



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ

ИУ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА

ИУ7 «ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

## КУРСОВАЯ РАБОТА

*НА ТЕМУ:*

*Разработка базы данных для хранения и  
обработки данных фитнес-клуба.*

Студент

**ИУ7-64Б**

(группа)

(подпись, дата)

(И.О. Фамилия)

Руководитель курсового  
проекта

(подпись, дата)

**Исаев А.Л.**

(И.О. Фамилия)

Консультант

(подпись, дата)

(И.О. Фамилия)

**2025 г.**

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой

ИУ7

(индекс)

И. В. Рудаков

(И.О. Фамилия)

(подпись)

28.02.2025 года

(дата)

## З А Д А Н И Е на выполнение курсовой работы

по дисциплине «Базы данных»

Студент группы ИУ7-64Б Цховребова Яна Роландовна

(Фамилия, имя, отчество)

Тема курсовой работы Разработка базы данных для хранения и обработки данных фитнес-клуба.

Направленность КР (учебная, исследовательская, практическая, др.) учебная

Источник тематики (кафедра,  
предприятие, НИР)

кафедра

### Задание

Провести анализ предметной области фитнес-клубов для определения структуры данных, необходимых для хранения и обработки информации о клиентах, тренерах, тренировках и платежах. Сформулировать требования к базе данных, включая ограничения, обеспечивающие корректность и целостность данных. Определить перечень пользователей системы (администраторы, тренеры, клиенты) и их права доступа к данным. Разработать ER-модель базы данных. Спроектировать SQL-триггеры, обеспечивающие автоматическое обновление данных, например, продление статуса абонента после оплаты или контроль количества мест на тренировке. Спроектировать ролевую модель на уровне базы данных, позволяющую разграничить доступ к данным для администраторов, тренеров и клиентов. Выбрать технологический стек для реализации проекта. Реализовать схему базы данных с учетом проектных решений. Разработанные сущности и связи протестировать на корректность. Провести исследование производительности базы данных при увеличении объема данных и количестве одновременно выполняемых запросов, а также с использованием кеширования и без него.

### Оформление курсовой работы:

Расчетно-пояснительная записка на 18-40 листах формата А4. Презентация к курсовой работе на 6-15 слайдах.

Дата выдачи задания «28» февраля 2025 г.

Руководитель курсовой работы

(подпись, дата)

Исаев А.Л.

(И.О. Фамилия)

Студент

(подпись, дата)

(И.О. Фамилия)

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

КАЛЕНДАРНЫЙ ПЛАН  
на выполнение курсовой работы

по дисциплине «Базы данных»

Студент группы ИУ7-64Б Цховребова Яна Роландовна

(Фамилия, имя, отчество)

Тема курсовой работы

Разработка базы данных для хранения и обработки данных фитнес-клуба.

№ п/п	Наименование этапов выпускной квалификационной работы	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Руководитель КП	Куратор
1.	Задание на выполнение курсовой работы			Исаев А.Л.	
2.	1 модуль	Планируемая дата		Исаев А.Л.	
3.	2 модуль	Планируемая дата		Исаев А.Л.	
4.	Оформление РПЗ (Отчета)	Планируемая дата		Исаев А.Л.	
5.	Подготовка доклада и презентации (при необходимости)	Планируемая дата		Исаев А.Л.	
6.	Защита курсовой работы	Планируемая дата		Исаев А.Л.	

Студент \_\_\_\_\_  
(подпись, дата)

Руководитель работы \_\_\_\_\_  
(подпись, дата)

## РЕФЕРАТ

# СОДЕРЖАНИЕ

<b>РЕФЕРАТ</b>	<b>4</b>
<b>ВВЕДЕНИЕ</b>	<b>7</b>
<b>1 Аналитическая часть</b>	<b>8</b>
1.1 Анализ предметной области . . . . .	8
1.2 Формализация задачи . . . . .	9
1.3 Формализация данных . . . . .	10
1.4 Формализация пользователей и их прав доступа . . . . .	11
1.5 Модели данных . . . . .	13
1.5.1 Реляционная модель данных . . . . .	13
1.5.2 Модель данных «ключ-значение» . . . . .	16
1.5.3 Документная модель данных . . . . .	16
1.6 Обзор существующих решений . . . . .	17
<b>2 Конструкторская часть</b>	<b>22</b>
2.1 Разработка базы данных . . . . .	22
2.2 Разработка сущностей базы данных . . . . .	23
2.3 Разработка ограничений целостности данных . . . . .	27
2.4 Разработка ролевой модели . . . . .	32
2.5 Разработка функции, процедур и триггеров . . . . .	33
<b>3 Технологическая часть</b>	<b>36</b>
3.1 Средства реализации . . . . .	36
3.2 Реализация базы данных . . . . .	36
3.2.1 Реализация триггеров . . . . .	44
3.2.2 Создание ролевой модели . . . . .	47
3.3 Тестирование . . . . .	48
3.4 Интерфейс для взаимодействия с базой данных . . . . .	51
<b>4 Исследовательская часть</b>	<b>52</b>
4.1 Технические характеристики . . . . .	52

<b>ЗАКЛЮЧЕНИЕ</b>	<b>57</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>59</b>

# ВВЕДЕНИЕ

В последние годы наблюдается значительный рост интереса к физической активности и занятиям спортом, что, в свою очередь, привело к увеличению числа услуг, ориентированных на благополучие, здоровье и активный образ жизни, стремящихся удовлетворить растущий спрос и потребности общества [1].

Современные фитнес-клубы представляют собой не только места для занятий спортом, но и полноценные бизнес-структуры, в которых осуществляется широкий спектр процессов: регистрация и обслуживание клиентов, планирование и проведение тренировок, управление персоналом и тренерами, ведение финансовой отчетности и многое другое. В связи с этим возникает необходимость в создании надежной и гибкой информационной системы, основой которой является качественно спроектированная база данных.

**Цель работы** – разработка базы данных для хранения и обработки данных фитнес-клуба.

Для достижения цели курсовой работы необходимо выполнить следующие **задачи**:

- 1) провести анализ предметной области и формализовать задачу;
- 2) описать структуру базы данных и ее пользователей;
- 3) провести анализ моделей данных и выбрать наиболее подходящую;
- 4) спроектировать требуемую базу данных;
- 5) спроектировать триггеры для автоматического обновления данных;
- 6) выбрать средства реализации базы данных;
- 7) реализовать спроектированную базу данных и обеспечить её функциональность согласно проектным решениям;
- 8) реализовать интерфейс для доступа к базе данных;
- 9) провести исследование производительности базы данных при увеличении объема данных, количества одновременных запросов, а также с использованием кеширование и без него.

# 1 Аналитическая часть

## 1.1 Анализ предметной области

**Фитнес-клуб** – это учреждение, оснащённое оборудованием для физических упражнений, предоставляющие услуги в области физической активности и здоровья [1, 2].

**Главная цель** фитнес-клуба: обеспечение комфортных условий для физической активности клиентов, улучшении их здоровья и поддержании высокого уровня физической формы [1, 2, 3].

**Основные функции** фитнес-клуба:

- обеспечение тренировок и активного отдыха для клиентов;
- предоставление абонементов с различными условиями для доступа к услугам клуба;
- управление расписанием тренировок;
- учет посещений, тренировки и прогресса клиентов [3].

**Клиенты** фитнес-клуба – физические лица, которые заинтересованы в поддержании физической активности и улучшении здоровья. Например, люди, занимающиеся спортом для поддержания физической формы или проходящие реабилитацию для восстановления после травм [2, 3].

**Абонементы** являются основным способом доступа клиентов к услугам фитнес-клуба, и их типы определяются самим клубом в зависимости от потребностей и предпочтений клиентов. Например, могут быть предложены ежемесячные абонементы (предоставляющие доступ на один месяц) либо годовые абонементы (с выгодными условиями на длительный период) [3].

Основными **сотрудниками** фитнес-клуба являются:

- **администраторы**, которые управляют клиентской базой, занимаются продажей абонементов, отслеживанием посещаемости и предоставляют информацию о клубе.
- **тренеры** – специалисты, которые проводят тренировки для клиентов [3].



## 1.2 Формализация задачи

Целью данной курсовой работы является разработка базы данных для хранения и обработки информации фитнес-клуба.

Предметная область задачи охватывает деятельность фитнес-клуба, включая управление данными о клиентах (их личной информации, тренировках, истории посещений), тренерах (их расписаниях, специализациях) и организации тренировок, расписаний, а также систему учета абонементов и оплат.

Цель создания базы данных – автоматизировать процессы учета и повысить эффективность функционирования фитнес-клуба, а также улучшить обслуживание клиентов.

Для взаимодействия с базой данных фитнес-клуба необходимо разработать интерфейс, который обеспечит функциональный доступ к ключевым операциям системы в зависимости от роли пользователя.

Интерфейс должен предусматривать следующие возможности:

- регистрация и авторизация пользователей;
- приобретение абонементов и управление ими;
- запись на тренировки и управление расписанием.

В рамках данной курсовой работы не рассматривается вопрос обеспечения конфиденциальности персональных данных пользователей и интеграции платежной системы:

- все данные, включая контактные и учетные данные пользователей, будут храниться в базе данных без использования специализированных механизмов защиты конфиденциальности;
- не реализуется функциональность для работы с платежными системами и интеграция с внешними платёжными сервисами для обработки финансовых транзакций или хранения данных о платежах.

### 1.3 Формализация данных

На рисунке 1 изображена диграмма сущность–связь в нотации Чена.



Рисунок 1 – ER-диаграмма фитнес-клуба в нотации Чена

В базе данных для фитнес-клуба можно выделить следующие ключевые сущности.

1. **Пользователь** – основная сущность, представляющая всех участников системы: клиентов, тренеров и администраторов.
2. **Роль пользователя** – сущность, определяющая уровень доступа и права пользователя в системе.
3. **Тренер** – сущность, связанная с пользователем, имеющим соответствующую роль тренера.
4. **Специализация** – сущность, которая определяет специализацию и используется для описания тренировок и компетенций тренеров.
5. **Тип абонемента** – сущность, описывающая варианты абонементов, устанавливаемые фитнес-клубом.
6. **Абонемент** – сущность, отражающая приобретённый пользователем абонемент.
7. **Заказ и позиция заказа** – сущности, которые фиксируют процесс приобретения абонементов пользователями.
8. **Платеж** – сущность, отражающая факт оплаты заказа.
9. **Зал** – сущность, которая содержит информацию о помещениях для проведения тренировок.
10. **Тренировка** – сущность, представляющая собой запланированное мероприятие – тренировку.
11. **Посещение** – сущность, которая фиксирует факт участия клиента в конкретной тренировке.

#### 1.4 Формализация пользователей и их прав доступа

Взаимодействовать с базой данных будут четыре вида пользователей. На рисунке 2 приведена диаграмма вариантов использования базы данных.

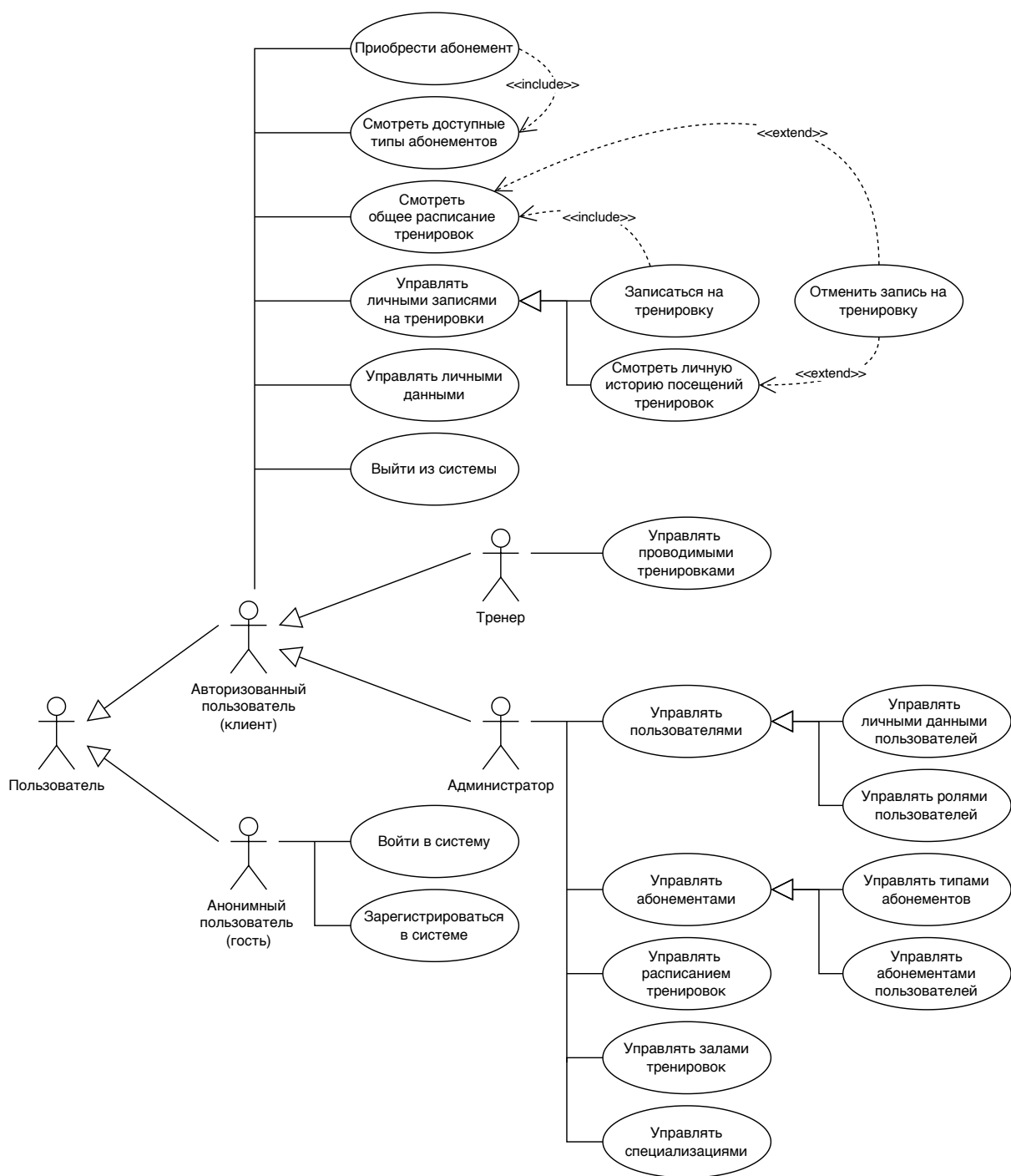


Рисунок 2 – Диаграмма вариантов использования базы данных

**Гость** – анонимный пользователь, который может выполнять только определенные действия, такие как регистрация и вход в систему.

Авторизованные пользователи:

— **клиент** – пользователь, который имеет минимальные права, ограни-

ченные только возможностью взаимодействовать с данными, относящимися к его учетной записи и обслуживанию (например, заказами, абонементом и посещениями).

- **тренер** – пользователь, имеющий права клиента, а также имеющий доступ к данным, связанным с его тренерской деятельностью.
- **администратор** – пользователь, который имеет полный доступ к любым данным.

## 1.5 Модели данных

**Модель данных** представляет собой формализованное описание структуры информационных единиц и операций с ними в информационной системе, которые определяют логическую организацию базы данных и способы хранения, организации и обработки данных [4, с. 4].

Существует множество моделей данных, которые делятся на дореляционные (иерархические, сетевые, основанные на инвертированных списках), реляционные, постреляционные модели (например, NoSql-модели: ключ-значение, столбцовые, документные и графовые). Каждая из этих моделей имеет свои особенности и применяется в различных областях [4].

Основываясь на рейтинге из [5], в дальнейшем будут рассмотрены наиболее популярные в настоящее время модели данных.

### 1.5.1 Реляционная модель данных

Реляционная модель была предложена Э. Коддом в 1970 году в статье [6] и основана на теории отношений, опирается на математическое понятие  $n$ -арного отношения, что представляет собой подмножество декартового произведения.

Основными понятиями реляционных баз данных являются тип данных, домен, атрибут, кортеж, отношение, первичный ключ.

**Тип данных** в реляционной модели данных аналогичен типу данных в языках программирования и включает символы, числа, битовые строки, специализированные числовые данные (например, «деньги») и «темпоральные» данные (дата, время).

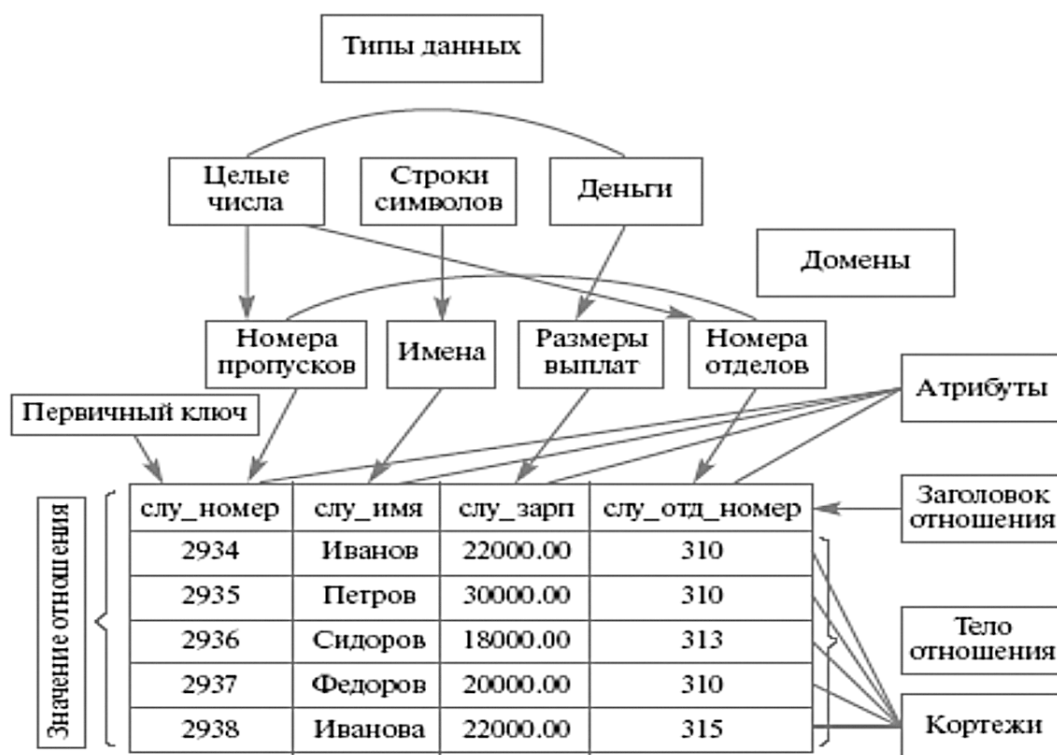


Рисунок 3 – Основные понятия реляционной модели данных [4, с. 31]

**Домен** – множество допустимых значений для типа данных.

**Атрибут** – некоторая характеристика объекта (сущности), имеющая уникальное имя внутри отношения, через которое к нему производится обращение.

Атрибут, значение которого *идентифицирует кортеж*, называется **ключом** отношения. В зависимости от количества атрибутов, входящих в ключ, различают простые и сложные (составные) ключи.

**Первичный ключ** – уникальный атрибут (или их комбинация), который идентифицирует кортежи в таблице. Свойства: *уникальность* (в любой момент времени никакие два кортежа отношения не должны иметь одно и то же значение); *минимальность* (ни один из атрибутов не может быть исключен из набора атрибутов первичного ключа без нарушения свойств уникальности).

В любой из таблиц может оказаться несколько наборов атрибутов, которые можно выбрать в качестве ключа, такие наборы называются **потенциальными** и **возможными** ключами.

**Внешний ключ** – атрибут, который ссылается на первичный ключ

другого отношения, обеспечивая связь между таблицами.

**Кортеж** – это совокупность значений, где каждое значение (или элемент) соответствует определенному атрибуту, и эти значения принадлежат соответствующему домену атрибута.

Множество всех кортежей образует **отношение**, которое является подмножеством декартового произведения доменов, причем количество кортежей в отношении называется *мощностью отношения*.

**Схема отношения (заголовок отношения)** – набор упорядоченных пар  $\langle A, T \rangle$ , где  $A$  – имя атрибута, а  $T$  – домен. Количество атрибутов в схеме называется *степенью отношения*.

**Базовое отношение** – содержит атрибуты и первичный ключ, а **производное отношение** – используется для связи между таблицами.

Таким образом, **реляционная база данных** – это множество отношений, представленных в виде таблиц, где заголовок – схема отношения, а строки – кортежи.

#### **Фундаментальные свойства отношений:**

- отсутствие кортежей-дубликатов;
- отсутствие упорядоченности кортежей;
- отсутствие упорядоченности атрибутов;
- атомарность значений атрибутов (является базовым требованием для реляционных баз данных – требование нормализованных отношений или отношений, представленных в первой нормальной форме).

Реляционная модель данных, по К. Дейту, состоит из трех частей.

1. **Структурная часть** – описывает организацию данных, включая таблицы (отношения), атрибуты и их типы, а также связи между таблицами.
2. **Манипуляционная часть** – включает операции над данными, такие как выборка, добавление, обновление и удаление данных, что осуществляется через язык запросов (например, SQL).

3. **Целостная часть** – обеспечивает соблюдение целостности данных, включая ограничения (например, уникальность значений, ссылки между таблицами через внешние ключи), чтобы поддерживать корректность и непротиворечивость данных в базе [4, С. 30-35].

### 1.5.2 Модель данных «ключ-значение»

**Модель данных ключ-значение** представляет собой структуру данных, которая использует ассоциативный массив, где каждый элемент состоит из *уникального ключа* и *связанного с ним значения*. Значение может быть любым типом данных, но оно не имеет структуры.

Эта модель поддерживает **две основные операции**:

- **получение значения по ключу** – если ключ существует, возвращается связанное с ним значение, иначе – NULL (специальное значение, которое используется в базах данных для обозначения отсутствия данных или неизвестного значения).
- **запись значения по ключу** – позволяет добавить или обновить значение для определенного ключа (также возможно установить время жизни ключа, после чего он будет автоматически удален) [4, С. 89-91].

### 1.5.3 Документная модель данных

Модель документная расширяет представление модели «ключ-значение», позволяя хранить более сложные структуры данных – документы.

**Документная модель данных** – это подход к организации и хранению данных, при котором данные представлены в виде документов.

**Основные характеристики документной модели:**

- 1) документ как основная единица хранения;
- 2) **документ** – это структурированный набор данных, который может содержать различные пары *ключ-значение* и может включать
  - простые типы данных: строки, числа, булевы значения;
  - упорядоченные списки значений;



- вложенные документы: другие объекты, которые могут быть представлены в формате пары ключ-значение;
  - сложные типы данных: например, даты, бинарные данные и другие;
- 3) отсутствие операций соединения – связанные данные обычно хранятся в одном документе [4, С. 92-96].

## Выбор модели данных

Для базы данных фитнес-клуба была выбрана реляционная модель данных, поскольку она наиболее подходит для работы с хорошо структурированными данными и поддерживает сложные связи между сущностями, такими как клиенты, тренеры, абонементы и расписания.

Для кэширования данных была выбрана модель «ключ-значение», так как она подходит для быстрого доступа к данным, часто запрашиваемым пользователями, таким как актуальные типы абонементов или специализации тренеров, которые не требуют хранения сложного структурированного набора данных, например, документов.

## 1.6 Обзор существующих решений

### 1С:Фитнес клуб

**1С:Фитнес клуб** – это программное решение для автоматизации фитнес-клубов и спортивных учреждений, созданное компанией «Лаборатория программного обеспечения».

Приложение включает в себя функционал для работы с клиентской базой, маркетинга, расчета заработной платы, учета посещений, CRM-систему, интеграции с внешними сервисами и мобильные приложения для персонала.

Приложение не предоставляет пробный период, но предлагает несколько тарифных планов, подходящих для разных масштабов бизнеса. Тарифы варьируются от 1 990 рублей в месяц за облачную подписку с базовым набором функций до 90 000 рублей за покупку программы с расширенными возможностями.

← → ☆ **Соколова Виктория Семеновна (Бывший член клуба)** x

Главное Членства, пакеты услуг Посещения Занятия Взаиморасчеты Договоры Задачи История Файлы

Записать и закрыть Записать Задача Согласие... Еще ?

**Бывший член клуба**

Фамилия:

Имя:

Отчество:

Дата рождения:  Пол: ☐ Мужской ☒ Женский

Контакты + Добавить контакт

Телефон:

Email:

Адрес:

Удостоверение личности + Добавить документ

Паспорт, 1212 11112456, выдан 07.05.2003 г. 10 ОМС г. Москвы

Карты + Выдать карту

Карта №032565653

Маркетинг

Канал привлечения:

Рекламный источник:


Интересы + Добавить интерес

Персональные менеджеры

Отдела продаж:

Отдела сервиса:

**Невыполненные задачи (0)**



Членства/пакеты услуг + Продать членство/пакет услуг

Автомобили + Добавить автомобиль

Родственники + Добавить родственника

#Теги + Добавить тег

Заметки + Добавить заметку

Комментарий:

Рисунок 4 – Интерфейс администратора для управления данными клиента фитнес-клуба приложения 1С:Фитнес клуб

← → ☆ **Расписание занятий** x

Занятия: Все Персональные Групповые Аренда зала По: Сотруднику Помещению Нет Утро Вечер

Отбор по: Сотруднику Помещению Услуге Группе Клиенту По сегментам: Сотрудников Услуг Клиентов

Починков О.П. Ман А.П. Мыскина В.О. Федоров И.И. Антонов Ф.А. Немирович Д.А.

Период: День Неделя Месяц Значение: Сегодня Завтра < Сегодня > Мастер заполнения

	Зал боевых единоборств	Зал групповых программ	Тренажерный зал
15:00			
16:00		16:00-16:55, ABL, Немирович Д. А., (0 из 30)	
17:00			
18:00	18:00-19:30, MMA - 90мин., Антонов Ф. А., (0)	18:00-18:55, BOSU, Мыскина В. О., (0 из 30)	
19:00			
20:00		20:00-20:55, DANCE MIX, Антонов Ф. А., (0 из 30)	
21:00			
22:00			

Рисунок 5 – Интерфейс администратора для управления расписание тренировок фитнес-клуба приложения 1С:Фитнес клуб

Для работы приложения необходима операционная система Windows. Для онлайн версии требуется постоянный интернет на компьютере.

## fitness365

**fitness365** – это веб-система, специально разработанная для комплексного управления фитнес-клубом. Она предоставляет широкий набор инструментов, которые охватывают все аспекты работы клуба: от работы с клиентами до ведения статистики и аналитики.

fitness365 имеет интегрированную CRM-систему, которая позволяет клубу управлять базой клиентов – каждому пользователю предоставляется персонализированный доступ.

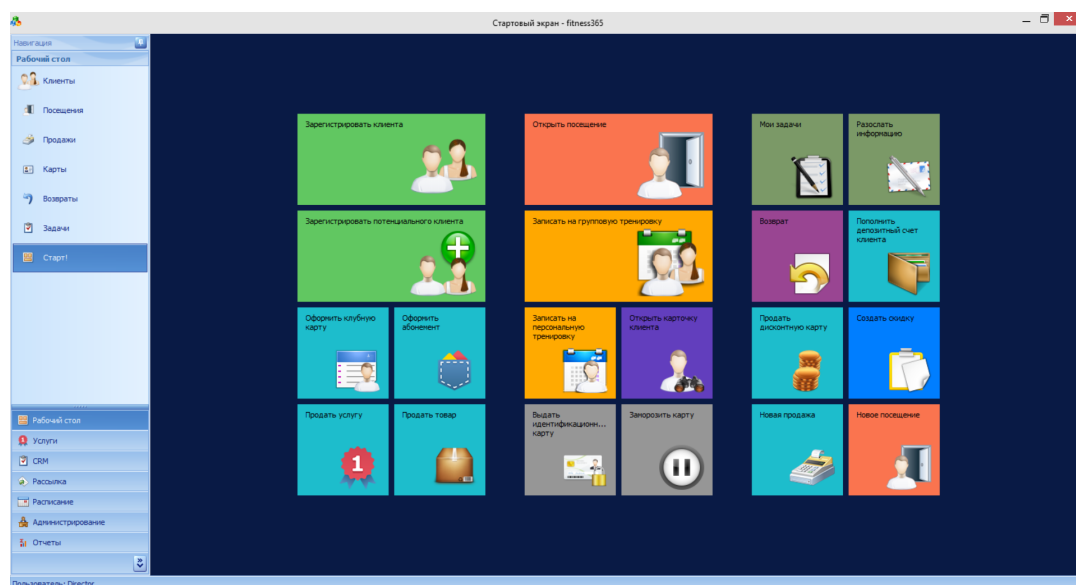


Рисунок 6 – Интерфейс главной страницы администратора фитнес-клуба приложения fitness365

Тарифы fitness365 предлагают различные опции для фитнес-клубов. Например, тариф «Онлайн» стоит 2 000 рублей в месяц, включает доступ через интернет и возможность работы с неограниченным числом клиентов, но поддерживает 5 рабочих мест. Для клубов, где проблемы с интернетом, доступна настольная версия за разовый платеж 44 000 рублей. Кроме того, есть дополнительные опции за дополнительную оплату.

Для работы приложения необходима операционная система Windows. Для онлайн версии требуется постоянный интернет на компьютере.

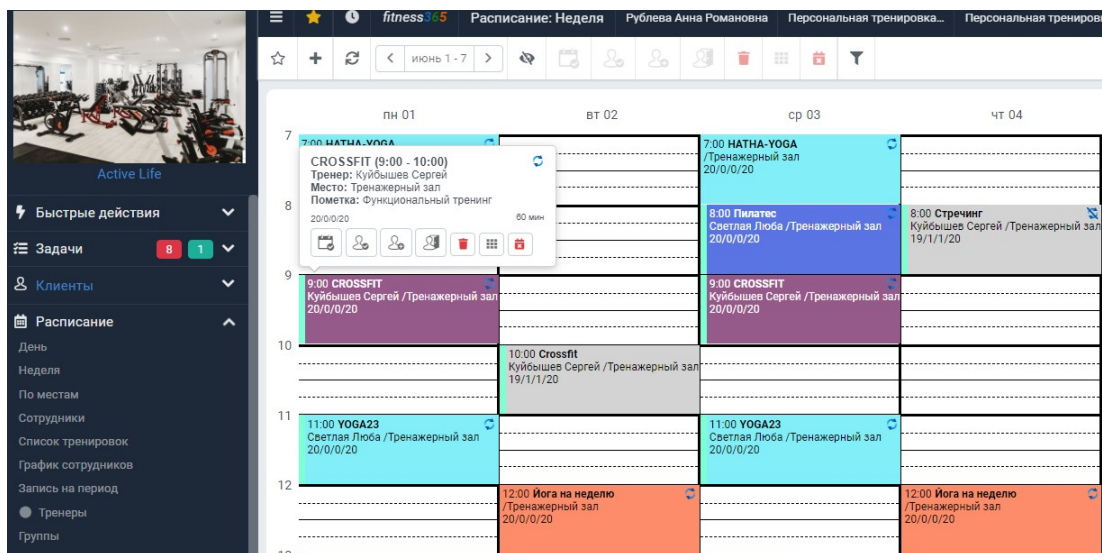


Рисунок 7 – Интерфейс администратора для управления расписанием фитнес-клуба приложения fitness365

## Сравнение решений

Рассматриваемые решения, такие как 1С:Фитнес Клуб и fitness365, могут быть избыточными для конкретных нужд фитнес-клуба, что делает их сложными и громоздкими для использования в условиях небольших или специализированных клубов. Также они ориентированы на платформу Windows и предлагают доступ через интернет, что ограничивает их использование преимущественно стационарными компьютерами – это может быть неудобно для тренеров и клиентов, которым необходим постоянный доступ к системе, особенно в условиях мобильности. Кроме того, все рассматриваемые решения являются достаточно дорогими, что может стать препятствием для небольших фитнес-клубов.

Мобильные устройства позволяют обеспечить доступ к системе в любом месте и в любое время, поэтому разрабатываемое приложение для доступа к базе данных будет мобильным, ориентированным на платформу iOS. Также приложение будет сфокусировано на ключевых функциях, таких как управление расписанием для тренеров и клиентов, самостоятельный контроль личного кабинета. Кроме того, приложение будет бесплатным, что делает его более доступным для широкого круга пользователей.

<b>Функциональность для администратора</b>	<b>1С:Фитнес Клуб</b>	<b>fitness365</b>
Управление расписанием, залами и персоналом	+	+
Работа с абонементом и услугами	+	+
Ведение клиентской базы и CRM	+	+
<b>Функциональность для клиента</b>	<b>1С:Фитнес Клуб</b>	<b>fitness365</b>
Наличие личного кабинета	+	+
Покупка абонементов онлайн	+	+
Самостоятельная запись на тренировки	+	+
<b>Функциональность для тренера</b>	<b>1С:Фитнес Клуб</b>	<b>fitness365</b>
Доступ к информации о клиентах	–	+
Просмотр и управление своим расписанием	–	–
Мобильный доступ к системе	–	–

Таблица 1 – Сравнение функциональности по 1С:Фитнес Клуб и fitness365

## Вывод

В данном разделе проведен анализ предметной области фитнес-клубов, описана структура базы данных, рассмотрены пользователи базы данных и их права доступа. Также проведен анализ различных моделей данных, в результате чего для хранения данных выбрана реляционная модель данных, а для кэширования – модель «ключ-значение». Рассмотрены существующие решения и их особенности.

## 2 Конструкторская часть

В данном разделе будет представлена структура проектируемой базы данных для фитнес-клуба, включая описание сущностей, ограничений целостности данных, а также модели ролей пользователей. Также будут описаны используемые функции, процедуры и триггеры, обеспечивающие корректность и безопасность работы с данными.

### 2.1 Разработка базы данных

На рисунке 8 представлена схема проектируемой базы данных.

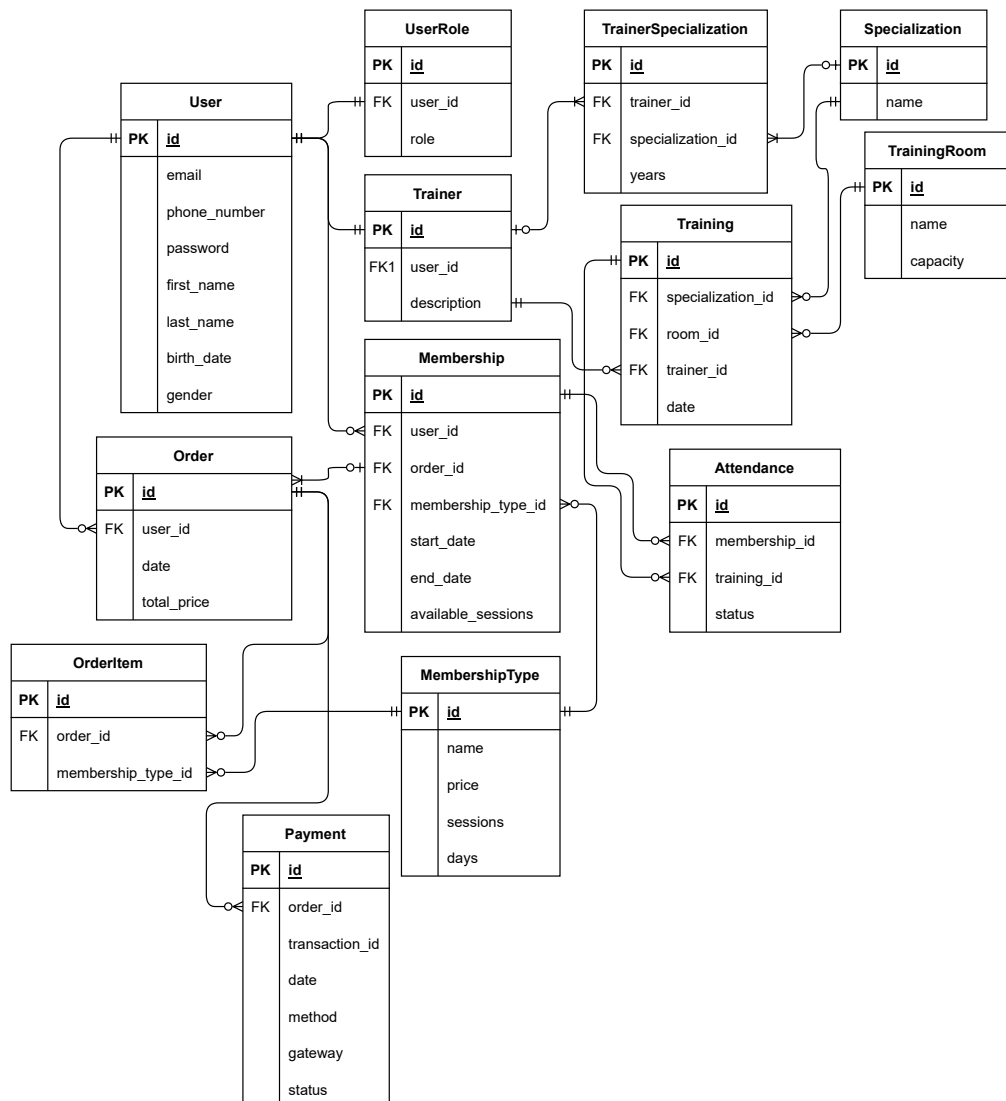


Рисунок 8 – ER-диаграмма базы данных фитнес-клуба

## 2.2 Разработка сущностей базы данных

Далее будут описаны тринадцать сущностей проектируемой базы данных.

**User** – сущность, представляющая зарегистрированного пользователя системы.

Атрибуты:

- *id* – уникальный идентификатор пользователя;
- *first\_name* – имя;
- *last\_name* – фамилия;
- *email* – адрес электронной почты;
- *password* – зашифрованный пароль;
- *phone\_number* – номер телефона;
- *birth\_date* – дата рождения;
- *gender* – пол пользователя.

**UserRole** – сущность, представляющая роли конкретных пользователей.

Атрибуты:

- *id* – уникальный идентификатор роли;
- *role* – название роли пользователя;
- *user\_id* – внешний ключ, ссылающийся на пользователя.

**Trainer** – сущность, представляющая дополнительная информация о пользователях-тренерах.

Атрибуты:

- *id* – уникальный идентификатор тренера;
- *user\_id* – внешний ключ, ссылающийся на пользователя;

- *description* – краткое описание или биография.

**Specialization** – сущность, представляющая типы специализаций тренеров и тренировок.

Атрибуты:

- *id* – уникальный идентификатор специализации;
- *name* – название специализации.

**TrainerSpecialization** – сущность, представляющая связь между тренерами и их специализациями, то есть специализации тренеров.

Атрибуты:

- *id* – уникальный идентификатор специализации тренера;
- *trainer\_id* – внешний ключ, ссылающийся на тренера;
- *specialization\_id* – внешний ключ, ссылающийся на специализацию;
- *years* – опыт тренера в годах для специализации.

**TrainingRoom** – сущность, представляющая залы, в которых проводятся тренировки.

Атрибуты:

- *id* – уникальный идентификатор зала;
- *name* – название зала;
- *capacity* – вместимость зала.

**MembershipType** – сущность, представляющая типы абонементов, доступных для покупки.

Атрибуты:

- *id* – уникальный идентификатор типа абонемента;
- *name* – название типа абонемента;
- *days* – длительность действия в днях;



- *price* – стоимость;
- *sessions* – количество доступных тренировок.

**Membership** – сущность, представляющая абонементы, приобретённые пользователями.

Атрибуты:

- *id* – уникальный идентификатор абонемента;
- *user\_id* – внешний ключ, ссылающийся на пользователя-клиента;
- *membership\_type\_id* – внешний ключ, ссылающийся на тип абонемента;
- *start\_date* – дата начала действия;
- *end\_date* – дата окончания действия;
- *available\_sessions* – количество доступных (оставшихся) тренировок.

**Order** – сущность, представляющая оформленные пользователями заказы.

Атрибуты:

- *id* – уникальный идентификатор заказа;
- *user\_id* – внешний ключ, ссылающийся на пользователя;
- *date* – дата оформления;
- *total\_price* – общая цена заказа.

**OrderItem** – сущность, представляющая позиции внутри заказа.

Атрибуты:

- *id* – уникальный идентификатор позиции заказа;
- *order\_id* – внешний ключ, ссылающийся на заказ;
- *membership\_type\_id* – внешний ключ, ссылающийся тип абонемента.

**Payment** – сущность, представляющая информацию об оплате заказов.

Атрибуты:

- *id* – уникальный идентификатор платежа;
- *order\_id* – внешний ключ, ссылающийся на заказ;
- *transaction\_id* – идентификатор транзакции, получаемый от платежной системы;
- *date* – дата платежа;
- *method* – способ оплаты;
- *gateway* – платежный шлюз;
- *status* – статус оплаты.

**Training** – сущность, представляющая тренировки.

Атрибуты:

- *id* – уникальный идентификатор тренировки;
- *trainer\_id* – внешний ключ, ссылающийся на тренера;
- *room\_id* – внешний ключ, ссылающийся на зал;
- *specialization\_id* – внешний ключ, ссылающийся на специализацию тренировки.

**Attendance** – сущность, представляющая записи на тренировку пользователей.

Атрибуты:

- **id** – уникальный идентификатор записи на тренировку;
- **training\_id** – внешний ключ, ссылающийся на тренировку;
- **user\_id** – внешний ключ, ссылающийся пользователя-клиента;
- **status** – статус участия.

## 2.3 Разработка ограничений целостности данных

Далее в таблицах 2–14, представлены ограничения целостности данных, разработанные для сущностей проектируемой базы данных.

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
role	строка	Не NULL, значения: ('клиент', 'тренер', 'администратор'), длина до 31 символов
user_id	UUID	Внешний ключ (ссылается на User), уникальный, каскадное удаление

Таблица 2 – Ограничения атрибутов сущности UserRole

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
user_id	UUID	Внешний ключ (ссылается на User), уникальный, каскадное удаление
description	строка	Не NULL, длина до 511 символов

Таблица 3 – Ограничения атрибутов сущности Trainer

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
name	строка	Не NULL, длина до 127 символов, уникальное

Таблица 4 – Ограничения атрибутов сущности Specialization

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
name	строка	Не NULL, длина до 63 символов, уникальное
capacity	целое число	Не NULL, больше 0

Таблица 5 – Ограничения атрибутов сущности TrainingRoom

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
name	строка	Не NULL, уникальное, длина до 127 символов
price	число с фиксированной точностью	Не NULL, больше или равно 0.0, по умолчанию 0.0
sessions	целое число	Не NULL, больше 0, по умолчанию 1
days	целое число	Не NULL, больше 0, по умолчанию 1

Таблица 6 – Ограничения атрибутов сущности MembershipType

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
order_id	UUID	Внешний ключ (ссылается на Order), каскадное удаление
membership_type_id	UUID	Внешний ключ (ссылается на MembershipType), каскадное удаление

Таблица 7 – Ограничения атрибутов сущности OrderItem

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
user_id	UUID	Внешний ключ (ссылается на User), при удалении родительской записи в связанной таблице поле устанавливается в NULL
date	метка времени	Не NULL, по умолчанию текущая метка времени, значение не меньше текущей метки времени
total_price	число с фиксированной точностью	Не NULL, больше или равно 0.0, значение по умолчанию 0.0

Таблица 8 – Ограничения атрибутов сущности Order

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
email	строка	Не NULL, уникален, проверка по регулярному выражению, которое соответствует формату email (например, user@example.com), длина до 127 символов
phone_number	строка	Не NULL, уникален, проверка по регулярному выражению, которое соответствует телефонным номерам в международном формате (например, +1234567890), длина до 31 символов
password	строка	Не NULL, длина до 255 символов
first_name	строка	Не NULL, проверка по регулярному выражению, которое позволяет только буквы (с возможным дефисом внутри имени), длина до 127 символов
last_name	строка	Не NULL, проверка по регулярному выражению, которое позволяет только буквы (с возможным дефисом внутри фамилии), длина до 127 символов
birth_date	дата	Не NULL, проверка на возраст: от 14 до 120 лет
gender	строка	Не NULL, значения: ('мужской', 'женский'), длина до 31 символов

Таблица 9 – Ограничения атрибутов сущности User

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
membership_id	UUID	Внешний ключ (ссылается на Membership), каскадное удаление
training_id	UUID	Внешний ключ (ссылается на TrainingSession), при удалении родительской записи в связанной таблице поле устанавливается в NULL
status	строка	Не NULL, длина до 63 символов, значения: ('посетил', 'ожидает', 'отсутствовал'), по умолчанию 'ожидает'

Таблица 10 – Ограничения атрибутов сущности Attendance

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
trainer_id	UUID	Внешний ключ (ссылается на Trainer), при удалении родительской записи в связанной таблице поле устанавливается в NULL
room_id	UUID	Внешний ключ (ссылается на TrainingRoom), при удалении родительской записи в связанной таблице поле устанавливается в NULL
specialization_id	UUID	Внешний ключ (ссылается на Specialization), при удалении родительской записи в связанной таблице поле устанавливается в NULL
date	метка времени	Не NULL

Таблица 11 – Ограничения атрибутов сущности Training

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
order_id	UUID	Внешний ключ (ссылается на Order), при удалении родительской записи в связанной таблице поле устанавливается в NULL
transaction_id	строка	Не NULL, уникальное, длина до 255 символов
date	метка времени	Не NULL, по умолчанию текущая метка времени
method	строка	Не NULL, значения: ('наличные', 'кредитная карта', 'банковский перевод'), длина до 63 символов, по умолчанию 'наличные'
gateway	строка	Может быть NULL, длина до 63 символов, по умолчанию NULL
status	строка	Не NULL, длина до 63 символов, значения: ('ожидает', 'оплачен', 'отменен'), по умолчанию 'ожидает'.

Таблица 12 – Ограничения атрибутов сущности Payment

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
trainer_id	UUID	Внешний ключ (ссылается на Trainer), каскадное удаление
specialization_id	UUID	Внешний ключ (ссылается на Specialization), уникальная комбинация (trainer_id, specialization_id), каскадное удаление
years	целое число	Не NULL, больше или равно 0

Таблица 13 – Ограничения атрибутов сущности TrainerSpecialization

Атрибут	Тип данных	Ограничение
id	UUID	Уникальный идентификатор
user_id	UUID	Внешний ключ (ссылается на User), каскадное удаление
order_id	UUID	Внешний ключ (ссылается на Order), при удалении родительской записи в связанной таблице поле устанавливается в NULL
membership_type_id	UUID	Внешний ключ (ссылается на MembershipType), при удалении родительской записи в связанной таблице поле устанавливается в NULL
start_date	дата	Может быть NULL, по умолчанию NULL
end_date	дата	Может быть NULL, по умолчанию NULL, не раньше start_date
available_sessions	целое число	Не NULL, больше или равно 0, по умолчанию 0

Таблица 14 – Ограничения атрибутов сущности Membership

## 2.4 Разработка ролевой модели

Для обеспечения безопасности и разграничения доступа к данным разработанной базы данных фитнес-клуба необходимо разработать ролевую модель управления доступом.

### Роли базы данных

#### 1. Гость (Guest)

- имеет минимальные права;
- имеет доступ только к функциям регистрации и аутентификации;
- не имеет прямого доступа к таблицам базы данных.



## 2. Клиент (Client)

- является зарегистрированным и аутентифицированным пользователем системы;
- может оформлять и оплачивать заказы, то есть приобретать абонементы;
- имеет возможность записываться на тренировки, просматривать расписание и свои персональные данные;
- имеет доступ только к собственным записям в таблицах: User, Membership, Order, OrderItem, Payment, Attendance;
- может просматривать публичную информацию, содержащуюся в таблицах: Specialization, TrainerSpecialization, MembershipType, Training, Training.

## 3. Тренер (Trainer)

- является поднаследником клиента (наследует все его права), но обладает расширенными возможностями;
- имеет доступ к записям в таблицах Training и Attendance, где он указан в качестве тренера;
- может просматривать информацию о своих клиентах и их посещениях тренировок;
- имеет право изменять данные только тех тренировок (включая посещения этих тренировок), которые он проводит.

4. **Администратор (Admin)** обладает полными правами на управление базой данных.

## 2.5 Разработка функции, процедур и триггеров

Триггер, алгоритм которого представлен на рисунке 9, предназначен для автоматического обновления количества доступных тренировок у абонента пользователя в таблице Membership в зависимости от изменения статуса посещения тренировки в таблице Attendance.

Триггер, алгоритм которого представлен на рисунке 10, предназначен для проверки вместимости зала перед тем, как записать пользователя на тренировку. Он предотвращает запись, если в зале нет свободных мест для выбранной тренировки.

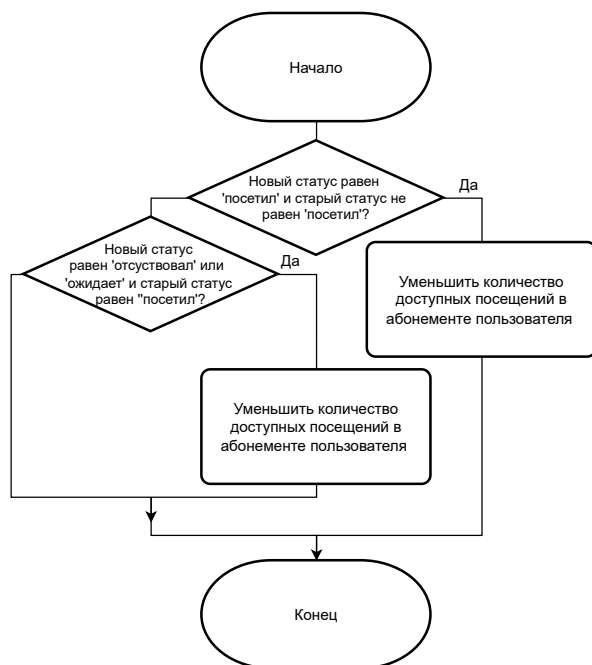


Рисунок 9 – Схема алгоритма работы триггера для обновления данных после изменения статуса посещения тренировки

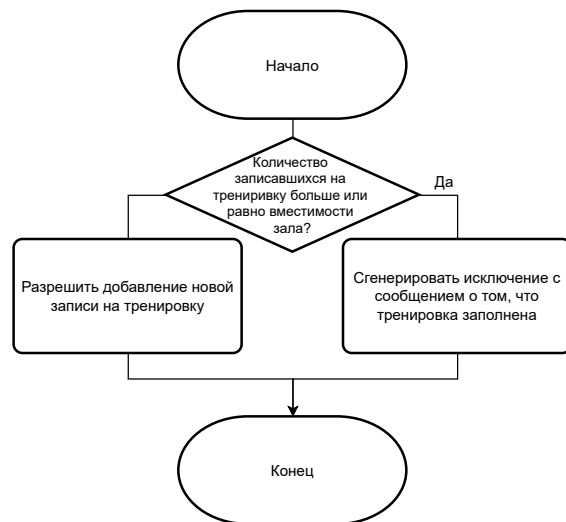


Рисунок 10 – Схема алгоритма работы триггера для проверки данных перед добавлением записи на тренировку

Триггер, алгоритм которого представлен на рисунке 11, предназначен для автоматического создания или обновления абонемента пользователя после успешного выполнения платежа. Этот триггер срабатывает, когда в таблице Payment появляется новая запись или статус платежа изменяется на «оплачено».

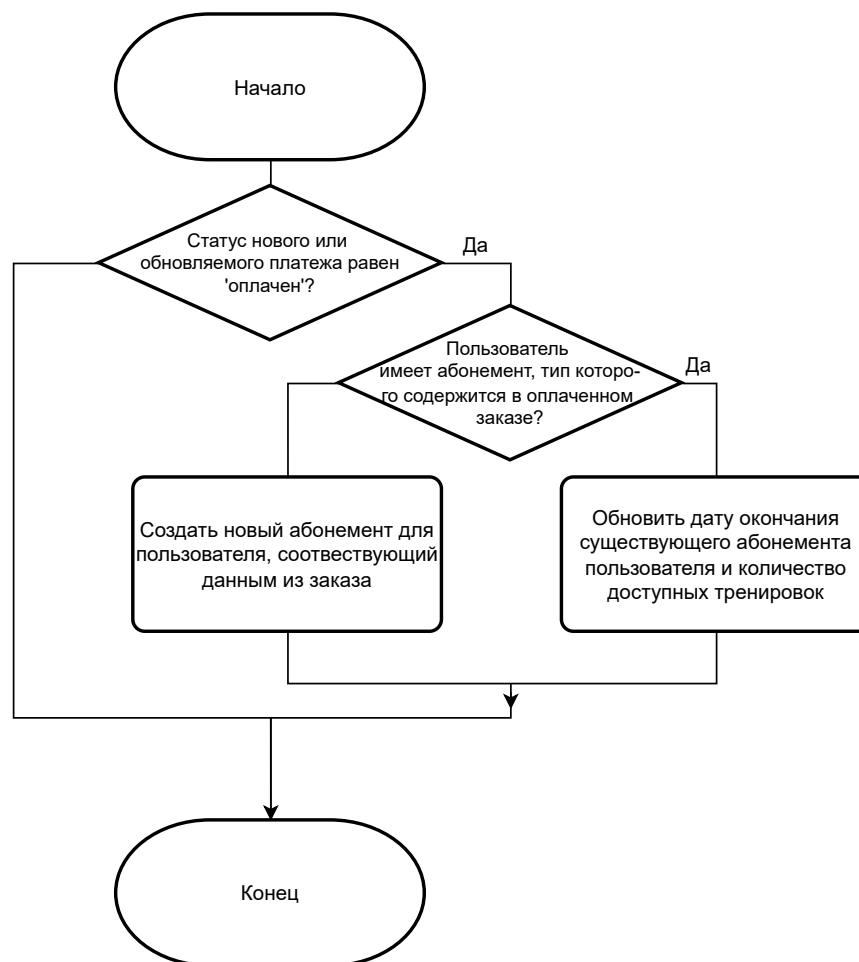


Рисунок 11 – Схема алгоритма работы триггера для обновления или добавления данных абонемента пользователя после оплаты заказа

## Вывод

В данном разделе были разработаны база данных, ее сущности и ограничения целостности данных, а также спроектированы триггеры, обеспечивающие автоматическое обновление данных.

## 3 Технологическая часть

### 3.1 Средства реализации

В качестве системы управления базами данных выбрана **PostgreSQL** [7], которая обеспечивает надёжность, поддерживает механизмы целостности данных и безопасность на уровне ролей и строк (RLS), что важно для строгого разграничения прав доступа между пользователями. PostgreSQL является одной из самых популярных систем управления базами данных [5], и у меня есть опыт работы с ней.

Для кэширования данных используется **Redis** [8], что помогает повысить производительность и сократить нагрузку на базу данных. Redis является популярным и проверенным решением [5], и у меня есть опыт работы с ним.

Язык программирования **Swift** [9] выбран для разработки приложения, поскольку он идеально интегрируется с экосистемой iOS, поддерживает все необходимые инструменты для работы с базой данных и создания графического интерфейса. Также имеется опыт работы с Swift.

Выбраны среды разработки **Xcode** [10] и **pgAdmin 4** [11], так как они предоставляют все необходимые инструменты для создания мобильного приложения и управления базой данных.

### 3.2 Реализация базы данных

Далее представлены реализации спроектированной базы данных: ее сущностей, ограничений целостности данных, ролевой модели и триггеров.

#### Реализация сущностей

Реализации создания таблиц представлены на листингах 1-13.

Листинг 1 – Реализация создания отношения User

```
1 CREATE TABLE "User" (  
2     id                UUID,  
3     email             TEXT,  
4     phone_number      TEXT,  
5     "password"        TEXT,
```

```

6      first_name      TEXT,
7      last_name       TEXT,
8      gender          TEXT,
9      birth_date      DATE
10 );

```

#### Листинг 2 – Реализация создания отношения UserRole

```

1 CREATE TABLE "UserRole" (
2     id          UUID,
3     user_id     UUID,
4     "role"      TEXT
5 );

```

#### Листинг 3 – Реализация создания отношения Trainer

```

1 CREATE TABLE "Trainer" (
2     id          UUID,
3     user_id     UUID,
4     description  TEXT
5 );

```

#### Листинг 4 – Реализация создания отношения Specialization

```

1 CREATE TABLE "Specialization" (
2     id          UUID,
3     name        TEXT
4 );

```

#### Листинг 5 – Реализация создания отношения TrainerSpecialization

```

1 -- Специализация тренера
2 CREATE TABLE "TrainerSpecialization" (
3     id          UUID,
4     trainer_id  UUID,
5     specialization_id  UUID,
6     years       INT
7 );

```

#### Листинг 6 – Реализация создания отношения MembershipType

```

1 CREATE TABLE "MembershipType" (
2     id          UUID,
3     name        TEXT,
4     price       NUMERIC,
5     sessions    INT,
6     days        INT
7 );

```

### Листинг 7 – Реализация создания отношения TrainingRoom

```
1 CREATE TABLE "TrainingRoom" (  
2     id          UUID,  
3     name        TEXT,  
4     capacity    INT  
5 );
```

### Листинг 8 – Реализация создания отношения Order

```
1 CREATE TABLE "Order" (  
2     id          UUID,  
3     user_id     UUID,  
4     "date"      TIMESTAMP,  
5     total_price NUMERIC  
6 );
```

### Листинг 9 – Реализация создания отношения OrderItem

```
1 CREATE TABLE "OrderItem" (  
2     id          UUID,  
3     order_id    UUID,  
4     membership_type_id  UUID  
5 );
```

### Листинг 10 – Реализация создания отношения Payment

```
1 CREATE TABLE "Payment" (  
2     id          UUID,  
3     order_id    UUID,  
4     transaction_id  TEXT,  
5     "date"      TIMESTAMP,  
6     "method"    TEXT,  
7     gateway     TEXT,  
8     status      TEXT  
9 );
```

### Листинг 11 – Реализация создания отношения Membership

```
1 CREATE TABLE "Membership" (  
2     id          UUID,  
3     user_id     UUID,  
4     order_id    UUID,  
5     membership_type_id  UUID,  
6     start_date  DATE,  
7     end_date    DATE,  
8     available_sessions  INT  
9 );
```

## Листинг 12 – Реализация создания отношения Training

```
1 CREATE TABLE "Training" (  
2     id                UUID,  
3     specialization_id UUID,  
4     room_id           UUID,  
5     trainer_id        UUID,  
6     "date"            TIMESTAMP  
7 );
```

## Листинг 13 – Реализация создания отношения Attendance

```
1 CREATE TABLE "Attendance" (  
2     id                UUID,  
3     membership_id     UUID,  
4     training_id        UUID,  
5     status             TEXT  
6 );
```

## Реализация ограничений целостности данных

Реализации ограничений целостности данных таблиц представлены на листингах 14-26.

## Листинг 14 – Реализация ограничений целостности данных отношения User

```
1 ALTER TABLE "User"  
2     ADD CONSTRAINT "pk:user.id" PRIMARY KEY (id),  
3     ALTER COLUMN id SET DEFAULT gen_random_uuid(),  
4     ADD CONSTRAINT "chk:user.email:notnull" CHECK (email IS NOT NULL),  
5     ADD CONSTRAINT "chk:user.email:length" CHECK (length(email) < 128),  
6     ADD CONSTRAINT "chk:user.email:regexp" CHECK (  
7         email ~ '^[A-Za-z0-9._%~]+@[A-Za-z0-9.-]+[.][A-Za-z]+$'),  
8     ADD CONSTRAINT "uq:user.email" UNIQUE (email),  
9     ADD CONSTRAINT "chk:user.phone_number:notnull" CHECK (  
10        phone_number IS NOT NULL),  
11    ADD CONSTRAINT "chk:user.phone_number:length" CHECK (  
12        length(phone_number) < 32),  
13    ADD CONSTRAINT "chk:user.phone_number:regexp" CHECK (  
14        phone_number ~ '^\+\\d{10,15}$'),  
15    ADD CONSTRAINT "uq:user.phone_number" UNIQUE (phone_number),  
16    ADD CONSTRAINT "chk:user.password:notnull" CHECK (  
17        "password" IS NOT NULL),  
18    ADD CONSTRAINT "chk:user.first_name:notnull" CHECK (  
19        first_name IS NOT NULL),  
20    ADD CONSTRAINT "chk:user.first_name:length" CHECK (  
21        length(first_name) < 128),  
22    ADD CONSTRAINT "chk:user.first_name:regexp" CHECK (  
23
```

```

23 first_name ~ ' ^([a-zA-Za-яА-ЯёЁ]+(?:-[a-zA-Za-яА-ЯёЁ]+)?)$' ),
24 ADD CONSTRAINT "chk:user.last_name:notnull" CHECK (
25     last_name IS NOT NULL),
26 ADD CONSTRAINT "chk:user.last_name:length" CHECK (
27     length(last_name) < 128),
28 ADD CONSTRAINT "chk:user.last_name:regexp" CHECK (
29     last_name ~ ' ^([a-zA-Za-яА-ЯёЁ]+(?:-[a-zA-Za-яА-ЯёЁ]+)?)$' ),
30 ADD CONSTRAINT "chk:user.gender:notnull" CHECK (gender IS NOT NULL),
31 ADD CONSTRAINT "chk:user.gender:length" CHECK (length(gender) < 32),
32 ADD CONSTRAINT "chk:user.gender:regexp" CHECK (
33     gender IN ( 'мужской', 'женский' )),
34 ADD CONSTRAINT "chk:user.birth_date:notnull" CHECK (
35     birth_date IS NOT NULL),
36 ADD CONSTRAINT "chk:user.birth_date" CHECK (
37     birth_date BETWEEN
38         CURRENT_DATE - INTERVAL '120_years' AND
39         CURRENT_DATE - INTERVAL '14_years'
40 );

```

Листинг 15 – Реализация ограничений целостности данных отношения  
UserRole

```

1 ALTER TABLE "UserRole"
2     ADD CONSTRAINT "pk:user_role.id" PRIMARY KEY (id),
3     ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4     ADD CONSTRAINT "fk:user_role.user_id" FOREIGN KEY (user_id)
5         REFERENCES "User"(id) ON DELETE CASCADE,
6     ADD CONSTRAINT "uq:user_role.user_id" UNIQUE (user_id),
7     ADD CONSTRAINT "chk:user_role.role:notnull" CHECK ("role" IS NOT NULL),
8     ADD CONSTRAINT "chk:user_role.role:length" CHECK (length("role") < 32),
9     ADD CONSTRAINT "chk:user_role.role:regexp" CHECK (
10         "role" IN ( 'клиент', 'тренер', 'администратор' )
11 );

```

Листинг 16 – Реализация ограничений целостности данных отношения  
Trainer

```

1 ALTER TABLE "Trainer"
2     ADD CONSTRAINT "pk:trainer.id" PRIMARY KEY (id),
3     ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4     ADD CONSTRAINT "fk:trainer.user_id" FOREIGN KEY (user_id)
5         REFERENCES "User"(id) ON DELETE CASCADE,
6     ADD CONSTRAINT "uq:trainer.user_id" UNIQUE (user_id),
7     ALTER COLUMN "description" SET DEFAULT 'Нет_описания.',
8     ADD CONSTRAINT "chk:trainer.description:notnull" CHECK (
9         description IS NOT NULL),
10    ADD CONSTRAINT "chk:trainer.description:length" CHECK (
11        length(description) < 512);

```



Листинг 17 – Реализация ограничений целостности данных отношения  
Specialization

```
1 ALTER TABLE "Specialization"
2   ADD CONSTRAINT "pk:specialization.id" PRIMARY KEY (id),
3   ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4   ADD CONSTRAINT "chk:specialization.name:notnull" CHECK (
5     name IS NOT NULL),
6   ADD CONSTRAINT "chk:specialization.name:length" CHECK (
7     length(name) < 128),
8   ADD CONSTRAINT "uq:specialization.name" UNIQUE (name);
```

Листинг 18 – Реализация ограничений целостности данных отношения  
TrainerSpecialization

```
1 ALTER TABLE "TrainerSpecialization"
2   ADD CONSTRAINT "pk:trainer_specialization.id" PRIMARY KEY (id),
3   ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4   ADD CONSTRAINT "fk:trainer_specialization.trainer_id"
5     FOREIGN KEY (trainer_id) REFERENCES "Trainer"(id) ON DELETE CASCADE,
6   ADD CONSTRAINT "fk:trainer_specialization.specialization_id"
7     FOREIGN KEY (specialization_id)
8     REFERENCES "Specialization"(id) ON DELETE CASCADE,
9   ADD CONSTRAINT "uq:trainer_specialization.trainer+specialization"
10    UNIQUE (trainer_id, specialization_id),
11   ALTER COLUMN years SET DEFAULT 0,
12   ADD CONSTRAINT "chk:trainer_specialization.years:notnull" CHECK (
13     years IS NOT NULL),
14   ADD CONSTRAINT "chk:specialization.name:unsigned" CHECK (years >= 0);
```

Листинг 19 – Реализация ограничений целостности данных отношения  
TrainingRoom

```
1 ALTER TABLE "TrainingRoom"
2   ADD CONSTRAINT "pk:training_room.id" PRIMARY KEY (id),
3   ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4   ADD CONSTRAINT "chk:training_room.name:notnull" CHECK (name IS NOT NULL),
5   ADD CONSTRAINT "chk:training_room.name:length" CHECK (length(name) < 64),
6   ADD CONSTRAINT "uq:training_room.name" UNIQUE (name),
7   ADD CONSTRAINT "chk:training_room.capacity:notnull" CHECK (
8     capacity IS NOT NULL),
9   ADD CONSTRAINT "chk:training_room.capacity" CHECK (capacity > 0);
```

Листинг 20 – Реализация ограничений целостности данных отношения Order

```
1 ALTER TABLE "Order"
2   ADD CONSTRAINT "pk:order.id" PRIMARY KEY (id),
3   ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4   ADD CONSTRAINT "fk:order.user_id" FOREIGN KEY (user_id)
```

```

5 REFERENCES "User"(id) ON DELETE SET NULL,
6 ALTER COLUMN "date" SET DEFAULT CURRENT_TIMESTAMP,
7 ADD CONSTRAINT "chk:order.date:notnull" CHECK ("date" IS NOT NULL),
8 ALTER COLUMN total_price SET DEFAULT 0.0,
9 ADD CONSTRAINT "chk:order.total_price:notnull" CHECK (
10     total_price IS NOT NULL),
11 ADD CONSTRAINT "chk:order.total_price:unsigned" CHECK (total_price >= 0);

```

Листинг 21 – Реализация ограничений целостности данных отношения  
OrderItem

```

1 ALTER TABLE "OrderItem"
2     ADD CONSTRAINT "pk:order_item.id" PRIMARY KEY (id),
3     ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4     ADD CONSTRAINT "fk:order_item.order_id" FOREIGN KEY (order_id)
5         REFERENCES "Order"(id) ON DELETE CASCADE,
6     ADD CONSTRAINT "fk:order_item.membership_type_id"
7         FOREIGN KEY (membership_type_id)
8         REFERENCES "MembershipType"(id) ON DELETE CASCADE;

```

Листинг 22 – Реализация ограничений целостности данных отношения  
Payment

```

1 ALTER TABLE "Payment"
2     ADD CONSTRAINT "pk:payment.id" PRIMARY KEY (id),
3     ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4     ADD CONSTRAINT "fk:payment.order_id" FOREIGN KEY (order_id)
5         REFERENCES "Order"(id) ON DELETE SET NULL,
6     ADD CONSTRAINT "chk:payment.transaction_id:notnull" CHECK (
7         transaction_id IS NOT NULL),
8     ADD CONSTRAINT "chk:payment.transaction_id:length" CHECK (
9         length(transaction_id) < 256),
10    ADD CONSTRAINT "uq:payment.transaction_id" UNIQUE (transaction_id),
11    ALTER COLUMN "date" SET DEFAULT CURRENT_TIMESTAMP,
12    ADD CONSTRAINT "chk:payment.date:notnull" CHECK ("date" IS NOT NULL),
13    ALTER COLUMN "method" SET DEFAULT 'наличные',
14    ADD CONSTRAINT "chk:payment.method:notnull" CHECK ("method" IS NOT NULL),
15    ADD CONSTRAINT "chk:payment.method:length" CHECK (length("method") < 64),
16    ADD CONSTRAINT "chk:payment.method:regexp" CHECK (
17        "method" IN ('наличные', 'кредитная_карта', 'банковский_перевод')),
18    ALTER COLUMN "gateway" SET DEFAULT NULL,
19    ADD CONSTRAINT "chk:payment.gateway:length" CHECK (
20        length("gateway") < 64),
21    ALTER COLUMN "status" SET DEFAULT 'ожидает',
22    ADD CONSTRAINT "chk:payment.status:notnull" CHECK ("status" IS NOT NULL),
23    ADD CONSTRAINT "chk:payment.status:length" CHECK (length("status") < 64),
24    ADD CONSTRAINT "chk:payment.status" CHECK (
25        status IN ('ожидает', 'оплачен', 'отменен'));

```

Листинг 23 – Реализация ограничений целостности данных отношения  
MembershipType

```
1 ALTER TABLE "MembershipType"
2 ADD CONSTRAINT "pk:membership_type.id" PRIMARY KEY (id),
3 ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4 ADD CONSTRAINT "chk:membership_type.name:notnull" CHECK (
5 name IS NOT NULL),
6 ADD CONSTRAINT "chk:membership_type.name:length" CHECK (
7 length(name) < 128),
8 ADD CONSTRAINT "uq:membership_type.name" UNIQUE (name),
9 ALTER COLUMN price SET DEFAULT 0.0,
10 ADD CONSTRAINT "chk:membership_type.price:notnull" CHECK (
11 price IS NOT NULL),
12 ADD CONSTRAINT "chk:membership_type.price:unsigned" CHECK (price >= 0.0),
13 ALTER COLUMN sessions SET DEFAULT 1,
14 ADD CONSTRAINT "chk:membership_type.sessions:notnull" CHECK (
15 sessions IS NOT NULL),
16 ADD CONSTRAINT "chk:membership_type.sessions:unsigned" CHECK (
17 sessions > 0),
18 ALTER COLUMN days SET DEFAULT 1,
19 ADD CONSTRAINT "chk:membership_type.days:notnull" CHECK (
20 days IS NOT NULL),
21 ADD CONSTRAINT "chk:membership_type.days:unsigned" CHECK (days > 0);
```

Листинг 24 – Реализация ограничений целостности данных отношения  
Training

```
1 ALTER TABLE "Training"
2 ADD CONSTRAINT "pk:training.id" PRIMARY KEY (id),
3 ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4 ADD CONSTRAINT "fk:training.specialization_id"
5 FOREIGN KEY (specialization_id)
6 REFERENCES "Specialization"(id) ON DELETE SET NULL,
7 ADD CONSTRAINT "fk:training.room_id" FOREIGN KEY (room_id)
8 REFERENCES "TrainingRoom"(id) ON DELETE SET NULL,
9 ADD CONSTRAINT "fk:training.trainer_id" FOREIGN KEY (trainer_id)
10 REFERENCES "Trainer"(id) ON DELETE SET NULL,
11 ALTER COLUMN "date" SET DEFAULT CURRENT_TIMESTAMP,
12 ADD CONSTRAINT "chk:training.date:notnull" CHECK ("date" IS NOT NULL);
```

Листинг 25 – Реализация ограничений целостности данных отношения  
Membership

```
1 ALTER TABLE "Membership"
2 ADD CONSTRAINT "pk:membership.id" PRIMARY KEY (id),
3 ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4 ADD CONSTRAINT "fk:membership.user_id" FOREIGN KEY (user_id)
5 REFERENCES "User"(id) ON DELETE CASCADE,
```

```

6  ADD CONSTRAINT "fk:membership.order_id" FOREIGN KEY (order_id)
7      REFERENCES "Order"(id) ON DELETE SET NULL,
8  ADD CONSTRAINT "fk:membership.membership_type_id"
9      FOREIGN KEY (membership_type_id)
10     REFERENCES "MembershipType"(id) ON DELETE SET NULL,
11  ALTER COLUMN start_date SET DEFAULT NULL,
12  ALTER COLUMN end_date SET DEFAULT NULL,
13  ADD CONSTRAINT "chk:membership.dates:order" CHECK(
14      (start_date IS NULL AND end_date IS NULL) OR (start_date IS NOT NULL
15      AND end_date IS NOT NULL AND start_date <= end_date)),
16  ADD CONSTRAINT "uq:membership.membership_type+user"
17      UNIQUE(membership_type_id, user_id),
18  ALTER COLUMN available_sessions SET DEFAULT 0,
19  ADD CONSTRAINT "chk:membership.available_sessions:notnull" CHECK (
20      available_sessions IS NOT NULL),
21  ADD CONSTRAINT "chk:membership.available_sessions:unsigned" CHECK (
22      available_sessions >= 0);

```

Листинг 26 – Реализация ограничений целостности данных отношения Attendance

```

1  ALTER TABLE "Attendance"
2      ADD CONSTRAINT "pk:attendance.id" PRIMARY KEY (id),
3      ALTER COLUMN id SET DEFAULT gen_random_uuid(),
4      ADD CONSTRAINT "fk:attendance.membership_id" FOREIGN KEY (membership_id)
5          REFERENCES "Membership"(id) ON DELETE CASCADE,
6      ADD CONSTRAINT "fk:attendance.training_id" FOREIGN KEY (training_id)
7          REFERENCES "Training"(id) ON DELETE SET NULL,
8      ADD CONSTRAINT "uq:attendance.membership+training"
9          UNIQUE (membership_id, training_id),
10     ALTER COLUMN status SET DEFAULT 'ожидает',
11     ADD CONSTRAINT "chk:attendance.status:notnull" CHECK (
12         status IS NOT NULL),
13     ADD CONSTRAINT "chk:attendance.status:length" CHECK (
14         length(status) < 64),
15     ADD CONSTRAINT "chk:attendance.status:regexp" CHECK (
16         status IN ('посетил', 'отсутствовал', 'ожидает'));

```

### 3.2.1 Реализация триггеров

Реализации триггеров для: автоматического обновления количества доступных занятий после посещения тренировки, записи на тренировку с учетом проверки количества доступных мест, создания абонемента или продления его после успешного платежа – представлены на листингах 27, 28 и 29 соответственно.

Листинг 27 – Реализация триггера для автоматического обновления количества доступных занятий после посещения тренировки

```

1 CREATE OR REPLACE FUNCTION update_available_sessions()
2 RETURNS TRIGGER AS $$
3 BEGIN
4     IF (NEW.status = 'посетил' OR NEW.status = 'отсутствовал')
5         AND (OLD.status <> 'посетил' AND OLD.status <> 'отсутствовал')
6     THEN
7         UPDATE "Membership"
8         SET available_sessions = available_sessions - 1
9         WHERE id = NEW.membership_id;
10    ELSIF (NEW.status = 'ожидает')
11        AND (OLD.status = 'посетил' OR OLD.status = 'отсутствовал')
12    THEN
13        UPDATE "Membership"
14        SET available_sessions = available_sessions + 1
15        WHERE id = NEW.membership_id;
16    END IF;
17    RETURN NEW;
18 END;
19 $$ LANGUAGE plpgsql;
20 CREATE OR REPLACE TRIGGER attendance_status_update
21 AFTER UPDATE OF status ON "Attendance"
22 FOR EACH ROW
23 EXECUTE FUNCTION update_available_sessions();

```

Листинг 28 – Реализация триггера для записи на тренировку с учетом проверки количества доступных мест

```

1 CREATE OR REPLACE FUNCTION check_room_capacity()
2 RETURNS TRIGGER AS $$
3 BEGIN
4     IF (TG_OP = 'INSERT') OR (TG_OP = 'UPDATE' AND NEW.training_id IS
5         DISTINCT FROM OLD.training_id)
6     THEN
7         IF (SELECT COUNT(*) FROM "Attendance" WHERE training_id =
8             NEW.training_id) >= (SELECT capacity FROM "TrainingRoom" WHERE id
9             = (SELECT room_id FROM public.training WHERE id =
10                NEW.training_id))
11        THEN RAISE EXCEPTION 'Тренировка уже заполнена';
12        END IF;
13    END IF;
14    RETURN NEW;
15 END; $$ LANGUAGE plpgsql;
16 CREATE OR REPLACE TRIGGER check_room_capacity_trigger
17 BEFORE INSERT ON "Attendance" FOR EACH ROW
18 EXECUTE FUNCTION check_room_capacity();

```

Листинг 29 – Реализация триггера для создания абонемента или продления его после успешного платежа

```
1 CREATE OR REPLACE FUNCTION create_membership_after_payment()
2 RETURNS TRIGGER AS $$
3 DECLARE item RECORD;
4 BEGIN
5     IF (TG_OP = 'INSERT' AND NEW.status = 'оплачено') OR (TG_OP = 'UPDATE'
6         AND NEW.status = 'оплачено' AND OLD.status IS DISTINCT FROM
7         'оплачено')
8     THEN
9         FOR item IN
10             SELECT oi.membership_type_id, o.user_id, mt.duration_days,
11                 mt.number_of_sessions
12             FROM "OrderItem" oi
13             JOIN "Order" o ON o.id = oi.order_id
14             JOIN "MembershipType" mt ON mt.id = oi.membership_type_id
15             WHERE o.id = NEW.order_id
16         LOOP
17             IF EXISTS (SELECT 1 FROM "Membership" WHERE user_id =
18                 item.user_id AND membership_type_id = item.membership_type_id)
19             THEN -- обновление
20                 UPDATE "Membership"
21                 SET
22                     end_date = CURRENT_DATE + INTERVAL '1_day' *
23                     item.duration_days,
24                     available_sessions = available_sessions +
25                     item.number_of_sessions,
26                     order_id = NEW.order_id
27                 WHERE user_id = item.user_id
28                 AND membership_type_id = item.membership_type_id;
29             ELSE
30                 INSERT INTO public.membership (id, user_id,
31                     membership_type_id, order_id,
32                     start_date, end_date, available_sessions)
33                 VALUES (gen_random_uuid(), item.user_id,
34                     item.membership_type_id, NEW.order_id, CURRENT_DATE,
35                     CURRENT_DATE + INTERVAL '1_day' * item.duration_days,
36                     item.number_of_sessions);
37             END IF;
38         END LOOP;
39     END IF;
40     RETURN NEW;
41 END;
42 $$ LANGUAGE plpgsql;
43 CREATE OR REPLACE TRIGGER create_membership_after_payment_trigger
44 AFTER INSERT OR UPDATE ON "Payment" FOR EACH ROW
45 EXECUTE FUNCTION create_membership_after_payment();
```

### 3.2.2 Создание ролевой модели

Реализация ролевой модели представлена на листингах 30, 31 и 32.

Листинг 30 – Реализация роли Guest (гость)

```
1 CREATE ROLE guest WITH LOGIN;  
2 GRANT EXECUTE ON FUNCTION register_user(  
3     UUID, TEXT, TEXT, TEXT, TEXT, TEXT, DATE, TEXT  
4 ) TO guest;  
5 GRANT EXECUTE ON FUNCTION login_user(TEXT, TEXT) TO guest;
```

Листинг 31 – Реализация роли Admin (администратор)

```
1 CREATE ROLE admin WITH LOGIN PASSWORD 'admin' SUPERUSER BYPASSRLS;  
2 GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin;
```

Листинг 32 – Реализация ролей Client (клиент) и Trainer (тренер)

```
1 CREATE ROLE client WITH LOGIN PASSWORD 'client';  
2 CREATE ROLE trainer WITH LOGIN PASSWORD 'trainer';  
3 GRANT client TO trainer;  
4 GRANT ALL ON "User" TO client;  
5 GRANT ALL ON "User" TO trainer;  
6 ALTER TABLE "User" ENABLE ROW LEVEL SECURITY;  
7 CREATE POLICY personal_client_trainer_user_policy ON "User" FOR ALL TO  
8     client, trainer USING (id = current_user_id());  
8 CREATE POLICY public_client_trainer_user_policy ON "User" FOR SELECT TO  
9     client, trainer USING (TRUE);  
9 GRANT ALL ON "Trainer" TO client;  
10 GRANT ALL ON "Trainer" TO trainer;  
11 ALTER TABLE "Trainer" ENABLE ROW LEVEL SECURITY;  
12 CREATE POLICY public_client_trainer_trainer_policy ON "Trainer" FOR SELECT  
13     TO client, trainer USING (TRUE);  
13 CREATE POLICY personal_trainer_trainer_policy ON "Trainer" FOR ALL TO  
14     trainer USING (user_id = current_user_id());  
14 GRANT ALL ON "Membership" TO client;  
15 GRANT ALL ON "Membership" TO trainer;  
16 ALTER TABLE "Membership" ENABLE ROW LEVEL SECURITY;  
17 CREATE POLICY client_trainer_membership_policy ON "Membership" FOR ALL TO  
18     client, trainer USING (user_id = current_user_id());  
18 CREATE POLICY trainer_membership_policy ON "Membership" FOR SELECT TO  
19     trainer USING (TRUE);  
19 GRANT ALL ON "Order" TO client;  
20 GRANT ALL ON "Order" TO trainer;  
21 ALTER TABLE "Order" ENABLE ROW LEVEL SECURITY;  
22 CREATE POLICY client_trainer_order_policy ON "Order" FOR ALL TO client,  
23     trainer USING (user_id = current_user_id());  
23 GRANT ALL ON "OrderItem" TO client;
```

```

24 GRANT ALL ON "OrderItem" TO trainer;
25 ALTER TABLE "OrderItem" ENABLE ROW LEVEL SECURITY;
26 CREATE POLICY client_trainer_order_item_policy ON "OrderItem" FOR ALL TO
    client, trainer USING ( order_id IN (
27     SELECT id FROM "Order" WHERE user_id = current_user_id()));
28 GRANT ALL ON "Payment" TO client;
29 GRANT ALL ON "Payment" TO trainer;
30 ALTER TABLE "Payment" ENABLE ROW LEVEL SECURITY;
31 CREATE POLICY client_trainer_payment_policy ON "Payment" FOR ALL TO client,
    trainer USING ( order_id IN (
32     SELECT id FROM "Order" WHERE user_id = current_user_id()));
33 GRANT ALL ON "Attendance" TO client;
34 GRANT ALL ON "Attendance" TO trainer;
35 ALTER TABLE "Attendance" ENABLE ROW LEVEL SECURITY;
36 CREATE POLICY client_trainer_attendance_policy ON "Attendance" FOR ALL TO
    client, trainer USING ( membership_id IN (
37     SELECT id FROM "Membership" WHERE user_id = current_user_id()));
38 CREATE POLICY trainer_attendance_policy ON "Attendance" FOR ALL TO trainer
    USING (training_id IN (
39     SELECT id FROM "Training" WHERE trainer_id = (
40     SELECT id FROM "Trainer" WHERE user_id = current_user_id())));
41 GRANT SELECT ON "Specialization" TO client;
42 GRANT SELECT ON "Specialization" TO trainer;
43 GRANT SELECT ON "TrainerSpecialization" TO client;
44 GRANT SELECT ON "TrainerSpecialization" TO trainer;
45 GRANT ALL ON "Training" TO trainer;
46 GRANT ALL ON "Training" TO client;
47 ALTER TABLE "Training" ENABLE ROW LEVEL SECURITY;
48 CREATE POLICY client_training_select_policy ON "Training" FOR SELECT TO
    client, trainer USING (TRUE);
49 CREATE POLICY trainer_training_update_policy ON "Training" FOR ALL TO
    trainer USING ( trainer_id = (
50     SELECT id FROM public.trainer WHERE user_id = current_user_id()));
51 GRANT SELECT ON "MembershipType" TO client;
52 GRANT SELECT ON "MembershipType" TO trainer;
53 GRANT SELECT ON "TrainingRoom" TO client;
54 GRANT SELECT ON "TrainingRoom" TO trainer;

```

### 3.3 Тестирование

На рисунках 12 и 13 изображено тестирование триггера для создания абонемента или продления его после успешного платежа.

На рисунках 14 и 15 изображено тестирование триггеров для: автоматического обновления количества доступных занятий после посещения тренировки, записи на тренировку с учетом проверки количества доступных мест.



Рисунок 12 – Обновления статуса записи таблицы Payment

Рисунок 13 – Добавление записи в таблицу Payment

fitnessClub=# select * from "Training";									
id	specialization_id	room_id	trainer_id	date					
a2f115f7-2d8c-427b-a312-565ea4495acb	8402d7cf-508f-4882-9941-3f0ca1a114d4	aaal2674-7b6b-404b-be55-cce491dfeff4	c9911674-7b6b-404b-be55-cce491dfe6b4	2025-04-09 10:00:00					
(1 row)									
fitnessClub=# select * from "TrainingRoom";									
id	name	capacity							
aaal2674-7b6b-404b-be55-cce491dfeff4	3a11	1							
(1 row)									
fitnessClub=# select * from "Membership";									
id	user_id	order_id	membership_type_id	start_date	end_date	available_sessions			
a082aeaa-9e32-4f07-bf6f-acbbe7a2119	c221d3fb-f664-4592-a9e7-2da7ccfb2056	110c07ff-a6c5-4799-9ac8-61da6f572e25	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	4			
99c06953-abe8-412b-b7d4-0f571ab27212	aba9dde6-6438-41c2-a7c6-84171483d174	d204d435-ecf7-40b5-9238-f41c9a005911	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	4			
(2 rows)									
fitnessClub=# insert into "Attendance" (membership_id, training_id) values('a082aeaa-9e32-4f07-bf6f-acbbe7a2119', 'a2f115f7-2d8c-427b-a312-565ea4495acb');									
INSERT 0 1									
fitnessClub=# select * from "Membership";									
id	user_id	order_id	membership_type_id	start_date	end_date	available_sessions			
a082aeaa-9e32-4f07-bf6f-acbbe7a2119	c221d3fb-f664-4592-a9e7-2da7ccfb2056	110c07ff-a6c5-4799-9ac8-61da6f572e25	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	4			
99c06953-abe8-412b-b7d4-0f571ab27212	aba9dde6-6438-41c2-a7c6-84171483d174	d204d435-ecf7-40b5-9238-f41c9a005911	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	4			
(2 rows)									
fitnessClub=# update "Attendance" set status = 'none';									
UPDATE 1									
fitnessClub=# select * from "Membership";									
id	user_id	order_id	membership_type_id	start_date	end_date	available_sessions			
99c06953-abe8-412b-b7d4-0f571ab27212	aba9dde6-6438-41c2-a7c6-84171483d174	d204d435-ecf7-40b5-9238-f41c9a005911	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	4			
a082aeaa-9e32-4f07-bf6f-acbbe7a2119	c221d3fb-f664-4592-a9e7-2da7ccfb2056	110c07ff-a6c5-4799-9ac8-61da6f572e25	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	3			
(2 rows)									
fitnessClub=# update "Attendance" set status = 'occupied';									
UPDATE 1									
fitnessClub=# select * from "Membership";									
id	user_id	order_id	membership_type_id	start_date	end_date	available_sessions			
99c06953-abe8-412b-b7d4-0f571ab27212	aba9dde6-6438-41c2-a7c6-84171483d174	d204d435-ecf7-40b5-9238-f41c9a005911	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	4			
a082aeaa-9e32-4f07-bf6f-acbbe7a2119	c221d3fb-f664-4592-a9e7-2da7ccfb2056	110c07ff-a6c5-4799-9ac8-61da6f572e25	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	4			
(2 rows)									
fitnessClub=# update "Attendance" set status = 'occupied';									
UPDATE 1									
fitnessClub=# select * from "Membership";									
id	user_id	order_id	membership_type_id	start_date	end_date	available_sessions			
99c06953-abe8-412b-b7d4-0f571ab27212	aba9dde6-6438-41c2-a7c6-84171483d174	d204d435-ecf7-40b5-9238-f41c9a005911	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	4			
a082aeaa-9e32-4f07-bf6f-acbbe7a2119	c221d3fb-f664-4592-a9e7-2da7ccfb2056	110c07ff-a6c5-4799-9ac8-61da6f572e25	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	4			
(2 rows)									
fitnessClub=# update "Attendance" set status = 'occupied';									
UPDATE 1									
fitnessClub=# select * from "Membership";									
id	user_id	order_id	membership_type_id	start_date	end_date	available_sessions			
99c06953-abe8-412b-b7d4-0f571ab27212	aba9dde6-6438-41c2-a7c6-84171483d174	d204d435-ecf7-40b5-9238-f41c9a005911	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	4			
a082aeaa-9e32-4f07-bf6f-acbbe7a2119	c221d3fb-f664-4592-a9e7-2da7ccfb2056	110c07ff-a6c5-4799-9ac8-61da6f572e25	d1011674-7b6b-404b-be55-cce491dfeff4	2025-04-09	2025-05-09	3			
(2 rows)									

Рисунок 14 – Добавление записи в таблицу Attendance и обновление ее статуса

```
fitnessClub=# insert into "Attendance" (membership_id, training_id) values('99c06953-abe8-412b-b7d4-0f571ab27212', 'a2f115f7-2d8c-427b-a312-565ea4495acb');
ERROR: Тренировка уже заполнена
CONTEXT: PL/pgSQL function check_room_capacity() line 22 at RAISE
```

Рисунок 15 – Добавление записи в таблицу Attendance при отсутствии доступных мест на тренировку

### 3.4 Интерфейс для взаимодействия с базой данных

Для взаимодействия с базой данных реализовано мобильное приложение для платформы iOS. Визуальное представление интерфейса приведено на рисунках ??- ?? в приложении А.

#### Вывод

В данном разделе представлены выбранные средства реализации, приведены реализации сущностей, ограничений целостности данных, ролевой модели и триггеров. Также описано тестирование триггеров и продемонстрирован интерфейс, обеспечивающий взаимодействие с базой данных.

## 4 Исследовательская часть

В данном разделе представлены результаты исследований, проведённых для оценки производительности системы при увеличении объёма данных, анализа скорости получения ответа в зависимости от количества одновременных запросов, а также сравнения скорости обработки с использованием кэширования и без него.

### 4.1 Технические характеристики

Исследования проводилось на устройстве со следующими техническими характеристиками:

- операционная система macOS 14.6.1 [12];
- оперативная память (RAM) 16 ГБ;
- процессор (CPU) 2 ГГц 4-ядерный Intel Core i5 (8 логических ядер) [13].

Во время проведения исследования, устройство было подключено к сети электропитания и не было нагружено сторонними приложениями, за исключением встроенных приложений окружения.

### Влияние объема данных на время выполнения

В таблице 15 приведены результаты исследования времени выполнения операций для различных количеств пользователей в секундах. На рисунке 16 изображено графическое представление данных из таблицы 15.

Таблица 15 – Время выполнения операций для различных количеств пользователей

Количество пользователей	INSERT	UPDATE	DELETE	SELECT
10	0.008671	0.001736	0.001214	0.000585
25	0.011783	0.004547	0.003441	0.000304
50	0.018182	0.009041	0.005436	0.000386
100	0.037794	0.017999	0.011490	0.000625
250	0.123677	0.052245	0.029886	0.000983
500	0.185770	0.093386	0.055787	0.001626
750	0.204760	0.179636	0.091073	0.002351
1000	0.252232	0.206964	0.126073	0.003015

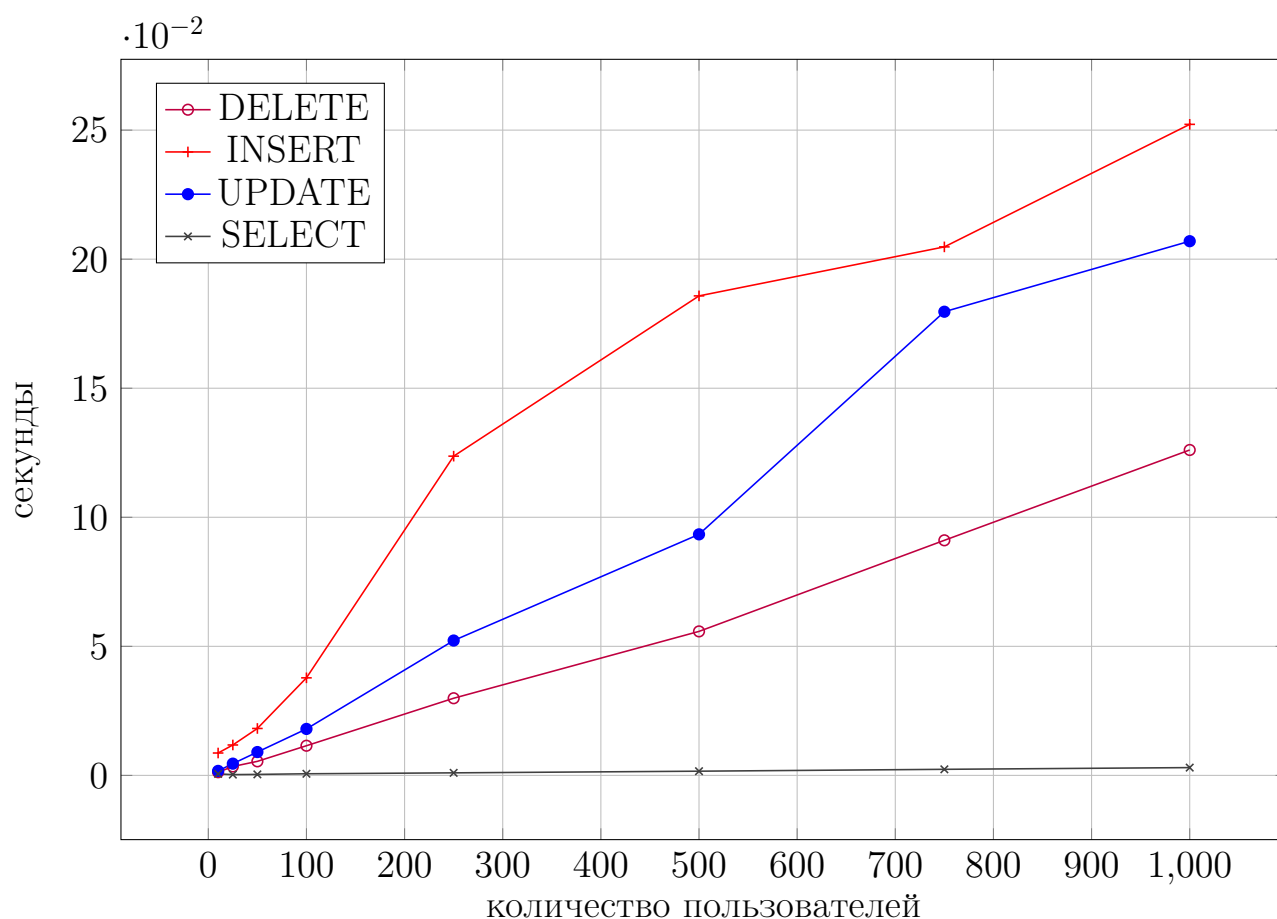


Рисунок 16 – Зависимость времени выполнения операций от количества пользователей

## Влияние количества одновременных запросов на отклик

В рамках исследования была оценена производительность выборки данных из таблицы User при различных уровнях параллельной нагрузки на базу данных. Каждый тест длился 10 секунд, в течение которых пользователи базы данных (клиенты) выполняли запросы.

В таблице 16 приведены результаты исследования времени отклика от количества клиентов в миллисекундах. На рисунке 17 изображено графическое представление данных из таблицы 16.

Таблица 16 – Результаты нагрузочного тестирования

Количество клиентов	Среднее время отклика (мс)	Количество запросов
1	0.272817	36 550
10	1.516456	65 898
25	5.964658	41 914
50	11.333167	44 124
75	13.321161	56 332

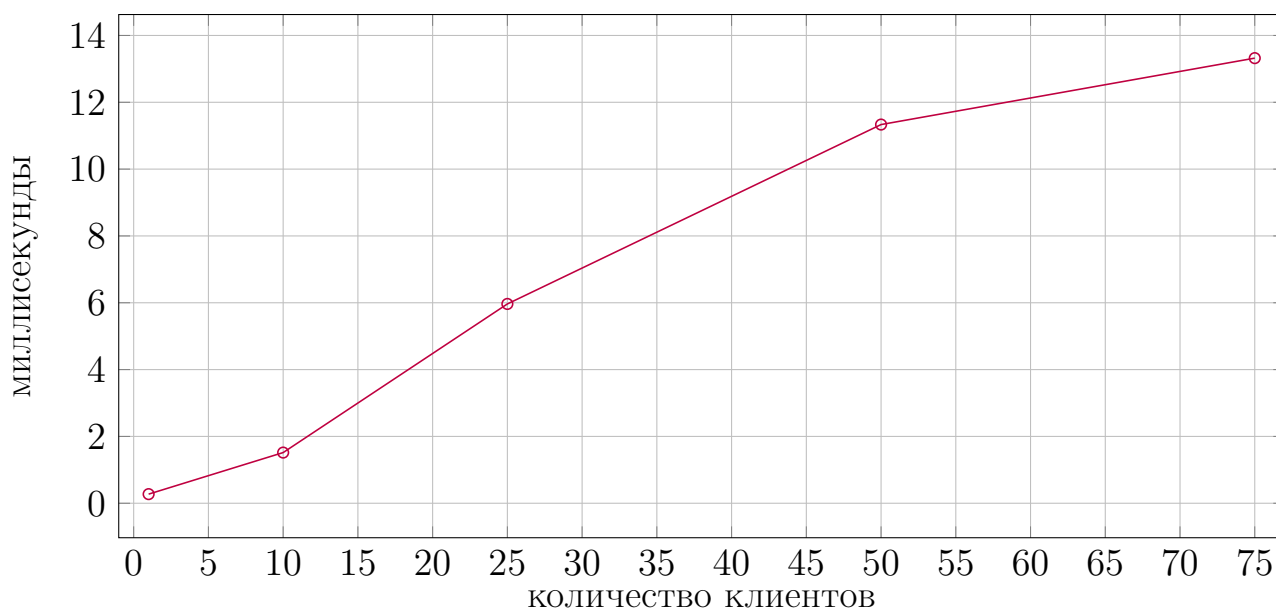


Рисунок 17 – Зависимость времени отклика от количества клиентов

## Влияние кэширования на скорость запросов

В рамках исследования была оценена производительность запросов к данным, получаемым из базы данных PostgreSQL и Redis-кэша.

Основной задачей было извлечение информации о пользователях, родившихся после 1 января 2003 года и имеющих женский пол, с последующей сортировкой по дате рождения.

Для каждого из источников данных были замерены времена отклика на запросы. Каждый запрос к кэшу и базе данных выполнялся 30 раз. Исследование проводилось на протяжении нескольких минут, с интервалом в 5 секунд между запросами. Кэш в Redis хранился в течение 10 секунд.

Среднее время выполнения запросов к Redis-кэшу (Кэш) составило 0.00188 секунды, для базы данных PostgreSQL (БД) – 0.00246 секунды. На рисунке 18 изображено графическое представление полученных данных.

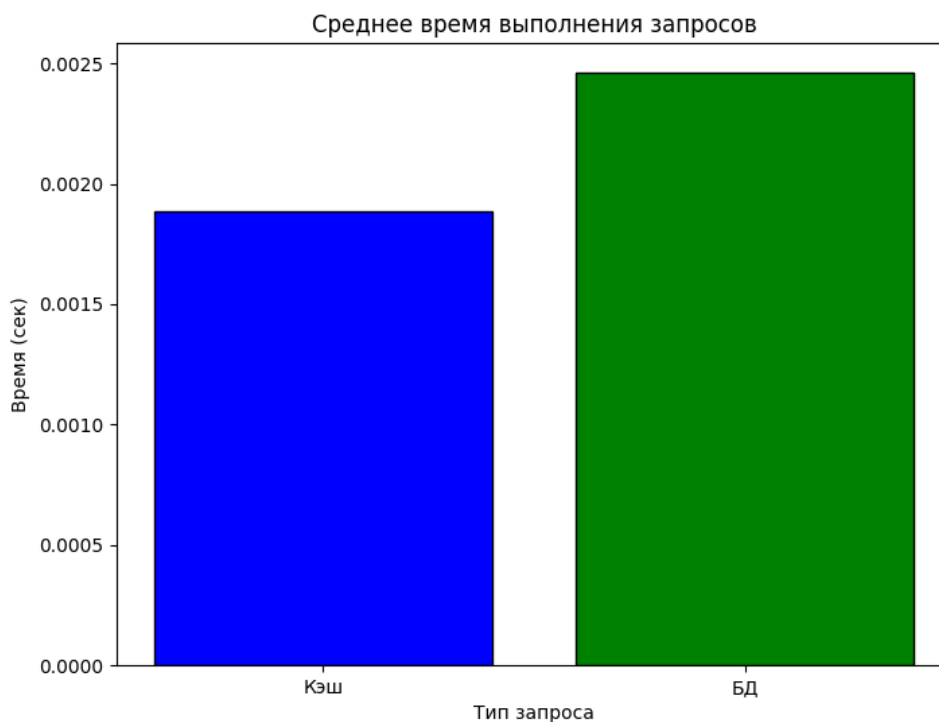


Рисунок 18 – Сравнение времени выполнения запросов к кэшу и базе данных

## Вывод

Результаты первого исследования показали, что с увеличением числа записей в таблице пользователей время выполнения операций увеличивается:

- время выполнения операции INSERT увеличивается с 0.0087 секунд при 10 пользователях до 0.2522 секунд при 1000 пользователях;
- время выполнения операции UPDATE увеличивается с 0.0017 секунд при 10 пользователях до 0.2070 секунд при 1000 пользователей;
- время выполнения операции DELETE увеличивается с 0.0012 секунд при 10 пользователях до 0.1261 секунд при 1000 пользователей;
- время выполнения операции SELECT увеличивается с 0.0006 секунд при 10 пользователях до 0.0030 секунд при 1000 пользователей.

Время выполнения операций значительно увеличивается при росте количества записей в таблице, особенно для операций изменения данных (INSERT, UPDATE, DELETE), в то время как операция SELECT остается сравнительно стабильной.

Результаты второго исследования показали, что с увеличением количества клиентов время отклика на запросы также увеличивается. Так, при 10 клиентах среднее время отклика составило 1.5165 миллисекунды, а при 75 клиентах – 13.3212 миллисекунды. Это свидетельствует о том, что с ростом числа одновременных запросов наблюдается значительное увеличение времени отклика, что указывает на увеличение нагрузки на базу данных.

Результаты третьего исследования показали, что кэширование ускоряет выполнение запросов. Среднее время выполнения запроса к Redis-кэшу составило 0.00188 секунды, что на 0.00058 секунды быстрее, чем запросы к базе данных PostgreSQL, где среднее время составило 0.00246 секунды. Это подтверждает эффективность использования кэширования для ускорения обработки данных.



## ЗАКЛЮЧЕНИЕ

Цель курсовой работы, заключающаяся в разработке базы данных для хранения и обработки данных фитнес-клуба, была успешно достигнута. В ходе работы были выполнены все поставленные задачи.

Был проведен анализ предметной области и формализована задача, что позволило точно определить требования к базе данных. Разработана структура базы данных с определением ролей пользователей системы. После анализа различных моделей данных была выбрана наиболее оптимальная модель для данного проекта.

В процессе проектирования базы данных были спроектированы необходимые сущности и их взаимосвязи, а также ролевая модель для разграничения прав доступа. Также были разработаны триггеры для автоматического обновления данных и внедрены ограничения целостности данных, что обеспечило корректность хранимой информации. Все проектные решения были успешно реализованы, а триггеры протестированы.

Для взаимодействия с базой данных было реализовано мобильное iOS-приложение.

Проведенные исследования производительности базы данных показали, что кеширование значительно улучшает время отклика по сравнению с прямыми запросами к базе данных. Также было установлено, что с увеличением объема данных и количества одновременных запросов время отклика системы увеличивается, что указывает на рост нагрузки на базу данных.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. IHRSA. The 2019 IHRSA Global Report. – Boston, MA, USA: Racquet and Sportsclub Associatio, 2019.
2. León-Quismondo J. Service Perceptions in Fitness Centers: IPA Approach by Gender and Age / León-Quismondo J., García-Unanue J., and Burillo P. // International Journal of Environmental Research and Public Health. 2020. Vol. 17, no. 8. p. 2844.
3. Bates M. Health Fitness Management: A Comprehensive Resource for Managing and Operating Programs and Facilities. 2nd edition. – Champaign, IL, USA: Human Kinetics, 2019.
4. Аврунев О. Е. Стасышин В. М. Модели баз данных. – Новосибирск: Новосибирский государственный технический университет, 2018.
5. Рейтинг систем управления базами данных (DB-Engines). Режим доступа: <https://db-engines.com/en/ranking> (дата обращения: 25.03.2025).
6. Codd E. F. A Relational Model of Data for Large Shared Data Banks // Communications of the ACM. 1970. Vol. 13, no. 6. P. 377–387.
7. PostgreSQL Global Development Group. PostgreSQL: The world's most advanced open source database. Режим доступа: <https://www.postgresql.org/> (дата обращения: 27.03.2025).
8. Sanfilippo Salvatore. Redis: A persistent key-value database. Режим доступа: <https://redis.io/> (дата обращения: 27.03.2025).
9. Apple Inc. Swift Programming Language. Режим доступа: <https://swift.org> (дата обращения: 27.03.2025).
10. Apple Inc. Xcode. Режим доступа: <https://developer.apple.com/xcode/> (дата обращения: 26.03.2025). 2025.
11. pgAdmin Development Team. pgAdmin 4. Режим доступа: <https://www.pgadmin.org/> (дата обращения: 26.03.2025).

12. Apple Inc. macOS 14 Sonoma. Режим доступа:  
<https://www.apple.com/macos/sonoma/> (дата обращения: 09.04.2025).
13. Документация процессора Intel Core i5-10210U [Электронный ресурс].  
Режим доступа: <https://ark.intel.com/content/www/us/en/ark/products/195436/intel-core-i510210u-processor-6m-cache-up-to-4-20-ghz.html> (Дата обращения: 17.10.2024).