**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**Московский государственный университет технологии и управления имени К.Г. Разумовского (Первый казачий университет)**

**Университетский колледж информационных технологий**

Специальность: 09.02.03 «Программирование в компьютерных системах»

**КУРСОВОЙ ПРОЕКТ**

По модулю ПМ.02 «Разработка и администрирование баз данных»

МДК.02.02 «Технология разработки и защиты баз данных»

на тему «Система резервирования мест в гостиницах»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**УКИТ 09.02.03.2016\_303.015ПЗ**

Группа \_\_\_*П-303\_\_\_*

Студент \_\_\_\_\_\_\_\_\_\_\_ Ильин П.М.

(*личная подпись*)

Руководители проекта \_\_\_\_\_\_\_\_\_\_\_ Ларионова Е.А.

(*личная подпись*)

**МОСКВА 2016**

Содержание

[**ВВЕДЕНИЕ** 3](#_Toc478087686)

[**1.** **ОБЩАЯ ЧАСТЬ** 4](#_Toc478087687)

[**1.1. Анализ предметной области.** 4](#_Toc478087688)

[**1.2. Жизненный цикл базы данных.** 5](#_Toc478087689)

[**1.3. Защита базы данных.** 7](#_Toc478087690)

[**1.4. Выбор и характеристика СУБД.** 9](#_Toc478087691)

[**1.5. Выбор и характеристика среды разработки приложения.** 10](#_Toc478087692)

[2. **СПЕЦИАЛЬНАЯ ЧАСТЬ** 11](#_Toc478087693)

[**2.1. Постановка задачи.** 11](#_Toc478087694)

[**2.2. Архитектура информационной системы.** 11](#_Toc478087695)

[**2.3. Логическая модель БД.** 12](#_Toc478087696)

[**2.4. Нормализация таблиц.** 13](#_Toc478087698)

[**2.5. Описание таблиц.** 15](#_Toc478087699)

[**2.6. Другие объекты базы данных** 16](#_Toc478087701)

[**2.7. Разработка приложения.** 18](#_Toc478087702)

[**2.7.2. Используемая технология доступа к данным.** 18](#_Toc478087703)

[**2.7.3 Описание процесса отладки приложения.** 18](#_Toc478087704)

[**2.7.4 Защита информационной системы.** 18](#_Toc478087705)

[**2.8. Инструкция пользователю.** 18](#_Toc478087706)

[**2.8.1 Общие сведения об информационной системе.** 18](#_Toc478087707)

[**2.8.2 Требования к техническим средствам.** 18](#_Toc478087708)

[**2.8.3 Требования к программным средствам.** 19](#_Toc478087709)

[**2.8.4 Формы ввода.** 19](#_Toc478087710)

[**ЗАКЛЮЧЕНИЕ** 22](#_Toc478087711)

[**Приложение 1.** 23](#_Toc478087712)

[**Приложение 2.** 23](#_Toc478087713)

[**СПИСОК ЛИТЕРАТУРЫ** 67](#_Toc478087714)

**ВВЕДЕНИЕ**

Курсовой проект посвящен разработке прикладной программы, которая позволит резервировать места в гостиницах.

Приложение состоит из нескольких частей. Части, отвечающей за заполнение базы данных информацией о гостиницах и комнатах в них, и части, отвечающей за расселение постояльцев, посредством выбора подходящей гостиницы, комнаты и заполнения анкеты.

Целью курсового проекта является приобретение навыков разработки локальных и удалённых баз данных и приложений для работы с ними с использованием объектно-ориентированного и визуального программирования.

Задачи курсового проекта:

* Анализ предметной области;
* Проектирование базы данных средствами ErwinModeler и MSSQLServer 2008 R2
* Разработка приложения средствами MS Visual Studio 2013 (C#)
* Защита базы данных и приложения должна осуществляется логином и паролем администратора или пользователя.

Возможности приложения:

* Добавление, изменение данных о гостиницах;
* Добавление, изменение данных о комнатах в гостиницах;
* Резервирование мест комнат в гостиницах.

1. **ОБЩАЯ ЧАСТЬ**

**1.1. Анализ предметной области.**

Приложение позволяет пользователю быстро и просто предоставить человеку информацию о свободных местах в гостиницах и так же легко зарезервировать для него комнату на определённый срок.

Приложение состоит из нескольких частей. Части, отвечающей за заполнение базы данных информацией о гостиницах и комнатах в них, и части, отвечающей за расселение постояльцев, посредством выбора подходящей гостиницы, комнаты и заполнения анкеты.

Выделим базовые сущности этой предметной области:

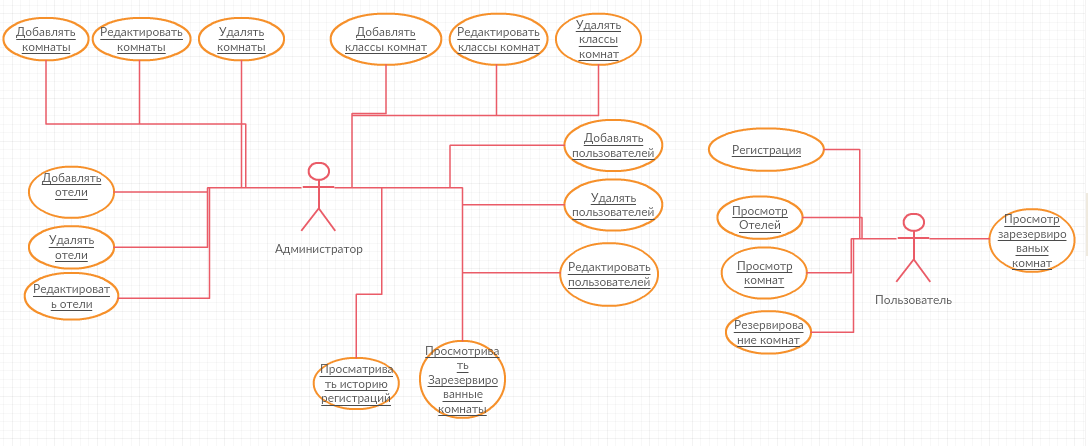
**Гостиница** – Атрибуты – Номер, название, адрес, количество звёзд.

**Комната** – Атрибуты – Номер комнаты, номер гостиницы, тип комнаты.

**Тип комнаты** – Атрибуты – Код типа, тип, стоимость

**Пользователь** – Атрибуты – Фамилия, имя, отчество, логин, пароль, код пользователя, номер телефона.

**Резервированные** – Атрибуты – Код пользователя, номер гостиницы, номер комнаты, дата заселения, дата выселения.

Далее следует ER - диаграмма приложения "Система резервирования мест в гостиницах"

**1.2. Жизненный цикл базы данных.**

Жизненный цикл БД- это процесс проектирования, реализации и управления БД. Жизненный цикл БД состоит из 7-ми этапов:

1) Планирование разработки

2) Проверка осуществимости

3) Определение требований

4) Концептуальное проектирование

5) Логическое проектирование

6) Физическое проектирование

7) Оценка работы и поддержка базы данных

1. Предварительное планирование базы данных – важный этап в процессе перехода от разрозненных данных к интегрированным. На этом этапе собирается информация об используемых и находящихся в процессе разработки прикладных программ и файлах, связанных с ними. Она помогает установить связи между текущими приложениями и то, как используется их информация. Кроме того, позволяет определить будущее требования в БД. Информация документируется в виде обобщенной концептуальной модели данных.

2. Проверка осуществимости предполагает подготовку отчетов по трем вопросам:

1) Есть ли технология – необходимое оборудование и программное обеспечение - для реализации запланированной БД (технологическая осуществимость)

2) Имеется ли персонал, средства и эксперты для успешного осуществления плана создания БД (операционная осуществимость)

3) Окупится ли запланированная БД (экономическая эффективность)

3. Определение требований. На этом этапе определяются:

* Цели базы данных
* Информационные потребности различных структурных подразделений и их руководителей
* Требование к оборудованию
* Требование к программному обеспечению

4. Концептуальное проектирование. На этом этапе создаются подробные модели пользовательских преставлений данных предметной области. Затем они интегрируются в концептуальную модель, которая фиксирует все элементы корпоративных данных, подлежащих загрузке в базу данных. Эту модель еще называют концептуальной схемой базы данных.

5. Логическое проектирование. На этом этапе осуществляется выбор типа модели данных. Концептуальная модель отображается в логическую модель, основанную на структурах, характерных для выбора модели.

6. Физическое проектирование. На этом этапе логическая модель расширяется характеристиками, необходимыми для определения способов физического хранения базы данных, типа устройств для хранения, методов доступа к данным базы, требуемого объема памяти, правил сопровождения базы данных и др.

7. Оценка и поддержка базы данных. Оценка включает опрос пользователей на предмет выяснения, какие их информационные потребности остались неучтенными. При необходимости в спроектированную БД вносятся изменения. Пользователи обучаются работе с БД. По мере расширения и изменения потребностей бизнеса поддержка БД обеспечивается путем внесения изменений, добавления новых данных, разработки новых прикладных программ, работающих с БД.

Сроки разработки и сдачи курсового проекта 29.03.16- 23.05.16.

Начало работы 14.03.16-30.05.16.

**1.3. Защита базы данных.**

Необходимость защиты баз данных заключается в том, чтобы не произошла утечка данных.

Методы защиты БД:

* Защита паролем;
* Шифрование данных и программ;
* Разграничение прав доступа к объектам БД;
* Защита полей и записей таблиц БД.
* Защита от SQL инъекций

Защита паролем представляет собой простой и эффективный способ защиты БД от несанкционированного доступа. Пароли устанавливаются пользователями или администраторами БД. Учет и хранение паролей выполняется самой СУБД. Парольная защита БД считается слабым средством, особенно если пароль не шифруется. Основной ее недостаток состоит в том, что все пользователи, использующие одинаковый пароль, с точки зрения вычислительной системы неразличимы. Неудобство парольной защиты для пользователя состоит в том, что пароль надо запоминать или записать. При небрежном отношении к записям пароль может стать достоянием других.

Более мощным средством защиты данных от просмотра является их шифрование. Шифрование – это преобразование читаемого текста в нечитаемый текст, при помощи некоторого алгоритма; применяется для защиты уязвимых данных.

Процесс дешифрования восстанавливает данные в исходное состояние.

В целях контроля использования основных ресурсов СУБД во многих системах имеются средства установления прав доступа к объектам БД. Права доступа определяют возможные действия над объектами. Владелец объекта (пользователь, создавший объект), а также администратор БД имеют все права. Остальные пользователи к разным объектам могут иметь различные уровни доступа. Разрешение на доступ к конкретным объектам базы данных сохраняется в файле рабочей группы.

Файл рабочей группы содержит данные о пользователях группы и считывается во время запуска. Файл содержит следующую информацию: имена учетных записей пользователей, пароли пользователей, имена групп, в которые входят пользователи.

По отношению к таблицам могут предусматриваться следующие права доступа:

* просмотр (чтение) данных;
* изменение (редактирование) данных;
* добавление данных;

К данным, имеющимся в таблице, могут применяться меры защиты по отношению к отдельным полям и отдельным записям.

Защита данных в полях таблиц предусматривает следующие уровни прав доступа:

* полный запрет доступа;
* только чтение;
* разрешение всех операций (просмотр, ввод новых значений, удаление и изменение).

Для предотвращения SQL инъекций следует соблюдать два простых правила:

1. Не помещать в БД данные без обработки.

Это можно сделать либо с помощью подготовленных выражений, либо обрабатывая параметры вручную.

Если запрос составляется вручную, то

- все числовые параметры должны быть приведены к нужному типу

2. Не помещать в запрос управляющие структуры и идентификаторы, введенные пользователем.

В данном курсовом проекта защита информационной системы сделана с помощью логина и пароля для администратора и осуществление входа пользователя с помощью ввода соответственного пароля.

**1.4. Выбор и характеристика СУБД.**

В качестве СУБД для проекта выступает Microsoft SQL Server 2008 R2. Выбор обусловлен наличием опыта работы в данной СУБД и кодирования на языке SQL, а также удобством работы с ней в связке в выбранной IDE. Кроме того, СУБД существует в бесплатной редакции Express, а по студенческой подписке в сервисе Microsoft Azure можно бесплатно развернуть облачное хранилище на основе Microsoft SQL Server (Azure SQL). Из минусов СУБД следует отметить её низкую производительность на слабых машинах, а также отсутствие возможности работы на ОС, отличной от Windows.

**1.5. Выбор и характеристика среды разработки приложения.**

MS Visual studio 2013 – это среда для разработки программного обеспечения на языках: Visual Basic, C, C++, C#. Интегрированная среда разработки предоставляет возможность разработки приложений под платформу windows.

* Реализован основной набор элементов управления
* Редактор форм и инспектор объектов
* Интерфейс отладки
* Мощный редактор кода, включающий систему подсказок, гипертекстовую навигацию по исходным текстам, автозавершение кода и рефакторинг
* Имеет собственный формат управления пакетами
* Поддерживаемые для компиляции Microsoft Windows (Win32, Win64),

# 2. СПЕЦИАЛЬНАЯ ЧАСТЬ

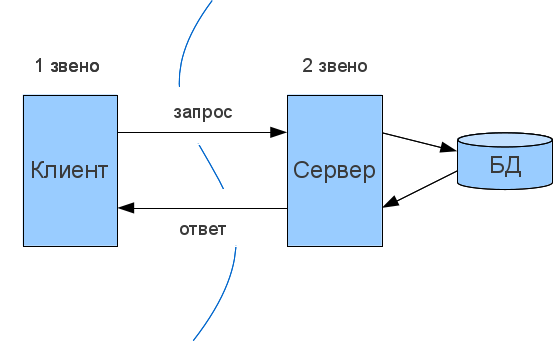
## **2.1. Постановка задачи.**

Спроектировать средствами MS SQL SERVER 2008 R2 базу данных и создать приложение, позволяющее добавлять гостиницы и комнаты в базу, резервировать комнаты в гостиницах, заполнять анкету постояльца, вести журнал по заселению, выселению постояльцев.

Предусмотреть возможность добавления, удаления, редактирования записей. Обеспечить защиту базы данных и приложения.

**2.2. Архитектура информационной системы.**

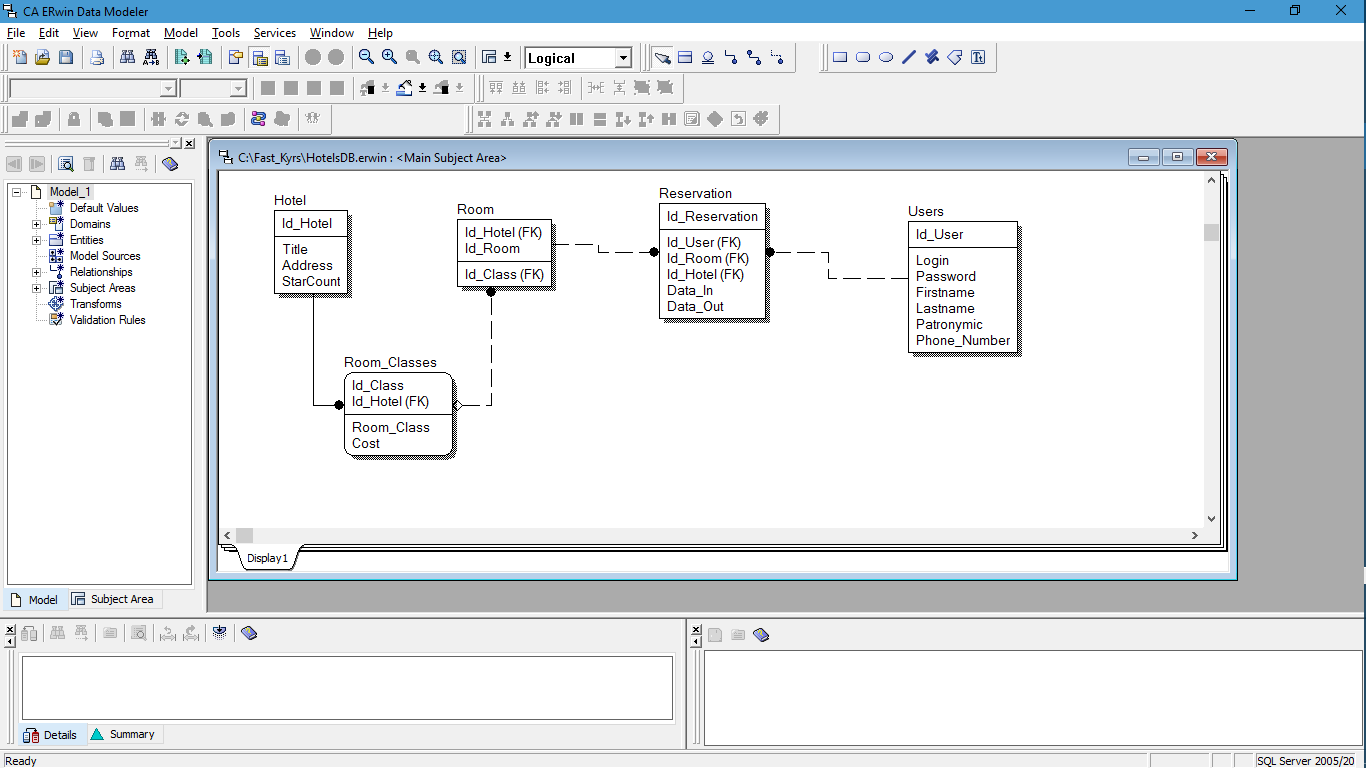
Архитектура «клиент-сервер» - вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиком услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.



|  |  |
| --- | --- |
| **Преимущества** | **Недостатки** |
| Одновременный доступ | Высокая стоимость |
| Высокая производительность | Высокая нагрузка на сервер |
| Низкий трафик |  |
| Высокая защита |  |
| Высокая роль сервера |  |

**2.3. Логическая модель БД.**

Для построения логической модели БД используется средства автоматизации проектирования ErwinModeler.



**2.4. Нормализация таблиц.**

Отношение находится в 1 нормальной форме, если все атрибуты просты, далее неделимы.

Отношение находится во 2 нормальной форме, если оно соответствует требованиям 1 нормальной формы, и каждый не ключевой атрибут функционально зависит от первичного ключа.

**Таблица 1.** Гостиница

Номер гостиницы \*

Название

Адрес

Количество звёзд

**Таблица 2.** Комната

Номер гостиницы \*

Номер комнаты \*

Тип комнаты

**Таблица 3.** Тип комнаты

Код типа \*

Тип

Стоимость

**Таблица 4.** Пользователь

Код пользователя \*

Логин

Пароль

Фамилия

Имя

Отчество

Номер телефона

**Таблица 5.** Резервирование

Номер записи \*

Код пользователя

Номер отеля

Номер комнаты

Дата заселения

Дата выселения

Отношение находится в 3 нормальной форме, если оно соответствует требованиям 2 нормальной формы, и каждый не ключевой атрибут не транзитивно зависит от первичного ключа.

**2.5. Описание таблиц.**

Hotel

|  |  |  |
| --- | --- | --- |
| Id\_Hotel | Varchar(20) | Номер отеля |
| Title | Varchar(20) | Название гостиницы |
| Address | Varchar(20) | Адрес гостиницы |
| StarCount | int | Количество звёзд |

Room\_Classes

|  |  |  |
| --- | --- | --- |
| Id\_Class | Int | Код типа комнаты |
| ID\_Hotel | Varchar(20) | Номер отеля |
| Room\_Class | Varchar(20) | Описание типа |
| Cost | int | Стоимость |

Room

|  |  |  |
| --- | --- | --- |
| Id\_Room | Varchar(20) | Номер комнаты |
| Id\_Hotel | Varchar(20) | Номер отеля |
| Id\_Class | Int | Код типа комнаты |

Users\_

|  |  |  |
| --- | --- | --- |
| Id\_User | Int | Код пользователя |
| User\_Login | Varchar(20) | Логин |
| User\_Password | Varchar(20) | Пароль |
| Lastname | Varchar(20) | Фамилия |
| Firstname | Varchar(20) | Имя |
| Patronymic | Varchar(20) | Отчество |
| Phone\_Number | Varchar(20) | Номер телнфона |

**Reservation**

|  |  |  |
| --- | --- | --- |
| Id\_Reservation | int | Код записи |
| Id\_Room | Varchar(20) | Номер комнаты |
| Id\_User | Int | Код пользователя |
| Id\_Hotel | Varchar(20) | Номер гостиницы |
| Data\_In | Datetime | Дата заселения |
| Data\_Out | Datetime | Дата выселения |

**2.6. Другие объекты базы данных**

**Хранимы процедуры**

|  |  |
| --- | --- |
| proc\_Add\_Classes | Добавление типа комнаты |
| proc\_Add\_Hotel | Добавление отеля |
| proc\_Add\_Reservation | Резервирование |
| proc\_Add\_Room | Добавление комнаты |
| proc\_Add\_User | Создание пользователя |
| proc\_Clear\_Old\_Reservations | Удаление резервирований с истёкшим сроком |
| proc\_Delete\_Hotel | Удаление отеля |
| proc\_Delete\_Room | Удаление комнаты |
| proc\_Delete\_Room\_Class | Удаление типа клмнаты |
| proc\_Delete\_User | Удаление пользователя |
| proc\_DeleteMyResrv | Удаление Резервирования |
| proc\_Edit\_Class | Редактирование типа комнаты |
| proc\_Login | Аутентификация |
| proc\_Select\_Free | Просмотр свободных комнат |
| proc\_Select\_Hotels | Просмотр отелей |
| proc\_Select\_My\_Reservation | Просмотр зарезервированных пользователем комнат |
| re\_backup | Резервное копирование базы данных |

**Представления**

|  |  |
| --- | --- |
| Reservations | Кем заняты номера |

**2.7. Разработка приложения.**

**2.7.2. Используемая технология доступа к данным.**

Для доступа к базе данных используется технология ADO.NET.Подключение происходит путем создания SqlConnection из пространства имен System.Data.SqlClient и создания SqlCommand для описания команды в SqlServer.

**2.7.3 Описание процесса отладки приложения.**

* Синтаксические ошибки.

**2.7.4 Защита информационной системы.**

Защита информационной системы сделана с помощью логина и пароля для администратора и осуществление входа пользователя с помощью ввода соответствующего пароля.

**2.8. Инструкция пользователю.**

**2.8.1 Общие сведения об информационной системе.**

Информационная система предназначена для резервирования комнат. В ней ведется хранение и учет данных о гостиницах, так же есть данные о зарегистрированных пользователях.

**2.8.2 Требования к техническим средствам.**

В состав технических средств должен входить компьютер, включающий:

– процессоры Intel, AMD;

– клавиатуру;

– видеокарту;

– монитор;

– жесткий или SSD-диск.

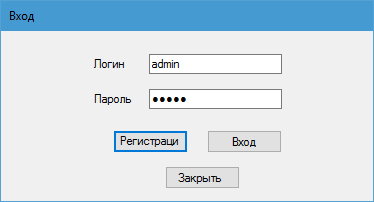
**2.8.3 Требования к программным средствам.**

Для работы необходимы такие программы:

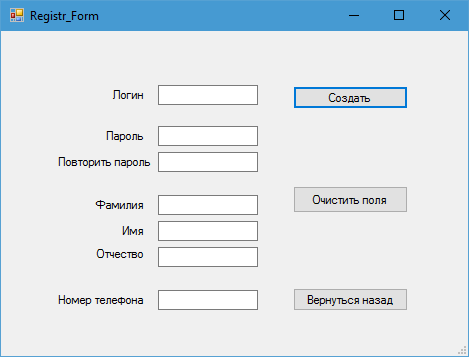
* MS Server 2008 версии R2
* .Net Framework 4.5.0
* MS Visual Studio 2013
* Операционная система: Windows

**2.8.4 Формы ввода.**

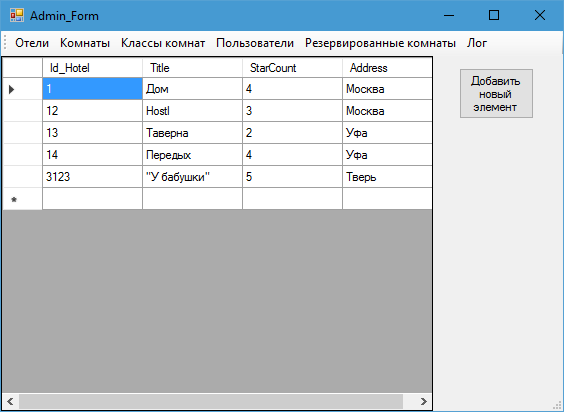
1. При запуске программы появиться окно для ввода логина и пароля, а так же регистрации.



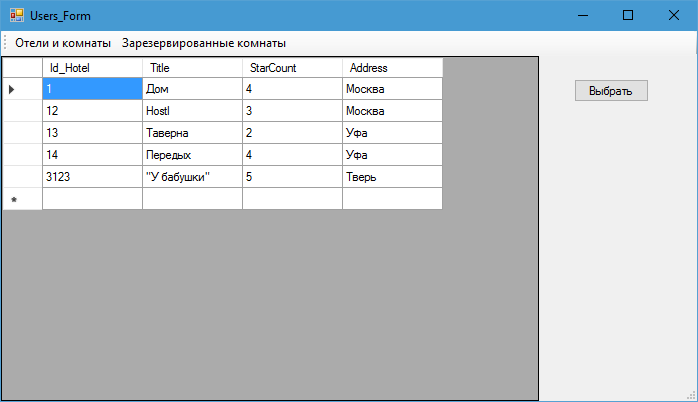
1. Окно регистрации



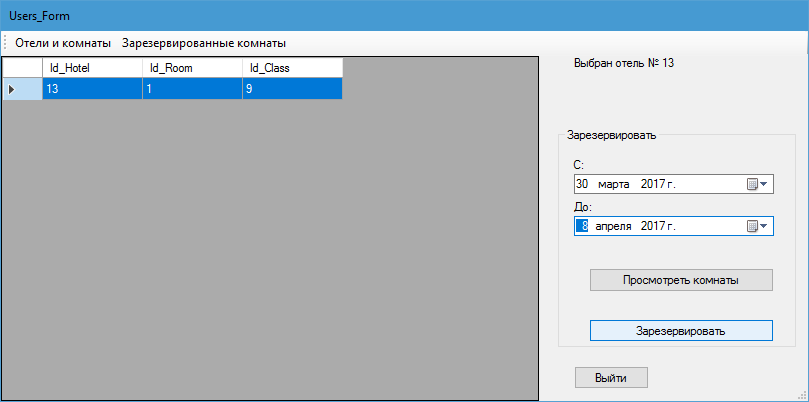
1. Окно Администратора



1. Окно пользователя



1. Резервирование



**ЗАКЛЮЧЕНИЕ**

Курсовой проект посвящен разработке программы для резервирования мест в гостиницах.

В процессе выполнения курсового проекта:

1. Изучен анализ предметной области
2. Спроектирована база данных по экзаменационным билетам
3. Разработано приложение средствами MS Visual Studio 2013 (C#)
4. Осуществлена защита БД и приложения с помощью логина и пароля для администратора и пользователя.

Возможности приложения:

* Работа с данными о гостиницах;
* Работа с данными о комнатах;
* Регистрация пользователей;
* Резервирование комнат.

Дальнейшее развитие темы курсового проекта предполагает:

1. Возможно создание сайта;
2. Приложение для пользователей смартфонов;
3. Улучшение интерфейса;
4. Расширение функционала;
5. Добавление подсказок.

**Приложение 1.**

SQL скрипты на создание и заполнение базы данных, на создание Гостиницы, пользователей, комнат, типов комнат, резервирования.

# Приложение 2.

CREATE TABLE [dbo].[Hotel](

[Id\_Hotel] [varchar](20) NOT NULL,

[Title] [varchar](20) NULL,

[Address] [varchar](20) NULL,

[StarCount] [int] NULL,

CONSTRAINT [XPKHotel] PRIMARY KEY CLUSTERED

(

[Id\_Hotel] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

CREATE TABLE [dbo].[Users\_](

[Id\_User] [int] IDENTITY(1,1) NOT NULL,

[User\_Login] [varchar](20) NULL,

[User\_Password] [varchar](20) NULL,

[Firstname] [varchar](20) NULL,

[Lastname] [varchar](20) NULL,

[Patronymic] [varchar](20) NULL,

[Phone\_Number] [varchar](20) NULL,

CONSTRAINT [XPKUsers] PRIMARY KEY CLUSTERED

(

[Id\_User] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

CREATE TABLE [dbo].[Room\_Classes](

[Id\_Class] [int] IDENTITY(1,1) NOT NULL,

[Room\_Class] [varchar](20) NULL,

[Id\_Hotel] [varchar](20) NOT NULL,

[Cost] [int] NULL,

CONSTRAINT [XPKRoom\_Classes] PRIMARY KEY CLUSTERED

(

[Id\_Class] ASC,

[Id\_Hotel] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

CREATE TABLE [dbo].[Room](

[Id\_Room] [varchar](20) NOT NULL,

[Id\_Hotel] [varchar](20) NOT NULL,

[Id\_Class] [int] NULL,

CONSTRAINT [XPKRoom] PRIMARY KEY CLUSTERED

(

[Id\_Hotel] ASC,

[Id\_Room] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

CREATE TABLE [dbo].[Reservation](

[Id\_Reservation] [int] IDENTITY(1,1) NOT NULL,

[Id\_Room] [varchar](20) NOT NULL,

[Id\_User] [int] NOT NULL,

[Id\_Hotel] [varchar](20) NOT NULL,

[Data\_In] [datetime] NULL,

[Data\_Out] [datetime] NULL,

CONSTRAINT [XPKReservation] PRIMARY KEY CLUSTERED

(

[Id\_Reservation] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: Trigger [dbo].[trigg\_reglog] Script Date: 03/23/2017 23:10:06 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER TRIGGER [dbo].[trigg\_reglog]

ON [dbo].[Users\_] AFTER INSERT

AS

IF EXISTS (SELECT \* FROM inserted)

BEGIN

DECLARE @login varchar(20)

DECLARE @l varchar(20)

DECLARE @f varchar(20)

DECLARE @p varchar(20)

SELECT @login = (SELECT inserted.User\_Login FROM inserted)

SELECT @l = (SELECT inserted.Lastname FROM inserted)

SELECT @f = (SELECT inserted.Firstname FROM inserted)

SELECT @p = (SELECT inserted.Patronymic FROM inserted)

INSERT INTO Reg\_Log

VALUES (@login, @l, @f, @p, GETDATE())

END

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: View [dbo].[Reservations] Script Date: 03/23/2017 23:10:38 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[Reservations]

AS

SELECT dbo.Reservation.Id\_Reservation, dbo.Reservation.Id\_Room, dbo.Users\_.Firstname, dbo.Users\_.Lastname, dbo.Users\_.Patronymic, dbo.Users\_.Phone\_Number, dbo.Reservation.Data\_In,

dbo.Reservation.Data\_Out

FROM dbo.Reservation INNER JOIN

dbo.Users\_ ON dbo.Reservation.Id\_User = dbo.Users\_.Id\_User

GO

EXEC sys.sp\_addextendedproperty @name=N'MS\_DiagramPane1', @value=N'[0E232FF0-B466-11cf-A24F-00AA00A3EFFF, 1.00]

Begin DesignProperties =

Begin PaneConfigurations =

Begin PaneConfiguration = 0

NumPanes = 4

Configuration = "(H (1[40] 4[20] 2[20] 3) )"

End

Begin PaneConfiguration = 1

NumPanes = 3

Configuration = "(H (1 [50] 4 [25] 3))"

End

Begin PaneConfiguration = 2

NumPanes = 3

Configuration = "(H (1 [50] 2 [25] 3))"

End

Begin PaneConfiguration = 3

NumPanes = 3

Configuration = "(H (4 [30] 2 [40] 3))"

End

Begin PaneConfiguration = 4

NumPanes = 2

Configuration = "(H (1 [56] 3))"

End

Begin PaneConfiguration = 5

NumPanes = 2

Configuration = "(H (2 [66] 3))"

End

Begin PaneConfiguration = 6

NumPanes = 2

Configuration = "(H (4 [50] 3))"

End

Begin PaneConfiguration = 7

NumPanes = 1

Configuration = "(V (3))"

End

Begin PaneConfiguration = 8

NumPanes = 3

Configuration = "(H (1[56] 4[18] 2) )"

End

Begin PaneConfiguration = 9

NumPanes = 2

Configuration = "(H (1 [75] 4))"

End

Begin PaneConfiguration = 10

NumPanes = 2

Configuration = "(H (1[66] 2) )"

End

Begin PaneConfiguration = 11

NumPanes = 2

Configuration = "(H (4 [60] 2))"

End

Begin PaneConfiguration = 12

NumPanes = 1

Configuration = "(H (1) )"

End

Begin PaneConfiguration = 13

NumPanes = 1

Configuration = "(V (4))"

End

Begin PaneConfiguration = 14

NumPanes = 1

Configuration = "(V (2))"

End

ActivePaneConfig = 0

End

Begin DiagramPane =

Begin Origin =

Top = 0

Left = 0

End

Begin Tables =

Begin Table = "Reservation"

Begin Extent =

Top = 6

Left = 38

Bottom = 126

Right = 207

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Users\_"

Begin Extent =

Top = 6

Left = 245

Bottom = 126

Right = 414

End

DisplayFlags = 280

TopColumn = 3

End

End

End

Begin SQLPane =

End

Begin DataPane =

Begin ParameterDefaults = ""

End

Begin ColumnWidths = 9

Width = 284

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

End

End

Begin CriteriaPane =

Begin ColumnWidths = 11

Column = 1440

Alias = 900

Table = 1170

Output = 720

Append = 1400

NewValue = 1170

SortType = 1350

SortOrder = 1410

GroupBy = 1350

Filter = 1350

Or = 1350

Or = 1350

Or = 1350

End

End

End

' , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'Reservations'

GO

EXEC sys.sp\_addextendedproperty @name=N'MS\_DiagramPaneCount', @value=1 , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'Reservations'

GO

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Add\_Classes] Script Date: 03/23/2017 23:12:39 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Add\_Classes]

@class varchar(20),

@id\_hotel varchar(20),

@cost int

AS

INSERT INTO Room\_Classes (Id\_Hotel, Room\_Class, Cost)

VALUES (@id\_hotel, @class, @cost)

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Add\_Hotel] Script Date: 03/23/2017 23:12:48 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Add\_Hotel]

@id\_hotel varchar(20),

@title varchar(20),

@address varchar(20),

@starcount int

AS

INSERT INTO Hotel (Id\_Hotel, Title, [Address], StarCount)

VALUES (@id\_hotel, @title, @address,@starcount)

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Add\_Room] Script Date: 03/23/2017 23:13:09 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Add\_Room]

@id\_hotel varchar(20),

@id\_room varchar(20),

@id\_class int

AS

INSERT INTO Room (Id\_Hotel, Id\_Room, Id\_Class)

VALUES (@id\_hotel, @id\_room, @id\_class)

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Add\_Reservation] Script Date: 03/23/2017 23:13:39 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Add\_Reservation]

@id\_hotel varchar(20),

@id\_room varchar(20),

@id\_user int,

@data\_in date,

@day\_count int

AS

INSERT INTO Reservation (Id\_Hotel, Id\_Room, Id\_User, Data\_In, Data\_Out)

VALUES (@id\_hotel, @id\_room, @id\_user, @data\_in, DATEADD(dd, @day\_count, @data\_in))

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Add\_User] Script Date: 03/23/2017 23:13:52 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Add\_User]

@login varchar(20),

@pass varchar(20),

@firstname varchar(20),

@lastname varchar(20),

@patronymic varchar(20),

@number varchar(20)

AS

INSERT INTO Users\_ (User\_Login, User\_Password, Lastname, Firstname, Patronymic, Phone\_Number)

VALUES (@login, @pass, @firstname, @lastname, @patronymic, @number)

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_DeleteMyResrv] Script Date: 03/28/2017 11:29:30 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_DeleteMyResrv]

@id\_reserv int

AS

DELETE FROM Reservation WHERE Id\_Reservation = @id\_reserv

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Delete\_User] Script Date: 03/28/2017 11:29:28 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Delete\_User]

@idUser int

AS

DELETE FROM Users\_ WHERE Users\_.Id\_User = @idUser

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Delete\_Room\_Class] Script Date: 03/28/2017 11:29:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Delete\_Room\_Class]

@idHotel varchar(20),

@idClass int

AS

DELETE FROM Room\_Classes WHERE Room\_Classes.Id\_Hotel = @idHotel AND

Room\_Classes.Id\_Class = @idClass

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Delete\_Room] Script Date: 03/28/2017 11:29:25 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Delete\_Room]

@idHotel varchar(20),

@idRoom varchar(20)

AS

DELETE FROM Room WHERE Room.Id\_Hotel = @idHotel AND

Room.Id\_Room = @idRoom

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Delete\_Hotel] Script Date: 03/28/2017 11:29:23 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Delete\_Hotel]

@idHotel int

AS

DELETE FROM Hotel WHERE Hotel.Id\_Hotel = @idHotel

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Clear\_Old\_Reservations] Script Date: 03/28/2017 11:29:20 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Clear\_Old\_Reservations]

AS

DELETE FROM Reservation WHERE Reservation.Data\_Out < GETDATE()

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_DeleteMyResrv] Script Date: 03/23/2017 23:14:05 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_DeleteMyResrv]

@id\_reserv int

AS

DELETE FROM Reservation WHERE Id\_Reservation = @id\_reserv

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_DeleteMyResrv] Script Date: 03/23/2017 23:14:05 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_DeleteMyResrv]

@id\_reserv int

AS

DELETE FROM Reservation WHERE Id\_Reservation = @id\_reserv

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Login] Script Date: 03/23/2017 23:14:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Login]

@login varchar(20),

@pass varchar(20)

AS

SELECT u.Id\_User

FROM Users\_ AS u

WHERE u.User\_Login = @login AND u.User\_Password = @pass

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Select\_Free] Script Date: 03/28/2017 11:28:54 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Select\_Free]

@id\_hotel varchar(20),

@data\_in date,

@data\_out date

AS

/\*SELECT \* FROM Room

WHERE Room.Id\_Hotel = @id\_hotel\*/

IF EXISTS (SELECT \* FROM Reservation AS R

WHERE R.Id\_Hotel = @id\_hotel)

BEGIN

SELECT Room.Id\_Hotel, Room.Id\_Room, Room.Id\_Class FROM Room

WHERE NOT Room.Id\_Room = ANY (SELECT Reservation.Id\_Room FROM Reservation WHERE Reservation.Id\_Hotel = @id\_hotel)

AND Room.Id\_Hotel = @id\_hotel

END

ELSE

SELECT \* FROM Room WHERE Room.Id\_Hotel = @id\_hotel

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Select\_Hotels] Script Date: 03/23/2017 23:14:40 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Select\_Hotels]

AS

SELECT Hotel.Id\_Hotel, Hotel.Title, Hotel.StarCount, Hotel.[Address]

FROM Hotel

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[proc\_Select\_My\_Reservation] Script Date: 03/23/2017 23:14:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[proc\_Select\_My\_Reservation]

@user int

AS

SELECT Hotel.Title, R.Id\_Room, R.Data\_In, R.Data\_Out

FROM Reservation AS R, Hotel

WHERE R.Id\_User = @user

USE [HotelsDB]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[re\_backup] Script Date: 03/24/2017 02:53:12 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[re\_backup]

AS

DECLARE @date varchar(25)

DECLARE @name varchar(100)

SELECT @date = CONVERT(VARCHAR(30), GETDATE(), 104)

SELECT @name = 'D:\SQLB\Backup\_'+@date+'.Bak'

BACKUP DATABASE HotelsDB

TO DISK = @name

WITH FORMAT,

MEDIANAME = 'Z\_SQLServerBackups',

NAME = 'Full Backup of AdventureWorks2012';

**Приложение для администратора**

**Admin\_Form.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kyrs

{

public partial class Admin\_Form : Form

{

DB\_Work wdb;

int Selector = 0;

public Admin\_Form(DB\_Work \_wdb)

{

InitializeComponent();

this.wdb = \_wdb;

}

private void BC\_Hotel\_Click(object sender, EventArgs e)

{

dataGridView1 = wdb.FillHotel(dataGridView1);

Selector = 0;

}

private void BC\_Room\_Click(object sender, EventArgs e)

{

Selector = 1;

dataGridView1 = wdb.FillRoom(dataGridView1);

if (Selector == 3) CB\_SelectUser.Visible = true;

else CB\_SelectUser.Visible = false;

}

private void BC\_Class\_Click(object sender, EventArgs e)

{

Selector = 2;

dataGridView1 = wdb.FillRoomClasses(dataGridView1);

if (Selector == 3) CB\_SelectUser.Visible = true;

else CB\_SelectUser.Visible = false;

}

private void BC\_Reservation\_Click(object sender, EventArgs e)

{

Selector = 3;

if (Selector == 3) CB\_SelectUser.Visible = true;

else CB\_SelectUser.Visible = false;

//dataGridView1 = wdb.FillReservation(dataGridView1, int.Parse(CB\_SelectUser.Text.ToString()));

}

private void BC\_User\_Click(object sender, EventArgs e)

{

Selector = 4;

if (Selector == 3) CB\_SelectUser.Visible = true;

else CB\_SelectUser.Visible = false;

dataGridView1 = wdb.FillLogins(dataGridView1);

}

private void B\_AddInfo\_Click(object sender, EventArgs e)

{

switch (Selector)

{

case 0: { var NewHotel = new New\_Hotel\_Form(wdb); NewHotel.ShowDialog(); } break;

case 1: { var NewRoom = new New\_Room\_Form(wdb); NewRoom.ShowDialog(); } break;

case 2: { var NewClasses = new New\_Classes(wdb); NewClasses.ShowDialog(); } break;

case 3: { } break;

case 4: { var NewLogin = new Registr\_Form(wdb); NewLogin.ShowDialog(); } break;

}

}

private void Admin\_Form\_MouseClick(object sender, MouseEventArgs e)

{

if (Selector == 3) CB\_SelectUser.Visible = true;

else CB\_SelectUser.Visible = false;

}

private void CB\_SelectUser\_SelectedValueChanged(object sender, EventArgs e)

{

// dataGridView1 = wdb.FillReservation(dataGridView1, int.Parse(CB\_SelectUser.Text));

}

private void Admin\_Form\_FormClosing(object sender, FormClosingEventArgs e)

{

// Application.Restart();

}

private void BC\_Log\_Click(object sender, EventArgs e)

{

Selector = -1;

if (Selector == 3) CB\_SelectUser.Visible = true;

else CB\_SelectUser.Visible = false;

dataGridView1 = wdb.FillLog(dataGridView1);

}

private void Admin\_Form\_Load(object sender, EventArgs e)

{

CB\_SelectUser = wdb.FillBoxUser(CB\_SelectUser);

}

private void CB\_SelectUser\_VisibleChanged(object sender, EventArgs e)

{

if (CB\_SelectUser.Visible)

button1.Visible = true;

else

button1.Visible = false;

}

private void button1\_Click(object sender, EventArgs e)

{

dataGridView1 = wdb.FillReservation(dataGridView1, int.Parse(CB\_SelectUser.Text));

}

private void B\_UnLogin\_Click(object sender, EventArgs e)

{

Application.Restart();

}

private void B\_Delete\_Info\_Click(object sender, EventArgs e)

{

switch (Selector)

{

case 0:

{

if (wdb.DeleteHotel(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString()) == -1)

MessageBox.Show("Этот элемент невозможно удалить, пока он используется");

dataGridView1 = wdb.FillHotel(dataGridView1);

} break;

case 1:

{

if (wdb.DeleteRoom(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString(), (dataGridView1[1, dataGridView1.CurrentRow.Index].Value.ToString())) == -1)

MessageBox.Show("Этот элемент невозможно удалить, пока он используется");

dataGridView1 = wdb.FillRoom(dataGridView1);

} break;

case 2:

{

if (wdb.DeleteRoomClass(dataGridView1[2, dataGridView1.CurrentRow.Index].Value.ToString(), int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString())) == -1)

MessageBox.Show("Этот элемент невозможно удалить, пока он используется");

dataGridView1 = wdb.FillRoomClasses(dataGridView1);

} break;

case 3: { } break;

case 4:

{

if (wdb.DeleteUser(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString())) == -1)

MessageBox.Show("Этот элемент невозможно удалить, пока он используется");

dataGridView1 = wdb.FillLogins(dataGridView1);

} break;

}

}

}

}

**New\_Classes.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kyrs

{

public partial class New\_Classes : Form

{

DB\_Work wdb;

public New\_Classes(DB\_Work \_wdb)

{

InitializeComponent();

this.wdb = \_wdb;

CB\_IdHotel = wdb.FiilBoxIdHotel(CB\_IdHotel);

}

private void button1\_Click(object sender, EventArgs e)

{

wdb.AddClasses(CB\_IdHotel.Text, E\_Class.Text, int.Parse(E\_Cost.Text));

this.Close();

}

private void button2\_Click(object sender, EventArgs e)

{

CB\_IdHotel.SelectedIndex = 0;

E\_Class.Text = "";

E\_Cost.Text = "";

}

private void button3\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**New\_Hotel\_Form.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kyrs

{

public partial class New\_Hotel\_Form : Form

{

DB\_Work wdb;

public New\_Hotel\_Form(DB\_Work \_wdb)

{

InitializeComponent();

this.wdb = \_wdb;

}

private void New\_Hotel\_Form\_Load(object sender, EventArgs e)

{

}

private void B\_Add\_Click(object sender, EventArgs e)

{

//wdb.AddHotel(E\_IdHotel.Text, E\_Title.Text, E\_Address.Text, int.Parse(N\_StarCount.TextAlign.ToString()));

wdb.AddHotel(E\_IdHotel.Text, E\_Title.Text, E\_Address.Text, int.Parse(textBox1.Text));

this.Close();

}

private void B\_Cancle\_Click(object sender, EventArgs e)

{

this.Close();

}

private void B\_Clear\_Click(object sender, EventArgs e)

{

E\_Address.Text = "";

E\_IdHotel.Text = "";

E\_Title.Text = "";

textBox1.Text = "";

}

}

}

**New\_Room\_Form.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kyrs

{

public partial class New\_Room\_Form : Form

{

DB\_Work wdb;

public New\_Room\_Form(DB\_Work \_wdb)

{

InitializeComponent();

this.wdb = \_wdb;

CB\_IdHotel = wdb.FiilBoxIdHotel(CB\_IdHotel);

CB\_Class = wdb.FillBoxClass(CB\_Class);

}

private void New\_Room\_Form\_Load(object sender, EventArgs e)

{

}

private void B\_Add\_Click(object sender, EventArgs e)

{

wdb.AddRoom(CB\_IdHotel.Text,E\_IdRoom.Text,CB\_IdHotel.SelectedIndex);

this.Close();

}

private void B\_Clear\_Click(object sender, EventArgs e)

{

CB\_Class.SelectedIndex = 0;

E\_IdRoom.Text = "";

CB\_IdHotel.SelectedIndex = 0;

}

private void B\_Cancle\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}**Приложение для пользователя**

**Users\_Form.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Kyrs

{

public partial class Users\_Form : Form

{

DB\_Work wdb;

int Selector = 0;

String Selected\_Hotel;

bool roomUp = false;

bool UslesVar = false;

public Users\_Form(DB\_Work \_wdb)

{

InitializeComponent();

this.wdb = \_wdb;

}

private void Users\_Form\_Load(object sender, EventArgs e)

{

dataGridView1 = wdb.FillHotel(dataGridView1);

}

private void BS\_Rooms\_Click(object sender, EventArgs e)

{

UslesVar = false;

Selector = 0;

roomUp = false;

B\_ReservDelete.Visible = false;

L\_Hotel.Visible = true;

dataGridView1 = wdb.FillHotel(dataGridView1);

}

private void BS\_Reservations\_Click(object sender, EventArgs e)

{

UslesVar = true;

Selector = 0;

B\_ReservDelete.Visible = true;

L\_Hotel.Visible = false;

groupBox1.Visible = false;

dataGridView1 = wdb.FillReservation(dataGridView1, wdb.ActivUser);

}

private void Users\_Form\_FormClosing(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

private void B\_Select\_Click(object sender, EventArgs e)

{

//dataGridView1 = wdb.FillFree(dataGridView1, dataGridView1[0, SelectedString].Value.ToString());

if (wdb.DeleteMyReservation(int.Parse(dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString())) != -1)

MessageBox.Show("Выбранное резеревирование было отменено!");

else

MessageBox.Show("Произошла ошибка"+wdb.ex.ToString());

dataGridView1 = wdb.FillReservation(dataGridView1, wdb.ActivUser);

}

private void B\_UnLogin\_Click(object sender, EventArgs e)

{

Application.Restart();

}

private void B\_SelectDay\_Click(object sender, EventArgs e)

{

dataGridView1 = wdb.FillFree(dataGridView1, Selected\_Hotel, dateTimePicker1.Value, dateTimePicker2.Value);

roomUp = true;

Selector = 1;

}

private void B\_Reserv\_Click(object sender, EventArgs e)

{

wdb.AddReservation(Selected\_Hotel, dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString(), wdb.ActivUser, dateTimePicker1.Value, dateTimePicker2.Value);

MessageBox.Show("Комната зарезервирована.");

Selector = 0;

}

private void dataGridView1\_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

switch (Selector)

{

case 0:

{

Selected\_Hotel = dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString();

if (!UslesVar)

groupBox1.Visible = true;

L\_Hotel.Text = "Выбран отель № " + Selected\_Hotel;

} break;

case 1:

{

if (groupBox1.Visible)

if (roomUp)

groupBox1.Text = "Зарезервировать комнату № " + dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString();

} break;

}

}

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

dateTimePicker2.Value = dateTimePicker1.Value;

}

private void dateTimePicker2\_ValueChanged(object sender, EventArgs e)

{

if (dateTimePicker2.Value < dateTimePicker1.Value)

dateTimePicker2.Value = dateTimePicker1.Value;

}

}

}

**Registr\_Form.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kyrs

{

public partial class Registr\_Form : Form

{

DB\_Work wdb;

public Registr\_Form(DB\_Work \_wdb)

{

InitializeComponent();

this.wdb = \_wdb;

}

private void Registr\_Form\_Load(object sender, EventArgs e)

{

}

private void B\_Create\_Click(object sender, EventArgs e)

{

if ((E\_Firstname.Text == "") &&

(E\_Lastname.Text == "") &&

(E\_NewLogin.Text == "") &&

(E\_NewPass.Text == "") &&

(E\_NewPassRe.Text == "") &&

(E\_Number.Text == "") &&

(E\_Patronymic.Text == ""))

MessageBox.Show("Не все поля заполнены!");

else

if (E\_NewPass.Text != E\_NewPassRe.Text)

MessageBox.Show("Пароли не совпадают!");

else

if (wdb.AddNewUser(E\_NewLogin.Text, E\_NewPass.Text, E\_Lastname.Text, E\_Firstname.Text, E\_Patronymic.Text, E\_Number.Text) == 0)

{

MessageBox.Show("Пользователь создан.");

Application.Restart();

}

else

MessageBox.Show("Что-то не так!" + wdb.ex.ToString());

}

private void B\_Empty\_Click(object sender, EventArgs e)

{

E\_Firstname.Text = "";

E\_Lastname.Text = "";

E\_NewLogin.Text = "";

E\_NewPass.Text = "";

E\_NewPassRe.Text = "";

E\_Number.Text = "";

E\_Patronymic.Text = "";

}

private void B\_LoginForm\_Click(object sender, EventArgs e)

{

Application.Restart();

}

}

}

**Общая часть приложения**

**DB\_Work.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

using System.Windows.Forms;

using System.Data;

public class DB\_Work

{

public String connectionString = "";

public Exception ex;

public int ActivUser;

public DB\_Work()

{

connectionString = @"Data Source=TPUKCTEP; Initial Catalog=HotelsDB;Integrated Security=True";

}

public int AddNewUser(String \_login, String \_pass, String \_lastname, String \_firstname, String \_patronymic, String \_number)

{

String sqlExpression = "proc\_Add\_User";

int info = 0;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@login", \_login);

command.Parameters.AddWithValue("@pass", \_pass);

command.Parameters.AddWithValue("@firstname", \_firstname);

command.Parameters.AddWithValue("@lastname", \_lastname);

command.Parameters.AddWithValue("@patronymic", \_patronymic);

command.Parameters.AddWithValue("@number", \_number);

var result = command.ExecuteNonQuery();

}

catch (Exception \_ex)

{

ex = \_ex;

info = -1;

}

finally

{

connection.Close();

}

return info;

}

public int AddClasses(String \_idHotel, String \_class, int \_cost)

{

String sqlExpression = "proc\_Add\_Classes";

int info = 0;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@id\_hotel", \_idHotel);

command.Parameters.AddWithValue("@class", \_class);

command.Parameters.AddWithValue("@cost", \_cost);

var result = command.ExecuteNonQuery();

}

catch (Exception \_ex)

{

ex = \_ex;

info = -1;

}

finally

{

connection.Close();

}

return info;

}

public int AddHotel(String \_idHotel, String \_title, String \_address, int \_startCount)

{

String sqlExpression = "proc\_Add\_Hotel";

int info = 0;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@id\_hotel", \_idHotel);

command.Parameters.AddWithValue("@title", \_title);

command.Parameters.AddWithValue("@address", \_address);

command.Parameters.AddWithValue("@starcount", \_startCount); ;

var result = command.ExecuteNonQuery();

}

catch (Exception \_ex)

{

ex = \_ex;

info = -1;

}

finally

{

connection.Close();

}

return info;

}

public int AddRoom(String \_idHotel, String \_idRoom, int \_idClass)

{

String sqlExpression = "proc\_Add\_Room";

int info = 0;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@id\_hotel", \_idHotel);

command.Parameters.AddWithValue("@id\_room", \_idRoom);

command.Parameters.AddWithValue("@id\_class", \_idClass);

var result = command.ExecuteNonQuery();

}

catch (Exception \_ex)

{

ex = \_ex;

info = -1;

}

finally

{

connection.Close();

}

return info;

}

public int AddReservation(String \_idHotel, String \_idRoom, int \_idUser, DateTime \_dataIn, DateTime \_dateOut)

{

String sqlExpression = "proc\_Add\_Reservation";

int info = 0;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@id\_hotel", \_idHotel);

command.Parameters.AddWithValue("@id\_room", \_idRoom);

command.Parameters.AddWithValue("@id\_user", \_idUser);

command.Parameters.AddWithValue("@data\_in", \_dataIn);

command.Parameters.AddWithValue("@data\_out", \_dateOut);

var result = command.ExecuteNonQuery();

}

catch (Exception \_ex)

{

ex = \_ex;

info = -1;

}

finally

{

connection.Close();

}

return info;

}

public DataGridView FillHotel(DataGridView \_dgv)

{

DataGridView dgv = \_dgv;

String sqlExpression = "proc\_Select\_Hotels";

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataSet ds = new DataSet();

adapter.Fill(ds);

dgv.DataSource = ds.Tables[0];

}

catch (Exception \_ex)

{

ex = \_ex;

}

finally

{

connection.Close();

}

return dgv;

}

public int GetLogin(String \_login, String \_pass)

{

int i = 1;

String sqlExpression = "proc\_Login";

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@login", \_login);

command.Parameters.AddWithValue("@pass", \_pass);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

ActivUser = int.Parse(reader["Id\_User"].ToString());

}

catch (Exception \_ex)

{

ex = \_ex;

i = -1;

}

finally

{

connection.Close();

}

return i;

}

public DataGridView FillReservation(DataGridView \_dgv, int \_idUser)

{

DataGridView dgv = \_dgv;

String sqlExpression = "proc\_Select\_My\_Reservation";

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@user", \_idUser);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataSet ds = new DataSet();

adapter.Fill(ds);

dgv.DataSource = ds.Tables[0];

}

catch (Exception \_ex)

{

ex = \_ex;

}

finally

{

connection.Close();

}

return dgv;

}

public ComboBox FiilBoxIdHotel(ComboBox \_cb)

{

ComboBox cb = \_cb;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Hotel", connection);

DataTable tbl = new DataTable();

adapter.Fill(tbl);

cb.DataSource = tbl;

cb.DisplayMember = "Id\_Hotel";

cb.ValueMember = "id";

}

catch (Exception \_ex)

{

ex = \_ex;

}

finally

{

connection.Close();

}

return cb;

}

public ComboBox FillBoxClass(ComboBox \_cb)

{

ComboBox cb = \_cb;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Room\_Classes", connection);

DataTable tbl = new DataTable();

adapter.Fill(tbl);

cb.DataSource = tbl;

cb.DisplayMember = "Id\_Class";

cb.ValueMember = "id";

}

catch (Exception \_ex)

{

ex = \_ex;

}

finally

{

connection.Close();

}

return cb;

}

public ComboBox FillBoxUser(ComboBox \_cb)

{

ComboBox cb = \_cb;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

SqlDataAdapter adapter = new SqlDataAdapter("SELECT \* FROM Users\_", connection);

DataTable tbl = new DataTable();

adapter.Fill(tbl);

cb.DataSource = tbl;

cb.DisplayMember = "Id\_User";

cb.ValueMember = "id";

}

catch (Exception \_ex)

{

ex = \_ex;

}

finally

{

connection.Close();

}

return cb;

}

public DataGridView FillRoom(DataGridView \_dgv)

{

DataGridView dgv = \_dgv;

String sqlExpression = "SELECT \* FROM Room";

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataSet ds = new DataSet();

adapter.Fill(ds);

dgv.DataSource = ds.Tables[0];

}

catch (Exception \_ex)

{

ex = \_ex;

}

finally

{

connection.Close();

}

return dgv;

}

public DataGridView FillRoomClasses(DataGridView \_dgv)

{

DataGridView dgv = \_dgv;

String sqlExpression = "SELECT \* FROM Room\_Classes";

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataSet ds = new DataSet();

adapter.Fill(ds);

dgv.DataSource = ds.Tables[0];

}

catch (Exception \_ex)

{

ex = \_ex;

}

finally

{

connection.Close();

}

return dgv;

}

public DataGridView FillLogins(DataGridView \_dgv)

{

DataGridView dgv = \_dgv;

String sqlExpression = "SELECT \* FROM Users\_ WHERE Id\_User <> 1";

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataSet ds = new DataSet();

adapter.Fill(ds);

dgv.DataSource = ds.Tables[0];

}

catch (Exception \_ex)

{

ex = \_ex;

}

finally

{

connection.Close();

}

return dgv;

}

public DataGridView FillFree(DataGridView \_dgv, String \_idHotel, DateTime \_dataIn, DateTime \_dataOut)

{

DataGridView dgv = \_dgv;

String sqlExpression = "proc\_Select\_Free";

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@id\_hotel", \_idHotel);

command.Parameters.AddWithValue("@data\_in", \_dataIn);

command.Parameters.AddWithValue("@data\_out", \_dataOut);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataSet ds = new DataSet();

adapter.Fill(ds);

dgv.DataSource = ds.Tables[0];

}

catch (Exception \_ex)

{

ex = \_ex;

}

finally

{

connection.Close();

}

return dgv;

}

public DataGridView FillLog(DataGridView \_dgv)

{

DataGridView dgv = \_dgv;

String sqlExpression = "SELECT \* FROM Reg\_Log";

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataSet ds = new DataSet();

adapter.Fill(ds);

dgv.DataSource = ds.Tables[0];

}

catch (Exception \_ex)

{

ex = \_ex;

}

finally

{

connection.Close();

}

return dgv;

}

public int DeleteMyReservation(int \_idReservation)

{

String sqlExpression = "proc\_DeleteMyResrv";

int info = 0;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@id\_reserv", \_idReservation);

var result = command.ExecuteNonQuery();

}

catch (Exception \_ex)

{

ex = \_ex;

info = -1;

}

finally

{

connection.Close();

}

return info;

}

public int ClearOldReservation()

{

String sqlExpression = "proc\_Clear\_Old\_Reservations";

int info = 0;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

var result = command.ExecuteNonQuery();

}

catch (Exception \_ex)

{

ex = \_ex;

info = -1;

MessageBox.Show(ex.ToString());

}

finally

{

connection.Close();

}

return info;

}

public int DeleteHotel(String \_idHotel)

{

String sqlExpression = "proc\_Delete\_Hotel";

int info = 0;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@idHotel", \_idHotel);

var result = command.ExecuteNonQuery();

}

catch (Exception \_ex)

{

ex = \_ex;

info = -1;

}

finally

{

connection.Close();

}

return info;

}

public int DeleteRoom(String \_idHotel, String \_idRoom)

{

String sqlExpression = "proc\_Delete\_Room";

int info = 0;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@idHotel", \_idHotel);

command.Parameters.AddWithValue("@idRoom", \_idRoom);

var result = command.ExecuteNonQuery();

}

catch (Exception \_ex)

{

ex = \_ex;

info = -1;

}

finally

{

connection.Close();

}

return info;

}

public int DeleteRoomClass(String \_idHotel, int \_idClass)

{

String sqlExpression = "proc\_Delete\_Room\_Class";

int info = 0;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@idHotel", \_idHotel);

command.Parameters.AddWithValue("@idClass", \_idClass);

var result = command.ExecuteNonQuery();

}

catch (Exception \_ex)

{

ex = \_ex;

info = -1;

}

finally

{

connection.Close();

}

return info;

}

public int DeleteUser(int \_idUser)

{

String sqlExpression = "proc\_Delete\_User";

int info = 0;

SqlConnection connection = new SqlConnection();

connection.ConnectionString = connectionString;

try

{

connection.Open();

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@idUser", \_idUser);

var result = command.ExecuteNonQuery();

}

catch (Exception \_ex)

{

ex = \_ex;

info = -1;

}

finally

{

connection.Close();

}

return info;

}

}

# СПИСОК ЛИТЕРАТУРЫ

1. Архитектура Клиент-сервер <http://www.4stud.info/networking/lecture5.html>
2. Справочник по SQL Server <https://msdn.microsoft.com/>
3. Исследование предметной области по http://www.booking.com/