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

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Теоретическая информатика и компьютерные технологии

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ**

**ПО КУРСУ БАЗЫ ДАННЫХ**

**НА ТЕМУ:**

Цифровизация тренировок в спортзале

Студент

*подпись, дата фамилия, и.о.*

Научный руководитель

*подпись, дата фамилия, и.о.*

*2022*

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc114671047)

[1 Проблематика 5](#_Toc114671048)

[1.1 Клиентская боль 5](#_Toc114671049)

[1.2 Проблема спортзала 6](#_Toc114671050)

[2 Путь к решению 7](#_Toc114671051)

[3 Проектирование 9](#_Toc114671052)

[3.1 Концептуальная архитектура 9](#_Toc114671053)

[3.2 Построение модели данных 9](#_Toc114671054)

[3.2.1 Сущности в реляционной модели 11](#_Toc114671055)

[3.2.1.1 Customer 11](#_Toc114671056)

[3.2.1.2 Weighing 12](#_Toc114671057)

[3.2.1.3 Training 13](#_Toc114671058)

[3.2.1.4 Strength\_Exercise 14](#_Toc114671059)

[3.2.1.5 Strength\_Exercise\_Type 16](#_Toc114671060)

[3.2.1.6 Approach 16](#_Toc114671061)

[3.2.1.7 Running\_Exercise 18](#_Toc114671062)

[3.2.1.8 Running\_Exercise\_Type 20](#_Toc114671063)

[3.2.1.9 Trainer 21](#_Toc114671064)

[3.2.1.10 Trainer\_Type 21](#_Toc114671065)

[3.2.2 Правила обеспечения связей между отношениями 23](#_Toc114671066)

[3.2.2.1 Customer-Weighing 23](#_Toc114671067)

[3.2.2.2 Customer-Training 23](#_Toc114671068)

[3.2.2.3 Training-Strength\_Exercise 24](#_Toc114671069)

[3.2.2.4 Training-Running\_Exercise 25](#_Toc114671070)

[3.2.2.5 Running\_Exercise\_Type -Running\_Exercise 26](#_Toc114671071)

[3.2.2.6 Strength\_Exercise\_Type-Strength\_Exercise 26](#_Toc114671072)

[3.2.2.7 Strength\_Exercise-Approach 27](#_Toc114671073)

[3.2.2.8 Trainer-Strength\_Exercise 28](#_Toc114671074)

[3.2.2.9 Trainer-Running\_Exercise 29](#_Toc114671075)

[3.2.2.10 Trainer\_Type-Trainer 30](#_Toc114671076)

[4 Реализация 31](#_Toc114671077)

[4.1 Реализация базы данных 31](#_Toc114671078)

[4.2 Написание веб-приложения 31](#_Toc114671079)

[4.3 Пользовательский интерфейс 38](#_Toc114671080)

[4.3.1 Страница регистрации 38](#_Toc114671081)

[4.3.2 Страница входа 38](#_Toc114671082)

[4.3.3 Страница тренировок клиента 39](#_Toc114671083)

[4.3.4 Страница упражнений в тренировке 40](#_Toc114671084)

[4.3.5 Страница подходов в силовой тренировке 41](#_Toc114671085)

[5 Тестирование 43](#_Toc114671086)

[ЗАКЛЮЧЕНИЕ 46](#_Toc114671087)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc114671088)

[ПРИЛОЖЕНИЕ А 48](#_Toc114671089)

ВВЕДЕНИЕ

Спорт – любимый многими досуг и способ поддержания себя в форме. По статистике более 60% [1] население России с той или иной периодичностью занимаются спортом. Одно из наиболее популярных мест для поддержания своего физического здоровья – спортзал, он доступен во все времена года и в шаговой доступности практически в любом городе России.

Современный мир предоставляет возможность автоматизировать контроль за многими бизнес-процессами с помощью информационных технологий – мобильных приложений, web-приложений, баз данных, которые осуществляют хранение информации.

Целью данной курсовой работы является цифровизация нахождения клиента в спортзале, что непременно принесёт выгоду как ему, так и бизнесу.

# Проблематика

## Клиентская боль

Частая проблема, с которой сталкиваются клиенты спортзала, является сложность в фиксации результатов тренировки (сколько человек пробежал на беговой дорожке, за какое время, сколько раз сделал приседания со штангой и т. д.). Отсутствие документирования приводит к следующим проблемам:

1. человек забывает, что он делал на предыдущей тренировке, на каких весах остановился и сколько подходов в упражнении он делал. Из-за этого в следующий раз он может выбрать неправильный вес и в итоге потренироваться не так эффективно, как бы он мог.
2. Отсутствие возможности отследить прогресс своих тренировок, понять, например, что он за несколько месяцев совершенно не улучшил свои показатели, а значит его подход к занятиям нуждается в пересмотре.
3. Невозможность проанализировать свои тренировки и их качество, например количество отдыха между подходами является важным параметром, так как только короткий отдых между тренировками вызывает более выраженный подъём гормона роста и тестостерона, а значит было бы полезно его отслеживать.

Для решения этой проблемы в настоящее время люди используют следующие способы:

1. Запись тренировок в тетрадь. Такой способ имеет ряд очевидных недостатков в виде возможности потерять записи, а также необходимости постоянно носить с собой ручку и тетрадь;
2. Использование специальных приложений для записи тренировок, например, Nike Training и Adidas Running. Проблемой в этом случае является необходимость вручную вводить информацию об упражнениях, а также постоянно иметь при себе телефон (большое количество людей предпочитают во время занятий уходить в “оффлайн”).

## Проблема спортзала

В настоящее время большинство спортзалов фактически являются “стоянкой” для тренажёров с тренерами, куда пускают клиентов, которые заплатили деньги. Говоря другими словами, фитнес центры не имеют практически никакой информации о своих клиентах, они не знают, какие упражнения те делают, в какое время привыкли посещать спортзал, сколько времени в нём проводят.

# Путь к решению

Путь к решению проблемы отсутствия у спортзала информации о пользователе – использование технологии Gym Training Tracking, которая позволяет пользователю получать максимально просто (с минимальным количество действий, которые от него требуются) подробную информацию о своей тренировке в спортзале. Данная технология предполагает следующее:

1. Введение спортзалом в эксплуатацию “умных” тренажёров, способных полностью фиксировать упражнения, которые на них выполняют, а также передавать эту информацию в приложение пользователя;
2. Создание “пропуска” для пользователя в виде ключ карты или браслета, способные взаимодействовать со всем “умным” оборудованием в спортзале для сообщения им идентификатора клиента;
3. Добавление турникетов на вход и на выход с устройством для чтения идентифицирующего устройства клиента (карты или браслета) и способных аутентифицировать клиента (например, по отпечатку пальца). Помимо аутентификации их функция – фиксировать дату входа клиента в спортзал и дату выхода из него;
4. Создание приложения для пользователя, в котором он сможет отслеживать свои тренировки;

После введение этой системы пользователь получит возможность просто отслеживать свои тренировки в спортзале (что решит проблемы, изложенные в Разделе 1.1).

В свою очередь спортзал станет обладать большим количеством информации о своих клиентах. Рассмотрим некоторые примеры использования этой информации:

1. Имея информацию о времени входа и выхода клиентов фитнес-центр сможет понять, когда зал наиболее загружен или, наоборот, наименее заполнен. Рассмотрим примеры использования этой информации:
   1. Оптимизация количества необходимых тренеров, а также периодичности уборки в зависимости от загруженности;
   2. Введение скидок на посещение зала в определённые часы, когда он наименее заполнен;
2. Имея полную информацию о тренировках клиента фитнес-центр получает возможность применять таргетированную рекламу, например:
   1. Известная проблема при тренировках – остановка прогресса в силовых показателях (например, когда три месяца подряд не можешь подтянуться больше десяти раз). Фитнес-центр может отслеживать такие ситуации и делать специальные предложения клиенту, например: “Остановился прогресс? Мы знаем, что делать! Записывайся к тренеру Сергею и он поможет решить эту проблему”. Это, бесспорно, повысит прибыль бизнеса;
3. Спортзал станет способен предупреждать травму клиентов, анализируя их тренировки, параметры спортсмена, а также его опыт (травмированный клиент может больше не прийти в зал, что понизит возможную прибыль).

# Проектирование

## Концептуальная архитектура

Во взаимодействии участвуют следующие акторы:

1. *Устройства записи*. Это вещи в спортзале с доступом в интернет, которые передают различные данные о пользователе на *Web-сервер*. Среди них:
   1. Тренажеры, публикующие данные об упражнении пользователя;
   2. Турникеты на входе и на выходе, публикующее данные о входе пользователя в спортзал или о выходе пользователя из спортзала;
   3. “Умные весы”, публикующие данные о взвешивании пользователя.
2. *Web-сервер*. Его функции:
   1. Получение данных от *устройств записи* и сохранение их в базе данных;
   2. Взаимодействие с *клиентским приложение* для отображения информации о пользователе;
3. *Клиентской приложение*. Отображает пользователю информацию о тренировках, профиле, взвешиваниях, которую оно получает от *Web-сервера*.

## Построение модели данных

В ходе исследование предметной области были сформулированы требования к реализуемой базе данных, на основе которых была построена модель “сущность-связь” (Рисунок 1), которая впоследствии была преобразована в реляционную модель данных (Рисунок 2).

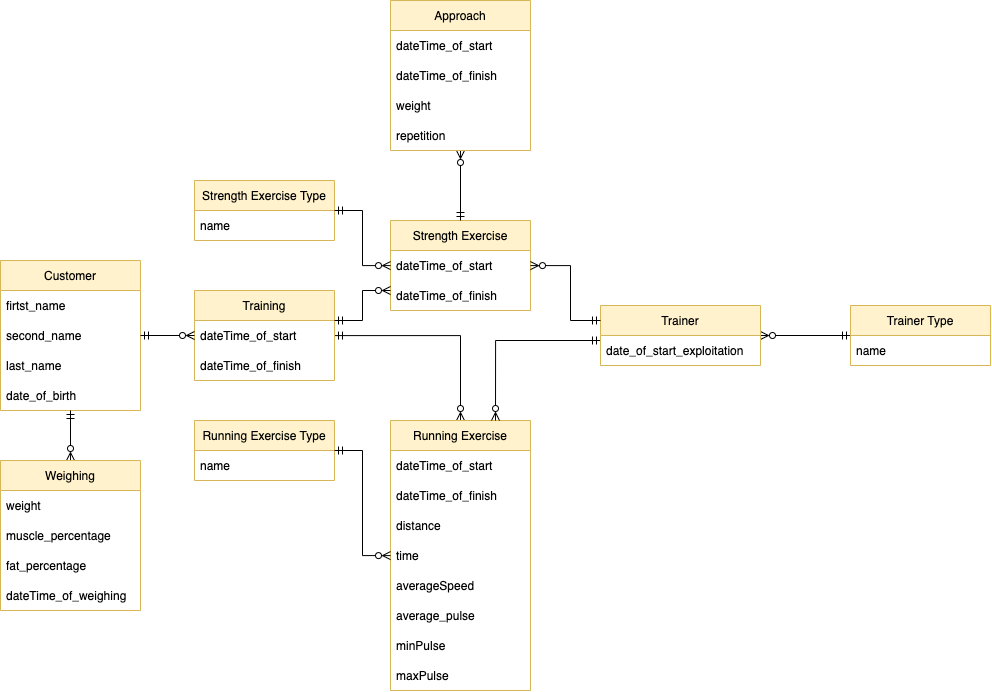


Рисунок 1. Модель данных "сущность-связь".

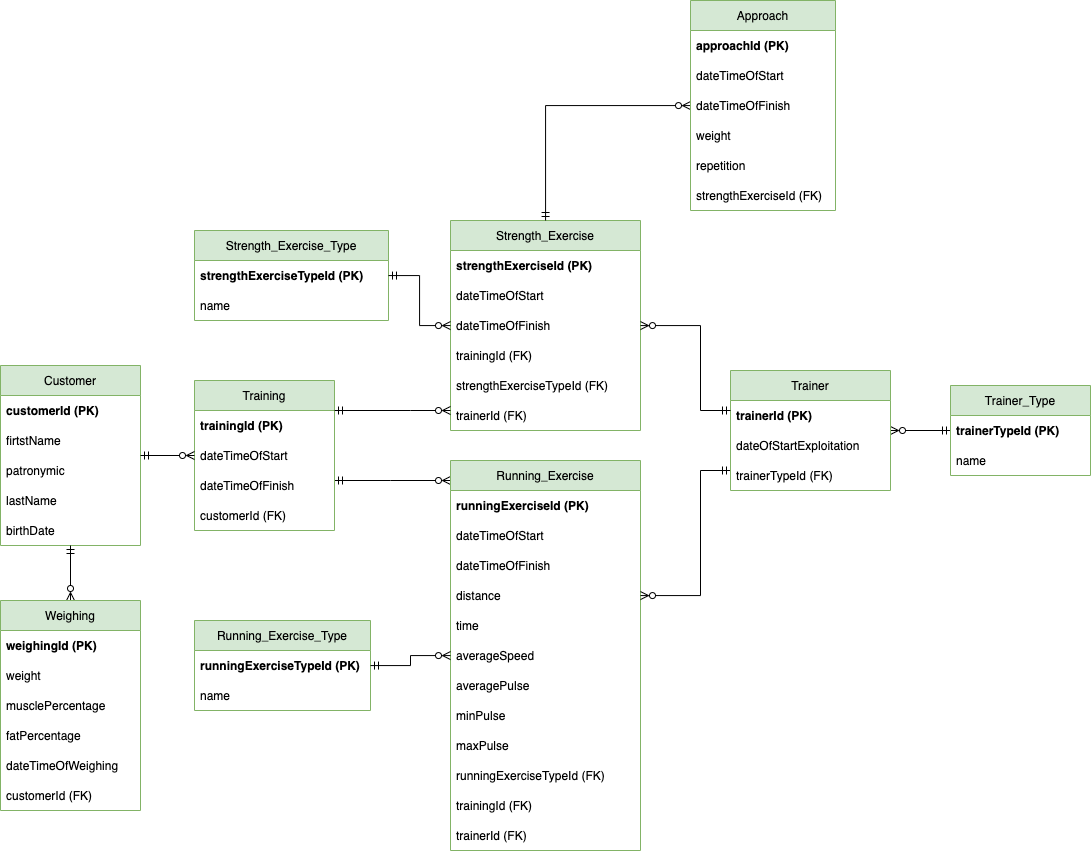


Рисунок 2. Реляционная модель данных.

Далее представлено описание сущностей и их атрибутов из реляционной модели данных (Рисунок 2).

### Сущности в реляционной модели

#### Customer

Данное отношение содержит данные о клиенте спортзала. Таблица 1 содержит атрибуты отношения Customer.

Таблица 1. Атрибуты отношения Customer.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Ключ** | **Тип** | **Значение** | **Описание** |
| customerId | PK | int | NOT NULL | Содержит уникальный идентификатор клиента.  Представляет собой суррогатный ключ. |
| firstName | no | varChar (50) | NOT NULL | Содержит имя клиента. |
| patronymic | no | varChar (50) | NOT NULL | Содержит отчество клиента. |
| lastName | no | varChar (50) | NOT NULL | Содержит фамилию клиента. |
| birthDate | no | Datetime | NOT NULL | Содержит дату рождения клиента. |

#### Weighing

Данное отношение содержит данные о взвешивании клиента, которое он совершил в спортзале. Таблица 2 содержит атрибуты отношения Weighing.

Таблица 2. Атрибуты отношения Weighing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Ключ** | **Тип** | **Значение** | **Описание** |
| weighingId | PK | int | NOT NULL | Содержит уникальный идентификатор взвешивания.  Представляет собой суррогатный ключ. |
| weight | no | float | NOT NULL | Содержит вес клиента, которые определили весы при взвешивании. |
| musclePercentage | no | int | NOT NULL | Содержит процент мышц относительно веса.  Значение находится в диапазоне от 1 до 100. |
| fatPercentage | no | int | NOT NULL | Содержит процент жира относительно веса.  Значение находится в диапазоне от 1 до 100. |
| dateTimeOfWeighing | no | timeStamp | NOT NULL | Содержит дату и время взвешивания. |
| customerId | FK | int | NOT NULL | Содержит уникальный идентификатор клиента, который взвешивался.  Представляет собой ссылку на экземпляр отношения Customer. |

#### Training

Данное отношение содержит информацию о тренировке, которую совершил пользователь в спортзале. Таблица 3 содержит атрибуты отношения Training.

Таблица 3. Атрибуты отношения Training.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Ключ** | **Тип** | **Значение** | **Описание** |
| trainingId | PK | int | NOT NULL | Содержит уникальный идентификатор тренировки.  Представляет собой суррогатный ключ. |
| dateTimeOfStart | no | timeStamp | NOT NULL | Содержит дату и время начала тренировки. |
| dateTimeOfFinish | no | timeStamp | NULL | Содержит дату и время окончания тренировки. |
| customerId | FK | int | NOT NULL | Содержит уникальный идентификатор клиента, который тренировался.  Представляет собой ссылку на экземпляр отношения Customer. |

#### Strength\_Exercise

Данное отношение содержит информацию о силовом упражнении, совершенном клиентом в спортзале в рамках конкретной тренировки. На тренировке может быть совершено несколько силовых упражнений. Примеры силовых упражнений:

1. Жим лёжа;
2. Подтягивания;
3. Тяга верхнего блока;
4. Отжимания.

Таблица 4 содержит атрибуты отношения Strength\_Exercise.

Таблица 4. Атрибуты отношения Strength\_Exercise.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Ключ** | **Тип** | **Значение** | **Описание** |
| strengthExerciseId | PK | int | NOT NULL | Содержит уникальный идентификатор силового упражнения.  Представляет собой суррогатный ключ. |
| dateTimeOfStart | no | TimeStamp | NOT NULL | Содержит дату и время начала силового упражнения. |
| dateTimeOfFinish | no | TimeStamp | May NULL | Содержит дату и время окончания тренировки. |
| trainingId | FK | int | NOT NULL | Содержит уникальный идентификатор тренировки, на которой выполнялось силовое упражнение.  Представляет собой ссылку на экземпляр отношения Training. |
| strengthExerciseTypeId | FK | int | NOT NULL | Содержит уникальный идентификатор типа силового упражнения.  Представляет собой ссылку на экземпляр отношения Strength\_Exercise\_Type. |
| trainerId | FK | int | NOT NULL | Содержит уникальный идентификатор тренажёра, на котором выполнялось упражнение.  Представляет собой ссылку на экземпляр отношения Trainer. |

#### Strength\_Exercise\_Type

Данное отношение содержит информацию о допустимых типах силовых упражнениях. Таблица 5 содержит атрибуты отношения Strength\_Exercise\_Type.

Таблица 5. Атрибуты отношения Strength\_Exercise\_Type.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Ключ** | **Тип** | **Значение** | **Описание** |
| strengthExerciseTypeId | PK | int | NOT NULL | Содержит уникальный идентификатор типа силового упражнения.  Представляет собой суррогатный ключ. |
| name | no | varChar (50) | NOT NULL | Содержит название типа силового упражнения. Например, “тяга верхнего блока”. |

#### Approach

Данное отношение содержит информацию об одном повторении в рамках силового упражнения. Например: силовое упражнение состоит из 4 повторений: 10 подтягиваний, 15 подтягиваний, 20 подтягиваний, 25 подтягиваний. Таблица 6 содержит атрибуты отношения Approach.

Таблица 6. Атрибуты отношения Approach.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Ключ** | **Тип** | **Значение** | **Описание** |
| repetitionId | PK | int | NOT NULL | Содержит уникальный идентификатор повторения.  Представляет собой суррогатный ключ. |
| dateTimeOfStart | no | timeStamp | NOT NULL | Содержит дату и время начала выполнения повторения. |
| dateTimeOfFinish | no | timeStamp | NOT NULL | Содержит дату и время окончания выполнения повторения. |
| weight | no | float | NOT NULL | Содержит вес, который использовался в повторении (в кг). Например, 60 кг.  Значение находится в диапазоне от 0 до 1000. |
| repetition | no | int | NOT NULL | Содержит количество повторений в подходе. |
| strengthExerciseId | FK | int | NOT NULL | Содержит уникальный идентификатор силового упражнения, в рамках которого выполняется подход.  Содержит ссылку на экземпляр отношения Strength\_Exercise |

#### Running\_Exercise

Данное отношение содержит информацию об одном беговом упражнении клиент. Примеры беговых упражнений:

1. Бег на беговой дорожке;
2. Работа на велосипедном тренажере.

Таблица 7 содержит атрибуты отношения Running\_Exercise.

Таблица 7. Атрибуты отношения Running\_Exercise.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Ключ** | **Тип** | **Значение** | **Описание** |
| runningExerciseId | PK | int | NOT NULL | Содержит уникальный идентификатор бегового упражнения.  Представляет собой суррогатный ключ. |
| dateTimeOfStart | no | timeStamp | NOT NULL | Содержит дату и время начала бегового упражнения. |
| dateTimeOfFinish | no | timeStamp | NOT NULL | Содержит дату и время окончания тренировки. |
| distance | no | float | NOT NULL | Содержит дистанцию (в километрах), которую преодолел клиент в данном упражнении. |
| time | no | float | NOT NULL | Содержит время, за которое клиент сделал беговое упражнение (в минутах). |
| averageSpeed | no | int | NOT NULL | Содержит среднее значение скорости, с которой клиент выполнил беговое упражнение (км/ч). |
| averagePulse | no | int | NULL | Содержит среднее значение пульса клиента, которое было у него в момент выполнения упражнения.  Значение находится в диапазоне от 0 до 300. |
| minPulse | no | int | NULL | Содержит минимальное значение пульса, которое было у клиента в момент выполнения бегового упражнения.  Значение находится в диапазоне от 0 до 300. |
| maxPulse | no | int | NULL | Содержит максимальное значение пульса, которое было у клиента в момент выполнения бегового упражнения  Значение находится в диапазоне от 0 до 300. |
| runningExerciseTypeId | FK | int | NOT NULL | Содержит уникальный идентификатор типа бегового упражнения.  Представляет собой ссылку на экземпляр отношения Running\_Exercise\_Type. |
| trainingId | FK | int | NOT NULL | Содержит уникальный идентификатор тренировки, на которой выполнялось беговое упражнение.  Представляет собой ссылку на экземпляр отношения Training. |
| trainerId | FK | int | NOT NULL | Содержит уникальный идентификатор тренажёра, на котором выполнялось упражнение.  Представляет собой ссылку на экземпляр отношения Trainer. |

#### Running\_Exercise\_Type

Данное отношение содержит информацию о типе бегового упражнения. Таблица 8 содержит атрибуты отношения Running\_Exercise\_Type.

Таблица 8. Атрибуты отношения Running\_Exercise\_Type.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Ключ** | **Тип** | **Значение** | **Описание** |
| strengthExerciseTypeId | PK | int | NOT NULL | Содержит уникальный идентификатор типа бегового упражнения.  Представляет собой суррогатный ключ. |
| name | no | varChar (50) | NOT NULL | Содержит название типа силового упражнения. Например, “бег”. |

#### Trainer

Данное отношение содержит информацию о тренажере в спортзале. Таблица 9 содержит атрибуты отношения Trainer.

Таблица 9. Атрибуты отношения Trainer.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Ключ** | **Тип** | **Значение** | **Описание** |
| trainerId | PK | int | NOT NULL | Содержит уникальный идентификатор тренажёра.  Представляет собой суррогатный ключ. |
| dateOfStartExploitation | no | dateTime | NOT NULL | Содержит время начала эксплуатации тренажёра. |
| trainerTypeId | FK | int | NOT NULL | Содержит уникальный идентификатор типа тренажёра.  Представляет собой ссылку на экземпляр отношения Trainer\_Type. |

#### Trainer\_Type

Данное отношение содержит информацию о типе тренажёра. Таблица 10 содержит атрибуты отношения Trainer\_Type.

Таблица 10. Атрибуты отношения Trainer\_Type.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя столбца** | **Ключ** | **Тип** | **Значение** | **Описание** |
| trainerTypeId | PK | int | NOT NULL | Содержит уникальный идентификатор типа тренажёра.  Представляет собой суррогатный ключ. |
| name | no | varChar (50) | NOT NULL | Содержит название типа тренажёра. Например, “беговая дорожка”. |

### Правила обеспечения связей между отношениями

Для связей между отношениями из реляционной модели данных были разработаны правила обеспечения ограничений минимальной кардинальности, которые описаны в следующих разделах.

#### Customer-Weighing

Связь между отношениями Customer и Weighing имеет тип M-O, 1-N, так как взвешивание может быть записано только для одного существующего клиента. Клиент может иметь любое количество взвешиваний, в том числе и 0, если он ни разу не взвешивался. Таблица 11 содержит описание правил для связи Customer-Weighing.

Таблица 11. Правила для связи Customer-Weighing.

|  |  |  |
| --- | --- | --- |
| **Операция** | **Действия для Customer (Родитель)** | **Действия для Weighing (Ребёнок)** |
| Вставка | – | При корректности FK customerId |
| Изменение первичного ключа или внешнего ключа | Запрещено – суррогатный ключ | Запрещено – нельзя изменять перезаписать взвешивания на другого клиента |
| Удаление | Каскадное удаление | – |

#### Customer-Training

Связь между отношениями Customer и Training имеет тип M-O, 1-N, так как тренировка может быть записано только для одного существующего клиента. Клиент может иметь любое количество тренировок, в том числе и 0, если он ни разу не тренировался. Таблица 12 содержит описание правил для связи Customer-Training.

Таблица 12. Правила для связи Customer-Training.

|  |  |  |
| --- | --- | --- |
| **Операция** | **Действия для Customer (Родитель)** | **Действия для Training (Ребёнок)** |
| Вставка | – | При корректности FK customerId |
| Изменение первичного ключа или внешнего ключа | Запрещено – суррогатный ключ | Запрещено – нельзя перезаписать тренировку на другого клиента |
| Удаление | Каскадное удаление | – |

#### Training-Strength\_Exercise

Связь между отношениями Training и Strength\_Exercise имеет тип M-O, 1-N, так как силовое упражнение может быть сделано лишь в существующей тренировке (причем только в одной). Тренировка может содержать любое количество силовых упражнений, в том числе и 0, если клиент не сделал на тренировке ни одного силового упражнения. Таблица 13 содержит описание правил для связи Training-Strength\_Exercise.

Таблица 13. Правила для связи Training-Strength\_Exercise.

|  |  |  |
| --- | --- | --- |
| **Операция** | **Действия для Training (Родитель)** | **Действия для Strength\_Exercise (Ребёнок)** |
| Вставка | – | При корректности FK trainingId |
| Изменение первичного ключа или внешнего ключа | Запрещено – суррогатный ключ | Запрещено – нельзя перезаписать упражнение в другую тренировку |
| Удаление | Каскадное удаление | – |

#### Training-Running\_Exercise

Связь между отношениями Training и Running\_Exercise имеет тип M-O, 1-N, так как беговое упражнение может быть сделано лишь в существующей тренировке (причем только в одной). Тренировка может содержать любое количество беговых упражнений, в том числе и 0, если клиент не сделал на тренировке ни одного бегового упражнения. Таблица 14 содержит описание правил для связи Training-Running\_Exercise.

Таблица 14. Правила для связи Training-Running\_Exercise.

|  |  |  |
| --- | --- | --- |
| **Операция** | **Действия для Training (Родитель)** | **Действия для Running\_Exercise (Ребёнок)** |
| Вставка | – | При корректности FK trainingId |
| Изменение первичного ключа или внешнего ключа | Запрещено – суррогатный ключ | Запрещено – нельзя перезаписать упражнение в другую тренировку |
| Удаление | Каскадное удаление | – |

#### Running\_Exercise\_Type -Running\_Exercise

Связь между отношениями Running\_Exercise\_Type и Running\_Exercise имеет тип M-O, 1-N, так как беговое упражнение может иметь строго один тип, а несколько беговых упражнений могут иметь одинаковый тип (также возможно, что тип бегового упражнения существует, но ни одного упражнения такого типа нет). Таблица 15 содержит описание правил для связи Running\_Exercise\_Type-Running\_Exercise.

Таблица 15. Правила для связи Running\_Exercise\_Type-Running\_Exercise.

|  |  |  |
| --- | --- | --- |
| **Операция** | **Действия для Running\_Exercise\_Type (Родитель)** | **Действия для Running\_Exercise (Ребёнок)** |
| Вставка | – | При корректности FK runningExerciseId |
| Изменение первичного ключа или внешнего ключа | Запрещено – суррогатный ключ | Запрещено, нельзя изменить тип бегового упражнения |
| Удаление | Каскадное удаление | – |

#### Strength\_Exercise\_Type-Strength\_Exercise

Связь между отношениями Strength\_Exercise\_Type и Strength\_Exercise имеет тип M-O, 1-N, так как силовое упражнение может иметь строго один тип, а несколько силовых упражнений могут иметь одинаковый тип (также возможно, что тип силового упражнения существует, но ни одного упражнения такого типа нет). Таблица 16 содержит описание правил для связи Strength\_Exercise\_Type-Strength\_Exercise.

Таблица 16. Правила для связи Strength\_Exercise\_Type-Strength\_Exercise.

|  |  |  |
| --- | --- | --- |
| **Операция** | **Действия для Strength\_Exercise\_Type (Родитель)** | **Действия для Strength\_Exercise (Ребёнок)** |
| Вставка | – | При корректности FK strengthExerciseTypeId |
| Изменение первичного ключа или внешнего ключа | Запрещено – суррогатный ключ | Запрещено, нельзя изменить тип силового упражнения |
| Удаление | Каскадное удаление | – |

#### Strength\_Exercise-Approach

Связь между отношениями Strength\_Exercise и Approach имеет тип M-O, 1-N, так как подход на тренажёре всегда выполнятся в рамках одного конкретного силового упражнения клиента, а одно упражнение может содержать любое количество подходов. Таблица 17 содержит описание правил для связи Strength\_Exercise\_Approach.

Таблица 17. Правила для связи Strength\_Exercise-Approach.

|  |  |  |
| --- | --- | --- |
| **Операция** | **Действия для Strength\_Exercise (Родитель)** | **Действия для Approach (Ребёнок)** |
| Вставка | – | При корректности FK strengthExerciseId |
| Изменение первичного ключа или внешнего ключа | Запрещено – суррогатный ключ | Запрещено, подход строго привязан к упражнению |
| Удаление | Каскадное удаление | – |

#### Trainer-Strength\_Exercise

Связь между отношениями Trainer и Strength\_Exercise имеет тип M-O, 1-N, так как силовое упражнение всегда выполняется на одном конкретном тренажёре, на одном тренажере также может быть выполнено любое количество силовых упражнений (в том числе и ноль, если на нём никто не занимался). Таблица 18 содержит описание правил для связи Trainer-Strength\_Exercise.

Таблица 18. Правила для связи Trainer-Strength\_Exercise.

|  |  |  |
| --- | --- | --- |
| **Операция** | **Действия для Trainer (Родитель)** | **Действия для Strength\_Exercise (Ребёнок)** |
| Вставка | – | При корректности FK trainerId |
| Изменение первичного ключа или внешнего ключа | Запрещено – суррогатный ключ | Запрещено, упражнение строго привязано к тренажёру |
| Удаление | Каскадное удаление | – |

#### Trainer-Running\_Exercise

Связь между отношениями Trainer и Running\_Exercise имеет тип M-O, 1-N, так как беговое упражнение всегда выполняется на одном конкретном тренажёре, на одном тренажере также может быть выполнено любое количество беговых упражнений (в том числе и ноль, если на нём никто не занимался). Таблица 19 содержит описание правил для связи Trainer-Running\_Exercise.

Таблица 19. Правила для связи Trainer-Running\_Exercise.

|  |  |  |
| --- | --- | --- |
| **Операция** | **Действия для Trainer (Родитель)** | **Действия для Running\_Exercise (Ребёнок)** |
| Вставка | – | При корректности FK trainerId |
| Изменение первичного ключа или внешнего ключа | Запрещено – суррогатный ключ | Запрещено, упражнение строго привязано к тренажёру |
| Удаление | Каскадное удаление | – |

#### Trainer\_Type-Trainer

Связь между отношениями Trainer\_Type и Training имеет тип M-O, 1-N, так как у каждого тренажёра может быть строго один тип, разные тренажёры могут иметь одинаковый тип, также в спортзале может не быть тренажёра определённого типа. Таблица 20 содержит описание правил для связи Trainer\_Type-Trainer.

Таблица 20. Правила для связи Trainer\_Type-Trainer.

|  |  |  |
| --- | --- | --- |
| **Операция** | **Действия для Trainer\_Type (Родитель)** | **Действия для Trainer (Ребёнок)** |
| Вставка | – | При корректности FK trainerTypeId |
| Изменение первичного ключа или внешнего ключа | Запрещено – суррогатный ключ | Запрещено, тип тренажёра запрещено менять |
| Удаление | Каскадное удаление | – |

# Реализация

Был реализован RESTful API сервис, представляющий собой web-приложение, включающее в себя две компоненты *Web-сервер*, *клиентское приложение,* а также функционал *устройств записи* по отправки различных данных о пользователе. Для реализации приложения использовался фреймворк flask [2], написанный на языке Python. В качестве базы данных, используемой для хранения данных, была использована PostgreSQL [3]. Для взаимодействия с базой данных PostgreSQL использовалась библиотека psycopg [4]. Для отправки HTTP запросов (для симуляции работы *устройств записи*) использовалось приложение [5].

## Реализация базы данных

Была создана локальная база данных под название gym\_training\_tracking, в которой были инициализированы все отношения (листинг 11), предусмотренные моделью данных (Рисунок 2), а также все необходимые правила обеспечения минимальной кардинальности (смотреть раздел Правила обеспечения связей между отношениями). Для работы с СУБД PostgreSQL были написаны функции с использованием библиотеки psycopg (листинг 12).

## Написание веб-приложения

Было написано web-приложение с использованием фреймворка flask. Рассмотрим наиболее интересный момент – обработка запросов от *устройств записи.* Для использования доступны следующие запросы:

1. Запросы от входного турникета. Запрос представлен на листинге 1, а его обработчик на листинге 2.
2. Запросы от турникета на выход. Запрос представлен на листинге 3, а его обработчик на листинге 4. Заметим, что так как турникет не знает id созданной пользователем тренировки, то приходится использовать query параметр lastTraining.
3. Запросы на добавление взвешивания от “умных весов”. Запрос представлен на листинге 5, а его обработчик на листинге 6.
4. Запросы от беговых тренажёров на добавление беговых упражнений. Запрос представлен на листинге 7, а его обработчик на листинге 8. Заметим, что так как тренажёр не знает идентификатор тренировки пользователя, приходится использовать query параметры.
5. Запросы от силовых тренажёров на добавление силовых упражнений. Запрос представлен на листинге 9, а его обработчик на листинге 10. Заметим, что так как тренажёр не знает идентификатор тренировки пользователя, приходится использовать query параметры.

Листинг 1. Пример запроса от входного турникета.

POST http://localhost:3000/customers/5/trainings

{

"datetimeOfStart": "2022-09-13 10:23:54"

}

Листинг 2. Обработчик запроса от входного турникета.

#Турникет на входе фиксирует приход клиента

def customer\_came(customer\_id):

request\_data = request.get\_json()

datetime\_of\_start = request\_data['datetimeOfStart']

status = db.insert('training', {

'datetimeOfStart': datetime\_of\_start,

'customerId': customer\_id

})

if status:

return "training added successfully"

else:

return "training cannot be added"

Листинг 3. Пример запроса от турникета на выход (завершение посетителем тренировки).

PATCH http://localhost:3000/customers/5/trainings?&lastTraining=true

{

"datetimeOfFinish": "2022-09-13 12:23:54"

}

Листинг 4. Обработчик запроса от входного турникета.

#Клиент выходит из спортзала, тренировка окончена

@app.route('/customers/<int:customer\_id>/trainings', methods=['PATCH'])

def complete\_training(customer\_id):

queries = request.args.to\_dict()

if queries.get('lastTraining'):

request\_data = request.get\_json()

datetime\_of\_finish = request\_data['datetimeOfFinish']

trainingId = db.get\_last\_training(customer\_id)

if db.complete\_training(trainingId, datetime\_of\_finish):

return "training completed successfully"

return "training cannot be completed"

Листинг 5. Пример запроса на публикацию нового взвешивания пользователя от “умных весов”.

POST http://localhost:3000/customers/5/weighings

{

"weight": 65,

"musclePercentage": 30,

"fatPercentage": 9,

"datetimeOfWeighing": "2022-09-13 12:23:54"

}

Листинг 6. Обработчик запроса от “умных весов”.

#весы добавили данные о взвешивании клиента

@app.route('/customers/<int:customer\_id>/weighings', methods=['POST'])

def add\_weighing(customer\_id):

request\_data = request.get\_json()

if db.insert('weighing', {

'weight': request\_data['weight'],

'musclePercentage': request\_data['musclePercentage'],

'fatPercentage': request\_data['fatPercentage'],

'datetimeOfWeighing': request\_data['datetimeOfWeighing'],

'customerId': customer\_id}):

return "weighing added successfully"

return "weighing cannot be added"

Листинг 7. Пример запроса на публикацию нового бегового упражнения пользователя от бегового тренажёра

POST http://localhost:3000/customers/5/trainings?addExercise=True&exerciseType=running

{

"datetimeOfStart": "2022-09-14 12:23:54",

"datetimeOfFinish": "2022-09-14 13:23:54",

"distance": 12.2,

"time": 60,

"averageSpeed": 11,

"averagePulse": 115,

"minPulse": 60,

"maxPulse": 140,

"runningExerciseTypeId": 1,

"trainerId": 1

}

Листинг 8. Обработчик запроса на публикацию бегового упражнения.

#Беговой тренажёр добавляет новую тренировку пользователя

def add\_running\_exercise(data, training\_id):

if db.insert('running\_exercise', {

'datetimeOfStart': data['datetimeOfStart'],

'datetimeOfFinish': data['datetimeOfFinish'],

'distance': data['distance'],

'time': data['time'],

'averageSpeed': data['averageSpeed'],

'averagePulse': data['averagePulse'],

'minPulse': data['minPulse'],

'maxPulse': data['maxPulse'],

'runningExerciseTypeId': data['runningExerciseTypeId'],

'trainerId': data['trainerId'],

'trainingId': training\_id

}):

return "running exercise added successfully"

else:

return "running exercise cannot be added"

Листинг 9. Пример запроса на публикацию нового силового упражнения пользователя от силового тренажёра

POST http://localhost:3000/customers/5/trainings?addExercise=True&exerciseType=strength

{

"datetimeOfStart": "2022-09-18 12:00:00",

"datetimeOfFinish": "2022-09-18 14:00:00",

"strengthExerciseTypeId": 1,

"trainerId": 2,

"approaches": [

{

"datetimeOfStart": "2022-09-18 12:00:00",

"datetimeOfFinish": "2022-09-18 12:03:00",

"weight": 35,

"repetition": 12

},

{

"datetimeOfStart": "2022-09-18 12:04:00",

"datetimeOfFinish": "2022-09-18 12:07:00",

"weight": 40,

"repetition": 10

},

{

"datetimeOfStart": "2022-09-18 12:10:00",

"datetimeOfFinish": "2022-09-18 12:13:00",

"weight": 45,

"repetition": 8

},

{

"datetimeOfStart": "2022-09-18 12:16:00",

"datetimeOfFinish": "2022-09-18 12:18:00",

"weight": 50,

"repetition": 6

}

]

}

Листинг 10. Обработчик запроса на публикацию силового упражнения.

#Силовой тренажёр добавляет новую тренировку пользователя

def add\_strength\_exercise(data, training\_id):

strengthExerciseId = db.insert\_strengthExercise({

'datetimeOfStart': data['datetimeOfStart'],

'datetimeOfFinish': data['datetimeOfFinish'],

'strengthExerciseTypeId': data['strengthExerciseTypeId'],

'trainerId': data['trainerId'],

'trainingId': training\_id

})[0]

if strengthExerciseId:

flag = True

for approach in data['approaches']:

flag = flag & db.insert('approach', {

'datetimeOfStart': approach['datetimeOfStart'],

'datetimeOfFinish': approach['datetimeOfFinish'],

'weight': approach['weight'],

'repetition': approach['repetition'],

'strengthExerciseId': strengthExerciseId

})

if flag:

return "strength exercise added successfully"

else: return "strength exercise cannot be added"

else:

return "strength exercise cannot be added"

## Пользовательский интерфейс

### Страница регистрации

Данная страница (Рисунок 3) содержит форму для регистрации клиента в спортзале. Для регистрации ему необходимо ввести номер телефона, электронную почту и пароль.



Рисунок 3. Страницы регистрации клиента.

### Страница входа

Данная страница (Рисунок 4) содержит форму для входа в аккаунт, в котором отображаются все сделанные им тренировки.

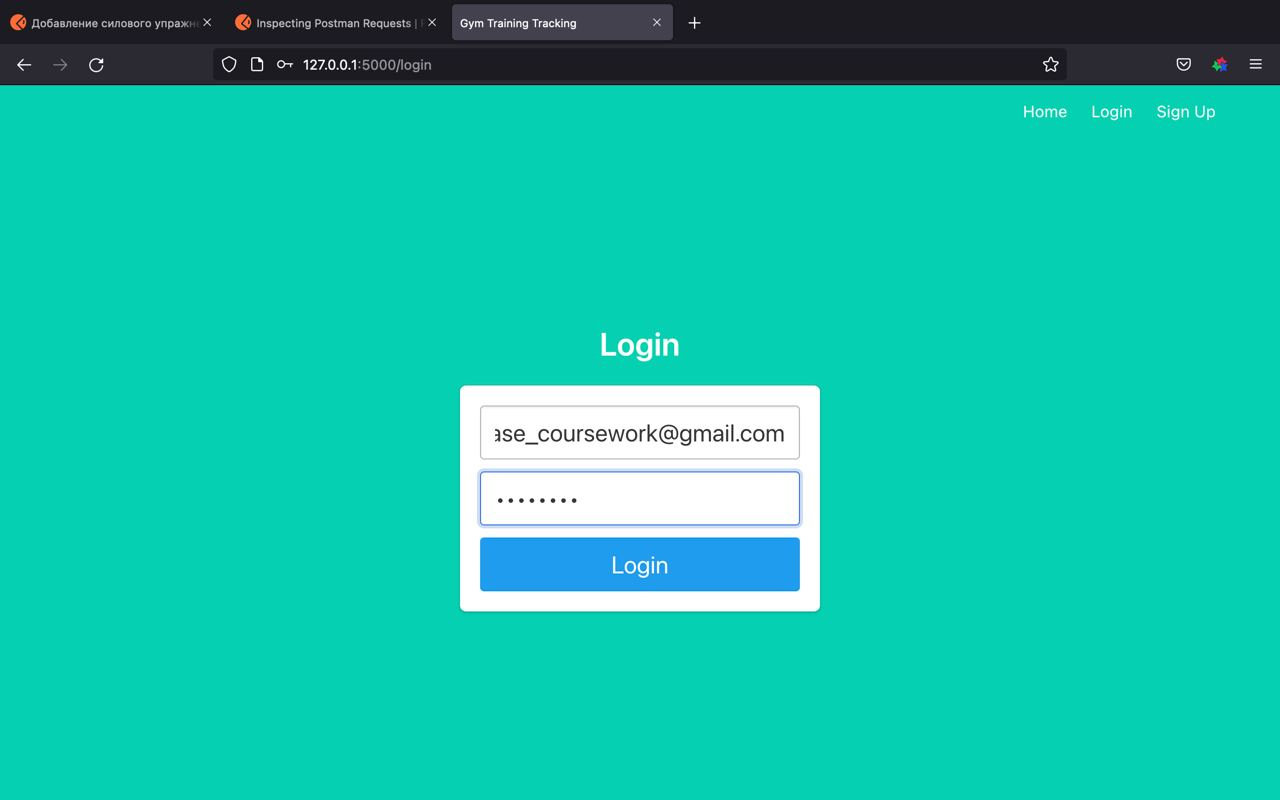


Рисунок 4. Страница аутентификации.

### Страница тренировок клиента

На данной странице (Рисунок 5) отображаются все тренировки клиента с указанием даты и времени начала и конца.

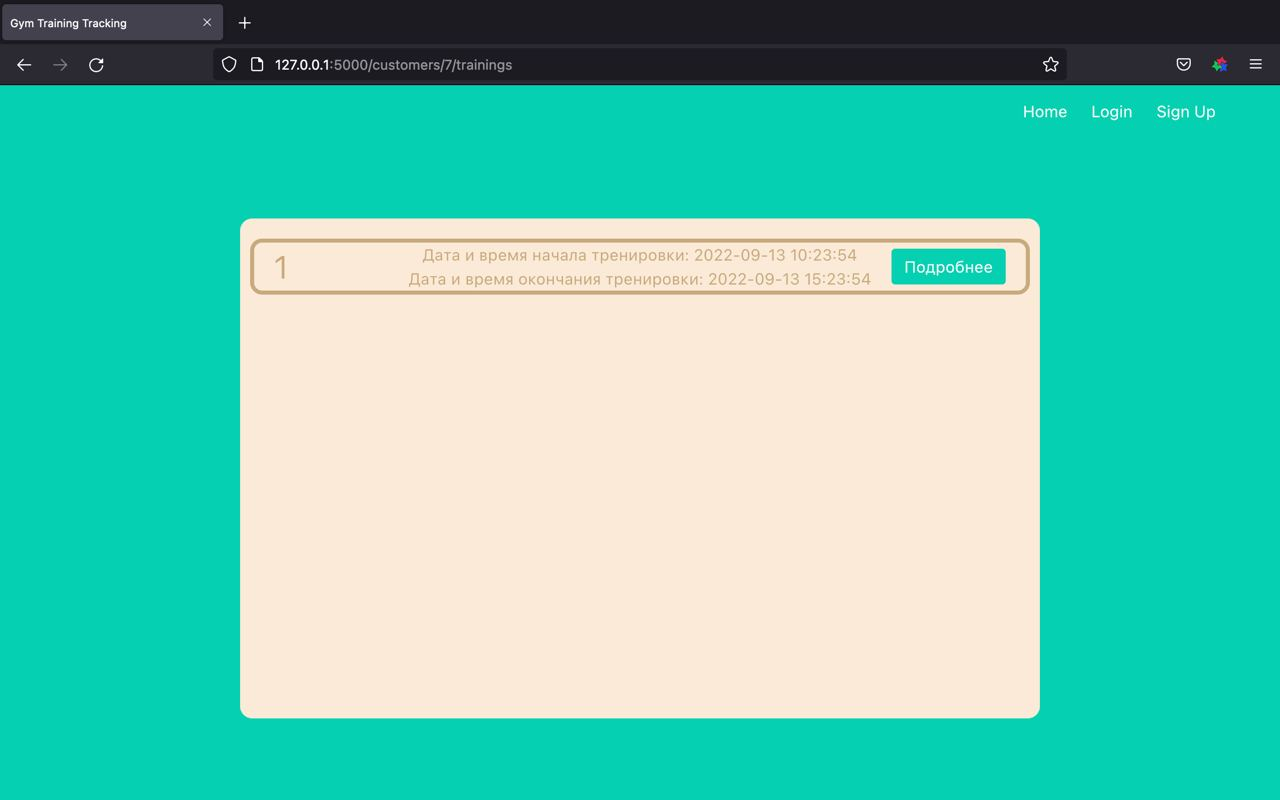


Рисунок 5. Страница тренировок клиента.

### Страница упражнений в тренировке

По клику на кнопку подробнее на странице тренировок открывается страница с перечислением всех упражнений, сделанных на ней (Рисунок 6). У беговых упражнений отображается полная информация об упражнении, а у силовых время и дата конца и начала с кнопкой “Подробнее”, при нажатии на которую пользователь переходит на страницу подходов в силовой тренировке.

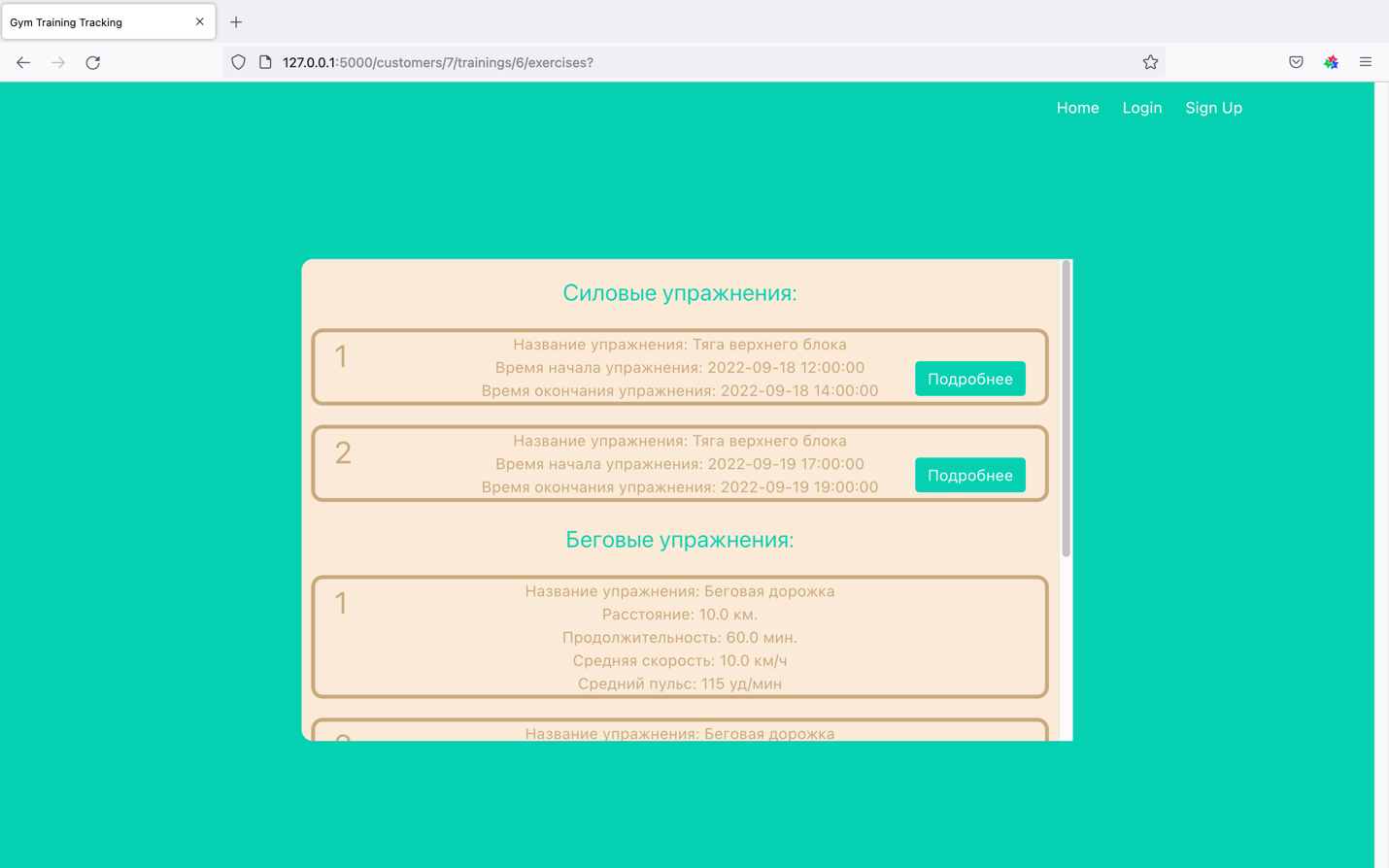
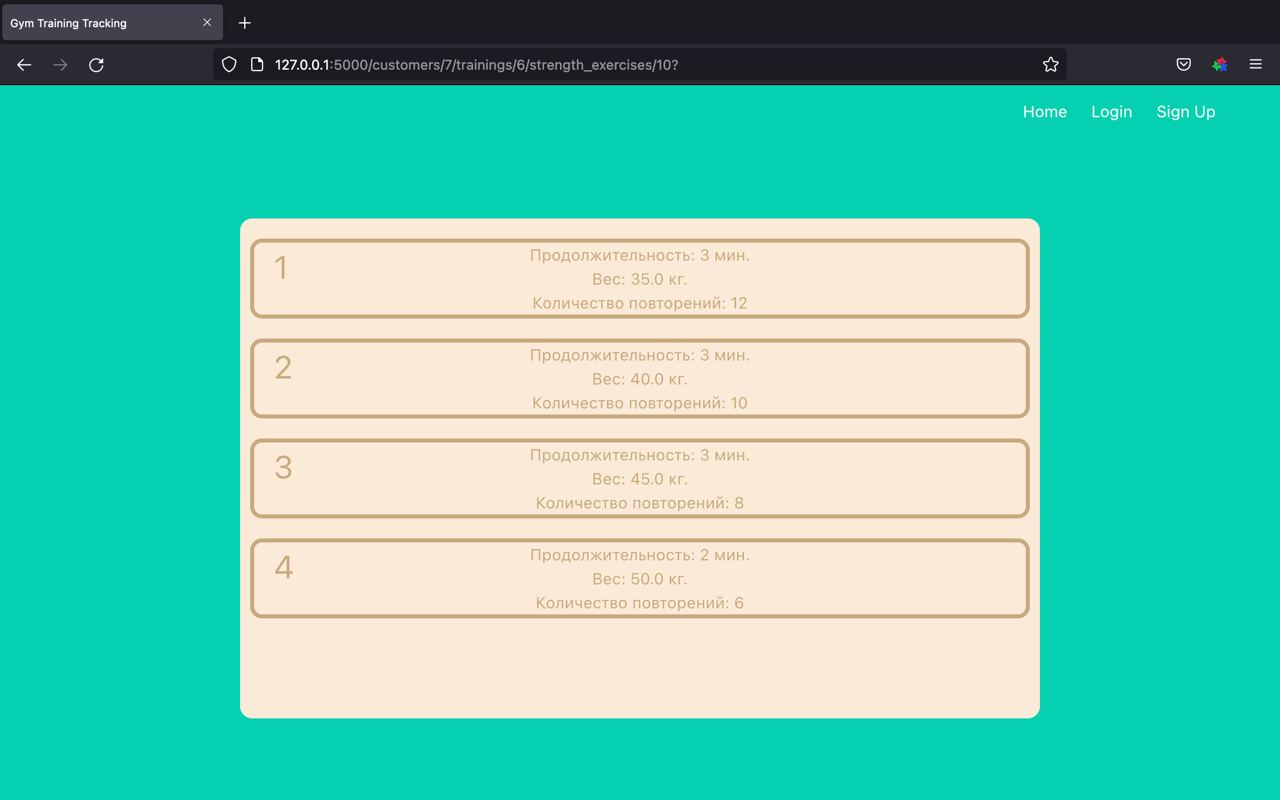


Рисунок 6. Страница упражнений в тренировке.

### Страница подходов в силовой тренировке

По клику на силовое упражнение в странице тренировок открывается вкладка, на которой представлено подробное описание всех подходов в рамках силового упражнения.



# Тестирование

Тестирование проводилось на оборудовании, оснащенном процессором Apple M1 8 core с базовой тактовой частотой 3.20 ГГц и оперативной памятью объёмом 16 ГБ. Оно состояло в последовательной проверке корректности работы нового функционала по мере его добавления. Например, проверки корректности произведения аутентификации. При выявлении факта некорректной работы производилось исправление неточностей в программе, которые привели к ней.

На Рисунке 7 демонстрируется работа клиентского приложения при вводе неправильных данных для аутентификации (электронной почты или пароля).

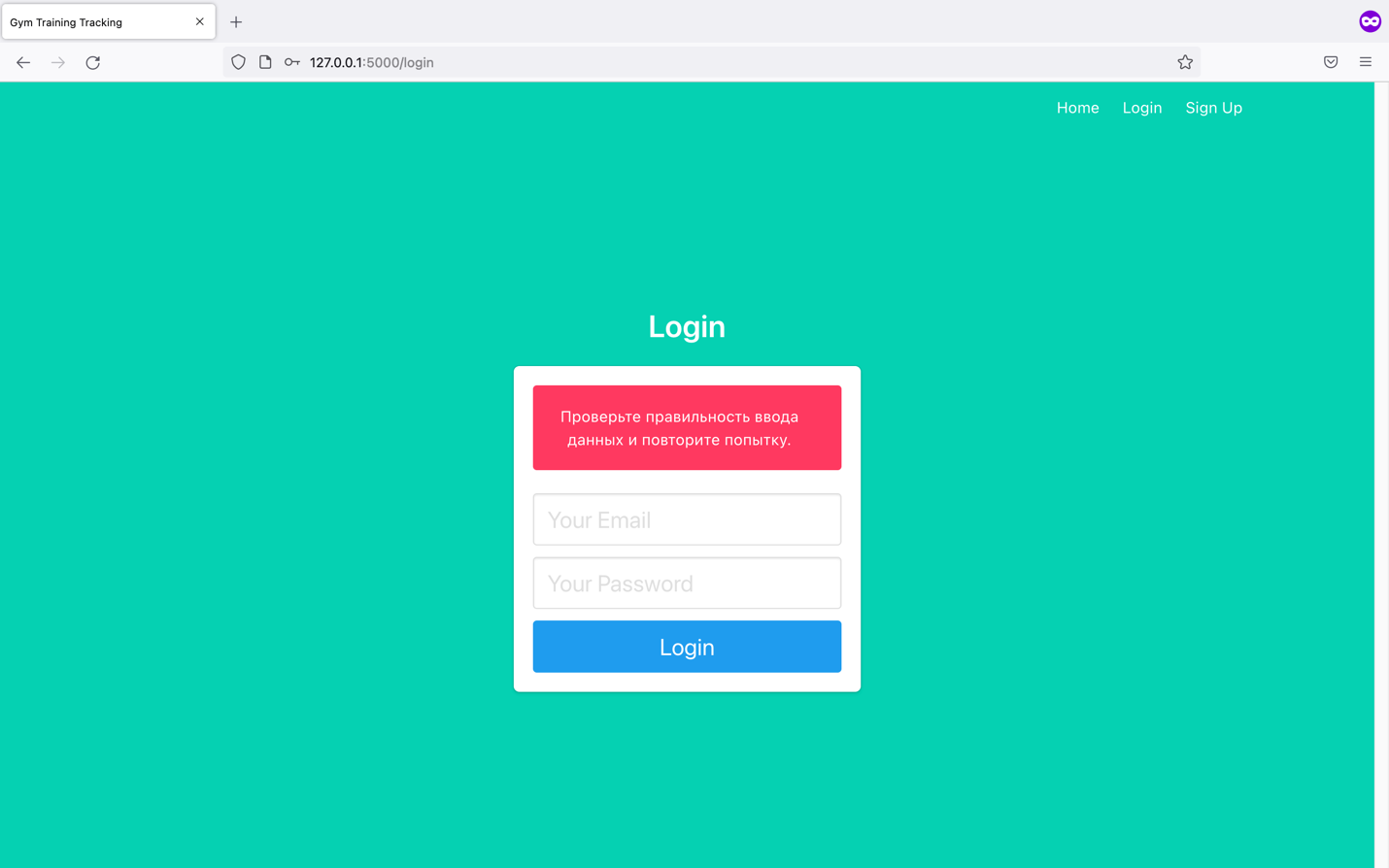


Рисунок 7. Проверка корректности работы системы аутентификации.

На рисунках 8, 9, 10 представлено тестирование процесса добавления новой тренировки. Сначала входной турникет отправляет HTTP запрос, содержащий информацию о том, что занятие началось (Рисунок 8), затем информация о тренировке добавляется в базу данных (Рисунок 9), далее происходит отображение информации в клиентском приложении пользователя (Рисунок 10).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8. Отправка HTTP запроса на создание тренировки.

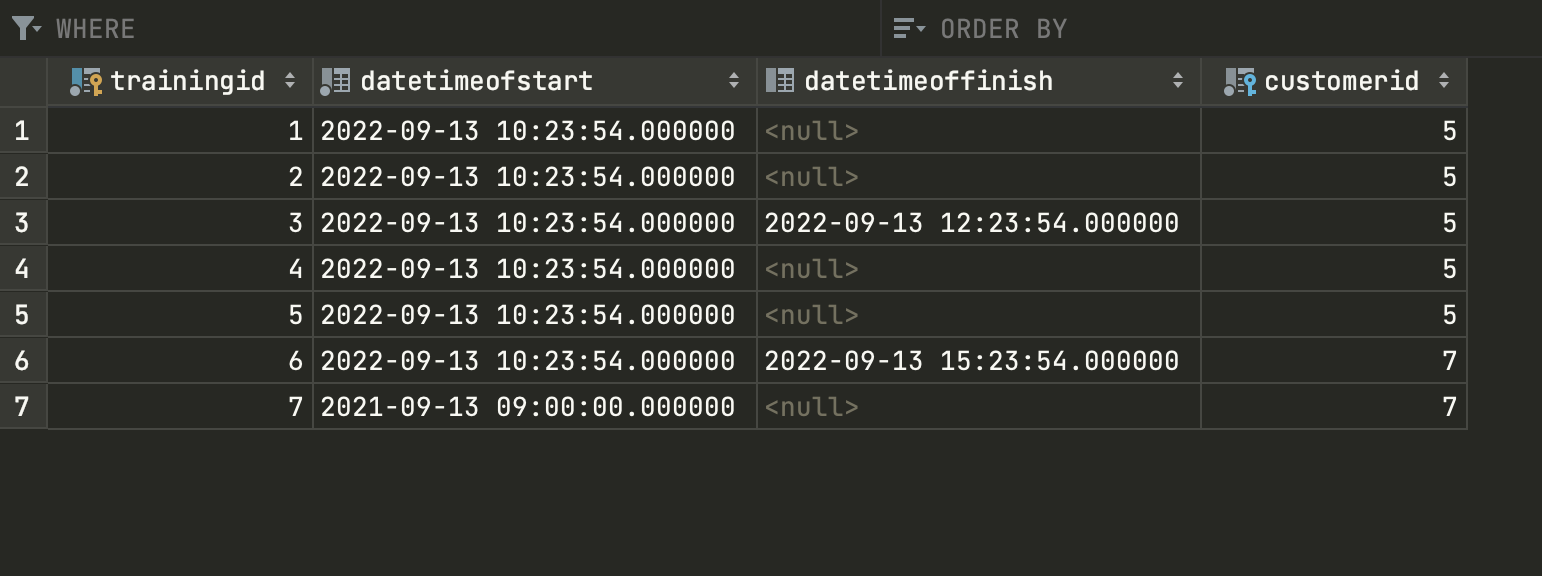


Рисунок 9. Демонстрация появления новой записи о тренировке (с индексом 7) в базе данных.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10. Демонстрация отображения новой тренировки у пользователя.

ЗАКЛЮЧЕНИЕ

В данной курсовой работе был разработан и протестирован прототип web-сервиса, предоставляющий базовый функционал для отслеживания тренировок клиентом. Спортзалы, которые внедрят данный сервис и реализуют необходимую инфраструктуру, смогут не только улучшить денежный поток, пользуясь расширенными данными о клиенте, но и сильно упростят жизнь своим посетителям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Статистика по вовлеченности в спорт среди россиян. [Электронный ресурс] – URL: https://ria.ru/20180828/1527330741.html
2. Документация фреймворка flask. [Электронный ресурс] – URL: https://flask.palletsprojects.com/en/2.2.x/
3. Документация базы данных PostgreSQL. [Электронный ресурс] –­ https://www.postgresql.org/
4. Документация библиотеки для работы с PostgreSQL. [Электронный ресурс] – https://pypi.org/project/psycopg2/
5. Документация сервиса Postman. [Электронный ресурс] – https://www.postman.com/

ПРИЛОЖЕНИЕ А

Листинг 11. Инициализация базы данных и создания в ней таблиц.

DROP TABLE IF EXISTS Customer CASCADE;

CREATE TABLE IF NOT EXISTS Customer (

customerId SERIAL PRIMARY KEY NOT NULL,

firstName VARCHAR (50) NULL,

patronymic VARCHAR (50) NULL,

lastName VARCHAR (50) NULL,

birthDate DATE NULL,

email VARCHAR (50) UNIQUE NOT NULL,

phoneNumber VARCHAR (11) UNIQUE NOT NULL,

password VARCHAR (50) NOT NULL

);

DROP TABLE IF EXISTS Weighing CASCADE;

CREATE TABLE IF NOT EXISTS Weighing (

weighingId SERIAL PRIMARY KEY NOT NULL,

weight FLOAT4 NOT NULL,

musclePercentage INT CHECK (1 <= musclePercentage AND musclePercentage <= 100) NOT NULL,

fatPercentage INT CHECK (1 <= musclePercentage AND musclePercentage <= 100) NOT NULL,

dateTimeOfWeighing TIMESTAMP NOT NULL,

customerId INT NOT NULL REFERENCES Customer (customerId)

);

DROP TABLE IF EXISTS Training CASCADE;

CREATE TABLE IF NOT EXISTS Training (

trainingId SERIAL PRIMARY KEY NOT NULL,

dateTimeOfStart TIMESTAMP NOT NULL,

dateTimeOfFinish TIMESTAMP NULL,

customerId INT NOT NULL REFERENCES Customer (customerId)

);

DROP TABLE IF EXISTS Trainer\_Type CASCADE;

CREATE TABLE IF NOT EXISTS Trainer\_Type (

trainerTypeId SERIAL PRIMARY KEY NOT NULL,

name VARCHAR (100) NOT NULL

);

DROP TABLE IF EXISTS Trainer CASCADE;

CREATE TABLE IF NOT EXISTS Trainer (

trainerId SERIAL PRIMARY KEY NOT NULL,

dateOfStartExploitation DATE NOT NULL,

trainerTypeId INT NOT NULL REFERENCES Trainer\_Type (trainerTypeId)

);

DROP TABLE IF EXISTS Strength\_Exercise\_Type CASCADE;

CREATE TABLE IF NOT EXISTS Strength\_Exercise\_Type (

strengthExerciseTypeId SERIAL PRIMARY KEY NOT NULL,

name VARCHAR (100) NOT NULL

);

INSERT INTO Strength\_Exercise\_Type (name) VALUES

('Тяга верхнего блока');

DROP TABLE IF EXISTS Strength\_Exercise CASCADE;

CREATE TABLE IF NOT EXISTS Strength\_Exercise (

strengthExerciseId SERIAL PRIMARY KEY NOT NULL,

dateTimeOfStart TIMESTAMP NOT NULL,

dateTimeOfFinish TIMESTAMP NULL,

trainingId INT NOT NULL REFERENCES Training (trainingId),

strengthExerciseTypeId INT NOT NULL REFERENCES Strength\_Exercise\_Type (strengthExerciseTypeId),

trainerId INT NOT NULL REFERENCES Trainer (trainerId)

);

DROP TABLE IF EXISTS Approach CASCADE;

CREATE TABLE IF NOT EXISTS Approach (

approachId SERIAL PRIMARY KEY NOT NULL,

dateTimeOfStart TIMESTAMP NOT NULL,

dateTimeOfFinish TIMESTAMP NULL,

weight FLOAT (1) NOT NULL,

repetition INT NOT NULL,

strengthExerciseId INT NOT NULL REFERENCES Strength\_Exercise (strengthExerciseId)

);

DROP TABLE IF EXISTS Running\_Exercise\_Type CASCADE;

CREATE TABLE IF NOT EXISTS Running\_Exercise\_Type (

runningExerciseTypeId SERIAL PRIMARY KEY NOT NULL,

name VARCHAR (100) NOT NULL

);

INSERT INTO Running\_Exercise\_Type (name) VALUES

('Беговая дорожка');

DROP TABLE IF EXISTS Running\_Exercise CASCADE;

CREATE TABLE IF NOT EXISTS Running\_Exercise (

runningExerciseId SERIAL PRIMARY KEY NOT NULL,

dateTimeOfStart TIMESTAMP NOT NULL,

dateTimeOfFinish TIMESTAMP NOT NULL,

distance FLOAT (1) NOT NULL,

time FLOAT (1) NOT NULL,

averageSpeed FLOAT (1) NOT NULL,

averagePulse INT CHECK (0 <= averagePulse AND averagePulse <= 300) NULL,

minPulse INT CHECK (0 <= minPulse AND minPulse <= 300) NULL,

maxPulse INT CHECK (0 <= maxPulse AND maxPulse <= 300) NULL,

runningExerciseTypeId INT NOT NULL REFERENCES Running\_Exercise\_Type (runningExerciseTypeId),

trainingId INT NOT NULL REFERENCES Training (trainingId),

trainerId INT NOT NULL REFERENCES Trainer (trainerId)

)

Листинг 12. Функции для взаимодействия с СУБД PostgreSQL.

def get\_db\_connection():

conn = psycopg2.connect(\*\*db\_config)

return conn

def execute\_select\_one\_query(query: str):

with get\_db\_connection().cursor() as cursor:

cursor.execute(query)

entry = cursor.fetchone()

return entry

def execute\_select\_all\_query(query: str):

with get\_db\_connection().cursor() as cursor:

cursor.execute(query)

entry = cursor.fetchall()

return entry

def insert(table\_name: str, entry: dict):

column\_names = ', '.join([k for k in entry.keys()])

values = ', '.join('%s' for \_ in entry.keys())

query = 'INSERT INTO {}({}) VALUES ({})'.format(table\_name, column\_names, values)

conn = get\_db\_connection()

try:

with conn.cursor() as cursor:

cursor.execute(query, list(entry.values()))

conn.commit()

except Exception as e:

conn.rollback()

print(e)

return False

return True

def insert\_strengthExercise(entry: dict):

column\_names = ', '.join([k for k in entry.keys()])

values = ', '.join('%s' for \_ in entry.keys())

query = 'INSERT INTO strength\_exercise ({}) VALUES ({}) RETURNING strengthexerciseid'.format(column\_names, values)

conn = get\_db\_connection()

try:

entry

with conn.cursor() as cursor:

cursor.execute(query, list(entry.values()))

entry = cursor.fetchone()

conn.commit()

return entry

except Exception as e:

conn.rollback()

print(e)

return False

def get\_all\_trainings(customer\_id: int):

return execute\_select\_all\_query('SELECT \* FROM training WHERE customerid={}'.format(customer\_id))

def user\_exists(email, phoneNumber: str):

user = 'SELECT \* FROM Customer WHERE email = \'{}\' OR phonenumber = \'{}\''.format(email, phoneNumber)

return not execute\_select\_one\_query(user) is None

def get\_consumer\_by\_email(email: str):

return execute\_select\_one\_query('SELECT \* FROM Customer WHERE email = \'{}\''.format(email))

def get\_by\_unique\_int(table: str, by: str, value: int) -> Union[Tuple, None]:

return execute\_select\_one\_query('SELECT \* FROM {} WHERE {}={}'.format(table, by, value))

def get\_all\_running\_exercise(training\_id: int):

return execute\_select\_all\_query('SELECT \* FROM running\_exercise WHERE trainingid={}'.format(training\_id))

def get\_last\_training(customerId: int):

return execute\_select\_one\_query('SELECT MAX (trainingId) FROM training WHERE customerid={}'.format(customerId))[0]

def complete\_training(trainingId: int, datetime\_of\_finish: str):

query = 'UPDATE training SET datetimeoffinish=\'{}\' WHERE trainingid={} RETURNING \*'.format(datetime\_of\_finish, trainingId)

conn = get\_db\_connection()

try:

with conn.cursor() as cursor:

cursor.execute(query)

conn.commit()

except Exception as e:

conn.rollback()

print(e)

return False

return True