МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Южно-Уральский государственный университет**

**(национальный исследовательский университет)»**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

**Разработка iOS - приложения - фоторедактор с применением шейдерной графики**

КУРСОВАЯ РАБОТА

по дисциплине

«Проектирование и архитектура программных систем»

ЮУрГУ – 09.03.04.20241.308-065.КР

|  |  |
| --- | --- |
| Нормоконтролер:3  доцент кафедры СП, к.ф.-м.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_А.Т. Латипова  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | Научный руководитель:  доцент кафедры СП, к.ф.-м.н.  \_\_\_\_\_\_\_\_\_\_А.Т. Латипова  Автор работы:  студент группы КЭ-404  \_\_\_\_\_\_\_\_\_\_М.В. Кобяков  Работа защищена  с оценкой: \_\_\_\_\_\_\_\_\_\_\_  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Южно-Уральский государственный университет**

**(национальный исследовательский университет)»**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

УТВЕРЖДАЮ

Зав. кафедрой СП

\_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский

15.02.2024 г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине «Программная инженерия»

студенту группы КЭ-404 Кобякову Марку Васильевичу,

обучающемуся по направлению 09.03.04 «Программная инженерия»

1. **Тема работы**

Разработка iOS - приложения - фоторедактора с применением технологии Metal

1. **Срок сдачи студентом законченной работы: 23**.10.2024 г.
2. **Исходные данные к работе2**
   1. Apple Developer Documentation. [Электронный ресурс]. URL: <https://developer.apple.com/documentation> (дата обращения 20.04.2024 г.)
   2. Metal Tutorial: Getting Started. [Электронный ресурс]. URL: <https://www.kodeco.com/7475-metal-tutorial-getting-started> (дата обращения 30.05.2024 г.)
3. Перечень подлежащих разработке вопросов
   1. Выполнить анализ предметной области.
   2. Спроектировать iOS-приложение.
   3. Реализовать iOS-приложение.
   4. Провести тестирование разработанного iOS-приложения.
4. **Дата выдачи задания:** 15.02.2024 г.

**Научный руководитель,**

Доцент кафедры СП, к.ф.-м.н., доцент А.Т. Латипова

**Задание принял к исполнению** М.В. Кобяков

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc180378080)

[1.1. Предметная область проекта 7](#_Toc180378081)

[1.2. Обзор аналогичных приложений для реализации проекта 7](#_Toc180378082)

[Вывод по первой главе 11](#_Toc180378083)

[2. ПРОЕКТИРОВАНИЕ 12](#_Toc180378084)

[2.1. Требования к системе 12](#_Toc180378085)

[2.2. Варианты использования системы 13](#_Toc180378086)

[2.3. Диаграмма деятельности 15](#_Toc180378087)

[2.4. Диаграмма последовательности 16](#_Toc180378088)

[2.5. Проектирование архитектуры системы 17](#_Toc180378089)

[2.6. Проектирование интерфейса приложения 19](#_Toc180378090)

[Вывод по второй главе 23](#_Toc180378091)

[3. РЕАЛИЗАЦИЯ 24](#_Toc180378092)

[3.1. Общая архитектура компонентов 24](#_Toc180378093)

[3.2. Программные средства реализации 25](#_Toc180378094)

[3.3. Реализация пользовательского интерфейса 25](#_Toc180378095)

[Вывод по третьей главе 33](#_Toc180378096)

[4. Тестирование системы 34](#_Toc180378097)

[ЗАКЛЮЧЕНИЕ 36](#_Toc180378098)

[ЛИТЕРАТУРА 38](#_Toc180378099)

[ПРИЛОЖЕНИЯ 39](#_Toc180378100)

[Приложение А. Спецификация вариантов использования 39](#_Toc180378101)

# **ВВЕДЕНИЕ**

**Актуальность**

В эпоху стремительного роста цифровой фотографии и повсеместного распространения мобильных устройств [1], фоторедакторы для платформы iOS приобрели особую значимость как инструменты, предоставляющие пользователю широкие возможности персонализации и улучшения визуального контента. Современные пользователи ожидают от приложений не только базовые функции обработки изображений, но и сложные алгоритмы, обеспечивающие высокую производительность и качество.

В этом контексте, разработка iOS-приложения для редактирования фотографий приобретает особую актуальность благодаря следующим причинам:

— Рост использования мобильных устройств: Смартфоны и планшеты становятся основными устройствами для съёмки и обработки изображений. Платформа iOS, как одна из ведущих мобильных операционных систем, обеспечивает пользователей инструментами высокого качества для работы с графикой.

— Востребованность графических приложений: Приложения для редактирования фотографий являются неизменной составляющей пользовательских предпочтений. Такие приложения предлагают широкий спектр функций — от базовых фильтров и эффектов до сложной ретуши и редактирования.

— Технологические преимущества Metal: Использование Metal, мощной графической технологии от Apple, позволяет разработчикам создавать высокопроизводительные графические приложения. Metal предоставляет низкоуровневый доступ к GPU, что позволяет реализовать сложные эффекты и визуализации, которые ранее были доступны только на настольных системах.

— Инновации в области шейдерной графики: Применение шейдеров позволяет значительно расширить функциональные возможности фоторедакторов, создавая уникальные стилистические эффекты и оптимизируя обработку изображений в реальном времени. Шейдеры, работающие в среде Metal, могут использоваться для реализации настраиваемых эффектов и фильтров, существенно улучшая качество итогового изображения.

Aктуальность разработки iOS-приложения фоторедактора с использованием Metal объясняется не только ростом потребности в мощных и удобных инструментах для обработки изображений, но и стремительным развитием технологий, обеспечивающих необходимые для этого ресурсы. В условиях динамично изменяющегося рынка мобильных приложений, инновационный подход к разработке графического ПО становится ключевым фактором конкурентоспособности и востребованности.

**Постановка задачи**

Целью курсовой работы является разработка мобильного приложения для редактирования фотографий. Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ существующих мобильных приложений для редактирования фотографий и выделить их сильные и слабые стороны.
2. Определить функциональные требования к разрабатываемому мобильному приложению.
3. Разработать архитектуру и интерфейс мобильного приложения, обеспечивая удобство и интуитивность использования для пользователя.
4. Реализовать функции мобильного приложения.
5. Провести тестирование разработанного мобильного приложения на соответствие поставленным требованиям и обеспечить его стабильную работу.

**Структура и содержание работы**

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 43 страниц, объем списка литературы – 5 источников.

В первой главе описывается анализ предметной области.

Вторая глава посвящена разработке интерфейса, проектированию архитектуры приложения.

В третьей главе идет описание реализации проекта на примере одного из компонентов.

В главе четыре приведено тестирование готовой, работающей системы.

В приложении А содержится спецификация вариантов использования.

1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **Предметная область проекта**

С развитием мобильных технологий и увеличением числа пользователей iOS-устройств, рынок приложений для редактирования фотографий значительно расширился. Фоторедакторы стали неотъемлемой частью повседневного использования мобильных устройств, предлагая широкий спектр инструментов для обработки изображений. Пользователи ожидают высокой производительности, простоты в использовании и разнообразия функциональных возможностей, таких как применение фильтров, корректировка яркости и контраста, обрезка и ретушь изображений.

Ключевой проблемой для разработчиков фоторедакторов является оптимизация обработки изображений, так как задачи рендеринга и изменения изображений могут быть ресурсоемкими, особенно для мобильных устройств с ограниченными вычислительными мощностями. В этом контексте использование технологии Metal, разработанной Apple, играет значительную роль, поскольку предоставляет низкоуровневый доступ к графическому процессору (GPU), что позволяет выполнять графические операции гораздо быстрее по сравнению с традиционными подходами.

Использование Metal обеспечивает более гибкую и производительную обработку изображений, особенно в приложениях, требующих работы в реальном времени. Metal позволяет разработчикам реализовывать сложные визуальные эффекты, такие как кастомные фильтры и цветокоррекцию, с минимальной нагрузкой на процессор устройства. В современных условиях это становится не просто преимуществом, а необходимостью для разработки конкурентоспособных фоторедакторов.

## **Обзор аналогичных приложений для реализации проекта**

Для разработки iOS-приложения-фоторедактора с применением Metal важно провести анализ существующих решений на рынке. Среди множества доступных мобильных фоторедакторов можно выделить несколько ведущих приложений, каждое из которых обладает своими особенностями и технологиями, на которых строится их функциональность.

**Adobe Photoshop Express [2]** — один из самых популярных фоторедакторов для мобильных устройств. Это приложение предлагает широкий спектр инструментов для обработки изображений, включая фильтры, корректировку яркости и контраста, обрезку и другие функции. Основным преимуществом Adobe Photoshop Express является его интеграция с экосистемой Adobe, что позволяет пользователям легко переносить свои работы между мобильным устройством и компьютером. Однако приложение не использует Metal для обработки графики, полагаясь на менее производительные технологии, такие как Core Image.

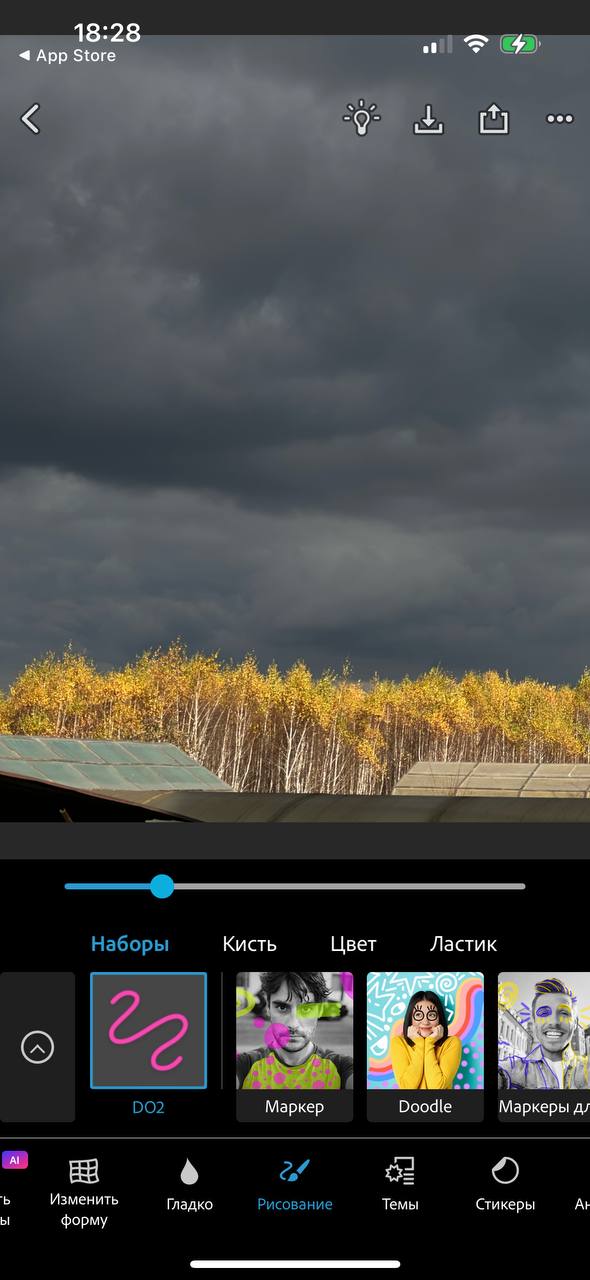


Рисунок 1 – Интерфейс Adobe Photoshop Express

**Snapseed** [3]— еще одно мощное мобильное приложение для редактирования фотографий, разработанное Google. Snapseed предлагает пользователям множество инструментов, включая уникальные возможности, такие как выборочная коррекция и фильтры HDR. Основным преимуществом Snapseed является его простота в использовании и поддержка работы с файлами RAW, что делает его популярным среди профессионалов и любителей. Однако, как и в случае с Adobe Photoshop Express, Snapseed использует стандартные графические API и не поддерживает Metal, что ограничивает его производительность на устройствах iOS.

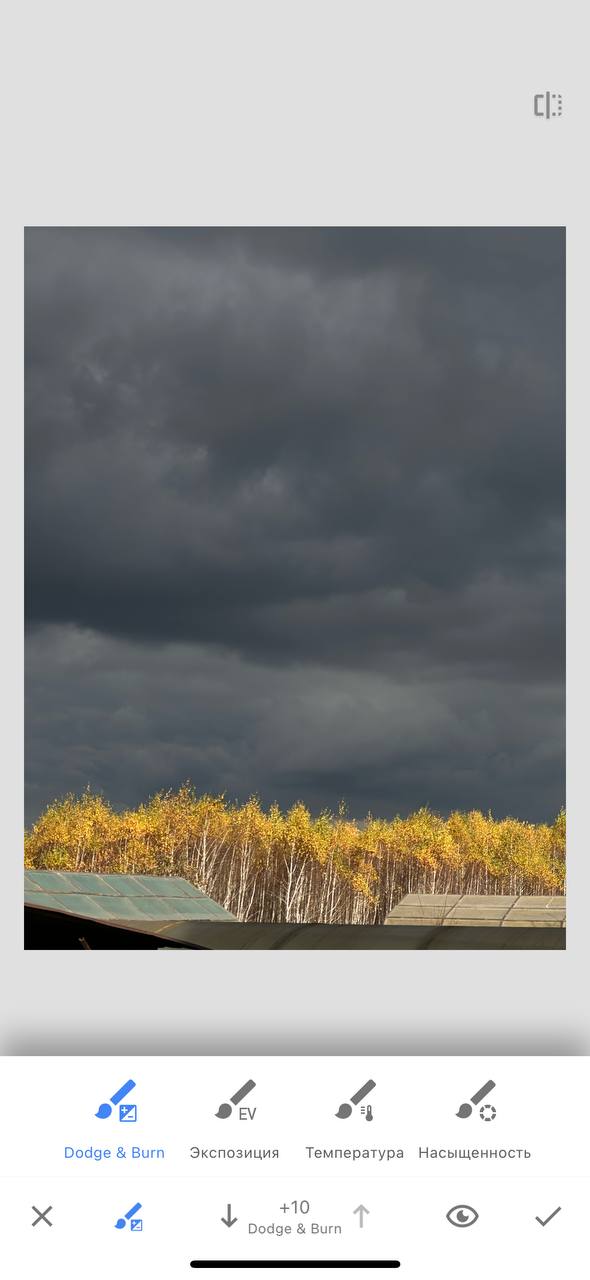


Рисунок 2 – Интерфейс Snapseed

**VSCO [4]** — это приложение, которое фокусируется на применении фильтров и создании уникального стиля изображений. Оно предоставляет пользователям обширную библиотеку готовых фильтров, что делает его особенно популярным среди фотографов-любителей. Однако, как и два предыдущих приложения, VSCO также не использует Metal для обработки графики, что ограничивает его возможности по рендерингу изображений в реальном времени.



Рисунок 3 – Интерфейс VSCO

Анализ этих решений показывает, что ни одно из популярных приложений для редактирования фотографий на платформе iOS не использует Metal в полной мере. Это открывает уникальные возможности для создания фоторедактора, который будет превосходить конкурентов по производительности, особенно в задачах, связанных с рендерингом изображений в реальном времени и применением сложных графических эффектов.

## Вывод по **первой** главе

Анализ существующих решений на рынке фоторедакторов показал, что большинство популярных приложений для iOS не используют возможности Metal для оптимизации работы с графикой, что открывает широкие возможности для разработки нового приложения с использованием этой технологии. Применение Metal позволит достичь значительных улучшений в производительности, гибкости и качестве обработки изображений по сравнению с текущими аналогами. Использование низкоуровневого доступа к графическим ресурсам через Metal делает возможным создание уникальных визуальных эффектов, а также ускоренную обработку изображений в реальном времени, что отвечает высоким ожиданиям современных пользователей.

# **ПРОЕКТИРОВАНИЕ**

## **Требования к системе**

В ходе проектирования iOS-приложения для редактирования фото, использующего технологию **Metal**, были определены следующие функциональные и нефункциональные требования.

**Функциональные требования**

Функциональные требования определяют действия, которые должна выполнять программа:

1. Приложение должно предоставлять возможность выбора фотографии из галереи устройства и загрузки её в редактор.
2. Приложение должно предоставлять возможность сохранения отредактированных фотографий обратно в галерею устройства.
3. Приложение должно поддерживать возможность рисования на фотографии пятью различными инструментами:
   * **Ручка** — изменяет ширину линии в зависимости от скорости перемещения, создавая вариативную толщину.
   * **Маркер** — полупрозрачная линия, также изменяющая ширину в зависимости от скорости перемещения.
   * **Карандаш** — фиксированная толщина линии.
   * **Ластик** — удаление нарисованного содержимого.
   * **Блюр** — инструмент для размытия области изображения.
4. Приложение должно предоставлять возможность отмены последних действий с помощью функции "Назад".
5. Приложение должно предоставлять возможность полного отката всех изменений и возвращения к исходной версии изображения.

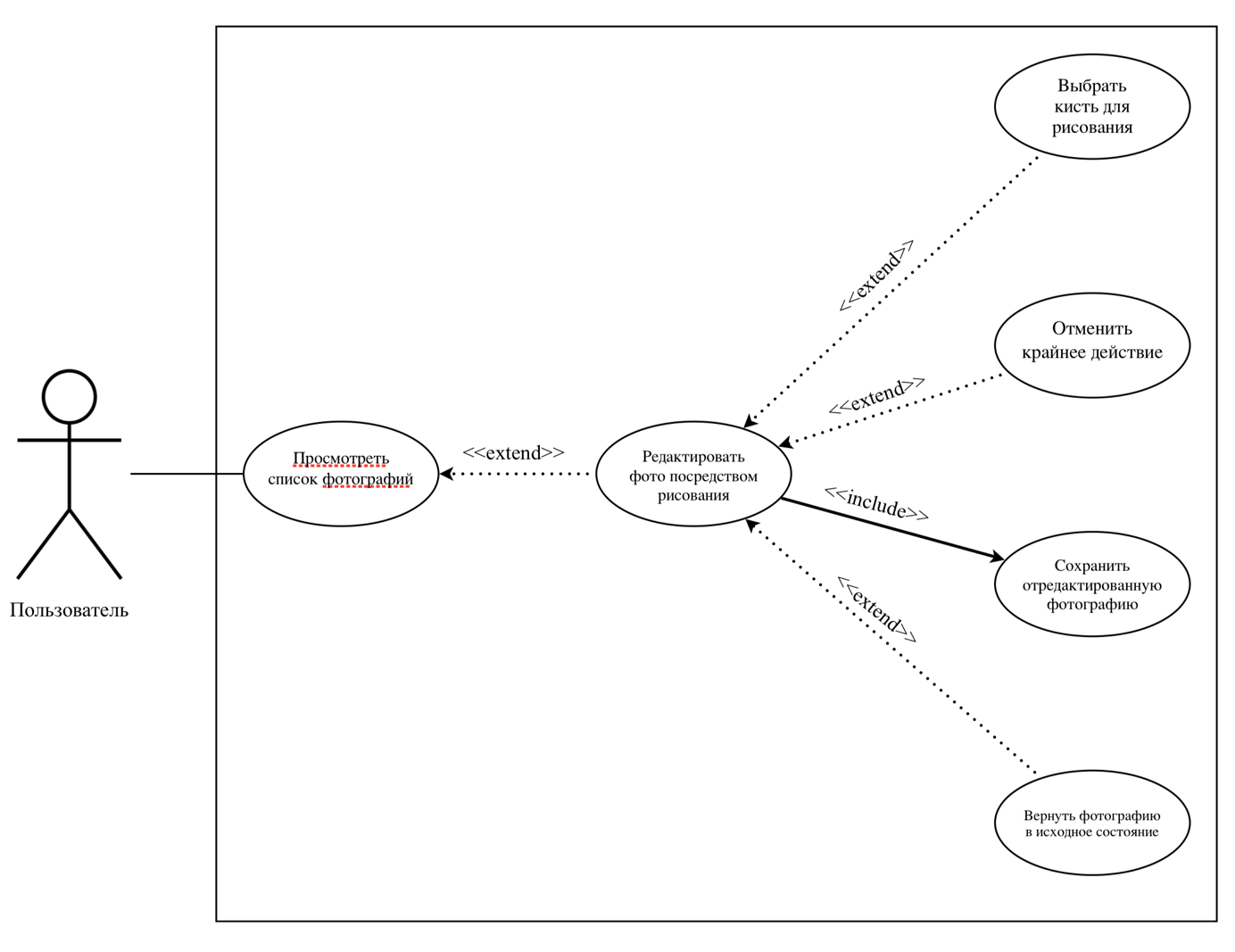
**Нефункциональные требования**

**Нефункциональные требования** (англ. **Non-functional requirements**, NFR) — это спецификации, которые описывают **качество работы системы** или её **ограничения**, но не относятся напрямую к функциональному поведению системы. Они не определяют, **что** система должна делать, а описывают, **как** она должна выполнять свои функции.

1. Вся бизнес-логика и графический движок для рисования должны быть реализованы с использованием **Metal** для обеспечения высокой производительности и плавного рендеринга.
2. Приложение должно иметь интуитивно понятный интерфейс для взаимодействия с фотографиями, включая понятные элементы управления для выбора кисти, отмены действий и сохранения результатов.
3. Пользовательский интерфейс должен быть адаптирован для работы на iOS с поддержкой современных устройств и экранов с различными разрешениями.
4. Приложение должно поддерживать оптимизированную работу с изображениями высокого разрешения без значительной потери производительности.
5. Архитектура приложения должна соответствовать принципам **Clean Architecture [5]**, чтобы обеспечить модульность, возможность тестирования и лёгкость расширения функциональности.
6. Приложение должно поддерживать обработку изображений в реальном времени с минимальными задержками.
7. Приложение должно предоставлять возможность интеграции с фотогаллереей пользователя для сохранения и загрузки изображений.
8. Безопасность данных пользователей, особенно при работе с изображениями, должна быть обеспечена через защищённые механизмы взаимодействия с галереей устройства.

## **Варианты использования системы**

В соответствии с требованиями была построена диаграмма вариантов использования, которая представлена на рисунке 4. Диаграмма отражает модель взаимодействия актера «Пользователь» с разрабатываемым приложением.

  
Рисунок 4 – Диаграмма вариантов использования

В системе предусмотрен единственный актер «Пользователь», он получает доступ ко всем функциям программы.

## **Диаграмма деятельности**

На рисунке 6 представлена диаграмма деятельности, отражающая редактирование фотографии из пользовательской галереи.

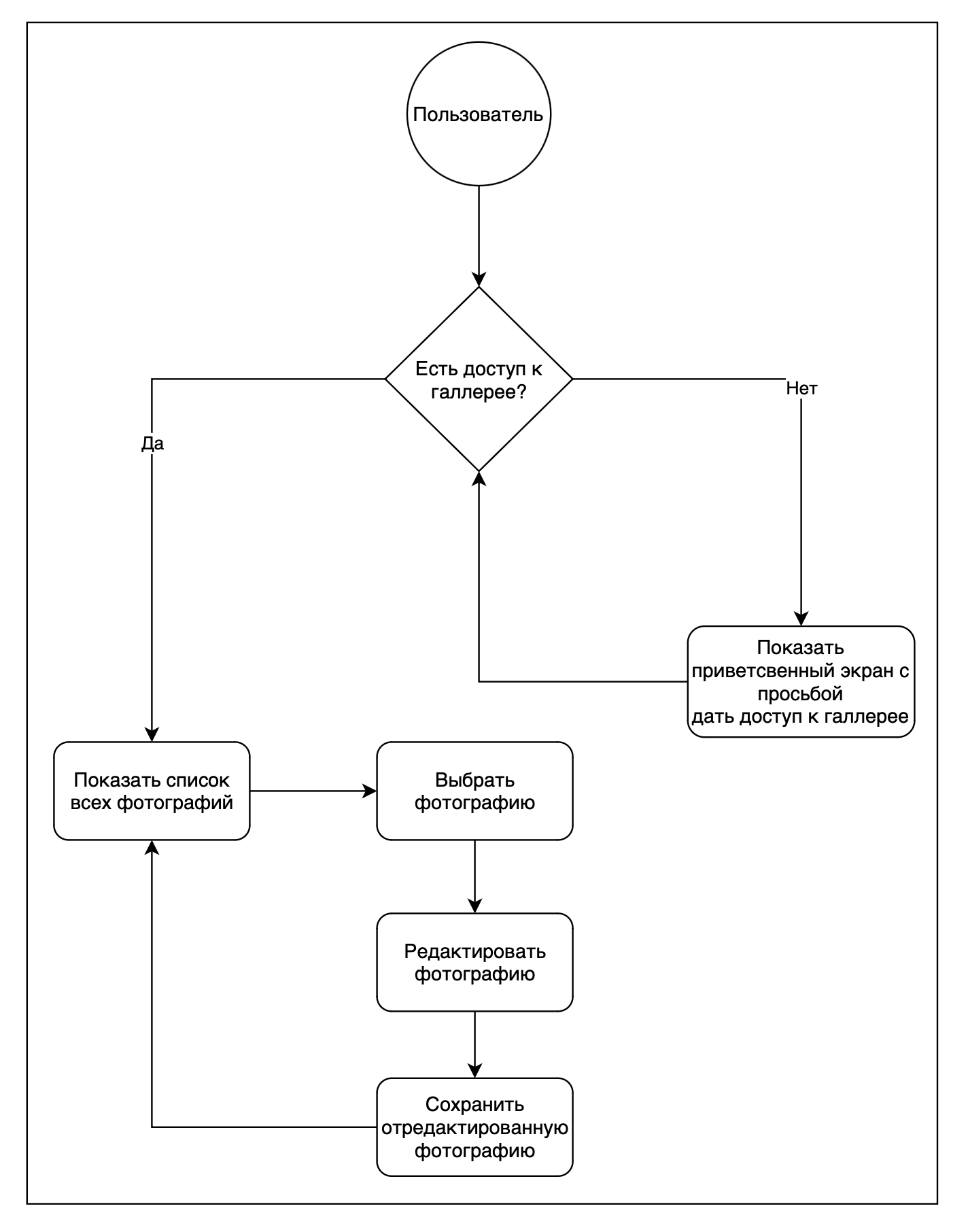


Рисунок 6 – Диаграмма деятельности редактирования фотографии

## **Диаграмма последовательности**

На рисунке 7 представлена диаграмма последовательности редактирования фотографии из пользовательской галереи.

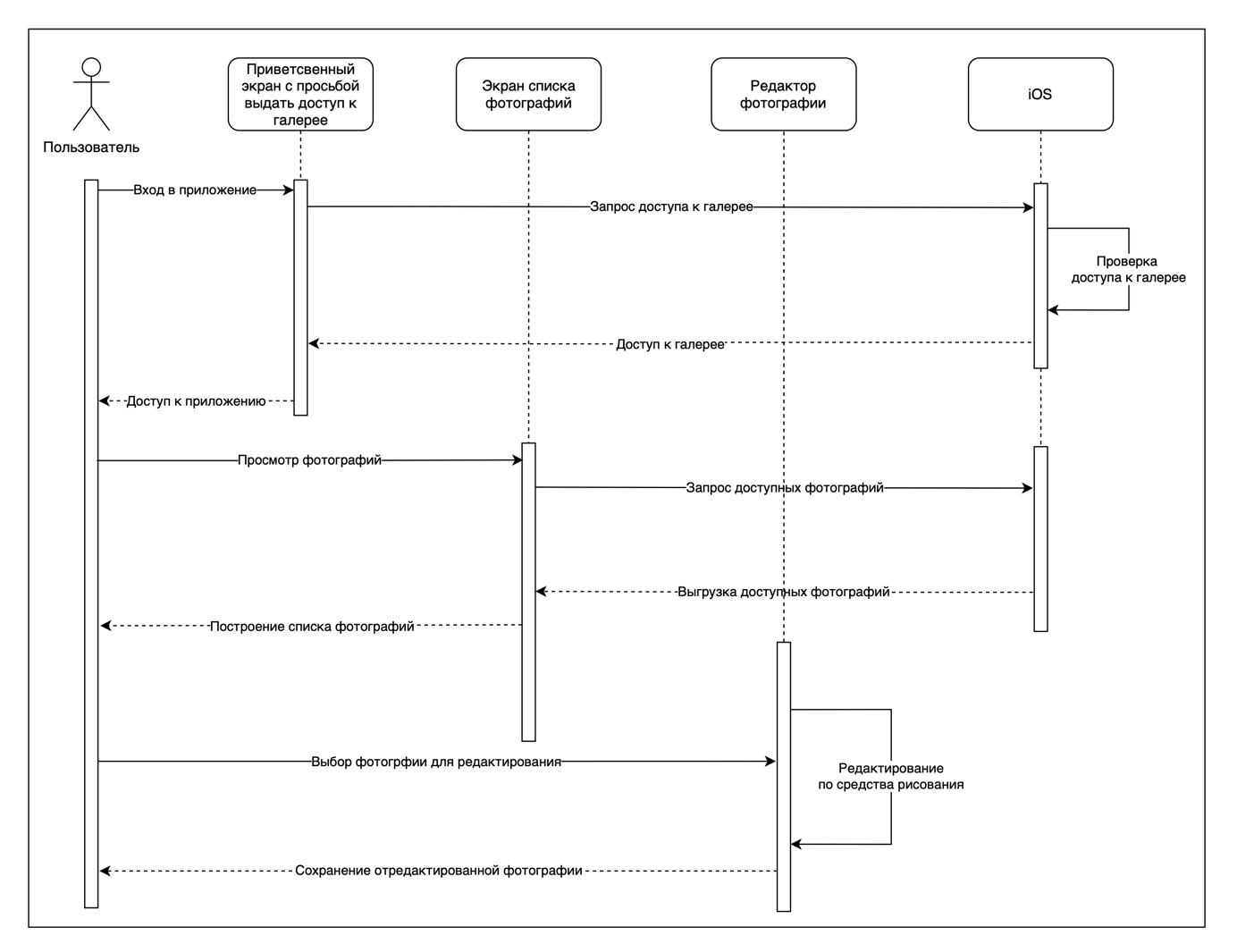
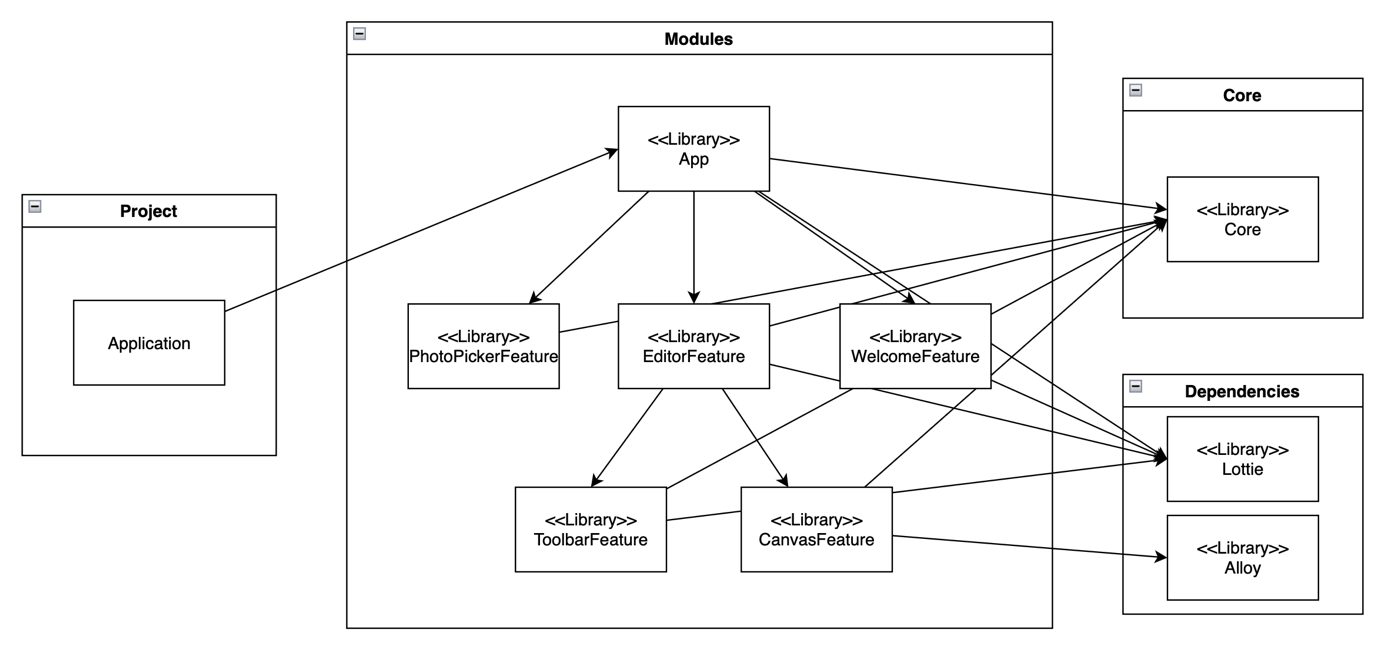


Рисунок 7 – Диаграмма последовательности добавления фотографии с камеры

## **Проектирование архитектуры системы**

В данном разделе представлено проектирование архитектуры приложения с помощью диаграммы компонентов, которая отражает структуру системы и её ключевые элементы. Спроектированная архитектура приложения представлена на рисунке 5 в виде диаграммы компонентов.

  
Рисунок 5 – Архитектура системы

Приложение состоит из нескольких модулей (библиотек), каждый из которых отвечает за определённую функциональность. Основные библиотеки разделены на две категории: фичи (features) и основные зависимости (Core и Dependencies).

**Основные модули (features):**

1. **App** - модуль является точкой входа приложения. Он собирает зависимости из всех фичей, таких как **WelcomeFeature**, **PhotoPickerFeature**, **EditorFeature**.
2. **WelcomeFeature** – модуль отвечает за отображение приветственного экрана приложения. Здесь используются зависимости из **Core** для основной логики и анимации через библиотеку **Lottie** (упрощенная интеграция через **Dependencies**).
3. **PhotoPickerFeature** - модуль предоставляет функциональность для выбора фотографий из галереи устройства. В нём отсутствуют зависимости от внешних библиотек, и он использует только функционал из **Core** для базовой логики работы.
4. **EditorFeature** - Модуль для редактирования изображений. Он имеет зависимости от нескольких фичей, таких как **ToolbarFeature** и **CanvasFeature**, которые помогают в построении интерфейса и рендеринге.
5. **ToolbarFeature** – модуль управляет логикой и интерфейсом панели инструментов для редактирования изображений. Этот модуль также использует **Lottie** для визуальных эффектов.
6. **CanvasFeature** - модуль для работы с холстом и рендерингом изображений. Он использует **Alloy** для работы с графикой через Metal, что обеспечивает высокую производительность для рендеринга изображений и рисования.

**Вспомогательные модули:**

1. **Core** - модуль содержит общие для всех фичей компоненты, такие как модели данных, цвета, и основные вспомогательные функции. Все фичи зависят от **Core**, что создаёт общий базовый уровень для всего приложения.
2. **Dependencies** - модуль отвечает за управление внешними зависимостями, такими как **Lottie** и **Alloy**.

Архитектура разделена на модули, где каждая фича реализована как отдельный модуль с чёткими зависимостями от общих компонентов. Такая архитектура позволяет легко управлять проектом, добавлять новые фичи и тестировать существующие.

## **Проектирование интерфейса приложения**

Дизайн главного экрана представлен на рисунке 9. Экран содержит список фотографий.

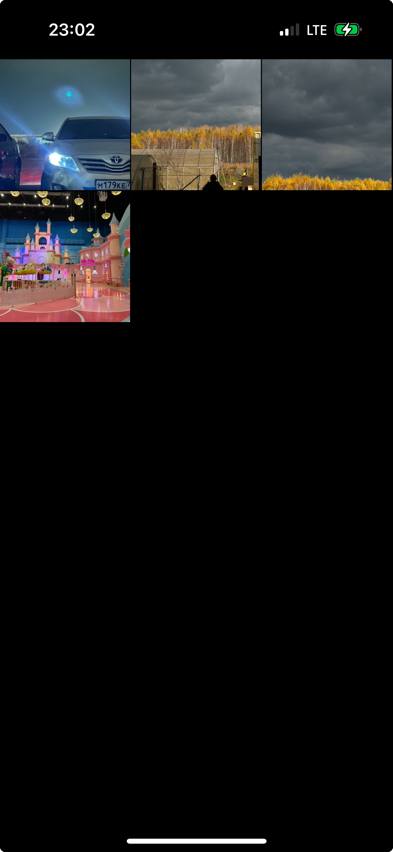


Рисунок 9 – Дизайн главного экрана

По клику на фотографию, пользователь попадает на экран редактирования фотографии, который представлен на рисунке 10. В нижней части экрана находится панель инструментов для рисования. Пользователь может выбрать один из четырех типов кистей: Ручка, Маркер, Карандаш и Ластик. Выбранная кисть выделяется большим размером по отношению к другим.



Рисунок 10 – Главный экран

При повторном нажатии на кисть, включается режим настройки отдельной кисти, представленный на рисунке 11. Пользователь может настроить ширину линии, а также режим рисования.



Рисунок 11 – Настройка кисти

Кроме того, на панели доступна кнопка для выбора цвета. Нажатие на эту кнопку открывает окно выбора цветов, где пользователь может выбрать цвет с помощью спектра или бегунков. Также можно настраивать прозрачность цвета для рисования. Функционал выбора цвета представлен на рисунке 12.

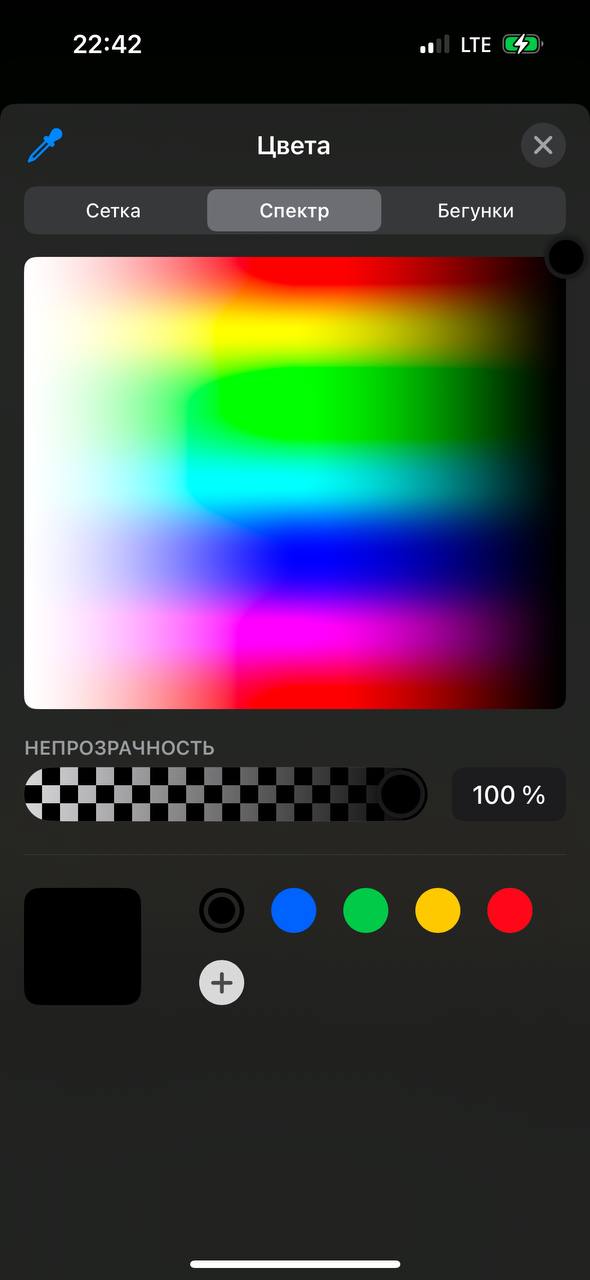


Рисунок 12 – Выбор цвета для кисти

По клику на кнопку «Назад» пользователь может либо отменить последнее действие. По клику на кнопку «Очистить» изображение возвращается в исходное состояние. Для того чтобы прекратить редактирование пользователь может нажать на «Крестик» слева внизу, чтобы отменить все изменения, либо нажать на «Стрелку вниз», чтобы сохранить результат в галерею.

## **Вывод по второй главе**

В ходе проектирования системы были определены функциональные требования, на основе которых была построена диаграмма вариантов использования, а также диаграмма деятельности и диаграмма последовательности. Также были определены ключевые участники системы (актеры).

Была спроектирована архитектура приложения на уровне компонентов. Были спроектированы макеты основных экранов, которые обеспечивают представление о будущей функциональности системы и о пользовательском интерфейсе.

# **РЕАЛИЗАЦИЯ**

## **Общая архитектура компонентов**

Разрабатываемое приложение построено на архитектурном подходе MVP. MVP (Model-View-Presenter) — это один из популярных архитектурных паттернов, который разделяет обязанности между тремя ключевыми компонентами: **Model**, **View** и **Presenter**. В контексте разработки на iOS с использованием **UIKit**, MVP помогает улучшить структурирование кода и облегчить тестирование логики. Рассмотрим каждый компонент и их роль в этом паттерне:

1. **Model**:
   * Отвечает за бизнес-логику приложения и работу с данными. Это может быть слой, взаимодействующий с сетью, базой данных или API.
   * **Model** никогда не взаимодействует напрямую с **View**.
2. **View** (представление):
   * Это **UIViewController** и другие элементы UI, которые отвечают за отображение данных пользователю.
   * **View** ничего не знает о бизнес-логике или способах получения данных. Она выполняет исключительно отображение данных и получение пользовательских вводов.
   * **View** отправляет события в **Presenter**, который управляет логикой.
3. **Presenter**:
   * Связующее звено между **Model** и **View**.
   * **Presenter** получает данные от **Model**, обрабатывает их и передаёт **View** для отображения.
   * Обрабатывает пользовательские действия и передаёт их в **Model** для соответствующих действий.
   * **Presenter** не должен ничего знать о конкретных реализациях **View** (например, какие конкретно элементы UI используются).

Примерная схема взаимодействия компонентов в рамках MVP представлена на рисунке 13.

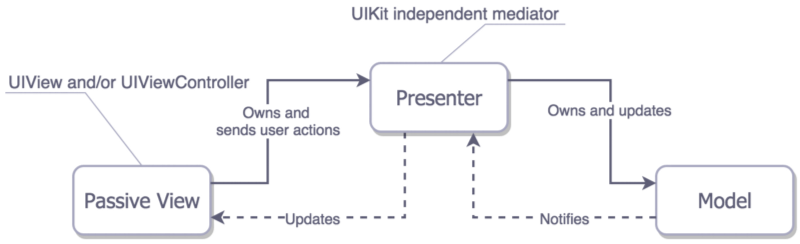


Рисунок 13 – MPV

## **Программные средства реализации**

Для реализации iOS-приложения использовались следующие инструменты:

* Язык программирования Swift версии 5.9;
* UIKit, соответствующий последней версии iOS;
* MetalKit, соответствующий последней версии iOS;
* Alloy версии 0.18.2 предоставляет набор инструментов для более простого использования Metal;
* Lottie версии 4.5.0 для создания анимаций;

## **Реализация пользовательского интерфейса**

За неимением смысла рассматривать каждый отдельный компонент, остановимся на основополагающем «CanvasFeature». Файловая структура «CanvasFeature» представлена на рисунке 14.

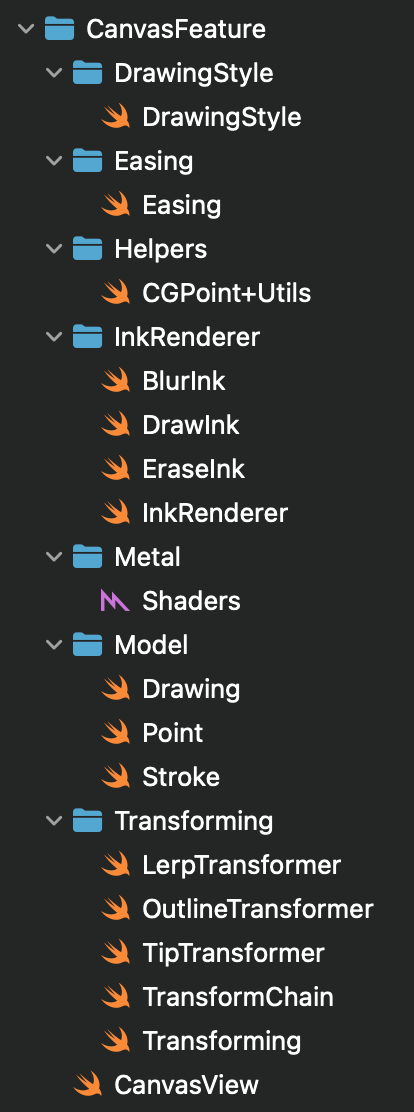


Рисунок 14 – Файловая иерархия «CanvasFeature»

Доменными моделями компонента являются:

* DrawingStyle – рисунок 15:

Описывает текущий стиль линии, где:

— **color**: цвет линии;

— **strokeSize**: ширина линии;

— **ink**: вид чернила (draw, blur, erease);

— **tipType**: кончик линии (default, arrow);

— **transforming**: трансформации, производимые над линиями;

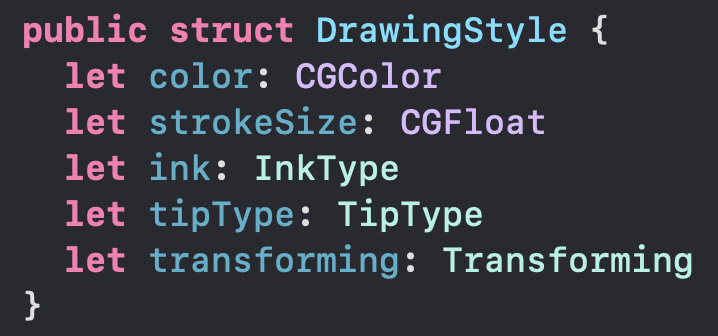


Рисунок 15 – **DrawingStyle**

* **InkRenderer** – рисунок 16:

Описывает интерфейс для всех видов чернил, использующих Metal-рендеринг, где:

— **texture**: ссылка на текущую текстуру, используемую при рендеринге;

— **setImage(\_ texture: MTLTexture?)**: метод для установки изображения (текстуры), с которым будет работать рендерер;

— **setMask(\_ texture: MTLTexture?)**: метод для установки маски, используемой для применения эффектов чернил;

— **update()**: метод для обновления рендера в зависимости от изменений текстур и масок.



Рисунок 16 – **InkRenderer**

* **Drawing** – рисунок 17:

Структура, описывающая текущий слой рисунка, где:

— **strokes**: массив линий (Stroke), составляющих слой;

— **color**: цвет рисунка (CGColor);

— **ink**: тип чернил, используемых для рисунка (InkType).

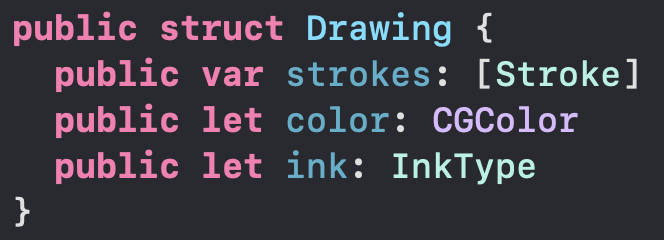


Рисунок 17 – **Drawing**

* **Stroke** – рисунок 19:

Структура, описывающая линию рисунка, состоящую из множества точек (Point), где:

— **points**: массив точек, составляющих линию (Point);

— **strokeSize**: ширина линии (CGFloat);

— **isClosed**: флаг, обозначающий, является ли линия замкнутой (Bool);

— **isFilled**: флаг, обозначающий, залита ли линия (Bool).

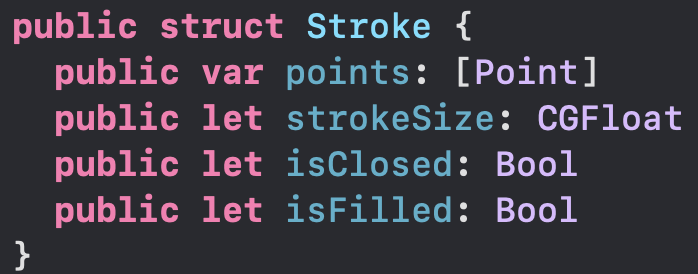


Рисунок 18 – **Stroke**

* **Point** – рисунок 19:

Структура, описывающая точку линии, где:

— **location**: координаты точки на экране (CGPoint);

— **time**: время появления точки в рамках рисунка

(CFTimeInterval);

— **force**: сила нажатия, применяемая к точке (CGFloat).

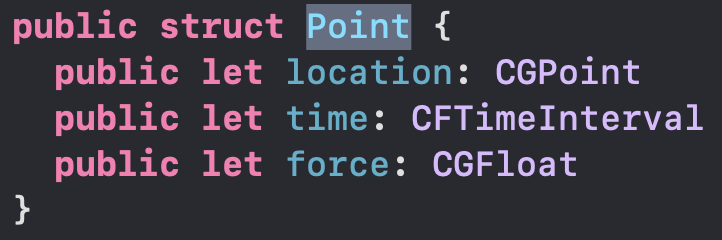


Рисунок 19 – **Point**

Были статически описанны пять реализаций **DrawingStyle** (pen, marker, pencil, blur, erase). Для того, чтобы реализовать плавность линий и реалистичность линий, были применены различные алгоритмы сглаживания, а также динамического изменения ширины линии, особенно сложной в этом плане можно считать **DrawingStyle.pen**, код которой представлен на рисунке 20.

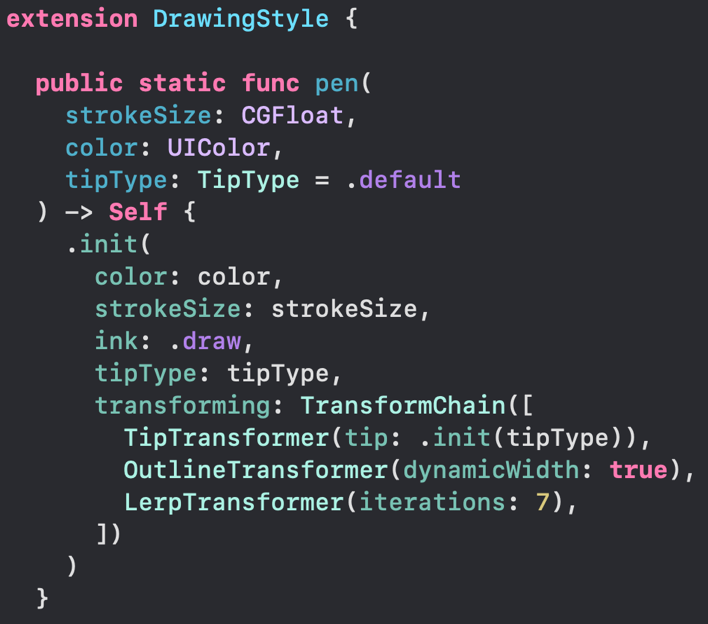


Рисунок 20 - **DrawingStyle.pen**

Главный интерес здесь представляют:

* **LerpTransformer** – выполняет многоэтапное сглаживание линий в рисунке, с помощью линейной интерполяции, чтобы сделать их более плавными. Это достигается за счёт вычисления промежуточных точек между соседними точками на линии, которые затем многократно обновляются в процессе итераций. В результате пользователь получает визуально сглаженный рисунок без резких переходов.
* **OutlineTransformer – для** каждой точки линии вычисляется нормаль, которая определяет смещение влево и вправо, создавая "левую" и "правую" сторону линии, формируя замкнутую обводку. Дополнительно алгоритм сглаживает скорость движения по линии и плавно изменяет ширину в её начале и конце для создания более естественного вида. В коде используются алгоритмы линейной интерполяции для сглаживания скоростей, вычисления нормалей для построения обводки, а также функции easing для плавного уменьшения ширины линии.
* **LerpTransformer** – добавляет стрелку на конец линии, если задан соответствующий тип наконечника. Он вычисляет размер стрелки в зависимости от ширины линии, определяет угол наклона линии и добавляет стрелку как отдельную часть линии, делая рисунок визуально более выразительным.

Однако это лишь часть всего, основная работа происходит в имплементациях **InkRenderer**: **BlurInk**, **DrawInk**, **EraseInk**.Принцип работы каждого не сильно отличается от другого, поэтому приведу подробное описание **DrawInk:** Класс **DrawInk** реализует рендеринг изображения с использованием Metal и работает с текстурами изображения и маски. Он загружает шейдеры для обработки данных, настраивает состояние графического конвейера и создаёт offscreen-рендерер для работы с текстурами вне экрана. Метод **setImage** задаёт текстуру изображения, а **setMask** задаёт маску, которая применяется к этому изображению. Во время рендеринга метод **update** выполняет обработку текстур через графический буфер и сохраняет результат в новую текстуру для дальнейшего использования. Частичная реализация представлена на рисунке 21.



Рисунок 21 – **DrawInk**

Примечательны особенно здесь 2 строчки:

— defaultLibrary.makeFunction(name: "vertexFunc"):

Вершинный шейдер — это первая стадия рендеринга на GPU, которая обрабатывает отдельные вершины объектов. В этом шейдере определяются координаты каждой вершины в пространстве. Вершинный шейдер также может передавать информацию другим шейдерам (например, UV-координаты для текстурирования). Этот вершинный шейдер предназначен для обработки вершин двумерного прямоугольника в пространстве экрана и сопоставления его с текстурными координатами. Он преобразует вершины, задаёт позиции для рендеринга и вычисляет UV-координаты для текстуры. Реализация представлена на рисунке 22.

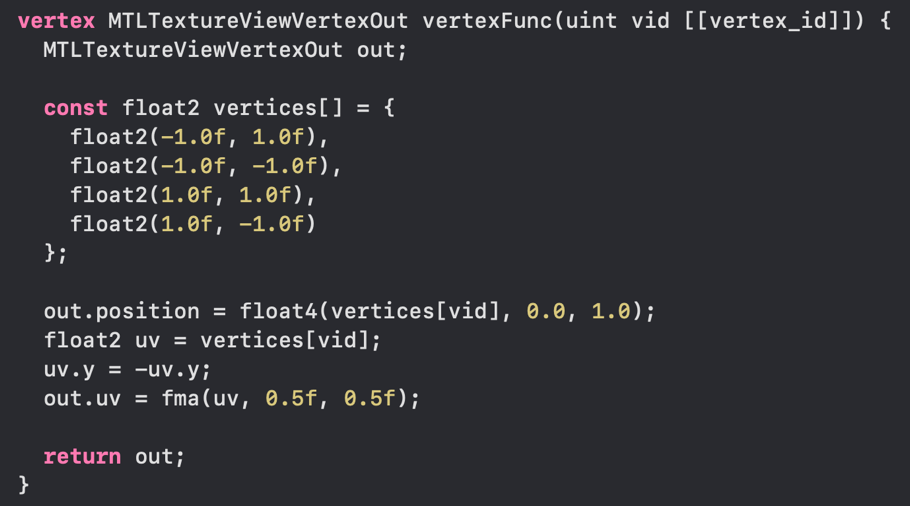


Рисунок 22 - **vertexFunc**

— defaultLibrary.makeFunction(name: "fragmentDrawingFunc")

Фрагментный шейдер — это стадия рендеринга, которая обрабатывает пиксели изображения (фрагменты). Этот шейдер отвечает за определение цвета каждого пикселя объекта. Обычно фрагментный шейдер получает на вход данные от вершинного шейдера (например, цвет или координаты текстуры) и вычисляет, как должен выглядеть каждый пиксель на экране. Этот фрагментный шейдер **fragmentDrawingFunc** выполняет операцию смешивания двух текстур с использованием альфа-компоненты одной из текстур для управления прозрачностью. Реализация представлена на рисунке 23

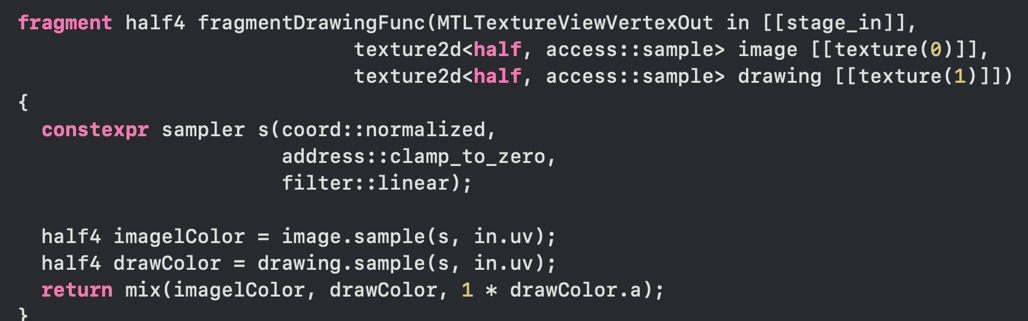


Рисунок 23 – **fragmentDrawingFunc**

## **Вывод по третьей главе**

В рамках данной главы был подробно рассмотрен компонент «CanvasFeature», однако это лишь часть работы, всего в рамках приложения было реализовано 6 компонентов, сверстано 3 экрана и реализованы основные операции для редактирования путем рисования по фотографии.

# **Тестирование системы**

Функциональное тестирование - это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определенных задачах решать задачи, необходимые пользователям.

С использование методологии функционального тестирования была проверена работа iOS-приложения (таблица 1).

Таблица 1 – Функциональное тестирование

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название теста** | | **Шаги** | **Ожидаемый результат** | | **Тест пройден?** |
| **1** | Редактирование фото путем рисования на фотографии | 1. Нажать на фотографию в списке  2. В открывшемся редакторе выбрать карандаш  3. Нарисовать что-либо | | | В процессе рисования отображается то, что рисует пользователь | **Да** |
| **2** | Запрет доступа к галерее | 1. На приветственном экране нажать на кнопку «Дать доступ»  2. В открывшемся окошке запретить доступ | | | Приветственный экран не пропал, список фото не появился, теперь надо идти в настройки для того чтобы выдать доступ | **Да** |
| **3** | Разрешение доступа к галерее | 1. На приветственном экране нажать на кнопку «Дать доступ»  2. В открывшемся окошке разрешить доступ | | | Приветственный экран пропал, список фото появился, теперь можно редактировать фотографии | **Да** |
| **4** | Сохранение отредактированной фотографии | 1. На экране редактирования что-то нарисовать  2. Нажать на кнопку «Сохранить» | | | Экран редактирования пропал, появился экран списка фото, в нем появилась отредактированная фотография | **Да** |
| **5** | Отмена отредактированной фотографии | 1. На экране редактирования что-то нарисовать 2. Нажать кнопку «Отмена» | | | Экран редактирования пропал, появился экран списка фото, в нем ничего не поменялось | **Да** |

**Вывод по четвертой главе**

Четвертая глава завершилась успешным функциональным тестированием приложения. Все тестовые сценарии были выполнены без ошибок, подтверждая корректность работы всех ключевых функций. Это свидетельствует о том, что приложение соответствует заявленным требованиям и готово к дальнейшим этапам разработки, таким как интеграционное тестирование или развертывание.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы был разработан фоторедактор с применением технологии Metal. При этом были решены следующие задачи:

1) проведен анализ существующих мобильных приложений для редактирования фотографий и выделены их сильные и слабые стороны;

2) определены функциональные требования к разрабатываемому мобильному приложению;

3) разработаны архитектура и интерфейс мобильного приложения, обеспечивающие удобство и интуитивность использования для пользователя;

4) реализованы функции мобильного приложения;

5) проведено тестирование разработанного мобильного приложения на соответствие поставленным требованиям.

Проведенный анализ существующих решений в сфере мобильных приложений для редактирования фотографий позволил определить ключевые требования к разрабатываемому приложению и сформулировать его функциональные особенности.

Разработанная архитектура и интерфейс мобильного приложения обеспечивают удобство и интуитивность использования. Реализованы ключевые функции, такие как плавное и эстетичное рисование с помощью набора различных инструментов.

Проведенное тестирование подтвердило соответствие приложения поставленным требованиям и обеспечило его стабильную работу на обеих платформах.

В результате выполнения работы было создано iOS – приложение - фоторедактор с применением технологии Metal для редактирования фотографий, которое повышает удобство пользователей и обеспечивает стабильную работу на различных устройствах с операционной системой iOS.

Данная работа имеет следующие пути дальнейшего развития:

1. Добавление новых инструментов для редактирования, таких как работа с текстом и фигурами.
2. Внедрение функций для сохранения и синхронизации фотографий в облачных хранилищах.
3. Интеграция с популярными социальными сетями для упрощения обмена отредактированными фотографиями.
4. Добавление черновиков, чтобы реализовать возможность возвращаться к редактированию спустя время.

# **ЛИТЕРАТУРА**

1. Mobile Phone Statistics [Электронный ресурс]URL: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world (дата обращения 15.04.2024 г.)
2. Adobe Photoshop Express [Электронный ресурс] URL: https://www.adobe.com/ru/products/photoshop-express.html (дата обращения 15.04.2024 г.)
3. Snapseed [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.niksoftware.snapseed&hl=ru&pli=1 (дата обращения 15.04.2024 г.)
4. VSCO [Электронный ресурс] URL: https://www.vsco.co/ (дата обращения 15.04.2024 г.)
5. Роберт М. Чистая архитектура, искусство разработки программного обеспечения // Издательский Дом«Питер», 2024. – С. 202-209.

# **ПРИЛОЖЕНИЯ**

## **Приложение А. Спецификация вариантов использования**

Таблица 1 – спецификация ВИ «Дать доступ к галерее»

|  |
| --- |
| Прецедент: дать доступ к галерее |
| ID: 1 |
| Краткое описание:  Чтобы пользоваться приложением необходимо выдать доступ |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: Доступа нет, либо доступ запрещен |
| Основной поток:  1. Прецедент начинается, когда пользователь запускает приложение  2. Пользователь попадает на экран приветствия с просьбой выдать доступ  3. Пользователь выдает доступ |
| Постусловия:  Получен список фотографий из галереи |
| Альтернативные потоки:  1. Пользователь запретил доступ у фото  2. Система не может корректно работать |

Таблица 2– спецификация ВИ «Просмотреть список фотографий»

|  |
| --- |
| Прецедент: просмотреть список фотографий |
| ID: 2 |
| Краткое описание:  Просмотр всех доступных фотографий |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: Выдан доступ к галерее |
| Основной поток:  1. Прецедент начинается, когда пользователь заходит в приложение.  2. Пользователь попадает на главный экран приложения |
| Постусловия:  Нет |
| Альтернативные потоки:  Нет |

Таблица 3– спецификация ВИ «Редактировать посредством рисования»

|  |
| --- |
| Прецедент: редактировать посредством рисования |
| ID: 3 |
| Краткое описание:  Рисование на фотографии |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: 1. Пользователь находится на экране списка фотографий |
| Основной поток:  1. Прецедент начинается, когда пользователь нажимает на фотографию  2. Пользователь редактирует фотографию |
| Постусловия:  Нет |
| Альтернативные потоки:  Нет |

Таблица 4 – спецификация ВИ «Сохранить фотографию»

|  |
| --- |
| Прецедент: сохранить фотографию |
| ID: 4 |
| Краткое описание:  Сохранение фотографии |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия:  1. Пользователь редактирует фотографию |
| Основной поток:  1. Прецедент начинается, когда пользователь сохраняет редактируемое изображение  2. Система получает **UIImage** от пользователя и сохраняет его в галерею |
| Постусловия:  Нет |
| Альтернативные потоки:  Нет |

Таблица 5 – спецификация ВИ «Отменить редактирование»

|  |
| --- |
| Прецедент: отменить редактирование |
| ID: 5 |
| Краткое описание:  Отмена редактирование фотографии |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия:  1. Пользователь редактирует фотографию |
| Основной поток:  1. Прецедент начинается, когда пользователь отменяет редактирование  2. Система не получает **UIImage** от пользователя и не сохраняет его в галерею |
| Постусловия:  В галерее не появилось новой фотографии |
| Альтернативные потоки:  Нет |