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

Содержание

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. Данная практика проходила в ООО «Малленом Системс», одной из ведущих компаний в области разработки высокотехнологичных решений.

Целью прохождения практики было освоение навыков анализа и ревьюирования программного кода, а также приобретение компетенций, необходимых 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 для выбора наилучшего решения в соответствии с требованиями технического задания.

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

ООО "Малленом Системс" — российская компания, специализирующаяся на разработке и внедрении систем машинного зрения и видеоаналитики для промышленных и транспортных предприятий. Созданная в 2011 году на основе команды из Санкт-Петербургского политехнического университета, компания развивает решения на базе технологий искусственного интеллекта и машинного обучения, в том числе для анализа изображений и обработки данных в реальном времени. Основные направления деятельности компании включают разработку систем контроля качества продукции, отслеживания и идентификации товаров, а также решения для промышленного и транспортного секторов. К примеру, для железнодорожной отрасли компания предлагает систему распознавания номеров вагонов, что способствует автоматизации учёта и контроля на станциях Компания активно разрабатывает и адаптирует программные и аппаратные решения для различных отраслей, включая металлургию, фармацевтику и логистику. Она создала и внедрила систему ВИСКОНТ. Фарма, которая выполняет задачи сериализации и агрегации лекарственных средств для отслеживания их оборота. Эта система интегрируется с ERP-платформами и позволяет отслеживать продукцию по всему логистическому циклу, что особенно актуально в фармацевтической отрасли. По структуре "Малленом Системс" включает отделы научных исследований и разработки, где трудится команда из более чем 80 специалистов, включая доктора и кандидатов наук.

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

Основные должности в компании:

Инженер-программист

разработка приложений под ОС Windows;

интеграция с алгоритмами машинного обучения;

программирование UI;

реализация алгоритмов машинного зрения;

доработка существующих проектов;

оптимизация и рефакторинг.

Специалист по машинному обучению

дообучение / улучшение существующих нейросетей, используемых в production;

создание и обучение нейросетей;

анализ современных моделей на применимость их бизнес-задачам компании;

визуализация данных;

работа с датасетами.

Инженер

проработка и согласование технических заданий по проектам;

подбор оборудования и комплектующих, разработка спецификаций;

подготовка оборудования к инсталляции;

выполнение проектно-изыскательских работ;

выполнение пусконаладочных работ на объектах внедрения (служебные командировки);

обучение операционного персонала Заказчика;

техническая поддержка клиентов;

разработка технической документации

Специалист по тестированию ПО

ручное тестирование;

составление тестовых сценариев;

поддержка и расширение документации по продуктам проекта;

документирование и верификация дефектов, контроль исправления выявленных ошибок разработчиком;
взаимодействие с командой разработки и технической поддержки;
тестирование продуктов проекта;
актуализация документации по продуктам проекта.

Менеджер по продажам

Обработка входящих запросов от клиентов.
Ведение коммерческих переговоров с клиентами, консультирование о продуктах Малленом Системс для транспортной отрасли.
Подготовка ТКП (совместно с техническими специалистами), согласование конфигурации продукции под каждую задачу, подбор оборудования под проект.
Заключение договоров (совместно с юристом) и их сопровождение.
Контроль работы по отгрузке и доставке товаров покупателям по заключенным договорам, подготовка товара к отправке.
Контроль оплаты договоров клиентами.
Участие в торгах на поставку продукции Компании на торговых площадках
Ведение информационных баз клиентов и партнеров, документооборота.

2 Ревьюирование программных продуктов

2.1 Ревьюирование программного кода в соответствии с технической документацией

1. Диаграмма компонентов
Диаграмма компонентов показывает основные модули системы и их зависимости. Диаграмма представлена на Рисунке 1.

Описание компонентов:

- main.py: Главный модуль, запускает приложение.
- main_window.py: Определяет пользовательский интерфейс (GUI).

- image_processing.py: Предоставляет функции обработки изображений.
- file_utils.py: Реализует утилиты для работы с файлами.
- PyQt5: Внешняя библиотека для создания GUI.
- Pillow (PIL): Внешняя библиотека для обработки изображений.

Связи компонентов:

- main.py зависит от main_window.py.
- main_window.py зависит от image_processing.py и file_utils.py.
- image_processing.py использует Pillow.
- main_window.py зависит от PyQt5.

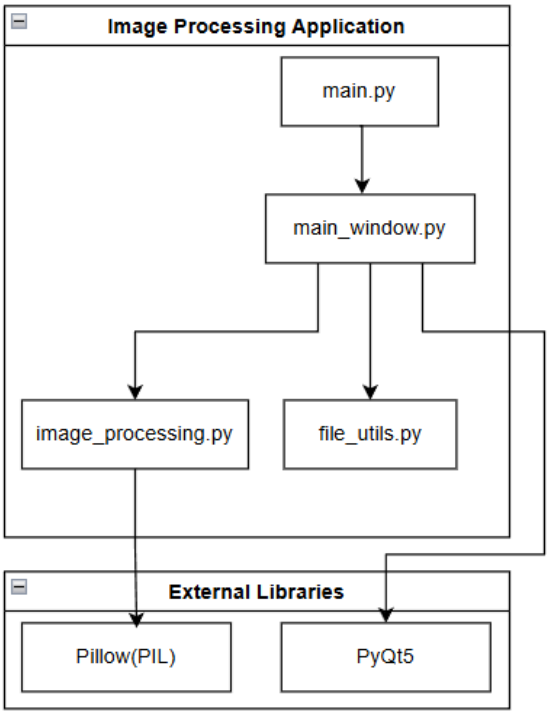


Рисунок 1- Диаграмма компонентов

2. Диаграмма сценариев использования

Показывает основные взаимодействия пользователя с системой. Диаграмма представлена на Рисунке 2.

Основные сценарии:

1. Выбрать изображение:

- Пользователь вводит путь или выбирает изображение через диалоговое окно.
- Программа отображает изображение.

2. Конвертация в градации серого:
   - Пользователь нажимает кнопку "Convert to Grayscale".
   - Программа вызывает функцию обработки изображения и отображает результат.
3. Перемещение изображения:
   - Пользователь нажимает кнопку "Move Image".
   - Программа перемещает файл в указанную папку.

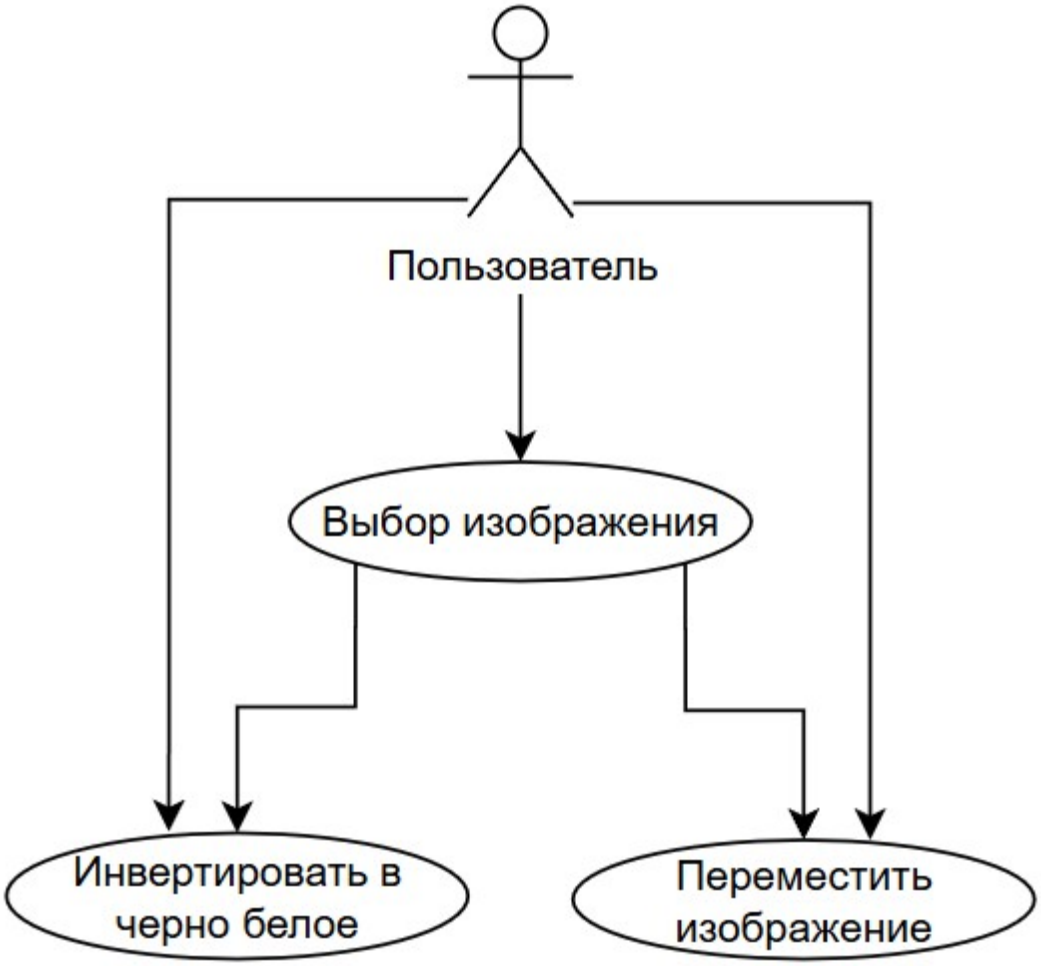


Рисунок 2 - Диаграмма сценариев использования

3. Диаграмма последовательности

Отражает последовательность вызовов для сценария "Конвертация в градации серого". Диаграмма представлена на Рисунке 3.

1. Пользователь нажимает кнопку "Convert to Grayscale".
2. MainWindow.handle_grayscale():
    - Проверяет корректность пути.
    - Вызывает convert_to_grayscale(image_path).
3. convert_to_grayscale():
    - Загружает изображение.
    - Конвертирует в градации серого.
    - Сохраняет новое изображение.
4. MainWindow.display_image():
    - Отображает преобразованное изображение.
5. Выводит сообщение об успешной операции.

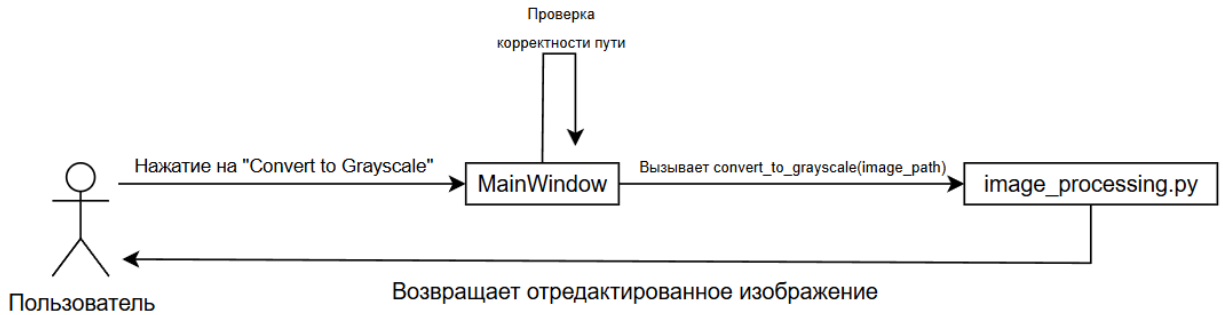


Рисунок 3 - Диаграмма последовательности

4. Диаграмма деятельности

Показывает общий процесс для сценария "Перемещение изображения". Диаграмма представлена на Рисунке 4.

1. Начало.
2. Пользователь нажимает "Move Image".
3. Открывается диалог выбора директории.
4. Программа проверяет:

- Выбран файл.
- Директория указана.

5. Если условия выполнены:
    - Перемещает файл.
    - Выводит сообщение об успешной операции.
6. Если условия не выполнены:
    - Выводит предупреждение.
7. Конец.

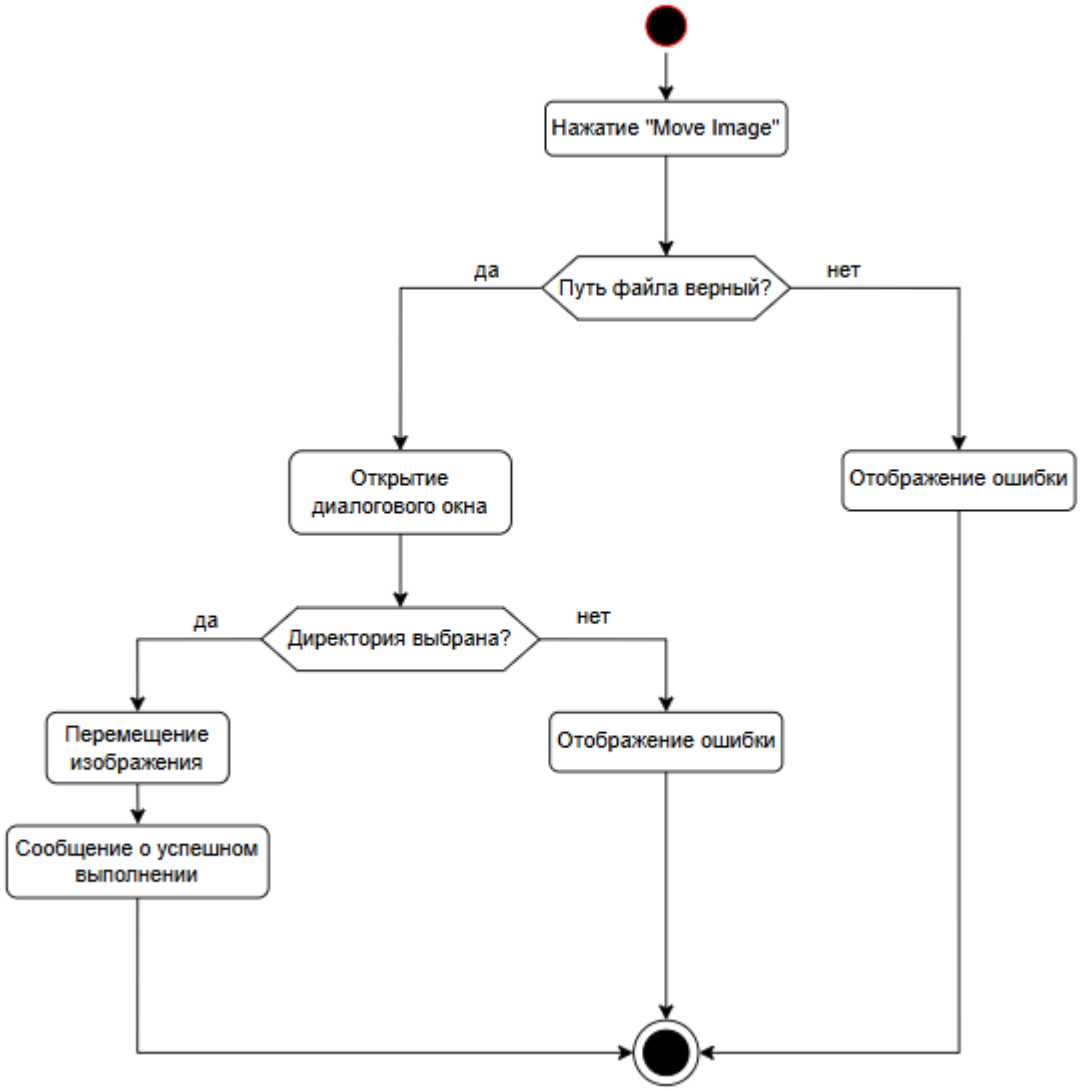


Рисунок 4 - Диаграмма деятельности

2.2 Измерение характеристик компонент программного продукта

Измерение характеристик компонентов программного продукта является ключевым этапом оценки его качества и соответствия заданным требованиям. Этот процесс направлен на анализ функциональных и нефункциональных характеристик, что позволяет выявить достоинства и недостатки программного обеспечения (ПО) и принять обоснованные решения по его доработке.

Цели измерения характеристик

1. Оценка качества: определение уровня соответствия характеристик ПО установленным требованиям.
2. Идентификация проблем: выявление дефектов, недостатков производительности и отклонений от норм.
3. Оптимизация разработки: определение эффективности используемых технологий и подходов.
4. Сравнение решений: выбор наилучших компонентов на основе измеряемых метрик.

Основные характеристики для измерения

1. Функциональность: проверка соответствия функций программного продукта требованиям технического задания.
    - Корректность.
    - Соответствие спецификациям.
2. Надежность: способность системы функционировать без отказов в определенных условиях.
    - Устойчивость к ошибкам.
    - Время безотказной работы.
3. Производительность: время отклика, пропускная способность и использование ресурсов.
    - Время выполнения операций.
    - Загрузка процессора, памяти.
4. Удобство использования (юзабилити): легкость освоения и применения ПО.

- Время на выполнение задач пользователем.
- Частота ошибок пользователя.

5. Сопровождаемость: легкость внесения изменений, адаптация к новым условиям.
    - Простота модификации кода.
    - Доступность документации.
6. Переносимость: способность программы работать на разных платформах.
    - Совместимость с различными операционными системами.
    - Минимальные изменения для переноса.

Методы измерения

1. Статический анализ: анализ исходного кода с помощью автоматизированных инструментов.
    - Проверка стиля.
    - Выявление избыточного или неиспользуемого кода.
2. Динамическое тестирование: измерение характеристик во время выполнения программы.
    - Нагрузочное тестирование.
    - Стресс-тесты.
3. Метрики качества ПО: расчет формализованных показателей.
    - Количество строк кода (LOC).
    - Плотность ошибок.
    - Покрытие тестами (Code Coverage).

Инструменты измерения

- Статический анализ: SonarQube, Pylint, Checkstyle.
- Тестирование производительности: Apache JMeter, LoadRunner.
- Измерение метрик кода: CodeClimate, ESLint, ReSharper.

2.3 Исследование созданного программного кода с использованием специализированных программных средств

1. Статический анализ кода

Цель: Выявление ошибок, нарушение стандартов кодирования и возможных улучшений.

Инструменты:

- pylint: Проверка качества кода.
- flake8: Анализ стиля и синтаксиса.
- mypy: Проверка аннотаций типов.

2. Динамический анализ

Цель: Изучение производительности, использования памяти и обработки исключений.

Инструменты:

- memory_profiler: Анализ памяти.
- timeit: Измерение времени выполнения функций.
- pytest: Тестирование функциональности.

3. Визуализация структуры вызовов

Цель: Построить граф вызовов для анализа взаимодействия между модулями.

Инструмент:

- pycallgraph: Построение графов вызовов.

2.4 Сравнительный анализ программных продуктов и средств разработки

1. PyQt5 (GUI - main_window.py)

PyQt5 — это мощный фреймворк для создания графических интерфейсов.

Возможности:

- Широкий функционал: Поддерживает создание сложных интерфейсов с множеством виджетов (кнопки, текстовые поля, окна диалогов и т.д.).
- Кроссплатформенность: Работает на Windows, macOS и Linux.
- Поддержка стилей: Можно настраивать внешний вид интерфейса через CSS или встроенные темы.
- Интеграция с Python: Позволяет легко связывать интерфейс с логикой программы.

Недостатки:

- Сложность: Более громоздкий синтаксис по сравнению с другими GUI-библиотеками, такими как Tkinter.
- Зависимость от Qt: Большой объем библиотеки увеличивает размер программы.
- Лицензия: Для коммерческого использования требуется приобрести лицензию.

Альтернативы:

1. Tkinter:
    - Простая встроенная библиотека Python.
    - Подходит для небольших проектов.
    - Менее мощная по сравнению с PyQt5.
2. Kivy:
    - Подходит для кроссплатформенной разработки, включая мобильные устройства.
    - Имеет интуитивный дизайн, но меньше возможностей для сложных интерфейсов.
3. PySide:

- Аналог PyQt5 с лицензией LGPL (подходит для коммерческого использования).

2. Pillow (PIL) (Обработка изображений - image_processing.py)

Pillow — это мощная библиотека для работы с изображениями.

Возможности:

- Широкий спектр инструментов: Поддержка различных форматов изображений, возможность изменения размеров, применения фильтров, преобразования в градации серого и т.д.
- Простота использования: Легко интегрируется в проекты.
- Кроссплатформенность: Работает на всех популярных ОС.

Недостатки:

- Ограничения в обработке: Не подходит для высокопроизводительных задач по сравнению с библиотеками на C++ (например, OpenCV).
- Поддержка форматов: Хотя поддерживает многие форматы, для специализированных задач (например, DICOM) требуются дополнительные библиотеки.

Альтернативы:

1. OpenCV:
    - Более производительная библиотека для обработки изображений и компьютерного зрения.
    - Подходит для сложных задач, таких как распознавание лиц или обработки видео.
2. ImageIO:
    - Легковесная библиотека для чтения и записи изображений.
- Подходит для базовой обработки.

3. Стандартная библиотека Python (Работа с файлами - file_utils.py)

Python предлагает мощные встроенные модули для работы с файлами и каталогами.

Возможности:

- Модули os и shutil: Позволяют работать с файловой системой, копировать, перемещать, удалять файлы и директории.
- Простота интеграции: Не требуют установки сторонних библиотек.
- Поддержка всех ОС.

Недостатки:

- Отсутствие высокоуровневых функций: Нет инструментов для работы с облачными хранилищами или специальными файловыми форматами.
- Неинтуитивный интерфейс: Работа с путями может быть сложной без дополнительных библиотек, таких как pathlib.

Альтернативы:

1. pathlib (Python 3.4+):
    - Современный способ работы с путями файловой системы.
    - Интуитивно понятный и удобный синтаксис.
2. watchdog:
    - Подходит для мониторинга изменений в файловой системе в реальном времени.
3. HDFS/S3 SDKs:
    - Для работы с распределенными хранилищами данных (например, Hadoop или Amazon S3).

3 Выполненные задания

В ходе производственной практики мне было поручено разработать модули для обработки изображений с использованием языка программирования Python и его библиотек PyQt5 и Pillow (PIL). Первый модуль должен выполнять

преобразование цветовой гаммы изображений в черно-белый формат. Второй модуль предназначен для перемещения изображений.

Программа должна предоставлять графический интерфейс, который позволяет пользователю вручную задавать путь к изображению и просматривать результат обработки. После выполнения изменений изображение автоматически сохраняется в той же директории, что и оригинал, при этом исходный файл остается неизменным.

main.py:

Основной модуль программы, отвечающий за запуск приложения. Этот модуль создает экземпляр приложения (QApplication) и вызывает главное окно (MainWindow), в котором реализован весь пользовательский интерфейс и функционал. Создание этого модуля началось с подключения библиотеки PyQt5, необходимой для создания графического интерфейса. Затем была написана функция main(), которая инициализирует приложение, создает объект главного окна, отображает его и запускает цикл обработки событий. Этот модуль выполняет роль "точки входа" программы, где начинается выполнение кода. Исходный код модуля представлен на рисунке 5.

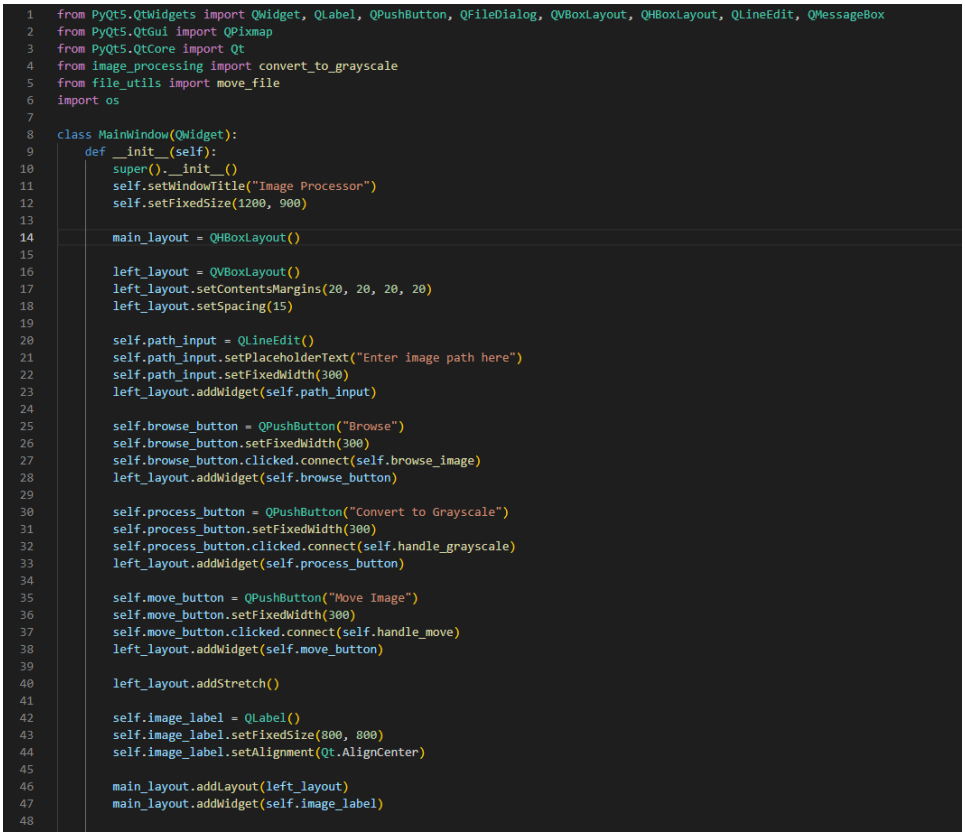
```python
import sys
from PyQt5.QtWidgets import QApplication
from main_window import MainWindow

def main():
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())

if __name__ == "__main__":
    main()
```

Рисунок 5 - Код модуля main.py

main_window.py

Этот модуль реализует основной интерфейс приложения, а также обрабатывает действия пользователя, такие как выбор изображения, преобразование в черно-белый формат и перемещение файла.

1. Инициализация окна:
   Использовал класс QWidget для создания базового окна приложения с фиксированным размером.
2. Создание интерфейса:
   В интерфейс добавлены:
   - Поле ввода для пути к файлу (QLineEdit), чтобы пользователь мог ввести или указать путь вручную.
   - Кнопки для выполнения действий: "Browse" для выбора изображения, "Convert to Grayscale" для преобразования изображения и "Move Image" для перемещения файла.
3. Логика обработки действий:
   - Выбор изображения: Метод browse_image() использует диалоговое окно для выбора изображения и отображает его.
   - Преобразование в черно-белый: Метод handle_grayscale() вызывает функцию convert_to_grayscale() из модуля image_processing.py, чтобы преобразовать изображение.
   - Перемещение изображения: Метод handle_move() использует функцию move_file() из модуля file_utils.py для перемещения изображения в новую директорию.Исходный код модуля представлен на рисунке 6.



```python
from PyQt5.QtWidgets import QWidget, QLabel, QPushButton, QFileDialog, QVBoxLayout, QHBoxLayout, QLineEdit, QMessageBox
from PyQt5.QtGui import QPixmap
from PyQt5.QtCore import Qt
from image_processing import convert_to_grayscale
from file_utils import move_file
import os

class MainWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Image Processor")
        self.setFixedSize(1200, 900)

        main_layout = QHBoxLayout()

        left_layout = QVBoxLayout()
        left_layout.setContentsMargins(20, 20, 20, 20)
        left_layout.setSpacing(15)

        self.path_input = QLineEdit()
        self.path_input.setPlaceholderText("Enter image path here")
        self.path_input.setFixedWidth(300)
        left_layout.addWidget(self.path_input)

        self.browse_button = QPushButton("Browse")
        self.browse_button.setFixedWidth(300)
        self.browse_button.clicked.connect(self.browse_image)
        left_layout.addWidget(self.browse_button)

        self.process_button = QPushButton("Convert to Grayscale")
        self.process_button.setFixedWidth(300)
        self.process_button.clicked.connect(self.handle_grayscale)
        left_layout.addWidget(self.process_button)

        self.move_button = QPushButton("Move Image")
        self.move_button.setFixedWidth(300)
        self.move_button.clicked.connect(self.handle_move)
        left_layout.addWidget(self.move_button)

        left_layout.addStretch()

        self.image_label = QLabel()
        self.image_label.setFixedSize(800, 800)
        self.image_label.setAlignment(Qt.AlignCenter)

        main_layout.addLayout(left_layout)
        main_layout.addWidget(self.image_label)

```

Рисунок 6 - Код модуля main_window.py

image_processing.py

Этот модуль выполняет обработку изображений, а именно преобразование изображения в черно-белый формат.

1. Подключил библиотеку Pillow, которая предоставляет функции для работы с изображениями.
2. Реализовал функцию convert_to_grayscale(), которая открывает файл изображения, преобразует его в черно-белый формат (L — режим яркости) и сохраняет в той же директории с добавлением суффикса _grayscale к имени файла.
3. Функция возвращает путь к обработанному файлу, чтобы результат можно было использовать в интерфейсе.

```
49          self.setLayout(main_layout)
50
51      def browse_image(self):
52          file_path, _ = QFileDialog.getOpenFileName(self, "Open Image", "", "Images (*.png *.xpm *.jpg)")
53          if file_path:
54              self.path_input.setText(file_path)
55              self.display_image(file_path)
56
57      def display_image(self, image_path):
58          pixmap = QPixmap(image_path)
59          self.image_label.setPixmap(pixmap.scaled(self.image_label.size(), Qt.KeepAspectRatio))
60
61      def handle_grayscale(self):
62          image_path = self.path_input.text()
63          if not image_path or not os.path.exists(image_path):
64              QMessageBox.warning(self, "Warning", "Please select a valid image file.")
65              return
66          try:
67              grayscale_path = convert_to_grayscale(image_path)
68              self.display_image(grayscale_path)
69              QMessageBox.information(self, "Success", f"Grayscale image saved as {grayscale_path}")
70          except Exception as e:
71              QMessageBox.critical(self, "Error", f"Failed to convert image: {e}")
72
73      def handle_move(self):
74          image_path = self.path_input.text()
75          if not image_path or not os.path.exists(image_path):
76              QMessageBox.warning(self, "Warning", "Please select a valid image file.")
77              return
78          new_dir = QFileDialog.getExistingDirectory(self, "Select Destination Folder")
79          if new_dir:
80              try:
81                  new_path = move_file(image_path, new_dir)
82                  QMessageBox.information(self, "Success", f"Image moved to {new_path}")
83              except Exception as e:
84                  QMessageBox.critical(self, "Error", f"Could not move image: {e}")
```

Рисунок 7 - Код модуля image_processing.py

ЗАКЛЮЧЕНИЕ

Практика в ООО "Малленом Системс" позволила мне улучшить навыки анализа и разработки программного обеспечения, а также получить опыт работы с современными инструментами и библиотеками Python. Работа над проектом помогла мне лучше понять процессы проектирования, тестирования и оптимизации программных продуктов.В ходе производственной практики мной был проведен всесторонний анализ программных продуктов и их компонентов. Работа включала ревьюирование кода, измерение характеристик производительности,использование инструментов анализа и сравнительный обзор применяемых технологий.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

UML - https://practicum.yandex.ru/blog/uml-diagrammy/

Пример измерения скорости используя Time -

https://www.geeksforgeeks.org/how-to-check-the-execution-time-of-python

Диаграммы - https://app.diagrams.net/

Работа с модулями Python - https://metanit.com/python/tutorial/2.10.php

Работа с библиотеками https://metanit.com/sharp/tutorial/3.46.php

ПРИЛОЖЕНИЯ

Вид программы представлен на Рисунке 8.

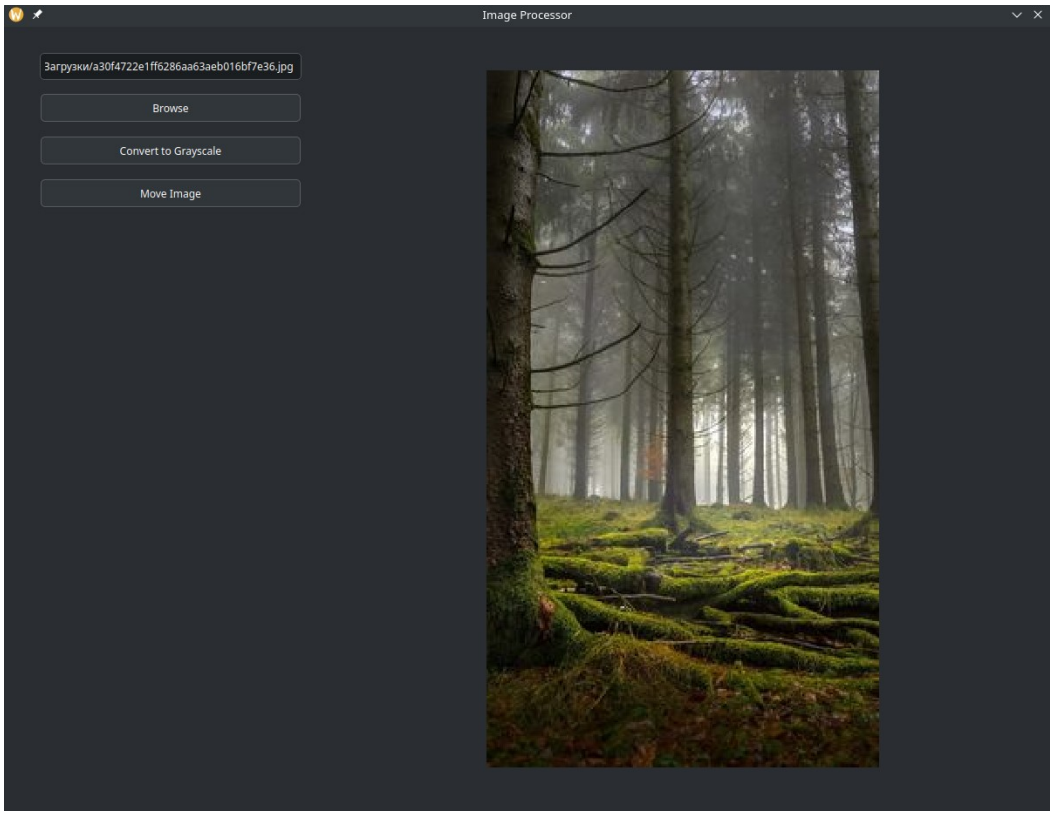


Рисунок 8 - Внешний вид программы