Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»**

(ВлГУ)

Колледж инновационных технологий и предпринимательства

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ**

по дисциплине «Теория разработки и защиты баз данных»

Тема: «Аудиторный фонд многокорпусного здания: разработка и администрирование базы данных, разработка клиентского приложения»

Выполнил:

ст. гр. ПКсп-116

Молодцов Д.М

Приняли:

Павлова О.Н.

Куприянов А.А.

Владимир 2019

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc23889074)

[1 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 4](#_Toc23889075)

[1.1 ER-ДИАГРАММА 4](#_Toc23889076)

[2 ПОДГОТОВКА БАЗЫ ДАННЫХ 6](#_Toc23889077)

[2.1 СЛОВАРЬ ДАННЫХ 6](#_Toc23889078)

[2.2 ПРОЦЕДУРЫ И ФУНКЦИИ 9](#_Toc23889079)

[3 РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ 11](#_Toc23889080)

[4 ИМПОРТ И ЭКСПОРТ ДАННЫХ 13](#_Toc23889081)

[4.1 ИМПОРТ ДАННЫХ ИЗ MS EXCEL 13](#_Toc23889082)

[4.2 ИМПОРТ ДАННЫХ С ПОМОЩЬЮ BULK INSERT 15](#_Toc23889083)

[4.3 КОМАНДНЫЙ ФАЙЛ 16](#_Toc23889084)

[5 УПРАВЛЕНИЕ ДОСТУПОМ 19](#_Toc23889085)

[6 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 20](#_Toc23889086)

[6.1 СТРУКТУРА ПРИЛОЖЕНИЯ 20](#_Toc23889087)

[6.2 ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС 21](#_Toc23889088)

[ЗАКЛЮЧЕНИЕ 26](#_Toc23889089)

[ПРИЛОЖЕНИЕ А 27](#_Toc23889090)

[ПРИЛОЖЕНИЕ Б 29](#_Toc23889091)

[ПРИЛОЖЕНИЕ В 31](#_Toc23889092)

[ПРИЛОЖЕНИЕ Г 35](#_Toc23889093)

[ПРИЛОЖЕНИЕ Д 37](#_Toc23889094)

[ПРИЛОЖЕНИЕ Е 40](#_Toc23889095)

[ПРИЛОЖЕНИЕ Ж 44](#_Toc23889096)

# Введение

В данной работе стоит задача создать программный модуль «Ресторанный бизнес: бронирование столиков», который может быть использован в различных ресторанах. Так же, для хранения данных, которыми будет оперировать программа, необходимо разработать базу данных.

Пользователь приложения может выполнять следующие действия:

* Просматривать информацию о столике (при необходимости, это можно делать по дате).
* Бронировать столик.
* Отменять бронирование.

Для того чтобы забронировать столик, пользователь должен будет ввести: имя, фамилию, номер телефона, дату и время, на которое бронируется столик.

# Проектирование базы данных

## ER-диаграмма

База данных учета столиков в ресторане должна иметь следующий набор сущностей с, присущими им, атрибутами:

* Информация о столике: Номер столика, количество мест, цвет, материал, форму и картинку с изображением столика.
* Информация о бронировании: Номер бронирования, имя клиента, фамилия клиента, номер телефона клиента, время бронирования и дата бронирования.

Для связи этих двух сущностей служит отдельная ассоциативная таблица, реализующая тип связи «много-много».

Так же, чтобы было проще вносить изменения в такие атрибуты сущности “Table”, как: цвет, форма, материал, на каждый из этих свойств была создана отдельная сущность (Color, Material, Form-Factor, соответственно) и уже внутри сущности, описывающей столик, были добавлены ссылки на значения в этих таблицах.

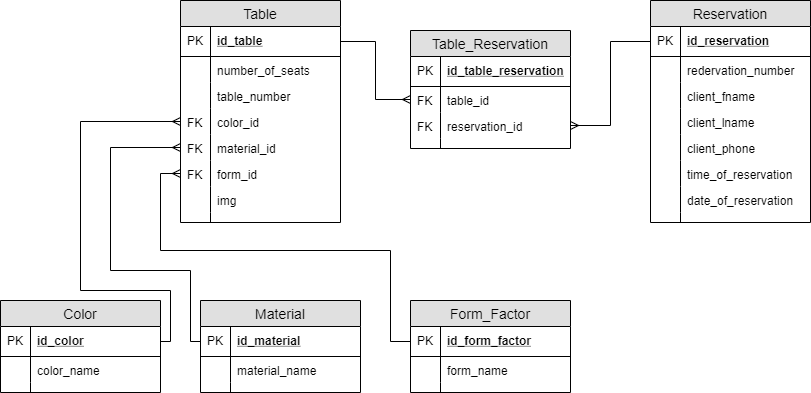


Рисунок 1.1 – ER-диаграмма

# подготовка базы данных

## Словарь данных

|  |  |  |  |
| --- | --- | --- | --- |
| Table | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | id\_talble | Да | Идентификационный номер столика |
|  | table\_number | Да | Номер столика |
|  | number\_of\_seats | Да | Количество мест за столиком |
| Внешний | color\_id | Да | Внешний ключ к таблице Color |
| Внешний | material\_id | Да | Внешний ключ к таблице Material |
| Внешний | form\_id | Да | Внешний ключ к таблице Form\_Factor |
|  | img | Нет | Массив байт для хранения картинки |

|  |  |  |  |
| --- | --- | --- | --- |
| Reservation | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | id\_reservation | Да | Идентификационный номер брони |
|  | reservation\_number | Да | Номер бронирования |
|  | Client\_fname | Да | Имя клиента |
|  | client\_lname | Да | Фамилия клиента |
|  | client\_phone | Да | Номер телефона клиента |
|  | time\_of\_reservation | Да | Время бронирования |
|  | date\_of\_reservation | Да | Дата бронирования |

|  |  |  |  |
| --- | --- | --- | --- |
| Table\_Reservation | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | id\_table\_reservation | Да | Идентификационный номер ассоциативной таблицы |
| Внешний | table\_id | Да | Внешний ключ к таблице Table |
| Внешний | reservation\_id | Да | Внешний ключ к таблице Reservation |

|  |  |  |  |
| --- | --- | --- | --- |
| Color | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | id\_color | Да | Идентификационный номер цвета |
|  | color\_name | Да | Название цвета |

|  |  |  |  |
| --- | --- | --- | --- |
| Material | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | id\_material | Да | Идентификационный номер материала |
|  | material\_name | Да | Название материала |

|  |  |  |  |
| --- | --- | --- | --- |
| Form\_Factor | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | id\_form\_factor | Да | Идентификационный номер формы |
|  | form\_name | Да | Название формы |

Итоговая диаграмма классов, сгенерированная после создания всех таблиц выглядит следующим образом:

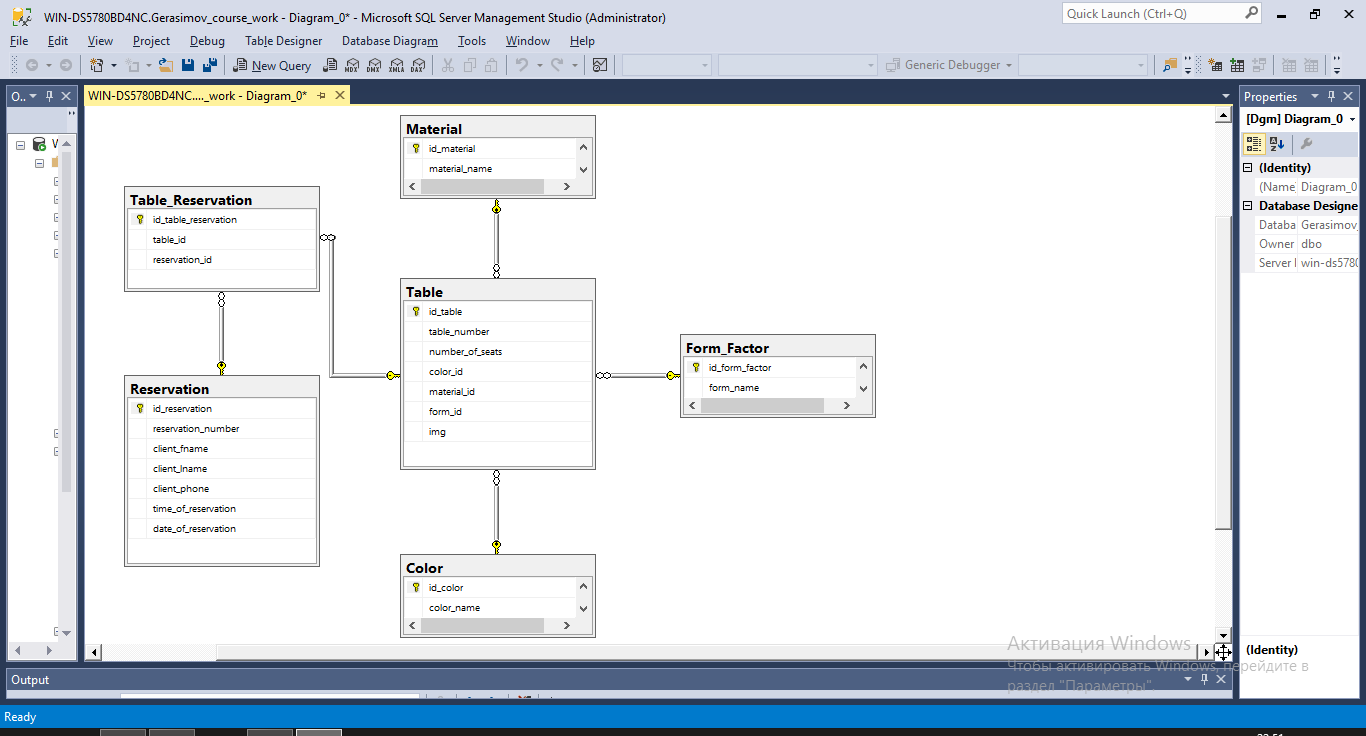


Рисунок 2.1 – Диаграмма базы данных

Код создания таблиц можно посмотреть в приложении А.

## Процедуры и функции

Всего в базе данных реализованы: две хранимых процедуры и одна скалярная функция.

Первая хранимая процедура называется TableInfoOnDate и она предназначена для того, чтобы вывести всю информацию о столике по определенной дате. Эта процедура помогает узнать, забронирован ли столик данного числа.

На вход данная процедура принимает два параметра @idTable и @date, которые представляют собой номер столика и дату, соответственно.

На выходе формируется выборка, в которую будут входить следующие поля:

* reservation\_number;
* client\_fname;
* client\_lname;
* client\_phone;
* date\_of\_reservation;
* time\_of\_reservation.

Затем была разработана скалярная функция, которая ведет расчет количества забронированных столиков по определенной дате.

На вход данная функция принимает: @date для определения даты.

Выходным параметром данной функции является числовое значение, которое будет определять количество забронированных столиков в день, равный @date.

Вторая хранимая процедура, по своей сущности, является больше вспомогательной, т.к. в ней всего лишь вызывается созданная выше скалярная функция. Это сделано для более удобного вызова функции из клиентского приложения.

Данная процедура в качестве входных параметров принимает дату, по которой нужно определить количество бронирований, а на выход подается число, типа int, записанное в ячейку. Код создания хранимых процедур и функций можно посмотреть в приложении В.

# резервное копирование и восстановление

Для резервного копирования с использованием командного файла изначально необходимо написать sql-скрипт, который будет выполнять резервное копирование базы данных и сохранит созданный скрипт в виде файла с расширением .sql.

Пример такого скрипта представлен далее:

use master

go

backup database Gerasimov\_course\_work

to disk='D:\Backups\CourseWork\course\_work.bak'

go

Затем, чтобы запустить данный скрипт, необходимо с помощью любого текстового редактора создать файл с расширением .bat или .cmd, внутри которого прописать следующий код:

sqlcmd -S WIN-DS5780BD4NC -i SQLQuery\_backup.sql

Теперь, созданный таким образом, файл при запуске будет создавать резервную копию базы данных в указанной директории.

Аналогичным путем создадим sql-скрипт и командный файл для запуска процесса восстановления базы данных из резервной копии.

Sql-скрипт для восстановления:

use master

go

restore database Gerasimov\_course\_work

from disk = 'D:\Backups\CourseWork\course\_work.bak' with replace

go

Код командного файла:

sqlcmd -S WIN-DS5780BD4NC -i SQLQuery\_restore.sql

# ИМПОрт и экспорт данных

## Импорт данных из MS Excel

Перед процессом импорта необходимо было создать источник, из которого будет происходить импорт данных. Таким источником будет служить таблица, созданная в MS Excel. Для импорта или экспорта данных из Excel-таблицы используется мастер импорта и экспорта SQL Server. На начальной странице настройки необходимо выбрать источник данных.

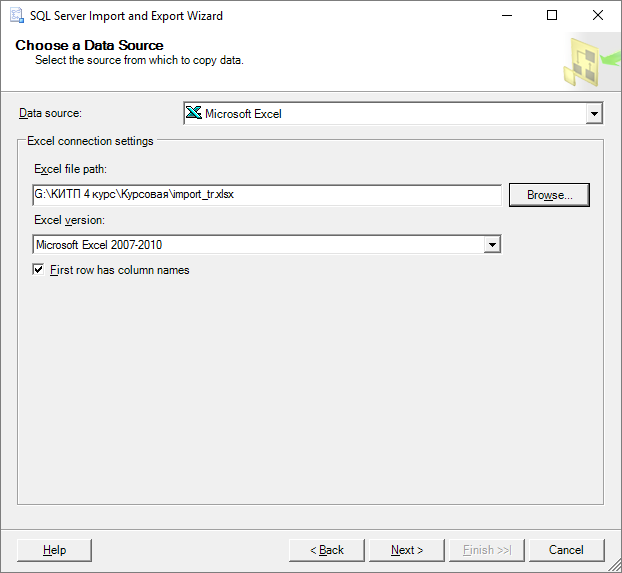


Рисунок 4.2 – Выбор источника импорта данных

На следующей странице нужно указать конечную цель импорта. В данном случае, это база данных курсовой работы.

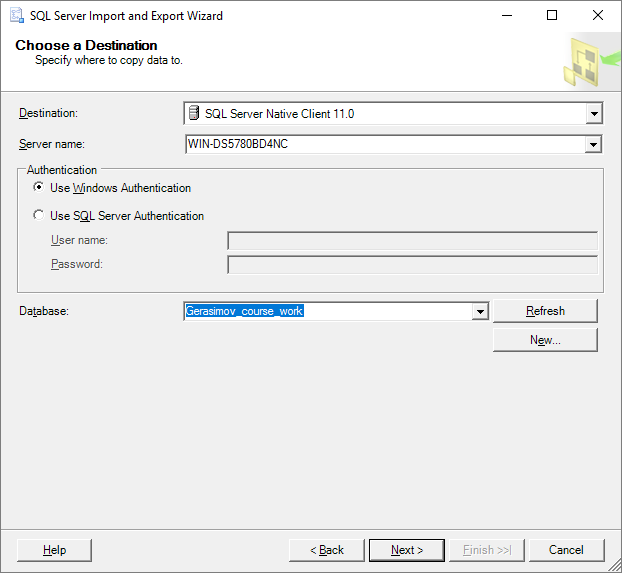


Рисунок 4.3 – Место назначения импорта данных

На следующей странице необходимо сопоставить страницы MS Excel и таблицы базы данных, для которых эти страницы предназначены.

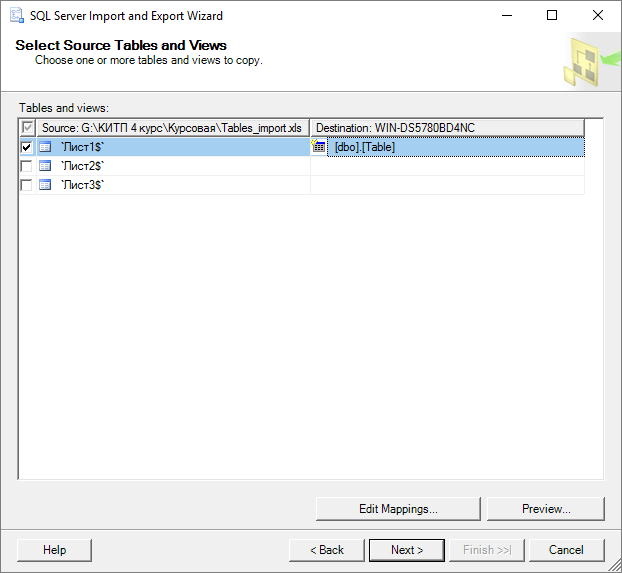


Рисунок 4.4 – Сопоставление страниц с таблицами

По завершению импорта данных, мастер импорта/экспорта уведомит пользователя о том, что импорт данных произошел успешно.

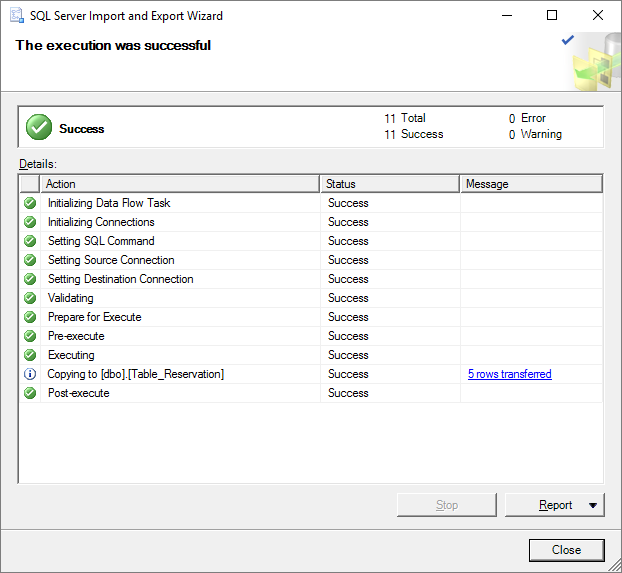


Рисунок 4.4 – Завершение импорта данных

## Импорт данных с помощью BULK INSERT

Sql-скрипт, который импортирует данные с помощью конструкции BULK INSERT выглядит следующим образом:

bulk insert Reservation from 'G:\КИТП 4 курс\Курсовая\Reservations\_import.csv'

with(keepidentity, fieldterminator=';', rowterminator='\n');

## Командный файл

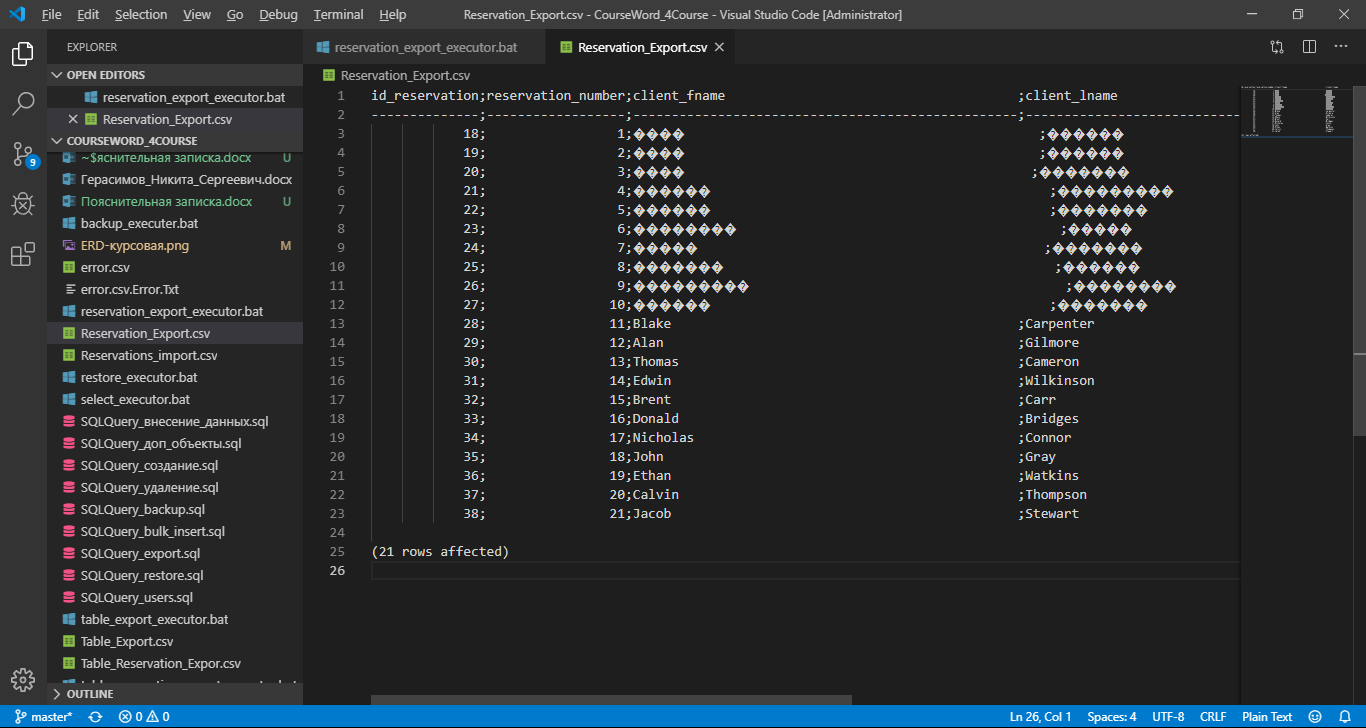
Согласно заданию необходимо создать командный файл, который будет выполнять sql-скрипт на экспорт данных в указанный файл, а так же командный файл, который будет выводить на экран всю информацию из какой-либо таблицы.

Для того, чтобы таким путем произвести экспорт данных в указанный файл нужно указать для какого сервера и какой базы данных будет выполняться команда. Затем написать код самого sql-скрипта, а после указать файл, в который будет записываться результат.

Пример экспорта для таблицы “Reservation” выглядит следующим образом:

sqlcmd -S WIN-DS5780BD4NC -d Gerasimov\_course\_work -Q "select \* from Reservation" -o "Reservation\_Export.csv" -s";" -w 700

Если запустить данный файл, то в директории, где он лежит можно найти файл Reservation\_Export.csv со следующим содержимым:

Рисунок 4.5 – Пример экспортированных данных

Аналогичным способом создадим командный файл, который будет выводить на экран необходимые данные. Текст такого файла выглядит следующим образом:

sqlcmd -S WIN-DS5780BD4NC

-d Gerasimov\_course\_work

-Q "select

r.client\_fname, r.client\_lname, r.time\_of\_reservation, r.date\_of\_reservation

from Table\_Reservation tr

join [Table] t on tr.table\_id=t.id\_table

join Reservation r on tr.reservation\_id=r.id\_reservation"

Теперь, если в консоли запустить данный командный файл, то на экране можно будет увидеть следующий результат:

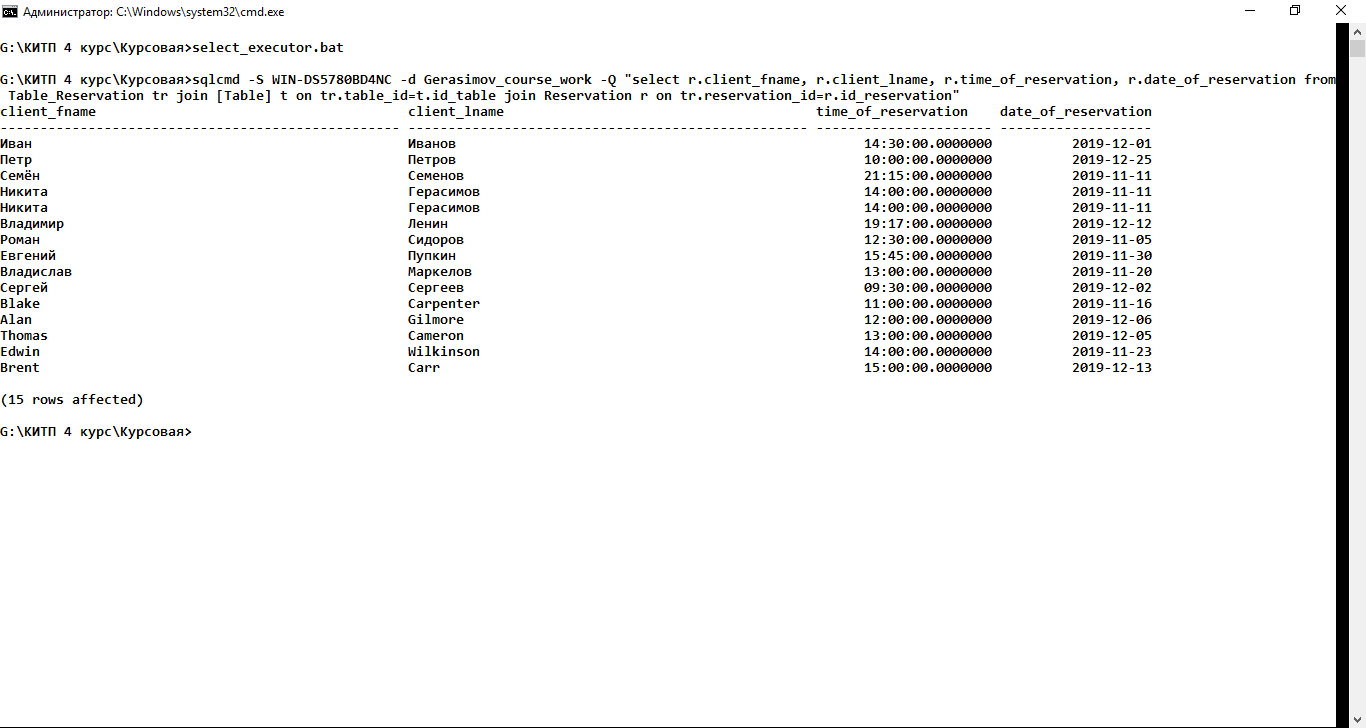


Рисунок 4.6 – Выполнение командного файла на вывод

# Управление доступом

Для управления доступом в базе данных с помощью sql-скрипты были созданы две роли:

* Manager, который имеет неограниченные возможности по работе с данными в базе данных.
* Hostes, основной задачей которого является лишь внесение данных в таблицы база данных. Соответственно, для него ограничена возможность удаления данных из базы.

Для каждой из ролей были созданы два пользователя со своими логинами и паролями соответственно.

Sql-скрипт, выполняющий создание вышеперечисленных ролей и пользователей представлен в приложении В.

# разработка клиентского приложения

## Структура приложения

Структура приложения состоит из следующих слоев:

* слой пользователя;
* слой логики;
* слой данных.

Слой пользователя представляет собой формы с определенными контроллерами (таблицы, кнопки, поля ввода и т.д.), которые реализованы с помощью Windows Forms.

Слой логики отвечает за взаимодействия пользователя с базами данных. На этом уровне обрабатываются данные, которые предоставляются пользователю или базе данных. На этом уровне происходит валидация входных данных, чтобы все корректно вносилось в базу данных.

Слой данных представляет собой модель ранее созданной базы данных, который полностью копирует модель, реализованную в SSMS. В приложении этот слой реализован с помощью Entity Framework.

## Пользовательский интерфейс

При запуске приложения перед пользователем появляется окно, которое содержит:

* строку поиска;
* таблицу со всеми столиками;
* форму для поиска информации о столике по дате;
* форму для подсчета количества бронирований по дате.

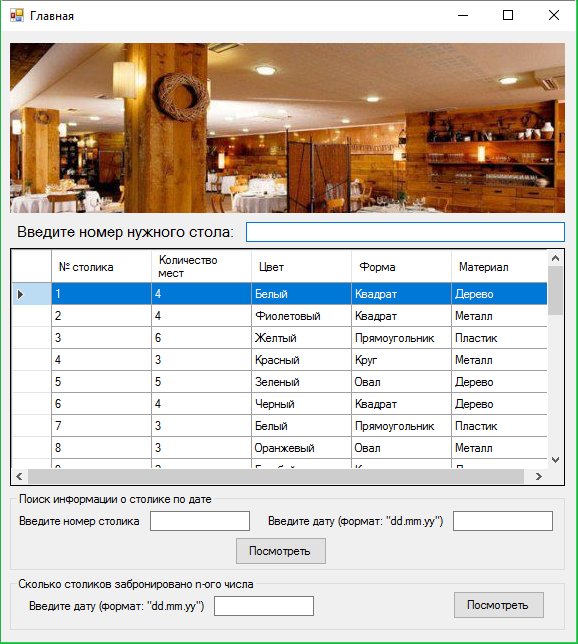


Рисунок 6.1 – Главная форма

Для того чтобы вывести информацию о столике по дате, необходимо ввести в поля номер столика и требуемую дату, а затем нажать на кнопку «Посмотреть». Для примера выведем информацию по столику №1 за 1.12.2019. Перед нами откроется следующая форма:

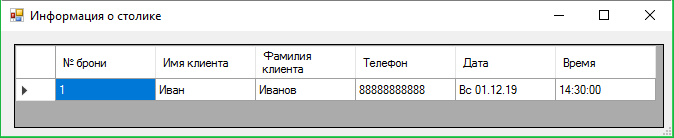


Рисунок 6.2 – Информация о столике

Для того чтобы узнать количество забронированных столиков по определенной дате, нужно ввести дату на форме, предназначенной для подобного поиска и нажать на кнопку «Посмотреть». Для примера проверим дату 01.12.2019. Получим следующее окно:

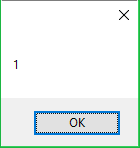


Рисунок 6.3 – Количество бронирований 01.12.19

Для того чтобы осуществить поиск нужного стола, в поле поиска нужно ввести номер искомого стола, таблица автоматически обновится. Чтобы вернуть все обратно – нужно стереть все из поля поиска.

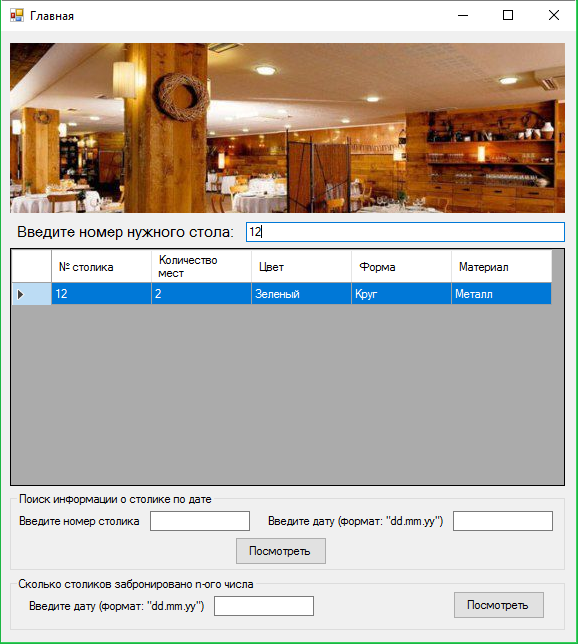


Рисунок 6.4 – Поиск стола по номеру

Двойной клик по записи в таблице со столиками приведет к тому, что откроется форма подробной информации по столику со всеми полями и таблице, в которой будет отображаться список все бронирований для этого столика.

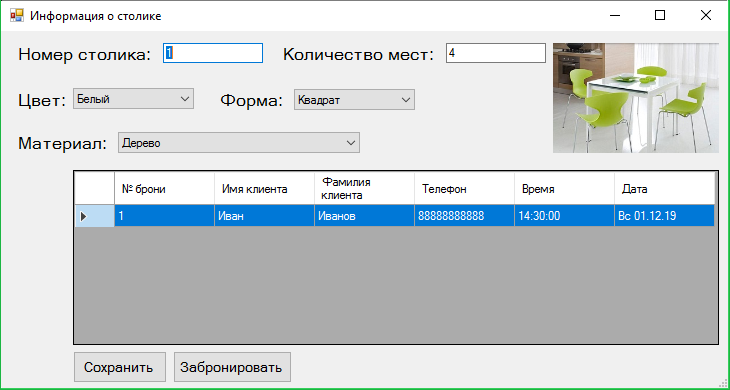


Рисунок 6.3 – Подробная информация по столику

На данной форме можно менять характеристики столика, а так добавить новое бронирование для него нажатием кнопки «Забронировать».

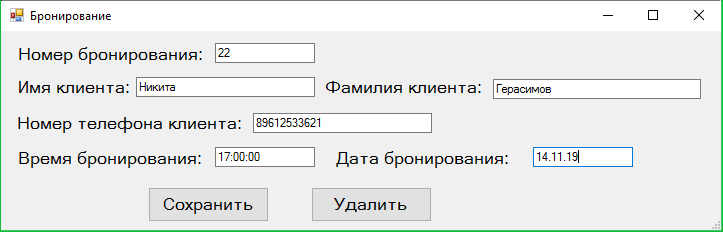


Рисунок 6.4 – Бронирование столика

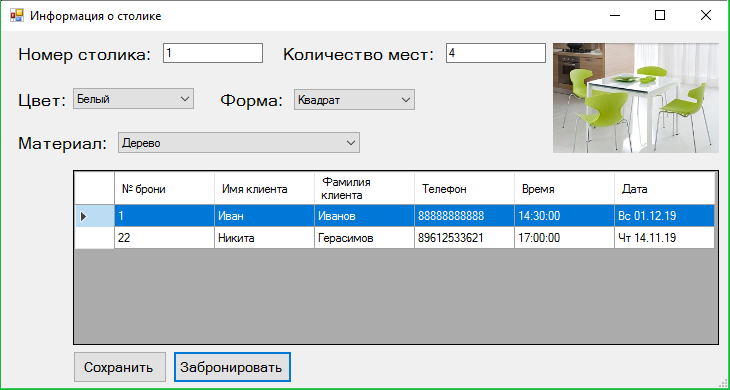


Рисунок 6.5 – Запись добавлена в таблицу

На данной форме так же можно удалить запись о бронировании. Для этого надо дважды кликнуть на запись с бронированием и на открывшейся форме нажать кнопку «Удалить».

Для примера, удалим ранее добавленную запись.

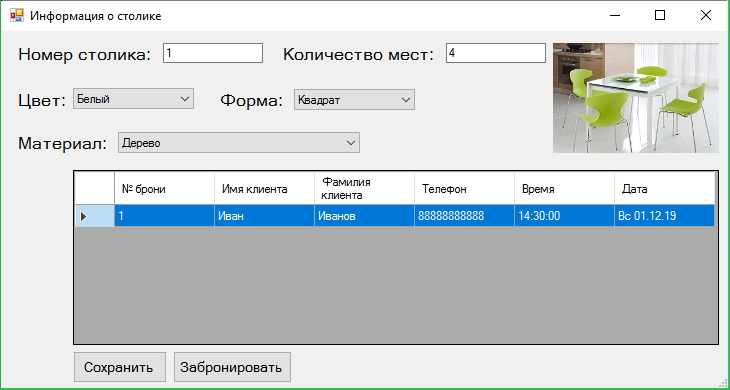


Рисунок 6.6 – Запись удалена

Так же с помощью показанных выше форм можно изменять информацию о бронировании (изменить время, дату, телефон). При сохранении изменений логическая часть приложения проверит, доступна ли желаемая дата и нет ли для данного столика бронирований в это время. Если столик в это время уже занят – пользователь получит ошибку.

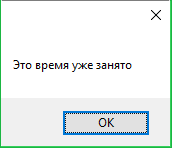


Рисунок 6.7 – Ошибка бронирования

# заключение

Результатом выполнения данной работы стала база данных, позволяющая организовать работу системы учёта списка столиков. Разработанная модель удовлетворяет всем требованиям, предъявленным в задании, позволяет добавлять, удалять и редактировать записи списка. Для работы с базой данных было разработано приложение, которое грамотно реализует все функции, которые необходимы для системы, описанной в задании.

Программа разработана таким образом, чтобы избежать возможного появления ошибок при работе с сервером базы данных. Там, где они могут встретиться, это предусмотрено и обрабатывается должным образом, а пользователь будет об этом уведомлен.

Работа была выполнена в среде программирования Visual Studio 2017. и Microsoft SQL Server Management Studio.

Во время выполнения курсовой работы изучены и закреплены навыки создания sql-скриптов для проектирования базы данных.

# приложение а

Создание БД

create database Gerasimov\_course\_work

go

use Gerasimov\_course\_work

create table Color

(

id\_color int primary key identity,

color\_name varchar(20) not null

);

go

create table Material

(

id\_material int primary key identity,

material\_name varchar(20) not null

);

go

create table Form\_Factor

(

id\_form\_factor int primary key identity,

form\_name varchar(20) not null

);

go

create table [Table]

(

id\_table int primary key identity,

table\_number int not null unique,

number\_of\_seats int not null,

color\_id int references Color(id\_color) not null,

material\_id int references Material(id\_material) not null,

form\_id int references Form\_Factor(id\_form\_factor) not null,

img varbinary(MAX)

);

go

create table Reservation

(

id\_reservation int primary key identity,

reservation\_number int not null unique,

client\_fname varchar(50) not null,

client\_lname varchar(50) not null,

client\_phone varchar(11) not null,

time\_of\_reservation Time not null,

date\_of\_reservation Date not null,

);

go

create table Table\_Reservation

(

id\_table\_reservation int primary key identity,

table\_id int references [Table](id\_table) not null,

reservation\_id int references Reservation(id\_reservation) not null

);

# приложение Б

Заполнение таблиц

insert into AudienceType(TypeOf, Cabinet,Capacity,TeacherId,CorpID)

values

('Лабаратория',1,1,1,201),

('Лабаратория',2,1,1,202),

('Лабаратория',3,1,1,203),

('Лабаратория',4,1,1,204),

('Лабаратория',5,1,1,205),

('Лабаратория',6,1,1,206),

('Лабаратория',7,1,1,207),

('Лабаратория',8,1,1,208),

('Лабаратория',9,1,1,209),

('Лабаратория',10,1,1,2010);

insert into Corps(CorpNumber,NumberOfAudiences)

values

('1A',200),

('2A',200),

('3A',200),

('4A',200),

('5A',200),

('6A',200),

('7A',200),

('8A',200),

('9A',200),

('10A',200);

insert into Teacher(Fname,Mname,Lname,Phone,Birthday)

values

('Августин','Петров','Михайлович','+79996132441',CONVERT(datetime,'02.01.2001')),

('Августина','Петрова','Михайловна','+79996132441',CONVERT(datetime,'02.01.2001')),

('Юлий','Гай','Цезарь','+79996132441',CONVERT(datetime,'02.01.2001')),

('Исаак','Ньютон','СЭР','+79996132441',CONVERT(datetime,'02.01.2001')),

('Будда ','Шакьямуни','Буддович','+79996132441',CONVERT(datetime,'02.01.2001')),

('Цай ','Лунь','Чай','+79996132441',CONVERT(datetime,'02.01.2001')),

('Галилей','Галилео','Галилеович','+79996132441',CONVERT(datetime,'02.01.2001')),

('Тестер1','Тестер1','тестер1','+79996132441',CONVERT(datetime,'02.01.2001')),

('Тестер2','Тестер2','Тестер2','+79996132441',CONVERT(datetime,'02.01.2001')),

('Тестер3','Тестер3','Тестер3','+79996132441',CONVERT(datetime,'02.01.2001')),

('Августин','Петров','Михайлович','+79996132441',CONVERT(datetime,'02.01.2001'));

# приложение В

Создание дополнительных объектов

use Gerasimov\_course\_work

go

create procedure TableInfoOnDate

@idTable int,

@date date

as

begin

select

r.reservation\_number,

r.client\_fname,

r.client\_lname,

r.client\_phone,

r.date\_of\_reservation,

r.time\_of\_reservation

from

Table\_Reservation tr

join Reservation r on tr.reservation\_id = r.id\_reservation

join [Table] t on tr.table\_id = t.id\_table

where

t.table\_number = @idTable

and r.date\_of\_reservation = @date

end

go

create function AmountReservedTableOnDate

(

@date date

)

returns int

as

begin

declare @response int

select

@response = count(\*)

from

Reservation r

where

r.date\_of\_reservation = @date

return @response

end

go

create procedure CallAmounFunction

@date date

as

begin

select

dbo.AmountReservedTableOnDate(@date)

end

go

create trigger Room\_D

on [Table]

instead of delete

as

delete from Table\_Reservation where table\_id=(select id\_table from deleted)

delete from [Table] where id\_table=(select id\_table from deleted)

go

create trigger Reservation\_D

on Reservation

instead of delete

as

delete from Table\_Reservation where reservation\_id=(select id\_reservation from deleted)

delete from Reservation where id\_reservation=(select id\_reservation from deleted)

go

create trigger Color\_D

on Color

instead of delete

as

delete from [Table] where color\_id=(select id\_color from deleted)

delete from Color where id\_color=(select id\_color from deleted)

go

create trigger Material\_D

on Material

instead of delete

as

delete from [Table] where material\_id=(select id\_material from deleted)

delete from Material where id\_material=(select id\_material from deleted)

go

create trigger Form\_Factor\_D

on Form\_Factor

instead of delete

as

delete from [Table] where form\_id=(select id\_form\_factor from deleted)

delete from Form\_Factor where id\_form\_factor=(select id\_form\_factor from deleted)

go

Создание ролей и пользователей

use Gerasimov\_course\_work

go

create login GNS\_first

with password = '123';

create user GNS\_first

for login GNS\_first

create role Manager authorization GNS\_first

create login GNS\_second

with password = '123';

create user GNS\_second

for login GNS\_second

create role Hostes authorization GNS\_second

deny delete

to GNS\_second;

# приложение г

Код основной формы

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

CenterToScreen();

this.DataGridViewTablesInitialize();

}

private void DataGridViewTablesInitialize()

{

dataGridViewTables.DataSource = DBOjbects.Entities.Table.ToList();

dataGridViewTables.Columns["table\_number"].HeaderText = "№ столика";

dataGridViewTables.Columns["number\_of\_seats"].HeaderText = "Количество мест";

dataGridViewTables.Columns["Color"].HeaderText = "Цвет";

dataGridViewTables.Columns["Material"].HeaderText = "Материал";

dataGridViewTables.Columns["Form\_Factor"].HeaderText = "Форма";

dataGridViewTables.Columns["Table\_Reservation"].Visible = false;

dataGridViewTables.Columns["id\_table"].Visible = false;

dataGridViewTables.Columns["color\_id"].Visible = false;

dataGridViewTables.Columns["material\_id"].Visible = false;

dataGridViewTables.Columns["form\_id"].Visible = false;

dataGridViewTables.Columns["img"].Visible = false;

}

private void dataGridViewTables\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

Table table = (Table)this.dataGridViewTables.Rows[e.RowIndex].DataBoundItem;

SecondaryForm form = new SecondaryForm(table);

form.ShowDialog();

this.DataGridViewTablesInitialize();

}

private void textBoxSearch\_TextChanged(object sender, EventArgs e)

{

dataGridViewTables.DataSource = DBOjbects.Entities.Table

.Where(t => t.table\_number.ToString().Contains(textBoxSearch.Text)).ToList();

}

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

{

try

{

int tableNum = int.Parse(textBoxTableNum.Text);

DateTime date = DateTime.Parse(textBoxDate.Text);

TableInfoOnDateForm form = new TableInfoOnDateForm(tableNum, date);

form.ShowDialog();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void textBoxTableNum\_KeyPress(object sender, KeyPressEventArgs e)

{

char temp = e.KeyChar;

if (!Char.IsDigit(temp) && temp != 8)

e.Handled = true;

}

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

{

try

{

DateTime date = DateTime.Parse(textBoxDateForAmount.Text);

MessageBox.Show(DBOjbects.Entities.CallAmounFunction(date).ToList()[0].Value.ToString());

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

# приложение д

Код формы с подробной информацией о столике

public partial class SecondaryForm : Form

{

private Table Table { get; }

public SecondaryForm(Table table)

{

InitializeComponent();

this.Table = table;

this.Fill();

}

private void Fill()

{

comboBoxColor.DataSource = DBOjbects.Entities.Color.Select(Col => Col.color\_name).ToList();

comboBoxColor.SelectedItem = Table.Color.color\_name;

comboBoxFormFactor.DataSource = DBOjbects.Entities.Form\_Factor.Select(f => f.form\_name).ToList();

comboBoxFormFactor.SelectedItem = Table.Form\_Factor.form\_name;

comboBoxMaterial.DataSource = DBOjbects.Entities.Material.Select(m => m.material\_name).ToList();

comboBoxMaterial.SelectedItem = Table.Material.material\_name;

textBoxTableNum.Text = Table.table\_number == 0 ?

Convert.ToString(DBOjbects.Entities.Table.ToList().Last().table\_number + 1) :

Table.table\_number.ToString();

textBoxNumOfSeats.Text = Table.number\_of\_seats.ToString();

dataGridViewReservations.DataSource = GetReservations();

dataGridViewReservations.Columns["id\_reservation"].Visible = false;

dataGridViewReservations.Columns["Table\_Reservation"].Visible = false;

dataGridViewReservations.Columns["reservation\_number"].HeaderText = "№ брони";

dataGridViewReservations.Columns["client\_fname"].HeaderText = "Имя клиента";

dataGridViewReservations.Columns["client\_lname"].HeaderText = "Фамилия клиента";

dataGridViewReservations.Columns["client\_phone"].HeaderText = "Телефон";

dataGridViewReservations.Columns["time\_of\_reservation"].HeaderText = "Время";

dataGridViewReservations.Columns["date\_of\_reservation"].HeaderText = "Дата";

using (var ms = new MemoryStream(Table.img))

{

pictureBox.Image = Image.FromStream(ms);

}

pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;

}

private List<Reservation> GetReservations()

{

List<Reservation> reservations = new List<Reservation>();

foreach(Table\_Reservation tr in Table.Table\_Reservation)

{

reservations.Add(DBOjbects.Entities.Reservation.FirstOrDefault

(res => res.id\_reservation == tr.reservation\_id));

}

return reservations;

}

private void dataGridViewReservations\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

Reservation reservation = (Reservation)this.dataGridViewReservations.Rows[e.RowIndex].DataBoundItem;

ReservationForm form = new ReservationForm(reservation, this.Table.id\_table);

form.ShowDialog();

this.Fill();

}

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

{

Reservation reservation = new Reservation();

ReservationForm form = new ReservationForm(reservation, this.Table.id\_table);

form.ShowDialog();

this.Fill();

}

private void textBoxTableNum\_KeyPress(object sender, KeyPressEventArgs e)

{

char temp = e.KeyChar;

if (!Char.IsDigit(temp) && temp != 8)

e.Handled = true;

}

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

{

try

{

this.Table.table\_number = int.Parse(textBoxTableNum.Text);

this.Table.number\_of\_seats = int.Parse(textBoxNumOfSeats.Text);

this.Table.material\_id = DBOjbects.Entities.Material.FirstOrDefault(m =>

m.material\_name == comboBoxMaterial.SelectedItem).id\_material;

this.Table.color\_id = DBOjbects.Entities.Color.FirstOrDefault(c =>

c.color\_name == comboBoxColor.SelectedItem).id\_color;

this.Table.form\_id = DBOjbects.Entities.Form\_Factor.FirstOrDefault(f =>

f.form\_name == comboBoxFormFactor.SelectedItem).id\_form\_factor;

DBOjbects.Entities.SaveChanges();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

# приложение е

Код формы резервирования

public partial class ReservationForm : Form

{

private Reservation Reservation { get; }

private int Table\_id { get; }

public ReservationForm(Reservation reservation, int table\_id)

{

InitializeComponent();

this.Reservation = reservation;

this.Table\_id = table\_id;

this.Fill();

}

private void Fill()

{

textBoxReservNum.Text = Reservation.reservation\_number == 0 ?

Convert.ToString(DBOjbects.Entities.Reservation.ToList().Last().reservation\_number + 1) :

Reservation.reservation\_number.ToString();

textBoxClientFname.Text = Reservation.client\_fname;

textBoxClientLname.Text = Reservation.client\_lname;

textBoxClientPhone.Text = Reservation.client\_phone;

textBoxReservTime.Text = Reservation.time\_of\_reservation.ToString("t");

textBoxReservDate.Text = Reservation.date\_of\_reservation.Date.ToString("d").Substring(3);

}

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

{

try

{

if (IsFieldsEmpty())

throw new Exception("Все поля обязательные");

if (textBoxClientPhone.Text.Length != 11)

throw new Exception("Неверный формат номера телефона");

this.Reservation.reservation\_number = Int16.Parse(textBoxReservNum.Text);

this.Reservation.client\_fname = textBoxClientFname.Text;

this.Reservation.client\_lname = textBoxClientLname.Text;

this.Reservation.client\_phone = textBoxClientPhone.Text;

this.Reservation.date\_of\_reservation = DateTime.Parse(textBoxReservDate.Text);

this.Reservation.time\_of\_reservation = TimeSpan.Parse(textBoxReservTime.Text);

if (IsDateTimeAvailable())

{

if (!DBOjbects.Entities.Reservation.ToArray().Contains(this.Reservation))

{

DBOjbects.Entities.Reservation.Add(this.Reservation);

Table\_Reservation table\_Reservation = new Table\_Reservation()

{

table\_id = this.Table\_id,

reservation\_id = this.Reservation.id\_reservation

};

DBOjbects.Entities.Table\_Reservation.Add(table\_Reservation);

}

DBOjbects.Entities.SaveChanges();

this.Close();

}

else

throw new Exception("Это время уже занято");

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

try

{

if (DBOjbects.Entities.Reservation.ToArray().Contains(this.Reservation))

{

if (DBOjbects.Entities.Table\_Reservation.Where(tr => tr.reservation\_id == this.Reservation.id\_reservation) != null)

{

DBOjbects.Entities.Table\_Reservation.RemoveRange(

DBOjbects.Entities.Table\_Reservation.Where(tr => tr.reservation\_id == this.Reservation.id\_reservation).ToList());

}

DBOjbects.Entities.Reservation.Remove(this.Reservation);

DBOjbects.Entities.SaveChanges();

}

this.Close();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void textBoxReservNum\_KeyPress(object sender, KeyPressEventArgs e)

{

char temp = e.KeyChar;

if (!Char.IsDigit(temp) && temp != 8)

e.Handled = true;

}

private bool IsFieldsEmpty()

{

if(textBoxReservNum.Text.Length==0

||textBoxClientFname.Text.Length==0

|| textBoxClientLname.Text.Length==0

|| textBoxClientPhone.Text.Length==0

|| textBoxReservTime.Text.Length==0

|| textBoxReservDate.Text.Length == 0)

{

return true;

}

return false;

}

private bool IsDateTimeAvailable()

{

DateTime date = this.Reservation.date\_of\_reservation;

TimeSpan time = this.Reservation.time\_of\_reservation;

List<Table\_Reservation> allReservations = DBOjbects.Entities.Table\_Reservation

.Where(tr => tr.table\_id == this.Table\_id).ToList();

foreach(Table\_Reservation tr in allReservations)

{

Reservation temp = DBOjbects.Entities.Reservation.FirstOrDefault(r =>

r.id\_reservation == tr.reservation\_id);

if (this.Reservation.id\_reservation == temp.id\_reservation)

return true;

if (temp.date\_of\_reservation == date &&

(time.Hours >= temp.time\_of\_reservation.Hours - 5

&& time.Hours <= temp.time\_of\_reservation.Hours + 5))

return false;

else if (temp.date\_of\_reservation == date &&

(time.Hours <= temp.time\_of\_reservation.Hours - 5

|| time.Hours >= temp.time\_of\_reservation.Hours + 5))

return true;

}

return true;

}

}

# приложение ж

Код формы для вывода результата хранимой процедуры

public partial class TableInfoOnDateForm : Form

{

private int tableNum { get; }

private DateTime date { get; }

public TableInfoOnDateForm(int TableNum, DateTime date)

{

InitializeComponent();

this.tableNum = TableNum;

this.date = date;

DataGridViewInfoInitialize();

}

private void DataGridViewInfoInitialize()

{

dataGridViewInfo.DataSource = DBOjbects.Entities.TableInfoOnDate(this.tableNum, this.date).ToList();

dataGridViewInfo.Columns["reservation\_number"].HeaderText = "№ брони";

dataGridViewInfo.Columns["client\_fname"].HeaderText = "Имя клиента";

dataGridViewInfo.Columns["client\_lname"].HeaderText = "Фамилия клиента";

dataGridViewInfo.Columns["client\_phone"].HeaderText = "Телефон";

dataGridViewInfo.Columns["date\_of\_reservation"].HeaderText = "Дата";

dataGridViewInfo.Columns["time\_of\_reservation"].HeaderText = "Время";

}

}