**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский арктический горно-политехнический колледж»**

**КУРСОВОЙ ПРОЕКТ**

По дисциплине МДК.05.02 Разработка кода информационных систем

**Разработка автоматизированной информационной системы «Администратор гостиницы»**

Выполнил студент гр. ИСП-21 /\_\_\_\_\_\_\_\_\_\_\_\_\_/ Ганеева Вера Андреевна/

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Научный руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Данил Павлович/

(подпись) (Ф.И.О.)

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ 3**](#_Toc195210664)

[**ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 5**](#_Toc195210665)

[**1.1 Платформа .NET 5**](#_Toc195210666)

[**1.2 Язык программирования С# 6**](#_Toc195210667)

[**1.3 Windows Presentation Foundation (WPF) 7**](#_Toc195210668)

[**1.4 СУБД SQL server 9**](#_Toc195210669)

[**1.5 Microsoft SQL Server Management Studio 10**](#_Toc195210670)

[**1.6 Entity Framework 12**](#_Toc195210671)

[**ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 14**](#_Toc195210672)

[**2.1 Разработка диаграммы ERD 14**](#_Toc195210673)

[**2.2 Разработка базы данных 15**](#_Toc195210674)

[**ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 20**](#_Toc195210675)

[**3.1 Разработка прототипа информационной системы 20**](#_Toc195210676)

[**3.2 Программирование информационной системы 24**](#_Toc195210677)

[**3.2.1 Разработка модуля «Авторизация» 24**](#_Toc195210678)

[**3.2.2 Разработка модуля «Главное окно менеджера» 28**](#_Toc195210679)

[**3.2.3 Разработка модуля «Главное окно горничной» 31**](#_Toc195210680)

[**3.2.4 Разработка модуля «Главное окно администратора» 32**](#_Toc195210681)

[**3.2.5 Разработка модуля «Отчеты» 36**](#_Toc195210682)

[**ЗАКЛЮЧЕНИЕ 42**](#_Toc195210683)

[**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 43**](#_Toc195210684)

# ВВЕДЕНИЕ

Современная гостиничная индустрия характеризуется высокой конкуренцией и динамично меняющимися требованиями клиентов. Для успешного функционирования и развития гостиничного бизнеса необходимо эффективное управление всеми аспектами деятельности, начиная от бронирования номеров и заканчивая учетом финансовых операций. В этом контексте автоматизация процессов играет ключевую роль, позволяя оптимизировать работу персонала, повысить качество обслуживания и увеличить прибыльность предприятия.

Информационная система "Администратор гостиницы" предназначена для автоматизации основных бизнес-процессов гостиницы. Система призвана решить ряд актуальных задач, связанных с управлением номерным фондом, взаимодействием с клиентами, контролем загрузки гостиницы, ведением учета и формированием отчетности. Внедрение такой системы позволит повысить эффективность работы персонала, минимизировать вероятность ошибок, связанных с человеческим фактором, и улучшить качество обслуживания гостей.

В рамках курсовой работы будут рассмотрены основные функциональные требования к системе, описана ее архитектура, выбранные технологии разработки, а также проведен анализ эффективности внедрения. Особое внимание будет уделено вопросам безопасности данных и интеграции с другими системами, например, системами онлайн-бронирования. Разработка информационной системы "Администратор гостиницы" способствует модернизации гостиничного бизнеса и повышению его конкурентоспособности на современном рынке.

**Объект:** информационная система «Администратор гостиницы».

**Предмет:** анализ бизнес-процессов «Администратор гостиницы».

**Цель работы:** разработать информационную систему «Администратор гостиницы».

**Задачи:**

* выбрать инструментарий;
* спроектировать базу данных;
* разработать информационную систему.

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

# Платформа .NET

.NET — это бесплатная кроссплатформенная платформа разработчика с открытым кодом для создания различных видов приложений. Он может запускать программы, написанные на нескольких языках, с C# самым популярным. Она использует высокопроизводительную среду выполнения, которая используется в рабочей среде многих высокомасштабируемых приложений.

Чтобы узнать, как скачать .NET и начать написание первого приложения, см. начало работы.

Платформа .NET разработана для обеспечения производительности, производительности, безопасности и надежности. Она обеспечивает автоматическое управление памятью с помощью сборщика мусора (GC). Это типобезопасный и безопасный для памяти, из-за использования GC и строгих компиляторов языка. Он предлагает параллелизм посредством async/await и примитивов Task. Он включает в себя большой набор библиотек, имеющих широкие функциональные возможности и оптимизированный для производительности в нескольких операционных системах и архитектурах микросхем.

.NET имеет следующие точки разработки:

* Производительность является полноценной со средой выполнения, библиотеками, языком и инструментами, которые способствуют опыту работы разработчика.
* безопасный код является основной вычислительной моделью, а небезопасный код обеспечивает дополнительную оптимизацию вручную.
* статический и динамический код поддерживаются, что позволяет использовать широкий набор различных сценариев.
* взаимодействие с нативным кодом и аппаратные встроенные функции обеспечивают низкие затраты и высокую точность (сырой доступ к API и инструкциям).
* Код переносится на разных платформах (архитектура ОС и микросхем), а целевая платформа обеспечивает специализацию и оптимизацию.
* Адаптация между доменами программирования (облачные, клиентские, игровые) включена с помощью специализированных реализаций модели программирования общего назначения.
* отраслевые стандарты, такие как OpenTelemetry и gRPC, предпочитаются по сравнению с решениями bespoke.

.NET поддерживается корпорацией Майкрософт и сообществом. Она регулярно обновляется, чтобы пользователи могли развертывать безопасные и надежные приложения в рабочей среде.

# Язык программирования С#

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

C# уже не молодой язык и как и вся платформа .NET уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 13, которая вышла 12 ноября 2024 года вместе с релизом .NET 9.

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Поэтому, если вы знакомы с одним из этих языков, то овладеть C# будет легче.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей.

Язык входит в семью С-подобных языков. Синтаксис приближен к Java и C++. Его особенности:

* статистическая типизация,
* поддерживается полиморфизм,
* поддерживается перегрузка операторов,
* доступна делегация, атрибуты, события, обобщенные типы и анонимные функции.
* Разработка Microsoft много особенностей унаследовала у Delphi, Smalltalk и Java.

При этом создатели нового языка исключили из своего детища многие практики и спецификации, считающиеся «проблемными».

С# популярен за счет своей «простоты». Простоты для современных программистов и больших команд разработчиков, чтобы те могли в сжатые сроки создавать функциональные и производительные приложения. Этому способствуют нетипичные конструкции языка и специфичный синтаксис, помогающий максимально органично реализовать намеченные функции.

# Windows Presentation Foundation (WPF)

Технология WPF (Windows Presentation Foundation) является частью экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

Одной из важных особенностей является использование языка декларативной разметки интерфейса XAML, основанного на XML: вы можете создавать насыщенный графический интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#, VB.NET и F#, либо совмещать и то, и другое.

Что предлагает WPF:

* использование традиционных языков .NET-платформы - C#, F# и VB.NET для создания логики приложения;
* возможность декларативного определения графического интерфейса с помощью специального языка разметки XAML, основанном на xml и представляющем альтернативу программному созданию графики и элементов управления, а также возможность комбинировать XAML и C#/VB.NET;
* независимость от разрешения экрана: поскольку в WPF все элементы измеряются в независимых от устройства единицах, приложения на WPF легко масштабируются под разные экраны с разным разрешением;
* новые возможности, которых сложно было достичь в WinForms, например, создание трехмерных моделей, привязка данных, использование таких элементов, как стили, шаблоны, темы и др;
* хорошее взаимодействие с WinForms, благодаря чему, например, в приложениях WPF можно использовать традиционные элементы управления из WinForms;
* богатые возможности по созданию различных приложений: это и мультимедиа, и двухмерная и трехмерная графика, и богатый набор встроенных элементов управления, а также возможность самим создавать новые элементы, создание анимаций, привязка данных, стили, шаблоны, темы и многое другое;
* аппаратное ускорение графики - вне зависимости от того, работаете ли вы с 2D или 3D, графикой или текстом, все компоненты приложения транслируются в объекты, понятные Direct3D, и затем визуализируются с помощью процессора на видеокарте, что повышает производительность, делает графику более плавной;
* создание приложений под множество ОС семейства Windows.

# СУБД SQL server

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

SQL Server был создан компанией Microsoft. Первая версия вышла в 1987 году. А текущей версией является версия 2022, которая вышла в ноябре 2022 году и которая будет использоваться в текущем руководстве.

SQL Server долгое время был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux.

SQL Server характеризуется такими особенностями как:

* производительность. SQL Server работает очень быстро;
* надежность и безопасность. SQL Server предоставляет шифрование данных;
* простота. С данной СУБД относительно легко работать и вести администрирование.

Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД.

Для организации баз данных MS SQL Server использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных.

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов. Используя первичный ключ, мы можем ссылаться на определенную строку в таблице. Соответственно две строки не могут иметь один и тот же первичный ключ.

Через ключи одна таблица может быть связана с другой, то есть между двумя таблицами могут быть организованы связи. А сама таблица может быть представлена в виде отношения ("relation").

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

# Microsoft SQL Server Management Studio

Среда управления SQL Server (SSMS) — это интегрированная среда для управления любой инфраструктурой SQL. Используйте SSMS для доступа к компонентам SQL Server, базам данных Azure SQL, управляемым инстансам Azure SQL, SQL Server на виртуальной машине Azure и Azure Synapse Analytics, а также для их настройки, управления, администрирования и разработки. SSMS — это единая комплексная утилита, которая объединяет широкий набор графических инструментов с множеством редакторов скриптов для обеспечения доступа к SQL Server для разработчиков и администраторов баз данных любого уровня подготовки.

Некоторые возможности SSMS:

* создание баз данных SQL Server, скриптов SQL и заданий SQL Server Agent;
* подключение к экземплярам базы данных SQL;
* создание, изменение и удаление объектов базы данных SQL (таблиц, индексов и схем);
* управление и мониторинг экземпляров базы данных SQL;
* генерация запросов и скриптов;
* развёртывание, отслеживание и обновление компонентов уровня данных, используемых приложениями.

Среда управления SQL Server (SSMS) позволяет управлять объектами служб Analysis Services, например создавать резервные копии и обрабатывать объекты.

SSMS предоставляет проект «Службы сценариев анализа», в котором можно разрабатывать и сохранять сценарии, написанные на многомерных выражениях (MDX), выражениях анализа данных (DAX), расширениях интеллектуального анализа данных (DMX) и XML для анализа (XMLA).

Эти сценарии используются для выполнения задач управления или воссоздания таких объектов, как базы данных и экземпляры в кубах служб Analysis Services. Например, вы можете создать сценарий XMLA в проекте сценариев служб Analysis Services для создания новых объектов непосредственно в существующем экземпляре. Эти проекты можно сохранить как часть решения и интегрировать с системой управления исходным кодом.

В целом, SQL Server Management Studio (SSMS) — это мощный инструмент, предоставляющий полную среду для управления базами данных SQL Server. Благодаря набору инструментов и функций SSMS упрощает разработку, управление и поддержку баз данных, позволяя пользователям более эффективно работать с экземплярами SQL Server.

# 1.6 Entity Framework

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Первая версия Entity Framework - 1.0 вышла еще в 2008 году и представляла очень ограниченную функциональность, базовую поддержку ORM (object-relational mapping - отображения данных на реальные объекты) и один единственный подход к взаимодействию с бд - Database First. С выходом версии 4.0 в 2010 году многое изменилось - с этого времени Entity Framework стал рекомендуемой технологией для доступа к данным, а в сам фреймворк были введены новые возможности взаимодействия с бд - подходы Model First и Code First.

Дополнительные улучшения функционала последовали с выходом версии 5.0 в 2012 году. И наконец, в 2013 году был выпущен Entity Framework 6.0, обладающий возможностью асинхронного доступа к данным.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес. Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами в БД.

Entity Data Model состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга).

На концептуальном уровне происходит определение классов сущностей, используемых в приложении.

Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных.

Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

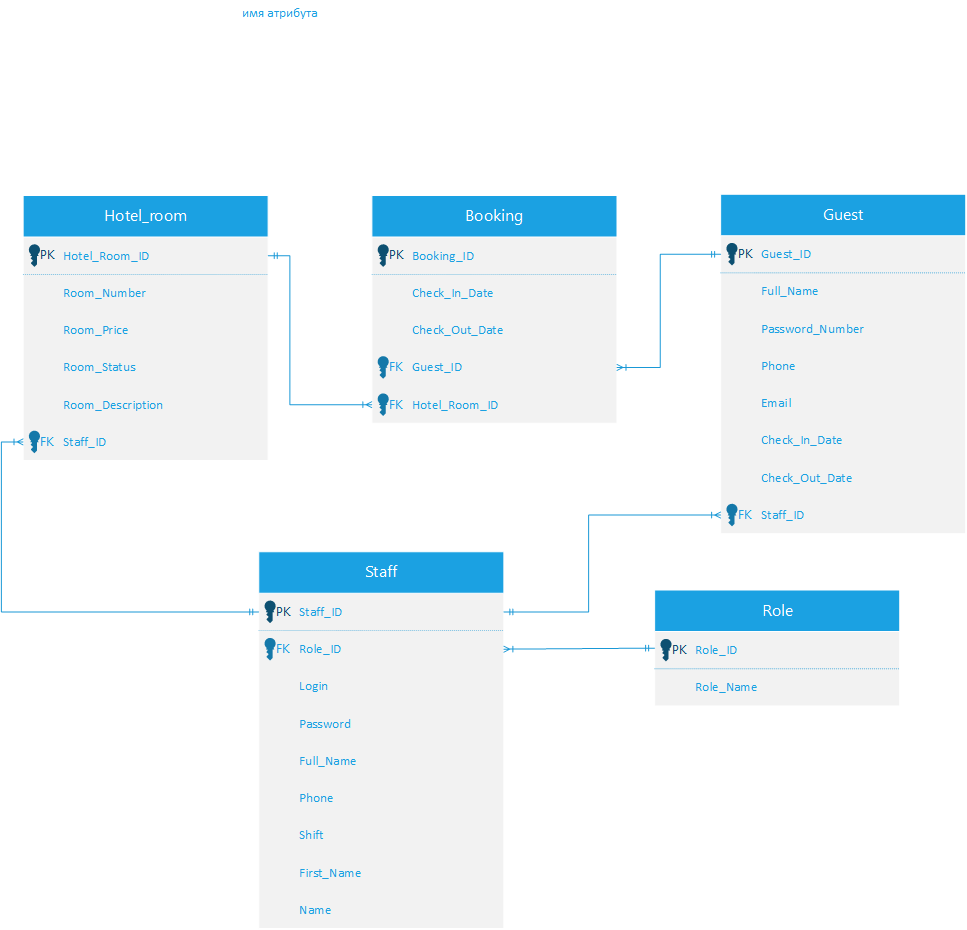
# ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

# 2.1 Разработка диаграммы ERD

Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса. ER-диаграммы делятся на концептуальные и физические. В отличие от физических, в концептуальных ER-диаграммах не учитываются особенности конкретной базы данных. Впоследствии сущности концептуальных ER-диаграмм становятся таблицами, атрибуты — колонками, а связи реализуются путем миграции ключевых атрибутов родительских сущностей и создания внешних ключей.

ER-диаграммы (или ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей. Существуют три типа связей в ER-диаграмме: 1:1 – «один-к-одному», 1:N – «один-ко-многим», M:N - «многие-ко-многим». Атрибуты в ER-диаграмме — это характеристики или свойства сущностей. Они детализируют сущности, добавляя им контекста и уникальности.

Чтобы создать ER-диаграмму, были заданы сущности, определены связи между ними, добавлены атрибуты.



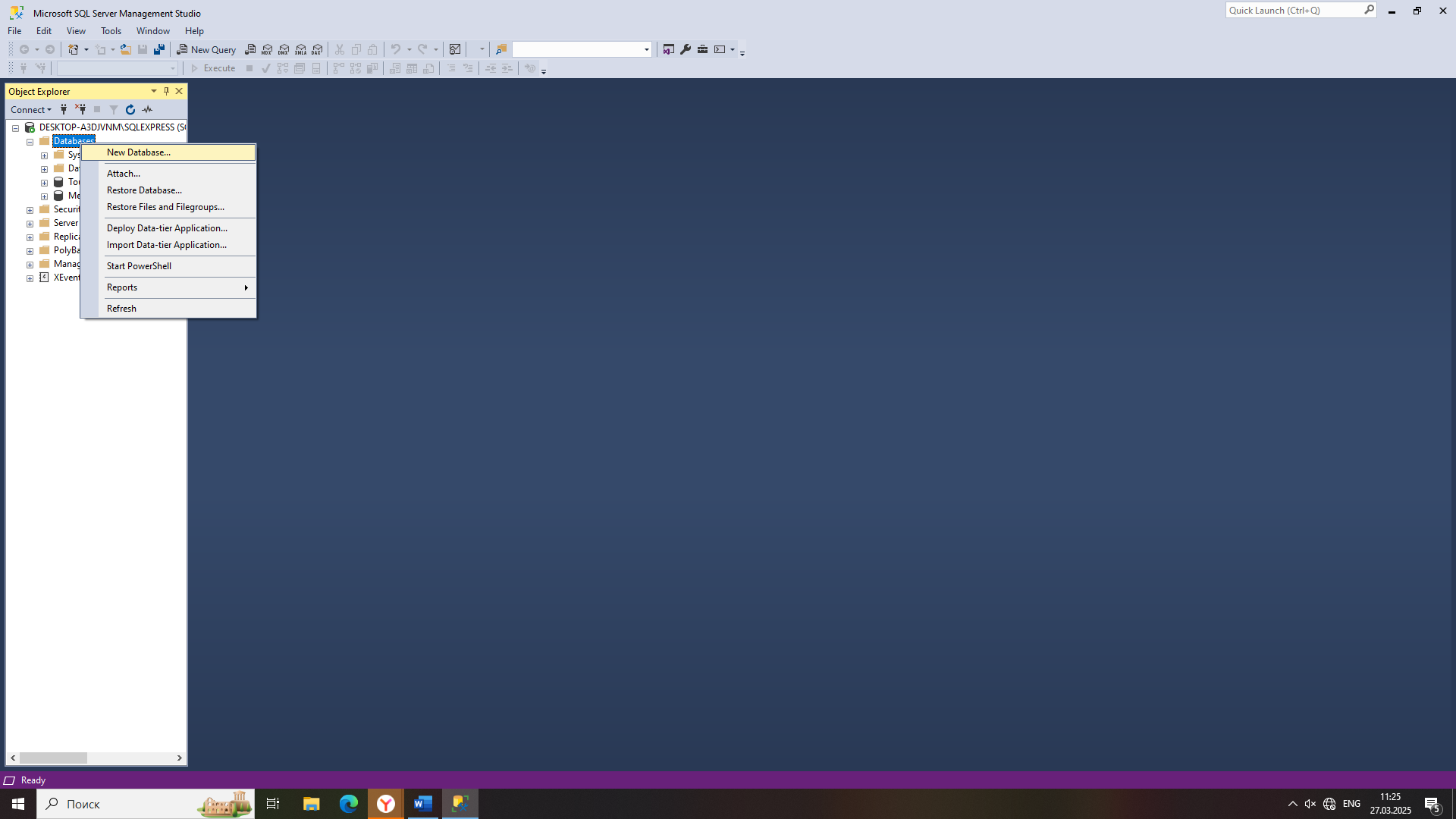
*Рисунок 2.1. ER-диаграмма*

В данной работе используется 5 таблиц базы данных. Сущность Hotel\_Room (Номер отеля) имеет атрибуты: ID (уникальный ключ), Room\_Number (Номер отеля), Room\_Price (Цена номера), Room\_Status (Статус номера), Room\_Description (Описание номера), Staff\_ID (уникальный ключ персонала). Сущность Booking (Бронирование) имеет атрибуты: ID (уникальный ключ), Check\_In\_Date (Дата заселения), Check\_Out\_Date (Дата выселения), Guest\_ID (уникальный ключ гостя), Hotel\_Room\_ID (уникальный ключ номера отеля). Сущность Guest имеет атрибуты: ID (уникальный ключ), Full\_Name (ФИО), Passport\_Number (Номер паспорта), Phone (Номер телефона), Email, Check\_In\_Date (Дата заселения), Check\_Out\_Date (Дата выселения), Staff\_ID (уникальный ключ персонала). Сущность Staff имеет атрибуты: ID (уникальный ключ), Role\_ID (уникальный ключ ролей), Login, Password (Пароль), Full\_Name (ФИО), Phone (Номер телефона), Shift (Смена). Сущность Role (Роль) имеет атрибуты: ID (уникальный ключ), Role\_Name (Название роли).

# 2.2 Разработка базы данных

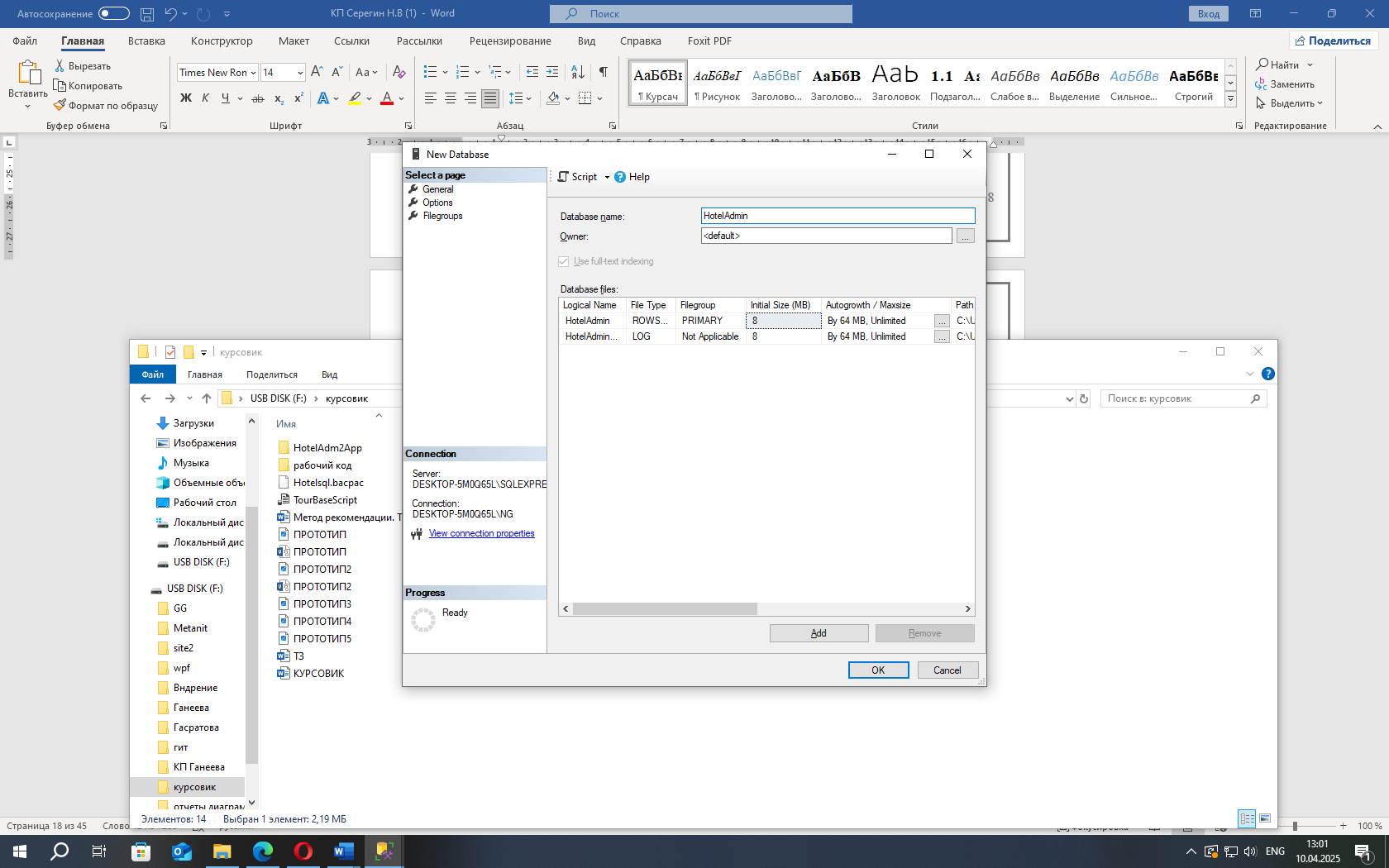
Разработка базы данных будет происходить в программе SSMS (SQL Server Management Studio).

**SQL Server Management Studio (SSMS)**является важным инструментом для эффективного управления **базами данных SQL Server**. Разработанная компанией **Microsoft**, **SSMS**предоставляет комплексную интегрированную среду, в которой **администраторы баз данных**, разработчики и другие пользователи могут эффективно работать со своими **экземплярами SQL Server**.



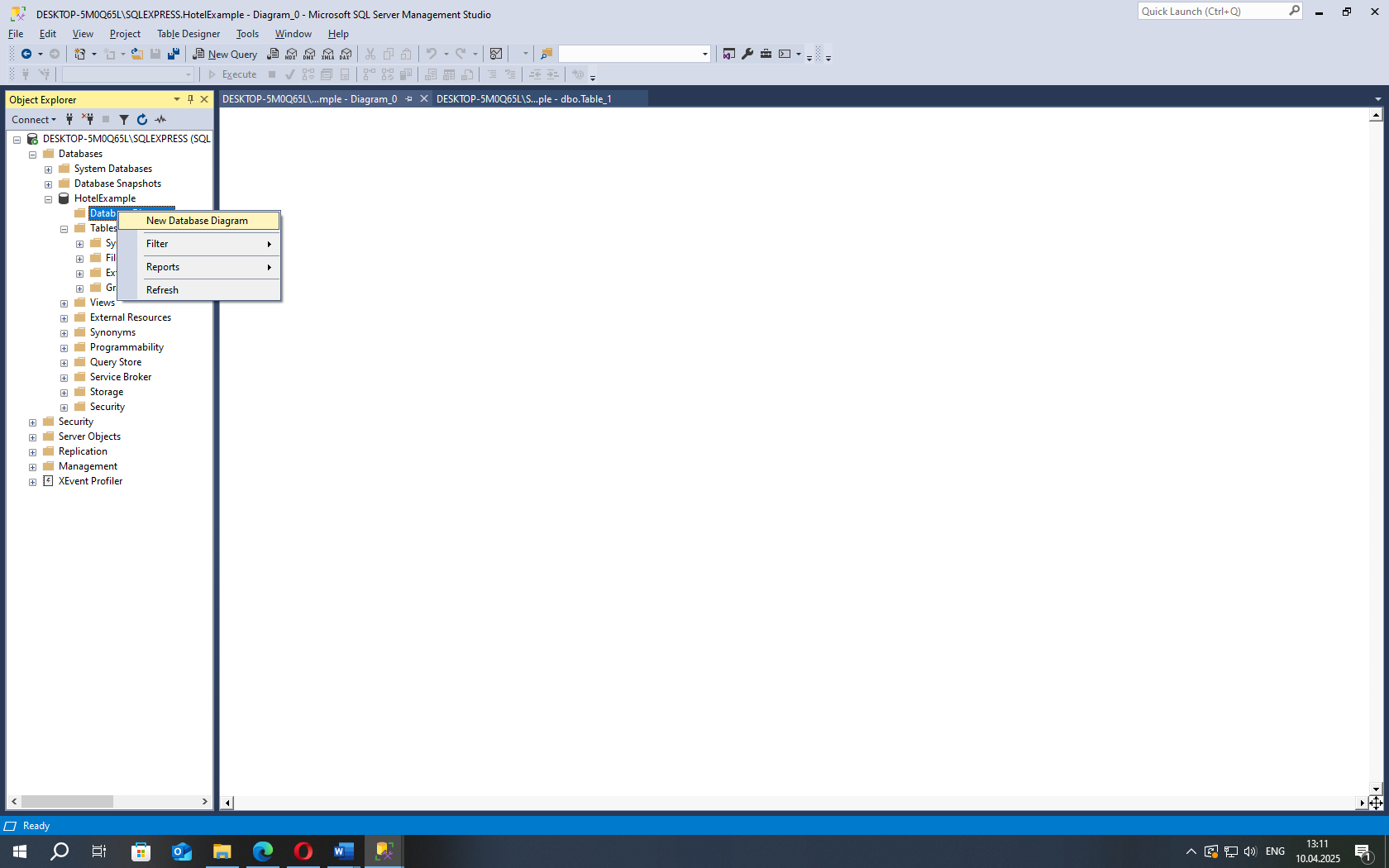
*Рисунок 2.2. Создание новой базы данных*

В программе SSMS необходимо нажать на правую кнопку мыши на папке Database и выбрать New Database для создания новой базы данных.



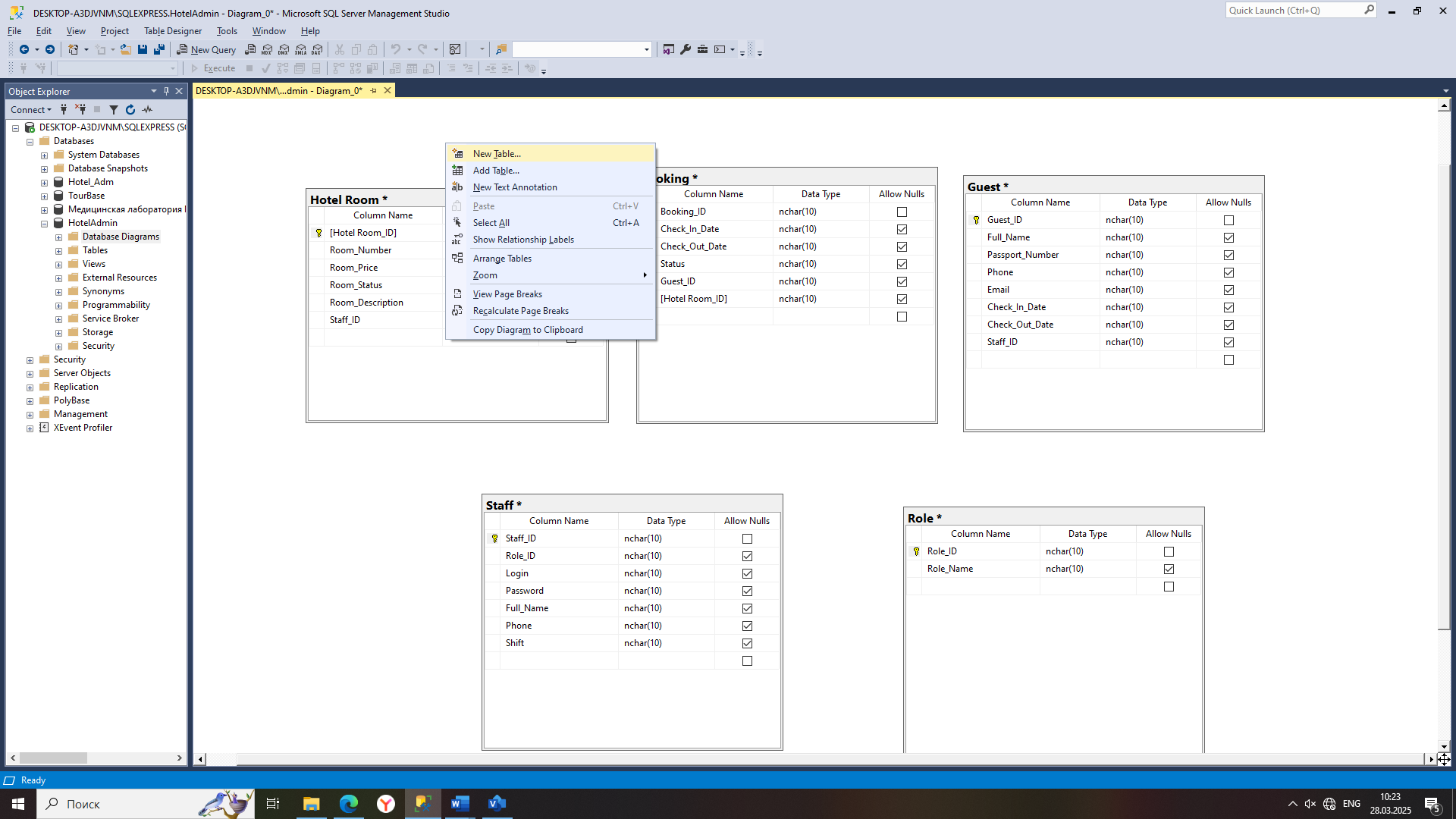
*Рисунок 2.3. Диалоговое окно создания базы данных*

После чего откроется диалоговое окно, в котором есть возможность дать имя базе данных.



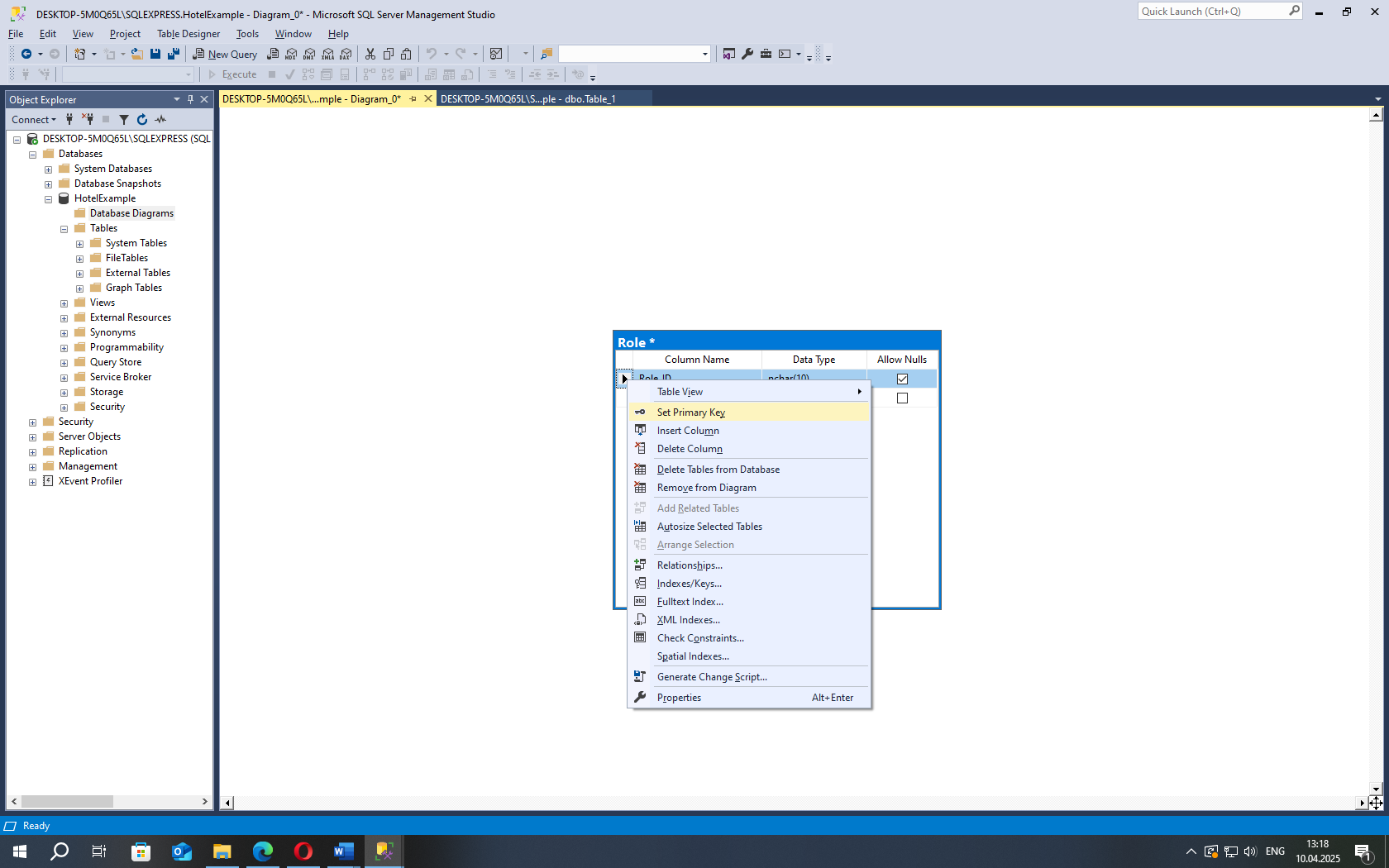
*Рисунок 2.4. Cоздание диаграммы базы данных*

Чтобы создать диаграмму базы данных необходимо нажать правой кнопкой мыши на Database Diagrams и выбрать пункт New Database Diagram.



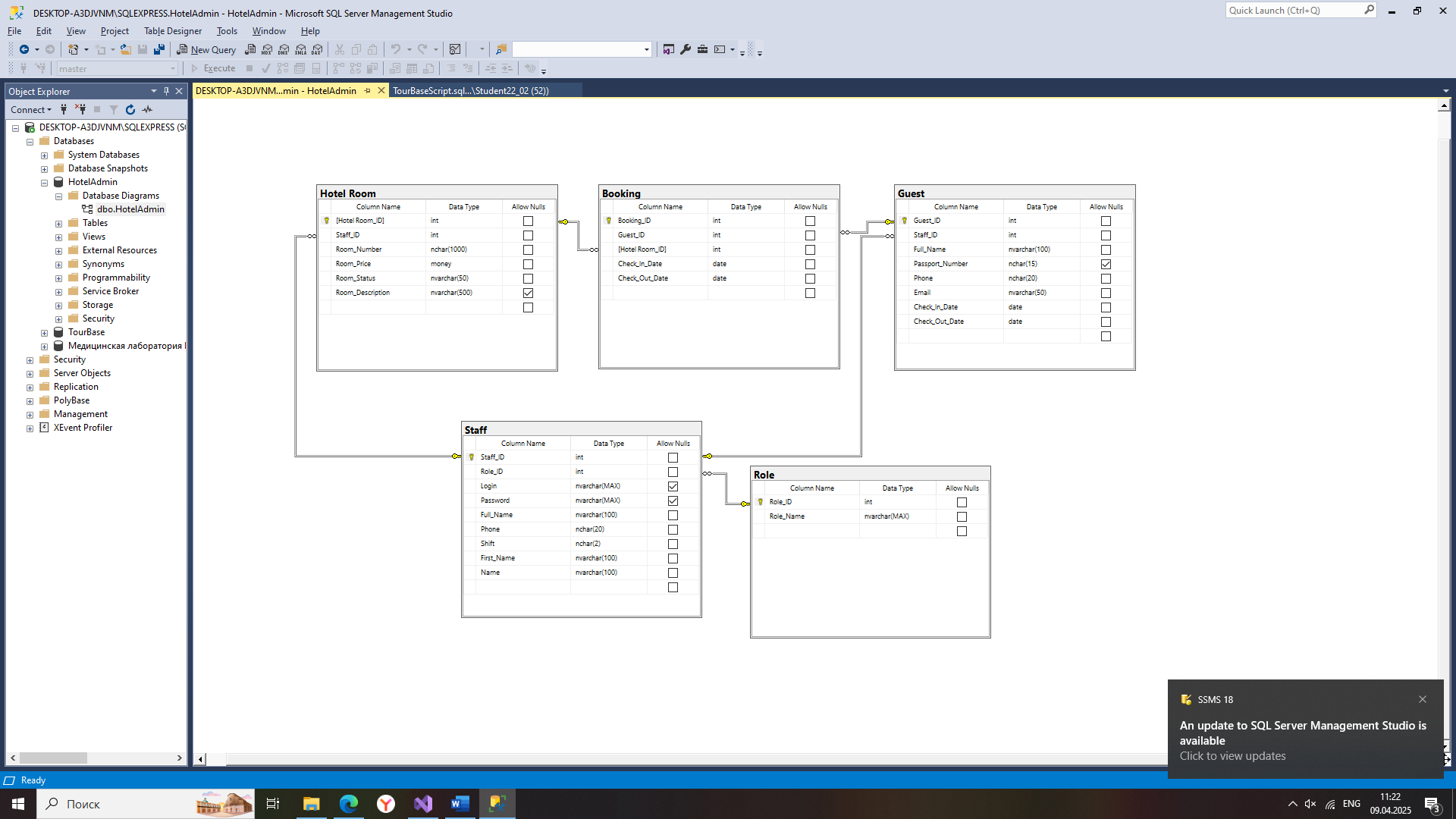
*Рисунок 2.5. Создание таблицы*

После создания базы данных необходимо создать таблицы, для этого нужно нажать на любое место на экране правой кнопкой мыши.



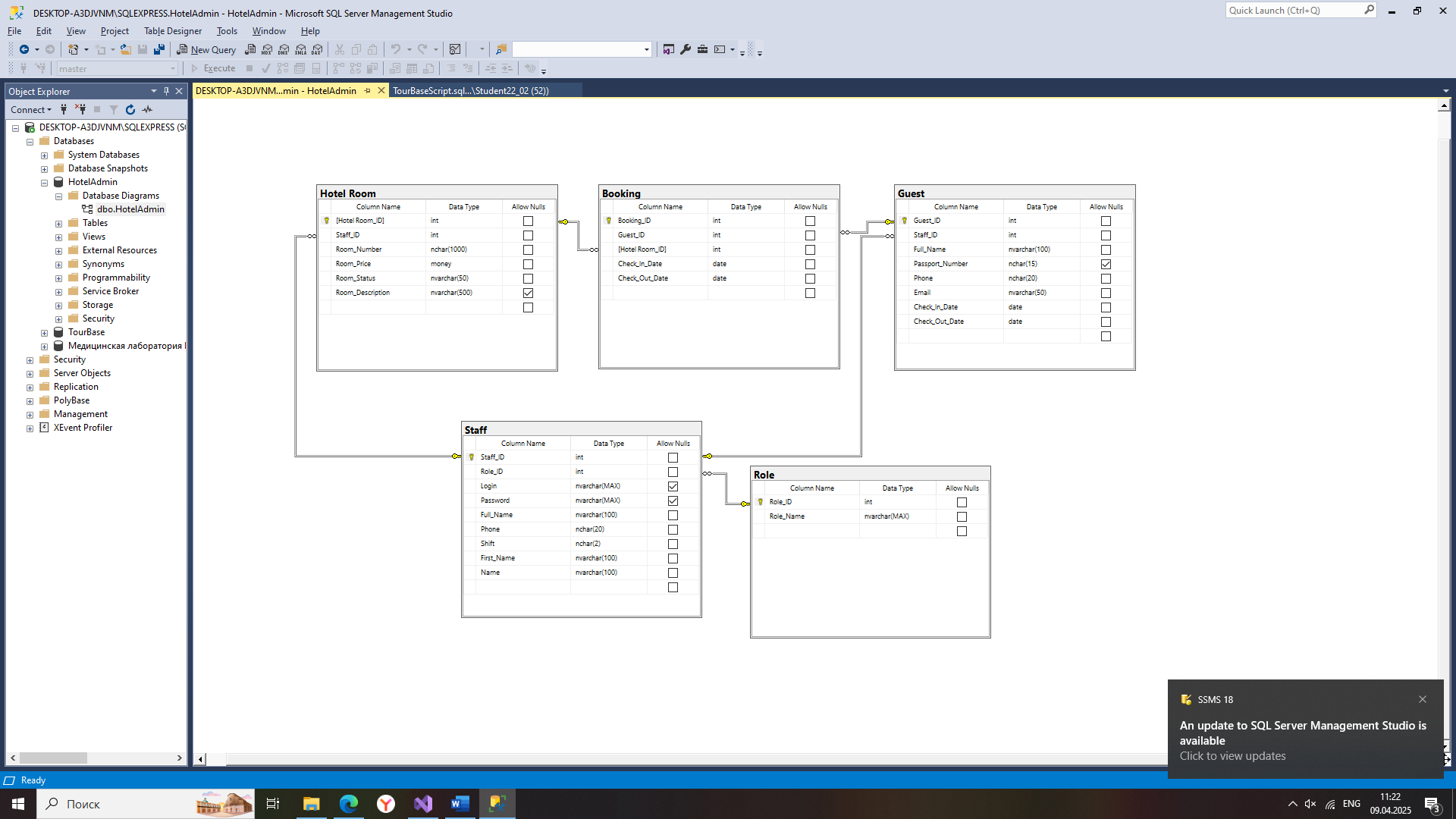
*Рисунок 2.6. Присваивание первичного ключа*

Для создания связей между таблицами следует задать первичные ключи.

**

*Рисунок 2.7. Связь между таблицами*

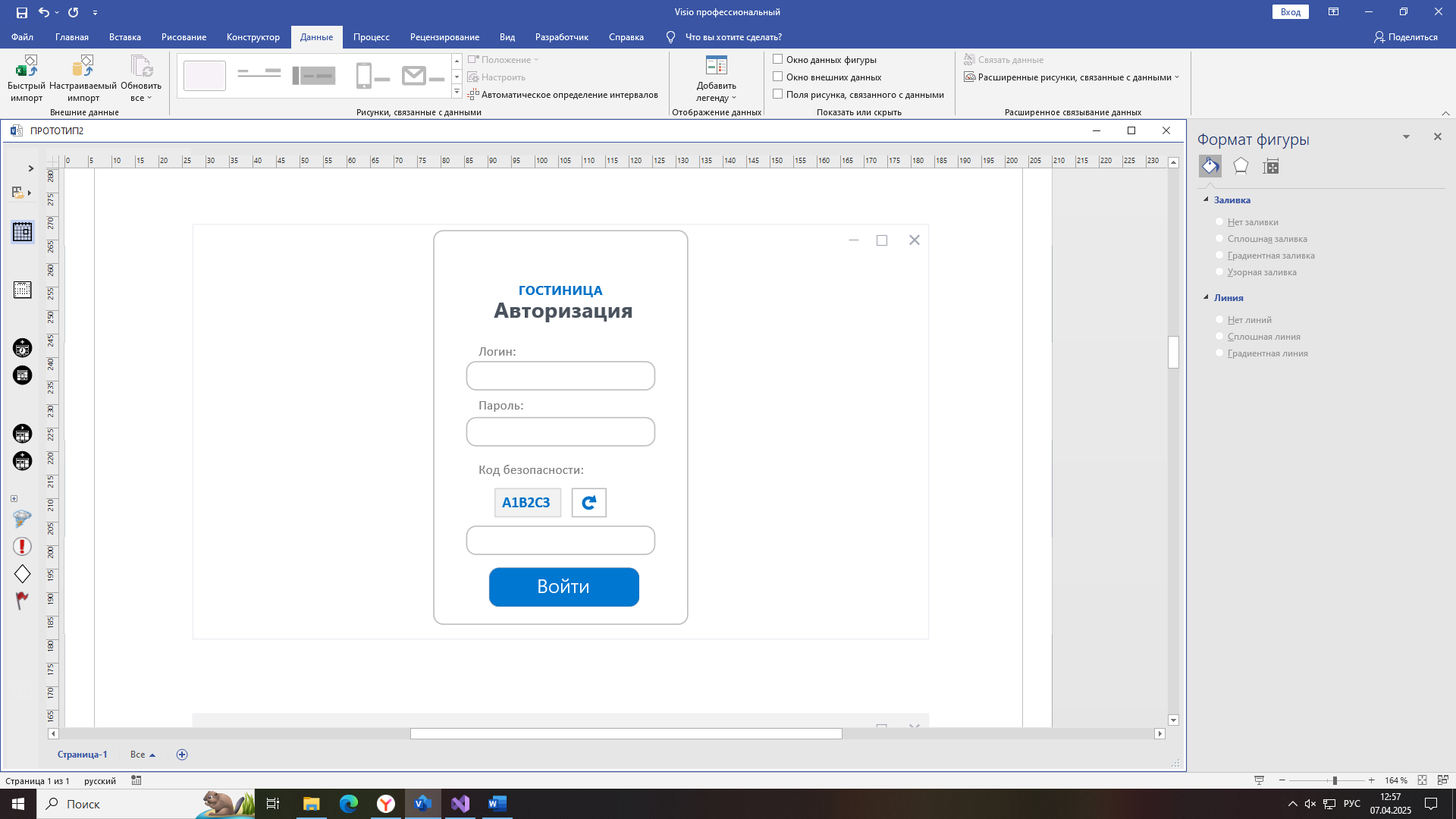
Для создания связей необходимо потянуть мышкой от первичного ключа одной таблицы до вторичного ключа другой таблицы.

**

*Рисунок 2.8. Диаграмма базы данных*

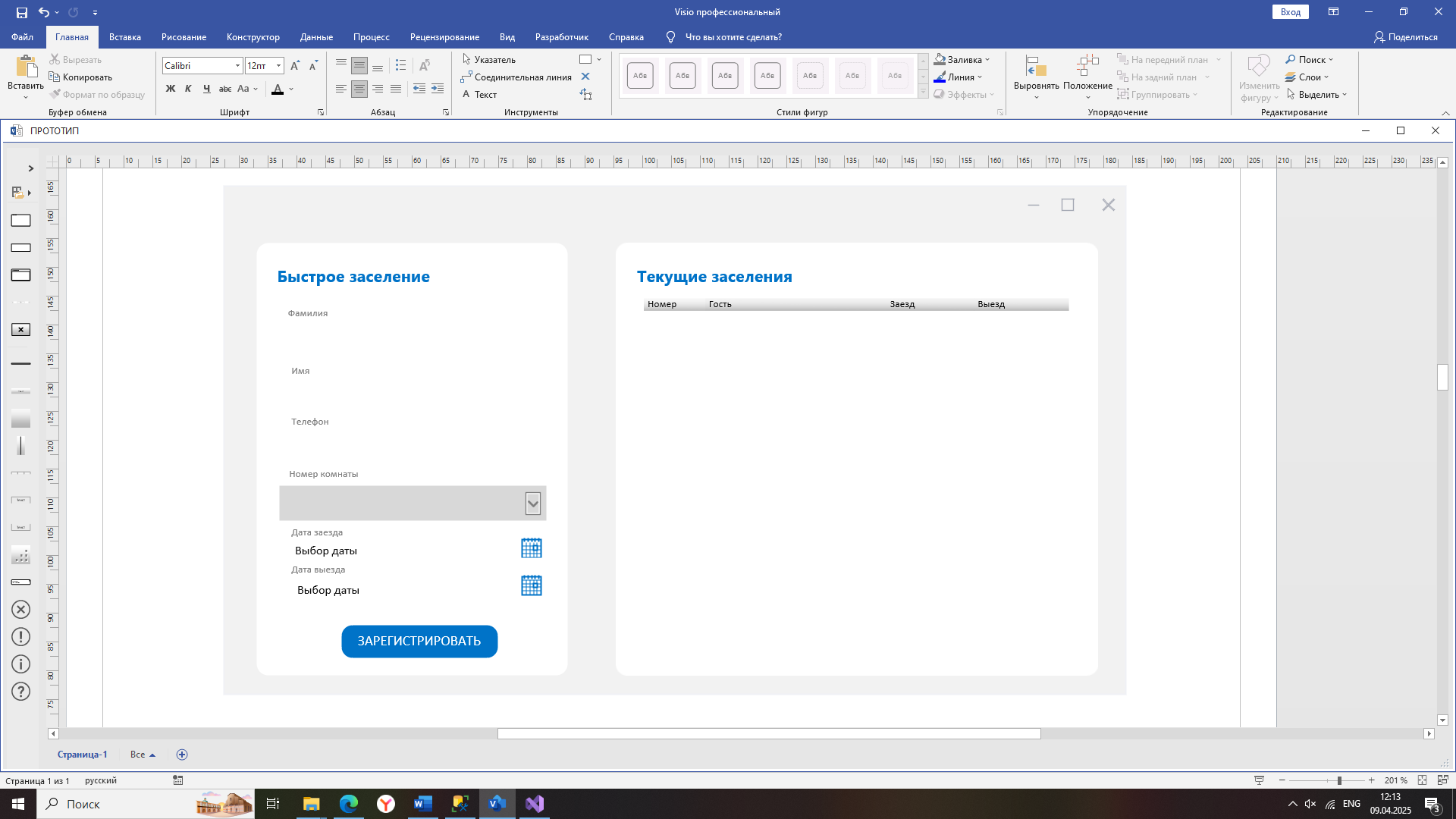
# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 3.1 Разработка прототипа информационной системы



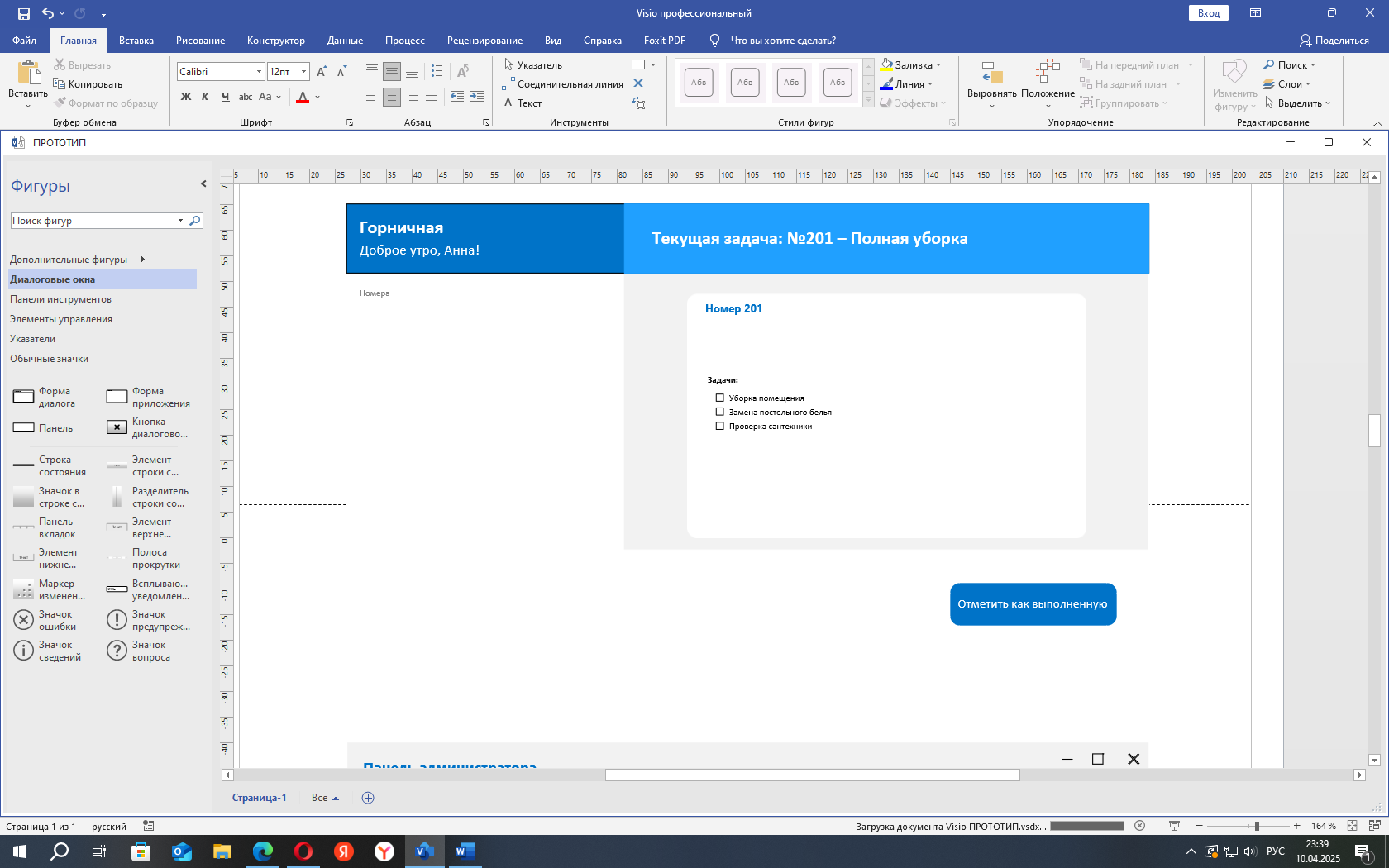
*Рисунок 3.1. Окно авторизации*

Данное окно открывается первым при запуске программы. На окне есть возможность ввести логин и пароль пользователя, а также есть проверка кода безопасности, который надо будет ввести после трех неудачных попыток входа.



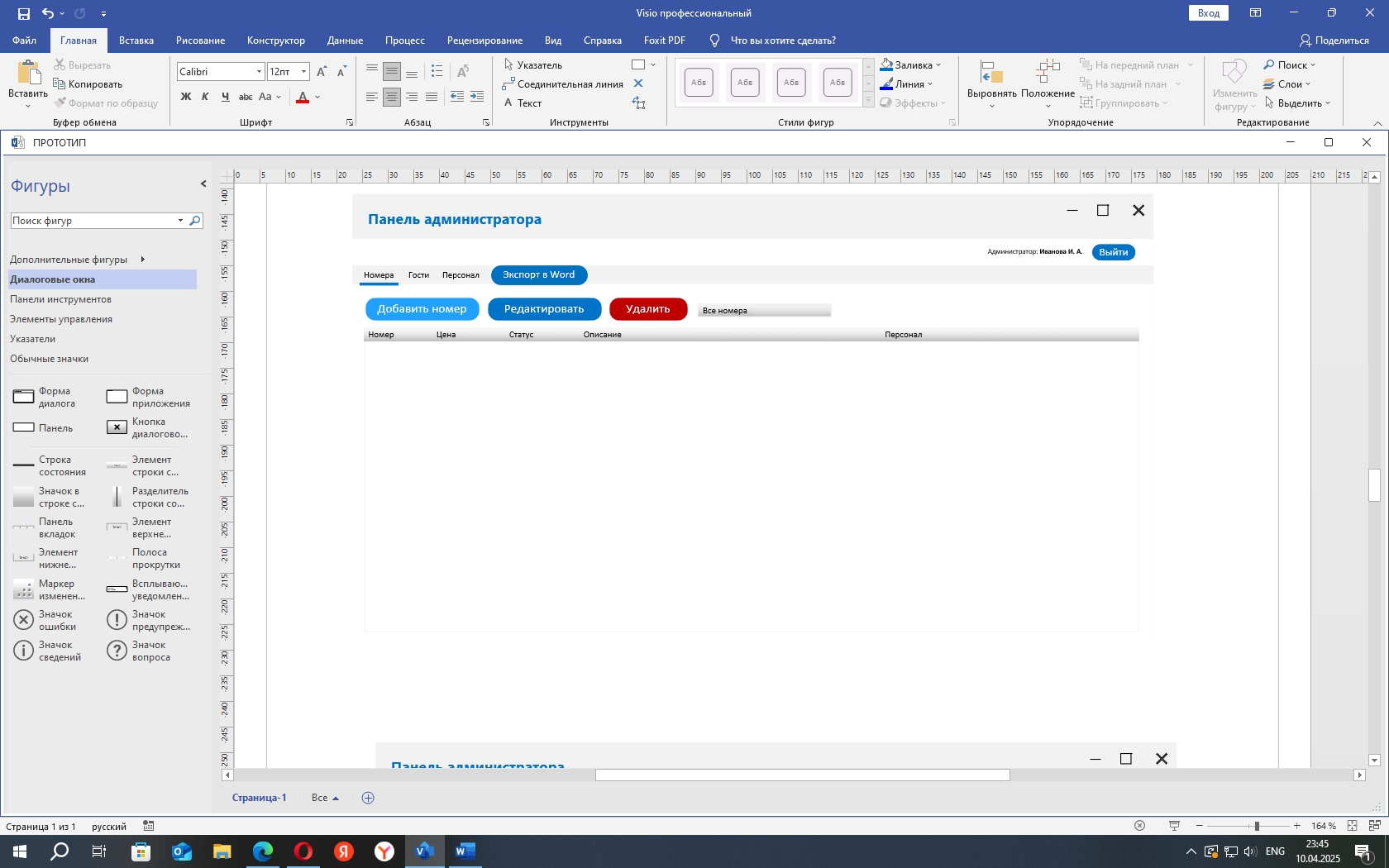
*Рисунок 3.2.* *Главное окно менеджера*

Данное окно открывается при вводе логина manager. На нем есть возможность зарегистрировать гостей, ввести их данные и выбрать номер комнаты и дату заезда. Также у менеджера есть возможность просматривать текущие заселения.



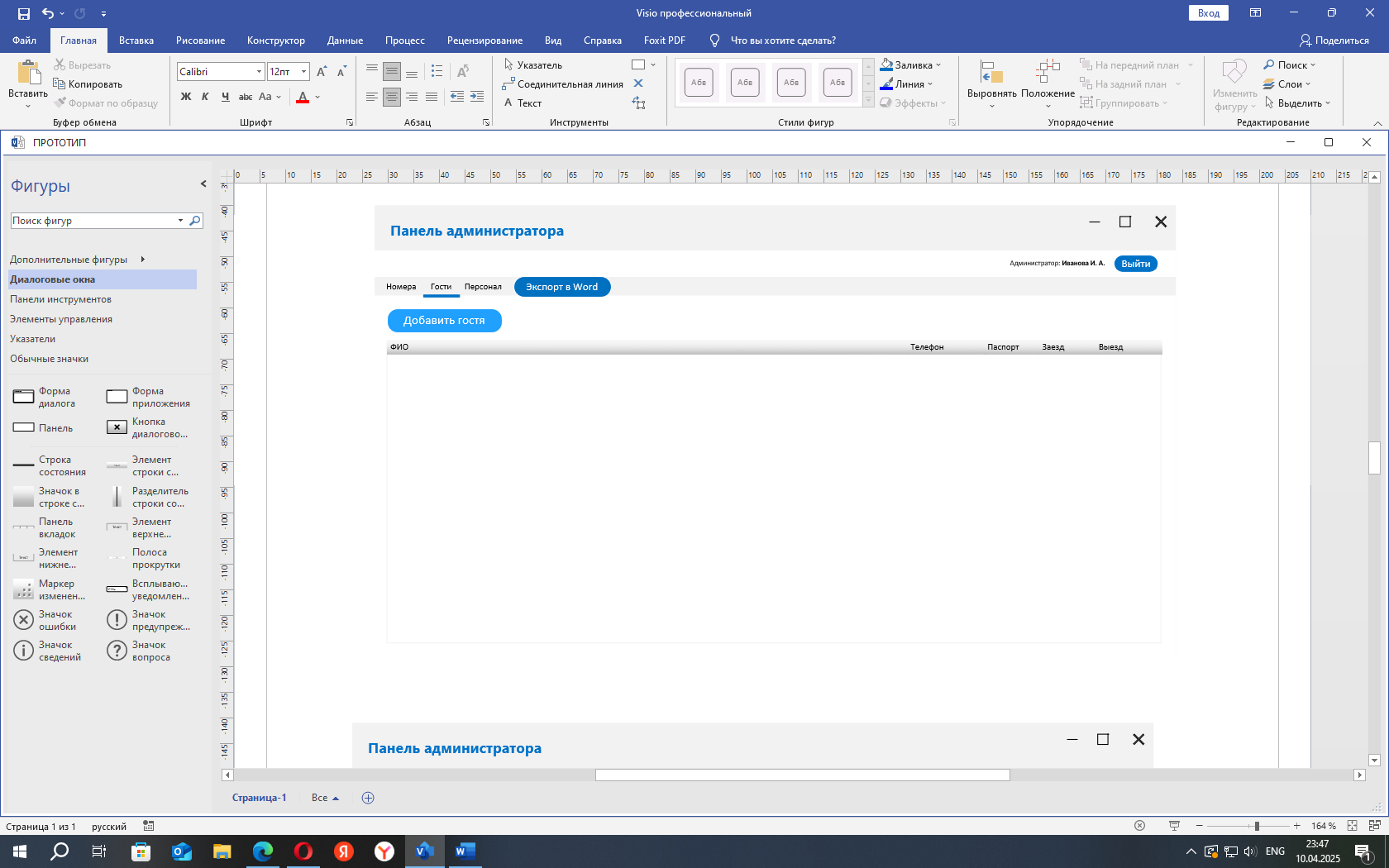
*Рисунок 3.**3. Главное окно горничной*

Данное окно открывается при вводе логина housemaid. У горничной есть возможность просматривать список номеров и указывать выполненные задачи.



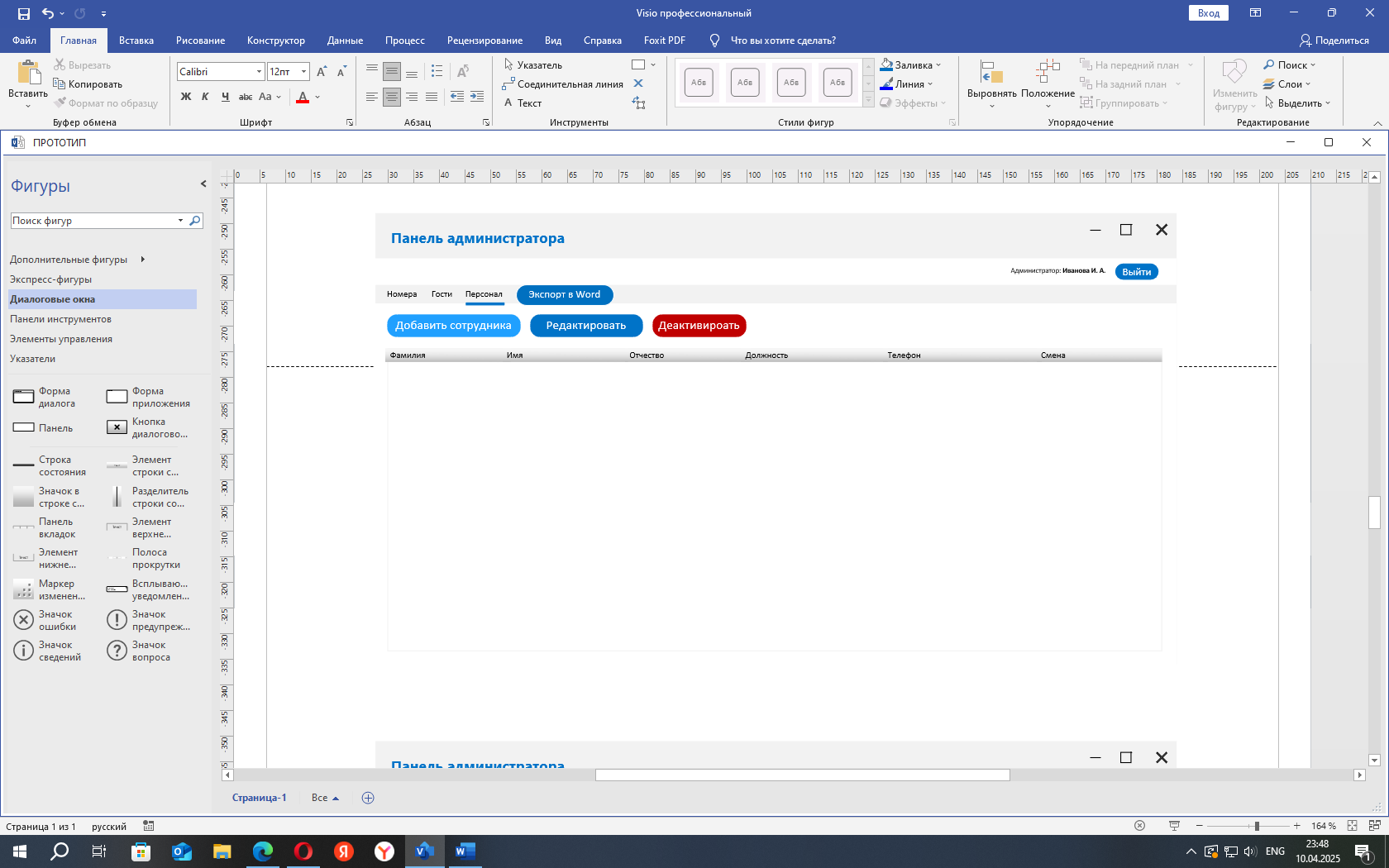
*Рисунок 3.4. Главное окно администратора: Номера*

Данное окно открывается при вводе логина admin. На нем можно просматривать список номеров, гостей, персонала и посмотреть отчеты. Администратор имеет возможность добавить номер, редактировать номер и удалить.



*Рисунок 3.5. Главное окно администратора: Гости*

Во вкладке «Гости» администратор может просматривать список гостей, их данные, а также добавлять гостей.

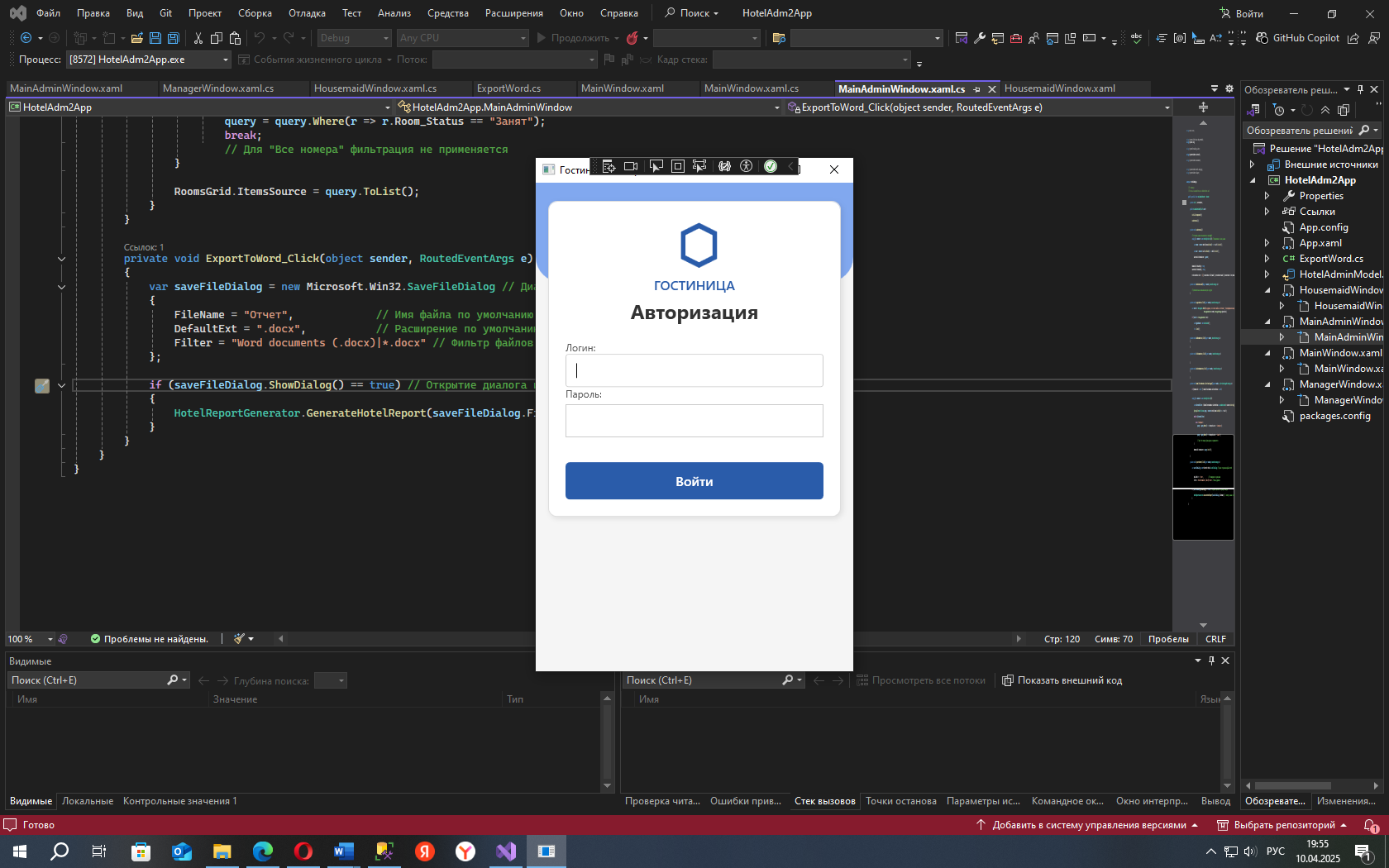


*Рисунок 3.6. Главное окно администратора: Персонал*

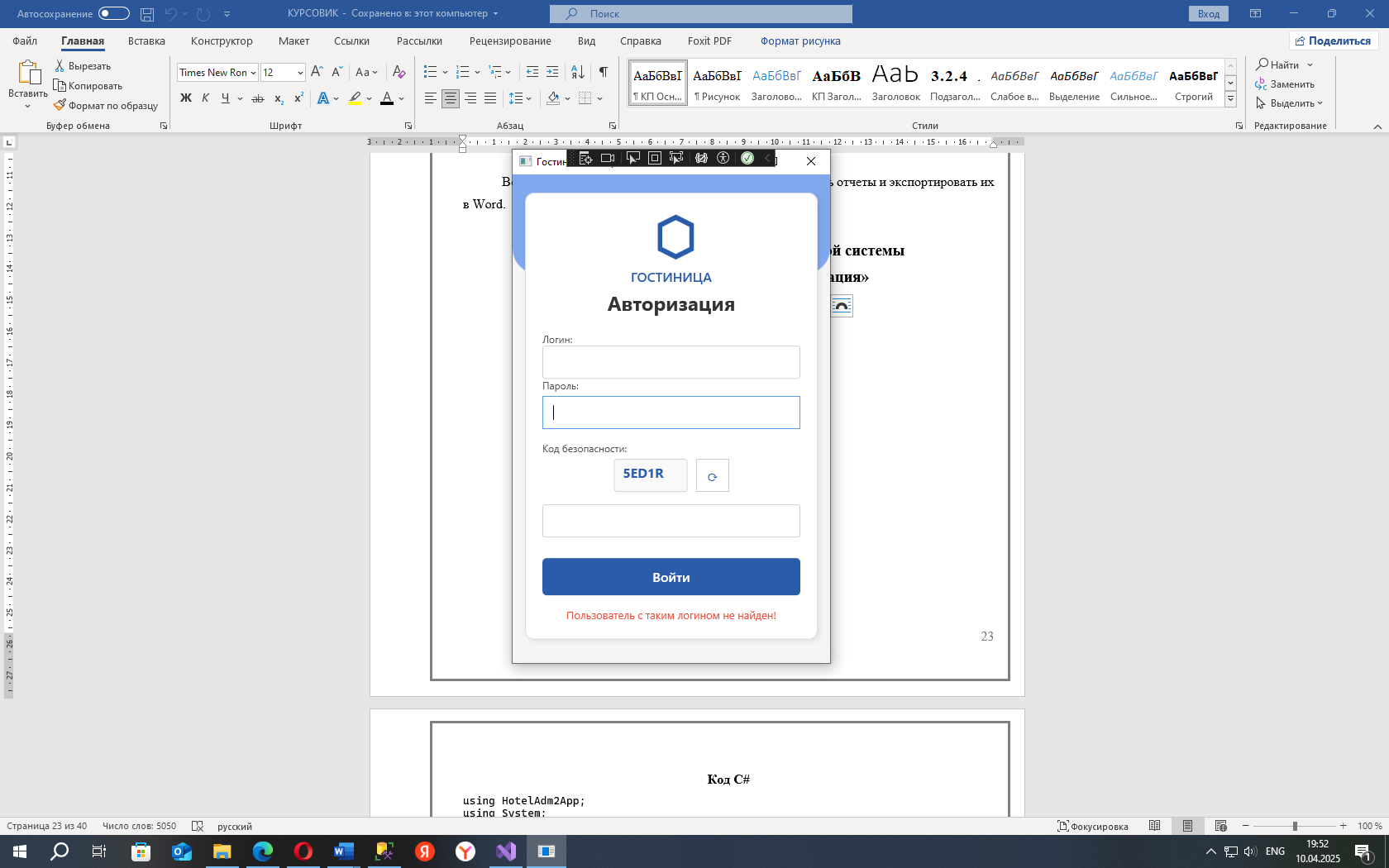
Во вкладке «Персонал» администратор может просматривать список сотрудников, их данные. Также администратор может добавить сотрудника, редактировать список и деактивировать сотрудника.

# 3.2 Программирование информационной системы

# 3.2.1 Разработка модуля «Авторизация»



*Рисунок 3.7. Авторизация*

**

*Рисунок 3.8. Авторизация с капчей*

После 3 неудачных попыток входа появляется окно с кодом подтверждения.

**Код C#**

using HotelAdm2App;

using System;

using System.Linq;

using System.Windows;

using System.Windows.Media.Animation;

namespace HotelAdm2App

{

public partial class MainWindow : Window

{

private readonly Random \_random = new Random();

private int \_loginAttempts;

private bool \_isCaptchaVisible;

public MainWindow()

{

InitializeComponent();

Loaded += delegate (object s, RoutedEventArgs e) { txtUsername.Focus(); };

// Изначально скрываем элементы капчи

txtCaptcha.Visibility = Visibility.Collapsed;

txtCaptchaInput.Visibility = Visibility.Collapsed;

btnRefreshCaptcha.Visibility = Visibility.Collapsed;

CodeTxt.Visibility = Visibility.Collapsed;

BorderCaptcha.Visibility = Visibility.Collapsed;

\_isCaptchaVisible = false;

}

private void GenerateCaptcha()

{

const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

char[] captcha = new char[5];

for (int i = 0; i < 5; i++)

{

captcha[i] = chars[\_random.Next(chars.Length)];

}

txtCaptcha.Text = new string(captcha);

txtCaptchaInput.Clear();

}

private void RefreshCaptcha\_Click(object sender, RoutedEventArgs e)

{

GenerateCaptcha();

}

private void Login\_Click(object sender, RoutedEventArgs e)

{

string username = txtUsername.Text.Trim();

string password = txtPassword.Password.Trim();

string captcha = txtCaptchaInput.Text.Trim();

// Проверка пустых полей

if (string.IsNullOrWhiteSpace(username))

{

HandleLoginFailure("Введите логин!");

return;

}

if (string.IsNullOrWhiteSpace(password))

{

HandleLoginFailure("Введите пароль!");

return;

}

// Проверка капчи, если она видима

if (\_isCaptchaVisible && !ValidateCaptcha(captcha))

{

return;

}

try

{

using (var db = new HotelsqlEntities()) // Заменил GetContext() на прямое создание

{

// Проверка подключения к базе данных

try

{

db.Database.Connection.Open();

db.Database.Connection.Close();

}

catch (Exception ex)

{

HandleLoginFailure($"Ошибка подключения к базе данных: {ex.Message}");

CheckCaptchaVisibility();

return;

}

// Сначала проверяем существование пользователя

var user = db.Staff.FirstOrDefault(s => s.Login == username);

if (user == null)

{

HandleLoginFailure("Пользователь с таким логином не найден!");

CheckCaptchaVisibility();

return;

}

// Отдельно проверяем пароль

if (user.Password != password)

{

HandleLoginFailure("Неверный пароль!");

CheckCaptchaVisibility();

return;

}

// Если дошли сюда, значит логин и пароль верны

OpenDashboardWindow(user);

Close();

}

}

catch (Exception ex)

{

HandleLoginFailure($"Ошибка входа: {ex.Message}");

CheckCaptchaVisibility();

}

}

private bool ValidateCaptcha(string captcha)

{

if (string.IsNullOrWhiteSpace(captcha))

{

HandleLoginFailure("Введите код подтверждения!");

GenerateCaptcha();

return false;

}

if (captcha != txtCaptcha.Text)

{

HandleLoginFailure("Неверный код подтверждения!");

GenerateCaptcha();

return false;

}

return true;

}

private void HandleLoginFailure(string message)

{

ShowError(message);

\_loginAttempts++;

}

private void CheckCaptchaVisibility()

{

if (\_loginAttempts >= 3 && !\_isCaptchaVisible)

{

txtCaptcha.Visibility = Visibility.Visible;

txtCaptchaInput.Visibility = Visibility.Visible;

btnRefreshCaptcha.Visibility = Visibility.Visible;

CodeTxt.Visibility = Visibility.Visible;

BorderCaptcha.Visibility = Visibility.Visible;

\_isCaptchaVisible = true;

GenerateCaptcha();

}

}

private void OpenDashboardWindow(Staff user)

{

Window dashboardWindow = null;

if (user.Role != null)

{

switch (user.Role.Role\_Name)

{

case "Администратор":

dashboardWindow = new MainAdminWindow(user);

break;

case "Менеджер":

dashboardWindow = new ManagerWindow(user);

break;

case "Горничная":

dashboardWindow = new HousemaidWindow(user);

break;

default:

MessageBox.Show($"Роль '{user.Role.Role\_Name}' не поддерживается", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

if (dashboardWindow == null)

{

MessageBox.Show("Не удалось определить роль пользователя", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

dashboardWindow.Show();

}

private void ShowError(string message)

{

txtError.Text = message;

txtError.Visibility = Visibility.Visible;

DoubleAnimation anim = new DoubleAnimation

{

From = 0,

To = 1,

Duration = TimeSpan.FromSeconds(0.3)

};

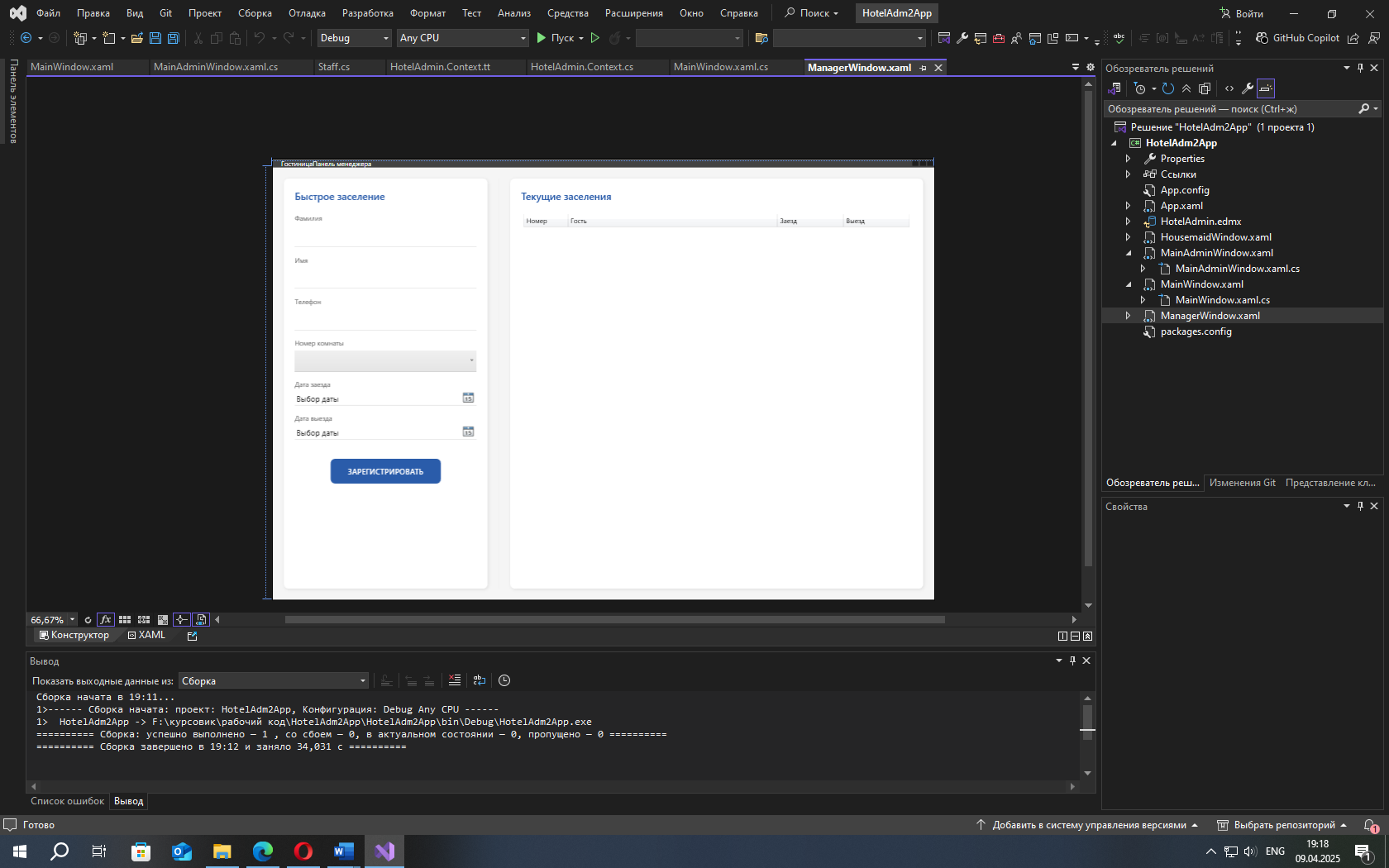
txtError.BeginAnimation(OpacityProperty, anim);

}

}

}

# 3.2.2 Разработка модуля «Главное окно менеджера»

**

*Рисунок 3.9.* Главное окно менеджера

**Код C#**

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace HotelAdm2App

{

/// <summary>

/// Логика взаимодействия для ManagerWindow.xaml

/// </summary>

public partial class ManagerWindow : Window

{

public ObservableCollection<Registration> CurrentRegistrations { get; set; }

public ManagerWindow(Staff user)

{

InitializeComponent();

Loaded += OnWindowLoaded;

// Инициализация коллекции текущих заселений

CurrentRegistrations = new ObservableCollection<Registration>();

CurrentRegistrationsGrid.ItemsSource = CurrentRegistrations;

// Установка дат по умолчанию

dpCheckIn.SelectedDate = DateTime.Now;

dpCheckOut.SelectedDate = DateTime.Now.AddDays(1);

}

private void OnWindowLoaded(object sender, RoutedEventArgs e)

{

// Дополнительная инициализация при загрузке

}

private void RegisterButton\_Click(object sender, RoutedEventArgs e)

{

// Валидация данных

if (string.IsNullOrWhiteSpace(txtLastName.Text) ||

string.IsNullOrWhiteSpace(txtFirstName.Text) ||

string.IsNullOrWhiteSpace(txtPhone.Text) ||

cmbRoomNumber.SelectedItem == null)

{

MessageBox.Show("Пожалуйста, заполните все обязательные поля!",

"Ошибка",

MessageBoxButton.OK,

MessageBoxImage.Warning);

return;

}

// Создание новой регистрации

var newRegistration = new Registration

{

RoomNumber = (cmbRoomNumber.SelectedItem as ComboBoxItem)?.Content.ToString().Split(' ')[0],

GuestName = $"{txtLastName.Text} {txtFirstName.Text}",

CheckInDate = dpCheckIn.SelectedDate ?? DateTime.Now,

CheckOutDate = dpCheckOut.SelectedDate ?? DateTime.Now.AddDays(1)

};

// Добавление в список

CurrentRegistrations.Add(newRegistration);

// Очистка формы

txtLastName.Text = string.Empty;

txtFirstName.Text = string.Empty;

txtPhone.Text = string.Empty;

cmbRoomNumber.SelectedIndex = -1;

dpCheckIn.SelectedDate = DateTime.Now;

dpCheckOut.SelectedDate = DateTime.Now.AddDays(1);

MessageBox.Show("Гость успешно зарегистрирован!",

"Успех",

MessageBoxButton.OK,

MessageBoxImage.Information);

}

private void CurrentRegistrationsGrid\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

}

public class Registration

{

public string RoomNumber { get; set; }

public string GuestName { get; set; }

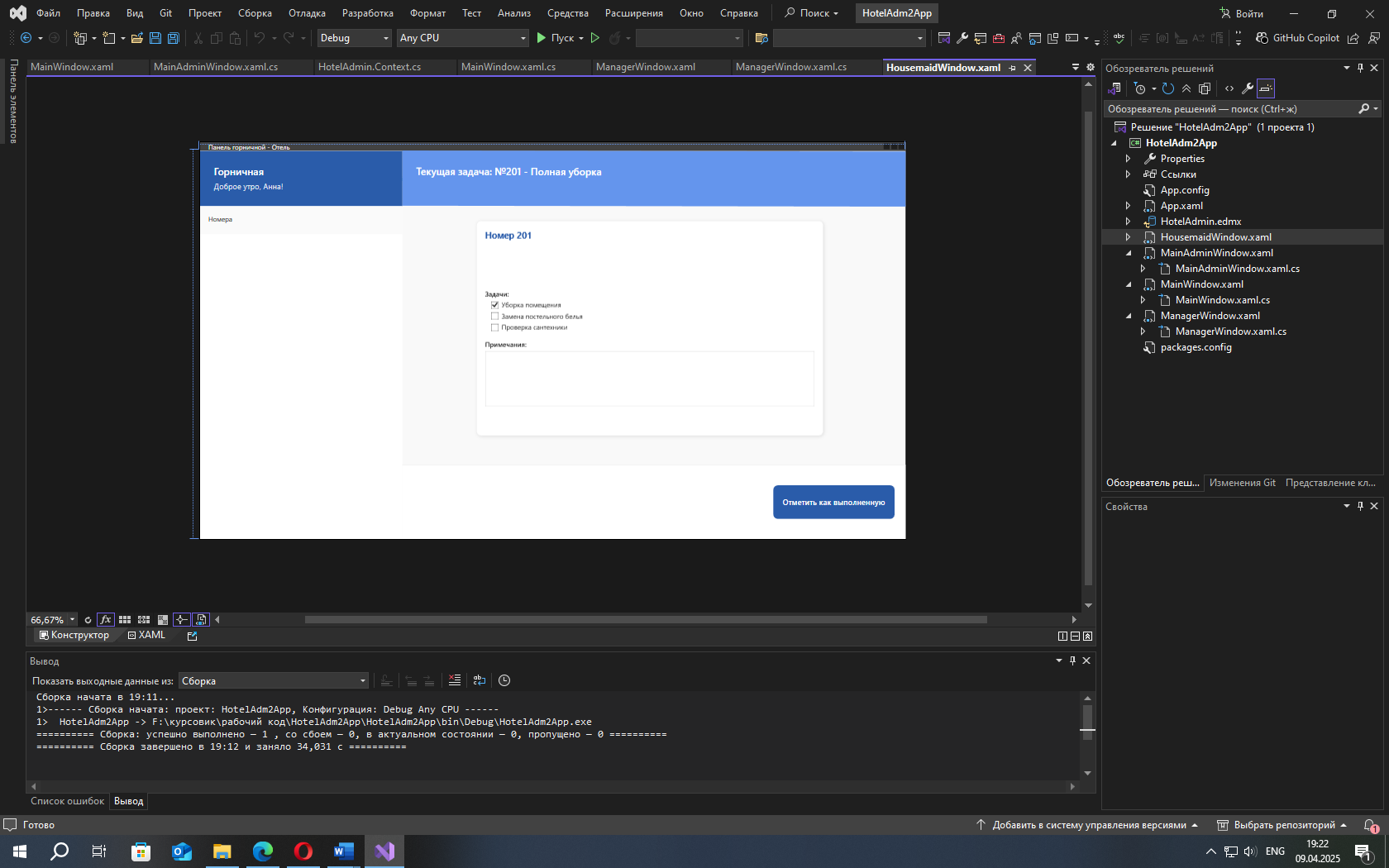
public DateTime CheckInDate { get; set; }

public DateTime CheckOutDate { get; set; }

}

}

# 3.2.3 Разработка модуля «Главное окно горничной»



*Рисунок 3.10.* Главное окно горничной

**Код C#**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace HotelAdm2App

{

/// <summary>

/// Логика взаимодействия для HousemaidWindow.xaml

/// </summary>

public partial class HousemaidWindow : Window

{

public HousemaidWindow(Staff user)

{

InitializeComponent();

Loaded += OnWindowLoaded;

}

private void OnWindowLoaded(object sender, RoutedEventArgs e)

{

}

private string GetDayTime()

{

int hour = DateTime.Now.Hour;

if (hour < 12) return "утро";

if (hour < 18) return "день";

return "вечер";

}

private void CompleteTask\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show("Задача отмечена как выполненная!", "Успех",

MessageBoxButton.OK, MessageBoxImage.Information);

}

private void ComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

}

public class RoomTask

{

public string Number { get; set; }

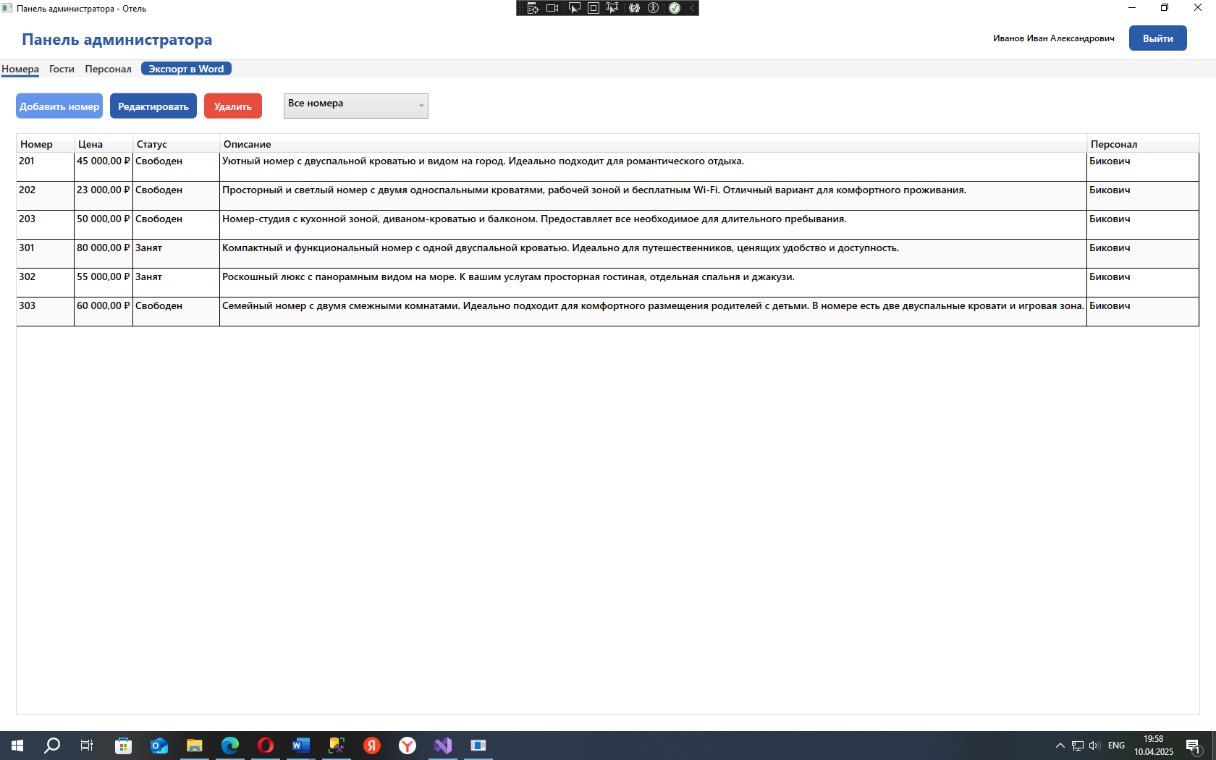
public string Status { get; set; }

public string Time { get; set; }

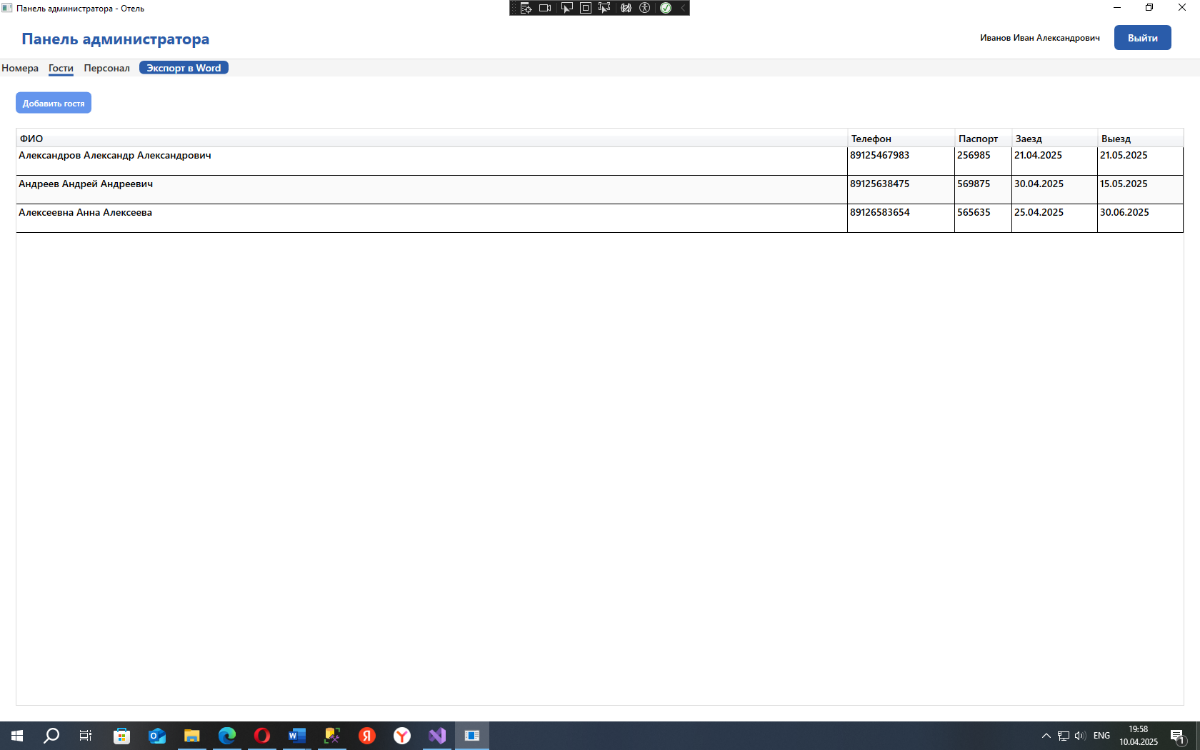
}

}

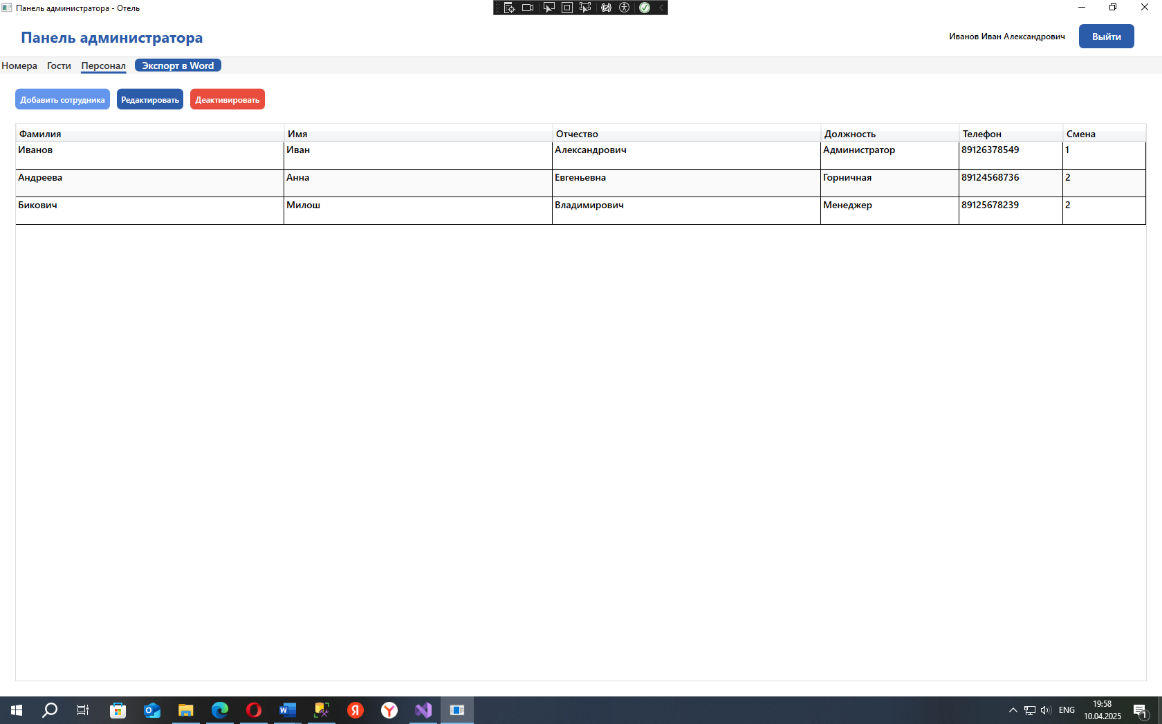
# 3.2.4 Разработка модуля «Главное окно администратора»



*Рисунок 3.11. Главное окно администратора: Номера*

**

*Рисунок 3.12. Главное окно администратора: Гости*

**

*Рисунок 3.13. Главное окно администратора: Персонал*

**Код C#**

using Azure.Core;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Data.Entity;

namespace HotelAdm2App

{

/// <summary>

/// Логика взаимодействия для AdminWindow.xaml

/// </summary>

public partial class MainAdminWindow : Window

{

private Staff \_currentUser;

public MainAdminWindow(Staff user)

{

InitializeComponent();

\_currentUser = user;

LoadUserData();

}

private void LoadUserData()

{

// Установка данных пользователя в интерфейс

using (var context = new HotelsqlEntities()) // Подключение к базе данных

{

var rooms = context.Hotel\_Room.Include(r => r.Staff).ToList();

var guests = context.Guest.Include(g => g.Staff).ToList();

var user = context.Staff.Include(s => s.Role).ToList();

RoomsGrid.ItemsSource = rooms;

GuestsGrid.ItemsSource = guests;

StaffGrid.ItemsSource = user;

}

RoomsGrid.IsReadOnly = true;

GuestsGrid.IsReadOnly = true;

StaffGrid.IsReadOnly = true;

txtCurrentUser.Text = $"{\_currentUser.Full\_Name} {\_currentUser.Name} {\_currentUser.First\_Name}";

}

private void OnWindowLoaded(object sender, RoutedEventArgs e)

{

// Дополнительная инициализация при загрузке

}

private void LogoutButton\_Click(object sender, RoutedEventArgs e)

{

var result = MessageBox.Show("Вы уверены, что хотите выйти из системы?", "Подтверждение выхода",

MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

var loginWindow = new MainWindow();

loginWindow.Show();

this.Close();

}

}

private void AddRoomButton\_Click(object sender, RoutedEventArgs e)

{

}

private void EditRoomButton\_Click(object sender, RoutedEventArgs e)

{

}

private void DeleteRoomButton\_Click(object sender, RoutedEventArgs e)

{

}

private void RoomFilterComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (RoomsGrid == null || RoomFilterComboBox.SelectedItem == null)

return;

using (var context = new HotelsqlEntities())

{

var selectedFilter = (RoomFilterComboBox.SelectedItem as ComboBoxItem)?.Content.ToString();

IQueryable<Hotel\_Room> query = context.Hotel\_Room.Include(r => r.Staff);

switch (selectedFilter)

{

case "Свободные":

query = query.Where(r => r.Room\_Status == "Свободен");

break;

case "Занятые":

query = query.Where(r => r.Room\_Status == "Занят");

break;

// Для "Все номера" фильтрация не применяется

}

RoomsGrid.ItemsSource = query.ToList();

}

}

private void ExportToWord\_Click(object sender, RoutedEventArgs e)

{

var saveFileDialog = new Microsoft.Win32.SaveFileDialog // Диалог сохранения файла Word

{

FileName = "Отчет", // Имя файла по умолчанию

DefaultExt = ".docx", // Расширение по умолчанию

Filter = "Word documents (.docx)|\*.docx" // Фильтр файлов

};

if (saveFileDialog.ShowDialog() == true) // Открытие диалога и проверка выбора

{

HotelReportGenerator.GenerateHotelReport(saveFileDialog.FileName); // Экспорт данных в Word

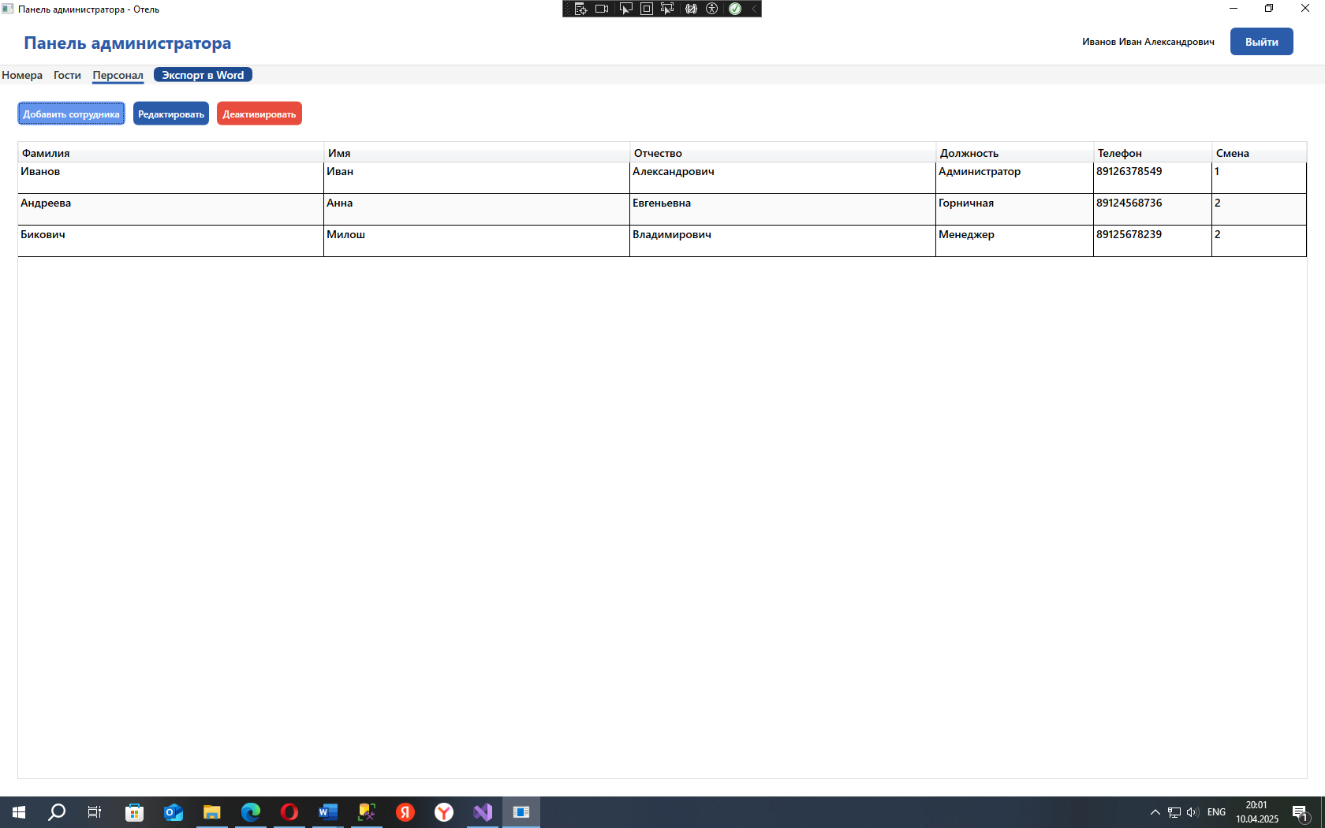
}

}

}

}

# 3.2.5 Разработка модуля «Отчеты»



*Рисунок 3.14. Отчеты*

При нажатии на кнопку Экспорт в Word создается документ с отчетами.

**Код C#**

using HotelAdm2App;

using Microsoft.Office.Interop.Word;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Windows;

using Word = Microsoft.Office.Interop.Word;

namespace HotelAdm2App

{

class HotelReportGenerator

{

public static void GenerateHotelReport(string filePath)

{

try

{

using (var context = new HotelsqlEntities())

{

var data = new

{

Bookings = context.Booking.Include("Hotel\_Room").Include("Guest").ToList(),

Guests = context.Guest.Include("Staff").ToList(),

Staffs = context.Staff.Include("Role").ToList(),

Hotel\_Rooms = context.Hotel\_Room.Include("Staff").ToList(),

Roles = context.Role.ToList()

};

if (File.Exists(filePath)) File.Delete(filePath);

Word.Application wordApp = new Word.Application();

wordApp.Visible = false;

Word.Document doc = wordApp.Documents.Add();

SetDocumentStyles(doc);

AddTitle(doc, "Отчет по управлению отелем");

AddBookingsSection(doc, data.Bookings);

AddPageBreak(doc);

AddGuestsSection(doc, data.Guests);

AddPageBreak(doc);

AddHotelRoomsSection(doc, data.Hotel\_Rooms);

AddPageBreak(doc);

AddStaffSection(doc, data.Staffs);

AddPageBreak(doc);

AddRolesSection(doc, data.Roles);

doc.SaveAs2(filePath);

doc.Close();

wordApp.Quit();

ReleaseWordObjects(doc, wordApp);

OpenGeneratedReport(filePath);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при создании отчета: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private static void OpenGeneratedReport(string filePath)

{

try

{

Process.Start(new ProcessStartInfo(filePath) { UseShellExecute = true });

}

catch (Exception ex)

{

MessageBox.Show($"Не удалось открыть отчет: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private static void SetDocumentStyles(Word.Document doc)

{

doc.Content.Font.Name = "Times New Roman";

doc.Content.Font.Size = 14;

doc.Content.ParagraphFormat.LineSpacing = 18f;

doc.Content.ParagraphFormat.SpaceBefore = 0;

doc.Content.ParagraphFormat.SpaceAfter = 0;

doc.Content.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

}

private static void AddTitle(Word.Document doc, string text)

{

Word.Paragraph title = doc.Paragraphs.Add();

title.Range.Text = text;

title.Range.Font.Bold = 1;

title.Range.Font.Size = 16;

title.Format.SpaceBefore = 0;

title.Format.SpaceAfter = 0;

title.Format.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

title.Range.InsertParagraphAfter();

}

private static void AddPageBreak(Word.Document doc)

{

Word.Paragraph lastParagraph = doc.Paragraphs.Add();

lastParagraph.Range.InsertBreak(Word.WdBreakType.wdPageBreak);

}

private static void AddBookingsSection(Word.Document doc, List<Booking> bookings)

{

AddSectionTitle(doc, "Бронирования");

Word.Table table = CreateTable(doc, new string[] { "ID", "Гость", "Номер", "Дата заезда", "Дата выезда" });

foreach (var item in bookings)

{

AddRowToTable(table, new string[] {

item.Booking\_ID.ToString(),

item.Guest?.Full\_Name ?? "Неизвестно",

item.Hotel\_Room?.Room\_Number ?? "Неизвестно",

item.Check\_In\_Date.ToShortDateString(),

item.Check\_Out\_Date.ToShortDateString()

});

}

FormatTable(table);

}

private static void AddGuestsSection(Word.Document doc, List<Guest> guests)

{

AddSectionTitle(doc, "Гости");

Word.Table table = CreateTable(doc, new string[] { "ID", "ФИО", "Паспорт", "Телефон", "Email", "Заезд", "Выезд", "Сотрудник" });

foreach (var item in guests)

{

AddRowToTable(table, new string[] {

item.Guest\_ID.ToString(),

item.Full\_Name,

item.Passport\_Number,

item.Phone,

item.Email,

item.Check\_In\_Date.ToShortDateString(),

item.Check\_Out\_Date.ToShortDateString(),

item.Staff?.Full\_Name ?? "Неизвестно"

});

}

FormatTable(table);

}

private static void AddHotelRoomsSection(Word.Document doc, List<Hotel\_Room> rooms)

{

AddSectionTitle(doc, "Номера отеля");

Word.Table table = CreateTable(doc, new string[] { "ID", "Номер", "Цена", "Статус", "Описание", "Сотрудник" });

foreach (var item in rooms)

{

AddRowToTable(table, new string[] {

item.Hotel\_Room\_ID.ToString(),

item.Room\_Number,

item.Room\_Price.ToString("C"),

item.Room\_Status,

item.Room\_Description,

item.Staff?.Full\_Name ?? "Неизвестно"

});

}

FormatTable(table);

}

private static void AddStaffSection(Word.Document doc, List<Staff> staff)

{

AddSectionTitle(doc, "Сотрудники");

Word.Table table = CreateTable(doc, new string[] { "ID", "ФИО", "Логин", "Телефон", "Смена", "Роль" });

foreach (var item in staff)

{

AddRowToTable(table, new string[] {

item.Staff\_ID.ToString(),

item.Full\_Name,

item.Login,

item.Phone,

item.Shift,

item.Role?.Role\_Name ?? "Неизвестно"

});

}

FormatTable(table);

}

private static void AddRolesSection(Word.Document doc, List<Role> roles)

{

AddSectionTitle(doc, "Роли");

Word.Table table = CreateTable(doc, new string[] { "ID", "Название роли" });

foreach (var item in roles)

{

AddRowToTable(table, new string[] {

item.Role\_ID.ToString(),

item.Role\_Name

});

}

FormatTable(table);

}

private static void AddSectionTitle(Word.Document doc, string title)

{

Word.Paragraph sectionTitle = doc.Paragraphs.Add();

sectionTitle.Range.Text = title;

sectionTitle.Range.Font.Bold = 1;

sectionTitle.Range.Font.Size = 14;

sectionTitle.Format.SpaceBefore = 0;

sectionTitle.Format.SpaceAfter = 0;

sectionTitle.Format.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

sectionTitle.Range.InsertParagraphAfter();

}

private static Word.Table CreateTable(Word.Document doc, string[] headers)

{

Word.Table table = doc.Tables.Add(doc.Range(doc.Content.End - 1), 1, headers.Length);

for (int i = 0; i < headers.Length; i++)

{

table.Cell(1, i + 1).Range.Text = headers[i];

table.Cell(1, i + 1).Range.Font.Bold = 1;

table.Cell(1, i + 1).Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

}

return table;

}

private static void AddRowToTable(Word.Table table, string[] values)

{

table.Rows.Add();

int rowIndex = table.Rows.Count;

for (int i = 0; i < values.Length; i++)

{

table.Cell(rowIndex, i + 1).Range.Text = values[i] ?? "";

table.Cell(rowIndex, i + 1).Range.Font.Bold = 0;

table.Cell(rowIndex, i + 1).Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

}

}

private static void FormatTable(Word.Table table)

{

table.Columns.AutoFit();

table.Borders.Enable = 1;

foreach (Word.Row row in table.Rows)

{

foreach (Word.Cell cell in row.Cells)

{

cell.VerticalAlignment = Word.WdCellVerticalAlignment.wdCellAlignVerticalCenter;

}

}

}

private static void ReleaseWordObjects(params object[] objects)

{

foreach (var obj in objects)

{

try

{

System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);

}

catch { }

finally

{

GC.Collect();

}

}

}

}

}

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана автоматизированная информационная система «Администратор гостиницы», направленная на оптимизацию и повышение эффективности управления гостиничным бизнесом. Система позволяет автоматизировать ключевые процессы, такие как бронирование номеров, управление персоналом, учет гостей и формирование отчетности, что способствует минимизации ошибок, связанных с человеческим фактором, и улучшению качества обслуживания клиентов.

Для реализации проекта был выбран современный технологический стек, включающий платформу .NET, язык программирования C#, фреймворк WPF для создания графического интерфейса, а также СУБД SQL Server для хранения и управления данными.

В процессе работы были выполнены следующие задачи:

- проведен анализ бизнес-процессов гостиницы и сформулированы требования к системе;

- разработана ER-диаграмма и спроектирована база данных, включающая сущности для управления номерами, гостями, персоналом и бронированиями;

- реализованы модули системы, такие как авторизация, главные окна для администратора, менеджера и горничной, каждый из которых предоставляет функционал, соответствующий роли пользователя.

Разработанная система обладает интуитивно понятным интерфейсом, обеспечивает безопасность данных и может быть интегрирована с другими системами, например, для онлайн-бронирования. Внедрение данной системы позволит гостинице повысить конкурентоспособность, улучшить управляемость бизнес-процессами и повысить удовлетворенность гостей.

<https://github.com/yunochka/Hotel-Admin.git>

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Горнец, Н.Н. Архитектура информационных системы: учебник / Н.Н Горец, А.Г. Рощин. – М.: Академия, 2022. - 320с.

2. ГОСТ Р 7.0.100-2018. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2018. – 52 с.

3. Дейт К. Введение в системы баз данных / К. Дейт. – 8-е изд. – М.: Вильямс, 2020. – 1328 с.

4. Иванова, Е.Л. Разработка алгоритмов динамического ценообразования для гостиничным систем. -М: МГУ, 2022. – 180 с.

5. Корнеев, И.К. Автоматизация управления гостиничным предприятием на основе CRM-систем // Информационные технологии в бизнесе. – 2021. - №3. – С. 45-50.

6. Microsoft Docs. Руководство по Entity Framework Core. – 2024.

7. 1С: Отель. Техническая документация. – 2024.