# МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ижевский государственный технический университет  
 имени М.Т. Калашникова»

Институт непрерывного профессионального образования

Кафедра «Прикладная математика и информационные технологии»

|  |  |
| --- | --- |
|  | «Выпускная квалификационная работа допущена к защите»  Руководитель специальности «Прикладная информатика (по отраслям)»  к. ф.-м. н., доцент  «\_\_» \_\_\_\_ 2022 г.\_\_\_\_\_\_\_\_ А. А. Айзикович |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

«Разработка поисковой системы»

|  |  |
| --- | --- |
| Студент  Группы Д18-021-1б  «10» июня 2022 г. | К.Р. Гусев |
| Руководитель выпускной квалификационной работы  Преподаватель ИНПО  «10» июня 2022 г. | А.В. Селезнева |
| Нормоконтроль  Старший преподаватель | Н.А. Рычина |

«10» июня 2022 г.

Ижевск

2022г

**ЗАДАНИЕ**

**На выпускную квалификационную работу**

**Студенту группы Д18-021-1б Гусеву Кириллу Родионовичу**

1. **Тема работы**: Разработка поисковой системы.
2. **Цель и задачи**:

Целью является разработка поискового приложения, посредством Visual Studio 2022 Community Edition с помощью «Windows Forms». Данное приложение позволит искать информацию в файлах со следующими расширениями:

1. «docx»;
2. «txt»;
3. «pdf».

В соответствии с поставленной целью в работе должны быть решены следующие задачи:

1. изучить теоретический материал по данной теме;
2. разработать графический интерфейс;
3. разработать алгоритмы поиска;
4. протестировать приложение;
5. исправить ошибки.
6. **Краткое содержание и объем работы**.

Выпускная квалификационная работа должна состоять из введения, трех глав, заключения, списка используемой литературы и приложений.

Во введении необходимо раскрыть актуальность темы выпускной квалификационной работы, поставить цель и обозначить задачи.

В первой главе необходимо изучить язык программирования C#, платформу «.net», раскрыть понятие компьютерного зрения и рассмотреть библиотеки языка C#.

Во второй главе будет описан процесс разработки приложения с использованием машинного зрения для извлечения информации конкретно из файлов «pdf».

В третьей главе привести результаты тестирования приложения, исправить ошибки.

Итоги работы представить в заключении.

Подготовить список используемой литературы.

Приложения должны содержать: ссылку на приложение, код приложения.

1. **Срок сдачи студентом законченной выпускной квалификационной работы на кафедру** «10» июня 2022г.

|  |  |
| --- | --- |
| Руководитель выпускной квалификационной работы | Селезнева А.В. |
| Руководитель специальности 09.02.05 «Прикладная информатика (по отраслям)» | Айзикович А.А. |

**ОГЛАВЛЕНИЕ**

ВВЕДЕНИЕ 3

ГЛАВА 1. ИЗУЧЕНИЕ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА 5

1.1. Язык C# и платформа «.NET» 5

1.2. Необходимые библиотеки С# 7

1.3. Компьютерное зрение 8

1.4. Windows Forms 9

ГЛАВА 2. РАЗРАБОТКА ПРИЛОЖЕНИЯ 11

2.1. Создание графического интерфейса приложения 11

2.2. Программный код 20

ГЛАВА 3. ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ 29

3.1. Тестирование программного обеспечения 29

3.2. Исправление ошибок 33

ЗАКЛЮЧЕНИЕ 37

СПИСОК ЛИТЕРАТУРЫ 38

ПРИЛОЖЕНИЕ А

## **ВВЕДЕНИЕ**

**Актуальность данной темы** обусловлена тем, что в процессе деятельности, как бытовой, так и профессиональной, современный человек постоянно сталкивается с необходимостью найти информацию в файлах.

**Суть проблемы работы**: существующие аналоги приложения не всегда верно распознают информацию в файлах.

В файлах с расширением «pdf» не всегда информация представлена в виде текста, например, в виде картинки или скана документа. И из-за этого существующие приложения не могут найти информацию в данных файлах.

Для извлечения информации из файлов формата «pdf» будет предоставляться пользователю выбор либо использоваться компьютерное зрение, предварительно конвертируя страницы в изображения, либо обыкновенный поиск без возможности поиска информации с изображений.

**Цель исследования**: разработка приложения для поиска информации с использованием компьютерного зрения.

**Задачи**:

1. изучить теоретический материал по теме;
2. разработать графический интерфейс;
3. разработать алгоритмы поиска;
4. протестировать приложение;
5. исправить ошибки.

**Объект исследования**: windows forms со сторонними библиотеками.

**Предмет исследования**: программное обеспечение (поисковое приложение).

Поставленные цели, сформулированные задачи, предмет и объект исследования обусловили структуру работы, которая состоит из введения, трёх глав, заключения, списка использованной литературы и ряда приложений, иллюстрирующих основное содержание выпускной квалификационной работы.

В данной работе будет создано программное обеспечение для поиска содержимого в файлах. Теоретической частью работы будет анализ библиотек языка C#, а также возможности windows forms. В практической части будет представлен разбор создания приложения, отчеты после ее тестирования, разбор ошибок.

Данная выпускная квалификационная работа отличается высокой **практической значимостью**. Созданное в данном проекте программное обеспечение найдет широкое применение, как среди обычных пользователей, так и сотрудников компаний.

В дальнейшем, возможно, данный проект может иметь поддержку, ввиду того, что существует множество форматов файлов, а также появление новых.

Глава 1. Изучение теоретического материала

* 1. Язык C# и платформа «.NET»

На данный момент язык программирования C# один из самых мощных и быстро развивающихся языков в ИТ-отрасли. На нем пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

Язык программирования C# уже не молодой, как и вся платформа «.NET» уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio «.NET» в феврале 2002 года [15].

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java.

Данный язык является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И язык C# продолжает активно развиваться, и с каждой новой версией в нем появляется все больше интересных функций [1].

Язык C# был создан специально для работы с фреймворком «.NET», однако само понятие «.NET» несколько шире [12].

Фреймворк «.NET» представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты [13]:

1. основой платформы является общеязыковая среда исполнения «Common Language Runtime», благодаря чему .NET поддерживает несколько языков: наряду с C# это VB.NET, C++, F#, а также различные диалекты других языков, привязанные к «.NET», например, Delphi.NET. При компиляции код на любом из этих языков «переводится» в сборку на общем языке CIL (Common Intermediate Language). Поэтому при определенных условиях мы можем сделать отдельные модули одного приложения на отдельных языках;
2. кроссплатформенность – данный фреймворк является переносимой платформой. Используя различные технологии на платформе «.NET», можно разрабатывать приложения на языке C# для самых разных платформ –Windows, MacOS, Linux, Android, iOS;
3. мощная библиотека классов «.NET» представляет единую для всех поддерживаемых языков библиотеку классов. Вне зависимости от того, какое приложение будет написано на языке программирования C#, будет задействована библиотека классов «.NET»;
4. общеязыковая среда исполнения «Common Language Runtime», и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке предназначены технологии ADO.NET и Entity Framework Core. Для построения графических приложений с богатым насыщенным интерфейсом – технологии WPF и WinUI, для создания более простых графических приложений –Windows Forms. Для разработки кроссплатформенных мобильных и десктопных приложений – Xamarin [14] или MAUI. Для создания веб-сайтов и веб-приложений – ASP.NET и т.д;
5. к этому стоит добавить активной развивающийся и набирающий популярность Blazor – фреймворк, который работает поверх «.NET» и который позволяет создавать веб-приложения как на стороне сервера, так и на стороне клиента. А в будущем будет поддерживать создание мобильных и, возможно, десктоп-приложений [4];
6. согласно ряду тестов веб-приложений, построенных с помощью других технологий, отличаются меньшей производительностью чем приложения, написанные на «.NET»;
7. также еще следует отметить такую особенность языка C# и фреймворка «.NET», как автоматическая сборка мусора. А это значит, что в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.
   1. Необходимые библиотеки C#

Для написания данной работы потребуется стандартный набор библиотек, который подключаются автоматически при создании проекта, а также сторонние библиотеки, установленные вручную. Кроме того, могут потребоваться права администратора для создания папок, файлов и их чтения.

Библиотека «System.Threading.Tasks» упрощает работу с многопроцессорными, многоядерными система. Кроме того, она упрощает работу по созданию новых потоков. Поэтому обычно рекомендуется использовать именно ее классы для создания многопоточных приложений. В основе этой библиотеки лежит концепция задач, каждая из которых описывает отдельную продолжительную операцию. В библиотеке классов «.NET» задача представлена специальным классом – классом «Task», который находится в пространстве имен «System.Threading.Tasks». Данный класс описывает отдельную задачу, которая запускается асинхронно в одном из потоков из пула потоков. Хотя ее также можно запускать синхронно в текущем потоке [5].

Пространство имен «System.IO» в «.NET» – это область библиотек базовых классов, посвященная службам файлового ввода-вывода, а также ввода-вывода из памяти. Многие типы из пространства имен «System.IO» сосредоточены на программных манипуляциях физическими каталогами и файлами. Дополнительные типы предоставляют поддержку чтения и записи данных в строковые буферы, а также области памяти [6].

Для извлечения текста из документов «docx» и «pdf» отлично подойдет сторонняя библиотека «TikaOnDotNet.TextExtraction» [18].

Библиотека «BitMiracle.Docotic.Pdf» – позволит конвертировать документ с расширением «pdf» в изображения для дальнейшего взаимодействия с ними [19].

Библиотека «Emgu.CV», которая считывает информацию с изображения, будет нужна, так как в файлах с расширением «pdf» информация не всегда представлена в виде текста, например, в виде картинки или скана документа. По этой причине существующие приложения не могут найти информацию в данных файлах [20].

* 1. Компьютерное зрение

Компьютерное зрение (Computer Vision, CV) — это область искусственного интеллекта, связанная с анализом изображений и видео. Она включает в себя набор методов, которые наделяют компьютер способностью «видеть» и извлекать информацию из увиденного [2].

Чтобы научить компьютер «видеть», используются технологии машинного обучения. Собирается множество данных, которые позволяют выделить признаки и комбинации признаков для дальнейшей идентификации похожих объектов [3].

OpenCV (библиотека компьютерного зрения с открытым исходным кодом) – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков. Может свободно использоваться в академических и коммерческих целях — распространяется в условиях лицензии BSD [7].

Emgu CV является кроссплатформенным «.NET» дополнением для библиотеки OpenCV для обработки изображений. Разработано для работы с «.NET» совместимыми языками, такими как C #, VB, VC ++, IronPython и т.д., может быть использовано в Visual Studio, Xamarin, работает с Windows, Linux, Mac OS X, IOS, Android и Windows Phone [10].

* 1. Windows Forms

Windows Forms – это технология интеллектуальных клиентов для «NET Framework». Она представляет собой набор управляемых библиотек, упрощающих выполнение стандартных задач, таких как чтение из файловой системы и запись в нее. С помощью такой среды разработки, как Visual Studio, можно создавать интеллектуальные клиентские приложения Windows Forms, которые отображают информацию, запрашивают ввод от пользователей и обмениваются данными с удаленными компьютерами по сети [8].

В Windows Forms форма – это визуальная поверхность, на которой выводится информация для пользователя. Обычно приложение Windows Forms строится путем помещения элементов управления на форму и написания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления – это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных.

При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события с помощью кода и обрабатывает события при их возникновении.

Windows Forms включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы. Список всех элементов управления, которые можно использовать в форме, представлены в разделе Элементы управления для использования в формах Windows Forms. Если существующий элемент управления не удовлетворяет потребностям, в Windows Forms можно создать пользовательские элементы управления с помощью класса UserControl.

В состав Windows Forms входят многофункциональные элементы пользовательского интерфейса, позволяющие воссоздавать возможности таких сложных приложений, как Microsoft Office. Используя элементы управления «ToolStrip» и «MenuStrip», можно создавать панели инструментов и меню, содержащие текст и рисунки, подменю и другие элементы управления, такие как текстовые поля и поля со списками.

Используя функцию перетаскивания конструктора Windows Forms в Visual Studio, можно легко создавать приложения Windows Forms. Достаточно выделить элемент управления курсором и поместить его в нужное место на форме. Для преодоления трудностей, связанных с выравниванием элементов управления, конструктор предоставляет такие средства, как линии сетки и линии привязки. И при использовании Visual Studio, и при компиляции из командной строки вы можете использовать элементы управления FlowLayoutPanel, TableLayoutPanel и SplitContainer для создания сложных макетов форм за меньшее время.

Наконец, если нужно создать свои собственные элементы пользовательского интерфейса, пространство имен System.Drawing содержит широкий набор классов, необходимых для отрисовки линий, кругов и других фигур непосредственно на форме.

Глава 2. Разработка приложения

* 1. Создание графического интерфейса приложения

Стоит начать с того, что, обращаясь к мобильному приложению или сайту, пользователь намеревается найти конкретную информацию, которая нужна ему здесь и сейчас, а значит, его не должны стеснять проблемы, связанные с неудобной навигацией, затянутыми откликами приложения на его действия. Поэтому рассмотрим основные моменты, на которые стоит обратить внимание и которые помогут в разработке более удобного и эффективного приложения.

Интерфейс должен быть простым и понятным с первого взгляда, у пользователя не должно возникать вопросов, как взаимодействовать с приложением для достижения собственных целей. Кроме того, число шагов, которые сделает пользователь для достижения цели должно быть минимальным. У пользователя, работающего с грамотно разработанным интерфейсом, не возникнет вопросов, и он не сможет сделать ошибку, производя то или иное действие, пользователь должен понимать на интуитивном уровне, какие изменения это повлечет.

Требования к дизайну мобильных приложений достаточно простые. Цветовая гамма приложения не должна быть излишне яркой, пестрой, чтобы не рябить в глазах пользователя, однако сочетания цветов должны быть контрастными – это эффектно выглядит на дисплеях и, что самое главное, позволяет продолжать пользоваться приложениях в неудачных условиях освещенности. Дизайн должен быть простым и понятным, нужно избегать непонятных иконок, нетривиальных пояснений. Система навигации должна быть такой же простой, понятной. Если в приложении планируется создание большого количества разных разделов, их можно разделить на небольшое количество обобщенных, считается что для получения нужной информации пользователь готов пройти через 4 уровня вложенности. Списки, с помощью которых также может быть реализовано меню не должны превышать 5-9 элементов. Можно уведомлять пользователя о тех или иных изменениях с помощью всплывающих окон.

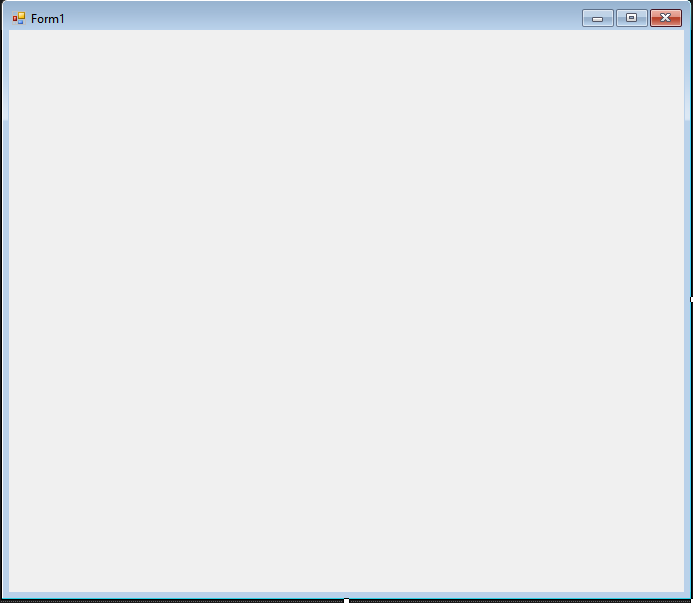
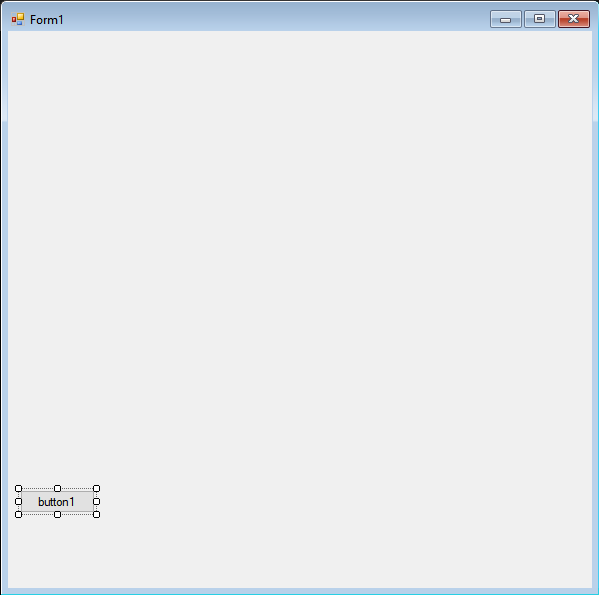
После создания проекта у нас появляется пустое приложение, в которое можем добавлять нужные элементы (Рис. 2.1).

Рисунок 2.1. Пустое приложение

Приступим к добавлению элементов. Сначала добавим кнопку, по нажатию на которую, будет происходить поиск. Для этого перетаскиваем элемент «button» на форму (Рис. 2.2).

 Рисунок 2.2. Добавление кнопки «поиск»

После, добавляем поля «textbox», в которых будут записываться искомые слова, а также месторасположение, по которому, в дальнейшем, будет происходить поиск. Для этого, также, перетащим с панели элементов два «textbox», как это было представлено выше (Рис. 2.3).

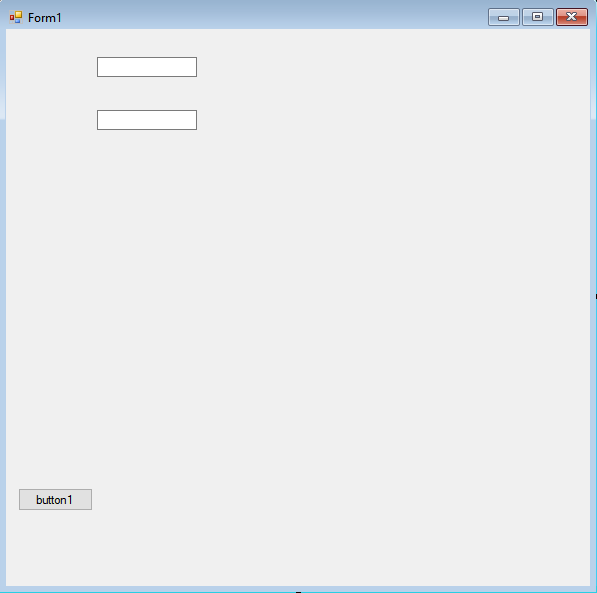
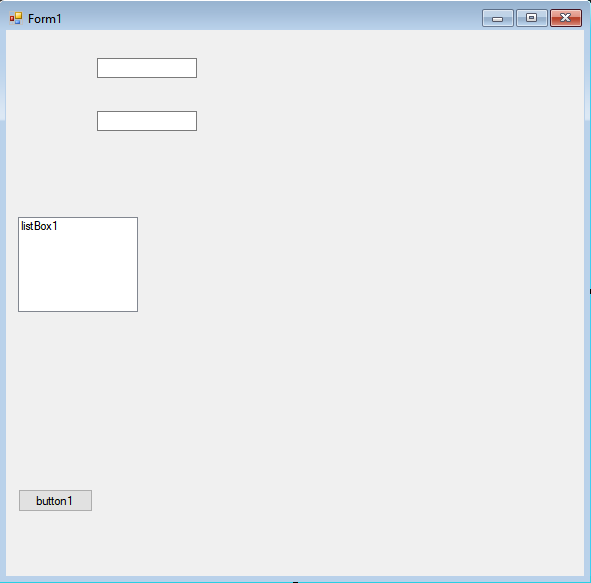


Рисунок 2.3. Добавление текстовых полей

Во время выполнения алгоритма поиска, при удачном результате, нужно отображать все найденные файлы, где встречается искомая фраза. С этой задачей отлично спариваться элемент «listbox». Перетаскиваем данный элемент в окно приложения (Рис. 2.4).

 Рисунок 2.4. Добавление «listbox»

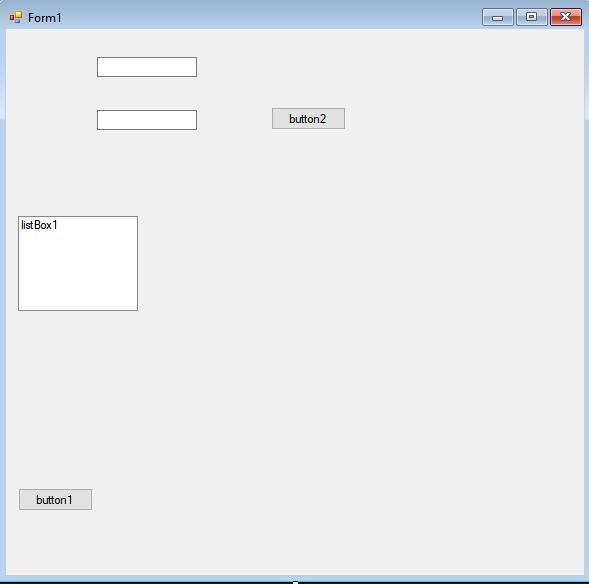
Следующим шагом будет добавление в приложение кнопки, благодаря которой, сможем выбирать директорию или файл, в зависимости от ситуации, где будет происходить поиск (Рис. 2.5).

Рисунок 2.5. Добавление кнопки «обзор»

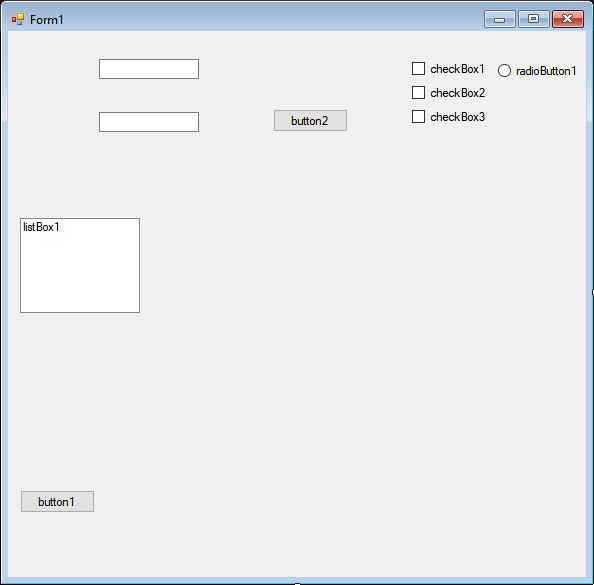
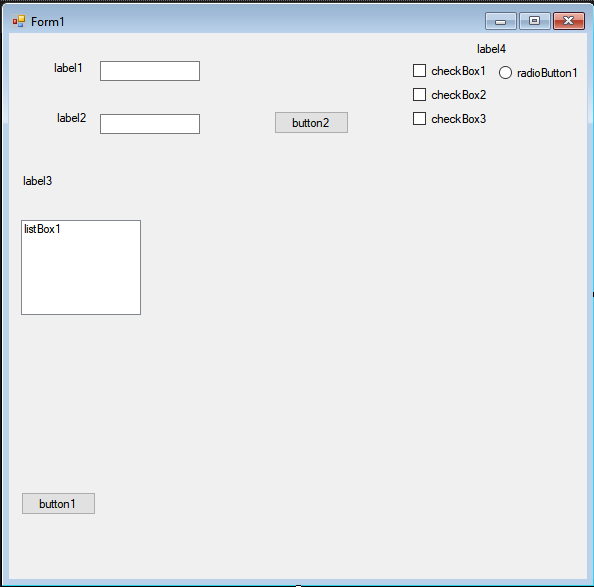
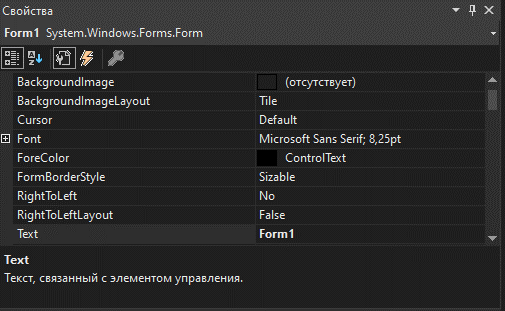
Дальше добавляем «checkbox» и «radiobutton», чтобы пользователь мог выбирать между стандартным поиском и с использованием компьютерного зрения. В случае выбора стандартного поиска, можно будет из трех предложенных вариантов расширений файлов исключить одно расширение или же два (Рис. 2.6).

Рисунок 2.6. Добавление «checkbox» и «radiobutton»

Одним из последних этапов в создании интерфейса будет добавление ко всем, выше созданным, элементам подписей. Для этого используем элемент «метка» или же «label» (Рис. 2.7.).

 Рисунок 2.7. Добавление меток

Заключительным этапом станет придание, созданным элементам, нормальный вид, чтобы пользователь мог без труда определить, что и где находится. Для этого воспользуемся, у каждого элемента, окном свойств (Рис. 2.8.).

 Рисунок 2.8. Окно свойств

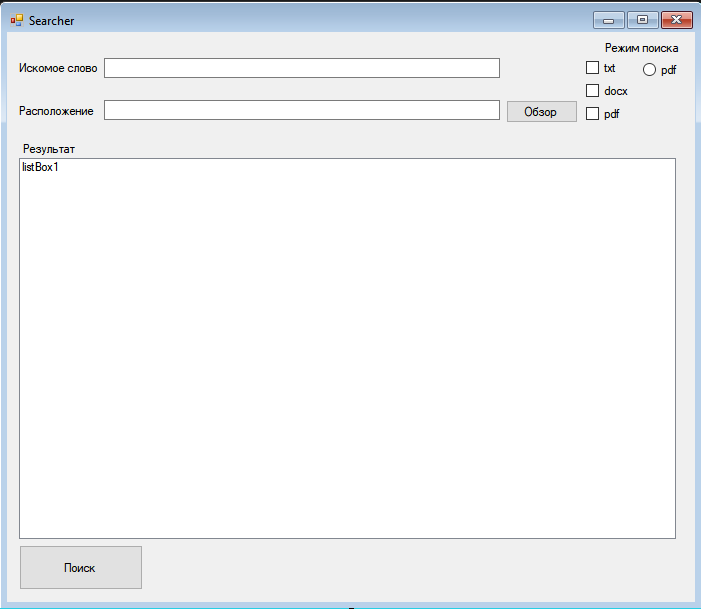
После выполненных выше указанных пунктов, графический интерфейс готов (Рис. 2.9.).

Рисунок 2.9. Готовый графический интерфейс

* 1. Программный код

Графический интерфейс готов, значит можно писать код, чтобы приложение начало выполнять свои функции.

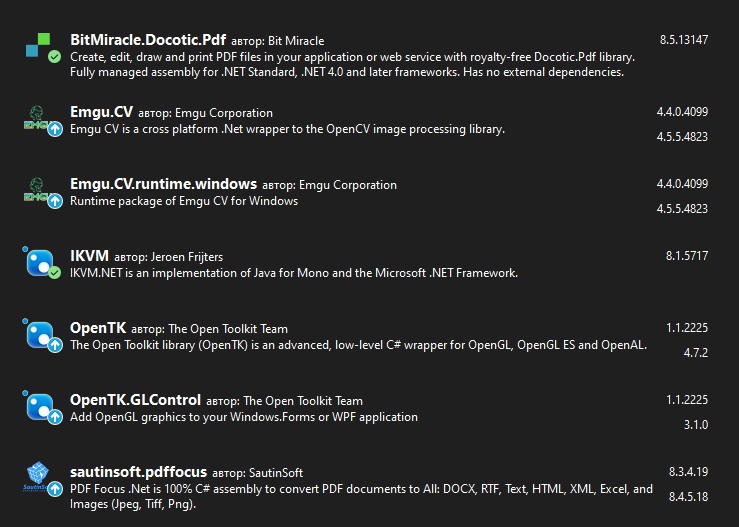
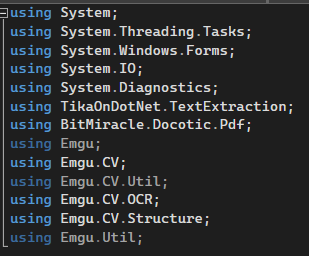
Первым шагом будет скачать необходимые библиотеки. С помощью «nuget» скачиваем их, если они не являются стандартными [9] (Рис. 2.10.).

Рисунок 2.10. Скаченные библиотеки

Для использования установленных библиотек необходимо их подключение в программном коде (Рис.2.11.).

 Рисунок 2.11. Подключение библиотек

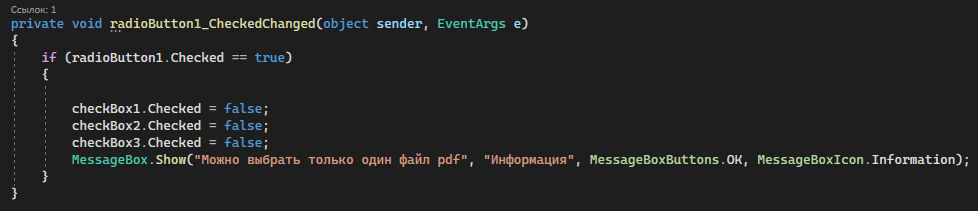
Первым делом реализуем выбор режима поиска. Когда активно любое поле «checkbox», будет происходить стандартный поиск, при этом делая поле «radiobutton» не выбранным. В случае, когда выбран «radiobutton», будет происходить поиск с использованием компьютерного зрения, делая не активными поля «checkbox» (Рис.2.12) (Рис. 2.13).

Рисунок 2.12. Выбор «radiobutton»

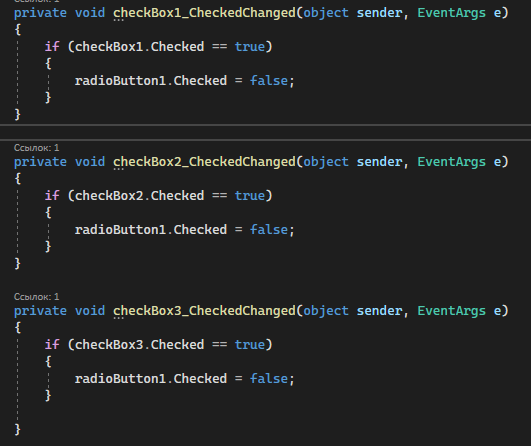


Рисунок 2.13. Выбор «checkbox»

При запуске приложения нужно вывести в строку «расположение» (путь), по которому находится это приложение. Так же, для использования компьютерного зрения, необходимо знать расположение обученных моделей (Рис.2.14.) [16].

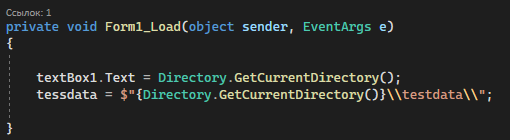


Рисунок 2.14. Расположение программы и обученных моделей

К кнопке «обзор» добавить следующие действия (Рис. 2.15.):

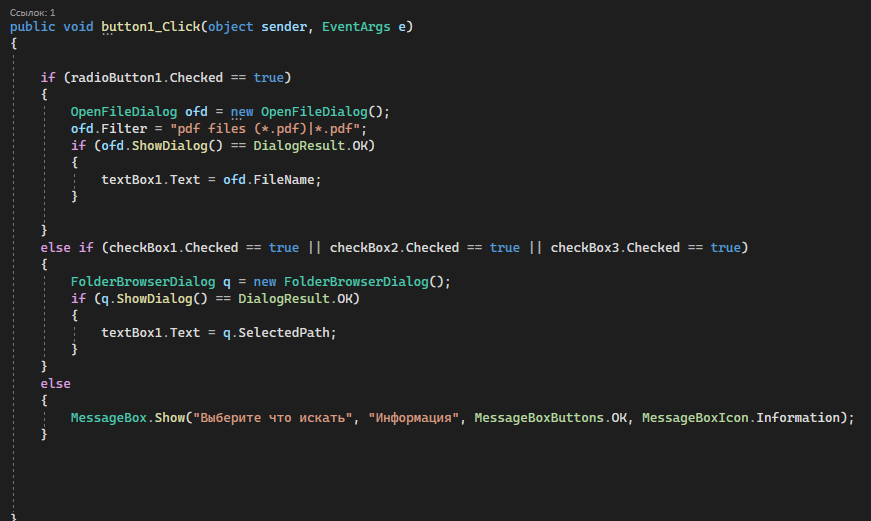
1. если ничего не выбрано, то выводить соответствующие сообщение;
2. если выбран какой-то из «checkbox», то открывать файловый диалог, с предложением пользователю выбрать директорию, где осуществлять поиск и записывать в «расположение»;
3.  а если выбран «radiobutton», то открывать уже файловый диалог с выбором только одного файла формата «pdf» и так же записать путь в «расположение».

Рисунок 2.15. Кнопка «обзор»

По нажатию на кнопку поиск должно происходить следующие (Рис. 2.16.):

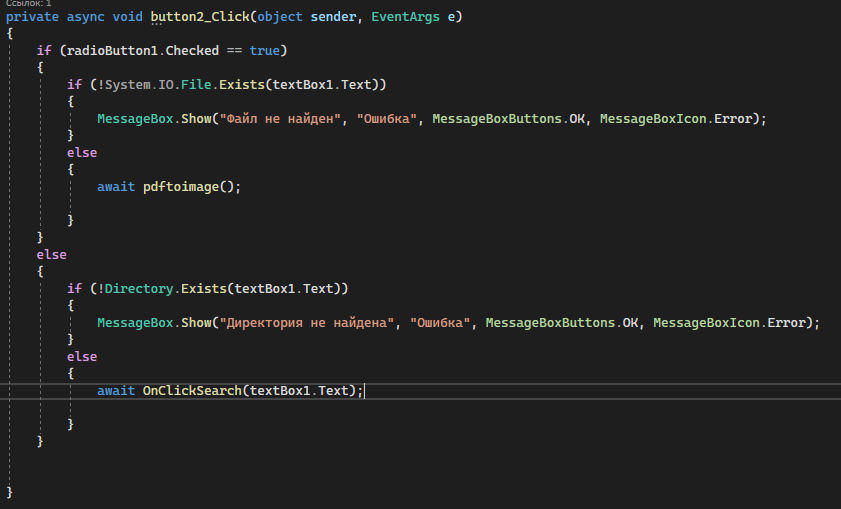
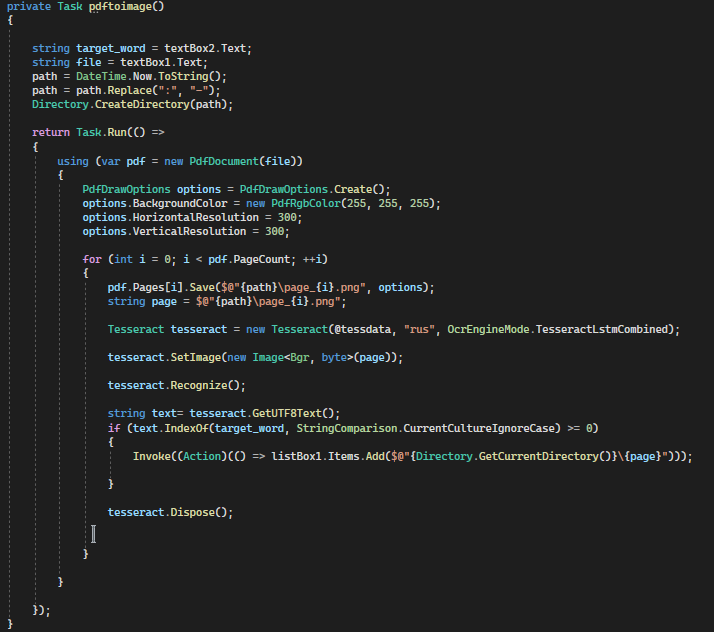
1. проверять, если нажат «radiobutton», существует ли данный файл по выбранному расположению, если да, то вызывать функцию поиска с использованием компьютерного зрения, иначе выводить информацию о том, что файл не существует;
2. если «radiobutton» не выбран, то проверять существует ли выбранный путь в системе, если да, то вызвать функции стандартного поиска, иначе выдавать ошибку.

Рисунок 2.16. Кнопка «поиск»

Далее, если был выбран вариант с компьютерным зрением, асинхронно выполнятся функция (Рис. 2.17.) [11] [17]. Алгоритм функции следующий:

1. записываются в переменные искомое слово и путь до файла;
2. создается папка, где будут храниться созданные изображения;
3. каждая страница преобразовывается в изображение;
4. с помощью компьютерного зернения и заранее обученных моделей, осуществляется поиск искомого слова;
5. если есть совпадение, добавить путь до изображения в «listbox».

 Рисунок 2.17. Алгоритм компьютерного зрения

При стандартном поиске происходит проверка условия - выбраны все, несколько или один «checkbox». При выполнении данного условия, происходит поиск всех документов с нужным расширением и затем передаётся в качестве аргумента в соответствующие функции. Если нет нужных документов, то выведется сообщение об их отсутствии.

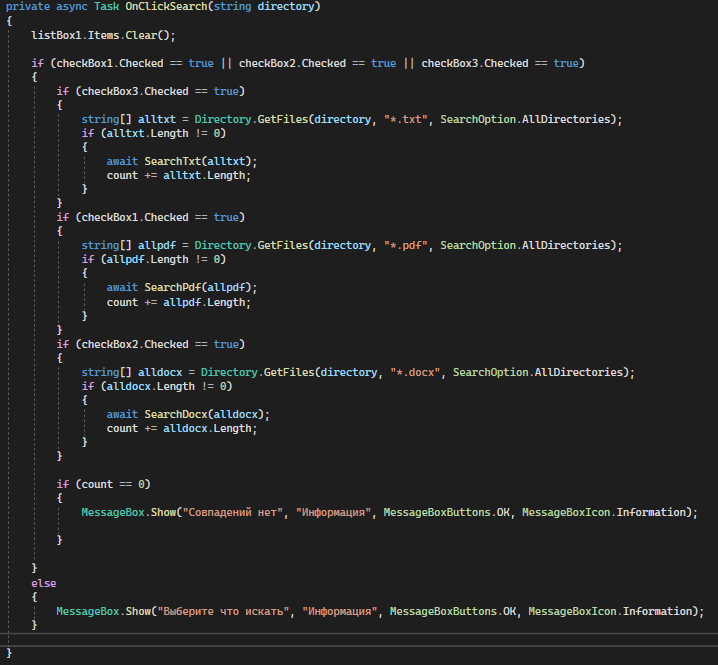
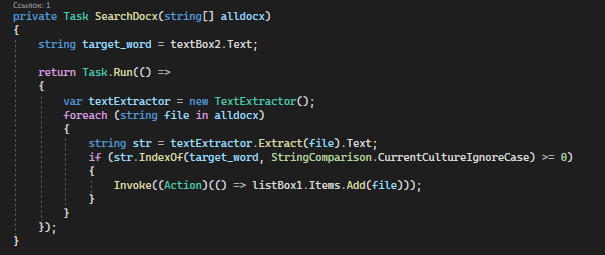
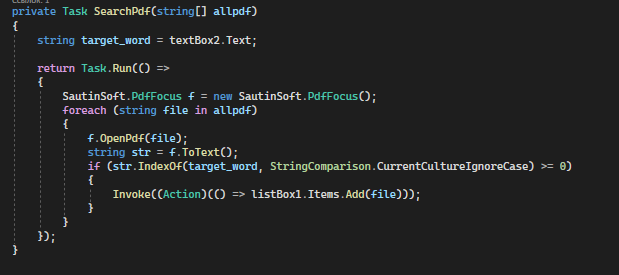
В ситуации, когда выше упомянутое условие не выполнено появится сообщение, о необходимости выбрать режим поиска (Рис. 2.18.).

Рисунок 2.18. Функция выбора

Стандартный алгоритм поиска текста заключается в следующем (Рис. 2.19.) (Рис. 2.20.) (Рис. 2.21.):

1. открывается файл;
2. происходит считывание всего текста;
3. поиск в считанном тексте;
4. если поиск удачный, добавляется путь до файла в «listbox».

 Рисунок 2.19. Алгоритм поиска в «docx» файлах

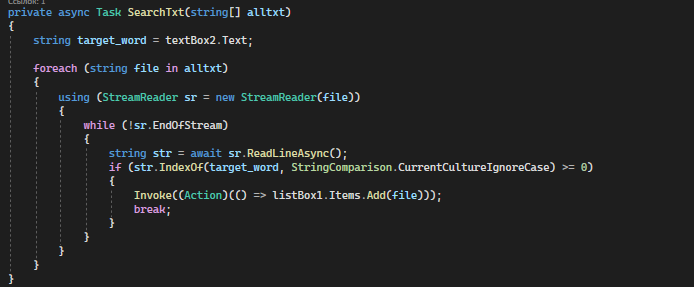
 Рисунок 2.20. Алгоритм поиска в «pdf» файлах

Рисунок 2.21. Алгоритм поиска в «txt» файлах

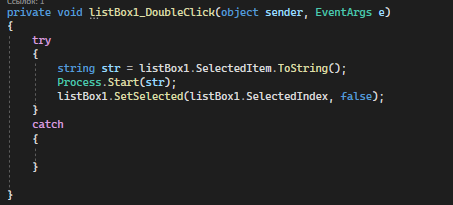
После поиска в «listbox» появились файлы, далее необходимо сделать так, чтобы по двойному нажатию открывался выбранный файл (Рис. 2.22.).

Рисунок 2.22. Открытие файла из «listbox»

Разработка завершена и приложение готово к тестированию.

Код всей программы будет в приложениях после доработок и исправлений, по выводам из третьей главы.

Глава 3. ТЕстирование приложениЯ

* 1. Тестирование программного обеспечения

После создания любого программного обеспечения необходимо протестировать его на ошибки или недочёты, чтобы в дальнейшем оно хорошо работало и было успешным. Одному разработчику не провести качественный анализ созданного приложения, поэтому необходимо привлечь мнение со стороны. Тестировщики должны быть внимательными к деталям и уметь развёрнуто описывать свои наблюдения.

Для тестирования приложения независимым пользователям было предложено воспользоваться приложением и оценить по следующим критериям:

1. наличие ошибок;
2. удобство интерфейса;
3. скорость выполнения операций.

Командой тестировщиков был произведён подробный анализ. Во время тестирования созданного программного обеспечения были выявлены следующие недочёты:

1. при изменении размера самой формы, элементы внутри её не изменяли свой размер. Если растягивать форму, тем самым увеличивая её в размерах, то это не критично, в обратном случае, некоторые элементы можно не наблюдать (Рис. 3.1.) (Рис.3.2.);

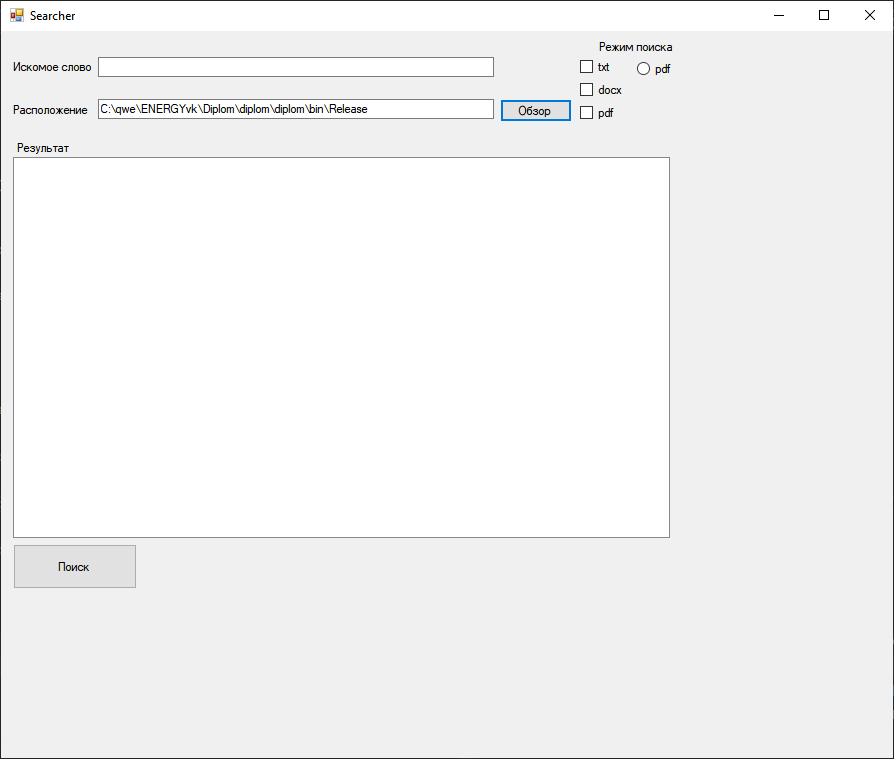
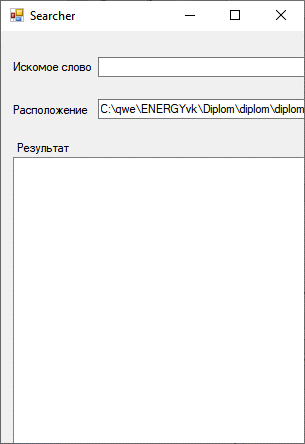
 Рисунок 3.1. Увеличение размера формы

Рисунок 3.2. Уменьшение размера формы

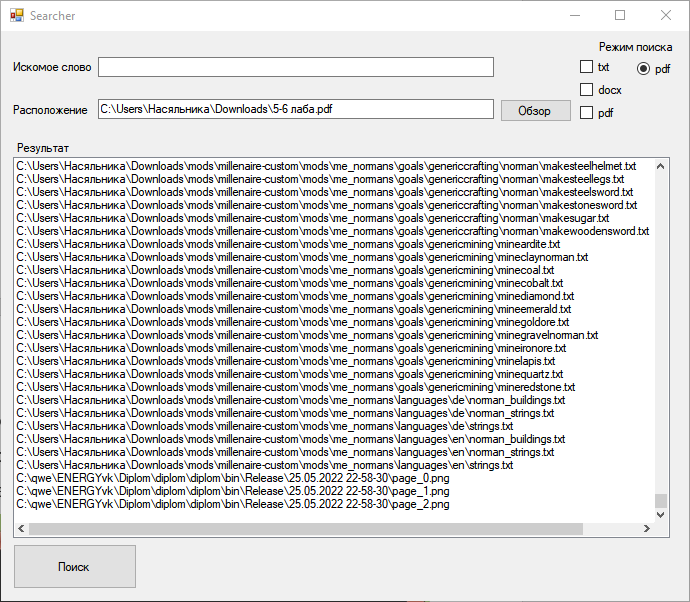
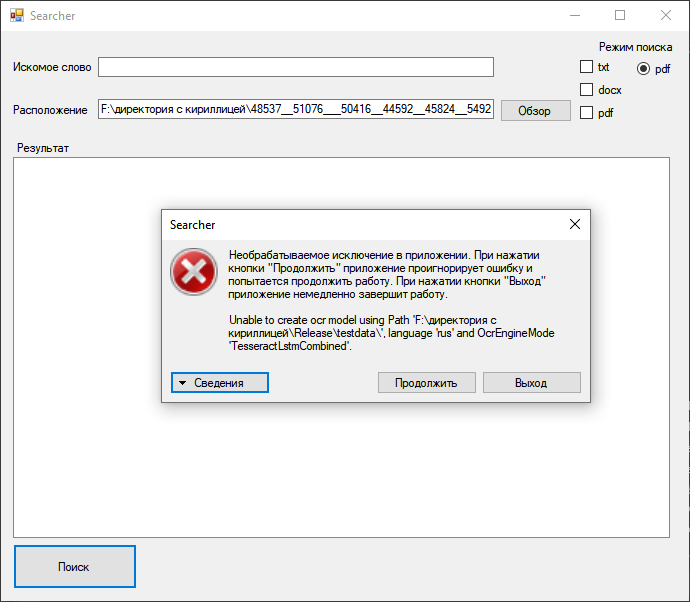
1. при пустом поле «искомое слово» поиск происходит медленнее, чем могло бы быть, особенно это заметно при использовании компьютерного зрения;
2. отсутствует проверка на тип файла, когда используется компьютерное зрение;
3. не очищается элемент «listbox» при поиске с помощью компьютерного зрения (Рис. 3.3.);

Рисунок 3.3. Тестирование приложения

1. при использовании приложения из директории, с использованием кириллицы, возникает ошибка (Рис. 3.4.);

 Рисунок 3.4. Директория с кириллицей

1. невозможно остановить поиск во время его выполнения, при повторном нажатии на кнопку поиск начнется заново, после того как закончится предыдущий;
2. отсутствие возможности просмотра прогресса поиска.

Учтя все пожелания и замечания тестировщиков, исправив ошибки и доработав приложение, можно будет его выложить в открытый доступ.

## **Исправление ошибок**

Приняв во внимание выявленные в ходе тестирования недочёты и ошибки, была совершена работа по их исправлению и корректировке.

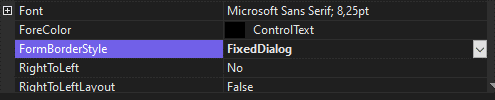
Первым делом было принято решение исправить изменение размера формы. Это делается очень просто, на панели свойств формы напротив «FormBorderStyle» указать «FixedDialog» (Рис. 3.5.). После этого размеры формы невозможно изменить.

Рис. 3.5. Свойства формы

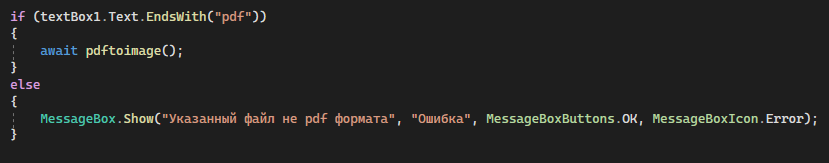
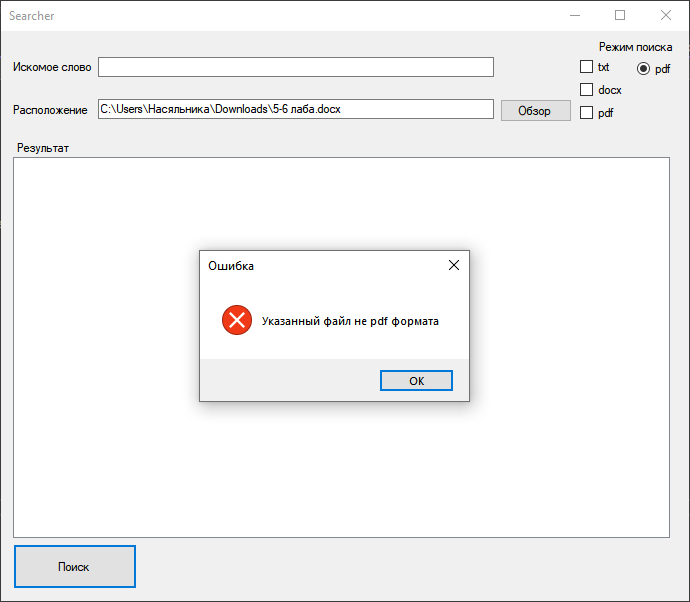
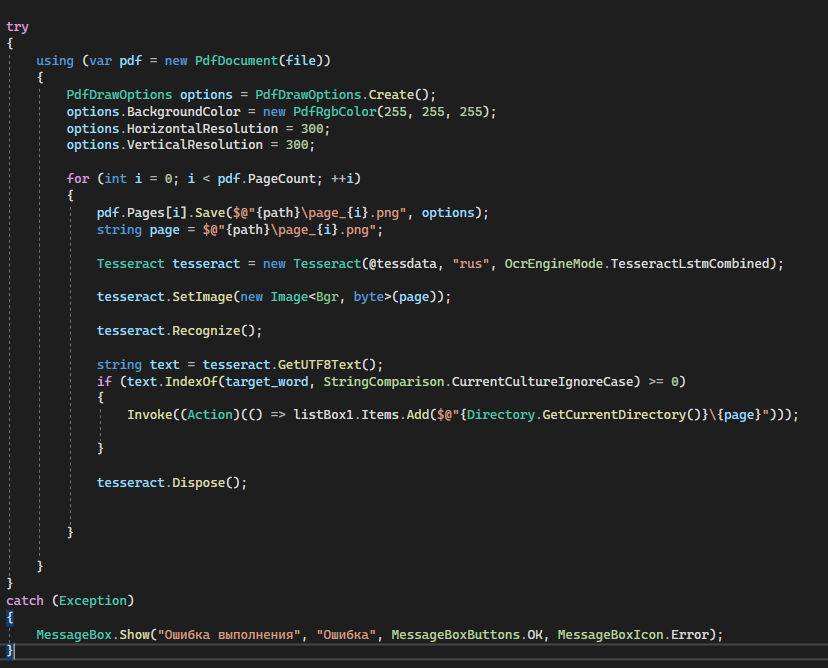
Следующим шагом будет добавление проверки на формат файла, при использовании компьютерного зрения. Достигается данный пункт, благодаря одному условному блоку и функцией «EndsWith», которая проверяет кончается ли выбранная строка на указанные символы (Рис. 3.6.). Если выбрать любое другое расширение файлов, будет показываться уведомление о недопустимом расширении файла (Рис.3.7).

Рис. 3.6. Условный блок

 Рисунок 3.7. Уведомление

Так же нужно очищать поле «результаты поиска» при поиске, с использованием компьютерного зрения. Данная проблема решается, написанием строчки «listBox1.Items.Clear()» в функцию, где добавляются результаты работы библиотеки «Tesseract».

После анализа ошибки, связанной с кириллицей, было выявлено, что ошибка возникает при использовании компьютерного зрения. Для её обработки подойдет оператора «try-cathch» (Рис. 3.8.). Оператор «try-catch» состоит из блока «try», за которым следует одно или несколько предложений «catch», задающих обработчики для различных исключений. При возникновении исключения общеязыковая среда выполнения ищет оператор «catch», который обрабатывает это исключение.

 Рисунок 3.8. Оператор «try-catch»

Следующим шагом будет ускорение поиска, если поле «искомое слово» пустое. На данный момент происходит поиск в содержимом, что сказывается на быстроте выполнения. Для этого будет проверяться пустая ли строка, если да, то в окне вывода будут записываться все файлы с указанным расширением.

Предпоследним шагом в доработке, будет добавление кнопки «остановить», по нажатию на которую останавливается процесс выполнения поиска. Реализовать это легко – необходимо проверять каждый раз является ли переменная, которая создана для текущей задачи, нужного значения. По нажатию на кнопку значение будет меняться, из-за чего сработает условие, и работа прекратится.

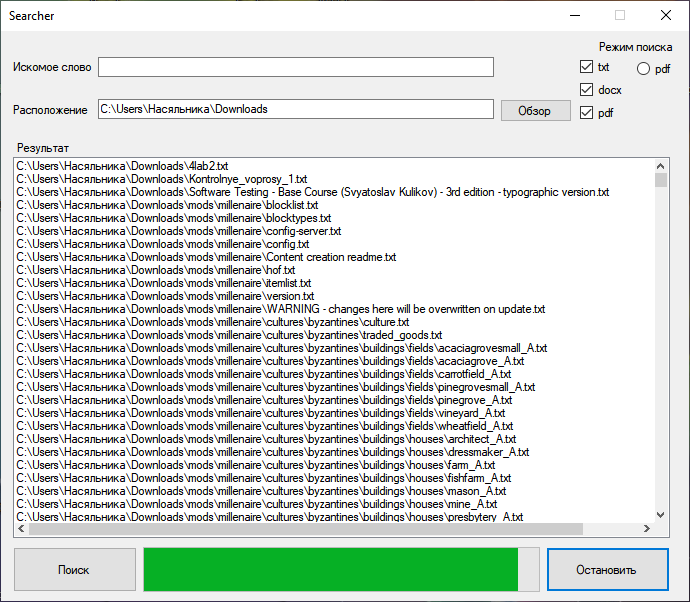
Последним шагом будет добавление прогрессбара, чтобы пользователь мог видеть работает программа или нет. Для этого создадим элемент «progressbar»(Рис. 3.9.).

Рисунок 3.9. Конечный вид приложения

После исправления всех ошибок и недочётов группе пользователей было предложено ещё раз протестировать приложение. Все пользователи отозвались положительно и отметили, что пользоваться приложением стало намного приятнее.

# **ЗАКЛЮЧЕНИЕ**

Создание приложения – это трудоёмкий процесс, включающий в себя немалое количество шагов к реализации поставленных задач, а также физических вложений. В ходе работы были успешно реализованы все поставленные задачи:

1. изучен теоретический материал по данной теме;
2. разработан графический интерфейс;
3. разработан алгоритмы поиска;
4. протестирован приложение;
5. исправлены ошибки.

В теоретической части работы дано определение языку программирования С#, а также были рассмотрены необходимые библиотеки C#. Раскрыто понятие компьютерного зрения. Также произведен анализ платформы «.net» и «windows forms».

В ходе работы над практической частью было создано программное обеспечение в виде приложения. Для этого были применены технологии асинхронных функций. Также с целью выявления ошибок и недочетов было проведено тестирование приложения и исправление ошибок.

Главная цель работы была достигнута: все пользователи, задействованные в тестировании продукта, подтвердили, что с использованием приложения время на поиск файлов и информации сократилось, следовательно, удалось создать продукт, отвечающий заданным требованиям.

Результаты проекта: создано программное обеспечение, направленное на поиск файлов, а также информации в них.

# **СПИСОК ЛИТЕРАТУРЫ**

* + - 1. Язык С# [Электронный ресурс] – Режим доступа: https://melimde.com/yazik-c-i-platforma-net.html (Дата обращения 01.03.2022).
      2. Компьютерное зрение [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Компьютерное\_зрение (Дата обращения 01.03.2022).
      3. Применение компьютерного зрения [Электронный ресурс] – Режим доступа: https://trends.rbc.ru/trends/industry/5f1f007e9a794756fafbfa83 (Дата обращения 01.03.2022).
      4. Что такое Blazor [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/blazor/1.1.php (Дата обращения 01.03.2022).
      5. Библиотека «system.threading.tasks» [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/api/system.threading.tasks?view=net-6.0 (Дата обращения 20.04.2022).
      6. Библиотека «System.IO» [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/api/system.io?view=netcore-3.1 (Дата обращения 20.04.2022).
      7. Применение OpenCV [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/260741/ (Дата обращения 20.04.2022).
      8. Понятие Windows Forms [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-5.0 (Дата обращения 20.04.2022).
      9. Понятие NuGet [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/nuget/what-is-nuget (Дата обращения 20.04.2022).
      10. OCR для PDF [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/507226/ (Дата обращения 20.04.2022).
      11. Асинхронные методы [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/tutorial/13.3.php (Дата обращения 20.04.2022).
      12. Фреймворк «.net» [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/.NET\_Framework (Дата обращения 20.04.2022).
      13. Понятие фрейморка [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Фреймворк (Дата обращения 20.04.2022).
      14. «Xamarin» [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/xamarin/ (Дата обращения 20.04.2022).
      15. Visual studio [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019 (Дата обращения 20.04.2022).
      16. EmguCV в C# [Электронный ресурс] – Режим доступа: https://www.youtube.com/watch?v=d5YiW3B5ka8&t=1021s (Дата обращения 20.04.2022).
      17. Использование tesseract в C# [Электронный ресурс] – Режим доступа: https://www.youtube.com/watch?v=iFjGkOl95vM&t=286s (Дата обращения 20.04.2022).
      18. Библиотека «tikaondotnet» [Электронный ресурс] – Режим доступа: https://github.com/KevM/tikaondotnet (Дата обращения 20.04.2022).
      19. Библиотека «bitmiracle» [Электронный ресурс] – Режим доступа: https://bitmiracle.com/pdf-library/ (Дата обращения 20.04.2022).
      20. Библиотека «emgu» [Электронный ресурс] – Режим доступа: https://www.emgu.com/wiki/index.php/Main\_Page (Дата обращения 20.04.2022).

ПРИЛожение А

https://drive.google.com/drive/folders/1zRIbp4JgIt\_SrGvw6KzttkC6nvWbFihx?usp=sharing