**Міністерство освіти і науки України**

**Державний університет інтелектуальних технологій та зв’язку**

Кафедра інженерії програмного забезпечення

**КУРСОВИЙ ПРОЄКТ**

з дисципліни “Організація баз даних та знать”

за темою: “Розробка інформаційної системи для співробітників фітнес-клубу”

студента 3-го курсу, групи ІПЗ-3.01

напряму підготовки 121 “Інженерія

програмного забезпечення”

Колюхова Олексія Ігоровича

Керівник Малахов Є. В.

Національна шкала

Кількість балів: Оцінка ECTS

Малахов Є.В.

(підпис) (прізвище та ініціали)

Члени комісії:

(підпис) (прізвище та ініціали)

(підпис) (прізвище та ініціали)

**ЗМІСТ**

[ВСТУП 2](#_iulkkcs899r9)

[1 ПОСТАНОВКА ЗАДАЧІ 4](#_me2gep3qsn4p)

[2 ІНФОРМАЦІЙНЕ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 6](#_erdp51slojzn)

[3 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 13](#_oucs04x3xm6n)

[4 ВИБІР ТЕХНОЛОГІЙ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ ЗАСТОСУНКА 15](#_bvd79832odg7)

[5 ПРОГРАМНА МОДЕЛЬ ЗАСТОСУНКА 16](#_j3hcn2nw4tib)

[6 БЕЗПЕКА ІНФОРМАЦІЙНОЇ СИСТЕМИ 19](#_b6pidhcwwncz)

[7 СТВОРЕННЯ БАЗИ ДАНИХ 22](#_l3py633a1z8b)

[8 ЗАПИТИ ДО БАЗИ ДАНИХ ДЛЯ ВИРІШЕННЯ](#_jdwrrkbfmkje)

[ПОСТАВЛЕНИХ ЗАДАЧ 26](#_jdwrrkbfmkje)

[9 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ 32](#_q5kdgum3z1x4)

[9.1 Реалізація веб-сервера 32](#_dok5dnll2eix)

[9.2 Опис реалізованого REST API 33](#_ao8holyelbfu)

[9.3 JSON-структури даних 33](#_1ipmfqd8lx3w)

[9.4 Міграція бази даних 35](#_cvkqvqhri0mh)

[9.5 Програмна реалізація роботи з БД 35](#_5wd879xihs0s)

[10 ІНСТРУКЦІЯ КОРИСТУВАЧА 36](#_v6hmza3ri3v)

[ВИСНОВКИ 46](#_qucc35kkg6cp)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 48](#_pmqv1bbdnfx3)

[ДОДАТОК А. КОД ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ЗАСТОСУНКУ 49](#_pmc8avzaivp1)

[ДОДАТОК Б. ФРОНТЕНД ЧАСТИНА ВЕБ-ЫНТЕРФЕЙСУ 75](#_qfqcyeub1b70)

# ВСТУП

Сама тема “Інформаційна система для співробітників фітнес-клубу” є актуальною у зв’язку з необхідністю автоматизації та оптимізації роботи комерційних підприємств, що надають послуги у сфері здоров’я та спорту. Фітнес-індустрія активно зростає, і важливо забезпечити ефективне управління клієнтами, персоналом, розкладом тренувань та станом обладнання. Саме тому створення інформаційної системи з чітким розмежуванням ролей та функцій є надзвичайно актуальним завданням.

Мотив до написання саме такого продукту виник з мети створити гнучке та просте у використанні рішення, яке забезпечує ефективну взаємодію між адміністрацією клубу, реєстраторами та тренерами, спрощує повсякденні операції та зменшує кількість помилок, пов’язаних з людським фактором.

Мета курсового проєкту: Розробити веб-застосунок для внутрішнього використання у фітнес-клубах, який дозволяє адміністратору керувати персоналом і залами, реєстратору — вести облік клієнтів, абонементів та тренувань, а тренеру — переглядати свій особистий розклад.

Об’єкти дослідження: інформаційна система для фітнес-клубу (результат проєкту), мова запитів SQL, діалект PostgreSQL, мова програмування Python, фреймворк Flask, бібліотека SQLAlchemy для ORM, HTML, CSS, JavaScript для фронтенду.

Створення застосунку поділялося на наступні етапи:

1. визначення цільових ролей користувачів та функціоналу кожної з них;
2. побудова концептуальної моделі бази даних та зв’язків між сутностями;
3. проєктування структури інтерфейсу та логіки доступу до функцій;
4. реалізація серверної логіки на Flask з використанням SQLAlchemy;
5. створення динамічного клієнтського інтерфейсу з HTML/CSS/JS;
6. тестування роботи системи та підключення до PostgreSQL;
7. підготовка демонстраційної бази даних та початкових користувачів.

Результатом роботи є повноцінний веб-застосунок із ролями, авторизацією, CRUD-операціями, фільтрацією, формами, модальними вікнами та розширеним функціоналом для кожної категорії працівників фітнес-клубу.

# 

# 1 ПОСТАНОВКА ЗАДАЧІ

В системі визначено наступні класи користувачів:

1) Реєстратор: - створює та редагує клієнтів, абонементи, заняття з особистим тренером, та редагує інформацію про стан обладнання.

2) Адміністратор - адмініструє систему, створює, редагує та видаляє тренерів, зали, обладнання. Має можливість видалити клієнта та абонемент.

3) Тренер - може переглянути усі свої заняття.

Задачі описано у вигляді таблиці: див. табл. 1

Таблиця 1 - Таблиця задач по користувачам

| **№** | **Задача** | **Вхідні дані** | **Вихідні дані** |
| --- | --- | --- | --- |
| Адміністратор | | | |
| А3 | Видалити клієнта | id | Відсутні |
| А4 | Створити тренера | id, ФІО, спеціалізація, телефон, основний зал | Новий тренер |
| А5 | Редагувати тренера | ФІО, спеціалізація, телефон, основний зал | Оновлена інформація про тренера |
| А6 | Видалити тренера | id | Відсутні |
| А9 | Видалити абонемент | id | Відсутні |
| А13 | Створити обладнання | id, назва, стан | Нове обладнання |
| А14 | Редагувати обладнання | назва, стан | Оновлена інформація про обладнання |
| А15 | Видалити обладнання | id | Відсутні |
| А16 | Створити зал | id, адреса, ємність, обладнання | Новий зал |
| А17 | Редагувати зал | адреса, ємність, обладнання | Оновлена інформація про зал |
| А18 | Видалити зал | id | Відсутні |
| Тренер | | | |
| В1 | Переглянути свій розклад | id тренера | Тренування, у яких в якості [id тренера] вказаний id тренера |
| Реєстратор | | | |
| С1 | Створити клієнта | id, ФІО, телефон, id тренера, дата реєстрації, id абонемента | Новий клієнт |
| С2 | Редагувати клієнта | ФІО, телефон, id тренера, дата реєстрації, id абонемента | Оновлена інформація про клієнта |
| С3 | Створити абонемент | id, дата створення, тривалість, кількість індивідуальних занять | Новий абонемент |
| С4 | Редагувати абонемент | дата створення, тривалість, кількість індивідуальних занять | Оновлена інформація про абонемент |
| С5 | Створити тренування | id, назва, id тренера, id клієнта, id зала, дата та час початку, тривалість | Нове тренування |
| С6 | Редагувати тренування | назва, id тренера, id клієнта, id зала, дата та час початку, тривалість | Оновлена інформація про тренування |
| С7 | Видалити тренування | id | Відсутні |
| С8 | Редагувати стан обладнання | id, стан | Оновлена інформація про стан обладнання |

# 2 ІНФОРМАЦІЙНЕ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

Для виконання заданих умов, виявляємо наступні сутності:

Для реалізації інформаційної системи фітнес-клубу виявлено такі ключові сутності:

1. Клієнт — фізична особа, яка користується послугами клубу. Зберігаються основні персональні дані (ПІБ, телефон), дата реєстрації, прив’язка до персонального тренера, основного залу та поточного абонемента.
2. Тренер — співробітник клубу, який проводить індивідуальні заняття з клієнтами. Має ПІБ, номер телефону, спеціалізацію та закріплений основний зал. Кожен тренер може бути відповідальним за проведення тренувань.
3. Абонемент — контракт між клієнтом та клубом, що визначає термін дії та кількість дозволених індивідуальних занять. Абонементи належать конкретному клієнту і можуть використовуватись для створення тренувань.
4. Тренування — подія, яка фіксує заплановану активність клієнта з конкретним тренером у визначеному залі. Має дату, час початку, тривалість, а також пов’язаний абонемент, на підставі якого воно було організоване.
5. Зал — фізичне приміщення, де проводяться тренування. Має унікальну адресу, ємність, та список доступного обладнання. Один зал може бути основним для кількох тренерів та клієнтів.
6. Обладнання — інвентар, який використовується під час тренувань. Кожна одиниця обладнання закріплена за конкретним залом, має назву та індикатор стану (1–5).

Формалізація звʼязків:

1. Зв’язок між Тренерами та Клієнтами  
   Опис: На одному тренері можуть працювати багато клієнтів, а кожен клієнт прив’язаний до лише одного тренера.  
   Формалізація: В таблиці Client (Клієнт) атрибут trener\_id виконує роль зовнішнього ключа, який посилається на id в таблиці Trener (Тренер). Це створює зв’язок «один-до-багатьох»: один тренер → багато клієнтів.
2. Зв’язок між Клієнтами та Абонементами  
   Опис: Один клієнт може мати більше одного абонементу. Значення абонементів зберігаються у відповідному полі клієнта.  
   Формалізація: В таблиці Client (Клієнт) атрибут sub\_id дозволяє зберігати множину значень, кожне з яких є посиланням на id у таблиці Subscription (Абонемент). Це забезпечує зв’язок «один-до-багатьох», адже один клієнт може отримати кілька абонементів.
3. Зв’язок між Абонементами та Тренуваннями  
   Опис: Один абонемент може бути використаний для проведення численних тренувань. При цьому кожне тренування вказує на конкретний абонемент, який використовується клієнтом.  
   Формалізація: В таблиці Training (Тренування) атрибут sub\_id є зовнішнім ключем, що посилається на id таблиці Subscription (Абонемент). Важлива умова: значення sub\_id в тренуванні має співпадати з одним із численних значень sub\_id клієнта, що гарантує узгодженість даних.
4. Зв’язок між Клієнтами та Тренуваннями  
   Опис: Один клієнт може відвідувати багато тренувань, а кожне тренування належить конкретно до одного клієнта.  
   Формалізація: Атрибут client\_id в таблиці Training (Тренування) використовується як зовнішній ключ, який посилається на id таблиці Client (Клієнт).
5. Зв’язок між Тренерами та Тренуваннями  
   Опис: Один тренер проводить безліч тренувань, проте кожне тренування проводиться лише одним тренером.  
   Формалізація: Атрибут trener\_id в таблиці Training (Тренування) служить зовнішнім ключем, який посилається на id таблиці Trener (Тренер).
6. Зв’язок між Тренуваннями та Залом  
   Опис: Одне тренування проводиться в одному залі, а один зал може приймати багато тренувань.  
   Формалізація: Атрибут room\_id в таблиці Training (Тренування) використовується як зовнішній ключ, який посилається на id таблиці Room (Зал).
7. Зв’язок між Залом та Обладнанням  
   Опис: Один зал може бути оснащений одним або декількома елементами обладнання, а кожне обладнання прив’язане до конкретного залу.  
   Формалізація: Атрибут equipments\_id в таблиці Room (Зал) використовується як зовнішній ключ, що посилається на id таблиці Equipment (Обладнання). Допускається зберігання множини значень для представлення всіх наявних обладнань у залі.
8. Зв’язок між Тренером та Залом  
   Опис: Тренер може мати лише один основний зал, а кожен зал може бути основним для багатьох тренерів.  
   Формалізація: Атрибут main\_room\_id в таблиці Trener (Тренер) використовується як зовнішній ключ, що посилається на id таблиці Room (Зал).

Враховуючи ці сутності, та їх логічні зв’язки, можна сформулювати наступну ERD діаграму (рис. 2):

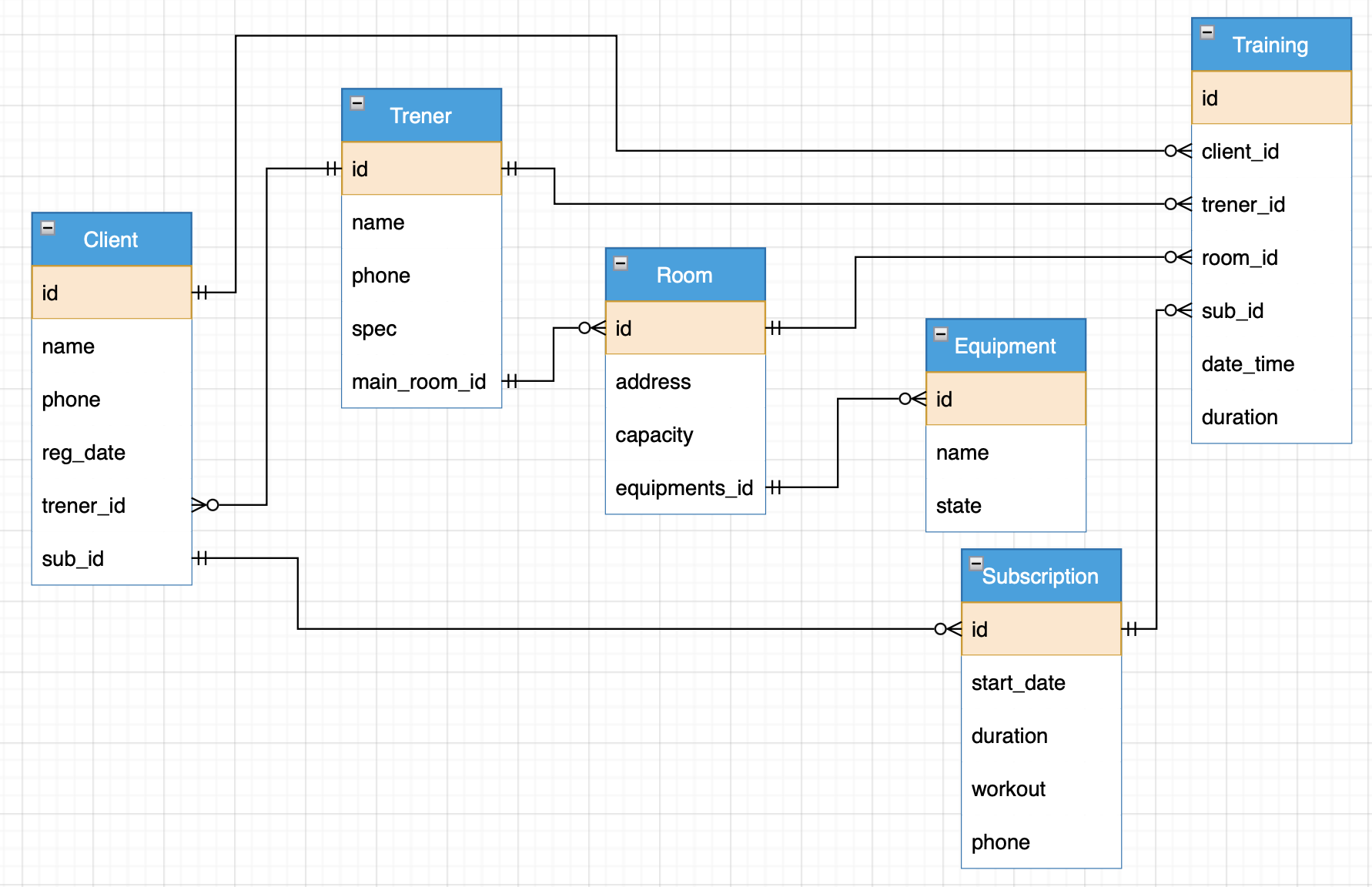


Рисунок 2 - ERD-діаграма для предметної області «Інформаційна система для співробітників фітнес-клубу»

Вона складається з наступних атрибутів (табл. 2):

Таблиця 2 - Опис атрибутів сутностей

| Імʼя атрибута | Призначення атрибута | Обмеження |
| --- | --- | --- |
| Client (Клієнт) | | |
| id | Ідентифікатор клієнта | Первинний ключ |
| name | ФІО клієнта | Не порожнє |
| phone | Номер телефону клієнта | Не порожнє;  12 символів;  Цифри |
| reg\_date | Дата реєстрації клієнта | Не порожнє;  Формат дати; |
| trener\_id | Ідентифікатор особистого тренера | Зовнішній ключ для зв'язку з сутністю Trener (id) |
| sub\_id | Ідентифікатор абонемента | Зовнішній ключ для зв'язку з сутністю Subscription (id);  Може містити декілька значень |
| Trener (Тренер) | | |
| id | Ідентифікатор тренера | Первинний ключ |
| name | ФІО тренера | Не порожнє |
| phone | Номер телефону тренера | Не порожнє;  12 символів;  Цифри |
| spec | Спеціалізація тренера | Одне або декілька значень з множини (“Універсал”, “Силові”, “Кросфіт”, “Кардіо”, “Йога”, “Розтяжка”, “Пілатес”);  Не порожнє |
| main\_room\_id | Основний зал тренера | Зовнішній ключ для зв'язку з сутністю Room (id);  Не порожнє; |
| Subscription (Абонемент) | | |
| id | Ідентифікатор абонемента | Первинний ключ |
| start\_date | Дата створення абонемента | Не порожнє;  Формат дати; |
| duration | Тривалість абонементу у днях | Не порожнє;  Число;  >= 0 |
| workout | Кількість тренувань з тренером | Не порожнє;  Число;  >= 0 |
| Training (Тренування) | | |
| id | Ідентифікатор тренування | Первинний ключ |
| client\_id | Ідентифікатор клієнта | Зовнішній ключ для зв'язку з сутністю Client (id);  Не порожнє |
| sub\_id | Ідентифікатор абонемента | Зовнішній ключ для зв'язку з сутністю Subscription (id);  Не порожнє;  Має бути у множині значень sub\_id сутності Client |
| trener\_id | Ідентифікатор тренера | Зовнішній ключ для зв'язку з сутністю Trener (id);  Не порожнє |
| room\_id | Ідентифікатор зала | Зовнішній ключ для зв'язку з сутністю Room (id);  Не порожнє |
| date\_time | Дата та час початку тренування | Не порожнє;  Формат дати з часом; |
| duration | Тривалість тренування (хв) | Не порожнє;  Число;  >= 30 |
| Equipment (Обладнання) | | |
| id | Ідентифікатор обладнання | Первинний ключ |
| name | Назва обладнання | Не порожнє; |
| state | Стан обладнання | Не порожнє;  Число;  Від 1 до 5 |
| Room (Зал) | | |
| id | Ідентифікатор | Первинний ключ |
| address | Адреса зали | Не порожнє |
| capacity | Ємність залу | Не порожнє;  Число; > 0 |
| equipments\_id | Обладнання залу | Зовнішній ключ Equipment(id), одне або декілька значень з множини всіх Equipment |

# 

1. Клієнт може бути повністю видалений із системи. При цьому всі пов’язані з ним абонементи та тренування також автоматично видаляються, щоб забезпечити цілісність бази даних.
2. Тренування може бути створене лише для дійсного абонемента, який належить обраному клієнту. Якщо обраний абонемент не співпадає з клієнтом, система забороняє збереження.
3. Тренер не може бути призначений клієнту, якщо його основний зал не співпадає з основним залом клієнта. Вибір тренера обмежується лише тими, хто закріплений за вказаним залом.
4. Абонемент не може бути створений без вказання клієнта, і створення абонемента автоматично прив’язує його до відповідного клієнта.
5. Тренування не може бути створене, якщо його дата виходить за межі строку дії вибраного абонемента. Перевірка виконується при створенні чи редагуванні.
6. Доступ до створення, редагування та видалення сутностей обмежено відповідно до ролі користувача:

* Адміністратор: керує тренерами, залами, обладнанням, має доступ до видалення інформації про тренерів, залі, абонементи, клієнтів, обладнання;
* Реєстратор: керує клієнтами, абонементами, тренуваннями;
* Тренер: має лише доступ до перегляду власного розкладу.

# 3 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Архітектура застосунку інформаційної системи для фітнес-клубу поділяється на три шари:

1. Шар даних — відповідає за зберігання даних та забезпечує їхню цілісність і логічні зв’язки між сутностями. Цей шар представлено базою даних PostgreSQL, яка містить таблиці, що відповідають описаним сутностям (Client, Trener, Subscription, Training, Room, Equipment).
2. Шар доступу до даних та бізнес-логіки — реалізує обробку запитів користувачів, забезпечує CRUD-операції з урахуванням бізнес-обмежень і правил системи. Цей шар реалізований за допомогою серверної частини застосунку на основі Python з використанням фреймворку Flask та бібліотеки SQLAlchemy для зручного керування об'єктами бази даних. API системи побудовано за принципом REST (Representational State Transfer), що забезпечує простоту і масштабованість взаємодії з клієнтською частиною застосунку через стандартні HTTP-запити (GET, POST тощо).
3. Шар представлення — графічний інтерфейс користувача, реалізований за допомогою технологій HTML, CSS та JavaScript. Використовується підхід серверного рендерингу (server-side rendering), який доповнюється динамічними елементами на JavaScript для покращення зручності взаємодії користувачів із застосунком.

Для реалізації інформаційної системи обрано монолітну архітектуру. Цей підхід дозволяє розробити єдиний сервіс, який містить усі необхідні функції: авторизацію користувачів, роботу з базою даних, обробку запитів та генерацію інтерфейсу. Монолітна архітектура забезпечує простоту розгортання, легкість підтримки та є оптимальним рішенням для невеликих і середніх застосунків.

Веб-застосунок надає наступний функціонал відповідно до ролей:

1. Адміністратор: створення, редагування, видалення тренерів, залів та обладнання, керування клієнтами та абонементами;
2. Реєстратор: ведення клієнтів, створення та керування абонементами і тренуваннями;
3. Тренер: перегляд власного розкладу тренувань.

# 4 ВИБІР ТЕХНОЛОГІЙ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ ЗАСТОСУНКА

Для створення бази даних використано PostgreSQL — надійну та високопродуктивну об’єктно-реляційну СУБД, яка ефективно обробляє складні запити та забезпечує підтримку транзакцій.

Серверну частину розроблено на мові програмування Python, яка популярна у веб-розробці завдяки своїй простоті та великій кількості готових бібліотек. Обрано фреймворк Flask за його легкість у використанні, мінімалізм і високу швидкість розробки. Для взаємодії з базою даних застосовано ORM SQLAlchemy, що дозволяє працювати з даними у вигляді об’єктів Python, спрощуючи код і забезпечуючи легкість підтримки.

Frontend-частина застосунку реалізована за допомогою класичних веб-технологій HTML, CSS і JavaScript. Для інтерфейсу застосовано принцип server-side рендерингу з динамічним оновленням окремих елементів сторінок за допомогою JavaScript, що дозволяє створити швидкий і зручний інтерфейс без значних ускладнень та додаткових витрат на навчання персоналу.

Для управління проєктом бази даних використано інструмент pgAdmin, що надає зручний графічний інтерфейс для створення, редагування та управління структурами даних і користувачами.

Застосування зазначених технологій забезпечило високу продуктивність, простоту розробки та подальшого супроводу інформаційної системи.

# 5 ПРОГРАМНА МОДЕЛЬ ЗАСТОСУНКА

Проєкт інформаційної системи для фітнес-клубу складається з наступних компонентів:

1) стартова точка програми, файл `app.py`, що виконує повний цикл життя програми: ініціалізацію застосунку Flask, конфігурацію підключення до бази даних, налаштування маршрутів, обробку запитів та запуск веб-сервера;

2) визначення моделей даних (класи SQLAlchemy), які поширюються між компонентами: `User`, `Client`, `Trener`, `Subscription`, `Training`, `Room`, `Equipment`;

3) конфігурації застосунку, що зберігаються в змінних середовища або безпосередньо в коді (`app.config`) – зокрема параметри підключення до БД PostgreSQL, секретні ключі для сесій та авторизації;

4) вхідні точки API та веб-застосунку (маршрути Flask), які визначені за допомогою декораторів `@app.route` та обробляють HTTP-запити (GET, POST тощо);

5) контролери обробки запитів до API, що реалізовані у вигляді функцій-маршрутів Flask і відповідають за бізнес-логіку застосунку, авторизацію користувачів та CRUD-операції над сутностями;

6) представлення (рендерери) веб-сторінок, що виконуються через функції `render\_template` Flask, які використовують шаблонізатор Jinja2 для генерації динамічного HTML-коду на основі переданих даних;

7) шар доступу до бази даних у вигляді незалежного від реалізації інтерфейсу, що надає бібліотека SQLAlchemy, дозволяючи легко змінювати реалізацію взаємодії з БД;

8) реалізація доступу до бази даних через СУБД PostgreSQL з використанням SQLAlchemy ORM, що дозволяє абстрагуватися від SQL-запитів та керувати об’єктами бази даних як звичайними об’єктами Python;

9) модулі-утиліти та допоміжні функції, такі як `generate\_password\_hash`, `check\_password\_hash` (бібліотека Werkzeug) для роботи з паролями, `flash` для передачі повідомлень користувачу, та стандартні бібліотеки Python для роботи з датами, часом та валідацією даних.

Структура застосунку організована у вигляді наступних директорій та файлів:

- `/app.py` — головний файл застосунку Flask, містить основну конфігурацію, маршрутизацію та логіку застосунку.

- `/models.py` — містить визначення моделей даних SQLAlchemy.

- `/create\_users.py` — допоміжний скрипт для початкового створення користувачів з різними ролями.

- `/templates/` — папка, що містить HTML-шаблони для веб-сторінок:

- `login.html` – сторінка авторизації;

- `admin.html` – панель адміністратора;

- `registrar.html` – панель реєстратора;

- `trener.html` – панель тренера.

- `/static/` — папка, що містить статичні ресурси застосунку (CSS, JavaScript файли, зображення тощо).

- `/utils/` — додаткові утилітарні функції для спрощення роботи з датами, валідацією та рендерингом шаблонів.

- `/login` – сторінка авторизації;

- `/admin` – головна сторінка адміністратора, звідки доступні функції керування залами, тренерами, обладнанням, клієнтами, абонементами та тренуваннями;

- `/registrar` – головна сторінка реєстратора, де здійснюється створення та редагування клієнтів, абонементів і тренувань;

- `/trener` – головна сторінка тренера, яка відображає його розклад тренувань.

Навігація між сторінками реалізована за допомогою стандартних переходів через веб-інтерфейс із використанням посилань (`<a>`-тегів) та форм (`<form>`-тегів), які обробляються відповідними маршрутами Flask.

Карта переходів по веб-сторінкам сайту (Див. рисунок 5):

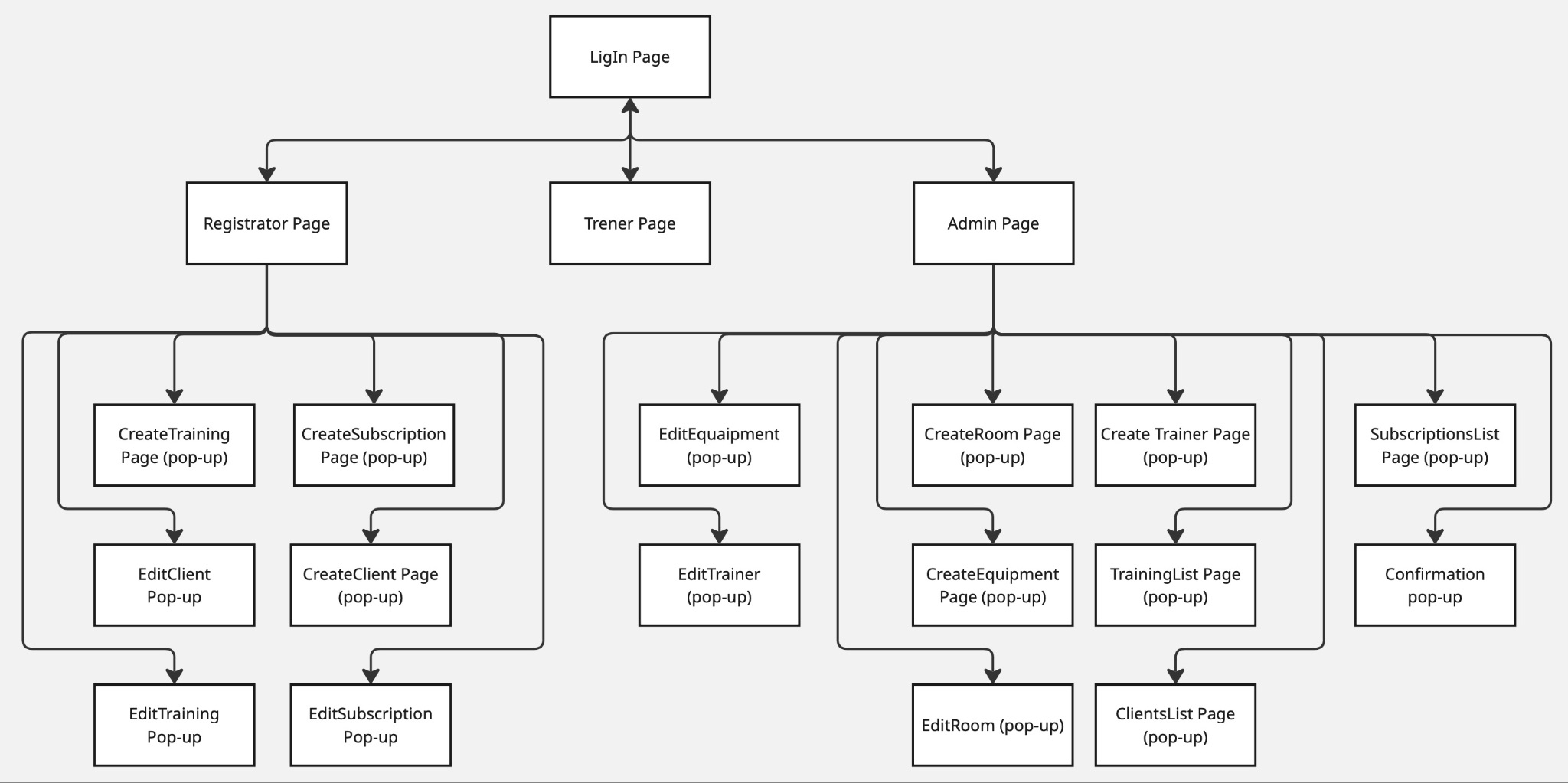


Рисунок 5 - Карта переходів по сторінкам сайту

# 6 БЕЗПЕКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

Інформаційна система для співробітників фітнес-клубу реалізує чітке розмежування доступу до функціоналу та даних на основі ролей користувачів. Авторизація виконується за допомогою облікових записів, які попередньо створюються адміністратором системи із захешованими паролями, використовуючи бібліотеку Werkzeug (метод generate\_password\_hash та check\_password\_hash). Всі користувачі зберігаються у таблиці User з чітко вказаною роллю: admin (адміністратор), registrar (реєстратор), trener (тренер).

Для підвищення безпеки та запобігання вразливостям, у системі реалізовані наступні механізми:

1. Валідація вхідних даних на стороні сервера для запобігання SQL-ін’єкціям та інших типів атак.
2. Використання ORM SQLAlchemy, що додатково захищає систему від SQL-ін’єкцій завдяки автоматичному екрануванню всіх параметрів.
3. Вся авторизація базується на сесіях Flask, ключі до яких захищені секретним ключем додатку (app.secret\_key).

Визначено три глобальні ролі користувачів системи та їхні привілеї:

1. Адміністратор – має повні права керування користувачами, тренерами, залами, обладнанням, абонементами та клієнтами, включаючи створення, редагування та видалення.
2. Реєстратор – може керувати клієнтами, абонементами та тренуваннями (створення, редагування, видалення).
3. Тренер – має доступ тільки для перегляду власного розкладу тренувань.

Глобальна таблиця прав доступу для інформаційної системи наведена в табл. 6.1:

Таблиця 6.1 - Права доступу для глобальних ролей

| Таблиця | Тренер | Реєстратор | Адміністратор |
| --- | --- | --- | --- |
| Таблиці | | | |
| users | R | R | CRUD |
| trener | R | R | CRUD |
| client | R | CRU | CRUD |
| subscription | R | CRU | CRUD |
| training | R | CRU | CRUD |
| room | R | R | CRUD |
| equipment | R | RU | CRUD |

В таблиці 6.1 позначено:

* **C** – Create, право на створення;
* **R** – Read, право на читання;
* **U** – Update, право на зміну;
* **D** – Delete, право на видалення.

Для розмежування доступу до даних у базі PostgreSQL створено ролі відповідно до глобальних ролей системи (лістинг 6.1):

create role administrator;

create role registrar;

create role trener;

Лістинг 6.1 - Створення ролей

Після створення таблиць встановлено правила розмежування привілеїв до таблиць за ролями (лістинг 6.2):

-- Права адміністратора

grant all privileges on table users to administrator;

grant all privileges on table trener to administrator;

grant all privileges on table client to administrator;

grant all privileges on table subscription to administrator;

grant all privileges on table training to administrator;

grant all privileges on table room to administrator;

grant all privileges on table equipment to administrator;

-- Права реєстратора

grant select on table users to registrar;

grant select on table trener to registrar;

grant all privileges on table client to registrar;

grant all privileges on table subscription to registrar;

grant all privileges on table training to registrar;

grant select on table room to registrar;

grant update (state) on table equipment to registrar;

-- Права тренера

grant select on table users to trener;

grant select on table trener to trener;

grant select on table client to trener;

grant select on table subscription to trener;

grant select on table training to trener;

grant select on table room to trener;

grant select on table equipment to trener;

Лістинг 6.2 - Надання привілей до таблиць по ролям

Додаткове розмежування доступу на рівні бізнес-логіки застосунку гарантує, що кожен тренер може переглядати лише свої власні тренування. Такий підхід забезпечує високий рівень інформаційної безпеки та конфіденційності даних у системі.

# 

# 7 СТВОРЕННЯ БАЗИ ДАНИХ

Наведено структуру бази даних для інформаційної системи фітнес-клубу — таблиці, їхні ключі, обмеження, а також тригери, які забезпечують бізнес-логіку на рівні СУБД.

У схемі створено такі основні таблиці:

1. users — користувачі системи (адміністратори, реєстратори, тренери)
2. rooms — зали фітнес-клубу
3. equipment — обладнання, закріплене за залами
4. trainers — тренери, закріплені за основними залами
5. clients — клієнти клубу, з прив’язкою до тренера, залу та активного абонемента
6. subscriptions — абонементи клієнтів (дата початку, тривалість, ліміт індивідуальних тренувань)
7. trainings — заплановані тренування (дата, час, тривалість, клієнт, тренер, зал, абонемент)

Всі первинні ключі — типу serial, зовнішні ключі — з ON DELETE CASCADE або SET NULL відповідно до правил (Лістинг 7.1).

create table if not exists clients (

id serial primary key,

name varchar(100) not null,

phone varchar(12) not null check (phone ~ '^\d{11,12}$'),

reg\_date date not null default current\_date,

main\_room\_id integer not null

references rooms(id) on delete restrict deferrable,

trainer\_id integer null

references trainers(id) on delete set null deferrable,

subscription\_id integer null

references subscriptions(id) on delete set null deferrable

);

Лістинг 7.1 - Створені в схемі БД таблиці сутностей

rooms(id) — PK

equipment(room\_id) → rooms(id) ON DELETE SET NULL

trainers(main\_room\_id) → rooms(id) ON DELETE RESTRICT

subscriptions(client\_id) → clients(id) ON DELETE CASCADE

trainings має FK на всі сутності — при видаленні клієнта/тренера/абонемента/зали відповідні тренування видаляються.

Усі varchar-поля з телефоном мають CHECK-констрейнт для цифр.

Для пришвидшення пошуку створено індекси на зовнішні ключі (Лістинг 7.2).

create index idx\_trainings\_client on trainings(client\_id);

create index idx\_trainings\_trainer on trainings(trainer\_id);

create index idx\_subscriptions\_client on subscriptions(client\_id);

Лістинг 7.2 - Додаткові обмеження та індекси

Перевірка та оновлення ліміту занять при створенні тренування (Лістинг 7.3).

-- Функція: перед додаванням training перевіряє, що

-- підписка ще дійсна та є вільні заняття

create or replace function check\_and\_decrement\_subscription() returns trigger as $$

declare

exp\_date date;

begin

select start\_date + (duration - 1) into exp\_date

from subscriptions

where id = new.subscription\_id;

if current\_date > exp\_date then

raise exception 'Subscription expired';

end if;

if (select workout from subscriptions where id=new.subscription\_id) <= 0 then

raise exception 'No remaining trainings on subscription';

end if;

-- зменшуємо лічильник занять

update subscriptions

set workout = workout - 1

where id = new.subscription\_id;

return new;

end;

$$ language plpgsql;

drop trigger if exists trg\_sub\_decrement on trainings;

create trigger trg\_sub\_decrement

before insert on trainings

for each row execute function check\_and\_decrement\_subscription();

Лістинг 7.3 - Перевірка та оновлення ліміту занять при створенні тренування

Заборона запису тренування поза основним залом клієнта/тренера (Лістинг 7.4).

-- Функція: перевіряє, що тренування відбувається в залі,

-- закріпленому за клієнтом і тренером

create or replace function validate\_training\_room() returns trigger as $$

declare

c\_room int; t\_room int;

begin

select main\_room\_id into c\_room from clients where id=new.client\_id;

select main\_room\_id into t\_room from trainers where id=new.trainer\_id;

if new.room\_id <> c\_room or new.room\_id <> t\_room then

raise exception 'Training room mismatch with client or trainer main room';

end if;

return new;

end;

$$ language plpgsql;

drop trigger if exists trg\_room\_validate on trainings;

create trigger trg\_room\_validate

before insert or update on trainings

for each row execute function validate\_training\_room();

Лістинг 7.4 - Заборона запису тренування поза основним залом клієнта/тренера

У додатку Б наведено повний набір DDL-скриптів (створення типів, таблиць, індексів, тригерів), які необхідно виконати для розгортання бази даних фітнес-системи.

# 8 ЗАПИТИ ДО БАЗИ ДАНИХ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

Нижче наведено всі основні SQL-запити, які виконує сервер для реалізації CRUD-функцій вашої інформаційної системи фітнес-клубу. Параметри позначені як $1, $2 тощо відповідно до позиційних аргументів.

1) Авторизація користувача (Лістинг 8.1)

SELECT id, password\_hash, role

FROM "user"

WHERE username = $1;

Лістинг 8.1 - Перевірка логіна

Створення залу (Лістинг 8.2).

INSERT INTO room(address, capacity)

VALUES($1, $2)

RETURNING id;

Лістинг 8.2 - INSERT залу

Редагування залу (Лістинг 8.3).

UPDATE room

SET address = COALESCE($2, address),

capacity = COALESCE($3, capacity)

WHERE id = $1;

Лістинг 8.3 - UPDATE залу

Видалення залу (Лістинг 8.4).

UPDATE equipment

SET room\_id = NULL

WHERE room\_id = $1;

DELETE FROM training

WHERE room\_id = $1;

DELETE FROM room

WHERE id = $1;

Лістинг 8.4 - DELETE залу із попереднім очищенням

Створення тренера (Лістинг 8.5).

INSERT INTO trener(name, phone, spec, main\_room\_id)

VALUES($1, $2, $3, $4)

RETURNING id;

Лістинг 8.5 - INSERT тренера

Редагування тренера (Лістинг 8.6).

UPDATE trener

SET name = COALESCE($2, name),

phone = COALESCE($3, phone),

spec = COALESCE($4, spec),

main\_room\_id = COALESCE($5, main\_room\_id)

WHERE id = $1;

Лістинг 8.6 - UPDATE тренера

Видалення тренера (Лістинг 8.7).

UPDATE client

SET trainer\_id = NULL

WHERE trainer\_id = $1;

-- Повернути заняття абонементам

UPDATE subscription AS s

SET workout = workout + sub.count

FROM (

SELECT sub\_id AS id, COUNT(\*) AS count

FROM training

WHERE trainer\_id = $1

GROUP BY sub\_id

) AS sub

WHERE s.id = sub.id;

DELETE FROM training

WHERE trainer\_id = $1;

DELETE FROM trener

WHERE id = $1;

Лістинг 8.7 - DELETE тренера із оновленням клієнтів та абонементів

Створення обладнання (Лістинг 8.8).

INSERT INTO equipment(name, state, room\_id)

VALUES($1, $2, $3)

RETURNING id;

Лістинг 8.8 - INSERT обладнання

Редагування обладнання (Лістинг 8.9).

UPDATE equipment

SET name = COALESCE($2, name),

state = COALESCE($3, state),

room\_id = COALESCE($4, room\_id)

WHERE id = $1;

Лістинг 8.9 - UPDATE обладнання

Видалення обладнання (Лістинг 8.10).

DELETE FROM equipment

WHERE id = $1;

Лістинг 8.10 - DELETE обладнання

Створення клієнта (Лістинг 8.11).

INSERT INTO client(name, phone, reg\_date, main\_room\_id, trainer\_id)

VALUES($1, $2, $3, $4, $5)

RETURNING id;

Лістинг 8.11 – INSERT клієнта

Редагування клієнта (Лістинг 8.12).

UPDATE client

SET name = COALESCE($2, name),

phone = COALESCE($3, phone),

reg\_date = COALESCE($4, reg\_date),

main\_room\_id = COALESCE($5, main\_room\_id),

trainer\_id = COALESCE($6, trainer\_id)

WHERE id = $1;

Лістинг 8.12 – UPDATE клієнта

Видалення клієнта (Лістинг 8.13).

UPDATE client

SET subscription\_id = NULL

WHERE id = $1;

DELETE FROM training

WHERE client\_id = $1;

DELETE FROM subscription

WHERE client\_id = $1;

DELETE FROM client

WHERE id = $1;

Лістинг 8.13 – DELETE клієнта з каскадним очищенням

Створення абонемента (Лістинг 8.14).

INSERT INTO subscription(start\_date, duration, workout, client\_id)

VALUES($1, $2, $3, $4)

RETURNING id;

Лістинг 8.14 – INSERT абонемента

Редагування абонемента (Лістинг 8.15).

UPDATE subscription

SET start\_date = COALESCE($2, start\_date),

duration = COALESCE($3, duration),

workout = COALESCE($4, workout)

WHERE id = $1;

Лістинг 8.15 – UPDATE абонемента

Видалення абонемента (Лістинг 8.16).

UPDATE client

SET subscription\_id = NULL

WHERE subscription\_id = $1;

DELETE FROM training

WHERE subscription\_id = $1;

DELETE FROM subscription

WHERE id = $1;

Лістинг 8.16 – DELETE абонемента з оновленням клієнта та тренувань

Створення тренування (Лістинг 8.17).

INSERT INTO training(client\_id, trainer\_id, subscription\_id, room\_id, date\_time, duration)

VALUES($1, $2, $3, $4, $5, $6)

RETURNING id;

Лістинг 8.17 – INSERT тренування

Редагування тренування (Лістинг 8.18).

UPDATE training

SET date\_time = COALESCE($2, date\_time),

duration = COALESCE($3, duration)

WHERE id = $1;

Лістинг 8.18 – UPDATE тренування

Видалення тренування (Лістинг 8.19).

UPDATE subscription AS s

SET workout = workout + 1

FROM training AS t

WHERE t.id = $1

AND s.id = t.subscription\_id;

DELETE FROM training

WHERE id = $1;

Лістинг 8.19 – DELETE тренування з поверненням ліміту абонементу

Усі ці запити виконуються через SQLAlchemy або безпосередньо за допомогою db.session.execute(...) у вашому Flask-додатку.

# 9 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

## 9.1 Реалізація веб-сервера

Вхідна точка Flask-додатку знаходиться в `app.py` у блоці `if \_\_name\_\_ == '\_\_main\_\_':` (Лістинг 9.1). На старті виконується:

1. Завантаження конфігурації (URI БД, секретний ключ)

2. Ініціалізація схеми (міграція через `db.create\_all()`)

3. Завантаження початкових користувачів (`create\_users.py`)

4. Налаштування режиму відлагодження, хосту та порту

5. Запуск вбудованого сервера

# app.py, кінець файлу

if \_\_name\_\_ == '\_\_main\_\_':

# 1) Завантаження налаштувань (може бути з файлу чи env)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgresql://...'

app.secret\_key = 'supersecretkey'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

# 2) Міграція схеми та ініціалізація користувачів

with app.app\_context():

db.create\_all()

# якщо хочете — тут можна викликати create\_users.py logic

# 3) Запуск сервера

app.run(host='0.0.0.0', port=5000, debug=True)

Лістинг 9.1 - Реалізація веб-сервера

## 9.2 Опис реалізованого REST API

Усі ендпоінти починаються з кореня `/` для веб-інтерфейсу та `/api/` для JSON-інтерфейсу (Таблиця 9.2).

| Метод | Шлях | Опис |
| --- | --- | --- |
| POST | /login | Авторизація (форма логіну) |
| GET | /logout | Вихід із системи |
| GET | /admin | Сторінка адміністратора |
| POST | /admin/room/create | Створити зал |
| POST | /admin/room/edit/<id> | Редагувати зал |
| GET | /admin/room/delete/<id> | Видалити зал |
| POST | /admin/trener/create | Створити тренера |
| POST | /admin/trener/edit/<id> | Редагувати тренера |
| GET | /admin/trener/delete/<id> | Видалити тренера |
| POST | /admin/equipment/create | Створити обладнання |
| POST | /admin/equipment/edit/<id> | Редагувати обладнання |
| GET | /admin/equipment/delete/<id> | Видалити обладнання |
| GET | /registrar | Сторінка реєстратора |
| POST | /registrar/client/create | Створити клієнта |
| POST | /registrar/client/edit/<id> | Редагувати клієнта |
| GET | /registrar/client/delete/<id> | Видалити клієнта |
| POST | /registrar/sub/create | Створити абонемент |
| POST | /registrar/sub/edit/<id> | Редагувати абонемент |
| GET | /registrar/sub/delete/<id> | Видалити абонемент |
| POST | /registrar/training/create | Створити тренування |
| POST | /registrar/training/edit/<id> | Редагувати тренування |
| GET | /registrar/training/delete/<id> | Видалити тренування |
| GET | /trener | Сторінка тренера |

Таблиця 9.2 - Опис реалізованого REST API

## 9.3 JSON-структури даних

Для AJAX-запитів або зовнішнього API в body передаються JSON-об’єкти (Лістингг 9.3)

// 1) Room

{

"address": "вул. Спортивна, 10",

"capacity": 30

}

// 2) Trener

{

"name": "Іван Іванов",

"phone": "380501234567",

"spec": "Кардіо",

"main\_room\_id": 1

}

// 3) Equipment

{

"name": "Тренажер X",

"state": 4,

"room\_id": 1

}

// 4) Client

{

"name": "Марія Петрівна",

"phone": "380671234567",

"reg\_date": "2025-04-01",

"main\_room\_id": 1,

"trener\_id": 2

}

// 5) Subscription

{

"start\_date": "2025-04-01",

"duration": 30,

"workout": 10,

"client\_id": 1

}

// 6) Training

{

"client\_id": 1,

"trener\_id": 2,

"sub\_id": 1,

"room\_id": 1,

"date\_time": "2025-04-23T15:00:00",

"duration": 60

}

Лістинг 9.3 - JSON-структури даних

## 9.4 Міграція бази даних

Міграція бази даних (Лістинг 9.4). У `app.py` це викликається перед першим `app.run()`.

"date\_time": "2025-04-23T15:00:00",

"duration": 60

}

Лістинг 9.4 - Міграція бази даних

## 9.5 Програмна реалізація роботи з БД

Усі операції з базою проходять через `db.session`. У прикладі CRUD-методів використовується шаблон (Лістинг 9.5.1)

"date\_time": "2025-04-23T15:00:00",

"duration": 60

}

Лістинг 9.5.1 - Програмна реалізація роботи з БД

Для читання використовується простий `Model.query.filter\_by(...)`. Приклад (Лістинг 9.5.2).

user = User.query.filter\_by(username=login).first()

if user and check\_password\_hash(user.password\_hash, pwd):

# …

Лістинг 9.5.2 - Приклад читання

# 10 ІНСТРУКЦІЯ КОРИСТУВАЧА

Робота користувача з веб-застосунком починається з меню логіна (Див. рисунок 10.1). Тут користувач вводить Логін та Пароль в залежності від своєї ролі: Адміністратор/Реєстратор/Тренер.

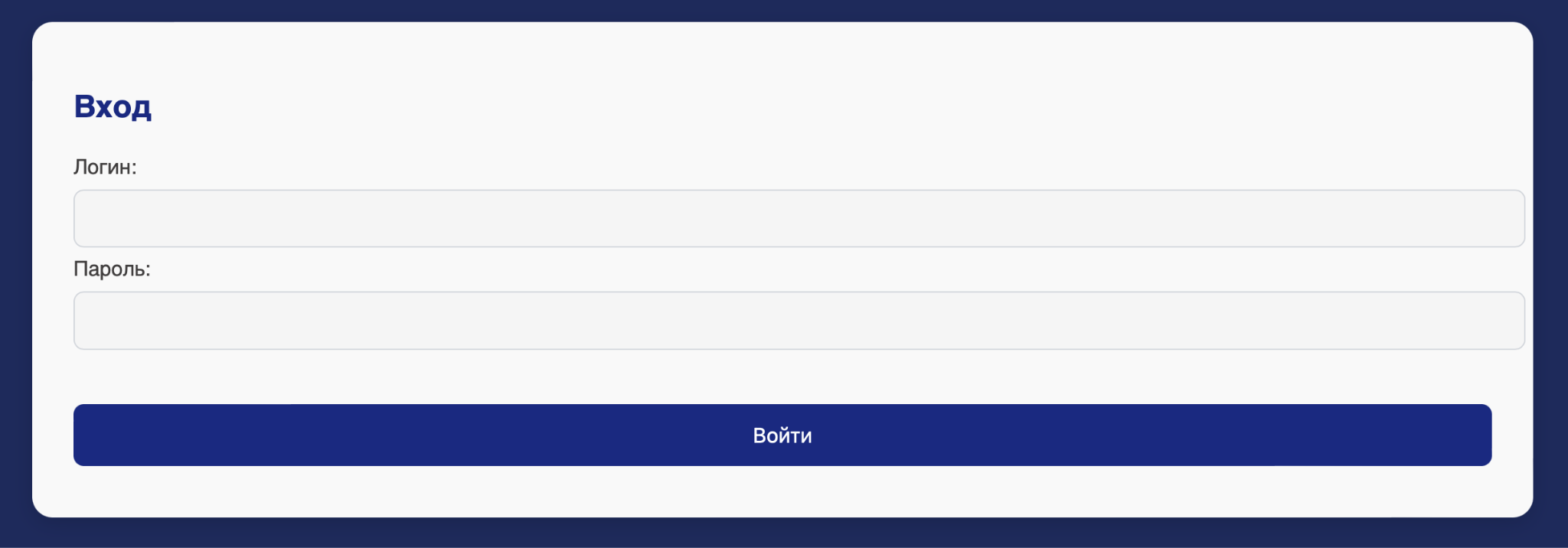


Рисунок 10.1 - Форма авторизації

Кнопка Войти перенаправить користувача на відповідну сторінку - панель керування, в залежності від введених данних. Ввод невірного паролю або логіну не дозволить користувачу потрапити на будь-яку сторінку.

Якщо користувач ввів данні для входу в панель керування адміністратора, то система відкриє сторінку панелі курування аддміністратора (Див. рисунок 10.2).

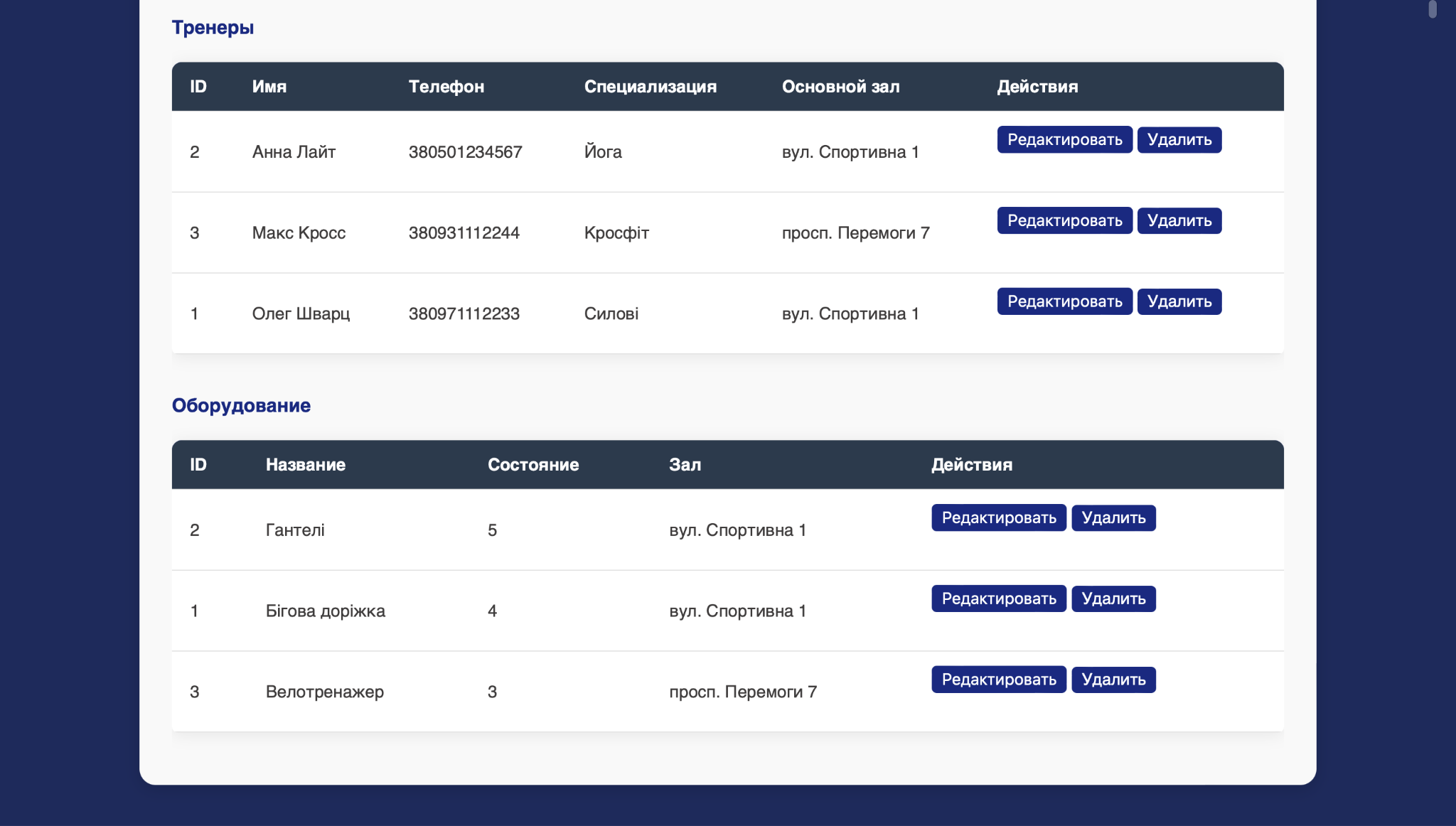
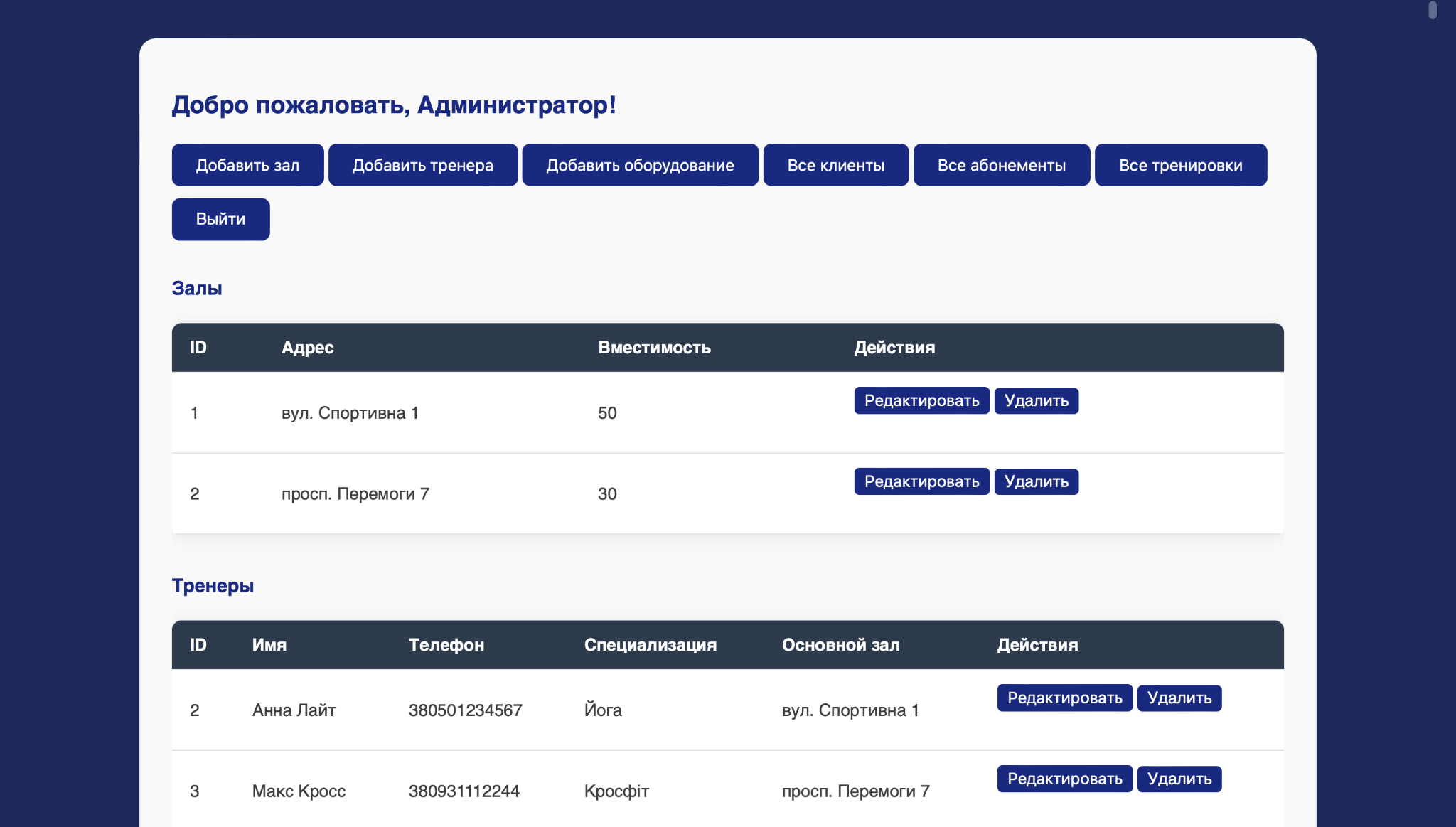


Рисунок 10.2 - Панель керування адміністратора

Адміністратор може відкрити Pop-up вікна для створення нового залу (Див. рисунок 10.3), створення нового тренера (Див. рисунок 10.4), створення нового обладнання (Див. рисунок 10.5).

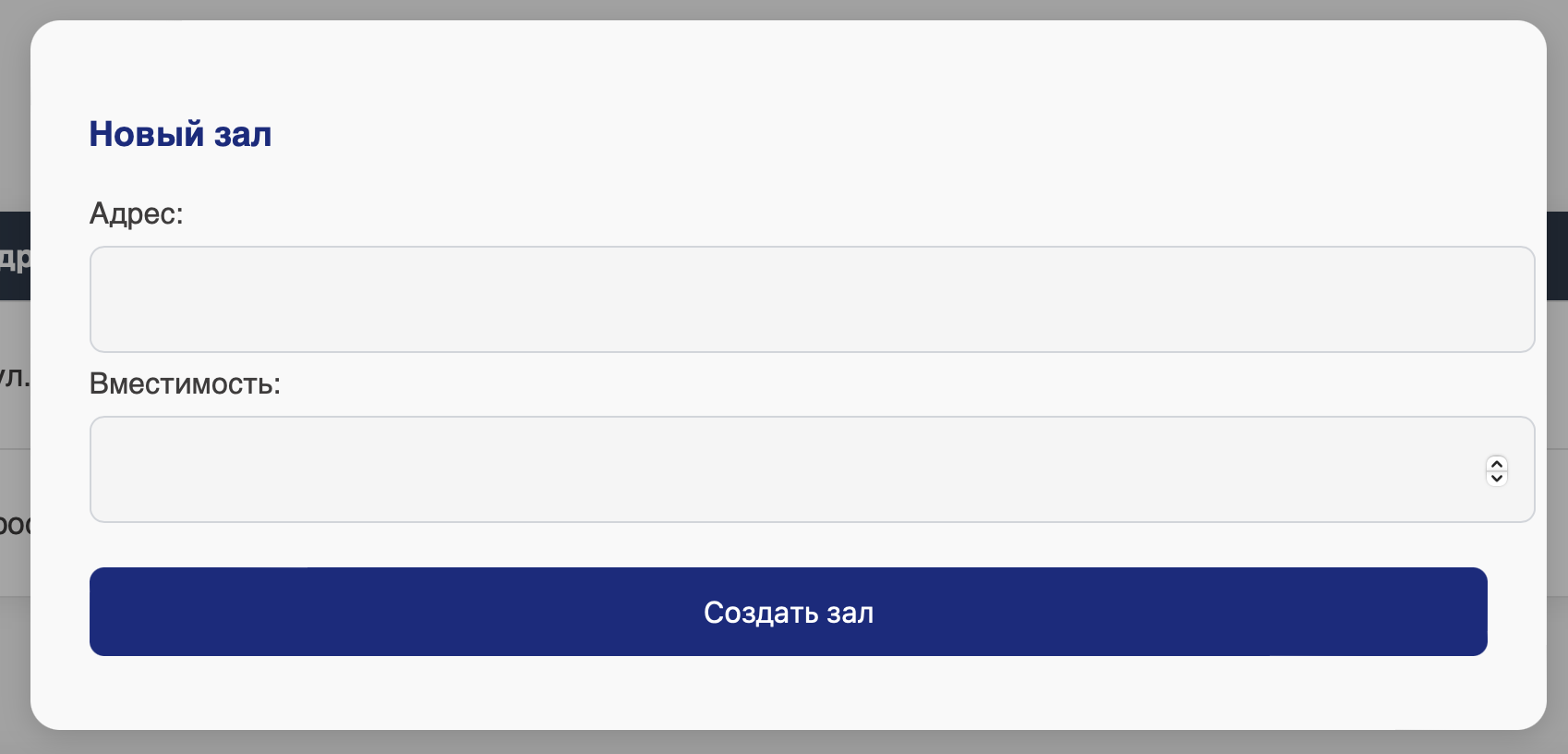


Рисунок 10.3 - Pop-up вікно для створення нового залу



Рисунок 10.4 - Pop-up вікно для створення нового тренера

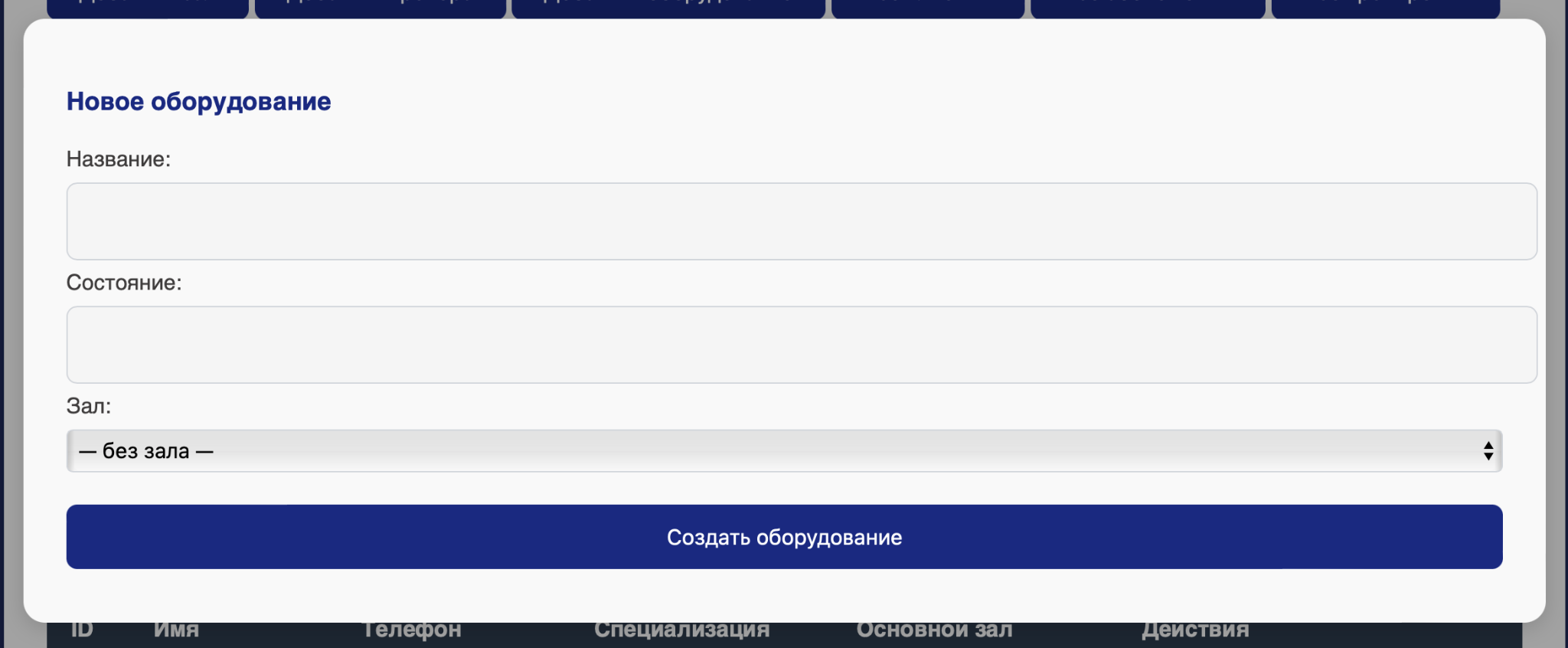


Рисунок 10.5 - Pop-up вікно для створення нового обладнання

Такод в адміністратора є можливість переглянути (відкрити Pop-up) меню із списками з можливістю видалення: Усі клієнти (Див. рисунок 10.6), Усі абонементи (Див. рисунок 10.7), та Усі тренування (див. рисунок 10.8).

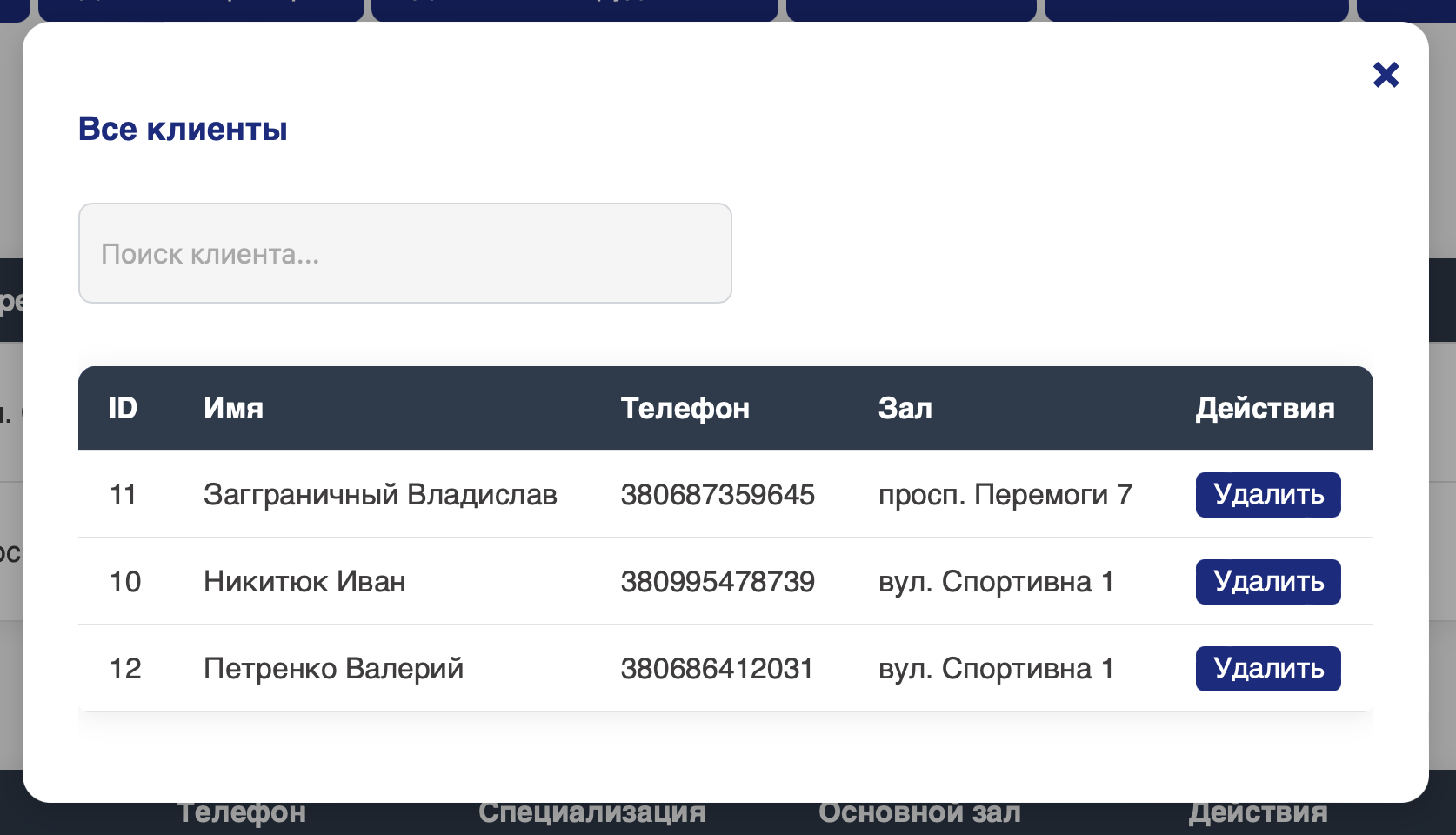


Рисунок 10.6 - Pop-up вікно Усі клієнти

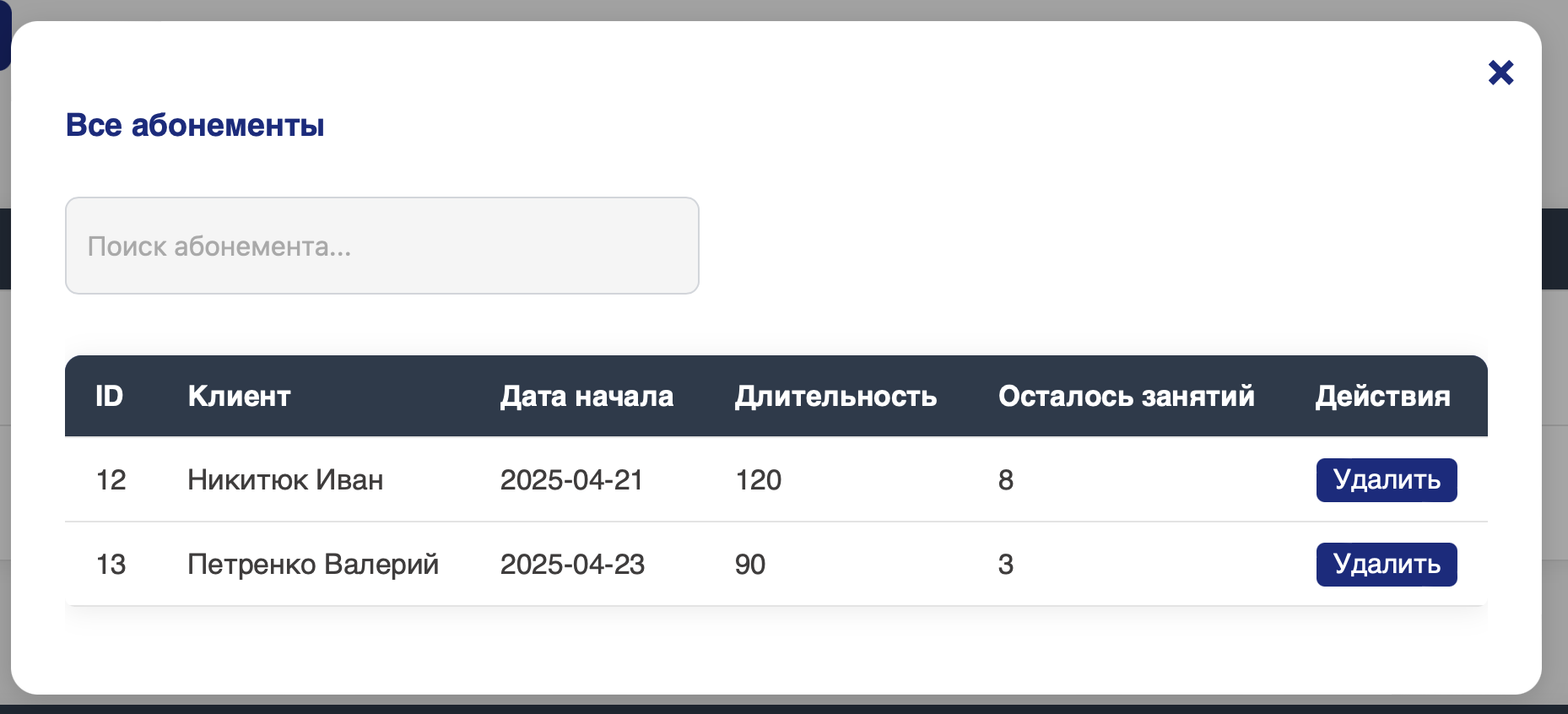


Рисунок 10.7 - Pop-up вікно Усі абонементи

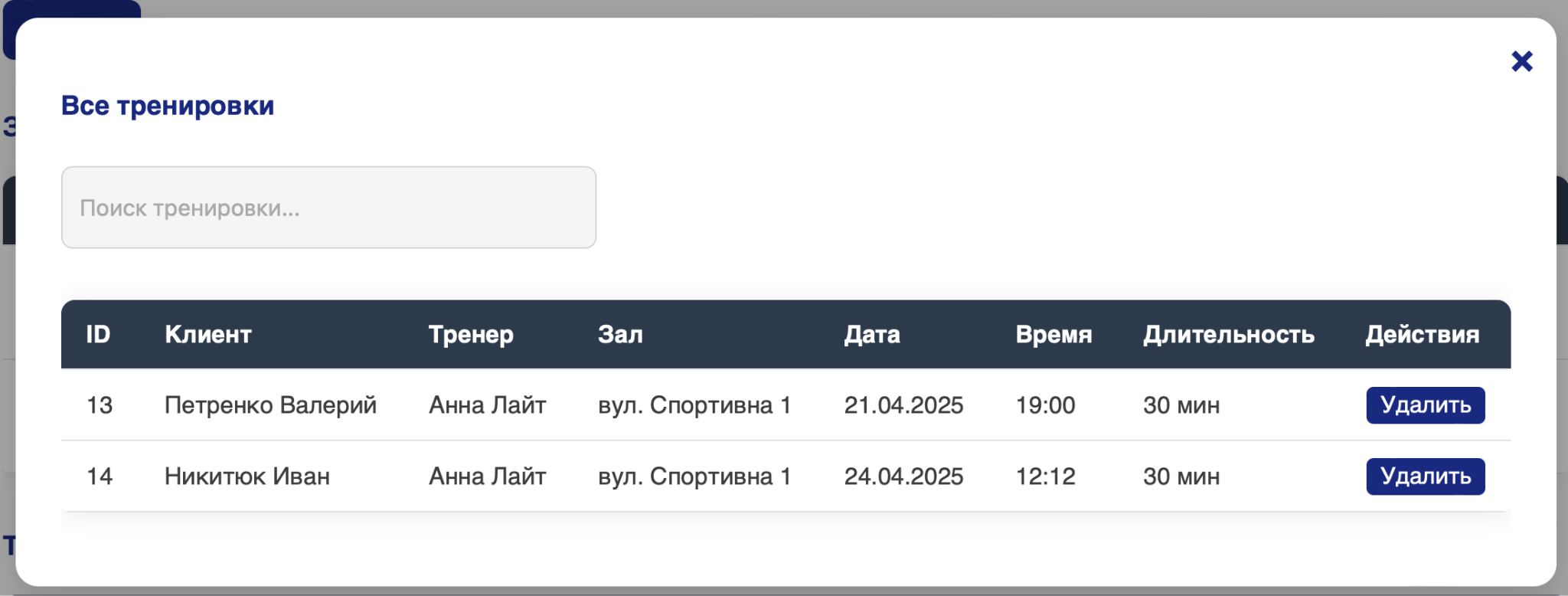


Рисунок 10.8 - Pop-up вікно Усі тренування

При натисканні кнопки видалити у будь-якого обʼєкта - відкривається Pop-up вікно з підтвердженням дії (Див. рисунок 10.9).

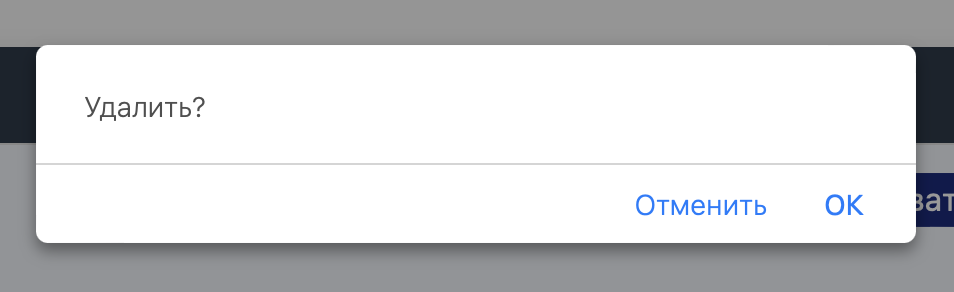


Рисунок 10.9 - Pop-up вікно з підтвердженням видалення

При натисканні кнопки редагувати у таблиці зал - відкриється Pop-up вікно редагування зали, навпроти якої користувач натиснув редагувати (Див. рисунок 10.10). При редагуванні обладнання - Pop-up вікно редагування обладнання (Див. рисунок 10.11), та Pop-up вікно редагування тренера (Див. рисунок 10.12) при редагуванні тренера.

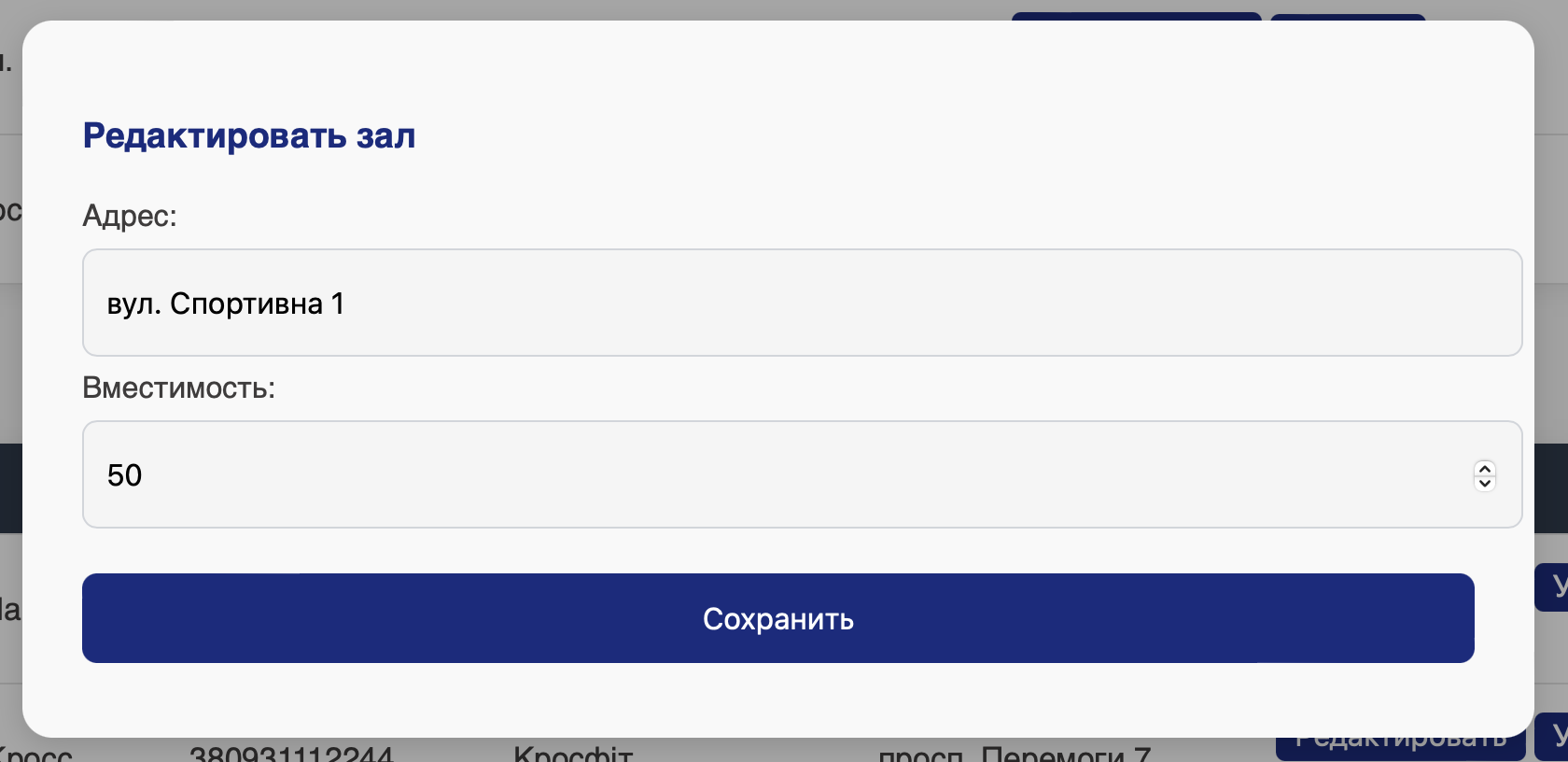


Рисунок 10.10 - Pop-up вікно редагування зали

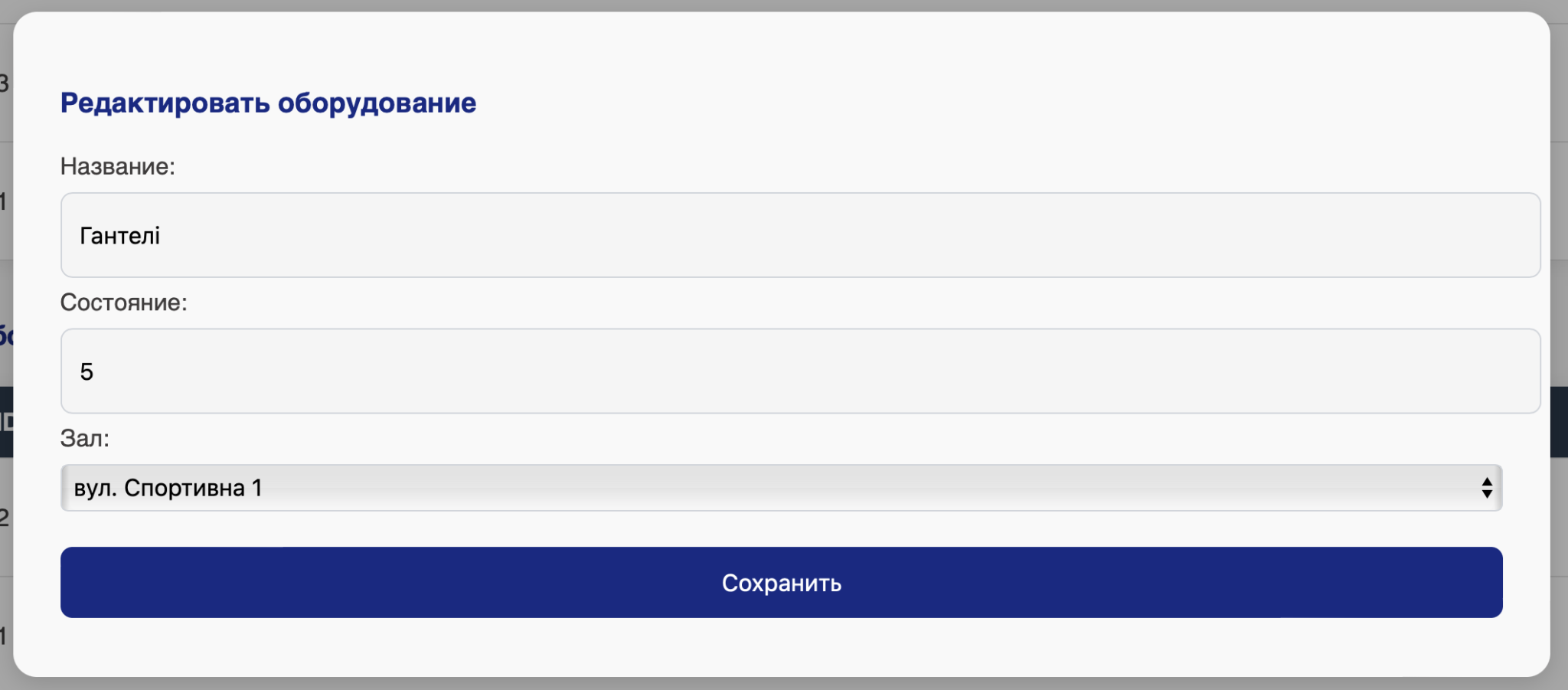


Рисунок 10.11 - Pop-up вікно редагування обладнання

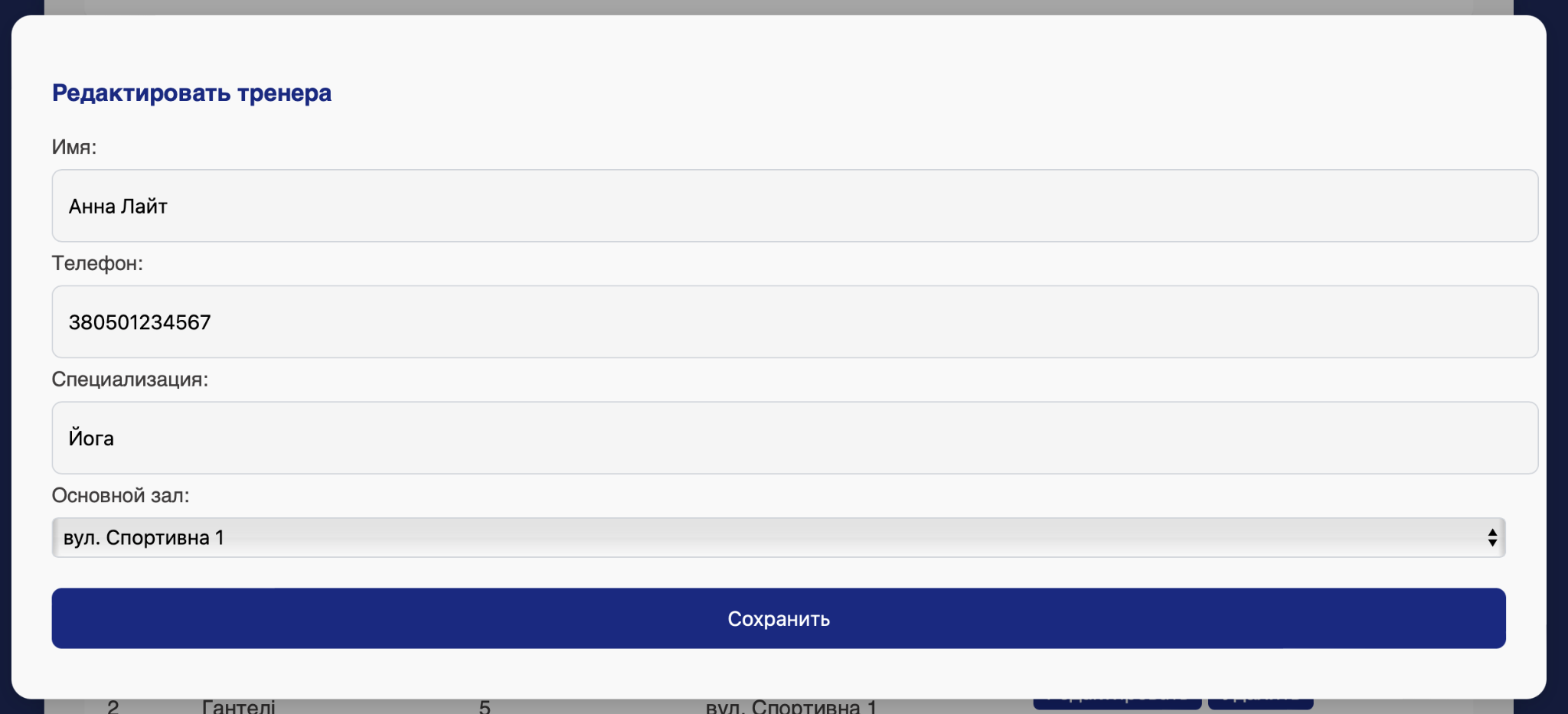


Рисунок 10.12 - Pop-up вікно редагування тренера

Якщо користувач увійшов в систему як тренер, то відкриється сторінка перегляду тренувань тренера (Див. рисунок 10.13).

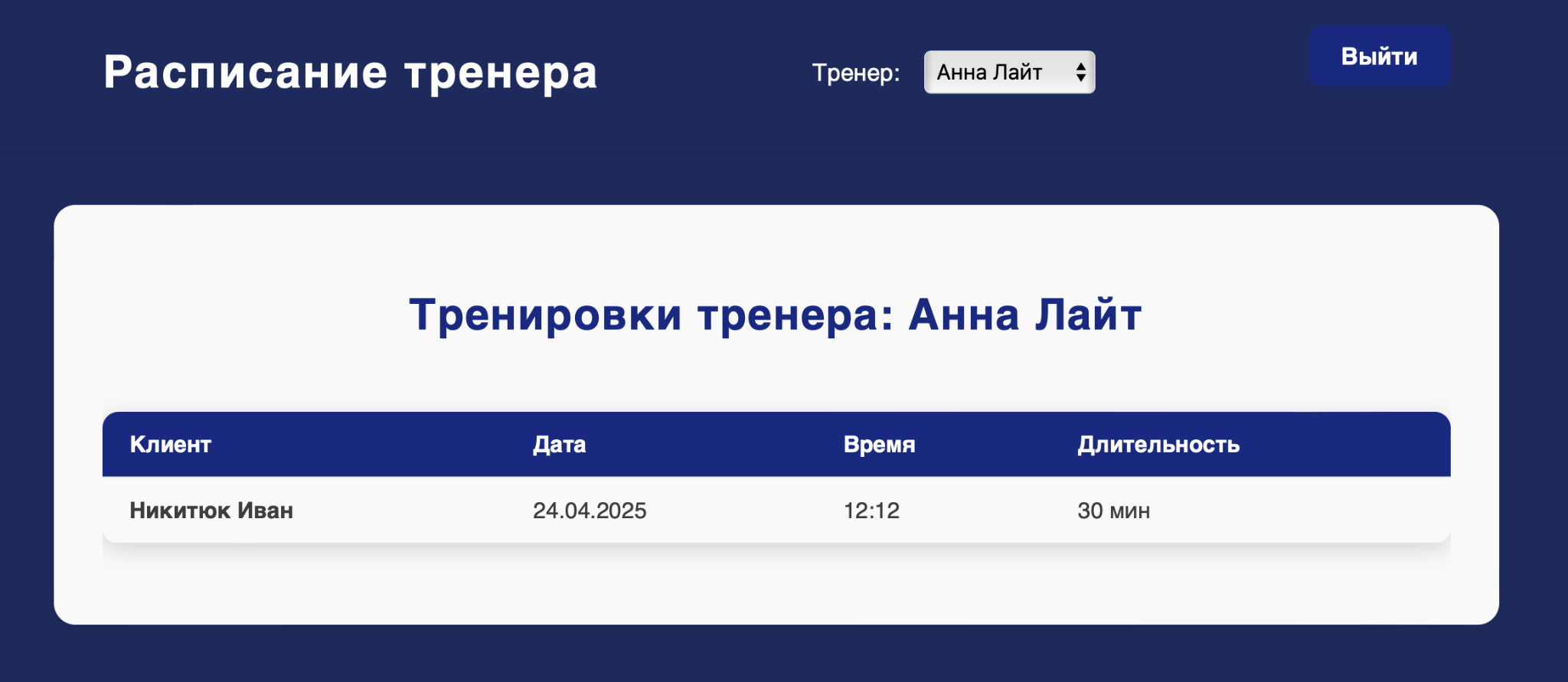


Рисунок 10.13 - Сторінка перегляду тренувань тренера

Якщо користувач увійшов в систему як реєстратор, то відкриється сторінка панелі керування реєстратора (Див. рисунок 10.14).

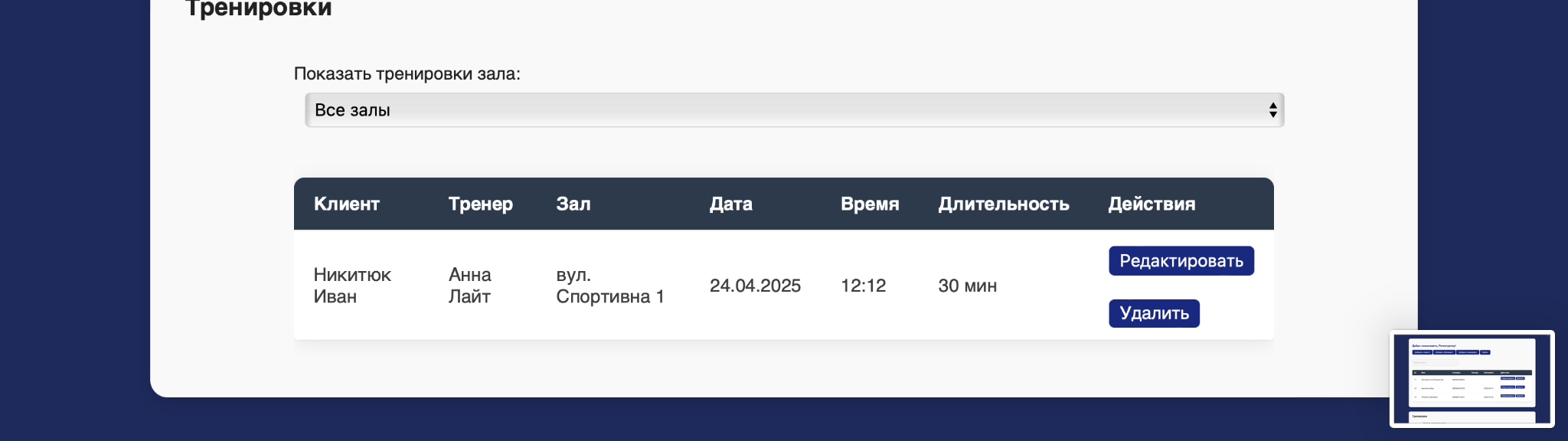
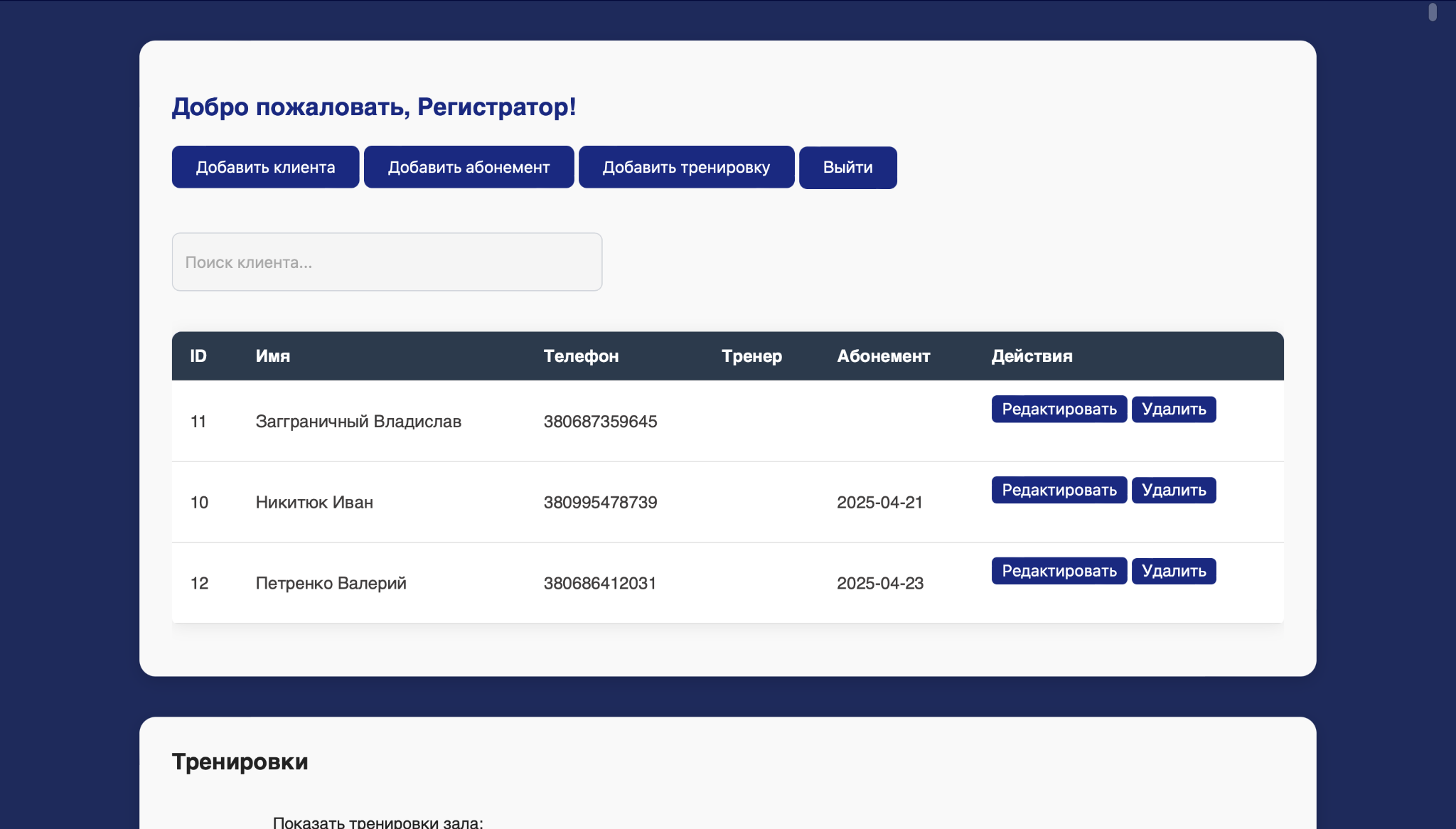


Рисунок 10.14 - Сторінка панелі керування реєстратора

Використовуючи основне навігаційне меню вгорі реєстратор може потрапити на Pop-up вікно створення клієнта (Див. рисунок 10.15), Pop-up вікно створення абонемента (Див. рисунок 10.16) або на Pop-up вікно створення тренування (Див. рисунок 10.17).

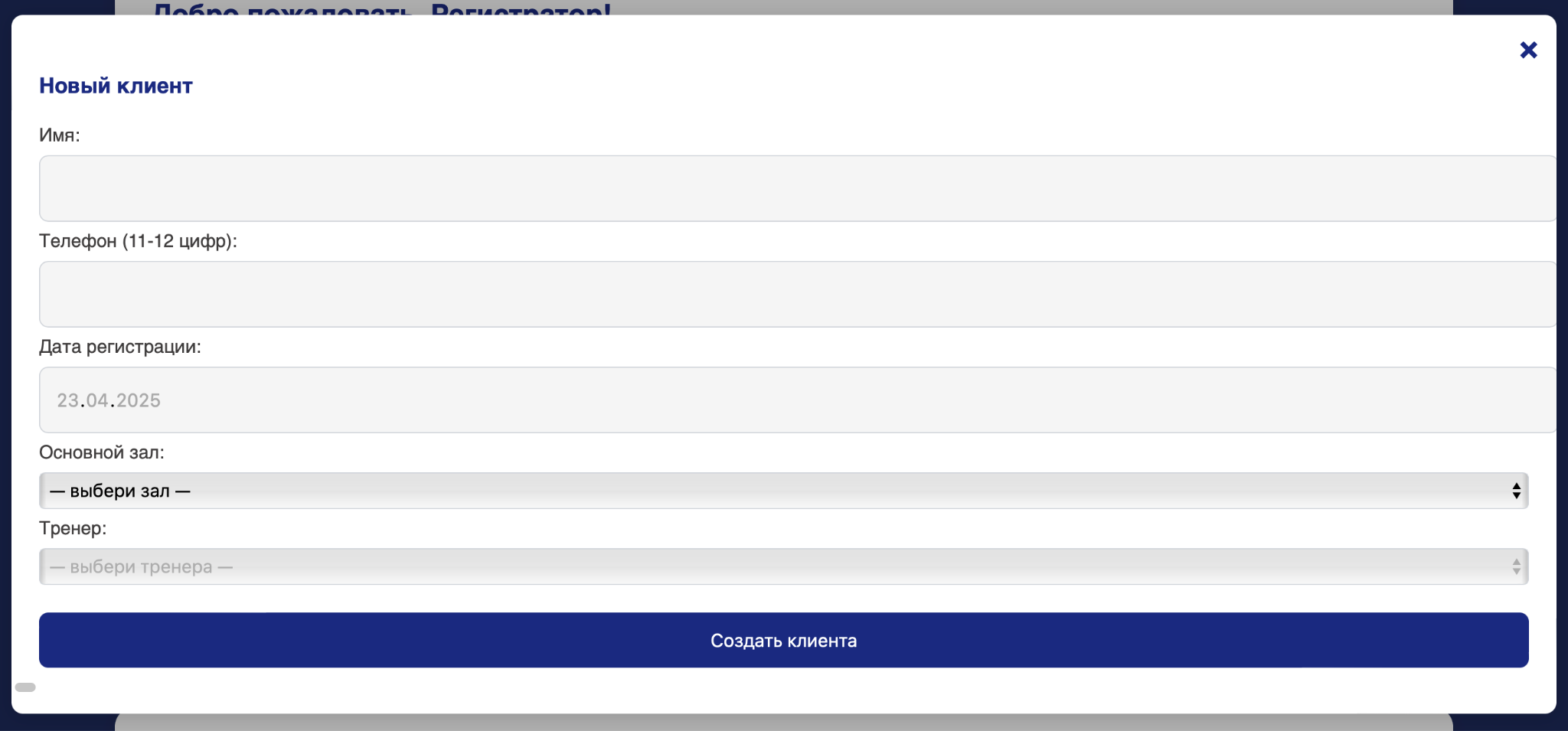


Рисунок 10.15 - Pop-up вікно створення клієнта

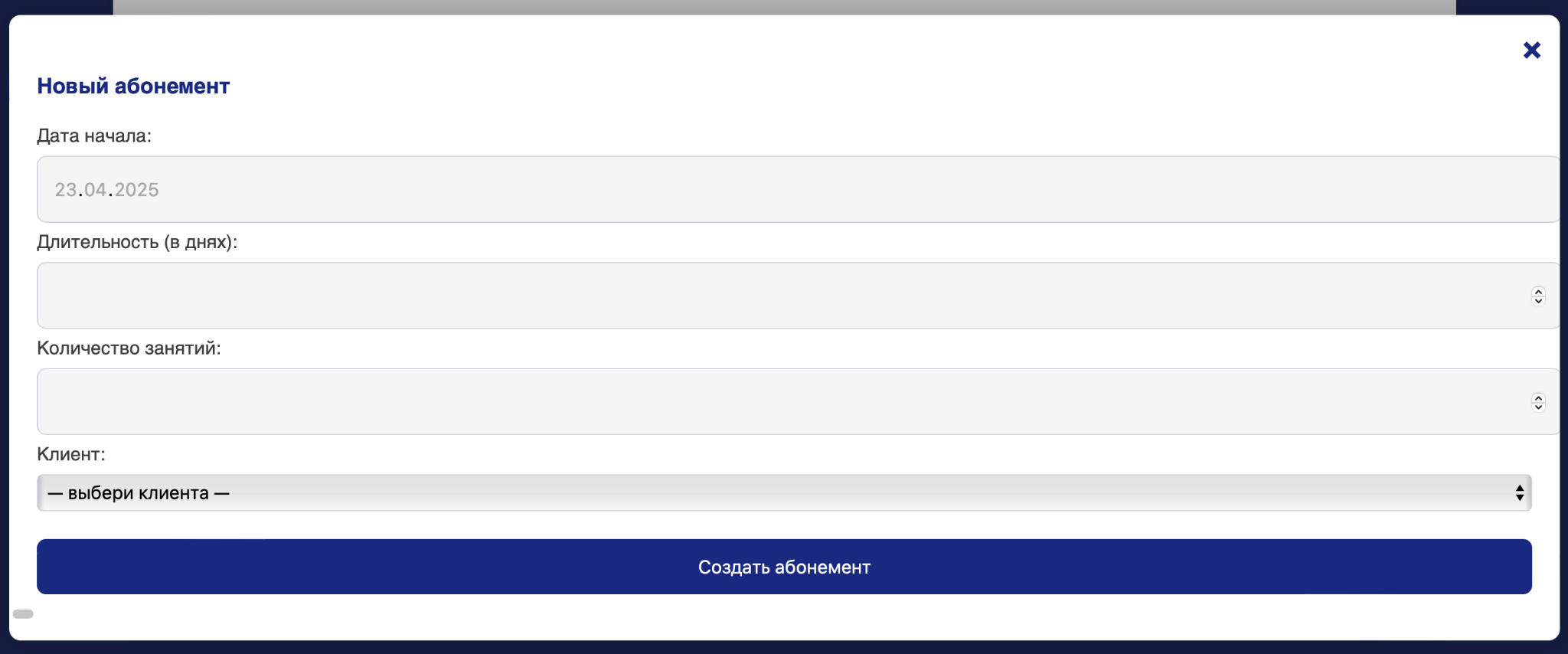


Рисунок 10.16 - Pop-up вікно створення абонемента

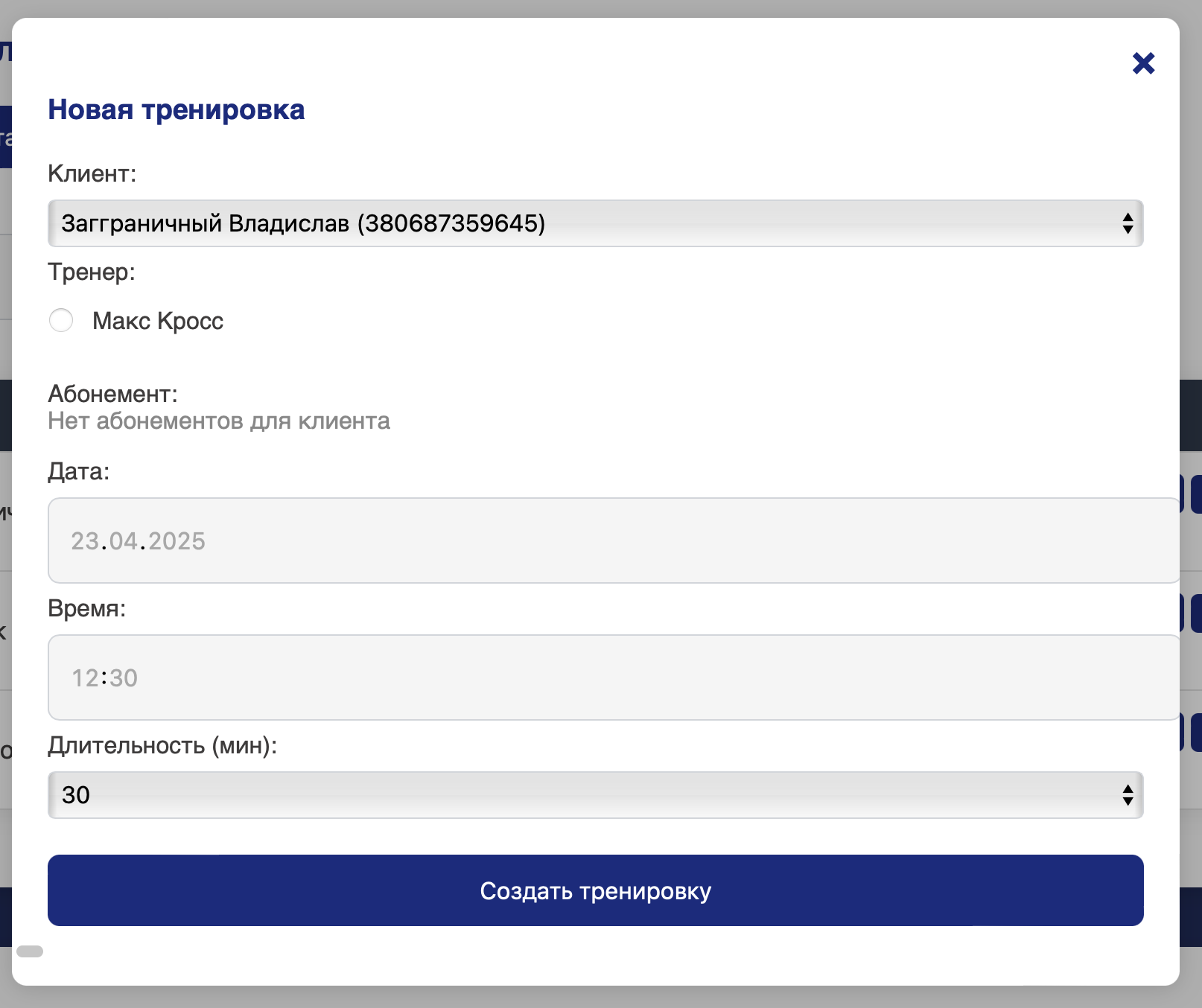


Рисунок 10.17 - Pop-up вікно створення тренування

При натискані на кнопку редагування показується Pop-up вікно редагування клієнта (Див. рисунок 10.18), Pop-up вікно редагування абонемента (Див. рисунок 10.19) або Pop-up вікно редагування тренування (Див. рисунок 10.20).

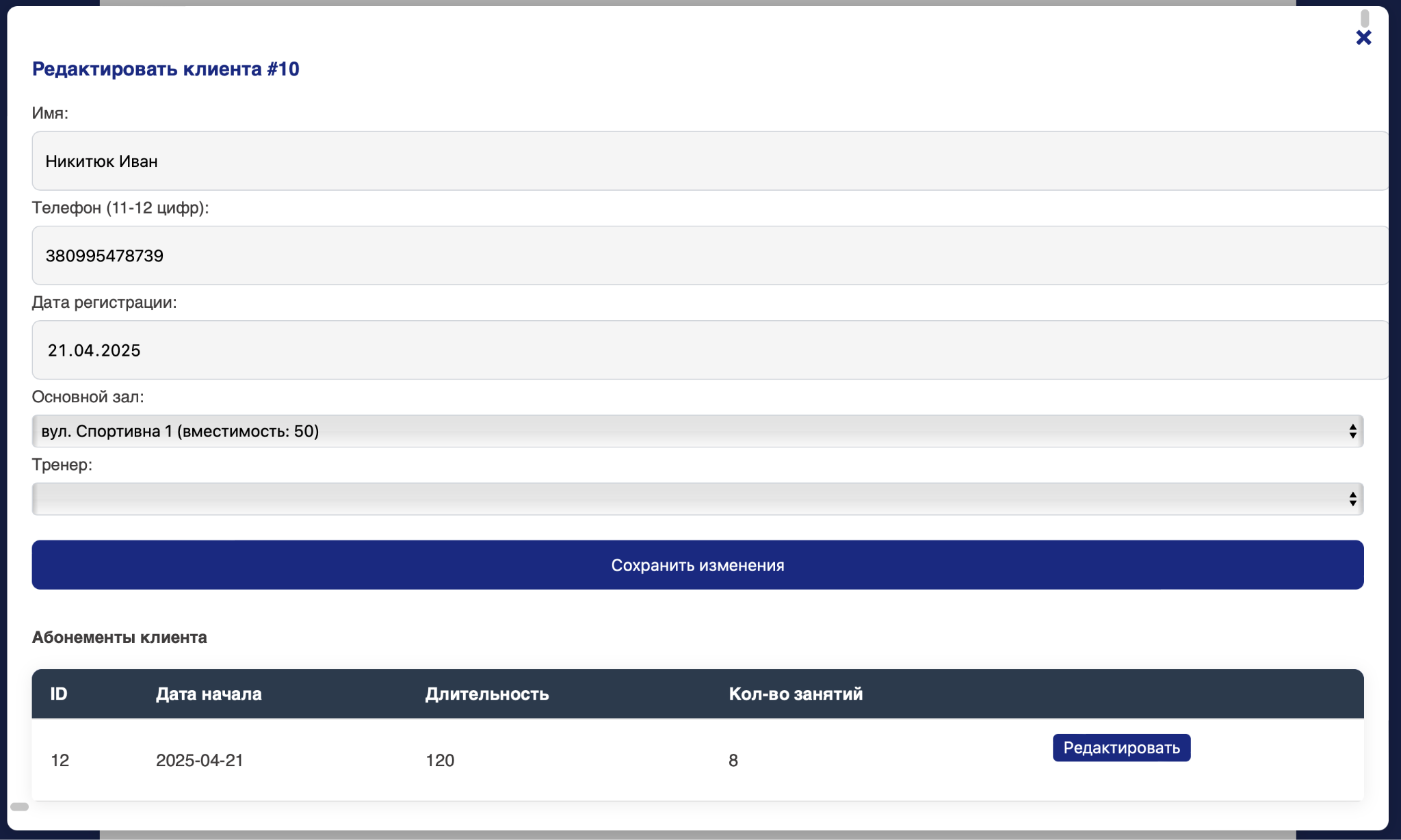


Рисунок 10.18 - Pop-up вікно редагування клієнта

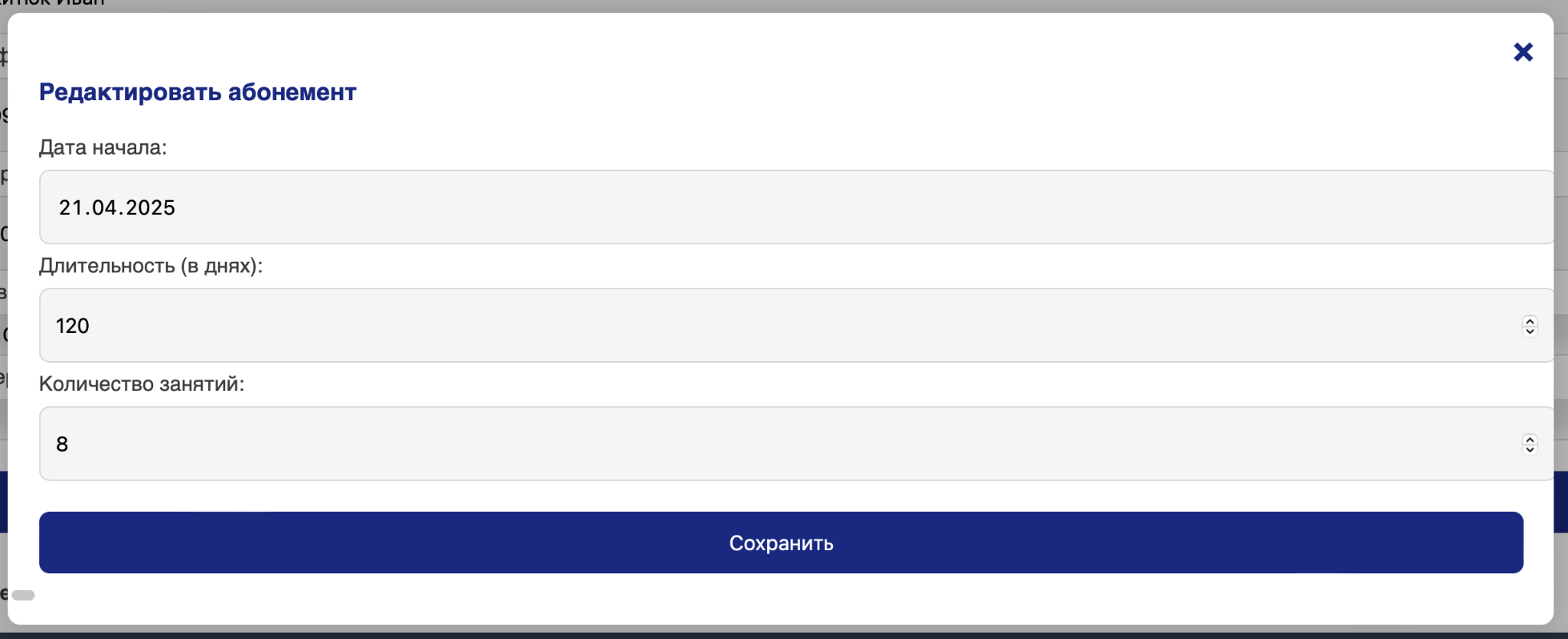


Рисунок 10.19 - Pop-up вікно редагування абонемента

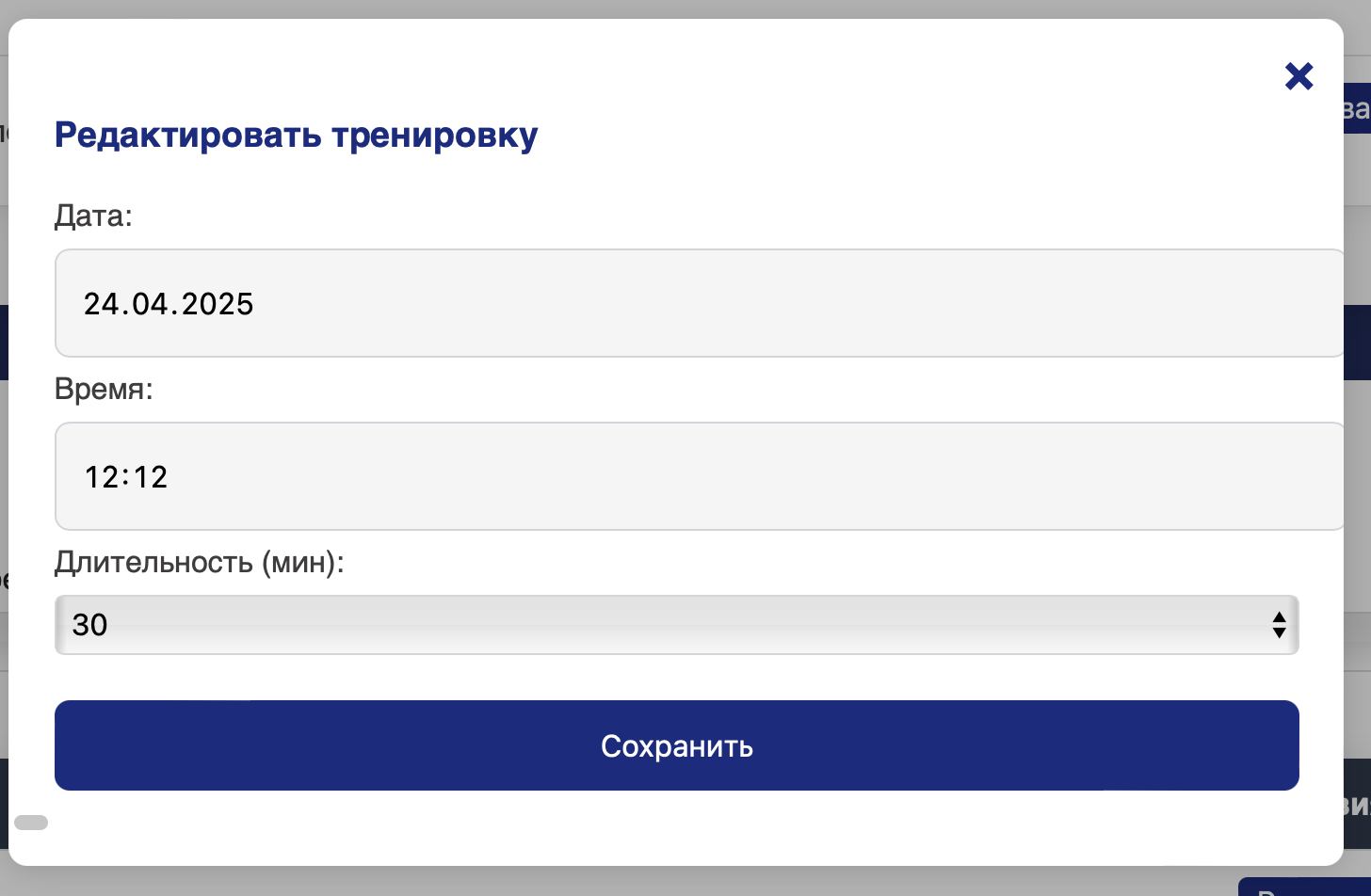


Рисунок 10.20 - Pop-up вікно редагування тренування

# 

# ВИСНОВКИ

У ході виконання курсового проєкту створено повноцінну веб-інформаційну систему для співробітників фітнес-клубу. Вона включає три ролі користувачів (адміністратор, реєстратор, тренер), реалізовані механізми авторизації та розмежування прав, CRUD-операції для всіх сутностей (зали, обладнання, тренери, клієнти, абонементи, тренування) та зручний інтерфейс із модальними вікнами і фільтраціями. Серверна частина побудована на Flask із використанням SQLAlchemy, фронтенд — на HTML/CSS/JavaScript, а дані зберігаються у PostgreSQL згідно з ER-діаграмою.

При реалізації застосовано знання з дисциплін «Організація баз даних та знань», «Веб-технології» та «Веб-дизайн»:

Побудова концептуальної та логічної моделі даних, нормалізація таблиць, налаштування зовнішніх ключів і тригерів для підтримки цілісності;

Проєктування трирівневої архітектури (шар даних, шар бізнес-логіки, шар представлення) та RESTful API;

Розробка адаптивного інтерфейсу з урахуванням UX-принципів, використання модульного CSS і JavaScript для динамічності;

Реалізація безпечних практик: валідація введених даних, хешування паролів, сесійна аутентифікація.

Система пройшла багаторазове функціональне тестування: перевірялися всі сценарії роботи (створення, редагування, видалення, фільтрація), обробка некоректних введень та помилок з’єднання з БД. В ході тестування виявлено й усунено неточності у валідації дат, у підрахунку ліміту занять за абонементом, а також дрібні помилки в UI-скриптах. Завдяки цьому досягнуто стабільної та передбачуваної роботи системи.

Звіт курсового проєкту оформлено згідно з ДСТУ та внутрішніми вимогами університету: наведено повний опис предметної області, ER-діаграми, структуру проекту, перелік SQL-запитів, кодові фрагменти та інструкцію з розгортання. README містить покрокову інструкцію для запуску, а коментарі в коді полегшують його розуміння та підтримку.

Розроблена інформаційна система готова до подальшого масштабування та інтеграції з мобільними клієнтами або сторонніми сервісами (SMS/Email-повідомлення, аналітика відвідувань, інтеграція з CRM). Можливе впровадження додаткових модулів — наприклад, онлайн-оплати абонементів, планування групових тренувань або інтеграція з API фітнес-трекерів.

Отже, курсова робота виконана повністю: створено працездатний продукт, що демонструє набуті компетентності та може бути використаний як основа для реального комерційного чи відкрите-соурсного рішення у сфері управління фітнес-клубом. Всі поставлені завдання досягнуто, а отриманий результат відповідає вимогам та цілям проєкту.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Малахов Є.В., Бази даних. Коспект лекцій. : Одеса : Державний університет інтелектуальних технологій та зв’язку. Редакція: 09.2022. – 204с.

2. PostgreSQL Global Development Group. PostgreSQL Documentation: Режим доступу: <https://www.postgresql.org/docs/> .

3. Google. Golang Documentation: Режим доступу: <https://go.dev/doc/>

4. Python 3.13.3 documentation: Режим доступу: <https://docs.python.org/3/>

# ДОДАТОК А. КОД ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ЗАСТОСУНКУ

**Програмний код Backend-частини застосунку**

Лістинг А.1 - Програмний код app.py

from flask import Flask, render\_template, request, redirect, session, url\_for, flash

from flask\_sqlalchemy import SQLAlchemy

from werkzeug.security import generate\_password\_hash, check\_password\_hash

import re

from datetime import date, timedelta, datetime

app = Flask(\_\_name\_\_)

app.secret\_key = 'supersecretkey'

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgresql://postgres:123456@localhost:5432/postgres'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

# ===== MODELS =====

class User(db.Model):

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(80), unique=True, nullable=False)

password\_hash = db.Column(db.String(128), nullable=False)

role = db.Column(db.String(20), nullable=False) # admin, trener, registrar

class Room(db.Model):

id = db.Column(db.Integer, primary\_key=True)

address = db.Column(db.String(255), nullable=False)

capacity = db.Column(db.Integer, nullable=False)

equipments = db.relationship('Equipment', backref='room', lazy=True)

class Equipment(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(100), nullable=False)

state = db.Column(db.Integer, nullable=False)

room\_id = db.Column(db.Integer, db.ForeignKey('room.id'), nullable=False)

class Trener(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(100), nullable=False)

phone = db.Column(db.String(12), nullable=False)

spec = db.Column(db.String(100), nullable=False)

main\_room\_id = db.Column(db.Integer, db.ForeignKey('room.id'), nullable=False)

main\_room = db.relationship('Room', backref='trainers', lazy=True)

class Subscription(db.Model):

id = db.Column(db.Integer, primary\_key=True)

start\_date = db.Column(db.Date, nullable=False)

duration = db.Column(db.Integer, nullable=False)

workout = db.Column(db.Integer, nullable=False)

client\_id = db.Column(db.Integer, db.ForeignKey('client.id'), nullable=False)

class Client(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(100), nullable=False)

phone = db.Column(db.String(12), nullable=False)

reg\_date = db.Column(db.Date, nullable=False)

main\_room\_id = db.Column(db.Integer, db.ForeignKey('room.id'), nullable=False)

trener\_id = db.Column(db.Integer, db.ForeignKey('trener.id'), nullable=True)

sub\_id = db.Column(db.Integer, db.ForeignKey('subscription.id'), nullable=True)

class Training(db.Model):

id = db.Column(db.Integer, primary\_key=True)

client\_id = db.Column(db.Integer, db.ForeignKey('client.id'), nullable=False)

sub\_id = db.Column(db.Integer, db.ForeignKey('subscription.id'), nullable=False)

trener\_id = db.Column(db.Integer, db.ForeignKey('trener.id'), nullable=False)

room\_id = db.Column(db.Integer, db.ForeignKey('room.id'), nullable=False)

date\_time = db.Column(db.DateTime, nullable=False)

duration = db.Column(db.Integer, nullable=False)

# ===== AUTH & DASHBOARDS =====

@app.route('/')

def index():

return redirect(url\_for('login'))

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

user = User.query.filter\_by(username=request.form['username']).first()

if user and check\_password\_hash(user.password\_hash, request.form['password']):

session['user\_id'], session['role'] = user.id, user.role

return redirect(url\_for(f"{user.role}\_dashboard"))

flash('Неверный логин или пароль')

return render\_template('login.html')

@app.route('/logout')

def logout():

session.clear()

return redirect(url\_for('login'))

@app.route('/admin')

def admin\_dashboard():

if session.get('role')!='admin': return redirect(url\_for('login'))

trainers = Trener.query.all()

rooms = Room.query.all()

equipment = Equipment.query.all()

clients = Client.query.all()

subs = Subscription.query.all()

trainings = Training.query.all()

return render\_template('admin.html', trainers=trainers, rooms=rooms, equipment=equipment, clients=clients, subs=subs, trainings=trainings)

@app.route('/registrar')

def registrar\_dashboard():

if session.get('role')!='registrar': return redirect(url\_for('login'))

clients = Client.query.all()

subs = Subscription.query.all()

trainers = Trener.query.all()

rooms = Room.query.all()

current\_date = date.today()

from datetime import datetime, timedelta

now = datetime.now()

# Только будущие и текущие тренировки

all\_trainings = Training.query.all()

trainings = []

for t in all\_trainings:

end\_time = t.date\_time + timedelta(minutes=t.duration)

if end\_time >= now:

trainings.append(t)

return render\_template('registrar.html', clients=clients, subs=subs, trainings=trainings, trainers=trainers, rooms=rooms, current\_date=current\_date)

@app.route('/trener')

def trener\_dashboard():

if session.get('role')!='trener': return redirect(url\_for('login'))

trener\_id = request.args.get('trener\_id', type=int)

all\_trainers = Trener.query.all()

clients = Client.query.all()

now = datetime.now()

if trener\_id:

selected\_trener = Trener.query.get\_or\_404(trener\_id)

else:

selected\_trener = all\_trainers[0] if all\_trainers else None

# Только будущие и текущие тренировки

trainings = []

if selected\_trener:

all\_trainings = Training.query.filter\_by(trener\_id=selected\_trener.id).all()

for t in all\_trainings:

end\_time = t.date\_time + timedelta(minutes=t.duration)

if end\_time >= now:

trainings.append(t)

return render\_template('trener.html', all\_trainers=all\_trainers, selected\_trener=selected\_trener, trainings=trainings, clients=clients)

# ===== CRUD FOR ADMIN =====

@app.route('/admin/room/create', methods=['POST'])

def admin\_create\_room():

db.session.add(Room(address=request.form['address'], capacity=request.form['capacity']))

db.session.commit(); return redirect(url\_for('admin\_dashboard'))

@app.route('/admin/trener/create', methods=['POST'])

def admin\_create\_trener():

db.session.add(Trener(

name=request.form['name'], phone=request.form['phone'],

spec=request.form['spec'], main\_room\_id=request.form['main\_room\_id']))

db.session.commit(); return redirect(url\_for('admin\_dashboard'))

@app.route('/admin/room/edit/<int:id>', methods=['POST'])

def admin\_edit\_room(id):

r = Room.query.get\_or\_404(id)

r.address = request.form['address']

r.capacity = request.form['capacity']

db.session.commit()

return redirect(url\_for('admin\_dashboard'))

@app.route('/admin/trener/edit/<int:id>', methods=['POST'])

def admin\_edit\_trener(id):

t = Trener.query.get\_or\_404(id)

t.name = request.form['name']

t.phone = request.form['phone']

t.spec = request.form['spec']

t.main\_room\_id = request.form['main\_room\_id']

db.session.commit()

return redirect(url\_for('admin\_dashboard'))

@app.route('/admin/equipment/create', methods=['POST'])

def admin\_create\_equipment():

db.session.add(Equipment(

name=request.form['name'],

state=request.form['state'],

room\_id=request.form['room\_id'] or None

))

db.session.commit()

return redirect(url\_for('admin\_dashboard'))

@app.route('/admin/equipment/edit/<int:id>', methods=['POST'])

def admin\_edit\_equipment(id):

e = Equipment.query.get\_or\_404(id)

e.name = request.form['name']

e.state = request.form['state']

e.room\_id = request.form['room\_id'] or None

db.session.commit()

return redirect(url\_for('admin\_dashboard'))

@app.route('/admin/equipment/delete/<int:id>')

def admin\_delete\_equipment(id):

e = Equipment.query.get\_or\_404(id)

db.session.delete(e)

db.session.commit()

return redirect(url\_for('admin\_dashboard'))

@app.route('/admin/room/delete/<int:id>')

def admin\_delete\_room(id):

# Оборудование из этого зала становится без зала

Equipment.query.filter\_by(room\_id=id).update({'room\_id': None})

# Удаляем все тренировки, связанные с этим залом

Training.query.filter\_by(room\_id=id).delete()

r = Room.query.get\_or\_404(id)

db.session.delete(r)

db.session.commit()

return redirect(url\_for('admin\_dashboard'))

@app.route('/admin/trener/delete/<int:id>')

def admin\_delete\_trener(id):

# Обнуляем trener\_id у всех клиентов, у которых был этот тренер

Client.query.filter\_by(trener\_id=id).update({'trener\_id': None})

# Для всех тренировок этого тренера вернуть по 1 тренировке соответствующим абонементам

trainings = Training.query.filter\_by(trener\_id=id).all()

sub\_ids = set()

for t in trainings:

sub\_ids.add(t.sub\_id)

for sub\_id in sub\_ids:

sub = Subscription.query.get(sub\_id)

if sub:

sub.workout += Training.query.filter\_by(trener\_id=id, sub\_id=sub\_id).count()

# Удаляем все тренировки, связанные с этим тренером

Training.query.filter\_by(trener\_id=id).delete()

db.session.commit()

t = Trener.query.get\_or\_404(id)

db.session.delete(t)

db.session.commit()

return redirect(url\_for('admin\_dashboard'))

@app.route('/admin/sub/delete/<int:id>')

def admin\_delete\_sub(id):

sub = Subscription.query.get\_or\_404(id)

# Обнуляем sub\_id у клиента, если этот абонемент был привязан

client = Client.query.filter\_by(sub\_id=sub.id).first()

if client:

client.sub\_id = None

# Удаляем все тренировки, связанные с этим абонементом

Training.query.filter\_by(sub\_id=sub.id).delete()

db.session.delete(sub)

db.session.commit()

return redirect(url\_for('admin\_dashboard'))

@app.route('/admin/training/delete/<int:id>')

def admin\_delete\_training(id):

training = Training.query.get\_or\_404(id)

sub = Subscription.query.get(training.sub\_id)

if sub:

sub.workout += 1

db.session.delete(training)

db.session.commit()

return redirect(url\_for('admin\_dashboard'))

@app.route('/admin/client/delete/<int:id>')

def admin\_delete\_client(id):

c = Client.query.get\_or\_404(id)

# Сначала обнуляем sub\_id у клиента, чтобы не было ссылки на абонемент

c.sub\_id = None

db.session.commit()

# Удаляем все тренировки клиента

Training.query.filter\_by(client\_id=c.id).delete()

# Удаляем все абонементы клиента

Subscription.query.filter\_by(client\_id=c.id).delete()

db.session.delete(c)

db.session.commit()

return redirect(url\_for('admin\_dashboard'))

# ===== CRUD FOR REGISTRAR =====

@app.route('/registrar/client/create', methods=['POST'])

def reg\_create\_client():

new\_client = Client(

name=request.form['name'],

phone=request.form['phone'],

reg\_date=request.form['reg\_date'],

main\_room\_id=request.form['main\_room\_id'],

trener\_id=request.form['trener\_id'] or None

)

db.session.add(new\_client)

db.session.commit()

return redirect(url\_for('registrar\_dashboard'))

@app.route('/registrar/client/edit/<int:id>', methods=['POST'])

def reg\_edit\_client(id):

c = Client.query.get\_or\_404(id)

c.name = request.form['name']

c.phone = request.form['phone']

c.reg\_date = request.form['reg\_date']

c.main\_room\_id = request.form['main\_room\_id']

c.trener\_id = request.form['trener\_id'] or None

db.session.commit()

return redirect(url\_for('registrar\_dashboard'))

@app.route('/registrar/client/delete/<int:id>')

def reg\_delete\_client(id):

c = Client.query.get\_or\_404(id)

# Сначала обнуляем sub\_id у клиента, чтобы не было ссылки на абонемент

c.sub\_id = None

db.session.commit()

# Удаляем все тренировки клиента

Training.query.filter\_by(client\_id=c.id).delete()

# Удаляем все абонементы клиента

Subscription.query.filter\_by(client\_id=c.id).delete()

db.session.delete(c)

db.session.commit()

return redirect(url\_for('registrar\_dashboard'))

@app.route('/registrar/sub/create', methods=['POST'])

def reg\_create\_sub():

new\_sub = Subscription(

start\_date=request.form['start\_date'],

duration=int(request.form['duration']),

workout=int(request.form['workout']),

client\_id=int(request.form['client\_id'])

)

db.session.add(new\_sub)

db.session.commit()

# Привязка абонемента к клиенту (делаем sub\_id у клиента равным id нового абонемента)

client = Client.query.get(new\_sub.client\_id)

client.sub\_id = new\_sub.id

db.session.commit()

return redirect(url\_for('registrar\_dashboard'))

@app.route('/registrar/sub/edit/<int:sub\_id>', methods=['POST'])

def reg\_edit\_sub(sub\_id):

sub = Subscription.query.get\_or\_404(sub\_id)

sub.start\_date = request.form['start\_date']

sub.duration = int(request.form['duration'])

sub.workout = int(request.form['workout'])

db.session.commit()

return redirect(url\_for('registrar\_dashboard'))

@app.route('/registrar/training/create', methods=['POST'])

def reg\_create\_training():

client\_id = int(request.form['client\_id'])

trener\_id\_raw = request.form['trener\_id']

if not trener\_id\_raw or trener\_id\_raw == 'None':

flash('Не выбран тренер!')

return redirect(url\_for('registrar\_dashboard'))

trener\_id = int(trener\_id\_raw)

sub\_id = int(request.form['sub\_id'])

date\_str = request.form['date']

time\_str = request.form['time']

duration = int(request.form['duration'])

client = Client.query.get\_or\_404(client\_id)

room\_id = client.main\_room\_id

from datetime import datetime

dt = datetime.strptime(f"{date\_str} {time\_str}", "%Y-%m-%d %H:%M")

sub = Subscription.query.get\_or\_404(sub\_id)

if sub.workout <= 0:

flash('У этого абонемента закончились тренировки!')

return redirect(url\_for('registrar\_dashboard'))

training = Training(client\_id=client\_id, trener\_id=trener\_id, sub\_id=sub\_id, room\_id=room\_id, date\_time=dt, duration=duration)

db.session.add(training)

sub.workout -= 1

db.session.commit()

return redirect(url\_for('registrar\_dashboard'))

@app.route('/registrar/training/edit/<int:training\_id>', methods=['POST'])

def reg\_edit\_training(training\_id):

training = Training.query.get\_or\_404(training\_id)

date\_str = request.form['date']

time\_str = request.form['time']

duration = int(request.form['duration'])

from datetime import datetime

dt = datetime.strptime(f"{date\_str} {time\_str}", "%Y-%m-%d %H:%M")

training.date\_time = dt

training.duration = duration

db.session.commit()

return redirect(url\_for('registrar\_dashboard'))

@app.route('/registrar/training/delete/<int:id>')

def reg\_delete\_training(id):

training = Training.query.get\_or\_404(id)

# Возвращаем тренировку абонементу

sub = Subscription.query.get(training.sub\_id)

if sub:

sub.workout += 1

db.session.delete(training)

db.session.commit()

return redirect(url\_for('registrar\_dashboard'))

# ===== RUN =====

if \_\_name\_\_=='\_\_main\_\_':

app.run(debug=True)

Лістинг А.2 - Програмний код create\_users.py

from app import app, db, User

from werkzeug.security import generate\_password\_hash

with app.app\_context():

db.create\_all()

admin = User(username='admin', password\_hash=generate\_password\_hash('adminpass', method='pbkdf2:sha256'), role='admin')

reg = User(username='reg', password\_hash=generate\_password\_hash('regpass', method='pbkdf2:sha256'), role='registrar')

trener = User(username='trener', password\_hash=generate\_password\_hash('trenerpass', method='pbkdf2:sha256'), role='trener')

db.session.add\_all([admin, reg, trener])

db.session.commit()

print("Пользователи успешно добавлены!")

Лістинг А.3 - Програмний код FitnessServer Backup (бекап бази данних та сервера)

--

-- PostgreSQL database cluster dump

--

-- Started on 2025-04-21 13:23:36 EEST

SET default\_transaction\_read\_only = off;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

--

-- Roles

--

CREATE ROLE kolyhov;

ALTER ROLE kolyhov WITH SUPERUSER INHERIT CREATEROLE CREATEDB LOGIN REPLICATION BYPASSRLS;

CREATE ROLE postgres;

ALTER ROLE postgres WITH SUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN NOREPLICATION NOBYPASSRLS PASSWORD 'SCRAM-SHA-256$4096:+QSiNc4fTa+t4W2Xxdh5bA==$xfEPs0x5E6iCBbsudY+rIU4x+4BTNYnaAaiyMS5uzNE=:sQfhcaMymUcozJEtlgzlWb5GunIP22bIuTqzVKS+NQY=';

--

-- User Configurations

--

--

-- Databases

--

--

-- Database "template1" dump

--

\connect template1

--

-- PostgreSQL database dump

--

-- Dumped from database version 14.17 (Homebrew)

-- Dumped by pg\_dump version 17.0

-- Started on 2025-04-21 13:23:36 EEST

SET statement\_timeout = 0;

SET lock\_timeout = 0;

SET idle\_in\_transaction\_session\_timeout = 0;

SET transaction\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SELECT pg\_catalog.set\_config('search\_path', '', false);

SET check\_function\_bodies = false;

SET xmloption = content;

SET client\_min\_messages = warning;

SET row\_security = off;

--

-- TOC entry 4 (class 2615 OID 2200)

-- Name: public; Type: SCHEMA; Schema: -; Owner: kolyhov

--

-- \*not\* creating schema, since initdb creates it

ALTER SCHEMA public OWNER TO kolyhov;

--

-- TOC entry 3667 (class 0 OID 0)

-- Dependencies: 4

-- Name: SCHEMA public; Type: ACL; Schema: -; Owner: kolyhov

--

REVOKE USAGE ON SCHEMA public FROM PUBLIC;

GRANT ALL ON SCHEMA public TO PUBLIC;

-- Completed on 2025-04-21 13:23:36 EEST

--

-- PostgreSQL database dump complete

--

--

-- Database "postgres" dump

--

\connect postgres

--

-- PostgreSQL database dump

--

-- Dumped from database version 14.17 (Homebrew)

-- Dumped by pg\_dump version 17.0

-- Started on 2025-04-21 13:23:36 EEST

SET statement\_timeout = 0;

SET lock\_timeout = 0;

SET idle\_in\_transaction\_session\_timeout = 0;

SET transaction\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SELECT pg\_catalog.set\_config('search\_path', '', false);

SET check\_function\_bodies = false;

SET xmloption = content;

SET client\_min\_messages = warning;

SET row\_security = off;

--

-- TOC entry 4 (class 2615 OID 2200)

-- Name: public; Type: SCHEMA; Schema: -; Owner: kolyhov

--

-- \*not\* creating schema, since initdb creates it

ALTER SCHEMA public OWNER TO kolyhov;

SET default\_tablespace = '';

SET default\_table\_access\_method = heap;

--

-- TOC entry 218 (class 1259 OID 16428)

-- Name: client; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.client (

id integer NOT NULL,

name character varying(100) NOT NULL,

phone character(12) NOT NULL,

reg\_date date NOT NULL,

trener\_id integer,

sub\_id integer,

main\_room\_id integer

);

ALTER TABLE public.client OWNER TO postgres;

--

-- TOC entry 217 (class 1259 OID 16427)

-- Name: client\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.client\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.client\_id\_seq OWNER TO postgres;

--

-- TOC entry 3756 (class 0 OID 0)

-- Dependencies: 217

-- Name: client\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.client\_id\_seq OWNED BY public.client.id;

--

-- TOC entry 212 (class 1259 OID 16394)

-- Name: equipment; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.equipment (

id integer NOT NULL,

name character varying(100) NOT NULL,

state integer NOT NULL,

room\_id integer,

CONSTRAINT equipment\_state\_check CHECK (((state >= 1) AND (state <= 5)))

);

ALTER TABLE public.equipment OWNER TO postgres;

--

-- TOC entry 211 (class 1259 OID 16393)

-- Name: equipment\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.equipment\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.equipment\_id\_seq OWNER TO postgres;

--

-- TOC entry 3757 (class 0 OID 0)

-- Dependencies: 211

-- Name: equipment\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.equipment\_id\_seq OWNED BY public.equipment.id;

--

-- TOC entry 210 (class 1259 OID 16386)

-- Name: room; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.room (

id integer NOT NULL,

address character varying(255) NOT NULL,

capacity integer NOT NULL,

CONSTRAINT room\_capacity\_check CHECK ((capacity > 0))

);

ALTER TABLE public.room OWNER TO postgres;

--

-- TOC entry 209 (class 1259 OID 16385)

-- Name: room\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.room\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.room\_id\_seq OWNER TO postgres;

--

-- TOC entry 3758 (class 0 OID 0)

-- Dependencies: 209

-- Name: room\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.room\_id\_seq OWNED BY public.room.id;

--

-- TOC entry 216 (class 1259 OID 16419)

-- Name: subscription; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.subscription (

id integer NOT NULL,

start\_date date NOT NULL,

duration integer NOT NULL,

workout integer NOT NULL,

client\_id integer NOT NULL,

CONSTRAINT subscription\_duration\_check CHECK ((duration >= 0)),

CONSTRAINT subscription\_workout\_check CHECK ((workout >= 0))

);

ALTER TABLE public.subscription OWNER TO postgres;

--

-- TOC entry 215 (class 1259 OID 16418)

-- Name: subscription\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.subscription\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.subscription\_id\_seq OWNER TO postgres;

--

-- TOC entry 3759 (class 0 OID 0)

-- Dependencies: 215

-- Name: subscription\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.subscription\_id\_seq OWNED BY public.subscription.id;

--

-- TOC entry 220 (class 1259 OID 16445)

-- Name: training; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.training (

id integer NOT NULL,

client\_id integer NOT NULL,

sub\_id integer NOT NULL,

trener\_id integer NOT NULL,

room\_id integer NOT NULL,

date\_time timestamp without time zone NOT NULL,

duration integer NOT NULL,

CONSTRAINT training\_duration\_check CHECK ((duration >= 30))

);

ALTER TABLE public.training OWNER TO postgres;

--

-- TOC entry 219 (class 1259 OID 16444)

-- Name: training\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.training\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.training\_id\_seq OWNER TO postgres;

--

-- TOC entry 3760 (class 0 OID 0)

-- Dependencies: 219

-- Name: training\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.training\_id\_seq OWNED BY public.training.id;

--

-- TOC entry 214 (class 1259 OID 16407)

-- Name: trener; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.trener (

id integer NOT NULL,

name character varying(100) NOT NULL,

phone character(12) NOT NULL,

spec character varying(100) NOT NULL,

main\_room\_id integer

);

ALTER TABLE public.trener OWNER TO postgres;

--

-- TOC entry 213 (class 1259 OID 16406)

-- Name: trener\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.trener\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.trener\_id\_seq OWNER TO postgres;

--

-- TOC entry 3761 (class 0 OID 0)

-- Dependencies: 213

-- Name: trener\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.trener\_id\_seq OWNED BY public.trener.id;

--

-- TOC entry 222 (class 1259 OID 16473)

-- Name: user; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public."user" (

id integer NOT NULL,

username character varying(80) NOT NULL,

password\_hash character varying(128) NOT NULL,

role character varying(20) NOT NULL

);

ALTER TABLE public."user" OWNER TO postgres;

--

-- TOC entry 221 (class 1259 OID 16472)

-- Name: user\_id\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public.user\_id\_seq

AS integer

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE public.user\_id\_seq OWNER TO postgres;

--

-- TOC entry 3762 (class 0 OID 0)

-- Dependencies: 221

-- Name: user\_id\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public.user\_id\_seq OWNED BY public."user".id;

--

-- TOC entry 3562 (class 2604 OID 16431)

-- Name: client id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.client ALTER COLUMN id SET DEFAULT nextval('public.client\_id\_seq'::regclass);

--

-- TOC entry 3559 (class 2604 OID 16397)

-- Name: equipment id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.equipment ALTER COLUMN id SET DEFAULT nextval('public.equipment\_id\_seq'::regclass);

--

-- TOC entry 3558 (class 2604 OID 16389)

-- Name: room id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.room ALTER COLUMN id SET DEFAULT nextval('public.room\_id\_seq'::regclass);

--

-- TOC entry 3561 (class 2604 OID 16422)

-- Name: subscription id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.subscription ALTER COLUMN id SET DEFAULT nextval('public.subscription\_id\_seq'::regclass);

--

-- TOC entry 3563 (class 2604 OID 16448)

-- Name: training id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.training ALTER COLUMN id SET DEFAULT nextval('public.training\_id\_seq'::regclass);

--

-- TOC entry 3560 (class 2604 OID 16410)

-- Name: trener id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.trener ALTER COLUMN id SET DEFAULT nextval('public.trener\_id\_seq'::regclass);

--

-- TOC entry 3564 (class 2604 OID 16476)

-- Name: user id; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."user" ALTER COLUMN id SET DEFAULT nextval('public.user\_id\_seq'::regclass);

--

-- TOC entry 3745 (class 0 OID 16428)

-- Dependencies: 218

-- Data for Name: client; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.client (id, name, phone, reg\_date, trener\_id, sub\_id, main\_room\_id) FROM stdin;

11 Загграничный Владислав 380687359645 2025-04-17 \N \N 2

12 Петренко Валерий 380686412031 2025-04-01 \N \N 1

10 Никитюк Иван 380995478739 2025-04-21 \N 12 1

\.

--

-- TOC entry 3739 (class 0 OID 16394)

-- Dependencies: 212

-- Data for Name: equipment; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.equipment (id, name, state, room\_id) FROM stdin;

2 Гантелі 5 1

3 Велотренажер 3 2

1 Бігова доріжка 4 1

\.

--

-- TOC entry 3737 (class 0 OID 16386)

-- Dependencies: 210

-- Data for Name: room; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.room (id, address, capacity) FROM stdin;

1 вул. Спортивна 1 50

2 просп. Перемоги 7 30

\.

--

-- TOC entry 3743 (class 0 OID 16419)

-- Dependencies: 216

-- Data for Name: subscription; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.subscription (id, start\_date, duration, workout, client\_id) FROM stdin;

12 2025-04-21 120 8 10

\.

--

-- TOC entry 3747 (class 0 OID 16445)

-- Dependencies: 220

-- Data for Name: training; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.training (id, client\_id, sub\_id, trener\_id, room\_id, date\_time, duration) FROM stdin;

13 12 12 2 1 2025-04-21 19:00:00 30

14 10 12 2 1 2025-04-24 12:12:00 30

\.

--

-- TOC entry 3741 (class 0 OID 16407)

-- Dependencies: 214

-- Data for Name: trener; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public.trener (id, name, phone, spec, main\_room\_id) FROM stdin;

1 Олег Шварц 380971112233 Силові 1

2 Анна Лайт 380501234567 Йога 1

3 Макс Кросс 380931112244 Кросфіт 2

\.

--

-- TOC entry 3749 (class 0 OID 16473)

-- Dependencies: 222

-- Data for Name: user; Type: TABLE DATA; Schema: public; Owner: postgres

--

COPY public."user" (id, username, password\_hash, role) FROM stdin;

1 admin pbkdf2:sha256:1000000$xYL1TxFvoy34b8YS$827b5b8232b977f10ddf76e02ead735aabc1ac23b876cc7e6899c81ce7c012b7 admin

2 reg pbkdf2:sha256:1000000$YXLU478G0y84uYXw$4e86064a88bd8e9115aed5c44634830b6f819744a31d3ef5565d2a5455aec96a registrar

3 trener pbkdf2:sha256:1000000$TWpTdkNgLMpwCBcp$7bc55a0579dcdd35c797b9953e96e2827dc69a456016457875fc7053cb60ccd5 trener

\.

--

-- TOC entry 3763 (class 0 OID 0)

-- Dependencies: 217

-- Name: client\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.client\_id\_seq', 12, true);

--

-- TOC entry 3764 (class 0 OID 0)

-- Dependencies: 211

-- Name: equipment\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.equipment\_id\_seq', 4, true);

--

-- TOC entry 3765 (class 0 OID 0)

-- Dependencies: 209

-- Name: room\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.room\_id\_seq', 4, true);

--

-- TOC entry 3766 (class 0 OID 0)

-- Dependencies: 215

-- Name: subscription\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.subscription\_id\_seq', 12, true);

--

-- TOC entry 3767 (class 0 OID 0)

-- Dependencies: 219

-- Name: training\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.training\_id\_seq', 14, true);

--

-- TOC entry 3768 (class 0 OID 0)

-- Dependencies: 213

-- Name: trener\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.trener\_id\_seq', 4, true);

--

-- TOC entry 3769 (class 0 OID 0)

-- Dependencies: 221

-- Name: user\_id\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public.user\_id\_seq', 3, true);

--

-- TOC entry 3579 (class 2606 OID 16433)

-- Name: client client\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.client

ADD CONSTRAINT client\_pkey PRIMARY KEY (id);

--

-- TOC entry 3573 (class 2606 OID 16400)

-- Name: equipment equipment\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.equipment

ADD CONSTRAINT equipment\_pkey PRIMARY KEY (id);

--

-- TOC entry 3571 (class 2606 OID 16392)

-- Name: room room\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.room

ADD CONSTRAINT room\_pkey PRIMARY KEY (id);

--

-- TOC entry 3577 (class 2606 OID 16426)

-- Name: subscription subscription\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.subscription

ADD CONSTRAINT subscription\_pkey PRIMARY KEY (id);

--

-- TOC entry 3581 (class 2606 OID 16451)

-- Name: training training\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.training

ADD CONSTRAINT training\_pkey PRIMARY KEY (id);

--

-- TOC entry 3575 (class 2606 OID 16412)

-- Name: trener trener\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.trener

ADD CONSTRAINT trener\_pkey PRIMARY KEY (id);

--

-- TOC entry 3583 (class 2606 OID 16478)

-- Name: user user\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."user"

ADD CONSTRAINT user\_pkey PRIMARY KEY (id);

--

-- TOC entry 3585 (class 2606 OID 16480)

-- Name: user user\_username\_key; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."user"

ADD CONSTRAINT user\_username\_key UNIQUE (username);

--

-- TOC entry 3590 (class 2606 OID 16501)

-- Name: client client\_main\_room\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.client

ADD CONSTRAINT client\_main\_room\_id\_fkey FOREIGN KEY (main\_room\_id) REFERENCES public.room(id);

--

-- TOC entry 3591 (class 2606 OID 16439)

-- Name: client client\_sub\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.client

ADD CONSTRAINT client\_sub\_id\_fkey FOREIGN KEY (sub\_id) REFERENCES public.subscription(id);

--

-- TOC entry 3592 (class 2606 OID 16434)

-- Name: client client\_trener\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.client

ADD CONSTRAINT client\_trener\_id\_fkey FOREIGN KEY (trener\_id) REFERENCES public.trener(id);

--

-- TOC entry 3586 (class 2606 OID 16401)

-- Name: equipment equipment\_room\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.equipment

ADD CONSTRAINT equipment\_room\_id\_fkey FOREIGN KEY (room\_id) REFERENCES public.room(id);

--

-- TOC entry 3588 (class 2606 OID 16496)

-- Name: subscription fk\_subscription\_client; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.subscription

ADD CONSTRAINT fk\_subscription\_client FOREIGN KEY (client\_id) REFERENCES public.client(id);

--

-- TOC entry 3589 (class 2606 OID 16486)

-- Name: subscription subscription\_client\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.subscription

ADD CONSTRAINT subscription\_client\_id\_fkey FOREIGN KEY (client\_id) REFERENCES public.client(id);

--

-- TOC entry 3593 (class 2606 OID 16452)

-- Name: training training\_client\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.training

ADD CONSTRAINT training\_client\_id\_fkey FOREIGN KEY (client\_id) REFERENCES public.client(id);

--

-- TOC entry 3594 (class 2606 OID 16467)

-- Name: training training\_room\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.training

ADD CONSTRAINT training\_room\_id\_fkey FOREIGN KEY (room\_id) REFERENCES public.room(id);

--

-- TOC entry 3595 (class 2606 OID 16457)

-- Name: training training\_sub\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.training

ADD CONSTRAINT training\_sub\_id\_fkey FOREIGN KEY (sub\_id) REFERENCES public.subscription(id);

--

-- TOC entry 3596 (class 2606 OID 16462)

-- Name: training training\_trener\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.training

ADD CONSTRAINT training\_trener\_id\_fkey FOREIGN KEY (trener\_id) REFERENCES public.trener(id);

--

-- TOC entry 3587 (class 2606 OID 16413)

-- Name: trener trener\_main\_room\_id\_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.trener

ADD CONSTRAINT trener\_main\_room\_id\_fkey FOREIGN KEY (main\_room\_id) REFERENCES public.room(id);

--

-- TOC entry 3755 (class 0 OID 0)

-- Dependencies: 4

-- Name: SCHEMA public; Type: ACL; Schema: -; Owner: kolyhov

--

REVOKE USAGE ON SCHEMA public FROM PUBLIC;

GRANT ALL ON SCHEMA public TO PUBLIC;

-- Completed on 2025-04-21 13:23:36 EEST

--

-- PostgreSQL database dump complete

--

-- Completed on 2025-04-21 13:23:36 EEST

--

-- PostgreSQL database cluster dump complete

--

# ДОДАТОК Б. ФРОНТЕНД ЧАСТИНА ВЕБ-ЫНТЕРФЕЙСУ

Лістинг Б.1 - login.html

<!DOCTYPE html>

<html>

<head>

<title>Вход в систему</title>

<link rel="stylesheet" href="/static/style.css">

</head>

<body>

<div class="container">

<h2>Вход</h2>

<form method="POST">

<label>Логин:</label><br>

<input type="text" name="username" style="height:1.2rem;"><br>

<label>Пароль:</label><br>

<input type="password" name="password" style="height:1.2rem;"><br><br>

<input type="submit" value="Войти"">

</form>

{% if error %}

<p class="error">{{ error }}</p>

{% endif %}

</div>

</body>

</html>

Лістинг Б.2 - admin.html

<!DOCTYPE html>

<html>

<head>

<title>Панель администратора</title>

<link rel="stylesheet" href="/static/style.css">

<style>

.styled-table {

border-collapse: collapse;

margin: 20px 0;

font-size: 1.05em;

min-width: 700px;

width: 100%;

background: #fff;

border-radius: 10px 10px 6px 6px;

overflow: hidden;

box-shadow: 0 2px 16px rgba(0,0,0,0.08);

}

.styled-table thead tr {

background-color: #2d3a4b;

color: #fff;

text-align: left;

font-weight: bold;

}

.styled-table th, .styled-table td {

padding: 14px 18px;

border-bottom: 1px solid #e3e3e3;

}

.styled-table tbody tr {

border-bottom: 1px solid #e3e3e3;

transition: background 0.2s;

}

.styled-table tbody tr:hover {

background: #f4f6fb;

}

.styled-table td {

vertical-align: middle;

}

@media (max-width: 900px) {

.styled-table, .styled-table thead, .styled-table tbody, .styled-table th, .styled-table td, .styled-table tr {

display: block;

}

.styled-table thead tr {

display: none;

}

.styled-table td {

padding: 10px 8px;

border: none;

position: relative;

}

.styled-table td:before {

content: attr(data-label);

font-weight: bold;

display: block;

margin-bottom: 4px;

color: #4f8cff;

}

}

#modalRoom, #modalTrener, #modalEquipment {

position: fixed !important;

top: 0; left: 0; width: 100vw; height: 100vh;

display: flex;

align-items: center;

justify-content: center;

background: rgba(0,0,0,0.35);

z-index: 3000 !important;

}

</style>

</head>

<body>

<div class="container">

<h2>Добро пожаловать, Администратор!</h2>

<button id="btnAddRoom">Добавить зал</button>

<button id="btnAddTrener">Добавить тренера</button>

<button id="btnAddEquipment">Добавить оборудование</button>

<button id="btnShowClients">Все клиенты</button>

<button id="btnShowSubs">Все абонементы</button>

<button id="btnShowTrainings">Все тренировки</button>

<a href="/logout">Выйти</a>

<div style="overflow-x:auto; margin-top:30px;">

<h3>Залы</h3>

<table class="styled-table">

<thead>

<tr><th>ID</th><th>Адрес</th><th>Вместимость</th><th>Действия</th></tr>

</thead>

<tbody>

{% for r in rooms %}

<tr>

<td>{{ r.id }}</td>

<td>{{ r.address }}</td>

<td>{{ r.capacity }}</td>

<td>

<button class="btnEditRoom" data-id="{{ r.id }}" data-address="{{ r.address }}" data-capacity="{{ r.capacity }}">Редактировать</button>

<a href="/admin/room/delete/{{ r.id }}" onclick="return confirm('Удалить?')">Удалить</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

<h3 style="margin-top:40px;">Тренеры</h3>

<table class="styled-table">

<thead>

<tr><th>ID</th><th>Имя</th><th>Телефон</th><th>Специализация</th><th>Основной зал</th><th>Действия</th></tr>

</thead>

<tbody>

{% for t in trainers %}

<tr>

<td>{{ t.id }}</td>

<td>{{ t.name }}</td>

<td>{{ t.phone }}</td>

<td>{{ t.spec }}</td>

<td>{% for r in rooms %}{% if r.id == t.main\_room\_id %}{{ r.address }}{% endif %}{% endfor %}</td>

<td>

<button class="btnEditTrener" data-id="{{ t.id }}" data-name="{{ t.name }}" data-phone="{{ t.phone }}" data-spec="{{ t.spec }}" data-main\_room\_id="{{ t.main\_room\_id }}">Редактировать</button>

<a href="/admin/trener/delete/{{ t.id }}" onclick="return confirm('Удалить?')">Удалить</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

<h3 style="margin-top:40px;">Оборудование</h3>

<table class="styled-table">

<thead>

<tr><th>ID</th><th>Название</th><th>Состояние</th><th>Зал</th><th>Действия</th></tr>

</thead>

<tbody>

{% for e in equipment %}

<tr>

<td>{{ e.id }}</td>

<td>{{ e.name }}</td>

<td>{{ e.state }}</td>

<td>{% for r in rooms %}{% if r.id == e.room\_id %}{{ r.address }}{% endif %}{% endfor %}{% if not e.room\_id %}<span style="color:#888;">нет</span>{% endif %}</td>

<td>

<button class="btnEditEquipment" data-id="{{ e.id }}" data-name="{{ e.name }}" data-state="{{ e.state }}" data-room\_id="{{ e.room\_id }}">Редактировать</button>

<a href="/admin/equipment/delete/{{ e.id }}" onclick="return confirm('Удалить?')">Удалить</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

<!-- Модалка: Добавить/Редактировать зал -->

<div id="modalRoom" class="modal" style="display:none;">

<div class="modal-content">

<span class="close" id="closeRoom">&times;</span>

<h3 id="roomTitle">Новый зал</h3>

<form id="formRoom" method="POST">

<label>Адрес:</label>

<input type="text" name="address" required>

<label>Вместимость:</label>

<input type="number" name="capacity" required>

<input type="submit" id="submitRoom" value="Создать зал">

</form>

</div>

</div>

<!-- Модалка: Добавить/Редактировать тренера -->

<div id="modalTrener" class="modal" style="display:none;">

<div class="modal-content">

<span class="close" id="closeTrener">&times;</span>

<h3 id="trenerTitle">Новый тренер</h3>

<form id="formTrener" method="POST">

<label>Имя:</label>

<input type="text" name="name" required>

<label>Телефон:</label>

<input type="text" name="phone" maxlength="12" required>

<label>Специализация:</label>

<input type="text" name="spec" required>

<label>Основной зал:</label>

<select name="main\_room\_id" required>

<option value="">— выбери зал —</option>

{% for r in rooms %}

<option value="{{ r.id }}">{{ r.address }}</option>

{% endfor %}

</select>

<input type="submit" id="submitTrener" value="Создать тренера">

</form>

</div>

</div>

<!-- Модалка: Добавить/Редактировать оборудование -->

<div id="modalEquipment" class="modal" style="display:none;">

<div class="modal-content">

<span class="close" id="closeEquipment">&times;</span>

<h3 id="equipmentTitle">Новое оборудование</h3>

<form id="formEquipment" method="POST">

<label>Название:</label>

<input type="text" name="name" required>

<label>Состояние:</label>

<input type="text" name="state" required>

<label>Зал:</label>

<select name="room\_id">

<option value="">— без зала —</option>

{% for r in rooms %}

<option value="{{ r.id }}">{{ r.address }}</option>

{% endfor %}

</select>

<input type="submit" id="submitEquipment" value="Создать оборудование">

</form>

</div>

</div>

<!-- Модалка: Все клиенты -->

<div id="modalAllClients" class="modal" style="display:none;position:fixed;top:0;left:0;width:100vw;height:100vh;z-index:2000;align-items:center;justify-content:center;background:rgba(0,0,0,0.35);">

<div class="modal-content" style="background:#fff;min-width:600px;max-width:95vw;max-height:90vh;overflow:auto;position:relative;z-index:2100;">

<span class="close" id="closeAllClients" style="position:absolute;right:16px;top:12px;font-size:28px;cursor:pointer;">&times;</span>

<h3>Все клиенты</h3>

<input type="text" id="searchAllClients" placeholder="Поиск клиента..." style="margin-bottom:16px;width:100%;max-width:350px;">

<div style="overflow-x:auto;">

<table class="styled-table">

<thead>

<tr><th>ID</th><th>Имя</th><th>Телефон</th><th>Зал</th><th>Действия</th></tr>

</thead>

<tbody id="allClientsTableBody">

{% for c in clients %}

<tr>

<td>{{ c.id }}</td>

<td>{{ c.name }}</td>

<td>{{ c.phone }}</td>

<td>{% for r in rooms %}{% if r.id == c.main\_room\_id %}{{ r.address }}{% endif %}{% endfor %}</td>

<td><a href="/admin/client/delete/{{ c.id }}" onclick="return confirm('Удалить клиента?')">Удалить</a></td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

<!-- Модалка: Все абонементы -->

<div id="modalAllSubs" class="modal" style="display:none;position:fixed;top:0;left:0;width:100vw;height:100vh;z-index:2000;align-items:center;justify-content:center;background:rgba(0,0,0,0.35);">

<div class="modal-content" style="background:#fff;min-width:600px;max-width:95vw;max-height:90vh;overflow:auto;position:relative;z-index:2100;">

<span class="close" id="closeAllSubs" style="position:absolute;right:16px;top:12px;font-size:28px;cursor:pointer;">&times;</span>

<h3>Все абонементы</h3>

<input type="text" id="searchAllSubs" placeholder="Поиск абонемента..." style="margin-bottom:16px;width:100%;max-width:350px;">

<div style="overflow-x:auto;">

<table class="styled-table">

<thead>

<tr><th>ID</th><th>Клиент</th><th>Дата начала</th><th>Длительность</th><th>Осталось занятий</th><th>Действия</th></tr>

</thead>

<tbody id="allSubsTableBody">

{% for s in subs %}

<tr>

<td>{{ s.id }}</td>

<td>{% for c in clients %}{% if c.id == s.client\_id %}{{ c.name }}{% endif %}{% endfor %}</td>

<td>{{ s.start\_date }}</td>

<td>{{ s.duration }}</td>

<td>{{ s.workout }}</td>

<td><a href="/admin/sub/delete/{{ s.id }}" onclick="return confirm('Удалить абонемент?')">Удалить</a></td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

<!-- Модалка: Все тренировки -->

<div id="modalAllTrainings" class="modal" style="display:none;position:fixed;top:0;left:0;width:100vw;height:100vh;z-index:2000;align-items:center;justify-content:center;background:rgba(0,0,0,0.35);">

<div class="modal-content" style="background:#fff;min-width:700px;max-width:98vw;max-height:90vh;overflow:auto;position:relative;z-index:2100;">

<span class="close" id="closeAllTrainings" style="position:absolute;right:16px;top:12px;font-size:28px;cursor:pointer;">&times;</span>

<h3>Все тренировки</h3>

<input type="text" id="searchAllTrainings" placeholder="Поиск тренировки..." style="margin-bottom:16px;width:100%;max-width:350px;">

<div style="overflow-x:auto;">

<table class="styled-table">

<thead>

<tr><th>ID</th><th>Клиент</th><th>Тренер</th><th>Зал</th><th>Дата</th><th>Время</th><th>Длительность</th><th>Действия</th></tr>

</thead>

<tbody id="allTrainingsTableBody">

{% for t in trainings %}

<tr>

<td>{{ t.id }}</td>

<td>{% for c in clients %}{% if c.id == t.client\_id %}{{ c.name }}{% endif %}{% endfor %}</td>

<td>{% for tr in trainers %}{% if tr.id == t.trener\_id %}{{ tr.name }}{% endif %}{% endfor %}</td>

<td>{% for r in rooms %}{% if r.id == t.room\_id %}{{ r.address }}{% endif %}{% endfor %}</td>

<td>{{ t.date\_time.strftime('%d.%m.%Y') }}</td>

<td>{{ t.date\_time.strftime('%H:%M') }}</td>

<td>{{ t.duration }} мин</td>

<td><a href="/admin/training/delete/{{ t.id }}" onclick="return confirm('Удалить тренировку?')">Удалить</a></td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

<script>

function toggleModal(id, show) {

document.getElementById(id).style.display = show ? 'flex' : 'none';

}

document.getElementById('btnAddRoom').onclick = ()=> {

document.getElementById('roomTitle').innerText = 'Новый зал';

document.getElementById('formRoom').action = '/admin/room/create';

document.getElementById('submitRoom').value = 'Создать зал';

document.getElementById('formRoom').reset();

toggleModal('modalRoom', true);

};

document.getElementById('closeRoom').onclick = ()=> toggleModal('modalRoom', false);

document.querySelectorAll('.btnEditRoom').forEach(btn=>{

btn.onclick = ()=> {

document.getElementById('roomTitle').innerText = 'Редактировать зал';

document.getElementById('formRoom').action = `/admin/room/edit/${btn.dataset.id}`;

document.getElementById('submitRoom').value = 'Сохранить';

document.querySelector('#formRoom input[name=address]').value = btn.dataset.address;

document.querySelector('#formRoom input[name=capacity]').value = btn.dataset.capacity;

toggleModal('modalRoom', true);

};

});

document.getElementById('btnAddTrener').onclick = ()=> {

document.getElementById('trenerTitle').innerText = 'Новый тренер';

document.getElementById('formTrener').action = '/admin/trener/create';

document.getElementById('submitTrener').value = 'Создать тренера';

document.getElementById('formTrener').reset();

toggleModal('modalTrener', true);

};

document.getElementById('closeTrener').onclick = ()=> toggleModal('modalTrener', false);

document.querySelectorAll('.btnEditTrener').forEach(btn=>{

btn.onclick = ()=> {

document.getElementById('trenerTitle').innerText = 'Редактировать тренера';

document.getElementById('formTrener').action = `/admin/trener/edit/${btn.dataset.id}`;

document.getElementById('submitTrener').value = 'Сохранить';

document.querySelector('#formTrener input[name=name]').value = btn.dataset.name;

document.querySelector('#formTrener input[name=phone]').value = btn.dataset.phone;

document.querySelector('#formTrener input[name=spec]').value = btn.dataset.spec;

document.querySelector('#formTrener select[name=main\_room\_id]').value = btn.dataset.main\_room\_id;

toggleModal('modalTrener', true);

};

});

document.getElementById('btnAddEquipment') && (document.getElementById('btnAddEquipment').onclick = ()=> {

document.getElementById('equipmentTitle').innerText = 'Новое оборудование';

document.getElementById('formEquipment').action = '/admin/equipment/create';

document.getElementById('submitEquipment').value = 'Создать оборудование';

document.getElementById('formEquipment').reset();

toggleModal('modalEquipment', true);

});

document.getElementById('closeEquipment') && (document.getElementById('closeEquipment').onclick = ()=> toggleModal('modalEquipment', false));

document.querySelectorAll('.btnEditEquipment').forEach(btn=>{

btn.onclick = ()=> {

document.getElementById('equipmentTitle').innerText = 'Редактировать оборудование';

document.getElementById('formEquipment').action = `/admin/equipment/edit/${btn.dataset.id}`;

document.getElementById('submitEquipment').value = 'Сохранить';

document.querySelector('#formEquipment input[name=name]').value = btn.dataset.name;

document.querySelector('#formEquipment input[name=state]').value = btn.dataset.state;

document.querySelector('#formEquipment select[name=room\_id]').value = btn.dataset.room\_id || '';

toggleModal('modalEquipment', true);

};

});

document.getElementById('btnShowClients').onclick = ()=> toggleModal('modalAllClients', true);

document.getElementById('closeAllClients').onclick = ()=> toggleModal('modalAllClients', false);

document.getElementById('btnShowSubs').onclick = ()=> toggleModal('modalAllSubs', true);

document.getElementById('closeAllSubs').onclick = ()=> toggleModal('modalAllSubs', false);

document.getElementById('searchAllClients').addEventListener('input', function() {

const f = this.value.toLowerCase();

document.querySelectorAll('#allClientsTableBody tr').forEach(r=>{

r.style.display = r.innerText.toLowerCase().includes(f) ? '' : 'none';

});

});

document.getElementById('searchAllSubs').addEventListener('input', function() {

const f = this.value.toLowerCase();

document.querySelectorAll('#allSubsTableBody tr').forEach(r=>{

r.style.display = r.innerText.toLowerCase().includes(f) ? '' : 'none';

});

});

document.getElementById('btnShowTrainings') && (document.getElementById('btnShowTrainings').onclick = ()=> toggleModal('modalAllTrainings', true));

document.getElementById('closeAllTrainings') && (document.getElementById('closeAllTrainings').onclick = ()=> toggleModal('modalAllTrainings', false));

document.getElementById('searchAllTrainings') && document.getElementById('searchAllTrainings').addEventListener('input', function() {

const f = this.value.toLowerCase();

document.querySelectorAll('#allTrainingsTableBody tr').forEach(r=>{

r.style.display = r.innerText.toLowerCase().includes(f) ? '' : 'none';

});

});

window.onclick = function(event) {

if(event.target.classList.contains('modal')) {

event.target.style.display = 'none';

}

}

</script>

</body>

</html>

Лістинг Б.3 - registrar.html

<!DOCTYPE html>

<html>

<head>

<title>Панель регистратора</title>

<link rel="stylesheet" href="/static/style.css">

<style>

</style>

</head>

<body>

<div class="container">

<h2>Добро пожаловать, Регистратор!</h2>

{% with messages = get\_flashed\_messages() %}

{% if messages %}

<div class="error">{{ messages[0] }}</div>

{% endif %}

{% endwith %}

<button id="btnAddClient">Добавить клиента</button>

<button id="btnAddSub">Добавить абонемент</button>

<button id="btnAddTraining">Добавить тренировку</button>

<a href="/logout">Выйти</a>

<input type="text" id="clientSearch" placeholder="Поиск клиента..." style="margin:20px 0;width:100%;max-width:400px;">

<div style="overflow-x:auto;">

<table id="clientsTable" class="styled-table">

<thead>

<tr><th>ID</th><th>Имя</th><th>Телефон</th><th>Тренер</th><th>Абонемент</th><th>Действия</th></tr>

</thead>

<tbody>

{% for c in clients %}

<tr data-id="{{ c.id }}" data-name="{{ c.name }}" data-phone="{{ c.phone }}" data-reg-date="{{ c.reg\_date }}" data-trener-id="{{ c.trener\_id }}" data-sub-id="{{ c.sub\_id }}" data-main-room-id="{{ c.main\_room\_id }}">

<td>{{ c.id }}</td>

<td class="td-name">{{ c.name }}</td>

<td class="td-phone">{{ c.phone }}</td>

<td class="td-trener">{% for t in trainers %}{% if t.id == c.trener\_id %}{{ t.name }}{% endif %}{% endfor %}</td>

<td class="td-sub">{% for s in subs %}{% if s.id == c.sub\_id %}{{ s.start\_date }}{% endif %}{% endfor %}</td>

<td>

<button class="btnEditClient">Редактировать</button>

<a href="/registrar/client/delete/{{ c.id }}" onclick="return confirm('Удалить?')">Удалить</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

<div id="modalClient" class="modal" style="display:none;">

<div class="modal-content">

<span class="close" id="closeClient">&times;</span>

<h3 id="clientTitle">Новый клиент</h3>

<form id="formClient" method="POST">

<label>Имя:</label>

<input type="text" name="name" required>

<label>Телефон (11-12 цифр):</label>

<input type="text" name="phone" maxlength="12" pattern="\d{11,12}" title="11 или 12 цифр, без пробелов" required>

<label>Дата регистрации:</label>

<input type="date" name="reg\_date" required>

<label>Основной зал:</label>

<select name="main\_room\_id" id="roomSelect" required>

<option value="">— выбери зал —</option>

{% for r in rooms %}

<option value="{{ r.id }}">{{ r.address }} (вместимость: {{ r.capacity }})</option>

{% endfor %}

</select>

<label>Тренер:</label>

<select name="trener\_id" id="trainerSelect" disabled>

<option value="">— выбери тренера —</option>

{% for t in trainers %}

<option value="{{ t.id }}" data-room-id="{{ t.main\_room\_id }}">{{ t.name }}</option>

{% endfor %}

</select>

<input type="submit" id="submitClient" value="Создать клиента">

</form>

<!-- Таблица абонементов клиента -->

<div id="clientSubsTable" style="display:none;margin:30px 0;">

<h4>Абонементы клиента</h4>

<table class="styled-table">

<thead>

<tr><th>ID</th><th>Дата начала</th><th>Длительность</th><th>Кол-во занятий</th><th></th></tr>

</thead>

<tbody id="subsTableBody">

</tbody>

</table>

</div>

</div>

</div>

<!-- Модалка: Добавить абонемент -->

<div id="modalSub" class="modal" style="display:none;">

<div class="modal-content">

<span class="close" id="closeSub">&times;</span>

<h3>Новый абонемент</h3>

<form action="/registrar/sub/create" method="POST">

<label>Дата начала:</label>

<input type="date" name="start\_date" required>

<label>Длительность (в днях):</label>

<input type="number" name="duration" required>

<label>Количество занятий:</label>

<input type="number" name="workout" required>

<label>Клиент:</label>

<select name="client\_id" required>

<option value="">— выбери клиента —</option>

{% for c in clients %}

<option value="{{ c.id }}">{{ c.name }} (тел: {{ c.phone }})</option>

{% endfor %}

</select>

<input type="submit" value="Создать абонемент">

</form>

</div>

</div>

<!-- Модалка: Редактировать абонемент -->

<div id="modalEditSub" class="modal" style="display:none;">

<div class="modal-content">

<span class="close" id="closeEditSub">&times;</span>

<h3>Редактировать абонемент</h3>

<form id="formEditSub" method="POST">

<input type="hidden" name="sub\_id">

<label>Дата начала:</label>

<input type="date" name="start\_date" required>

<label>Длительность (в днях):</label>

<input type="number" name="duration" required>

<label>Количество занятий:</label>

<input type="number" name="workout" required>

<input type="submit" value="Сохранить">

</form>

</div>

</div>

<!-- Модалка: Добавить тренировку -->

<div id="modalTraining" class="modal" style="display:none;">

<div class="modal-content">

<span class="close" id="closeTraining">&times;</span>

<h3>Новая тренировка</h3>

<form id="formTraining" method="POST" action="/registrar/training/create">

<label>Клиент:</label>

<select name="client\_id" id="trainingClientSelect" required>

<option value="">— выбери клиента —</option>

{% for c in clients %}

<option value="{{ c.id }}" data-room-id="{{ c.main\_room\_id }}" data-trener-id="{{ c.trener\_id }}">{{ c.name }} ({{ c.phone }})</option>

{% endfor %}

</select>

<label>Тренер:</label>

<div id="trainingTrenerRadios" style="margin-bottom:16px;"></div>

<input type="hidden" name="trener\_id" id="trainingTrenerHidden" required>

<label>Абонемент:</label>

<div id="trainingSubRadios" style="margin-bottom:16px;"></div>

<input type="hidden" name="sub\_id" id="trainingSubHidden" required>

<label>Дата:</label>

<input type="date" name="date" id="trainingDate" required>

<label>Время:</label>

<input type="time" name="time" id="trainingTime" required>

<label>Длительность (мин):</label>

<select name="duration" required>

<option value="30">30</option>

<option value="60">60</option>

<option value="90">90</option>

<option value="120">120</option>

<option value="150">150</option>

</select>

<input type="submit" value="Создать тренировку">

</form>

</div>

</div>

<div class="container" style="margin-top:40px;">

<h3 style="color:#222; background:none; border-radius:0; padding:0; margin:0; margin-bottom: 2.5rem; font-size:1.4em;">Тренировки</h3>

<div style="padding:0; max-width:900px; margin:auto; background:none; box-shadow:none; border-radius:0;">

<div style="margin-bottom:18px;">

<label for="roomFilter" style="font-weight:500; color:#222; background:none; border-radius:0; padding:0; font-size:1em;">Показать тренировки зала:</label>

<select id="roomFilter" style="padding:7px 14px;border-radius:6px;font-size:1em; margin-left:10px;">

<option value="all">Все залы</option>

{% for r in rooms %}

<option value="{{ r.id }}">{{ r.address }}</option>

{% endfor %}

</select>

</div>

<div style="overflow-x:auto;">

<table class="styled-table" id="trainingsTable">

<thead>

<tr>

<th>Клиент</th>

<th>Тренер</th>

<th>Зал</th>

<th>Дата</th>

<th>Время</th>

<th>Длительность</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

{% for t in trainings|sort(attribute='room\_id') %}

<tr data-room-id="{{ t.room\_id }}">

<td>{% for c in clients %}{% if c.id == t.client\_id %}{{ c.name }}{% endif %}{% endfor %}</td>

<td>{% for tr in trainers %}{% if tr.id == t.trener\_id %}{{ tr.name }}{% endif %}{% endfor %}</td>

<td>{% for r in rooms %}{% if r.id == t.room\_id %}{{ r.address }}{% endif %}{% endfor %}</td>

<td>{{ t.date\_time.strftime('%d.%m.%Y') }}</td>

<td>{{ t.date\_time.strftime('%H:%M') }}</td>

<td>{{ t.duration }} мин</td>

<td>

<button class="btnEditTraining" data-id="{{ t.id }}">Редактировать</button>

<a href="/registrar/training/delete/{{ t.id }}" onclick="return confirm('Удалить тренировку?')">Удалить</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

<!-- Модалка: Редактировать тренировку -->

<div id="modalEditTraining" class="modal" style="display:none;">

<div class="modal-content">

<span class="close" id="closeEditTraining">&times;</span>

<h3>Редактировать тренировку</h3>

<form id="formEditTraining" method="POST">

<input type="hidden" name="training\_id">

<label>Дата:</label>

<input type="date" name="date" required>

<label>Время:</label>

<input type="time" name="time" required>

<label>Длительность (мин):</label>

<select name="duration" required>

<option value="30">30</option>

<option value="60">60</option>

<option value="90">90</option>

<option value="120">120</option>

<option value="150">150</option>

</select>

<input type="submit" value="Сохранить">

</form>

</div>

</div>

<style>

.modal { position:fixed; top:0; left:0; width:100vw; height:100vh; background:rgba(0,0,0,0.3); display:flex; align-items:center; justify-content:center; z-index:1000; }

.modal-content { background:#fff; border-radius:10px; padding:32px 24px; min-width:320px; max-width:90vw; box-shadow:0 2px 16px rgba(0,0,0,0.15); position:relative; }

.close { position:absolute; right:16px; top:12px; font-size:28px; cursor:pointer; }

.styled-table {

border-collapse: collapse;

margin: 20px 0;

font-size: 1.05em;

min-width: 700px;

width: 100%;

background: #fff;

border-radius: 10px 10px 6px 6px;

overflow: hidden;

box-shadow: 0 2px 16px rgba(0,0,0,0.08);

}

.styled-table thead tr {

background-color: #2d3a4b;

color: #fff;

text-align: left;

font-weight: bold;

}

.styled-table th, .styled-table td {

padding: 14px 18px;

border-bottom: 1px solid #e3e3e3;

}

.styled-table tbody tr {

border-bottom: 1px solid #e3e3e3;

transition: background 0.2s;

}

.styled-table tbody tr:hover {

background: #f4f6fb;

}

.styled-table td {

vertical-align: middle;

}

@media (max-width: 900px) {

.styled-table, .styled-table thead, .styled-table tbody, .styled-table th, .styled-table td, .styled-table tr {

display: block;

}

.styled-table thead tr {

display: none;

}

.styled-table td {

padding: 10px 8px;

border: none;

position: relative;

}

.styled-table td:before {

content: attr(data-label);

font-weight: bold;

display: block;

margin-bottom: 4px;

color: #4f8cff;

}

}

</style>

<script>

const roomSelect = document.getElementById('roomSelect');

const trainerSelect = document.getElementById('trainerSelect');

// --- Открытие/закрытие модалок ---

function toggleModal(id, show) {

document.getElementById(id).style.display = show ? 'flex' : 'none';

}

document.getElementById('btnAddClient').onclick = ()=> {

document.getElementById('clientTitle').innerText = 'Новый клиент';

document.getElementById('formClient').action = '/registrar/client/create';

document.getElementById('submitClient').value = 'Создать клиента';

document.getElementById('formClient').reset();

trainerSelect.disabled = true;

Array.from(trainerSelect.options).forEach(opt => {

if (opt.value) opt.hidden = true;

});

toggleModal('modalClient', true);

};

document.getElementById('closeClient').onclick = ()=> toggleModal('modalClient', false);

document.getElementById('btnAddSub').onclick = ()=> toggleModal('modalSub', true);

document.getElementById('closeSub').onclick = ()=> toggleModal('modalSub', false);

document.getElementById('btnAddTraining').onclick = ()=> {

document.getElementById('formTraining').reset();

document.getElementById('trainingTrenerRadios').innerHTML = '';

document.getElementById('trainingTrenerHidden').value = '';

document.getElementById('trainingSubRadios').innerHTML = '';

document.getElementById('trainingSubHidden').value = '';

toggleModal('modalTraining', true);

};

// Собираем все абонементы по клиентам для быстрого доступа в JS

const allSubs = {};

{% for c in clients %}

allSubs[{{c.id}}] = [

{% for s in subs if s.client\_id==c.id %}

{id: {{s.id}}, start\_date: '{{s.start\_date}}', duration: {{s.duration}}, workout: {{s.workout}}},

{% endfor %}

];

{% endfor %}

// --- Редактировать клиента ---

document.querySelectorAll('.btnEditClient').forEach(btn=>{

btn.onclick = ()=> {

const tr = btn.closest('tr');

const clientId = tr.dataset.id;

document.querySelector('input[name=name]').value = tr.dataset.name;

document.querySelector('input[name=phone]').value = tr.dataset.phone;

document.querySelector('input[name=reg\_date]').value = tr.dataset.regDate;

// Показываем тренеров по текущему залу клиента

const roomId = tr.dataset.mainRoomId;

roomSelect.value = roomId;

trainerSelect.disabled = false;

Array.from(trainerSelect.options).forEach(opt => {

if (!opt.value) return;

opt.hidden = (opt.dataset.roomId !== roomId);

});

trainerSelect.value = tr.dataset.trenerId;

document.getElementById('clientTitle').innerText = `Редактировать клиента #${clientId}`;

document.getElementById('formClient').action = `/registrar/client/edit/${clientId}`;

document.getElementById('submitClient').value = 'Сохранить изменения';

// --- Таблица абонементов клиента ---

const subsTable = document.getElementById('clientSubsTable');

const subsBody = document.getElementById('subsTableBody');

subsBody.innerHTML = '';

(allSubs[clientId]||[]).forEach(s=>{

subsBody.innerHTML += `<tr><td>${s.id}</td><td>${s.start\_date}</td><td>${s.duration}</td><td>${s.workout}</td><td><button class='btnEditSub' data-subid='${s.id}' data-start='${s.start\_date}' data-duration='${s.duration}' data-workout='${s.workout}'>Редактировать</button></td></tr>`;

});

subsTable.style.display = (allSubs[clientId]||[]).length ? 'block' : 'none';

toggleModal('modalClient', true);

};

});

// --- Открытие модалки редактирования абонемента ---

document.addEventListener('click', function(e) {

if(e.target.classList.contains('btnEditSub')) {

e.preventDefault();

const btn = e.target;

document.querySelector('#formEditSub input[name=sub\_id]').value = btn.dataset.subid;

document.querySelector('#formEditSub input[name=start\_date]').value = btn.dataset.start;

document.querySelector('#formEditSub input[name=duration]').value = btn.dataset.duration;

document.querySelector('#formEditSub input[name=workout]').value = btn.dataset.workout;

document.getElementById('formEditSub').action = `/registrar/sub/edit/${btn.dataset.subid}`;

toggleModal('modalEditSub', true);

}

});

document.getElementById('closeEditSub').onclick = ()=> toggleModal('modalEditSub', false);

// --- Поиск ---

document.getElementById('clientSearch').addEventListener('input', function() {

const f = this.value.toLowerCase();

document.querySelectorAll('#clientsTable tbody tr').forEach(r=>{

r.style.display = r.innerText.toLowerCase().includes(f) ? '' : 'none';

});

});

// --- Закрытие по клику вне модалки ---

window.onclick = function(event) {

if(event.target.classList.contains('modal')) {

event.target.style.display = 'none';

}

}

roomSelect.addEventListener('change', function () {

const selectedRoom = this.value;

trainerSelect.disabled = !selectedRoom;

Array.from(trainerSelect.options).forEach(opt => {

if (!opt.value) return;

opt.hidden = (opt.dataset.roomId !== selectedRoom);

});

trainerSelect.value = '';

});

trainerSelect.disabled = true;

// --- Фильтрация тренеров и абонементов по клиенту и абонементу ---

const clientSelect = document.getElementById('trainingClientSelect');

const trenerRadiosDiv = document.getElementById('trainingTrenerRadios');

const trenerHidden = document.getElementById('trainingTrenerHidden');

const subRadiosDiv = document.getElementById('trainingSubRadios');

const subHidden = document.getElementById('trainingSubHidden');

// Собираем тренеров по залам

const allTrainers = [

{% for t in trainers %}

{id: {{ t.id }}, name: '{{ t.name }}', main\_room\_id: '{{ t.main\_room\_id }}'},

{% endfor %}

];

function updateSubRadios(clientId) {

const subs = allSubs[clientId] || [];

subRadiosDiv.innerHTML = '';

let selectedSub = '';

if (subs.length) {

// Выбираем первый с workout > 0

for (let s of subs) {

if (parseInt(s.workout,10) > 0) { selectedSub = s.id; break; }

}

if (!selectedSub) selectedSub = subs[0].id;

subs.forEach(s => {

if (parseInt(s.workout,10) <= 0) return;

const checked = (selectedSub && s.id == selectedSub) ? 'checked' : '';

subRadiosDiv.innerHTML += `<label style='margin-right:18px;'><input type='radio' name='sub\_radio' value='${s.id}' ${checked}> №${s.id} (осталось: ${s.workout})</label>`;

});

subHidden.value = selectedSub;

subRadiosDiv.querySelectorAll('input[type=radio][name=sub\_radio]').forEach(radio => {

radio.addEventListener('change', function() {

subHidden.value = this.value;

});

});

} else {

subRadiosDiv.innerHTML = '<span style="color:#888;">Нет абонементов для клиента</span>';

subHidden.value = '';

}

}

function updateTrenerRadios(clientId) {

const clientOption = clientSelect.querySelector(`option[value='${clientId}']`);

const selectedRoom = clientOption?.dataset.roomId;

const mainTrenerId = clientOption?.dataset.trenerId;

const trainers = allTrainers.filter(t => t.main\_room\_id === selectedRoom);

trenerRadiosDiv.innerHTML = '';

let selectedTrener = mainTrenerId || (trainers[0] && trainers[0].id);

if (trainers.length) {

trainers.forEach(t => {

const checked = (selectedTrener && t.id == selectedTrener) ? 'checked' : '';

trenerRadiosDiv.innerHTML += `<label style='margin-right:18px;'><input type='radio' name='trener\_radio' value='${t.id}' ${checked}> ${t.name}</label>`;

});

trenerHidden.value = selectedTrener;

trenerRadiosDiv.querySelectorAll('input[type=radio][name=trener\_radio]').forEach(radio => {

radio.addEventListener('change', function() {

trenerHidden.value = this.value;

});

});

} else {

trenerRadiosDiv.innerHTML = '<span style="color:#888;">Нет тренеров для выбранного зала</span>';

trenerHidden.value = '';

}

}

clientSelect.addEventListener('change', function() {

const clientId = this.value;

updateSubRadios(clientId);

updateTrenerRadios(clientId);

});

subRadiosDiv.addEventListener('change', function(e) {

if (e.target && e.target.name === 'sub\_radio') {

const clientId = clientSelect.value;

updateTrenerRadios(clientId);

}

});

// При открытии модалки всегда выбираем первого клиента, если есть

const btnAddTraining = document.getElementById('btnAddTraining');

if (btnAddTraining) {

btnAddTraining.onclick = ()=> {

document.getElementById('formTraining').reset();

trenerRadiosDiv.innerHTML = '';

trenerHidden.value = '';

subRadiosDiv.innerHTML = '';

subHidden.value = '';

// Если есть клиенты, выбрать первого и обновить тренеров и абонементы

if (clientSelect.options.length > 1) {

clientSelect.selectedIndex = 1;

clientSelect.dispatchEvent(new Event('change'));

}

toggleModal('modalTraining', true);

};

}

// Не даём отправить форму без выбранного тренера и абонемента

const formTraining = document.getElementById('formTraining');

formTraining.addEventListener('submit', function(e) {

if (!trenerHidden.value) {

e.preventDefault();

alert('Выберите тренера!');

}

if (!subHidden.value) {

e.preventDefault();

alert('Выберите абонемент!');

}

});

document.querySelectorAll('.btnEditTraining').forEach(btn => {

btn.onclick = () => {

const tr = btn.closest('tr');

const tId = btn.dataset.id;

// Получаем данные из строки

const date = tr.children[3].innerText.split('.')[2] + '-' + tr.children[3].innerText.split('.')[1] + '-' + tr.children[3].innerText.split('.')[0];

const time = tr.children[4].innerText;

const duration = parseInt(tr.children[5].innerText);

document.querySelector('#formEditTraining input[name=training\_id]').value = tId;

document.querySelector('#formEditTraining input[name=date]').value = date;

document.querySelector('#formEditTraining input[name=time]').value = time;

document.querySelector('#formEditTraining select[name=duration]').value = duration;

document.getElementById('formEditTraining').action = `/registrar/training/edit/${tId}`;

toggleModal('modalEditTraining', true);

};

});

document.getElementById('closeEditTraining').onclick = ()=> toggleModal('modalEditTraining', false);

document.getElementById('roomFilter').addEventListener('change', function() {

const val = this.value;

document.querySelectorAll('#trainingsTable tbody tr').forEach(tr => {

tr.style.display = (val === 'all' || tr.dataset.roomId === val) ? '' : 'none';

});

});

</script>

</body>

</html>

Лістинг Б.4 - trener.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Панель тренера</title>

<link rel="stylesheet" href="/static/style.css">

<style>

body {

margin: 0;

font-family: 'Segoe UI', sans-serif;

}

.topbar {

padding: 32px 0 24px 0;

margin-bottom: 40px;

box-shadow: 0 2px 16px rgba(0, 0, 0, 0.08);

}

.topbar-inner {

color: white;

max-width: 1000px;

margin: 0 auto;

display: flex;

align-items: center;

justify-content: space-between;

gap: 32px;

}

.topbar span {

font-size: 2.1em;

font-weight: 700;

letter-spacing: 1.5px;

}

.topbar form {

display: flex;

align-items: center;

gap: 18px;

}

.topbar a {

font-weight: 600;

font-size: 1.1em;

text-decoration: none;

transition: color .2s;

}

.topbar a:hover {

text-decoration: underline;

}

.container {

max-width: 1000px;

margin: 0 auto;

border-radius: 16px;

box-shadow: 0 2px 16px rgba(0, 0, 0, 0.1);

padding: 36px;

}

h2 {

text-align: center;

margin-bottom: 30px;

font-size: 2em;

font-weight: 700;

letter-spacing: 1px;

}

table.styled-table {

width: 100%;

border-collapse: collapse;

border-radius: 12px;

overflow: hidden;

}

.styled-table thead tr {

font-weight: bold;

text-align: left;

}

.styled-table th,

.styled-table td {

padding: 14px 20px;

font-size: 1.05em;

}

.styled-table td {

vertical-align: middle;

}

.styled-table td:first-child {

font-weight: 600;

}

.styled-table td[colspan] {

text-align: center;

color: #888;

font-size: 1.1em;

}

</style>

</head>

<body>

<div class="topbar">

<div class="topbar-inner">

<span>Расписание тренера</span>

<form id="trenerSelectForm">

<label for="trenerSelectDropdown" style="font-size:1.1em;">Тренер:</label>

<select id="trenerSelectDropdown">

{% for t in all\_trainers %}

<option value="{{ t.id }}" {% if t.id == selected\_trener.id %}selected{% endif %}>{{ t.name }}</option>

{% endfor %}

</select>

</form>

<a href="/logout">Выйти</a>

</div>

</div>

<div class="container">

<h2>Тренировки тренера: <span id="trenerName">{{ selected\_trener.name }}</span></h2>

<div style="overflow-x:auto;">

<table class="styled-table">

<thead>

<tr>

<th>Клиент</th>

<th>Дата</th>

<th>Время</th>

<th>Длительность</th>

</tr>

</thead>

<tbody id="trainingsTableBody">

{% for t in trainings %}

<tr>

<td>{% for c in clients %}{% if c.id == t.client\_id %}{{ c.name }}{% endif %}{% endfor %}</td>

<td>{{ t.date\_time.strftime('%d.%m.%Y') }}</td>

<td>{{ t.date\_time.strftime('%H:%M') }}</td>

<td>{{ t.duration }} мин</td>

</tr>

{% else %}

<tr><td colspan="4">Нет тренировок</td></tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

<script>

document.getElementById('trenerSelectDropdown').onchange = function() {

const trenerId = this.value;

window.location = '/trener?trener\_id=' + trenerId;

};

</script>

</body>

</html>

Лістинг Б.5 - style.css

/\* Основные цвета \*/

:root {

--primary: #1a2980; /\* Глубокий синий \*/

--secondary: #2d3a4b; /\* Тёмно-серый \*/

--white: #f9f9f9; /\* Светлый фон \*/

--background: #1e2a5b; /\* Тёмно-синий фон \*/

--text: #3e3c3c; /\* Светлый текст \*/

--shadow: 0 4px 16px rgba(0,0,0,0.15);

}

body {

background: var(--background);

color: var(--text);

font-family: 'Inter', sans-serif;

margin: 0;

min-height: 100vh;

}

/\* Контейнеры \*/

.container {

max-width: 1100px;

margin: 40px auto;

background: var(--white);

border-radius: 16px;

padding: 32px;

box-shadow: var(--shadow);

}

h2, h3 {

color: var(--primary);

font-weight: 600;

margin-bottom: 24px;

}

button, a {

background: var(--primary);

color: white;

border: none;

padding: 12px 24px;

border-radius: 8px;

transition: 0.2s;

cursor: pointer;

text-decoration: none;

font-size: 1rem;

margin-bottom: 1.5rem;

}

button:hover, a:hover {

background: #152066;

}

/\* Модальные окна \*/

.modal {

background: rgba(0,0,0,0.8);

display: none;

align-items: center;

justify-content: center;

}

.modal-content {

background: var(--white);

border-radius: 16px;

padding: 32px;

max-width: 80%;

max-height: 90%;

overflow-y: auto;

box-shadow: var(--shadow);

}

.close {

color: var(--primary);

font-weight: bold;

position: absolute;

top: 16px;

right: 16px;

cursor: pointer;

}

/\* Таблицы \*/

.styled-table {

width: 100%;

border-collapse: collapse;

margin: 24px 0;

background: var(--white);

border-radius: 12px;

overflow: hidden;

box-shadow: var(--shadow);

}

.styled-table thead {

background: var(--primary);

color: white;

}

.styled-table th, .styled-table td {

padding: 16px;

border-bottom: 1px solid #e4e4e4;

}

.styled-table tbody tr:hover {

background: #eef2ff;

}

.styled-table td:first-child {

font-weight: 500;

}

.styled-table td button, .styled-table td a.button-link {

font-size: 0.95em;

padding: 4px 10px;

border-radius: 5px;

line-height: 1.2;

}

.styled-table td a {

font-size: 0.95em;

padding: 4px 10px;

border-radius: 5px;

line-height: 1.2;

}

/\* Формы \*/

input, select {

width: 100%;

font-size: 1rem;

height: 2rem;

padding: 12px;

margin: 0.5rem 0;

border: 1px solid #d1d5db;

border-radius: 8px;

background: #f5f5f5;

}

input[type="submit"] {

background: var(--primary);

color: white;

margin-top: 16px;

height: 48px;

font-size: 1rem;

border-radius: 8px;

border: none;

cursor: pointer;

padding: 0 24px;

transition: 0.2s;

}

input[type="submit"]:hover {

background: #152066;

}

#trainingTrenerRadios label, #trainingSubRadios label {

display: flex;

align-items: center;

margin-bottom: 8px;

margin-right: 0;

white-space: nowrap;

}

#trainingTrenerRadios input[type=radio], #trainingSubRadios input[type=radio] {

margin-right: 0;

width: 30px;

}

/\* Адаптивность \*/

@media (max-width: 768px) {

.container {

padding: 24px;

}

.styled-table th, .styled-table td {

padding: 12px;

}

}