МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(МОСКОВСКИЙ ПОЛИТЕХ)

Факультет информационных технологий
Кафедра «Инфокогнитивные технологии»

КУРСОВОЙ ПРОЕКТ

на тему: *«Разработка онлайн-сервиса для оценки плотности инфраструктуры города Москва»*
Направление подготовки 09.03.03 «Прикладная информатика»
Профиль «Корпоративные информационные системы»

Выполнил:

студент группы 221-361

Конопский Кирилл Сергеевич

19.01.2024

_____
(подпись)

Москва 2023

Введение

В условиях стремительного городского развития мы сталкиваемся с необходимостью более глубокого исследования инфраструктуры мегаполисов, таких как Москва. Городская инфраструктура выступает важным элементом формирования комфортной среды для жителей, определяя качество жизни и уровень развития общества. Однако, существующие методы анализа инфраструктуры зачастую ограничены и не позволяют полноценно оценить её структуру

Следовательно, возникает актуальная проблема необходимости более точных и современных методов оценки городской инфраструктуры. Существующие методы, такие как анализ карт и статистические данные, часто оказываются недостаточно детализированными и не способны выявить сложные взаимосвязи между объектами инфраструктуры. Отсутствие точной оценки плотности инфраструктуры может привести к неоптимальному использованию ресурсов, неудовлетворенности потребителей и неправильному городскому планированию.

На текущий момент существует ограниченное количество онлайн-ресурсов для оценки городской инфраструктуры. Однако, многие из них ограничиваются базовой визуализацией данных, не предоставляя глубокого анализа и структурированной информации. Существующие способы оценки плотности инфраструктуры не всегда учитывают пространственные взаимосвязи и динамику развития города, что снижает их эффективность.

Целью данной курсовой работы является разработка инновационного онлайн-сервиса, базирующегося на передовых алгоритмах, способных провести глубокий анализ инфраструктуры города Москвы. Разрабатываемый сервис будет учитывать пространственные зависимости, выявлять кластеры и выбросы в структуре города, а также предоставлять интерактивные инструменты для визуализации данных. Такой подход позволит существенно улучшить оценку плотности инфраструктуры и предоставить ценные инструменты для принятия обоснованных решений в области городского планирования и развития.

2. Цель и Задачи работы

Цель работы заключается в разработке онлайн-сервиса для оценки плотности инфраструктуры города Москва с применением алгоритма DBSCAN, предоставляя пользователям возможность проанализировать структуры городской среды и наглядно просмотреть плотность инфраструктуры по средству карты.

Основные задачи выделенные в ходе работы:

1. Провести анализ используемых датасетов: В первую очередь необходимо изучить все атрибуты используемых датасетов. Это поможет понять некоторые потребности для алгоритма кластеризации и работы сайта;
2. Разработать макет дизайна сайта на платформе «Figma» и по нему сверстать основные страницы: Разработка макета поможет в гармоничной верстке основных страниц сайтов и выстраивании логики интернет-ресурса;
3. Разработать базу данных: Необходимо разработать базу данных на основе предметной области;
4. Разработать серверную часть сайта: Необходимо разработать серверную часть приложения. Необходимо обеспечить безопасность передачи данных, по средству параметризированных запросов, и эффективное управление информацией;
5. Разработать процедуру для кластеризации по алгоритму DBSCAN: Необходимо разработать эффективный алгоритм кластеризации по алгоритму DBSCAN используя процедуры на MySQL;
6. Добавить интерактивность на сайт: Необходимо реализовать Ajax запросы и подключить API Яндекс Карт, для отображения выделенных кластеров;

В работе использовано 5 датасетов общей размерности в 21 225 записей, ниже приведены ссылки на используемые датасеты:

1. Общественное питание в Москве (https://data.mos.ru/opendata/1903): Набор данных позволяет получить подробную информацию о предприятиях общественного питания в городе Москве в части наименования, точного адреса, специализации, уточнить телефон, график работы и местоположение объектов на карте города;
2. Поликлиники детские (https://data.mos.ru/opendata/505);
3. Поликлиническая помощь взрослым (https://data.mos.ru/opendata/503);
4. Больницы детские и специализированные (https://data.mos.ru/opendata/502);
5. Стоматологические поликлиники детские (https://data.mos.ru/opendata/506);

При разработке использовались технологии такие как Python фреймворк Django используемый для серверной части сайта, далее физическая модель базы данных создана на СУБД MySQL, для верстки и интерактивности страниц сайта использовались технологии HTML\CSS, BootStrap, JavaScript JQuery, API Яндекс Карт. Для создания макета страниц использован сервис Figma.

3. Проектирование приложения

В ходе работы планируется разработать такие функциональные возможности как авторизация, регистрация и отображение профиля пользователя. Будет реализована визуализация и интерактивность по средствам добавления таблицы с выборкой данных и карты для отображения кластеров или же выбросов, и других визуальных элементов, например, уведомления о действиях пользователя. На страницах визуализации данных планируется возможность индивидуального переименования технических числовых меток на пользовательские (доступно только авторизированному пользователю).

При проектировании веб-приложения использовалась модель MVT (Model-View-Templates), поэтому для всех страниц будет реализован общий шаблон.

Интерфейс веб-приложения будет состоять из 6 страниц: главная страница, страница исследований, карта, профиль, авторизация и регистрация. Ниже представлена карта сайта (рис. 1).

Рисунок 1 – Карта сайта

Как мы видим из карты сайта из главной страницы мы сможем перейти почти на все страницы, и страница регистрации будет доступна только из авторизации. Так же можно отметить что из каждой страницы можно перейти на страницы «Исследования», «Карта», «Профиль» или «Авторизация» смотря какой пользователь, авторизированный или нет

Структура приложения построена на основе шаблона проектирования MVT используя средства фреймворка Django. Веб-приложение будет состоять из двух приложений (модулей): «account», «clusters». Так же можно отметить

что для взаимодействия с базой данных используется встроенная ORM Django, по средствам моделей, которые отражают в себе таблицы базы данных.

Модуль «account» - отвечает за управление пользователями, авторизацию, регистрацию и профили.

Модуль «clusters» - отвечает за обновление данных в базе данных по средствам API сайта data.mos.ru и для отправки данных на фрон для визуализации.

Рассмотрим файловую структуру проекта: в каждом модуле есть файлы «url.py» для хранения URL-паттернов, «views.py» для хранения функций используемых при загрузке страниц, папка «models» для хранения моделей используемых для работы с базой данных. HTML страницы хранятся в папке «templates», а статические файлы такие как JavaScript, CSS или картинки хранятся в папке «static»

Далее рассмотрим базу данных, в общей сложности она состоит из 13 таблиц (общая диаграмма базы данных представлена на рис. 2):

1. «auth_permission», «django_migrations», «auth_user_user_permision», «django_session», «django_content_type»: Таблицы, необходимые для стабильной работы Django-приложения.
2. «cluster_name», «cluster_name_increased»: Хранят имена кластеров и выбросов для каждого пользователя.
3. «clusters_food», «clusters_polyclinic»: Хранят данные из датасетов о точках общественного питания и поликлиниках\больницах.
4. «clusters_bundles», «neighbor»: Таблицы, используемые для процедуры кластеризации, таблица clusters_bundles необходима для соединения нескольких датасетов для кластеризации по координатам.
5. «django_apschedule_djangojob»,«django_apschedule_djangojobexecution»: Хранят информацию о задачах для обновления датасетов через API.
6. «auth_user»: Хранит информацию о пользователе и о том когда он был в последний раз авторизирован.

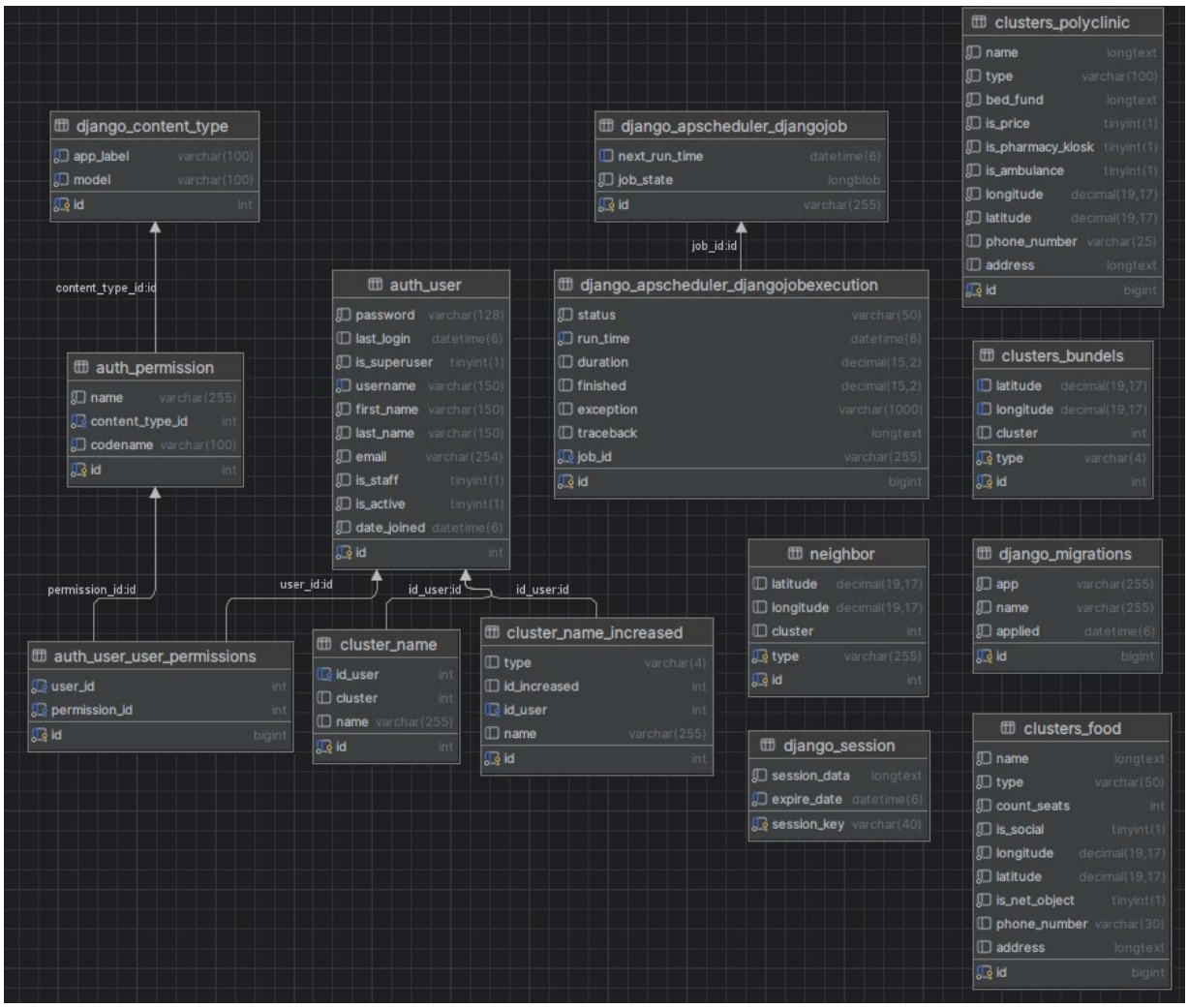

Рисунок 2 – Общая схема базы данных

Рассмотрим некоторые таблицы более подробно:

Таблица «clusters_food» хранит информацию из датасета о точках общественного питания. Хранятся такие данные как name (имя точки), type (тип точки питания), count_seats (количество сидячих мест), is_social (имеется ли социальная льгота), longitude и latitude (широта и долгота), is_net_object (сетевой ли объект), phone_number (номер телефона), address (адрес).

Таблица «clusters_polyclinic» хранит информацию из датасетов о поликлиниках или больницах. Хранятся такие данные как name (имя точки), type (тип точки питания), bed_found (количество коек), is_price (имеется ли платные услуги), longitude и latitude (широта и долгота), is_pharmacy_kiosk (есть ли аптечный киоск),), is_ambulance (наличие скорой помощи), phone_number (номер телефона), address (адрес).

Таблица «cluster_name» хранит в себе названия кластеров каждого пользователя, имеет такие атрибуты как id (идентификатор), id_user

(идентификатор пользователя который изменил название кластера), cluster (номер кластера у которого сменили название), name (название кластера), имеется связь один ко многим к таблице пользователей auth_user (один) – cluster_name (многим)

Таблица «cluster_name_increased» практически такая же как и «cluster_name», только она хранит значения именя каждого выброса у каждого пользователя.

Таблица «neighbor» является вспомогающей таблицей для работы алгоритма кластеризации DBSCAN, для сохранения соседей которых необходимо проверить на принадлежность к рассмотренному кластеру, подробнее в пункте «Реализация приложения».

Таблица «django_apscheduler_djangojob» предназначена для сохранения созданных в коде задач, для того чтобы они продолжали свою работу даже после перезагрузки сервера.

Таблица «django_apscheduler_djangojobexecution» предназначена для хранения истории вызова задач из таблицы «django_apscheduler_djangojob» и имеется связь один ко многим с таблицей «django_apscheduler_djangojob» (один).

4. Реализация приложения

Основополагающий аспект данной работы - это процедура кластеризации по алгоритму DBSCAN, рассмотрим подробнее реализацию данной процедуры. Для начала мы выбираем любую точку из таблицы и рассматриваем ее соседей в приделах указного радиуса eps (рис. 3)

```
DELETE
FROM neighbor
WHERE 1;
IF (SELECT cluster FROM clusters_bundels WHERE id = _id and type = _type) is null THEN

    INSERT INTO neighbor (type, id, latitude, longitude, cluster)
    SELECT c1.type, c1.id, c1.latitude, c1.longitude, cluster
    FROM clusters_bundels c1
    WHERE ((type, id) != (_type, _id)) and  SQRT(POW(c1.latitude - _latitude, 2) + POW(c1.longitude - _longitude, 2)) <= eps;

    IF (SELECT COUNT(*) FROM neighbor) >= minPts THEN
        SET current_cluster := current_cluster + 1;
        UPDATE clusters_bundels
        SET cluster = current_cluster
        WHERE _type = type
          and _id = id;
```

Рисунок 3 – Блок кода процедуры кластеризации

Далее проверяем чтобы количество найденных соседей было больше или равно заданному минимальному числу соседей (minPts), если оно проходит проверку, то точка становиться базовой и образует кластер, потом рассматриваем всех соседей базовой точки, если соседи этой точки тоже являются базовыми точками, то присваиваем им нынешний кластер и заносим соседей этой точки в таблицу точек которые необходимо просмотреть, если же у рассматриваемой точки меньше минимального количества соседей, то значит эта точка является граничной, ей присваивается нынешний кластер, но ее соседей мы уже в не заносим в пул точек которых необходимо рассмотреть. Так мы проходим по пока не закончиться пул соседей, и потом уже дальше ищем базовую точку, которая будет основополагающей для нового кластера, точки которым не присвоился ни один будут считаться выбросами.

Еще рассмотрим построение Яндекс карт на сайте более подробно (рис. 4). Сначала мы выбираем точку где будет центрироваться карта при запуске страницы, далее мы уже создаем объект карты указывая id блока в HTML, где будет располагаться в карте и указываем степень приближения в картах

```
let center : number[]  = [55.75573294633685, 37.61887033217221];
1+ usages  new *
function init() : void  {
    let map : ymaps.Map  = new ymaps.Map('map', {
        center: center,
        zoom: 10
    });
```

Рисунок 4 – Блок кода с генерацией карты

Следующий блок кода (рис. 5) отвечает за отображение точек на карте и добавление кластеризации для того что бы при отдалении точки собирались в единый кластер. Так же мы через параметры balloonContent указываем какую информацию отображать при нажатии на саму метку на карте. И после указываем минимальное количество точек для того что бы образовать кластер.

```
function createPoint(data) : void  {
    let points : any[]  = []
    for (let i : number  = 0; i < data.length; i++) {
        let point = data[i];
        points[i] = (new ymaps.Placemark([point.latitude, point.longitude], {
            hintContent: point.name,
            balloonContentHeader: point.name,
            balloonContentBody: `${point.phone_number}<br>${point.address}`,
            balloonContentFooter: ""
        }, {
            iconColor: '#1207f5'
        }));
    }
    clusterer = new ymaps.Clusterer({minClusterSize: [20]});
    clusterer.add(points);
    map.geoObjects.add(clusterer)
}
```

Рисунок 5 – Блок кода создания точек на карте

Следующее рассмотрим, то как обновляются данные через API сайта data.mos.ru . Ниже предоставлен блок кода с обновлением данных датасетов и автоматический запуск процедуры кластеризации (рис. 6), это сделано для того что бы данные после обновления были сразу подготовлены для работы с сайтом. Далее рассмотрим по подробнее обновление датасета с точками общественного питания в Москве.

```
@staticmethod
def update_items():
    ClustersBundels.objects.all().delete()

    Food.update_food()
    Polyclinic.update_polyclinic()

    with connection.cursor() as cursor:
        cursor.execute("CALL update_bundle();")
        cursor.execute("CALL runDBScanTest(0.02, 4);")
        cursor.close()
    print("Update successful")
```

Рисунок 6 – Блок кода с обновлением данных датасета

На рисунке 7 представлен блок кода с обновлением сначала бы запрашиваем общее количество записей датасета, для того что бы получать все данные, так как данный API предоставляет только получение данных в диапазоне до 1000 записей. И потом благодаря функции «create_food» создаем объект который после подлежит загрузке в БД. После создания всех объектов датасета мы их грузим в базу данных используя процедуру bulk_create которая загружает сразу весь массив объектов на сервер базы данных.

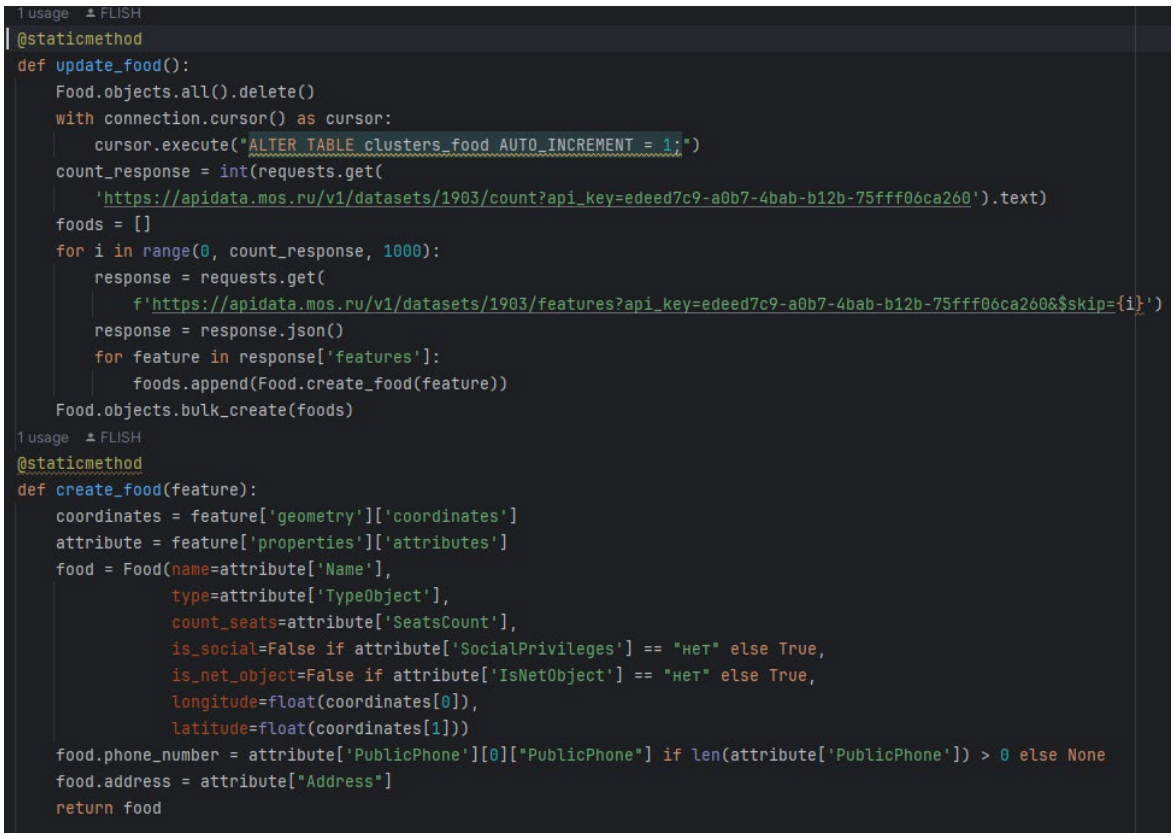

Рисунок 7 – Блок кода обновления одной таблицы

Так же обновление данных происходит по расписанию (каждые 2 дня) и через специальные средства реализован механизм восстановлении задачи при перезапуске сервера. То есть после перезапуска мы проверяем записи в базе данных на наличие задачи, если есть такая задача то мы ее запускаем, иначе создаем новую и делаем запись в БД (рис. 8).

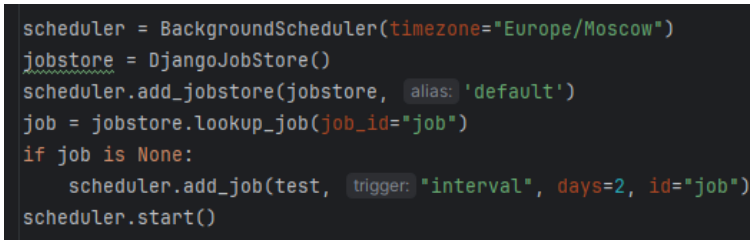

Рисунок 8 – Блок кода с создание задачи на обновление данных

5. Основные сценарии использования приложения

Первый сценарий использования приложения — это регистрация и вход в систему. Задача перед пользователем, который не имеет аккаунт, зарегистрироваться на сайте и войти в систему.

Первым шагом когда пользователь заходит на сайт он попадает на главную страницу, откуда для регистрации он должен пройти сначала на страницу авторизации нажав на кнопку «Войти» (рис. 9).

Рисунок 9 – Главная страница сайта с указание на кнопку «Войти»

Далее попадая на страницу авторизации пользователь увидит надпись «Нету аккаунта? Зарегистрироваться», где при нажатии на надпись «Зарегистрироваться» он будет перенаправлен на страницу регистрации (рис. 10)

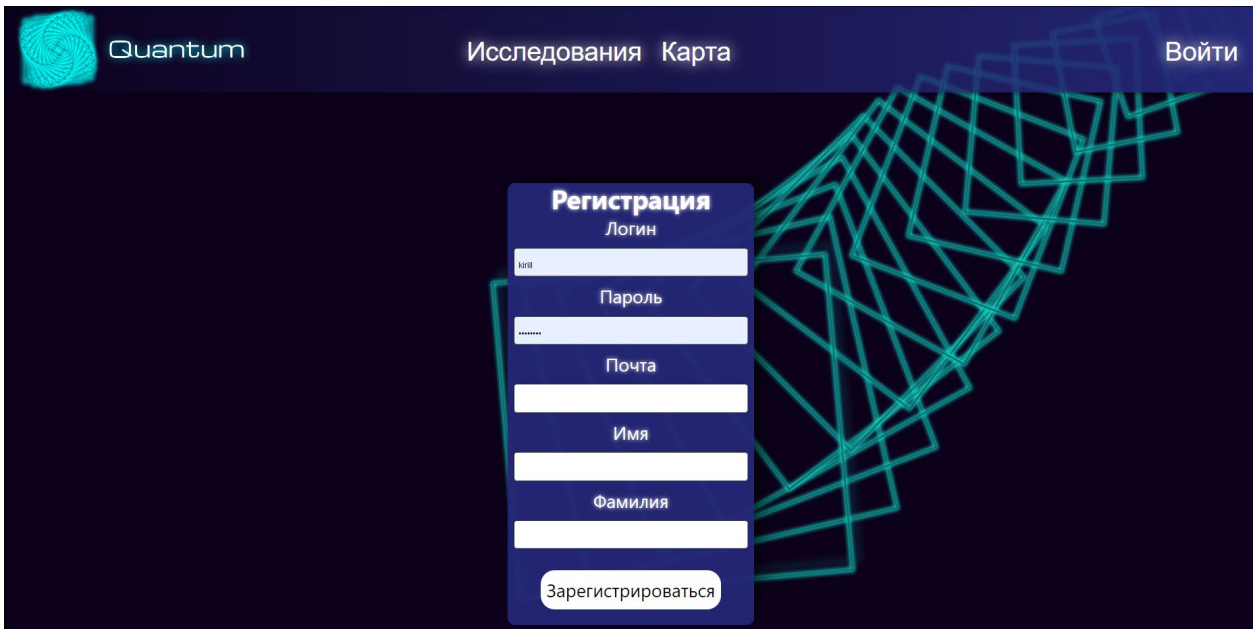

Рисунок 10 – Страница регистрации

На этой странице пользователь заполнит свои данные и нажмет на кнопку «Зарегистрироваться» после чего система его добавит в список пользователей и автоматически авторизует, перенаправляя на главную страницу, с приветствующем уведомлением (рис. 11)

Рисунок 11 – Главная страница с авторизированным пользователем

Далее рассмотрим процесс что пользователь захочет рассмотреть кластеры и выбросы. Первым делом пользователю нужно будет из главной страницы перейти на страницу «Исследования» (рис. 12) или же если он захочет посмотреть точки наглядно на карте, то нужно будет перейти на страницу «Карта» (рис. 13), но на этой страницы будет доступно только

переименование кластеров, в отличии от страницы «Исследования», где доступна функция по мимо переименования кластеров и переименование выбросов.

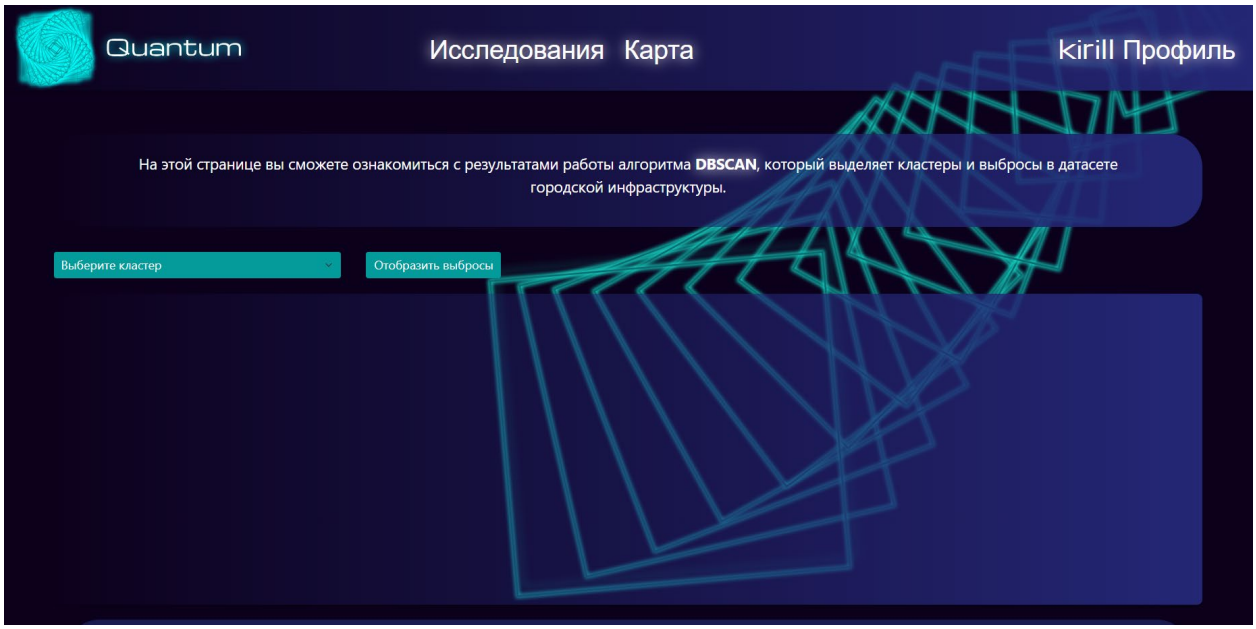

Рисунок 12 – Страница «Исследования» без выбранного кластера

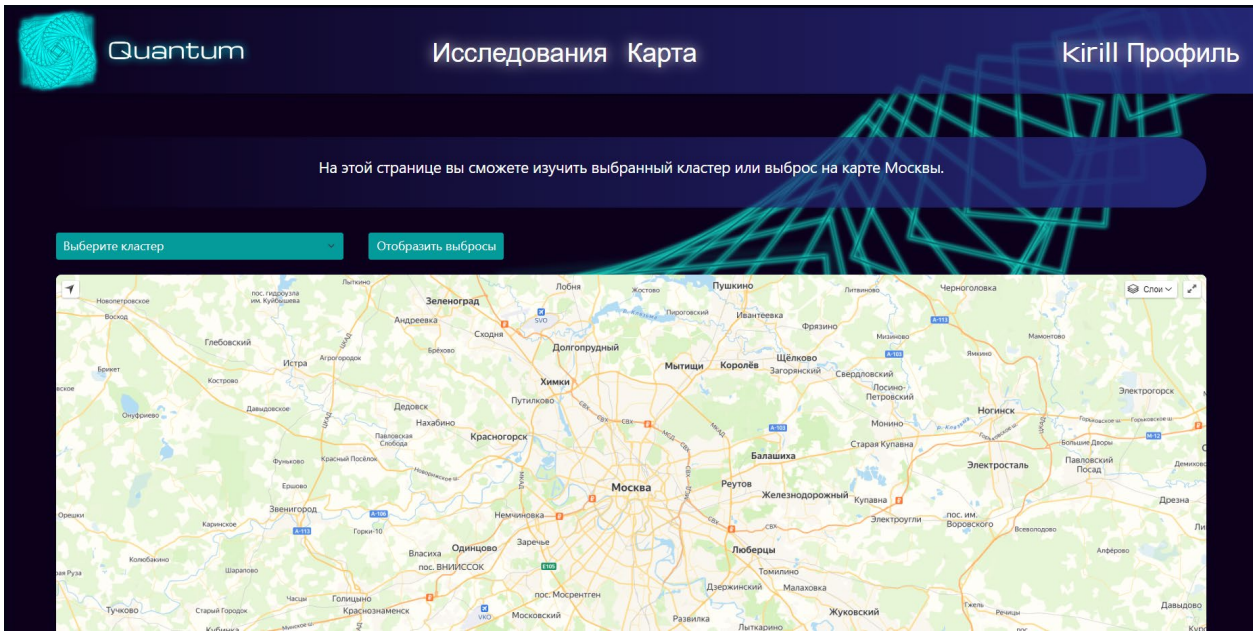

Рисунок 13 – Страница Карты

Далее, чтобы отобразить кластеры нужно выбрать из списка название кластера если пользователь их переименовывал или название в виде технической числовой метки, если же пользователь захочет посмотреть все кластеры, то ему нужно будет необходимо нажать на кнопку «Отобразить

выбросы». Приведем пример того что пользователь решил посмотреть первый кластер и поменять его название (рис. 14).

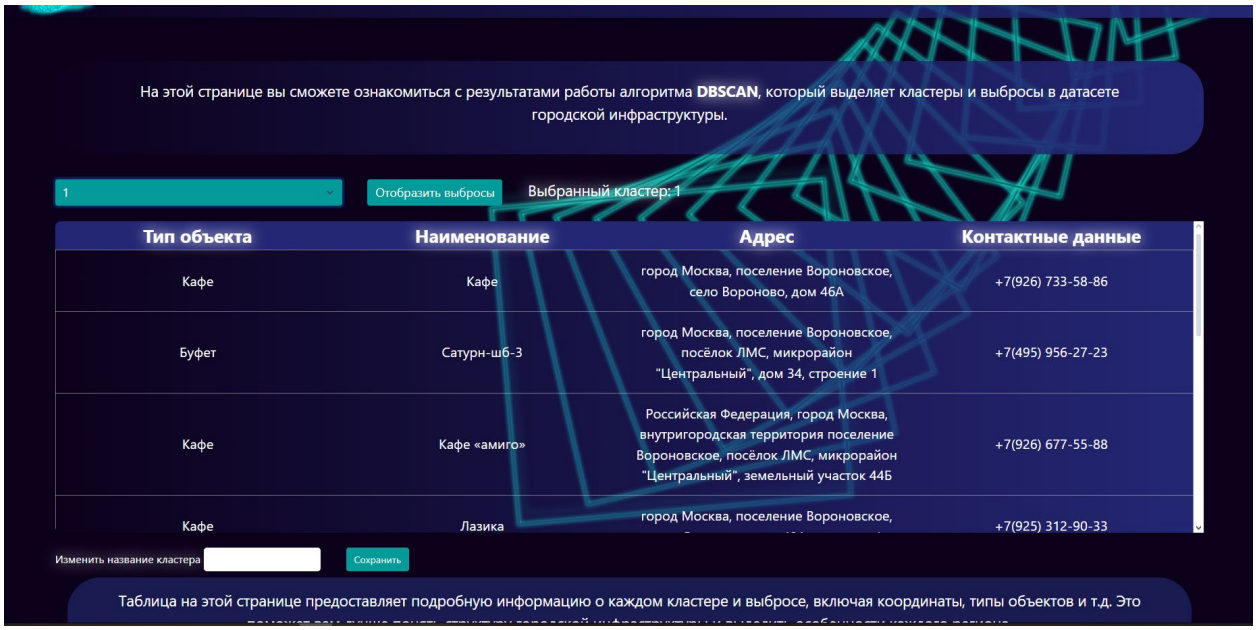

Рисунок 14 – Страница Исследования с выбранным первым кластером

Для того что бы пользователь после просмотра всей подборки (10 записей) поменял название кластера ему необходимо ввести название кластера в поле подписанное «Изменить название кластера» и нажать на кнопку сохранить. После этого произойдет смена названия кластера (рис. 15).

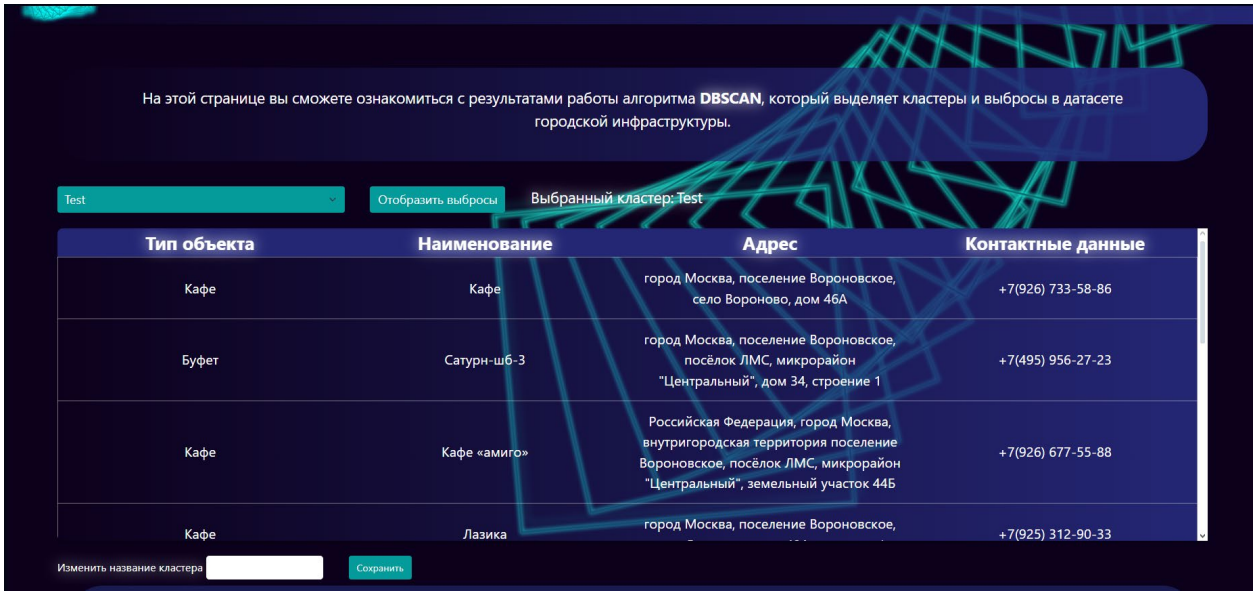

Рисунок 15 – Страница Исследования после изменения названия кластера

Заключение

В ходе разработки онлайн-сервиса были достигнуты важные моменты, направленные на более глубокий анализ инфраструктуры города Москва. Целью работы было создание онлайн-сервиса, позволяющего пользователям визуализировать данные о городской среде, анализировать данные с учетом пространственных зависимостей и кластеризации.

У разработанного онлайн-сервиса можно выделить такие преимущества как, то что использования алгоритма DBSCAN позволяет провести точный анализ инфраструктуры, выделяя кластеры и выбросы, что улучшает качество получаемых результатов. Еще можно выделить интерактивную визуализацию, которая дает возможность взаимодействовать с результатами кластеризации на карте Москвы, что делает информацию более понятной для пользователей. Так же можно выделить гибкость и расширяемость благодаря фреймворку Django, который обеспечивает гибкость расширения функционала.

Недостатки же которые можно выделить это то что есть ограничение по датасетам, то есть учитываются не все объекты инфраструктуры.

Возможные расширения онлайн-сервиса:

1. Расширение Поддержки Других Городов: Возможность адаптации приложения для анализа инфраструктуры в других городах, что делает его универсальным инструментом для городского анализа;
2. Добавление функционала обратной связи;
3. Добавить функционал экпорта данных исследования

В заключение, разработанный онлайн сервис представляет собой важный шаг в направлении создания эффективных инструментов для анализа и планирования городской инфраструктуры. Он открывает новые возможности для исследования и принятия обоснованных решений в области городского развития.

Ссылка на репозиторий:
https://github.com/FLISHL1/Course_progect2024Winter/tree/master/djangoProject

Список литературы и интернет-ресурсов

1. Руководство по веб-фреймворку Django [Электронный ресурс]. URL: https://metanit.com/python/django/ (дата обращения 18.01.2024)

2. Документация по веб-фреймворку Djang [Электронный ресурс]. URL:https://docs.djangoproject.com/en/5.0/ (дата обращения 18.01.2024)

3. Все, что вам нужно знать об алгоритме DBSCAN [Электронный ресурс]. URL: https://skine.ru/articles/87061/ (дата обращения 18.01.2024)

4. Интересные алгоритмы кластеризации, часть вторая: DBSCAN [Электронный ресурс]. URL: https://skine.ru/articles/87061/ (дата обращения 18.01.2024)

5. Руководство по HTML\CSS [Электронный ресурс]. URL: https://developer.mozilla.org/ru/docs/Learn/CSS (дата обращения 18.01.2024)