бюджетное профессиональное образовательное учреждение Вологодской области
«Череповецкий лесомеханический техникум им. В.П. Чкалова»

Специальность **09.02.07** «Информационные системы и программирование»

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

ПП по ПМ.03 РЕВЬЮИРОВАНИЕ ПРОГРАММНЫХ МОДУЛЕЙ

Выполнил студент 3 курса группы ИС-____

_____

подпись _____

место практики

_____

наименование юридического лица, ФИО ИП

Период прохождения:

с «___» _____ 2025 г.

по «___» _____ 2025 г.

Руководитель практики от предприятия
должность_____

_____

подпись_____

Руководитель практики от техникума: Материкова А.А.

_____

Оценка:_____

«___» _____2025 года

МП

г. Череповец

2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Название предприятия: ООО «Малленом Системс»

Сроки прохождения практики: с 17.11.25 по 30.11.25

Цель: целью практики является освоение процессов ревьюирования программных модулей, включая анализ качества кода, оценку архитектуры, измерение характеристик компонентов, построение UML-моделей и использование специализированных инструментов для исследования программного обеспечения.

Задачи:

1. Изучить теоретические основы ревьюирования программного кода, включая техники анализа, критерии качества и принципы построения архитектуры.

2. Провести обратное проектирование программного продукта, построив UML-диаграммы.

3. Выполнить ревьюирование программных модулей.

4. Измерить характеристики компонентов программного продукта.

5. Применить специализированные инструменты анализа.

6. Выполнить сравнительный анализ средств разработки, используемых при создании программного продукта.

1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРЕДПРИЯТИЯ

«Малленом Системс» — российская IT-компания, специализируется на компьютерном зрении, промышленной видеоаналитике и решениях на базе машинного обучения / нейронных сетей.

Зарегистрирована 28.02.2011

Генеральный директор — Анна Э. Живиця.

1.1 Организационная структура предприятия

1. Генеральный директор (CEO) — стратегическое управление, внешние связи, крупные клиенты/партнёры.

2. Технический директор (CTO) — отвечает за R&D, архитектуру продуктов, научные разработки.

3. Раздел «Разработка и R&D»:

– Отдел разработки ПО (backend, frontend, встраиваемое ПО).

– Отдел машинного обучения / Data Science (обучение и поддержка нейросетей, подготовка данных).

– Отдел встраиваемых систем и аппаратных интеграций (smart-камера, SDK).

4. Отдел качества (QA, тестирование видеоаналитики, валидация моделей).

5. Отдел внедрения и проекта (Project Management, System Integrators, монтаж/пусконаладка у заказчика).

6. Отдел DevOps / SRE — CI/CD, облачная инфраструктура, поставка ПО на объекты.

7. Отдел продаж и маркетинга — коммерция, работа с партнёрами, поддержка клиентов.

8. Отдел техподдержки / сервисная поддержка (поддержка инсталлированных решений).

9. Отдел по безопасности и промышленной интеграции (для отраслевых решений: ПБ, интеграция с системами предприятия).

10. Административно-хозяйственные подразделения: HR, финансы/бухгалтерия, юристы, логистика.

1.2 Внутренний распорядок работы предприятия, охрана труда ИТ-специалистов

Внутренний распорядок работы

Начало и окончание рабочего дня фиксируется через систему учёта рабочего времени (табель/электронный пропуск).

– Личные встречи/статусы проектов: ежедневные стендапы (Scrum), еженедельные планёрки по проектам, ежемесячные отчёты по KPI.

– Политика доступа к коду/данным: разграничение прав в репозиториях (git), двухфакторная аутентификация, политика работы с конфиденциальными данными заказчиков.

– Обучение и развитие: внутренние курсы, хакатоны, корпоративные мероприятия.

– Охрана труда и безопасность для ИТ-специалистов

– Рабочие места: эргономичные кресла, регулируемые столы, монитор(ы) с защитой от мерцания. Регулярная проверка электрооборудования.

– Санитарно-гигиенические нормы: освещение, температурный режим, вентиляция, нормы по шуму.

– Медосмотры: периодические медосмотры в соответствии с трудовым законодательством (если предусмотрено локально для конкретных должностей).

– Профилактика зрительного напряжения: регламентированные перерывы от экранной работы, рекомендации по настройке дисплеев (яркость/контраст), возможность использования специальной оптики.

– Электробезопасность: инструкции по работе с оборудованием, запрет на несанкционированное вмешательство в серверное/сетевое оборудование.

– Инструктажи и обучение: вводный инструктаж по охране труда, противопожарный инструктаж, обучение по информационной безопасности (обращение с персональными данными, соблюдение политики конфиденциальности).

1.3 Должностные инструкции ИТ-специалистов предприятия

1) Инженер-программист (Software Engineer)

Цель должности: разработка и поддержка компонентов серверной/встроенной части ПО.

Обязанности:

– Разработка и сопровождение модулей ПО (backend / SDK)

– Участие в проектировании архитектуры

– Написание unit/integration тестов, участие в code review

– Сопровождение выпуска релизов, устранение багов

– Работа с системой контроля версий и CI/CD

2) Инженер по машинному обучению (ML Engineer / Data Scientist)

Цель: разработка, обучение и внедрение моделей компьютерного зрения.

Обязанности:

– Подготовка датасетов, аннотирование; обучение нейросетей, оптимизация моделей

– Валидация точности/robustness в реальных условиях (ночь, засорённые номера и т.д.)

– Пакетирование моделей для встраивания в SDK/камера

– Совместная работа с инженерами по продукту и QA

3) DevOps / SRE

Цель: поддержка инфраструктуры, CI/CD, развёртываний у заказчика и в облаках.

Обязанности:

– Настройка CI/CD, контейнеризация (Docker/Kubernetes), мониторинг

– Автоматизация сборок и деплоёв; сопровождение серверной инфраструктуры

4) Инженер по внедрению / системный интегратор

Цель: развёртывание систем на объектах заказчика, интеграция с локальными системами предприятия.

Обязанности:

– Монтаж/настройка камер и серверов, интеграция с доступом/складской/транспортной системой

– Тестирование на рабочем объекте, обучение персонала заказчика

– Оформление актов приёмки и сопровождение сервисных заявок

5) QA инженер (тестирование)

Обязанности: разработка тест-планов, автоматизация тестов, проверка точности алгоритмов на реальных видеоданных, регрессионное тестирование.

6) IT-поддержка / сервисная поддержка

Обязанности: первичная техническая поддержка клиентов, обработка тикетов, базовая диагностика и эскалация сложных проблем в разработку/внедрение.

2 РЕВЬЮИРОВАНИЕ ПРОГРАММНЫХ ПРОДУКТОВ

2.1 Ревьюирование программного кода в соответствии с технической документацией

Ревьюирование программного кода (code review) — это процесс анализа созданного программистом исходного кода с целью проверки его качества, соответствия требованиям технического задания, соблюдения стандартов разработки и выявления ошибок ещё до этапа тестирования. Этот процесс является важнейшим элементом обеспечения качества программного обеспечения и служит инструментом повышения надёжности, безопасности и поддерживаемости программного продукта.

Основными задачами ревьюирования являются:

– Проверка соответствия кода технической документации и архитектурным требованиям

– Выявление логических ошибок и недочётов в алгоритмах

– Оценка качества структуры кода: читаемости, модульности, повторяемости и соблюдения принципов проектирования (SOLID, DRY, KISS)

– Анализ корректности обработки исключений и нештатных ситуаций

– Проверка соответствия выбранным стилевым стандартам языка программирования

Ревью может проводиться вручную (чтение и анализ исходных файлов) или с использованием автоматизированных инструментов статического анализа. Комбинированный подход считается наиболее эффективным, так как позволяет обнаружить как синтаксические, так и архитектурные проблемы.

2.2 Измерение характеристик компонентов программного продукта

После создания программных модулей проводится измерение их характеристик. Это необходимо для оценки эффективности работы системы и определения возможных точек оптимизации. Оценка характеристик включает:

1. Скоростные показатели

Определяются временные затраты выполнения ключевых операций.

Например:

– Время загрузки данных
– Время выполнения вычислительных задач
– Скорость обработки изображений
– Производительность отдельных функций

2. Ресурсные показатели

Характеризуют потребление:

– Оперативной памяти
– Дискового пространства
– Использования процессора

3. Структурные показатели

Включают:

– Количество строк кода
– Уровень связности модулей
– Степень модульности
– Количество внешних зависимостей

2.3 Исследование созданного программного кода с использованием специализированных программных средств

Для анализа и проверки качества программного обеспечения используются специализированные инструменты, которые позволяют автоматизировать различные этапы ревьюирования.

Основные группы инструментов:

1. Инструменты статического анализа

Осуществляют автоматическую проверку кода без выполнения программы. Они способны обнаруживать синтаксические проблемы, потенциальные ошибки, уязвимости и несоответствия стилевым стандартам.

Примеры:

– Pylint, flake8, mypy, bandit (Python)

– SonarQube — комплексная система анализа качества

2. Инструменты динамического анализа

Проверяют программу в процессе выполнения и анализируют:

– Утечки памяти

– Исключения

– Тормоза производительности

– Некорректные состояния объектов

3. Инструменты визуального проектирования

Используются для построения UML-диаграмм, анализа архитектуры и документирования.

Примеры:

– diagrams.net (draw.io)

– StarUML

– Visual Paradigm

2.4 Сравнительный анализ программных продуктов и средств разработки

Сравнительный анализ проводится для выбора оптимальных технологий, библиотек и подходов разработки, соответствующих задачам создаваемого программного продукта.

Анализ может включать сравнение:

1. Языков программирования

Например:

– Python — высокий уровень абстракции, быстрая разработка, богатая экосистема

– C/C++ — высокая производительность, низкоуровневый контроль

– JavaScript — универсальность, Web-ориентированная разработка

Python выбирают для быстрого прототипирования, разработки десктоп-приложений, научных вычислений и работы с изображениями.

3. Инструментов разработки

Сравниваются возможности IDE:

– PyCharm — мощная аналитика, удобство для больших проектов

– VS Code — лёгкий, быстрый, с отличной системой расширений

4. Характеристик готовых решений

Сравнение аналогичных программных продуктов позволяет определить:

– Функциональную полноту

– Стабильность

– Удобство интерфейса

– Производительность

– Гибкость расширения

3 ВЫПОЛНЯЕМЫЕ ЗАДАНИЯ

1. Разработка ПО

1.1 Разработка программных модулей

В ходе выполнения работы был разработан программный продукт, представляющий собой настольное приложение для загрузки и обработки растровых изображений. Основная цель разработки — реализовать набор базовых функций редактирования (обрезка, изменение размера, преобразование изображения в оттенки серого, просмотр параметров), а также обеспечить удобный графический интерфейс пользователя.

Проект был выполнен в модульной архитектуре и состоит из трёх основных компонентов:

1. Модуль обработки изображений — image_processor.py

2. Модуль графического интерфейса — ui.py

3. Точка входа и запуск приложения — main.py

Программный код разрабатываемых модулей представлен в приложении А.

1.2 Описание архитектуры и подхода к разработке

В процессе разработки использовался принцип разделения ответственности.

Бизнес-логика полностью вынесена в отдельный класс ImageProcessor, в котором реализованы:

– Загрузка изображения

– Выделение области и обрезка

– Преобразование в градации серого

– Изменение размера

– Получение информации о файле

– Сброс к исходной версии

Пользовательский интерфейс (ui) не содержит логики обработки и отвечает только за отображение и взаимодействие с пользователем. UI вызывает методы ImageProcessor, получает результат и выводит на экран.

1.3 Принцип работы и взаимодействие модулей

1.3.1 Пользовательский интерфейс (ui.py)

UI создан на базе библиотеки tkinter и обеспечивает:

– Загрузку изображения через диалоговое окно

– Загрузку изображения методом drag & drop

– Отображение исходного изображения

– Визуальное выделение области мышью

– Предпросмотр обработанного изображения

– Отображение параметров изображения (размер, формат, вес)

– Кнопки управления: «Загрузить фото», «Обрезать фото», «Применить чёрно-белый фильтр», «Сбросить изменения»

1.3.2 Модуль обработки (image_processor,py)

ImageProcessor обеспечивает выполнение всех операций над изображениями. Реализация базируется на библиотеке Pillow.

Основные методы:

– load_image(path) – загрузка изображения

– crop(x1, y1, x2, y2) – обрезка изображения

– to_grayscale() – применение чёрно-белого фильтра

– resize(width, height) – изменение размера изображения

– get_info() – получение метаданных

– reset_to_original() – возврат исходного изображениями

1.3.3 Запуск (main.py)

Файл запуска программы содержит:

– Импорт UI

– Создание экземпляра интерфейса и запуск основного цикла

1.4 Комментарии к программной реализации

В процессе разработки были добавлены структурные и поясняющие комментарии:

– О назначении каждого модуля

– Документирование всех методов внутри ImageProcessor

– Разделение зон логики внутри ui.py

– Комментарии к обработчикам событий мыши

– Пояснения к системе выделения области для обрезки

– Комментарии к обработке изображения и предпросмотру

Эти комментарии были добавлены для повышения читаемости кода и упрощения его ревьюирования.

1.5 Инструкция по применению программы

Перед запуском убедитесь, что у вас установлены следующие компоненты: Python 3.10 или выше, tkinter, tkinterdnd2, pillow

Для запуска программы запустите файл main.py, откроется окно программы.

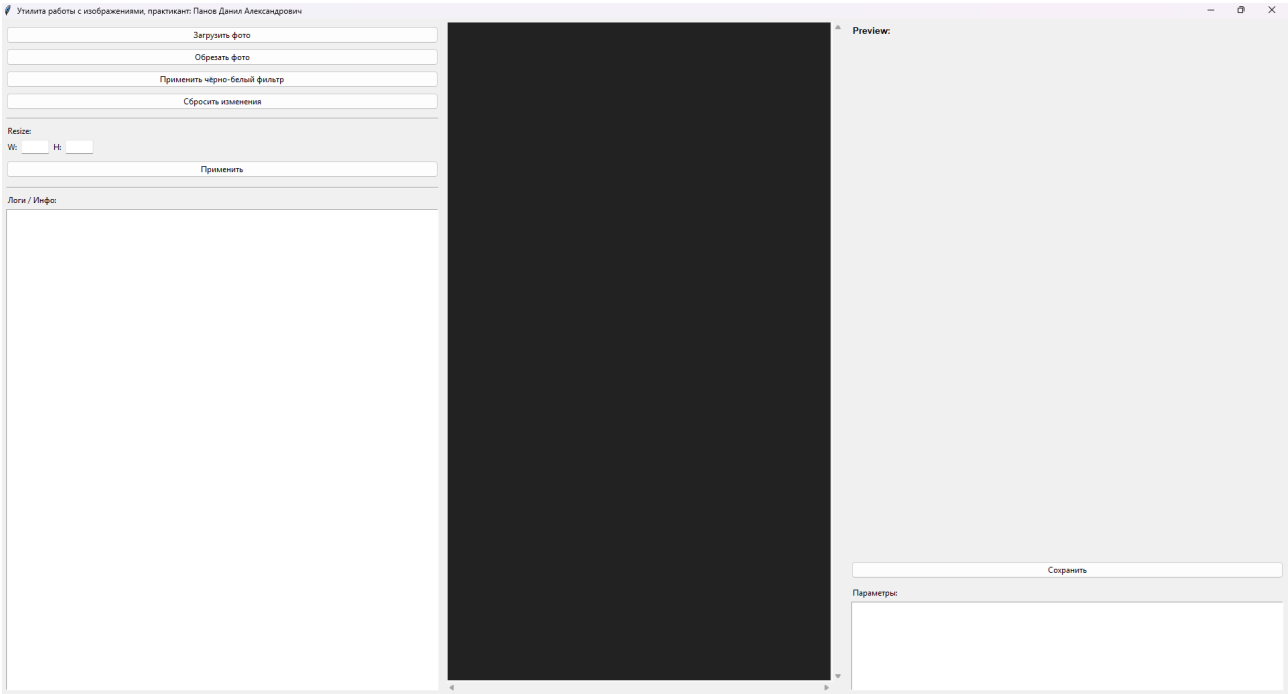

Рисунок 1 — Интерфейс программы

Основные функции:

1. Загрузка изображения: нажать на кнопку «Загрузить фото» и выбрать нужное изображение или перетащить нужное фото на Канвас.

Поддерживаемые форматы изображений: PNG, JPG, JPEG, BMP

2. Обрезка изображения:

– Удерживая левую кнопку мыши, выделить на изображении область

– Нажать кнопку «Обрезать»

– Результат отобразится в окне предпросмотра

3. Применение чёрно-белого фильтра

– Нажать кнопку «Применить чёрно-белый фильтр»

– Цветная картинка станет монохромной

4. Изменение размера

– В поле ширины и высоты указать новые размеры

– Нажать кнопку «Применить»

5. Просмотр параметров изображения

С правой стороны отображаются:

– Формат файла

– Размер изображения в пикселях

– Цветовая модель

– Размер файла в байтах

6. Сохранение результата

– Нажать кнопку «Сохранить»

– Выбор формата осуществляется автоматически по расширению файла

2. Ревьюирование программных модулей

2.1 Обратное проектирование

2.1.1 Компонентная диаграмма

Цель: показать модули и внешние зависимости.

Элементы:

– UI (ui.py) — графический интерфейс (tkinter, tkinterdnd2)

– ImageProcessor (image_processor.py) — бизнес-логика обработки изображений

– Main (main.py) — точка входа (создаёт UI)

– Внешние библиотеки: Pillow, tkinterdnd2, tkinter

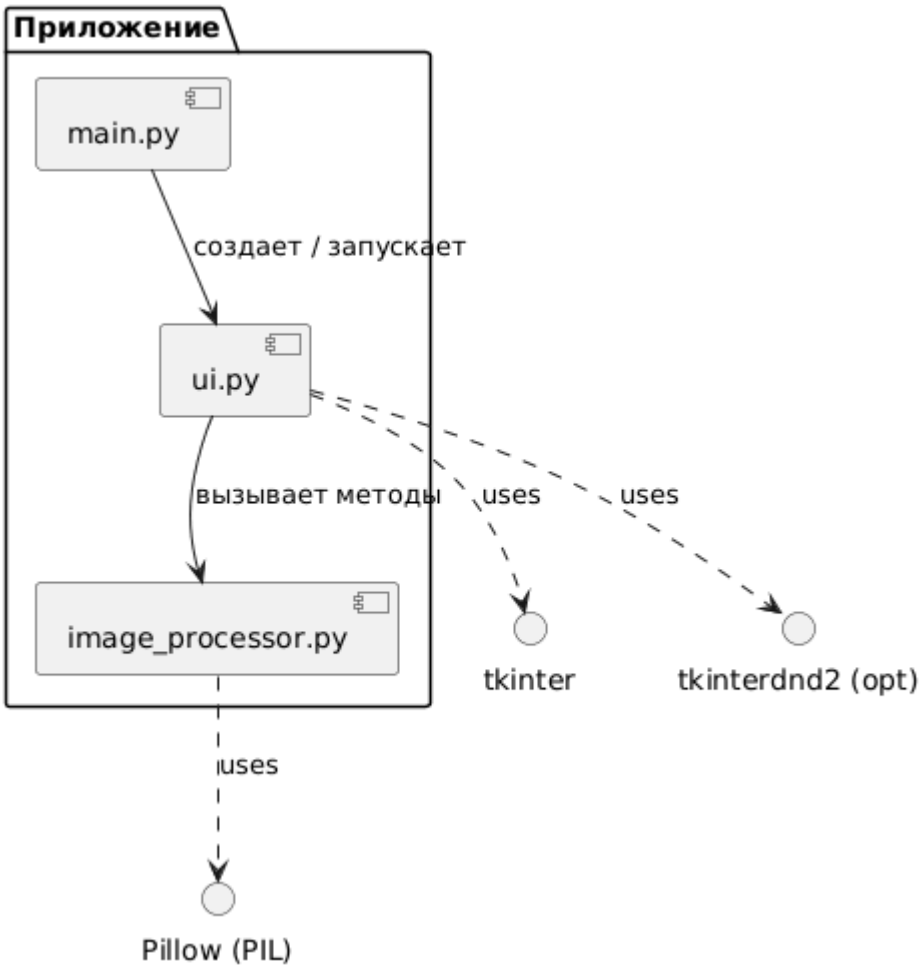

Рисунок 2 — Диаграмма компонентов

2.1.2 Диаграмма прецедентов использования

Акторы:

– Пользователь (User)

Прецеденты:

– Загрузить изображение (кнопка / DnD)

– Выделить область

– Обрезать изображение

– Применить фильтр (Ч/Б)

– Изменить размер

– Сохранить результат

– Просмотреть параметры файла

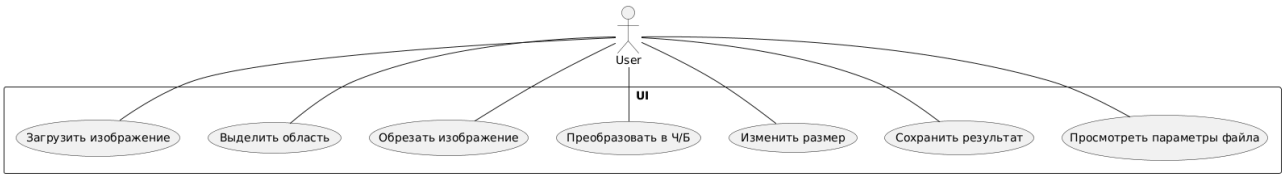

Рисунок 3 — Диаграмма прецедентов использования

2.1.3 Диаграмма последовательностей

Сценарий: Пользователь выделяет область, нажимает «Обрезать»

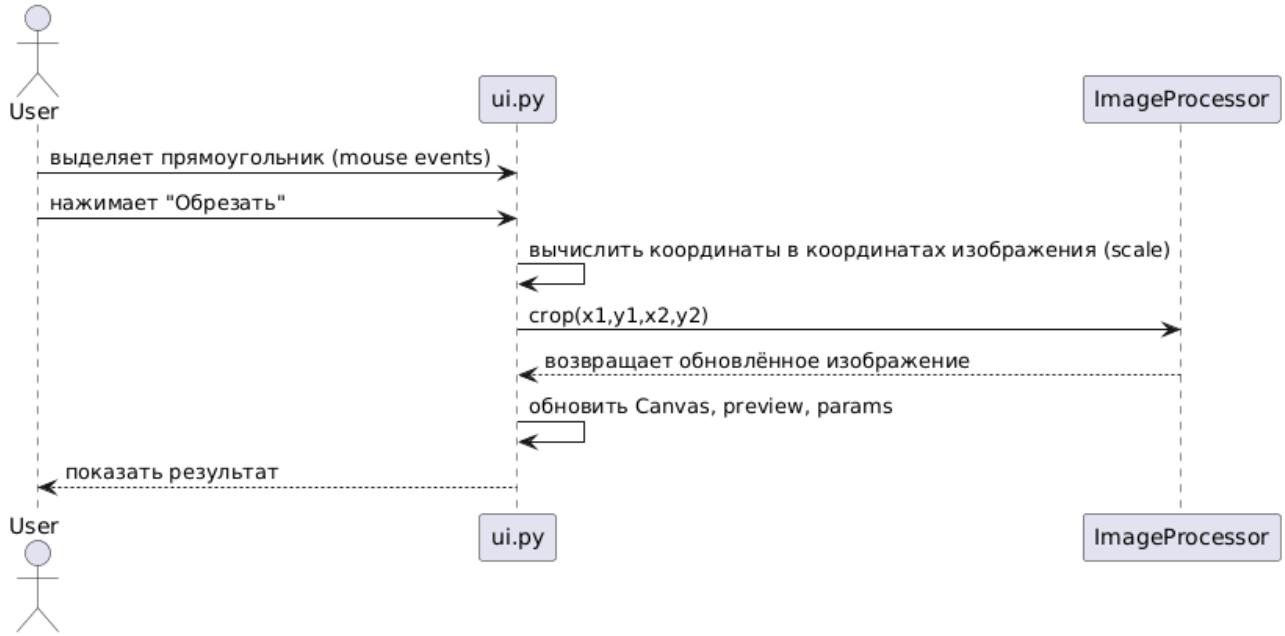

Рисунок 4 — Диаграмма последовательностей

2.1.4 Диаграмма деятельности

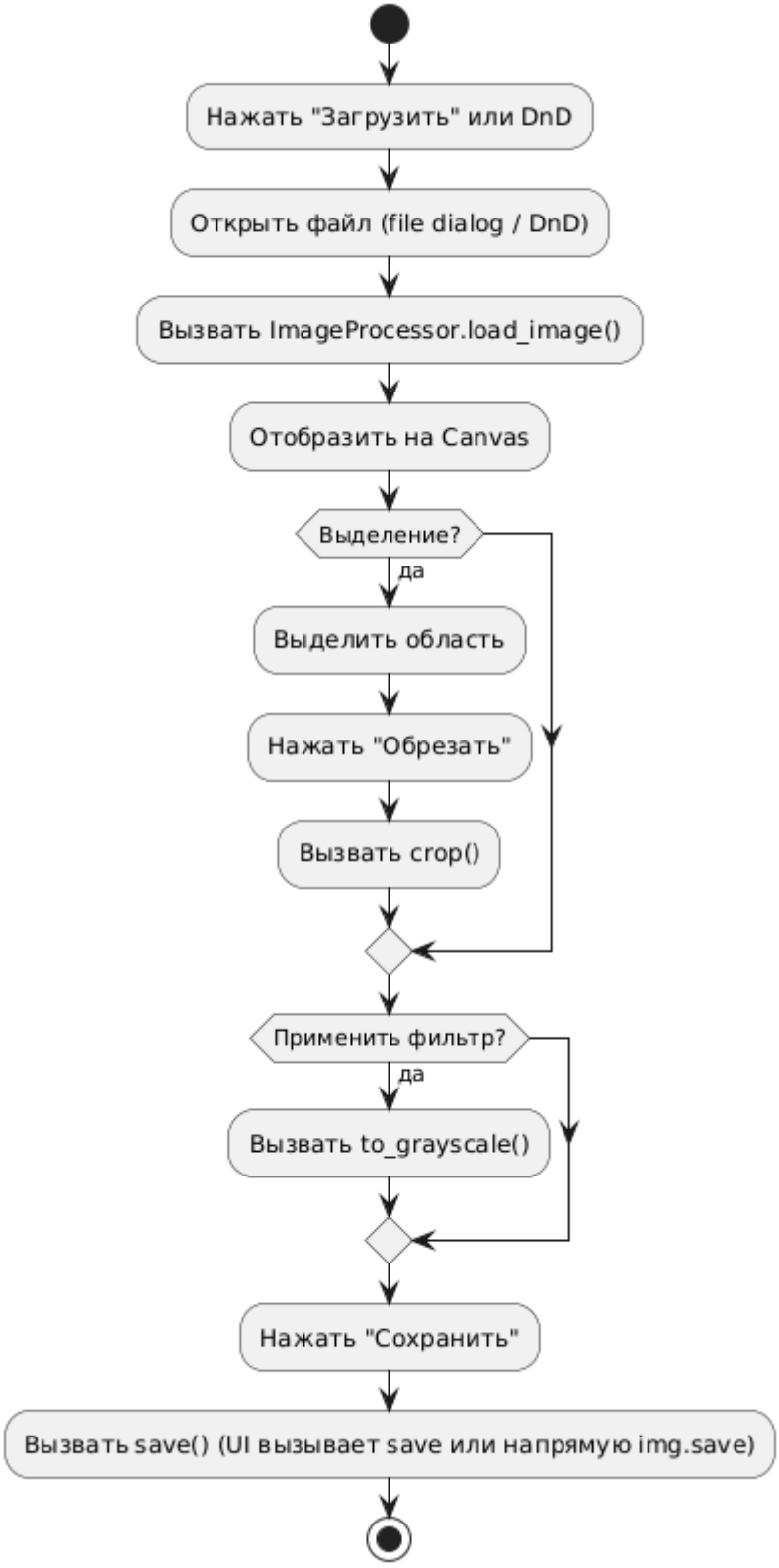

Рисунок 5 — Диаграмма деятельности

2.2 Производительность и размеры

Были проведены измерения на трёх тестовых изображениях: маленькое (800x600), среднее (2000x1500), большое (4000x3000).

Файловые метрики проекта:

– image_processor.py — 2 704 байта, 79 строк

– ui.py — 13 085 байтов, 321 строка

– main.py — 119 байтов, 8 строк

Результаты измерений:

2.2.1 Маленькое изображение (800x600)

Таблица 1 — Скоростные показатели программы на маленьком изображении

Операция	Среднее время (сек)	Количество прогонов
Загрузка	0.00082	5
Обрезка	0.00364	20
Градации серого	0.00416	20
Получение информации	0.00016	100
Сброс к оригиналу	0.07059	20

Вывод: на маленьких изображениях все операции проходят практически мгновенно (меньше 5 миллисекунд), кроме сброса состояния.

2.2.2 Среднее изображение (2000x1500)

Таблица 2 — Скоростные показатели программы на среднем изображении

Операция	Среднее время (сек)	Количество прогонов
Загрузка	0.00107	5
Обрезка	0.00980	20
Градации серого	0.02156	20
Получение информации	0.00016	100
Сброс к оригиналу	0.05612	20

Вывод: средние изображения обрабатываются за десятки миллисекунд, интерфейс остаётся отзывчивым.

2.2.3 Большое изображение (4000x3000)

Таблица 3 — Скоростные показатели программы на большом изображении

Операция	Среднее время (сек)	Количество прогонов
Загрузка	0.00309	5
Обрезка	0.03660	20
Градации серого	0.08110	20
Получение информации	0.00016	100
Сброс к оригиналу	0.05645	20

Вывод: даже для больших изображений операции не превышают 80–100 миллисекунд, что является отличным результатом для настольного приложения на Python.

2.3 Анализ средств разработки

Разработка программного продукта велась на языке Python с использованием интегрированной среды разработки Visual Studio Code (VS Code). Данный инструмент был выбран благодаря своей гибкости, широкому набору расширений, поддержке систем контроля версий и удобству отладки.

2.3.1 Характеристика языка программирования Python

Python был выбран как основной язык разработки по следующим причинам:

1. Высокий уровень абстракции — упрощает работу с файлами, изображениями и пользовательским интерфейсом.

2. Богатая экосистема библиотек:

– Pillow — для обработки изображений

– tkinter — для создания графического интерфейса

– tkinterdnd2 — для реализации Drag & Drop

3. Читаемость кода — важный фактор при ревьюировании и сопровождении.

4. Кроссплатформенность — программа работает на Windows, Linux и macOS.

5. Удобство профилирования и отладки — встроенные инструменты позволяют легко проводить замеры производительности.

2.3.2 Выбор и обоснование среды разработки Visual Studio Code (VS Code)

В ходе разработки использовалась среда Visual Studio Code, предоставляющая широкие возможности для Python-разработки.

Преимущества VS Code для конкретного проекта:

1. Поддержка расширений

Были использованы следующие расширения:

– Python (Microsoft) — подсветка синтаксиса, анализатор кода, автодополнение, выявление ошибок

– Pylance — быстрый и умный движок анализа типов

– Code Runner — запуск отдельных файлов без лишней настройки

2. Эффективная система отладки

VS Code позволяет:

– Ставить точки останова

– Просматривать значения переменных в реальном времени

– Осуществлять пошаговое выполнение программы

– Отслеживать исключения

– Анализировать стек вызовов

2.3.3 Используемые технологические средства

1. Библиотека Pillow

Используется для:

– Загрузки изображений

– Изменения размера

– Обрезки

– Преобразования в оттенки серого

– Получения информации о файле

Преимущества:

– Активная поддержка

– Высокое быстродействие

– Широкая совместимость с форматами

2. Tkinter

Стандартная библиотека Python для создания GUI-приложений.

Использовалась для:

– Построения основного окна

– Размещения кнопок, полей ввода, панелей

– Отрисовки изображения на Канвасе

– Работы с событиями мыши.

Преимущества:

– Идёт в стандартной поставке Python

– Простота разработки

– Минимальные системные требования

3. tkinterdnd2 (опционально)

Использована для реализации поддержки Drag & Drop.

Позволяет:

– Перетаскивать изображения прямо в окно приложения

– Автоматически определять путь к файлу

2.3.4 Инструменты анализа и профилирования

При исследовании производительности применялись:

– time.perf_counter() — точный таймер для микросекундных замеров

ЗАКЛЮЧЕНИЕ

В ходе прохождения производственной практики были изучены и практически освоены методы и инструменты ревьюирования программных модулей. На основе предоставленного программного продукта была выполнена комплексная работа, включающая анализ исходного кода, оценку архитектуры, измерение характеристик модулей, построение UML-диаграмм и исследование программного обеспечения с использованием специализированных средств.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Технология Git [Электронный ресурс] – режим доступа: https://git-scm.com/book/ru/v2/Введение-Что-такое-Git%3F

2. Объектное моделирование UML [Электронный ресурс] – режим доступа: https://www.uml.org

3. Построение диаграмм UML [Электронный ресурс] – режим доступа: https://practicum.yandex.ru/blog/uml-diagrammy/

4. Пример измерения скорости используя Time [Электронный ресурс] – режим доступа: https://www.geeksforgeeks.org/python/how-to-check-the-executiontime-of-python-script/

5. Документация Visual Studio Code [Электронный ресурс] – режим доступа: https://code.visualstudio.com/docs

6. Руководство по ревью кода [Электронный ресурс] – режим доступа: https://www.atlassian.com/agile/code-reviews

7. Документация Python [Электронный ресурс] – режим доступа: https://docs.python.org/3/

8. Документация Pillow [Электронный ресурс] – режим доступа: https://pillow.readthedocs.io/

9. Официальное руководство по профилированию Python [Электронный ресурс] – режим доступа: https://docs.python.org/3/library/profile.html

10. Документация по Tkinter [Электронный ресурс] – режим доступа: https://docs.python.org/3/library/tkinter.html

ПРИЛОЖЕНИЯ

Приложение А.

```python
import os
from PIL import Image, ImageOps

class ImageProcessor:
    def __init__(self):
        self.path = None
        self.original = None
        self.current = None
        self._format = None

    def load_image(self, path: str):
        if not os.path.exists(path):
            raise FileNotFoundError(path)
        img = Image.open(path).convert("RGBA")
        self.path = path
        self.original = img.copy()
        self.current = img.copy()
        self._format = img.format or os.path.splitext(path)[1].lstrip('.').upper()
        return self.current

    def crop(self, x1: int, y1: int, x2: int, y2: int):
        if self.current is None:
            raise RuntimeError("No image loaded")
        w, h = self.current.size
        x1c = max(0, min(w, int(round(x1))))
        x2c = max(0, min(w, int(round(x2))))
        y1c = max(0, min(h, int(round(y1))))
        y2c = max(0, min(h, int(round(y2))))
        if x2c <= x1c or y2c <= y1c:
            raise ValueError("Invalid crop coordinates")
        self.current = self.current.crop((x1c, y1c, x2c, y2c))
        return self.current

    def to_grayscale(self):
        if self.current is None:
            raise RuntimeError("No image loaded")
        rgb = self.current.convert("RGB")
        gray = ImageOps.grayscale(rgb).convert("RGBA")
        self.current = gray
        return self.current

    def resize(self, width: int, height: int, keep_aspect: bool = False):
        if self.current is None:
            raise RuntimeError("No image loaded")
        w, h = self.current.size
        if keep_aspect:
            ratio = min(width / w, height / h)
            new_w = max(1, int(round(w * ratio)))
            new_h = max(1, int(round(h * ratio)))
        else:
            new_w = max(1, int(width))
            new_h = max(1, int(height))
        self.current = self.current.resize((new_w, new_h), Image.LANCZOS)
        return self.current

    def get_info(self) -> dict:
        if self.current is None:
            return {}
        w, h = self.current.size
        try:
            size = os.path.getsize(self.path) if self.path else None
        except:
            size = None
        return {
            "path": self.path,
            "format": self._format,
            "width": w,
            "height": h,
            "filesize_bytes": size,
            "mode": self.current.mode
        }

    def get_image(self):
        return self.current.copy() if self.current else None

    def reset_to_original(self):
        if self.original is not None:
            self.current = self.original.copy()
            return self.current
```

Рисунок 6 — Модуль обработки изображений

```python
import os
import tkinter as tk
from tkinter import ttk, filedialog, messagebox
from PIL import Image, ImageTk
from image_processor import ImageProcessor

# Основной класс интерфейса
class ImageApp:
    def __init__(self):
        # drag & drop
        try:
            from tkinterdnd2 import TkinterDnD, DND_FILES
            self._dnd_mod = TkinterDnD
            self._dnd_const = DND_FILES
            self.root = TkinterDnD.Tk()
            self._dnd_enabled = True
        except ImportError:
            self.root = tk.Tk()
            self._dnd_enabled = False

        self.root.title("Утилита работы с изображениями, практикант: Панов Данил Александрович")
        self.root.geometry("1100x700")
        self.root.minsize(900, 600)

        self.processor = ImageProcessor()

        self.sel_start = None
        self.sel_end = None
        self.sel_rect_id = None
        self.display_scale = 1.0

        self._build_ui()
        self._bind_events()

        if self._dnd_enabled:
            self.canvas.drop_target_register(self._dnd_const)
            self.canvas.dnd_bind('<<Drop>>', self._on_drop)

    def _build_ui(self):
        self.root.columnconfigure(1, weight=1)
        self.root.rowconfigure(0, weight=1)

        # Левая панель
        left = ttk.Frame(self.root, width=220, padding=(8,8))
        left.grid(row=0, column=0, sticky="ns")
        left.grid_propagate(False)

        ttk.Button(left, text="Загрузить фото", command=self.on_load).pack(fill="x", pady=4)
        ttk.Button(left, text="Обрезать фото", command=self.on_crop).pack(fill="x", pady=4)
        ttk.Button(left, text="Применить чёрно-белый фильтр", command=self.on_grayscale).pack(fill="x", pady=4)
        ttk.Button(left, text="Сбросить изменения", command=self.on_reset).pack(fill="x", pady=4)

        ttk.Separator(left, orient="horizontal").pack(fill="x", pady=8)
        ttk.Label(left, text="Resize:").pack(anchor="w")
        rf = ttk.Frame(left)
        rf.pack(fill="x", pady=4)
        ttk.Label(rf, text="W:").grid(row=0, column=0)
        self.entry_w = ttk.Entry(rf, width=6)
        self.entry_w.grid(row=0, column=1, padx=4)
        ttk.Label(rf, text="H:").grid(row=0, column=2)
        self.entry_h = ttk.Entry(rf, width=6)
        self.entry_h.grid(row=0, column=3, padx=4)
        self.keep_aspect = tk.BooleanVar(value=True)
        ttk.Button(left, text="Применить", command=self.on_resize).pack(fill="x", pady=6)

        ttk.Separator(left, orient="horizontal").pack(fill="x", pady=8)
        ttk.Label(left, text="Логи / Инфо:").pack(anchor="w")
        self.log_text = tk.Text(left, height=10, wrap="word")
        self.log_text.pack(fill="both", expand=True, pady=(4,0))

        # Канвас (область для выделения области обрезки)
        canvas_frame = ttk.Frame(self.root, padding=(4,4))
        canvas_frame.grid(row=0, column=1, sticky="nsew")
        canvas_frame.rowconfigure(0, weight=1)
        canvas_frame.columnconfigure(0, weight=1)

        self.canvas = tk.Canvas(canvas_frame, bg="#222222")
        self.canvas.grid(row=0, column=0, sticky="nsew")
        self.hbar = ttk.Scrollbar(canvas_frame, orient="horizontal", command=self.canvas.xview)
        self.vbar = ttk.Scrollbar(canvas_frame, orient="vertical", command=self.canvas.yview)
        self.canvas.configure(xscrollcommand=self.hbar.set, yscrollcommand=self.vbar.set)
        self.hbar.grid(row=1, column=0, sticky="ew")
        self.vbar.grid(row=0, column=1, sticky="ns")
```

Рисунок 7 — пользовательский интерфейс

```
        # Правая панель
        right = ttk.Frame(self.root, width=260, padding=(8,8))
        right.grid(row=0, column=2, sticky="ns")
        right.grid_propagate(False)

        ttk.Label(right, text="Preview:", font=("TkDefaultFont", 10, "bold")).pack(anchor="w")
        self.preview_label = ttk.Label(right)
        self.preview_label.pack(fill="both", expand=True, pady=(8,8))

        ttk.Button(right, text="Сохранить", command=self.on_save).pack(fill="x", pady=4)
        ttk.Label(right, text="Параметры:").pack(anchor="w", pady=(8,0))
        self.params_text = tk.Text(right, height=8, wrap="word")
        self.params_text.pack(fill="both", pady=(4,0))

    def _bind_events(self):
        self.canvas.bind("<ButtonPress-1>", self._on_press)
        self.canvas.bind("<B1-Motion>", self._on_motion)
        self.canvas.bind("<ButtonRelease-1>", self._on_release)
        self.canvas.bind("<Double-Button-1>", self._on_double)

# Загрузка изображения с диска
    def on_load(self):
        path = filedialog.askopenfilename(
            title="Открыть файл", filetypes=[("Images", "*.png;*.jpg;*.jpeg;*.bmp;*.tiff"), ("All", "*.*")]
        )
        if path:
            try:
                self.processor.load_image(path)
            except Exception as e:
                messagebox.showerror("Error", str(e))
                return
            self._display(self.processor.get_image())
            self._update_params()

# Загрузка изображения через DND
    def on_crop(self):
        if self.sel_start is None:
            messagebox.showinfo("Info", "Нет выделенной области")
            return
        img = self.processor.get_image()
        bbox = self.canvas.bbox("IMG")
        if not bbox:
            x0, y0 = 0, 0
        else:
            x0, y0 = bbox[0], bbox[1]
        (x1, y1), (x2, y2) = self.sel_start, self.sel_end
        rel_x1 = (x1 - x0) / self.display_scale
        rel_y1 = (y1 - y0) / self.display_scale
        rel_x2 = (x2 - x0) / self.display_scale
        rel_y2 = (y2 - y0) / self.display_scale
        try:
            self.processor.crop(rel_x1, rel_y1, rel_x2, rel_y2)
        except Exception as e:
            messagebox.showerror("Error", str(e))
            return
        self._display(self.processor.get_image())
        self._update_preview()
        self._update_params()
        self._clear_sel()

# Ч/Б фильтр
    def on_grayscale(self):
        try:
            self.processor.to_grayscale()
        except Exception as e:
            messagebox.showerror("Error", str(e))
            return
        self._display(self.processor.get_image())
        self._update_preview()
        self._update_params()

# Изменения разрешения
    def on_resize(self):
        w = self.entry_w.get().strip()
        h = self.entry_h.get().strip()
        if not w or not h:
            messagebox.showinfo("Info", "Введите W и H")
            return
```

Рисунок 8 — Пользовательский интерфейс

```
        try:
            wi = int(w)
            hi = int(h)
            keep = bool(self.keep_aspect.get())
            self.processor.resize(wi, hi, keep_aspect=keep)
        except Exception as e:
            messagebox.showerror("Error", str(e))
            return
        self._display(self.processor.get_image())
        self._update_preview()
        self._update_params()

# Ресет изображения
    def on_reset(self):
        self.processor.reset_to_original()
        self._display(self.processor.get_image())
        self._update_preview()
        self._update_params()
        self._clear_sel()

    def on_save(self):
        img = self.processor.get_image()
        if img is None:
            messagebox.showinfo("Info", "Нет изображения для сохранения")
            return
        path = filedialog.asksaveasfilename(defaultextension=".png",
                                            filetypes=[("PNG","*.png"),("JPEG","*.jpg;*.jpeg"),("BMP","*.bmp")])
        if not path:
            return
        try:
            img.convert("RGB").save(path)
            messagebox.showinfo("Saved", f"Сохранено: {path}")
        except Exception as e:
            messagebox.showerror("Error", str(e))

    def _on_press(self, event):
        if not self.processor.get_image():
            return
        self.sel_start = (self.canvas.canvasx(event.x), self.canvas.canvasy(event.y))
        self.sel_end = self.sel_start
        if self.sel_rect_id:
            self.canvas.delete(self.sel_rect_id)
        x, y = self.sel_start
        self.sel_rect_id = self.canvas.create_rectangle(x, y, x+1, y+1, outline="cyan", dash=(4,2), width=2)

    def _on_motion(self, event):
        if self.sel_start is None or not self.sel_rect_id:
            return
        cx, cy = self.canvas.canvasx(event.x), self.canvas.canvasy(event.y)
        self.sel_end = (cx, cy)
        x0, y0 = self.sel_start
        self.canvas.coords(self.sel_rect_id, x0, y0, cx, cy)

    def _on_release(self, event):
        if self.sel_start is None:
            return
        self.sel_end = (self.canvas.canvasx(event.x), self.canvas.canvasy(event.y))
        x0, y0 = self.sel_start
        x1, y1 = self.sel_end
        self.sel_start = (min(x0, x1), min(y0, y1))
        self.sel_end = (max(x0, x1), max(y0, y1))
        if abs(self.sel_end[0] - self.sel_start[0]) < 5 or abs(self.sel_end[1] - self.sel_start[1]) < 5:
            self._clear_sel()

    def _on_double(self, event):
        self._clear_sel()

    def _clear_sel(self):
        if self.sel_rect_id:
            self.canvas.delete(self.sel_rect_id)
        self.sel_start = None
        self.sel_end = None
        self.sel_rect_id = None

    def _display(self, pil_img):
        self.canvas.delete("all")
        if pil_img is None:
            return
```

Рисунок 9 — Пользовательский интерфейс

```
            return
        canvas_w = max(200, self.canvas.winfo_width() or 800)
        canvas_h = max(200, self.canvas.winfo_height() or 600)
        img_w, img_h = pil_img.size
        margin = 20
        avail_w = max(100, canvas_w - margin)
        avail_h = max(100, canvas_h - margin)
        scale = min(avail_w / img_w, avail_h / img_h, 1.0)
        self.display_scale = scale if scale > 0 else 1.0

        disp = pil_img.resize((int(img_w * scale), int(img_h * scale)), Image.LANCZOS)
        self._tk_img = ImageTk.PhotoImage(disp)
        self.canvas.create_image(margin//2, margin//2, anchor="nw", image=self._tk_img, tags=("IMG",))
        self.canvas.configure(scrollregion=self.canvas.bbox("all"))

        self._update_preview()
        self._update_params()

# Превью
    def _update_preview(self):
        img = self.processor.get_image()
        if img is None:
            self.preview_label.config(image="", text="(нет картинки)")
            return
        max_side = 240
        w, h = img.size
        scale = min(max_side / w, max_side / h, 1.0)
        disp = img.resize((int(w * scale), int(h * scale)), Image.LANCZOS)
        self._preview = ImageTk.PhotoImage(disp)
        self.preview_label.config(image=self._preview)

# Обновление параметров изображения
    def _update_params(self):
        info = self.processor.get_info()
        self.params_text.delete("1.0", tk.END)
        if info:
            lines = [
                f"Путь: {info.get('path')}",
                f"Формат: {info.get('format')}",
                f"Ширина: {info.get('width')} px",
                f"Высота: {info.get('height')} px",
                f"Режим: {info.get('mode')}",
                f"Размер файла: {self._human_size(info.get('filesize_bytes'))}"
            ]
            self.params_text.insert(tk.END, "\n".join(lines))
            self._log("Обновлены параметры: " + os.path.basename(info.get("path") or ""))
        else:
            self.params_text.insert(tk.END, "(нет изображения)")

    def _human_size(self, size):
        if size is None:
            return "N/A"
        for unit in ['B','KB','MB','GB']:
            if size < 1024:
                return f"{size:.0f} {unit}"
            size /= 1024
        return f"{size:.2f} TB"

    def _log(self, msg):
        self.log_text.insert(tk.END, msg + "\n")
        self.log_text.see(tk.END)

    def _on_drop(self, event):
        data = getattr(event, "data", None)
        if not data:
            return
        raw = data.strip()
        if raw.startswith("{") and raw.endswith("}"):
            raw = raw[1:-1]
        path = raw.split()[0].strip("{}")
        if os.path.exists(path):
            try:
                self.processor.load_image(path)
                self._display(self.processor.get_image())
                self._update_params()
            except Exception as e:
                messagebox.showerror("Error", str(e))
        else:
            self._log(f"DnD: файл не найден {path}")

    def run(self):
        self.root.mainloop()
```

Рисунок 10 — Пользовательский интерфейс