ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА
(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем
(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»
(освітньо-кваліфікаційний рівень)

Застосування технологій інтернет для створення розподіленої системи підтримки діяльності фітнес-центру
Application of Internet technologies to create a distributed support system for fitness center activities

Виконав: студент денної форми навчання
спеціальності 126 – Інформаційні системи та технології
(шифр і назва напряму підготовки, спеціальності)
Освітня програма «Інформаційні системи та технології»
(назва освітньої програми)
Васеньшев Богдан Олегович
(прізвище, ім'я, по-батькові)

Керівник Розновець О.І. Подписано цифровой подписью: Olga Roznovets Дата: 2023.06.16 10:15:19 +03'00'
(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент к.т.н., доц. Волощук Л.А.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№___ від «___»_____ 2023 р.
Завідувач кафедри
_____ Євгеній МАЛАХОВ
(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___
протокол № __ від «_ _» ____ 2023 р.
Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)
Голова ЕК
_____ Володимир ВИЧУЖАНІН
(підпис) (ім'я, прізвище)

Одеса - 2023

АНОТАЦІЯ

Мета кваліфікаційної роботи – автоматизація бізнес-процесів фітнес-центру, що спеціалізується на збереженні і покращенні стану здоров'я клієнтів, за допомогою створення розподіленої інформаційної системи підтримки діяльності фітнес-центру. Користувачами розподіленої системи є клієнти та співробітники фітнес-центру (тренери, медпрацівники та менеджер). Реалізація системи виконана з використанням мов Java та JavaScript, СУБД PostgreSQL, сервера Apache Tomcat та бібліотек для розробки інтерфейсів користувача React і React Native.

Архітектура системи відповідає шаблону MVC. Взаємодія програмних компонентів заснована на дотриманні принципів архітектурного стилю REST.

У розробленій системі забезпечена автоматизація обліку клієнтів та персоналу фітнес-центру. Система дозволяє полегшити їх взаємодію та допомагає організувати ефективний менеджмент часу. Особливістю системи є надання тренеру функції формування індивідуальної програми тренувань з урахуванням фізіологічних особливостей клієнта, що робить тренування ефективними і безпечними.

Результатом кваліфікаційної роботи є розподілена інформаційна система підтримки діяльності фітнес-центру зі зручним інтерфейсом. Система є кросплатформною і призначена для роботи на персональних комп'ютерах та на мобільних пристроях під управлінням ОС Android та iOS.

ABSTRACT

The purpose of the graduation work is to automate the business processes of the fitness center, which specializes in maintaining and improving the health of clients, by creating a distributed support information system for fitness center activities.

Users of this distributed system are clients and employees of the fitness center (trainers, medical staff and manager). The system was implemented using the Java and JavaScript languages, the PostgreSQL DBMS, the Apache Tomcat server, and libraries for the development of React and React Native user interfaces.

The architecture of the information system corresponds to the MVC pattern. The interaction of software components is based on the principles of REST architectural style.

In the developed system accounting of customers and staff of the fitness center is automated. The system facilitates their interaction and helps organize effective time management. A feature of the system is providing the trainer with the function of forming an individual training program taking into account the client's physiological characteristics, which makes training effective and safe.

The result of the graduation work is a distributed support information system for fitness center activities with a convenient interface. The system is cross-platform and is designed to work on personal computers and mobile devices running Android and iOS.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

DTO (Data Transfer Object) – шаблон проектування, який визначає спосіб передачі даних між компонентами системи. DTO є об'єктом, який призначений лише для передачі даних та не містить будь-якої поведінки.

Entity class – клас, який представляє сутність в базі даних і відповідає за технічну інформацію про зв’язки між іншими сутностями і спосіб зберігання відповідної сутності.

JWT (JSON Web Token) – стандарт токенів, що використовується для передачі облікової інформації між клієнтом та сервером.

Хук – елемент фреймворку React, який дозволяє розробляти статичні чи динамічні компоненти та реагувати на них створенням функцій шляхом виклику певних методів захоплення з компонента.

ВСТУП

Проблема фізичного здоров'я актуальна завжди, а особливо в наш час, оскільки сучасна людина веде малорухливий спосіб життя, що призводить до різних проблем зі здоров’ям: біль у спині, варикозні розширення вен, головний біль, ожиріння та інші проблеми [1]. Через різноманітність проблем зі здоров'ям до кожної людини потрібен індивідуальний підхід.

Фітнес-центри та спортзали України пропонують послуги персонального тренера чи групові заняття. На жаль, не всі люди мають можливість і час відвідувати тренера та групові заняття, а деякі люди просто мають бажання займатися без тренера, але по інструкції. Дуже зручно, коли тренер може скласти програму тренувань клієнту та зберегти її в електронному вигляді для клієнта, супроводжуючи це описом кожної вправи. Застосовуючи технології інтернет, у будь-який час клієнт може скористатися програмою та переглянути вказівки тренера. Тренер може коригувати інструкції для клієнта та ускладнювати або, навпаки, спрощувати набір вправ.

Дуже часто клієнт не здатен адекватно оцінити можливості свого організму, особливо, якщо у минулому він отримував травми або у нього наявні хронічні захворювання. Збір і зберігання медичних даних клієнта та їх надання тренеру є дуже важливими чинниками для успішного формування індивідуальної програми тренувань та запобігання завданню шкоди здоров’ю [2].

Також багатьом людям для вмотивованості не вистачає наочної демонстрації свого прогресу. За допомогою періодичного вимірювання фізіологічних покажчиків клієнта можна формувати дані про прогрес клієнта для надання йому мотивації та дисципліни.

Мета даної кваліфікаційної роботи – автоматизація бізнес-процесів фітнес-центру, що спеціалізується на збереженні і покращенні стану здоров’я клієнтів, за допомогою створення розподіленої інформаційної системи підтримки діяльності фітнес-центру. Створення подібної системи дасть

можливість клієнту фітнес-центру віддалено взаємодіяти з його працівниками, а також:

− клієнт отримає інструменти для організації своїх тренувань, відстеження своїх фізіологічних даних та прогресу;

− працівники фітнес-центру отримають інструменти для полегшення ведення обліку клієнтів, їхніх медичних обстежень та тренувань, а також для організації свого розпорядку.

Для досягнення зазначеної мети необхідно розв’язати наступні задачі:

− виконати аналіз предметної області;

− визначити категорії користувачів і сформулювати їхні вимоги до створюваної системи;

− обрати архітектуру і шаблон проектування для створення розподіленої системи;

− спроектувати базу даних;

− обрати технології та засоби створення розподіленої системи;

− з допомогою обраних засобів створити БД, а також клієнтське програмне забезпечення, яке дасть можливість різним категоріям користувачів ефективно маніпулювати даними предметної області відповідно до їхніх повноважень;

− забезпечити захист системи від несанкціонованого доступу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Визначення користувачів розподіленої системи підтримки діяльності фітнес-центру та їх задач

Фітнес-центр – це заклад, який виконує оздоровчу та соціальну функцію. Він призначений для занять спортом і поєднує у собі різні типи програм тренувань та заходів для оздоровлення клієнта. Одні фітнес-центри концентрують свою увагу на досягнені спортивних цілей, а інші – на збереженні і покращенні стану здоров’я клієнта.

Система, що створюється, розробляється для фітнес-центрів, які мають своєю метою надання послуг з тренувань з орієнтацією на оздоровлення клієнтів. Клієнт може звернутися до фітнес-центру з метою вирішення своїх проблем зі здоров'ям або підтримки свого фізичного стану, враховуючи існуючі проблеми і їх обмеження.

Створювана інформаційна система повинна виконувати наступні задачі:

- облік абонементів клієнтів фітнес-центру;
- призначення клієнту персоналізованої програми тренувань з урахуванням його фізіологічних особливостей;
- зберігання історії фізіологічного стану клієнта;
- зберігання даних медичних обстежень клієнта;
- облік працівників фітнес-центру та керування розкладом їх роботи.

Виділяються наступні категорії користувачів:

- клієнт – відвідувач фітнес-центру, якому надаються послуги з тренувань та медичних обстежень;
- тренер – працівник фітнес-центру, який проводить тренування клієнтів та надає інформаційні послуги з тренувань;
- медпрацівник – працівник фітнес-центру, який контролює фізіологічний стан клієнта, вимірюючи його фізіологічні покажчики та проводячи обстеження;
- менеджер – працівник фітнес-центру, який керує обліковими записами клієнтів та співпрацівників, абонементами та подіями у фітнес-центрі.

Список задач користувачів проектованої системи представлений у табл. 1.1.

Таблиця 1.1 – Список задач користувачів розподіленої системи підтримки діяльності фітнес-центру

Номер	Задача	Вхідні дані	Вихідні дані
Клієнт			
К1	Переглянути свій профіль	Ідентифікатор клієнта	Ім'я, прізвище, стать, електронна пошта, телефон, дата народження, дата початку членства у фітнес-клубі, загальна інформація про себе, ім'я тренера, тип абонемента
К2	Переглянути профіль свого тренера	Ідентифікатор тренера	Ім'я, прізвище, стать, електронна пошта, телефон, дата народження, стаж, кваліфікація, дата прийому на роботу, загальна інформація про себе
К3	Переглянути програму тренувань, що призначена тренером	Ідентифікатор клієнта	Кількість повторів, кількість підходів, час (хв.), вага (кг), рекомендація, назва програми та опис
К4	Переглянути покажчики своїх фізіологічних даних	Ідентифікатор клієнта	Вага, зріст, процент води в організмі, процент жиру в організмі
К5	Переглянути календар своїх задач	Ідентифікатор клієнта	Назва та час запланованих задач
Тренер			
Т1	Переглянути свій профіль	Ідентифікатор тренера	Ім'я, прізвище, стать, електронна пошта, телефон, дата народження, стаж, кваліфікація, дата прийому на роботу, загальна інформація про себе
Т2	Переглянути своїх клієнтів	Ідентифікатор тренера	Ім'я, прізвище, стать, електронна пошта, телефон, дата народження, дата початку членства у фітнес-клубі, загальна інформація про себе, тип абонемента
Т3	Переглянути профіль клієнта	Ідентифікатор клієнта	ПІБ, дата народження, вік, про себе, тип абонемента, дата початку членства у фітнес-клубі, ПІБ тренера
Т4	Переглянути покажчики фізіологічних даних клієнта	Ідентифікатор клієнта, дата	Вага, зріст, процент води в організмі, процент жиру в організмі

Продовження таблиці 1.1

Номер	Задача	Вхідні дані	Вихідні дані
Т5	Переглянути календар своїх задач	Ідентифікатор тренера	Назва та час запланованих задач
Т6	Призначити персональну програму тренувань клієнту	Ідентифікатор клієнта, назви фізичних прав, опис, кількість повторів і підходів, вага, час, порядок виконання	Повтори, підходи, час (хв), вага (кг), рекомендація, назва програми та опис
Т7	Переглянути медичні обстеження клієнта	Ідентифікатор клієнта	Дата обстеження, думка, діагноз, рекомендація
Т8	Переглянути список запланованих персональних тренувань на певну дату	Ідентифікатор тренера, дата	Заголовки персональних тренувань і відповідні їм ПІБ клієнтів
Т9	Переглянути заплановане персональне тренування	Ідентифікатор персонального тренування	Назва, опис, час початку, час закінчення, фактичний час початку, фактичний час закінчення, ПІБ клієнта
Т10	Розпочати персональне тренування	Ідентифікатор персонального тренування, час початку	Оновлене персональне тренування з зазначеним часом його початку
Т11	Закінчити персональне тренування	Ідентифікатор персонального тренування, час закінчення	Оновлене персональне тренування з зазначеним часом його кінця
Медпрацівник			
МП1	Переглянути покажчики фізіологічних даних клієнта	Ідентифікатор клієнта	Вага, зріст, процент води в організмі, процент жиру в організмі
МП2	Переглянути профіль клієнта	Ідентифікатор клієнта	ПІБ, дата народження, вік, загальні дані про клієнта які саме, ПІБ тренера
МП3	Переглянути календар своїх задач	Ідентифікатор медпрацівника	Назва та час запланованих задач за певну дату
МП4	Переглянути список запланованих медичних обстежень на певну дату	Ідентифікатор медпрацівника, дата	Заголовки обстежень і відповідні їм імена клієнтів
МП5	Переглянути заплановане медичне обстеження	Ідентифікатор обстеження	Назва, опис, час початку, час закінчення, фактичний час початку, фактичний час закінчення, ПІБ клієнта
МП6	Розпочати медичне обстеження	Ідентифікатор медичного обстеження, час початку	Оновлене медичне обстеження з зазначеним часом його початку
МП7	Закінчити медичне обстеження	Ідентифікатор медичного обстеження, час закінчення	Оновлене медичне обстеження з зазначеним часом його кінця

Продовження таблиці 1.1

Номер	Задача	Вхідні дані	Вихідні дані
МП8	Вносити фізіологічні дані клієнта	Ідентифікатор прийома, вага, зріст, процент води в організмі, процент жиру в організмі	Оновлені дані про фізіологічний стан клієнта: вага, зріст, процент води в організмі, процент жиру в організмі
МП9	Вносити дані медичного обстеження	Ідентифікатор прийома, дата обстеження, підсумки лікаря, діагноз і рекомендації	Нове медичне обстеження
Менеджер			
М1	Переглянути список клієнтів	Критерій пошуку: ПІБ, дата народження	ПІБ клієнтів
М2	Переглянути список тренерів	Критерій пошуку: ПІБ, дата народження,	ПІБ тренерів
М3	Переглянути список медпрацівників	Критерій пошуку: ПІБ, дата народження	ПІБ медпрацівників
Менеджер			
М4	Переглянути профіль клієнта	Ідентифікатор клієнта	ПІБ, дата народження, вік, загальна інформація про себе, тип абонемента, дата початку членства у фітнес-клубі, ПІБ тренера
М5	Переглянути профіль тренера	Ідентифікатор тренера	ПІБ, дата народження, вік, загальні дані про тренера, дата початку роботи, ПІБ клієнтів
М6	Переглянути профіль медпрацівника	Ідентифікатор медпрацівника	ПІБ, дата початку роботи
М7	Реєструвати нових клієнтів	Адреса електронної пошти, ПІБ, вік, номер телефону	Обліковий запис клієнта з присвоєним ідентифікатором, тимчасовий пароль
М8	Призначити абонемент клієнту	Різновид абонементу, ідентифікатор клієнта	Новий абонемент
М9	Зупинити абонемент клієнта	Ідентифікатор абонемента, статус «Відмінений»	Абонемент зі статусом «Відмінений»
М10	Призначити клієнту тренера	Ідентифікатор клієнта, Ідентифікатор тренера	Оновлений профіль клієнта з призначеним йому тренером
М11	Призначити графік роботи тренера	Ідентифікатор тренера, робочі дні, робочі часи, тривалість тренувань, інтервали між ними	Новий графік роботи тренера
М12	Призначити персональне тренування для клієнта	Ідентифікатор клієнта, ідентифікатор тренера, дата та час тренування	Нове персональне тренування, змінений графік тренера
М13	Відмінити персональне тренування для клієнта	Ідентифікатор персонального тренування, статус «Відмінено»	Персональне тренування зі статусом «Відмінено»
М14	Редагувати робочі часи тренера	Ідентифікатор тренера, робочі дні, часи, тривалість тренувань, інтервали між ними	Оновлений графік роботи тренера

Продовження таблиці 1.1

Номер	Задача	Вхідні дані	Вихідні дані
М15	Призначити графік роботи медпрацівника	Ідентифікатор медпрацівника, робочі дні, часи, тривалість прийомів, інтервали між ними	Новий графік роботи медпрацівника
Менеджер			
М16	Призначити обстеження для клієнта	Ідентифікатор клієнта, ідентифікатор медпрацівника, дата та час обстеження	Нове персональне обстеження
М17	Відмінити обстеження	Ідентифікатор персонального обстеження	Персональне обстеження зі статусом «Відмінено»
М18	Отримати ранжування абонементів по їх кількості	Відсутні	Типи підписок, кількість клієнтів для кожного типу та ранг типів підписок.
М19	Отримати кількість нових клієнтів за попередні місяці	Відсутні	Місяці і відповідна їм кількість клієнтів
М20	Отримати кількість прийомів по днях тижня	Відсутні	Дні тижня і відповідні їм кількості прийомів
М18	Редагувати робочі часи медпрацівника	Ідентифікатор медпрацівника, робочі дні, часи, тривалість прийомів, інтервали між ними	Оновлений графік роботи медпрацівника
М19	Створити групове заняття	Час та дата проведення, назва, фотографія, ідентифікатор тренера, опис	Нове групове заняття
М20	Видалити обліковий запис тренера	Ідентифікатор тренера	Відсутні
М21	Видалити обліковий запис медпрацівника	Ідентифікатор медпрацівника	Відсутні

1.2 Огляд аналогів створюваної системи

Далі визначається, які інструменти вже використовуються для розв’язання задач, наведених у табл. 1.1, та розглянути їх переваги та недоліки.

Gym Master [3] – інформаційна система, що розроблена для управління фітнес-центрами та спортивними клубами. Вона надає функціонал для управління клієнтами та розкладом тренувань. Але інтерфейс Gym Master є застарілим і не забезпечує достатньої зручності для користувачів. Крім того, система не надає функціоналу для стеження за станом здоров’я клієнта.

MindBody [4] є системою, яка надає можливість управління клієнтами і персоналом. Однак, впровадження MindBody у роботу фітнес-центру може бути складним, особливо для невеликих фітнес-центрів з обмеженими ресурсами, через високу вартість програмного продукту. Ця система також не підтримує функціоналу стеження за станом здоров'я клієнта.

Zen Planner [5] – інформаційна система, розроблена спеціально для фітнес-бізнесу. Вона надає засоби для управління клієнтами і розкладом тренувань та засоби комунікації. Інтерфейс Zen Planner є перенавантаженим і незручним. Zen Planner, як і раніше розглянуті системи, не підтримує функціоналу стеження за станом здоров'я клієнта.

На сьогоднішній день в Україні, на відміну від зарубіжжя, інформаційні системи для фітнес-центрів не поширені і не мають місцевих аналогів. Також слід зазначити, що жодна з розглянутих систем-аналогів не надає функцій для стеження за станом здоров'я клієнта і ведення обліку його фізіологічних покажчиків. Ця інформація корисна насамперед для тренера, оскільки фізіологічні покажчики клієнта напряму впливають на вибір для нього тих або інших вправ та навантажень, а це, в свою чергу, допомагає уникнути можливих ускладнень від неправильної тренувальної програми. Тому одними з основних функцій створюваної системи повинні бути збір та зберігання інформації про медичну історію клієнта та результати періодичного вимірювання його фізіологічних покажчиків.

Для обґрунтування доцільності розробки розподіленої системи підтримки діяльності фітнес-центру використані методології Jobs to be done [6] та Матриця аналогів [7]. У табл. 1.2 наведено порівняння системи, що розроблюється, з її конкурентами за допомогою методології Jobs to be done. У цій таблиці описані певні незадоволені бажання, обмеження та каталізатори, пов'язані з взаємодією тренера та клієнта в контексті тренувань і здоров'я. Також вказані варіанти вибору систем, що можуть розв'язати ці проблеми. Матриця аналогів конкурентних програмних продуктів наведена у табл. 1.3. Представлені переваги системи, що розроблюється, перед системами-аналогами за допомогою порівняння функціоналу цих систем.

Таблиця 1.2 – Порівняння системи, що створюється, з її конкурентами за допомогою методології Jobs to be done

Незадоволені бажання	Обмеження	Каталізатори	Варіанти вибору
Постійна взаємодія тренера та клієнта	Тренер та клієнт не мають змоги завжди обмінюватися даними дистанційно у зручному вигляді	Інтернет	GymMaster, Zen Planner, Mindbody, InGym (система, що створюється)
Забезпечення постійного доступу з боку тренера або клієнта до програми тренувань та свого графіку у будь-якому місці в будь-який час	Інтерфейс комп'ютера не підходить через його немобільність	Фреймворки мобільного користувацького інтерфейсу	GymMaster, Mindbody, InGym (система, що створюється)
Стеження за станом здоров'я клієнта	У будь-який час до даних стану здоров'я клієнта повинні мати доступ медпрацівник та тренер клієнта, щоб порівняти поточні фізіологічні покажчики клієнта з попередніми	Інтернет	InGym (система, що створюється)
Електронний облік працівників	Незручність обліку у бумажному вигляді	Конкуренція серед великої кількості фітнес-центрів	GymMaster, Mindbody, Zen Planner InGym (система, що створюється)

Таблиця 1.3 – Матриця аналогів конкурентних програмних продуктів

Параметри порівняння	GymMaster	Mindbody	Zen Planner	InGym (система, що створюється)
Стеження за станом здоров'я клієнта	Ні	Ні	Ні	Так
Наявність мобільного застосунку	Так	Так	Ні	Так
Облік працівників та клієнтів	Так	Так	Так	Так

Висновок до розділу 1

Аналіз існуючих програмних продуктів для підтримки діяльності фітнес-центру визначив ряд недоліків, пов’язаних з їх використанням. До найбільш поширеною проблемою є відсутність функції стеження за станом здоров’я клієнта фітнес-центру для правильної організації його тренувань і зниження небезпеки для здоров’я.

Також не всі системи мають підтримку мобільних застосунків для тренерів і клієнтів, що дозволяють їм обмінюватись інформацією про тренування в дистанційному режимі, що може бути особливо корисним для тих клієнтів, які часто подорожують. А для потенційних клієнтів наявність зручного та функціонального мобільного застосунку може стати вирішальним фактором при виборі фітнес-центру [2].

Доцільність створення розроблюваної інформаційної системи для підтримки діяльності фітнес-центру ґрунтується на можливості вирішення описаних проблем завдяки програмній реалізації функцій, відсутніх у застосунках-аналогах. Зокрема, це стосується ведення обліку фізіологічних покажчиків клієнтів та надання їх, по-перше, клієнту для відстеження свого прогресу, так і, по-друге, тренеру для призначення підходящих вправ, що є важливим кроком у забезпеченні безпечного та результативного тренування.

Крім того, завдяки застосуванню технологій Інтернет для створення програмного продукту досягається його платформна незалежність. Розробка клієнтських застосунків для різних платформ надає можливість клієнтам та працівникам фітнес-центру отримувати доступ до функцій, що надаються інформаційною системою, за допомогою браузера з персональних комп’ютерів та з мобільних пристроїв, які працюють під управлінням мобільних операційних систем.

2 ПРОЕКТУВАННЯ РОЗПОДІЛЕНОЇ СИСТЕМИ ПІДТРИМКИ ДІЯЛЬНОСТІ ФІТНЕС-ЦЕНТРУ

2.1 Архітектура та шаблон проектування

Розподілена інформаційна система підтримки діяльності фітнес-центру розробляється з застосуванням інтернет технологій як веб-застосунок. Вона не потребує великої кількості ресурсів для того, щоб надати користувачеві графічний інтерфейс. Для цього вистачає браузера. У свою чергу доступ до системи через браузер повністю вирішує проблему кросплатформності.

Система використовує трирівневу клієнт-серверну архітектуру, що передбачає розмежування рівнів програмного забезпечення на рівень користувацького інтерфейсу, рівень реалізації прикладних алгоритмів і засобів обробки даних (сервер застосунків), а також рівень серверу баз даних, при цьому кожен з рівнів може функціонувати на різних платформах. Трирівнева архітектура є найбільш поширеною для розробки веб-застосунків і дозволяє підвищити масштабованість, надійність та забезпечити високий рівень безпеки.

Як шаблон проектування обраний MVC, згідно якому клієнт відокремлений від серверної частини. Взаємодія відбувається за допомогою контролера, який приймає запит від клієнта і викликає відповідний метод моделі. Відповідь від моделі контролер передає назад клієнту. У роботі використана варіація шаблону MVC, розроблена компанією Apple (рис. 2.1) [8].

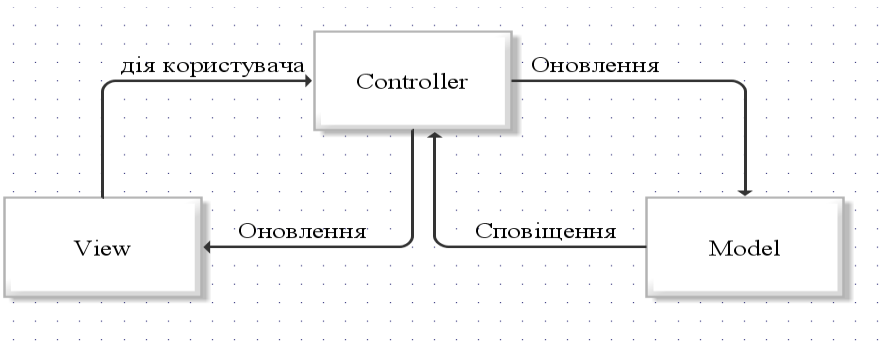


Рисунок 2.1– Шаблон проектування MVC

Нижче перелічені переваги використання даного шаблону.

– Розділення відповідальностей: логіка програми чітко поділяється на три компоненти: модель, представлення і контролер, що сприяє модульності програмного продукту і його легкий підтримці.

– Перевикористання коду: завдяки розділенню на модель, представлення і контролер, об'єкти стають більш перевикористовуваними, що дозволяє використовувати одні й ті ж моделі та види в різних частинах програми або навіть у різних проектах.

– Легкість розширення: завдяки використанню інтерфейсів та взаємодії через контролер, у програму можна додавати нові функціональні можливості без зміни вже існуючих компонентів та ризику порушення роботи інших частин системи.

– Простіше тестування: можна легко написати юніт-тести для окремих компонентів (наприклад, моделей та контролерів), перевіряючи правильність їхньої роботи незалежно від інтерфейсу користувача.

– Сумісність з іншими технологіями: використання шаблону MVC дозволяє легко інтегрувати різні компоненти та використовувати різні бібліотеки, що покращує швидкість розробки та підтримку.

Для взаємодії програмних компонентів рівня користувацького інтерфейсу та сервера застосунків обраний REST [9] – стиль архітектури програмного забезпечення для розподілених систем, таких як World Wide Web, згідно якому управління інформацією ґрунтується на використанні протоколу HTTP. Головна особливість RESTful-застосунків полягає в тому, що сервер не зберігає клієнтського контексту між транзакціями, і кожна транзакція повинна містити усі дані, необхідні для виконання певного запиту. Це забезпечує надійність та масштабованість RESTful-застосунків.

Взаємодія програмних компонентів рівня користувацького інтерфейсу та рівня сервера застосунків при використанні архітектурного стилю REST показана на рис. 2.2.

Рисунок 2.2 – Взаємодія програмних компонентів рівня користувацького інтерфейсу та рівня сервера застосунків при використанні архітектурного стилю REST

2.2 Інформаційне моделювання предметної області

Опис сутностей розглянутої предметної області «фітнес-центр», їх атрибутів і обмежень подано в табл. 2.1.

Таблиця 2.1 – Опис сутностей предметної області «фітнес-центр»

Ім'я атрибута	Призначення атрибута	Обмеження атрибута
user_credentials (дані для входу у систему)		
id	Ідентифікатор даних для входу	Первинний ключ
username	Ім'я користувача	Не порожнє, максимальна довжина 50 символів
passwords	Пароль користувача	Не порожнє, максимальна довжина 256 символів
role	Роль користувача	Не порожнє, значення з множини(«TRAINER», «PARAMEDIC», «MANAGER», «CLIENT»)
employee (працівники)		
employee_type	Тип працівника	Не порожнє, значення з множини («TRAINER», «PARAMEDIC», «MANAGER»)
id	Ідентифікатор працівника	Первинний ключ
about	Інформація про працівника	Максимальна довжина 256 символів
birthday	Дата народження	Не порожнє, не пізніше поточної дати, більше 18 років
Email	Електронна пошта	Не порожнє, мак максимальна довжина 62 символа, шаблон %_@_%._%

Продовження таблиці 2.1

Ім'я атрибута	Призначення атрибута	Обмеження атрибута
experience	Досвід роботи	Більше або дорівнює нулю
first_name	Ім'я	Не порожнє, максимальна довжина 50 символів
gender	Стать	Не порожнє, значення з множини («MALE», «FEMALE»)
hire_date	Дата прийому на роботу	Не порожнє, не пізніше поточної дати
last_name	Прізвище	Не порожнє, максимальна довжина 50 символів
phone	Телефон	Не порожнє, максимальна довжина 13 символів, шаблон +[0-9]{12}
qualifications	Кваліфікація	Максимальна довжина 256
user_credentials_id	Ідентифікатор користувача	Зовнішній ключ для зв’язку з таблицею user_credentials
client (клієнти)		
id	Унікальний ідентифікатор запису	Первинний ключ
about	Інформація про клієнта	Максимальна довжина 256
birthday	Дата народження клієнта	Не порожнє, не пізніше поточної дати
email	Електронна пошта клієнта	Не порожнє, мак максимальна довжина 62 символа, шаблон %_@_%._%
first_name	Ім'я клієнта	Не порожнє, максимальна довжина 50 символів
gender	Стать клієнта	Не порожнє, значення з множини («MALE», «FEMALE»)
last_name	Прізвище клієнта	Не порожнє, максимальна довжина 50 символів
phone	Номер телефону клієнта	Не порожнє, максимальна довжина 13 символів, шаблон +[0-9]{12}
start_date	Дата початку співпраці з клієнтом	Не порожнє, не пізніше поточної дати
trainer_id	Ідентифікатор тренера	Зовнішній ключ для зв’язку з таблицею employee
user_credentials_id	Ідентифікатор облікових даних користувача	Зовнішній ключ для зв’язку з таблицею user_credentials
working_hours (робочі часи)		
id	Унікальний ідентифікатор	Первинний ключ
employee_id	Ідентифікатор працівника	Зовнішній ключ для зв’язку з таблицею employee
finish_time	Час завершення робочих годин	Не порожнє, не раніше start_time
start_time	Час початку робочих годин	Не порожнє, майбутній час

Продовження таблиці 2.1

Ім'я атрибута	Призначення атрибута	Обмеження атрибута
appointment (прийоми)		
appointment_type	Тип призначення	Не порожнє, значення з множини («PERSONAL_TRAINING», «MEDICAL_EXAMINATION»)
id	Унікальний ідентифікатор	Первинний ключ
actual_finish_time	Фактичний час завершення	Не порожнє, не раніше actual_start_time
actual_start_time	Фактичний час початку	Не порожнє, майбутній час
client_id	Ідентифікатор клієнта	Зовнішній ключ для зв’язку з таблицею client
description	Опис призначення	До 255 символів
employee_id	Ідентифікатор працівника	Зовнішній ключ для зв’язку з таблицею employee
finish_time	Час завершення призначення	Не порожнє, пізніше start_time
name	Назва призначення	До 255 символів
start_time	Час початку призначення	Не порожнє, майбутній час
fat_percentage	Відсоток жиру	Не порожнє, більше нуля
height	Зріст	Не порожнє, більше нуля
water_percentage	Відсоток води	Не порожнє, більше нуля
weight	Вага	Не порожнє, більше нуля
medical_examination (медичні огляди)		
id	Унікальний ідентифікатор	Первинний ключ
appointment_id	Ідентифікатор призначення	Зовнішній ключ для звязку з таблицею appointment
date_of_examination	Дата медичного обстеження	Поточний або минулий час
diagnosis	Діагноз	До 256 символів
opinion	Висновок	До 256 символів
recommendation	Рекомендації	До 256 символів
exercise (вправи)		
id	Унікальний ідентифікатор	Первинний ключ
description	Опис вправи	До 256 символів
exercise_type	Тип вправи	Не порожнє, значення з множини («STRENGTH», «CARDIO», «FLEXIBILITY»)
name	Назва вправи	До 255 символів
routine_exercise (вправи у програмі тренувань)		
id	Унікальний ідентифікатор	Первинний ключ
description	Опис вправи	До 256 символів
repetitions	Кількість повторень	Більше або дорівнює нулю
sets	Кількість підходів	Більше або дорівнює нулю
weightKg	Вага знаряддя	Більше або дорівнює нулю
timeMin	Час виконання вправи	Більше або дорівнює нулю
exercise_id	Ідентифікатор вправи	Зовнішній ключ для зв’язку з таблицею exercise

Продовження таблиці 2.1

Ім'я атрибута	Призначення атрибута	Обмеження атрибута
routine_id	Ідентифікатор рутини	Зовнішній ключ для зв’язку з таблицею routine
routine (програми тренувань)		
id	Унікальний ідентифікатор	Первинний ключ
description	Опис програми тренувань	До 256 символів
finish_date	Дата завершення програми тренувань	Не раніше start_date
name	Назва програми тренувань	До 255 символів
start_date	Дата початку програми тренувань	Поточна або майбутня дата
client_id	Ідентифікатор клієнта	Зовнішній ключ для зв’язку з таблицею client
trainer_id	Ідентифікатор тренера	Зовнішній ключ для зв’язку з таблицею trainer
subscription (абонементи)		
id	Унікальний ідентифікатор	Первинний ключ
subscription_type	Назва абонемента	Не порожнє, значення з множини («LIGHT», « MEDIUM», «MAX»)
training_limit	Кількість пресональних тренувань згідно з абонементом	Не порожнє, більше нуля
client_subscription (абонементи клієнта)		
id	Унікальний ідентифікатор	Первинний ключ
start_date	Дата початку абонемента	Поточна дата або майбутня
finish_date	Дата кінця абонемента	Не раніше start_date
status	Статус абонемента	Не порожнє, значення з множини («ACTIVE», «CANCELED», «EXPIRED»)
training_left	Кількість тренувань, що залишилось	Більше або дорівнює нулю
subscription_id	Ідентифікатор абонемента	Зовнішній ключ для зв’язку з таблицею subscription
client_id	Ідентифікатор клієнта	Зовнішній ключ для зв’язку з таблицею client

Далі розглянуті зв’язки між сутностями БД.

Зв’язок між user_credentials і employee. У облікових даних користувача може бути тільки один працівник. У свою чергу у працівника може бути тільки одні облікові дані. Отже між цими двома таблицями існує зв’язок один-до-одного, безумовний з обох боків. Для формалізації зв’язку у таблицю працівника доданий зовнішній ключ, який посилається на ідентифікатор його облікових даних.

Зв’язок між user_credentials і client. Облікові дані користувача системи можуть належати тільки одному клієнту. У свою чергу у клієнта можуть бути тільки одні облікові дані користувача. Отже між цими двома таблицями існує зв’язок один-до-одного, безумовний з обох боків. Для формалізації зв’язку доданий зовнішній ключ у таблицю клієнта, який посилається на ідентифікатор облікового запису.

Зв’язок між employee і appointment. У працівника може бути багато прийомів. У свою чергу прийом може проводити лише один працівник. Отже між цими двома таблицями існує зв’язок один-до-багатьох, безумовний з обох боків. Для формалізації цього зв'язку ідентифікатор працівника доданий в таблицю прийомів у вигляді зовнішнього ключа.

Зв’язок між client і appointment. У клієнта може бути багато прийомів. У свою чергу на прийомі може бути тільки один клієнт. Отже між цими двома таблицями існує зв’язок один-до-багатьох, безумовний з обох боків. Для формалізації цього зв'язку ідентифікатор клієнта доданий в таблицю прийомів у вигляді зовнішнього ключа.

Зв’язок між medical_examination і appointment. У прийома може бути багато медичних обстежень. У свою чергу одне медичне обстеження виконується під час одного прийому. Отже між цими двома таблицями існує зв’язок один-до-багатьох, безумовний з обох боків. Для формалізації цього зв'язку ідентифікатор прийома доданий в таблицю медичного обстеження у вигляді зовнішнього ключа.

Зв’язок між exercise і routine. Одна вправа може входити в багато програм тренувань. У свою чергу програму тренувань можуть складати багато вправ. Отже між цими двома таблицями існує зв’язок багато-до-багатьох, безумовний з обох боків. Для формалізації цього зв'язку додана проміжна таблиця routine_exercise, в яку поміщені первинні ключі таблиць exercise і routine.

Зв’язок між employee і working_hours. У працівника може бути багато варіантів розкладу (робочих годин). У свою чергу у робочих годин може бути тільки один працівник. Отже між цими двома таблицями існує зв’язок один-

до-багатьох, безумовний з обох боків. Для формалізації цього зв'язку ідентифікатор employee доданий в таблицю working_hours у вигляді зовнішнього ключа.

Зв’язок між client і subscription. У клієнта може бути багато абонементів різних типів. У свою чергу у типа абонемента може бути теж багато клієнтів, які ним користуються. Отже між цими двома таблицями існує зв’язок багато-до-багатьох, безумовний з обох боків. Для формалізації цього зв'язку додана проміжна таблиця client_subscription, в яку поміщені первинні ключі таблиць client і subscription.

Зв’язок між client і employee. У тренера може бути багато клієнтів. У свою чергу у клієнта може бути лише один тренер. Отже між цими двома таблицями існує зв’язок один-до-багатьох, безумовний з обох боків. Для формалізації цього зв'язку ідентифікатор employee доданий в таблицю client у вигляді зовнішнього ключа.

ER-діаграма для розподіленої системи підтримки діяльності фітнес-центру приведена у додатку В. Для побудови діаграми використаний програмний пакет Navicat.

2.3 Модель клієнтського застосунку

Клієнтський інтерфейс розподіленої системи підтримки діяльності фітнес-центру складається з веб-інтерфейсу для браузера та мобільного інтерфейсу користувача. Веб-інтерфейс призначений для використання працівниками, яким не потребується постійно переміщуватись: це менеджер та медпрацівник. У свою чергу для клієнта та тренера розроблений мобільний застосунок, який може працювати під управлінням операційних систем IOS та Android.

Веб-інтерфейс менеджера включає в себе різні розділи та функціональність для керування клієнтами та співробітниками. Ієрархію веб-сторінок для менеджера зображено на рис. 2.3.

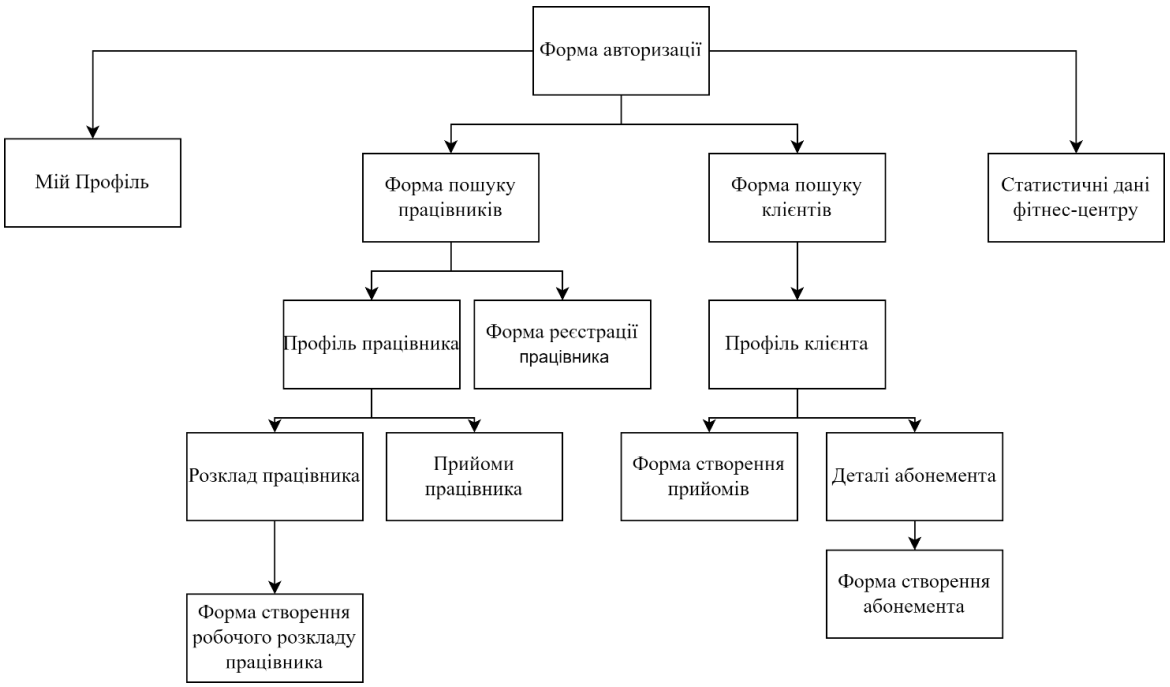


Рисунок 2.3 – Ієрархія форм підсистеми менеджера

Веб-інтерфейс медпрацівника надає зручні засоби для керування профілем медпрацівника, перегляду графіка роботи та виконання медичних обстежень, включаючи заповнення форми перевірки фізіологічних покажчиків клієнта. Ієрархію веб-сторінок для медпрацівника зображено на рис. 2.4.

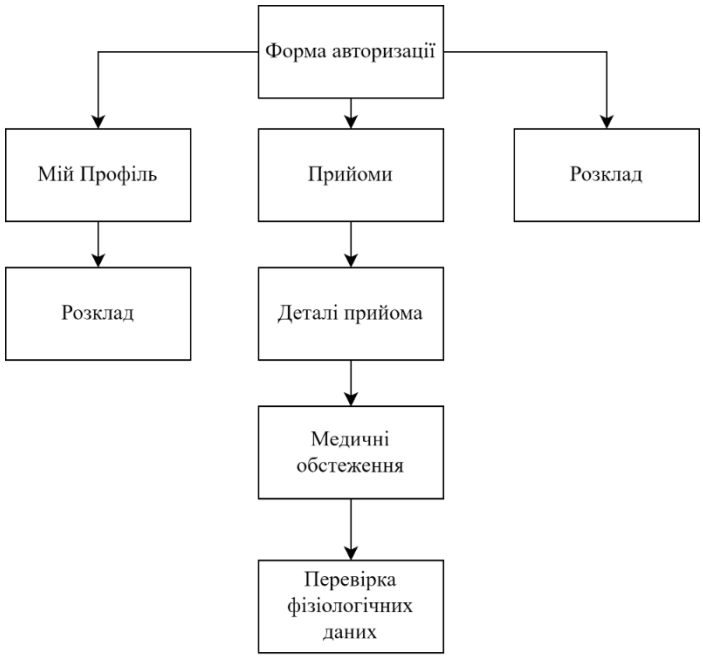


Рисунок 2.4 – Ієрархія форм підсистеми медпрацівника

Клієнтський мобільний застосунок для тренерів надає їм зручний інтерфейс для керування їхнім профілем, перегляду графіка роботи та персональних тренувань клієнтів фітнес-центру. Також застосунок надає доступ до списку клієнтів, які є підопічними тренера, разом з даними їхніх медичних обстежень та історією вимірювання фізіологічних покажчиків, а також до управління програмами тренувань.

Ієрархію веб-сторінок для користувача тренера зображено на рис. 2.5.

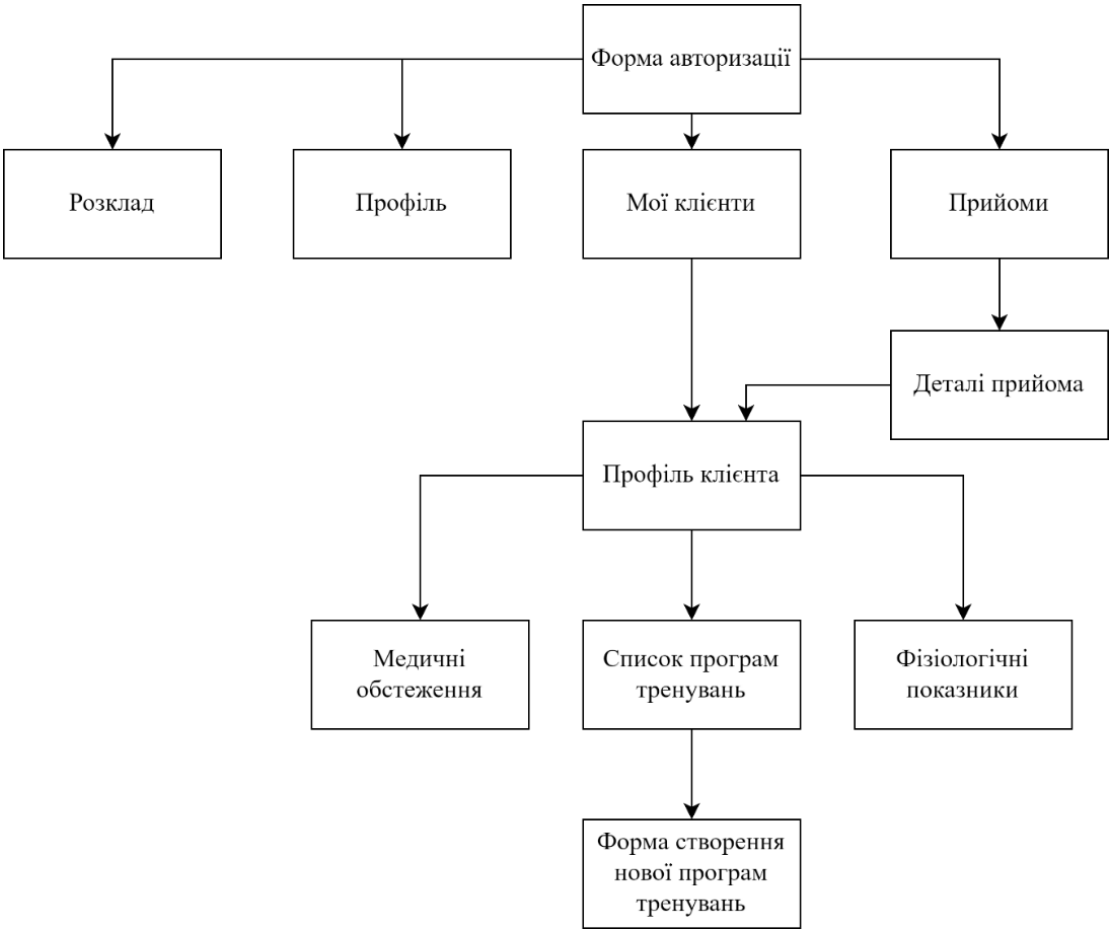


Рисунок 2.5 – Ієрархія форм підсистеми тренера

Клієнтський мобільний застосунок для клієнтів фітнес-центру надає їм зручний інтерфейс для перегляду даних про свої заплановані тренування та медичні обстеження. Крім цього, клієнт має змогу переглядати детальну інформацію про його персональні програми тренувань разом з описом вправ з зазначенням кількості підходів і повторів та рекомендаціями тренера.

Застосунок також надає клієнту доступ до історії його медичних обстежень та результатів періодичних вимірювань фізіологічних покажчиків.

Ієрархію веб-сторінок для клієнта фітнес-центру зображено на рис. 2.6.

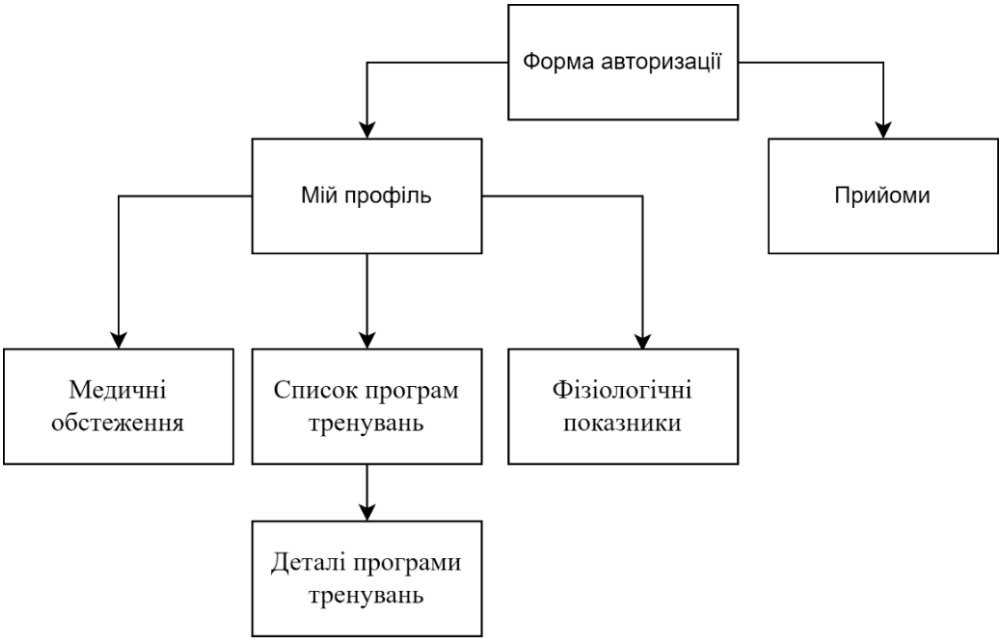


Рисунок 2.6 – Ієрархія форм підсистеми клієнта

3 РЕАЛІЗАЦІЯ РОЗПОДІЛЕНОЇ СИСТЕМИ ПІДТРИМКИ ДІЯЛЬНОСТІ ФІТНЕС-ЦЕНТРУ

3.1 Засоби реалізації розподіленої системи підтримки діяльності фітнес-центру

Java [10] – мова високого рівня з підтримкою віртуальної машини, яка служить рівнем абстракції між кодом та апаратурою обчислювальної системи, забезпечуючи виконання скомпільованого байт-коду на різноманітних платформах. Java підтримує автоматичне управління пам'яттю та багатопоточність, що дозволяє максимально ефективно використовувати процесорний час.

Spring Framework [11] – один з найпопулярніших фреймворків для створення веб-програм на Java, основною перевагою якого є можливість розробки програми як набору слабопов'язаних компонентів: чим менше компоненти програми знають один про одного, тим простіше розробляти новий та підтримувати існуючий функціонал програми.

JavaScript [12] – одна з найпопулярніших мов програмування для розробки веб-застосунків. JavaScript підтримує кросплатформність та має широкий спектр використання, включаючи реалізацію фронтенду і бекенду, створення веб-застосунків, розробку мобільних та гібридних застосунків. Вона також є основою для багатьох популярних фреймворків та бібліотек як, наприклад, React.

PostgreSQL [13] – широко розповсюджена система управління базами даних з відкритим вихідним кодом, що вільно поширюється та має велику спільноту користувачів та розробників. Вона підтримує широкий набір різноманітних типів даних, також надаючи можливість користувачам створювати власні типи даних. PLpg/SQL (процедурне розширення мови SQL, що використовується у PostgreSQL) дозволяє створювати тригери, процедури та функції, які можуть повертати значення. PostgreSQL може обробляти досить

значні обсяги даних та індексувати їх для забезпечення більш швидкого доступу до них.

Intellij Idea Community Edition [14] – це середовище розробки програм на Java, яке поширюється безкоштовно і має наступні переваги:

– підсвічування ключових слів та помилок, автопідстановка;

– перехід до оголошення, пошук використань, пошук за текстовим рядком, файлом чи назвою;

– відображення ієрархії класів та викликів, властивостей та дій класу;

– перейменування класів, якостей та процесів;

– можливість ставити точки переривання.

React [15] – JavaScript-бібліотека для розробки веб-інтерфейсів, яка дозволяє створювати ефективні та масштабовані застосунки. Вона базується на компонентній моделі і використовує віртуальну об’єктну модель компонентів для швидкого оновлення інтерфейсу.

React Native [16] – фреймворк, який базується на React, і дозволяє розробляти мобільні застосунки для платформ Android та iOS з використанням мови JavaScript. Він використовує компонентну модель, яка дозволяє перевикористовувати код і компоненти, що дозволяє економити час та зусилля при розробці мобільних застосунків для різних платформ.

Apache Tomcat [17] – модульний сервер, що спрощує розробку та розгортання веб-застосунків і може працювати практично на всіх комп'ютерних платформах. Він є контейнером сервлетів, які розширюють функціональність веб-сервера і дозволяють виконувати застосунки, що написані з використанням мови Java. Tomcat має просте налаштування та конфігурацію, що робить його легким у використанні. Він підтримує стандартні структури проектів Java, такі як WAR (Web Archive) файл. Spring Framework пропонує зручний інтерфейс для налаштування сервера через конфігураційний файл. Також Tomcat має велику активну спільноту розробників, яка надає підтримку, документацію та рішення для різних проблем.

3.2 Створення бази даних

SQL-запити на створення таблиць бази даних містяться у додатку А.

Далі детально розглянуто створення таблиць subscription і client_subscription. Для визначення декількох полів спочатку знадобиться створити користувацькі типи даних (лістинг 3.1):

```
create type subscription_status_type as enum (
     'active',
     'canceled',
     'expired'
     );

create type subscription_status_type as enum (
     'active',
     'canceled',
     'expired'
     );
```

Лістинг 3.1 – Створення користувацьких типів даних

У коді наведені SQL-запити, які створюють типи subscription_status_type і subscription_type_type за допомогою команди create type. Обидва типи використовують перерахування (enum), що обмежує значення, які можуть бути використані для відповідних полів.

Для створення таблиці subscription потрібно виконати наступний запит:

```
create table subscription (
  id bigserial not null,
  subscription_type subscription_type_type not null,
  training_limit integer check(training_limit>0),
  primary key (id)
);
```

Поле id є унікальним ідентифікатором, значення якого автоматично збільшується за допомогою зазначення типу bigserial, і виступає в ролі первинного ключа. Поле subscription_type має тип subscription_type_type і є обов'язковим (not null). Поле training_limit є цілочисельним полем і має

обмеження (check(training_limit > 0)), що вимагає, щоб значення поля було більше нуля.

Для створення таблиці client_subscription потрібно виконати наступний запит:

```
create table client_subscription (
  id bigserial not null,
  client_id bigint  not null references client(id),
  finish_date date check (finish_date > start_date) not null,
  start_date date check (start_date >= current_timestamp) not
null,
  status subscription_status_type not null,
  subscription_id bigint  not null references subscription(id),
  training_left integer  not null,
  primary key (id)
);
```

Поле id є унікальним ідентифікатором, значення якого автоматично збільшується за допомогою зазначення типу bigserial, і виступає як первинний ключ. Поля finish_date і start_date мають тип date і обмеження (check(finish_date>start_date) і check(start_date>= current_timestamp)) відповідно. Ці обмеження вимагають, щоб значення поля finish_date було більше значення поля start_date, а значення поля start_date було не меншим за поточну дату та час. Поле status має тип subscription_status_type і є обов'язковим (not null). Поле training_left є цілочисельним полем і обов'язковим (not null).

Таблиця client_subscription має два зовнішні ключі. Поле subscription_id має тип bigint і використовує зовнішній ключ для посилання на поле id таблиці subscription. references subscription(id) означає, що атрибут subscription_id залежить від первинного ключа id таблиці subscription. Поле client_id має тип bigint і використовує зовнішній ключ для посилання на поле id таблиці client. references client(id) означає, що атрибут client_id залежить від первинного ключа id таблиці client.

3.3 Розв’язання задач користувачів на рівні бази даних

Для розв’язання задач К1, Т3, МП2, та М4 необхідно виконати наступний запит:

```
SELECT id, first_name, last_name, email, phone, birthday,
start_date, about, gender, trainer_id, user_credentials_id
FROM client WHERE id = <client_id>;
```

Для розв’язання задачі К2 необхідно виконати наступний запит:

```
SELECT e.id, e.first_name, e.last_name, e.email, e.phone,
e.birthday, e.about, e.gender, e.hire_date, e.experience,
e.qualifications
FROM client c JOIN employee e ON c.trainer_id = e.id
WHERE c.id = <client_id>;
```

Для розв’язання задачі К3 необхідно виконати наступний запит:

```
SELECT r.id, r.name, r.description, r.start_date, r.finish_date
FROM client c JOIN routine r ON c.id = r.client_id
WHERE c.id = <client_id>;
```

Для розв’язання задач К4, Т4 та МП1 необхідно виконати наступний запит:

```
SELECT app.id, app.start_time, app.weight, app.height,
app.fat_percentage, app.water_percentage
FROM client c JOIN appointment app ON c.id = app.client_id
WHERE c.id = <client_id>;
```

Для розв’язання задачі К5 необхідно виконати наступний запит:

```
SELECT appointment_type, id, actual_finish_time,
actual_start_time, appointment_status, client_id, description,
employee_id, finish_time, name, start_time
FROM appointment
WHERE client_id = <client_id>;
```

Для розв’язання задач Т1, М5, М6 необхідно виконати наступний запит:

```
SELECT id, first_name, last_name, email, phone, birthday, about,
gender, hire_date, experience, qualifications
FROM employee
WHERE id = <employee_id>;
```

Для розв’язання задачі Т2 необхідно виконати наступний запит:

```
SELECT c.id, c.first_name, c.last_name, c.email, c.phone,
c.birthday, c.about, c.gender, c.start_date
FROM employee e JOIN client c ON e.id = c.trainer_id
WHERE e.id = <trainer_id>;
```

Для розв’язання задач Т5 і МП5 необхідно викликати запити, що містяться у лістингу 3.2.

```
SELECT id, start_time, finish_time
  FROM working_hours
  WHERE employee_id = <employee_id>;
SELECT id, appointment_type, actual_start_time,
actual_finish_time, appointment_status, client_id, description
  FROM appointment
  WHERE employee_id = <employee_id>;
```

Лістинг 3.2 – Запити для розв’язання задач Т5 і МП5

Для розв’язання задачі Т6 необхідно викликати запити, що містяться у лістингу 3.3.

```
INSERT INTO routine_exercise (routine_id, exercise_id,
recommendation, repetitions, sets, minutes, kilos)
VALUES (<routine_id>, <exercise_id>, '<recommendation>',
<repetitions>, <sets>, <minutes>, <kilos>);
INSERT INTO routine (client_id, trainer_id, name, description,
start_date, finish_date)
VALUES  (<client_id>,  <trainer_id>,  '<name>',  '<description>',
'<start_date>', '<finish_date>');
```

Лістинг 3.3 – Запити для розв’язання задачі Т6

Для розв’язання задачі Т7 необхідно виконати наступний запит:

```
SELECT id, appointment_id, date_of_examination, diagnosis,
opinion, recommendation
FROM medical_examination
WHERE appointment_id IN (
      SELECT id
      FROM appointment
      WHERE client_id = <client_id>
      );
```

Для розв’язання задачі Т8 необхідно виконати наступний запит:

```
SELECT appointment.id, appointment.start_time,
appointment.finish_time, appointment.name, client.first_name,
client.last_name
FROM appointment JOIN client ON appointment.client_id = client.id
WHERE appointment.appointment_type = 'PERSONAL_TRAINING'
     AND DATE(appointment.start_time) = '<date>';
```

Для розв’язання задачі Т9 необхідно виконати наступний запит:

```
SELECT appointment.id, appointment.start_time,
appointment.finish_time, appointment.name, client.first_name,
client.last_name
FROM appointment JOIN client ON appointment.client_id = client.id
WHERE appointment.appointment_type = 'PERSONAL_TRAINING'
     AND appointment.id = <appointment_id>;
```

Для розв’язання задач Т10, МП7 необхідно виконати наступний запит:

```
UPDATE appointment
SET appointment_status = 'In Progress', actual_start_time =
CURRENT_TIMESTAMP
WHERE id = <appointment_id>;
```

Для розв’язання задач Т11, МП6 необхідно виконати наступний запит:

```
UPDATE appointment
SET appointment_status = 'Finished', actual_finish_time =
CURRENT_TIMESTAMP
WHERE id = <appointment_id>; WHERE id = <employee_id>;
```

Для розв’язання задачі МП8 необхідно виконати наступний запит:

```
INSERT INTO body_check (appointment_id, date, fat_percentage,
height, water_percentage, weight)
VALUES (<appointment_id>, <date>, <fat_percentage>, <height>,
<water_percentage>, <weight>);
```

Для розв’язання задачі МП9 необхідно виконати наступний запит:

```
INSERT INTO medical_examination (appointment_id,
date_of_examination, diagnosis, opinion, recommendation)
VALUES (<appointment_id>, <date_of_examination>, <diagnosis>,
<opinion>, <recommendation>);
```

Для розв’язання задачі М1 необхідно виконати наступний запит:

```
SELECT id, first_name, last_name, email, phone FROM client;
```

Для розв’язання задачі М2 необхідно виконати наступний запит:

```
SELECT id, first_name, last_name, email, phone
FROM employee
WHERE employee_type = 'TRAINER';
```

Для розв’язання задачі М3 необхідно виконати наступний запит:

```
SELECT id, first_name, last_name, email, phone FROM employee
WHERE employee_type = 'PARAMEDIC';
```

Для розв’язання задачі М7 необхідно виконати наступний запит:

```
INSERT INTO client (first_name, last_name, email, phone)
VALUES ('<first_name>', '<last_name>', '<email>', '<phone>');
```

Для розв’язання задачі М8 необхідно виконати наступний запит:

```
INSERT INTO client_subscription (client_id, start_date,
finish_date, status, subscription_id, training_left)
VALUES (<client_id>, '<start_date>', '<finish_date>', '<status>',
<subscription_id>, <training_left>);
```

Для розв’язання задачі М9 необхідно виконати наступний запит:

```
UPDATE client_subscription SET status = 'CANCELED'
WHERE client_id = <client_id>;
```

Для розв’язання задачі М10 необхідно виконати наступний запит:

```
UPDATE client SET trainer_id = <trainer_id> WHERE id = <client_id>;
```

Для розв’язання задачі М11, М15 необхідно виконати наступний запит:

```
INSERT INTO working_hours (employee_id, start_time, finish_time)
VALUES (<employee_id>, '<start_time>', '<finish_time>');
```

Для розв’язання задачі М12 необхідно виконати наступний запит:

```
INSERT INTO appointment (appointment_typ e, actual_start_time,
client_id, employee_id, start_time, appointment_status)
VALUES ('PERSONAL_TRAINING', '<actual_start_time>', <client_id>,
<employee_id>, '<start_time>', 'SCHEDULED');
```

Для розв’язання задачі М13 необхідно виконати наступний запит:

```
UPDATE appointment SET appointment_status = 'CANCELED'
WHERE client_id = <client_id> AND employee_id = <employee_id>
AND appointment_type = ' PERSONAL_TRAINING ';
```

Для розв’язання задачі М16 необхідно виконати наступний запит:

```
INSERT INTO appointment (appointment_type, actual_start_time,
client_id, employee_id, start_time, appointment_status)
VALUES ('MEDICAL_OBSERVATION, '<actual_start_time>',
<client_id>, <employee_id>, '<start_time>', 'SCHEDULED');
```

Для розв’язання задачі М17 необхідно виконати наступний запит:

```
UPDATE appointment SET appointment_status = 'CANCELED'
WHERE client_id = <client_id> AND employee_id = <employee_id>
AND appointment_type = ' MEDICAL_OBSERVATION';
```

Для розв’язання задачі М18 необхідно виконати запити, що містяться у лістингу 3.4:

```
WITH subscription_counts AS (
SELECT cs.subscription_id, COUNT(*) AS subscription_count
FROM client_subscription cs
GROUP BY cs.subscription_id
)
SELECT  s.subscription_type,
subscription_counts.subscription_count, RANK() OVER (ORDER BY
subscription_counts.subscription_count DESC) AS
subscription_rank
FROM subscription_counts
JOIN subscription s ON s.id =
subscription_counts.subscription_id;
```

Лістинг 3.4 – Запити для розв’язання задачі М18

Для розв’язання задачі М19 необхідно виконати наступний запит:

```
SELECT DATE_TRUNC('month', start_date)::DATE AS month_start,
COUNT(*) AS clients_count
FROM client
GROUP BY month_start
ORDER BY month_start DESC;
```

Для розв’язання задачі М20 необхідно виконати наступний запит:

```
SELECT TO_CHAR(start_time, 'Day') AS weekday, COUNT(*) AS
business_level
FROM appointment
GROUP BY weekday
ORDER BY MIN(EXTRACT(DOW FROM start_time));
```

Також, враховуючи задачі інформаційної системи М12 та М16 для пошуку вільних часових проміжків працівника для прийому на певну дату, створена функція get_available_timeslots (додаток Б, лістинг Б.1). Приклад виклику цієї функції, що як параметри приймає дату, ідентифікатор співпрацівника та потрібний часовий інтервал:

```
SELECT start_time, finish_time FROM
get_available_timeslots(CAST(:date AS DATE), :employeeId,
:intervalValue)
```

Для перевірки прийомів на існування часової накладки створений тригер appointment_overlap_trigger (додаток Б, лістинг Б.2). Він спрацьовує кожного разу після створення нового прийому і перевіряє чи є у даного співпрацівника часова накладка з новим прийомом.

Для розв'язання задачі М8 створений тригер, що спрацьовує кожного разу перед створенням персонального тренування і має ім'я check_and_update_training_left_trigger (додаток Б, лістинг Б.3). Тригер виконує перевірку існування у клієнта абонемента, а також зменшення кількості тренувань, що залишились, на один і переведення абонементу у статус «Закінчений», якщо кількість доступних тренувань дорівнює нулю.

3.4 Технологія доступу до розподіленої системи підтримки діяльності фітнес-центру

Технологія доступу до розподіленої системи підтримки діяльності фітнес-центру подана на схемі, що зображена на рис 3.1.

Користувацький застосунок розроблений з використанням бібліотеки React, і складається з трьох основних компонентів: Router, Components та Service. Router є рішенням для клієнтської маршрутизації в React. При зміні URL Router відображає відповідні компоненти без необхідності виконувати запит до сервера. Components – це набір компонентів, які використовуються для побудови користувацького інтерфейсу. При переході на існуючий URL, Router відображає відповідний компонент. Для того, щоб отримати дані для відображення, компоненти використовують Service. Service містять функції, що виконують запити до сервера і отримують відповідні дані, які потім використовуються компонентами користувацького застосунку.

Сервер застосунків, що працює під управлінням Apache Tomcat та Spring Framework, складається з наступних шарів: REST Controller, Service,

Repository. Шар REST Controller приймає запити від клієнта на певних URL. REST Controller може приймати запити, які також містять додаткову інформацію як, наприклад, тіло запиту. Зазвичай, таке тіло конвертується Spring в DTO об’єкт. DTO об’єкти використовуються виключно для передачі інформації клієнту. Але для того, щоб взаємодіяти з базою даних потрібно конвертувати DTO в Entity об’єкт. Entity об’єкти зберігають не тільки звичайну, а й технічну інформацію, яка пов’язана з тим, як ця сутність зберігається у базі даних. Тому з причин безпеки для обміну даними між сервером і клієнтом використовуються DTO об’єкти. За допомогою шару Repository, відбуваються запити до бази даних. Цей шар завжди приймає або повертає Entity об’єкти.

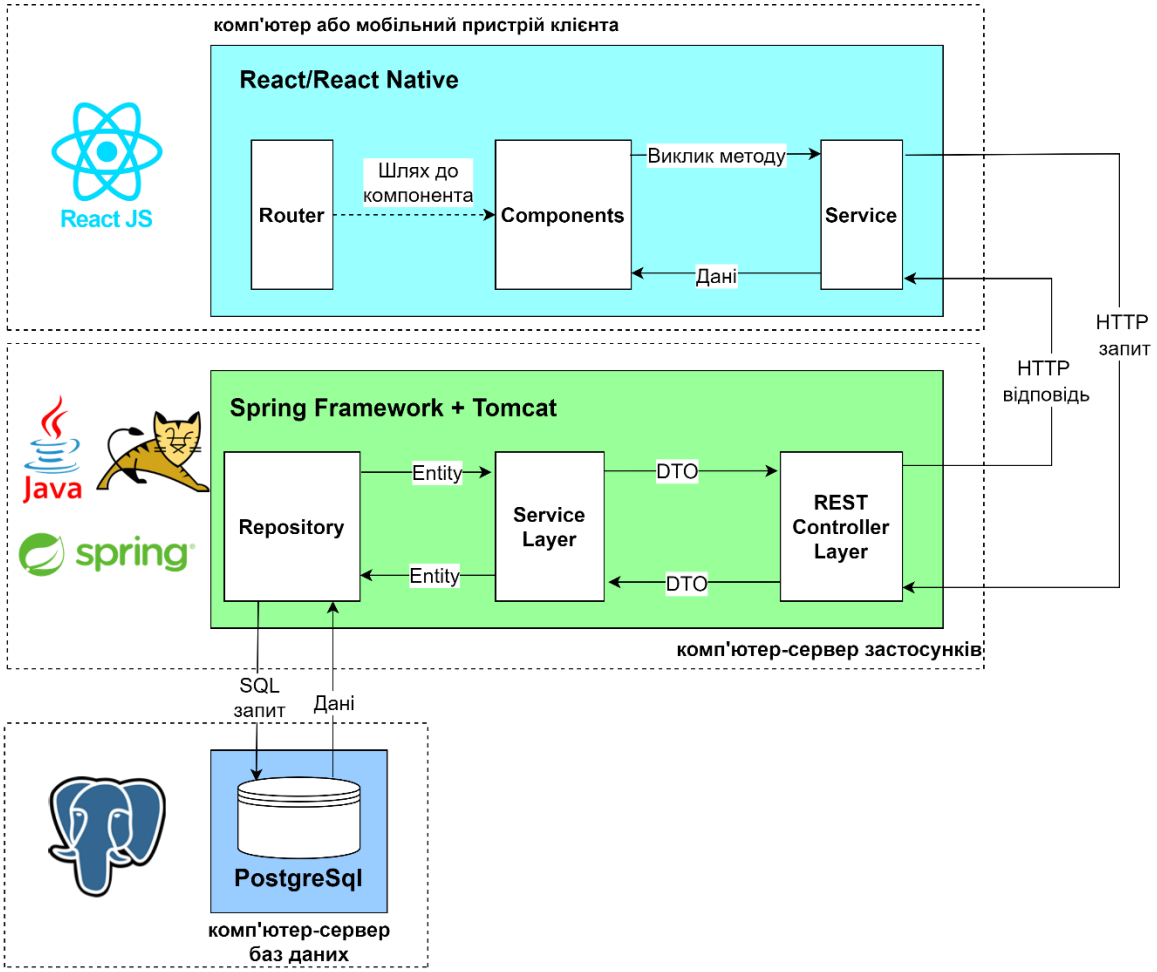


Рисунок 3.1 – Технологія доступу до розподіленої системи підтримки діяльності фітнес-центру

3.5 Програмна реалізація розподіленої системи підтримки діяльності фітнес-центру

Програмна модель створеної розподіленої системи підтримки діяльності фітнес-центру описана узагальненою UML-діаграмою (рис. 3.2), яка має наступні елементи:

– клас Entity, який репрезентує сутність бази даних;

– клас DTO, який використовується для передачі лише необхідних для клієнта даних, ігноруючи технічні дані, потрібні для збереження в базі даних;

– контролер, який приймає запити від клієнта;

– сервісний клас, що надає методи, які реалізують бізнес-логіку системи;

– клас-репозиторій, що відповідає за логіку запитів до бази даних.

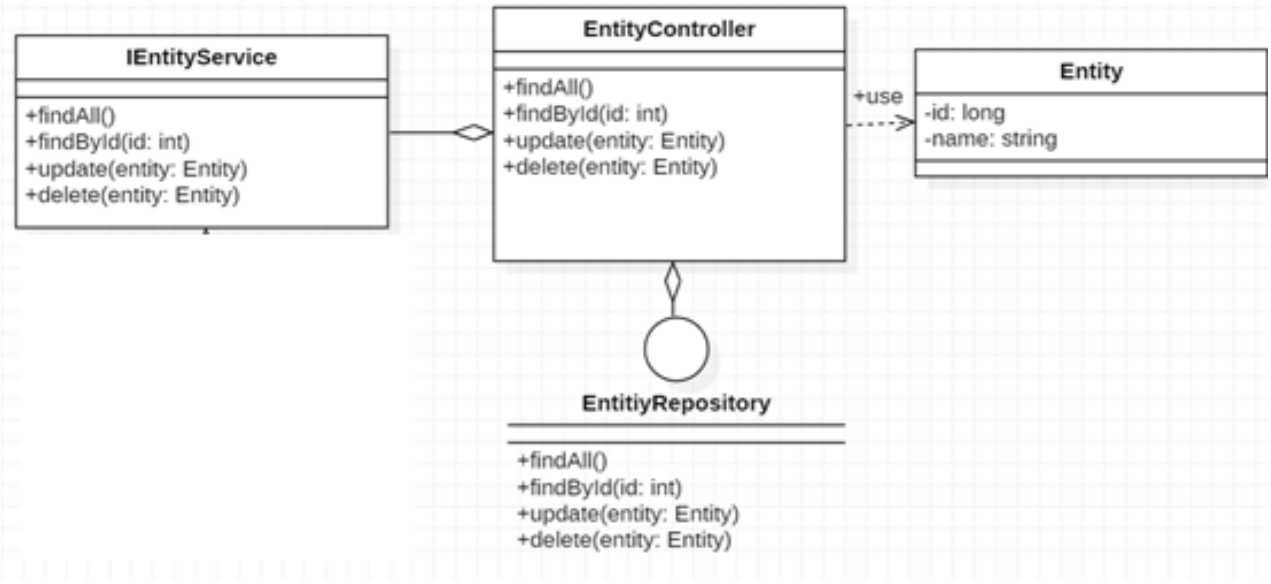


Рисунок 3.2 – Узагальнена діаграма класів розподіленої системи підтримки діяльності фітнес-центру

При реалізації розподіленої системи підтримки діяльності фітнес-центру створено 116 класів. Типова структура компонентів програмного забезпечення на прикладі взаємодії класів Controller, Service, Repository та Mapper (відображає Enitity на DTO і навпаки) наведена у додатку Г.

Клас Employee (рис 3.3) представляє працівника і містить усі необхідні поля, які відповідають цій сутності. Клас Employee являє собою Entity, тобто він призначений для того, щоб відповідати за відповідну таблицю у базі даних і містить інформацію для її коректного збереження, наприклад, опис зв’язків між сутностями. Як і багато інших Entity-класів, Employee має батьківський клас BaseEntity, що відповідає за опис стратегії генерації унікальних ідентифікаторів.

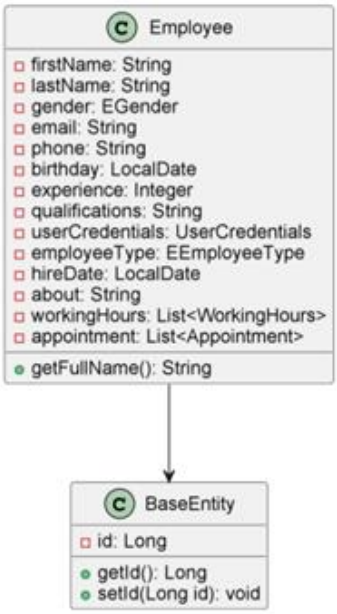


Рисунок 3.3 – Діаграма класу Employee

Далі розглядаються класи EmployeeController та AppointmentService (рис. 3.4). EmployeeController відповідає за отримання та обробку запитів, що надходять від клієнта. У собі він містить багато методів, які виконуються, якщо надходить запит на відповідну адресу. Саме цей контролер відповідає за функції, безпосередньо пов’язані з задачами працівників. Основні методи контролера:

– getProfile: повертає дані профілю автентифікованого працівника;

– getProfileById: повертає інформацію профілю працівника за його ідентифікатором;

– search: здійснює пошук працівників за заданим рядком пошуку, повертає список знайдених працівників;

– createWorkingHours: створює робочі години для вказаного працівника, повертає список створених робочих годин;

– getLastDayOfWork: для вказаного працівника повертає останній день роботи;

– getAvailableTimeslots: повертає доступні години для запису до вказаного працівника на задану дату.

Рисунок 3.4 – Діаграма класів EmployeeController та AppointmentService

EmployeeController містить в собі різні сервісні класи, що відповідають за бізнес-логіку системи. Кожен бізнес клас приймає і повертає DTO об’єкти, які призначені лише для передачі потрібної інформації. Прикладом такого класу є AppointmentService. Він призначений для керування прийомами та містить наступні методи:

– createAppointment: створює нову зустріч на основі переданого об'єкта AppointmentDTO;

– cancelAppointment: скасовує задану зустріч;

– getAppointmentsByEmployeeId: повертає інформацію про зустрічі для вказаного ідентифікатора працівника на задану дату;

– getAppointmentById: повертає дані про зустріч за її ідентифікатором;

– updateActualStartTime: оновлює фактичний час початку зустрічі для заданої зустрічі;

– updateActualFinishTime: оновлює фактичний час завершення зустрічі для заданої зустрічі.

Зазвичай кожен сервіс містить класи для конвертації Entity об’єктів в DTO об’єкти. Також вони містять класи-репозиторії для того, щоб виконувати запити до бази даних. Наприклад, на рис. 3.5 наведена UML-діаграма зв’язків класів AppointmentService з AppointmentMapper та AppointmentRepository.

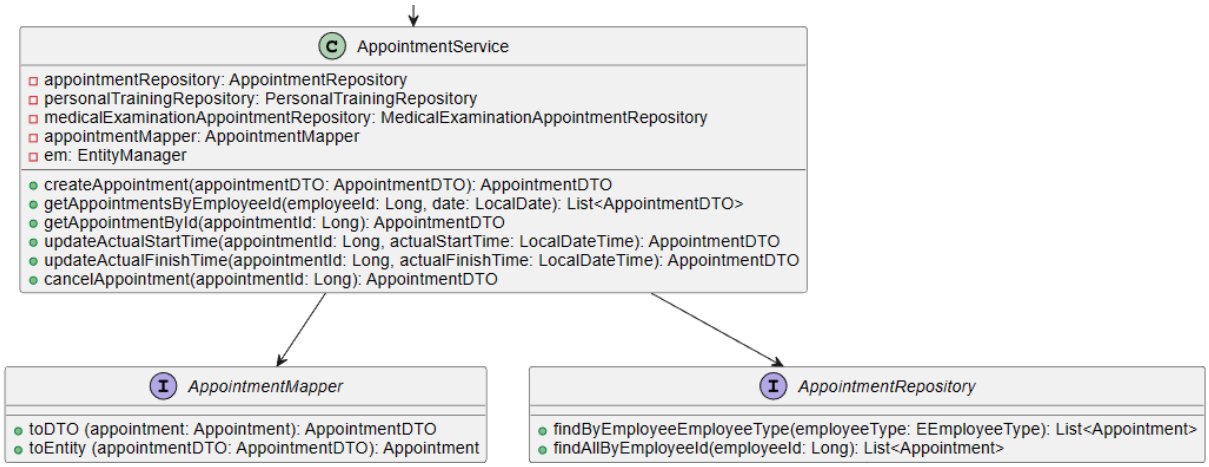


Рисунок 3.5 – UML-діаграма зв’язків класів AppointmentService з AppointmentMapper та AppointmentRepository

Spring Framework дозволяє використовувати так звані специфікації для побудови запитів до бази даних. Специфікації Spring надають можливість створювати динамічні запити до бази даних на основі певних умов або критеріїв. Specification – це інтерфейс, який описує умови, що повинні виконуватись для запиту до бази даних. Він має метод toPredicate, який приймає об'єкт Root, CriteriaQuery та CriteriaBuilder і повертає предикат, що

представляє умови запиту. У лістингу 3.5 наведений приклад специфікації для пошуку прийомів за заданою датою.

```
static Specification<Appointment> byDate(LocalDate date) {
     LocalDateTime startOfDay = date.atStartOfDay();
     LocalDateTime endOfDay = date.plusDays(1).atStartOfDay()
     return (root, query, builder) ->
     builder.between(root.get(Appointment_.startTime),
     startOfDay, endOfDay);
}
```

Лістинг 3.5 – Приклад специфікації для пошуку прийомів за заданою датою

Для зручності специфікації розташовуються у відповідному репозиторії. Наприклад, на рис. 3.6 показана UML-діаграма зв’язку інтерфейсу AppointmentRepository з Specs.

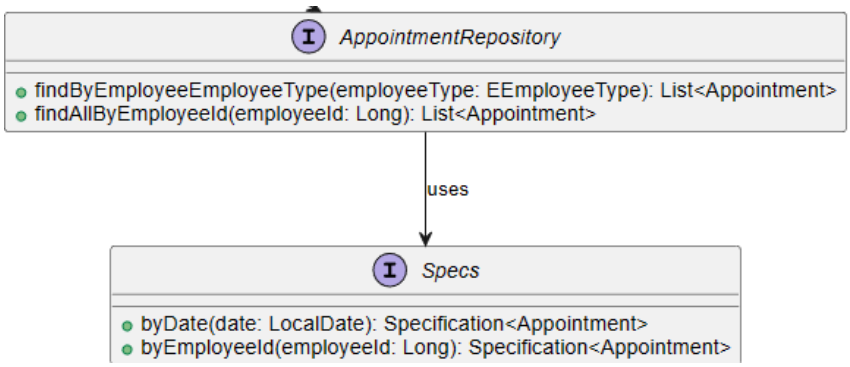


Рисунок 3.6 – UML-діаграма зв’язку інтерфейсу AppointmentRepository з Specs

3.6 Програмна реалізація веб-застосунків для медпрацівника та менеджера

Веб-інтерфейс застосунків для медпрацівника та менеджера розроблені за допомогою бібліотеки React. Оскільки при використанні React застосунок розбивається на незалежні компоненти, які можна повторно використовувати, це дозволило зробити розробку гнучкою та швидкою.

Компоненти можуть містити як візуальні елементи, так і логіку.

Для опису реалізації клієнтської частини застосунку на React застосована діаграма компонентів (рис 3.7), на якій зображена структура інтерфейсів менеджера та медпрацівника. Головний компонент містить два інші компоненти – шаблони інтерфейсу. Вони в свою чергу, містять відповідні їм навігаційні панелі, що дозволяють переходити на потрібну сторінку, яка відображується в шаблоні інтерфейсу в залежності від обраного пункту навігації.

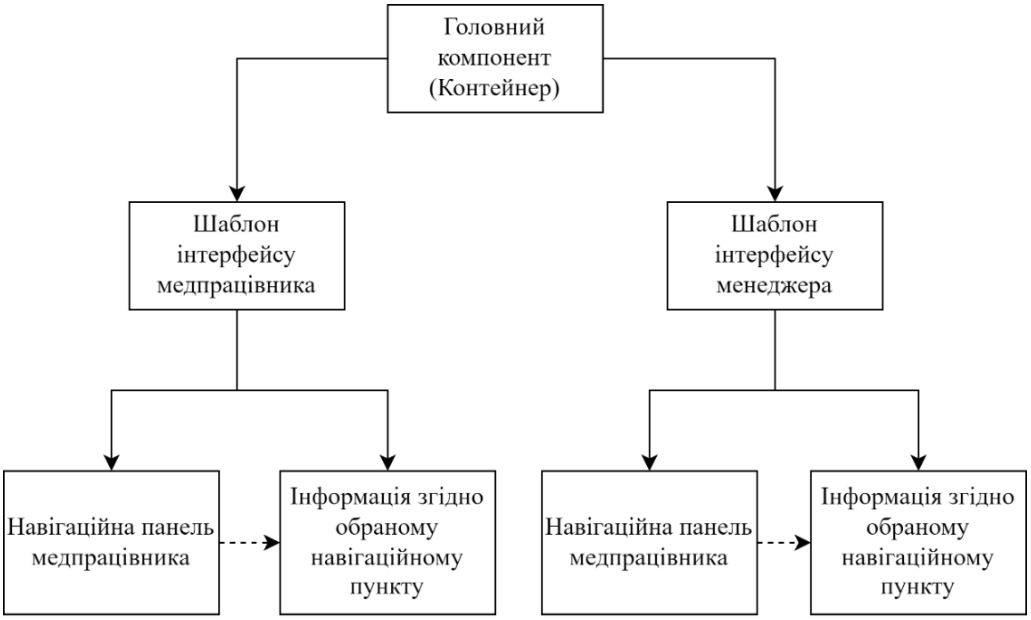


Рисунок 3.7– Структура веб-інтерфейсів менеджера та медпрацівника

У роботі застосована бібліотека React Router, що автоматизує та надає зручні засоби для навігації між компонентами застосунку. Наприклад, у лістингу 3.6 міститься програмний код, що формує структуру шляхів, за якими розташовуються відповідні компоненти інтерфейсу медпрацівника. Кожен компонент Route представляє якийсь шлях, що задається властивістю path, і відображає елемент, який передається у властивість element. Компоненти Route можуть бути вкладені. Тоді шлях вкладеного компонента починається з шляху його батьківського компонента.

```
<Route path="/paramedic" element={<ParamedicLayout />}>
<Route index element={<MyProfile />} />
<Route path="schedule" element={<EmployeeSchedulePage />} />
<Route path="appointments" element={<AppointmentsSearch />} />
     <Route path="appointments/:appointmentId"
element={<FormProvider><AppointmentMultiStepForm
/></FormProvider>} />
</Route>
```

Лістинг 3.6 – Приклад ієрархії URL шляхів у інтерфейсі медпрацівника

Варто зупинитись на компоненті для авторизації працівників веб-застосунку для менеджера та медпрацівника (додаток Д, лістинг Д.1). Він використовується для отримання від користувача інформації про його ім'я та пароль. Після надсилання форми на сервер відбувається перевірка валідності введених даних.

Компонент використовує хук useState для збереження стану змінних username, password, error і fieldError, тобто для збереження стану форми входу користувача. Змінні username та password визначаються за допомогою useState і початковим значенням порожнього рядку. Ці змінні відображають значення полів вводу на формі.

Функція handleLogin виконується при надсиланні форми. Вона перевіряє, чи заповнені обидва поля username та password. Якщо поля не заповнені, встановлюються відповідні помилки у змінну fieldError. Якщо ж поля заповнені, виконується спроба входу, використовуючи сервіс AuthService.login. Якщо вхід успішний, отримується токен доступу, роль та ідентифікатор користувача. Ці дані зберігаються у локальному сховищі (localStorage) і передаються функції setAuth для збереження.

Для ввода тексту розроблений користувацький компонент InputWithError (додаток Д, лістинг Д.2). Він використовується для відображення поля вводу у формі разом з повідомленням про помилку, якщо така виникла. Компонент отримує на вхід різні властивості, такі як value, label, placeholder, type, onChange, id, name, errorMessage, defaultValue і disabled.

Значення value відображає значення поля вводу, label відображає мітку для поля, placeholder встановлює підказку у полі вводу, type визначає тип поля (наприклад, text, password тощо).

Функція onChange викликається при зміні значення поля вводу і передає нове значення. id та name використовуються для ідентифікації поля вводу. defaultValue встановлює початкове значення поля вводу, а disabled вказує, чи поле вводу має бути вимкненим для редагування.

Всередині компонента InputWithError також відображається контейнер форми з полем вводу, який отримує пропси для відображення відповідних значень. Крім того, якщо присутня errorMessage (повідомлення про помилку), то воно відображається як текст помилки.

Цей компонент може бути використаний у інших формах, де потрібно відображати поля вводу разом з повідомленнями про помилки. Він дозволяє зручно керувати відображенням полів вводу та повідомленнями про помилки в одному компоненті.

3.7 Програмна реалізація мобільних застосунків тренера та клієнта

Мобільні застосунки під управлінням React Native використовують інші підходи реалізації навігації аніж настільні додатки через специфіку взаємодії з мобільним пристроєм. Але філософія React працює і у мобільних застосунках.

У роботі застосована бібліотека React Navigation, що автоматизує та надає зручні засоби для навігації між компонентами застосунку. Наприклад, у лістингу 3.7 міститься програмний код, що формує структуру шляхів, за якими розташовуються відповідні компоненти інтерфейсу тренера. У коді використовується компонент Tab.Navigator з React Navigation. Він створює навігаційну панель з вкладками, де кожна вкладка відповідає певному екрану додатку. Наприклад, на вкладці «Profile» відображається компонент PorfileScreen, а на вкладці «Clients» – компонент MyClientsStackScreen. Таким чином, користувач може перемикатись між різними екранами, використовуючи навігаційні вкладки.

```
<Tab.Navigator>
     <Tab.Screen name="Profile" component={PorfileScreen}/>
     <Tab.Screen name="Clients" component={MyClientsStackScreen}
options={{ headerShown: false }}/>
     <Tab.Screen name="Appointments" component=
{AppointmentsStackScreen} options={{ headerShown: false }} />
     <Tab.Screen name="Schedule" component={MyClientsScreen} />
</Tab.Navigator>
```

Лістинг 3.7 – Приклад ієрархії шляхів мобільного користувацького інтерфейсу тренера

Деякі вкладки відображають так звані стек-компоненти як, наприклад, AppointmentsStackScreen. У контексті React Navigation, стек є одним з видів навігації і використовується для управління переходами між екранами додатку в стилі «стеку» або «назад-вперед». У лістингу 3.8 використовується createNativeStackNavigator для створення стеку навігації AppointmentsStack. В цьому стеку визначені екрани, пов'язані зі списком призначень, деталізацією призначення, медичними обстеженнями та перевіркою стану здоров'я. Компонент AppointmentsStackScreen використовується для відображення цього стеку навігації. Користувач може проглядати список призначень та взаємодіяти з відповідними екранами, щоб отримати детальну інформацію.

```
const AppointmentsStack = createNativeStackNavigator();
const AppointmentsStackScreen = () => {
    return ( <AppointmentsStack.Navigator>
        <AppointmentsStack.Screen name="Appointments List"
component={AppointmentsScreen} />
        <AppointmentsStack.Screen name="Appointment Details"
component={AppointmentDetailsScreen} />
        <AppointmentsStack.Screen name="Medical Examinations"
component={MedicalExaminationsScreen} />
        <AppointmentsStack.Screen name="Body Checks"
component={BodyCheckScreen} />
      </AppointmentsStack.Navigator>
      );  };
```

Лістинг 3.8 – Приклад стеку екранів тренера, які відповідають за прийоми

3.8 Безпека інформаційної системи

Для реалізації безпеки розподіленої системи підтримки діяльності фітнес-центру використані JWT токени. Після успішного проходження користувачем процесу автентифікації, сервер генерує JWT токен і повертає його клієнту. Клієнт зберігає токен в локальному сховищі (LocalStorage або Cookies), і використовує його для кожного запиту до сервера. При отриманні запиту сервер перевіряє підпис токена, розшифровує його тіло та перевіряє правильність інформації в ньому, таку як термін дії токена, роль користувача тощо. Якщо токен є дійсним та валідним, сервер дозволяє доступ до запитуваних ресурсів або виконує запитану дію (рис. 3.8).

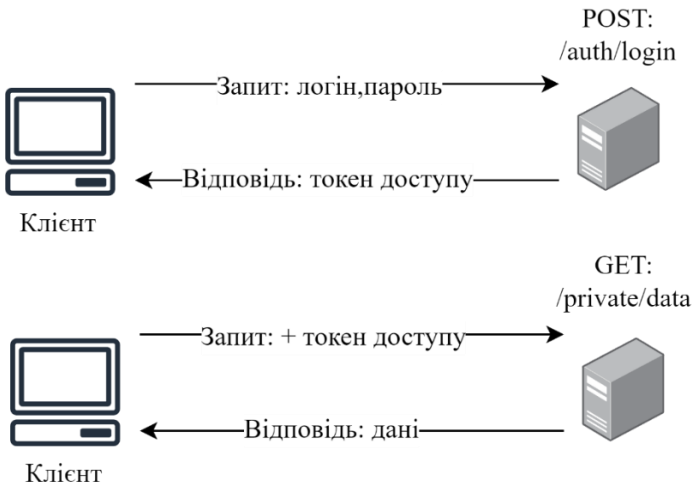


Рисунок 3.8 – Схема процесу роботи інформаційної системи під управлінням фреймворку Spring з використанням JWT

У сервері застосунків за допомогою функціоналу Spring Framework додається фільтр обробки токенів, який оброблює кожен вхідний HTTP-запит і отримує заголовок Authorization цього запиту. Цей заголовок повинен містити JWT токен. Фільтр перевіряє на дійсність поточний токен, виходячи з правильності шифрування і строку придатності токенів. Якщо токен є дійсним, то у контекст безпеки Spring встановлюється відповідна авторизація. В іншому разі, користувач не отримує прав доступу.

Для того щоб отримати JWT токен, клієнтський застосунок повинен надіслати запит з обліковими даними. У разі отримання дійсних облікових даних, генерується JWT токен на їх основі. Приклад генерації JWT токену наведено у лістингу 3.9. Метод setSubject встановлює суб'єкта, тобто ім'я користувача, що авторизувався. Метод setIssuedAt встановлює час створення токену, а метод setExpiration – час закінчення терміну придатності токену. Для того, щоб зашифрувати дані і створити токен, використовується метод signWith. Як аргументи він приймає секретний ключ і алгоритм шифрування. У розробленій системі використовується алгоритм HS256.

```
return Jwts
.builder()
.setSubject(userDetails.getUsername())
.setIssuedAt(new Date(System.currentTimeMillis()))
.setExpiration(new Date(System.currentTimeMillis() + 1000 * 60))
.signWith(getSignInKey(), SignatureAlgorithm.HS256)
.compact();
```

Лістинг 3.9 – Генерація JWT токена на основі облікових даних клієнта

Клієнтський застосунок використовує бібліотеку Axios, за допомогою якої можна надсилати HTTP-запити. Axios підтримує перехват запитів за допомогою так званих перехоплювачів (interceptors), що дозволяють обробляти кожен запит до того, як він буде надісланий. Використовуючи цей функціонал, за наявності токену, він додається у заголовок Authorization (лістинг 3.10).

```
$api.interceptors.request.use(
  async (config) => {
   const accessToken = await AsyncStorage.getItem('accessToken');
   if (accessToken) {
     config.headers['Authorization'] = `Bearer ${accessToken}`;}
       return config; },
   (error) => { return Promise.reject(error); } );
```

Лістинг 3.10 – Перехоплення HTTP запиту і додавання JWT токена до заголовка Authorization

4 ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА

4.1 Опис веб-застосунків менеджера та медпрацівника

При потраплянні на сайт працівнику фітнес-центру необхідно ввести свої облікові дані (рис. 4.1). Якщо введена інформація є недійсною, то виводиться відповідне повідомлення.

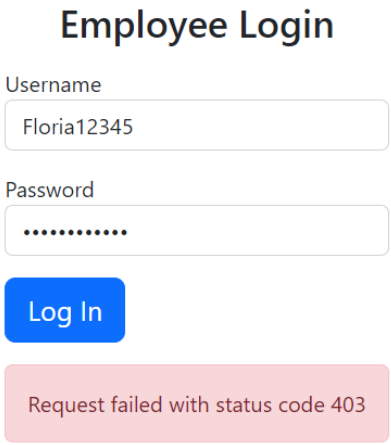


Рисунок 4.1 – Форма авторизації

Після успішної автентифікації та авторизації користувач побачить свій профіль і відповідну навігаційну панель згідно зі своєю роллю.

4.1.1 Опис інтерфейсу менеджера

Якщо користувач має роль менеджера, то він побачить навігаційну панель, яка наведена на рис. 4.2.

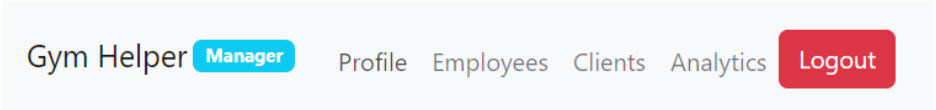


Рисунок 4.2 – Навігаційна панель менеджера

Якщо перейти на вкладку навігаційної панелі Profile, то можна отримати інформацію про поточного авторизованого користувача (рис. 4.3).

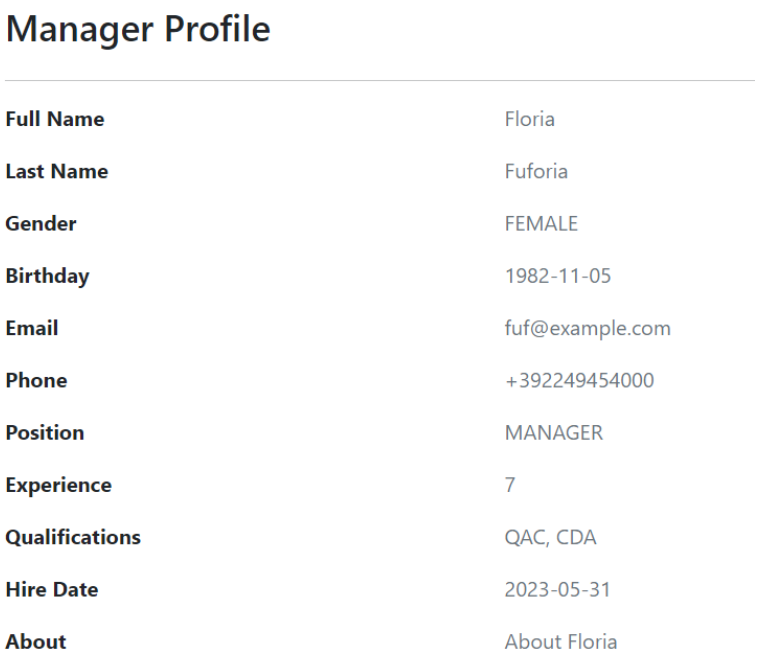


Рисунок 4.3 – Інформація про поточного авторизованого користувача

Вкладка Employees надає функціонал обліку персоналу. Вона представляє собою компонент пошуку з можливістю фільтрації працівників за їх посадою (медпрацівник або тренер) та ім’ям (рис 4.4).

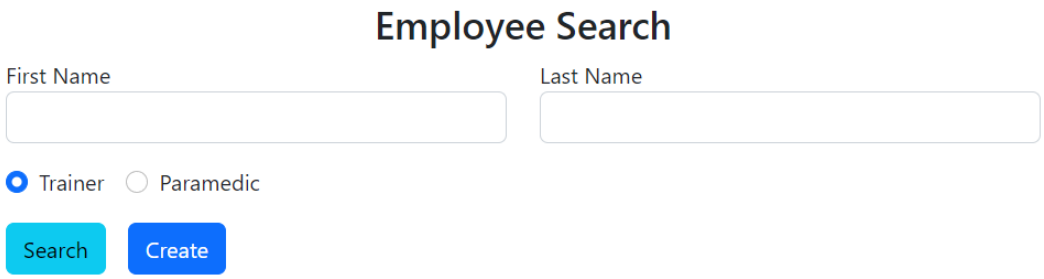


Рисунок 4.4 – Форма пошуку працівників

Після натискання кнопки Search відбувається пошук за заданими параметрами і виводиться список працівників, які підпадають під ці параметри

(рис. 4.5). При натисканні на ім’я працівника, можна перейти на його профіль і отримати доступ до його персональної інформації, розкладу та прийомів.

#	Name	birthday	Phone
1	Ryann Leathart	1982-05-11	+392249454000
2	John Doe	1980-01-01	+392249454000
3	Jane Smith	1985-02-02	+392249454000
4	Michael Johnson	1983-06-15	+392249454000
5	Emily Williams	1987-09-20	+392249454000
6	David Brown	1981-04-05	+392249454000
7	Jessica Jones	1984-07-10	+392249454000
8	Daniel Anderson	1986-03-25	+392249454000
9	Sarah Taylor	1988-08-08	+392249454000
10	Christopher Wilson	1989-12-12	+392249454000

Рисунок 4.5 – Список працівників

Тренери та медпрацівники мають схожі сторінки профілю. Приклад профілю тренера наведений на рис. 4.6.

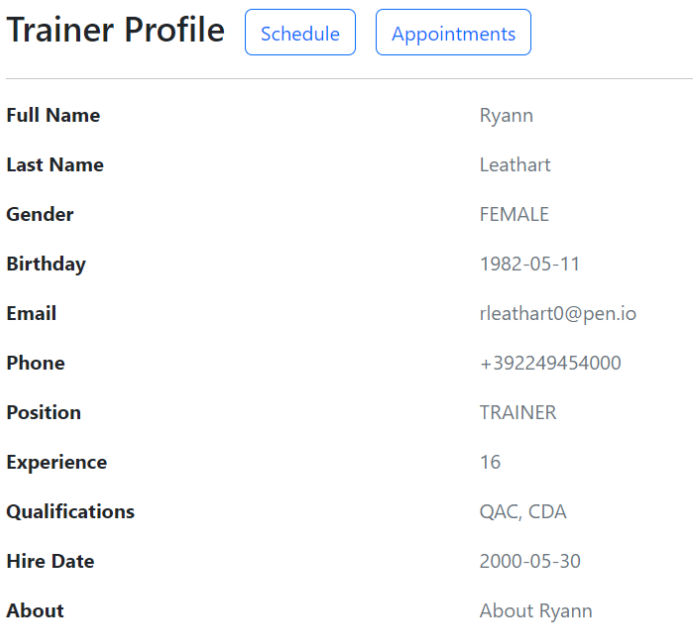


Рисунок 4.6 – Профіль тренера

На сторінці профілю працівників є дві кнопки: Schedule та Appointments. При натисканні кнопки Schedule виводиться календар з розкладом і прийомами (тренуваннями чи обстеженнями) працівника (рис. 4.7).

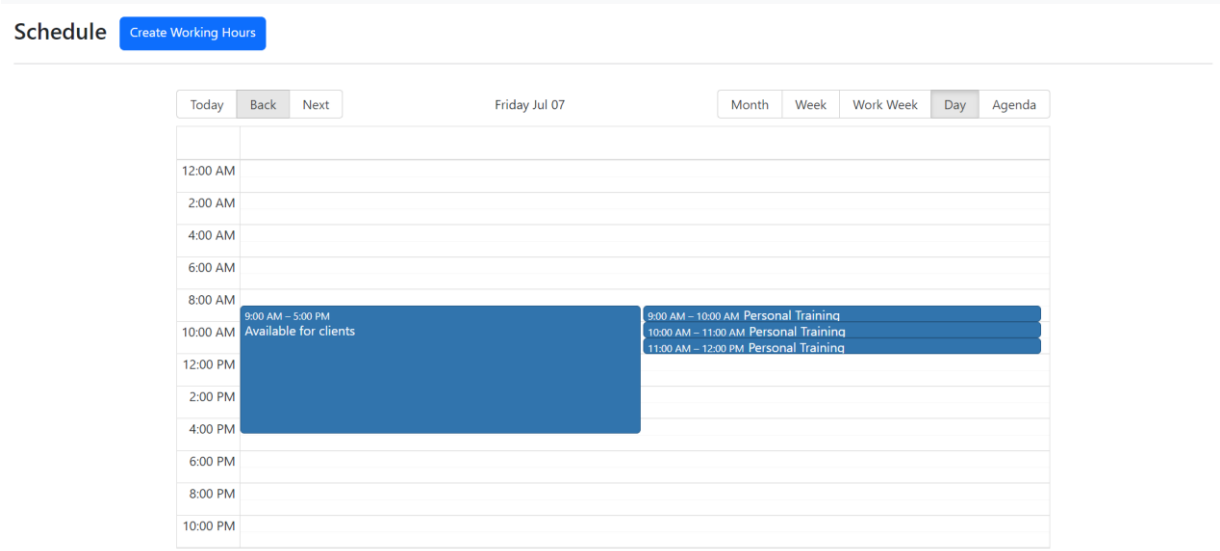


Рисунок 4.7 – Розклад працівника

Якщо в працівника немає графіка роботи, то виводиться відповідне інформаційне повідомлення (рис 4.8).

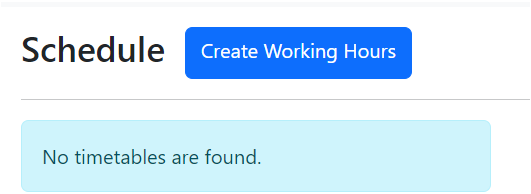


Рисунок 4.8 – Інформаційне повідомлення про відсутність робочого розкладу

Для того, щоб створити робочий графік працівника, потрібно натиснути на кнопку Create Working Hours. Після цього відобразиться відповідна форма (рис. 4.9). Менеджер може ввести час початку і кінця роботи працівника. Далі потрібно обрати робочі дні тижня і встановити початкову і кінцеву дату, до

якої буде повторюватись це правило робочого розкладу працівника. Після підтвердження менеджера перенаправить на сторінку з розкладом працівника.

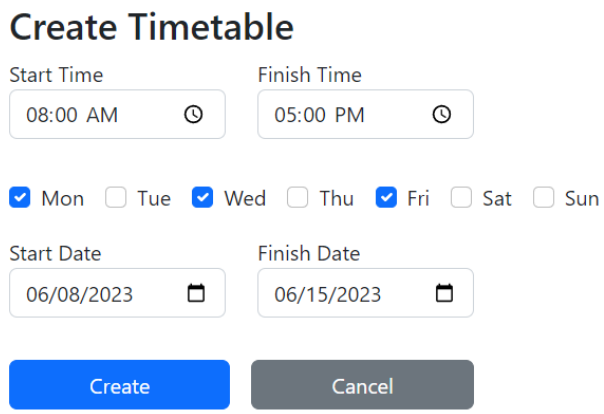


Рисунок 4.9 – Форма створення робочих часів працівника

Також на сторінці працівника можна переглянути його прийоми за певну дату і, якщо потрібно, відмінити прийом (рис. 4.10). При натисканні кнопки Cancel прийом буде відмінено.

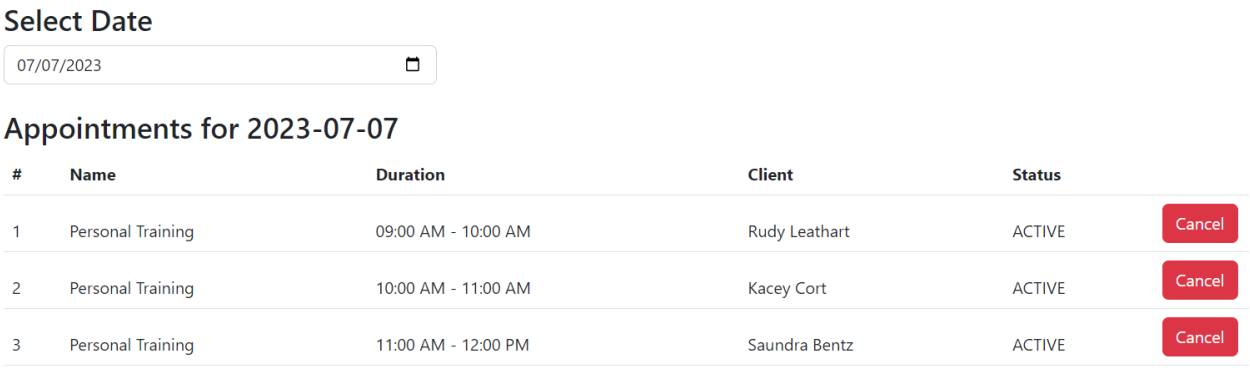


Рисунок 4.10 – Список прийомів працівника

На сторінці пошуку працівників можна додати нового працівника натиснувши кнопку Create. Після цього виводится форма вибору посади працівника (рис 4.11).

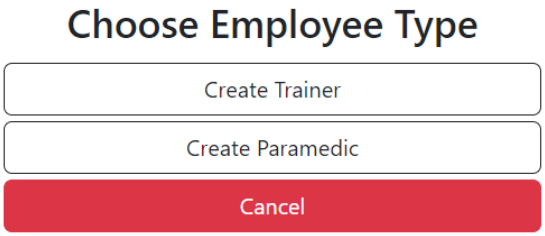


Рисунок 4.11 – Форма вибору посади працівника, якого потрібно додати

Після обрання посади відкривається форма, де потрібно ввести дані працівника (рис. 4.12).

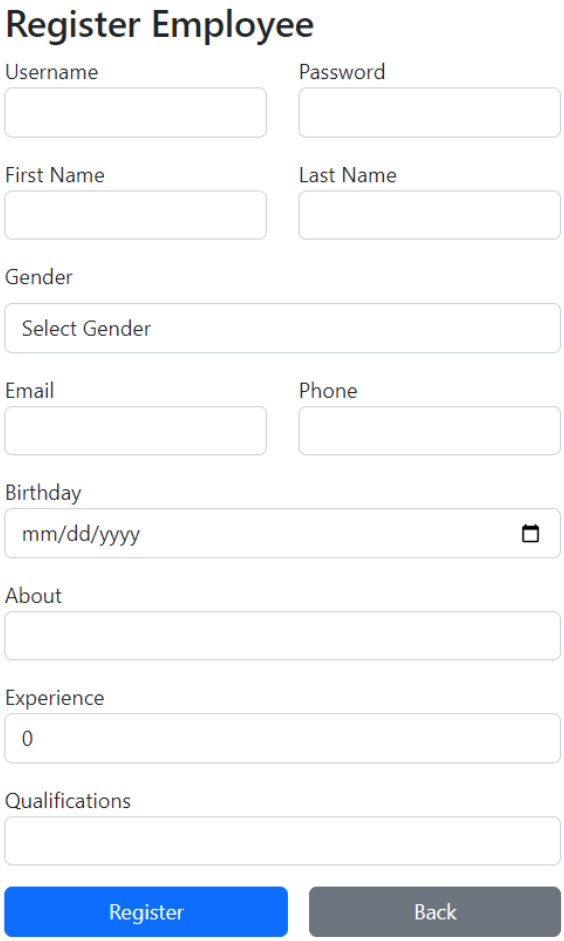


Рисунок 4.12 – Форма створення співпрацівника

При переході на вкладку Client, яка розташована на навігаційній панелі, менеджер отримує доступ до функціоналу обліку клієнтів. Аналогічно з обліком персоналу, ця сторінка має форму пошуку клієнтів за ім'ям (рис. 4.13).

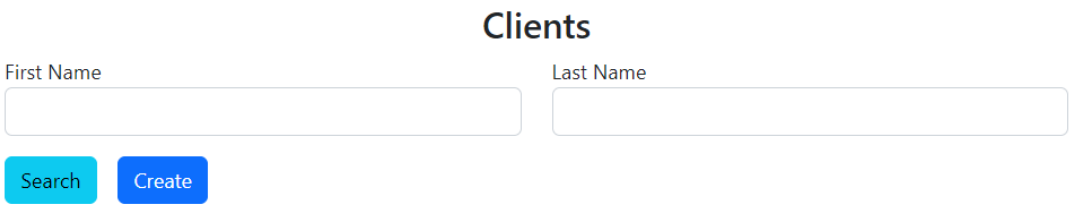


Рисунок 4.13 – Форма пошуку клієнтів

Після натискання кнопки Search відбувається пошук клієнтів із заданими параметрами і виводиться список, який аналогічний списку працівників. Якщо натиснути на ім’я клієнта, то можна перейти на його профіль (рис. 4.14).

Рисунок 4.14 – Профіль клієнта

За допомогою кнопок Training Appointment та Observation appointment можна створити персональне тренування або медичне обстеження відповідно. Якщо клієнт не має абонемента, то ці кнопки не відображаються.

Створення обох типів прийомів відбувається однаковим чином. Спочатку менеджеру пропонується обрати працівника і його робочі часи (рис. 4.15). Також для оптимального вибору разом з робочими часами працівника надаються дані про його прийоми.

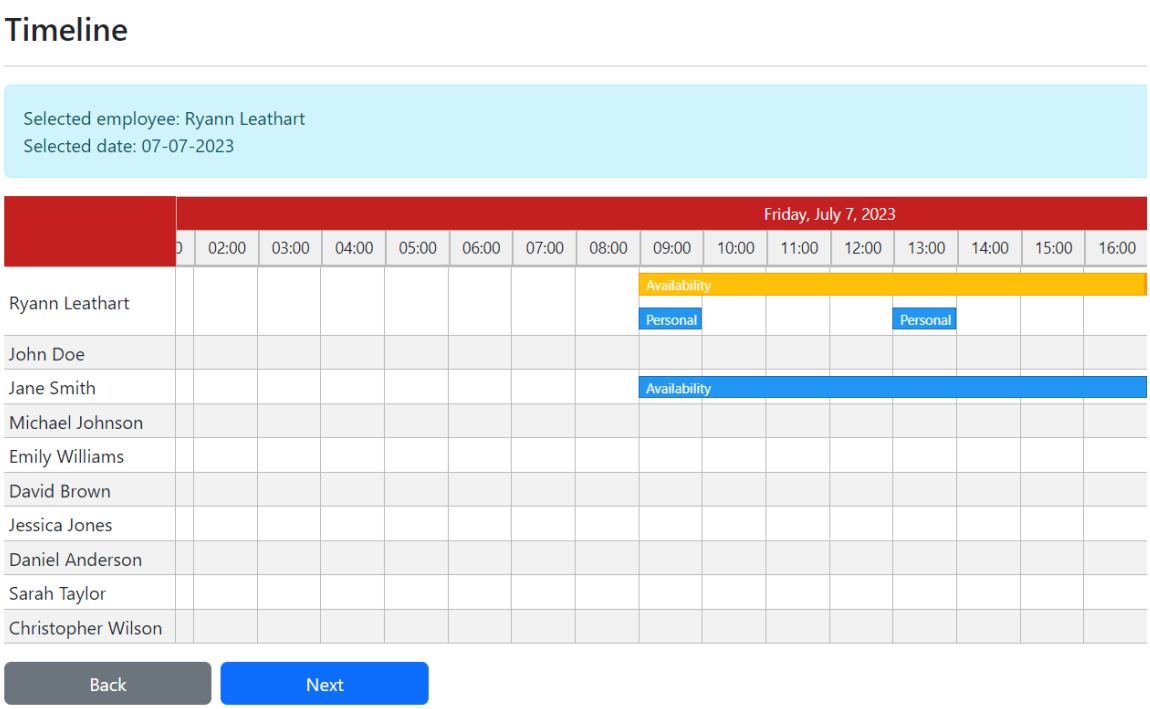


Рисунок 4.15 – Вибір працівника і його робочого часу

Наступний крок – це обрання вільних часових проміжків за обрану дату. При бажані можна обрати іншу дату на календарі (рис 4.16).

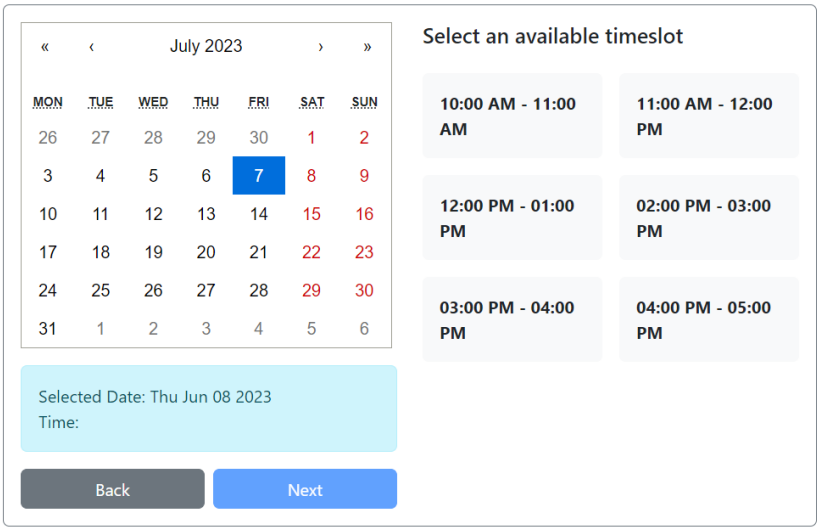


Рисунок 4.16 – Обрання часового проміжку при створенні нового прийому

Далі відбувається заповнення назви та опису прийому і підтвердження його створення (рис. 4.17).

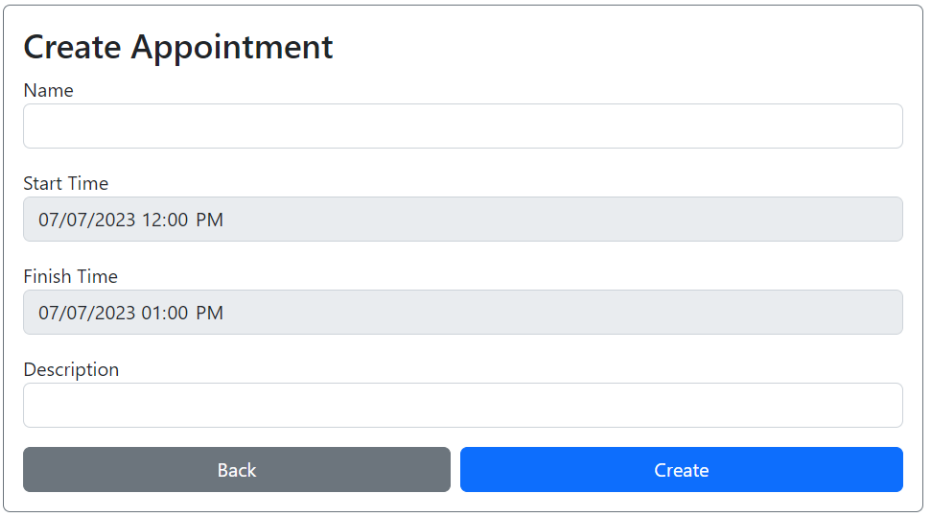


Рисунок 4.17 – Форма заповнення деталей прийому

За допомогою кнопки Assign Subscription, що розташована на профілі клієнта, можна переглянути абонемент клієнта і відмінити його (рис. 4.18).

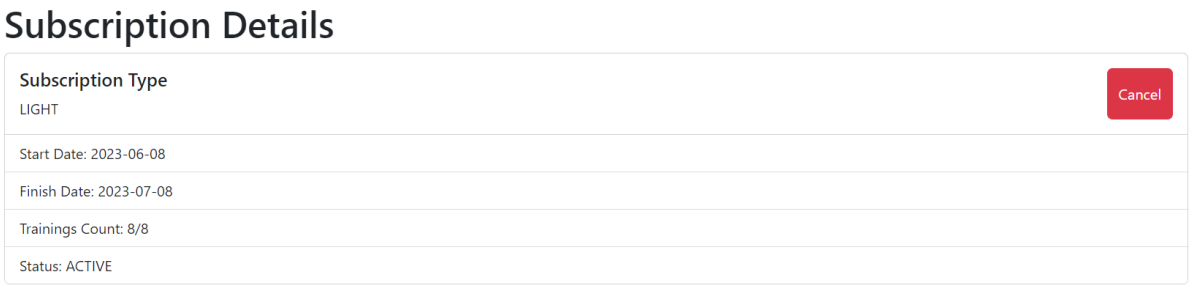


Рисунок 4.18 – Деталі абонемента

Якщо абонемента немає, то виводиться відповідне повідомлення з кнопкою для створення нового абонемента (рис. 4.19).

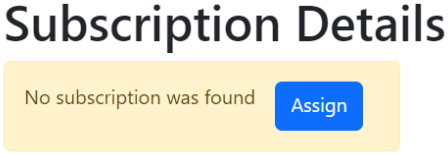


Рисунок 4.19 – Повідомлення про відсутність абонемента

При натисканні на кнопку Assign виводиться форма створення нового абонемента з можливістю обрати один з варіантів абонемента (рис. 4.20).

Рисунок 4.20 – Форма створення нового абонемента

Пункт навігаційної панелі Analytics відповідає за надання статистичних даних фітнес-центру (рис. 4.21): рейтинг популярності абонементів, кількість нових клієнтів помісячно та рівень зайнятості фітнес-центру по днях тижня. Дані для зручності та наочності подаються у вигляді графіків та діаграм.

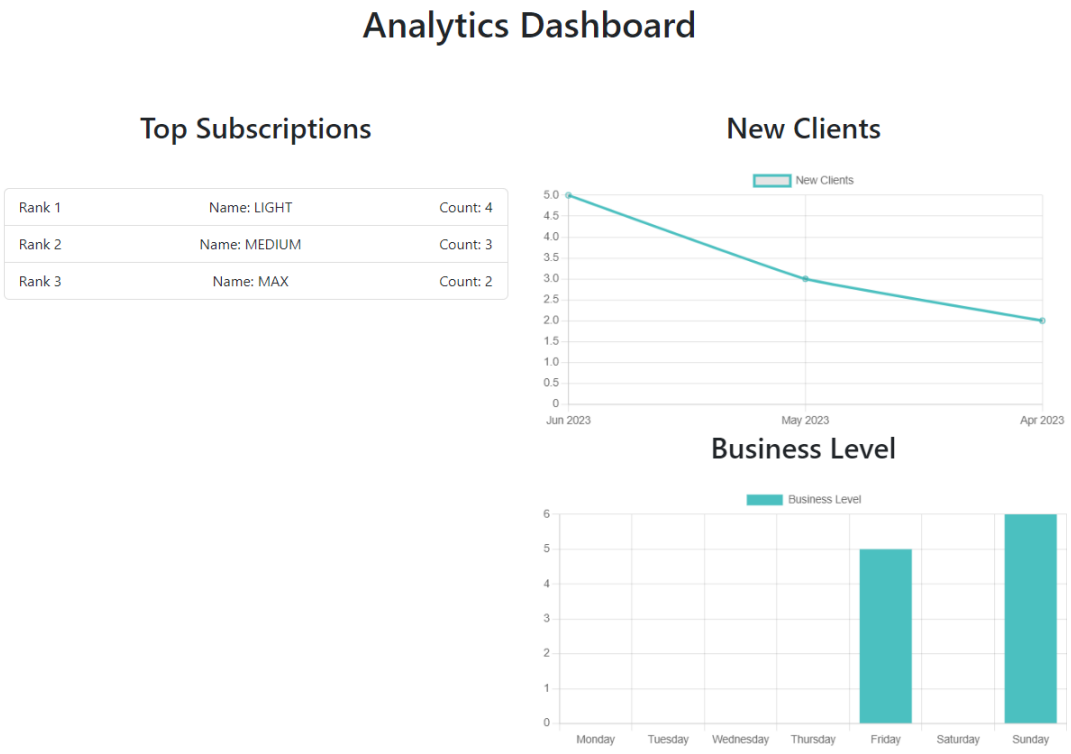


Рисунок 4.21 – Статистичні дані фітнес-центру

4.1.2 Опис інтерфейсу медпрацівника

Профіль медпрацівника виглядає так само як і профіль менеджера. Але на відміну від менеджера він має свою специфічну навігаційну панель (рис. 4.22).

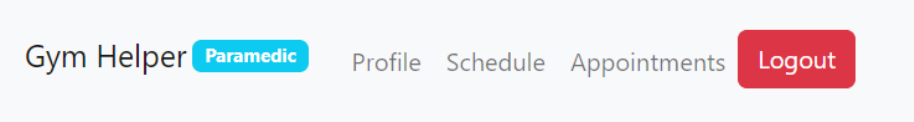


Рисунок 4.22 – Навігаційна панель медпрацівника

При натисканні кнопки Schedule медпрацівник отримує інформацію про свій розклад у тому самому вигляді як і менеджер отримує інформацію про розклад працівників. При натисканні кнопки Appointments навігаційної панелі медпрацівник отримує інформацію про свої заплановані медичні обстеження на певну дату (рис. 4.23).

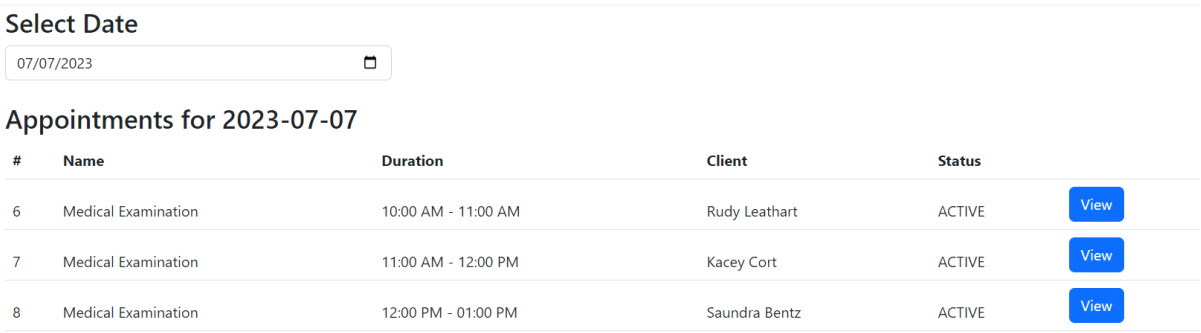


Рисунок 4.23 – Список запланованих обстежень на певну дату

При натисканні кнопки View, розташованої на елементі списку, можна відкрити деталі запланованого обстеження і розпочати його (рис. 4.24). Для того, щоб розпочати прийом, потрібно натиснути кнопку Start. Після цього встановиться фактичний час початку прийому і стане доступною кнопка Finish

для закінчення прийому. Кнопки Next та Previous дозволяють керувати кроками обстеження.

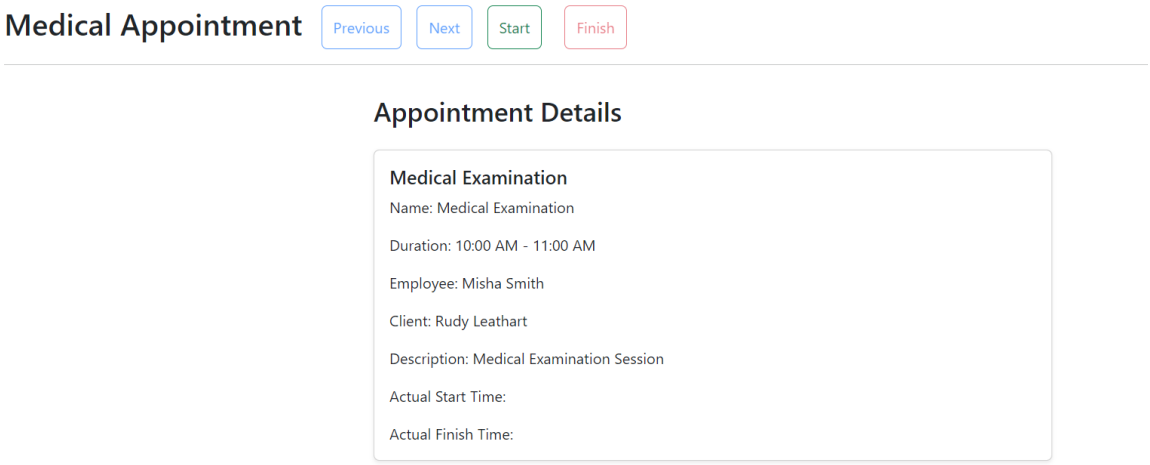


Рисунок 4.24 – Форма деталей прийому

Другий крок медичного обстеження – це фіксування записів медпрацівника (рис. 4.25). З лівого боку розташовані форми для додавання записів, а з правого – історія медичних записів клієнта.

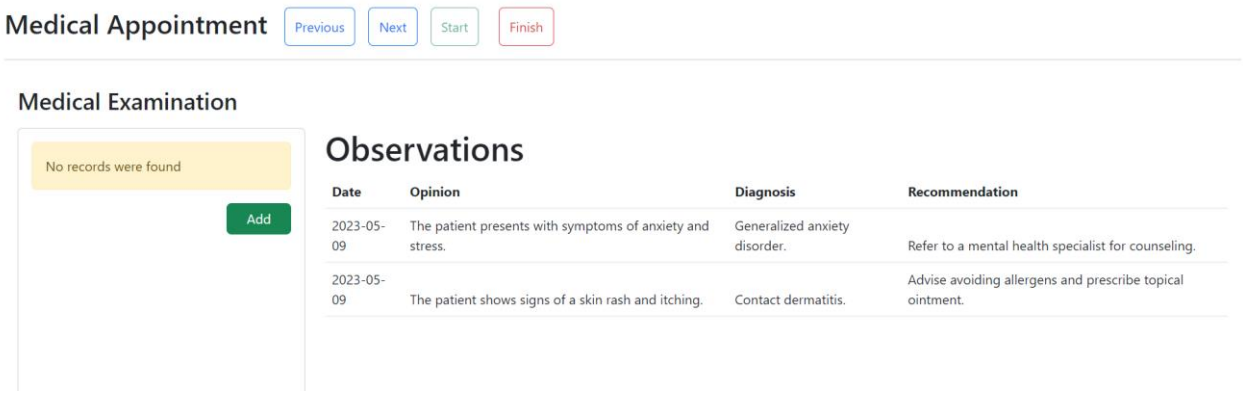


Рисунок 4.25 – Форма медичних обстежень

На третьому кроці при бажанні можна зафіксувати фізіологічні дані клієнта (рис 4.26). З лівого боку розташована форма для внесення даних, а з правого – історія вимірів фізіологічних покажчиків клієнта у вигляді графіку, що наочно демонструє зміни з плином часу.

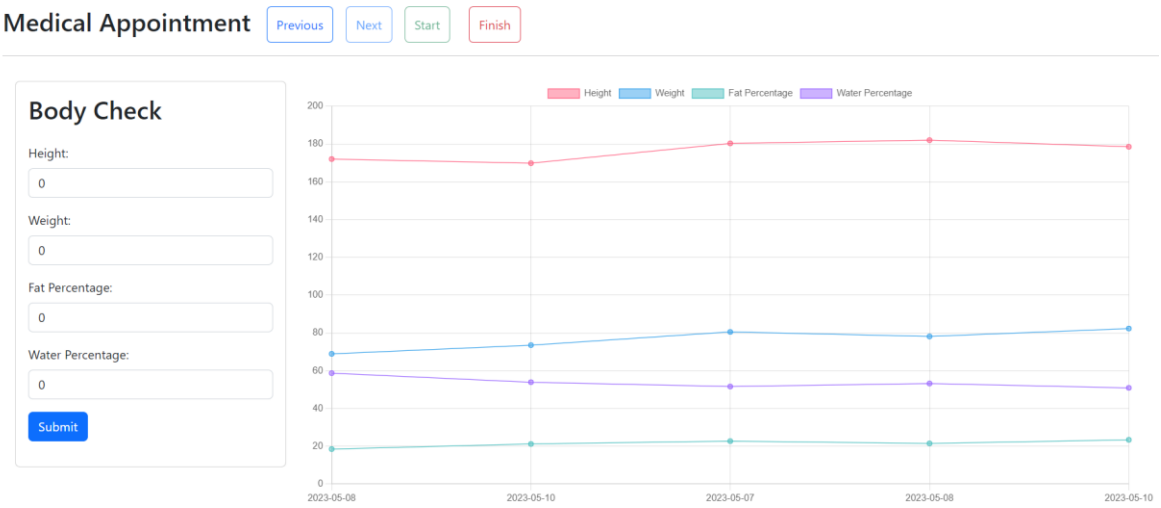


Рисунок 4.26 – Форма вимірів фізіологічних покажчиків клієнта

4.2 Опис мобільних застосунків тренера та клієнта фітнес-центру

4.2.1 Опис інтерфейсу тренера

При відкритті застосунку тренеру потрібно ввести свої облікові дані (рис. 4.27). Після успішної авторизації відображається профіль тренера (рис. 4.28).

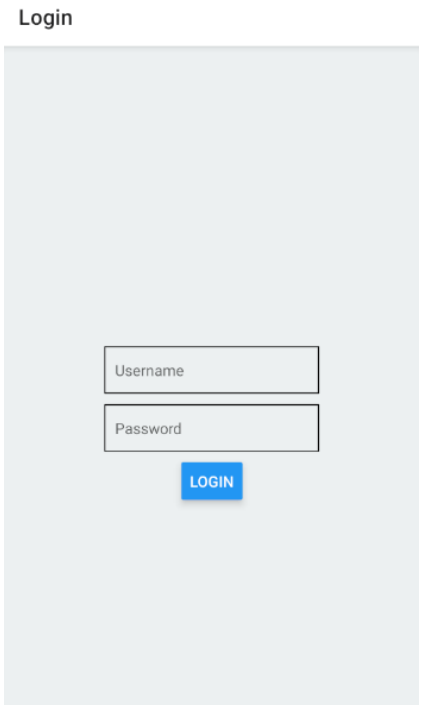


Рисунок 4.27 – Форма авторизації мобільного застосунку

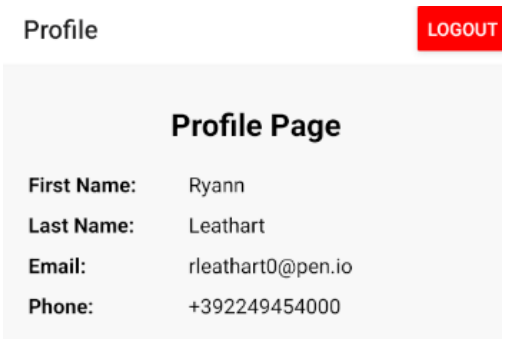


Рисунок 4.28 – Профіль тренера

Тренер має навігаційну панель, що зображена на рис. 4.29.

Рисунок 4.29 – Навігаційна панель тренера

При натисканні кнопки Clients відкривається екран, що відображає список клієнтів, які закріплені за тренером (рис. 4.30).

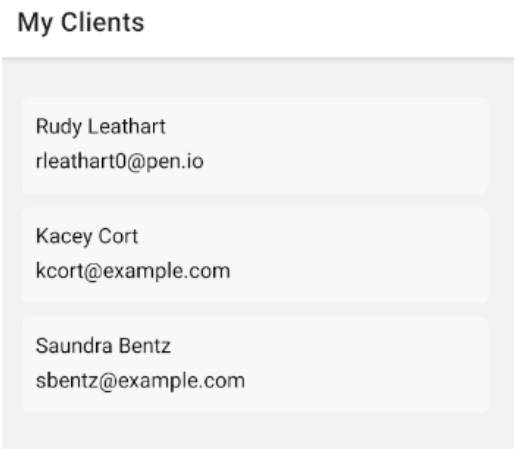
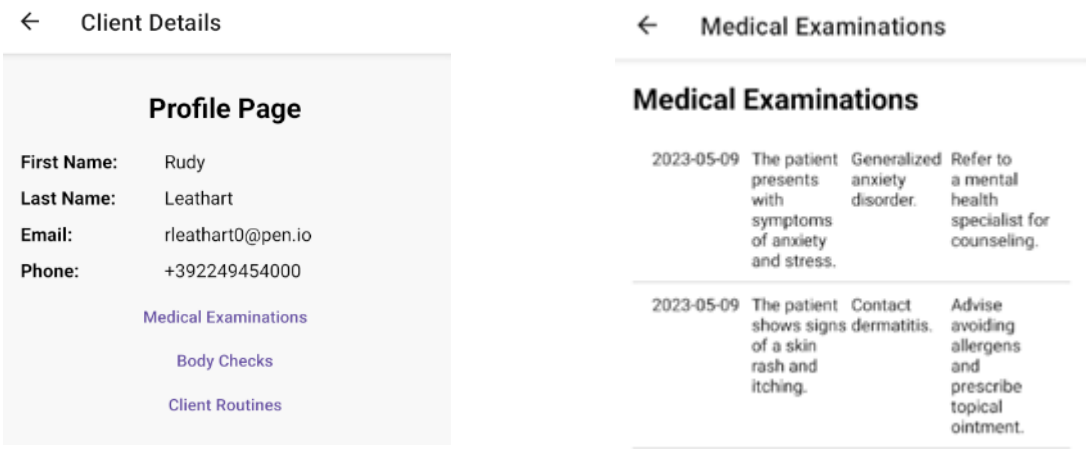


Рисунок 4.30 – Список клієнтів тренера

При натисканні на запис клієнта відкривається профіль клієнта з персональною інформацію та можливістю відкрити дані його медичних обстежень, дані фізіологічних покажчиків та програми тренувань (рис. 4.31, а).

При натисканні на кнопку Medical Examinations показується екран з медичними обстеженнями клієнта (рис. 4.31, б).



а) б)

Рисунок 4.31 – Профіль клієнта (а) та дані про медичні обстеження клієнта (б)

При натисканні на кнопку Body Checks відображається екран з фізіологічними покажчиками клієнта (рис. 4.32).

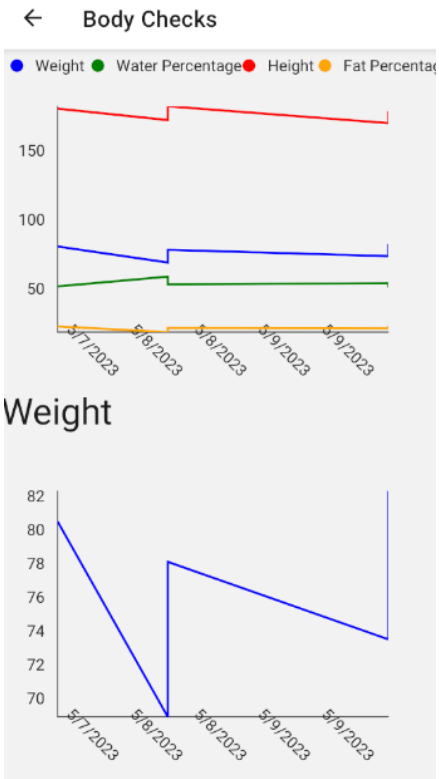
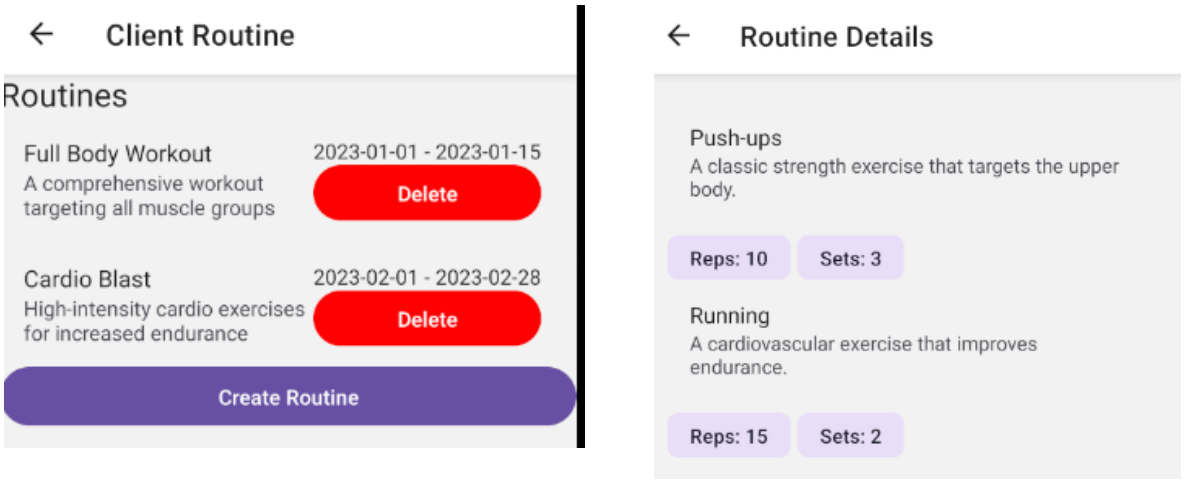


Рисунок 4.32 – Фізіологічні покажчики клієнта

При натисканні на кнопку Client Routines відображається екран, що містить список програм тренувань клієнта з можливістю їх видалення (рис. 4.33, а).

При натисканні на назву програми тренувань можна побачити опис кожної з вправ, що в неї входять, з зазначенням кількості підходів та повторів (рис. 4.33, б).



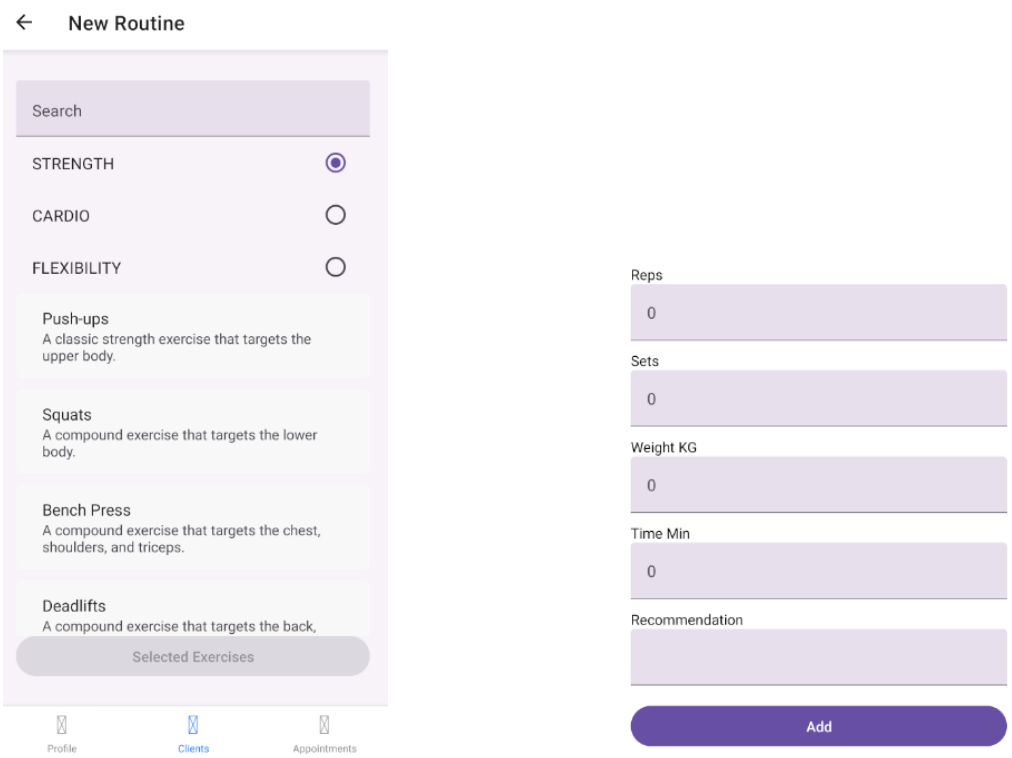
а) б)

Рисунок 4.33 – Програми тренувань клієнта (а) та деталі програми тренувань (б)

Для створення нової програми тренувань потрібно натиснути кнопку Create Routine, після чого відображається екран, що містить форму пошуку вправи за ім’ям та типом вправи (рис. 4.34, а).

Для того, щоб обрати вправу, потрібно на неї натиснути. Після цього з’являється форма налаштування вправи (кількість підходів, кількість повторів, час виконання та вага) і додавання рекомендацій (рис. 4.34, б).

Переглянути вправи, які були додані до програми тренувань, можна за допомогою натискання кнопки Selected Exercises, що знаходиться внизу форми створення програми тренувань. Відкривається екран, що містить стислу інформацію про додані вправи (рис. 4.35).



а) б)

Рисунок 4.34 – Форма пошуку вправи (а) та форма налаштування вправи (б)

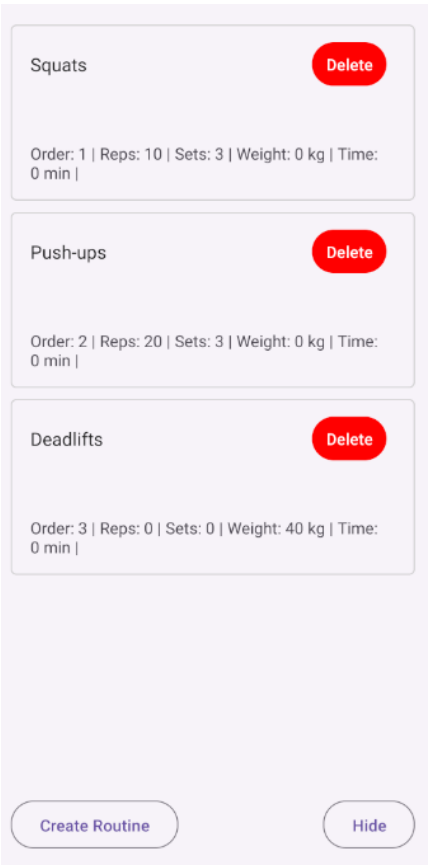


Рисунок 4.35 – Обрані для програми тренувань вправи

Порядок обраних вправ можна змінювати шляхом переміщення елементів списку.

Для того, щоб створити програму тренувань з обраними та налаштованими вправами, потрібно натиснути кнопку Create Routine. Після цього з'явиться форма для заповнення деталей програми тренувань, що зображена на рис. 4.36.

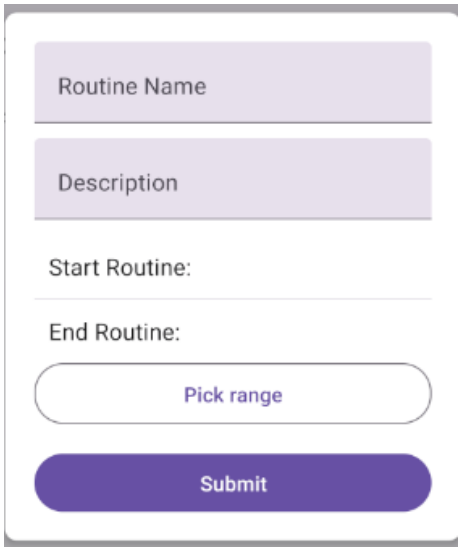
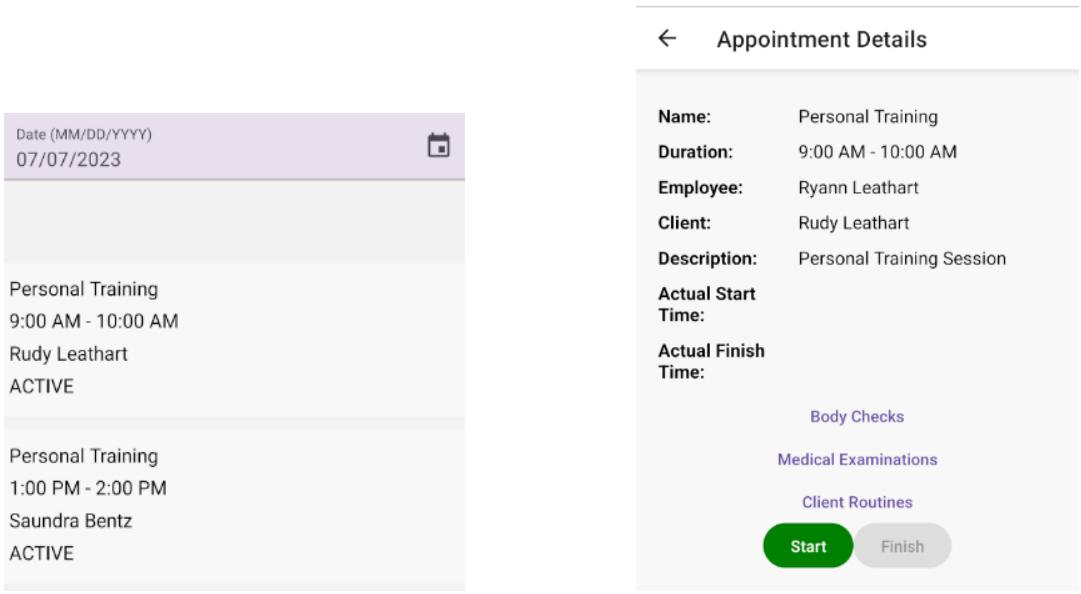


Рисунок 4.36 – Форма заповнення деталей програми тренувань

Для того, щоб тренер міг переглянути свої заплановані тренування з клієнтами, йому потрібно натиснути кнопку Appointments на панелі навігації. Після цього буде виведений список персональних тренувань за певну дату (рис. 4.37, а). Про тренування відображаються такі дані як час тренування, ім’я клієнта та статус тренування.

При натисканні на персональне тренування відкривається екран його детального описання, що відображено на рис. 4.37, б. На За допомогою кнопок Start та Finish тренер повинен зафіксувати фактичний час початку і кінця тренування. З цієї ж форми тренер може отримати доступ до історії фізіологічних покажчиків клієнта, даних його медичних обстежень та всіх програм тренувань цього клієнта.



а) б)

Рисунок 4.37 – Список тренувань за певну дату (а) та форма проведення персонального тренування (б)

4.2.2 Опис інтерфейсу клієнта фітнес-центру

При відкритті застосунку клієнта фітнес-центру йому потрібно ввести свої облікові дані на формі авторизації. Після успішної авторизації відображається профіль клієнта (рис. 4.38), де він може переглянути історію своїх медичних обстежень фізіологічних покажчиків та персональної програми тренувань.

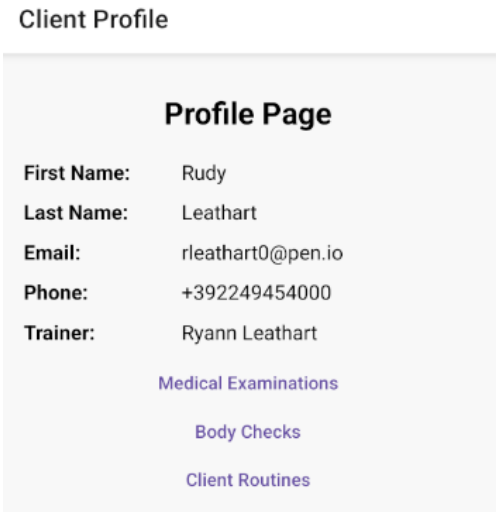
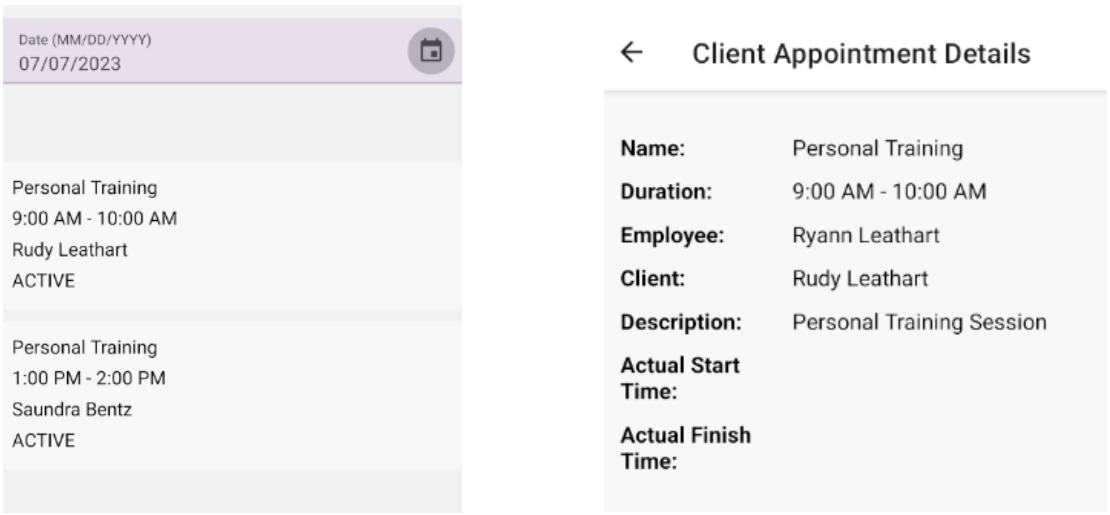


Рисунок 4.38 – Форма авторизації мобільного застосунку клієнта фітнес-центру

Клієнт фітнес-центру має навігаційну панель, що зображена на рис. 4.39.

Рисунок 4.39 – Навігаційна панель клієнта

Якщо перейти за пунктом Appointments навігаційної панелі, то можна переглянути медичні обстеження і персональні тренування клієнта за певну дату (рис 4.40, а). При натисканні на прийом можна переглянути детальну інформацію про відповідне медичне обстеження чи персональне тренування. На рис 4.40, б зображена детальна інформація про персональне тренування.



а) б)

Рисунок 4.40 – Список прийомів клієнта за певну дату (а) та форма деталей прийому (б)

ВИСНОВКИ

При виконанні дипломної роботи проведено дослідження діяльності фітнес-центрів, що спеціалізуються на збереженні та покращенні стану здоров’я клієнтів, та визначені бізнес-процеси, які потребують автоматизації, а саме: облік абонементів клієнтів фітнес-центру, призначення клієнту персоналізованої програми тренувань з урахуванням його фізіологічних особливостей, зберігання історії фізіологічного стану клієнта та даних його медичних обстежень, а також облік працівників фітнес-центру та керування їхнім розкладом роботи.

В результаті аналізу конкурентних програмних продуктів виявлено, що жоден з них не надає функціонал, акцентований на здоров’ї клієнта, що є важливим для зниження рівня безпеки під час зайняття спортом.

Створена розподілена система підтримки діяльності фітнес-центру забезпечує автоматизацію всіх вищезазначених бізнес-процесів. Перевагою системи є підтримка зберігання даних медичних обстежень клієнта та періодичних вимірювань його фізіологічних покажчиків. Це дозволяє слідкувати за станом здоров’я клієнта в динаміці, робити висновки і коригувати програми тренувань, що призначаються тренером.

Створена система є кросплатформною, для менеджера і медпрацівника передбачена можливість взаємодії з нею за допомогою персонального комп’ютера, а для тренера та клієнта – за допомогою мобільних пристроїв, що працюють під управлінням ОС Android та iOS. Кросплатформність досягнута завдяки використанню технологій Інтернет та засобів створення веб-застосунків, таких як мови програмування Java і JavaScript та бібліотеки React і React Native. Для зберігання даних використана СУБД PostgreSQL.

Система призначена для впровадження у фітнес-центрах з одним філіалом. Незважаючи на це, вона може бути легко розширена та доповнена необхідним функціоналом для підтримки діяльності мережі фітнес-центрів, а також, завдяки наявності REST API, може бути інтегрована з зовнішніми інформаційними системами медичних установ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. World Health Organization. Geneva: World Health Organization; 2020. Physical inactivity: a global public health problem [Електронний ресурс] – Режим доступу: https://www.who.int/dietphysicalactivity/factsheet_inactivity/en/
2. Васеньшев Б. О., Розновець О. І. Розподілена інформаційна система підтримки діяльності фітнес-центру / Інформатика, інформаційні системи та технології: тези доповідей двадцятої всеукраїнської конференції студентів і молодих науковців. // Одеса, 28 квітня 2023 р. – Одеса, 2023. – с.93-94
3. GymMaster, Resource for Customers [Електронний ресурс] – Режим доступу: https://www.who.int/dietphysicalactivity/factsheet_inactivity/en/
4. MindBody, Business [Електронний ресурс] – Режим доступу: https://www.mindbodyonline.com/business
5. ZenPlanner, Resources [Електронний ресурс] – Режим доступу: https://zenplanner.com/resources
6. Alan Klement, The Jobs to be Done Data Model [Електронний ресурс] – Режим доступу: https://jtbd.info/the-jobs-to-be-done-data-model-b270f6fc445
7. Sohel, Shanewaz & Rahman, Abu & Uddin, Md. Competitive Profile Matrix (CPM) as a Competitors' Analysis Tool: A Theoretical Perspective // International Journal of Human Potential Development, – 2014. – 40-47.
8. Apple Developer, The Model-View-Controller (MVC) design pattern [Електронний ресурс] – Режим доступу: https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html
9. REST API [Електронний ресурс] – Режим доступу: https://wiki.cuspu.edu.ua/index.php/REST_API
10. The Destination for Java Developers [Електронний ресурс] – Режим доступу: https://dev.java/
11. Spring [Електронний ресурс] – Режим доступу: https://spring.io/
12. JavaScript [Електронний ресурс] – Режим доступу: https://www.javascript.com/

13. PostgreSQL [Електронний ресурс] – Режим доступу: https://www.postgresql.org/
14. IntelliJ IDEA [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/idea/
15. React [Електронний ресурс] – Режим доступу: https://react.dev/
16. React Native [Електронний ресурс] – Режим доступу: https://reactnative.dev/
17. Apache Tomcat [Електронний ресурс] – Режим доступу: https://tomcat.apache.org/

ДОДАТОК А

Сценарій створення бази даних

```
CREATE DOMAIN phone VARCHAR(13)
     CHECK(VALUE SIMILAR TO '\+[0-9]{12}');

CREATE DOMAIN email VARCHAR(255)
     CHECK(VALUE LIKE '%_@_%._%');

CREATE TYPE appointment_status AS ENUM (
    'ACTIVE',
    'CANCELED',
    'FINISHED'
);

CREATE TYPE appointment_type AS ENUM (
    'MEDICAL_EXAMINATION',
    'PERSONAL_TRAINING'
);

CREATE TYPE employee_type AS ENUM (
    'MANAGER',
    'TRAINER',
    'PARAMEDIC'
);

CREATE TYPE exercise_type AS ENUM (
    'STRENGTH',
    'CARDIO',
    'FLEXIBILITY'
);

CREATE TYPE gender_type AS ENUM (
    'MALE',
    'FEMALE',
    'NONBINARY'
);

CREATE TYPE role_type AS ENUM (
    'MANAGER',
    'TRAINER',
    'PARAMEDIC',
    'CLIENT'
);
```

```
CREATE TYPE subscription_status_type AS ENUM (
    'ACTIVE',
    'CANCELED',
    'EXPIRED'
);

CREATE TYPE subscription_type_type AS ENUM (
    'LIGHT',
    'MEDIUM',
    'MAX'
);

CREATE TABLE user_credentials (
  id serial NOT NULL PRIMARY KEY,
  password varchar(256) NOT NULL,
  role role_type NOT NULL,
  username varchar(50) NOT NULL
);

CREATE TABLE employee (
  employee_type employee_type NOT NULL,
  id serial NOT NULL PRIMARY KEY,
  about varchar(256),
  birthday date  CHECK (birthday <= current_date - interval '18
years'),
  email email NOT NULL,
  experience int4 CHECK (experience>0) NOT NULL,
  first_name varchar(50) NOT NULL,
  gender gender_type NOT NULL,
  hire_date date CHECK (hire_date <= current_date) NOT NULL,
  last_name varchar(50) NOT NULL,
  phone phone NOT NULL,
  qualifications varchar(256),
  user_credentials_id int8 NOT NULL REFERENCES user_credentials
(id)
);

CREATE TABLE client (
  id serial NOT NULL PRIMARY KEY,
  about varchar(256),
  birthday date  CHECK (birthday <= current_date - interval '18
years'),
  email email NOT NULL,
  first_name varchar(50) NOT NULL,
  gender gender_type NOT NULL,
  last_name varchar(50) NOT NULL,
```

```
  phone phone NOT NULL,
  start_date date CHECK (start_date <= current_date) NOT NULL,
  trainer_id int8 NOT NULL REFERENCES employee (id) NOT NULL,
  user_credentials_id int8 NOT NULL REFERENCES user_credentials
(id)
);

CREATE TABLE appointment (
  appointment_type appointment_type,
  id serial NOT NULL PRIMARY KEY,
  actual_finish_time timestamp(6) CHECK (actual_finish_time >
actual_start_time),
  actual_start_time timestamp(6) CHECK (actual_start_time <=
current_timestamp),
  client_id int8 NOT NULL REFERENCES client (id),
  description varchar(255),
  employee_id int8 NOT NULL REFERENCES employee (id),
  finish_time timestamp(6) CHECK(finish_time>start_time) NOT NULL,
  name varchar(255),
  start_time timestamp(6) CHECK (start_time>= current_timestamp)
NOT NULL,
  appointment_status appointment_status,
  fat_percentage float8 check(fat_percentage>0) NOT NULL,
  height float8 check(height>0) NOT NULL,
  water_percentage float8 check(water_percentage>0) NOT NULL,
  weight float8 check(weight>0) NOT NULL
);

CREATE TABLE exercise (
  id serial NOT NULL PRIMARY KEY,
  description varchar(256) NOT NULL,
  exercise_type exercise_type NOT NULL,
  name varchar(256) NOT NULL
);

CREATE TABLE medical_examination (
  id serial NOT NULL PRIMARY KEY,
  appointment_id int8 NOT NULL REFERENCES appointment (id),
  date_of_examination DATE CHECK (date_of_examination <=
current_date) NOT NULL,
  diagnosis varchar(256) NOT NULL,
  opinion varchar(256) NOT NULL,
  recommendation varchar(256)
);

CREATE TABLE routine (
```

```
  id serial NOT NULL PRIMARY KEY,
  description varchar(255),
  finish_date date,
  name varchar(255),
  start_date date,
  client_id int8 NOT NULL REFERENCES client (id),
  trainer_id int8 NOT NULL REFERENCES employee (id)
);

CREATE TABLE routine_exercise (
  id serial NOT NULL PRIMARY KEY,
  description varchar(255),
  repetitions int4 CHECK(repetitions>=0),
  recommendation varchar(255),
  sets int4 CHECK(sets>=0),
  time_min int4 CHECK(time_min>=0),
  weight_kg int4 CHECK(weight_kg>=0),
  exercise_id int8 NOT NULL REFERENCES exercise (id),
  routine_id int8 NOT NULL REFERENCES routine (id)
);

CREATE TABLE working_hours (
  id serial NOT NULL PRIMARY KEY,
  employee_id int8 NOT NULL REFERENCES employee (id),
  finish_time timestamp(6) check(finish_time>start_time) NOT NULL,
  start_time timestamp(6) check(start_time>=current_timestamp)
NOT NULL
);

create table subscription (
  id bigserial not null,
  subscription_type subscription_type_type NOT NULL,
  training_limit integer check(training_limit>0),
  primary key (id)
);

create table client_subscription (
  id bigserial not null,
  client_id bigint  NOT NULL REFERENCES client(id),
  finish_date date check (finish_date > start_date) NOT NULL,
  start_date date check (start_date >= current_date) NOT NULL,
  status subscription_status_type  NOT NULL,
  subscription_id bigint  NOT NULL REFERENCES subscription(id),
  training_left integer  NOT NULL,
  primary key (id)
);
```

ДОДАТОК Б

Тригери і функції

Функція для отримання вільних часових проміжків з зазначеним інтервалом враховуючи зайняті проміжки та робочий розклад відповідного співпрацівника (лістинг Б.1).

```
CREATE OR REPLACE FUNCTION getFreeTimeslots(
    p_date DATE,
    p_employee_id INTEGER,
    p_interval_value INTEGER
)
RETURNS TABLE (start_time TIMESTAMP, finish_time TIMESTAMP) AS $$
DECLARE
    p_start_time TIMESTAMP;
    p_end_time TIMESTAMP;
BEGIN
    p_start_time := p_date::timestamp;
     p_end_time := p_date::timestamp + '23 hours 59 minutes 59
seconds'::interval;
    RETURN QUERY
         WITH timeslots AS (
            SELECT
                generate_series AS start_time,
                     (generate_series + (p_interval_value || '
minutes')::interval) AS finish_time
                FROM generate_series(p_start_time, p_end_time -
(p_interval_value || ' minute')::interval, (p_interval_value || '
minute')::interval)
        ),
        slots_in_use AS (
            SELECT DISTINCT t.start_time, t.finish_time
            FROM timeslots AS t
                 JOIN appointment AS r ON ( ((t.start_time,
t.finish_time) OVERLAPS (r.start_time, r.finish_time)))
            ORDER BY t.start_time, t.finish_time
        ),
        working_schedule_slots AS (
            SELECT DISTINCT t.start_time, t.finish_time
            FROM timeslots AS t
```

Лістинг Б.1 – Функція отримання вільних часових проміжків

```
          JOIN   working_schedule   AS   ws   ON   p_employee_id=
ws.employee_id  AND  (  ((t.start_time,  t.finish_time)  OVERLAPS
(ws.start_time, ws.finish_time)))
            ORDER BY t.start_time, t.finish_time),
        free_timeslots AS (
            SELECT * FROM timeslots as x
            WHERE
                NOT EXISTS (
                SELECT 1
                FROM slots_in_use AS s
                WHERE x.start_time = s.start_time
                AND x.finish_time = s.finish_time
            )
            AND
            EXISTS(
            SELECT 1
                FROM working_schedule_slots AS wss
                WHERE x.start_time = wss.start_time
                AND x.finish_time = wss.finish_time
            )
        ) SELECT * FROM free_timeslots;
END;
$$ LANGUAGE plpgsql;
```

Лістинг Б.1, лист 2

Тригерна функція та тригер для перевірки існування прийому у заданого працівника при створенні нового прийому або оновлення вже існуючого для того, щоб уникнути накладки (лістинг Б.2).

```
CREATE OR REPLACE FUNCTION check_appointment_overlap()
  RETURNS TRIGGER AS $$
BEGIN
  IF EXISTS (
    SELECT 1
    FROM appointment
    WHERE employee_id = NEW.employee_id
        AND  (start_time,  finish_time)  OVERLAPS  (NEW.start_time,
NEW.finish_time)  )
```

Лістинг Б.2 – Тригерна функція та тригер для перевірки існування прийому у заданого працівника

```
     THEN
    RAISE EXCEPTION 'An overlapping appointment already exists.';
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER appointment_overlap_trigger
BEFORE INSERT OR UPDATE ON appointment
FOR EACH ROW
EXECUTE FUNCTION check_appointment_overlap();
```

Лістинг Б.2, лист 2

Тригерна функція та тригер для зменшення кількості решти тренувань, що залишилися згідно з існуючим абонементом після створення нової персонального тренування. Якщо тренування вичерпались, то абонемент переходить у статус «Закінчений» (лістинг Б.3).

```
CREATE OR REPLACE FUNCTION check_and_update_training_left()
RETURNS TRIGGER AS $$
DECLARE
    v_training_left INTEGER;
BEGIN
    IF NEW.appointment_type = 'PERSONAL_TRAINING' THEN
             BEGIN
            SELECT training_left
            INTO STRICT v_training_left
            FROM client_subscription
            WHERE client_id = NEW.client_id
               and status='ACTIVE';
        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                RAISE EXCEPTION 'No client subscription found for
the client.';
        END;
        UPDATE client_subscription
        SET training_left = training_left - 1
        WHERE client_id = NEW.client_id;
```

Лістинг Б.3 – Тригерна функція та тригер для зменшення кількості тренувань, що залишились, і змінення статус абонемента на «Закінчений»

```
          SELECT training_left
            INTO STRICT v_training_left
            FROM client_subscription
            WHERE client_id = NEW.client_id
               and status='ACTIVE';
        IF v_training_left = 0 THEN
            UPDATE client_subscription
            SET status = 'EXPIRED'
            WHERE client_id = NEW.client_id;
        END IF;
   END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_and_update_training_left_trigger
BEFORE INSERT ON appointment
FOR EACH ROW
EXECUTE FUNCTION check_and_update_training_left();
```

Лістинг Б.3, лист 2

ДОДАТОК В

ER-діаграма бази даних фітнес-центру

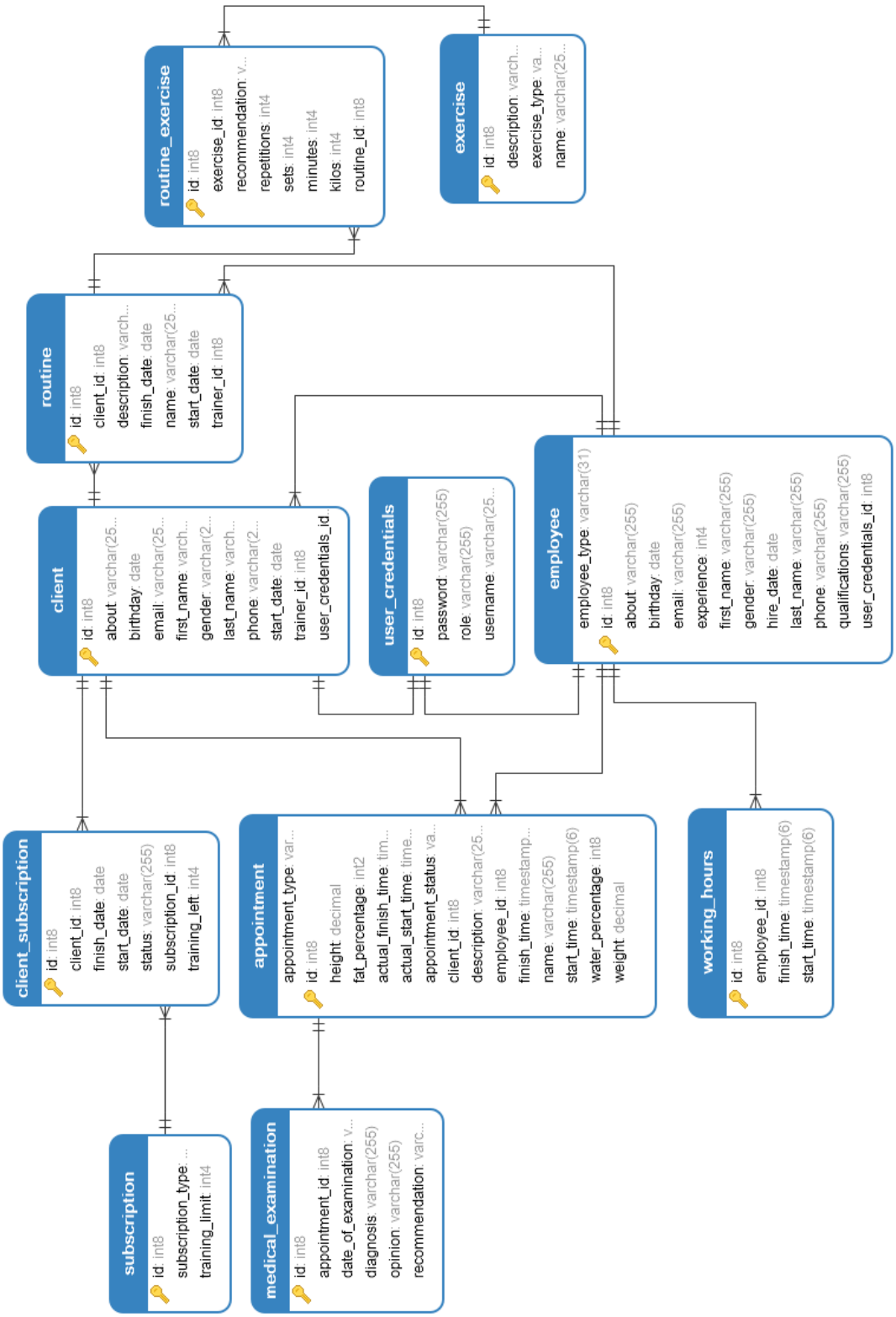


Рисунок В.1 – ER-діаграма бази даних фітнес-центру

ДОДАТОК Г

Типова структура компонентів програмного забезпечення на прикладі взаємодії класів Controller, Service, Repository та Mapper

Рисунок Г.1 – Типова структура компонентів програмного забезпечення на прикладі взаємодії класів Controller, Service, Repository та Mapper

ДОДАТОК Д

Приклади програмної реалізації компонентів користувацьких інтерфейсів

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import AuthService from '../api/AuthService';
import InputWithError from '../component/InputWithError';
import useAuth from '../hooks/useAuth';
import * as USERS from '../constants/users';
const LoginForm = () => {
    const [username, setUsername] = useState('');
    const [password, setPassword] = useState('');
    const [error, setError] = useState('');
    const [fieldError, setFieldError] = useState({});
    const { setAuth } = useAuth();
    const navigate = useNavigate();
    const handleLogin = async (e) => {
        e.preventDefault();
        const errors = {};
        if (!username) {
            errors.username = 'Username is required';
        }
        if (!password) {
            errors.password = 'Password is required';
        }
        setFieldError(errors);
        if (Object.keys(errors).length === 0) {
            try {
                setError('')
                const response = await
AuthService.login(username, password);
                resetState();
                const accessToken = response?.data?.token;
                const role = response?.data?.role;
                const id = response?.data?.id;
                localStorage.setItem('accessToken',
accessToken);
                setAuth({ role, accessToken, username, id })
                let path = "/error"
                if (USERS.ROLES.MANAGER === role) {
                    path = "/manager"
                } else if (USERS.ROLES.PARAMEDIC === role) {
                    path = "/paramedic"
```

```
                }
                navigate(path, { replace: true });
            } catch (error) {
                console.error(error)
                setError(error.message); }
        }
    };
    const resetState = () => {
        setUsername('');
        setPassword('');
    }
    return (
        <main className='d-flex justify-content-center align-
items-center min-vh-100'>
            <form onSubmit={handleLogin} autocomplete="on">
                <h1 className='h3 mb-3 text-center'>Employee
Login</h1>
                <InputWithError
                    type="text"
                    label="Username"
                    errorMessage={fieldError.username}
                    id="username"
                    value={username}
                    onChange={(e) =>
setUsername(e.target.value)}
                />
                <InputWithError
                    type="password"
                    label="Password"
                    errorMessage={fieldError.password}
                    id="password"
                    value={password}
                    onChange={(e) => {
setPassword(e.target.value) }}
                />
                <button type="submit" className='btn btn-lg btn-
primary float-right mb-3'>Log In</button>
                {error && <p className='alert alert-danger'
>{error}</p>}
            </form>
        </main >
    );
};
export default LoginForm;
```

Лістинг Д.1 – Форма авторизації працівника

```
const InputWithError = ({ value, label, placeholder, type,
onChange, id, name, errorMessage, defaultValue, disabled = false
}) => {
    return (
        <div className="form-group mb-3">
            <InputField
                value={value}
                id={id}
                placeholder={placeholder}
                type={type}
                onChange={onChange}
                name={name}
                label={label}
                defaultValue={defaultValue}
                disabled={disabled}
            />
            {errorMessage && <p className='text-
danger'>{errorMessage}</p>}
        </div>
    )
}
```

Лістинг Д.2 – Користувацький компонент для вводу даних