Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет радиотехники и электроники

Кафедра информационных радиотехнологий

Дисциплина: Базы данных и системы управления базами данных

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему

**АВТОМАТИЗАЦИЯ УЧЕТА АБИТУАРИЕНТОВ**

БГУИР КР 1-08 01 01-07 002 ПЗ

Студент: гр. 944691 Бойко В.Б.

Руководитель: кандидат физико-математических наук ассистент, доцент Киринович И.Ф.

Минск 2020

**реферат**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ](#_ujifew4aqlks) 6

1 ПОСТАНОВКА ЗАДАЧИ 7

1.1 Описание предметной области 7

1.2 Обзор существующих аналогов 7

1.3 Цели и задачи проектирования 8

1.4 Функциональное назначение и требования 9

2 ПРОЕКТИРОВАНИЕ 12

2.1 Системный анализ предметной области 12

2.2 Инфологический процесс проектирования 13

2.3 Логический этап проектирования 16

2.4 Обоснования выбора технологии и среды ее реализации 20

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 21

3.1 Физическая структура базы данных 21

3.2 Реализация проекта базы данных 25

3.2.1 Реализация проекта базы данных 25

3.2.2 Создание представлений (готовых запросов) 30

3.2.3 Назначение прав доступа 31

3.2.4 Создание индексов 32

3.3 Структура приложения 32

4 ТЕСТИРОВАНИЕ 33

5 ОПИСАНИЕ ПРИМЕНЕНИЯ 43

ЗАКЛЮЧЕНИЕ 51

Список использованных источников 52

Приложение А. Схема базы данных 53

Приложение Б. Диаграмма вариантов использования 54

Приложение В. Листинг кода 55

# 

# **ВВЕДЕНИЕ**

В текущее время активно развивается отрасль – информационная индустрия, связанная с производством технических средств, методов и технологий для решения разного рода задач. Эта индустрия тесно связана с развитием компьютерных технологий. Возрастает спрос на новые технологии, от человека требуется способность к интеллектуальному труду и творчеству. В результате появились противоречия между ограниченными возможностями человека по восприятию и переработке информации и существующими массивами хранящейся и передаваемой информации.

Для решения подобных проблем применяются автоматизированные базы данных. Они стали неотъемлемой частью практически всех компьютерных систем – от отрасти до отдельного предприятия.

Базы данных (БД) — это совокупность структур, предназначенных для хранения больших объемов информации и программных модулей, осуществляющих управление данными, их выборку, сортировку и другие подобные действия.

СУБД - это программное обеспечение или служба, используемая для создания и управления базами данных на основе реляционной модели. Когда им нужно получить доступ к данным, они подключаются к серверу СУБД. SQL Server является одним из многих вариантов программного обеспечения СУБД.

Компьютеры, которые устанавливают и запускают программное обеспечение СУБД, называются клиентами. SQL Server является одним из вариантов программного обеспечения СУБД. Многие компании используют SQL Server для хранения данных.

Создание базы данных по автоматизации учета поступления абитуриентов в учреждения образования существенно ускорит процесс поступления абитуриентов.

Преимущество базы данных по автоматизации учета поступления абитуриентов в учреждения образования:

* быстрое внесение данных абитуриента ускоряет процесс подачи документов и отбора на специальности;
* поиск по базе данных в разы быстрее поиска информации об абитуриенте в бумажной документации.
* готовая база данных используется более быстрее для анализа и процесса приема абитуриентов.

## **1. ПОСТАНОВКА ЗАДАЧИ**

### **1.1. Описание предметной области**

Программное средство создается для автоматизации приема абитуриентов в учреждения образования. База должна содержать информацию о факультетах, специальностях, первой специальности, второй специальности, третьей специальность, поданных документах и пользователях.

Целью курсового проекта является разработать программный модуль для автоматизации набора абитуриентов в учреждения образования. Информацию хранится в базе данных SQL Server.

Для работы с данными должны быть предусмотрены два: модуль администратора и модуль пользователя.

Для создания ER-модели необходимо выделить сущности предметной области:

* пользователи (код пользователя, логин, пароль, права администратора);
* факультеты (код факультета, название факультета).
* специальность (код специальности, название специальности, код факультета);
* первая специальность (код первой специальности, код специальности);
* вторая специальность (код второй специальности, код специальности);
* третья специальность (код третей специальности, код специальности);
* поданные данные абитуриента (код поданных данных, код пользователя, имя, фамилия, образование, средний балл, результат первого ЦТ, результат второго ЦТ, результат третьего ЦТ, код первой специальности, код второй специальности, код третьей специальности).

Помимо этого, должен быть администратор, который имеет возможность добавлять, удалять или обновлять существующие данные.

**1.2 Обзор существующих аналогов**

Для того чтобы стать **студентом вуза**, успешно сдать **централизованное тестирование**недостаточно. Требуется подать документы на ту **специальность**, поступление на которую возможно с набранным количеством баллов. Автоматизация учета абитуриентов облегчает работу не только будущим студентам, но и сотрудникам вузов. Данная система позволит в кратчайшие сроки принимать информацию об абитуриента и более быстрее выбрать поступившего в учреждение образования абитуриента.

### Среди аналогов автоматизированного учета абитуриентов является: автоматизированная система подачи заявлений и зачисления (АСПЗиЗ), разработанная в учреждении образования «Белорусский государственный университет информатики и радиоэлектроники». Все что нужно сделать претенденту на студенческий билет – это прийти в пункт регистрации ЦТ. При оформлении заявления ему предложат указать не только вуз и специальность, на которую он хочет поступать, но и несколько вариантов (в порядке приоритета) профильных специальностей в этом вузе или других учебных заведениях. После этого можно отправляться на испытания и ждать результатов. Как только они станут известны, специальная компьютерная программа определит, в какое учебное заведение, из ранее перечисленных, абитуриент проходит по конкурсу. Остается только подать оригиналы документов в вуз. Если не устраивает предложенный вариант либо не хватает нужных для поступления на выбранные специальности баллов, возможна итерация – оставшиеся вакансии возвращаются в общую базу, и абитуриенты заполняют заявления повторно [1].

Автоматизированная система позволит сберечь средства и волнения абитуриентов, увеличит их шансы на поступление.

### **1.3 Цели и задачи проектирования**

Целью курсового проекта является реализация автоматизированного учета абитуриентов с базой данных и упрощения взаимодействия абитуриентов и работников учреждения образования.

Для реализации данной цели необходимо:

* описать и проанализировать предметную область;
* выбрать, среди существующих, необходимы для реализации технологии;
* спроектировать и сформировать базу данных;
* создать интерфейс взаимодействия программы и пользователя;
* соединить интерфейс и модуль взаимодействия с базой данных.

### **1.4 Функциональное назначение и требования**

Разрабатываемая автоматизированная система должна иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено.

При разработке форм необходимо соблюдать определенные требования: формы в приложении должны иметь стандартные элементы управления данными, не должны быть перегружены информацией.

При конструировании форм, в ряде случаев необходимо предусмотреть возможность защиты данных от изменения, установить ограничения на некорректный ввод данных.

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – системы меню, диалоговых окон и элементов управления.

При проектировании концептуального прототипа предполагается, что при загрузке программы первой будет загружаться форма Авторизации. На ней будут расположены поля ввода Логина и Пароля, а также кнопка «Войти» и «Регистрация».

Первым этапом работы автоматизированной системы является, просмотр факультетов и специальностей учреждения образование, после авторизация – предоставление прав доступа.

В рамках этапа авторизации необходимо считать данные из таблицы с учетными записями пользователей следующего вида:

* ID\_User;
* Login;
* Password;
* Admin (данное поле служит для разделения в правах администраторов и пользователей).

После ввода пользователем своих персональных данных (логина и пароля) и сверки со считанной из базы данных информацией необходимо предусмотреть возможность входа:

* в качестве администратора (Admin = True),
* в качестве пользователя (Admin = False);

Модуль администратора включает следующие подмодули (с указанием функциональных возможностей):

* просмотр пользователей;
* назначение администратора;
* добавление, удаление, редактирование факультетов;
* добавление, удаление, редактирование специальностей;
* просмотр поданных заявок абитуриентов.

Рекомендуется: выполнять проверку новых учетных записей на уникальность логина.

Модуль пользователя включает подмодуль работы с данными со следующими функциональными возможностями:

* просмотр списка специальностей в выбранном факультете;
* регистрация пользователя
* подача заявки на поступление в учреждение образования;
* редактирование профиля.

Следующим этапом работы автоматизированной системы является работа с данными, которая становится доступной только после прохождения авторизации.

Для работы с данными должны быть предусмотрены два функциональных модуля: модуль администратора и модуль пользователя.

Для реализации перечисленных модулей/подмодулей необходимо создавать меню в соответствии с правами доступа.

В программном средстве необходимо предусмотреть:

* проверка введенных пользователем данных;
* возможность возврата назад (навигация), завершения работы программного средства;
* обратную связь с пользователем, например, вывод сообщения об успешности удаления/редактирования записи и т.д;

Навигацию, корректное завершение работы программного средства. Предусмотреть комментарии исходного кода программы (как минимум в части объявления структур данных, прототипов функций, нетривиальной логики). Дальнейшая работа ведется с данными из базы данных.

Исключить использование сложных условных и циклических алгоритмических конструкций: предусматривать максимальный уровень вложенности не более трех.

Вторым этапом работы автоматизированной системыявляется работа с данными, которая становится доступной только после прохождения авторизации.

Для работы с данными должны быть предусмотрены функциональных модуля: модуль администратора и модуль пользователя.

Для реализации перечисленных модулей/подмодулей необходимо создавать меню с соответствующими пунктами.

Возможность возврата назад (навигация), завершения работы программного средства.

Навигацию, корректное завершение работы программного средства.

## 

## **2 ПРОЕКТИРОВАНИЕ**

### **2.1 Системный анализ предметной области**

На этапе системного анализа предметной области дается подробное словесное и графическое описание объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами.

Используя язык UML, желательно создавать различные виды диаграмм, позволяющих визуализировать описание предметной области.

Основными типами диаграмм являются:

* диаграмма вариантов использования;
* диаграмма взаимодействия
* и иные диаграммы.

На рисунке 2.1 показан пример диаграммы вариантов использования.

Системный анализ завершается:

- подробным неформальным описанием объектов  предметной области, которое требуется для решения конкретных задач базой данных;

- формулировкой конкретных задач, которые будут решаться с использованием данной базы данных с кратким описанием алгоритмов их решения;

- описанием выходных данных, которые должны генерироваться программным средством;

- описанием входных данных, которые служат основанием для заполнения базы данных.

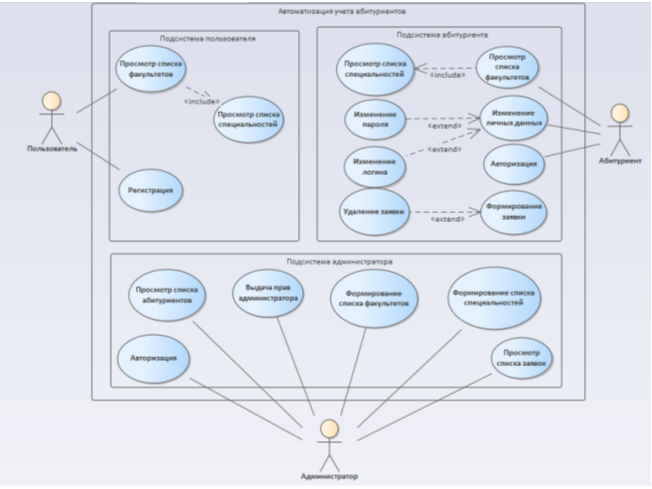


Рисунок 2.1 – Диаграмма вариантов использования

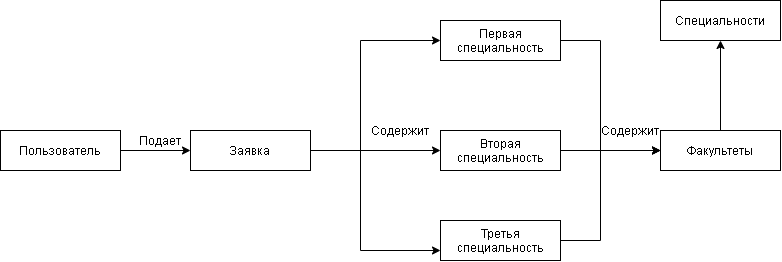
### **2.2 Инфологический этап проектирования**

Автоматизированная система создается для организации работы учреждения образования во время вступительной компании. База должна содержать информацию о пользователях, факультетах, специальностях и поданных заявок на поступление в учреждение образования.

В соответствии с предметной областью база данных проектируется с учетом следующих особенностей:

* каждый абитуриент способен оформить одну заявку на несколько специальностей в учреждении образования;
* первая специальность может быть у многих заявок, но у каждой заявки может быть только одна заявка;
* вторая специальность может быть у многих заявок, но у каждой заявки может быть только одна заявка;
* третья специальность может быть у многих заявок, но у каждой заявки может быть только одна заявка;
* у каждой первой специальности может быть только одна специальность, и каждая специальность может быть только один раз первой специальностью;
* у каждой второй специальности может быть только одна специальность, и каждая специальность может быть только один раз второй специальностью;
* у каждой третьей специальности может быть только одна специальность, и каждая специальность может быть только один раз третьей специальностью;
* факультет имеет много специальностей, но у специальности может быть только один факультет.

Анализ выявленных сущностей позволяет построить ER–диаграмму как показано на рисунке 2.1.

Рисунок2.1 – ER-диаграмма предметной области

Определим группы пользователей, их основные задачи и запросы к базе данных:

Абитуриент:

* просмотр списка факультетов и специальностей
* подача заявки на поступление в учреждение образования;
* редактирование профиля;

Администратор:

* просмотр списка факультетов и специальностей;
* просмотр данных абитуриентов и их сортировка;
* назначение прав администратора;
* просмотр, редактирование, добавление, удаление и сортировка информации о факультетах;
* просмотр, редактирование, добавление, удаление и сортировка информации о специальностях;
* просмотр поданных заявок и их сортировка;

Для преобразования ER-диаграммы в схему базы данных приведем уточненную ER-диаграмму, содержащую атрибуты сущностей как показано на рисунке 2.2

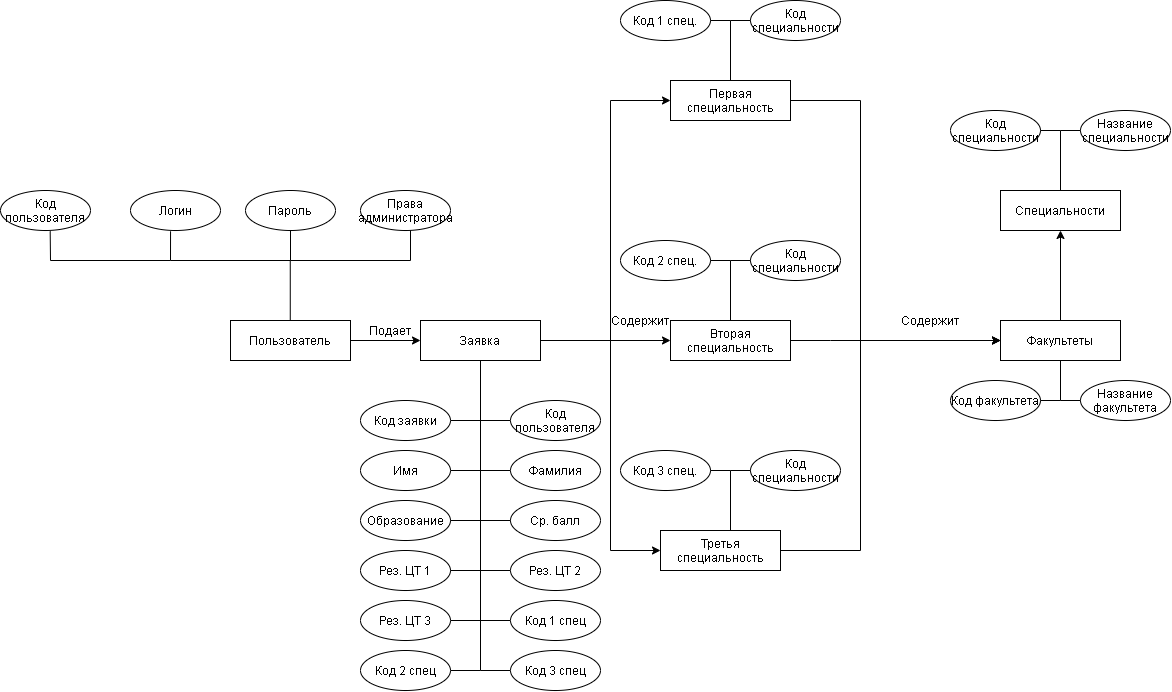


Рисунок *2.2* Уточненная ER-диаграмма предметной области

**Описание операционной обстановки**

Для определения требований к операционной обстановке необходимо примерно знать объём работы учреждения образования а также иметь представление о характере и интенсивности запросов.

Объём внешней памяти, необходимый для функционирования программного средства складывается из двух составляющих: память, занимаемая модулями СУБД (ядро, утилиты, вспомогательные программы), и память, отводимая под данные (МД). Для реальных баз данных обычно наиболее существенным является МД.

На основе результатов анализа предметной области можно приблизительно оценить объём памяти, требуемой для хранения данных. Примем ориентировочно, что:

* происходит около 7000 подач заявок в год (по 0,05К на заявку);
* во время вступительной компании работает около 100 сотрудников (по 0,1К на сотрудников);
* предоставляется около 100 специальностей (по 0,05К на специальность);
* сервисом пользуется около 7000 абитуриентов в год (по 0,05К на абитуриаента).

Тогда объём памяти для хранения данных за первый год примерно составит:

Mд = 2(7000\*0,05+100\*0,1+100\*0,05+7000\*0,05) = 1430 К

Коэффициент 2 необходим для того, чтобы учесть необходимость выделения памяти под дополнительные структуры (например, индексы). Объём памяти будет очищаться ежегодно после каждой вступительной программы.

Необходимый объём оперативной памяти определяется на основании анализа интенсивности запросов и объёма результирующих данных.

**2.3 Логический этап проектирования**

На основе созданных сущностей (пользователи, заявка, первая специальность, вторая специальность, третья специальность, факультет, специальность) составим реляционные отношения.

Каждое реляционное отношение соответствует одной сущности предметной области и все атрибуты этой сущности. Для каждого отношения определяются первичный ключ и внешние ключи.

В сущности «Пользователи» первичным ключом является атрибут ID пользователя. Описание сущности «Пользлватели», предназначенное для хранения данных о всех зарегистрированных пользователях, представлено в таблице 2.1.

|  |  |  |
| --- | --- | --- |
| Таблица 2.1 – Описание отношения «Пользователи» | | |
| Название атрибута | Заполнение | Описание |
| 1 | 2 | 3 |
| ID пользователя | Обязательное (ключевое поле) | Идентификационный номер брони |
| Логин | Обязательное | Логин пользователя |
| Пароль | Обязательное | Пароль пользователя |
| Права | Обязательное | Права администратора |

В сущности «Заявки» первичным ключом является атрибут ID заявки. Описание сущности «Заявки», предназначенной для хранения данных о всех заявках абитуриентов, представлено в таблице 2.2.

|  |  |  |
| --- | --- | --- |
| Таблица 2.2 – Описание отношения «Заявки» | | |
| Название атрибута | Заполнение | Описание |
| 1 | 2 | 3 |
| ID заявки | Обязательное (ключевое поле) | Идентификационный номер заявки |
| ID пользователя | Обязательное | Идентификационный номер брони |
| Фамилия | Обязательное | Фамилия пользователя |
| Имя | Обязательное | Имя пользователя |
| Образование | Обязательное | Образование пользователя |
| Аттестат | Обязательное | Средний балл аттестат пользователя |
| ЦТ 1 | Обязательное | Результат централизованного тестирования по первому предмету |
| ЦТ 2 | Обязательное | Результат централизованного тестирования по второму предмету |
| ЦТ 3 | Обязательное | Результат централизованного тестирования по третьему предмету |
| ID первой специальности | Обязательное | Идентификационный номер первой специальности |
| ID второй специальности | Необязательное | Идентификационный номер второй специальности |
| ID третьей специальности | Необязательное | Идентификационный номер третьей специальности |

В сущности «Первая специальность» первичным ключом является атрибут ID первой специальности. Описание сущности «Первая специальность», предназначенной для хранения первой выбранной специальности, представлено в таблице 2.3.

|  |  |  |
| --- | --- | --- |
| Таблица 2.3 – Описание отношения «Первая специальность» | | |
| Название атрибута | Заполнение | Описание |
| 1 | 2 | 3 |
| ID первой специальности | Обязательное (ключевое поле) | Идентификационный номер первой специальности |
| ID специальности | Обязательное | Идентификационный номер специальности |

В сущности «Вторая специальность» первичным ключом является атрибут ID второй специальности. Описание сущности «Вторая специальность», предназначенной для хранения второй выбранной специальности, представлено в таблице 2.4.

|  |  |  |
| --- | --- | --- |
| Таблица 2.4 – Описание отношения «Вторая специальность» | | |
| Название атрибута | Заполнение | Описание |
| 1 | 2 | 3 |
| ID второй специальности | Обязательное (ключевое поле) | Идентификационный номер второй специальности |
| ID специальности | Обязательное | Идентификационный номер специальности |

В сущности «Третья специальность» первичным ключом является атрибут ID третьей специальности. Описание сущности «Третья специальность», предназначенной для хранения третьей выбранной специальности, представлено в таблице 2.5.

|  |  |  |
| --- | --- | --- |
| Таблица 2.5 – Описание отношения «Третья специальность» | | |
| Название атрибута | Заполнение | Описание |
| 1 | 2 | 3 |
| ID третьей специальности | Обязательное (ключевое поле) | Идентификационный номер третьей специальности |
| ID специальности | Обязательное | Идентификационный номер специальности |

В сущности «Факультет» первичным ключом является атрибут ID факультета. Описание сущности «Факультет», предназначенной для хранения факультетов, представлено в таблице 2.6.

|  |  |  |
| --- | --- | --- |
| Таблица 2.6 – Описание отношения «Факультет» | | |
| Название атрибута | Заполнение | Описание |
| 1 | 2 | 3 |
| ID факультета | Обязательное (ключевое поле) | Идентификационный номер факультета |
| Факультет | Обязательное | Наименование факультета |

В сущности «Cпециальность» первичным ключом является атрибут ID специальности. Описание сущности «Специальность», предназначенной для хранения специальностей, представлено в таблице 2.7.

|  |  |  |
| --- | --- | --- |
| Таблица 2.7 – Описание отношения «Специальность» | | |
| Название атрибута | Заполнение | Описание |
| 1 | 2 | 3 |
| ID специальности | Обязательное (ключевое поле) | Идентификационный номер специальности |
| ID факультета | Обязательное | Идентификационный номер факультета |
| Специальность | Обязательное | Наименование специальности |

Нормализация полученных отношений

Как было описано ранее, нормализация базы данных устраняет избыточность данных и тем самым позволяет избежать нарушения целостности базы данных при изменении данных, то есть избежать аномалий изменения.

Приведем разработанную базу данных к 3НФ.

Первая нормальная форма (1NF) предполагает, что каждое сохраняемое данное должно представлять скалярное значение, а отношения не должны содержать повторяющихся строк.

1НФ. Каждое сохраняемое данное представляет скалярное значение, а отношения не содержат повторяющиеся строки.

2НФ. В нашем случае составные первичные ключи отсутствуют.

3НФ. В отношениях, приведенных выше, транзитивные зависимости отсутствуют.

Схема базы данных после нормализации приведена на рисунке 2.3.

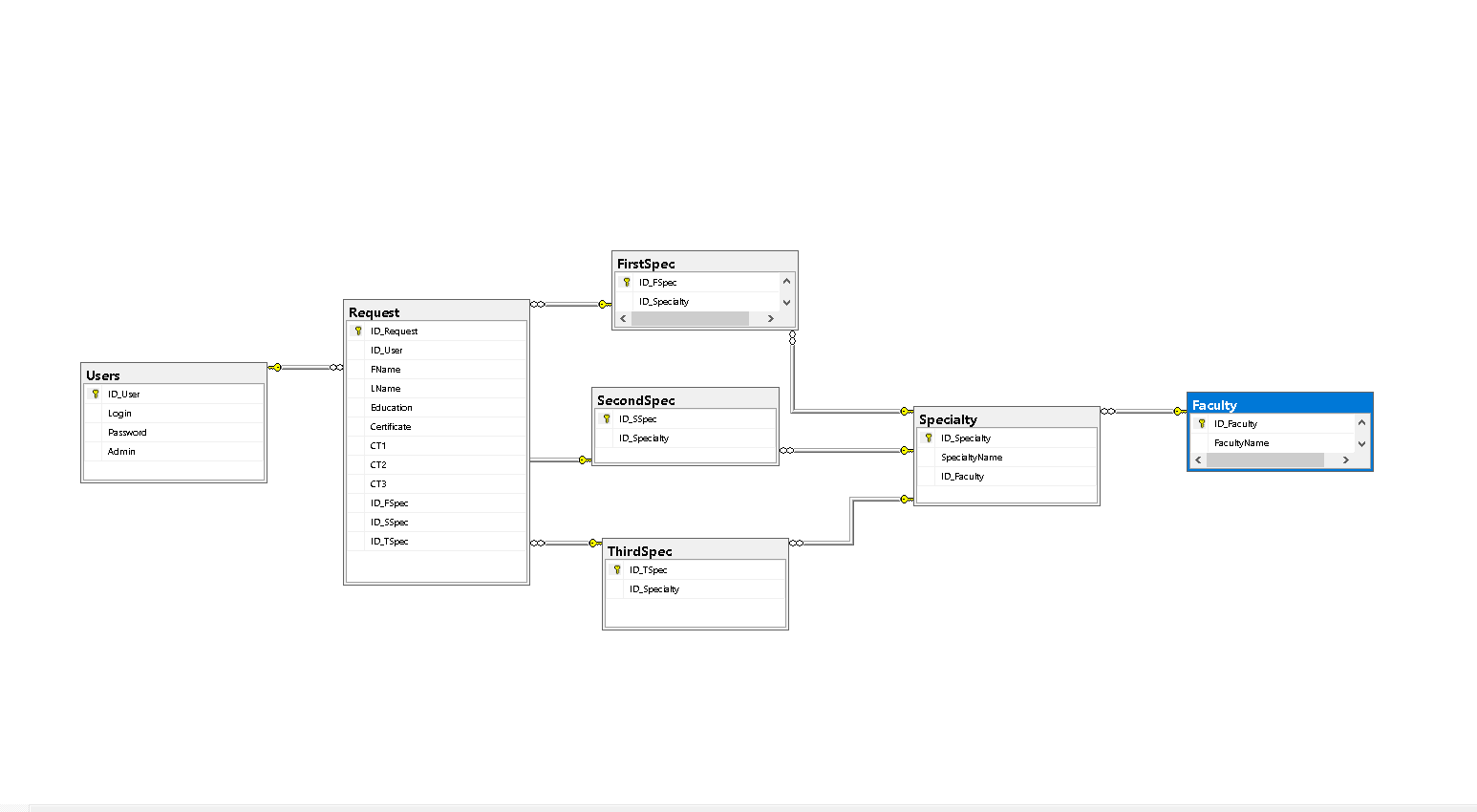


Рисунок 2.3 – Схема данных после нормализации

### **2.4 Обоснование выбора технологий и описание среды реализации**

Для реализации курсового проекта была выбраны такие технологии: язык программирования C#, СУБД MS SQL Server, Entity Framework для взаимодействия с базой данных, JSON для хранения подключения к базе данных. Так как данные технологии существуют довольно давно и являются общедоступными - это позволяет находить любую требуемую для разработки документацию, что облегчает и ускоряет процесс разработки.

LINQ (Language-Integrated Query) представляет простой и удобный язык запросов к источнику данных. В качестве источника данных может выступать объект, реализующий интерфейс IEnumerable (например, стандартные коллекции, массивы), набор данных DataSet, документ XML. Но вне зависимости от типа источника LINQ позволяет применить ко всем один и тот же подход для выборки данных

Windows Forms –  [интерфейс программирования приложений](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) (API), отвечающий за [графический интерфейс пользователя](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) и являющийся частью [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework). Данный интерфейс упрощает доступ к элементам интерфейса [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows) за счет создания обёртки для существующего [Win32 API](https://ru.wikipedia.org/wiki/Windows_API) в [управляемом коде](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4). Причём управляемый код – классы, реализующие [API](https://ru.wikipedia.org/wiki/API) для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

В качестве среды разработки была выбрана Microsoft Visual Studio Community edition. Среда разработки является одной из самых популярных и удобных для написания кода, имея в своём арсенале большой объем различных средств для подсказок в синтаксисе, уже готовых конструкций и инструментов для дебага.

Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

В ней имеется встроенный дизайнер для создания форм взаимодействия приложения с пользователем, что актуально для данного курсового проекта.

**3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

### **3.1 Физическая структура базы данных**

Физическая структура базы данных определяет тип и свойства данных, которые будут записаны в память компьютера.

Правила перехода к физической модели следующие: каждое отношение превращается в файл базы данных, каждый столбец - в поле файла, каждая строка – в запись файла. Этап физического моделирования базы данных включает в себя определение состава файлов и их заполнение исходными данными в соответствии с ограничениями, допущениями и особенностями предметной области.

На стадии физического проектирования происходит непосредственная реализация базы данных в Microsoft SQL Server. Были созданы следующие таблицы: пользователи, заявки, первая специальность, вторая специальность, третья специальность, факультет и специальность.

Таблица «Пользователи» хранит информацию о всех зарегистрированных абитуриентах в системе. Структура приведена в таблице 3.1.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 3.1 – Структура таблицы «Пользователи» | | | |
| Имя поля | Тип данных | Размер, байт | Описание |
| ID пользователя | Int | 4 | Идентификационный номер пользователя (первичный ключ) |
| Логин | Varchar | 52 | Логин |
| Пароль | Varchar | 52 | Пароль |
| Права | bit | 1 | Права администратора |

Таблица «Первая специальность» хранит информацию о всех поданных первых специальностях. Структура приведена в таблице 3.2.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 3.2 – Структура таблицы «Первая специальность» | | | |
| Имя поля | Тип данных | Размер, байт | Описание |
| ID первой специальности | Int | 4 | Идентификационный номер первой специальности (первичный ключ) |
| ID специальности | Int | 4 | Идентификационный номер специальности (внешний ключ) |

Таблица «Заявки» хранит информацию о всех поданных заявках абитуриентов. Структура приведена в таблице 3.3.

Таблица 3.3 – Структура таблицы «Заявки»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| ID запроса | Int | 4 | Идентификационный номер запроса (первичный ключ) |
| ID пользователя | Int | 4 | Идентификационный номер пользователя (внешний ключ) |
| Фамилия | Varchar | 52 | Фамилия пользователя |
| Имя | Varchar | 52 | Имя пользователя |
| Имя поля | Тип данных | Размер, байт | Описание |
| Образование | Varchar | 52 | Образование пользователя |
| Аттестат | Int | 4 | Средний балл аттестата пользователя |
| ЦТ 1 | Int | 4 | Результат централизованного тестирования по первому предмету |
| ЦТ 2 | Int | 4 | Результат централизованного тестирования по второму предмету |
| ЦТ 3 | Int | 4 | Результат централизованного тестирования по третьему предмету |
| ID первой специальности | Int | 4 | Идентификационный номер первой специальности (внешний ключ) |
| ID второй специальности | Int | 4 | Идентификационный номер второй специальности (внешний ключ) |
| ID третьей специальности | Int | 4 | Идентификационный номер третьей специальности (внешний ключ) |

Таблица «Вторая специальность» хранит информацию о всех поданных вторых специальностях. Структура приведена в таблице 3.4.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 3.4 – Структура таблицы «Вторая специальность» | | | |
| Имя поля | Тип данных | Размер, байт | Описание |
| ID второй специальности | Int | 4 | Идентификационный номер второй специальности (первичный ключ) |
| ID специальности | Int | 4 | Идентификационный номер специальности (внешний ключ) |

Таблица «Третья специальность» хранит информацию о всех поданных третьих специальностях. Структура приведена в таблице 3.5.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 3.5 – Структура таблицы «Третья специальность» | | | |
| Имя поля | Тип данных | Размер, байт | Описание |
| ID третьей специальности | Int | 4 | Идентификационный номер третьей специальности (первичный ключ) |
| ID специальности | Int | 4 | Идентификационный номер специальности (внешний ключ) |

Таблица «Специальности» хранит информацию о всех специальностях в учреждении образования. Структура приведена в таблице 3.6.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 3.6 – Структура таблицы «Специальности» | | | |
| Имя поля | Тип данных | Размер, байт | Описание |
| ID специальности | Int | 4 | Идентификационный номер специальности (первичный ключ) |
| Специальность | Varchar | 52 | Наименование специальности |
| ID факультета | Int | 4 | Идентификационный номер факультета (внешний ключ) |

Таблица «Факультеты» хранит информацию о всех факультетах в учреждении образования. Структура приведена в таблице 3.7.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 3.7 – Структура таблицы «Факультеты» | | | |
| Имя поля | Тип данных | Размер, байт | Описание |
| ID факультета | Int | 4 | Идентификационный номер факультета (первичный ключ) |
| Факультет | Varchar | 52 | Наименование факультета |

Структура базы данных показана на рисунке 3.1.

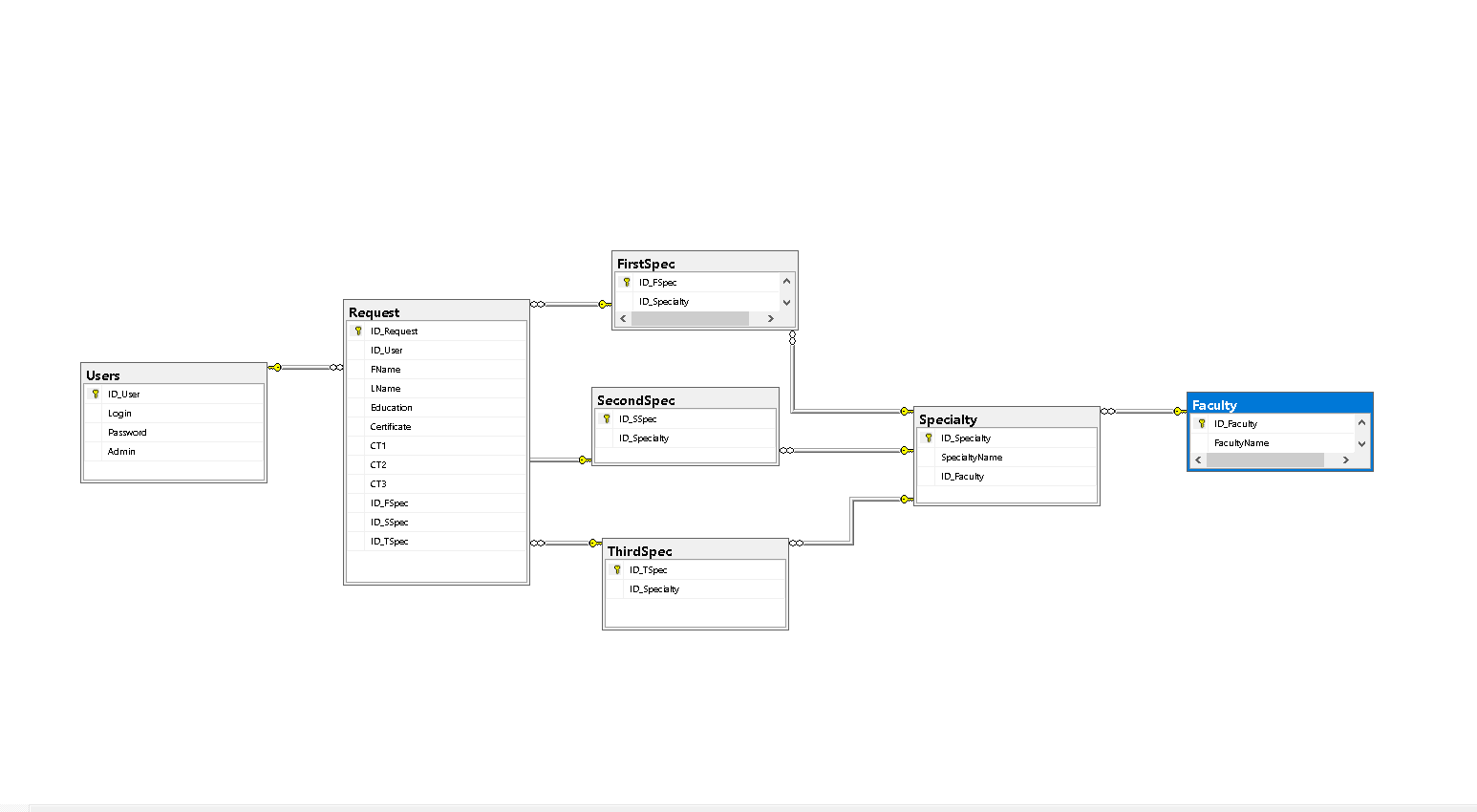


Рисунок 3.1 – Физическая структура базы данных

Для каждого отношения в табл. 3.1-3.7 указаны атрибуты с их внутренним названием, типом и длиной. Типы данных обозначаются так: N – числовой, C – символьный тип фиксированной длины, V – символьный тип переменной длины, D – дата (этот тип имеет стандартную длину, зависящую от СУБД, поэтому она не указывается). О правилах выбора типов данных подробно рассказано в [1].

Опишем для каждой группы пользователей права доступа к каждой таблице. Права доступа должны быть распределены так, чтобы для каждого объекта базы данных был хотя бы один пользователь, который имеет право добавлять и удалять данные из объекта. Права приведены в таблице 3.8.

Используются следующие сокращения:

* s – чтение данных (select);
* i – добавление данных (insert);
* u – модификация данных (update);
* d – удаление данных(delete).

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 3.8 – Права доступа к таблицам | | |  |
| Таблицы | Группы пользователей (роли) | |
| Пользователь | Администратор |
| Пользователи | SIUD | SIUD |
| Заявки | SIUD | SIUD |
| Первая специальность | S | SIUD |
| Вторая специальность | S | SIUD |
| Третья специальность | S | SIUD |
| Специальности | S | SIUD |
| Факультеты | S | SIUD |

Права доступа назначает администратор.

### **3.2. Реализация проекта базы данных**

### **3.2.1. Создание таблиц**

### Отношение Пользователи:

CREATE TABLE [dbo].[Users](

[ID\_User] [int] IDENTITY(1,1) NOT NULL,

[Login] [nvarchar](25) NOT NULL,

[Password] [nvarchar](25) NOT NULL,

[Admin] [bit] NOT NULL,

PRIMARY KEY CLUSTERED

(

[ID\_User] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

### GO

1. Отношение заявки:

CREATE TABLE [dbo].[Request](

[ID\_Request] [int] IDENTITY(1,1) NOT NULL,

[ID\_User] [int] NOT NULL,

[FName] [nvarchar](20) NOT NULL,

[LName] [nvarchar](20) NOT NULL,

[Education] [nvarchar](50) NOT NULL,

[Сertificate] [int] NOT NULL,

[CT1] [int] NOT NULL,

[CT2] [int] NOT NULL,

[CT3] [int] NOT NULL,

[ID\_FSpec] [int] NOT NULL,

[ID\_SSpec] [int] NULL,

[ID\_TSpec] [int] NULL,

PRIMARY KEY CLUSTERED

(

[ID\_Request] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Request] WITH CHECK ADD FOREIGN KEY([ID\_FSpec])

REFERENCES [dbo].[FirstSpec] ([ID\_FSpec])

GO

ALTER TABLE [dbo].[Request] WITH CHECK ADD FOREIGN KEY([ID\_SSpec])

REFERENCES [dbo].[SecondSpec] ([ID\_SSpec])

GO

ALTER TABLE [dbo].[Request] WITH CHECK ADD FOREIGN KEY([ID\_TSpec])

REFERENCES [dbo].[ThirdSpec] ([ID\_TSpec])

GO

ALTER TABLE [dbo].[Request] WITH CHECK ADD FOREIGN KEY([ID\_User])

REFERENCES [dbo].[Users] ([ID\_User])

GO

1. Отношение первая специальность:

CREATE TABLE [dbo].[FirstSpec](

[ID\_FSpec] [int] IDENTITY(1,1) NOT NULL,

[ID\_Specialty] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

[ID\_FSpec] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[FirstSpec] WITH CHECK ADD FOREIGN KEY([ID\_Specialty])

REFERENCES [dbo].[Specialty] ([ID\_Specialty])

GO

1. Отношение вторая специальность:

CREATE TABLE [dbo].[SecondSpec](

[ID\_SSpec] [int] IDENTITY(1,1) NOT NULL,

[ID\_Specialty] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

[ID\_SSpec] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[SecondSpec] WITH CHECK ADD FOREIGN KEY([ID\_Specialty])

REFERENCES [dbo].[Specialty] ([ID\_Specialty])

GO

1. Отношение третья специальность:

CREATE TABLE [dbo].[ThirdSpec](

[ID\_TSpec] [int] IDENTITY(1,1) NOT NULL,

[ID\_Specialty] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

[ID\_TSpec] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[ThirdSpec] WITH CHECK ADD FOREIGN KEY([ID\_Specialty])

REFERENCES [dbo].[Specialty] ([ID\_Specialty])

GO

1. Отношение специальности:

CREATE TABLE [dbo].[Specialty](

[ID\_Specialty] [int] IDENTITY(1,1) NOT NULL,

[SpecialtyName] [nvarchar](150) NOT NULL,

[ID\_Faculty] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

[ID\_Specialty] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Specialty] WITH CHECK ADD FOREIGN KEY([ID\_Faculty])

REFERENCES [dbo].[Faculty] ([ID\_Faculty])

GO

1. Отношение факультеты

CREATE TABLE [dbo].[Faculty](

[ID\_Faculty] [int] IDENTITY(1,1) NOT NULL,

[FacultyName] [nvarchar](100) NOT NULL,

PRIMARY KEY CLUSTERED

(

[ID\_Faculty] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

1. Заполнение отношения пользователя

INSERT INTO [dbo].[Users]

([Login]

,[Password]

,[Admin])

VALUES

(‘floliq’,

,'1337228',

,1)

1. Заполнение отношения заявки

INSERT INTO [dbo].[Request]

([ID\_User]

,[FName]

,[LName]

,[Education]

,[Сertificate]

,[CT1]

,[CT2]

,[CT3]

,[ID\_FSpec]

,[ID\_SSpec]

,[ID\_TSpec])

VALUES

(1

,’Бойко’

,’Владислав’

,’Общее среднее’

,50

,100

,100

,100

,32

,4

,6)

1. Заполнение отношения первая специальность

INSERT INTO [dbo].[FirstSpec]

([ID\_Specialty])

VALUES

(2)

1. Заполнение отношения вторая специальность

INSERT INTO [dbo].[SecondSpec]

([ID\_Specialty])

VALUES

(2)

1. Заполнение отношения Третья специальность

INSERT INTO [dbo].[ThirdSpec]

([ID\_Specialty])

VALUES

(2)

1. Заполнение отношения Специальности

INSERT INTO [dbo].[Specialty]

([SpecialtyName]

,[ID\_Faculty])

VALUES

(‘Профессиональное обучение (информатика)’>

,4)

1. Заполнение отношения факультеты

INSERT INTO [dbo].[Faculty]

([FacultyName])

VALUES

(‘ФРЭ’)

**3.2.2. Создание представлений (готовых запросов)**

1. Список всех пользователей

create view allUsers

as

SELECT \*

FROM [dbo].[Users]

ORDER BY [FName]

1. Список всех запросов

create view allRequests

as

SELECT \*

FROM [dbo].[Requests]

ORDER BY [FName]

1. Список всех специальностей

create view allSpecialities

as

SELECT \*

FROM [dbo].[Speciality]

ORDER BY [ID\_Faculty]

1. Список всех факультетов

create view allFaculties

as

SELECT \*

FROM [dbo].[Faculty]

ORDER BY [FacultyName]

**3.2.3. Назначение прав доступа**

Права доступа пользователей предоставляются с помощью команды GRANT. Рассмотрим для примера администратора admin. Права доступа к таблицам Пользователи и Заявки могут быть описаны следующим образом:

grand select, insert, update, delete on [dbo.Users]

to admin;

grand select, insert, update, delete on [dbo.Requests]

to admin;

Права доступа пользователя user к таблицам Пользователи и Заявки могут быть описаны следующим образом:

grand select, insert, update on [dbo.Users]

to user;

grand select, insert, update, delete on [dbo.Requests]

to user;

Права доступа пользователя user к таблицам Факультеты и Специальности могут быть описаны следующим образом:

grand select on [dbo.Faculty]

to user;

grand select on [dbo.Specitility]

to user;

**3.2.4. Создание индексов**

Для повышения производительности поиска данных были проиндексированы некоторые столбцы таблиц Бронь и Тур.

Индексирование поля ID менеджера в таблице Бронь:

create index userrid

on [dbo.Users] ([ID\_User]);

Индексирование поля ID специальности в таблице Специальность:

create index specid

on [dbo.Speciality] ([ID\_Speciality]);

**3.3 Структура приложения**

Структура приложения представлена в таблице 3.9

Таблица 3.9 – Структура приложения

|  |  |
| --- | --- |
| Имя файла | Назначение |
| AutRegApplicants.exe | Исполняемый файл проекта |
| AutRegApplicants.pdb | Файл подключения к базе данный |
| EntityFramework.dll | Файл подключение взаимодействия с объектами |
| EntityFramework.SqlServer.dll | Файл подключения взаимодействия с объектами MS SQL Server |
| EntityFramework.SqlServer.xml | Файл подключегния взаимодействия с объектами MS SQL Server |
| MaterialSkin.dll | Файл подключения графического интерфейса |

**4 ТЕСТИРОВАНИЕ**

Тестирование программного обеспечения - это проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом.

Тестирование проводится с целью повысить вероятность того, что приложение, предназначенное для тестирования, будет работать правильно при любых обстоятельствах, повысить вероятность того, что приложение, предназначенное для тестирования, будет соответствовать всем описанным требованиям, предоставление актуальной информации о состоянии продукта на данный момент.

В курсовом проекте было использовано ручное(мануальное) тестирование, что позволило смоделировать поведение будущего пользователя программного продукта. Учитывая размер программы и ее глубину, был выбран этот вариант из-за его скорости и эффективности на фоне остальных.

При открытии программы пользователя предоставляется главное окно с возможность выбора специальности и перехода в окно авторизации (рис. 4.1 и 4.2).

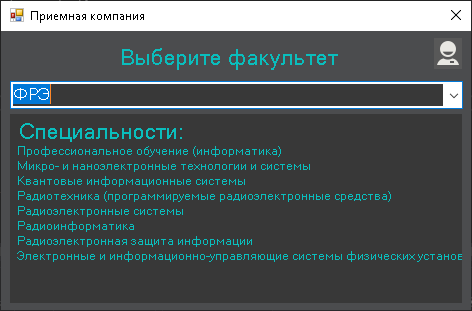


Рисунок 4.1 – Главное окно

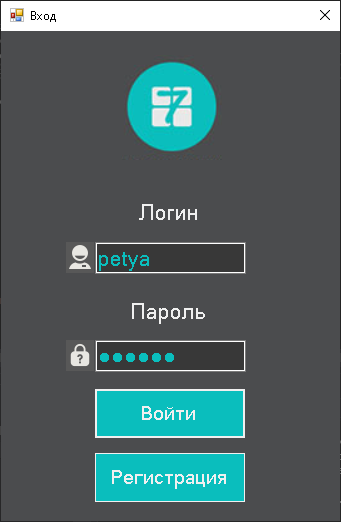


Рисунок 4.2 – Форма авторизации

Для авторизации пользователя необходимо ввести правильный логин и пароль.

В случае, если введённый логин не существует в системе или неправильный пароль, отображается сообщение, об отсутствии данного пользователя в системе (рисунок 4.3).

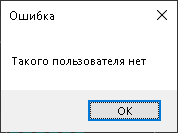


Рисунок 4.3 – Сообщение об отсутствии пользователя в системе

В окне авторизации присутствует возможность регистрации нового пользователя. Форма регистрации представлена в рисунке 4.4.

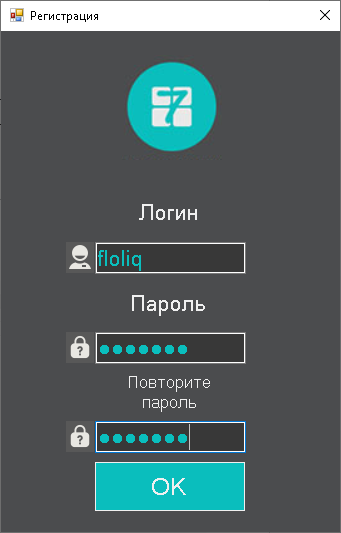


Рисунок 4.4 – Окно регистрации

В случае заполнения данных и существовании пользователя в системе предоставляется сообщение о том, что пользователь уже существует в системе (рисунок 4.5). При условии несовпадении полей «Пароль» и «Повторите пароль» отображается сообщение об несовпадении пароля и подтверждающего пароля (рисунок 4.6)

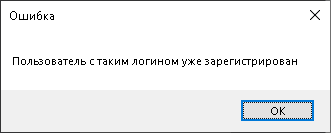


Рисунок 4.5 – Сообщение о существовании пользователя в системе

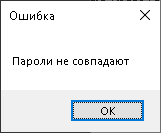


Рисунок 4.6 – Сообщение об несовпадении пароля и подтверждающего пароля

После успешной авторизации или регистрации открывается главное окно программы (рисунок 4.7).

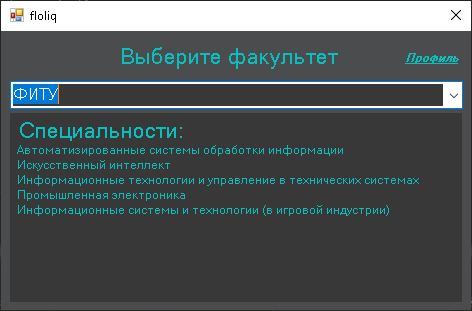


Рисунок 4.7 – Главное окно программы после успешной авторизации или регистрации

При нажатии на текстовую ссылку «Профиль» открывается окно личного профиля пользователя (рисунок 4.8).

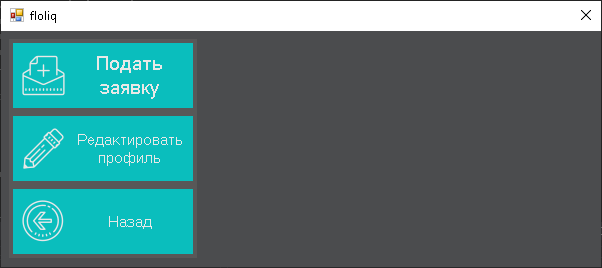


Рисунок 4.8 – Окно личного кабинета пользователя

В личном кабинете пользователя представлена возможность перехода в окно подачи заявки, редактирования профиля и возврата на главное окно программы. При нажатии на кнопку «Подать заявку», открывается окно для подачи заявок (рисунок 4.9).

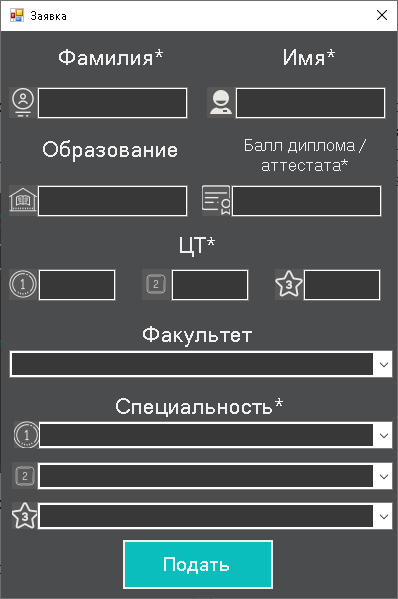


Рисунок 4.9 – Окно подачи заявки

Для подачи заявки необходимо заполнить все обязательные поля. Исключением является поле «Специальность» (возможен выбор только одной специальности).

При условии не заполнения всех обязательных полей и нажатия на кнопку «Подать», отображается сообщение о отсутствии заполнения обязательных полей (рисунок 4.10).

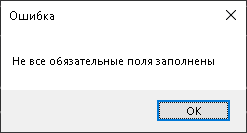


Рисунок 4.10 – Сообщение о отсутствии заполнения обязательных полей

После успешного заполнения полей подачи заявки, осуществляется переход в личный кабинет пользователя (рисунок 4.8).

Для редактирования профиля личного кабинете, необходимо нажать на кнопку «Редактировать профиль». После нажатия отображается формы для редактирования логина или пароля (рисунок 4.10).

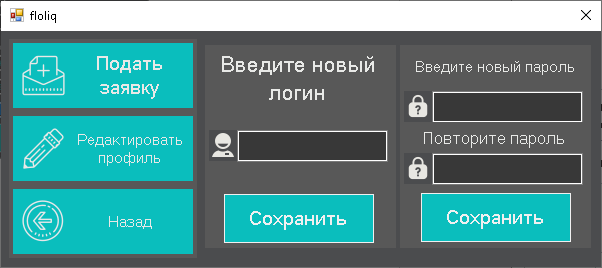


Рисунок 4.10 – Редактирование профиля

В случае изменения логина на существующий, отображается сообщение об существовании пользователя в системе (рисунок 4.5). При условии несовпадении полей ввода «Пароль» и «Подтвердите пароль» появляется сообщение об несовпадении пароля и подтверждающего пароля (рисунок 4.6).

При авторизации пользователя с правами администратора открывается окно администратора, в котором возможен просмотр всех пользователей, факультетов, специальностей и заявок (рисунок 4.11).

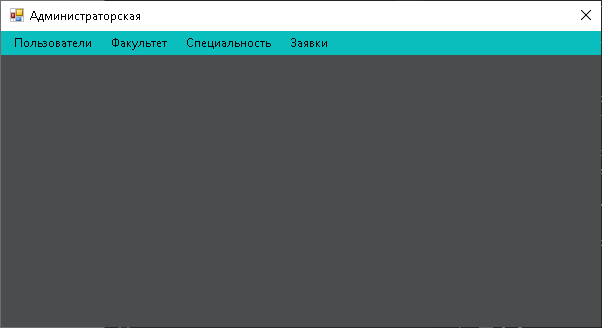


Рисунок 4.11 – Окно администратора

При нажатии на строку меню и выборах полей «Факультет» и «Специальности», открывается контекстное меню с выбором пунктов «Добавить», «Удалить», «Редактировать» (рисунок 4.12).

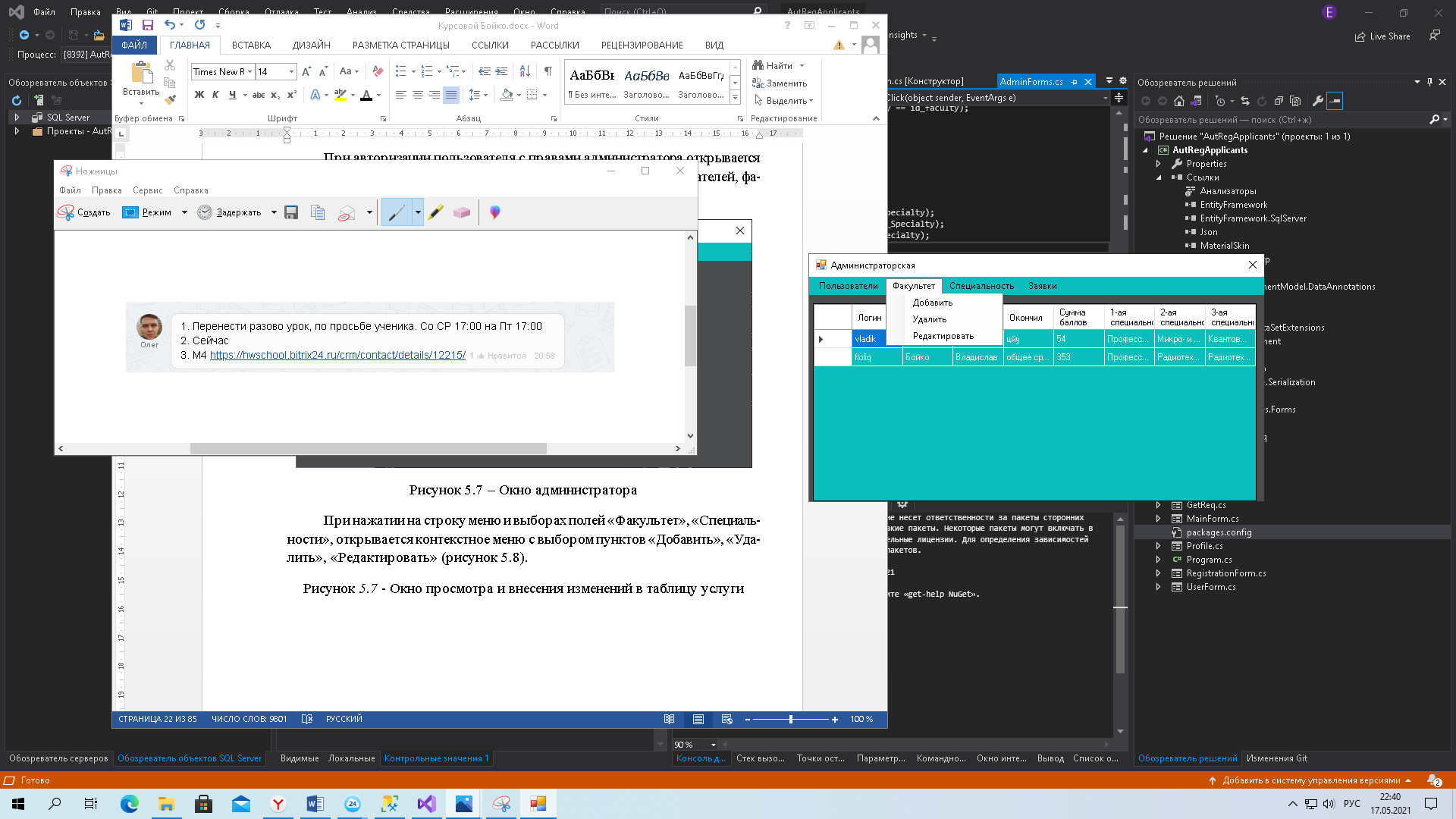


Рисунок 4.12 – Контекстное меню

При нажатии на одно из полей контекстного меню («Добавить», «Редактировать» или «Удалить») в строках меню «Факультет» и «Специальности», отображается выбранная информация, а также появляется возможность для добавления, удаления и редактирования записей (рисунки 4.13-5.15). Заполнения данных производится в текстовом поле.

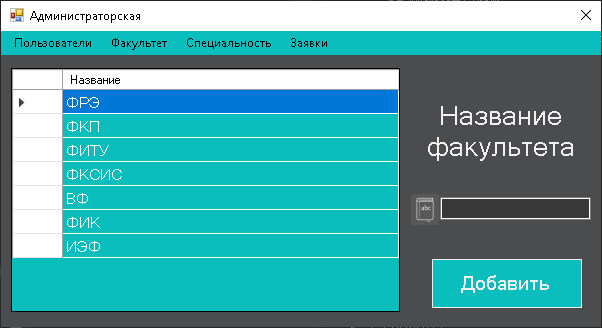


Рисунок 4.13 – Выбор поля «Добавить»

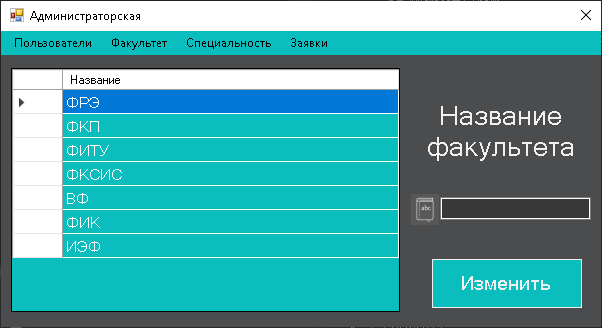


Рисунок 4.14 – Выбор поля «Редактировать»

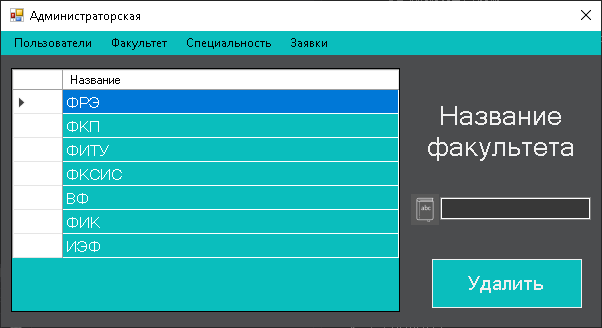


Рисунок 4.15 – Выбор поля «Удалить»

В случае добавления факультета и специальности, которые уже существуют в программе, появляется сообщение об существовании данного факультета или специальности (рисунок 4.16-4.17).

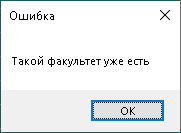


Рисунок 4.16 – Сообщение об существовании введенного факультета

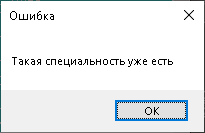


Рисунок 4.17 – Сообщение об существовании введенной специальности

При удалении факультета или специальности, как представлено на рисунке 4.15. Необходимо ввести название факультета, который существует. При условии ввода несуществующего факультета отображается сообщение об отсутствии данной специальности или факультета (рисунок 4.18-4.19).

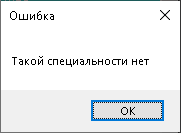


Рисунок 4.18 – Сообщение об отсутствии введенной специальности

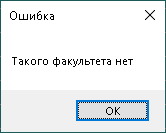


Рисунок 4.19 – Сообщение об отсутствии введенного факультета

Для редактирования специальности или факультета, необходимо нажать на поле «Редактировать», выбрать интересующую специальность и факультет, изменить значение текстового поля «Название специальности» и нажать на кнопку «Изменить» (рисунок 4.20).

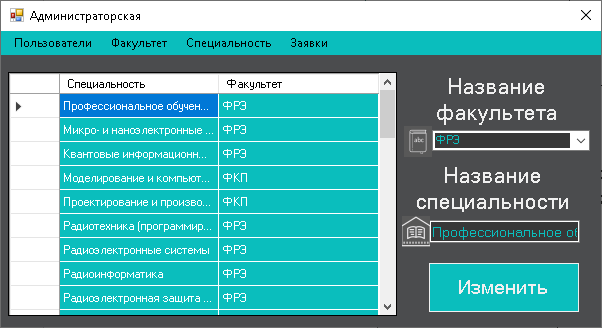


Рисунок 4.20 – Пример редактирования полей

Таким образом, программа является полностью работоспособной и правильно реагирует на возникновении исключительных ситуаций.

## **5 ОПИСАНИЕ ПРИМЕНЕНИЯ**

При открытии программы пользователя предоставляется главное окно с возможность выбора специальности и перехода в окно авторизации (рис. 5.1 и 5.2).

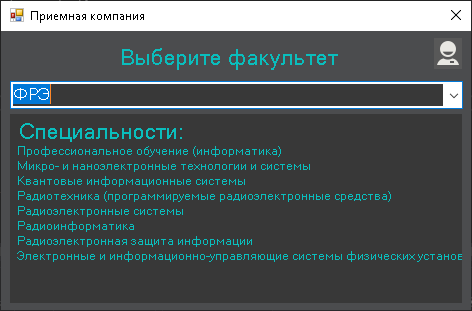


Рисунок 5.1 – Главное окно

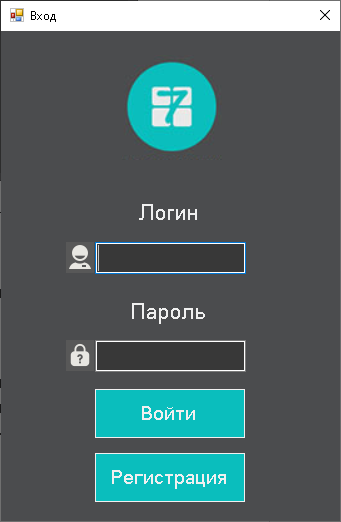


Рисунок 5.2 – Форма авторизации

В окне авторизации присутствует возможность входа в личную учетную запись и регистрации. Форма регистрации представлена в рисунке 5.3.

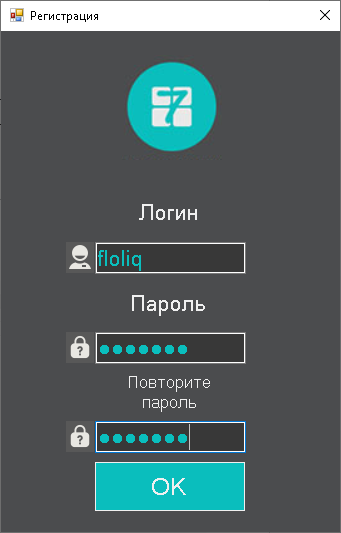


Рисунок 5.3 – Окно регистрации

После ввода логина и пароля открывается главное окно программы с возможность перехода в личный кабинет пользователя (рисунок 5.4). При нажатии на текст «Профиль» открывается личный кабинет пользователя (рисунок 5.5).

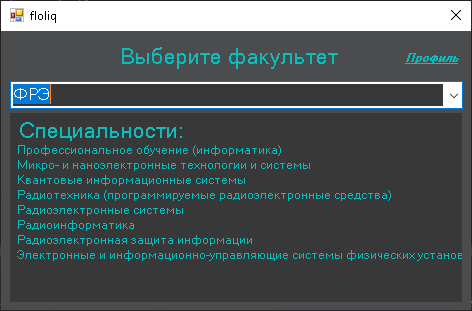


Рисунок 5.4 – Главное окно программы с возможность перехода в личный кабинет

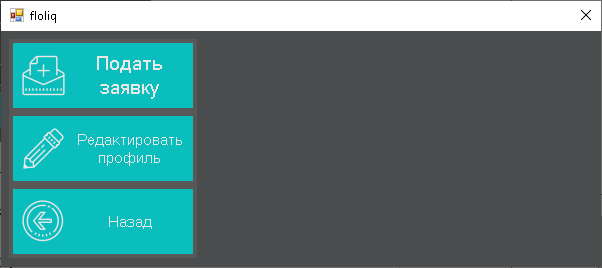


Рисунок 5.5 – Личный кабинет

В личном кабинете пользователя представлена возможность перехода в окно подачи заявки (рисунок 5.6), редактирования профиля (рисунок 5.7) и возврата на главное окно программы.

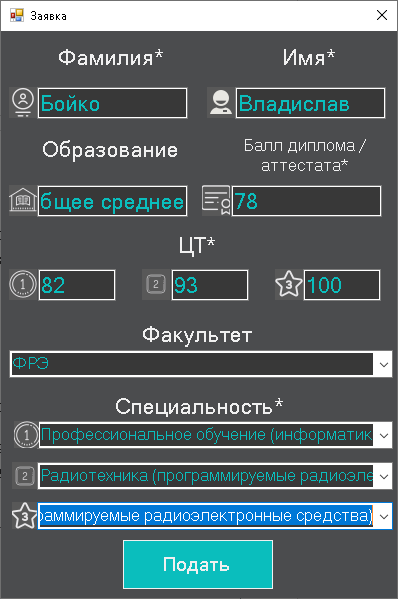


Рисунок 5.6 – Окно подачи заявки

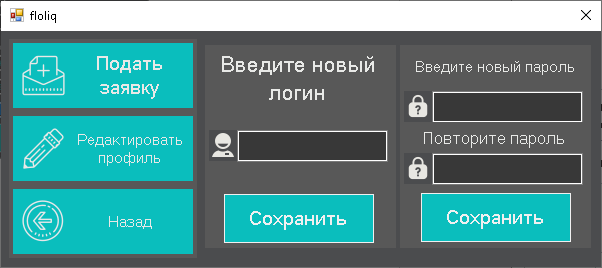


Рисунок 5.7 – Редактирование профиля

В случае авторизации пользователя с правами администратора открывается окно администратора, в котором возможен просмотр всех пользователей, факультетов, специальностей и заявок (рисунок 5.8).

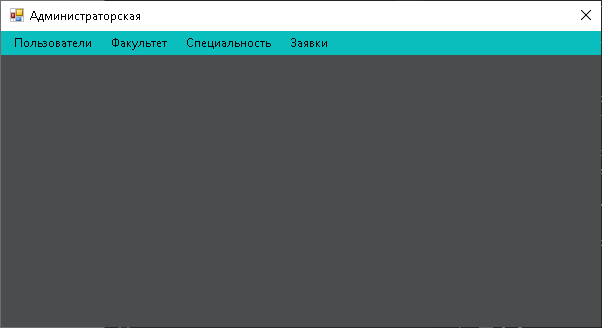


Рисунок 5.8 – Окно администратора

При нажатии на строку меню и выборах полей «Факультет» и «Специальности», открывается контекстное меню с выбором пунктов «Добавить», «Удалить», «Редактировать» (рисунок 5.9).

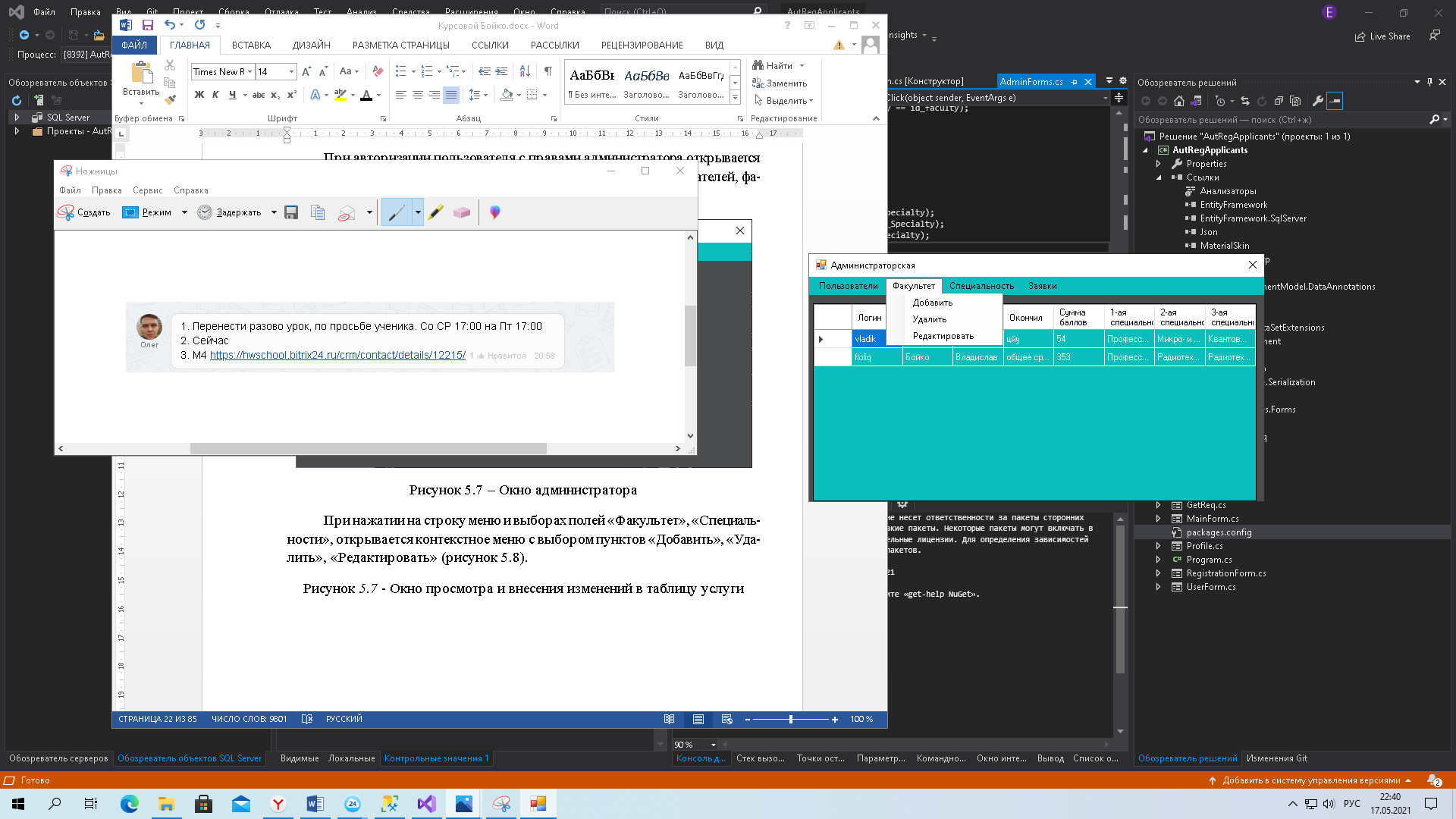


Рисунок 5.9 – Контекстное меню

При нажатии на одно из полей контекстного меню («Добавить», «Редактировать» или «Удалить») в строках меню «Факультет» и «Специальности», отображается выбранная информация, а также появляется возможность для добавления, удаления и редактирования записей (рисунки 5.10-5.12)

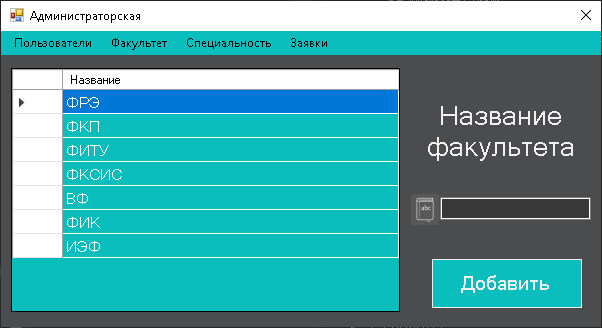


Рисунок 5.10 – Выбор поля «Добавить»

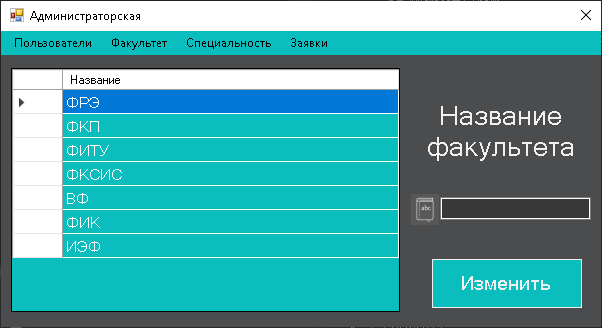


Рисунок 5.11 – Выбор поля «Редактировать»

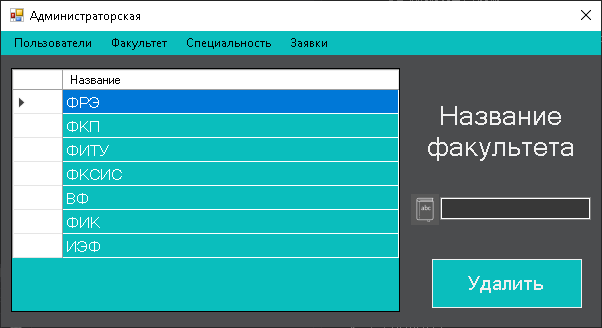


Рисунок 5.12 – Выбор поля «Удалить»

В случае нажатии на строку меню и выбора поля «Заявки» отображается информация о заявках, поданные абитуриентами (рисунок 5.13)

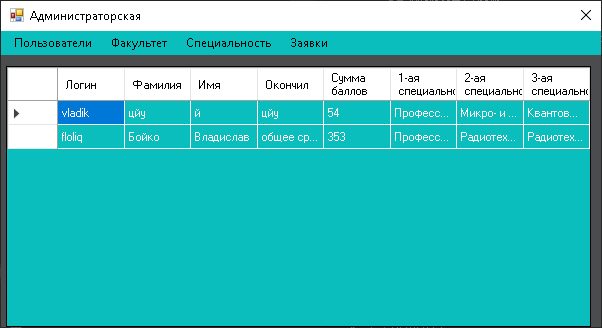


Рисунок 5.13 – Информация о заявках, поданные абитуриентами

При выборе строки меню «Пользователи» открывается контекстное меню с полями «Просмотр» и «Назначить администратором» (рисунок 5.14).

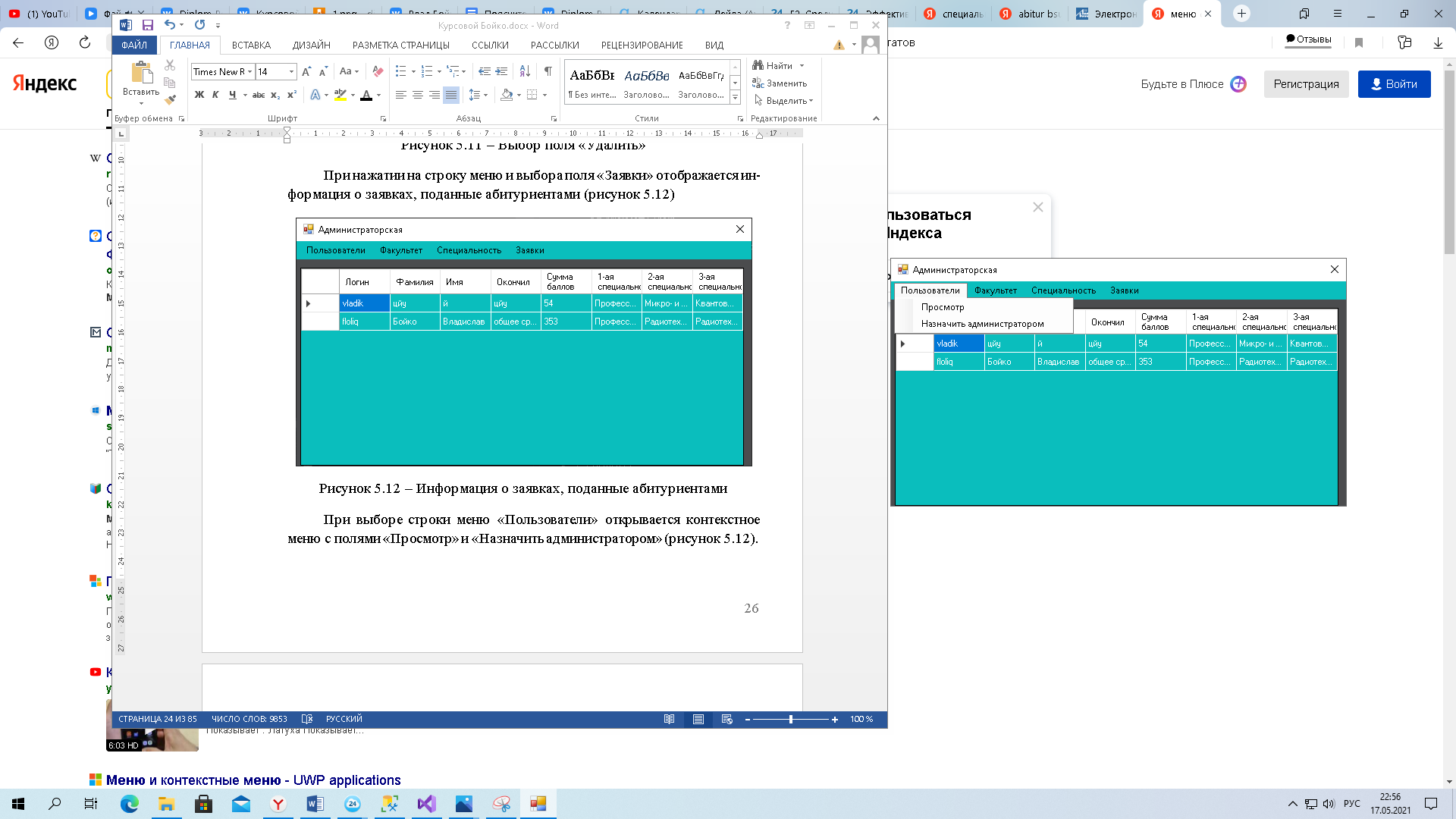
э

Рисунок 5.14 – Контекстное меню при выборе строки меню «Пользователи»

В случае выборе поля контекстного меню «Просмотр» в строке меню «Пользователи», администратору предоставляется возможно просмотреть всех пользователей, зарегистрированных в данной программе (рисунок 5.15). При выборе поля контекстного меню «Назначить администратором» в строке меню «Пользователи», предоставляется дополнительная информации для выбора логина пользователя, которому нужно передать права администратора (рисунок 5.16).

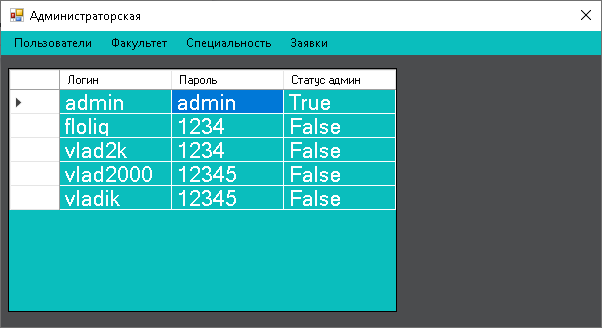


Рисунок 5.15 – Информация о всех зарегистрированных пользователях

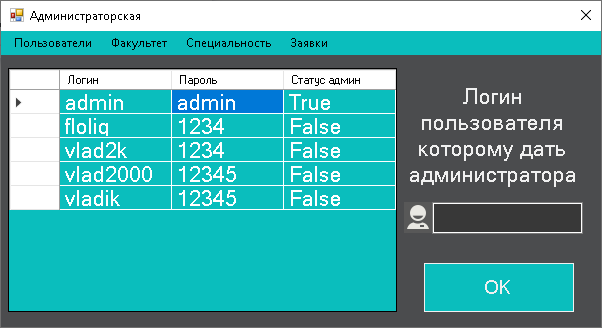


Рисунок 5.16 – Информация о всех зарегистрированных пользователях с возможностью предоставления прав администратора

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта были получены навыки работы с системой управления базами данных, проектирования и непосредственно разработки базы данных, а также подключения ее к приложению для персонального компьютера. Для выполнения работы было выбрано приложение для персонального компьютера.

В результате курсового проектирования был разработана автоматизированная система учета абитуриентов с широким функционалом.

В процессе проектирования и разработки данного программного модуля были задействованы язык программирования C#, Windows Forms, СУБД MS SQL Server.

После проектирования был написан код программы. С помощью SQL Server была создана полноценная база данных для системы учета абитуриентов.

Кроме того, было проведено тестирование программы, которое подтвердило правильность, работоспособность и эффективность программы.

Проект разработан в полном объеме и полностью соответствует поставленной цели и готов к использованию.

# **Список использованных источников**

[1] «Автоматизированная система зачисления поможет выбрать ВУЗ» –Режим доступа: https://www.sb.by/articles/avtomatizirovannaya-sistema-zachisleniya-pomozhet-vybrat-vuz.html

[2] «Руководство по Entity Framework Core» [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/entityframeworkcore/5.1.php

[3] «Создание приложения “База данных Книг”» [Электронный ресурс]. – Режим доступа: https://studlearn.com/works/details/baza-dannykh-knig-c-wpf-sqlite-868

[4] «Windows Forms documentation» [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/windows-forms-overview?view=netframeworkdesktop-4.8

[5] «C# Tutorials» [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/.

[6] «Metanit» [Электронный ресурс]. — Режим доступа: https://metanit.com/sharp.

# 

# **Приложение А**

*(обязательное)*

Схема базы данных

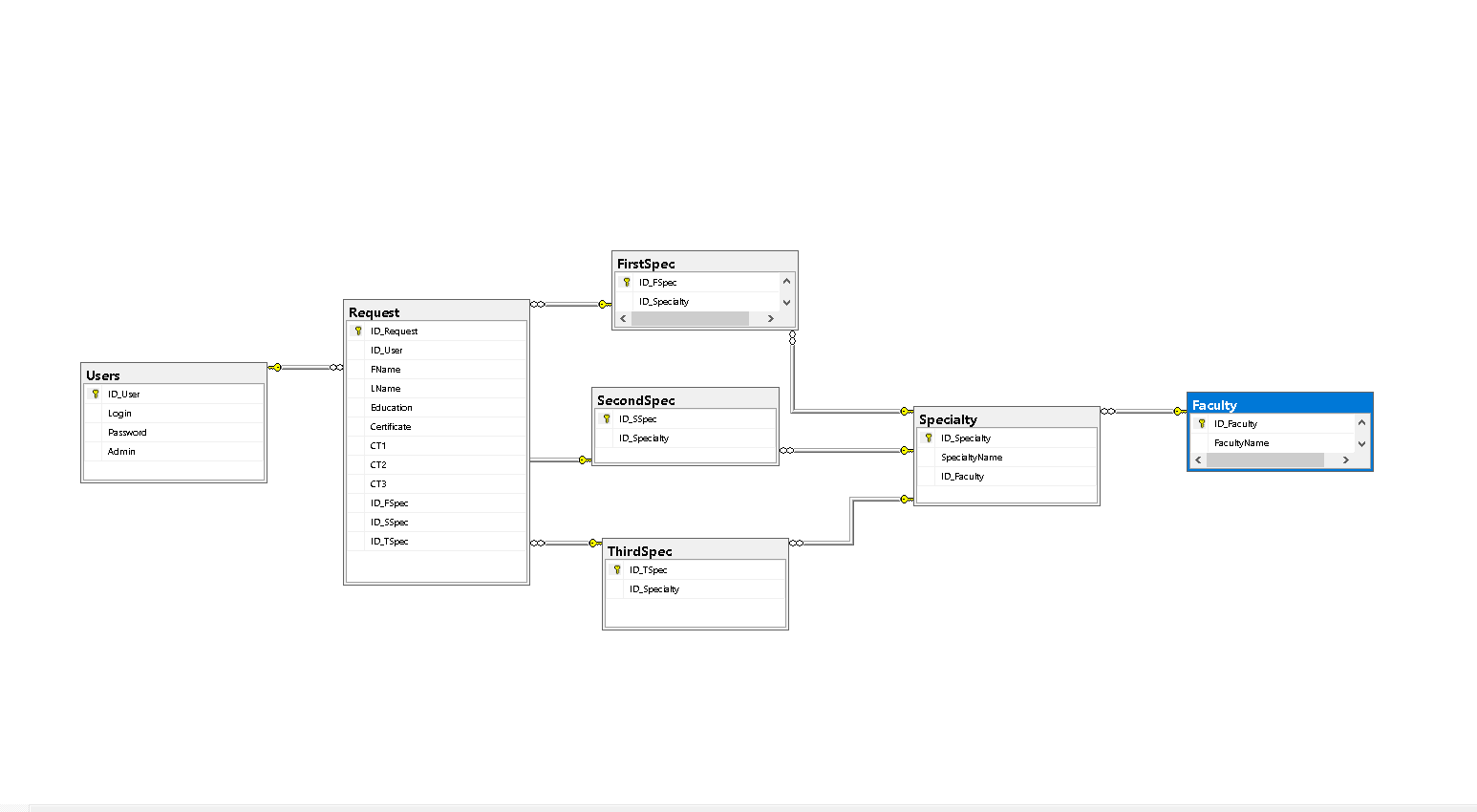


Рисунок *А.1* — Схема базы данных

# 

# **Приложение Б**

*(обязательное)*

Диаграмма вариантов использования

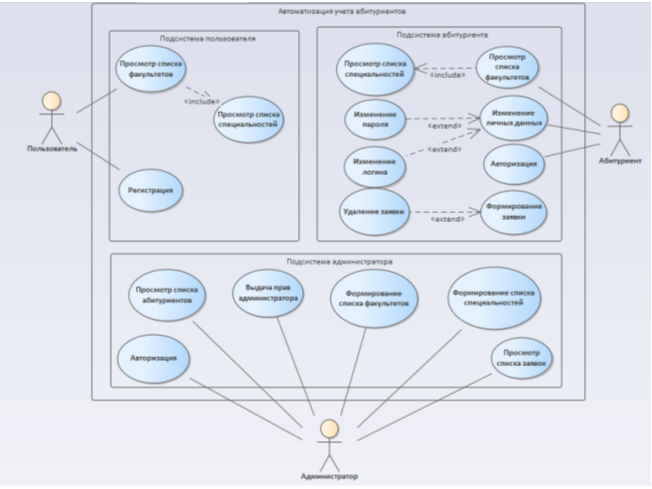


Рисунок *Б.1* — Диаграмма вариантов использования

# **Приложение В**

*(обязательное)*

Листинг кода

UserForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AutRegApplicants

{

public partial class UserForm : Form

{

public UserForm()

{

InitializeComponent();

this.BackColor = Color.FromArgb(75, 76, 78);

label1.BackColor = Color.FromArgb(75, 76, 78);

label4.BackColor = Color.FromArgb(75, 76, 78);

label1.ForeColor = Color.FromArgb(10, 190, 189);

label2.ForeColor = Color.FromArgb(10, 190, 189);

label3.ForeColor = Color.FromArgb(10, 190, 189);

label4.ForeColor = Color.FromArgb(10, 190, 189);

comboBox1.BackColor = Color.FromArgb(56, 56, 56);

comboBox1.ForeColor = Color.FromArgb(10, 190, 189);

panel1.BackColor = Color.FromArgb(56, 56, 56);

}

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

{

this.Text = UserData.UserName;

comboBox1.Items.Clear();

IEnumerable<Faculty> faculties = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

faculties = db.Faculty;

foreach (var i in faculties)

{

comboBox1.Items.Add(i.FacultyName.ToString());

}

IEnumerable<Users> users = db.Users;

foreach (var i in users)

{

if(i.Login.ToString()==this.Text)

{

UserData.UserID = i.ID\_User;

UserData.UserName = i.Login;

UserData.UserPassword = i.Password;

UserData.UserStatus = i.Admin;

}

}

}

}

private void comboBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = true;

}

private void comboBox1\_TextChanged(object sender, EventArgs e)

{

label2.Text = "";

label3.Text = "";

if (comboBox1.Text != "")

{

panel1.Visible = true;

label2.Text = "Специальности:";

}

IEnumerable<Specialty> specialties = null;

Faculty faculty = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

faculty = db.Faculty.FirstOrDefault(i => i.FacultyName == comboBox1.Text.ToString());

specialties = db.Specialty.Where(i => i.ID\_Faculty == faculty.ID\_Faculty);

foreach (var i in specialties)

{

label3.Text += i.SpecialtyName.ToString() + "\r";

}

}

}

private void label4\_Click(object sender, EventArgs e)

{

Profile profile = new Profile();

this.Hide();

profile.Show();

}

}

}

RegistrationForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AutRegApplicants

{

public partial class RegistrationForm : Form

{

public RegistrationForm()

{

InitializeComponent();

this.BackColor = Color.FromArgb(75, 76, 78);

label1.BackColor = Color.FromArgb(75, 76, 78);

label2.BackColor = Color.FromArgb(75, 76, 78);

label3.BackColor = Color.FromArgb(75, 76, 78);

textBox1.BackColor = Color.FromArgb(56, 56, 56);

textBox2.BackColor = Color.FromArgb(56, 56, 56);

textBox3.BackColor = Color.FromArgb(56, 56, 56);

textBox1.ForeColor = Color.FromArgb(10, 190, 189);

textBox2.ForeColor = Color.FromArgb(10, 190, 189);

textBox3.ForeColor = Color.FromArgb(10, 190, 189);

button2.BackColor = Color.FromArgb(10, 190, 189);

pictureBox2.BackColor = Color.FromArgb(88, 88, 88);

pictureBox3.BackColor = Color.FromArgb(88, 88, 88);

pictureBox4.BackColor = Color.FromArgb(88, 88, 88);

this.Text = "Регистрация";

}

private void button2\_Click(object sender, EventArgs e)

{

if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "")

{

string login = textBox1.Text;

if (textBox2.Text == textBox3.Text)

{

Users user = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

user = db.Users.FirstOrDefault(u => u.Login == login);

}

if(user!=null)

{

MessageBox.Show("Пользователь с таким логином уже зарегистрирован", "Ошибка");

}

else

{

UserData.UserName = login;

Users users = new Users

{

Login = textBox1.Text,

Password = textBox2.Text,

Admin = false

};

AutRegApplicantsEntities db = new AutRegApplicantsEntities();

db.Users.Add(users);

db.SaveChanges();

UserForm userForm = new UserForm();

this.Hide();

userForm.Show();

}

}

else

{

MessageBox.Show("Пароли не совпадают", "Ошибка");

}

}

else

{

MessageBox.Show("Не все поля заполнены", "Ошибка");

}

}

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

{

}

}

}

AutForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AutRegApplicants

{

public partial class AutForm : Form

{

public AutForm()

{

InitializeComponent();

this.BackColor = Color.FromArgb(75, 76, 78);

label1.BackColor = Color.FromArgb(75, 76, 78);

label2.BackColor = Color.FromArgb(75, 76, 78);

textBox1.BackColor = Color.FromArgb(56,56,56);

textBox2.BackColor = Color.FromArgb(56, 56, 56);

textBox1.ForeColor = Color.FromArgb(10, 190, 189);

textBox2.ForeColor = Color.FromArgb(10, 190, 189);

button1.BackColor = Color.FromArgb(10, 190, 189);

button2.BackColor = Color.FromArgb(10, 190, 189);

pictureBox2.BackColor = Color.FromArgb(88,88,88);

pictureBox3.BackColor = Color.FromArgb(88, 88, 88);

}

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

{

if (textBox1.Text != "" && textBox2.Text != "")

{

string login = textBox1.Text;

string password = textBox2.Text;

Users user = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

user = db.Users.FirstOrDefault(u => u.Login == login && u.Password == password);

}

if (user != null)

{

UserData.UserName = user.Login;

UserData.UserID = user.ID\_User;

UserData.UserStatus = user.Admin;

UserData.UserPassword = user.Password;

if (user.Admin == true)

{

AdminForms adminForms = new AdminForms();

this.Hide();

adminForms.Show();

}

else if (user.Admin == false)

{

UserForm userForm = new UserForm();

this.Hide();

userForm.Show();

}

}

else

{

MessageBox.Show("Такого пользователя нет", "Ошибка");

}

}

else

{

MessageBox.Show("Не все поля заполнены", "Ошибка");

}

}

private void button2\_Click(object sender, EventArgs e)

{

RegistrationForm registrationForm = new RegistrationForm();

this.Hide();

registrationForm.Show();

}

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

{

}

}

}

MainForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AutRegApplicants

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

this.BackColor = Color.FromArgb(75, 76, 78);

label1.BackColor = Color.FromArgb(75, 76, 78);

label1.ForeColor = Color.FromArgb(10, 190, 189);

label2.ForeColor = Color.FromArgb(10, 190, 189);

label3.ForeColor = Color.FromArgb(10, 190, 189);

comboBox1.BackColor = Color.FromArgb(56, 56, 56);

comboBox1.ForeColor = Color.FromArgb(10, 190, 189);

panel1.BackColor = Color.FromArgb(56, 56, 56);

pictureBox1.BackColor = Color.FromArgb(88, 88, 88);

}

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

{

comboBox1.Items.Clear();

IEnumerable<Faculty> faculties = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

faculties = db.Faculty;

foreach (var i in faculties)

{

comboBox1.Items.Add(i.FacultyName.ToString());

}

}

}

private void comboBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = true;

}

private void comboBox1\_TextChanged(object sender, EventArgs e)

{

label2.Text = "";

label3.Text = "";

if (comboBox1.Text!="")

{

panel1.Visible = true;

label2.Text = "Специальности:";

}

IEnumerable<Specialty> specialties = null;

Faculty faculty = null;

using (AutRegApplicantsEntities db=new AutRegApplicantsEntities())

{

faculty = db.Faculty.FirstOrDefault(i => i.FacultyName == comboBox1.Text.ToString());

specialties = db.Specialty.Where(i => i.ID\_Faculty == faculty.ID\_Faculty);

foreach(var i in specialties)

{

label3.Text += i.SpecialtyName.ToString() + "\r";

}

}

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

AutForm autForm = new AutForm();

this.Hide();

autForm.Show();

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

}

}

}

Profile

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AutRegApplicants

{

public partial class Profile : Form

{

public void RegForm()

{

Request request = null;

using (AutRegApplicantsEntities db=new AutRegApplicantsEntities())

{

request = db.Request.FirstOrDefault(c => c.ID\_User == UserData.UserID);

}

if (request != null)

{

DialogResult dialogResult=MessageBox.Show(

"Вы уже подали заявку, хотете удалить и подать заного?",

"Ошибка",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

if(dialogResult==DialogResult.Yes)

{

AutRegApplicantsEntities db = new AutRegApplicantsEntities();

Request request1 = db.Request.FirstOrDefault(c => c.ID\_User == UserData.UserID);

db.Request.Remove(request1);

db.SaveChanges();

GetReq getReq = new GetReq();

this.Hide();

getReq.Show();

}

}

else

{

GetReq getReq = new GetReq();

this.Hide();

getReq.Show();

}

}

public void BackForm()

{

UserForm userForm = new UserForm();

this.Hide();

userForm.Show();

}

public void EditUser()

{

panel5.Visible = true;

panel5.Location = new Point(200, 10);

}

public Profile()

{

InitializeComponent();

this.BackColor = Color.FromArgb(75, 76, 78);

pictureBox1.BackColor = Color.FromArgb(10, 190, 189);

pictureBox2.BackColor = Color.FromArgb(10, 190, 189);

pictureBox3.BackColor = Color.FromArgb(10, 190, 189);

pictureBox4.BackColor = Color.FromArgb(75, 76, 78);

pictureBox5.BackColor = Color.FromArgb(75, 76, 78);

pictureBox6.BackColor = Color.FromArgb(75, 76, 78);

panel1.BackColor = Color.FromArgb(88, 88, 88);

panel6.BackColor = Color.FromArgb(88, 88, 88);

panel7.BackColor = Color.FromArgb(88, 88, 88);

panel2.BackColor = Color.FromArgb(10, 190, 189);

panel3.BackColor = Color.FromArgb(10, 190, 189);

panel4.BackColor = Color.FromArgb(10, 190, 189);

label1.BackColor = Color.FromArgb(10, 190, 189);

label2.BackColor = Color.FromArgb(10, 190, 189);

label3.BackColor = Color.FromArgb(10, 190, 189);

label1.ForeColor = Color.FromArgb(232, 233, 231);

label2.ForeColor = Color.FromArgb(232, 233, 231);

label3.ForeColor = Color.FromArgb(232, 233, 231);

button1.BackColor = Color.FromArgb(10, 190, 189);

textBox1.BackColor = Color.FromArgb(56, 56, 56);

textBox1.ForeColor = Color.FromArgb(10, 190, 189);

label4.BackColor = Color.FromArgb(88, 88, 88);

label4.ForeColor = Color.FromArgb(232, 233, 231);

label5.BackColor = Color.FromArgb(88, 88, 88);

label5.ForeColor = Color.FromArgb(232, 233, 231);

label6.BackColor = Color.FromArgb(88, 88, 88);

label6.ForeColor = Color.FromArgb(232, 233, 231);

textBox2.BackColor = Color.FromArgb(56, 56, 56);

textBox2.ForeColor = Color.FromArgb(10, 190, 189);

textBox3.BackColor = Color.FromArgb(56, 56, 56);

textBox3.ForeColor = Color.FromArgb(10, 190, 189);

button2.BackColor = Color.FromArgb(10, 190, 189);

}

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

{

this.Text = UserData.UserName;

}

private void panel4\_Click(object sender, EventArgs e)

{

BackForm();

}

private void label3\_Click(object sender, EventArgs e)

{

BackForm();

}

private void panel3\_Click(object sender, EventArgs e)

{

EditUser();

}

private void label2\_Click(object sender, EventArgs e)

{

EditUser();

}

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

{

if (textBox1.Text != "")

{

Users user = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

user = db.Users.FirstOrDefault(u => u.Login == textBox1.Text);

}

if (user != null)

{

MessageBox.Show("Пользователь с таким логином уже зарегистрирован", "Ошибка");

}

else

{

AutRegApplicantsEntities db = new AutRegApplicantsEntities();

var users = db.Users.Where(c => c.Login == UserData.UserName).FirstOrDefault();

users.Login = textBox1.Text.ToString();

UserData.UserName = textBox1.Text.ToString();

db.SaveChanges();

this.Text = textBox1.Text;

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

panel5.Location = new Point(1200, 10);

}

}

else

MessageBox.Show("Поле для нового логина не заполнено", "Ошибка");

}

private void button2\_Click(object sender, EventArgs e)

{

if (textBox2.Text==""|| textBox3.Text == "")

MessageBox.Show("Не все поля заполнены", "Ошибка");

else

if (textBox2.Text != textBox3.Text)

MessageBox.Show("Пароли не совпадают", "Ошибка");

else

{

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

var users = db.Users.Where(c => c.Login == UserData.UserName).FirstOrDefault();

users.Password = textBox2.Text.ToString();

db.SaveChanges();

UserData.UserPassword = textBox2.Text.ToString();

textBox2.Text = "";

textBox3.Text = "";

textBox1.Text = "";

panel5.Location = new Point(1200, 10);

}

}

}

private void label1\_Click(object sender, EventArgs e)

{

RegForm();

}

private void panel2\_Click(object sender, EventArgs e)

{

RegForm();

}

}

}

GetReq

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AutRegApplicants

{

public partial class GetReq : Form

{

public GetReq()

{

InitializeComponent();

this.BackColor = Color.FromArgb(75, 76, 78);

button1.BackColor = Color.FromArgb(10, 190, 189);

pictureBox1.BackColor = Color.FromArgb(88, 88, 88);

pictureBox2.BackColor = Color.FromArgb(88, 88, 88);

pictureBox3.BackColor = Color.FromArgb(88, 88, 88);

pictureBox4.BackColor = Color.FromArgb(88, 88, 88);

pictureBox5.BackColor = Color.FromArgb(88, 88, 88);

pictureBox6.BackColor = Color.FromArgb(88, 88, 88);

pictureBox7.BackColor = Color.FromArgb(88, 88, 88);

pictureBox8.BackColor = Color.FromArgb(88, 88, 88);

pictureBox9.BackColor = Color.FromArgb(88, 88, 88);

pictureBox10.BackColor = Color.FromArgb(88, 88, 88);

label1.BackColor = Color.FromArgb(75, 76, 78);

label2.BackColor = Color.FromArgb(75, 76, 78);

label3.BackColor = Color.FromArgb(75, 76, 78);

label4.BackColor = Color.FromArgb(75, 76, 78);

label5.BackColor = Color.FromArgb(75, 76, 78);

label6.BackColor = Color.FromArgb(75, 76, 78);

label7.BackColor = Color.FromArgb(75, 76, 78);

textBox1.BackColor = Color.FromArgb(56, 56, 56);

textBox1.ForeColor = Color.FromArgb(10, 190, 189);

textBox2.BackColor = Color.FromArgb(56, 56, 56);

textBox2.ForeColor = Color.FromArgb(10, 190, 189);

textBox3.BackColor = Color.FromArgb(56, 56, 56);

textBox3.ForeColor = Color.FromArgb(10, 190, 189);

textBox4.BackColor = Color.FromArgb(56, 56, 56);

textBox4.ForeColor = Color.FromArgb(10, 190, 189);

textBox5.BackColor = Color.FromArgb(56, 56, 56);

textBox5.ForeColor = Color.FromArgb(10, 190, 189);

textBox6.BackColor = Color.FromArgb(56, 56, 56);

textBox6.ForeColor = Color.FromArgb(10, 190, 189);

textBox7.BackColor = Color.FromArgb(56, 56, 56);

textBox7.ForeColor = Color.FromArgb(10, 190, 189);

comboBox1.BackColor = Color.FromArgb(56, 56, 56);

comboBox1.ForeColor = Color.FromArgb(10, 190, 189);

comboBox2.BackColor = Color.FromArgb(56, 56, 56);

comboBox2.ForeColor = Color.FromArgb(10, 190, 189);

comboBox3.BackColor = Color.FromArgb(56, 56, 56);

comboBox3.ForeColor = Color.FromArgb(10, 190, 189);

comboBox4.BackColor = Color.FromArgb(56, 56, 56);

comboBox4.ForeColor = Color.FromArgb(10, 190, 189);

}

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

{

comboBox1.Items.Clear();

comboBox2.Items.Clear();

comboBox3.Items.Clear();

comboBox4.Items.Clear();

IEnumerable<Faculty> faculties = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

faculties = db.Faculty;

foreach (var i in faculties)

{

comboBox1.Items.Add(i.FacultyName.ToString());

}

}

}

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

{

if ((textBox1.Text == "") || (textBox2.Text == "") || (textBox3.Text == "") || (textBox4.Text == "") || (textBox5.Text == "") || (textBox6.Text == "") || (textBox7.Text == "") || (comboBox2.Text == ""))

{

MessageBox.Show("Не все обязательные поля заполнены", "Ошибка");

}

else

{

if (Convert.ToInt32(textBox4.Text) < 1 || Convert.ToInt32(textBox4.Text) > 100)

{

MessageBox.Show("Не правильный балл диплома/аттестата", "Ошибка");

}

else if (Convert.ToInt32(textBox5.Text) < 1 || Convert.ToInt32(textBox5.Text) > 100 || Convert.ToInt32(textBox6.Text) < 1 || Convert.ToInt32(textBox6.Text) > 100 || Convert.ToInt32(textBox7.Text) < 1 || Convert.ToInt32(textBox7.Text) > 100)

{

MessageBox.Show("Не правильный балл ЦТ", "Ошибка");

}

else

{

AutRegApplicantsEntities db = new AutRegApplicantsEntities();

IEnumerable<Specialty> specialties = db.Specialty;

IEnumerable<FirstSpec> firstSpecs = db.FirstSpec;

IEnumerable<SecondSpec> secondSpecs = db.SecondSpec;

IEnumerable<ThirdSpec> thirdSpecs = db.ThirdSpec;

string s1 = "";

string s2 = "";

string s3 = "";

foreach (var i in specialties)

{

if(i.SpecialtyName==comboBox2.Text)

{

foreach (var j in firstSpecs)

{

if(j.ID\_Specialty==i.ID\_Specialty)

{

s1 = j.ID\_Specialty.ToString();

}

}

}

}

if(comboBox3.Text!="")

{

foreach (var i in specialties)

{

if (i.SpecialtyName == comboBox3.Text)

{

foreach (var j in secondSpecs)

{

if (j.ID\_Specialty == i.ID\_Specialty)

{

s2 = j.ID\_Specialty.ToString();

}

}

}

}

}

if (comboBox4.Text != "")

{

foreach (var i in specialties)

{

if (i.SpecialtyName == comboBox4.Text)

{

foreach (var j in thirdSpecs)

{

if (j.ID\_Specialty == i.ID\_Specialty)

{

s3 = j.ID\_Specialty.ToString();

}

}

}

}

}

if(s2==""||s3=="")

{

if(s2=="")

{

Request request = new Request

{

ID\_User = UserData.UserID,

FName = textBox1.Text.ToString(),

LName = textBox2.Text.ToString(),

Education = textBox3.Text.ToString(),

Сertificate = Convert.ToInt32(textBox4.Text),

CT1 = Convert.ToInt32(textBox5.Text),

CT2 = Convert.ToInt32(textBox6.Text),

CT3 = Convert.ToInt32(textBox7.Text),

ID\_FSpec = Convert.ToInt32(s1),

};

db.Request.Add(request);

db.SaveChanges();

}

else

{

Request request = new Request

{

ID\_User = UserData.UserID,

FName = textBox1.Text.ToString(),

LName = textBox2.Text.ToString(),

Education = textBox3.Text.ToString(),

Сertificate = Convert.ToInt32(textBox4.Text),

CT1 = Convert.ToInt32(textBox5.Text),

CT2 = Convert.ToInt32(textBox6.Text),

CT3 = Convert.ToInt32(textBox7.Text),

ID\_FSpec = Convert.ToInt32(s1),

ID\_SSpec = Convert.ToInt32(s2)

};

db.Request.Add(request);

db.SaveChanges();

}

}

else

{

Request request = new Request

{

ID\_User = UserData.UserID,

FName = textBox1.Text.ToString(),

LName = textBox2.Text.ToString(),

Education = textBox3.Text.ToString(),

Сertificate = Convert.ToInt32(textBox4.Text),

CT1 = Convert.ToInt32(textBox5.Text),

CT2 = Convert.ToInt32(textBox6.Text),

CT3 = Convert.ToInt32(textBox7.Text),

ID\_FSpec = Convert.ToInt32(s1),

ID\_SSpec = Convert.ToInt32(s2),

ID\_TSpec = Convert.ToInt32(s3)

};

db.Request.Add(request);

db.SaveChanges();

}

Profile profile = new Profile();

this.Hide();

profile.Show();

}

}

}

private void comboBox1\_TextChanged(object sender, EventArgs e)

{

comboBox2.Items.Clear();

comboBox3.Items.Clear();

comboBox4.Items.Clear();

IEnumerable<Specialty> specialties = null;

Faculty faculty = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

faculty = db.Faculty.FirstOrDefault(i => i.FacultyName == comboBox1.Text.ToString());

specialties = db.Specialty.Where(i => i.ID\_Faculty == faculty.ID\_Faculty);

foreach (var j in specialties)

{

comboBox2.Items.Add(j.SpecialtyName.ToString());

}

}

}

private void comboBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = true;

}

private void comboBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = true;

}

private void comboBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = true;

}

private void comboBox4\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = true;

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

char l = e.KeyChar;

if ((l < 'А' || l > 'я') && l != '\b' && l != '.')

{

e.Handled = true;

}

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

char l = e.KeyChar;

if ((l < 'А' || l > 'я') && l != '\b' && l != '.')

{

e.Handled = true;

}

}

private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

char l = e.KeyChar;

if ((l < 'А' || l > 'я') && l != '\b' && l != '.' && l != ' ')

{

e.Handled = true;

}

}

private void textBox4\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if ((e.KeyChar <= 47 || e.KeyChar >= 58) && number != 8 && number != 44) //цифры, клавиша BackSpace и запятая а ASCII

{

e.Handled = true;

}

}

private void textBox5\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if ((e.KeyChar <= 47 || e.KeyChar >= 58) && number != 8 ) //цифры, клавиша BackSpace и запятая а ASCII

{

e.Handled = true;

}

}

private void textBox6\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if ((e.KeyChar <= 47 || e.KeyChar >= 58) && number != 8) //цифры, клавиша BackSpace и запятая а ASCII

{

e.Handled = true;

}

}

private void textBox7\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if ((e.KeyChar <= 47 || e.KeyChar >= 58) && number != 8) //цифры, клавиша BackSpace и запятая а ASCII

{

e.Handled = true;

}

}

private void comboBox2\_TextChanged(object sender, EventArgs e)

{

comboBox3.Items.Clear();

IEnumerable<Specialty> specialties = null;

Faculty faculty = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

faculty = db.Faculty.FirstOrDefault(i => i.FacultyName == comboBox1.Text.ToString());

specialties = db.Specialty.Where(i => i.ID\_Faculty == faculty.ID\_Faculty);

foreach (var j in specialties)

{

comboBox3.Items.Add(j.SpecialtyName.ToString());

}

}

}

private void comboBox3\_TextChanged(object sender, EventArgs e)

{

comboBox4.Items.Clear();

IEnumerable<Specialty> specialties = null;

Faculty faculty = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

faculty = db.Faculty.FirstOrDefault(i => i.FacultyName == comboBox1.Text.ToString());

specialties = db.Specialty.Where(i => i.ID\_Faculty == faculty.ID\_Faculty);

foreach (var j in specialties)

{

comboBox4.Items.Add(j.SpecialtyName.ToString());

}

}

}

}

}

AdminForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace AutRegApplicants

{

public partial class AdminForms : Form

{

public void PrintUser()

{

IEnumerable<Users> users = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

users = db.Users;

dataGridView1.ColumnCount = 3;

dataGridView1.RowCount = 1;

dataGridView1.RowCount = dataGridView1.RowCount + 1;

int k = 0;

foreach (DataGridViewColumn row in dataGridView1.Columns)

{

if (k == 0)

{

row.HeaderCell.Value = "Логин";

}

if (k == 1)

{

row.HeaderCell.Value = "Пароль";

}

if (k == 2)

{

row.HeaderCell.Value = "Статус админ";

}

k++;

}

k = 0;

foreach (var i in users)

{

dataGridView1[0, k].Value = i.Login.ToString();

dataGridView1[1, k].Value = i.Password.ToString();

dataGridView1[2, k].Value = i.Admin.ToString();

dataGridView1.RowCount = dataGridView1.RowCount + 1;

k++;

}

dataGridView1.RowCount = dataGridView1.RowCount - 2;

}

}

public void PrintFaculty()

{

IEnumerable<Faculty> faculties = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

faculties = db.Faculty;

dataGridView2.ColumnCount = 1;

dataGridView2.RowCount = 1;

dataGridView2.RowCount = dataGridView2.RowCount + 1;

int k = 0;

foreach (DataGridViewColumn row in dataGridView2.Columns)

{

if (k == 0)

{

row.HeaderCell.Value = "Название";

}

}

k = 0;

foreach (var i in faculties)

{

dataGridView2[0, k].Value = i.FacultyName.ToString();

dataGridView2.RowCount = dataGridView2.RowCount + 1;

k++;

}

dataGridView2.RowCount = dataGridView2.RowCount - 2;

}

}

public void PrintSpec()

{

IEnumerable<Specialty> specialties = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

specialties = db.Specialty;

dataGridView3.ColumnCount = 2;

dataGridView3.RowCount = 1;

dataGridView3.RowCount = dataGridView3.RowCount + 1;

int k = 0;

foreach (DataGridViewColumn row in dataGridView3.Columns)

{

if (k == 0)

{

row.HeaderCell.Value = "Специальность";

}

if (k == 1)

{

row.HeaderCell.Value = "Факультет";

}

k++;

}

k = 0;

IEnumerable<Faculty> faculty = db.Faculty;

foreach (var i in specialties)

{

dataGridView3[0, k].Value = i.SpecialtyName.ToString();

foreach (var j in faculty)

{

if(i.ID\_Faculty==j.ID\_Faculty)

{

dataGridView3[1, k].Value = j.FacultyName.ToString();

}

}

dataGridView3.RowCount = dataGridView3.RowCount + 1;

k++;

}

dataGridView3.RowCount = dataGridView3.RowCount - 2;

}

}

public AdminForms()

{

InitializeComponent();

this.BackColor = Color.FromArgb(75, 76, 78);

menuStrip1.BackColor = Color.FromArgb(10, 190, 189);

dataGridView1.BackgroundColor = Color.FromArgb(10, 190, 189);

dataGridView1.DefaultCellStyle.BackColor=Color.FromArgb(10, 190, 189);

dataGridView1.DefaultCellStyle.ForeColor = Color.FromArgb(255,255,255);

dataGridView1.GridColor = Color.FromArgb(255, 255, 255);

dataGridView2.BackgroundColor = Color.FromArgb(10, 190, 189);

dataGridView2.DefaultCellStyle.BackColor = Color.FromArgb(10, 190, 189);

dataGridView2.DefaultCellStyle.ForeColor = Color.FromArgb(255, 255, 255);

dataGridView2.GridColor = Color.FromArgb(255, 255, 255);

dataGridView3.BackgroundColor = Color.FromArgb(10, 190, 189);

dataGridView3.DefaultCellStyle.BackColor = Color.FromArgb(10, 190, 189);

dataGridView3.DefaultCellStyle.ForeColor = Color.FromArgb(255, 255, 255);

dataGridView3.GridColor = Color.FromArgb(255, 255, 255);

dataGridView4.BackgroundColor = Color.FromArgb(10, 190, 189);

dataGridView4.DefaultCellStyle.BackColor = Color.FromArgb(10, 190, 189);

dataGridView4.DefaultCellStyle.ForeColor = Color.FromArgb(255, 255, 255);

dataGridView4.GridColor = Color.FromArgb(255, 255, 255);

label1.BackColor = Color.FromArgb(75, 76, 78);

textBox1.BackColor = Color.FromArgb(56, 56, 56);

textBox1.ForeColor = Color.FromArgb(10, 190, 189);

button1.BackColor = Color.FromArgb(10, 190, 189);

pictureBox2.BackColor = Color.FromArgb(88, 88, 88);

label2.BackColor = Color.FromArgb(75, 76, 78);

textBox2.BackColor = Color.FromArgb(56, 56, 56);

textBox2.ForeColor = Color.FromArgb(10, 190, 189);

button2.BackColor = Color.FromArgb(10, 190, 189);

pictureBox1.BackColor = Color.FromArgb(88, 88, 88);

label3.BackColor = Color.FromArgb(75, 76, 78);

label4.BackColor = Color.FromArgb(75, 76, 78);

textBox4.BackColor = Color.FromArgb(56, 56, 56);

textBox4.ForeColor = Color.FromArgb(10, 190, 189);

button3.BackColor = Color.FromArgb(10, 190, 189);

pictureBox3.BackColor = Color.FromArgb(88, 88, 88);

pictureBox4.BackColor = Color.FromArgb(88, 88, 88);

comboBox1.BackColor = Color.FromArgb(56, 56, 56);

comboBox1.ForeColor = Color.FromArgb(10, 190, 189);

dataGridView1.ReadOnly = true;

dataGridView2.ReadOnly = true;

dataGridView3.ReadOnly = true;

dataGridView4.ReadOnly = true;

}

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

{

}

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

{

label1.Visible = false;

textBox1.Visible = false;

pictureBox2.Visible = false;

button1.Visible = false;

panel1.Location = new Point(5, 35);

panel2.Location = new Point(5, 2035);

panel3.Location = new Point(5, 3035);

panel4.Location = new Point(5, 4035);

PrintUser();

}

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

{

if(textBox1.Text=="")

{

MessageBox.Show("Введите имя пользователя","Ошибка");

}

else

{

using (AutRegApplicantsEntities db=new AutRegApplicantsEntities())

{

Users user = null;

user = db.Users.FirstOrDefault(c => c.Login == textBox1.Text);

if(user!=null)

{

if (user.Admin == true)

{

MessageBox.Show("Пользователь уже является администратором", "Ошибка");

}

else

{

user.Admin = true;

db.SaveChanges();

}

}

else

{

MessageBox.Show("Такого пользователя нет", "Ошибка");

}

}

}

PrintUser();

textBox1.Text = "";

}

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

{

label1.Visible = true;

textBox1.Visible = true;

pictureBox2.Visible = true;

button1.Visible = true;

panel1.Location = new Point(5, 35);

panel2.Location = new Point(5, 2035);

panel3.Location = new Point(5, 3035);

panel4.Location = new Point(5, 4035);

PrintUser();

}

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

{

PrintFaculty();

panel2.Location = new Point(5, 35);

panel1.Location = new Point(5, 1035);

panel3.Location = new Point(5, 3035);

panel4.Location = new Point(5, 4035);

button2.Text = "Добавить";

}

private void button2\_Click(object sender, EventArgs e)

{

if(button2.Text=="Добавить")

{

if (textBox2.Text != "")

{

AutRegApplicantsEntities db = new AutRegApplicantsEntities();

Faculty faculty = null;

faculty = db.Faculty.FirstOrDefault(c => c.FacultyName == textBox2.Text);

if (faculty == null)

{

Faculty faculty1 = new Faculty

{

FacultyName = textBox2.Text.ToString()

};

db.Faculty.Add(faculty1);

db.SaveChanges();

}

else

{

MessageBox.Show("Такой факультет уже есть", "Ошибка");

}

textBox2.Text = "";

PrintFaculty();

}

else

{

MessageBox.Show("Введите название факультета", "Ошибка");

}

}

else if(button2.Text == "Удалить")

{

if (textBox2.Text != "")

{

AutRegApplicantsEntities db = new AutRegApplicantsEntities();

Faculty faculty = null;

faculty = db.Faculty.FirstOrDefault(c => c.FacultyName == textBox2.Text);

if (faculty == null)

{

MessageBox.Show("Такого факультета нет", "Ошибка");

}

else

{

db.Faculty.Remove(faculty);

db.SaveChanges();

}

textBox2.Text = "";

PrintFaculty();

}

else

{

MessageBox.Show("Введите название факультета", "Ошибка");

}

}

else if(button2.Text == "Изменить")

{

if (textBox2.Text != "")

{

AutRegApplicantsEntities db = new AutRegApplicantsEntities();

Faculty faculty = null;

int id = dataGridView2.CurrentCell.RowIndex;

//textBox2.Text = dataGridView2[0, id].Value.ToString();

faculty = db.Faculty.FirstOrDefault(c => c.FacultyName == textBox2.Text);

if (faculty == null)

{

string fak = dataGridView2[0, id].Value.ToString();

Faculty faculty1 = db.Faculty.FirstOrDefault(c => c.FacultyName == fak);

faculty1.FacultyName = textBox2.Text.ToString();

db.SaveChanges();

}

else

{

MessageBox.Show("Такой факультет уже есть", "Ошибка");

}

textBox2.Text = "";

PrintFaculty();

}

else

{

MessageBox.Show("Выберите факультет", "Ошибка");

}

}

}

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

{

PrintFaculty();

panel2.Location = new Point(5, 35);

panel1.Location = new Point(5, 1035);

panel3.Location = new Point(5, 3035);

panel4.Location = new Point(5, 4035);

button2.Text = "Удалить";

}

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

{

PrintFaculty();

panel2.Location = new Point(5, 35);

panel1.Location = new Point(5, 1035);

panel3.Location = new Point(5, 2035);

panel4.Location = new Point(5, 4035);

button2.Text = "Изменить";

}

private void dataGridView2\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridView2\_Click(object sender, EventArgs e)

{

if (button2.Text == "Изменить")

{

int id = dataGridView2.CurrentCell.RowIndex;

textBox2.Text = dataGridView2[0, id].Value.ToString();

}

}

private void редактироватьToolStripMenuItem1\_Click(object sender, EventArgs e)

{

PrintSpec();

panel2.Location = new Point(5, 2035);

panel1.Location = new Point(5, 1035);

panel3.Location = new Point(5, 35);

panel4.Location = new Point(5, 4035);

button3.Text = "Добавить";

comboBox1.Items.Clear();

IEnumerable<Faculty> faculties = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

faculties = db.Faculty;

foreach (var i in faculties)

{

comboBox1.Items.Add(i.FacultyName.ToString());

}

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (button3.Text == "Добавить")

{

if (comboBox1.Text != "")

{

if (textBox4.Text != "")

{

AutRegApplicantsEntities db = new AutRegApplicantsEntities();

Specialty specialty = null;

specialty = db.Specialty.FirstOrDefault(c => c.SpecialtyName == textBox4.Text);

if (specialty == null)

{

IEnumerable<Faculty> faculties = db.Faculty;

int id\_faculty = 0;

foreach (var i in faculties)

{

if (i.FacultyName == comboBox1.Text)

id\_faculty = i.ID\_Faculty;

}

Specialty specialty1 = new Specialty

{

SpecialtyName = textBox4.Text.ToString(),

ID\_Faculty = id\_faculty

};

db.Specialty.Add(specialty1);

db.SaveChanges();

IEnumerable <Specialty> specialties= db.Specialty;

int id\_spec = 0;

foreach (var i in specialties)

{

if (i.SpecialtyName == textBox4.Text)

id\_spec = i.ID\_Specialty;

}

FirstSpec firstSpec = new FirstSpec

{

ID\_Specialty = id\_spec

};

db.FirstSpec.Add(firstSpec);

db.SaveChanges();

SecondSpec secondSpec = new SecondSpec

{

ID\_Specialty = id\_spec

};

db.SecondSpec.Add(secondSpec);

db.SaveChanges();

ThirdSpec thirdSpec = new ThirdSpec

{

ID\_Specialty = id\_spec

};

db.ThirdSpec.Add(thirdSpec);

db.SaveChanges();

}

else

{

MessageBox.Show("Такая специальность уже есть", "Ошибка");

}

textBox4.Text = "";

PrintSpec();

}

else

{

MessageBox.Show("Введите название специальности", "Ошибка");

}

}

else

{

MessageBox.Show("Выберите факультет", "Ошибка");

}

}

else if (button3.Text == "Удалить")

{

if (textBox4.Text != "")

{

//AutRegApplicantsEntities db = new AutRegApplicantsEntities();

//Faculty faculty = null;

//faculty = db.Faculty.FirstOrDefault(c => c.FacultyName == textBox2.Text);

//if (faculty == null)

//{

// MessageBox.Show("Такого факультета нет", "Ошибка");

//}

//else

//{

// db.Faculty.Remove(faculty);

// db.SaveChanges();

//}

//textBox2.Text = "";

//PrintFaculty();

AutRegApplicantsEntities db = new AutRegApplicantsEntities();

Specialty specialty = null;

IEnumerable<Faculty> faculties = db.Faculty;

int id\_faculty = 0;

foreach (var i in faculties)

{

if (i.FacultyName == comboBox1.Text)

{

id\_faculty = i.ID\_Faculty;

}

}

specialty = db.Specialty.FirstOrDefault(c => c.SpecialtyName == textBox4.Text && c.ID\_Faculty == id\_faculty);

if(specialty==null)

{

MessageBox.Show("Такой специальности нет", "Ошибка");

comboBox1.Text = "";

textBox4.Text = "";

}

else

{

FirstSpec firstSpecs = db.FirstSpec.FirstOrDefault(c => c.ID\_Specialty == specialty.ID\_Specialty);

SecondSpec secondSpec = db.SecondSpec.FirstOrDefault(c => c.ID\_Specialty == specialty.ID\_Specialty);

ThirdSpec thirdSpec = db.ThirdSpec.FirstOrDefault(c => c.ID\_Specialty == specialty.ID\_Specialty);

db.FirstSpec.Remove(firstSpecs);

db.SaveChanges();

db.SecondSpec.Remove(secondSpec);

db.SaveChanges();

db.ThirdSpec.Remove(thirdSpec);

db.SaveChanges();

db.Specialty.Remove(specialty);

db.SaveChanges();

comboBox1.Text = "";

textBox4.Text = "";

PrintSpec();

}

}

else

{

MessageBox.Show("Выберите специальность для удаления", "Ошибка");

}

}

else if (button3.Text == "Изменить")

{

if (textBox4.Text != "")

{

AutRegApplicantsEntities db = new AutRegApplicantsEntities();

Specialty specialty = null;

int id = dataGridView3.CurrentCell.RowIndex;

IEnumerable<Faculty> faculties = db.Faculty;

int id\_faculty = 0;

foreach (var i in faculties)

{

if(i.FacultyName==comboBox1.Text)

{

id\_faculty = i.ID\_Faculty;

}

}

specialty = db.Specialty.FirstOrDefault(c => c.SpecialtyName == textBox4.Text && c.ID\_Faculty == id\_faculty);

if(specialty==null)

{

string spec = dataGridView3[0, id].Value.ToString();

Specialty specialty1 = db.Specialty.FirstOrDefault(c => c.SpecialtyName == spec);

specialty1.SpecialtyName = textBox4.Text.ToString();

foreach (var i in faculties)

{

if (i.FacultyName == comboBox1.Text)

id\_faculty = i.ID\_Faculty;

}

specialty1.ID\_Faculty = id\_faculty;

db.SaveChanges();

}

else

{

MessageBox.Show("Такая специальность уже есть", "Ошибка");

}

textBox4.Text = "";

PrintSpec();

}

else

{

MessageBox.Show("Выберите специальность", "Ошибка");

}

}

}

private void редактироватьToolStripMenuItem2\_Click(object sender, EventArgs e)

{

PrintSpec();

panel2.Location = new Point(5, 2035);

panel1.Location = new Point(5, 1035);

panel3.Location = new Point(5, 35);

panel4.Location = new Point(5, 4035);

button3.Text = "Изменить";

comboBox1.Items.Clear();

IEnumerable<Faculty> faculties = null;

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

faculties = db.Faculty;

foreach (var i in faculties)

{

comboBox1.Items.Add(i.FacultyName.ToString());

}

}

}

private void dataGridView3\_Click(object sender, EventArgs e)

{

if (button3.Text == "Изменить"|| button3.Text == "Удалить")

{

int id = dataGridView3.CurrentCell.RowIndex;

comboBox1.Text = dataGridView3[1, id].Value.ToString();

textBox4.Text = dataGridView3[0, id].Value.ToString();

}

}

private void удалитьToolStripMenuItem1\_Click(object sender, EventArgs e)

{

button3.Text = "Удалить";

PrintSpec();

panel2.Location = new Point(5, 2035);

panel1.Location = new Point(5, 1035);

panel3.Location = new Point(5, 35);

panel4.Location = new Point(5, 4035);

comboBox1.Items.Clear();

}

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

{

panel2.Location = new Point(5, 2035);

panel1.Location = new Point(5, 1035);

panel3.Location = new Point(5, 3035);

panel4.Location = new Point(5, 35);

using (AutRegApplicantsEntities db = new AutRegApplicantsEntities())

{

IEnumerable<Request> req = db.Request;

dataGridView4.ColumnCount = 8;

dataGridView4.RowCount = 1;

dataGridView4.RowCount = dataGridView4.RowCount + 1;

int k = 0;

foreach (DataGridViewColumn row in dataGridView4.Columns)

{

if (k == 0)

{

row.HeaderCell.Value = "Логин";

}

if (k == 1)

{

row.HeaderCell.Value = "Фамилия";

}

if (k == 2)

{

row.HeaderCell.Value = "Имя";

}

if (k == 3)

{

row.HeaderCell.Value = "Окончил";

}

if (k == 4)

{

row.HeaderCell.Value = "Сумма баллов";

}

if (k == 5)

{

row.HeaderCell.Value = "1-ая специальность";

}

if (k == 6)

{

row.HeaderCell.Value = "2-ая специальность";

}

if (k == 7)

{

row.HeaderCell.Value = "3-ая специальность";

}

k++;

}

k = 0;

foreach (var i in req)

{

IEnumerable<Users> users = db.Users;

string login = null;

foreach (var j in users)

{

if (j.ID\_User == i.ID\_User)

login = j.Login;

}

dataGridView4[0, k].Value = login;

dataGridView4[1, k].Value = i.FName.ToString();

dataGridView4[2, k].Value = i.LName.ToString();

dataGridView4[3, k].Value = i.Education.ToString();

dataGridView4[4, k].Value = (i.Сertificate + i.CT1 + i.CT2 + i.CT3).ToString();

IEnumerable<Specialty> specialties = db.Specialty;

IEnumerable<FirstSpec> firstSpecs = db.FirstSpec;

IEnumerable<SecondSpec> secondSpecs = db.SecondSpec;

IEnumerable<ThirdSpec> thirdSpecs = db.ThirdSpec;

string spec1 = null;

string spec2 = null;

string spec3 = null;

foreach (var sp1 in firstSpecs)

{

if(i.ID\_FSpec==sp1.ID\_FSpec)

{

foreach(var sp2 in specialties)

{

if(sp1.ID\_Specialty==sp2.ID\_Specialty)

{

spec1 = sp2.SpecialtyName.ToString();

}

}

}

}

dataGridView4[5, k].Value = spec1;

foreach (var sp1 in secondSpecs)

{

if (i.ID\_SSpec == sp1.ID\_SSpec)

{

foreach (var sp2 in specialties)

{

if (sp1.ID\_Specialty == sp2.ID\_Specialty)

{

spec2 = sp2.SpecialtyName.ToString();

}

}

}

}

dataGridView4[6, k].Value = spec2;

foreach (var sp1 in thirdSpecs)

{

if (i.ID\_TSpec == sp1.ID\_TSpec)

{

foreach (var sp2 in specialties)

{

if (sp1.ID\_Specialty == sp2.ID\_Specialty)

{

spec3 = sp2.SpecialtyName.ToString();

}

}

}

}

dataGridView4[7, k].Value = spec3;

dataGridView4.RowCount = dataGridView4.RowCount + 1;

k++;

}

dataGridView4.RowCount = dataGridView4.RowCount - 2;

}

}

}

}