|  |
| --- |
| **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ** |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
|  |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |

|  |
| --- |
| **КУРСОВОЙ ПРОЕКТ** |
|  |
| по междисциплинарному курсу: МДК.02.02. Технология разработки и зашиты баз данных. |
|  |
| на тему: Разработка информационной системы приёмной комиссии колледжа. |
|  |
| студента группы 090203-9o-20/1  специальности 09.02.03 Программирование в компьютерных системах |
| Лофенфельда Георгия Владимировича |

|  |  |  |
| --- | --- | --- |
| Студент |  | Г.В Лофенфельд |
| Руководитель курсового проекта |  | Е.А. Ларионова |
| Председатель ПЦК специальности 09.02.03 Программирование в компьютерных системах |  | А.И. Глускер |

|  |  |
| --- | --- |
| Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2023 г. | |
| Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| Заведующий отделением № 1 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И.А. Миланова |

Москва

2023

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc148516900)

[1 ОБЩАЯ ЧАСТЬ 4](#_Toc148516901)

[1.1 Назначения и цели создания системы 4](#_Toc148516902)

[1.2 Обзор и анализ предметной области 4](#_Toc148516903)

[1.3 Жизненный цикл базы данных 5](#_Toc148516904)

[1.4 Выбор и характеристика СУБД 6](#_Toc148516905)

[1.5 Выбор и характеристика среды разработки приложения 7](#_Toc148516906)

[2 СПЕЦИАЛЬНАЯ ЧАСТЬ 10](#_Toc148516907)

[2.1 Постановка задачи 10](#_Toc148516908)

[2.2 Архитектура информационной системы 10](#_Toc148516909)

[2.3 Логическая модель базы данных 11](#_Toc148516910)

[2.4 Нормализация таблиц 11](#_Toc148516911)

[2.5 Описание таблиц 12](#_Toc148516912)

[2.6 Разработка приложения 14](#_Toc148516913)

[2.6.1 Решение главной задачи проекта 14](#_Toc148516914)

[2.6.2 Тестирование приложения 21](#_Toc148516915)

[2.6.3 Защита информационной системы 22](#_Toc148516916)

[2.7 Инструкция пользователю 22](#_Toc148516917)

[2.7.1 Общие сведения об информационной системе 23](#_Toc148516918)

[2.7.2 Требования к техническим средствам 24](#_Toc148516919)

[2.7.3 Требования к программным средствам 24](#_Toc148516920)

[2.7.4 Настройка информационной системы 25](#_Toc148516921)

[2.7.5 Формы ввода 26](#_Toc148516922)

[ЗАКЛЮЧЕНИЕ 28](#_Toc148516923)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 30](#_Toc148516924)

[ПРИЛОЖЕНИЕ A.SQL скрипты на создание и заполнение базы данных, 31](#_Toc148516925)

[ПРИЛОЖЕНИЕ Б. Код программы 31](#_Toc148516926)

# ВВЕДЕНИЕ

Курсовой проект посвящен разработке информационной системы приёмной комиссии колледжа.

Основная цель проекта – создание приложения, связанного с базой данных MS SQL Server, используя С# WPF, функционал которого обеспечивает автоматизацию работы приёмной комиссии колледжа.

База данных создаётся для создания, просмотра, поиска, редактирования и удаления заявлений абитуриентов. БД должна содержать данные о персональных данных абитуриентов, данных для входа в учетную запись сотрудника, текущих специальностях и количества мест на них, а также на различные формы финансирования.

Информационная система приёмной комиссии колледжа предназначена для сотрудников приёмной комиссии колледжа. Сотрудник имеет доступ к созданию заявлений, их поиску, редактированию, а также удалению.

Информационная система приёмной комиссии колледжа является актуальной и необходимой в наше время, потому что автоматизация позволяет ускорить и оптимизировать бизнес-процессы, такие как создание заявлений, просмотр их количества, редактирование заявлений, установка количества мест и просмотр проходного балла. Система автоматизации предоставляет сотрудникам простой способ создать заявление, без необходимости использовать бумажный носитель. Сотрудникам система предоставляет возможность легко просматривать заявления и вести учёт данных о абитуриентах. На достижение этих целей направлен курсовой проект.

При разработке приложения были рассмотрены существующие приложения в данной предметной области.

# ОБЩАЯ ЧАСТЬ

## Назначения и цели создания системы

Назначение: Система предназначена для автоматизации работы приёмной комиссии колледжа.

Цель: Реализация учебного плана. Получение знаний и умений в проектировании базы данных и написании приложения с использованием C# WPF.

## Обзор и анализ предметной области

База данных создаётся для информационного обслуживания абитуриентов, желающих подать заявление на поступление. База данных должна содержать персональные данные об абитуриентах, позволять их просматривать, редактировать, удалять.

В соответствии с предметной областью система строится с учётом следующих особенностей:

* Система авторизации сотрудника,
* возможность добавлять заявления,
* возможность просматривать заявления,
* возможность удалять заявления,
* возможность редактировать заявления,

Выделим базовые сущности этой предметной области:

**Сотрудники**. Атрибуты сотрудников – логин, пароль.

**Абитуриенты**. Атрибуты абитуриентов – идентификатор абитуриента, фамилия, имя, отчество, дата рождения

**Аттестаты**. Атрибуты аттестатов – идентификатор аттестата, идентификатор абитуриента, средний балл, место образования, уровень образования.

**Заявления.** Атрибуты заявлений – идентификатор заявления, идентификатор абитуриента, код специальности, учебный год.

**Специальности**. Атрибуты специальностей – код специальности, количество бюджетных мест, количество внебюджетных мест.

Система автоматизации работы приёмной комиссии колледжа предназначена для сотрудников. Сотрудники имеют возможность создавать заявления, просматривать их, редактировать, удалять.

Анализ существующих систем  показал, что основными функциями приложений в данной предметной области являются создание заявления, их просмотр, редактирование, удаление. Исходя из полученных данных, приложение будет разрабатываться с учётом перечисленных функций.

## Жизненный цикл базы данных

Жизненный цикл базы данных состоит из шести этапов:

1. Предварительное планирование

На этом этапе были определены цели и задачи, которые база данных должна решать, а также ресурсы, которые необходимы для ее создания и поддержания.

1. Проверка осуществимости

На этом этапе была проведена оценка возможности создания базы данных с учетом ресурсов, которые будут задействованы.

1. Определение требований

На этом этапе был произведён сбор информации о том, какую информацию необходимо хранить в базе данных, как она будет использоваться, кто будет работать с базой данных и какие функции должна включать разрабатываемая система.

1. Концептуальное проектирование

На этом этапе были созданы основные модели данных, для организации информации в базе данных. Также были разработаны схемы, отношения и между данными.

1. Реализация

На этом этапе было произведено создание структуры базы данных, написание кода для её реализации, создание пользовательского интерфейса и тестирование системы.

1. Оценка работы и поддержка базы данных.

На этом этапе была произведена оценка работы базы данных, были выявлены проблемы и их решение. Этот этап также включает в себя поддержание базы данных в актуальном состоянии и обучение пользователей ее использованию.

Выбор и характеристика СУБД

Для разработки информационной системы применялась концепция клиент-серверного взаимодействия, которая предусматривает существование серверов и клиентов, использующих определенные сервисы.

Архитектура клиент-сервер была выбрана потому, что клиенты и серверы могут работать на разных компьютерах в сети, что позволяет распределять нагрузку и обеспечивать более высокую доступность и производительность приложений.

Клиент-серверная архитектура позволяет масштабировать приложения по мере необходимости, добавляя дополнительные серверы или клиенты.

Разделение приложения на клиентскую и серверную части позволяет изменять или обновлять каждую часть независимо от другой.

Серверная часть приложения может обеспечивать более высокий уровень безопасности с помощью механизмов аутентификации, авторизации и шифрования данных.

Клиент-серверная архитектура позволяет централизованно управлять хранилищами данных, приложениями и доступом к ним.

Клиент-серверная архитектура позволяет оптимизировать производительность приложения, разделяя задачи между клиентом и сервером.

Клиент-серверная архитектура обеспечивает возможность добавления новых функций и модулей без необходимости изменения всего приложения.

Для реализации клиент-серверной архитектуры была выбрана СУБД MS SQL Server, так как к ней предоставляется бесплатный доступ. Также ПО MS SQL Server Management Studio для работы с MS SQL Server упрощает разработку, так как предоставляет удобный и интуитивно понятный интерфейс, а также имеет встроенный компилятор для кода на SQL языке. Сравнение SQL Server с другими СУБД представлено на Рисунке 1.



Рисунок 1. Сравнение СУБД

Выбор и характеристика среды разработки приложения

Для создания приложения была применена технология C# WPF ADO.NET, которая позволяет полностью контролировать запросы к базе данных. Эта технология довольно проста в использовании и предоставляет удобный способ создания подключения к базе данных, хранимой процедуры или текстового запроса. Кроме того, она поддерживает практически любую СУБД.

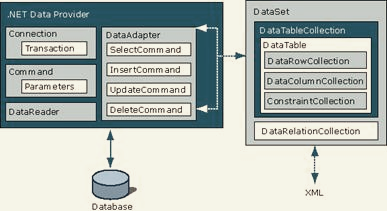


Рисунок 1.1. Схема работы ADO.NET

ADO.NET позволяет взаимодействовать с базами данных с помощью языка программирования C# и языка SQL

Плюсами ADO.NET являются:

1. Основной нужный функционал включен в ядро .NET Framework
2. Простота в использовании, для работы достаточно хорошо знать SQL язык
3. Стабильность и быстрота работы

Минусами ADO.NET являются:

1. Неудобство работы с объектами
2. Много повторяющихся блоков кода
3. Необходимость написания SQL-запросов

Entity Framework - инструмент ORM (Object-Relational Mapping), позволяющий взаимодействовать с базами данных, используя объектно-ориентированный подход.

Плюсами Entity Framework являются:

1. Простота в использовании и повышение уровня абстракции
2. Можно работать с данными на уровне объектов вместо работы непосредственно с запросами SQL

Минусами Entity Framework являются:

1. Большой объем данных, который нужно хранить в памяти при использовании внутренних объектов Entity Framework

Исходя из вышеперечисленного, выбор был сделан в пользу ADO.NET.

В качестве среды разработки была выбрана Microsoft Visual Studio, так как она предоставляет множество удобств для разработчика:

* 1. Интуитивно понятный интерфейс. Интерфейс Visual Studio был разработан, чтобы сделать работу программиста простой и удобной. Все инструменты и функции легко доступны благодаря простой структуре меню и панелей инструментов.
  2. Visual Studio содержит все необходимые инструменты и функции, такие как редактор кода, отладчик, компилятор и многие другие.
  3. Visual Studio имеет мощный отладчик, который позволяет находить и исправлять ошибки в коде. Он также поддерживает множество отладочных функций, таких как точки остановок, просмотр переменных и значения, трассировка стека вызовов и др.

Microsoft Visual Studio - это мощная и удобная интегрированная среда разработки, которая упрощает создание приложений и повышает продуктивность разработчика.

# СПЕЦИАЛЬНАЯ ЧАСТЬ

Постановка задачи

Средствами MS SQL SERVER спроектировать базу данных и создать приложение средствами С#, позволяющее, автоматизировать работу приёмной комиссии колледжа. Абитуриенты подают заявления, сотрудники вносят их в систему для дальнейшей работы.

Необходимо обеспечить ввод, редактирование и просмотр данных в удобной для сотрудника форме.

Система должна обеспечивать возможность добавления новых заявлений абитуриентов.

Обеспечить защиту базы данных и приложения.

Архитектура информационной системы

Архитектура информационной системы приёмной комиссии колледжа состоит из нескольких компонентов. В качестве основных компонентов можно выделить клиентское приложение C# WPF и базу данных MS SQL Server.

Сотруднику предоставляется приложение, которое обеспечивает возможность добавлять, просматривать, редактировать и удалять заявления абитуриентов. База данных содержит информацию о авторизационных данных сотрудников, абитуриентах, заявлениях, текущих специальностях и количества мест на них. База данных позволяет обеспечить быстрый и надежный доступ к информации для всех компонентов системы.

Архитектура информационной системы приёмной комиссии колледжа удобная в использовании и обеспечивает защиту данных. Она позволяет управлять заявлениями и изменять функционал в зависимости от текущих специальностей и количества мест на них.

Логическая модель базы данных

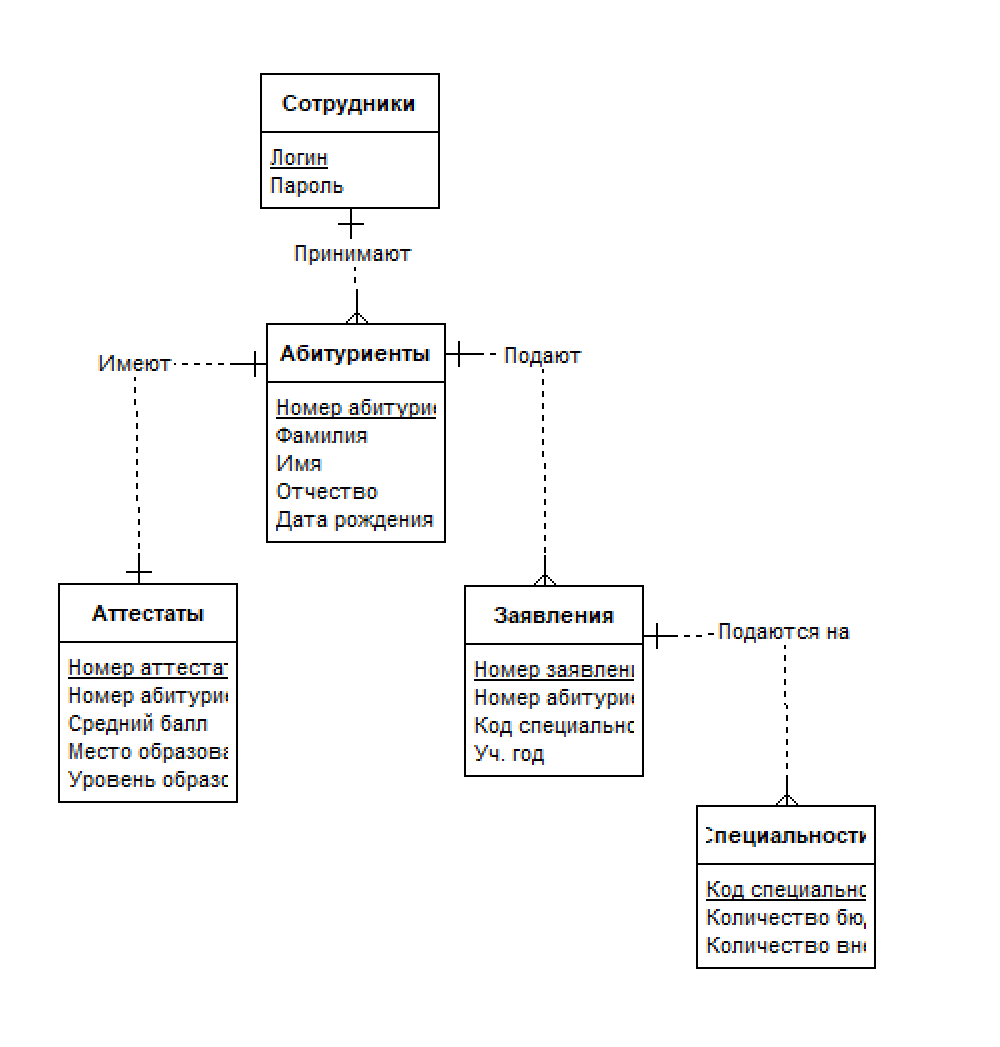
****

Рисунок 1.2. Логическая модель данных

Нормализация таблиц

Таблицы базы данных приведены к третьей нормальной форме. Данная форма предназначена для уменьшения дублирования данных, исключения аномалии данных, обеспечения ссылочной целостности и упрощения управления данными.

В третьей нормальной форме атрибуты таблиц зависят только от первичного ключа и все элементы таблиц неделимы.

Все таблицы в базе данных приведены к третьей нормальной форме так как:

1. Каждое поле содержит только одно значение и что все поля уникальны, вся информация разделена на более мелкие и более точные единицы, что обеспечивает более эффективное и точное управление базой данных. (Первая нормальная форма)
2. Каждое поле таблиц зависит от первичного ключа. Это позволяет обеспечить более эффективное управление базой данных, так как можно избежать дублирования информации и других аномалий данных. (Вторая нормальная форма)
3. Отсутствует транзитивная зависимость. Транзитивная зависимость - неключевые столбцы зависят от значений других неключевых столбцов. Это позволяет избежать аномалий данных. (Третья нормальная форма)

Описание таблиц

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1 – Employees (Сотрудники) | | |  |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| Login | Номер записи | nvarchar(50) | PK |
| Password | Стоимость | nvarchar(50) |  |
| Таблица 2 – Applicants (Абитуриенты) | | |  |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| Applicant\_ID | Номер абитуриента | int | PK, FK |
| LastName | Фамилия абитуриента | nvarchar(50) |  |
| FirstName | Имя абитуриента | nvarchar(50) |  |
| MiddleName | Отчество абитуриента | nvarchar(50) |  |
| dateOfBirth | Дата рождения абитуриента | datetime |  |
| Таблица 3 – Certificates (Аттестаты) | | |  |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| Certificate\_ID | Номер аттестата | nvarchar(50) | PK |
| Avarage\_Score | Средний балл | float |  |
| Place\_of\_education | Место получения | nvarchar(50) |  |
| Level\_of\_education | Уровень образования | nvarchar(50) |  |
| Applicant\_ID | Номер абитуриента | int | FK |

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 4 – Statements (Заявления) | | |  |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| Statement\_ID | Номер мастера | Int | PK |
| Applicant\_ID | Фамилия | int | FK |
| Academic\_year | Имя | date |  |
| Specialty\_Code | Отчество | nvarchar(50) |  |
| Таблица 5 – Specialties (Специальности) | | |  |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| Specialty\_Code | Код специальности | nvarchar(50) | PK, FK |
| Budget\_places | Количество бюджетных мест | int |  |
| Extra\_budgetary\_places | Количество внебюджетных мест | int |  |

Разработка приложения

На основе предметной области было разработано приложение. Создание приложения происходило в несколько этапов.

На первом этапе был определён функционал приложения и была изучена информация о среде разработки Microsoft Visual Studio, языке C#, системы WPF и ADO.NET.

В функционал приложения вошли:

* Возможность авторизации сотрудника
* Создание заявлений
* Просмотр заявлений
* Редактирование заявлений
* Удаление заявлений
* Изменение количества мест

На следующем этапе был разработан удобный и интуитивно понятный интерфейс, для легкого осуществления операций пользователями в системе.

После этого была начата работа над реализацией функционала приложения.

Для работы с базой данных были реализованы SQL ззапросы, которые отправляются из приложения.

На следующем этапе было проведено тестирование приложение и отладка.

Решение главной задачи проекта

Главная задача проекта по созданию информационной системы приёмной комиссии колледжа состоит в создании приложения, которое будет позволять создавать, просматривать, редактировать, удалять заявления.

Подзадачи, которые необходимо решить для реализации главной задачи проекта:

1. Реализовать возможность авторизации сотрудника

SQL: SELECT \* FROM Employees  
C#: public bool Auth(string login, string password)

{

DataBase db = new();

SqlCommand command\_1 = new SqlCommand("SELECT \* FROM Employees", db.connection);

SqlDataReader reader\_1 = command\_1.ExecuteReader();

bool auth\_flag = false;

while (reader\_1.Read())

{

if (login == reader\_1["Login"].ToString() && password == reader\_1["Password"].ToString())

{

auth\_flag = true;

break;

}

}

if (!auth\_flag)

{

MessageBox.Show("Аккаунт не существует");

}

return auth\_flag;

}

1. Реализовать возможность добавлять заявления

SQL: INSERT INTO Applicants VALUES ('{numberForApplicant}','{lastName}', '{firstName}', '{middleName}', '{dateOfBirth}')

INSERT INTO Certificates VALUES ('{sertificateId}','{numberForApplicant}', '{avarageScore}', '{placeOfEducation}', '{levelOfEducation}')

INSERT INTO Statements VALUES ('{numberForStatement}','{numberForApplicant}', '{specialtyCode}', '{academicYear}')

C#: public void SaveButton(object sender, RoutedEventArgs e)

{

string

lastName = lastNameTextBox.Text,

firstName = firstNameTextBox.Text,

middleName = middleNameTextBox.Text,

dateOfBirth = dateOfBirthTextBox.Text,

sertificateId = certificateIdTextBox.Text,

placeOfEducation = placeOfEducationTextBox.Text,

levelOfEducation = levelofEducationTextBox.Text,

avarageScore = avarageScoreTextBox.Text,

specialtyCode = specialtyCodeTextBox.Text,

academicYear = academicYearTextBox.Text;

bool flagForCorrectSpecialtyCode = false;

SqlCommand commandForCheckSpecialtyCodes = new SqlCommand($"SELECT Specialty\_Code FROM Specialties", db.connection);

SqlDataReader readerForSpecialtiesCodes = commandForCheckSpecialtyCodes.ExecuteReader();

while (readerForSpecialtiesCodes.Read())

{

if (specialtyCode == readerForSpecialtiesCodes["Specialty\_Code"].ToString())

{

flagForCorrectSpecialtyCode = true;

break;

}

}

readerForSpecialtiesCodes.Close();

string datePattern = @"(0?[1-9]|[12][0-9]|3[01]).(0?[1-9]|1[012]).((19|20)\d\d)";

if (

lastName == "" || firstName == "" ||

!Regex.IsMatch(dateOfBirth, datePattern) || sertificateId == "" ||

placeOfEducation == "" || levelOfEducation == "" ||

!Regex.IsMatch(academicYear, datePattern) || !flagForCorrectSpecialtyCode

)

{

MessageBox.Show("Введенны некорректные данные");

}

else

{

int numberForApplicant = SearchFreeNumberForApplicantNumber();

int numberForStatement = SearchFreeNumberForStatementNumber();

if (dialog.FileName != "")

{

File.Copy(dialog.FileName, System.Environment.CurrentDirectory + $"/img/{numberForStatement}.jpg");

}

SqlCommand commandInsertApplicant = new SqlCommand($"INSERT INTO Applicants VALUES ('{numberForApplicant}','{lastName}', '{firstName}', '{middleName}', '{dateOfBirth}')", db.connection);

SqlCommand commandInsertCertificate = new SqlCommand($"INSERT INTO Certificates VALUES ('{sertificateId}','{numberForApplicant}', '{avarageScore}', '{placeOfEducation}', '{levelOfEducation}')", db.connection);

SqlCommand commandInsertStatement = new SqlCommand($"INSERT INTO Statements VALUES ('{numberForStatement}','{numberForApplicant}', '{specialtyCode}', '{academicYear}')", db.connection);

commandInsertApplicant.ExecuteNonQuery();

commandInsertCertificate.ExecuteNonQuery();

commandInsertStatement.ExecuteNonQuery();

HomeWindow homeWindow = new();

Close();

homeWindow.ShowDialog();

}

}

1. Реализовать возможность просматривать заявления

SQL: SELECT LastName, FirstName, MiddleName, dateOfBirth, Specialty\_Code, Avarage\_Score, Academic\_year, Statement\_ID FROM Applicants INNER JOIN Statements ON Applicants.Applicant\_ID = Statements.Applicant\_ID INNER JOIN Certificates ON Applicants.Applicant\_ID = Certificates.Applicant\_ID

C#: private void GetData(object sender, RoutedEventArgs e)

{

DataBase db = new();

SqlCommand command\_GetData = new SqlCommand("SELECT LastName, FirstName, MiddleName, dateOfBirth, Specialty\_Code, Avarage\_Score, Academic\_year, Statement\_ID FROM Applicants INNER JOIN Statements ON Applicants.Applicant\_ID = Statements.Applicant\_ID INNER JOIN Certificates ON Applicants.Applicant\_ID = Certificates.Applicant\_ID", db.connection);

SqlDataReader reader\_GetData = command\_GetData.ExecuteReader();

model = new BindingList<HomePageDataModel>() { };

while (reader\_GetData.Read())

{

model.Add(new HomePageDataModel()

{

lastName = reader\_GetData["LastName"].ToString(),

firstName = reader\_GetData["FirstName"].ToString(),

middleName = reader\_GetData["MiddleName"].ToString(),

dateOfBirth = reader\_GetData["dateOfBirth"]?.ToString()?.TrimEnd('0', ':', '0', '0', ':', '0'),

specialtyCode = reader\_GetData["Specialty\_Code"].ToString(),

averageScore = (double)reader\_GetData["Avarage\_Score"],

dateOfStatement = reader\_GetData["Academic\_year"].ToString()?.TrimEnd('0', ':', '0', '0', ':', '0'),

numberOfStatement = (int)reader\_GetData["Statement\_ID"],

} );

}

reader\_GetData.Close();

var sorted\_model = model.OrderByDescending(x => x.averageScore).ToList();

HomeDataGrid.ItemsSource = sorted\_model;

countOfStatementsLabel.Content = $"Количество заявлений: {sorted\_model.Count}";

}

1. Реализовать возможность редактировать заявления

SQL: UPDATE Applicants SET LastName='{lastName}', FirstName='{firstName}', MiddleName='{middleName}', dateOfBirth='{dateOfBirth}' WHERE Applicant\_ID='{currentApplicantId} UPDATE Certificates SET Certificate\_ID='{sertificateId}', Avarage\_Score='{avarageScore}', Place\_of\_education='{placeOfEducation}', Level\_of\_education='{levelOfEducation}' WHERE Applicant\_ID='{currentApplicantId}' UPDATE Statements SET Specialty\_Code='{specialtyCode}', Academic\_year='{academicYear}' WHERE Applicant\_ID='{currentApplicantId}'

C#: private void SaveStatementChangeButton(object sender, RoutedEventArgs e)

{

int currentApplicantId = 0;

string

lastName = lastNameTextBox.Text,

firstName = firstNameTextBox.Text,

middleName = middleNameTextBox.Text,

dateOfBirth = dateOfBirthTextBox.Text,

sertificateId = certificateIdTextBox.Text,

placeOfEducation = placeOfEducationTextBox.Text,

levelOfEducation = levelofEducationTextBox.Text,

specialtyCode = specialtyCodeTextBox.Text,

academicYear = academicYearTextBox.Text,

avarageScore = avarageScoreTextBox.Text,

datePattern = @"(0?[1-9]|[12][0-9]|3[01]).(0?[1-9]|1[012]).((19|20)\d\d)";

string str = (string)numberOsStatementLabel.Content;

string currentNumber = new string(str.Where(t => char.IsDigit(t)).ToArray());

SqlCommand selectCurrentApplicantIdCommand = new SqlCommand($"SELECT Applicant\_ID FROM Statements WHERE Statement\_ID = {currentNumber}", db.connection);

SqlDataReader selectCurrentApplicantIdReader = selectCurrentApplicantIdCommand.ExecuteReader();

while (selectCurrentApplicantIdReader.Read())

{

currentApplicantId = (int)selectCurrentApplicantIdReader["Applicant\_ID"];

}

selectCurrentApplicantIdReader.Close();

if (

lastName == "" || firstName == "" ||

!Regex.IsMatch(dateOfBirth, datePattern) || sertificateId == "" ||

placeOfEducation == "" || levelOfEducation == "" ||

!Regex.IsMatch(academicYear, datePattern)

)

{

MessageBox.Show("Введенны некорректные данные");

}

else

{

if (dialog.FileName != "" && !File.Exists(dialog.FileName))

{

File.Copy(dialog.FileName, System.Environment.CurrentDirectory + $"/img/{numberOsStatementLabel}.jpg");

}

SqlCommand commandUpdateApplicant = new SqlCommand($"UPDATE Applicants SET LastName='{lastName}', FirstName='{firstName}', MiddleName='{middleName}', dateOfBirth='{dateOfBirth}' WHERE Applicant\_ID='{currentApplicantId}'", db.connection);

SqlCommand commandUpdateCertificate = new SqlCommand($"UPDATE Certificates SET Certificate\_ID='{sertificateId}', Avarage\_Score='{avarageScore}', Place\_of\_education='{placeOfEducation}', Level\_of\_education='{levelOfEducation}' WHERE Applicant\_ID='{currentApplicantId}'", db.connection);

SqlCommand commandUpdateStatement = new SqlCommand($"UPDATE Statements SET Specialty\_Code='{specialtyCode}', Academic\_year='{academicYear}' WHERE Applicant\_ID='{currentApplicantId}'", db.connection);

commandUpdateApplicant.ExecuteNonQuery();

commandUpdateCertificate.ExecuteNonQuery();

commandUpdateStatement.ExecuteNonQuery();

HomeWindow homeWindow = new();

Close();

homeWindow.ShowDialog();

}

}

1. Реализовать возможность удалять заявления

SQL: DELETE FROM Certificates WHERE Applicant\_ID = {currentApplicantId}; DELETE FROM Statements WHERE Applicant\_ID = {currentApplicantId}; DELETE FROM Applicants WHERE Applicant\_ID = {currentApplicantId}

C#: private void DeleteStatement\_Button(object sender, RoutedEventArgs e)

{

var result = MessageBox.Show("Вы точно хотите удалить запись?", "", MessageBoxButton.YesNo, MessageBoxImage.Warning);

int currentApplicantId = 0;

if (result == MessageBoxResult.Yes)

{

string str = (string)numberOsStatementLabel.Content;

string currentNumber = new string(str.Where(t => char.IsDigit(t)).ToArray());

SqlCommand selectCurrentApplicantIdCommand = new SqlCommand($"SELECT Applicant\_ID FROM Statements WHERE Statement\_ID = {currentNumber}", db.connection);

SqlDataReader selectCurrentApplicantIdReader = selectCurrentApplicantIdCommand.ExecuteReader();

while (selectCurrentApplicantIdReader.Read())

{

currentApplicantId = (int)selectCurrentApplicantIdReader["Applicant\_ID"];

}

selectCurrentApplicantIdReader.Close();

SqlCommand deleteStatementCommand = new SqlCommand($"DELETE FROM Certificates WHERE Applicant\_ID = {currentApplicantId}; DELETE FROM Statements WHERE Applicant\_ID = {currentApplicantId}; DELETE FROM Applicants WHERE Applicant\_ID = {currentApplicantId}", db.connection);

deleteStatementCommand.ExecuteNonQuery();

if (File.Exists(System.Environment.CurrentDirectory + $"/img/{currentNumber}.jpg")) File.Delete(System.Environment.CurrentDirectory + $"/img/{currentNumber}.jpg");

HomeWindow homeWindow = new HomeWindow();

Close();

homeWindow.ShowDialog();

}

}

Тестирование приложения

Таблица 6 – Метод проверки требований к приложению

|  |  |  |  |
| --- | --- | --- | --- |
| № | Действие | Результат | Ожидаемый результат |
| 1 | Запуск приложения | Приложение подключается к базе данных | Приложение подключается базе данных |
| 2 | Сотрудник нажимает кнопку “Войти» | Производится вход и из БД загружаются все заявления в datagrid | Производится вход и из БД загружаются все заявления из datagrid |
| 3 | Сотрудник нажимает кнопку «Добавить» | Открывается окно добавления | Открывается окно добавления |
| 4 | Сотрудник заполняет поля при добавлении заявления | Данные вводятся с учетом ограничений на тип данных и длину | Система не дает ввести некорректные данные |
| 5 | Сотруднике сохраняет заполненное заявление | Заявление сохранено в базу данных | Заявление сохранено в базу данных |
| 6 | Сотрудник нажимает кнопку поиска заявления | Появляется окно для поиска заявления | Появляется окно для поиска заявления |
| 7 | Сотрудник вводит номер заявления и нажимает кнопку | Открывается искомое заявление | Открывается искомое заявление |
| 8 | Сотрудник нажимает кнопку «удалить» при редактировании заявления | Открывается окно подтверждение, при подтверждении запись удаляется, при отмене остается | Открывается окно подтверждение, при подтверждении запись удаляется, при отмене остается |
| 9 | Сотрудник нажимает кнопку «сохранить» во время редактирования заявления | Редактированное заявление сохраняется в базу данных | Редактированное заявление сохраняется в базу данных |

Защита информационной системы

В качестве механизмов обеспечения безопасности в базах данных использовались различные средства защиты в MS SQL Server и ADO.NET.

Использованные средства:

Аутентификация и авторизация: SQL Server предоставляет возможность настройки аутентификации пользователей, имеющих доступ к базам данных, а также настройки их прав на чтение, запись и изменение данных в таблицах.

Аутентификация и авторизация были реализованы с помощью контроля логинов и паролей.

Защита от инъекций: Один из способов атаки на базы данных - это инъекция SQL-кода, которая позволяет получить доступ к базе данных. Для устранения данной уязвимости использовались параметризованные запросы и предварительная валидации данных программным путём.

Использование параметров команд: ADO.NET позволяет использовать параметры команд для задания параметров запросов к базе данных, что помогает защитить данные от инъекций SQL-кода и предотвратить ошибки при работе с данными.

Инструкция пользователю

Инструкция для клиента:

1. Откройте приложение.
2. Введите логин и пароль
3. Нажмите кнопку «Войти»
4. Для добавления заявления нажмите кнопку «Добавить заявление»
5. Для поиска заявления нажмите кнопку «Просмотр, редактирование, удаление», введите в появившемся окне номер заявления и нажмите кнопку «Найти»
6. Для редактирования заявления нажмите кнопку «Просмотр, редактирование, удаление», введите в появившемся окне номер заявления и нажмите кнопку «Найти», внесите необходимые изменения, нажмите кнопку «сохранить».
7. Для удаления заявления нажмите кнопку «Просмотр, редактирование, удаление», введите в появившемся окне номер заявления и нажмите кнопку «Найти», далее нажмите кнопку «удалить» и подтвердите свое действие во всплывающем окне.

8. Для изменения количества мест по специальности с определенной формой финансирования нажмите кнопку «Изменить количество мест».

9. Для выхода из учетной записи нажмите кнопку «Выйти» на главной странице.

10. Для открытия справки по использованию нажмите кнопку «Справка» на главной странице.

11. Для фильтрации заполните необходимые поля на главной странице и нажмите кнопку «Обновить».

12. Для сохранения отчета об абитуриентах с проходным баллом нажмите «Сохранить отчёт».

Общие сведения об информационной системе

Информационная система приёмной комиссии колледжа – это комплект программных средств, которые помогают упорядочить и ускорить процессы, связанные с созданием заявлений, учетом поступающих абитуриентов.

Основными элементами информационной системы приёмной комиссии колледжа являются:

1. Компьютер с установленным на него соответствующим программным обеспечением, предназначенным для учета поступающих абитуриентов и работы с их заявлениями.
2. Приложение, написанное на C# WPF с использованием технологии ADO.NET, которое реализует подключение к базе данных системы и позволяет получать из неё данные и изменять их.
3. База данных MS SQL Server, которая хранит в себе данные, нужные для автоматизации работы системы.

Информационная система приёмной комиссии колледжа позволяет повысить эффективность работы, что позволяет экономить время и ресурсы, а также оптимизировать работу сотрудников. Также, система упрощает работу и обработку данных.

Требования к техническим средствам

Для корректной работы приложения требуется соблюдать данные технические характеристики:

1. Минимальный объём оперативной памяти: 2 ГБ
2. Минимальные требования к монитору: Super VGA с разрешением 800x600 пикселей или более высоким.
3. Свободное место на диске: 30 МБ (Без установки SQL Server)
4. Доступ в Интернет
5. Тип процессора: AMD Opteron, AMD Athlon 64, Intel Xeon с поддержкой Intel EM64T, Intel Pentium IV с поддержкой EM64T.
6. Быстродействие процессора: частота 2,0 ГГц и выше

Требования к программным средствам

Для корректной работы приложения, программное обеспечение должно соответствовать данному списку:

1. Операционная система: Windows 10 или Windows 11
2. MS SQL Server Management Studio
3. База данных в MS SQL Server
4. .NET Framework
5. Последние драйверы для системы

Настройка информационной системы

Для интеграции информационной системы приёмной комиссии колледжа в работу понадобится:

1. Проверить, что система соответствует минимальным системным требованиям
2. Установить всё необходимое программное обеспечение
3. Настроить MS SQL Server
4. Выполнить скрипты на создание базы данных
5. Заменить строку подключения в приложении
6. Раздать сотрудникам логины и пароли

Формы ввода

В системе использовались следующие формы ввода:

1. Окно авторизации

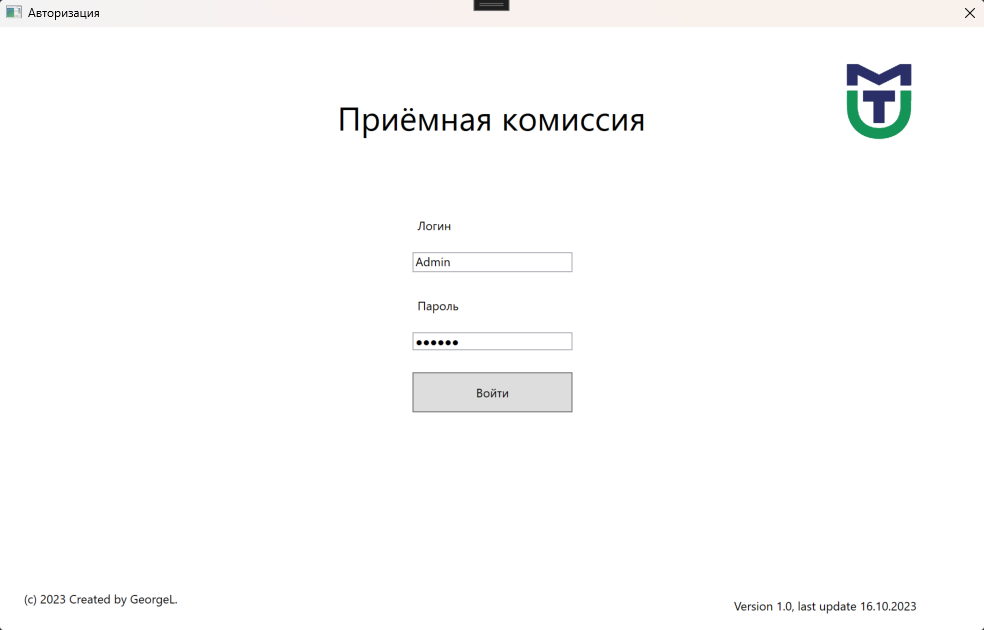


Рисунок 1.3 – Авторизация

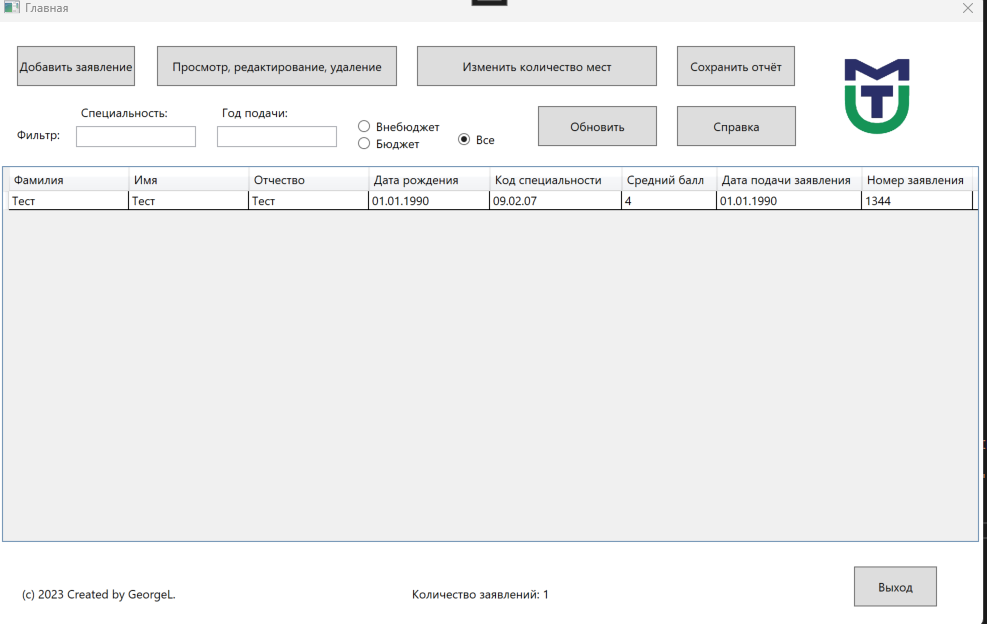
1. Фильтрация в главном окне 

Рисунок 1.4 – Главное окно

1. Добавление заявления

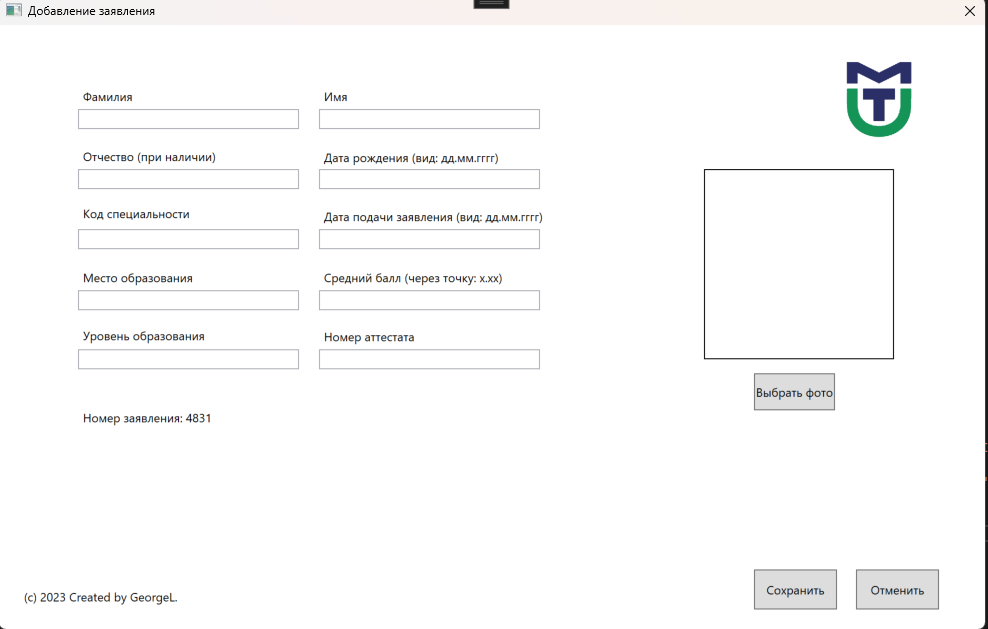


Рисунок 1.5 – Добавление заявления

1. Поиск заявления

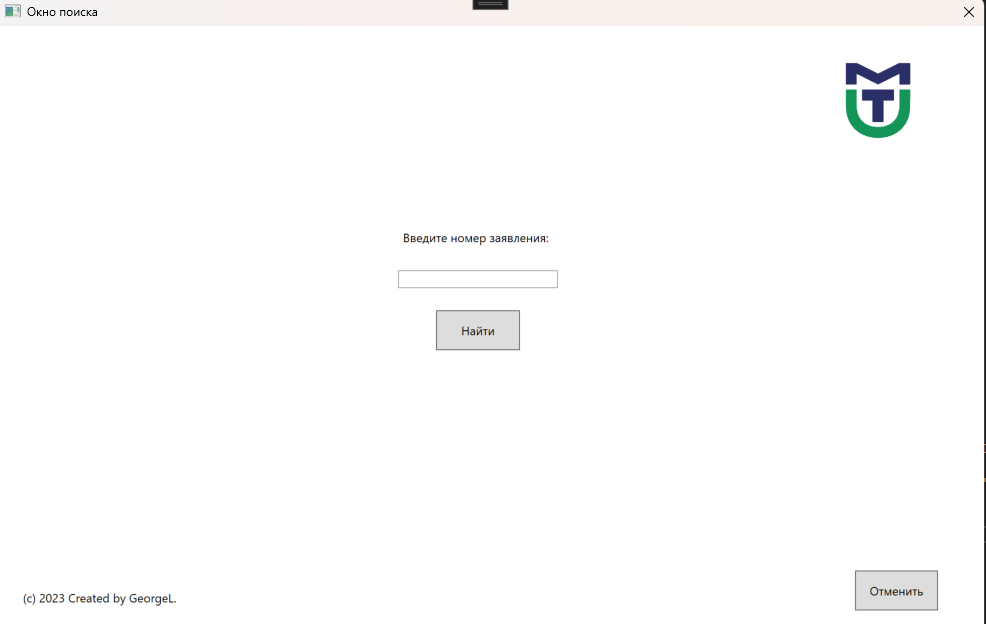


Рисунок 1.6 – Поиск заявления

1. Просмотр, редактирование, удаление заявления

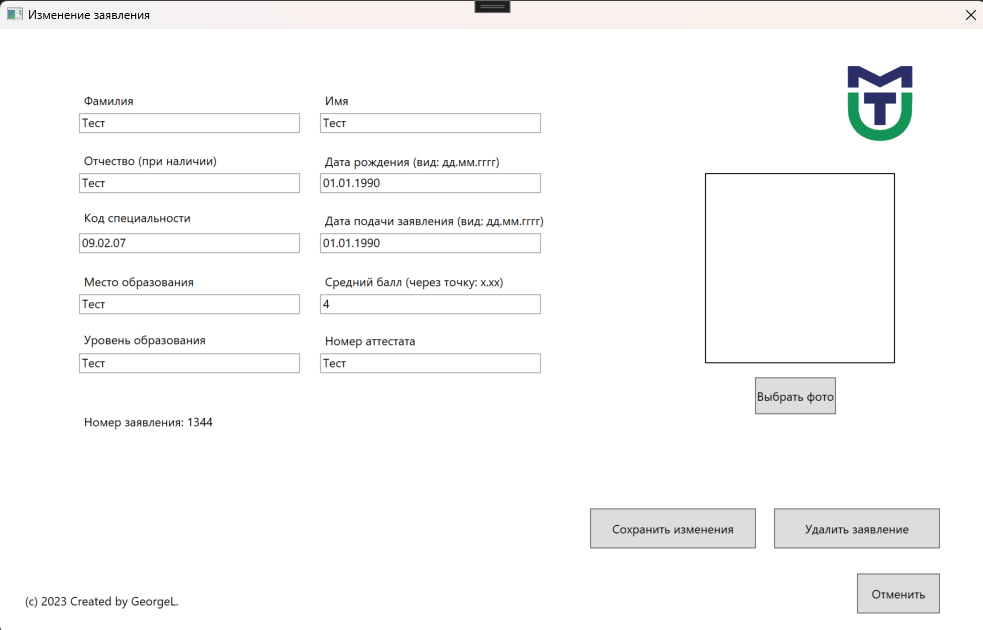


Рисунок 1.7 – Просмотр, редактирование, удаление заявления

# ЗАКЛЮЧЕНИЕ

Курсовой проект посвящен разработке информационной системы приёмной комиссии колледжа. В ходе его выполнения были изучены основные принципы работы приёмной комиссии колледжа и принципы работы с C# WPF, ADO.NET, интеграции базы данных MS SQL Server в приложение и особенности языка SQL. Также были проанализированы существующие решения в данной предметной области.

Была спроектирована база данных MS SQL Server и разработано приложение c использованием C# WPF и ADO.NET, которое позволяет упростить процессы создания заявления абитуриентов и их учёта.

Основной целью проекта было изучение языка C#, возможностей работы MS SQL Server и ADO.NET, а также способов увеличения эффективности работы приёмной комиссии колледжа, что было достигнуто благодаря автоматизации рутинных процессов при помощи интеграции базы данных и программного кода на C#. Были также изучены методы тестирования и защиты приложения.

Разработанное приложение имеет большой потенциал для использования и может быть усовершенствовано в дальнейшем.

Дальнейшее развитие темы курсового проекта предполагает расширение функционала приложения, например, возможность работать с заявлениями из разных подразделений. Также, предполагается дальнейшее изучение C# WPF и возможностей MS SQL Server.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Заполнить DataGridView – URL: https://www.cyberforum.ru/windows-forms/thread1988785.html

2. Beginners guide to accessing SQL Server through C# - URL: https://www.codeproject.com/Articles/4416/Beginners-guide-to-accessing-SQL-Server-through-C

3. WPF Calendar disable date selection - URL: https://stackoverflow.com/questions/56188607/wpf-calendar-disable-date-selection

4. ADO.NET. - URL: https://ru.wikipedia.org/wiki/ADO.NET

5. Сохранение изображения в БД C# – URL: https://habr.com/ru/articles/700406/

6. Сохранение и извлечение файлов из базы данных – URL: https://metanit.com/sharp/adonetcore/2.13.php

# ПРИЛОЖЕНИЕ A.SQL скрипты на создание и заполнение базы данных,

# ПРИЛОЖЕНИЕ Б. Код программы

AddingAStatementWindow.xaml

<Window x:Class="Commission.AddingAStatementWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Commission"

mc:Ignorable="d"

Title="Добавление заявления" Height="640" Width="1000" ResizeMode="NoResize" WindowStartupLocation="CenterScreen">

<Grid RenderTransformOrigin="0.51,0.503">

<Label Content="(с) 2023 Created by GeorgeL." HorizontalAlignment="Left" Margin="20,558,0,0" VerticalAlignment="Top" Width="179"/>

<Button Content="Отменить" HorizontalAlignment="Left" Margin="857,544,0,0" VerticalAlignment="Top" Height="40" Width="83" Click="Cansel\_button"/>

<Image HorizontalAlignment="Left" Height="102" Margin="835,24,0,0" VerticalAlignment="Top" Width="90" Source="/MGUTU.jpg"/>

<Button Content="Сохранить" HorizontalAlignment="Left" Margin="755,544,0,0" VerticalAlignment="Top" Height="40" Width="83" Click="SaveButton"/>

<Label x:Name="numberOsStatementLabel" Content="[nubmerOfStatement]" HorizontalAlignment="Left" Margin="79,379,0,0" VerticalAlignment="Top" Width="205"/>

<TextBox MaxLength="34" x:Name="lastNameTextBox" HorizontalAlignment="Left" Margin="79,84,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="34" x:Name="middleNameTextBox" HorizontalAlignment="Left" Margin="79,144,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<Label Content="Фамилия" HorizontalAlignment="Left" Margin="79,58,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Имя" HorizontalAlignment="Left" Margin="320,58,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Отчество (при наличии)" HorizontalAlignment="Left" Margin="79,118,0,0" VerticalAlignment="Top" Width="221"/>

<TextBox MaxLength="34" x:Name="placeOfEducationTextBox" HorizontalAlignment="Left" Margin="79,265,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="34" x:Name="firstNameTextBox" HorizontalAlignment="Left" Margin="320,84,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="4" x:Name="avarageScoreTextBox" HorizontalAlignment="Left" Margin="320,265,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<Label Content="Дата рождения (вид: дд.мм.гггг)" HorizontalAlignment="Left" Margin="320,119,0,0" VerticalAlignment="Top" Width="230"/>

<Label Content="Код специальности" HorizontalAlignment="Left" Margin="79,175,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Средний балл (через точку: x.xx)" HorizontalAlignment="Left" Margin="320,239,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Место образования" HorizontalAlignment="Left" Margin="79,239,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Уровень образования" HorizontalAlignment="Left" Margin="79,297,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Номер аттестата" HorizontalAlignment="Left" Margin="320,298,0,0" VerticalAlignment="Top" Width="221"/>

<TextBox MaxLength="34" x:Name="certificateIdTextBox" HorizontalAlignment="Left" Margin="320,324,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<Label Content="Дата подачи заявления (вид: дд.мм.гггг)" HorizontalAlignment="Left" Margin="320,178,0,0" VerticalAlignment="Top" Width="230"/>

<Button Content="Выбрать фото" HorizontalAlignment="Left" Margin="755,348,0,0" VerticalAlignment="Top" Height="37" Width="81" Click="choosePhoto"/>

<Image x:Name="ApplicantImage" HorizontalAlignment="Left" Height="190" Margin="705,144,0,0" VerticalAlignment="Top" Width="190" RenderTransformOrigin="0.5,0.5"/>

<Rectangle HorizontalAlignment="Left" Height="190" Margin="705,144,0,0" Stroke="Black" VerticalAlignment="Top" Width="190"/>

<TextBox MaxLength="8" x:Name="specialtyCodeTextBox" HorizontalAlignment="Left" Margin="79,204,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="34" x:Name="levelofEducationTextBox" HorizontalAlignment="Left" Margin="79,324,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="10" x:Name="dateOfBirthTextBox" HorizontalAlignment="Left" Margin="320,144,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="10" x:Name="academicYearTextBox" HorizontalAlignment="Left" Margin="320,204,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

</Grid>

</Window>

AddingAStatementWindow.xaml.cs

using Microsoft.SqlServer.Server;

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Commission

{

/// <summary>

/// Логика взаимодействия для AddingAStatementWindow.xaml

/// </summary>

public partial class AddingAStatementWindow : Window

{

OpenFileDialog dialog = new OpenFileDialog();

DataBase db = new();

public AddingAStatementWindow()

{

InitializeComponent();

SearchFreeNumberForStatementNumber();

}

/// <summary>

/// Кнопка отмены изменений и возврата в главное окно

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Cansel\_button(object sender, RoutedEventArgs e)

{

HomeWindow homeWindow = new();

Close();

homeWindow.ShowDialog();

}

/// <summary>

/// Кнопка выбора фотографии

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void choosePhoto(object sender, RoutedEventArgs e)

{

dialog.Filter = "Файлы рисунков (\*.png, \*.jpg)|\*.png;\*.jpg";

dialog.ShowDialog();

if (dialog.FileName != "") ApplicantImage.Source = new BitmapImage(new Uri(dialog.FileName));

}

/// <summary>

/// Кнопка сохранения данных в БД с их предварительной проверкой

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void SaveButton(object sender, RoutedEventArgs e)

{

string

lastName = lastNameTextBox.Text,

firstName = firstNameTextBox.Text,

middleName = middleNameTextBox.Text,

dateOfBirth = dateOfBirthTextBox.Text,

sertificateId = certificateIdTextBox.Text,

placeOfEducation = placeOfEducationTextBox.Text,

levelOfEducation = levelofEducationTextBox.Text,

avarageScore = avarageScoreTextBox.Text,

specialtyCode = specialtyCodeTextBox.Text,

academicYear = academicYearTextBox.Text;

bool flagForCorrectSpecialtyCode = false;

SqlCommand commandForCheckSpecialtyCodes = new SqlCommand($"SELECT Specialty\_Code FROM Specialties", db.connection);

SqlDataReader readerForSpecialtiesCodes = commandForCheckSpecialtyCodes.ExecuteReader();

while (readerForSpecialtiesCodes.Read())

{

if (specialtyCode == readerForSpecialtiesCodes["Specialty\_Code"].ToString())

{

flagForCorrectSpecialtyCode = true;

break;

}

}

readerForSpecialtiesCodes.Close();

string datePattern = @"(0?[1-9]|[12][0-9]|3[01]).(0?[1-9]|1[012]).((19|20)\d\d)";

if (

lastName == "" || firstName == "" ||

!Regex.IsMatch(dateOfBirth, datePattern) || sertificateId == "" ||

placeOfEducation == "" || levelOfEducation == "" ||

!Regex.IsMatch(academicYear, datePattern) || !flagForCorrectSpecialtyCode

)

{

MessageBox.Show("Введенны некорректные данные");

}

else

{

int numberForApplicant = SearchFreeNumberForApplicantNumber();

int numberForStatement = SearchFreeNumberForStatementNumber();

if (dialog.FileName != "")

{

File.Copy(dialog.FileName, System.Environment.CurrentDirectory + $"/img/{numberForStatement}.jpg");

}

SqlCommand commandInsertApplicant = new SqlCommand($"INSERT INTO Applicants VALUES ('{numberForApplicant}','{lastName}', '{firstName}', '{middleName}', '{dateOfBirth}')", db.connection);

SqlCommand commandInsertCertificate = new SqlCommand($"INSERT INTO Certificates VALUES ('{sertificateId}','{numberForApplicant}', '{avarageScore}', '{placeOfEducation}', '{levelOfEducation}')", db.connection);

SqlCommand commandInsertStatement = new SqlCommand($"INSERT INTO Statements VALUES ('{numberForStatement}','{numberForApplicant}', '{specialtyCode}', '{academicYear}')", db.connection);

commandInsertApplicant.ExecuteNonQuery();

commandInsertCertificate.ExecuteNonQuery();

commandInsertStatement.ExecuteNonQuery();

HomeWindow homeWindow = new();

Close();

homeWindow.ShowDialog();

}

}

/// <summary>

/// Метод ищущий свободный номер в базе данных для номера заявления

/// </summary>

public int SearchFreeNumberForStatementNumber()

{

Random rnd = new Random();

int number = rnd.Next(1000, 9999);

SqlCommand command\_1 = new SqlCommand("SELECT Statement\_ID FROM Statements", db.connection);

SqlDataReader reader\_1 = command\_1.ExecuteReader();

while (reader\_1.Read())

{

if (number == (int)reader\_1["Statement\_ID"])

{

SearchFreeNumberForStatementNumber();

}

}

numberOsStatementLabel.Content = $"Номер заявления: {number}";

reader\_1.Close();

return number;

}

/// <summary>

/// Метод ищущий свободный номер в базе данных для абитуриента

/// </summary>

public int SearchFreeNumberForApplicantNumber()

{

Random rnd = new Random();

int number = rnd.Next(1000, 9999);

SqlCommand command\_1 = new SqlCommand("SELECT Applicant\_ID FROM Applicants", db.connection);

SqlDataReader reader\_1 = command\_1.ExecuteReader();

while (reader\_1.Read())

{

if (number == (int)reader\_1["Applicant\_ID"])

{

SearchFreeNumberForApplicantNumber();

}

}

reader\_1.Close();

return number;

}

}

}

Authorization.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

namespace Commission

{

/// <summary>

/// Класс необходимый для процесса авторизации пользователя

/// </summary>

internal class Authorization

{

/// <summary>

/// Метод авторизации

/// </summary>

/// <param name="login">Введённый логин в textbox в окне авторизации</param>

/// <param name="password">Введённый пароль в textbox в окне авторизации</param>

/// <returns>Флаг успешности авторизации</returns>

public bool Auth(string login, string password)

{

DataBase db = new();

SqlCommand command\_1 = new SqlCommand("SELECT \* FROM Employees", db.connection);

SqlDataReader reader\_1 = command\_1.ExecuteReader();

bool auth\_flag = false;

while (reader\_1.Read())

{

if (login == reader\_1["Login"].ToString() && password == reader\_1["Password"].ToString())

{

auth\_flag = true;

break;

}

}

if (!auth\_flag)

{

MessageBox.Show("Аккаунт не существует");

}

return auth\_flag;

}

}

}

ChangeStatementWindow.xaml.cs

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.IO;

using System.Text.RegularExpressions;

namespace Commission

{

/// <summary>

/// Логика взаимодействия для ChangeStatementWindow.xaml

/// </summary>

public partial class ChangeStatementWindow : Window

{

DataBase db = new();

OpenFileDialog dialog = new OpenFileDialog();

public ChangeStatementWindow(int currentStatementNumber)

{

InitializeComponent();

SqlCommand choiseCurrentStatement = new SqlCommand($"SELECT LastName, FirstName, MiddleName, dateOfBirth, Specialty\_Code, Academic\_year, Place\_of\_education, Avarage\_Score, Level\_of\_education, Certificate\_ID, Statement\_ID FROM Applicants INNER JOIN Statements ON Applicants.Applicant\_ID = Statements.Applicant\_ID INNER JOIN Certificates ON Applicants.Applicant\_ID = Certificates.Applicant\_ID WHERE Statement\_ID = {currentStatementNumber}", db.connection);

SqlDataReader choiseCurrentStatement\_reader = choiseCurrentStatement.ExecuteReader();

while (choiseCurrentStatement\_reader.Read())

{

lastNameTextBox.Text = choiseCurrentStatement\_reader["LastName"].ToString();

firstNameTextBox.Text = choiseCurrentStatement\_reader["FirstName"].ToString();

middleNameTextBox.Text = choiseCurrentStatement\_reader["MiddleName"].ToString();

dateOfBirthTextBox.Text = choiseCurrentStatement\_reader["dateOfBirth"].ToString()?.TrimEnd('0', ':', '0', '0', ':', '0');

specialtyCodeTextBox.Text = choiseCurrentStatement\_reader["Specialty\_Code"].ToString();

academicYearTextBox.Text = choiseCurrentStatement\_reader["Academic\_Year"].ToString()?.TrimEnd('0', ':', '0', '0', ':', '0');

placeOfEducationTextBox.Text = choiseCurrentStatement\_reader["Place\_of\_education"].ToString();

avarageScoreTextBox.Text = choiseCurrentStatement\_reader["Avarage\_Score"].ToString();

levelofEducationTextBox.Text = choiseCurrentStatement\_reader["Level\_of\_education"].ToString();

certificateIdTextBox.Text = choiseCurrentStatement\_reader["Certificate\_ID"].ToString();

numberOsStatementLabel.Content = $"Номер заявления: {currentStatementNumber}";

if (File.Exists(System.Environment.CurrentDirectory + $"/img/{currentStatementNumber}.jpg"))

{

ApplicantImage.Source = new BitmapImage(new Uri(System.Environment.CurrentDirectory + $"/img/{currentStatementNumber}.jpg"));

}

}

choiseCurrentStatement\_reader.Close();

}

/// <summary>

/// Кнопка выхода к главному окну

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Cansel\_button(object sender, RoutedEventArgs e)

{

HomeWindow homeWindow = new HomeWindow();

Close();

homeWindow.ShowDialog();

}

/// <summary>

/// Кнопка выбора фотографии

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void choosePhoto(object sender, RoutedEventArgs e)

{

dialog.Filter = "Файлы рисунков (\*.png, \*.jpg)|\*.png;\*.jpg";

dialog.ShowDialog();

if (dialog.FileName != "") ApplicantImage.Source = new BitmapImage(new Uri(dialog.FileName));

}

private void DeleteStatement\_Button(object sender, RoutedEventArgs e)

{

var result = MessageBox.Show("Вы точно хотите удалить запись?", "", MessageBoxButton.YesNo, MessageBoxImage.Warning);

int currentApplicantId = 0;

if (result == MessageBoxResult.Yes)

{

string str = (string)numberOsStatementLabel.Content;

string currentNumber = new string(str.Where(t => char.IsDigit(t)).ToArray());

SqlCommand selectCurrentApplicantIdCommand = new SqlCommand($"SELECT Applicant\_ID FROM Statements WHERE Statement\_ID = {currentNumber}", db.connection);

SqlDataReader selectCurrentApplicantIdReader = selectCurrentApplicantIdCommand.ExecuteReader();

while (selectCurrentApplicantIdReader.Read())

{

currentApplicantId = (int)selectCurrentApplicantIdReader["Applicant\_ID"];

}

selectCurrentApplicantIdReader.Close();

SqlCommand deleteStatementCommand = new SqlCommand($"DELETE FROM Certificates WHERE Applicant\_ID = {currentApplicantId}; DELETE FROM Statements WHERE Applicant\_ID = {currentApplicantId}; DELETE FROM Applicants WHERE Applicant\_ID = {currentApplicantId}", db.connection);

deleteStatementCommand.ExecuteNonQuery();

if (File.Exists(System.Environment.CurrentDirectory + $"/img/{currentNumber}.jpg")) File.Delete(System.Environment.CurrentDirectory + $"/img/{currentNumber}.jpg");

HomeWindow homeWindow = new HomeWindow();

Close();

homeWindow.ShowDialog();

}

}

private void SaveStatementChangeButton(object sender, RoutedEventArgs e)

{

int currentApplicantId = 0;

string

lastName = lastNameTextBox.Text,

firstName = firstNameTextBox.Text,

middleName = middleNameTextBox.Text,

dateOfBirth = dateOfBirthTextBox.Text,

sertificateId = certificateIdTextBox.Text,

placeOfEducation = placeOfEducationTextBox.Text,

levelOfEducation = levelofEducationTextBox.Text,

specialtyCode = specialtyCodeTextBox.Text,

academicYear = academicYearTextBox.Text,

avarageScore = avarageScoreTextBox.Text,

datePattern = @"(0?[1-9]|[12][0-9]|3[01]).(0?[1-9]|1[012]).((19|20)\d\d)";

string str = (string)numberOsStatementLabel.Content;

string currentNumber = new string(str.Where(t => char.IsDigit(t)).ToArray());

SqlCommand selectCurrentApplicantIdCommand = new SqlCommand($"SELECT Applicant\_ID FROM Statements WHERE Statement\_ID = {currentNumber}", db.connection);

SqlDataReader selectCurrentApplicantIdReader = selectCurrentApplicantIdCommand.ExecuteReader();

while (selectCurrentApplicantIdReader.Read())

{

currentApplicantId = (int)selectCurrentApplicantIdReader["Applicant\_ID"];

}

selectCurrentApplicantIdReader.Close();

if (

lastName == "" || firstName == "" ||

!Regex.IsMatch(dateOfBirth, datePattern) || sertificateId == "" ||

placeOfEducation == "" || levelOfEducation == "" ||

!Regex.IsMatch(academicYear, datePattern)

)

{

MessageBox.Show("Введенны некорректные данные");

}

else

{

if (dialog.FileName != "" && !File.Exists(dialog.FileName))

{

File.Copy(dialog.FileName, System.Environment.CurrentDirectory + $"/img/{numberOsStatementLabel}.jpg");

}

SqlCommand commandUpdateApplicant = new SqlCommand($"UPDATE Applicants SET LastName='{lastName}', FirstName='{firstName}', MiddleName='{middleName}', dateOfBirth='{dateOfBirth}' WHERE Applicant\_ID='{currentApplicantId}'", db.connection);

SqlCommand commandUpdateCertificate = new SqlCommand($"UPDATE Certificates SET Certificate\_ID='{sertificateId}', Avarage\_Score='{avarageScore}', Place\_of\_education='{placeOfEducation}', Level\_of\_education='{levelOfEducation}' WHERE Applicant\_ID='{currentApplicantId}'", db.connection);

SqlCommand commandUpdateStatement = new SqlCommand($"UPDATE Statements SET Specialty\_Code='{specialtyCode}', Academic\_year='{academicYear}' WHERE Applicant\_ID='{currentApplicantId}'", db.connection);

commandUpdateApplicant.ExecuteNonQuery();

commandUpdateCertificate.ExecuteNonQuery();

commandUpdateStatement.ExecuteNonQuery();

HomeWindow homeWindow = new();

Close();

homeWindow.ShowDialog();

}

}

}

}

// Добавить кнопку удаления фото

// При изменении фото - предыдущее удалить, новое установить

// ругается на avarageScore в запросе

ChangeStatementWindow.xaml

<Window x:Class="Commission.ChangeStatementWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Commission"

mc:Ignorable="d"

Title="Изменение заявления" Height="640" Width="1000" ResizeMode="NoResize" WindowStartupLocation="CenterScreen">

<Grid>

<Image HorizontalAlignment="Left" Height="102" Margin="835,24,0,0" VerticalAlignment="Top" Width="90" Source="/MGUTU.jpg"/>

<Label Content="(с) 2023 Created by GeorgeL." HorizontalAlignment="Left" Margin="20,558,0,0" VerticalAlignment="Top" Width="179"/>

<Button Content="Отменить" HorizontalAlignment="Left" Margin="857,544,0,0" VerticalAlignment="Top" Height="40" Width="83" Click="Cansel\_button"/>

<Label x:Name="numberOsStatementLabel" Content="[nubmerOfStatement]" HorizontalAlignment="Left" Margin="79,379,0,0" VerticalAlignment="Top" Width="205"/>

<TextBox MaxLength="34" x:Name="lastNameTextBox" HorizontalAlignment="Left" Margin="79,84,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="34" x:Name="middleNameTextBox" HorizontalAlignment="Left" Margin="79,144,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<Label Content="Фамилия" HorizontalAlignment="Left" Margin="79,58,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Имя" HorizontalAlignment="Left" Margin="320,58,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Отчество (при наличии)" HorizontalAlignment="Left" Margin="79,118,0,0" VerticalAlignment="Top" Width="221"/>

<TextBox MaxLength="34" x:Name="placeOfEducationTextBox" HorizontalAlignment="Left" Margin="79,265,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="34" x:Name="firstNameTextBox" HorizontalAlignment="Left" Margin="320,84,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="4" x:Name="avarageScoreTextBox" HorizontalAlignment="Left" Margin="320,265,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<Label Content="Дата рождения (вид: дд.мм.гггг)" HorizontalAlignment="Left" Margin="320,119,0,0" VerticalAlignment="Top" Width="230"/>

<Label Content="Код специальности" HorizontalAlignment="Left" Margin="79,175,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Средний балл (через точку: x.xx)" HorizontalAlignment="Left" Margin="320,239,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Место образования" HorizontalAlignment="Left" Margin="79,239,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Уровень образования" HorizontalAlignment="Left" Margin="79,297,0,0" VerticalAlignment="Top" Width="221"/>

<Label Content="Номер аттестата" HorizontalAlignment="Left" Margin="320,298,0,0" VerticalAlignment="Top" Width="221"/>

<TextBox MaxLength="34" x:Name="certificateIdTextBox" HorizontalAlignment="Left" Margin="320,324,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<Label Content="Дата подачи заявления (вид: дд.мм.гггг)" HorizontalAlignment="Left" Margin="320,178,0,0" VerticalAlignment="Top" Width="230"/>

<Button Content="Выбрать фото" HorizontalAlignment="Left" Margin="755,348,0,0" VerticalAlignment="Top" Height="37" Width="81" Click="choosePhoto"/>

<Image x:Name="ApplicantImage" HorizontalAlignment="Left" Height="190" Margin="705,144,0,0" VerticalAlignment="Top" Width="190" RenderTransformOrigin="0.5,0.5"/>

<Rectangle HorizontalAlignment="Left" Height="190" Margin="705,144,0,0" Stroke="Black" VerticalAlignment="Top" Width="190"/>

<TextBox MaxLength="8" x:Name="specialtyCodeTextBox" HorizontalAlignment="Left" Margin="79,204,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="34" x:Name="levelofEducationTextBox" HorizontalAlignment="Left" Margin="79,324,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="10" x:Name="dateOfBirthTextBox" HorizontalAlignment="Left" Margin="320,144,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<TextBox MaxLength="10" x:Name="academicYearTextBox" HorizontalAlignment="Left" Margin="320,204,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="221" Height="20"/>

<Button Content="Удалить заявление" HorizontalAlignment="Left" Margin="774,479,0,0" VerticalAlignment="Top" Height="40" Width="166" Click="DeleteStatement\_Button"/>

<Button Content="Сохранить изменения" HorizontalAlignment="Left" Margin="590,479,0,0" VerticalAlignment="Top" Height="40" Width="166" Click="SaveStatementChangeButton"/>

</Grid>

</Window>

DataBase.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace Commission

{

/// <summary>

/// Класс необходимый для создания соединения с БД

/// </summary>

public class DataBase

{

/// <summary>

/// Свойство подключения

/// </summary>

public SqlConnection connection = new SqlConnection("Data Source = localhost\\SQLEXPRESS;\nInitial Catalog = Commission;\nIntegrated Security = true\n");

/// <summary>

/// Конструктор класса, открывающий подключение

/// </summary>

public DataBase()

{

try

{

connection.Open();

}

catch (Exception ex)

{

MessageBox.Show($"Невозможно создать подключение к базе данных:\n{ex}");

Environment.Exit(1);

}

}

}

}

HomePageDataModel.cs

using System;

using System.Collections.Generic;

using System.Data.SqlTypes;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Commission

{

/// <summary>

/// Модель данных выводимой таблицы в главном окне

/// </summary>

public class HomePageDataModel

{

public string? lastName { get; set; }

public string? firstName { get; set; }

public string? middleName { get; set; }

public string? dateOfBirth { get; set; }

public string? specialtyCode { get; set; }

public double averageScore { get; set; }

public string? dateOfStatement { get; set; }

public int numberOfStatement { get; set; }

}

}

HomeWindow.xaml

<Window x:Class="Commission.HomeWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Commission"

mc:Ignorable="d"

Title="Главная" Height="640" Width="1000" ResizeMode="NoResize" WindowStartupLocation="CenterScreen" Loaded="GetData">

<Grid>

<Button Content="Выход" HorizontalAlignment="Left" Margin="857,544,0,0" VerticalAlignment="Top" Click="Exit\_btn" Height="40" Width="83"/>

<Button Content="Справка" HorizontalAlignment="Left" Margin="680,84,0,0" VerticalAlignment="Top" Click="Info\_btn" Height="40" Width="119"/>

<DataGrid IsReadOnly="True" x:Name="HomeDataGrid" CanUserResizeColumns="False" CanUserResizeRows="False" CanUserReorderColumns="False" CanUserSortColumns="False" CanUserAddRows="False" CanUserDeleteRows="False" Margin="5,144,5,85" AutoGenerateColumns="False" IsSynchronizedWithCurrentItem="True" SelectionMode="Single" SelectionUnit="Cell">

<DataGrid.Columns>

<DataGridTextColumn

Header="Фамилия"

IsReadOnly="True" Width="120" Binding="{Binding Path = lastName}"/>

<DataGridTextColumn

Header="Имя"

IsReadOnly="True" Width="120" Binding="{Binding Path = firstName}"/>

<DataGridTextColumn

Header="Отчество"

IsReadOnly="True" Width="120" Binding="{Binding Path = middleName}"/>

<DataGridTextColumn

Header="Дата рождения"

IsReadOnly="True" Width="121" Binding="{Binding Path = dateOfBirth}"/>

<DataGridTextColumn

Header="Код специальности"

IsReadOnly="True" Width="132" Binding="{Binding Path = specialtyCode}"/>

<DataGridTextColumn

Header="Средний балл"

IsReadOnly="True" Width="95" Binding="{Binding Path = averageScore}"/>

<DataGridTextColumn

Header="Дата подачи заявления"

IsReadOnly="True" Width="145" Binding="{Binding Path = dateOfStatement}"/>

<DataGridTextColumn

Header="Номер заявления"

IsReadOnly="True" Width="111" Binding="{Binding Path = numberOfStatement}"/>

</DataGrid.Columns>

</DataGrid>

<Button Content="Сохранить отчёт" HorizontalAlignment="Left" Margin="680,24,0,0" VerticalAlignment="Top" Height="40" Width="118"/>

<Button Content="Изменить количество мест" HorizontalAlignment="Left" Margin="420,24,0,0" VerticalAlignment="Top" Height="40" Width="240"/>

<Button Content="Просмотр, редактирование, удаление" HorizontalAlignment="Left" Margin="160,24,0,0" VerticalAlignment="Top" Height="40" Width="240" Click="SearchButton"/>

<Button Content="Добавить заявление" HorizontalAlignment="Left" Margin="20,24,0,0" VerticalAlignment="Top" Height="40" Width="118" Click="OpenAddingAStatement"/>

<Label Content="Фильтр:&#xA;" HorizontalAlignment="Left" Margin="15,99,0,0" VerticalAlignment="Top" Height="25" Width="59"/>

<Label Content="Специальность:" HorizontalAlignment="Left" Margin="79,77,0,0" VerticalAlignment="Top" Width="120"/>

<Label Content="Год подачи:" HorizontalAlignment="Left" Margin="220,77,0,0" VerticalAlignment="Top" Width="120"/>

<Label Content="(с) 2023 Created by GeorgeL." HorizontalAlignment="Left" Margin="20,558,0,0" VerticalAlignment="Top" Width="179"/>

<Image HorizontalAlignment="Left" Height="102" Margin="835,24,0,0" VerticalAlignment="Top" Width="90" Source="MGUTU.jpg"/>

<TextBox HorizontalAlignment="Left" Margin="220,104,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Height="21"/>

<TextBox HorizontalAlignment="Left" Margin="79,104,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Height="21"/>

<Label x:Name="countOfStatementsLabel" Content="[count of statements]" HorizontalAlignment="Left" Margin="410,558,0,0" VerticalAlignment="Top" Width="220" Height="26"/>

<Button Content="Обновить" HorizontalAlignment="Left" Margin="541,84,0,0" VerticalAlignment="Top" Height="40" Width="119"/>

<RadioButton Content="Бюджет" HorizontalAlignment="Left" Margin="360,114,0,0" VerticalAlignment="Top"/>

<RadioButton Content="Внебюджет" HorizontalAlignment="Left" Margin="360,97,0,0" VerticalAlignment="Top"/>

<RadioButton IsChecked="True" Content="Все" HorizontalAlignment="Left" Margin="460,110,0,0" VerticalAlignment="Top"/>

</Grid>

</Window>

HomeWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data.SqlClient;

using System.Data.SqlTypes;

using System.IO;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Commission

{

/// <summary>

/// Главная страница

/// </summary>

public partial class HomeWindow : Window

{

public BindingList<HomePageDataModel>? model;

/// <summary>

/// Конструктор главной страницы

/// </summary>

public HomeWindow()

{

InitializeComponent();

}

/// <summary>

/// Кнопка выхода к окну авторизации

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Exit\_btn(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

Close();

}

/// <summary>

/// Открытие файла-справки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Info\_btn(object sender, RoutedEventArgs e)

{

MessageBox.Show(File.ReadAllText(System.Environment.CurrentDirectory + "/Info.txt"));

}

/// <summary>

/// Событие при загрузке главной страницы - заполняет DataGrid из базы данных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void GetData(object sender, RoutedEventArgs e)

{

DataBase db = new();

SqlCommand command\_GetData = new SqlCommand("SELECT LastName, FirstName, MiddleName, dateOfBirth, Specialty\_Code, Avarage\_Score, Academic\_year, Statement\_ID FROM Applicants INNER JOIN Statements ON Applicants.Applicant\_ID = Statements.Applicant\_ID INNER JOIN Certificates ON Applicants.Applicant\_ID = Certificates.Applicant\_ID", db.connection);

SqlDataReader reader\_GetData = command\_GetData.ExecuteReader();

model = new BindingList<HomePageDataModel>() { };

while (reader\_GetData.Read())

{

model.Add(new HomePageDataModel()

{

lastName = reader\_GetData["LastName"].ToString(),

firstName = reader\_GetData["FirstName"].ToString(),

middleName = reader\_GetData["MiddleName"].ToString(),

dateOfBirth = reader\_GetData["dateOfBirth"]?.ToString()?.TrimEnd('0', ':', '0', '0', ':', '0'),

specialtyCode = reader\_GetData["Specialty\_Code"].ToString(),

averageScore = (double)reader\_GetData["Avarage\_Score"],

dateOfStatement = reader\_GetData["Academic\_year"].ToString()?.TrimEnd('0', ':', '0', '0', ':', '0'),

numberOfStatement = (int)reader\_GetData["Statement\_ID"],

} );

}

reader\_GetData.Close();

var sorted\_model = model.OrderByDescending(x => x.averageScore).ToList();

HomeDataGrid.ItemsSource = sorted\_model;

countOfStatementsLabel.Content = $"Количество заявлений: {sorted\_model.Count}";

}

/// <summary>

/// Открытие окна добавления заявления при нажатии на соответствующую кнопку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void OpenAddingAStatement(object sender, RoutedEventArgs e)

{

AddingAStatementWindow addingAStatementWindow = new();

Close();

addingAStatementWindow.ShowDialog();

}

private void SearchButton(object sender, RoutedEventArgs e)

{

SearchWindow searchWindow = new SearchWindow();

Close();

searchWindow.ShowDialog();

}

}

}

MainWindow.xaml

<Window x:Class="Commission.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Commission"

mc:Ignorable="d"

Title="Авторизация" Height="640" Width="1000" ResizeMode="NoResize" WindowStartupLocation="CenterScreen">

<Grid>

<Button Content="Войти" HorizontalAlignment="Center" Margin="0,345,0,0" VerticalAlignment="Top" Width="160" Height="40" Click="Auth\_btn"/>

<PasswordBox MaxLength="24" x:Name="textbox\_password" HorizontalAlignment="Center" Margin="0,305,0,0" VerticalAlignment="Top" Width="160" Password="adm123"/>

<Label Content="Пароль" HorizontalAlignment="Center" Margin="0,265,0,0" VerticalAlignment="Top" Width="160" Height="40"/>

<TextBox MaxLength="24" x:Name="textbox\_login" HorizontalAlignment="Center" TextWrapping="Wrap" VerticalAlignment="Top" Width="160" Height="20" Background="White" Margin="0,225,0,0" Text="Admin"/>

<Label Content="Логин" HorizontalAlignment="Center" Margin="0,185,0,0" VerticalAlignment="Top" Width="160" Height="40"/>

<Label Content="Version 1.0, last update 16.10.2023" HorizontalAlignment="Left" Margin="730,565,0,0" VerticalAlignment="Top" Width="220" Height="40"/>

<Label Content="Приёмная комиссия" HorizontalAlignment="Center" Margin="0,63,0,0" VerticalAlignment="Top" Width="320" FontSize="33" Height="77"/>

<Image HorizontalAlignment="Left" Height="102" Margin="835,24,0,0" VerticalAlignment="Top" Width="90" Source="/MGUTU.jpg"/>

<Label Content="(с) 2023 Created by GeorgeL." HorizontalAlignment="Left" Margin="20,558,0,0" VerticalAlignment="Top" Width="179"/>

</Grid>

</Window>

MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Reflection.PortableExecutable;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Commission

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

/// <summary>

/// Реакция на кнопку входа в аккаунт. При успешности - открывает главное окно

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

public void Auth\_btn(object sender, RoutedEventArgs e)

{

Authorization auth = new Authorization();

bool auth\_result = auth.Auth(textbox\_login.Text, textbox\_password.Password.ToString());

if (auth\_result)

{

HomeWindow homeWindow = new HomeWindow();

Close();

homeWindow.ShowDialog();

}

}

}

}

SearchWindow.xaml

<Window x:Class="Commission.SearchWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Commission"

mc:Ignorable="d"

Title="Окно поиска" Height="640" Width="1000" ResizeMode="NoResize" WindowStartupLocation="CenterScreen">

<Grid>

<Image HorizontalAlignment="Left" Height="102" Margin="835,24,0,0" VerticalAlignment="Top" Width="90" Source="/MGUTU.jpg"/>

<Label Content="(с) 2023 Created by GeorgeL." HorizontalAlignment="Left" Margin="20,558,0,0" VerticalAlignment="Top" Width="179"/>

<Button Content="Отменить" HorizontalAlignment="Left" Margin="857,544,0,0" VerticalAlignment="Top" Height="40" Width="83" Click="Cansel\_button"/>

<Label Content="Введите номер заявления: " HorizontalAlignment="Left" Margin="400,198,0,0" VerticalAlignment="Top" Width="160"/>

<TextBox MaxLength="4" x:Name="statementNumberTextBox" HorizontalAlignment="Left" Margin="400,244,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="160"/>

<Button Content="Найти" HorizontalAlignment="Left" Margin="438,284,0,0" VerticalAlignment="Top" Height="40" Width="84" Click="EnterStatementNumberButton"/>

</Grid>

</Window>

SearchWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Diagnostics.Eventing.Reader;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Commission

{

/// <summary>

/// Логика взаимодействия для SearchWindow.xaml

/// </summary>

public partial class SearchWindow : Window

{

public SearchWindow()

{

InitializeComponent();

}

/// <summary>

/// Отмена действия и возврат в главное окно

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Cansel\_button(object sender, RoutedEventArgs e)

{

HomeWindow homeWindow = new();

Close();

homeWindow.ShowDialog();

}

private void EnterStatementNumberButton(object sender, RoutedEventArgs e)

{

bool checkNumberOnBase = false;

if (int.TryParse(statementNumberTextBox.Text, out int number) && number > 999 && number < 10000)

{

DataBase db = new DataBase();

SqlCommand command\_1 = new SqlCommand("SELECT Statement\_ID FROM Statements", db.connection);

SqlDataReader reader\_1 = command\_1.ExecuteReader();

while (reader\_1.Read())

{

if (number == (int)reader\_1["Statement\_ID"])

{

checkNumberOnBase = true;

break;

};

}

if (!checkNumberOnBase)

{

MessageBox.Show("Данного номера не существует");

}

else

{

ChangeStatementWindow changeStatementWindow = new ChangeStatementWindow(number);

Close();

changeStatementWindow.ShowDialog();

}

}

else MessageBox.Show("Введён некорректный номер");

}

}

}