо, в іншій групі чи під керівництвом іншого тренера) протягом терміну дії абонементу. Для відпрацювання пропущеного заняття відвідувач створює запит, вказавши дату, час і групу, в якій бажає відвідати тренування. Після створення запиту адміністратор або тренер обраної групи підтверджує або, якщо досягнуто максимальної кількості тих, хто бажає відвідати тренування, відхиляє його;

- забезпечує легкий доступ до інформації. Відвідувачі, батьки юних вихованців мають можливість переглядати розклад занять, отримувати повідомлення про оновлення або зміни в графіку, переглядати відомості про тренерів спортивного закладу, включаючи професійний досвід, досягнення і спеціалізацію;

— відстежує прогрес відвідувачів, завдяки чому відвідувачі можуть переглядати статистику власної відвідуваності, записувати персональні рекорди, а також отримувати детальний звіт про досягнення за різними аспектами спортивного навчання;

— здійснює мотивацію через бонусну систему. Застосунок передбачає використання системи бонусів з метою стимуляції користувачів до нових досягнень і підтримки активного способу життя. Спортсмени мають можливість накопичувати FitCoin-и за високу частоту відвідування, досягнення різноманітних цілей та участь у спортивних подіях.

Для розробки веб-застосунку, спрямованого на організацію діяльності спортивної школи, обрано клієнт-серверну архітектуру, яка передбачає розділення функціоналу між клієнтською і серверною частинами [3]. Клієнтська частина відповідає за відображення інтерфейсу користувача і взаємодію з ним, тоді як серверна частина обробляє логіку бізнес-процесів, виконує запити до бази даних і забезпечує обмін інформацією з клієнтом.

Клієнтську частину веб-застосунку розроблено з використанням фреймворку Angular [4]. Обраний інструмент дає можливість структурувати код у вигляді компонентів, що значно покращує читабельність, полегшує підтримку і розвиток проєкту. В інтерфейсі користувача є розділи «Тренування» (з опціями перегляду активності, персональних рекордів, створення запитів на відпрацювання), «Розклад», «Тренери», «Події» та «Fit-Gifts» (система заохочень).

Серверну частину веб-застосунку розроблено з використанням фреймворку NestJS [5], який є доцільним для побудови надійних масштабованих серверних додатків на Node.js. Фреймворк поєднує в собі елементи об'єктно-орієнтованого програмування, функціонального програмування і функціонально-реактивного програмування. Оскільки фреймворк NestJS синтаксично та архітектурно схожий на Angular-фреймворк, обидва використовують мову програмування TypeScript, то поєднання їхнього використання дасть можливість спростити архітектуру застосунку в цілому, перевикористати код та зменшити кількість помилок завдяки суворій типізації.

Системою керування базами даних обрано PostgreSQL, яка має вбудований механізм транзакцій і забезпечує дотримання ACID-властивостей [6]. Використання механізму транзакцій є критично важливим для таких операцій, як модифікація розкладу (включаючи запис/скасування присутності відвідувача на тренуванні), обмін бонусів на заохочувальні презенти, оновлення персональних даних користувачів тощо. Такий підхід до керування даними забезпечить високий рівень їхньої цілісності і зменшуватиме ризик втрати інформації, що є необхідним для успішного функціонування веб-застосунку.

Взаємодію між PostgreSQL і NestJS здійснено через бібліотеку TypeORM, яка базується на технології програмування ORM (Object-relational mapping) [7]. Ця технологія надає можливість взаємодіяти з базою даних веб-застосунку, перетворюючи об'єкти програми на записи в базі даних і навпаки.

Висновки. Розроблений веб-застосунок сприятиме покращенню процесів керування роботою спортивних закладів, роблячи їх більш ефективними і зручними для тренерів, відвідувачів-спортсменів, батьків юних вихованців та адміністрації.

Перелік посилань:

1. Game-changing gym management software. URL: <https://www.glofox.com/gym-management-software/>
2. Школа акробатики Jump Up. URL: <https://jump-up.com.ua/>
3. Client-Server Model. URL: <https://www.geeksforgeeks.org/client-server-model/>
4. Angular: веб-сайт. URL: <https://angular.io/>
5. NestJS: веб-сайт. URL: <https://docs.nestjs.com/>
6. PostgreSQL Transaction. URL: <https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-transaction/>
7. TypeORM: веб-сайт. URL: <https://typeorm.io/>

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»



ДИПЛОМ

III ступеня

ЗА АКТИВНУ УЧАСТЬ
У **XXI МІЖНАРОДНІЙ НАУКОВО-ПРАКТИЧНІЙ КОНФЕРЕНЦІЇ**
МОЛОДИХ ВЧЕНИХ ТА СТУДЕНТІВ
«СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ»

нагороджується бакалаврантка 4-го курсу
кафедри цифрових технологій в енергетиці

Холодницька Аріна Юріївна

Директор НН ІА ЕЕ



Євген ПИСЬМЕННИЙ

2024

