минобрнауки россии

федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |  |
| --- | --- |
| Институт (факультет) | Информационных технологий |
| Кафедра | Математическое и программное обеспечение ЭВМ |

КУРСОВАЯ РАБОТА

|  |
| --- |
| по дисциплине С# - программирование |

|  |  |
| --- | --- |
| на тему | Разработка программного обеспечения на языке C# |
|  | |

|  |
| --- |
| Выполнил студент группы 1ПИб-01-1оп-21 |
| *группа* |
| направления подготовки (специальности) |
| 09.03.04 Программная инженерия |
| *шифр, наименование* |
| Фомина Ксения Сергеевна |
| *фамилия, имя, отчество* |

|  |
| --- |
| Руководитель |
| Шаханов Н.И. |
| *фамилия, имя, отчество* |
| Доцент |
| *должность* |

|  |
| --- |
| Дата представления работы |
| «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 \_\_\_ г. |
|  |
| Заключение о допуске к защите |
|  |
|  |
|  |
|  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| количество баллов |
| Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2024

*Год*

Оглавление

[Введение 3](#_Toc168575334)

[1. Описание предметной области 4](#_Toc168575335)

[2. Организация работы в команде 5](#_Toc168575336)

[3. Проектирование программного обеспечения 7](#_Toc168575337)

[3.1. Выбор платформы 7](#_Toc168575338)

[3.2. Фреймворка Prism 7](#_Toc168575339)

[3.2. MVVM 8](#_Toc168575340)

[4. Проектирование базы данных 10](#_Toc168575341)

[Заключение 11](#_Toc168575342)

[Список литературы 12](#_Toc168575343)

[Приложение 1. Техническое задание 13](#_Toc168575344)

[Приложение 2. Текст программы 20](#_Toc168575345)

[Приложение 3. Руководство пользователя 21](#_Toc168575346)

# Введение

В эпоху цифровизации и активного развития технологий обработки изображений, возникает необходимость в создании эффективных инструментов для автоматизации процессов, связанных с анализом визуальной информации. Одним из ключевых аспектов такой автоматизации является автоматическая разметка изображений — процесс, при котором машина самостоятельно идентифицирует и классифицирует различные объекты на цифровых фотографиях и графических изображениях. Разработка приложения по автоматической разметке изображений стоит на пересечении областей компьютерного зрения, машинного обучения и искусственного интеллекта и представляет собой сложную задачу, требующую глубоких знаний и креативного подхода.

Цель данной работы — разработка программного приложения, способного автоматически распознавать и маркировать объекты на изображениях для их дальнейшего анализа или использования в системах, основанных на визуальных данных. Важность такого приложения трудно переоценить, учитывая растущий объем визуального контента, который ежедневно генерируется пользователями и организациями по всему миру.

# Описание предметной области

Автоматическая разметка и обработка сгенерированных изображений — это комплексная область, занимающаяся изучением и разработкой методов и алгоритмов для создания, анализа и модификации визуального контента. Она охватывает процессы генерации изображений с помощью компьютерных технологий и последующей их автоматизированной разметки для различных целей, включая, но не ограничиваясь, машинное обучение, компьютерное зрение и автоматизированное тестирование.

# Организация работы в команде

Правильная организация работы в команде важна для достижения общих целей проекта и повышения эффективности работы.

Для работы в команде существует большое количество различных инструментов.

Для отслеживания прогресса в решении задач использовалась доска задач в сервисе Notion — это относительно новый сервис для создания заметок и текстовых документов, списков дел, баз данных, таблиц, канбан-досок, баз знаний, ведения проектов и совместной работы. Наиболее значимыми в рамках работы стали возможности совместной работы и ведения досок задач (рис.1).

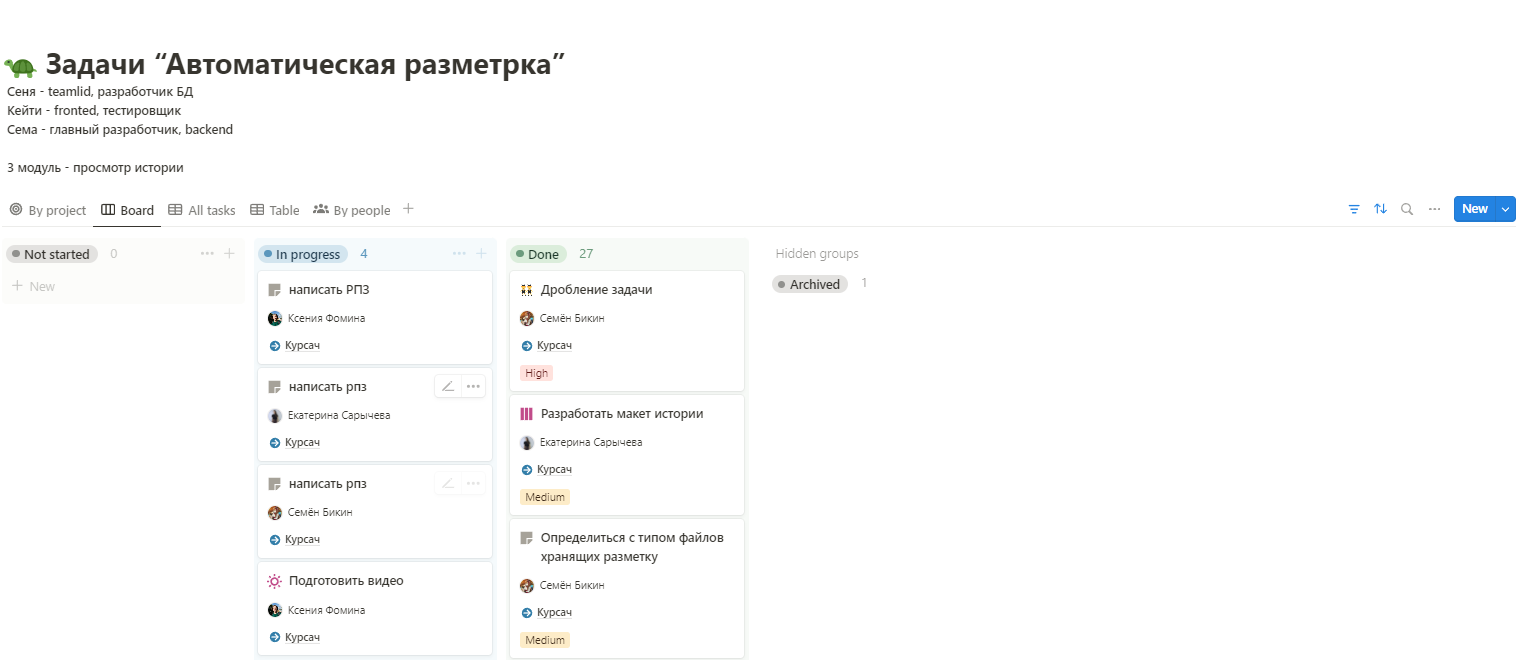


Рис.1. Доска задач в Notion

Лидер команды должен формулировать задачи, следить за их выполнением, назначать ответственных и инициировать организационные задачи с определенной периодичностью, для отслеживания прогресса и назначения новых задач, определения векторов последующих действий.

В качестве платформы для собраний был выбран Discord — это бесплатный мессенджер, который позволяет обмениваться голосовым, видео и текстовым чатом с друзьями, игровыми сообществами и разработчиками. Помимо возможностей организовывать собрания в голосовых чатах, в текстовых чатах можно делать разветвление на различные ветки (рис.2).

Кроме необходимости организовывать собрания и отслеживания прогресса в решении задач команде разработчиков нужна платформа для контроля версий проекта. Git — это консольная утилита, для отслеживания и ведения истории изменения файлов в проекте (рис.3). Чаще всего его используют для кода, но можно и для других файлов.

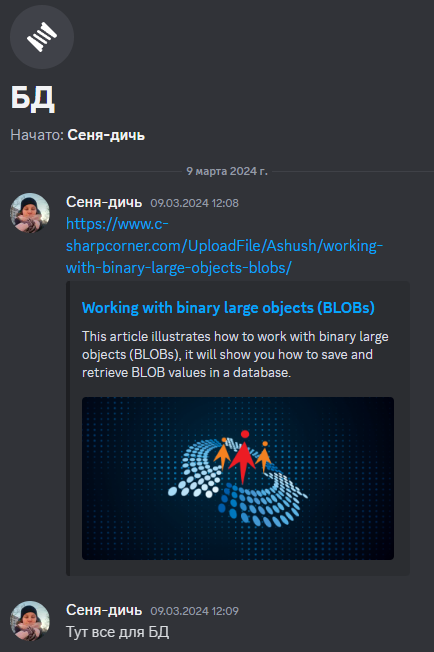


Рис.2. Ветка разработчика баз данных

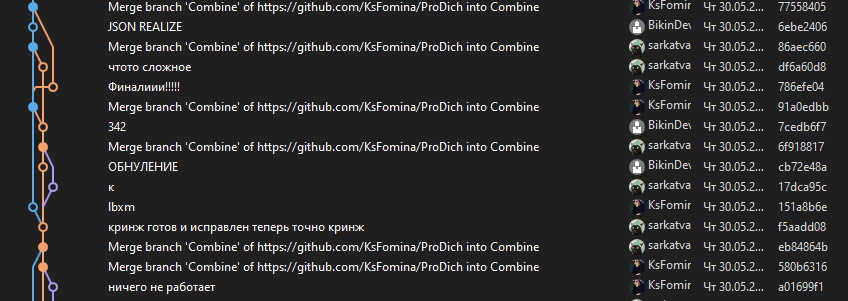


Рис.3. Действия на ветке

# Обзор инструментов

## Выбор платформы

В качестве среды разработки была выбрана Visual Studio, которая обладает рядом весомых преимуществ:

1. Интегрированная среда разработки (IDE): Visual Studio — это мощная и удобная в использовании IDE с богатым набором функций для разработки, отладки и тестирования кода.

2. Совместимость с C#: Visual Studio является родной средой для разработки на C#, что обеспечивает глубокую интеграцию с языком и .NET Framework, позволяя использовать все преимущества языка.

3. Продуктивность и эффективность: Visual Studio предлагает множество инструментов для повышения продуктивности, таких как IntelliSense (автодополнение кода), серверы сборки, профилировщики и инструменты для работы с базами данных.

4. Широкие возможности для командной работы: Visual Studio поддерживает интеграцию с системами контроля версий, такими как Git, и предлагает инструменты для совместной работы и непрерывной интеграции.

5. Расширяемость: IDE может быть расширена с помощью плагинов и расширений, что позволяет адаптировать среду под конкретные задачи и потребности проекта.

## 3.2. Фреймворка Prism

Prism предназначено для разработчиков WPF приложений с многочисленными экранами, расширенным взаимодействием с пользователем, визуализацией данных, и со сложным взаимодействие представлений и бизнес логики. Эти приложения обычно взаимодействуют с многочисленными back-end системами и службами и, используя многоуровневую архитектуру, могут быть физически развернуты на нескольких уровнях.

Основная идея Prism заключается в разделении приложения на отдельные модули, которые могут быть разработаны и поддерживаться независимо друг от друга. Каждый модуль имеет свою функциональность и может быть интегрирован в основное приложение. Это позволяет создавать гибкие и масштабируемые приложения, которые легко изменяются и расширяются.

Prism предоставляет различные инструменты и функции для упрощения разработки. Некоторые из них включают модель просмотра (MVVM), инверсию управления (IoC), навигацию, внедрение зависимостей и обработку событий.

## MVVM

Паттерн MVVM (Model-View-ViewModel) позволяет отделить логику приложения от визуальной части (представления). Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения. MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View) (рис.4).

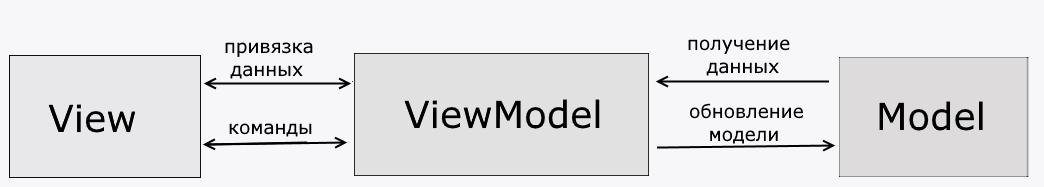


Рис.4. Паттерн MVVM

**Model** описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления.

Нередко модель реализует интерфейсы INotifyPropertyChanged или INotifyCollectionChanged, которые позволяют уведомлять систему об изменениях свойств модели. Благодаря этому облегчается привязка к представлению, хотя опять же прямое взаимодействие между моделью и представлением отсутствует.

**View** или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление — это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.

**ViewModel** или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны.

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать [1].

Связь между этими тремя составными частями осуществляется с помощью двух ключевых механизмов:

• Связывание данных (Data Binding): Механизм связывания данных позволяет установить связь между свойствами модели представления и элементами интерфейса представления. Таким образом, изменения в модели представления автоматически отражаются в представлении, и наоборот.

• Команды (Commands): Команды позволяют модели представления определить действия, которые могут быть выполнены в ответ на пользовательские действия. Например, кнопка может быть привязана к команде в модели представления, и при нажатии кнопки будет выполнена соответствующая команда. Команды позволяют централизованно управлять логикой приложения и обрабатывать пользовательские действия.

Prism предоставляет базовые классы и инфраструктуру для реализации MVVM:

• BaseBindableBase — это базовый класс в Prism, который реализует интерфейс INotifyPropertyChanged. Он предоставляет функциональность для уведомления представления об изменении свойств модели представления. При изменении значения свойства, модель представления автоматически уведомляет представление о необходимости обновиться.

• DelegateCommand — это класс, который реализует интерфейс ICommand и предоставляет механизм для определения команд, которые могут быть выполнены из представления. DelegateCommand позволяет определить методы, которые будут вызываться при выполнении команды, а также определить условия, при которых команда может быть выполнена (CanExecute) [2].

• EventAggregator — это класс, предоставляемый Prism, который служит для обмена сообщениями (событиями) между различными компонентами приложения. Он предоставляет механизм публикации и подписки на события, что позволяет модулям и компонентам взаимодействовать между собой без явной привязки. EventAggregator помогает реализовать слабую связанность между различными частями приложения [2].

# Разработка приложения

В рамках разработки WPF приложения мной были организован репозиторий git, разработаны база данных и класс для работы с ней, а также объединены все части, разработанные другими участниками команды.

4.1.Разработка диаграммы классов и диаграммы пакетов с учетом инверсии управления (IoC)

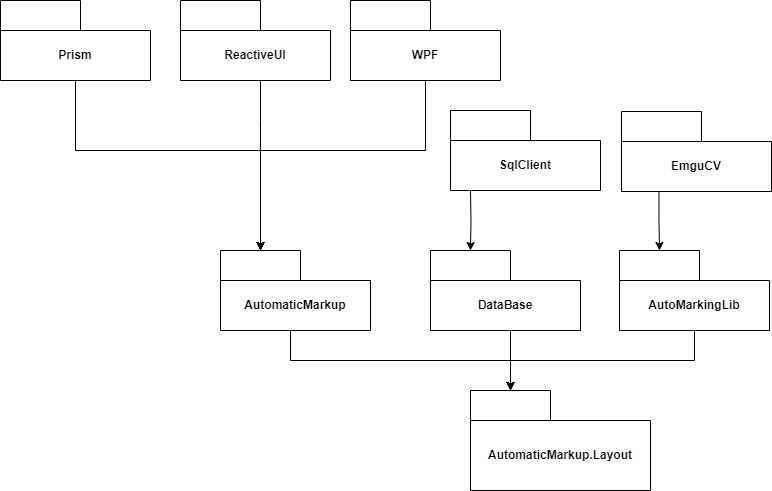


Рис.5. Диаграмма пакетов

На этой диаграмме отражены созданные пакеты и сторонние пакеты, которые использовались при написании программы. Пакеты Prism, ReactiveUI, WPF, SqlClient, EmguCV подключаются к другим пакетам, котФорые были разработаны участниками команды.

Пакет AutomaticMarkup – это оболочка, которая содержит основные инструкции запуска приложения. Пакет AutomaticMarkingLib содержит логику разметки. Пакет DataBase содержит логику работы с базой данных. И главный пакет, в котором расписана разметка и взаимодействие с другими пакетами AutomaticMarkup.Layout.

# Проектирование базы данных

# Заключение

# Список литературы

1. Паттерн MVVM. Электронный ресурс: https://metanit.com (дата обращения 10.05.2024)
2. Ключевые механизмы. Электронный ресурс https://habr.com (дата обращения 15.5.2024)

# Приложение 1. Техническое задание

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

(наименование структурного подразделения)

Кафедра математического и программного обеспечения ЭВМ

(наименование кафедры)

С# - программирование

(наименование дисциплины в соответствии с учебным планом)

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ,

д.т.н., профессор Ершов Е.В.

«\_\_\_» апреля 2023 г.

Разработка программного обеспечения на языке C#

Техническое задание на курсовую работу

Листов \_\_

Руководитель: Шаханов Н.И.

Исполнитель: студент гр. 1ПИб-01-01оп-21

Фомина Ксения Сергеевна

2024 год

Введение

Курсовая работа посвящена командной разработке программы для автоматической на языке С#, которая будет иметь возможность обрабатывать входные данные (изображение и маска) и создавать по ним разметку, сохранять результаты и поддерживать историю.

Данная документация выдвигает требования к разработке и оформлению сопутствующей документации.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине «С# - программирование», выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 14 марта 2024 года.

Наименование темы разработки: «Разработка программного обеспечения на языке C#»

2. Назначение разработки

Основная задача курсовой работы: освоить на практике материал, полученный в ходе изучения дисциплины «С# - программирование».

3. Требования к программе

3.1. Требования к функциональным характеристикам

Разработать приложение для авторазметки. Приложение должно выделять шарообразные камни угля на сгенерированном изображении, на основе его маски. Помимо этого, приложение должно поддерживать историю разметки, которая будет хранить дату и время разметки, у пользователя должна быть возможность восстановить файлы разметки с помощью истории. В ходе разработки учесть следующие требования:

* В качестве базы спроектировать WPF-приложение с использованием паттерна MVVM и/или использованием Фреймворка Prism.
* Придерживаться принципам SOLID2 и DRY3.
* Избегать наследования там, где его можно заменить ассоциацией. Особое внимание уделяйте классам «Моделям» и классам «Сервисам».
* Избегать написание кода в code-behind4.
* Выделить стили и темы оформления элементов пользовательского интерфейса в отдельные словари ресурсов.
* Реализовать файловый ввод/вывод данных.
* Предусмотреть обработку различных исключительных ситуаций.
* Работа всех функций должна быть проверена и результаты проверки оформлены протоколом тестирования.

3.2. Требования к надёжности

В местах, где могут возникнуть неожиданные ошибки предусмотреть обработку исключений.

3.3. Условия эксплуатации

Программу следует запускать на компьютере в закрытом помещении с обогревом и (или) охлаждением при следующих условиях окружающей среды:

• Уровень атмосферного давления: от 70 кПа до 106 кПа;

• Предел абсолютной влажности воздуха должен быть равен 25 гр/м;

• Температура окружающего воздуха от +16°C до +28°C;

• Запыленность воздуха не более 0,75 мг/м³;

• Знание основ работы в операционной системе Windows.

3.4. Требования к составу и параметрам технических средств

Для нормального функционирования программного средства минимальный состав и параметры технических средств должны соответствовать нижеследующему:

• Процессор с тактовой частотой не менее 2000 MHz, частота 2,3 Ghz;

• Оперативная память 5Gb и выше;

• Архитектура с разрядностью 32 бит или 64 бит;

• Наличие компьютерной мыши, клавиатуры, монитора (для персонального компьютера).

3.5. Требования к информационной и программной совместимости

Минимальные требования для информационной и программной совместимости:

• Операционная система (Windows 7 и выше);

• Наличие компьютерного приложения Visual Studio 2019, если установлена более ранняя версия Visual Studio, необходимо установить дополнительно набор инструментов платформы v142 и пакет SDK для Windows версией 10.0.

3.6. Требования к маркировке и упаковке

Распространение через средства коммуникации (для некоммерческого использования).

3.7. Требования к транспортированию и хранению

Файлы, необходимые для корректной работы, необходимо записать на CD-диск.

3.8. Специальные требования

Отсутствуют.

4. Требования к программной документации

4.1. Содержание расчётно-пояснительной записки:

* Оглавление;
* введение;
* Описание предметной области;
* Организация работы в команде;
* Проектирование ПО;
* тестирование;
* заключение;
* список литературы;
* техническое задание;
* руководство пользователя;
* текст программы.

4.2. Требования к оформлению

Требования к оформлению, установленные ГОСТ, должны быть выполнены на протяжении всей работы без каких-либо изменений (в табл. П1.1).

Таблица П1.1

Требования к оформлению

|  |  |
| --- | --- |
| Документ | Печать на отдельных листах формата А4 (20х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б. |
| Страницы | Ориентация — книжная; отдельные страницы, при необходимости, альбомная. Поля: верхнее, нижнее — по 2 см, левое — 3 см, правое — 2 см. |
| Абзацы | Межстрочный интервал — 1,5, перед и после абзаца — 0. |
| Шрифты | Кегль — 14. В таблицах шрифт 12. Шрифт листинга — 8 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: «Рис.Х. Название В» приложениях: «Рис.П.3. Название» |
| Таблицы | Подписывается: над таблицей, выравнивание по правому: «Таблица Х». В следующей строке по центру Название Надписи в «шапке» (имена столбцов, полей) — по центру. В теле таблицы (записи) текстовые значения — выравнены по левому краю, числа, даты — по правому. |

5. Стадии и этапы разработки

Стадии и этапы разработки представлены в таблице П1.2.

Таблица П1.2

Этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Определение темы для курсовой работы | 20.03.2023 | Утверждена тема разработки | Выполнено |
| Оформление техническое задания | 10.05.2023 | Выполненное техническое задание | Выполнено |
| Логическое проектирование | 30.04.2023 | Создан алгоритм решения задания | Выполнено |
| Физическое проектирование | 05.05.2023 | Создана программа | Выполнено |
| Тестирование | 5.06.2023 | Проверка правильности расчетов построения | Выполнено |

Продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| Написание РПЗ | 8.06.2023 | Создано РПЗ по курсовой работе | Выполнено |

6. Порядок контроля и приёмки

Порядок контроля и приёма представлены в таблице П1.3.

Таблица П1.3

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| Утверждение технического задания | 19.05.2023 | Документ «Техническое задание» проверен |  |

Продолжение таблицы П.1.3

|  |  |  |  |
| --- | --- | --- | --- |
| Демонстрация работы первой версии программы | 30.05.2023 | Работа программы проверена |  |
| Поиск ошибок в программе | 30.05.2023 | Все найденные ошибки исправлены |  |
| Демонстрация окончательной версии программы | 5.06.2023 | Работа программы проверена |  |
| Защита курсовой работы | 10.06.2023 | Курсовая работа защищена |  |

# Приложение 2. Текст программы

# Приложение 3. Руководство пользователя