

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

КУРСОВОЙ ПРОЕКТ
по дисциплине «Введение в нереляционные базы данных»
Тема: Сервис для indie групп - поиск замен музыкантов, реп.точки,
концерты

Студент гр. 1304	_____	Заика Т.П.
Студент гр. 1304	_____	Кардаш Я.Е.
Студент гр. 1304	_____	Стародубов М.В.
Преподаватель	_____	Заславский М.М.

Санкт-Петербург
2024

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

Студент Заика Т.П.

Студент Кардаш Я.Е.

Студент Стародубов М.В.

Группа 1304

Тема проекта: Сервис для indie групп - поиск замен музыкантов, реп.точки, концерты

Исходные данные:

Предложенная тема проекта с соответствующей нереляционной СУБД ArangoDB, требования к реализации конечного приложения через этапы: Согласована и сформулирована тема курсовой, Use case, Модель данных, Прототип «Хранение и представление», Прототип «Анализ», App is ready, Пояснительная записка.

Содержание пояснительной записки:

«Введение», «Сценарии использования», «Модель данных», «Разработанное приложение», «Выводы», «Приложения», «Литература»

Предполагаемый объем пояснительной записки:

Не менее 30 страниц.

Дата выдачи задания: 25.09.2024

Дата сдачи реферата: 22.12.2024

Дата защиты реферата: 24.12.2024

Студент гр. 1304

Заика Т.П.

Студент гр. 1304

Кардаш Я.Е.

Студент гр. 1304

Стародубов М.В.

Преподаватель

Заславский М.М.

АННОТАЦИЯ

В курсовом проекте реализовано приложения на заданную тему с использованием соответствующей нереляционной СУБД ArangoDB. Представлена актуальность решаемой проблемы, описано предлагаемое решение. Разработаны сценарии использования и проведено сравнение моделей данных. Кратко описано разработанное приложение с использованием снимков экрана. В выводах указаны достигнутые результаты, недостатки и пути улучшения решения, будущее развитие решения.

SUMMARY

The course project implements applications on a given topic using the corresponding non-relational DBMS ArangoDB. The relevance of the problem being solved is presented, the proposed solution is described. Usage scenarios are developed and a comparison of data models is conducted. The developed application is briefly described using screenshots. The conclusions indicate the achieved results, shortcomings and ways to improve the solution, and future development of the solution.

СОДЕРЖАНИЕ

Введение	6
1. Введение	7
1.1. Актуальность решаемой проблемы	7
1.2. Постановка задачи	7
1.3. Предлагаемое решение	7
1.4. Качественные требования к решению	7
2. Сценарии использования	9
2.1. Макет UI	9
2.2. Сценарии использования для задачи	9
2.3. Вывод о том, какие операции преобладают	11
3. Модель данных	12
3.1. Нереляционная модель данных	12
3.2. Аналог модели данных для SQL СУБД	20
3.3. Сравнение моделей	27
4. Разработанное приложение	29
4.1. Краткое описание	29
4.2. Используемые технологии	29
4.3. Снимки экрана приложения	29
5. Выводы	35
5.1. Достигнутые результаты	35
5.2. Недостатки и пути для улучшения полученного решения	35
5.3. Будущее развитие решения	35
Заключение	36
Литература	37
Приложение А. Документация по сборке и развертыванию приложения	38
Приложение Б. Инструкция для пользователя	39

ВВЕДЕНИЕ

Целью курсового проекта является реализация приложения на тему «Сервис для indie групп - поиск замен музыкантов, реп.точки, концерты» с использованием нереляционной СУБД ArangoDB. При подготовке к реализации рассмотрены сценарии использования и исследованы и сравнены нереляционная и реляционная модели данных. Для реализации использовались такие технологии и языки программирования, как Python, Javascript, FastAPI, Pydantic, Pyarango, React.

1. ВВЕДЕНИЕ

1.1. Актуальность решаемой проблемы

В современном обществе появляется множество музыкантов, которые объединяются в инди-группы. Таким группам необходимо контактировать между собой с целью сотрудничества и обмена опытом. Для удобства данного взаимодействия предлагается реализовать веб-сервис, где инди-группы смогут дружить и сотрудничать.

1.2. Постановка задачи

Сделать веб-сервис, где indie-группы смогут дружить и сотрудничать. Профили групп, музыкантов, реп.точки, объявления, комментарии, рейтинги, статистика.

1.3. Предлагаемое решение

Каждый пользователь (музыканты, владельцы музыкальных заведений) создает свою страничку, на которой он может указать свои навыки владения музыкальными инструментами или предоставляемые ими услуги. Пользователи могут объединяться в группы. В приложении доступен список свободных репетиционных и концертных точек, студий звукозаписи. Пользователи могут оставлять объявления о проведении концертов или поиске недостающих членов группы.

В качестве базы данных приложение будет использовать ArangoDB, взаимодействие с базой данных будет происходить с помощью языка программирования Python. Пользовательский интерфейс будет реализован с помощью React. Взаимодействие с серверной стороной будет происходить с помощью FastAPI.

1.4. Качественные требования к решению

- Приложение должно реализовывать все оговоренные сценарии использования через пользовательский интерфейс.
- Приложение должно использовать нереляционную СУБД и по максимуму использовать ее функционал.
- Приложение должно иметь авторизацию.
- Приложение должно иметь возможность просмотра, добавление и изменения данных из нереляционной СУБД.
- Пользовательский интерфейс приложения должен предоставлять возможность фильтрации и поиска данных.
- Приложение разворачивается при помощи docker-compose.
- Приложение должно предоставлять возможность импорта и экспорта всех данных.
- Приложение должно иметь serverside пагинацию.
- Приложение должно иметь отображение статистики данных в пользовательском интерфейсе.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет UI



Рисунок 1 – Макет UI

2.2. Сценарии использования для задачи

2.2.1. Импорта данных

2.2.1.a. Как пользователь загружает данные в программу массово

Для того, чтобы пользователь мог загрузить данные в программу массово, необходимо последовать на страницу статистики.

Затем необходимо нажать на кнопку «Импортировать данные» и загрузить в файл с компьютера с данными в формате .json с содержимым по следующей структуре:

```
{  
  "static": [ ],  
  "vertices": {  
    "users": [],  
    "groups": [],  
    "announcements": [],  
    "places": []  
  },  
  "edges": {  
    "users": {  
      "groups": []  
    }  
  }  
}
```

```

    },
    "stars": {
        "users": [],
        "groups": [],
        "announcements": []
    },
    "comments": {
        "announcements": []
    },
    "producer_announcements": {
        "users": [],
        "groups": []
    }
}
}

```

После загрузки будет сказано об успешности или неуспешности операции. Затем через 5 секунд приложение будет автоматически перезагружено и пользователю нужно будет войти в профиль еще раз.

2.2.1.б. Как пользователь загружает данные в программу вручную

Для того, чтобы пользователь смог загрузить данные в программу вручную, он должен зарегистрироваться, создать группу, создать место, создать объявление, поставить звезды пользователям, группам и объявлениям, написать комментарий к объявлению.

2.2.2. Представления данных

2.2.2.а. Как данные отображаются для пользователя и что он должен сделать для их отображения

Данные отображаются для пользователя в веб-браузере по адресу <http://localhost:3000> в виде разделов с списками пользователей, групп, объявлений и мест, статистика отображается в виде суммарных значений и графика.

Для их полного отображения пользователь должен зарегистрироваться в приложении или войти в свой профиль.

2.2.3. Анализа данных

2.2.3.a. Какие действия должен сделать пользователь для проведения анализа данных

Для проведения анализа данных пользователь должен перейти на любую из страниц сущностей, то есть перейти в раздел списка пользователей, групп, мест или объявлений и произвести фильтрацию.

Также на странице статистики можно просмотреть суммарные значения и графики. При просмотре графиков также можно выбрать сущность, отфильтровать ее и выбрать атрибуты для отображения.

2.2.4. Экспорта данных

2.2.4.a. Как пользователю получить копию хранимых в программе данных в машиночитаемом формате

Для того, чтобы пользователь смог получить копию хранимых в программе данных в машиночитаемом формате необходимо перейти на страницу статистики и нажать на кнопку «Экспорт данных». При нажатии произойдет загрузка данных в формате .json с именем data.json. Данные описаны по структуре, указанно в пункте 2.2.1.a.

2.3. Вывод о том, какие операции преобладают

В приложении пользователи чаще всего создают профили, объявления, места, ставят звезды пользователям, объявлениям и местам, комментируют объявления, однако при каждом переходе между разделами производятся операции чтения из БД. Пользователи в большинстве своем чаще смотрят контент, чем создают его, поэтому в приложении преобладает операция чтения.

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель данных

3.1.1. Графическое представление модели

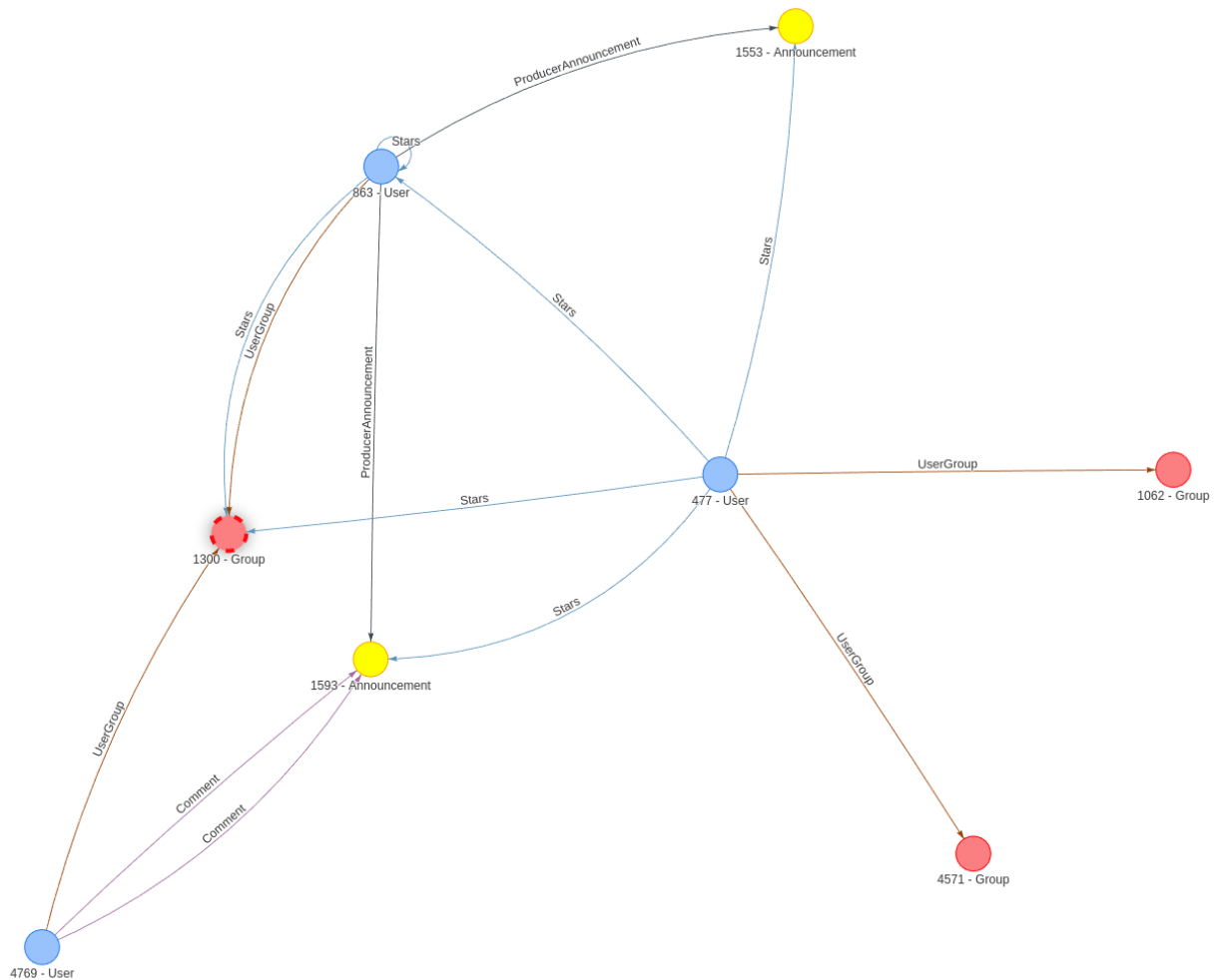


Рисунок 2 – Графическое представление нереляционной базы данных

3.1.2. Описание назначений коллекций, типов данных и сущностей

Коллекция Announcement

Назначение

Хранение информации об объявлениях

```
{  
  "title": "Announcement",  
  "description": "Коллекция объявлений",  
  "type": "object",  
}
```

```

"properties": {
  "creation_date": {
    "description": "Дата публикации",
    "type": "number"
  },
  "content": {
    "description": "Содержимое объявления",
    "type": "string"
  },
  "tag": {
    "description": "Ассоциированный тэг",
    "type": "string"
  }
},
"required": ["creation_date", "content", "tag"]
}

```

Суммарный объем данных: 530 байт.

Коллекция Comment

Назначение

Хранение информации о комментариях к объявлениям

```

{
  "title": "Comment",
  "description": "Коллекция комментариев",
  "properties": {
    "creation_date": {
      "description": "Дата создания",
      "type": "number"
    },
    "content": {
      "description": "Содержимое комментария",
      "type": "string"
    }
  },
  "required": ["creation_date", "content"]
}

```

Суммарный объем данных: 241 байт.

Коллекция Group

Назначение

Хранение информации о группах

```

{
  "title": "Group",
  "description": "Коллекция групп",
  "properties": {
    "name": {

```

```

    "description": "Название группы",
    "type": "string"
  },
  "creation_date": {
    "description": "Дата создания",
    "type": "number"
  },
  "last_edit_date": {
    "description": "Дата последнего изменения",
    "type": "number"
  },
  "avatar_uri": {
    "description": "Ссылка на аватар",
    "type": "string"
  },
  "genres": {
    "description": "Список жанров группы",
    "type": "array",
    "items": {
      "type": "string"
    },
    "uniqueItems": true
  }
},
"required": ["name", "creation_date", "last_edit_date", "avatar_uri", "genres"]
}

```

Суммарный объем данных: 348 байт.

Коллекция Place

Назначение

Хранение информации о местах

```

{
  "title": "Place",
  "description": "Коллекция мест",
  "properties": {
    "user_id": {
      "description": "Идентификатор пользователя-создателя",
      "type": "string"
    },
    "name": {
      "description": "Название места",
      "type": "string"
    },
    "creation_date": {
      "description": "Дата создания",
      "type": "number"
    },
    "last_edit_date": {
      "description": "Дата последнего изменения",
      "type": "number"
    }
  }
}

```

```

"avatar_uri": {
  "description": "Ссылка на аватар",
  "type": "string"
},
"type": {
  "description": "Тип места",
  "type": "string"
},
"address": {
  "description": "Адрес места",
  "type": "string"
},
"phone_number": {
  "description": "Контактный номер",
  "type": "string"
},
"area": {
  "description": "Площадь",
  "type": "number"
},
"equipment": {
  "description": "Список оборудования",
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "name": {
        "description": "Название оборудования",
        "type": "string"
      },
      "amount": {
        "description": "Количество оборудования",
        "type": "number"
      }
    },
    "required": ["name", "amount"]
  },
  "uniqueItems": true
},
"required": ["user_id", "name", "creation_date", "last_edit_date", "avatar_uri", "type", "address",
"phone_number", "area", "equipment"]
}

```

Суммарный объем данных: 1572 байт.

Коллекция **ProducerAnnouncement**

Назначение

Хранение информации об отношениях между создателями объявлений и объявлениями

```

{
  "title": "ProducerAnnouncement",

```

```
"description": "Коллекция отношений между создателями объявлений и объявлениями"
}
```

Суммарный объем данных: 66 байт.

Коллекция Stars

Назначение

Хранение информации о поставленных звездах

```
{
  "title": "Stars",
  "description": "Коллекция звезд"
}
```

Суммарный объем данных: 41 байт.

Коллекция Static

Назначение

Хранение информации о статичных данных

```
{
  "title": "Static",
  "description": "Коллекция статичных данных",
  "properties": {
    "equipment": {
      "description": "Список доступного оборудования",
      "type": "array",
      "items": {
        "type": "string"
      },
      "uniqueItems": true
    },
    "tags": {
      "description": "Список доступных тегов",
      "type": "array",
      "items": {
        "type": "string"
      },
      "uniqueItems": true
    },
    "talents": {
      "description": "Список доступных талантов",
      "type": "array",
      "items": {
        "type": "string"
      },
      "uniqueItems": true
    },
    "genres": {
      "description": "Список доступных жанров",
```



```

        "type": "array",
        "items": {
            "type": "string"
        },
        "uniqueItems": true
    },
    "required": ["equipment", "tags", "talents", "genres"]
}

```

Суммарный объем данных: 8200 байт.

Коллекция User

Назначение

Хранение информации о пользователе

```

{
    "title": "User",
    "description": "Коллекция пользователей",
    "properties": {
        "first_name": {
            "description": "Имя пользователя",
            "type": "string"
        },
        "last_name": {
            "description": "Фамилия пользователя",
            "type": "string"
        },
        "creation_date": {
            "description": "Дата создания",
            "type": "number"
        },
        "last_edit_date": {
            "description": "Дата последнего изменения",
            "type": "number"
        },
        "avatar_uri": {
            "description": "Ссылка на аватар",
            "type": "string"
        },
        "talents": {
            "description": "Список жанров группы",
            "type": "array",
            "items": {
                "type": "string"
            },
            "uniqueItems": true
        }
    },
    "required": ["first_name", "last_name", "creation_date", "last_edit_date", "avatar_uri", "talents"]
}

```

Суммарный объем данных: 492 байт.

Коллекция UserGroup

Назначение

Хранение информации об отношениях между пользователями и группами

```
{
  "title": "UserGroup",
  "description": "Коллекция отношений между пользователями и группами",
  "properties": {
    "join_date": {
      "description": "Дата присоединения пользователя к группе",
      "type": "number"
    }
  }
  "required": ["join_date"]
}
```

Суммарный объем данных: 32 байта.

3.1.3. Оценка объема информации, хранимой в модели

В качестве переменной используется количество пользователей U .

Таблица Static не зависит от числа пользователей, хранится в единственном экземпляре и имеет размер 8200 байт.

Количество групп выразим через количество пользователей как $G = 0.6U$.

Для каждого пользователя приходится в среднем 2 таланта, 30 объявлений, 500 звезд суммарно, 50 комментариев, 2 места. У каждого места в среднем по 10 различных видов оборудования.

Для каждой группы приходится 30 объявлений и 2 жанра.

Тогда примерный объем модели можно выразить следующей формулой:

$$\begin{aligned} V(U) &= 8200 + U \cdot (492 + 30 \cdot (66 + 530) + 500 \cdot 41 + 50 \cdot 241 + 2 \cdot 1572) \\ &\quad + 0.6 \cdot U \cdot (348 + 30 \cdot (66 + 530) + 32) \\ &= 8200 + 65022 \cdot U \text{ байт} \end{aligned}$$

3.1.4. Примеры запросов

Текст запросов

```
1. Самая популярная группа
FOR g in Group
LET stars_amount = (
  FOR s IN Stars
  FILTER s._to == g._id
  COLLECT WITH COUNT INTO amount
  RETURN amount
)[0]
SORT stars_amount DESC
LIMIT 1
RETURN {
```

```

_id: g._id,
name: g.name,
stars_amount: stars_amount
}

```

2. Места с типом “Концертная точка”

```

FOR p IN Place
  FILTER p.type == "Концертная точка"
  RETURN p

```

3. Редактирование данных профиля пользователем

```

LET key = PARSE_IDENTIFIER("User/477").key
UPDATE key
WITH {
  first_name: "Thomas",
  last_name: "York",
  last_edit_date: DATE_FORMAT(DATE_NOW(), "%yyyy-%mm-%ddT%hh:%ii:%ss.%fffZ"),
  avatar_uri: "assets/avatars/thomas-new.png"
}
IN User
RETURN NEW

```

4. Записи с тегом “концерт” (в порядке уменьшения звезд на них)

```

FOR a in Announcement
  LET stars_amount = (
    FOR s IN Stars
      FILTER s._to == a._id
      COLLECT WITH COUNT INTO amount
    RETURN amount
  )[0]
  SORT stars_amount DESC
  RETURN MERGE(a, {
    stars_amount: stars_amount
  })

```

5. Все анонсы пользователя

```

FOR pa IN ProducerAnnouncement
  FILTER pa._from == @userId
  FOR a IN Announcement
    FILTER pa._to == a._id
  RETURN a

```

6. Все комментарии пользователя

```

FOR c IN Comments
  FILTER c._from == @userId
  RETURN c

```

7. Все авторы комментариев к определенному анонсу

```

FOR a IN Announcement
  FILTER a._id == @ann_id
  FOR c IN Comment
    FILTER c._to == a._id
    FOR u IN User
      FILTER c._from == u._id
    RETURN u

```

8. Все группы, чьи пользователи поставили звезду на определенный анонс

```

FOR s IN Star
  FILTER s._to == @ann_id
FOR ug IN UserGrop
  FILTER ug._from == s._from
FOR g in Group
  FILTER ug._to == g._id
RETURN g

```

3.2. Аналог модели данных для SQL СУБД

3.2.1. Графическое представление модели

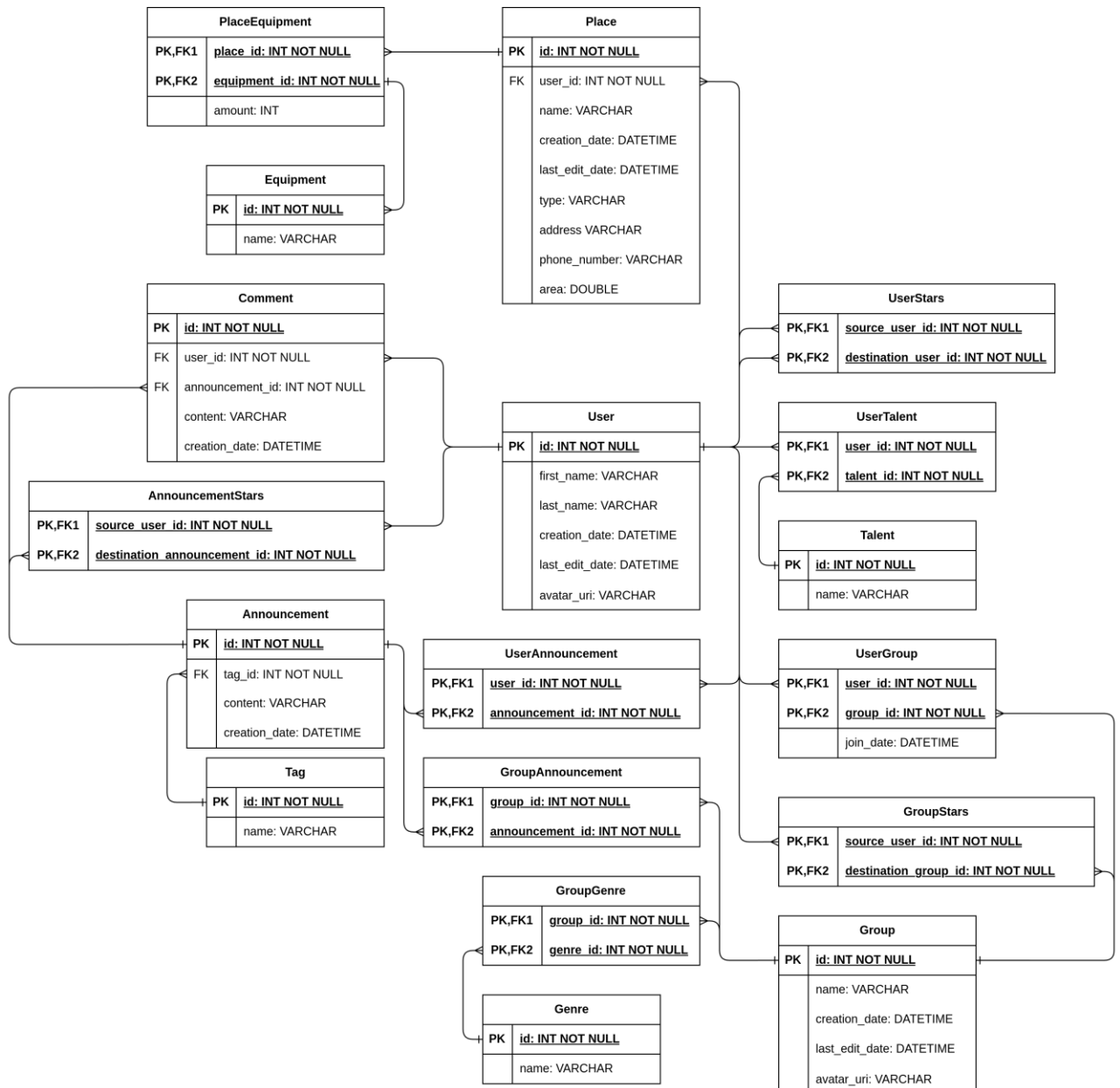


Рисунок 3 – Графическое представление SQL СУБД

3.2.2. Описание назначений коллекций, типов данных и сущностей

Таблица User

Назначение

Информация о пользователе

Типы данных

- id: INT NOT NULL
- first_name: VARCHAR
- last_name: VARCHAR
- creation_data: DATETIME
- last_edit_date: DATETIME
- avatar_uri: VARCHAR

Суммарный объем данных: 785 байт.

Таблица UserStars

Назначение

Информация о полученных звездах пользователя

Типы данных

- source_user_id: INT NOT NULL
- destination_user_id: INT NOT NULL

Суммарный объем данных: 8 байт.

Таблица UserTalent

Назначение

Информация о музыкальных талантах пользователя

Типы данных

- user_id: INT NOT NULL
- talent_id: INT NOT NULL

Суммарный объем данных: 8 байт.

Таблица UserGroup

Назначение

Информация о группе, в которой состоит пользователь

Типы данных

- user_id: INT NOT NULL

- group_id: INT NOT NULL
- join_date: DATETIME

Суммарный объем данных: 16 байт.

Таблица UserAnnouncement

Назначение

Информация об объявлении, которое получает пользователь

Типы данных

- user_id: INT NOT NULL
- announcement_id: INT NOT NULL

Суммарный объем данных: 8 байт.

Таблица Talent

Назначение

Информация о музыкальном таланте

Типы данных

- id: INT NOT NULL
- name: VARCHAR

Суммарный объем данных: 259 байт.

Таблица Group

Назначение

Информация о группе

Типы данных

- id: INT NOT NULL
- name: VARCHAR
- creation_date: DATETIME
- last_edit_date: DATETIME
- avatar_uri: VARCHAR

Суммарный объем данных: 530 байт.

Таблица GroupAnnouncement

Назначение

Информация об объявлении группы

Типы данных

- group_id: INT NOT NULL
- announcement_id: INT NOT NULL

Суммарный объем данных: 8 байт.

Таблица GroupStars

Назначение

Информация о полученных звездах группы

Типы данных

- source_user_id: INT NOT NULL
- destination_group_id: INT NOT NULL

Суммарный объем данных: 8 байт.

Таблица GroupGenre

Назначение

Информация о жанрах группы

Типы данных

- group_id: INT NOT NULL
- genre_id: INT NOT NULL

Суммарный объем данных: 8 байт.

Таблица Genre

Назначение

Информация о всех жанрах

Типы данных

- id: INT NOT NULL
- name: VARCHAR

Суммарный объем данных: 259 байт.

Таблица Announcement

Назначение

Информация об объявлении

Типы данных

- id: INT NOT NULL
- tag_id: INT NOT NULL
- content: VARCHAR
- creation_date: DATETIME

Суммарный объем данных: 271 байт.

Таблица AnnouncementStars

Назначение

Информация о звездах объявление

Типы данных

- source_user_id: INT NOT NULL
- destination_announcement_id: INT NOT NULL

Суммарный объем данных: 8 байт.

Таблица Comment

Назначение

Информация о комментарии

Типы данных

- id: INT NOT NULL
- user_id: INT NOT NULL
- announcement_id: INT NOT NULL
- content: VARCHAR
- creation_date: DATETIME

Суммарный объем данных: 275 байт.

Таблица Tag

Назначение

Информация о теге

Типы данных

- id: INT NOT NULL
- name: VARCHAR

Суммарный объем данных: 259 байт.

Таблица Place

Назначение

Информация о месте, репточке

Типы данных

- id: INT NOT NULL
- user_id: INT NOT NULL
- name: VARCHAR
- creation_date: DATETIME
- last_edit_date: DATETIME
- type: VARCHAR
- address: VARCHAR
- phone_number: VARCHAR
- area: DOUBLE

Суммарный объем данных: 1052 байт.

Таблица Equipment

Назначение

Информация об оборудовании

Типы данных

- id: INT NOT NULL
- name: VARCHAR

Суммарный объем данных: 259 байт.

Таблица PlaceEquipment

Назначение

Информация об оборудовании, доступном на репитиционных точках

Типы данных

- place_id: INT NOT NULL
- equipment_id: INT NOT NULL
- amount: INT

Суммарный объем данных: 12 байт.

3.2.3. Оценка объема информации, хранимой в модели

В качестве переменной используется количество пользователей U .

Таблицы Tag, Talent, Equipment и Genre не зависят от числа пользователей (имеют 50, 20, 20 и 20 объектов соответственно) и имеют размеры 12950 байт, 5180 байт, 5180 байт и 5180 соответственно.

Количество групп выразим через количество пользователей как $G = 0.6U$.

Для каждого пользователя приходится в среднем 2 таланта, 30 объявлений, 500 звезд суммарно, 50 комментариев, 2 места. У каждого места в среднем по 10 различных видов оборудования.

Для каждой группы приходится 30 объявлений и 2 жанра.

Тогда примерный объем модели можно выразить следующей формулой:

$$\begin{aligned} V(U) &= 12950 + 5180 + 5180 + 5180 + U \\ &\quad \cdot (785 + 2 \cdot 8 + 30 \cdot (8 + 271) + 500 \cdot 8 + 50 \cdot 275 + 2 \\ &\quad \cdot (1052 + 10 \cdot 12)) + 0.6 \cdot U \cdot (530 + 30 \cdot (8 + 271) + 2 \cdot 8) \\ &= 28490 + 34560 \cdot U \text{ байт} \end{aligned}$$

3.2.4. Примеры запросов

Текст запросов

1. Самая популярная группа

```
SELECT g.id, g.name, COUNT(gs.source_user_id) AS star_count
FROM Group g
LEFT JOIN GroupStars gs ON g.id = gs.destination_group_id
GROUP BY g.id, g.name
ORDER BY star_count DESC
LIMIT 1;
```

3. Места с типом “Концертная точка”

```
SELECT *
FROM Places
WHERE type = "Концертная точка"
```

3. Редактирование данных профиля пользователем

```
UPDATE User
SET
    first_name = NEW.first_name,
    last_name = NEW.last_name,
    creation_date = NEW.creation_date,
    last_edit_date = now(),
    avatar_url = NEW.avatar_url
WHERE id = NEW.id;
```

4. Записи с тегом “концерт” (в порядке уменьшения звезд на них)

```
SELECT a.*, COUNT(as.id) AS star_count
FROM Announcements a
LEFT JOIN AnnouncementsStars as ON a.id = as.destination_announcements_id
```

```
WHERE a.tag_id = (SELECT id FROM Tags WHERE tag_name = 'концерт')
GROUP BY a.id
ORDER BY star_count DESC;
```

5. Все объявления пользователя

```
SELECT Announcement.* FROM Announcement
JOIN UserAnnouncement ON Announcement.id = UserAnnouncement.announcement_id
WHERE UserAnnouncement.user_id = 0;
```

6. Все комментарии пользователя

```
SELECT * FROM Comments
WHERE user_id = 0;
```

7. Все авторы комментариев к определенному объявлению

```
SELECT DISTINCT User.* FROM User
JOIN Comment ON User.id = Comment.user_id
WHERE Comment.announcement_id = 0;
```

8. Все группы, чьи пользователи поставили звезду к заданному объявлению

```
SELECT DISTINCT Group.* FROM Group
JOIN UserGroup ON UserGroup.group_id = Group.id
JOIN AnnouncementStars ON UserGroup.user_id = AnnouncementStars.source_user_id
WHERE AnnouncementStars.destination_announcement_id = 1;
```

3.3. Сравнение моделей

3.3.1. Удельный объем информации

NoSQL - $8200 + 65022 * U$ байт.

SQL - $28490 + 34560 * U$ байт.

NoSQL будет требовать значительно больше памяти, чем SQL-решение.

3.3.2. Запросы по отдельным юзкейсам

3.3.2.1. Количество запросов для совершения юзкейсов в зависимости от числа объектов в БД и прочих параметров

1. Самая популярная группа

- NoSQL - 2 запроса.
- SQL - 1 запрос.

2. Места с типом “Концертная точка”

- NoSQL - 1 запрос.
- SQL - 1 запрос.

3. Редактирование данных профиля пользователем.

- NoSQL - 1 запрос.
- SQL - 2 запроса.

4. Записи с тегом “концерт” (в порядке уменьшения звезд на них)

- NoSQL - 2 запроса.
- SQL - 1 запрос.

3.3.2.2 Количество задействованных коллекций

1. Самая популярная группа

- NoSQL - 2 коллекции.
- SQL - 2 таблицы.

2. Места с типом “Концертная точка”

- NoSQL - 1 коллекция.
- SQL - 1 таблица.

3. Редактирование данных профиля пользователем.

- NoSQL - 1 коллекция.
- SQL - 2 таблицы.

4. Записи с тегом “концерт” (в порядке уменьшения звезд на них)

- NoSQL - 2 коллекции.
- SQL - 2 таблицы.

3.3.3. Вывод

- Удобнее использовать NoSQL так как необходимо реализовывать множество связей “многие ко многим”.
- NoSQL требует значительно больше памяти, чем SQL-решение.
- Количество запросов и задействованных коллекций в NoSQL и SQL примерно равны.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

Приложение имеет название «GigHunt», что можно перевести как «Охота за концертами».

Приложение состоит из трех контейнеров – frontend, backend и arango.

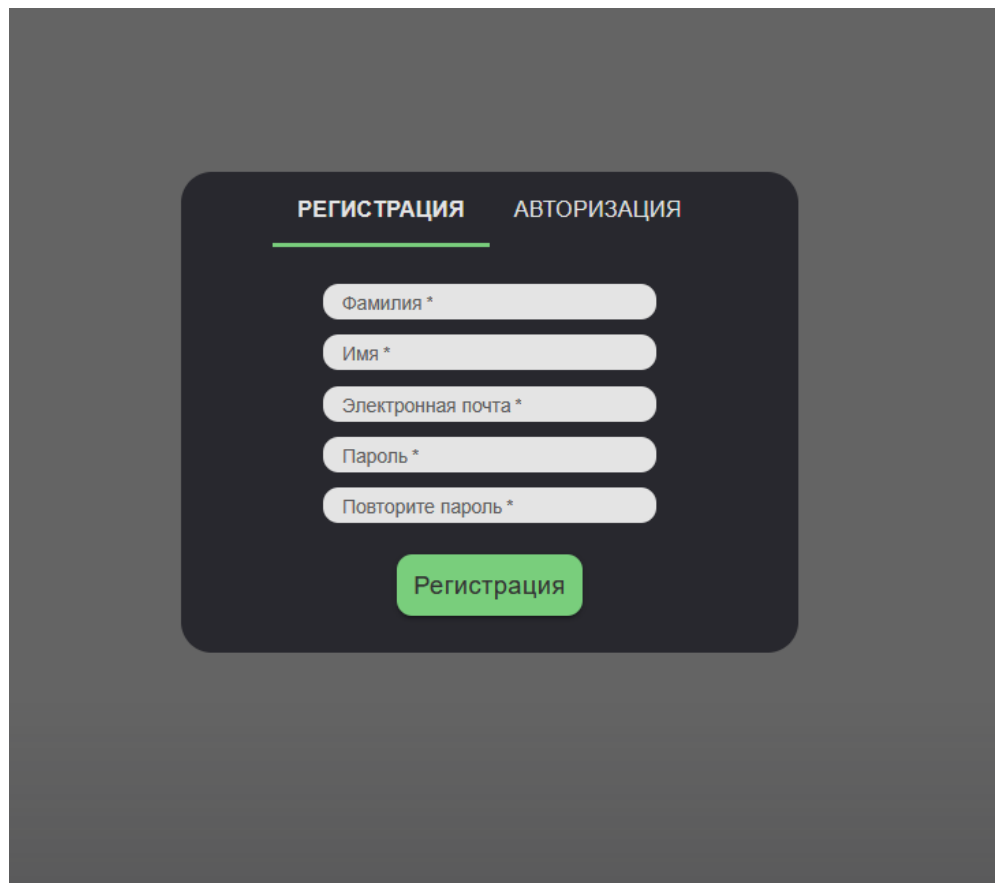
Используется классическая архитектура связи frontend и backend.

В приложение каждый пользователь регистрируется и имеет свою страничку. На ней он может указать свои навыки владения музыкальными инструментами или предоставляемые услуги. Пользователи могут создавать и объединяться в группы. В приложении доступен список свободных репетиционных и концертных точек, новые точки могут быть созданы пользователями. Пользователи могут оставлять объявления на личной странице или на странице группы, могут комментировать объявления, ставить «звезды» на объявления, группы или исполнителей. Приложение предоставляет возможность фильтрации пользователей, групп, объявлений и точек. В приложении есть возможность сделать импорт и экспорт данных, просмотреть статистику.

4.2. Используемые технологии

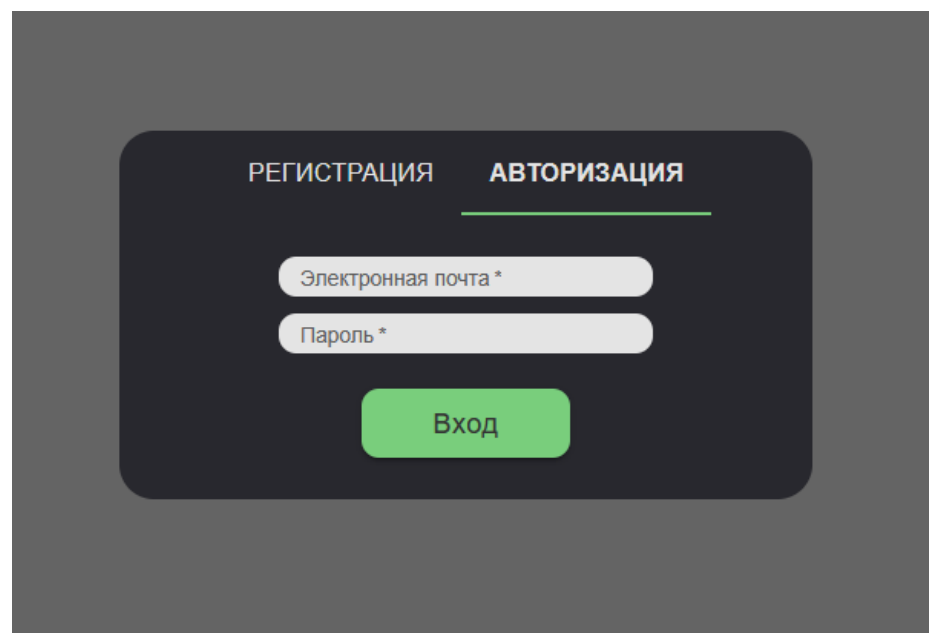
Использовались следующие технологии: ArangoDB, Python, Javascript, Pydantic, Pyarango, FastAPI, React, Docker, Docker compose.

4.3. Снимки экрана приложения



The image shows a registration form on a dark background. At the top, there are two tabs: "РЕГИСТРАЦИЯ" (Registration) and "АВТОРИЗАЦИЯ" (Authorization). The "РЕГИСТРАЦИЯ" tab is selected, indicated by a green underline. Below the tabs, there are five input fields: "Фамилия *" (Surname), "Имя *" (Name), "Электронная почта *" (Email), "Пароль *" (Password), and "Повторите пароль *" (Repeat password). All fields are currently empty. Below the input fields is a green button labeled "Регистрация" (Registration).

Рисунок 4 – страница регистрации



The image shows an authorization form on a dark background. At the top, there are two tabs: "РЕГИСТРАЦИЯ" (Registration) and "АВТОРИЗАЦИЯ" (Authorization). The "АВТОРИЗАЦИЯ" tab is selected, indicated by a green underline. Below the tabs, there are two input fields: "Электронная почта *" (Email) and "Пароль *" (Password). Both fields are currently empty. Below the input fields is a green button labeled "Вход" (Login).

Рисунок 5 - страница авторизации

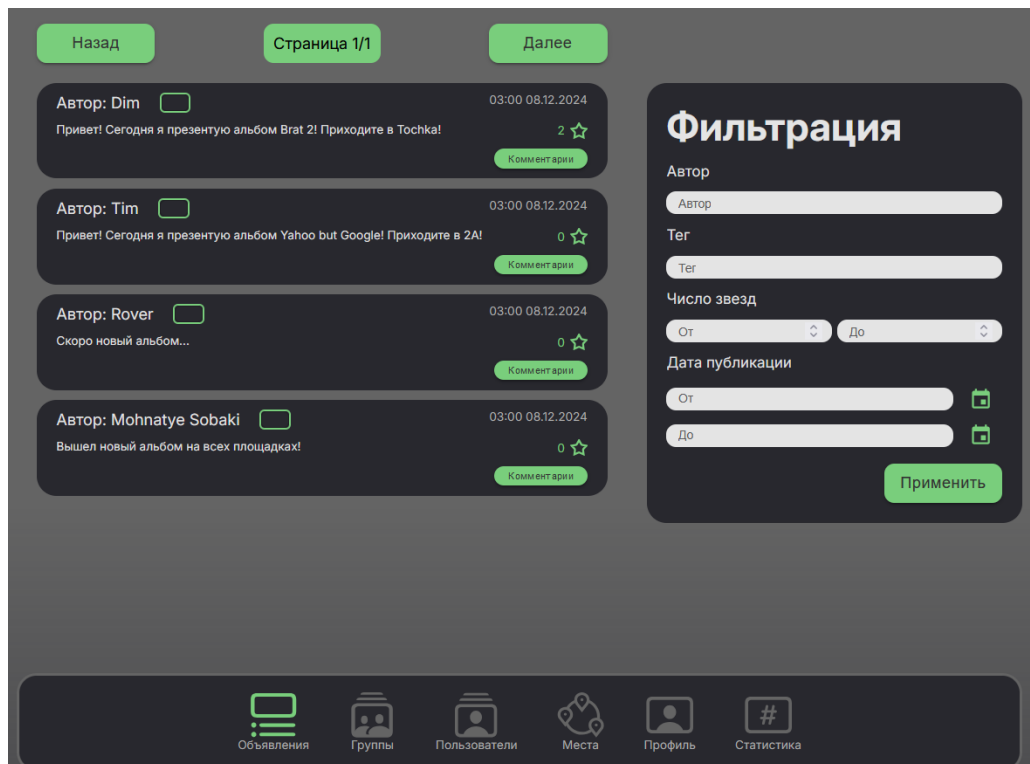


Рисунок 6 – страница списка объявлений

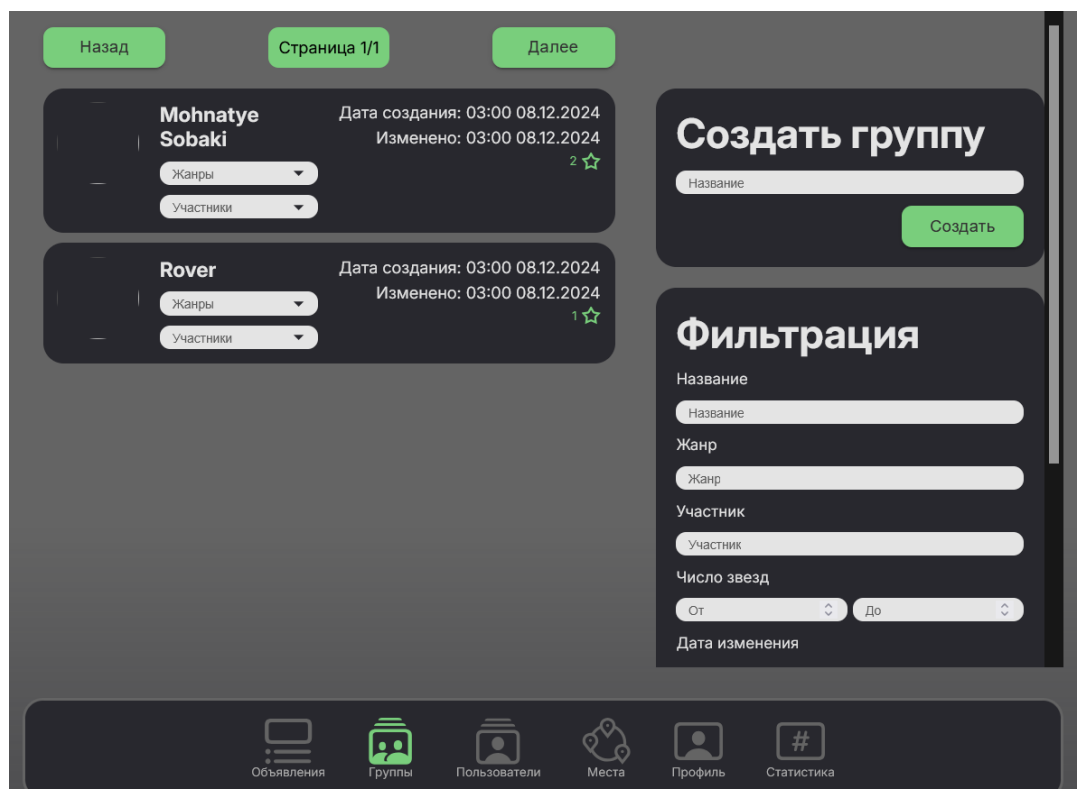


Рисунок 7 – страница списка групп

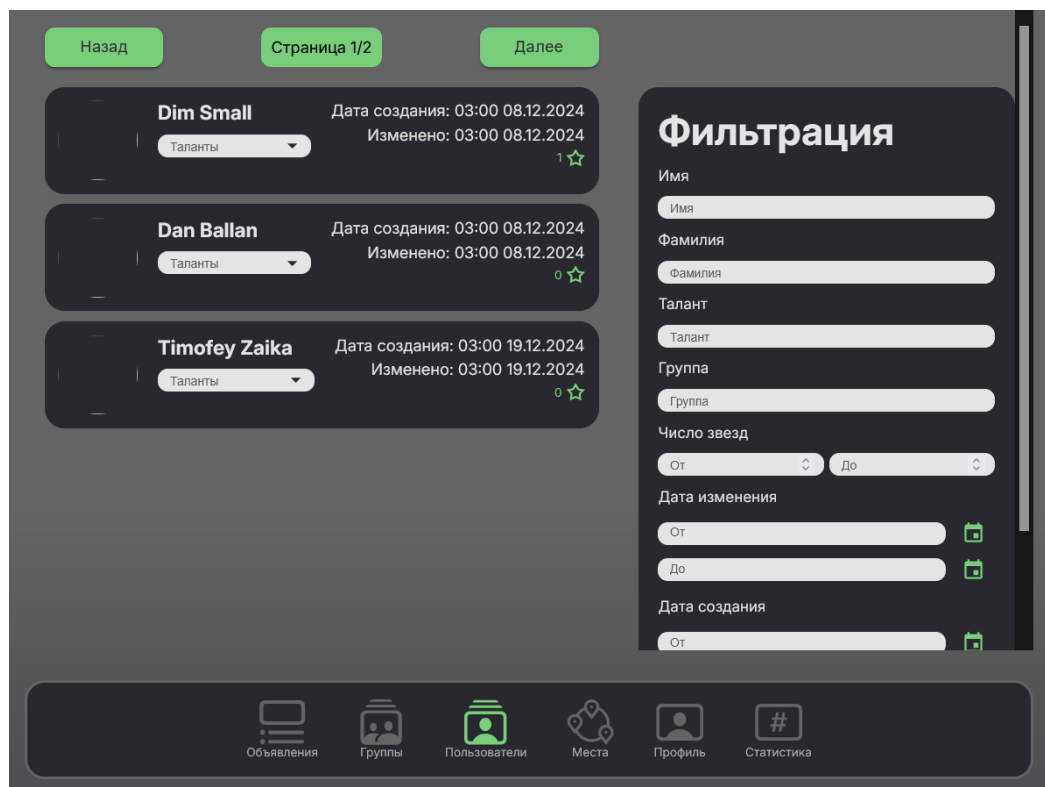


Рисунок 8 – страница списка пользователей

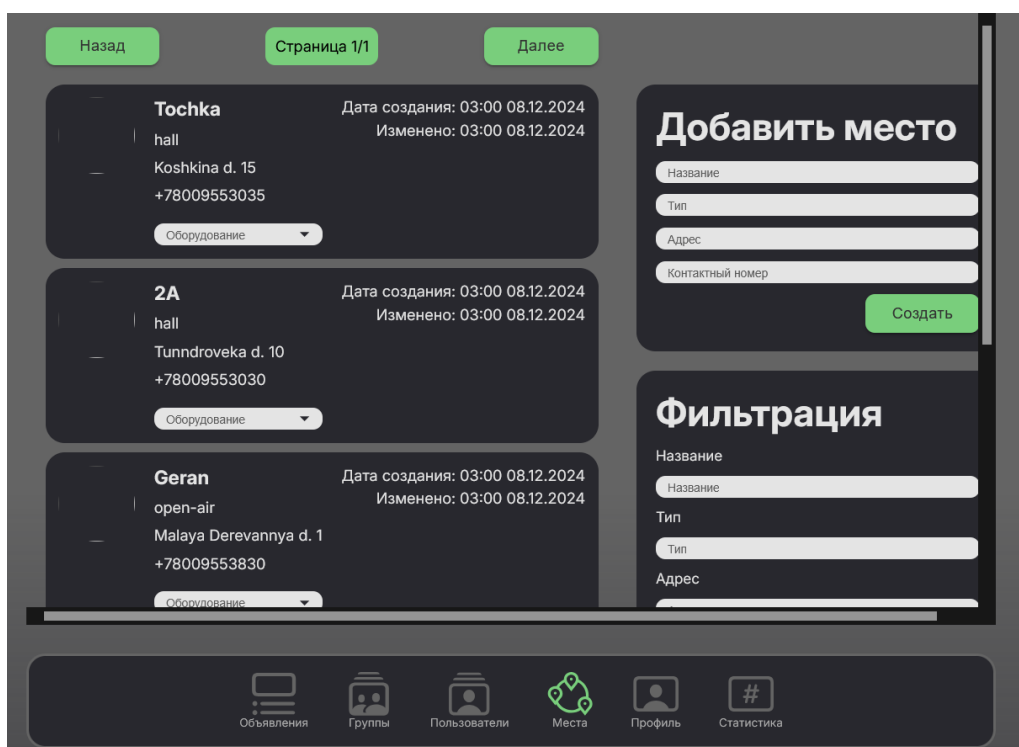


Рисунок 9 – страница списка мест

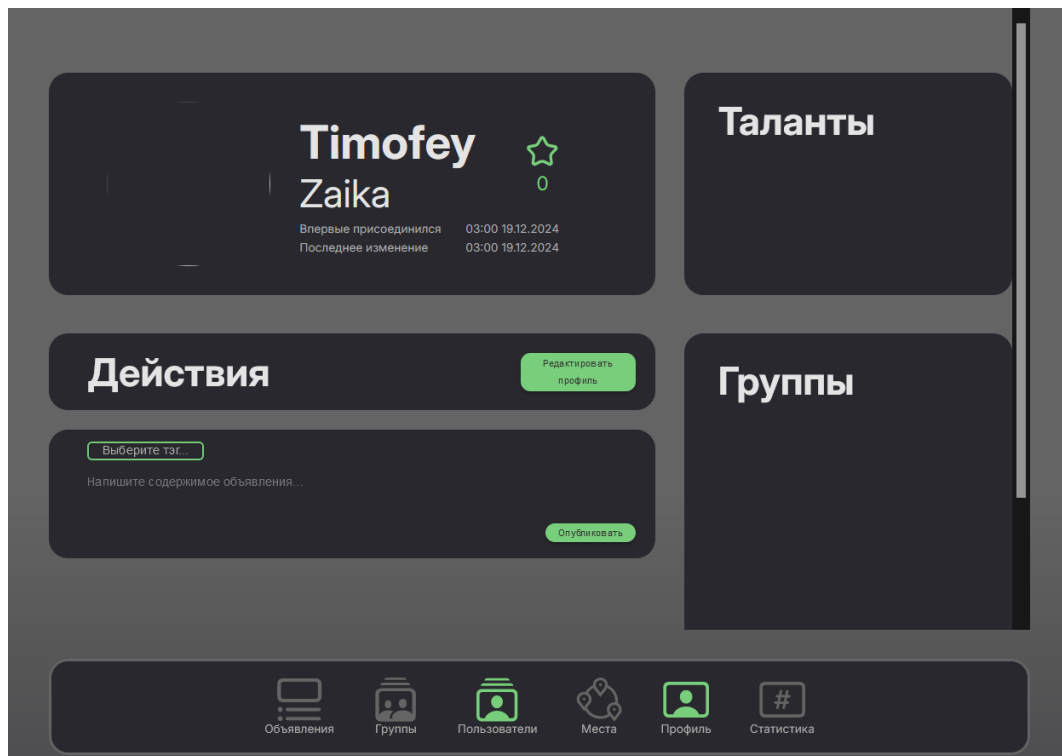


Рисунок 10 – страница профиля

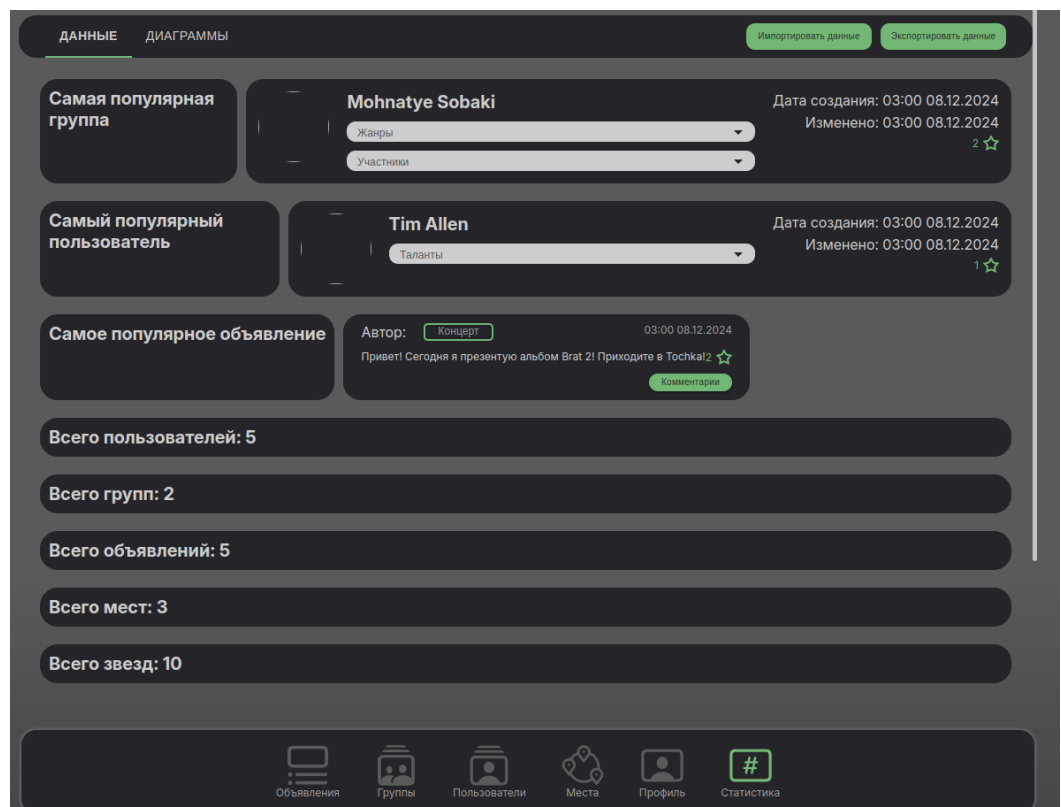


Рисунок 11 – страница статистики с данными

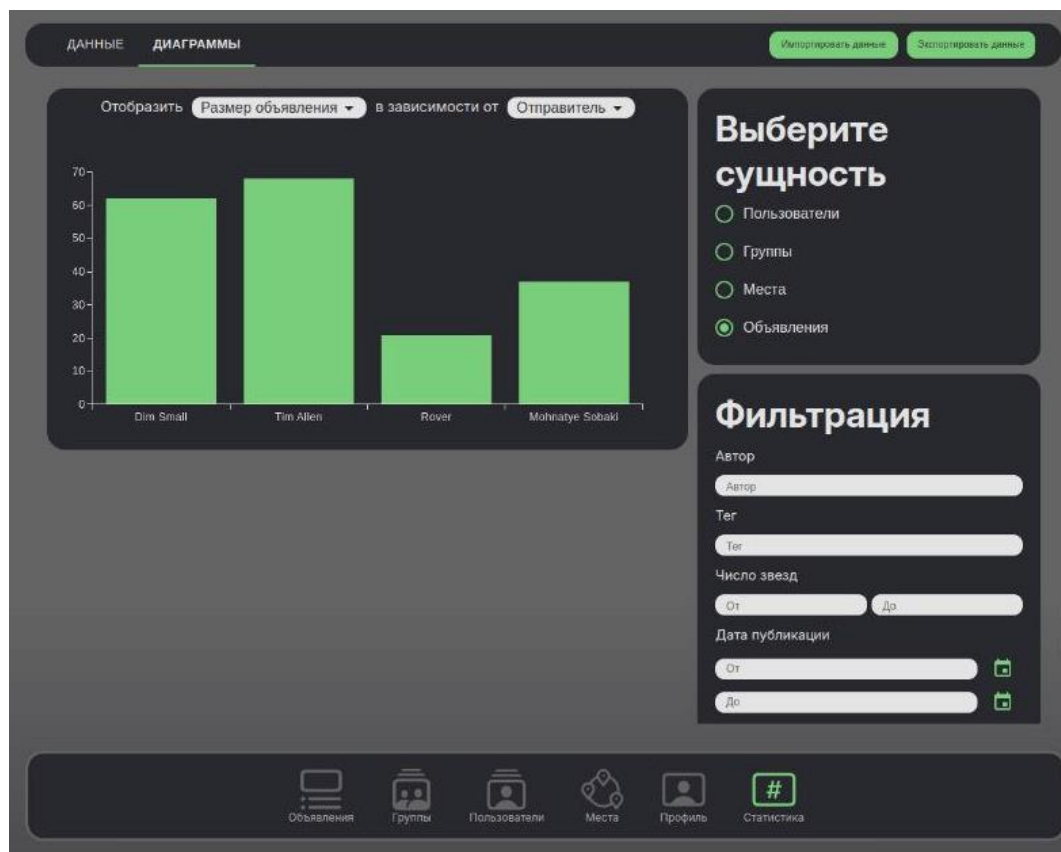


Рисунок 12 – страница статистики с диаграммами

5. ВЫВОДЫ

5.1. Достигнутые результаты

Успешно разработан веб-сервис, где инди-группы смогут дружить и сотрудничать. Реализованы пользовательские интерфейсы групп, музыкантов, реп. точек, объявлений, комментариев, рейтинга и статистики. Приложение реализует все оговоренные сценарии использования и использует нереляционную СУБД ArangoDB.

5.2. Недостатки и пути для улучшения полученного решения

У приложения можно выделить следующие недостатки:

- Отсутствие мультимедийных данных в пользовательском интерфейсе (аватары пользователей, групп, мест).
- Отсутствие возможности выхода из профиля.

В связи с недостатками можно выделить следующие пути для улучшения полученного решения:

- Добавление мультимедийных данных.
- Предоставление возможности выхода из профиля.

5.3. Будущее развитие решения

В будущем планируется развивать решение в сторону удобства пользовательского интерфейса, а именно работы с видом и размером текста, унификацией размера отдельных блоков интерфейса. Также планируется улучшать производительность приложения путем рефакторинга кода с целью уменьшения повторного обращения к нереляционной СУБД.

ЗАКЛЮЧЕНИЕ

Разработано приложение на тему «Сервис для indie групп - поиск замен музыкантов, реп.точки, концерты» с использованием нереляционной СУБД ArangoDB. Приложение реализует все необходимые сценарии использования. Продемонстрирована эффективность применения нереляционной СУБД путем сравнения с реляционной моделью. Разработанное приложение позволяет инди-группам дружить и сотрудничать, а также предоставляет пользовательский интерфейс профилей групп, музыкантов, мест, объявлений, комментариев, рейтингов и статистике, что свидетельствует о выполнении задачи в соответствии с поставленной целью.

ЛИТЕРАТУРА

1. <https://github.com/moevm/nosql2h24-indie>
2. <https://docs.arangodb.com/stable>

ПРИЛОЖЕНИЕ А

ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

- 1 Выполнить загрузку приложения с github при помощи команды:
`git clone git@github.com:moevm/nosql2h24-indie.git`

Для корректного выполнения данной команды у пользователя должен быть доступ к репозиторию и на аккаунте в github зарегистрирован личный ssh-ключ.

- 2 Проследовать в папку с проектом и выполнить команду по сборке и запуску приложения при помощи команд
`cd nosql2h24-indie`
`docker compose up --build`

Для корректного выполнения данной команды в системе пользователя должен быть установлен Docker и Docker compose.

После выполнения данных команд приложение будет установлено и развернуто на личном компьютере пользователя.

ПРИЛОЖЕНИЕ Б

ИНСТРУКЦИЯ ДЛЯ ПОЛЬЗОВАТЕЛЯ

- 1 Для начала работы с приложением необходимо проследовать по адресу <http://localhost:3000>

Данная страница является стартовой при работе с приложением.

- 2 Далее необходимо зайти или зарегистрироваться в приложении. Если вход был произведен ранее, то вместо страниц входа и регистрации будет представлена страница профиля пользователя.
- 3 После входа или регистрации пользователь может использовать весь функционал приложения без ограничений, используя панель навигации для переключения между разделами. Пользователю также доступна статистика приложения и возможность импорта и экспорта данных. Enjoy it!