

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: «Заказ столиков в ресторане: разработка и администрирование
базы данных, разработка клиентского приложения»

Выполнил:

ст. гр. ПКсп-116

Герасимов Н.С.

Приняли:

Павлова О.Н.

Куприянов А.А.

Владимир 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ.....	4
1.1 ER-ДИАГРАММА	4
2 ПОДГОТОВКА БАЗЫ ДАННЫХ.....	6
2.1 СЛОВАРЬ ДАННЫХ	6
2.2 ПРОЦЕДУРЫ И ФУНКЦИИ.....	8
3 РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ	10
4 ИМПОРТ И ЭКСПОРТ ДАННЫХ.....	11
4.1 ИМПОРТ ДАННЫХ ИЗ MS EXCEL	11
4.2 ИМПОРТ ДАННЫХ С ПОМОЩЬЮ BULK INSERT	13
4.3 КОМАНДНЫЙ ФАЙЛ.....	13
5 УПРАВЛЕНИЕ ДОСТУПОМ	16
6 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ.....	17
6.1 СТРУКТУРА ПРИЛОЖЕНИЯ	17
6.2 ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС	18
ЗАКЛЮЧЕНИЕ	22
ПРИЛОЖЕНИЕ А.....	23
ПРИЛОЖЕНИЕ Б	25
ПРИЛОЖЕНИЕ В	27
ПРИЛОЖЕНИЕ Г	30
ПРИЛОЖЕНИЕ Д.....	32
ПРИЛОЖЕНИЕ Е	35
ПРИЛОЖЕНИЕ Ж	38

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.

1 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

1.1 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.

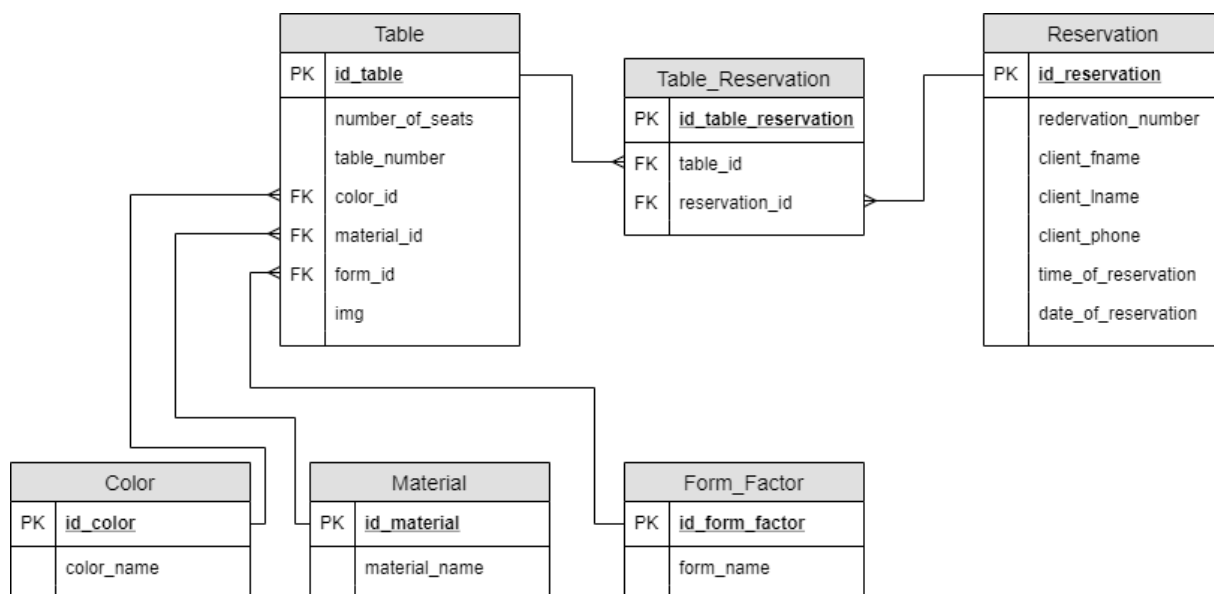


Рисунок 1.1 – ER-диаграмма

2 ПОДГОТОВКА БАЗЫ ДАННЫХ

2.1 Словарь данных

Таблица 1 – таблица сущности Table

Table			
Ключ	Поле	Обязательное	Примечание
Первичный	id_table	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

Таблица 2 – таблица сущности Reservation

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

Таблица 3 – таблица ассоциативной сущности Table_Reservation

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

Таблица 4 – таблица сущности Color

Color			
Ключ	Поле	Обязательное	Примечание
Первичный	id_color	Да	Идентификационный номер цвета
	color_name	Да	Название цвета

Таблица 5 – таблица сущности Material

Material			
Ключ	Поле	Обязательное	Примечание
Первичный	id_material	Да	Идентификационный номер материала
	material_name	Да	Название материала

Таблица 6 – таблица сущности Form_Factor

Form_Factor			
Ключ	Поле	Обязательное	Примечание

Первичный	id_form_factor	Да	Идентификационный номер формы
	form_name	Да	Название формы

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

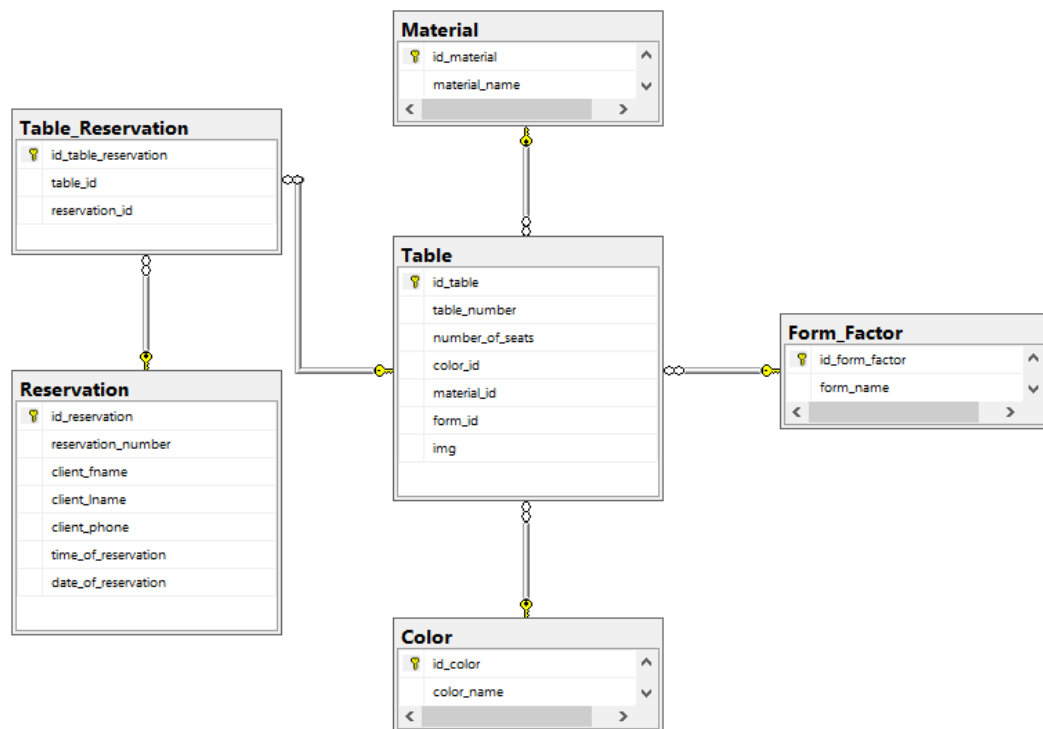


Рисунок 2.1 – Диаграмма базы данных

Код создания таблиц можно посмотреть в приложении А.

2.2 Процедуры и функции

Всего в базе данных реализованы: 2 хранимых процедуры и 1 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 2 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.

3 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'
```

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
```

Код командного файла:

```
sqlcmd -S WIN-DS5780BD4NC -i SQLQuery_restore.sql
```

4 ИМПОРТ И ЭКСПОРТ ДАННЫХ

4.1 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.

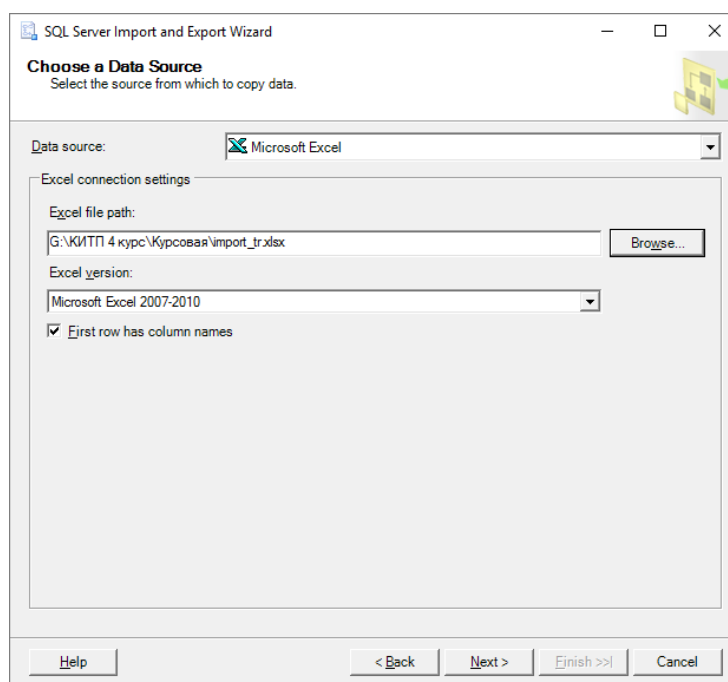


Рисунок 4.2 – Выбор источника импорта данных

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

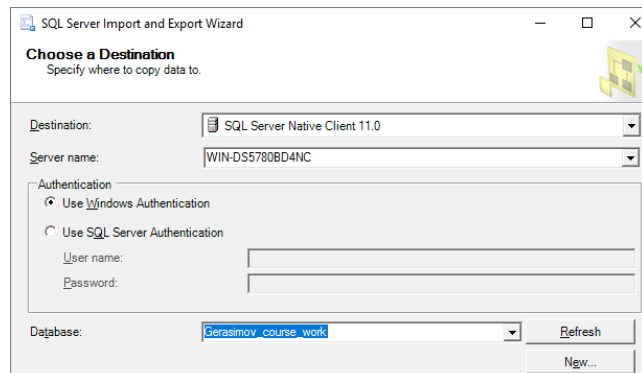


Рисунок 4.3 – Место назначения импорта данных

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

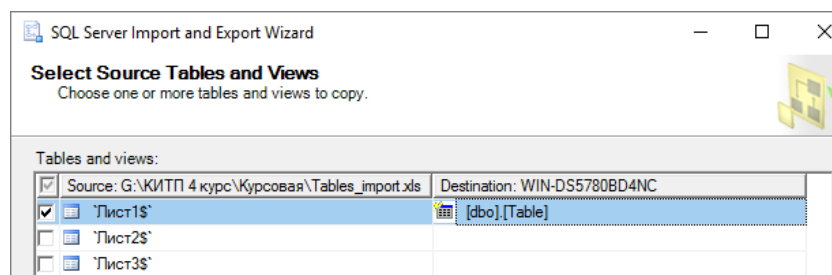


Рисунок 4.4 – Сопоставление страниц с таблицами

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

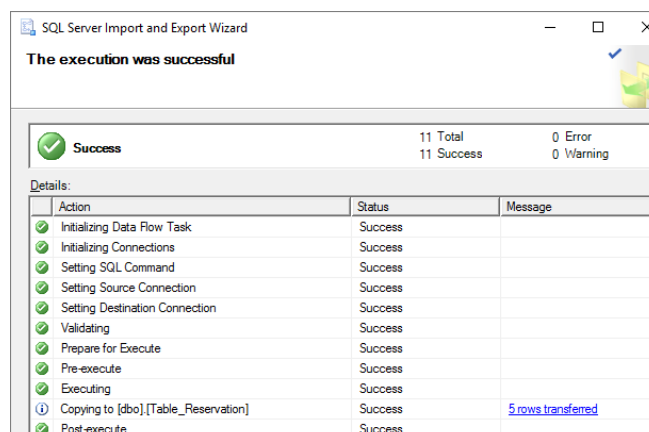


Рисунок 4.4 – Завершение импорта данных

4.2 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:

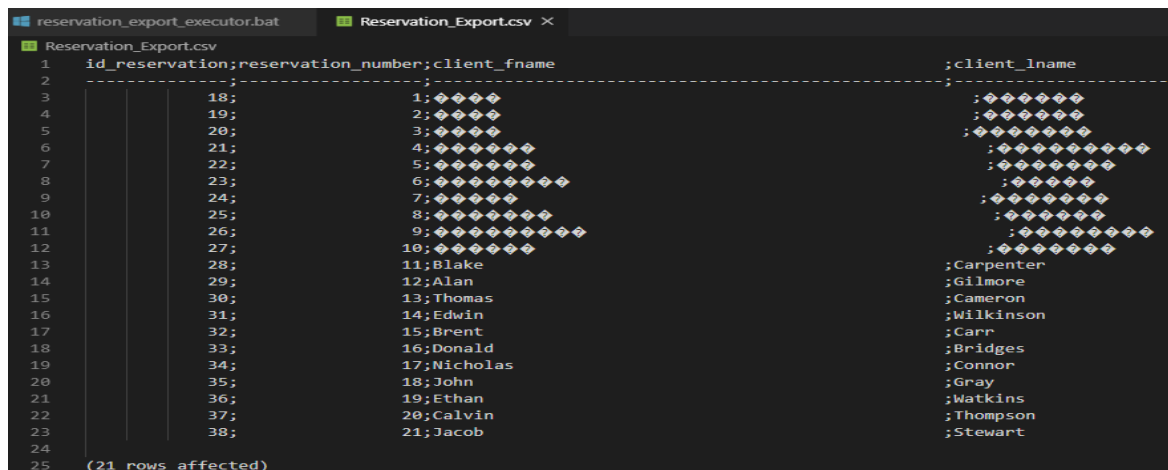


Рисунок 4.5 – Пример экспортированных данных

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:

```
G:\КИП 4 курс\Курсовая>select_executor.bat
G:\КИП 4 курс\Курсовая>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"
client_fname                client_lname                time_of_reservation         date_of_reservation
-----
Иван                        Иванов                      14:30:00.0000000          2019-12-01
Петр                        Петров                      10:00:00.0000000          2019-12-25
Семён                      Семенов                     21:15:00.0000000          2019-11-11
Никита                     Герасимов                   14:00:00.0000000          2019-11-11
Никита                     Герасимов                   14:00:00.0000000          2019-11-11
Владимир                   Ленин                       19:17:00.0000000          2019-12-12
Роман                      Сидоров                     12:30:00.0000000          2019-11-05
Евгений                    Пупкин                      15:45:00.0000000          2019-11-30
Владислав                  Маркелов                    13:00:00.0000000          2019-11-20
Сергей                     Сергеев                     09:30:00.0000000          2019-12-02
Blake                      Carpenter                   11:00:00.0000000          2019-11-16
Alan                      Gilmore                     12:00:00.0000000          2019-12-06
Thomas                    Cameron                     13:00:00.0000000          2019-12-05
Edwin                     Wilkinson                   14:00:00.0000000          2019-11-23
Brent                      Carr                        15:00:00.0000000          2019-12-13

(15 rows affected)
```

Рисунок 4.6 – Выполнение командного файла на вывод

5 УПРАВЛЕНИЕ ДОСТУПОМ

Для управления доступом в базе данных с помощью sql-скрипты были созданы 2 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.

6 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

6.1 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.

6.2 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.

Главная

Введите номер нужного стола:

	№ столика	Количество мест	Цвет	Форма	Материал
▶	1	4	Белый	Квадрат	Дерево
	2	4	Фиолетовый	Квадрат	Металл
	3	6	Желтый	Прямоугольник	Пластик
	4	3	Красный	Круг	Металл
	5	5	Зеленый	Овал	Дерево
	6	4	Черный	Квадрат	Дерево
	7	3	Белый	Прямоугольник	Пластик
	8	3	Оранжевый	Овал	Металл

Поиск информации о столике по дате

Введите номер столика Введите дату (формат: "dd.mm.yy")

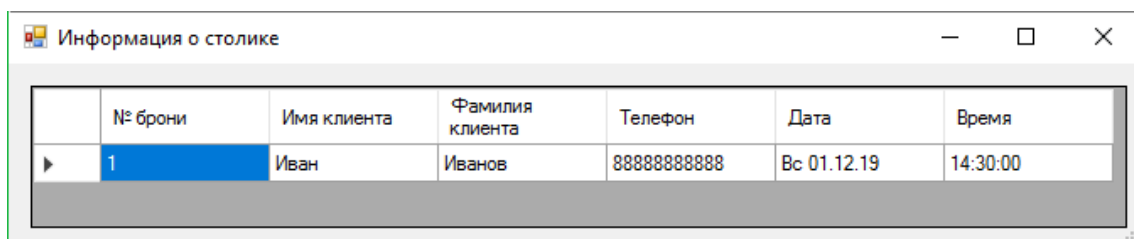
Сколько столиков забронировано n-ого числа

Введите дату (формат: "dd.mm.yy")

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

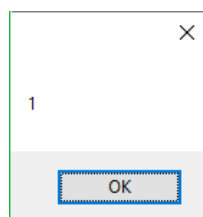
«Посмотреть». Для примера выведем информацию по столику №1 за 1.12.2019. Перед нами откроется следующая форма:



	№ брони	Имя клиента	Фамилия клиента	Телефон	Дата	Время
▶	1	Иван	Иванов	88888888888	Вс 01.12.19	14:30:00

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:

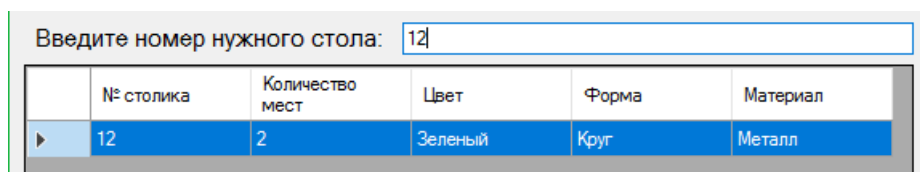


1

OK

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.



Введите номер нужного стола: 12

	№ столика	Количество мест	Цвет	Форма	Материал
▶	12	2	Зеленый	Круг	Металл

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.

Информация о столике

Номер столика: Количество мест:

Цвет: Форма:

Материал:

	№ брони	Имя клиента	Фамилия клиента	Телефон	Время	Дата
▶	1	Иван	Иванов	8888888888	14:30:00	Вс 01.12.19

Рисунок 6.3 – Подробная информация по столику

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

Бронирование

Номер бронирования:

Имя клиента: Фамилия клиента:

Номер телефона клиента:

Время бронирования: Дата бронирования:

Рисунок 6.4 – Бронирование столика

	№ брони	Имя клиента	Фамилия клиента	Телефон	Время	Дата
▶	1	Иван	Иванов	8888888888	14:30:00	Вс 01.12.19
	22	Никита	Герасимов	89612533621	17:00:00	Чт 14.11.19

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.

	№ брони	Имя клиента	Фамилия клиента	Телефон	Время	Дата
▶	1	Иван	Иванов	888888888888	14:30:00	Вс 01.12.19

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.

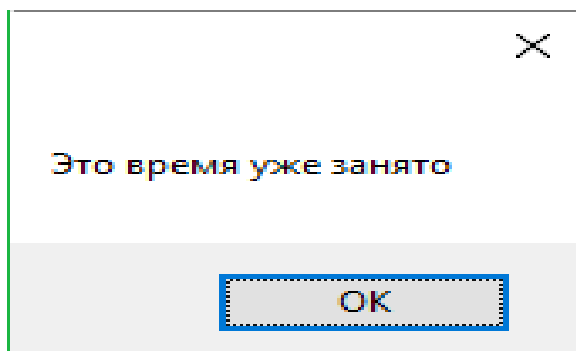


Рисунок 6.7 – Ошибка бронирования

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

```
use Gerasimov_course_work
```

```
insert into Color (color_name)
values
```

```
    ('Красный'),
    ('Оранжевый'),
    ('Желтый'),
    ('Зеленый'),
    ('Голубой'),
    ('Синий'),
    ('Фиолетовый'),
    ('Белый'),
    ('Черный'),
    ('Коричневый')
```

```
insert into Material (material_name)
values
```

```
    ('Дерево'),
    ('Металл'),
    ('Пластик')
```

```
insert into Form_Factor (form_name)
values
```

```
    ('Круг'),
    ('Квадрат'),
    ('Овал'),
    ('Прямоугольник')
```

```
insert into [Table] (table_number, number_of_seats, color_id,
material_id, form_id)
values
```

```
    (1, 3, 8, 1, 1),
    (2, 4, 7, 2, 2),
    (3, 6, 3, 3, 4),
    (4, 3, 1, 2, 1),
    (5, 5, 4, 1, 3),
    (6, 4, 9, 1, 2),
    (7, 3, 8, 3, 4),
    (8, 3, 2, 2, 3),
    (9, 2, 5, 1, 1),
    (10, 4, 6, 1, 1)
```

```
insert into Reservation (reservation_number, client_fname,
client_lname, client_phone, time_of_reservation, date_of_reservation)
values
```

```
    (1, 'Иван', 'Иванов', '888888888888', '13:30', '12-1-2019'),
    (2, 'Петр', 'Петров', '999999999999', '10:00', '12-25-2019'),
    (3, 'Семён', 'Семенов', '777777777777', '21:15', '11-11-2019'),
```

```

(4, 'Никита', 'Герасимов', '666666666666', '18:00', '11-13-2019'),
(5, 'Михаил', 'Зубенко', '555555555555', '14:48', '11-19-2019'),
(6, 'Владимир', 'Ленин', '444444444444', '19:17', '12-12-2019'),
(7, 'Роман', 'Сидоров', '333333333333', '12:30', '11-5-2019'),
(8, 'Евгений', 'Пупкин', '222222222222', '15:45', '11-30-2019'),
(9, 'Владислав', 'Маркелов', '111111111111', '13:00', '11-20-
2019'),
(10, 'Сергей', 'Сергеев', '000000000000', '9:30', '12-2-2019')

```

```

insert into Table_Reservation(table_id, reservation_id)
values

```

```

(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10)

```

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 =
        DBOjbecks.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 = DBOjbecks.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 =
        DBObjbects.Entities.Color.Select(Col => Col.color_name).ToList();
        comboBoxColor.SelectedItem = Table.Color.color_name;
        comboBoxFormFactor.DataSource =
        DBObjbects.Entities.Form_Factor.Select(f => f.form_name).ToList();
        comboBoxFormFactor.SelectedItem =
        Table.Form_Factor.form_name;
        comboBoxMaterial.DataSource =
        DBObjbects.Entities.Material.Select(m => m.material_name).ToList();
        comboBoxMaterial.SelectedItem =
        Table.Material.material_name;
        textBoxTableNum.Text = Table.table_number == 0 ?
        Convert.ToString(DBObjbects.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,
DataGridViewCellEventArgs e)
    {
        Reservation reservation =
        (Reservation)this.dataGridViewReservations.Rows[e.RowIndex].DataBoundIt
        em;
        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(DBOjbjects.Entities.Reservation.ToList().Last().reserva
tion_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";
    }
}
```