*Институт Транспорта и связи*

**Курсовая работа**

По дисциплине

«Базы и банки данных»

Тема: «**Разработка информационной системы для выбранной предметной области**»

Студент: Кузнецов Никита

Группа: 4701BD

Рига

2019 г.

Оглавление

[Задание на контрольную работу 3](#_Toc10316131)

[Анализ предметной области 4](#_Toc10316132)

[Краткое описание 4](#_Toc10316133)

[Пользователи информационной системы 4](#_Toc10316134)

[Бизнес процесс 4](#_Toc10316135)

[Разработка модели базы данных 5](#_Toc10316136)

[Реализация БД в СУБД 7](#_Toc10316137)

[Выбор базы данных и среды создания приложения 7](#_Toc10316138)

[Реализация таблиц 7](#_Toc10316139)

[Реализация запросов 9](#_Toc10316140)

[Разработка и создание приложения пользователя 12](#_Toc10316141)

[Разработка и создание форм 12](#_Toc10316142)

[Навигация по приложению 18](#_Toc10316143)

[Код программы 18](#_Toc10316144)

[Выводы 23](#_Toc10316145)

[Список используемых ресурсов 24](#_Toc10316146)

Задание на контрольную работу

На основе анализа, выбранной предметной области, спроектировать и реализовать базу данных, которая:

* включает не менее 5 сущностей
* имеет удобный пользовательский интерфейс
* обеспечивающий ввод и корректировку данных в таблицы БД
* получение с помощью запросов необходимой информации
* вывод документов на печать в виде отчетов

Процесс разработки информационной системы должен состоять из следующих шагов:

* Анализ предметной области
* Определение требований и функций системы
* Разработка модели базы данных
* Создание таблиц и схемы базы данны
* Выбор СУБД
* Разработка и создание форм для заполнения таблиц базы данных
* Разработка и создание запросов
* Разработка и создание приложения пользователя
* Автоматизация приложения
* Справочная система приложения

****Анализ предметной области****

Краткое описание

Выбранная предметная область – отель. Отель предлагает услуги аренды апартаментов для группы людей на определенный период времени.

Для администрирования этого процесса необходимо иметь расширяемый список апартаментов, календарь бронирований, базу клиентов, изменяемые таблицы с расценками для разных классов апартаментов.

Существенной необходимостью является отображение данных о загруженности отеля в текущий момент.

Двери апартаментов оборудованы бесконтактными электронными замками.

Во время бронировании необходимо производить привязку ключа к конкретному апартаменту и клиенту.

При потере бесконтактного ключа нужно иметь возможность деактивировать старый и выдать новый ключ.

Так же присутствует нужда в мониторинге состояния апартаментов, какие из них открыты, а какие закрыты, для своевременной и оперативной уборки.

****Пользователи информационной системы****

**Данная система разрабатывается для отеля средних размеров. Единственным пользователем системы будет администратор или его заместитель. Исходя из этих условий нужда в дополнительной безопасности, и ограничении доступа отсутствует, так как система будет однопользовательской.**

****Бизнес процесс****

***Add clients***

**Первым делом администратору системы необходимо делом добавить всех новых постояльцев в список клиентов, если они там не записаны**

***Make booking***

**Оформление бронирования начинается с установления даты заезда и выезда, а также класса апартаментов. Далее производится автоматический поиск подходящих под критерии свободных апартаментов. Если таковые найдены, администратор вносит в список гостей, добавляет для каждого постояльца бесконтактный ключ. И по необходимости корректирует автоматически подсчитанную цену.**

***Bookings edit***

**Администратор может просмотреть список всех бронирований и при возникновении нужды, скорректировать информацию.**

***Key list***

**До того, как привязать какой-либо ключ к клиенту, его необходимо занести в общий список всех ключей. При утере ключа его нужно деактивировать.**

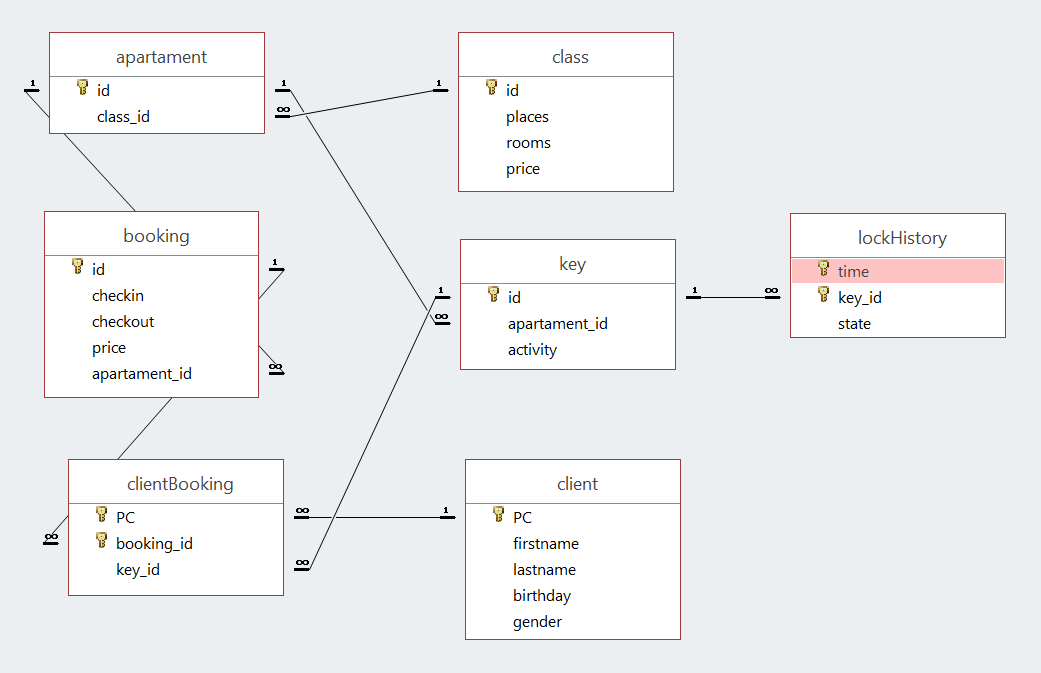
***Open/closed apartaments.***

**Для проведения уборки, есть возможность достать список со всеми закрытыми номерами. В этих апартаментах следует провести данную процедуру в первую очередь, чтобы минимально беспокоить постояльцев по их возвращению в номер.**

****Разработка модели базы данных****

**После анализа предметной области, мы можем выбрать сущности для нашей базы и создать ER-модель.**

* **Class**
* **Apartment**
* **Client**
* **Booking**
* **ClientBooking**
* **Key**
* **LockHistory**

****

**Рисунок 1. ER модель**

**Все сущности имеют ключи, большинство имеют простой ключ, несколько сущностей имеют составной ключ. Так же все сущности имеют множество атрибутов, некоторые из них являются внешними ключами.**

****Реализация БД в СУБД****

**Результирующая база данных имеет 7 таблиц, и 34 sql-запроса.**

****Выбор базы данных и среды создания приложения****

**Для реализации данного проекта я выбрал связку Microsoft Access 2016 + C# Windows Forms. На мой взгляд Access достаточно удобная система для создания структуры базы данных: создания таблиц, связывания, выставления ссылочной целостности. Access имеет различные мощные инструменты, которые позволяют выполнять некоторые задания с большой легкостью. Но от идеи создавать формы в Access я отказался, по той причине, что созданные в нем формы, требуют наличия проприетарного программного обеспечения для дальнейшей работы. Приложение написанное в среде .Net с использованием API Windows Form в конечном результате замечательно поддерживает работу с созданной в Access файлом базы данных так как для своей работы использует Microsoft Jet Database Engine (как и Access).**

****Реализация таблиц****

***Apartment***

* **Ключ:**
  + **id – Number**
* **Поля:**
* **class\_id – Short text**

***Booking***

* **Ключ:**
* **id – Autonumber**
* **Поля:**
* **checkin – DateTime**
* **checkout – DateTime**
* **price – Currency**
* **apartament\_id – Number**

***Class***

* **Ключ:**
* **id – shortext**
* **Поля:**
* **places – number**
* **rooms – number**
* **price – Currency**

***Client***

* **Ключ:**
* **PC– Short text**
* **Поля:**
* **firstname – Short text**
* **lastname – Short text**
* **birthday – DateTime**
* **gender – Boolean**

***ClientBooking***

* **Составной ключ:**
* **PC – Short text**
* **booking\_id - Number**
* **Поля:**
* **key\_id – Short text**

***Key***

* **Ключ:**
* **id – Short text**
* **Поля:**
* **apartament\_id – Number**
* **activity – Boolean**

***LockHistory***

* **Составной ключ:**
* **time – datetime**
* **key\_id – Short text**
* **Поля:**
* **state – boolean**

****Реализация запросов****

***Примеры:***

**Search Client**

**Запрос предназначен для поиска клиента в форме со всеми клиентами.**

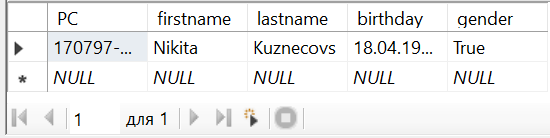
**SELECT PC, firstname, lastname, birthday, gender**

**FROM client**

**WHERE (PC LIKE '%' + ? + '%') OR**

**(firstname LIKE '%' + ? + '%') OR**

**(lastname LIKE '%' + ? + '%')**

****

**Рис.2 Результат запроса Search Client с параметром: *nik***

**Find Free Apartments**

**Данный запрос используется при осуществлении бронирования для поиска свободных апартаментов в определенный временной период.**

**SELECT apartament.id, apartament.class\_id**

**FROM (apartament LEFT OUTER JOIN**

**(SELECT DISTINCT apartament\_id**

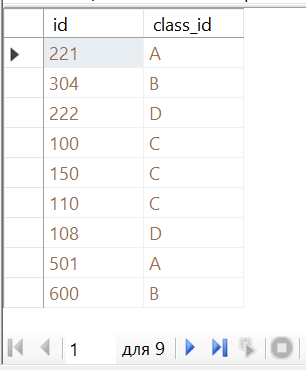
**FROM booking**

**WHERE (checkin BETWEEN ? AND ?) OR**

**(checkout BETWEEN ? AND ?) OR**

**(checkin < ?) AND (checkout > ?)) nf ON apartament.id = nf.apartament\_id)**

**WHERE (nf.apartament\_id IS NULL)**

****

**Рис 3. Результат выполнения запроса Find Free Apartments**

**Last used key for apartment**

**Поиск ключа, который последний взаимодействовал с замком номера.**

**SELECT TOP 1 key\_id**

**FROM lockHistory**

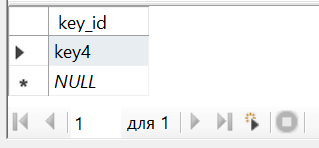
**WHERE (key\_id IN**

**(SELECT id AS key\_id**

**FROM [key]**

**WHERE (apartament\_id = ?)))**

**ORDER BY [time] DESC**

****

**Рис.4 Результат выполнения запроса Last used key for apartament.**

**Booking intersection count**

**Данный запрос используется для валидации во время создания и редактирования записи о бронировании, чтобы убедиться, что данный номер не занят в выбранный временной период.**

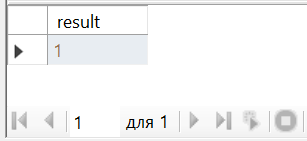
**SELECT COUNT(\*) AS result**

**FROM booking**

**WHERE (apartament\_id = ?) AND (checkin <= ?) AND (checkout >= ?) OR**

**(apartament\_id = ?) AND (checkin >= ?) AND (checkin < ?) OR**

**(apartament\_id = ?) AND (checkout > ?) AND (checkout <= ?)**

****

**Рис.5.Результат выполнения запроса Booking intersection count**

**(Данный номер занят в выбранный временной период, если валидация включена, пользователь получит уведомление )**

**Apartment state**

**Запрос проверяет открыт ли в данный момент номер или закрыт.**

**FROM lockHistory**

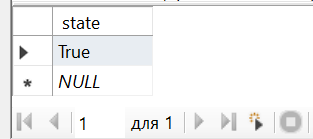
**WHERE (key\_id IN**

**(SELECT id AS key\_id**

**FROM [key]**

**WHERE (apartament\_id = ?)))**

**ORDER BY [time] DESC**

****

**Рис.6 Результат выполнения запроса Apartament State (Номер закрыт)**

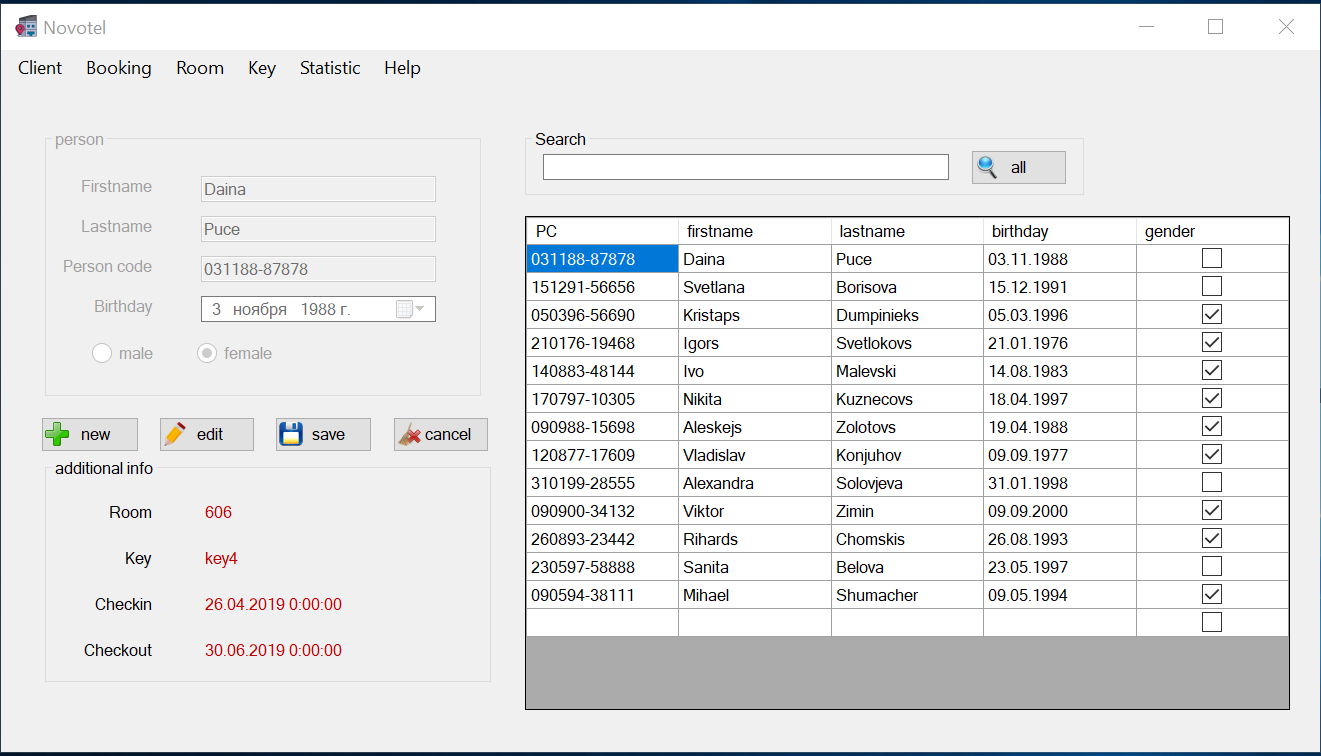
****Разработка и создание приложения пользователя****

****Разработка и создание форм****

**Client - > Client list**

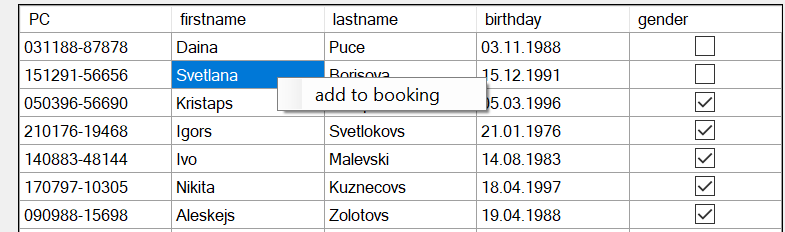
**Через эту форму осуществляется доступ к списку клиентов и поиск по нему. В список нужно вносить новых постояльцев, перед оформлением бронирования. Имеется возможность редактирования записей о уже существующих клиентах.**

**Окошко в нижнем левом углу отображает дополнительную информацию о выбранном клиенте – последний номер проживания, ключ и дату.**

****

**Рис.7. Форма Client list**

**Для добавления клиента в список нового бронирования, нужно нажать правой кнопкой мыши по строке в таблице и выбрать пункт add to booking.**

****

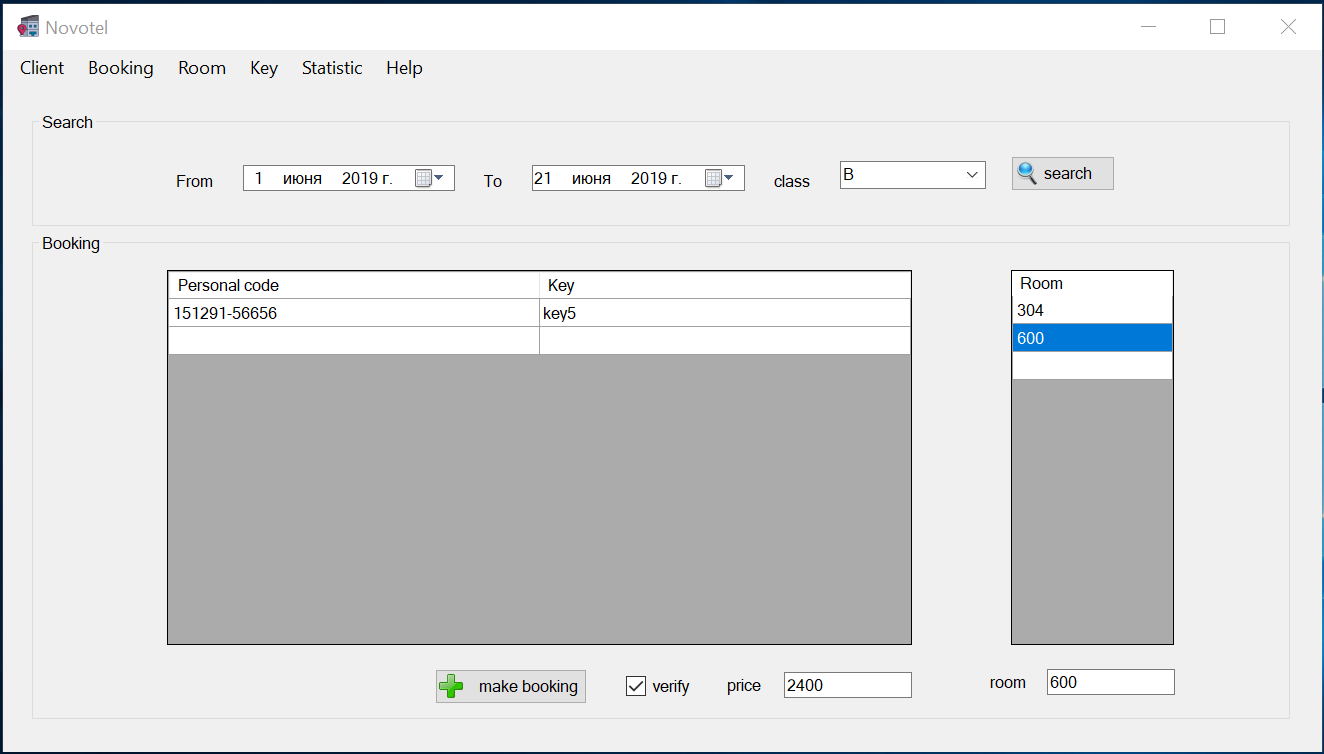
**Рис.8 Добавление клиента в новое бронирование**

**Booking -> Make booking**

**Данная форма одна из основных форм в приложении. Ее функция – создание нового бронирования. После выбора даты проживания и класса апартамента, производится поиск свободных апартаментов, подсчитывается цена. В таблице уже присутствуют персональные коды клиентов, выбранных пользователем в форме *client list.***

**Далее для каждого постояльца выбирается свободный ключ.**

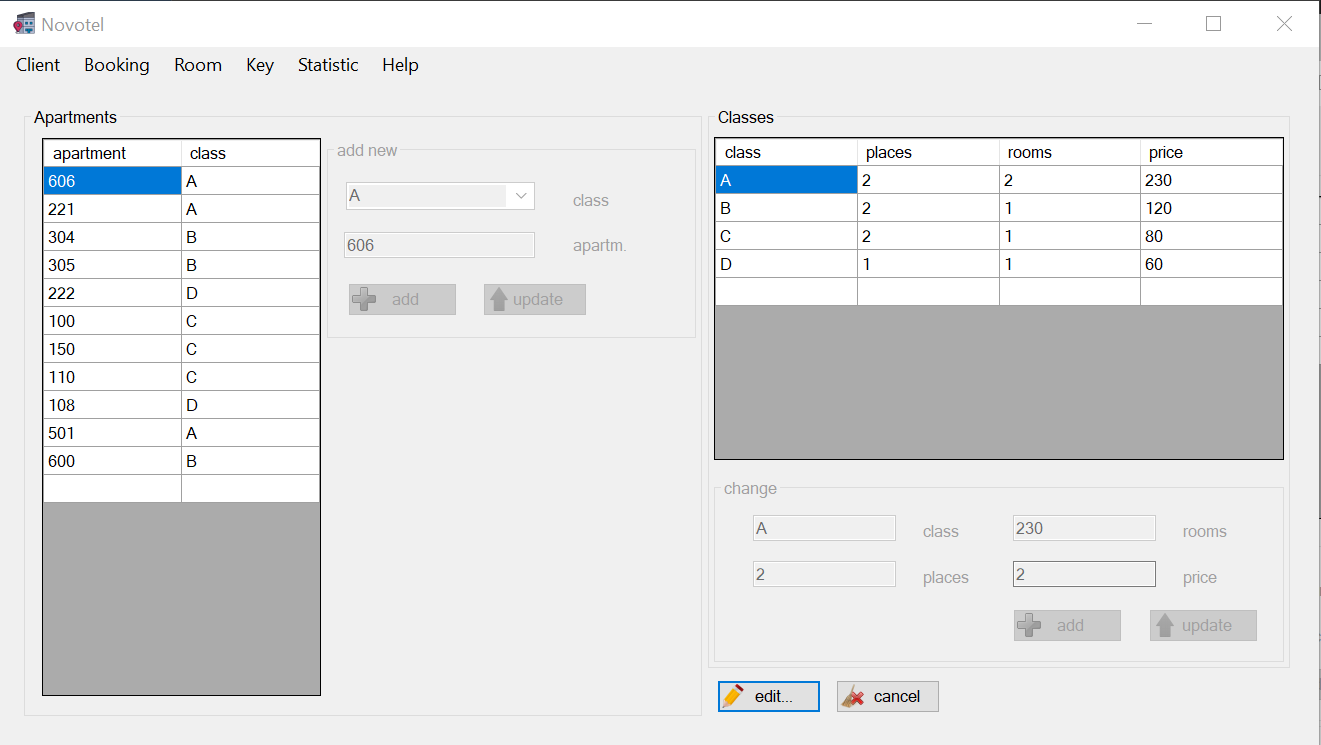
**При нажатии кнопки Make booking, если все данные были введены корректно, производится добавление бронирования в базу.**

****

**Рис. 9 Форма Make booking**

**Room -> Apartment list**

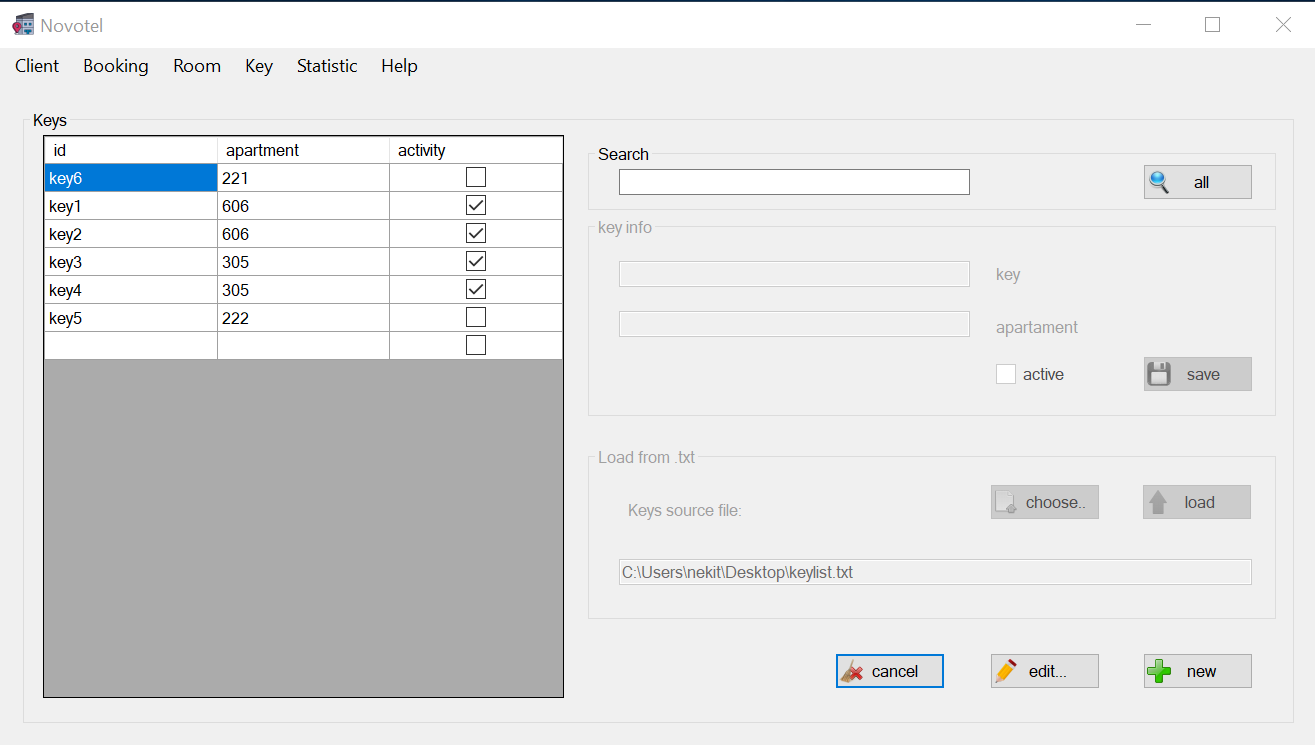
**В этой форме можно просматривать список всех апартаментов, и список всех их классов. А так же, добавлять новые записи в эти списки, и редактировать их.**

****

**Рис.10 Форма Apartment list**

**Key->Key list**

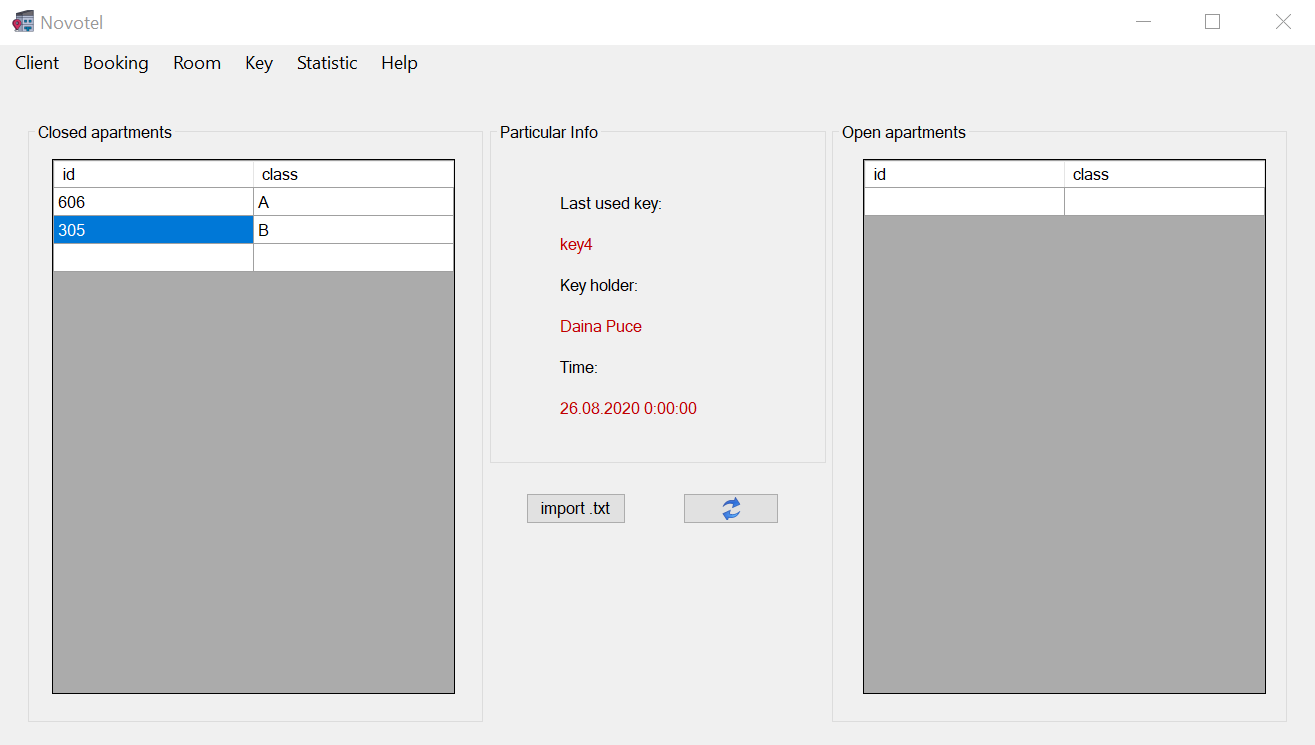
**Данная форма служит для просмотра, редактирования и добавления новых ключей. Имеется функция добавления за раз множества ключей из текстового файла.**

****

**Рис.11 Форма Key list**

**Room->Open/closed**

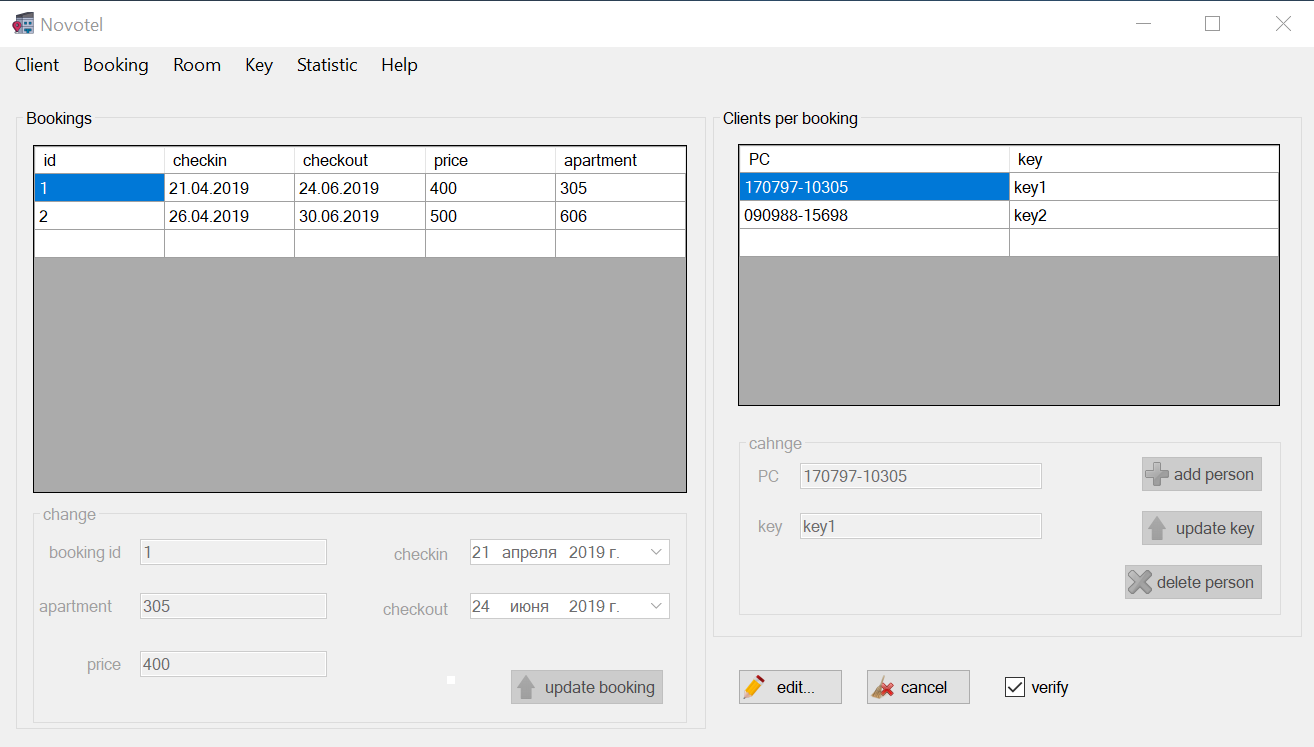
**Данная форма используется для просмотра открытых и закрытых апартаментов. Так же имеется функция импортирования этой информации в текстовый файл**

****

**Рис.12 Форма Open/closed**

**Booking->Booking list**

**Данная форма предоставляет возможность просмотра всех бронирований, и их редактирования. В левая таблица отображает список всех бронирований, в правой таблице все клиенты и ключи, связанные с этим бронированием.**

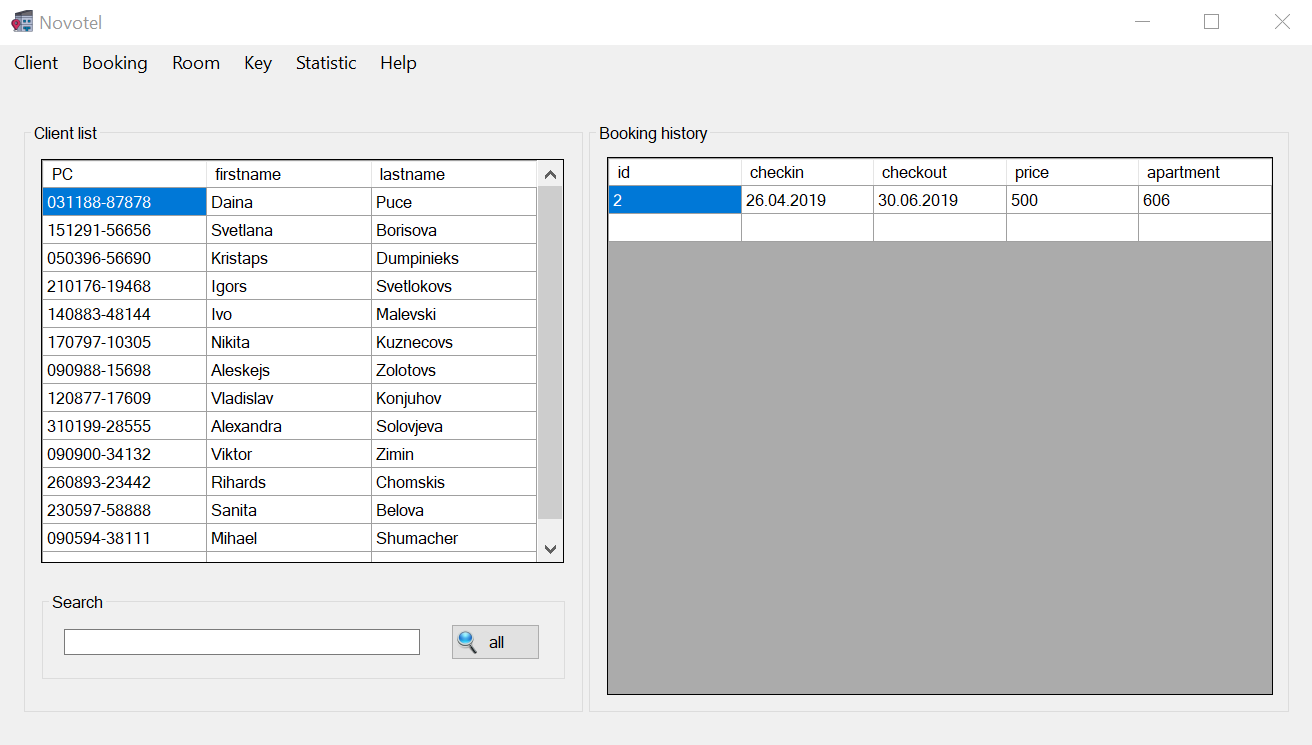
****

**Рис.13 Форма Booking list**

**Client->Client history**

**Эта форма нужна для просмотра истории бронирования отдельного клиента.**

**Имеется функция поиска клиента по персональному коду, имени или фамилии.**

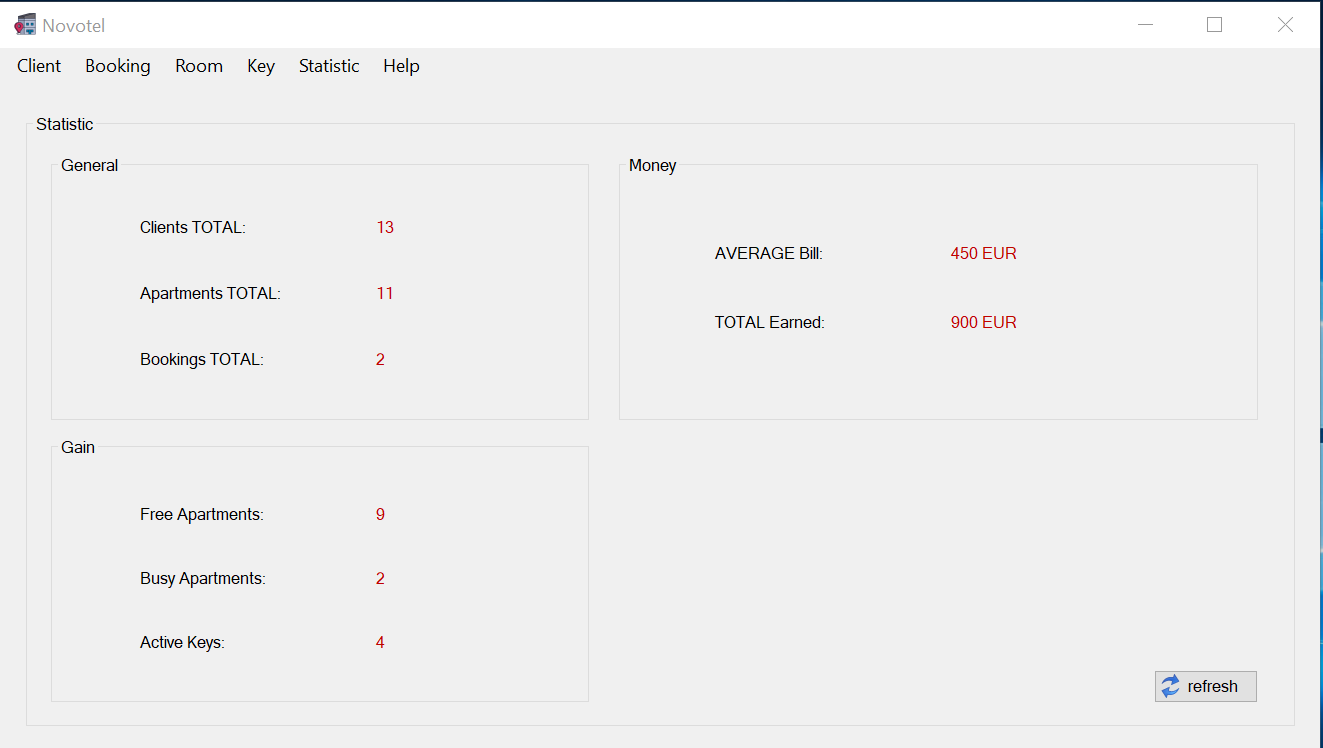
****

**Рис.14 Форма Client history**

**Statistic -> Numbers**

**На этой форме можно отслеживать некоторую статистику:**

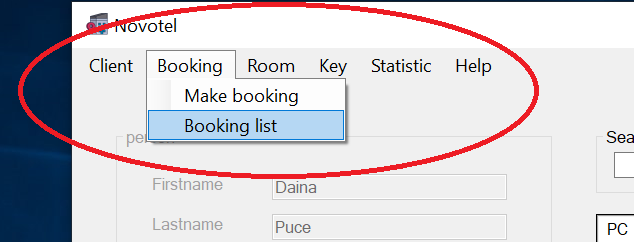
* **Общее количество клиентов в базе**
* **Количество апартаментов в отеле**
* **Число всех бронирований**
* **Количество свободных на данных момент апартаментов**
* **Количество занятых апартаментов**
* **Количество активных ключей (= числу постояльцев на данных момент)**
* **Средний чек за бронирование**
* **Общая сумма заработанных денег**

****

**Рис.15 Форма со статистикой**

****Навигация по приложению****

**Все формы связаны между собой навигационной панелью (Tool bar) в верхней части формы.**

****

**Рис.16 Навигационная панель приложения**

****Код программы****

**Весь код доступен для просмотра в репозитории:** <https://github.com/nekitkee/Novotel>

***Пример программного кода для формы Booking List***

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Novotel

{

public partial class BookingList : UserControl

{

public BookingList()

{

InitializeComponent();

}

//on cell ENTER - FILL client window with related data

private void dataGridViewBooking\_CellEnter(object sender, DataGridViewCellEventArgs e)

{

try

{

DataGridViewCellCollection CellCol = dataGridViewBooking.CurrentRow.Cells;

int bookingId = (int)CellCol[0].Value;

clientBookingTableAdapter.FillByBookingId(hotelDbDataSet.clientBooking, bookingId);

textBoxBookingId.Text = CellCol[0].Value.ToString();

textBoxApartament.Text = CellCol[4].Value.ToString();

textBoxPrice.Text = CellCol[3].Value.ToString();

dateTimePickerFrom.Value = (DateTime)CellCol[1].Value;

dateTimePickerTo.Value = (DateTime)CellCol[2].Value;

}catch(Exception ex) { MessageBox.Show(ex.Message); }

}

private void BookingList\_Load(object sender, EventArgs e)

{

try

{

//fill and bind

bookingTableAdapter.Fill(hotelDbDataSet.booking);

bookingBindingSource.DataSource = hotelDbDataSet.booking;

clientBookingBindingSource.DataSource = hotelDbDataSet.clientBooking;

}

catch (Exception ex) { MessageBox.Show(ex.Message); }

}

//on cell ENTER in client grid

private void dataGridViewClientBooking\_CellEnter(object sender, DataGridViewCellEventArgs e)

{

try

{

DataGridViewCellCollection CellCol = dataGridViewClientBooking.CurrentRow.Cells;

textBoxPC.Text = CellCol[0].Value.ToString();

textBoxKey.Text = CellCol[1].Value.ToString();

}catch(Exception ex) { MessageBox.Show(ex.Message); }

}

//EDIT

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

{

groupBoxChangeBookingClient.Enabled = true;

groupBoxChangeBooking.Enabled = true;

MessageBox.Show("THIS IS ADVANTAGE SETTINGS , PLEASE BE CAREFUL WHEN CHANGING DATA");

}

////CANCEL

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

{

groupBoxChangeBookingClient.Enabled = false;

groupBoxChangeBooking.Enabled = false;

}

//ADD CLIENT TO BOOKING

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

{

try

{

string PC = textBoxPC.Text;

int bookingId = int.Parse(textBoxBookingId.Text);

string key = textBoxKey.Text;

int apartament = int.Parse(textBoxApartament.Text);

int? clientExist = queriesTableAdapter1.PCexist(PC);

if (clientExist.Value == 0)

throw new Exception("client doesnt exist");

bool? keyExist = keyTableAdapter1.KeyActivity(key);

if (!keyExist.HasValue)

throw new Exception("key doesnt exist");

clientBookingTableAdapter.Insert(PC, bookingId, key);

keyTableAdapter1.UpdateQuery(apartament, true, key);

clientBookingTableAdapter.FillByBookingId(hotelDbDataSet.clientBooking, bookingId);

groupBoxChangeBookingClient.Enabled = false;

groupBoxChangeBooking.Enabled = false;

}

catch(Exception ex) { MessageBox.Show(ex.Message); }

}

//UPDATE CLIENT TO BOOKING

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

{

try

{

string PC = textBoxPC.Text;

int bookingId = int.Parse(textBoxBookingId.Text);

string key = textBoxKey.Text;

int apartament = int.Parse(textBoxApartament.Text);

int? clientExist = queriesTableAdapter1.PCexist(PC);

if (clientExist.Value == 0)

throw new Exception("client doesnt exist");

bool? keyExist = keyTableAdapter1.KeyActivity(key);

if (!keyExist.HasValue)

throw new Exception("key doesnt exist");

clientBookingTableAdapter.UpdateQuery(bookingId, key, PC);

keyTableAdapter1.UpdateQuery(apartament, true, key);

clientBookingTableAdapter.FillByBookingId(hotelDbDataSet.clientBooking, bookingId);

groupBoxChangeBookingClient.Enabled = false;

groupBoxChangeBooking.Enabled = false;

}

catch (Exception ex) { MessageBox.Show(ex.Message); }

}

//delete client from booking

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

{

try

{

string PC = textBoxPC.Text;

int bookingId = int.Parse(textBoxBookingId.Text);

string key = textBoxKey.Text;

int apartament = int.Parse(textBoxApartament.Text);

clientBookingTableAdapter.Delete(PC, bookingId, key);

keyTableAdapter1.UpdateQuery(apartament, false, key);

clientBookingTableAdapter.FillByBookingId(hotelDbDataSet.clientBooking, bookingId);

groupBoxChangeBookingClient.Enabled = false;

groupBoxChangeBooking.Enabled = false;

}

catch (Exception ex) { MessageBox.Show(ex.Message); }

}

//update booking info

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

{

try

{

int bookingId = int.Parse(textBoxBookingId.Text);

int apartament = int.Parse(textBoxApartament.Text);

DateTime from = dateTimePickerFrom.Value;

DateTime to = dateTimePickerTo.Value;

decimal price = decimal.Parse(textBoxPrice.Text);

// check dates SQL

int? intersec = queriesTableAdapter1.BookingIntersectionCount(bookingId, from, to, bookingId, from, to, bookingId, from, to);

if (intersec.Value != 0)

throw new Exception("Apartament is not free on this days");

bookingTableAdapter.UpdateQuery(from, to, price, apartament, bookingId);

bookingTableAdapter.Fill(hotelDbDataSet.booking);

groupBoxChangeBookingClient.Enabled = false;

groupBoxChangeBooking.Enabled = false;

}

catch (Exception ex) { MessageBox.Show(ex.Message); }

}

}

}

****Выводы****

**После выполнения курсовой работы я могу с честностью заявить, что я улучшил свои знания и понимание в области реляционных баз данных. Все необходимые требования, которые были поставлены перед началом выполнения были успешно реализованы.**

**Так же я закрепил свои навыки использования объектно-ориентированного языка программирования С# и Windows Forms API.**

**Во время выполнения работы я столкнулся с некоторыми сложностями: те запросы, которые отлично работали в конструкторе запросов внутри Visual Studio 2017 переставали работать при их вызове непосредственно из кода приложения. Для решения этой проблемы пришлось использовать дополнительный инструмент создания запросов LINQ. Раньше мне не доводилось с ним работать, но как оказалось его синтаксис сильно напоминает SQL.**

**Так же как замечание для самого себя хотел бы отметить, что если бы данное приложение предназначалось для реального использования, я бы обязательно занялся дальнейшей разработкой следующей версии в целях расширения функциональности, проработки дизайна и более лучшего пользовательского опыта.**

****Список используемых ресурсов****

1. Хранилище документации Майкрософт <https://docs.microsoft.com/ru-ru/>
2. Учебно-методическое пособие “Практикум по Access” Юнусов С. М 2005
3. The Largest Web Developer Site on the Internet <https://www.w3schools.com/sql/>