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 «___» _____ 2024 г.

по «___» _____ 2024 г.

Руководитель практики от предприятия
должность_____

_____

подпись_____

Руководитель практики от
техникума: Материкова А.А.

_____

Оценка:_____

«___» _____2024 года

МП

г. Череповец

2024

Оглавление

Введение

Производственная практика является важным этапом профессиональной подготовки будущего специалиста, предоставляющим возможность закрепить теоретические знания, получить практические навыки и ознакомиться с реальными процессами разработки программного обеспечения. Практика проходила в ООО «Малленом Системс».

Целью прохождения практики было получения навыков анализа и ревьюирования программного кода, а также приобретение необходимых компетенций, для оценки качества программного обеспечения и выбора оптимальных инструментов разработки.

В рамках выполнения поставленной цели были сформулированы следующие задачи:

1. Осуществление ревьюирования программного кода в соответствии с технической документацией.
2. Измерение характеристик компонентов программного продукта для определения их соответствия заданным критериям.
3. Исследование созданного программного кода с использованием специализированных средств с целью выявления ошибок и отклонений от алгоритма.
4. Проведение сравнительного анализа программных продуктов и средств разработки.

В ходе практики были изучены основные этапы разработки программного обеспечения в компании, а также применены полученные ранее знания в реальных условиях производственного процесса.

1 Общая характеристика предприятия

Малленом Системс – ведущая российская компания в области разработки и внедрения систем компьютерного зрения, промышленной видео аналитики на основе технологий машинного зрения и искусственного интеллекта (машинное обучение, нейронные сети глубокого обучения) и интеллектуальной обработки данных.

Малленом Системс была создана в 2011 году на базе команды ученых и программистов Санкт-Петербургского политехнического университета Петра Великого. Сегодня в компании более 100 сотрудников. Глубокие компетенции в сфере машинного зрения и большой опыт успешной реализации проектов на промышленных предприятиях позволяет успешно решать большой спектр задач в различных отраслях. В Центре исследований и разработки интеллектуальных систем ведется работа по созданию новых решений и развитию продуктов компании.

В основе разработанных в компании систем лежат как собственные решения на базе нейронных сетей и детерминированных алгоритмов анализа изображений, так и алгоритмы от мирового лидера в области машинного зрения – компании Cognex.

1.1 Организационная структура предприятия

Организационная структура компании включает исследовательские и проектные отделы, которые работают над задачами в сфере машинного зрения и аналитики. В штате компании более 80 сотрудников, включая экспертов с учеными степенями, что позволяет ей успешно решать комплексные научно-технические задачи. Внутренняя структура также обеспечивает гибкость в управлении проектами для удовлетворения специфических потребностей клиентов в различных отраслях

1.2 Внутренний распорядок работы предприятия, охрана труда на предприятии.

В ООО «Малленом Системс» большое внимание уделяется охране труда, обеспечению безопасности и созданию комфортных условий для сотрудников.

Основные меры:

- **Система управления охраной труда:** разработка и контроль мероприятий по безопасности труда.
- **Обучение и инструктажи:** вводные и плановые инструктажи, информирование о правилах работы с оборудованием.
- **Профилактика заболеваний:** организация эргономичных рабочих мест, регулярные перерывы для отдыха глаз и упражнений.
- **Медицинское обеспечение:** ежегодные медосмотры и консультации для предотвращения профессиональных заболеваний.
- **Противопожарная безопасность:** системы пожаротушения, планы эвакуации и регулярные тренировки.

Такая политика компании снижает риски, повышает безопасность и эффективность работы сотрудников.

Продолжительность рабочего времени определяется долей ставки. Режим работы может быть установлен для работника индивидуально, по согласованию с руководителем, но при условии отработки нормы рабочего времени за неделю.

1.3 Должностные инструкции ИТ-специалистов предприятия

1. Общие положения

1.1. Настоящая должностная инструкция определяет должностные обязанности, права и ответственность Техника Общества с ограниченной ответственностью «Малленом Системс» (далее – Техник, Общество).

1.2. Техник относится к категории специалистов.

1.3. Техник принимается на работу и увольняется приказом генерального директора или уполномоченным им лицом.

1.4. На должность Техника назначается лицо, без предъявления требований к образованию и опыту работы.

1.6. Техник должен знать:

– методы автоматической и автоматизированной проверки работоспособности программного обеспечения;

– основные виды диагностических данных и способы их представления;

– языки, утилиты и среды программирования, и средства пакетного выполнения процедур;

– типовые метрики программного обеспечения;

– основные методы измерения и оценки характеристик программного обеспечения;

– методы создания и документирования контрольных примеров и тестовых наборов данных;

– правила, алгоритмы и технологии создания тестовых наборов данных;

– требования к структуре и форматам хранения тестовых наборов данных;

– методы и средства проверки работоспособности программного обеспечения;

– среду проверки работоспособности и отладки программного обеспечения;

1.4. Техник должен знать и уметь:

– писать программный код процедур проверки работоспособности программного обеспечения на выбранном языке программирования под руководством наставника;

– использовать выбранную среду программирования для разработки процедур проверки работоспособности программного обеспечения на выбранном языке программирования;

– применять методы и средства проверки работоспособности программного обеспечения;

– анализировать значения полученных характеристик программного обеспечения;

– документировать результаты проверки работоспособности программного обеспечения;

2. Должностные обязанности

Техник выполняет следующие должностные обязанности:

2.1 Выполняет работу по проведению необходимых технических расчетов;

2.2 Осуществляет наладку, настройку, регулировку и опытную проверку оборудования и систем, следит за его исправным состоянием;

2.3 Принимает участие в проведение экспериментов и испытаний;

2.4 Принимает участие в разработке программ, инструкций и другой технической документации, в изготовлении макетов, а также в испытаниях и экспериментальных работах;

3. Права

Техник имеет право:

3.1. Участвовать в обсуждении проектов решений, в совещаниях по их подготовке и выполнению.

3.2. Запрашивать у непосредственного руководителя разъяснения и уточнения по данным поручениям, выданным заданиям.

3.3. Запрашивать по поручению непосредственного руководителя и получать от других работников организации необходимую информацию, документы, необходимые для исполнения поручения.

4. Обязанности и ответственность

Техник обязан:

4.1. Соблюдать локально-нормативные акты Общества.

4.2. Не разглашать информацию и сведения, являющиеся коммерческой тайной.

4.3. Использовать только принятые в Обществе программные инструменты и технологию разработки программного обеспечения.

4.4. Соблюдать трудовую и производственную дисциплину, правила и нормы охраны труда, требования производственной санитарии и гигиены, требования противопожарной безопасности.

Ведущий программист привлекается к ответственности:

2 Ревьюирование программных продуктов

2.1 Ревьюирование программного кода в соответствии с технической документацией

1 Диаграмма компонентов:

Диаграмма компонентов показывает основные компоненты программы.

Описание компонентов

- UI.py: Основной модуль который содержит в себе пользовательский интерфейс.
- processing.py: Второй модуль в котором храниться логика обработки изображения.

Связи компонентов:

UI.py связан с processing.py

processing.py связан с UI.py

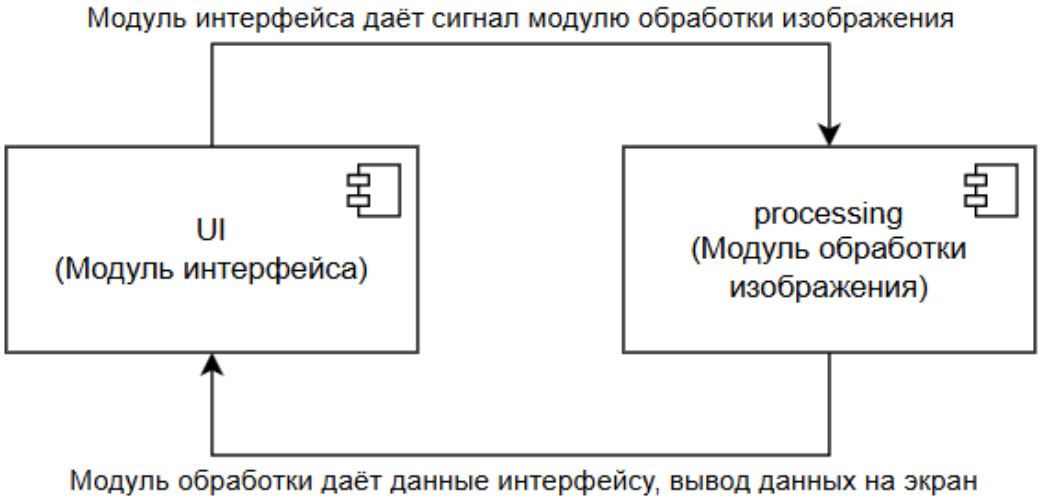

Рисунок 1

2. Диаграмма сценарии использования

Показывает основные взаимодействия пользователя с программой.

Основные сценарии:

1. Выбор изображения:
   - Пользователь нажимает на кнопку “Выбрать изображение” у него открывается папка с изображениями и он выбирает нужное изображение в формате PNG.
   - Программа отображает изображение.
2. Вывод данных о изображении.
   - После того как пользователь выбрал изображение программа собирает данные о изображении и выводит это на интерфейс
3. Переименование изображения:
   - Пользователь нажимает кнопку “переименовать”

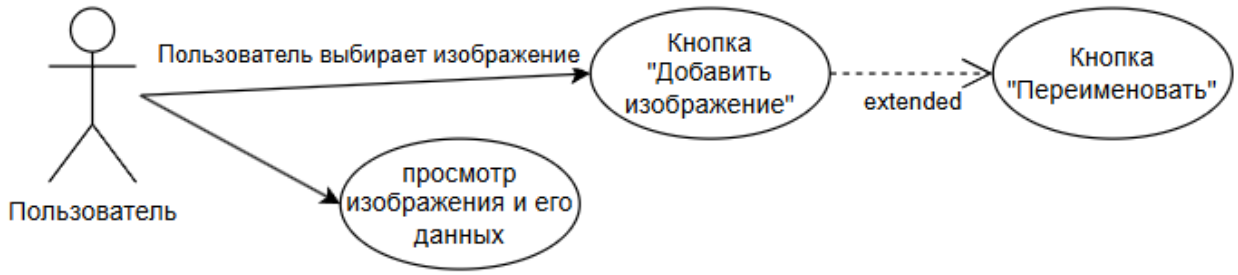

Рисунок 2

3. Диаграмма последовательности

Последовательность вывода данных изображения

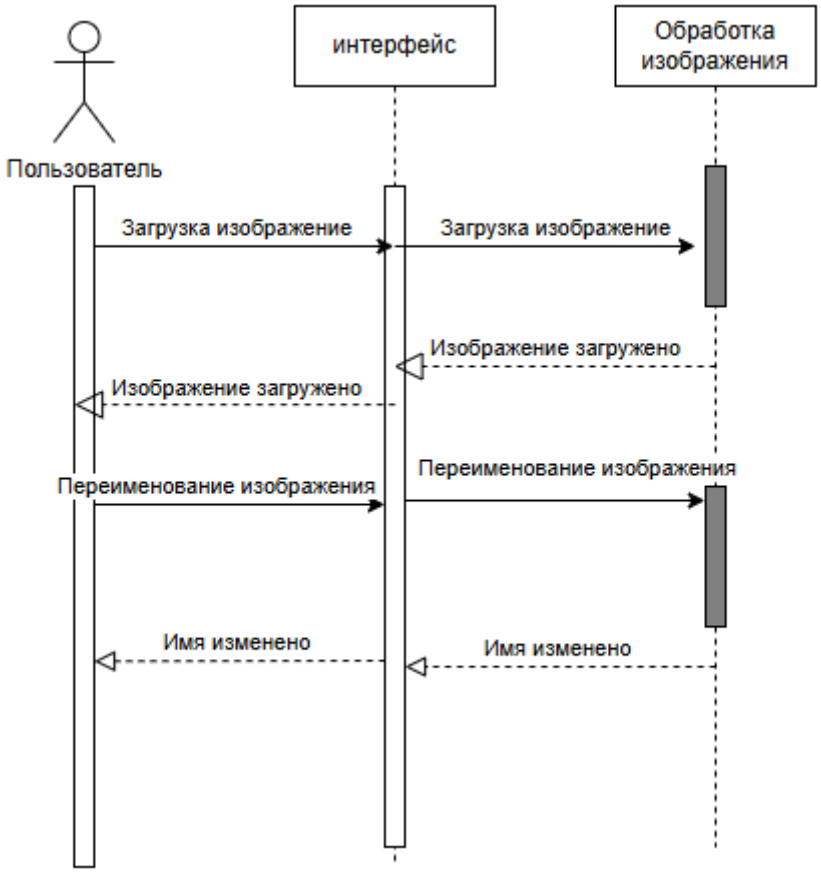

Рисунок 3

4. Диаграмма деятельности.

Общий процессы переименования файла.

1. Начало.
2. Пользователь нажимает “Переименовать”
   - Открывается файловый менеджер.
3. Программа проверяет:
   - Расширение файла
4. Если условия выполнены:
   - Переименовывает изображение.
   - Выводит сообщение об успешном переименовании.
5. Если условия не выполнены
   - Выводит ошибку
6. Конец.

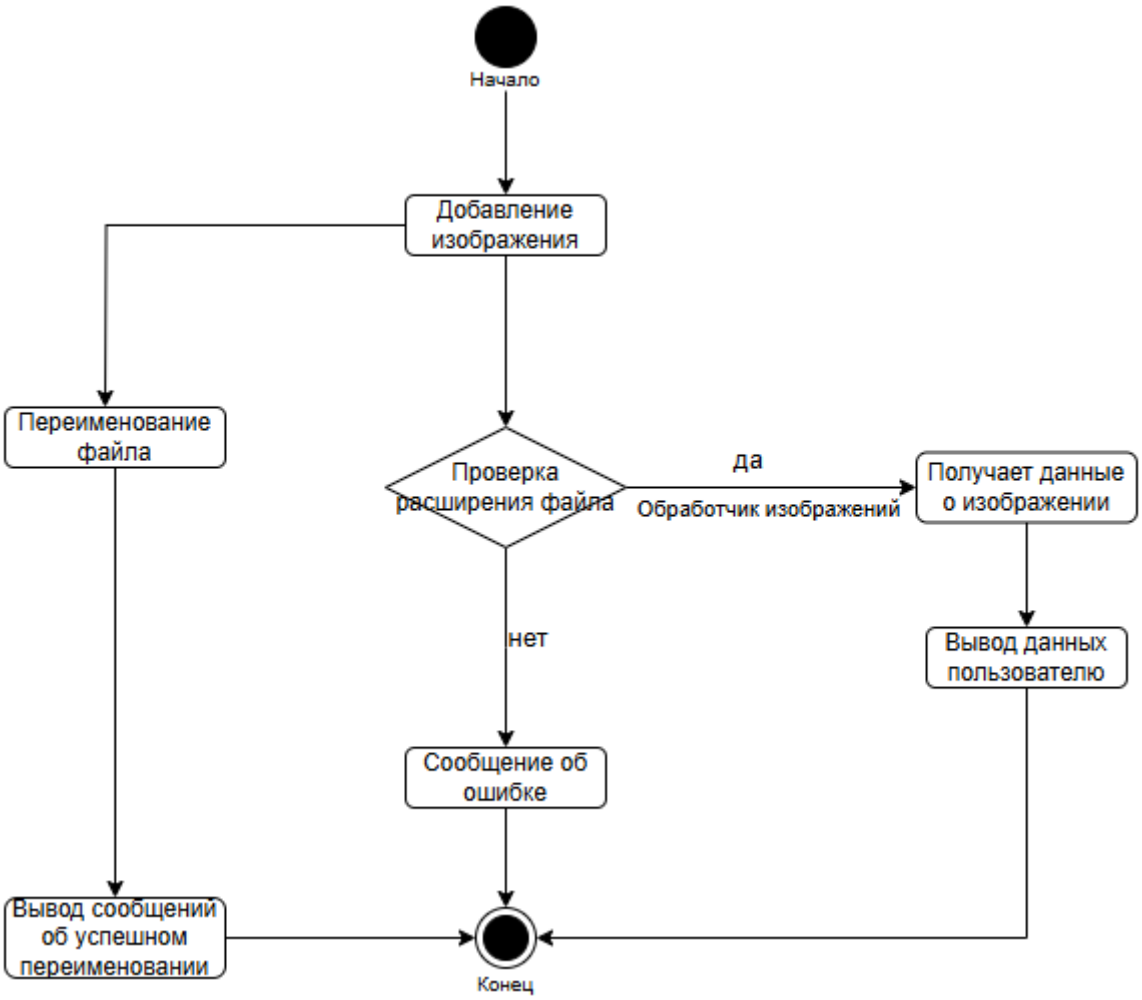
Начало
Добавление изображения
Переименование файла
Проверка расширения файла
да
Обработчик изображений
Получает данные о изображении
Вывод данных пользователю
нет
Сообщение об ошибке
Вывод сообщений об успешном переименовании
Конец

Рисунок 4

2.2 Измерение характеристик компонент программного продукта

Измерение характеристик компонентов программного продукта – данный процесс включает анализ функциональных и нефункциональных аспектов, что помогает выявить сильные и слабые стороны программного обеспечения, а также принятии решения о необходимости его доработки.

Цели измерения характеристик:

1. **Оценка качества**: Определение степени соответствия характеристик ПО установленным стандартам.
2. **Идентификация проблем**: Выявление багов, недочетов в производительности и несоответствий нормам.
3. **Оптимизация разработки**: Оценка эффективности примененных технологий и методов.
4. **Сравнение решений**: Выбор лучших компонентов на основании полученных метрик.

Основные характеристики для измерения:

1. **Функциональность**:
    - **Корректность**: Соответствует ли продукт заявленной логике?
    - **Соответствие спецификациям**: Выполняет ли ПО все функции согласно техническому заданию?
2. **Надежность**:
    - **Устойчивость к ошибкам**: Способен ли продукт стабильно работать при возникновении непредвиденных ситуаций?
    - **Время безотказной работы**: Как долго система может функционировать без сбоев?

3. **Производительность**:
    - **Время выполнения операций**: Насколько быстро выполняются задачи?
    - **Использование ресурсов**: Как сильно загружаются процессор и память?
4. **Удобство использования**:
    - **Время на выполнение задач пользователем**: Сколько времени требуется пользователям для достижения цели?
    - **Частота ошибок пользователя**: Часто ли пользователи допускают ошибки при работе с продуктом?
5. **Сопровождаемость**:
    - **Простота модификации кода**: Легко ли вносить изменения в кодовую базу?
    - **Доступность документации**: Есть ли необходимая документация для поддержки и развития продукта?
6. **Переносимость**:
    - **Совместимость с разными ОС**: Может ли программа работать на различных операционных системах?
    - **Минимальные изменения для переноса**: Требуется ли значительное изменение кода для адаптации под новую платформу?

Методы измерения:

1. **Статический анализ**:
    - Анализируется исходный код с использованием специализированных инструментов.

- Проверяется стиль написания кода, выявляется избыточный или неиспользованный код.

2. **Динамическое тестирование**:
   - Измеряются характеристики программы непосредственно во время её исполнения.
   - Проводятся нагрузочные тесты и стресс-тесты для проверки устойчивости системы.
3. **Метрики качества ПО**:
   - Рассчитываются количественные показатели, такие как количество строк кода (LOC), плотность ошибок и покрытие тестами (Code Coverage).

Инструменты измерения:

- **Статический анализ**: SonarQube, ESLint, Pylint.
- **Тестирование производительности**: JMeter, Gatling.
- **Измерение метрик кода**: SonarQube, VS Code Metrics.

2.3 Исследование созданного программного кода с использованием специальных программных средств

Статический анализ

Цель: Выявление ошибок, нарушений стандартов кодирования и потенциальных улучшений.

Используемые инструменты:

- **pylint**: Проверяет качество кода.
- **mypy**: Проверяет аннотации типов.
- **flake8**: Анализирует стиль и синтаксис.

Динамический анализ

Цель: Исследование производительности, использование памяти и обработка исключений.

Используемые инструменты:

- **timeit**: Измерение времени выполнения функций.
- **memory_profiler**: Анализ потребления памяти.
- **pytest**: Проведение тестирования функциональности.

Визуализация структуры вызовов

Цель: Создание графа вызовов для анализа взаимодействий между модулями.

Используемый инструмент:

- **pycallgraph**: Генерация графов вызовов.

2.4 Сравнительный анализ программных продуктов и их средств разработки

Для работы с кодом я использовал визуальную среду Visual Studio Code, для меня эта среда очень удобная с простым интерфейсом и низким порогом входа. Существует много разных сред программирования, ниже я приведу пример этих сред и опишу их преимущества и недостатки.

Visual Studio Code (VSC) – это популярный кроссплатформенный редактор кода от компании Microsoft.

Преимущества

1. **Кросс-платформенность**: VSC работает на Windows, macOS и Linux, что делает его универсальным инструментом для разработчиков независимо от их операционной системы.
2. **Расширения**: Огромная библиотека плагинов позволяет адаптировать редактор под конкретные задачи разработки. Можно установить

расширения для поддержки различных языков программирования, инструментов сборки, тестировочных фреймворков и многое другое.

3. **Интеграция с Git**: Встроенная поддержка системы контроля версий Git упрощает работу с репозиториями прямо из интерфейса редактора.

4. **Поддержка множества языков программирования**: Поддерживает синтаксическое выделение, автодополнение и другие функции для большого количества языков, включая Python, JavaScript, C++, Go, Rust и многие другие.

Недостатки

1. **Потребляет много ресурсов**: Хотя VSC легкий по сравнению с другими IDE, он все же может потреблять значительное количество оперативной памяти при работе с большими проектами или множеством открытых файлов.

2. **Ограниченность по сравнению с полноценными IDE**: Некоторые разработчики могут упустить возможности полноценной интегрированной среды разработки (IDE), такие как сложные инструменты рефакторинга, анализ кода и управление зависимостями.

3. **Проблемы с производительностью при большом количестве плагинов**: Установка слишком большого числа расширений может замедлить работу редактора и сделать его менее отзывчивым.

4. **Неидеальная интеграция с некоторыми языками и инструментами**: Хотя VSC поддерживает множество языков и технологий, иногда возникают проблемы с интеграцией специфических инструментов или библиотек.

PyCharm мощная и функциональная IDE, идеально подходящая для профессиональных разработчиков на Python.

Преимущества

1. **Поддержка различных фреймворков:** PyCharm поддерживает работу с популярными веб-фреймворками, такими как Django, Flask, а также с инструментами для работы с данными, такими как NumPy, Pandas и другие библиотеки машинного обучения.

2. **Работа с версиями:** Поддерживается работа с Git, Mercurial и другими системами контроля версий прямо из интерфейса IDE. Можно легко сравнивать версии файлов, делать коммиты и пушить изменения.

3. **Плагины и расширения:** PyCharm имеет обширную экосистему плагинов, которые позволяют расширить функциональность среды разработки под конкретные задачи. Например, плагины для работы с Docker, SQL, SSH и многие другие.

4. **Кросс-платформенность:** PyCharm доступен для Windows, macOS и Linux, что делает его удобным выбором для разработчиков, работающих на разных операционных системах.

Недостатки

1. **Высокие системные требования**: PyCharm требует значительных ресурсов системы, особенно при работе с большими проектами. На слабых компьютерах он может работать медленно или зависать.

2. **Платная лицензия**: Профессиональная версия PyCharm платная, и стоимость лицензии может быть высокой для индивидуальных разработчиков или небольших команд.

3. **Уровень сложности для новичков**: Несмотря на наличие документации и обучающих материалов, PyCharm может показаться сложным для начинающих программистов, так как содержит много настроек и функций.

4. **Медленная загрузка**: Запуск PyCharm может занимать некоторое время, особенно после обновления или установки новых плагинов.

5. **Ограниченная поддержка других языков**: Хотя PyCharm ориентирован на Python, поддержка других языков программирования менее развита. Если вам нужно работать с несколькими языками одновременно, возможно, стоит рассмотреть альтернативные IDE.

Sublime Text — это популярный текстовый редактор, который часто используется разработчиками благодаря своей легкости, скорости и гибкости. Он подходит для написания кода на различных языках программирования

Преимущества

1. **Легкость и скорость**: Sublime Text известен своей скоростью запуска и быстротой работы даже на старых машинах. Это особенно важно для тех, кто работает с большим количеством файлов или проектов.
2. **Настраиваемость**: Редактор предлагает огромное количество настроек и плагинов, позволяющих адаптировать его под любые нужды. Вы можете изменить тему, шрифт, цвета синтаксиса, добавить новые функции через плагины и многое другое.
3. **Многосекционное редактирование**: Возможность одновременной работы с несколькими файлами или частями одного файла значительно упрощает разработку. Вы можете открыть несколько панелей и работать сразу над несколькими фрагментами кода.
4. **Подсветка синтаксиса**: Редактор поддерживает подсветку синтаксиса для множества языков программирования, что улучшает читаемость кода и помогает быстро находить ошибки.

Недостатки

1. **Не бесплатная лицензия**: Хотя Sublime Text можно использовать бесплатно без ограничений по времени, периодически появляется

напоминание о покупке лицензии. Это может раздражать некоторых пользователей.

2. **Ограниченность встроенной функциональности**: По сравнению с полноценными IDE, такими как Visual Studio Code или PyCharm, Sublime Text имеет меньше встроенных функций, таких как отладка, интеграция с системами контроля версий и прочее. Эти возможности можно добавить через плагины, но это требует дополнительных усилий.
3. **Сложность освоения для новичков**: Настройка и использование всех возможностей Sublime Text может потребовать времени и опыта. Новичкам может быть сложно разобраться со всеми функциями и плагинами.
4. **Проблемы с производительностью на больших проектах**: При работе с очень большими проектами или файлами Sublime Text может начать тормозить или потреблять слишком много памяти.

3. Выполняемые задания

В ходе производственной практики мне было дано задание разработать модули для обработки изображения используя на выбор два языка Python и C#, я выбрал python и использовал библиотеки Tkinter и Pillow (PIL). Нужно было создать два разных модуля, первый модуль осуществлял собор данных с изображения и выводил эти данные на экран и второй модуль это графический интерфейс.

Программа должна предоставлять графический интерфейс, которой позволяет пользователю выбирать изображение и просматривать дату создания, размер изображения и размер файла. Программа ещё должна переименовывать файл, переименовывая файл автоматический заменяется в той папке, где он лежал.

Заключение

Практика в ООО “Малленом Системс” позволила мне улучшить навык анализа и разработки программного обеспечения и улучшила знание работы с некоторыми библиотеками Python также работа над проектом помогла мне понять процесс проектирования. В ходе практики мной был проведен анализ программных продуктов а также работа включала в себя ревьюирование кода, измерение характеристик производительности.

Список использованных источников

1. Работа с модулями Python - https://metanit.com/python/tutorial/2.10.php
2. Работа с библиотеками https://metanit.com/sharp/tutorial/3.46.php
3. UML - https://practicum.yandex.ru/blog/uml-diagrammy/?
4. Пример измерения скорости используя Time - https://www.geeksforgeeks.org/how-to-check-the-execution-time-of-python script/
5. Диаграммы https://app.diagrams.net/

Приложения

Вид программы:

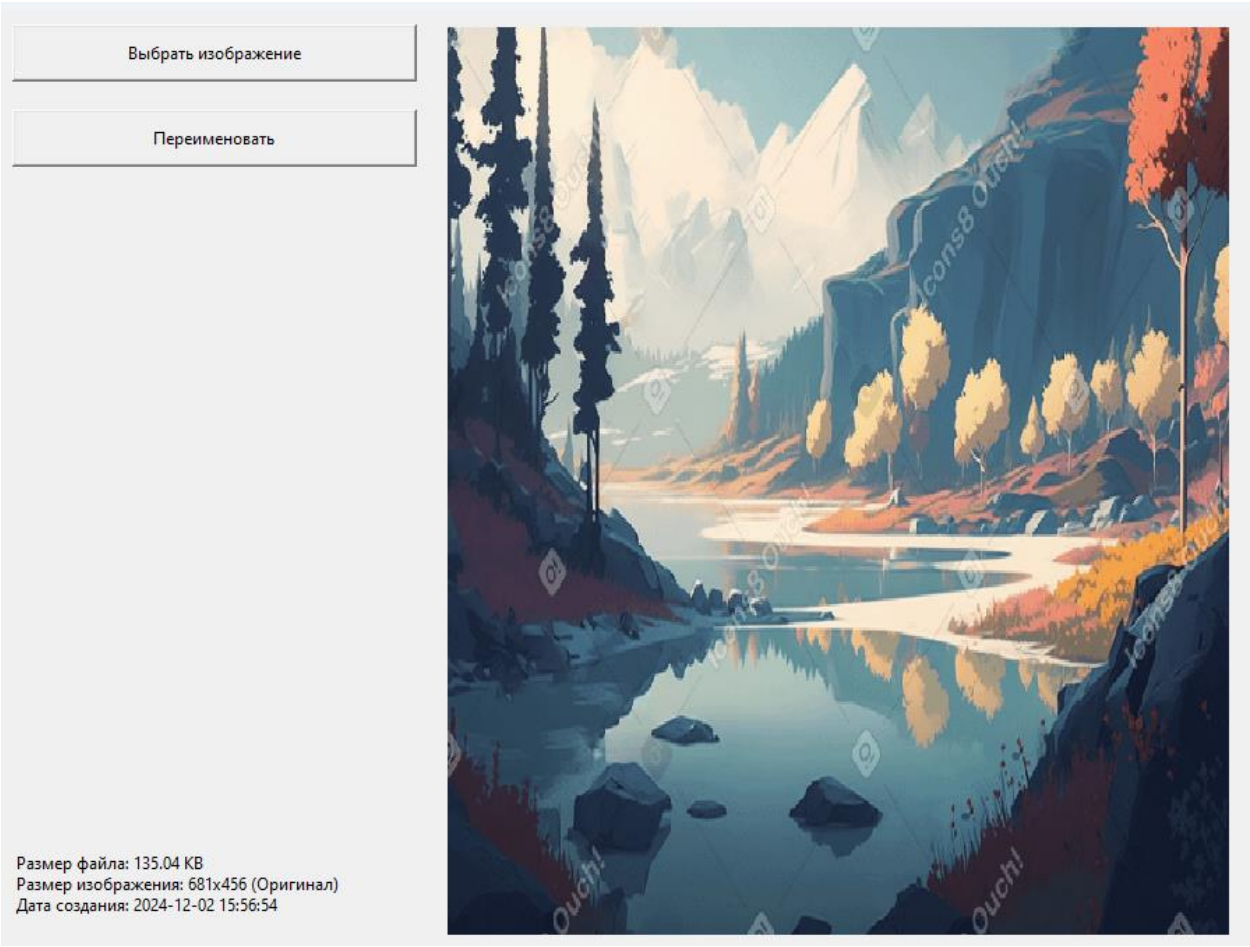

Рисунок 5