

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

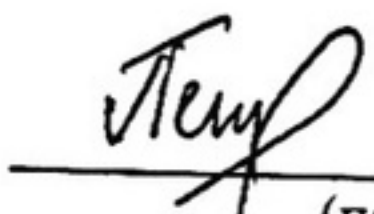
Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

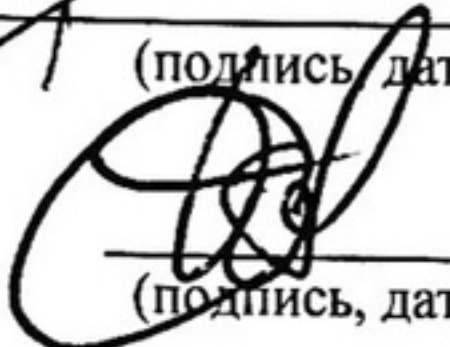
на тему

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ
ПОИСКА ЖИЛЬЯ

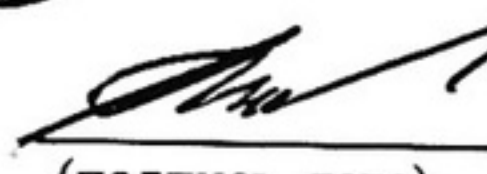
Студент

 Петров Олег Михайлович
(подпись, дата) (ФИО полностью)

Руководитель

 31.05.2023 Р.Н. Федюнин
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

 31.05.2023 А.С. Бычков
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 2.06.2023 г. № 14)

Заведующий кафедрой

 М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

06.06.2023 № 3)

Секретарь ГЭК

 Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2023

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

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

«26» 04 2023г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Петров Олег Михайлович гр. 19ВВП1 факультета вычислительной техники.

2. Тема работы «Программное обеспечение для поиска жилья»

Тема утверждена приказом ПГУ № 458/0 от "26" "04" 2023 г.

3. Руководитель бакалаврской работы к.т.н., доцент Федюнин Р.Н.

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

Разработать приложение «Программное обеспечение для поиска жилья», выполняющее следующие функции:

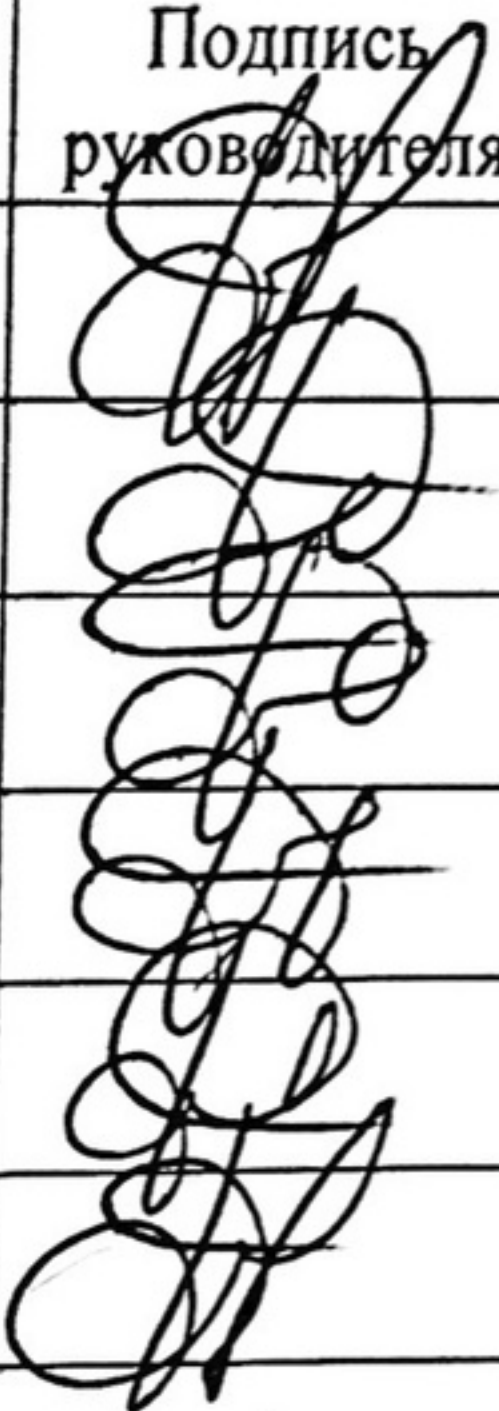
- авторизация и регистрация разных типов пользователей: гость, менеджер, авторизованный пользователь;
- сбор, обработка и визуализация данных об объектах недвижимости;
- обработка информации о бронировании объекта недвижимости авторизованными пользователями;
- осуществление функции поиска по сайту.

Приложение должно быть спроектировано с применением паттерна «Модель, Вид, Контроллер». Клиентская часть программы должна обладать понятным графическим интерфейсом. Ядро программы должно включать функционал сбора и передачи данных в БД через специализированный API.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить аналоги существующих приложения для поиска жилья, изучить способы проектирования и разработки собственного клиент-серверного приложения с применением инструментов Node JS.

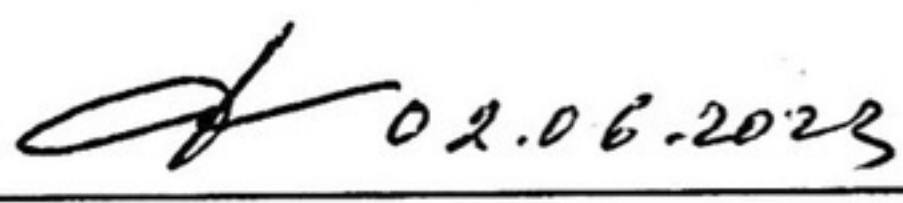
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	26.04.2023-30.04.2023	
2.	Анализ требований	10	31.04.2023-02.05.2023	
3.	Выбор технологий, применяемых при проектировании и разработке	50	03.05.2023-07.05.2023	
4.	Описание алгоритмов	40	08.05.2023-11.05.2023	
5.	Разработка приложения «Программное обеспечение для поиска жилья »	150	12.05.2023-29.05.2023	
6.	Оформление пояснительной записки	24	30.05.2023-31.05.2023	

Дата выдачи задания "26" апреля 2023 г.

Руководитель бакалаврской работы  Федюнин Р.Н.
(подпись, дата)

Задание к исполнению принял студент  Петров О.М.
(подпись, дата)

Работу к защите допустить
Декан факультета  02.06.2023 Фионова Л.Р.
(подпись, дата)

Реферат

Работа содержит страницы формата А4, рисунков, таблицы, источников, приложения.

БАЗА ДАННЫХ, ПАТТЕРН *MVC*, *SQL*, *MYSQL*, *NODEJS*.

Объект разработки – Программное приложение для поиска жилья.

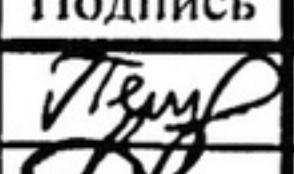
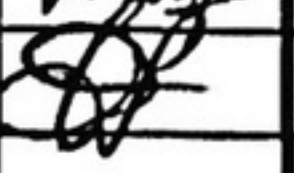
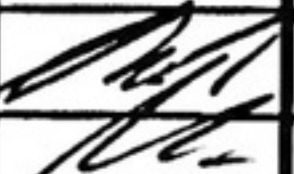
Целью дипломной работы является разработка приложения для поиска жилья в рамках собранных и сохраненных в СУБД данных. Цель дипломного проекта достигнута путем заполнения и сбора информации в хранилище данных, введенных сотрудниками профильных организаций.

В процессе дипломного проектирования были решены задачи сбора информации, структурирования и обработки информации о жилье, о потенциальных гостях и бронировании жилья.

В результате проведенной работы создано приложение, протестированы программные средства, удовлетворяющие поставленной задаче.

В пояснительной записке представлены: анализ предметной области, проектирование и реализация приложения, технология разработки и отладки программы, раздел БЖД.

Графическая часть содержит модели БД и *UML*-диаграммы приложения.

					ПГУ1.090301.18.001 ПЗ			
Изм.	Лист	№ докум	Подпись	Дата	Программное обеспечение для поиска жилья Пояснительная записка	Лит.	Лист	Листов
Разраб.		Петров О.М.					2	129
Провер.		Федюнин Р.Н.						
Н. Контр.		Бычков А.С.						
Утверд.		Митрохин М.А.				ФВТ, каф. ВТ гр. 19ВВП1		

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Программное обеспечение для поиска жилья	7
1.1 Предпроектный анализ	7
1.1.1 Анализ предметной области и аналогов.....	7
1.1.2 Техническое задание на разработку программного обеспечения.....	11
1.1.3 Анализ стека технологий	13
1.1.4 Проектирование приложения и описание процессов.....	17
1.1.5 Описание системы и программы с помощью диаграммы развертывания	29
1.2 Проектирование и реализация программного обеспечения для поиска жилья	31
1.2.1 Проектирование базы данных приложения	31
1.2.2 Проектирование клиентской части (интерфейса).....	49
1.2.3 Проектирование и реализация серверной части приложения.....	64
1.2.4 Выводы по подпункту	68
1.3 Технология отладки клиент серверных приложений	69
1.3.1 Установка и проверка работоспособности прт.....	73
1.3.2 Установка и проверка работоспособности composer	73
1.3.3. Установка и проверка работоспособности OpenServer.....	75
1.3.4 Установка и проверка работоспособности проекта	77
1.3.5 Установка и проверка работоспособности СУБД	78
2 БЖД (рекомендации для оператора ЭВМ)	80

Закключение	82
Список использованных источников	83
Приложение А – <i>UML</i> -диаграммы	84
Приложение А.1 – Диаграмма вариантов использования.....	85
Приложение А.2 – Диаграмма развертывания	86
Приложение А.3 – Диаграмма развертывания	87
Приложение А.4 – Диаграмма развертывания	88
Приложение Б – Листинг программы	89
Приложение В – Скриншоты программы.....	110
Приложение Г – Презентация	115

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

Паттерн – шаблон проектирования

СУБД – система управления базами данных

ЭВМ – электронно-вычислительная машина

API (Application Programming Interface) – программный интерфейс приложения

CRUD – акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (англ. create), чтение (read), модификация (update), удаление (delete).

MVC (Model-View-Controller) – «Модель-Представление-Контроллер» или «Модель-Вид-Контроллер»

				ПГУ1.090301.18.001 ПЗ	Лист
					5
Изм.	Лист	№ докум	Подпись Дата		

Введение

Услуга, предлагаемая через интернет, по определению, является формой веб-приложения. Учитывая доступность, многие веб-приложения разрабатываются для функций, которые ранее не требовали онлайн-доступа что делает доступным большинство услуг, а также бронирование жилья.

Благодаря наличию мобильных устройств и web приложений, таких как *Airbnb*, бронировать жилье или сравнивать условия проживания в той или иной локации можно с любого портативного устройства. Это является причиной растущей популярности индустрии онлайн-поиска и бронирования жилья. По данным *Globe News Wire*, объем мирового рынка онлайн-путешествий к 2030 году 141,8 % по сравнению с 2022 годом.

Выше сказанное показывает актуальность тематики реализуемого в дипломной работе приложения.

С технической точки зрения для реализации устойчивого к трафику приложения для бронирования жилья необходимо реализовать программное приложение с архитектурой состоящей из следующих частей: фронтенд (интерфейс), служба управления отелем, служба обслуживание клиентов (поиск и бронирование), служба просмотра статуса бронирования и СУБД.

Для реализации описанной архитектуры приложения дипломником реализованы следующие этапы дипломной работы:

В первой главе проведен предпроектный анализ: исследована предметная область, исследован стек технологий, проведен сравнительный анализ инструментов реализации проекта, описаны пользовательские роли и функционал приложения. Подробно описана архитектура приложения и реализация, показаны *UML*-диаграммы, основные части кода с описанием и приведено описание СУБД. Описана технология развертывания web приложений для работы.

Во второй главе описан раздел безопасности жизнедеятельности.

				ПГУ1.090301.18.001 ПЗ	Лист
					6
Изм.	Лист	№ докум	Подпись Дата		

1 Программное обеспечение для поиска жилья

1.1 Предпроектный анализ

1.1.1 Анализ предметной области и аналогов

Создаваемое приложение предназначено для помощи трем категориям пользователей (гости, авторизированные пользователи и менеджеры отелей) найти требуемый вариант для бронирования жилья в режиме онлайн.

Пользователю гость приложение позволяет просмотреть доступные на сайте варианты объектов недвижимости. Авторизированному пользователю ресурс позволяет подобрать требуемый под его запрос объект недвижимости для бронирования. Пользователю менеджер приложение позволяет иметь актуальную информацию о гостях и объектах недвижимости, которой данный менеджер управляет.

Первый этап реализации проекта по разработке приложения, это рассмотреть и проанализировать уже функционирующие приложения-аналоги.

Данное исследование позволяет выявить сильные и слабые стороны реализованных проектов и определиться с архитектурой будущего приложения, и с инструментарием для реализации проекта (таблица 1.1).

Таблица 1.1 - Описание аналогов реализуемой системы

Наименование	Функционал	
	плюсы	Минусы
Booking.com количество загрузок 100 000 000	<ul style="list-style-type: none">– сравнение отелей;– регистрация, предложения и скидки;– оплата в приложении;– отзывы об отелях;– запрос на ресурсы «третьей стороны».	<ul style="list-style-type: none">– СУБД находятся за пределами РФ;– не соблюдаются требования 152 ФЗ РФ «О персональных данных».

<i>Ostrovok</i> количество загрузок 1 000 000	<ul style="list-style-type: none"> – сравнение отелей; – регистрация, предложения и скидки; – оплата в приложении; – отзывы об отелях. 	<ul style="list-style-type: none"> – отсутствует возможность запроса на ресурсы «третьей стороны».
<i>HomeToGo</i> количество загрузок 500 000	<ul style="list-style-type: none"> – проверка профиля; – результат поиска; – сравнение цен; – возможность фильтрации; – сохранение избранного; – списки недвижимости; – запросы на бронирование. 	<ul style="list-style-type: none"> – СУБД находятся за пределами РФ; – не соблюдаются требования 152 ФЗ РФ «О персональных данных»; – отсутствует локализация на русском языке; – отдельные приложения для гостей и менеджеров.
<i>FlipKey</i> количество загрузок 300 000	<ul style="list-style-type: none"> – регистрация заезда и отъезда; – профиль недвижимости и хранение данных; – поиск и фильтрацию; – опцию бесплатного включения в список; – платежный шлюз; – услуги мгновенного бронирования; 	<ul style="list-style-type: none"> – СУБД находятся за пределами РФ; – не соблюдаются требования 152 ФЗ РФ «О персональных данных»; – не совместим с браузерами Microsoft.
<i>Vrbo</i> количество загрузок 1 000 000	<ul style="list-style-type: none"> – поисковый фильтр; – возможности просмотра и поиска; – информация профиля пользователя; – виртуальный помощник; – фотографии и отзывы; – описания недвижимости; – бронирование и оплата; 	<ul style="list-style-type: none"> – СУБД находятся за пределами РФ; – не соблюдаются требования 152 ФЗ РФ «О персональных данных»; – нет функций обмена сообщениями в приложении.

В приведенных в таблице 1.1. системах бронирования и поиска жилья выявлены общие недостатки:

- в импортных системах отсутствует обеспечение требований 152 ФЗ РФ «О персональных данных», в рамках которого хостинг и информация должны размещаться на физических и виртуальных серверах РФ,
- в импортных системах отключена система платежей российскими платежными системами (в частности МИР),
- в некоторых импортных и большинстве российских систем отсутствует поддержка API и библиотек «третьей стороны». Например, отсутствует взаимная интеграция Yandex путешествия и Ostrovok.ru.

Проанализировав существующие аналоги и функциональные плюсы и минусы существующих систем, принято решение реализовать программное обеспечение для поиска и бронирования жилья со следующими функциональными характеристиками (рисунок 1.1):

- пользователь вне зависимости от статуса (менеджер или авторизованный гость) имеет право авторизации и аутентификации;
- менеджер имеет возможность размещать информацию об объекте недвижимости вместе с ценами, правилами, доступными удобствами и прочими характеристиками объекта недвижимости;
- авторизованный гость имеет возможность поиска и аренды жилья, выполняет поиск места в приложении;
- авторизованный гость в процессе поиска задает пункт назначения, ценовой диапазон и другие критерии поиска;
- авторизованный гость, найдя подходящий объект и место для аренды, делает запрос на бронирование службе менеджера;
- менеджер получает запрос на бронирование, и решает, одобрить заказ или отклонить;

				ПГУ1.090301.18.001 ПЗ	Лист
					9
Изм.	Лист	№ докум	Подпись Дата		

- если менеджер одобряет запрос, пользователю приходит уведомление о возможности заселения;
- после оказания услуги гостю объект недвижимости снова получает статус свободно;
- менеджер имеет возможность просматривать аккаунт гостя для анализа;
- анонимный гость имеет право посетить ресурс с целью просмотра объектов недвижимости.

Входными данными являются:

- сведения об объектах недвижимости (отелях);
- сведения о параметрах конкретного объекта недвижимости;
- сведения о пользователе;
- сведения о датах и условиях бронирования;
- сообщения пользователей.

Выходными данными должны быть:

- хранимые в БД данные;
- сформированные отчеты и информация о бронировании объекта недвижимости.

Выше описанные функции (рисунок 1.1) будут реализованы с учетом требований 152 ФЗ РФ «О персональных данных».

Также будет учтена возможность использования API платформы Yandex для работы поиска и авторизации в рамках приложения.

Отдельным сервисом для описания является работа платежной системы, она в рамках данной работы реализована не будет.

				ПГУ1.090301.18.001 ПЗ	Лист
					10
Изм.	Лист	№ докум	Подпись Дата		

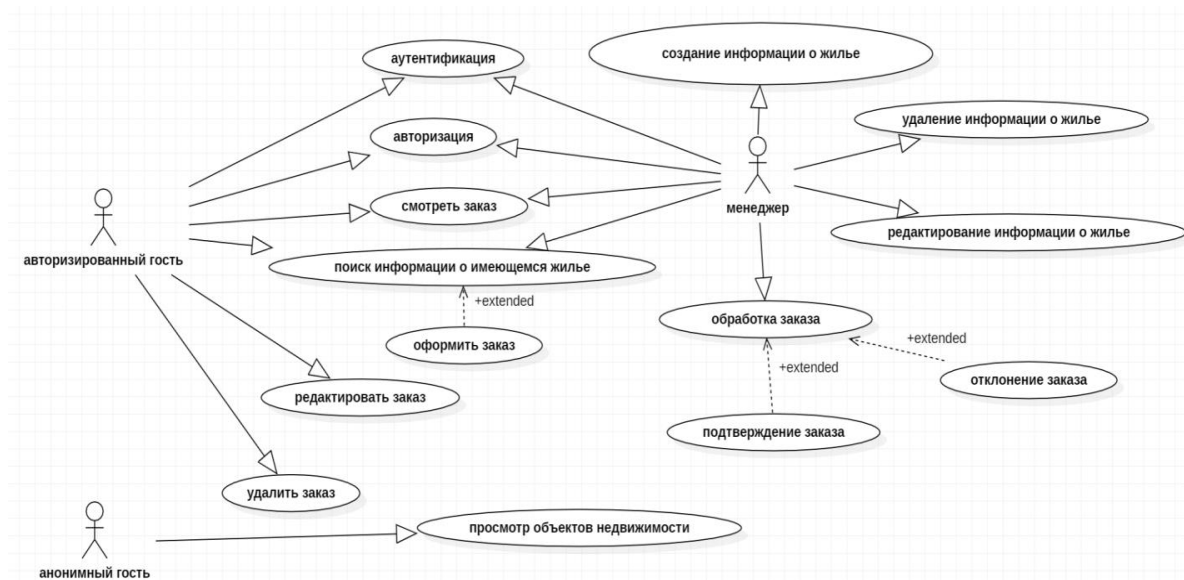


Рисунок 1.1 – диаграмма вариантов использования

1.1.2 Техническое задание на разработку программного обеспечения

1.1.2.1 Основание для разработки

Основанием для разработки является задание на дипломное проектирование, выданное и утвержденное д.т.н., заведующим кафедрой ВТ ПГУ Митрохиным М.А.

1.1.2.2 Назначение разработки

Программное обеспечение для поиска жилья предназначено для эффективного поиска и бронирования пользователями программы объектов недвижимости. Программа предназначена для формирования актуального списка объектов недвижимости и его размещения с помощью менеджеров системы.

Программа имеет трехзвенную архитектуру, реализованную на базе паттерна MVC с помощью фреймворков и библиотек для реализации современного web приложения.

1.1.2.3 Требования к надежности программы

При функционировании программы предусмотрен контроль вводимых и выбираемых данных, с последующим контролем вывода результата. В случае

ошибки ввода-вывода выдаются соответствующие предупреждающие сообщения с возможностью исправления ошибок и продолжения работы.

1.1.2.4 Требования к составу и параметрам технических средств

Система разработана и может функционировать на *IBM*-совместимых персональных компьютерах со следующей минимальной конфигурацией:

- процессор *Intel Core i3* 2.4 ГГц и выше;
- оперативная память 8 Гб и выше;
- жесткий диск *SSD* или *HDD* 1 Тбайт и выше;
- монитор 24 дюйма или выше.

Рабочая версия программного обеспечения предназначена для развертывания на виртуальном сервере с характеристиками не ниже описанных выше.

1.1.2.5 Требования к программной документации

Комплект разрабатываемых документов должен включать: текст и описание программы, описание применения программы.

1.1.2.6 Стадии и этапы разработки

Стадии и этапы разработки программного обеспечения для поиска жилья описаны в таблице 1.2.

Таблица 1.2 – Стадии и этапы разработки ПО

№ п/п	наименование этапов работы	объем работы	срок выполнения
1	Анализ предметной области	8%	
2	Составление технического задания	6%	
3	Выбор инструментов разработки	8%	
4	Проектирование системы	12%	

№ п/п	наименование этапов работы	объем работы	срок выполнения
5	Разработка программы	25%	
6	Разработка СУБД	25%	
7	Выполнение задания по БЖД	8%	
8	Оформление пояснительной записки	8%	

1.1.3 Анализ стека технологий

Выбор корректного для реализации программы стека технологий позволяет реализовать приложение с требуемым функционалом и техническими характеристиками, а также реализовать масштабируемую программу с хорошей технической поддержкой.

Реализация трехзвенной архитектуры приложения подразумевает выбор инструментария для реализации фронтенда, серверной части ПО и СУБД.

1.1.3.1 Сравнительный анализ инструментария для реализации фронтенда

Сравнительный анализ выделяет три популярных инструмента для реализации фронтенда реализуемого приложения *Angular*, *React* и *Express.js*.

Angular - это полноценный фреймворк для разработки веб-приложений. Он включает в себя все, что нужно для построения масштабируемых, сложных приложений, включая систему компонентов, роутинг и управление состоянием. *Angular* больше подходит для крупных проектов.

React - это библиотека для создания пользовательских интерфейсов. Он использует компонентную архитектуру, позволяя создать многоразовые и повторно используемые компоненты, которые могут динамически обновляться. *React* подходит для создания небольших веб-приложений, которые имеют большое количество пользовательских интерфейсов.

Express.js - это фреймворк для создания серверных и клиентских приложений на *Node.js*. Подходит для создания *API*, обработки маршрутизации и создания обработчиков запросов. *Express.js* хорошо подходит для создания *REST API*, обработки запросов, которые не должны обновлять интерфейс пользователя напрямую или не нуждаются в полной функциональности фреймворка на клиентской стороне.

В рамках проекта создается минимально жизнеспособный продукт (*MVP*) для реализации которого достаточно инструментов *React*, так как он лаконичен и содержит минимальный набор функций необходимый для написания интерфейса.

1.1.3.2 Сравнительный анализ инструментария для реализации серверной части приложения

Для реализации серверной части требуется фреймворк и язык с поддержкой *MVC* паттернов: *Ruby on Rails*, *Django*, *Node.js* и *Meteor.js* – фреймворки для создания веб-приложения в рамках *MVC*.

Ниже приведен сравнительный анализ на основании, которого выбран инструментарий разработки серверной части приложения.

Ruby on Rails - фреймворк для языка *Ruby*, который помогает быстро создавать веб-приложения. Он предлагает множество готовых решений, включая *ORM*, систему маршрутизации и генераторы кода, что делает его высокопроизводительным фреймворком. Но данный инструмент генерирует код, который медленно исполняется на сервере по сравнению с тремя другими инструментами, как результат данный инструмент не будет использован в проекте.

Django - фреймворк для языка *Python*, который объединяет набор встроенных функций и открытое сообщество для создания высокопроизводительных веб-приложений. *Django* предоставляет средства для создания веб-приложений, включая *ORM*, маршрутизацию, систему шаблонов и административную панель.

				ПГУ1.090301.18.001 ПЗ	Лист
					14
Изм.	Лист	№ докум	Подпись Дата		

Meteor.js - фреймворк, который сочетает клиент-серверную архитектуру, шаблоны и реактивные данные для быстрой разработки современных веб-приложений. Так как отсутствуют встроенные механизмы работы со сторонними *API* данный инструмент не рассматривается для использования в проекте.

Node.js - это платформа *JavaScript* для создания быстрых и масштабируемых веб-приложений. *Node.js* является отличным выбором, если требуется создавать веб-приложение с использованием *JavaScript*. *Node.js* позволяет быстро создавать узлы для взаимодействия с базами данных, а также сторонними *API*.

Так как скорость работы и работа со сторонними *API* является важной для реализуемого приложения, то *Node.js* выбран для реализации серверной части приложения.

1.1.3.3 Сравнительный анализ *hosting* провайдеров

Для размещения web приложения на сервер требуется услуга хостинга. Выбор произведен между двумя хостинг провайдерами, которые имеют физические серверы на территории РФ и соблюдают 152 ФЗ РФ «О персональных данных»: *OpenStack* и *DigitalOcean*.

– *OpenStack* - это открытая платформа облачных услуг, которая позволяет пользователю создавать свой облачный кластер. *OpenStack* обеспечивает масштабируемость и гибкость в настройке облачной инфраструктуры. *OpenStack* также предлагает больше контроля над средой хостинга и большую гибкость в управлении ресурсами по сравнению с *DigitalOcean*.

– *DigitalOcean* - это надежная облачная платформа, которая позволяет пользователям создавать виртуальные машины и машинные образы для быстрой разработки. Онлайн панели настройки обеспечивают простой и понятный пользовательский интерфейс управления. Хостинг провайдер предлагает быстрые серверы и высокоскоростное соединение с пользователем.

				ПГУ1.090301.18.001 ПЗ	Лист
					15
Изм.	Лист	№ докум	Подпись Дата		

Так как для реализации дипломного проекта требуется легкая в настройке и простая для развертывания приложения облачная платформа (хостинг), *DigitalOcean* является подходящим решением.

1.1.3.4 Сравнительный анализ СУБД

Выбор базы данных (БД) и системы управления базой данных (СУБД) является важным решением, от которого зависят архитектурные решения, при реализации приложения.

Ниже представлен сравнительный анализ представителей реляционных БД и *NoSQL*.

СУБД *MySQL*, *MSSQL*, *MongoDB* и *Azure DocumentDB* - это четыре разных типа базы данных.

СУБД *MySQL* - это свободное и открытое программное обеспечение для управления реляционными базами данных.

СУБД *MySQL* является одной из самых популярных баз данных в мире и используется в большом количестве веб-приложений. *MySQL* предоставляет надежную, стабильную и простую в использовании реляционную базу данных.

СУБД *Microsoft SQL* - закрытая платформа управления реляционными базами данных, разработанная *Microsoft*. *MSSQL* используется преимущественно в системах корпоративного уровня. *MSSQL* обеспечивает надежность, высокую производительность и соответствие стандартам безопасности.

СУБД *MongoDB* - открытое программное обеспечение, которое использует *NoSQL* подход к управлению данными.

MongoDB предоставляет гибкость и масштабируемость в использовании с неструктурированными данными веб-приложений. Система использует *JSON*-подобный формат данных, который позволяет быстрое чтение и запись данных.

СУБД *Azure DocumentDB* - облачный *NoSQL* сервис управления базами данных (с закрытым исходным кодом и форматом представления данных),

который был разработан *Microsoft. Azure DocumentDB* предоставляет надежность, масштабируемость и гибкость в использовании с неструктурированными данными.

В связи с тем, что для быстрой реализации проекта и его надежного использования требуется простая и надежная реляционная база данных с высокой производительностью и широкой поддержкой, то *MySQL* является лучшим выбором.

1.1.4 Проектирование приложения и описание процессов

Проектирование приложения реализовано с помощью *UML* диаграмм.

UML - стандартный язык, используемый для описания, проектирования, моделирования и документации программного обеспечения. *UML* включает в себя нотации для различных видов диаграмм, каждая из которых предназначена для описания определенных аспектов системы.

Существует следующие видов диаграмм *UML*:

- диаграммы классов. Диаграммы классов используются для описания статических аспектов системы, таких как классы, интерфейсы, отношения между классами и наследование;

- диаграмма вариантов использования позволяет представить функциональное поведение системы. Диаграммы вариантов использования часто используются для описания требований к системе, которые должны выполняться;

- диаграмма развертывания предназначена для описания архитектуры и распределения системы на сервера или компьютеры и показывает, как различные компоненты системы связаны между собой и на каком оборудовании они размещаются и работают;

- диаграммы последовательности. Диаграммы последовательности используются для описания взаимодействия между объектами в сценарии использования;

				ПГУ1.090301.18.001 ПЗ	Лист
					17
Изм.	Лист	№ докум	Подпись Дата		

– диаграммы деятельности. Диаграммы деятельности используются для описания последовательности действий и процессов, которые выполняет система;

– диаграммы состояний. Диаграммы состояний используются для описания жизненного цикла объекта и переходов между различными состояниями.

Каждая диаграмма имеет свои характеристики, включая основные элементы и отношения между ними. Они также могут использоваться в сочетании с другими диаграммами *UML* для создания полного описания системы. *UML* является мощным инструментом для описания, документирования и моделирования программных систем, и его эффективное использование может улучшить процесс разработки ПО.

В связи с тем, что в выше представленных разделах функциональные требования для пользователей системы: гостя и менеджера определены, с помощью диаграмм деятельности определим основные прецеденты (маршруты) пользователей в программе.

1.1.4.1 Описание программы с помощью диаграммы деятельности

Авторизация заключается в выполнении следующих шагов (рисунок 1.2):

- отображение страницы авторизации;
- ввод учетных данных пользователя;
- если данные верны, то переход на главную страницу, иначе вывод информации об ошибке.

Анонимный гость так же попадает на страницу авторизации, при указании признака «Гость» он переходит на главную страницу для анонимных посетителей, на которой может посмотреть список и описание объектов недвижимости.

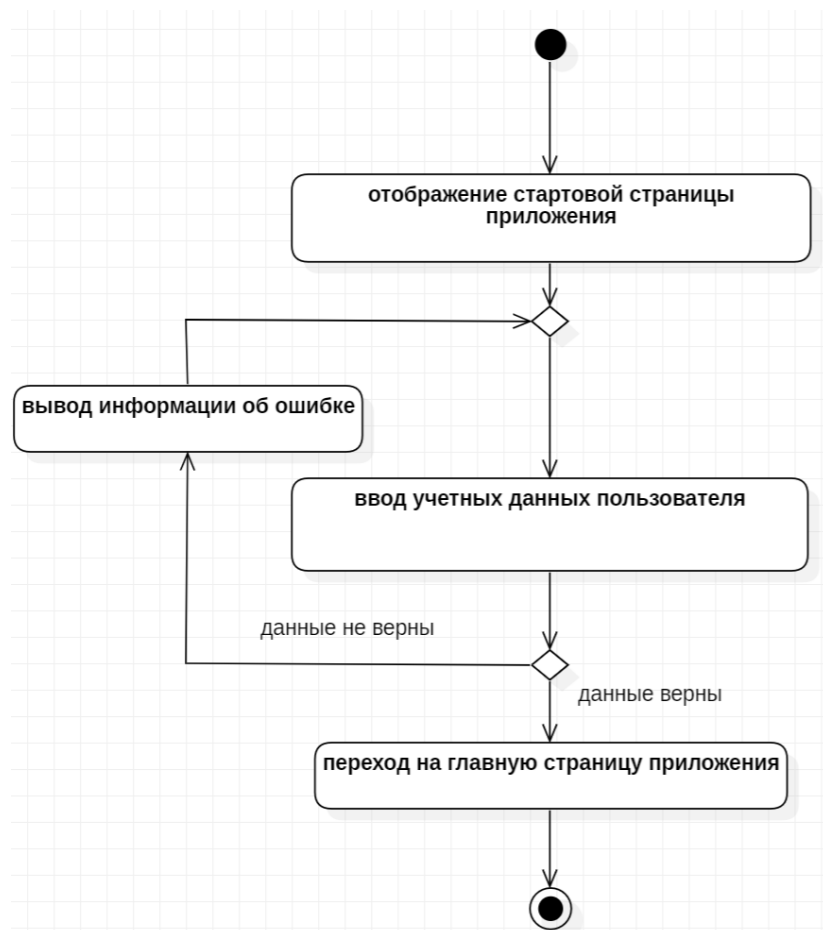


Рисунок 1.2 – Диаграмма деятельности: авторизация (гость/менеджер)

Регистрация заключается в выполнении следующих шагов (рисунок 1.3):

- отображение страницы с полями регистрации;
- ввод учетных данных пользователя;
- если данные верны, то переход на главную страницу, иначе вывод информации об ошибке.

Страница регистрации гостя и менеджера будут отличаться по количеству и набору полей.

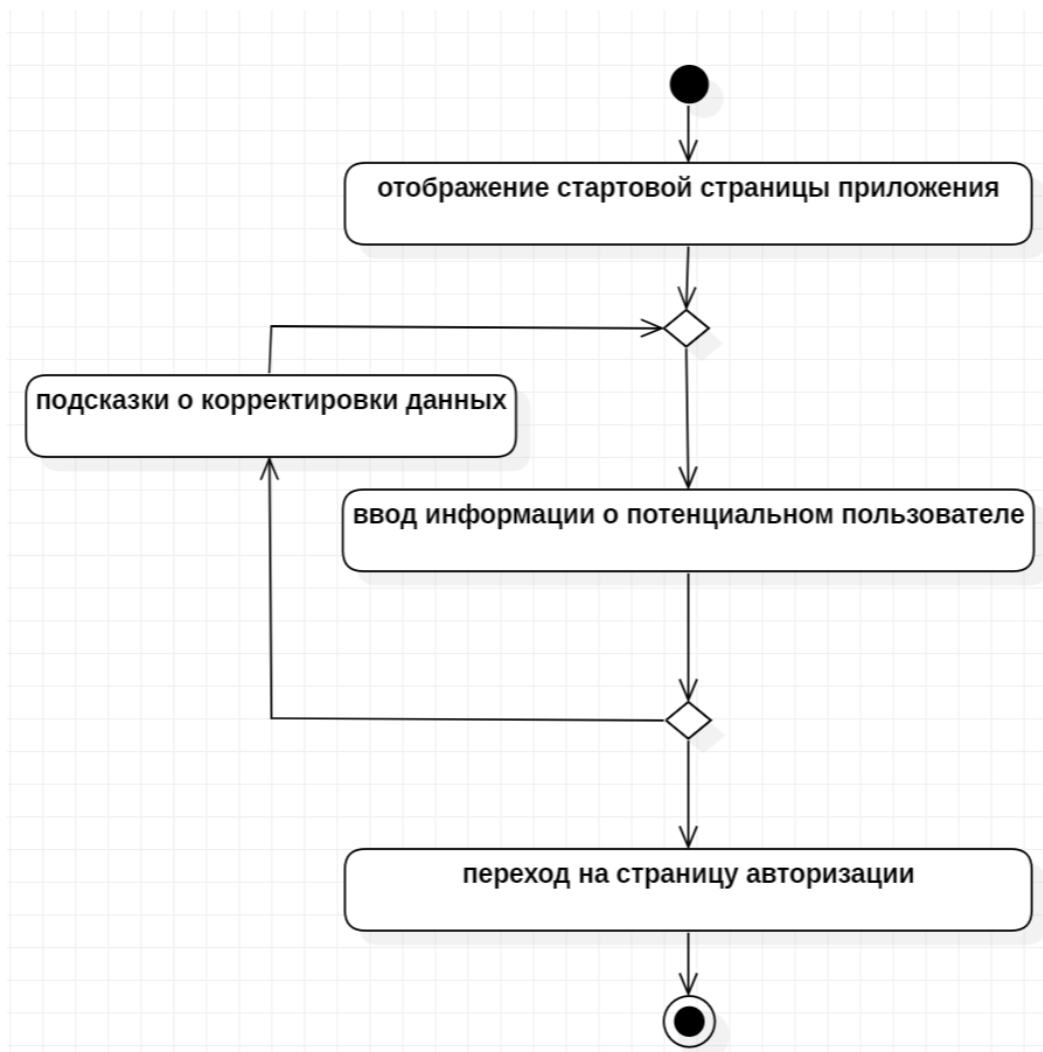


Рисунок 1.3 – Диаграмма деятельности: регистрация (гость/менеджер)

Поиск заключается в выполнении следующих шагов (рисунок 1.4):

- отображение главной страницы приложения;
- переход с главной страницы приложения на страницу поиска;
- ввод данных и критериев поиска;
- если данные верны, то переход на страницу результата поиска, иначе вывод информации об ошибке.

Страница поиска гостя и менеджера будут отличаться по количеству и набору полей.

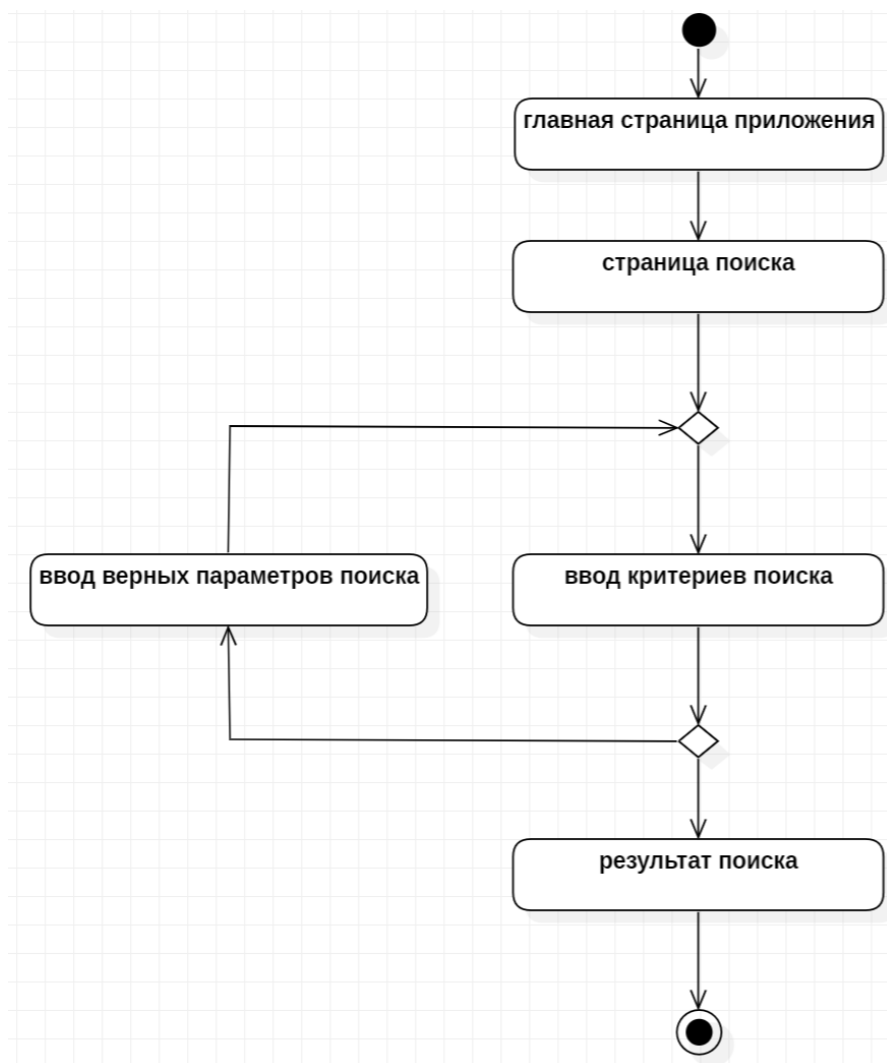


Рисунок 1.4 – Диаграмма деятельности: поиск имеющихся объектов недвижимости (гость/менеджер)

Процесс бронирования заключается в выполнении следующих шагов (рисунок 1.5):

- отображение страницы списка имеющихся отелей;
- выбор конкретного отеля из списка отелей;
- выбор конкретной комнаты отеля из списка доступных комнат отеля;
- выбор количества человек для заселения;
- выбор иных параметров заселения в комнату отеля;
- формирование сводной информации о бронировании как результат процесса бронирования.

На каждом этапе при вводе информации осуществляется проверка, если данные верны, то переход на следующий этап бронирования, иначе вывод информации об ошибке и возможность корректировки со стороны пользователя.

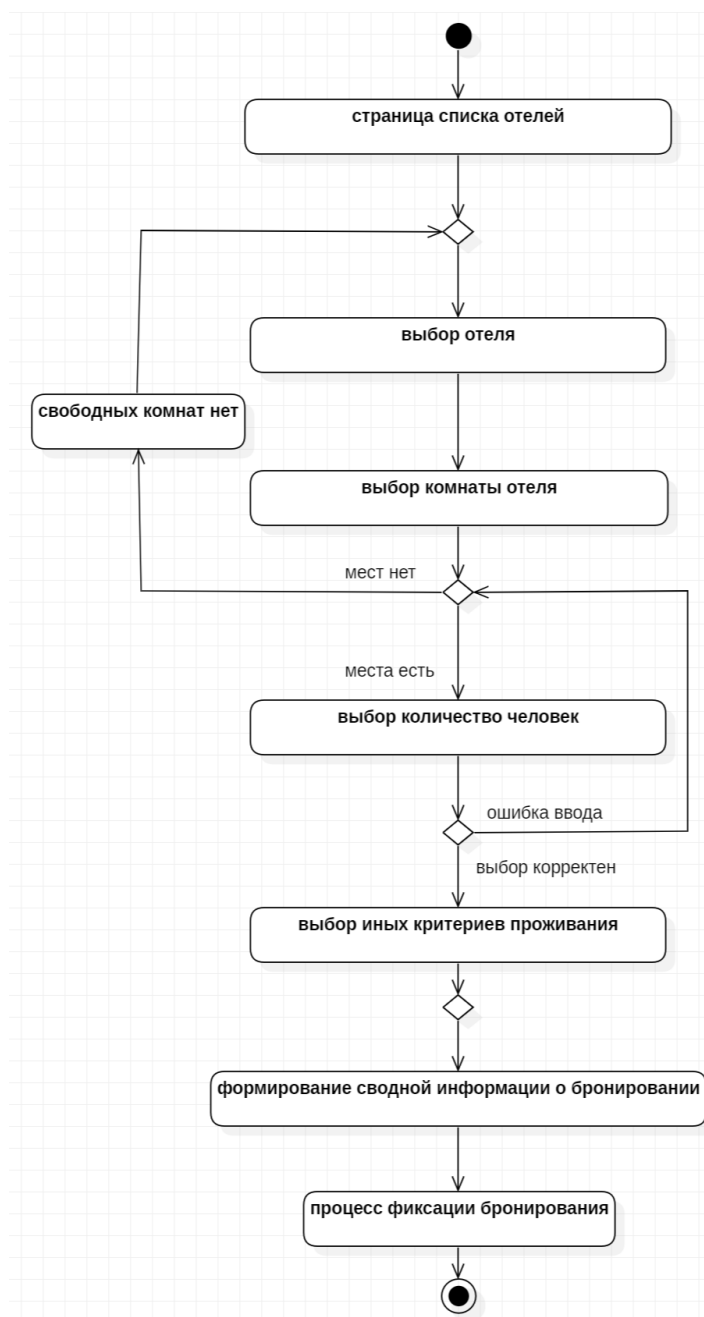


Рисунок 1.5 – Диаграмма деятельности: бронирование имеющихся объектов недвижимости (гость)

Процесс изменения заказа заключается в выполнении следующих шагов (рисунок 1.6):

- отображение страницы списка заказов пользователя;
- выбор конкретного заказа из списка;
- изменение информации о выбранном заказе;
- фиксация изменений о выбранном заказе.

на каждом этапе изменения заказа осуществляется проверка, данные верны, то они будут внесены и зафиксированы в БД, иначе вывод информации об ошибке, что заказов нет или введенные данные требуют корректировки со стороны пользователя.

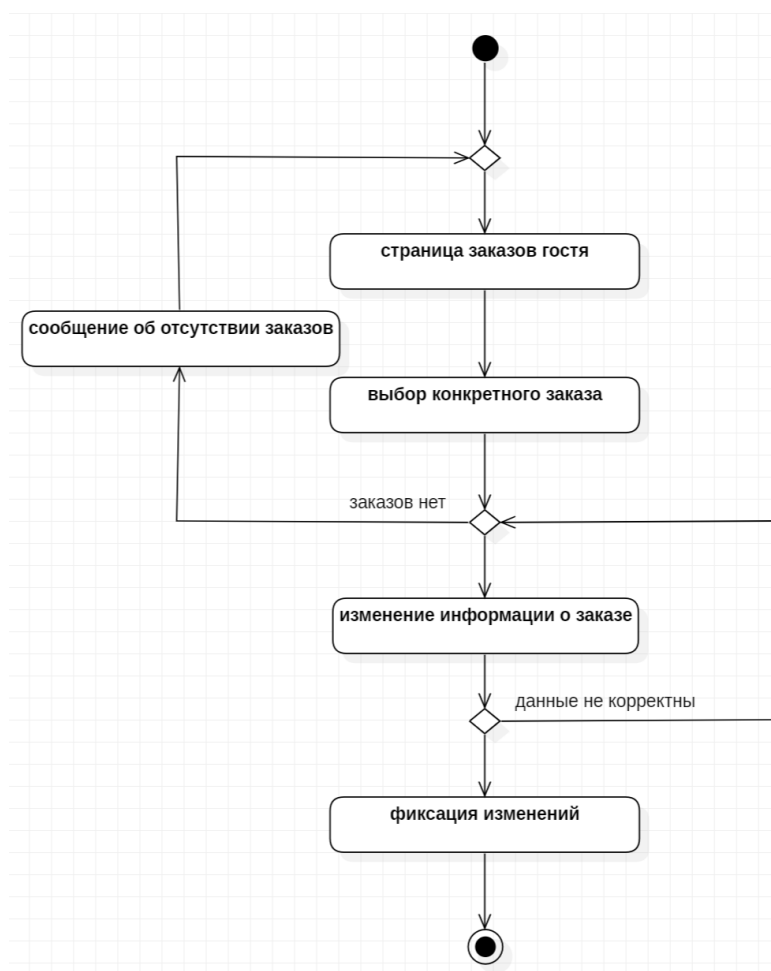


Рисунок 1.6 – Диаграмма деятельности: изменение информации о заказе (гость)

Процесс удаления заказа заключается в выполнении следующих шагов (рисунок 1.7):

- отображение страницы списка заказов пользователя;
- выбор конкретного заказа из списка;
- удалить информацию о выбранном заказе;
- обновить список заказов.

На каждом этапе удаления заказа осуществляется проверка, если списка заказов нет или данные требуют корректировки со стороны пользователя, то система проинформирует об этом.

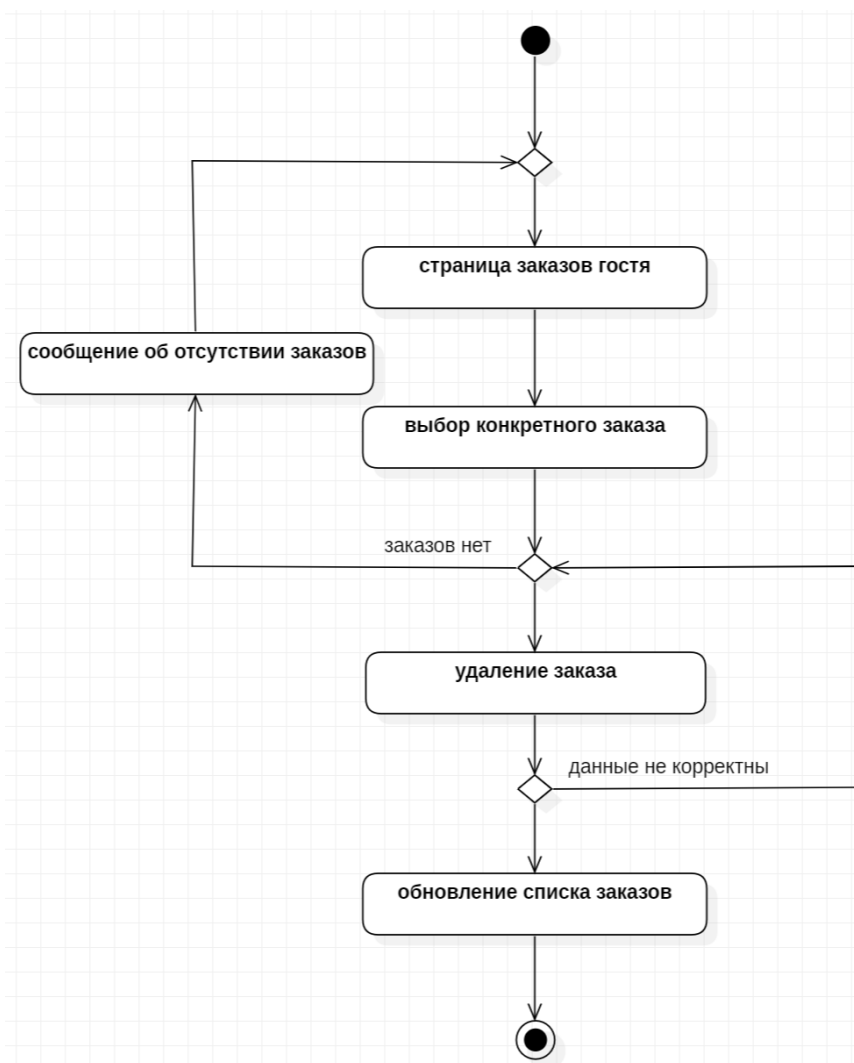


Рисунок 1.7 – Диаграмма деятельности: удаление заказа (гость)

Процесс создания объекта недвижимости заключается в выполнении следующих шагов (рисунок 1.8):

- отображение страницы создания объекта недвижимости (с полями ввода информации);
- ввод актуальных данных о вновь создаваемом объекте недвижимости;
- обновление списка объектов в СУБД.

На этапе ввода информации об объекте недвижимости осуществляется проверка, если поля не заполнены или данные требуют корректировки со стороны пользователя, то система проинформирует об этом.

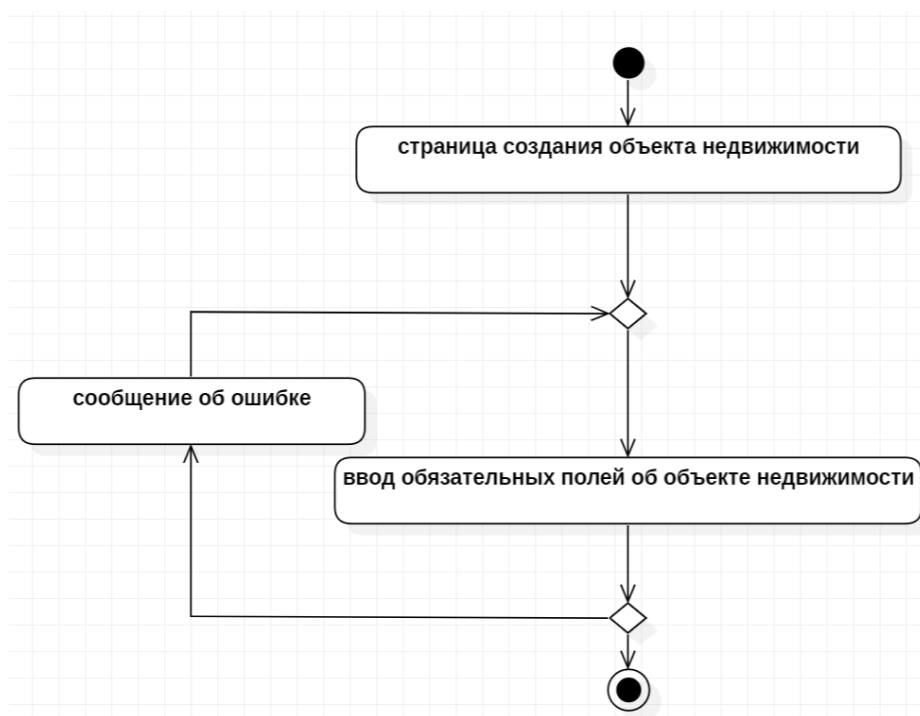


Рисунок 1.8 – Диаграмма деятельности: Создание объекта недвижимости (менеджер)

Процесс редактирования информации об объекте недвижимости заключается в выполнении следующих шагов (рисунок 1.9):

- отображение страницы редактирования объекта недвижимости (со списком доступных объектов);
- если список пуст, вернуться на главную страницу приложения;
- выбор требуемого объекта недвижимости из списка;
- ввод актуальных данных об объекте недвижимости;
- обновление списка объектов в СУБД.

На этапе ввода информации об объекте недвижимости осуществляется проверка, если поля не заполнены или данные требуют корректировки со стороны пользователя, то система проинформирует об этом.

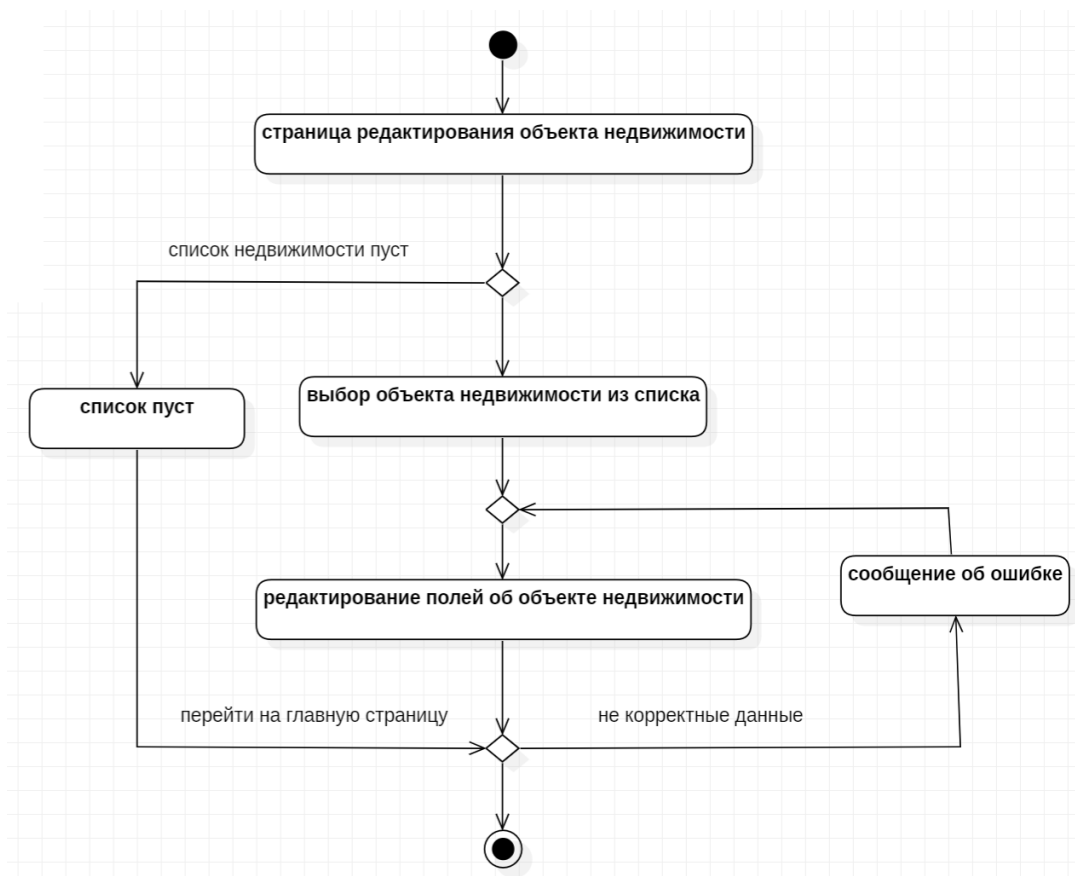


Рисунок 1.9 – Диаграмма деятельности: Редактирование информации об объекте недвижимости (менеджер)

Процесс удаления объекта недвижимости заключается в выполнении следующих шагов (рисунок 1.10):

- отображение страницы удаления объекта недвижимости (со списком доступных объектов);
- если список пуст, вернуться на главную страницу приложения;
- выбор требуемого объекта недвижимости из списка;
- удаление записи и информации об объекте недвижимости;
- обновление списка объектов в СУБД.

На этапе удаления информации об объекте недвижимости осуществляется проверка, если процесс удаления по какой-либо причине прошел не корректно, то система проинформирует об этом.

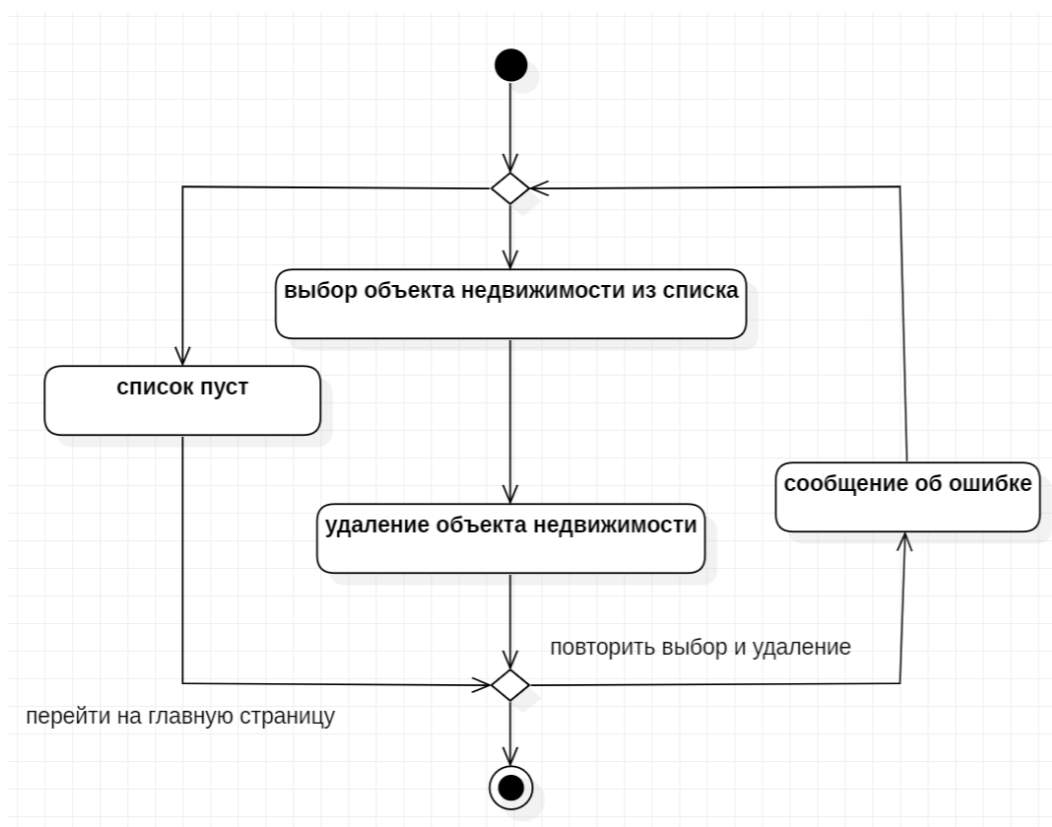


Рисунок 1.10 – Диаграмма деятельности: Удаление объекта недвижимости (менеджер)

Процесс подтверждения заказа пользователя гость заключается в выполнении следующих шагов (рисунок 1.11):

- отображение страницы полученных заказов (со списком доступных заказов);
- если список пуст, вернуться на главную страницу приложения;
- выбор требуемого заказа из списка;
- просмотр записи и информации о заказе и о потенциальном госте;
- подтверждение/отклонение заказа;
- отправка почтового уведомления гостю о подтверждении/отклонении заказа.

На этапе работы с заказами осуществляется проверка, если список заказов пуст, то система проинформирует об этом.

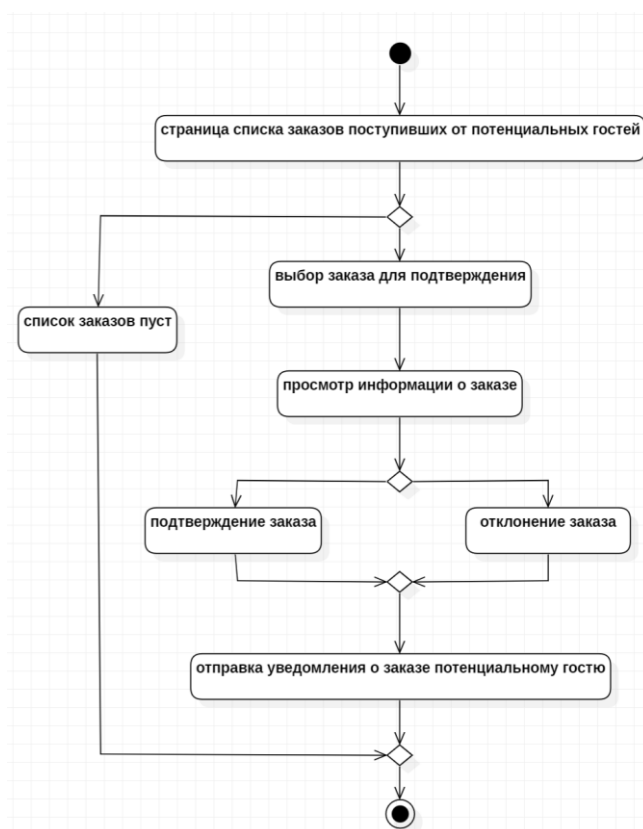


Рисунок 1.11 – Диаграмма деятельности: Подтверждение заказа гостя (менеджер)

1.1.5 Описание системы и программы с помощью диаграммы развертывания

Для описания архитектуры приложения выбрана *UML* диаграмма развертывания, так как эта диаграмма *UML* может описать как физическую, так и логическую архитектуру системы. Диаграмма отражает как компоненты системы размещаются и взаимодействуют между собой в рамках сети (рисунок 1.12).

Диаграмма развертывания предоставляет общее понимание того, как система развернута, и помогает в планировании реализации программы.

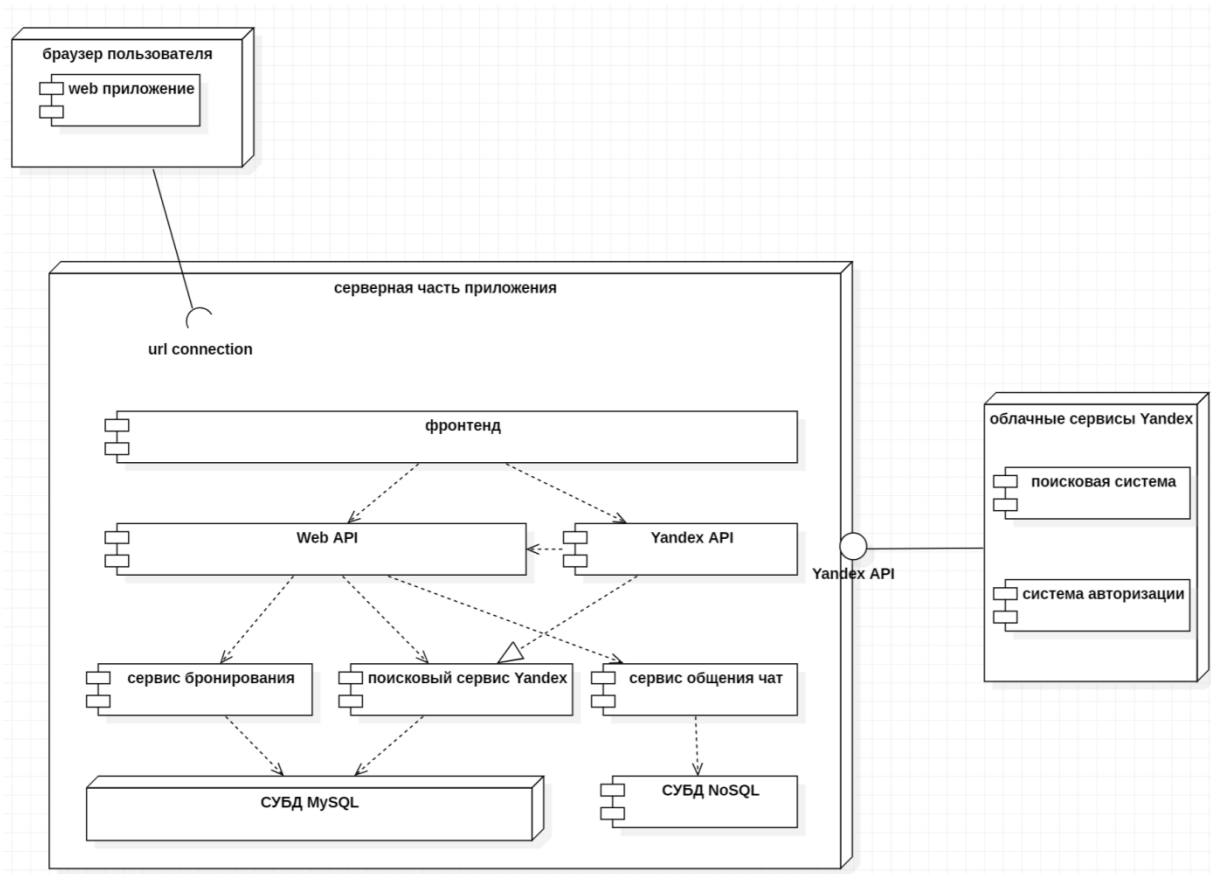


Рисунок 1.12 – Диаграмма развертывания: архитектура приложения

Как видно из рисунка, дипломный проект реализует классическую трехзвенную архитектуру приложения с тонким клиентом. URL запрос из браузера клиента поступает на сервер и фиксируется на фронтенде

приложения, с последующей передачей запроса на логический уровень серверной части приложения.

Слой *WEB API* производит разбор запроса пользователя, после анализа запрос отправляется на обработку одному из сервисов приложения.

Сервисы связанные с обработкой заказов, бронированием заказов, поиском заказов реализованы в рамках паттерна *MVC* с использованием реляционной СУБД *MySQL* и поддержкой сервисов *Yandex* для поиска и авторизации.

Для сервис чата оставлены функции «заглушки», которые позволяют или реализовать собственный чат или использовать библиотеку стороннего чата. Так как чат не является основным сервисом приложения его СУБД выделена в отдельным блок *NoSQL* (для простоты реализации и возможности масштабирования приложения).

В результате анализа обоснована актуальность выбранной темы, определена архитектура реализуемого приложения и прописан функцию.

				ПГУ1.090301.18.001 ПЗ	Лист
					30
Изм.	Лист	№ докум	Подпись Дата		

1.2 Проектирование и реализация программного обеспечения для поиска жилья

Проектирование программного приложения состоит из трех этапов:

- проектирование и реализация базы данных;
- макетирование и верстка интерфейсов пользователя с учетом представления СУБД;
- проектирование и реализация основной логики приложения с учетом представления СУБД и интерфейса пользователя.

1.2.1 Проектирование базы данных приложения

Проектирование базы данных приложения (рисунок 1.13) реализовано в дипломном проекте в три этапа: концептуальная модель данных, логическая модель данных, физическая модель данных.

Концептуальная модель данных - схема с минимальным количеством подробностей. Цель подхода заключается в возможности отобразить общую структуру модели и всю архитектуру системы.

Логическая модель данных содержит подробную информацию, по сравнению с концептуальной моделью. Цель подхода определить подробные операционные и транзакционные сущности. Логическая модель не зависит от технологии, в которой она будет применяться.

Физическая модель данных реализована на основе логической модели. В ней должны присутствовать технические подробности для составления и внедрения самой базы данных.

1.2.1.1 Логическая модель описание сущностей и их атрибутов

Модель данных реализуемой БД состоит из пяти связанных между собой предметных разделов: страны и города, компании владельцы объектов недвижимости, отели и номерной фонд, бронирование, гости.

Раздел страны и города состоит из двух сущностей (таблица 1.3 и 1.4): «Страны», «Города».

				ПГУ1.090301.18.001 ПЗ	Лист
					31
Изм.	Лист	№ докум	Подпись Дата		

Таблица 1.3 – Атрибуты сущности «Страны»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор страны	Целый	Да
2	Наименование страны	Строка	Да

Таблица 1.4 – Атрибуты сущности «Города»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор города	Целый	Да
2	Наименование города	Строка	Да
3	Почтовый индекс города	Целый	Нет

В разделе «Страны и города» находятся два словаря, каждый из которых содержит списки стран и городов, упомянутых в полном адресе компании или отеля. Поскольку города часто имеют несколько почтовых индексов, было принято решение использовать единственным «основной» почтовый индекс города.

Таблица 1.5 – Атрибуты сущности «Компания»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор компании	Целый	Да
2	Название компании	Строка	Да
3	Идентификационный номер налогоплательщика	Целый	Да

№	Наименование атрибута	Тип	Обязательный
4	Адрес электронной почты этой компании	Строка	Нет
5	Полный адрес компании, включая название улицы, номер и почтовый индекс	Строка	Да
6	Детали о компании	Строка	Нет
7	Признак активации компания в системе бронирования	Логический	Да

Таблица 1.6 – Атрибуты сущности «Учетная запись компании» (менеджера)

Таблица 1.7 – Атрибуты сущности «План компании»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор учетной записи	Целый	Да
2	Наименование плана	Строка	Да
3	Детали плана	Строка	Нет
4	Минимальное количество комнат	Целый	Да
5	Максимально количество комнат	Целый	Да
6	Помесячная стоимость комнаты	Числовое значение	Да

№	Наименование атрибута	Тип	Обязательный
8	Временная метка создания учетной записи	Целый	Да
9	Временная метка обновления учетной записи	Целый	Да
10	Признак связи с менеджера с компанией	Логический	Да

В разделе «Компании владельцы объектов недвижимости» связь между компаниями и планами будет реализована с помощью таблицы-словаря «Компания-План». Данный механизм позволит компании изменить планы в соответствии со своими текущими потребностями.

Таблица 1.8 – Атрибуты сущности «Затраты компании»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор статьи затрат	Целый	Да
2	Сумма затрат	Числовое значение	Да
3	Период затрат	Строка	Нет
4	Временная метка выставления счета	Целый	Да
5	Временная метка оплаты счета	Целый	Да
6	Временная метка отмены счета	Целый	Да

Ключевой таблицей в разделе «Отели и номерной фонд» является сущность «Гостиница». В данной логической модели описания сущностей и

их атрибутов использован термин «гостиница» для описания набора комнат. Тем не менее, эта сущность может быть использована для описания отдельных квартир, moteлей, общежитий и иных объектов недвижимости.

Таблица 1.9 – Атрибуты сущности «Отели»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор отеля	Целый	Да
2	Наименование отеля	Строка	Да
3	Описание	Строка	Нет
4	Ссылка на город	Целый	Да
№	Наименование атрибута	Тип	Обязательный
5	Ссылка на компанию	Целый	Да
6	Признак активации отеля	Логический	Да

Для каждого отеля используется уникальное поле «Наименование отеля», текстовое описание, его идентификатор категории и ссылки на идентификатор компании, управляющей отелем, и идентификатор города, в котором находится отель. Атрибут активации указывает, работает ли этот отель или нет.

Таблица 1.10 – Атрибуты сущности «Категория отеля»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор категории отеля	Целый	Да
2	Категория отеля	Строка	Да

Таблица 1.11 – Атрибуты сущности «Комната»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор комнаты в отеле	Целый	Да
2	Наименование комнаты	Строка	Да
3	Описание комнаты	Строка	Нет
4	Ссылка на отель	Целый	Да
5	Ссылка на тип комнаты	Целый	Да
№	Наименование атрибута	Тип	Обязательный
6	Цена комнаты	Числовое значение	Да

Таблица 1.12 – Атрибуты сущности «Тип комнаты»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор типа комнаты в отеле	Целый	Да
2	Тип комнаты в отеле	Строка	Да

Предметный раздел «Бронирование» состоит из таблиц 1.13 – 1.16

Таблица 1.13 – Атрибуты сущности «Бронирование»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор бронирования	Целый	Да
2	Идентификатор гостя	Целый	Да

№	Наименование атрибута	Тип	Обязательный
3	Начальная дата бронирования	Временная метка	Да
4	Конечная дата бронирования	Временная метка	Да
5	Временная метка создания записи о бронировании	Целый	Да
6	Временная метка обновления записи о бронировании	Целый	Да
7	Скидка	Целый	Нет
8	Сумма бронирования	Целый	Да

Таблица 1.14 – Атрибуты сущности «Канал бронирования»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор канала бронирования	Целый	Да
2	Наименование канала бронирования	Строка	Да
3	Описание	Строка	Нет

Каналы бронирования - список сайтов и агрегаторов, с которыми реализуемое приложение поддерживает синхронизацию. Список хранится в таблице «Канал бронирования». Каналы, поддерживающие работу со сторонними приложениями: «Yandex», «Островок».

Таблица 1.15 – Атрибуты сущности «Статус бронирования»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор статуса бронирования	Целый	Да
2	Статус бронирования	Строка	Да

Таблица 1.16 – Атрибуты сущности «Гости»

№	Наименование атрибута	Тип	Обязательный
1	Идентификатор гостя	Целый	Да
2	Имя	Строка	Да
3	Фамилия	Строка	Да
4	Электронный почтовый адрес	Строка	Да
№	Наименование атрибута	Тип	Обязательный
5	Телефон	Строка	Нет
6	Локация	Строка	Нет
7	Детали и предпочтения гостя	Строка	Нет

1.2.1.2 Физическая модель описание сущностей и их атрибутов

При реализации физической модели БД будут созданы дополнительные таблицы, связывающие между собой отдельные логические модули базы данных (рисунок 1.13).

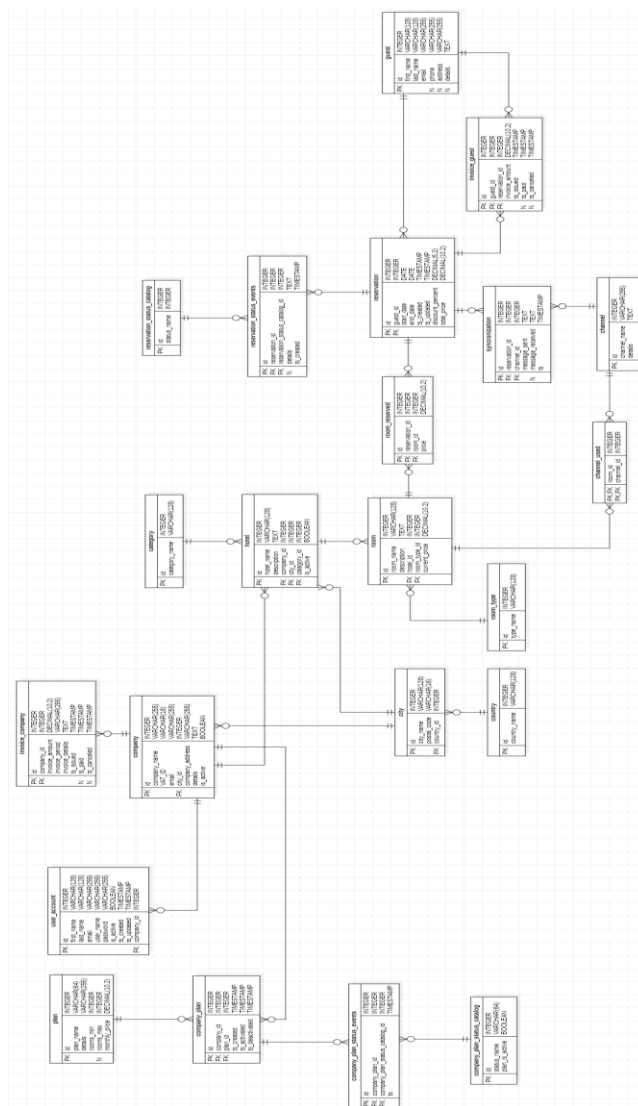


Рисунок 1.13 – Диаграмма базы данных программного обеспечения для поиска жилья

Таблица 1.17 – Описание полей таблицы «Country»

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ, обязательный
2	country_name	varchar(128)	Да

Таблица 1.18 – Описание полей таблицы «City»

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ, обязательный
2	city_name	varchar(128)	Обязательный
3	postal_code	varchar(16)	Необязательный
4	country_id		Внешний ключ

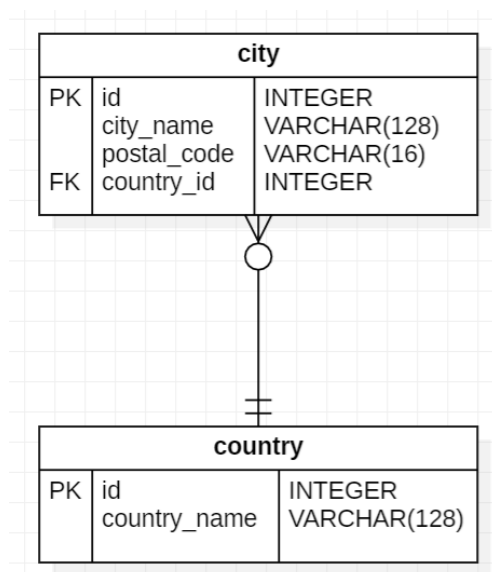


Рисунок 1.14 – Диаграмма базы данных предметного раздела страны и города.

Таблица 1.19 – Описание полей таблицы «Company»

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ, обязательный
2	company_name	varchar(255)	Обязательный
3	VAT_ID	varchar(16)	Обязательный

№	Имя поля	Тип	Примечание
4	email	varchar(255)	Необязательный
5	city_id	int	Внешний ключ
6	company_address	varchar(255)	Обязательный
7	details	text	Необязательный
8	is_active	boolean	Обязательный

Таблица 1.20 – Описание полей таблицы «User_accaunt»

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ, обязательный
2	first_name	varchar(255)	Обязательный
3	last_name	varchar(16)	Обязательный
4	email	varchar(255)	Необязательный
5	user_name	int	Внешний ключ
6	password	varchar(255)	Обязательный
7	is_active	boolean	Обязательный
8	ts_created	timestamp	Обязательный

Таблица 1.21 – Описание полей таблицы «Invoice_company»

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ, обязательный
2	company_id	int	Внешний ключ
3	invoice_amount	decimal(10,2)	Обязательный
4	invoice_period	varchar(255)	Обязательный
5	invoice_details	text	Необязательный
8	ts_issued	timestamp	Обязательный
9	ts_paid	timestamp	Обязательный
10	ts_canceled	timestamp	Обязательный

Таблица 1.22 – Описание полей таблицы «Plan»

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ, обязательный
2	plan_name	varchar(64)	Обязательный
3	details	varchar(255)	Необязательный
4	rooms_min	int	Обязательный
5	rooms_max	int	Обязательный
6	monthly_price	decimal(10,2)	Обязательный

Таблица 1.23 – Описание полей таблицы «Company_Plan» (таблица связка)

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ, обязательный
2	company_id	varchar(64)	Внешний ключ
3	plan_id	varchar(255)	Внешний ключ
4	ts_created	timestamp	Обязательный
5	ts_paid	timestamp	Обязательный
6	ts_canceled	timestamp	Обязательный

Таблица 1.24 – Описание полей таблицы «Company_Plan_Status_Events» (таблица связка)

№	Имя поля	Тип	Примечание
1	Id	int	Первичный ключ, обязательный
2	company_plan_id	int	Внешний ключ
3	company_plan_status_catalog_id	int	Внешний ключ
4	ts	timestamp	Обязательный

Таблица 1.25 – Описание полей таблицы «Company_Plan_Status_Cataloge»

№	Имя поля	Тип	Примечание
1	id	Int	Первичный ключ, обязательный
2	status_name	varchar(64)	

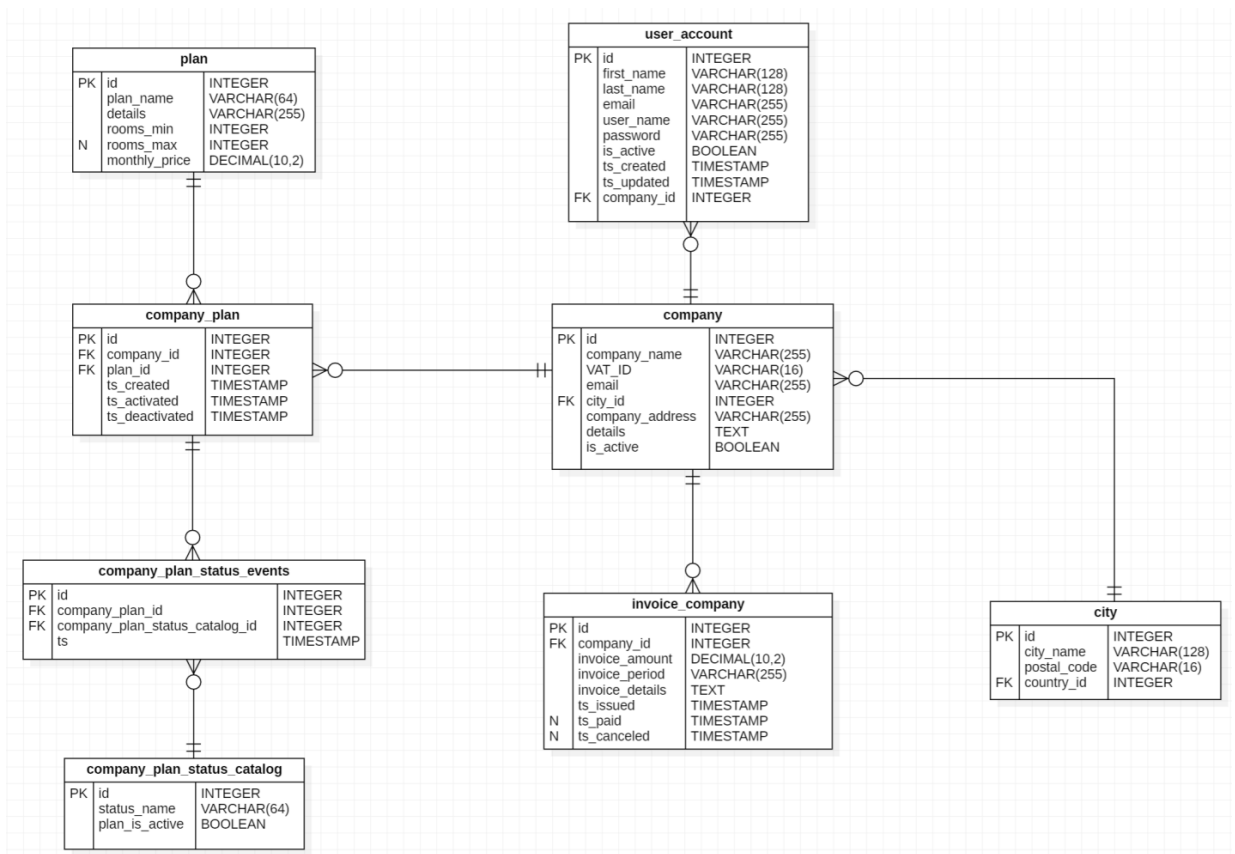


Рисунок 1.15 – Диаграмма базы данных предметного раздела компании.

Таблица 1.26 – Описание полей таблицы «Reservation»

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ
2	guiest_id	int	Внешний ключ
3	start_date	date	Обязательный
4	end_date	date	Обязательный
5	ts_created	timestamp	Обязательный
6	ts_update	timestamp	Обязательный
7	discount_percent	decimal(5,2)	Обязательный
8	total_price	decimal(10,2)	Обязательный

Таблица 1.27 – Описание полей таблицы «Reservation_Status_Events» (таблица связка)

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ, обязательный
2	reservation_id	int	Внешний ключ
3	reservation_status_catalog_id	int	Внешний ключ
4	details	text	Необязательный
5	ts_created	timestamp	Обязательный

Таблица 1.28 – Описание полей таблицы «Reservation_Status_Catalog»

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ, обязательный
2	status_name	int	Обязательный

Таблица 1.29 – Описание полей таблицы «Synchronization» (таблица связка)

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ, обязательный
2	reservation_id	int	Внешний ключ
3	channel_id	int	Внешний ключ
4	message_sent	text	Обязательный

№	Имя поля	Тип	Примечание
5	message_recived	text	Обязательный
6	ts	timestamp	Обязательный

Таблица 1.30 – Описание полей таблицы «Channel»

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ, обязательный
2	channel_name	varchar(255)	Обязательный
3	details	text	Обязательный

Таблица 1.31 – Описание полей таблицы «Channel_Used» (таблица связка)

№	Имя поля	Тип	Примечание
1	room_id	int	Первичный ключ, внешний ключ, Обязательный
2	channel_id	varchar(255)	Первичный ключ, внешний ключ, Обязательный

Таблица 1.32 – Описание полей таблицы «Room_Reserved» (таблица связка)

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ
2	reservation_id	int	Внешний ключ
3	room_id	int	Внешний ключ
4	Price	decimal(10,2)	Обязательный

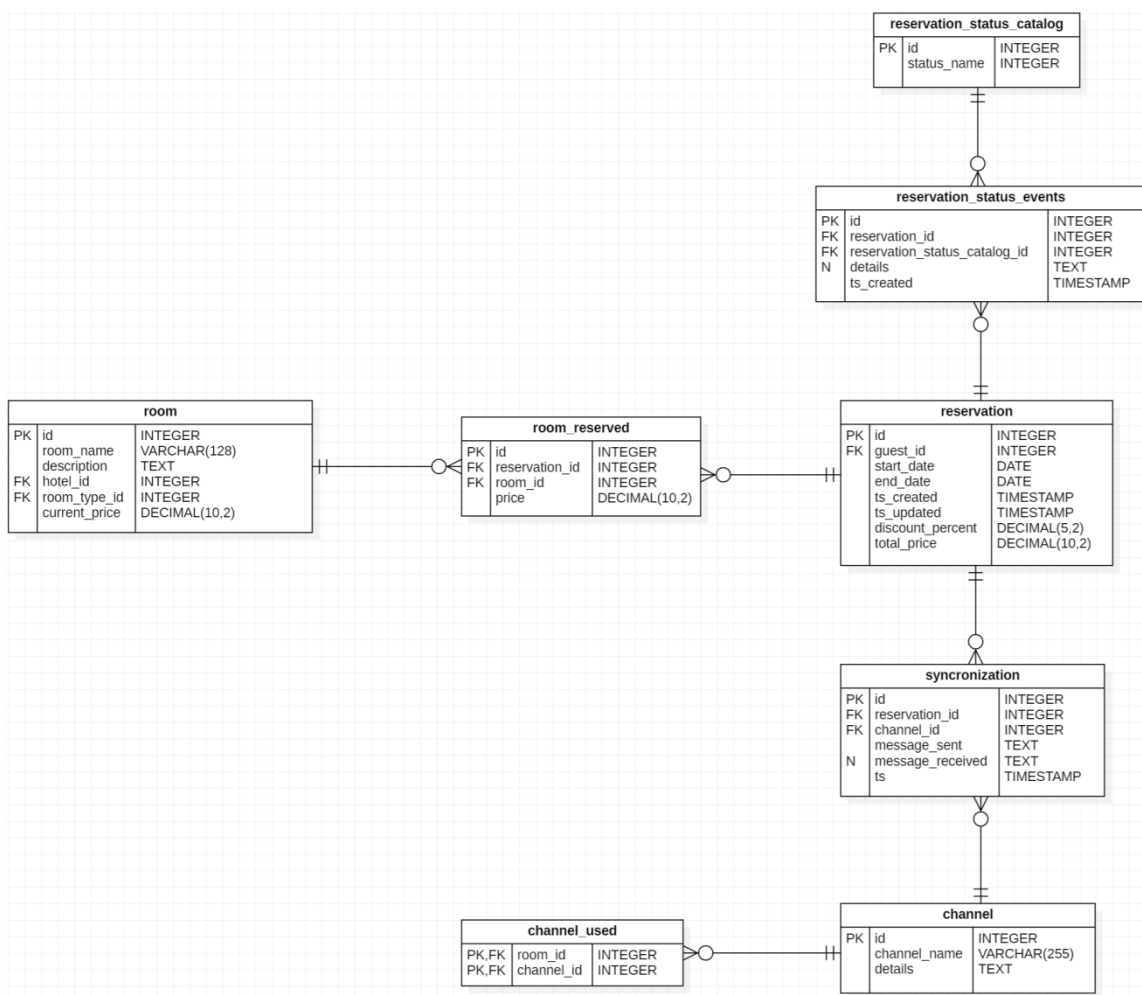


Рисунок 1.16 – Диаграмма базы данных предметного раздела бронирование.

Таблица 1.33 – Описание полей таблицы «Guest»

№	Имя поля	Тип	Примечание
1	Id	int	Первичный ключ
2	first_name	varchar(128)	Обязательный
3	last_name	varchar(128)	Обязательный
4	Email	varchar(255)	Обязательный
5	phone	varchar(255)	Обязательный
6	address	varchar(255)	Обязательный
7	details	text	Необязательный

Таблица 1.34 – Описание полей таблицы «Invoice_Guest»

№	Имя поля	Тип	Примечание
1	id	int	Первичный ключ
2	guest_id	int	Обязательный
3	reservation_id	int	Обязательный
4	invoice_amount	varchar(255)	Обязательный
5	ts_issued	timestamp	Обязательный
6	ts_paid	timestamp	Обязательный
7	ts_canceled	timestamp	Обязательный

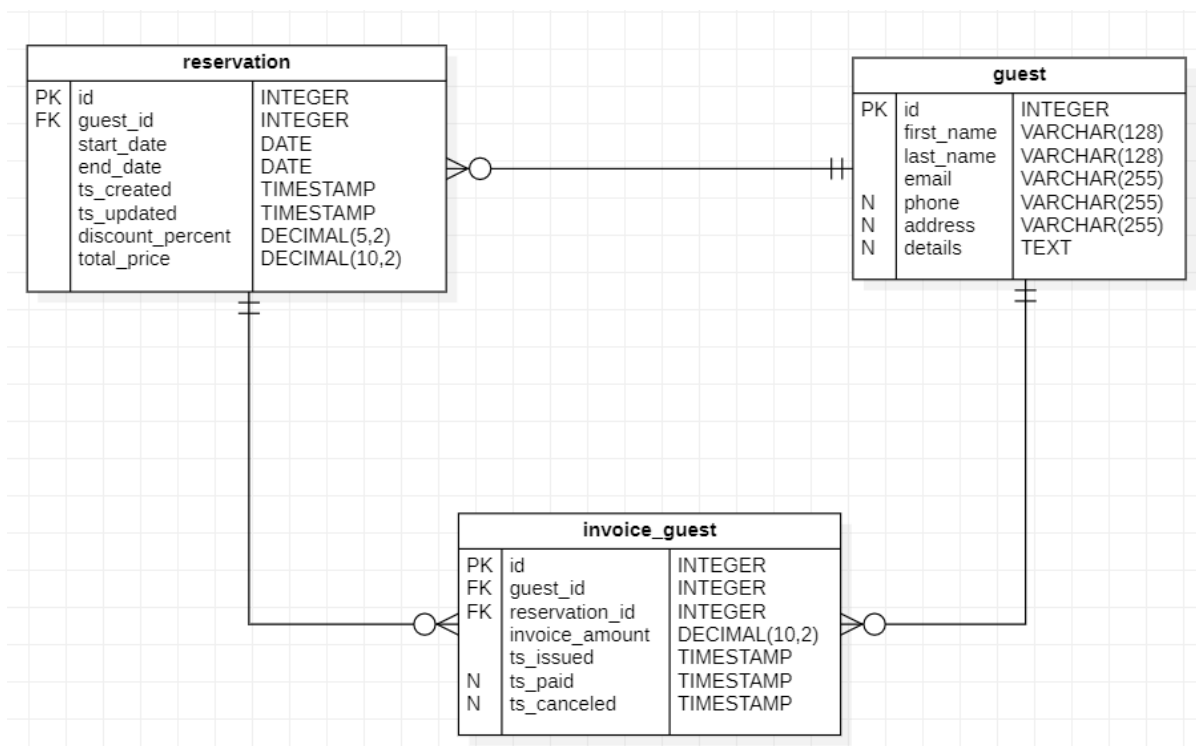


Рисунок 1.17 – Диаграмма базы данных предметного раздела гости.

1.2.2 Проектирование клиентской части (интерфейса)

Проектирование клиентской части приложения базируется на функциональных требованиях, к приложению описанных в первой части. Данные требования позволяют сформировать или отдельную страницу интерфейса или элемент страницы для его реализации.

Ниже представлена структура сайта с точки зрения реализации функционала: менеджера, анонимного гостя, авторизованного пользователя (рисунки 1.18 -1.20).

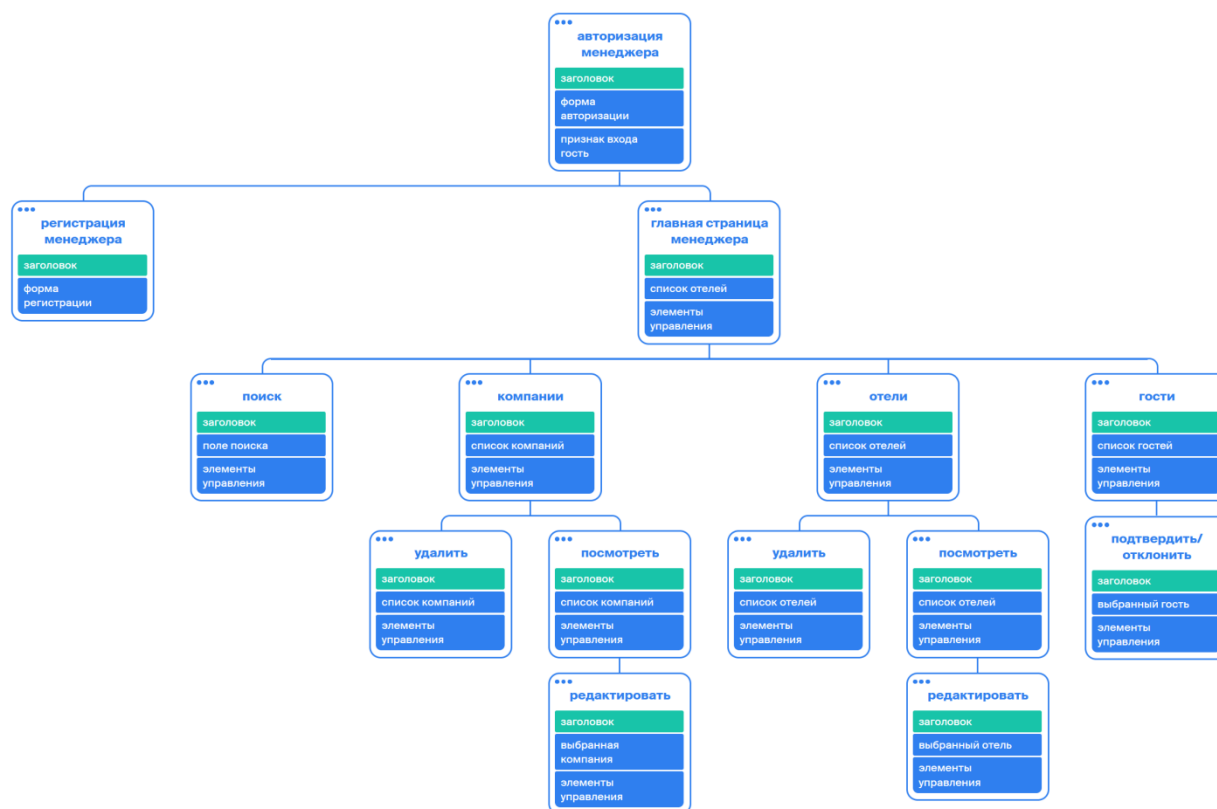


Рисунок 1.18 – Структура страниц для роли «Менеджер».

Структура и функционал страниц на рисунке 1.18. позволяют работать с объектами и функционалом доступным менеджеру:

При запуске системы менеджеру отображается страница авторизации, которая содержит в себя форму, где расположены поля ввода логина и пароля. После успешной авторизации менеджер автоматически перенаправляется на главную страницу, где расположен компонент навигации, позволяющий реализовать функционал работы с объектами: компании, отели, гости. Навигация доступна на всех страницах веб-приложения.

При необходимости менеджер имеет право зарегистрироваться в системе, для этого на странице авторизации предусмотрен вход на страницу регистрации, на которой предусмотрены поля регистрации.

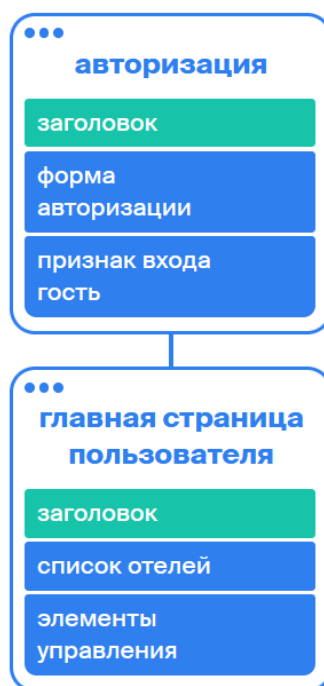


Рисунок 1.19 – Структура страниц для роли «Анонимный гость».

Структура и функционал страниц на рисунке 1.19. позволяют работать с объектами и функционалом доступным анонимному пользователю:

При запуске системы менеджеру отображается страница авторизации, которая содержит в себя форму, где расположены поля ввода логина и пароля. Данная форма содержит элемент управления, который позволяет установить признак анонимного входа.

Анонимному пользователю доступна главная страница приложения с ограниченным количеством элементов управления и информационная панель по тематике сайта.

Данный подход применен для увеличения количества зарегистрированных пользователей. С целью сбора информации о потенциальных потребителях сайта.

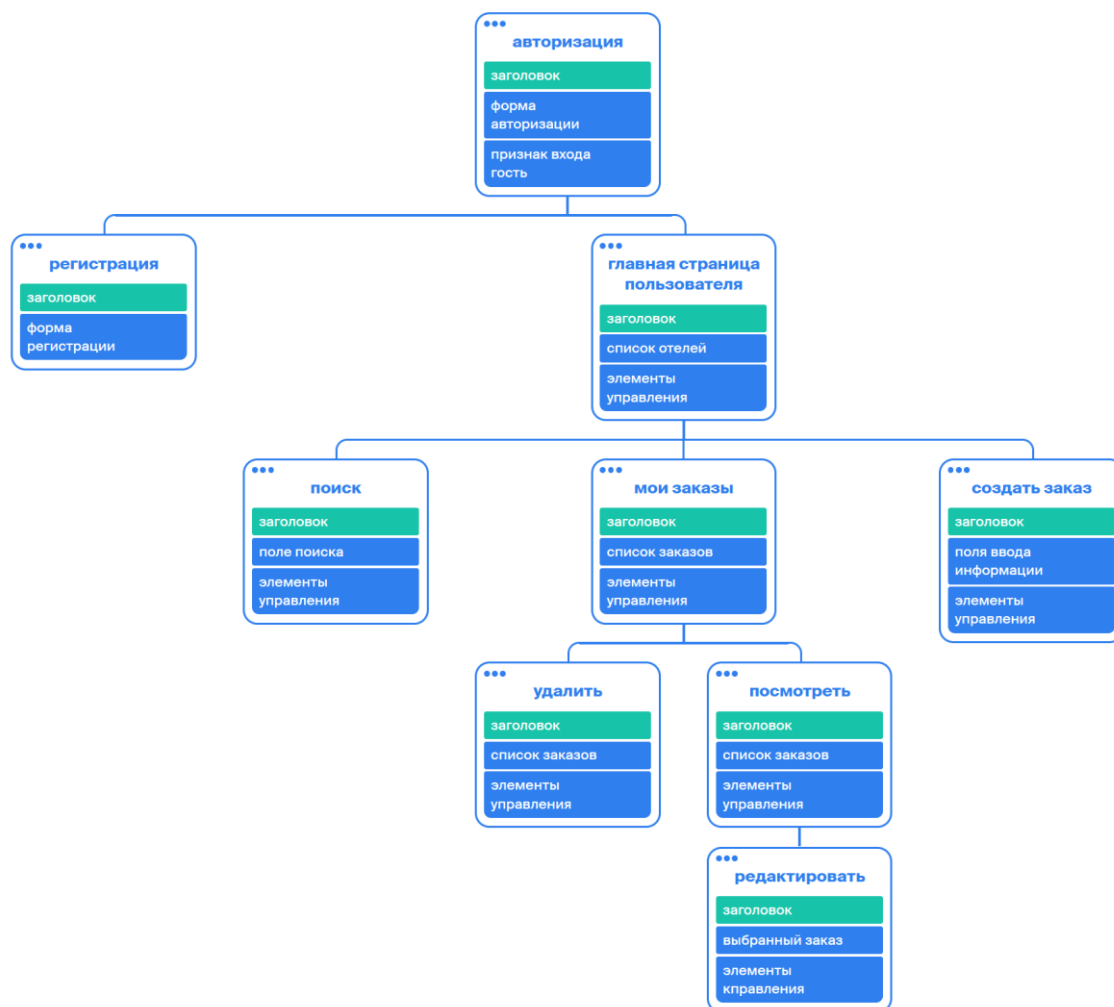


Рисунок 1.20 – Структура страниц для роли «Авторизированный гость».

Структура и функционал страниц на рисунке 1.20. позволяют работать с объектами и функционалом доступным авторизированному пользователю:

При запуске приложения пользователь попадает на страницу авторизации, после успешной авторизации пользователь попадает на главную страницу приложения.

Главная страница приложения содержит информацию об объектах недвижимости и навигационные элементы. Элементы навигации позволяют переходить на страницы с выделенным функционалом или определенной информацией об объекте недвижимости.

1.2.2.1 Проектирование клиентской части (макеты страниц)

Для реализации страниц интерфейса были разработаны макеты, на основании которых была реализована верстка интерфейса. Интерфейс в данном приложении определенная стандартами граница между взаимодействующими независимыми объектами. Интерфейс задает параметры, процедуры и характеристики взаимодействия объектов. Интерфейс пользователя - элементы и компоненты программы, которые способны оказывать влияние на взаимодействие пользователя с программным обеспечением.

Далее приводится описание страниц и их элементов с указанием типа и назначения используемых компонент.

Элементами страницы авторизации пользователей являются следующие инструменты (рисунок 1.21.):

- Форма авторизации – тип Form;
- поле ввода логина - тип Input type;
- поле ввода пароля - тип Input type;
- кнопка регистрации - тип Button;
- кнопка входа в систему тип Button;
- кнопка признака гостевого входа – тип CheckBox.

Рисунок 1.21 – Макет страницы авторизации пользователей

Элементами страницы регистрации являются следующие инструменты (рисунок 1.22.):

- форма регистрации – тип Form;
- поле ввода имени - тип Input type;
- поле ввода фамилии - тип Input type;
- поле ввода логина - тип Input type;
- поле ввода пароля - тип Input type;
- поле ввода подтверждение пароля - тип Input type;
- кнопка регистрации - тип Button;
- кнопка сброса - тип Button.

Регистрация пользователя

Имя

Фамилия

Логин

Пароль

Подтверждение пароля

Рисунок 1.22 - Макет страницы регистрации пользователя

Элементами главной страницы неавторизованного пользователя являются следующие инструменты (рисунок 1.23.):

- форма страницы – тип Form;
- ссылки навигации - тип href;

- таблица о доступных объектах недвижимости – тип table;
- фотографии объектов недвижимости – тип image;
- кнопки перехода - тип Button.

Элементами главной страницы неавторизованного пользователя являются следующие инструменты (рисунок 1.23.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица о доступных объектах недвижимости – тип table,
- фотографии объектов недвижимости – тип image,
- кнопки перехода - тип Button.

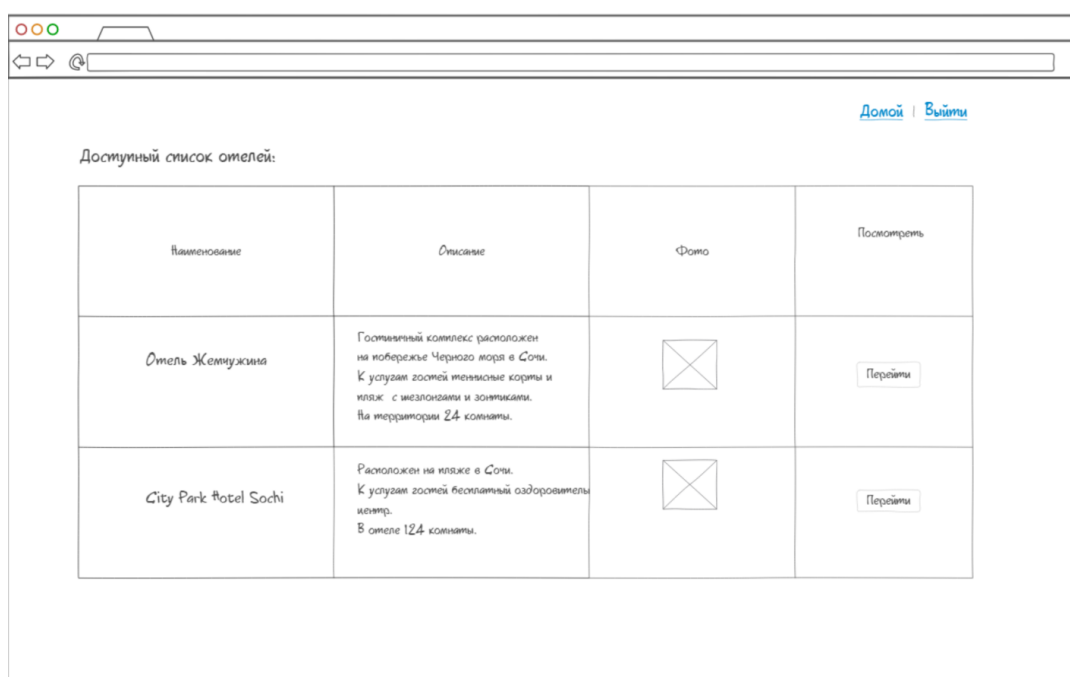


Рисунок 1.23 – Макет главной страницы неавторизованного пользователя

Элементами главной страницы авторизованного пользователя являются следующие инструменты (рисунок 1.24.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица о доступных объектах недвижимости – тип table;

- фотографии объектов недвижимости – тип image;
- кнопки создания заказа - тип Button;
- кнопки просмотра объекта недвижимости - тип Button.

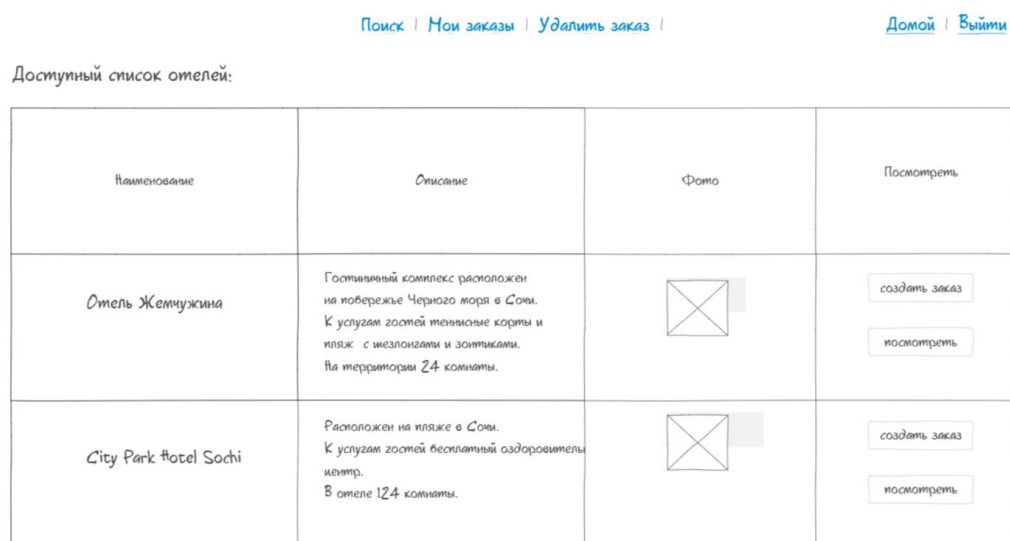


Рисунок 1.24 – Макет главной страницы авторизованного пользователя

Элементами страницы «мои заказы» являются следующие инструменты (рисунок 1.25.):

- форма страницы – тип Form;
- счетчик заказов – тип label,
- ссылки навигации - тип href;
- таблица о заказанных объектах недвижимости – тип table;
- фотографии объектов недвижимости – тип image;
- навигация периода резервирования -
- кнопка заказа - тип Button;
- кнопки отмены заказа - тип Button.

Наименование	Описание заказа	Фото	Действие
Отель Жемчужина	забронирована: 1 комната начало брони: 2.06.2023 окончание брони: 3.06.2023 стоимость 1 ночи: 2000 рублей стоимость всего: 14000 рублей		<div>удалить</div> <div>посмотреть</div>
City Park Hotel Sochi	забронирована: 1 комната начало брони: 12.07.2023 окончание брони: 14.07.2023 стоимость 1 ночи: 3000 рублей стоимость всего: 12000 рублей		<div>удалить</div> <div>посмотреть</div>

Рисунок 1.25 – Макет страницы авторизованного пользователя

«Мои заказы»

Страница поиска имеет интеграцию с Yandex API для поиска по заданным критериям (рисунок 1.26). Элементами страницы «поиск» являются следующие инструменты:

- форма страницы – тип Form;
- ссылки навигации - тип href;
- сгенерированная таблица о найденных объектах недвижимости – тип table;
- фотографии объектов недвижимости – тип image;
- кнопки просмотра объекта недвижимости - тип Button;
- поисковая строка – тип Input type;
- кнопки поиска - тип Button.

[Домой](#) | [Выйти](#)

Поиск:

Наименование	Описание объекта	Фото	Действие
City Park Hotel Sochi	<p>Расположен на пляже в Сочи.</p> <p>К услугам гостей бесплатный оздоровительный центр.</p> <p>В отеле 124 комнаты.</p>		<input type="button" value="посмотреть"/>

Рисунок 1.26 – Макет страницы авторизованного пользователя Поиск

Элементами страницы «создать заказ» являются следующие инструменты (рисунок 1.27.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица о доступном номерном фонде – тип table;
- фотографии объектов недвижимости – тип image;
- кнопки заказа - тип Button;
- кнопки отмены заказа - тип Button;
- поля определения периода бронирования (календарь) – InputType.

[создать заказ](#)
[Домой](#) | [Выйти](#)

Создать заказ Отель Жемчужина:

Даты бронирования

Дата заказа

Дата отъезда

Тип номера	Количество	Цена	Заказать	Заказано
Стандарт	12	20000	<input type="button" value="Да"/> <input type="button" value="Нет"/>	0
Люкс	64	25000	<input type="button" value="Да"/> <input type="button" value="Нет"/>	10

Рисунок 1.27 – Макет страницы авторизованного пользователя «Создать заказ»

Элементами страницы «Посмотреть информацию об объекте» являются следующие инструменты (рисунок 1.28.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица с информацией в текстовом виде – тип table;
- фотографии объектов недвижимости – тип image.

[Домой](#) | [Выйти](#)

Посмотреть *Отель Жемчужина*:

Информация об объекте недвижимости

Компания: ООО "Турист"

Наименование: *Отель Жемчужина*

Страна: Россия

Город: Сочи

Адрес: Московский переулок, дом 2

Почтовый индекс: 354000

Номерной фонд – ит:

Стандарт: 20

Люкс: 64

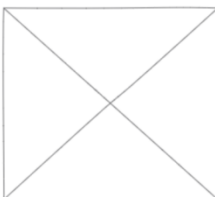



Рисунок 1.28 – Макет страницы авторизованного пользователя
(Посмотреть информацию об объекте)

Элементами страницы «Посмотреть информацию об объекте» являются следующие инструменты (рисунок 1.29.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица с информацией в текстовом виде – тип table;
- фотографии объектов недвижимости – тип image.

Доступные заказы – 2 шт.:

Наименование	Описание заказа	Фото	Действие
Отель Жемчужина	забронировано: 1 комната начало брони: 2.06.2023 окончание брони: 3.06.2023 стоимость 1 ночи: 2000 рублей стоимость всего: 14000 рублей		<input type="button" value="удалить"/>
City Park Hotel Sochi	забронировано: 1 комната начало брони: 12.07.2023 окончание брони: 14.07.2023 стоимость 1 ночи: 3000 рублей стоимость всего: 12000 рублей		<input type="button" value="удалить"/>

Рисунок 1.29 – Макет страницы авторизованного пользователя «Удалить заказ»

Элементами страницы «Редактировать заказ» являются следующие инструменты (рисунок 1.30.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица с информацией о доступных заказах – тип table;
- фотографии объектов недвижимости – тип image;
- кнопки редактирования заказа - тип Button.

Доступные заказы – 2 шт.:

Редактировать заказы – 2 шт.:

Наименование	Описание заказа	Фото	Действие
Отель Жемчужина	забронировано: 1 комната начало брони: 2.06.2023 окончание брони: 3.06.2023 стоимость 1 ночи: 2000 рублей стоимость всего: 14000 рублей		<input type="button" value="редактировать"/>
City Park Hotel Sochi	забронировано: 1 комната начало брони: 12.07.2023 окончание брони: 14.07.2023 стоимость 1 ночи: 3000 рублей стоимость всего: 12000 рублей		<input type="button" value="редактировать"/>

Рисунок 1.30 – Макет страницы авторизованного пользователя Редактировать заказ

Элементами страницы «Редактировать выбранный заказ» являются следующие инструменты (рисунок 1.31.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица о доступном номерном фонде – тип table;
- фотографии объектов недвижимости – тип image;
- кнопки заказа - тип Button;
- кнопки отмены заказа - тип Button;
- поля определения периода бронирования (календарь) – InputType.

[Домой](#) | [Выйти](#)

Редактировать заказ *Отель Жемчужина.*:

Даты бронирования

Дата заезда Дата отъезда

Тип номера	Количество	Цена	Заказать		Заказано
Стандарт	12	2000	<input type="button" value="Да"/>	<input type="button" value="Отмена"/>	0
Люкс	64	2500	<input type="button" value="Да"/>	<input type="button" value="Отмена"/>	10

Рисунок 1.31 – Редактировать выбранный заказ

Элементами главной страницы менеджера являются следующие инструменты (рисунок 1.32.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица управления компаниями, отелями гостями – тип table;
- кнопки перехода на списки компаний, отелей и гостей - тип Button;
- информация о количестве зарегистрированных компаний, отелей и гостей – тип text.

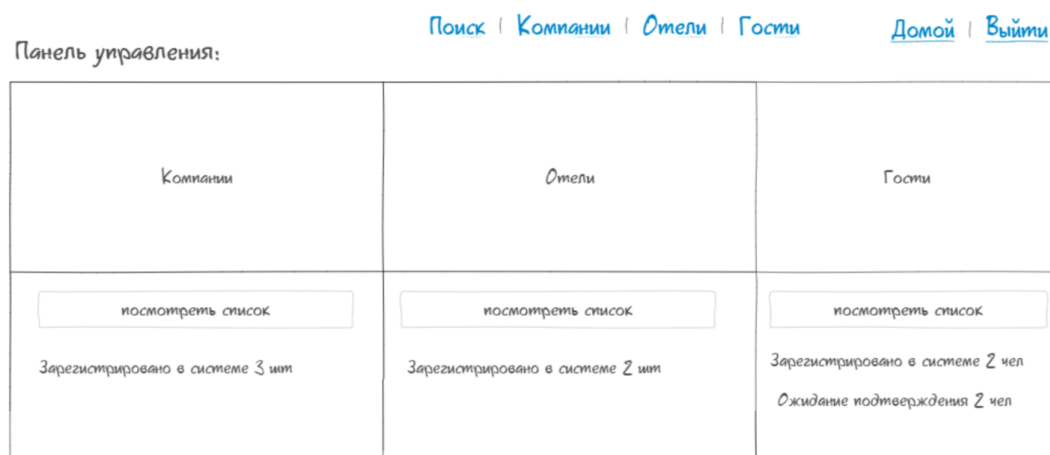


Рисунок 1.32 – Макет страницы главной страницы менеджера

Элементами страницы «Компании» являются следующие инструменты (рисунок 1.33.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица управления компаниями – тип table;
- кнопка посмотреть информацию о компании - тип Button;
- кнопка удалить информацию о компании - тип Button.

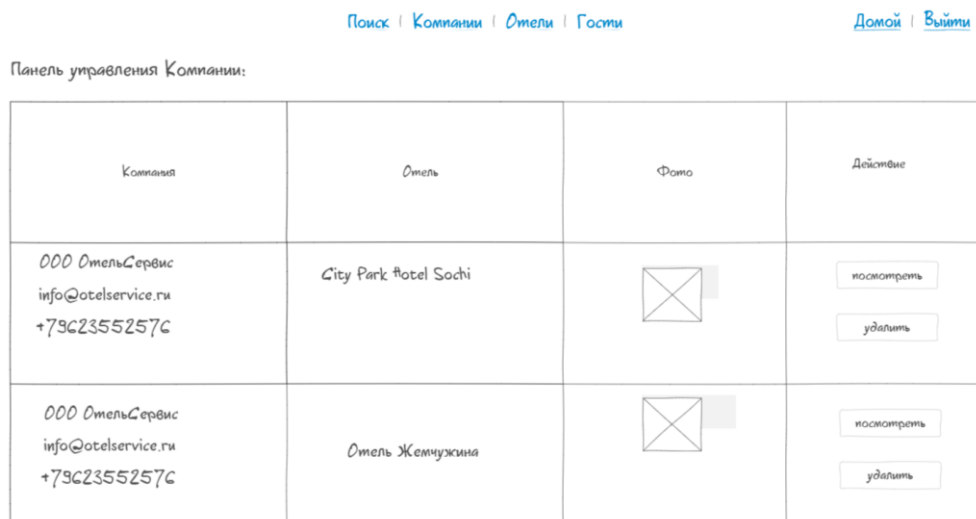


Рисунок 1.33 – Макет страницы главной страницы менеджера (компания)

Элементами страницы «Отели» являются следующие инструменты (рисунок 1.34.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица управления компаниями – тип table;
- кнопка посмотреть информацию об отеле - тип Button;
- кнопка удалить информацию о отеле - тип Button.

[Поиск](#) | [Компании](#) | [Отели](#) | [Гости](#)
[Домой](#) | [Выйти](#)

Панель управления отелями:

Отель	Описание	Фото	Действие
Отель Жемчужина	Гостиничный комплекс расположен на побережье Черного моря в Сочи. К услугам гостей теннисные корты и пляж с шезлонгами и зонтиками. На территории 24 комнаты.		<div>посмотреть</div> <div>удалить</div>
City Park Hotel Sochi	Расположен на пляже в Сочи. К услугам гостей бесплатный оздоровительный центр. В отеле 124 комнаты.		<div>посмотреть</div> <div>удалить</div>

Рисунок 1.34 – Макет страницы главной страницы менеджера (отели)

Элементами страницы «Гости» являются следующие инструменты (рисунок 1.35.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица управления потенциальными гостями – тип table;
- кнопка подтвердить регистрацию гостя - тип Button;
- кнопка отклонить регистрацию гостя - тип Button.

[Поиск](#) | [Компании](#) | [Отели](#) | [Гости](#)

[Домой](#) | [Выйти](#)

Панель управления Гости:

Гость	Отель	Фото	Действие
Олег Петров oleg_petrov@mail.ru +79623552576	City Park Hotel Sochi Стандарт 1 шт дата начала: 10.12.2023 дата окончания: 15.12.2023		<input type="button" value="подтвердить"/> <input type="button" value="отклонить"/>
Олег Петров oleg_petrov@mail.ru +79623552576	City Park Hotel Sochi Люкс 1 шт дата начала: 10.09.2023 дата окончания: 15.09.2023		<input type="button" value="подтвердить"/> <input type="button" value="отклонить"/>

Рисунок 1.35 – Макет страницы главной страницы менеджера (гости)

Элементами страницы «Гости» являются следующие инструменты (рисунок 1.35.):

- форма страницы – тип Form;
- ссылки навигации - тип href;
- таблица управления потенциальными гостями – тип table;
- кнопка подтвердить регистрацию гостя - тип Button;
- кнопка отклонить регистрацию гостя - тип Button.

1.2.3 Проектирование и реализация серверной части приложения

Для реализации основной логики приложения требуется:

- создать базовый каркас MVC приложения, со всеми настроенными зависимостями;
- реализовать набор модулей вид, для ввода и визуализации информации;
- реализовать маршруты и контроллеры для обработки запросов пользователей;
- реализовать модели данных для связи бизнес логики приложения и СУБД.

1.2.3.1 Создание базового MVC приложения

Для создания базового приложения требуется выполнить команды шаблонизатора, запустив команду: `express hotel_booking --view=pug`

В результате будет локально создан каталог `hotel_booking` и подкаталоги проектных файлов.

```
create : hotel_booking
create : hotel_booking /package.json
create : hotel_booking /app.js
create : hotel_booking /public
create : hotel_booking /public/javascripts
create : hotel_booking /routes
create : hotel_booking /views
create : hotel_booking /controllers
create : hotel_booking /models
```

После инструкций описанных выше инструкций можно локально запустить в браузере шаблонное приложение (рисунок 1.36)

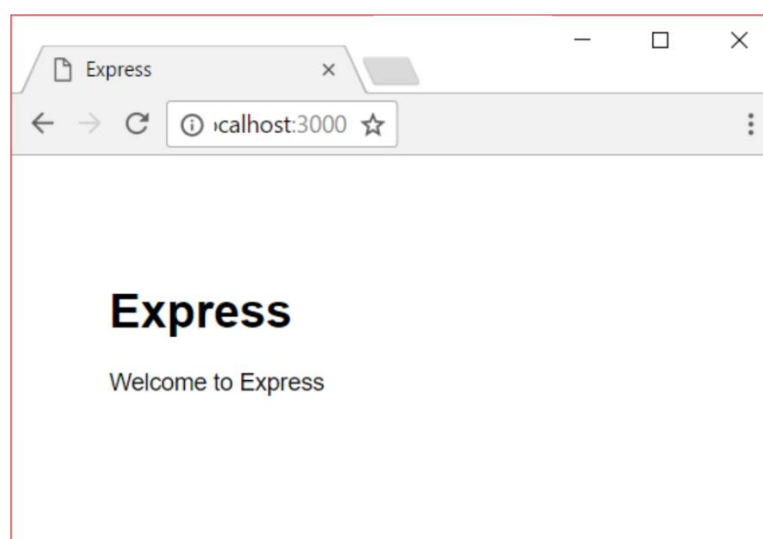


Рисунок 1.36 – Начальное приложение.

На рисунках 1.37 и 1.38 показана работа MVC приложения с точки зрения взаимодействия слоев модель, вид, контроллер и показаны какие параметры и данные слои приложения передают и обрабатывают для получения требуемого результата.

Связку модулей MVC проиллюстрируем на примере кода.

Форма «Авторизация» фиксирует введенный логин и пароль пользователя с помощью конструкции представленной ниже:

```
<div class="col-lg-4 col-md-6 col-sm-12 text-center position">
    <form action="/login" method="POST">
        <label>User Mail Address</label>
        <input type="text" name="mail" class="form-
control" placeholder="Email" required>
        <label>Password</label>
        <input type="password" name="pass"
class="form-control" placeholder="Password" required>
        <p><input type="checkbox"> Remember
Password</p>
        <input type="submit" value="Login"
class="btn btn-info btn-lg">
    </form>
</div>
```

Маршруты роутеров страницы авторизации определены в файле /routes/login.js и направляют запрос на соответствующий контроллер router.get('/', user_login.index)

Данный маршрут способен вызвать контролер проверки логина и пароля который представлен следующим кодом:

```
exports.getLogin = (req, res, next) => {
    res.render('user/loginAccount', { user: "", msg: [], err: []
});
}

exports.postLogin = (req, res, next) => {
    var connectDB = mysql.createConnection({
        host: "localhost",
        user: "root",
        password: "",
        database: "hotel"
    });
    data = "SELECT * " +
        "FROM user " +
        "WHERE email = " + mysql.escape(req.body.mail) +
        " AND password = " + mysql.escape(req.body.pass);
    connectDB.query(data, (err, result) => {
        if (err) throw err;
        else {
            res.render('user/loginAccount', { user: "", msg: [],
err: ["Please Check Your information again"] });
        }
    })
}
```

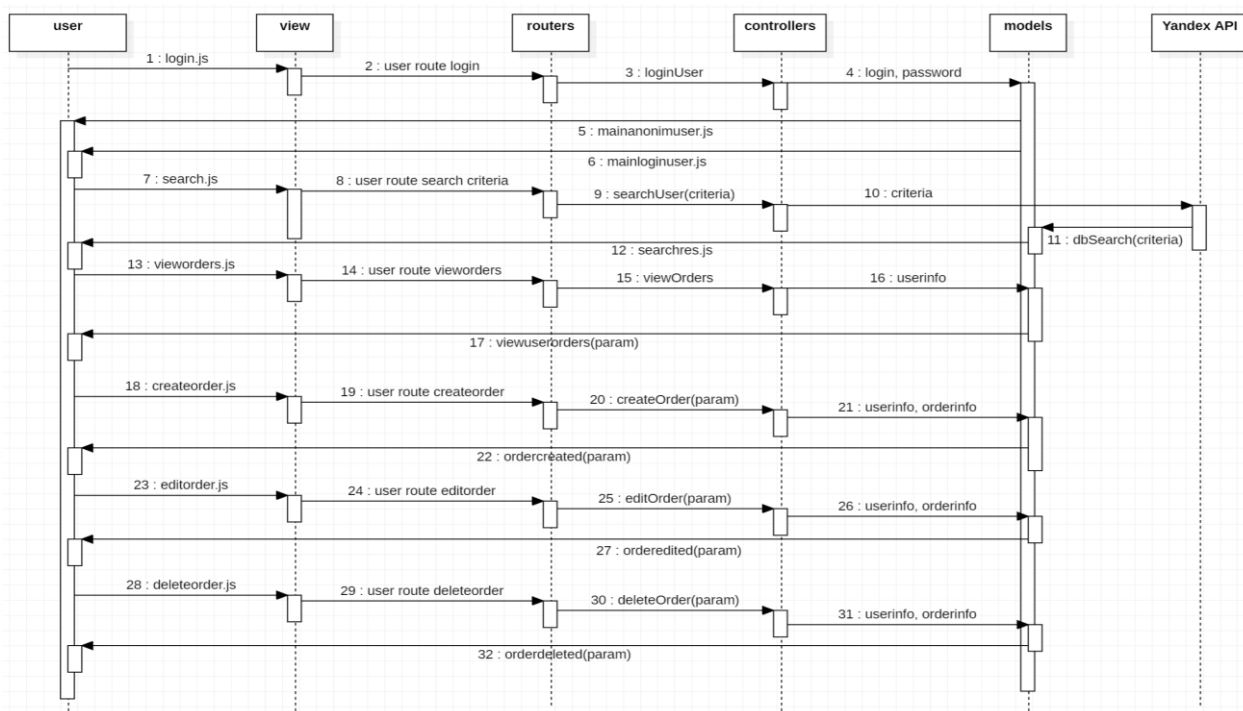


Рисунок 1.37 – MVC взаимодействие слоев приложения (пользователь и ГОСТЬ

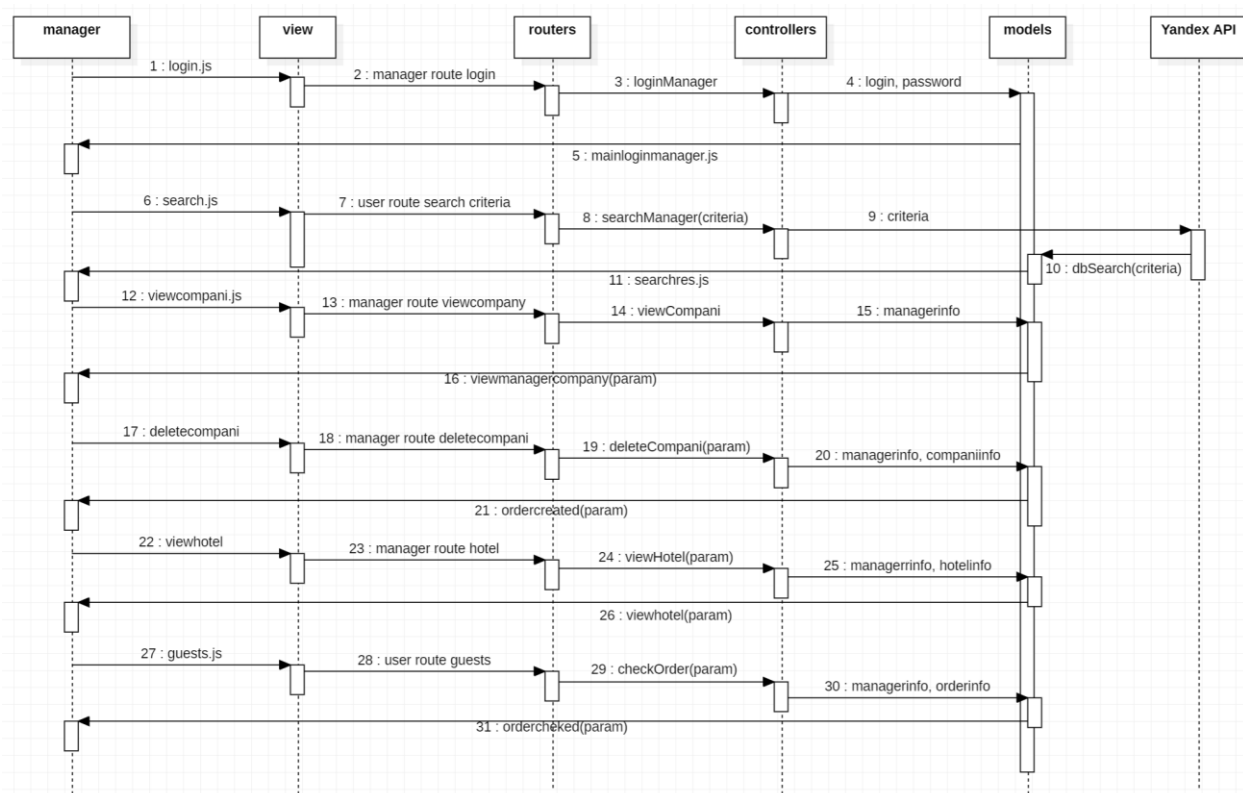


Рисунок 1.38 – MVC взаимодействие слоев приложения (менеджер)

В результате будут осуществлен вход на главную страницу приложения авторизованного пользователя системы (рисунок 1.39).

Рисунок 1.39 – Результат работы MVC маршрута login.

Аналогичным образом реализованы все функции приложения.

В Приложении А представлен полный листинг кода системы с комментариями.

1.2.4 Выводы по подпункту

В результате проектирования и реализации программного обеспечения во втором подпункте проведены следующие виды работы:

- спроектирована и реализована программная часть приложения (листинг кода приведен в Приложении В);
- спроектирована и реализована СУБД приложения (примеры работоспособных запросов SQL приведены в Приложении С).

1.3 Технология отладки клиент серверных приложений

Для инсталляции пакета разработки *NodeJS* на рабочую станцию под управлением *Windows* требуется скачать с официального сайта разработчиков требуемую версию инсталлятора (рисунок 1.40).

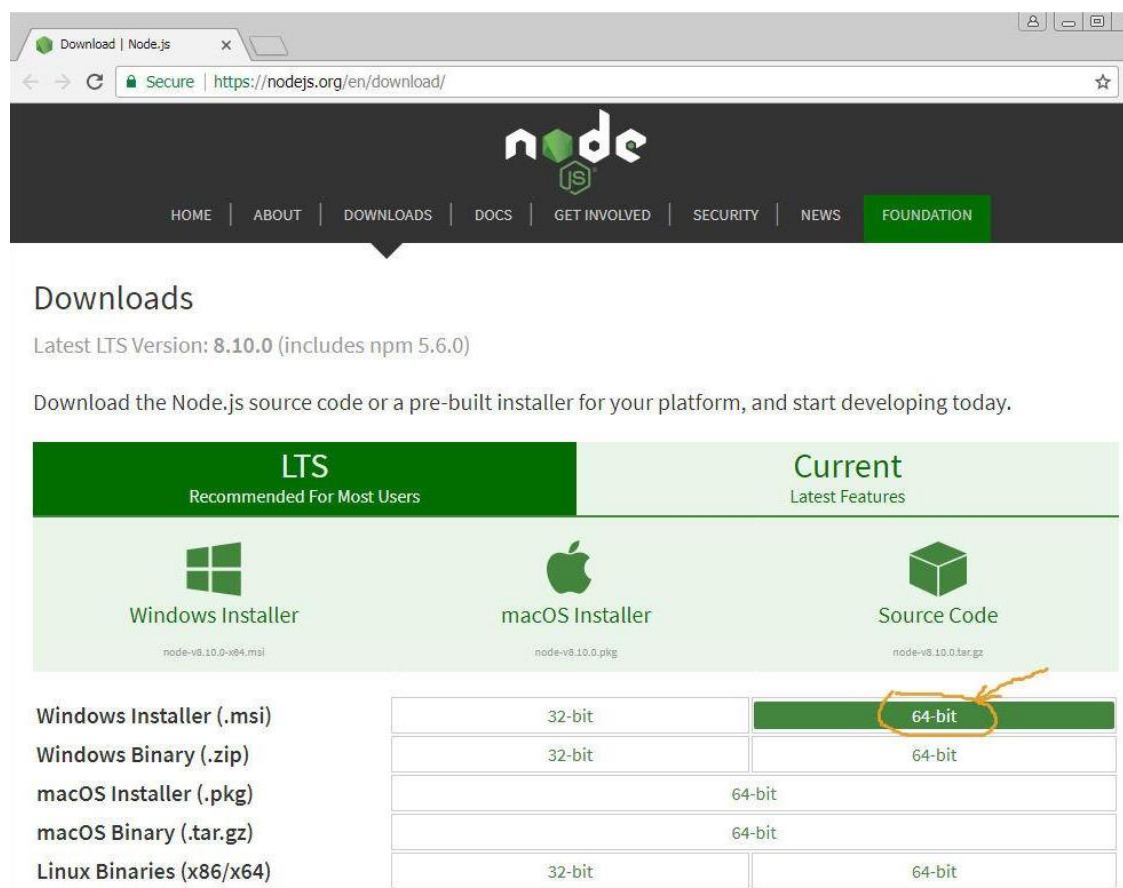


Рисунок 1.40 - Панель «Advanced system settings»

Для того, разработки приложений в операционной системе *Windows* понадобится программа *Visual C++ Redistributable for Visual Studio 2019*, которую можно скачать с официального сайта компании *Microsoft*. Установка этой программы осуществляется установщиком с диалоговым окном, нужно просто нажимать кнопку «Далее».

После установки *Visual C++ Redistributable* скачиваем инсталлятор *NodeJS* официального сайта.



Рисунок 1.41 – Процесс установки *NodeJS* шаг 1

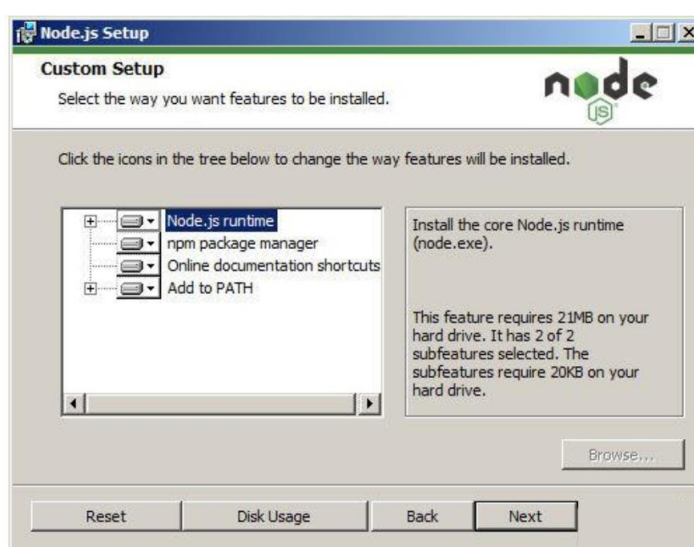


Рисунок 1.42 – Процесс установки *NodeJS* шаг 2

После установки *NodeJS* нужно добавить в системную переменную *PATH* для того, чтобы можно было запускать модули *NodeJS* из любого каталога, для этого:

- нажимаем *win+x* и переходим в настройки *System* (рисунок 1.43);
- переходим в *Advanced system settings* (рисунок 1.44);
- выбираем *Environment Variables* (рисунок 1.45) ;
- выбираем переменную *PATH* и нажимаем кнопку *Edit* (рисунок 1.46)
- нажимаем кнопку *New* и вводим адрес каталога с распакованным *NodeJS*, например *C:\NodeJS*,

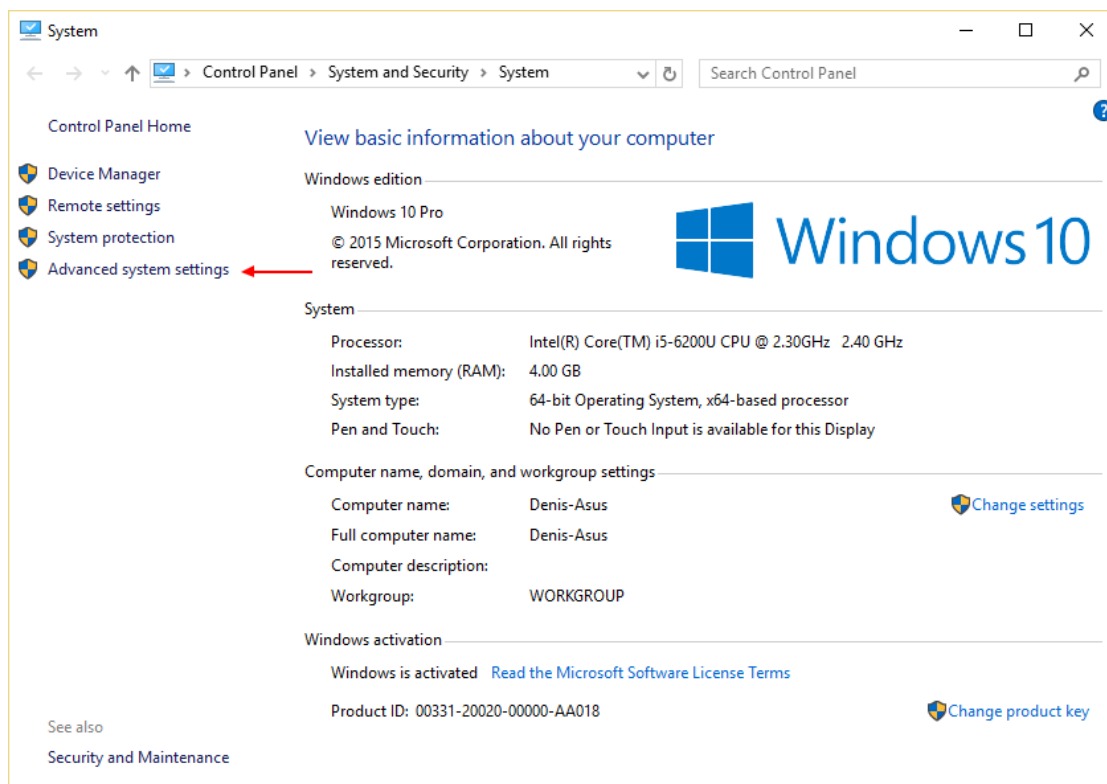


Рисунок 1.43 - Панель «Advanced system settings»

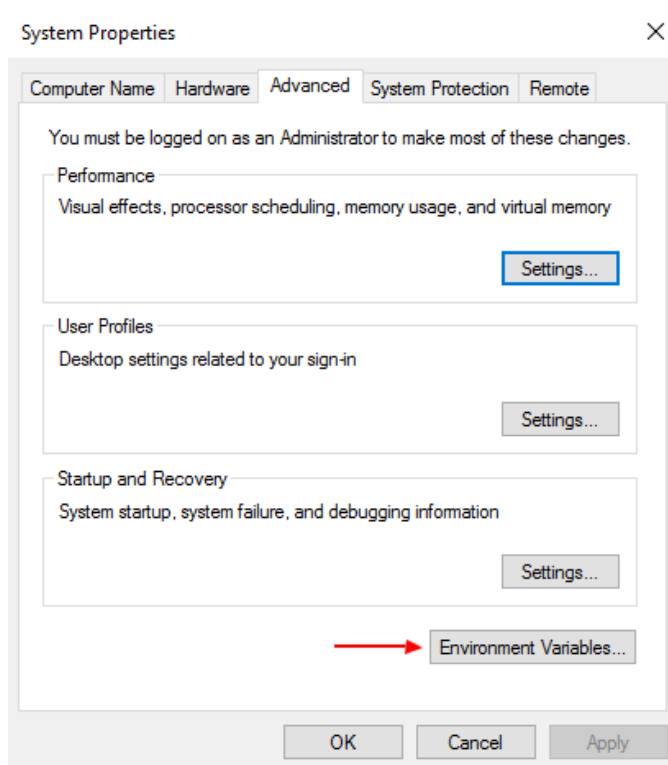


Рисунок 1.44 - Диалог Environment Variables

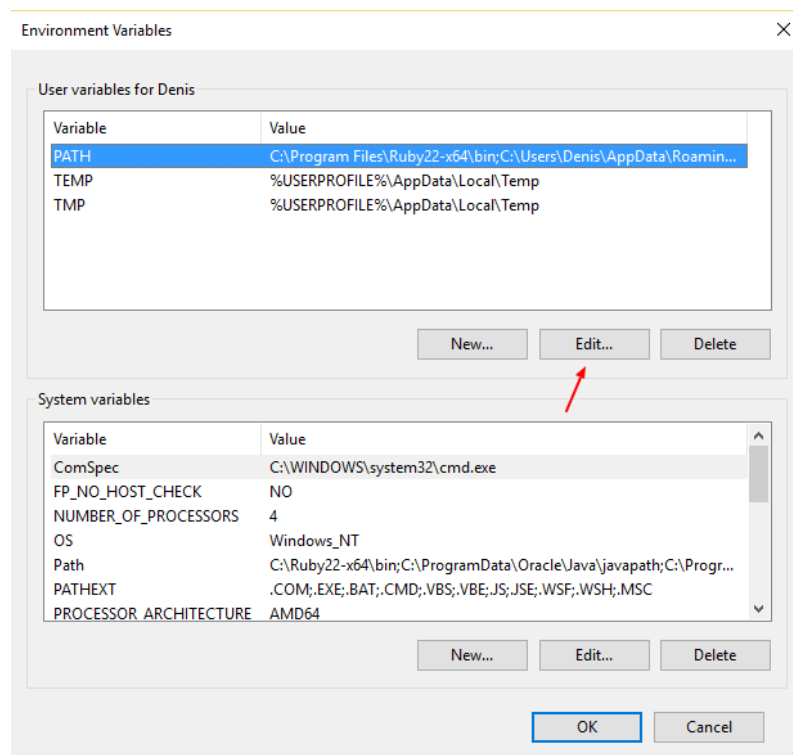


Рисунок 1.45 - Переменная *PATH*

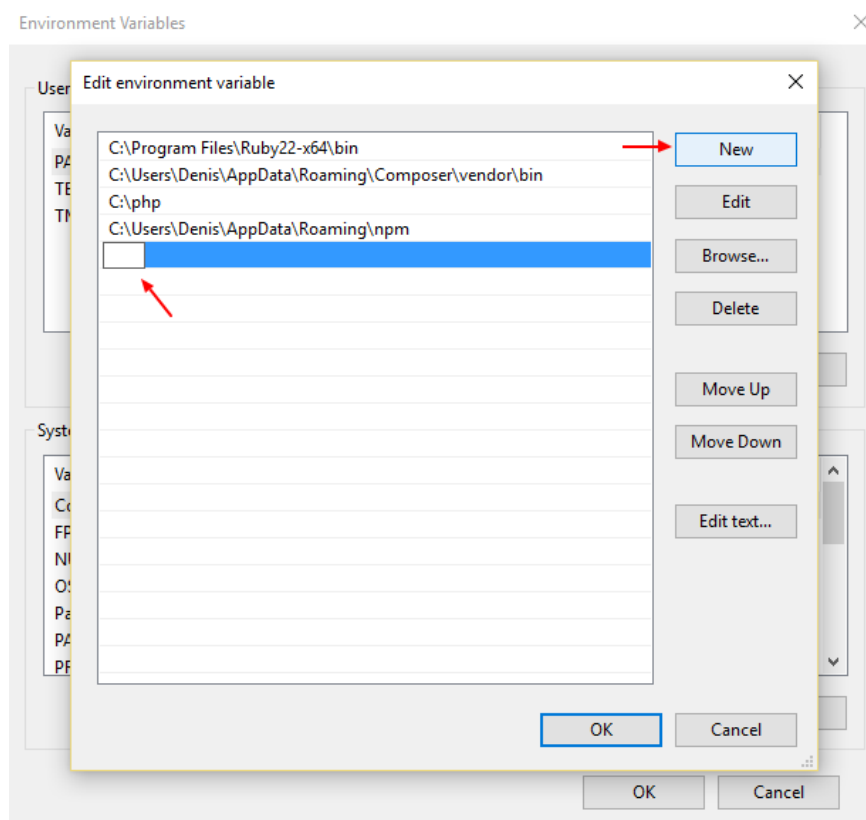


Рисунок 1.46 - Редактирование переменной *PATH*

Теперь нужно настроить *NodeJS* для работы с СУБД *MySQL*. В файле настроек *node.ini-production* требуется убрать комментарии (убираем знак ;) перед строчками:

- extension=node_curl.dll;
- extension= node _gd2.dll;
- extension= node _mbstring.dll;
- extension= node _openssl.dll;
- extension= node _pdo_mysql.dll;
- extension= node _pdo_sqlite.dll;
- extension= node _sqlite3.dll.

1.3.1 Установка и проверка работоспособности npm

Так как *npm* является частью экосистемы *NodeJS*, то для того, чтобы получить доступ к *NPM* нужно установить *NodeJS*.

У *NodeJS* есть официальный установщик для *Windows*, поэтому потребуется:

скачать установщик с официального сайта *NodeJS*;

запустить скаченный установщик и нажать кнопку *Next*, попутно соглашаясь с условиями эксплуатации.

Для того, чтобы проверить, установился ли *NodeJS* и *NPM*, откройте консоль *Windows* выполните команды: `node -v` и `npm -v` (рисунок 5).

Если команды выдали версии установленного программного обеспечения, то это значит, что всё успешно установлено и можно работать.

1.3.2 Установка и проверка работоспособности composer

Composer - пакетный менеджер уровня приложений для языка программирования *NodeJS*. Он предоставляет средства по управлению зависимостями в JS-приложении. Для работы *Composer*, необходим уже установленный *NodeJS*.

Для установки последней версии *Composer* на операционную систему *Windows* достаточно скачать файл-установщик с официального сайта. Он автоматически запишет *Composer* в переменную окружения *PATH*, и вызов менеджера будет возможен из любой директории.

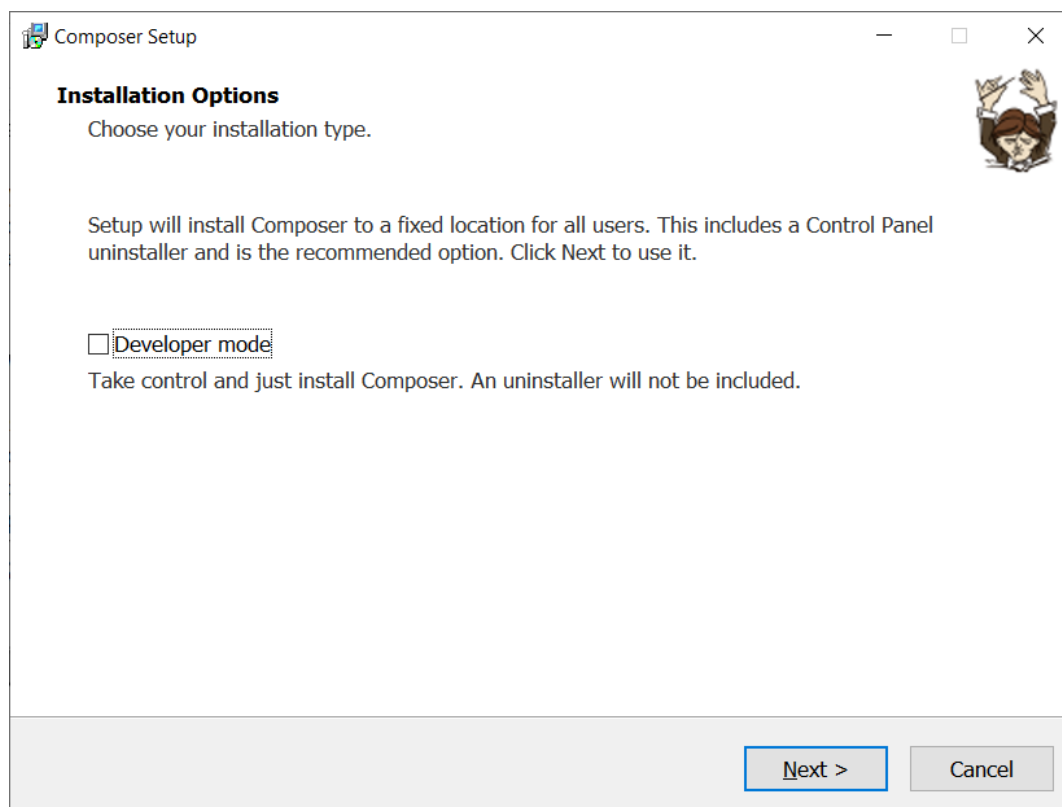


Рисунок 1.47 - Инсталляция *Composer*

Далее необходимо следовать инструкциям установщика. По завершению установки нужно запустить команду *composer* и убедиться, что *Composer* заработал корректно в командной строке (рисунок 1.48).

```

Composer
composer version 1.9.3 2020-02-04 12:58:49

usage:
  command [options] [arguments]

options:
  -h, --help                Display this help message
  -q, --quiet                Do not output any message
  -V, --version              Display this application version
  --ansi                     Force ANSI output
  --no-ansi                  Disable ANSI output
  -n, --no-interaction       Do not ask any interactive question
  --profile                  Display timing and memory usage information
  --no-plugins               Whether to disable plugins.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  --no-cache                 Prevent use of the cache
  -vv|vvv, --verbose         Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

available commands:
  about      Shows the short information about Composer.
  archive    Creates an archive of this composer package.
  browse     Opens the package's repository URL or homepage in your browser.
  check-platform-reqs Check that platform requirements are satisfied.
  clear-cache Clears composer's internal package cache.
  clearcache Clears composer's internal package cache.
  config     Sets config options.
  create-project Creates new project from a package into given directory.
  depends    Shows which packages cause the given package to be installed.
  diagnose   Diagnoses the system to identify common errors.
  dump-autoload Dumps the autoloader.
  dumpautoload Dumps the autoloader.
  exec       Executes a vendored binary/script.
  global     Allows running commands in the global composer dir ($COMPOSER_HOME).
  help       Displays help for a command
  home       Opens the package's repository URL or homepage in your browser.
  i          Installs the project dependencies from the composer.lock file if present, or falls back on the composer.json.
  info       Shows information about packages.
  init       Creates a basic composer.json file in current directory.
  install    Installs the project dependencies from the composer.lock file if present, or falls back on the composer.json.
  licenses   Shows information about licenses of dependencies.
  list       Lists commands
  outdated   Shows a list of installed packages that have updates available, including their latest version.
  prohibits  Shows which packages prevent the given package from being installed.
  remove     Removes a package from the require or require-dev.
  require    Adds required packages to your composer.json and installs them.
  run        Runs the scripts defined in composer.json.
  run-script Runs the scripts defined in composer.json.
  search     Searches for packages.
  self-update Updates composer.phar to the latest version.
  selfupdate Updates composer.phar to the latest version.
  show       Shows information about packages.
  status     Shows a list of locally modified packages, for packages installed from source.
  suggests   Shows package suggestions.
  u          Upgrades your dependencies to the latest version according to composer.json, and updates the composer.lock file.
  update     Upgrades your dependencies to the latest version according to composer.json, and updates the composer.lock file.
  upgrade    Upgrades your dependencies to the latest version according to composer.json, and updates the composer.lock file.
  validate   Validates a composer.json and composer.lock.
  why        Shows which packages cause the given package to be installed.
  why-not    Shows which packages prevent the given package from being installed.

```

Рисунок 1.48 - *Composer* в командной строке

1.3.3. Установка и проверка работоспособности OpenServer

Для установки *OpenServer* требуется скачать инсталлятор с официального сайта, пройти стандартные этапы инсталляции, после чего сервер автоматически запустится в операционной системе.

После инсталляции потребуются пройти этапы настройки сервера для работы с *NodeJS*.

Для локальной работы достаточно чтобы настройки были в положении показанном на рисунках 1.49, 1.50, 1.51.

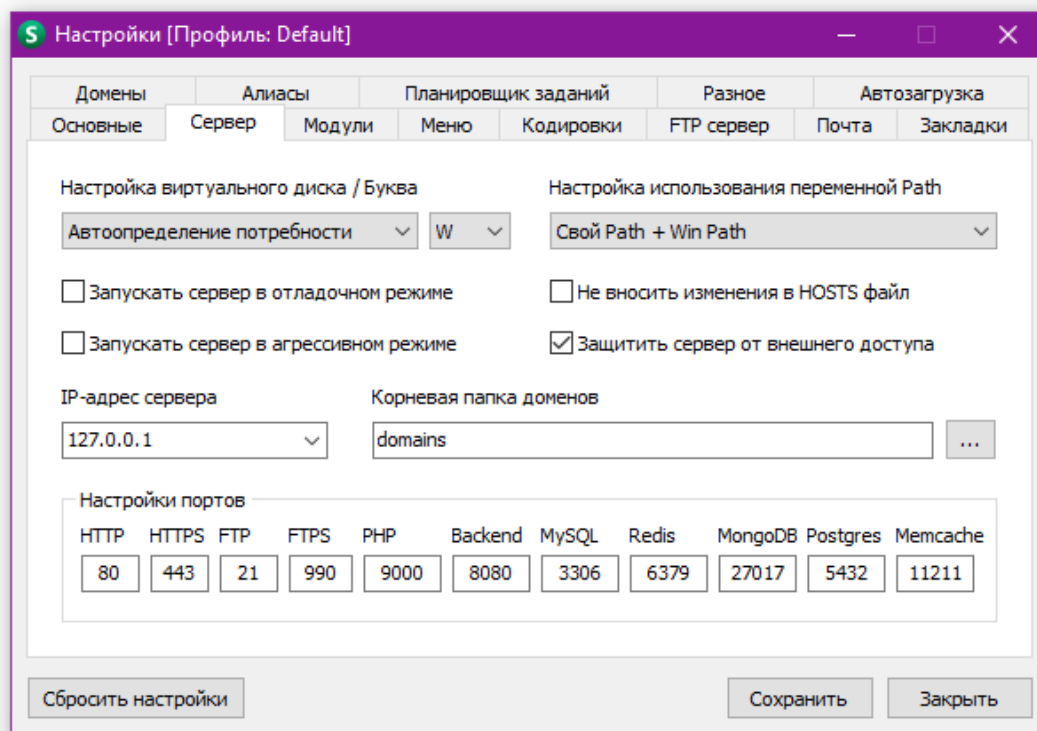


Рисунок 1.49 - Настройка *OpenServer*. Шаг 1

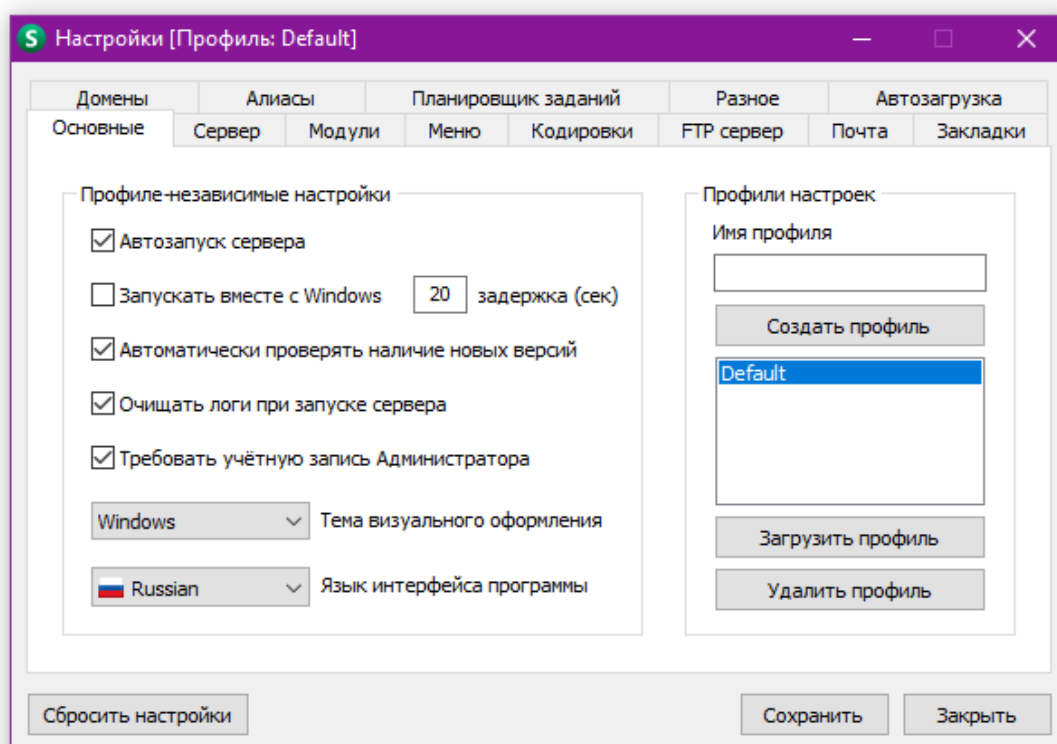


Рисунок 1.50 - Настройка *OpenServer*. Шаг 2

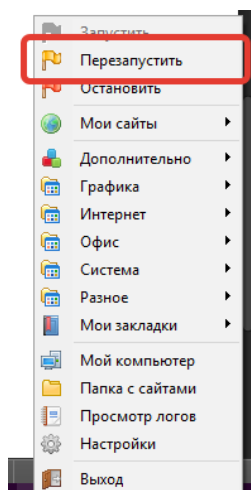


Рисунок 1.51 - Настройка *OpenServer*. Шаг 4

1.3.4 Установка и проверка работоспособности проекта

Для реализации проекта необходимо связать *VS*. Для создания проекта со структурой *NodeJS* в терминале *VS* выполняется две команды:

```
cd api
```

```
composer create-project nodejs/js/
```

После чего в *VS* формируется структура проекта в рамках которой можно реализовать свой код (рисунок 1.52).

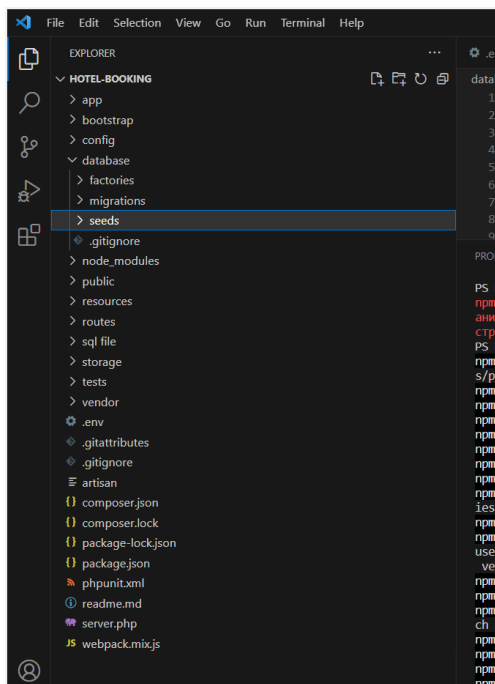


Рисунок 1.52 - Проект *NodeJS* в *VS*

Изм.	Лист	№ докум	Подпись Дата

Далее для формирования в проекте зависимостей, пакетов и связки проекта с СУБД требуется выполнить следующие команды:

`npm install`

`composer update --ignore-platform-reqs --dev`

`composer require --dev nodejs/dusk:^2.0 --ignore-platform-reqs`

Результаты исполнения команд в терминале VS показаны на рисунке 1.53.

```

Migrating: 2019_05_11_144019_add_category_rooms
Migrated: 2019_05_11_144019_add_category_rooms
PS D:\Hotel-Booking> php artisan db:seed --class=DatabaseSeeder
Seeding: CountrySeed
Seeding: RoleSeed
Seeding: UserSeed
PS D:\Hotel-Booking> php artisan serve
Laravel development server started: <http://127.0.0.1:8000>
[Fri May 19 20:12:50 2023] PHP 7.4.0 Development Server (http://127.0.0.1:8000) started
[Fri May 19 20:13:06 2023] 127.0.0.1:51451 Accepted
[Fri May 19 20:13:06 2023] 127.0.0.1:51452 Accepted
[Fri May 19 20:13:07 2023] 127.0.0.1:51452 Closing
[Fri May 19 20:13:08 2023] 127.0.0.1:51451 Closing
[Fri May 19 20:13:08 2023] 127.0.0.1:51459 Accepted
[Fri May 19 20:13:08 2023] 127.0.0.1:51459 Closing
[Fri May 19 20:13:09 2023] 127.0.0.1:51461 Accepted
[Fri May 19 20:13:09 2023] 127.0.0.1:51461 [200]: GET /adminlte/bootstrap/css/bootstrap.min.css
[Fri May 19 20:13:09 2023] 127.0.0.1:51462 Accepted
[Fri May 19 20:13:09 2023] 127.0.0.1:51463 Accepted
[Fri May 19 20:13:09 2023] 127.0.0.1:51462 [200]: GET /quickadmin/css/select2.min.css
[Fri May 19 20:13:09 2023] 127.0.0.1:51464 Accepted
[Fri May 19 20:13:09 2023] 127.0.0.1:51463 [200]: GET /adminlte/css/AdminLTE.min.css
[Fri May 19 20:13:09 2023] 127.0.0.1:51461 Closing
[Fri May 19 20:13:09 2023] 127.0.0.1:51465 Accepted
[Fri May 19 20:13:09 2023] 127.0.0.1:51464 [200]: GET /adminlte/css/custom.css
[Fri May 19 20:13:09 2023] 127.0.0.1:51470 Accepted
[Fri May 19 20:13:09 2023] 127.0.0.1:51465 [200]: GET /adminlte/css/skins/skin-blue.min.css
[Fri May 19 20:13:09 2023] 127.0.0.1:51462 Closing
[Fri May 19 20:13:09 2023] 127.0.0.1:51474 Accepted
[Fri May 19 20:13:09 2023] 127.0.0.1:51470 [200]: GET /adminlte/js/bootstrap.min.js
[Fri May 19 20:13:09 2023] 127.0.0.1:51463 Closing
[Fri May 19 20:13:09 2023] 127.0.0.1:51482 Accepted
[Fri May 19 20:13:09 2023] 127.0.0.1:51474 [200]: GET /adminlte/js/select2.full.min.js
[Fri May 19 20:13:09 2023] 127.0.0.1:51464 Closing
[Fri May 19 20:13:09 2023] 127.0.0.1:51488 Accepted
[Fri May 19 20:13:09 2023] 127.0.0.1:51482 [200]: GET /adminlte/js/main.js
[Fri May 19 20:13:09 2023] 127.0.0.1:51465 Closing
  
```

Рисунок 1.53 - Запуск проекта на локальном сервере

1.3.5 Установка и проверка работоспособности СУБД

Для развертывания и настройки СУБД MySQL при работе с NodeJS проектами требуется в файле `env` проекта переменной `DB_DATABASE` присвоить наименование базы данных с которой он будет работать (рисунок 1.54).

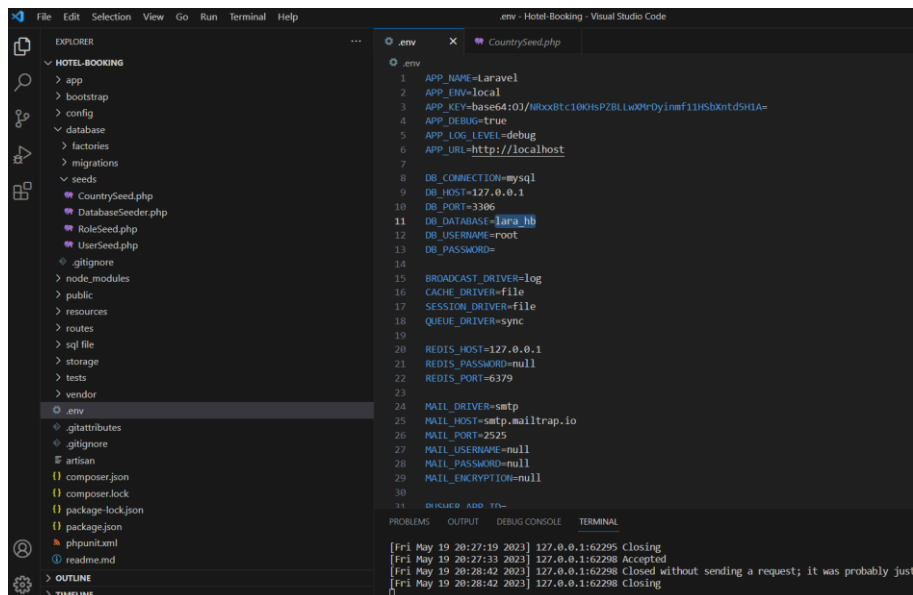


Рисунок 1.54 - Установка переменной *DB_DATABASE*

В программной среде СУБД требуется создать базу данных, с которой будет связан проект (рисунок 1.55).

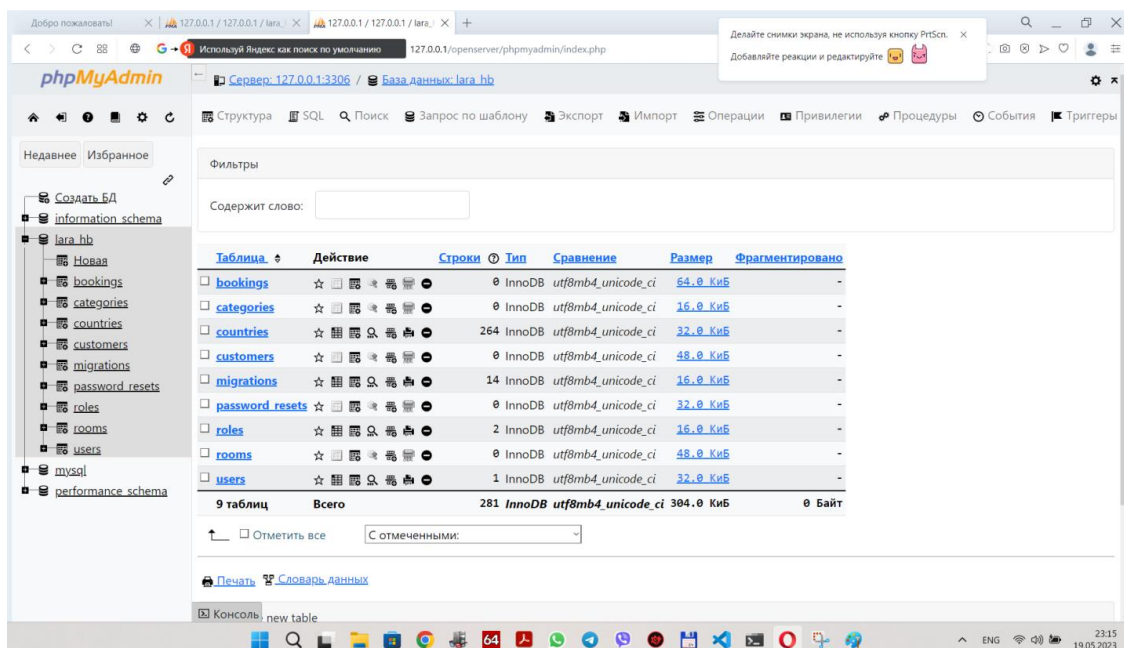


Рисунок 1.55 - Рабочее пространство БД

После всех проведенных выше настроек, можно запустить проект на исполнение и протестировать полученные результаты.

2 БЖД (рекомендации для оператора ЭВМ)

Появление современных компьютеров произвело техническую революцию во всех сферах жизни человека.

Главный недостаток современных компьютеров заключается в их негативном влиянии на здоровье человека. Для минимизации негативного влияния компьютера были разработаны требования по безопасности, при работе с ПЭВМ.

В состав данных требований входят следующие пункты:

- ПЭВМ должны соответствовать требованиям настоящих санитарных правил, и каждый их тип подлежит санитарноэпидемиологической экспертизе с оценкой в испытательных лабораториях, аккредитованных в установленном порядке;

- допустимые уровни звукового давления и уровней звука, создаваемых ПЭВМ, не должны превышать установленных норм;

- временные допустимые уровни электромагнитных полей, создаваемых ПЭВМ, не должны превышать установленных норм;

- концентрации вредных веществ, выделяемых ПЭВМ в воздух помещений, не должны превышать предельно допустимых концентраций, установленных для атмосферного воздуха;

- мощность экспозиционной дозы мягкого рентгеновского излучения в любой точке на расстоянии 0,05 м от экрана и корпуса ВДТ (на электронно-лучевой трубке) при любых положениях регулировочных устройств не должна превышать 1 мкЗв/час (100 мкР/час);

- документация на проектирование, изготовление и эксплуатацию ПЭВМ не должна противоречить требованиям настоящих Санитарных правил.

Также необходимо чтобы используемое ПЭВМ, поддерживало возможность поворота корпуса и его фиксации в установленном положении. Расцветка ПЭВМ должна состоять из спокойных цветов, не отвлекающих

				ПГУ1.090301.18.001 ПЗ	Лист
					80
Изм.	Лист	№ докум	Подпись Дата		

внимание. При организации условий труда также следует учитывать, требования к освещённости рабочего места сотрудника:

– помещения, предназначенные для размещения рабочих мест пользователей персональных компьютеров, должны иметь естественное и искусственное освещение;

– естественное освещение помещений, предназначенных для размещения рабочих мест, должно осуществляться через световые проемы, ориентированные преимущественно на север и северо-восток;

– в случае иной ориентации световых проемов, необходимо предусматривать эффективные средства регулирования интенсивности естественного освещения;

– оконные проемы в помещениях, предназначенных для использования компьютеров, необходимо оборудовать устройствами регулирования интенсивности естественного освещения типа жалюзи, занавесей, внешних козырьков и др. Уровень шума в рабочем помещении, также регулируется санитарными нормами. Зачастую главным источником шума в рабочем помещении является офисное оборудование. Для снижения подобного шума необходимо предусмотреть специализированные амортизирующие прокладки, предусмотренные нормативными документами. Уровень шума на рабочих местах не должен превышать 50 дБА.

Соблюдение описанных ранее мер является обязательным требованием для любого рабочего места, так как оно позволяет снизить уровень негативного влияния на здоровье человека при работе с ПЭВМ.

				ПГУ1.090301.18.001 ПЗ	Лист
					81
Изм.	Лист	№ докум	Подпись Дата		

Заключение

В процессе работы над ВКР было разработано программное обеспечение для поиска жилья. Использование разработки упрощает бронирование жилья клиентам приложения, а управляющим жилого фонда предоставляет информацию о потенциальных гостях.

В процессе работы были автоматизированы следующие задачи: обработка сведений об объектах недвижимости (отелях), обработка данных о параметрах конкретного объекта недвижимости, обработка сведений о пользователе, обработка сведений о датах и условиях бронирования.

В разделе анализа состояния вопроса дипломного проекта выявлена и обоснована необходимость реализации приложения.

В разделе проектирования построена информационная модель программы, и показана поэтапная реализация программы. В данном разделе было разработано программное обеспечение с учётом устранения выявленных при анализе недостатков, дано описание реализации приложения и базы данных.

В разделе безопасность жизнедеятельности были описаны рекомендации для оператора ЭВМ.

На основании вышесказанного можно сделать вывод о том, что задачи поставленные в рамках дипломного проекта выполнены полностью.

				ПГУ1.090301.18.001 ПЗ	Лист
					82
Изм.	Лист	№ докум	Подпись Дата		

Список использованных источников

1. Бибо Б. У., Кац И.М. Руководство по *JavaScript*; Символ-плюс - М., 2021. - 624 с.
2. Дронов В. Д. *NodeJs* в *Web* разработке; БХВ-Петербург - М., 2020. - 736 с.
3. Маркин С.К. React для профессионалов; БХВ-Петербург - М., 2022. - 458 с.
4. Дунаев В. А. Паттерное программирование для *WEB*; Питер - М., 2019. - 400 с.
5. Матюкин П.В. Скриптовые языки; БХВ-Петербург - М., 2018. - 816 с.

				ПГУ1.090301.18.001 ПЗ	Лист
					83
Изм.	Лист	№ докум	Подпись Дата		

Приложение А

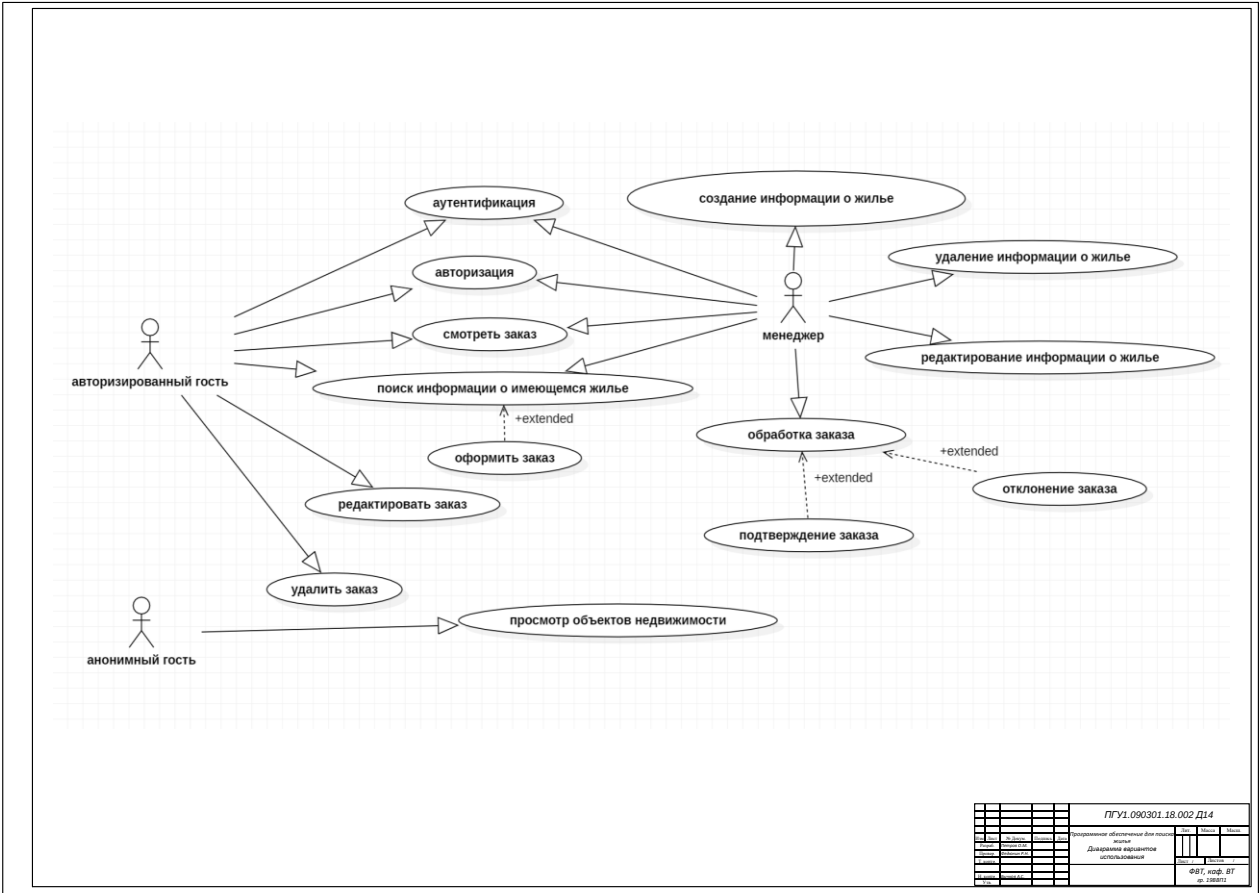
(Обязательное)

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ПОИСКА ЖИЛЬЯ

UML-диаграммы

				ПГУ1.090301.18.001 ПЗ	Лист
					84
Изм.	Лист	№ докум	Подпись Дата		

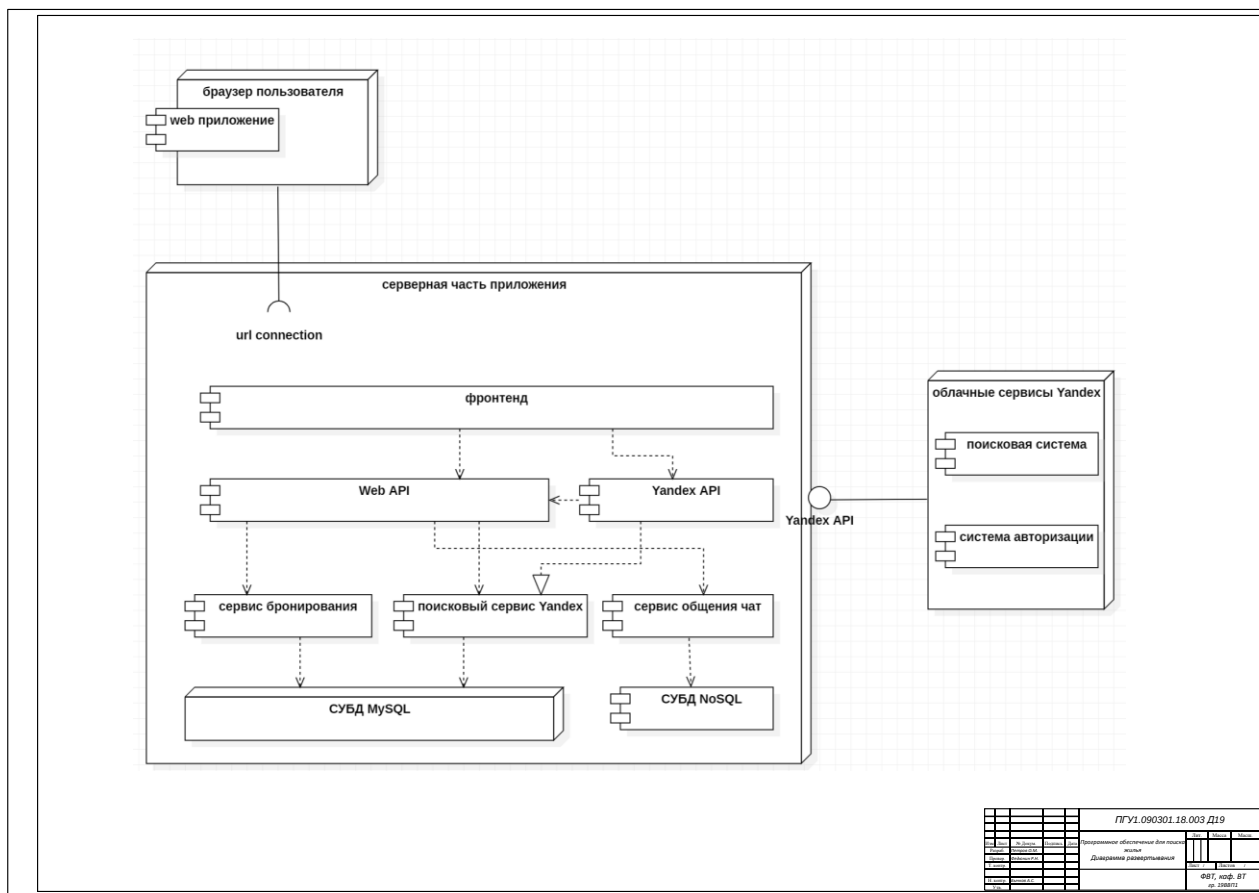
Приложение А.1 – Диаграмма вариантов использования



ПГУ1.090301.18.002 Д14			
Изм.	Лист	№ докум	Подпись Дата
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10
11	11	11	11
12	12	12	12
13	13	13	13
14	14	14	14
15	15	15	15
16	16	16	16
17	17	17	17
18	18	18	18
19	19	19	19
20	20	20	20
21	21	21	21
22	22	22	22
23	23	23	23
24	24	24	24
25	25	25	25
26	26	26	26
27	27	27	27
28	28	28	28
29	29	29	29
30	30	30	30
31	31	31	31
32	32	32	32
33	33	33	33
34	34	34	34
35	35	35	35
36	36	36	36
37	37	37	37
38	38	38	38
39	39	39	39
40	40	40	40
41	41	41	41
42	42	42	42
43	43	43	43
44	44	44	44
45	45	45	45
46	46	46	46
47	47	47	47
48	48	48	48
49	49	49	49
50	50	50	50
51	51	51	51
52	52	52	52
53	53	53	53
54	54	54	54
55	55	55	55
56	56	56	56
57	57	57	57
58	58	58	58
59	59	59	59
60	60	60	60
61	61	61	61
62	62	62	62
63	63	63	63
64	64	64	64
65	65	65	65
66	66	66	66
67	67	67	67
68	68	68	68
69	69	69	69
70	70	70	70
71	71	71	71
72	72	72	72
73	73	73	73
74	74	74	74
75	75	75	75
76	76	76	76
77	77	77	77
78	78	78	78
79	79	79	79
80	80	80	80
81	81	81	81
82	82	82	82
83	83	83	83
84	84	84	84
85	85	85	85
86	86	86	86
87	87	87	87
88	88	88	88
89	89	89	89
90	90	90	90
91	91	91	91
92	92	92	92
93	93	93	93
94	94	94	94
95	95	95	95
96	96	96	96
97	97	97	97
98	98	98	98
99	99	99	99
100	100	100	100

Изм.	Лист	№ докум	Подпись	Дата

Приложение А.2 – Диаграмма развертывания



ПГУ1.090301.18.003 Д19			
Исполнитель	Заказчик	Исполнитель	Исполнитель
Программа обеспечения для поставщика			
Доверенность на развёртывание			
ФБТ, код: БТ			
от 200001			

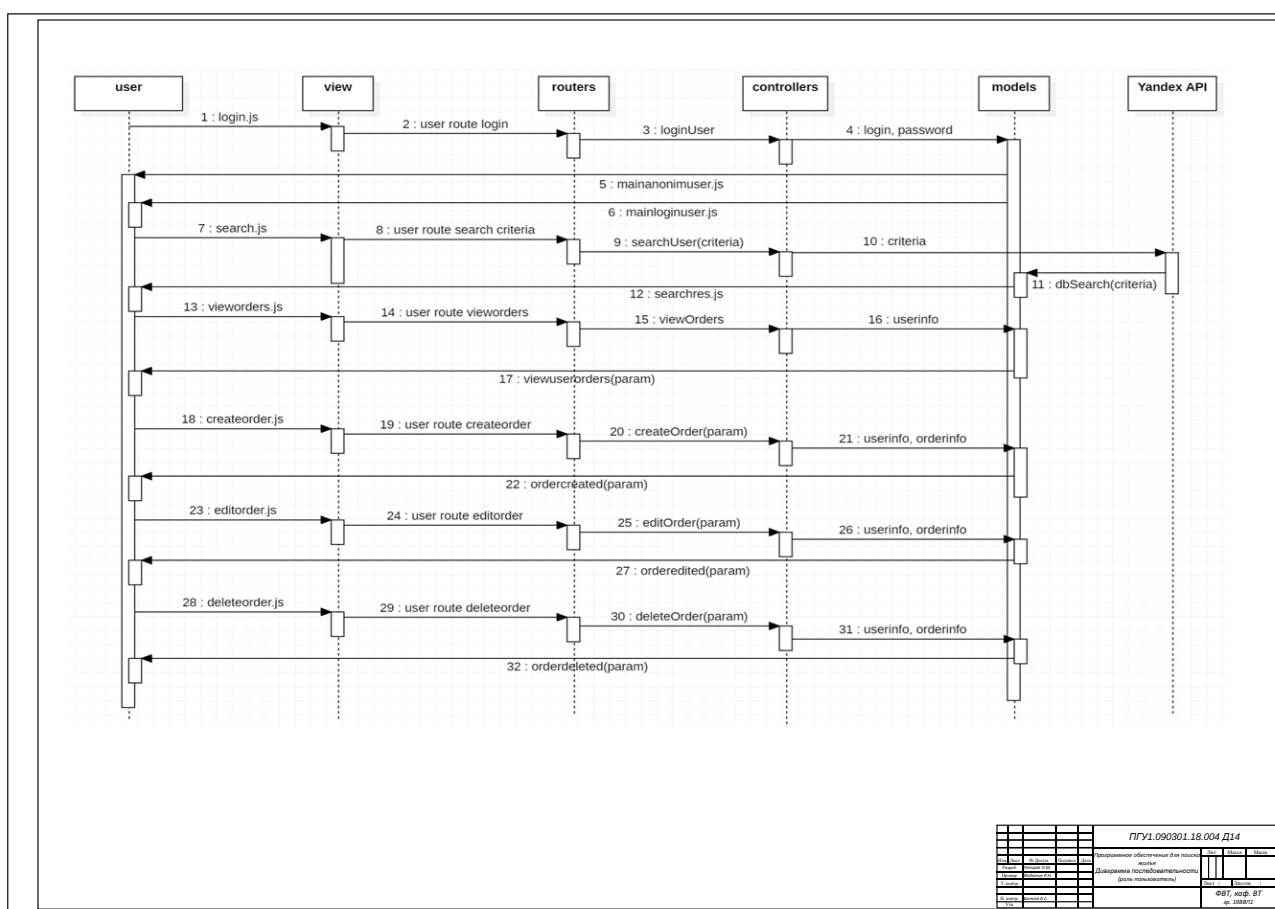
Изм.	Лист	№ докум	Подпись Дата

ПГУ1.090301.18.001 ПЗ

Лист

86

Приложение А.3 – Диаграмма развертывания



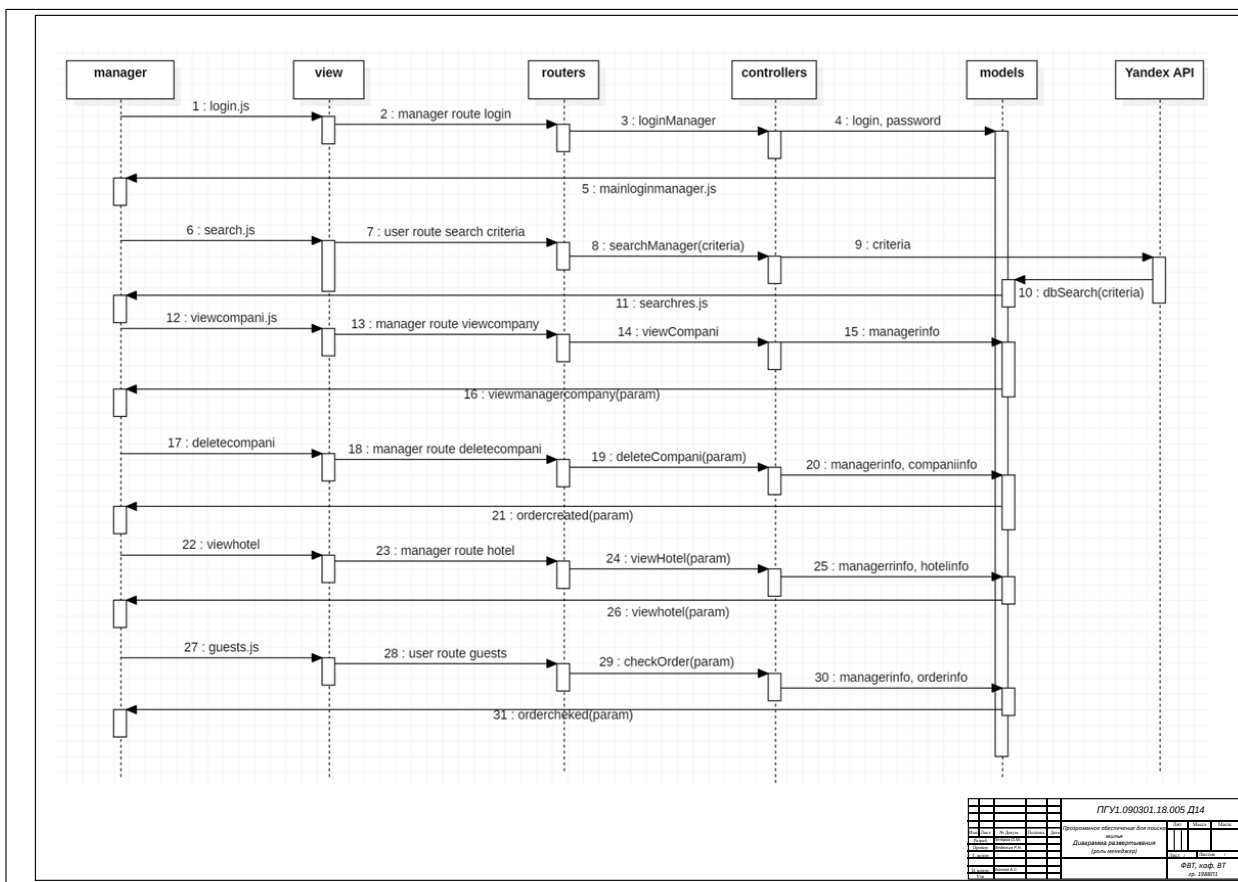
Изм.	Лист	№ докум	Подпись Дата

ПГУ1.090301.18.001 ПЗ

Лист

87

Приложение А.4 – Диаграмма развертывания



Изм.	Лист	№ докум	Подпись	Дата

ПГУ1.090301.18.001 ПЗ

Лист

88

Приложение Б

(Обязательное)

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ПОИСКА ЖИЛЬЯ

Листинг программы

				ПГУ1.090301.18.001 ПЗ	Лист
					89
Изм.	Лист	№ докум	Подпись Дата		

Модуль app.js

```
const express = require('express');
const path     = require('path');
const bodyParser = require('body-parser');
const session = require('express-session');

const app = express();

app.set('view engine', 'ejs');
app.set('views', 'views');

//модули
const userRouter = require('./routes/user');
const adminRouter = require('./routes/admin');

app.use(
  session({
    secret: 'secret',
    resave: true,
    saveUninitialized: false
  })
);

app.use(bodyParser.urlencoded({ extended: false }));
app.use(express.static(path.join(__dirname, '/public')));
app.use(userRouter);
app.use("/admin" ,adminRouter);

app.listen(3000, () => console.log("Сервер запущен..."));
```

Модуль User.js

```
const express = require('express');
const path = require('path');

const router = express.Router();
const userController = require('../controllers/user');
router.get('/', userController.getHome);
router.route('/login')
  .get(userController.getLogin)
  .post(userController.postLogin)
router.route('/createaccount')
  .get(userController.getCreateAccount)
  .post(userController.postCreateAccount);
router.route('/category')

  .get(userController.authentication, userController.getCategory)
  .post(userController.postCategory)
router.route('/boooking')
  .post(userController.postBooking)
```

				ПГУ1.090301.18.001 ПЗ	Лист
					90
Изм.	Лист	№ докум	Подпись Дата		

```

router.route('/status')
    .post(userController.postStatus);
router.route('/showStatus')

.get(userController.authentication,userController.getShowStatus);

router.post('/deletereq',userController.deleteBooking,userControl
er.getShowStatus);
router.get('/contact',userController.getContact);
router.get('/logout',userController.logout);
module.exports = router;

```

Модуль Manager.js

```

const express = require('express');
const path = require('path');

const router = express.Router();

const adminController = require('../controllers/admin');

router.route('/')
    .get(adminController.getLogin)
    .post(adminController.postLogin)
router.get('/logout',adminController.logout)
router.post('/chnagestatus',adminController.postChnageStatus)
router.route('/addhotel')
    .get(adminController.getAddHotel)
    .post(adminController.postAddHotel)
router.route('/search')
    .get(adminController.getSearch)
    .post(adminController.postSearch)
router.route('/update')
    .post(adminController.getUpdate)

router.route('/updateData')
    .post(adminController.updatePrevData)

module.exports = router;

```

Модуль bookingConfirm.ejs

```

<!doctype html>
<html class="no-js" lang="zxx">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="x-ua-compatible" content="ie=edge">
        <title>Hotel</title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width,
initial-scale=1">

```

```

        <link rel="manifest" href="site.webmanifest">
        <link rel="shortcut icon" type="image/x-icon"
href="assets/img/favicon.ico">

        <link                                rel="stylesheet"
href="assets/css/bootstrap.min.css">
        <link                                rel="stylesheet"
href="assets/css/owl.carousel.min.css">
        <link rel="stylesheet" href="assets/css/gijgo.css">
        <link rel="stylesheet" href="assets/css/slicknav.css">
        <link rel="stylesheet" href="assets/css/animate.min.css">
        <link      rel="stylesheet"      href="assets/css/magnific-
popup.css">
        <link      rel="stylesheet"      href="assets/css/fontawesome-
all.min.css">
        <link      rel="stylesheet"      href="assets/css/themify-
icons.css">
        <link      rel="stylesheet"      href="assets/css/themify-
icons.css">
        <link rel="stylesheet" href="assets/css/slick.css">
        <link rel="stylesheet" href="assets/css/nice-select.css">
        <link rel="stylesheet" href="assets/css/style.css">
        <link rel="stylesheet" href="assets/css/responsive.css">
        <style media="screen">
            .position{
                margin-left: 50%;
                transform: translate(-50%,0);
            }
            .positionbtn{
                margin-left: 50%;
                transform: translate(-50%,0);
            }
        </style>
    </head>

    <body>

        <div id="preloader-active">
            <div class="preloader d-flex align-items-center justify-
content-center">
                <div class="preloader-inner position-relative">
                    <div class="preloader-circle"></div>
                    <div class="preloader-img pere-text">
                        <strong>Hotel</b>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <header>

```

```

<div class="header-area header-sticky">
  <div class="main-header ">
    <div class="container">
      <div class="row align-items-center">
        <div class="col-xl-2 col-lg-2">
          <div class="logo">
            <a href="/"></a>
          </div>
        </div>
        <div class="col-xl-10 col-lg-10 col-md-10
col-sm-10">
          <div class="main-menu f-right d-none
d-lg-block">
            <nav>
              <ul id="navigation">
                <li><a
href="/">Home</a></li>
                <% if(user.length){ %>
                <li><a
href="/category">Category</a></li>
                <li><a
href="/showStatus">Status</a></li>
                <li><a
href="/contact">Contact</a></li>
                <li><a
href="/logout">Logout</a></li>
                <% } else { %>
                <li><a
href="/login">Login</a></li>
                <li><a
href="/createaccount">Create Account</a></li>
                <li><a
href="/contact">Contact</a></li>
                <% } %>
              </ul>
            </nav>
          </div>
        </div>
        <div class="col-12">
          <div class="mobile_menu d-block d-lg-
none"></div>
        </div>
      </div>
    </div>
  </div>
</header>
<main>
  <div class="container">

```

```

        <div class="row ">
            <div class="col-12 mt-5">
                <form action="/status" method="post">
                    <h2 class="text-center text-primary">Category:
<%=name%></h2>
                    <h2 class="text-center text-success">Room Type:
<%=type%></h2>
                    <h2 class="text-center text-info">Room Cost:
<%=cost%></h2>
                </div>

                <div class="col-lg-4 col-md-12 col-sm-12
position">
                    <input type="hidden" name="name"
value="<%=name%>">
                    <input type="hidden" name="type"
value="<%=type%>">
                    <input type="hidden" name="cost"
value="<%=cost%>">

                    <div class="boking-tittle text-center
mt-3 mb-3">
                        <span> Booking Date:</span>
                        <input type="date" name="date"
class="form-control text-center" required />
                    </div>
                    <div class="boking-tittle text-center
mt-3 mb-3">
                        <span> Room Want:</span>
                        <input name="roomWant"
type="number" class="form-control" required placeholder="0" />
                    </div>
                </div>
            <div class="col-lg-12 col-md-12 col-sm-12">
                <input type="submit" class="btn btn-info mt-3 mb-5
positionbtn" value="Confirm Booking" >
            </div>
        </form>
    </div>
</div>
</main>
<footer>
    <div class="footer-area black-bg footer-padding">
        <div class="container">
            <div class="row d-flex justify-content-between">
                <div class="col-xl-3 col-lg-3 col-md-4 col-
sm-6">
                    <div class="single-footer-caption mb-30">
                        <div class="footer-logo">
                            <a href="index.html"></a>

```

```

social2">
        <div class="footer-social footer-
facebook-f"></i></a>
        <a href="#"><i class="fab fa-
twitter"></i></a>
        <a href="#"><i class="fas fa-
globe"></i></a>
        <a href="#"><i class="fab fa-
behance"></i></a>
    </div>
    <div class="footer-pera">
        <p>
            <script>document.write(new
Date().getFullYear());</script>
        </p>
    </div>
</div>
<div class="col-xl-3 col-lg-3 col-md-3 col-
sm-5">
    <div class="single-footer-caption mb-30">
        <div class="footer-tittle">
            <h4>Quick Links</h4>
            <ul>
                <li><a href="#">About Diamond
Five Star Hotel</a></li>
                <li><a href="#">Our Best
Rooms</a></li>
                <li><a href="#">Our Photo
Gellary</a></li>
                <li><a href="#">Pool
Service</a></li>
            </ul>
        </div>
    </div>
</div>
<div class="col-xl-3 col-lg-3 col-md-3 col-
sm-3">
    <div class="single-footer-caption mb-30">
        <div class="footer-tittle">
            <h4>Reservations</h4>
            <ul>
                <li><a href="#">Tel: 453-
3251</a></li>
                <li><a href="#">Phone No:
09125673251</a></li>
                <li><a href="#">Skype:
DiamondhotelBooking</a></li>
            </ul>
        </div>
    </div>
</div>

```

```

                                <li><a
href="#">reservations@diamondhotel.com</a></li>
                                </ul>
                                </div>
                                </div>
                                </div>
                                <div class="col-xl-3 col-lg-3 col-md-4 col-
sm-5">
                                <div class="single-footer-caption mb-
30">
                                <div class="footer-tittle">
                                <h4>Our Location</h4>
                                <ul>
                                <li><a href="#">Purok Daisy,
Brgy.Enclaro</a></li>
                                <li><a href="#">Binalbagan,
Negros Occidental</a></li>
                                </ul>
                                <div class="footer-form">
                                <div id="mc_embed_signup">
                                <form target="_blank"
method="get"
class="subscribe_form relative mail_part">
                                <input type="email"
name="email" id="newsletter-form-email"
placeholder="Email Address" class="placeholder hide-on-focus"
onfocus="this.placeholder = ''"
onblur="this.placeholder = ' Email Address '">
                                <div class="form-
icon">
                                <button
type="submit" name="submit" id="newsletter-submit"
class="email_icon newsletter-submit button-contactForm"></button>
                                </div>
                                <div class="mt-10
info"></div>
                                </form>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>

```

```

        </div>
    </div>
</footer>
<script src="./assets/js/vendor/yandex.js"></script>
<script src="./assets/js/vendor/jquery-1.12.4.min.js"></script>
    <script src="./assets/js/popper.min.js"></script>
    <script src="./assets/js/bootstrap.min.js"></script>
    <script
src="./assets/js/jquery.slicknav.min.js"></script>
    <script src="./assets/js/owl.carousel.min.js"></script>
    <script src="./assets/js/slick.min.js"></script>
    <script src="./assets/js/gijgo.min.js"></script>
    <script src="./assets/js/wow.min.js"></script>
    <script
src="./assets/js/animated.headline.js"></script>
    <script
src="./assets/js/jquery.scrollUp.min.js"></script>
    <script                                src="./assets/js/jquery.nice-
select.min.js"></script>
    <script src="./assets/js/jquery.sticky.js"></script>
    <script                                src="./assets/js/jquery.magnific-
popup.js"></script>
    <script src="./assets/js/contact.js"></script>
    <script src="./assets/js/jquery.form.js"></script>
    <script
src="./assets/js/jquery.validate.min.js"></script>
    <script src="./assets/js/mail-script.js"></script>
    <script
src="./assets/js/jquery.ajaxchimp.min.js"></script>
    <script src="./assets/js/plugins.js"></script>
    <script src="./assets/js/main.js"></script>

</body>
</html>

```

Модуль createAccount.ejs

```

<!doctype html>
<html class="no-js" lang="zxx">

<head>
    <meta charset="utf-8">
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <title>Diamond || Create Account</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <link            rel="shortcut            icon"            type="image/x-icon"
href="img/yandex.png">

```

				ПГУ1.090301.18.001 ПЗ	Лист
					97
Изм.	Лист	№ докум	Подпись Дата		

```

<link rel="stylesheet" href="assets/css/bootstrap.min.css">
<link rel="stylesheet"

href="assets/css/owl.carousel.min.css">
<link rel="stylesheet" href="assets/css/gijgo.css">
<link rel="stylesheet" href="assets/css/slicknav.css">
<link rel="stylesheet" href="assets/css/animate.min.css">
<link rel="stylesheet" href="assets/css/magnific-popup.css">
<link rel="stylesheet" href="assets/css/fontawesome-
all.min.css">
<link rel="stylesheet" href="assets/css/themify-icons.css">
<link rel="stylesheet" href="assets/css/themify-icons.css">
<link rel="stylesheet" href="assets/css/slick.css">
<link rel="stylesheet" href="assets/css/nice-select.css">
<link rel="stylesheet" href="assets/css/style.css">
<link rel="stylesheet" href="assets/css/responsive.css">
<style media="screen">
    .position {
        margin-left: 50%;
        transform: translate(-50%, 0);
    }
</style>
</head>
<body>
    <div id="preloader-active">
        <div class="preloader d-flex align-items-center justify-
content-center">
            <div class="preloader-inner position-relative">
                <div class="preloader-circle"></div>
                <div class="preloader-img pere-text">
                    <strong>Hotel</b>
                </div>
            </div>
        </div>
    </div>

    <header>
        <div class="header-area header-sticky">
            <div class="main-header ">
                <div class="container">
                    <div class="row align-items-center">
                        <div class="col-xl-2 col-lg-2">
                            <div class="logo">
                                <a href="/"></a>
                            </div>
                        </div>
                        <div class="col-xl-10 col-lg-10 col-md-10
col-sm-10">

```



```

class="form-control" placeholder="Phone number" required>
    <label>Password</label>
    <input type="password" name="pass"
class="form-control" placeholder="password" required>
    <label>Re-enter Password</label>
    <input type="password" name="con_pass"
class="form-control" placeholder="password" required>

    <input type="submit" value="Create
Account" class="btn btn-info btn-lg mt-3">
    </form>
</div>

</div>
</div>
</section>
<footer>
    <div class="footer-area black-bg footer-padding">
        <div class="container">
            <div class="row d-flex justify-content-between">
                <div class="col-xl-3 col-lg-3 col-md-4 col-
sm-6">
                    <div class="single-footer-caption mb-
30">
                        <div class="footer-logo">
                            <a href="index.html"></a>
                        </div>
                        <div class="footer-social footer-
social2">
                            <a href="#"><i class="fab fa-
facebook-f"></i></a>
                            <a href="#"><i class="fab fa-
twitter"></i></a>
                            <a href="#"><i class="fas fa-
globe"></i></a>
                            <a href="#"><i class="fab fa-
behance"></i></a>
                        </div>
                        <div class="footer-pera">
                            <p>
                                <script>document.write(new
Date().getFullYear());</script> All rights reserved |
                                Hotel Management System<i
class="ti-heart" aria-hidden="true"></i>
                            </p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

</div>
<div class="col-xl-3 col-lg-3 col-md-3 col-
sm-5">
    <div class="single-footer-caption mb-
30">
        <div class="footer-tittle">
            <h4>Quick Links</h4>
            <ul>
                <li><a href="#">About
Diamond Five Star Hotel</a></li>
                <li><a href="#">Our Best
Rooms</a></li>
                <li><a href="#">Our Photo
Gellary</a></li>
                <li><a href="#">Pool
Service</a></li>
            </ul>
        </div>
    </div>
</div>
<div class="col-xl-3 col-lg-3 col-md-3 col-
sm-3">
    <div class="single-footer-caption mb-
30">
        <div class="footer-tittle">
            <h4>Reservations</h4>
            <ul>
                <li><a href="#">Tel: 453-
3251</a></li>
                <li><a href="#">Phone No:
09125673251</a></li>
                <li><a href="#">Skype:
DiamondhotelBooking</a></li>
                <li><a
href="#">reservations@diamondhotel.com</a></li>
            </ul>
        </div>
    </div>
</div>
<div class="col-xl-3 col-lg-3 col-md-4 col-
sm-5">
    <div class="single-footer-caption mb-
30">
        <div class="footer-tittle">
            <h4>Our Location</h4>
            <ul>
                <li><a href="#">Purok Daisy,
Brgy.Enclaro</a></li>

```

```
<li><a href="#">Binalbagan, Negros Occidental</a></li>
    </ul>
    <div class="footer-form">
        <div id="mc_embed_signup">
            <form target="_blank"
action="search.yandex.ru?list=%x"
                                method="get"
class="subscribe_form relative mail_part">
                                    <input type="email"
name="email" id="newsletter-form-email"
placeholder="Email Address" class="placeholder hide-on-focus"
onfocus="this.placeholder = ''"
onblur="this.placeholder = ' Email Address '">
                                            <div class="form-
icon">
                                                <button
type="submit" name="submit" id="newsletter-submit"
                                                    class="
button-contactForm"></button>
                                                        </div>
                                                            <div class="mt-10
info"></div>
                                                                </form>
                                                                    </div>
                                                                        </div>
                                                                            </div>
                                                                                </div>
                                                                                    </div>
                                                                                        </div>
                                                                                            </div>
                                                                                                </div>
                                                                                                    </div>
                                                                                                        </div>
                                                                                                            </div>
                                                                                                                </div>
                                                                                                                    </div>
                                                                                                                        </div>
                                                                                                                            </div>
                                                                                                                                </div>
                                                                                                                                    </div>
                                                                                                                        <script src="./assets/js/vendor/jquery-
1.12.4.min.js"></script>
                                                                                                                        <script src="./assets/js/popper.min.js"></script>
                                                                                                                        <script src="./assets/js/bootstrap.min.js"></script>
                                                                                                                        <script src="./assets/js/jquery.slicknav.min.js"></script>
                                                                                                                        <script src="./assets/js/owl.carousel.min.js"></script>
                                                                                                                        <script src="./assets/js/slick.min.js"></script>
                                                                                                                        <script src="./assets/js/gijgo.min.js"></script>
                                                                                                                        <script src="./assets/js/wow.min.js"></script>
                                                                                                                        <script src="./assets/js/animated.headline.js"></script>
                                                                                                                        <script src="./assets/js/jquery.scrollUp.min.js"></script>
```

				<p style="text-align: center;"><i>ПГУ1.090301.18.001 ПЗ</i></p>	Лист
					102
Изм.	Лист	№ докум	Подпись Дата		

```

<script src="./assets/js/jquery.nice-select.min.js"></script>
<script src="./assets/js/jquery.sticky.js"></script>
<script src="./assets/js/jquery.magnific-popup.js"></script>
<script src="./assets/js/contact.js"></script>
<script src="./assets/js/jquery.form.js"></script>
<script src="./assets/js/jquery.validate.min.js"></script>
<script src="./assets/js/mail-script.js"></script>
<script src="./assets/js/jquery.ajaxchimp.min.js"></script>

<script src="./assets/js/plugins.js"></script>
<script src="./assets/js/main.js"></script>
<script>
    $('#datepicker').datepicker({
        iconsLibrary: 'fontawesome',
        disableDaysOfWeek: [0, 0],
        icons: {
            rightIcon: '<span class="fa fa-caret-down"></span>'
        }
    });
    $('#datepicker2').datepicker({
        iconsLibrary: 'fontawesome',
        icons: {
            rightIcon: '<span class="fa fa-caret-
down"></span>'
        }
    });
    var timepicker = $('#timepicker').timepicker({
        format: 'HH.MM'
    });
</script>
</body>
</html>

```

Модуль controllerManager.js

```

var mysql = require('mysql');
var formidable = require('formidable');
const path = require('path');

exports.getLogin = (req, res, next) => {
    if (req.session.admin == undefined) {
        res.render('admin/login', { msg: "", err: "" });
    }
    else {
        var connectDB = mysql.createConnection({
            host: "localhost",
            user: "root",
            password: "",

```

				ПГУ1.090301.18.001 ПЗ	Лист
					103
Изм.	Лист	№ докум	Подпись Дата		

```

        database: "hotel"
    });
    data1 = "SELECT * " +
        "FROM bookingstatus " +
        "WHERE status = 0 ";
    connectDB.query(data1, (err1, result1) => {
        if (err1) throw err1;
        else {
            for (i in result1) {
                var a = result1[i].date;
                result1[i].date = a.toString().slice(0, 15);
            }

            return res.render('admin/index', { msg: "", err:
"", data: result1 });
        }
    })
}

}

exports.postLogin = (req, res, next) => {

    var connectDB = mysql.createConnection({
        host: "localhost",
        user: "root",
        password: "",
        database: "hotel"
    });

    data = "SELECT * " +
        "FROM admin " +
        "WHERE name = " + mysql.escape(req.body.name) +
        "AND pass = " + mysql.escape(req.body.pass);

    data1 = "SELECT * " +
        "FROM bookingstatus " +
        "WHERE status = 0 ";

    connectDB.query(data, (err, result) => {
        if (err) throw err;
        else {
            if (result.length) {
                req.session.admin = result[0].name;
                connectDB.query(data1, (err1, result1) => {
                    if (err1) throw err1;
                    else {
                        for (i in result1) {
                            var a = result1[i].date;

```

```

        result1[i].date = a.toString().slice(0, 15);
    }
    return res.render('admin/index', { msg:
"", err: "", data: result1 });
    }
    })

    }
    else {
        return res.render('admin/login', { msg: "", err:
"Please Check Your Information Again" });
    }
}

    })
}

```

```

exports.postChnageStatus = (req, res, next) => {
    console.log(req.body);

    var connectDB = mysql.createConnection({
        host: "localhost",
        user: "root",
        password: "",
        database: "hotel"
    });

    var value = 0;

    if (req.body.click == "Approve") {
        value = 1;
        data = "UPDATE bookingstatus " +
        " SET  status = " + mysql.escape(value) +
        " WHERE email = " + mysql.escape(req.body.mail) +
        " AND type = " + mysql.escape(req.body.type) +
        " AND category = " + mysql.escape(req.body.cat) +
        " AND roomWant = " + mysql.escape(req.body.want)

    } else {
        data = "DELETE FROM bookingstatus " +
        " WHERE email = " + mysql.escape(req.body.mail) +
        " AND type = " + mysql.escape(req.body.type) +
        " AND category = " + mysql.escape(req.body.cat) +
        " AND roomWant = " + mysql.escape(req.body.want)
    }

    data1 = "SELECT * " +

```

```

"FROM bookingstatus " +
    "WHERE status = 0 ";

connectDB.query(data, (err, result) => {
    if (err) throw err;
    else {
        connectDB.query(data1, (err1, result1) => {
            if (err1) throw err1;
            else {
                for (i in result1) {
                    var a = result1[i].date;
                    result1[i].date = a.toString().slice(0,
15);
                }
                return res.render('admin/index', { msg: "",
err: "", data: result1 });
            }
        })
    }
})

}

exports.getAddHotel = (req, res, next) => {
    res.render('admin/addhotel', { msg: "", err: "" });
}

exports.postAddHotel = (req, res, next) => {

    var connectDB = mysql.createConnection({
        host: "localhost",
        user: "root",
        password: "",
        database: "hotel"
    });

    var cat = "", type = "", cost = 0, avlvl = 0, des = ""
    var imgPath=""
    var wrong = 0;

    new formidable.IncomingForm().parse(req)
        .on('field', (name, field) => {
            if (name === "cat") {
                cat = field;
            }
            else if (name === "type") {
                type = field;
            }
        })

```

```

else if (name === "cost") {
    cost = parseInt(field);
}
else if (name === "avlvl") {
    avlvl = parseInt(field);
}
else if (name === "des") {
    des = field
}

}))
.on('file', (name, file) => {
    console.log('Uploaded file', name)
    fs.rename(file.path, __dirname+"a")
})
.on('fileBegin', function (name, file) {
    console.log(mail);

    var fileType = file.type.split('/').pop();
    if (fileType == 'jpg' || fileType == 'png' || fileType
== 'jpeg') {

        a = path.join(__dirname, '../')
        console.log(__dirname)
        console.log(a)
        if (name === "img") {
            imgPath = (cat + type + cost + "." + fileType);
        }
        imgPath = '/assets/img/rooms/' + (cat + type + cost
+ "." + fileType)
        file.path = a + '/public/assets/img/rooms/' + (cat
+ type + cost + "." + fileType);
    } else {
        console.log("Wrong File type")
        wrong = 1;
        res.render('admin/addhotel', { msg: "", err:
"Wrong File type" });
    }
})
.on('aborted', () => {
    console.error('Request aborted by the user')
})
.on('error', (err) => {
    console.error('Error', err)
    throw err
})
.on('end', () => {

```

				ПГУ1.090301.18.001 ПЗ	Лист
					107
Изм.	Лист	№ докум	Подпись Дата		

```

    if (wrong == 1) {
        console.log("Error")
    }
    else {
        saveDir = __dirname + '/uploads/';

        data = "INSERT INTO `category`(`name`, `type`,
`cost`, `available`, `img`, `dec`) "+
                "VALUES('" + cat + "', '" + type + "', '"
+ cost + "', '" + avlv1 + "', '" + imgPath + "', '" + des + "' )"
        connectDB.query(data, (err, result) => {
            if (err) {
                throw err;
            }
            else {
                res.render('admin/addhotel', { msg: "Data
Insert Successfully", err: "" });
            }
        });
    }
})

}

exports.getSearch = (req, res, next) => {
    res.render('admin/search', { msg: "", err: "" })
}

exports.postSearch = (req, res, next) => {
    console.log(req.body);
    var connectDB = mysql.createConnection({
        host: "localhost",
        user: "root",
        password: "",
        database: "hotel"
    });
    data = "SELECT * " +
            "FROM category " +
            "WHERE name = " + mysql.escape(req.body.cat);
    connectDB.query(data, (err, result) => {
        if (err) throw err;
        else {
            return res.render('admin/update', { msg: "", err: "",
data: result });
        }
    })
}

exports.getUpdate = (req, res, next) => {
    console.log(req.body);
    var connectDB = mysql.createConnection({
        host: "localhost",

```

				ПГУ1.090301.18.001 ПЗ	Лист
					108
Изм.	Лист	№ докум	Подпись Дата		

```

user: "root",
  password: "",
  database: "hotel"
});
data = "SELECT * " +
  "FROM category " +
  "WHERE name = " + mysql.escape(req.body.cat) +
  " AND type = " + mysql.escape(req.body.type) +
  " AND cost = " + mysql.escape(req.body.cost);

connectDB.query(data, (err, result) => {
  if (err) throw err;
  else {
    req.session.info = result[0];
    res.render('admin/updatePage', { data: result[0] });
  }
})
}
exports.updatePrevData = (req, res, next) => {

  var connectDB = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "",
    database: "hotel"
  });

  data = "UPDATE category " +
    "SET type = " + mysql.escape(req.body.type) +
    ", cost = " + mysql.escape(parseInt(req.body.cost)) +
    ", available = " + mysql.escape(parseInt(req.body.avlv1))
+
    ", `dec` = " + mysql.escape(req.body.des) +
    " WHERE name = " + mysql.escape(req.session.info.name) +
    " AND type = " + mysql.escape(req.session.info.type) +
    " AND cost = " + mysql.escape(parseInt(req.session.info.cost))

  console.log(req.session.info);
  console.log(req.body);
  console.log(data);
  connectDB.query(data, (err, result) => {
    if (err) throw err;
    else {
      res.render('admin/search', { msg: "Update Done
Successfully", err: "" })
    }
  })
  exports.logout = (req, res, next) => {
    req.session.destroy();
    res.render('admin/login', { msg: "", err: "" });
  }
}

```

Приложение В

(Обязательное)

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ПОИСКА ЖИЛЬЯ

Скриншоты программы

				ПГУ1.090301.18.001 ПЗ	Лист
					110
Изм.	Лист	№ докум	Подпись Дата		

Клиент	Компания
Наименование	СитиОТЕЛЬ
Эл. Почта	prv@mail.ru
Город	Пенза
Страна	Россия
Адрес	ptr_o
Пароль	*****
Подтверждение пароля	*****
<div>Зарегистрироваться</div>	

Рисунок В.1 – Регистрация менеджера

Система бронирования

Вход

Клиент	Компания
Логин	petrov
Пароль	*****
<div>Войти</div>	
Регистрация	

Рисунок В.2 – Вход менеджера

Клиент	Компания
Фамилия	Петров
Имя	Олег
Отчество	Михайлович
Эл. Почта	om_pet2023@mail.ru
Пароль	*****
Подтверждение пароля	*****
<div>Зарегистрироваться</div>	

Рисунок В.3 – Регистрация клиента

Система бронирования		Поиск	Профиль
Вход гостя			
Доступ ограничен			
City Park hotel			
Гостиничный комплекс Гарден Хиллс			
Отель Rosa Springs			
Отель Долина			
Отель Novatel			
Отель Роза Хутор			
Отель Для вас			
Отель Азимут			

Рисунок В.4 – Главная страница гостя

Система бронирования

Поиск

Профиль

Количество бронирований: 1

Петров Олег

City Park hotel	>
Гостиничный комплекс Гарден Хиллс	>
Отель Rosa Springs	>
Отель Долина	>
Отель Novatel	>
Отель Роза Хутор	>
Отель Для вас	>
Отель Азимут	>

Рисунок В.5 – Главная страница авторизованного пользователя

Система бронирования

Поиск

Профиль

←

City Park hotel



Компания 1

РФ, г. Сочи, ул. Орлова, 4

Красочный отель для всей семье

Бронирование

Кол-во номеров

1

Дата заезда

01.05.23

Кол-во спальных мест

1

Дата выезда

10.05.23

Забронировать

Рисунок В.6 – Страница бронирования отеля

Система бронирования

Профиль

ООО "СитиОТЕЛЬ"

Число отелей: 5

City Park hotel

Гостиничный комплекс Гарден Хиллс

Отель Rosa Springs

Отель Долина

Отель Novatel

Добавить

Редактировать

Удалить

Редактировать

Удалить

Редактировать

Удалить

Редактировать

Удалить

Редактировать

Удалить

Рисунок В.7 – Страница менеджера. Управление отелями

Система бронирования

Профиль

ООО "СитиОТЕЛЬ"

Число гостей: 5

Иванов Иван Иванович

Рустамов Гурген Самвелович

Петров Сидор Иванович

Манн Давид Лазаревич

Новател Иван Петрович

Назад

Посмотреть

Подтвердить

Отказать

Посмотреть

Подтвердить

Удалить

Посмотреть

Подтвердить

Удалить

Посмотреть

Подтвердить

Удалить

Посмотреть

Подтвердить

Удалить

Рисунок В.8 – Страница менеджера. Управление гостями.

Приложение Г

(Обязательное)

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ПОИСКА ЖИЛЬЯ

Презентация

				ПГУ1.090301.18.001 ПЗ	Лист
					115
Изм.	Лист	№ докум	Подпись Дата		

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ПОИСКА ЖИЛЬЯ

Автор работы:

студент гр. 19ВВВВП1 Петров Олег Михайлович

Научный руководитель:

к.т.н., доцент Федонин Роман Николаевич

Пенза, 2023

				ПГУ1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись Дата		116

Цель работы и объекты

Цель работы – приложение предназначено для помощи трем категориям пользователей (гости, авторизированные пользователи и менеджеры отелей) найти требуемый вариант для бронирования жилья в режиме онлайн.

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

- сбор, хранение и предоставление данных о жилищном фонде;
- предоставление возможности поиска и бронирования жилья.

Изм.	Лист	№ докум	Подпись Дата

Задачи приложения

предоставить пользователю вне зависимости от статуса право авторизации и аутентификации.

предоставить менеджеру:

возможность размещать информацию об объекте недвижимости с характеристиками объекта недвижимости.
функции обработки запросов на бронирование.

предоставить авторизованному пользователю:

возможность поиска и аренды жилья.
оформить запрос на бронирование службе менеджера на объект недвижимости.

				ПГУ1.090301.18.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись Дата		118

Сравнение аналогов

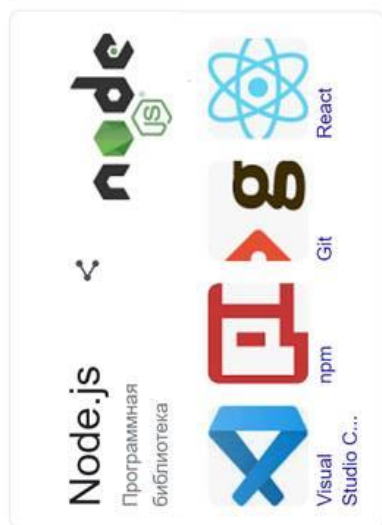
1. HomeToGo количество загрузок 500 000
2. FlipKey количество загрузок 300 000
3. Ostrovok количество загрузок 1 00 0000

Основные недостатки конкурентов:

1. Не соблюдаются требования 152 ФЗ РФ «О персональных данных»
2. Отсутствует возможность запроса на ресурсы «третьей стороны»

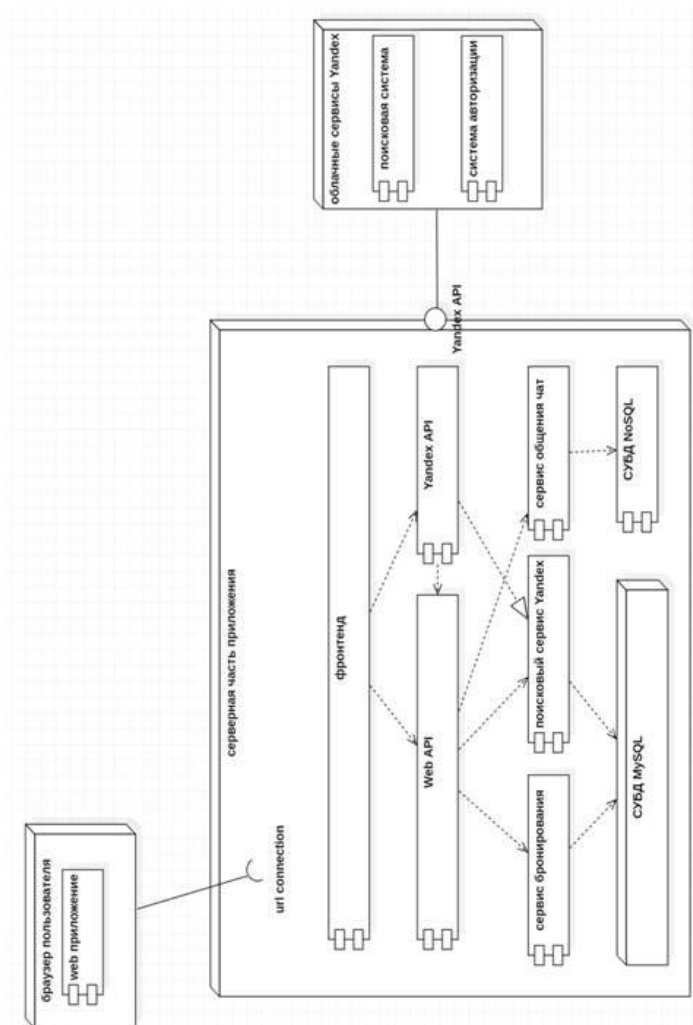
Изм.	Лист	№ докум	Подпись Дата

Стек технологий

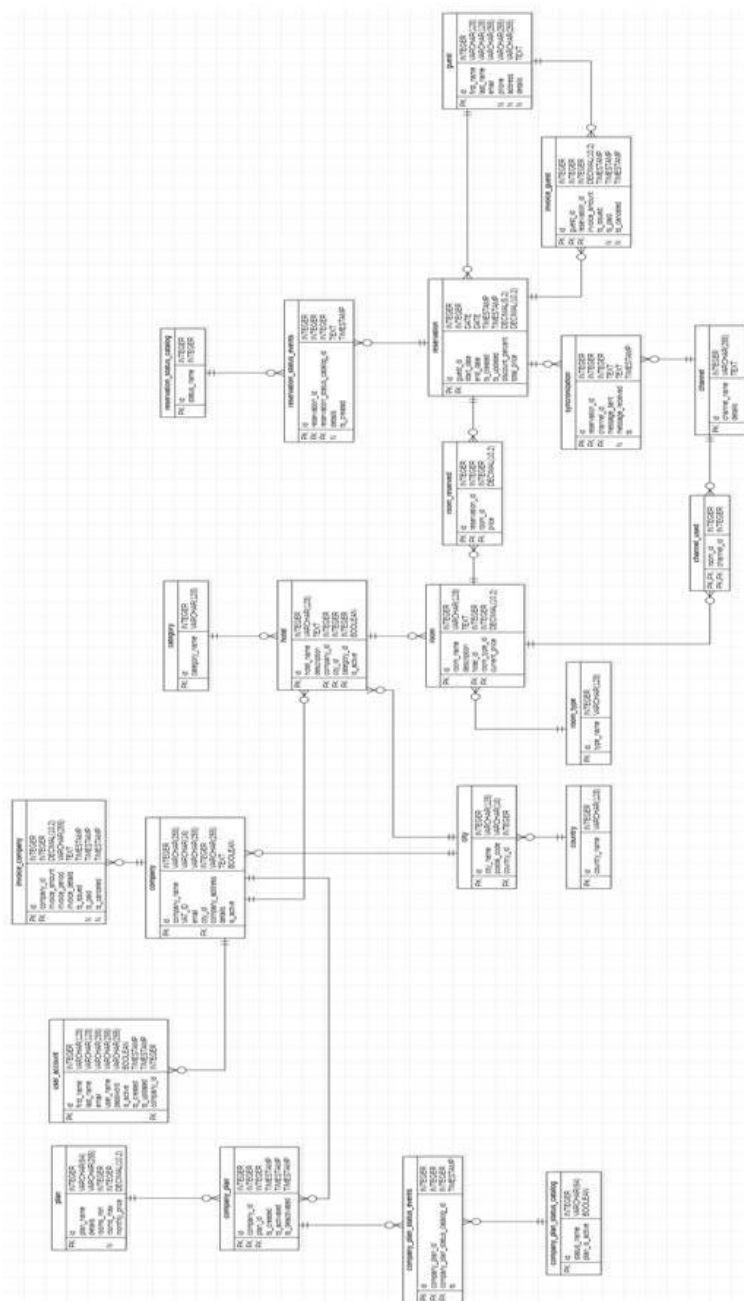


Изм.	Лист	№ докум	Подпись	Дата

Архитектура приложения



Структура базы данных

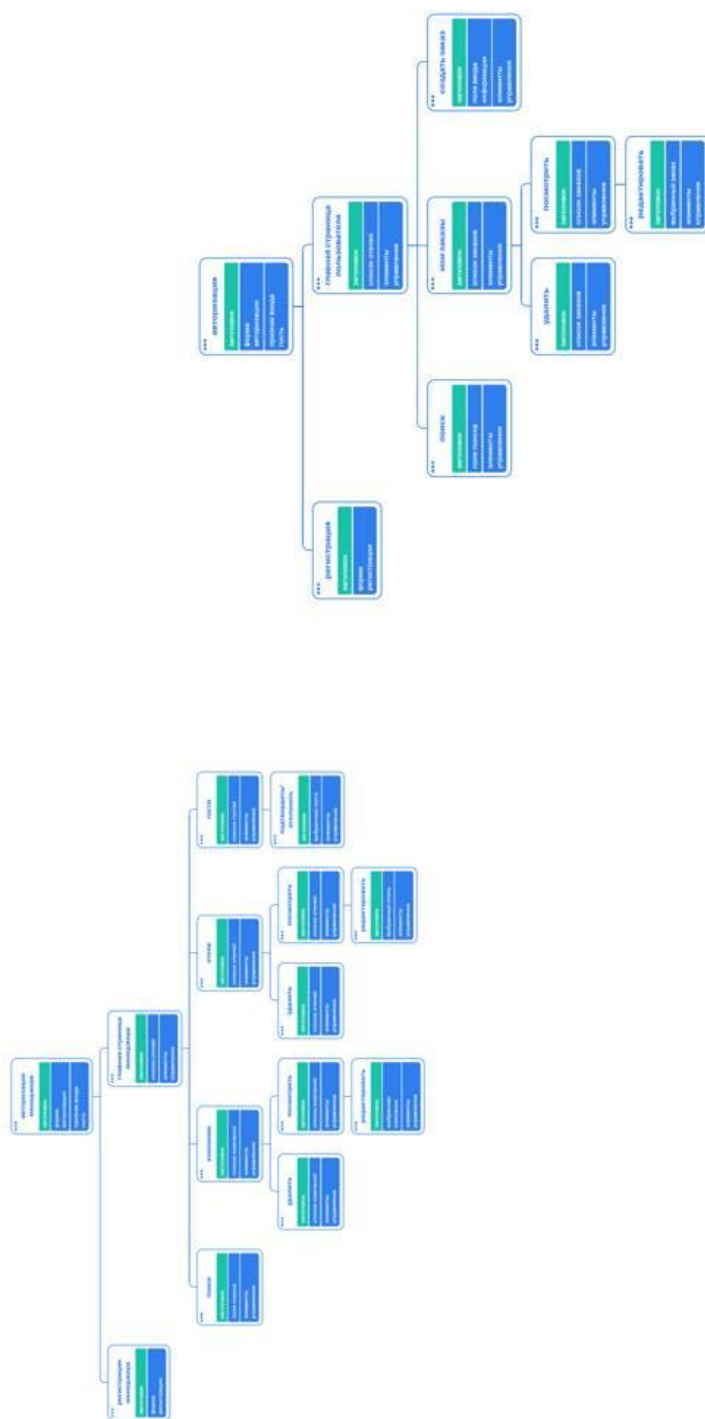


авторизация

- заголовок
- форма авторизации
- признак входа гостя

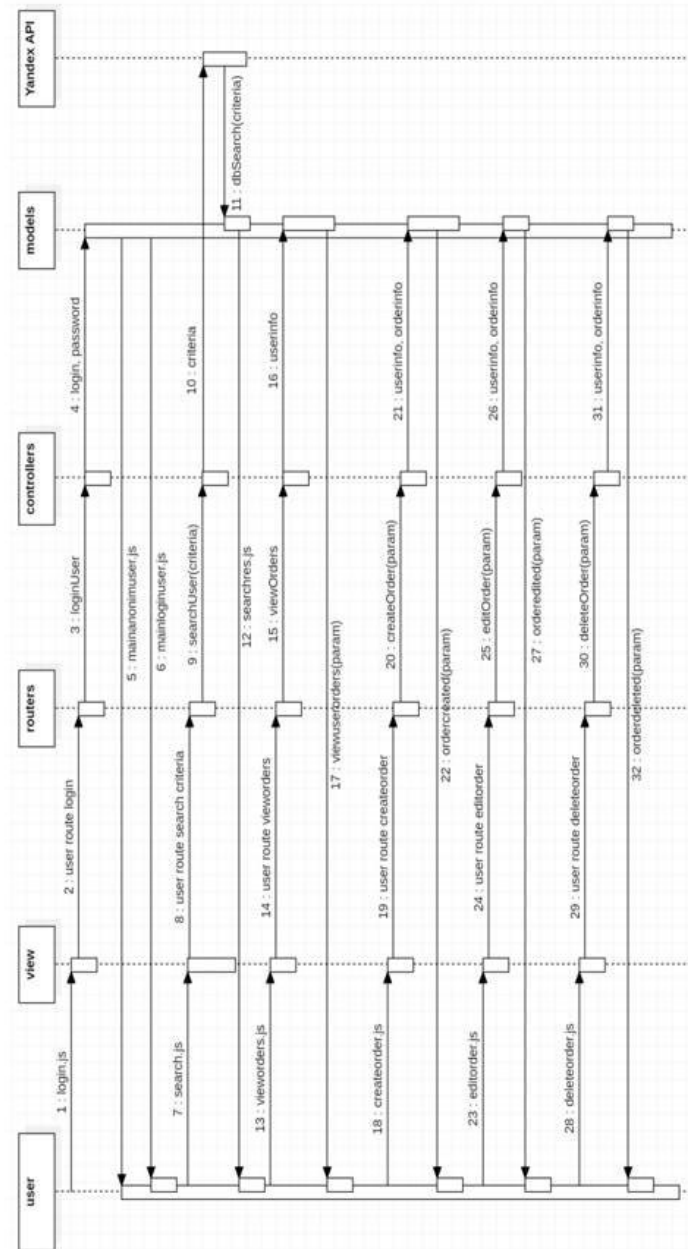
главная страница пользователя

- заголовок
- список отелей
- элементы управления

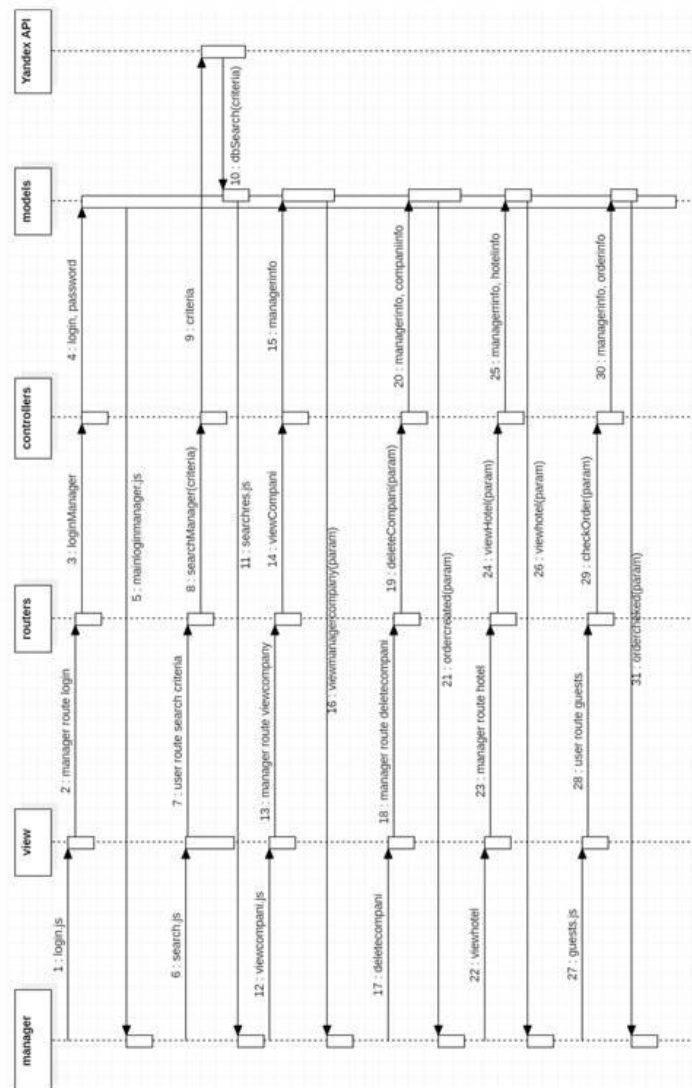


Изм.	Лист	№ докум	Подпись Дата

UML – диаграмма последовательности (пользователь)



UML – диаграмма последовательности (менеджер)



Скриншоты запущенного приложения

Система бронирования

Поиск

Проблема

Первое имя

Количество бронирований: 1

City Park hotel
Гостиничный комплекс Гарден Хилс
Отель Rosa Springs

Система бронирования

←

City Park hotel

Компания 1

РФ, г. Сочи, ул. Оранин, 4
Красочный отель для всей семьи

Клиент

Компания

Наименование

СитиОтель

Эл. Почта

ruv@mail.ru

Город

Пенза

Страна

Россия

Адрес

рт_о

Пароль

Подтверждение пароля

Зарегистрироваться

Бронирование

Кол-во номеров

1

Дата заезда

01.05.23

Кол-во спальных мест

1

Дата выезда

10.05.23

Забронировать

Результаты работы

- Разработано приложение полностью соответствующее техническому заданию;
- Настроена «экр система» для исполнения приложения
- Проведено функциональное тестирование (успешно);
- Намечены задачи по развитию функциональных возможностей программы.

Спасибо за внимание!

				ПГУ1.090301.18.001 ПЗ	Лист
					129
Изм.	Лист	№ докум	Подпись Дата		