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-31

Богданова Варвара Юрьевна

подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

место практики ООО «Малленом Системс»

Период прохождения:

с «08» 12 2024 г.

по «21» 12 2024 г.

Руководитель практики от

Предприятия должность:

Южакова Н.В., специалист по кадрам

подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

МПРуководитель практики от

техникума: Материкова А.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 года

г. Череповец

2024

Оглавление

[Введение 3](#_Toc184983657)

[1. Общая характеристика компании 4](#_Toc184983658)

[1.1 О компании 4](#_Toc184983659)

[1.3 Направления деятельности: 4](#_Toc184983660)

[1.4 Разработки и бренды компании: 4](#_Toc184983661)

[1.5 Организационная структура 5](#_Toc184983662)

[1.6 Охрана труда 5](#_Toc184983663)

[1.7 Должностные обязанности техника 5](#_Toc184983664)

[2. Ревьюирование программных продуктов 7](#_Toc184983665)

[2.1 Ревьюирование программного кода в соответствии с технической документацией 7](#_Toc184983666)

[2.2 Измерение характеристик компонентов программного продукта 7](#_Toc184983667)

[2.3 Исследование созданного программного кода с использованием специализированных программных средств 7](#_Toc184983668)

[2.4 Сравнительный анализ программных продуктов и средств разработки 7](#_Toc184983669)

[3. Выполняемые задания 8](#_Toc184983670)

[Заключение 20](#_Toc184983671)

[Список использованных источников 20](#_Toc184983672)

[Приложение 1 20](#_Toc184983673)

[Приложение 2 21](#_Toc184983674)

[Приложение 3 23](#_Toc184983675)

[Приложение 4 25](#_Toc184983676)

# Введение

Место прохождения практики ООО «Малленом Системс»

Сроки прохождения с 08.12.2024 по 21.12.2024

Цель: освоение основного вида деятельности по направлению «Ревьюирование программных модулей» и формирование соответствующих общих и профессиональных компетенций.

Задачи:

1. Осуществить ревьюирование программного кода в соответствии с технической документацией
2. Выполнить измерение характеристик компонентов программного продукта
3. Произвести исследование созданного программного кода с использованием специализированных программных средств
4. Провести сравнительный анализ программных продуктов и средств разработки

# Общая характеристика компании

# О компании

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

1.2 Деятельность компании ООО «Малленом Системс»

Профиль компании – реализация наукоемких IT-проектов, направленных на повышение эффективности и безопасности промышленных предприятий в сфере транспорта, машиностроения, нефтегазовой, металлургической, пищевой, фармацевтической, алмазодобывающей, атомной промышленности и других отраслях. Компания осуществляет продажи по всей России, в страны СНГ и за рубеж

# 1.3 Направления деятельности:

* Визуальный контроль продукции на производстве
* Прослеживание продукции в алкогольной, фармацевтической и других отраслях промышленности
* Видеоконтроль и учет автомобильного и железнодорожного транспорта
* Автоматизация взвешивания автомобильного и железнодорожного транспорта
* Обнаружение и отслеживание людей, их действий и событий на видео
* Видеоаналитика с применением БПЛА (дронов)
* Заказная разработка систем компьютерного зрения и интеллектуальной обработки данных

# 1.4 Разработки и бренды компании:

* ВИСКОНТ – интеллектуальные системы машинного зрения для контроля качества и прослеживания продукции
* EYECONT – интеллектуальная система обнаружения и сопровождения людей на видео
* АВТОМАРШАЛ – система контроля доступа и учета автотранспорта на основе технологии распознавания автономеров
* VIRIS – адаптивная смарт-камера с высоким уровнем распознавания номеров автомобилей для контроля и учета автотранспорта, контроля нарушений ПДД и др. задач безопасности
* АВТОМАРШАЛ.ВЕСОВАЯ - аппаратно-программный комплекс, решающий полный перечень задач автоматизации автомобильных весовых
* АРДИС (ARDIS) – семейство решений по идентификации вагонов, управлению отгрузкой продукции ж/д транспортом, коммерческому осмотру вагонов, контролю передвижения и местонахождения вагонов, построенное на платформе системы распознавания номеров вагонов
* AVEDEX – программное обеспечение для анализа и подсчета транспортных и пешеходных потоков

# 1.5 Организационная структура

Линейная, есть отделы компании и в них есть руководители.

* Центр по развитию интеллектуальных систем, отдел разработки ПО. Проектирование, разработка, оптимизация ПО для клиентов компании.
* Производственно-технический отдел. Отдел с инженерами, которые проводят пусконаладочные работы на предприятиях, проектируют местонахождение оборудования на предприятии и устанавливают его.
* АУП (Административно-управленческий персонал). Руководство компании, которое формирует стратегии развития, управляет отделами, планирует деятельность предприятия, обеспечивает внешние коммуникации компании на выставках, в СМИ.
* Группа Маркетинга. Формирование маркетинговой стратегии компании, внутренний и внешний PR-компании, продвижение бренда и продуктов на рынке.
* Коммерческий отдел. Продажа продуктов компании заказчикам, поиск новых клиентов, участие в PR-продвижении компании.
* Отдел технической поддержки и контроля качества. Техническая поддержка пользователей и тестировка ПО на выявление ошибок и проблем.
* Отдел акселерационных и образовательных программ. Разработка и проведение обучающих курсов по машинному зрению и языку программирования, PR компании на рынке образовательных учреждений
* Отдел кадров. Управление персоналом компании, поиск, подбор, адаптация сотрудников, ведение кадрового документооборота, разработка стратегия развития персоналом предприятия
* Юридический отдел. Обработка всех документов в компании в соответствии с законодательством, взаимодействие с заказчиками и менеджерами по договорным обязательствам.
* Бухгалтерия. Ведение экономической деятельности предприятия, бухгалтерского учета, формирование бюджетов компании.
* •ОХР (общественно-хозяйственные рабочие). Поддержание чистоты, порядка на рабочих местах, ремонт, уборка служебных помещений.

# 1.6 Охрана труда

В компании "Малленом Системс" действует график работы 5/2, с 09:00 до 18:00. В штате компании есть отдельный специалист по охране труда, который проводит вводные инструктажи при приеме на работу и практике, а также занимается выдачей пропусков для пусконаладочных работ инженеров. В компании 20.09.2018 г. была проведена специальная оценка условий труда, согласно которой рабочие места, на территории которых установлены вредные производственные факторы, отсутствуют.

# 1.7 Должностные обязанности техника

* Выполняет работу по проведению необходимых технических расчетов;
* Осуществляет наладку, настройку, регулировку и опытную проверку оборудования и систем, следит за его исправным состоянием;
* Принимает участие в проведение экспериментов и испытаний;
* Принимает участие в разработке программ, инструкций и другой технической документации, в изготовлении макетов, а также в испытаниях и экспериментальных работах;
* Выполняет работу по сбору, обработке и накоплению исходных материалов, данных статистической отчетности, научно-технической информации;
* Составляет описания проводимых работ, необходимые спецификации, диаграммы, таблицы, графики и другую техническую документацию;
* Выполняет работу по оформлению плановой и отчетной документации, вносит необходимые изменения и исправления в техническую документацию в соответствии с решениями, принятыми при рассмотрении и обсуждении выполняемой работы;
* Систематизирует, обрабатывает и подготавливает данные для составления отчетов о работе;
* Принимает необходимые меры по использованию в работе современных технических средств.

# Ревьюирование программных продуктов

# Ревьюирование программного кода в соответствии с технической документацией

Ревьюирование программного кода в соответствии с технической документацией — это одна из профессиональных компетенций, которая входит в состав профессионального модуля «Ревьюирование программных продуктов».

Цель ревьюирования — повысить качество программного кода. В ходе этого процесса код подвергается проверке и критике, что помогает находить широкий класс ошибок и недоработок.

# Измерение характеристик компонентов программного продукта

Измерение характеристик программного продукта — это важная часть разработки, позволяющая оценить производительность приложения, его эффективность и использование ресурсов. В Python для этих задач можно использовать различные подходы.

* Измерение времени выполнения измеряется с использованием библиотеки time. При открытии изображения функция load\_image записывает текущее время перед началом выполнения операции и затем снова после завершения. Временной интервал между этими двумя ударами во времени и дает значение, которое мы интерпретируем как затраты времени на загрузку изображения (Приложение 3).
* Измерение использования памяти можно использовать функцию sys.getsizeof(), которая возвращает размер объекта в байтах. После открытия изображения выводятся размеры модулей sys и PIL.Image (Приложение 3).

# 2.3 Исследование созданного программного кода с использованием специализированных программных средств

# Сравнительный анализ программных продуктов и средств разработки

# Выполняемые задания

Задание 1. Создать GIT репозиторий для производственной практики. a. Report (Отчет) b. Src (Задания/Исходный код) c. Docs (Документы практики)

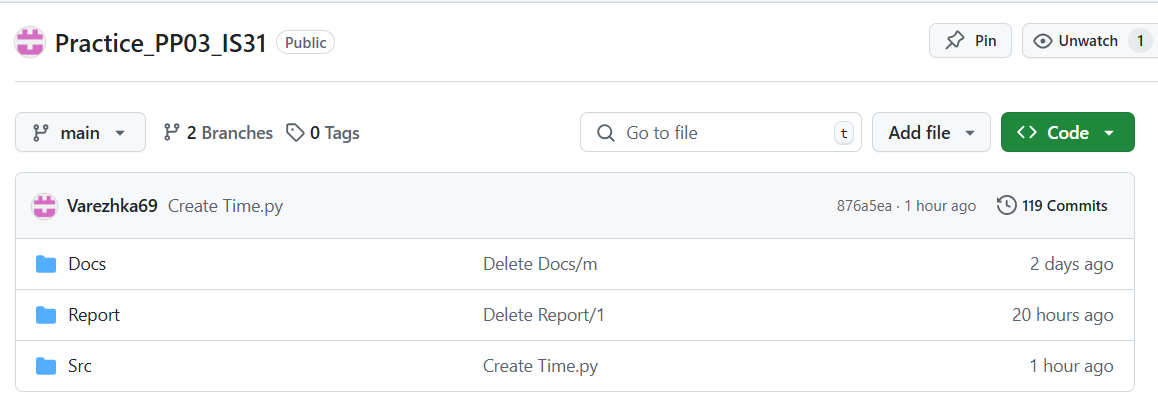


Рис.3.1. – Репозиторий

Задание 2. Разработать модули:

a. Модуль обработки и работы с изображениями. 1. Выдает информацию об изображении (Размер, Разрешение, Дата создания) 2. Переименование названия изображения.

1. Модуль ImageInfo предназначен для получения информации об изображениях, загруженных с использованием объекта image\_loader. Этот модуль предоставляет класс ImageInfo, который включает методы для извлечения размера, разрешения и даты создания изображения

import os

from datetime import datetime

class ImageInfo:

def \_\_init\_\_(self, image\_loader):

self.image\_loader = image\_loader

def get\_image\_info(self):

"""Возвращает информацию об изображении: размер, разрешение, дата создания."""

if self.image\_loader.image is None:

raise ValueError("Изображение не загружено.")

info = {

'Размер (байты)': os.path.getsize(self.image\_loader.image\_path),

'Разрешение': self.image\_loader.image.size, # (ширина, высота)

'Дата создания': self.get\_creation\_date()

}

return info

def get\_creation\_date(self):

"""Возвращает дату создания изображения."""

creation\_time = os.path.getctime(self.image\_loader.image\_path)

return datetime.fromtimestamp(creation\_time)

1. Модуль ImageLoader предназначен для загрузки изображений из файловой системы с использованием библиотеки PIL (Python Imaging Library, известной также как Pillow).

import os

from PIL import Image

class ImageLoader:

def \_\_init\_\_(self, image\_path):

self.image\_path = image\_path

self.image = None

self.load\_image()

def load\_image(self):

"""Загружает изображение и проверяет его наличие."""

if os.path.exists(self.image\_path):

self.image = Image.open(self.image\_path)

else:

raise FileNotFoundError(f"Файл {self.image\_path} не найден.")

1. Модуль ImageRename представляет собой класс ImageRenamer, который отвечает за переименование изображений. Он использует объект image\_loader, который загружает или изображение, чтобы упростить процесс переименования и обновления текущего пути к изображению.

import os

class ImageRenamer:

def \_\_init\_\_(self, image\_loader):

self.image\_loader = image\_loader

def rename\_image(self, new\_name):

"""Переименовывает изображение."""

new\_path = os.path.join(os.path.dirname(self.image\_loader.image\_path), new\_name)

os.rename(self.image\_loader.image\_path, new\_path)

self.image\_loader.image\_path = new\_path # Обновляем путь к изображению

self.image\_loader.load\_image() # Загружаем новое изображение

1. Модуль MainModul предназначен для загрузки изображения, получения информации о нем и изменения его имени.

from module\_image\_loader import ImageLoader

from module\_image\_info import ImageInfo

from module\_image\_renamer import ImageRenamer

def main():

image\_path = 'path/to/your/image.jpg' # Укажите путь к вашему изображению

image\_loader = ImageLoader(image\_path)

# Получаем информацию об изображении

image\_info = ImageInfo(image\_loader)

info = image\_info.get\_image\_info()

print(info)

# Переименовываем изображение

new\_name = 'new\_image\_name.jpg'

renamer = ImageRenamer(image\_loader)

renamer.rename\_image(new\_name)

print(f'Изображение переименовано в {new\_name}')

if \_\_name\_\_ == "\_\_main\_\_":

main()

b. Модуль взаимодействия с пользователем и формирование и хранений данных. 1. Отображается графический интерфейс для пользователя. 2. Пользователь задает путь к изображению, модуль передает данные модулю работы с изображением и выдает результат о данных изображения. Графический интерфейс для взаимодействия с пользователем: PyQT (приоритетнее) или Tkinter.

1. Модуль Gui создает простое приложение для отображения информации об изображениях с использованием библиотеки tkinter.

import tkinter as tk

from tkinter import filedialog, messagebox

from image\_handler import get\_image\_info

class ImageInfoApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Информация об изображении")

# Кнопка выбора изображения

self.btn\_select\_image = tk.Button(root, text="Выбрать изображение", command=self.load\_image)

self.btn\_select\_image.pack(pady=20)

# Метка для отображения информации о изображении

self.label\_info = tk.Label(root, text="", wraplength=300)

self.label\_info.pack(pady=20)

def load\_image(self):

# Открываем диалог выбора файла

file\_path = filedialog.askopenfilename(title="Выберите изображение",

filetypes=(("Image files", "\*.jpg;\*.jpeg;\*.png;\*.gif"),

("All files", "\*.\*")))

if file\_path:

self.display\_image\_info(file\_path)

def display\_image\_info(self, file\_path):

try:

info\_text = get\_image\_info(file\_path)

self.label\_info.config(text=info\_text)

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось получить информацию об изображении:\\n{e}")

1. Модуль ImageHandler использует библиотеку PIL (Python Imaging Library) для работы с изображениями. Основная функция данного модуля — извлечение и форматирование информации о изображении.

from PIL import Image

def get\_image\_info(file\_path)

# Открываем изображение и получаем его информацию

with Image.open(file\_path) as img:

width, height = img.size

format = img.format

# Возвращаем отформатированное сообщение с информацией об изображении

return f"Файл: {file\_path}\\nФормат: {format}\\nРазмер: {width}x{height} пикселей"

1. Модуль MainModul используется для создания графического интерфейса на базе библиотеки Tkinter. Он запускает приложение, которое предоставляет функционал для работы с изображениями.

import tkinter as tk

from gui import ImageInfoApp

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = ImageInfoApp(root)

root.mainloop()

Задание 3. Провести обратное проектирование используя графический язык UML. Создать и описать диаграммы Компонентов, Сценариев использования, Последовательностей, Деятельности.

a. Модуль обработки и работы с изображениями.

1. Диаграмма компонентов

Пользовательский интерфейс: содержит элементы управления, позволяющие пользователю загружать изображения, получать их информацию и переименовывать их. Например, кнопки и текстовые поля. Обработчик изображений (ImageProcessor): является центральным элементом системы, который содержит логику для обработки изображений. Система файлов: содержит функции для работы с файловой системой, такие как проверка существования файла, получение его размера и даты создания, а также переименование файла

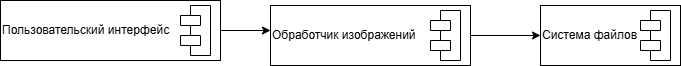


Рис.3.2. – Диаграмма компонентов

1. Диаграмма сценариев использования

Пользователь взаимодействует с системой, инициируя процесс загрузки изображения и получая информацию о нем. Система (ImageProcessor) отвечает за выполнение задач: загрузку изображения, получение информации об изображении и изменением имени файла.

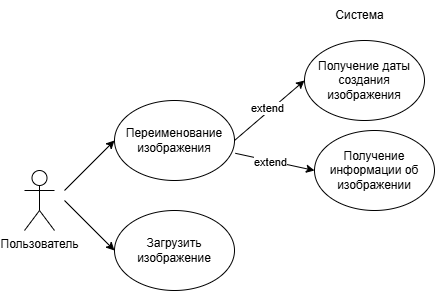


Рис.3.3. – Диаграмма сценариев использования

1. Диаграмма последовательностей

Пользователь: Инициирует действия с изображением. Интерфейс: Представляет уровень взаимодействия пользователя с системой, обрабатывает ввод и вывод информации. Система: Класс ImageProcessor, который отвечает за обработку изображений



Рис.3.4. – Диаграмму последовательностей

1. Диаграмма деятельности

Пользователь: Начинает процесс, указывая путь к изображению. Фронтенд: Отправляет запрос на бэкенд. Бэкенд: ImageProcessor: Инициализирует процесс обработки изображения. Загружает изображение. Проверяет наличие файла. Получает информацию об изображении: Если изображение загружено, бэкенд извлекает метаданные (размер, разрешение, дата создания). Возвращает информацию на фронтенд. Фронтенд: Отображает информацию пользователю. Пользователь: указывает новое имя. Фронтенд: Отправляет запрос на бэкенд на переименование изображения. Бэкенд: ImageProcessor: Выполняет процесс переименования. Возвращает результат операции. Фронтенд: Отображает результат операции пользователю.



Рис.3.5. – Диаграмма деятельности

b. Модуль взаимодействия с пользователем и формирование и хранений данных.

1. Диаграмма компонентов

Пользователь взаимодействует с компонентом пользовательского интерфейса (кнопка) для выбора изображения. После выбора изображения происходит взаимодействие с компонентом файлового диалога, который возвращает путь к файлу. Этот путь передаётся в обработчик изображений (PIL), который открывает изображение и извлекает информацию о нём. Эта информация отображается в метке компонента пользовательского интерфейса. В случае ошибки компоненты пользовательского интерфейса и обработки ошибок взаимодействуют для информирования пользователя об ошибках.

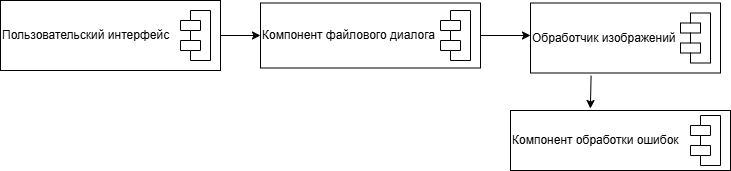


Рис.3.6. – Диаграмма компонентов

1. Диаграмма сценариев использования

Пользователь нажимает кнопку "Выбрать изображение". Открывается диалоговое окно для выбора файла изображения. Пользователь выбирает изображение. Приложение обрабатывает выбранный файл. Приложение открывает выбранное изображение и извлекает информацию о его размере (ширина, высота) и формате. Если произошла ошибка (например, файл не является изображением или поврежден), приложение отображает сообщение об ошибке.



Рис.3.7. – Диаграмма сценариев использования

1. Диаграмма последовательностей

Пользователь инициирует действие, нажимая кнопку "Выбрать изображение". Интерфейс получает это событие и вызывает метод load\_image(). Система отображает диалог выбора файла. Пользователь выбирает файл изображения и нажимает "Открыть". Система открывает изображение с помощью PIL и получает его свойства - высоту, ширину и формат.Система формирует строку с информацией. В случае ошибки система отображает сообщение об ошибке. Интерфейс обновляет отображение информации в приложении.

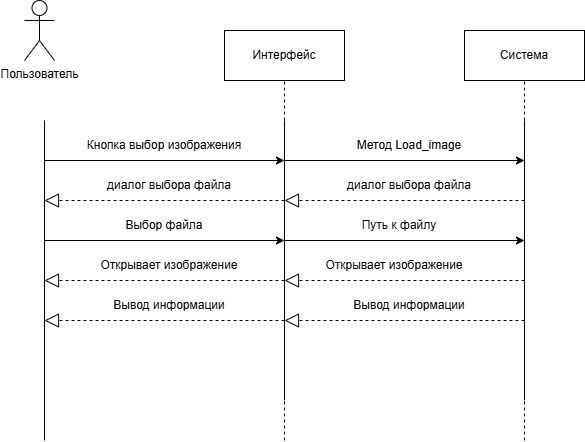


Рис.3.8. – Диаграмма последовательностей

1. Диаграмма деятельности

Пользователь: инициирует процесс, нажав кнопку. Фронтенд: отвечает за интерфейс и взаимодействие с пользователем, включая отображение диалоговых окон и меток. Бэкенд (Система): выполняет бизнес-логику обработки изображений, включая загрузку и информацию о файле.

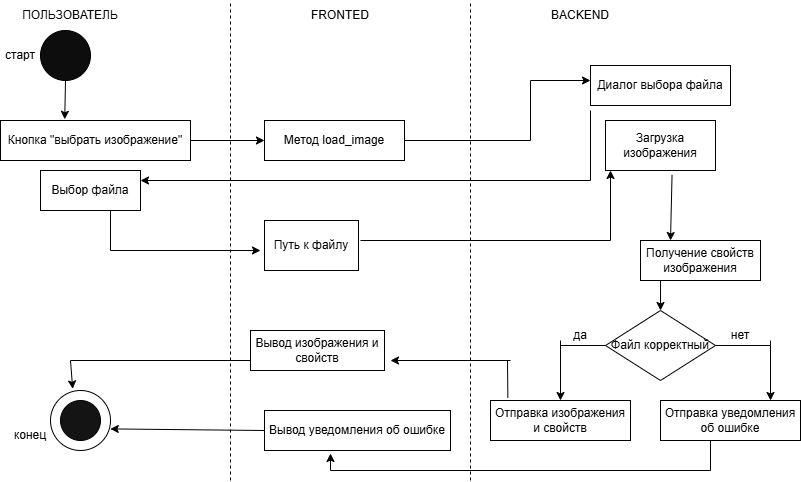


Рис.3.9. – Диаграмма деятельности

Задание 4. Провести скоростные показатели программы, и прочие характеристики (размеры). В скоростных показателях измерить каждую операцию в модулях.

Таб.3.1 – Скоростные показатели программ

|  |  |
| --- | --- |
| Модуль обработки и работы с изображениями | Время выполнения load\_image: 0.021416 секунд. Время выполнения get\_creation\_date: 0.000077 секунд. Время выполнения get\_image\_info: 0.012246 секунд |
| Модуль ImageInfo | Время получения размера (с): 0.00015687942504882812, 'Время получения разрешения (с): 9.5367431640625e-07, Время получения даты создания (с): 6.771087646484375e-05 |
| Модуль ImageLoader | Изображение загружено за 0.039974 секунд. |
| Модуль ImageRename | Время на создание нового пути: 0.000041 секунд. Время на переименование файла: 0.013678 секунд. Время на обновление пути и загрузку изображения: 0.000004 секунд. |
| Модуль MainModul | Время выполнения load\_image: 0.021446 секунд. Время выполнения get\_creation\_date: 0.000068 секунд. Время выполнения get\_image\_info: 0.012225 секунд. |
| Модуль взаимодействия с пользователем и формирование и хранений данных | Время загрузки изображения 0,022676 секунд. |
| Модуль ImageHandler | Время открытия изображения: 0,028281. Время получения информации: 0,027827. |
| Модуль Gui | Время выбора файла: 5,5707 секунд. |
| Модуль MainModul | Время выполнения: 2,947237 секунд. |

Таб.3.2. – Размеры программ

|  |  |
| --- | --- |
| Модуль обработки и работы с изображениями | Размер (байты): 876340. Размер объекта get\_image\_info в памяти: 184 байт. |
| Модуль ImageInfo | Размер объекта 48 байт |
| Модуль ImageLoader | Размер объекта 48 байт. |
| Модуль ImageRename | Размер объекта 48 байт. |
| Модуль MainModul | Размер объекта 76640 байт. |
| Модуль взаимодействия с пользователем и формирование и хранений данных | Размер: 876254. Размер модуля sys 72 байт. Размер модуля PIL\_Image 72 байт. |
| Модуль ImageHandler | Размер объекта 88 байт. |
| Модуль Gui | Размер объекта 88 байт. |
| Модуль MainModul | Размер объекта 88 байт. |

# Заключение

В ходе практики были получены следующие практические навыки:

1. Осуществление ревьюирования программного кода в соответствии с технической документацией
2. Выполнение измерений характеристик компонентов программного продукта
3. Произведение исследования созданного программного кода с использованием специализированных программных средств
4. Проведение сравнительного анализа программных продуктов и средств разработки

Также были сформированы общие и профессиональные компетенции, такие как умение выбирать способы решения задач профессиональной деятельности, осуществлять поиск, анализ и интерпретацию информации.

# Список использованных источников

1. <https://metanit.com/python/tutorial/2.10.php>
2. <https://metanit.com/sharp/tutorial/3.46.php>
3. <https://app.diagrams.net/>
4. <https://www.mallenom.ru/?ysclid=m4l2ditzeb835497887>
5. <https://practicum.yandex.ru/blog/uml-diagrammy/>
6. <https://www.geeksforgeeks.org/how-to-check-the-execution-time-of-python-script/>

# Приложение 1

Модуль обработки и работы с изображениями

import os

from PIL import Image

from datetime import datetime

class ImageProcessor:

def \_\_init\_\_(self, image\_path):

self.image\_path = image\_path

self.image = None

self.load\_image()

def load\_image(self):

"""Загружает изображение и проверяет его наличие."""

if os.path.exists(self.image\_path):

self.image = Image.open(self.image\_path)

else:

raise FileNotFoundError(f"Файл {self.image\_path} не найден.")

def get\_image\_info(self):

"""Возвращает информацию об изображении: размер, разрешение, дата создания."""

if self.image is None:

raise ValueError("Изображение не загружено.")

info = {

'Размер (байты)': os.path.getsize(self.image\_path),

'Разрешение': self.image.size, # (ширина, высота)

'Дата создания': self.get\_creation\_date()

}

return info

def get\_creation\_date(self):

"""Возвращает дату создания изображения."""

creation\_time = os.path.getctime(self.image\_path)

return datetime.fromtimestamp(creation\_time)

def rename\_image(self, new\_name):

"""Переименовывает изображение."""

new\_path = os.path.join(os.path.dirname(self.image\_path), new\_name)

os.rename(self.image\_path, new\_path)

self.image\_path = new\_path # Обновляем путь к изображению

self.load\_image() # Загружаем новое изображение

# Приложение 2

Модуль взаимодействия с пользователем и формирование и хранений данных

import tkinter as tk

from tkinter import filedialog, messagebox

from PIL import Image

class ImageInfoApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Информация об изображении")

# Кнопка выбора изображения

self.btn\_select\_image = tk.Button(root, text="Выбрать изображение", command=self.load\_image)

self.btn\_select\_image.pack(pady=20)

# Метка для отображения информации о изображении

self.label\_info = tk.Label(root, text="", wraplength=300)

self.label\_info.pack(pady=20)

def load\_image(self):

# Открываем диалог выбора файла

file\_path = filedialog.askopenfilename(title="Выберите изображение",

filetypes=(("Image files", "\*.jpg;\*.jpeg;\*.png;\*.gif"),

("All files", "\*.\*")))

if file\_path:

self.display\_image\_info(file\_path)

def display\_image\_info(self, file\_path):

try:

# Открываем изображение и получаем его информацию

with Image.open(file\_path) as img:

width, height = img.size

format = img.format

# Отображаем информацию

info\_text = f"Файл: {file\_path}\\nФормат: {format}\\nРазмер: {width}x{height} пикселей"

self.label\_info.config(text=info\_text)

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось получить информацию об изображении:\\n{e}")

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = ImageInfoApp(root)

root.mainloop()

# Приложение 3

Код для измерения скоростных показателей и размера Модуля обработки и работы с изображениями

import os

import sys

from PIL import Image

from datetime import datetime

import time

class ImageProcessor:

def \_\_init\_\_(self, image\_path):

self.image\_path = image\_path

self.image = None

self.load\_image()

def measure\_time(func):

"""Декоратор для измерения времени выполнения функции."""

def wrapper(\*args, \*\*kwargs):

start\_time = time.time()

result = func(\*args, \*\*kwargs)

end\_time = time.time()

execution\_time = end\_time - start\_time

print(f"Время выполнения {func.\_\_name\_\_}: {execution\_time:.6f} секунд")

return result

return wrapper

@measure\_time

def load\_image(self):

"""Загружает изображение и проверяет его наличие."""

if os.path.exists(self.image\_path):

self.image = Image.open(self.image\_path)

else:

raise FileNotFoundError(f"Файл {self.image\_path} не найден.")

@measure\_time

def get\_image\_info(self):

"""Возвращает информацию об изображении: размер, разрешение, дата создания."""

if self.image is None:

raise ValueError("Изображение не загружено.")

info = {

'Размер (байты)': os.path.getsize(self.image\_path),

'Разрешение': self.image.size, # (ширина, высота)

'Дата создания': self.get\_creation\_date()

}

print(f"Размер объекта get\_image\_info в памяти: {sys.getsizeof(info)} байт")

return info

@measure\_time

def get\_creation\_date(self):

"""Возвращает дату создания изображения."""

creation\_time = os.path.getctime(self.image\_path)

return datetime.fromtimestamp(creation\_time)

@measure\_time

def rename\_image(self, new\_name):

"""Переименовывает изображение."""

new\_path = os.path.join(os.path.dirname(self.image\_path), new\_name)

os.rename(self.image\_path, new\_path)

self.image\_path = new\_path # Обновляем путь к изображению

self.load\_image() # Загружаем новое изображение

# Приложение 4

Код для измерения скоростных показателей и размера Модуля взаимодействия с пользователем и формирование и хранений данных

import tkinter as tk

from tkinter import filedialog, messagebox

from PIL import Image

import time

import sys

class ImageInfoApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Информация об изображении")

# Кнопка выбора изображения

self.btn\_select\_image = tk.Button(root, text="Выбрать изображение", command=self.load\_image)

self.btn\_select\_image.pack(pady=20)

# Метка для отображения информации о изображении

self.label\_info = tk.Label(root, text="", wraplength=300)

self.label\_info.pack(pady=20)

def load\_image(self):

# Открываем диалог выбора файла

file\_path = filedialog.askopenfilename(title="Выберите изображение",

filetypes=(("Image files", "\*.jpg;\*.jpeg;\*.png;\*.gif"),

("All files", "\*.\*")))

if file\_path:

self.display\_image\_info(file\_path)

def display\_image\_info(self, file\_path):

# Измеряем время открытия изображения

start\_time = time.time()

try:

# Открываем изображение и получаем его информацию

with Image.open(file\_path) as img:

width, height = img.size

img\_format = img.format

loading\_time = time.time() - start\_time

info\_text = f"Файл: {file\_path}\\nФормат: {img\_format}\\nРазмер: {width}x{height} пикселей\\n" \

f"Время загрузки: {loading\_time:.6f} секунд"

self.label\_info.config(text=info\_text)

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось получить информацию об изображении:\\n{e}")

# Выводим информацию о размерах использованных библиотек

self.print\_sys\_info()

def print\_sys\_info(self):

# Выводим размеры модулей sys и PIL

print(f"Размер модуля sys: {sys.getsizeof(sys)} байт")

try:

from PIL import Image

print(f"Размер модуля PIL.Image: {sys.getsizeof(Image)} байт")

except ImportError:

print("Модуль PIL не найден")

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = ImageInfoApp(root)

root.mainloop()