Разработка базы данных для хранения и обработки данных фитнес-клуба

Выполнила:
студент 3 курса
группы ИУ7-64Б
Цховребова Яна Роландовна

Научный руководитель:
Гаврилова Ю.М.

Москва, 2025 г.

Цель и задачи

Цель работы: разработать базу данных для хранения и обработки данных фитнес-клуба

Задачи:

- Проанализировать предметную область и формализовать задачу
- Проанализировать модели данных и выбрать наиболее подходящую
- Спроектировать требуемую базу данных
- Спроектировать триггеры для автоматического обновления данных
- Выбрать средства реализации базы данных
- Реализовать спроектированную базу данных и обеспечить её функциональность согласно проектным решениям;
- Реализовать интерфейс доступа к базе данных
- Исследовать производительность базы данных при увеличения объема данных, количества одновременных запросов, а также с использованием кеширование и без него

Анализ предметной области

Фитнес-клуб — учреждение, предоставляющее услуги в области физической активности и здоровья, оснащённое тренажёрами и другим оборудованием

Основные функции:

— Проведение тренировок и предоставление возможности активного отдыха;

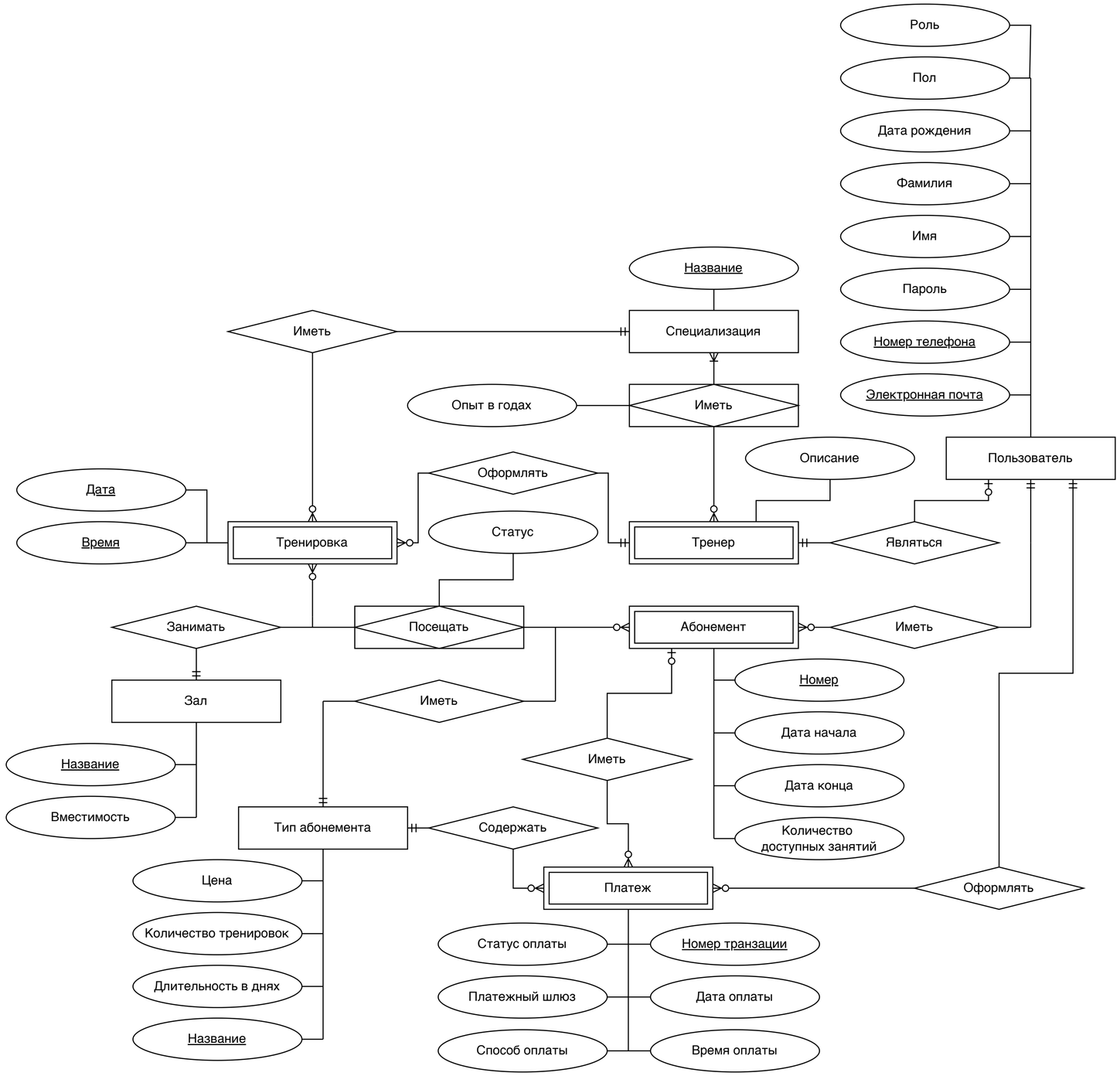
— Продажа абонементов с разными условиями доступа;

— Управление расписанием тренировок

— Учёт посещений и прогресса клиентов.

Анализ существующих решений

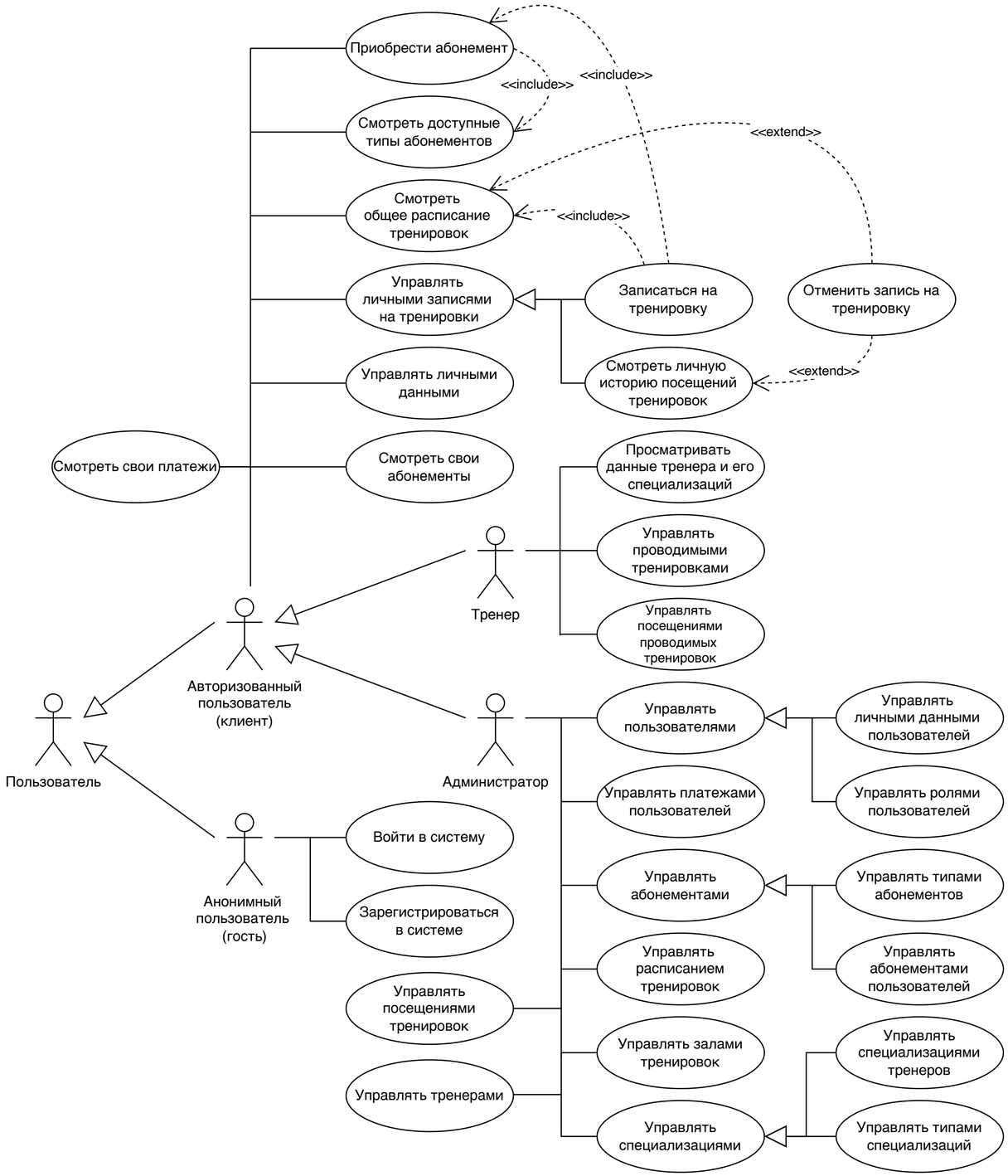
Функциональность для администратора	**1С:Фитнес Клуб**	**fitness365**
Управление расписанием, залами и персоналом	+	+
Работа с абонементами и услугами	+	+
Ведение клиентской базы и CRM	+	+
Функциональность для клиента	**1С:Фитнес Клуб**	**fitness365**
Наличие личного кабинета	+	+
Покупка абонементов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



Формализация данных

Ключевые сущности:

- Пользователь
- Тренер
- Специализация
- Тип абонемента
- Абонемент
- Платеж
- Зал для тренировок
- Тренировка
- Посещение



Формализация пользователей и их прав доступа

Пользователи:

Гость — доступ к регистрации и входу

Клиент — доступ к собственным данным: абонементы, тренировки, посещения

Тренер — права клиента + доступ к данным тренировок

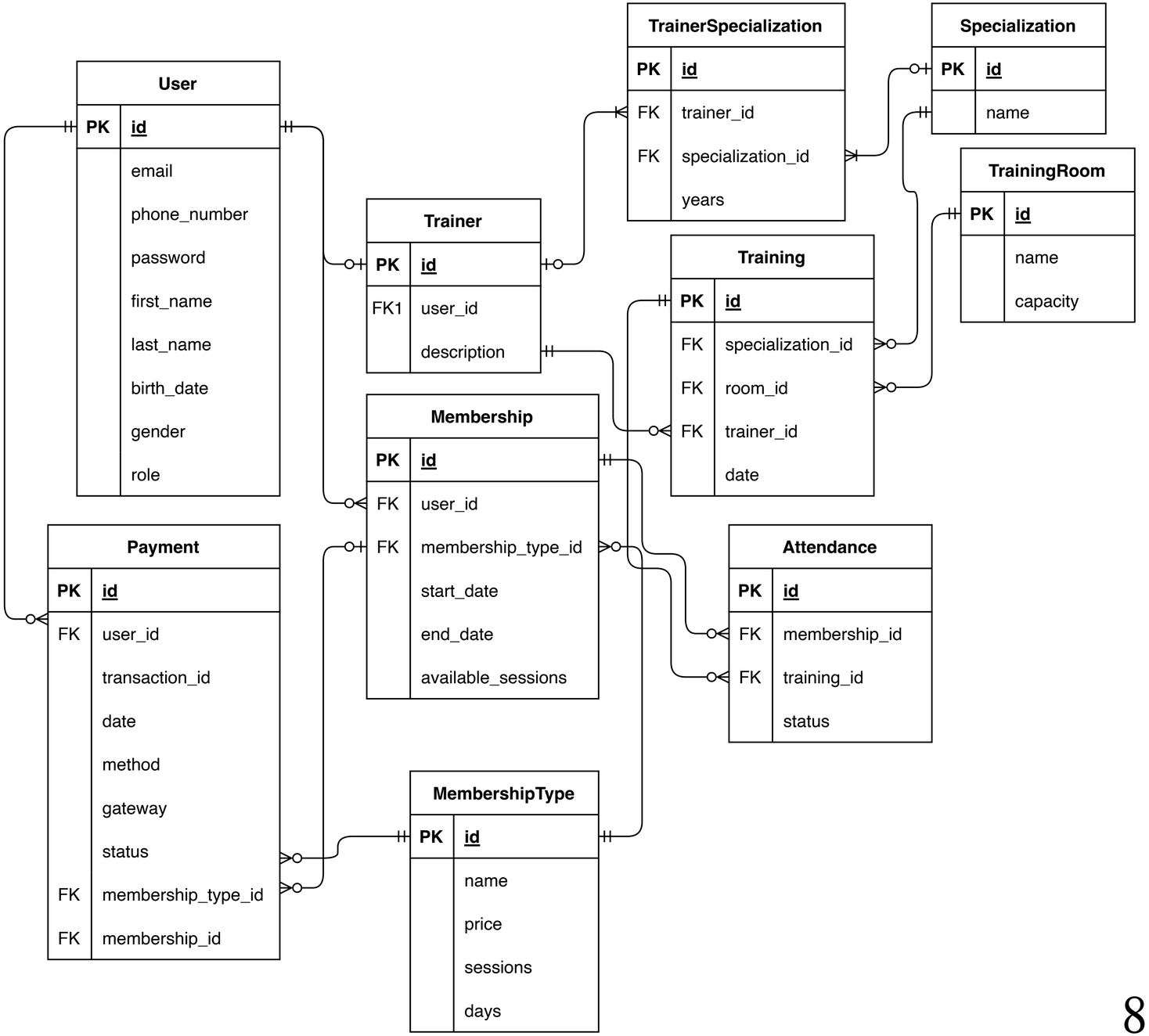
Администратор — полный доступ ко всем данным

Роли пользователей определяют уровень доступа к функциям и данным базы.

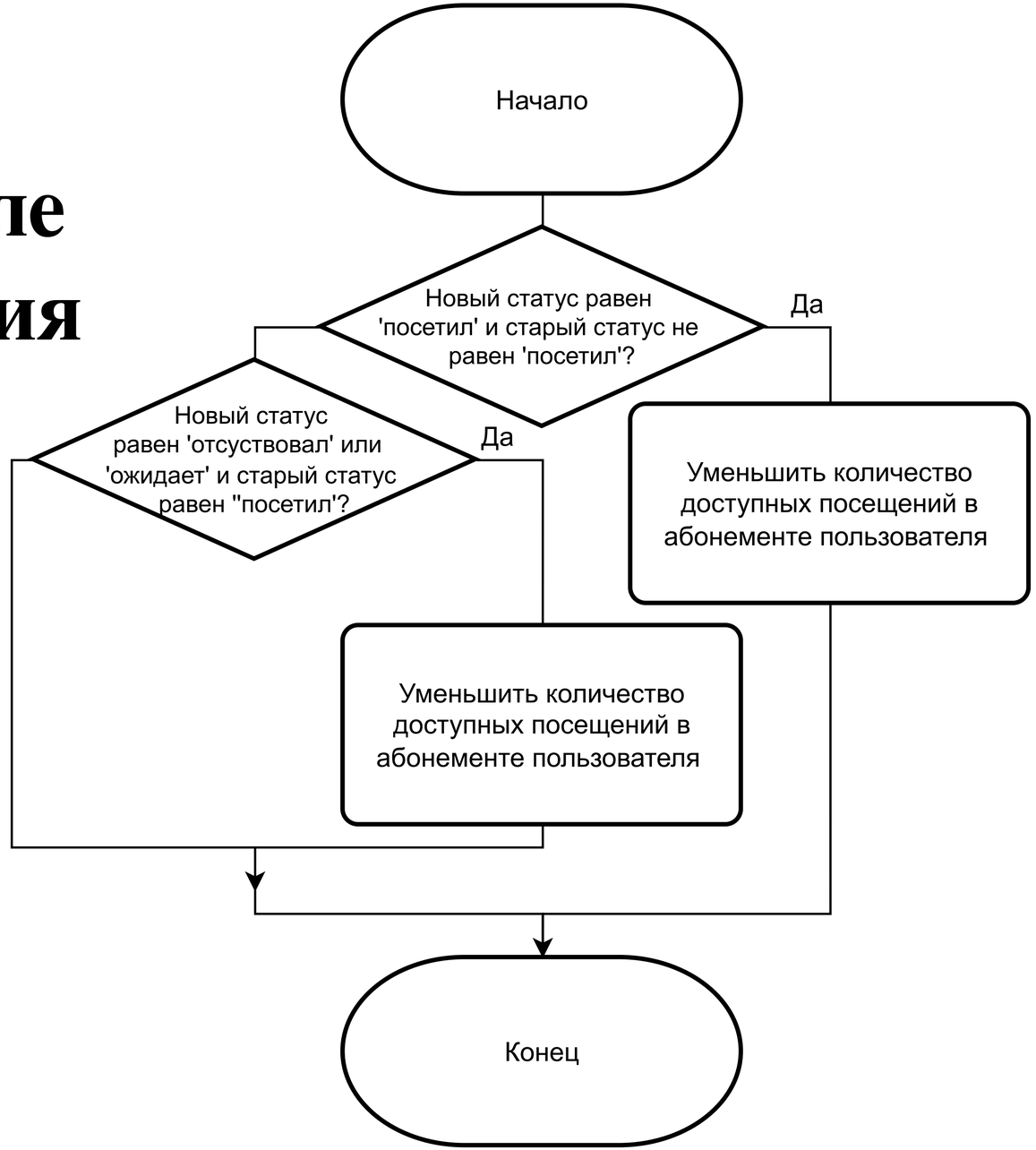
Модели данных

Реляционная модель данных	данные представлены в виде отношений (таблиц), каждая строка которых является кортежем, а столбцы — атрибутами с определёнными доменами; между отношениями устанавливаются связи с помощью ключей (первичных и внешних); схема данных строго фиксирована
Модель «ключ–значение»	данные представлены в виде пар «ключ — значение»; значения не имеют внутренней структуры; основные операции — получение и запись по ключу
Документная модель данных	данные хранятся в документах; которые могут содержать вложенные структуры и списки; нет фиксированной схемы; структура документа определяется динамически

Диаграмма «сущность-связь»



Алгоритм работы триггера для обновления данных после изменения статуса посещения тренировки



Алгоритм работы триггера для проверки данных перед добавлением записи на тренировку

Выбор СУБД

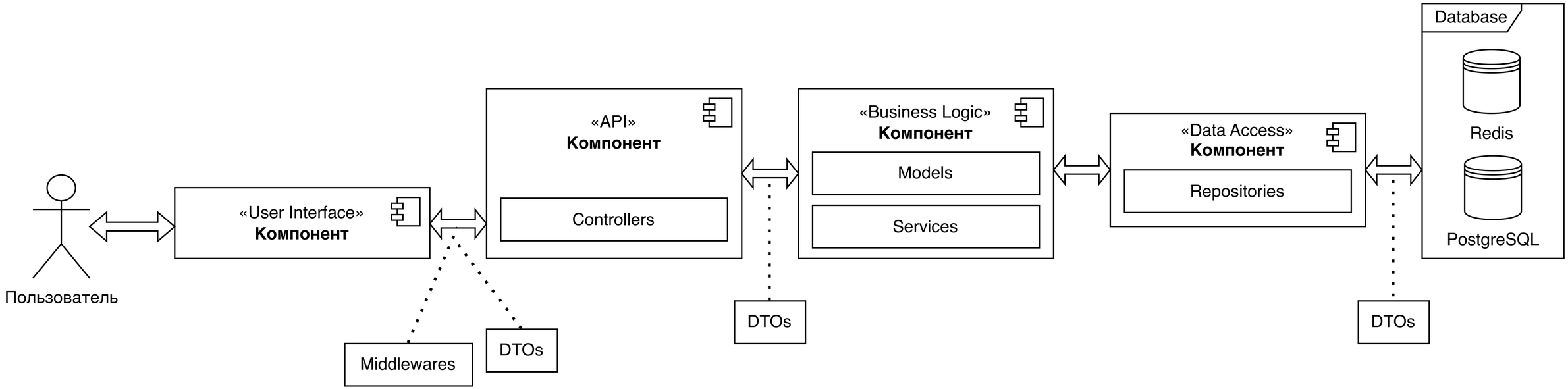
Критерий	PostgreSQL	MySQL	MS SQL	Oracle
Поддержка надежности и ACID-свойств	+	+/-	+	+
Совокупная стоимость владения	+	+	+/-	-

*СУБД - система управления базами данных

Технологический стек

СУБД	PostgreSQL
Кэширование данных	Redis
Язык программирования	Swift
Среды разработки	Xcode, pgAdmin 4

Структура ПО



*ПО - программное обеспечение

Технические характеристики

— Операционная система macOS 14.6.1

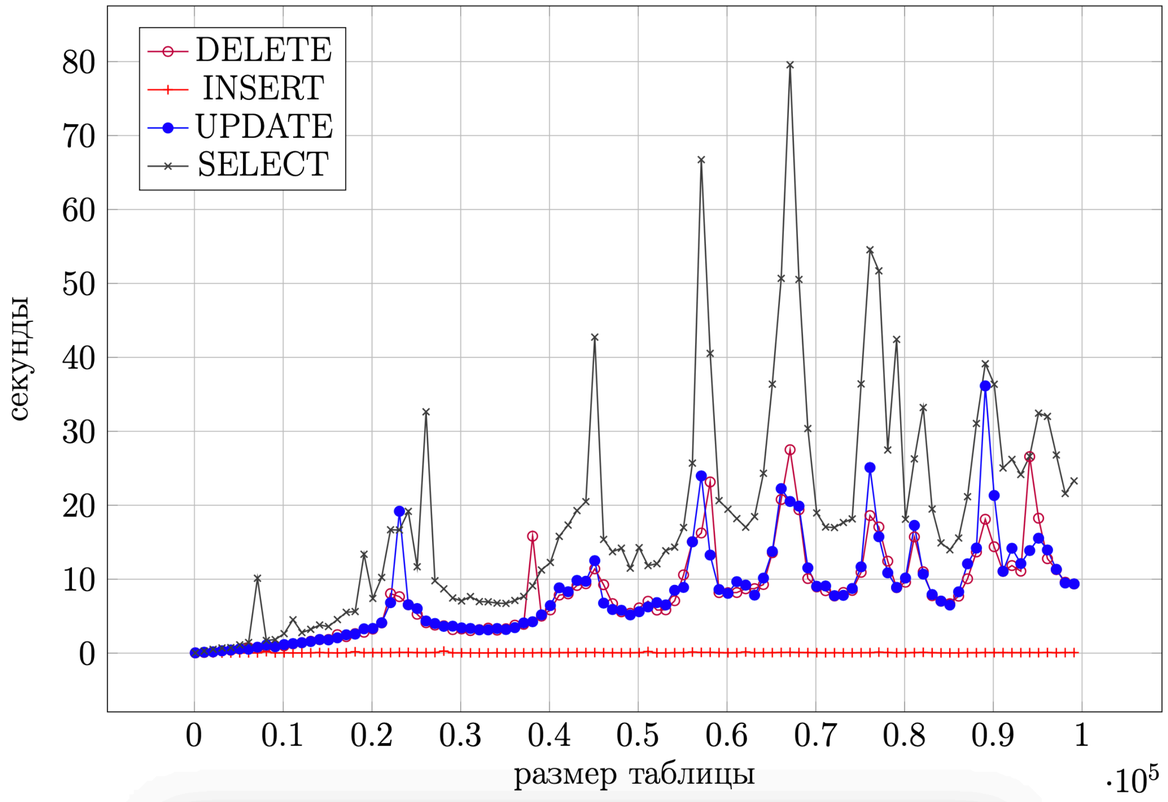
— Оперативная память (RAM) 16 ГБ

— Процессор (CPU) 2 ГГц 4-ядерный Intel Core i5 (8 логических ядер)

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

Зависимость времени выполнения операций от объема данных

- Среднее время выполнения запросов INSERT/SELECT/UPDATE/DELETE при объёмах таблицы от 100 до 99100 строк (шаг 1 000)
- Для каждой операции выполнено по 10 повторов




```
INSERT INTO "User"(
   id,
   email, phone_number, password,
   first_name, last_name,
   gender, birth_date, role
) VALUES (
   u_id,
   'test'||i||'@example.com',
   '+79001234567',
   'pass',
   'Имя', 'Фамилия', 'мужской',
   CURRENT_DATE - INTERVAL '30 years',
   'клиент'
);
```

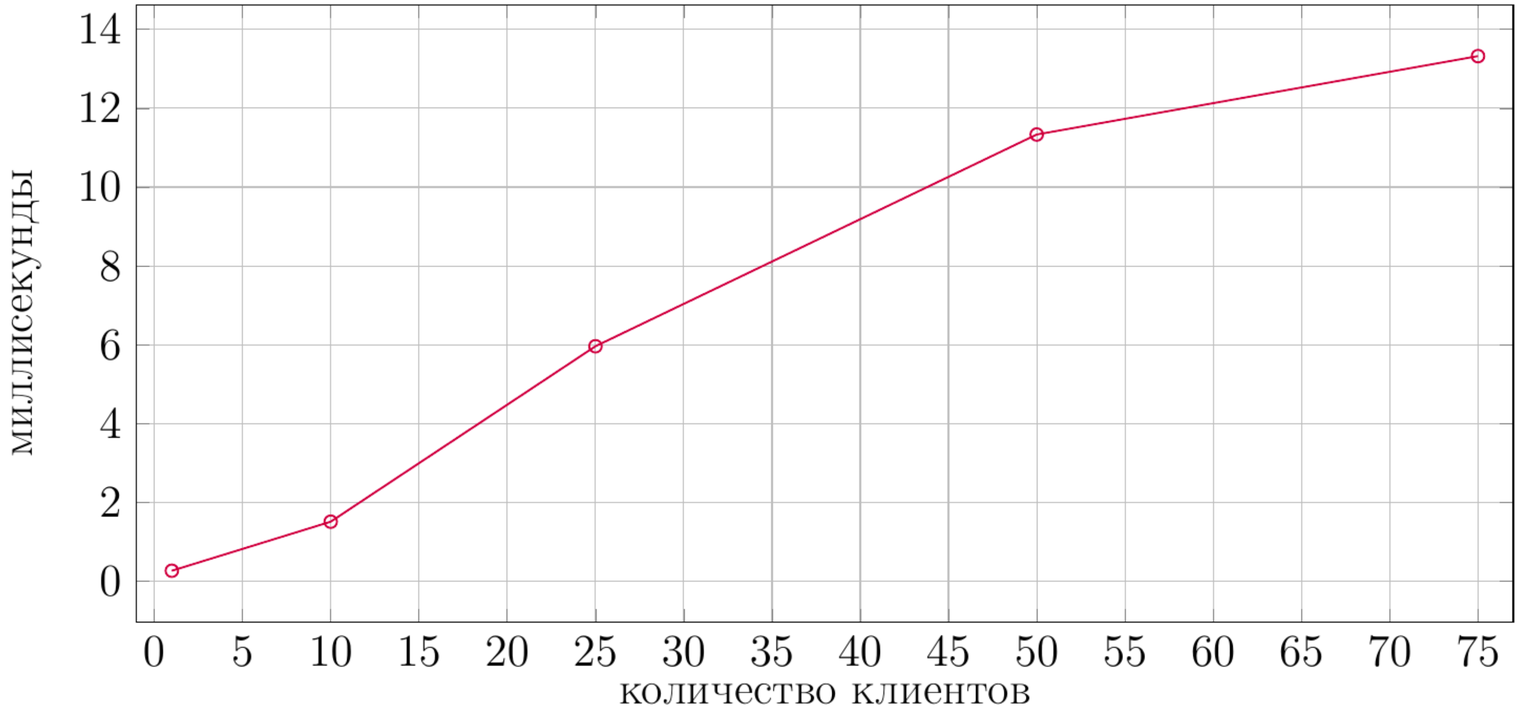
```
UPDATE "User"
SET first_name = 'Обновлено'
WHERE id = u_id ;
```

```
DELETE FROM "User" WHERE id = u_id;
```

```
SELECT id INTO u_id
FROM "User"
ORDER BY random()
LIMIT 1 ;
```

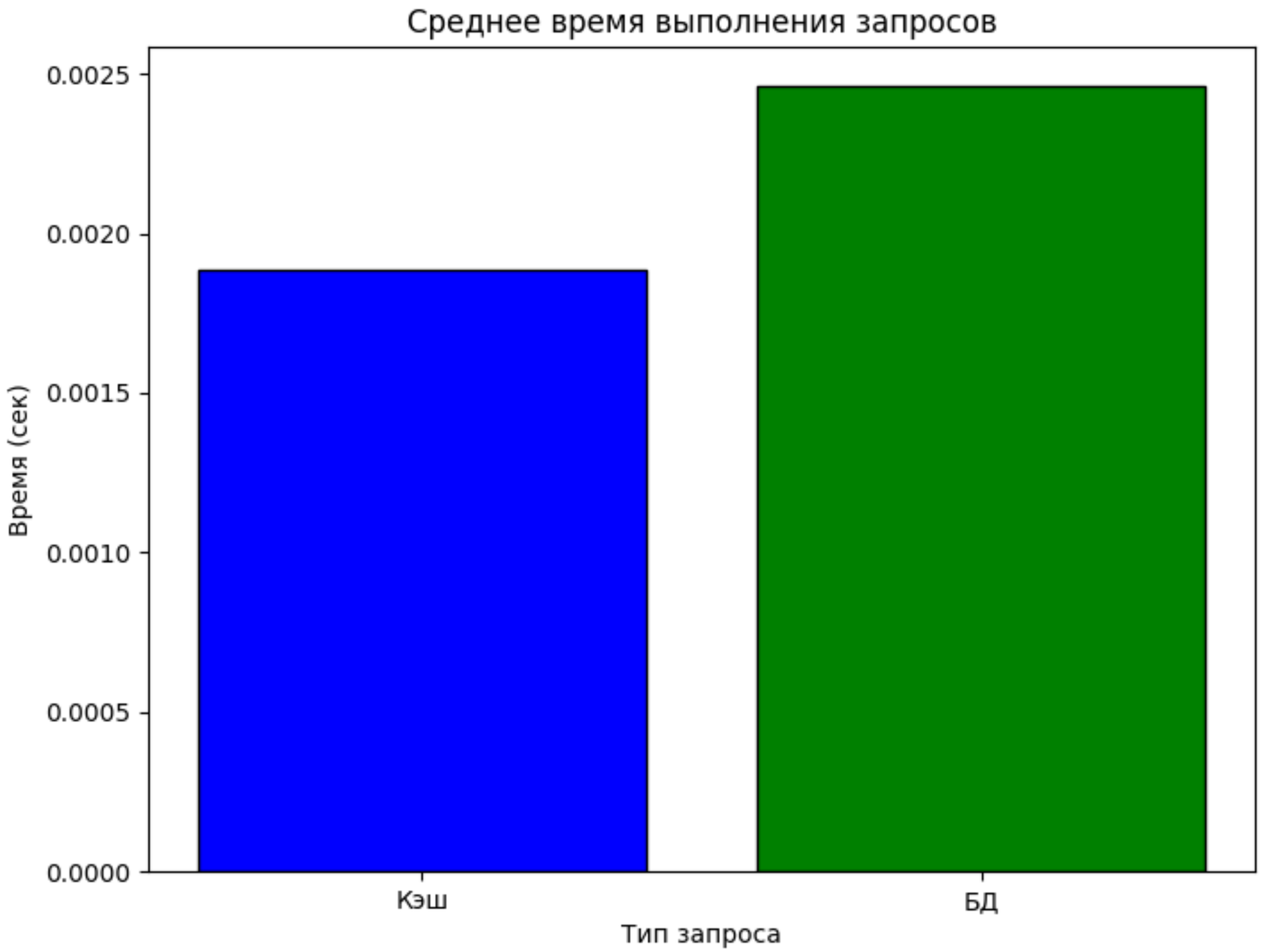
Зависимость времени отклика от количества пользователей базы данных

- Измерено время отклика при различном количестве клиентов
- Каждый замер длился 10 секунд
- Сравнение проводилось на одинаковых данных




```
SELECT *
FROM "User"
WHERE gender = 'мужской';
```

Сравнение времени выполнения запросов к кэшу и базе данных



- Оценка времени выборки данных из БД и кэша
- Выполнено 30 запросов с интервалом в 5 секунд
- Время жизни кэша в Redis — 10 секунд
- Сравнение проводилось на одинаковых данных

```
SELECT * FROM "User" u
JOIN "Membership" m
  ON m.user_id = u.id
JOIN "Order" o
  ON o.id = m.order_id
WHERE birth_date > '2003-01-01'
  AND gender = 'женский'
ORDER BY birth_date DESC;
```

Заключение

Цель курсовой работы успешно достигнута

Выполненные следующие задачи:

- Проведен анализ предметной области и определены требования к базе данных
- Спроектирована структура базы данных с ролями пользователей и выбран тип модели данных
- Реализованы сущности, связи, ограничения целостности и триггеры для автоматического обновления данных
- Создан интерфейс для взаимодействия с базой данных
- Проведено исследование производительности:
  - Время выполнения операций SELECT, UPDATE и DELETE увеличивается в большей степени относительно времени выполнения операции INSERT при увеличении объема данных
  - Рост числа одновременных запросов увеличивает время отклика системы
  - Кэширование с Redis снижает время обработки запросов по сравнению с прямыми запросами к PostgreSQL