**ВВЕДЕНИЕ**

Сложно переоценить значение АРМ в современном мире, ведь именно благодаря им люди получили превосходные условия управления данными БД. Именно в АРМ можно управлять данными любой БД, координировать и сортировать данные.

Тема данного проекта ­­­– АРМ «ВУЗ». Данный АРМ предназначена для управления работой ВУЗа.

Цели и задачи проекта практики:

* закрепление теоретических знаний и практических умений по учебным дисциплинам «Конструирование программ и языки программирования», «Математическое моделирование», «Системы управления базами данных» изученные в процессе теоретического обучения;
* закрепление, расширение и углубление теоретических и практических знаний и умений, полученных в процессе обучения;
* закрепление знаний и умений, связанных с технологией обработки информации;
* закрепление и углубление знаний и умений, полученных при изучении цикла специальных учебных дисциплин;
* закрепление практических навыков по составлению алгоритмов и реализации их на персональном компьютере;
* приобретение навыков оперативного управления технологическим процессом обработки информации.

Основными вопросами программы являлись:

1. Изучение алгоритмов и написание программ на языке C#.
2. Работа с массивами, строками на языке C#.
3. Работа с вычислительной геометрией и численными алгоритмами.
4. Изучение классов и их объектов.
5. Работа с перечислениями, структурами и интерфейсами.
6. Изучение и работа с многопоточными приложениями.
7. Работа с файлами и каталогами, коллекциями и классами-прототипами.
8. Изучение пользовательского интерфейса.
9. Разработка Windows-приложений.
10. Создание и подключение баз данных в среде программирования Visual Studio 2019.

Разделы пояснительной записки:

1. Раздел «Постановка задачи» содержит формулировку задания с указанием всех функций проекта, определение круга задач, которые должны быть автоматизированы или модифицированы, описание входных и выходных данных в проекте.
2. Раздел «Вычислительная система» содержит описание языка программирования, среды разработки и операционной системы, выбранных для реализации проекта, их достоинств и недостатков, минимальные требования к аппаратным и операционным ресурсам.
3. Раздел «Проектирование» содержит описание внешнего пользовательского интерфейса, описание физической модели базы данных, алгоритм работы сайта.
4. Раздел «Описание программы» содержит описание всех функций, модулей, классов, входящих в приложение, интерфейса программы.
5. Раздел «Отладка и испытание программы» содержит тестовые примеры проверки работы программы в нормальных и экстремальных условиях, описание реакции программы на тесты, анализ ожидаемых и полученных результатов.

«Заключение» содержит краткую формулировку проблемы и пути ее решения, использованные методы и средства, найденные нетрадиционные способы разработки, возможность модификации.

**1 ПОСТАНОКА ЗАДАЧИ**

* 1. **Формулировка задания**

Необходимо разработать приложение для предметной области «ВУЗ». В приложении необходимо реализовать следующие функции:

* 1. Вывод таблиц.
* Студенты;
* Группы;
* Предметы;
* Специальности;
* Преподаватели;
* Мероприятия;
* Абитуриенты;
* Пересдачи.
  1. Вывод запросов.
  2. Модификация таблицы.
* добавление в базу данных;
* изменение в базе данных;
* удаление в базе данных.

Необходимо реализовать специальные выборки по таблицам, такие как вывод всех студентов со средним баллом ниже заданного, преподавателей только высшей категории.

* 1. **Обоснование необходимости автоматизации**

Предметная область – это часть реального мира, подлежащая изучению с целью создания базы данных для автоматизации процесса управления. Наборы принципов, которые определяют организацию логической структуры хранения данных в базе, называются моделями данных.

В качестве предметной области данного курсового проекта рассматривается база данных "ВУЗ".

Аналоги такой предметной области обычно используются локально в качестве АРМ для предприятий, поэтому сложно выделить нескольлко явных аналогов данного приложения, такими аналогами могут быть например:

1. АРМ Aviasales, которая предназначена для хранения и модификации базы данных. Aviasales – это компания по удобной продаже авиа-билетов.
2. АРМ тренажерного зала, в которой может храниться информация о клиентах, абониментах и учете оборудования.
3. АРМ интернет-магазина, в которой может храниться и модифицироваться информация о покупателях и товарах, реализована корзина и оформляется заказ.
   1. **Определение данных и их представление**

Входные данные – величины, которые задаются до начала работы алгоритма или определяются динамически во время работы.

Входные данные программы:

1. Данные, вводимые пользователем для добавления в БД;
2. Данные, вводимые пользователем для изменения значений БД;
3. Данные, вводимые пользователем для удаления данных из БД;
4. Данные, вводимые пользователем для фильтрации значений в БД.

Выходные данные – это те данные, которые получаются в результате выполнения запроса на поиск.

Выходные данные программы:

1. Результат добавления записи;
2. Результат фильтрации.
3. **Вычислительная система**
   1. **Обоснование выбора языка программирования**

C# – объектно-ориентированный язык программирования. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML. C# разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR.

* 1. **Обоснование выбора среды разработки**

Выбор среды разработки исходил из желания найти альтернативу Microsoft Visual Studio. Выбор пал на среду разработки Rider от компании JetBrains.

1) Первая причина выбора Rider: в отличии от Visual Studio, Rider не зацикливается на 32-битных процессах. Даже если у Rider есть процессы, доступные только для back-end, например, SWEA (Solution-Wide Analysis), создание кода будет проходить гладко без каких-либо пауз или сбоев. И как отмечает большинство пользователей работавших с Visual Studio и Rider, последний работает гораздо стабильней и быстрее чем Visual Studio;

2) JetBrains Rider является кроссплатформенным, он может работать на платформах Windows, Mac или Linux с одинаковой функциональностью и стабильностью. Visual Studio работает преимущественно на платформе Windows. И если есть необходимость перейти на Linux или Mac, то необходимо будет приобретать дополнительные решения: Visual Studio Code (для Linux) и Visual Studio для Mac.  Главным недостатком является то, что версии Visual Studio для Mac и Linux имеют разный функционал и внешний вид, к которому придётся привыкать. Rider, как внешне, так и по своим функциям, одинаков на всех платформах, поэтому, если пользователь решит переключиться с Windows на Mac или Linux, он получит уже привычную среду разработки и не будет тратить драгоценное время на обучение;

3) Среда Rider включает большинство функций, популярного расширения Visual Studio для разработчиков .NET – ReSharper. В составе Rider есть внушительный набор для рефакторинга, проверки кода и контекстных действий для всех поддерживаемых им языков и технологий. У Visual Studio также есть наборы рефакторингов и проверки ошибок кода, но гораздо более ограниченный, чем те, что предоставлены в Rider и ReSharper;

4) У Rider есть огромное количество функций, унаследованных от платформы IntelliJ:

* Поддержка систем контроля версий: помимо Git и Mercurial, Rider работает с CVS и Subversion. Интеграция VSTS доступна через специальный плагин, поддерживаемый компанией Microsoft;
* Rider (с помошью DataGrip) поддерживает подключение к базам данных и SQL. Пользователям Visual Studio в большинстве случаев нужно будет использовать ODBC;
* Поддержка возможностей для front-end разработки с использованием JavaScript, TypeScript, CSS, HTML, LESS, Sass и т. д. Доступна в Rider благодаря тому, что продукт включает в своем составе, функции специализированного ПО для веб-разработки – JetBrains WebStorm;
* В среде разработки Rider также присутствует возможность интеграции со многими трекерами проблем, такими как Team Foundation Server и Visual Studio Team Services. Также он поддерживает JIRA Software, YouTrack и другие решения, а также большое количество высококачественных специализированных плагинов, разработанных для IntelliJ и ReSharper, большинство их которых бесплатные. Visual Studio тоже поддерживает различные плагины, но бесплатными из них являются еденицы;

5) Решения и проекты, с которыми работает JetBrains Rider, полностью совместимы с Visual Studio, и не используют проприетарные форматы.

Недостаток Rider: решения с использованием Windows Forms работают чаще всего некорректно.

* 1. **Требования к конфигурации программного и аппаратного обеспечения**

Программа будет реализована в операционной системе Mac OS Catalina

Основа системы — POSIX-совместимая операционная система Darwin, являющаяся свободным программным обеспечением. Её ядром является XNU, в котором используется микроядро Mach и стандартные службы BSD. Все возможности Unix в macOS доступны через консоль.

Поверх этой основы в Apple разработано много проприетарных компонентов, таких как Cocoa и Carbon, Quartz.

macOS отличается высокой устойчивостью.

В macOS (как и в любой UNIX-системе) используется вытесняющая многозадачность и защита памяти, позволяющие запускать несколько изолированных друг от друга процессов, каждый из которых не может прервать или модифицировать все остальные. На архитектуру macOS повлияла OpenStep, которая была задумана как переносимая операционная система (например, NeXTSTEP была перенесена с оригинальной платформы 68k компьютера NeXT до приобретения NeXTSTEP компанией Apple). Аналогичным образом OpenStep была перенесена на PowerPC в рамках проекта Rhapsody.

Графический интерфейс в macOS получил название Aqua. Использование закруглённых углов, полупрозрачных элементов и светлых полосок также повлияло на внешний вид первых моделей iMac. После выхода первой версии Mac OS X другие разработчики тоже стали использовать интерфейс Aqua.

Основами macOS являются:

* подсистема с открытым кодом — Darwin (ядро Mach и набор утилит BSD);
* среда программирования Core Foundation (Carbon API, Cocoa API и Java API);
* графическая среда Aqua (QuickTime, Quartz Extreme и OpenGL);
* технологии Core Image, Core Animation, CoreAudio и CoreData.

Минимальные требования к аппаратным и операционным ресурсам для разработки:

* процессор: 4-ядерный Intel Core i7 2,4 GHz;
* оперативная память: 8 GB 1600 MHz DDR3;
* видеокарта: NVIDIA GeForce GT 650M 1 GB, Intel HD Graphics 4000 1536 MB;
* операционная система: Mac OS Catalina 10.15.4.

1. **Проектирование**
   1. **Проектирование интерфейса**

Графический интерфейс пользователя – система средств для взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т. п.).

Интерфейс будет выполнен в темно-синих и голубых тонах. Выбор такой цветовой гаммы обусловлен темой приложения и стремлению к минимализму разрабатываемого ПО.

На вкладке «Хранимый» выводятся 4 основные таблицы: Студенты, Группы, Предметы и Специальности. На данной вкладке расположены такие элементы, как таблицы, панели управления таблицами и подписи для каждой из таблиц.

На вкладке «Преподаватель» находятся такие элементы, как таблицы для вывода всех преподавателей, поле списка для фильтрации значений, кнопки, которые выполняют фильтрацию и подписи к этим кнопкам. Также расположены поля ввода для каждого поля таблицы и кнопки модификации таблицы, такие как «Добавить», «Изменить», «Обновить» и «Удалить» которые взаимодействуют с каждым из полей ввода.

На вкладке «Преподаватель» находятся такие элементы, как таблица для вывода всех преподавателей, поле списка для фильтрации значений, кнопки, которые выполняют фильтрацию и подписи к этим кнопкам. Также расположены поля ввода для каждого поля таблицы и кнопки модификации таблицы, такие как «Добавить», «Изменить», «Обновить» и «Удалить» которые взаимодействуют с каждым из полей ввода.

На вкладке «Мероприятия» находятся такие элементы, как таблица для вывода всех мероприятий, два поля списка для фильтрации значений, кнопки, которые выполняют группировку значений. Также расположены поля ввода для каждого поля таблицы и кнопки модификации таблицы, такие как «Добавить», «Изменить», «Обновить» и «Удалить» которые взаимодействуют с каждым из полей ввода.

На вкладке «Абитуриент» находятся такие элементы, как таблица для вывода всех абитуриентов, поле списка для фильтрации значений, кнопки, которые выполняют фильтрацию, поле ввода и подписи к этим кнопкам. Также расположены поля ввода для каждого поля таблицы и кнопки модификации таблицы, такие как «Добавить», «Изменить», «Обновить» и «Удалить» которые взаимодействуют с каждым из полей ввода.

На вкладке «Пересдачи» находятся такие элементы, как таблица для вывода всех пересдач, два поля списка для фильтрации значений, кнопки, которые выполняют группировку значений. Также расположены поля ввода для каждого поля таблицы и кнопки модификации таблицы, такие как «Добавить», «Изменить», «Обновить» и «Удалить» которые взаимодействуют с каждым из полей ввода.

* 1. **Описание алгоритма**

Алгоритм добавления записи в базу данных, состоит в подключении БД и создании транзакции. Затем выполняется запрос на добавление записи и поля таблицы связываются с полями ввода в приложении, затем происходит откат транзакции и обновление таблицы.

Алгоритм обновления таблицы в базе данных, состоит в подключении БД и создании транзакции. Затем выполняется запрос на получение всех записей, затем происходит откат транзакции и создание списка из полученных записей, после чего они выводятся в компонент таблицы.

Алгоритм фильтрации записей в базе данных, состоит в подключении БД и создании транзакции. Затем выполняется запрос на чтение записи и поля таблицы связываются с полями ввода в приложении. Выполняется запрос на фильтрацию по признаку, затем происходит откат транзакции и вывод отфильтрованных значений в компонент списка.

* 1. **Логическая модель данных**

В процессе разработки ПС было выделено 8 сущностей. Информация о них была помещена в соответствующие таблицы. Информация о студентах хранится в таблице 3.1.

Таблица 3.1 – Студенты

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **1** | **2** | **3** |
| id | int | Уникальный идентификатор студента (первичный ключ) |
| name | string | Имя студента |
| group | int | Id группы студента (внешний ключ) |

Информация о группах хранится в таблице 3.2.

Таблица 3.2 – Группы

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **1** | **2** | **3** |
| id | int | Уникальный идентификатор группы (первичный ключ) |
| name | string | Номер группы |
| course | int | Курс |
| speciality | int | Id специальности (внешний ключ) |

Информация о предметах хранится в таблице 3.3.

Таблица 3.3 – Предметы

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **1** | **2** | **3** |
| id | int | Уникальный идентификатор предмета (первичный ключ) |
| name | string | Название предмета |
| course | int | Курс |
| speciality | int | Id специальности (внешний ключ) |

Информация о специальностях хранится в таблице 3.4.

Таблица 3.4 – Специальности

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **1** | **2** | **3** |
| id | int | Уникальный идентификатор специальности (первичный ключ) |
| name | string | Название специальности |
| course | int | Курс |
| entry mark | int | Проходной балл |

Информация о преподавателях хранится в таблице 3.5.

Таблица 3.5 – Преподаватели

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **1** | **2** | **3** |
| id | int | Уникальный идентификатор преподавателя (первичный ключ) |
| name | string | Имя |
| phone number | string | Номер телефона |
| category | string | Категория |
| birthday | date | Дата рождения |

Информация о мероприятиях хранится в таблице 3.6.

Таблица 3.6 – Мероприятия

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **1** | **2** | **3** |
| id | int | Уникальный идентификатор мероприятия (первичный ключ) |
| name | string | Название |
| responsible group | int | Id ответственной группы |
| responsible professor | int | Id ответственного преподавателя |

Информация об абитуриентах хранится в таблице 3.7.

Таблица 3.7 – Абитуриенты

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **1** | **2** | **3** |
| id | int | Уникальный идентификатор абитуриента (первичный ключ) |
| name | string | Имя |
| avg mark | double | Средний балл |
| birthday | date | Дата рождения |

Информация о пересдачах хранится в таблице 3.8.

Таблица 3.8 – Пересдачи

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Описание** |
| **1** | **2** | **3** |
| id | int | Уникальный идентификатор пересдачи (первичный ключ) |
| professor id | int | Id профессора, принимавшего (внешний ключ) |
| student id | int | Id студента, который сдавал (внешний ключ) |
| subject | int | Id предмета (внешний ключ) |
| mark | int | Полученная отметка |

**3.4 Физическая модель базы данных**

В ходе разработки и проектирования базы данных, выявлены сущности и связи, представленные на рисунке 3.1.

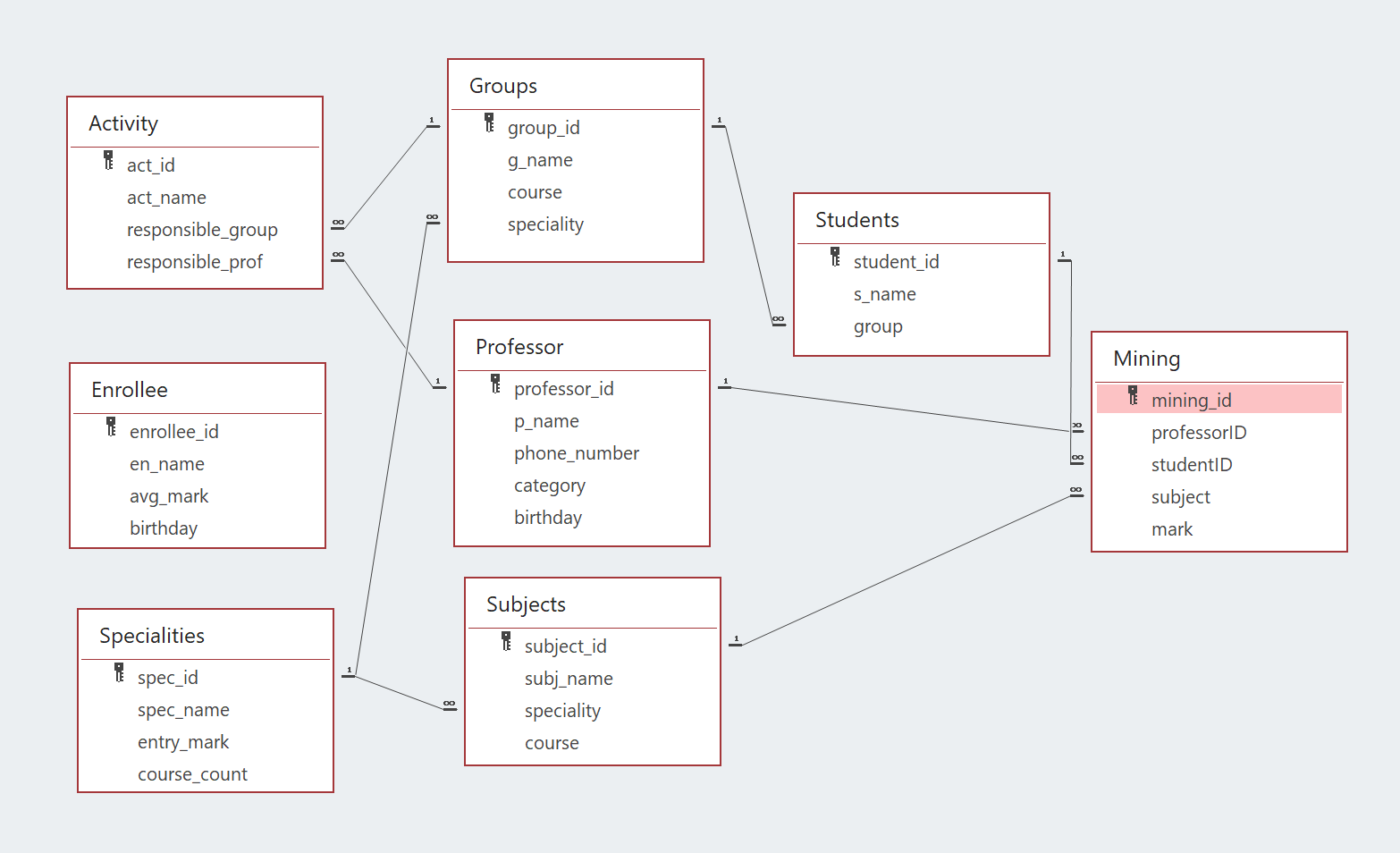


Рисунок 3.1 – Схема данных

1. **ОПИСАНИЕ ПРОГРАММЫ**
   1. **Логическая структура**

В ходе выполнения поставленных задач был реализован следующий функционал:

private void updateStudents(int groudId = 0) – получение списка пользователей

private void CreateConnection(IDbConnection dbConnection, string connectString, Action<IDbConnection> action) – создание подключения к базе данных

private void updateSpeciality() – получение списка специальностей

private void updateGroups() – получение списка групп

private void updateTeachers() – получение списка преподавателей

private void updateDisciplines() – получение списка дисциплин

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

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

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

private void button4\_Click(object sender, EventArgs e) – сброс фильтра для студентов

private void button8\_Click(object sender, EventArgs e) – добавление специальности

private void button7\_Click(object sender, EventArgs e) – удаление специальности

private void button6\_Click(object sender, EventArgs e) – добавление группы

private void button5\_Click(object sender, EventArgs e) – удаление группы

private void button10\_Click(object sender, EventArgs e) – добавление преподавателя

private void button9\_Click(object sender, EventArgs e) – удаление преподавателя

private void button11\_Click(object sender, EventArgs e) – вывод расписания для определенной группы

* 1. **Физическая структура**

В ходе выполнения проекта были реализованы следующие классы:

public class Form1 – главная форма программы.

public class Group – модель группы.

public class Disciplines – модель дисциплины.

public class Schedule – модель расписания.

public class Speciality – модель специальности.

public class Student – модель студента.

public class Teacher – является моделью продукта, каждое свойство которой представляет соответствующее поле базы данных.

* 1. **Особенности интерфейса**

Интерфейс вкладки «Студенты» представлен на рисунке 4.1:

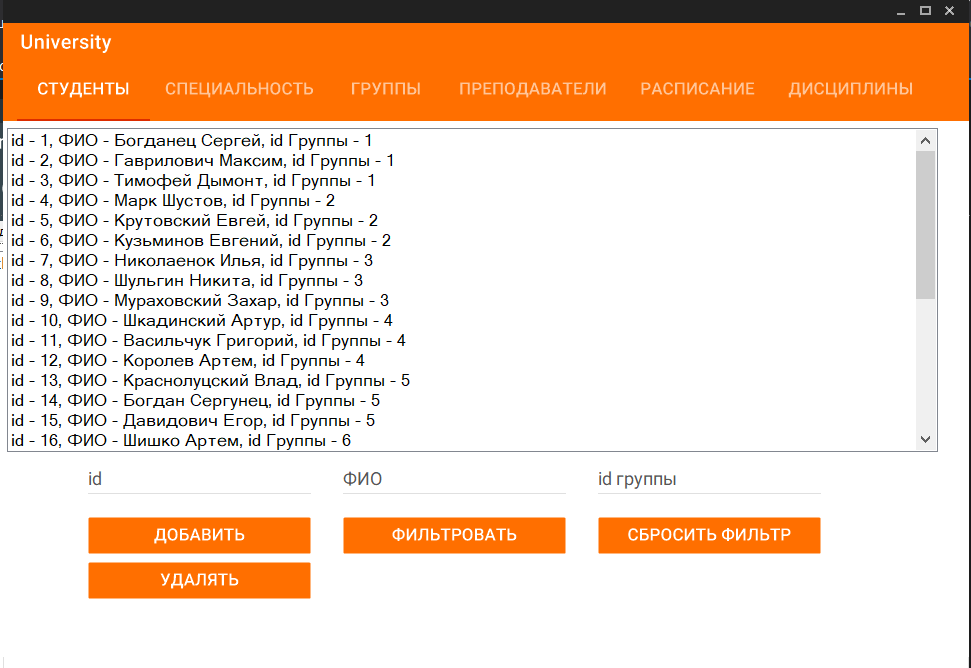


Рисунок 4.1 – Интерфейс вкладки «Студенты»

Интерфейс вкладки «Специальность» представлен на рисунке 4.2:

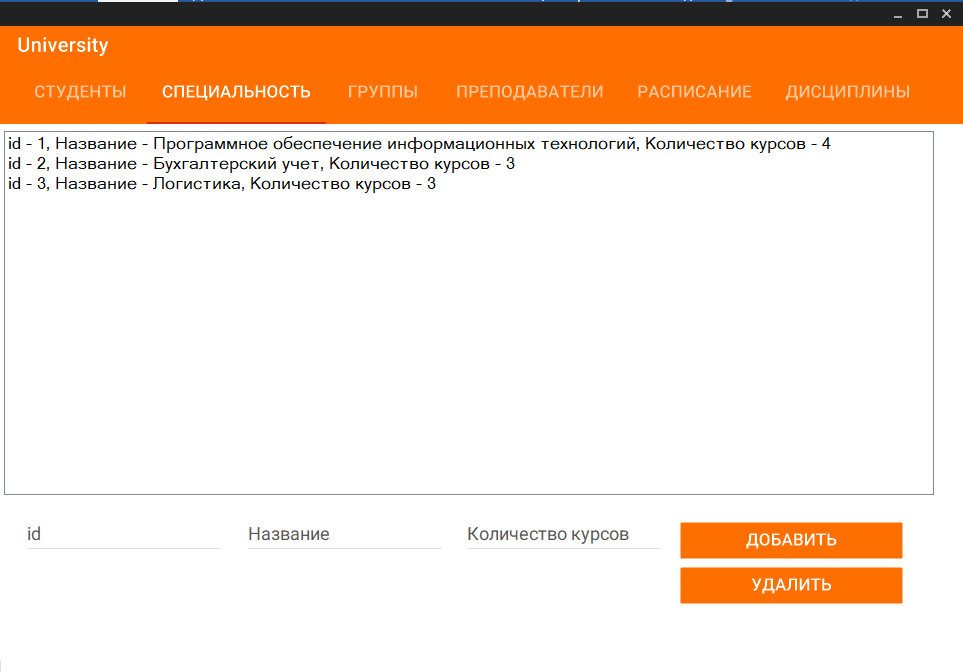


Рисунок 4.2 – Интерфейс вкладки «Специальность»

Интерфейс вкладки «Группы» представлен на рисунке 4.3:

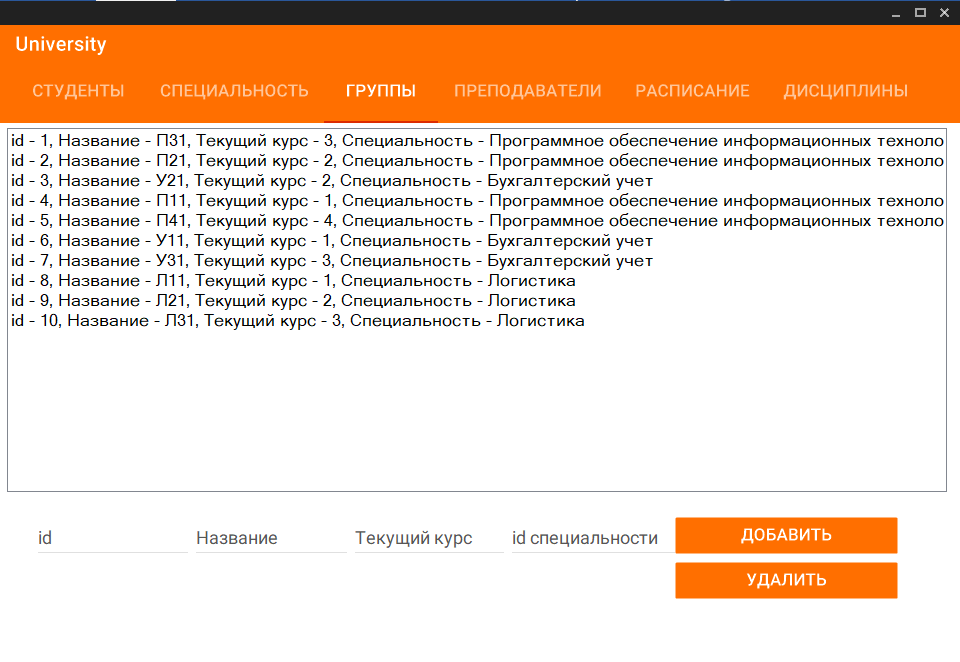


Рисунок 4.3 – Интерфейс вкладки «Группы»

Интерфейс вкладки «Преподаватели» представлен на рисунке 4.4:

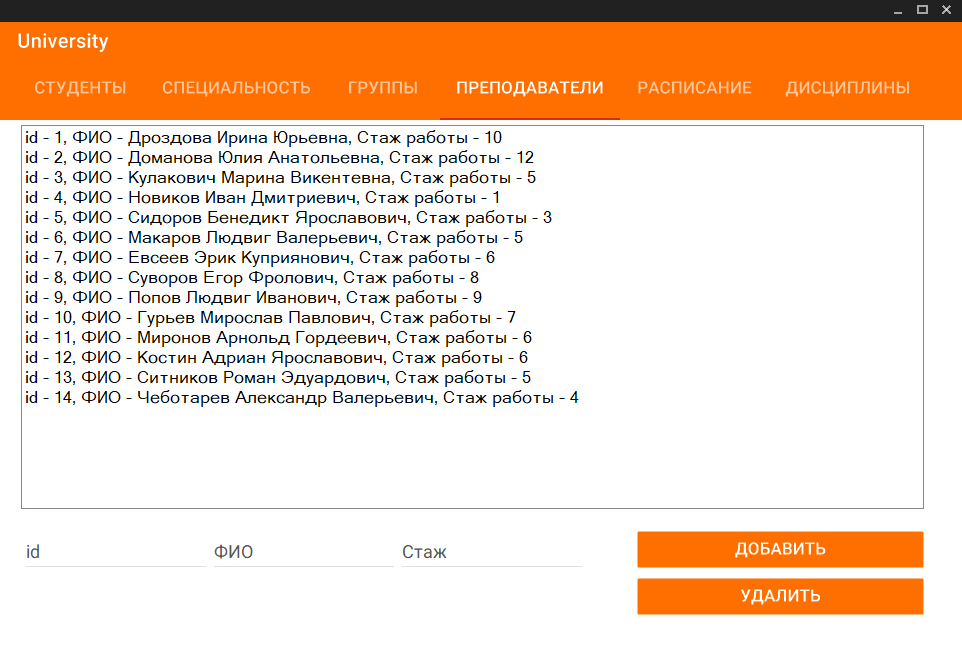


Рисунок 4.4 – Интерфейс вкладки «Преподаватели»

Интерфейс вкладки «Расписание» представлен на рисунке 4.5:

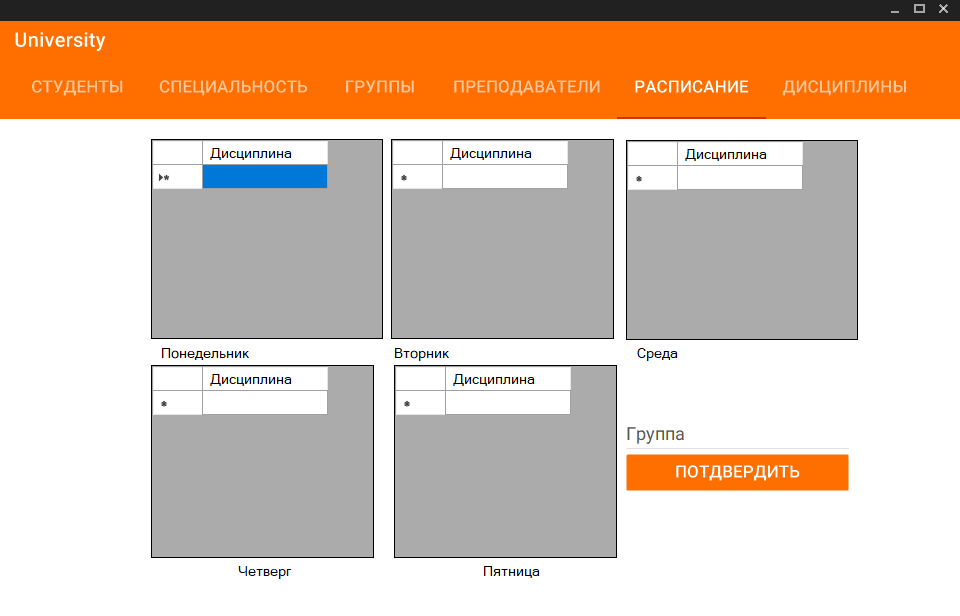


Рисунок 4.5 – Интерфейс вкладки «Расписание»

Интерфейс вкладки «Дисциплины» представлен на рисунке 4.6:

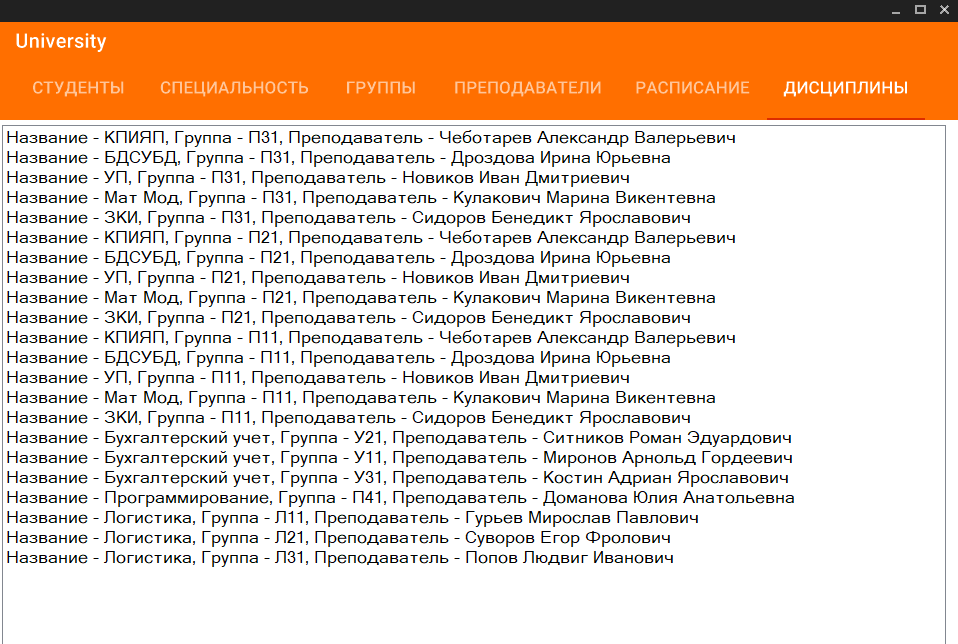


Рисунок 4.6 – Интерфейс вкладки «Дисциплины»

1. **ОТЛАДКА И ИСПЫТАНИЕ ПРОГРАММЫ** 
   1. **Тестовые примеры**

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом, а также проверка правильности работы программы в целом и её составных частей. (ISO/IEC TR 19759:2005). [1, с. 204-205]

Тестирование данного программного продукта будет проводится нефункциональным тестированием (UI тесты):

Тест 1: Протестировать интерфейс формы Form1 на переход на вкладку «Студенты».

Тест 2: Протестировать интерфейс формы Form1 на переход на вкладку «Специальность».

Тест 3: Протестировать интерфейс формы Form1 на переход на вкладку «Группы».

Тест 4: Протестировать интерфейс формы Form1 на переход на вкладку «Преподаватели».

Тест 5: Протестировать интерфейс формы Form1 на переход на вкладку «Расписание».

Тест 6: Протестировать интерфейс формы Form1 на переход на вкладку «Дисциплины».

* 1. **Анализ полученных результатов**

В результате проведения тестирования были получены следующие результаты:

Тест 1: После перехода на вкладку «Студенты» открывается вкладка с соответствующими компонентами. Результаты теста представлены на   
рисунке 5.1:

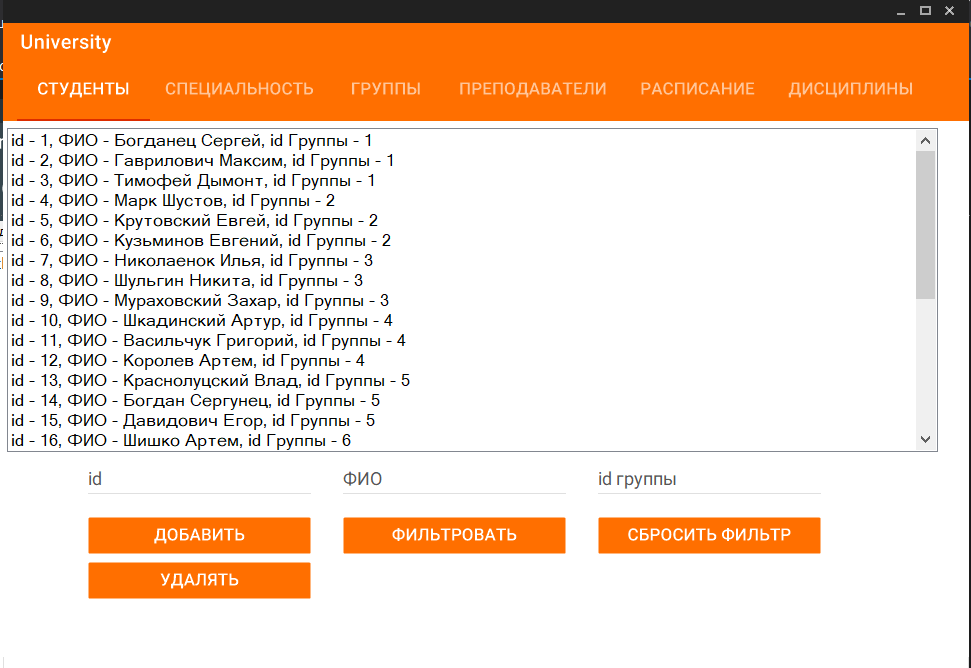


Рисунок 5.1 – Результат открытия вкладки «Студенты»

Тест 2: После перехода на вкладку «Специальность» открывается вкладка с соответствующими компонентами. Результаты теста представлены на рисунке 5.2:

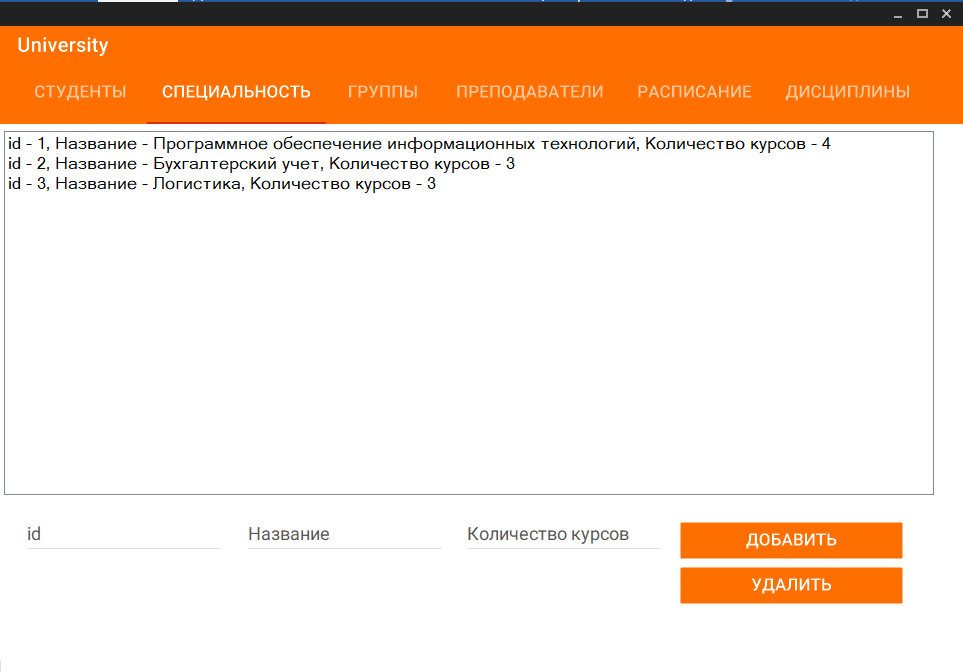


Рисунок 5.2 – Результат открытия вкладки «Специальность»

Тест 3: После перехода на вкладку «Группы» открывается вкладка с соответствующими компонентами. Результаты теста представлены на   
рисунке 5.3:

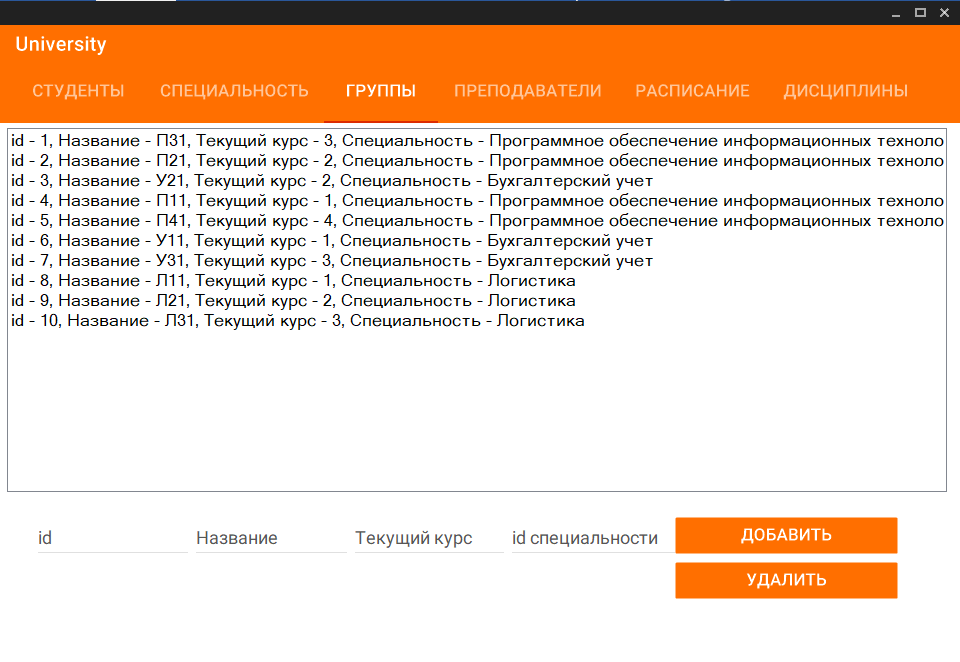


Рисунок 5.3 – Результат открытия вкладки «Группы»

Тест 4: После перехода на вкладку «Преподаватели» открывается вкладка с соответствующими компонентами. Результаты теста представлены на рисунке 5.4:

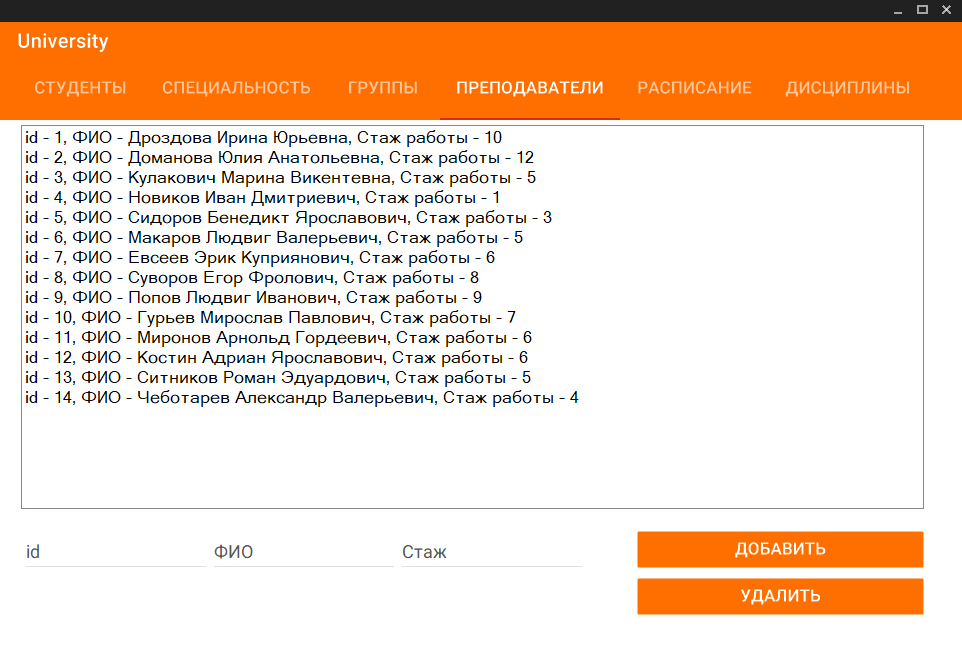


Рисунок 5.4 – Результат открытия вкладки «Преподаватели»

Тест 5: После перехода на вкладку «Расписание» открывается вкладка с соответствующими компонентами. Результаты теста представлены на   
рисунке 5.5:

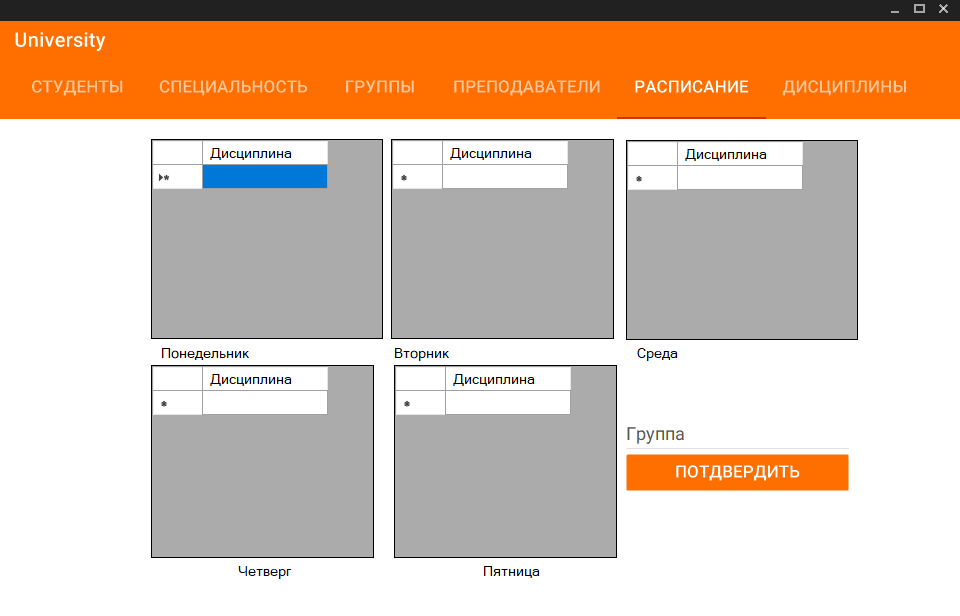


Рисунок 5.5 – Результат открытия вкладки «Расписание»

Тест 6: После перехода на вкладку «Дисциплины» открывается вкладка с соответствующими компонентами. Результаты теста представлены на рисунке 5.6:

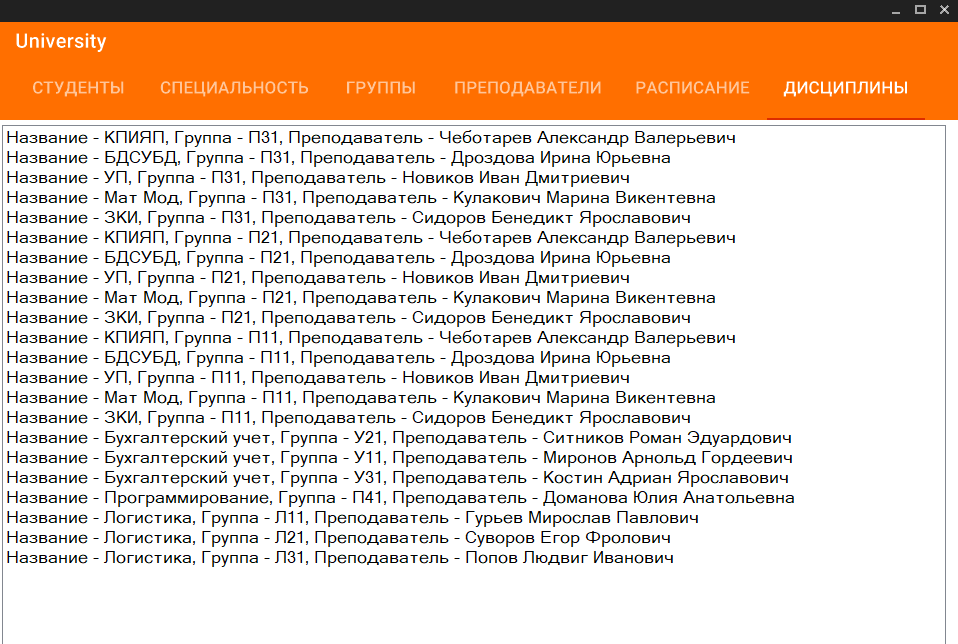


Рисунок 5.6 – Результат открытия вкладки «Дисциплины»

**ЗАКЛЮЧЕНИЕ**

В ходе разработки проекта был использован язык программирования C# и среда разработки Visual Studio 2019.

В проекте были реализованы функции вывода таблиц БД, их обновления, запросы к таблицам БД, а также панели управления соответствующих таблиц с функциями добавления, удаления и изменения данных в таблицах.

В результате прохождения практики, я научился:

* планировать основные этапы технологического процесса обработки информации;
* строить и разрабатывать алгоритмы, графы на языке С#;
* разрабатывать программные средства с использованием объектно-ориентированного программирования на языке С#;
* работать с библиотеками классов Microsoft NET Framework;
* разрабатывать программные средства с использованием Windows-приложений средствами библиотеки классов Microsoft NET Framework;
* создавать и подключать базу данных на языке C# к разрабатываемому проекту.

По завершению учебной практики были подведены итоги и обсуждены основные критические моменты за время прохождения учебной практики по программированию.

**Список использованных источников**

1. Ошероув Рой. Искусство автономного тестирования. – «ДМК», 2013 г. – с. 360.
2. Гращенко Л. А. [Обобщенная модель угроз информационной безопасности визуальных интерфейсов пользователя](https://web.archive.org/web/20140407082709/http:/www.diktan.ru/tw_files2/urls_15/38/d-37250/7z-docs/2.pdf#page=36) Орловского государственного технического университета. Серия: Информационные системы и технологии. 2006 г. – с. 41– 45.
3. Либерти Д. Язык программирования C# Программирование на C#. – Санкт-Петербург.  2003 г. – с. – 688 .
4. Ходин А.В., Шуклин Д.А., Шалобаев Е.В. Особенности UX-дизайна интернет-проектов Альманах научных работ молодых ученых университета ИТМО: Статья в сборнике трудов конференции 2016 г. – с. 85 – 90.
5. Документация ASP.NET [Электронный ресурс] Режим доступа: <https://api.github.com/repos/aspnet/AspNetCore> Дата доступа: 02.04.2020.
6. Документация Apache [Электронный ресурс] Режим доступа: <https://www.apachefriends.org/ru/about.html> Дата доступа: 01.06.2020.
7. Документация JetBrains [Электронный ресурс] Режим доступа: https://www.jetbrains.com/ru-ru/rider/ Дата доступа: 01.04.2020.
8. Документация MariaDB [Электронный ресурс] Режим доступа: <https://ru.wikipedia.org/wiki/MariaDB> Дата доступа: 01.04.2020.