СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Производственная практика была пройдена в период с 17 ноября 2025 года по 30 ноября 2025 года в организации ООО «Малленом Системс». Основными задачами в ходе ее прохождения являлись осуществление ревьюирования программного кода на соответствие технической документации, выполнение измерений характеристик компонентов программного продукта для проверки их соответствия установленным критериям, исследование созданного программного кода с применением специализированных программных средств с целью выявления ошибок и отклонений от заданных алгоритмов, а также проведение сравнительного анализа программных продуктов и средств разработки для определения наилучшего решения согласно критериям технического задания.

Целью прохождения практики выступало всестороннее освоение и практическое закрепление компетенций в области обеспечения качества программного обеспечения. Данная цель достигалась через активное участие в полном цикле контроля качества, включающем статический анализ кода, валидацию архитектурных решений, динамическое тестирование компонентов, сравнительную оценку инструментария и формирование обоснованных предложений по оптимизации продукта в соответствии с требованиями технической документации.

1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРЕДПРИЯТИЯ (ОРГАНИЗАЦИИ)

1.1. Организационная структура предприятия

Общество с ограниченной ответственностью «Малленом Системс» функционирует на основе функциональной организационной структуры, которая адаптирована для реализации сложных научно-технических проектов. В структуре компании выделяются несколько ключевых подразделений. Центральным звеном является отдел исследований и разработок, который включает в себя команды по компьютерному зрению, машинному обучению, алгоритмистов и бэкенд-разработчиков. Данный отдел отвечает за создание и внедрение информационных технологий. Проектный офис управляет полным жизненным циклом проектов, начиная от предпродажной подготовки и планирования и заканчивая внедрением и сдачей результатов заказчику. Проектные менеджеры в рамках этого офиса координируют деятельность всех отделов по конкретным контрактам. Отдел тестирования и обеспечения качества обеспечивает высокий уровень выпускаемых программных продуктов посредством проведения функционального, интеграционного и нагрузочного тестирования, тесно взаимодействуя с отделом разработки на всех этапах. Отдел системной интеграции и технической поддержки занимается развертыванием решений у заказчика, интеграцией с существующей инфраструктурой, а также предоставляет техническую поддержку и консультации. Коммерческий отдел отвечает за маркетинг, продажи и работу с клиентами. Бухгалтерия и администрация обеспечивают функционирование финансовой и хозяйственной деятельности компании. Такая структура позволяет эффективно распределять ресурсы и поддерживать высокий уровень специализации сотрудников.

1.2. Внутренний распорядок работы предприятия, охрана труда ИТ-специалистов

В компании «Малленом Системс» установлен стандартный рабочий график с девяти часов утра до шести часов вечера с понедельника по пятницу, с предусмотренным часовым перерывом на обед. Для сотрудников отдела исследований и разработок действует гибкий график и возможность удаленной работы, что связано со спецификой их творческой и интеллектуальной деятельности. В сфере охраны труда и техники безопасности для ИТ-специалистов соблюдается ряд важных положений. Рабочие места организованы с учетом эргономических требований и оснащены регулируемой мебелью, мониторами с антибликовым покрытием и поддержкой высокого разрешения. Сотрудникам рекомендовано выполнение перерывов для снижения нагрузки на зрительный аппарат и опорно-двигательную систему. В области электробезопасности все оборудование проходит регулярные проверки на соответствие установленным нормам, а использование несертифицированной техники запрещено. Помещения оснащены системами пожарной сигнализации и первичными средствами пожаротушения, с персоналом проводятся регулярные инструктажи. Руководство компании уделяет внимание созданию благоприятного психологического климата, минимизации стрессовых факторов и профилактике профессионального выгорания, в том числе через использование гибкого графика и организацию корпоративных мероприятий. Все сотрудники подписывают соглашение о неразглашении коммерческой тайны и строгом соблюдении правил информационной безопасности.

1.3. Должностные инструкции ИТ-специалистов предприятия

На предприятии действуют четко определенные должностные инструкции, которые регламентируют зоны ответственности и предъявляемые требования к квалификации сотрудников. Основные должности в техническом

департаменте включают инженера-программиста, в обязанности которого входит разработка, тестирование и отладка программных модулей, написание технической документации, участие в код-ревью и интеграция компонентов системы. От данного специалиста требуются знания языков программирования, таких как Python и C#, опыт работы с фреймворками и библиотеками, понимание принципов объектно-ориентированного программирования, алгоритмов и структур данных. Специалист по обеспечению качества занимается разработкой тестовых планов и сценариев, проведением ручного и автоматизированного тестирования, составлением отчетов об ошибках и регрессионным тестированием. Он должен владеть методологиями тестирования, иметь опыт работы с системами отслеживания ошибок и навыки написания автотестов. Системный архитектор отвечает за проектирование высокоуровневой архитектуры программных систем, выбор технологического стека, техническое руководство проектами и контроль за соблюдением архитектурных стандартов. Для этой должности необходимы глубокие знания в области паттернов проектирования, микросервисной и монолитной архитектуры, опыт масштабирования систем и знания различных систем управления базами данных. Инженер-проектировщик в области DevOps и SRE выполняет задачи по настройке процессов непрерывной интеграции и доставки, управлению облачной и локальной инфраструктурой, а также мониторингу производительности систем. Требованиями к нему являются опыт администрирования операционных систем Linux, знание систем контейнеризации, оркестрации и инструментов мониторинга.

2 РЕВЬЮИРОВАНИЕ ПРОГРАММНЫХ ПРОДУКТОВ

2.1. Ревьюирование программного кода в соответствии с технической документацией

Ревьюирование программного кода представляет собой систематический процесс проверки исходного кода, осуществляемый разработчиками с целью выявления дефектов, улучшения качества кода и обеспечения его соответствия требованиям технической документации. Данный процесс является критически важным элементом обеспечения качества программного обеспечения и проводится до этапа тестирования. Основной задачей ревьюирования является обнаружение ошибок в логике, нарушений стандартов кодирования, потенциальных уязвимостей безопасности, а также расхождений между реализованной функциональностью и спецификациями, закрепленными в техническом задании, проектной документации или пользовательских историях. Эффективное ревьюирование способствует не только снижению количества дефектов, но и распространению знаний о проекте среди членов команды, унификации стиля программирования и повышению сопровождаемости кодовой базы.

2.2. Измерение характеристик компонент программного продукта

Измерение характеристик компонент программного продукта является неотъемлемой частью процесса контроля качества и управления разработкой. Данная деятельность направлена на получение количественных метрик, объективно отражающих различные аспекты качества программного обеспечения. К таким метрикам относятся, но не ограничиваются ими, следующие: цикломатическая сложность, которая количественно оценивает структурную сложность программы и влияет на тестируемость и сопровождаемость; количество строк кода; степень покрытия кода тестами; объем потребляемой памяти и время отклика системы; коэффициент повторного использования кода; а также количество обнаруженных дефектов на тысячу строк кода. Анализ этих метрик позволяет принимать обоснованные

решения о необходимости рефакторинга, оценивать риски, связанные с интеграцией отдельных модулей, и прогнозировать трудозатраты на дальнейшую поддержку и развитие продукта.

2.3. Исследование созданного программного кода с использованием специализированных программных средств

Исследование созданного программного кода с привлечением специализированных программных средств представляет собой автоматизированный или полуавтоматизированный анализ, направленный на выявление скрытых дефектов, слабых мест и отклонений от стандартов. В отличие от ручного ревью, данный подход использует инструменты статического и динамического анализа. Статические анализаторы, такие как SonarQube, ESLint, Pylint или Checkstyle, выполняют проверку исходного кода без его запуска, выявляя синтаксические ошибки, нарушения стиля, потенциальные утечки памяти и сложные для поддержки конструкции. Динамические анализаторы, напротив, исследуют поведение программы во время ее выполнения, позволяя обнаружить ошибки времени выполнения, проблемы с производительностью и конкурентностью. Использование этих средств значительно повышает эффективность проверки, обеспечивая постоянный контроль качества на протяжении всего жизненного цикла разработки.

2.4. Сравнительный анализ программных продуктов и средств разработки

Сравнительный анализ программных продуктов и средств разработки представляет собой методологическую процедуру выбора оптимальных технологических решений на основе объективного сопоставления их характеристик. Проведение такого анализа является необходимым условием для принятия взвешенных архитектурных и управленческих решений. Процесс анализа, как правило, включает определение набора критериев, значимых для конкретного проекта. Такими критериями могут являться

функциональная полнота, производительность, масштабируемость, безопасность, стоимость владения, качество документации, активность сообщества и доступность квалифицированных кадров. В рамках сравнения средств разработки оцениваются интегрированные среды разработки, системы управления версиями, фреймворки для тестирования и инструменты непрерывной интеграции. Результатом сравнительного анализа является обоснованная рекомендация по выбору программного продукта или инструментария, который в наибольшей степени соответствует техническим и бизнес-требованиям проекта, что в конечном итоге способствует созданию качественного и конкурентоспособного программного обеспечения.

3 ВЫПОЛНЯЕМЫЕ ЗАДАНИЯ

3.1. Ревьюирование программного кода в соответствии с технической документацией

В рамках производственной практики был разработан программный продукт "Утилита обработки изображений", состоящий из двух основных модулей: модуля обработки изображений и модуля пользовательского интерфейса. Ревьюирование программного кода проводилось на основе технических требований, изложенных в задании. Библиотека обработки изображений, реализованная в модуле processing.py, была оформлена как отдельный пакет Python с четко определенным классом ImageProcessor, что полностью соответствует требованию о модульности. Код модуля был проверен на соответствие функциональным требованиям, включая операции изменения размера, применения фильтров повышения резкости и выделения контуров, получения информации об изображении и его сохранения. В модуле пользовательского интерфейса ui.py был реализован графический интерфейс, соответствующий предоставленному макету, с поддержкой загрузки изображений через диалоговое окно и методом drag-and-drop, отображением исходного и обработанного изображения, а также возможностью задания параметров обработки. Архитектура приложения, основанная на разделении логики и представления, соответствует рекомендуемому паттерну MVC. Проверка показала, что код корректно реализует всю заявленную функциональность, описанную в техническом задании.

3.2. Измерение характеристик компонент программного продукта

Для измерения характеристик компонентов программного продукта был проведен ряд тестов и замеров. С целью оценки производительности основных операций в класс ImageProcessor был добавлен декоратор measure_time, который фиксирует время выполнения каждого метода. Результаты замеров продемонстрировали приемлемую скорость работы операций загрузки, обработки и сохранения изображений стандартных размеров. Для проверки

функциональной корректности модуля обработки был разработан набор модульных тестов в файле test_processing.py, включающий тесты на загрузку изображения, получение информации, изменение размера, применение фильтров, сохранение и ведение журнала действий. Все тесты были успешно пройдены, что подтверждает соответствие модуля обработки заданным критериям качества. Визуальные результаты обработки, а также вывод информации об изображении в пользовательском интерфейсе также были проверены и признаны корректными. Скриншот результатов успешного выполнения модульных тестов (См. Рисунок 1 — Результат тестов модульного тестирования). Скриншот результата замера скорости выполнения операций (См. Рисунок 2 — Результат теста скорости выполнения операций).

3.3. Исследование созданного программного кода с использованием специализированных программных средств

Исследование созданного программного кода проводилось с использованием специализированных средств тестирования и анализа. Для автоматизации модульного тестирования применялся фреймворк pytest, который позволил проверить корректность работы всех ключевых функций модуля обработки изображений. Написание тестов способствовало выявлению и устранению потенциальных ошибок на ранних этапах разработки. Для анализа архитектуры и документирования проекта был использован графический язык UML. В рамках обратного проектирования были разработаны диаграммы, наглядно демонстрирующие структуру и поведение системы: UML-диаграмма структуры модулей (См. Рисунок 3 — UML-диаграмма структуры модулей), диаграмма компонентов (См. Рисунок 4 — Диаграмма компонентов), диаграмма сценариев использования (См. Рисунок 5 — Диаграмма сценариев использования), диаграмма последовательности для операции обработки изображения (См. Рисунок 6 — Диаграмма последовательности для операций обработки изображения) и диаграмма деятельности (См. Рисунок 7 — Диаграмма деятельности) Эти диаграммы

обеспечили лучшее понимание взаимодействия между компонентами системы и соответствия реализованной архитектуры поставленным задачам.

3.4. Сравнительный анализ программных продуктов и средств разработки

В процессе разработки был проведен сравнительный анализ программных продуктов и средств разработки для обоснования выбранного технологического стека. В качестве языка программирования был выбран Python ввиду его высокой производительности разработки, богатой экосистемы библиотек и кроссплатформенности. Для создания графического интерфейса был использован фреймворк PyQt6, который был предпочтен Tkinter благодаря своей мощи, поддержке современных функций, таких как drag-and-drop, возможностим глубокой стилизации и отличной документации. Для обработки изображений была выбрана библиотека Pillow (PIL), являющаяся стандартом де-факто в экосистеме Python для работы с изображениями и поддерживающая широкий спектр форматов и операций. Логирование операций реализовано с помощью стандартного модуля logging, а для хранения истории действий используется формат JSON, что позволило обойтись без привлечения внешних систем управления базами данных и минимизировать зависимости. Выбранный стек технологий в полной мере соответствует функциональным и техническим требованиям проекта, что подтверждается работоспособностью и стабильностью конечного программного продукта.

ЗАКЛЮЧЕНИЕ

В ходе прохождения производственной практики были успешно выполнены все поставленные задачи и достигнута цель, заключавшаяся в освоении и практическом закреплении компетенций в области обеспечения качества программного обеспечения. Практическая деятельность включала активное участие в полном цикле разработки и контроля качества программного продукта.

Была изучена организационная структура и деятельность предприятия ООО «Малленом Системс». В рамках практики разработана программа «Утилита обработки изображений», состоящая из двух модулей: библиотеки обработки изображений и графического интерфейса пользователя. Для данного программного продукта проведено ревьюирование кода, измерение характеристик компонентов, исследование кода с использованием специализированных средств тестирования и анализа, а также выполнен сравнительный анализ примененных средств разработки.

В результате проведенной работы были освоены практические навыки статического и динамического анализа кода, модульного тестирования, измерения производительности, проектирования архитектуры приложения и документирования проектных решений с использованием языка UML. Полученный опыт способствовал углублению понимания процессов обеспечения качества программного обеспечения и методов разработки, соответствующих современным отраслевым стандартам.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лутц, М. Изучаем Python, 5-е изд. / М. Лутц. – СПб. : Символ-Плюс, 2022.

2. Документация по Python 3.12 [Электронный ресурс]. – Режим доступа: https://docs.python.org/3/ (дата обращения: 25.11.2025).

3. Документация по фреймворку PyQt6 [Электронный ресурс]. – Режим доступа: https://www.riverbankcomputing.com/static/Docs/PyQt6/ (дата обращения: 25.11.2025).

4. Документация по библиотеке Pillow (PIL) [Электронный ресурс]. – Режим доступа: https://pillow.readthedocs.io/en/stable/ (дата обращения: 25.11.2025).

5. Яндекс Практикум. UML-диаграммы: виды, примеры и применение [Электронный ресурс]. – Режим доступа: https://practicum.yandex.ru/blog/uml-diagrammy/ (дата обращения: 25.11.2025).

6. GeeksforGeeks. How to check the execution time of Python script? [Электронный ресурс]. – Режим доступа: https://www.geeksforgeeks.org/python/how-to-check-the-execution-time-of-python-script/ (дата обращения: 25.11.2025).

7. Техническое задание на разработку программных модулей в рамках производственной практики ПП.03. – ООО «Малленом Системс», 2025.

```
test_processing.py::test_load_image PASSED                                                                                [ 12%]
test_processing.py::test_load_nonexistent PASSED                                                                          [ 25%]
test_processing.py::test_get_info PASSED                                                                                  [ 37%]
test_processing.py::test_transform_resize PASSED                                                                          [ 50%]
test_processing.py::test_transform_filters PASSED                                                                         [ 62%]
test_processing.py::test_save_image PASSED                                                                                [ 75%]
test_processing.py::test_log_action PASSED                                                                                [ 87%]
test_processing.py::test_transform_no_resize PASSED                                                                       [100%]

================================================== 8 passed in 0.31s ===================================================
```

Рисунок 1 — Результат тестов модульного тестирования

```
I:\Praktika\PP03\Src>main.py
_load_history выполнена за 0.02ms
__init__ выполнена за 0.08ms
load выполнена за 17.57ms
get_info выполнена за 0.01ms
transform выполнена за 0.44ms
log_action выполнена за 0.35ms
transform выполнена за 15.09ms
transform выполнена за 0.45ms
transform выполнена за 11.94ms
transform выполнена за 0.47ms
transform выполнена за 12.04ms
transform выполнена за 0.48ms
transform выполнена за 12.01ms
transform выполнена за 0.47ms
transform выполнена за 12.04ms
transform выполнена за 0.56ms
transform выполнена за 11.98ms
transform выполнена за 0.49ms
transform выполнена за 12.21ms
transform выполнена за 0.49ms
transform выполнена за 11.95ms
transform выполнена за 0.46ms
transform выполнена за 12.08ms
transform выполнена за 0.45ms
save выполнена за 55.86ms
log_action выполнена за 0.29ms
```

Рисунок 2 — Результат теста скорости выполнения операций

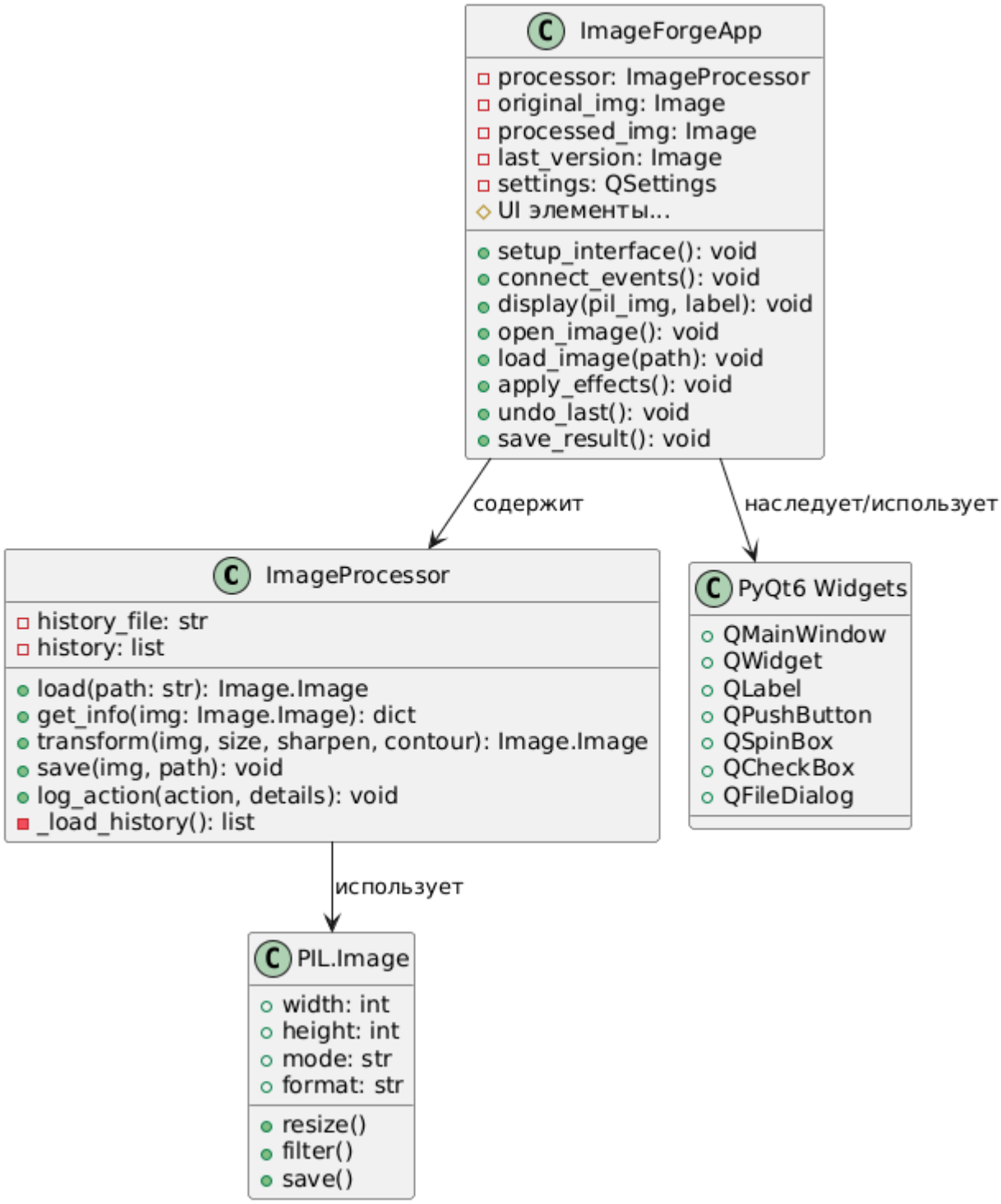

Рисунок 3 — UML-диаграмма структуры модулей

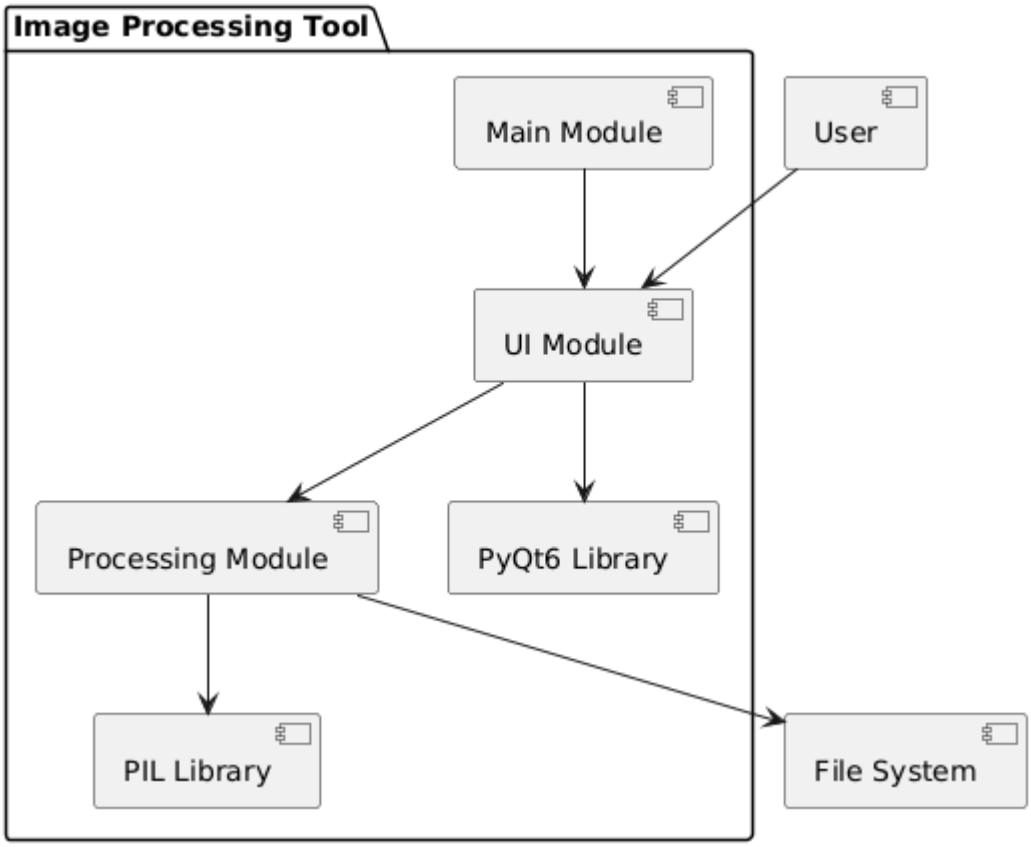

Рисунок 4 — Диаграмма компонентов

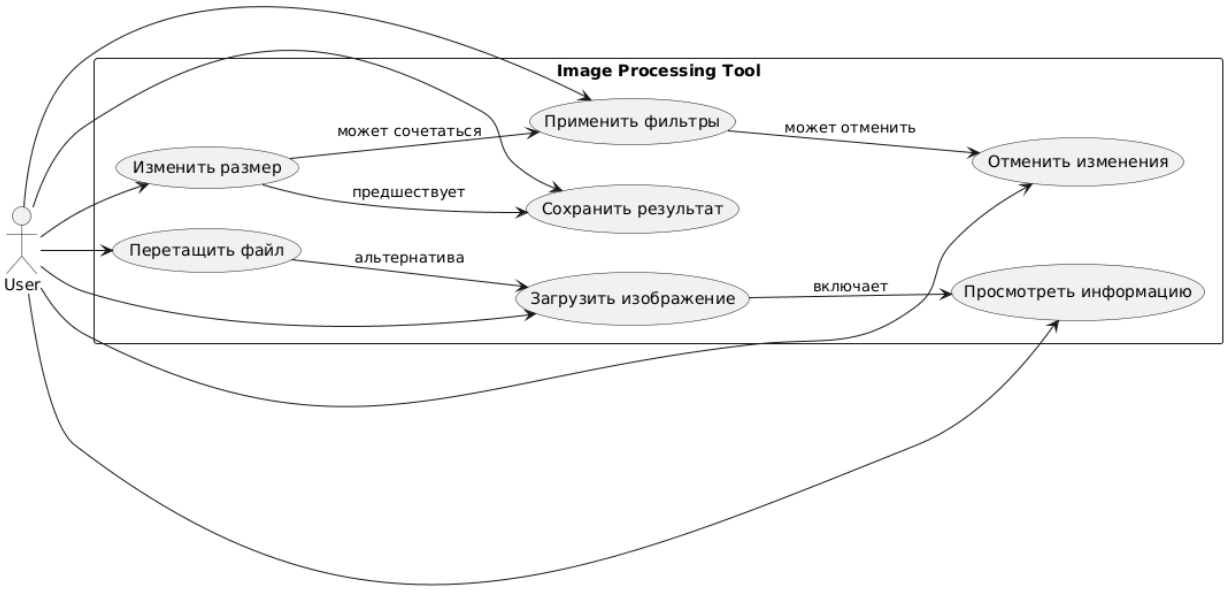

Рисунок 5 — Диаграмма сценариев использования

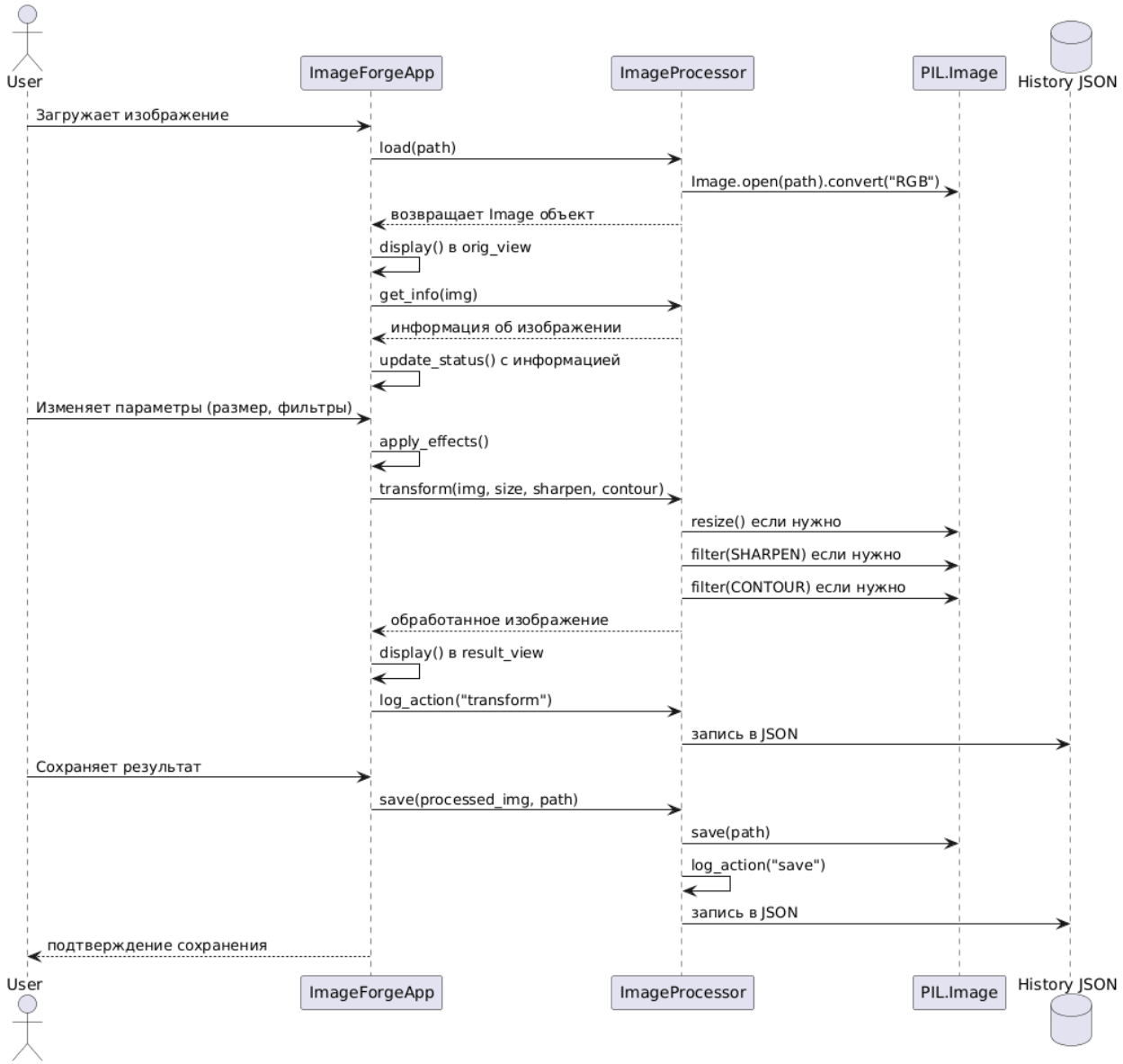

Рисунок 6 — Диаграмма последовательности для операций обработки изображения

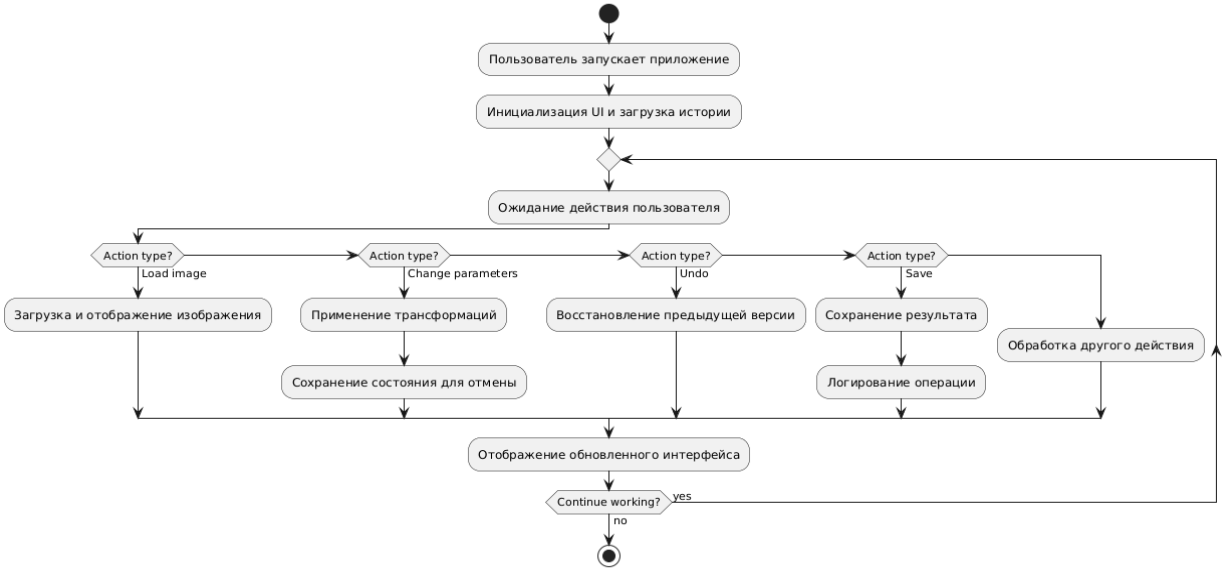

Рисунок 7 — Диаграмма деятельности