МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: ИС Тренажерный зал

Студент гр. 1303 _____ Насонов Я.К.

Студент гр. 1303 _____ Иванов А.С.

Студент гр. 1303 _____ Попандопуло А.Г.

Преподаватель _____ Заславский М.М.

Санкт-Петербург,
2024

ЗАДАНИЕ

Студент Насонов Я.К.

Студент Иванов А.С.

Студент Попандопуло А.Г.

Группа 1303

Тема: ИС Тренажерный зал

Исходные данные:

Необходимо реализовать веб-приложение для цифровизации работы с абонементами и процессов взаимодействия клиентов с тренерами.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Выводы»

«Приложения»

«Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 05.09.2024

Дата сдачи реферата: 20.12.2024

Дата защиты реферата: 20.12.2024

Студент	_____	Насонов Я.К.
Студент	_____	Иванов А.С.
Студент	_____	Попандопуло А.Г.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В рамках проекта было разработано веб-приложение для тренажерного зала, предоставляющее пользователям удобный доступ к основным услугам. Приложение сочетает возможности как для клиентов, так и для тренеров: клиентам предоставлена возможность приобретать, продлевать и замораживать абонементы, записываться на занятия, создаваемые тренерами; помимо создания занятий, тренера имеют возможность просматривать статистику о проведенных занятиях и информацию о записавшихся клиентах. Реализована система поиска и фильтрации для просмотра пользователей, статистики и информации о занятиях.

Серверная составляющая реализована с использованием Spring Framework, клиентская – Vue.js. в качестве СУБД использована MongoDB.

SUMMARY

The project involved developing a web application for a gym, providing users with convenient access to basic services. The application combines features for both clients and trainers: clients are given the opportunity to purchase, renew and freeze memberships, sign up for classes created by trainers; in addition to creating classes, trainers have the opportunity to view statistics on completed classes and information on registered clients. A search and filter system has been implemented to view users, statistics and information on classes.

The server component is implemented using the Spring Framework, the client component is Vue.js. MongoDB is used as a DBMS.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Актуальность проблемы

Разработка веб-приложений для спортивных залов становится всё более востребованной в условиях цифровизации и растущего интереса к здоровому образу жизни. Современные пользователи, как клиенты так и тренеры, ожидают удобного и быстрого доступа к услугам через интернет, включая возможность приобретения абонементов, записи на тренировки и управления своим расписанием в режиме онлайн.

Веб-приложение для спортивного зала решает задачи автоматизации записи и учета клиентов, снижает нагрузку на администраторов, улучшает пользовательский опыт и повышает лояльность посетителей.

1.2. Постановка задачи

Задача проекта заключается в разработке веб-приложения, которое позволяет клиентам:

- Просматривать расписание занятий и записываться на интересующие
- Фильтровать занятия по различным параметрам;
- Управлять своим абонементом (продление, заморозка)

И позволяет тренерам:

- Управлять своим расписанием, создавая и изменяя занятия
- Просматривать контактную информацию о своих клиентах
- Просматривать статистику о проведенных занятиях

1.3. Предлагаемое решение

Создание веб-приложения с использованием Vue.js для клиентской части вкупе со Spring Framework для серверной составляющей и MongoDB для хранения данных.

1.4. Качественные требования к решению

Решение должно обладать интуитивно понятным пользовательским интерфейсом, быть надежным и расширяемым .

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет UI

Ниже представлены макеты страниц приложения.

Рисунок 1. Информационная страница и форма входа

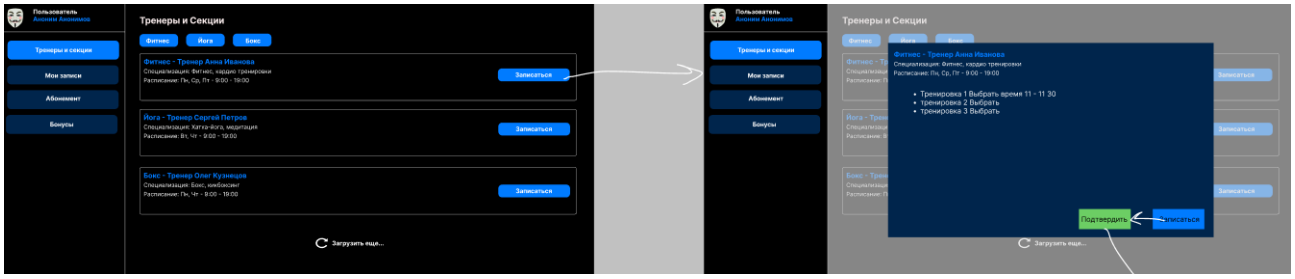

Рисунок 2. Просмотр занятий и запись

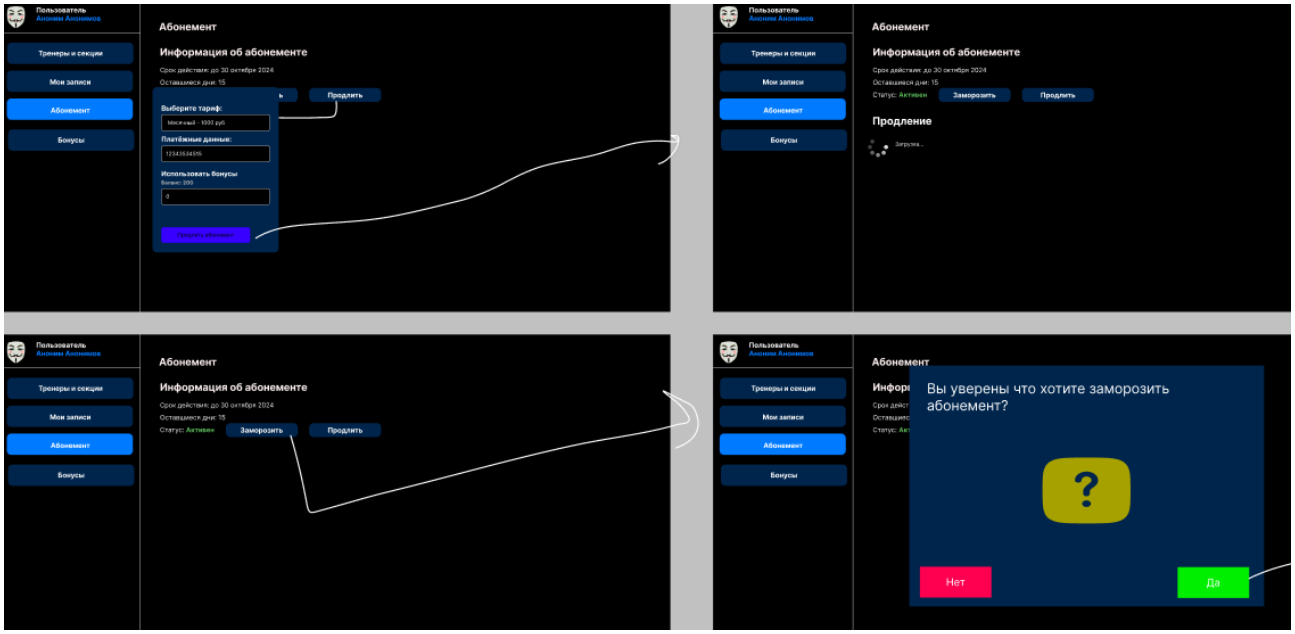

Рисунок 3. Продление и заморозка абонемента

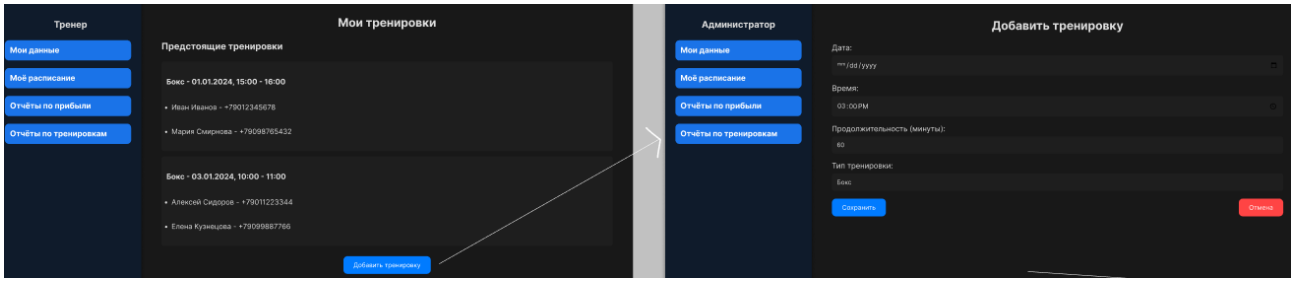

Рисунок 4. Просмотр расписания тренера и добавление тренировки

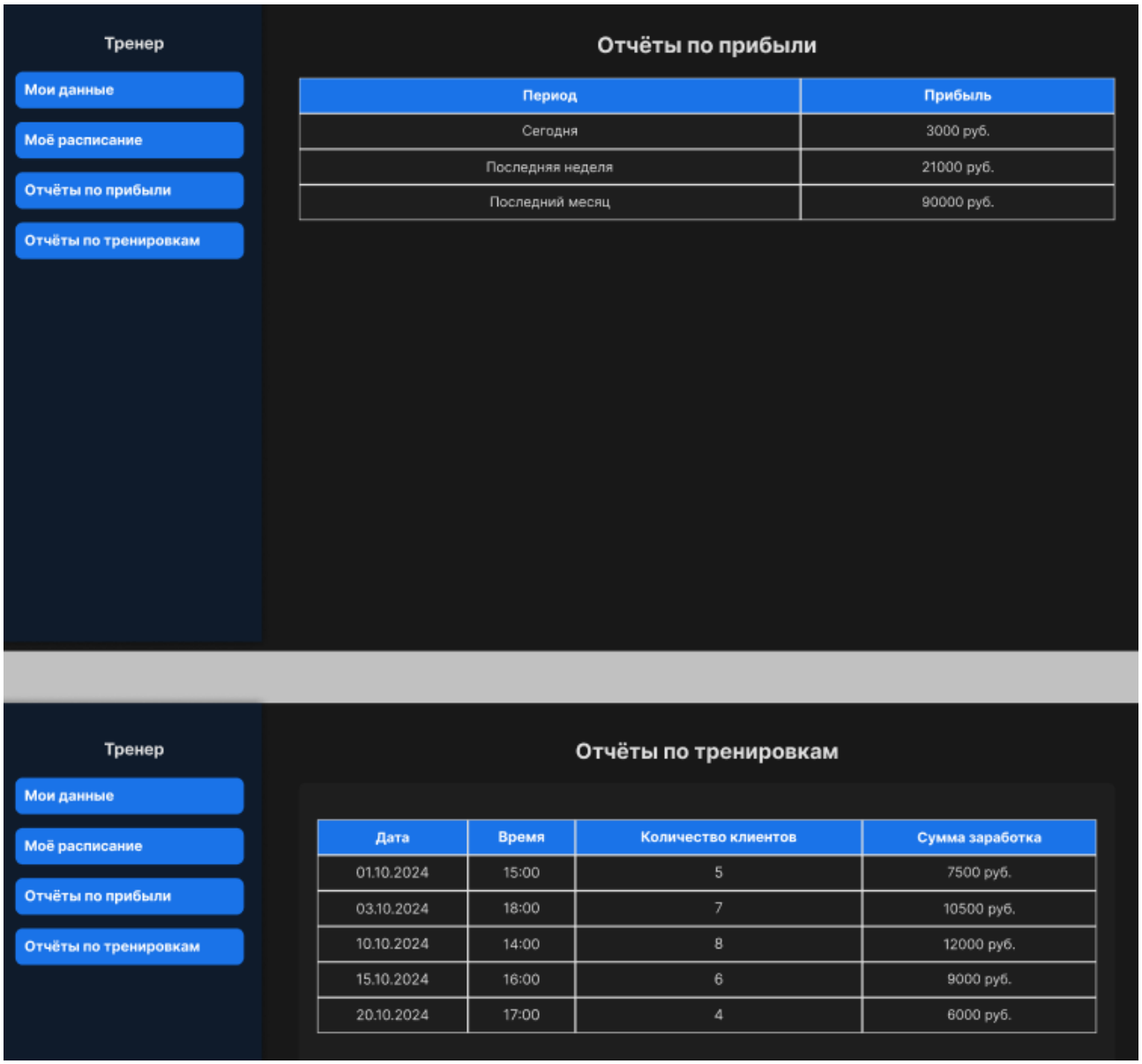

Рисунок 5. Просмотр статистики тренера

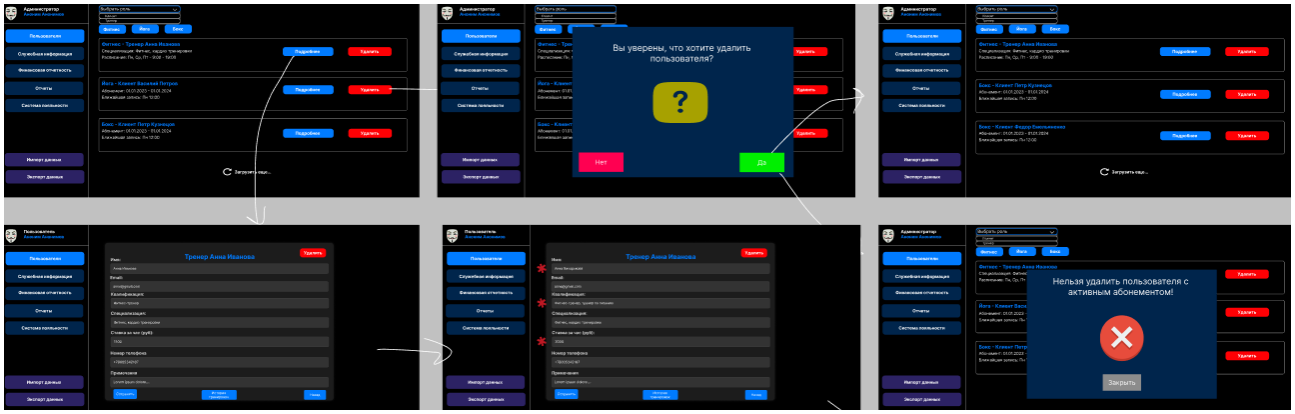

Рисунок 6. Редактирование пользователей администратором

Рисунок 7. Просмотр статистики администратором

2.2. Сценарий использования для импорта данных

а) Сценарий использования: Импорт данных

- **Действующее лицо:** Администратор
- **Предусловие:** Администратор авторизован в системе.

Основной сценарий:

1. Администратор открывает веб-приложение и авторизуется.
2. В разделе администрирования выбирает опцию "Импорт данных".
3. Приложение проверяет структуру и формат данных (json).
4. Приложение загружает данные в базу данных.

5. Приложение уведомляет администратора об успешном завершении импорта.

Альтернативный сценарий:

- Если файл имеет неправильный формат
- Система не смогла валидировать загруженный файл:
    i. Приложение выводит сообщение об ошибке, указывая на неверный формат, либо ошибку валидации.
    ii. Пользователь имеет возможность загрузить другой файл

Результат: Данные успешно импортированы в базу данных и доступны для работы в системе.

б)

1) Сценарий использования: Управление расписанием

- **Действующее лицо:** Тренер
- **Предусловие:** Тренер авторизован в системе.

Основной сценарий:

1. Тренер заходит в раздел "Моё расписание".
2. Приложение отображает текущее расписание тренера.
3. Тренер выбирает опцию "Добавить время для тренировок".
4. Приложение предлагает выбрать дату, время и продолжительность тренировки.
5. Тренер вводит данные и нажимает "Сохранить".
6. Приложение обновляет его расписание и отображает добавленные занятия.
7. Тренер может удалять или изменять доступное время для тренировок.

Альтернативный сценарий:

- Тренер пытается добавить занятие на время, которое пересекается с уже существующим.
    - Приложение уведомляет о конфликте времени и предлагает выбрать другое время.

Результат: Расписание тренера обновлено, и доступное время для тренировок добавлено в систему.

2) Сценарий использования: Запись на тренировку

- **Действующее лицо:** Клиент
- **Предусловие:** Клиент авторизован в системе и у него есть активный абонемент.

Основной сценарий:

1. Клиент заходит в раздел "Тренеры и секции" и выбирает тренера или секцию.
2. Приложение отображает расписание тренера с доступными временными слотами.
3. Клиент выбирает свободный слот и нажимает "Записаться".
4. Приложение проверяет наличие свободного места на тренировку.
5. Приложение подтверждает запись клиента на тренировку и отображает уведомление об успешной записи.
6. Клиент получает уведомление о записи на занятие

Альтернативный сценарий:

- Все места на выбранную тренировку заняты.
    - Приложение уведомляет клиента об отсутствии свободных мест.

- **Результат:** Клиент успешно записан на тренировку.

2.3. Сценарий использования для представления данных

1) Сценарий использования: Просмотр доступных тренеров и секций

- **Действующее лицо:** Клиент
- **Предусловие:** Клиент авторизован в системе.

Основной сценарий:

1. Клиент заходит в раздел "Тренеры и секции" на странице профиля
2. Приложение отображает список доступных тренеров
3. Клиент имеет возможность фильтрации списка по имени, квалификации и секции
4. Клиент просматривает расписание выбранного тренера или секции.

Альтернативный сценарий:

- В системе нет доступных тренеров или секций.
    - Приложение уведомляет клиента, что на данный момент нет доступных тренеров или секций.

Результат: Клиент получает информацию о доступных тренерах и секциях.

2) Сценарий использования: Просмотр записей на занятия

- **Действующее лицо:** Клиент
- **Предусловие:** Клиент авторизован в системе.

Основной сценарий:

1. Клиент заходит в раздел "Мои записи" на главной странице.
2. Приложение отображает список всех предстоящих записей на тренировки.

3. Клиент имеет возможность фильтрации записей по временному диапазону, тренерам, залам и секциям.
4. Клиент просматривает информацию занятиях (дата, время, тренер).

Результат: Клиент получает информацию о своих записях на тренировки.

3) Сценарий использования: Просмотр списка записанных клиентов на свои тренировки

- **Действующее лицо:** Тренер
- **Предусловие:** Тренер авторизован в системе и у него есть запланированные тренировки.

Основной сценарий:

1. Тренер заходит в раздел "Мои тренировки".
2. Приложение отображает список предстоящих тренировок.
3. Тренер выбирает конкретную тренировку, чтобы просмотреть список клиентов, записанных на неё.
4. Приложение отображает список клиентов с их именами и контактной информацией (если доступно).
5. Тренер просматривает список и может подготовиться к тренировке.

Результат: Тренер видит список клиентов, записанных на его тренировки.

4) Сценарий использования: Просмотр отчётности по проведённым тренировкам

- **Действующее лицо:** Тренер
- **Предусловие:** Тренер авторизован в системе.

Основной сценарий:

1. Тренер заходит в раздел "Отчёты".

2. Приложение отображает список проведённых тренировок за определённый период.
3. Тренер может выбрать период для фильтрации (например, за последнюю неделю, месяц).
4. Приложение отображает список тренировок с датой, временем, количеством клиентов, и суммой заработка за каждую тренировку.
5. Тренер просматривает отчёт по проведённым тренировкам.

Результат: Тренер получает отчёт о своих тренировках за выбранный период.

5) Сценарий использования: Полный доступ к информации о клиентах и тренерах

- **Действующее лицо:** Администратор
- **Предусловие:** Администратор авторизован в системе.

Основной сценарий:

1. Администратор заходит в раздел "Пользователи".
2. Приложение отображает список всех клиентов и тренеров.
3. Администратор выбирает конкретного пользователя (клиента или тренера) для просмотра его профиля.
4. Приложение отображает полную информацию о выбранном пользователе (личные данные, записи на тренировки, история абонементов и т.д.).
5. Администратор просматривает данные.

Результат: Администратор получает доступ к полной информации о клиентах и тренерах.

6) Сценарий использования: Просмотр служебной информации (информация о залах)

- **Действующее лицо:** Администратор

- **Предусловие:** Администратор авторизован в системе.

Основной сценарий:

1. Администратор заходит в раздел "Служебная информация".
2. Приложение отображает данные о залах, их тренерах и секциях
3. Администратор может фильтровать залы по рабочим дням, секциям и тренерам
4. Администратор просматривает данные.

Результат: Администратор получает информацию о залах

2.4. Сценарий использования для анализа данных

1) Сценарий использования: Просмотр своей прибыли за выбранный период

- **Действующее лицо:** Тренер
- **Предусловие:** Тренер авторизован в системе.

Основной сценарий:

1. Тренер заходит в раздел "Отчёты".
2. Приложение отображает сумму прибыли тренера за выбранные период на основе его часовой ставки и проведённых тренировок.
3. Тренер просматривает отчёт.

Результат: Тренер видит свою прибыль за период.

2) Сценарий использования: Ведение статистики по абонементам и занятиям

- **Действующее лицо:** Администратор
- **Предусловие:** Администратор авторизован в системе.

Основной сценарий:

1. Администратор заходит в раздел "Отчёты".
2. Приложение предлагает выбрать отчёт по абонементам или по статистике занятий.
3. Для отчёта по абонементам приложение отображает список всех активных, завершённых или истекающих абонементов.
4. Администратор может фильтровать абонементы по клиентам, дате окончания или статусу.
5. Для отчёта по занятиям приложение отображает количество проведённых тренировок, их тип (йога, бокс и т.д.), количество клиентов на каждой тренировке и рассчитывает ее посещаемость.
6. Администратор просматривает и/или экспортирует данные.

Результат: Администратор видит отчёты по абонементам и статистике занятий.

2.5. Сценарий использования для экспорта данных

Сценарий использования: Массовый экспорт данных из БД

- **Действующее лицо:** Администратор
- **Предусловие:** Администратор авторизован в системе.

Основной сценарий:

1. Администратор открывает веб-приложение и авторизуется.
2. В разделе администрирования выбирает опцию "Экспорт данных".
3. Приложение генерирует файл json с экспортированными данными.
4. Администратор загружает файл на своё устройство.

Альтернативный сценарий:

- Если возникают проблемы с доступом к данным или с генерацией файла:

i. Приложение выводит сообщение об ошибке.

Результат: Данные экспортированы из базы данных и сохранены на устройстве администратора.

2.6. Вывод

Для нашего решения операции чтения несколько преобладают над записью, так как запись в целом выполняется со стороны клиентов и тренеров, они же имеют возможность просмотра данных в виде списков и таблиц. Вместе с тем, просмотр статистики выполняется администратором.

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель данных

Графическое представление модели

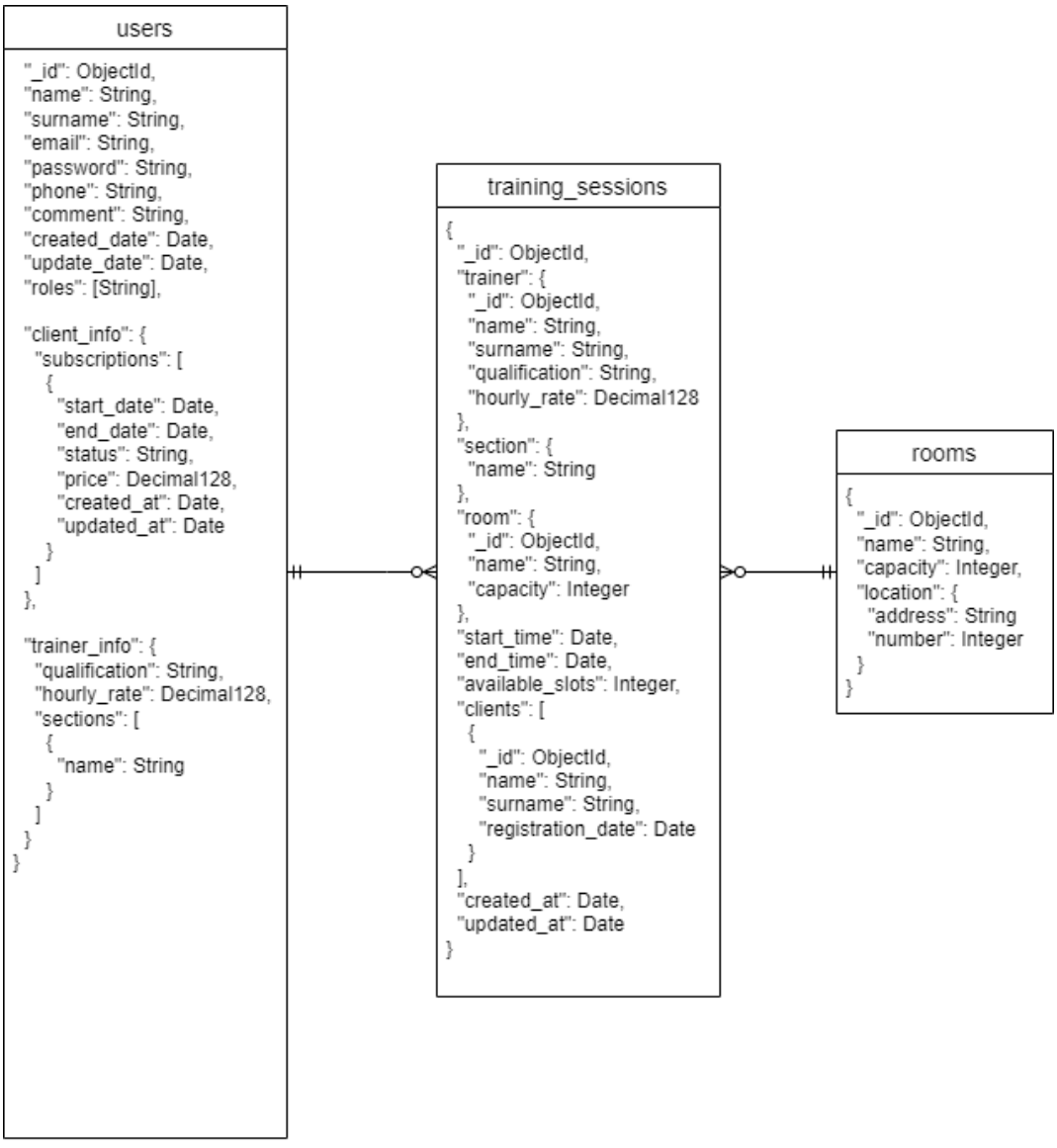

Рисунок 8. Нереляционная модель данных

Описание назначений коллекций, типов данных и сущностей:

1. users (пользователи):

- **_id**: ObjectId — уникальный идентификатор пользователя.
- **name** и **surname**: String — имя и фамилия пользователя.
- **email**: String — электронная почта.

- **password**: String — пароль пользователя.
- **phone**: String — номер телефона.
- **comment**: String — дополнительные комментарии.
- **created_date** и **updated_date**: Date — даты создания и обновления записи.
- **roles**: [String] — список ролей пользователя.
- **client_info**: Информация о клиентах:
    - **subscriptions**: [Object] — информация о подписках:
        - **start_date**, **end_date**, **status**, **price**, **created_at**, **updated_at**: Date и Decimal128 — детали подписки.
- **trainer_info**: Информация о тренере:
    - **qualification**, **hourly_rate**: String и Decimal128 — квалификация и ставка.
    - **sections**: [Object] — секции, которые тренер ведет:
        - **name**: String — название секции.

2. training_sessions (тренировочные сессии):

- **_id**: ObjectId — уникальный идентификатор сессии.
- **trainer**: Object — информация о тренере:
    - **_id**, **name**, **surname**, **qualification**, **hourly_rate**: ObjectId, String, Decimal128.
- **section**: Object — название секции:
    - **name**: String.
- **room**: Object — информация о зале:
    - **_id**, **name**, **capacity**: ObjectId, String, Integer.
- **start_time**, **end_time**: Date — время начала и окончания сессии.
- **available_slots**: Integer — доступные места.
- **clients**: [Object] — список клиентов:
    - **_id**, **name**, **surname**, **registration_date**: ObjectId, String, Integer, Date.
- **created_at**, **updated_at**: Date — даты создания и обновления записи.

3. rooms (залы):

- **_id**: ObjectId — уникальный идентификатор зала.

- **name**: String — название.
- **capacity**: Integer — вместимость.
- **location**: Object — местоположение:
    - **address**, **number**: String, Integer.

Оценка объема информации, хранимой в модели:

- ObjectId занимает 12 байт.
- Средний пользователь имеет 3 подписки.
- Средний пользователь записывается на 8 тренировок.
- Доля тренеров от числа пользователей - 10%
- В среднем тренер ведет 3 секции
- В среднем тренер проводит 12 тренировок
- Среднее число залов - 10
- Среднее число акций - 5
- Среднее число транзакций по очкам лояльности у пользователя - 8

users:

- В среднем у пользователя 1-2 роли `12 + 100 + 100 + 255 + 255 + 20 + 255 + 8 + 8 + 8`
- client_info: `4 + 3 * (8 + 8 + 20 + 16 + 8 + 8) + 8 * (4 + 20 + 255 + 8)`
- trainer_info: `100 + 16 + 3 * 50`
- Объем информации для клиента: `1021 + 208 = 2504 байт`
- Объем информации для тренера: `1021 + 266 = 1287 байт`

rooms:

- `12 + 100 + 4 + 255 + 4 = 375 байт` на 1 зал

training_sessions:

- trainer: `12 + 100 + 100 + 100 + 16 = 328`
- section: `50`
- room: `12 + 100 + 4 = 116 8 + 8 + 4 + 8 * (12 + 100 + 100 + 4 + 8) + 8 + 8 = 1828`
- Итого для одной записи тренировки `328 + 50 + 116 + 1828 = 2322 байт`

Примеры запросов для совершения сценариев использования

1) Поиск тренера по квалификации и секции

```
db.users.find({
  roles: "trainer",
  "trainer_info.qualification": "Certified Cardio Specialist",
  "trainer_info.sections.name": "Cardio Training"
});
```

2) Добавление нового абонемента

```
db.users.updateOne(
  { _id: ObjectId("653ef3a8a3e34567bcdf5671") }, // ID клиента
  {
    $push: {
      "client_info.subscriptions": {
        start_date: new Date("2024-11-01"),
        end_date: new Date("2025-11-01"),
        status: "ACTIVE",
        price: NumberDecimal("99.99"),
        created_at: new Date(),
        updated_at: new Date()
      }
    }
  }
);
```

3) Просмотр активных абонементов клиента

```
db.users.findOne(
  { _id: ObjectId("653ef3a8a3e34567bcdf1234") }, // ID клиента
  {
    "client_info.subscriptions": {
      $elemMatch: {
        status: "ACTIVE",
        end_date: { $gte: new Date() } // Проверяем, что абонемент еще активен
      }
    }
  }
);
```

4) Создание новой тренировки

```
db.training_sessions.insertOne({
  trainer: {
    _id: ObjectId("653ef3a8a3e34567bcdf1234"),
    name: "John",
    surname: "Doe",
    qualification: "Certified Personal Trainer",
    hourly_rate: NumberDecimal("50.00")
  },
  section: {
    _id: ObjectId("653ef3a8a3e34567bcdf4321"),
    name: "Strength Training"
  },
  room: {
    _id: ObjectId("653ef3a8a3e34567bcdf5678"),
    name: "Main Gym Hall",
```

```
    capacity: 20
  },
  start_time: new Date("2024-11-10T09:00:00Z"),
  end_time: new Date("2024-11-10T10:30:00Z"),
  available_slots: 5,
  clients: [],
  created_at: new Date(),
  updated_at: new Date()
});
```

5) Запись клиента на тренировку

```
db.training_sessions.updateOne(
  { _id: ObjectId("653ef3a8a3e34567bcdf6789") },
  { $push: { clients: {
      _id: ObjectId("653ef3a8a3e34567bcdf5671"),
      name: "Jane",
      surname: "Smith",
      loyalty_points: 200
    }},
    $inc: { available_slots: -1 } // Уменьшаем доступные места
  }
);
```

6) Просмотр клиентов записанных на тренировку:

```
db.training_sessions.findOne(
  { _id: ObjectId("653ef3a8a3e34567bcdf6789") },
  { clients: 1 }
);
```

7) Поиск тренировок по секции и дате

```
db.training_sessions.find({
  "section.name": "Strength Training",
  start_time: {
    $gte: new Date("2024-11-01T00:00:00Z"),
    $lt: new Date("2024-12-01T00:00:00Z")
  }
});
```

8) Поиск тренеров, у которых загрузка занятий в среднем меньше N процентов

- Для выполнения такого запроса небходимо воспользоваться Aggregation Framework

```
db.training_sessions.aggregate([
  {
    $project: {
      "trainer_id": "$trainer._id",
      "trainer_name": "$trainer.name",
      "trainer_surname": "$trainer.surname",
      "available_slots": "$available_slots",
      "clients_count": { $size: "$clients" }
    }
```

```
  },
  {
    $group: {
      _id: {
        trainer_id: "$trainer_id",
        trainer_name: "$trainer_name",
        trainer_surname: "$trainer_surname"
      },
      average_load_percentage: {
        $avg: {
          $cond: [
            { $gt: ["$available_slots", 0] },
            { $multiply: [{ $divide: ["$clients_count",
"$available_slots"] }, 100] },
            0
          ]
        }
      }
    }
  },
  {
    $match: {
      average_load_percentage: { $lt: N }  // N - процент нагрузки
тренера
    }
  },
  {
    $project: {
      trainer_id: "$_id.trainer_id",
      trainer_name: "$_id.trainer_name",
      trainer_surname: "$_id.trainer_surname",
      average_load_percentage: 1
    }
  }
])
```

9) Поиск N самых загруженных тренеров

- Для подобного запроса также применим Aggregation Framework:

```
db.training_sessions.aggregate([
  {
    $project: {
      "trainer_id": "$trainer._id",
      "trainer_name": "$trainer.name",
      "trainer_surname": "$trainer.surname",
      "available_slots": "$available_slots",
      "clients_count": { $size: "$clients" }
    }
  },
  {
    $group: {
      _id: {
```

```
        trainer_id: "$trainer_id",
        trainer_name: "$trainer_name",
        trainer_surname: "$trainer_surname"
      },
      max_load_percentage: {
        $max: {
          $multiply: [
            { $cond: [
              { $gt: ["$available_slots", 0] },
              { $divide: ["$clients_count", "$available_slots"] },
              0
            ]},
            100
          ]
        }
      }
    }
  },
  {
    $sort: { max_load_percentage: -1 }
  },
  {
    $limit: N  // N - искомое количество тренеров
  },
  {
    $project: {
      trainer_id: "$_id.trainer_id",
      trainer_name: "$_id.trainer_name",
      trainer_surname: "$_id.trainer_surname",
      max_load_percentage: 1
    }
  }
])
```

3.2. Аналог модели данных для SQL СУБД

Графическое представление модели

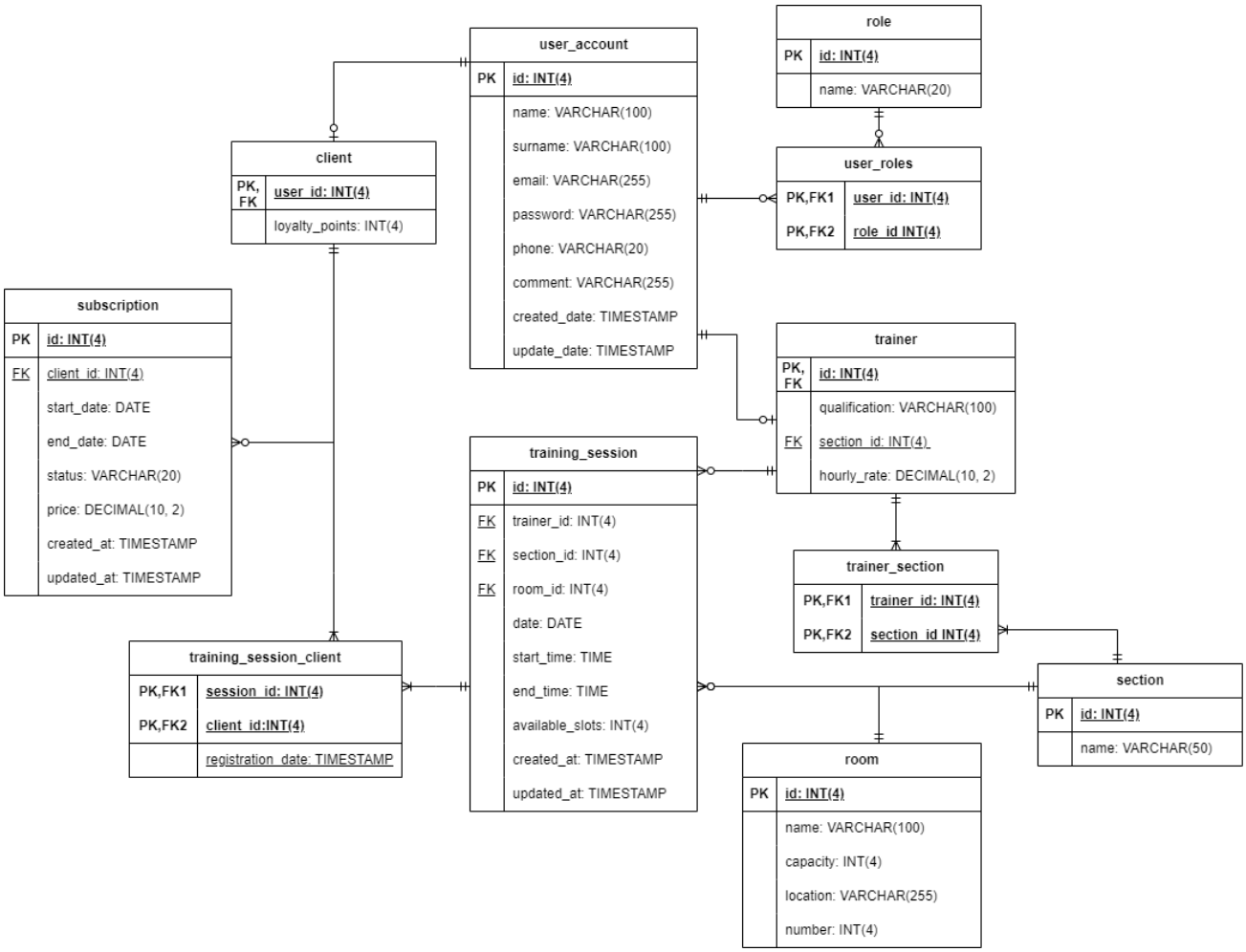

Рисунок 9. Реляционная модель данных

Описание данных сущностей:

1. user_account

- **id**: Уникальный идентификатор пользователя.
- **name**: Имя пользователя.
- **surname**: Фамилия пользователя.
- **email**: Электронная почта пользователя.
- **password**: Хэшированный пароль.
- **phone**: Номер телефона пользователя.
- **comment**: Дополнительные комментарии о пользователе.

- **created_date**: Дата и время создания записи.
- **update_date**: Дата и время последнего обновления записи.

2. role

- **id**: Уникальный идентификатор роли.
- **name**: Название роли (например, "trainer", "admin").

3. user_roles

- **user_id**: Идентификатор пользователя из user_account.
- **role_id**: Идентификатор роли из role.

4. client

- **user_id**: Идентификатор пользователя из user_account.
- **loyalty_points**: Текущее количество очков лояльности клиента.

5. trainer

- **user_id**: Идентификатор пользователя из user_account.
- **qualification**: Квалификация тренера.
- **section_id**: Идентификатор секции из section.
- **hourly_rate**: Почасовая ставка тренера.

6. section

- **id**: Уникальный идентификатор секции.
- **name**: Название секции.

7. trainer_section

- **trainer_id**: Идентификатор тренера из trainer.
- **section_id**: Идентификатор секции из section.

8. room

- **id**: Уникальный идентификатор зала.
- **name**: Название зала.
- **capacity**: Вместимость зала.
- **location**: Адрес или описание местоположения.
- **number**: Номер помещения.

9. subscription

- **id**: Уникальный идентификатор абонемента.

- **client_id**: Идентификатор клиента из client.
- **start_date**: Дата начала действия абонемента.
- **end_date**: Дата окончания действия абонемента.
- **status**: Состояние абонемента (ACTIVE, EXPIRED, FROZEN).
- **price**: Стоимость абонемента.
- **created_at**: Дата создания записи.
- **updated_at**: Дата последнего обновления записи.

10. training_session

- **id**: Уникальный идентификатор тренировки.
- **trainer_id**: Идентификатор тренера из trainer.
- **section_id**: Идентификатор секции из section.
- **room_id**: Идентификатор зала из room.
- **date**: Дата проведения тренировки.
- **start_time**: Время начала тренировки.
- **end_time**: Время окончания тренировки.
- **available_slots**: Количество доступных мест.
- **created_at**: Дата создания записи.
- **updated_at**: Дата последнего обновления записи.

11. training_session_client

- **session_id**: Идентификатор тренировки из training_session.
- **client_id**: Идентификатор клиента из client.
- **registration_date**: Дата регистрации клиента на тренировку.

Оценка объема информации, хранимой в модели

- Приведем максимальные объемы записей для каждой таблицы:

user_account

- `4 + 100 + 100 + 255 + 255 + 20 + 255 + 8 + 8 = 1005 байт`

role

- `4 + 20 = 24 байт`

user_roles

- `4 + 4 = 8 байт`

client

- `4 + 4 = 8 байт`

trainer

- `4 + 100 + 4 + 10 = 118 байт`

section

- `4 + 50 = 54 байта`

trainer_section

- `4 + 4 = 8 байт`

room

- `4 + 100 + 4 + 255 + 4 = 367 байт`

subscription

- `4 + 4 + 4 + 4 + 20 + 10 + 8 + 8 = 62 байта`

training_session

- `4 + 4 + 4 + 4 + 4 + 8 + 8 + 4 + 8 + 8 = 56 байт`

training_session_client

- `4 + 4 + 4 = 12 байт`

promotion

- `4 + 100 + 255 + 4 + 4 + 6 + 4 + 8 + 8 = 393 байт`

loyalty_points_history

- `4 + 4 + 4 + 20 + 255 + 8 = 295 байт`

Положим, что:

- Средний пользователь имеет 3 подписки.
- Средний пользователь записывается на 8 тренировок.
- Среднее количество секций - 5
- Доля тренеров от числа пользователей - 10%
- В среднем тренер ведет 3 секции
- В среднем тренер проводит 12 тренировок
- Среднее число залов - 10

- Среднее число акций - 5
- Среднее число транзакций по очкам лояльности: 8

При количестве пользователей, равном **u**:

Формула общего объема данных:

$$V(u)=(1005+62\cdot3+8\cdot(12+295))\cdot u+24\cdot3+54\cdot5+(3\cdot8+12\cdot56)\cdot0.1u+10\cdot367+5\cdot393$$

$$V(u)=1255u+72+270+101.6u+3630+2005$$

$$V(u)=3716.6u+5977$$

Примеры запросов:

1) Поиск тренера по квалификации и секции

```
SELECT t.user_id, u.name, u.surname, t.qualification, s.name AS
section, t.hourly_rate
  FROM trainer t
       JOIN user_account u ON t.user_id = u.id
       JOIN section s ON t.section_id = s.id
 WHERE t.qualification LIKE @qualification
   AND s.name = @section;
```

2) Добавление нового абонемента

```
INSERT INTO subscription (client_id, start_date, end_date, status,
price, created_at, updated_at)
     VALUES (@client_id, @start_date, @end_date, @status, @price,
NOW(), NOW());
```

3) Просмотр активных абонементов клиента

```
SELECT *
  FROM subscription
 WHERE client_id = @client_id
   AND status = 'ACTIVE';
```

4) Создание новой тренировки

```
INSERT INTO training_session (trainer_id, section_id, room_id,
date, start_time, end_time, available_slots, created_at,
updated_at)
     VALUES (@trainer_id, @section_id, @room_id, @date,
@start_time, @end_time, @available_slots, NOW(), NOW());
```

5) Запись клиента на тренировку

```
INSERT INTO training_session_client (session_id, client_id)
     VALUES (@session_id, @client_id);
```

6) Просмотр клиентов, записанных на тренировку

```
SELECT c.user_id, u.name, u.surname
```

```
  FROM training_session_client tsc
       JOIN client c ON tsc.client_id = c.user_id
       JOIN user_account u ON c.user_id = u.id
 WHERE tsc.session_id = @session_id;
```

7) Поиск тренировок по секции и дате

```
SELECT ts.id, ts.date, ts.start_time, ts.end_time,
ts.available_slots, r.name AS room
  FROM training_session ts
       JOIN section s ON ts.section_id = s.id
       JOIN room r ON ts.room_id = r.id
 WHERE s.name = @section
   AND ts.date = @date;
```

8) Поиск тренеров, у которых загрузка занятий в среднем меньше N процентов

```
SELECT t.user_id AS trainer_id,
       u.name, u.surname,
       AVG(CASE WHEN ts.available_slots > 0
                THEN (COUNT(tc.client_id) / ts.available_slots) *
100
                ELSE 0 END) AS average_load_percentage
FROM trainer t
JOIN user_account u ON t.user_id = u.id
JOIN training_session ts ON t.user_id = ts.trainer_id
LEFT JOIN training_session_client tc ON ts.id = tc.session_id
GROUP BY t.user_id, u.name, u.surname
HAVING AVG(CASE WHEN ts.available_slots > 0
                THEN (COUNT(tc.client_id) / ts.available_slots) *
100
                ELSE 0 END) < :N;
```

9) Поиск самых загруженных тренеров

```
SELECT t.user_id AS trainer_id, u.name, u.surname,
       MAX(CASE WHEN ts.available_slots > 0
                THEN (COUNT(tc.client_id) / ts.available_slots) *
100
                ELSE 0 END) AS max_load_percentage
FROM trainer t
JOIN user_account u ON t.user_id = u.id
JOIN training_session ts ON t.user_id = ts.trainer_id
LEFT JOIN training_session_client tc ON ts.id = tc.session_id
GROUP BY t.user_id, u.name, u.surname
ORDER BY max_load_percentage DESC
LIMIT @top_count;
```

3.3. Сравнение моделей

1) Удельный объем информации

- В реляционной модели данные распределены между различными таблицами, что позволяет минимизировать избыточность данных за счет нормализации. В MongoDB неоднократно применена денормализация, где данные копируются в разные коллекции для повышения производительности при чтении. Из-за этого:
- Реляционная модель экономичнее с точки зрения хранения, так как информация дублируется минимально. Например, роли пользователей, секции, в которых работают тренеры, и связи пользователей с тренировками хранятся в отдельных таблицах, и каждый объект записывается только один раз. Нереляционная модель имеет более высокий объем данных из-за дублирования вложенных объектов, таких как trainer и clients внутри training_sessions, а также sections в trainer_info и client_info. Это делает MongoDB модель более объемной, что было показано ранее

2) Запросы по отдельным юзкейсам

Для выполнения различных операций, рассмотрим несколько примеров:

Получение списка тренировок для клиента с указанием тренера, секции и зала

- Реляционная модель: требуется выполнить JOIN-запросы между таблицами training_session, training_session_client, trainer, room и section, чтобы собрать все нужные данные. Реляционная база данных хорошо оптимизирована для сложных выборок, и индексирование позволяет выполнять такие запросы достаточно быстро, но объем данных может повлиять на производительность.
- Нереляционная модель: благодаря денормализации, коллекция training_sessions уже содержит необходимые данные о trainer, section,

room, что позволяет обойтись одним запросом к этой коллекции. В MongoDB такие запросы выполняются быстрее, однако обновление данных может потребовать дополнительных действий (например, обновление информации о тренере для всех записей с его участием).

Добавление новой секции и назначение тренера

- Реляционная модель: необходимо создать новую запись в section, а также обновить таблицу trainer_section для связывания секции с тренером. Это занимает 2 запроса, но данные остаются централизованными и легко поддерживаются.
- Нереляционная модель: если секции тренеров хранятся в документе trainer_info, потребуется обновление каждого документа тренера при изменении секции. MongoDB поддерживает обновления, но синхронизация данных может быть сложной.

Поиск всех клиентов, записавшихся на тренировки в определенной секции

- Реляционная модель: сложный запрос с несколькими JOIN для связи таблиц training_session_client, training_session, и section.
- Нереляционная модель: можно выполнить запрос по коллекции training_sessions, где информация о клиентах, секциях и тренировках денормализована, что позволяет быстро получить все данные.

Количество задействованных коллекций

Юзкейс	**Реляционная модель (число таблиц)**	**Нереляционная модель (число коллекций)**
Список тренировок для клиента	5 (training_session, training_session_client, trainer, room, section)	1 (training_sessions)
Назначение тренера на секцию	2 (section, trainer_section)	1 (users)
Клиенты, записанные на тренировки секции	3 (training_session_client, training_session, section)	1 (training_sessions)

3) Вывод:

- MongoDB (нереляционная модель) удобна для чтения и простых выборок, требующих минимальных вложенных запросов. Она лучше подходит для сценариев, где изменения данных происходят не часто, а чтение — основная операция. Она требует больше памяти из-за денормализации данных, но может ускорить выборки.
- Реляционная база данных подходит для сложных и связанных данных, где важна согласованность и минимизация дублирования. Она требует больше времени на выполнение сложных запросов, но экономит память и упрощает обновление данных.

Проект имеет структуру с большим количеством вложенных данных (тренировки, секции, клиенты) и ориентирован на частое чтение с относительно редкими изменениями данных, и **нереляционная модель MongoDB представляется представляется подходящим выбором**, так как она позволит:

- Сократить количество запросов за счет вложенности данных. Например, получение всех записей о тренировках для клиента или информация о тренере и его секциях доступны в рамках одной коллекции (training_sessions), без необходимости многократных JOIN.
- Обеспечить гибкость в моделировании данных. В MongoDB проще вносить изменения в структуру данных по мере развития системы, добавляя или удаляя поля без строгих требований к схеме. Например, добавление новых свойств у пользователя или секции не требует миграций схемы, что ускоряет процесс разработки.
- Упростить масштабирование. MongoDB хорошо поддерживает горизонтальное масштабирование, что пригодится, если база данных будет расширяться, а проект разрастаться.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1 Краткое описание

Весь код приложения разделен на две части: серверную и клиентскую.

Серверная составляющая реализована с использованием фреймворка Spring и служит элементом, обеспечивающим корректную передачу данных из БД на клиентскую часть для дальнейшего отображения. При этом передача данных представляемых таблицами, происходит с использованием serverside пагниации для оптимизации. Таким образом, клиентской части предоставляется RESTful API, через который и происходит взаимодействие с приложением и БД.

Клиентская составляющая обращается к предоставляемому API и отображает данные в удобочитаемом виде. При разработке использовалась модульная архитектура, с организацией по папкам компонентов, страниц, виджетов и общих ресурсов. Глобальные стили определены в style.css, а Vite используется для сборки и оптимизации. Проект поддерживает TypeScript для строгой типизации. Разработанная верстка предусматривает последующее добавление разделов для каждой категории пользователей. После сборки, развертывание и обслуживание обеспечивается с помощью Nginx и Docker.

4.2 Использованные технологии

БД: MongoDB.

Backend: Spring Framework, Java

Frontend: TypeScript, Vue3

4.3 Схема экранов приложения

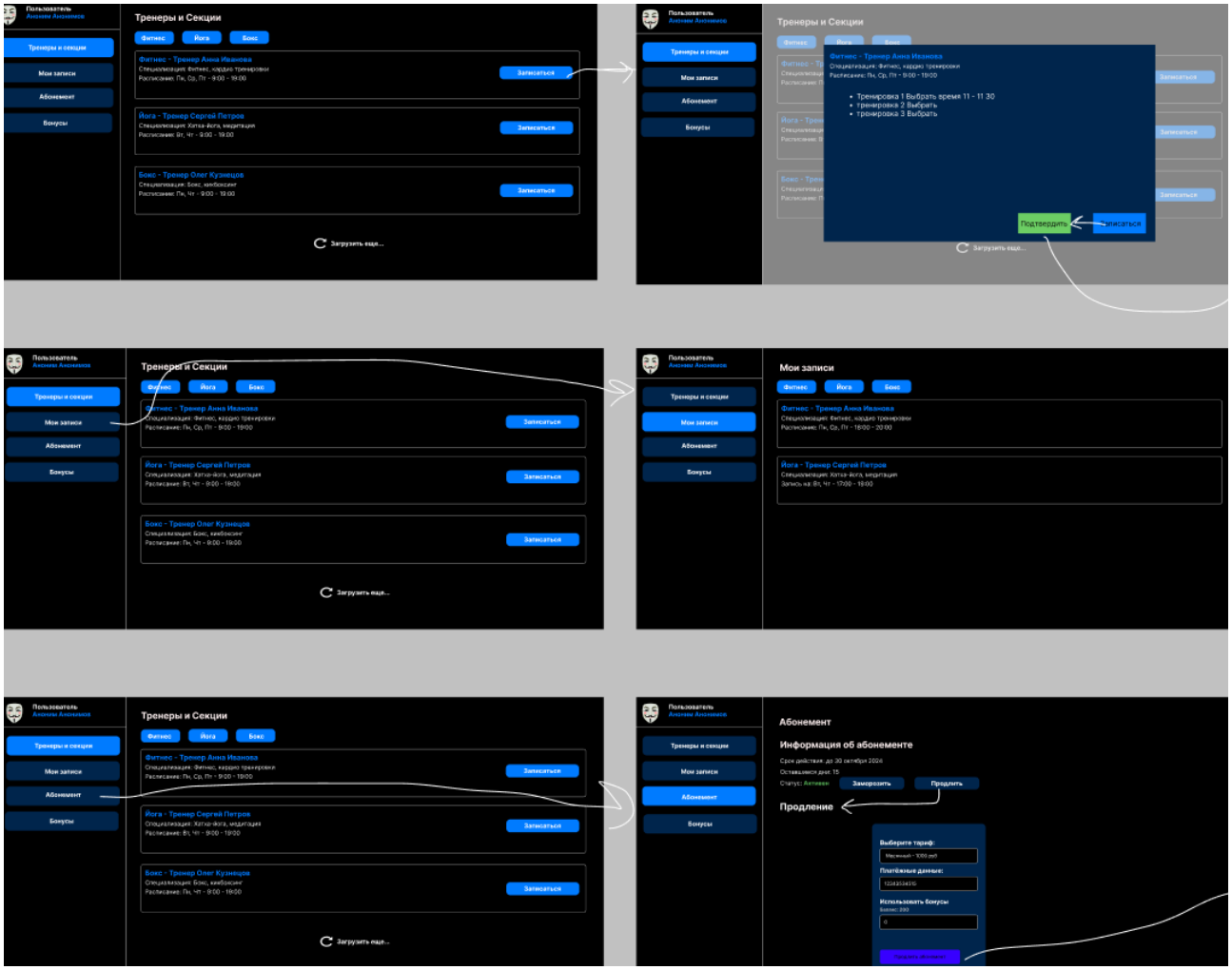

Рисунок 10. Схема экранов клиента

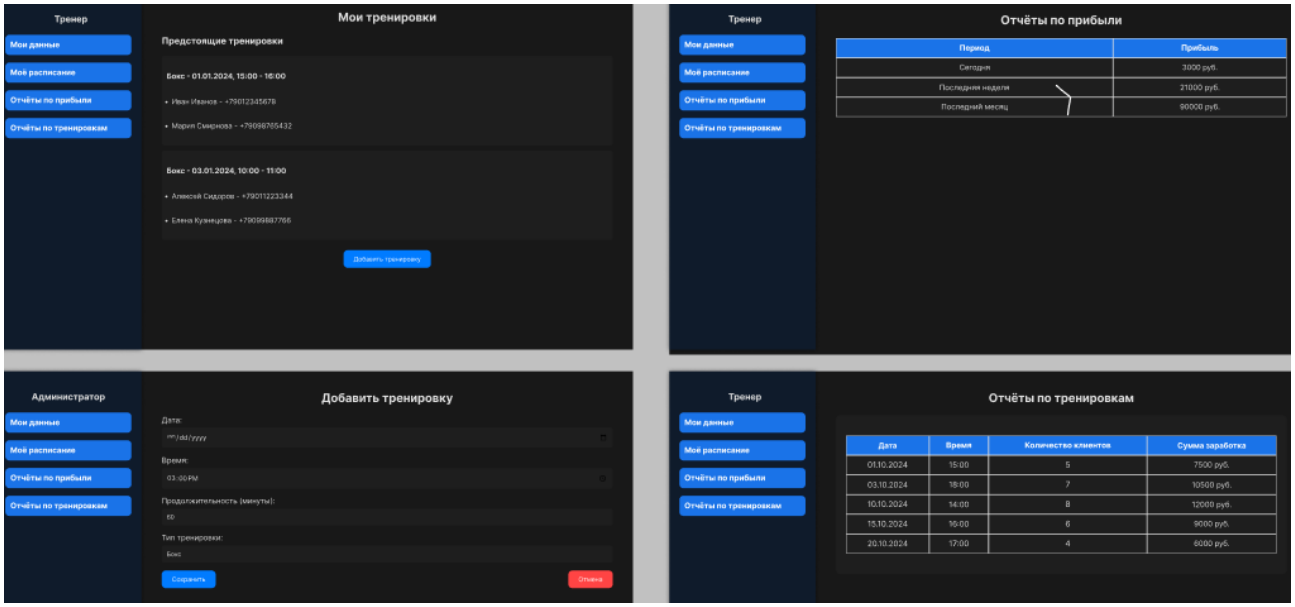

Рисунок 11. Схема экранов тренера

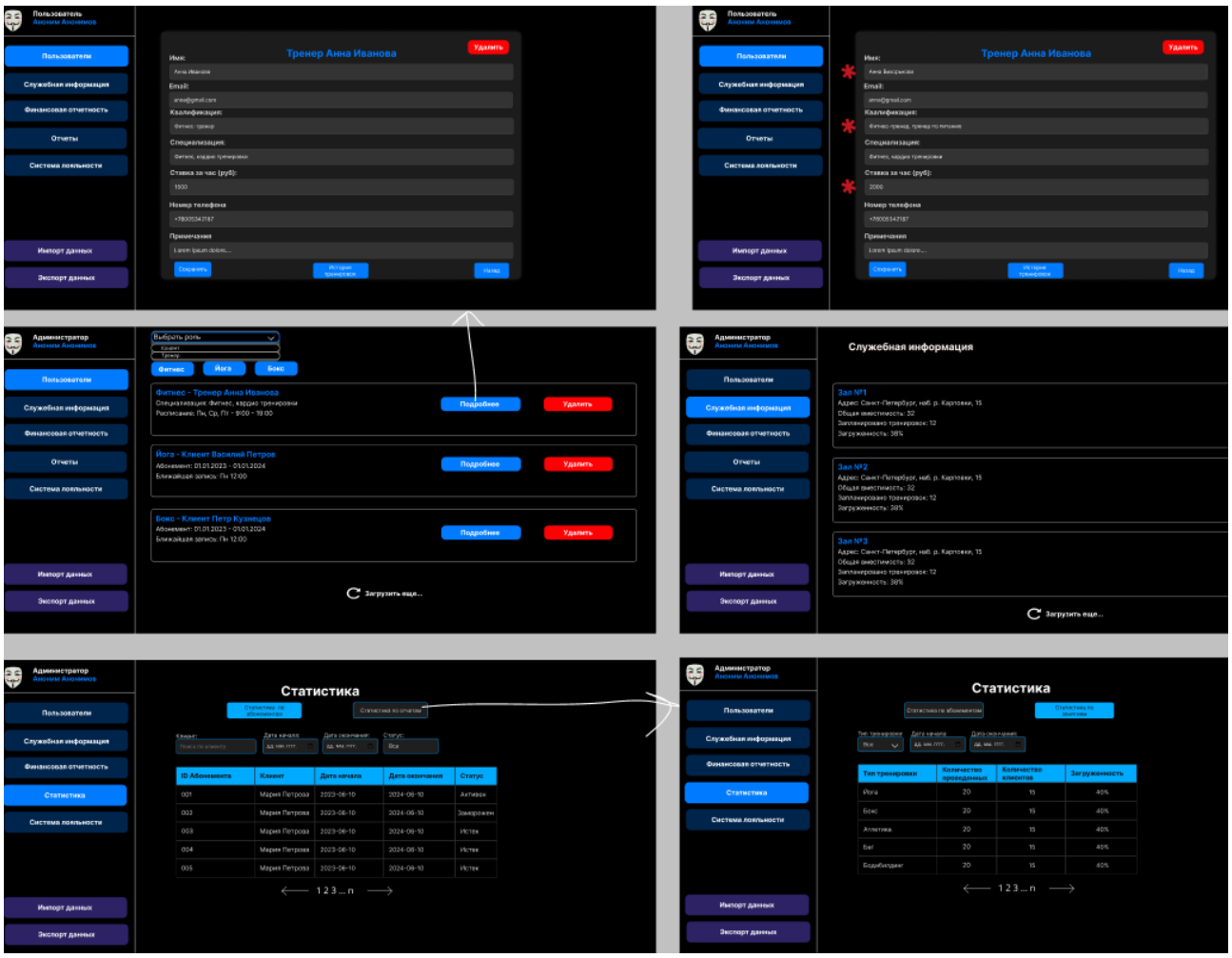

Рисунок 12. Схема экранов администратора

5. ВЫВОДЫ

5.1 Достигнутые результаты

В ходе работы было разработано приложение "IronGym", представляющее собой веб-приложение, которое предоставляет пользователям возможности управления расписанием занятий и тренировочным процессом. Приложение ориентировано на предоставление клиентам удобного интерфейса для просмотра расписания тренировок, записи на занятия, а также управления своими абонементами.

Для тренеров реализована возможность управления своим расписанием, добавления новых тренировок, а также отслеживания количества записанных участников, просмотр их контактной информации и дохода по занятиям.

Администраторы могут управлять пользователями и залами, анализировать статистику по абонементам и занятиям, импортировать и экспортировать данные, что способствует удобству переноса и резервного копирования данных.

5.2 Недостатки и пути для улучшения

На данный момент в приложении отсутствует работа с медиа: нет возможности прикрепления изображений для аккаунтов пользователей и для залов. Кроме того, для секций были бы полезные краткие видео-описания от лица тренеров, что улучшило бы пользовательский опыт.

Приложение доступно только на русском языке, таким образом, представляется возможность для охвата большей аудитории, с помощью поддержки локализации.

Также целевой платформой для приложения выступали персональные компьютеры, использование разработанного веб приложения на мобильных устройствах представляется ограниченным, что могло бы приголиться как клиентам, так и тренерам. Таким образом, необходимо рассмотреть возможность адаптации верстки для мобильных устройств и/или создания мобильной версии приложения.

5.3 Будущее развитие решения

В будущем планируется внедрение работы с медиа контентом, как с изображениями, так и с видеороликами для улучшения пользовательского опыта. Также планируется внедрения бонусной системы и настраиваемых акций, что повысит лояльность клиентов. Адаптация верстки веб-приложения для различных устройств будет полезна пользователям на различных платформах и потенциально расширит охват аудитории. Внедрение локализации так же положительно скажется на возможном охвате аудитории.

6. ПРИЛОЖЕНИЯ

6.1 Документация по сборке и развертыванию приложения.

1. Склонировать репозиторий проекта по ссылке [1] с помощью команды *git clone*.
2. Выполнить сборку образов: *docker-compose build --no-cache*.
3. Произвести запуск контейнеров командой: *docker-compose up*.

Клиентская часть приложения доступна в браузере по адресу http://127.0.0.1:3000. По умолчанию присутствует возможность входа с данными отладочных пользователей, указанными в README файле репозитория.

6.2 Инструкция для пользователя.

Массовый импорт-экспорт данных:

Импорт и экспорт данных доступен только пользователю с ролью администратора, по умолчанию предоставляется отладочный администратор с логином admin@gmail.com и паролем admin. При массовом импорте-экспорте используется единый файл exported_data,json. При нажатии на раздел «Экспорт» и кнопку «Скачать», файл будет сохранен на компьютер пользователя. По окончанию импорта система отобразит уведомление об успехе либо ошибке (например при несоответствии формата).

Требования:

В системе должна быть доступна платформа контейнеризации Docker и быть свободными порты 8080 и 3000.

Возможности:

- Для клиента – просмотр занятий и запись, редактирование информации своего профиля, управление абонементом, включая продление и заморозку, просмотр своих записей
- Для тренера – управление своим расписанием, включая создание и изменение занятий, редактирование информации своего профиля, просмотр контактной информации о своих клиентах и прибыли за проведенные занятия

- Для администратора – управление всеми пользователями, включая удаление и редактирование информации, массовый импорт и экспорт, просмотр статистики по всем абонементам и проведенным занятиям, просмотр служебной информации.

7. ЛИТЕРАТУРА

1. Репозиторий проекта - [Электронный ресурс]. - URL: https://github.com/moevm/nosql2h24-gym
2. Vue.js. – [Электронный ресурс]. – URL: https://vuejs.org/ (дата обращения: 19.12.2024).
3. MongoDB. – [Электронный ресурс]. – URL: https://www.mongodb.com/ (дата обращения: 19.12.2024).
4. Vue Datepicker. – [Электронный ресурс]. – URL: https://vue3datepicker.com/ (дата обращения: 19.12.2024).
5. Spring Framework – [Электронный ресурс]. – URL: https://spring.io/ (дата обращения: 19.12.2024).