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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: ИС Справочник образовательных организаций Санкт-Петербурга

Студент гр. 0382 _____ Афанасьев Н.С.

Студент гр. 0383 _____ Подопригора И.П.

Студент гр. 0383 _____ Пенкин М.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2023

ЗАДАНИЕ

Студенты

Афанасьев Н.С.

Подопригора И.П.

Пенкин М.В.

Группы 0382, 0383

Тема проекта: Разработка ИС справочника образовательных организаций Санкт-Петербурга.

Исходные данные:

Необходимо реализовать веб-сервис ИС справочника образовательных организаций Санкт-Петербурга.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарий использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 20.09.2023

Дата сдачи реферата: 23.12.2023

Дата защиты реферата: 23.12.2023

Студент гр. 0382		Афанасьев Н.С.
Студент гр. 0383		Подопригора И.П.
Студент гр. 0383		Пенкин М.В.
Преподаватель		Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была сформулирована и утверждена индивидуальная тема – разработка ИС справочника образовательных организаций Санкт-Петербурга. Во внимание будут приниматься такие аспекты как производительность и удобство разработки. Найти исходный код и всю дополнительную информацию можно по ссылке: https://github.com/moevm/nosql2h23-teaching

ANNOTATION

As part of this course, it was supposed to develop an application in a team on one of the given topics. An individual topic was formulated and approved - the development of the IP directory of educational organizations in St. Petersburg. Aspects such as performance and ease of development will be taken into account. You can find the source code and all additional information at the link: https://github.com/moevm/nosql2h23-teaching

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

Цель работы – создать высокопроизводительное и удобное решение для веб-сервиса справочника образовательных организаций Санкт-Петербурга.

Было принято решение разработать веб-приложение, предоставляющее возможность получать статистику образовательных организаций, осуществлять поиск и просмотр конкретных учебных заведений, а также выполнять функции добавления, редактирования и удаления их данных в роли администратора.

2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение с использованием СУБД Memcached, а также с возможностью локального разворота приложения с помощью docker-compose.

3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

3.1. Макеты UI

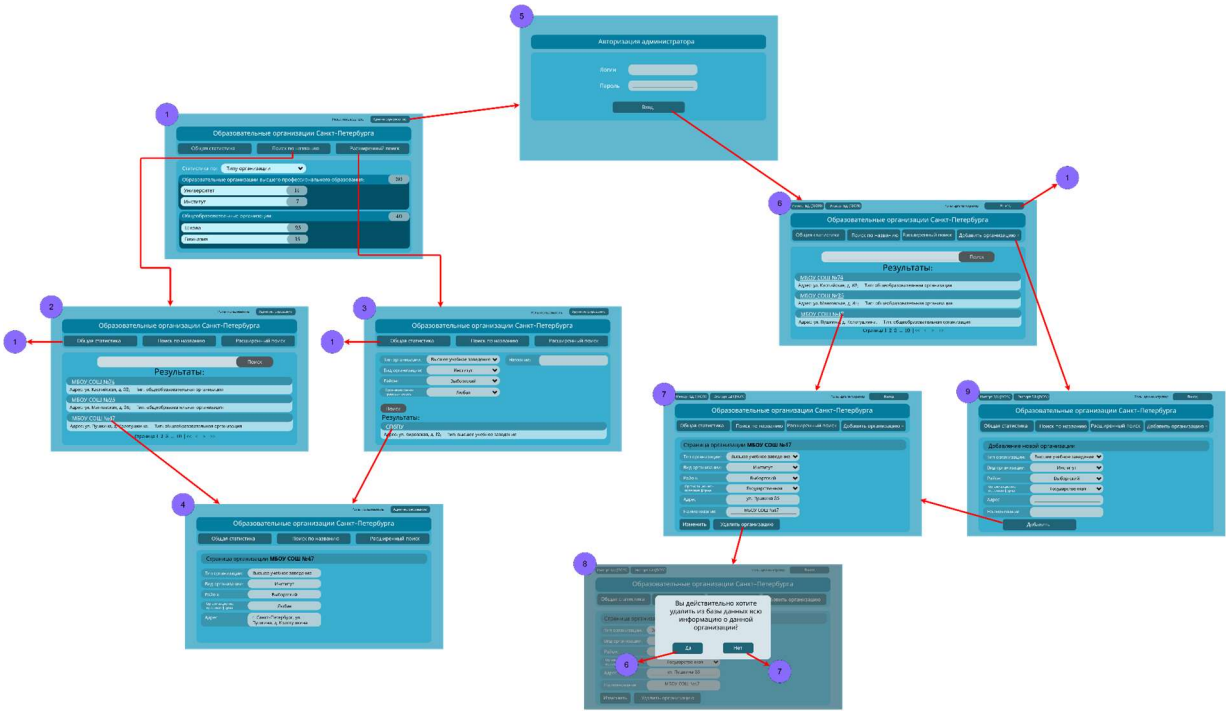

Рисунок 1 – Макет UI

3.2. Описание сценариев использования

3.2.1. Сценарий использования - «Общая статистика образовательных организаций»:

Действующее лицо: Пользователь

- при заходе на сайт пользователь попадает на эту страницу(стр. №1)
- пользователь выбирает фильтр, по которому будет выдана статистика
- пользователь может перейти на нее нажатием кнопки "Общая статистика" с любой страницы приложения

3.2.2. Сценарий использования - «Поиск по названию образовательной организации»:

Действующее лицо: Пользователь

- пользователь нажимает кнопку "Поиск по названию"
- попадает на страницу поиска организации по названию(стр. №2)
- вводит полное или частичное название искомой образовательной организации
- нажимает кнопку "Поиск"
- получает список найденных организаций

3.2.3. Сценарий использования - «Расширенный поиск по названию образовательной организации»:

Действующее лицо: Пользователь

- пользователь нажимает кнопку "Расширенный поиск"
- попадает на страницу расширенного поиска организации(стр. №3)
- выбирает параметры образовательной организации: тип организации, вид организации, район, название и т.д.
- нажимает кнопку "Поиск"

3.2.4. Сценарий использования - «Переход на страницу организации»:

Действующее лицо: Пользователь

- пользователь нажимает на определенную организацию из результатов поиска(по названию или расширенному)
- переходит страницу выбранной организации(стр №4)

3.2.5. Сценарий использования - «Авторизация администратора»:

Действующее лицо: Администратор

- в начале администратор нажимает кнопку "Администрирование" с любой страницы с ролью пользователя
- проходит окно авторизации(стр. №5)
- попадает на главную страницу(стр. №6) в режиме администрирования(роль: Администратор)

3.2.6. Сценарий использования - «Добавление образовательной организации»:

Действующее лицо: Администратор

- администратор нажимает на кнопку "Добавить организацию"
- попадает на страницу добавления образовательной организации(стр. №9)
- вводит параметры добавляемой организации: тип организации, вид организации, район, название и т.д.
- нажимает кнопку "Добавить"
- попадает на страницу созданной образовательной организации(стр. №7)

3.2.7. Сценарий использования - «Изменение образовательной организации»:

Действующее лицо: Администратор

- администратор ищет организацию подобно пользователю
- переходит на страницу образовательной организации(стр. №7)
- нажимает кнопку "Изменить"

3.2.8. Сценарий использования - «Удаление образовательной организации»:

Действующее лицо: Администратор

- администратор ищет организацию подобно пользователю
- переходит на страницу образовательной организации(стр. №7)
- нажимает кнопку "Удалить"
- подтверждает удаление в модальном окне(стр. №8)

3.2.9. Сценарий использования - «Массовый импорт/экспорт»:

Действующее лицо: Администратор

- администратор нажимает соответствующую кнопку в левом верхнем углу экрана для массового импорта/экспорта
- открывается файловое диалоговое окно, в котором администратор выбирает файл в который будет экспортирована БД / из которого будет импортирована БД

4. МОДЕЛЬ ДАННЫХ

4.1. Нереляционная модель данных

4.1.1. Описание модели

Так как Memcached - это БД типа ключ-значение, то все данные должны быть представлены в виде словаря. Исходные данные имеют табличную форму (для всех организаций большое количество одинаковых полей), поэтому в качестве ключей используются сложные наименования вида "[поле]:[id]" (например, по ключу "name:233" можно получить наименование организации с ID 233). В качестве ключа используется ОГРН (основной государственный регистрационный номер), который гарантированно существует и различается для каждой организации. Для упрощения фильтрации и поиска созданы несколько специализированных ключей:

- Ключ **"ids"** хранит сериализированный массив ОГРН всех организаций
- Ключ **"opf_type:[i]"** хранит сериализированный массив ОГРН организаций имеющих ОПФ с порядковым номером i (для обеспечения быстродействия описания ENUM-ов будут хранится на сервере, а не в БД). Например, по ключу "opf_type:2" можно получить список всех муниципальных образований.
- Ключи **"org_type:[i]"**, **"org_sub_type:[i]"**, **"org_category:[i]"**, **"org_location:[i]"** хранят аналогичную информацию.

4.1.2. Оценка объема информации, хранимой в модели.

Информация по каждой организации хранится в парах ключ-значение, для соответствующих ключей выделяется места в байт для значения: number:[id] - 4, name:[id] - 512, short_name:[id] - 256, adress:[id] - 512, fias[id] - 36, mngr_pos:[id] - 256, mngr_name:[id] - 256, opf:[id] - 1, type:[id] - 1, sybtype:[id]

- 1, category:[id] - 1, tel:[id] - 15, location:[id] - 1, e_diary:[id] - 1, website:[id] - 60, email:[id] - 60, в сумме получается 1973 байт на хранение информации об одной организации и плюс к этому 322 байт занимают ключи, получаем 2295.

По ключам ids, opf_type:[n_1], org_type:[n_2], org_subtype:[n_3], org_category:[n_4], org_location:[n_5], хранятся массивы с id(ОГРН) организаций, 13+1 байт на id и разделитель в массиве, итого значения по перечисленным выше ключам занимают 6 * 14n = 84n байт, где n - число организаций в БД, ключи занимают 64 байт.

Объем всей БД в байт: 2295n + 84n + 64 = 2379n + 64, где n - число организаций в БД.

4.1.3. Избыточность модели и направление роста

Чистый объем данных: 1973n Объем БД: 2379n + 64 Объем БД примерно в 1.2 раза больше чистого объема данных. Увеличение объема модели линейно зависит от количества добавляемых организаций.

4.1.4. Графическое представление модели

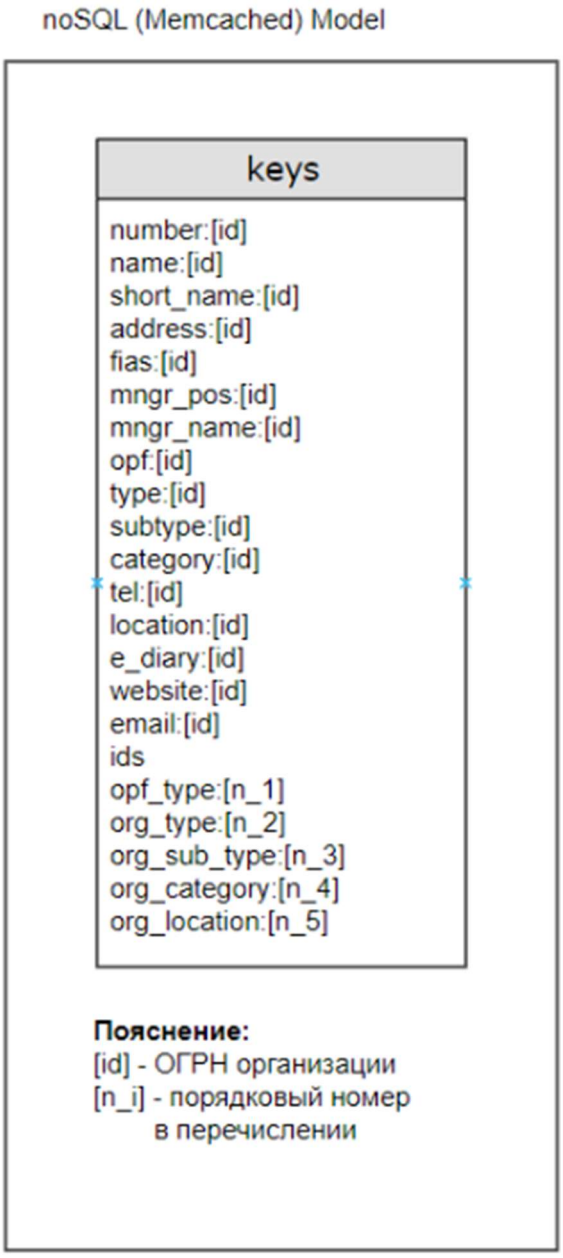

Рисунок 3 - Модель нереляционной БД.

4.1.5 Примеры запросов

Инструменты работы Memcached весьма ограничены, поэтому внутри БД можно выполнять только простые операции. В примерах ниже идёт работа с организацией ОГРН которого - 1000000000000.

- Получение полной информации об организации:

```
get number:1000000000000
```

```
get name:1000000000000
get short_name:1000000000000
get address:1000000000000
get fias:1000000000000
get mngr_pos:1000000000000
get mngr_name:1000000000000
get opf:1000000000000
get type:1000000000000
get subtype:1000000000000
get category:1000000000000
get tel:1000000000000
get location:1000000000000
get e_diary:1000000000000
get website:1000000000000
get email:1000000000000
```

Тип: получение данных, количество запросов: 16

- Добавление новой организации:

```
set number:1000000000000 0 0 3
123
set name:1000000000000 0 0 40
Учебное заведение №123
set short_name:1000000000000 0 0 11
УЗ №123
set address:1000000000000 0 0 33
г. Санкт-Петербург
set fias:1000000000000 0 0 1
-
set mngr_pos:1000000000000 0 0 20
Заведующий
```

```
set mngr_name:1000000000000 0 0 38
Иванов Иван Иванович
set opf:1000000000000 0 0 1
1
set type:1000000000000 0 0 1
1
set subtype:1000000000000 0 0 1
1
set category:1000000000000 0 0 1
1
set tel:1000000000000 0 0 14
+7 777 7777777
set location:1000000000000 0 0 1
1
set e_diary:1000000000000 0 0 1
1
set website:1000000000000 0 0 15
www.example.com
set email:1000000000000 0 0 17
example@gmail.com
```

Тип: добавление данных, количество запросов: 16

- Изменение данных об организации:

```
set website:1000000000000 0 0 16
www.example1.com
set email:1000000000000 0 0 18
example1@gmail.com
```

Тип: изменение данных, количество запросов: m, где m - количество изменяемых полей

- Удаление организации

```
delete number:1000000000000
delete name:1000000000000
delete short_name:1000000000000
delete address:1000000000000
delete fias:1000000000000
delete mngr_pos:1000000000000
delete mngr_name:1000000000000
delete opf:1000000000000
delete type:1000000000000
delete subtype:1000000000000
delete category:1000000000000
delete tel:1000000000000
delete location:1000000000000
delete ogrn:1000000000000
delete e_diary:1000000000000
delete website:1000000000000
delete email:1000000000000
```

Тип: удаление данных, количество запросов: 16

- Получение списка ОГРН организаций в определённом районе:

```
get org_location:3
```

Тип: получение данных, количество запросов: 1

4.2. Реляционная модель данных

4.2.1. Описание сущностей и типов данных

Модель состоит из одной сущности **organization**, которая хранит информацию об организации. В качестве ключа используется ОГРН (основной государственный регистрационный номер), который гарантированно

существует и различается для каждой организации. Так как некоторые поля содержат значения из определённого списка, то для уменьшения используемой памяти и предотвращения ошибок были использованы перечисления (ENUM). В этой модели, в отличие от модели Memcached, отсутствует информация о статистике, так как её проще получить через sql-запросы.

Описание типов данных:

INT (4 байт) - число

BIGINT (8 байт) - большое число

VARCHAR (1 байт на символ) - строка

BOOL (1 байт) - булево значение

ENUM (1 байт) - перечисление

4.2.2. Оценка объема информации, хранимой в модели

Посчитаем для одной организации: 8(ogrn) + 4(number) + 2559(все строки) + 15(все Enum) + 1(bool e_diary) = 2313 байт. В итоге получим 2313*N байт (N - количество организаций).

4.2.3. Избыточность модели и направление роста

В модели отсутствуют избыточные связи между таблицами, поскольку она представлена всего одной таблицей. Значительное преимущество также заключается в том, что первичный ключ не является избыточным элементом, поскольку содержит реальный идентификатор организации из реальной жизни, который, к тому же, отображается на веб-сайте.

Так как сущность одна, а именно таблица "Организации", рост модели будет пропорционален количеству организаций.

4.2.4. Пример запросов

- Получение полной информации об организации:

```
SELECT * FROM organization WHERE ogrn =
1000000000000;
```

Тип: получение данных, количество запросов: 1

- Добавление новой организации:

```
INSERT INTO organization
(ogrn, number, name, short_name, address, mngr_pos,
mngr_name, opf, type, subtype, category, tel,
location, e_diary, website, email)
VALUES
(1000000000000, 123, 'Учебное заведение №123', 'УЗ
№123', 'г. Санкт-Петербург', 'Заведующий', 'Иванов
Иван Иванович', 1, 1, 1, 1, '+7 777 7777777', 1, 1,
'www.example.com', 'example@gmail.com');
```

Тип: добавление данных, количество запросов: 1

- Изменение данных об организации:

```
UPDATE organization SET website = 'www.example1.com',
email = 'example1@gmail.com' WHERE ogrn =
1000000000000;
```

Тип: изменение данных, количество запросов: 1

- Удаление организации:

```
DELETE FROM organization WHERE ogrn = 1000000000000;
```

Тип: удаление данных, количество запросов: 1

- Получение списка ОГРН организаций в определённом районе:
  SELECT ogrn FROM organization WHERE location = 3;
  Тип: получение данных, количество запросов: 1

4.2.5. Графическое представление модели

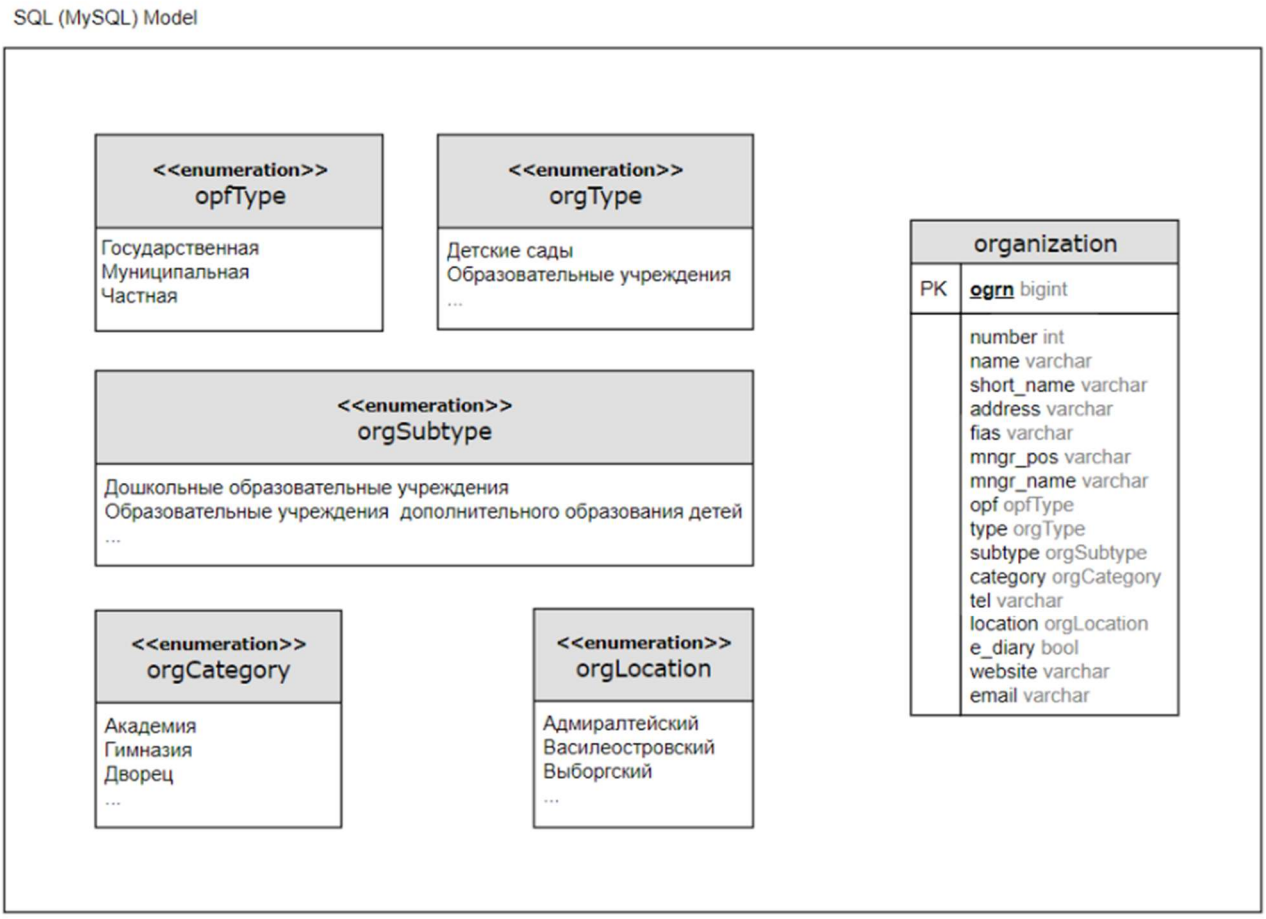

Рисунок 4 - Модель реляционной БД

4.3. Сравнение моделей

В Memcached для некоторых сценариев (например, при получении полной информации об организации, добавлении или удалении новой организации) требуется в 16 раз больше запросов, чем в MySQL, что может повлиять на скорость работы. Также количество операций для изменения

нескольких полей организации в Memcached зависит линейно от количества таких полей, в то время как в MySQL такой проблемы нет.

Memcached: 2379n + 64 байт (где n - число организаций в БД). MySQL: 2313n байт. MySQL занимает меньший объем памяти по сравнению с Memcached.

4.4. Вывод

С точки зрения удельного объема информации SQL является более эффективным выбором, требуя меньше оперативной памяти по сравнению с Memcached. Кроме того, в Memcached необходимо значительно больше запросов для выполнения определенных действий, что может повлиять на скорость работы.

Таким образом, на основании удельного объема информации и количества запросов по отдельным юзкейсам, SQL выглядит предпочтительнее для данной задачи, обеспечивая более эффективное использование ресурсов и более высокую производительность.

5. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

5.1. Краткое описание

Back-end представляет из себя приложение, разработанное на языке программирования JS в среде выполнения NodeJS. В качестве серверной библиотеки использовался Express. Сервер предоставляет REST API для получения, просмотра, изменения, добавления и удаления информации об образовательных организациях. Обычный пользователь ограничен просмотром и поиском информации. Администратор может модифицировать информацию, а также экспортировать и импортировать данные БД.

Front-end основан на языке шаблонов Pug и реализован через Server-Side Rendering: пользователю предоставляется уже готовая страница со всеми стилями и данными, когда он делает запрос внутри приложения.

5.2. Использованные технологии

БД: Memcached

Back-end: NodeJS, expressJS.

Front-end: Шаблонизатор Pug.

5.3. Снимки экрана приложения

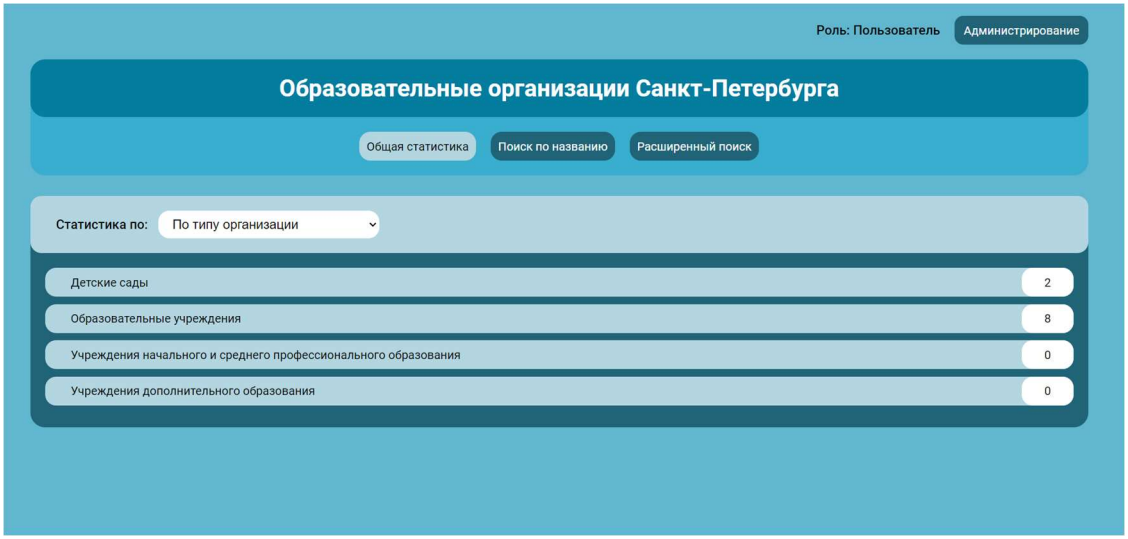

Рисунок 5 – Главная страница (Статистика)

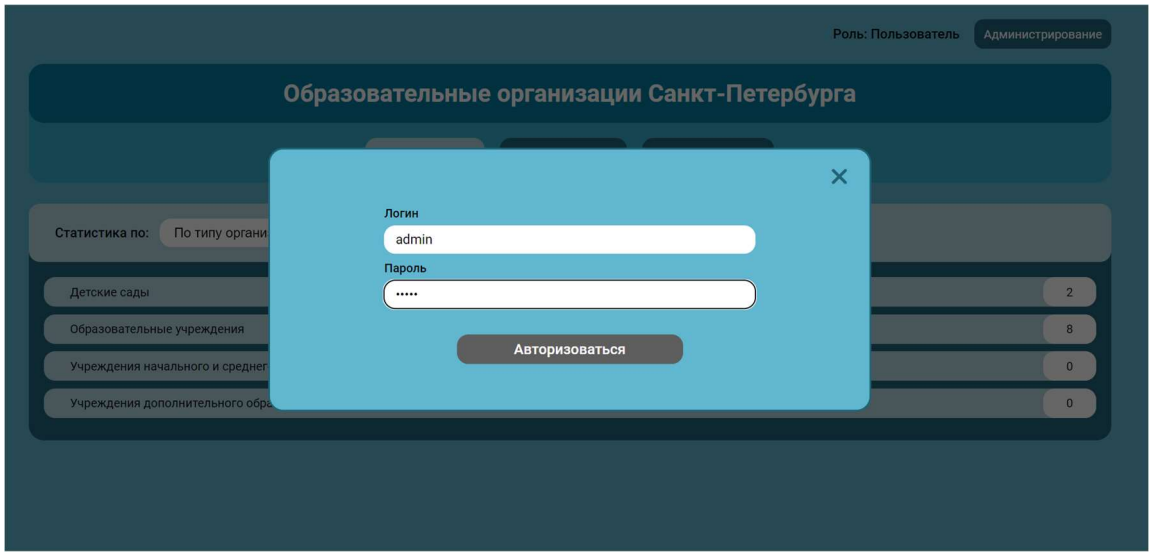

Рисунок 6 – Окно авторизации

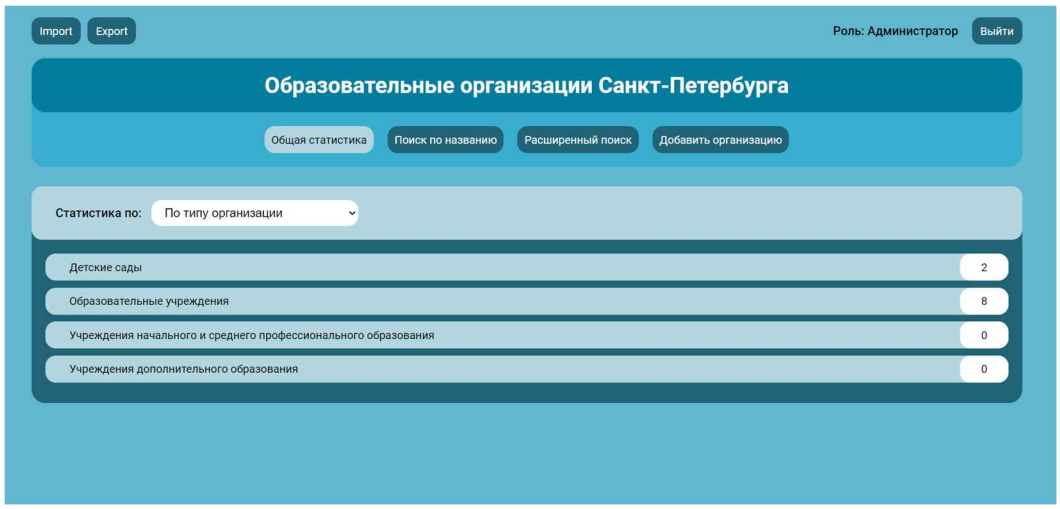

Рисунок 7 – Главная страница (Статистика), администратор

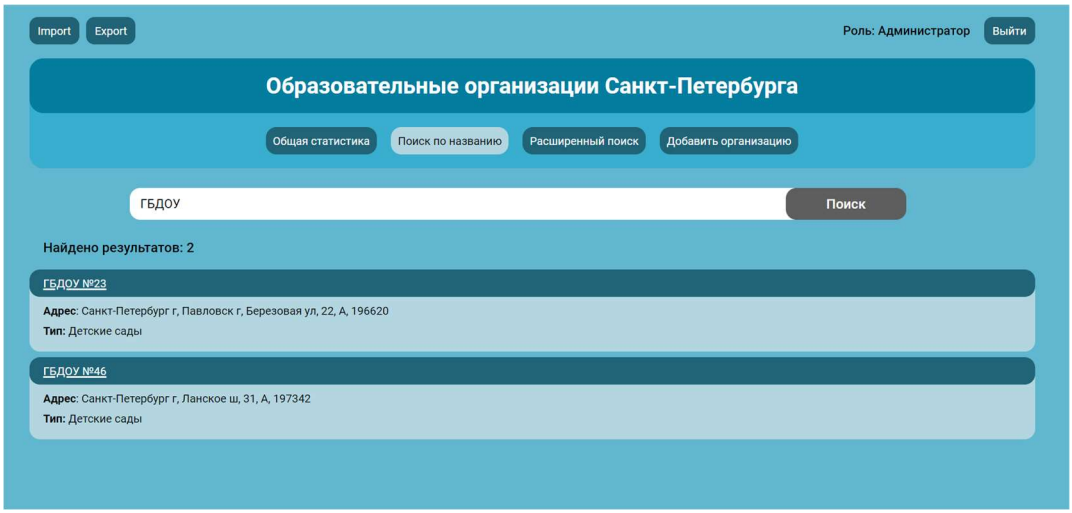

Рисунок 8 – Страница поиска по названию

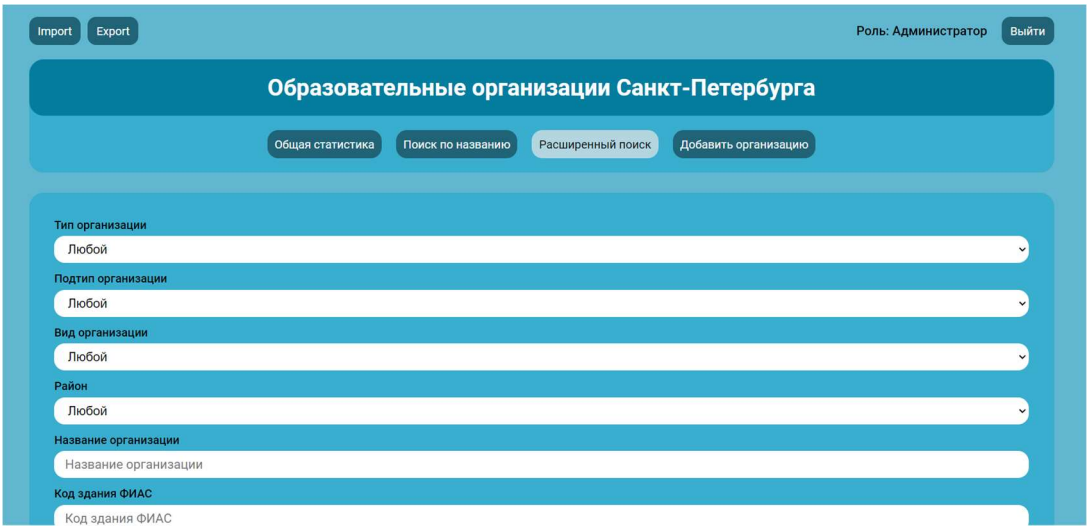

Рисунок 9 – Страница детального поиска

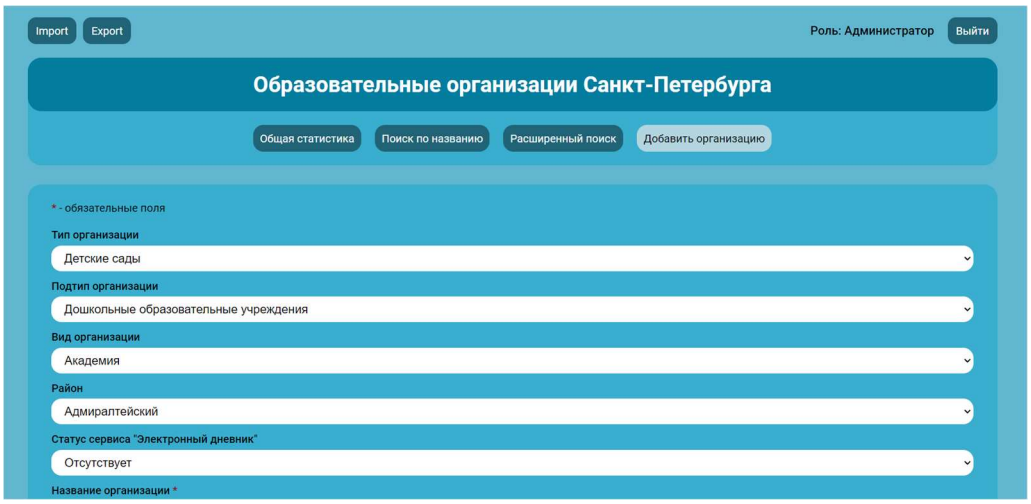

Рисунок 10 – Страница добавления организации

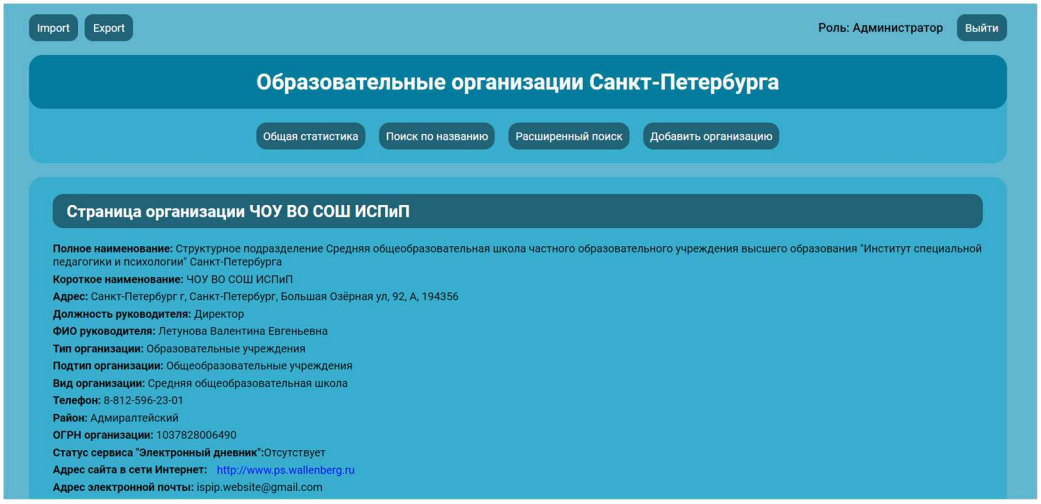

Рисунок 11 – Страница организации

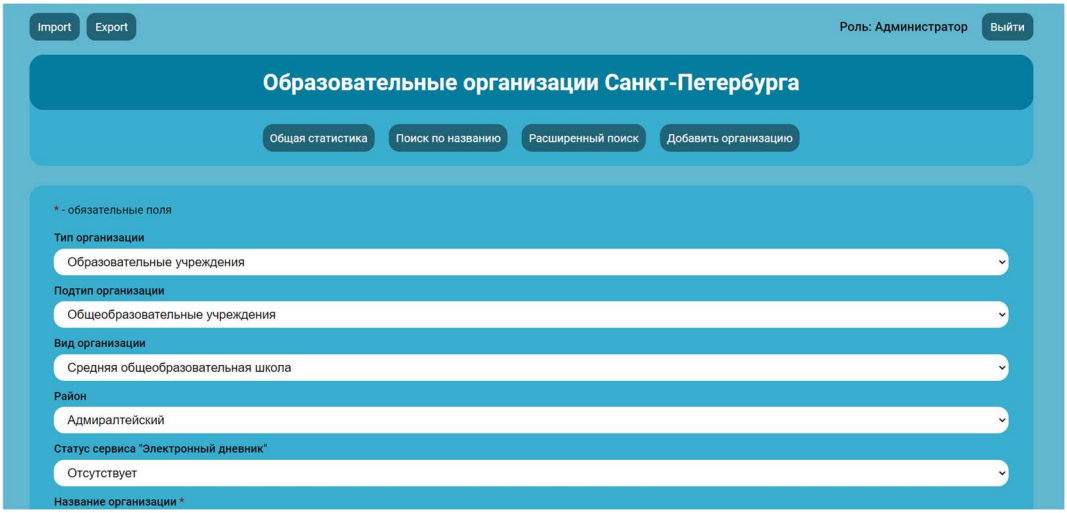

Рисунок 12 – Страница изменения организации

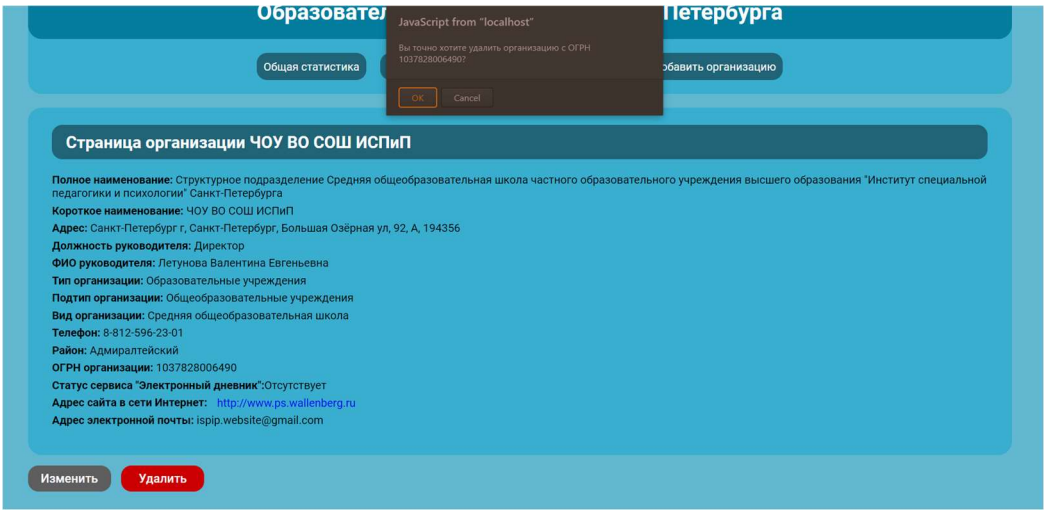

Рисунок 13 – Удаление организации

6. ВЫВОД

6.1. Достигнутые результаты

В ходе работы было разработано приложение веб-сервиса справочника образовательных организаций Санкт-Петербурга с использованием нереляционной базы данных Memcached. Была реализована вся заявленная функциональность, а также добавлена возможностью локального разворота приложения с помощью docker-compose.

6.2. Недостатки и пути для улучшения полученного решения

Некоторые запросы к БД с сервера реализованы недостаточно оптимально: местами можно ограничиться меньшим количеством запросов, а также отправлять некоторые группы запросов не по отдельности, а синхронно. Такие изменения требует дальнейшего тщательного анализа.

6.3. Будущее развитие решения

Планируется улучшение front-end вёрстки, улучшение обращения к БД, а также рефакторинг кода.

7. ПРИЛОЖЕНИЯ

Документация по сборке и развертыванию приложения

1. Склонировать репозиторий (указан в списке литературы)
2. Зайти в папку ./nosql2h23-teaching/
3. Запустить локальный разворот приложения через docker-compose с помощью следующих команд:
   - sudo docker-compose build --no-cache
   - sudo docker-compose up –d
4. Перейти в любом удобном браузере по адресу: 127.0.0.1:3000

8. ЛИТЕРАТУРА

1. Ссылка на github проекта: https://github.com/moevm/nosql2h23-teaching/
2. Документация Memcached: https://memcached.org
3. Документация библиотеки memcached для NodeJS: https://www.npmjs.com/package/memcached
4. База данных образовательных организаций Санкт-Петербурга: https://petersburgedu.ru/institution