



КУРСОВОЙ ПРОЕКТ

По дисциплине: МДК 01.01 Разработка программных модулей

Тема: Разработка программного модуля «Система учета и анализа данных о гостях отеля»

Специальность 09.02.07 «Информационные системы и программирование»

**Выполнил студент(ка) группы _____ А.А. Копченова
313ИС-22**

Руководитель _____ Л.А.Хамрилова

Москва 2024



УТВЕРЖДАЮ
Зам. директора КМПО

_____ **С.Ф. Гасанов**
«_____» _____ **2024 г.**

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

По дисциплине: МДК 01.01 Разработка программных модулей

Специальность 09.02.07 «Информационные системы и программирование»

Студент группы 313ИС-22 Анна Копченова

ТЕМА: Разработка программного модуля «Система учета и анализа данных о гостях отеля»

Дата выдачи задания «_____» _____
2024 г.

Срок сдачи проекта «_____» _____
2024 г.

Москва 2024

Перечень вопросов, подлежащих разработке:

ВВЕДЕНИЕ

1. Описание предметной области
 - 1.1 Введение в предметную область
 - 1.2 Анализ готовых решений
2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ
 - 2.1. Общее назначение программы
 - 2.2. Требования к функциям, выполняемым системой
 - 2.3. План тестирования
3. РЕАЛИЗАЦИЯ ПРОЕКТНОЙ СИСТЕМЫ
 - 3.1. Описание среды разработки
 - 3.1.1. Выбор и описание программных инструментов
 - 3.1.2. Обоснование выбора инструментария по разработке
 - 3.2. Разработка программного модуля
 - 3.2.1. Реализация пользовательского интерфейса программы
 - 3.2.2. Описание кодом функциональных узлов модуля
 - 3.2.3. Результат работы и тестирования

ЗАКЛЮЧЕНИЕ

Исходные данные: 27.02.2006 №152-ФЗ, 27.02.2006 №149-ФЗ, 26.07.2017 №187-ФЗ, ГОСТ Р ИСО/МЭК 12207, ГОСТ 19.005–85, ГОСТ 19.105–78

Перечень разрабатываемых материалов для визуализации: ER-диаграмма базы данных, диаграмма прецедентов, архитектура программного модуля, экранные формы интерфейса, результаты тестирования.

Задание выдал:

Руководитель курсового проекта

Л. А. Хамрилова

Задание принял к исполнению

А. А. Копченова

Рассмотрено

на заседании предметно-цикловой комиссии
информационных технологий и системного
администрирования

Протокол № ____ от «__» _____ 2024 г.

Председатель ПЦК _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	5
1.1 Введение в предметную область	5
2 ПРОЕКТИРОВАНИЕ СИСТЕМ.....	9
2.1 Общее назначение систем	9
2.2 Требования к функциям, выполняемой системой	9
2.3 План тестирования.....	12
3 РЕАЛИЗАЦИЯ ПРОЕКТНОЙ СИСТЕМЫ.....	14
3.1 Описание среды разработки.....	14
3.1.1 Выбор и описание программных инструментов.....	14
3.1.2 Обоснование выбора инструментария по разработке	15
3.2 Разработка программного модуля	15
3.2.1 Реализация пользовательского интерфейса.....	16
3.2.2 Описание кодом функциональных узлов модуля	17
3.2.3 Результат работы и тестирования	17
ЗАКЛЮЧЕНИЕ	24
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	25
ПРИЛОЖЕНИЕ 1 Диаграмма прецедентов.....	26
ПРИЛОЖЕНИЕ 2 План-тест.....	30
ПРИЛОЖЕНИЕ 3 Функционал кнопок	54
ПРИЛОЖЕНИЕ 4 Листинг кода.....	69

ВВЕДЕНИЕ

В современном мире, где путешествия и деловые поездки стали неотъемлемой частью жизни многих людей, гостиничный бизнес играет ключевую роль в обеспечении комфорта и удобства для гостей. Эффективное управление этим бизнесом требует наличия современных информационных технологий, которые позволяют автоматизировать процессы учета и анализа данных о гостях.

Целью данного курсового проекта является разработка программного модуля «Система учета и анализа данных о гостях», который позволит автоматизировать процессы сбора, хранения и обработки, о гостях в гостиницы. Этот модуль будет предназначен для использования администраторами гостиниц, которые смогут получать доступ к необходимой информации о гостях, их пожеланиях и истории пребывания в отеле.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучение и анализ существующих решений по автоматизации учета документов в организациях.
2. Определение требований и функционала для разрабатываемого программного модуля.
3. Проектирование архитектуры программного модуля.
4. Разработка программного модуля на основе выбранных технологий.
5. Тестирование и отладка созданного программного продукта.

Объект исследования в данной работе является процесс автоматизации учета информации о гостях в гостинице.

Предметом исследования является разработка программного модуля «Система учета и анализа данных о гостях отеля», предназначенного для автоматизации процессов управления информацией о гостях в гостиничном бизнесе.

Разработка системы учета и анализа данных о гостях отеля является актуальной задачей в условиях современной гостиничной индустрии.

- Повышение эффективности: Автоматизация процессов учета и анализа данных о гостях позволяет сократить время на ручную обработку информации, повысить точность и оперативность, а также оптимизировать работу персонала.
- Улучшение сервиса: Сбор и анализ данных о гостях позволяет лучше понять потребности клиентов, персонализировать обслуживание и повысить удовлетворенность гостей.
- Принятие обоснованных решений: Анализ данных о гостях помогает принимать взвешенные решения по управлению отелем, оптимизации цен, разработке маркетинговых стратегий.

Данная тема является актуальной и перспективной задачей, способной значительно повысить эффективность работы гостиничного бизнеса, улучшить качество обслуживания и принять обоснованные решения по управлению отелем.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Введение в предметную область

Современные системы учета и анализа данных о гостях отеля (СУ-АДГО) представляют собой комплексные решения, позволяющие оптимизировать гостиничный бизнес. Они автоматизируют ключевые процессы, такие как управление бронированием, учет гостей, управление номерами и финансовыми операциями. СУАДГО обеспечивает централизованное хранение информации о гостях, повышая точность учета и исключая потери данных. Анализ данных позволяет выявлять тенденции и оптимизировать работу гостиницы.

Функционал СУАДГО:

Управление бронированием (создание, изменение, отмена); Учет гостей (персональные данные, история посещений, предпочтения); Управление номерами (доступность, назначение гостям); Управление финансами (расчеты с гостями, отчетность); Анализ данных (оптимизация управления, выявление тенденций).

Преимущества СУАДГО:

Ускорение и оптимизация бизнес-процессов; Повышение точности учета.; Улучшение качества обслуживания (персонализация услуг); Увеличение прибыли.

Принципы разработки:

Интеграция с другими системами гостиницы (бронирование, CRM, управление доступом); Использование реляционных баз данных для эффективного управления информацией.

СУАДГО обеспечивает надежную защиту персональных данных гостей и соответствует законодательным требованиям. Технологии программирования: Разработка СУАДГО требует использования современных языков программирования и технологий веб-разработки.

Современные СУАДГО должны удовлетворять следующим требованиям см. «Таблица 1»:

Требования	Нормативный документ
Регулирует обработку персональных данных в РФ.	Федеральный закон от 27.02.2006 №152-ФЗ «О персональных данных» [1]
Регулирует сферу информационных технологий в России, в том числе использование ИТ в гостиничной сфере	Федеральный закон от 27.02.2006 №149-ФЗ «Об информации, информационных технологиях и защите информации» [2]
Обеспечении защиты персональных данных в инфраструктуре Российской Федерации.	Федеральный закон от 26.07.2017 №187-ФЗ «О безопасности критической информационной инфраструктуры Российской Федерации» [3]
Устанавливает требования к защите информации в процессе разработки и эксплуатации ПО.	ГОСТ Р ИСО/МЭК 12207 "Информационные технологии. Процессы жизненного цикла программных средств» [4]
Стандарт устанавливает условные графические обозначения элементов структуры Р-схем, а также правила их выполнения автоматическим и (или) ручным способами.	ГОСТ 19.005–85 «Р-схемы алгоритмов и программ. Обозначения условные графические правила выполнения» [5]
Стандарт устанавливает общие требования к оформлению программных документов для вычислительных машин, комплексов и систем, независимо от их назначения и области применения и предусмотренных стандартами ЕСПД для любого выполнения документов на различных носителях данных.	ГОСТ 19.105–78 «Общие требования к программным документам» [6]

Продолжение таблицы 1

Требования	Нормативный документ
Стандарт устанавливает требования к содержанию и оформлению программного документа «Руководство программиста», определённого ГОСТ 19.101.-77.	ГОСТ 19.504–79 «Руководство программиста. Требования к содержанию и оформлению» [7]

Разработка СУАДГО будет осуществляться в соответствии с этапами модели водопада и использованием выбранных инструментов:

1. Анализ требований: Определение функциональных и нефункциональных требований к системе.
2. Проектирование: Разработка архитектуры системы, структуры базы данных и интерфейсов.
3. Разработка: Реализация кода системы в соответствии с проектной документацией.
4. Тестирование: Проверка функциональности и безопасности системы.
5. Внедрение: Установка и настройка системы в гостинице.
6. Обучение: Обучение персонала работе с СУАДГО.
7. Техническая поддержка: Обеспечение бесперебойной работы системы и предоставление необходимой технической помощи.

Разработка СУАДГО является сложным и многогранным процессом, требующим глубокого понимания требований гостиничного бизнеса, современных технологий и законодательных аспектов. Однако, инвестиции в разработку такой системы оправданы выгодами, которые она приносит, позволяя гостинице повысить эффективность работы, улучшить качество обслуживания и увеличить прибыль.

1.2 Анализ готовых решений

Существует множество готовых решений для автоматизации гостиничного бизнеса, предлагающих функции учета и анализа данных о гостях. Рассмотрим несколько наиболее популярных систем:

1. Oracle Hospitality Opera: Комплексная система управления отелями, охватывающая все основные функции, от бронирования до управления персоналом, и включающая аналитические инструменты для принятия управленческих решений.
2. Micros Fidelio Suite: Гибкая система управления отелями с широким функционалом автоматизации и мощными аналитическими инструментами для глубокого анализа данных о гостях.
3. Infor HMS: Надежное и масштабируемое интегрированное решение для управления гостиницами любого размера, включающее аналитику данных для принятия оптимальных решений.
4. Protel Systems: это интуитивное и гибко настраиваемое программное обеспечение для управления гостиницами, охватывающее учет гостей, бронирование и финансовый учет.

Таким образом, анализируя готовую программную продукцию, мы видим, что существующие решения предлагают широкий функционал и инструменты для учета и анализа данных о гостях отеля. Выбор конкретного решения зависит от размера гостиницы, ее специфических требований и бюджета. В рамках данной курсовой работы сосредоточимся на разработке собственного решения, что позволит более гибко учитывать конкретные нужды и требования гостиничного бизнеса.

2 ПРОЕКТИРОВАНИЕ СИСТЕМ

2.1 Общее назначение систем

Разработка программного модуля «Система учета и анализа данных о гостях отеля» направлена на автоматизацию процессов, связанных с управлением информацией о клиентах, их предпочтениях и поведении в отеле. Основное назначение системы заключается в создании единой базы данных, которая позволит эффективно собирать, хранить и обрабатывать информацию о гостях, а также анализировать полученные данные для улучшения качества предоставляемых услуг.

Внедрение системы принесет отелю следующие преимущества:

1. Снижение операционных затрат: Автоматизация рутинных операций освобождает персонал для более важных задач, повысив эффективность работы.
2. Увеличение доходов: Персонализированные предложения, основанные на анализе данных о гостях, приведут к росту заполняемости номеров и средней стоимости проживания.
3. Повышение качества обслуживания: Лучшее понимание потребностей гостей обеспечит более высокий уровень удовлетворенности и лояльности.
4. Оптимизация маркетинговых стратегий: Анализ данных позволит повысить эффективность рекламных кампаний и рациональнее использовать бюджет [8].

2.2 Требования к функциям, выполняемой системой

Исходя из анализа существующих программных решений в области гостиничного бизнеса и потребностей пользователей, функциональные задачи разрабатываемой системы учета и анализа данных о гостях отеля включают следующие ключевые требования:

1. Управление данными о гостях: Система должна обеспечивать безопасный и контролируемый доступ к информации о гостях (ФИО, контакты, история посещений, предпочтения), с возможностью добавления, редактирования и удаления данных.

2. Система бронирования: Автоматизированное создание, изменение и отмена бронирований с проверкой доступности номеров и предложением альтернатив.
3. Интеграция: Возможность интеграции с бухгалтерскими системами и системами управления доходами для комплексного управления отелем.
4. Управление услугами: Добавление, изменение и удаление информации об услугах отеля (SPA, экскурсии и т. д.).
5. Пользовательский интерфейс: интуитивно понятный и настраиваемый интерфейс для пользователей с различным уровнем подготовки.

Для разработки и создания программы необходимо провести анализ выбранной среды разработки, составить диаграмму ER-диаграмму (см. ПРИЛОЖЕНИЕ 1, рисунок 1) и диаграмму сценариев (см. ПРИЛОЖЕНИЕ 1, рисунок 2). Кроме того, необходимо разработать пользовательский интерфейс.

Выделим основные объекты и представим их описание в «Таблица 2»
Таблица 2 – Объекты предметной области

	Наименование объекта	Краткое описание
1	Гости (Guests)	Посетителя отеля, которые забронировали или заехали в номер.
2	Администратор (Admins)	Сотрудник отеля, выполняющий различные функции: регистрация гостей, учет базы данных, формирование заказов услуг.
3	Номера (Rooms)	Комнаты в отеле, предлагаемые для проживания. Бронируются гостями, используются для проживания, убираются персоналом.
4	Бронь (Booking)	Запись о бронировании номера. Создается гостем, подтверждается/отменяется персоналом.
5	Услуги (Services)	Предлагаемые отелем услуги (завтраки, уборка номера, Wi-Fi, прачечная и т. д.). Заказываются гостями, оказываются персоналом.
6	Счета (Bills)	Финансовый документ, выставленный гостю. Создается персоналом, оплачивается гостем.

С последующими разборами каждой таблицы можно ознакомиться в (ПРИЛОЖЕНИЕ 1).

После описания объектов предметной области приведем отбор атрибутов для каждого объекта: Сущность «Гости» состоит из 9 атрибутов. В ней описаны личные данные посетителя (ПРИЛОЖЕНИЕ 1, таблица 3).

Сущность «Персонал» состоит из 9 атрибутов. Таблица содержит данные персонала, исполняемая должность (ПРИЛОЖЕНИЕ 1, таблица 4).

Таблица сущности «Номера» состоит из 5 атрибутов. Она содержит тип, номер, стоимость и доступность номера (ПРИЛОЖЕНИЕ 1, таблица 5).

Сущность «Бронь» состоит из 7 атрибутов. Данная таблица содержит: дату заезда и выезда, статус бронирования, дополнительные заметки, внешний ключ для связи с таблицей «Гости» и с таблицей «Бронирование» (ПРИЛОЖЕНИЕ 1, таблица 6).

Таблица сущности «Услуги» состоит из 3 атрибутов. Она содержит данные о названии услуги и ее стоимости. Будет приведен общий прайс для гостей отеля (ПРИЛОЖЕНИЕ 1, таблица 7).

Сущность «Счета» состоит из 6 атрибутов. Данная таблица содержит данные о дате восстановления счета, сумме, статусе оплаты, способе оплаты, внешний ключ связи с таблицей «Бронь» (ПРИЛОЖЕНИЕ 1, таблица 8).

После создания всех таблиц сущностей, установим связи между ними:

Связь Гости – Бронь

У одного гостя, может быть, несколько забронированных номеров. Связь один ко многим (ПРИЛОЖЕНИЕ 1, рисунок 3).

Связь Номер – Бронь

Один номер может быть забронирован несколькими гостями, но в каждом бронировании участвует только один номер. Связь один ко многим (ПРИЛОЖЕНИЕ 1, рисунок 4).

Связь Бронирование – Заказы услуг

Одно бронирование может включать несколько заказов услуг, но каждый заказ услуг привязан к одному бронированию. Связь один ко многим (ПРИЛОЖЕНИЕ 1, рисунок 5).

Связь Заказы услуг – Услуги

Один заказ связан с одной конкретной услугой, но одна услуга может быть заказана в разных заказах. Связь один ко многим (ПРИЛОЖЕНИЕ 1, рисунок 6).

Связь Персонал – Заказы услуг

Один сотрудник может выполнить один заказ, каждая услуга привязана ко многим сотрудникам. Связь один ко многим (ПРИЛОЖЕНИЕ 1, рисунок 7)

Связь Бронирование – Счета

Каждое бронирование связано с одним счетом, которое в свою очередь берет информацию из «Заказы услуг», «Бронирование». Связь один ко многим (ПРИЛОЖЕНИЕ 1, рисунок 8).

2.3 План тестирования

Цель тестирования программного модуля «Система учета и анализа данных о гостях отеля» заключается в обеспечении подтверждения его качества и соответствия заранее определенным функциональным и нефункциональным требованиям. Тестирование направлено на выявление ошибок, оценку производительности и оценку удобства использования.

Задачи тестирования включают:

- Проверить функциональность модуля: убедиться, что все функции работают в соответствии с требованиями.
- Проверить надежность и стабильность: убедиться, что система стабильно работает при различных нагрузках и сценариях использования.
- Проверить безопасность данных: убедиться, что данные пользователей защищены от несанкционированного доступа.
- Проверить удобство использования: убедиться, что система проста в использовании и понятна для пользователя.

Тестирование системы включает 10 объектов:

Создание/авторизацию пользователей, регистрацию гостей, бронирование/отмену бронирования, заказ/оплату услуг, анализ данных о гостях, отчетность и безопасность. Для каждого объекта будут разработаны тест-кейсы с указанием ожидаемых и фактических результатов. В тестировании будут использованы методы функционального, нагрузочного, безопасности и юзабилити тестирования.

Таким образом, программный модуль «Система учета и анализа данных о гостях отеля» удовлетворяет потребности гостей и персонала, автоматизируя процессы, такие как бронирование номеров, регистрация заезда/выезда, управление финансами и обслуживание клиентов. Реализация модуля снижает рутинную работу персонала, повышает качество обслуживания гостей и оптимизирует управление отелем.

3 РЕАЛИЗАЦИЯ ПРОЕКТНОЙ СИСТЕМЫ

3.1 Описание среды разработки

Для реализации проекта «Система учета и анализа данных о гостях отеля» понадобится среда разработки PyCharm с установленным интерпретатором Python, СУБД SQLite3 (или другая, в зависимости от масштаба проекта), библиотеки Python sqlite3, re, tkinter, pyperclip для работы с базами данных, обработкой текстов, созданием графического интерфейса и буфером обмена соответственно, а также навыки работы с языком SQL и стандартными инструментами для работы с документами, например, Word.

3.1.1 Выбор и описание программных инструментов

Выбор программных инструментов для проекта «Система учета и анализа данных о гостях отеля» основан на критериях эффективности, доступности, надежности и интеграции.

1. Язык программирования: Python 3.10+ выбран за его популярность, богатые библиотеки, простоту и активное сообщество, что обеспечивает гибкость для разработки как серверной, так и клиентской частей. [11].
2. Система управления базами данных (СУБД): используется SQLite 3 за простоту развертывания, подходящая для проекта среднего масштаба с возможностью перехода на PostgreSQL или MySQL. [15].
3. Выбрана среда разработки PyCharm Professional Edition (или Community) за ее богатый функционал, включая отладку, автодополнение кода и интеграцию с Git, что повышает производительность разработки [10].
4. Библиотеки Python:
 - sqlite3: Стандартная библиотека Python для взаимодействия с базой данных SQLite. Обеспечивает все необходимые функции для работы с базой данных в рамках проекта.
 - re: Библиотека регулярных выражений, необходимая для валидации данных и обработки текстовых строк [14].

- tkinter: Стандартная библиотека для создания графического пользовательского интерфейса. Выбор tkinter обусловлен его простотой и широким распространением [12].
- pyperclip: Библиотека для удобной работы с буфером обмена операционной системы. Упрощает копирование и вставку данных пользователем.
- openpyxl: Библиотека для работы с файлами Excel (.xlsx) в Python. Позволяет создавать, редактировать и читать данные из таблиц Excel, а также управлять форматированием ячеек, стилями и формулами. [13]

Выбор инструментов обеспечивает оптимальное соотношение производительности, доступности и простоты использования, что важно для успешного проекта. Все инструменты широко распространены и хорошо документированы.

3.1.2 Обоснование выбора инструментария по разработке

Выбор PyCharm и Python обусловлен их высокой производительностью, удобством использования, широкими возможностями и поддержкой большого активного сообщества.

3.2 Разработка программного модуля

Разработка программного модуля «Система учета и анализа данных о гостях отеля» включает в себя следующие этапы:

- Разработка базы данных: Создание таблиц в СУБД (PostgreSQL или SQLite), определение типов данных, связей между таблицами и ограничений.
- Разработка логики приложения: Написание кода на Python для реализации функций системы (создание пользователей, регистрация гостей, бронирование номеров, отмена бронирования, заказ услуг, оплата услуг, анализ данных о гостях, отчетность).
- Разработка графического интерфейса: Создание GUI с помощью Tkinter или PyQt для удобного взаимодействия пользователей с системой.

- Тестирование: Проведение комплексного тестирования всех функций системы (функциональное, нагрузочное, безопасности и юзабилити-тестирование).
- Документирование: Создание документации к системе, включая техническое задание, архитектурную диаграмму, схему базы данных, описание функций и т. д.

3.2.1 Реализация пользовательского интерфейса

Пользовательский интерфейс спроектирован с учетом принципов удобства использования и эффективности. Он состоит из нескольких взаимосвязанных окон, каждое из которых выполняет определенную функцию:

Окно авторизации: представляет собой форму для входа в систему с полями для ввода логина и пароля. После успешной аутентификации пользователя (проверки логина и пароля с данными из базы данных) открывается главное окно приложения (ПРИЛОЖЕНИЕ 3, рисунок 3).

Главное меню: представляет собой многооконное приложение с использованием вкладок (ttk. Notebook) (ПРИЛОЖЕНИЕ 3, рисунок 3.1).

Каждая вкладка предоставляет доступ к специфическому разделу функциональности системы:

- Управление гостями: Добавление, редактирование и удаление информации о гостях отеля (ПРИЛОЖЕНИЕ 3, рисунок 3.2).
- Управление бронями: Создание, редактирование и отмена бронирований номеров (ПРИЛОЖЕНИЕ 3, рисунок 3.3).
- Управление номерами: Просмотр информации о номерах и изменение их статуса (доступность/занятость) (ПРИЛОЖЕНИЕ 3, рисунок 3.4).
- Управление счетами: Просмотр и управление счетами гостей, включая информацию об оплате. Анализ итоговой суммы за все оплаченные брони (ПРИЛОЖЕНИЕ 3, рисунок 3.5).
- Управление услугами. Просмотр и управление услугами, изменение их статуса (ПРИЛОЖЕНИЕ 3, рисунок 3.6).

3.2.2 Описание кодом функциональных узлов модуля

1. «Войти». Кнопка авторизации администратор. После ввода логина и пароля, администратор нажимает на нее для проверки введенных данных с базой. При успешной аутентификации открывается соответствующий интерфейс (ПРИЛОЖЕНИЕ 3, рисунок 3.7).
2. «Выйти». Завершает текущую сессию пользователя, закрывает все активные окна и возвращает интерфейс к стартовому окну авторизации (ПРИЛОЖЕНИЕ 3, рисунок 3.8).
3. «Гости». Позволяет добавить информацию о новом госте в систему. Содержит дополнительные кнопки «Редактировать», «Удалить», «Копировать», строка «Поиск» (ПРИЛОЖЕНИЕ 3, рисунок 3.9–13).
4. «Номера». Данное окно позволяет управлять номерами, добавлять их, удалять и редактировать. Также есть поиск номеров по любым критериям (ПРИЛОЖЕНИЕ 3, рисунок 3.14–16).
5. «Услуги». Предоставляет список услуг. Можно менять доступность услуги, цену. Добавлять новые услуги, удалять и редактировать услуги (ПРИЛОЖЕНИЕ 3, рисунок 3.17–20).
6. «Счета». В этом окне можно сменить статус на Оплачен/Не оплачен. Также удалить счет и отчет (ПРИЛОЖЕНИЕ 3, рисунок 3.22–24).
7. «Бронирования». Представлены брони клиентов, их номера, дата заезда/выезда и статус. Их можно редактировать, удалять и добавлять новые. В данном окне администратор сразу добавляет выбранные гостем услуги и рассчитать стоимость проживания (ПРИЛОЖЕНИЕ 3, рисунок 3.25–30).

3.2.3 Результат работы и тестирования

Кейс 1. Авторизация. Тестируемая функция: Проверка корректности входа в систему с различными наборами данных. Тестовый набор:

Ввод корректных логина и пароля. Ожидаемый результат: успешный вход в систему и переход на главную страницу (ПРИЛОЖЕНИЕ 2, рисунок 2.1).

Ввод некорректного логина и пароля. Ожидаемый результат: сообщение об ошибке «Неверный логин или пароль!» (ПРИЛОЖЕНИЕ 2, рисунок 2.2).

Ввод пустых полей логина и пароля. Ожидаемый результат: сообщение об ошибке «Неверный логин или пароль!» (ПРИЛОЖЕНИЕ 2, рисунок 2.3).

Ввод логина с пробелами в начале\конце. Ожидаемый результат: сообщение об ошибке «Неверный логин или пароль!» (ПРИЛОЖЕНИЕ 2, рисунок 2.4).

- Результат: Все тесты прошли успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Кейс 2. Главное меню. Гость. Добавить гостя. Тестируемая функция: Проверка корректности добавления данных гостей. Тестовый набор:

Ввод только фамилии и имени. Ожидаемый результат: Новый гость добавлен в таблицу Guests с пустыми значениями для остальных полей. Отображается сообщение об успехе (ПРИЛОЖЕНИЕ 2, рисунок 2.5).

Оставить поля пустыми. Ожидаемый результат: Гость не добавлен. Отображается «Имя и фамилия обязательны для заполнения» (ПРИЛОЖЕНИЕ 2, рисунок 2.6).

Нажать на крестик окна или использование функции `on_close_add_window`. Ожидаемый результат: Окно добавление гостя закрывается без добавления гостя (ПРИЛОЖЕНИЕ 2, рисунок 2.7).

- Результат: все тесты прошли успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Кейс 3. Главное меню. Гость. Редактировать гостя. Тестируемая функция: проверка корректности редактирования данных. Тестовый набор:

Выбрать гостя и изменить все поля на корректные значения. Ожидаемый результат: Данные гостя в БД обновлены. Сообщение «Данные гостя обновлены» (ПРИЛОЖЕНИЕ 2, рисунок 2.8).

Выбрать гостя, изменить фамилию и email. Ожидаемый результат: только указанные поля обновлены в БД. Сообщение «Данные гостя обновлены» (ПРИЛОЖЕНИЕ 2, рисунок 2.9).

Не вносить изменения. Ожидаемый результат: Сообщение «Данные гостя обновлены». БД не изменена (ПРИЛОЖЕНИЕ 2, рисунок 2.10).

Нажать на кнопку «Редактировать гостя» без выбора гостя. Ожидаемый результат: Предупреждение о необходимости выбрать гостя. Сообщение «Выберите гостя для редактирования!» (ПРИЛОЖЕНИЕ 2, рисунок 2.11).

Удалить поля Имя и Фамилия. Ожидаемый результат: Сообщение «Имя и фамилия обязательны для сохранения!» (ПРИЛОЖЕНИЕ 2, рисунок 2.12).

Результат: все тесты прошли успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Кейс 4. Главное меню. Гость. Удалить гостя. Тестируемая функция: проверка корректности удаления данных гостей. Тестовый набор:

Выбрать гостя и нажать на кнопку «Удалить гостя». Ожидаемый результат: Предупреждение «Вы уверены, что хотите удалить гостя?»; сообщение «Гость удален!» (ПРИЛОЖЕНИЕ 2, рисунок 2.13).

Не выбирать гостя и нажать «Удалить гостя». Ожидаемый результат: Сообщение «Выберите гостя для удаления» (ПРИЛОЖЕНИЕ 2, рисунок 2.14).

Создать бронь для гостя, удалить гостя. Ожидаемый результат: Предупреждение. Сообщение «Невозможно удалить гостя, так как у него есть брони!» (ПРИЛОЖЕНИЕ 2, рисунок 2.15).

- Результат: все тесты прошли успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Кейс 5. Главное меню. Номер. Добавить номер. Тестируемая функция: проверка корректности добавления данных гостей. Тестовый набор:

Заполнение всех полей корректными данными. Ожидаемый результат: Номер добавлен (ПРИЛОЖЕНИЕ 2, рисунок 2.16).

Частичное заполнения полей. Ожидаемый результат: «Все поля обязательны для заполнения!» (ПРИЛОЖЕНИЕ 2, рисунок 2.17).

Ввод существующего id номера. Ожидаемый результат: Предупреждение. «Номер уже существует!» (ПРИЛОЖЕНИЕ 2, рисунок 2.18).

Ввод буквами в поле «Цена». Ожидаемый результат: Предупреждение. «Цена должна быть числом!» (ПРИЛОЖЕНИЕ 2, рисунок 2.19).

Ввод буквами в поле «Номер». Ожидаемый результат: Предупреждение. «Номер должен быть числом» (ПРИЛОЖЕНИЕ 2, рисунок 2.20).

- Результат: все тесты прошли успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Кейс 6. Главное меню. Номер. Редактировать. Тестируемая функция: проверка корректности редактирования данных гостей. Тестовый набор:

Не выбирать номер и нажать на кнопку «Редактировать». Ожидаемый результат: Предупреждение. «Выберите номер для редактирования!» (ПРИЛОЖЕНИЕ 2, рисунок 2.21).

Выделить номер и нажать на кнопку «Редактировать». Ожидаемый результат: Данные обновлены в БД. Сообщение «Данные номера обновлены!» (ПРИЛОЖЕНИЕ 2, рисунок 2.22).

- Результат: все тесты прошли успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Кейс 7. Главное меню. Бронирования. Добавить бронь. Тестируемая функция: проверка корректности добавления брони. Тестовый набор:

Все поля пустые. Ожидаемый результат: Предупреждение. Предупреждение. «Выберите гостя и номер комнаты!» (ПРИЛОЖЕНИЕ 2, рисунок 2.23).

Частичное заполнения полей. Ожидаемый результат: «Все поля обязательны для заполнения!» (ПРИЛОЖЕНИЕ 2, рисунок 2.24).

Ввод буквами в поле «Дата заезда и выезда». Ожидаемый результат: Данные автоматически не вводятся (ПРИЛОЖЕНИЕ 2, рисунок 2.25).

Ввели дату не полностью. Ожидаемый результат: Предупреждение. «Дата должна быть в формате ДД.ММ.ГГГГ!» (ПРИЛОЖЕНИЕ 2, рисунок 2.26).

Не ввели гостя и номер комнаты. Ожидаемый результат: Предупреждение. «Выберите гостя и номер комнаты!» (ПРИЛОЖЕНИЕ 2, рисунок 2.27).

- Результат: все тесты прошли успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Кейс 8. Главное меню. Бронирование. Редактировать. Тестируемая функция: проверка корректности редактирования ввода данных гостей. Тестовый набор:

Не выбирать бронь и нажать на кнопку «Редактировать». Ожидаемый результат: Предупреждение. «Выберите бронь для редактирования!» (ПРИЛОЖЕНИЕ 2, рисунок 2.28).

Ввели дату не полностью. Ожидаемый результат: Предупреждение. «Дата должна быть в формате ДД.ММ.ГГГГ!» (ПРИЛОЖЕНИЕ 2, рисунок 2.29).

Не ввели гостя и номер комнаты. Ожидаемый результат: Предупреждение. «Выберите гостя и номер комнаты!» (ПРИЛОЖЕНИЕ 2, рисунок 2.30).

- Результат: все тесты прошли успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Кейс 9. Главное меню. Бронирования. Услуги. Тестируемая функция: проверка корректности добавления услуг для данных. Тестовый набор:

Выбрать бронь и нажать на кнопку «Услуги». Ожидаемый результат: «Изменения успешно сохранены!» (ПРИЛОЖЕНИЕ 2, рисунок 2.31).

Не выбрать бронь и нажать на кнопку «Услуги». Ожидаемый результат: Предупреждение. «Выберите бронь для управления услугами!»

- Результат: все тесты прошли успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Кейс 10. Главное меню. Бронирования. Рассчитать. Тестируемая функция: проверка корректности удаления данных гостей. Тестовый набор:

Не выбирать бронь и нажать на кнопку «Рассчитать». Ожидаемый результат: Предупреждение. «Выберите бронь для расчета!»

Выбрать бронь и нажать на кнопку «Рассчитать». Ожидаемый результат: «Счет успешно создан! Общая стоимость: [Итоговая стоимость]!»

Ввести в поле «Количество дней проживания» буквенные значения. Ожидаемый результат: Предупреждение. «Некорректный данные. Введите положительное число!» (ПРИЛОЖЕНИЕ 2, рисунок 2.32).

Ввести положительные данные. Ожидаемый результат: «Счет успешно создан! Общая стоимость: [Итоговая стоимость]!» (ПРИЛОЖЕНИЕ 2, рисунок 2.33).

Кейс 11. Главное меню. Счета. Удалить. Тестируемая функция: проверка корректности удаления данных гостей. Тестовый набор:

Не выбрать бронь и нажать на кнопку «Удалить». Ожидаемый результат: Предупреждение. «Выберите бронь для удаления!»

Выбрать бронь и нажать на кнопку «Удалить». Ожидаемый результат: Удален (ПРИЛОЖЕНИЕ 2, рисунок 2.34).

- Результат: все тесты прошли успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Кейс 12. Главное меню. Счета. Оплачен. Тестируемая функция: проверка корректности удаления данных гостей. Тестовый набор:

Нажать на кнопку «Оплачен». Ожидаемый результат: Статус оплаты изменен на «Оплачен!»

- Результат: тест прошел проверку успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Кейс 13. Главное меню. Счета. Создать отчет. Тестируемая функция: проверка создания документа-отчет об итоговой сумме всех броней Excel. Тестовый набор:

Нажать на кнопку «создать отчет». Ожидаемый результат: Отчет успешно создан: Счета_отчет.xlsx (ПРИЛОЖЕНИЕ 2, рисунок 2.35).

- Результат: тест прошел проверку успешно. Система корректно обрабатывает различные сценарии ввода данных и выдает соответствующие сообщения об ошибках.

Таким образом, в третьей главе курсовой работы был описан процесс разработки программного модуля системы управления отелем. Был осуществлен выбор инструментов разработки, реализован пользовательский интерфейс и ключевые функции системы, такие как управление бронированием, регистрация гостей, управление номерами и финансовые операции. Проведенное тестирование показало, что система работает стабильно и соответствует заявленным требованиям, обеспечивая удобство использования для персонала отеля и гостей. С разработанным продуктом можно ознакомиться там [9].

ЗАКЛЮЧЕНИЕ

Цель курсовой работы — разработка программного модуля «Система учета и анализа данных о гостях отеля» — успешно достигнута. Созданное решение направлено на автоматизацию ключевых процессов гостиничного бизнеса, повышение качества обслуживания гостей и оптимизацию работы персонала.

В ходе выполнения работы были решены следующие задачи:

- Проведен анализ существующих решений для учета и управления данными, что позволило определить основные функциональные требования к системе.
- Выполнен выбор инструментов и технологий разработки.
- Реализован удобный и интуитивно понятный пользовательский интерфейс.
- Созданы ключевые функции системы: бронирование номеров, регистрация гостей, управление номерами и финансовые операции.

Тестирование системы показало ее стабильность и соответствие поставленным требованиям. Выявленные ошибки устранены, что гарантирует надежность работы.

Программный модуль обладает рядом преимуществ: автоматизирует рутинные операции, улучшает качество обслуживания гостей, ускоряет обработку запросов и снижает нагрузку на персонал. Кроме того, решение оптимизирует управление ресурсами отеля, что положительно сказывается на экономической эффективности бизнеса. Гибкость системы позволяет адаптировать ее под конкретные требования гостиницы.

Таким образом, разработанный программный модуль представляет собой современный инструмент для управления гостиничным бизнесом, удовлетворяющий потребности персонала и гостей, а также обеспечивающий стабильность, удобство использования и экономическую выгоду.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Стандарты и законодательные материалы

1. Федеральный закон от 27.02.2006 №152-ФЗ «О персональных данных»
2. Федеральный закон от 27.02.2006 №149-ФЗ «Об информации, информационных технологиях и защите информации»
3. Федеральный закон от 26.07.2017 №187-ФЗ «О безопасности критической информационной инфраструктуры Российской Федерации»
4. ГОСТ Р ИСО/МЭК 12207 "Информационные технологии. Процессы жизненного цикла программных средств»
5. ГОСТ 19.005–85 «Р-схемы алгоритмов и программ. Обозначения условные графические правила выполнения»
6. ГОСТ 19.105–78 «Общие требования к программным документам»
7. ГОСТ 19.504–79 «Руководство программиста. Требования к содержанию и оформлению»

Монографии

8. Бай, Т. В., Гостиничный менеджмент. Управление гостиничным предприятием. Южноуральск: Русайнс, 2023
9. Программный модуль «Система учета и анализа данных о гостях отеля»: <https://github.com/qanme/GuestAccountingSystem>

Интернет-источники

10. Описание PyCharm: <https://www.jetbrains.com/pycharm/features/>
11. Описание Python: https://www.nic.ru/help/osnovy-yazyka-programmirovaniya-python_11662.html
12. Описание Tkinter: <https://metanit.com/python/tkinter/1.1.php>
13. Описание openpyxl: <https://openpyxl.readthedocs.io/en/stable/>
14. Описание библиотеки Re: <https://docs.python.org/3/library/re.html>
15. Описание SQLite: <https://www.sqlite.org/about.html>

ПРИЛОЖЕНИЕ 1 Диаграмма прецедентов

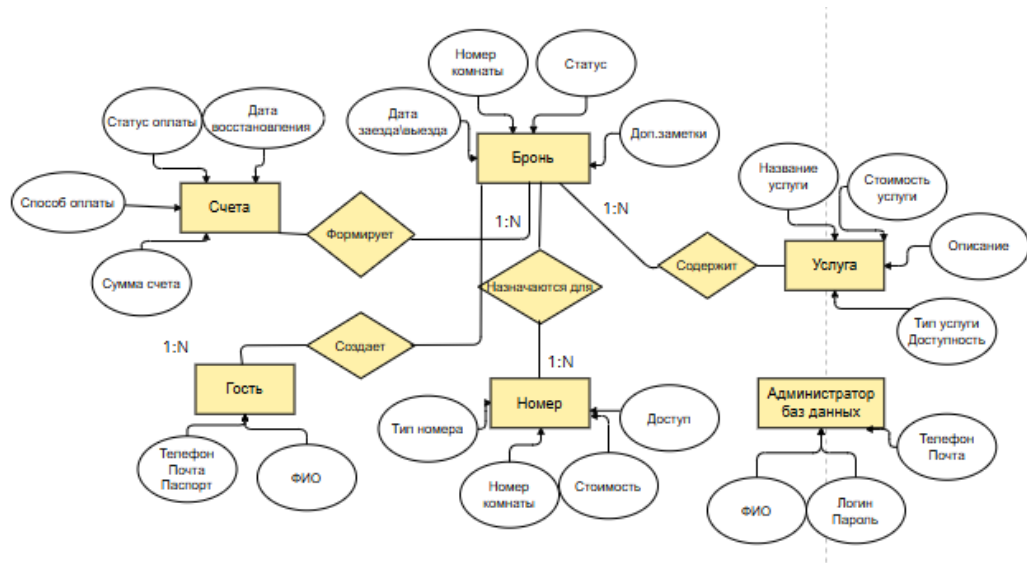


Рисунок 1 – ER-диаграмма базы данных «Систему учета и анализа данных о гостях отеля»

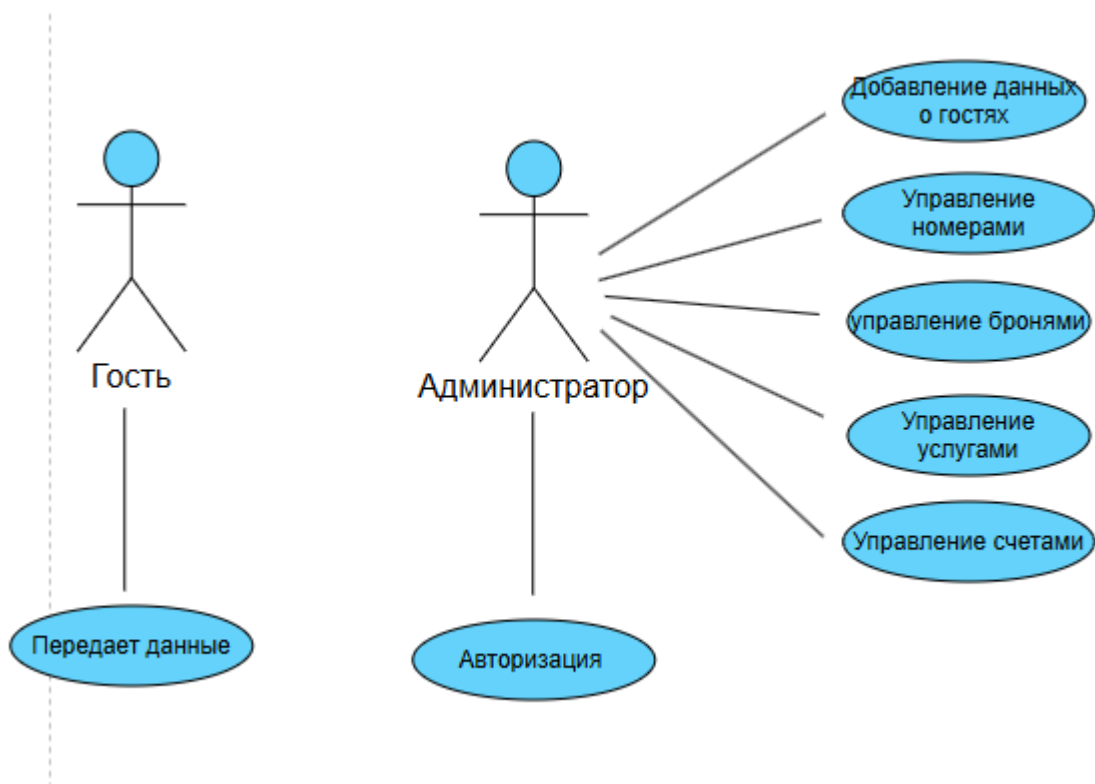


Рисунок 2 – Диаграмма прецедентов

Таблица 3 – Сущность «Гости»

Атрибут	Тип	Описание
Guests_ID	INT PRIMARY KEY	Уникальный идентификатор гостя
Last_name	VARCHAR (100)	Фамилия гостя
First_name	VARCHAR (100)	Имя гостя
Middle_name	VARCHAR (100)	Отчество гостя
Phone	VARCHAR (15)	Номер телефона
Email	VARCHAR (255)	Почта
Passport	VARCHAR (20)	Паспорт

Таблица 4 – Сущность «Администратор»

Атрибут	Тип	Описание
Admins_ID	INT PRIMARY KEY	Уникальный идентификатор администратора
Last_name	VARCHAR (100)	Фамилия персонала
First_name	VARCHAR (100)	Имя персонала
Middle_name	VARCHAR (100)	Отчество персонала
Phone	VARCHAR (15)	Номер телефона
Email	VARCHAR (255)	Почта
Login	VARCHAR (50)	Логин
Password	VARCHAR (255)	Пароль

Таблица 5 – Сущность «Номера»

Атрибут	Тип	Описание
Room_number	INTEGER	Уникальный идентификатор номера, номер комнаты
Room_type	VARCHAR (50)	Тип номера
Price	REAL	Стоимость
Availability	BOOLEAN	Доступен ли номер

Таблица 6 – Сущность «Бронь»

Атрибут	Тип	Описание
Booking_ID	INT PRIMARY KEY	Уникальный идентификатор брони
Guests_ID	INT FOREIGN KEY REFERENCES	Идентификатор гостя, внешний ключ
Room_ID	INT FOREIGN KEY REFERENCES	Идентификатор номера, внешний ключ
Checking_date	DATE	Дата заезда
Checkout_date	DATE	Дата выезда
Status	VARCHAR (50)	Статус брони (например, «Подтверждено», «Отменено», «Завершено»)
Notes	VARCHAR (500)	Дополнительные заметки

Таблица 7 – Сущность «Услуги»

Атрибут	Тип	Описание
Services_ID	INT PRIMARY KEY	Уникальный идентификатор услуги
Services_name	VARCHAR (100)	Название услуги
Price	REAL	Стоимость услуги
Description	TEXT	
Service_type	VARCHAR (50)	
Availability	BOOLEAN	

Таблица 8 – Сущность «Счета»

Атрибут	Тип	Описание
Bill_ID	INT PRIMARY KEY	Уникальный идентификатор услуги
Booking_ID	INT FOREIGN KEY REFERENCES	Идентификатор брони, внешний ключ
Bill_date	DATE	Дата восстановления счета
Total_amount	RAEL	Сумма счета
Payment_status	VARCHAR (50)	Статус оплаты

Продолжение таблицы 8

Атрибут	Тип	Описание
Payment_method	VARCHAR (50)	Способ оплаты

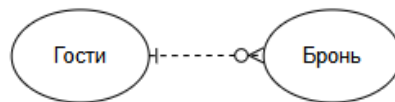


Рисунок 3 – Связь «Гости – Бронь»

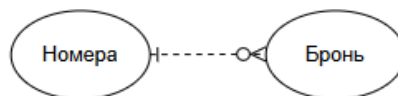


Рисунок 4 – Связь «Номера – Бронь»

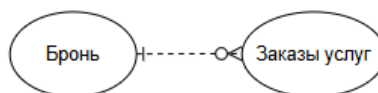


Рисунок 5 – Связь «Бронь – Заказы услуг»

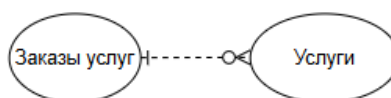


Рисунок 6 - Связь «Заказы услуг – Услуги»

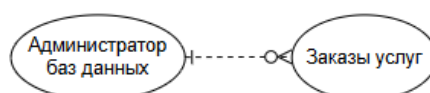


Рисунок 7 – Связь «Персонал – Заказы услуг»

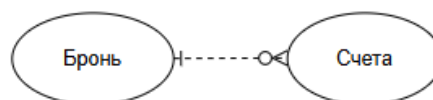


Рисунок 8 - Связь «Бронь – Счета»

ПРИЛОЖЕНИЕ 2 План-тест

Таблица 2.1 – План-тест

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
1	Авторизация	Login	Валидный логин	Успешная авторизация	Успех

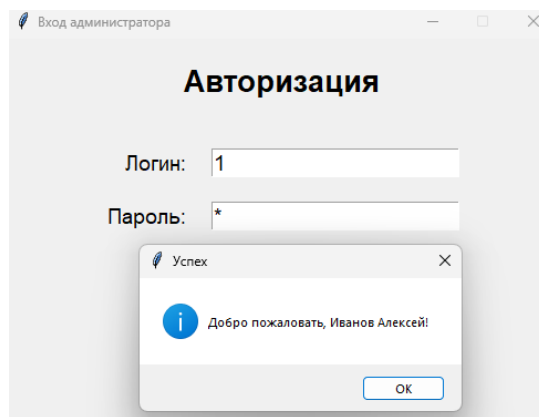


Рисунок 2.1

			Не валидный логин	Сообщение об ошибке «Неверный логин или пароль»	Успех
--	--	--	-------------------	---	-------

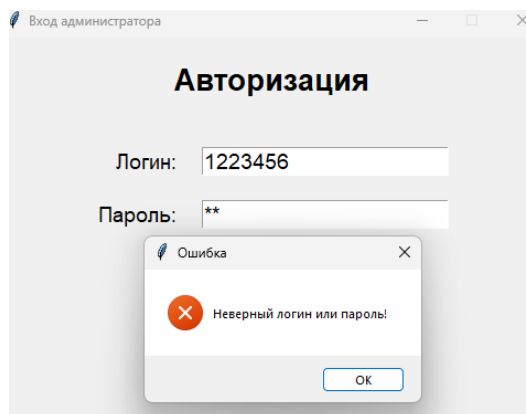
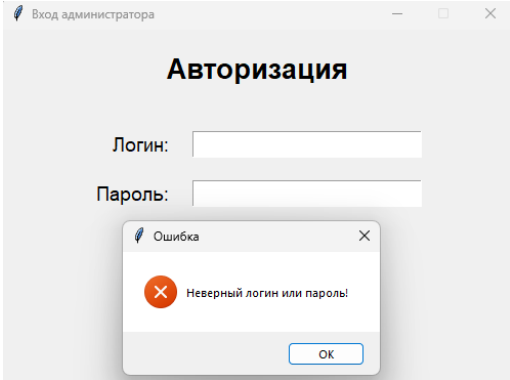
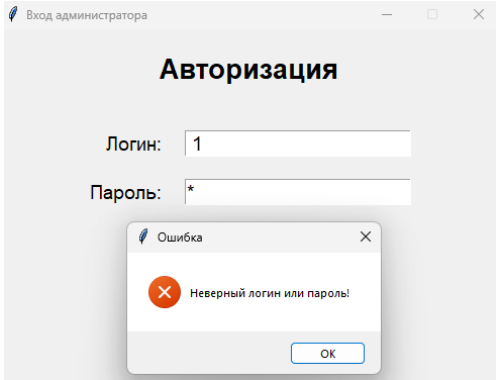


Рисунок 2.2

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
			Пустое поле	Сообщение об ошибке «Неверный логин или пароль»	Успех
 <p>Рисунок 2.3</p>					
			Логин с пробелами в начале/конце	Сообщение об ошибке «Неверный логин или пароль»	Успех
 <p>Рисунок 2.4</p>					
		Password	Валидный логин	Успешная авторизация	Успех

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
			Не валидный логин	Сообщение об ошибке «Неверный логин или пароль»	Успех
			Пустое поле	Сообщение об ошибке «Неверный логин или пароль»	Успех
2	Главное меню. Гости	Добавить гостя	Заполнение всех полей корректными данными	Гость добавлен	Успех
		Добавить гостя	Заполнены только фамилия и имя	Новый гость добавляется в таблицу Guests с пустыми значениями для остальных полей. Отображается сообщение об успехе.	Успех

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
<div data-bbox="521 448 1109 1232" data-label="Image"> </div> <p>Рисунок 2.5</p>					
			Оба поля пустые	Гость не добавлен. Отображается «Имя и фамилия обязательны для заполнения»	Успех

Продолжение таблицы 2.1

№	Наименование функциональ-сти	Наименова-ние поля	Тестовый набор	Ожидаемый результат	Результат те-стирования
---	------------------------------	--------------------	----------------	---------------------	-------------------------

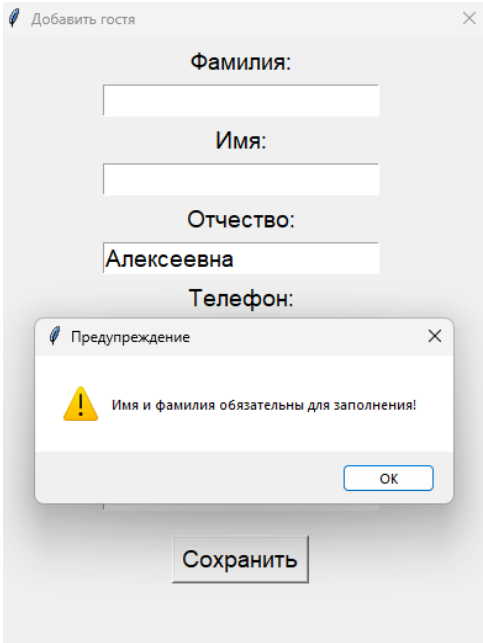


Рисунок 2.6

		Все поля	Нажатие на крестик окна или использование функции on_close_add_window	Окно добавление гостя закрывается без добавления гостя	Успех
--	--	----------	---	--	-------

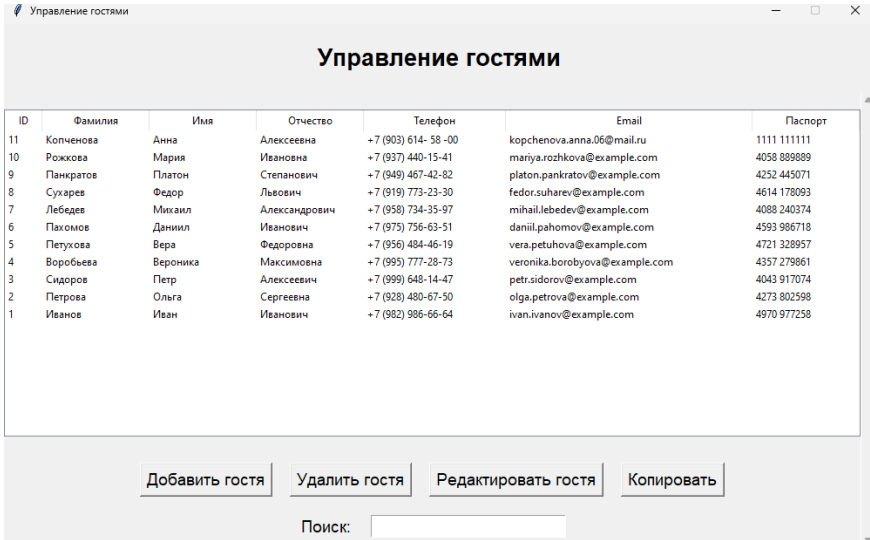
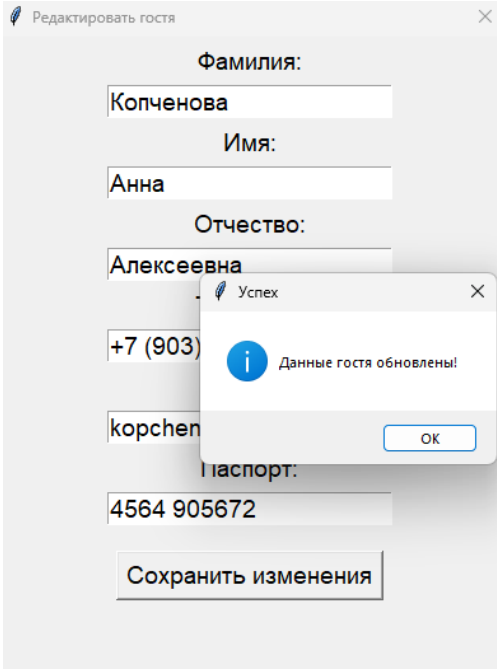
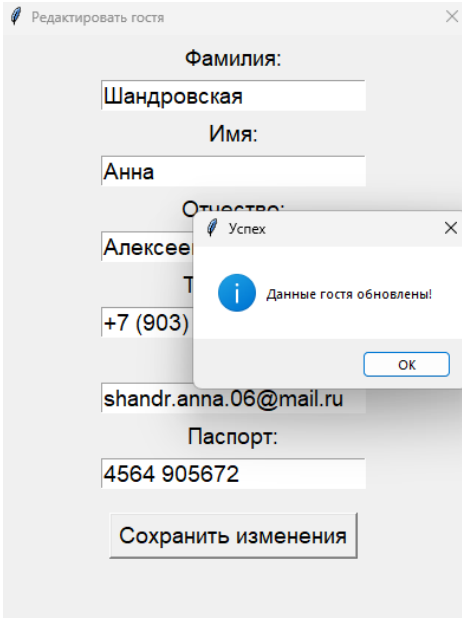
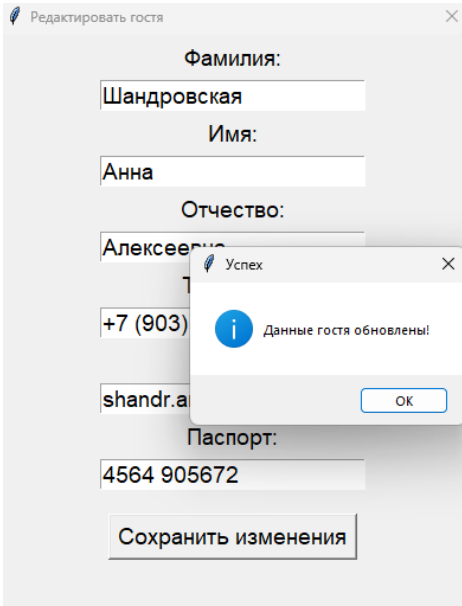


Рисунок 2.7

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
		Редактировать гостя	Выбрать гостя; изменить все поля на корректные значения	Данные гостя в БД обновлены. Сообщение «Данные гостя обновлены»	Успех
 <p>Рисунок 2.8</p>					
			Выбрать гостя, изменить фамилию и email.	Только указанные поля обновлены в БД. Сообщение «Данные гостя обновлены»	Успех

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
<div></div> <p>Рисунок 2.9</p>					
			Не вносить изменения	Сообщение «Данные гостя обновлены». БД не изменена	Успех
<div></div> <p>Рисунок 2.10</p>					

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
			Нажатие на кнопку «Редактировать гостя» без выбора гостя	Предупреждение о необходимости выбрать гостя. Сообщение «Выберите гостя для редактирования!»	Успех

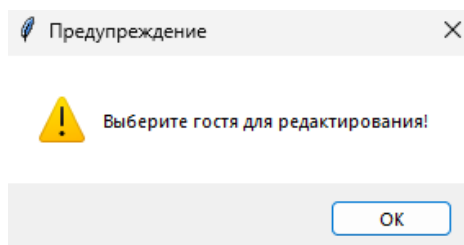


Рисунок 2.11

			Удалить поля Имя и Фамилия	Сообщение «Имя и фамилия обязательны для сохранения!»	Успех
--	--	--	----------------------------	---	-------

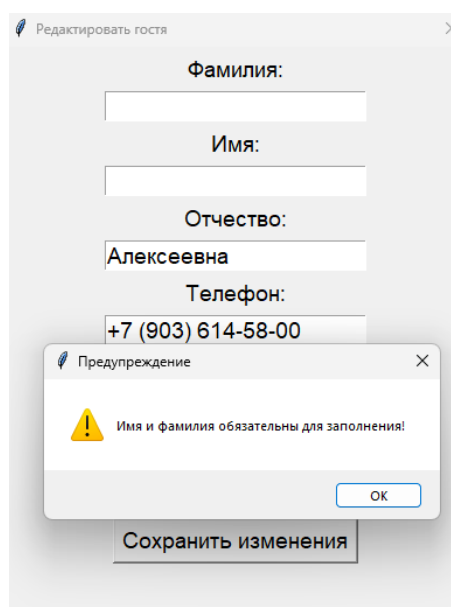
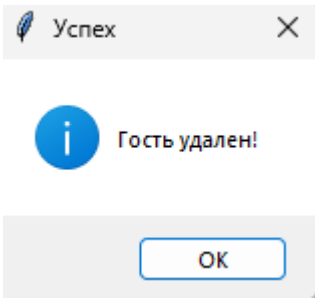
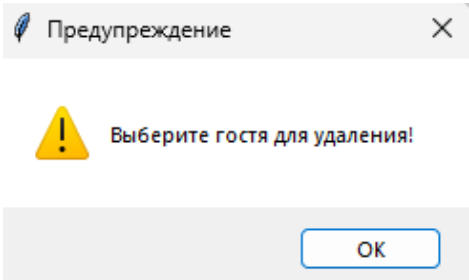


Рисунок 2.12

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
		Удалить гостя	Выбрать гостя и нажать на кнопку «Удалить гостя»	Предупреждение «Вы уверены, что хотите удалить гостя?»; сообщение «Гость удален!»	Успех
 <p>Рисунок 2.13</p>					
			Не выбирать гостя и нажать «Удалить гостя»	Сообщение «Выберите гостя для удаления»	Успех
 <p>Рисунок 2.14</p>					
			Создать бронь для гостя, удалить гостя	Предупреждение. Сообщение «Невозможно удалить гостя, так как у него есть брони!»	Успех

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
---	-------------------------------	-------------------	----------------	---------------------	------------------------

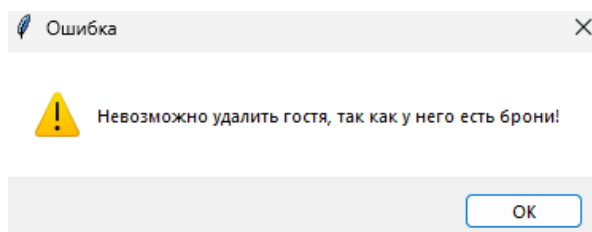


Рисунок 2.15

		Копировать	Выделить гостя, нажать на кнопку «Копировать»	Сообщение «Данные скопированы в буфер обмена!»	Успех
			Нажать на кнопку «Копировать»	Предупреждение. Сообщение «Выберите гостя для копирования данных!»	Успех
3	Главное меню. Номера	Добавить номер	Заполнение всех полей корректными данными	Номер добавлен	Успех

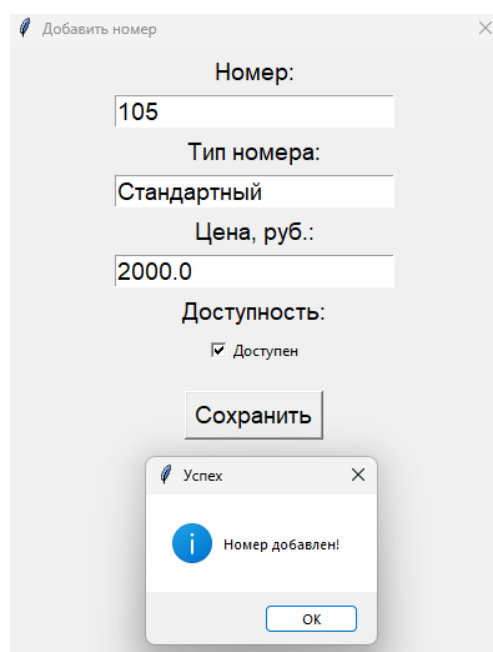
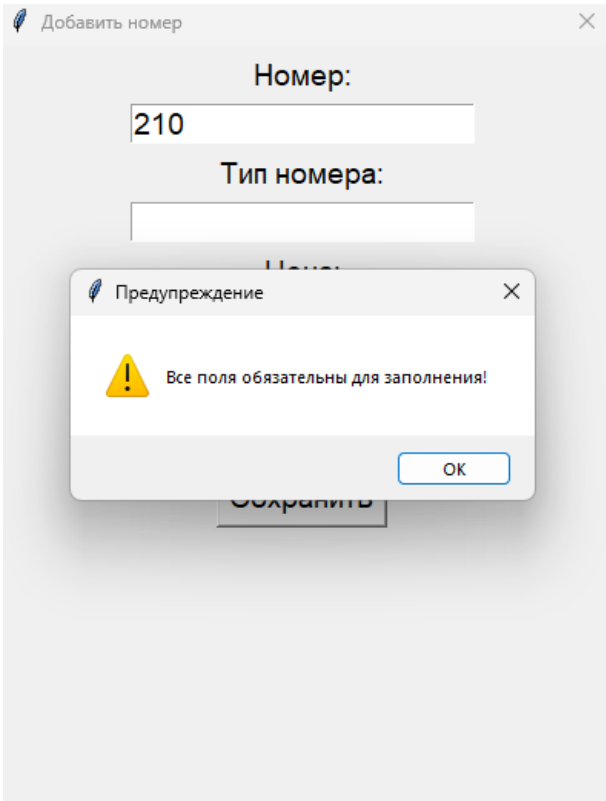


Рисунок 2.16

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
			Все поля пустые	Предупреждение. «Все поля обязательны для заполнения!»	Успех
			Частичное заполнения полей	«Все поля обязательны для заполнения!»	Успех
 <p>Рисунок 2.17</p>					
			Ввод существующего id номера	Предупреждение. «Номер уже существует!»	Успех

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
---	-------------------------------	-------------------	----------------	---------------------	------------------------

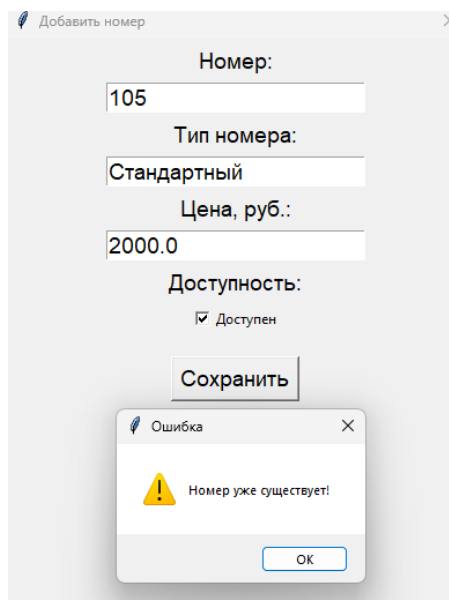


Рисунок 2.18

			Ввод буквами в поле «Цена»	Предупреждение. «Цена должна быть числом!»	Успех
--	--	--	----------------------------	--	-------

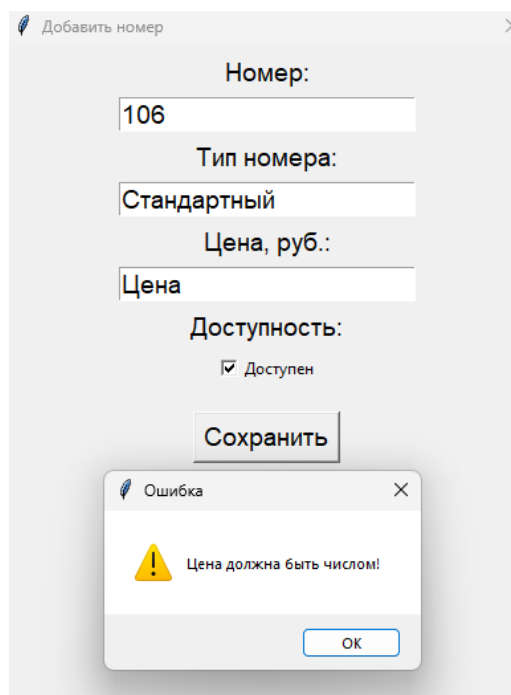
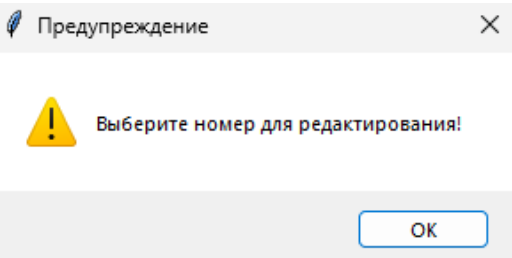


Рисунок 2.19

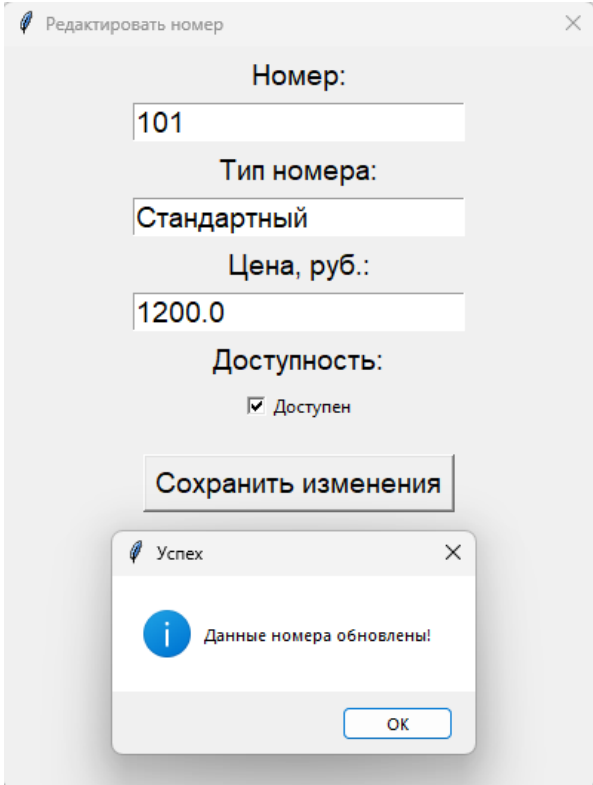
Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
			Ввод буквами в поле «Номер»	Предупреждение. «Номер должен быть числом»	Успех
<div data-bbox="521 609 1149 1453" data-label="Image"> </div> <p>Рисунок 2.20</p>					
		Удалить номер	Не выбирать номер и нажать на кнопку «Удалить номер»	Предупреждение. «Выберите номер для удаления!»	Успех

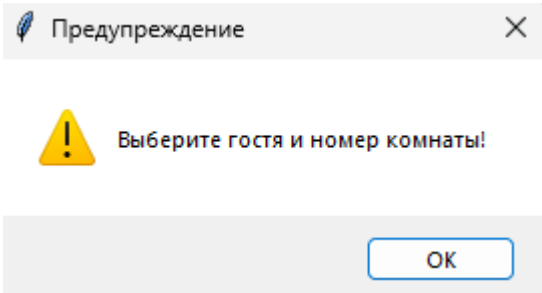
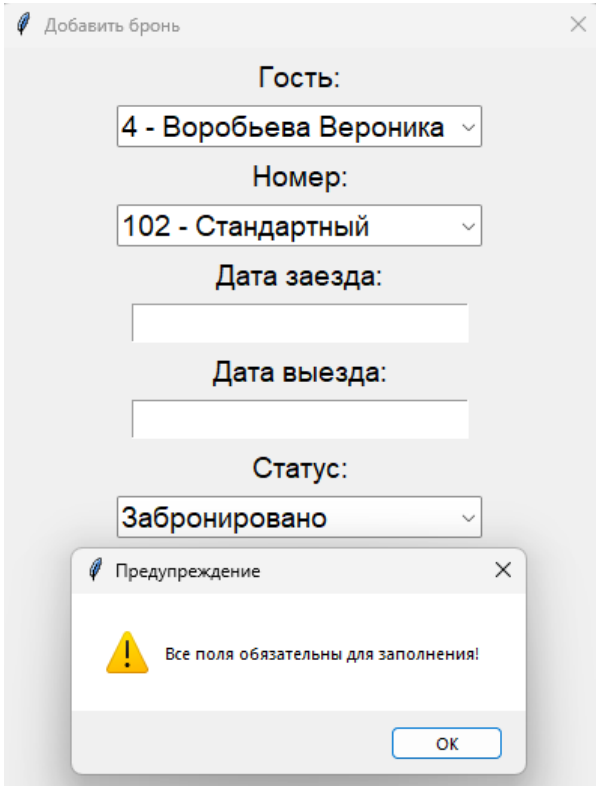
Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
			Выделить номер и нажать на кнопку «Удалить номер»	Данные обновлены в БД. Сообщение «Номер удален!»	Успех
			Оформить бронь на номер и удалить его	Предупреждение. «Невозможно удалить номер, так как на него есть брони!»	Успех
		Редактировать	Не выбирать номер и нажать на кнопку «Редактировать»	Предупреждение. «Выберите номер для редактирования!»	Успех
 <p>Рисунок 2.21</p>					
			Выделить номер и нажать на кнопку «Редактировать»	Данные обновлены в БД. Сообщение «Данные номера обновлены!»	Успех

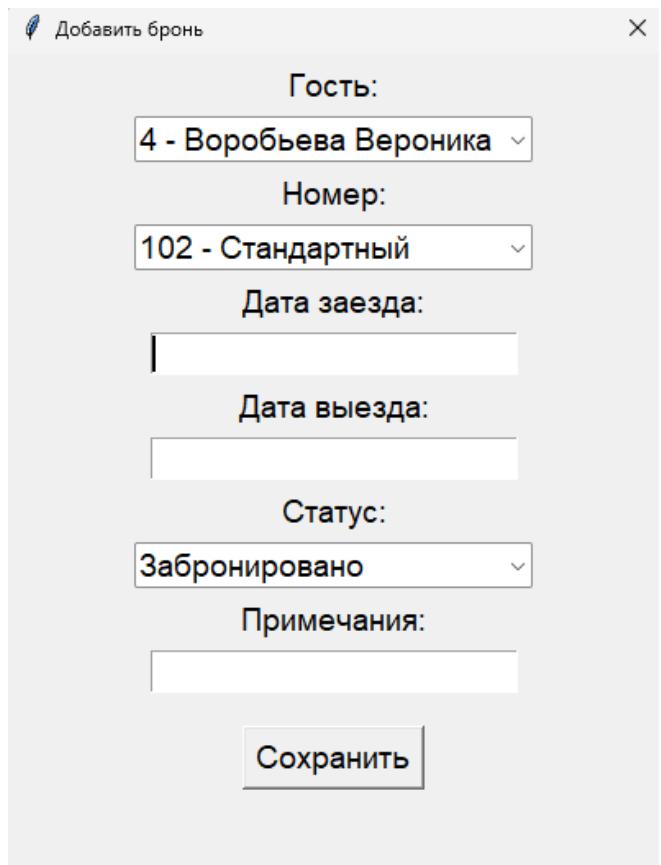
Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
 <p>Рисунок 2.22</p>					
			Ввод существующего id номера	Предупреждение. «Номер уже существует!»	Успех
			Частичное заполнения полей	«Все поля обязательны для заполнения!»	Успех
4	Главное меню. Бронирование	Добавить бронь	Заполнение всех полей корректными данными	Бронь добавлена	Успех
			Все поля пустые	Предупреждение. «Выберите гостя и номер комнаты!»	Успех

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
 <p>Рисунок 2.23</p>					
			Частичное заполнение полей	«Все поля обязательны для заполнения!»	Успех
 <p>Рисунок 2.24</p>					
			Ввод буквами в поле «Дата заезда и выезда»	Данные автоматически не вводятся	Успех

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
<div data-bbox="509 470 1173 1332" data-label="Form">  </div> <p>Рисунок 2.25</p>					
			Ввели дату не полностью	Предупреждение. «Дата должна быть в формате ДД.ММ.ГГГГ!»	Успех

Продолжение таблицы 2.1



№	Наименование функциональ- ности	Наименова- ние поля	Тесто- вый набор	Ожидаемый результат	Результат те- стирования
				<div><div><div>Добавить бронь</div><div><div>Гость:</div><div>4 - Воробьева Вероника</div></div><div><div>Номер:</div><div>102 - Стандартный</div></div><div><div>Дата заезда:</div><div>12.12.202</div></div><div><div>Дата выезда:</div><div>14.12.2024</div></div><div><div>Статус:</div></div></div><div><div>Предупреждение</div><div><div>!</div><div>Дата должна быть в формате ДД.ММ.ГГГГ!</div></div><div>ОК</div></div></div>	
			Не ввели гостя и номер комнаты	Предупрежде- ние. «Выберите гостя и номер комнаты!»	Успех

Рисунок 2.26

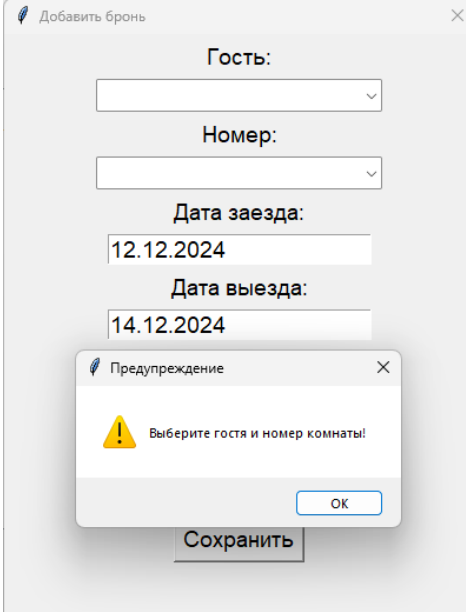
Продолжение таблицы 2.1

№	Наименование функционально-сти	Наименова-ние поля	Тесто-вый набор	Ожидаемый результат	Результат те-стирования
<div><div><div><div>Добавить бронь</div><div><div>Гость:</div><div></div></div><div><div>Номер:</div><div></div></div><div><div>Дата заезда:</div><div>12.12.2024</div></div><div><div>Дата выезда:</div><div>14.12.2024</div></div><div><div>Предупреждение</div><div><div>!</div><div>Выберите гостя и номер комнаты!</div></div><div>OK</div></div><div>Сохранить</div></div></div></div> <div>Рисунок 2.27</div>					
		Удалить бронь	Не выбирать бронь и нажать на кнопку «Удалить»	Предупрежде-ние. «Выберите бронь для уда-ления!»	Успех
			Выделить бронь и нажать на кнопку «Уда-лить»	Бронь удалена	Успех
		Редактировать бронь	Не выбирать бронь и нажать на кнопку «Ре-дактировать»	Предупрежде-ние. «Выберите бронь для ре-дактирования!»	Успех

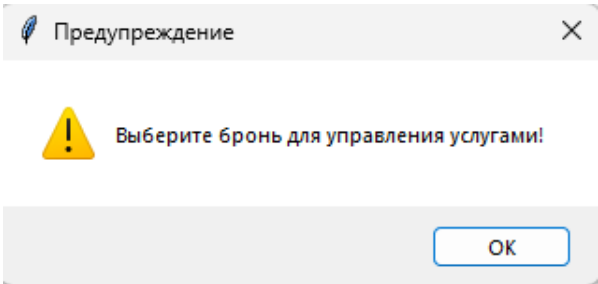
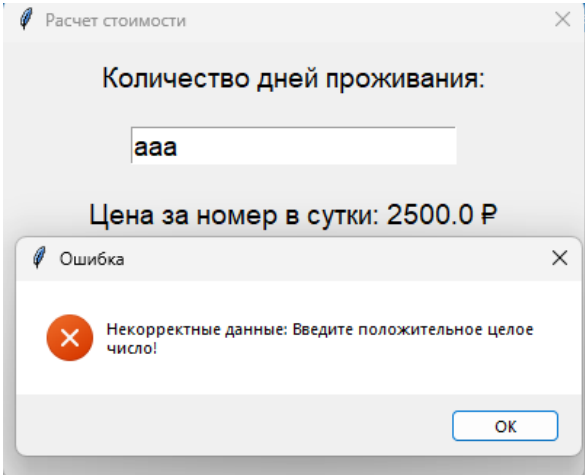
Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
<div><div>Предупреждение</div><div> Выберите бронь для редактирования!</div><div>OK</div></div> <p>Рисунок 2.28</p>					
			Выделить номер и нажать на кнопку «Редактировать»	Данные обновлены в БД. Сообщение «Данные брони обновлены!»	Успех
			Частичное заполнения полей	«Все поля обязательны для заполнения!»	Успех
			Ввели дату не полностью	Предупреждение. «Дата должна быть в формате ДД.ММ.ГГГГ!»	Успех
<div><div>Редактировать бронь</div><div>Гость: 5 - Петухова Вера</div><div>Номер: 202 - Люкс</div><div>Дата заезда: 12.12.20</div><div>Дата выезда: 14.12.2024</div><div>Статус:</div><div><div>Предупреждение</div><div> Дата должна быть в формате ДД.ММ.ГГГГ!</div><div>OK</div></div></div> <p>Рисунок 2.29</p>					

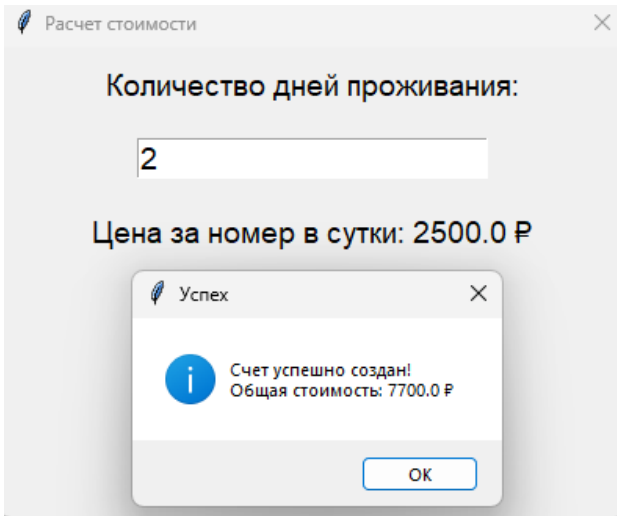
Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
			Не ввели гостя и номер комнаты	Предупреждение. «Выберите гостя и номер комнаты!»	Успех
<div></div> <p>Рисунок 2.30</p>					
		Услуги	Не выбирать бронь и нажать на кнопку «Услуги»	Предупреждение. «Выберите бронь для управления услугами!»	Успех
			Выбрать бронь и нажать на кнопку «Услуги»	Работает корректно. Переходит на следующее окно.	Успех

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
 <p>Рисунок 2.31</p>					
		Рассчитать	Не выбирать бронь и нажать на кнопку «Рассчитать»	Предупреждение. «Выберите бронь для расчета!»	Успех
			Выбрать бронь и нажать на кнопку «Рассчитать»	Работает корректно. Переходит на следующее окно.	Успех
			Ввести в поле «Количество дней проживания:» буквенные значения	Предупреждение. «Некорректные данные: Введите положительное число!»	Успех
 <p>Рисунок 2.32</p>					

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
			Ввести положительные данные	«Счет успешно создан! Общая стоимость: [Итоговая стоимость]!»	Успех
 <p>Рисунок 2.33</p>					
		Счет	Не выбирать бронь и нажать на кнопку «Редактировать»	Предупреждение. «Выберите бронь для расчёта!»	Успех
			Выбрать бронь и нажать на кнопку «Счет»	«Счет успешно создан!»	Успех
			Вводим отрицательное число дней	«Введите положительное число!»	Успех

Продолжение таблицы 2.1

№	Наименование функциональности	Наименование поля	Тестовый набор	Ожидаемый результат	Результат тестирования
			Вводим количество дней – 0	«Количество дней должно быть больше 0!»	Успех
	Главное меню. Счета	Удалить	Не выбирать бронь и нажать на кнопку «Удалить»	Предупреждение. «Выберите счет для удаления!»	Успех
			Выбрать бронь и нажать на кнопку «Удалить»	Счет удален	Успех

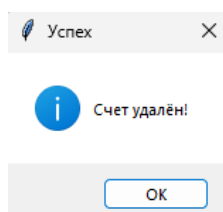


Рисунок 2.34

		Оплачен	Нажать на кнопку «Оплачен»	Статус оплаты изменен на «Оплачен»!	Успех
		Создать отчет	Нажать на кнопку «создать отчет».	Отчет успешно создан: Счета_отчет.xlsx	Успех

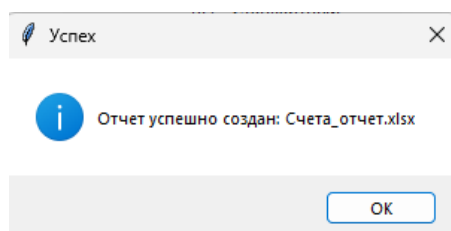
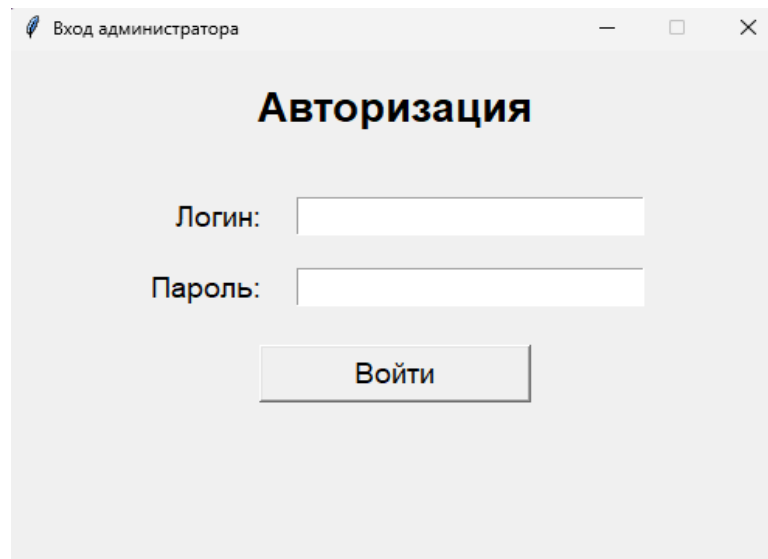


Рисунок 2.25

ПРИЛОЖЕНИЕ 3 Функционал кнопок



The screenshot shows a window titled "Вход администратора" (Administrator Login). The main heading is "Авторизация" (Authorization). Below it, there are two input fields: "Логин:" (Login) and "Пароль:" (Password). A "Войти" (Login) button is positioned below the password field.

Рисунок 3 – Окно авторизации



The screenshot shows a window titled "Панель администратора" (Administrator Panel). The main heading is "Панель управления отелем" (Hotel Management Panel). Below the heading, it says "Администратор: Иванов Алексей" (Administrator: Ivanov Alexey). There are five buttons arranged in a grid: "Гости" (Guests), "Номера" (Rooms), "Услуги" (Services), "Счета" (Accounts), and "Бронирования" (Reservations). A "Выйти" (Logout) button is located at the bottom center.

Рисунок 3.1 – Панель администратора

Управление гостями

Управление гостями

ID	Фамилия	Имя	Отчество	Телефон	Email	Паспорт
11	Копченова	Анна	Алексеевна	+7 (903) 614- 58 -00	kopchenova.anna.06@mail.ru	1111 111111
10	Рожкова	Мария	Ивановна	+7 (937) 440-15-41	mariya.rozhkova@example.com	4058 889889
9	Панкратов	Платон	Степанович	+7 (949) 467-42-82	platon.pankratov@example.com	4252 445071
8	Сухарев	Федор	Львович	+7 (919) 773-23-30	fedor.suharev@example.com	4614 178093
7	Лебедев	Михаил	Александрович	+7 (958) 734-35-97	mihail.lebedev@example.com	4088 240374
6	Пахомов	Даниил	Иванович	+7 (975) 756-63-51	daniil.pahomov@example.com	4593 986718
5	Петухова	Вера	Федоровна	+7 (956) 484-46-19	vera.petuhova@example.com	4721 328957
4	Воробьева	Вероника	Максимовна	+7 (995) 777-28-73	veronika.borobyova@example.com	4357 279861
3	Сидоров	Петр	Алексеевич	+7 (999) 648-14-47	petr.sidorov@example.com	4043 917074
2	Петрова	Ольга	Сергеевна	+7 (928) 480-67-50	olga.petrova@example.com	4273 802598
1	Иванов	Иван	Иванович	+7 (982) 986-66-64	ivan.ivanov@example.com	4970 977258

Добавить гостя
Удалить гостя
Редактировать гостя
Копировать

Поиск:

Рисунок 3.2 - Управление гостями

Управление бронями

Управление бронями

ID	Гость	Номер	Дата заезда	Дата выезда	Статус	Примечания
1	5 - Петухова Вера	202 - Люкс	12.12.2024	14.12.2024	Забронировано	

Добавить бронь
Удалить бронь
Редактировать бронь
Услуги
Рассчитать

Поиск:

Рисунок 3.3 – Управление номерами

Управление номерами

Управление номерами

Номер	Тип номера	Цена	Доступность
101	Стандартный	1000.0	1
102	Стандартный	1000.0	1
103	Стандартный	1500.0	0
104	Стандартный	2000.0	1
202	Люкс	2500.0	1
203	Люкс	3000.0	0
204	Люкс	3500.0	1
303	Апартаменты	8000.0	1
304	Апартаменты	10000.0	1

Добавить номер
Удалить номер
Редактировать номер

Поиск:

Рисунок 3.4 – Управление номерами

Управление счетами

Управление счетами

ID счета	ID брони	Гость	Комната	Сумма, руб.	Статус оплаты
9	1	1 - Иванов Иван	101 - Стандартный	16350.0	Не оплачен
8	2	3 - Сидоров Петр	202 - Люкс	14250.0	Оплачен
7	4	5 - Петухова Вера	104 - Стандартный	12900.0	Оплачен
6	5	7 - Лебедев Михаил	303 - Апартаменты	17000.0	Не оплачен
5	6	10 - Рожкова Мария	304 - Апартаменты	22700.0	Оплачен
4	7	2 - Петрова Ольга	102 - Стандартный	5250.0	Оплачен

Оплачен
Удалить счет
Создать отчет

Поиск:

Рисунок 3.5 – Управление счетами

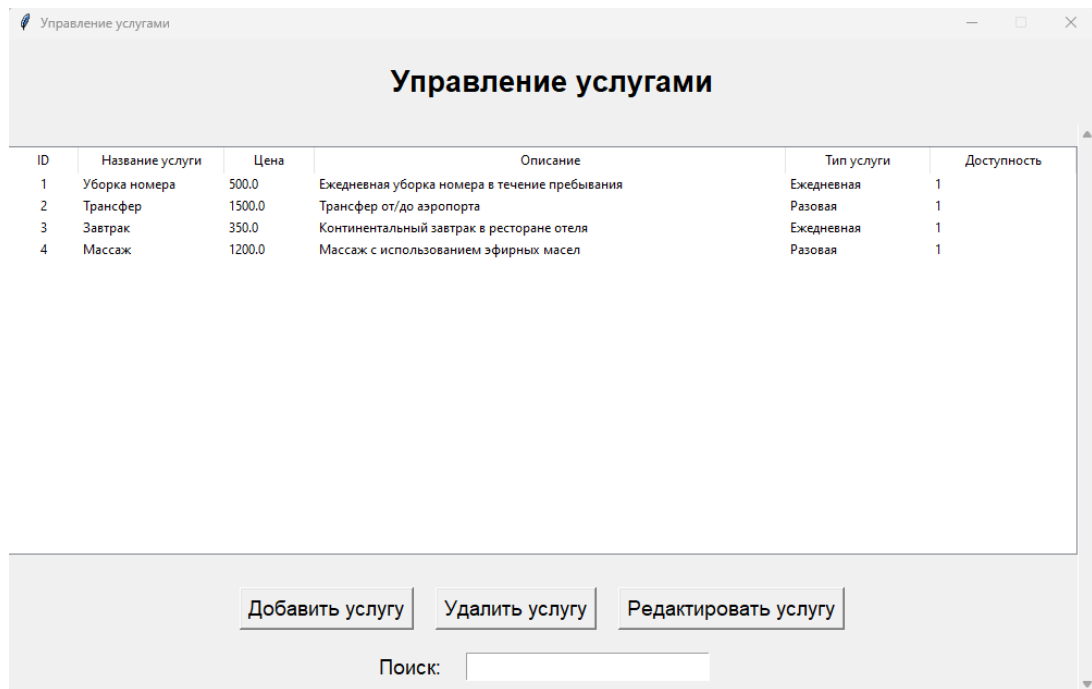


Рисунок 3.6 – Управление услугами

```

89  # Окно входа
    2 usages
90  def open_login_window():
91      global login_window, entry_login, entry_password
92      login_window = tk.Tk()
93      setup_window(login_window, title="Вход администратора", width=500, height=350)
94
95      # Заголовок
96      label_login = tk.Label(login_window, text="Авторизация", font=("Arial", 20, "bold"))
97      label_login.pack(pady=20)
98
99      # Поля ввода
100     frame_inputs = tk.Frame(login_window)
101     frame_inputs.pack(pady=10)
102
103     label_user = tk.Label(frame_inputs, text="Логин:", font=("Arial", 14))
104     label_user.grid(row=0, column=0, padx=10, pady=10, sticky="e")
105
106     entry_login = tk.Entry(frame_inputs, font=("Arial", 14))
107     entry_login.grid(row=0, column=1, padx=10, pady=10)
108
109     label_password = tk.Label(frame_inputs, text="Пароль:", font=("Arial", 14))
110     label_password.grid(row=1, column=0, padx=10, pady=10, sticky="e")
111
112     entry_password = tk.Entry(frame_inputs, font=("Arial", 14), show="*")
113     entry_password.grid(row=1, column=1, padx=10, pady=10)

```

Рисунок 3.7 - Функция для кнопки «Войти»

```

83  # Функция для выхода в окно авторизации
    1 usage
84  def logout():
85      root.destroy() # Закрываем главное меню
86      open_login_window() # Открываем окно входа
87

```

Рисунок 3.8 – Функция для кнопки «Выйти»

```

41 # Добавление нового гостя
42 1 usage
43 def add_guest(tree):
44     def save_guest():
45         guest_data = {
46             "last_name": entry_last_name.get(),
47             "first_name": entry_first_name.get(),
48             "middle_name": entry_middle_name.get(),
49             "phone": entry_phone.get(),
50             "email": entry_email.get(),
51             "passport": entry_passport.get(),
52         }
53
54         if not guest_data["last_name"] or not guest_data["first_name"]:
55             messagebox.showwarning( title: "Предупреждение", message: "Имя и фамилия обязательны для заполнения!")
56             return
57
58         try:
59             query = '''
60                 INSERT INTO Guests (last_name, first_name, middle_name, phone, email, passport)
61                 VALUES (?, ?, ?, ?, ?, ?)
62             '''
63             params = tuple(guest_data.values())
64             execute_query(DB_PATH, query, params)
65
66             messagebox.showinfo( title: "Успех", message: "Гость добавлен!")
67             add_window.destroy()
68             load_guests(tree) # Обновляем таблицу
69         except Exception as e:
70             messagebox.showerror( title: "Ошибка", message: f"Не удалось добавить гостя: {e}")

```

Рисунок 3.9 – Гости. Функция для кнопки «Добавить гостя»

```

111 # Удаление гостя с подтверждением
112 1 usage
113 def delete_guest(tree):
114     selected_item = tree.selection()
115     if not selected_item:
116         messagebox.showwarning( title: "Предупреждение", message: "Выберите гостя для удаления!")
117         return
118
119     guest_id = tree.item(selected_item)["values"][0]
120
121     try:
122         # Проверка наличия брони у гостя
123         query = "SELECT COUNT(*) FROM Bookings WHERE guest_id = ?"
124         result = execute_query(DB_PATH, query, params: (guest_id,))
125         count_bookings = result[0][0] # Получаем количество бронирований для данного гостя
126
127         if count_bookings > 0:
128             messagebox.showwarning( title: "Ошибка", message: "Невозможно удалить гостя, так как у него есть брони!")
129             return
130
131         # Если у гостя нет брони, удаляем его
132         confirmation = messagebox.askyesno( title: "Подтверждение удаления", message: "Вы уверены, что хотите удалить этого гостя?")
133         if confirmation:
134             query = "DELETE FROM Guests WHERE guest_id = ?"
135             execute_query(DB_PATH, query, params: (guest_id,))
136             messagebox.showinfo( title: "Успех", message: "Гость удален!")
137             load_guests(tree)
138
139     except Exception as e:
140         messagebox.showerror( title: "Ошибка", message: f"Не удалось удалить гостя: {e}")

```

Рисунок 3.10 – Гости. Функция для кнопки «Удалить гостя»

```

142 # Редактирование данных гостя
143 1 usage
144 def edit_guest(tree):
145     selected_item = tree.selection()
146     if not selected_item:
147         messagebox.showwarning( title: "Предупреждение", message: "Выберите гостя для редактирования!")
148         return
149     guest_data = tree.item(selected_item)["values"]
150
151     def save_changes():
152         updated_data = {
153             "last_name": entry_last_name.get(),
154             "first_name": entry_first_name.get(),
155             "middle_name": entry_middle_name.get(),
156             "phone": entry_phone.get(),
157             "email": entry_email.get(),
158             "passport": entry_passport.get(),
159         }
160
161         if not updated_data["last_name"] or not updated_data["first_name"]:
162             messagebox.showwarning( title: "Предупреждение", message: "Имя и фамилия обязательны для заполнения!")
163             return
164
165         guest_id = guest_data[0]
166
167         try:
168             query = """
169                 UPDATE Guests
170                 SET last_name = ?, first_name = ?, middle_name = ?, phone = ?, email = ?, passport = ?
171                 WHERE guest_id = ?
172             """
173             params = tuple(updated_data.values()) + (guest_id,)
174             execute_query(DB_PATH, query, params)
175
176             messagebox.showinfo( title: "Успех", message: "Данные гостя обновлены!")
177             edit_window.destroy()
178             load_guests(tree)
179         except Exception as e:
180             messagebox.showerror( title: "Ошибка", message: f"Не удалось обновить данные гостя: {e}")
181

```

Рисунок 3.11 – Гости. Функция для кнопки «Редактировать гостя»

```

226 # Копирование данных гостя
227 1 usage
228 def copy_guest(tree):
229     selected_item = tree.selection()
230     if not selected_item:
231         messagebox.showwarning( title: "Предупреждение", message: "Выберите гостя для копирования данных!")
232         return
233     guest_data = tree.item(selected_item)["values"]
234
235     # Собираем данные в строку, которую будем копировать в буфер обмена
236     clipboard_text = f"Фамилия: {guest_data[1]}\n" \
237                     f"Имя: {guest_data[2]}\n" \
238                     f"Отчество: {guest_data[3]}\n" \
239                     f"Телефон: {guest_data[4]}\n" \
240                     f"Email: {guest_data[5]}\n" \
241                     f"Паспорт: {guest_data[6]}"
242
243     # Копируем в буфер обмена
244     pyperclip.copy(clipboard_text)
245
246     # Выводим сообщение об успешном копировании
247     messagebox.showinfo( title: "Успех", message: "Данные скопированы в буфер обмена!")

```

Рисунок 3.12 – Гости. Функция для кнопки «Копировать»

```

315     # Событие, которое срабатывает при каждом изменении текста в поле ввода
316     def on_search_change(event):
317         search_query = search_entry.get() # Получаем текст из поля поиска
318         load_guests(tree, search_query) # Выполняем поиск
319
320     search_entry.bind("<KeyRelease>", on_search_change) # Привязываем событие к изменению текста
321
322     load_guests(tree)
323

```

Рисунок 3.13 – Гости. Функция для поля «Поиск»

```

105     # Удаление номера
106     1 usage
107     def delete_room(tree):
108         selected_item = tree.selection()
109         if not selected_item:
110             messagebox.showwarning( title: "Предупреждение", message: "Выберите номер для удаления!")
111             return
112
113         room_number = tree.item(selected_item)["values"][0]
114
115         try:
116             # Проверка наличия брони для номера
117             query = "SELECT COUNT(*) FROM Bookings WHERE room_number = ?"
118             result = execute_query(DB_PATH, query, params: (room_number,))
119             count_bookings = result[0][0] # Получаем количество бронирований для этого номера
120
121             if count_bookings > 0:
122                 messagebox.showwarning( title: "Ошибка", message: "Невозможно удалить номер, так как на него есть брони!")
123                 return
124
125             # Подтверждение удаления номера
126             confirmation = messagebox.askyesno( title: "Подтверждение удаления", message: "Вы уверены, что хотите удалить этот номер?")
127             if confirmation:
128                 query = "DELETE FROM Rooms WHERE room_number = ?"
129                 execute_query(DB_PATH, query, params: (room_number,))
130                 messagebox.showinfo( title: "Успех", message: "Номер удалён!")
131                 load_rooms(tree)
132
133             except Exception as e:
134                 messagebox.showerror( title: "Ошибка", message: f"Не удалось удалить номер: {e}")

```

Рисунок 3.14 – Номера. Функция для кнопки «Удалить номер»

```

152     def save_changes():
153         updated_data = {
154             "room_type": entry_room_type.get(),
155             "room_number": entry_room_number.get(),
156             "price": entry_price.get(),
157             "availability": var_availability.get(),
158         }
159
160         if not updated_data["room_type"] or not updated_data["room_number"] or not updated_data["price"]:
161             messagebox.showwarning( title: "Предупреждение", message: "Все поля обязательны для заполнения!")
162             return
163
164         if not updated_data["room_number"].isdigit():
165             messagebox.showwarning( title: "Ошибка", message: "Номер должен быть числом!")
166             return
167
168         if not updated_data["price"].replace( __old: '.', __new: '', __count: 1).isdigit():
169             messagebox.showwarning( title: "Ошибка", message: "Цена должна быть числом!")
170             return

```

Рисунок 3.15 – Номера. Функция для кнопки «Редактировать номер»

```

29 # Добавление нового номера
1 usage
30 def add_room(tree):
31     def save_room():
32         room_data = {
33             "room_type": entry_room_type.get(),
34             "room_number": entry_room_number.get(),
35             "price": entry_price.get(),
36             "availability": var_availability.get(),
37         }
38
39         if not room_data["room_type"] or not room_data["room_number"] or not room_data["price"]:
40             messagebox.showwarning( title: "Предупреждение", message: "Все поля обязательны для заполнения!")
41             return
42
43         if not room_data["room_number"].isdigit():
44             messagebox.showwarning( title: "Ошибка", message: "Номер должен быть числом!")
45             return
46
47         if not room_data["price"].replace( __old: '.', __new: '', __count: 1).isdigit():
48             messagebox.showwarning( title: "Ошибка", message: "Цена должна быть числом!")
49             return
50
51         # Проверка на существование такого же номера
52         query_check_existing = "SELECT COUNT(*) FROM Rooms WHERE room_number = ?"
53         result = execute_query(DB_PATH, query_check_existing,
54                                 params: (room_data["room_number"],))
55         if result[0][0] > 0:
56             messagebox.showwarning( title: "Ошибка", message: "Номер уже существует!")
57             return
58
59         try:
60             query = '''
61                 INSERT INTO Rooms (room_number, room_type, price, availability)
62                 VALUES (?, ?, ?, ?)
63             '''
64             params = (room_data["room_number"], room_data["room_type"], float(room_data["price"]),
65                     room_data["availability"])
66             execute_query(DB_PATH, query, params)
67
68             messagebox.showinfo( title: "Успех", message: "Номер добавлен!")
69             add_window.destroy()
70             load_rooms(tree)
71         except Exception as e:
72             messagebox.showerror( title: "Ошибка", message: f"Не удалось добавить номер: {e}")
73
74     add_window = tk.Toplevel()
75     add_window.title("Добавить номер")
76     add_window.transient(room_window)
77     add_window.resizable( width: False, height: False)
78     setup_window(add_window, title: "Добавить номер", width: 400, height: 500)
79
80     def on_close_add_window():
81         add_window.destroy()

```

Рисунок 3.15 – Номера. Функция для кнопки «Добавить номер»

```

291     # Поле поиска
292     search_frame = tk.Frame(room_window)
293     search_frame.pack(pady=10)
294
295     tk.Label(search_frame, text="Поиск:", font=("Arial", 14)).grid(row=0, column=0, padx=10)
296
297     search_entry = tk.Entry(search_frame, font=("Arial", 14))
298     search_entry.grid(row=0, column=1, padx=10)
299
300     def on_search_change(event):
301         search_query = search_entry.get()
302         load_rooms(tree, search_query)
303
304     search_entry.bind("<KeyRelease>", on_search_change)
---
```

Рисунок 3.16 – Номера. Функция для поля «Поиск»

```

132 # Удаление услуги
133 1 usage
134 def delete_service(tree):
135     selected_item = tree.selection()
136     if not selected_item:
137         messagebox.showwarning( title: "Предупреждение", message: "Выберите услугу для удаления!")
138         return
139
140     service_id = tree.item(selected_item)["values"][0]
141
142     try:
143         # Проверяем, используется ли услуга в актуальных бронях
144         query_check = """
145             SELECT COUNT(*)
146             FROM Booking_Services bs
147             JOIN Bookings b ON bs.booking_id = b.booking_id
148             WHERE bs.service_id = ? AND b.status IN ('Забронировано', 'Проживание', 'Выполнено', 'Отменено')
149             """
150         result = execute_query(DB_PATH, query_check, params: (service_id,))
151         if result[0][0] > 0: # Если услуга связана с актуальными бронями
152             messagebox.showwarning(
153                 title: "Удаление невозможно",
154                 message: "Эта услуга используется в бронях и не может быть удалена."
155             )
156             return
157
158         # Подтверждение удаления
159         confirmation = messagebox.askyesno( title: "Подтверждение удаления", message: "Вы уверены, что хотите удалить эту услугу?")
160         if confirmation:
161             query_delete = "DELETE FROM Services WHERE service_id = ?"
162             execute_query(DB_PATH, query_delete, params: (service_id,))
163             messagebox.showinfo( title: "Успех", message: "Услуга удалена!")
164             load_services(tree)
165
166     except Exception as e:
167         messagebox.showerror( title: "Ошибка", message: f"Не удалось удалить услугу: {e}")

```

Рисунок 3.17 - Услуги. Функция для кнопки «Удалить услугу»


```

50 # Добавление новой услуги
51 1 usage
52 def add_service(tree):
53     def save_service():
54         service_data = {
55             "service_name": entry_service_name.get(),
56             "price": entry_price.get(),
57             "description": entry_description.get( index1: "1.0", index2: "end-1c"), # Получаем текст из поля Text
58             "service_type": service_type_var.get(),
59             "availability": availability_var.get()
60         }
61
62         if not service_data["service_name"] or not service_data["price"]:
63             messagebox.showwarning( title: "Предупреждение", message: "Все поля обязательны для заполнения!")
64             return
65
66         if not service_data["price"].replace( __old: '.', __new: '', __count: 1).isdigit():
67             messagebox.showwarning( title: "Ошибка", message: "Цена должна быть числом!")
68             return
69
70         try:
71             # Включаем описание услуги, тип и доступность в запрос
72             query = '''
73             INSERT INTO Services (service_name, price, description, service_type, availability)
74             VALUES (?, ?, ?, ?, ?)
75             '''
76             params = (
77                 service_data["service_name"],
78                 float(service_data["price"]),
79                 service_data["description"],
80                 service_data["service_type"],
81                 service_data["availability"]
82             )
83             execute_query(DB_PATH, query, params)
84
85             messagebox.showinfo( title: "Успех", message: "Услуга добавлена!")
86             add_window.destroy()
87             load_services(tree)
88         except Exception as e:
89             messagebox.showerror( title: "Ошибка", message: f"Не удалось добавить услугу: {e}")
90
91     add_window = tk.Toplevel()
92     add_window.title("Добавить услугу")
93     add_window.transient(service_window)
94     add_window.resizable( width: False, height: False)
95     setup_window(add_window, title: "Добавить услугу", width: 400, height: 600) # Увеличиваем высоту окна для описания
96
97     def on_close_add_window():
98         add_window.destroy()
99
100     add_window.protocol( name: "WM_DELETE_WINDOW", on_close_add_window)

```

Рисунок 3.18 – Услуги. Функция для кнопки «Добавить услугу»

```

178 def save_changes():
179     updated_data = {
180         "service_name": entry_service_name.get(),
181         "price": entry_price.get(),
182         "description": entry_description.get( index1: "1.0", index2: "end-1c"), # Получаем текст из поля Text
183         "service_type": service_type_var.get(),
184         "availability": availability_var.get()
185     }
186
187     if not updated_data["service_name"] or not updated_data["price"]:
188         messagebox.showwarning( title: "Предупреждение", message: "Все поля обязательны для заполнения!")
189         return
190
191     if not updated_data["price"].replace( __old: '.', __new: '', __count: 1).isdigit():
192         messagebox.showwarning( title: "Ошибка", message: "Цена должна быть числом!")
193         return
194

```

Рисунок 3.19 – Услуги. Функция для кнопки «Редактировать услугу»

```

342 def on_search_change(event):
343     search_query = search_entry.get()
344     load_services(tree, search_query)
345
346     search_entry.bind("<KeyRelease>", on_search_change)
347
348     = load_services(tree)
349

```

Рисунок 3.20 – Услуги. Функция для поля «Поиск»

```

100 # Функция для экспорта данных в Excel
101 usage
102 def export_to_excel(tree):
103     # Подтверждение действия
104     if not messagebox.askyesno( title: "Подтверждение", message: "Вы уверены, что хотите создать отчет в Excel?"):
105         return
106
107     try:
108         # Создаём новый Excel-файл
109         wb = Workbook()
110         ws = wb.active
111         ws.title = "Счета"
112
113         # Получаем заголовки из дерева
114         headers = [tree.heading(col)["text"] for col in tree["columns"]]
115         column_widths = [tree.column(col)["width"] for col in tree["columns"]]
116
117         # Заполняем заголовки в Excel
118         for col_num, (header, width) in enumerate(zip(headers, column_widths), start=1):
119             cell = ws.cell(row=1, column=col_num, value=header)
120             cell.font = Font(bold=True) # Заголовки жирным
121             # Настроиваем ширину столбцов Excel (переводим в условные единицы)
122             ws.column_dimensions[cell.column_letter].width = width // 6 # Примерное преобразование
123
124         # Заполняем данные из таблицы
125         total_paid = 0 # Для подсчета суммы оплаченных счетов
126         for row_num, item in enumerate(tree.get_children(), start=2):
127             values = tree.item(item, "values")
128             for col_num, value in enumerate(values, start=1):
129                 ws.cell(row=row_num, column=col_num, value=value)
130             # Если счет оплачен, добавляем его сумму к итогу
131             if values[-1] == "Оплачен":
132                 total_paid += float(values[4])

```

Рисунок 3.21 – Счета. Функция кнопки «Отчет»

```

46 # Удаление счета
47 1 usage
48 def delete_bill(tree):
49     selected_item = tree.selection()
50     if not selected_item:
51         messagebox.showwarning( title: "Предупреждение", message: "Выберите счет для удаления!")
52         return
53     bill_id = tree.item(selected_item)["values"][0]
54
55     try:
56         # Подтверждение удаления счета
57         confirmation = messagebox.askyesno( title: "Подтверждение удаления", message: "Вы уверены, что хотите удалить этот счет?")
58         if confirmation:
59             query = "DELETE FROM Bills WHERE bill_id = ?"
60             execute_query(DB_PATH, query, params: (bill_id,))
61             messagebox.showinfo( title: "Успех", message: "Счет удалён!")
62             load_bills(tree)
63
64     except Exception as e:
65         messagebox.showerror( title: "Ошибка", message: f"Не удалось удалить счет: {e}")

```

Рисунок 3.22 – Счета. Функция для кнопки «Удалить счет»

```

68 # Изменение статуса оплаты
69 1 usage
70 def change_payment_status(tree):
71     selected_item = tree.selection()
72     if not selected_item:
73         messagebox.showwarning( title: "Предупреждение", message: "Выберите счет для изменения статуса оплаты!")
74         return
75     bill_id = tree.item(selected_item)["values"][0]
76     current_status = tree.item(selected_item)["values"][5] # Столбец "Статус оплаты"
77
78     if current_status == "Не оплачен":
79         confirmation = messagebox.askyesno(
80             title: "Подтверждение оплаты",
81             message: "Вы уверены, что гость произвел оплату?"
82         )
83         if not confirmation:
84             return
85
86     # Определяем новый статус
87     new_status = "Оплачен" if current_status == "Не оплачен" else "Не оплачен"
88
89     try:
90         query = "UPDATE Bills SET payment_status = ? WHERE bill_id = ?"
91         execute_query(DB_PATH, query, params: (new_status, bill_id))
92         messagebox.showinfo( title: "Успех", message: f"Статус оплаты изменен на '{new_status}'!")
93         load_bills(tree) # Перезагрузка данных
94     except Exception as e:
95         messagebox.showerror( title: "Ошибка", message: f"Не удалось изменить статус оплаты: {e}")

```

Рисунок 3.23 – Счета. Функция для кнопки «Оплачен»

```

145     # Поле поиска
146     search_frame = tk.Frame(bill_window)
147     search_frame.pack(pady=10)
148
149     tk.Label(search_frame, text="Поиск:", font=("Arial", 14)).grid(row=0, column=0, padx=10)
150
151     search_entry = tk.Entry(search_frame, font=("Arial", 14))
152     search_entry.grid(row=0, column=1, padx=10)
153
154     def on_search_change(event):
155         search_query = search_entry.get()
156         load_bills(tree, search_query)
157
158     search_entry.bind("<KeyRelease>", on_search_change)

```

Рисунок 3.24 – Счета. Функция для поля «Поиск»

```

51     # Добавление брони
52     1 usage
53     def add_booking(tree):
54         def save_booking():
55             # Получение ID гостя и номера комнаты из выбранных значений
56             selected_guest = var_guest.get()
57             selected_room = var_room.get()
58             if selected_guest and selected_room:
59                 booking_data = {
60                     "guest_id": selected_guest.split(" - ")[0], # Извлечение guest_id
61                     "room_number": selected_room.split(" - ")[0], # Извлечение room_number
62                     "checking_date": entry_checking_date.get(),
63                     "checkout_date": entry_checkout_date.get(),
64                     "status": var_status.get(),
65                     "notes": entry_notes.get(),
66                 }
67             else:
68                 messagebox.showwarning( title: "Предупреждение", message: "Выберите гостя и номер комнаты!")
69                 return
70             # Проверка обязательных данных
71             if not booking_data["checking_date"] or not booking_data["checkout_date"]:
72                 messagebox.showwarning( title: "Предупреждение", message: "Все поля обязательны для заполнения!")
73                 return
74

```

Рисунок 3.25 – Бронирования. Функция для кнопки «Добавить бронь»

```

152     # Удаление брони
153     1 usage
154     def delete_booking(tree):
155         selected_item = tree.selection()
156         if not selected_item:
157             messagebox.showwarning( title: "Предупреждение", message: "Выберите бронь для удаления!")
158             return
159         booking_id = tree.item(selected_item)["values"][0]
160
161         # Подтверждение удаления
162         confirmation = messagebox.askyesno( title: "Подтверждение удаления", message: "Вы уверены, что хотите удалить эту бронь?")
163         if not confirmation:
164             return

```

Рисунок 3.26 - Бронирования. Функция для кнопки «Удалить бронь»

```

593     def on_search_change(event):
594         search_query = search_entry.get()
595         load_bookings(tree, search_query)
596
597     search_entry.bind("<KeyRelease>", on_search_change)
598

```

Рисунок 3.27 – Бронирования. Функция для поля «Поиск»

```

194 # Редактирование брони
195 1 usage
196 def edit_booking(tree):
197     selected_item = tree.selection()
198     if not selected_item:
199         messagebox.showwarning( title: "Предупреждение", message: "Выберите бронь для редактирования!")
200         return
201
202     booking_data = tree.item(selected_item)["values"]
203
204     def save_changes():
205         # Получение ID гостя и номера комнаты из выбранных значений
206         selected_guest = var_guest.get()
207         selected_room = var_room.get()
208         if selected_guest and selected_room:
209             updated_data = {
210                 "guest_id": selected_guest.split(" - ")[0], # Извлечение guest_id
211                 "room_number": selected_room.split(" - ")[0], # Извлечение room_number
212                 "checking_date": entry_checking_date.get(),
213                 "checkout_date": entry_checkout_date.get(),
214                 "status": var_status.get(),
215                 "notes": entry_notes.get(),
216             }
217

```

Рисунок 3.28 - Бронирования. Функция для кнопки «Редактировать бронь»

```

402 1# Интерфейс управления услугами в брони
403 1 usage
404 def manage_services(tree):
405     selected_booking = tree.selection()
406     if not selected_booking:
407         messagebox.showwarning( title: "Предупреждение", message: "Выберите бронь для управления услугами!")
408         return
409     booking_id = tree.item(selected_booking[0], "values")[0] # Получаем ID брони
410
411     def save_service_changes():
412         try:
413             delete_query = "DELETE FROM Booking_Services WHERE booking_id = ?"
414             execute_query(DB_PATH, delete_query, params: (booking_id,))
415
416             insert_query = "INSERT INTO Booking_Services (booking_id, service_id) VALUES (?, ?)"
417             for row_id in services_tree.get_children():
418                 row_data = services_tree.item(row_id, option: "values")
419                 service_id = row_data[0]
420                 selected = row_data[5] == "✓"
421
422                 if selected:
423                     execute_query(DB_PATH, insert_query, params: (booking_id, service_id))
424
425             messagebox.showinfo( title: "Успех", message: "Изменения успешно сохранены!")
426             services_window.destroy()
427             load_bookings(tree) # Обновляем таблицу броней
428         except Exception as e:
429             messagebox.showerror( title: "Ошибка", message: f"Не удалось сохранить изменения: {e}")
430
431     def load_service_data_with_checkbox(search_query=""):
432         for row in services_tree.get_children():
433             services_tree.delete(row)

```

Рисунок 3.29 – Бронирования. Функция для кнопки «Услуги» в брони

```

311 # Интерфейс для создания счета
312 1 usage
313 def calculate_bill(tree):
314     selected_booking = tree.selection()
315     if not selected_booking:
316         messagebox.showwarning( title: "Предупреждение", message: "Выберите бронь для расчета!")
317         return
318     booking_id, status = tree.item(selected_booking[0], "values")[0:2]
319
320     # Получаем данные о стоимости номера
321     query = '''
322     SELECT r.price
323     FROM Bookings b
324     JOIN Rooms r ON b.room_number = r.room_number
325     WHERE b.booking_id = ?
326     '''
327     booking_details = execute_query(DB_PATH, query, params: (booking_id,))
328
329     if not booking_details:
330         messagebox.showerror( title: "Ошибка", message: "Данные о брони не найдены!")
331         return

```

Рисунок 3.30 – Бронирования. Функция для кнопки «Рассчитать» в брони

ПРИЛОЖЕНИЕ 4 Листинг кода

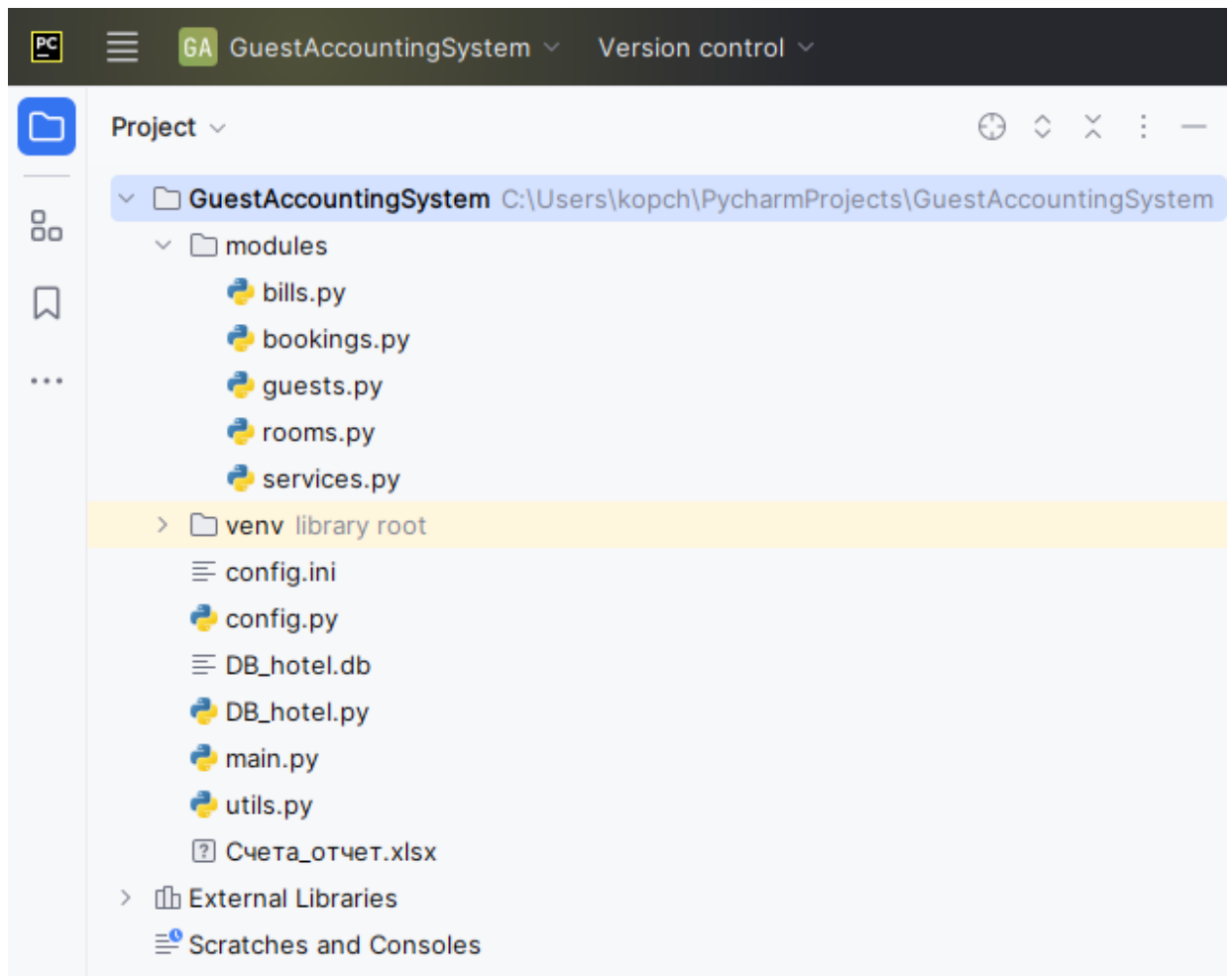


Рисунок 4.1 – Листинг кода