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 4](#_Toc24482119)

[1. Проектирование базы данных 5](#_Toc24482120)

[1.1 ER-диаграмма 5](#_Toc24482121)

[2. подготовка базы данных 6](#_Toc24482122)

[2.1 Словарь данных 6](#_Toc24482123)

[2.2 Процедуры и функции 7](#_Toc24482124)

[3 резервное копирование и восстановление 9](#_Toc24482125)

[3.1 Резервное копирование 9](#_Toc24482126)

[4 ИМПОрт и экспорт данных 14](#_Toc24482127)

[4.1 Импорт данных из MS Excel 14](#_Toc24482128)

[4.2 Экспорт данных в csv 17](#_Toc24482129)

[4.3 Импорт данных с помощью BULK INSERT 18](#_Toc24482130)

[5 Управление доступом 20](#_Toc24482131)

[6 разработка клиентского приложения 21](#_Toc24482132)

[6.1 Структура приложения 21](#_Toc24482133)

[6.2 Пользовательский интерфейс 21](#_Toc24482134)

[6.2.1 Главная форма 22](#_Toc24482135)

[6.2.2 Раздел администратора 26](#_Toc24482136)

[6.2.3 Форма о программе 28](#_Toc24482137)

[заключение 29](#_Toc24482138)

[Список использованных источников 30](#_Toc24482139)

[ПРИЛОЖЕНИЕ А 31](#_Toc24482140)

[Создание БД 31](#_Toc24482141)

[ПРИЛОЖЕНИЕ Б 32](#_Toc24482142)

[Заполнение таблиц 32](#_Toc24482143)

[ПРИЛОЖЕНИЕ В 33](#_Toc24482144)

[Создание дополнительных объектов 33](#_Toc24482145)

[Создание ролей и пользователей 37](#_Toc24482146)

[ПРИЛОЖЕНИЕ Г 38](#_Toc24482147)

[Код основной формы 38](#_Toc24482148)

[ПРИЛОЖЕНИЕ Д 41](#_Toc24482149)

[Код формы редактирования и добавления аудиторий 41](#_Toc24482150)

[ПРИЛОЖЕНИЕ Ж 45](#_Toc24482151)

[Код административной формы 45](#_Toc24482152)

[ПРИЛОЖЕНИЕ З 49](#_Toc24482153)

[Код формы проверки пароля 49](#_Toc24482154)

[ПРИЛОЖЕНИЕ К 50](#_Toc24482155)

[Код формы смены пароля 50](#_Toc24482156)

[ПРИЛОЖЕНИЕ Л 51](#_Toc24482157)

[Код формы «о программе» 51](#_Toc24482158)

# Введение

В данной работе стоит задача создать программный модуль «Аудиторный фонд многокорпусного здания», который может быть использован в различных многокорпусных учреждениях. Так же, для хранения данных, которыми будет оперировать программа, необходимо разработать базу данных.

Пользователь приложения может выполнять следующие действия:

* Просматривать информацию об аудитории.
* Добавлять и изменять аудиторию.
* Удалять аудиторию.

Для того чтобы добавить аудиторию, пользователь должен будет ввести: корпус, вместимость корпуса, аудиторию, вместимость аудитории, ФИО преподавателя ответственного за данную аудиторию, его личный номер, а также его дату рождения.

# Проектирование базы данных

## ER-диаграмма

База данных учета аудиторий в многокорпусном здании должна иметь следующий набор сущностей с, присущими им, атрибутами:

* Информация о корпусе: Номер корпуса и его вместимость.
* Информация о преподавателе: Фамилия, Имя, Отчество, номер телефона и дата рождения.
* Информация об аудитории: Номер аудитории и ее вместимость, так же, корпус где находится аудитория и ответственного за аудиторию.

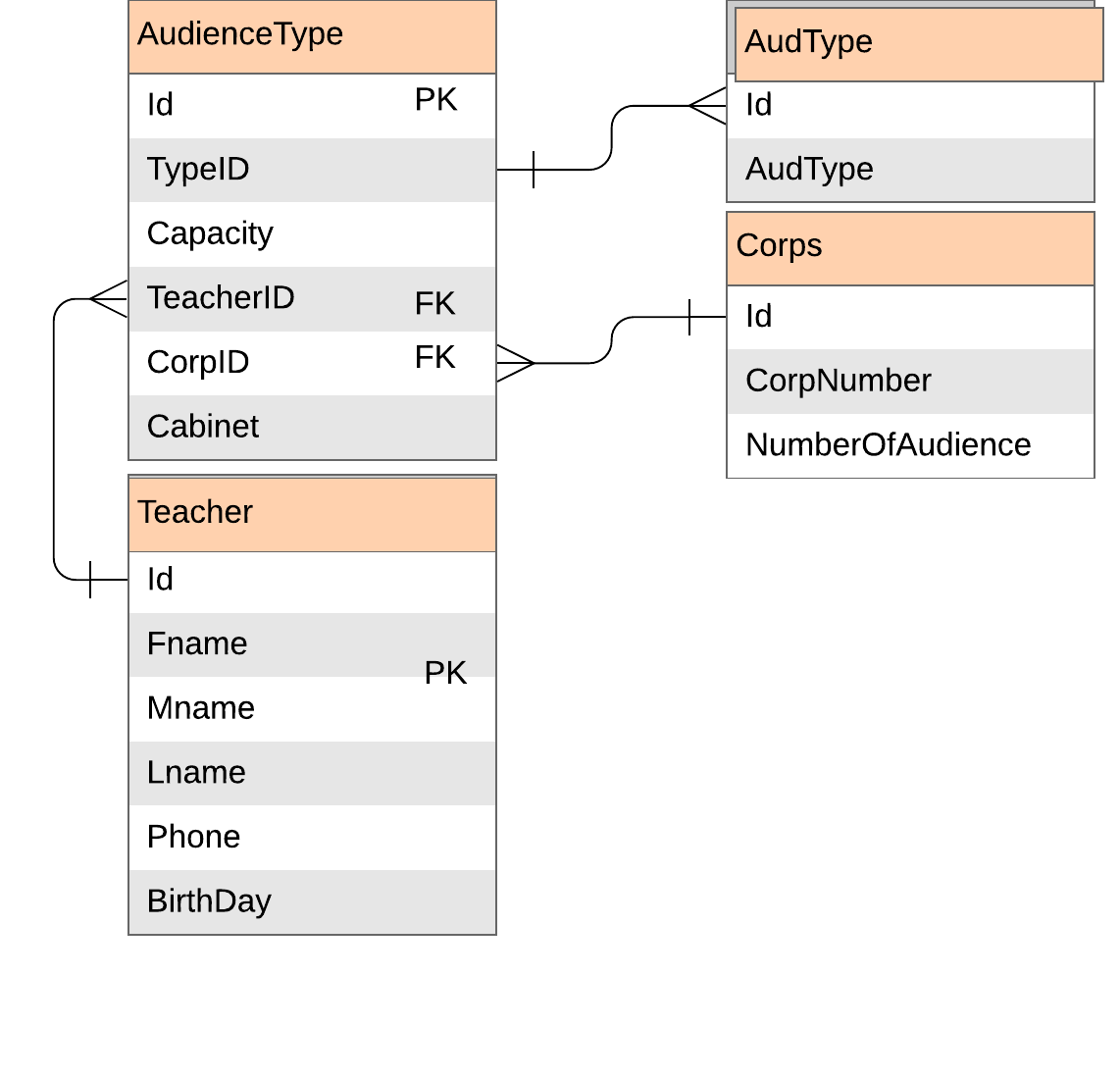


Рисунок 1.1 – ER-диаграмма

# подготовка базы данных

## Словарь данных

Таблица№1-Словарь данных для таблицы Corps

|  |  |  |  |
| --- | --- | --- | --- |
| Corps | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | Id | Да | Идентификационный номер |
|  | CorpNumber | Да | Номер аудитории |
|  | NumberOfAudiences | Да | Количество аудиторий |

Таблица№2-Словарь данных для таблицы Teacher

|  |  |  |  |
| --- | --- | --- | --- |
| Teacher | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | Id | Да | Идентификационный номер |
|  | Fname | Да | Имя преподавателя |
|  | Mname | Да | Фамилия преподавателя |
|  | Lname | Да | Отчество преподавателя |
|  | Phone | Да | Номер телефона преподавателя |
|  | Birthday | Да | Дата рождения преподавателя |

Таблица№3-Словарь данных для таблицы AudienceType

|  |  |  |  |
| --- | --- | --- | --- |
| AudienceType | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | ID | Да | Идентификационный номер |
|  | TypeOf | Да | Тип аудитории |
|  | Capacity | Да | Вместимость аудитории |
|  | Cabinet | Да | Номер кабинета |
| Внешний | TeacherID | Да | Внешний ключ преподавателя |
| Внешний | CorpID | Да | Внешний ключ корпуса |

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

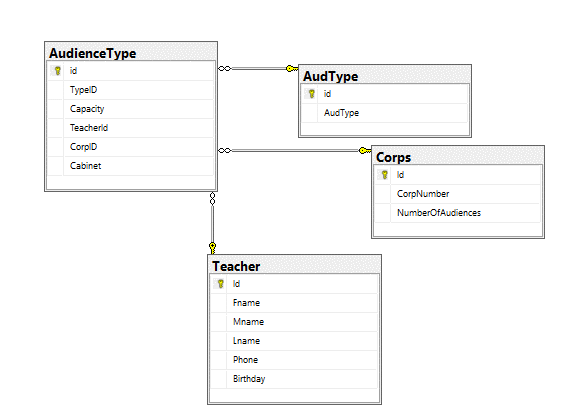


Рисунок 2.1.1 – Диаграмма базы данных

Код создания таблиц можно посмотреть в приложении А.

## 2.2 Процедуры и функции

Всего в базе данных реализованы: одна хранимая процедура и одна скалярная функция.

Процедура называется GetCountOfAllRecords() и она предназначена для того, чтобы вывести количество записей в таблице.

На выходе формируется число, которое отображает кол-во записей в таблице.

Затем была разработана скалярная функция, которая ведет расчет вместимости корпуса, она называется gethumancapacity(@corp).

На вход данная функция принимает: @corp для определения корпуса.

Выходным параметром данной функции является числовое значение, которое будет определять вместимость людей в корпусе.

Код создания хранимых процедур и функций можно посмотреть в приложении В.

# резервное копирование и восстановление

## 3.1 Резервное копирование

Для резервного копирования был реализован административный раздел, вход в который был реализован через ввод пароля. Смотреть рисунки 3.1.1 и 3.1.2 .

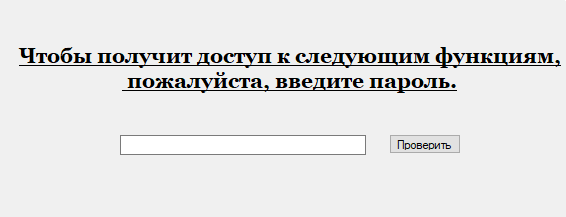


Рисунок 3.1.1 – Ввод пароля

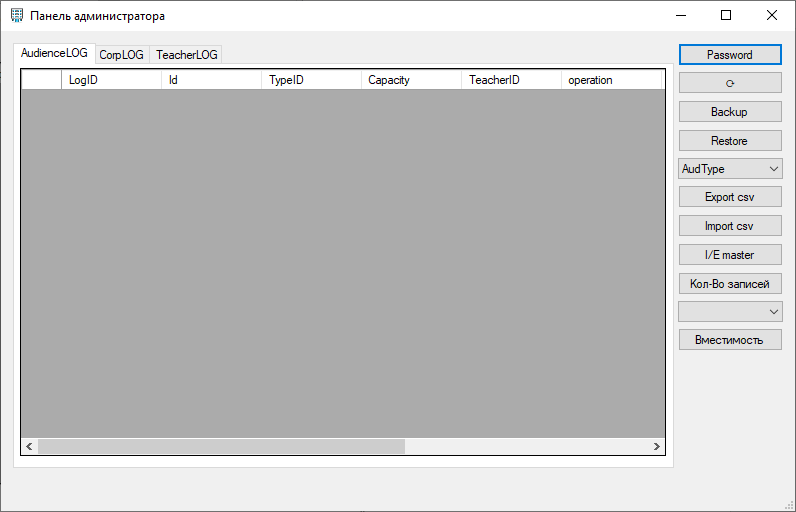


Рисунок 3.1.2 – Административная форма

В административной форме находится кнопка «Backup» по нажатию срабатывает событие, которые вызывает saveFileDialog для выбора места сохранения - смотреть рисунок 3.1.3, после нажатия кнопки «Сохранить» срабатывает запрос на создания резервной копии, листинг которого представлен ниже.

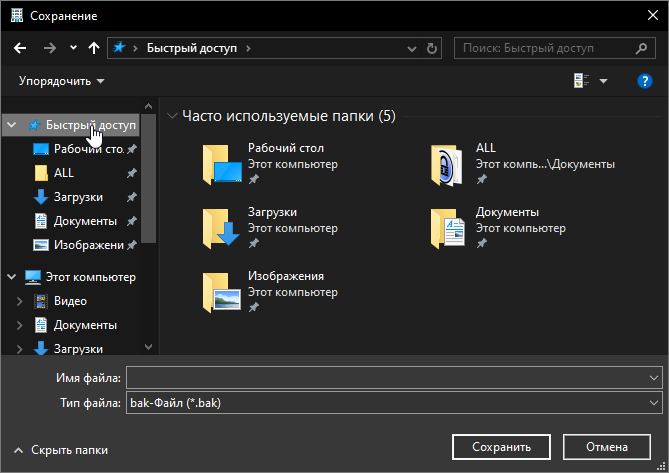


Рисунок 3.1.3 – Диалог для выбора места сохранения

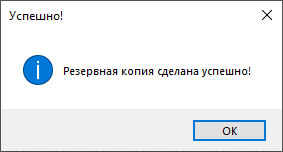


Рисунок 3.1.4 –Уведомление о успешном резервном копировании

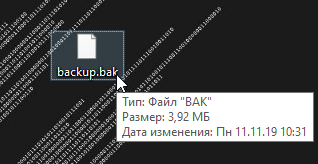
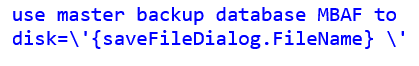
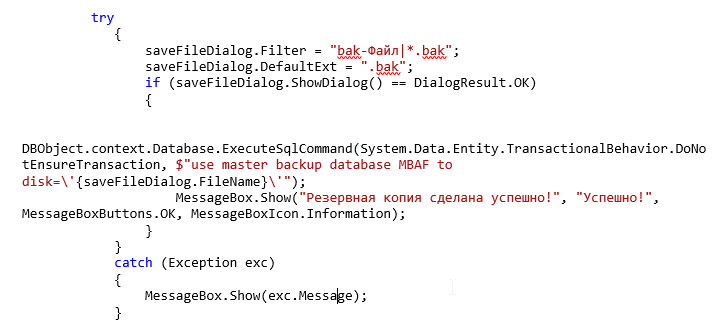


Рисунок 3.5 –Файл бекапа

SQL-запрос для создания резервной копии показан ниже:



Листинг формирования запроса из кода программы:



3.2 Восстановление базы данных из резервной копии

Аналогичным образом создается запрос на восстановление базы данных.

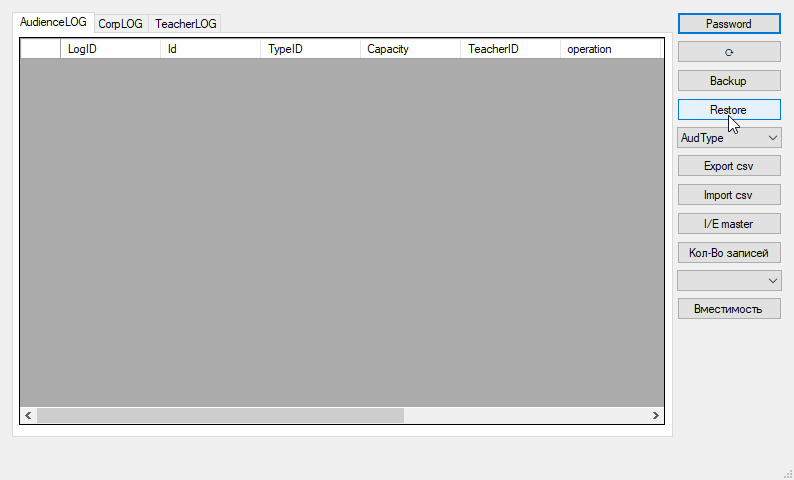


Рисунок3.2.1 –Кнопка восстановления

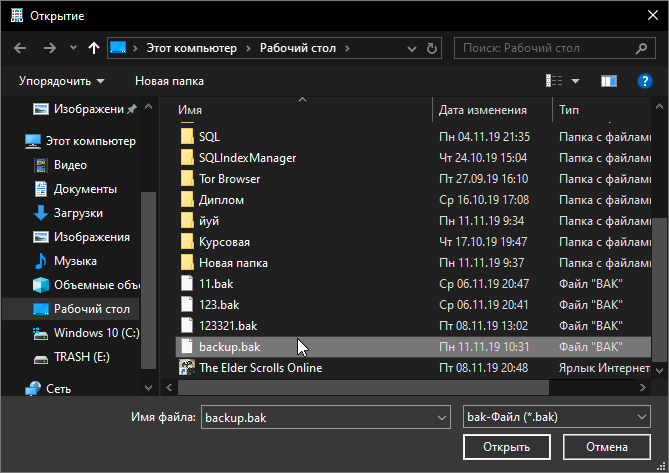
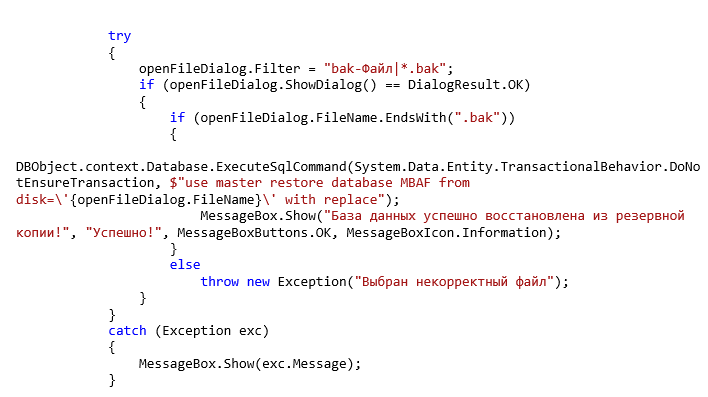


Рисунок 3.2.2 –Форма выбора файла бекапа

Sql-запрос для восстановления:



Листинг формирования запроса из кода программы:



# ИМПОрт и экспорт данных

## 4.1 Импорт данных из MS Excel

За мастер импорта/экспорта отвечает кнопка, которая запускает процесс DTSWizard.exe.

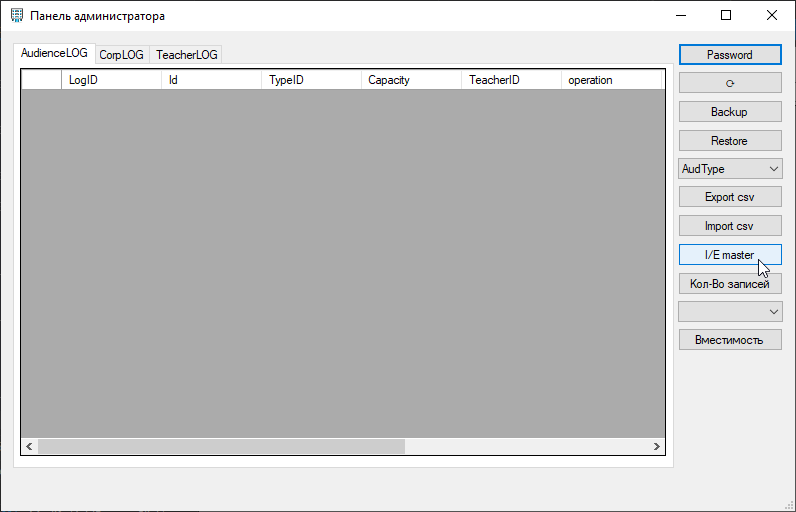


Рисунок 4.1.1 – Кнопка вызова мастера импорта/экспорта.

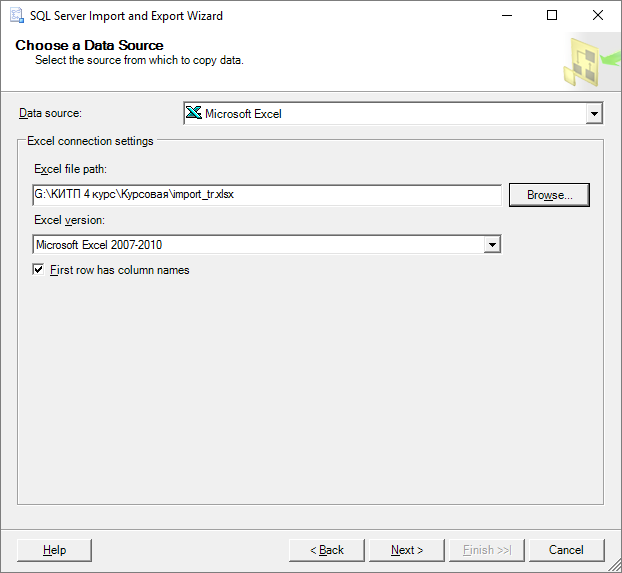


Рисунок 4.1.2 – Выбор источника импорта данных

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

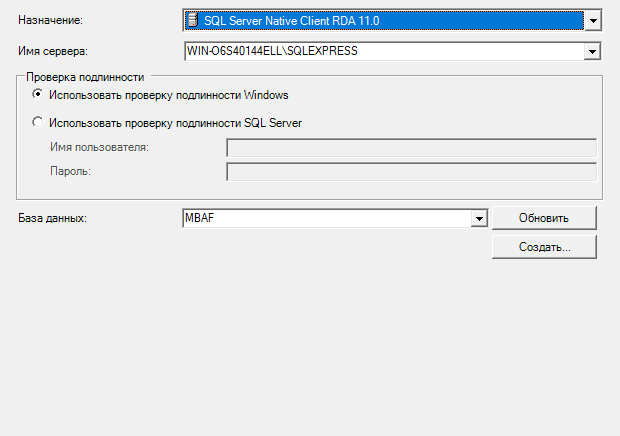


Рисунок 4.1.3 – Место назначения импорта данных

На следующей странице необходимо сопоставить страницы MS Excel и таблицы базы данных – смотреть рисунок 4.1.4, для которых эти страницы предназначены.

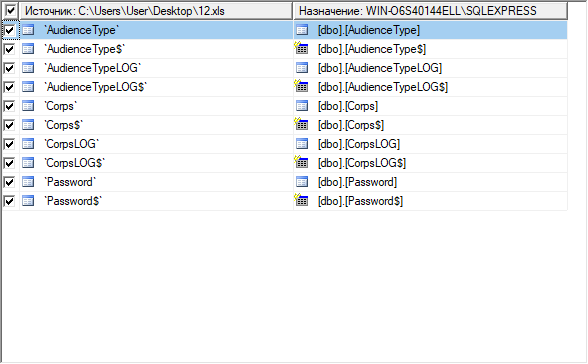


Рисунок 4.1.4 – Сопоставление страниц с таблицами

По завершению импорта данных, мастер импорта/экспорта уведомит пользователя о том, что импорт данных произошел успешно, что наглядно видно на рисунке 4.1.5.

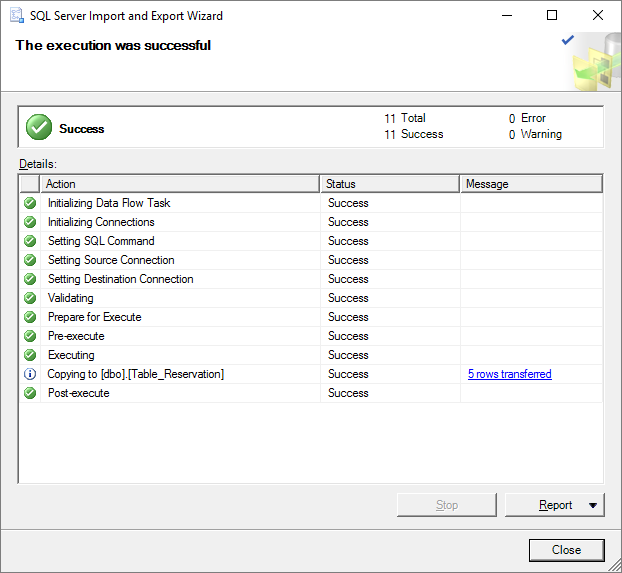


Рисунок 4.1.5 – Завершение импорта данных

## 

## Экспорт данных в csv

Для того, чтобы произвести экспорт данных в указанный файл необходимо указать в corpscomboBox для какой таблицы будет выполняться команда смотреть рисунок 4.2.1, а после в форме сохранения указать файл, в который будет записываться результат.

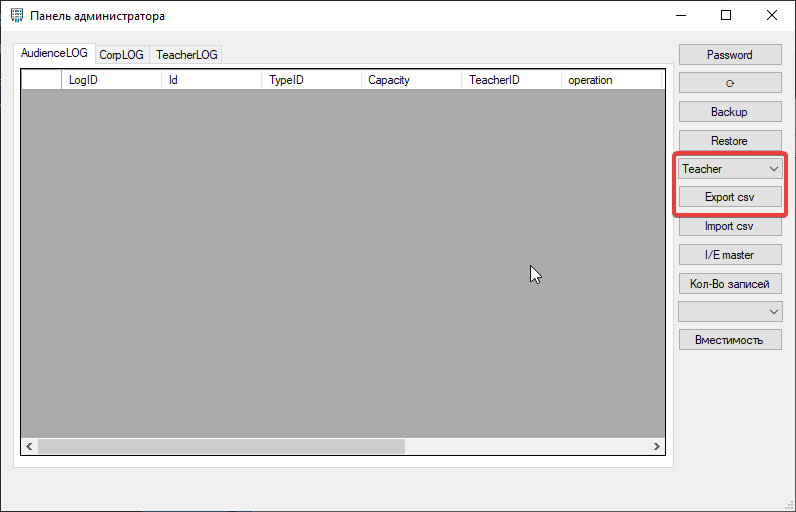


Рисунок 4.2.1 – Административная форма

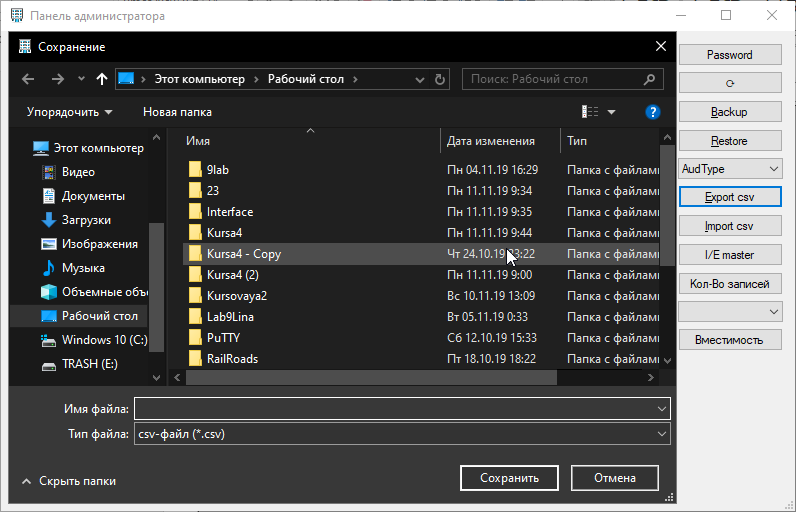


Рисунок 4.2.2 – Экспорт в csv

Пример экспорта для таблицы “Teacher” выглядит следующим образом:

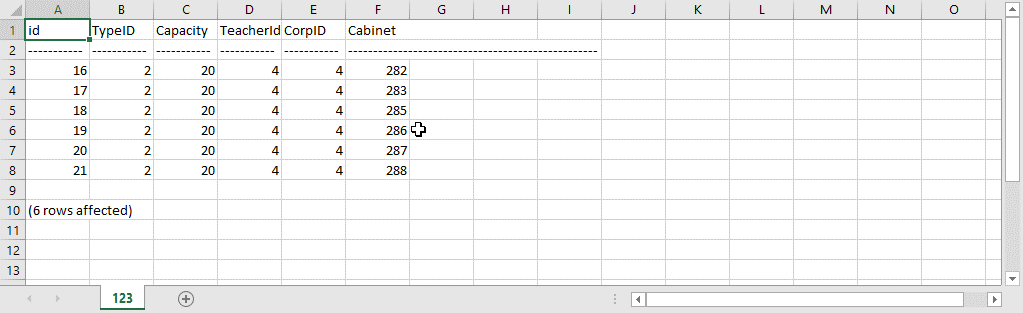


Рисунок 4.2.2 – Пример экспортированных данных

## 4.3 Импорт данных с помощью BULK INSERT

На административной форме находится кнопка «Import CSV» смотреть рисунок 4.3.1.

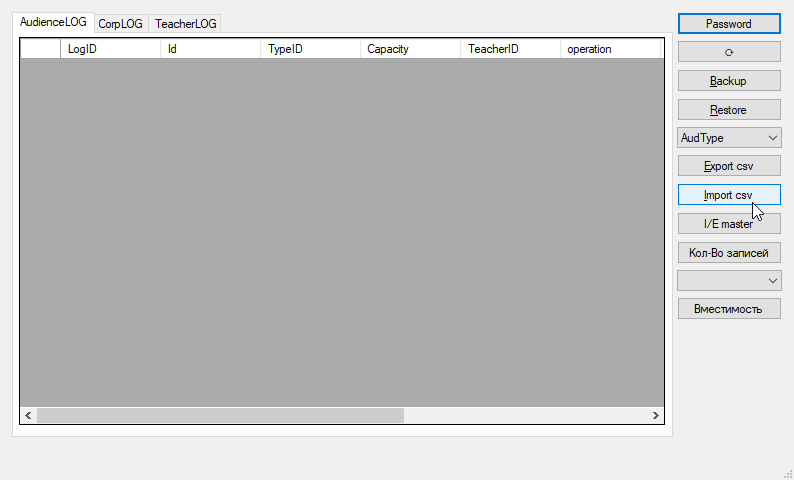


Рисунок 4.3.1 – Кнопка «Import csv»

После нажатия данной кнопки вызывается OpenFileDialog в котором пользователь выбирает файл, из которого надо импортировать данные. Смотреть рисунок 4.3.2.

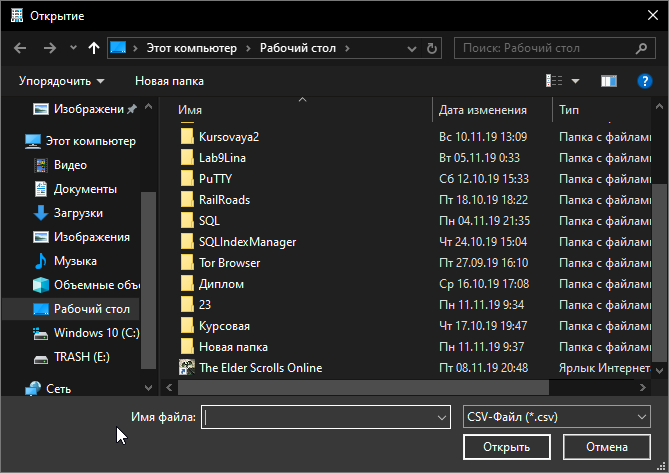
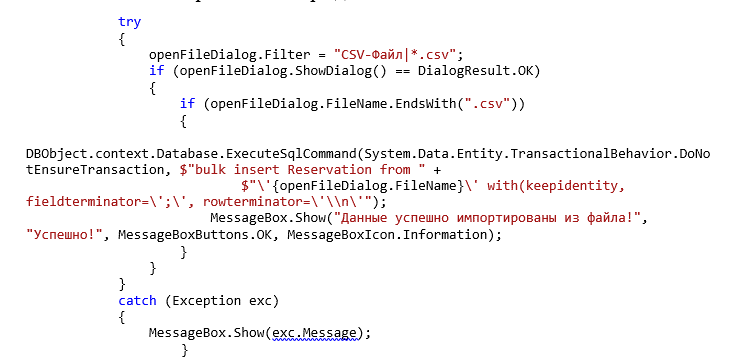


Рисунок 4.3.2 – Диалог выбора файла

Листинг запроса на импорт данных:



# 

# Управление доступом

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

* Root, который имеет неограниченные возможности по работе с данными в базе данных.
* User, основной задачей которого является лишь внесение данных в таблицы базы 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

Приложение состоит из:

* MainForm - главная форма которая отвечает за вывод таблицы AudienceType и за вызов других форм.
* AdminForm - Форма для просмотра логов, а так же для администрирования базы данных.
* AboutForm – Форма содержит описание программа и ссылки для быстрой связи с разработчиком.
* UnlockForms – Данная форма отвечает за проверку пароля.

## Пользовательский интерфейс

При запуске приложения перед пользователем появляется окно, которое содержит:

* строку поиска;
* таблицу со всеми аудиториями;
* Кнопку вызова формы администратора;
* ToolStrip для работы с таблицей;

## Главная форма

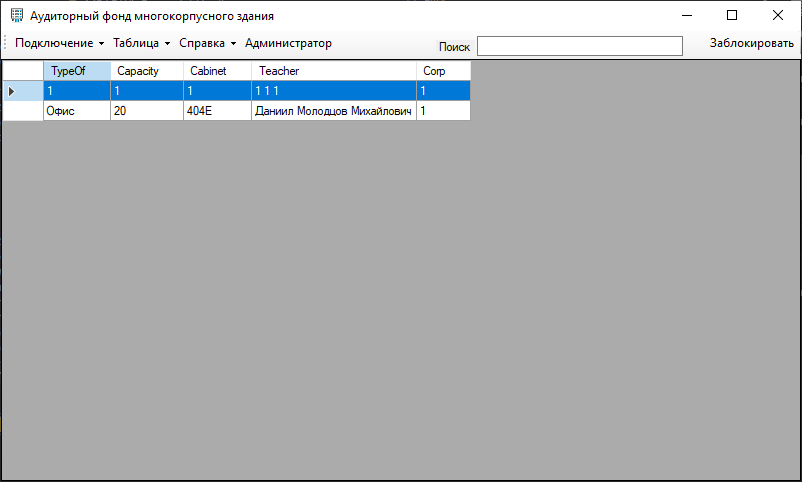


Рисунок 6.2.1.1 – Главная форма

Для того чтобы вывести информацию об аудитории, необходимо ввести в поле поиска любую информацию по аудитории. Для примера выведем информацию по аудиториям, которые находятся в 1 корпусе.

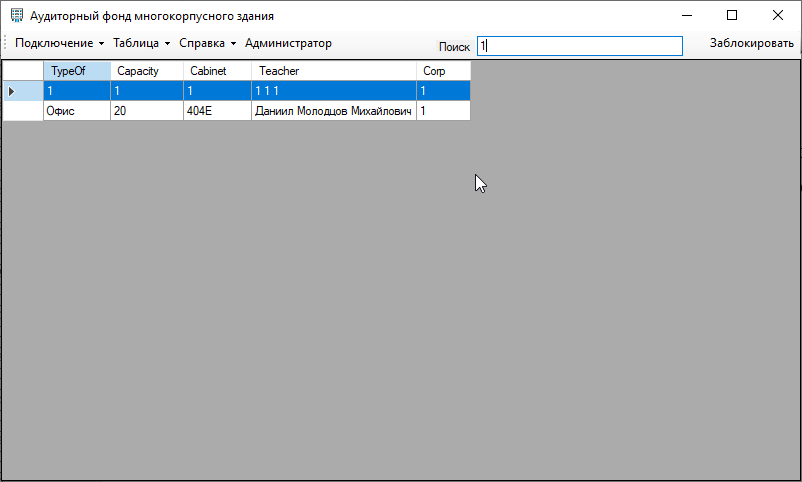


Рисунок 6.2.1.2 – Работа поиска

Для того чтобы узнать количество аудиторий, нужно зайти на административную форму и нажать на GetAllRecords.

Получим следующее окно:

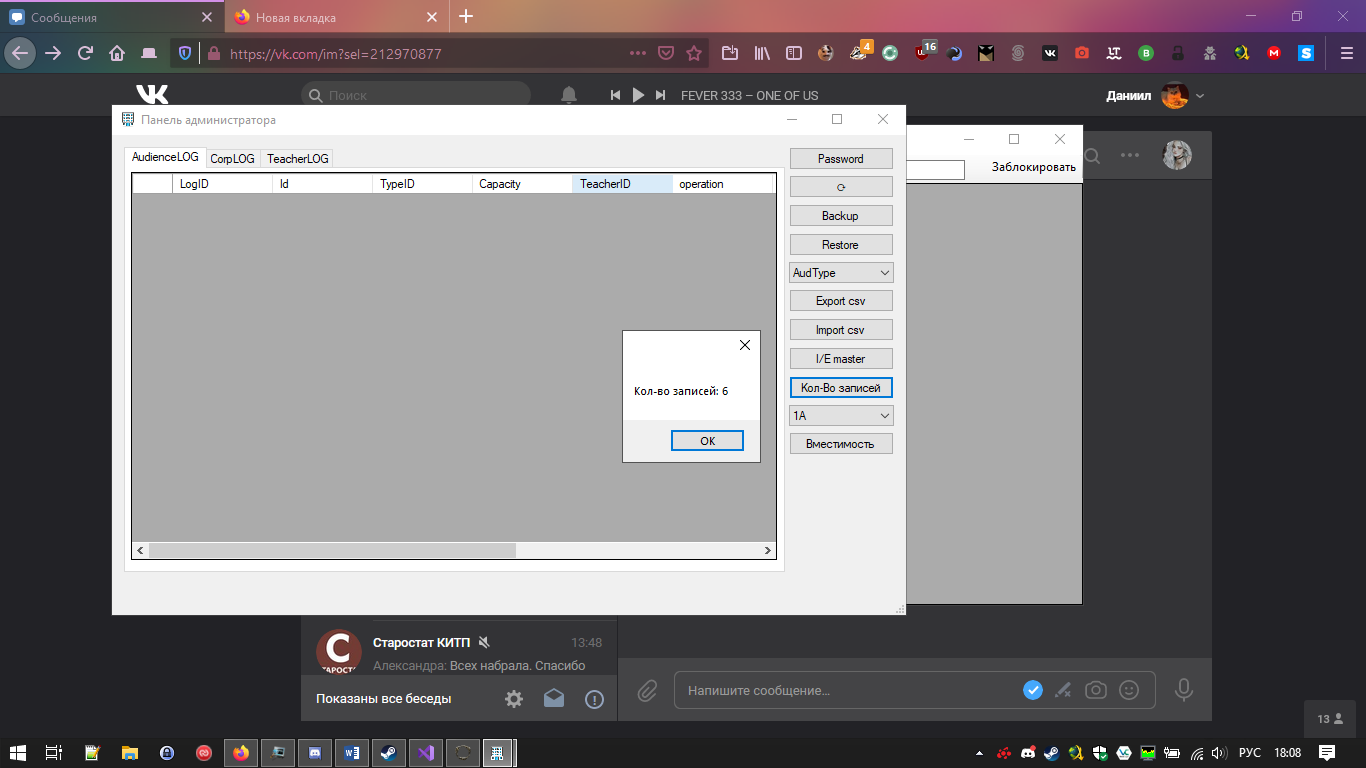


Рисунок 6.2.1.3 – Результат работы процедуры

Двойной клик по записи в таблице с аудиториями приведет к тому, что откроется форма подробной информации по аудитории со всеми полями, в которой будет отображаться список всей информации для этой аудитории.

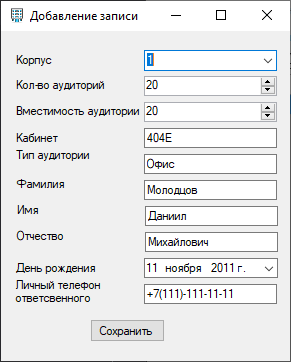


Рисунок 6.2.1.4 – Подробная информация по аудитории

На данной форме можно менять характеристики аудитории, а также добавить новую аудиторию для чего нажатием кнопки «Сохранить».

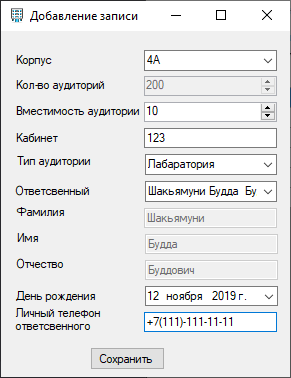


Рисунок 6.2.1.5. – Добавление аудитории

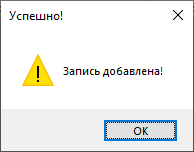


Рисунок 6.2.1.6 – Уведомление о добавлении

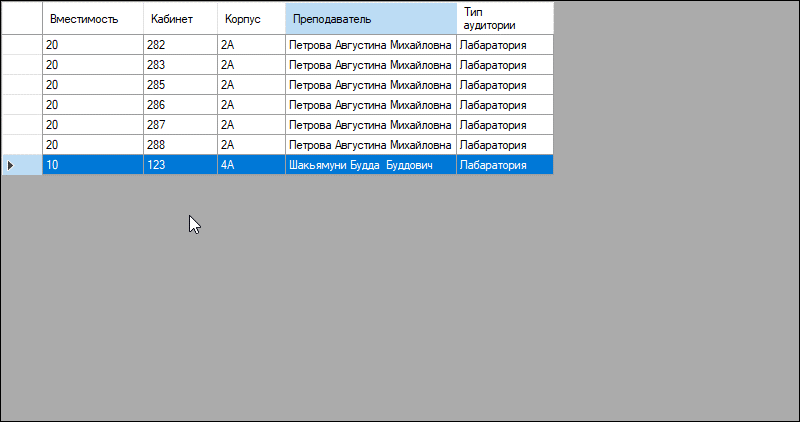


Рисунок 6.2.1.7 – Запись добавлена в таблицу

На главной форме можно удалить запись об аудитории. Для этого надо кликнуть левой кнопкой мыши, а затем в ToolStrip выбрать “удалить аудиторию”.

Для примера, удалим ранее добавленную запись.

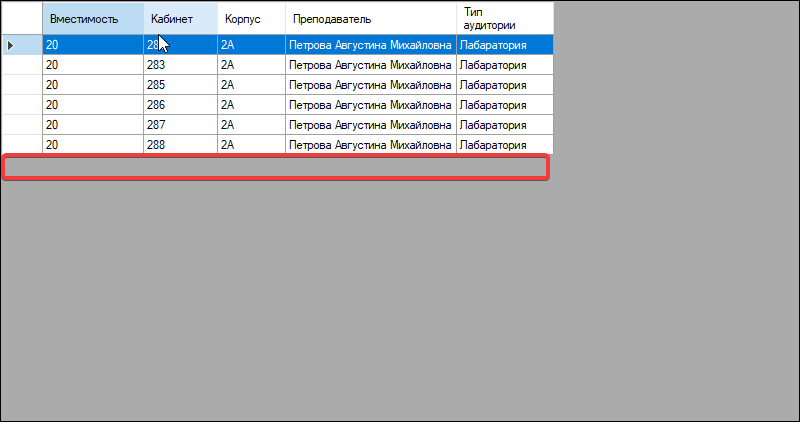


Рисунок 6.2.1.8 – Запись удалена

Что бы изменить формы достаточно дважды кликнуть на нужную нам запись. Изменим запись кабинета 288.

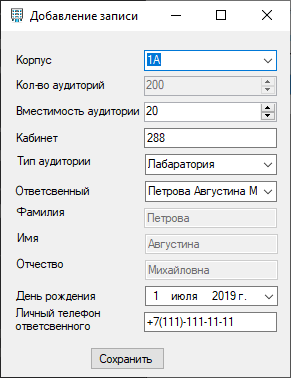


Рисунок 6.2.1.9 – Исходные данные записи

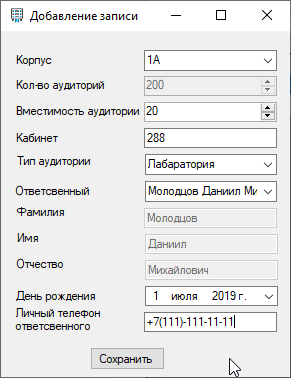


Рисунок 6.2.1.10 – Изменение данных записи

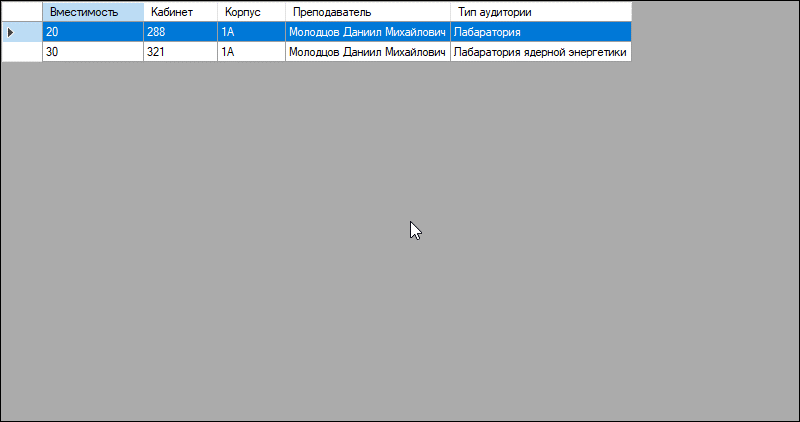


Рисунок 6.2.1.11-Измененая запись

После изменения данных необходимо сохранить изменения в базе данных либо отменить изменения закрыв форму.

## Раздел администратора

На данной форме находятся панели логов для таблиц, а так же панель выполнения произвольной SQL команды, смотреть рисунок 6.2.2.1.

Вход в данную форму происходит через ввод пароля, смотреть рисунок 6.2.2.2.

Если пароль был введен не верно, форма уведомит пользователя об этом, смотреть рисунок 6.2.2.3.

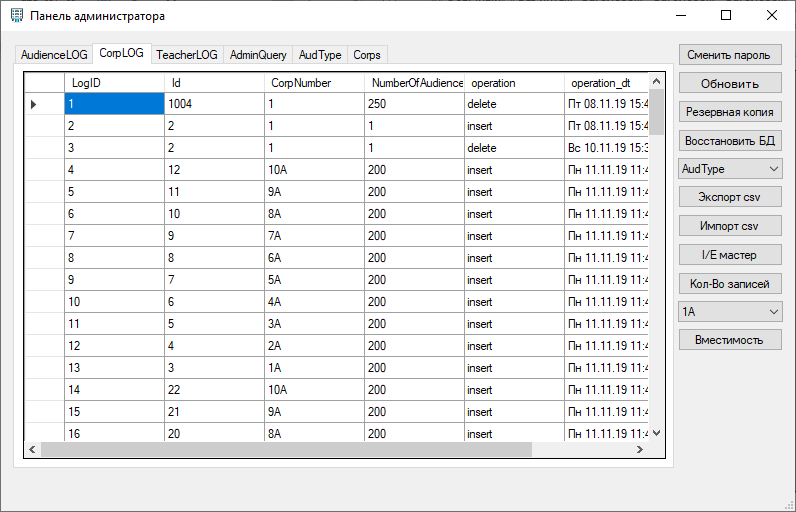


Рисунок 6.2.2.1-Форма администратора

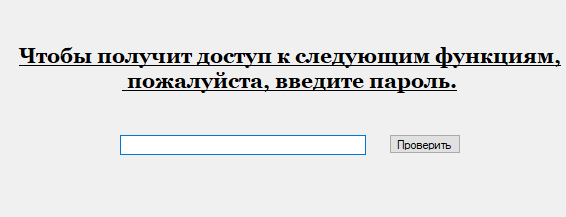


Рисунок 6.2.2.2-Форма ввода пароля

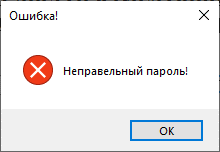


Рисунок 6.2.2.3-Ошибка ввода пароля

## Форма о программе

Данная форма служит для быстрой связи с разработчиком, данную форму можно увидеть на рисунке 6.2.3.1 что находится ниже.

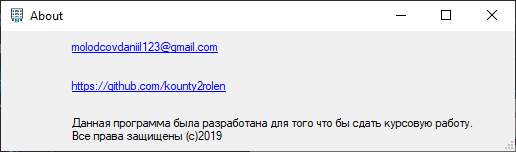


Рисунок 6.2.3.1-Форма о программе

# заключение

Результатом выполнения данной работы стала база данных, позволяющая организовать работу системы учёта аудиторий. 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 2019 и Microsoft SQL Server Management Studio 18.2.

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

# Список использованных источников

1. Дополнительная документация по SQL // Документация по SQL и всё что с ним связано URL: <https://www.sql.ru/docs/>
2. Техническая документация SQL // Техническая документация по SQL Server URL: <https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-2016>
3. Документация по EntityFramework // EntityFramework-EF URL: <https://docs.microsoft.com/ru-ru/ef/>
4. Руководство по EntityFramework // Руководство по ADO.NET EntityFramework 6 URL:<https://metanit.com/sharp/entityframework/>

# ПРИЛОЖЕНИЕ А

## Создание БД

create table Teacher

(

Id int primary key identity(1,1),

Fname text not null default 'None',

Mname text not null default 'None',

Lname text not null default 'None',

Phone text not null default 'None',

Birthday Datetime

)

go

create table AudienceType

(

Id int primary key identity(1,1),

TypeOf text not null default 'None',

Capacity int not null default 0,

ResponsibleID int foreign key references Teacher(Id)

)

go

create table corps

(

Id int primary key identity(1,1),

CorpNumber text not null default '0A',

NumberOfAudiences int default 0,

)

go

# 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

create table AudienceTypeLOG

(

Id int,

TypeOf text ,

Capacity int,

ResponsibleID int,

operation VARCHAR(20),

operation\_dt DATETIME ,

operation\_user VARCHAR(255)

)

go

create table CorpsLOG

(

Id int ,

CorpNumber text,

NumberOfAudiences int,

operation VARCHAR(20),

operation\_dt DATETIME ,

operation\_user VARCHAR(255)

)

go

create table TeacherLOG

(

Id int ,

Fname text,

Mname text,

Lname text,

Phone text,

Birthday Datetime,

operation VARCHAR(20),

operation\_dt DATETIME ,

operation\_user VARCHAR(255)

)

go

create trigger AudienceType\_u on AudienceType

after update

as

begin

INSERT INTO AudienceTypeLOG

--LOG old string

SELECT id,

TypeID,

Capacity,

TeacherId,

'update\_old',

Getdate(),

Suser\_name(),

CorpID,

Cabinet

from deleted

--LOG new string

INSERT INTO AudienceTypeLOG

SELECT id,

TypeID,

Capacity,

TeacherId,

'update\_new',

Getdate(),

Suser\_name(),

CorpID,

Cabinet

FROM inserted

END

go

CREATE TRIGGER AudienceType\_i ON AudienceType

after INSERT

AS

BEGIN

INSERT INTO AudienceTypeLOG

SELECT id,

TypeID,

Capacity,

TeacherId,

'insert',

Getdate(),

Suser\_name(),

CorpID,

Cabinet

FROM inserted

END

go

CREATE TRIGGER Audience\_d ON AudienceType

after DELETE

AS

BEGIN

INSERT INTO AudienceTypeLOG

SELECT id,

TypeID,

Capacity,

TeacherId,

'insert',

Getdate(),

Suser\_name(),

CorpID,

Cabinet

FROM deleted

END

go

create trigger Corps\_u on corps

after update

as

begin

INSERT INTO TeacherLOG

--LOG old string

SELECT id,

CorpNumber,

NumberOfAudiences,

'update\_old',

Getdate(),

Suser\_name()

from deleted

--LOG new string

INSERT INTO TeacherLOG

SELECT id,

CorpNumber,

NumberOfAudiences,

'update\_new',

Getdate(),

Suser\_name()

FROM inserted

END

go

CREATE TRIGGER Corps\_i ON corps

after INSERT

AS

BEGIN

INSERT INTO TeacherLOG

SELECT id,

CorpNumber,

NumberOfAudiences,

'insert',

Getdate(),

Suser\_name()

FROM inserted

END

go

CREATE TRIGGER Corps\_d ON corps

after DELETE

AS

BEGIN

INSERT INTO TeacherLOG

SELECT id,

CorpNumber,

NumberOfAudiences,

'delete',

Getdate(),

Suser\_name()

FROM deleted

END

go

create trigger Teacher\_u on Teacher

after update

as

begin

INSERT INTO TeacherLOG

--LOG old string

SELECT id,

Fname,

Mname,

Lname,

Phone,

Birthday,

'update\_old',

Getdate(),

Suser\_name()

FROM deleted

--LOG new string

INSERT INTO TeacherLOG

SELECT id,

Fname,

Mname,

Lname,

Phone,

Birthday,

'update\_new',

Getdate(),

Suser\_name()

FROM inserted

END

go

CREATE TRIGGER Teacher\_i ON Teacher

after INSERT

AS

BEGIN

INSERT INTO TeacherLOG

SELECT id,

Fname,

Mname,

Lname,

Phone,

Birthday,

'insert',

Getdate(),

Suser\_name()

FROM inserted

END

go

CREATE TRIGGER Teacher\_d ON Teacher

after DELETE

AS

BEGIN

INSERT INTO TeacherLOG

SELECT id,

Fname,

Mname,

Lname,

Phone,

Birthday,

'delete',

Getdate(),

Suser\_name()

FROM deleted

END

go

SET ansi\_padding OFF

go

## Создание ролей и пользователей

use MBAF

go

create login userlogin

with password = '123';

create user user1

for login userlogin

create role root authorization userlogin

create login notrootlogin

with password = '123';

create user nonroot

for login notrootlogin

create role authorization notrootlogin

deny delete

to notrootlogin;

# ПРИЛОЖЕНИЕ Г

## Код основной формы

namespace MBAF

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

this.CenterToScreen();

DGVRefresh();

}

/// <summary>

/// Заполнение главной таблицы

/// и сокрытие ненужных столбцов

/// </summary>

void DGVRefresh()

{

MainDataGridView.DataSource = DBObject.context.AudienceType.ToList();

MainDataGridView.Columns["id"].Visible = false;

MainDataGridView.Columns["Teacherid"].Visible = false;

MainDataGridView.Columns["Corpid"].Visible = false;

MainDataGridView.Columns["TypeId"].Visible = false;

MainDataGridView.Columns["Capacity"].HeaderText = "Вместимость";

MainDataGridView.Columns["Cabinet"].HeaderText = "Кабинет";

MainDataGridView.Columns["Corps"].HeaderText = "Корпус";

MainDataGridView.Columns["Teacher"].HeaderText = "Преподаватель";

MainDataGridView.Columns["AudType"].HeaderText = "Тип аудитории";

}

/// <summary>

/// Показывает форму добавления аудитории

/// </summary>

/// <param name="audience">Аудитория</param>

/// <param name="teacher">Преподаватель</param>

/// <param name="corps">Корпус</param>

void ShowAddAudience(AudienceType audience,Teacher teacher,Corps corps)

{

WinForms.AddAudience add;

if (MainDataGridView.DataSource != null)

{

add = new WinForms.AddAudience(audience,teacher,corps);

add.ShowDialog();

add.Dispose();

DGVRefresh();

}

else

MessageBox.Show("Таблица не подключена! Пожалуйста подключите таблицу и попробуйте еще раз", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

/// <summary>

/// Удаление строки по индексу выбранному в DGV

/// </summary>

void DeleteRow()

{

int? id = 0;

if (MainDataGridView.DataSource != null)

if (MainDataGridView.CurrentRow != null)

{

id = Convert.ToInt32(MainDataGridView.CurrentRow.Cells[0].Value);

if (id != null)

{

EntityModel.AudienceType audience = DBObject.context.AudienceType.Where(c => c.id == id).FirstOrDefault();

DBObject.context.AudienceType.Remove(audience);

DBObject.context.SaveChanges();

DGVRefresh();

}

else

MessageBox.Show("Таблица не подключена! Пожалуйста подключите таблицу и попробуйте еще раз", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

MessageBox.Show("Запись не выбрана! Пожалуйста выберете запись и попробуйте еще раз", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

void ShowAdminForm()

{

WinForms.Administrativ.AdminForm admin = new WinForms.Administrativ.AdminForm();

Lockscreen();

admin.ShowDialog();

admin.Dispose();

}

void Lockscreen()

{

WinForms.Administrativ.UnlockForm unlock = new WinForms.Administrativ.UnlockForm();

unlock.ShowDialog();

unlock.Dispose();

}

#region события

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

{

AudienceType audience = new AudienceType();

Teacher teacher = new Teacher();

Corps corps = new Corps();

ShowAddAudience(audience, teacher, corps);

}

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

{

DeleteRow();

}

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

{

WinForms.About about = new WinForms.About();

about.ShowDialog();

about.Dispose();

}

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

{

ShowAdminForm();

}

private void ToolStripButton3\_Click(object sender, EventArgs e)

{

Lockscreen();

}

private void MainDataGridView\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

AudienceType audiencetype = (AudienceType)MainDataGridView.Rows[e.RowIndex].DataBoundItem;

Teacher teacher = audiencetype.Teacher;

Corps corps = audiencetype.Corps;

ShowAddAudience(audiencetype,teacher, corps);

}

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

{

MainDataGridView.DataSource = DBObject.context.AudienceType.Where(c => c.Teacher.Lname.Contains(searchTextBox.Text)|| c.Teacher.Fname.Contains(searchTextBox.Text) ||

c.Teacher.Mname.Contains(searchTextBox.Text) || c.AudType.AudType1.Contains(searchTextBox.Text) || c.Corps.CorpNumber.ToString().Contains(searchTextBox.Text)

|| c.Capacity.ToString().Contains(searchTextBox.Text) ||c.Cabinet.Contains(searchTextBox.Text)).ToList();

}

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

{

DGVRefresh();

}

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

{

MainDataGridView.DataSource = null;

}

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

{

if (MainDataGridView.DataSource != null)

{

DGVRefresh();

}

}

#endregion

}

}

# ПРИЛОЖЕНИЕ Д

## Код формы редактирования и добавления аудиторий

namespace MBAF.WinForms

{

public partial class AddAudience : Form

{

private readonly AudienceType audience;

private readonly Teacher teacher;

private readonly Corps corps;

private readonly AudType audType = new AudType();

public AddAudience(AudienceType audtype, Teacher teacher, Corps corp)

{

this.audience = audtype;

this.teacher = teacher;

this.corps = corp;

InitializeComponent();

CorpsComboBox.DataSource = DBObject.context.Corps.Select(c => c.CorpNumber).Distinct().ToList();

CorpsComboBox.SelectedIndex = -1;

//Заполнение комбобокса Corps

audTypeComboBox.DataSource = DBObject.context.AudType.Select(c => c.AudType1).Distinct().ToList();

responseComboBox.DataSource= DBObject.context.Teacher.Select(c =>c.Mname+" "+c.Fname+" "+c.Lname).Distinct().ToList();

//Заполнение комбокоса aud

if (DBObject.context.AudienceType.Where(c => c.id == audtype.id).Count() > 0)

Fill();

}

/// <summary>

/// Заполнение полей

/// </summary>

void Fill()

{

CorpsComboBox.SelectedItem = corps.CorpNumber;

audTypeComboBox.SelectedItem = audience.AudType.AudType1;

responseComboBox.SelectedItem = teacher.ToString();

AuditorNumericUpDown.Value = (int)corps.NumberOfAudiences;

AuditoryCapacityNumericUpDown.Value = audience.Capacity;

CabinetTextBox.Text = audience.Cabinet;

dateTimePicker.Value = (DateTime)teacher.Birthday;

PhoneMaskedTextBox.Text = teacher.Phone;

}

#region addrecord

/// <summary>

/// Проверка и заполнение обьектов

/// </summary>

void Fillentity()

{

try

{

teacher.Mname = MnameTextBox.Text;

teacher.Fname = FnameTextBox.Text;

teacher.Lname = LnameTextBox.Text;

teacher.Phone = PhoneMaskedTextBox.Text;

teacher.Birthday = dateTimePicker.Value;

corps.CorpNumber = CorpsComboBox.Text;

corps.NumberOfAudiences = (int)AuditorNumericUpDown.Value;

if (DBObject.context.AudType.Where(c => c.AudType1 == audTypeComboBox.Text).Count() <= 0)

{

audType.AudType1 = audTypeComboBox.Text;

DBObject.context.AudType.Add(audType);

audience.TypeID = audType.id;

}

audience.TypeID = DBObject.context.AudType.Where(c => c.AudType1 == audTypeComboBox.Text).Select(c => c.id).FirstOrDefault();

audience.Capacity = (int)AuditoryCapacityNumericUpDown.Value;

audience.TeacherId = teacher.Id;

audience.CorpID = corps.Id;

audience.Cabinet = CabinetTextBox.Text;

if (DBObject.context.Corps.Where(c => c.CorpNumber == corps.CorpNumber).Count() == 0)

{

DBObject.context.Corps.Add(corps);

}

else

{

audience.CorpID = DBObject.context.Corps.Where(c => c.CorpNumber == corps.CorpNumber).Select(c => c.Id).FirstOrDefault();

}

if (DBObject.context.Teacher.Where(c => c.Fname + c.Mname + c.Lname == teacher.Fname + teacher.Mname + teacher.Lname).Count() == 0)

{

DBObject.context.Teacher.Add(teacher);

}

else

{

audience.TeacherId = DBObject.context.Teacher.Where(c => c.Fname + c.Mname + c.Lname == teacher.Fname + teacher.Mname + teacher.Lname).Select(c => c.Id).FirstOrDefault();

}

if (DBObject.context.AudienceType.Where(c => c.Cabinet == audience.Cabinet && c.CorpID == audience.CorpID).Count() == 0)

DBObject.context.AudienceType.Add(audience);

else if(audience.id==0)

{

throw new AlreadyContainsException("Запись уже существует");

}

MessageBox.Show("Запись добавлена!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

DBObject.context.SaveChanges();

Close();

Dispose();

}

catch (AlreadyContainsException exc)

{

MessageBox.Show(exc.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception exc)

{

MessageBox.Show("Проверьте введеные данные \n" + exc.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

Fillentity();

}

#endregion

#region maintaince

void corpsSelected()

{

if (CorpsComboBox.SelectedIndex != -1)

{

var a = (decimal)DBObject.context.Corps.Where(c => c.CorpNumber == CorpsComboBox.Text).Select(c => c.NumberOfAudiences).FirstOrDefault();

AuditorNumericUpDown.Value = a;

AuditorNumericUpDown.Enabled = false;

}

else

{

AuditorNumericUpDown.Value = 0;

AuditorNumericUpDown.Enabled = true;

}

}

//Тоже самое что и ниже

private void CorpsComboBox\_SelectedIndexChanged(object sender, EventArgs e)

{

corpsSelected();

}

/// <summary>

/// если выберут ответсвенного то смысла в текстбоксах не будет,

//так что заполняем их данными для пользвоателя и убираем

/// </summary>

void responseSelected()

{

if (responseComboBox.SelectedIndex != -1&& responseComboBox.Text!=string.Empty)

{

var a = DBObject.context.Teacher.Where(c => c.Mname + " " + c.Fname + " " + c.Lname == responseComboBox.SelectedItem.ToString());

FnameTextBox.Text = a.Select(c => c.Fname).FirstOrDefault();

MnameTextBox.Text = a.Select(c => c.Mname).FirstOrDefault();

LnameTextBox.Text = a.Select(c => c.Lname).FirstOrDefault();

FnameTextBox.Enabled = false;

MnameTextBox.Enabled = false;

LnameTextBox.Enabled = false;

}

else

{

FnameTextBox.Enabled = true;

MnameTextBox.Enabled = true;

LnameTextBox.Enabled = true;

}

}

private void responseComboBox\_SelectedIndexChanged(object sender, EventArgs e) => responseSelected();

#endregion//"Технисческий раздел"

private void responseComboBox\_TextUpdate(object sender, EventArgs e)

{

responseSelected();

}

}

class AlreadyContainsException : Exception

{

public AlreadyContainsException(string message)

: base(message)

{ }

}//Класс для вызова исключения если запись уже существует

}

# ПРИЛОЖЕНИЕ Ж

## Код административной формы

namespace MBAF.WinForms.Administrativ

{

public partial class AdminForm : Form

{

public AdminForm()

{

InitializeComponent();

fill();

}

//Вызов формы смены пароля

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

{

ChangePassword changePasswor = new ChangePassword();

changePasswor.ShowDialog();

changePasswor.Dispose();

}

/// <summary>

/// заполнение таблиц

/// </summary>

void fill()

{

corpscomboBox.DataSource = DBObject.context.Corps.Select(c => c.CorpNumber).ToList();

TablecomboBox.DataSource = DBObject.context.Database.SqlQuery<string>(@"use MBAF Select TABLE\_NAME from INFORMATION\_SCHEMA.TABLES where

TABLE\_NAME not like 'sysdiagrams'").ToList();

TeacherLogDataGridView.DataSource = DBObject.context.TeacherLOG.ToList();

TeacherLogDataGridView.Columns["operation\_dt"].SortMode = DataGridViewColumnSortMode.Automatic;

CorpLogDataGridView.DataSource = DBObject.context.CorpsLOG.ToList();

CorpLogDataGridView.Columns["operation\_dt"].SortMode = DataGridViewColumnSortMode.Automatic;

AudienceLOGDataGridView.DataSource = DBObject.context.AudienceTypeLOG.ToList();

AudienceLOGDataGridView.Columns["operation\_dt"].SortMode = DataGridViewColumnSortMode.Automatic;

audTypeDataGridView.DataSource = DBObject.context.AudType.ToList();

corpsDataGridView.DataSource = DBObject.context.Corps.ToList();

}

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

{

fill();

}

#region adminstrativJOB

/// <summary>

/// создание бекапа

/// </summary>

void Backup()

{

try

{

saveFileDialog.Filter = "bak-Файл|\*.bak";

saveFileDialog.DefaultExt = ".bak";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

DBObject.context.Database.ExecuteSqlCommand(System.Data.Entity.TransactionalBehavior.DoNotEnsureTransaction, $"use master backup database MBAF" +

$" to disk=\'{saveFileDialog.FileName}\'");

MessageBox.Show("Резервная копия сделана успешно!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

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

{

Backup();

}

/// <summary>

/// Восстанавление данных в SQL с файла бекапа

/// </summary>

void Restore()

{

try

{

openFileDialog.Filter = "bak-Файл|\*.bak";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

if (openFileDialog.FileName.EndsWith(".bak"))

{

DBObject.context.Database.ExecuteSqlCommand(System.Data.Entity.TransactionalBehavior.DoNotEnsureTransaction, $"use master restore database MBAF from disk=\'{openFileDialog.FileName}\' with replace");

MessageBox.Show("База данных успешно восстановлена из резервной копии!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

throw new Exception("Выбран некорректный файл");

}

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

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

{

Restore();

}

void Export()

{

try

{

saveFileDialog.Filter = "csv-файл|\*.csv";

saveFileDialog.DefaultExt = ".csv";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

Process.Start("sqlcmd.exe", $"-S WIN-O6S40144ELL\\SQLEXPRESS -d MBAF -Q \"select \* from {TablecomboBox.SelectedItem}\" " +

$"-o \"{saveFileDialog.FileName}\" -s" + ';' + " -w 700 ").WaitForExit();

MessageBox.Show("Таблица успешно экпортирована!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

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

{

Export();

}

void Import()

{

try

{

openFileDialog.Filter = "CSV-Файл|\*.csv";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

if (openFileDialog.FileName.EndsWith(".csv"))

{

DBObject.context.Database.ExecuteSqlCommand(System.Data.Entity.TransactionalBehavior.DoNotEnsureTransaction, $"bulk insert Reservation from " +

$"\'{openFileDialog.FileName}\' with(keepidentity, fieldterminator=\';\', rowterminator=\'\\n\'");

MessageBox.Show("Данные успешно импортированы из файла!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

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

{

Import();

}

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

{

Process.Start(@"C:\Program Files\Microsoft SQL Server\140\DTS\Binn\DTSWizard.exe");

}

#endregion

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

{

var result = DBObject.context.GetCountOfAllRecords()?.FirstOrDefault().ToString() ?? "Ошибка";

MessageBox.Show("Кол-во записей: " + result);

}

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

{

var result = DBObject.context.Database.SqlQuery<int?>($"use MBAF select dbo.GetHumanCapacity({ corpscomboBox.SelectedItem.ToString()})");

MessageBox.Show(result.FirstOrDefault()?.ToString() ?? "Таблица не выбрана!");

}

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

{

try

{

DBObject.context.Database.ExecuteSqlCommand(queryTextBox.Text);

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

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

{

if (audTypeDataGridView.SelectedRows.Count > 0)

{

AudType audType = (AudType)audTypeDataGridView.CurrentRow.DataBoundItem;

DBObject.context.AudType.Remove(audType);

fill();

}

}

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

{

if (corpsDataGridView.SelectedRows.Count > 0)

{

Corps corps = (Corps)corpsDataGridView.CurrentRow.DataBoundItem;

DBObject.context.Corps.Remove(corps);

fill();

}

}

}

}

# ПРИЛОЖЕНИЕ З

## Код формы проверки пароля

namespace MBAF.WinForms.Administrativ

{

public partial class UnlockForm : Form

{

readonly int password = -1;

public UnlockForm()

{

InitializeComponent();

CenterToParent();

if (DBObject.context.Password.First().Password1 != -1)

password =(int) DBObject.context.Password.FirstOrDefault().Password1;

}

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

{

if (password != -1)

{

if (PasswordtextBox.Text.GetHashCode() == password)

{

Close();

Dispose();

}

else

{

MessageBox.Show("Неправельный пароль!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Пожалуйста установите пароль!");

this.Close();

this.Dispose();

}

}

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

{

Close();

}

}

}

# ПРИЛОЖЕНИЕ К

## Код формы смены пароля

namespace MBAF.WinForms.Administrativ

{

public partial class ChangePassword : Form

{

readonly int password;

public ChangePassword()

{

InitializeComponent();

password = (int)DBObject.context.Password.FirstOrDefault().Password1;

}

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

{

EntityModel.Password pass = DBObject.context.Password.FirstOrDefault();

if (OldPasswordTextBox.Text.GetHashCode() == password||password==-1)

{

if (NewPasswordTextBox.Text == RepeatPasswordTextBox.Text)

{

pass.Password1 = NewPasswordTextBox.Text.GetHashCode();

DBObject.context.SaveChanges();

MessageBox.Show("Пароль успешно изменен!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

MessageBox.Show("Пароли не совпадают, проверьте введеные данные!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

MessageBox.Show("Старый пароль не верен", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

# ПРИЛОЖЕНИЕ Л

## Код формы «о программе»

namespace MBAF.WinForms

{

public partial class About : Form

{

public About()

{

InitializeComponent();

}

private void About\_FormClosing(object sender, FormClosingEventArgs e)

{

this.Dispose();

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

System.Diagnostics.Process.Start("mailto:molodcovdaniil123@gmail.com");

}

private void linkLabel2\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

System.Diagnostics.Process.Start("https://github.com/kounty2rolen");

}

}

}