**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

«Турагентство»

Выполнил студент гр. ИСП-22 /\_\_\_\_\_\_\_/Бородина Эмилия Александровна /

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

**ПРОВЕРИЛ**

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

(подпись) (Ф.И.О.)

Воркута

2025

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ**

**ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ**

**1.1 Платформа .NET**

**1.2 Язык программирования С#**

**1.3 Windows Presentation Foundation (WPF)**

**1.4 СУБД SQL server**

**1.5 Microsoft SQL Server Management Studio**

**1.6 Entity Framework**

**ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ**

**2.1 Разработка диаграммы ERD**

**2.2 Разработка базы данных**

**ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ**

**3.1 Разработка прототипа информационной системы**

**3.2 Программирование информационной системы**

**3.2.1 Разработка модуля «Авторизация»**

**3.2.2 Разработка модуля «Главное окно»**

**3.2.3 Разработка модуля «Страниц»**

**3.2.4 Разработка модуля «Добавить клиента»**

**3.2.5 Разработка модуля «Отчет»**

**ЗАКЛЮЧЕНИЕ**

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

## **ВВЕДЕНИЕ**

Туризм играет ключевую роль в экономике, культуре и социальной жизни, а в условиях глобализации услуги турагентств становятся все более востребованными, что делает эту сферу динамично развивающейся.

Актуальность в разработке приложения для автоматизации турагентств обусловлена необходимостью, направленной на сокращение людских ресурсов и повышение эффективности. Разработка информационной системы позволит минимизировать ошибки, возникающие при бумажном документообороте, а также обеспечит удобный доступ к нужным данным.

**Объект:** информационная система «Турагентство»

**Предмет:** анализ бизнес-процессов «Турагентство»

**Цель:** разработать информационную систему «Турагентство»

**Задачи:**

• выбрать инструментарий;

• спроектировать базу данных;

• разработать информационную систему

# **ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ**

# **ПЛАТФОРМА .NET**

Платформа .NET — это фреймворк от Microsoft, который позволяет использовать одни и те же пространства имён, библиотеки и API для разных языков. Чаще всего это четыре языка из семейства .NET:

C#,Visual Basic,Visual C++,F#.

Когда вы создаёте программу на одном из этих языков, в самом начале вы подключаете пространство имён System. Если бы не .NET, то для каждого из этих языков пришлось бы создавать отдельный System. То есть нарушился бы один из главных принципов программирования — **DRY** (англ. Don’t repeat yourself — не повторяйся).На момент написания статьи наиболее распространён .NET Framework, меньшей популярностью пользуется .NET Core. Возможно, когда вы будете читать эту статью, уже выйдет .NET 5, который объединит в себе оба фреймворка. Поэтому в статье используется название .NET.

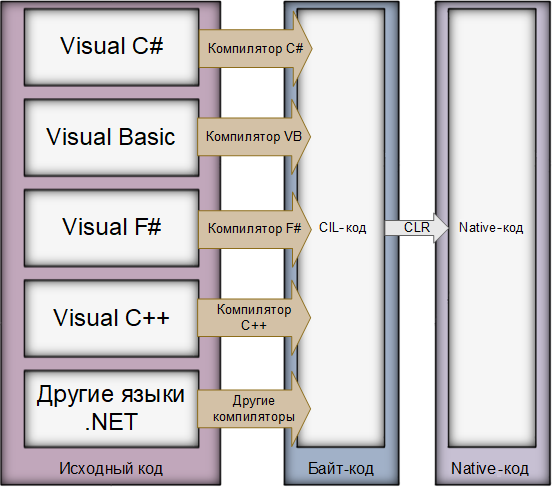
**Для чего нужен .NET**

Обычным пользователям может показаться, что это какие-то программистские штуки, которые никак не влияют на их жизнь. На самом деле в этом есть смысл и для них. Если бы не .NET, пользователям пришлось бы устанавливать среду исполнения для программ на каждом языке. То есть чтобы запустить приложение на Visual Basic, нужно скачать среду выполнения для Visual Basic. Если же программа написана на C#, то придётся скачивать среду и для неё. Это очень быстро забьёт всё место на компьютере немного отличающимися копиями одних и тех же библиотек. Для программистов это тоже важно, потому что даёт возможность развивать одну среду, которая используется сразу для четырёх языков. Иначе обычным разработчикам приходилось бы ждать, пока выйдет новая версия библиотек для их языка. Менее популярные языки, вроде F#, получали бы обновление намного позже C#.

Кроме основных языков есть также и другие, которые поддерживаются .NET. Среди них COBOL, Fortran, Haskell и даже Java — вы можете ознакомиться с полным списком. На этих языках часто написаны старые (legacy) проекты, которые сложно перевести на новую технологию. .NET позволяет переписать часть программы на COBOL под стандарты .NET, а потом просто писать новые части на более современном языке, вроде Visual Basic.

**Как это работает**

Принцип работы достаточно простой, хотя выглядит запутанным. В основном — из-за схожих названий: CLR, CLI и CIL. Для начала посмотрите на это изображение:



*Рис 1.1.1 «Пример принципа работы»*

Это **CLI** (англ. ***Common Language Infrastructure*** — общеязыковая инфраструктура). Она определяет, как работает .NET (а также другие похожие фреймворки вроде Mono и DotGNU).

В CLI у каждого языка есть свой компилятор. Но программы компилируются не в нативный код (исполняемый), а в промежуточный байт-код **CIL** (англ. ***Common Intermediate Language*** — общий промежуточный язык).

Например, если написать программу, которая выводит надпись «Hello, World!», на разных языках, то во всех она скомпилируется в такой промежуточный байт-код:

|  |
| --- |
| .assembly Hello {}  .method public static void Main() cil managed  {  .entrypoint  .maxstack 1  ldstr "Hello, World!"  call void [mscorlib]System.Console::WriteLine(string)  ret  } |

Когда вы запускаете программу, написанную на одном из языков семейства .NET, её байт-код передаётся дальше по цепи в общеязыковую исполняющую среду CLR (***Common Language Runtime***). Там этот байт-код компилируется в нативный и уже начинает выполняться. До 2014 года .NET работал только в операционной системе Windows, однако потом был создан .NET Core — кроссплатформенная версия фреймворка, которая в скором времени заменит основную версию.

### **1.2 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 функционалов. C# — объектно-ориентированный язык, он сделан так, чтобы работать с кодом было проще. Код делится на фрагменты (объекты), каждый из которых можно редактировать и улучшать, не переписывая при этом весь остальной код. Кроме того, в C# есть много функционала, который уже реализован — просто берите его и используйте. Например, если вам нужен алгоритм сортировки, можно не писать его с нуля, он уже есть. Достаточно написать команду “.sort” — и все отсортировано.

У C# синтаксис проще, чем, например, у Python. В нем есть фигурные скобки, которые размечают блоки кода. В результате воспринимать такой код легче. C# — неплохой язык для начинающих, изучить его будет проще, чем тот же C++. Если же вы раньше имели дело с C-подобными языками, то будет еще легче, синтаксис окажется для вас очень знакомым. Сейчас язык C# стал опенсорсным: каждый может внести в него свой вклад, но Microsoft при этом сохраняет контроль, поддерживает философию продукта, выбирает лучшие предложения. Обновление языка происходит раз в год.

# **1.3 WINDOWS PRESENTATION FOUNDATION (WPF)**

Платформа Windows Presentation Foundation (WPF) позволяет создавать клиентские приложения для настольных систем Windows с привлекательным пользовательским интерфейсом.

В основе WPF лежит независимый от разрешения векторный модуль визуализации, использующий возможности современного графического оборудования. Возможности этого модуля расширяются с помощью комплексного набора функций разработки приложений, которые включают в себя язык XAML, элементы управления, привязку к данным, макет, двумерную и трехмерную графику, анимацию, стили, шаблоны, документы, мультимедиа, текст и типографические функции. WPF является частью .NET, поэтому вы можете создавать приложения, включающие другие элементы .NET API. Программирование с помощью WPF ­­­­- WPF существует в виде подмножества типов .NET, которые по большей части находятся в пространстве имен System.Windows. Если ранее вы создавали приложения с помощью .NET, используя управляемые технологии, такие как ASP.NET и Windows Forms, основные принципы программирования с помощью WPF должны быть вам знакомы: вы создаете экземпляры классов, задаете свойства, вызываете методы и обрабатываете события — все это с использованием своего любимого языка программирования .NET, например C# или Visual Basic. WPF включает в себя дополнительные конструкции программирования, которые расширяют возможности свойств и событий: свойства зависимостей и перенаправленные события.

Разметка и код программной части. WPF позволяет разрабатывать приложения, используя как разметку, так и код программной части, что привычно для разработчиков на ASP.NET. Разметка XAML обычно используется для определения внешнего вида приложения, а управляемые языки программирования (код программной части) — для реализации его поведения. Такое разделение внешнего вида и поведения имеет ряд преимуществ.

Затраты на разработку и обслуживание снижаются, так как разметка, определяющая внешний вид, не связана тесно с кодом, обуславливающим поведение. Повышается эффективность разработки, так как дизайнеры, занимающиеся внешним видом приложения, могут работать параллельно с разработчиками, реализующими поведение приложения. Глобализация и локализация приложений WPF упрощена.

**1.4 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. Изначально язык SQL был разработан в компании IBM для системы баз данных, которая называлась System/R. При этом сам язык назывался SEQUEL (Structured English Query Language). Хотя в итоге ни база данных, ни сам язык не были впоследствии официально опубликованы, по традиции сам термин SQL нередко произносят как "сиквел".

В 1979 году компания Relational Software Inc. разработала первую систему управления баз данных, которая называлась Oracle и которая использовала язык SQL. В связи с успехом данного продукта компания была переименована в Oracle. Впоследствии стали появляться другие системы баз данных, которые использовали SQL. В итоге в 1989 году Американский Национальный Институт Стандартов (ANSI) кодифицировал язык и опубликовал его первый стандарт. После этого стандарт периодически обновлялся и дополнялся. Последнее его обновление состоялось в 2011 году. Но несмотря на наличие стандарта нередко производители СУБД используют свои собственные реализации языка SQL, которые немного отличаются друг от друга.

Выделяются две разновидности языка SQL: PL-SQL и T-SQL. PL-SQL используется в таких СУБД как Oracle и MySQL. T-SQL (Transact-SQL) применяется в SQL Server. Собственно, поэтому в рамках текущего руководства будет рассматриваться именно T-SQL. В зависимости от задачи, которую выполняет команда T-SQL, он может принадлежать к одному из следующих типов:

DDL (Data Definition Language / Язык определения данных). К этому типу относятся различные команды, которые создают базу данных, таблицы, индексы, хранимые процедуры и т.д. В общем определяют данные.

В частности, к этому типу мы можем отнести следующие команды:

CREATE: создает объекты базы данных (саму базу данных, таблицы, индексы и т.д.)

ALTER: изменяет объекты базы данных

DROP: удаляет объекты базы данных

TRUNCATE: удаляет все данные из таблиц

DML (Data Manipulation Language / Язык манипуляции данными). К этому типу относят команды на выбор данных, их обновление, добавление, удаление - в общем все те команды, с помощью которыми мы можем управлять данными.

К этому типу относятся следующие команды:

SELECT: извлекает данные из БД

UPDATE: обновляет данные

INSERT: добавляет новые данные

DELETE: удаляет данные

DCL (Data Control Language / Язык управления доступа к данным). К этому типу относят команды, которые управляют правами по доступу к данным. В частности, это следующие команды:

GRANT: предоставляет права для доступа к данным

REVOKE: отзывает права на доступ к данным

**1.5 MICROSOFT SQL SERVER MANAGEMENT STUDIO**

SQL Server Management Studio (SSMS) — это интегрированная среда для управления любой инфраструктурой SQL, от SQL Server до базы данных SQL Azure. SSMS предоставляет средства для настройки, мониторинга и администрирования экземпляров SQL Server и баз данных. Используйте SSMS для развертывания, мониторинга и обновления компонентов уровня данных, используемых приложениями, а также для создания запросов и скриптов. Используйте SSMS для запросов, проектирования и управления базами данных и хранилищами данных, где бы они ни находились — на локальном компьютере или в облаке. Клиентам, которым требуется кроссплатформенный компаньон SSMS для управления SQL и другими базами данных Azure, используйте Azure Data Studio. Microsoft SQL Server Management Studio (SSMS) — это утилита для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Она включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера.

Некоторые возможности SSMS:

Обозреватель объектов. Графический пользовательский интерфейс для просмотра и управления объектами в каждом экземпляре SQL Server. Редактор SQL кода. Позволяет писать и выполнять SQL запросы и инструкции. Просмотр плана выполнения запроса. Позволяет выявлять наиболее ресурсоёмкие операции в запросе для оптимизации скорости его выполнения. Обозреватель решений. Помогает управлять решениями и проектами, сгруппированными из одного или нескольких взаимосвязанных файлов. Конструктор таблиц. Визуальный инструмент для разработки таблиц в базах данных. SSMS реализована только под Windows.

# **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.

Способы взаимодействия с БД

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных

Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере.

Code first: разработчик создает класс модели данных, которые будут храниться в бд, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

# **ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ**

# **2.1 РАЗРАБОТКА ДИАГРАММЫ ERD**

Создание ER-диаграммы, или диаграммы "Сущность-Связь", позволяет визуализировать структуру базы данных и взаимосвязи между её компонентами. Вот основные шаги для создания ER-диаграммы:

1. Определите сущности: выделите основные объекты, такие как "Клиент", "Тур", "Туроператор", "Бронь" и т.д.

2. Определите атрибуты: определите свойства каждой сущности, например, у "Клиента" могут быть "Имя", "Email", "Phone".

3. Определите связи: определите, как сущности взаимодействуют друг с другом. Например, "Клиент" может делать "Бронь", а "Бронь" может содержать "Тур".

4. Выберите тип связи: определите кардинальность:

● «Один-к-одному» — один экземпляр сущности связан только с одним экземпляром другой сущности. Например, пассажир рейса и его место в самолете.

● «Один-ко-многим» — один экземпляр сущности связан со множеством экземпляров другой сущности. Например, у одного пассажира может быть несколько единиц багажа, при этом каждая единица багажа может быть связана только с одним пассажиром.

● «Многие-ко-многим» — множество экземпляров одной сущности связаны со множеством экземпляров другой сущности. Например, аэропорт обслуживает несколько авиакомпаний. При этом каждая авиакомпания может обслуживаться в нескольких аэропортах.

5. Нарисуйте диаграмму: используйте программное обеспечение или инструменты, такие как Lucidchart, draw.io или Microsoft Visio, чтобы визуализировать сущности, атрибуты и связи.

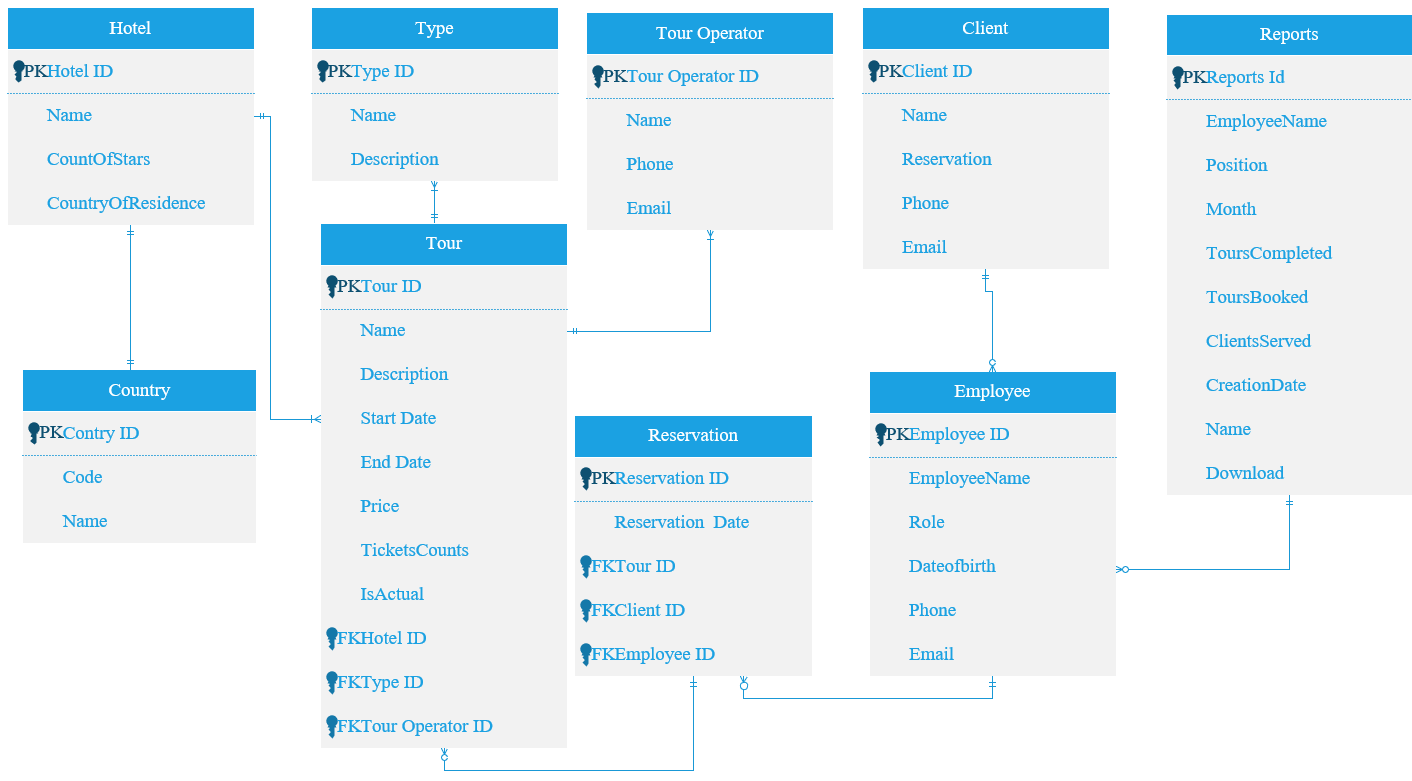
Это поможет вам создать четкое представление о структуре вашей базы данных. Модели «сущность-связь» традиционно используют для разработки программного обеспечения. При этом для метода нет конкретной области разработки: для создания любого ПО нужно работать с данными и транслировать их пользователям. Поэтому ER-модели строят и для интернет-магазина, и для корпоративного портала компании.

Обычно ER-модель создают в двух случаях:

● когда перед началом проекта ещё не понятно, с какими данными предстоит работать;

● когда нужно создать новую базу данных или добавить таблицу в уже существующую.

Чем больше в системе сущностей и связей, тем важнее построить ER-модель до начала разработки ПО. На практике над простыми системами можно работать без концептуальной ER-модели. Например, программа для выдачи талонов электронной очереди — простая система, в которой всего две сущности — номер окна и номер очереди.



*Рис. 2.1.1 «ER-Diagram по системе Турагентство»*

# **2.2 Разработка базы данных**

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

SQL Server Management Studio (SSMS) — это интегрированная среда для управления любой инфраструктурой SQL. Используйте SSMS для доступа, настройки, управления, администрирования и разработки всех компонентов SQL Server, Базы данных SQL Azure, Управляемого экземпляра SQL Azure, SQL Server на виртуальной машине Azure и Azure Synapse Analytics. SSMS предоставляет единую комплексную служебную программу, которая сочетает в себе широкую группу графических инструментов с множеством многофункциональных редакторов сценариев для предоставления доступа к SQL Server разработчикам и администраторам баз данных любого уровня квалификации.

Среда SQL Server Management Studio (SSMS) позволяет управлять объектами служб Analysis Services, например выполнять резервное копирование и обработку объектов.

Для того чтобы создать в программу SSMS базу данных необходимо нажать правой кнопкой мышки по пункту в меню справа Databases (Базы данных) и нажать на пункт New Database (Создать Базу данных) как показано на рисунке 2.2.1.

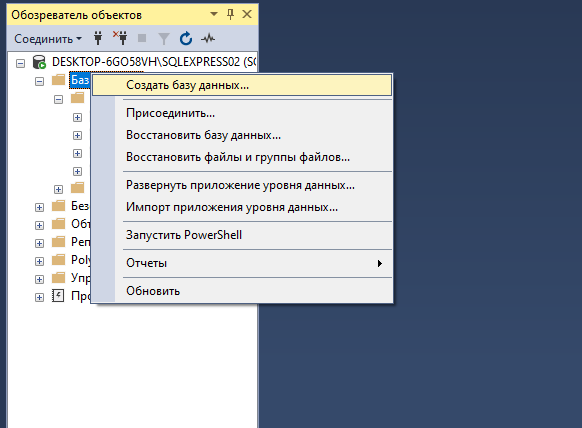


Рисунок 2.2.1 Меню создания базы данных

После откроется диалоговое окно, в котором нужно дать название Базе данных как показано на рисунке 2.2.2

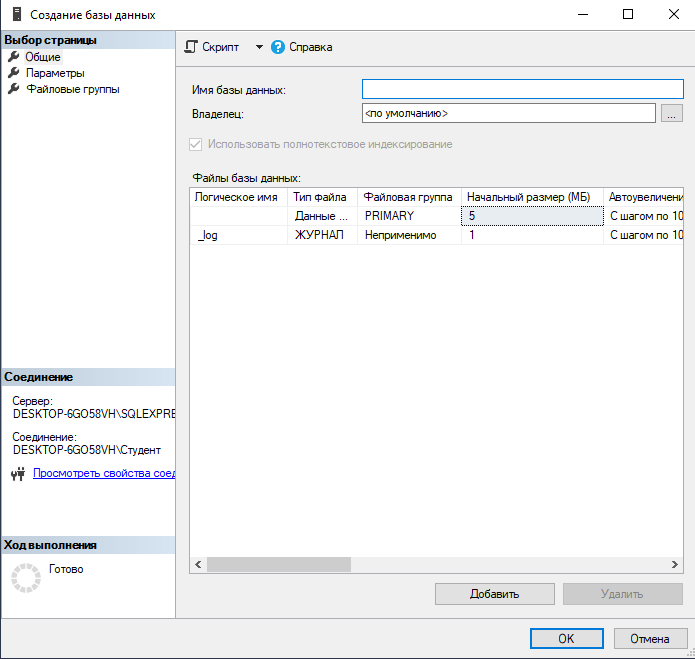
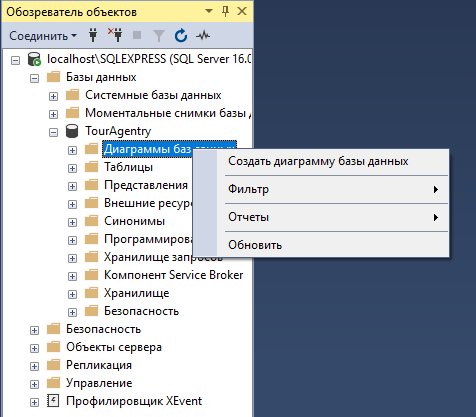


Рисунок 2.2.2 Диалоговое окно создания базы данных

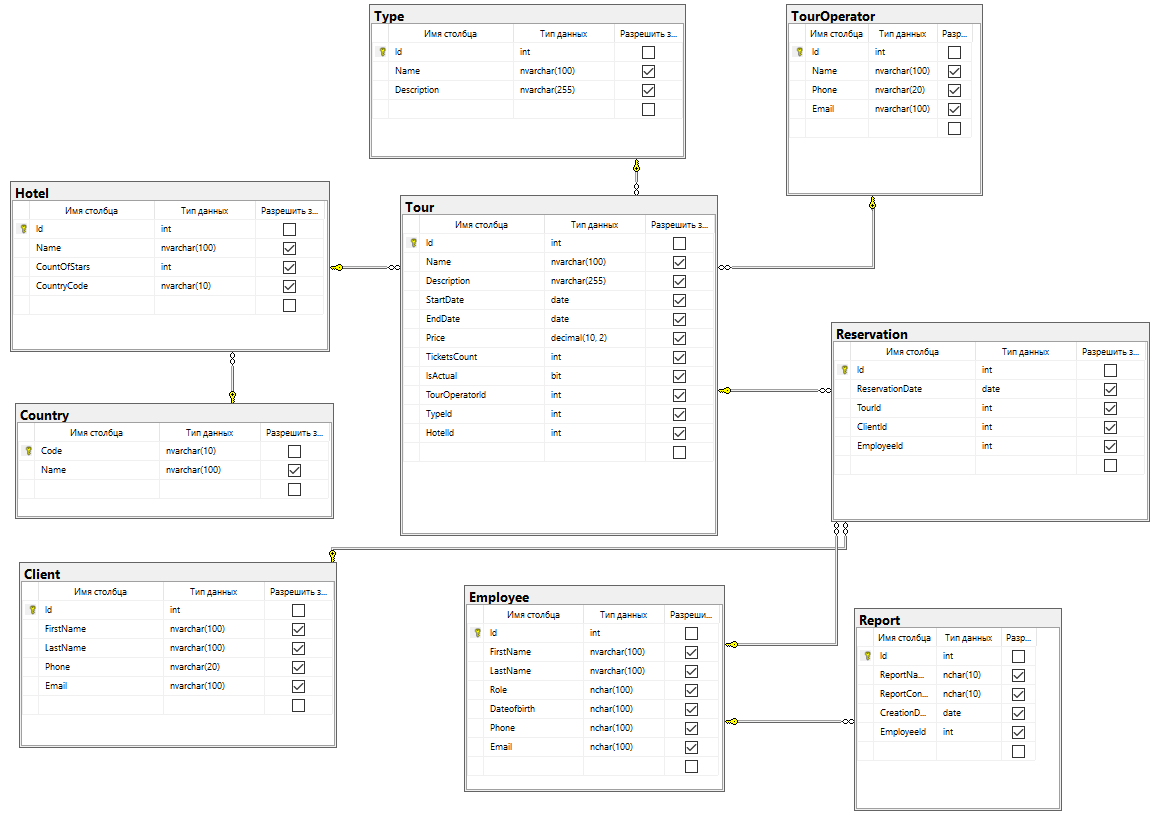
Для того чтобы создать диаграмму базы данных необходимо раскрыть в меню базу данных и нажать правой кнопкой мыши по Database Diagrams (Диаграммы базы данных) и выбрать пункт New Database Diagram (Создать Диаграмму Базы данных) как показано на рисунке 2.3.



*Рисунок 2.2.3 Диалоговое окно создания диаграмм базы данных*

После этого нам покажется поле для создания диаграмм, на котором мы можем создавать таблицы с названием столбцов, их типом данных и зависимостями.

Для создания таблицы данных необходимо нажать правой кнопкой мыши в любом месте диаграммы, выбираем «New Table…»(Новая таблица…) и даем название для нашей таблицы. Для информационной системы «Турагентство» необходимо создать девять таблиц: Employee (Сотрудник), Tour\_operator (Тур оператор), Reservation (Бронирование туров), Tour (Туры), Hotel (Отели), Client(Клиент), Country(Страна где находится отель), Type(Тип тура), Report(Отчет).После чего заполним таблицы необходимыми полями и типами данных из ERD.



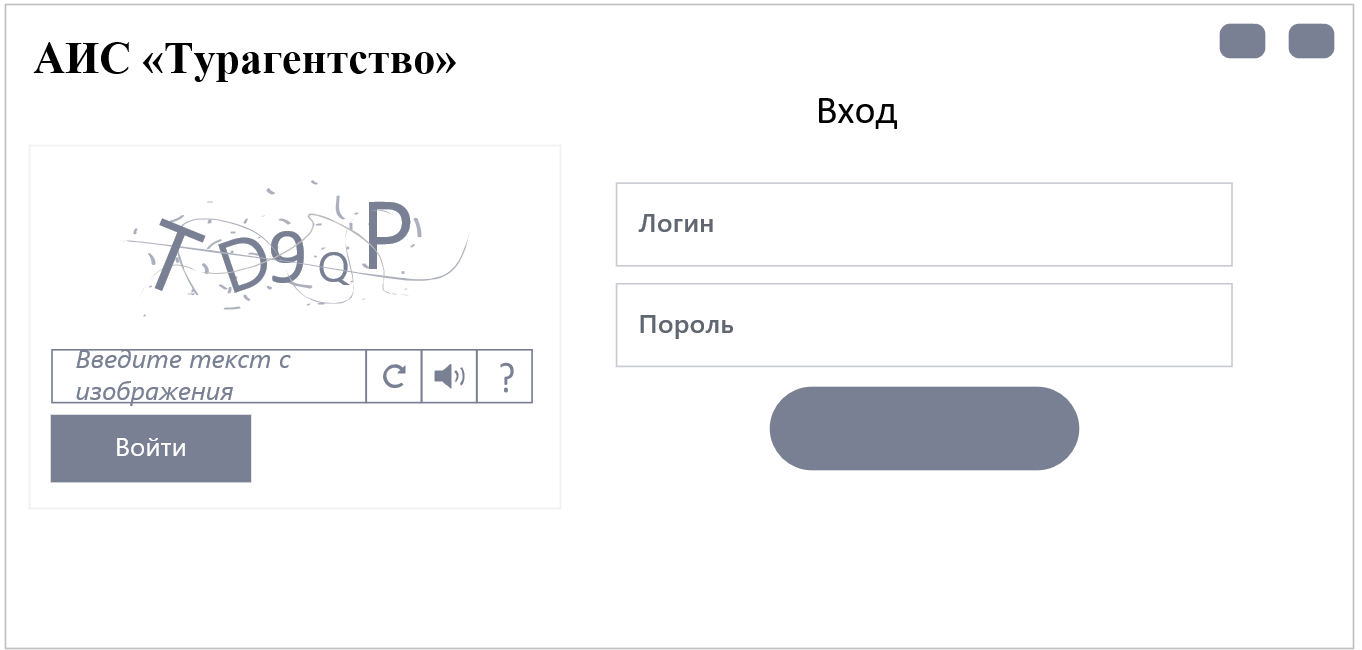
*Рисунок 2.2.4 Все таблицы базы данных*

Далее создаем первичный ключ и удерживая ключ создаем связи как в ERD сущностях. После сохраняем таблицы сочетанием клавиш «Ctrl + S» и в высветившимся окне пишем название диаграммы.

Созданные таблицы можно увидеть в «Object Explorer» кликнув на название базы данных и перейдя в папку «Tables».

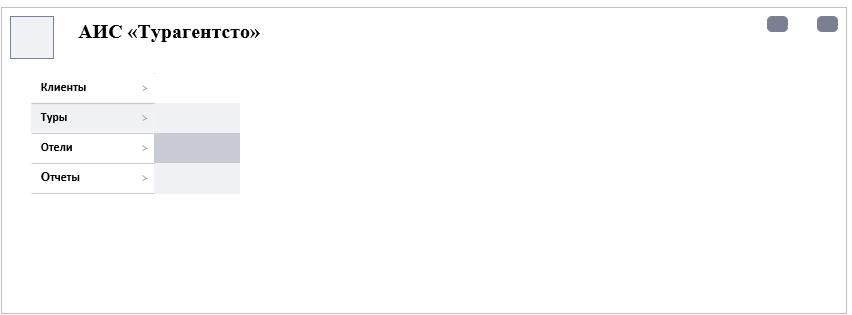
# **ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ**

## **3.1 Разработка прототипа информационной системы**



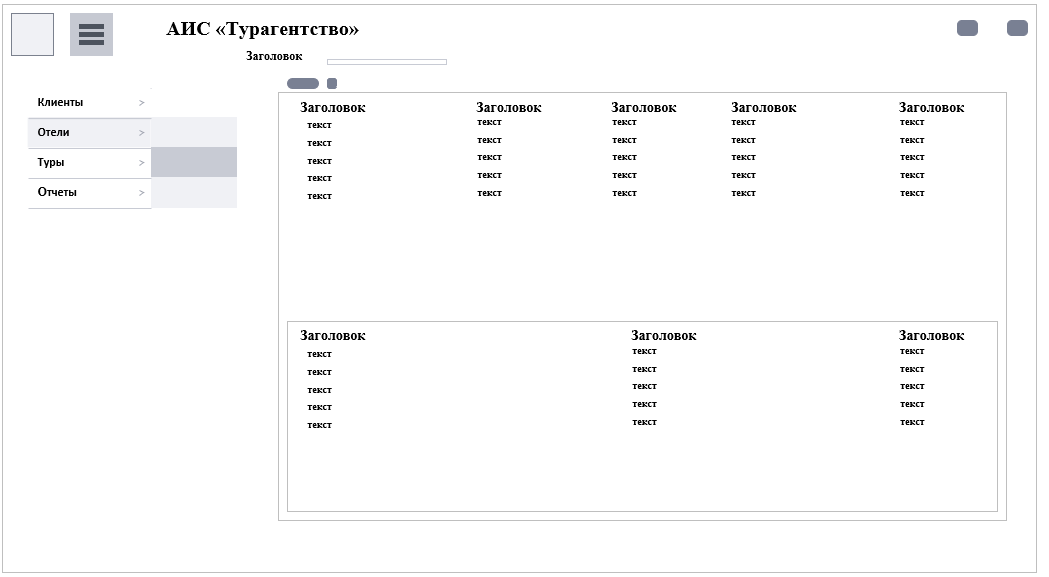
*Рис 3.1.1 Окно авторизации*

На рисунке 3.1.1 представлено окно «Авторизация». На котором выполняется вход в систему и прохождение капчи.



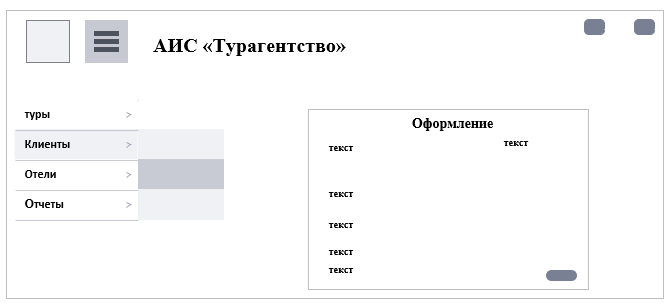
*Рис 3.1.2 Главное окно*

На рисунке 3.1.2 представлено «Главное окно», на котором показы страницы для взаимодействий.

**

*Рис 3.1.3 Страницы*

На рисунке 3.1.3 представлено окно «Страницы», на котором отображается как пример одна из страниц, где выводится список данных и взаимодействий.

**

*Рис 3.1.4 Добавить клиента*

На рисунке 3.1.4 представлено окно «Добавить клиента», на котором сотрудник может ввести данные и сохранить их.

**

*Рис 3.1.5 Отчеты*

На рисунке 3.1.5 представлено окно «Отчеты». На котором показаны отчеты от сотрудников.

## **3.2 Программирование информационной системы**

## **3.2.1 Разработка модуля «Авторизация»**

using System.Linq;

using System.Timers;

using System.Windows.Controls;

using System.Windows.Threading;

using System.Windows;

using System;

using ToursApp;

using System.Text;

using System.Collections.Generic;

namespace ToursApp

{

public partial class LoginWindow : Window

{

private string \_generatedCaptcha;

private readonly Random \_random = new Random();

public LoginWindow()

{

InitializeComponent();

GenerateCaptcha();

}

// Генерация случайной капчи

private void GenerateCaptcha()

{

const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&\*()\_+-=,.?";

int captchaLength = 4; // Длина капчи

StringBuilder captchaBuilder = new StringBuilder();

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

{

int index = \_random.Next(chars.Length);

captchaBuilder.Append(chars[index]);

}

\_generatedCaptcha = captchaBuilder.ToString();

CaptchaText.Text = \_generatedCaptcha;

}

// Обработчик кнопки "Обновить"

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

{

GenerateCaptcha();

CaptchaInput.Text = string.Empty;

ErrorText.Visibility = Visibility.Collapsed;

}

// Обработчик кнопки "Войти"

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

{

ErrorText.Visibility = Visibility.Collapsed;

if (string.IsNullOrWhiteSpace(LoginBox.Text))

{

ShowError("Введите логин.");

return;

}

if (string.IsNullOrWhiteSpace(PasswordBox.Password))

{

ShowError("Введите пароль.");

return;

}

if (string.IsNullOrWhiteSpace(CaptchaInput.Text))

{

ShowError("Введите капчу.");

return;

}

if (!CaptchaInput.Text.Equals(\_generatedCaptcha))

{

ShowError("Капча введена неверно.");

GenerateCaptcha(); // Обновить капчу при ошибке

CaptchaInput.Text = string.Empty;

return;

}

// Проверка логина и пароля

var users = new Dictionary<string, (string Password, string Role)>

{

{ "admin", ("123", "admin") },

{ "manager", ("345", "manager") },

{ "operator", ("567", "operator") }

};

if (users.ContainsKey(LoginBox.Text) && users[LoginBox.Text].Password == PasswordBox.Password)

{

MessageBox.Show("Успешный вход!", "Добро пожаловать", MessageBoxButton.OK, MessageBoxImage.Information);

string role = users[LoginBox.Text].Role; // Извлекаем роль пользователя

MainWindow mainWindow = new MainWindow(role); // Передаем роль в конструктор MainWindow

mainWindow.Show();

Close();

}

else

{

ShowError("Неверный логин или пароль.");

}

}

// Отображение сообщения об ошибке

private void ShowError(string message)

{

ErrorText.Text = message;

ErrorText.Visibility = Visibility.Visible;

}

//события фокуса для TextBox

private void LoginBox\_GotFocus(object sender, RoutedEventArgs e)

{

LoginBox.BorderBrush = System.Windows.Media.Brushes.Blue;

}

private void LoginBox\_LostFocus(object sender, RoutedEventArgs e)

{

LoginBox.BorderBrush = new System.Windows.Media.SolidColorBrush((System.Windows.Media.Color)System.Windows.Media.ColorConverter.ConvertFromString("#D1D1D1"));

}

}

}

## **3.2.2 Разработка модуля «Главное окно»**

using System.Windows;

using System.Windows.Controls;

namespace ToursApp

{

public partial class MainWindow : Window

{

private readonly string \_userRole;

public MainWindow(string role)

{

InitializeComponent();

\_userRole = role;

Manager.MainFrame = MainFrame;

// Устанавливаем начальную страницу

MainFrame.Navigate(new ToursPage(\_userRole));

ConfigureAccess(); // Настраиваем видимость кнопок при инициализации

}

// Метод для настройки видимости кнопок на основе роли

private void ConfigureAccess()

{

switch (\_userRole.ToLower())

{

case "admin":

// Админ имеет доступ ко всем кнопкам

ToursButton.Visibility = Visibility.Visible;

ClientButton.Visibility = Visibility.Visible;

HotelButton.Visibility = Visibility.Visible;

ReportsButton.Visibility = Visibility.Visible;

EmployeesButton.Visibility = Visibility.Visible;

LogoutButton.Visibility = Visibility.Visible;

break;

case "manager":

// Менеджер: Туры, Клиенты, Отели, Отчеты, Выход

ToursButton.Visibility = Visibility.Visible;

ClientButton.Visibility = Visibility.Visible;

HotelButton.Visibility = Visibility.Visible; // Исправлено: добавлен доступ к Отели

ReportsButton.Visibility = Visibility.Visible;

EmployeesButton.Visibility = Visibility.Collapsed; // Нет доступа

LogoutButton.Visibility = Visibility.Visible;

break;

case "operator":

// Туроператор: Туры, Клиенты, Отели, Отчеты, Выход

ToursButton.Visibility = Visibility.Visible;

ClientButton.Visibility = Visibility.Visible;

HotelButton.Visibility = Visibility.Visible;

ReportsButton.Visibility = Visibility.Visible;

EmployeesButton.Visibility = Visibility.Collapsed; // Нет доступа

LogoutButton.Visibility = Visibility.Visible;

break;

default:

MessageBox.Show("Неизвестная роль пользователя.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

Application.Current.Shutdown();

break;

}

}

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

{

MainFrame.Navigate(new ToursPage(\_userRole));

}

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

{

MainFrame.Navigate(new ReportsPage());

}

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

{

MainFrame.Navigate(new ClientsPage());

}

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

{

MainFrame.Navigate(new HotelsPage(\_userRole));

}

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

{

if (\_userRole == "admin")

{

MainFrame.Navigate(new EmployeePage());

}

else

{

MessageBox.Show("Доступ запрещён для вашей роли!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

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

{

Application.Current.Shutdown();

}

}

}

## **3.2.3 Разработка модуля «Страниц»**

using System;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

namespace ToursApp

{

public partial class ClientsPage : Page

{

public ClientsPage()

{

InitializeComponent();

LoadClients();

}

private void LoadClients()

{

try

{

// Проверяем, что DGridClients инициализирован

if (DGridClients == null)

{

MessageBox.Show("DataGrid DGridClients не инициализирован!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

using (var context = new TourAgentryEntities())

{

// Проверяем, что контекст создан

if (context == null)

{

MessageBox.Show("Контекст базы данных не инициализирован!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Загружаем клиентов

var clients = context.Client.ToList();

if (clients == null || !clients.Any())

{

MessageBox.Show("Список клиентов пуст или не удалось загрузить данные!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

DGridClients.ItemsSource = clients;

}

}

catch (Exception ex)

{

// Подробное сообщение об ошибке

MessageBox.Show($"Ошибка при загрузке клиентов: {ex.Message}\n{ex.InnerException?.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

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

{

try

{

if (Manager.MainFrame != null)

{

// Переходим на страницу ClientFormationPage

Manager.MainFrame.Navigate(new ClientFormationPage());

}

else

{

MessageBox.Show("MainFrame не инициализирован! Убедитесь, что Manager.MainFrame настроен корректно.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при переходе на страницу ClientFormationPage: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

## **3.2.4 Разработка модуля «Добавить клиента»**

using System;

using System.Collections.Generic;

using System.IO;

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.Navigation;

using System.Windows.Shapes;

using Microsoft.Office.Interop.Word;

namespace ToursApp

{

public partial class ClientFormationPage : System.Windows.Controls.Page

{

public ClientFormationPage()

{

InitializeComponent();

}

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

{

try

{

// Проверяем, что обязательные поля для клиента заполнены

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

string.IsNullOrWhiteSpace(ClientPhoneTextBox.Text) ||

string.IsNullOrWhiteSpace(ClientEmailTextBox.Text))

{

MessageBox.Show("Заполните все поля клиента (ФИО, телефон, email)!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Проверяем, что обязательные поля для тура заполнены

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

string.IsNullOrWhiteSpace(TicketsCountTextBox.Text) ||

string.IsNullOrWhiteSpace(TourCostTextBox.Text) ||

string.IsNullOrWhiteSpace(NightsCountTextBox.Text))

{

MessageBox.Show("Заполните все поля тура (название, количество билетов, стоимость, количество ночей)!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Проверка числовых полей для тура

if (!int.TryParse(TicketsCountTextBox.Text, out int ticketsCount))

{

MessageBox.Show("Количество билетов должно быть числом!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (!decimal.TryParse(TourCostTextBox.Text, out decimal tourCost))

{

MessageBox.Show("Стоимость должна быть числом!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (!int.TryParse(NightsCountTextBox.Text, out int nightsCount))

{

MessageBox.Show("Количество ночей должно быть числом!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Разделяем ФИО на имя и фамилию

var fullName = ClientNameTextBox.Text.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

if (fullName.Length < 2)

{

MessageBox.Show("Введите имя и фамилию клиента через пробел!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

using (var context = new TourAgentryEntities())

{

// Проверяем существование тура

var tour = context.Tour.FirstOrDefault(t => t.Name == TourNameTextBox.Text);

if (tour == null)

{

MessageBox.Show("Указанный тур не найден в базе данных!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Проверяем существование сотрудника (можно заменить на текущего пользователя)

var employee = context.Employee.FirstOrDefault();

if (employee == null)

{

MessageBox.Show("Сотрудник не найден в базе данных!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Проверяем существование отеля (можно сделать выбор отеля в UI)

var hotel = context.Hotel.FirstOrDefault();

if (hotel == null)

{

MessageBox.Show("Отель не найден в базе данных!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Создаем нового клиента

var newClient = new Client

{

Name = ClientNameTextBox.Text,

Phone = ClientPhoneTextBox.Text,

Email = ClientEmailTextBox.Text

};

context.Client.Add(newClient);

context.SaveChanges(); // Сохраняем клиента, чтобы получить его Id

// Создаем новую запись бронирования

var newReservation = new Reservation

{

ReservationDate = DateTime.Now,

TourId = tour.Id, // Используем Id найденного тура

ClientId = newClient.Id,

EmployeeId = employee.Id, // Используем Id найденного сотрудника

HotelId = hotel.Id // Используем Id найденного отеля

};

context.Reservation.Add(newReservation);

context.SaveChanges(); // Сохраняем бронирование

MessageBox.Show("Клиент и бронирование успешно сохранены!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении данных: {ex.Message}\nInner Exception: {ex.InnerException?.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

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

{

Microsoft.Office.Interop.Word.Application wordApp = null;

Microsoft.Office.Interop.Word.Document doc = null;

try

{

// Указываем полный путь к шаблону

string templatePath = @"C:\Users\Эмилия\Desktop\ToursApp\ToursApp\Templates\Оформления\_тура.docx";

if (!System.IO.File.Exists(templatePath))

{

MessageBox.Show("Шаблон документа 'Оформления\_тура.docx' не найден по пути: " + templatePath, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Проверка обязательных полей

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

string.IsNullOrWhiteSpace(EmployeePositionTextBox.Text) ||

string.IsNullOrWhiteSpace(ClientNameTextBox.Text) ||

string.IsNullOrWhiteSpace(ClientPhoneTextBox.Text) ||

string.IsNullOrWhiteSpace(ClientEmailTextBox.Text) ||

string.IsNullOrWhiteSpace(TourNameTextBox.Text) ||

string.IsNullOrWhiteSpace(TicketsCountTextBox.Text) ||

string.IsNullOrWhiteSpace(TourCostTextBox.Text) ||

string.IsNullOrWhiteSpace(NightsCountTextBox.Text))

{

MessageBox.Show("Заполните все обязательные поля!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Проверка числовых полей

if (!int.TryParse(TicketsCountTextBox.Text, out \_))

{

MessageBox.Show("Количество билетов должно быть числом!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (!decimal.TryParse(TourCostTextBox.Text, out \_))

{

MessageBox.Show("Стоимость должна быть числом!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (!int.TryParse(NightsCountTextBox.Text, out \_))

{

MessageBox.Show("Количество ночей должно быть числом!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

wordApp = new Microsoft.Office.Interop.Word.Application();

wordApp.Visible = false;

doc = wordApp.Documents.Open(templatePath);

// Замена текста в документе

ReplaceTextInDocument(doc, "ФИО сотрудника: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", $"ФИО сотрудника: {EmployeeNameTextBox.Text}");

ReplaceTextInDocument(doc, "Должность: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", $"Должность: {EmployeePositionTextBox.Text}");

ReplaceTextInDocument(doc, "ФИО заказчика: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", $"ФИО заказчика: {ClientNameTextBox.Text}");

ReplaceTextInDocument(doc, "Телефон: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", $"Телефон: {ClientPhoneTextBox.Text}");

ReplaceTextInDocument(doc, "Электронная почта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", $"Электронная почта: {ClientEmailTextBox.Text}");

ReplaceTextInDocument(doc, "Название тура: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", $"Название тура: {TourNameTextBox.Text}");

ReplaceTextInDocument(doc, "Количество билетов: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", $"Количество билетов: {TicketsCountTextBox.Text}");

ReplaceTextInDocument(doc, "Стоимость (за всё): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", $"Стоимость (за всё): {TourCostTextBox.Text}");

ReplaceTextInDocument(doc, "Количество ночей: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", $"Количество ночей: {NightsCountTextBox.Text}");

ReplaceTextInDocument(doc, "Подпись сотрудника: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", $"Подпись сотрудника: {EmployeeNameTextBox.Text}");

ReplaceTextInDocument(doc, "Подпись заказчика: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", $"Подпись заказчика: {ClientNameTextBox.Text}");

ReplaceTextInDocument(doc, "Дата оформления: \_\_\_ «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г.",

$"Дата оформления: {DateTime.Now:dd «MMMM» yyyy} г.");

doc.Save();

doc.Close();

MessageBox.Show("Отчёт успешно сформирован! Изменения сохранены в: " + templatePath, "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

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

}

finally

{

if (doc != null)

{

try { doc.Close(false); System.Runtime.InteropServices.Marshal.ReleaseComObject(doc); }

catch { }

}

if (wordApp != null)

{

try { System.Runtime.InteropServices.Marshal.ReleaseComObject(wordApp); }

catch { }

}

doc = null;

wordApp = null;

GC.Collect();

GC.WaitForPendingFinalizers();

}

}

private void ReplaceTextInDocument(Microsoft.Office.Interop.Word.Document doc, string oldText, string newText)

{

Microsoft.Office.Interop.Word.Range range = doc.Content;

range.Find.ClearFormatting();

range.Find.Execute(FindText: oldText, ReplaceWith: newText, Replace: Microsoft.Office.Interop.Word.WdReplace.wdReplaceAll);

}

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

{

if (NavigationService != null && NavigationService.CanGoBack)

{

NavigationService.GoBack();

}

}

}

}

## **3.2.5 Разработка модуля «Отчет»**

using System;

using System.Windows;

using System.Windows.Controls;

namespace ToursApp

{

public partial class ReportFormationPage : Page

{

public ReportFormationPage()

{

InitializeComponent();

TbCreationDate.Text = DateTime.Now.ToString("dd.MM.yyyy");

}

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

{

try

{

// Проверка обязательных полей

if (string.IsNullOrWhiteSpace(TbEmployeeName.Text))

{

MessageBox.Show("Введите имя сотрудника!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (CbPosition.SelectedItem == null)

{

MessageBox.Show("Выберите должность!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (CbMonth.SelectedItem == null)

{

MessageBox.Show("Выберите месяц!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (!int.TryParse(TbToursBooked.Text, out int toursBooked))

{

MessageBox.Show("Количество забронированных туров должно быть числом!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (!int.TryParse(TbClientsServed.Text, out int clientsServed))

{

MessageBox.Show("Количество клиентов должно быть числом!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Получение выбранной должности и месяца

string position = ((ComboBoxItem)CbPosition.SelectedItem).Content.ToString();

string selectedMonth = ((ComboBoxItem)CbMonth.SelectedItem).Content.ToString();

// Преобразование месяца в DateTime

int monthNumber = Array.IndexOf(new[]

{

"Январь", "Февраль", "Март", "Апрель", "Май", "Июнь",

"Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"

}, selectedMonth) + 1;

DateTime? monthDate = monthNumber > 0

? new DateTime(DateTime.Now.Year, monthNumber, 1)

: (DateTime?)null;

using (var context = new TourAgentryEntities())

{

var report = new Report

{

EmployeeName = TbEmployeeName.Text,

Position = position,

Month = monthDate,

ToursBooked = toursBooked,

ClientsServed = clientsServed,

CreationDate = DateTime.TryParse(TbCreationDate.Text, out DateTime creationDate)

? creationDate

: (DateTime?)null,

Name = $"Report\_{DateTime.Now:yyyyMMdd\_HHmmss}",

Download = null

};

context.Report.Add(report);

context.SaveChanges();

}

MessageBox.Show("Отчет успешно сохранен! Обновите список отчетов вручную, если необходимо.", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

NavigationService?.GoBack();

}

catch (Exception ex)

{

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

}

}

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

{

NavigationService?.GoBack();

}

}

}

# **ЗАКЛЮЧЕНИЕ**

В рамках выполнения курсового проекта была разработана автоматизированная информационная система "Турагентство" для управления системой, ведения учётов клиентов, туров и отелей. При создании системы использовались современные технологии: платформа .NET, язык программирования C#, WPF для интерфейса, SQL Server, в качестве СУБД, и Entity Framework для работы с данными.

Система включает модули авторизации с проверкой логина, пароля и капчи, главное меню с адаптацией под роли пользователей (администратор, менеджер, туроператор), функционал для работы с документами и запросами, а также формирование отчетов с возможностью экспорта в Word. База данных построена по реляционной модели и содержит таблицы для пользователей, отелей, клиентов, туров,отчетов, что отражено в ER-диаграмме. Система полностью соответствует требованиям технического задания, включая разграничение прав доступа, контроль вводимых данных и защиту от несанкционированного доступа.

Внедрение системы позволит сократить время на оформление тура для клиента с 8-10 минут до 1-2 минут, а формирование отчетов с 6-12 часов до 3-4 минут, что значительно повысит эффективность работу турагентства. Для дальнейшего развития системы рекомендуется интеграция с MS Office и расширение функционала, например, добавление мобильного или веб-интерфейса. Разработанная информационная система успешно реализована, соответствует поставленным задачам и готова к внедрению, обеспечивая автоматизацию ключевых процессов архивного учета.

Все этапы разработки - от выбора инструментария до программирования модулей - выполнены в полном объеме, что подтверждает достижение целей курсового проекта.

<https://github.com/emilifres/KursovaTourAgentry>

# **СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Введение в Entity Framework Core [Электронный ресурс] — режим доступа:

<https://metanit.com/sharp/efcore/1.1.php>

1. Введение в MS SQL Server и T-SQL [Электронный ресурс] — режим доступа:

<https://metanit.com/sql/sqlserver/1.1.php>

1. Документация по C# [Электронный ресурс] — режим доступа:

<https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/overview>

1. Документация по Microsoft SQL [Электронный ресурс] — режим доступа:

<https://learn.microsoft.com/ru-ru/ssms/sql-server-management-studio-ssms>

1. Полное руководство по языку программирования С# 13 и платформе .NET 9. [Электронный ресурс] — режим доступа:

<https://metanit.com/sharp/tutorial/1.1.php>

1. Руководство по WPF [Электронный ресурс] — режим доступа:

<https://metanit.com/sharp/wpf/1.php>

<https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/>

1. Что такое ER-диаграмма и как ее создать? [Электронный ресурс] — режим доступа: <https://www.lucidchart.com/pages/ru/erd-diagram>