

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

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

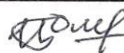
Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Информационная система сервисного
обслуживания гостиничного комплекса**

Студент

 14.06.21
(подпись, дата)

Помякшев Илья Николаевич
(ФИО полностью)

Руководитель

 14.06.21
(подпись, дата)

Балашова И. Ю.
(фамилия, инициалы)

Нормоконтролёр

 22.06.21
(подпись, дата)

Такташкин Д.В.
(фамилия, инициалы)

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

Заведующий кафедрой

(подпись)

Макарычев П.П.
(фамилия, инициалы)

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

Секретарь ГЭК

(подпись)

Попова Н.А.
(фамилия, инициалы)

Пенза, 2021

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

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
П.П. Макарычев
« 21 » января 2021 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Помякшев Илья Николаевич гр. 17ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Информационная система сервисного обслуживания гостиничного комплекса

Тема утверждена приказом ПГУ № 0027 а/о от " 21 " января 2021

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

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

1) *Назначение: информационная система предназначена для управления*

заявкам и их оформлением, каталогом услуг и пользователями в виде веб-приложения

2) *Среда выполнения: любой интернет-браузер актуальной версии*

Средства разработки: языки программирования C#, SQL, JS

Инструментальные средства: Visual Studio 2019, Microsoft SSMS 18

Технологии разработки: ООП

3) *Основные функции информационной системы сервисного обслуживания*

гостиничного комплекса:

а) просмотр сервисного каталога;

б) подбор услуги по критериям;

в) управление сервисным каталогом;

г) оформления заявки и ее отслеживание;

д) изменение статуса заявки;

е) управление учетными записями пользователей.

4. Перечень подлежащих разработке вопросов

Анализ предметной области

Анализ требований

Проектирование

Реализация

Контроль качества ПО:

– Тестирование

– Анализ качества кода

План разработки и оценка бюджета

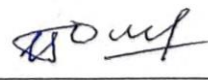
5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований	10	14.04.2021 – 24.04.2021	
2	Проектирование	16	25.04.2021 – 09.05.2021	
3	Реализация	17	10.05.2021 – 25.05.2021	
4	Контроль качества	9	26.05.2021 – 03.06.2021	
5	План разработки бюджета	2	04.06.2021 – 06.06.2021	
6	Оформление пояснительной записки	7	07.06.2021 – 14.06.2021	

Дата выдачи задания " 28 " декабря 2021

Руководитель бакалаврской работы  / И.О. Власов

(подпись, дата)

Задание к исполнению принял студент 
(подпись, дата)

Работу к защите допустить

Декан факультета

(подпись, дата)

Реферат

Пояснительная записка содержит 103 листа 42 рисунка, 15 использованных источников.

ИНФОРМАЦИОННАЯ СИСТЕМА, СЕРВИСНОЕ ОБСЛУЖИВАНИЕ,
UML, БАЗА ДАННЫХ, КЛИЕНТ, СЕРВЕР, SQL, ASP.NET

Объект разработки – информационная система сервисного обслуживания гостиничного комплекса.

Цель работы – разработать информационную систему, позволяющую осуществлять подбор услуг в веб-приложении по различным критериям с возможностью оформления заявки на обслуживание.

Технология разработки – для разработки серверной части использовалась СУБД Microsoft SSMS 18 и язык SQL, клиентская часть была реализована на языке C# на платформе ASP.NET Core 5.0 в среде разработки Visual Studio 2019 с использованием фреймворка MVC.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная часть для информационной системы сервисного обслуживания гостиничного комплекса. Все требования к разработке приложения соблюдены.

					ПГУ 09.03.04–08BP171.16 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Информационная система сервисного обслуживания гостиничного комплекса	Лит.	Лист	Листов	
Разраб.		Помякшев И.Н.		14.06.21					
Провер.		Балашова И. Ю.		14.06.21			3	103	
Н. Контр.		Такташкин Д.В.		22.06.21	Пояснительная записка	Группа 17ВП1			
Утверд.		Макарычев П.П.							

Содержание

Введение.....	6
1 Анализ предметной области и требований.....	8
1.1 Анализ предметной области информационной системы сервисного обслуживания гостиничного комплекса.....	8
1.2 Анализ программных аналогов информационной системы сервисного обслуживания гостиничного комплекса.....	9
1.3 Постановка задачи информационной системы сервисного обслуживания гостиничного комплекса.....	18
1.4 Анализ бизнес-процессов информационной системы сервисного обслуживания гостиничного комплекса.....	19
1.5 Анализ требований информационной системы сервисного обслуживания гостиничного комплекса.....	24
1.5.1 Анализ функциональных требований.....	25
1.5.2 Анализ нефункциональных требований.....	34
1.6 Выбор инструментальных средств.....	35
2 Проектирование информационной системы сервисного обслуживания гостиничного комплекса.....	37
2.1 Базовая архитектура системы.....	37
2.2 Логическая модель базы данных.....	39
2.3 Проектирование пользовательского интерфейса.....	40
3 Реализация и тестирование.....	52
3.1 Физическая модель базы данных.....	52
3.2 Реализация программных средств.....	55
3.3 Тестирование приложения.....	60
3.3.1 Выбор режима тестирования.....	60
3.3.2 Функциональное тестирование.....	60
3.3.3 Метрики кода.....	61
3.3.4 Модульное тестирование.....	64
4 Планирование и оценка бюджета проекта.....	67

4.1 Календарное планирование и оценка бюджета проекта	68
Заключение	70
Список использованных источников	71
Приложение А. Глоссарий	73
Приложение Б. Листинг скрипта создания таблиц базы данных на языке SQL.....	75
Приложение В. Листинг представлений и триггеров на языке SQL.....	81
Приложение Г. Листинг клиентского приложения на языке C#.....	85
Приложение Д. Результаты тестирования	88
Приложение Е. Спецификация сущностей.....	93

Введение

Разнообразие интересов потребителей гостиничных услуг определяет современную сегментацию мирового гостиничного рынка и разнообразие гостиниц. Более 50% путешествий составляют деловые поездки, примерно 40% отводится туристическим поездкам и 10% на другие цели. Эта структура потребительского спроса определяет и спрос на отели соответствующего качества и направленности [1].

Кроме того, в настоящее время информационные технологии значительно изменили жизнь современного человека. Модернизация коснулась в первую очередь вопросов развития бизнеса. Те компании, которые умело взяли на заметку новую тенденцию информационных технологий, получили для себя колоссальную выгоду. Не исключением стала и гостиничная сфера.

Так по результатам различных исследований во время карантинных мер 2020 года показатели продаж бизнес-отелей стремительно падали к нулю и в конце года так и не смогли выйти на уровень 2019 года. Наибольшие убытки были зафиксированы среди отелей Москвы. По статистике, в столице заполняемость в 2020 составила лишь 58% против 77% в 2019 году. Около трети небольших гостиниц и хостелов окончательно закрылись. По самым оптимистичным прогнозам, гостиничная отрасль будет восстанавливаться в течение полутора лет, но некоторые эксперты считают, что достичь предыдущих показателей возможно будет только через 3-5 лет [2].

В подобных условиях бесконтактные сервисы, онлайн регистрация, бронирование, виртуальные туры – за несколько месяцев всё это прочно вошло в список услуг, которые предлагают гостиницы и обеспечивают это благодаря различным информационным системам.

Данная проблема весьма актуальна, ведь качественное обслуживание в современных реалиях просто не может обойтись без подобной системы, которая обеспечит потребителю быстрый доступ к сервису, позволит, не

обращаясь за помощью к специалисту получить ответ на возникший вопрос, оформить заявку и получить ее своевременное обслуживание. Кроме того, такое приложение позволит расширить базу клиентов и станет отличным способом прорекламировать бизнес.

Таким образом, целью выпускной квалификационной работы является разработка информационной системы сервисного обслуживания гостиничного комплекса в виде веб-приложения.

Для достижения указанной цели были поставлены следующие задачи:

- проведение анализа предметной области и существующих информационных систем аналогичного назначения;
- постановка задачи на разработку информационной системы сервисного обслуживания гостиничного комплекса;
- определение требований к разрабатываемой системе;
- проектирование будущей информационной системы для предметной области;
- реализация и тестирование информационной системы сервисного обслуживания гостиничного комплекса.

1 Анализ предметной области и требований

1.1 Анализ предметной области информационной системы сервисного обслуживания гостиничного комплекса

Доходы от международного туризма составляют около триллиона долларов. На долю туризма приходится 8% мирового экспорта, более 31% международного рынка услуг и более 100 млн. рабочих мест по всему миру. После нефтяной промышленности он стал второй отраслью хозяйства [3].

Туризм принадлежит к сфере услуг и является одной из крупнейших и динамичных отраслей экономики. Высокие темпы его развития, большие объемы валютных поступлений активно влияют на различные сектора экономики, что способствует формированию собственной туристской индустрии [4].

Выделяют несколько основных направлений, по которым развивается гостиничный бизнес. Основные из них тесно связаны с расширением услуг, которые предлагают гостиницы.

На основании ст. 1 Федерального закона от 24.11.1996 N 132-ФЗ «Об основах туристской деятельности в Российской Федерации» гостиничные услуги – это комплекс услуг по предоставлению физическим лицам средства размещения и иных услуг, предусмотренных правилами предоставления гостиничных услуг в Российской Федерации, утвержденными Правительством Российской Федерации, которые предоставляются индивидуальными предпринимателями и юридическими лицами [5].

Гостям, проживающим в коллективных средствах размещения, должен быть предоставлен минимальный ассортимент услуг [5]:

- круглосуточный прием;
- услуги общественного питания по ГОСТ Р 50764 или условиях для самостоятельного приготовления пищи;

- ежедневная уборка жилой комнаты (включая заправку постелей) и санузла (кроме общежитий, лагерей труда и отдыха, туристских приютов, стоянок, кемпинга);
- смена постельного белья – не менее одного раза в неделю, смена полотенец – не менее одного раза в три дня;
- хранение ценностей, багажа;
- медицинская помощь: вызов скорой помощи, пользование аптечкой;
- туристская информация.

Проведенный анализ позволил выявить основные задачи разрабатываемой информационной системы, ими являются обеспечение клиентов возможностью оформления как минимальных услуг, так и дополнительных, заключение контрактов с гостями, контроль учетных записей пользователей, клиентских заявок на обслуживание, обновление каталога услуг и т.д.

1.2 Анализ программных аналогов информационной системы сервисного обслуживания гостиничного комплекса

В настоящее время существует огромное количество гостиничных предприятий, в том числе и альтернативных средств размещения. У подавляющего большинства из них основной функционал при работе с клиентом сконцентрирован на бронировании, порой это одностраничные сайты-визитки с контактной информацией. В подобных случаях потребителю не остается другого выбора, кроме как личного посещения или же звонка специалисту, который может и не ответить на возникшие вопросы. По результатам опросов подавляющее большинство клиентов сферы услуг предпочитает анонимность, а также возможность обойтись без помощи третьих лиц в процессе оформления заявки. Именно такие задачи и помогает решить правильно спроектированная информационная система.

С помощью подобных сервисов, предоставляющих возможность предварительного отображения оказываемых услуг, их самостоятельное оформление и контроль, и обеспечивается успешная работа гостиничных предприятий.

Изучение проблемы недостаточной информационной демонстрации, анализ предоставляемых дополнительных услуг, а также работа сервисного обслуживания осуществлялось на примере трех различных сайтов популярных гостиничных предприятий по всей России:

- гостиница «Heliopark Cruise» (Пенза) – местный доступный отель, получивший высокие оценки от гостей;
- бутик-отель «Рыбзавод» (Астраханская область) – входит в пятерку лучших загородных отелей России по версии «National Geographic Traveler Russia 2019»;
- гостиничный комплекс «Измайлово Альфа» (Москва) – получил награду «Travellers Choice 2017» от TripAdvisor.

Сайт гостиницы «Heliopark Cruise» встречает пользователя контактной информацией и выделенным блоком спецпредложений, ниже представлена основная информация об отеле и контакты для связи, также присутствует возможность забронировать номер с предоплатой в размере 100% (рисунок 1). Личный кабинет не будет доступен новым пользователям, так же, как и регистрация на сайте, учетная запись будет выдаваться индивидуально по усмотрению администрации.

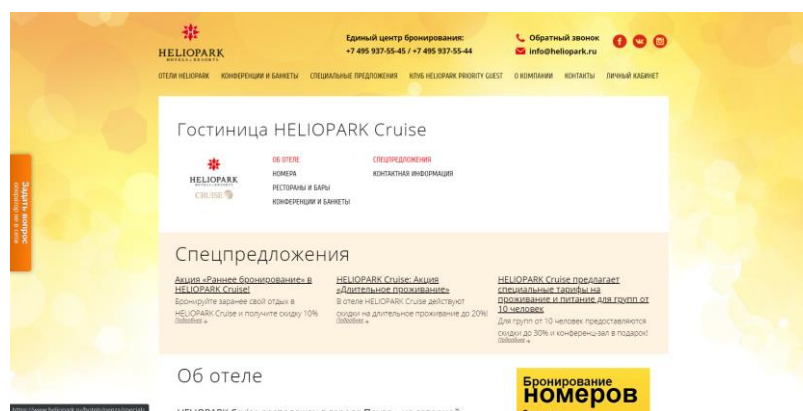


Рисунок 1 – Начальная страница сайта гостиницы «Heliopark Cruise»

Потенциальный гость может ознакомиться с перечнем услуг и обслуживания только по средствам текстовой информации, которая содержится в разделе с краткой информацией об отеле (рисунок 2).



Отель соответствует категории 2* и располагает номерами различных категорий – от стандарта до двухкомнатного люкса.

Мы можем гарантировать Вам:

- высокий уровень сервиса;
- отдых в тихом комфортабельном номере;
- бесплатный высокоскоростной доступ в Интернет;
- легкий перекус или плотный завтрак, обед или ужин в кафе;
- бесплатную парковку для гостей отеля.

Номера оборудованы комфортабельной мебелью, кабельным телевидением, системой индивидуального климат-контроля, холодильником и сейфом.

Особым преимуществом является расположение отеля на въезде в город, при этом дорога от отеля до центра города займет всего 15 минут!

Гостиница HELIOPARK Cruise рассчитана на бизнес - туристов, а также на гостей, путешествующих через город транзитом, последние оценят удобное расположение отеля и большую парковку.

Бронирование в отеле:
 Телефон: +7 (8412) 64-34-33
 Факс: +7 (8412) 64-34-33
 E-mail: crui@heliopark.ru
 Адрес: 440015, Россия, г. Пенза, ул. Аустрина д. 144а

Единый центр бронирования:
 +7 (495) 937-55-45
 Заказ мероприятий: corporate@heliopark.ru
 Отдел частных продаж: booking@heliopark.ru
 Для туристических компаний: sales@heliopark.ru

Номерной фонд: 61

Время заезда: 14.00 Время выезда: 12.00

HELIOPARK Cruise
 Пенза
 5.0/5.0
 10 отзывов
 TripAdvisor

Рисунок 2 – Основная информация на сайте гостиницы «Heliopark Cruise»

Кроме этого в навигационной панели имеется ссылка на круиз-кафе, которое расположено на первом этаже отеля, с кратким меню и графиком его работы, а также скромной галереей.

В целом сайт понятен и структурирован, но не является эстетичным; дизайн сайта давно не обновлялся, как и возможно информация на нем, различные функциональные блоки выполнены в различных стилях, что говорит о несогласованности разработки и привлечении сторонних готовых решений. Получить минимальную информацию о сервисном обслуживании клиент может на главной странице, чтобы узнать больше, в том числе более полный перечень услуг, их стоимость и политику предоставления нужно будет связаться со специалистом по номеру, указанному в контактах.

Бутик-отель «Рыбзавод» представляет собой загородный дизайнерский отель премиум-класса на берегу Старой Волги. Попадая на официальный сайт предприятия клиент уже может ощутить, на сколько сотрудникам отеля важно качество предоставляемых услуг и их внимание к деталям – сайт

разработан с использованием современных технологий, анимирован, органично выполнен и представлен в едином стиле. Уже стартовая страница предоставляет функционал, который позволяет подробно ознакомиться с отелем, узнать о его инфраструктуре, истории, перечнем предоставляемых услуг с соответствующим визуальным оформлением, представленном в виде анимации, фото и видео (рисунок 3).

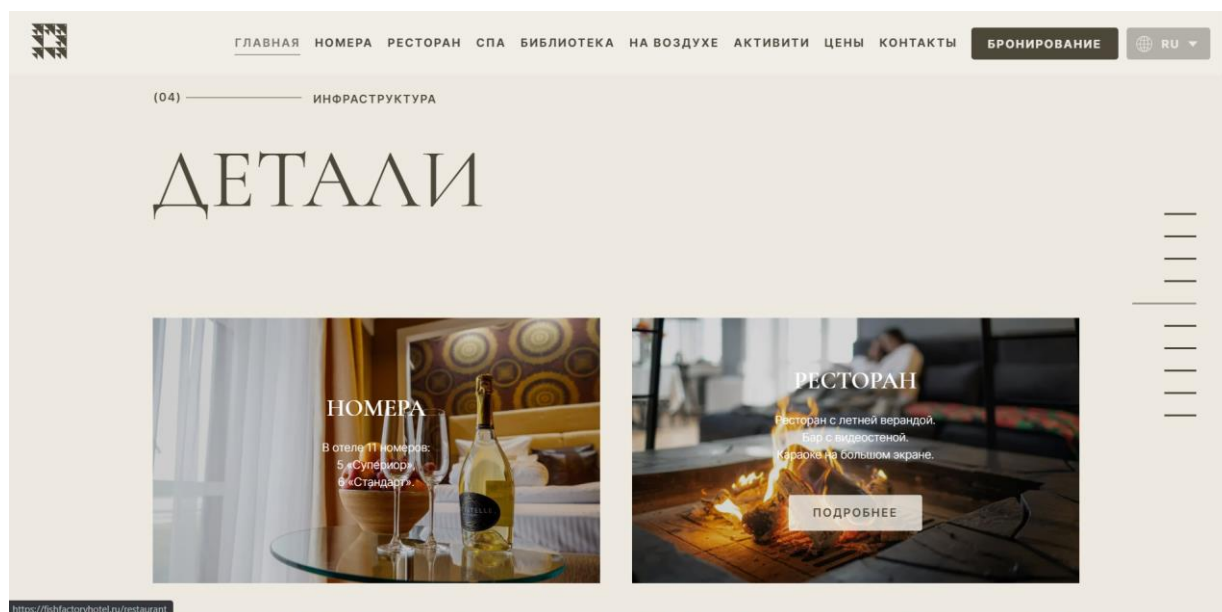


Рисунок 3 – Основная информация на сайте бутик-отеля «Рыбзавод»

Бутик-отель «Рыбзавод» обеспечивает потребителя их услуг современным дружественным интерфейсом, с помощью которого он имеет возможность узнать подробнее о всех перечисленных разделах инфраструктуры, получить краткое описание сервисов и их визуальное исполнение с помощью представленной галереи.

По средствам начальной страницы или навигационной панели пользователь информационной системы может узнать более подробно об интересующем его сервисе либо забронировать номер. Рассмотрим вариант «Ресторан – Подробнее», перейдя на новую страницу с более детальным описанием предоставляемых услуг, объединенных в данную категорию.

Страница выполнена с помощью ярких блоков, объединенных в виде слайдера-карусели, фотографии представлены с кратким описанием предоставляемой услуги (рисунок 4).

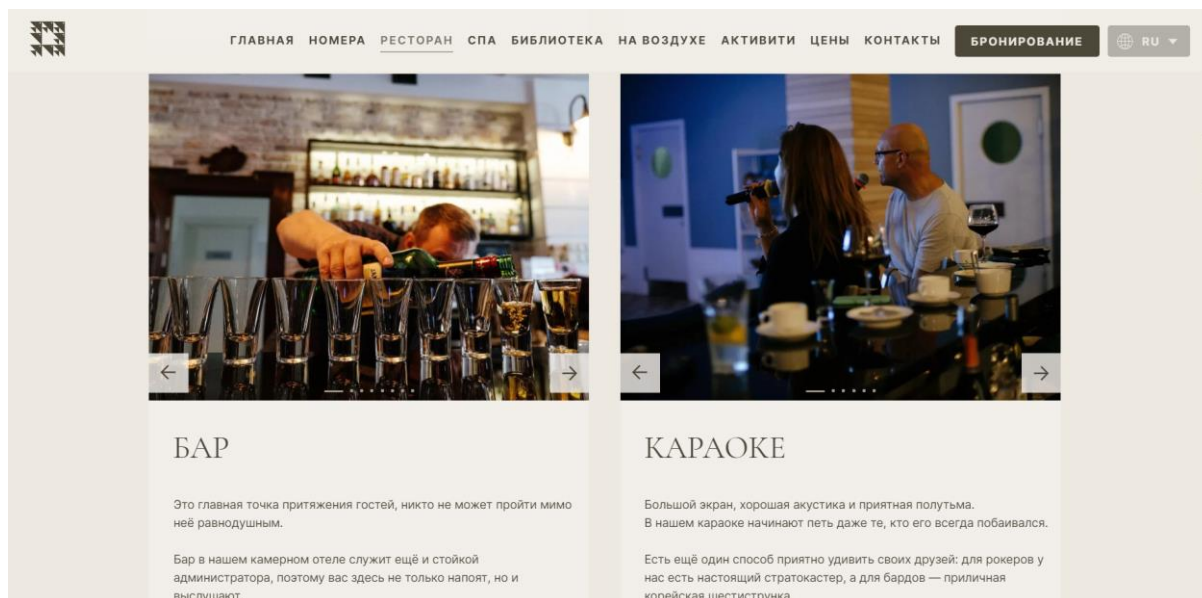


Рисунок 4 – Раздел «Ресторан» бутик-отеля «Рыбзавод»

Изучив сайт детальнее, пользователь сможет перейти в раздел «Цены», где будут перечислены средние стоимости за проживание, включенный в эту стоимость сервис обслуживания, а также краткий перечень услуг без указания стоимости. Указаны также условия размещения и контакты для взаимодействия, отсутствие личного кабинета говорит о том, что сайт является яркой визиткой, но не средством помощи гостям в получении сервисного обслуживания напрямую и дистанционно.

Гостиничный комплекс (ГК) «Измайлово Альфа» со стартовой станицы демонстрирует пользователю функциональную слайдер-карусель номеров, обеспечивающих также возможности их бронирования, 3D просмотра, получения краткой информации. При дальнейшем знакомстве с системой клиент может получить контактную информацию о комплексе, свежую новостную ленту о недавних изменениях. Сайт предоставляет дружелюбный интерфейс со стандартным органичным оформлением, но что самое главное в рамках проведения сравнительного анализа пользователь

может взаимодействовать с сервисом ГК с самого старта благодаря двум функциональным кнопкам по углам (рисунок 5).

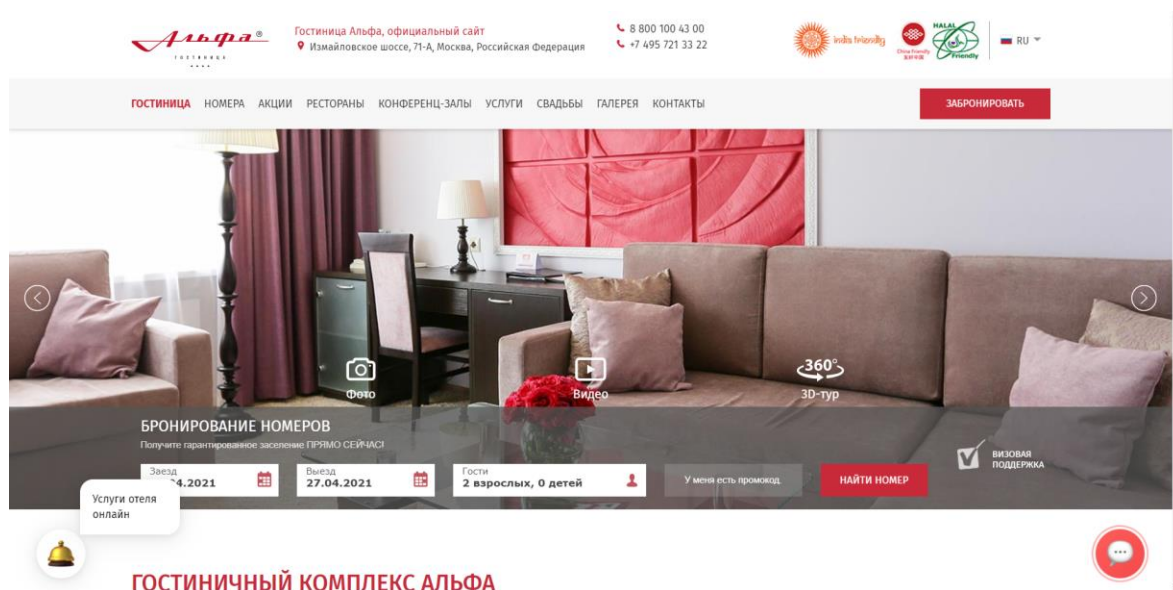


Рисунок 5 – Начальная страница ГК «Измайлово Альфа»

Правая отвечает за онлайн консультацию по интересующим вопросам с чат-ботом, который может быть заменен на реального сотрудника в случае затруднения по нахождению решения.

Левая предоставляет возможность ознакомиться с перечнем услуг его потребителю, а что самое главное – воспользоваться ими, а так как возможность авторизации на сайте отсутствует, пользователь может сделать это даже не являясь клиентом ГК (рисунок 6).

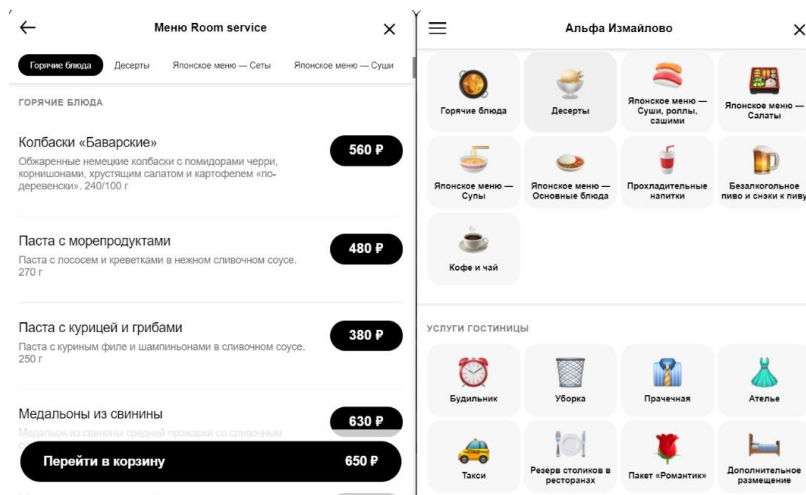


Рисунок 6 – Панель управления услугами ГК «Измайлово Альфа»

Таким образом, можно ознакомиться с полным перечнем услуг, их ценами и кратким описанием, недостатком является отсутствие фактических изображений услуг, например, заказываемых блюд. Для завершения оформления заявки клиент должен перейти в оформленную корзину товаров, указать способ оплаты (в том числе и наличными), ввести контактные данные, а также номер комнаты и дату заезда в нее – такой функционал гарантирует, что услугу получит соответствующий заказу и проживанию гость (рисунок 7).

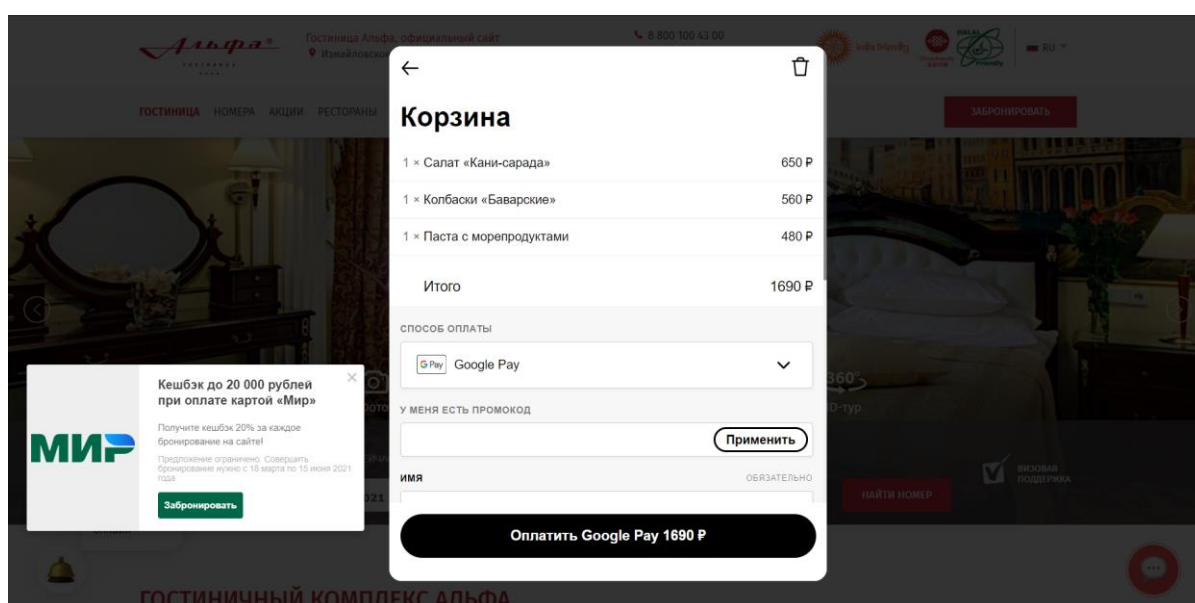


Рисунок 7 – Завершение оформление заявки

Также на территории ГК «Измайлово Альфа» оказываются и другие услуги, о которых можно детально узнать из навигационной панели. Например, аренда помещений, посещение тренажерного зала и ночного клуба.

Для сравнительного анализа программных продуктов-аналогов разрабатываемой системы были выделены такие критерии сравнения как:

- полнота представленной информации об оказываемых услугах (минимальная, средняя, полная);
- указание цен на большую часть предоставляемых услуг;
- возможность заказа дополнительных услуг отеля через сайт;

- наличие личного кабинета пользователя для отслеживания истории заказов;
- современный интерфейс взаимодействия (визуальное оформление);
- вариативность способов оплаты через сайт;
- наличие различных фильтров для сортировки услуг;
- наличие возможности онлайн консультации со специалистом.

Результаты сравнительного анализа информационных систем рассмотренных гостиничных комплексов представлен в таблице 1.

Таблица 1 – Сравнительный анализ продуктов-аналогов

Критерий для сравнения	«Heliopark Cruise»	«Рыбзавод»	«Измайлово Альфа»
Степень представленной информации об оказываемых услугах	Минимальная	Средняя	Полная
Указание цен на большую часть предоставляемых услуг	Нет	Да	Да
Возможность заказа дополнительных услуг отеля через сайте	Нет	Нет	Да
Наличие личного кабинета пользователя для отслеживания истории заказов	Да, но скрыт	Нет	Нет
Современный интерфейс взаимодействия	Нет	Да	Нет
Вариативность способов оплаты через сайт	Нет	Нет	Да
Наличие различных фильтров для сортировки услуг	Нет	Нет	Нет

Продолжение таблицы 1

Критерий для сравнения	«Heliopark Cruise»	«Рыбзавод»	«Измайлово Альфа»
Наличие возможности онлайн консультации со специалистом	Нет	Нет	Да

В результате сравнения представленных информационных систем (ИС) отелей было выявлено, что почти ни одна полностью не удовлетворяет минимальным требованиям клиента для оформления заявок на получение услуг через сайт и их детальным ознакомлением. Функционал большинства из них не позволяет детально ознакомиться с перечнем услуг, включая их стоимость. Так же было выявлено, что практически полностью отсутствует возможность авторизации во всех ИС, как и сохранение истории заказов, и отслеживание выполнения соответственно.

Исходя из результатов сравнения, было принято решение разработать информационную систему гостиничного комплекса с применением современных технологий, которая будет поддерживать работу с категориями пользователей, авторизацию и регистрацию, реализовывать их личные кабинеты, в которых будет отображаться информация об оформленных заявках. Кроме того, предусмотреть отображение полного каталога услуг с сортировкой и указанием цены, также подключить средства онлайн поддержки и администрирования.

Таким образом, принято решение в ходе выполнения выпускной квалификационной работы реализовать собственное ПО информационной системы, при разработке которого будут учтены преимущества и недостатки, рассмотренных выше программных аналогов.

1.3 Постановка задачи информационной системы сервисного обслуживания гостиничного комплекса

Данная выпускная квалификационная работа посвящена разработке информационной системы сервисного обслуживания гостиничного комплекса.

Главной задачей разрабатываемого ПО является возможность оформления заявки гостем через предоставляемый сервис. Процесс оформления реализуется в три этапа:

- предварительное оформление контракта клиента с гостиничным комплексом;
- авторизация или регистрация гостя на сайте;
- добавление необходимых услуг в корзину клиента и их оплата для подтверждения заказа.

Система ориентирована на конечного пользователя, не обладающего высокой квалификацией в области вычислительной техники. Разрабатываемое веб-приложение должно обладать удобным, простым и легко осваиваемым интерфейсом. Кроме того, должна быть разработана БД, которая будет обеспечивать надежное хранение и доступ к информации.

В разрабатываемом ПО информационной системы сервисного обслуживания следует реализовать следующие функции:

- оформление заявки гостем комплекса через веб-приложение путем добавления необходимых услуг в корзину и их последующей оплаты;
- выбор места предоставления услуг клиентом путем указания нужного контракта при подтверждении заявки;
- возможность удобной навигации, поиска и фильтрации предоставляемой информации;
- дружелюбный, визуально эстетичный интерфейс, который будет отлавливать исключительные ситуации и предостерегать об ошибках;

- возможность краткой аналитики работы сервиса гостиничного комплекса администратором путем формирования наглядных графиков и таблиц;
- возможность добавления, редактирования и удаления новых категорий услуг, самих услуг, а также пользователей и их контрактов администратором;
- предоставление возможности визуального контроля выполнения заявок для клиента, а также изменения их статусов для администратора;
- реализация обратной связи, путем добавления возможности комментирования и выставления рейтинга клиентом для приобретенных услуг.

Предполагаемой аудиторией будут являться гостиничные фирмы, предприятия и комплексы, заинтересованные в обеспечении онлайн поддержки и формирования заявок.

1.4 Анализ бизнес-процессов информационной системы сервисного обслуживания гостиничного комплекса

В рамках выпускной квалификационной работы необходимо разработать базу данных и приложение для информационной системы сервисного обслуживания гостиничного комплекса.

Выделим основные понятия предметной области «Сервисное обслуживание гостиничного комплекса»: заявка, контракт, услуга, категория услуг, учетная запись, роль. Основные функции: добавление, обслуживание и контроль заявки; анализ обращений клиентов, контроль предоставляемых услуг, управление учетными записями и сервисным каталогом. Рассмотрим подробнее описание предметной области.

Исходя из анализа предметной области и поставленной задачи были получены бизнес-процессы, представленные в таблицах 2 – 3.

Таблица 2 – Бизнес-процессы для подсистемы администратора

№ п/п	Бизнес-процесс	Исполнитель	Входные данные		Выходные данные	
			Поставщик	Содержание	Потребитель	Содержание
1	Добавление услуги	Администратор	Администратор	Категория услуги, сведения о ней	Гость	Новая услуга доступная для заказа
2	Удаление услуги	Администратор	Гостиничный комплекс	Идентификатор услуги	Гость	Услуга удалена
3	Изменение услуги	Администратор	Гостиничный комплекс	Идентификатор услуг	Гость	Обновление услуги
4	Добавление категории	Администратор	Администратор	Сведения о категории	Гость	Новая категория доступная для просмотра
5	Удаление категории	Администратор	Гостиничный комплекс	Идентификатор категории	Гость	Категория удалена
6	Изменение категории	Администратор	Гостиничный комплекс	Идентификатор категории	Гост	Обновление категории
7	Контроль учетных записей	Администратор	Гостиничный комплекс	Идентификатор заявки	Гость	Изменение данных
8	Аналитика заявок	Администратор	Гостиничный комплекс	Выбор соответствующей страницы	Администратор	Данные представленные в виде таблицы и графиков
9	Смена статуса заявки	Администратор	Гостиничный комплекс	Выбор нужного состояния	Гость	Изменение данных заявки

Также были получены бизнес-процессы для подсистемы клиента, потребителя гостиничных услуг.

Таблица 3 – Описание бизнес-процессов для подсистемы клиента

№ п/п	Бизнес-процесс	Исполнитель	Входные данные		Выходные данные	
			Поставщик	Содержание	Потребитель	Содержание
1	Создание заявки на обслуживание	Гость	Гостиничный комплекс	Категория необходимой услуги, оплата	Гость	Предоставление необходимой услуги
2	Удаление заявки	Гость	Гостиничный комплекс	Номер заявки	Гость	Заявка удалена, возвращение оплаты
3	Просмотр услуги	Гость	Гостиничный комплекс	Выбор соответствующей страницы, категории услуги	Гость	Данные об имеющихся и доступных услугах для оформления

Продолжение таблицы 3

№ п/п	Бизнес-процесс	Исполнитель	Входные данные		Выходные данные	
			Поставщик	Содержание	Потребитель	Содержание
4	Добавление услуги в избранное	Гость	Гостиничный комплекс	Выбор нужной услуги	Гость	Быстрый доступ к необходимой услуге
5	Отправка обратной связи	Гость	Гость	Вид услуги, оценка, отзыв	Гостиничный комплекс	Отзыв по оказанной услуге

В результате моделирования были описаны бизнес-процессы предметной области «Сервисное обслуживание гостиничного комплекса».

Для того чтобы наиболее полно представлять себе работу данной системы, следует провести концептуальное проектирование БД, используемое для описания предметной области, состоящей из множества сущностей и причинно-следственных связей между ними.

Исходя из описания бизнес процессов можно выделить такие пункты концептуальной модели предметной области, как описание типов сущностей предметной области, описание типов связей между типами сущностей предметной области и графическое представление концептуальной модели БД в виде диаграммы «сущность-связь».

Построение концептуальной модели состоит из описания типов сущностей предметной области (таблица 4) и описания типов связей между типами сущностей (таблица 5).

Таблица 4 – Описание сущностей предметной области

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Пользователь	Зарегистрированные пользователи в информационной системе	Стержневая	10000	100% в год
2	Статья	Статьи, размещенные на сайте	Характеристическая	150	100% в год
3	Избранное	Избранные услуги пользователей	Ассоциативная	5000	100% в год

Продолжение таблицы 4

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
4	Отзыв	Отзывы клиентов с рейтинговой системой	Ассоциативная	1000	100% в год
5	Категория услуг	Категории предоставляемых услуг гостиничного комплекса	Стержневая	100	10% в год
6	Услуга	Услуги, предоставляемые гостиничным комплексом	Характеристическая	1000	70% в год
7	История цен	Прайс услуг, в котором цены могут меняться в условиях БД	Характеристическая	3000	100% в год
8	Здание	Корпуса зданий для размещения гостей в гостиничном комплексе	Обозначающая	3	1% в год
9	Комната	Номера для размещения гостей в гостиничном комплексе	Характеристическая	200	1% в год
10	Контракт	Контракты, заключающиеся с гостями	Ассоциативная	8000	100% в год
11	Заявка	Заявки гостей	Ассоциативная	30000	100% в год
12	Заказ	Заказы гостей	Характеристическая	5000	100% в год
13	Состояние заявки	Связь заявки и состояния	Ассоциативная	100000	100% в год
14	Состояния	Состояния для заявок	Стержневая	5	1% в год

Описание типов связей между типами сущностей предметной области системы сервисного обслуживания приведено в таблице 5.

Таблица 5 – Описание типов связей между типами сущностей предметной области

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность
1	Пользователь	Здание	Один администратор может быть привязан к нескольким зданиям	1:M	Не обязательно

Продолжение таблицы 5

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность
2	Пользователь	Контракт	Один клиент может иметь несколько контрактов	1:M	Обязательно
3	Пользователь	Услуга	Разные пользователи могут пользоваться разными услугами	M:M	Обязательно
4	Пользователь	Статья	У одной статьи может быть указан только один автор	1:1	Обязательно
5	Здание	Комната	В одном корпусе находится много комнат (номеров)	1:M	Обязательно
6	Комната	Контракт	На одну комнату заключается много контрактов	1:M	Обязательно
7	Контракт	Заказ	На один контракт может оформляться только один заказ	1:1	Обязательно
8	Заказ	Заявка	В одном заказе может содержаться несколько заявок	1:M	Обязательно
9	Заявка	Состояния	Разные заявки могут иметь разные состояния	M:M	Обязательно
10	Категория услуг	Услуга	В одну категорию входит несколько услуг	1:M	Обязательно
11	Услуга	Заявка	На одну услугу может составляться несколько заявок	1:M	Обязательно
12	Услуга	История цен	У одной услуги может быть несколько изменений цен	1:M	Обязательно
13	Услуга	Избранное	В избранном может быть несколько услуг	1:M	Обязательно
14	Услуга	Отзыв	В избранном может быть несколько услуг	1:M	Обязательно
15	Пользователь	Избранное	Один клиент может иметь несколько избранных услуг	1:M	Обязательно
16	Пользователь	Отзыв	Один клиент может быть автором отзывов на несколько услуг	1:M	Обязательно
17	Категория услуг	Категория услуг	У одной категории может быть несколько подкатегорий	1:M	Не обязательно
18	Заявка	Состояние заявки	Заявка может находиться в нескольких состояниях	1:M	Обязательно
19	Состояния	Состояние заявки	В состоянии заявки может быть указано только одно состояние	1:M	Обязательно

По результатам формирования сущностей предметной области была спроектирована концептуальная модель, построенная в нотации IDEF1X. Данная модель представлена виде диаграммы «сущность-связь», на которой выделены сущности, их атрибуты и связи между ними (рисунок 8).

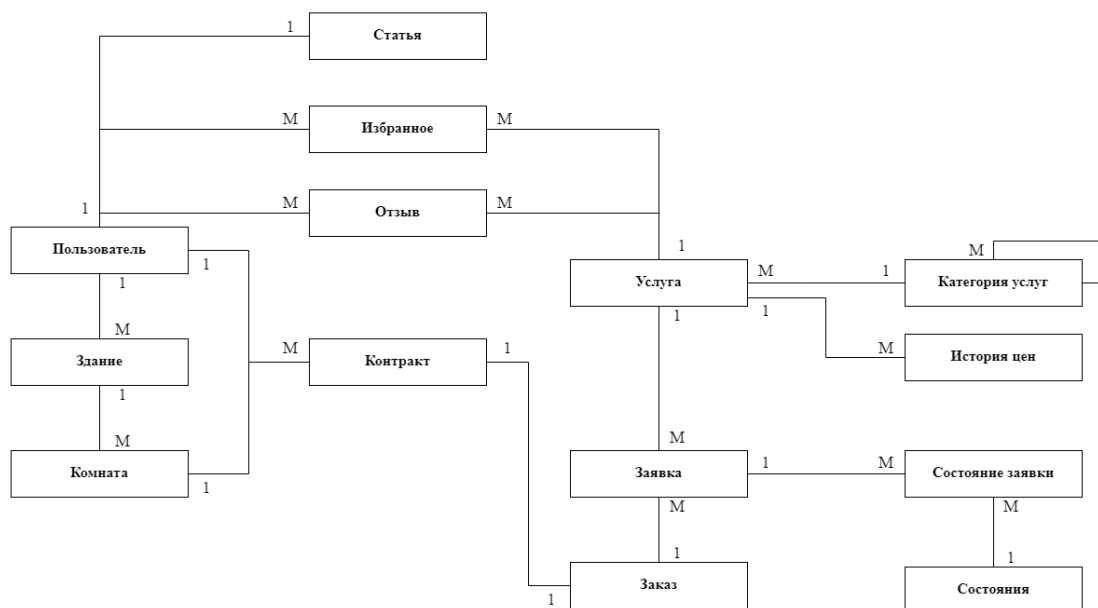


Рисунок 8 – Диаграмма «сущность-связь»

Спроектированная диаграмма визуально показывает связи между выделенными сущностями, а также впоследствии будет использована для построения логической схемы.

1.5 Анализ требований информационной системы сервисного обслуживания гостиничного комплекса

Анализ требований – часть процесса разработки программного обеспечения, включающая в себя сбор требований к программному обеспечению (ПО), их систематизацию, выявление взаимосвязей, а также документирование. Можно выделить нефункциональные и функциональные требования.

Нефункциональные требования – это свойства, которыми должен обладать проект. Также к нефункциональным требованиям относятся ограничения проекта, но они не должны определять его поведение.

Функциональные требования определяют задачи или действия, которые проект может исполнить или же, что проект выполнять не должен.

1.5.1 Анализ функциональных требований

Перед разрабатываемой информационной системой стоит задача обеспечения необходимого функционала как для клиента, так и для администратора гостиничного комплекса. Данная система должна позволять клиенту совершать следующие действия:

- регистрация пользователя;
- аутентификация пользователя;
- поиск услуг по названию;
- поиск услуг по категориям;
- фильтрация по критериям;
- добавление услуги в список избранных товаров;
- отправление обратной связи;
- добавление услуги в корзину;
- оформление заявки с последующим обслуживанием.

Для администратора информационная система должна предоставлять функционал управления предоставляемым сервисом отеля:

- аутентификация пользователя;
- просмотр данных о сервисе в удобной форме;
- добавление новых позиций услуг;
- добавление новых категорий услуг;
- контроль оказания заявок;
- создание контрактов для клиента;
- управление учетными записями.

Пользователем является неавторизованный посетитель веб приложения, возможности которого ограничены и заключаются в просмотре статичных страниц с общей информацией, а также каталога услуг, но без возможности оформления заявки и т.п.

Клиент выступает в роли зарегистрированного, авторизованного пользователя. Он может просматривать каталог услуг посредством перехода

в нужную категорию. На странице категории клиент имеет возможность выставить критерии для фильтрации выводимых услуг. Также он может воспользоваться строкой поиска услуги по названию, чтобы сразу найти нужную, зная ее наименование. Конкретная услуга может быть добавлена пользователем в список избранного или же в корзину для дальнейшего оформления заявки и ее обслуживания. Важным пунктом является то, что не каждый авторизованный клиент, может оформлять заявки, а только тот, кто имеет контракт с гостиничным комплексом.

Таким образом система фильтрует пользователей, не проживающих на территории отеля, а также обеспечивает оказание услуги конкретному гостю в конкретный номер.

Проектирование диаграмм будет проводится в среде ArgoUML 0.34. Исходя из описанных требований, была разработана диаграмма вариантов использования для актера клиент, представленная на рисунке 9.

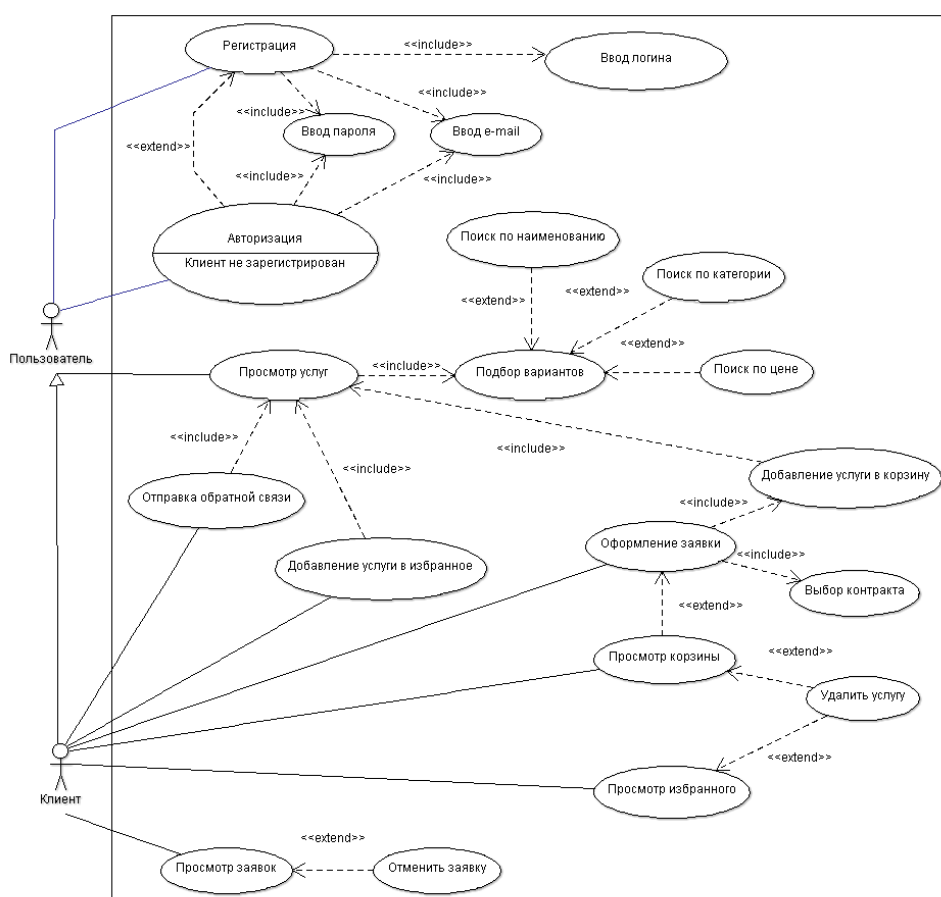


Рисунок 9 – Диаграмма вариантов использования для клиента

На рисунке 10 представлена диаграмма вариантов использования для актера администратор, пользователя системы, который имеет доступ к особым страницам, обеспечивающим функционал по управлению данными.

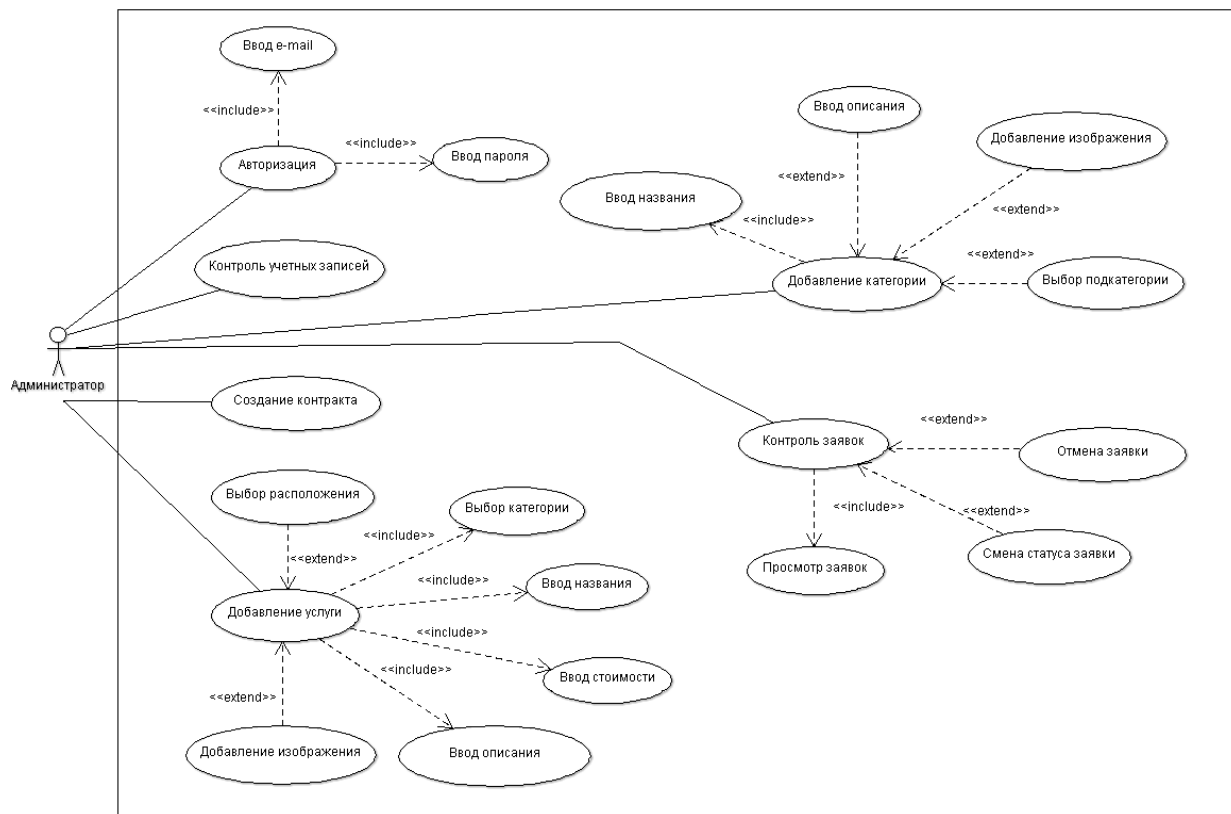


Рисунок 10 – Диаграмма вариантов использования для администратора

Данные диаграммы описывают функциональные требования для системы. В результате проектирования было выделено три актера: пользователь, который предоставляет общий функционал, а также клиент и администратор.

Администратор так же должен быть авторизован при помощи специальной электронной почты и пароля, которые хранятся в базе данных. Введя их, пользователь, кроме как просмотра каталога отеля со стороны клиента, получает доступ к специальной странице – панели администрирования. Средствами предоставляемого интерфейса он может контролировать процесс оказания заявок, управлять услугами, а также

Для взаимодействия с системой пользователю необходимо авторизоваться. В соответствующие поля он вводит электронную почту и пароль. Если клиент с такими данными зарегистрирован, то сервер перенаправляет его на домашнюю страницу, откуда он начинает оформление подходящих ему услуг. Для этого можно воспользоваться поиском по наименованию, или же подобрать услуги вручную. Сам клиент может отфильтровать каталог посредством выбора нужных характеристик. После получения критериев для фильтрации или же названия услуги, система возвращает список с нужными данными, которые отображаются на странице в удобной форме. Далее гость гостиничного комплекса добавляет необходимую услугу в корзину, данные которой не сохраняются в БД, так как в качестве хранилища используется сессия.

На данном этапе пользователь может продолжить формирование корзины или же оформить заказ с имеющимися заявками. Для этого система перенаправит его на соответствующую страницу заполнения формы для заказа. После заполнения данных о заказе и его оплате содержащиеся в нем заявки, а также сам заказ будут добавлены в БД.

В заключение клиент может контролировать процесс выполнения услуги в личном кабинете, запрашивая ее статус от сервера.

При проектировании информационной системы оформления заявок и получением услуг гостиничного комплекса были составлены следующие бизнес процессы. Они представлены в виде диаграмм, созданных при помощи CA ERwin Process Modeler. Это CASE-средство визуального моделирования, поддерживающее такие методологии моделирования, как IDEF0, IDEF3 и DFD.

IDEF0 является методологией графического описания деятельности какой-либо организации как множества связанных функций. С ее помощью возможно исследовать процессы организации, не связывая их с объектами, обеспечивающими их реализацию [6].

Контекстная диаграмма бизнес-процессов проектируемой системы в нотации IDEF0 представлена на рисунке 12, рассмотрен более детально основной процесс клиента по оформлению заявки на обслуживание.

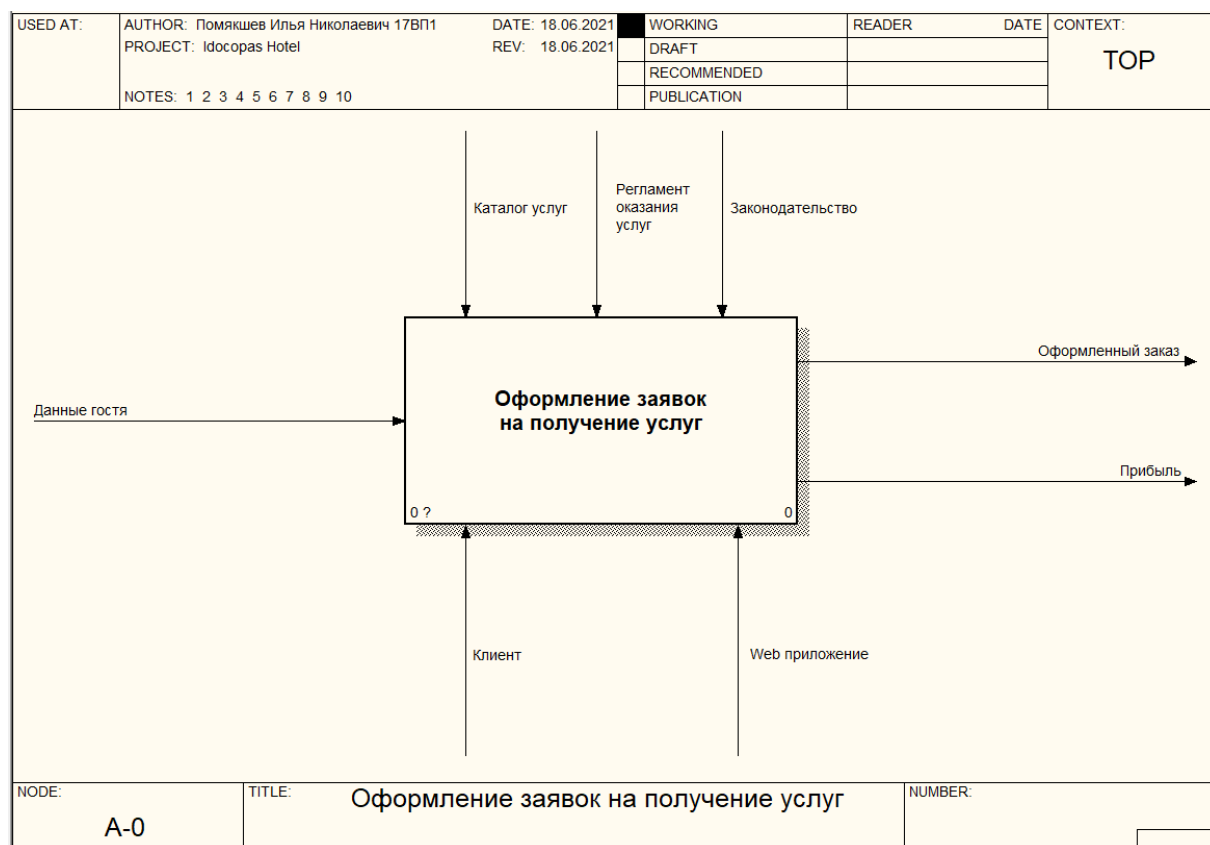


Рисунок 12 – Контекстная диаграмма в нотации моделирования IDEF0

На вход поступают данные гостя, пытающегося оформить заявку на получение услуги. Механизмами представлены клиент, и само веб-приложение. Управляющим воздействием являются законодательство, регламентирующее правила оказания услуг, а также сам сервисный каталог гостиничного комплекса. Результатом будет оформленный заказ на обслуживание, содержащий необходимую заявку, а также прибыль от приобретения клиентом данной услуги.

Далее производится разбиение системы на фрагменты, для ее более подробного анализа. Главный процесс был разделен на следующие компоненты: поиск и просмотр услуг, добавление услуг в корзину, подтверждение заказа, а также сохранение заявок в БД. На рисунке 13 представлена декомпозированная диаграмма.

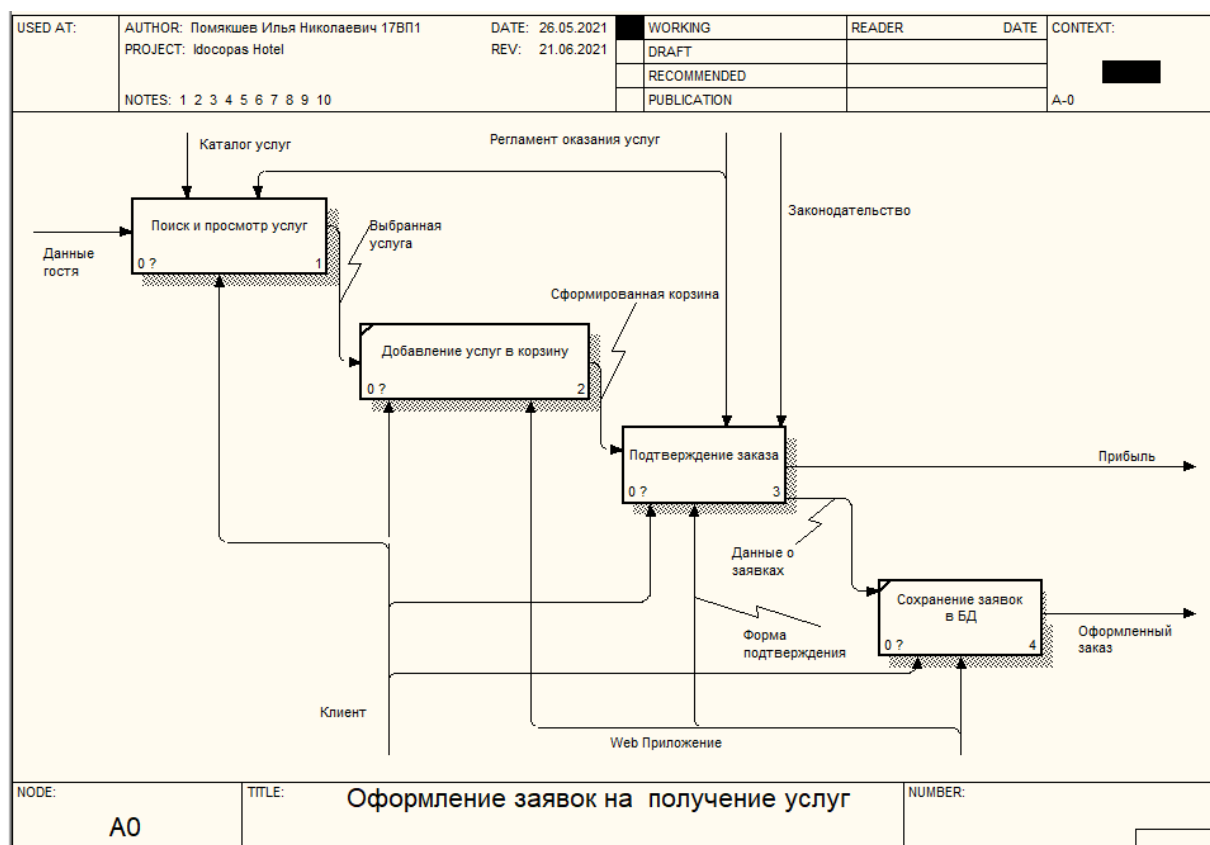


Рисунок 13 – Декомпозированная контекстная диаграмма в нотации моделирования IDEF0

На вход поступают данные об успешной авторизации клиента, который производит анализ представленного каталога услуг. Затем нужную услугу он добавляет в корзину, чтобы в дальнейшем приступить к оформлению покупки. После успешного подтверждения заказа и оформления заявка будет добавлена в БД, тем самым будет управляться информационной системой и принята на обслуживание.

Декомпозиция бизнес процесса подтверждение заказа происходит в нотации DFD. Это методология структурного анализа, описывающая внешние источники данных, потоки данных, логические функции, хранилища и потоки данных по отношению к системе [6].

Декомпозиция процесса подтверждения заказа представлена на рисунке 14.

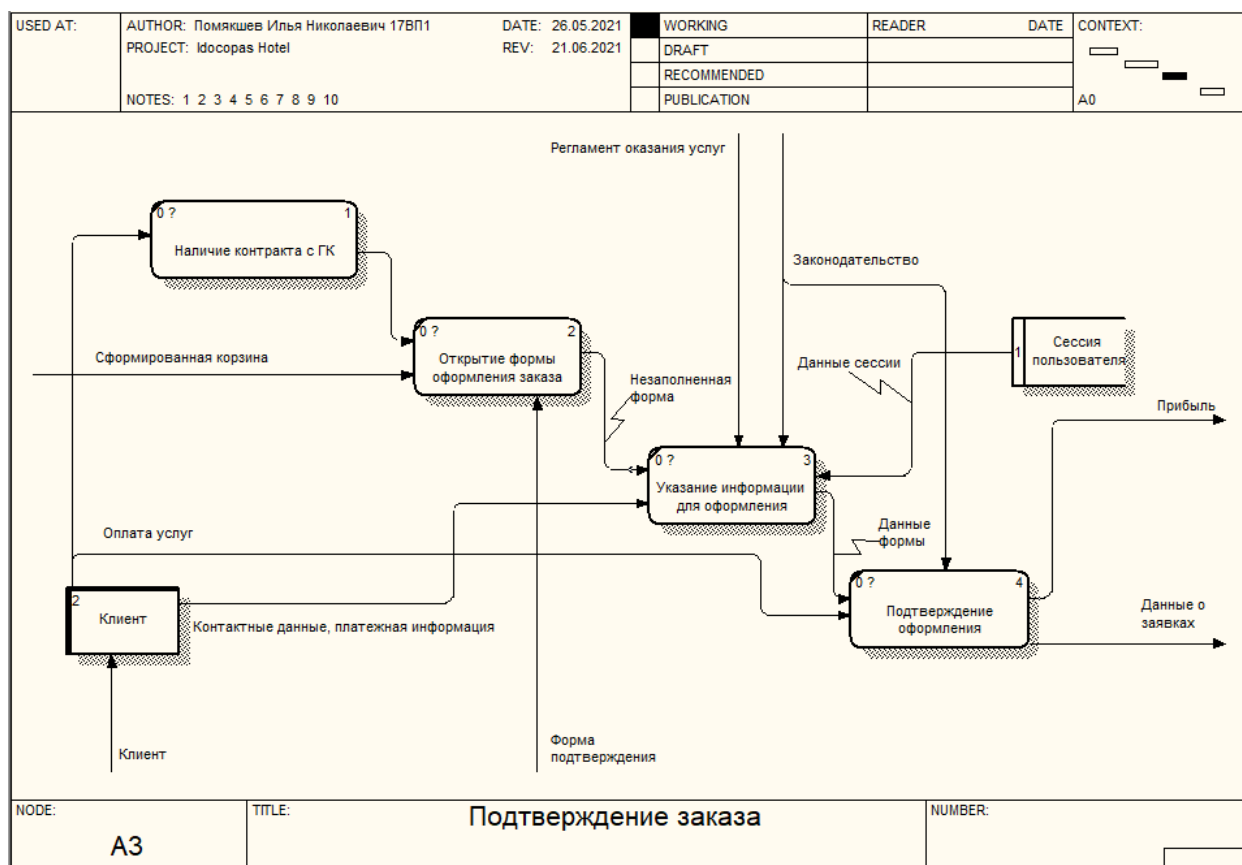


Рисунок 14 – Диаграмма декомпозиции в нотации моделирования DFD

Прежде всего для успешного оформления заявки пользователь должен иметь контракт с гостиничным комплексом (ГК), тем самым являясь его гостем. После чего пользователь открывает форму оформления заказа и вводит данные, необходимые для ее заполнения, в том числе номер необходимого контракта, который содержит информацию о текущем проживании клиента на территории ГК. Далее происходит подтверждение оформления, путем валидации введенных данных, а также списанием средств с указанных реквизитов.

Декомпозиция бизнес процесса поиска и просмотра услуг происходит в нотации IDEF3. Данная методология является стандартом документирования технологических процессов. Она предоставляет возможности для наглядного исследования и моделирования сценариев процессов, происходящих на предприятии. [6].

Декомпозиция бизнес процесса поиска и просмотра услуг представлена на рисунке 15.

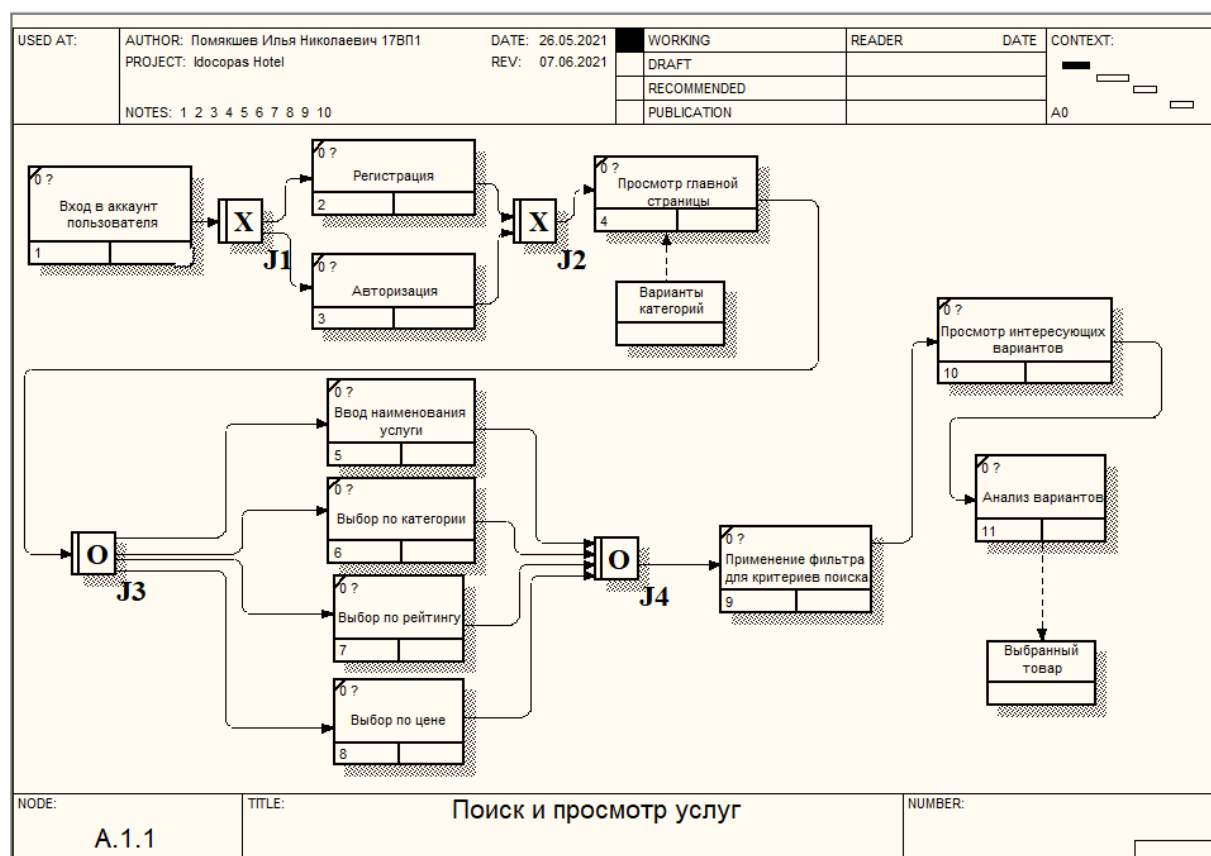


Рисунок 15 – Диаграмма декомпозиции в нотации моделирования IDEF3

Просмотр услуг для клиента становится доступен только после регистрации или авторизации. Выполнение этого шага показано перекрестком, исключающим «ИЛИ». После этого клиент попадает на главную страницу и ему доступен просмотр категорий. Далее пользователь попадает на еще один перекресток асинхронного «ИЛИ». Он может сразу ввести название нужного ему товара и найти его путем критериев фильтрации. После применения одного или же нескольких из перечисленных способов, клиенту будет предоставлен результат в виде отфильтрованных (отсортированных) услуг. Далее после просмотра подробной информации об услугах, их сравнении и анализе, клиент может перейти к их оформлению в виде заявки, что является завершающим шагом.

1.5.2 Анализ нефункциональных требований

Разрабатываемая информационная система должна соответствовать следующим нефункциональным требованиям.

Информационная система должна запускаться на веб-странице при помощи любого из современных браузеров. Система будет разработана с помощью кроссплатформенной среды ASP.NET Core MVC, предназначенной для создания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов, которая сочетает в себе эффективность и аккуратность архитектуры «модель-представление-контроллер» [7]. Проектирование будет вестись с помощью визуальной среды Visual Studio 2019.

Данная среда является кроссплатформенной и высокопроизводительной, позволяет выполнять следующие задачи [8]:

- создавать веб-приложения и службы, приложения Интернета вещей и серверные части для мобильных приложений;
- создавать веб-приложения и службы, приложения Интернета вещей и серверные части для мобильных приложений;
- использовать избранные средства разработки в Windows, macOS и Linux;
- выполнять развертывания в облаке или локальной среде;
- запускать в .NET Core.

Кроме этого ASP.NET легко интегрируется с распространенными клиентскими платформами и библиотеками, в том числе Blazor, Angular, React и Bootstrap, также одним из отличительных моментов платформы является применение паттерна MVC, который упрощает процесс создания архитектурно правильных приложений, легко поддерживающих расширение.

Для разработки информационной системы гостиничного комплекса были выделены следующие нефункциональные требования:

- необходимо исключить возникновение ситуаций, приводящих к аварийному завершению работы системы;
- разработать удобный и функциональный интерфейс для пользователя системы, корректно работающий для разных разрешений экрана.

ПО информационной системы гостиничного комплекса должно обладать простым, удобным пользователю интерфейсом, позволяющим использовать все ключевые функции системы.

1.6 Выбор инструментальных средств

Для хранения данных о работы сервисного обслуживания гостиничного комплекса было принято решение об использовании реляционной БД, а в качестве СУБД – Microsoft SQL Server 2019. Данная СУБД отличается от других высокой отказоустойчивостью и производительностью, а также поддерживает декларированный язык программирования SQL для модификации и управления данными в БД [9]. Выбор SQL связан с тем, что он наиболее популярен среди языков программирования для работы с БД и является одним из лучших средств для реализации ПО архитектуры клиент-сервер. SQL служит связующим звеном между взаимодействующей с пользователем клиентской и серверной системами, позволяя каждой из них сосредоточиться на выполнении своих функций [10].

Разработка информационной системы сервисного обслуживания будет проводиться с помощью проектирования разнообразных моделей. С этой целью был выбран унифицированный язык моделирования UML, который объединил в себе наилучшие приемы разработки и моделирования ПО. Выбор UML связан с тем, что он позволяет описать систему практически со всех возможных точек зрения и затронуть разные аспекты поведения системы [6].

Проектирования логической и физической моделей БД предметной области осуществлялось с помощью CASE-средства ERwin Data Modeler 7.3.

Для разработки ПО данной системы был выбран язык объектно-ориентированного программирования (ООП) С#. Данный выбор связан с тем, что он поддерживается огромным количеством инструментов, надежен и устойчив благодаря обработке исключений и безопасности типов. С# располагает набором библиотек, представляющих возможность как быстрого создания графического интерфейса пользователя приложения, так и установления взаимодействия с БД [8]. В частности, была подключена ORM-система Entity Framework Core 5.0, которая используется для проецирования таблиц БД на объекты С# и изначально создана для Microsoft SQL Server, таким образом достигается согласованность используемых компонентов [11]. Подобная технология позволяет достичь высокого уровня абстракции над данными, а также предоставляет возможность написания LINQ запросов к БД, что упрощает создание кода, а также делает его более понятным и органичным. Интегрированная среда разработки – Microsoft Visual Studio 2019.

Для создания графического интерфейса пользователя был использован язык разметки HTML, таблицы стилей CSS и JS веб-сценарии, кроме этого был подключен фреймворк Bootstrap 5.0. Главное преимущество данного фреймворка – это, несомненно, заложенная внутрь адаптивная сетка, которая позволяет автоматически изменять свои размеры при изменении ширины окна [12]. Адаптивность является ключевым элементом при разработке интерфейса приложения, т.к. независимо от роли пользователя, элементы формы должны корректно отображаться на различных устройствах. Bootstrap содержит в себе готовые решения для формирования подобного интерфейса и легко интегрируется с ASP.NET [7].

Для аутентификации и авторизации пользователей был использован пакет ASP.NET Core Identity. Данный API с открытым исходным кодом управляет пользователями, паролями, данными профилирования, ролями, утверждениями, маркерами, подтверждением электронной почты и т.д., позволяя сосредоточиться на разработке основных элементов.

В качестве продукта управления проектами был выбран Microsoft Project 2016, позволяющий оптимизировать планирование проекта путем распределения ресурсов по стоящим задачам, анализа объема работ, составления расписания критического пути, а также наглядно показать весь процесс разработки на сетевом графе и диаграмме Ганта [13].

2 Проектирование информационной системы сервисного обслуживания гостиничного комплекса

2.1 Базовая архитектура системы

Веб-приложения являются определенными типами программ, построенных по архитектуре «клиент-сервер». Главная особенность таких приложений состоит в том, что само Веб-приложение находится и выполняется на сервере, в то время как клиент при этом получает только результаты работы. Работа приложения строится на основе полученных HTTP запросов от пользователя (клиента), их обработке и выдачи результата.

К преимуществам данной архитектуры можно отнести [14]:

- низкую нагрузку на средство клиента и информационную сеть, в связи с передачей лишь запросов и ответов на них;
- данные, хранящиеся на сервере, как правило, защищены гораздо лучше большинства клиентов;
- данные в систему вводятся только один раз, вследствие чего исчезает необходимость в дублировании информации и снижается риск ошибок в системе.

Главным недостатком данной архитектуры является зависимость информационной сети и сервера, т.к. неработоспособность сервера может сделать неработоспособной всю сеть, из-за чего будет невозможна работа с БД [14]. При разработке информационной системы сервисного обслуживания на ASP.Net Core 5.0 в шаблоны проекта уже включается веб-сервер Kestrel. Kestrel – это кроссплатформенный веб-сервер ASP.Net Core на основе Libuv, Libuv – это кроссплатформенная библиотека асинхронного ввода-вывода [8].

В рамках выполнения выпускной квалификационной работы и проверки работоспособности запросы получались из внутренней сети, при развертывании рекомендуется использовать обратный прокси-сервер, для первоначальной обработки запроса и передачи на Kestrel. Примером может быть IIS или же Apache. Конечная архитектура представлена на рисунке 16.

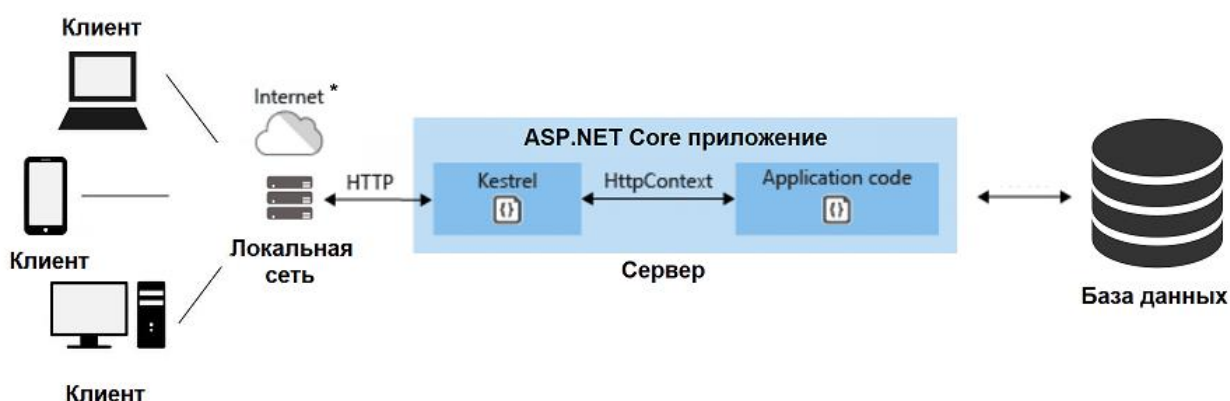


Рисунок 16 – Клиент-серверная архитектура системы

В представленной архитектуре (смотри рисунок 16) показаны три основных компонента системы: БД, сервер и клиент.

Главным компонентом является база данных, которая хранит информацию о пользователях, их контрактах, заявках, каталоге услуг гостиничного комплекса и т.д.

Центральный компонент представлен в виде сервера Kestrel, располагающегося на отдельной станции вместе с БД и имеющего доступ к ней.

В роли третьего компонента выступает клиентское приложение, являющееся интерфейсом взаимодействия между пользователем и сервером.

Клиент при необходимости отправляет HTTP запрос на обновление данных к серверу, после чего Kestrel преобразует его, отправляет на выполнение команду приложению. Если задача не требует обращения к БД, то формируется ответ и возвращается клиенту, иначе предварительно предпринимается попытка обновления данных через БД.

2.3 Проектирование пользовательского интерфейса

Опираясь на диаграмму вариантов использования и анализ требований, был спроектирован интерфейс взаимодействия для клиента информационной системы. Пользовательский интерфейс приложения должен обеспечивать привычное и удобное представление информации, простое и эффективное выполнение основных функций пользователей, при этом сохраняя внешнюю привлекательность, а также адаптивность.

На рисунках 18, 19 представлен графический интерфейс стартовой страницы гостиничного комплекса.

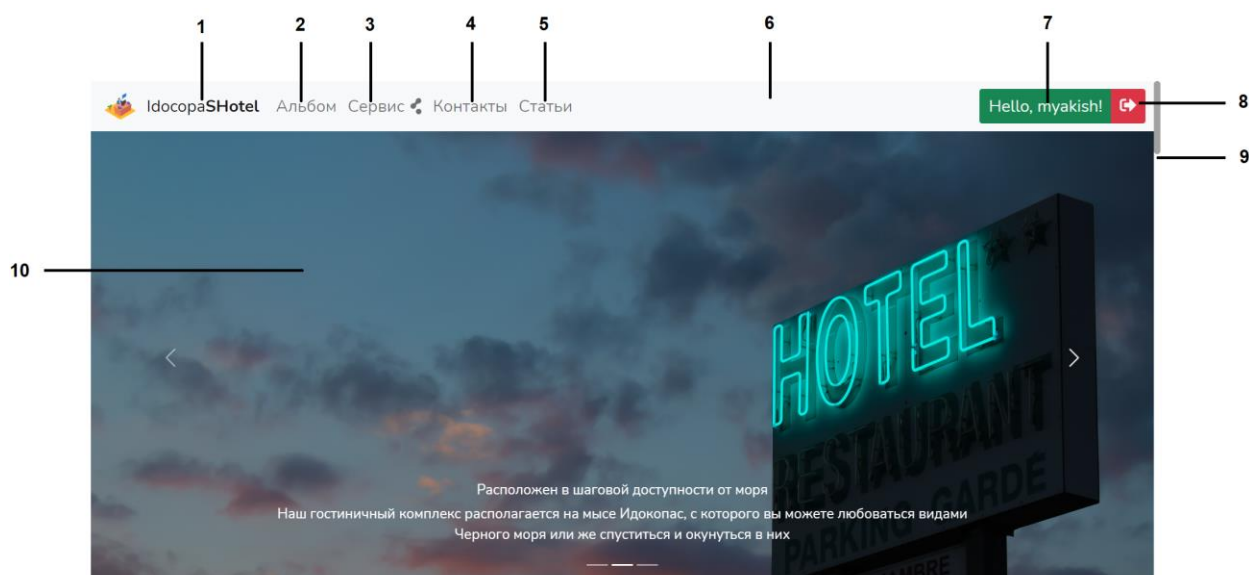


Рисунок 18 – Графический интерфейс домашней страницы
(стартовая страница)

Рассмотрим основные элементы интерфейса, изображенные на рисунках 18, 19:

- 1 – ссылка на домашнюю страницу;
- 2 – ссылка на страницу, содержащую фотографии работы ГК для детального ознакомления с его работой;
- 3 – ссылка на страницу, отвечающую за работу с сервисом, основным вариантом при работе с информационной системой;
- 4 – ссылка на страницу, содержащую контактную информацию;

- 5 – ссылка на страницу, содержащую обновляемые статьи о работе ГК;
- 6 – закреплённая сверху навигационная панель;
- 7 – ссылка, перенаправляющая на страницу личного кабинета и его настройки;
- 8 – кнопка выхода из системы и учётной записи;
- 9 – ползунок, позволяющий перемещаться по легенде сайта для более детального ознакомления с ним и получения подсказок;
- 10 – шапка домашней страницы, выполненная в виде слайдера изображений с подписями (карусель).

Графический интерфейс страницы авторизации и регистрации изображен на рисунках 20, 21. Данная область представлена 2 вкладками, переходя по которым пользователь может авторизоваться либо зарегистрироваться.

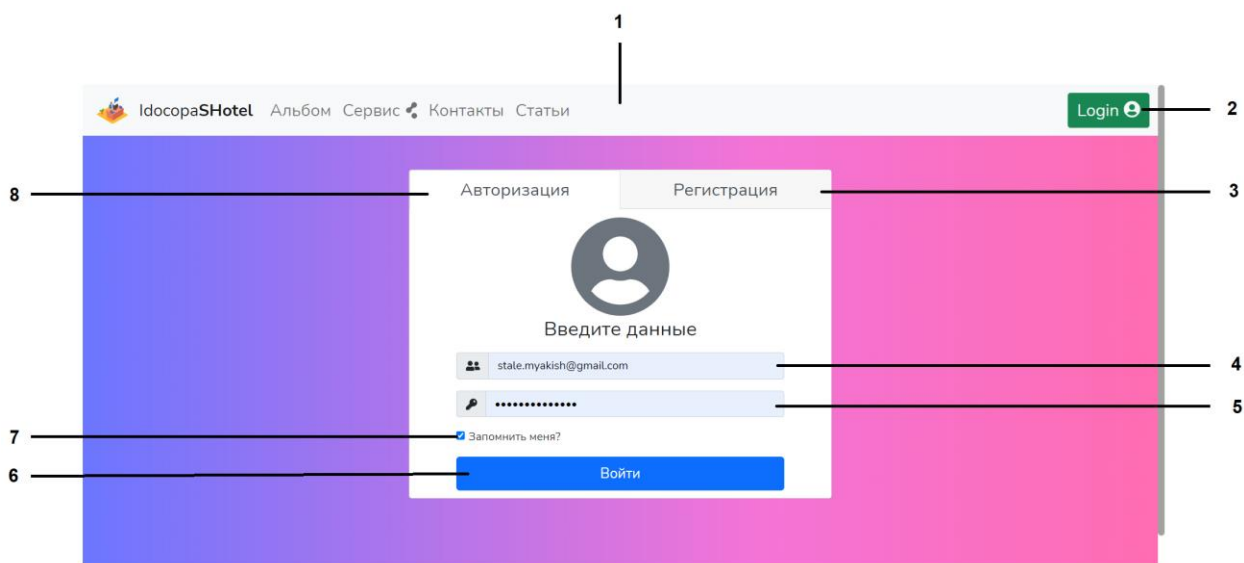


Рисунок 20 – Графический интерфейс вкладки «Авторизация»

Рассмотрим основные элементы интерфейса, изображенные на рисунках 20, 21:

- 1 – закреплённая сверху навигационная панель;
- 2 – ссылка на страницу авторизации;
- 3 – ссылка на страницу регистрации;
- 4 – поле, предназначенное для ввода email пользователя;

- 5 – поле, предназначенное для ввода пароля пользователя;
- 6 – кнопка отправки введенных данных, попытка авторизации;
- 7 – поле для указания хранения Cookies ради автоматической авторизации;
- 8 – текущая вкладка авторизации, выделяется белым.

Рассмотрим составляющие второй вкладки, предназначенной для регистрации (рисунок 21):

- 1 – кнопка отправки данных формы для попытки регистрации, валидация полей;
- 2 – поля, предназначенные для ввода и подтверждения пароля, используемого при авторизации;
- 3 – пример сообщения о неуспешной валидации (не заполнено поле «Имя»);

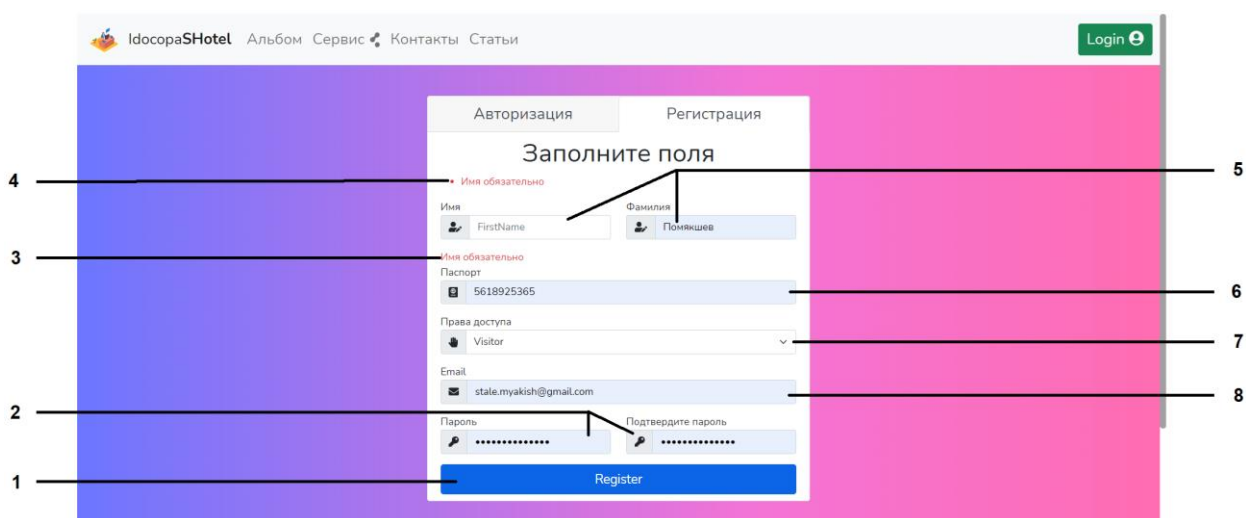


Рисунок 21 – Графический интерфейс вкладки «Регистрация»

- 4 – список всех ошибок, возникших при регистрации;
- 5 – поля, предназначенные для ввода имени и фамилии пользователя;
- 6 – поле, предназначенные для ввода паспорта пользователя;
- 7 – выпадающий список для выбора роли учетной записи, по умолчанию доступна только роль «Visitor» – клиент без контракта с ГК;
- 8 – поля, предназначенные для ввода email пользователя, используемого при авторизации.

После авторизации клиент сможет перейти в основную область сайта, отвечающую за работу с сервисом информационной системы. Интерфейс данной страницы представлен на рисунке 22.

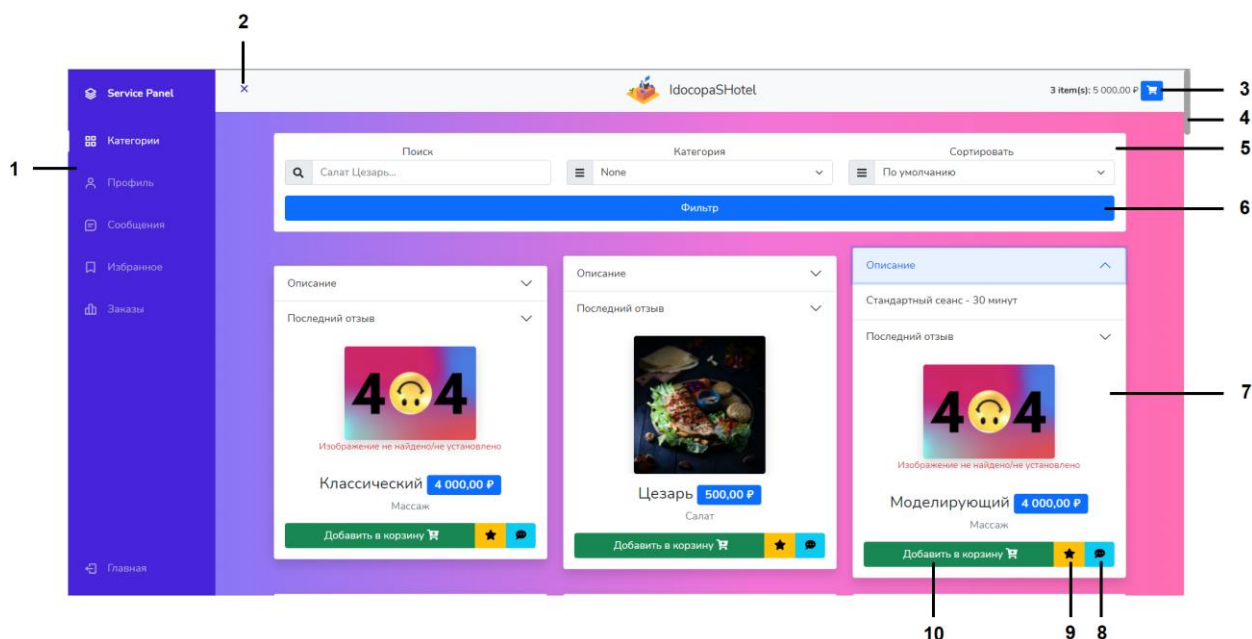


Рисунок 22 – Графический интерфейс страницы «Сервис»

Рассмотрим основные элементы интерфейса страницы «Сервис» (смотри рисунок 22):

- 1 – адаптивное боковое меню навигации;
- 2 – кнопка для сворачивания меню навигации;
- 3 – ссылка на страницу, отображающую информацию в корзине, краткая подпись слева о количестве и общей стоимости услуг;
- 4 – ползунок для перемещения по странице;
- 5 – область фильтрации услуг (прямой поиск, фильтр по категории, способы сортировки);
- 6 – кнопка применения критериев фильтрации;
- 7 – карточка услуги, визуально предоставляющая основные данные о ней;
- 8 – ссылка, открывающая модальное окно для написания отзыва об услуге;

9 – кнопка добавления услуги в список избранного для быстрого доступа к ней;

10 – кнопка для добавления услуги в корзину, с последующим переходом в нее.

Рассмотрим интерфейс корзины и его основные элементы, изображенные на рисунке 23:

1 – адаптивное боковое меню навигации (свернутый вид);

2 – кнопка для разворачивания меню навигации;

3 – выпадающий список для изменения отображаемого количества услуг на странице таблицы;

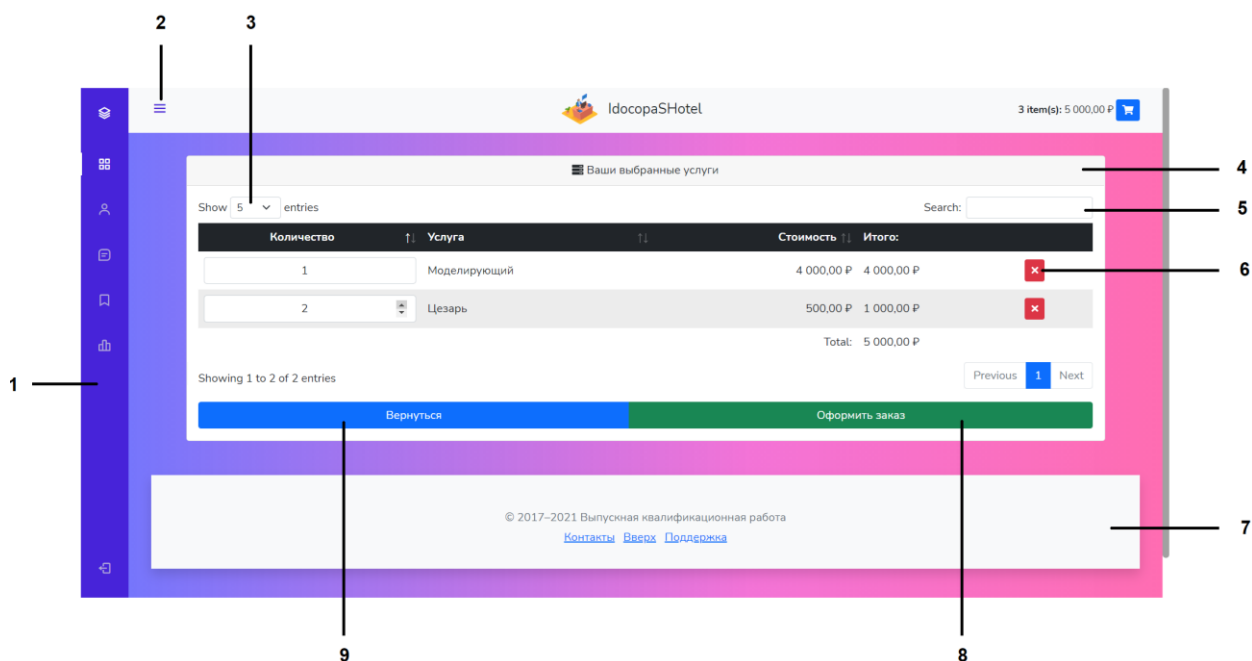


Рисунок 23 – Графический интерфейс страницы «Корзина»

4 – карточка, содержащая информацию о корзине;

5 – поле поиска услуг в корзине;

6 – кнопка удаления услуги и всех ее повторений из корзины;

7 – подвал сайта;

8 – кнопка отправки данных корзины на сервер для дальнейшего формирования заказа;

9 – ссылка на предыдущую страницу, с которой был осуществлен переход в корзину.

На этом основной функционал работы клиента для оформления заявки рассмотрен. Теперь опишем основные элементы интерфейса подсистемы администратора. Главная страница администрирования представлена на рисунке 24:

- 1 – панель, отображающая краткую аналитику по заказам среди основной категории ГК, «Рестораны и бары»;
- 2 – адаптивное боковое меню навигации;

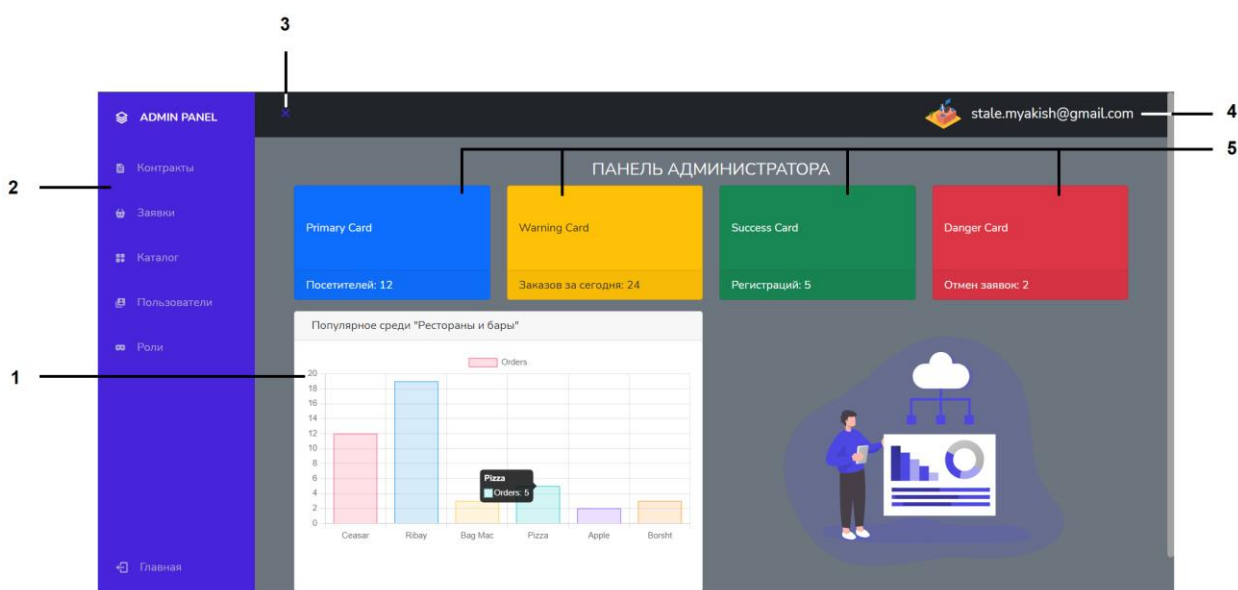


Рисунок 24 – Графический интерфейс панели администрирования

- 3 – кнопка для сворачивания меню навигации;
- 4 – почта текущего администратора;
- 5 – четыре блока, отображающих информацию о работе сайта: посетителях, заказах, регистрациях и отменах заявок за день.

Рассмотрим основную страницу по управлению каталогом услуг, куда можно перейти выбрав вариант «Каталог» на боковом навигационном меню. Данная страница содержит как возможность управления услугами, так и их категориями, для краткости будет рассмотрен схожий функционал при управлении их данными.

Основные элементы страницы администрирования «Каталог» (рисунок 25):

- 1 – кнопка для разворачивания меню навигации;

2 – область фильтрации отображения (отображать какую-то конкретную таблицу или обе, отображать только неактивные варианты);

3 – кнопка применения критериев фильтрации;

4 – карточка, содержащая информацию об услугах в виде таблицы;

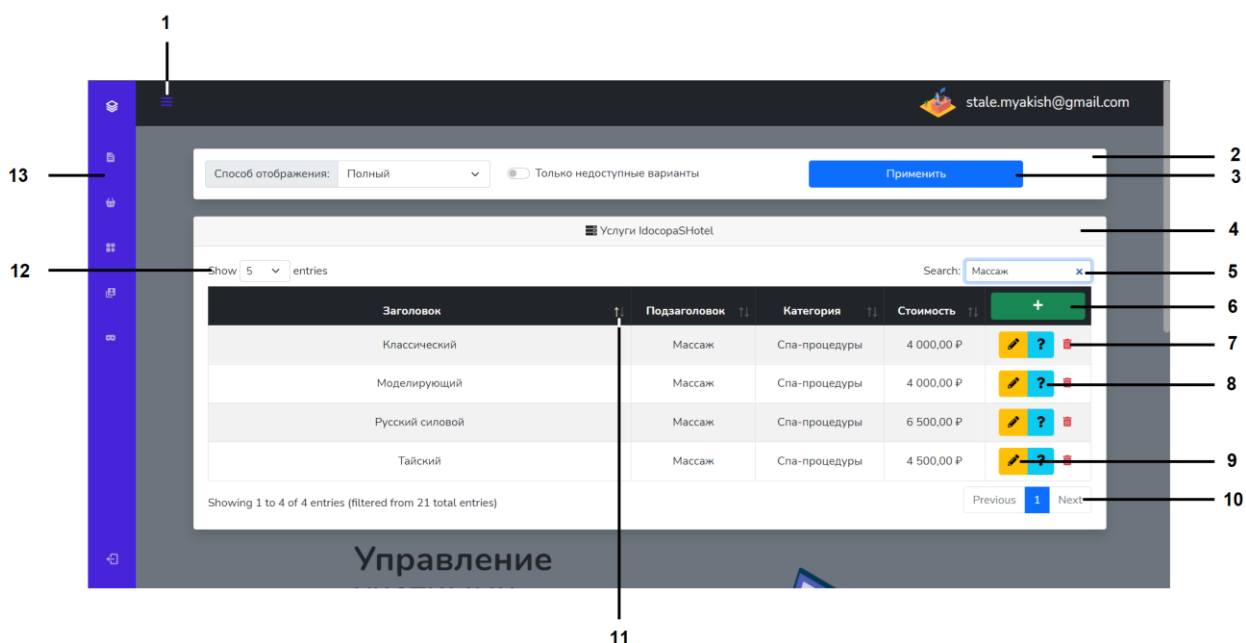


Рисунок 25 – Графический интерфейс страницы «Каталог»

5 – поле поиска услуг в таблице;

6 – ссылка на модальное окно добавления новой услуги;

7 – ссылка на модальное окно подтверждения удаления услуги;

8 – ссылка на модальное окно подробной информации об услуге;

9 – ссылка на модальное окно изменения услуги;

10 – постраничная навигация в таблице;

11 – сортировка записей для выбранной колонки в таблице;

12 – выпадающий список для изменения отображаемого количества услуг на странице таблицы;

13 – адаптивное боковое меню навигации (свернутый вид).

Рассмотрим вариант изменения услуги, который представлен модальным окном, что открывается поверх исходной страницы и обеспечивает удобство и плавность в работе. Интерфейс идентичен созданию

услуги, т.к. отличия заключаются лишь в заполнении полей данными в случае их изменения (рисунок 26):

- 1 – поле для ввода заголовка услуги;
- 2 – модальное окно;
- 3 – поле для ввода подзаголовка услуги;
- 4 – кнопка закрытия модального окна (то же при выборе затемненной области);

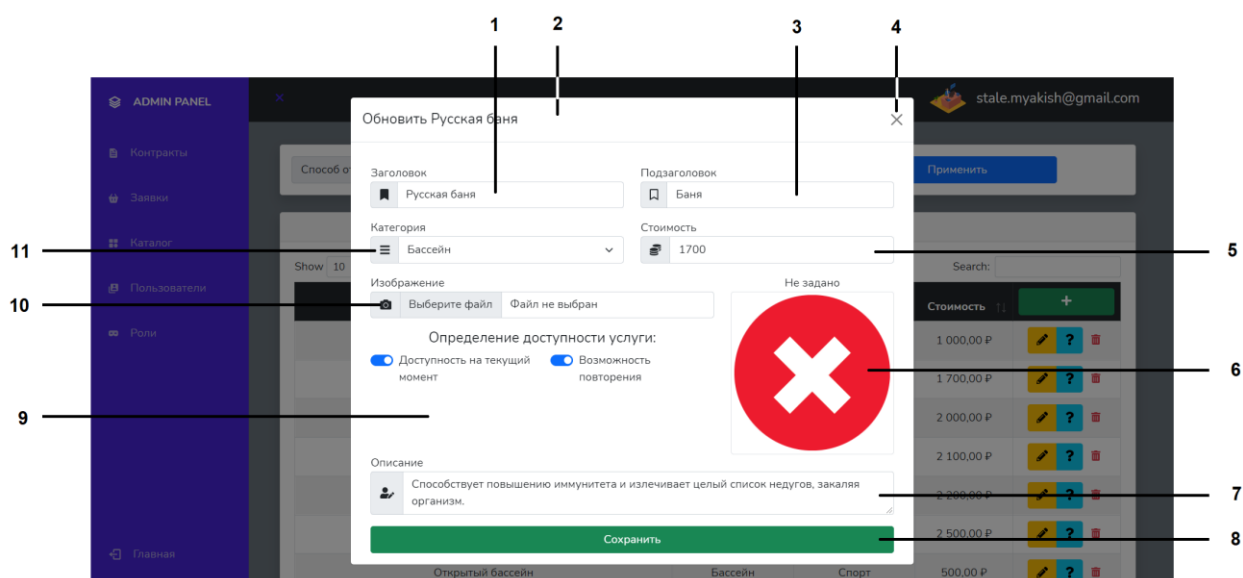


Рисунок 26 – Графический интерфейс модального окна изменения услуги

- 5 – поле для ввода стоимости услуги;
- 6 – область, отображающая установленное изображение для услуги;
- 7 – поле для ввода описания услуги;
- 8 – кнопка отправки данных формы для попытки сохранения изменений, валидация полей;
- 9 – область, содержащая поля установки доступности услуги и возможности ее повторения при оформлении;
- 10 – кнопка для открытия диалогового окна для выбора файла изображения;
- 11 – выпадающий список, содержащие все категории для указания необходимой.

Также рассмотрим основные элементы интерфейса подробной информации об услуге, изображенные на рисунке 27:

- 1 – путь к изображению и его представление;
- 2 – ссылка на информацию о категории (обновление модального окна);
- 3 – заголовок рассматриваемой услуги;

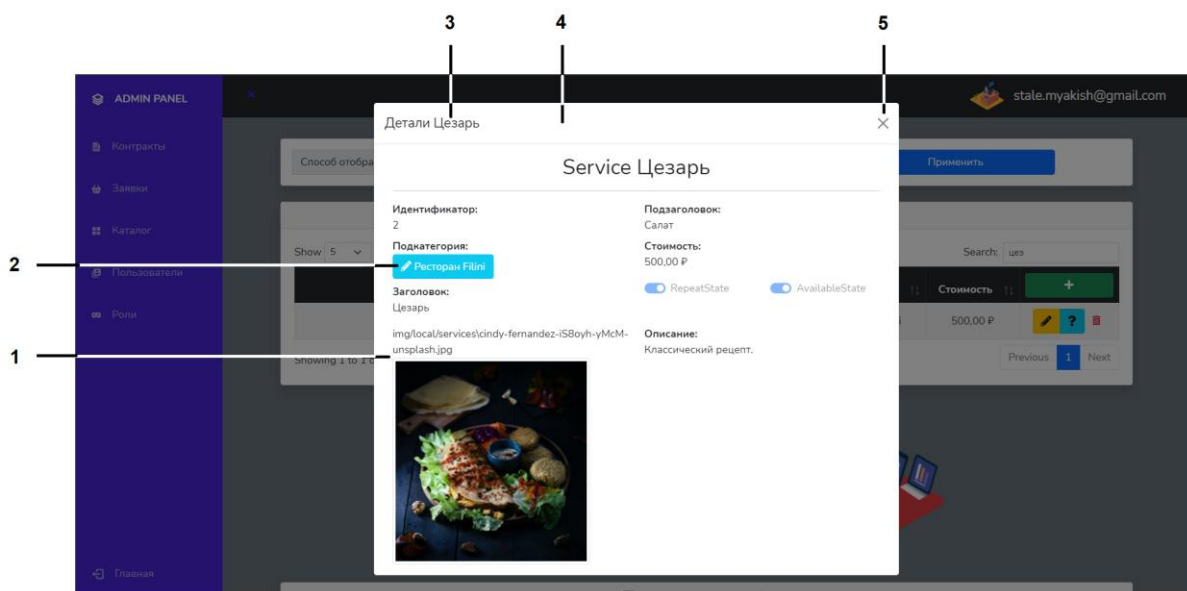


Рисунок 27 – Графический интерфейс модального окна подробной информации об услуге

- 4 – модальное окно;
- 5 – кнопка закрытия модального окна (то же при выборе затемненной области).

Наконец рассмотрим крайний вариант управления данными – удаление записи. На рисунке 28 представлено модальное окно для удаления категории, а именно страница подтверждения:

- 1 – кнопка подтверждения удаления услуги;
- 2 – таблица с краткой информацией об услуге;
- 3 – модальное окно;
- 4 – кнопка закрытия модального окна (то же при выборе затемненной области).

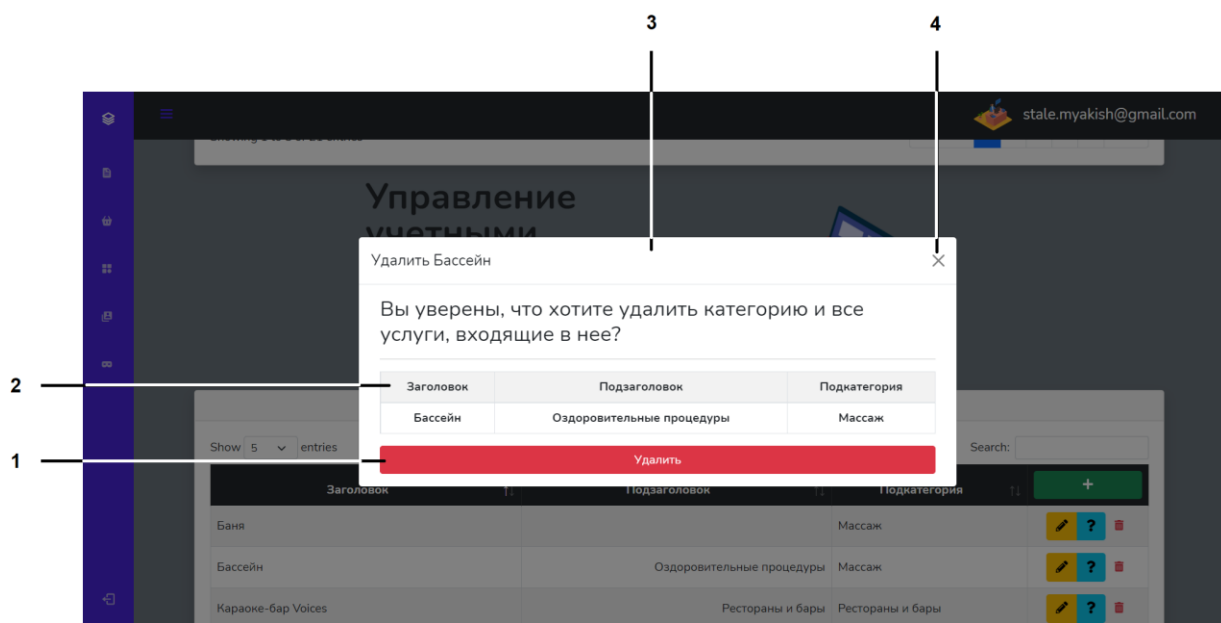


Рисунок 28 – Графический интерфейс модального окна подтверждения удаления услуги

Одним из бизнес-процессов администратора является также управление учетными записями пользователей, если основную информацию можно изменить с помощью личного кабинета, то предоставить права доступа может только администратор. С этой целью был спроектирован интерфейс по работе с ролям информационной системы.

На рисунке 29 изображен интерфейс страницы управления правами доступа. Функционал данной старницы позволяет просматривать пользователей, состоящих в ралличных ролях, а также изменять это состояние.

Кроме того имеется возможность полностью удалить роль, элементами страницы являются:

- 1 – кнопка для сохранения результатов изменения;
- 2 – область, содержащая таблицу с пользователями, которые не является участниками выбранной роли;
- 3 – область, содержащая таблицу с пользователями, которые является участниками выбранной роли;
- 4 – отмена действий.

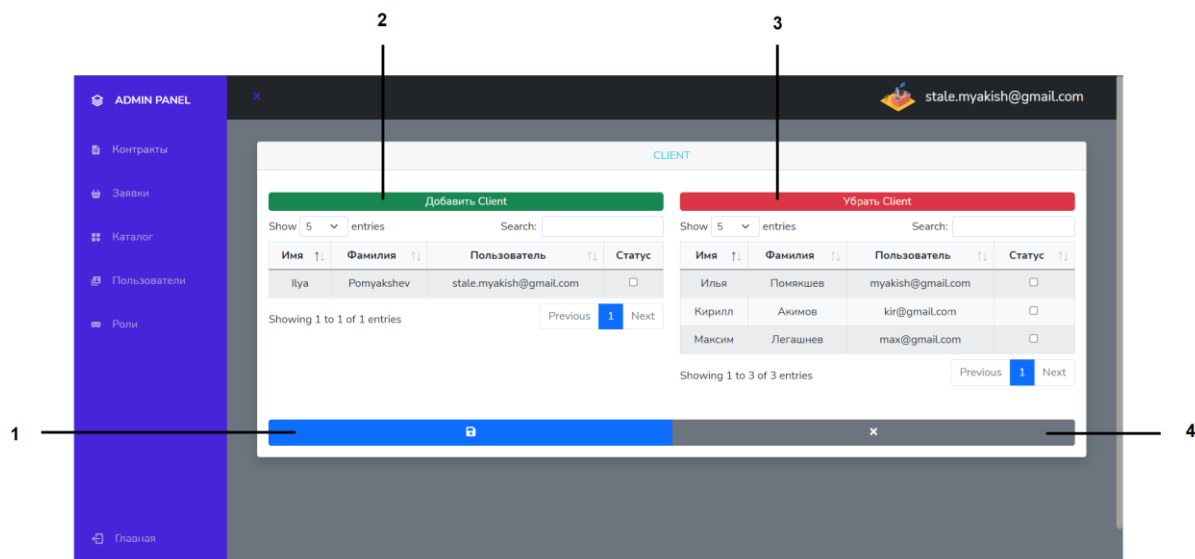


Рисунок 29 – Графический интерфейс страницы управления ролями

На рисунке 30 изображен интерфейс страницы управления заявками, последней рассматриваемой и необходимой для полного функционирования системы. Благодаря этой странице пользователь может как просматривать заказы и их составляющие заявки, так и изменять их статусы. Описание отображенных элементов:

- 1 – список оформленных клиентских заказов;
- 2 – статус заказа («Выполнен» – все включенные заявки оказаны, в противном случае – «Выполняется...»);

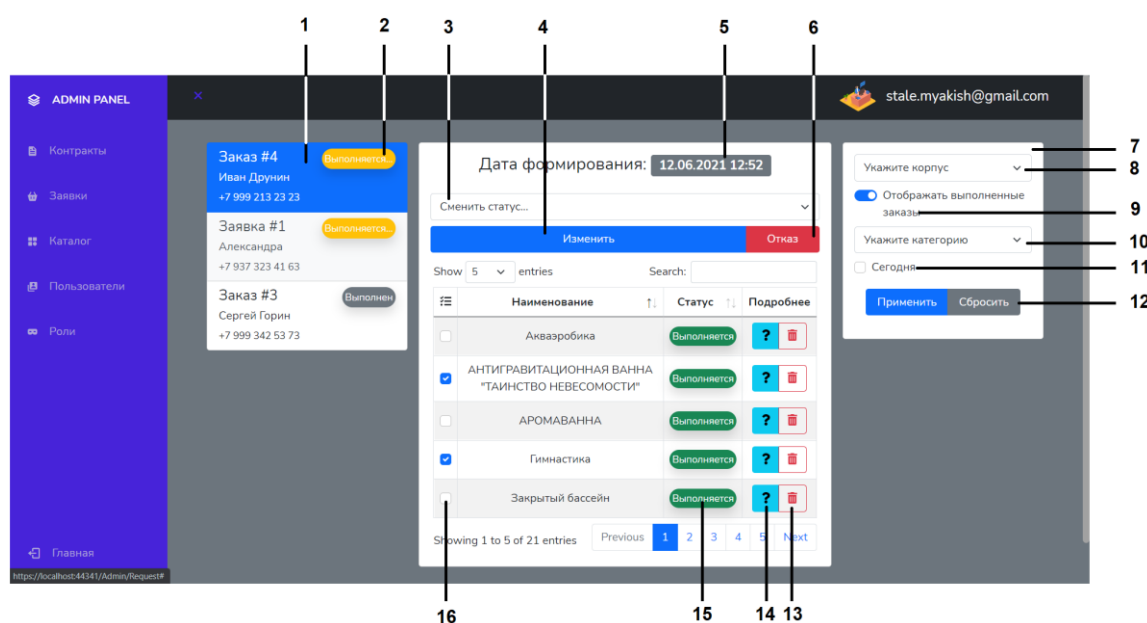


Рисунок 30 – Графический интерфейс страницы заявок

- 3 – выпадающий список возможных статусов заявок;
- 4 – кнопка сохранения изменений статусов у заявок;
- 6 – моментальный статус «Отказ» для заявки;
- 7 – панель фильтрации данных;
- 8 – выпадающий список корпусов ГК, фильтрации по корпусу;
- 9 – вариант полного отображения заказов;
- 10 – выпадающий список категорий, фильтрация по категории;
- 11 – вариант просмотра заказов за текущий день;
- 12 – кнопки применения и сброса фильтров;
- 13 – статус «Отказ» для выбранной заявки;
- 14 – подробная информация о выбранной заявке;
- 15 – текущий статус заявки;
- 16 – поле для множественного выбора заявок, для одновременного изменения статусов нескольких.

На этом описание основного функционала подсистем клиента и администратора закончен. Были описаны схожие элементы интерфейса для CRUD операций с записями в БД, основные возможности спроектированной системы средствами интерфейса. Разработка велась с применением фреймворка Bootstrap 5.0.

3 Реализация и тестирование

3.1 Физическая модель базы данных

На основе логической модели БД была построена физическая модель, изображенная на рисунке 31 и содержащая: полную атрибутивную физическую модель БД с учетом возможностей целевой СУБД Microsoft SQL Server 2019 в нотации IDEF1X, построенную в среде ERwin Data Modeler с указанием отношений. Спецификации сущностей (описание атрибутов для каждой сущности) приведены в приложении Е.

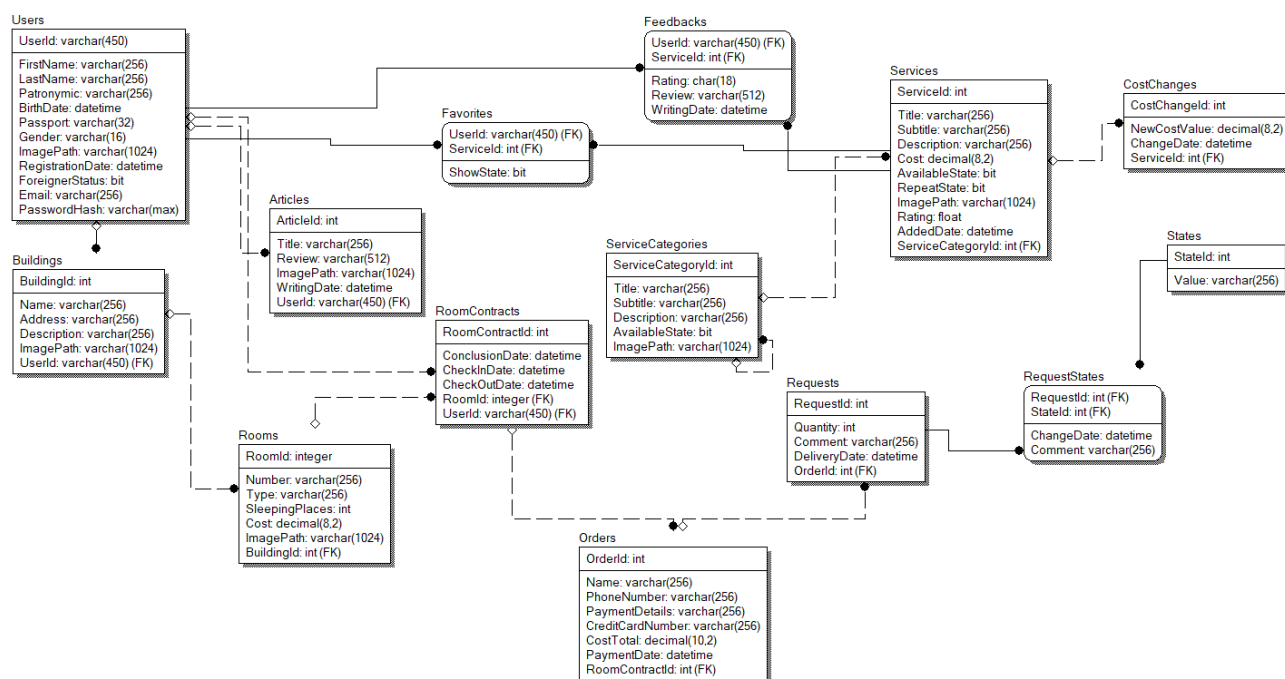


Рисунок 31 – Физическая модель базы данных

При переходе от логической модели БД к физической типы сущностей превращаются в таблицы БД, атрибуты – в столбцы физической БД, уникальные идентификаторы – в первичные ключи (Primary Keys – PK), которые не могут содержать NULL значений, а отношения между таблицами представлены с помощью внешних ключей (Foreign Keys – FK). Следующим шагом после построения физической модели БД в Erwin Data Modeler является генерация кода создания самой БД.

Перед началом проектирования и разработки классов, была дополнена имеющаяся БД шаблонными сущностями Core Identity. Созданная база была спроецирована на объекты С# с помощью Entity Framework Core 5.0 – данный метод разработки клиент-серверного приложения именуется DataBaseFirst, создание моделей из имеющейся БД. После добавления сгенерированных классов и контекста данных был использован обратный принцип – CodeFirst, создание БД из спроектированных моделей.

Первоначально изменив класс User, т.к. для использования Core Identity в рамках разработки, нужно унаследовать базовые классы, необходимые для работы данного API. Данная система позволяет пользователям создавать учетные записи, аутентифицироваться, управлять учетными записями или использовать для входа на сайт учетные записи внешних провайдеров, таких как Facebook, Google, Microsoft, Twitter и других [8].

В итоге была сгенерирована новая БД средствами Entity Framework и Core Identity, которая отличается от первоначальной наличием дополнительных таблиц, которые упрощают работу со входом в приложение:

- RoleClaims – соответствует таблице связи ролей и объектов claims;
- UserClaims – таблица, хранящая набор claims, представляет иную модель авторизации по сравнению с ролями;
- Roles – содержит определения ролей;
- UserRoles – таблица, устанавливающая для пользователей определенные роли;
- UserLogins – таблица логинов пользователя.

В итоге была получена БД, схема которой приведена на рисунке 32 в нотации IDEF1X, разработанная с помощью MySQL Workbench 8.0.

Скрипт для создания объектов БД на языке DDL SQL, приведен в приложении В.

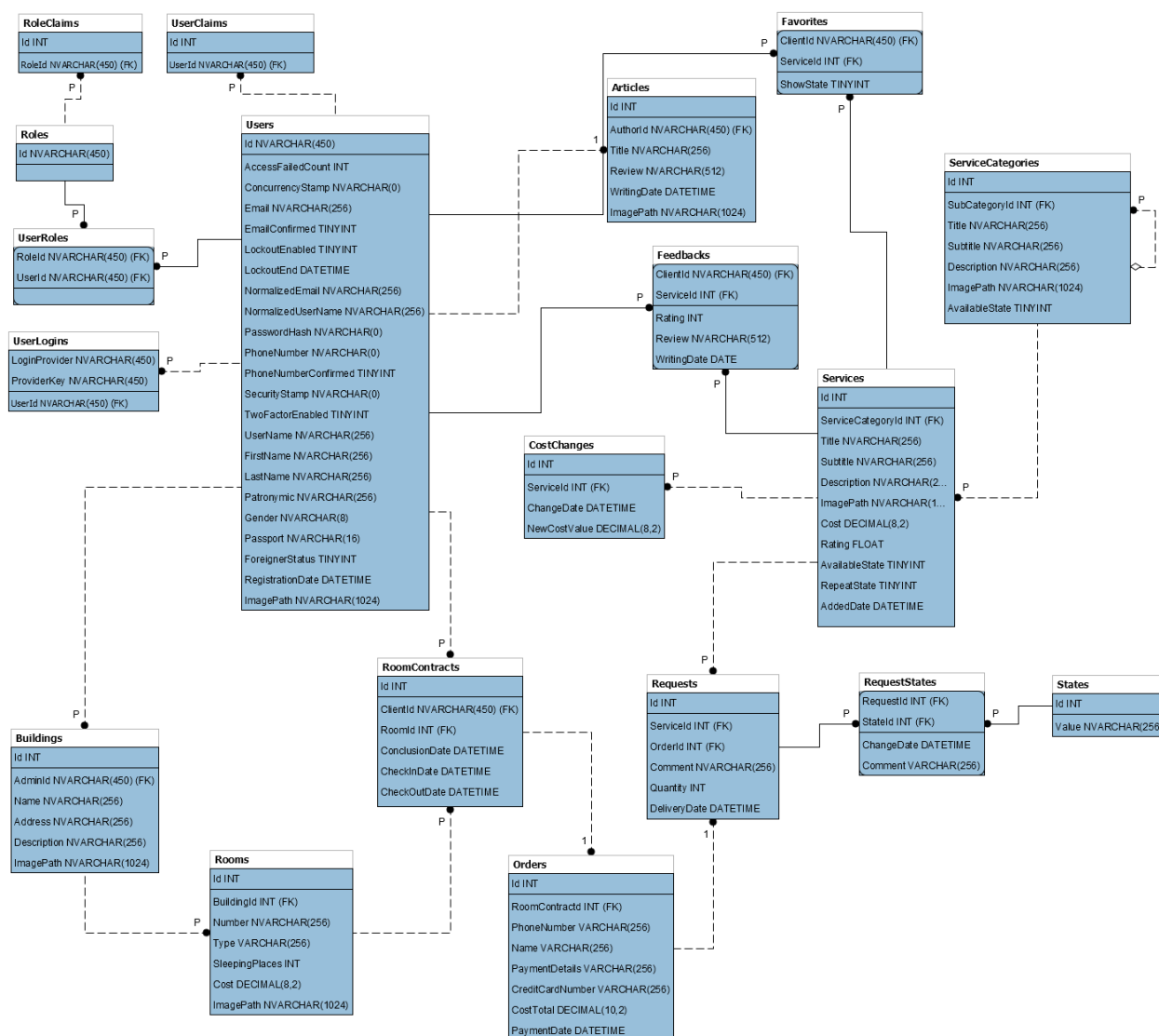


Рисунок 32 – Конечная схема базы данных

Была использована комплексная встроенная система поддержки авторизации и аутентификации, которая позволяет использовать функцию входа в пользовательском интерфейсе для веб-приложений ASP.NET, поэтому при концептуальном и логическом проектировании индивидуальные особенности средств не учитывались, т.к. данная часть работы может быть реализована по-разному (JWT-токены, Cookies, Claims и т.п.), в зависимости от выбранных средств.

Скрипт созданной БД представлен в приложении Б.

3.2 Реализация программных средств

Разработка информационной системы гостиничного комплекса подразделяется на две части: клиентскую и серверную. В роли клиентской части выступает приложение для работы с БД.

В качестве сервера приложений будет разработан REST сервис.

REST веб-сервис является CRUD приложением (Create, Read, Update, Delete). Основой REST приложения является HTTP, так вся работа с REST сервисами происходит через HTTP запросы.

Взаимодействие с RESTfull API происходит через следующие запросы: GET, POST, UPDATE, DELETE.

Особенность REST в том, что сервер не запоминает состояние пользователя между запросами — в каждом запросе передаётся вся информация, необходимая для выполнения операции [14]. Также реализованы представления, триггеры, хранимые процедуры и функции для упрощения дальнейшей разработки серверной части (приложение В).

На рисунке 33 приведены описания контроллеров подсистемы администратора для работы архитектурного паттерна MVC.

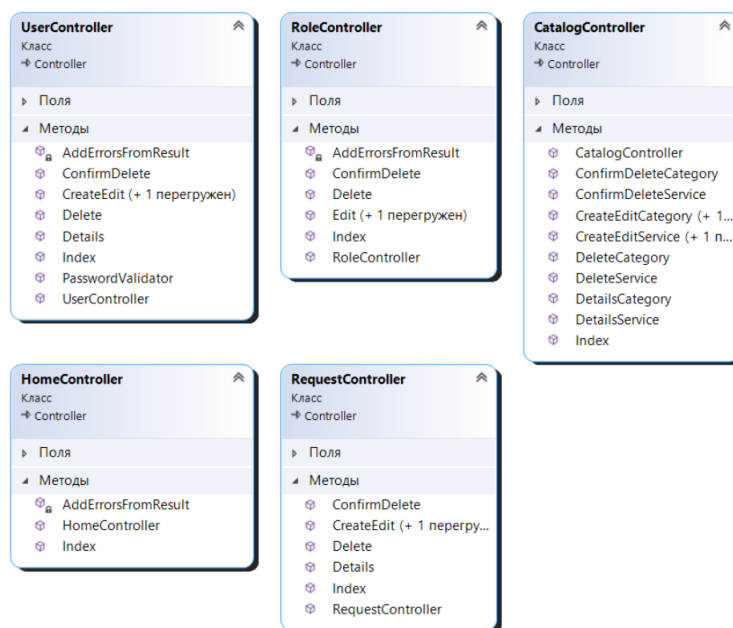


Рисунок 33 – Диаграмма программных классов контроллеров для администратора

Практически каждый метод контроллера обрабатывает один из вышеописанных вариантов запроса RESTfull API, возвращая какой-либо результат, в большинстве случаев определенное представление.

Краткая спецификация спроектированных классов контроллеров приведена в таблице 6.

Таблица 6 – Описание контроллеров подсистемы администратора

№	Класс	Назначение
1.	UserController.cs	Контроллер, обеспечивающий работу с представлениями, связанными с данными о пользователях и обработку запросов от них.
2.	RoleController.cs	Контроллер, обеспечивающий работу с представлениями, связанными с данными о категориях пользователей (ролях) и обработку запросов от них.
3.	CatalogController.cs	Контроллер, обеспечивающий работу с представлениями, связанными с данными об услугах и их категориях и обработку запросов от них.
4.	HomeController.cs	Контроллер, обеспечивающий работу с представлениями на стартовой странице администратора, анализом данных.
5.	RequestController.cs	Контроллер, обеспечивающий работу с представлениями, связанными с данными о заявках и заказах клиентов и обработку запросов от них.

На рисунке 34 отображены основные классы контроллеров для подсистемы клиента.

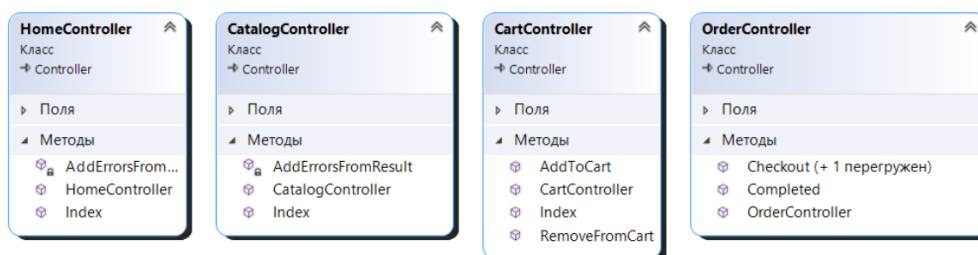


Рисунок 34 – Диаграмма программных классов контроллеров для клиента

Краткая спецификация спроектированных классов контроллеров приведена в таблице 7.

Таблица 7 – Описание контроллеров подсистемы клиента

№	Класс	Назначение
1.	HomeController.cs	Контроллер, обеспечивающий работу с домашней страницей сервисной области клиента, быстрым поиском услуг.
2.	CatalogController.cs	Контроллер, обеспечивающий работу с представлениями, связанными с данными об услугах и их категориях и обработки запросов от них, добавлением в корзину, избранное, комментированием.
3.	CartController.cs	Контроллер, обеспечивающий работу с представлениями, связанными с данными об выбранных услугах для оформления, дальнейшим формированием заказа, корзиной хранимой в сессии.
4.	OrderController.cs	Контроллер, обеспечивающий работу с представлениями, предоставляющими возможность оформления заказа, его оплаты и контроля.

На рисунке 35 отображены основные классы, которые были спроецированы с помощью ORM-системы Entity Framework Core, а также вспомогательные для работы с ними.

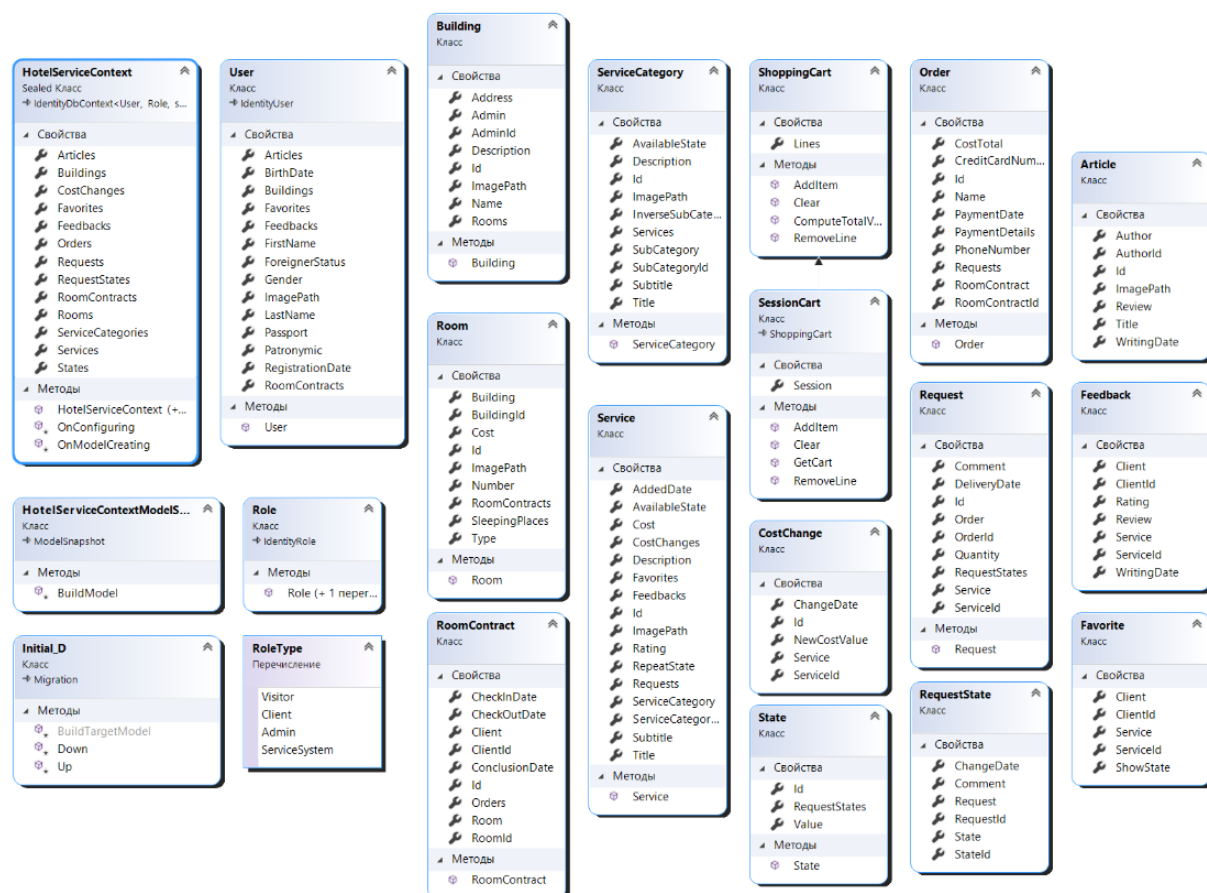


Рисунок 35 – Диаграмма моделей для взаимодействия с БД

Определенные названия классов соответствуют идентичным таблицам в БД, а их свойства соответствующим колонкам и зависимостям, по этой причине ниже в таблице 8 будет приведено описание только ранее нерассмотренных сущностей.

Таблица 8 – Описание вспомогательных моделей для взаимодействия с БД

№	Класс	Назначение
1.	HotelServiceContext.cs	Содержит точку входа в приложение.
2.	Initial_D.cs	Основной файл метаданных миграции, который содержит все применяемые действия и используется Entity Frameworkом.
3.	HotelServiceContextModel Snapshot.cs	Содержит текущее состояние модели, используется при создании следующей миграции.
4.	RoleType.cs	Содержит enum-перечисление для обозначения возможных ролей.
5.	ShoppingCart.cs	Базовый класс, содержит методы для работы корзины клиента.
6.	SessionCart.cs	Класс, унаследованный от ShoppingCart.cs, реализующий поведение в сессии (удаление данных по истечении времени, перезапуске приложения)

На рисунке 36 отображены основные классы, которые отвечают за конфигурацию и развертывание приложения.

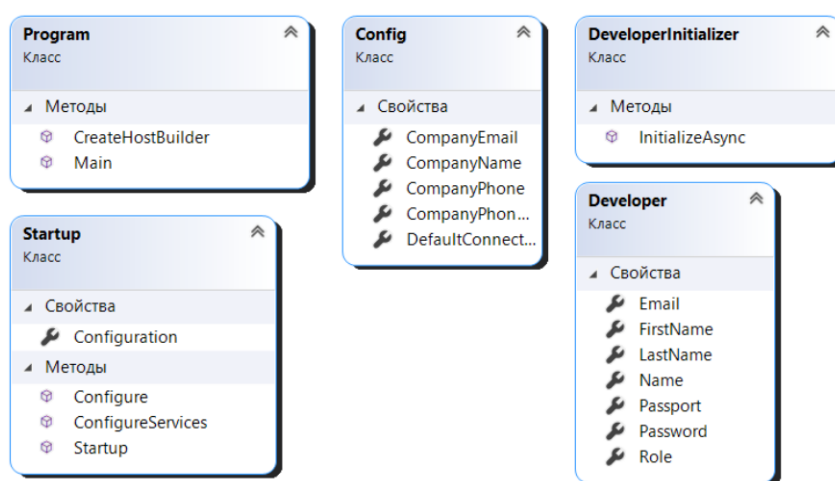


Рисунок 36 – Диаграмма программных классов для конфигурации

Краткое описание классов конфигурации приведено в таблице 9. Program.cs и Startup.cs являются обязательными для инфраструктуры

ASP.NET Core, остальные были созданы дополнительно для упрощения работы.

Таблица 9 – Описание классов конфигурации приложения

№	Класс	Назначение
1.	Program.cs	Содержит точку входа в приложение, главный файл, с которого и начинается его выполнение. Код этого файла настраивает и запускает веб-хост, в рамках которого разворачивается сайт.
2.	Startup.cs	Содержит логику обработки входящих запросов, определяет систему маршрутизации, подключаемые службы, настраивает внедрение зависимостей и т.п.
3.	Config.cs	Статичный класс, хранящий информацию, считанную при запуске из файла .json с пользовательской конфигурацией приложения.
4.	DeveloperInitializer.cs	Содержит метод инициализации, который создает начального пользователя системы, разработчика с полными правами доступа.
5.	Developer.cs	Статичный класс, хранящий информацию, считанную при запуске, определяющую универсального пользователя разработчика, которому доступны все подсистемы.

Для полной информации о физическом представлении ПО системы была спроектирована диаграмма, проецирующая программную архитектуру на аппаратную. Данная диаграмма выполняет роль визуализации компонентов системы и их развертывания на физических узлах (рисунок 37).

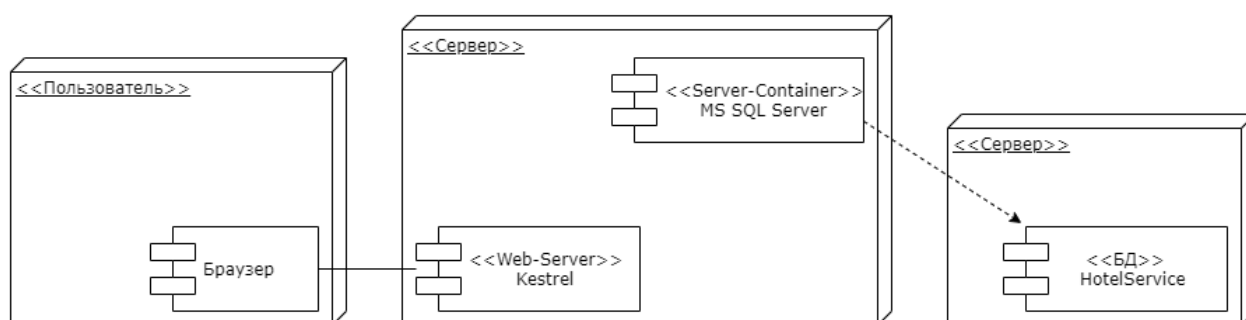


Рисунок 37 – Диаграмма развертывания

Другими словами, диаграмма развертывания показывает топологию и аппаратные средства системы.

3.3 Тестирование приложения

3.3.1 Выбор режима тестирования

Программный продукт обладает набором свойств, которые характеризуют его качество и соответствие требованиям пользователей или заказчика. Исходя из этого задача обеспечения качества разработки программного продукта является одной из самых приоритетных в процессе создания программного обеспечения. Эффективным средством в решении данной задачи является тестирование.

Тестирование ПО производится по двум основным причинам.

1. Выявление ошибок и недочетов в программном коде, из-за которых могут возникнуть критические ситуации, не дающие ПО работать должным образом или прекращающие его работу вовсе.
2. Проверка поведения и удобства интерфейса ПО с учетом внешних воздействий.

С технической точки зрения тестирование заключается в выполнении приложения на некотором множестве исходных данных и сверке получаемых результатов с заранее известными с целью установить соответствие различных свойств и характеристик приложения заказанным свойствам. Для более точного анализа программного продукта применяются различные методологии, которые охватывают разные наборы свойств программного обеспечения.

3.3.2 Функциональное тестирование

Функциональное тестирование представляет собой построение плана тестирования, составленного на основе определенных вариантов использования. Данный вид тестирования базируется на описании тестов и получении на их основе результатов.

При тестировании клиентской части приложения параллельно и происходит тестирование сервера приложений, так как клиент обращается сервисам сервера приложений. Произведем тестирование роли

администратора, как наиболее близкой к осуществлению управления с БД. Функциональные тесты клиентского приложения представлены в таблице 10, результаты тестирования в Приложении Д.

Таблица 10 – Функциональные тесты для клиентской части приложения

Бизнес процесс	Параметры теста	Ожидаемый результат	Результат
Добавление новой услуги	[{"Шаурма", "То, чего так не хватает...", 1, "img", 200, 1, 1, «Знаменитый рецепт Антонио С.Г..»}]	Будет создана новая услуга и добавлена на сайт.	Тест пройден (рисунок Д.1, Д.2)
Контроль учетных записей	Добавление пользователя "myakish" к роли Admin	Пользователь с именем «myakish» добавлен к роли	Тест пройден (рисунок Д.3, Д.4, Д.5)

Таким образом было проведено функциональное тестирование области администратора, которое включало в себя обработку ошибок, добавление новых записей, а также редактирование уже имеющихся, тестирование производилось на таблицах. По результатам тестирования можно убедиться, что приложение работает корректно.

3.3.3 Метрики кода

Метрики кода являются совокупностью оценок ПО, служащей рекомендательными характеристиками и представляющую более полную информацию о программном коде. Применение метрик позволяет руководителям проектов и предприятий изучить сложность разработанного или разрабатываемого проекта, оценить объем работ, стилистику разрабатываемой программы и усилия, потраченные каждым разработчиком для реализации того или иного решения [15].

Комплексный показатель качества кода (Maintainability Index) представляет собой наиболее часто используемый пример метрики. Данная метрика рассчитывается по следующей формуле:

$$MI = \max(0, (171 - 5.2 * \ln(HV) - 0.23 * CC - 16.2 * \ln(LoC)) * 100 / 171),$$

где HV (Halstead Volume) – вычислительная сложность;

CC (Cyclomatic Complexity) – структурная сложность кода, т.е. количество различных ветвей в коде;

LoC (Lines of Code) – количество строк кода.

Значения комплексного показателя качества кода может находиться между 0 и 100. Они показывают относительную сложность поддержки кода: чем больше значение данной метрики, тем легче поддерживать код.

Для определения вычислительной сложности программы сначала следует рассчитать словарь и длину программы, а затем вычислить на их основе HV . Словарь программы рассчитывается по формуле:

$$\eta = \eta_1 + \eta_2,$$

где η_1 – число простых операторов и операций в программе;

η_2 – число простых операндов (констант и переменных) в программе.

Длина программы рассчитывается по следующей формуле:

$$N = N_1 + N_2,$$

где N_1 – общее число простых операторов и операций в программе;

N_2 – общее число простых операндов в программе.

Вычислительная сложность рассчитывается по формуле:

$$HV = N * \log_2 \eta,$$

Структурную сложность программы можно рассчитать по следующей формуле:

$$CC = KOP + КОЦ + 1,$$

где KOP – количество операторов разветвления;

$КОЦ$ – количество операторов цикла.

Значение структурной сложности больше 10 свидетельствует о затруднении ее функционального тестирования (например, методов «белого ящика») и требует упрощение программного кода приложения [15].

Среда разработки Microsoft Visual Studio позволяет автоматически вычислять метрики. Произведем вычисления для разрабатываемого ПО информационной системы сервисного обслуживания ГК. Метрики исходного кода, вычисленные с помощью Microsoft Visual Studio, приведены на рисунке 38.

Иерархия	Индекс удобства п...	Сложность организаци...	Глубина наследования	Взаимозависимость к...	Строки исходного ко...	Строки исполняемого к...
HotelService.Tests (Debug)	86	7	1	13	104	47
HotelService (Debug)	83	1 354	5	314	5 348	3 043
HotelService.Repositories.Interfaces	100	8	0	7	21	0
HotelService.Models.ViewModels.Client	100	8	1	4	16	0
HotelService.Areas.Identity	97	1	1	5	10	1
HotelService.Models.Base	97	248	3	28	361	79
HotelService.Components	94	2	2	4	17	2
HotelService.Models.ViewModels.Admin	93	46	1	16	97	60
HotelService.Controllers	89	8	3	9	39	7
HotelService.Areas.Identity.Pages.Account.Manage	88	44	2	23	130	54
AspNetCore	84	667	4	81	0	1 275
HotelService.Service	83	28	3	28	62	12
HotelService.Areas.Identity.Pages.Account	82	56	2	46	242	94
HotelService.Repositories	80	27	1	28	98	45
HotelService.Infrastructure	79	19	2	26	100	22
HotelService.Models	74	57	5	68	473	254
HotelService.Areas.Client.Controllers	74	20	3	40	163	56
HotelService.Areas.Admin.Controllers	67	104	3	57	693	273
HotelService	64	7	1	76	132	55
HotelService.Migrations	19	4	2	37	2 694	754

Рисунок 38 – Метрики исходного кода проекта

Проанализируем полученные результаты. В данном проекте комплексный показатель кода (MI) находится на высоком уровне, т.к. его среднее значение составляет 86. Это свидетельствует о легкости поддержки исходного кода. Структурная сложность кода (СС) находится на среднем уровне.

Большой глубиной наследования в данном проекте обладают классы-модели, описывающие взаимодействие с БД, а также между представлениями и контроллерами. Остальные классы обладают глубиной наследования не более 4. Это свидетельствует о том, что другим людям, не знакомым с данным проектом, будет просто понять, где определяются или переопределяются конкретные методы и поля класса.

Чем больше показатель взаимозависимости (связности) классов, тем сложнее поддерживать данные классы. По результатам вычисленных метрик (смотри рисунок 38) показатель взаимозависимости относительно невысокий, в рамках использования механизма внедрения зависимостей, а также конкретных классов моделей, проецируемых на таблицы БД.

Сделав анализ метрик исходного кода, можно сделать вывод о довольно качественной разработке информационной системы сервисного обслуживания.

3.3.4 Модульное тестирование

Модульное тестирование (юнит-тестирование) позволяет проверить на корректность каждый модуль исходного кода программного продукта отдельно. Юнит-тесты пишутся на этапе кодирования приложения для избавления от дефектов и ошибок в будущем. Для написания юнит-тестов в данном проекте применялся xUnit.net – фреймворк тестирования для платформы .NET.

Приведем пример тестирования работы основного компонента на стороне клиента ShoppingCart.cs для работы с корзиной с помощью сессий.

Класс ShoppingCart относительно прост, но в нем присутствует ряд важных линий поведения, которые должны корректно работать. Неправильно функционирующая корзина нарушит работу всего приложения.

Для размещения тестов в проекте HotelService.Tests создан новый файл по имени CartTests.cs.

Код данного файла представлен в листинге 1.

В файле ShoppingCart.cs описаны следующие методы:

- метод AddItem;
- метод RemoveLine;
- метод ComputeTotalValue;
- метод Clear.

Метод AddItem(Base.Service service, int quantity) добавляет выбранный элемент к корзине, если элемент первый, то создает корзину. Принимает на вход: экземпляр выбранной услуги и количество повторений. Возвращает void.

RemoveLine(Base.Service product) удаляет выбранную услугу из корзины. Принимает на вход: экземпляр таймера услуги. Возвращает void.

Метод `ComputeTotalValue()` подсчитывает общую стоимость элементов в корзине с учетом повторений. Возвращает `void`.

Метод `Clear ()` удаляет корзину. Возвращает `void`.

Листинг 1 – Код файла `ShoppingCart.cs`

```
public class ShoppingCart
{
    public List<Request> Lines { get; set; } = new List<Request>();
    public virtual void AddItem(Base.Service service, int quantity)
    {
        var Line = Lines.FirstOrDefault(p => p.Service.Id == service.Id);
        if (Line == null)
        {
            Lines.Add(new Request
            {
                Service = service,
                Quantity = quantity
            });
        }
        else
        {
            Line.Quantity += quantity;
        }
    }
    public virtual void RemoveLine(Base.Service product) =>
        Lines.RemoveAll(l => l.Service.Id == product.Id);
    public virtual decimal ComputeTotalValue() => Lines.Sum(e =>
        e.Service.Cost * e.Quantity);
    public virtual void Clear() => Lines.Clear();
}
```

Тесты в `xUnit` определяются в виде методов, к которым применяются атрибуты `Fact` и `Theory`. Атрибут `Fact` указывает, что методы представляет тест.

Каждый метод содержит три логических части - `Arrange`, `Act` и `Assert`. Они помечены соответствующими комментариями.

Секции `Arrange` и `Act` представляют обычный код на языке `C#`. А секция `Assert` использует одноименный класс `Assert`, который определен в библиотеке `xUnit.net`.

Код модульного теста для проверки работы таймера показан в листинге Д.1 приложение Д.

Метод `Can_Add_New_Lines ()` создает две услуги, добавляет их в корзину. После в области сравнения проверяет на соответствие данных по количеству и соответствие экземпляров.

Метод `Can_Add_Quantity_For_Existing_Lines()` создает две услуги, добавляет в корзину разное количество их повторений. Далее идет проверка на соответствие добавленных и действительных данных.

Метод `Can_Remove_Line ()` создает 3 услуги, добавляет в корзину разное количество их повторений. После удаляется строка корзины (все повторения услуги) и проводится проверка на количество строк в корзине.

Метод `Calculate_Cart_Total ()` создает 2 услуги, добавляет в корзину разное их количество повторений. После проверяет итоговую стоимость на основе добавленных данных сравнивая с ожидаемой.

Метод `Can_Clear_Contents()` создает 2 услуги, добавляет их в корзину. После очищает ее и проверяет на наличие строк.

В результате выполнения юнит-тестирования, описанного выше, можно сделать вывод, что методы файла `ShoppingCart.cs` работают корректно.

Также составим таблицу, в которой опишем каждый тест и полученный результат. Описания юнит-тестов показаны в таблице 11.

Таблица 11 – Описание юнит-тестов

№	Название тестового метода	Описание теста	Результат
1	<code>Can_Add_New_Lines</code>	Тестирование создания корзины и добавления в нее услуг	Успешно
2	<code>Can_Add_Quantity_For_Existing_Lines</code>	Тестирование добавления услуг и обновления счетчика их повторений	Успешно
3	<code>Can_Remove_Line</code>	Тестирование удаления строк в корзине	Успешно
4	<code>Calculate_Cart_Total</code>	Тестирование подсчета общей стоимости всех строк в корзине с учетом повторений	Успешно
5	<code>Can_Clear_Contents</code>	Тестирование удаления корзины и ее очистки	Успешно

На рисунке 39 отображены результаты выполнения тестирования средствами IDE.

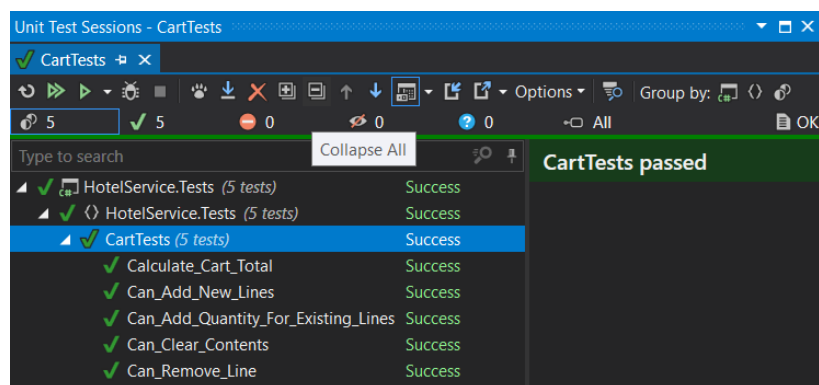


Рисунок 39 – Результаты тестирования работы корзины с помощью xUnit

Все юнит-тесты были успешно пройдены. Это означает, что компонент «Корзина» работает корректно.

4 Планирование и оценка бюджета проекта

Для достижения положительного результата в проектном управлении необходимо рационального применения своих знаний и инструментальных средств, а также наличие заранее сформулированного конкретного плана [13].

В таблице 12 представлено описание затрачиваемых ресурсов.

Таблица 12 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Руководитель	Трудовой	200,00 Р/ч	300,00 Р/ч
ASP.Net Разработчик	Трудовой	250,00 Р/ч	400,00 Р/ч
Дизайнер	Трудовой	150,00 Р/ч	230,00 Р/ч
Офис	Затраты		
Интернет	Затраты	500 Р	
ПЭВМ на ОС Win10	Материальный	40 000,00 Р	

Из представленной таблицы видно, что реализацией проекта будут заниматься три программиста и один дизайнер, которым для работы нужны компьютеры и доступ к интернету.

4.1 Календарное планирование и оценка бюджета проекта

Задача календарного планирования является эффективным инструментом управления проектами фирмы. Календарный план даёт четкое представление об этапах производства, длительности этапов, количестве и видах привлекаемых ресурсов на каждом этапе.

Проведем необходимые вычисления с использованием выбранного средства управления программными проектами Microsoft Project 2016. На рисунке 40 представлен список задач, было выделено 24 задачи. Началом проекта была принята дата 14.04.2021.

	i	Название задачи	Длитель	Начало	Окончание	Пре	Названия ресурсов
1		Начало проекта	0 дней	Ср 14.04.21	Ср 14.04.21		Руководитель
2		Анализ требований	6 дней	Пн 26.04.21	Пн 03.05.21		
3		Анализ приложений аналогов	2 дней	Пн 26.04.21	Вт 27.04.21	1	Руководитель;ASP.Net Разработчик;Интернет
4		Анализ рынка гостиничных услуг	1 день	Ср 28.04.21	Ср 28.04.21	3	Руководитель;Интернет
5		Консультация с поставщиками и потребителями гостиничных услуг	2 дней	Чт 29.04.21	Пт 30.04.21	4	Руководитель;Офис
6		Принятие окончательного решения	1 день	Пн 03.05.21	Пн 03.05.21	5	Руководитель
7		Анализ требований завершен	0 дней	Пн 03.05.21	Пн 03.05.21	6	
8		Выбор средств реализации проекта	4 дней	Ср 14.04.21	Пн 19.04.21		
9		Анализ возможных программных решений	2 дней	Ср 14.04.21	Пт 16.04.21	1	ASP.Net Разработчик[200%];Руководитель;ПЭВМ на ОС Win10[3 Единиц];Интернет;Офис
10		Выбор программного решения для реализации приложения	1 день	Пн 19.04.21	Пн 19.04.21	9	ASP.Net Разработчик[200%];Руководитель
11		Выбор программного решения завершен	0 дней	Пн 19.04.21	Пн 19.04.21	10	
12		Проектирование приложения	5 дней	Вт 04.05.21	Пн 10.05.21		
13		Проектирование структур данных	5 дней	Вт 04.05.21	Пн 10.05.21	7	Офис[100 000,00 Р];Интернет[8 000,00 Р];ASP.Net Разработчик[200%]
14		Проектирование архитектуры приложения	4 дней	Вт 04.05.21	Пт 07.05.21	7;11	ASP.Net Разработчик;Интернет
15		Проектирование пользовательского интерфейса	2 дней	Вт 04.05.21	Ср 05.05.21	7	Дизайнер;Интернет
16		Реализация приложения	22 дней	Вт 11.05.21	Ср 09.06.21		
17		Реализация структур данных	5 дней	Вт 11.05.21	Пн 17.05.21	13	ASP.Net Разработчик[200%]
18		Реализация запросов к серверу	10 дней	Вт 18.05.21	Пн 31.05.21	14;17	ASP.Net Разработчик[200%]
19		Реализация пользовательского интерфейса	7 дней	Вт 01.06.21	Ср 09.06.21	15;18	ASP.Net Разработчик[200%];Дизайнер[50%]
20		Реализация проекта завершена	0 дней	Ср 09.06.21	Ср 09.06.21	19	
21		Тестирование приложения	10 дней	Вт 01.06.21	Пн 14.06.21		
22		Тестирование запросов к серверу	1 день	Вт 01.06.21	Вт 01.06.21	18	ASP.Net Разработчик[200%]
23		Тестирование пользовательского интерфейса	1 день	Чт 10.06.21	Чт 10.06.21	19	ASP.Net Разработчик;Дизайнер
24		Исправление возникших ошибок	2 дней	Пт 11.06.21	Пн 14.06.21	22;23	ASP.Net Разработчик[200%];Интернет
25		Тестирование завершено	0 дней	Пн 14.06.21	Пн 14.06.21	24	

Рисунок 40 – Задачи проекта в среде Microsoft Project

Основными задачами будут являться анализ требований, выбор средств реализации проекта, проектирование приложения, реализация приложения и тестирование приложения.

На рисунке 41 представлена диаграмма Ганта, построенная в среде Microsoft Project на основе введенных данных.

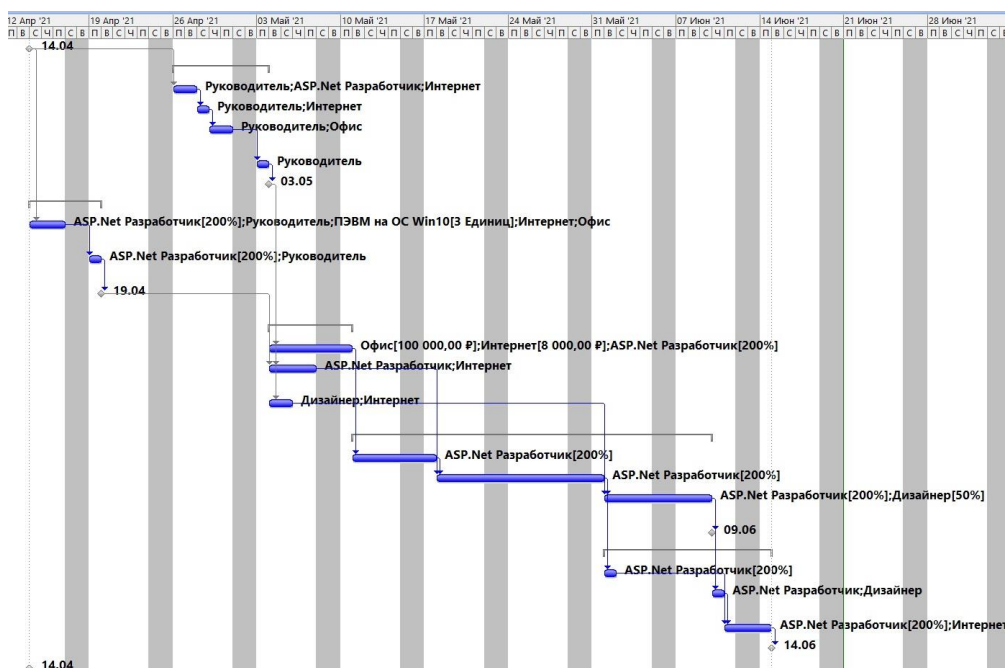


Рисунок 41 – Диаграмма Ганта в среде Microsoft Project

Также с помощью средств среды был получен обзорный отчет проекта планирования процесса разработки информационной системы, на рисунке 42 представлен обзор проекта.

Даты			
Начало:	Ср 14.04.21	Окончание:	Пн 14.06.21
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	44 дней	Оставшаяся:	44 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	44 дней	Процент завершени	0%
Трудозатраты			
Текущий план:	628 ч	Оставшиеся:	628 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	628 ч	Процент завершени	0%
Затраты			
Текущий план:	376 600,00 Р	Оставшиеся:	376 600,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	376 600,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи:	26	Трудовые ресурсы:	5
Выполняющиеся задачи:	0	Трудовые ресурсы с прев. дост	0
Завершенные задачи:	0	Материальные ресурсы:	1
Итого задач:	26	Итого ресурсов:	6

Рисунок 42 – Обзорный отчет проекта

Из рисунка видно, что общая длительность проекта составляет 44 дня, а предполагаемые затраты – 376600 Р.

Заключение

В результате выпускной квалификационной работы разработано программное обеспечение информационной системы сервисного обслуживания, состоящее из клиентского приложения и серверной части. В основе работы данной системы лежит REST API сервис для работы с БД.

Клиентская часть работы была реализована с помощью языка разметки HTML, фреймворка Bootstrap 5, а также дополнительных стилей и веб-сценариев в интеграционной системе разработки Microsoft Visual Studio 2019. В серверной части ПО разработка велась на языке C# с использованием ASP.NET Core MVC, а также с помощью Entity Framework Core и СУБД Microsoft SQL Server 2019 была реализована БД, включающая в себя таблицы, представления, хранимые процедуры и функции, а также триггеры.

В процессе выполнения работы проведено функциональное тестирование и модульное тестирование основных элементов. На основе анализа результатов тестирования можно сделать вывод о корректности и надежности ПО информационной системы.

На этапе анализа качества кода был рассмотрен комплексный показатель качества кода, находящийся на достаточно высоком уровне и лишь в некоторых классах опускается ниже 80 %, что свидетельствует об относительной простоте поддержки исходного кода.

В ходе работы произведено планирование и оценка бюджета разработки проекта с помощью программного продукта Microsoft Project 2016, в процессе которого были выяснены длительность разработки проекта, равная 44 дням, и бюджет, равный 376600 рублям. Результат оценки бюджета и длительности разработки проекта свидетельствует о его экономической выгоды.

Таким образом, в результате выполнения выпускной квалификационной работы была разработана информационная система сервисного обслуживания гостиничного комплекса

Список использованных источников

1. Гостиничное дело: учебник / под ред. д. иск., проф. Н.М. Мышьяковой, к.культ., доц. С.Г. Шкуропат. – СПб.: Изд-во СПбГУСЭ, 2013. – 314 с.
2. Журнал Hotelier.PRO - издание для владельцев отелей [Электронный ресурс]. – URL: <https://hotelier.pro/item/all/> (дата обращения: 25.05.2021).
3. Управление гостиничным предприятием: учебное пособие / Р.Я.Вакуленко, Е.А.Кочкурова. – М.: Университетская книга; Логос, 2008 – 320 с. – (Новая университетская библиотека)
4. TravelLine – Каким должен быть контент на гостиничном сайте. Часть 1. [Электронный ресурс]. – URL: <https://www.travelline.ru/blog/kakim-dolzhen-byt-kontent-na-gostinichnom-sayte/> (дата обращения: 25.05.2021).
5. Правила предоставления гостиничных услуг в Российской Федерации. Утверждены Постановлением Правительства Российской Федерации 25 апреля 1997 г. N 490.
6. Арлоу, Д. UML 2 и Унифицированный процесс. / Арлоу Д., Нейштадт А. – СПб.: Символ-Плюс, 2007.
7. Адам Фримен, ASP.NET Core MVC 2 с примерами на С# для профессионалов, 7-е издание: Пер. с англ. – СПб.: ООО «Диалектика», 2019. – 1008 с.
8. Документация по ASP.NET [Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-5.0> (дата обращения 12.10.2020), свободный.
9. Хомоненко, А.Д. Базы данных. Учебник для вузов / А.Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев. – ДМК. : Пресс, 2013. – 232 с.
10. Руководство по MS SQL Server 2019 [Электронный ресурс] – Режим доступа: <https://metanit.com/sql/sqlserver/> (дата обращения 17.12.2020), свободный.

11. Руководство по Entity Framework Core [Электронный ресурс] – Режим доступа: <https://metanit.com/sql/sqlserver/> (дата обращения 18.12.2020), свободный.
12. Документация по Bootstrap [Электронный ресурс] – Режим доступа: <https://getbootstrap.com/docs/5.0/getting-started/introduction/> (дата обращения 05.11.2020), свободный.
13. Куперштейн, В. И. Microsoft Project в управлении проектами. / В. И. Куперштейн, А.В. Цветкова. – СПб.: БХВ-Петербург, 2011. – 415 с.
14. Архитектура REST [Электронный ресурс] – Режим доступа: <https://habr.com/ru/post/38730/> (дата обращения 10.11.2020), свободный.
15. Тестирование программного обеспечения. Фундаментальные концепции менеджмент бизнес-приложений: Пер. с англ./ Сэм, Джек Фолк, Енг Кек Нгуен. – К.: Издательство «ДиаСофт», 2001. – 544 с.

Приложение А
ГЛОССАРИЙ
(справочное)

База данных (БД) – организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей.

Система управления базами данных (СУБД) – совокупность языковых и программных средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных одним или несколькими пользователями.

Объектно-ориентированное программирование (ООП) – это методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

API (англ. Application Programming Interface – программный интерфейс приложения) – это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

Веб-приложение – клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера.

Нормальная форма (НФ) – свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных.

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

Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Приложение Б
ЛИСТИНГ СКРИПТА СОЗДАНИЯ ТАБЛИЦ
БАЗЫ ДАННЫХ НА ЯЗЫКЕ SQL
(рекомендуемое)

```

--CREATE DATABASE HotelService;

USE HotelService;

-- -----
-- Table [HotelService].[dbo].[Users]
-- -----

DROP TABLE IF EXISTS [dbo].[Users];

CREATE TABLE [dbo].[Users]
(
    [Id] NVARCHAR(450) NOT NULL,
    [FirstName] NVARCHAR(256) NOT NULL,
    [LastName] NVARCHAR(256) NOT NULL,
    [Patronymic] NVARCHAR(256) DEFAULT NULL,
    [Gender] NVARCHAR(16) DEFAULT NULL,
    [Passport] NVARCHAR(32) NOT NULL UNIQUE,
    [ForeignerStatus] BIT DEFAULT 0 NULL,
    [RegistrationDate] DATETIME2 DEFAULT getdate() NOT
NULL,
    [ImagePath] NVARCHAR(1024) DEFAULT NULL
)

GO

-- -----
-- Table [HotelService].[dbo].[Buildings]
-- -----

DROP TABLE IF EXISTS [dbo].[Buildings] ;

CREATE TABLE [dbo].[Buildings] (
    [Id] INT NOT NULL IDENTITY,
    [AdminId] NVARCHAR(450) DEFAULT NULL,
    [Name] NVARCHAR(256) NOT NULL,
    [Address] NVARCHAR(256) NOT NULL,
    [Description] NVARCHAR(256) NULL,
    [ImagePath] NVARCHAR(1024) DEFAULT NULL,

    FOREIGN KEY ([AdminId]) REFERENCE
S [dbo].[Users]([Id]) ON DELETE SET NULL,
    PRIMARY KEY ([Id])
)

GO

-- -----
-- Table [HotelService].[dbo].[Rooms]
-- -----

DROP TABLE IF EXISTS [dbo].[Rooms];

CREATE TABLE [dbo].[Rooms] (
    [Id] INT NOT NULL IDENTITY,
    [BuildingId] INT NOT NULL,
    [Number] NVARCHAR(256) NOT NULL,
    [Type] NVARCHAR(256) DEFAULT 'STD' NOT NUL
L,
    CONSTRAINT CK_Type CH
ECK([Type] = 'STD' OR [Type] = 'Family Room' OR [Type] = 'Suite'
OR [Type] = 'Royal Su
ite' OR [Type] = 'Presidential Suite' OR [Type] = 'Honeymoon Suite'),

```

```

        [SleepingPlaces]                INT                DEFAULT 1 NOT NULL,
        [Cost]                          DECIMAL(8,2)         DEFAULT 5000.00 NOT N
ULL,
        [ImagePath]                     NVARCHAR(1024)        DEFAULT NULL,
                                           FOREIGN KEY          ([BuildingId]) REFERE
NCES [dbo].[Buildings]([Id]) ON DELETE CASCADE,
                                           PRIMARY KEY          ([Id])
    )
GO
-- -----
-- Table [HotelService].[dbo].[ServiceCategories]
-- -----
DROP TABLE IF EXISTS [dbo].[ServiceCategories];

CREATE TABLE [ServiceCategories] (
    [Id]                                INT                NOT NULL IDENTITY,
    [SubCategoryId]                     INT                DEFAULT NULL,
    [Title]                             NVARCHAR(256)       NOT NULL UNIQUE,
    [Subtitle]                          NVARCHAR(256)       DEFAULT NULL,
    [Description]                       NVARCHAR(512)       DEFAULT NULL,
    [ImagePath]                         NVARCHAR(1024)      DEFAULT NULL,
    [AvailableState]                    BIT                DEFAULT 1 NOT NULL,
                                           FOREIGN KEY          ([SystemEmployeeId])
REFERENCES [dbo].[Users]([Id]) ON DELETE SET NULL,
                                           FOREIGN KEY          ([SubCategoryId]) REF
ERENCES [dbo].[ServiceCategories]([Id]) ON DELETE NO ACTION,
                                           PRIMARY KEY          ([Id])
    )
GO
-- -----
-- Table [HotelService].[dbo].[Services]
-- -----
DROP TABLE IF EXISTS [Services];

CREATE TABLE [dbo].[Services] (
    [Id]                                INT                NOT NULL IDENTITY,
    [ServiceCategoryId]                 INT                NOT NULL,
    [Title]                             NVARCHAR(256)       NOT NULL,
    [Subtitle]                          NVARCHAR(256)       NOT NULL,
    [Description]                       NVARCHAR(512)       DEFAULT NULL,
    [ImagePath]                         NVARCHAR(1024)      DEFAULT NULL,
    [Cost]                              DECIMAL(8,2)         DEFAULT 1000.00 NOT N
ULL,
    [Rating]                            FLOAT                DEFAULT NULL
    CONSTRAINT CK_Service
s_Rating CHECK([Rating] >= 0 or [Rating] <= 10),
    [AvailableState]                    BIT                DEFAULT 1 NOT NULL,
    [RepeatState]                       BIT                DEFAULT 1 NOT NULL,
    [AddedDate]                         DATE                DEFAULT getdate() NOT
NULL,
                                           FOREIGN KEY          ([ServiceCategoryId])
REFERENCES [dbo].[ServiceCategories]([Id]) ON DELETE CASCADE,
                                           PRIMARY KEY          ([Id])
    )
GO
-- -----
-- Table [HotelService].[dbo].[Feedbacks]

```

```

-----
DROP TABLE IF EXISTS [dbo].[Feedbacks];

CREATE TABLE [dbo].[Feedback] (
    [ClientId] NVARCHAR(450) NOT NULL,
    [ServiceId] INT NOT NULL,
    [Rating] INT DEFAULT 5 NOT NULL
    CONSTRAINT CK_Feedbac
k_Rating CHECK([Rating] >= 0 or [Rating] <= 5),
    [Review] NVARCHAR(512) DEFAULT NULL,
    [WritingDate] DATETIME2 DEFAULT getdate() NOT
NULL,
    FOREIGN KEY ([ClientId]) REFERENC
ES [dbo].[Users]([Id]) ON DELETE CASCADE,
    FOREIGN KEY ([ServiceId]) REFEREN
CES [dbo].[Services]([Id]) ON DELETE CASCADE,
    PRIMARY KEY ([ClientId], [Service
Id])
)

GO
-----
-- Table [HotelService].[dbo].[RoomContracts]
-----
DROP TABLE IF EXISTS [dbo].[RoomContracts];

CREATE TABLE [dbo].[RoomContracts] (
    [Id] INT NOT NULL IDENTITY,
    [ClientId] NVARCHAR(450) NOT NULL,
    [RoomId] INT NOT NULL,
    [ConclusionDate] DATETIME2 DEFAULT getdate() NOT
NULL,
    [CheckInDate] DATETIME2 DEFAULT getdate() NOT
NULL,
    [CheckOutDate] DATETIME2 NULL,
    FOREIGN KEY ([ClientId]) REFERENC
ES [dbo].[Users]([Id]) ON DELETE CASCADE,
    FOREIGN KEY ([RoomId]) REFERENCES
[dbo].[Rooms]([Id]) ON DELETE CASCADE,
    PRIMARY KEY ([Id])
)

GO
-----
-- Table [HotelService].[dbo].[Articles]
-----
DROP TABLE IF EXISTS [dbo].[Articles];

CREATE TABLE [dbo].[Articles] (
    [Id] INT NOT NULL IDENTITY,
    [AuthorId] NVARCHAR(450) NOT NULL,
    [Title] NVARCHAR(256) NOT NULL,
    [Review] NVARCHAR(512) NOT NULL,
    [ImagePath] NVARCHAR(1024) DEFAULT NULL,
    [WritingDate] DATETIME2 DEFAULT getdate() NOT
NULL,
    FOREIGN KEY ([AuthorId]) REFERENC
ES [dbo].[Users]([Id]) ON DELETE CASCADE,
    PRIMARY KEY ([Id])
)

```

```

GO
-- -----
-- Table [HotelService].[dbo].[CostChanges]
-- -----
DROP TABLE IF EXISTS [dbo].[CostChanges];

CREATE TABLE [dbo].[CostChanges] (
    [Id] INT NOT NULL IDENTITY,
    [ServiceId] INT NOT NULL,
    [ChangeDate] DATETIME2 DEFAULT getdate() NOT
NULL,
    [NewPriceValue] DECIMAL(8,2) NOT NULL,
    FOREIGN KEY ([ServiceId]) REFEREN
CES [dbo].[Services]([Id]) ON DELETE CASCADE,

    PRIMARY KEY ([Id])
)

GO
-- -----
-- Table [HotelService].[dbo].[Favorites]
-- -----
DROP TABLE IF EXISTS [dbo].[Favorites];

CREATE TABLE [dbo].[Favorites] (
    [ClientId] NVARCHAR(450) NOT NULL,
    [ServiceId] INT NOT NULL,
    [ShowState] BIT DEFAULT 1 NOT NULL,

    FOREIGN KEY ([ClientId]) REFERENC
ES [dbo].[Users]([Id]) ON DELETE CASCADE,
    FOREIGN KEY ([ServiceId]) REFEREN
CES [dbo].[Services]([Id]) ON DELETE CASCADE,
    PRIMARY KEY ([ClientId], [Service
Id])
)

GO
-- -----
-- Table [HotelService].[dbo].[ShoppingCarts]
-- -----
DROP TABLE IF EXISTS [dbo].[Orders];

CREATE TABLE [dbo].[Orders] (
    [Id] INT NOT NULL IDENTITY,
    [RoomContractId] INT NOT NULL,
    [PhoneNumber] NVARCHAR(256) NOT NULL,
    [Name] NVARCHAR(256) NOT NULL,
    [PaymentDetails] NVARCHAR(256) DEFAULT NULL,
    [CreditCardNumber] NVARCHAR(256) NOT NULL,
    [PaymentDate] DATETIME2 DEFAULT getdate() NOT
NULL,
    [CostTotal] DECIMAL(10,2) NOT NULL,

    FOREIGN KEY ([RoomContractId]) RE
FERENCES [dbo].[RoomContracts]([Id]) ON DELETE CASCADE,
    PRIMARY KEY ([Id])
)

GO
-- -----
-- Table [HotelService].[dbo].[Requests]
-- -----

```

```

DROP TABLE IF EXISTS [dbo].[Requests];

CREATE TABLE [dbo].[Requests] (
    [Id] INT NOT NULL IDENTITY,
    [OrderId] INT DEFAULT NULL,
    [ServiceId] INT NOT NULL,
    [Comment] NVARCHAR(256) NULL,
    [Quantity] INT DEFAULT 1 NOT NULL,
    [DeliveryDate] DATETIME2 DEFAULT getdate() NOT
    NULL,

    FOREIGN KEY ([OrderId]) REFERENCE
S [dbo].[Orders]([Id]) ON DELETE CASCADE,

    FOREIGN KEY ([ServiceId]) REFEREN
CES [dbo].[Services]([Id]) ON DELETE CASCADE,
    PRIMARY KEY ([Id])
)

GO
-- -----
-- Table [HotelService].[dbo].[States]
-- -----
DROP TABLE IF EXISTS [dbo].[States];

CREATE TABLE [dbo].[States] (
    [Id] INT NOT NULL IDENTITY,
    [Value] NVARCHAR(256) NOT NULL UNIQUE,

    PRIMARY KEY ([Id])
)

GO
-- -----
-- Table [HotelService].[dbo].[RequestStates]
-- -----
DROP TABLE IF EXISTS [dbo].[RequestStates];

CREATE TABLE [dbo].[RequestStates] (
    [RequestId] INT NOT NULL,
    [StateId] INT NOT NULL,
    [ChangeDate] DATETIME2 DEFAULT getdate() NOT
    NULL,
    [Comment] NVARCHAR(256) NULL,

    FOREIGN KEY ([RequestId]) REFEREN
CES [dbo].[Requests]([Id]) ON DELETE CASCADE,

    FOREIGN KEY ([StateId]) REFERENCE
S [dbo].[States]([Id]) ON DELETE CASCADE,
    PRIMARY KEY ([RequestId], [StateI
d])
)

GO

```

Приложение В
ЛИСТИНГ ПРЕДСТАВЛЕНИЙ И ТРИГГЕРОВ
НА ЯЗЫКЕ SQL
(рекомендуемое)

Листинг В.1 – Представление UsersRequests

```

USE HotelService;
GO
-- Информация по заявкам
CREATE VIEW UsersRequests AS
SELECT Orders.Name AS [Name],
       Orders.PhoneNumber AS Phone,
       Orders.PaymentDate AS [Date],
       [Services].Title AS [Service],
       [Services].Cost AS Cost,
       Requests.Quantity AS Quantity,
       Rooms.Number AS Number,
       Buildings.Name AS Building

FROM Orders
INNER JOIN Requests ON Requests.OrderId = Orders.Id
INNER JOIN [Services] ON Requests.ServiceId = [Services].Id
INNER JOIN RoomContracts ON RoomContracts.Id = Orders.RoomContractId
INNER JOIN Rooms ON Rooms.Id = RoomContracts.RoomId
INNER JOIN Buildings ON Rooms.BuildingId = Buildings.Id

```

Листинг В.2 – Представление OccupiedRooms

```

GO
-- Информация по номерам
CREATE VIEW OccupiedRooms AS
SELECT Rooms.Number AS [Number],
       Rooms.Type AS [Type],
       RoomContracts.CheckInDate AS [CheckInDate],
       RoomContracts.CheckOutDate AS [CheckOutDate],
       Users.FirstName AS [FirstName],
       Users.LastName AS [LastName]

FROM RoomContracts
INNER JOIN Rooms ON RoomContracts.RoomId = Rooms.Id
INNER JOIN Users ON RoomContracts.ClientId = Users.Id

```

Листинг В.3 – Представление RoomsRequests

```

GO
-- Информация по заявкам в номер
CREATE VIEW RoomsRequests AS
SELECT Rooms.Number AS Number,
       Requests.Quantity AS Quantity,
       Services.Title AS Service,
       Services.Cost*Quantity AS Cost

FROM Orders
INNER JOIN Requests ON Requests.OrderId = Orders.Id
INNER JOIN [Services] ON Requests.ServiceId = [Services].Id

```

```
INNER JOIN RoomContracts ON RoomContracts.Id = Orders.RoomContractId
INNER JOIN Rooms ON Rooms.Id = RoomContracts.RoomId
```

Листинг В.4 – Процедура UpdateRequestStatus

```
GO
CREATE PROCEDURE UpdateRequestStatus
    @status INT,
    @request INT
AS

INSERT INTO RequestStates(StateId, RequestId) VALUES (@status, @request)
```

Листинг В.5 – Процедура AddService

```
-- Добавление услуги
GO
CREATE PROCEDURE AddService
    @title NVARCHAR(256),
    @cost DECIMAL(8,2),
    @subtitle NVARCHAR(256),
    @category INT,
    @description NVARCHAR(256) = NULL
AS
INSERT INTO Services(Title, Subtitle, ServiceCategoryId, Cost, Description)
VALUES(@title, @subtitle, @category, @cost, @description)
```

Листинг В.6 – Процедура FindUserByName

```
-- Поиск гостя
GO
CREATE PROCEDURE FindUserByName
    @fname NVARCHAR(256),
    @lname NVARCHAR(256)
AS
SELECT Passport, Email, Id, FirstName, LastName FROM Users WHERE FirstName =
@fname AND LastName = @lname
```

Листинг В.7 – Триггер Service_CostChanged_INSERT

```
GO
CREATE TRIGGER Service_CostChanged_INSERT
ON [dbo].[Services]
AFTER INSERT, UPDATE
AS
INSERT INTO [dbo].[CostChanges] (ServiceId, NewCostValue)
SELECT Id, Cost
FROM INSERTED
```

Листинг В.8 – Триггер Request_InitialStatus_INSERT

```
GO
CREATE TRIGGER Request_InitialStatus_INSERT
ON [dbo].[Requests]
AFTER INSERT
AS
INSERT INTO [dbo].[RequestStates](RequestId, StateId)
SELECT Id, 1
FROM INSERTED
```

Приложение Г
ЛИСТИНГ ПРИЛОЖЕНИЯ НА ЯЗЫКЕ C#
(рекомендуемое)

Листинг Г.1 – Код классаStartup.cs

```

namespace HotelService
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add
        services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            Configuration.Bind("ConnectionStrings", new Config());
            Configuration.Bind("Developer", new Developer());

            services.Configure<CookiePolicyOptions>(options =>
            {
                // This lambda determines whether user consent for non-
                essential cookies is needed for a given request.
                options.CheckConsentNeeded = context => true;
                options.MinimumSameSitePolicy = SameSiteMode.None;
            });

            services.AddDbContext<HotelServiceContext>(options =>
options.UseSqlServer(Config.DefaultConnection));
            services.AddDatabaseDeveloperPageExceptionFilter();

            services.AddIdentity<User, Role>(options =>
            {
                options.SignIn.RequireConfirmedAccount = false;
                options.Password.RequireLowercase = false;
                options.Password.RequireUppercase = false;
                options.Password.RequireNonAlphanumeric = false;
                options.Password.RequiredLength = 5;
            }).AddDefaultTokenProviders()
            .AddDefaultUI()
            .AddEntityFrameworkStores<HotelServiceContext>();

            services.AddScoped<ICatalogManager, CatalogManager>();
            services.AddScoped<IOrderManager, OrderManager>();
            services.AddScoped<ShoppingCart>(SessionCart.GetCart);
            services.AddSingleton<IHttpContextAccessor,
HttpContextAccessor>();

            services.AddCoreAdmin("Admin", "Developer");
            services.AddControllersWithViews().AddRazorRuntimeCompilation();
            services.AddRazorPages();
            services.AddDistributedMemoryCache();
            services.AddSession();
        }

        // This method gets called by the runtime. Use this method to
        configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment
env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
        }
    }
}

```

```

        app.UseMigrationsEndPoint();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
        // The default HSTS value is 30 days. You may want to change
this for production scenarios, see https://aka.ms/aspnetcore-hsts.
        app.UseHsts();
    }

    app.UseStaticFiles();
    app.UseHttpsRedirection();

    app.UseStatusCodePages();
    app.UseSession();
    app.UseRouting();

    app.UseAuthentication();
    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        // маршрут для области Admin
        endpoints.MapAreaControllerRoute(
            name: "Admin_Area",
            areaName: "Admin",
            pattern: "Admin/{controller=Home}/{action=Index}/{id?}");

        // маршрут для области Client
        endpoints.MapAreaControllerRoute(
            name: "Client_Area",
            areaName: "Client",
            pattern:
"Client/{controller=Home}/{action=Index}/{id?}");

        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
        endpoints.MapRazorPages();
    });
}

```

При разработке приложения применялся архитектурный паттерн MVC.

Исходный код приложения доступен по ссылке:

<https://github.com/MyakishHentai/HotelService>

Область моделей и контекста данных:

<https://github.com/MyakishHentai/HotelService/tree/main/HotelService/Models>

Область администрирования расположена по пути:

<https://github.com/MyakishHentai/HotelService/tree/main/HotelService/Areas/Admin>

Область клиента расположена по пути:

<https://github.com/MyakishHentai/HotelService/tree/main/HotelService/Areas/Client>

Приложение Д
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ
(рекомендуемое)

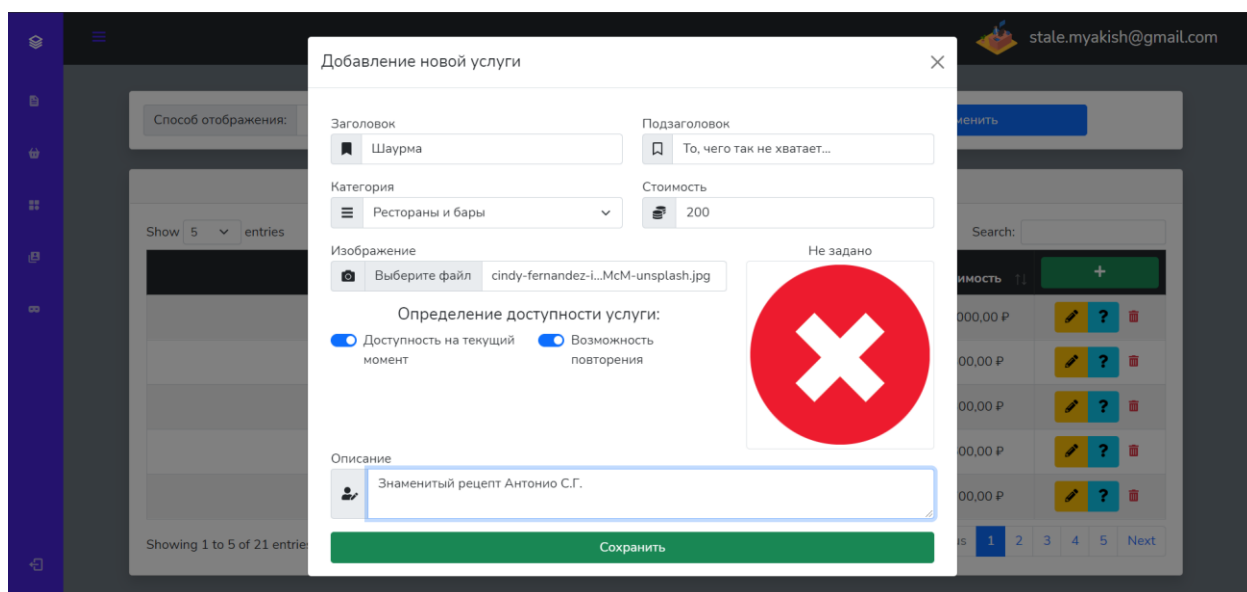


Рисунок Д.1 – Модальное окно добавления новой услуги с заполненными данными

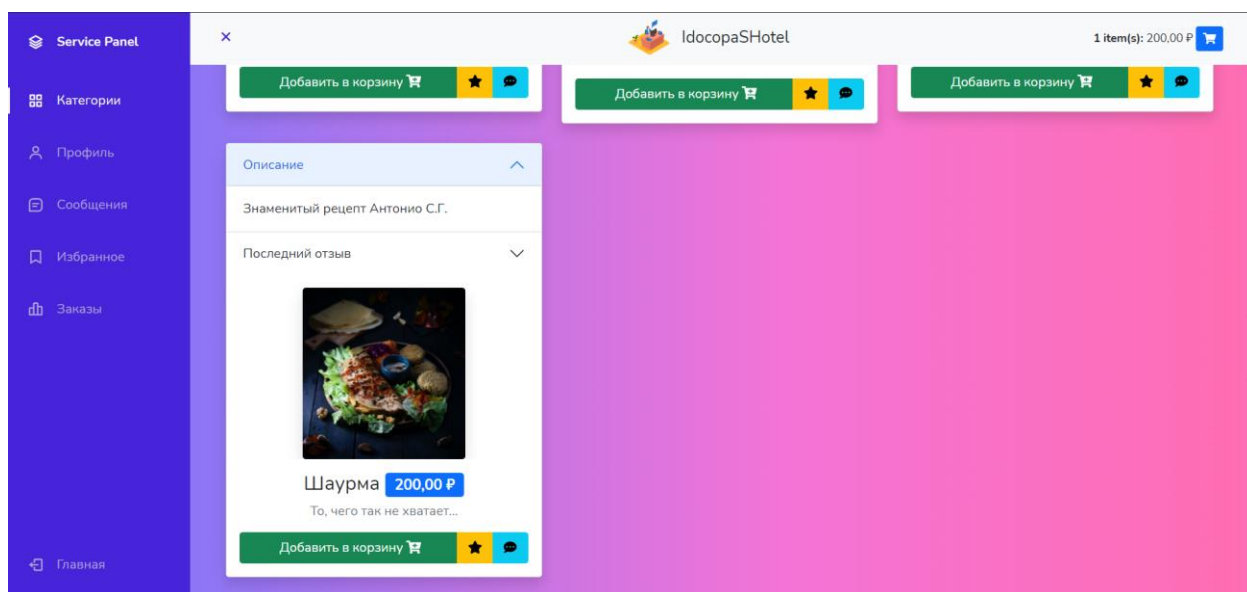


Рисунок Д.2 – Результат добавления новой услуги в области клиента

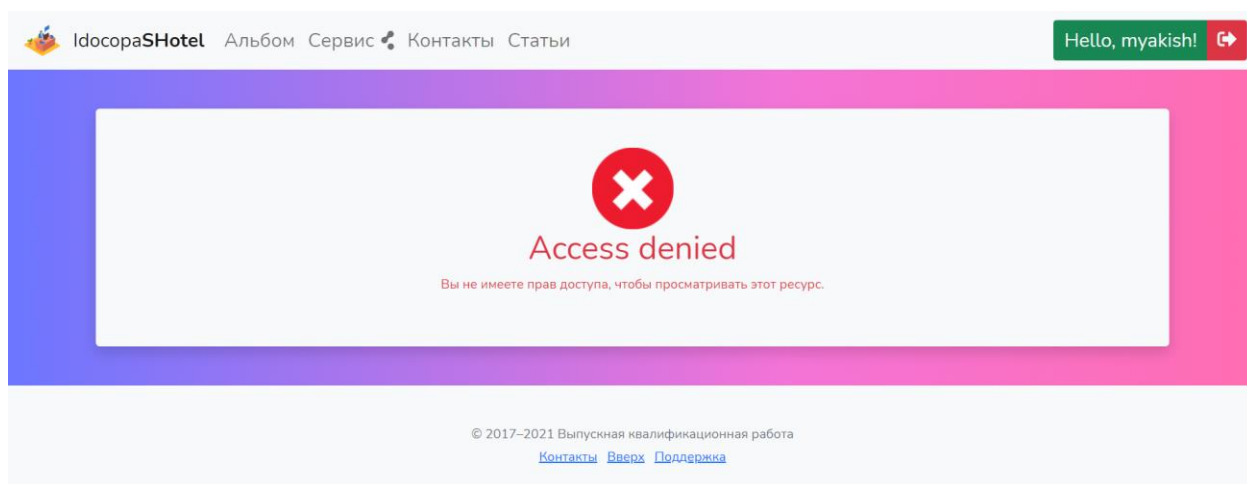


Рисунок Д.3 – Проверка наличия прав доступа при попытке перехода на панель администратора из учетной записи пользователя «myakish»

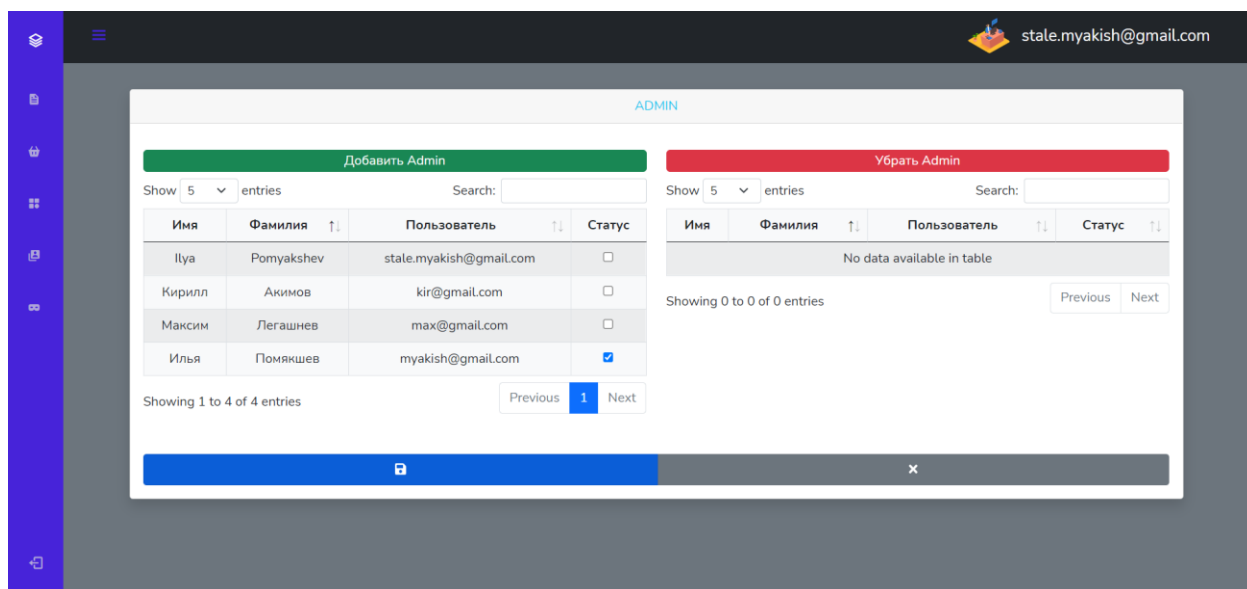


Рисунок Д.4 – Добавление прав администратора пользователю «myakish» администратором

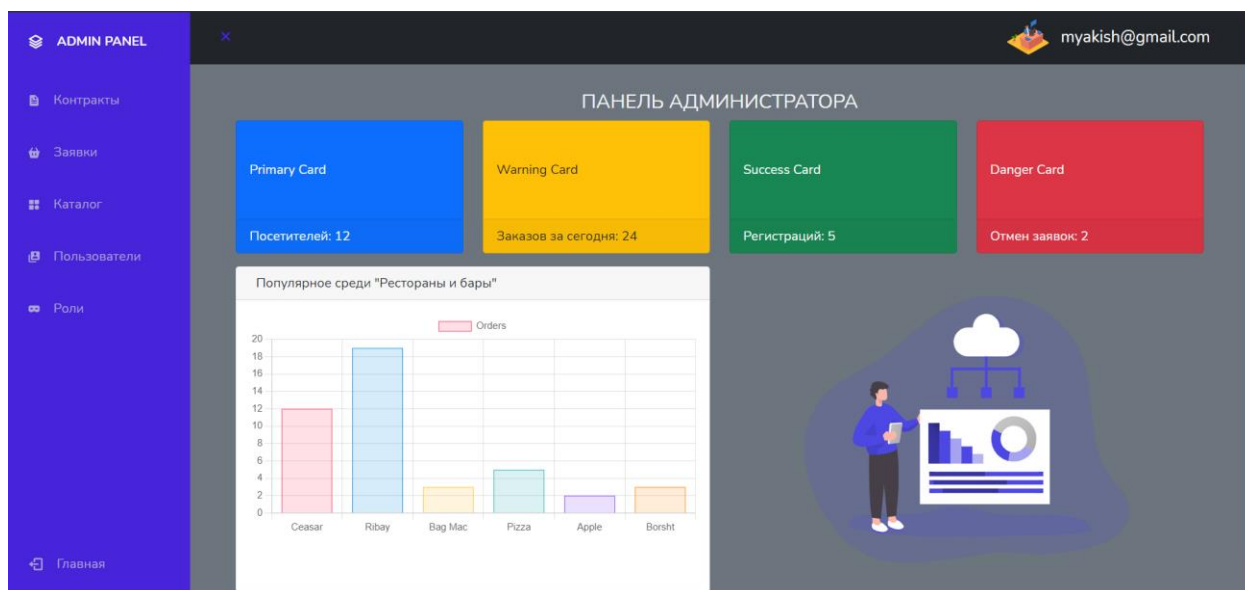


Рисунок Д.5 – Результат добавления к роли Admin пользователя «myakish»

Листинг Д.1 – Код модульного теста CartTests.cs

```
public class CartTests
{
    [Fact]
    public void Can_Add_New_Lines()
    {
        // Arrange - create some test products
        var Service1 = new ServiceItem { Id = 1, Title = "P1", Cost = 200
.00M};
        var Service2 = new ServiceItem { Id = 2, Title = "P2", Cost = 400
.00M };

        // Arrange - create a new cart
        var Target = new ShoppingCart();
        // Act
        Target.AddItem(Service1, 1);
        Target.AddItem(Service2, 1);
        Request[] Results = Target.Lines.ToArray();
        // Assert
        Assert.Equal(2, Results.Length);
        Assert.Equal(Service1, Results[0].Service);
        Assert.Equal(Service2, Results[1].Service);
    }

    [Fact]
    public void Can_Add_Quantity_For_Existing_Lines()
    {
        var Service1 = new ServiceItem { Id = 1, Title = "P1", Cost = 200
.00M };
        var Service2 = new ServiceItem { Id = 2, Title = "P2", Cost = 400
.00M };

        // Arrange - create a new cart
        var Target = new ShoppingCart();
        // Act
        Target.AddItem(Service1, 1);
        Target.AddItem(Service2, 1);
        Target.AddItem(Service1, 10);
        Request[] Results = Target.Lines
```

```

        .OrderBy(c => c.Service.Id).ToArray();
        // Assert
        Assert.Equal(2, Results.Length);
        Assert.Equal(11, Results[0].Quantity);
        Assert.Equal(1, Results[1].Quantity);
    }

    [Fact]
    public void Can_Remove_Line()
    {
        // Arrange - create some test products
        var Service1 = new ServiceItem { Id = 1, Title = "P1", Cost = 200
.00M };
        var Service2 = new ServiceItem { Id = 2, Title = "P2", Cost = 400
.00M };
        var Service3 = new ServiceItem { Id = 3, Title = "P2", Cost = 400
.00M };

        // Arrange - create a new cart
        var Target = new ShoppingCart();
        // Arrange - add some products to the cart
        Target.AddItem(Service1, 1);
        Target.AddItem(Service2, 3);
        Target.AddItem(Service3, 5);
        Target.AddItem(Service2, 1);
        // Act
        Target.RemoveLine(Service2);
        // Assert
        Assert.Empty(Target.Lines.Where(c => c.Service == Service2));
        Assert.Equal(2, Target.Lines.Count());
    }

    [Fact]
    public void Calculate_Cart_Total()
    {
        var Service1 = new ServiceItem { Id = 1, Title = "P1", Cost = 200
.00M };
        var Service2 = new ServiceItem { Id = 2, Title = "P2", Cost = 400
.00M };

        // Arrange - create a new cart
        var Target = new ShoppingCart();
        // Act
        Target.AddItem(Service1, 1);
        Target.AddItem(Service2, 1);
        Target.AddItem(Service1, 3);
        decimal Result = Target.ComputeTotalValue();
        // Assert
        Assert.Equal(1200.00M, Result);
    }

    [Fact]
    public void Can_Clear_Contents()
    {
        // Arrange - create some test products
        var Service1 = new ServiceItem { Id = 1, Title = "P1", Cost = 200
.00M };
        var Service2 = new ServiceItem { Id = 2, Title = "P2", Cost = 400
.00M };

        // Arrange - create a new cart
        var Target = new ShoppingCart();
        // Arrange - add some items
        Target.AddItem(Service1, 1);
        Target.AddItem(Service2, 1);
        // Act - reset the cart
        Target.Clear(); Assert.Empty(Target.Lines);}}

```

Приложение Е
СПЕЦИФИКАЦИЯ СУЩНОСТЕЙ
(рекомендуемое)

Таблица Е.1 – Описание атрибутов для сущности Users (пользователи)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
Id	Id	символьный	450	«A-Z, 0-9, -»	нет	РК	3556ECC9-0F31-434D	
FirstName	Имя	символьный	256	«А-я»	нет		Антон	
LastName	Фамилия	символьный	256	«А-я, -»	нет		Иванов	
Patronymic	Отчество	символьный	256	«А-я, -»	да		Петрович	
Gender	Пол	символьный	16	«муж, жен»	да		муж	
Passport	Паспорт	символьный	32	«0-9»	нет		5612392562	
Foreigner Status	Является иностранцем	битовый	1	NULL, 1, 0	да		1	
RegistrationDate	Дата регистрации	датавремя	19	01/01/0001 - 31/12/9999	нет		2020-10-11 12:12:12	
ImagePath	Путь к изображению	символьный	1024		да		~/img/local/users/12.png	

Описание атрибутов сущности «Buildings (Корпуса зданий)» представлено в таблице Е.2.

Таблица Е.2 – Описание атрибутов для сущности Buildings (Корпуса зданий)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
Id	ID	Целый	10	«0-9»	нет	РК	1	

Продолжение таблицы Е.2

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
AdminId	Id администратора, системы обслуживания	символьный	450	«А-Z, 0-9, -»	да	FK	3556 ECC 9- 0F3 1- 434 D- 8452 - E7D F1E 4A4 DE0	
Name	Название	символьный	256	«А-я»	нет		Idoc opas Delt a	
Addresses	Адрес	символьный	256		нет		53.1 8225 3, 45.0 0383 4	
Description	Описание	символьный	256		да		Лучший корпус с большим разнообразием услуг	
ImagePath	Путь к изображению	символьный	1024		да		~/img/local/buildings/12.png	

Описание атрибутов сущности Rooms (Гостиничные номера) представлено в таблице Е.3.

Таблица Е.3 – Описание атрибутов для сущности Rooms (Гостиничные номера)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
Id	ID	Целый	10	«0-9»	нет	РК	1	
BuildingId	Id корпуса здания	целый	10	«0-9»	нет	FK	11	
Number	Номер	символьный	256		нет		C11	
Type	Тип	символьный	256	«STD, Family Room, Royal Suite»	нет		STD	
SleepingPlaces	Количество спальных мест	целый	10	«0-9»	нет		3	
Cost	Стоимость	вещественный	8, 2	«0-9»	нет		2000.00	
ImagePath	Путь к изображению	символьный	1024		да		~/img/local/buildings/12.png	

Описание сущности ServiceCategories (Категории услуг) представлено в таблице Е.4.

Таблица Е.4 – Описание атрибутов для сущности ServiceCategories (Категории услуг)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
Id	ID	целый	10	«0-9»	нет	РК	1	

Продолжение таблицы Е.4

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
SubCategoryId	Id подкатегории	целый	10	«0-9»	да	FK	135	
Title	Название категории	символьный	256	«А-я»	нет		Ресторан	
Subtitle	Название подкатегории	символьный	256	«А-я»	да		Кухня	
Description	Описание	символьный	512		да		1001 блюдо всех кухонь мира	
ImagePath	Путь к изображению	символьный	1024		да		~/img/local/categories/12.png	
AvailableState	Состояние доступности	битовый	1	1, 0	нет		1	

Описание сущности Services (Услуги) представлено в таблице Е.5.

Таблица Е.5 – Описание атрибутов для сущности Services (Услуги)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
Id	ID	целый	10	«0-9»	нет	PK	1	
SubCategoryId	Id подкатегории	целый	10	«0-9»	нет	FK	225	
Title	Название категории	символьный	256	«А-я»	нет		Тайский	
Subtitle	Уточнение, обычно корпус	символьный	256	«А-я»	нет		IdocopasDelta	

Продолжение таблицы Е.5

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
Description	Описание	символьный	512		да		Массаж, который заставит почувствовать себя еще моложе	
ImagePath	Путь к изображению	символьный	1024		да		~/img/local/services/12.png	
Cost	Стоимость	дробный	8, 2	«0-9»	нет		1500.00	
Rating	Рейтинг	вещественный	1	«1-5»	да		5	
AvailableState	Состояние доступности	битовый	1	1, 0	нет		1	
RepeatState	Возможность повторения	битовый	1	1, 0	нет		1	*п в заказе
AddedDate	Дата добавления услуги	дата	19	01/01/0001 - 31/12/9999	нет		2020-11-11	

Описание сущности Feedbacks (Отзывы) представлено в таблице Е.6.

Таблица Е.6 – Описание атрибутов для сущности Feedbacks (Отзывы)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ClientId	ID клиента	символьный	450	«A-Z, 0-9, -»	нет	PK	3556ECC9-0F31-434D-8452-E7DF1E4A4DE0	

Продолжение таблицы Е.6

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
Service Id	ID услуги	Целый	10	«0-9»	нет	FK	22	
Rating	Рейтинг	целый	1	«1-5»	нет		5	
Review	Отзыв	символьный	512		да		Все понравилось, рекомендую	
Writing Date	Дата написания	датавремя	19	01/01/0001 - 31/12/9999	нет		2020-12-11 15:22:56	

Описание сущности RoomContracts (Контракты на номера) представлено в таблице Е.7.

Таблица Е.7 – Описание атрибутов для сущности RoomContracts (Контракты на номера)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
Id	ID	целый	10		нет	PK	12	
ClientId	ID клиента	символьный	450	«A-Z, 0-9, -»	нет	FK	3556ECC9-0F31-434D-8452-E7DF1E4A4DE0	
RoomId	ID номера клиента	целый	10	«0-9»	нет	FK	12	
ConclusionDate	Дата заключения контракта	датавремя	19	01/01/0001 - 31/12/9999	нет		2020-10-25 14:43:11	

Продолжение таблицы Е.7

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
CheckInDate	Дата въезда	дата	19	01/01/0001 - 31/12/9999	нет		2020-11-02 16:00:00	
CheckOutDate	Дата выезда	дата	19	01/01/0001 - 31/12/9999	да		2020-11-10 16:00:00	

Описание сущности Articles (Статьи) представлено в таблице Е.8.

Таблица Е.8 – Описание атрибутов для сущности Articles (Статьи)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ID	ID	целый			нет	PK	1	
AuthorId	Id подкатегории	символьный	450		нет	FK	153	
Title	Название	символьный	256		нет		Новое меню	
Review	Содержание	символьный	512		Нет		Попробуйте новое меню в ресторане Omega.	
ImagePath	Путь к изображению	символьный	1024		да		~/img/local/articles/12.png	
WritingDate	Дата написания	датавремя	19	01/01/0001 - 31/12/9999	нет		2020-09-22 16:56:12	

Описание сущности PriceChanges (Цены) представлено в таблице Е.9.

Таблица Е.9 – Описание атрибутов для сущности PriceChanges (Цены)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ID	ID	целый	10	«0-9»	нет	РК	1	
Service Id	Id услуги	целый		«0-9»	нет	FK	178	
Change Date	Дата изменения	датавремя	1	01/01/0001 - 31/12/9999	нет		2020-12-31 14:22:15	
NewPriceValue	Новое значение стоимости	вещественный	8, 2	«0-9»	нет		2500.00	

Описание сущности Favorites (Избранное) представлено в таблице Е.10.

Таблица Е.10 – Описание атрибутов для сущности Favorites (Избранное)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
ClientId	Id клиента	символьный	450	«0-9»	нет	РК	3556ECC9-0F31-434D	
ServiceId	Id услуги	целый	10	«0-9»	нет	FK	1	
ShowState	Состояние видимости	битовый	1	1, 0	нет		1	

Описание сущности Requests (Заявки) представлено в таблице Е.11.

Таблица Е.11 – Описание атрибутов для сущности Requests (Заявки)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
Id	ID	целый	10	«0-9»	нет	РК	1	
Service Id	Id услуги	целый	10	«0-9»	нет	FK	7	
Quantity	Повторения	целый	10	«0-9»	нет		1	
Comment	Комментарий	символьный	256		да		Поострее	
DeliveryDate	Дата назначения	датавремя	12	01/01/0001 - 31/12/9999	нет		2020-12-31 14:22:15	

Описание сущности States (Состояния) представлено в таблице Е.12.

Таблица Е.12 – Описание атрибутов для сущности States (Состояния)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
Id	ID	целый	10	«0-9»	нет	РК	1	
Value	Значение	символьный	256		да		Отказ	

Описание сущности RequestStates (Состояния заявок) представлено в таблице Е.13.

Таблица Е.13 – Описание атрибутов для сущности RequestStates (Состояния заявок)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
RequestId	Id заявки	целый	10	«0-9»	нет	FK	1	
StateId	Id состояния	целый	10	«0-9»	нет	FK	4	
Quantity	Повторения	целый	10	«0-9»	нет		1	
Comment	Комментарий	символьный	256		да		Выполнен	
ChangeDate	Дата изменения	датавремя	12	01/01/0001 - 31/12/9999	нет		2020-12-31 14:22:15	

Описание сущности RequestStates (Состояния заявок) представлено в таблице Е.14.

Таблица Е.14 – Описание атрибутов для сущности Orders (Заказы)

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения Null	Роль	Пример	Примечание
Id	ID	целый	10	«0-9»	нет	PK	1	
RoomContractId	Id контракта с ГК	целый	10	«0-9»	нет	FK	7	
CreditCardNumber	Номер карты	символьный	256		да		4407-1233-1233-5555	
PaymentDate	Дата оплаты	датавремя	12	01/01/0001 31/12/9999	нет		2020-12-31 14:22:15	