СОДЕРЖАНИЕ

ВВЕДЕНИЕ

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

Для достижения этой цели необходимо выполнить следующие задачи:

1) проанализировать предметную область фитнес-клубов;
2) сформулировать требования и ограничения к разрабатываемой базе данных и веб-приложению фитнес-клуба;
3) спроектировать сущности базы данных и ограничения целостности фитнес-клуба;
4) спроектировать ролевую модель на уровне базы данных фитнес-клуба;
5) выбрать средства реализации базы данных и веб-приложения для демонстрации ее работы;
6) разработать сущности базы данных фитнес-клуба и реализованные ограничения целостности базы данных;
7) исследовать зависимость времени выполнения запроса поиска в базе данных от наличия индекса.

1 Аналитическая часть

1.1 Описание предметной области

Предметной областью является фитнес-индустрия. Типовой фитнес-клуб предлагает разные фитнес-активности, тренировки и услуги, направленные на улучшение физической формы, приведение в тонус тела и поддержание обего физического и психического благополучия.

Чаще всего в фитнес-клубе есть тренера, клиенты и администраторы.

1) **Клиент** — человек, который ходит в фитнес-центр для получения услуг.

2) **Тренер** — это сотрудник фитнес-центра, который проводит тренировки для клиентов.

3) **Администратор** — это сотрудник фитнес-центра, ответственный за управление расписанием фитнес-центра, а также решающий все вопросы клиентов и тренеров.

1.2 Анализ существующих решений

В эпоху цифровых технологий автоматизация процессов имеет огромное значение. Традиционные методы записи на тренировки через звонки или сообщения тренерам часто неудобны и не отвечают потребностям современных клиентов. Поэтому разработка решений, позволяющих клиентам записываться на тренировки онлайн, имеет большое значение. А так же одной из целей создания фитнес-клуба является мотивирование людей заниматься спортом, так как в России только 4% населения регулярно занимаются в фитнес-клубах, что в 3 раза меньше, чем в Европе [1].

На данный момент в России существует множество фитнес-клубов. Я рассмотрю самые популярные из них, а именно XFit [2], WeGym [3] и фитнес-клуб СССР [4].

Существующие решения анализировались по следующим критериям:

1) возможность просмотра расписания с фильтрацией;

2) наличие онлайн-записи на тренировки;

3) разнообразие групповых занятий (более 3 видов тренировок);

4) возможность просмотра информации о тренерах;

5) наличие бонусной системы.

Таблица 1.1 – Анализ существующих решений

	XFit	WeGym	CCCP
Расписание с фильтрацией	+	+	-
Разнообразие групповых занятий	+	+	-
Онлайн-запись на тренировки	-	-	-
Возможность просмотра информации о тренерах	+	+	+
Наличие бонусной системы	+	-	-

1.3 Формализация задачи

В ходе выполнения курсовой работы необходимо спроектировать и разработать базу данных для хранения и обработки данных фитнес-клуба, а так же веб-приложение для взаимодействия с ним. На основании анализа предметной области в фитнес-клубе могут быть выделены следующие роли.

1) Гость — это посетитель фитнес-центра, который еще не зарегистрировался на сайте или не вошел в свою учетную запись. Он не имеет доступа к личным данным, функциям бонусной программы, онлайн-записи на тренировки или другим привилегиям, доступным только зарегистрированным пользователям.

2) Клиент — это зарегистрированный пользователь фитнес-центра, который имеет доступ к своей учетной записи. Он может просматривать расписание занятий, записываться на тренировки, участвовать в бонусной программе, получать информацию о тренировках, услугах и специальных предложениях.

3) Тренер — это сотрудник фитнес-центра, который проводит тренировки для клиентов. Тренер может оформлять свои персональные тренировки в расписании и просматривать информацию о клиентах.

4) Администратор — это сотрудник фитнес-центра, ответственный за управление сайтом и контентом. Он имеет доступ к административным функциям, таким как создание и редактирование расписания тренировок, управление информацией о тренерах, настройка бонусной программы, управление акциями и специальными предложениями, а также учетом клиентов и их данных.

На рисунках 1.1–1.4 изображены диаграммы вариантов использования для разных типов пользователей.

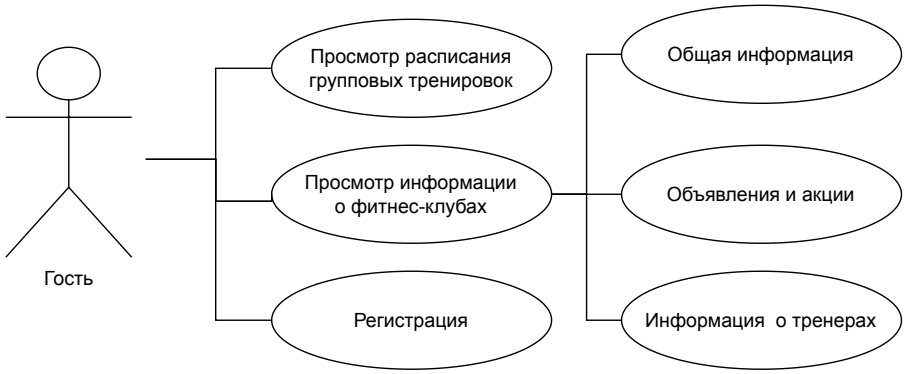

Рисунок 1.1 – Диаграмма вариантов использования для гостя

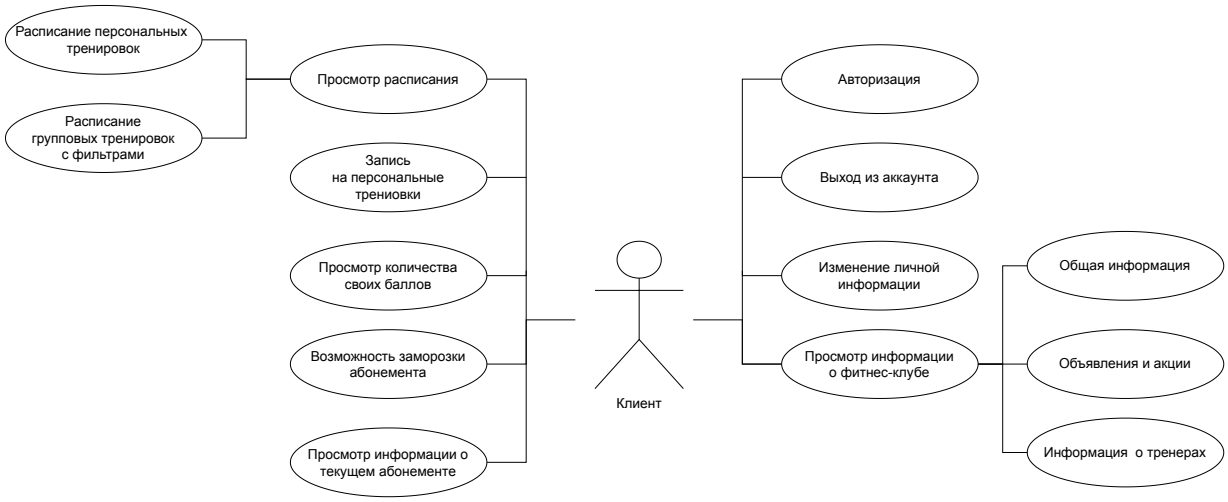

Рисунок 1.2 – Диаграмма вариантов использования диаграмма для клиента

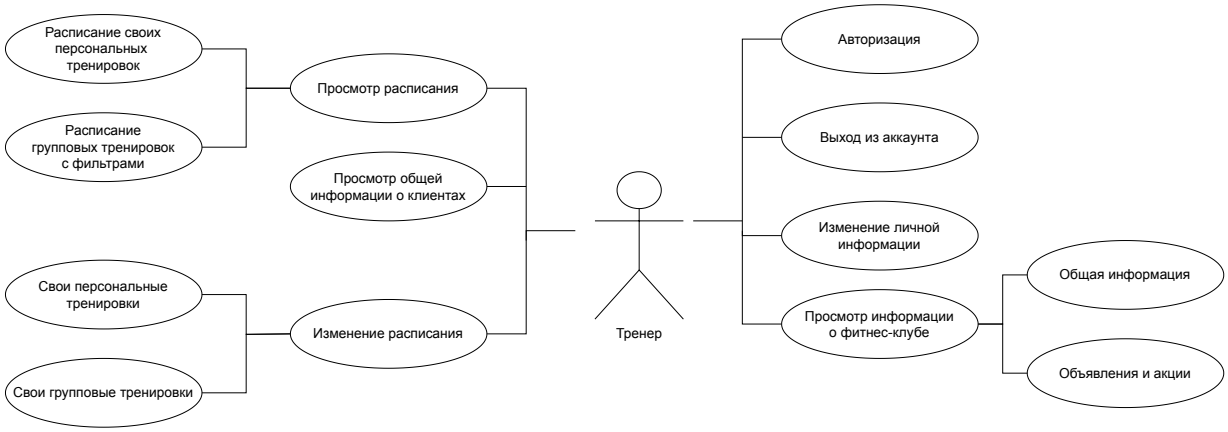

Рисунок 1.3 – Диаграмма вариантов использования диаграмма для тренера

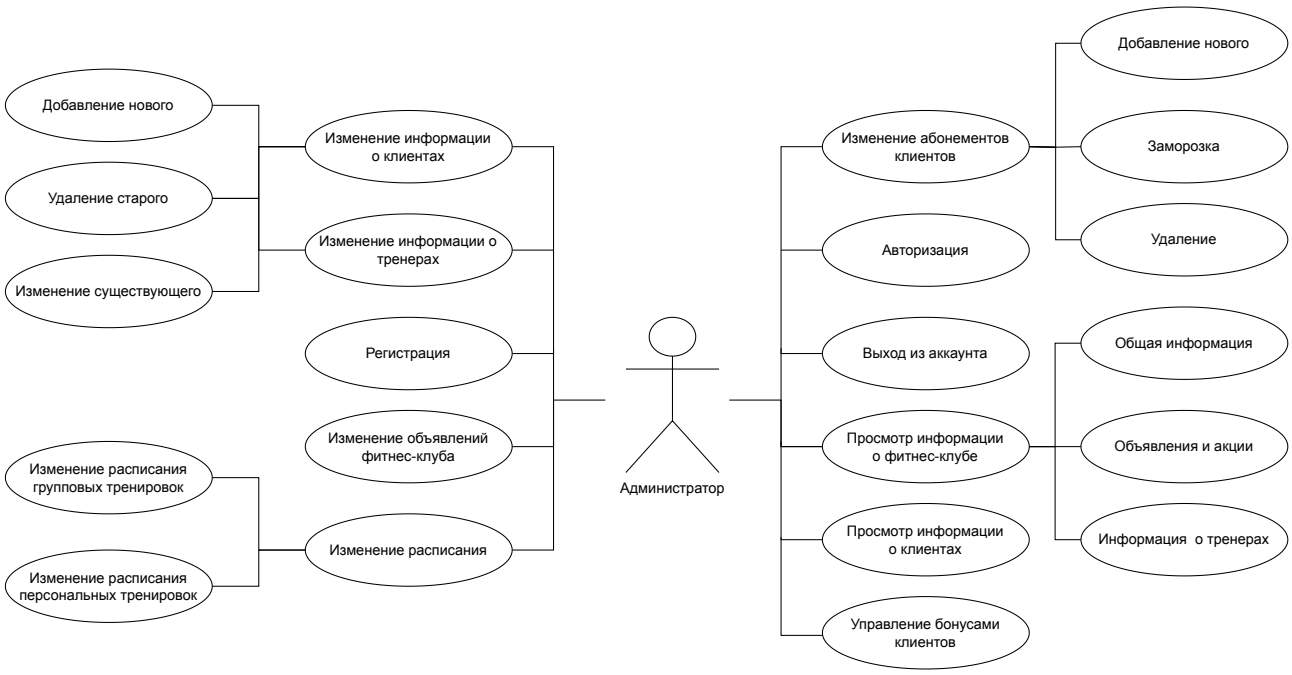

Рисунок 1.4 – Диаграмма вариантов использования диаграмма для администратора

Для создания базы данных фитнес-центра, были выделены следующие сущности:

1) пользователи;

2) администраторы;

3) клиенты;

4) тренера;

5) расписание;

6) абонементы;

7) бонусы;

8) тренировки;

9) услуги и товары.

ER-диаграмма сущностей в нотации Чена представлена на рисунке 1.5.

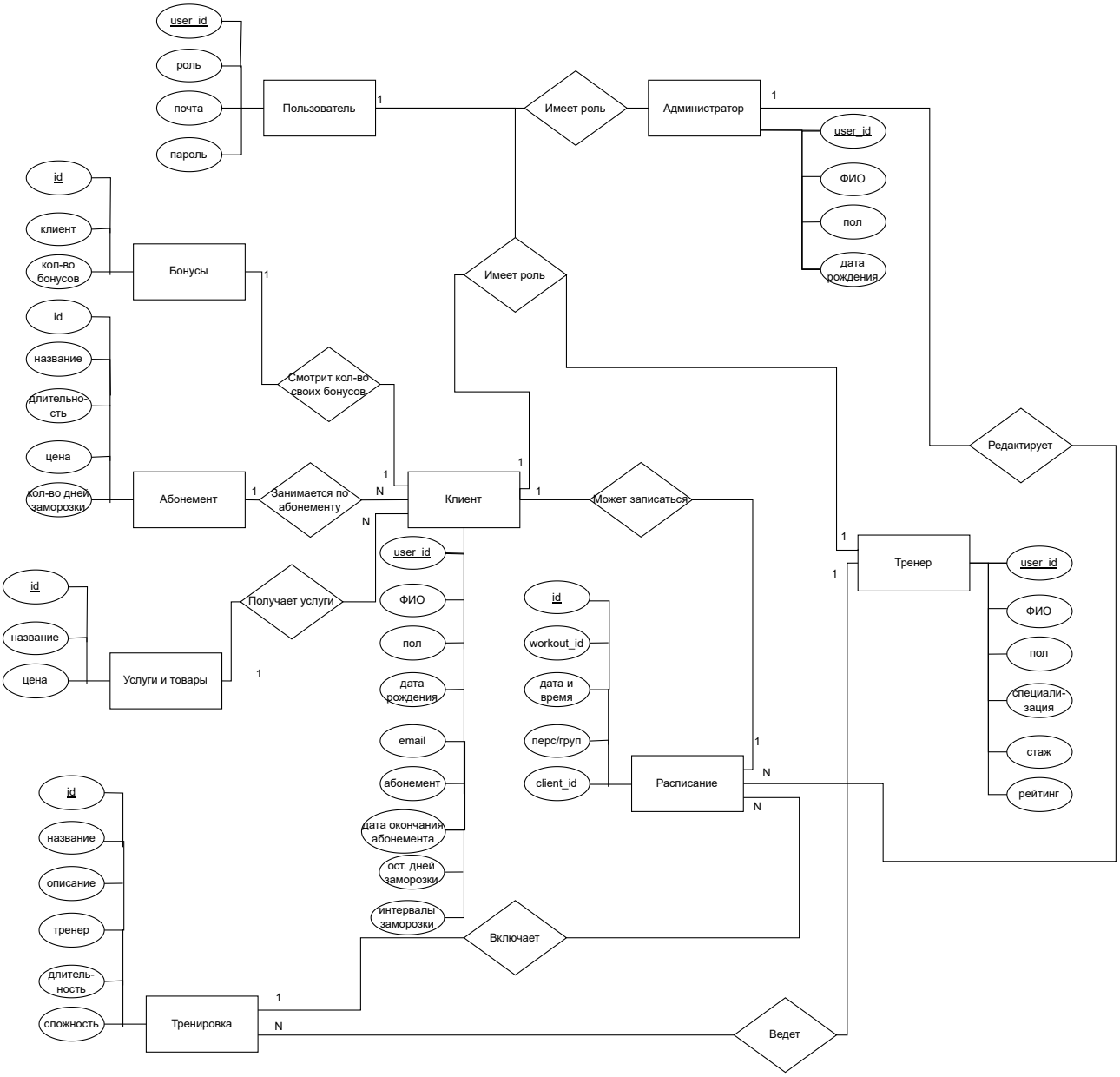

Рисунок 1.5 – ER-диаграмма сущностей в нотации Чена

1.4 Виды баз данных

База данных представляет собой совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязей в рассматриваемой предметной области. Она играет важную роль в хранении, управлении, анализе и обработке данных, что в конечном итоге облегчает принятие решений и повышает эффективность работы организаций и систем [5].

По типу модели данных базы данных разделяются на следующие [6]:

1) дореляционные;

2) реляционные;

3) постреляционные.

1.4.1 Дореляционные базы данных

Дореляционные базы данных — это тип баз данных, который находится за пределами реляционной модели. Они предоставляют альтернативные способы организации и хранения данных, не используя таблицы, столбцы и отношения, как это делается в реляционных базах данных.

Дореляционные базы данных можно разделить на 3 категории [7]:

1) с инвертированными списками;

2) иерархические;

3) сетевые.

В дореляционных базах данных на основе инвертированных списков каждый уникальный атрибут собирается в отдельный список, содержащий ссылки на элементы данных с соответствующим значением. Это позволяет гибко и эффективно осуществлять поиск, фильтрацию и полнотекстовый поиск данных. В иерархических базах данных данные огранизованы в иерархическую структуру подобно древовидной структуре. В такой базе данных каждый элемент данных имеет связь с одним родительским элементом и может иметь несколько дочерних элементов. В сетевых база данных информация организована в виде графа. В таких базах данных сущности представлены в виде вершин, а связи между сущностями — в виде ребер графа [8].

1.4.2 Реляционные базы данных

Реляционные базы данных основаны на реляционной модели, которая представляет данные в виде таблиц с записями и атрибутами. Каждая запись

имеет уникальный идентификатор (ключ), а каждый атрибут содержит значение для каждой записи [9].

Реляционные базы данных отделяют логические структуры данных, такие как таблицы, представления и индексы, от физических структур хранения данных. Это означает, что администраторы баз данных могут управлять физическим хранилищем данных независимо от доступа к данным [10].

Четыре важные свойства реляционной базы данных определяют ACID [10].

1) Атомарность: все операции в базе данных либо выполняются полностью, либо не выполняются вообще.

2) Согласованность: после завершения транзакции данные остаются в цельном, согласованном состоянии.

3) Изоляция: эффекты одной транзакции невидимы для других, пока она не завершится, чтобы избежать несогласованности данных.

4) Долговечность: реляционная база данных гарантирует, что изменения данных, подтвержденные транзакцией, сохранятся постоянно.

1.4.3 Постреляционные базы данных

Постреляционные базы данных представляют собой различные модели данных, которые расширяют и дополняют возможности классической реляционной модели [11]. Некоторые из основных типов постреляционных моделей представлены далее [6].

1) Объектно-реляционные базы данных: они комбинируют функциональность реляционных баз данных с возможностями объектно-ориентированного программирования, что позволяет хранить и обрабатывать сложные объекты со связями и наследованием, предоставляя расширенную модель данных.

2) Объектно-ориентированные базы данных: они предназначены для хранения объектов и связей между ними без использования традиционных таблиц и отношений. Это позволяет более натурально моделировать и

обрабатывать данные, основываясь на концепциях классов, объектов и наследования.

3) Многомерные базы данных: они предназначены для хранения и обработки многомерных данных, таких как данные OLAP (Online Analytical Processing). Многомерные базы данных оптимизированы для аналитических задач, где данные организованы по нескольким измерениям и иерархиям, что облегчает анализ и отчетность.

4) Прочие (NoSQL) базы данных: это семейство баз данных, которые не следуют традиционной реляционной модели и предлагают альтернативные подходы к хранению и обработке данных.

Одним из преимуществ постреляционной модели является возможность объединения связанных реляционных таблиц в одну постреляционную таблицу. Однако, недостатком такой модели является сложность обеспечения целостности и согласованности данных, что может представлять проблемы при работе с ней [11].

Выводы к аналитической части

В этой части работы был проведен анализ существующих решений и формализованна задача. Также выбрана реляционная модель данных, так как она подходит для решения поставленной задачи.

Таблица 1.2 – Сравнение баз данных

	Дореляционные БД	Реляционные БД	Постреляционные БД
Организация данных	Иерархическая или сетевая структура	Таблицы, столбцы, отношения	Модели данных, такие как объекты, документы, графы и др.
Поддерживаемые типы данных	Ограниченные	Все основные, которые может включать база данных	Расширенные, включая мультимедийные
Поддержка транзакционности	нет	есть	есть
Опыт работы исполнителя	нет	есть	нет

2 Конструкторская часть

2.1 Описание сущностей базы данных и их связей

Согласно ER-диаграмме сущностей в нотации Чена в проектируемой базе данных были выделены 9 таблиц для сущностей, которые уже определены в аналитической части, и 1 дополнительная таблица для реализации связи «многие ко многим» между клиентами и товарами. На рисунке 2.1 изображена диаграмма проектируемой базы данных.

Рассмотрим подробнее каждую таблицу и ее поля.

Таблица пользователей, содержащая информацию о пользователях фитнес-клуба, содержит поля, описанные в таблице 2.1.

Таблица 2.1 – Поля таблицы пользователей

Поле	Тип	Описание
id_user	целое	Идентификатор пользователя
role	строка	Роль пользователя (клиент, тренер, администратор)
email	строка	email пользователя
password	строка	Пароль пользователя

Таблица тренеров, содержащая информацию о тренерах фитнес-клуба, содержит поля, описанные в таблице 2.2.

Таблица 2.2 – Поля таблицы тренеров

Поле	Тип	Описание
id_trainer	целое	Идентификатор тренера
id_user	целое	Идентификатор пользователя
name	строка	ФИО тренера
gender	строка	Пол тренера
specialization	строка	Специализация тренера
experience	целое	Стаж тренера
rating	целое	Рейтинг тренера

Таблица администраторов, содержащая информацию об администраторах фитнес-клуба, содержит поля, описанные в таблице 2.3.

Таблица 2.3 – Поля таблицы администраторов

Поле	Тип	Описание
id_admin	целое	Идентификатор администратора
id_user	целое	Идентификатор пользователя
name	строка	ФИО администратора
data_of_birth	дата	Дата рождения администратора
gender	строка	Пол администратора

Таблица клиентов, содержащая информацию о клиентах фитнес-клуба, содержит поля, описанные в таблице 2.4.

Таблица 2.4 – Поля таблицы клиентов

Поле	Тип	Описание
id_client	целое	Идентификатор клиента
id_user	целое	Идентификатор пользователя
name	строка	ФИО клиента
gender	строка	Пол клиента
data_of_birth	дата	Дата рождения клиента
id_membership	целое	Идентификатор абонемента
membership_end	дата	Дата окончания абонемента

Таблица бонусов, содержащая информацию о бонусах клиентов фитнес-клуба, содержит поля, описанные в таблице 2.5.

Таблица 2.5 – Поля таблицы бонусов

Поле	Тип	Описание
id_bonus	целое	Идентификатор записи о бонусах
id_client	целое	Идентификатор клиента
count	целое	Количество бонусов

Таблица абонементов, содержащая информацию об абонементах фитнес-клуба, содержит поля, описанные в таблице 2.6.

Таблица 2.6 – Поля таблицы абонементов

Поле	Тип	Описание
id_membership	целое	Идентификатор абонемента
name	строка	Название абонемента
duration	промежуток времени	Длительность абонемента
price	целое	Цена абонемента
freezing	целое	Длительность возможной заморозки абонемента

Таблица тренировок, содержащая информацию о тренировках, которые проводятся в фитнес-клубе, содержит поля, описанные в таблице 2.7.

Таблица 2.7 – Поля таблицы тренировок

Поле	Тип	Описание
id_workout	целое	Идентификатор тренировки
name	строка	Название тренировки
description	строка	Описание тренировки
id_trainer	целое	Идентификатор тренера
duration	промежуток времени	Длительность тренировки
level	целое	Сложность тренировки (от 1 до 5)

Таблица расписания, содержащая информацию о расписании фитнес- клуба, содержит поля, описанные в таблице 2.8.

Таблица 2.8 – Поля таблицы расписания

Поле	Тип	Описание
id_record	целое	Идентификатор записи
id_workout	целое	Идентификатор тренировки
data_and_time	дата и время	Дата и время тренировки
id_client	целое	Идентификатор клиента (NULL, если групповая)

Таблица товаров, содержащая информацию о услугах и товарах фитнес-клуба, содержит поля, описанные в таблице 2.9.

Таблица 2.9 – Поля таблицы товаров

Поле	Тип	Описание
id_good	целое	Идентификатор товара/услуги
name	строка	Название товара/услуги
price	целое	Цена товара/услуги

Таблица, связывающая клиентов и товары, которые они приобрели, содержит поля, описанные в таблице 2.10.

Таблица 2.10 – Поля таблицы, связывающей клиентов и приобретенные ими товары

Поле	Тип	Описание
id	целое	Идентификатор записи о покупке
id_client	целое	Идентификатор клиента
id_good	целое	Идентификатор товара/услуги

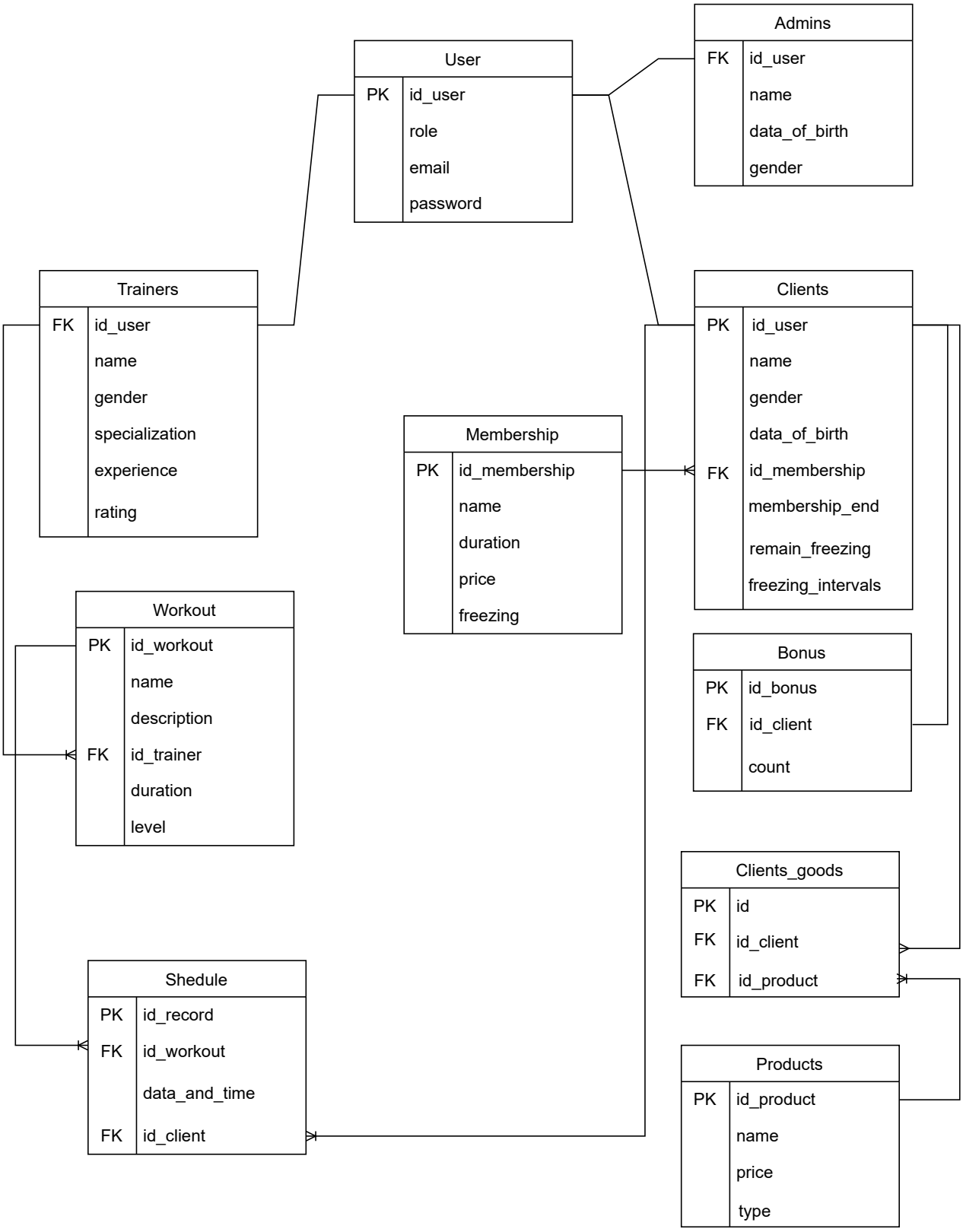

Рисунок 2.1 – Диаграмма проектируемой базы данных

2.2 Роли базы данных

На основе пунктов работы 1.1, 1.3 и 2.1 были определены следующие роли и их права доступа в проектируемой базы данных.

1) Гость: возможность просмотра данных таблиц Membership, Goods, Workout, Shedule и Trainers.

2) Клиент: возможность просмотра данных таблиц Membership, Goods, Workout, Shedule и Trainers, своей строки в таблице Bonus, а также добавлять записи в таблицу Shedule (с помощью записи на персональную тренировку).

3) Тренер: возможность просмотра данных таблиц Membership, Goods, Workout, Shedule Trainers и Clients, а также добавлять записи в таблицу Shedule (с помощью добавления персональных и групповых тренировок) и в таблицу Workout (с помощью создания новых тренировок).

4) Администратор: возможность просматривать и изменять все таблицы.

2.3 Триггер базы данных

При желании клиента купить товар важно правильно рассчитать количество бонусов, которое спишется и начислится, а так же нужно, чтобы при каждой покупке товара они сразу начислялись, иначе может произойти ситуация, что товар будет куплен, а бонусы не списаны.

Поэтому при каждой покупке товара должна вызываться функция, схема которой изображена на рисунке 2.2.

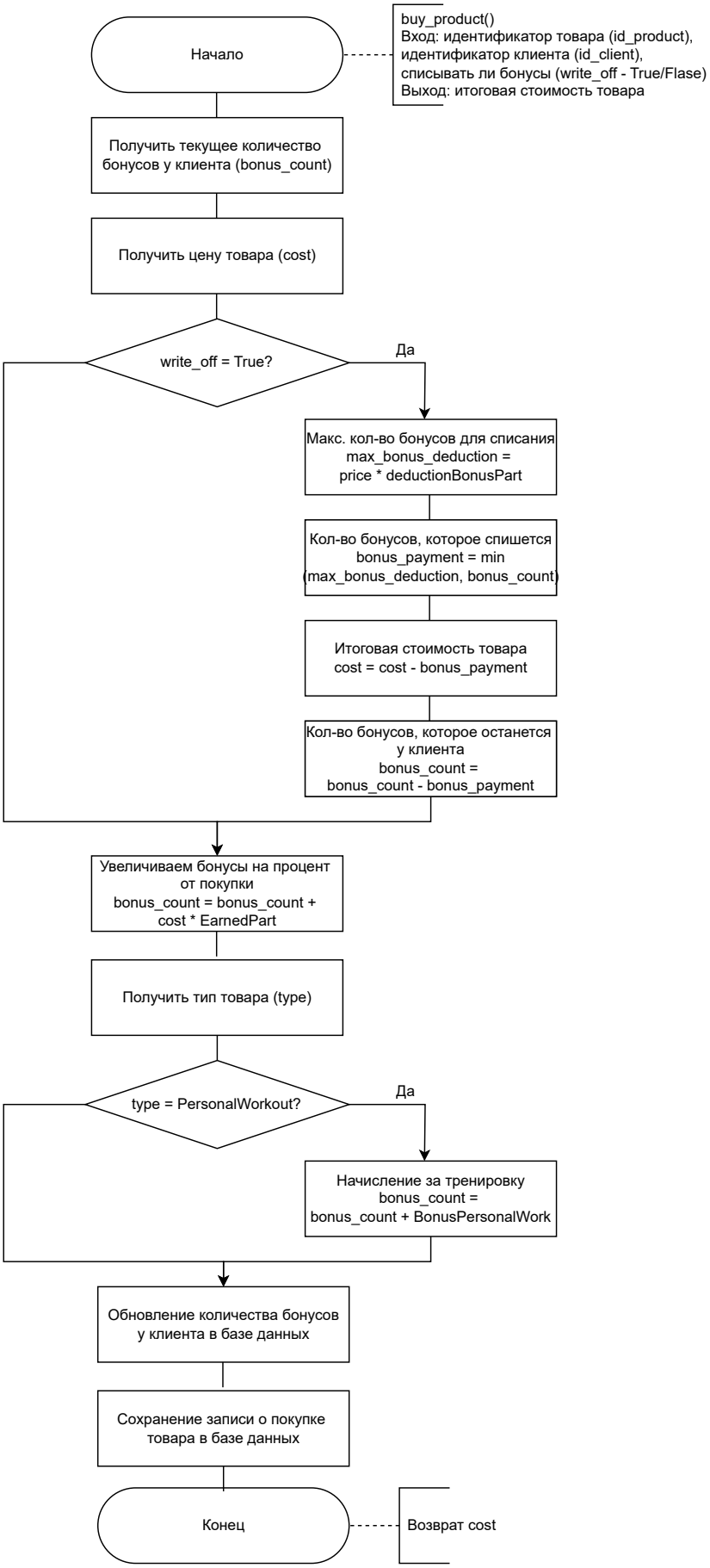

Рисунок 2.2 – Схема алгоритма вычисления итоговой стоимости и обновлении бонусов

Выводы к конструкторской части

В этой части работы были описаны сущности проектируемой базы данных, а также ее роли и триггер.

3 Технологическая часть

3.1 Средства реализации

Для решения поставленной задачи была выбрана СУБД $PostrgeSQL$, так как она поддерживает такие функции, как связность таблиц, транзакционность, поддержка сложных структур, таких как *json*, а так же имеется опыт работы с данной системой управления базами данных [12].

В качестве языка программирования для серверной части был выбран язык *C#* [13], так как в стандартной библиотеке языка присутствует поддержка всех структур данных, выбранных по результатам проектирования.

В качестве среды разработки была выбрана среда $Rider$, так как у него есть встроенный отладчик, а так же можно применять изменения в режиме реального времени.

Для реализации взаимодействия с базой данный используется фреймворк *Entity Framework* [14], который обеспечивает сопоставление отношений объектов. Так же поддерживает он запросы $LINQ$, отслеживание изменений, обновления и миграции схемы.

3.2 Создание таблиц

В части работы 2.1 были описаны таблицы базы данных, соответствующий код для создания этих таблиц приведен в листинге 3.1–3.3.

Листинг 3.1 – Код на SQL для создания таблицы Users

```
1 CREATE TABLE IF NOT EXISTS "Users" (
2     id UUID PRIMARY KEY,
3     role INT NOT NULL,
4     email VARCHAR(100) NOT NULL,
5     password VARCHAR(100) NOT NULL
6 );
```

Листинг 3.2 – Код на SQL для создания таблиц Trainers, Admins, Workouts, Memberships, Schedules

```sql
CREATE TABLE IF NOT EXISTS "Trainers" (
    id UUID PRIMARY KEY NOT NULL REFERENCES "Users" (id),
    name VARCHAR(50) NOT NULL,
    gender VARCHAR(10) NOT NULL CHECK (gender IN ('male',
        'female')),
    specialization VARCHAR(50) NOT NULL,
    experience INT NOT NULL,
    rating INT NOT NULL CHECK (rating >= 1 AND rating <= 5)
);

CREATE TABLE IF NOT EXISTS "Admins" (
    id UUID NOT NULL REFERENCES "Users" (id),
    name VARCHAR(50) NOT NULL,
    date_of_birth DATE NOT NULL,
    gender VARCHAR(10) NOT NULL CHECK (gender IN ('male', 'female'))
);

CREATE TABLE IF NOT EXISTS "Workouts" (
    id UUID PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    description VARCHAR(200) NOT NULL,
    trainer_id UUID NOT NULL REFERENCES "Trainers" (id),
    duration INTERVAL NOT NULL,
    level INT NOT NULL CHECK (level >= 1 AND level <= 5)
);
CREATE TABLE IF NOT EXISTS "Memberships" (
    id UUID PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    duration INTERVAL NOT NULL,
    price INT NOT NULL,
    freezing INT NOT NULL
);

CREATE TABLE IF NOT EXISTS "Schedules" (
    id UUID PRIMARY KEY,
    workout_id UUID NOT NULL REFERENCES "Workouts" (id),
    date_and_time TIMESTAMP NOT NULL,
    client_id UUID REFERENCES "Clients" (id)
);
```

Листинг 3.3 – Код на SQL для создания таблиц Clients, Products, ClientProducts, Bonuses

```
CREATE TABLE IF NOT EXISTS "Clients" (
    id UUID PRIMARY KEY NOT NULL REFERENCES "Users" (id),
    name VARCHAR(50) NOT NULL,
    gender VARCHAR(10) NOT NULL CHECK (gender IN ('male',
        'female')),
    date_of_birth DATE NOT NULL,
    membership_id UUID NOT NULL REFERENCES "Memberships" (id),
    membership_end DATE NOT NULL,
    remain_freezing INT,
    freezing_intervals json
);

CREATE TABLE IF NOT EXISTS "Products" (
    id UUID PRIMARY KEY,
    type VARCHAR(50) NOT NULL,
    name VARCHAR(50) NOT NULL,
    price INT NOT NULL
);

CREATE TABLE IF NOT EXISTS "ClientProducts" (
    id UUID PRIMARY KEY,
    id_client UUID NOT NULL REFERENCES "Clients" (id),
    id_product UUID NOT NULL REFERENCES "Products" (id)
);

CREATE TABLE IF NOT EXISTS "Bonuses" (
    id UUID PRIMARY KEY,
    client_id UUID NOT NULL REFERENCES "Clients" (id),
    count INT NOT NULL
);
```

3.3 Создание ролей на уровне базы данных

По спроектированным в части работы 2.2 ролям базы данных, были созданы роли на уровне базы данных, соответствующих код представлен в листинге 3.4.

Листинг 3.4 – Код на SQL для создания ролей

```
1  CREATE ROLE guest;
2  GRANT SELECT ON TABLE public."Products", public."Memberships",
       public."Schedules", public."Trainers", public."Workouts" TO
       guest;
3
4  CREATE ROLE client;
5  GRANT SELECT ON TABLE public."Bonuses", public."Products",
       public."Memberships", public."Schedules", public."Trainers",
       public."Workouts" TO client;
6  GRANT INSERT ON TABLE public."Schedules" TO client;
7  GRANT UPDATE ON TABLE public."Clients" TO client;
8
9  CREATE ROLE trainer;
10 GRANT SELECT ON TABLE public."Schedules", public."Clients",
       public."Workouts" TO trainer;
11 GRANT INSERT, UPDATE, DELETE ON TABLE public."Schedules",
       public."Workouts" TO trainer;
12 GRANT UPDATE ON TABLE public."Trainers" TO trainer;
13
14 CREATE ROLE admin;
15 GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin;
```

3.4 Создание триггера

Код для создания триггера и соответствующей функции, алгоритм которой был в части работы 2.3, представлен с листинге 3.5.

Листинг 3.5 – Код на SQL для создания триггера и функции

```
CREATE OR REPLACE FUNCTION update_bonus_count()
RETURNS TRIGGER AS $$
DECLARE
    product RECORD;
    total_cost NUMERIC;
    available_bonus_count INTEGER;
    bonus_payment INTEGER;
    max_bonus_count INTEGER;
    product_type VARCHAR(100);
BEGIN
    SELECT price INTO total_cost FROM public."Products" WHERE id =
       NEW.id_product;
    SELECT count INTO available_bonus_count
    FROM public."Bonuses"
    WHERE client_id = NEW.id_client;
    IF NEW.write_off THEN
    SELECT (total_cost * 0.5)
    INTO max_bonus_count;
    bonus_payment := LEAST(available_bonus_count, max_bonus_count);
    total_cost := total_cost - bonus_payment;
    available_bonus_count := available_bonus_count - bonus_payment;
    END IF;
    available_bonus_count := available_bonus_count + (total_cost *
       0.1);
    SELECT type INTO product_type FROM public."Products" WHERE id =
       NEW.id_product;
    IF product_type = 'PersonalWorkout' THEN
    available_bonus_count := available_bonus_count + 100;
    END IF;

    UPDATE public."Bonuses" SET count = available_bonus_count WHERE
       client_id = NEW.id_client;

    RETURN NULL
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER on_client_product_insert
AFTER INSERT ON public."ClientProducts"
FOR EACH ROW
EXECUTE FUNCTION update_bonus_count();
```

3.5 Тестирование

Проводилось тестирование функции, алгоритм которой приведен в части работы 2.3.

Были составлены следующие классы эквивалентности:

1) у клиента нет бонусов;
2) у клиента бонусов менее половины стоимости товара;
3) спроектировать сущности базы данных и ограничения целостности фитнес-клуба;
4) количество бонусов клиента равно половине стоимости товара;
5) у клиента бонусов более половины стоимости товара;
6) клиент покупает персональную тренировку.

Каждый тестовый случай анализировался со включенным списанием бонусов и без него.

3.6 Примеры веб-интерфейса программы

На рисунках 3.1 и 3.2 представлен интерфейс, который видит пользователь при входе на сайт.

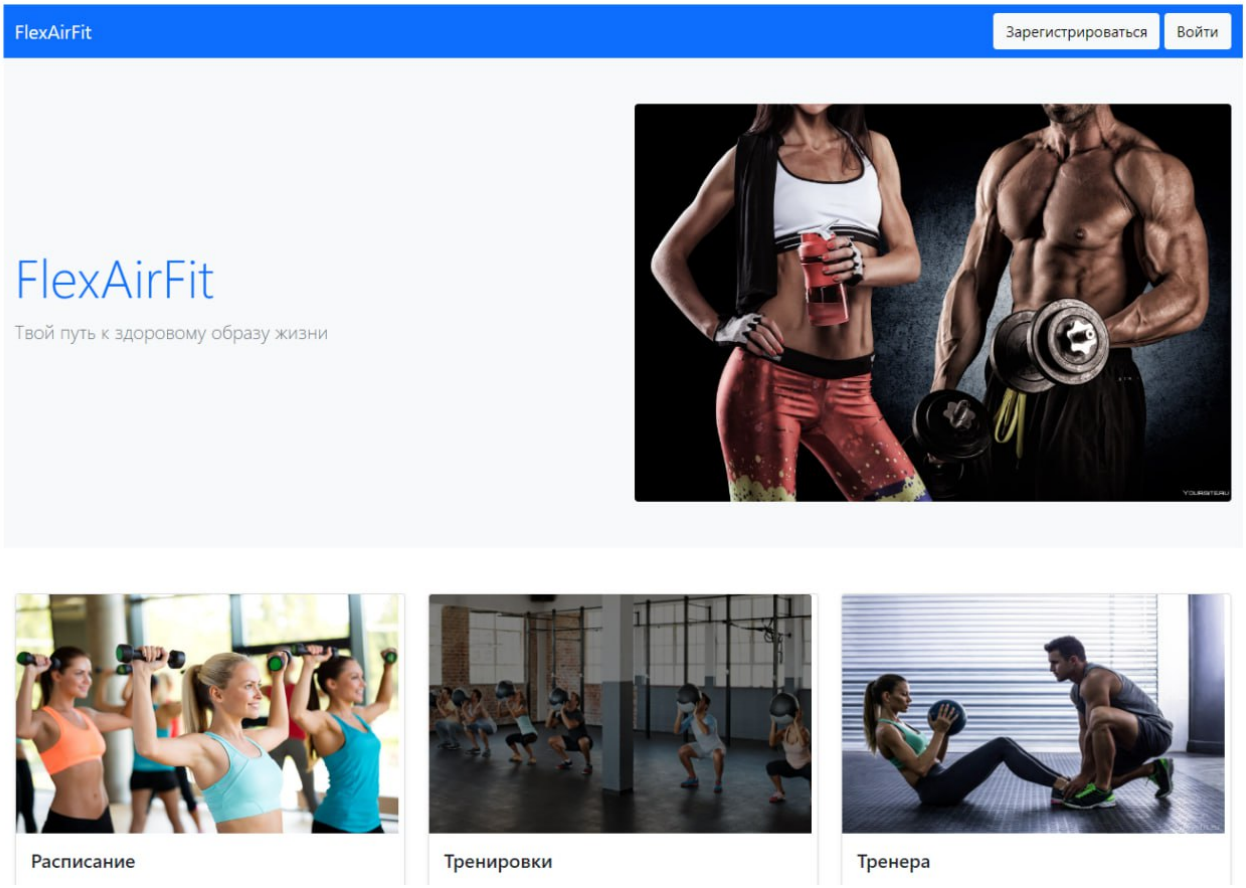

Рисунок 3.1 – Пример интерфейса

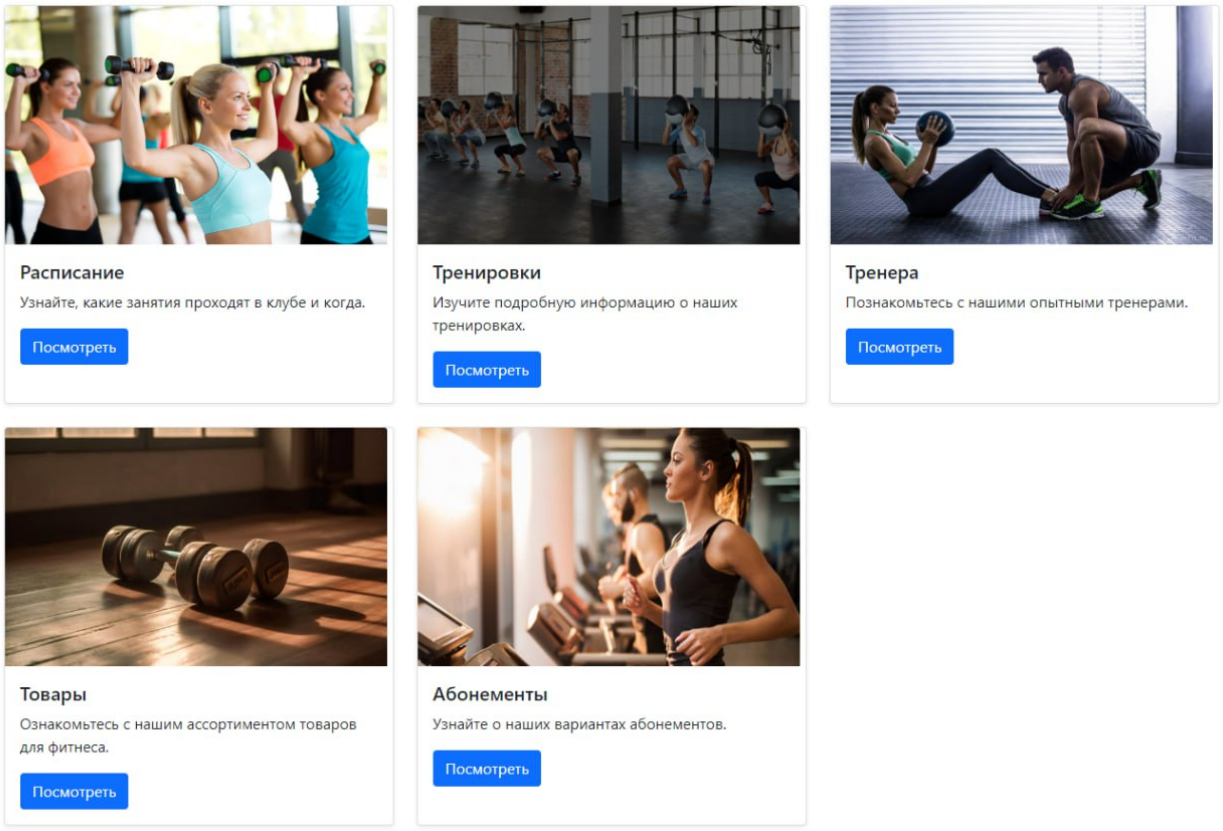

Рисунок 3.2 – Пример интерфейса

3.7 Выводы к технологической части

В данной части были рассмотрены средства реализации, разработано программное обеспечение, рассмотрен интерфейс и тестирование.

4 Исследовательская часть

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось исследование представлены далее:

1) операционная система Ubuntu 22.04.3 [15];

2) процессор *Intel Core i5-1135G7* 2.4 ГГц;

3) оперативная память 16 ГБ, *DDR4*.

Во время исследования времени ноутбук был нагружен только встроенными приложениями окружения, не был включен в сеть питания.

4.2 Описание исследования

Целью экперимента является исследование зависимости времени выполнения запроса к таблице базы данных при разном количестве записей в ней. А так же необходимо провести сравнительный анализ при наличии индекса в этой таблице и без него.

Для исследования использовалась таблица *Schedules*, так как чаще всего и клиентам, и тренерам требуется посмотреть расписание тренировок.

Индекс создавался на поле *date_and_time* с помощью следующей команды

```
1 CREATE INDEX IF NOT EXISTS IX_Schedules_DateAndTime ON
    public."Schedules" (date_and_time)
```

Замеры проводились для разного количества записей в таблице: от 10 до 5000. Так же при каждом количестве время замерялось по 100 раз и после этого усреднялось.

4.3 Результаты исследования

Результаты замеров приведены в таблице 4.1 (время в мс.).

Таблица 4.1 – Результаты замеров времени

Количество записей	Время без индекса	Время с индексом
10	0.001683	0.001860
100	0.010538	0.011872
250	0.030085	0.025258
500	0.070169	0.049925
1000	0.199291	0.112767
1500	0.361543	0.150151
2000	0.565231	0.203894
2500	0.743195	0.258923
3000	0.988928	0.332711
3500	1.582918	0.400369
4000	1.937912	0.453513
5000	2.363422	0.576650

Также на рисунке 4.1 приведены графические результаты замеров времени выполнения запроса.

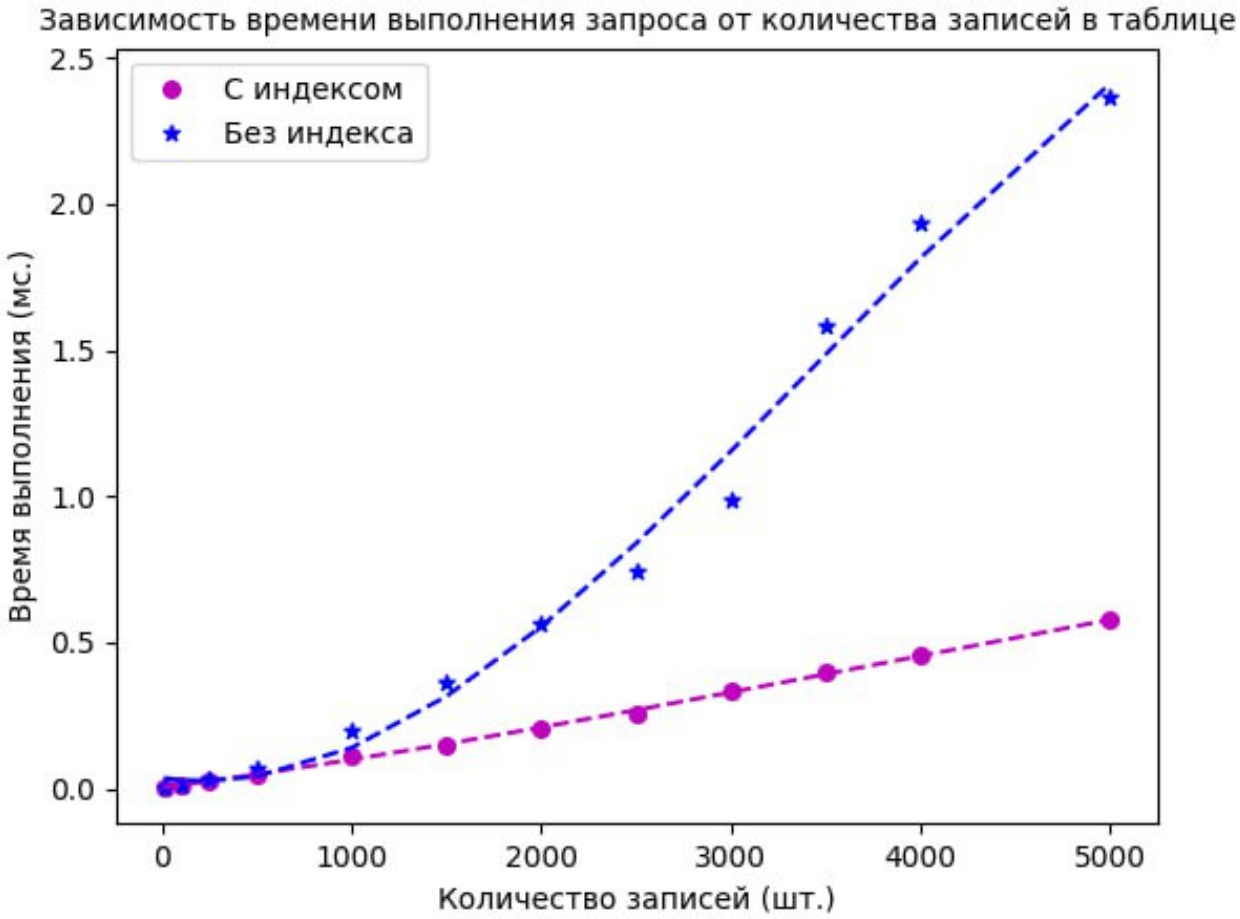

Рисунок 4.1 – Сравнение по времени выполнения запроса от количества записей в таблице с использованием индекса и без него

Выводы к исследовательской части

В данной части был проведен сравнительный анализ времени выполнения запроса к таблице базы данных для получения записей из расписания в конкретный день. Из проведенного эксперимента можно сделать вывод, что функция зависимости времени без использования индекса растет быстрее, чем с ним.

При количестве записей менее 250 время выполнения запроса без индекса меньше, чем с индексом, но при большем количестве запрос без использования индекса выполняется дольше. Причем чем больше количество записей, тем сильнее растет разница во времени. Так при количестве записей в таблице, равном 500, отношение времен между рассматриваемыми вариантами составляет 1.4 раза, а при количестве записей, равном 5000, уже 4.1 раза.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были выполнены следующие задачи:

1) проанализирована предметная область фитнес-клубов;
2) сформулированы требования и ограничения к разрабатываемой базе данных и веб-приложению фитнес-клуба;
3) спроектированы сущности базы данных и ограничения целостности фитнес-клуба;
4) спроектирована ролевая модель на уровне базы данных фитнес-клуба;
5) выбраны средства реализации базы данных и веб-приложения для демонстрации ее работы;
6) разработаны сущности базы данных фитнес-клуба и реализованные ограничения целостности базы данных;
7) исследована зависимость времени выполнения запроса поиска в базе данных от наличия индекса.

Все задачи выполнены, таким образом, цель работы достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Только 4% населения России регулярно занимаются в фитнес-клубах. 2023. URL: https://www.forbes.ru/sport/500472-tol-ko-4-naselenia-rossii-regularno-zanimautsa-v-fitnes-klubah.

2. Фитнес-клуб XFit [Электронный ресурс]. Режим доступа: `https://www.xfit.ru/` (дата обращения: 10.03.2024).

3. Фитнес-клуб WeGym [Электронный ресурс]. Режим доступа: `https://wegym.ru/` (дата обращения: 10.03.2024).

4. Фитнес-клуб СССР [Электронный ресурс]. Режим доступа: `https://fitness-cccp.ru/` (дата обращения: 10.03.2024).

5. Т.Е.Сергеева М.Ю.Сергеев. Базы данных: модели данных, проектирование, язык SQL. 2012.

6. Б.Г. Трусов И.В. Рудаков Ю.И. Терентьев С.С. Комалов С.В. Горин В.А. Крищенко. учебник по курсу «Информатика». 2011.

7. К.Дж.Дейт. Введение в базы данных. 2005.

8. Ю.М. Гаврилова. Конспект лекций по дисциплине «Базы данных». 2023.

9. Васильева К. Н. Хусаинова Г. Я. Реляционные базы данных. 2020.

10. What is a Relational Database (RDBMS)? [Электронный ресурс]. Режим доступа: `https://www.oracle.com/database/what-is-a-relational-database/` (дата обращения: 17.03.2024).

11. Постреляционная модель: понятие, достоинства и недостатки [Электронный ресурс]. Режим доступа: `https://studfile.net/preview/5407090/page:4/` (дата обращения: 17.03.2024).

12. PostrgeSQL [Электронный ресурс]. Режим доступа: https://www.postgresql.org/ (дата обращения: 21.05.2024).

13. C# [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 21.05.2024).

14. Entity Framework [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/ef/ (дата обращения: 21.05.2024).

15. Ubuntu 22.04.3 LTS (Jammy Jellyfish) [Электронный ресурс]. Режим доступа: https://releases.ubuntu.com/22.04/ (дата обращения: 01.02.2024).

ПРИЛОЖЕНИЕ А

Презентация к курсовой работе.