

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

ОФИСНЫЙ КОНСЬЕРЖ

Студент Кузнецов Алексей Сергеевич
(подпись, дата) 04.06.2025

Руководитель Токарев А.Н.
(подпись, дата) 04.06.2025

Нормоконтролёр Бычков А.С.
(подпись, дата) 04.06.2025

Работа допущена к защите (протокол заседания кафедры от 29.05.2025 № 16)

Заведующий кафедрой Митрохин М.А.
(подпись)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 9 июня 2025 № 10)

Секретарь ГЭК Гурин Е.И.
(подпись)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 05 » 05 2025г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Кузнецов Алексей Сергеевич гр. 21BVB1 факультета вычислительной техники.

2. Тема работы «Офисный консьерж» Тема утверждена
приказом ПГУ №601/о от "6" мая 2025 г.

3. Руководитель бакалаврской работы к.т.н., доцент Токарев А.Н.

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

Разработать приложение «Офисный консьерж», для автоматизации учета ресурсов: мебели, оргтехники и прочее, для закрепления сотрудников за их рабочими местами. Реализовать систему бронирования переговорных комнат.

Структура приложения должна включать в себя клиентскую и серверную части приложения, с возможностью хранения данных в формате JSON.

Клиентская часть программы должна обладать интуитивным понятным графическим интерфейсом.

Серверная часть должна включать в себя следующий функционал: получение данных от клиента, запись данных в файл в формате JSON, возврат информации к клиенту в обработанном виде.

Взаимодействие серверной и клиентской частей программы должно осуществляться с помощью *REST API*.

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной области, изучить аналоги существующих программ офисных консьержей, способы их реализации, изучить способы проектирования и разработки клиент-серверного приложения с применением *REST API* методологии. Необходимо реализовать программу, используя функционал фреймворка *ASP.Net core* и JS-библиотеки *React*.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	15%	26.04.2023-30.04.2023	
2	Анализ требований	4%	01.05.2023-02.05.2023	
3	Выбор технологий, применяемых при проектировании и разработке	15%	03.05.2023-07.05.2023	
4	Описание алгоритма	13%	08.05.2023-11.05.2023	
5	Разработка приложения «Офисный консьерж»	45%	12.05.2023-29.05.2023	
6	Оформление пояснительной записки	8%	30.05.2023-02.06.2023	

Дата выдачи задания "06" мая 2025 г.

Руководитель бакалаврской работы  06.05.2025 Токарев А.Н.
(подпись, дата)

Задание к исполнению принял студент  16.05.2025 Кузнецов А.С.
(подпись, дата)

Работу к защите допустить
Декан факультета  09.06.2025 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 79 листа, 31 рисунок, 1 таблицу, 5 источников и 3 приложения.

ОФИСНЫЙ КОНСЬЕРЖ, REST API, JSON

Цель работы – разработать ПО для автоматизации рутинной работы офис-менеджеров.

Объектами разработки являются:

- серверное приложение, которое принимает данные от клиента в формате JSON, записывает их в текстовый файл для хранения. Кортёж данных, записанных сервером, отображается на клиентской части приложения;
- клиентское приложение позволяет управлять ресурсами офиса, рабочими местами и бронированием переговорных комнат. Клиент фактически должен быть цифровым помощником офис менеджера;
- в результате дипломного проектирования разработано приложение, полностью соответствующее поставленной задаче ВКР;
- разработана серверная часть приложения, которая получает, сохраняет в текстовый файл данные в формате JSON и возвращает их клиенту;
- реализовано API для взаимодействия серверной и клиентской части приложения;
- создана клиентская часть приложения, которая отображает на сайте локацию офиса, рабочие места с помощью интерактивных областей, кнопки для редактирования и бронирования.

ПГУ 1.09.03.01.15.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Куницын А.С.		20.06.25
Провер.		Токарев А.Н.		20.06.25
Н. Контр.		Бычков А.С.		20.06.25
Утверд.		Митрохин М.А.		20.06.25

Лит.	Лист	Листов
	2	

Офисный консьерж
Пояснительная записка

ФВТ, каф. ВТ
гр.21ВВВ1

Содержание

Перечень сокращений и обозначений	5
Введение	6
1 Офисный консьерж	8
1.1 Предпроектный анализ	8
1.1.1 Актуальность	8
1.1.2 Описание потребностей пользователей	9
1.1.3 Анализ аналогов	10
1.1.4 Постановка задачи	12
1.1.5 Выбор языка программирования	14
1.1.6 Выбор фреймворка для серверной части приложения	17
1.1.7 Хранение данных.	18
1.1.8 Формат хранения JSON	19
1.1.9 Выбор платформы хостинга серверной части и клиентской	19
1.1.10 Выбор фреймворка для реализации графического интерфейса	20
1.1.11 Выбор платформы для разработки	22
1.1.12 Выводы по предпроектному анализу	22
1.2 Проектирование и разработка	23
1.2.1 Разработка серверной части приложения	23
1.2.2 Реализация основных HTTP-запросов	28
1.2.3 Разработка клиентской части приложения	30
1.3 Анализ результатов	34
1.3.1 Тестирование API	34
1.3.2 Тестирование пользовательского интерфейса	37
2 Охрана труда	41
2.1 Характеристика потенциальных опасностей и вредностей, которые могут возникнуть при работе с ЭВМ	41
2.2 Классификации и категорировании помещений, в которых производится разработка проектов	41

2.2.1 Классификация по взрывопожарной и пожарной опасности	41
2.2.2 Классификация по опасности поражения электрическим током	41
2.3 Санитарно-гигиенические нормы помещения	44
Заключение	45
Список использованных источников	46
Приложение А	47
Приложение А.1 – Алгоритм обращения к серверу.....	48
Приложение А.2 – Диаграмма компонентов	49
Приложение А.3 – Диаграмма развертывания	50
Приложение А.4 – Диаграмма классов	51
Приложение Б – Листинг программы.....	52
Приложение В – Презентация.....	74

Перечень сокращений и обозначений

БД – база данных

ООП – объектно-ориентированное программирование

ЭВМ – электронно-вычислительная машина

API (Application Programming Interface) – программный интерфейс приложения

ASP.Net core – платформа для разработки серверов для веб-приложения

Azure – облачная платформа от компании Microsoft

Backend – серверная часть приложения

Frontend – клиентская часть приложения

HTML (Hyper Text Markup Language) – язык гипертекстовой разметки

JS – язык программирования JavaScript, которые используется в основном для разработки веб-приложений

React – фреймворк для JavaScript

REST (Representational State Transfer) – передача репрезентативного состояния или передача «самоописываемого» состояния

UML – язык моделирования

URL – система унифицированных адресов электронных ресурсов.

UI – это компоненты интерфейса, которые отображаются пользователю

Введение

На данный момент для офис-менеджеров нет удобного приложения, которое закрывало бы все их потребности. Поэтому цифровая система офисного консьержа является актуальной в условиях современного корпоративного пространства, где повышенное внимание уделяется комфорту, удобству и эффективности работы сотрудников. Офисный консьерж представляет собой цифровой сервис, предназначенный для автоматизации и оптимизации различных административных и бытовых задач внутри офиса.

Целью данного проекта является создание приложения офисного консьержа, которое позволит автоматизировать выполнение рутинных задач, таких как бронирование переговорных комнат, информирование сотрудников о важных событиях и новостях компании, а также предоставление справочной информации о местоположении нужных инструментов для работы. Инструмент предназначен для повышения эффективности внутренней коммуникации, сокращения времени на выполнение административных процедур и улучшения общего уровня сервиса внутри организации.

Задачи, решаемые в процессе ВКР, требуют реализации: управление бронированием ресурсов (переговорных комнат, техники), учет расходных материалов и оборудования в цифровом виде, интеграцию с календарями и системами уведомлений, а также возможность взаимодействия с пользователями через чат-бот или интерфейс. Важным аспектом является обеспечение простоты использования системы, ее надежности и безопасности данных.

При разработке системы офисного консьержа учитываются необходимость интеграции с существующими корпоративными системами, настройка прав доступа для различных групп пользователей.

Данная дипломная работа включает три основных раздела:

– анализ аналогов на рынке цифровых консьержей, выбор технологий для реализации программного продукта с последующим тестированием;

– раздел разработки, в котором рассмотрены вопросы реализации программного обеспечения;

– раздел БЖД, в котором рассмотрены классификации опасных для работы помещений, проведен анализ условий зрительной работы.

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		7

1 Офисный консьерж

1.1 Предпроектный анализ

1.1.1 Актуальность

Тема "Офисный консьерж" обладает высокой актуальностью в современном бизнесе и корпоративной среде, поскольку эффективность внутренних процессов и уровень сервиса в офисах напрямую влияют на продуктивность сотрудников и общее впечатление о компании. В условиях растущей конкуренции и необходимости оптимизации ресурсов автоматизация рутинных задач становится все более востребованной.

Разработка системы офисного консьержа позволяет автоматизировать выполнение различных административных и бытовых функций, таких как бронирование переговорных комнат, учет расходных материалов, информирование сотрудников о важных событиях и предоставление справочной информации. Это значительно сокращает время на выполнение рутинных операций, повышает точность и надежность данных, а также создает более комфортные условия для работы сотрудников. Кроме того, внедрение такой системы способствует улучшению внутренней коммуникации, повышению уровня сервиса внутри организации и оптимизации использования офисных ресурсов.

В современном мире цифровых технологий автоматизированные решения для управления офисом становятся неотъемлемой частью эффективной корпоративной инфраструктуры.

Разработка офисного консьержа также имеет практическую ценность для студентов и специалистов в области информационных технологий, программирования и управления бизнес-процессами. Такой проект позволяет приобрести навыки работы с системами автоматизации, интеграции различных сервисов и обеспечения безопасности данных.

Таким образом, тема "Офисный консьерж" является актуальной как с точки зрения повышения эффективности работы внутри компании, так и с

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		8

перспективы внедрения современных технологий в повседневную деятельность организаций. Реализация данной системы способствует созданию более организованной, удобной и технологичной рабочей среды, что особенно важно в условиях динамично меняющегося делового мира.

1.1.2 Описание потребностей пользователей

Для начала, необходимо определить группу заинтересованных лиц в использовании офисного консержа. Можно предположить, что основными пользователями могут быть следующие группы:

- сотрудникам офиса для быстрого и удобного выполнения рутинных задач, таких как бронирование переговорных комнат, поиск информации о доступных ресурсах и услугах внутри компании;
- административному персоналу для учета и контроля за расходными материалами, техникой, мебелью и другими ресурсами офиса, а также для организации и планирования мероприятий;
- офис-менеджерам для автоматизации процессов управления расписанием, контроля за состоянием ресурсов и обеспечения комфортной рабочей среды.

После определения группы пользователей, которые заинтересованы в коммерческом использовании данные проект нужно определить зачем им это нужно.

- контроль за расходами, запасами и состоянием оборудования помогает снизить издержки и своевременно реагировать на потребности;
- быстрый доступ к информации о ресурсах, расписаниях и услугах способствует более слаженной работе команды;
- автоматизация рутинных задач сокращает время на выполнение стандартных операций и снижает вероятность ошибок;
- автоматизированное управление бронированием и информационной поддержкой повышает уровень удобства для сотрудников;
- систематизация данных о ресурсах и мероприятиях позволяет легко

отслеживать их использование и планировать будущие потребности.

1.1.3 Анализ аналогов

В рамках дипломной работы был проведен сравнительный анализ решений других компаний. Результаты предоставлены в таблице 1.

Таблица 1 - Анализ аналогов

Анализатор	Функционал	Удобство использования	Внешние интеграции	Наличие демо
Robin Powered	Бронирование, Карта офиса	Интуитивно понятный интерфейс	Интеграция с популярными календарями	Да
OfficeSpace Software	Бронирование, Учет мебели	Простой интерфейс	Интеграция с популярными календарями	Да

Robin Powered — это одна из ведущих платформ для управления офисными пространствами, разработанная компанией Robin Technologies, основанной в 2014 году в США. Компания была создана командой специалистов в области технологий и недвижимости с целью оптимизации использования офисных ресурсов. На рисунке 1 представлена начальная страница приложения.

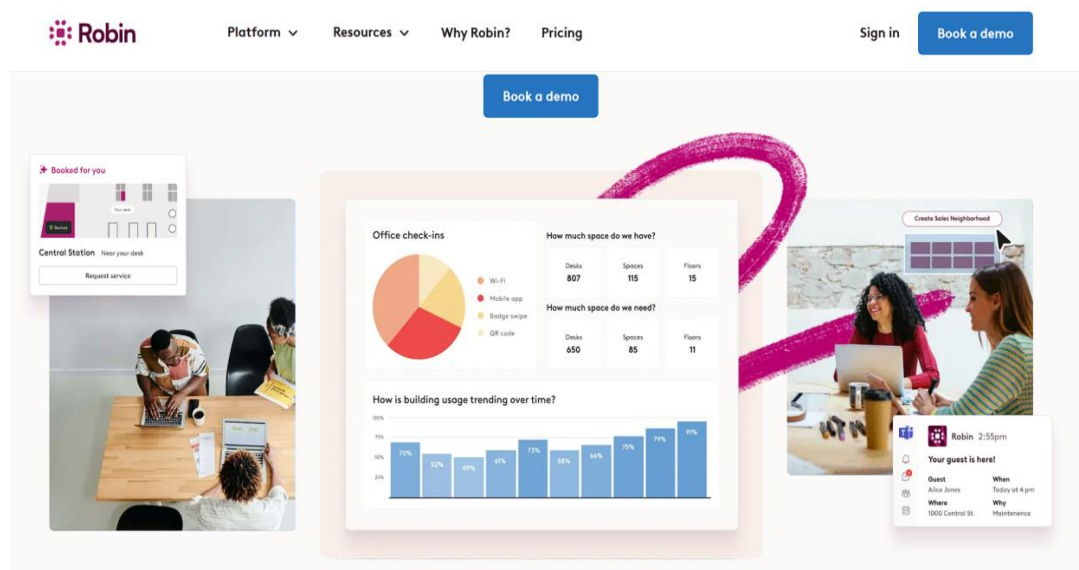


Рисунок 1 – Офисный консьерж Robin Powered

К функциональности Robin Powered можно отнести:

- бронирование переговорных комнат и рабочих мест через веб-интерфейс, мобильное приложение;
- визуальная карта офиса с отображением текущей занятости и доступных ресурсов;
- интеграция с популярными календарями Google Calendar, Microsoft Outlook и корпоративным Slack для автоматического обновления статусов бронирований и уведомлений.

Преимуществами Robin Powered являются:

- удобный и интуитивно понятный интерфейс, доступный как на ПК, так и на мобильных устройствах;
- поддержка мобильных приложений для быстрого доступа к информации в любой точке офиса;
- возможность автоматического обновления данных о занятости и бронированиях;

Недостатками Robin Powered является:

- ограниченный функционал по учету имущества, такого как оборудование или мебель;
- отсутствие встроенного чат-бота для взаимодействия с пользователями, все операции требуют использования интерфейса или интеграций.

OfficeSpace Software — это платформа для управления корпоративными офисами, разработанная компанией OfficeSpace Software Inc., основанной в 2004 году в США.

Начальная страница приложения представлена на рисунке 2.

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		11

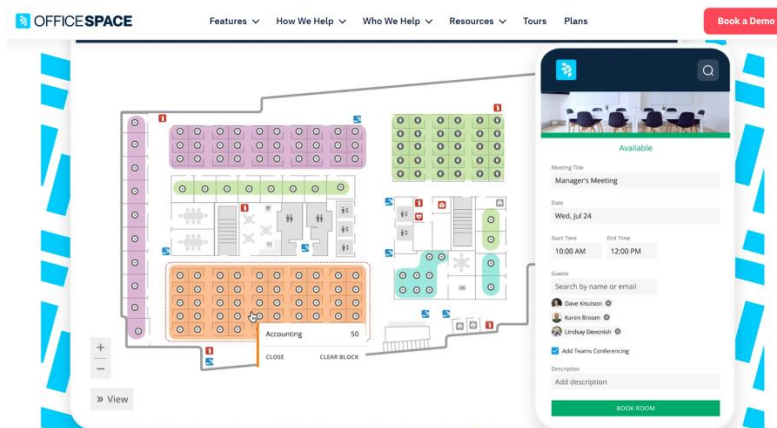


Рисунок 2 – Офисный консьерж OfficeSpace Software

К функциональности OfficeSpace Software можно отнести:

- управление бронированием рабочих мест, переговорных комнат и общих зон;
- учет мебели, оборудования и других ресурсов внутри офиса;
- создание отчетов по использованию пространства, анализ эффективности использования ресурсов;

К преимуществам OfficeSpace Software можно отнести:

- гибкая система отчетности позволяет анализировать использование пространства и планировать его оптимизацию;
- поддержка различных сценариев бронирования и учета ресурсов.

Недостатками OfficeSpace Software являются:

- нет встроенной поддержки чат-ботов или автоматизированных систем взаимодействия с пользователями;
- меньшая интеграция с внешними коммуникационными платформами по сравнению с Robin Powered.

1.1.4 Постановка задачи

Целью разработки является создание веб-сайта для навигации по офисному пространству, управления ресурсами (мебель, техника, расходные материалы) и автоматизации процессов бронирования и учета. Целевой аудиторией будут сотрудники компании, включая ограниченные группы (АО, бухгалтерия, отдел персонала).

Требования к основному функционалу офисного консьержа.

- офисный консьерж должен включать в себя навигацию по этажам;
- офисный консьерж должен включать в себя возможность учета мебели и техники;
- офисный консьерж должен включать в себя возможность учета расходных материалов;
- офисный консьерж должен иметь интеграцию с ботом который позволит пользователю быстро находить нужные ему предметы, просматривать за кем закреплено текущее место.
- данный проект офисного консьержа должен быть гибким для адаптации под разные офисы и доработки функционала.

Требования к интерфейсу:

- навигация простая и интуитивно понятная;
- цветовая индикация занятости рабочих мест;
- проекции этажей с возможностью интерактивного взаимодействия.

В разделе редактирование доступ должен быть ограничен только для определенного круга лиц

На рисунке 3 предоставлена диаграмма вариантов использования.

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		13

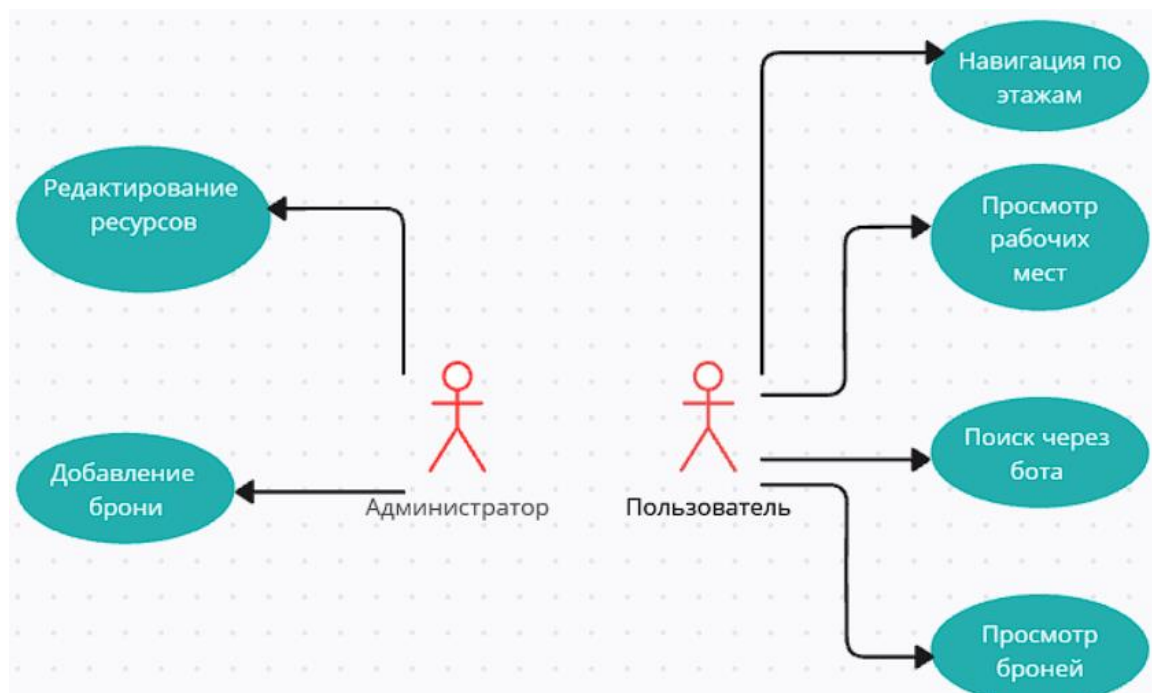


Рисунок 3 – UML диаграмма вариантов использования

1.1.5 Выбор языка программирования

Существует огромное количество языков программирования, которые могут использоваться для разработки серверной части приложений.



Рисунок 4 – Языки программирования

Рассмотрим несколько из них и сравним их по некоторым критериям.

Первым из рассмотренным языком будет Java.

Java является достаточно старым языком программирования, но это не

мешает ей быть одним из самых популярных языков в наше время. Java-приложения запускаются на виртуальной машине, что обеспечивает простоту переноса программы с одного устройства на другое. Он также имеет огромное количество библиотек, которые сильно ускорят разработку приложения.

Плюсы:

- прост в использовании и изучении;
- огромное количество библиотек и фреймворков;
- полная ориентированность на объекты.

Минусы:

- время компиляция и запуск приложения может составлять значительный промежуток времени;
- нет поддержки кода на старых версиях языка.

Вторым из рассмотренных языков будет C#.

C# (C-Sharp) — это современный объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он широко используется для разработки приложений под платформу Windows, а также для создания веб-приложений, игр и мобильных приложений с помощью различных фреймворков и инструментов.

Плюсы:

- прост в изучении и использовании, имеет большой количество уже реализованных функций в стандарте языка;
- обеспечивает высокую производительность и безопасность благодаря строгой типизации;
- поддержка современных парадигм программирования, таких как асинхронное программирование;
- хорошо интегрируется с платформой .NET, что позволяет создавать разнообразные приложения;
- обширная экосистема библиотек и фреймворков, включая ASP.Net для веб-разработки, Unity для разработки игр и Xamarin.

Минусы:

- основная ориентация на платформу Windows и .NET может ограничивать переносимость на другие операционные системы;
- требует установки среды разработки Visual Studio или других IDE, что может быть ресурсоемким;
- в некоторых случаях может быть сложнее оптимизировать производительность относительно других языков программирования низкого уровня.

Третьим языком, который будет рассмотрен Ruby.

Ruby также является один из популярным языком программирования для backend разработки, основанных на фреймворке Ruby on Rails. Он имеет простой и понятный синтаксис и хорошо подходит разработки серверной части приложения.

Плюсы:

- простой и элегантный синтаксис;
- большое количество готовых библиотек;
- эффективная поддержка многопоточности;
- огромное количество фреймворков для создания веб-приложений.

Минусы:

- медленнее, чем Java;
- проблемы с масштабированием.

Последним языком, который будет рассмотрен является одним из самых популярных языков и это – PHP.

PHP является одним из самых распространенных языков программирования, так как является достаточно легким для изучения.

Плюсы:

- легкий и простой язык программирования;
- огромное количество готовых библиотек и фреймворков;
- большое сообщество и множество ресурсов для обучения.

Минусы:

- ограниченная поддержка многопоточности;
- синтаксические особенности могут затруднять отладку.

Каждый из этих языков программирования имеет свои плюсы и минусы, и выбор зависит от конкретных потребностей проекта. Java - оптимальный выбор для крупных и высоконагруженных проектов, Python - для быстрой разработки прототипов, Ruby - для простоты в разработке, PHP - для веб-сайтов, которые работают с базами данных. В любом случае выбор языка зависит от требований конкретного проекта и уровня квалификации разработчиков. Учитывая все факторы, был выбран C#.

1.1.6 Выбор фреймворка для серверной части приложения

Для языка программирования C# существует большое количество различных фреймворков, которые выполняют свои конкретные задачи. Например, такой фреймворк Xamarin, он позволяет адаптировать код под смартфоны, включая системы Android и Ios, но этот фреймворк не подойдет под наши задачи, так как является инструментом для разработки под мобильные устройства.

Также для разработки на C# существует платформа Unity, довольно сильный инструмент для разработки игр, отлично подходит для новичков, встроенные шаблоны, отличную документацию и огромное количество других ресурсов, которые позволяют изучить программирование на этой платформе. Эта платформа специализируется только на разработке игр, поэтому не является той, которую можно выбрать для разработки серверной части веб-приложения.

Довольно востребованным фреймворком при работе с C# является Entity Framework, он позволяет обращаться к базе данных через код не используя SQL, что сильно облегчает интеграцию БД в проект.

ASP.NET Core - это фреймворк, который предоставляет инструменты для многоплатформенной разработки веб-приложений. Он предоставляет

огромное количество ресурсов для создания веб-сайтов с высокой безопасностью, также имеет шаблоны серверной части веб-приложений, что кратно ускоряет разработку проектов. Имеет хорошую структурированную документацию, что в совокупность с шаблонами позволяет новичкам в разработке серверной части на ASP.Net core быстрее начать реализовать идею своего приложения. К тому же компания Microsoft активно развивает и поддерживает эту платформу, регулярно выпускает обновления с различными фишками для разработчиков

Рассмотрев различные фреймворки для разработки на C#, был выбран ASP.Net core. Это фреймворк способен закрыть все потребности, которые могут возникнуть при разработке серверной части приложения. К тому же у него есть ряд значительных преимуществ перед другими фреймворками:

- ASP.Net core является одним из самых высоко-производительных среди веб-фреймворков;

- ASP.Net core является кроссплатформенным, что позволяет разрабатывать не только под Windows, но и под другие платформы, такие как Linux и macOS;

- ASP.Net core отлично интегрируется с другими инструментами от Microsoft, такие как Visual Studio, Azure, SQL Lite и другие, что значительно облегчает и ускоряет разработку проектов;

- одним из плюсов данной платформы является использования языка программирования C# - достаточно мощный инструмент, с огромным количеством современных библиотек и готовых решений. Также нельзя не отметить механизмы защиты от уязвимостей.

1.1.7 Хранение данных

Для организации хранения большого количества данных часто применяют БД. При использовании ASP.Net core в качестве библиотеки для взаимодействия с БД, наилучшим выбором будет использование Entity Framework Core. В текущем проекте объем данных будет небольшим, поэтому

принято решение использовать обычный блокнот(txt) для хранения. Все данные будет сохраняться в блокнот в формате JSON. Это сделано для облегчения записи и передачи данных, так как все данные, которые передаются по http запросам отправляются в формате JSON.

1.1.8 Формат хранения JSON

Формат JSON – это текстовый формат хранения и передачи различных данных, который основан на объектах JS. Этот формат широко применяется при обмене данных между клиентов и сервером. JSON имеется ряд существенных преимуществ:

- любой файлы в формате JSON можно открывать с помощью любого текстового редактора, включая встроенный блокнот в Windows;
- формат JSON имеется достаточно простой синтаксис;
- не зависит от языка программирования и может поддерживаться на любой платформе для разработки.

Исходя из всех его преимуществ выбора хранения в формате JSON является достаточно хорошим и перспективным решением, также не имеется никаких сложностей ни с хранением, ни с передачей, ни с интеграцией с платформами для программирования, ни с каким-либо языком программирования.

1.1.9 Выбор платформы хостинга серверной части и клиентской

В качестве платформы хостинга был выбран сервис AZURE, от компании Microsoft.

Выбор AZURE имеет достаточно преимуществ перед другими платформами для хостинга:

- гибкость и масштабируемость – Azure позволяет масштабировать все ресурсы и сервисы очень легко, в зависимости от потребностей пользователя;
- Azure может взаимодействовать с различными платформ для разработки, поддерживает действительно большое количество современных и

популярных языков программирования. Все это помогает обеспечить гибкость при разработке приложения и его развертывании на платформе Azure;

- экономичность – оплачиваешь только те ресурсы, которые используешь;

- гибридное использование – поддерживает облачные гибридные решения;

- безопасность – предоставляет надежную систему безопасности и некоторые встроенные функции, которые отвечают за безопасность;

- широкий выбор сервисов – огромное количество сервисов, которые подойдут для абсолютных любых задач.

Azure предоставляет достаточно надежное и современное решение как для хранения, так обработки данных.

Для хостинга клиентов части (React приложения) был выбран GitHub Pages, у этого сервиса достаточно много преимуществ:

- бесплатный хостинг – GitHub Pages является бесплатной платформой для развертывания статических сайтов, что является хорошим преимуществом для небольших проектов;

- простота развертывания - развернуть свой сайт достаточно простое действие и не требует каких-то специальных навыков;

- поддержка HTTPS – GitHub Pages предоставляет SSL-сертификаты автоматически;

- адаптивность – развернутые решения адаптируется под все устройства;

- доменные имена – владелец без особых махинаций может менять доменное имя сайта после развертывания.

1.1.10 Выбор фреймворка для реализации графического интерфейса

В настоящее время есть огромное количество различных фреймворков для веб-разработки, самыми популярными из них являются React и Vue

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		20

React – это фреймворк для JavaScript, который основан на раздельном создании UI компонентов, которые позволяют отображать весь пользовательский интерфейс. К тому же каждый отдельный компонент должен иметь свое состояние, которое и будет определять его отображение и поведение при взаимодействии клиента с веб-сайтом. Для создания каких-либо UI компонентов используется JSX – это HTML подобный код, в котором есть возможность использования переменных и некоторых логических выражений. React является достаточно быстрым за счёт того, что использует виртуальный DOM – это копия DOM. Обновление компонентов происходит очень быстро и не затрачивает лишние ресурсы на перерисовку DOM именно из за использования виртуального DOM.

Vue, является достаточно сильным инструментом для работы с UI компонентами, но есть множество особенностей, которыми он отличается от React. Например, Vue позволяет легко изучить фреймворк и быстро начать писать код за счет использования шаблонов HTML-синтаксиса, Vue имеется огромное количество различных плагинов и расширений, которые помогают ускорить разработку и повысить функциональность приложения до максимума [1].

Выбор React для использования с ASP.Net core является достаточно хорошим выбором по нескольким причинам:

- унифицированный хостинг – хостинг в одном приложении ASP.Net core устраняет проблемы с управлением отдельными платформами;
- простая интеграция API – в React и ASP.Net core существует четкое разделение между пользовательским интерфейсом и задачами, которые должны обрабатываться на стороне сервера;
- упрощенное визуальное редактирование - в среде разработки Visual Studio имеется достаточное количество базовых шаблонов серверной части приложения.

1.1.11 Выбор платформы для разработки

Для серверной части был выбран Visual Studio, потому что это мощная интегрированная среда разработки (IDE) от Microsoft, предназначенная для создания приложений на платформах Windows, .NET, C#, ASP.NET и других технологий. Данная платформа имеет огромное преимущество относительно других платформ для разработки программного обеспечения

Плюсы:

- поддержка .NET и ASP.NET: отлично подходит для разработки серверных приложений, API и веб-сервисов;
- инструменты отладки и тестирования: Встроенные средства позволяют быстро находить и исправлять ошибки;
- интеграция с базами данных: легко подключается к SQL Server и другим системам хранения данных;
- поддержка расширений: можно добавлять плагины для расширения функциональности.

Из минус данной платформы можно отметить только то, что полная версия Visual Studio работает только под Windows; есть версия Visual Studio for Mac, но она отличается по функционалу.

1.1.12 Выводы по предпроектному анализу

Выводы по предпроектному анализу:

- проведен анализ и выбраны инструменты для разработки;
- показана актуальность выбранной темы;
- описаны основные типы пользователей;
- описаны функциональные требования к разрабатываемому приложению.

1.2 Проектирование и разработка

Офисный консьерж состоит из клиентского и серверного приложения. Сервер собирает, обрабатывает и хранит информацию, которая приходит от клиента. Клиентами могут быть любые устройства, которые поддерживают браузер.

Приведенное на рисунке 5 программное обеспечение позволяет: упростить рутинную работу офис-менеджер, с помощью интерактивных областей на фоне эскиза офиса, а обычным сотрудникам с помощью чат-бота поможет в нахождении нужных ресурсов, с помощью списка броней для каждой комнаты не пропустить важные совещания или подстроить график своих переговоров.

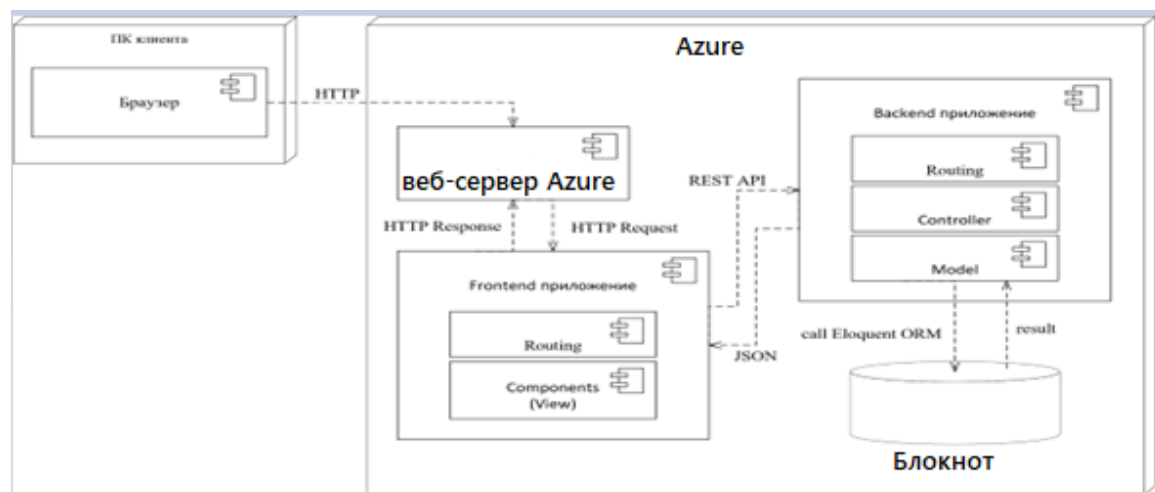


Рисунок 5 – UML диаграмма развертывания

1.2.1 Разработка серверной части приложения

Серверная часть приложения (Backend) разрабатывается в Visual Studio с использованием фреймворка ASP.Net core. Для начала необходимо создать шаблон проекта Web-API ASP.Net core в Visual Studio. Создан базовый вариант серверной части приложения с минимальным функционалом (шаблон проекта). Для проверки можно было перейти по следующему URL «<https://localhost:7065/swagger/index.html>». В данный момент ссылка защищена. Базовая страница приведена на рисунке 6.

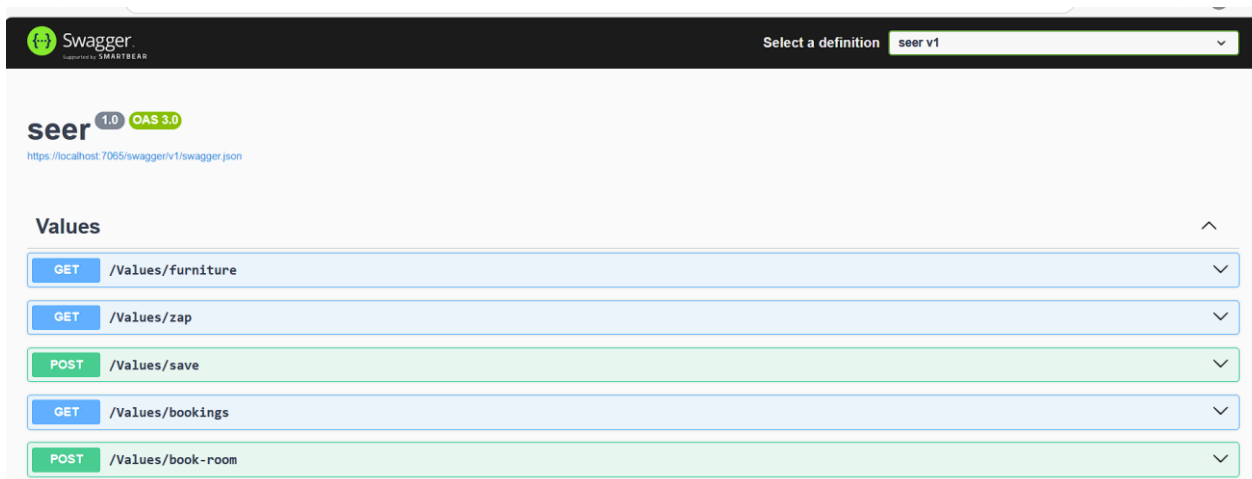


Рисунок 6 – Стартовая страница шаблона Web-API ASP.Net core

Далее было установлено несколько пакетов NuGet:

- BCrypt.Net – этот пакет нужен для хэширования данных, таких как пароли, и другая конфиденциальная информация которая не должна быть в открытом доступе. В случае утечки данных злоумышленникам будет довольно сложно получить первоначальную информацию;

- Microsoft.Identity.Client – Этот пакет содержит двоичные файлы библиотеки аутентификации Microsoft для .NET (MSAL.NET). MSAL.NET упрощает получение токенов из платформы идентификации Microsoft для разработчиков (ранее Azure AD v2.0), которые входят в систему с рабочими и школьными учетными записями, личными учетными записями Microsoft и социальными удостоверениями через Azure AD B2C. Токены предоставляют доступ к Microsoft Cloud API и любому другому API, защищенному платформой идентификации Microsoft. Данная версия Microsoft.Identity.Client поддерживает добавление функций аутентификации к вашим клиентам на основе .NET — .NET, .NET Framework, .NET MAUI. – описание библиотеки написано компанией Microsoft.

Все настройки приложения, включая настройку политики, добавление контроллеров, конфигурации конвейера обработки HTTP-запросов, настройку статической файловой системы и запуск самого сервера происходит в файле

«Program.cs», который приведен ниже.

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddCors

(options =>{ options.AddPolicy

("AllowAllOrigins",

builder => {

    builder.AllowAnyOrigin()

.AllowAnyMethod()

.AllowAnyHeader();

});

});

builder.Services.AddControllers();

builder.Services.AddScoped<IUserService,

UserService>();

builder.Services.AddEndpointsApiExplorer();
```

builder – имя основной переменной в которую мы добавляем службу CORS с политикой, а именно разрешаем любой источник, любые методы (GET, POST и т.д.) и любые заголовки, добавляем все действующие на сервере контроллеры для обработки http-запросов.

```
var app = builder.Build();

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI();

} app.UseCors("AllowAllOrigins");

var uploadsPath =Path.Combine(Directory.

GetCurrentDirectory(), "wwwroot", "image");

if (!Directory.Exists(uploadsPath))
```

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		25

```

{
    Directory.CreateDirectory(uploadsPath);
}
app.UseStaticFiles(new StaticFileOptions
{
    FileProvider = new PhysicalFileProvider(uploadsPath),
    RequestPath = "/image"
});
app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();
app.Run();

```

`app` – имя главной переменной в которой и происходит запуск нашего проекта, инициализация веб сайта с отображением наших запросов и другие базовые настройки проекта

Входной точкой в backend приложение являются запросы от клиента, которые приходят по URL адресу сервера. Запросы обрабатываются механизмом маршрутизации ASP.Net core.

ASP.Net core предоставляет мощный механизм маршрутизации для API. Он позволяет создавать маршруты для обработки запросов к API. Маршруты обрабатываются контроллерами.

Маршруты для API определяются в файле WeatherForecastController.cs. Для задания маршрута используется метод Route: в нем задаем имя. Например, [Route("[controller]")]. Далее указываем HTTP-метод (get, post, put, delete и т.д.) и название, которое и будет являться конечным маршрутом для запроса. Например, [HttpGet("furniture")] задает маршрут к get запросу который будет доступен по маршруту .../furniture.

Контроллеры используются для обработки запросов к API. Методы контроллера должны возвращать данные в формате JSON, так как API работает с данными в этом формате. В ASP.Net core, создание контроллеров можно осуществить с помощью создания класса, который будет наследоваться от

ControllerBase. Каждый отдельный HTTP-запрос является классом, в котором реализована функция, которая выполняется определенные действия, например запись в файл или возврат данных от сервера к клиенту.

В контроллере API происходит обработка полученных данных со стороны клиента, в класс передается в качестве параметра. Например, для обработки GET-запросов класс может принимать аргумент, это будет выглядеть так:

```
public IActionResult GetInventory([FromQuery] string place)
{
    var json = System.IO.File.ReadAllText(FurnitureDataFile,
        Encoding.UTF8);

    var allData = JsonSerializer.Deserialize<Dictionary<string,
        List<InventoryItem>>>(json)

        ?? new Dictionary<string, List<InventoryItem>>();

    if (allData.TryGetValue(place, out var items))
    {
        return Ok(items); }

    return Ok(new List<InventoryItem>());
}
```

Для POST-запросов структура будет выглядеть примерно также. Рассмотрим пример HTTP-запрос save:

```
[HttpPost("save")]
public IActionResult SaveInventory([FromBody]
    SaveRequest request)
{
    if (request == null || request.Items == null)
    {
        return BadRequest(new { error = "Invalid request" });
    }

    var json = System.IO.File.ReadAllText(FurnitureDataFile,
        Encoding.UTF8);

    var allData = JsonSerializer.Deserialize<Dictionary<string,
        List<InventoryItem>>>(json)
```

```

        ?? new Dictionary<string, List<InventoryItem>>();
        allData[request.Place] = request.Items;
        var options = new JsonSerializerOptions
        {
            WriteIndented = true,
            Encoder =
                System.Text.Encodings.Web.JavaScriptEncoder.UnsafeRelaxedJsonEscap
ing
        };
        System.IO.File.WriteAllText(FurnitureDataFile,
            JsonSerializer.Serialize(allData, options),
            Encoding.UTF8);
        return Ok(new { success = true }); }

```

Ниже на рисунке 7 предоставлены диаграмма компонентов.

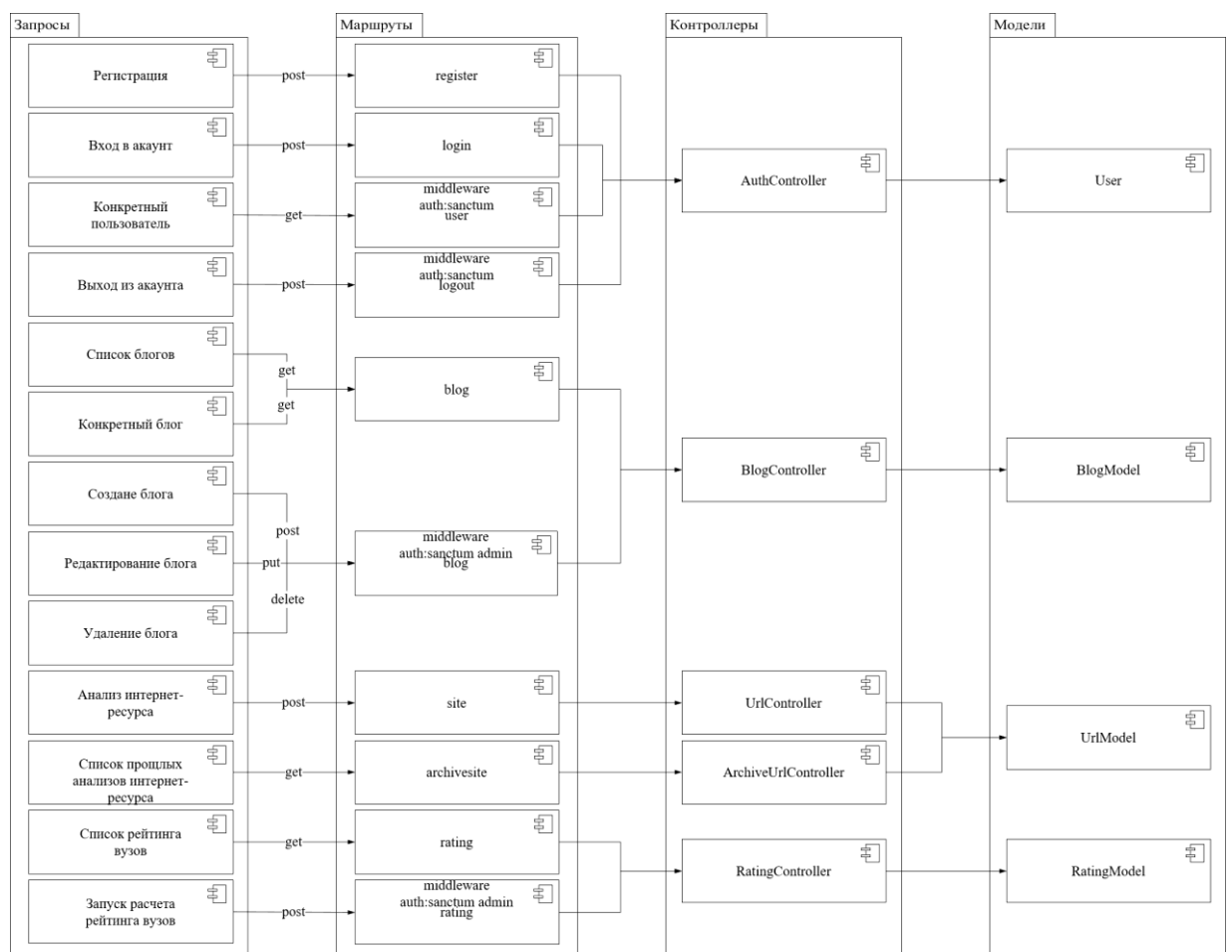


Рисунок 7 – UML диаграмма компонентов

1.2.2 Реализация основных HTTP-запросов

В рамках проекта был разработан REST API, включающий в себя несколько методов для взаимодействия с данными. Ниже рассмотрим каждый из них подробно.

Первым из HTTP-запросов рассмотрим GET-запрос, который расположен по адресу <https://localhost:7065/Values/furniture>. Этот метод предназначен для получения полной информации об интерактивных областях. Он возвращает все записи обо всех областях, например, метод возвращается всю информацию, которая хранится в “furniture.txt” это айди область и вся информация о ней: человек, который закреплен за этой областью или список предметов, которые хранятся в этой области. Также на некоторых этаж есть специальные области с инвентарем, информацию о которых этот метод тоже вернет.

```
[HttpGet("furniture")]
public IActionResult GetFurnitureData()
{
    try
    {
        if (!System.IO.File.Exists(FurnitureDataFile))
        {
            return NotFound(new { error = "Data file not found" });
        }

        var jsonData =
            System.IO.File.ReadAllText(FurnitureDataFile, Encoding.UTF8);
        try
        {
            var data =
                JsonSerializer.Deserialize<Dictionary<string,
                List<InventoryItem>>>>(jsonData);
            return Ok(data);
        }
    }
```

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		29

```

        catch (JsonException)
        {
            return BadRequest(new { error =
"Invalid JSON format in data file" });
        }
    }
}

```

Вторым из HTTP-запросов рассмотрим GET-запрос, который получается информацию о конкретной области и расположен по адресу <https://localhost:7065/Values/zap>. В этот метод со стороны клиента передается номер области, и по этому номеру отправляется информация об этой области обратно клиенту. Код этого HTTP-запроса был представил выше в качестве примера GET-запроса

Третьем будет рассмотрен HTTP-запрос типа POST, он расположен по адресу <https://localhost:7065/Values/save>. Этот метод позволяет редактировать информацию об интерактивных областях и сохраняет все информацию в текстовый файл, храниться все в формате JSON. Код этого HTTP-запроса был представил выше в качестве примера POST-запроса.

На рисунке 8 предоставлен алгоритм обращения к серверу.

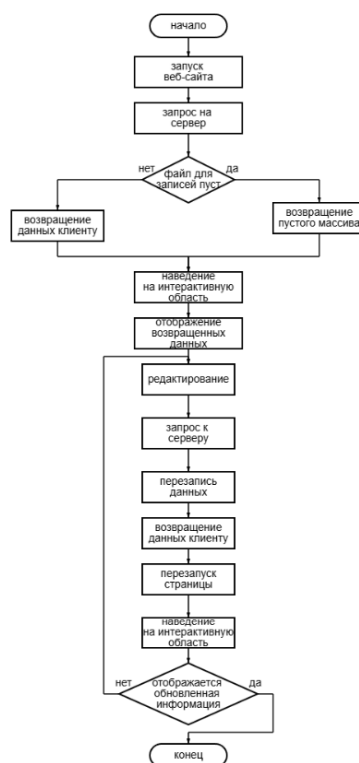


Рисунок 8 – Алгоритм обращения к серверу

1.2.3 Разработка клиентской части приложения

Разработка началась с подготовки окружения. Установлена последняя версия Node.js и создан новый проект React с помощью Create React App. Далее было продумано сколько различных UI компонентов будет в проекте. Всего их три, а именно:

- все, что связано с отображением интерактивных областей;
- редактирование содержимого областей;
- бронирование.

Все UI компоненты инициализируются и применяются в главном файле App.js, который приведен ниже

```
import React, { useState } from "react";
import InteractiveImage from "../InteractiveImage";
import BookingModal from "../BookingModal";
const App = () => {
  const [showBookingModal, setShowBookingModal] =
    useState(false);
  const handleAreaClick = (place) => {
    console.log(`Кликнули на место ${place}`);
  };
  return (
    <div style={styles.appContainer}>
      <div style={styles.appContent}>
        <InteractiveImage onAreaClick={handleAreaClick} />
        <div style={styles.app}>
          <div style={styles.content}>
            <button
              onClick={() => setShowBookingModal(true)}
              style={styles.bookingButton}
            >
              Забронировать переговорную
            <span style={styles.buttonHoverEffect}></span>

```

```

        </button>
      </div>
    </div>
    {showBookingModal && (
      <BookingModal onClose={() =>
setShowBookingModal(false)} />
    )}
  </div>
</div>
);
};
}
};
export default App;

```

Функция `handleAreaClick` обрабатывает любые взаимодействия с интерактивными областями и описана в `InteractiveImage`, при наведении отображается содержимое интерактивной области, при клике открывается область, в которой можно ввести пароль администратора и начать редактирование содержимого в интерактивной области

Под меню для редактирования ресурсов интерактивных областей представлена кнопка по которой можно вызвать начать бронирование переговорных комнат, за это отвечает `<button onClick={() => setShowBookingModal(true)}>` при нажатии на кнопку вызывается функция `onClick`, для пользователя появляется меню с выбором переговорной комнаты, временем начала и временем окончания бронирования, также для удобства выведен список актуальных броней. При выборке комнаты и времени бронирования останется только нажать на кнопку забронировать, тогда данные отправятся на сервер и в списке актуальных броней для текущей комнаты отобразятся актуальные данные, которые были записаны в текстовый файл в формате JSON на стороне сервера.

Ниже на рисунке 9 предоставлена диаграмма классов клиентской части.

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		32

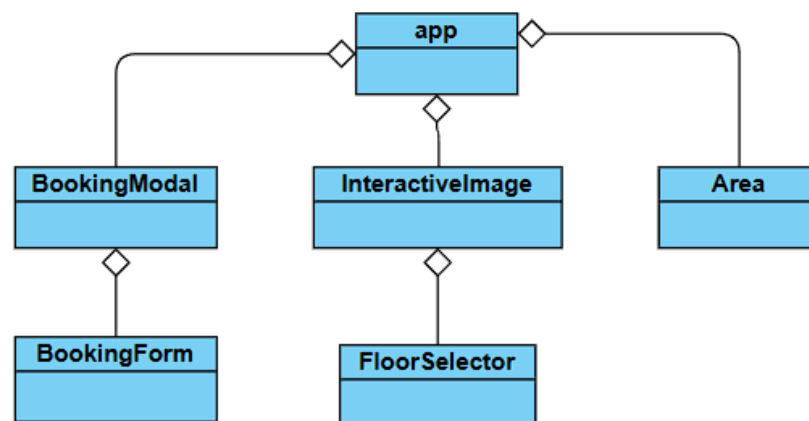


Рисунок 9 – Диаграмма классов клиентской части приложения

По результатам разработки было выполнена реализация всего функционала указанного в задании как для клиентской части приложения, так и для серверной части приложения.

1.3 Анализ результатов

1.3.1 Тестирование API

Для тестирования и отладки API можно воспользоваться программой Postman, терминальной командой curl или же открыть страницу backend приложения .../api. Рисунок 10 главной страницы, которая отображает весь список доступных маршрутов, что упрощает отладку, т.к. сразу видно доступность API и URL- адреса [3].

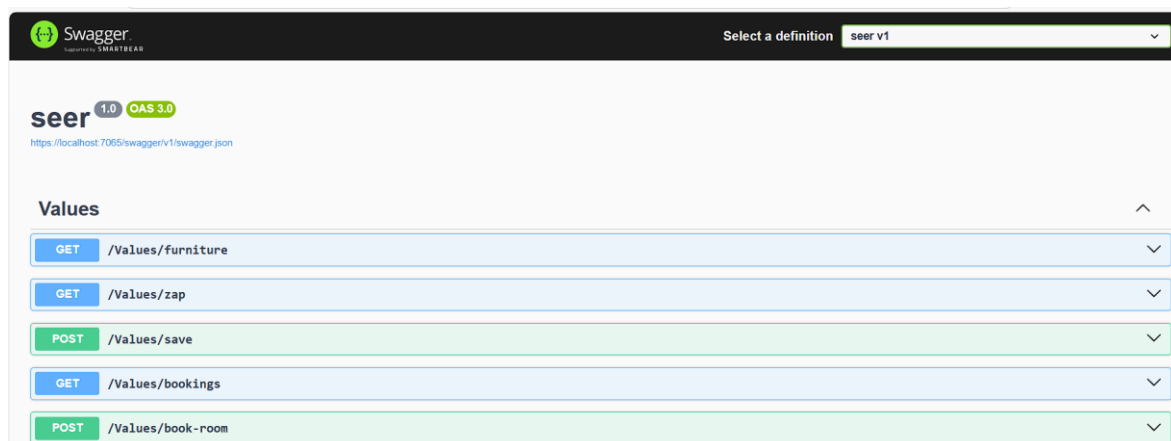


Рисунок 10 – Начальный экран Swagger

Можно протестировать работу с передачей информации от клиента к серверу. На рисунке 11 ниже представлен результат запроса к серверу от клиента.



Рисунок 11 – Результат запроса о получения информации

Далее может на самом сайте приложения увидеть информацию, которая отображается на интерактивных областях после успешного выполнения запроса от клиента к серверу. На рисунке 12 предоставлено получения данных от сервера

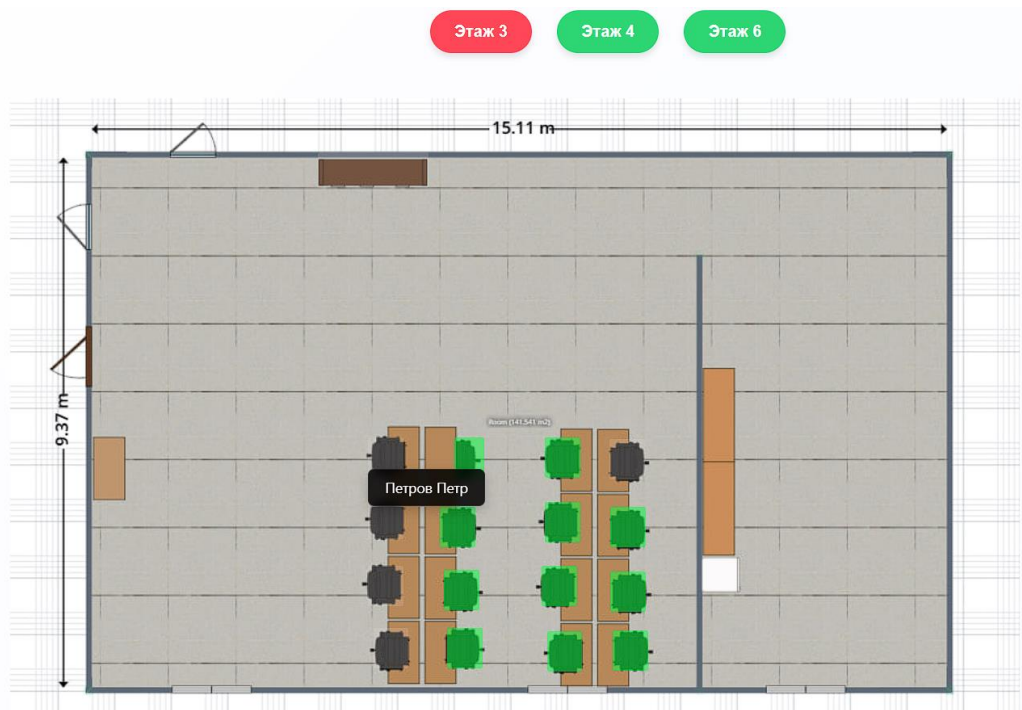


Рисунок 12 – Получены данные с сервера

Теперь можно попробовать запросить список броней. На рисунке 13 ниже представлен успешно полученный список броней.

Комната:

Комната 2

Начало:

дд.мм.гггг --:--

Конец:

дд.мм.гггг --:--

Забронировать

Бронирования для Room2:

21.04.2025, 18:32:00 - 21.04.2025, 22:35:00

22.04.2025, 14:16:00 - 30.04.2025, 19:14:00

Рисунок 13 – Результат повторного запроса блогов

Также доступно редактирование ресурсов, которые закреплены в интерактивных областях, но для этого нужно ввести пароль администратора. Введя пароль администратора, может изменить количество, изменить название, удалить любую позицию или добавить новую, введя названия и нажав на кнопку добавить. Ниже предоставлен рисунок 13, в котором видно меню редактирования после ввода пароля.

Рисунок 13 – Редактирование ресурсов интерактивной области после ввода пароля администратора

Изменив ресурсы в интерактивной области 1 и перезагрузив страницу, увидим, что запросы по API работают корректно, информация изменяется правильно. Это видно на рисунке 14.



Рисунок 14 – После изменений ресурсов в интерактивной области

1.3.2 Тестирование пользовательского интерфейса

В клиентской части приложения протестируем все функции, которые были реализованы. Ниже на рисунке представлена главная страница.

На рисунке 15 предоставлена главная страница приложения.

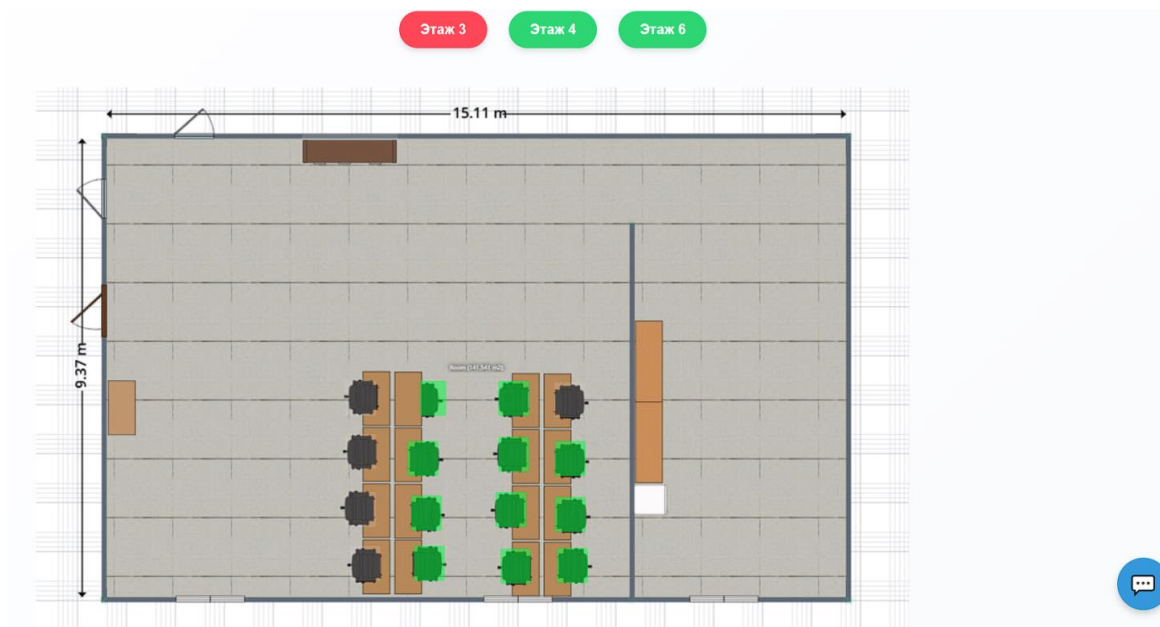


Рисунок 15 – Главная страница

На главной странице в самом верху расположены кнопки для переключения между этажами, чертеж выбранного этажа с интерактивными областями, некоторые из них горят зелеными – это свободные рабочие, не подсвеченные рабочие места – это занятые рабочие места, также мы видим шкафы, которые тоже заполнены. Справа внизу иконка чат-бота, при нажатии можно будет взаимодействовать с ботом. При наведении на интерактивные области мы увидим их содержимое (рисунок 15, рисунок 16).



Рисунок 16 – Эффект при наведении на занятую область

Изм.	Лист	№ докум.	Подпись	Дата

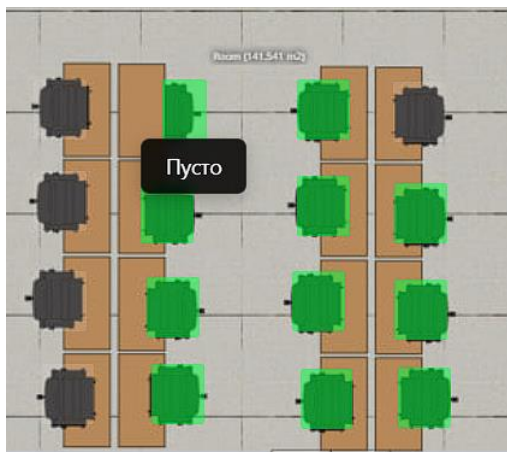


Рисунок 16 – Эффект при наведении на пустую область

При нажатии на интерактивную область для редактирования под чертежом появиться кнопка, нажав на нее появиться меню для редактирования, которое отображает текущую информацию об области, для редактирования нужно ввести пароль. Рисунок 17 до ввода пароля.

Информация о месте 3

Инвентарь для места 3

Введите пароль для редактирования

войти

Рабочее место Иванова Ивана

0 шт.

Закреть

Рисунок 17 – Меню редактирования

После ввода пароля появляется возможно редактировать содержимое области, а именно ввод новой позиции или редактирование старой, обновления применяются после перезагрузки. Рисунок 18 после ввода пароля

Инвентарь для места 3

Рабочее место Иванова Ивана

0

удалить

Новая позиция

добавить

Сохранить изменения

Закреть

Рисунок 18 – Меню редактирования после ввода пароля

Нажав на иконку чат-бота, открывается меню. Ниже представлен рисунок 29 меню чат-бота.

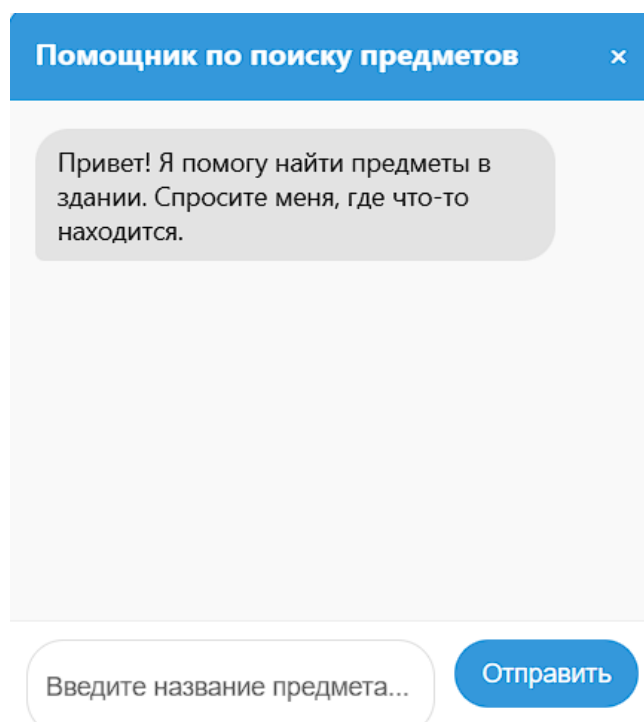


Рисунок 29 – Чат-бот

Введя название предмета и нажав кнопку отправить чат-бот найдет предметы и отправит список, где их можно найти. Также можно найти и сотрудников, которые закреплены за рабочей областью. На рисунке 30 предоставлен ответ от чат-бота.

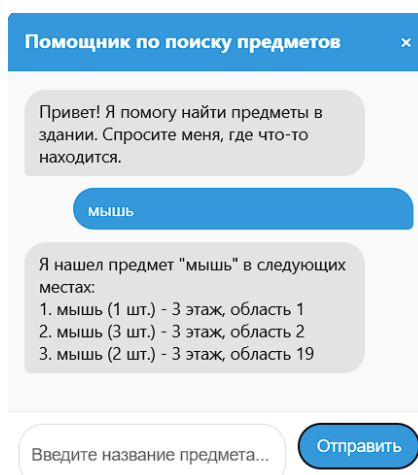


Рисунок 30 – Ответ чат-бота

В самом низу расположена кнопка “Забронировать переговорную”, нажав на нее откроется меню. По умолчанию выбрана комната 1 и для нее приведен список текущих броней, выбрав другую комнату, будет предоставлен список других комнат. Выбрав дату начала и дату окончания бронирования, можно записать новую бронь по кнопке забронировать. Меню бронирования предоставлено на рисунке 31.

Забронировать переговорную

Комната: Комната 1 ▾

Начало: дд.мм.гггг --:-- 📅

Конец: дд.мм.гггг --:-- 📅

Забронировать

Бронирования для Room1:

21.04.2025, 14:27:00 - 21.04.2025, 17:29:00

22.04.2025, 12:53:00 - 23.04.2025, 17:49:00

Рисунок 31 – Меню бронирования

В результаты тестирования все функции клиентской части приложения работают корректно, обращение к серверу по API осуществляется без ошибок

2 Охрана труда

В этом разделе рассмотрим ключевые условия безопасности и здоровья пользователей ПК, которые необходимо соблюдать при работе с программным обеспечением. Также, здесь будут описаны некоторые методы снижения вредных факторов, которые могут негативно отразиться на здоровье пользователей.

2.1 Характеристика потенциальных опасностей и вредностей, которые могут возникнуть при работе с ЭВМ

Каждодневно персональными электронно-вычислительными устройствами пользуются огромное количество людей для решений различных задач, но не смотря на всю их пользу они также могут вызвать ущерб не только для пользователя, но и для окружающих. Для уменьшения риска вреда, связанного с использованием ПК, важно строго соблюдать рекомендованные нормы безопасного использования персональных электронно-вычислительных устройств, которые включают в себя норму проведения времени за компьютером, расположение рабочего места, освещение, а также шума внутри помещения установленным нормам. К основным факторам риска относятся:

- чрезмерная нагрузка на глаза;
- нагрузка на тело, обусловленная неправильным положением за ПК;
- влияние электромагнитного излучения на пользователя.

2.2 Классификации и категорировании помещений, в которых производится разработка проектов

2.2.1 Классификация по взрывопожарной опасности

По пожарной опасности помещения подразделяются на несколько категорий А, Б, В1-В4, Г и Д, а здания - на категории А, Б, В, Г и Д.

Категория рабочих помещений определяется на основании горючих веществ, а также по их пожароопасным свойствам. Ниже приведены некоторые типы помещения и характеристики веществ, находящихся в помещении:

– к «А» относится к помещениям с повышенной взрывопожароопасностью, где присутствует наличие горючих газов или легковоспламеняющихся жидкостей при температуре не превышающей 28 °С. В таких помещениях может образовываться взрывоопасная смесь, которая при возгорании выделит вызвать давление взрыва. Также, в категорию «А» относится ряд веществ и материалов, способных взрываться и гореть при взаимодействии с кислородом, водой или даже друг с другом. Если такие вещества и материалы находится в помещении в превышающем норму количестве, то такое помещение будет считаться помещением с категорией «А»;

– к категории «Б» относят помещения, где существует взрывопожароопасность из-за наличия горючей пыли, волокон, и легковоспламеняющихся жидкостей если температура в помещении будет более 28 °С. При взрыве этих смесей, произойдет избыточный выброс давление, которое может превысить 5 кПа;

– «В1-В4» относятся к группе пожароопасности, здесь находятся трудногорючие жидкости, вещества или материалы, которые могут гореть при столкновении с водой или кислородом;

– «Г» - это категория, к которой относятся помещения с умеренной пожароопасностью, где присутствуют некоторые негорючие вещества или материалы в горячем, раскаленном состоянии;

– «Д» относится к помещениям с пониженной пожароопасностью, здесь присутствуют материалы горячие материалы, но они находятся в холодном состоянии, поэтому вероятность их воспламенения низка.

2.2.2 Классификация по опасности поражения электрическим током

Условия повышенной опасности поражения людей электрическим током:

- при наличии влажной среды в помещении, когда относительная влажность превышает 75%, то это может способствовать усилению проводимости электрического тока, что за собой может повлечь удар током;
- создать условия для небесной изоляции и охлаждения может наличие проводящей пыли, которая оседает на проводах, внутри машин, ПК. Несмотря на то, что такая пыль не способна вызвать пожар или взрыв, она способна вызвать риск получения удара током;
- наличии металлических, земляных, железобетонных или кирпичных оснований, также способствует увеличению риска получить электрический;
- наличие повышенной температуры в помещении, когда температура превышает 35°C продолжительно или 40°C кратковременно, вне зависимости от времени года.

Кроме того, существуют условия особой опасности поражения людей электрическим током, такие как:

- наличие сырости, включающей дождь, снег, частое опрыскивание и покрытие влагой потолка, стен и других поверхностей;
- наличие химически активной среды в помещении, что может увеличить риск поражения электрическим током;
- одновременное наличие двух или более условий повышенной опасности.

Таким образом, на основе этих условий можно выделить три категории опасности помещений: безопасные, с повышенной опасностью и особо опасные.

Для минимизации рисков поражения электрическим током необходимо работать в помещениях, которые соответствуют следующим критериям безопасности:

- температура воздуха должна быть в пределах рекомендованного

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		43

диапазона от 18°C до 20°C, при этом не должна опускаться ниже 30°C;

– помещение должно регулярно проветриваться для поддержания нормального уровня кислорода и уменьшения вероятности возникновения пожара или короткого замыкания;

– влажность воздуха не должна превышать 60%, так как высокий уровень влажности может привести к образованию конденсата на электрических приборах и аппаратах, что увеличивает риск поражения электрическим током.

2.3 Санитарно-гигиенические нормы помещения

Необходимые санитарно-гигиенические нормы могут включать в себя такие параметры, как сухость помещения, температура и количество света. Для обеспечения норм безопасных условий работы, существуют некоторые стандарты, которые необходимо соблюдать при работе, а именно:

– просторное пространство (одно рабочее место не менее 6 квадратных метров на одного человека, объем помещения минимум 20 квадратных метров на одного человека, а высота потолков не менее 3 метров);

– параметры микроклимата (температура 19-21 °C зимой и 20-23°C летом, влажность воздуха в диапазоне от 40% до 60%, и скорость воздуха не более 0,1 метра в секунду);

– освещение (при искусственном 300-500 лк, коэффициент пульсации не более 5% и яркость в диапазоне от 100 до 200 кд/ метр квадратный);

– по шумовым характеристикам есть два уровня эквивалентный уровень шума (не более 50 дБА) и максимальный уровень звука (не более 65дБА).

Были приведены санитарно-гигиенические нормативы, которые соответствуют нормам СанПиН 2.2.2/2.4.1340-03 к компьютерным местам работников.

Заключение

В результате ВКР было реализовано программное обеспечение «Офисный консьерж».

В процессе реализации ВКР получены навыки разработки WEB-приложений с использованием ASP.Net core на языке C#, и JavaScript-библиотеки React.

Были получены навыки работы с форматом JSON, навыки разработки собственного API и backend и frontend приложений как два отдельных проекта, взаимодействующих по REST API.

Во время выполнения ВКР разработаны: алгоритм взаимодействия клиента и сервера, приложения серверной и клиентской части, разработан собственный API,

Механизм взаимодействия приложений backend и frontend – REST API.

Дальнейшие этапы разработки программы предполагают:

- адаптацию решения под другие офисы;
- доработку системы бронирования;
- закрепление мебели и ресурсов с помощью инвентарных номеров;
- добавление личных аккаунтов для работников;
- добавление возможность взаимодействия между работниками.

Список использованных источников

1. Отличие React от Vue. [Электронный ресурс] / Режим доступа: <https://tproger.ru/articles/sravnenie-react-i-vue-chto-obshhego-i-v-chjom-raznica> свободный (дата обращения 02.05.2025).
2. Преимущества использования React с ASP.Net core. [Электронный ресурс] / Режим доступа: <https://www.c-metric.com/blog/using-react-with-asp-net-core/#:~:text=ASP.NET%20Core%20acts%20as,between%20the%20frontend%20and%20backend> свободный (дата обращения 02.05.2025).
3. Типы тестирования API. [Электронный ресурс] / Режим доступа: <https://practicum.yandex.ru/blog/kak-testirovat-api/> свободный (дата обращения 02.05.2023).
4. Официальная документация от Microsoft по ASP.Net core [Электронный ресурс] / Режим доступа: <https://learn.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-9.0> свободный (дата обращения 02.05.2025).
5. Официальная документация по React [Электронный ресурс] / Режим доступа: <https://ru.legacy.reactjs.org/docs/getting-started.html/> свободный (дата обращения 02.05.2025).

Приложение А

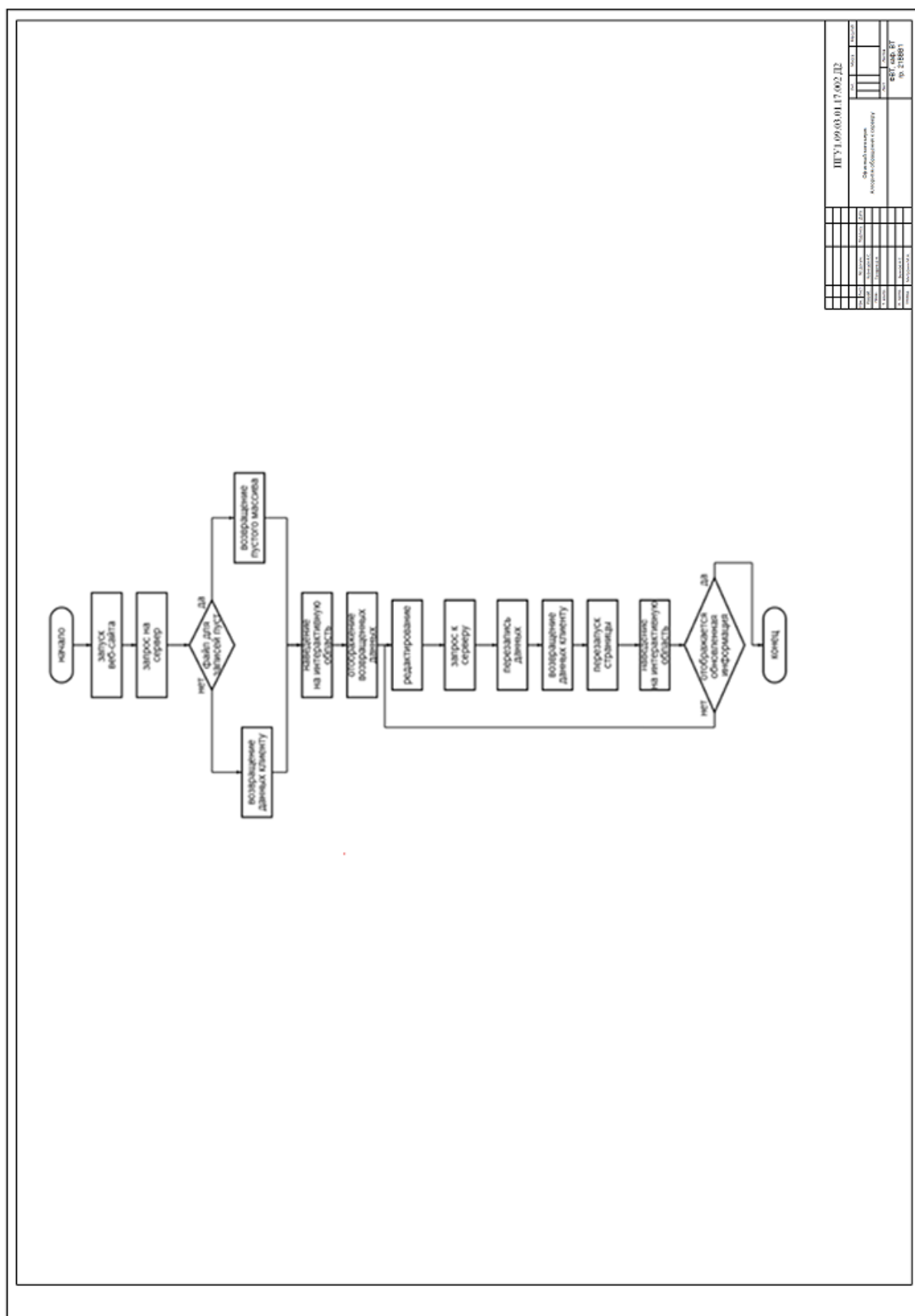
(Обязательное)

Офисный консьерж

UML-диаграммы, ГСА

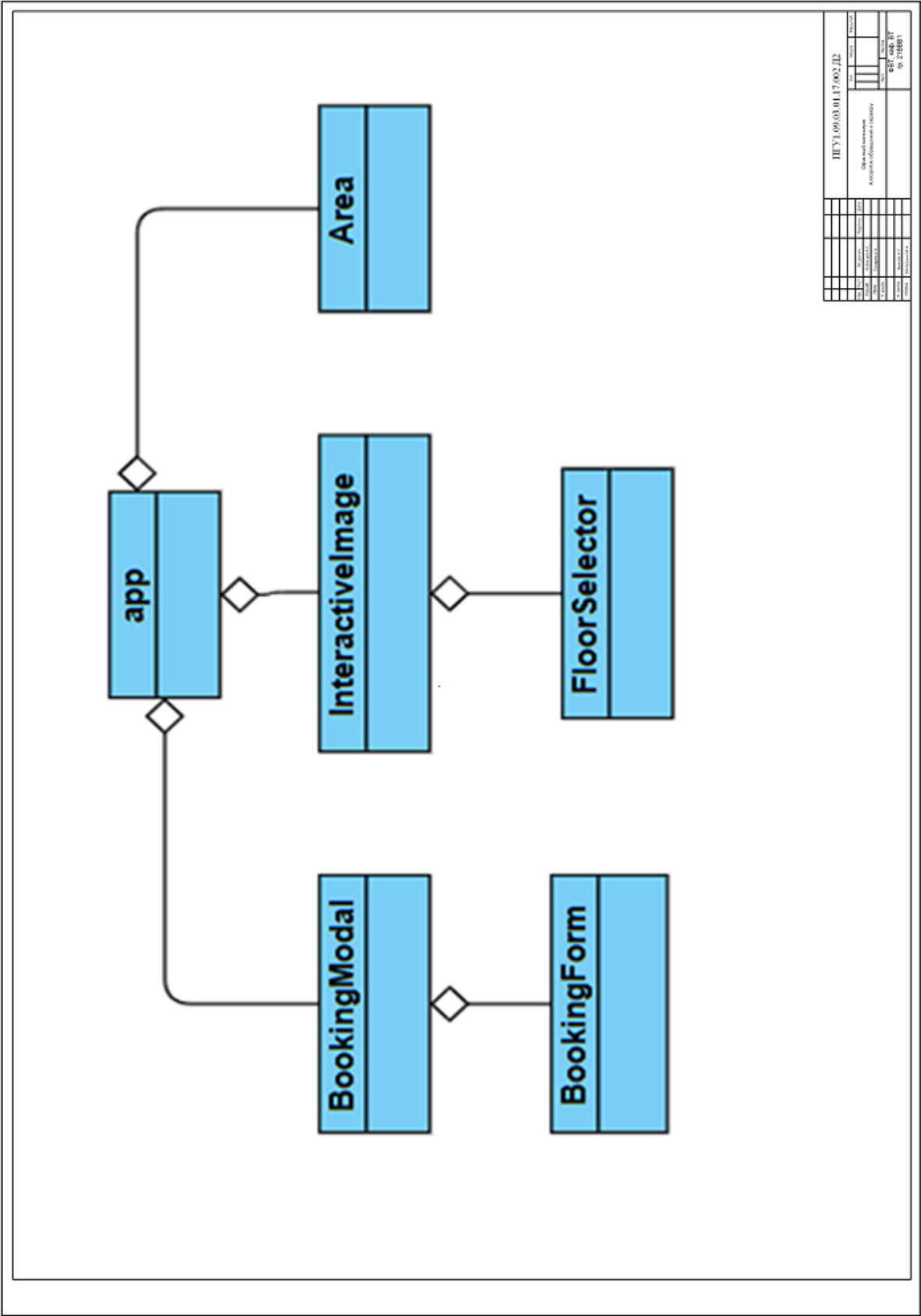
					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		47

Приложение А.1 – Граф-схема алгоритма. Алгоритм обращения к серверу



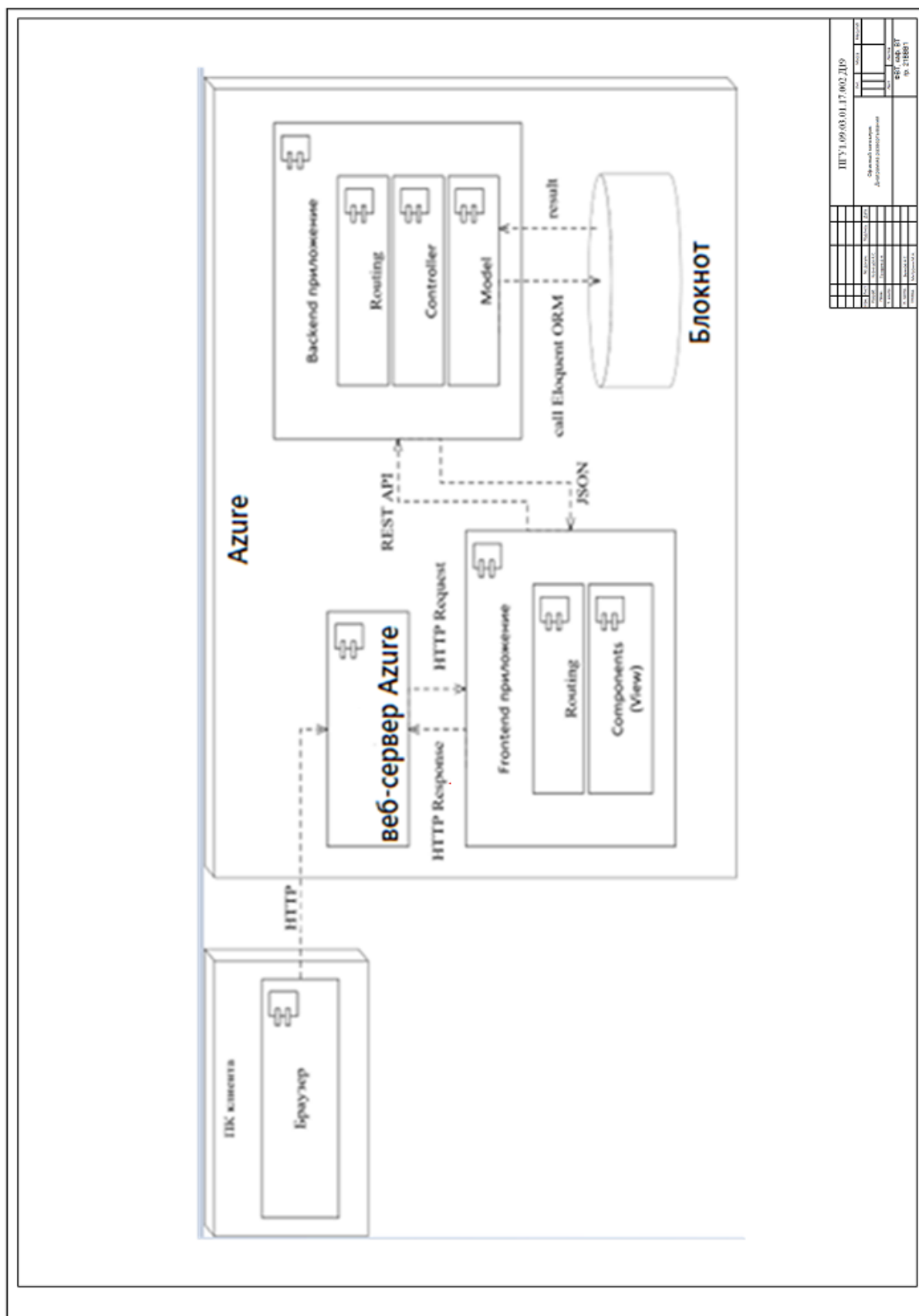
ПГУ 1.09.03.01.15.001 ПЗ		№	Дата	Стр.
Обработка информации		№	Дата	Стр.
Алгоритм обращения к серверу		№	Дата	Стр.
Файл: 1.09.03.01.15.001 ПЗ		№	Дата	Стр.
Итого: 15 стр.		№	Дата	Стр.

Приложение А.2 – Диаграмма классов

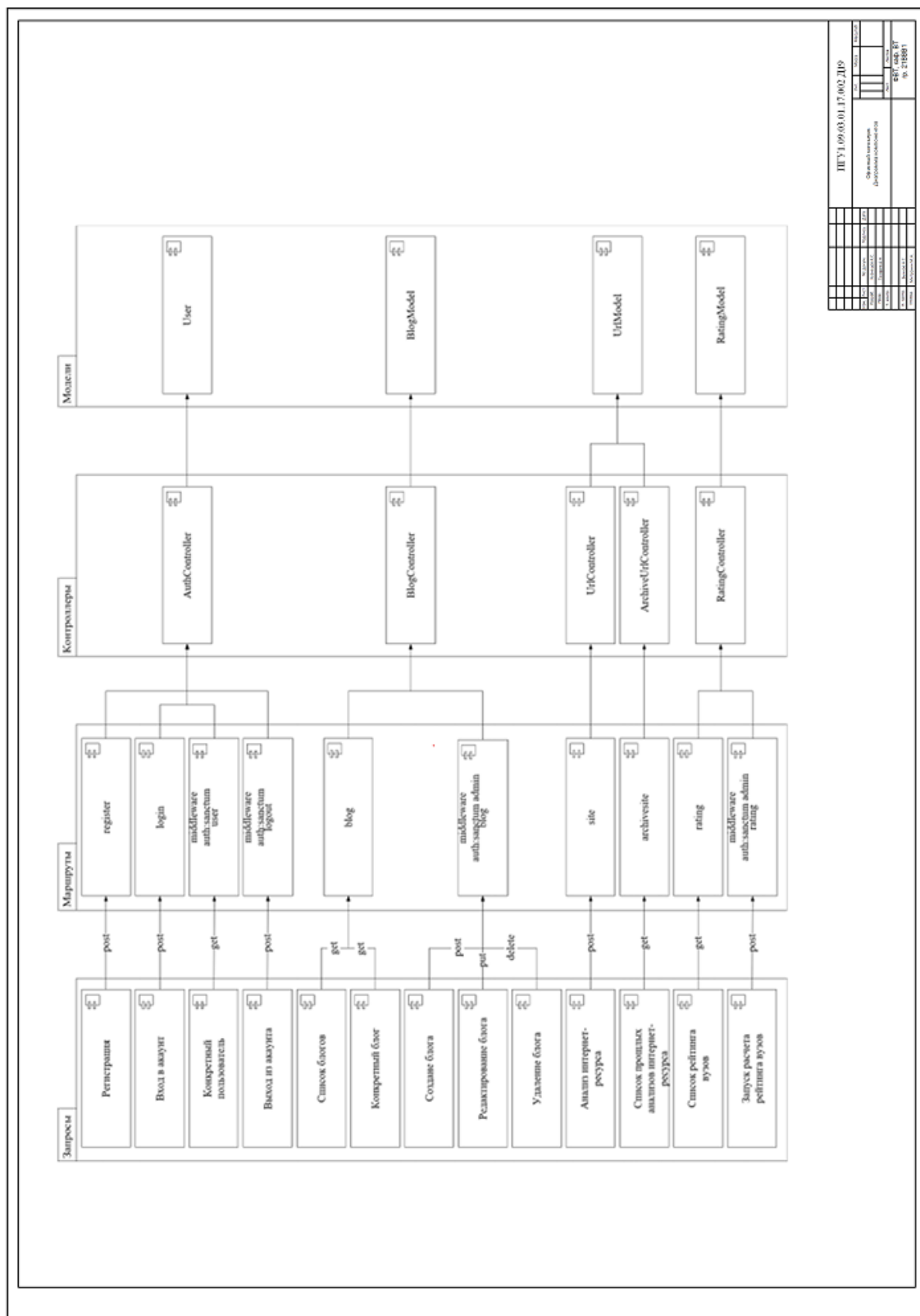


ПГУ1.09.03.01.17.002 Л2									
№ п/п	№ докум.	№ докум.	№ докум.	№ докум.	№ докум.	№ докум.	№ докум.	№ докум.	№ докум.
1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170
171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190
191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220
221	222	223	224	225	226	227	228	229	230
231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250
251	252	253	254	255	256	257	258	259	260
261	262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279	280
281	282	283	284	285	286	287	288	289	290
291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310
311	312	313	314	315	316	317	318	319	320
321	322	323	324	325	326	327	328	329	330
331	332	333	334	335	336	337	338	339	340
341	342	343	344	345	346	347	348	349	350
351	352	353	354	355	356	357	358	359	360
361	362	363	364	365	366	367	368	369	370
371	372	373	374	375	376	377	378	379	380
381	382	383	384	385	386	387	388	389	390
391	392	393	394	395	396	397	398	399	400
401	402	403	404	405	406	407	408	409	410
411	412	413	414	415	416	417	418	419	420
421	422	423	424	425	426	427	428	429	430
431	432	433	434	435	436	437	438	439	440
441	442	443	444	445	446	447	448	449	450
451	452	453	454	455	456	457	458	459	460
461	462	463	464	465	466	467	468	469	470
471	472	473	474	475	476	477	478	479	480
481	482	483	484	485	486	487	488	489	490
491	492	493	494	495	496	497	498	499	500
501	502	503	504	505	506	507	508	509	510
511	512	513	514	515	516	517	518	519	520
521	522	523	524	525	526	527	528	529	530
531	532	533	534	535	536	537	538	539	540
541	542	543	544	545	546	547	548	549	550
551	552	553	554	555	556	557	558	559	560
561	562	563	564	565	566	567	568	569	570
571	572	573	574	575	576	577	578	579	580
581	582	583	584	585	586	587	588	589	590
591	592	593	594	595	596	597	598	599	600
601	602	603	604	605	606	607	608	609	610
611	612	613	614	615	616	617	618	619	620
621	622	623	624	625	626	627	628	629	630
631	632	633	634	635	636	637	638	639	640
641	642	643	644	645	646	647	648	649	650
651	652	653	654	655	656	657	658	659	660
661	662	663	664	665	666	667	668	669	670
671	672	673	674	675	676	677	678	679	680
681	682	683	684	685	686	687	688	689	690
691	692	693	694	695	696	697	698	699	700
701	702	703	704	705	706	707	708	709	710
711	712	713	714	715	716	717	718	719	720
721	722	723	724	725	726	727	728	729	730
731	732	733	734	735	736	737	738	739	740
741	742	743	744	745	746	747	748	749	750
751	752	753	754	755	756	757	758	759	760
761	762	763	764	765	766	767	768	769	770
771	772	773	774	775	776	777	778	779	780
781	782	783	784	785	786	787	788	789	790
791	792	793	794	795	796	797	798	799	800
801	802	803	804	805	806	807	808	809	810
811	812	813	814	815	816	817	818	819	820
821	822	823	824	825	826	827	828	829	830
831	832	833	834	835	836	837	838	839	840
841	842	843	844	845	846	847	848	849	850
851	852	853	854	855	856	857	858	859	860
861	862	863	864	865	866	867	868	869	870
871	872	873	874	875	876	877	878	879	880
881	882	883	884	885	886	887	888	889	890
891	892	893	894	895	896	897	898	899	900
901	902	903	904	905	906	907	908	909	910
911	912	913	914	915	916	917	918	919	920
921	922	923	924	925	926	927	928	929	930
931	932	933	934	935	936	937	938	939	940
941	942	943	944	945	946	947	948	949	950
951	952	953	954	955	956	957	958	959	960
961	962	963	964	965	966	967	968	969	970
971	972	973	974	975	976	977	978	979	980
981	982	983	984	985	986	987	988	989	990
991	992	993	994	995	996	997	998	999	1000

Приложение А.3 – Диаграмма развертывания

[illegible]

Приложение А.4 – Диаграмма компонентов



Приложение Б

(Обязательное)

Офисный консьерж

Листинг программы

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		52

Файл App.js

```
import React, { useState } from "react";

import InteractiveImage from "../InteractiveImage";

import BookingModal from '../BookingModal';

const App = () => {

  const [showBookingModal, setShowBookingModal] = useState(false);

  const handleAreaClick = (place) => {

    console.log(`Кликнули на место ${place}`);

  };

  return (

    <div style={styles.appContainer}>

      <div style={styles.appContent}>

        <InteractiveImage onAreaClick={handleAreaClick} />

        <div style={styles.app}>

          <div style={styles.content}>

            <button

              onClick={() => setShowBookingModal(true)}

              style={styles.bookingButton}>

                Забронировать переговорную

              <span style={styles.buttonHoverEffect}></span>

            </button>

          </div></div>

          {showBookingModal && (

            <BookingModal onClose={() => setShowBookingModal(false)} />

          )}</div></div>);

  };

};
```

Файл area.js

```
import React, { useState, useEffect, useCallback } from 'react';
const API_BASE_URL = 'https://localhost:7065';
const ADMIN_PASSWORD = '123456789';
const InteractiveArea = ({ place, onClose }) => {
  const [menuVisible, setMenuVisible] = useState(false);
  const [items, setItems] = useState([]);
  const [newItemName, setNewItemName] = useState('');
  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState(null);
  const [isEditMode, setIsEditMode] = useState(false);
  const [passwordInput, setPasswordInput] = useState('');
  // загрузка данных
  const fetchData = useCallback(async () => {
    try {
      setIsLoading(true);
      setError(null);
      const response = await fetch(
        `${API_BASE_URL}/Values/zap?place=${encodeURIComponent(place)}`);
      if (!response.ok) throw new Error(`Ошибка сервера: ${response.status}`);
      const data = await response.json();
      setItems(Array.isArray(data) ? data : []);
    } catch (error) {
      console.error('Ошибка загрузки:', error);
      setError(`Ошибка загрузки: ${error.message}`);
    } finally {setIsLoading(false);} }, [place]);
  useEffect(() => {if (menuVisible) fetchData();
}, [menuVisible, fetchData]);
  // сохранение данных на сервере
  const saveData = useCallback(async () => {
    try {setIsLoading(true);setError(null);
      const response = await fetch(`${API_BASE_URL}/Values/save`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ place, items })),});
```

```

    if (!response.ok) {
      const errorText = await response.text();
      throw new Error(errorText || `Ошибка ${response.status}`);
    }
  } catch (error) {
    console.error('Ошибка сохранения:', error);
    setError(`Ошибка сохранения: ${error.message}`);
    throw error;
  } finally {
    setIsLoading(false);
  }
}, [place, items]);
// проверка пароля
const checkPassword = () => {
  if (passwordInput === ADMIN_PASSWORD) {
    setIsEditMode(true);
    setPasswordInput('');
    setError(null);
  } else {
    setError('Неверный пароль');
  }
};
// изменение наличия техники
const handleCountChange = (index, value) => {
  const newValue = Math.max(0, parseInt(value) || 0);
  setItems(prev => prev.map((item, i) =>
    i === index ? { ...item, count: newValue } : item
  ));
};
const handleNameChange = (index, value) => {
  setItems(prev => prev.map((item, i) =>
    i === index ? { ...item, name: value } : item
  ));
};
const addNewItem = () => {
  const trimmedName = newItemName.trim();

```

```

        if (trimmedName) {
            setItems(prev => [...prev, { name: trimmedName, count: 0 }]);
            setNewItemName('');
        }
    };

    const removeItem = (index) => {
        setItems(prev => prev.filter((_, i) => i !== index));
    };

    const handleSave = async () => {
        try {
            await saveData();
            await fetchData();
        } catch {
        }
    };

    return (
        <div style={{ position: 'relative' }}>
            <div
                onClick={() => setMenuVisible(!menuVisible)}
                style={styles.menuButton}>
                {isLoading ? 'Загрузка...' : `Информация о месте ${place}`}
                {error && <div style={styles.error}>{error}</div>}
            </div>
            {menuVisible && (
                <div style={styles.menuContainer}>
                    <h3 style={styles.title}>Инвентарь для места {place}</h3>
                    {!isEditMode && (
                        <div style={styles.passwordSection}>
                            <input
                                type="password"
                                value={passwordInput}
                                onChange={(e) => setPasswordInput(e.target.value)}
                                placeholder="Введите пароль для редактирования"
                                style={styles.passwordInput}
                                onKeyPress={(e) => e.key === 'Enter' &&

```

```

checkPassword() } />

        <button
            onClick={checkPassword}
            style={styles.passwordButton}
        >войти
        </button>

    </div>)}

    { /* отображение наличия техники */ }

    { items.length === 0 ? (
        <p style={styles.noData}>Нет данных</p>
    ) : (
        items.map((item, index) => (
            <div key={index} style={styles.itemRow}>
                {isEditMode ? (
                    <input
                        type="text"
                        value={item.name}
                        onChange={(e) => handleNameChange(index,
e.target.value)}
                        style={styles.nameInput}
                    />
                ) : (
                    <div style={styles.itemName}>{item.name}</div>)}
                {isEditMode ? (
                    <input
                        type="number"
                        min="0"
                        value={item.count}
                        onChange={(e) => handleCountChange(index,
e.target.value)}
                        style={styles.countInput}
                    />
                ) : (
                    <div style={styles.itemCount}>{item.count}
шт.</div>

```

```

    ) }
    {isEditMode && (
        <button
            onClick={() => removeItem(index)}
            style={styles.deleteButton}>
            Удалить
        </button> ) }
    </div>
    ) ) }
    { /* форма добавления */ }
    {isEditMode && (
        <div style={styles.addSection}>
            <input
                type="text"
                value={newItemName}
                onChange={(e) => setNewItemName(e.target.value)}
                placeholder="Новая позиция"
                style={styles.newItemInput}
                onKeyPress={(e) => e.key === 'Enter' && addNewItem()}
            /> <button
                onClick={addNewItem}
                disabled={!newItemName.trim()}
                style={styles.addButton}
            >Добавить</button></div>
        ) } { /* сохранение */ }
        {isEditMode && ( <button
            onClick={handleSave}
            style={styles.saveButton}
            disabled={isLoading}
            >{isLoading ? 'Сохранение...' : 'Сохранить изменения'}
        </button>
        ) } <button onClick={onClose}
            style={styles.closeButton}>Закрыть
        </button></div> ) </div> ) ; ;

```

```

    )}{isEditMode && (
      <button onClick={() => removeItem(index)}
        style={styles.deleteButton}
      >Удалить</button>)}</div>))){/* форма
добавления */}{isEditMode && (<div style={styles.addSection}>
  <input type="text" value={newItemName}
    onChange={(e) => setNewItemName(e.target.value)}
    placeholder="Новая позиция"
    style={styles.newItemInput}
    onPress={(e) => e.key === 'Enter' &&
addNewItem() }/><button
  onClick={addNewItem}
  disabled={!newItemName.trim()}
  style={styles.addButton}
  >Добавить</button></div>
)}{/* сохранение */}
{isEditMode && ( <button
  onClick={handleSave}
  style={styles.saveButton}
  disabled={isLoading}
  >{isLoading ? 'Сохранение...' : 'Сохранить изменения'}
</button>)}<button onClick={onClose}
  style={styles.closeButton}>Закрыть
</button></div>)}</div>);};

```

Файл BookingModal.js

```

import React, { useState, useEffect } from 'react';
import axios from 'axios';
import './BookingForm.css';
const API_BASE_URL = 'https://localhost:7065';
const BookingForm = () => {
  const [roomId, setRoomId] = useState('Room1');
  const [startTime, setStartTime] = useState('');
  const [endTime, setEndTime] = useState('');
  const [bookings, setBookings] = useState([]);
  const [loading, setLoading] = useState(false);

```

```

// Загружаем бронирования при изменении выбранной комнаты
useEffect(() => {if (roomId) {fetchBookings();}
}, [roomId]);const fetchBookings = async () => {
    setLoading(true);try {
const response = await
axios.get(`${API_BASE_URL}/Values/bookings?roomId=${room
Id}`);setBookings(response.data);

    } catch (error) {console.error('Ошибка при загрузке
бронирований:', error);
alert('Не удалось загрузить бронирования');

    } finally {setLoading(false);}};
const handleSubmit = async (e) => {
    e.preventDefault();if (!startTime || !endTime) {
        alert('Пожалуйста, укажите даты начала и конца
бронирования');return;}const bookingData = {
    roomId,startTime:
new Date(startTime).toISOString(),
    endTime: new Date(endTime).toISOString(),};

    try {await
axios.post(`${API_BASE_URL}/Values/book-room`,
bookingData);
    alert('Комната успешно забронирована!')
;fetchBookings();

    } catch (error) {
        console.error('Ошибка при бронировании:', error);
        alert(`Не удалось забронировать комнату:
${error.response?.data?.message || error.message}`);}};

const formatDate = (dateString) => {
    const date = new Date(dateString);
    return date.toLocaleString('ru-RU');};return (
<div className="container">
    <form onSubmit={handleSubmit}><div>
        <label>Комната:</label>

```

```

        <select value={roomId} onChange={ (e) =>
setRoomId(e.target.value)} >
            <option value="Room1">Комната 1</option>
            <option value="Room2">Комната 2</option>
            <option value="Room3">Комната 3</option>
        </select></div><div>
        <label>Начало:</label><input
            type="datetime-local"
            value={startTime}
            onChange={ (e) =>
setStartTime(e.target.value)} required /></div><div>
        <label>Конец:</label>
        <input type="datetime-local"
value={endTime} onChange={ (e) =>
setEndTime(e.target.value)}
            required /></div><button
type="submit">Забронировать
</button></form></div>

<h3>Бронирования для {roomId}</h3>
{loading ? (<p>Загрузка...</p>) : (
    <ul>{bookings.length > 0 ? (
        bookings.map((booking, index) => (
            <li key={index}>
                {formatDate(booking.startTime)}
- {formatDate(booking.endTime)}</li>))) : (<li>Нет
бронирований для этой комнаты
</li>)}</ul>)}</div></div>);};
export default BookingForm;

```

Файл InteractiveImage.js

```

import React, { useState, useEffect } from 'react';
import InteractiveArea from './area';
import styled, { keyframes } from 'styled-components';
const API_BASE_URL = 'https://localhost:7065';
const ADMIN_PASSWORD = '123456789';

```

```

const InteractiveImage = () => {
  const [activeImageIndex, setActiveImageIndex] = useState(0);
  const [isInteractiveAreaVisible, setInteractiveAreaVisible] =
useState(false);
  const [selectedPlace, setSelectedPlace] = useState('');
  const [furnitureData, setFurnitureData] = useState({});
  const [hoverInfo, setHoverInfo] = useState(null);
  const [showRoomInventory, setShowRoomInventory] = useState(false);
  const [currentRoom, setCurrentRoom] = useState('');
  const [roomItems, setRoomItems] = useState([]);
  const [isEditMode, setIsEditMode] = useState(false);
  const [passwordInput, setPasswordInput] = useState('');
  const [showInventoryButtons, setShowInventoryButtons] =
useState(false);
  const [chatOpen, setChatOpen] = useState(false);
  const [messages, setMessages] = useState([
    { text: "Привет! Я помогу найти предметы в здании. Спросите меня, где
что-то находится.", sender: 'bot' }
  ]);
  const [inputMessage, setInputMessage] = useState('');
  const [cellColors, setCellColors] = useState([]);
const images = [
  './планы этажей/планы этажей/floor3.jpg',
  './планы этажей/планы этажей/floor4.jpg',
  './планы этажей/планы этажей/floor6.jpg',];
const interactiveAreas = [
  [
    { id: '1', top: '336px', left: '83px', width: '30px', height:
'63px' },
    { id: '2', top: '54px', left: '305px', width: '110px', height:
'35px' },
    { id: '3', top: '336px', left: '355px', width: '35px', height:
'40px' },
    { id: '4', top: '397px', left: '355px', width: '35px', height:
'40px' },
    { id: '5', top: '463px', left: '355px', width: '35px', height:
'40px' },
  ]

```

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		62

```

{ id: '6', top: '525px', left: '360px', width: '35px', height:
'40px' },
    { id: '7', top: '336px', left: '440px', width: '30px', height:
'40px' },
    { id: '8', top: '405px', left: '426px', width: '35px', height:
'40px' },
    { id: '9', top: '467px', left: '430px', width: '35px', height:
'40px' },
    { id: '10', top: '525px', left: '433px', width: '35px', height:
'40px' },
    { id: '11', top: '336px', left: '530px', width: '35px', height:
'40px' },
    { id: '12', top: '400px', left: '530px', width: '35px', height:
'40px' },
    { id: '13', top: '463px', left: '527px', width: '35px', height:
'40px' },
    { id: '14', top: '528px', left: '532px', width: '35px', height:
'40px' },
    { id: '15', top: '338px', left: '594px', width: '35px', height:
'40px' },
    { id: '16', top: '405px', left: '595px', width: '35px', height:
'40px' },
    { id: '17', top: '469px', left: '595px', width: '35px', height:
'40px' },
    { id: '18', top: '527px', left: '597px', width: '35px', height:
'40px' },
    { id: '19', top: '454px', left: '685px', width: '38px', height:
'36px' },
    ],[
        { id: '20', top: '50px', left: '100px', width: '200px', height:
'200px' },
        { id: '21', top: '300px', left: '150px', width: '100px', height:
'100px' },
    ],[
        { id: '22', top: '60px', left: '200px', width: '150px', height:
'150px' },
        { id: '23', top: '400px', left: '250px', width: '80px', height:
'80px' },],],];

```

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		63

```

useEffect(() => {
    const fetchFurnitureData = async () => {
        try {
            const response = await
fetch(`${API_BASE_URL}/Values/furniture`);
            if (!response.ok) throw new Error('Network response was not
ok');
            const data = await response.json();
            setFurnitureData(data);
        } catch (error) {
            console.error('Ошибка при загрузке данных о мебели:', error);
        }
    };
    fetchFurnitureData();
}, []);

const handleAreaClick = (place) => {
    setSelectedPlace(place);
    setInteractiveAreaVisible(true);
};

const handleButtonClick = (index) => {
    setActiveImageIndex(index);
    setInteractiveAreaVisible(false);
    setHoverInfo(null);
    setShowInventoryButtons(true);
};const handleRoomButtonClick = (floor) => {
    setCurrentRoom(floor);
    setRoomItems(furnitureData[floor] || []);
    setShowRoomInventory(true);
    setIsEditMode(false);
    setPasswordInput('');
};

```

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		64

```

const handleAreaMouseEnter = (areaId) => {
  const items = furnitureData[areaId];
  if (items && items.length > 0) {
    const info = items.map(item =>
      `${item.name || item.Name}${(item.count || item.Count) > 0
? ` (${item.count || item.Count} шт.)` : ''}`
    ).join(', ');
    setHoverInfo({ id: areaId, content: info });
  } else {
    setHoverInfo({ id: areaId, content: "Пусто" });});
const handleAreaMouseLeave = () => {
  setHoverInfo(null);});
const checkPassword = () => {
  if (passwordInput === ADMIN_PASSWORD) {
    setIsEditMode(true);});
const handleSaveRoomItems = async () => {
try {
  const response = await fetch(`${API_BASE_URL}/Values/save`, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      place: currentRoom,
      items: roomItems.filter(item => item.name.trim() !== '')
    }),
  });if (!response.ok) throw new Error('Ошибка сохранения');
  const updatedData = { ...furnitureData, [currentRoom]:
roomItems };
  setFurnitureData(updatedData);
  setShowRoomInventory(false);
} catch (error) {
  console.error('Ошибка:', error);});

```

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		65

```

const handleItemChange = (index, field, value) => {
  const updated = [...roomItems];
  updated[index][field] = field === 'count' ? parseInt(value) || 0
: value;
  setRoomItems(updated);
  setFurnitureData(prevData => ({
    ...prevData,
    [selectedPlace]: updated
  }));
};
useEffect(() => {
  const newColors = roomItems.map(item => {
    return item.name.trim() === '' ? 'red' : 'green'; // Красный для
пустой ячейки и зеленый для заполненной
  });
  setCellColors(newColors);
}, [roomItems]);
const addNewItem = () => {
  setRoomItems([...roomItems, { name: '', count: 0 }]);
};
const removeItem = (index) => {
  const updated = [...roomItems];
  updated.splice(index, 1);
  setRoomItems(updated);
};const getTooltipPosition = (areaId) => {
  const area = interactiveAreas[activeImageIndex].find(a => a.id
=== areaId);
  if (!area) return { top: '0', left: '0' };
  return {
    top: `${parseInt(area.top) - 30}px`,
    left: area.left};};
const findItem = (itemName) => {

```

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		66

```

const results = [];

// Определяем этаж

const floorMapping = {

    '3 этаж': ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10',
'11', '12', '13', '14', '15', '16', '17', '18', '19'],

    '4 этаж': ['20', '21'],

    '6 этаж': ['22', '23']

};

Object.entries(furnitureData).forEach(([areaId, items]) => {

    items.forEach(item => {

        if (item.name &&
item.name.toLowerCase().includes(itemName.toLowerCase())) {

            let floor = '';

            for (const [key, value] of Object.entries(floorMapping))
{

                if (value.includes(areaId)) {

                    floor = key;

                    break;

                }

            }

            results.push({

                item: item.name,

                count: item.count,

                area: areaId,

                floor: floor

            });

        }

    });

});

return results;

};

```

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		67

```

const handleSendMessage = () => {
  if (!inputMessage.trim()) return;
  const userMessage = { text: inputMessage, sender: 'user' };
  setMessages(prev => [...prev, userMessage]);
  setInputMessage('');
  const foundItems = findItem(inputMessage);
  let botResponse;
  if (foundItems.length === 0) {
    botResponse = `Я не нашел предмет "${inputMessage}" в здании.`;
  } else {
    botResponse = `Я нашел предмет "${inputMessage}" в следующих
местах:\n`;
    foundItems.forEach((item, index) => {
      if(item.count === 0){
        botResponse += `\n${index + 1}. ${item.item} - ${item.floor},
область ${item.area}`;
      }else{
        botResponse += `\n${index + 1}. ${item.item} (${item.count}
шт.) - ${item.floor}, область ${item.area}`;
      }
    });
    setTimeout(() => {
      setMessages(prev => [...prev, { text: botResponse, sender: 'bot'
}]);
    }, 800);
  };
const handleKeyPress = (e) => {
  if (e.key === 'Enter') {
    handleSendMessage();
  }
};return (
<div>
  <ButtonContainer>
    {['Этаж 3', 'Этаж 4', 'Этаж 6'].map((label, index) => (
      <FloorButton
        key={index}
        onClick={() => handleButtonClick(index)}
        active={activeImageIndex === index}>

```

```

{label}

    </FloorButton>))}

</ButtonContainer>

<ImageWrapper>

<ImageContainer>

    <img

        src={images[activeImageIndex]}

        alt="Interactive"

        style={{ width: "1000px", height: "auto", display: 'block' }}

    />

    {interactiveAreas[activeImageIndex].map((area) => {
const items = furnitureData[area.id];
const isEmpty = !items || items.length === 0;
return (<divkey={area.id}

    onClick={() => handleAreaClick(area.id)}

    onMouseEnter={() => handleAreaMouseEnter(area.id)}

    onMouseLeave={handleAreaMouseLeave}

    style={{

        position: 'absolute',

        top: area.top,

        left: area.left,

        width: area.width,

        height: area.height,

        cursor: 'pointer',

        backgroundColor: isEmpty ? 'rgb(0, 255, 60)' : 'rgba(237,
223, 223, 0.3)',

        transition: 'background-color 0.2s',

        borderRadius: '2px',

        opacity: '50%',

    }}

    />

    );}})}

```

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		69

```

<div>
  <ChatToggleButton
onClick={() => setChatOpen(true)}
style={{
  position: 'fixed',
  bottom: '22px',
  right: '62px',
  zIndex: 1000,
}}
>
</ChatToggleButton>
  { /* Чат-бот */ }
  <ChatBotContainer isOpen={chatOpen}>
    <ChatHeader onClick={() => setChatOpen(false)}>
      Помощник по поиску предметов
    </ChatHeader>
    <ChatMessages>
      {messages.map((message, index) => (
        <Message key={index} className={message.sender}>
          {message.text.split('\n').map((line, i) => (
            <div key={i}>{line}</div>
          ))}</Message>))}
    </ChatMessages>
    <ChatInputContainer>
      <ChatInput
        type="text"
        value={inputMessage}
        onChange={(e) => setInputMessage(e.target.value)}
        onKeyDown={handleKeyPress}
        placeholder="Введите название предмета..." />

```

```

<SendButton onClick={handleSendMessage}>Отправить</SendButton>

    </ChatInputContainer>

  </ChatBotContainer>

</div>{hoverInfo && (
  <Tooltip style={getTooltipPosition(hoverInfo.id)}>
    {hoverInfo.content}
  </Tooltip>)}
</ImageContainer>
</ImageWrapper>
{showInventoryButtons && (
  <InventoryButtonsContainer>
    {activeImageIndex === 1 && (
      <FloorButton
        onClick={() => handleRoomButtonClick('floor4')}
        secondary
        >Этаж 4 - Список предметов
      </FloorButton>
    )}
    {activeImageIndex === 2 && (
      <FloorButton
        onClick={() => handleRoomButtonClick('floor6')}
        secondary
        >Этаж 6 - Список предметов
      </FloorButton>)}
  </InventoryButtonsContainer>)}
{isInteractiveAreaVisible && (
  <InteractiveArea
    place={selectedPlace}
    furniture={furnitureData[selectedPlace] || []}
    onClose={() => setInteractiveAreaVisible(false)}
  />)}

```

```

{showRoomInventory && (
  <RoomInventoryModal>
    <CloseButton onClick={() =>
setShowRoomInventory(false)}>×</CloseButton>

    <h3 style={{ marginTop: 0 }}>Инвентарь комнаты на {currentRoom
=== 'floor4' ? '4 этаже' : '6 этаже'}</h3>

    {!isEditMode && (
      <div style={{ marginBottom: 20 }}>
        <PasswordInput
          type="password"
          value={passwordInput}
          onChange={(e) => setPasswordInput(e.target.value)}
          placeholder="Пароль администратора"
          onPress={(e) => e.key === 'Enter' && checkPassword()}
        />
        <ActionButton
          onClick={checkPassword}
          style={{ backgroundColor: '#4CAF50', color: 'white' }}
        >
          Разрешить редактирование
        </ActionButton>
      </div>
    )}

    {roomItems.length === 0 ? (
      <p>Нет данных о предметах</p>
    ) : (
      <div>
        {roomItems.map((item, index) => (
          <InventoryItem key={index} style={{ backgroundColor:
cellColors[index] || 'transparent' }}>
            {isEditMode ? (

```

```

<><InventoryInput
    type="text"
    value={item.name}
    onChange={ (e) => handleItemChange(index, 'name',
e.target.value)}
    placeholder="Название предмета"
/><InventoryInput
    type="number"
    value={item.count}
    onChange={ (e) => handleItemChange(index, 'count',
e.target.value)}
    style={{ width: '80px' }}
    min="0"/>
<ActionButton
    onClick={() => removeItem(index)}
    style={{ backgroundColor: '#ff4444', color: 'white' }}
>Удалить</ActionButton>
</>) : (<
    <div style={{ flex: 1 }}>{item.name || 'Не указано'}</div>
    <div style={{ width: '60px' }}>{item.count} шт.</div>
</>)</InventoryItem>)))
</div>){isEditMode && (
<div style={{ marginTop: 20 }}>
    <ActionButton
        onClick={addNewItem}
        style={{ backgroundColor: '#2196F3', color: 'white' }}
    >+ Добавить предмет</ActionButton><ActionButton
        onClick={handleSaveRoomItems}
        style={{ backgroundColor: '#4CAF50', color: 'white',
marginLeft: 10 }}>Сохранить изменения
    </ActionButton></div>)</RoomInventoryModal>
)</div>);};export default InteractiveImage;

```

Приложение В

(Обязательное)

Офисный консьерж

Презентация

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		74

Офисный консьерж

Автор работы:

студент гр. 21ВВВ1 Кузнецов Алексей Сергеевич

Научный руководитель:

к.т.н., доцент Токарев Андрей Николаевич

Цель работы и объекты

Цель работы –разработать клиент-серверное приложение для автоматизации управления офисом.

Объектом проекта является клиент-серверное приложение, выполняющие следующие функции:

- навигация по этажам;
- учет мебели и ресурсов;
- интеграцию с чат-ботом;
- систему бронирования.

Кому это полезно?

- ◇ Офис-менеджерам
- ◇ Бухгалтерам
- ◇ Администраторам
- ◇ Работникам офиса

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		75

Используемые технологии

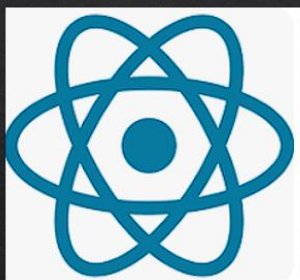


Диаграмма вариантов использования

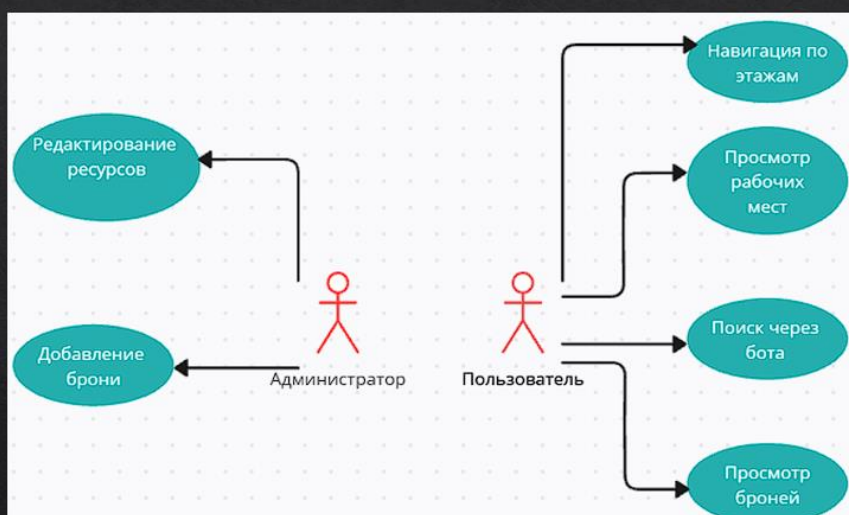
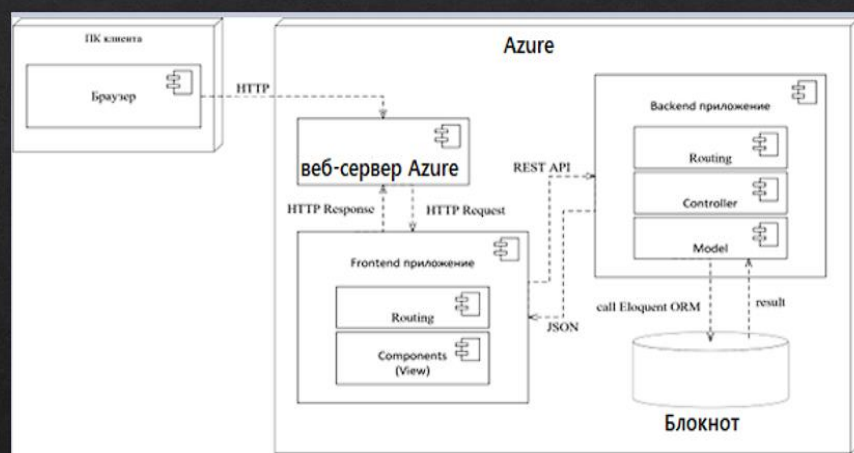
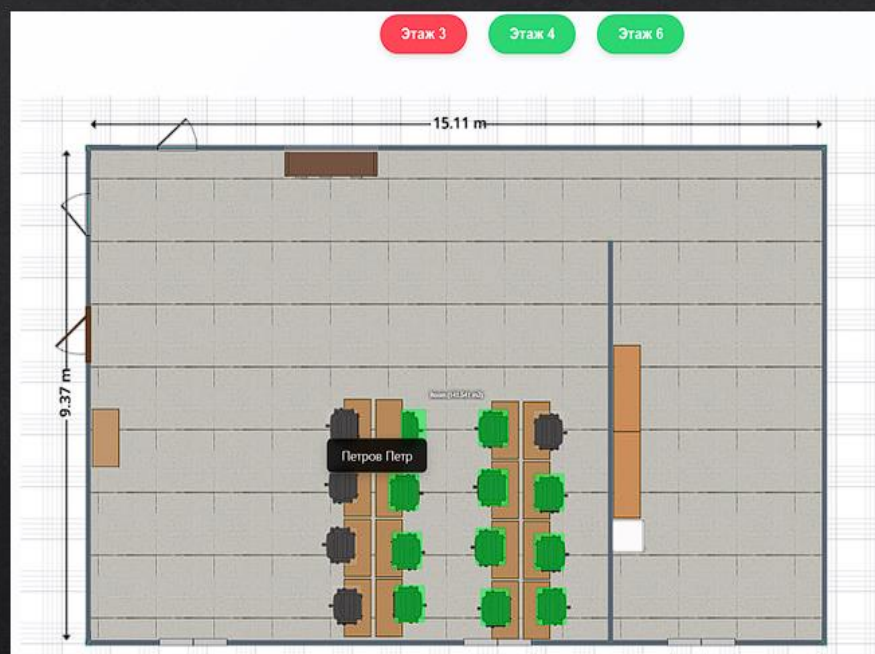


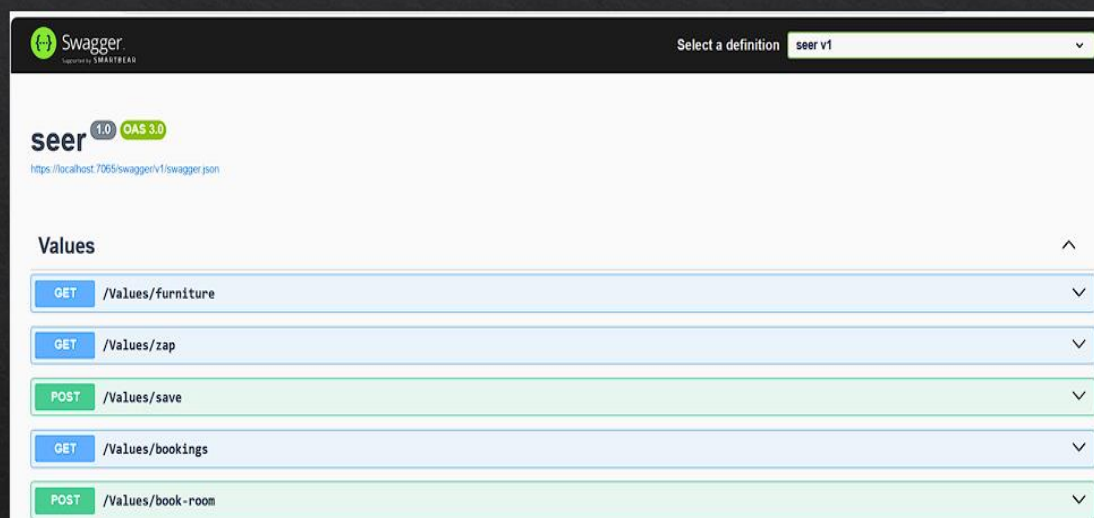
Диаграмма развертывания



Вид клиентской части



Тестирования API



Результаты работы

Разработана клиентская часть приложения, выполняющая следующие функции

- ♦ Навигация по этажам
- ♦ Учет ресурсов
- ♦ Закрепление рабочих мест
- ♦ Система бронирования
- ♦ Чат-бот

◆ СПАСИБО ЗА ВНИМАНИЕ!

					ПГУ1.09.03.01.15.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		79

