МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка веб-приложения «Виртуальная вселенная ОАО РЖД»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-580.ВКР

Научный руководитель,
доцент кафедры СП, к.т.н.
_____ Н.Ю. Долганина

Автор работы,
студент группы КЭ-404
_____ М.В. Рычков

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-404
Рычкову Максиму Владимировичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12) Разработка веб-приложения «Виртуальная вселенная ОАО РЖД».
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. ASP.NET 6.0. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/aspnet/core/release-notes/aspnetcore-6.0?view=aspnetcore-6.0 (дата обращения: 02.02.2024 г.).
3.2. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2021. – 352 с.
3.3. Cantor D, Jones B. WebGL Beginner’s Guide. – Packt Publishing, 2012. – 376 с.

4. **Перечень подлежащих разработке вопросов**

4.1. Проанализировать предметную область.
4.2. Спроектировать веб-приложение.
4.3. Реализовать веб-приложение.
4.4. Протестировать разработанное веб-приложение.

5. **Дата выдачи задания:** 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.т.н. Н.Ю. Долганина

Задание принял к исполнению М.В. Рычков

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 4
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6
1.1. Обзор аналогичных проектов 6
1.2. Обзор технологий для реализации проекта 8
2. ПРОЕКТИРОВАНИЕ 14
2.1. Анализ требований 14
2.2. Диаграмма вариантов использования 15
2.3. Хранилище данных 16
2.4. Архитектура веб-приложения 18
3. РЕАЛИЗАЦИЯ 20
3.1. Реализация серверной части 20
3.2. Реализация клиентской части 22
4. ТЕСТИРОВАНИЕ 33
ЗАКЛЮЧЕНИЕ 35
ЛИТЕРАТУРА 36

ВВЕДЕНИЕ

Актуальность

На сегодняшний день на предприятиях ОАО «РЖД» отсутствует единая карта предприятия, отражающая информацию об объекте и сотрудниках, которые в нем работают. Существуют лишь бумажные поэтажные планы, которые не позволяют оперативно получить информацию. Поиск информации ведется в нескольких информационных системах, часть из которых доступна только отделам, курирующим данную задачу, что затрудняет процесс управления недвижимым имуществом ОАО «РЖД» и определение параметров площадей, необходимых и достаточных для размещения офисных работников ОАО «РЖД».

Отсутствие автоматизации приводит к нерациональному использованию ресурсов ОАО «РЖД». Информация о номере кабинета, модели оборудования доступна через сайт, сопровождаемый одной командой, и доступна только ответственным лицам на предприятии. В то же время, информация о мебели хранится в другой системе, доступ так же имеют ограниченное количество лиц. Информация об условиях размещения работника (площадь кабинета, количество соседей-коллег по кабинету) зачастую предоставляется по запросу путем соотнесения данных из разных источников.

Такая разрозненность информации приводит к затруднениям в ее оперативном получении, анализе и принятии управленческих решений. Введение единого веб-приложения позволяет систематизировать информацию об объекте, снизить затраты на инвентаризацию, повысить доступность и информативность. Это также упростит процесс размещения сотрудников и использование недвижимого имущества ОАО «РЖД».

Таким образом, внедрение веб-приложения обеспечит целостность управления ресурсами ОАО «РЖД», увеличивая общую эффективность.

Постановка задачи

Целью выпускной квалификационной работы является разработка веб-приложения «Виртуальная вселенная ОАО РЖД». Для достижения поставленной цели необходимо решить следующие задачи:

1) проанализировать предметную область;
2) спроектировать веб-приложение;
3) реализовать веб-приложение;
4) протестировать разработанное веб-приложение.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 37 страниц, объем списка литературы – 15 источников.

В первой главе описывается анализ предметной области, обзор существующих технологий и инструментов для разработки 2D и 3D-приложений в веб-окружении.

Вторая глава посвящена проектированию разрабатываемой системы. Определяются функциональные и нефункциональные требования к системе.

В третьей главе описывается реализация и листинги.

Четвертая глаза посвящена тестированию веб-приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Обзор аналогичных проектов

В качестве примеров существующих аналогичных проектов, которые предоставляют возможность построения плана здания, были выбраны три сайта, описание которых приведено ниже.

Веб-сайт «Инженерный план»

«Инженерный план» – инструмент для проектирования зданий и помещений, позволяющий создавать стены с произвольной толщиной, устанавливать двери и окна, а также устанавливать крыши различных типов. Поддерживается работа с несколькими этажами и переключение между ними. Переключение между 2D и 3D-планами упрощает визуализацию и восприятие.

В «Инженерном плане» доступен каталог предметов, таких как мебель и бытовая техника, которые можно размещать на плане этажа. Однако «Инженерный план» поддерживает максимум 4 этажа, что ограничивает возможности проектирования многоэтажных зданий. На рисунке 1 представлен интерфейс веб-сайта «Инженерный план».

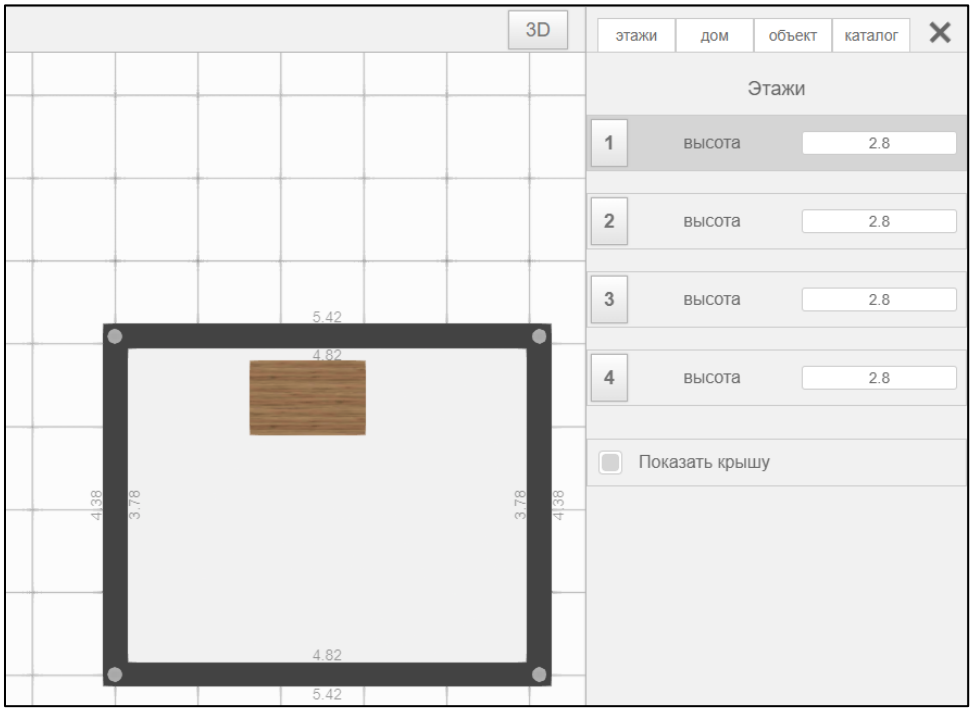

Рисунок 1 – Веб-сайт «Инженерный план»

Веб-сайт «Floorplancreator»

«Floorplancreator» – инструмент, который позволяет проектировать план этажа. Пользователи могут строить стены, добавлять лестницы, создавать новые этажи и переключаться между ними. Переключение между 2D и 3D-этажами обеспечивает удобную визуализацию плана этажа. Можно детально изменить внешний вид стены. Минусом веб-сайта является отсутствие возможности переключить язык интерфейса, отсутствие возможности добавлять предметы на план, окна и двери, что ограничивает его функциональность в проектировании интерьера. Интерфейс веб-сайта «Floorplancreator» представлен на рисунке 2.

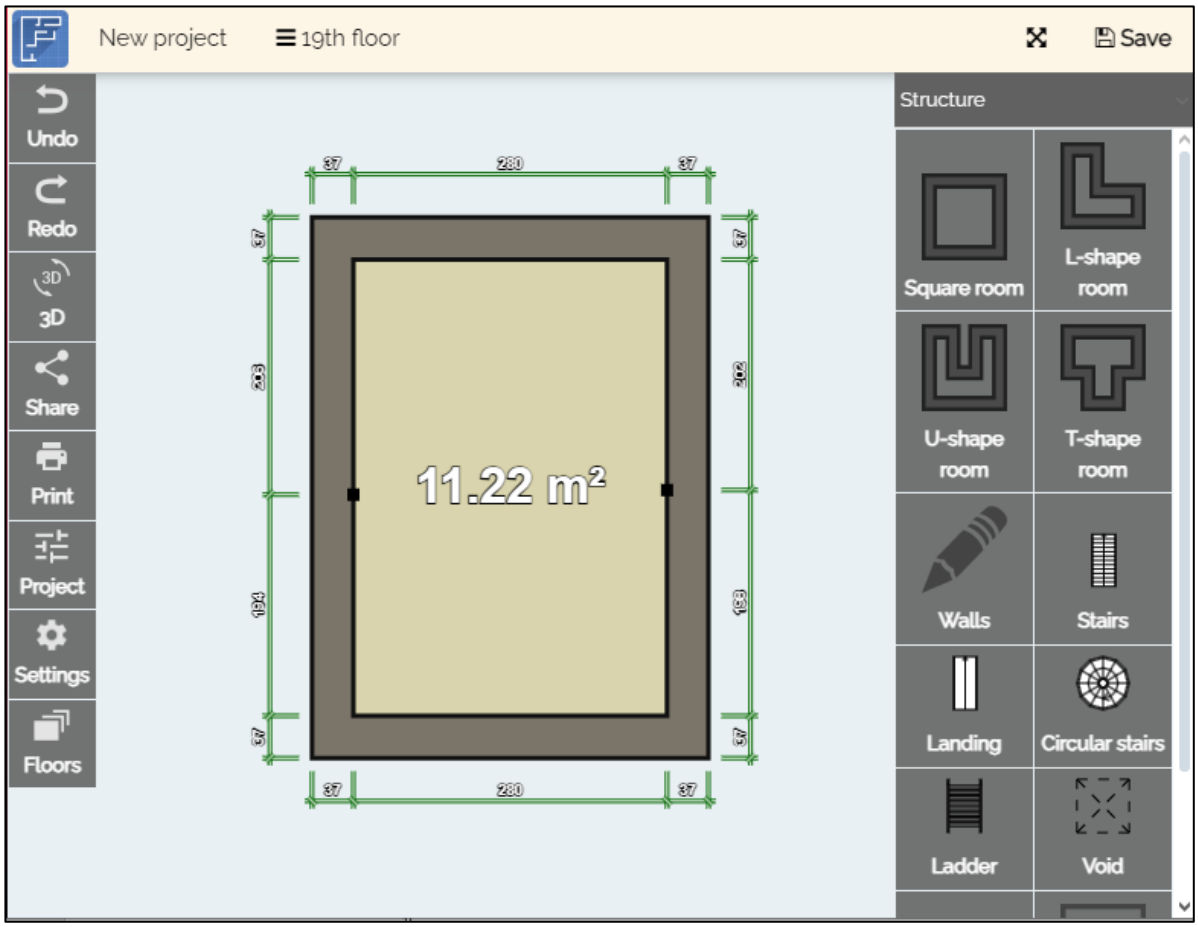

Рисунок 2 – Веб-сайт «Floorplancreator»

Веб-сайт «REMPLANNER»

«REMPLANNER» – инструмент, с помощью которого пользователи могут строить стены, устанавливать окна и двери, а также тщательно прорабатывать интерьер, размещая мебель и другие элементы. В «REM-

PLANNER» предусмотрена возможность добавления нескольких этажей, что позволяет проектировать многоэтажные здания и переключаться между этажами для детального просмотра. Одним из ключевых преимуществ является функция автоматического расчета площади комнаты. Переключение между 2D и 3D-планами предоставляет удобную визуализацию. Минусом «REMPLANNER» является то, что часть функция доступна только на платной основе. На рисунке 3 представлен интерфейс «REMPLANNER».

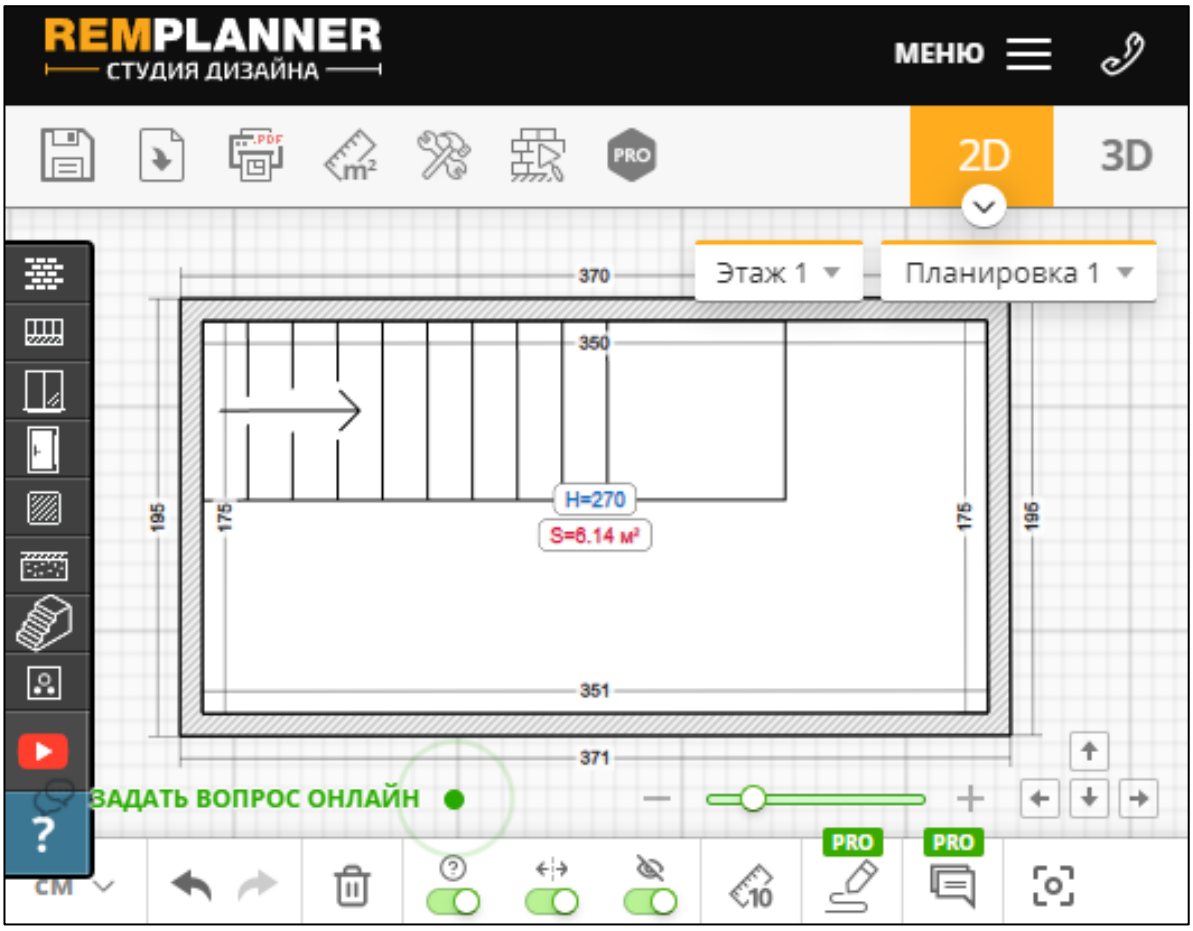

Рисунок 3 – Веб-сайт «REMPLANNER»

1.2. Обзор технологий для реализации проекта

С развитием интернет-технологий и возможностей современных браузеров стало возможным создание интерактивных 3D-приложений, работающих непосредственно в веб-браузере. На рынке существует несколько популярных технологий и фреймворков для разработки 3D-

приложений, таких как Three.js [1], A-Frame [2], Unity WebGL [3] и Babylon.js [4].

Three.js

Ниже приведены достоинства Three.js.

1. Three.js является одним из наиболее известных фреймворков для 3D-графики в браузере. Обширное сообщество и большое количество примеров и документации упрощают обучение и разработку.

2. Легко интегрируется с другими веб-технологиями и библиотеками, такими как WebGL.

3. Three.js поддерживает сложные анимации, тени, материалы и эффекты, что позволяет создавать разнообразные и реалистичные 3D-сцены.

Ниже приведены недостатки Three.js.

1. Для достижения продвинутых результатов может потребоваться значительное время на изучение.

2. На сложных сценах может возникнуть снижение производительности, особенно на менее мощных устройствах.

A-Frame

Ниже приведены достоинства A-Frame.

1. Фреймворк использует HTML-подобный синтаксис, что делает его доступным для широкого круга разработчиков.

2. Специализируется на создании виртуальной и дополненной реальности, предоставляя простые в использовании компоненты.

3. Наследует все преимущества Three.js, предлагая упрощенный интерфейс для работы с 3D-графикой.

Ниже приведены недостатки A-Frame.

1. Меньшая гибкость по сравнению с прямым использованием Three.js, особенно для создания сложных кастомных сцен и анимаций.

2. Может быть менее производительным на сложных сценах и устройствах с ограниченными ресурсами.

Unity WebGL

Ниже приведены достоинства Unity WebGL.

1. Предлагает огромные возможности для создания 3D-игр и приложений, включая физику, анимацию и управление объектами.

2. Имеет большое количество готовых ресурсов, таких как модели, звуки и скрипты, что ускоряет процесс разработки.

3. Кроссплатформенность.

Ниже приведены недостатки Unity WebGL.

1. Требует высокой квалификации и значительных ресурсов для создания и поддержки приложений.

2. Приложения на Unity WebGL могут быть большими по размеру, что увеличивает время загрузки и потребление ресурсов.

Babylon.js

Ниже приведены достоинства Babylon.js.

1. Babylon.js обладает интуитивным API и хорошо организованной документацией, что делает его привлекательным для начинающих разработчиков. Он предоставляет удобные инструменты для создания и управления 3D-сценами, объектами и анимациями, что упрощает процесс разработки.

2. Несмотря на свою простоту, Babylon.js обладает обширными возможностями для создания сложных 3D-приложений. Он поддерживает различные типы материалов, освещение, тени, частицы, анимации костей и многое другое, что позволяет создавать впечатляющие и реалистичные 3D-сцены.

3. Babylon.js оптимизирован для работы с аппаратным ускорением графики, что позволяет достичь высокой производительности даже на устройствах с ограниченными ресурсами. Это особенно важно для создания 3D-приложений, которые должны работать плавно и быстро.

4. У Babylon.js активное и дружелюбное сообщество разработчиков, которые готовы помочь новичкам и делиться опытом. Также фреймворк

имеет открытый исходный код, что позволяет вносить свои правки и улучшения.

5. Babylon.js поддерживает различные платформы и браузеры, что позволяет запускать созданные приложения на различных устройствах, включая компьютеры, смартфоны и планшеты.

6. Использование фреймворка Babylon.js для создания интерактивных 3D-приложений с интеграцией базы данных PostgreSQL [5] открывает широкие возможности для различных областей применения, таких как архитектурное проектирование, образование, игровая индустрия, виртуальный туризм и многое другое.

Ниже приведены недостатки Babylon.js.

1. Для простых и средних проектов Babylon.js может быть идеальным выбором. Однако при разработке крупных проектов с большим количеством сложных 3D-объектов и сложной логикой управления сцены, возникают сложности в поддержке и масштабировании кода. В таких случаях приходится прибегать к оптимизациям и дополнительному упорядочиванию кода, что требует опыта и времени.

2. Несмотря на простоту использования и интуитивный API, для реализации сложных эффектов и возможностей требуется углубленное изучение и время. Освоение всех функций и тонкостей фреймворка может занять значительное время, что может быть трудностью для начинающих разработчиков, стремящихся к созданию сложных приложений.

3. Работа с 3D-ресурсами, такими как текстуры и модели, часто требует значительных объемов данных, что может привести к увеличению времени загрузки приложений и зависимости от скорости интернет-соединения. Это может негативно сказаться на пользовательском опыте, особенно в условиях ограниченного доступа к сети.

Для реализации отрисовки 2D-графики в браузере существуют такие решения как PixiJS [6], CreateJS [7], Two.js [8].

CreateJS

Ниже приведены достоинства CreateJS.

1. Набор включает в себя библиотеки для работы с холстом, создания анимаций, обработки звука и предварительной загрузки ресурсов. Это позволяет охватить широкий спектр задач, связанных с разработкой мультимедийных приложений.

2. Обширная документация и множество примеров делают CreateJS доступным для разработчиков с разным уровнем опыта.

3. Кроссплатформенность.

Ниже приведены недостатки CreateJS.

1. В сложных анимациях и при обработке большого количества графических элементов могут возникать проблемы с производительностью, особенно на менее мощных устройствах.

2. Несмотря на наличие модуля для предварительной загрузки ресурсов, загрузка больших мультимедийных файлов может замедлить начальную загрузку приложения.

Two.js

Ниже приведены достоинства Two.js.

1. Векторная графика обеспечивает высокое качество изображения при любых масштабах, что идеально подходит для создания адаптивных и масштабируемых приложений.

2. Поддержка Canvas и WebGL позволяет использовать Two.js для создания высокопроизводительных приложений, работающих плавно даже на устройствах с ограниченными ресурсами.

3. Кроссплатформенность.

Ниже приведены недостатки Two.js.

1. В сравнении с другими фреймворками, такими как PixiJS, Two.js может предложить меньший набор инструментов и возможностей для создания сложных графических эффектов и взаимодействий.

2. При работе с очень большими сценами или множеством анимаций может возникнуть снижение производительности, особенно на устройствах с ограниченными ресурсами.

PixiJS

Ниже приведены достоинства PixiJS.

1. Производительность: благодаря тому, что PixiJS использует WebGL обеспечивается высокая производительность и возможность обрабатывать более сложные графические сцены.

2. Функциональность: PixiJS имеет много инструментов для работы с пользователем, предоставляя возможность интерактивно взаимодействовать с созданными объектами.

3. Сообщество: PixiJS имеет обширную документацию и множество примеров использования, что делает ее доступной для изучения.

Ниже приведены недостатки PixiJS.

1. PixiJS предоставляет низкоуровневый доступ к функциям WebGL и графики, но не включает готовые высокоуровневые компоненты, такие как интерфейсы, кнопки и сложные виджеты. Это может потребовать больше времени на разработку пользовательского интерфейса.

2. Для сложных проектов может потребоваться значительное время и усилия на оптимизацию кода и графики, чтобы избежать проблем с производительностью, особенно на устройствах с ограниченными ресурсами.

PixiJS и Babylon.js были выбраны для проекта благодаря своим уникальным преимуществам и возможностям. PixiJS предоставляет высокую производительность, достаточный набор функций и простоту для работы с 2D-графикой, что делает его идеальным выбором для создания интерактивных 2D-приложений и игр. Babylon.js, в свою очередь, предлагает возможности для создания 3D-сцен, обеспечивая высокую производительность и поддержку различных платформ.

2. ПРОЕКТИРОВАНИЕ

2.1. Анализ требований

Функциональные требования к системе

Функциональные требования описывают требуемое поведение системы. Требования, выдвинутые для реализации приложения, приведены ниже.

1. Пользователь должен иметь возможность просматривать 3D-модель этажа.
2. Пользователь должен иметь возможность просматривать 2D-план этажа.
3. Пользователь должен иметь возможность добавлять, перемещать и удалять объекты на 2D-плане.
4. Пользователь должен иметь возможность переименовать комнату.
5. Пользователь должен иметь возможность открыть каталог с мебелью на 2D-плане.
6. Пользователь должен иметь возможность открыть каталог с мебелью.

Нефункциональные требования к системе

Нефункциональные требования определяют свойства и ограничения которых должна придерживаться реализуемая система. Ниже приведены нефункциональные требования.

1. Данные должны храниться в базе данных, а в качестве СУБД использовать PostgreSQL.
2. 2D и 3D-клиенты должны быть написаны на JavaScript [9] и HTML.
3. 2D и 3D-модель этажа должны быть реализованы с использованием фреймворков PixiJS и Babylon.js.
4. Сервер приложения должен быть написан на языке программирования C#.

5. Сервер должен быть реализован с использованием ASP.NET 6.0 [10].

2.2. Диаграмма вариантов использования

Для веб-приложения была разработана диаграмма вариантов использования.

Пользователь может выполнять следующие функции:

1) просмотреть 2D-план этажа – пользователь просматривает 2D-план этажа;

2) просмотреть 3D-план этажа – пользователь просматривает 3D-план этажа;

3) построить стену – пользователь строит стену;

4) управление предметов – пользователь передвигает, удаляет или поворачивает предмет;

5) управление стеной – пользователь передвигает, удаляет стену, добавляет окно или дверь;

6) открыть/закрыть каталог – пользователь открывает/закрывает каталог;

7) добавить предмет – пользователь добавляет предмет из каталога;

8) управление дверью – пользователь перемещает, удаляет, изменяет параметры двери;

9) управление окном – пользователь перемещает, удаляет, изменяет параметры окна;

10) переключить вид – пользователь переключает вид между 2D и 3D-представлением этажа;

11) управление комнатой – пользователь переименовывает комнату;

12) добавить этаж – пользователь добавляет новый этаж;

13) выбор этажа – пользователь переключает выбранный этаж.

На рисунке 4 изображена диаграмма вариантов использования.

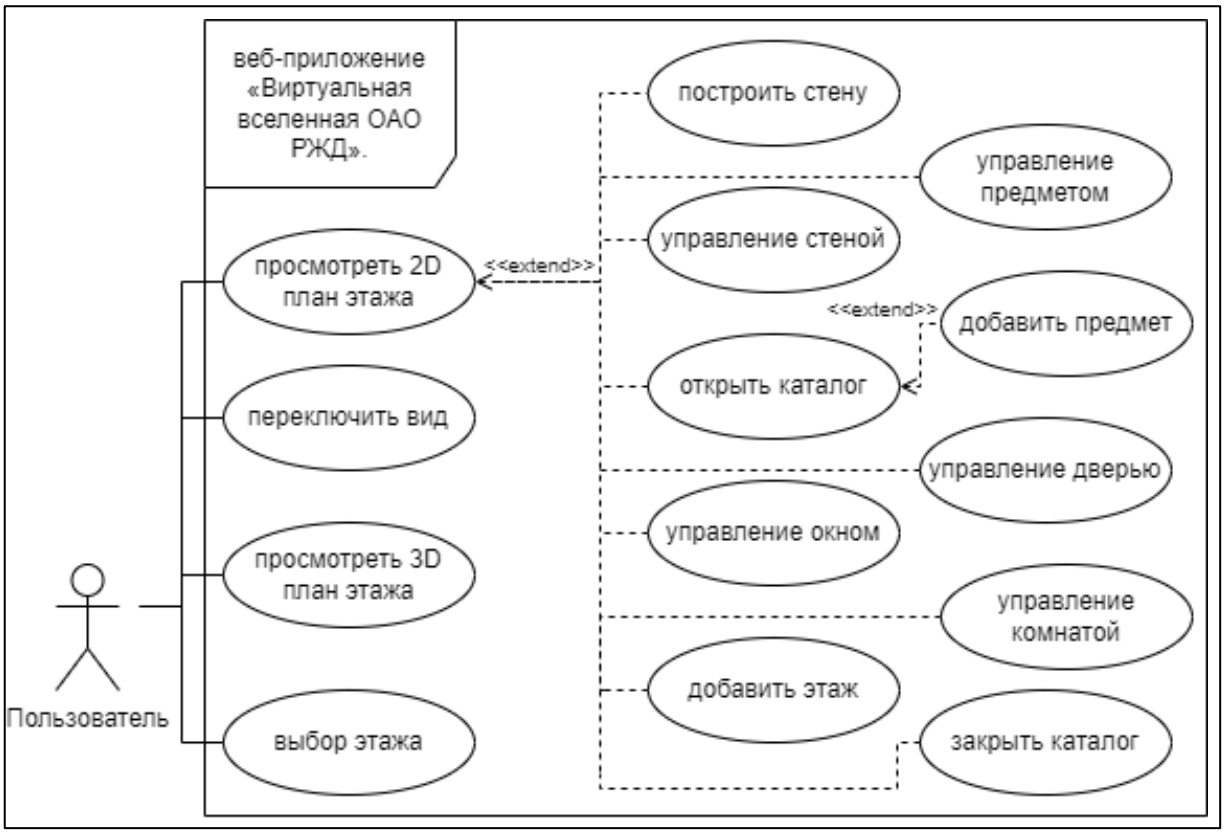

Рисунок 4 – Диаграмма вариантов использования

2.3. Хранилище данных

База данных является центральным хранилищем информации о различных элементах плана этажа. Информация о стенах, окнах, дверях и предметах хранится в таблицах, что обеспечивает структурированное и эффективное управление данными. В качестве хранилища используется СУБД PostgreSQL. СУБД PostgreSQL имеет в себе различные типы данных и обладает высокой расширяемостью, что позволяет создавать таблицы постепенно и легко изменять их. Для хранения информации была спроектирована база данных, которая состоит из 7 таблиц.

1. `Levels` – таблица, в которой хранятся номера этажей.
2. `Rooms` – таблица, в которой хранится название комнаты и углы из которых она состоит.
3. `Items` – таблица, в которой хранится название предмета, его местоположение и размер, угол поворота, путь к модели, Id родительского предмета и слой.

4. `Corners` – таблица, в которой хранятся координаты точек из которых состоит стена.

5. `Walls` – таблица в которой хранятся начало и конец стены.

6. `Doors` – таблица, в которой хранятся ширина и высота двери, а так же отступ от начальной точки стены.

7. `Windows` – таблица, в которой хранятся ширина и высота окна, а так же отступ от начальной точки стены и вершины.

На сервере определены сущности в виде объектов C# [11], из которых автоматически создаются все таблицы благодаря Entity Framework Core [12] и подходу Code First [13]. Подход Code First позволяет быстро и эффективно разрабатывать базу данных, избегая необходимости ручного создания таблиц и связей между ними через SQL код, что значительно сокращает время разработки.

Схема базы данных изображена на рисунке 5.

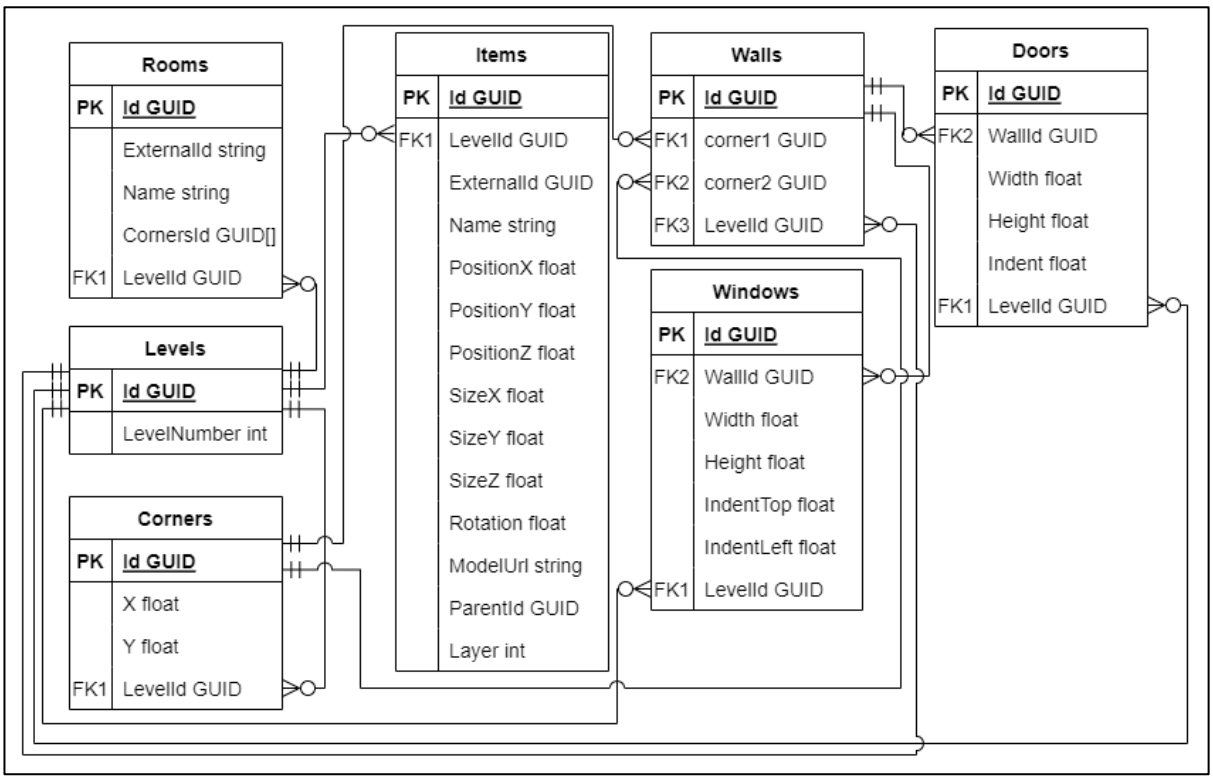

Рисунок 5 – Схема базы данных

2.4. Архитектура веб-приложения

Разрабатываемое веб-приложение имеет клиент-серверную архитектуру. На рисунке 6 приведена диаграмма компонентов разрабатываемого веб-приложения.

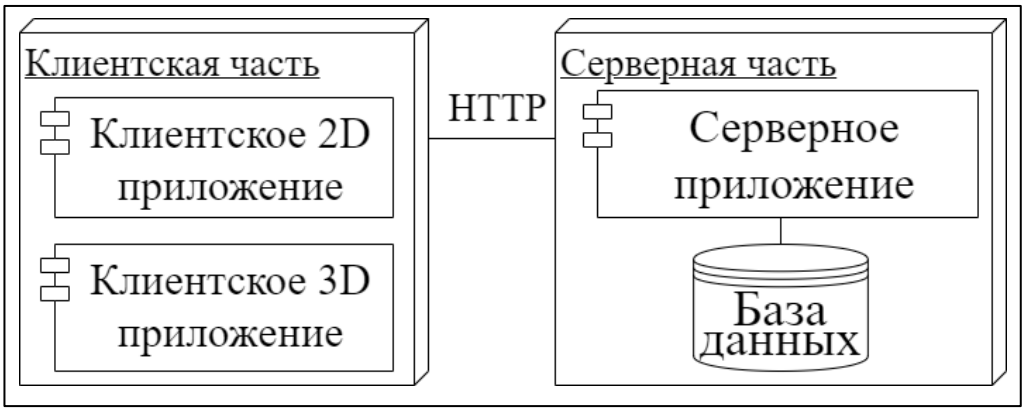

Рисунок 6 – Диаграмма компонентов

Ниже описаны ключевые элементы клиент-серверной архитектуры.

1. Клиент – это устройство или программа, отправляющая запросы к серверу для выполнения определенных задач.

2. Сервер – это система, обрабатывающая запросы от клиентов и предоставляющая им необходимые данные или услуги.

3. Протокол обмена данными – это набор правил, определяющих, каким образом клиенты и серверы взаимодействуют и обмениваются информацией.

Клиентская часть

Клиентская часть приложения отвечает за создание пользовательского интерфейса, который позволяет пользователю взаимодействовать с планом этажа. Визуализация плана выполняется с использованием графического ядра PixiJS для 2D-отображения и фреймворка Babylon.js для 3D-отображения. Эта часть обеспечивает визуализацию объектов, их взаимодействие и навигацию по плану. Пользовательский интерфейс включает в себя различные элементы управления, такие как кнопки для переключения этажей, открытия/закрытия каталога, удаления объектов. Пользователи

имеют возможность добавлять новые объекты и перемещать уже созданные. Кроме того, на клиенте присутствуют поля ввода и ползунки для изменения параметров объектов.

Серверная часть

Серверная часть приложения ответственна за взаимодействие с базой данных PostgreSQL, в которой хранится информация о различных объектах на плане этажа, реализована в соответствии с принципами чистой архитектуры. Этот подход гарантирует высокую гибкость, тестируемость и легкость в поддержке и развитии системы. Чистая архитектура обеспечивает четкое разделение системы на слои, что позволяет минимизировать зависимость между ними и улучшить управляемость проектом. Серверная часть представляет собой API, которое позволяет клиентской части отправлять запросы на создание, получение, обновление и удаление данных. API реализовано с подходом REST, что обеспечивает стандартизированный и эффективный способ обмена данными между клиентом и сервером. Когда клиент отправляет запрос на сервер, сервер обрабатывает запрос и отправляет соответствующий запрос в базу данных. После выполнения всех действий сервер возвращает клиенту JSON ответ если это требуется. Это позволяет клиенту интерпретировать полученные с сервера данные и проводить с ними дальнейшие операции. Здесь осуществляется обработка запросов от клиента, а также управление данными, включая их добавление, изменение и удаление. Эта часть обеспечивает эффективное взаимодействие между клиентской частью и хранилищем данных.

3. РЕАЛИЗАЦИЯ

3.1. Реализация серверной части

Хранилище данных

При разработке веб-приложения применяется подход Code First. Суть данного подхода заключается в задании сущностей объектами в коде, после чего генерируются соответствующие таблицы в базе данных. Код становится центральным местом определения структуры данных и их отношений, что упрощает реализацию приложения. Пример сущности для двери приведен в листинге 1.

Листинг 1 – Сущность `Door`

```
public class Door
{
    public Guid Id { get; set; }
    public Guid WallId { get; set; }
    public float Width { get; set; }
    public float Height { get; set; }
    public float Indent { get; set; }
    public Wall Wall { get; set; } = null!;
    public Level Level { get; set; } = null!;
}
```

Серверная часть

Как было сказано ранее, сервер разрабатывался с использованием подхода REST API [14]. Клиент обменивается с сервером информацией отправляя ему HTTP запросы:

1) POST – создание новой записи;
2) GET – получение существующей записи;
3) PUT – обновление данных записи;
4) DELETE – удаление записи.

С клиента сервер получает данные в виде объекта передачи данных (DTO), после чего в контроллере команда отправляется в обработчик.

Пример объекта передачи данных приведен в листинге 2.

Листинг 2 – Объект передачи данных `AddRoomDto`

```
public class AddRoomDto : IMapWith<Room>
{
    public string ExternalId { get; set; } = null!;
    public string Name { get; set; } = null!;
    public int LevelNumber { get; set; }
```

```
    public List<Guid> CornersId { get; set; } = new List<Guid>();
    public void Mapping(Profile profile)
    {
        profile.CreateMap<AddRoomDto, Room>();}}
```

DTO для создания комнаты реализует интерфейс `IMapWith` для автоматического проецирования данных из DTO в сущность Room с помощью библиотеки AutoMapper.

В `AddRoomDto` передается внешний идентификатор `ExternalId` и название комнаты `Name`, которые не могут быть null, номер этажа `LevelNumber` для разделения комнат по этажам, список углов из которых состоит комната `CornersId`.

Команда `AddRoomCommand` реализует интерфейс `IRequest<Guid>` и хранит в себе данные из DTO. В интерфейсе `IRequest` указывается тип данных в том случае, когда нам необходимо что-то вернуть клиенту. В нашем случае возвращается идентификатор созданной комнаты.

В листинге 3 представлена команда `AddRoomCommand`.

Листинг 3 – Команда `AddRoomCommand`

```
public class AddRoomCommand : IRequest<Guid>
{
    public AddRoomDto Body { get; set; }
}
```

В листинге 4 представлен метод контроллера, отправляющий команду на выполнение.

Листинг 4 – Метод `AddRoom`

```
[HttpPost]
public async Task<ActionResult> AddRoom([FromForm] AddRoomCommand command){
    return Ok(await _mediator.Send(command));
}
```

В методе `AddRoom` с помощью библиотеки MediatR команда отправляется в обработчик. Обработчик реализует интерфейс `IRequestHandler <AddRoomCommand, Guid>`, где `AddRoomCommand` – команда, которую необходимо обработать, `Guid` – тип данных, который необходимо вернуть. Обработчик представлен на листинге 5.

Листинг 5 – Обработчик `RoomCommandsHandler`

```
public class RoomCommandsHandler : IRequestHandler<AddRoomCommand, Guid>{
    private readonly IMetaDbContext _metaDbContext;
    private readonly IMapper _mapper;
    public RoomCommandsHandler(IMetaDbContext metaDbContext, IMapper map-
per){
        _metaDbContext = metaDbContext;
        _mapper = mapper;
    }
    public async Task<Guid> Handle(AddRoomCommand request, CancellationTo-
ken cancellationToken){
        var existRoom = await _metaDbContext.Rooms.SingleOrDefaultAsync(u
=> u.ExternalId == request.Body.ExternalId, cancellationToken);
        var level = await _metaDbContext.Levels.SingleOrDefaultAsync(u =>
u.LevelNumber == request.Body.LevelNumber, cancellationToken: cancella-
tionToken);
        if (level == null)
        {
            throw new Exception("Этаж не найден");
        }
        if (existRoom != null)
        {
            return existRoom.Id;
        }
        else{
            try
            {
                var room = _mapper.Map<Room>(request.Body);
                room.Level = level;
                await _metaDbContext.Rooms.AddAsync(room, cancellationTo-
ken);
                await _metaDbContext.SaveChangesAsync(cancellationToken);
                return room.Id;}
            catch (Exception ex){
                return Guid.Empty;
            }
        }
    }
}
```

В обработчике команды `AddRoomCommand` проверяется существование этажа и существующие комнаты с таким же набором углов. Если этаж существует и аналогичных комнат не найдено, то будет создана новая комната и обработчик вернет идентификатор созданной комнаты в контроллер, а контроллер вернет идентификатор клиенту.

3.2. Реализация клиентской части

Реализация этажей

Прежде чем начать размещать объекты на плане пользователю необходимо создать новый этаж. Для этого в левой части экрана находится па-

нель «Выбор этажа», которая была создана через библиотеку QuickSettings, которая позволяет быстро создавать элементы для управления веб-приложением. На панели есть кнопка «Добавить этаж» по нажатию на которую отправляется запрос на сервер, после чего создается новый этаж. Также на панели есть чекбокс «Нижний уровень», он нужен для того, чтобы этажи добавлялись в отрицательном направлении: -1 этаж, -2 этаж и т.д. Панель для управления этажами изображена на рисунке 7.

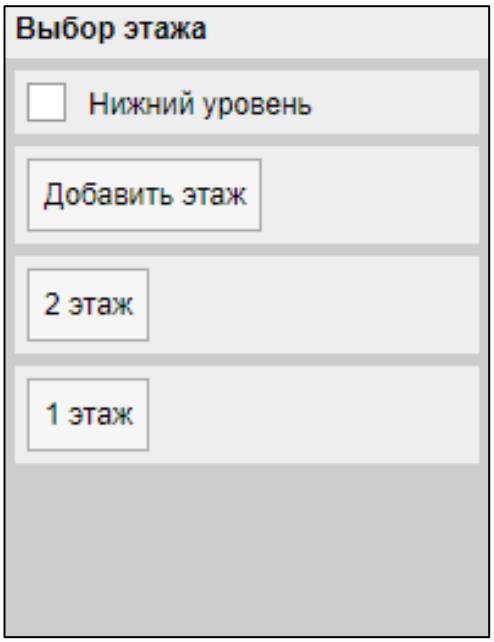

Рисунок 7 – Панель «Выбор этажа»

Реализация построения стен

Перед началом построения стен пользователю необходимо нажать кнопку «Строительство» на правой панели «Планировщик».

После того как пользователь перешел в режим строительства ему необходимо нажать ЛКМ в любом месте полотна. После первого нажатия за курсором будет следовать временная стена, которая показывает, как будет выглядеть стена в конечном итоге. На рисунке 8 изображена временная стена.

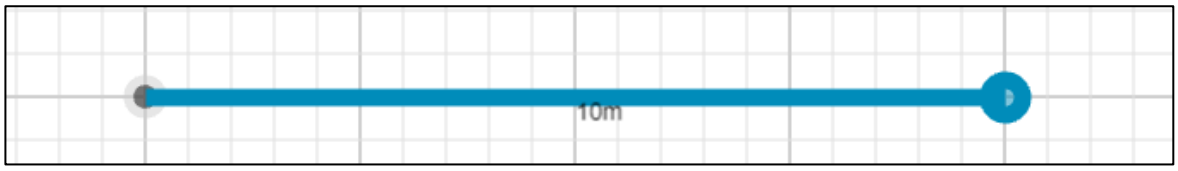

Рисунок 8 – Временная стена

После того как пользователь повторно нажмет ЛКМ, стена будет построена. Построенная стена в 2D и 3D изображена на рисунках 9 и 10.

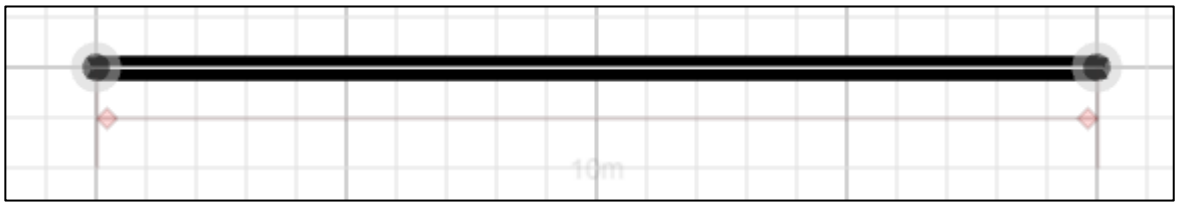

Рисунок 9 – Построенная стена в 2D

Рисунок 10 – Построенная стена в 3D

Метод создания стены представлен на листинге 6.

Листинг 6 – Метод создания стены

```
let corner1 = this.__lastPoint;
let corner2 = corner;
let newCorners = {point1: corner1, point2: corner2}
const data = new FormData()
data.append('Body.corner1', newCorners.point1.id)
data.append('Body.corner2', newCorners.point2.id)
data.append('Body.LevelNumber', this.__plan.level)
await fetch(`${serverAddress}/api/1.0/walls`, {
    method: 'POST',
    body: data
    })
.then(response => response.json())
.then((id) => {
```

```
this.__plan.newWall(newCorners.point1, new-Corners.point2, id);
})
```

В 3D стена представляет из себя прямоугольник, который масштабируется в соответствии с разницей между двумя точками и поворачивается на угол между этими точками. Метод отрисовки стен в 3D представлен в листинге 7.

Листинг 7 – Отрисовка стены в 3D

```
let createWall = function (_wall)
    {
        let point1 = _wall.corner1;
        let point2 = _wall.corner2;
        let windows = _wall.windows;
        let doors = _wall.doors;
        let center = BABYLON.Vector3.Center(point1, point2);
        let direction = point2.subtract(point1);
        let distance = direction.length();
        let height = 3;
        let depth = 0.2;
        let width = distance + depth;
        let wall = BABYLON.MeshBuilder.CreateBox("custom", {width: width,
height: height, depth: depth}, scene);
        wall.position = center.add(new BABYLON.Vector3(0, height / 2, 0));
        let angle = -Math.atan2(direction.z, direction.x);
        wall.rotation.y = angle;
        return wall;
    }
```

Реализация окна

Для добавления окна пользователю необходимо выделить стену нажав по ней ЛКМ, после чего на панели «Планировщик» появится кнопка «Добавить окно». После нажатия на кнопку на стене появится окно. Окно в 2D и 3D изображено на рисунках 11 и 12 соответственно.

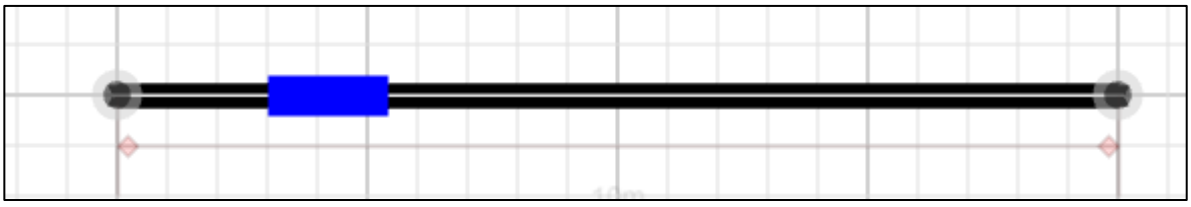

Рисунок 11 – Окно в 2D

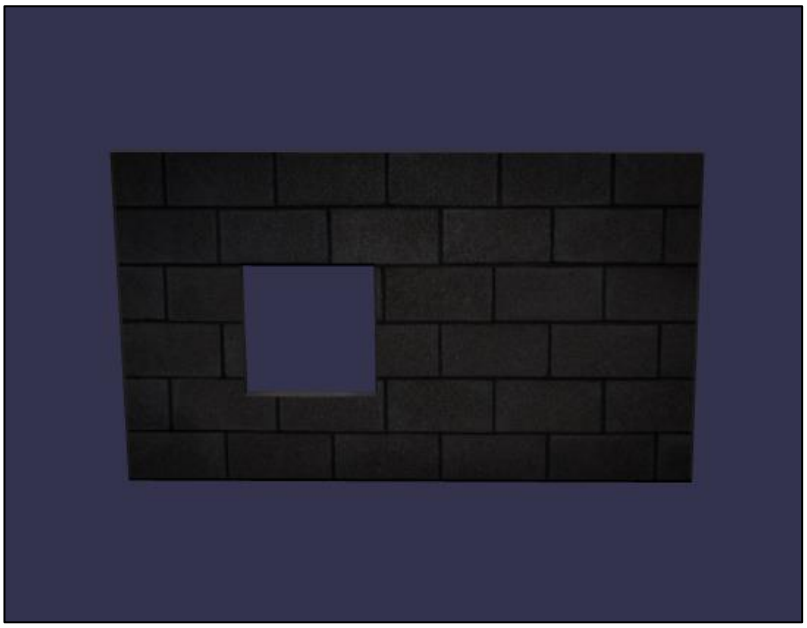

Рисунок 12 – Окно в 3D

Для построения окна в 3D создается прямоугольник, который соответствует размерам окну, после чего смещается к своей позиции и вырезается из стены. Построение окна в 3D представлено в листинге 8.

Листинг 8 – Построение окна в 3D

```
if (windows){
        windows.forEach((windowInfo) => {
            let windowWidth = windowInfo.width || 1;
            let windowHeight = windowInfo.height || 1;
            let windowLeftOffset = windowInfo.indentLeft || 0;
            let windowTopOffset = windowInfo.indentTop || 0;
            let windowHole = BABYLON.MeshBuilder.CreateBox("custom",{width:
windowWidth,height: windowHeight,depth: depth + 0.1}, scene);
            windowLeftOffset += depth / 2
            let localWindowPositionX = -width / 2 + windowLeftOffset + win-
dowWidth / 2;
            let localWindowPositionY = height / 2 - windowTopOffset - win-
dowHeight / 2;
            let windowHolePosition = new BABY-
LON.Vector3(localWindowPositionX, localWindowPositionY, 0);
            windowHolePosition = BABY-
LON.Vector3.TransformCoordinates(windowHolePosition, BABY-
LON.Matrix.RotationY(angle));
            windowHole.position = wall.position.add(windowHolePosition);
            windowHole.rotation.y = angle;
            let wallCSG = BABYLON.CSG.FromMesh(wall);
            let windowHoleCSG = BABYLON.CSG.FromMesh(windowHole);
            let wallWithHoleCSG = wallCSG.subtract(windowHoleCSG);
            let wallWithHole = wallWithHoleCSG.toMesh("custom",
wall.material, scene);
            wall.dispose();
            windowHole.dispose();
            wall = wallWithHole;
            });}
```

Если пользователь нажмет ЛКМ по окну, то откроется панель с настройками окна. Пользователь может изменять ширину окна, высоту окна и отступ окна от вершины стены.

Параметры окна изображены на рисунке 13.

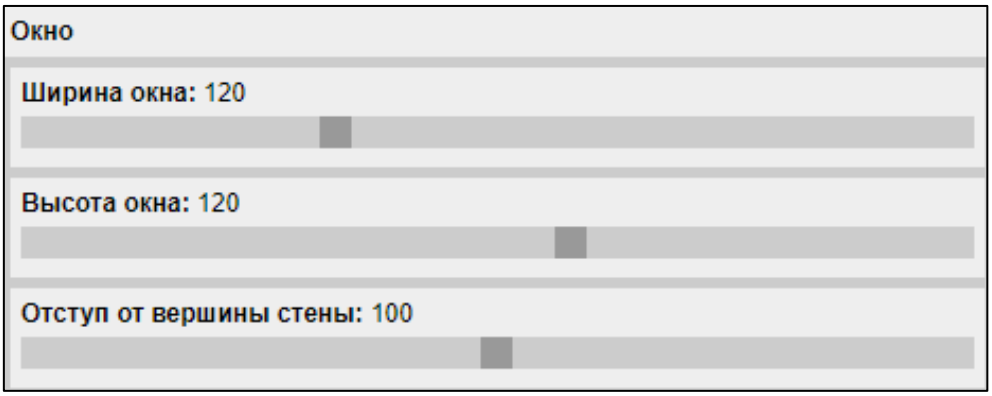

Рисунок 13 – Параметры окна

Реализация создания предметов

Для добавления предметов на план пользователю необходимо нажать кнопку «Открыть/закрыть каталог», которая открывает список доступных предметов. Каталог предметов изображен на рисунке 14.

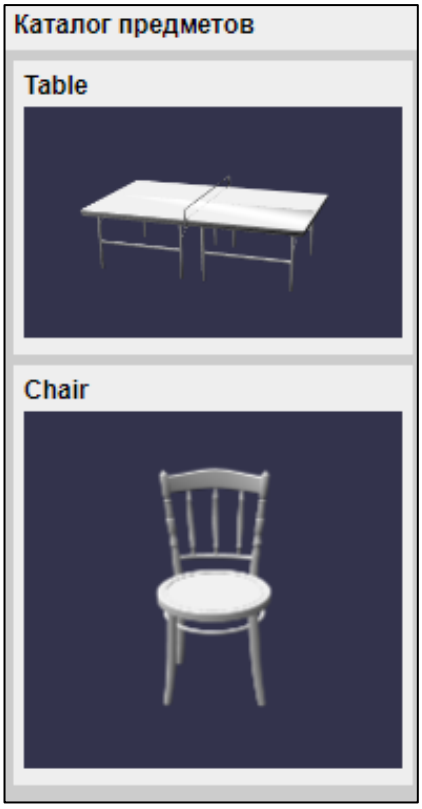

Рисунок 14 – Каталог предметов

После того как пользователь выберет нужный предмет из каталога, за курсором будет следовать временный предмет-превью, отображающий, как выбранный объект будет располагаться на плане. Предпросмотр предмета помогает пользователю определить его точное расположение и вид на плане, что можно увидеть на рисунке 15. После нажатия ЛКМ по полотну будет вызван метод `__addItemModeMouseUp`, после чего предмет будет установлен на этаже. В листинге 9 представлено создание мебели на клиенте.

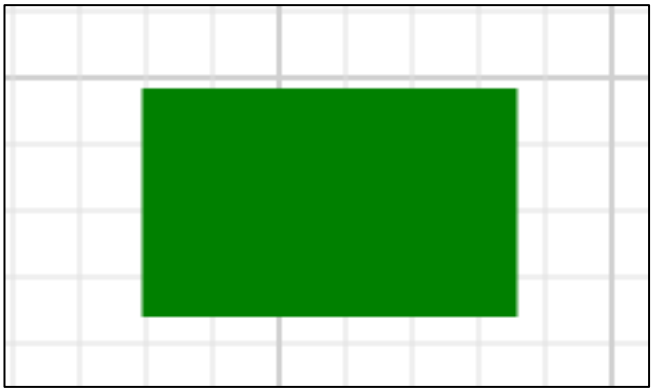

Рисунок 15 – Предпросмотр предмета

Листинг 9 – Создание предмета на клиенте

```
__addItemModeMouseUp(evt) {
        if (this.__mode === plannerModes.ADD_ITEM) {
            var newItem = {};
            let co = evt.data.getLocalPosition(this.__planContainer);
            let cmCo = new Vector2(co.x, co.y);
            cmCo.x = Dimensioning.pixelToCm(cmCo.x);
            cmCo.y = Dimensioning.pixelToCm(cmCo.y);
            let item = this.__plan.tempItem
            newItem.externalId = Utils.guide();
            newItem.name = item.name;
            newItem.positionX = cmCo.x;
            newItem.positionY = 0;
            newItem.positionZ = cmCo.y;
            newItem.sizeX = item.size.x;
            newItem.sizeY = item.size.y;
            newItem.sizeZ = item.size.z;
            newItem.rotation = 0;
            newItem.modelURL = item.modelURL;
            this.__plan.newItem(newItem);
            let data = new FormData();
            data.append("ExternalId", newItem.externalId);
            data.append("Name", newItem.name);
            data.append("PositionX",
newItem.positionX.toString().replace(".", ","));
            data.append("PositionY",
newItem.positionY.toString().replace(".", ","));
```

```
            data.append("PositionZ",
newItem.positionZ.toString().replace(".", ","));
            data.append("SizeX", newItem.sizeX.toString().replace(".",
","));
            data.append("SizeY", newItem.sizeY.toString().replace(".",
","));
            data.append("SizeZ", newItem.sizeZ.toString().replace(".",
","));
            data.append("Rotation", newItem.rotation);
            data.append("modelURL", newItem.modelURL);
            data.append("LevelNumber", this.__plan.level);
            fetch(`${serverAddress}/api/1.0/items`, {body: data, method:
"POST"})
        }
    }
```

На сервере из полученных данных создается новый предмет и сохраняется в базу данных. Создание предмета на сервере представлено в листинге 10.

Листинг 10 – Создание предмета на сервере

```
public async Task Handle(AddItemCommand request, CancellationToken cancel-
lationToken)
{
    var item = await _metaDbContext.Items.SingleOrDefaultAsync(u =>
u.ExternalId == request.ExternalId, cancellationToken);
    var level = await _metaDbContext.Levels.SingleOrDefaultAsync(u =>
u.LevelNumber == request.LevelNumber,
        cancellationToken);

    if (item != null)
{
        throw new Exception("Предмет уже существует");}
    if (level == null)
{
        throw new Exception("Этаж не найден");}
    item = new Item()
        {
        Id = Guid.NewGuid(),
        ExternalId = request.ExternalId,
        Name = request.Name,
        PositionX = request.PositionX,
        PositionZ = request.PositionZ,
        SizeX = request.SizeX,
        SizeY = request.SizeY,
        SizeZ = request.SizeZ,
        Rotation = request.Rotation,
        ModelUrl = request.modelURL,
        ParentId = request.parentId,
        Layer = 1,
        Level = level
        };
    await _metaDbContext.Items.AddAsync(item, cancellationToken);
    await _metaDbContext.SaveChangesAsync(cancellationToken);
}
```

Предмет на этаже в 2D изображен на рисунке 16.

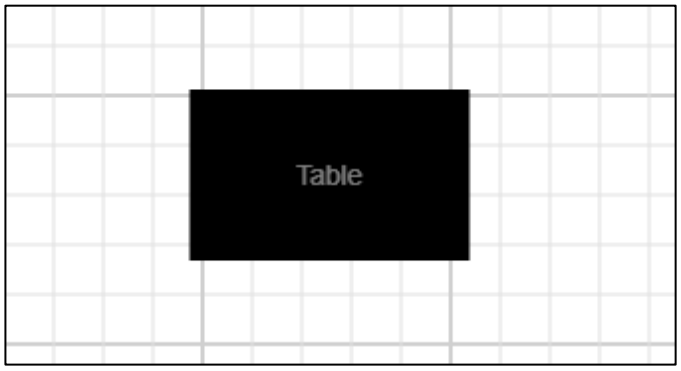

Рисунок 16 – Предмет на этаже в 2D

Предмет на этаже в 3D изображен на рисунке 17.

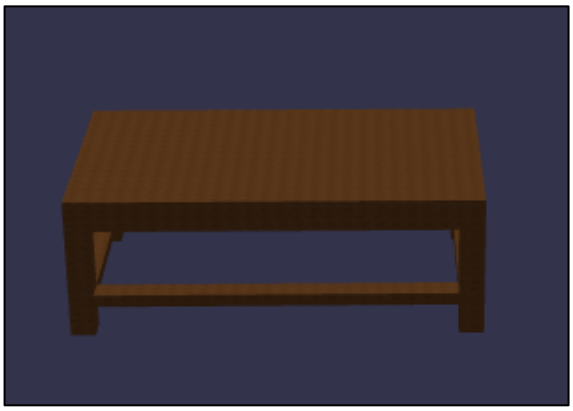

Рисунок 17 – Предмет на этаже в 3D

Реализация двери

Для добавления двери пользователю необходимо выделить стену нажав по ней ЛКМ, после чего на панели «Планировщик» появится кнопка «Добавить дверь». После нажатия на кнопку на стене появится дверь. Дверь в 2D и 3D изображена на рисунках 18 и 19.

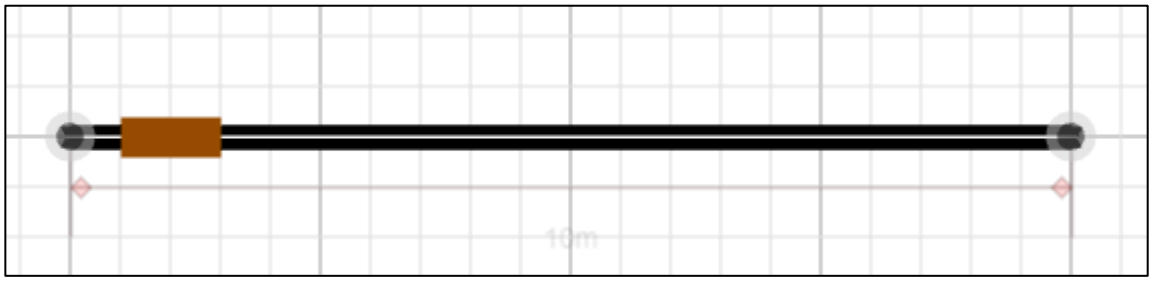

Рисунок 18 – Дверь в 2D

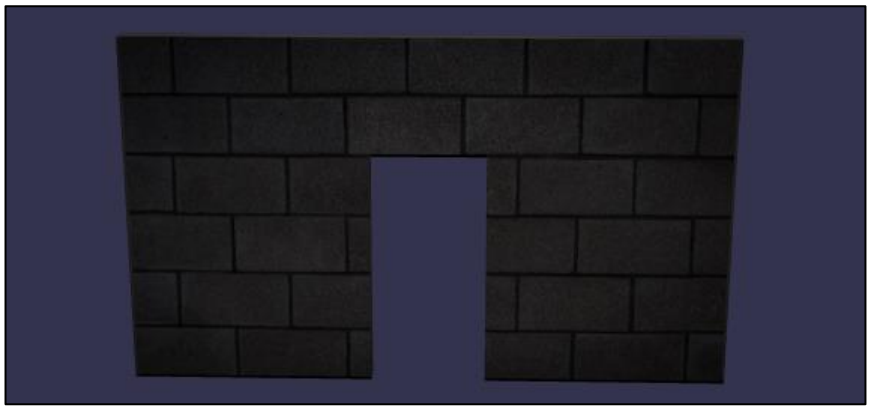

Рисунок 19 – Дверь в 3D

Для построения двери аналогично окну необходимо создать прямоугольник, соответствующий двери, после чего сместить его на местоположение двери и вырезать из стены. Метод для создания двери в 3D представлен на листинге 11.

Листинг 11 – Метод создания двери в 3D

```
if (doors) {
    doors.forEach((doorInfo) => {
        let doorWidth = doorInfo.width || 1;
        let doorHeight = doorInfo.height || 2;
        let doorLeftOffset = doorInfo.indentLeft || 0;

        let doorDepth = depth + 0.1;
        doorLeftOffset += depth / 2
        let localDoorPositionX = -width / 2 + doorLeftOffset + doorWidth /
2;
        let localDoorPositionY = -height / 2 + doorHeight / 2;
        let doorHolePosition = new BABYLON.Vector3(localDoorPositionX, lo-
calDoorPositionY, 0);
        doorHolePosition = BABY-
LON.Vector3.TransformCoordinates(doorHolePosition, BABY-
LON.Matrix.RotationY(angle));
        let doorHole = BABYLON.MeshBuilder.CreateBox("custom", {
            width: doorWidth,
            height: doorHeight,
            depth: doorDepth
        }, scene);
        doorHole.position = wall.position.add(doorHolePosition);
        doorHole.rotation.y = angle;
        let wallCSG = BABYLON.CSG.FromMesh(wall);
        let doorHoleCSG = BABYLON.CSG.FromMesh(doorHole);
        let wallWithHoleCSG = wallCSG.subtract(doorHoleCSG);
        let wallWithHole = wallWithHoleCSG.toMesh("custom", wall.material,
scene);
        wall.dispose();
        doorHole.dispose();
        wall = wallWithHole;
    });
}
```

Если пользователь нажмет ЛКМ по двери, то откроется панель с настройками двери. Пользователь может изменять ширину и высоту двери. Параметры двери изображены на рисунке 20.

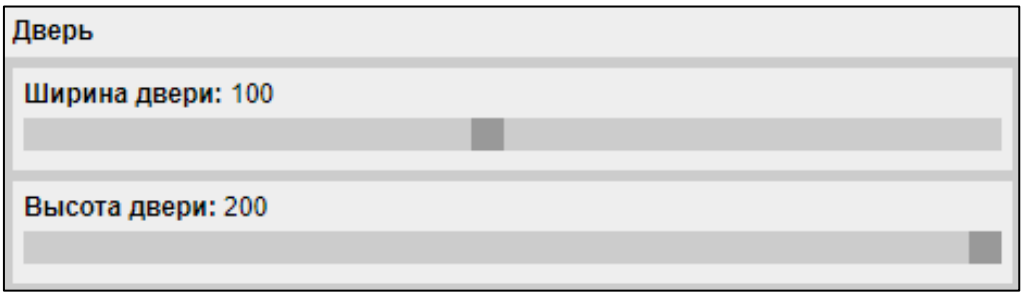

Рисунок 20 – Параметры двери

Реализация комнат

Для создания комнаты пользователю необходимо создать замкнутый контур из стен. При нажатии на комнату пользователю будет доступна возможность изменить название комнаты. Комната в 2D и 3D и изменение названия изображены на рисунках 21 и 22.

Рисунок 21 – Комната в 2D

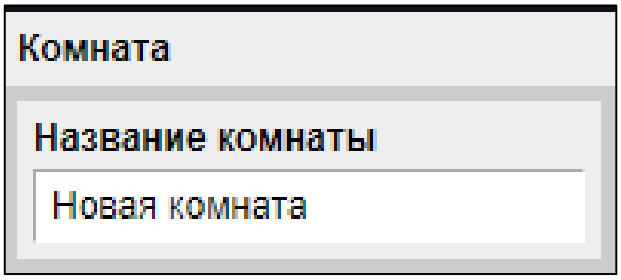

Рисунок 22 – Изменение названия комнаты

4. ТЕСТИРОВАНИЕ

В таблице 1 представлены результаты тестирования.

Таблица 1 – Результаты тестирования

№	Название теста	Действие	Ожидаемый результат	Результат теста
1	Перемещение мебели	Пользователь выбирает мебель и зажимает левую кнопку мыши, после чего передвигает мебель и отпускает левую кнопку мыши	В базе данных сохраняются новые данные местоположения мебели	Пройден
2	Создание мебели	Пользователь выбирает мебель из каталога и нажимает левую кнопку мыши на 2D-плане	На 2D-плане отрисовывается новая мебель и сохраняется в базу данных	Пройден
3	Удаление мебели	Пользователь выбирает мебель левой кнопкой мыши и нажимает кнопку «удалить»	Мебель удаляется с 2D-плана и удаляется из базы данных	Пройден
4	Изменение угла поворота мебели	Пользователь выбирает мебель левой кнопкой мыши и перемещает ползунок с углом поворота	Мебель поворачивается на 2D-плане и в базе данных сохраняется новый угол поворота	Пройден
5	Открытие каталога	Пользователь нажимает кнопку «открыть каталог»	На экране пользователя открывается каталог с мебелью	Пройден
6	Постройка стены	Пользователь нажимает кнопку «строить», после чего нажимает левую кнопку мыши в двух точках плана	На 2D-плане отрисовывается стена	Пройден
7	Удаление стены	Пользователь выбирает мебель левой кнопкой мыши и нажимает кнопку «удалить»	Стена удаляется с 2D-плана и удаляется из базы данных	Пройден
8	Перемещение стены	Пользователь выбирает стену и зажимает левую кнопку мыши, после чего передвигает мебель и отпускает левую кнопку мыши	В базе данных сохраняются новые данные местоположения точек стены	Пройден
9	Добавление окна	Пользователь выбирает стену и нажимает кнопку «добавить окно»	На 2D-плане отрисовывается окно и сохраняется в базу данных	Пройден
10	Добавление двери	Пользователь выбирает стену и нажимает кнопку «добавить дверь»	На 2D-плане отрисовывается дверь и сохраняется в базу данных	Пройден

Юзабилити тестирование

Юзабилити тестирование [15] – один из самых эффективных методов проверки удобства интерфейса. Целью являлась проверка удобства разработанного интерфейса. Для этого были сформированы следующие задачи:

1) создать новую стену;
2) создать новое окно;
3) создать новую дверь;
4) открыть каталог;
5) добавить предмет;
6) переместить объект;
7) переключить вид;
8) переключить этаж;
9) добавить этаж.

В проведенном юзабилити тестировании приняли 4 человека. При выполнении тестирования затруднений не возникло, задачи были выполнены успешно.

Тестирование пользовательского интерфейса

Тестирование пользовательского интерфейса было выполнено в следующих браузерах:

1) Opera GX;
2) Google Chrome;
3) Microsoft Edge;
4) Mozilla Firefox;
5) Яндекс Браузер.

В каждом из этих браузеров все элементы веб-приложения отображались корректно. В ходе проверки не было обнаружено никаких ошибок. Разработанное веб-приложение полностью соответствует заявленным требованиям и отличается удобством в использовании.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было разработано веб-приложение «Виртуальная вселенная ОАО РЖД».

В ходе работы решены следующие задачи:

1) проанализирована предметная область;
2) спроектировано веб-приложение;
3) реализовано веб-приложение;
4) протестировано разработанное веб-приложение.

В результате работы поставленные цели были достигнуты.

В будущем планируется реализовать следующие задачи:

1) разграничить по ролям обычного пользователя и администратора;
2) реализовать возможность интерактивного добавления/удаления предметов в каталоге;
3) добавить возможность разделения каталога на категории;
4) добавить функционал для синхронизации веб-приложения с базой данных турникетов для отображения текущего состояния сотрудника;
5) добавить функционал для изменения текстур у стен;
6) добавить функционал для добавления номера у мебели для упрощения инвентаризации;
7) добавить возможность отображения всех этажей одновременно.

ЛИТЕРАТУРА

1. Three.js. [Электронный ресурс] URL: https://threejs.org (дата обращения: 31.05.2024 г.).

2. A-Frame. [Электронный ресурс] URL: https://aframe.io (дата обращения: 31.05.2024 г.).

3. Unity WebGL. [Электронный ресурс] URL: https://docs.unity3d.com/Manual/webgl.html (дата обращения: 31.05.2024 г.).

4. Babylon.js. [Электронный ресурс] URL: https://doc.babylonjs.com (дата обращения: 31.05.2024 г.).

5. PostgreSQL. [Электронный ресурс] URL: https://www.postgresql.org (дата обращения: 31.05.2024 г.).

6. PixiJS. [Электронный ресурс] URL: https://pixijs.com (дата обращения: 31.05.2024 г.).

7. CreateJS. [Электронный ресурс] URL: https://createjs.com (дата обращения: 31.05.2024 г.).

8. Two.js. [Электронный ресурс] URL: https://two.js.org (дата обращения: 31.05.2024 г.).

9. JavaScript. [Электронный ресурс] URL: https://learn.javascript.ru (дата обращения: 31.05.2024 г.).

10. ASP.NET 6.0. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/aspnet/core/release-notes/aspnetcore-6.0?view=aspnetcore-6.0 (дата обращения: 31.05.2024 г.).

11. C#. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 31.05.2024 г.).

12. Центр документации EntityFramework. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/ef (дата обращения: 31.05.2024 г.).

13. Руководство. Начало работы с Entity Framework 6 Code First с помощью MVC 5. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-

mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application (дата обращения: 31.05.2024 г.).

14. Subramanian H. Hands-On RESTful API Design Patterns and Best Practices. / H. Subramanian, P. Raj // Packt Publishing, 2024. – 378 с.

15. Карпович Е.Е. Методы тестирования и отладки программного обеспечения: учебник. // Москва: МИСИС, 2020. – 136 с.