

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МО и ПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Интерактивный картографический сервис размещения
спортивных объектов города**

Студент	_____	<u>Ключников Андрей Николаевич</u>
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	<u>Балашова И.Ю.</u>
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролер	_____	<u>Такташкин Д.В</u>
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	<u>Макарычев П.П.</u>
	(подпись)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	_____	<u>Попова Н.А.</u>
	(подпись)	(фамилия, инициалы)

Пенза, 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ П.П. Макарычев
«__» _____ 2022 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент _____ Ключников Андрей Николаевич _____ гр. 18ВП1 факультета _____ ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы _____ Интерактивный картографический сервис размещения спортивных
объектов города _____

Тема утверждена приказом ПГУ № _____ от " __ " _____ 2022 г.

3. Руководитель работы _____ к.т.н., доцент кафедры «МО и ПЭВМ» Балашова И.Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: сервис для анализа размещения спортивных объектов города с
возможностью поиска и визуализации табличных данных.

2) Среда выполнения: операционная система Ubuntu 20.04, Web-браузер

Языковые средства разработки: Python, JavaScript, UML.

Инструментальные средства: PyCharm, WebStorm, PostGis, yarn, Draw.io,

Технологии разработки: ООП, принципы SOLID, шаблоны проектирования,
сервисно-ориентированный подход.

3) Основные функции приложения:

а) просмотреть слой карты;

б) просмотреть атрибутивную информацию спортобъекта;

в) сформировать отчет по выбранным гексагонам;

г) добавить новый спортивный объект на карту;

д) отфильтровать спортобъекты участвующие в формировании тепловых карт.

5. Перечень подлежащих разработке вопросов:

1) Анализ предметной области и постановка задачи;

2) Проектирование;

3) Реализация;

4) *Контроль качества программного обеспечения;*

5) *План разработки бюджета;*

6) *Оформление пояснительной записки.*

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	13.04.2022 – 27.04.2022	
2	Проектирование	25%	27.04.2022 – 10.05.2022	
3	Реализация	35%	10.05.2022 – 30.06.2022	
4	Контроль качества программного обеспечения	10%	31.05.2022 – 02.06.2022	
5	План разработки бюджета	5%	02.06.2022 – 03.06.2022	
6	Оформление пояснительной записки	15%	03.06.2022 – 14.06.2022	

Дата выдачи задания «22» _____ декабря _____ 2021 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 90 листов, 27 рисунков, 16 таблиц, 17 использованных источников и 4 приложения.

ПРОСТРАНСТВЕННЫЕ ДАННЫЕ, ВИЗУАЛИЗАЦИЯ, КЛИЕНТ, СЕРВЕР БАЗЫ ДАННЫХ, UML, REST, MVC.

Объект исследования – сервисы анализа размещения спортивных объектов.

Цель выпускной квалификационной работы – разработать сервис, позволяющий производить анализ размещения спортивных объектов.

Технологии разработки – разработка сервера приложений проводилась на языке программирования Python с использованием фреймворка GeoDjango, клиентская часть разрабатывалась на языке JavaScript с использованием фреймворка Vue.js, для создания сервера БД использовался PostgreSQL 12 с расширением для работы с пространственными данными PostGIS. В качестве среды разработки использовались интегрированные среды разработки PyCharm и WebStorm.

Среда исполнения – Linux, Web-браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная часть картографического сервиса. Все требования к разработке приложения соблюдены.

					ПГУ 09.03.04– 8ВР181.04 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Ключников А.Н.			Интерактивный картографический сервис размещения спортивных объектов города. Пояснительная записка	Лит.	Лист
Пров.		Балашова И.Ю.					
						4	90
Н. Контр.		Такташкин Д.В				Группа 18ВП1	
Утверд.		Макарычев П.П.					

Содержание

Введение.....	7
1 Анализ инструментов для анализа размещения спортивных объектов.....	8
1.1 Анализ предметной области и постановка задачи	8
1.2 Анализ программного обеспечения аналогичного назначения.....	11
1.3 Сравнительный анализ аналогов	12
1.4 Анализ функциональных требований	14
1.5 Анализ нефункциональных требований	17
2 Проектирование интерактивного картографического сервиса размещения спортивных объектов города	18
2.1 Архитектура приложения	18
2.2 Проектирование структур данных.....	19
2.2.1 Концептуальная модель базы данных сервиса.....	19
2.2.2 Логическая модель базы данных сервиса	20
2.3 Проектирование программных средств	21
2.4 Проектирование пользовательского интерфейса.....	23
3 Реализация веб-приложения.....	28
3.1 Реализация серверной части веб приложения	28
3.1.1 Реализация базы данных	28
3.1.2 Реализация серверных компонентов	30
4 Контроль качества программного обеспечения	34
4.1 Выбор режима тестирования.....	34
4.2 Тестирование серверной части веб-приложения	35
4.3 Тестирование клиентской части веб-приложения	36
4.4 Метрики устойчивости и абстрактности компонент	36
4.5 Метрики кода	38
4.6 Планирование разработки и оценка бюджета	40
Заключение	43
Список использованных источников	44
Приложение А. Глоссарий	46

Приложение Б. Исходный код серверной части	48
Приложение В. Исходный код веб-приложения.....	67
Приложение Г. Результаты тестирования веб-приложения	88

Введение

За последние десятилетия значительно повысилась доступность информационных систем и, в частности, географических информационных систем. Геоинформационная система (ГИС) – система сбора, хранения, анализа и графической визуализации пространственных данных и связанной с ними информации о необходимых объектах [1].

С повышением доступности и распространенности геоинформационных систем геоаналитика становится доступна для все большего спектра задач и областей. А современные веб-технологии позволяют создавать интерактивные картографические веб-приложения для доступа, к которым пользователю необходим только браузер.

С развитием данных технологий открывается все больше сфер применения интерактивных картографических сервисов. Одной из таких сфер является спортивная инфраструктура. Массовое увлечение спортом и здоровым образом жизни давно уже стало устойчивой тенденцией. В результате чего возникает потребность в наращивании спортивной инфраструктуры, следовательно, возникает задача выявления территорий с недостатком спортивных объектов и определения места для оптимального размещения нового спортивного объекта.

Из всего вышесказанного очевидно, что существует необходимость в картографическом сервисе, предоставляющем географическую аналитику для определения оптимальных мест для размещения спортивных объектов в зависимости как от населения, так и от оснащенности спортивными объектами.

Целью выпускной квалификационной работы является разработка сервиса анализа размещения спортивных объектов. Разработка включает в себя анализ требований, проектирование, реализация и тестирование программного обеспечения.

1 Анализ инструментов для анализа размещения спортивных объектов

1.1 Анализ предметной области и постановка задачи

В настоящее время существует устойчивая тенденция к росту числа людей, занимающихся спортом, вследствие чего возрастает нагрузка на существующую спортивную инфраструктуру.

Развитие спортивной индустрии поддерживается государством, выделяются значительные средства для строительства новой спортивной инфраструктуры и модернизации существующей [2]. Одной из важнейших задач при строительстве нового спортивного объекта является определение места размещения спортивного объекта. Для того, чтобы максимально эффективно размещать спортивную инфраструктуру, необходим сервис, позволяющий выявлять территории с наибольшей потребностью в спортивной инфраструктуре определенных типов.

Для того, чтобы определить требования к такому сервису, необходимо провести анализ наиболее вероятных вариантов использования.

Для эффективного размещения спортивного объекта требуется информация о уже существующей спортивной инфраструктуре, а именно:

- местоположение;
- тип спортивного объекта (бассейн, теннисный корт и тд);
- виды спорта, которыми возможно заниматься на спортивном объекте;
- площадь спортивного объекта;
- критерий значимости спортивного объекта (городского значения, окружного, районного) для оценки возможностей объекта для привлечения посетителей.

Также требуется информация о населении территорий, рассматриваемых для строительства новой спортивной инфраструктуры,

прежде всего информации о количестве населения на определенной территории, для анализа потенциальных посетителей спортивного объекта.

Для нормализации данных требуется также визуализация относительного показателя, такого как суммарная площадь спортивных объектов на человека на данной территории, что позволит более точно оценивать потребность территории в дополнительном обеспечении спортивной инфраструктуры.

Информацию о спортивных объектах необходимо разместить на карте. Данные о спортивных объектах можно представить в качестве тепловой карты оснащенности спортивных объектов, отображающей агрегированные показатели площадей спортивных объектов. Местоположение объектов требуется указать на карте маркерами, с возможностью отображения подробной информации об объекте кликом по его маркеру. Должна также присутствовать возможность отключения отображения маркеров спортивных объектов.

Информацию о населении территорий необходимо также представить в виде тепловой карты, отображающей размер населения данной области.

Для удобства восприятия тепловых карт целесообразным будет разбить всю территорию на гексагоны, для каждого из которых будет определяться агрегированный показатель, и соответственно присваиваться свой цветовой показатель. Вычислив агрегированные значения для каждого гексагона и сопоставив каждому значению определенный цвет получим тепловую карту.

Таким образом необходимо реализовать возможность просматривать три вида интерактивных карт:

- тепловая карта плотности спортивных объектов;
- тепловая карта населения;
- тепловая карта плотности спортивных объектов на одного человека.

Также для анализа потребности в спортивной инфраструктуре важно иметь информацию об оснащенности территории объектами определенного

типа или объектами, позволяющими заниматься конкретным видом спорта, следовательно, необходимо иметь возможности фильтрации спортивных объектов по следующим показателям:

- вид спорта;
- тип спортобъекта;
- критерий значимости.

Фильтрация должна осуществляться на всех трех тепловых картах, данной фильтрации также должны подчиняться и отображаемые маркеры спортивных объектов.

Важным функционалом также является возможность размещения на интерактивной карте спортивного объекта для строительства, что позволит оценить обеспеченность территории инфраструктурой в случае постройки данного спортивного объекта, эффективность расположения спортивного объекта и достаточность или недостаточность постройки данного спортивного объекта, а также даст возможности изменить параметры планируемого объекта для лучшего обеспечения потребностей территорий и экономии бюджета.

После нахождения территорий для потенциального места размещения спортивных объектов необходима возможность автоматизированного формирования отчета со сводными показателями выбранных территорий:

- площадь территории;
- население;
- суммарная площадь спортобъектов;
- суммарные площади спортивных объектов по виду спорта и по

типам спортивных объектов.

Исходя из всего описанного выше, целью является создание программного продукта на основе архитектуры, состоящей из серверного приложения, базы данных и клиентского веб-приложения.

1.2 Анализ программного обеспечения аналогичного назначения

Проанализировав рынок программных продуктов для анализа размещения спортивных объектов, можно выделить два направления:

- картографические сервисы, предоставляющие общую информацию об объектах инфраструктуры;
- аналитические геоинформационные платформы.

Существующие картографические сервисы, такие как Zoon, Яндекс.Карты и Google Maps, предоставляют только информацию с кратким описанием и адресом, а также отображают объекты на карте. При этом отсутствует возможность просмотра каких-либо тепловых карт и просмотра аналитики. Также данные сервисы не позволяют добавлять свою информацию для объектов на карте. Таким образом данная группа сервисов имеет ограниченные возможности для проведения анализа размещения спортивного объекта и не соответствует предъявляемым требованиям.

На рисунке 1 представлен пример карты по запросу «Бассейн» от картографического сервиса Яндекс.Карты.

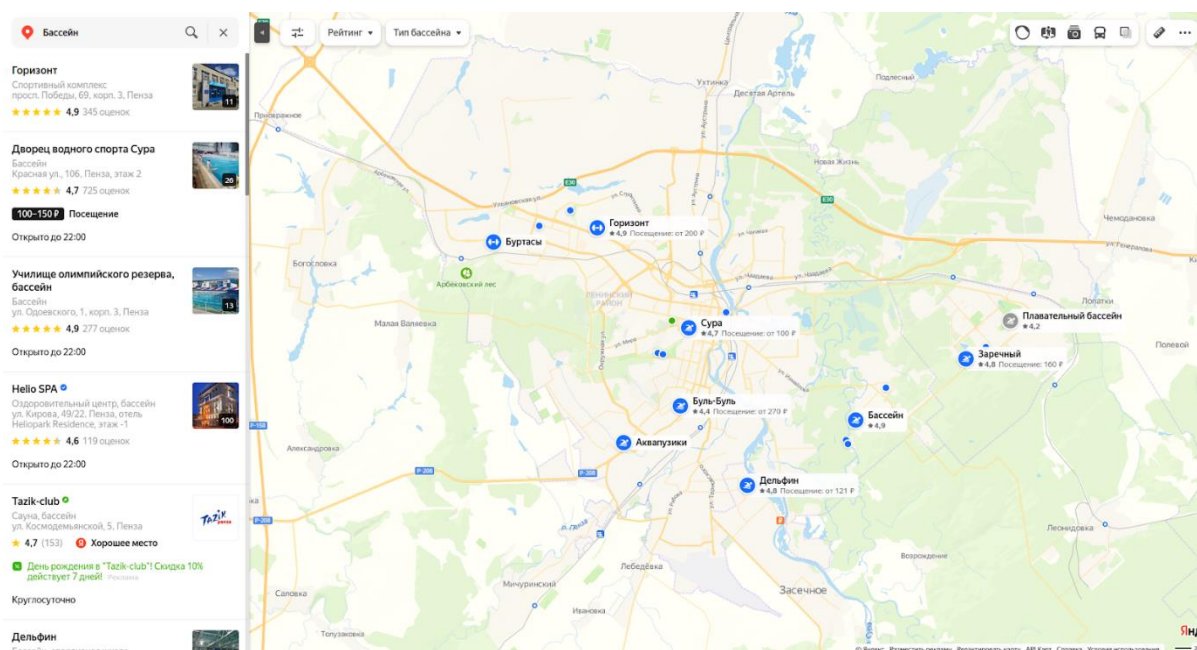


Рисунок 1 – Карта с бассейнами от сервиса Яндекс.Карты

Рассмотрены следующие аналитические геоинформационные платформы: Qlik sense, Tableau, ArcGis.

Qlik sense – представляет собой платформу для проведения аналитики с широким выбором инструментов, в том числе имеет возможности для проведения географического анализа. Сервис предоставляет веб-компонент позволяющий интегрировать интерактивную карту на любую веб-страницу. Также существует возможность создания отчетов на основе интерактивной карты [3].

Tableau – аналитическая платформа, позволяющая автоматически превращать пространственные данные в разнообразные интерактивные карты. Также Tableau предоставляет готовые наборы данных, такие как данные переписи, дохода и другие стандартные наборы демографических данных. Платформа имеет богатый функционал по созданию отчетов [4].

ArcGis – геоинформационная система, позволяющая создавать собственные интерактивные карты, используя свои данные либо используя данные загруженные другими пользователями. Предоставляет богатый набор инструментов для проведения геоинформационного анализа.

1.3 Сравнительный анализ аналогов

Проведем сравнительный анализ систем, предоставляющих возможности для геоаналитики в области размещения спортивных объектов.

Для проведения сравнительного анализа программных продуктов на основе исследования вариантов использования были выбраны критерии и составлены их коэффициенты значимости. Критерии и коэффициенты значимости представлены в таблице 1.

Таблица 1 – Критерии оценки систем размещения спортивных объектов

№	Наименование	Коэффициент значимости
	Функциональные возможности	
1	Просмотр тепловой карты	1
2	Просмотр маркеров объектов на карте	1
3	Автоматическое построения тепловой карты населения	0.5
4	Автоматическое построения тепловой карты оснащенности спортивными объектами	0.75
5	Автоматическое построения тепловой карты плотности спортивных объектов на одного человека	0.75

Продолжение таблицы 1

№	Наименование	Коэффициент значимости
6	Построение тепловой карты по загруженным данным	1
7	Создание отчетов	0.5
8	Возможность интерактивного добавления объектов	0.75
	Технические возможности	
9	Хранение данных в СУБД	0.5
10	Возможность внесения изменений	0.75
	Эргономика	
11	Адаптивность к размерам экрана устройств	0.25
12	Кроссбраузерность	0.25
13	Интуитивно-понятный интерфейс	0.5

Для оценки описанных выше критериев используется пятибалльная шкала соответствия, от 0 – «полное несоответствие», до 4 – «полное соответствие» требованиям. В таблице 2 представлены характеристики оценки.

Таблица 2 – Шкала баллов

Балл	Характеристика оценки
0	функция (качество) отсутствует, реализовать ее невозможно
1	функция (качество) отсутствует, но ее возможно реализовать путем доработки системы
2	похожие, но отличные от требуемых, функции или качества системы находятся в стадии разработки
3	реализованы похожие, но несколько отличные от требуемых, функции или качества системы
4	функция или качество безоговорочно реализовано

Для анализа были отобраны следующие сервисы и системы:

- Qlik sense;
- Tableau;
- ArcGis.

Заполним таблицу 3, оценив реализованный функционал описанных выше систем. Затем произведем расчет аддитивной суммы интегральной оценки для каждого сравниваемого продукта по следующей формуле:

$$O_{пп} = \sum_{i=1}^n (Z_i * K_i),$$

где $O_{пп}$ – интегральная оценка программного продукта;

n – количество критериев сравнения заявленных продуктов согласно таблице критериев;

Z_i – значение выполнения критерия (1 – выполнен, 0 – не выполнен);

K_i – коэффициент значимости критерия сравнения (от 0 до 1).

Таблица 3 – Критериальная оценка программных продуктов

№	Критерий	Программный продукт №		
		1	2	3
1	Просмотр тепловой карты	4	4	4
2	Просмотр маркеров объектов на карте	4	4	4
3	Автоматическое построения тепловой карты населения	1	4	1
4	Автоматическое построения тепловой карты оснащенности спортивными объектами	1	1	1
5	Автоматическое построения тепловой карты плотности спортивных объектов на одного человека	1	1	1
6	Построение тепловой карты по загруженным данным	3	4	1
7	Создание отчетов	2	4	1
8	Возможность интерактивного добавления объектов	4	4	3
9	Хранение данных в СУБД	4	4	3
10	Возможность внесения изменений	3	4	4
11	Адаптивность к размерам экрана устройств	4	4	4
12	Кроссбраузерность	4	4	4
13	Интуитивно-понятный интерфейс	3	4	3
	Интегральная оценка	24,75	29,5	21,75

Подводя итог сравнения можно сделать вывод, что лидером является зарубежная платформа аналитики Tableau. Но она, как и остальные рассматриваемые системы, требует тщательной настройки отображения тепловых карт и достаточно высокую стоимость. На втором месте идет Qlik sense. Затем с небольшим отрывом следует сервис ArcGis.

1.4 Анализ функциональных требований

В соответствии с выявленными функциональными требованиями была разработана диаграмма вариантов использования (рисунок 2).

Сценарии наиболее важных вариантов использования представлены на рисунках 3 – 5.

На рисунке 3 приведено описание основного порядка выполнения ролью «Пользователь» варианта использования «Просмотреть объединенный слой», позволяющий произвести анализ территории по показателю средней площади спортивных объектов на человека.

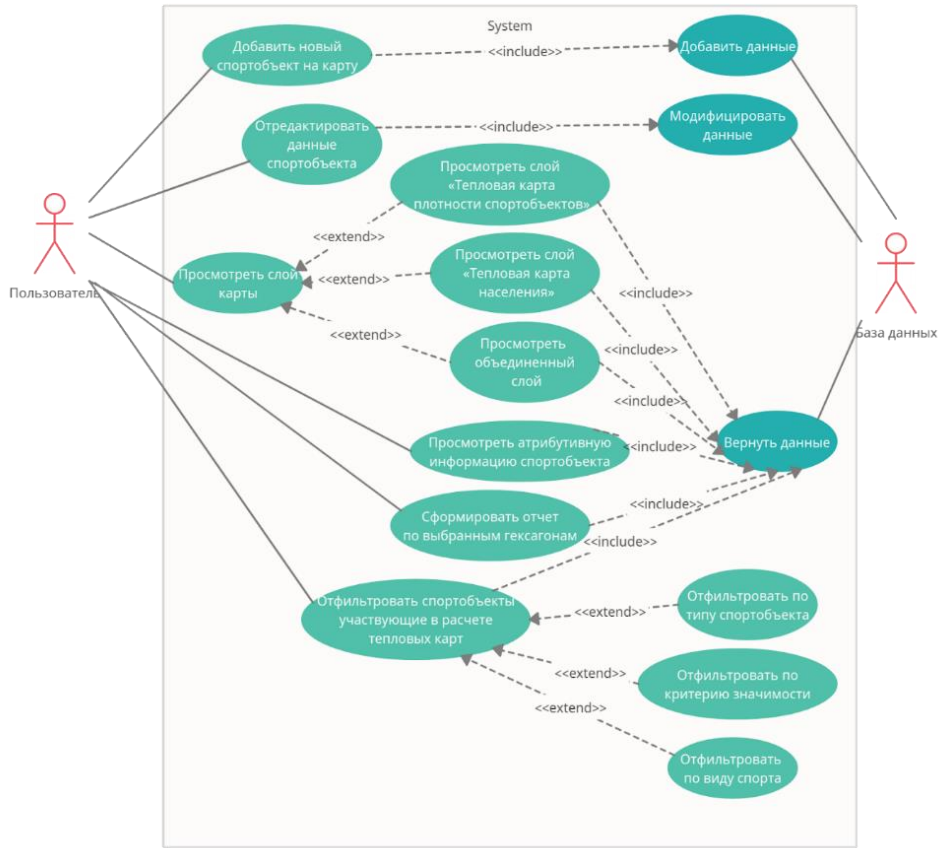


Рисунок 2 – Диаграмма вариантов использования

Прецедент: Просмотреть объединенный слой
ID:3
Краткое описание: Пользователь просматривает объединенный слой.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: 1. Пользователь открыл веб-приложение. 2. Пользователь включил отображение объединенного слоя.
Основной поток: 1. Пользователь настроил фильтрацию попадающих в расчет спортивных объектов. 2. Пользователь перешел на интересующую его область на карте.
Постусловия: 1. Приложение отображает тепловую карту площади спортобъектов на человека.
Альтернативные потоки: Нет

Рисунок 3 – Спецификация для варианта использования «Просмотреть объединенный слой»

На рисунке 4 приведено описание основного порядка выполнения ролью «Пользователь» варианта использования «Сформировать отчет по выбранным гексагонам».

Прецедент: Сформировать отчет по выбранным гексагонам
ID:5
Краткое описание: Пользователь просматривает текущий результат.
Главные актеры: Пользователь
Второстепенные актеры: Нет.
Предусловия: Пользователь открыл веб-приложение.
Основной поток: <ol style="list-style-type: none"> 1. Пользователь выбрал один из слоев карты. 2. Пользователь выбрал гексагоны, по которым необходимо сформировать отчет. 3. Пользователь нажал на кнопку формирования отчета.
Постусловия: Система отображает отчет по выбранным гексагонам.
Альтернативные потоки: Нет

Рисунок 4 – Спецификация для варианта использования «Сформировать отчет по выбранным гексагонам»

На рисунке 5 приведено описание основного порядка выполнения ролью «Пользователь» варианта использования «Просмотреть атрибутивную информацию спортобъекта».

Прецедент: Просмотреть атрибутивную информацию спортобъекта
ID:4
Краткое описание: Пользователь просматривает атрибутивную информацию выбранного спортобъекта.
Главные <u>актеры</u> : Пользователь
Второстепенные актеры: Нет.
Предусловия: Пользователь открыл веб-приложение.
Основной поток: <ol style="list-style-type: none"> 1. Пользователь включил отображение маркеров спортивных объектов. 2. Пользователь нажал на маркер спортивного объекта, информацию о котором хочет увидеть.
Постусловия: Система отображает атрибутивную информацию о выбранном объекте.
Альтернативные потоки: Нет

Рисунок 5 – Спецификация для варианта использования «Просмотреть атрибутивную информацию спортобъекта»

В результате были смоделированы бизнес-процессы, позволяющие пользователю производить анализ размещения спортивных объектов.

1.5 Анализ нефункциональных требований

Разработка серверной части приложения должна быть выполнена в интегрированной среде разработки PyCharm на языке программирования Python с использованием фреймворка Django. Необходимо реализовать RESTful веб-сервис. Клиентскую часть необходимо написать на языке Javascript с использованием фреймворка Vue.js в интегрированной среде разработки WebStorm. В качестве СУБД должна выступать PostgreSQL с расширением для работы с пространственными данными PostGIS. Взаимодействие с СУБД необходимо организовать при помощи, встроенной в фреймворк Django ORM. Приложение должно работать в браузерах: Firefox, Google Chrome, Safari.

2 Проектирование интерактивного картографического сервиса размещения спортивных объектов города

2.1 Архитектура приложения

При разработке веб-приложения использовалась трехуровневая архитектура. Схема архитектуры представлена на рисунке 6.

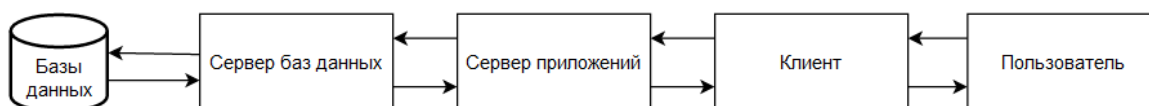


Рисунок 6 – Клиент-серверная архитектура

В трехуровневой архитектуре существует три слоя:

- слой представления;
- слой предметной области;
- слой источника данных.

Слой представления охватывает все, что имеет отношение к общению пользователя с системой. Источник данных – это подмножество функций, обеспечивающих взаимодействие со сторонними системами, которые выполняют задания в интересах приложения. Для большинства корпоративных приложений основная часть логики источника данных сосредоточена в коде СУБД. Слой предметной области описывает основные функции приложения, предназначенные для достижения поставленной перед ним цели. К таким функциям относятся вычисления на основе вводимых и хранимых данных, обработка команд, поступающих от слоя представления, а также передача информации слою источника данных [5].

Взаимодействие между слоем представления и слоем предметной области осуществляется с помощью REST API по протоколу HTTP. При необходимости слой предметной области посредством SQL-запросов взаимодействует со слоем источника данных, представленного СУБД PostgreSQL.

2.2 Проектирование структур данных

2.2.1 Концептуальная модель базы данных сервиса

Концептуальная модель представляет собой описание основных сущностей (таблиц) и связей между ними независимо от ограничений, вытекающих из моделей данных, поддерживаемая той или иной СУБД [6]. Часто на такой модели отображаются только имена сущностей (таблиц) без указания их атрибутов.

Графическое представление концептуальной модели базы данных в виде диаграммы «сущность-связь» для предметной области информационной системы представлено на рисунке 7.



Рисунок 7 – Графическое представление концептуальной модели

В таблице 4 представлены сущности, сформированные на основании входных и выходных данных бизнес-процессов.

Таблица 4 – Описание сущностей

№	Наименование	Описание	Количество экземпляров	Режим обновления
1	Департамент	Департамент(министерство и тд.)	200	Раз в 5 лет или реже
2	Спортивный объект	Спортивный объект города Москва	10000	Раз в день или реже
3	Гексагон	Гексагональная область на карте	2000	Раз в год
4	Вид спорта	Вид спорта	100	Раз в год или реже

Продолжение таблицы 4

№	Наименование	Описание	Количество экземпляров	Режим обновления
5	Спортзона	Спортивная зона внутри спортивного объекта	30000	Раз в день или реже
6	Тип спортзоны	Тип спортзоны(бассейн, тренажерный зал)	100	Раз в год или реже

В таблице 5 представлено описание связей между сущностями.

Таблица 5 – Описание связей

№	Связь	Описание	Мощность	Обязательность
1	department – facility	Принадлежность объекта департаменту(Ведомственная принадлежность)	1:M	Да
2	sport_type – facility	Определяет доступные виды спорта в спортивном объекте	M:N	Нет
3	sport_type – sport_area	Определяет доступные виды спорта в спортзоне	M:N	Нет
4	data_hex – facility	Принадлежность спортивного объекта гексагону(нахождение внутри гексагона)	1:M	Да
5	facility – sport_area	Принадлежность спортзоны спортивному объекту	1:M	Да
6	sport_area_type – sport_area	Определяет вид спортзоны	1:M	Да

Концептуальная модель построена в среде draw.io в нотации ERD (Entity Relationship Diagram) [7].

2.2.2 Логическая модель базы данных сервиса

Логическая модель – это расширение концептуальной модели данных. Она включает в себя все сущности, атрибуты, ключи и взаимосвязи, которые представляют бизнес-информацию и определяют бизнес-правила [8].

На основании концептуальной модели была сконструирована логическая модель базы данных (рисунок 8).

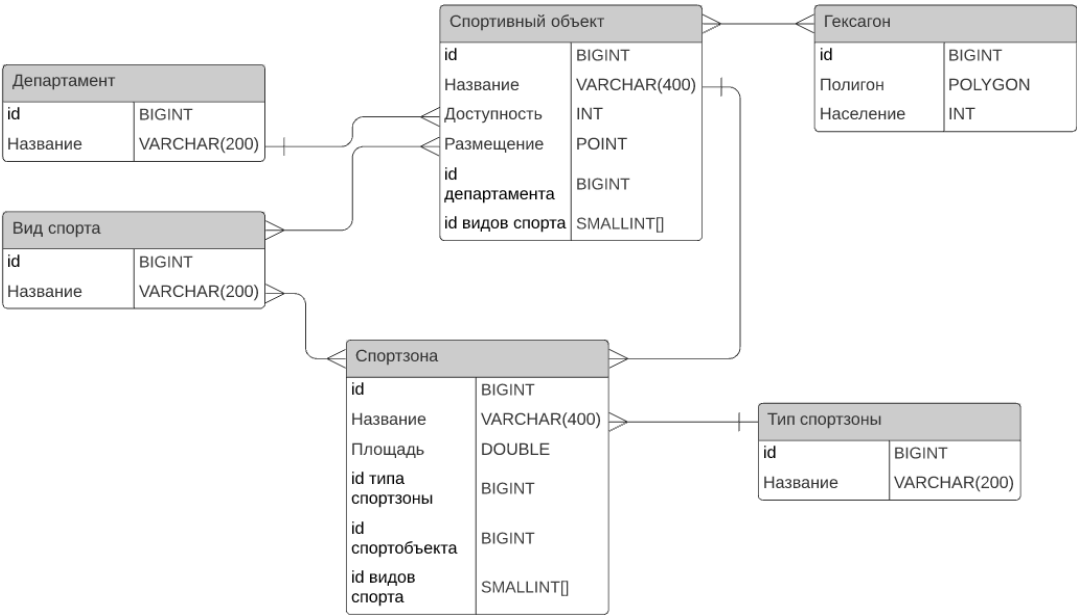


Рисунок 8 – Графическое представление логической модели базы данных

Концептуальная модель построена в среде lucidchart в нотации IDEF1X [9].

2.3 Проектирование программных средств

На основе модели базы данных была спроектирована диаграмма классов моделей Django (рисунок 9).

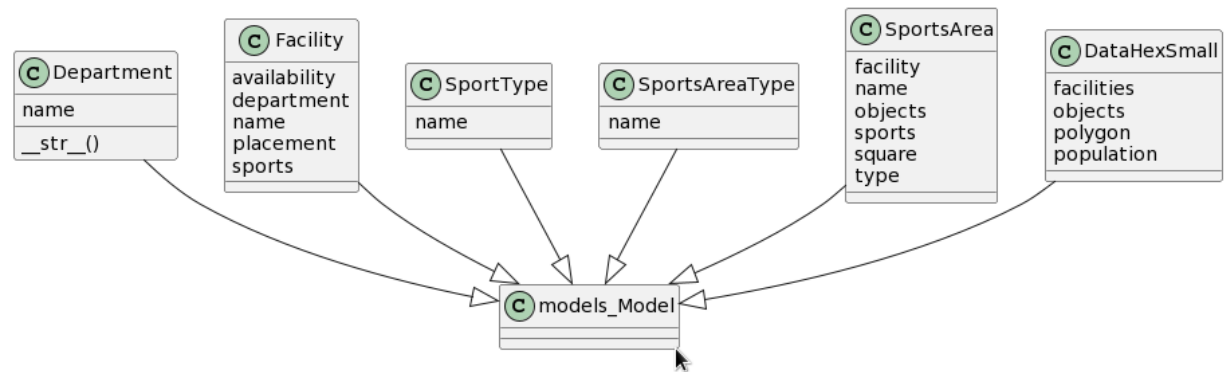


Рисунок 9 – Диаграмма классов моделей

В таблице 6 представлена спецификация классов моделей Django.

Таблица 6 – Спецификация классов

Класс	Описание	Поля	Описание	Методы	Описание
Department	Класс модели Department	name	Название департамента	__str__()	Возвращает текстовое представление объекта департамента
Facility	Класс модели Facility	name	Название спортивного объекта	__str__()	Возвращает текстовое представление объекта спортивного сооружения
		department	Департамент, к которому относится объект		
		availability	Доступность спортивного объекта (шаговая, районная, городская)		
		placement	Координаты объекта		
		sports	Идентификаторы видов спорта, для занятия которыми предназначен объект		
SportType	Класс модели SportType	name	Название вида спорта	__str__()	Возвращает текстовое представление объекта вида спорта
SportsArea	Класс модели SportsArea	name	Название спортзоны	__str__()	Возвращает текстовое представление объекта департамента
		square	Площадь спортзоны		
		type	Тип спортзоны		
		facility	Спортобъект, к которому относится спортзона		
		sports	Виды спорта, для занятия которыми предназначен объект		
SportsAreaType	Класс модели SportsAreaType	name	Название вида спортзоны	__str__()	Возвращает текстовое представление объекта вида спортзоны
DataHexSmall	Класс модели DataHexSmall	polygon	Координаты полигона	__str__()	Возвращает текстовое представление объекта гексагона
		population	Население, проживающее внутри гексагона		
		facilities	Спортивные объекты, в область влияния которых входит гексагон		

Классы моделей были унаследованы от родительского класса моделей фреймворка Django [10].

2.4 Проектирование пользовательского интерфейса

На основе диаграммы вариантов использования для клиентского приложения был спроектирован графический интерфейс. Пользовательский интерфейс должен обеспечивать простое и эффективное выполнение основных функций пользователей.

Пользовательское приложение представляет собой SPA-приложение (single page application). SPA – это реализация веб-приложения, которое загружает только один веб-документ, а затем обновляет содержимое тела этого единственного документа через API-интерфейсы JavaScript [11]. На рисунке 10 представлен основной экран приложения с интерактивной картой и выбранным слоем «Плотность спортивных объектов».



Рисунок 10 – Главная страница приложения со слоем «Плотность спортивных объектов»

Основные элементы интерфейса данного окна, представленного на рисунке 10:

1 – форма, позволяющая настроить фильтрацию спортивных объектов, участвующих в аналитике;

- 2 – кнопка, применяющая настроенную фильтрацию;
- 3 – кнопка открытия модального окна для управления интерактивной картой;
- 4 – кнопка, позволяющая перейти в режим построения отчета;
- 5 – переключатель слоев карты;
- 6 – чекбокс, управляющий отображением маркеров объектов;
- 7 – легенда карты;
- 8 – блок кнопок управления масштабом карты.

На рисунке 11 представлен основной экран приложения с выбранным слоем «Плотность населения». Пустые гексагоны означают отсутствие населения, проживающего на данной территории.

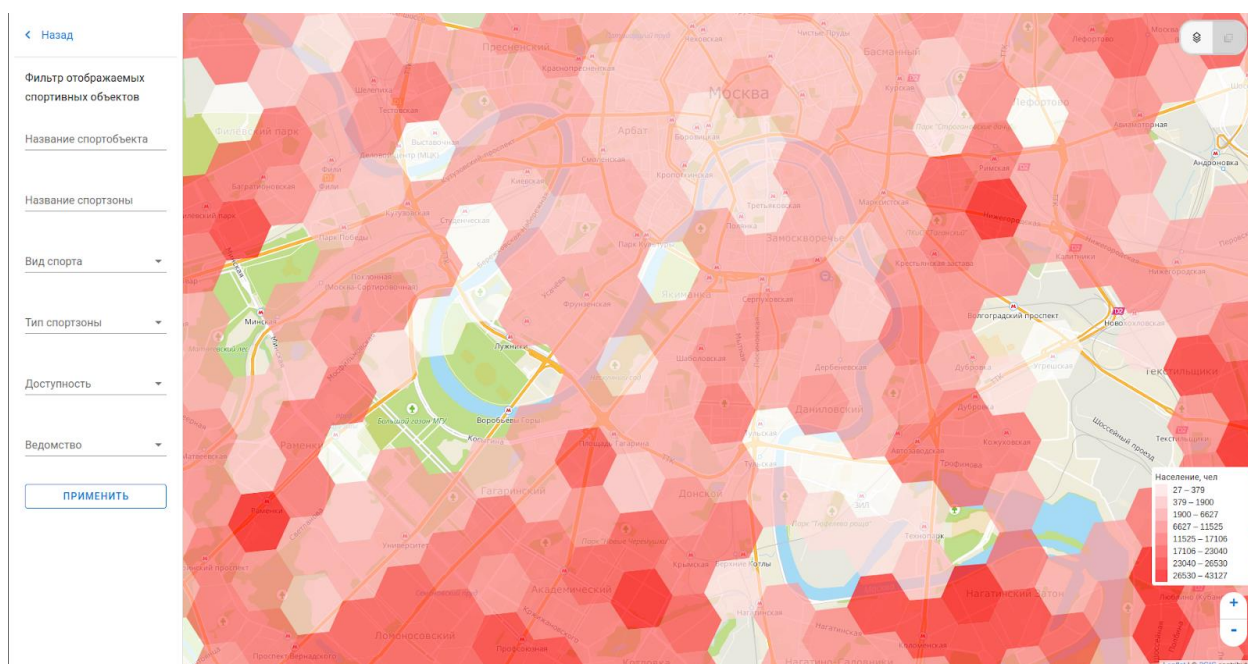


Рисунок 11 – Главная страница приложения со слоем «Плотность населения»

Из рисунка 11 видно, что в зависимости от выбранного слоя карты, легенда карты изменяется.

На рисунке 12 показана карта со слоем «Площадь спортивных объектов на человека», в левом сайдбаре выбрана форма фильтрации отображаемых маркеров спортивных объектов.

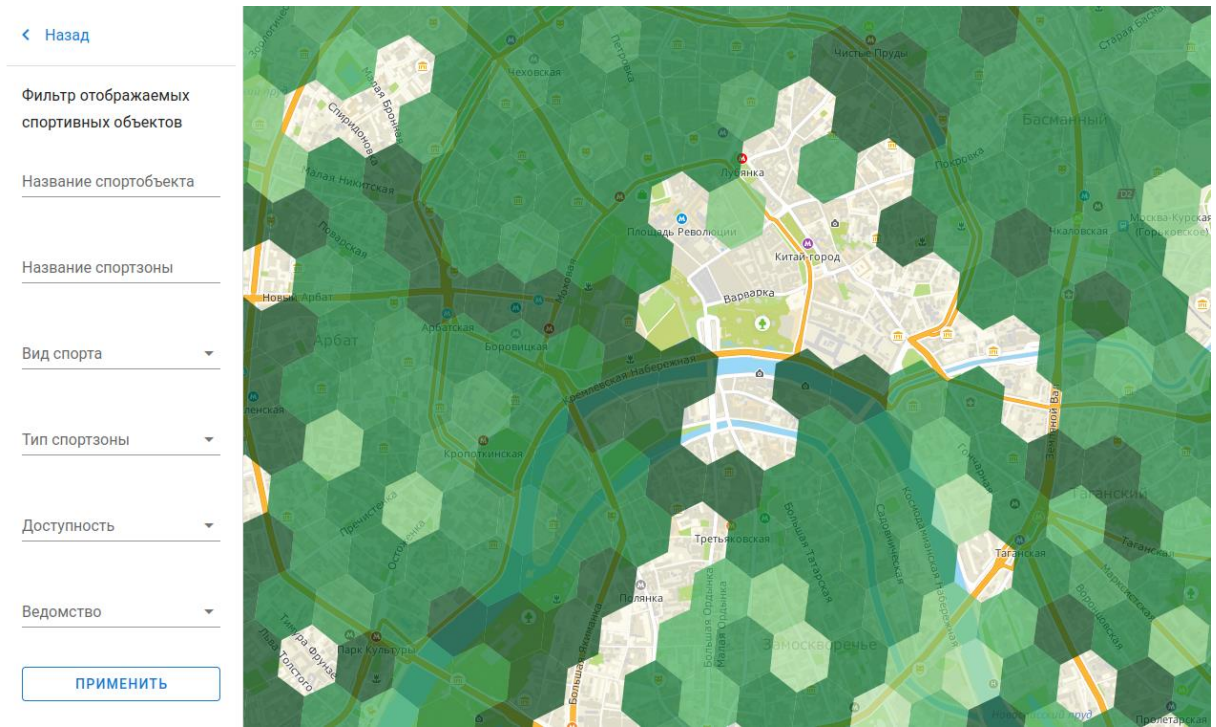


Рисунок 12 – Главная страница приложения со слоем «Площадь спортивных объектов на человека»

На рисунке 13 показана интерактивная карта с включенным отображением маркеров спортивных объектов. В левом сайдбаре отображается информация о выбранном спортивном объекте.



Рисунок 13 – Главная страница приложения с атрибутивной информацией по выбранному объекту

На рисунке 14 показан процесс выбора территорий для составления отчета.

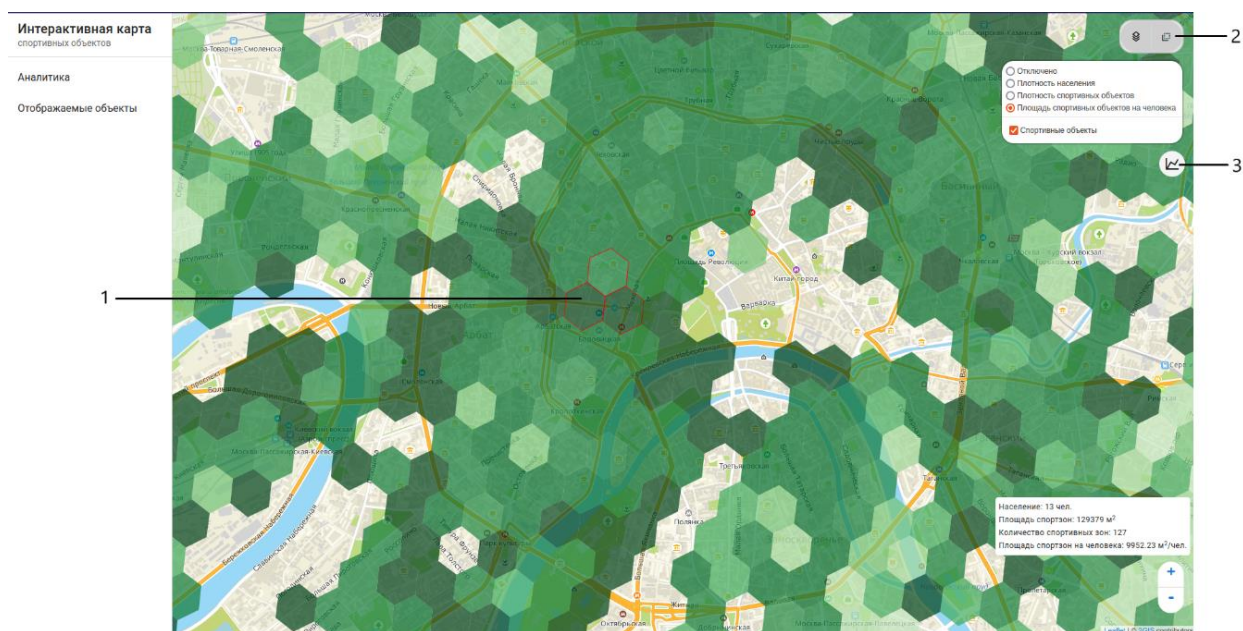


Рисунок 14 – Главная страница приложения с тремя wybranнми гексагонами

Основные элементы интерфейса данного окна, представленного на рисунке 14:

- 1 – территория, выбранная путем выбора гексагонов на карте левой клавишей мыши;
- 2 – кнопка, позволяющая перейти в режим построения отчета;
- 3 – кнопка генерации отчета.

На рисунках 15, 16 представлена страница отчета по выбранным территориям, полученная после выбора территорий на интерактивной карте для построения отчета и нажатия на кнопку генерации отчета. На странице отображены базовые метрики выбранной территории и расположение выбранной территории на карте (рисунок 15), а также график распределения площади спортивных зон на одного жителя выбранной территории по типу спортзоны (рисунок 16).

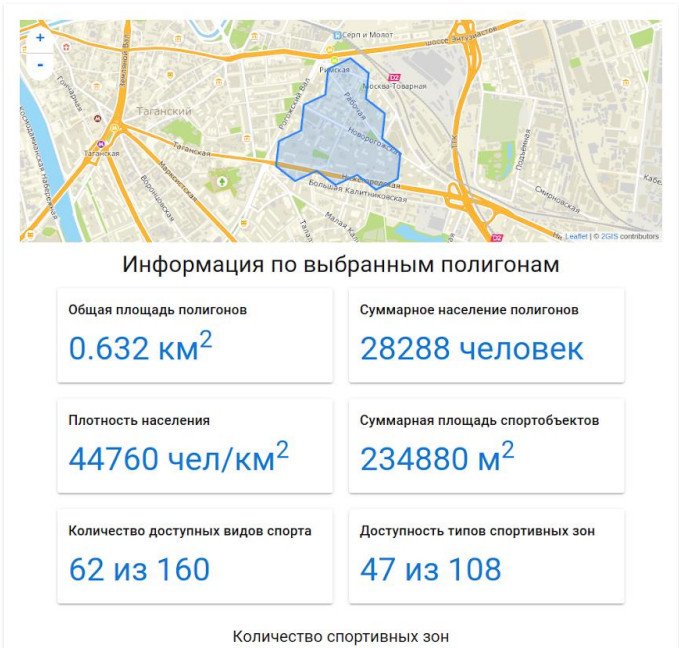


Рисунок 15 – Страница приложения с отчетом по выбранной на карте территории

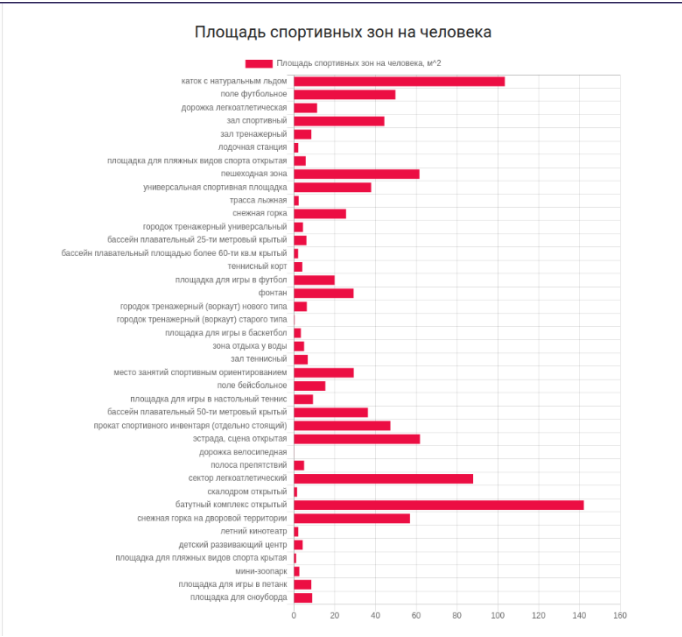


Рисунок 16 – Страница приложения с отчетом о площади спортивных зон по выбранной на карте территории

Таким образом, спроектированный пользовательский интерфейс удовлетворяет функциональным требованиям приложения.

3 Реализация веб-приложения

3.1 Реализация серверной части веб приложения

3.1.1 Реализация базы данных

Физическая модель – логическая модель базы данных, выраженная в терминах языка описания конкретной СУБД. В данной работе – в терминах СУБД PostgreSQL.

На основе концептуальной и логической модели была разработана физическая модель базы данных (рисунок 17).

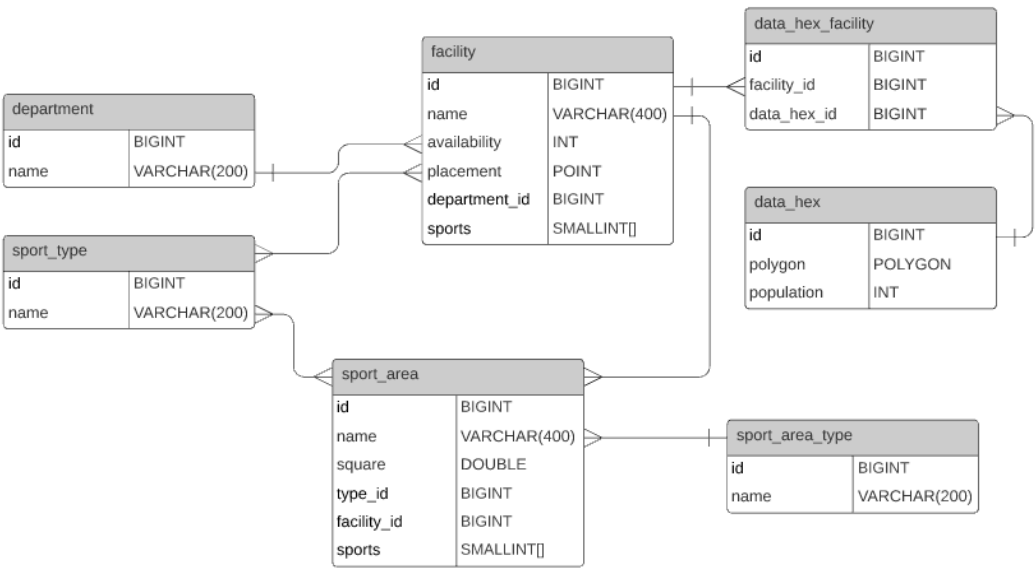


Рисунок 17 – Физическая модель базы данных

Согласно физической модели, была создана база данных в СУБД PostgreSQL.

В таблице 7 представлена спецификация сущности «Департамент».

Таблица 7 – Описание атрибутов сущности «Департамент»

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
id	Идентификатор	BIGINT	Нет	> 0	–	PK
name	Название департамента	VARCHAR(200)	Нет	Любое	–	–

В таблице 8 представлена спецификация сущности «Тип спорта».

Таблица 8 – Описание атрибутов сущности «Тип спорта»

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
id	Идентификатор	BIGINT	Нет	> 0	–	PK
name	Название вида спорта	VARCHAR(200)	Нет	Любое	–	–

В таблице 9 представлена спецификация сущности «Тип спортзоны».

Таблица 9 – Описание атрибутов сущности «Тип спортзоны»

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
id	Идентификатор	BIGINT	Нет	> 0	–	PK
name	Название вида спортзоны	VARCHAR(200)	Нет	Любое	–	–

В таблице 10 представлена спецификация сущности «Спортзона».

Таблица 10 – Описание атрибутов сущности «Спортзона»

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
id	Идентификатор	BIGINT	Нет	> 0	–	PK
name	Название спортзоны	VARCHAR(400)	Нет	Любое	–	–
square	Площадь спортзоны	DOUBLE PRECISION	Нет	> 0	–	–
type_id	Идентификатор типа спортзоны	BIGINT	Нет	> 0	–	FK
facility_id	Идентификатор спортобъекта, к которому относится спортзона	BIGINT	Нет	> 0	–	FK
sports	Идентификаторы видов спорта, для занятия которыми предназначен объект	BIGINT[]	Нет	> 0	–	FK

В таблице 11 представлена спецификация сущности «Спортивный объект».

Таблица 11 – Описание атрибутов сущности «Спортивный объект»

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
id	Идентификатор	BIGINT	Нет	> 0	–	PK
name	Название спортивного объекта	VARCHAR(400)	Нет	Любое	–	–
availability	Доступность спортивного объекта (шаговая, районная, городская)	SMALLINT	Нет	1 - 4	–	–
placement	Координаты объекта	POINT	Нет	Любое	–	–
department_id	Идентификатор департамента, к которому относится объект	BIGINT	Да	> 0	–	FK
sports	Идентификаторы видов спорта, для занятия которыми предназначен объект	BIGINT[]	Нет	> 0		FK

В таблице 12 представлена спецификация сущности «Гексагон».

Таблица 12 – Описание атрибутов сущности «Гексагон»

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
id	Идентификатор	BIGINT	Нет	> 0	–	PK
polygon	Координаты полигона	POLYGON	Нет	Любое	–	–
population	Население, проживающее внутри гексагона	INT	Нет	> 0	–	–

На основе физической модели была построена база данных.

3.1.2 Реализация серверных компонентов

Серверная часть приложения реализовывалась в среде программирования PyCharm Professional с использованием языка Python и

фреймворка Django. Взаимодействие с базой данной производилось посредством Django ORM.

Серверная часть включает в себя классы моделей, отображающие сущности базы данных, классы сериализаторов, которые сериализуют данные классов моделей в формат удобный для общения серверного и клиентского приложений, и классы представления, которые служат для реализации представлений.

На рисунке 18 изображена диаграмма классов представлений приложения.

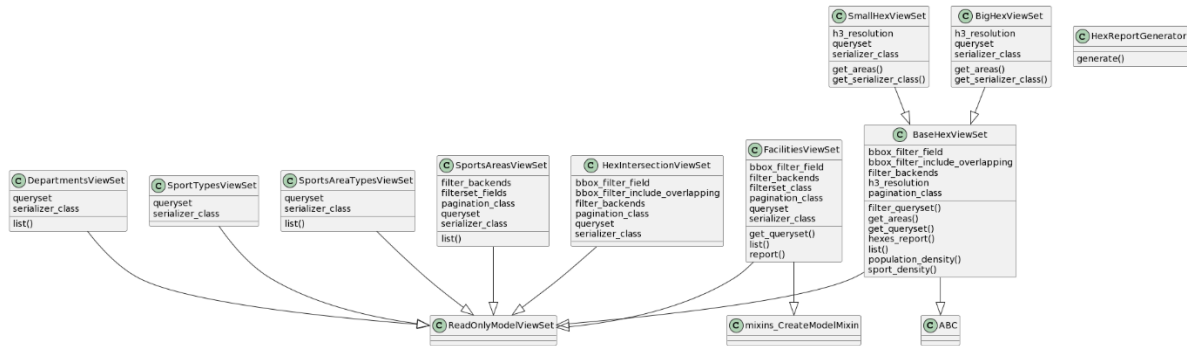


Рисунок 18 – Диаграмма классов представлений приложения

Серверное приложение предоставляет REST API для взаимодействия. Предоставляемые методы описаны в таблице 13.

Таблица 13 – Спецификация REST API веб-приложения

№	Назначение сервиса	HTTP метод	URI сервиса	Класс контроллера
1	Получение списка гексагонов	GET	/data-hex/big-hex	BigHexViewSet
2	Получение отчета по выбранным территориям	GET	/data-hex/big-hex/hexes-report	BigHexViewSet
3	Получение списка гексагонов с информацией о их населении	GET	/data-hex/big-hex/population-density	BigHexViewSet
4	Получение списка гексагонов с информацией о плотности спортивных объектов на человека	GET	/data-hex/big-hex/sport-density	BigHexViewSet
5	Получение информации о гексагоне	GET	/data-hex/big-hex/<id>	BigHexViewSet
6	Получение списка департаментов	GET	/departments	DepartmentsViewSet
7	Получение информации о департаменте	GET	/departments/<id>	DepartmentsViewSet

Продолжение таблицы 13

№	Назначение сервиса	HTTP метод	URI сервиса	Класс контроллера
8	Получение списка спортобъектов	GET	/facilities	FacilitiesViewSet
9	Получение информации о спортобъекте	GET	/facilities/<id>	FacilitiesViewSet
10	Получение подробной информации о спортобъекте	GET	/facilities/<id>/report	FacilitiesViewSet
11	Получение списка типов спортивных зон	GET	/sport-area-types	SportsAreaTypesViewSet
12	Получение информации о типе спортзоны	GET	/sport-area-types/<id>	SportsAreaTypesViewSet
13	Получение списка спортивных зон	GET	/sport-areas	SportsAreasViewSet
14	Получение информации о спортивной зоне	GET	/sport-areas/<id>	SportsAreasViewSet
15	Получение списка видов спорта	GET	/sport-types	SportTypesViewSet
16	Получение информации о виде спорта	GET	/sport-types /<id>	SportTypesViewSet

На рисунке 19 показана диаграмма компонентов серверной части приложения.

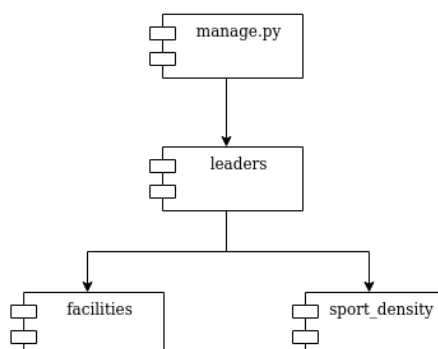


Рисунок 19 – Диаграмма компонентов

Рассмотрим каждый компонент:

- `manage.py` предназначен для выполнения административных команд, таких как создание и применение миграций, запуск сервера разработки, создание администратора и т.д.;
- `leaders` содержит настройки приложения и правила маршрутизации к приложениям `facilities` и `sport_density`;
- `facilities` – компонент, отвечающий за хранение и выдачу информации о спортивных объектах и спортивных зонах, а также содержит бизнес-логику, связанную с ними;

– sport_density – компонент, отвечающий за генерацию и отображение различных тепловых карт.

На рисунке 20 представлена диаграмма развертывания приложения, которая изображает основные компоненты системы.

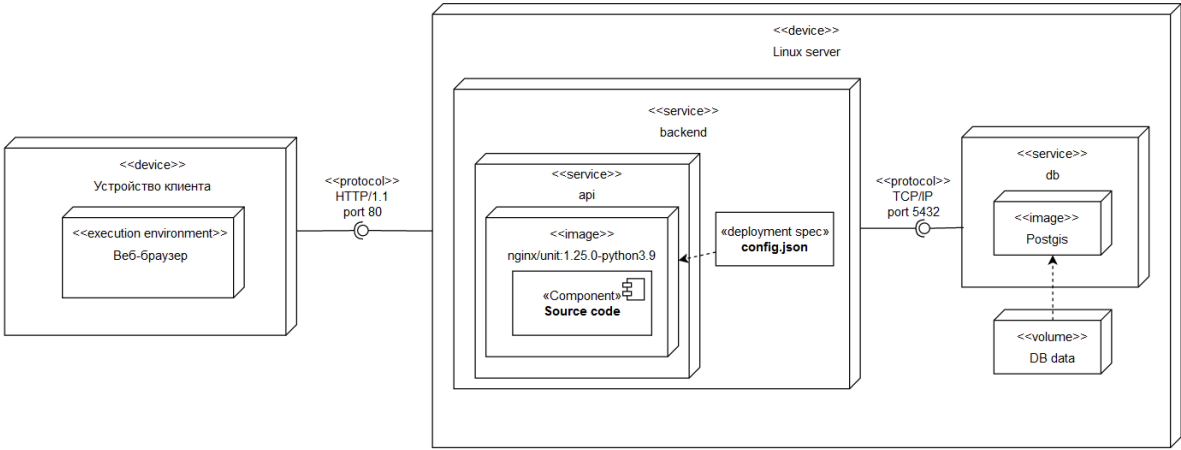


Рисунок 20 – Диаграмма развертывания

Из диаграммы видно, что приложение соответствует трехуровневой архитектуре. Сервер баз данных и сервер приложений обернуты в docker-контейнеры, что позволяет в дальнейшем легко разворачивать приложение на другие сервера с платформами, поддерживаемыми docker.

Docker позволяет хранить приложение со всем его окружением и зависимостями в контейнере и запускать его из изолированной среды контейнера. За описание запуска СУБД PostgreSQL отвечает сервис db. Образ контейнера с СУБД PostgreSQL скачивается из репозитория образов. Для связи с данным контейнером был открыт порт 5432. За описание запуска серверного приложения отвечает сервис api, основанный на образе nginx-unit. Для взаимодействия с данным контейнером по протоколу http был открыт порт 80.

Клиентское приложение запускается на устройствах пользователей в браузере. При открытии пользователем веб-приложения браузер получает все необходимые для запуска файлы клиентского приложения.

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования

Существует достаточно много видов тестирования программного продукта. Были рассмотрены следующие виды тестирования [12]:

- функциональное тестирование;
- модульное тестирование;
- интеграционное тестирование.

Функциональное тестирование – основное внимание уделяется бизнес-требованиям к приложению. Они проверяют только результат некоторого действия и не проверяют промежуточные состояния системы при выполнении этого действия.

Юнит тестирование или модульное тестирование – работают на очень низком уровне, близко к исходному коду приложения. Они заключаются в тестировании отдельных методов и функций классов, компонентов или модулей, используемых в ПО. Модульные тесты, как правило, не требуют больших расходов на автоматизацию и могут выполняться сервером непрерывной интеграции очень быстро.

Интеграционное тестирование – проверяется, хорошо ли работают вместе различные модули и сервисы, используемые приложением. Например, можно протестировать взаимодействие с базой данных или убедиться, что микросервисы работают вместе так, как задумано. Этот вид тестирования является более затратным, поскольку для проведения тестов требуется запуск различных компонентов приложения [13].

Было принято решение осуществить проверку программы с помощью функционального тестирования. Такой выбор обусловлен тем, что к программе были предъявлены в основном функциональные требования, выполнение которых нужно проверить.

4.2 Тестирование серверной части веб-приложения

В результате разработки программы было проведено модульное тестирование. Unit-тесты были разработаны с помощью фреймворка PyTest. Исходный код модульных тестов приведен в приложении. План тестирования приведен в таблице 14.

Таблица 14 – План модульного тестирования

Тест	Описание	Ожидаемый результат	Аktуальный результат
test_big_hex_list	Проверка правильности выдачи результата по запросу /data-hex/big-hex	Пройден	Пройден (рисунок 21)
test_big_hex_hexes_report	Проверка правильности выдачи результата по запросу /data-hex/big-hex/hexes-report	Пройден	Пройден (рисунок 21)
test_big_hex_sport_density	Проверка правильности выдачи результата по запросу /data-hex/big-hex/sport-density	Пройден	Пройден (рисунок 21)
test_big_hex_population_density	Проверка правильности выдачи результата по запросу /data-hex/big-hex/population-density	Пройден	Пройден (рисунок 21)

Результаты модульного тестирования приведены на рисунке 21, из них видно, что все тесты успешно пройдены.

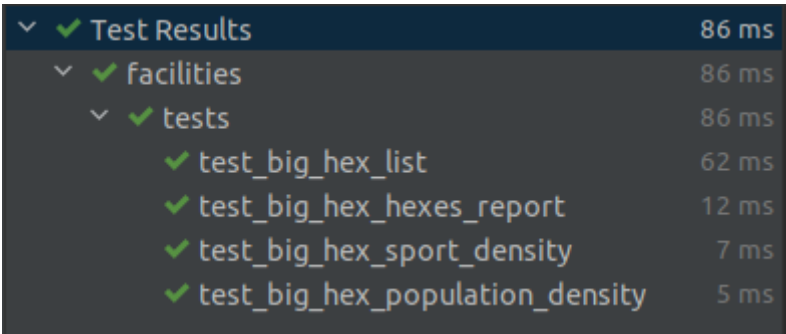


Рисунок 21 – Результаты модульного тестирования

В ходе тестирования были выявлены и исправлены ошибки в коде серверной части веб-приложения.

4.3 Тестирование клиентской части веб-приложения

Для основных вариантов использования были выполнены функциональные тесты. План тестирования приведен в таблице 15.

Таблица 15 – План модульного тестирования

№ п/п	Назначение теста	Описание	Результаты теста
1	Тест предназначен для проверки варианта использования «Просмотреть объединенный слой»	Выбор слоя карты «Площадь спортивных объектов на человека»	Пройден (рисунок Г.1)
2	Тест предназначен для проверки варианта использования «Сформировать отчет по выбранным гексагонам»	Выбор трех гексагонов и нажатие кнопки формирования отчета	Пройден (рисунок Г.2)
3	Тест предназначен для проверки варианта использования «Отфильтровать спортивные объекты участвующие в расчете тепловых карт»	Установка и применение фильтра спортивных объектов.	Пройден (рисунок Г.3)

В ходе тестирования были выявлены и исправлены ошибки в коде программы.

4.4 Метрики устойчивости и абстрактности компонент

Метрика устойчивости для компоненты отражает количественную меру работы, которую необходимо произвести при изменении других зависимых компонент.

Для вычисления позиционной устойчивости кода используется следующая формула:

$$I = \frac{Ce}{Ca + Ce},$$

где Ce – количество классов вне данного компонента, зависящих от классов внутри компонента;

Ca – количество классов внутри данного компонента, зависящих от классов вне компонента.

Величина меры неустойчивости изменяется в диапазоне от $[0, 1]$. Если $I = 0$, то компонент максимально устойчив, если $I = 1$, то компонент максимально неустойчив.

Для компонента `sport_density` была рассчитана величина метрики неустойчивости:

$$I = \frac{3}{3+3} = 0.5.$$

Мерой абстрактности компонента служит метрика A . Ее значение определяется простым отношением количества интерфейсов и абстрактных классов к общему числу классов в компоненте [14].

$$A = \frac{Na}{Nc},$$

где Na – количество абстрактных классов в компоненте;

Nc – количество классов в нем же.

Значение метрики A изменяется в диапазоне от 0 до 1. 0 означает полное отсутствие абстрактных классов в компоненте, а 1 означает, что компонент не содержит ничего, кроме абстрактных классов.

Для компонента `sport_density` была рассчитана метрика абстрактности A :

$$A = \frac{2}{7} = 0.285.$$

Вычислив меру устойчивости компоненты и ее абстрактность можно отобразить их на графике. На вертикальной оси необходимо отложить меру абстрактности A , а на горизонтальной оси – меру неустойчивости I .

Компоненты который располагаются в верхнем левом углу или в нижнем правом являются максимально качественным. На практике часто случается, так что полученная точка лежит на диагонали между точкой (1,0) и точкой (0,1). Такая диагональ называется главная последовательность (рисунок 22).

Для вычисления количественной оценки используется формула вычисления расстояние до главной последовательности:

$$D = \left| \frac{A + I - 1}{\sqrt{2}} \right|.$$

Чем ближе точка располагается к главной последовательности, тем лучше.

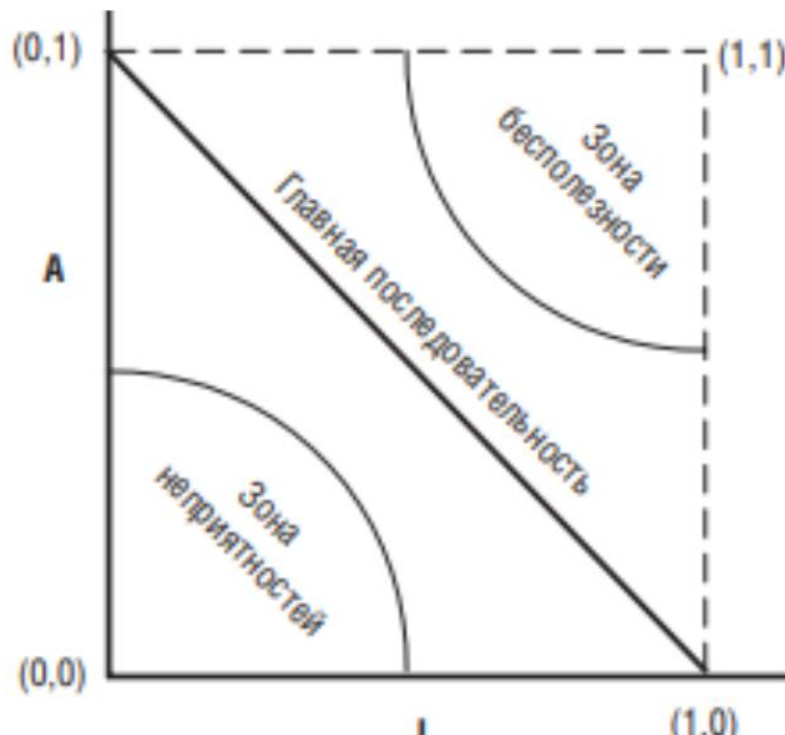


Рисунок 22 – Зоны исключения

Было вычислено расстояние до главной последовательности:

$$D = \left| \frac{0.285 + 0.5 - 1}{\sqrt{2}} \right| = 0.15.$$

Расстояние отношений метрик первой компоненты до главной последовательности достаточно близко к нулю, а значит, ее отношение абстрактности к устойчивости является правильным.

4.5 Метрики кода

Метрика программного обеспечения – мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций [15].

Был проведен анализ качества кода реализованной программы. С помощью библиотеки Radon были получены результаты вычисления индекса удобства поддержки для модулей приложения, а также индекс удобства поддержки для всего проекта. Результаты вычисления индекса приведены на рисунках 23 и 24.

```

backend/manage.py - A (85.46)
backend/sport_density/admin.py - A (100.00)
backend/sport_density/serializers.py - A (100.00)
backend/sport_density/apps.py - A (100.00)
backend/sport_density/urls.py - A (100.00)
backend/sport_density/models.py - A (42.68)
backend/sport_density/__init__.py - A (100.00)
backend/sport_density/views.py - A (30.85)
backend/leaders/urls.py - A (100.00)
backend/leaders/settings.py - A (85.90)
backend/leaders/asgi.py - A (100.00)
backend/leaders/__init__.py - A (100.00)
backend/leaders/wsgi.py - A (100.00)
backend/facilities/admin.py - A (100.00)
backend/facilities/serializers.py - A (100.00)
backend/facilities/apps.py - A (100.00)
backend/facilities/urls.py - A (100.00)
backend/facilities/models.py - A (100.00)
backend/facilities/filters.py - A (100.00)
backend/facilities/__init__.py - A (100.00)
backend/facilities/views.py - A (62.76)
backend/facilities/management/__init__.py - A (100.00)
backend/facilities/management/commands/load_sport_objects.py - A (58.53)
backend/facilities/management/commands/__init__.py - A (100.00)

```

Рисунок 23 – Результаты вычисления индекса удобства поддержки для модулей приложения

```

94 blocks (classes, functions, methods) analyzed.
Average complexity: A (1.7659574468085106)

```

Рисунок 24 – Результаты вычисления индекса удобства поддержки для проекта

Индекс удобства поддержки показывает, насколько удобно и легко поддерживать данный код. Рассчитывается по формуле:

$$MI = \text{MAX} \left(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * \frac{100}{171} \right),$$

где HV – Halstead Volume, вычислительная сложность;

CC – Cyclomatic Complexity, структурная сложность кода;

LoC – количество строк исходного кода. [16]

Нормой является значение от 20 до 100. В данном случае значения метрики для каждого модуля выше 20, что означает, что код относительно удобен в поддержке.

4.6 Планирование разработки и оценка бюджета

Планирование разработки и оценка бюджета была выполнена при помощи программы для управления проектами Microsoft Project [17]. Проект был начат 14 апреля 2022 года.

При планировании разработки было выделено 30 задач. Список задач и их описание представлено на рисунке 25.

Название задачи ▼	Длительность ▼	Начало ▼	Окончание ▼	Предшественни ▼
Начало	0 дней	Вт 01.03.22	Вт 01.03.22	
▲ Анализ требований	9 дней	Вт 01.03.22	Пт 11.03.22	
Изучение предметной области	5 дней	Вт 01.03.22	Пн 07.03.22	1
Анализ аналогичных решений	4 дней	Вт 08.03.22	Пт 11.03.22	3
Анализ требований завершен	0 дней	Пт 11.03.22	Пт 11.03.22	4
▲ Выбор средств реализации проекта	5 дней	Вт 08.03.22	Пн 14.03.22	
Анализ возможных программных средств	4 дней	Вт 08.03.22	Пт 11.03.22	3
Выбор программного средства для реализации	1 день	Пн 14.03.22	Пн 14.03.22	7
Выбор средств завершен	0 дней	Пн 14.03.22	Пн 14.03.22	8
▲ Реализация проекта	13 дней	Вт 15.03.22	Чт 31.03.22	
Разработка структуры данных	3 дней	Вт 15.03.22	Чт 17.03.22	5;9
▲ Разработка серверной части приложения	10 дней	Пт 18.03.22	Чт 31.03.22	
Программирование классов моделей данных	3 дней	Пт 18.03.22	Вт 22.03.22	11
Разработка документации API	3 дней	Ср 23.03.22	Пт 25.03.22	13
Реализация API	4 дней	Пн 28.03.22	Чт 31.03.22	14
▲ Разработка пользовательского интерфейса	10 дней	Пт 18.03.22	Чт 31.03.22	
Разработка макета интерфейса приложения	3 дней	Пт 18.03.22	Вт 22.03.22	11
Разработка приложения без взаимодействия с API	3 дней	Ср 23.03.22	Пт 25.03.22	17
Настройка взаимодействия клиентского приложения с сервером	4 дней	Пн 28.03.22	Чт 31.03.22	14;18
Реализация проекта завершена	0 дней	Чт 31.03.22	Чт 31.03.22	15;19
▲ Отладка	10 дней	Пт 01.04.22	Чт 14.04.22	
Написание unit-тестов для серверной части приложения	5 дней	Пт 01.04.22	Чт 07.04.22	20
Тестирование пользовательского интерфейса	5 дней	Пт 01.04.22	Чт 07.04.22	20
Исправление ошибок	5 дней	Пт 08.04.22	Чт 14.04.22	23;22
Тестирование завершено	0 дней	Чт 14.04.22	Чт 14.04.22	24
▲ Внедрение картографического сервиса в информационные системы города	8 дней	Пт 15.04.22	Вт 26.04.22	
Внедрение программного средства	5 дней	Пт 15.04.22	Чт 21.04.22	25
Поиск и исправления ошибок при внедрении	3 дней	Пт 22.04.22	Вт 26.04.22	27
Внедрение завершено	0 дней	Вт 26.04.22	Вт 26.04.22	28
Конец проекта	0 дней	Вт 26.04.22	Вт 26.04.22	29

Рисунок 25 – Список задач проекта

Для реализации проекта необходимо выделить основные ресурсы, их типы и стоимость. Описание ресурсов представлено в таблице 16.

Таблица 16 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Бэкенд разработчик	Трудовой	450,00 Р/ч	900,00 Р/ч
Фронтенд разработчик	Трудовой	450,00 Р/ч	900,00 Р/ч
Дизайнер	Трудовой	350,00 Р/ч	700,00 Р/ч

На рисунке 26 представлена диаграмма Ганта, построенная в среде Microsoft Project на основе введенных данных.

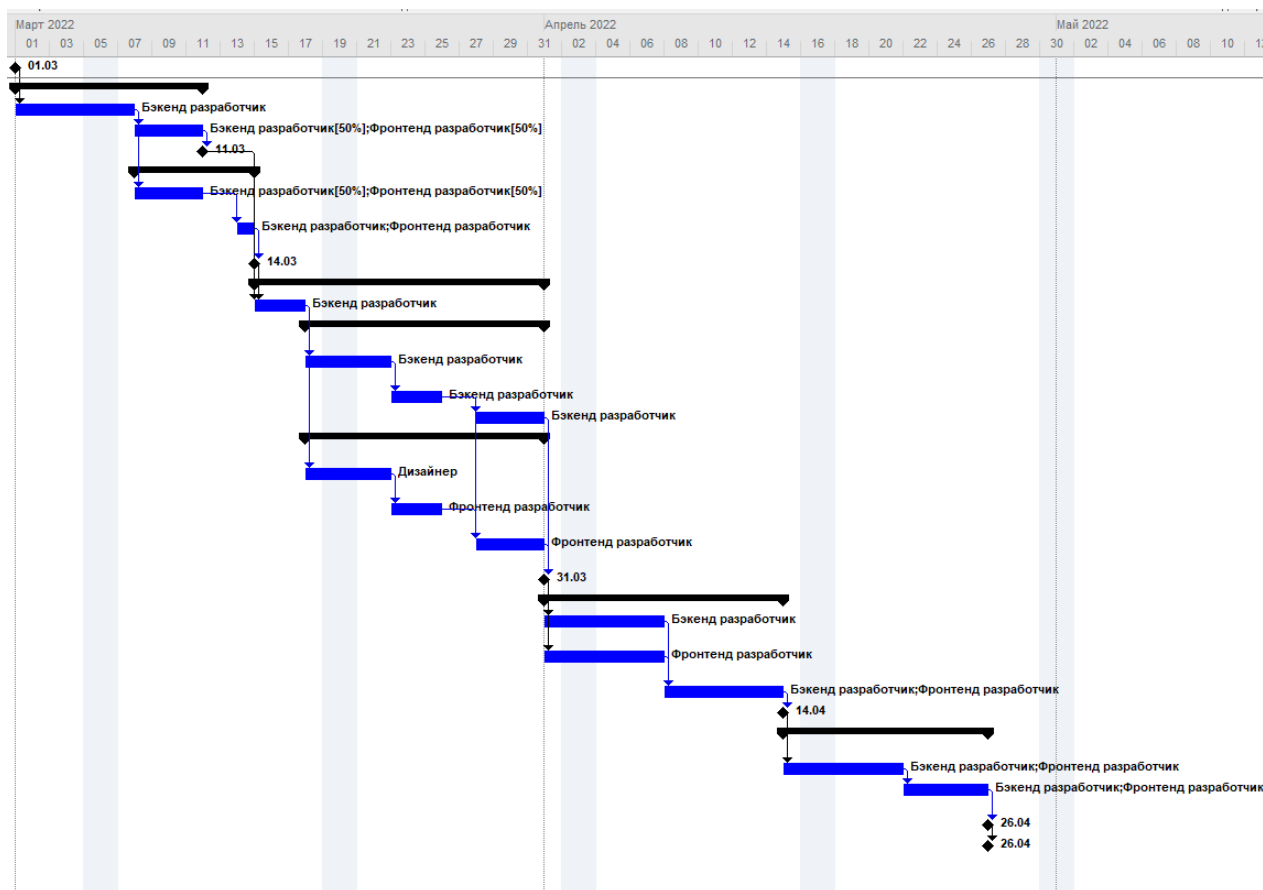


Рисунок 26 – Диаграмма Ганта

На рисунке 27 представлен обзорный отчет проекта.

ОБЗОР ЗАТРАТ

ВТ 01.03.22 - ВТ 26.04.22



СОСТОЯНИЕ ЗАТРАТ

Состояние затрат для задач верхнего уровня.

Название	Фактически затраты	Оставшиеся затраты	Базовые затраты	Затраты	Отклонение по стоимости
Начало	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽
Анализ требований	0,00 ₽	32 400,00 ₽	0,00 ₽	32 400,00 ₽	32 400,00 ₽
Выбор средств реализации проекта	0,00 ₽	21 600,00 ₽	0,00 ₽	21 600,00 ₽	21 600,00 ₽
Реализация проекта	0,00 ₽	80 400,00 ₽	0,00 ₽	80 400,00 ₽	80 400,00 ₽
Отладка	0,00 ₽	61 200,00 ₽	0,00 ₽	61 200,00 ₽	61 200,00 ₽
Внедрение картографического сервиса в информационные системы города	0,00 ₽	57 600,00 ₽	0,00 ₽	57 600,00 ₽	57 600,00 ₽
Конец проекта	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽

Рисунок 27 – Обзорный отчет проекта

Из отчета, представленного на рисунке 27, видно, что предполагаемые затраты на разработку составляют 253 200 рублей, а общая длительность проекта – 41 день.

Заключение

В результате выполнения выпускной квалификационной работы разработано и протестировано интерактивный картографический сервис размещения спортивных объектов города.

На стадии анализа требований проведен сравнительный анализ программного обеспечения аналогичного назначения, выявлены функциональные и нефункциональные требования.

Для проекта осуществлено структурное и календарное планирование, а также оценена стоимость разработки приложения. По итогу на реализацию проекта необходим 41 день и 253 200 рублей.

На стадии проектирования приложения разработана концептуальная, логическая и физическая модели базы данных, спроектирована архитектура программного продукта.

Реализация серверной части происходила в среде разработки PyCharm Professional на языке программирования Python с использованием фреймворка Django. В качестве системы управления базами данных выбрана система PostgreSQL с расширением для работы с пространственными данными PostGIS, а взаимодействия с ней производились посредством Django ORM.

Клиентская часть разрабатывалась в текстовом редакторе WebStorm на языке JavaScript с использованием фреймворка Vue.js.

Приложение тестировалось двумя видами тестирования. Серверная часть протестирована с помощью модульное тестирования, а клиентская часть с помощью функционального. Все тесты были пройдены.

Таким образом, задача, поставленная по реализации интерактивного картографического сервиса размещения спортивных объектов города, выполнена в полном объеме.

Список использованных источников

1. Географическая информационная. [Электронный ресурс]. URL: <http://www.gisa.ru/13058.html> (дата обращения: 23.04.2022).
2. Распоряжение правительства РФ от 03.06.2019 «Об утверждении Стратегии развития спортивной индустрии до 2035 года» № 1188. [Электронный ресурс]. URL: http://www.consultant.ru/document/cons_doc_LAW_326728/ (дата обращения: 23.04.2022).
3. Геоаналитика Qlik GeoAnalytics – расширение текущих возможностей Qlik в области визуализации данных на карте. [Электронный ресурс]. URL: <https://biconsult.ru/products/geoanalitika-qlik-geoanalytics-rasshirenie-tekushchih-vozmozhnostey-qlik-v-oblasti> (дата обращения 24.04.2022).
4. Возможности для работы с пространственными данными в Tableau. [Электронный ресурс]. URL: <https://www.tableau.com/solutions/maps> (дата обращения 24.04.2022).
5. Фаулер М. Архитектура корпоративных программных приложений. – Москва.: Вильямс, 2006. – ISBN 5-8459-0579-6.
6. Костюкова А. Базы данных: общий курс. Часть 1: учебное пособие. – Москва.: Академия Естествознания, 2018. – ISBN 978-5-327-538-7.
7. Документация case-средства Lucidchart. [Электронный ресурс]. URL: <https://www.lucidchart.com/pages/er-diagrams> (дата обращения 20.05.2021).
8. Моделирование данных: обзор. [Электронный ресурс]. URL: <https://habr.com/ru/post/556790/> (дата обращения 21.05.2022).
9. Верников Г. Основы методологии IDEF1X. [Электронный ресурс]. URL: <https://www.cfin.ru/vernikov/idef/idef1x.shtml> (дата обращения 24.04.2022).
10. Документация фреймворка Django. [Электронный ресурс]. URL: <https://docs.djangoproject.com/en/3.2/> (дата обращения 24.04.2022).

11. SPA (Single-page application). [Электронный ресурс]. URL: <https://developer.mozilla.org/en-US/docs/Glossary/SPA> (дата обращения 21.05.2022).
12. Куликов С. Тестирование программного обеспечения. Базовый курс (3- изд.). – Минск.: EPAM Systems, 2021. – 300 с.
13. Виды тестирования. [Электронный ресурс]. URL: <https://www.atlassian.com/ru/continuous-delivery/software-testing/types-of-software-testing> (дата обращения 21.05.2022).
14. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – Санкт-Петербург.: Питер, 2018. – ISBN 978-5-4461-0772-8.
15. Метрика программного обеспечения. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Метрика_программного_обеспечения (дата обращения 25.04.2022).
16. Средства оценки качества кода MS Visual Studio. [Электронный ресурс] URL: <https://habr.com/ru/post/111524/> (Дата обращения: 25.04.2022).
17. Скороход С. Управление проектами средствами Microsoft Project. Курс лекций. – Москва.: Интуит НОУ, 2016. – 319 с.

Приложение А
ГЛОССАРИЙ
(справочное)

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

REST API – архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных предназначенный для передачи гипертекстовых документов.

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

ORM (Object-Relational Mapping) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

СУБД – система управления базами данных.

Приложение Б
ЛИСТИНГ ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг Б.1 – Файл facilities/models.py

```

from django.contrib.gis.db import models
from django.contrib.postgres.aggregates import ArrayAgg
from django.contrib.postgres.fields import ArrayField
from django.db.models import Sum, Func, F

class AvailabilityType(models.IntegerChoices):
    city = 1
    area = 2
    district = 3
    step_availability = 4

class Department(models.Model):
    name = models.CharField(max_length=200)

    def __str__(self):
        return self.name

class Facility(models.Model):
    name = models.CharField(max_length=400)
    department = models.ForeignKey(Department, on_delete=models.CASCADE,
null=True)
    availability =
models.PositiveSmallIntegerField(choices=AvailabilityType.choices)
    placement = models.PointField()
    sports = ArrayField(models.PositiveSmallIntegerField(), size=70)

class SportType(models.Model):
    name = models.CharField(max_length=200)

    class Meta:
        db_table = "facilities_sport_type"

class SportsAreaType(models.Model):
    name = models.CharField(max_length=200)

    class Meta:
        db_table = "facilities_sports_area_type"

class SportsAreaQuerySet(models.QuerySet):
    def get_stats(self):
        return self.aggregate(
            total_sport_square=Sum("square"),
            area_types=ArrayAgg("type"),
        )

    def get_all_sports(self):
        return (
            self.annotate(all_sports=Func(F("sports"), function="unnest"))
            .values_list("all_sports", flat=True)
            .distinct()
        )

    def get_sports_coverage(self):
        return (
            self.values('type')
            .annotate(Sum('square'))

```

```

    )

class SportsArea(models.Model):
    name = models.CharField(max_length=400)
    facility = models.ForeignKey(
        Facility, on_delete=models.CASCADE, related_name="areas"
    )
    type = models.ForeignKey(SportsAreaType, on_delete=models.CASCADE)
    sports = ArrayField(models.PositiveSmallIntegerField(), size=20)
    square = models.FloatField(null=True)
    objects = SportsAreaQuerySet.as_manager()

class Meta:
    db_table = "facilities_sports_area"

```

Листинг Б.2 – Файл facilities/views.py

```

from django.contrib.postgres.aggregates import ArrayAgg
from django.db.models import Sum
from django.utils.decorators import method_decorator
from django.views.decorators.cache import cache_page
from django_filters import rest_framework as filters
from rest_framework.decorators import action
from rest_framework.pagination import LimitOffsetPagination
from rest_framework.response import Response
from rest_framework.viewsets import ReadOnlyModelViewSet
from rest_framework import mixins
from rest_framework_gis.filters import TMSTileFilter

from facilities.filters import FacilityFilter
from facilities.models import (
    Department,
    Facility,
    SportType,
    SportsAreaType,
    SportsArea,
)
from facilities.serializers import (
    DepartmentSerializer,
    FacilitySerializer,
    SportTypeSerializer,
    SportsAreaTypeSerializer,
    SportsAreaSerializer,
    FacilityDetailSerializer,
)

class DepartmentsViewSet(ReadOnlyModelViewSet):
    queryset = Department.objects.all()
    serializer_class = DepartmentSerializer

    @method_decorator(cache_page(60 * 60 * 6))
    def list(self, request, *args, **kwargs):
        return super().list(request, *args, **kwargs)

class FacilitiesViewSet(ReadOnlyModelViewSet, mixins.CreateModelMixin):
    queryset = (
        Facility.objects.prefetch_related("areas")
        .annotate(square=Sum("areas__square"))
        .all()
    )

```

```

serializer_class = FacilitySerializer
filter_backends = [filters.DjangoFilterBackend, TMSTileFilter]
filterset_class = FacilityFilter
# bbox_filter_include_overlapping = True
bbox_filter_field = 'placement'
pagination_class = LimitOffsetPagination

@method_decorator(cache_page(60 * 60 * 6))
@action(detail=True)
def report(self, request, pk=None):
    facility = self.queryset.select_related("department").get(pk=pk)
    serializer = FacilityDetailSerializer(facility)
    return Response(serializer.data)

def get_queryset(self):
    if self.action == "list":
        return self.queryset.annotate(
            area_types=ArrayAgg("areas__type", distinct=True)
        )
    return super(FacilitiesViewSet, self).get_queryset()

@method_decorator(cache_page(60 * 60 * 2))
def list(self, request, *args, **kwargs):
    return super(FacilitiesViewSet, self).list(request, *args, **kwargs)

class SportTypesViewSet(ReadOnlyModelViewSet):
    queryset = SportType.objects.all()
    serializer_class = SportTypeSerializer

class SportsAreaTypesViewSet(ReadOnlyModelViewSet):
    queryset = SportsAreaType.objects.all()
    serializer_class = SportsAreaTypeSerializer

@method_decorator(cache_page(60 * 60 * 6))
def list(self, request, *args, **kwargs):
    return super().list(request, *args, **kwargs)

class SportsAreasViewSet(ReadOnlyModelViewSet):
    queryset = SportsArea.objects.all()
    serializer_class = SportsAreaSerializer
    filter_backends = [filters.DjangoFilterBackend]
    filterset_fields = ["facility", "type"]
    pagination_class = LimitOffsetPagination

@method_decorator(cache_page(60 * 60 * 6))
def list(self, request, *args, **kwargs):
    return super().list(request, *args, **kwargs)

```

Листинг Б.3 – Файл facilities/serializers.py

```

from rest_framework import serializers
from rest_framework.serializers import ModelSerializer

from facilities.models import (
    Department,
    Facility,
    SportType,
    SportsAreaType,
    SportsArea,
)

```

```

class DepartmentSerializer(ModelSerializer):
    class Meta:
        model = Department
        fields = "__all__"

class SportsAreaSerializer(ModelSerializer):
    class Meta:
        model = SportsArea
        fields = "__all__"

class FacilitySerializer(ModelSerializer):
    square = serializers.FloatField()

    class Meta:
        model = Facility
        fields = ("id", "name", "availability", "placement", "department",
"square")

    def update(self, instance, validated_data):
        instance.name = validated_data.get('name', instance.name)
        instance.availability = validated_data.get('availability',
instance.availability)
        instance.placement = validated_data.get('placement',
instance.placement)
        instance.department = validated_data.get('department',
instance.department)
        instance.square = validated_data.get('square', instance.square)
        instance.save()
        return instance

    def create(self, validated_data):
        instance = Facility.objects.create(
            name=validated_data.get('name'),
            availability=validated_data.get('availability'),
            placement=validated_data.get('placement'),
            department=validated_data.get('department'),
            square=validated_data.get('square'),
        )
        return instance

class SportTypeSerializer(ModelSerializer):
    class Meta:
        model = SportType
        fields = "__all__"

class SportsAreaTypeSerializer(ModelSerializer):
    class Meta:
        model = SportsAreaType
        fields = "__all__"

class FacilityDetailSerializer(ModelSerializer):
    areas = SportsAreaSerializer(many=True, read_only=True)
    department = DepartmentSerializer(read_only=True)

    class Meta:
        model = Facility
        fields = ("name", "availability", "areas", "department")

```

Листинг Б.4 – Файл facilities/filters.py

```

from django import forms
from django_filters import rest_framework as filters

from facilities.models import Facility, AvailabilityType

class NoValidationMultipleChoiceField(forms.MultipleChoiceField):
    def validate(self, value):
        return True

class SportsFilter(filters.Filter):
    field_class = NoValidationMultipleChoiceField

    def __init__(self, *args, **kwargs):
        self.conjoined = kwargs.pop('conjoined', False)
        super().__init__(*args, **kwargs)

    def filter(self, qs, values):
        return qs.filter(areas__sports__contains=[int(value) for value in
values])

class FacilityFilter(filters.FilterSet):
    name = filters.CharFilter(field_name="name", lookup_expr="icontains")
    area_name = filters.CharFilter(field_name="areas__name",
lookup_expr="icontains")
    area_type = filters.BaseInFilter(field_name="area_types",
lookup_expr="contains")
    availability = filters.MultipleChoiceFilter(
        field_name="availability", choices=AvailabilityType.choices
    )
    sports = SportsFilter()

    class Meta:
        model = Facility
        fields = [
            "name",
            "department",
            "availability",
            "area_name",
            "area_type",
            "sports",
        ]

```

Листинг Б.5 – Файл facilities/tests.py

```

from typing import TYPE_CHECKING

import pytest

if TYPE_CHECKING:
    from rest_framework.test import APIClient

@pytest.mark.django_db
def test_big_hex_list(client: 'APIClient', hex_without_facilities,
hex_with_facilities, hex_without_population):
    response = client.get('/api/v1/data-hex/big-hex')
    assert response.status_code == 200
    data = response.json()

```

```

assert data['count'] == 2
ids = {hex_['id'] for hex_ in data['results']}
assert ids == {1, 2}
hexes = sorted(data['results'], key=lambda hex_: hex_['id'])
response_hex_without_facilities = hexes[0]
response_hex_with_facilities = hexes[1]
assert response_hex_without_facilities['square'] is None
assert response_hex_without_facilities['square_by_person'] is None
assert response_hex_without_facilities['areas_count'] == 0
assert response_hex_with_facilities['square'] == 1100
assert response_hex_with_facilities['square_by_person'] == 1.1
assert response_hex_with_facilities['areas_count'] == 2

@pytest.mark.django_db
def test_big_hex_hexes_report(client: 'APIClient', hex_without_facilities,
hex_with_facilities, hex_without_population):
    response = client.get('/api/v1/data-hex/big-hex/hexes-
report?ids=1&ids=2&ids=3')
    assert response.status_code == 200
    data = response.json()
    assert len(data['polygon']['coordinates']) == 3
    del data['polygon']
    assert data == {
        'total_square': 2.2119828, # 3 hexagons square by h3
        'total_population': 1100,
        'area_types_counter': {
            '1': 1,
            '2': 1
        },
        'total_sport_square': 1100.0,
        'sports': [1, 2, 3],
        'area_types_coverage': {
            '1': 1000.0,
            '2': 100.0
        }
    }

@pytest.mark.django_db
def test_big_hex_sport_density(client: 'APIClient', hex_without_facilities,
hex_with_facilities, hex_without_population):
    response = client.get('/api/v1/data-hex/big-hex/sport-density')
    assert response.status_code == 200
    data = response.json()
    assert data['count'] == 3
    hexes = sorted(data['results'], key=lambda hex_: hex_['id'])
    response_hex_without_facilities = hexes[0]
    response_hex_with_facilities = hexes[1]
    response_hex_without_population = hexes[2]

    assert response_hex_without_facilities['square'] is None
    assert response_hex_without_facilities['population'] == 100
    assert response_hex_without_facilities['areas_count'] == 0

    assert response_hex_with_facilities['square'] == 1100
    assert response_hex_with_facilities['population'] == 1000
    assert response_hex_with_facilities['areas_count'] == 2

    assert response_hex_without_population['square'] is None
    assert response_hex_without_population['population'] == 0
    assert response_hex_without_population['areas_count'] == 0

```

```

@pytest.mark.django_db
def test_big_hex_population_density(
    client: 'APIClient', hex_without_facilities, hex_with_facilities,
    hex_without_population):
    response = client.get('/api/v1/data-hex/big-hex/population-density')
    assert response.status_code == 200
    data = response.json()
    assert data['count'] == 2
    hexes = sorted(data['results'], key=lambda hex_: hex_['id'])
    response_hex_without_facilities = hexes[0]
    response_hex_with_facilities = hexes[1]

    assert response_hex_without_facilities['population'] == 100
    assert response_hex_with_facilities['population'] == 1000

```

Листинг Б.6 – Файл facilities/urls.py

```

from rest_framework.routers import DefaultRouter

from facilities.views import (
    DepartmentsViewSet, FacilitiesViewSet, SportTypesViewSet,
    SportsAreaTypesViewSet,
    SportsAreasViewSet
)

router = DefaultRouter(trailing_slash=False)
router.register('departments', DepartmentsViewSet)
router.register('facilities', FacilitiesViewSet)
router.register('sport-types', SportTypesViewSet)
router.register('sport-areas', SportsAreasViewSet)
router.register('sport-area-types', SportsAreaTypesViewSet)

urlpatterns = [
    *router.urls
]

```

Листинг Б.7 – Файл sport_density/models.py

```

from typing import List

import h3
import numpy as np
from django.contrib.gis.db import models
from django.contrib.postgres.fields import ArrayField
from django.db.models import Sum, Count, F, Q

from facilities.models import Facility

def get_areas_filter(sport_ids=None, area_type=None, availability=None,
    department=None):
    areas_filter = None
    filters = []
    if sport_ids:
        filters.append(Q(facilities__areas__sports__contains=sport_ids))
    if area_type is not None:
        filters.append(Q(facilities__areas__type=area_type))
    if availability is not None:
        filters.append(Q(facilities__availability=availability))
    if department is not None:
        filters.append(Q(facilities__department=department))

    if filters:

```

```

        areas_filter = filters[0]
        for filter_ in filters:
            areas_filter &= filter_
    return areas_filter

def calculate_bins(values, quantiles):
    values = [value for value in values if value]
    if not values:
        return []
    return list(np.quantile(values, quantiles))

class HexesQuerySet(models.QuerySet):
    def annotate_areas(self, areas_filter=None):
        return self.annotate(
            square=Sum("facilities__areas__square", filter=areas_filter),
            areas_count=Count("facilities__areas", filter=areas_filter),
        )

    def annotate_square_by_person(self):
        return self.annotate(square_by_person=F("square") / F("population"))

    def populated(self):
        return self.filter(population__gt=0)

    def color_bins_by_square(self, areas_filter, quantiles=None) ->
    List[int]:
        if not quantiles:
            quantiles = np.linspace(0, 1, 11)
        squares = self.annotate_areas(areas_filter).values_list('square',
flat=True)
        return calculate_bins(squares, quantiles)

    def color_bins_by_square_per_person(self, areas_filter, quantiles=None) -
> List[int]:
        if not quantiles:
            quantiles = np.linspace(0, 1, 11)
        squares = (
            self.populated()
            .annotate_areas(areas_filter)
            .annotate_square_by_person()
            .values_list('square_by_person', flat=True)
        )
        return calculate_bins(squares, quantiles)

    def to_h3_list(self, resolution):
        hexes = []
        for hex_ in self:
            hex_ = h3.geo_to_h3(
                hex_.polygon.centroid.x, hex_.polygon.centroid.y, resolution
            )
            hexes.append(hex_)
        return hexes

class DataHexSmall(models.Model):
    polygon = models.PolygonField()
    population = models.PositiveIntegerField()
    facilities = models.ManyToManyField(Facility, related_name='hexes')
    objects = HexesQuerySet.as_manager()

class Meta:
    db_table = 'sport_density_data_hex_small'

```

```

class DataHexBig(models.Model):
    polygon = models.PolygonField()
    population = models.PositiveIntegerField()
    facilities = models.ManyToManyField(Facility, related_name='big_hexes')
    objects = HexesQuerySet.as_manager()

    class Meta:
        db_table = 'sport_density_data_hex_big'

class ColorBins(models.Model):
    sport_id = models.PositiveIntegerField(null=True)
    availability = models.PositiveIntegerField(null=True)
    is_big_hexes = models.BooleanField(default=True)
    bins = ArrayField(models.IntegerField(), size=20)
    area_type = models.PositiveIntegerField(null=True)

    class Meta:
        abstract = True

class SquareColorBins(ColorBins):
    class Meta:
        db_table = 'sport_density_square_color_bins'

class SquarePerPersonColorBins(ColorBins):
    class Meta:
        db_table = 'sport_density_square_per_person_color_bins'

```

Листинг Б.8 – Файл sport_density /views.py

```

from abc import ABC, abstractmethod
from collections import Counter

import h3
from django.utils.decorators import method_decorator
from django.views.decorators.cache import cache_page
from django_filters import rest_framework as filters
from rest_framework.decorators import action
from rest_framework.pagination import LimitOffsetPagination
from rest_framework.response import Response
from rest_framework.viewsets import ReadOnlyModelViewSet
from rest_framework_gis.filters import TMSTileFilter, InBoundingBoxFilter

from facilities.management.commands.load_sport_objects import
calculate_color_bins_for_hexes_by_square, \
    calculate_color_bins_for_hexes_by_square_per_person
from facilities.models import SportsArea
from sport_density.models import DataHexSmall, DataHexBig, get_areas_filter,
SquareColorBins, SquarePerPersonColorBins
from sport_density.serializers import (
    DataHexSmallSerializer,
    BaseHexIntersectionsSerializer,
    DataHexBigSerializer,
    BaseDataHexSmallSerializer,
    BaseDataHexBigSerializer,
    DetailDataHexBigSerializer,
    DetailDataHexSmallSerializer, SquareColorBinsSerializer,
    SquarePerPersonColorBinsSerializer,
)

```

```

class BaseHexViewSet(ReadOnlyModelViewSet, ABC):
    filter_backends = [TMSTileFilter, InBBBoxFilter]
    bbox_filter_field = "polygon"
    bbox_filter_include_overlapping = True
    pagination_class = LimitOffsetPagination
    h3_resolution = 8

    @abstractmethod
    def get_areas(self):
        pass

    def get_queryset(self):
        if self.action == "sport_density":
            return self.queryset.annotate_areas()
        if self.action in ("hexes_report", "population_density", "list"):
            return self.queryset

        return (
            self.queryset
            .populated()
            .annotate_areas()
            .annotate_square_by_person()
        )

    def filter_queryset(self, queryset):
        if self.action in ("population_density", "list"):
            queryset = queryset.populated()
        if self.action == "hexes_report":
            hex_ids = self.request.query_params.getlist("ids", [])
            queryset = queryset.filter(pk__in=[int(hex_id) for hex_id in
hex_ids])
        elif self.action in ("list", "sport_density"):
            sport_ids = self.request.query_params.getlist("sports", [])
            area_type = self.request.query_params.get("area_type", None)
            availability = self.request.query_params.get("availability",
None)

            department = self.request.query_params.get("department", None)
            areas_filter = get_areas_filter(
                sport_ids=sport_ids,
                area_type=area_type,
                availability=availability,
                department=department
            )
            queryset = queryset.annotate_areas(areas_filter=areas_filter)
            if self.action == 'list':
                queryset = queryset.annotate_square_by_person()

        return super().filter_queryset(queryset)

    @action(detail=False, url_path="sport-density")
    def sport_density(self, request, *args, **kwargs):
        return self.list(request, *args, **kwargs)

    @action(detail=False, url_path="population-density")
    def population_density(self, request, *args, **kwargs):
        return self.list(request, *args, **kwargs)

    @method_decorator(cache_page(60 * 60 * 6))
    @action(detail=False, url_path="hexes-report")
    def hexes_report(self, request, *args, **kwargs):
        qs = self.filter_queryset(self.get_queryset())
        if not qs:

```

```

        return Response([])
    areas = self.get_areas()
    report_generator = HexReportGenerator()
    report = report_generator.generate(qs, areas, self.h3_resolution)
    return Response(report)

    @method_decorator(cache_page(60 * 60 * 6))
    def list(self, request, *args, **kwargs):
        return super().list(request, *args, **kwargs)

class HexReportGenerator:
    @staticmethod
    def generate(hexes, areas, h3_resolution) -> dict:
        total_population = 0
        for hex_ in hexes:
            total_population += hex_.population

        hexes = hexes.to_h3_list(h3_resolution)
        total_square = h3.hex_area(h3_resolution) * len(hexes)

        all_sports = list(set(areas.get_all_sports()))
        area_types_coverage = {}
        for row in areas.get_sports_coverage():
            area_types_coverage[row["type"]] = row["square__sum"]
        areas_stats = areas.get_stats()
        area_types_counter = Counter(areas_stats["area_types"])
        polygon = h3.h3_set_to_multi_polygon(hexes, geo_json=True)
        return {
            "polygon": {
                "type": "MultiPolygon",
                "coordinates": polygon,
            },
            "total_square": total_square,
            "total_population": total_population,
            "area_types_counter": area_types_counter,
            "total_sport_square": areas_stats["total_sport_square"],
            "sports": all_sports,
            "area_types_coverage": area_types_coverage,
        }

class SmallHexViewSet(BaseHexViewSet):
    queryset = DataHexSmall.objects.all()
    serializer_class = DetailDataHexSmallSerializer
    h3_resolution = 9

    def get_serializer_class(self):
        if self.action == "sport_density":
            return DataHexSmallSerializer
        if self.action == "population_density":
            return BaseDataHexSmallSerializer
        return self.serializer_class

    def get_areas(self):
        return SportsArea.objects.filter(
            facility__hexes__id__in=self.request.query_params.getlist("ids")
        ).distinct()

class BigHexViewSet(BaseHexViewSet):
    queryset = DataHexBig.objects.all()
    serializer_class = DetailDataHexBigSerializer
    h3_resolution = 8

```

```

def get_serializer_class(self):
    if self.action == "sport_density":
        return DataHexBigSerializer
    if self.action == "population_density":
        return BaseDataHexBigSerializer
    return self.serializer_class

def get_areas(self):
    return SportsArea.objects.filter(

facility__big_hexes__id__in=self.request.query_params.getlist("ids")
    ).distinct()

class HexIntersectionViewSet(ReadOnlyModelViewSet):
    queryset = DataHexSmall.objects.all()
    serializer_class = BaseHexIntersectionsSerializer
    filter_backends = [TMSTileFilter, InBBBoxFilter]
    bbox_filter_field = "polygon"
    bbox_filter_include_overlapping = True
    pagination_class = LimitOffsetPagination

class ColorBinsViewSet(ReadOnlyModelViewSet, ABC):
    queryset = SquareColorBins.objects.all()
    serializer_class = SquareColorBinsSerializer
    pagination_class = LimitOffsetPagination
    filter_backends = (filters.DjangoFilterBackend,)
    filterset_fields = ("sport_id", "availability", "area_type",
"is_big_hexes")
    calculation_func = calculate_color_bins_for_hexes_by_square
    model = SquareColorBins

    @method_decorator(cache_page(60 * 60 * 6))
    def list(self, request, *args, **kwargs):
        sport_id = self.request.query_params.get("sport_id", None)
        availability = self.request.query_params.get("availability", None)
        is_big_hexes = self.request.query_params.get("is_big_hexes", None)
        area_type = self.request.query_params.get("area_type", None)
        if sport_id is not None:
            sport_id = int(sport_id)
            sport_ids = [sport_id]
        else:
            sport_ids = None
        if availability:
            availability = int(availability)
        if area_type:
            area_type = int(area_type)
        if is_big_hexes is not None:
            is_big_hexes = is_big_hexes == 'true'

        queryset = self.get_queryset().filter(
            sport_id=sport_id, availability=availability,
area_type=area_type, is_big_hexes=is_big_hexes
        )
        if not queryset:
            bins = self.__class__.calculation_func(sport_ids, availability,
is_big_hexes, area_type=area_type)
            queryset = self.model.objects.create(
                sport_id=sport_id,
                availability=availability,
                bins=bins,
                is_big_hexes=is_big_hexes

```

```

        )
        queryset = [queryset]

        serializer = self.get_serializer(queryset, many=True)
        return Response(serializer.data)

class SquareColorBinsViewSet(ColorBinsViewSet):
    queryset = SquareColorBins.objects.all()
    model = SquareColorBins
    serializer_class = SquareColorBinsSerializer
    calculation_func = calculate_color_bins_for_hexes_by_square

class SquarePerPersonColorBinsViewSet(ColorBinsViewSet):
    queryset = SquarePerPersonColorBins.objects.all()
    model = SquarePerPersonColorBins
    serializer_class = SquarePerPersonColorBinsSerializer
    calculation_func = calculate_color_bins_for_hexes_by_square_per_person

```

Листинг Б.9 – Файл sport_density/urls.py

```

from rest_framework.routers import DefaultRouter

from sport_density.views import SmallHexViewSet, HexIntersectionViewSet,
BigHexViewSet, SquareColorBinsViewSet, \
    SquarePerPersonColorBinsViewSet

router = DefaultRouter(trailing_slash=False)
router.register('small-hex', SmallHexViewSet)
router.register('big-hex', BigHexViewSet)
router.register('small-hex-intersections', HexIntersectionViewSet)
router.register('color-bins-by-square', SquareColorBinsViewSet)
router.register('color-bins-by-square-per-person',
SquarePerPersonColorBinsViewSet)

urlpatterns = [
    *router.urls
]

```

Листинг Б.10 – Файл sport_density/serializers.py

```

from rest_framework import serializers
from rest_framework.serializers import ModelSerializer

from sport_density.models import DataHexSmall, DataHexBig, SquareColorBins,
SquarePerPersonColorBins

class BaseDataHexSmallSerializer(ModelSerializer):
    class Meta:
        model = DataHexSmall
        fields = ['id', 'polygon', 'population']

class DataHexSmallSerializer(ModelSerializer):
    square = serializers.FloatField()
    areas_count = serializers.IntegerField()

    class Meta:
        model = DataHexSmall
        fields = ['id', 'polygon', 'population', 'square', 'areas_count']

```

```

class DetailDataHexSmallSerializer(DataHexSmallSerializer):
    square_by_person = serializers.FloatField()

    class Meta:
        model = DataHexSmall
        fields = ['id', 'polygon', 'population', 'square', 'areas_count',
'square_by_person']

class BaseDataHexBigSerializer(ModelSerializer):
    class Meta:
        model = DataHexBig
        fields = ['id', 'polygon', 'population']

class DataHexBigSerializer(ModelSerializer):
    square = serializers.FloatField()
    areas_count = serializers.IntegerField()

    class Meta:
        model = DataHexBig
        fields = ['id', 'polygon', 'population', 'square', 'areas_count']

class DetailDataHexBigSerializer(DataHexBigSerializer):
    square_by_person = serializers.FloatField()

    class Meta:
        model = DataHexBig
        fields = ['id', 'polygon', 'population', 'square', 'areas_count',
'square_by_person']

class BaseHexIntersectionsSerializer(ModelSerializer):

    class Meta:
        model = DataHexSmall
        fields = "__all__"

class SquareColorBinsSerializer(ModelSerializer):

    class Meta:
        model = SquareColorBins
        fields = ("sport_id", "availability", "area_type", "is_big_hexes",
"bins")

class SquarePerPersonColorBinsSerializer(ModelSerializer):

    class Meta:
        model = SquarePerPersonColorBins
        fields = ("sport_id", "availability", "area_type", "is_big_hexes",
"bins")

```

Листинг Б.11 – Файл manage.py

```

#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

```

```
def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'leaders.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

Листинг Б.12 – Файл conftest.py

```
import pytest
from django.contrib.gis.geos import Point

from facilities.management.commands.load_sport_objects import
load_zone_types, load_sport_types, load_departments
from facilities.models import Facility, SportsArea, AvailabilityType
from sport_density.models import DataHexBig

@pytest.fixture(autouse=True)
def enable_db_access_for_all_tests(db):
    pass

@pytest.fixture(autouse=True)
def initial():
    load_zone_types()
    load_sport_types()
    load_departments()

# 55.869061,37.63058
@pytest.fixture
def facility_with_areas():
    facility = Facility.objects.create(
        id=1,
        name="Test1",
        department_id=219165,
        availability=AvailabilityType.city,
        placement=Point(x=55.866005, y=37.634515),
        sports=[1, 2, 3]
    )
    SportsArea.objects.create(
        name="Test1",
        facility=facility,
        type_id=1,
        sports=[1, 2],
        square=1000,
    )
    SportsArea.objects.create(
        name="Test2",
        facility=facility,
        type_id=2,
```

```

        sports=[2, 3],
        square=100,
    )
    facility.refresh_from_db()
    return facility

@pytest.fixture
def hex_without_facilities():
    return DataHexBig.objects.create(
        id=1,
        polygon='POLYGON ((55.55732563770245 37.23141540684026,
55.56172093174474 37.23484333250231, 55.56563267989374 37.23042979464858,
55.56514868209219 37.22258820378101, 55.56075326101143 37.21916165066352,
55.55684196474057 37.22357531567695, 55.55732563770245 37.23141540684026))',
        population=100
    )

@pytest.fixture
def hex_with_facilities(facility_with_areas):
    hex_ = DataHexBig.objects.create(
        id=2,
        polygon='POLYGON ((55.86170816602239 37.62929451099481,
55.86217628266556 37.63722430227906, 55.8665684387861 37.64073291417827,
55.87049261371315 37.63631035111111, 55.87002417880192 37.62837902353578,
55.86563188726147 37.62487179551921, 55.86170816602239 37.62929451099481))',
        population=1000
    )
    hex_.facilities.add(facility_with_areas)
    return hex_

@pytest.fixture
def hex_without_population():
    hex_ = DataHexBig.objects.create(
        id=3,
        polygon='POLYGON ((55.45473611716478 37.19629017932863,
55.45082302749073 37.20069386145767, 55.45130474301492 37.20851211662382,
55.45569987354988 37.21192818071274, 55.45961341496008 37.20752462380547,
55.45913137406909 37.19970487739715, 55.45473611716478 37.19629017932863))',
        population=0
    )
    return hex_

```

Листинг Б.13 – Файл config.json

```

{
    "listeners": {
        " *:8000": {
            "pass": "routes"
        }
    },
    "routes": [
        {
            "match": {
                "uri": "/django_static/*"
            },
            "action": {
                "share": "/app/static/"
            }
        }
    ],
}

```

```

    {
      "match": {
        "uri": "/admin/*"
      },
      "action": {
        "pass": "applications/django"
      }
    },
    {
      "match": {
        "uri": "/report/*"
      },
      "action": {
        "pass": "applications/django"
      }
    },
    {
      "match": {
        "uri": "/api/*"
      },
      "action": {
        "pass": "applications/django"
      }
    },
    {
      "action": {
        "share": "/app/frontend_build/"
      }
    }
  ],
  "applications": {
    "django": {
      "type": "python 3.9",
      "path": "/app/",
      "module": "leaders.wsgi",
      "environment": {
        "DJANGO_SETTINGS_MODULE": "leaders.settings"
      }
    }
  }
}

```

Листинг Б.14 – Файл Dockerfile

```

FROM node:12.22-buster as frontend
RUN mkdir /app
WORKDIR /app
COPY ./frontend/yarn.lock /app/yarn.lock
COPY ./frontend/package.json /app/package.json
RUN yarn
COPY ./frontend /app
RUN yarn build

FROM nginx/unit:1.25.0-python3.9 as api
RUN mkdir /app
WORKDIR /app

RUN apt update
RUN apt install -y libpq-dev curl gcc gdal-bin
RUN curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/get-poetry.py | python -

```

```

COPY ./backend/poetry.toml /app/poetry.toml
COPY ./backend/pyproject.toml /app/pyproject.toml
COPY ./backend/poetry.lock /app/poetry.lock
RUN $HOME/.poetry/bin/poetry install
COPY ./backend /app
RUN python manage.py collectstatic --noinput
COPY --from=frontend /app/dist/ /app/frontend_build/

```

Листинг Б.15 – Файл docker-compose.yaml

```

version: "3.5"

services:
  db:
    image: postgis/postgis:latest
    container_name: db
    env_file:
      - backend/.env
    volumes:
      - db:/data/db
    environment:
      POSTGRES_USER: ${DB_USER:-admin}
      POSTGRES_PASSWORD: ${DB_PASSWORD:-admin}
      POSTGRES_DB: "geo"
    ports:
      - "5432:5432"

  api:
    build: .
    environment:
      DB_USER: ${DB_USER:-admin}
      DB_PASSWORD: ${DB_PASSWORD:-admin}
      DB_HOST: ${DB_HOST:-db}
    volumes:
      - ./config.json:/var/lib/unit/conf.json
    ports:
      - "80:8000"

volumes:
  db: {}

```

Приложение В
ЛИСТИНГ ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг В.1 – Файл index.html

```
<!DOCTYPE html>
<html lang="">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link rel="icon" href="<%= BASE_URL %>favicon.ico">
    <link
href="https://fonts.googleapis.com/css?family=Roboto:100,300,400,500,700,900"
rel="stylesheet">
    <link
href="https://cdn.jsdelivrivr.net/npm/@mdi/font@6.x/css/materialdesignicons.min.
css" rel="stylesheet">
    <title>Интерактивная карта для размещения спортивной
инфраструктуры</title>
  </head>
  <body>
    <noscript>
      <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't
work properly without JavaScript enabled. Please enable it to
continue.</strong>
    </noscript>
    <div id="app"></div>
    <!-- built files will be auto injected -->
  </body>
</html>
```

Листинг В.2 – Файл App.vue

```
<template>
  <div id="app">
    <router-view />
  </div>
</template>

<script>
export default {
  name: 'App',
  async mounted() {
    this.$store.dispatch("getDepartments");
    this.$store.dispatch("getAreaTypes");
    this.$store.dispatch("getSportTypes");
  },
}
</script>

<style>
#app {
  width: 100%;
  height: 100%;
}
</style>
```

Листинг В.3 – Файл main.js

```
import Vue from 'vue'
import App from './App.vue'
import store from "./core/store.js"
import vuetify from '@/plugins/vuetify'
import router from '@/plugins/router'
```

```

import { Icon } from 'leaflet';

Vue.config.productionTip = false

new Vue({
  store,
  vuetify,
  router,
  render: h => h(App),
}).$mount('#app')

delete Icon.Default.prototype._getIconUrl;
Icon.Default.mergeOptions({
  iconRetinaUrl: require('leaflet/dist/images/marker-icon-2x.png'),
  iconUrl: require('leaflet/dist/images/marker-icon.png'),
  shadowUrl: require('leaflet/dist/images/marker-shadow.png'),
});

```

Листинг В.4 – Файл router.js

```

import Vue from 'vue'
import VueRouter from 'vue-router'
import Main from '../components/Main'
import Report from '../components/Report'

const routes = [
  { path: '/', component: Main },
  { path: '/report/:isBig/:ids', component: Report }
]

const router = new VueRouter({
  routes,
  mode: "history",
})

Vue.use(VueRouter)

export default router

```

Листинг В.5 – Файл Main.vue

```

<template>
  <v-app>
    <MainMenu />
    <v-main>
      <MainMap />
    </v-main>
  </v-app>
</template>
<script>
import MainMap from "../MainMap";
import MainMenu from "../MainMenu";
export default {
  components: {
    MainMap,
    MainMenu,
  },
};
</script>
<style>
</style>

```

Листинг В.6 – Файл MainMap.vue

```

<template>
  <l-map
    ref="map"
    style="height: 100%; position: relative; z-index: 1"
    :zoom="startZoom"
    :minZoom="11"
    :maxZoom="18"
    :center="center"
    @update:bounds="update"
    @update:zoom="onZoom"
    @overlayadd="overlayadd"
    @overlayremove="overlayremove"
    @baselayerchange="baselayerchange"
    :options="{zoomControl: false}"
  >
    <l-control-zoom position="bottomright"></l-control-zoom>
    <l-control position="topright" >
      <v-btn-toggle multiple class="selectBtn" >
        <v-btn @click='handleLayerControl' >
          <v-icon small>mdi-layers-triple-outline</v-icon>
        </v-btn>
        <v-btn @click='multiSelect = !multiSelect' :disabled="baseLayer &&
baseLayer.name !== layerNames[3]">
          <v-icon small>mdi-select-multiple</v-icon>
        </v-btn>
      </v-btn-toggle>
    </l-control>
    <l-control-layers ref="control" :collapsed="false"
position="topright"></l-control-layers>
    <l-control v-if="selectedHexes.length && multiSelect"
position="topright">
      <v-btn
        :to="{ path:
`/report/${isBigHexes}/${JSON.stringify(selectedHexes.map(el=>el.hexPoly.id))
}`}"
        target="_blank"
        depressed
        fab
        height="40px"
        width="40px"
        class="mr-2"
      >
        <v-icon>mdi-chart-line</v-icon>
      </v-btn>
    </l-control>

    <l-control position="bottomright" v-if="baseLayer && baseLayer.name !==
'Отключено'">
      <div v-if="!showHexLegend">
        <v-sheet width="150" class="legend" rounded>
          <div>
            <div v-if="baseLayer.name === layerNames[1]" >
              <div class="text-caption">Население, чел</div>
              <div v-for="(el,idx) in hexBins.slice(0,hexBins.length - 1)"
:key="idx">
                <i :style="{background: colors[0][idx+1]}"></i>
                {{Math.round(el)}} {{hexBins[idx + 1] ? ' &ndash; ' +
Math.round(hexBins[idx + 1]) : "+"}}<br>
              </div>
            </div>
            <div v-if="baseLayer.name === layerNames[2]" >

```

```

        <div class="text-caption">Площадь спортобъектов,
м<sup>2</sup></div>
        <div v-for="(el,idx) in hexBins.slice(0,hexBins.length - 1)"
:key="idx">
            <i :style="{background: colors[1][idx+1]}"></i>
            {{Math.round(el)}} {{hexBins[idx + 1] ? ' &ndash; ' +
Math.round(hexBins[idx + 1]) : "+"}}<br>
        </div>
    </div>
    <div v-if="baseLayer.name === layerNames[3]" >
        <div class="text-caption">Площадь спортобъектов на человека,
м<sup>2</sup>/чел.</div>
        <div v-for="(el,idx) in hexBins.slice(0,hexBins.length - 1)"
:key="idx">
            <i :style="{background: colors[1][idx+1]}"></i>
            {{el.toFixed(2)}} {{hexBins[idx + 1] ? ' &ndash; ' +
hexBins[idx + 1].toFixed(2) : "+"}}<br>
        </div>
    </div>
</v-sheet>
</div>
<div v-else>
    <v-sheet class="hexLegend" rounded>
        <div class="text-caption">Население:
{{this.selectedHexes[0].hexPoly.population}} чел.</div>
        <div v-if="this.selectedHexes[0].hexPoly.square ||
this.selectedHexes[0].hexPoly.square === null" class="text-caption">Площадь
спортзон:
            {{Math.round(this.selectedHexes[0].hexPoly.square)}}
м<sup>2</sup></div>
        <div v-if="this.selectedHexes[0].hexPoly.areas_count ||
this.selectedHexes[0].hexPoly.areas_count === 0" class="text-
caption">Количество спортивных зон:
            {{Math.round(this.selectedHexes[0].hexPoly.areas_count)}}
        </div>
        <div v-if="this.selectedHexes[0].hexPoly.square &&
this.selectedHexes[0].hexPoly.population" class="text-caption">Площадь
спортзон на человека:
            {{(this.selectedHexes[0].hexPoly.square /
this.selectedHexes[0].hexPoly.population).toFixed(2)}} м<sup>2</sup>/чел.
        </div>
    </v-sheet>
</div>
</l-control>

<l-tile-layer ref="tileLayer" :url="url" :attribution="attribution"></l-
tile-layer>
<l-layer-group ref="circles" :visible='true' :name="layerNames[0]" layer-
type="overlay">
</l-layer-group>
<l-layer-group
name='Отключено'
layer-type="base"
:visible='true'
></l-layer-group>
<hexes-layer-group
:name="layerNames[1]"
:hexes="baseLayer && baseLayer.name === layerNames[1]? hexes : []"
:colormap="colors[0]"
color="red"
:bins="hexBins"
opacityBy="population"

```

```

      layerType="base"
    >
      <template v-slot:popup="{prop}">
        <p>Population: {{Math.round(prop.population)}}</p>
        <p>Density: {{ isBigHexes? Math.round( prop.population / 0.7373) :
Math.round( prop.population / 0.105)}}</p>
      </template>
    </hexes-layer-group>

    <hexes-layer-group
      :name="layerNames[2]"
      :hexes="baseLayer && baseLayer.name === layerNames[2]? hexes : []"
      :colormap="colors[1]"
      color="green"
      :bins="hexBins"
      opacityBy="square"
      layerType="base">
      <template v-slot:popup="{prop}">
        <p>Areas count: {{prop.areas_count }}</p>
        <p>Square: {{ Math.round(prop.square) }}</p>
      </template>
    </hexes-layer-group>

    <hexes-layer-group
      :name="layerNames[3]"
      :hexes="baseLayer && baseLayer.name === layerNames[3]? hexes : []"
      :colormap="colors[1]"
      color="green"
      :bins="hexBins"
      :canMultiSelect="multiSelect"
      opacityBy="square_by_person"
      layerType="base">
      <template v-slot:popup="{prop}">
        <p>Areas count: {{ prop.areas_count }}</p>
        <p>Square: {{ Math.round(prop.square) }}</p>
        <p>Square by person: {{ Math.round(prop.square_by_person) }}</p>
      </template>
    </hexes-layer-group>
  </l-map>
</template>

<script>
import {LMap, LTileLayer, LControlLayers, LLayerGroup,LControlZoom, LControl}
from 'vue2-leaflet';
import L from "leaflet";
import icon from "../icon.png";
import "leaflet.markercluster/dist/leaflet.markercluster-src"
import HexesLayerGroup from './HexesLayerGroup.vue';
import Colormap from "colormap";
export default {
  components: {
    LMap,
    LTileLayer,
    LControlLayers,
    LControlZoom,
    LControl,
    LLayerGroup,
    HexesLayerGroup,
  },
  computed: {
    console: ()=> console,
    selectedHexes(){
      return this.$store.getters.selectedHexes;
    },
  },

```

```

facilities() {
  return this.$store.getters.newFacilities;
},
selectedFacility() {
  return this.$store.getters.selectedFacility;
},
facilityFilter() {
  return this.$store.getters.facilityFilter;
},
sportFilter() {
  return this.$store.getters.sportFilter;
},
facilityReport() {
  return this.$store.getters.facilityReport;
},
maxOpacityPerFacility() {
  if (this.$store.getters.facilities.length > 1000) return 0.07;
  if (this.$store.getters.facilities.length > 500) return 0.12;
  return 0.3
},
hexes() {
  return this.$store.getters.hexes;
},
layerNames() {
  return this.$store.getters.layerNames;
},
hexBins() {
  return this.$store.getters.hexBins;
},
},
watch: {
  selectedFacility(newVal) {
    if (!newVal) return;
    this.center = newVal.placement.coordinates;
  },
  facilityFilter(newVal) {
    this.markers.clearLayers()
    this.$store.commit("CLEAR_TILE_LIST")
    this.$store.commit('CLEAR_NEW_FACILITIES_BUFFER')
    if (!newVal) return
    this.$store.dispatch(
      "getFacilitiesByTiles",
      {
        tiles: this.bbox2tiles(this.boundsToBbox(this.bounds),
this.getZoomForTiles()),
        facilityFilter: newVal
      }
    )
  },
  sportFilter() {
    this.$store.commit("CLEAR_HEXES")
    this.$store.commit("CLEAR_LAST_DENSITY_TILES")
    this.$store.commit("CLEAR_SELECTED_HEXES")
    this.updateHexBins();
    this.update(this.$refs.map.mapObject.getBounds())
  },
  facilities(fac) {
    if (!fac.length) return;
    fac.forEach(el => {
      let marker = L.marker(L.latLng(el.placement.coordinates));
      marker.on('click', () => this.showInformation(el))
      marker.bindTooltip(el.name)
      this.markers.addLayer(marker)
    });
  }
}

```

```

        this.$refs.circles.mapObject.addLayer(this.markers)
        this.$store.commit('CLEAR_NEW_FACILITIES_BUFFER')
    },
    hexBins() {
        this.colors = [
            this.createColorMap([
                {index: 0, rgb: [255, 0, 0]},
                {index: 0.5, rgb: [255, 125, 125]},
                {index: 1, rgb: [255, 255, 255]}
            ]),
            this.createColorMap('greens')
        ]
    },
    selectedHexes(val) {
        if (!val.length) {
            this.showHexLegend = false
            return
        }
        this.showHexLegend = true
    },
    data() {
        return {
            markers: L.markerClusterGroup({
                chunkedLoading: true,
                disableClusteringAtZoom: 14,
                removeOutsideVisibleBounds: true
            }),
            url: 'http://tile2.maps.2gis.com/tiles?x={x}&y={y}&z={z}',
            attribution:
                '&copy; <a target="_blank" href="http://law.2gis.ru/api-rules/">2GIS</a> contributors',
            startZoom: 14,
            zoom: 14,
            center: [55.751244, 37.618423],
            isBigHexes: false,
            bounds: null,
            icon: L.icon({
                iconUrl: icon,
                popupAnchor: [1, -35],
                className: "selected",
            }),
            redSquareWeights: [
                60.0, 140.0, 256.0, 420.0, 686.5
            ],
            greenSquareWeights: [
                686.5, 1092.0, 1720.0, 2734.8, 4956.0, 5000000.0
            ],
            zoomBorder: 14,
            activeLayers: [],
            baseLayer: null,
            canSelect: true,
            multiSelect: false,
            showHexLegend: false,
            colors: [
                this.createColorMap([
                    {index: 0, rgb: [255, 0, 0]},
                    {index: 0.5, rgb: [255, 125, 125]},
                    {index: 1, rgb: [255, 255, 255]}
                ]),
                this.createColorMap('greens')
            ]
        };
    },
    methods: {
        createColorMap(colorMap) {
            return Colormap({
                colormap: colorMap,
                nshades: this.hexBins?.length,
                format: 'hex',
                alpha: 1
            }).reverse();
        },
        handleLayerControl() {

```

```

    const classes = this.$el.querySelector('.leaflet-control-
layers').classList;
    if (classes.contains('hide')) {
      classes.remove('hide')
    } else {
      classes.toggle('hide')
    }
  },
  showInformation(facility) {
    console.log(facility.id);
    this.$store.dispatch('getFacilityReport', facility)
  },
  baselayerchange: function(layer) {
    this.baseLayer = layer
    this.$store.commit("CLEAR_HEXES")
    this.$store.commit("CLEAR_LAST_DENSITY_TILES")
    this.$store.commit("CLEAR_SELECTED_HEXES")
    this.$store.commit("SET_BASE_LAYER", layer.name)
    this.updateHexBins();
    this.update(this.$refs.map.mapObject.getBounds())
  },
  overlayadd: function(overlay) {
    this.activeLayers.push(overlay)
    this.update(this.$refs.map.mapObject.getBounds())
  },
  overlayremove: function(overlay) {
    this.activeLayers.splice(this.activeLayers.indexOf(overlay), 1)
    this.update(this.$refs.map.mapObject.getBounds())
  },
  updateHexBins: function() {
    const payload = {
      sport_id: this.sportFilter?.sports,
      availability: this.sportFilter?.availability,
      area_type: this.sportFilter?.area_type,
      is_big_hexes: this.isBigHexes
    }
    this.$store.dispatch("getHexBins", payload)
  },
  getRadius: function (availability) {
    if (availability === 4) return 500;
    if (availability === 3) return 1000;
    if (availability === 2) return 3000;
    if (availability === 1) return 5000;
  },
  getSquareCircleOpacity: function (square) {
    let opacity = 0;
    if (this.isSquareCircleGreen(square)) {
      for (const weight of this.greenSquareWeights) {
        opacity += 0.2
        if (square <= weight) break;
      }
    } else {
      opacity = 1
      for (const weight of this.redSquareWeights) {
        opacity -= 0.18
        if (square < weight) break;
      }
    }
    return opacity * this.maxOpacityPerFacility
  },
  isSquareCircleGreen: function (square) {
    return square > this.greenSquareWeights[0]
  },
  update: function (bounds) {

```

```

    this.bounds = bounds;
    let tiles = this.bbox2tiles(this.boundsToBbox(bounds),
this.getZoomForTiles())
    if (this.facilityFilter) {
        this.$store.dispatch(
            "getFacilitiesByTiles",
            {
                tiles,
                facilityFilter: this.$store.getters.facilityFilter
            }
        )
    }
}
switch (this.baseLayer?.name) {
    case this.layerNames[1]:
        if (this.isBigHexes) {
            this.$store.dispatch("getDensityBigHexes", {tiles});
        } else {
            this.$store.dispatch("getDensitySmallHexes", {tiles})
        }
        break;
    case this.layerNames[2]:
        if (this.isBigHexes) {
            this.$store.dispatch("getSportBigHexes", {tiles});
        } else {
            this.$store.dispatch("getSportSmallHexes", {tiles})
        }
        break;
    case this.layerNames[3]:
        if (this.isBigHexes) {
            this.$store.dispatch("getUnitingBigHexes", {tiles});
        } else {
            this.$store.dispatch("getUnitingSmallHexes", {tiles})
        }
        break;
    default:
        break;
}
},
onZoom: function (zoom) {
    if (this.isBigHexes && zoom >= 14) {
        this.isBigHexes = false;
        this.$store.commit("CLEAR_SELECTED_HEXES")
        this.updateHexBins();
    } else if (!this.isBigHexes && zoom < 14) {
        this.isBigHexes = true;
        this.$store.commit("CLEAR_SELECTED_HEXES")
        this.updateHexBins();
    }
},
getOpacity: function (population) {
    let opacity = 0;
    for (const val of (this.isBigHexes ? this.bigHexBins :
this.smallHexBins)) {
        opacity += 0.1
        if (population < val) break;
    }
    return opacity * 0.7
},
boundsToBbox: function (bounds) {
    return [bounds._southWest.lat, bounds._southWest.lng,
bounds._northEast.lat, bounds._northEast.lng]
},
lon2tile: function (lon, zoom) {
    return (Math.floor((lon + 180) / 360 * Math.pow(2, zoom)));
}

```

```

    },
    lat2tile: function (lat, zoom) {
        return (Math.floor((1 - Math.log(Math.tan(lat * Math.PI / 180) + 1 /
Math.cos(lat * Math.PI / 180)) / Math.PI) / 2 * Math.pow(2, zoom)));
    },
    tile2long: function (x, z) {
        return (x / Math.pow(2, z) * 360 - 180);
    },
    tile2lat: function (y, z) {
        const n = Math.PI - 2 * Math.PI * y / Math.pow(2, z);
        return (180 / Math.PI * Math.atan(0.5 * (Math.exp(n) - Math.exp(-n))));
    },
    bbox2tiles: function (bbox, zoom) {
        const y1 = this.lat2tile(bbox[1], zoom);
        const x1 = this.lon2tile(bbox[0], zoom);
        const y2 = this.lat2tile(bbox[3], zoom);
        const x2 = this.lon2tile(bbox[2], zoom);
        let tiles = []
        for (let i = Math.min(y1, y2); i <= Math.max(y1, y2); i++) {
            for (let j = Math.min(x1, x2); j <= Math.max(x1, x2); j++) {
                tiles.push([j, i, zoom])
            }
        }
        return tiles
    },
    getZoomForTiles() {
        return 13
    },
    getFacilityReport(id) {
        this.$store.dispatch("getFacilityReport",{ id })
    },
    getAvailabilityNameById(id) {
        return this.$store.getters.availabilities.find( item => item.id === id)
    },
    getSportsNameById(ids) {
        let res = "";
        ids.forEach( id => {
            res += this.$store.getters.sportTypes.find((sport) => sport.id ===
id).name + " "
        })
        return res
    },
    getAreaTypeById(id) {
        return this.$store.getters.areaTypes.find( item => item.id === id)
    },
    },
    mounted() {
        this.$nextTick(() => {
            this.update(this.$refs.map.mapObject.getBounds());
            this.handleLayerControl()
        })
    }
}
</script>
<style >
@import "~leaflet.markercluster/dist/MarkerCluster.css";
@import "~leaflet.markercluster/dist/MarkerCluster.Default.css";
@import url("https://unpkg.com/leaflet@1.0.0/dist/leaflet.css");
.icon{
    min-width: 44px !important;
}
.selectBtn {
    border: 2px solid rgba(0,0,0,0.2);
    border-radius: 20px !important;

```

```

    margin-right: 10px !important;
  }
  .selected {
    filter: hue-rotate(265deg);
    margin-left: -12px;
    margin-top: -41px;
  }
  .leaflet-control-layers{
    /* display: none; */
    margin-right: 20px !important;
    border-radius: 20px !important;
  }
  .hide{
    display: none;
  }
  .leaflet-control-layers-overlays{
    padding: 5px;
  }
  .leaflet-control-zoom{
    margin-bottom: 20px !important;
    margin-right: 20px !important;
    border: 0px !important;
  }
  .leaflet-control-zoom-in{
    width: 40px !important;
    height: 40px !important;
    border-top-left-radius: 20px !important;
    border-top-right-radius: 20px !important;
  }
  .leaflet-control-zoom-out{
    width: 40px !important;
    height: 40px !important;
    border-bottom-left-radius: 20px !important;
    border-bottom-right-radius: 20px !important;
  }
  .legend {
    padding: 5px;
  }
  .legend i{
    width: 18px;
    height: 18px;
    float: left;
    margin-right: 8px;
    opacity: 0.7;
  }
  .hexLegend{
    padding: 5px;
  }
</style>

```

Листинг В.7 – Файл MainMenu.vue

```

<template>
  <div>
    <v-layout fill-height>
      <v-navigation-drawer permanent stateless app>
        <div v-if="currentMenu===0" >
          <v-list-item >
            <v-list-item-content>
              <v-list-item-title class="text-h6">
                Интерактивная карта
              </v-list-item-title>
            </v-list-item-content>
          </v-list-item>
        </div>
      </v-navigation-drawer>
    </v-layout>
  </div>
</template>

```

```

        <v-list-item-subtitle>
            спортивных объектов
        </v-list-item-subtitle>
    </v-list-item-content>
</v-list-item>
<v-divider></v-divider>
<v-list>
    <v-list-item-group >
        <v-list-item @click="currentMenu = 1">
            <v-list-item-title>Аналитика</v-list-item-title>
        </v-list-item>
        <v-list-item @click="currentMenu = 2">
            <v-list-item-title>Отображаемые объекты</v-list-item-title>
        </v-list-item>
    </v-list-item-group>
</v-list>
</div>

<v-list nav v-if="currentMenu===1">
    <v-list-item-group>
        <v-list-item @click="currentMenu = 0" class="primary--text"><v-
icon left class="primary--text">mdi-chevron-left</v-icon><span
class="primary--text">Назад</span></v-list-item>
        <v-divider/>
    </v-list-item-group>
    <div class="text-subtitle-1 mx-4 mt-4">Фильтр спортивных объектов
включаемых в аналитику</div>
    <v-list dense nav>
        <v-list-item>
            <v-list-item-content>
                <v-autocomplete
                    clearable
                    label="Вид спорта"
                    :items="sportTypes"
                    :value="sportFilter.sports"
                    item-text="name"
                    item-value="id"
                    @change="(val) => sportFilter.sports = val"
                ></v-autocomplete>
            </v-list-item-content>
        </v-list-item>
        <v-list-item>
            <v-list-item-content>
                <v-autocomplete
                    clearable
                    label="Тип спортивной зоны"
                    :items="areaTypes"
                    item-text="name"
                    item-value="id"
                    :value="sportFilter.area_type"
                    @change="(val) => sportFilter.area_type = val"
                ></v-autocomplete>
            </v-list-item-content>
        </v-list-item>
        <v-list-item>
            <v-list-item-content>
                <v-autocomplete
                    clearable
                    label="Доступность"
                    :items="availabilities"
                    :value="sportFilter.availability"
                    item-text="name"
                    item-value="id"

```

```

        @change="(val) => sportFilter.availability = val"
      ></v-autocomplete>
    </v-list-item-content>
  </v-list-item>
  <v-list-item>
    <v-list-item-content>
      <v-autocomplete
        clearable
        :items="departments"
        :value="sportFilter.department"
        v-on:change="(val) => sportFilter.department = val"
        item-text="name"
        item-value="id"
        label="Ведомство"
      ></v-autocomplete>
    </v-list-item-content>
  </v-list-item>
  <v-list-item>
    <v-list-item-content>
      <v-btn color="primary" outlined @click="() =>
setSportFilters(sportFilter)">Применить</v-btn>
    </v-list-item-content>
  </v-list-item>
  <v-list-item v-if="isSportFilterSet">
    <v-list-item-content>
      <v-btn color="error" outlined @click="() =>
{$store.commit('SET_SPORT_FILTER',null)}">Очистить</v-btn>
    </v-list-item-content>
  </v-list-item>
</v-list>
</v-list>

<v-list nav v-if="currentMenu===2">
  <v-list-item-group >
    <v-list-item @click="currentMenu = 0"><v-icon left
class="primary--text">mdi-chevron-left</v-icon><span class="primary--
text">Назад</span></v-list-item>
    <v-divider/>
    <div class="text-subtitle-1 mx-4 mt-4 pb-0">Фильтр отображаемых
спортивных объектов</div>
  </v-list-item-group>
  <v-list dense nav>
    <v-list-item>
      <v-list-item-content>
        <v-text-field
          label="Название спортобъекта"
          :value="filter.name"
          @input="(val) => filter.name = val"
        ></v-text-field>
      </v-list-item-content>
    </v-list-item>
    <v-list-item>
      <v-list-item-content>
        <v-text-field
          label="Название спортзоны"
          :value="filter.area_name"
          @input="(val) => filter.area_name = val"
        ></v-text-field>
      </v-list-item-content>
    </v-list-item>
    <v-list-item>
      <v-list-item-content>
        <v-autocomplete

```

```

        multiple
        clearable
        label="Вид спорта"
        :items="sportTypes"
        :value="filter.sports"
        item-text="name"
        item-value="id"
        @change="(val) => filter.sports = val"
      </v-autocomplete>
    </v-list-item-content>
  </v-list-item>
  <v-list-item>
    <v-list-item-content>
      <v-autocomplete
        clearable
        label="Тип спортзоны"
        :items="areaTypes"
        :value="filter.area_type"
        item-text="name"
        item-value="id"
        @change="(val) => filter.area_type = val"
      ></v-autocomplete>
    </v-list-item-content>
  </v-list-item>
  <v-list-item>
    <v-list-item-content>
      <v-autocomplete
        clearable
        label="Доступность"
        :items="availabilities"
        :value="filter.availability"
        item-text="name"
        item-value="id"
        @change="(val) => filter.availability = val"
      ></v-autocomplete>
    </v-list-item-content>
  </v-list-item>
  <v-list-item>
    <v-list-item-content>
      <v-autocomplete
        clearable
        :items="departments"
        v-on:change="(val) => filter.department = val"
        :value="filter.department"
        item-text="name"
        item-value="id"
        label="Ведомство"
      ></v-autocomplete>
    </v-list-item-content>
  </v-list-item>
  <v-list-item>
    <v-list-item-content>
      <v-btn color="primary" outlined @click="() =>
setFilters(filter)">Применить</v-btn>
    </v-list-item-content>
  </v-list-item>
  <v-list-item v-if="isFacilityFilterSet">
    <v-list-item-content>
      <v-btn color="error" outlined @click="() =>
{$store.commit('SET_FACILITY_FILTER',null)}">Очистить</v-btn>
    </v-list-item-content>
  </v-list-item>
</v-list>
</v-list>

```

```

    </v-spacer/>

    <v-list-item v-if="isSportFilterSet && isFacilityFilterSet &&
currentMenu === 0">
      <v-list-item-content>
        <v-btn color="error" outlined @click="() =>
{$store.commit('SET_SPORT_FILTER',null);
$store.commit('SET_FACILITY_FILTER',null)}">Очистить все фильтры</v-btn>
      </v-list-item-content>
    </v-list-item>
  </v-navigation-drawer>

  <v-navigation-drawer hide-overlay stateless v-model='infoDrawer' app>
    <v-card v-if="facilityReport" elevation="0">
      <v-card-actions >
        <v-list-item @click="closeInfo"><v-icon left class="primary--
text">mdi-chevron-left</v-icon> Назад</v-list-item>
      </v-card-actions>
      <v-divider/>
      <v-card-title>
        <div >
          {{facilityReport.name}}
        </div>
        <div class="text-caption">
          {{availabilities.find( el => facilityReport.availability ===
el.id).name}}
        </div>
      </v-card-title>
      <v-card-subtitle>
        <div class="text-subtitle-1">
          {{facilityReport.department.name}}
        </div>
      </v-card-subtitle>
      <v-divider/>
      <v-card-text>
        <v-expansion-panels multiple flat accordion>
          <v-expansion-panel
            v-for="area in facilityReport.areas"
            :key="area.id"
          >
            <v-expansion-panel-header>
              {{area.name}}
            </v-expansion-panel-header>
            <v-expansion-panel-content>
              <div>
                <strong>Площадь:</strong> {{area.square}} м<sup>2</sup>
              </div>
              <div >
                <strong>Спортивные зоны:</strong> {{areaTypes.find(type
=> type.id === area.type).name}}
              </div>
              <div>
                <strong>Виды спорта:</strong> {{sportTypes.filter( type
=> area.sports.find(el => el === type.id)).map(el=> el.name).join(', ')}}
              </div>
            </v-expansion-panel-content>
          </v-expansion-panel>
        </v-expansion-panels>
      </v-card-text>
    </v-card>
  </v-navigation-drawer>
</v-layout >

```

```

</div>
</template>

```

```

<script>
export default {
  name: "MainMenu",
  computed: {
    console: ()=>console,
    departments() {
      return this.$store.getters.departments;
    },
    areaTypes() {
      return this.$store.getters.areaTypes;
    },
    sportTypes() {
      return this.$store.getters.sportTypes;
    },
    availabilities() {
      return this.$store.getters.availabilities;
    },
    facilities() {
      return this.$store.getters.facilities;
    },
    facilityReport(){
      return this.$store.getters.facilityReport;
    },
    isSportFilterSet(){
      return !!this.$store.getters.sportFilter
    },
    isFacilityFilterSet(){
      return !!this.$store.getters.facilityFilter
    },
  },
  watch: {
    facilityReport(val){
      this.showInfo(val)
    },
    showFilter(val) {
      this.filterDrawer = val
    }
  },
  data () {
    return {
      right: null,
      currentMenu: 0,
      filter: {},
      sportFilter:{},
      filterDrawer:false,
      infoDrawer: false,
    }
  },
  methods: {
    setFilters (filter) {
      this.$store.commit("SET_FACILITY_FILTER", {...filter})
    },
    setSportFilters (filter) {
      this.$store.commit("SET_SPORT_FILTER", {...filter})
    },
    setFacility (facility) {
      this.$store.dispatch("getSelectedFacility", { facility })
    },
    showInfo(facility){
      console.log(facility)
    }
  }
}

```

```

        this.infoDrawer = true
      },
      closeInfo() {
        this.infoDrawer = false
      }
    }
  }
}
</script>

<style>
</style>

```

Листинг В.8 – Файл Report.vue

```

<template>
  <v-app>
    <v-sheet
      color="white"
      elevation="1"
      width="1000"
      class="mx-auto mt-2 my-auto pa-6"
      v-if="hexReport"
    >
      <l-map style="height: 330px" :zoom="14" :center="center">
        <l-tile-layer :url="url" :attribution="attribution"></l-tile-
layer>
        <l-polygon ref="polygon" :lat-lngs="coordinates"></l-polygon>
      </l-map>
      <v-container class="px-14">
        <div class="text-h4 text-center pb-4">Информация по выбранным
полигонам </div>
        <v-row class="pb-4">
          <v-col
            cols="6"
          >
            <v-card height="140">
              <v-card-title>
                Общая площадь полигонов
              </v-card-title>
              <v-card-text class="text-h3 primary--text">
                {{hexReport.total_square.toFixed(3)}}
км<sup>2</sup>
              </v-card-text>
            </v-card>
          </v-col>
          <v-col
            cols="6"
          >
            <v-card height="140">
              <v-card-title>
                Суммарное население полигонов
              </v-card-title>
              <v-card-text class="text-h3 primary--text">
                {{hexReport.total_population}} человек
              </v-card-text>
            </v-card>
          </v-col>
          <v-col
            cols="6"
          >
            <v-card height="140">
              <v-card-title>
                Плотность населения

```

```

        </v-card-title>
        <v-card-text class="text-h3 primary--text">
            {{ Math.round(hexReport.total_population /
hexReport.total_square) }} чел/км<sup>2</sup>
        </v-card-text>
    </v-card>
</v-col>
<v-col
    cols="6"
>
    <v-card height="140">
        <v-card-title>
            Суммарная площадь спортивных объектов
        </v-card-title>
        <v-card-text class="text-h3 primary--text">
            {{Math.round(hexReport.total_sport_square)}}
<sup>2</sup>
        </v-card-text>
    </v-card>
</v-col>
<v-col
    cols="6"
>
    <v-card height="140">
        <v-card-title>
            Количество доступных видов спорта
        </v-card-title>
        <v-card-text class="text-h3 primary--text">
            {{hexReport.sports.length}} из 160
        </v-card-text>
    </v-card>
</v-col>
<v-col
    cols="6"
>
    <v-card height="140">
        <v-card-title>
            Доступность типов спортивных зон
        </v-card-title>
        <v-card-text class="text-h3 primary--text">
            {{Object.keys(this.hexReport.area_types_counter).length}} из 108
        </v-card-text>
    </v-card>
</v-col>
</v-row>
<div class="text-h5 text-center py-4">Количество спортивных
зон</div>
    <HorizontalBarChart
:chartdata="sportAreasChartDataset"></HorizontalBarChart>
    <div class="text-h5 text-center py-4">Площадь спортивных зон
на человека</div>
    <HorizontalBarChart
:chartdata="sportAreasSquareChartDataset"></HorizontalBarChart>
    <div class="text-h5 text-center py-4">Детальная
информация</div>
    <v-expansion-panels>
        <v-expansion-panel>
            <v-expansion-panel-header>
                Список доступных видов спорта
            </v-expansion-panel-header>
            <v-expansion-panel-content v-if="sportTypes.length">
                <ul class="text-body-2 ml-4" v-for="sport in
hexReport.sports" :key="sport">

```

```

        <li>{{sportTypes.find( ar => ar.id ===
+sport).name}}</li>
    </ul>
</v-expansion-panel-content>
</v-expansion-panel>
<v-expansion-panel>
    <v-expansion-panel-header>
        Суммарная площадь по типу спортивной зоны
    </v-expansion-panel-header>
    <v-expansion-panel-content v-if="areaTypes.length">
        <ul class="text-body-2 ml-4" v-for="area in
Object.entries(hexReport.area_types_coverage)" :key="area[0]">
            <li>{{areaTypes.find( ar => ar.id ===
+area[0]).name}}: {{Math.round(area[1])}} м<sup>2</sup></li>
        </ul>
    </v-expansion-panel-content>
</v-expansion-panel>
<v-expansion-panel>
    <v-expansion-panel-header>
        Количество спортивных зон
    </v-expansion-panel-header>
    <v-expansion-panel-content v-if="areaTypes.length">
        <ul class="text-body-2 ml-4" v-for="area in
Object.entries(hexReport.area_types_counter)" :key="Math.random(area[0])">
            <li>{{areaTypes.find( ar => ar.id ===
+area[0]).name}}: {{Math.round(area[1])}} шт.</li>
        </ul>
    </v-expansion-panel-content>
</v-expansion-panel>
</v-expansion-panels>
</v-container>
</v-sheet>
</v-app>

</template>
<script>
import HorizontalBarChart from "../charts/HorizontalBarChart";
import {LMap, LTileLayer, LPolygon} from 'vue2-leaflet';
export default {
    components: {
        LMap,
        LTileLayer,
        HorizontalBarChart,
        LPolygon
    },
    data () {
        return {
            url: 'http://tile2.maps.2gis.com/tiles?x={x}&y={y}&z={z}',
            attribution:
                '&copy; <a target="_blank" href="http://law.2gis.ru/api-
rules/">2GIS</a> contributors',
            zoom: 15,
        }
    },
    computed: {
        hexReport() {
            return this.$store.getters.hexReport
        },
        areaTypes() {
            if (this.$store.getters.areaTypes.length <= 1) return []
            return this.$store.getters.areaTypes;
        },
        sportTypes() {
            if (this.$store.getters.sportTypes.length <= 1) return []

```

```

        return this.$store.getters.sportTypes;
    },
    sportAreasChartDataset() {
        if (!this.hexReport || !this.areaTypes || !this.areaTypes.length)
    ) return;
        return {
            labels:
Object.keys(this.hexReport.area_types_counter).map(type_id =>
this.areaTypes.find( ar => ar.id === +type_id).name),
            datasets: [
                {
                    label: 'Количество спортивных зон',
                    backgroundColor: '#ec0e43',
                    data:
Object.values(this.hexReport.area_types_counter)
                },
            ]
        },
    },
    sportAreasSquareChartDataset() {
        if (!this.hexReport || !this.areaTypes || !this.areaTypes.length)
return;
        return {
            labels:
Object.keys(this.hexReport.area_types_coverage).map(type_id =>
this.areaTypes.find( ar => ar.id === +type_id).name),
            datasets: [
                {
                    label: 'Площадь спортивных зон на человека, м^2',
                    backgroundColor: '#ec0e43',
                    data:
Object.values(this.hexReport.area_types_coverage).map(square => square /
this.hexReport.total_population)
                },
            ]
        },
    },
    center() {
        if (!this.hexReport) return ;
        return this.getCentroid(this.coordinates)
    },
    coordinates() {
        if (!this.hexReport) return ;
        return this.hexReport.polygon.coordinates[0][0].map(i => [i[1],
i[0]])
    },
    },
    methods: {
        getCentroid (arr) {
            return arr.reduce(function (x,y) {
                return [x[0] + y[0]/arr.length, x[1] + y[1]/arr.length]
            }, [0,0])
        },
    },
    mounted() {
        this.$store.dispatch("getAreaTypes");
        this.$store.dispatch("getSportTypes");
        this.$store.dispatch("getHexReport", {isBig:
this.$route.params.isBig, ids: {ids: JSON.parse(this.$route.params.ids)}})
    },
}
</script>
<style>
</style>

```

Приложение Г
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ
(рекомендуемое)

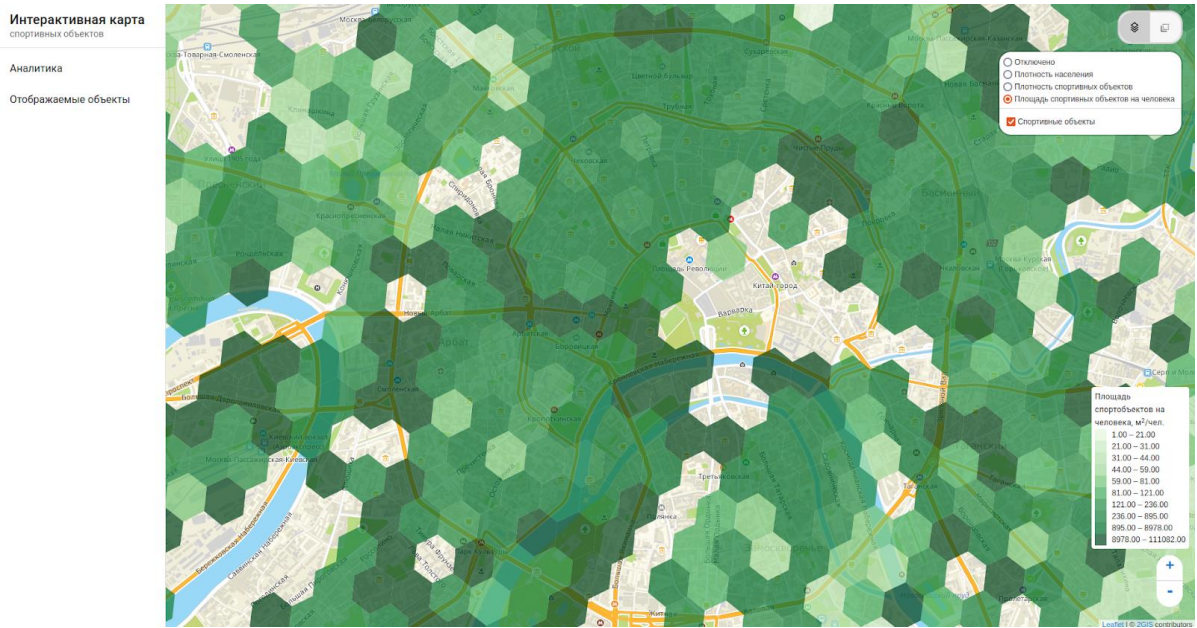


Рисунок Г.1 – Результаты тестирования варианта использования «Просмотреть объединенный слой»

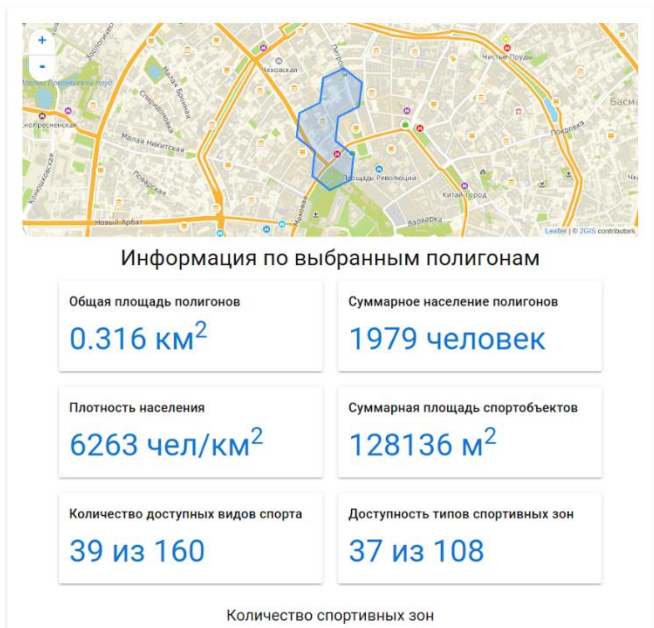


Рисунок Г.2 – Результаты тестирования варианта использования «Сформировать отчет по выбранным гексагонам»

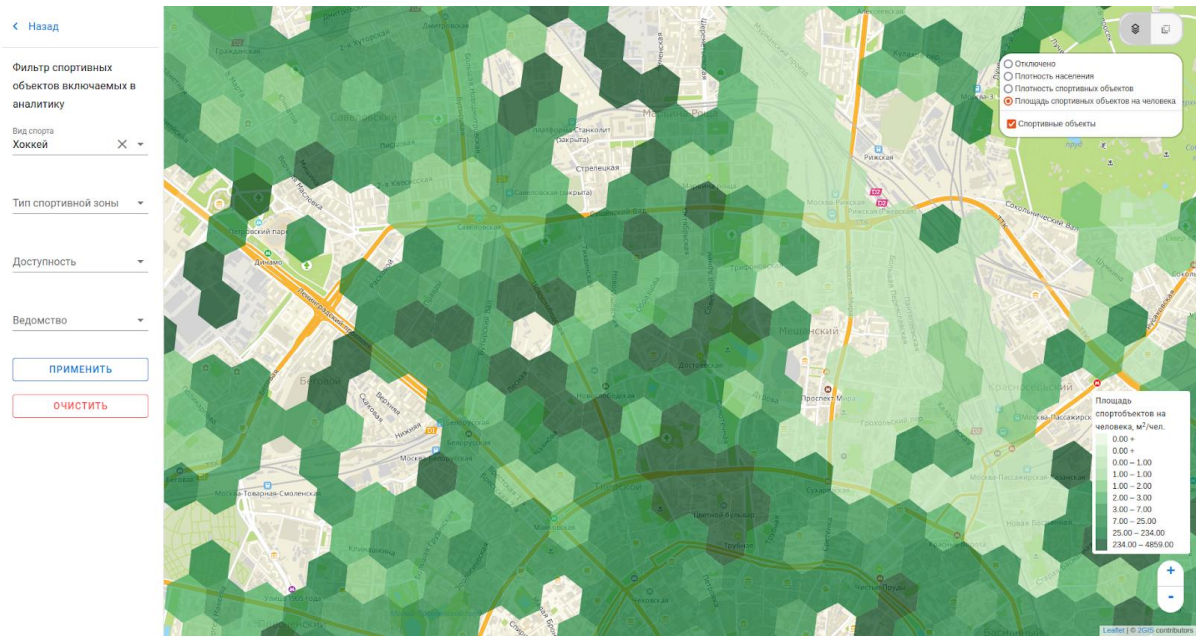


Рисунок Г.3 – Результаты тестирования варианта использования «Отфильтровать спортобъекты участвующие в расчете тепловых карт»