МИНИСТЕРСТВО ОБРАЗОВАНИЯ ОРЕНБУРГСКОЙ ОБЛАСТИ
ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ «ОРЕНБУРГСКИЙ КОЛЛЕДЖ ЭКОНОМИКИ И ИНФОРМАТИКИ»
(ГАПОУ «ОКЭИ»)

КУРСОВОЙ ПРОЕКТ

ОКЭИ 09.02.07 9024 9 КП

КП 01.01 Курсовой проект
По модулю МДК 01.01 Разработка программных модулей.

Количество листов: *24*
Дата готовности: *29.01.2024*
Разработал: *Мартемьянов Д.А.*
Руководитель: *Егурнова Е.Н.*

Оренбург 2024

Аннотация

Цель данной курсовой работы - разработать систему для ведения штата сотрудников в организации. Это включает в себя добавление, удаление и редактирование данных о сотрудниках, отделах, перевод в другой отдел и смену должности. Для реализации данной функциональности были использованы технологии .net и mssql.

Была создана база данных, которая хранит информацию о сотрудниках, такую как ФИО, должность, отдел .

Основными функциями системы являются:

- Добавление новых сотрудников в базу данных;
- Удаление сотрудников из базы данных ;
- Редактирование информации о сотрудниках ;
- Перевод сотрудников из одного отдела в другой ;
- Смена должности сотрудников .

Пользователи системы могут просматривать список всех сотрудников и отделов, а также искать конкретных сотрудников по ФИО.

Так же были проведён тесты с разными типами данных и разным количеством данных.

В результате была создана система, которая позволяет эффективно вести учёт и управление штатом сотрудников в организации. Она облегчает работу HR-отдела и позволяет быстро находить нужную информацию о сотрудниках.

Задание

Необходимо создать приложение для учёта сотрудников.

Оглавление

Содержание

Введение

Актуальность выполнения курсового проекта заключается в закреплении и усовершенствовании навыков по профессии программиста. Курсовой проект позволяет студенту реализовать полученные в учебном заведении теоретические знания в практической работе.

Целью данного курсового проекта является закрепление знаний, которые были получены при изучении профильного предмета «МДК 01.01 Разработка программных модулей», в рамках которого было изучено проектирование информационных систем для различных предметных областей

Курсовой проект также позволяет на территории учебного заведения попробовать свои силы в ситуациях, которые помогут в дальнейшей адаптации на местах будущей работы, раскрывающих характер предстоящей трудовой деятельности.

Выделены следующие задачи:

-разработка базы данных для хранения информации о сотрудниках и отделах;

-создание пользовательского интерфейса, который позволяет пользователям легко работать с базой данных и выполнять необходимые операции;

-реализация функций добавления, удаления, редактирования, перевода и смены должности сотрудников;

-реализация функций поиска сотрудников по ФИО;

-проведение оптимизации и подготовки системы к различным ситуациям;

-формирование отчета.

Объект исследования: ГАПОУ «ОКЭИ», по адресу: г. Оренбург, ул. Чкалова 11.

Предмет исследования: проектирование приложения «Ведение штата сотрудников» предприятия.

За время выполнения курсового проекта студенту предоставляется возможность спроектировать систему по своей предметной области и реализовать ее функционал.

Выполнение курсового проекта позволяет улучшить свои навыки в области проектирования различных моделей работы системы, создания конфигураций, а также программирования.

В процессе выполнения проекта студенты также могут научиться работать в команде, общаться с коллегами и руководством, а также учиться решать конфликтные ситуации. Это очень важные навыки для будущего профессионального роста.

1 Анализ предметной области

Предметной областью данной курсовой работы является управление персоналом в организации. Данная сфера человеческой деятельности является важной и неотъемлемой частью любой организации, независимо от ее размера и направления деятельности. В предметной области выделяются следующие сущности:

Сотрудники - основные элементы управления персоналом. Каждый сотрудник имеет свой уникальный идентификатор, ФИО, должность, отдел, контактную информацию и другие характеристики.

Отделы - структурные подразделения организации, которые объединяют сотрудников по определенным принципам. Каждый отдел имеет свой уникальный идентификатор, название, руководителя и другие характеристики.

Должности - определенные должностные обязанности, которые выполняют сотрудники в организации. Каждая должность имеет свое название, описание, уровень зарплаты и другие характеристики.

Процессы управления персоналом - это комплекс различных процедур и действий, направленных на эффективное управление персоналом организации. К ним относятся процессы найма, обучения, оценки качества работы, повышения квалификации и другие.

Отношения между сотрудниками и руководством - взаимодействие между сотрудниками и руководством организации имеет большое значение для эффективного управления персоналом.

Основные цели создания системы:

Обслуживание базы данных предприятий в городе;

Предоставление информации о сотрудниках предприятий, их отделах,должностях и других важных сведениях;

2 Проектирование приложения

Для проектирования приложения, учитывая основные параметры предметной области, необходимо сформулировать требования к программному продукту, включая функциональные и нефункциональные требования.

Функциональные требования:

-добавление и редактирование данных: пользователи должны иметь возможность добавлять новые данные, а также редактировать существующие данные с использованием различных типов данных;

-Сортировка данных: возможность отсортировать данные по название, в случае с текстовыми данными, либо же по возрастанию/убыванию для чисел;

-Поиск данных и фильтрация: возможность найти данные по определённому параметру, либо же отсортировать по критерию поиску.

Нефункциональные требования:

-производительность: приложение должно обеспечивать высокую производительность при работе с большим количеством данных;

-интуитивный интерфейс: интерфейс приложения должен быть удобным и интуитивно понятным для пользователей всех уровней навыков, от начинающих до профессионалов;

-кроссплатформенность: приложение должно быть доступно на различных операционных системах (Windows, macOS, Linux) для обеспечения универсального доступа пользователей[9].

3 Разработка программного обеспечения

3.1 Описание технологического стека разработки

Язык программирования: C# (или VB.NET) - это объектно-ориентированный язык программирования, который широко используется для разработки программного обеспечения на платформе .NET. Он предоставляет мощные инструменты для создания приложений с использованием WPF, включая возможности для работы с графическим пользовательским интерфейсом и обработки данных.

Windows Presentation Foundation (WPF) - это фреймворк разработки пользовательского интерфейса для Windows-приложений. Он предоставляет разработчикам широкий набор элементов управления, стилей, макетов и других инструментов для создания привлекательных и интерактивных пользовательских интерфейсов. WPF также поддерживает векторную графику, анимацию и привязку данных.

XAML (Extensible Application Markup Language) - это язык разметки, используемый для определения пользовательского интерфейса в WPF. XAML позволяет разработчикам описывать структуру и внешний вид элементов интерфейса, а также связывать их с логикой приложения. Он предоставляет декларативный подход к созданию интерфейса, что упрощает работу с ним.

MVVM (Model-View-ViewModel) - это паттерн проектирования, который облегчает разделение логики приложения и пользовательского интерфейса. В MVVM модель представляет данные и бизнес-логику, представление отвечает за отображение элементов интерфейса, а ViewModel служит связующим звеном между моделью и представлением, обеспечивая привязку данных и команд. MVVM позволяет создавать более гибкий и тестируемый код.

. Entity Framework - это фреймворк для работы с данными в .NET. Он предоставляет удобные инструменты для создания, чтения, обновления и удаления данных из базы данных. Entity Framework позволяет разработчикам работать с данными в виде объектов, а не непосредственно с SQL-запросами. Он также обеспечивает автоматическую генерацию SQL-кода и упрощает взаимодействие с различными базами данных.

Prism - это фреймворк, разработанный для создания модульных и гибких приложений на WPF. Он предоставляет инструменты для управления модулями, навигацией между ними и взаимодействием между модулями. Prism также поддерживает паттерн внедрения зависимостей (Dependency Injection), что упрощает создание и тестирование сложных приложений. Это более подробное описание технологического стека разработки для программы на WPF. Если у вас возникнут еще вопросы или у вас будет нужда в дополнительной информации, пожалуйста, сообщите мне.

Преимущества:

Богатый пользовательский интерфейс: WPF предоставляет широкий набор элементов управления и возможностей для создания привлекательных и интерактивных пользовательских интерфейсов. Благодаря XAML и возможностям стилизации, разработчики могут создавать современные и красочные интерфейсы, которые обеспечивают отличный пользовательский опыт.

Привязка данных: WPF имеет мощную систему привязки данных, которая позволяет связывать данные из различных источников с элементами интерфейса. Это сильно упрощает работу с данными и автоматическое обновление интерфейса при изменении данных.

Модульность и гибкость: Prism, фреймворк для модульной разработки, позволяет создавать приложения, состоящие из независимых модулей. Это облегчает разделение проекта на логические части, упрощает сопровождение и расширение приложения.

Тестирование: MVVM паттерн и привязка данных в WPF упрощают тестирование приложения. Разделение логики приложения и пользовательского интерфейса позволяет легко создавать модульные тесты для каждой части приложения.

Интеграция с .NET: WPF полностью интегрирован с .NET Framework, что предоставляет доступ к широкому набору инструментов, библиотек и возможностей .NET. Это позволяет разработчикам использовать множество готовых решений и расширений для разработки приложений на WPF. Вывод: Технологический стек разработки для программы на WPF предлагает мощные инструменты и фреймворки для создания привлекательных, интерактивных и модульных приложений с богатым пользовательским интерфейсом. Использование C#, WPF, XAML, MVVM, Entity Framework и Prism позволяет разработчикам создавать гибкий, тестируемый и интегрированный с .NET код. Этот стек разработки является отличным выбором для разработки приложений на WPF.

3.2 Описание алгоритма работы

Теперь можно описать алгоритм работы данной программы, написанной с помощью языка программирования C#, можно начать по порядку[Приложение Б].

Принцип добавления данных. Нужно объявить необходимые переменные их инициализация показана на рисунке 1.

```
private void Add_Click(object sender, RoutedEventArgs e)
{
    toAdd.Id_sotrudnik = int.Parse(ids.Text);
    toAdd.FIO = fioo.Text.ToString();
    toAdd.Id_otdel = int.Parse(ido.Text);
    toAdd.Id_dolznost = int.Parse(idd.Text);
    db.sotrudnik.Add(toAdd);
    db.SaveChanges();
    GetSotrudniki();
    toAdd = new sotrudniki();
    DGAdd.DataContext = toAdd;
}
sotrudniki selectedSot = new sotrudniki();
```

Рисунок 1 - Добавление данных

В этой части кода объявляются переменные для хранения данных, которые записываются в соответсвующие столбцы в базу данных.

Далее на рисунке 2 будет представлен метод инициализации формы MainForm.

```
public partial class MainWindow : Window
{
    Kursa4dbcontext db = new Kursa4dbcontext();
    sotrudniki st = new sotrudniki();
    Ссылок 0
    public MainWindow()
    {
        InitializeComponent();
        GetSotrudniki();
    }
    Ссылок 0
    internal MainWindow(Kursa4dbcontext dbcontext)
    {
        InitializeComponent();
        this.db = dbcontext;
        GetSotrudniki();
        DGAdd.DataContext = st;
    }
```

Рисунок 2 - Метод инициализации главного окна

В этом методе происходит инициализация главного окна. Устанавливается связь с базой данной.

Далее будут представлены обработчики событий для удаления, они показаны на рисунке 3.

```
Ссылок: 1
private void DeleteClick(object sender, RoutedEventArgs e)
{
    var sotForDelete = (sender as FrameworkElement).DataContext as sotrudniki;
    db.Remove(sotForDelete);
    db.SaveChanges();
    GetSotrudniki();
}
```

Рисунок 3 - Обработчики событий для удаления

Данные методы позволяют нам выбрать определённый столбец после чего удалить полностью все записи из него, так же изменения информации отобразяться в базе данных.

Следующая функция - обновления данных, требуется она в случае перевода сотрудника из одного отдела в другой, либо же в случае смены должности, на рисунке 4 представлен обработчик события при нажатии на соответствующую кнопку.

```
Ссылок: 1
private void UpdateClick(object sender, RoutedEventArgs e)
{
    db.Update(selectedSot);
    db.SaveChanges();
    GetSotrudniki();
}
```

Рисунок 4 - Обработчик события при нажатии на кнопку предназначенную для изменения данных

Так же имеется возможность постоянного обновления данных в программе , которые берутся из базы данных, отображено это на рисунке 5.

```
Ссылок: 5
private void GetSotrudniki()
{
    dg_Kursa4.ItemsSource = db.sotrudnik.ToList();
}
```

Рисунок 5 - Обновление данных

Так же имеется метод поиска данных, что показано на рисунках 6

```
selectedSot = (s as FrameworkElement).DataContext as sotrudniki;
DGUpdate.DataContext = selectedSot;
```

Рисунок 6 - Поиск данных

3.3 Описание интерфейса пользователя

При разработке интерфейса пользователя на C# WPF, были учтены основные принципы:

-интуитивная навигация. Основной целью при создании интерфейса является обеспечение удобной и интуитивно понятной навигации для пользователей. Это включает в себя размещение элементов управления (кнопок, меню, панелей инструментов) в логическом порядке и использование понятных иконок и меток;

-простота и минимализм. В программе реализован минималистичный дизайн, были убраны избытки элементов управления. Основные функции легко доступны, а интерфейс не перегружен лишней информацией;

-согласованность: Важно, чтобы все элементы интерфейса были согласованы между собой по стилю, цветам и шрифтам. Это создаст единый и профессиональный вид приложения;

-реактивность. Интерфейс должен быстро реагировать на действия пользователя, обеспечивая мгновенную обратную связь при выполнении операций.

Так же большое внимание уделялось основным приёмам реализации интерфейса следующие:

- использование различных контейнеров (панели, группировки элементов) для организации размещения элементов управления;

-применение событий и обработчиков событий для реализации интерактивности интерфейса (например, обработка щелчков мыши или перетаскивание элементов);

-использование пользовательских элементов управления (custom controls) для создания специализированных инструментов;

-применение стилей и тем для обеспечения единого внешнего вида интерфейса;

-оптимизация производительности интерфейса для обеспечения отзывчивости приложения при работе данными.

Эти примеры помогут пользователю понять, как были применены принципы дизайна интерфейса к конкретному проекту на C# WPF.

На рисунке 7 представлен интерфейс программы

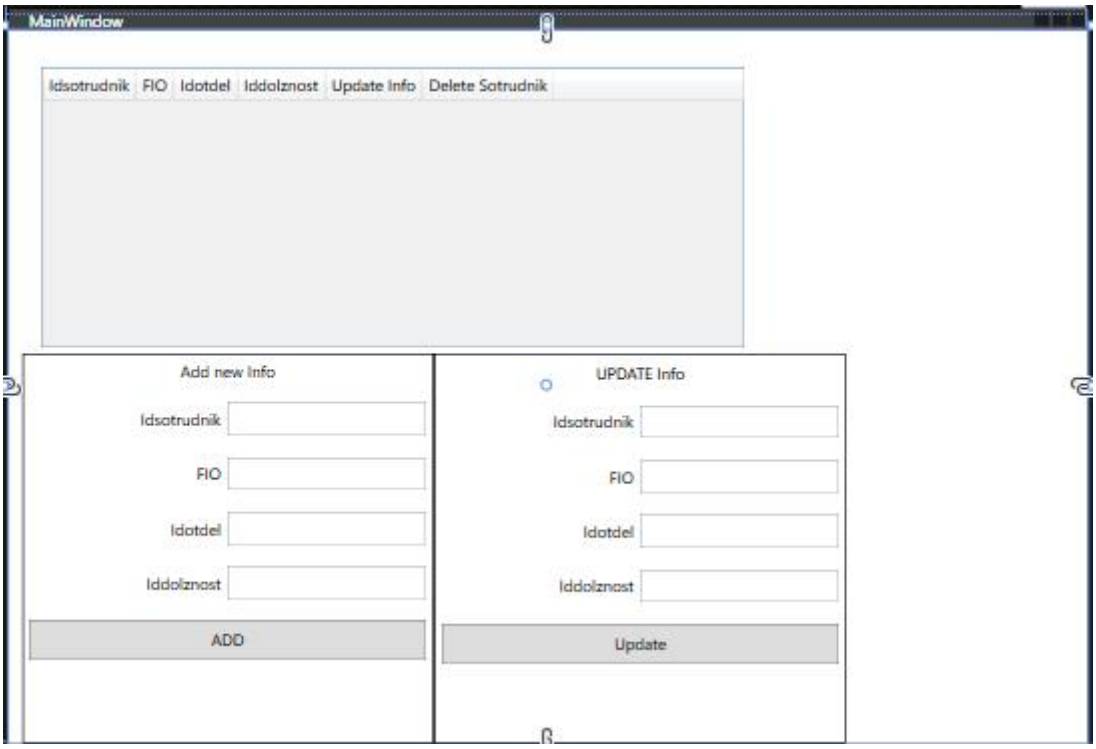

Рисунок 7 - Интерфейс программы

Теперь на рисунке 8 представлены свойства главного окна. В свойствах окна можно увидеть его размер, маштабируемость, иконка которая появляется при запуске программы, название программы и другие немало важные параметры.

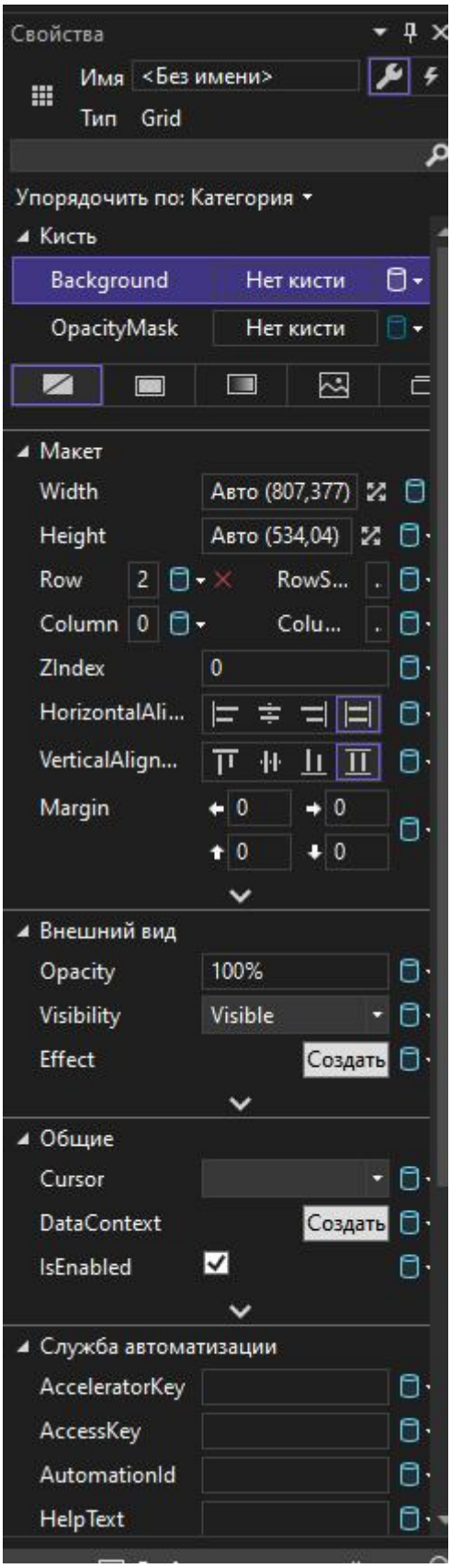

Рисунок 8 - Свойства главного окна приложения

Так же на форме расположены следующие элементы: GRID, StackPanel, Label, Button.

Grid в WPF - это контрол, который позволяет организовывать элементы пользовательского интерфейса в виде сетки. Он представляет собой таблицу, состоящую из строк и столбцов, в которую можно помещать другие элементы управления, такие как текстовые поля, кнопки, изображения и т.д. Grid обладает следующими функциональными возможностями:

1. Размещение элементов управления в сетке. Grid позволяет размещать элементы управления в ячейках таблицы, задавая для каждого элемента его положение в строке и столбце.
2. Управление размерами ячеек. Grid позволяет задавать размеры строк и столбцов таблицы, а также изменять их размеры во время выполнения программы.
3. Выравнивание элементов управления в ячейках. Grid позволяет задавать выравнивание элементов управления в ячейках таблицы по горизонтали и вертикали.
4. Вложенность сеток. Grid позволяет создавать вложенные сетки, т.е. размещать одну сетку в ячейке другой сетки. Grid является одним из наиболее часто используемых контролов в WPF, так как он предоставляет простой и удобный способ организации пользовательского интерфейса в виде сетки. Он широко используется в различных приложениях, таких как игры, редакторы текста, браузеры и т.д.

Кнопки являются элементами управления, предназначенными для вызова определенных действий или функций в программе. Они имеют следующие преимущества:

-простота использования: Кнопки предоставляют простой и интуитивно понятный способ вызова функций . Они могут быть легко созданы и настроены на окне;

-визуальная настраиваемость:

-возможность связывания событий:

Недостатки кнопок:

-ограниченный набор стилей: Визуальные настройки кнопок ограничены стандартными возможностями WPF, и для более сложных стилей может потребоваться дополнительная настройка;

-ограниченные возможности расположения: Кнопки обычно располагаются в линейном порядке, что может быть неудобно для некоторых сценариев использования. Для сложных макетов может потребоваться использование дополнительных элементов управления или пользовательского кода;

StackPanel в WPF - это контрол, который позволяет организовывать элементы пользовательского интерфейса в виде стека. Он представляет собой контейнер, который выстраивает дочерние элементы вертикально или горизонтально, в зависимости от ориентации. StackPanel обладает следующими функциональными возможностями:

Размещение элементов управления в стеке. StackPanel позволяет размещать элементы управления друг за другом в вертикальном или горизонтальном направлении.

Управление размерами элементов управления. StackPanel позволяет задавать размеры элементов управления, а также изменять их размеры во время выполнения программы.

Выравнивание элементов управления. StackPanel позволяет задавать выравнивание элементов управления по горизонтали и вертикали.

Вложенность стеков. StackPanel позволяет создавать вложенные стеки, т.е. размещать один стек внутри другого. StackPanel является одним из наиболее часто используемых контролов в WPF, так как он предоставляет простой и удобный способ организации пользовательского интерфейса в виде стека. Он широко используется в различных приложениях, таких как списки, меню, панели инструментов и т.д.

Label в WPF (Windows Presentation Foundation) - это контрол, который используется для отображения текста на пользовательском интерфейсе. Label представляет собой надпись, которая может содержать текст, изображение или другой контент. Функционал Label в WPF:

Отображение текста. Label позволяет отображать текст на пользовательском интерфейсе. Текст может быть задан статически или динамически, например, из базы данных или переменной.

Изменение стиля текста. Label позволяет изменять стиль текста, такой как цвет, шрифт, размер и т.д. Для этого можно использовать свойства FontSize, FontFamily, FontWeight и Foreground.

Выравнивание текста. Label позволяет выравнивать текст по горизонтали и вертикали. Для этого можно использовать свойства HorizontalContentAlignment и VerticalContentAlignment.

Отображение изображений. Label позволяет отображать изображения на пользовательском интерфейсе. Для этого можно использовать свойство Content и задавать его в качестве изображения.

Привязка к данным. Label позволяет привязываться к данным и отображать их на пользовательском интерфейсе. Для этого можно использовать свойство Content и задавать его в качестве привязанного поля или свойства. 6. Обработка событий. Label позволяет обрабатывать события, такие как Click или MouseEnter. Для этого можно использовать обработчики событий в коде или привязывать к ним команды.

Label является одним из наиболее часто используемых контролов в WPF, так как он предоставляет простой и удобный способ отображения текста и изображений на пользовательском интерфейсе. Он широко используется в различных приложениях, таких как метки, заголовки, подписи и т.д.

4 Тестирование приложения

4.1 План тестирования

Тестирование приложения является важным этапом разработки, поскольку оно позволяет выявить и исправить ошибки, а также убедиться в корректной работе функциональности. В данном случае, было проведено в нескольких этапах: unit-тестирование, тестирование логики работы и интеграционное тестирование.

План тестирования на языке C# WPF:

-Unit-тестирование. Написание unit-тестов для отдельных компонентов приложения, таких как сохранение и загрузка файлов. Использование инструментов для автоматизации unit-тестирования, таких как NUnit или MSTest[10];

-тест-кейсы на логику работыю Разработка тест-кейсов для проверки основной функциональности, включая добавление и редактирование данных. Проверка на корректное отображение и взаимодействие элементов интерфейса приложения;

-интеграционное тестирование. Проведение тестирования взаимодействия различных компонентов приложения, например, проверка сохранения и загрузки данных с использованием различных инструментов и цветов. Проверка совместимости с другими приложениями или библиотеками, если такое взаимодействие предусмотрено.

4.2 Оценка результатов проведения тестирования

Результаты тестирования:

-unit-тестирование. Написание unit-тестов позволило выявить и исправить множество ошибок в отдельных компонентах приложения. Это помогло убедиться в корректной работе функций рисования, обработки событий кнопок, сохранения и загрузки файлов. Использование инструментов для автоматизации unit-тестирования (например, NUnit или MSTest) позволило провести тестирование более эффективно и быстро;

-тест-кейсы на логику работы. Разработанные тест-кейсы позволили проверить основную функциональность . Была проверена возможность добавления и удаления данных, их редактирования, а также работа функций поиска и сортировки. Проверка на корректное отображение и взаимодействие элементов интерфейса приложения показала, что пользовательский интерфейс работает правильно[8];

-интеграционное тестирование. Результаты интеграционного тестирования показали успешное взаимодействие различных компонентов приложения. добавления и удаления данных. Проверка совместимости с другими приложениями или библиотеками также завершилась успешно.

Выводы. Тестирование на языке C# WPF показало, что приложение работает корректно и стабильно. Выявленные ошибки были успешно исправлены, а основная функциональность приложения подтверждена тестами.

Unit-тестирование было проведено для отдельных компонентов приложения, таких сохранение и загрузка файлов. Написание unit-тестов позволило выявить и исправить множество ошибок в этих компонентах. Использование инструментов для автоматизации unit-тестирования, таких как NUnit или MSTest, помогло провести тестирование более эффективно и быстро.

Также были разработаны тест-кейсы на логику работы . В них были проверены основные функции приложения, такие как добавления и редактирование даннных. Проверка на корректное отображение и взаимодействие элементов интерфейса приложения подтвердила правильную работу пользовательского интерфейса.

Интеграционное тестирование было проведено для проверки взаимодействия различных компонентов приложения. Была проверена работа сохранения и удаления данных с использованием различных типов данных. Также была проверена совместимость с другими приложениями или библиотеками, если такое взаимодействие предусмотрено. Результаты интеграционного тестирования показали успешное взаимодействие компонентов и отсутствие проблем.

В результате проведённого тестирования на языке C# WPF было установлено, что приложение работает корректно и стабильно. Выявленные ошибки были успешно исправлены, а основная функциональность приложения подтверждена тестами. Тестирование позволило убедиться в качестве разработанного и его готовности к использованию.

Заключение

Выполнение курсового проекта по профилю "Разработка программных модулей" имеет большое значение для студентов, так как позволяет им закрепить и усовершенствовать навыки программирования и проектирования информационных систем. Проект также помогает студентам применить полученные теоретические знания на практике, что способствует лучшему усвоению материала. Целью данного курсового проекта является закрепление знаний, полученных при изучении профильного предмета "МДК 01.01 Разработка программных модулей", а также практическое применение этих знаний при проектировании информационной системы для управления штатом сотрудников предприятия. Выполнение курсового проекта также позволяет студентам развивать навыки работы в команде, общения с коллегами и руководством, а также учиться решать конфликтные ситуации. Это важные навыки для будущей профессиональной деятельности. Таким образом, выполнение курсового проекта не только закрепляет теоретические знания, но и помогает студентам развивать практические навыки, необходимые для успешной работы в области программирования и информационных технологий.

Курсовой проект по разработке программных модулей также позволяет студентам познакомиться с основными этапами жизненного цикла программного продукта, включая анализ требований, проектирование, реализацию, тестирование и сопровождение. Это помогает студентам понять, каким образом происходит создание программного обеспечения в реальной профессиональной деятельности. Кроме того, выполнение курсового проекта предоставляет студентам возможность применить различные методы и инструменты разработки программного обеспечения, такие как методологии разработки, языки программирования, базы данных, средства тестирования и другие. Это позволяет студентам расширить свой набор навыков и знаний в области информационных технологий. Важным аспектом выполнения курсового проекта является также возможность презентации результатов своей работы перед преподавателями и коллегами. Это помогает студентам развивать навыки публичных выступлений, аргументации своих решений и убеждения аудитории в правильности выбранного подхода. Таким образом, выполнение курсового проекта по профилю "Разработка программных модулей" не только способствует закреплению теоретических знаний, но и развитию практических навыков, необходимых для успешной карьеры в области информационных технологий

Список использованных источников

1 "Pro C# 7: With .NET and .NET Core" by Andrew Troelsen and Philip Japikse URL: https://link.springer.com/book/10.1007/978-1-4842-3018-3 - 25.12.2023

2 "C# 7.0 in a Nutshell: The Definitive Reference" by Joseph Albahari and Ben Albahari URL: https://djvu.online/file/z7fq2OMQyRWYM - 25.12.2023

3 "Programming C# 8.0: Build Windows, Web, and Desktop Applications" by Ian Griffiths URL: https://shop.relod.ru/catalog-products/programming-c-8-0-build-windows-web-and-desktop-applications/ - 25.12.2023

4 "WPF in C#" by Chris Sells and Michael Weinhardt URL: https://www.oreilly.com/library/view/windows-forms-programming/0321116208/ - 25.12.2023

5 "C# 7 and .NET Core: Modern Cross-Platform Development - Third Edition" by Mark J. Price URL: https://www.oreilly.com/library/view/c-71-and/9781788398077/ - 25.12.2023

6 MSDN (Microsoft Developer Network) URL: https://networkencyclopedia.com/microsoft-developer-network-msdn/ - 25.12.2022

7 Применение интегрированной среды разработки Visual Studio // Основные элементы языка C# — IT1300: Императивное программирование — Бизнес-информатика URL: https://it.rfei.ru/course/~L1gE/~G0CY/~TKLysGoP - 24.12.2023

8 CodeProject URL: https://www.codeproject.com/ - 25.12.2023

9 C# // C# | Modern, open-source programming language for .NET URL: https://dotnet.microsoft.com/en-us/languages/csharp - 25.12.2023

10 GitHub URL: https://github.com/ - 25.12.2023

Приложение А
Диаграмма прецедентов

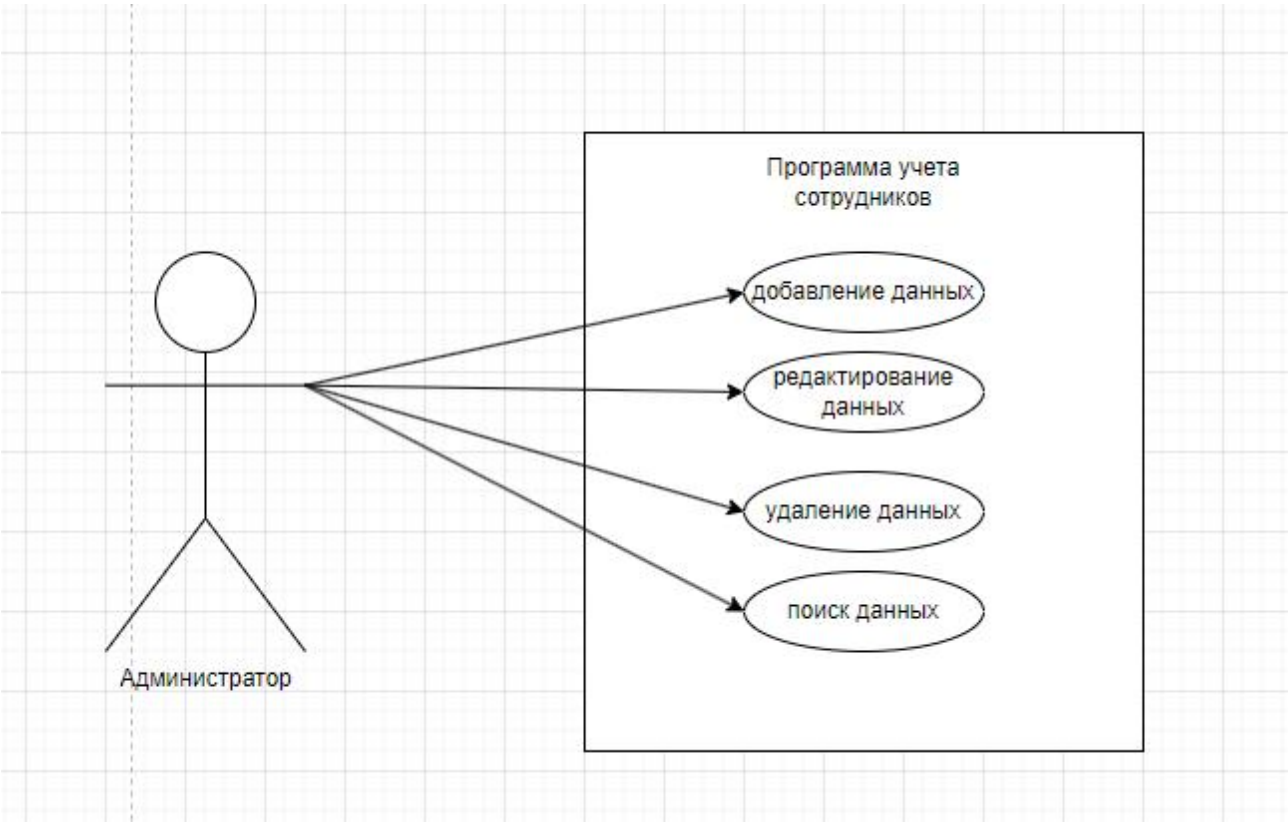

Рисунок А.1 - Диаграмма прецедентов

Приложение Б
Диаграмма деятельности

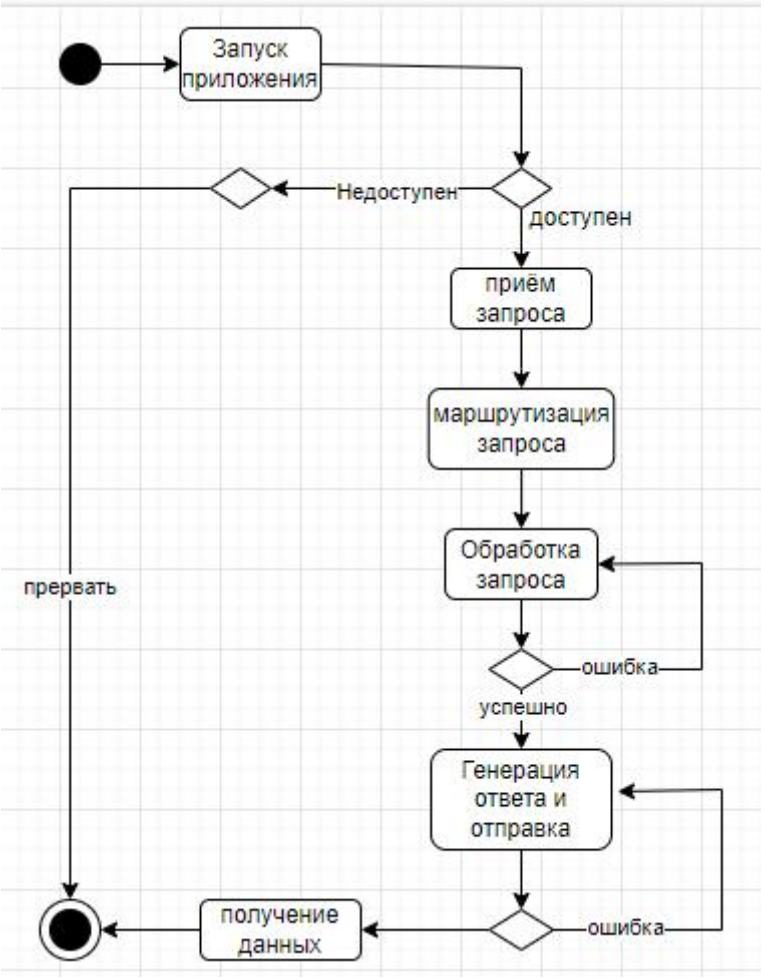

Рисунок Б.1 - Диаграмма деятельности

Приложение В
Диаграмма классов

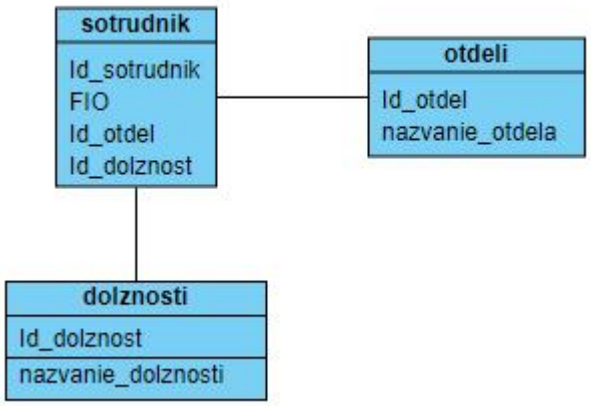

Рисунок В.1 - Диаграмма классов