МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
 ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
 **«Дальневосточный федеральный университет»**

**ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ  
 Департамент программной инженерии и искусственного интеллекта**

**РАЗРАБОТКА И РЕАЛИЗАЦИЯ ГРАФИЧЕСКОГО РАСТРОВОГО РЕДАКТОРА**

КУРСОВАЯ РАБОТА

по дисциплине «Технологии коллективной промышленной разработки информационных систем» по образовательной программе подготовки бакалавров по направлению 02.03.03 «Технология программирования»

Выполнили студенты гр. Б9121-02.03.03тп  
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Гальперин А. И.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Волчков Н. А.

Руководитель:  
Старший преподаватель ДПИиИИ ИМиКТ ДВФУ  
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Иваненко Ю. С.

г. Владивосток  
 2025

**Оглавление**

[Введение 3](#_Toc188983544)

[1. План проекта 5](#_Toc188983545)

[2. Разработка регламента проведения инспекции 6](#_Toc188983546)

[2.1. Верификация рабочих продуктов 6](#_Toc188983547)

[2.2. Критерии отнесения к формальной инспекции 6](#_Toc188983548)

[2.3. Перечень ролей участников инспекции и их обязанности 7](#_Toc188983549)

[2.4. Этапы инспекции 7](#_Toc188983550)

[2.5. Порядок организации 8](#_Toc188983551)

[2.6. Порядок подготовки и проведения 8](#_Toc188983552)

[2.7. Перечень статусов и степени важности замечаний 9](#_Toc188983553)

[2.8. Порядок верификации замечаний 10](#_Toc188983554)

[3. Модель состояний задач 11](#_Toc188983555)

[3.1. Перечень возможных состояний задачи и их интерпретация 11](#_Toc188983556)

[3.2. Правила создания новой задачи 11](#_Toc188983557)

[3.3. Правила перехода задачи из состояния в состояние 12](#_Toc188983558)

[4. Презентация проекта 14](#_Toc188983559)

[5. Требования к проекту 17](#_Toc188983560)

[5.1. Требования к подсистеме Editor (ED) 17](#_Toc188983561)

[5.2. Требования к подсистеме AI Fill (AI) 18](#_Toc188983562)

[5.3. Требования к подсистеме Configuration (CFG) 18](#_Toc188983563)

[6. Архитектура проекта 19](#_Toc188983564)

[7. Измерения проекта 22](#_Toc188983565)

[7.1 Метрики для оценки эффективности процесса разработки 22](#_Toc188983566)

[7.2 Метрика качества программного продукта 22](#_Toc188983567)

[8. Перечень задач проекта 24](#_Toc188983568)

[9. Рекомендации по кодированию 26](#_Toc188983569)

[9.1. Рекомендации 26](#_Toc188983570)

[9.2. Требования 27](#_Toc188983571)

[9.3. Запреты 27](#_Toc188983572)

# Введение

Промышленная разработка информационных систем включает в себя множество этапов, начиная от разработки плана проекта, заканчивая тестированием проекта для чего, очевидно необходимо множество специалистов различных профилей, а также унифицированные методы коммуникации между ними, с помощью которых можно разделить обязанности членов команды по их специализации.

Исходя из описанного выше необходимо использовать определенные технологии коллективной разработки для повышения эффективности работы в группе и соответствия конечного продукта заявленным требованиям.

В данной курсовой работе рассматривается задача коллективной разработки проекта «Разработка и реализация графического растрового редактора» и составление технической документации к нему.

Таким образом, целью курсовой работы является разработка разработки проекта «Разработка и реализация графического растрового редактора» с использованием подходов коллективной промышленной разработки.

Для достижения поставленной цели необходимо решить следующие задачи:

* разработать план проекта;
* разработать регламент проведения инспекции;
* разработать модель состояний задач;
* разработать презентацию проекта;
* разработать требования к проекту;
* разработать архитектуру проекта;
* разработать измерения проекта;
* разработать перечь задач проекта;
* разработать рекомендации по кодированию;
* разработать план тестирования проекта;
* протестировать проект.

# План проекта

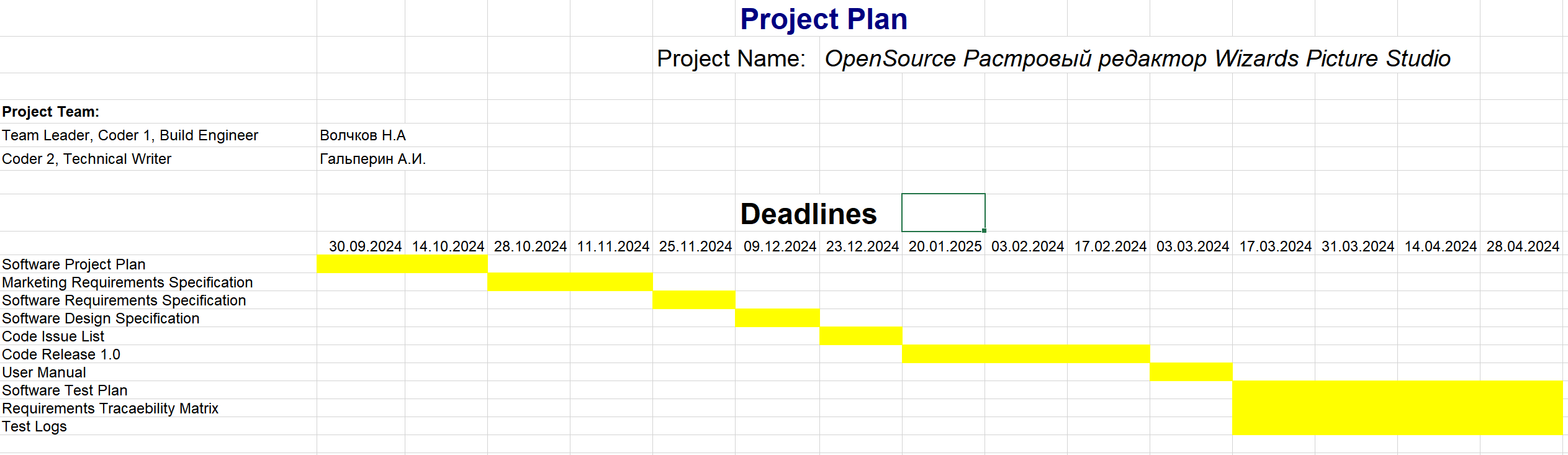


Рисунок 1 – План проекта с финальными датами

Т а б л и ц а 1 - Команда проекта

|  |  |
| --- | --- |
| **Команда** | |
| **Team Leader** | **Волчков Н.** |
| **Coder 1** | **Гальперин А.** |
| **Coder 2** | **Волчков Н.** |
| **Build Engineer** | **Гальперин А.** |
| **Technical Writer** | **Волчков Н.** |

На рисунке 1 можно увидеть план проекта с названием этапов, их описания и финальных дат выполнения. В таблице 1 перечислен состав команды и роли участников.

# Разработка регламента проведения инспекции

## Верификация рабочих продуктов

Верификация рабочих продуктов — важный элемент процесса обеспечения их качества. Современные методы программирования предполагают использование стандартов, подходов и механизмов для проведения таких верификаций в формате инспекций.

Инспекция представляет собой мероприятие, направленное на обеспечение качества рабочих продуктов проектов, включая разработку ПО и другие виды деятельности. Она проводится разработчиками, а при необходимости – с участием представителей заказчика.

Цели инспекции:

* Выявление ошибок в функциях, логике, содержании или реализации рабочих продуктов на ранних стадиях, предотвращая их распространение.
* Уточнение замысла или реализации продукта для всех заинтересованных сторон через их вовлеченность.
* Оценка, оптимизация и улучшение качества рабочего продукта.

## Критерии отнесения к формальной инспекции

В таблице 2 перечислены критерии отнесения к формальной инспекции.

Т а б л и ц а 2 – критерии для различных типов рабочих продуктов

|  |  |  |
| --- | --- | --- |
|  | **Формальная инспекция в случае CRUD-**операций **над** | **Неформальная инспекция в случае CRUD-операций над** |
| **Требования** | >3 требованием | При любом изменении |
| **Документы дизайна** | >3 объектом дизайна внешнего вида готового продукта |
| **Код** | >50 строчками |
| **Тесты** | >5 тестами |

## Перечень ролей участников инспекции и их обязанности

В таблице 3 перечислены роли и обязанности, которые они подразумевают.

Т а б л и ц а 3 - Роли и их описание

|  |  |
| --- | --- |
| **Роль** | **Обязанности** |
| Автор (Coder 1) | Участник, сделавший инспектируемые изменения в существующем рабочем продукте |
| Инспектор, проверяющий (Coder 2) | Участник, ответственный за эффективную проверку инспектируемого рабочего продукта. |

Так как в команде два человека, то роли не фиксированы и менялись в зависимости от того, кто внес изменения в проект (тот выступал в роли автора, соответственно).

## Этапы инспекции

1. Планирование инспекции.

Определяются дата, время, формат (очный/заочный) и платформа (при заочной инспекции).

1. Подготовка к инспекции.

Определяются дата, время, формат (очный/заочный) и платформа (при заочной инспекции).

1. Проведение собрания.

Обсуждаются замечания инспектора. Присутствуют автор и инспектор продукта.

1. Завершение инспекции.

При необходимости доработок: автор фиксирует замечания, разрабатывает план исправлений и согласовывает его с инспектором. При отсутствии доработок: инспектор одобряет слияние ветки в основную («develop»).

## Порядок организации

1. **Автор** загружает изменения в репозиторий и приглашает инспектора для проверки.
2. **Инспектор** готовиться к инспекции и составляет протокол ошибок, которые отправляет автору**.**
3. **Автор** исправляет ошибки и отправляет исправленную версию кода инспектору.
4. **Инспектор** проверяет корректность исправлений и подтверждает их, завершая инспекцию.

## Порядок подготовки и проведения

1. **Уведомление:** Автор уведомляет инспектора о назначении проверки за 1-2 дня.
2. **Подготовка:** Инспектору предоставляется от 3 до 7 дней для изучения изменений и составления протокола в зависимости от объема изменений и сложности кода.
3. **Собрание по инспекции:** оперативно обсуждаются ошибки и способы их исправления.
4. **Исправления и проверка:** после исправления ошибок автор отправляет обновленный код инспектору, который проверяет и подтверждает их корректность.

## Перечень статусов и степени важности замечаний

В таблице 4 перечислены замечания и их описания.

Т а б л и ц а 4 - Замечания и их описание

|  |  |
| --- | --- |
| **Статус замечания** | **Описание** |
|  | |
| **Дефект (Defect)** | Проблема, которая найдена на фазе, отличной от той, на которой внесена. |
| **Ошибка (Error)** | Ошибка присутствует в коде на той же фазе разработки. |
| **Комментарий (Comment)** | Предложения, улучшения и рекомендации для будущего релиза. |
| **Замечание для исследования (Investigate)** | Проблема требует дополнительного анализа и не может быть решена сразу. |

В таблице 5 перечислены степени важности замечаний и их описания.

Т а б л и ц а 5 - Степень важности и их описание

|  |  |
| --- | --- |
| **Степень важности** | **Описание** |
|  | |
| **Критическая (Critical)** | Ошибка приводит к невозможности запуска программы. |
| **Особо важная (Major)** | Ошибка влияет на выполнение программы и вызывает сбои. |
| **Средняя (Moderate)** | Ошибка проявляется в специфических условиях и временно. |
| **Мелкая (Minor)** | Ошибка во вводе/выводе данных или валидации данных. |
| **Другие (Other)** | Предложения по улучшению производительности или оптимизации кода. |

## Порядок верификации замечаний

После завершения обсуждения в рамках инспекции:

1. Автор исправляет ошибки, отмечая все исправления в протоколе;
2. Инспектор проверяет исправления, подтверждая, что все замечания учтены в обновленном коде;
3. После подтверждения исправлений формальная инспекция считается успешно завершенной.

# Модель состояний задач

## Перечень возможных состояний задачи и их интерпретация

Каждая задача, являясь отражением делового процесса, проходит определенные состояния. Сначала идет создание задачи, потом идет выполнение работ по задаче, после выполнения задача завершается.

Ниже представлен перечень состояний задач для проекта:

* New – новая подзадача.
* Analysis – в процессе анализа. В это состояние подзадачу переводит член команды после того, как начнёт её анализ.
* Forward – в данном случае имеет значение «переданный на дальнейшую разработку». В это состояние задача переводится CCB после анализа при назначении задачи на разработку конкретному члену команды.
* Coding – кодирование. В это состояние задача переводится разработчиком при начале работы по кодированию, связанному с задачей.
* Inspected – проинспектировано. В это состояние задача переводится разработчиком после завершения кодирования и инспектирования изменений рабочего продукта.
* Integrated – интегрировано. Переводится членом команды, осуществляющим интеграцию изменений в основную ветку рабочего продукта после успешной интеграции этих изменений.
* Tested – протестировано. Переводится членом команды, осуществляющим тестирование изменений в рабочий продукт (tester).
* Closed – закрыто. В это состояние задача переводится CCB по результатам отчёта о тестировании сделанных изменений.

## Правила создания новой задачи

1. Кто может создавать задачи:

Задачи могут создавать Coder 1 или Coder 2, в зависимости от их ответственности в проекте;

Также, в случае критической ошибки или необходимости улучшения процесса, задачи могут инициироваться Team Lead.

1. Когда следует создавать задачу:

Для исправления ошибок в коде или тестах:

Для добавления новых функций или улучшения существующего функционала;

Для реализации изменений по результатам обсуждений в ходе код-ревью или инспекций;

Для создания задачи в соответствии с фидбеком от QA-инженеров или пользователей системы.

1. Инструменты для создания задач:

GitHub Issues, где задачи группируются по релизам и критическим обновлениям;

Используются шаблоны задач для более четкого описания требований и ожидаемых результатов.

## Правила перехода задачи из состояния в состояние

Состояния задач всегда идут последовательно друг за другом, в некоторых случаях пункты могут опускаться или повторяться. Схема перехода из состояния в состояние показана на рисунке 2.

Изображение выглядит как диаграмма, текст, снимок экрана, линия

Автоматически созданное описание

Рисунок 2 – Состояния задачи

# Презентация проекта

На рисунке 3 представлена титульная страница презентации.

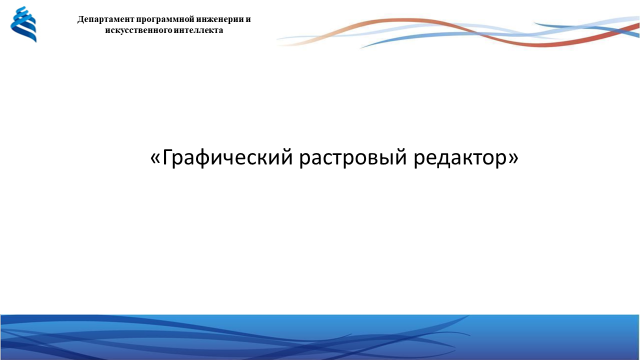


Рисунок 3 - Титульная страница

На рисунке 4 демонстрируется команда разработки и описание разрабатываемого проекта.



Рисунок 4 - Команда

На рисунке 5 изображены ключевые преимущества проекта.

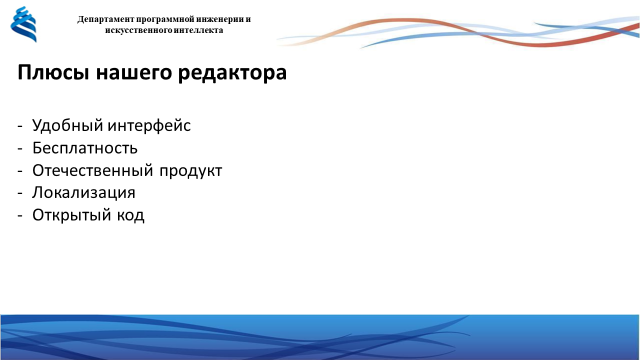


Рисунок 5 – Преимущества проекта

Перспективы развития проекта представлены на рисунке 6.

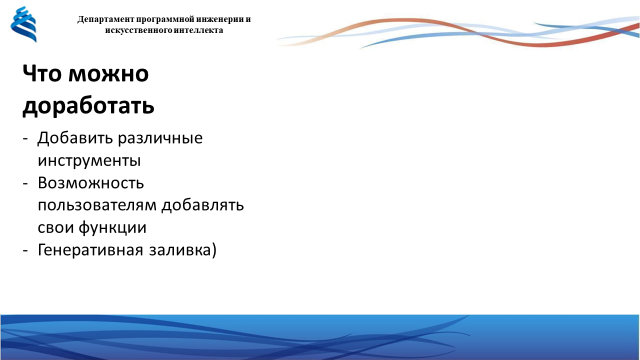


Рисунок 6 – Перспективы развития проекта

На рисунке 7 продемонстрирован прототип проекта.

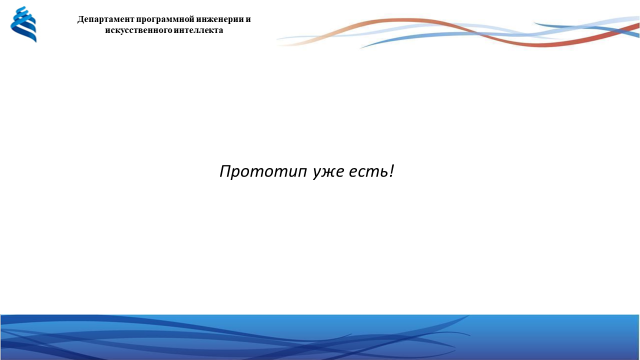


Рисунок 7 – Прототип проекта

Последний слайд на рисунке 8.

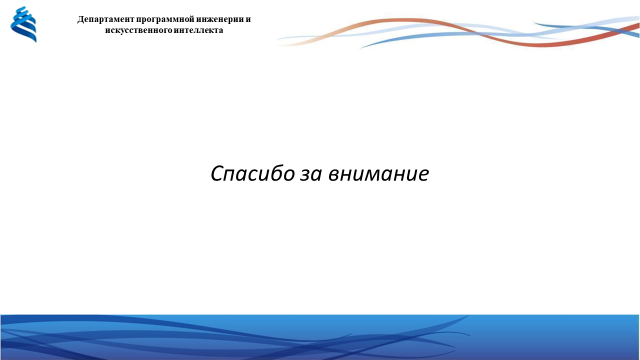


Рисунок 8 – «Спасибо за внимание»

# Требования к проекту

Программный продукт «Графический растровый редактор» состоит из следующих подсистем:

* Editor (ED) – подсистема «Редактор» для базовой работы с растровыми изображениями (рисование, редактирование, слои).
* AI Fill (AI) – подсистема «Нейросетевая заливка», которая отвечает за генерацию фрагментов изображения по текстовому описанию.
* Configuration (CFG) – подсистема «Управление конфигурацией», отвечающая за хранение настроек (кисть, формат сохранения и т. д.).

## Требования к подсистеме Editor (ED)

Требование REQ\_ED\_001

Приложение должно предоставлять инструменты рисования: кисть (с настраиваемым размером и цветом), ластик, пипетку для выбора цвета.

Требование REQ\_ED\_002

Пользователю должна быть доступна работа со слоями (создание, удаление, отображение/скрытие, переупорядочивание).

Требование REQ\_ED\_003

При нажатии на кнопку «Создать» пользователь должен видеть диалог для задания ширины и высоты нового холста; созданное изображение открывается в отдельной вкладке редактора.

Требование REQ\_ED\_004

При нажатии на кнопку «Сохранить» пользователь должен иметь возможность выбрать формат (PNG или JPEG) и путь сохранения; редактор сохраняет текущее состояние холста.

## Требования к подсистеме AI Fill (AI)

Требование REQ\_AI\_001

При нажатии на инструмент «Нейросетевая заливка» пользователю должно отображаться поле для ввода текстового описания.

Требование REQ\_AI\_002

При подтверждении пользователем текстового описания, система должна сгенерировать фрагмент изображения соответствующего размера (с учётом выделенной области на холсте) и заменить им текущую выделенную область.

Требование REQ\_AI\_003

Процесс генерации должен быть асинхронным: интерфейс не должен блокироваться, должна быть отображена индикаторная панель или сообщение о процессе генерации.

Требование REQ\_AI\_004

Подсистема должна обеспечивать корректное масштабирование сгенерированных фрагментов, чтобы совпасть с размерами выделенной области без искажений пропорций.

## Требования к подсистеме Configuration (CFG)

Требование REQ\_CFG\_001

Приложение должно сохранять настройки пользователя (предпочитаемый размер кисти, цвет и т. д.) и восстанавливать их при повторном запуске.

Требование REQ\_CFG\_002

Приложение должно иметь возможность сохранять путь для автосохранения файлов и обеспечивать возможность восстановления работы после сбоя.

# Архитектура проекта

Архитектура программного обеспечения относится к фундаментальным структурам программной системы и дисциплине создания таких структур и систем. Каждая структура включает элементы программного обеспечения, отношения между ними, а также свойства как элементов, так и отношений. Архитектура программной системы – это метафора, аналогичная архитектуре здания. Он функционирует как план для системы и проекта разработки, в котором излагаются задачи, которые должны быть выполнены командами разработчиков.

На рисунке 9 представлена архитектурно-контекстная диаграмма программного средства.

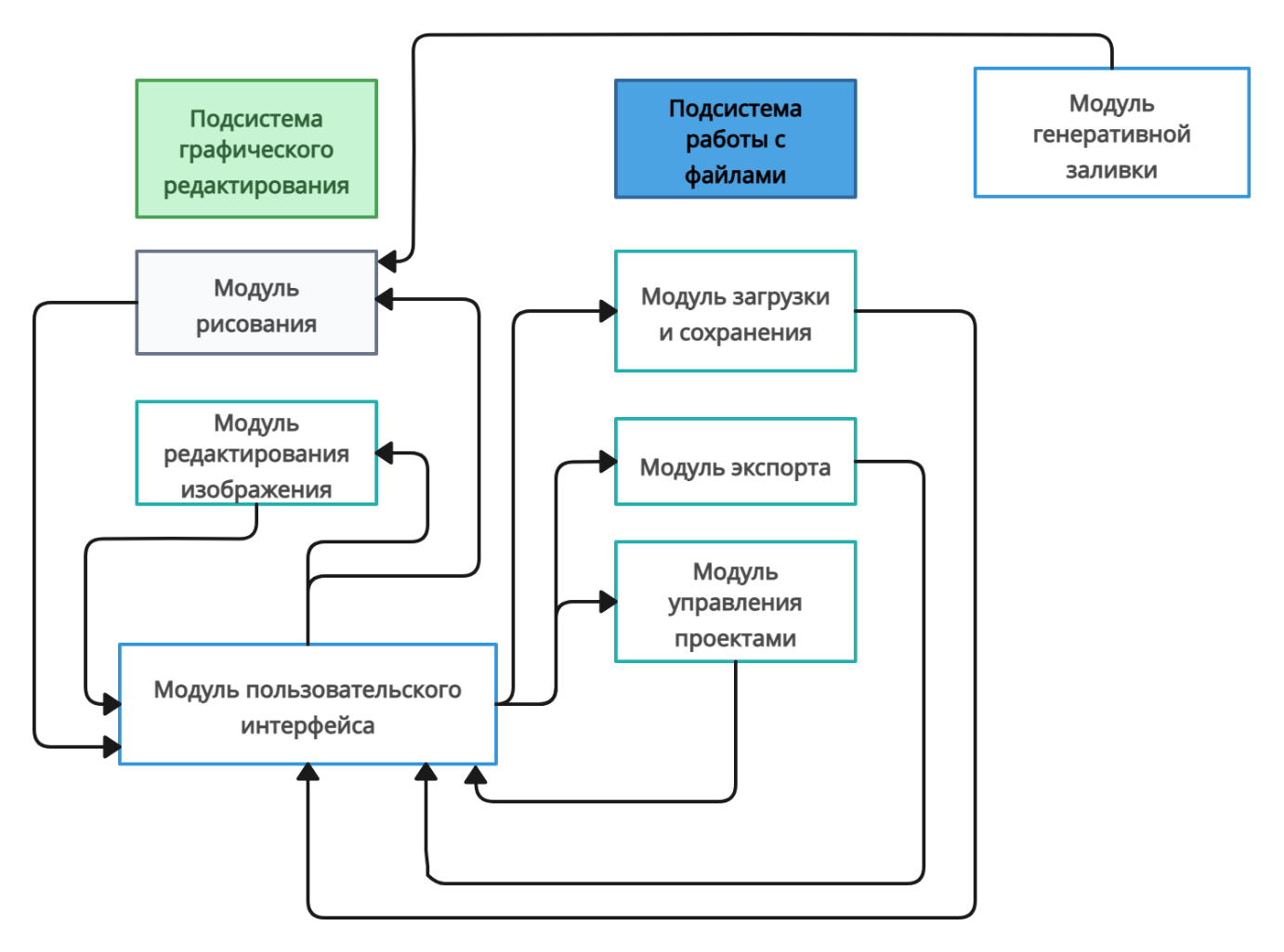
****

Рисунок 9 - Архитектурно-контекстная диаграмма программного средства

На рисунке 10 представлена Use-case диаграмма взаимодействия клиента с системой.

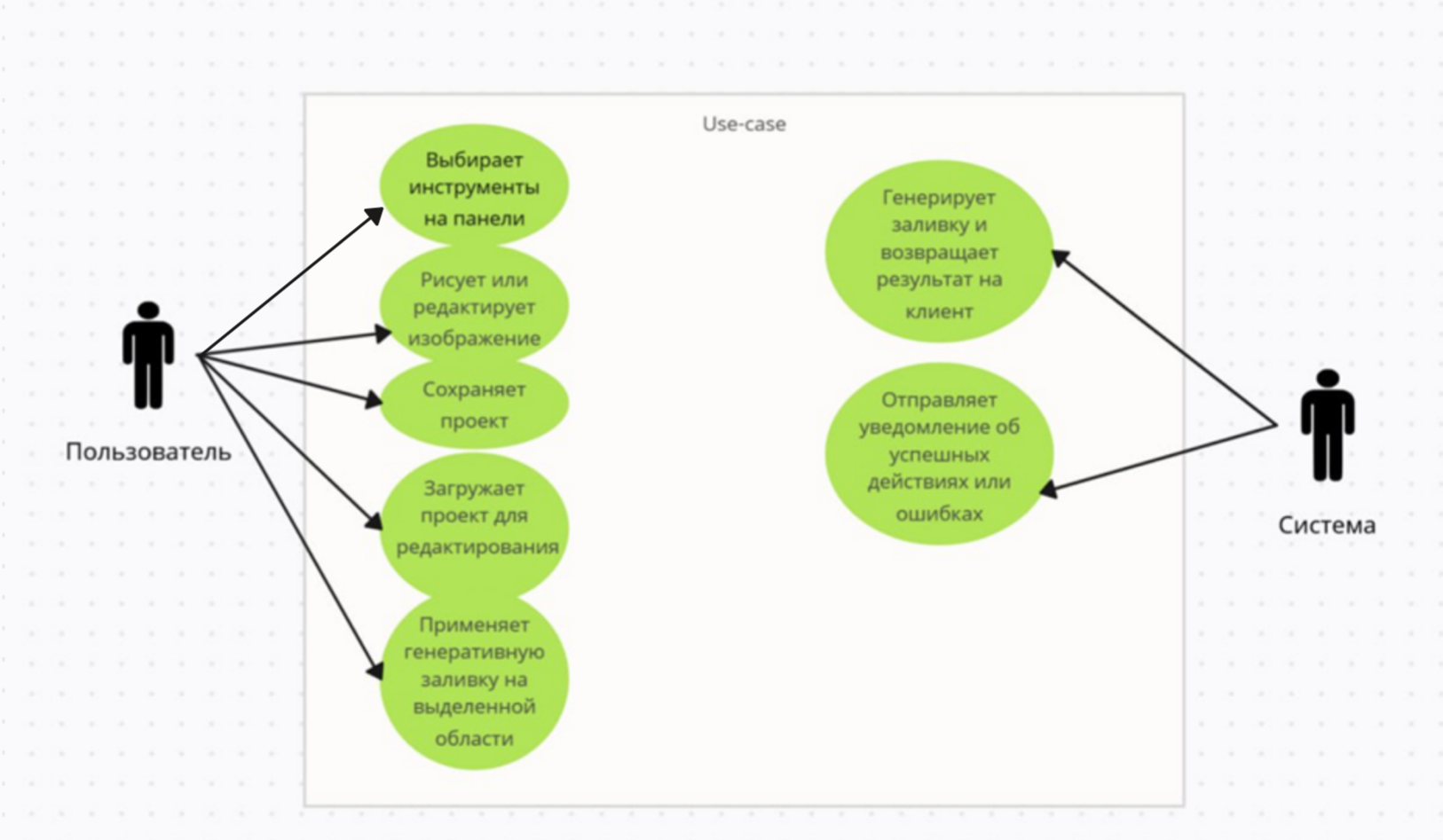


Рисунок 10 - Use-case диаграмма

На рисунке 11 представлена ER-диаграмма.

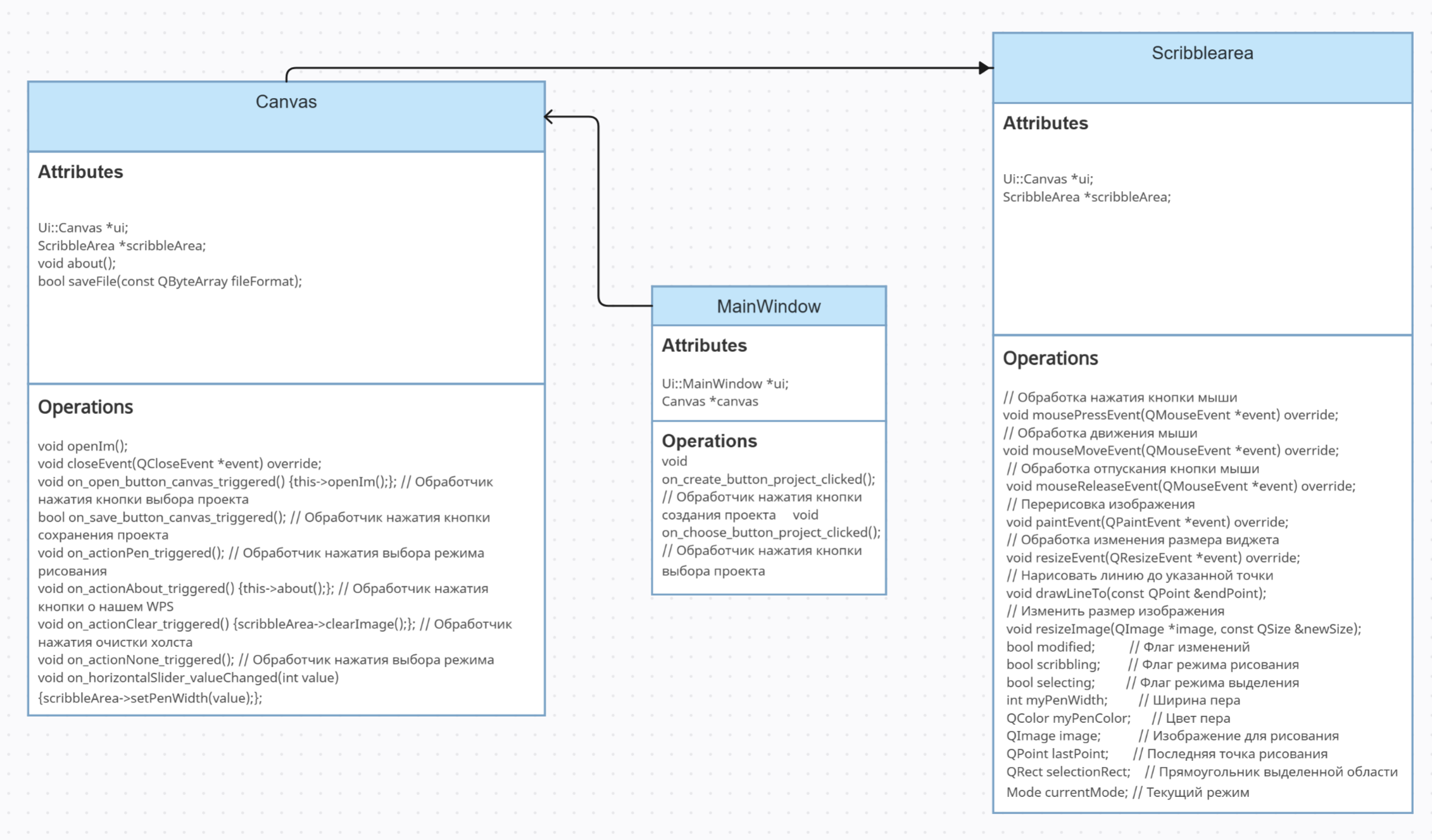


Рисунок 17 – ER-диаграмма

# Измерения проекта

Контроль за производственным процессом и его результатами является ключевым видом деятельности на современном предприятии, производящем программное обеспечение на заказ. В силу специфики такого продукта, как программное обеспечение, для оценки эффективности процесса разработки и качества конечного продукта применяются особые методы. Комплекс мероприятий, направленных на количественную оценку эффективности работы компании, называется программой измерений компании. Программа измерений выполняется как в рамках отдельных проектов, осуществляемых компанией, так и в рамках определённых видов деятельности компании

**7.1 Метрики для оценки эффективности процесса разработки**

project\_size – размер разрабатываемого, сумма количества документов дизайна, количества требований, количества тестов и количества строк кода без комментариев.

error\_count – общее количество найденных ошибок в процессе разработки.

errors – общее количество найденных ошибок на этапе разработки

missed\_errors – общее количество не найденных ошибок на этапе разработки.

**7.2 Метрика качества программного продукта**

c\_req – количество требований, e\_req – количество требований в которых допущена ошибка.

c\_design – количество документов дизайна (диаграмм, схем и т.д.), e\_design – количество документов дизайна в которых допущена ошибка.

e\_coding – количество ошибок допущенных в коде.

c\_test – количество тестов, e\_test -количество тестов в которых допущена ошибка.

Каждое из слагаемых может выступать в роли отдельной метрики.

# Перечень задач проекта

Задачи для подсистемы «UI»

Задача TASK\_UI\_001

Название: Реализация области рисования.

Описание: Реализовать область рисования в графическом редакторе. Обеспечить функционал для рисования с использованием инструментов, таких как кисть и ластик. Область должна поддерживать масштабирование.

Задача TASK\_UI\_002

Название: Меню настройки инструментов.

Описание: Реализовать меню настройки параметров инструментов рисования. Пользователь должен иметь возможность изменять цвет, размер кисти. Меню должно быть интуитивно понятным и визуально привлекательным.

Задачи для подсистемы «Генеративная заливка»

Задача TASK\_GZ\_001

Название: Реализация инструмента генеративной заливки.

Описание: Реализовать функционал генеративной заливки для выделенных областей изображения. Пользователь должен иметь возможность настроить параметры заливки, такие как промт и область.

Задача TASK\_GZ\_002

Название: Уведомление об ошибке при генеративной заливке.

Описание: Реализовать систему уведомлений, сообщающую пользователю об ошибках в процессе генеративной заливки (например, некорректный промт или "У вас закончились бесплатные генерации").

Задачи для подсистемы «Файловая система»

Задача TASK\_FS\_001

Название: Сохранение изображений.

Описание: Реализовать функционал сохранения созданных изображений в популярных форматах (PNG, JPEG, BMP). Пользователь должен иметь возможность выбрать путь сохранения и имя файла.

Задача TASK\_FS\_002

Название: Загрузка изображений.

Описание: Реализовать возможность загрузки изображений для редактирования. Приложение должно поддерживать проверку форматов и размера загружаемых файлов.

# Рекомендации по кодированию

Для создания качественного кода на любом языке программирования, обладающего таким свойствами, как удобочитаемость (readability) и понятность (understandability), необходимо следовать хорошо определённым стандартам и руководствам. Особенно это актуально при коллективной разработке программ.

Любой стандарт кодирования призван определить набор правил, которые способствуют разработке более единообразного кода и минимизации числа общераспространенных ошибок в нем, не ущемляя при этом права разработчика на творчество.

## Рекомендации

1. Рекомендуется не допускать строки кода длиной более 120 символов. Это улучшит читаемость кода и позволит не тратить лишнее время на пролистывание ползунка до окончания строки.
2. При написании условий и циклов рекомендуется использовать фигурные скобки, даже если тело условия/цикла состоит из одного выражения. Это более явно обозначит тело условия/цикла и при необходимости добавления новой логики избавит от необходимости добавления скобок.
3. При обработке коллекций, требующей несколько операций, рекомендуется использовать возможности библиотеки Stream API (Java). Это улучшит читаемость кода и сократит кол-во строк.
4. Рекомендуется использовать конструкции `guard` для проверки условий, которые могут привести к раннему завершению выполнения функции (Swift). Это улучшит читаемость кода, делая основную логику более линейной и понятной.
5. При обработке коллекций рекомендуется использовать замыкания и функции высшего порядка - map, filter, reduce (Swift). Это может улучшить ясность и краткость кода.

## Требования

1. Требуется классы и интерфейсы именовать в UpperCamelCase, методы и переменные в – lowerCamelCase, и константы в – SNAKE\_CASE. Это позволит по стилю нейминга сразу понимать тип конструкции языка.
2. Требуется в импортах указывать конкретную сущность, не использовать \*. Это позволит быстрее определять откуда берется та или иная сущность.
3. Требуется давать осмысленные названия сущностям. Название сущности должно давать о понимании о её предназначении в коде.
4. Для полей, описывающих состояние класса, требуется использовать как можно более строгий модификатор доступа. Это позволит обеспечить инкапсуляцию и защитить внутреннюю логику.
5. Комментировать сложные блоки кода, алгоритмы, оставлять ToDo.

## Запреты

1. Запрещается оставлять ресурсы без закрытия. Все открытые ресурсы нужно закрывать, если они больше не используются.
2. Запрещается использовать магические числа и строки. Все значения должны быть вынесены в константы или конфигурацию.