Санкт-Петербургский государственный университет

РОГОВА Елена Александровна

Выпускная квалификационная работа

Разработка автоматизированной системы анализа и учета реестра объектов физической культуры и спорта

Уровень образования: бакалавриат

Направление 09.03.03 *«Прикладная информатика»*

Основная образовательная программа СВ.5078

«Прикладная информатика в области искусств и гуманитарных наук»

Профиль «Прикладная информатика в области искусств и гуманитарных наук»

Научный руководитель: заведующий кафедры информационных систем в искусстве и гуманитарных науках, доктор физ.-мат. наук, профессор Борисов Николай Валентинович

Консультант: старший преподаватель кафедры информационных систем в искусстве и гуманитарных науках Мбого Ирина Анатольевна

Рецензент: генеральный директор, Общество с ограниченной ответственностью «Русская коллекция СПб», Кудеров Дмитрий Евгеньевич

Санкт-Петербург

2022

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ИСКУССТВ

Кафедра информационных систем в искусстве и гуманитарных науках

Основная образовательная программа

«Прикладная информатика в области искусств и гуманитарных наук»

УТВЕРЖДАЮ

Заведующий
Кафедрой

ЗАДАНИЕ

По подготовке выпускной квалификационной работы студентки Роговой Елены Александровны

1. Тема работы: Разработка автоматизированной системы анализа и учета реестра объектов физической культуры и спорта.

2. Срок сдачи студентом законченной работы: июнь 2022 г.

3. Срок сдачи текста выпускной квалификационной работы для выкладывания в Blackboard: 15 мая 2022 г.

4. Исходные данные к работе: техническое задание Администрации Красногвардейского района, информация в Интернете

5. План-график выполнения дипломной работы

Номера и содержание этапов работы	**Плановая дата сдачи**
Формулировка и утверждение ТЗ совместно с Администрацией Красногвардейского района, поиск и отбор аналогов и референсов.	Ноябрь 2021
Планирование архитектуры веб-приложения, выбор и изучение технологий для реализации	Декабрь 2021

Реализация веб-приложения	Апрель 2022
Написание теоретической части диплома	15 мая 2022
Защита теоретической и практической частей диплома	Июнь 2022

Руководители от Кафедры: Заведующий кафедры доктор физ.-мат. наук, профессор Борисов Николай Валентинович

Консультанты по работе: старший преподаватель Мбого Ирина Анатольевна

Задание принял к исполнению

Подпись студента 

АННОТАЦИЯ

выпускной квалификационной работы

Роговой Елены Александровны

Разработка автоматизированной системы анализа и учета реестра объектов физической культуры и спорта

Целью данной работы является создание веб-приложения – системы учета реестра объектов физической культуры и спорта «SportsMap» для администрации Красногвардейского района города Санкт-Петербург.

ВКР включает в себя 7 глав. 1-2 главы посвящены анализу существующих решений для похожих систем, сбору требований заказчика, а также изучению предметной области в целом. На основе этого было составлено техническое задание, что описано и проанализировано в 3-4 главах. Далее в главах 5-7 описано создание дизайна и прототипа продукта (UI/UX), непосредственно веб-разработка. Бэкенд-разработка велась на платформе .NET с использованием фреймворка .NET Core на языке C#. Фронтенд-разработка велась, в основе своей, с помощью фреймворка React для JavaScript.

Прикладным результатом ВКР является веб-приложение с внедренной системой репрезентации геоданных в виде карты спортивных объектов. Имеется возможность поиска и фильтрации объектов в списке. Кроме того, доступна авторизация для администратора, способного совершать добавление, удаление, скрытие, изменение объектов и выгрузку/загрузку данных в таблице формата .xlsx.

В процессе работы использованы следующие программы: Visual Studio, Visual Studio Code, Figma, Trello, Mapbox Studio, Postman, Adobe Illustrator, GitHub Client.

Объем работы: 92 страницы текста, 15 рисунков, 19 источников литературы и 2 приложения.

Ключевые слова: веб-разработка, React, .NET Core, анализ требований, проектирование архитектуры, UI/UX дизайн, геоинформационная система, fullstack-разработка.

Abstract

Final project

Elena Rogova

Development of an automated system for analysis and accounting of the register of physical culture and sports facilities

The purpose of this work is to create a web application - a system of accounting for the register of physical culture and sports facilities "SportsMap" for the administration of the Krasnogvardeysky district of St. Petersburg.

The paper includes 7 chapters. 1-2 chapters are devoted to the analysis of existing solutions for similar systems, collecting customer requirements, as well as studying the subject area as a whole. Based on this, the terms of reference were compiled, which is described and analyzed in 3-4 chapters. Further on, chapters 5-7 describe the creation of a design and prototype of a product (UI/UX), directly web development. Backend development was conducted on the platform.NET

using a framework .NET Core in C#. Frontend development was carried out, basically, using the React framework for JavaScript.

The applied result of the WRC is a web application with an embedded geodata representation system in the form of a map of sports facilities. It is possible to search and filter objects in the list. In addition, authorization is available for an administrator who is able to add, delete, hide, modify objects and upload/download data in a format table.xlsx.

The following programs are used in the process of work: Visual Studio, Visual Studio Code, Figma, Trello, Mapbox Studio, Postman, Adobe Illustrator, GitHub Client.

Volume of work: 92 pages, 15 drawings, 19 literature sources and 2 appendices.

Keywords: web development, React, .NET Core, requirements analysis, architecture design, UI/UX design, geoinformation system, fullstack development.

Используемые термины, определения и сокращения

В настоящем работе применяются следующие термины с соответствующими определениями и сокращения:

Таблица 1. Определения, обозначения и сокращения

Термин/сокращение	Определение / пояснение
Система или СУОС	Система анализа и учета реестра объектов физической культуры и спорта.
ОС	Операционная система
ИС	Информационная система
СО	Спортивный объект
БД	База данных
ВИ	Вариант использования системы
Эндпоинт	Обращение к маршруту HTTP методом.
Контент-администратор	Пользователь, имеющий право изменять карточки СО.
Стейкхолдер	«Физическое лицо или организация, имеющая права, долю, требования или интересы относительно системы или её свойств, удовлетворяющих их потребностям и ожиданиям»[1].
ТЗ	Техническое задание на разработку системы
Фронтенд-разработка	Разработка пользовательского интерфейса на клиентской стороне приложения. Это то, что видит пользователь, открывая веб-страницу, и с чем он взаимодействует.

Термин/сокращение	Определение / пояснение
Бэкенд-разработка	Разработка программно-аппаратной части сервиса, отвечающей за функционирование его внутренней части.
Айдентика	Совокупность визуальных объектов, представляющих стиль и имидж продукта/компании
Фреймворк	Программная абстракция, в которой программное обеспечение, обеспечивающее общую функциональность, может быть изменено дополнительным пользовательским кодом, таким образом предоставляя прикладное программное обеспечение. Программные фреймворки могут включать программы поддержки, компиляторы, библиотеки кода, наборы инструментов и интерфейсы прикладного программирования (API), которые объединяют все различные компоненты для разработки проекта или системы.

Оглавление

Введение

В современном мире огромное значение имеет цифровизация и автоматизация всевозможных процессов, повышение эффективности их выполнения. Это касается и работы административного ресурса города Санкт-Петербурга, а в частности сектора спорта Красногвардейского района. Они, как никто, заинтересованы в популяризации спорта, в инструментах, помогающих населению города вести активный и здоровый образ жизни. Поэтому возникла необходимость создания системы учета реестра объектов физической культуры и спорта для администрации Красногвардейского района г. Санкт-Петербурга, с перспективой расширения на весь город.

Данная работа посвящена основным этапам разработки веб-приложения «SportsMap», которое в user-friendly формате покажет на карте местоположение спортивных объектов в районе и поможет пользователю найти подходящий именно ему. Полезная функциональность предусмотрена и для работников сектора спорта – возможность администрировать базу данных спортивных объектов, а также выгружать отчеты со всеми существующими данными в удобном формате Excel-таблицы.

Процесс разработки включает в себя и проведение работы с заказчиком продукта, а также составление ТЗ на разработку приложения, выбор основных программных инструментов.

Из всего выше сказанного, целью работы является непосредственно разработка веб-приложения системы учета и анализа объектов реестра физической культуры и спорта.

В качестве задач для достижения этой цели стоит выделить:

- Изучить предметную область

- Собрать требований заказчика и оформить в виде ТЗ
- Спроектировать интерфейс приложения и его архитектуру
- Создать прототип и дизайн-проект интерфейса
- Осуществить бэкенд- и фронтенд-разработку

Глава 1. Изучение предметной области

1.1 Назначение требуемой информационной системы

Цель разработки и внедрения системы – это повышение эффективности Сектора спорта Администрации Красногвардейского района за счет сокращения времени поиска интересующего СО и повышения прозрачности процесса.

Система должна решать следующие задачи (См. подробнее гл. 2, Приложение №1):

- Обеспечить регистрацию и хранение всей доступной информации о СО.
- Позволить каждому жителю Красногвардейского района ознакомиться с реестром СО и найти подходящий, основываясь на различных критериях.
- Быть наиболее доступной для инвалидов, предоставлять информацию о доступных для них СО
- Позволить проводить анализ ситуации с СО в Красногвардейском районе с помощью карты (плотность СО, их доступность и проч.)
- Улучшить доступ населения к информации о СО, тем самым повысив мотивацию людей вести здоровый образ жизни, что является одной из основополагающих целей Сектора спорта Администрации Красногвардейского района.

1.2 Геоинформационные системы: описание, основные понятия, развитие

Развитие информационных и компьютерных технологий предоставило возможность создавать системы для хранения и распространения больших объемов пространственной информации.

«Веб-картография — быстро развивающаяся область компьютерных технологий, не только обеспечивающая доступ пользователю (клиенту) к пространственным данным, но и предусматривающая возможность составления и редактирования карт с помощью инструментальных средств, предусматривающая обращение к удаленным базам данных, целенаправленный подбор источников, совмещение и комбинирование тематических слоев, проведение генерализации, классификации, выбор способов изображения и графических стилей.» [6, 8] Все это происходит благодаря геоинформационным системам, разработанным на основе специальных фреймворков, имеющих в своем распоряжении доступ к точным картографическим данным.

Будущая СУОС относится именно к таким геоинформационным систем, правда, на данном этапе разработки с немногими перечисленными выше функциями, хотя и самыми основными. Доступ к пространственным данным, редактирование карт, доступ к базам данных – вот, что планируется внедрить в СУОС. Однако, стоит отметить, что при последующих этапах разработки, развивающих систему, все остальные функции так же предполагается включить в систему.

1.3 Спортивные объекты: описание, классификация, место в системе

Развитие физической культуры при любых условиях невозможно без строительства материальной базы в виде спортивных сооружений, специально оборудованных и правильно эксплуатируемых. А доступ к данным об этих СО на сегодняшний день так же важен, ведь люди практически всю информацию получают с помощью веб-источников.

Рассмотрим, что же такое СО и как их классифицируют. Это необходимо для понимания предметной области и правильного моделирования системы.

«Спортивное сооружение — это специально построенное или приспособленное и соответственно оборудованное сооружение, предназначенное для проведения соревнований по различным видам спорта, учебно-тренировочных занятий физкультурой и спортом, а также используемое для активного отдыха и общефизической подготовки.» [7]

Все спортивные сооружения могут быть общего и ограниченного пользования. Последние входят в состав детских и лечебных учреждений, учреждений рекреации и т.д.

В зависимости от возраста и физических возможностей различных групп населения СО можно распределить следующим образом:

- площадки для физических занятий и игровые площадки для детей дошкольного возраста;
- оздоровительные и спортивные сооружения для школьников, юношества и молодежи;
- оздоровительные и спортивные сооружения с несложным оборудованием для «групп здоровья», групп ОФП и лечебной физкультуры.

По местам расположения спортивные сооружения делятся на:

- сооружения с простым оборудованием по месту жительства;
- площадки для игр и физических занятий в детских дошкольных учреждениях;
- школьные;
- сооружения с простым оборудованием в зонах рекреации.

Спортивные сооружения включают:

- основные сооружения: спортивное ядро, ванна бассейна для плавания, поле для хоккея, площадка для волейбола и т.д.;

- вспомогательные сооружения: раздевальные, душевые, массажные, судейские, другие административные, хозяйственные и технические помещения;
- комплекс сооружений и помещений для зрителей: трибуны, фойе, гардеробы, буфеты, санузлы и т.д.

В зависимости от функций, которые возложены на основное спортивное сооружение, оно может быть специализированным и универсальным. Кроме того, основные спортивные сооружения подразделяются на открытые, крытые, с трансформирующимся покрытием (соревнования и тренировочные занятия при хорошей погоде проводятся на открытом воздухе, при плохой и зимой — под крышей).

Физкультурно-спортивные сооружения города объединяются в многофункциональные (с возможностью проведения занятий и соревнований по нескольким различным видам спорта) или специализированные (для одного или нескольких родственных видов спорта) комплексы.

Сеть спортивных сооружений для каждого города всегда учитывается в процессе архитектурно-планировочного проектирования, которое служат основой для строительства и реконструкции городов. Предусматривается создание целостной системы спортивных сооружений.

При организации сети спортивных сооружений учитывают характеристики конкретного города. К ним относятся такие факторы, как число жителей с ростом на перспективу, демографический состав населения, наличие существующих спортивных сооружений и их транспортная доступность, спортивный интерес жителей, природно-климатические условия, традиции и др.

1.4 Результаты

Благодаря изученной и проанализированной информации можно планировать, с какими объектами придется работать, какие данные собирать и предоставлять и даже с какими административными структурами также придется взаимодействовать (комитет по благоустройству, комитет по архитектуре и проч.). И после проведенного анализа можно начинать планировать саму архитектуру приложения, которое все проанализированное в себе предоставит.

Немаловажной информацией является и описание понятия геоинформационных систем. Именно глубокое понимание сути ГИС позволит эффективно создать приложение.

Глава 2. Сбор и анализ требований

2.1 Теоретический обзор

«Процесс анализа требований к информационной системе включает следующие фазы:

- Разработка требований
- Выявление требований
- Анализ требований
- Спецификация требований
- Проверка требований
- Управление требованиями» [2]

Основной частью разработки требований, а, если быть точнее, их выявления является идентификация и опрос стейкхолдеров. На этом этапе выясняются основные функциональные и нефункциональные требования, видение продукта со стороны заказчика. Без этого этапа невозможна дальнейшая разработка в верном ключе.

Грамотное проведение такого интервью – фундамент правильно составленного технического задания. Именно ТЗ является юридическим документом, основным в разработке информационной системы, закрепляющим требования к продукту и подробности дальнейшей реализации со стороны программиста.

«ТЗ на создание (развитие или модернизацию - далее создания) системы является документом, определяющим требования и порядок создания информационной системы, в соответствии с которым проводится разработка АС и ее приемка при вводе в действие.» [5]

2.2 Первоначальная информация

Проект был предложен Администрацией Красногвардейского района города Санкт-Петербурга в качестве одного из вариантов выполнения

дипломной работы для участия в конкурсе на соискание премии правительства Санкт-Петербурга за выполнение дипломных проектов по заданию органов исполнительной власти Санкт-Петербурга на 2021/2022 учебный год.

Поэтому для проекта СУОС заказчиком является Администрация Красногвардейского района. В дальнейшем под заказчиком будет подразумеваться представитель Администрации.

На состоявшемся 11.11.2021 заседании конкурсной комиссии было принято решение об определении студентов, которые будут осуществлять подготовку дипломных проектов в рамках конкурса [4], где произошло назначение исполнителя проекта «Разработка автоматизированной системы анализа и учета реестра объектов физической культуры и спорта». Поэтому в дальнейшем под исполнителем будет подразумеваться персона автора курсовой.

Изначально со стороны Администрации при описании проекта было сформулировано следующее: «Программа с картой, где можно узнать информацию и статистическую информацию объектов физической культуры и спорта». Разумеется, такой формулировки недостаточно для успешной и грамотной разработки системы, поэтому ее необходимо расширить с помощью взаимодействия со стейкхолдерами.

2.3 Работа со стейкхолдерами

Среди особенностей данного этапа процесса разработки для проекта «Разработка автоматизированной системы анализа и учета реестра объектов физической культуры и спорта» и самого проекта в принципе стоит выделить следующие:

- Заказчиком является Администрация Красногвардейского района города Санкт-Петербурга. Есть некоторые особенности

коммуникации (это влияет и на выдвигаемые к проекту требования) с государственными структурами по сравнению с бизнес-структурами: проекты государственных структур зачастую не несут в себе цели материальной выгоды, социальная значимость выходит на первый план; общение с органами власти достаточно бюрократизировано; в государственных структурах нет тех поставленных на поток бизнес-процессов, как в классической бизнес-структуре.

- Имеется большое количество стейкхолдеров именно из государственных структур, владеющих совершенно разной информацией (в силу большого количества подконтрольных объектов, а значит и ответственных за них) и видящих проект с разной степенью точности.

С учетом вышеупомянутого, были проведены дистанционные интервью с тремя представителями администрации: Шмелевой Дарьей Николаевной (начальник Сектора Физической культуры и спорта Администрации Красногвардейского района), Кашиной Екатериной Станиславовной (зам. начальника Сектора Физической культуры и спорта Администрации Красногвардейского района) и Соминским Алексеем Игоревичем (директор Центра физкультуры и здоровья, один из главных идейных вдохновителей проекта и заинтересованных в его реализации лиц).

С помощью интервью удалось выяснить все, что было необходимо для анализа требований и дальнейшего составления ТЗ. Интервьюируемые имели довольно конкретное видение результата разработки, поэтому сложностей это не составило.

Из спорных моментов можно выделить лишь то, что на тот момент у заказчика не имелось общей структуры и базы данных с информацией по

всем СО. Это затрудняет составление некоторых требований к системе, так как неясно, какую информацию точно нужно будет сохранять в системе, но не является критичным.

Стейкхолдеры весьма адекватно оценивали результаты данного этапа разработки, что вселяет уверенность в том, что в дальнейшем конфликтых ситуаций не возникнет. А во избежание таковых было необходимо в том числе и составление документов, фиксирующих всю предстоящую работу над приложением.

2.4 Анализ и фиксация полученных данных

На основе проведенных интервью был составлен документ «Общие требования к информационной системе и обязательства исполнителя» (Приложение №1). Он является результатом анализа первичных требований.

Формат документа был выбран произвольный, но основанный на шаблонах и стандартах ТЗ. С помощью этого документа необходимо было показать заказчику серьезность намерений, а также представить то, как были переработаны его требования.

Документ описывает главные детали и требования к предстоящей разработке:

- Утверждение исполнителя и заказчика проекта
- Что из себя представляет проект и какие основные части необходимо реализовать для его создания
- Что будет реализовано на данном этапе разработки
- Какие функции будет предоставлять веб-приложение
- Обязанности исполнителя

Данный документ является, по своей сути, упрощенным вариантом ТЗ, с помощью которого можно согласовать главные детали разработки и в дальнейшем приступать к расширенному ТЗ.

Заказчик ознакомился с требованиями и был ими удовлетворен. Бюрократические процедуры, например, всевозможные подписи, было решено опустить.

Глава 3. Составление ТЗ

3.1 Принципы и стандарты создания ТЗ

ТЗ устанавливает общее устройство системы, объем работ и порядок разработки и приемки. Этот документ в том числе помогает заказчику понять всю важность и сложность задач.

Для составления ТЗ создано множество стандартов, как российских, так и международных:

• ГОСТ 34

• ISO/IEC/ IEEE 29148-2011

• RUP

• ГОСТ 19

• IEEE STD 830-1998

• SWEBOK, BABOK

Часто при разработке ТЗ для государственных проектов заказчики требуют соблюдения стандарта ГОСТ 34 [5], в данном проекте таких ограничений не было. Несмотря на это, данный стандарт действительно является наиболее подходящим и, возможно, самым понятным и доступным для заказчика.

«ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы рекомендует структуру ТЗ на создание именно системы, в которую входят ПО, аппаратное обеспечение, люди, которые работают с ПО, и автоматизируемые процессы.» [3]

Согласно ГОСТ 34 техническое задание должно включать следующие разделы:

1. Общие сведения

2. Назначение и цели создания (развития) системы

3. Характеристика объектов автоматизации

4. Требования к системе

5. Состав и содержание работ по созданию системы

6. Порядок контроля и приемки системы

7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

8. Требования к документированию

9. Источники разработки

Любой процессный стандарт дает только общие указания, но всегда нужен грамотный подход применительно к конкретному проекту. Пункт 2.2 ГОСТ 34.602-89 гласит: «В зависимости от вида, назначения, специфических особенностей объекта автоматизации и условий функционирования системы допускается оформлять разделы ТЗ в виде приложений, вводить дополнительные, исключать или объединять подразделы ТЗ». Также в п. 1.2. РД 34.698-90 указано: «Содержание документов является общим для всех видов АС и, при необходимости, может дополняться разработчиком документов в зависимости от особенностей создаваемой АС. Допускается включать в документы дополнительные разделы и сведения, объединять и исключать разделы».

Важно помнить, что ТЗ устанавливает не только требования к самой системе, но и регламентирует процесс ее создания, то есть в документе приводятся требования ко всем организационным мерам, выполнение которых необходимо для достижения результата.

3.2 Составление и согласование ТЗ

Необходимо было переработать шаблон ТЗ под текущий проект. С итоговыми пунктами содержания и самим ТЗ можно ознакомиться в Приложении №2.

Основными смысловыми частями можно выделить следующие:

1. *Общие положения по проекту, описание*

В этом блоке перечислены формальные подробности: утвержденные заказчик и исполнитель, их контакты, назначение документа, определения и сокращения, плановые сроки разработки. В целом, довольно стандартные детали для ТЗ.

Стоит выделить пункт с определениями и сокращениями. Он необходим в каждом ТЗ, ведь ТЗ составляют специалисты в том числе и для неспециалистов, коими часто являются заказчики продуктов. Чрезвычайно важно, чтобы каждый блок ТЗ был предельно понятен заказчику, а для этого не обойтись без предварительного объяснения базовых понятий.

2. *Требования к системе*

Этот пункт является самым важным во всем ТЗ как для разработчика, так и для заказчика. Именно здесь описано то, каким будет продукт в итоге, какими характеристика и функционалом он будет обладать.

Все требования разделены на функциональные и нефункциональные. Функциональные требования включают в себя непосредственное описание функционала программы. Для этого приведены диаграммы вариантов использования, макеты интерфейса, дизайн-референсы, карта сайта, описание бизнес-логики, диаграмма классов и описание архитектуры. Данные требования и их реализация были разработаны в рамках производственной практики и обязательны к выполнению со стороны разработчика. Подробнее об этом см. гл. 4.

Нефункциональные требования включают в себя общие требования к реализации продукта: совместимость, безопасность, надежность, эргономичность, производительность, лингвистическое обеспечение и прочее. Безусловно, также важнейшая составляющая ТЗ, без которой невозможно будет избежать конфликтов по поводу, например, того, что на данном этапе мобильная версия реализована не будет, о чем заказчика нигде, кроме этого пункта, официально оповестить нельзя.

3. Схема работ по созданию системы

Пункт подразумевает общие требования к согласованию и отчету перед заказчиком со стороны исполнителя. Также предлагаются перспективы дальнейшего развития проекта на следующих после текущего этапах.

Именно эти пункты являются необходимыми и достаточными в данном ТЗ. Многие важные составляющие, присутствующие в ГОСТ, исключены в силу того, что перенесены в разряд составляемых в процессе непосредственной разработки. Например, руководство пользователя до окончания разработки системы составлять не имеет смысла, да и практически невозможно, так как далеко не всегда предположительный образ и функционал системы абсолютно идентичен реализованному.

3.3. Анализ составленного ТЗ и перспективы его расширения

На данном этапе планирования и проектирования системы невозможно описать все процессы и всю структуру проекта, так как есть детали, которые напрямую определяются в процессе разработки. А если определить их в ТЗ, то придется оспаривать с заказчиком каждое изменение, составлять новую документацию.

Итак, как же можно расширить техническое задание и описание проекта на первых этапах разработки?

1. *Дизайн-проект*

На данный момент ТЗ содержит только дизайн-референсы и макеты, но для полноценного дизайна этого мало. Обычно дизайн-проект является отдельным документом, и согласовывать его нужно отдельно. Дизайн – это самая важная составляющая для Заказчика, ведь именно ее пользователи будут видеть и ассоциировать с продуктом.

2. *Таблица сроков и сдачи проекта поэтапно*

Этот пункт позволит держать под контролем этапы разработки и их сроки, так как этот документ будут подписывать обе стороны при сдаче части проекта. Также это позволит как разработчику, так и заказчику ориентироваться на конкретные даты, знать, к чему готовиться и чего ожидать. Это всегда положительно влияет на результативность разработки.

3. *Полные стандарты документирования*

Здесь подразумеваются стандарты документирования непосредственно кодовой базы, работы API, доступа к БД и прочее. То есть все, что непосредственно связано с работой продукта на программном уровне для упрощения дальнейших этапов разработки и поддержки кода.

4. *Полная схема БД после согласования полей с Заказчиком*

Диаграммы, описывающие БД, безусловно важны, так как в них содержится самое важное – что мы храним в нашей системе и как с этим работаем? Какие взаимодействия происходят между информационными сущностями? И как мы с ними можем работать?

5. *Используемые для реализации проекта технологии*

Пункт, необходимый для согласования с заказчиком по той причине, что не все технологии с открытым исходным кодом могут обеспечить успешную разработку и существование проекта. Зачастую в процессе приходится внедрять платные продукты, оформлять лицензии. Так как все

это происходит на средства заказчика, он обязательно должен быть осведомлен и согласен на подобное использование технологий.

3.4. Результаты

В результате работы над ТЗ был составлен документ, описывающий моменты разработки, требования к системе и пр. И теперь на его основе можно начинать непосредственный процесс разработки продукта. С материалами ТЗ можно ознакомиться в Приложении №2.

Предполагается расширять ТЗ и документацию уже вне разработок ВКР, но при дальнейшей работе над продуктом.

Глава 4. Создание диаграмм и проектирование архитектуры

4.1 Общая архитектура приложения

Для данного приложения лучшим решением является клиент-серверная архитектура. В архитектуре клиент-сервер сервер является обработчика клиентских запросов. Но можно сказать, что концептуально сервер выступает и как сервер данных, и как обработчик запросов.

Взаимодействие клиент-сервер происходит благодаря использованию протокола передачи данных. Клиент посылает запрос, используя протокол, понятный серверу. Не особо важно, какой именно тип клиента отправляет запрос и по какому протоколу, важно, чтобы и сервер, и клиент одновременно и совместно использовали один и тот же протокол передачи.

REST (Representational state transfer) был задуман как простой и однозначный интерфейс для управления данными, предполагавший базовые операции с сетевым хранилищем: извлечение данных (GET), сохранение (POST), изменение (PUT/PATCH) и удаление (DELETE). Этот перечень всегда сопровождался такими опциями, как обработка ошибок в запросе, разграничение доступа к данным и валидация входящих данных. [9]

REST имеет ряд архитектурных принципов:

- Независимость сервера от клиента - серверы и клиенты могут быть мгновенно заменены другими независимо друг от друга, так как интерфейс между ними не меняется.
- Уникальность адресов ресурсов - каждая единица данных имеет свой собственный уникальный URL, который является идентификатором.
- Независимость формата хранения данных от формата их передачи - сервер может поддерживать несколько различных форматов для

передачи одних и тех же данных, но хранит их в своем внутреннем формате, независимо от поддерживаемых.

- Присутствие в ответе всех необходимых метаданных - помимо самих данных сервер должен возвращать детали обработки запроса, например, сообщения об ошибках, различные свойства ресурса, необходимые для дальнейшей работы с ним, например, общее число записей в коллекции для правильного отображения постраничной навигации.

Тип данных в общении между клиентом и сервером – это объект, то есть перечень свойств и соответствующих им значений. Объект не обязательно будет реальной сущностью, хранящейся в базе данных в том виде, в котором он отправлен или получен. Например, учетные данные для авторизации передаются в виде объекта, но не являются самостоятельной сущностью. Наиболее функциональным и удобным форматом обмена данными является JSON. [14]

Чаще всего в общих информационных системах встречаются следующие 4 слоя:

- Слой представления
- Слой приложения
- Слой бизнес-логики
- Слой доступа к данным

Все вышеописанное будет использовано как при дальнейшем проектировании, так и при непосредственной разработке приложения. Результатом данного анализа можно назвать п. 3.5 Приложения №3.

4.2 Варианты использования

Диаграмма вариантов использования (англ. use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. [10]

Для построения данных диаграмм отлично подходит UML (англ. "Unified Modeling Language") – стандартизированный язык моделирования при проектировании программ, а для реализации – приложение draw.io.

Для того, чтобы описать основные варианты использования приложения, необходимо четко представлять, кто будет им пользоваться.

В нашей структуре таких типов персон две – незарегистрированный пользователь и контент-администратор. Отличаются они только тем, что контент-администратор имеет возможность изменять данные о СО, добавлять новые или удалять. Поэтому основным вариантом использования стоит считать тот, что относится к незарегистрированному пользователю – он базовый.

С результатами данного этапа проектирования можно ознакомиться в Приложении №3 – Рис. 2 (Действующие лица), Рис. 3 (ВИ контент-администратора) и Рис. 4 (ВИ незарегистрированного пользователя). Все это поможет в дальнейшем правильно составить карту сайта и макет интерфейса.

4.3 Карта сайта

На этом этапе планируется подробная иерархия страниц, их структура. Такой прототип должен быть понятен всем участникам процесса (программистами, дизайнерам, разработчикам, контент-менеджерам). Формализованная структура сайта – карта сайта, - становится отправной точкой.

Визуальная карта сайта — это скелет будущего сайта, изображённый графически, чётко структурированный и дающий информацию для оценки будущей работы.

Как правило, в ней содержится только информация о структуре страниц, но в последнее время появляются новые инструменты, где можно детализировать информацию и внести необходимые данные о визуальном или текстовом контенте проекта.

Чтобы разработать визуальный ориентир для сайта — можно просто перенести мысли на бумагу. Это первый простой (и любимый дизайнерами) способ. Просто рисуете в произвольной форме блоки и фигуры, прописываете, где и какой контент должен быть. Делаете это в любимом скетчбуке, блокноте. Или используете стикеры на доске.

Способ хороший, он пахнет старыми книгами. Но это не очень продуктивно при возможных изменениях со стороны заказчика. К тому же, мы живем в мире облачных технологий, почему бы не пользоваться более удобными инструментами?

Можно, например, задействовать Sketch или любой другой графический редактор. Проблемы могут возникнуть, если потребуется внести изменения на встрече с клиентом или дополнить карту новыми элементами. То есть, проблемы всё те же.

Последний вариант — специализированные сервисы по созданию визуальной карты. Это оптимальное решение, интерфейс удобный, и можно комментировать, делиться, хранить и обрабатывать информацию, не теряя во времени.

На правильной карте сайта сразу виден масштаб работ и цепочки взаимосвязей. От главной вы движетесь к внутренним страницам,

углубляясь в разделы и подразделы. Иерархическая структура зависит от проекта, но есть несколько обязательных условий.

На карте сайта должно быть понятно, какая страница к чему относится. Это достигается за счёт соединительных линий (связей) и вертикальной последовательности блоков (страниц). В специализированных сервисах это делается автоматически, при создании новых страниц.

Несмотря на всестороннее развитие и общедоступность облачных сервисов, качественных решений для создания визуальной карты сайта немного. Мною был выбран FlowMapp — профессиональный инструмент для подготовки визуальной карты сайта, сочетающий в себе возможности облачного хранилища файлов и проектной документации.

В сервисе есть функции создания и редактирования интерактивной карты, комментирования страниц, создания структуры каждой страницы, а также функция экспорта проектов в различные форматы.

Именно с помощью FlowMapp мне удалось создать удобоваримую карту сайта, с которой можно ознакомиться в Приложении №3, Рис. 4. Также прямо в приложении FlowMapp я описала каждую из страниц:

1. Главная страница

Главная страница должна в общих чертах описывать суть веб-сайта и приглашать пользователя воспользоваться его функционалом, то есть поискать СО на карте или с помощью страницы поиска.

1.1. Поиск СО

Страница поиска СО с возможностью фильтрации. На странице будет использована пагинация. Параметры фильтрации уточняются.

1.1.1. Информация о СО

Полная информация о СО. У администратора контента есть возможности: изменить поля, сохранить изменения, удалить объект. Названия полей уточняются. Основные предполагаемые поля:

- адрес
- название
- фото (загружается с помощью окна загрузки фотографий)
- доступность для инвалидов
- вид собственности (государственная/частная)
- экономическая доступность (бесплатный/платный доступ на СО)
- тип СО (фитнес-центр, спортивная площадка, детская спортивная школа, бассейн и т.д.)
- описание типов деятельности (какими видами спорта можно заниматься на данном СО)

1.2. Карта СО

Интерактивная карта Красногвардейского района с отмеченными на ней СО. Необходимы обозначения основных объектов для ориентации на территории (улицы, известные здания и т.д.). Можно выбрать интересующий СО и откроется модуль с основной информацией. Отсюда можно перейти на страницу с полной информацией.

1.2.1. Информация о СО – аналогично п. 1.1.1

1.3. О нас

Страница с информацией об Администрации Красногвардейского района, секторе спорта и физической культуры. Есть форма для предложения нового СО (см. следующий блок).

1.3.1. Форма для предложения нового СО

С помощью данной формы любой пользователь сможет предлагать добавление нового СО в базу. Далее формируется электронное письмо, отправляемое на почту контент-администратора. Контент-администратор вручную вводит данные о СО. Поля уточняются. Среди обязательных:

- адрес
- название
- назначение
- прочая информация
- e-mail для связи.

2. Вход администратора контента

Страница аутентификации пользователя с ролью "администратор контента". Должна содержать в себе поля для логина и пароля. При неверных данных - соответствующее всплывающее окно об ошибке "Неправильный логин или пароль". Страница не будет находиться в общей карте сайта, ссылка на нее должна быть напрямую передана администратору контента. После аутентификации пользователю предоставлены страницы, аналогичные тем, что предоставлены незарегистрированному пользователю. Но страницы видоизменены: Всегда есть упоминание о том, что человек прошел аутентификацию (например, в правом верхнем углу надпись: "Приветствую, *ИМЯ_ПОЛЬЗОВАТЕЛЯ*!") На страницах со списком СО и картой СО появляется опция "Добавить СО". Происходит переход на страницу, аналогичную странице с информацией о СО, но вместо данных – формы для их введения. Есть возможность добавить информацию. На странице с информацией о СО добавляется функционал изменения данных или удаления объекта.

4.4 Макеты интерфейса

Макет – это статическое визуальное представление концепции пользовательского интерфейса.

Для чего можно использовать макеты:

1) Обсуждение функциональности с заказчиком. Заказчики, как правило, люди очень занятые и зачастую физически не способные выделить время на чтение многостраничных документов (даже если это в их интересах). В этом смысле картинка стоит тысячи слов: она гарантирует, что у заказчика появится четкое представление о том, что именно будет сделано.

2) Оценка юзабилити. Макеты – это фактически первый артефакт в проекте, юзабилити которого уже можно и нужно оценивать. На этом этапе проще и дешевле всего устранить проблемы, если они обнаружатся.

3) Постановка задачи разработчикам. Наличие наглядной иллюстрации того, что должно быть сделано, гораздо полезнее, чем длинное текстовое описание (хотя, разумеется, оно также должно присутствовать, ибо не все можно отобразить на макете).

4) Постановка задачи дизайнерам. Это позволяет с большей простотой развивать дизайн продукта, а также не позволяет уходить от макета слишком далеко, дабы не испортить визуальную составляющую, запланированную в том числе самим заказчиком.

На сегодняшний день существует большое количество приложений, позволяющих легко и быстро сделать макеты интерфейса, составить их из готовых базовых элементов. К сожалению, все самые лучшие и удобные являются либо платными, либо сильно урезанными в бесплатном демо-варианте, поэтому они не подходят для данного проекта.

В силу вышесказанного, я выбрала бесплатный и простой в использовании инструмент – draw.io. В нем гармонизирован функционал и

минималистичность. Вообще в данном приложении можно проектировать практически все, что касается продукта. Но, конечно, без всяких изысков, какие, например, существуют в FlowMapp.

Итак, благодаря этому инструменту и уже готовой карте сайта мною был спроектирован интерфейс приложения с попыткой представить максимальный в своем юзабилити продукт. С макетом можно ознакомиться в Приложении №3, Рис. 5 (Макеты для роли администратора контента) и Рис. 6 (Макеты для роли незарегистрированного пользователя).

Логичным «продолжением» этого макета будет является дизайн-проект, для которого уже подобраны референсы, предоставленные в ТЗ (Приложение №2, Рис. 7). Примеры были подобраны на платформе Behance, в описании иллюстрации предоставлена ссылка на коллекцию работ.

4.5 Эндпоинты

Проект структуры эндпоинтов – важная составляющая любой проектной документации. Ведь именно веб-адреса позволяют взаимодействовать с сервером бизнес-логики.

Для данного этапа в качестве инструмента было выбрано приложение Postman.

Основное предназначение приложения Postman — создание коллекций с запросами к вашему API. Любой разработчик или тестировщик, открыв коллекцию, сможет с лёгкостью разобраться в работе сервиса. Ко всему прочему, Postman позволяет проектировать дизайн API и создавать на его основе Mock-сервер.

Ниже приведен список спроектированных эндпоинтов, а непосредственно API документацию в формате Open API можно увидеть в Приложении №1.

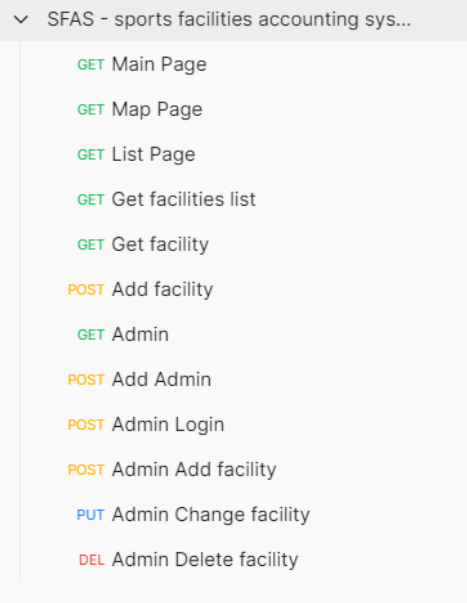

Рисунок 1 Список эндпоинтов

4.6 Модели объектов

Моделирование данных — это создание визуального представления о всей информационной системе либо ее части. Цель в том, чтобы проиллюстрировать типы данных, которые используются и хранятся в системе, отношения между этими типами данных, способы группировки и организации данных, их форматы и атрибуты.

Моделирование упрощает просмотр и понимание взаимосвязей между данными для разработчиков, архитекторов данных, бизнес-аналитиков и других заинтересованных лиц. Кроме того, моделирование данных помогает:

- Уменьшить количество ошибок при разработке программного обеспечения и баз данных.
- Унифицировать документацию на предприятии.
- Повысить производительность приложений и баз данных.
- Упростить отображение данных по всей организации.
- Улучшить взаимодействие между разработчиками и командами бизнес-аналитики.

- Упростить и ускорить процесс проектирования базы данных на концептуальном, логическом и физическом уровнях.

На данном этапе осведомленности о потребностях заказчика и виде данных о СО, удалось оформить лишь очень примерные модели сущностей. С ними можно ознакомиться в Приложении №3, Рис. 8 (Диаграмма классов СУОС).

Глава 5. Проектирование дизайна интерфейса

5.1 Дизайн-проект

5.1.1 Айдентика

Чтобы создать успешный и узнаваемый продукт или услугу в эпоху потребительства уже недостаточно просто качества. Придется позаботиться о маркетинге, а он начинается с айдентики. Айдентика – это те визуальные элементы, из которых складывается образ компании. По этим деталям мы узнаем бренды, точно так же, как по внешним признакам среди толпы мы узнаем своих знакомых.

Веб-приложение – тоже продукт, нуждающийся в разработке айдентики, в особенности логотипа и цветовой гаммы. Именно с этого и начинается проработка дизайна.

Если маркетинг начинается с айдентики, то айдентика с названия. От того насколько оно понятное и запоминающееся зависит успех новой компании. Важно подобрать такое название, к которому можно будет зарегистрировать соответствующий домен.

SportsMap – идеальное название для данного проекта. Оно отражает главнейшие составляющие карты спортивных объектов. Подходящих доменов для сайта в достатке, в мире под таким названием существует только американская спортивная радиосеть (https://www.sportsmapradio.com/).

Разработка логотипа – это основная и важнейшая задача разработки айдентики. Логотип – это знак, который соотносится с вашим брендом и транслирует его основные ценности. Он должен сочетаться с названием и слоганом. Намекать или прямо указывать на сферу вашей деятельности. Быть запоминающимся и простым, масштабируемым, хорошо смотреться

на разных носителях – как на маленькой визитке, так и на большом билборде.

Мною было разработано несколько вариантов логотипа, но действительно подходящим оказался только один – сочетание маркера геопозиции на онлайн-картах и баскетбольного мяча внутри. Достаточно простой, запоминающийся и точный символ для логотипа.

Логотип был создан в нескольких размерах, в векторном формате с помощью программы Adobe Illustrator – самого распространенного ПО для дизайнеров векторной графики, коей и являются логотипы. Векторный формат логотипа – обязательно условие, потому как на сайте все-таки вектор имеет преимущество. Вектор дает возможность изменять размер объекта без потери качества, так как такая графика лишь содержит «способ» построения картинки, который применяется при каждой загрузке файла.

Следующим этапом стал подбор цветовой гаммы. Мною было отобрано 16 цветов, из которых затем постепенно отбирались лучшие по результатам голосования потенциальных потребителей продукта. Группа опрашиваемых была в достаточной мере репрезентативна – многообразие возрастов, сфер деятельности, интересов.

На рис. 1 представлены «финалисты» – шесть наиболее удачных оттенков. После общения с заказчиком было принято решение использовать именно первый оттенок, как самый свежий и яркий, что отлично сочетается с тематикой продукта.

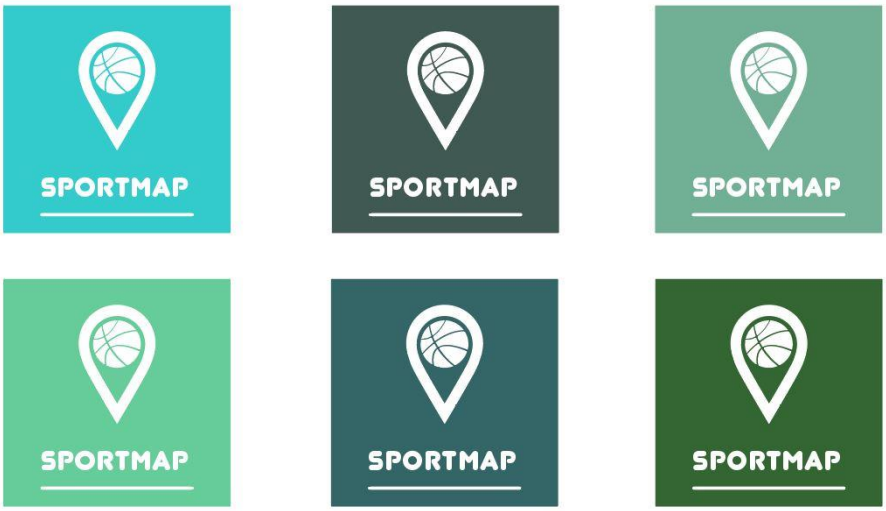

Рисунок 2 Варианты оттенков с логотипом

Непосредственно для дизайна интерфейса была разработана палетка оттенков разной яркости, вариации выбранного основного цвета. Ведь одним цветом на сайте не ограничиться, нужно как минимум с помощью цвета показывать состояние объектов вроде кнопок и т.д.

В будущем с расширением возможностей и значимости продукта имеет смысл разработать и брендбук – документ, который содержит полную информацию о компании: ее философия, цели, описание логотипа и фирменного стиля, правила оформления рекламных носителей и многое другое.

5.1.2 UI/UX проектирование

UX — это User Experience (дословно «опыт пользователя»). То есть это то, какой опыт/впечатление получает пользователь от работы с интерфейсом. Удается ли ему достичь цели и на сколько просто или сложно это сделать.

UI — это User Interface (дословно «пользовательский интерфейс») — то, как выглядит интерфейс и то, какие физические характеристики приобретает. Определяет, какого цвета будет ваше «изделие», удобно ли

будет человеку попадать пальцем в кнопки, читабельным ли будет текст и тому подобное.

UX/UI проектирование — это проектирование пользовательских интерфейсов, в которых удобство использования так же важно, как и внешний вид. Изучив и применив основные принципы такого проектирования можно достичь максимальной полезности и удобства приложения для пользователей, что конечно же сказывается на успехе продукта. Именно к этому и было стремление на данном этапе работы.

Огромное значение для юзабилити приложения имеет типографика. Было принято решение использовать в веб-приложении следующую комбинацию шрифтов:

- Quicksand
- Open Sans
- Junegull (для логотипа)

Все шрифты распространяются бесплатно и были найдены на платформе Google Fonts. Именно эти шрифты оказались наиболее подходящими и удобными для восприятия.

Мною были применены различные методики по улучшению юзабилити приложения, а именно: использование user-friendly иллюстраций, созданных с помощью платформы Sapiens (Рис. 2); подбор единообразных иконок для отображения состояний и объектов (все иконки векторные, взяты с платформы Flaticon https://www.flaticon.com/); организация состояний объектов при взаимодействии и проч.

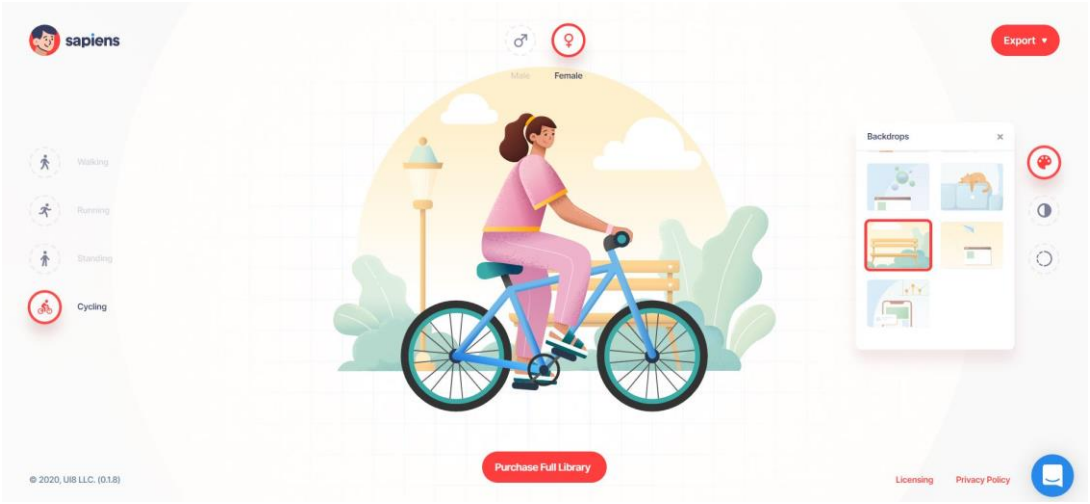

Рисунок 3 Пример создания иллюстрации на платформе Sapiens

Дизайн-проект было решено создавать с помощью такой программы как Figma. Это один из самых удобных, современных и популярных инструментов как для веб-дизайна, так и для прототипирования. Он позволяет создавать макеты, максимально адаптированные для разработчиков, которые эти макеты должны воплощать в жизнь.

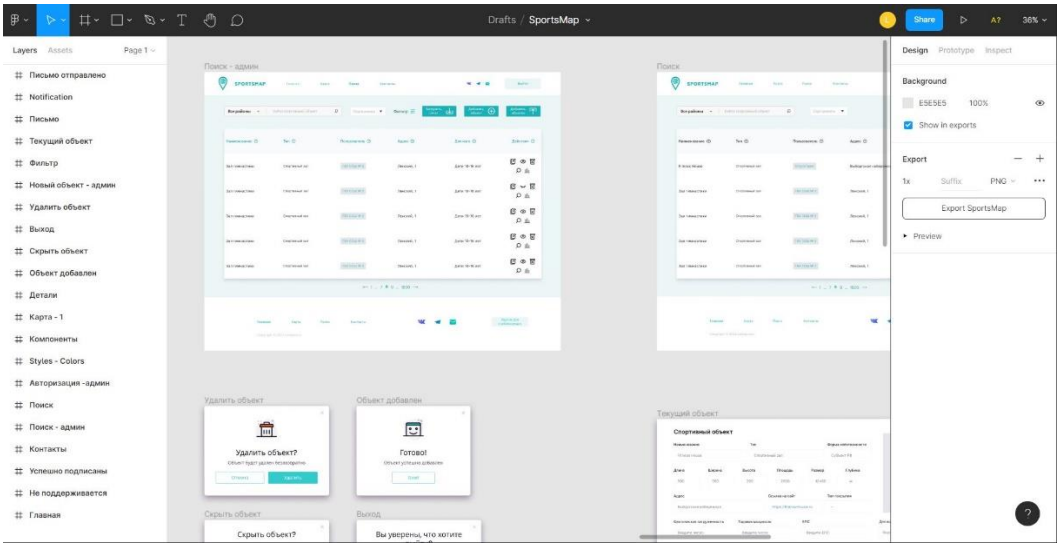

Рисунок 4 Пример интерфейса Figma в процессе работы

В дизайн-проекте мною проработаны возможные состояния объектов, дополнительные всплывающие модули, обратная связь пользователю о результате совершения тех или иных действий.

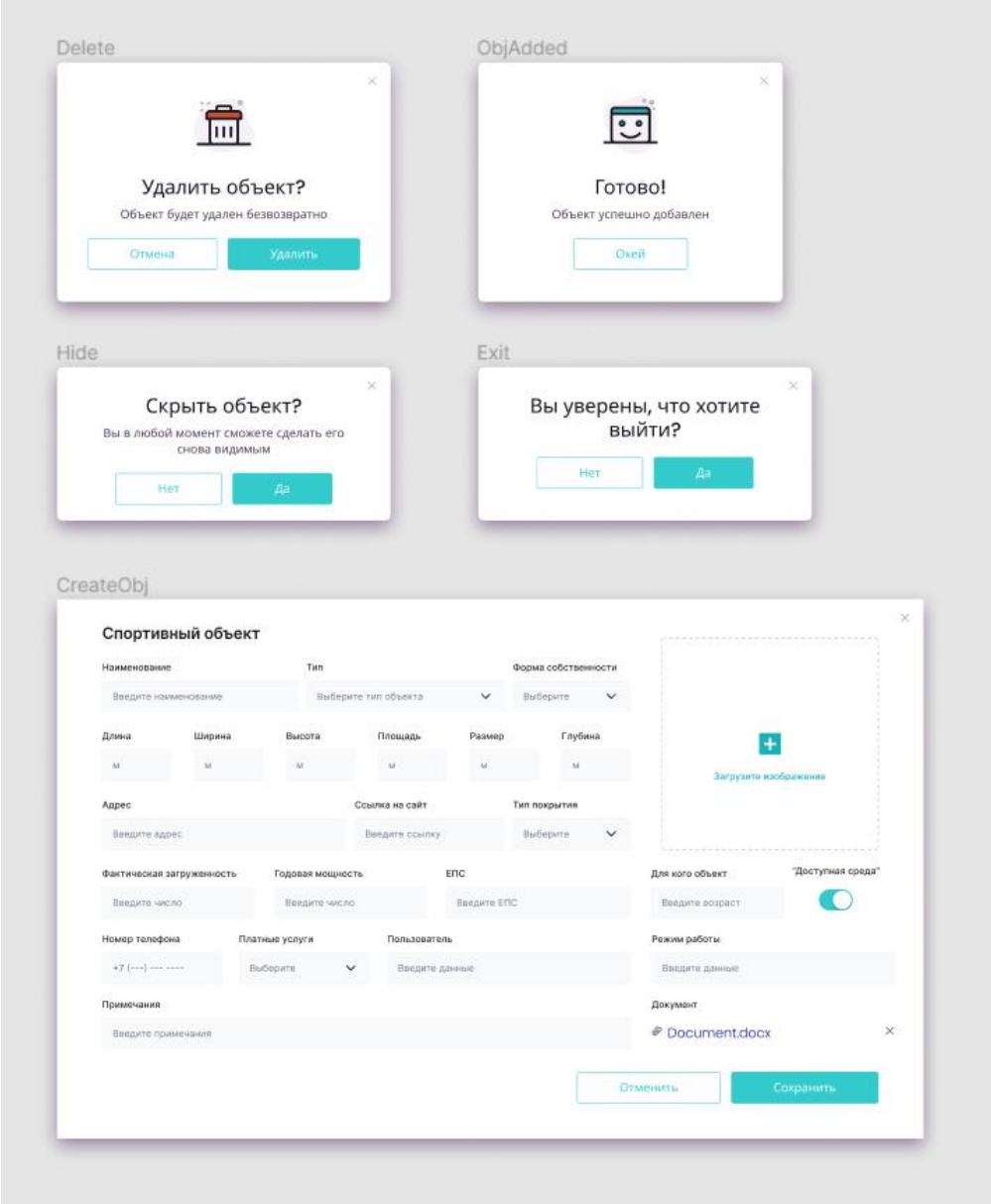

Рисунок 5 Пример модулей «SportsMap»

5.2 Прототипирование в Figma

В Figma предусмотрена функция прототипирования, которая помогает «запустить» приложение и отловить все ошибки в сценариях до передачи макетов в разработку. Но прежде чем запускать, нужно проработать все взаимодействия компонентов, указать их состояния, добавить необходимые ссылки и выстроить иерархию. Благодаря этому, у меня получилось воспользоваться (хоть и в ограниченной мере) приложением еще до его непосредственного создания в коде. Это позволило

увидеть недостатки дизайна и исправить их, также выстроилась четкая картина всей бизнес-логики.

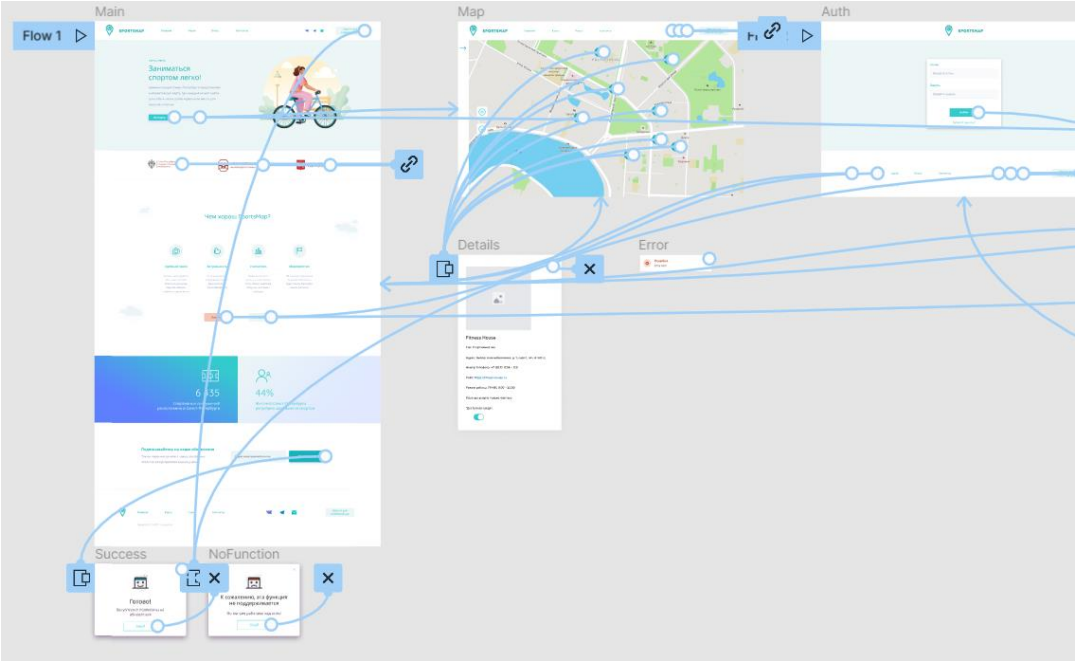

Рисунок 6 Пример взаимодействия объектов для прототипирования

5.3 Результаты

По итогам этапа проработки дизайна и прототипирования приложения заказчику была представлена ознакомительная версия для тестирования, где все стороны были удовлетворены.

Получились удачные макеты, удобные в использовании для дальнейшей разработки, и позволяющие еще до самой разработки увидеть в какой-то степени ее результат, попробовать в деле.

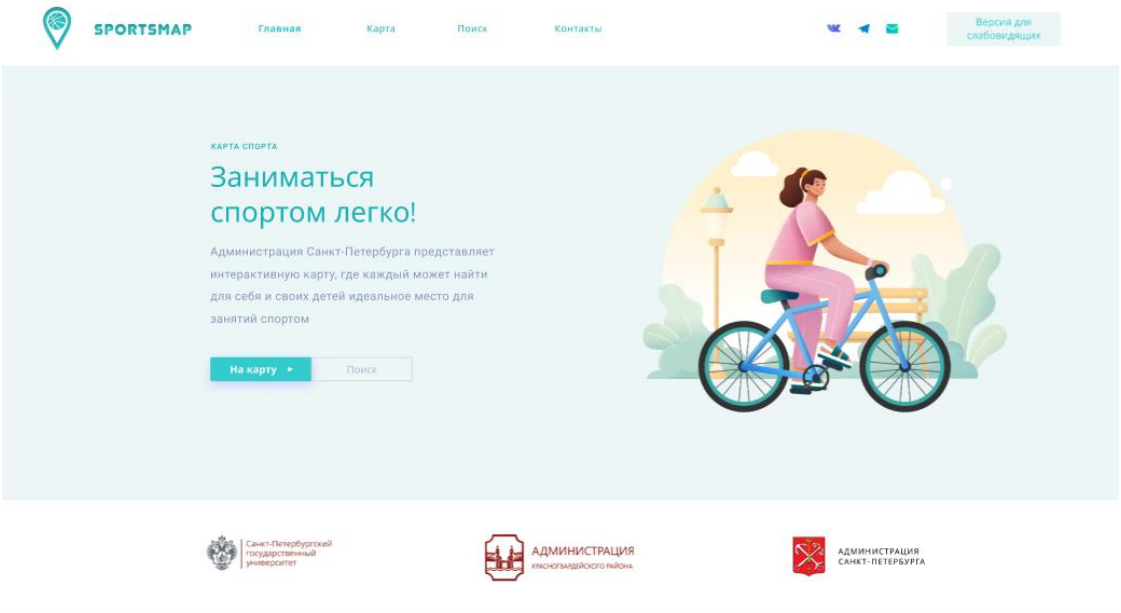

Рисунок 7 Главная страница

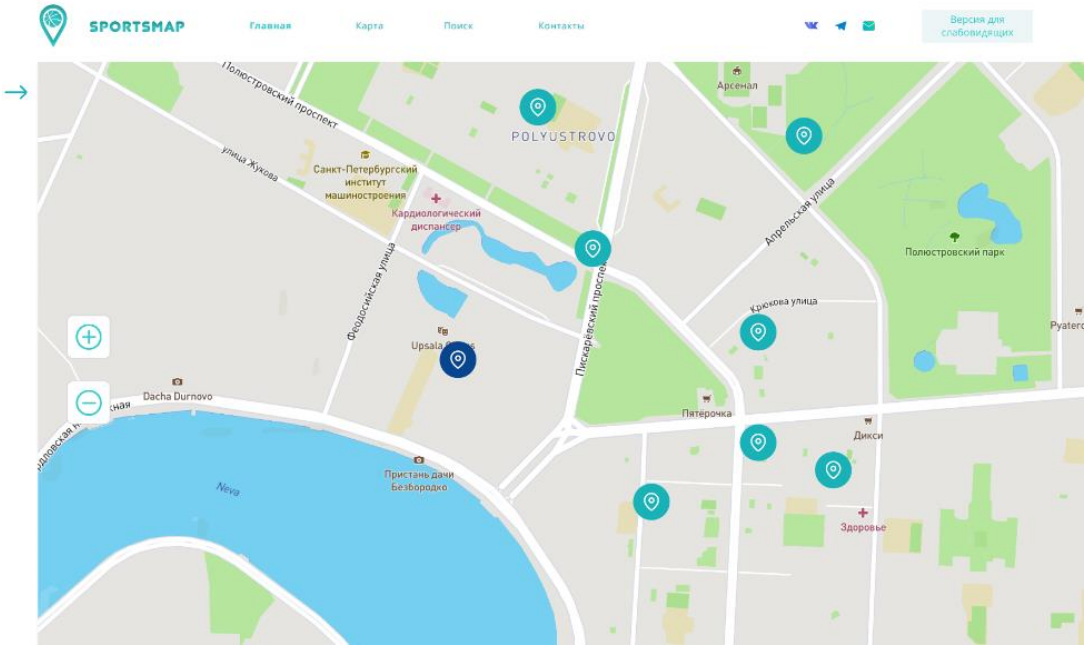

Рисунок 8 Карта

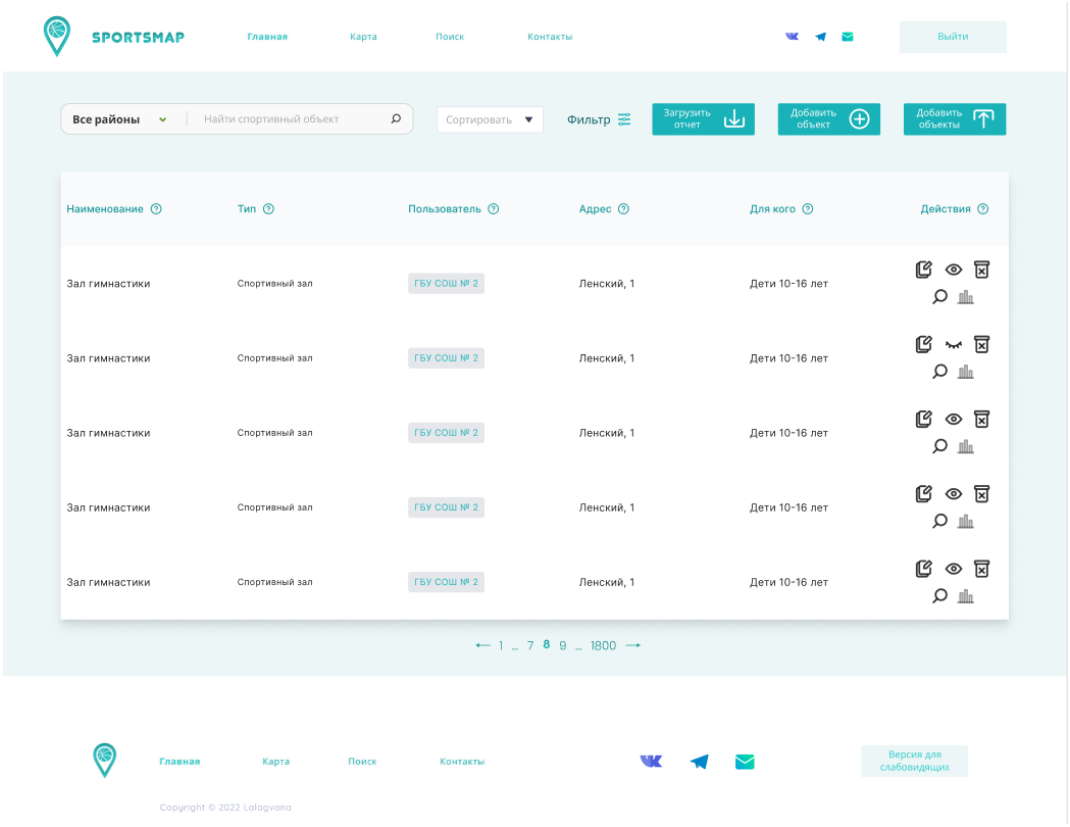

Рисунок 9 Поиск

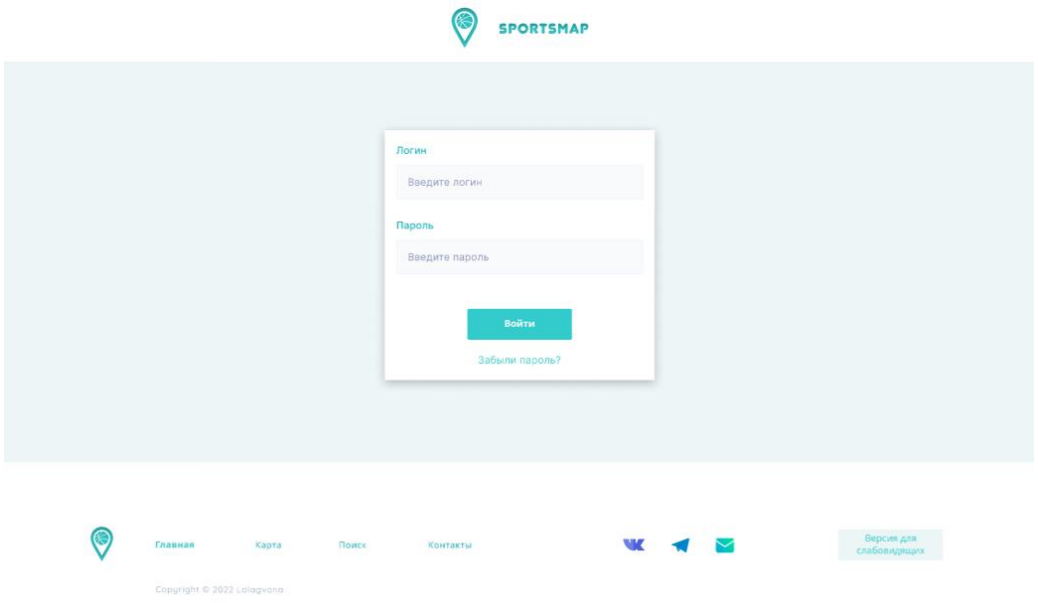

Рисунок 10 Вход

Глава 6. Бэкенд-разработка

6.1 Обзор программных решений и инструментов

Как известно экономическая устойчивость современных предприятий напрямую зависит от их информационного обеспечения. Развитие информационных технологий и систем, обеспечивающих работу таких предприятий, идет стремительно. Информационные системы, как правило, направлены на обеспечение бесперебойной работы предприятий, повышение экономической стабильности и автоматизации рутинных операций различных отделов предприятий. Общим правилом при реализации информационных систем является использование клиент-серверной архитектуры, которая предполагает наличие клиентской (фронтенд) и серверной (бэкенд) частей.

Наиболее популярными языками программирования при разработке серверной части являются Python, JavaScript, Java, PHP, C#. Однако в чистом виде данные языки редко применяются. Все чаще разработка ведется при помощи специальных программных сред, называемых фреймворками.

Фреймворки содержат в себе некий каркас программы, который позволяет ускорить разработку систем путем переиспользования компонентов, входящих в состав фреймворка, а также добавления компонентов, характеризующих особенности конкретной системы.

.NET - фреймворк, который использует Microsoft Visual Studio и предоставляет разработчику различные возможности использования фреймвор-ков, реализованных на других языках программирования, таких как Python, C#, C++ и т.д. Это мощнейший инструмент для разработки программного обеспечения [17].

Платформа ASP.NET Core от корпорации Microsoft, которая появилась сравнительно недавно [16], продолжает набирать свою

популярность. Проанализируем преимущества и недостатки новой технологии в сравнении со старой, обосновывая тем самым использование именно ASP.NET Core в текущем проекте.

Преимущество 1. Единый набор технологий для шаблонов MVC и API.

В ASP.NET MVC на начальном этапе предлагается выбрать шаблон проектирования, от которого будет сильно зависеть архитектура приложения, поскольку шаблоны требуют различных зависимостей. В ASP.NET Core же нет различий между построением проекта на фреймворках MVC или API, программист может переиспользовать компоненты.

Преимущество 2. Изменения структуры проекта

Структура проекта на ASP.NET Core более лаконична и не требует большинства файлов, которые программисту могут и не потребоваться. Приложение на ASP.NET Core аналогично обычному консольному приложению по своей структуре: имеется единая точка входа – файл Program со статическим методом Main, что позволяет отлаживать программу в едином стиле.

Также больше не требуется использовать файлы конфигурации. В ASP.NET Core можно гибко работать с конфигурацией приложения в любом формате (JSON, XML, CSV).

Преимущество 3. Кроссплатформенность

Приложения на ASP.NET MVC требуют для своего запуска среду исполнения .NET, что позволяет запускать такие приложения на Unix-подобных операционных системах только с использованием специальных сторонних средств, например, Mono. С технологией ASP.NET Core это не обязательно, потому что технология является полностью

кроссплатформенной и не требует зависимостей от ядра систем на базе Windows [16].

Преимущество 4. Встроенная поддержка внедрения зависимостей.

Внедрение зависимостей или Dependency Injection (DI) - это подход в программировании, позволяющий гибко управлять реализациями определенных интерфейсов во всем приложении [16]. Такой механизм позволяет писать слабо связанный код, очень удобный для модульного и интеграционного тестирования. В ASP.NET Core используется гибкая встроенная поддержка DI, не требующая подключения дополнительных программных библиотек. Однако по желанию программист может использовать привычные ему средства DI или даже реализовать свои.

Преимущество 5. Сквозной проброс зависимостей и единый пакет SDK

В процессе длительной работы над программным обеспечением, ошибки и уязвимости сторонних зависимостей устраняются, старые версии перестают поддерживаться, поэтому некоторые компоненты приложения следует постоянно обновлять для обеспечения безопасности [17].

Для приложений, написанных на ASP.NET MVC, общие зависимости необходимо подключать в каждый проект решения вследствие особенностей платформы. В будущем это может вызвать определенные проблемы. Например, если корневая сборка содержит зависимость, то все остальные проекты в данном решении должны подключить и скопировать эту зависимость локально.

В ASP.NET Core зависимость будет пробрасываться до вышележащих сборок вместе со ссылкой на корневую сборку. Это позволяет использовать одну и ту же версию программной библиотеки во всех проектах приложения.

Проекты решения, как правило, содержат одни и те же корневые зависимости для разработки, поэтому программирование на платформе ASP.NET Core зависит от целого набора таких зависимостей, внутри которого поддерживается строгое соблюдение версионности всех пакетов. Это обеспечивает их совместимость, потому что невозможно изменить отдельную зависимость без обновления всего пакета. Дополнительно такие наборы пакетов позволяют сэкономить место на диске, поскольку они расположены в локальной папке пользователя компьютера и не копируются в папки проектов приложения, а представляются ссылкой.

Преимущество 6. Открытый исходный код

В отличие от ASP.NET MVC исходный код платформы ASP.NET Core полностью в открытом доступе. Проект все так же развивается под руководством Microsoft, однако теперь любой желающий может отслеживать изменения и даже завести задачу для решения проблемы.

ASP.NET Core имеет и ряд недостатков, но они абсолютно несущественны ввиду специфики проекта. С уверенностью можно сказать, что ASP.NET Core в скором будущем не только заменит ASP.NET MVC, но и составит конкуренцию иным технологиям для создания веб-приложений. [16] Поэтому это действительно стоящий использования фреймворк.

6.2 Принципы SOLID.

При любой разработке имеются принципы, которым следуют программисты. Эти принципы были созданы выдающимися представителями этой профессии и стали уже классикой разработки.

К таким базовым принципам относится SOLID (сокр. от англ. single responsibility, open–closed, Liskov substitution, interface segregation и dependency inversion) — акроним для первых пяти принципов

объектно-ориентированного программирования и проектирования, названных Робертом Мартином в начале 2000-х.

При создании программных систем использование принципов SOLID способствует созданию такой системы, которую будет легко поддерживать и расширять в течение долгого времени. Принципы SOLID — это руководства, которые также могут применяться во время работы над существующим программным обеспечением для его улучшения. Стратегии гибкой и адаптивной разработки предполагают написание кода с соблюдением принципов SOLID. И, конечно же, они будут использованы в текущем проекте.

Избавиться от «признаков плохого проекта» помогают следующие 5 принципов [11]:

Инициал	Представляет	Название, понятие
S	SRP	Принцип единственной ответственности (single responsibility principle) Для каждого класса должно быть определено единственное назначение. Все ресурсы, необходимые для его осуществления, должны быть инкапсулированы в этот класс и подчинены только этой задаче.
O	OCP	Принцип открытости/закрытости (open-closed principle)

		«программные сущности должны быть открыты для расширения, но закрыты для модификации».
L	LSP	Принцип подстановки Лисков (Liskov substitution principle) «объекты в программе должны быть заменяемыми на экземпляры их подтипов без изменения правильности выполнения программы».
I	ISP	Принцип разделения интерфейса (interface segregation principle) «много интерфейсов, специально предназначенных для клиентов, лучше, чем один интерфейс общего назначения».
D	DIP	Принцип инверсии зависимостей (dependency inversion principle) «Зависимость на Абстракциях. Нет зависимости на что-то конкретное».

6.3 Создание и управление базой данных с помощью Entity Framework Core

БД - это набор файлов, содержащих информацию. Разрабатывая базу данных для пользователя, программист создает программу, которая обеспечивает работу с файлами данных [8].

Entity Framework является объектно-реляционным модулем сопоставления (O/RM), который позволяет разработчикам .NET работать с базой данных, используя объекты .NET.

Центральной концепцией Entity Framework является понятие сущности – набора данных, ассоциированного с определенным объектом. Так что данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Основу функциональности Entity Framework составляют следующие классы:

- DbContext: определяет контекст данных, используемый для взаимодействия с БД;

- DbModelBuilder: сопоставляет классы на языке C# с сущностями в БД;

- DbSet/DbSet<TEntity>: определяет набор сущностей, хранящихся в БД

Для обеспечения работы с БД в любом приложении через Entity Framework применяется класс UserContext, который взаимодействует с таблицами из БД. В конструкторе этого класса вызывается конструктор базового класса, в который передается имя строки подключения к базе данных.

Также в классе определено свойство Users, которое будет хранить набор объектов User. В классе контекста данных набор объектов представляет класс DbSet<T>. Через это свойство будет осуществляться связь с таблицей объектов User в БД.

Открытая архитектура, простота и доступность использования Entity Framework делают его исключительно эффективным относительно любого другого решения данных задач. [15]

Для работы с базами данных мною использовался Entity Framework Core. Entity Framework (EF) Core — это кроссплатформенная и расширяемая версия технологии доступа к данным Entity Framework. EF Core может использоваться как объектно-реляционный модуль сопоставления (O/RM), позволяя разработчикам .NET работать с базой данных с помощью объектов .NET. Использование моделей написанных на C# позволяет в значительной мере упростить процесс создания приложения.

EF Core поддерживает множество систем баз данных. Поэтому если возникнет необходимость сменить СУБД, то основные изменения в проекте будут касаться лишь конфигурации подключения к провайдерам БД. Код, который непосредственно работает с данными, останется прежним. Эта особенность позволяет использовать различные типы БД для разных сред [15].

Мною были созданы модели сущностей, подключена база данных, организованы автоматические миграции, а также создан DbContext, позволяющий с легкостью работать с данными из базы в коде. Для целевой СУБД введена вариативность – в зависимости от ситуации и потребностей можно выбрать MSSQL Server, а можно PostgreSQL.

6.4 Контроллеры, сервисы, DTO-модели

Стоит выделить некоторые ключевые моменты организации кода и применения паттернов. В данном проекте таковыми выступают контроллер, сервисы, DTO-модели.

Контроллер – это компонент, который обеспечивает связь между пользователем и приложением, представлением и хранилищем данных. Он содержит логику обработки запроса пользователя. Контроллер получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления, наполненного данными моделей.

Такое разграничение компонентов приложения позволяет реализовать концепцию разделение ответственности, при которой каждый компонент отвечает за свою строго очерченную сферу. В связи с чем легче построить работу над отдельными компонентами. И благодаря этому приложение легче разрабатывать, поддерживать и тестировать отдельные компоненты.

В данном проекте были созданы отдельные контроллеры для работы с основными сущностями:

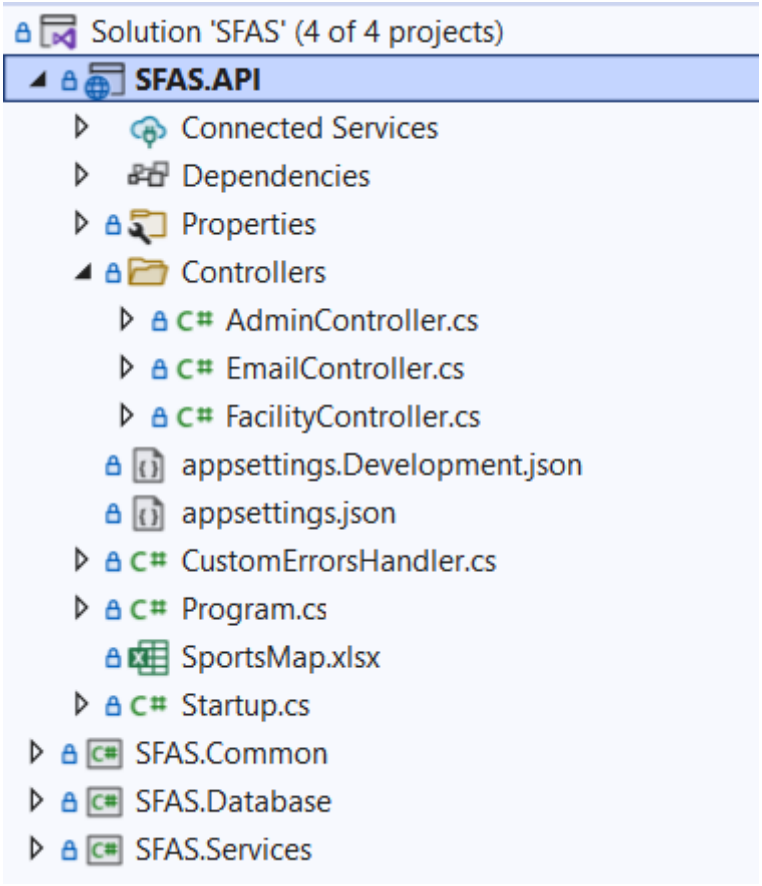

Рисунок 11 Структура проекта

Сервисы в данном проекте – основной источник бизнес-логики. Именно в сервисах происходит непосредственная обработка данных, их преобразование в нужный вид, работа с базой данных, валидация. Здесь сущностных единиц уже больше, так как сервисы, в отличие от контроллеров, не работают вовне, а значит содержат в себе внутреннюю работу и взаимодействие. Таких действий происходит, разумеется, количественно больше.

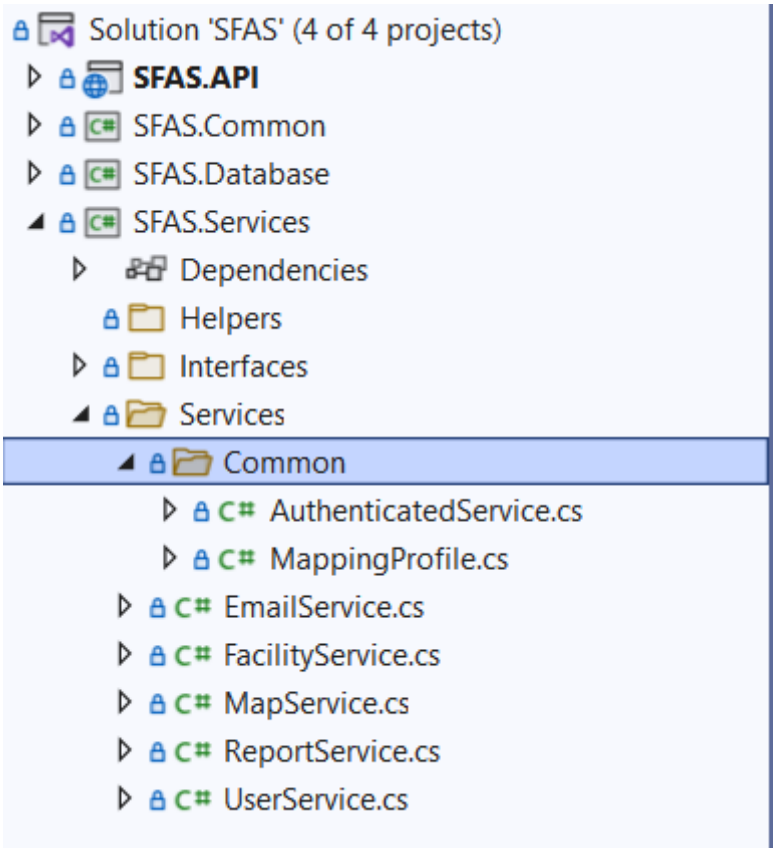

Рисунок 12 Структура проекта

DTO-модели – это объекты, которые переносят данные между процессами, чтобы уменьшить количество вызовов методов. Эта модель была впервые введена Мартином Фаулером в его книге EAA.

Фаулер объяснил, что основная цель – сократить количество обращений к серверу за счет поиска нескольких параметров в одном вызове. Это снижает нагрузку на сеть при таких операциях.

Другим преимуществом является инкапсуляция логики сериализации – механизм, который преобразует структуру объекта и данные в определенный формат, который можно сохранять и передавать. Это обеспечивает единую точку изменения в нюансах сериализации. Он также отделяет модели предметной области от уровня представления, позволяя обоим изменяться независимо.

Поэтому для каждого метода в контроллере в качестве параметров и модели результата мною были созданы DTO-модели, содержащие только необходимые поля данных.

6.5 Обработка Excel-таблиц

В современном мире разработки приложений нередко встает необходимость работы с Excel документами. Чаще всего это разного рода отчеты, но иногда xls/x файлы используются в качестве хранилища данных. Например, если пользователь должен иметь возможность загрузить данные в приложение или выгрузить, в человеко-читаемом виде, Excel де-факто является стандартом. Относительно дружелюбный интерфейс, прозрачная структура, в купе с его распространенностью – трудно навскидку назвать решение лучше.

Так и в текущем проекте возникла необходимость внедрения работы с таблицами. От заказчика поступил запрос на добавление функционала загрузки и выгрузки информации об объектах в удобно оформленных таблицах. Поэтому встал вопрос о выборе библиотеки для реализации такого решения.

Пожалуй, самой подходящей является популярная библиотека EPPlus, она довольно хорошо отражает концепцию Excel.

В качестве параметра, по которому происходил выбор библиотеки, помимо удобства использования, стоит выделить скорость и качество работы. В таблице представлены результаты тестов [12], по которым можно увидеть, что EPPPlus отлично себя показывает. Поэтому такое решение идеально по соотношению удобства и скорости.

Показатель	Native	EPPlus Compiled, Mark-II	EPPlus 4EPPlus 5	NPOI	Spire	Excel Interop
Время инициализации (мс)	0	239	241	368	722	1640
Ср. время на 1 проход (мс)	0,0004	0,003	0,9174	1,8996	7,7647	50,7194
Ср.кв. отклонение	0,035884	0,0	0,0	0,0	0,0	n/a
Точность	98,79%	100,0%	100,0%	100,0%	100,0%	n/a
Ошибки	0,0%	0,0%	0,3%	0,3%	0,28%	0,3%

Рисунок 13 Сравнение библиотек для работы с Excel

6.6 Результаты

По итогам разработки серверной части и использования всех вышеупомянутых инструментов и принципов, получился чистый, понятный C# код, производящий обработку данных, эффективную работу с базой данных, а также генерирующий и принимающий в качестве источника данных таблицы Excel.

Глава 7: Фронтенд-разработка

7.1 Обзор программных решений и инструментов

ReactJS является библиотекой с открытым исходным кодом, направленной на создание пользовательских интерфейсов мобильных и web-приложений. React представил концепцию виртуального DOM, которая считается одной из наиболее значительных преимуществ ReactJS. Благодаря характеристике виртуального DOM (Document Object Model), React отличается исключительной производительностью. С точки зрения уровня сложности, React является одним из самых простых в освоении, особенно по сравнению с Angular. React приобрел известность благодаря архитектуре на основе компонентов, которую другие платформы начали использовать гораздо позднее. Такой структурный подход позволяет сравнительно быстро и просто создавать интерфейс. [13]

Создать страницу с минимальным содержимым, конечно, проще и быстрее с помощью классического подхода, так как не требуется ничего устанавливать. Это может быть удобно для создания небольших страниц. Однако данная процедура совершается единожды за весь проект, поэтому такое преимущество незначительно при разработке крупных приложений.

По скорости отрисовки страниц все же выигрывает способ разработки веб-приложений с использованием React. Загрузка такой страницы в среднем происходит в три раза быстрее, чем аналогичной, разработанной с помощью классического подхода. [13, 19]

Bootstrap — свободный набор инструментов для создания сайтов и веб-приложений, включающий в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Leaflet — библиотека с открытым исходным кодом, написанная на JavaScript, предназначенная для отображения карт на веб-сайтах. Поддерживает большинство мобильных и стационарных платформ из числа тех, что поддерживают HTML5 и CSS3.

Наряду с OpenLayers и Google Maps API — это одна из наиболее популярных картографических JavaScript-библиотек, использующаяся на таких крупных сайтах, как Flickr, Foursquare, Craigslist, Data.gov, IGN, проектах Викимедиа, OpenStreetMap, Meetup, WSJ, MapBox, CloudMade, C artoDB и других.

Leaflet позволяет разработчику, не знакомому с ГИС, легко отображать растровые карты, состоящие из маленьких фрагментов — тайлов, с, возможно, дополнительными слоями, накладываемыми поверх основного. Слои могут быть интерактивными, например, отображать подсказку при клике по маркеру.

По всем упомянутым причинам было принято решение изучать данные фреймворки и использовать их для разработки клиентской части приложения.

7.2 Реализация

Для начала мною был подключен инструмент NodeJS, благодаря которому можно без проблем запускать проекты и устанавливать новые пакеты через команды npm.

В качестве среды разработки был выбран VisualStudio Code как самый эффективный и удобный сервис. Он позволяет легко запускать проект, подключать новые модули и расширения, работать с Git изменениями, видеть подсвеченный код (это ускоряет разработку и делает ее более качественной), а также автоматически отслеживать ошибки синтаксиса.

Среди представленных на рынке IDE ни одна не стоит рядом с VS Code по эффективности.

Логика файловой структуры проекта «SportsMap» такова:

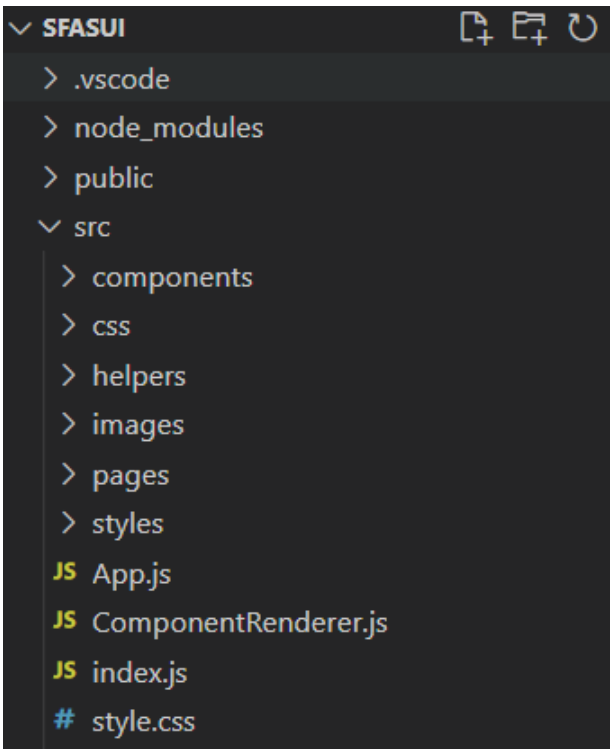

Рисунок 14 Файловая структура

Основа проекта – в файле App.js. Именно там происходит исходный запуск приложения и его компонентов. Компоненты расположены в папке components

Стоит подробнее рассмотреть понятие компонентов в ReactJS. Компоненты содержат составные части и модули страниц, из них генерируется целиком страница и все методы взаимодействия. Они помогают разделить интерфейс на независимые друг от друга элементы, которые в дальнейшем можно переиспользовать. На самом деле компоненты концептуально схожи с функциями в JavaScript – они принимают данные и возвращают React-элементы.

Компоненты могут ссылаться на другие, что как раз и позволяет использовать одну и ту же абстракцию для любого уровня детализации.

Кнопка, форма, диалоговое окно, экран: в приложениях React все они обычно являются компонентами.

Что наиболее удобно в концепции компонентов, так это как раз независимость и переиспользование. Это позволяет быстро изменить, например, сразу все кнопки в приложении с помощью изменения лишь одного объекта, при этом ничего не сломав в других местах. В целом, эти свойства соответствуют некоторым принципам SOLID, что также немаловажно для качества разрабатываемого продукта.

Редкие общие CSS стили (вроде шрифтов, например) установлены в файле style.css. Но в основном дополнительные стили компонентов, если это было необходимо, были прописаны в каждом отдельном Java Script файле с помощью библиотеки twin.macro. Это библиотека, которая преобразует описания TailwindCSS в объекты CSS. Классическим примером является следующий код:

```
import 'twin.macro';

export default function App() {
  return (
    <h1 tw="text-2xl text-blue-500 font-bold">Hello world</h1>
  );
}
```

Рисунок 15 Пример использования twin.macro

Таким образом устанавливаются стили для заголовка h1. Преимущество библиотеки состоит в том, что она значительно сокращает количество кода для стилей, делает его более читабельным (хотя какое-то время разработчику все же понадобится, чтобы привыкнуть), а также в ней присутствует отличное логирование для отладки синтаксиса в процессе разработки.

7.3 Результаты

В результате работы над клиентской составляющей веб-приложения было изучено множество инструментов и программного обеспечения, которые позволили создать визуальную часть приложения, готовую к использованию и взаимодействующую с серверной частью.

Пользовательский интерфейс практически полностью соответствует созданным ранее дизайн-проекту, прототипу, карте сайта, повторяет всю логику, стили и проч.

Код является гибким, удобным для быстрых изменений компонентов, дающим возможность переиспользования блоков кода, а также хорошо структурированным.

Заключение

В результате данной выпускной квалификационной работы было разработано веб-приложение «SportsMap» – система учета и анализа объектов реестра физической культуры и спорта Красногвардейского района города Санкт-Петербурга.

Разработанный функционал приложения, в полной мере отвечает поставленным перед ним требованиям и на данный момент оно успешно внедряется в работе администрации Красногвардейского района г. Санкт-Петербурга.

Гибкая архитектура дает возможности для расширения функционала приложения, что подразумевается на следующих этапах разработки. Например, для расширения представленной территории с границ района до всего города.

Цель и поставленные задачи выполнены полностью. Прикладным результатом ВКР является веб-приложение с внедренной системой репрезентации геоданных в виде карты спортивных объектов. Имеется возможность поиска и фильтрации объектов в списке. Кроме того, доступна авторизация для администратора, способного совершать добавление, удаление, скрытие, изменение объектов и выгрузку/загрузку данных в таблице формата .xlsx.

Приложение «SportsMap» определенно будет полезно как Красногвардейскому району, так и городу в целом. Ведь в перспективе планируется включение всех районов Санкт-Петербурга в систему – это уже обсуждается на уровне администрации города.

Список используемых источников

1. ISO/IEC 15288:2008, ISO/IEC 29148:2011 Systems and software engineering - Content of life-cycle information products (documentation)

2. Карл И. Вигерс. Разработка требований к программному обеспечению. — Русская редакция, 2004. — ISBN 5-7502-0240-2.

3. ГОСТ 34.602-89 Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы

4. Список участников конкурса на соискание премий Правительства Санкт-Петербурга за выполнение дипломных проектов по заданию исполнительных органов государственной власти Санкт-Петербурга в 2021/2022 учебном году, 2021 г., http://knvsh.gov.spb.ru/closedcontests/view/274/

5. Стандарты и шаблоны для ТЗ на разработку ПО, 2017г., https://habr.com/ru/post/328822/

6. Аникеева Ольга Сергеевна Публикация карт в сети Интернет: эволюция картографии // Наука. Инновации. Технологии. 2015. №2. URL: https://cyberleninka.ru/article/n/publikatsiya-kart-v-seti-internet-evolyutsiya-kartografii.

7. Сбитнева Оксана Анатольевна РОЛЬ СПОРТИВНЫХ СООРУЖЕНИЙ В ФИЗКУЛЬТУРНО-СПОРТИВНОЙ ДЕЯТЕЛЬНОСТИ СТУДЕНТОВ // Эпоха науки. 2021. №25. URL: https://cyberleninka.ru/article/n/rol-sportivnyh-sooruzheniy-v-fizkulturno-sportivnoy-deyatelnosti-studentov (дата обращения: 15.12.2021).

8. Ротанова Ирина Николаевна, Репин Никита Владимирович Подходы к созданию веб-атласа Алтае-Саянского экорегиона // Известия АлтГУ. 2014. №3 (83). URL: https://cyberleninka.ru/article/n/podhody-k-sozdaniyu-veb-atlasa-altae-sayanskogo-ekoregiona (дата обращения: 15.12.2021).

9. Танатканова Асель Кайратовна, Жамбаева Анар Куанышбековна Построение клиент-серверных приложений // Наука и образование сегодня. 2019. №6-2 (41). URL: https://cyberleninka.ru/article/n/postroenie-klient-servernyh-prilozheniy.

10. Использование диаграммы вариантов использования UML при проектировании программного обеспечения, https://habr.com/ru/post/566218/ (2021 г.)

11. Р. Мартин Принципы, паттерны и методики гибкой разработки на языке C#

12. https://habr.com/ru/company/arcadia/blog/498032/ Расчет формул из Excel на C#

13. Bachelorarbeit Eric Wohlgethan Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue.js

14. Калюжный Е. Р., Ксендзовский И. Д., Зариковская Н. В. Технологии разработки серверной части приложений и систем // Colloquium-journal. 2020. №16 (68). URL: https://cyberleninka.ru/article/n/tehnologii-razrabotki-servernoy-chasti-prilozheniy-i-sistem.

15. Кобылкин Д.С., Юсупова О.В. ПЕРСПЕКТИВА ПРИМЕНЕНИЯ СРЕДСТВ ENTITY FRAMEWORK ПРИ РАЗРАБОТКЕ СИСТЕМ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ ДАННЫХ НА ЯЗЫКЕ C# // Символ науки. 2021. №2. URL: https://cyberleninka.ru/article/n/perspektiva-primeneniya-sredstv-entity-framework-pri-razrabotke-sistem-raspredelennoyobrabotki-dannyh-na-yazyke-c.

16. Макаров Олег Сергеевич, Щенникова Елена Владимировна СРАВНИТЕЛЬНЫЙ АНАЛИЗ ТЕХНОЛОГИИ ASP.NET CORE // E-Scio. 2020. №7 (46). URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-tehnologii-asp-net-core.

17. Езжев Никита Андреевич РАЗРАБОТКА КОРПОРАТИВНОГО НОВОСТНОГО WEB-СЕРВИСА С ИСПОЛЬЗОВАНИЕМ .NET CORE И ANGULAR // Новые импульсы развития: вопросы научных исследований. 2020. №6-1. URL: https://cyberleninka.ru/article/n/razrabotka-korporativnogo-novostnogo-web-servisa-s-ispolzovaniem-net-core-i-angular.

18. Байнов Артем Маратович, Кривоногова Анастасия Евгеньевна, Николаев Алексей Сергеевич, Богомолова Ольга Игоревна Обзор современных фреймворков и инструметов, используемых для разработки web-приложений // Наука без границ. 2020. №1 (41). URL: https://cyberleninka.ru/article/n/obzor-sovremennyh-freymvorkov-i-instrumetov-ispolzuemyh-dlya-razrabotki-web-prilozheniy.

19. Горбачев А.А., Горбачева Е.С. Сравнение классического процесса реализации веб-приложений и подхода с использованием библиотеки React // Молодой исследователь Дона. 2020. №1 (22). URL: https://cyberleninka.ru/article/n/sravnenie-klassicheskogo-protsessa-realizatsii-veb-prilozheniy-i-podhoda-s-ispolzovaniem-biblioteki-react.

Приложения

Приложение №1. Общие требования к информационной системе и обязательства исполнителя.

Разработка автоматизированной системы анализа и учета реестра объектов физической культуры и спорта для Администрации Красногвардейского района.

Общие требования к информационной системе и обязательства исполнителя.

25 ноября 2021 г.

Разработка автоматизированной системы анализа и учета реестра объектов физической культуры и спорта для Администрации Красногвардейского района.

Общие требования к информационной системе и обязательства исполнителя.

Далее в документе будут использованы обозначения:

- Исполнитель – Рогова Елена Александровна, студентка Санкт-Петербургского государственного университета.
- Заказчик - Сектор физической культуры и спорта Администрации Красногвардейского района.

Проект представляет собой реализацию системы анализа и учета реестра объектов физической культуры и спорта.

Для создания данной системы необходимо реализовать:

- Дизайн UI
- Прототип системы
- Имплементация бизнес-логики
- Документация как по коду и установке приложения на сервере, так и по клиентскому использованию

Имплементация бизнес-логики подразумевает наличие следующих составляющих:

- База данных с существующими объектами спорта
- Администраторская панель для внесения изменений в базу
- Карта с объектами и информацией о них
- Поиск объектов по фильтрам
- Логика работы с упомянутыми составляющими

На данном этапе подразумевается создание четырех основных логических блоков:

1. Карта
2. Поиск
3. Администраторская панель
4. Прочее (контакты, информация об организации и т.д.)

Карта должна представлять собой карту Красногвардейского района с выделенными на ней объектами спорта. На карте так же должны быть отмечены основные объекты для ориентации на местности – улицы, дома, значимые строения. Каждый объект спорта должен иметь ссылку на соответствующую страницу с подробным описанием.

Описание объекта должно включать в себя всю предоставленную Заказчиком информацию: местоположение, фотографии, тип, доступность для людей с

ограниченными возможностями, площадь, тип собственности, технические особенности и т.д.

Поиск по фильтрам должен осуществляться на отдельной странице со списком всех существующих в базе объектов. Фильтрация должна быть реализована по основным характеристикам объекта: доступность, тип и т.д.

Фильтрация на карте на данном этапе разработки не предусмотрена.

Приложение должно быть разработано на платформе .NET. Сайт должен корректно отображаться в последних десктопных версиях браузеров: Opera, Chrome (и все браузеры на его основе), Mozilla. Мобильная версия сайта не предусмотрена на данном этапе разработки.

Веб-приложение должно позволять пользователям:

- осуществлять навигацию по сайту (переход между страницами);
- просматривать карту с объектами спорта Красногвардейского района
- производить поиск по объектам спорта Красногвардейского района
- просматривать страницы с описанием конкретных объектов спорта
- скачивать (при наличии необходимых прав доступа) различного рода документы и файлы;

Администраторская панель должна позволять:

- выполнять вход на сайт как администратор для возможности добавления/редактирования содержимого сайта.
- обновлять данные в реестре объектов

Исполнитель обязуется согласовывать основные процессы разработки и используемые сторонние сервисы в виде ТЗ, а также отчитываться о стадиях разработки.

По окончании работ Исполнитель обязан предоставить веб-приложение, кодовую базу, базу данных, исходные графические материалы по дизайну, документацию.

Исполнитель:

_____ (__Рогова Е. А.__)

(Подпись)

25 ноября 2021 г

Приложение №2. Техническое задание

Техническое задание
на разработку автоматизированной системы анализа и учета реестра объектов физической культуры и спорта

Санкт-Петербург

2021 г.

1. Общие положения

1.1 Полное наименование системы и ее условное обозначение

Полное наименование – Система анализа и учета реестра объектов физической культуры и спорта.

Сокращенное наименование – СУОС (Система Учёта Объектов Спорта).

Сокращенное наименование на английском языке – SFAS (Sports Facilities Accounting System)

Тип системы – веб-приложение.

1.2 Наименование организации-заказчика

Администрация Красногвардейского района Санкт-Петербурга

Адрес: 195027, Санкт-Петербург, Среднеохтинский пр., 50

Телефон: +7 (812) 227-43-64

Факс: +7 (812) 576-87-63

1.3 Исполнитель

Исполнитель определен в рамках конкурсной процедуры.

Рогова Елена Александровна, студентка Санкт-Петербургского государственного университета.

Телефон: +7 (920) 230-27-64
Электронная почта: st069125@student.spbu.ru, lalagvanan@gmail.com

1.4 Определения, обозначения и сокращения

Используемые в документе термины и определения приведены в таблице.

Таблица 1. Определения, обозначения и сокращения

Термин/сокращение	Определение / пояснение
Система или СУОС	Система анализа и учета реестра объектов физической культуры и спорта.
ОС	Операционная система
ИС	Информационная система
СО	Спортивный объект
БД	База данных
ВИ	Вариант использования системы
Эндпоинт	Это само обращение к маршруту отдельным HTTP методом. Эндпоинт выполняют конкретную задачу, принимают параметры и возвращают данные Клиенту.
Контент-администратор	Пользователь, имеющий право изменять карточки СО.

1.5 Назначение документа

В настоящем документе приводится полный набор требований к Системе, необходимых для реализации.

Подпись Заказчика и Исполнителя на настоящем документе подтверждает их согласие с нижеследующими фактами и условиями:

- При реализации необходимо выполнить работы в объёме, указанном в настоящем Техническом Задании.
- Все неоднозначности, выявленные в настоящем Техническом задании после его подписания, подлежат двухстороннему согласованию между Сторонами.

1.6 Плановые сроки начала и окончания работ

Начало работ производится непосредственно с момента подписания договора. Плановая дата окончания работ – 1.06.2022.

2. Цели и задачи системы

2.1 Цели системы

Целью разработки и внедрения Системы является повышение эффективности работы Сектора спорта Администрации Красногвардейского района за счет сокращения сроков поиска интересующего СО и повышения прозрачности процесса; обеспечения достоверности, целостности и актуальности информации о СО;

2.2 Задачи системы

Система должна решать следующие задачи:

- Обеспечить регистрацию и хранение всей доступной информации о СО.
- Позволить каждому жителю Красногвардейского района ознакомиться с реестром СО и найти подходящий, основываясь на различных критериях.
- Быть наиболее доступной для инвалидов, предоставлять информацию о доступных для них СО (в том числе о том, находится ли СО в перечне социально значимых).
- Позволить проводить анализ ситуации с СО в Красногвардейском районе с помощью карты (плотность СО, их доступность и проч.)
- Улучшить доступ населения к информации о СО, тем самым повысив мотивацию людей вести здоровый образ жизни, что является одной из целей Сектора спорта Администрации Красногвардейского района.

3. Требования к системе

3.1 Нефункциональные требования

3.1.1 Требования к совместимости

Система должна предоставлять пользователю возможность полноценной работы с использованием технологии WEB-доступа (работа с СУОС, используя возможности интернет-браузеров: Google Chrome; Яндекс.Браузер; Microsft Edge, Mozilla Firefox, Opera). Система не должна конфликтовать с известными средствами организации шифрования и частного доступа к данным. СУОС должна обеспечивать возможность подключаться к интерфейсу программы через WEB приложение (браузер) по протоколу http в пределах локально вычислительной сети и https, как в пределах локально вычислительной сети, так и вне ее, без участия дополнительных плагинов и модулей.

Мобильная веб-версия СУОС на данном этапе разработки не предусмотрена. СУОС должна отображаться в браузерах на мобильных устройствах так же, как и на ПК (необязательно адаптивно).

СУОС должна функционировать на базе свободно распространяемых компонентов с открытым исходным кодом. Использование иных компонентов должно быть согласовано с Заказчиком отдельно.

3.1.2 Требования к безопасности

СУОС должна обеспечивать сохранность и защиту персональных данных, в соответствии с Российским законодательством.

СУОС должна обеспечивать идентификацию каждого контент-администратора в системе, возможность определения авторства операций в Системе и обеспечивать отсутствие неавторизованных операций изменения данных, протоколировать в журнале (логе) процедуры аутентификации и действий администраторов в системе.

Система должна обеспечивать администраторам возможность добавления новых СО и информации о них в БД, добавления СО на карту.

Во время работы пользователям СУОС должны быть доступны только необходимые функции согласно их ролям.

В системе должны быть предусмотрены возможности ее последующей модернизации. Модернизация системы может быть обусловлена необходимостью перехода к использованию новых версий общесистемного программного обеспечения, новой версии прикладного программного обеспечения, миграции с одной системной платформы на другую в соответствии с изменившимися или возросшими потребностями пользователей.

СУОС должна предусматривать развитие по трем направлениям:

1. Территориальное масштабирование, т.е. подключение к системе новых районов Санкт-Петербурга, внедрение системы в прочих структурных подразделениях Администрации города Санкт-Петербурга. Должна быть возможность перехода СУОС на распределенный вариант работы системы.
2. Функциональное масштабирование, т.е. постепенное наращивание функциональных возможностей.
3. Наращивание аппаратных ресурсов, обеспечивающих функционирование системы.

3.1.3 Требования к надежности

Система должна допускать ежедневное круглосуточное функционирование. Допускается временная приостановка работы системы для проведения профилактических работ программно-аппаратного обеспечения сервера, на котором располагается система.

Необходимым условием функционирования СУОС является условие функционирования аппаратной части и сервера, на котором размещено приложение.

Система в целом должна сохранять работоспособность при некорректных действиях конечных пользователей.

Система должна обеспечивать восстановление работоспособности при появлении сбоев, аварий и отказов, возникающих на сервере и сетевом аппаратном обеспечении.

3.1.4 Требования к эргономике и технической эстетике

Интерфейс Системы должен быть прост, нагляден, интуитивно понятен и легок в освоении.

Система должна позволять просматривать всю основную информацию (карта Красногвардейского района с расположенными СО, атрибуты карточек СО) в одном окне.

Интерфейс СУОС должен отвечать следующим требованиям:

- Единый унифицированный интерфейс, реализованный на русском языке
- Однозначность в наименовании пунктов меню.

Сигнализацию об ошибках системы или выполнении ошибочных действий пользователем в виде индикаций на экране с информацией об ошибке и/или подсказкой о дальнейших действиях на русском языке;

Наличие вспомогательной индикации при выполнении длительных процессов.

Необходима реализация режима для слабовидящих.

Цветовое решение интерфейса должно быть выдержано в спокойных тонах, не вызывающих утомление зрения.

3.1.5 Требования к эксплуатации, техническому обслуживанию системы

Техническая и физическая защита аппаратных компонентов системы, носителей данных, бесперебойное энергоснабжение, резервирование ресурсов, текущее обслуживание реализуется техническими и организационными средствами, предусмотренными в инфраструктуре Заказчика.

3.1.6 Требования к производительности системы

Время реакции СУОС на открытие, сохранение, закрытие, вставка (любое действие) любого объекта системы без учета вложения не должно превышать 5 секунд.

3.1.7 Требования к лингвистическому обеспечению системы

Прикладное программное обеспечение Системы для организации взаимодействия с пользователем должно использовать русский язык.

Вся документация к СУОС должна быть разработана на русском языке.

3.2 Функциональные требования

3.2.1 Диаграммы Вариантов Использования

На Диаграммах представлены основные Варианты Использования Системы.

Диаграмма 1. Действующие лица.

На данной диаграмме представлена иерархия всех Пользователей Системы. Связь обобщения следует читать следующим образом: Пользователь наследует все поведение своего родителя, а также имеет свое поведение в Системе. Например, Администратор может делать все то, что

делает Незарегистрированный пользователь, но может еще и добавлять в Систему СО.

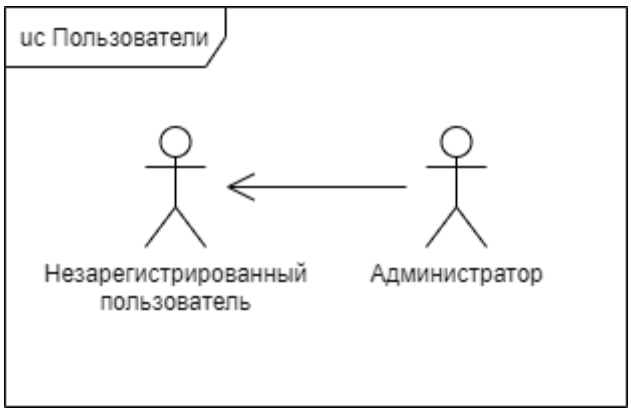

Рис. 16. Действующие лица

Диаграмма 2. ВИ входа и Управления карточками СО

Контент-администратор должен иметь ссылку на страницу аутентификации. На эту страницу нельзя попасть с остальных страниц веб-сайта. Аутентификация считается недействительной при закрытии окна с веб-сайтом.

Администратор входит в учётную запись и теперь имеет доступ к тем же страницам, что и незарегистрированный пользователь, но с расширенным функционалом. Основные действия администратора – Добавление СО, Удаление СО, Редактирование СО на странице с подробной информацией о СО.

Если при входе выясняется, что логин или пароль неверны, то пользователь получает соответствующее предупреждение.

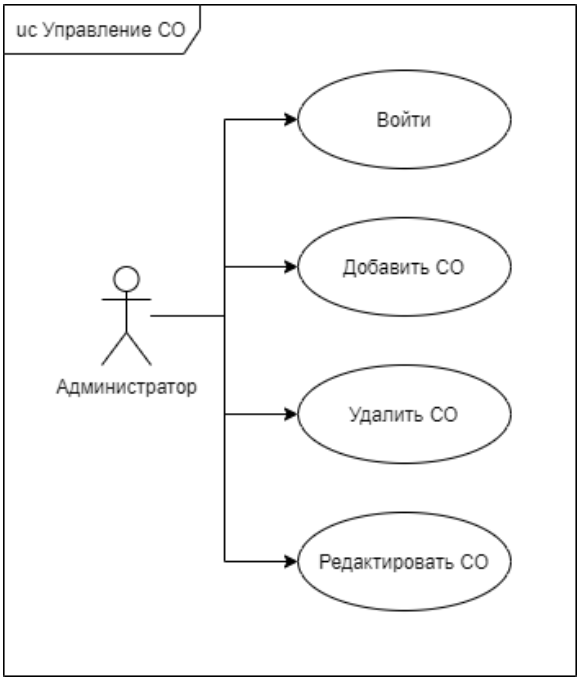

Рис. 17. ВИ входа и Управления карточками СО

Диаграмма 3. ВИ Просмотра информации о СО

Пользователь заходит на веб-сайт СУОС. В качестве основного его ВИ – Переход на страницу с картой сайта, где пользователь может ознакомиться с местоположением СО, а также узнать подробную информацию о каждом из них, перейдя на отдельную страницу СО.

В качестве альтернативных ВИ – использование поиска СО с фильтрацией и дальнейший переход на отдельную страницу СО, а также переход на страницу с информацией об организации и возможность подачи заявки на добавления СО в реестр.

При подаче заявки пользователь обязан указать свой e-mail для связи, также обязателен адрес и название СО. Остальная информация опциональна.

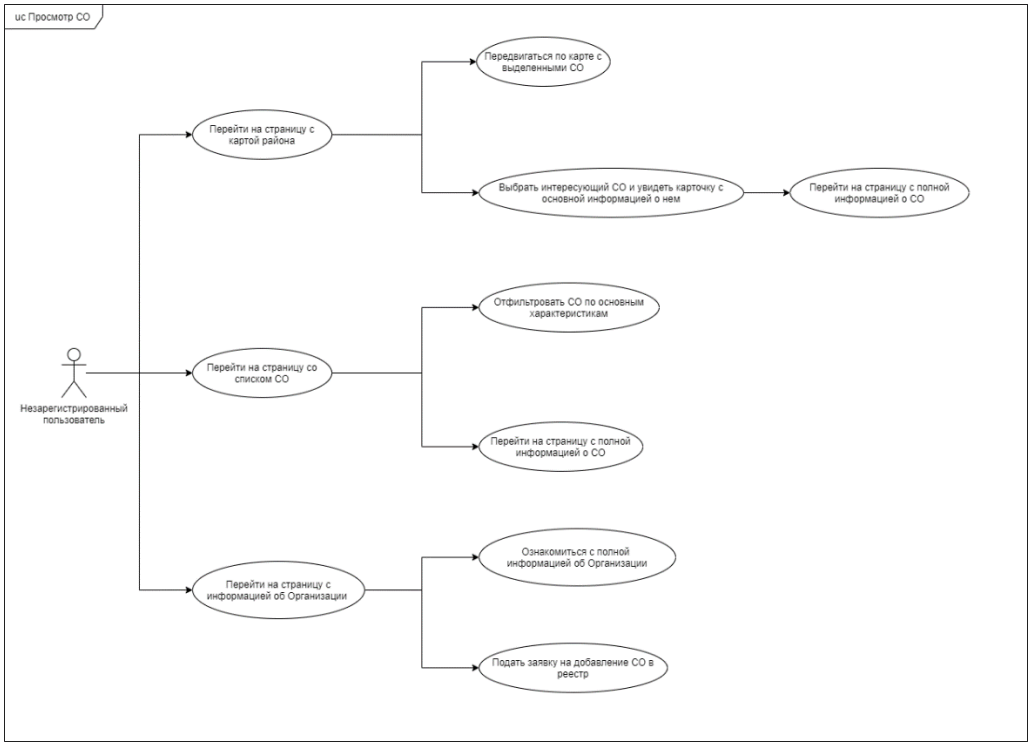

Рис. 18. ВИ Просмотра информации о СО

3.2.2 Логирование операций в Системе

При возникновении ошибок в БД должна добавляться следующая информация:

- Идентификатор пользователя
- Дата и время возникновения ошибки
- Сообщение ошибки

При удалении сущностей в Системе физически данные по ним не должны удаляться, а должна сущность помечаться, как удаленная. Должен сохраняться идентификатор пользователя, удалившего сущность.

Макеты интерфейса и дизайн-референсы

Ниже представлена карта сайта, макеты интерфейса для каждой страницы системы, а также дизайн-референсы, на основе которых будет

создан оригинальный дизайн.

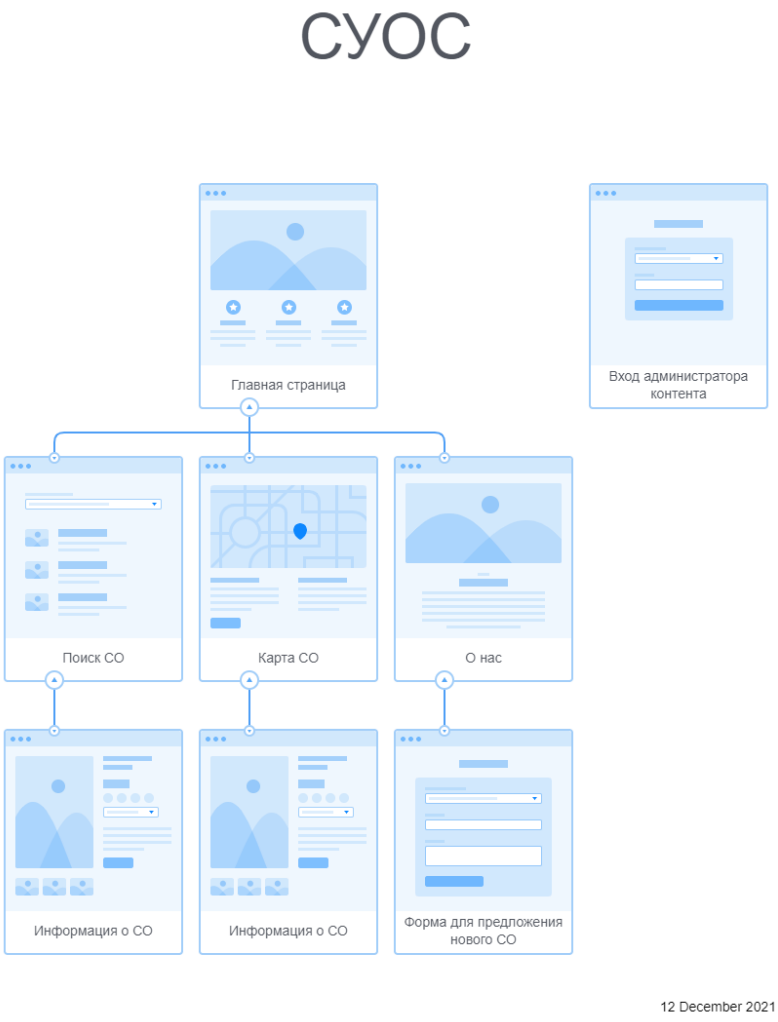

Рис. 19. Карта сайта

(См. актуальную версию по адресу: https://app.flowmapp.com/share/ef8666e21c8beaac7c34e45fcb833bb4/sitemap/)

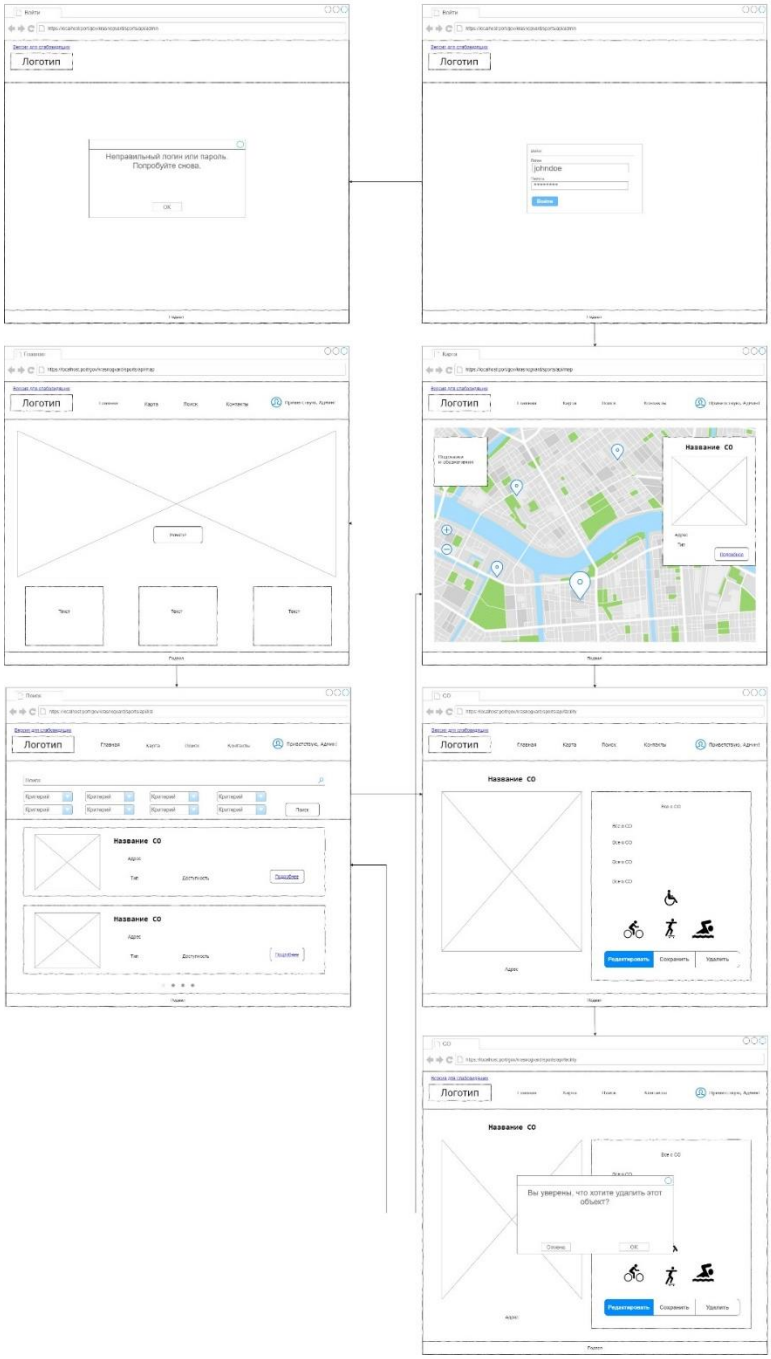

Рис.20. Макеты интерфейса. Роль – Администратор.

(См. актуальную версию по адресу: https://drive.google.com/file/d/1DqG-JxtUijV7pJSK6kTCg7sy0vGQ988V/view?usp=sharing)

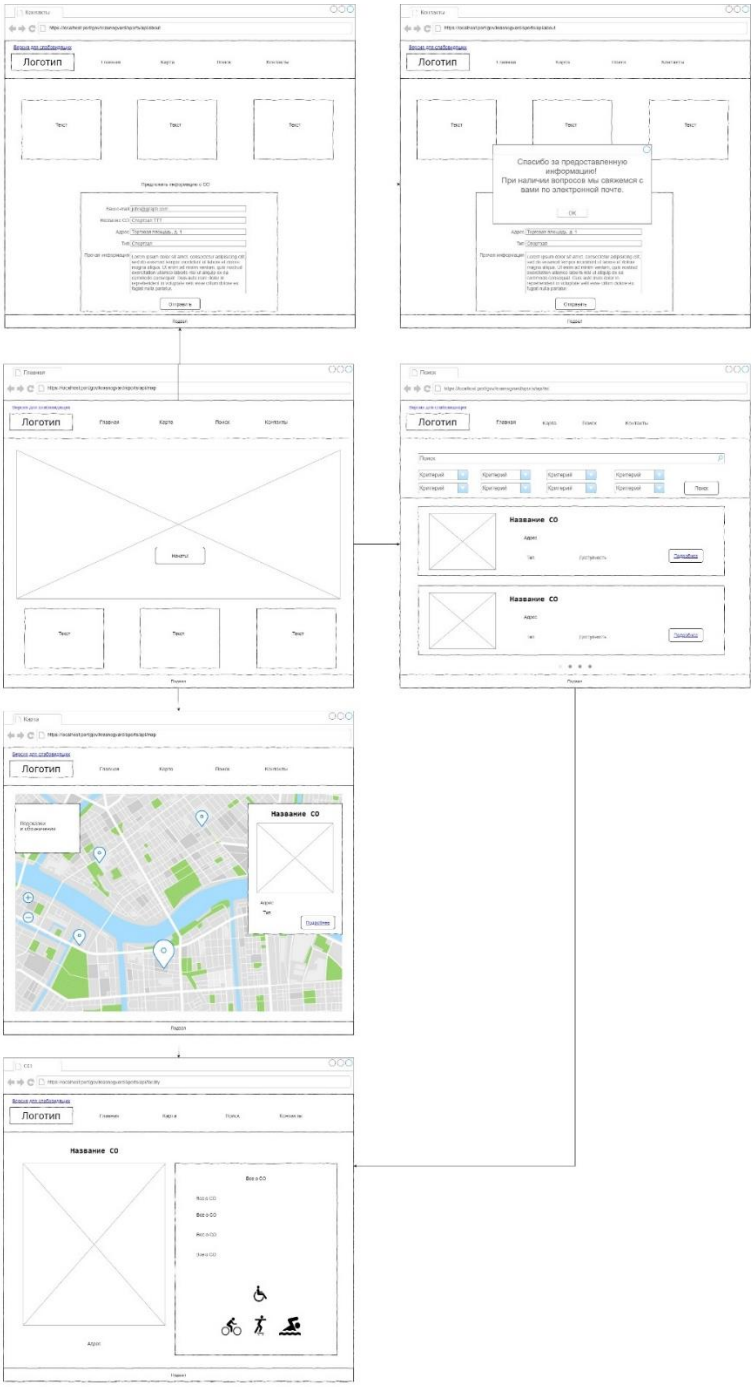

Рис.6. Макеты интерфейса. Роль – незарегистрированный пользователь.

(См. актуальную версию по адресу: https://drive.google.com/file/d/1DqG-JxtUijV7pJSK6kTCg7sy0vGQ988V/view?usp=sharing)

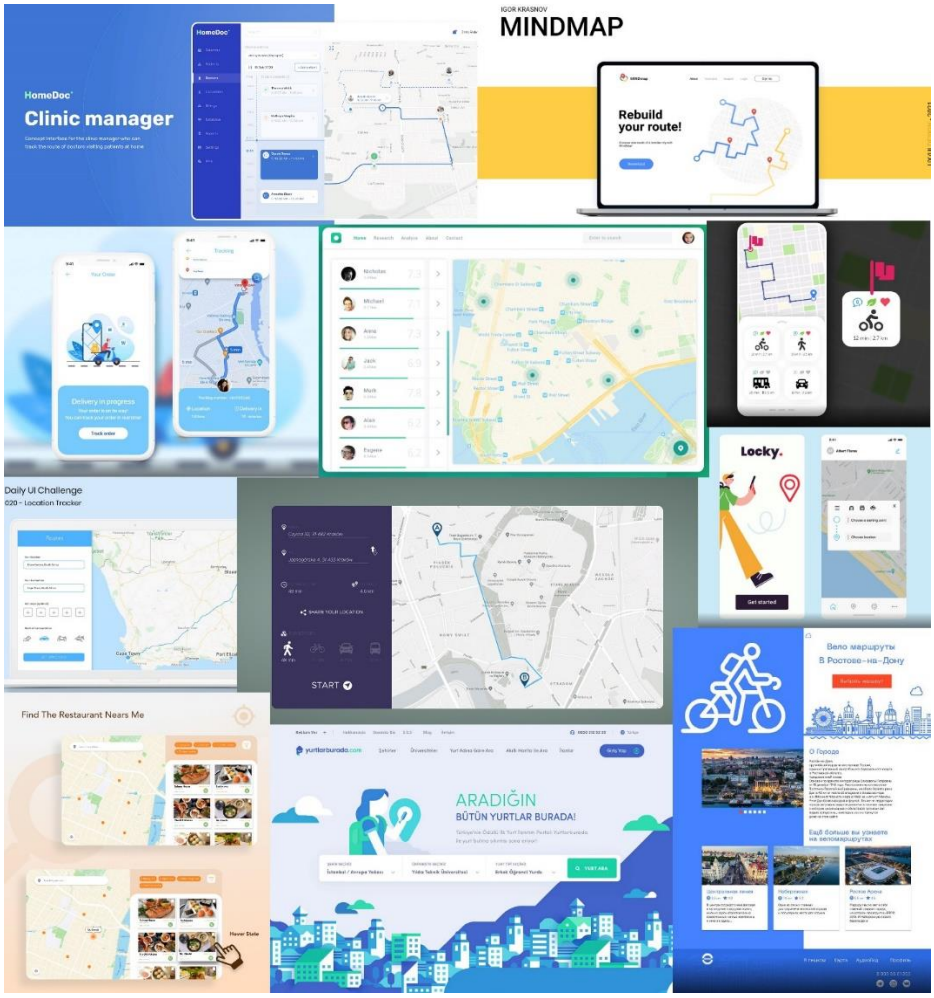

Рис. 7. Дизайн-референсы
(Источник:https://www.behance.net/collection/189938721/maps?tracking_source=undefined)

3.4 Модель данных

На Диаграммах показаны основные сущности Системы и их взаимосвязи между собой. Имеется большая вероятность изменения структуры (по согласованию с Заказчиком).

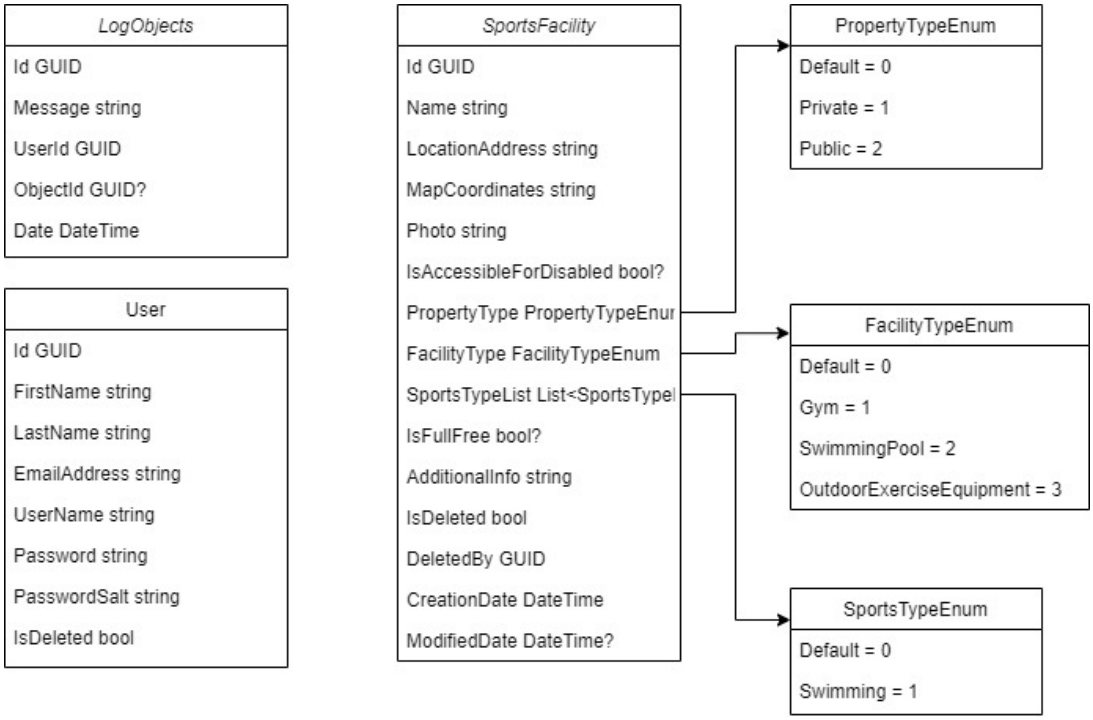

Рис. 8. Диаграмма классов СУОС

LogObjects – класс объектов логирования (в соответствии с п. 3.2.2).

User – класс пользователя (контент-администратора).

SportsFacility – класс СО, использует enum-классы для различных характеристик.

3.5 Описание архитектуры

Для создания СУОС была выбрана монолитная архитектура клиент-серверного приложения. Монолитная архитектура подразумевает хранение бизнес-логики на одном сервере и в одной кодовой базе.

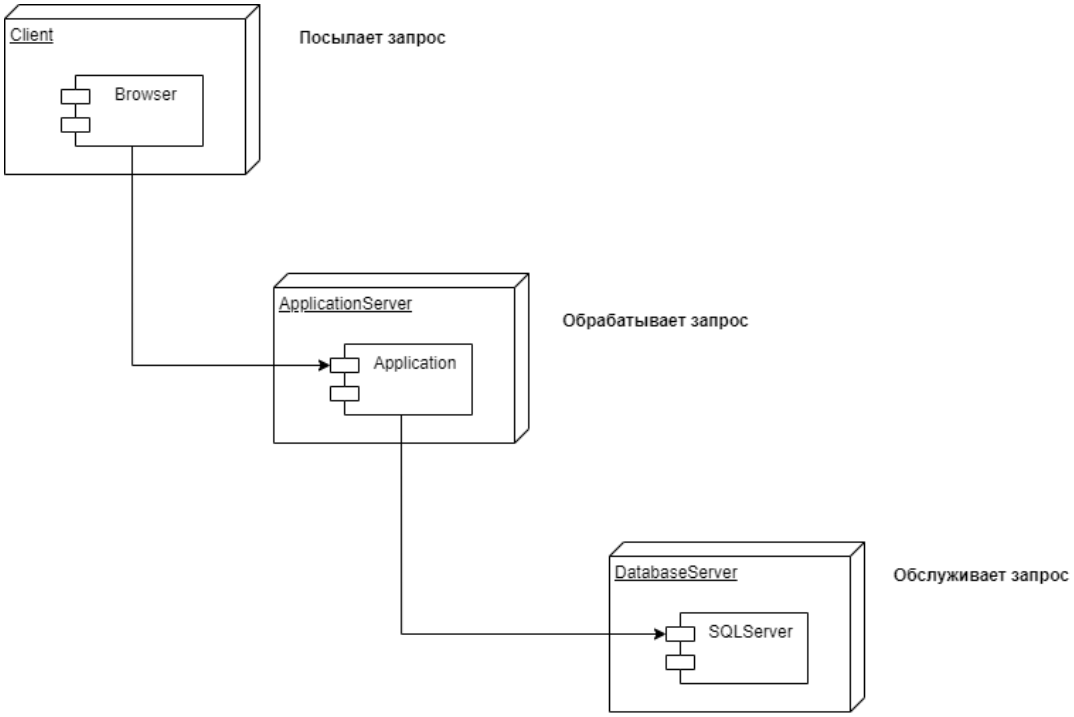

Рис. 9 Схема клиент-серверной архиткетуры

Клиент с помощью браузера посылает через HTTP/HTTPS протокол REST API запрос (Подробнее можно прочесть здесь: https://habr.com/ru/post/483202/). Далее сервер бизнес-логики обрабатывает запрос и посылает его на сервер БД, откуда возвращается ответ и далее попадает обратно к клиенту в браузер.

Клиентская часть предполагает реализацию с помощью языка JavaScript, слой бизнес-логики - с помощью платформы .NET Core, а БД предполагается на Microsoft SQL Server.

Прочие фреймворки и технологические инструменты будут определены в отдельном документе.

4. Схема работ по созданию системы

В рамках проекта предусматривается следующая схема работы:

- Подрядчик выполняет весь объём работ в рамках своих функциональных задач.

- Передача работы Заказчику может выполняться поэтапно, при этом завершение всего объема работ оформляется актом передачи разработанной системы, а также актом передачи разработанной документации.

4.1 Требования к приемке-сдаче проекта

Исполнитель должен предоставить следующий комплект поставки при сдаче проекта:

- Техническое задание
- Исходный код Системы
- Исполняемые модули Системы
- Тестовые сценарии
- Пользовательскую документацию

Приемо-сдаточные испытания должны проводиться по каждому этапу отдельно на сервере Заказчика в оговоренные отдельно сроки.

4.2 Перспективы развития

Предлагаются следующие схемы развития веб-приложения СУОС:

- Внедрение поиска с фильтрацией на карте.
- Добавление в Систему других районов города Санкт-Петербург.
- Построение маршрута от места нахождения пользователя до интересующего СО.
- Реализация мобильной версии приложения.

- Возможность регистрироваться обычным пользователям, чтобы иметь возможность добавлять СО в избранное.
- Возможность загружать данные о множестве СО в универсальном текстовом виде, чтобы убрать необходимость добавлять данные вручную.
- Система оценок и комментариев к СО пользователями.
- Система записи на СО с такой возможностью внутри СУОС.
- Возможность кооперироваться с другими пользователями системы, чтобы посещать СО совместно (с помощью чатов и проч.).
- Расширение ролей в системе (администратор, контент-администратор, зарегистрированный пользователь