**АННОТАЦИЯ**

Исполнитель: студент группы ИКБО-07-21 Чирва А.Ю.

Руководитель: доцент кафедры МОСИТ, к.т.н, доцент Лобанов А.А.

Выпускная квалификационная работа на тему «Система для структуризации нарративного дизайна с интеграцией со средой разработки».

Общее количество страниц включая приложения: 78.

Работа включает в себя 31 рисунок, 19 таблиц, 7 приложений.

Список источников информации включает 32 позиции.

Ключевые слова: нарративный дизайн, плагин, Unreal Engine, система структуризации.

СОДЕРЖАНИЕ

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 9](#_Toc199761341)

[ВВЕДЕНИЕ 10](#_Toc199761342)

[1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ 12](#_Toc199761343)

[1.1 Анализ предметной области 12](#_Toc199761344)

[1.1.1 Описание предметной области 12](#_Toc199761345)

[1.1.2 Функциональные, нефункциональные и пользовательские  
требования 13](#_Toc199761346)

[1.2 Анализ систем-аналогов 15](#_Toc199761347)

[1.2.1 Достоинства и недостатки систем-аналогов 15](#_Toc199761348)

[1.2.2 Обоснование выбора своей разработки 16](#_Toc199761349)

[1.2.3 Матрица рисков 17](#_Toc199761350)

[1.3 Техническое задание 18](#_Toc199761351)

[2 ПРОЕКТНЫЙ РАЗДЕЛ 19](#_Toc199761352)

[2.1 Функции системы 19](#_Toc199761353)

[2.1.1 Дерево функций 19](#_Toc199761354)

[2.1.2 Диаграмма вариантов использования 19](#_Toc199761355)

[2.1.3 Диаграмма последовательности 20](#_Toc199761356)

[2.2 Проектирование архитектуры системы 22](#_Toc199761357)

[2.2.1 Архитектура разрабатываемого ПО 22](#_Toc199761358)

[2.2.2 Архитектурная диаграмма 24](#_Toc199761359)

[2.2.3 Матрица доступа и роли 25](#_Toc199761360)

[3 ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 27](#_Toc199761361)

[3.1 Анализ выбранных средств разработки 27](#_Toc199761362)

[3.1.1 Технологии реализации 27](#_Toc199761363)

[3.2 Описание пользовательского взаимодействия 29](#_Toc199761364)

[3.2.1 Путь пользователя 29](#_Toc199761365)

[3.2.2 Описание пользовательского интерфейса 31](#_Toc199761366)

[3.3 Информационное обеспечение 34](#_Toc199761367)

[3.3.1 Логическая модель базы данных 34](#_Toc199761368)

[3.3.2 Словарь данных 34](#_Toc199761369)

[3.3.3 Диаграмма классов 36](#_Toc199761370)

[3.4 Программа и методика испытаний 36](#_Toc199761371)

[3.4.1 Объект испытаний 37](#_Toc199761372)

[3.4.2 Цель испытаний 37](#_Toc199761373)

[3.4.3 Объём испытаний 37](#_Toc199761374)

[3.4.4 Методика проведения испытаний 38](#_Toc199761375)

[3.4.5 Требования по испытаниям программных средств 39](#_Toc199761376)

[3.4.6 Перечень работ, проводимых после завершения испытаний 39](#_Toc199761377)

[3.4.7 Условия и порядок проведения испытаний 39](#_Toc199761378)

[3.4.8 Материально-техническое обеспечение испытаний 39](#_Toc199761379)

[3.4.9 Отчётность 40](#_Toc199761380)

[3.5 Расчёт вычислительной и ёмкостной сложности 40](#_Toc199761381)

[4 ЭКОНОМИЧЕСКИЙ РАЗДЕЛ 42](#_Toc199761382)

[4.1 Организация и планирование работ по теме 42](#_Toc199761383)

[4.1.1 Организация работ 42](#_Toc199761384)

[4.1.2 График проведения работ 44](#_Toc199761385)

[4.2 Расчёт стоимости проведения работ по теме 45](#_Toc199761386)

[4.2.1 Сырьё и материалы 45](#_Toc199761387)

[4.2.2 Статья «Основная заработная плата» 46](#_Toc199761388)

[4.2.3 Статья «Дополнительная заработная плата» 47](#_Toc199761389)

[4.2.4 Статья «Страховые взносы» 48](#_Toc199761390)

[4.2.5 Статья «Амортизационные отчисления» 48](#_Toc199761391)

[4.2.6 Прочие расходы 49](#_Toc199761392)

[4.2.7 Полная себестоимость работ 50](#_Toc199761393)

[ЗАКЛЮЧЕНИЕ 51](#_Toc199761394)

[CONCLUSION 52](#_Toc199761395)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 53](#_Toc199761396)

[ПРИЛОЖЕНИЯ 57](#_Toc199761397)

[ПРИЛОЖЕНИЕ А 58](#_Toc199761398)

[ПРИЛОЖЕНИЕ Б 61](#_Toc199761399)

[ПРИЛОЖЕНИЕ В 66](#_Toc199761400)

[ПРИЛОЖЕНИЕ Г 67](#_Toc199761401)

[ПРИЛОЖЕНИЕ Д 70](#_Toc199761402)

[ПРИЛОЖЕНИЕ Е 73](#_Toc199761403)

[ПРИЛОЖЕНИЕ Ж 74](#_Toc199761404)

# 

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

|  |  |
| --- | --- |
| БД | – База данных – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных. |
| СУБД | – Система управления базами данных |
| API | – Application programming interface |
| IDE | – Integrated Development Environment – интегрированная среда разработки |
| JSON | – JavaScript Object Notation |
| MVVM | – Паттерн проектирования Model-View-ViewModel |
| SQL | – Декларативный язык программирования для реляционных баз данных |
| UE | – Unreal Engine |

# ВВЕДЕНИЕ

Нарративный дизайн – это систематическое создание и структурирование сюжетного контента игры (историй, заданий, персонажей и других событий). Это связующее звено между игровым процессом и сценарием игры.

Нарративные дизайнеры – это роль в разработке игр, которая появилась недавно. Раньше эту роль выполняли геймдизайнеры, но с ростом игровых проектов понадобились отдельные специалисты, отвечающие за создание логичного и последовательного игрового сюжета. Нарративные дизайнеры отвечают за то, чтобы игровые механики выглядели уместно в игровом мире, принимая на себя часть обязанностей как сценариста, так и геймдизайнера.

Однако с увеличением сложности и масштабов современных игр возрастает и проблема эффективного взаимодействия между нарративными дизайнерами и программистами. Традиционные методы коммуникации и интеграции сюжетного контента в игровой движок часто требуют значительных затрат времени и усилий, так как работа ведется разрозненно в разных программах или вынуждает дизайнеров осваивать тяжеловесные технические инструменты движка. Это не только усложняет процесс разработки, но и увеличивает вероятность ошибок и несоответствий, приводя к потерям временных и, как следствие, финансовых ресурсов.

Целью выпускной квалификационной работы является создание системы структуризации нарративного дизайна для решение описанной проблемы. Она состоит из двух взаимосвязанных частей: десктопного приложения, предназначенного специально для нарративных дизайнеров, и плагина для игрового движка Unreal Engine 5. Приложение позволяет дизайнерам легко создавать и редактировать описание персонажей, локаций, событий, а затем экспортировать данные в удобном формате (JSON) непосредственно в среду разработки. Плагин же обеспечивает быструю и удобную интеграцию этих данных в Unreal Engine, позволяя программистам связывать сущности нарративного дизайна с конкретными игровыми объектами и элементами. Таким образом, данная система значительно упрощает взаимодействие между дизайнерами и программистами, сокращает время на интеграцию контента и минимизирует ошибки, делая процесс разработки более эффективным и продуктивным.

В Приложении Ж представлен графический материал (презентация) выпускной квалификационной работы.

В процессе написания выпускной квалификационной работы автор руководствовался следующими нормативными актами:

1. «О защите населения и территории от чрезвычайных ситуаций

природного и техногенного характера» от 21.12.1994 № 68-ФЗ.

2. «Об основах охраны здоровья граждан в Российской Федерации» от

21.11.2011 № 323-ФЗ.

3. «О гражданской обороне» от 12.02.1998 № 28-ФЗ.

4. Приказ Минздравсоцразвития РФ от 04.05.2012 № 477н «Об

утверждении перечня состояний, при которых оказывается первая помощь, и перечня мероприятий по оказанию первой помощи».

5. Трудовой кодекс Российской Федерации от 30.12.2001 № 197-ФЗ.

6. СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к

персональным электронно-вычислительным машинам и организации работы».

В процессе написания выпускной квалификационной работы автор руководствовался следующими стандартами: ГОСТ 7.32-2017 [1], ГОСТ Р 7.0.100-2018 [2], СМКО МИРЭА 7.5.1/03.П.30-19 [3], СМКО МИРЭА 7.5.1/03.П.67-19 [4], ФГОС ВО 3++ [5].

# ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

## Анализ предметной области

### Описание предметной области

Современная игровая индустрия стремительно развивается: возрастает сложность игровых проектов, увеличиваются масштабы игровых миров, а структура повествования становится всё более сложной. Появляются нелинейные сюжеты, требующие высокого уровня проработки. Нарративный дизайн как отдельное направление сформировался недавно, и вместе с этим возникла необходимость в инструментах, которые соответствуют специфике этой деятельности [6].

На практике нарративные дизайнеры используют разнообразные инструменты: от общих визуальных досок (например, Miro или Weje) до специализированных программ (таких как Story Architect). Однако эти решения либо не предоставляют специализированного функционала для структурирования нарратива, либо требуют глубоких технических знаний, так как работают непосредственно в среде игрового движка. В результате дизайнеры вынуждены либо адаптироваться к непривычным инструментам, либо передавать данные программистам вручную, что замедляет работу.

Системы, предназначенные для управления нарративом, как правило, не поддерживают экспорт данных в игровые движки и не масштабируются под проекты разных уровней — от инди до AAA. Также отсутствует чёткое разделение интерфейсов и функционала между участниками команды — дизайнерами и разработчиками.

Особенностью предметной области является наличие двух ролей с разными задачами и уровнями вовлечённости: нарративный дизайнер работает с сюжетной структурой, а программист отвечает за техническую реализацию в игровом движке. Это требует построения архитектуры системы с чётко выделенными интерфейсами, что позволяет пользователям с разными ролями работать совместно, при этом не вызывая конфликтов, а также поддерживать синхронизацию данных между компонентами создаваемого игрового проекта.

Модель бизнес-процесса представлена на рисунках А.1 – А.5 Приложения А в виде диаграмм в нотации IDEF0.

### Функциональные, нефункциональные и пользовательские требования

Пользовательские истории (User Story) — это краткое описание функциональности продукта с точки зрения конечного пользователя [7].

Пользовательские истории представлены в таблице 1.1.

Таблица 1.1 – Пользовательские истории

|  |  |  |
| --- | --- | --- |
| Заказчик | Действие | Цель |
| Нарративный дизайнер | создавать и редактировать локации, персонажей и события в проекте. | С целью организовать и структурировать нарратив игры, обеспечивая логичность и последовательность истории. |
| Нарративный дизайнер | визуализировать последовательность событий, их связи с локациями и персонажами. | С целью лучше понимать общую картину нарратива и легко вносить изменения. |
| Нарративный дизайнер | экспортировать нарративные данные (локации, события, персонажи) в Unreal Engine 5. | С целью интеграции нарратива в игровой движок для дальнейшей работы над игрой. |
| Разработчик на Unreal Engine 5 | импортировать нарративные данные из системы. | С целью синхронизировать работу нарративного дизайнера и реализацию в игровом мире. |
| Разработчик на Unreal Engine 5 | Привязать данные к объектам внутри движка | С целью ускорения навигации при работе с нарративом |
| Нарративный дизайнер | иметь возможность совместной работы с другими дизайнерами. | С целью эффективно разрабатывать сложные нарративы в команде. |

На основе пользовательских историй, представленных в таблице 1.1, были сформулированы следующие функциональные требования:

* У дизайнера должна быть возможность создавать, редактировать, удалять персонажей.
* У дизайнера должна быть возможность создавать, редактировать, удалять локации.
* У дизайнера должна быть возможность создавать, редактировать, удалять события.
* У дизайнера должна быть возможность связывать между собой события, персонажей, локации.
* У дизайнера должна быть возможность упорядочивать события в хронологическом порядке.
* У дизайнера должна быть возможность экспортировать информацию в Unreal engine 5.
* У разработчика должна быть возможность импортировать информацию в Unreal engine 5.
* У разработчика должна быть возможность создавать связи между сущностями в системе и объектами в движке.

После составления функциональных требований были также сформулированы нефункциональные требования проектируемого программного средства.

Нефункциональные требования относятся к атрибутам качества системы, которые определяют, как она работает, а не что именно она делает. В отличие от функциональных требований, которые определяют действия и задачи, которые должна выполнять система, нефункциональные требования фокусируются на общих характеристиках и поведении системы в различных условиях [8].

Нефункциональные требования к системе представлены в таблице 1.2.

Таблица 1.2 – Нефункциональные требования

|  |  |
| --- | --- |
| Критерий | Требование |
| Производительность | Система должна обеспечивать быструю загрузку и обработку данных (локаций, событий, персонажей), даже для крупных проектов с множеством элементов. |
| Время отклика при редактировании данных не должно превышать 1 секунды. |
| Масштабируемость | Система должна поддерживать работу с большими объемами данных (сотни локаций, событий и персонажей), без снижения производительности. |
| Возможность работы над проектами разных масштабов — от небольших инди-проектов до крупных AAA-игр. |
| Надёжность | Система должна обеспечивать сохранение данных без потерь, даже при сбоях, с регулярным созданием резервных копий. |
| Данные должны быть защищены от некорректных изменений и потери. |
| Совместимость | Система должна быть совместима с различными версиями Unreal Engine 5, а также поддерживать работу на основных платформах (Windows, macOS). |

## Анализ систем-аналогов

### Достоинства и недостатки систем-аналогов

Перед проектированием и разработкой системы необходимо провести анализ существующих аналогов.

Weje – аналог miro. Ограничений по доскам нет, можно создавать флоу чарты, редактировать файлы всей командой. Часто используется нарративными дизайнерами. Нет никакой интеграции с движком [9].

Quillscript Plugin – плагин для движка Unreal engine. Даёт возможность создавать сценарий и нарратив. Включая ветвление сюжета и интеграцию с кодом. Для использования нарративному дизайнеру необходимо скачать полновесную версию движка [10].

Сравнение системы с аналогами представлено в таблице 1.3.

Таблица 1.3 — Сравнение аналогичных решений

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | Weje | Quillscript Plugin | Разрабатываемая система |
| Тип продукта | Онлайн-доска, визуализатор | Плагин для Unreal Engine | Система из двух модулей: Десктопное приложение + плагин для Unreal Engine |
| Поддержка нарративного дизайна | Частично (без структуры и экспорта) | Да (создание сценария, ветвления, диалоги) | Да (полный цикл: персонажи, события, локации, связи) |
| Интеграция с Unreal Engine | Нет | Да (требует установку полной версии Unreal Engine) | Да (через плагин, без необходимости запуска полной среды Unreal Engine) |
| Удобство для дизайнера | Высокое, но общий инструмент | Низкое: работа в Unreal Engine требует технических знаний | Высокое: отдельное приложение с удобным UI |
| Удобство для разработчика | Низкое | Среднее | Высокое: разделение задач и форматов обмена |
| Формат передачи данных | Отсутствует | Встроенный Unreal Engine формат | JSON/XML |
| Поддержка совместной работы | Да | Частично (только через Unreal Engine и систему контроля версий) | Частично (с использованием системы контроля версий) |
| Система контроля версий | Встроенная через облачные платформы | Через Unreal Engine | Использует сторонние решения |
| Требования к ресурсам системы | Минимальные | Высокие (необходима поддержка полноценной версии Unreal Engine) | Умеренные (лёгкое приложение + плагин Unreal Engine) |

### Обоснование выбора своей разработки

Во всех системах для нарративного дизайнера, дизайнеру приходиться работать в полновесном движке, либо разработчику самостоятельно копировать работу нарративного дизайнера в движок. Разрабатываемая система позволяет дизайнеру и разработчику работать только с частью системы, в которой есть только необходимый им функционал и удобный для них интерфейс. Плагин спроектирован для движка Unreal Engine 5, так как он является наиболее часто используемым движком среди игровых студий среднего и большого размера, где наиболее часто есть отдельная роль нарративного дизайнера.

### Матрица рисков

Матрица рисков — это инструмент, который помогает определять, оценивать и управлять рисками в проекте. Матрица рисков изображается в таблице, где риски классифицируются, основываясь на вероятности их возникновения, а также степени влияния на проект. В матрице рисков также определяются наиболее эффективные стратегии реагирования на риск.

В матрице рисков отображаются следующие параметры:

* Риск (причина-риск-эффект). Описание риска, причин его возникновения и эффект, который он оказывает на проект.
* Стратегия. Способ разрешения потенциального риска.
* Основной план. Наиболее приоритетный набор действий для разрешения потенциального риска.
* Отходной план. Запасной способ разрешения потенциального риска.

Сформированная матрица рисков представлена в таблице 1.4.

Таблица 1.4 – Матрица рисков

|  |  |  |  |
| --- | --- | --- | --- |
| Риск (причина-риск-эффект) | Стратегия | Основной план | Отходной план |
| - Выход новой версии Unreal Engine без обратной совместимости  - плагин не работает  - остановка интеграции | Mitigate | - Поддерживать LTS‑ветку UE для производства  - Закрыть API‑слой плагина через абстракции, позволяющие быстро адаптировать код | - Создать отдельную команду «compat‑hotfix» для срочного портирования  - При необходимости заморозить обновление UE до выхода патча |
| - Сбой оборудования или ПО  - повреждение БД PostgreSQL  - потеря нарративных данных | Mitigate | - Ежедневное автоматическое резервное копирование  - Репликация БД на удалённый сервер | - Восстановление из последнего бэкапа  - Включение режима «только чтение» до завершения восстановления |
| - Недостаточная освоенность интерфейса дизайнерами  - ошибки при вводе данных  - увеличение времени разработки | Accept / Transfer | - Предоставить интерактивное onboarding‑видео  - Включить inline‑подсказки в UI | - Организовать экспресс‑обучение команды  - Назначить опытного пользователя в студии для оперативной помощи коллегам |

## Техническое задание

Техническое задание с учётом ГОСТ 19.201-78 «Единая система программной документации. Техническое задание. Требования к содержанию и оформлению» приведено в Приложении Б [11].

# ПРОЕКТНЫЙ РАЗДЕЛ

## Функции системы

### Дерево функций

Дерево функций — это наглядная картина функций, объединённых в логические группы с иерархическим разбиением каждой функции на более мелкие. Оно формируется с целью детального исследования функциональных возможностей системы и анализа совокупности функций, реализуемых на различных уровнях иерархии системы. Построенное дерево функций представлено на рисунке В.1 в Приложении В.

### Диаграмма вариантов использования

Для того, чтобы описать взаимодействие пользователей с системой, а также объекты и данные, манипулируемые данной системой, и их взаимодействие, необходимо воспользоваться моделями семейства UML [7].

Построенная диаграмма прецедентов представлена на рисунке 2.1

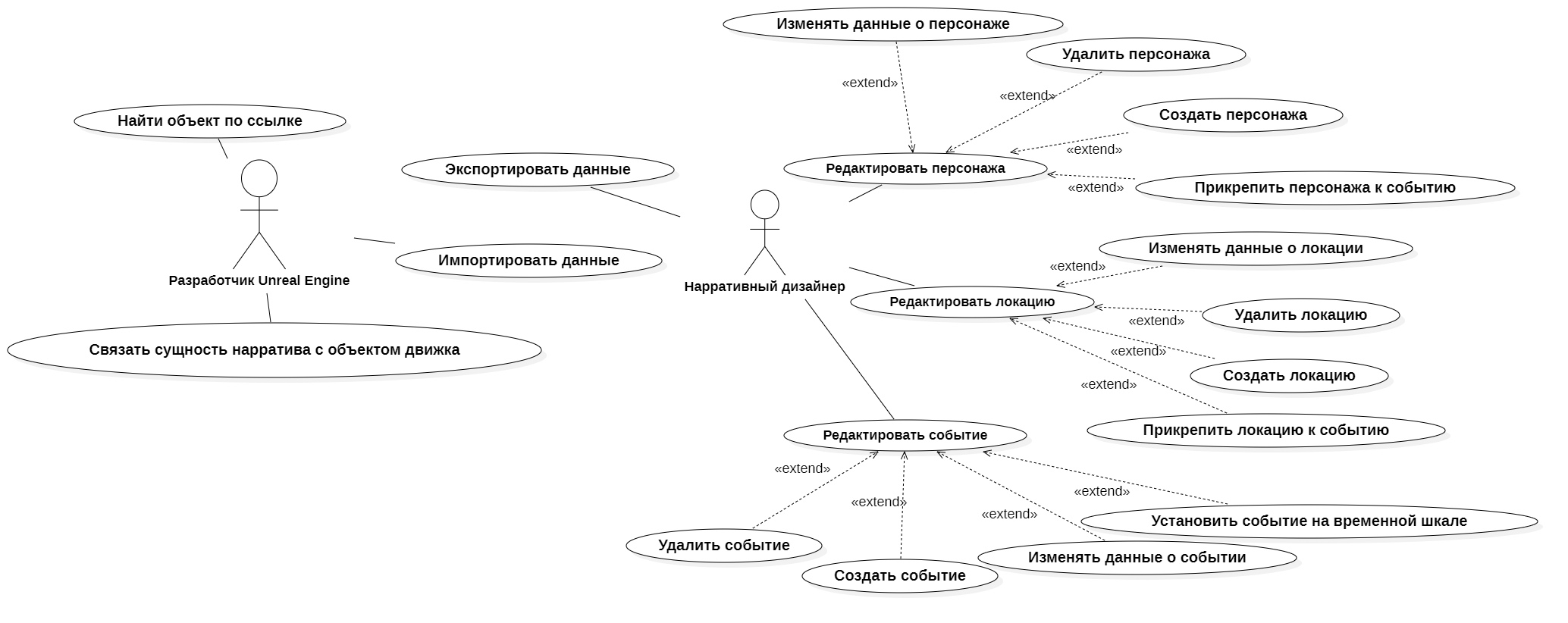


Рисунок 2.1 – Диаграмма вариантов использования

### Диаграмма последовательности

Чтобы подробно рассмотреть взаимодействия между компонентами системы, представленных на диаграмме вариантов использования используется диаграмма последовательности, отображающая систему в определенный момент времени [12].

Для представления диаграммы последовательности были выбраны две функции разрабатываемого программного продукта: создание нового события и импорт json-файла в плагине.

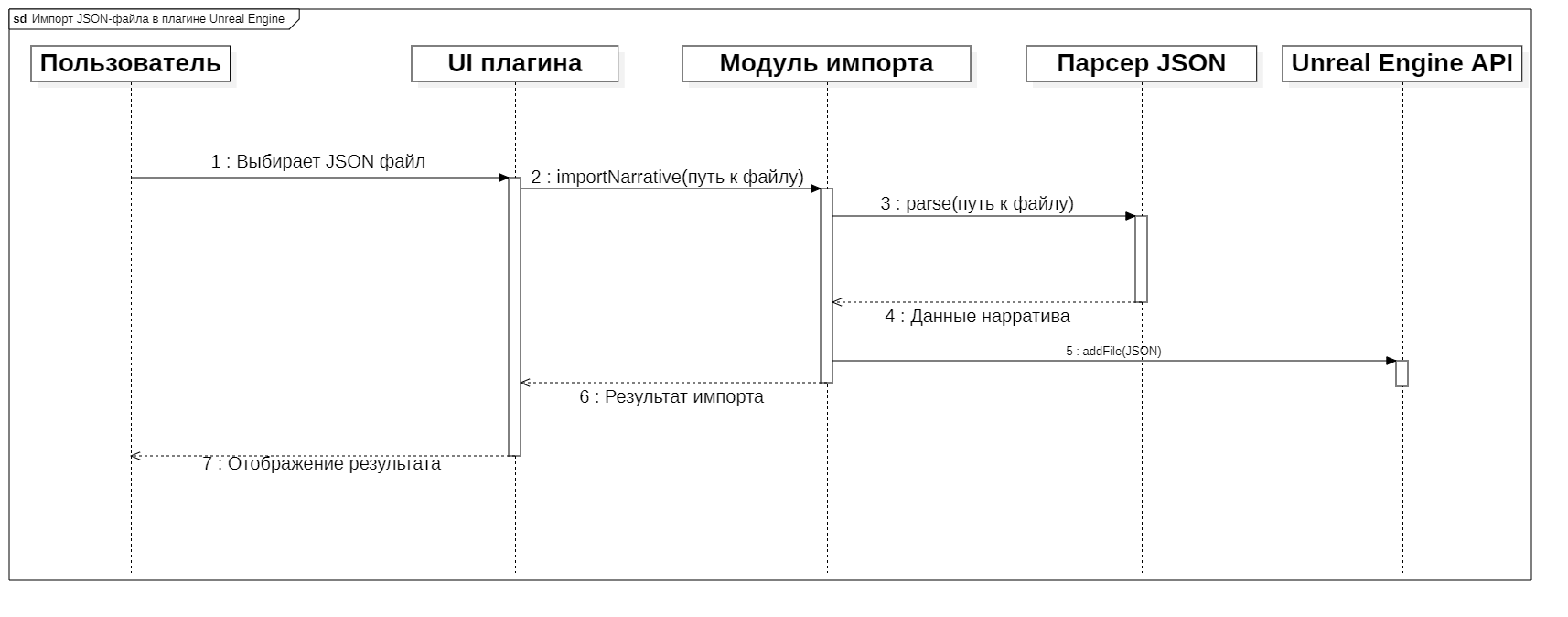
Диаграмма последовательности создания нового события представлена на рисунке 2.2.



Рисунок 2.2 – Диаграмма последовательности «Создание события»

Диаграмма последовательности для импорта JSON-файла в плагине Unreal Engine представлена на рисунке 2.3.

Рисунок 2.3 – Диаграмма последовательности «Импорт JSON-файла в плагине Unreal Engine»



## Проектирование архитектуры системы

### Архитектура разрабатываемого ПО

Для приложения была выбрана компонентно-ориентированная архитектура [13].

Компонентно-ориентированная архитектура предполагает выполнение приложения в виде набора компонентов, которые можно использовать как в текущем, так и в будущих проектах. Компонент представляет из себя блок кода, работающий либо самостоятельно, либо совместно с другими компонентами приложения.

Проектируемый программный продукт использует двухкомпонентную архитектуру — модель, в которой система разделена на два независимых, но взаимодействующих компонента: десктопное приложение и плагин для Unreal Engine​.

Двухкомпонентная архитектура — это способ организации программного обеспечения, при котором функциональность системы распределяется между отдельными приложениями или модулями. Каждый компонент выполняет строго определённые задачи, взаимодействуя с другим компонентом через формализованные точки интеграции (например, файлы JSON или API-интерфейсы). Такая архитектура позволяет эффективно разделить зоны ответственности, повысить модульность, гибкость и масштабируемость системы.

Компоненты архитектуры:

* Десктопное приложение – предназначено для нарративного дизайнера. Обеспечивает создание, редактирование и визуализацию нарративных сущностей (персонажей, событий, локаций), а также экспорт данных в формате JSON. Может работать полностью автономно и офлайн.
* Плагин Unreal Engine – устанавливается в игровом проекте и позволяет импортировать нарративные данные, созданные дизайнером, в игровую среду. Плагин реализует связывание сущностей нарратива с объектами Unreal Engine (Actors, Levels, Blueprints) и предоставляет интерфейс для взаимодействия разработчиков с этими данными​.

Обмен данными между компонентами осуществляется посредством файлов JSON, что упрощает контроль версий, повышает отказоустойчивость и позволяет использовать компоненты независимо друг от друга.

Для реализации архитектуры применяется чистая архитектура, в которой система разделяется на логические слои:

* Уровень представления (UI) — отвечает за взаимодействие с пользователем (Qt GUI, вкладки персонажей, событий, экспорт).
* Уровень приложения — управляет сценариями использования и координирует слои.
* Уровень домена — реализует бизнес-логику: связи между сущностями, правила хронологии.
* Уровень инфраструктуры — реализует работу с файлами JSON, подключение к базе данных, интеграцию с Unreal Engine.

Также в приложении применяется паттерн MVVM (Model-View-ViewModel), обеспечивающий:

* изоляцию бизнес-логики от интерфейса;
* повышение читаемости и тестируемости кода;
* реактивное обновление интерфейса при изменении модели;
* model: структуры данных (персонажи, события, связи);
* view: Qt-интерфейс приложения;
* viewModel: логика обработки действий пользователя, формирование данных для отображения, валидация.

Такой подход позволяет легко развивать проект, внедрять новые функции, упрощает командную разработку и сопровождение.

### Архитектурная диаграмма

Архитектурная диаграмма — это визуальное представление структуры системы, демонстрирующее, как её компоненты связаны друг с другом, как они взаимодействуют, и как через систему проходят данные [14].

К такому виду диаграмм относится диаграмма C4.

Диаграмма C4 (от англ. С4 model, Context Container Component Code model, модель «контекст-контейнер-компонент-код») — метод графической записи для моделирования архитектуры программных систем. Модель C4 описывает архитектуру с разных точек зрения, объясняет декомпозицию системы на контейнеры и компоненты, а также связи между этими элементами.

Диаграмма C4 представлена на рисунках 2.4-2.6.

Рисунок 2.4 – Диаграмма контекста

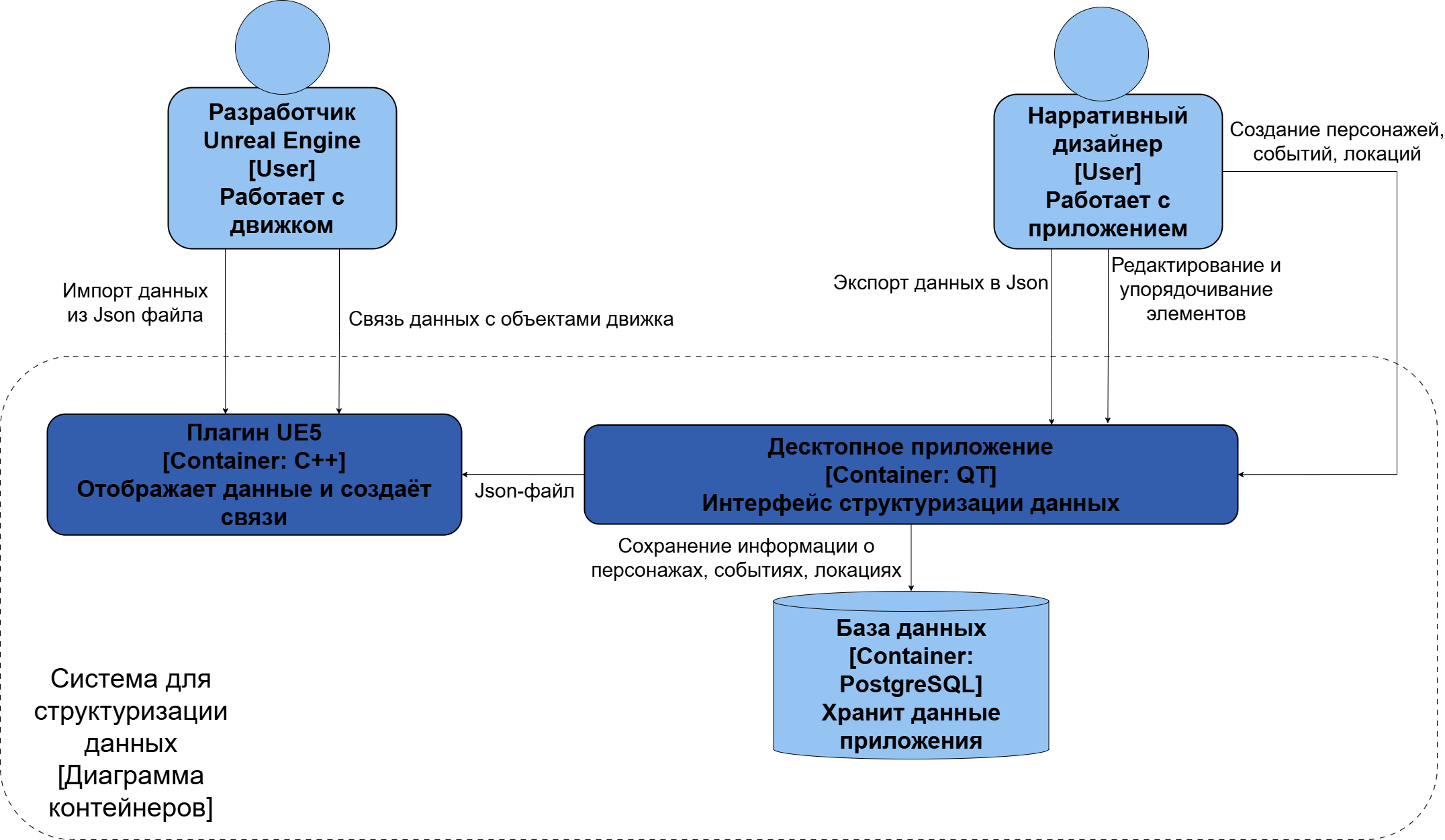
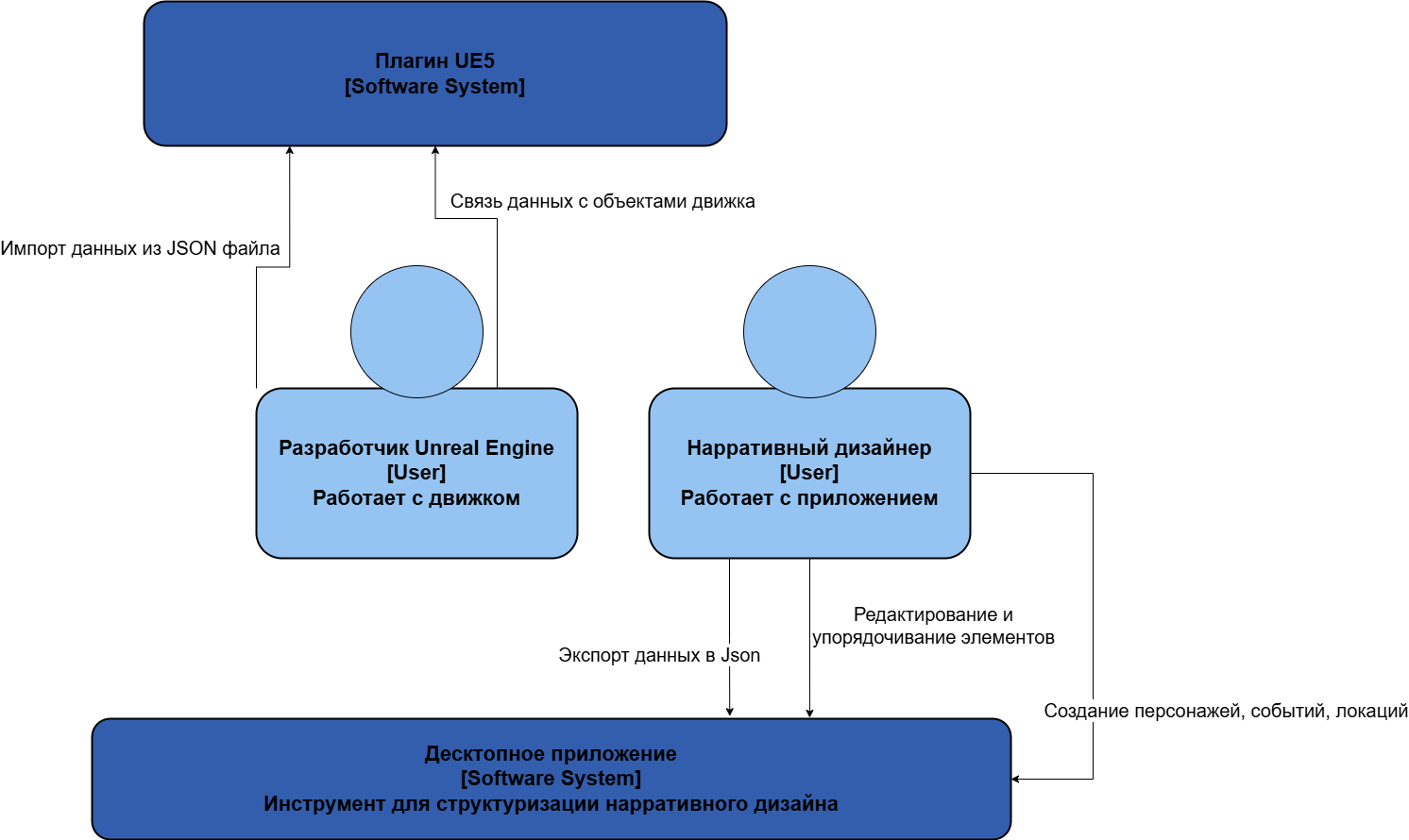


Рисунок 2.5 – Диаграмма контейнеров

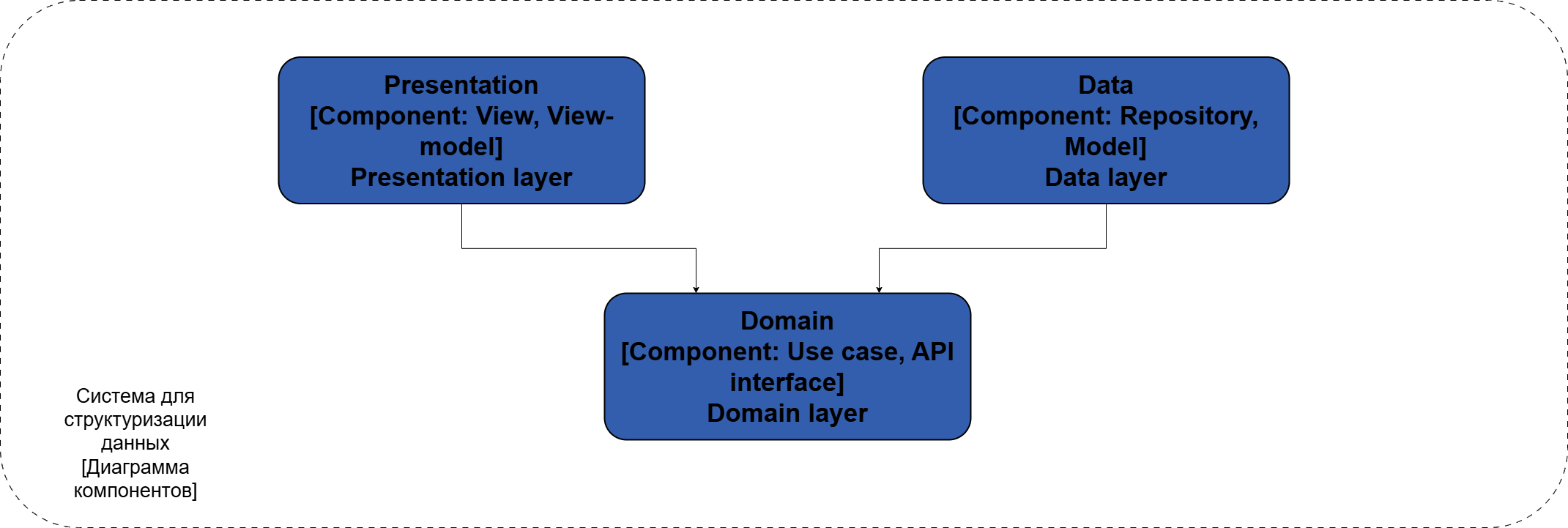


Рисунок 2.6 – Диаграмма компонентов

### Матрица доступа и роли

Для того, чтобы разграничить роли, а также права доступа к различны данным среди пользователей в разрабатываемой системе, необходимо построить матрицу разграничения прав доступа CRUD [14].

Матрица разграничения прав доступа CRUD (Create, Read, Update, Delete) представляет из себя таблицу, в которой каждой роли указывается соответствующие разрешения на выполнение различных действий с данными в базе данных. Матрица доступа представлена в таблице 2.1.

Таблица 2.1 – Матрица доступа

|  |  |  |
| --- | --- | --- |
| Модуль/Роль | Дизайнер | Разработчик |
| Проект | С, R, U, D | R |
| Персонажи | С, R, U, D | R |
| Локации | С, R, U, D | R |
| События | С, R, U, D | R |
| Связи персонажей | С, R, U, D | R |
| Экспорт JSON | C, R | R |
| Импорт в UE5 | - | C, R |
| Привязка к UE5 | - | C, R, U, D |

Где:

R – Чтение.

U – Модификация.

С – Создание.

D – Удаление.

Из таблицы 2.1 видно, что у дизайнера присутствуют все права на манипуляции с нарративными данными. Разработчик имеет права на работу с импортированными файлами внутри Unreal Engine.

# ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## Анализ выбранных средств разработки

### Технологии реализации

Существует несколько графических фреймворков, которые можно использовать для разработки программного обеспечения с графическим пользовательским интерфейсом.

* QT – это мощный кроссплатформенный фреймворк для разработки десктопных приложений [15].
* wxWidgets – ориентирован на создание нативных GUI благодаря поддержке нативных элементов ОС [16].
* GTKmm – обёртка для GTK+, широко используется в Linux [17].
* ImGui – фреймворк для создания отладочного интерфейса [18].
* SFML – фреймворк с акцентом на мультимедиа [19].

Сравнение фреймворков представлено в таблице 3.1.

Таблица 3.1 – Сравнение фреймворков

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерий | QT | wxWidgets | GTKmm | ImGui | SFML |
| Кроссплатформенность | Windows, macOS, Linux | Windows, macOS, Linux | macOS, Linux | Windows, macOS, Linux | Windows, macOS, Linux |
| Готовые виджеты | Есть | Ограничено | Есть | Ограничено | Через TGUI |
| Поддержка JSON | Есть | Ограниченно | Есть | Ручной парсинг | Нет |
| Простота проектирования дизайна | Есть Qt Designer | Проектирование в рамках доступных опций | Есть Glade | Только ручное | Только ручное |
| Интеграция с Unreal Engine | Нативная | Нет | Нет | Есть | Есть |

В результате сравнения был выбран QT из-за нативной поддержки Unreal Engine, удобного редактора Qt Designer и поддержки JSON.

Для хранения данных системы необходимо использовать базу данных. Далее представлена информация о различных системах управления базами данных:

* PostgreSQL – надёжная и производительная система управления базами данных с открытым исходным кодом [20].
* SQLite – лёгкая встраиваемая СУБД, работающая как обычный файл без сервера [21].
* MySQL — популярная реляционная СУБД с высокой производительностью и простотой настройки [22].
* MongoDB — документоориентированная NoSQL-база данных, удобная для хранения неструктурированных данных [23].

Сравнение этих систем приведено в таблице 3.2.

Таблица 3.2 – Сравнение СУБД

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Характеристика | SQLite | MySQL | MongoDB | PostgreSQL |
| Тип БД | Встраиваемая | Реляционная | NoSQL | Объектно-реляционное |
| Поддержка транзакций | Базовая | Да | Ограниченная | Да |
| Расширяемость | Нет | Ограниченная | Через плагины | Да |
| Работа с большими объемами данных | Неэффективна | Да | Да | Да |
| Поддержка SQL-стандарта | Частичная | Частичная | Нет | Полная |
| Поддержка сложных запросов | Ограничена | Да | Ограничена | Да |
| Распределённая репликация | Нет | Встроенная | Да | Gin, GiST, BRIN |

В результате сравнения была выбрана СУБД PostgreSQL, из-за хорошей расширяемости и поддержки SQL стандарта.

Для создания продукта возможно использовать следующие среды разработки:

* Visual Studio – мощная IDE от Microsoft с глубокой интеграцией с C++, C#, .NET и Unreal Engine [24].
* JetBrains Rider – кроссплатформенная IDE для .NET и Unreal Engine с поддержкой C++ через плагин ReSharper C++ [25].
* CLion – IDE от JetBrains для кроссплатформенной разработки на C и C++, с отличной поддержкой CMake и GDB [26].
* Qt Creator – IDE оптимизированная для создания интерфейсов и приложений на C++ с использованием фреймворка Qt [27].

Сравнение сред разработки приведено в таблице 3.3.

Таблица 3.3 – Сравнение IDE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | Visual Studio | Rider | CLion | Qt Creator |
| Поддержка C++ | Да | Через ReSharper C++ | Да | Да |
| Интеграция с Unreal Engine | Да | Нативная | Вручную | Через CMake |
| Поддержка Qt | Плагин | Через CMake | Через CMake | Встроенная |
| Поддержка GUI-дизайна | Ограничена | Нет | Нет | Qt Designer |
| Поддержка JSON | Плагины | Плагины | Плагины | Встроенно |
| Кроссплатформенность IDE | Windows | Windows, MacOS, Linux | Windows, MacOS, Linux | Windows, MacOS, Linux |
| Производительность | Низкая | Низкая | Средняя | Высокая |
| Стоимость | Бесплатно | Платно | Платно | Бесплатно |

В результате сравнения сред разработки был выбран Qt Creator из-за нативной поддержки Qt и удобной работы с GUI.

## Описание пользовательского взаимодействия

### Путь пользователя

Карта пользовательского пути (User flow) – схема, описывающая перемещения пользователя по экранам для выполнения своей задачи. Так как, проектируемая система состоит из двух модулей, были созданы две карты пользовательского пути. Путь пользователя в десктопном приложении представлен на рисунке 3.1.

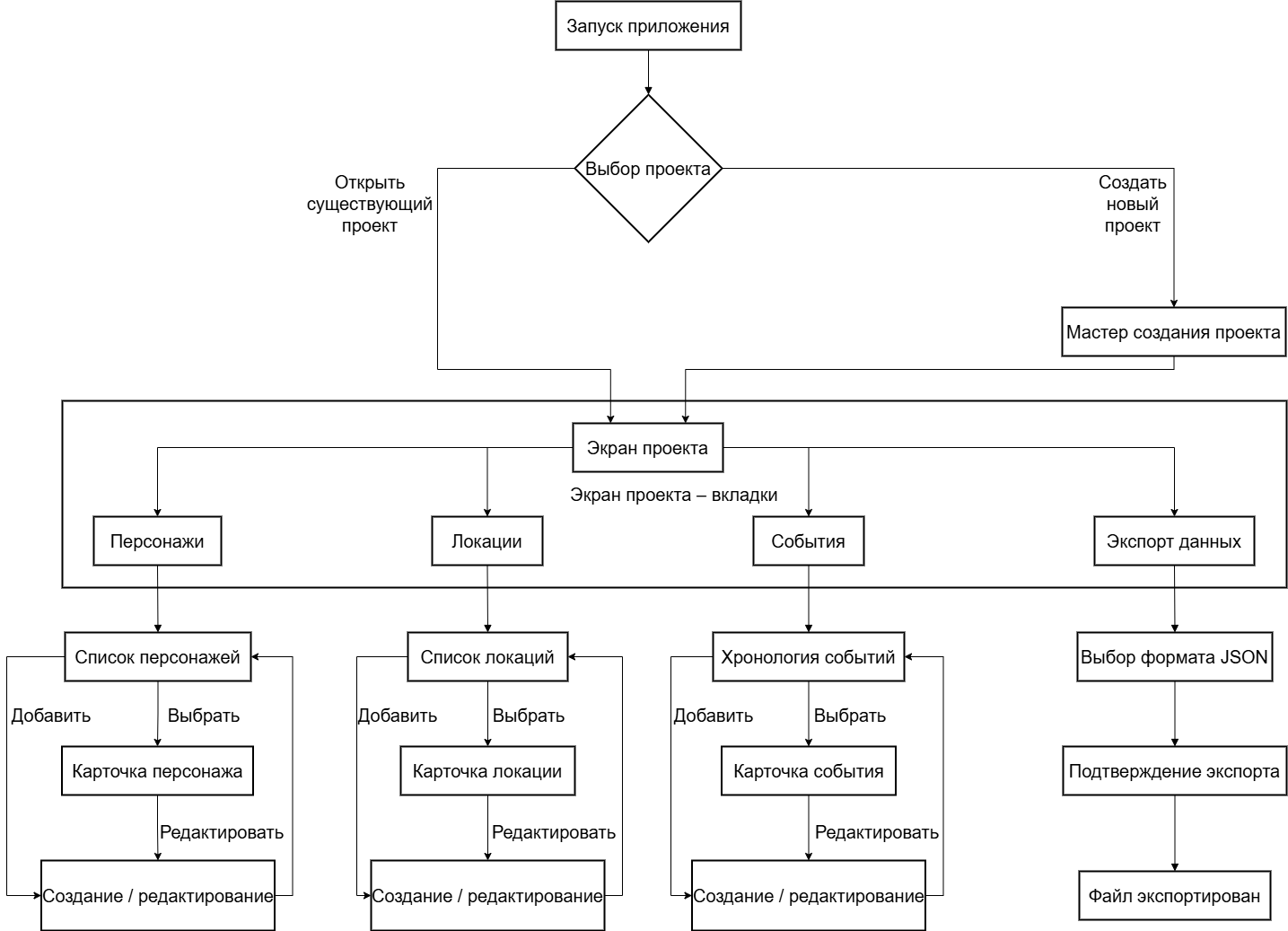


Рисунок 3.1 – Карта пользовательского пути пользователя в приложении

На рисунке 3.2 представлена карта пользовательского пути (User flow) – схема, описывающая перемещения пользователя по экранам плагина Unreal Engine.

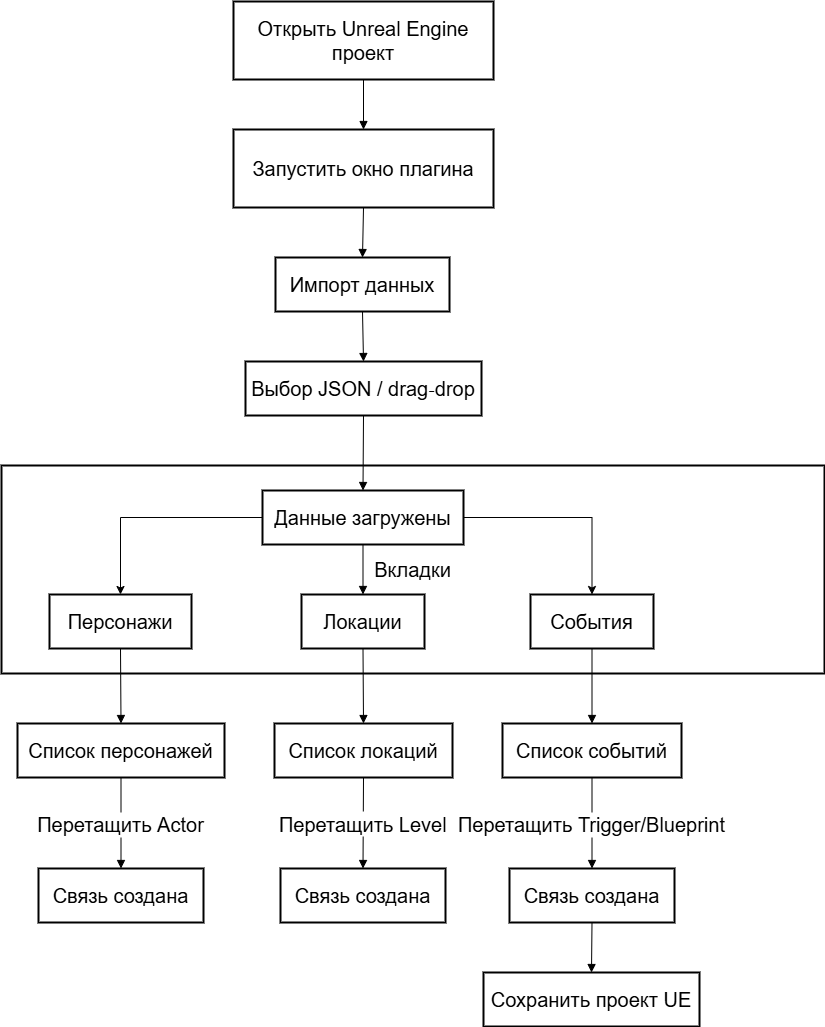


Рисунок 3.2 – Карта пользовательского пути пользователя в плагине

### Описание пользовательского интерфейса

Макет экранов пользовательского интерфейса был создан при помощи средства проектирования интерфейсов Figma [28].

Макет экранов представлен на рисунке 3.3.



Рисунок 3.3 – Макет интерфейсов

Приложение состоит из шести экранов:

1. Экран выбора проекта. Является начальным экраном приложения для нарративного дизайнера. На экране отображается список уже существующих проектов с возможностью выбора или кнопкой создания нового проекта. При выборе проекта открывается экран проекта.
2. Экран проекта. Это основной интерфейс, с которым взаимодействует нарративный дизайнер. Экран проекта содержит панель с четырьмя вкладками для переключения между экранами: «Персонажи», «Локации», «События» и «Экспорт данных».
3. Экран «Персонажи». Отображает карточки персонажей, организованные в виде сетки или списка. Каждая карточка включает изображение персонажа, его имя и краткое описание. При нажатии на карточку открывается подробная страница персонажа с расширенной информацией и возможностью редактирования данных.
4. Экран «Локации». Аналогичен экрану персонажей, отображает список или сетку локаций с изображениями, названиями и краткими описаниями. При нажатии на локацию пользователь переходит на страницу с детальной информацией и возможностью её редактирования.
5. Экран «События». Включает отображение хронологического порядка событий и позволяет просматривать их последовательность и связи с персонажами и локациями. События представлены в виде временной шкалы или списка. Каждое событие имеет название и краткое описание. При клике открывается подробная страница с возможностью редактирования и управления связями с другими сущностями.
6. Экран «Экспорт данных». Предоставляет возможность экспорта структурированных данных о персонажах, локациях и событиях в форматах JSON для последующего импорта в Unreal Engine или другие системы.

Интерфейс плагина включает четыре вкладки, аналогичные десктопному приложению, однако имеет некоторые отличия:

1. Вкладка «Персонажи» – отображает список персонажей без изображений и возможности редактирования текста. Позволяет перетаскивать и привязывать объекты-актеры из движка к персонажам.
2. Вкладка «Локации» – содержит список локаций, также без изображений и текста, с возможностью перетаскивать и связывать уровни и объекты движка с локациями.
3. Вкладка «События» – показывает хронологический порядок событий без редактируемых полей, но с возможностью связывать события с триггерами и другими объектами движка путем перетаскивания.
4. Вкладка «Импорт» – заменяет вкладку экспорта, предназначен для загрузки данных в формате JSON из приложения дизайнера, что позволяет интегрировать нарративные данные с объектами в среде разработки Unreal Engine.

## Информационное обеспечение

### Логическая модель базы данных

Логическая модель базы данных — это абстрактное представление структуры данных, которое используется для планирования и проектирования баз данных. Она описывает, как данные будут организованы и как они будут взаимодействовать друг с другом, не привязываясь к конкретной системе управления базами данных (СУБД).

Спроектированная логическая модель базы данных программного продукта представлена на рисунке 3.4.

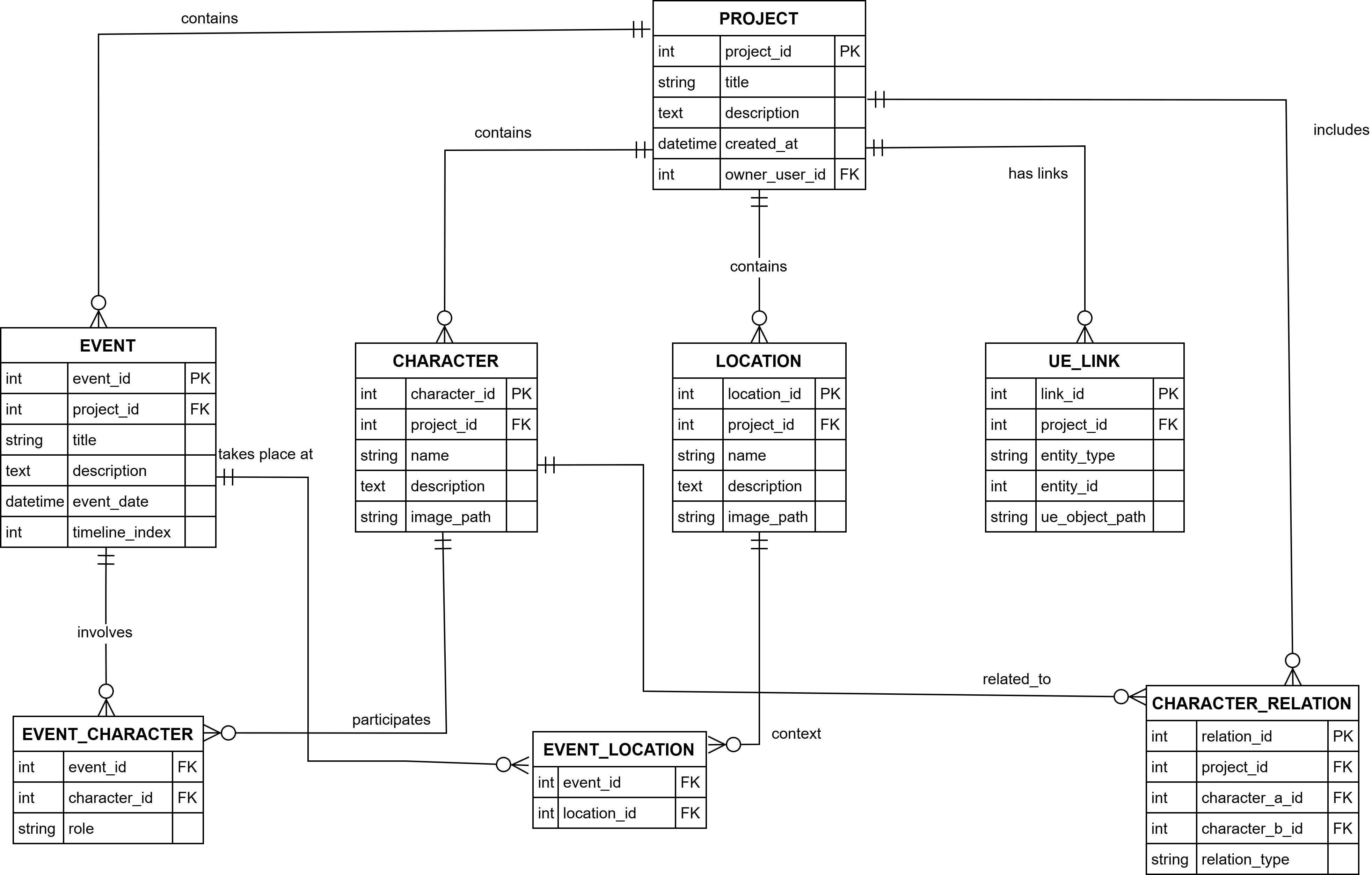


Рисунок 3.4 – Логическая модель базы данных

### Словарь данных

Словарь данных базы данных (БД) — это централизованный сбор информации о различных компонентах и структурах системы баз данных. Он включает сведения о таблицах, индексах, типах данных, ограничениях, связях и других объектах базы данных.

Исходя из логической модели базы данных построенной ранее, был составлен словарь данных, представленный в таблице 3.4.

Таблица 3.4 – Словарь данных

|  |  |
| --- | --- |
| Наименование | Описание |
| PROJECT | Хранит информацию о нарративных проектах: уникальный идентификатор проекта, заголовок, описание, дата создания, ID владельца проекта (пользователя). |
| CHARACTER | Хранит информацию о персонажах проекта: уникальный идентификатор персонажа, ID проекта, имя персонажа, описание, путь к изображению. |
| LOCATION | Хранит информацию о локациях: уникальный идентификатор локации, ID проекта, название локации, описание, путь к изображению. |
| EVENT | Хранит информацию о событиях проекта: уникальный идентификатор события, ID проекта, заголовок, описание, дата события, индекс в хронологии.  тренера, вид обучения (платный, бюджетный) |
| EVENT\_CHARACTER | Хранит связи между событиями и персонажами: ID события, ID персонажа, роль персонажа в событии. |
| EVENT\_LOCATION | Хранит связи между событиями и локациями: ID события, ID локации, где происходит событие. |
| CHARACTER\_RELATION | Хранит отношения между персонажами: уникальный идентификатор отношения, ID проекта, ID персонажа A, ID персонажа B, тип отношения (друг, враг, родитель и т.д.). |
| UE\_LINK | Хранит связи между сущностями нарратива и объектами Unreal Engine: уникальный идентификатор связи, ID проекта, тип сущности (персонаж, событие, локация), ID сущности, путь к объекту UE. |

Код реализующий БД в создаваемой системе представлен в Приложении Г.

### Диаграмма классов

Диаграмма классов — это один из основных видов UML-диаграмм (Unified Modeling Language), предназначенный для моделирования **структуры системы** с точки зрения **классов, их атрибутов, методов и связей между ними [29]**.

Диаграмма классов представлена на рисунке 3.5.

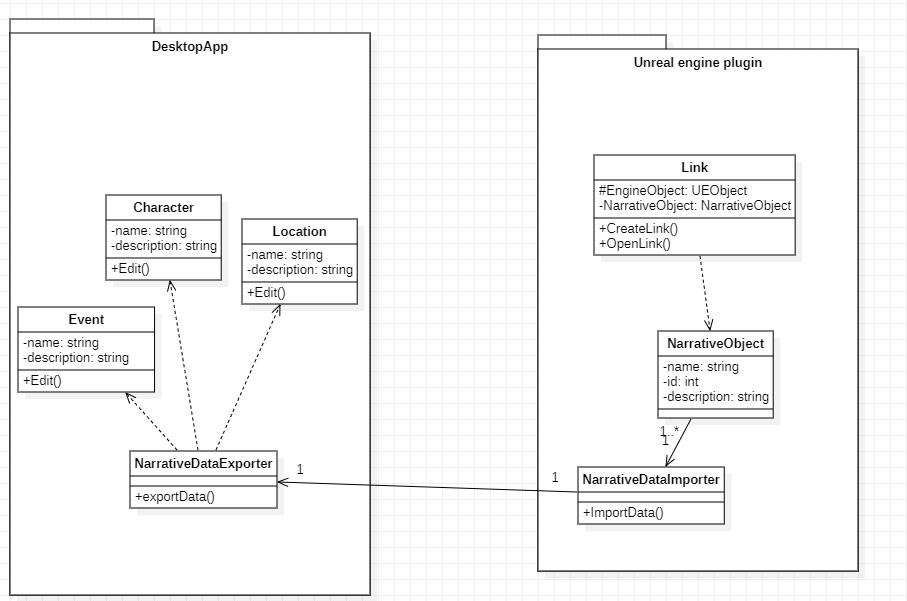


Рисунок 3.5 – Диаграмма классов

## Программа и методика испытаний

Программа и методика испытаний (ПМИ) — это технический документ, который описывает план и последовательность проведения испытаний оборудования, систем или материалов, а также методы, используемые для оценки их характеристик, надёжности и соответствия установленным требованиям [30].

### Объект испытаний

Наименование системы: Система для структуризации нарративного дизайна с интеграцией со средой разработки.

Область применения: Оптимизация взаимодействия нарративного дизайнера с разработчиков в процессе создания игр.

Условное обозначение системы: NSP.

### Цель испытаний

Документы, на основании которых проводятся испытания:

* Техническое задание на разработку NSP.
* Настоящая программа и методика испытаний.

Место проведения испытаний: Площадка Заказчика.

Продолжительность испытаний устанавливается Приказом Заказчика о составе приёмочной комиссии и проведении приёмочных испытаний.

### Объём испытаний

Перечень этапов испытаний и проверок:

* Функциональные тесты:

1. Создание нового проекта.
2. Создание персонажа.
3. Редактирование события.
4. Экспорт данных в JSON.
5. Импорт данных в плагин Unreal engine.
6. Связь объектов Unreal engine с элементами нарратива.

* Тесты пользовательского интерфейса (UI):
  1. Тестирование адаптивности интерфейса на различных разрешениях экрана.
* Нагрузочное тестирование:
  1. Проверка производительности базы данных при большом объёме данных.

### Методика проведения испытаний

Методика проведения испытаний – это документ, который описывает, как именно должны проводиться испытания программного обеспечения.

Методика проведения испытаний представлена на таблице 3.5.

Таблица 3.5 – Методика проведения испытаний

|  |  |  |
| --- | --- | --- |
| № | **Действие** | **Ожидаемый результат** |
| 1. | **Сценарий «Создание нового проекта»** | |
|  | Запустить приложение и нажать «Создать новый проект» | Открыто окно создания проекта |
|  | Ввести имя проекта и указать место сохранения | Имя и путь заданы |
|  | Нажать «Создать» | Проект создан, пользователь перенаправлен в главное окно |
| 2. | **Сценарий «Создание персонажа»** | |
|  | Открыть вкладку «Персонажи» | Вкладка «Персонажи» открыта |
|  | Заполнить поля (имя, описание) | Данные внесены |
|  | Нажать «Сохранить» | Персонаж успешно создан, отображается в списке персонажей |
| 3. | **Сценарий «Редактирование события»** | |
|  | Перейти во вкладку «События», выбрать событие | Открыта карточка события |
|  | Внести изменения (изменить участников, описание) | Изменения отображаются в форме |
|  | Нажать «Сохранить» | Событие обновлено |
| 4. | **Сценарий «Экспорт данных в JSON»** | |
|  | Перейти в «Файл» → «Экспорт» | Открыто окно экспорта |
|  | Выбрать расположение JSON | Данные внесены |
|  | Нажать «Экспортировать» | Файл успешно создан |
| 5. | **Сценарий «Импорт данных в Unreal Engine» (плагин)** | |
|  | Открыть Unreal Engine, перейти к плагину | Интерфейс плагина доступен |
|  | Перетащить JSON-файл в поле импорта | Нарративные сущности добавлены в систему |
| 6. | **Сценарий «Связь объектов Unreal Engine с элементами нарратива»** | |
|  | Перетащить соответствующий Actor на элемент в плагине | Связь установлена |
|  | Нажать на элемент в плагине | Открыто расположение Actor в проводнике Unreal Engine |

### Требования по испытаниям программных средств

Требования по испытаниям программных средств:

* Корректное выполнение всех функций, указанных в ТЗ проектируемого программного продукта.
* Отсутствие ошибок интерфейса.

### Перечень работ, проводимых после завершения испытаний

По результатам испытаний делается заключение о соответствии приложения NSP требованиям технического задания и возможности оформления акта сдачи приложения в опытную эксплуатацию. При этом производится (при необходимости) доработка программных средств и документации.

### Условия и порядок проведения испытаний

Испытания приложение NSP должны проводиться на целевом оборудовании Заказчика. Оборудование должно быть предоставлено в той конфигурации, которая запланирована для начального развёртывания системы, и указана в Техническом задании.

Во время испытаний проводится полное функциональное тестирование, согласно требованиям, указанным в Техническом задании.

При проведении приемочных испытаний доступ системе предоставляется ограниченному кругу пользователей.

### Материально-техническое обеспечение испытаний

Приёмочные испытания проводятся на программно-аппаратном комплексе Заказчика в следующей минимальной конфигурации:

Компьютер с следующей конфигурацией:

* Процессор: 6-ти ядерный процессор с тактовой частотой не менее 2.4 ГГц.
* Оперативная память (RAM): 8 ГБ.
* Встроенная память: наличие 200 МБ свободного пространства.
* Операционная система: Windows 10.

### Отчётность

Результаты испытаний приложения NSP, предусмотренные настоящей программой, фиксируются в протоколах, содержащих следующие разделы:

* Назначение испытаний и номер раздела требований ТЗ, по которому проводят испытание.
* Состав технических и программных средств, используемых при испытаниях.
* Указание методик, в соответствии с которыми проводились испытания, обработка и оценка результатов.
* Условия проведения испытаний и характеристики исходных данных;
* Средства хранения и условия доступа к тестирующей программе;
* Обобщённые результаты испытаний.
* Выводы о результатах испытаний и соответствии созданной системы определённому разделу требований ТЗ.

В протоколах могут быть занесены замечания персонала по удобству эксплуатации системы.

Этап проведения предварительных испытаний завершается оформлением «Акта предварительных и приемочных испытаний системы NSP».

## Расчёт вычислительной и ёмкостной сложности

Для расчёта выделяемых ресурсов необходимо рассчитать временную и ёмкостную сложность алгоритмов. В расчёте будет использована функция экспорта данных из базы данных в JSON формат, так как эта функция занимает наибольшее количество ресурсов рабочей станции. Исходный код функции приведён на листинге Д.1 Приложении Д.

Обозначения:

С – количество строк в таблице character.

L – количество строк в таблице location.

E – количество строк в таблице event.

EC – количество строк в таблице event\_character.

EL – количество строк в таблице event\_location.

ER – количество строк в таблице event\_relation.

U – количество строк в таблице UE\_link.

Расчёт временной сложности в худшем случае приведён в формуле 3.1:

(3.1)

Расчёт ёмкостной сложности приведён в формуле 3.2:

(3.2)

Обозначим сумму всех строк всех таблиц базы данных за n.

Временная сложность функции равна O(n).

Ёмкостная сложность функции равна O(n).

# ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

## Организация и планирование работ по теме

### Организация работ

В выполнении работ участвуют три человека (руководитель, консультант по экономической части, разработчик проекта):

* Руководитель ВКР (Лобанов Александр Анатольевич, кандидат технических наук, доцент) - контролирует этапы работы, вносит необходимые коррективы и оценивает выполненную работу.
* Консультант по экономической части (Петросян Лусинэ Эдуардовна, кандидат экономических наук, доцент) - консультирует при выполнении экономической части выпускной квалификационной работы.
* Разработчик проекта (Чирва Андрей Юрьевич, ИКБО-07-21) - планирует, проектирует и разрабатывает программное решение в соответствии с поставленной задачей.

Состав этапов работы с участвующими в разработке исполнителями и с указанием продолжительности работ, трудоёмкости (чел/дни) приведён в таблице 4.1.

Таблица 4.1 – Этапы разработки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название этапа** | **Исполнитель** | **Трудоёмкость, чел/дни** | **Продолжительность работ, дни** |
| 1 | **Этап планирования:** |  |  | **14** |
| 1.1 | Обследование объекта и обоснование необходимости создания проекта | Руководитель | 2 |
| Разработчик | 2 |
| Консультант | 1 |
| 1.2 | Анализ бизнес-процессов | Руководитель | 1 |
| Разработчик | 2 |
| 1.3 | Определение функциональных требований | Разработчик | 3 |
| 1.4 | Определение нефункциональных требований | Разработчик | 3 |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1.5 | Подготовка документации по требования | Руководитель | 2 |  |
| Разработчик | 2 |
|  |  |  |  |
| 1.6 | Оформление отчёта о выполненной работе | Разработчик | 2 |
| 2 | **Этап проектирования:** |  |  | **13** |
| 2.1 | Проектирование архитектуры | Руководитель | 3 |
| Разработчик | 6 |
| 2.2 | Проектирование базы данных | Разработчик | 3 |
| 2.3 | Разработка макетов и прототипов интерфейсов | Разработчик | 2 |
| 2.4 | Оформление отчёта о выполненной работе | Разработчик | 2 |
| 3 | **Этап разработки:** |  |  | **26** |
| 3.1 | Разработка основного функционала приложения | Разработчик | 10 |
| 3.2 | Разработка основного функционала плагина | Разработчик | 6 |
| 3.3 | Тестирование и исправление ошибок |  |  |
| 3.3.1 | Тестирование приложения | Разработчик | 3 |
| 3.3.2 | Тестирование плагина | Разработчик | 2 |
| 3.3.3 | Исправление выявленных ошибок | Руководитель | 2 |
| Разработчик | 3 |
| Консультант | 4 |
| 3.4 | Завершение разработки системы | Руководитель | 1 |
| Разработчик | 1 |
| Консультант | 1 |

Этапы проектирования и разработки приложения приведён в таблице 4.2.

Планирование сроков выполнялось с учётом:

* пятидневной рабочей недели;
* всех праздничных дней руководителя ВКР, консультанта по экономической части и студента;
* выходных и праздничных дней.

Таблица 4.2 – Календарный план выполнения проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Этап** | **Дата начала** | **Дата окончания** | **Количество рабочих дней** | **Исполнитель** |
| **1 Анализ требований и планирование** | | | | |
| 1.1 Обследование объекта и обоснование необходимости создания проекта | 10.02.2025 | 11.02.2025 | 2 | Разработчик проекта  Руководитель ПЭО  Руководитель ВКР |

Продолжение таблицы 4.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1.2 Анализ бизнес-процессов | 12.02.2025 | 13.02.2025 | 2 | Разработчик проекта  Руководитель ВКР |
| 1.3 Определение функциональных требований | 14.02.2025 | 18.02.2025 | 3 | Разработчик проекта |
| 1.4 Определение нефункциональных требований | 19.02.2025 | 21.02.2025 | 3 | Разработчик проекта |
| 1.5 Подготовка документации по требованиям | 24.02.2025 | 25.02.2025 | 2 | Разработчик проекта  Руководитель ВКР |
| 1.6 Оформление отчёта о выполненной работе | 26.02.2025 | 27.03.2025 | 2 | Разработчик проекта |
| **2 Проектирование и архитектура** | | | | |
| 2.1 Проектирование архитектуры | 28.02.2025 | 07.03.2025 | 6 | Разработчик проекта  Руководитель ВКР |
| 2.2 Проектирование базы данных | 10.03.2025 | 12.03.2025 | 3 | Разработчик проекта |
| 2.3 Разработка макетов и прототипов интерфейсов | 13.03.2025 | 14.03.2025 | 2 | Разработчик проекта |
| 2.4 Оформление отчёта о выполненной работе | 17.03.2025 | 18.03.2025 | 2 | Разработчик проекта |
| **3 Разработка системы** | | | | |
| 3.1 Разработка основного функционала приложения | 19.03.2025 | 01.04.2025 | 10 | Разработчик проекта |
| 3.2 Разработка основного функционала плагина | 02.03.2025 | 09.04.2025 | 6 | Разработчик проекта |
| 3.3 Тестирование и исправление ошибок |  |  |  |  |
| 3.3.1 Тестирование приложения | 10.04.2025 | 14.04.2025 | 3 | Разработчик проекта |
| 3.3.2 Тестирование плагина | 15.04.2025 | 16.04.2025 | 2 | Разработчик проекта |
| 3.3.3 Исправление выявленных ошибок | 17.04.2025 | 21.04.2025 | 3 | Разработчик проекта  Руководитель ПЭО  Руководитель ВКР |
| 3.4 Завершение разработки системы | 22.04.2025 | 22.04.2025 | 1 | Разработчик проекта  Руководитель ПЭО  Руководитель ВКР |

### График проведения работ

Диаграмма Ганта широко используется для визуализации хода выполнения задач, планирования ресурсов, графика рабочего времени и других данных, которые представляются набором временных интервалов. Она необходима для соблюдения сроков выполнения работ, и, соответственно, анализа хода проектирования. Необходимо отметить, что задачи могут выполняться как последовательно, так и параллельно [31].

Основываясь на данных, представленных в календарном плане выполнения проекта, была составлена диаграмма Ганта (Приложение Е).

## Расчёт стоимости проведения работ по теме

### Сырьё и материалы

В данной статье рассчитывается стоимость материалов, покупных изделий, комплектующих изделий и других материальных ценностей, расходуемых непосредственно в процессе выполнения работ. Так как проект не предполагает создания аппаратного продукта, то будет произведена только закупка канцелярских товаров. В процессе подсчёта транспортно-заготовительные расходы будут приравнены к 20% от общей стоимости затрат (4.1).

(4.1)

где:

ТЗР - транспортно-заготовительные расходы;

- итоговая стоимость материалов.

Результат расчёта стоимости материалов представлены в таблице 4.3.

Таблица 4.3 – Стоимость материалов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ п/п** | **Наименование материалов** | **Единицы измерения** | **Количество** | **Цена за единицу, руб.** | **Стоимость, руб.** |
| 1 | Бумага формата А4 | пачка | 1 | 574 | 574 |
| 2 | Картридж для принтера | шт | 2 | 4901 | 9802 |
| 3 | Канцелярский набор | шт | 1 | 731 | 731 |
| 4 | USB-накопитель объёмом в 64 Гб | шт | 1 | 1184 | 1184 |
| **Итого материалов** | | | | | **12291** |
| **Транспортно-заготовительные расходы** | | | | | **2458,2** |
| **Итого** | | | | | **14749,2** |

Затраты на комплектующие и изготовление программно-аппаратного комплекса целесообразно учитывать в отдельной таблице (Таблица 4.4).

Таблица 4.4 – Стоимость комплектующих

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ п/п** | **Наименование материалов** | **Единицы измерения** | **Количество** | **Цена за единицу, руб.** | **Стоимость, руб.** |
| 1 | Монитор | шт | 1 | 8748 | 8748 |
| 2 | Клавиатура | шт | 1 | 1903 | 1903 |
| 3 | Компьютерная мышь | шт | 1 | 638 | 638 |
| **Итого материалов** | | | | | **11289,0** |
| **Транспортно-заготовительные расходы** | | | | | **2257,8** |
| **Итого** | | | | | **13546,8** |

Таким образом общая сумма затрат по статье «Сырьё и материалы» составила 13546,8 руб.

### Статья «Основная заработная плата»

К данной статье относится оплата труда научных работников, инженерно-технических работников и рабочих, непосредственно занятых выполнение конкретной работы, а также заработная плата работников внештатного состава, привлекаемого к её выполнению (4.2).

(4.2)

где:

* ТС - тарифная ставка (оплата за день);
* ОК - месячных оклад участников проекта;
* 12 - число месяцев в году;
* НРВ - годовой фонд рабочего времени при сорокачасовой рабочей неделе (равен примерно 233 дням).

Расчёт основной заработной платы для каждого участника был произведён в соответствии с календарным графиком выполнения работ и отображён в таблице 4.5.

Таблица 4.5 – Расчёт основной заработной платы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Наименование этапа** | **Исполнитель,должность** | **Мес. оклад, руб.** | **Трудоёмкость, чел/день** | **Тарифная ставка, руб/день** | **Оплата за этап, руб.** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Планирование | Руководитель | 110 000 | 5 | 5665,2 | 28326 |
| Разработчик | 65 000 | 14 | 3347,6 | 46866,4 |
| Консультант | 90 000 | 1 | 4635,2 | 4635,2 |
| 2 | Проектирование | Руководитель | 120 000 | 3 | 6180,3 | 18540,9 |
| Разработчик | 65 000 | 13 | 3347,6 | 43518,8 |
| Консультант | 100 000 | 0 | 5150,2 | 0 |
| 3 | Разработка | Руководитель | 120 000 | 3 | 6180,3 | 18540,9 |
| Разработчик | 75 000 | 25 | 3862,7 | 96567,5 |
| Консультант | 100 000 | 5 | 5150,2 | 25751 |
| **Итого** | | | | | | 282746,7 |

### Статья «Дополнительная заработная плата»

К этой статье относятся выплаты, предусмотренные законодательством о труде за необработанное по уважительным причинам время; оплата очередных и дополнительных отпусков; времени, связанного с выполнением государственных и общественных обязанностей; выплата вознаграждения за выслугу лет и т.д. (4.3).

(4.3)

где:

* ДЗП - дополнительная заработная плата;
* ОЗП - основная заработная плата.

Также необходимо определить общую сумму оплаты труда (4.4).

(4.4)

где:

* ФОТ - фонд оплаты труда;
* ОЗП - основная заработная плата;
* ДЗП - дополнительная заработная плата.

### Статья «Страховые взносы»

В соответствии со статьёй 425 НК РФ в 2025 году для лиц, которые производят выплаты и вознаграждения физическим лицам, установлен Единый тариф страховых взносов 30% по следующим направлениям:

* Обязательное пенсионное страхование (ОПС).
* Обязательное социальное страхование на случай временной нетрудоспособности и в связи с материнством (ОСС).
* Обязательное медицинское страхование (ОМС).

Также в расчёте необходимо учесть ставку взносов на травматизм, которая может составлять от 0,2% до 8,5%. Для РТУ МИРЭА данная ставка составляет 0,2 (4.5).

(4.5)

где:

* СВ - страховые взносы;
* 30,2% - Единый тариф страховых вносов (30%) + ставка взносов за травматизм для РТУ МИРЭА (0,2%);
* ФОТ - фонд оплаты труда.

### Статья «Амортизационные отчисления»

Амортизация - отчисления части стоимости основных фондов для возмещения их износа. Амортизация включается в издержки производства [32]. Размер амортизационных отчислений определяется на основе установленных норм и балансовой стоимости основных фондов, на которые отчисляется амортизация (4.6).

(4.6)

где:

* А - месячная сумма амортизационных отчислений;
* С - первоначальная стоимость объекта;
* Т - срок полезного использования в месячных.

В таблице 4.6 представлен пример расчётов амортизационных отчислений по использованному во время разработки проекта оборудованию.

Таблица 4.6 – Расчёт амортизационных отчислений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Наименование оборудования** | **Первоначальная стоимость объекта, руб.** | **Срок полезного использования, мес.** | **Месячная сумма амортизационных отчислений, руб.** | **Период эксплуатации, мес.** | **Сумма, руб.** |
| Рабочая станция дизайнера | 33800 | 48 | 704,2 | 3 | 2112,6 |
| Рабочая станция разработчика | 85500 | 48 | 1781,3 | 3 | 5343,9 |
| **Итого** | | | | | 7456,5 |

### Прочие расходы

К прочим расходам прежде всего отнесены накладные расходы. Это расходы на содержание и ремонт зданий, сооружений, оборудований, инвентаря; затраты, сопутствующие основному производству, но не связанные с ним напрямую, не входящие в стоимость труда и материалов. Формула расчёта накладных расходов (4.7).

(4.7)

где:

* НР - накладные расходы.

### Полная себестоимость работ

Вышеперечисленные статьи затрат и результаты расчётов по ним приведены в таблице 4.7.

Таблица 4.7 – Себестоимость работ

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Номенклатура статей расходов** | **Затраты, руб.** | **Доля затрат, %** |
| 1 | Сырьё и материалы | 13546,8 | 1.6 |
| 2 | Основная заработная плата | 282746,7 | 35.2 |
| 3 | Дополнительная заработная плата | 56549,3 | 7.1 |
| 4 | Страховые взносы | 102467,4 | 12.8 |
| 5 | Амортизация | 7456,5 | 1 |
| 6 | Прочие расходы | 339296 | 42.3 |
| **Итого** | | **802062,7** | **100** |

Для визуализации долевого состава статей была составлена круговая диаграмма (Рисунок 4.1).

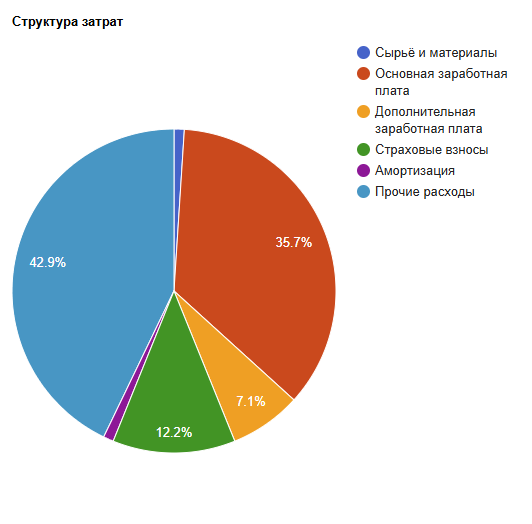


Рисунок 4.1 – Круговая диаграмма статей затрат

# ЗАКЛЮЧЕНИЕ

В процессе выполнения ВКР (выпускной квалификационной работы) был проведён анализ конкурентных решений, были выделены их сильные и слабые стороны, выбраны средства разработки, сформулированы и поставлены задачи разработки.

Согласно выбранной архитектуре, была спроектирована база данных, разработано приложение и плагин. Была составлена программа и методика испытаний.

Была рассчитана стоимость проведения работ. Построена диаграмма статей расходов.

В результате была спроектирована и разработана система для структуризации нарративного дизайна с интеграцией со средой разработки. Поставленные, перед выполнением ВКР, цели были достигнуты в полном объёме.

# CONCLUSION

In the process of completing the final qualification work, an analysis of competitive solutions was carried out, their strengths and weaknesses were highlighted, development tools were selected, development tasks were formulated and set.

According to the chosen architecture, a database was designed, an application and a plug-in were developed. The test methodology was drawn up.

The cost of carrying out the work was calculated. A diagram of cost was constructed.

As a result, a system for structuring narrative design with integration with the development environment was designed and developed. The goals set before the fulfilment of final qualification work were achieved in full.

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления": введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст: дата введения 2018-01-07. – URL: http://www.consultant.ru/document/cons\_doc\_LAW\_292293/ (дата обращения: 17.02.2025). – Текст: электронный.
2. ГОСТ Р 7.0.100-2018. СИБИД. Библиографическое описание. Общие требования и правила составления. Утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 3 декабря 2018 г. №1050-ст. – URL: https://docs.cntd.ru/document/1200161674 (дата обращения: 20.02.2025) – Текст: электронный.
3. СМКО МИРЭА 7.5.1/03.П.30-19. Порядок проведения государственной итоговой аттестации по образовательным программам высшего образования – программам бакалавриата, программам специалитета, и программам магистратуры. Введено в действие приказом РТУ МИРЭА от 17.06.2019 №965.– URL: https://www.mirea.ru/upload/iblock/3ba/ng7aomrcbiz0s6vpcdrrie83x9tj974d/Poryadok-provedeniya-GIA\_v\_1.3\_nov.pdf (дата обращения: 20.02.2025). – Текст: электронный.
4. СМКО МИРЭА 7.5.1/03.П.67-19. Положение о выпускной квалификационной работе студентов, обучающихся по образовательным программам подготовки бакалавров. Введено в действие приказом РТУ МИРЭА от 05.10.2018 №1320. – URL: https://www.mirea.ru/upload/iblock/f3f/nh3mswxf4yukarprfpy2inlaf8z533uh/Polozhenie-o-VKR-\_bak\_245.pdf (дата обращения: 24.02.2025). – Текст электронный.
5. ФГОС ВО 3++. Федеральный государственный образовательный стандарт высшего образования – бакалавриат по направлению подготовки 09.03.04 Программная инженерия. Утвержден приказом Министерства образования и науки Российской Федерации от 19 сентября 2017 г. №920. [Электронный ресурс], 2025 – URL: https://fgosvo.ru/uploadfiles/FGOS%20VO%203++/Bak/090304\_B\_3\_15062021.pdf (дата обращения: 24.02.2025). – Текст: электронный.
6. Нарративный Дизайн в медиакоммуникации [Электронный ресурс], 2025 – URL: https://lib.sevsu.ru/xmlui/bitstream/handle/123456789/11728/р\_250037.pdf (дата обращения 25.02.2025) – Текст электронный.
7. Использование диаграммы вариантов использования UML при проектировании ПО [Электронный ресурс], 2025 – URL: https://habr.com/ru/articles/566218/ (дата обращения: 27.02.2025) – Текст: электронный.
8. Проектирование информационных систем: учебное пособие. — М.: Юрайт, 2018. — 310 с. [Электронный ресурс], 2025 — URL: https://urait.ru/book/proektirovanie-informacionnyh-sistem-490725 (дата обращения: 01.03.2025) – Текст: электронный.
9. Weje knowledge base [Электронный ресурс], 2025 – URL: https://weje.io/knowledgebase (дата обращения 02.03.2025) – Текст: электронный.
10. Quillscript документация [Электронный ресурс], 2025 – URL: https://quillscript.ink/ (дата обращения 02.03.2025) – Текст: электронный.
11. ГОСТ 19.201-78. «Техническое задание. Требования к содержанию и оформлению» [Электронный ресурс], 2025 – URL: https://www.swrit.ru/doc/espd/19.201-78.pdf (дата обращения 03.03.2025) – Текст: электронный.
12. Использование диаграммы классов UML при проектировании ПО [Электронный ресурс], 2025 – URL: https://habr.com/ru/articles/572234/ (дата обращения: 04.03.2025) – Текст: электронный.
13. Введение в компонентно-ориентированный подход к программированию [Электронный ресурс], 2025 – URL: https://habr.com/ru/articles/243479/ (дата обращение 06.03.2025) – Текст: электронный.
14. Описание архитектуры системы с помощью C4 model [Электронный ресурс], 2025 – URL: https://bool.dev/blog/detail/c4-model-architecture (дата обращение 07.03.2025) – Текст: электронный.
15. QT Documentation [Электронный ресурс], 2025 – URL: https://doc.qt.io/ (дата обращения: 15.03.2025) – Текст: электронный.
16. WxWidgets Documentation [Электронный ресурс], 2025 – URL: https://docs.wxwidgets.org/latest/ (дата обращения: 16.03.2025) – Текст: электронный.
17. Gtkmm Documentation [Электронный ресурс], 2025 – URL: https://gtkmm.gnome.org/en/documentation.html (дата обращения: 17.03.2025) – Текст: электронный.
18. Imgui Documentation [Электронный ресурс], 2025 – URL: https://github.com/ocornut/imgui/blob/master/docs/README.md (дата обращения: 18.03.2025) – Текст: электронный.
19. SFML Documentation [Электронный ресурс], 2025 – URL: https://www.sfml-dev.org/documentation/3.0.0/ (дата обращения: 19.03.2025) – Текст: электронный.
20. PostgreSQL Documentation [Электронный ресурс], 2025 – URL: https://www.postgresql.org/docs/ (дата обращения: 21.03.2025) – Текст: электронный.
21. SQLite Documentation [Электронный ресурс], 2025 – URL: https://www.sqlite.org/docs.html (дата обращения: 22.03.2025) – Текст: электронный.
22. MySQL Documentation [Электронный ресурс], 2025 – URL: https://dev.mysql.com/doc/ (дата обращения: 23.03.2025) – Текст: электронный.
23. MongoDB Documentation [Электронный ресурс], 2025 – URL: https://www.mongodb.com/docs/ (дата обращения: 24.03.2025) – Текст: электронный.
24. Visual Studio product family documentation – URL: https://learn.microsoft.com/en-us/visualstudio/ (дата обращения: 26.03.2025) – Текст: электронный.
25. Learning Rider [Электронный ресурс], 2025 – URL: https://www.jetbrains.com/rider/documentation/# (дата обращения: 27.03.2025) – Текст: электронный.
26. Quick documentation CLion [Электронный ресурс], 2025 – URL: https://www.jetbrains.com/help/clion/viewing-inline-documentation.html (дата обращения: 28.03.2025) – Текст: электронный.
27. Qt Creator Documentation [Электронный ресурс], 2025 – URL: https://doc.qt.io/qtcreator/index.html (дата обращения: 29.03.2025) – Текст: электронный.
28. Figma Learn [Электронный ресурс], 2025 – URL: https://help.figma.com/hc/en-us (дата обращения: 02.04.2025) – Текст: электронный.
29. Использование диаграммы классов UML при проектировании ПО [Электронный ресурс], 2025 – URL: https://habr.com/ru/articles/572234/ (дата обращения: 6.04.2025) – Текст: электронный.
30. ГОСТ 19.301-79 Единая система программной документации. Программа и методика испытаний. Требования к содержанию и оформлению. [Электронный ресурс], 2025 – URL: https://www.swrit.ru/doc/espd/19.301-79.pdf (дата обращения: 10.04.2025) – Текст: электронный.
31. Диаграмма Ганта: шаблоны и примеры [Электронный ресурс], 2025 – URL: https://getcompass.ru/blog/posts/diagramma-ganta (дата обращения: 13.04.2025) – Текст: электронный.
32. Гавриленко, Т.Ю. Методические рекомендации по выполнению организационно-экономической части выпускных квалификационных работ: метод. указания / Т. Ю. Гавриленко, О. В. Григоренко, Е. К. Ткаченко. – Москва: РТУ МИРЭА, 2019. – Электрон. опт. диск (ISO).

# ПРИЛОЖЕНИЯ

Приложение А. Графический материал: модель бизнес-процесса в нотации IDEF0

Приложение Б. Техническое задание по ГОСТ 19.201-78

Приложение В. Графический материал: дерево функций

Приложение Г. Исходный код DatabaseManager.cpp

Приложение Д. Исходный код функции exportNarrativeToJson

Приложение Е. Графический материал: Диаграмма Ганта

Приложение Ж. Графический материал (презентация ВКР)

## ПРИЛОЖЕНИЕ А

**Графический материал: модель бизнес-процесса в нотации IDEF0**

Рисунок А.1 – Уровень А0

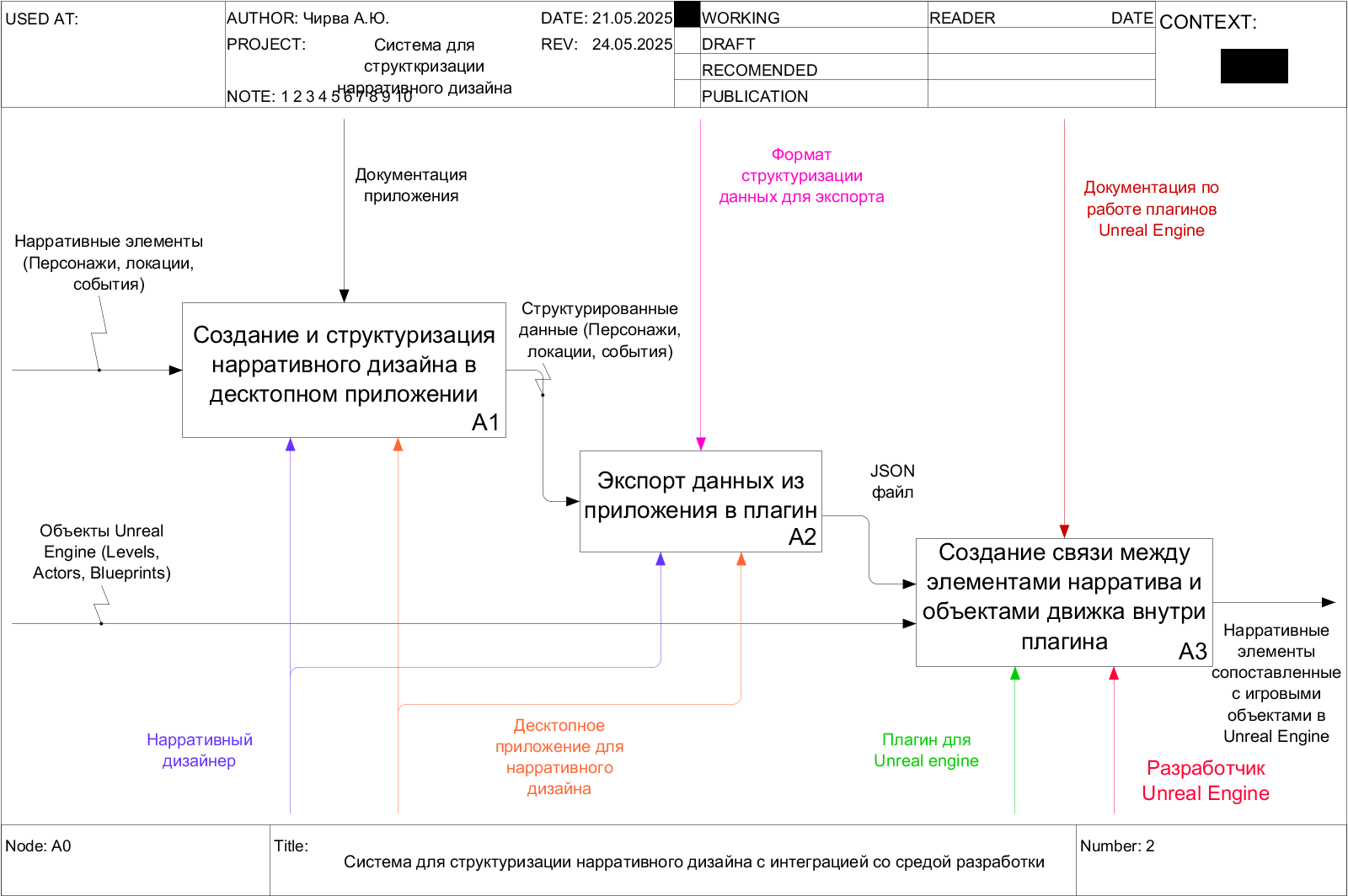
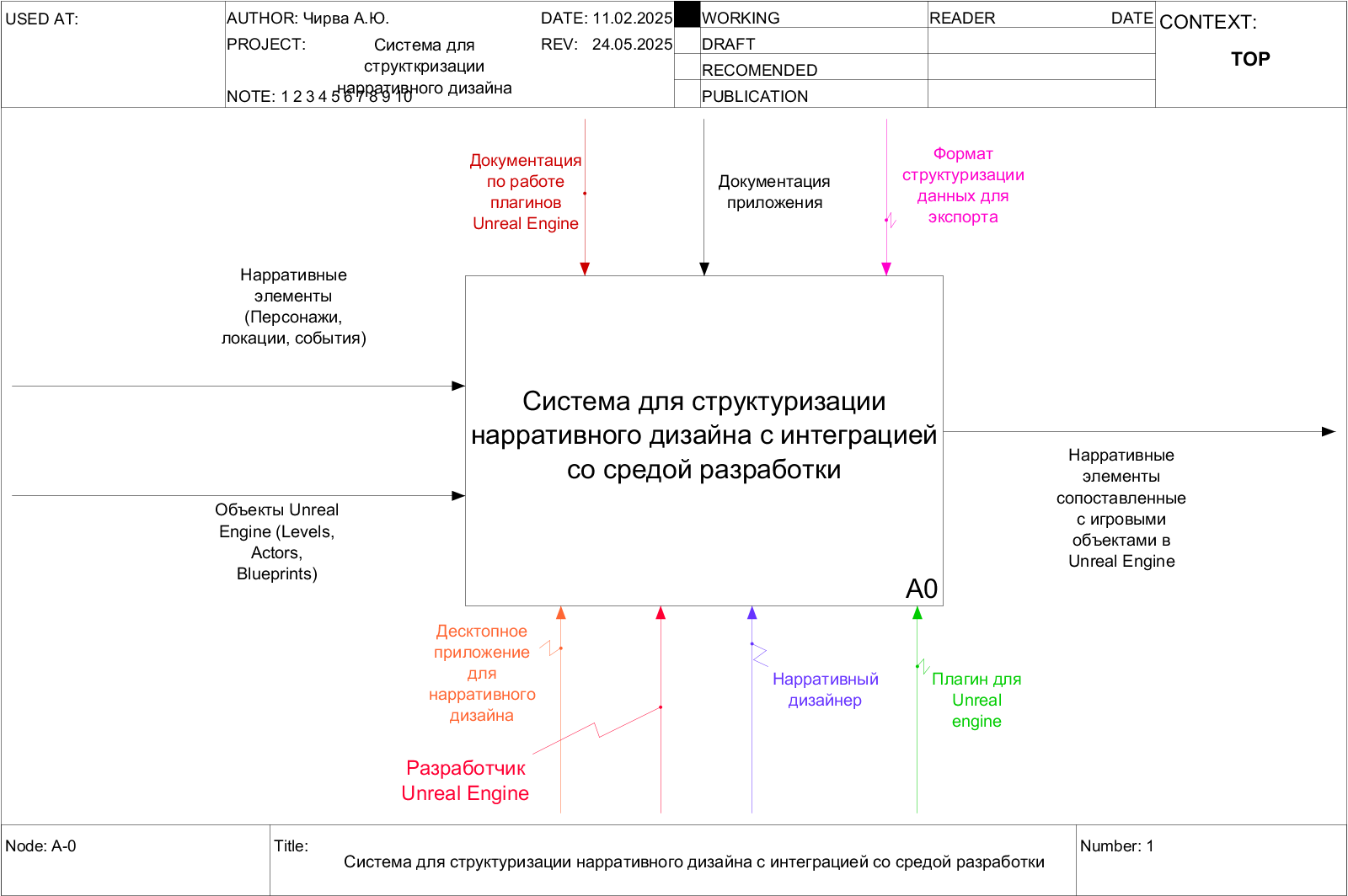


Рисунок А.2 – Декомпозиция блока А0

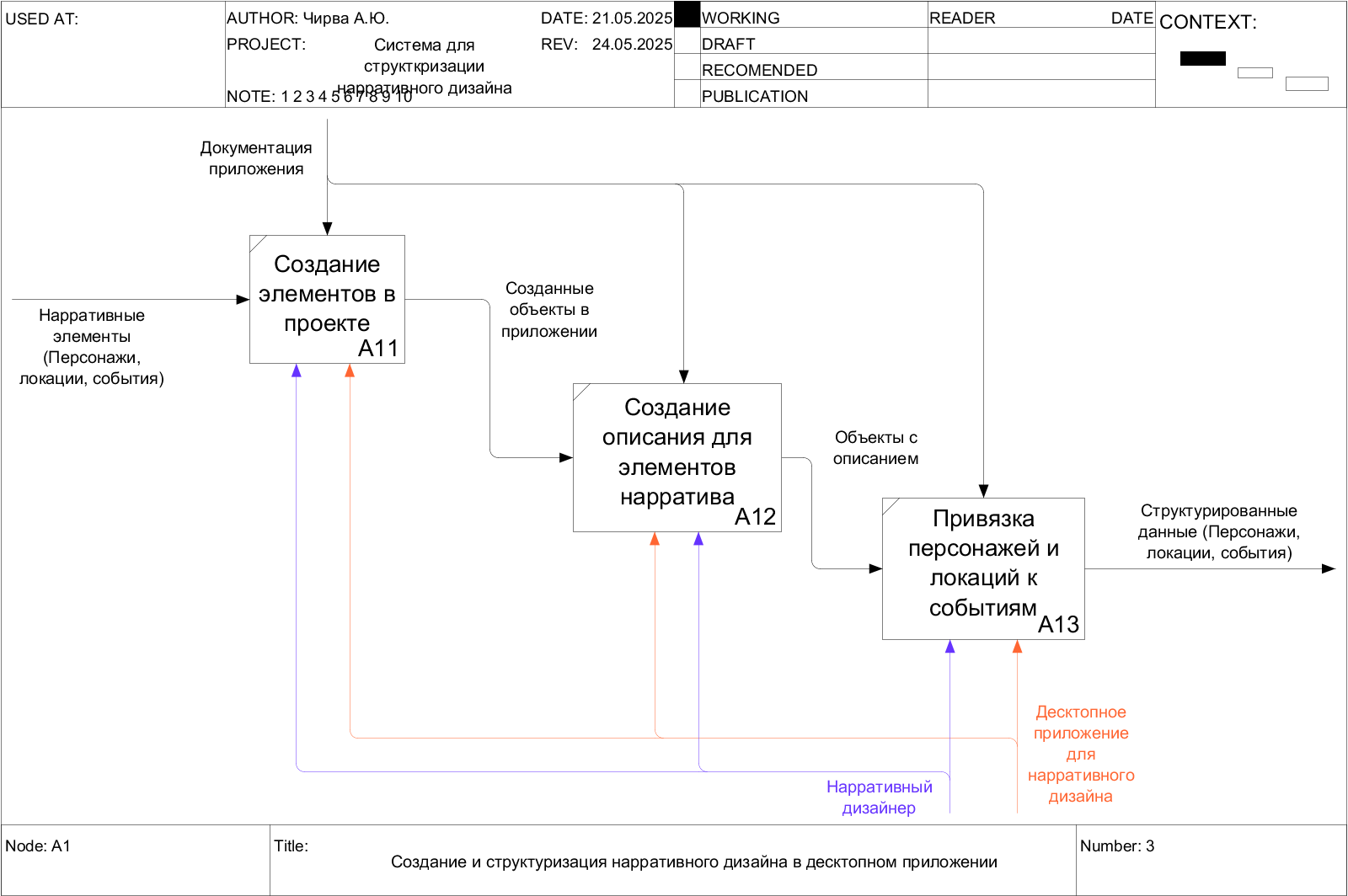


Рисунок А.3 – Декомпозиция блока А1

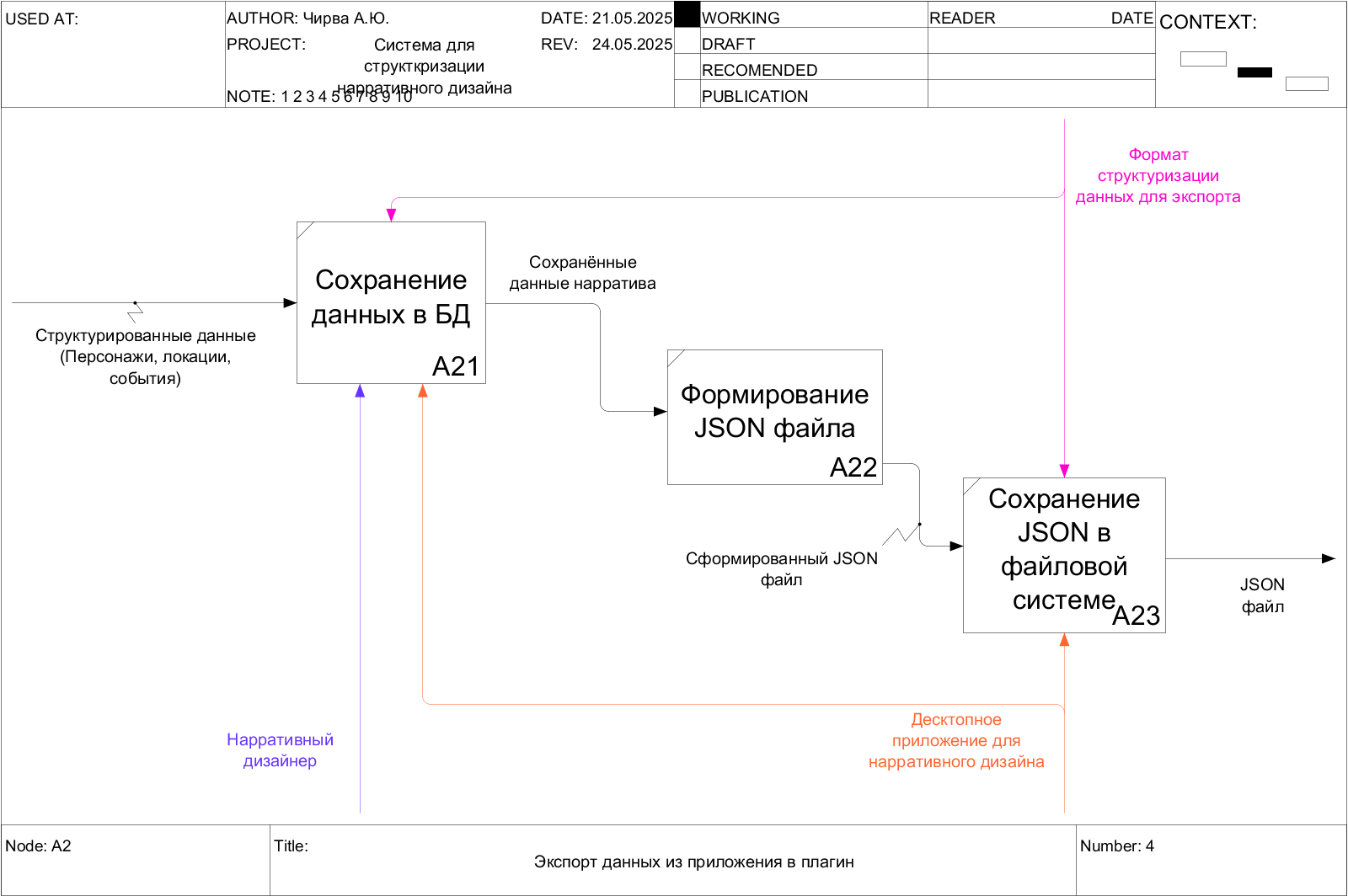


Рисунок А.4 – Декомпозиция блока А2

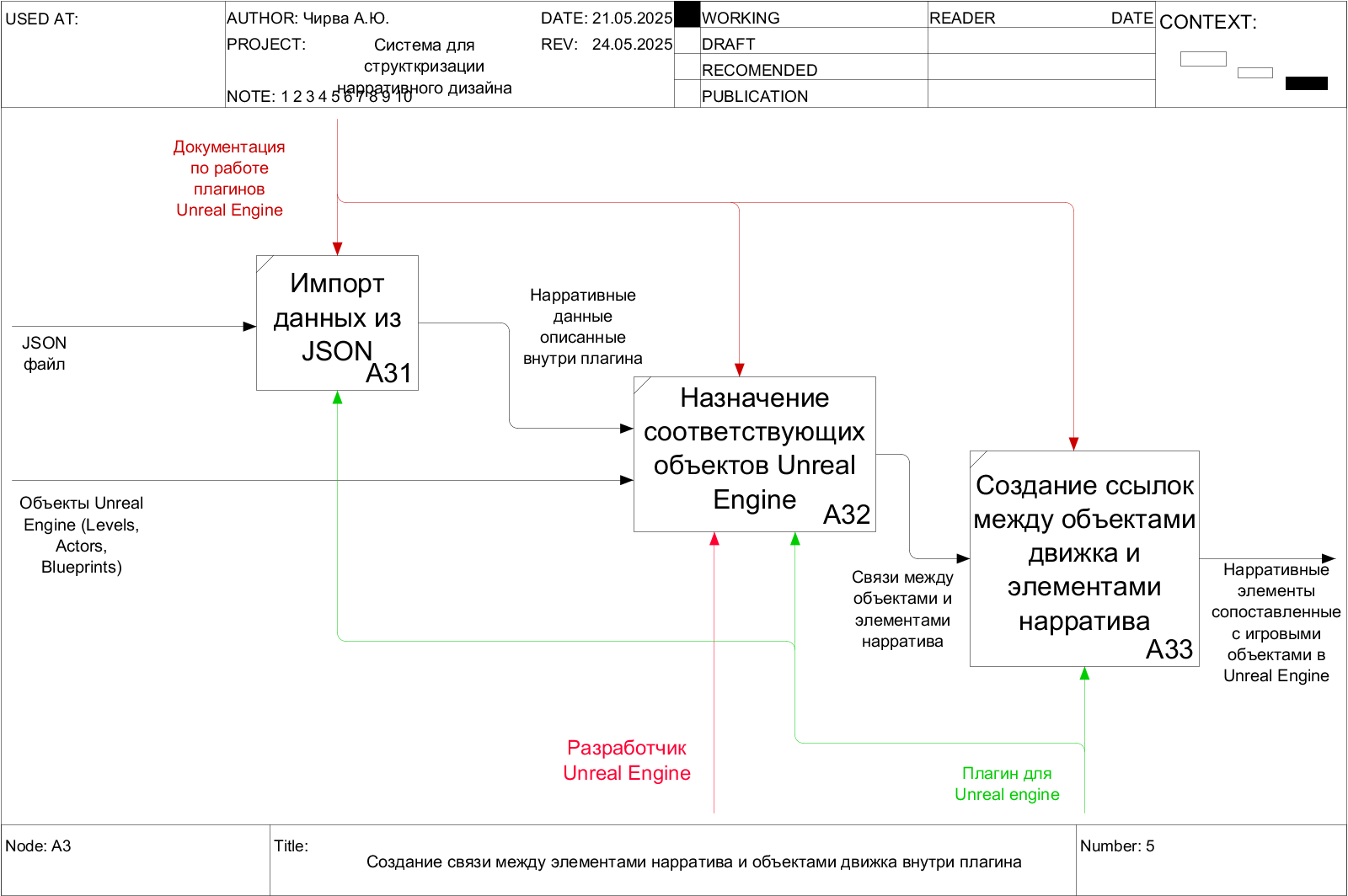


Рисунок А.5 – Декомпозиция блока А3

## ПРИЛОЖЕНИЕ Б

**Техническое задание по ГОСТ 19.201-78**

**Б.1 Введение**

Настоящее техническое задание определяет требования к разработке системы для структуризации нарративного дизайна с интеграцией в игровую среду Unreal Engine, состоящей из двух взаимосвязанных компонентов: десктопного приложения и плагина для движка.

**Б.2 Основание для разработки**

Наименование системы: Система структуризации нарративного дизайна с интеграцией в Unreal Engine. Условное обозначение: NSP.  
Шифр темы: NSP-UE5. Номер контракта: №1/11-11-2025 от 10.03.2025.

Заказчиком системы является РТУ МИРЭА. Разработчиком системы является ООО «Чирва».

При разработке системы и создании проектной и эксплуатационной документации разработчик должен руководствоваться следующими нормативными документами:

* ГОСТ 19.106-78. Единая система программной документации.
* ГОСТ Р 59793-2021. Информационные технологии. Автоматизированные системы. Стадии создания.
* ГОСТ 34.602–2020 Техническое задание на создание автоматизированной системы.
* ГОСТ Р 59795-2021. Информационные технологии. Автоматизированные системы. Требования к содержанию документов.
* ГОСТ 34.201–2020. Информационные технологии. Виды, комплектность и обозначение документов.

**Б.3 Назначение разработки**

Система предназначена для автоматизации процессов создания, структурирования и интеграции нарративного контента в игровые проекты, разработанные на платформе Unreal Engine 5, с целью улучшения взаимодействия нарративных дизайнеров и программистов.

**Б.4 Требования к системе**

**Б.4.1 Требования к функциональным характеристикам**

Система состоит из двух модулей:

1. Десктопное приложение

* Создание, редактирование и удаление персонажей, локаций и событий.
* Визуализация связей между элементами.
* Экспорт данных в формате JSON.
* Совместная работа нескольких дизайнеров.

1. Плагин для Unreal Engine 5

* Импорт данных в формате JSON.
* Привязка элементов нарратива к объектам движка (Actors, Levels, Blueprints).

**Б.4.2 Требования к надёжности**

Программное обеспечение не должно выходить из строя более чем на 3 минуты. Для устойчивости к потере данных необходимо регулярно производить выгрузку хранимой информации. Уровень надежности должен достигаться согласованным применением организационных, организационно технических мероприятий и программно-аппаратных средств. Надежность должна обеспечиваться за счет:

* Использование подходящих технических средств, соответствующих требованиям.
* Соблюдения правил эксплуатации и технического обслуживания программно-аппаратных средств.

**Б.4.3 Условия эксплуатации**

Техническим обслуживанием, ремонтом и хранением аппаратных средств занимаются сетевые инженеры-техники, специалисты по серверным технологиям и специалисты технической поддержки разработчика.

**Б.4.4 Требования к составу и параметрам технических средств**

Требования к составу и параметрам технических средств представлены в таблице Б.1.

Таблица Б.1 – Требования к техническому обеспечению

|  |  |  |
| --- | --- | --- |
| **Параметр** | **Минимальные требования** | **Рекомендуемые требования** |
| Операционная система | Windows 10 | Windows 11 |
| Процессор | Intel core i5-8400 / Amd ryzen 5 3600 | Intel core i7 9700 / Amd ryzen 5 3600X |
| Свободное место на жестком диске для установки | Не менее 200МБ | Не менее 200МБ |
| Размер оперативной памяти | 4 ГБ | 8 Гб |

**Б.4.5 Требования к информационной и программной совместимости**

Система должна соблюдать следующие требования к совместимости:

* Поддержка форматов данных: JSON.
* Совместимость с Unreal Engine версии >= 5.
* Исходный код: язык C++ и QML.
* Локализация интерфейса: русский и английский языки.
* Адаптивный интерфейс под разные разрешения экрана.

**Б.4.6 Требования к маркировке и упаковке**

Требования к маркировке и упаковке не предъявляются.

**Б.4.7 Требования к транспортированию и хранению**

Требования к транспортированию и хранению не предъявляются.

**Б.4.8 Специальные требования**

Специальные требования не предъявляются.

**Б.5 Требования к программной документации**

Проектная документация должна быть разработана в соответствии с ГОСТ 19.201-78 и должна включать следующие материалы:

* Руководство пользователя.
* Административная документация по развертыванию системы (инструкция по установке и настройке).
* Эксплуатационная документация.

Документация должна быть представлена в электронном виде в формате PDF.

**Б.6 Технико-экономические показатели**

Технико-экономические показатели не предъявляются.

**Б.7 Организация и планирование работ**

Стадии и этапы разработки представлены в таблице Б.2.

Таблица Б.2 – Стадии и этапы разработки

|  |  |  |
| --- | --- | --- |
| **Этапы работ** | **Содержание работ** | **Сроки** |
| Исследование и обоснование создания ИС | Поиск и анализ аналогов, исследование предметной области | 01.02.2025 – 15.02.2025 |
| Составление технического задания | Разработка функциональных и нефункциональных требований к системе | 16.02.2025 – 28.02.2025 |
| Эскизное проектирование | Разработка предварительных решений по выбранной теме | 01.03.2025 – 09.03.2025 |
| Техническое проектирование | Разработка диаграмм | 10.03.2025 – 17.03.2025 |
| Разработка макетов интерфейса | 18.03.2025 – 31.03.2025 |
| Разработка программной части | Разработка десктопного приложения | 01.04.2025 – 15.04.2025 |
| Разработка плагина | 16.04.2025 – 30.04.2025 |
| Разработка системы импорта и экспорта | 01.05.2025 – 15.05.2025 |
| Разработка БД | 15.05.2025 – 30.05.2025 |
| Тестирование | Проверка работоспособности системы в условиях, приближённых к реальным | 01.06.2025 – 05.06.2025 |
| Опытная эксплуатация | Эксплуатация с привлечением небольшого количества участников | 06.06.2025 – 12.06.2025 |
| Устранение замечаний, выявленных при эксплуатации | 13.06.2025 – 19.06.2025 |
| Ввод в эксплуатацию | Приёмка ИС в эксплуатацию | 20.06.2025 – 30.06.2025 |

Б.8 Порядок контроля и приёмки

В соответствии с разделом 5 необходимо на каждой стадии создания системы установить контроль и приемку результатов работ. На стадии 5 происходит прием готовой версии программного продукта (модели), а остальные результаты работ представляются в виде документов согласно таблице Б.2. Приемка этапа включает в себя рассмотрение и оценку объема работ и предоставленной технической документации в соответствии с требованиями технического задания. Организацию и проведение приемки системы должен осуществлять заказчик, а приемка системы должна производиться только после того, как будут выполнены все задачи системы. Заказчик обязан предоставить материальную часть (технические средства), проектную документацию и специально выделенный персонал. Последним этапом при приемке системы является составление акта приемки. Испытание системы будет происходить посредством ручного тестирования функций приложений и нагрузочного автоматизированного тестирования с помощью созданных утилит.

## ПРИЛОЖЕНИЕ В

**Графический материал: дерево функций**

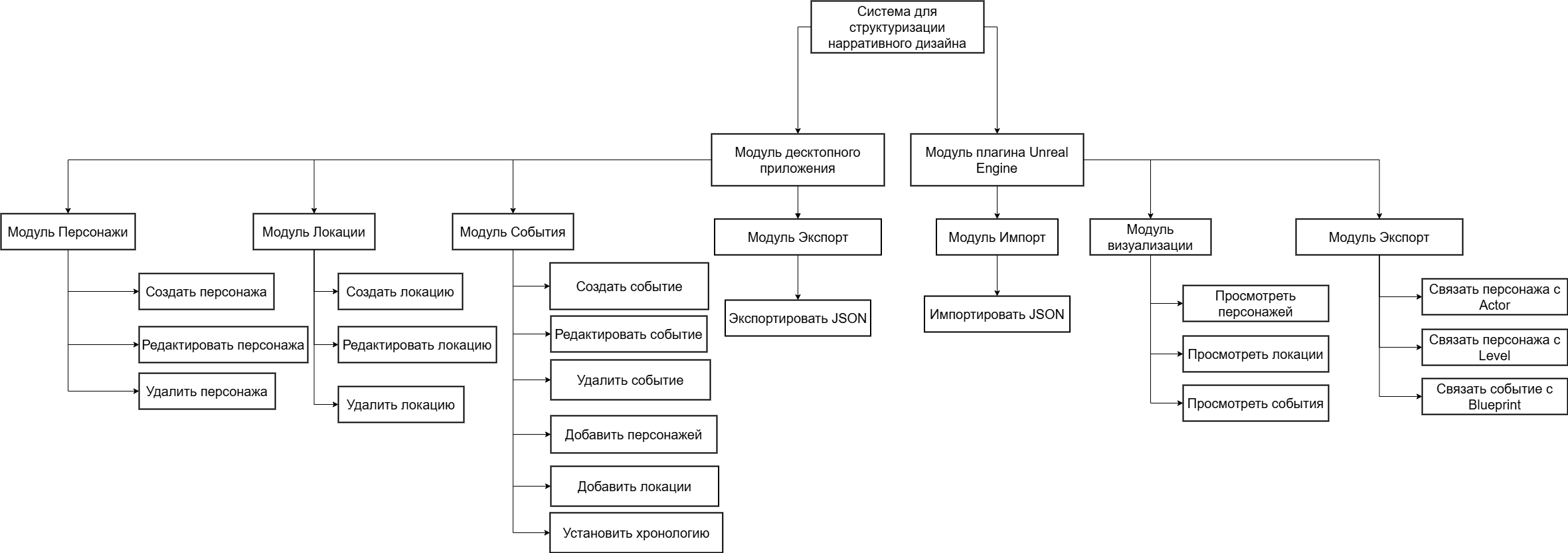


Рисунок В.1 – Дерево функций

## ПРИЛОЖЕНИЕ Г

**Исходный код DatabaseManager.cpp**

Листинг Г.1 – База данных приложения

|  |
| --- |
| #include "DatabaseManager.h"  SchemaManager::SchemaManager(const QSqlDatabase& db)  : m\_db(db)  {}  bool SchemaManager::exec(const QString& sql)  {  QSqlQuery query(m\_db);  if (!query.exec(sql)) {  qCritical() << "SQL error:" << query.lastError().text()  << "\nwhen executing:" << sql;  return false;  }  return true;  }  bool SchemaManager::createSchema()  {  bool ok = true;  ok &= exec(R"(  CREATE TABLE IF NOT EXISTS project (  project\_id SERIAL PRIMARY KEY,  title VARCHAR(255) NOT NULL,  description TEXT,  created\_at TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,  owner\_user\_id INTEGER  );  )");  ok &= exec(R"(  CREATE TABLE IF NOT EXISTS character\_ (  character\_id SERIAL PRIMARY KEY,  project\_id INTEGER NOT NULL REFERENCES project(project\_id) ON DELETE CASCADE,  name VARCHAR(255) NOT NULL,  description TEXT,  image\_path VARCHAR(512)  );  )"); |

Продолжение листинга Г.1

|  |
| --- |
| ok &= exec(R"(  CREATE TABLE IF NOT EXISTS location (  location\_id SERIAL PRIMARY KEY,  project\_id INTEGER NOT NULL REFERENCES project(project\_id) ON DELETE CASCADE,  name VARCHAR(255) NOT NULL,  description TEXT,  image\_path VARCHAR(512)  );  )");  ok &= exec(R"(  CREATE TABLE IF NOT EXISTS event\_ (  event\_id SERIAL PRIMARY KEY,  project\_id INTEGER NOT NULL REFERENCES project(project\_id) ON DELETE CASCADE,  title VARCHAR(255) NOT NULL,  description TEXT,  event\_date TIMESTAMP,  timeline\_index INTEGER  );  )");  ok &= exec(R"(  CREATE TABLE IF NOT EXISTS event\_character (  event\_id INTEGER NOT NULL REFERENCES event\_(event\_id) ON DELETE CASCADE,  character\_id INTEGER NOT NULL REFERENCES character\_(character\_id) ON DELETE CASCADE,  role VARCHAR(255),  PRIMARY KEY (event\_id, character\_id)  );  )");  ok &= exec(R"(  CREATE TABLE IF NOT EXISTS event\_location (  event\_id INTEGER NOT NULL REFERENCES event\_(event\_id) ON DELETE CASCADE,  location\_id INTEGER NOT NULL REFERENCES location(location\_id) ON DELETE CASCADE,  PRIMARY KEY (event\_id, location\_id)  );  )");  ok &= exec(R"(  CREATE TABLE IF NOT EXISTS character\_relation (  relation\_id SERIAL PRIMARY KEY,  project\_id INTEGER NOT NULL REFERENCES project(project\_id) ON DELETE CASCADE,  character\_a\_id INTEGER NOT NULL REFERENCES |

Продолжение листинга Г.1

|  |
| --- |
| character\_(character\_id) ON DELETE CASCADE,  character\_b\_id INTEGER NOT NULL REFERENCES  character\_(character\_id) ON DELETE CASCADE,  relation\_type VARCHAR(255)  );  )");  ok &= exec(R"(  CREATE TABLE IF NOT EXISTS ue\_link (  link\_id SERIAL PRIMARY KEY,  project\_id INTEGER NOT NULL REFERENCES project(project\_id) ON DELETE CASCADE,  entity\_type VARCHAR(50) NOT NULL,  entity\_id INTEGER NOT NULL,  ue\_object\_path VARCHAR(512)  );  )");  return ok;  } |

## ПРИЛОЖЕНИЕ Д

**Исходный код функции exportNarrativeToJson**

Листинг Д.1 – Функция exportNarrativeToJSON

|  |
| --- |
| #include <pqxx/pqxx>  #include <nlohmann/json.hpp>  #include <fstream>  #include <iostream>  using json = nlohmann::json;  struct NarrativeData {  json characters;  json locations;  json events;  json event\_characters;  json event\_locations;  json character\_relations;  json ue\_links;  };  NarrativeData fetchNarrativeData(pqxx::connection& conn, int project\_id) {  NarrativeData nd;  auto queryToJson = [&](const std::string& sql, json& arr) {  pqxx::work tx{conn};  pqxx::result r = tx.exec(sql);  for (auto row : r) {  json obj;  for (auto field : row) {  obj[field.name()] = field.is\_null() ? nullptr : field.as<std::string>();  }  arr.push\_back(std::move(obj));  }  };  queryToJson(  "SELECT character\_id, name, description, image\_path "  "FROM character "  "WHERE project\_id = " + std::to\_string(project\_id) + ";",  nd.characters  );  queryToJson(  "SELECT location\_id, name, description, image\_path "  "FROM location "  "WHERE project\_id = " + std::to\_string(project\_id) + ";", |

Продолжение листинга Д.1

|  |
| --- |
| nd.locations  );  queryToJson(  "SELECT event\_id, title, description, event\_date, timeline\_index "  "FROM event "  "WHERE project\_id = " + std::to\_string(project\_id) + ";",  nd.events  );  queryToJson(  "SELECT event\_id, character\_id, role "  "FROM event\_character "  "WHERE event\_id IN (SELECT event\_id FROM event WHERE project\_id = " + std::to\_string(project\_id) + ");",  nd.event\_characters  );  queryToJson(  "SELECT event\_id, location\_id "  "FROM event\_location "  "WHERE event\_id IN (SELECT event\_id FROM event WHERE project\_id = " + std::to\_string(project\_id) + ");",  nd.event\_locations  );  queryToJson(  "SELECT character\_a\_id, character\_b\_id, relation\_type "  "FROM character\_relation "  "WHERE project\_id = " + std::to\_string(project\_id) + ";",  nd.character\_relations  );  queryToJson(  "SELECT entity\_type, entity\_id, ue\_object\_path "  "FROM ue\_link "  "WHERE project\_id = " + std::to\_string(project\_id) + ";",  nd.ue\_links  );  return nd;  }  void exportNarrativeToJson(const std::string& conninfo,  int project\_id,  const std::string& output\_path) |

Продолжение листинга Д.1

|  |
| --- |
| {  try {  pqxx::connection conn{conninfo};  if (!conn.is\_open()) {  throw std::runtime\_error("Cannot open database connection");  }  NarrativeData nd = fetchNarrativeData(conn, project\_id);  json out;  out["project\_id"] = project\_id;  out["characters"] = nd.characters;  out["locations"] = nd.locations;  out["events"] = nd.events;  out["event\_characters"] = nd.event\_characters;  out["event\_locations"] = nd.event\_locations;  out["character\_relations"] = nd.character\_relations;  out["ue\_links"] = nd.ue\_links;  std::ofstream ofs(output\_path);  ofs << out.dump(4) << std::endl;  ofs.close();  std::cout << "Exported narrative data to " << output\_path << std::endl;  }  catch (const std::exception& e) {  std::cerr << "Error exporting narrative data: " << e.what() << std::endl;  }  } |

## ПРИЛОЖЕНИЕ Е

**Графический материал: диаграмма Ганта**

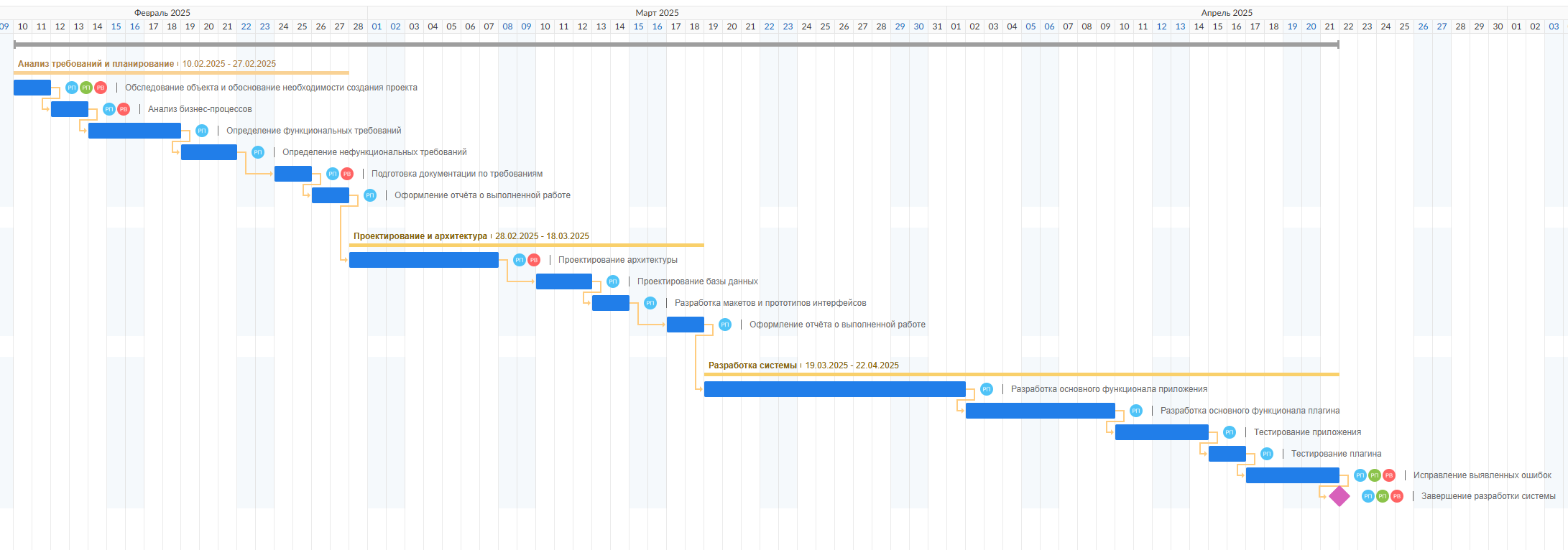


Рисунок Е.1 – Диаграмма Ганта

## ПРИЛОЖЕНИЕ Ж

**Графический материал (презентация ВКР)**

****

Рисунок Ж.1 – Слайд 1

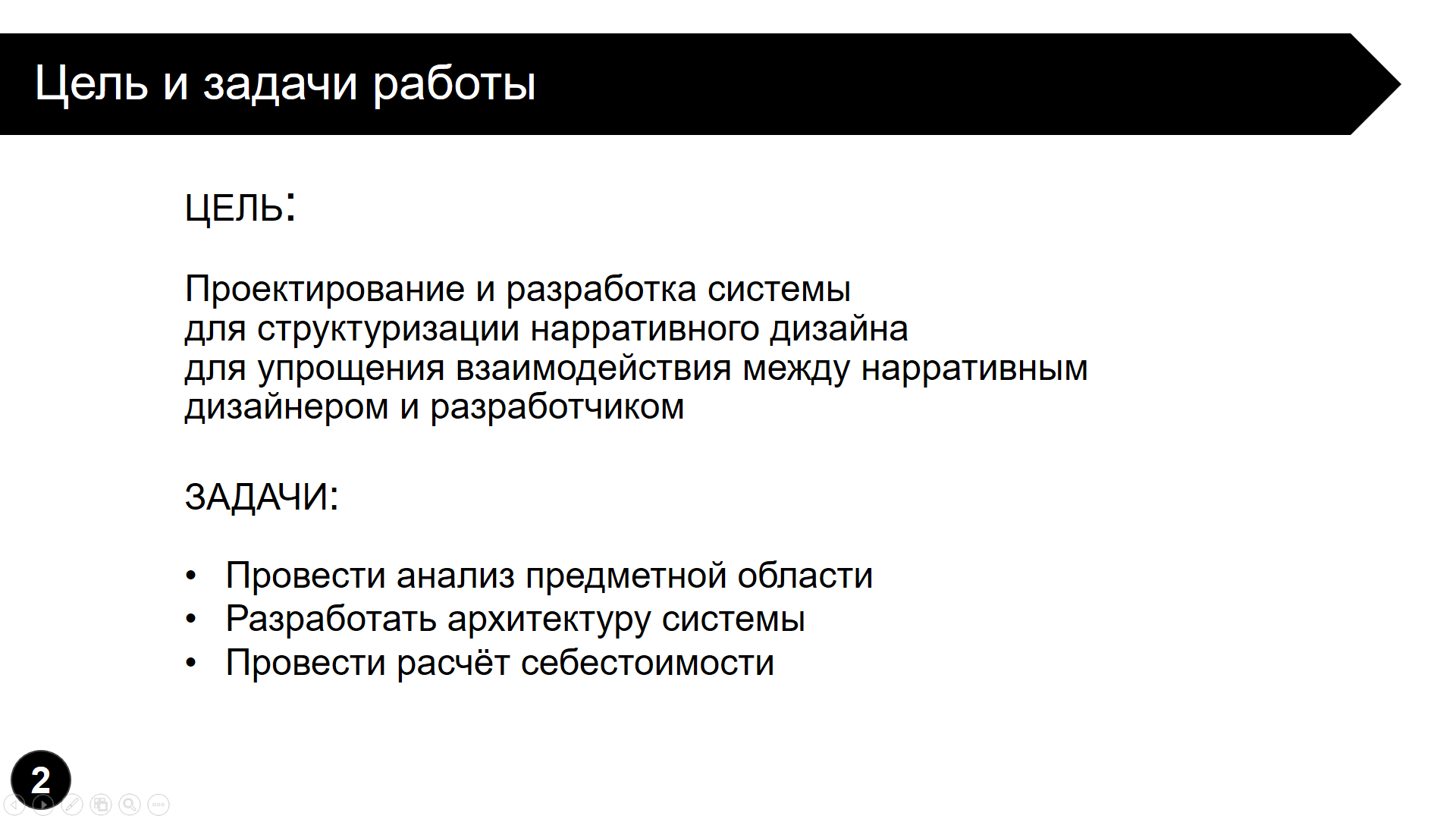


Рисунок Ж.2 – Слайд 2

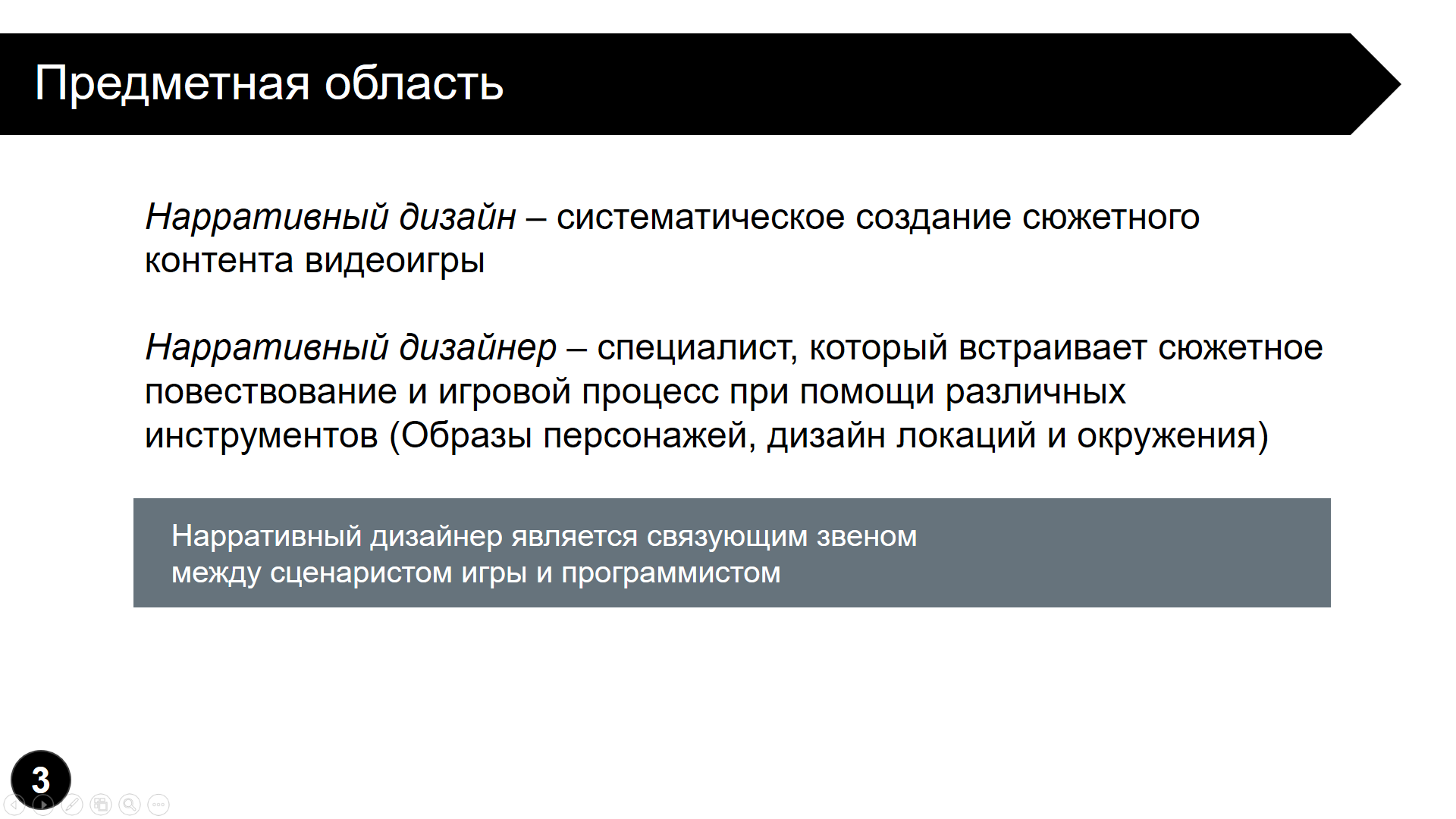


Рисунок Ж.3 – Слайд 3

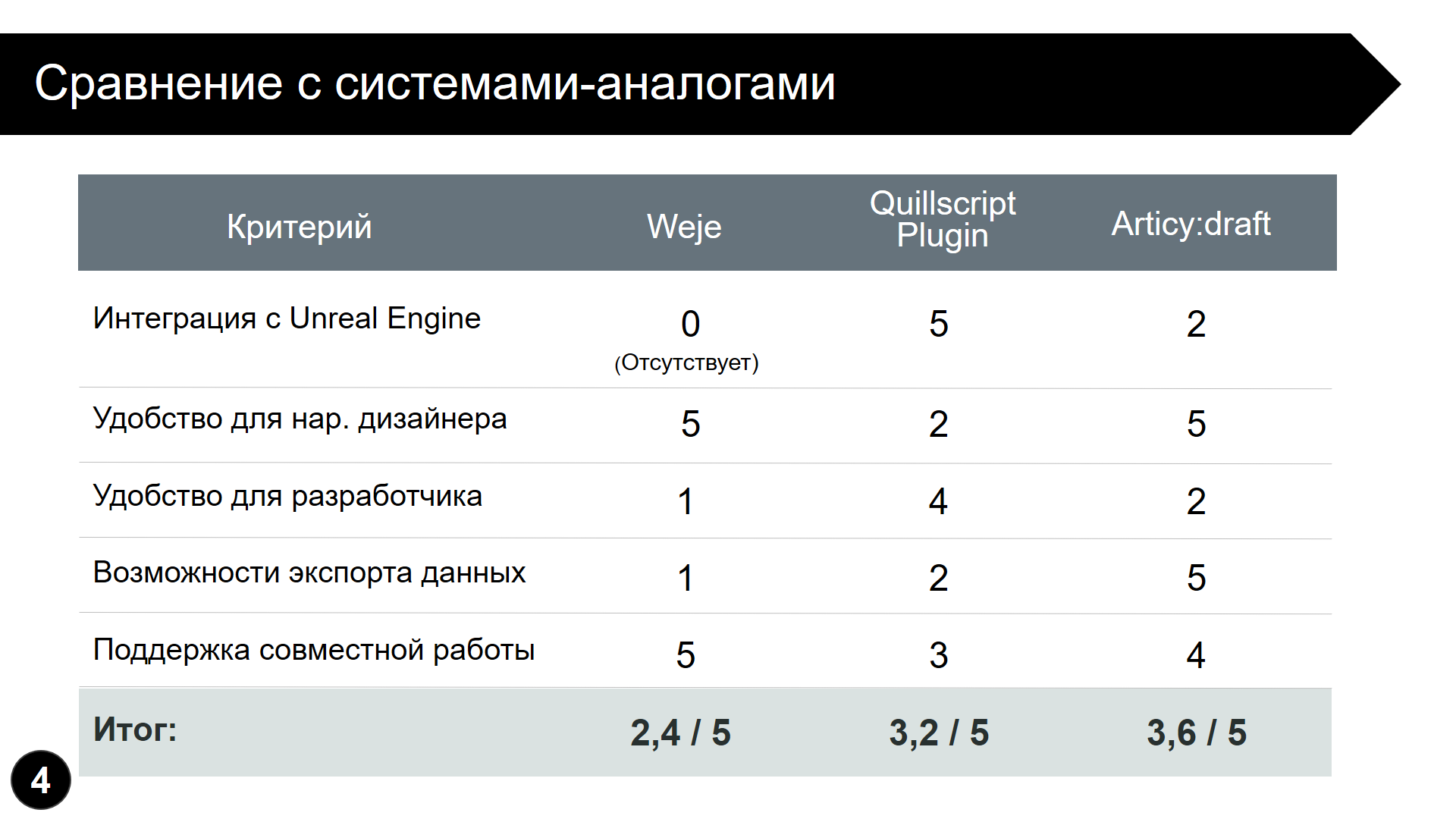


Рисунок Ж.4 – Слайд 4



Рисунок Ж.5 – Слайд 5

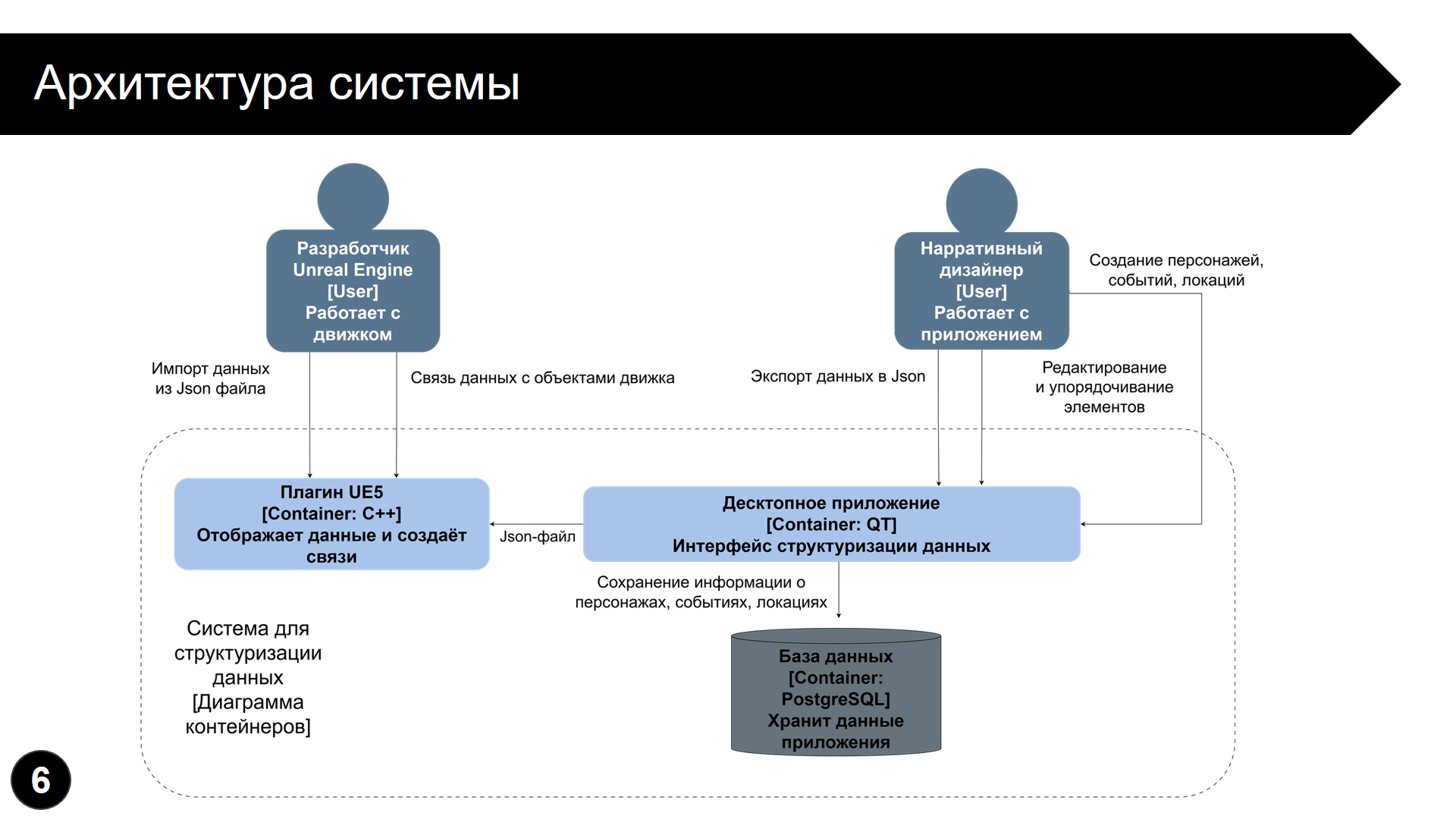


Рисунок Ж.6 – Слайд 6



Рисунок Ж.7 – Слайд 7



Рисунок Ж.8 – Слайд 8

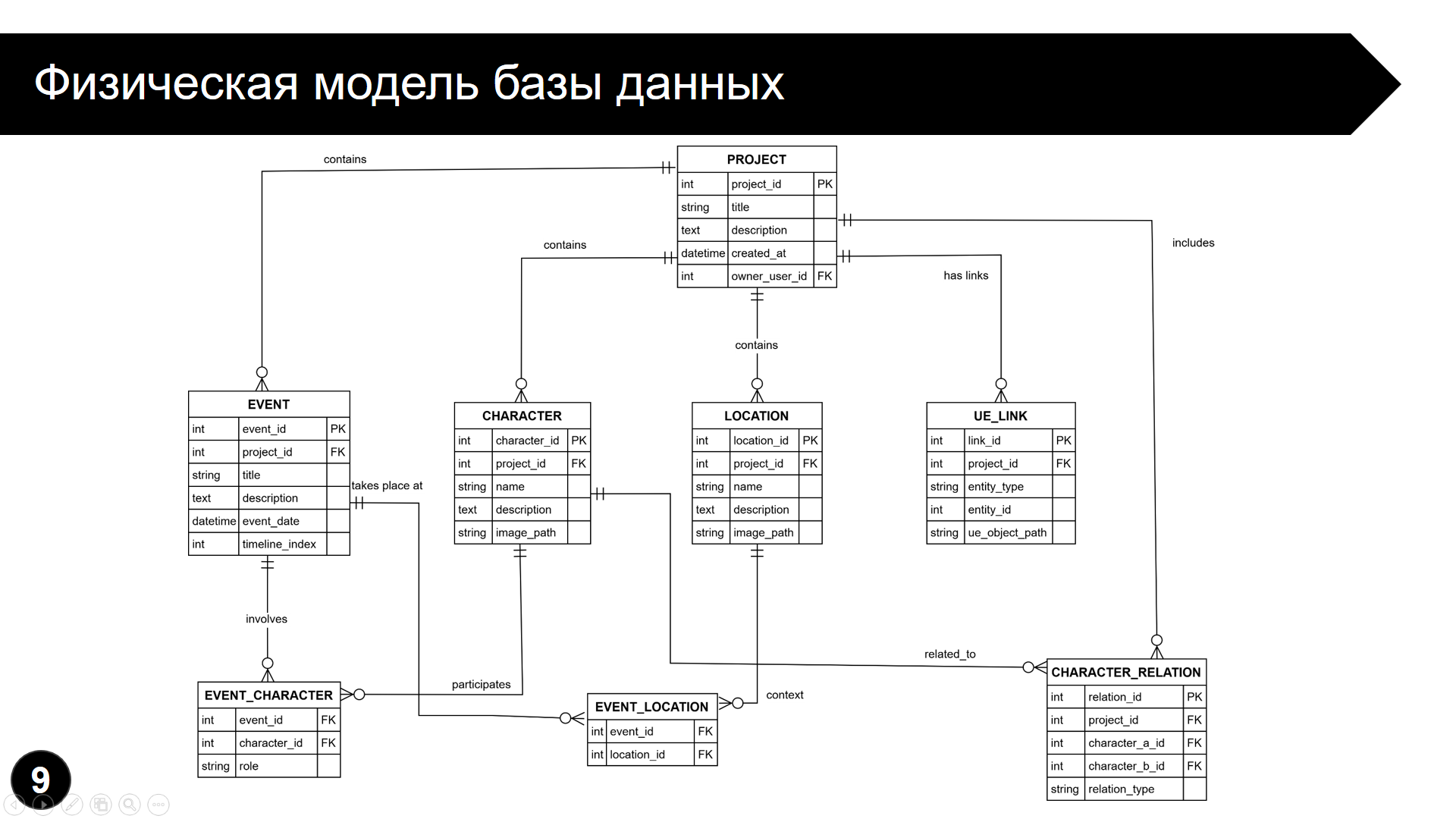


Рисунок Ж.9 – Слайд 9



Рисунок Ж.10 – Слайд 10

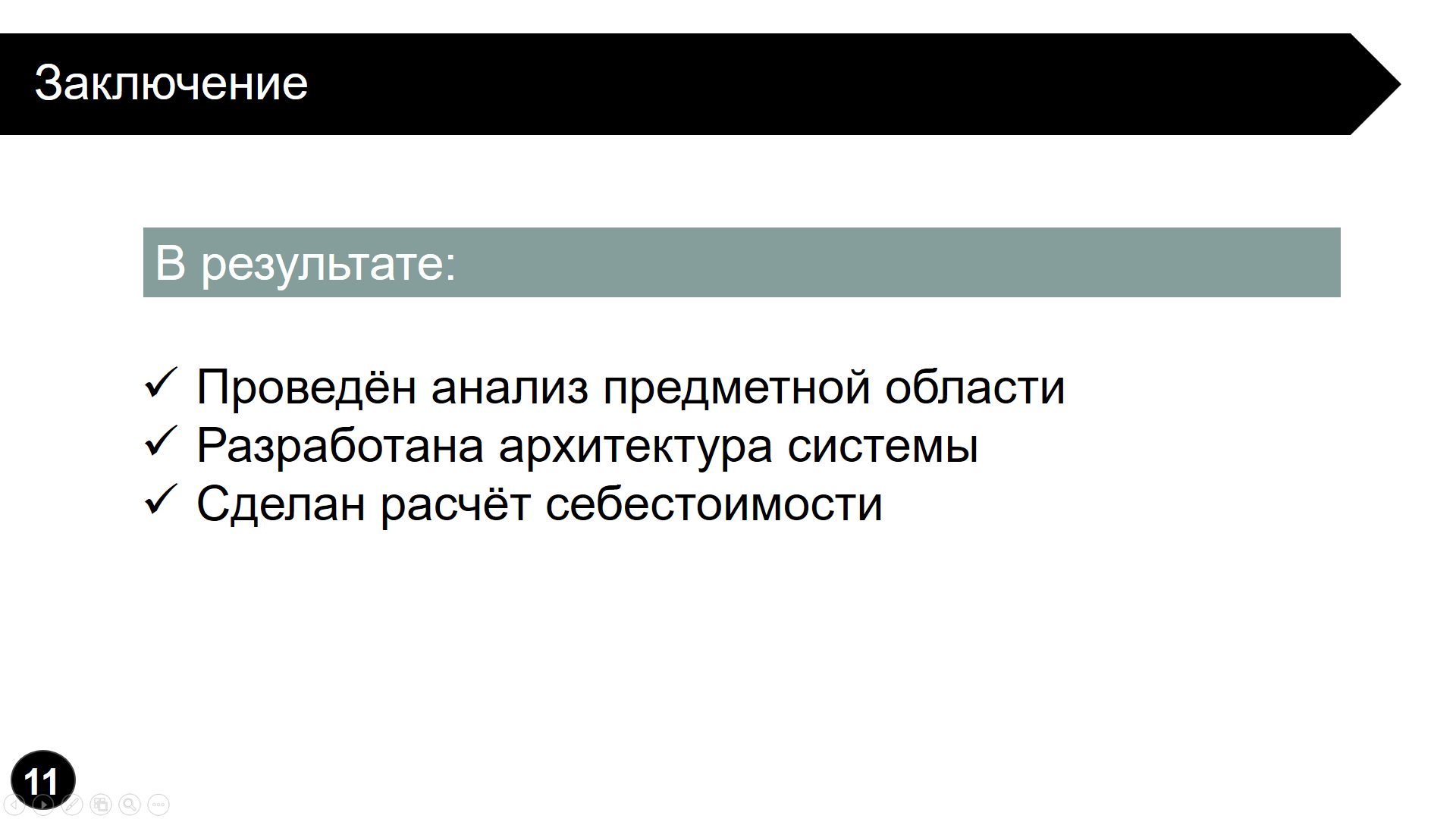


Рисунок Ж.11 – Слайд 11

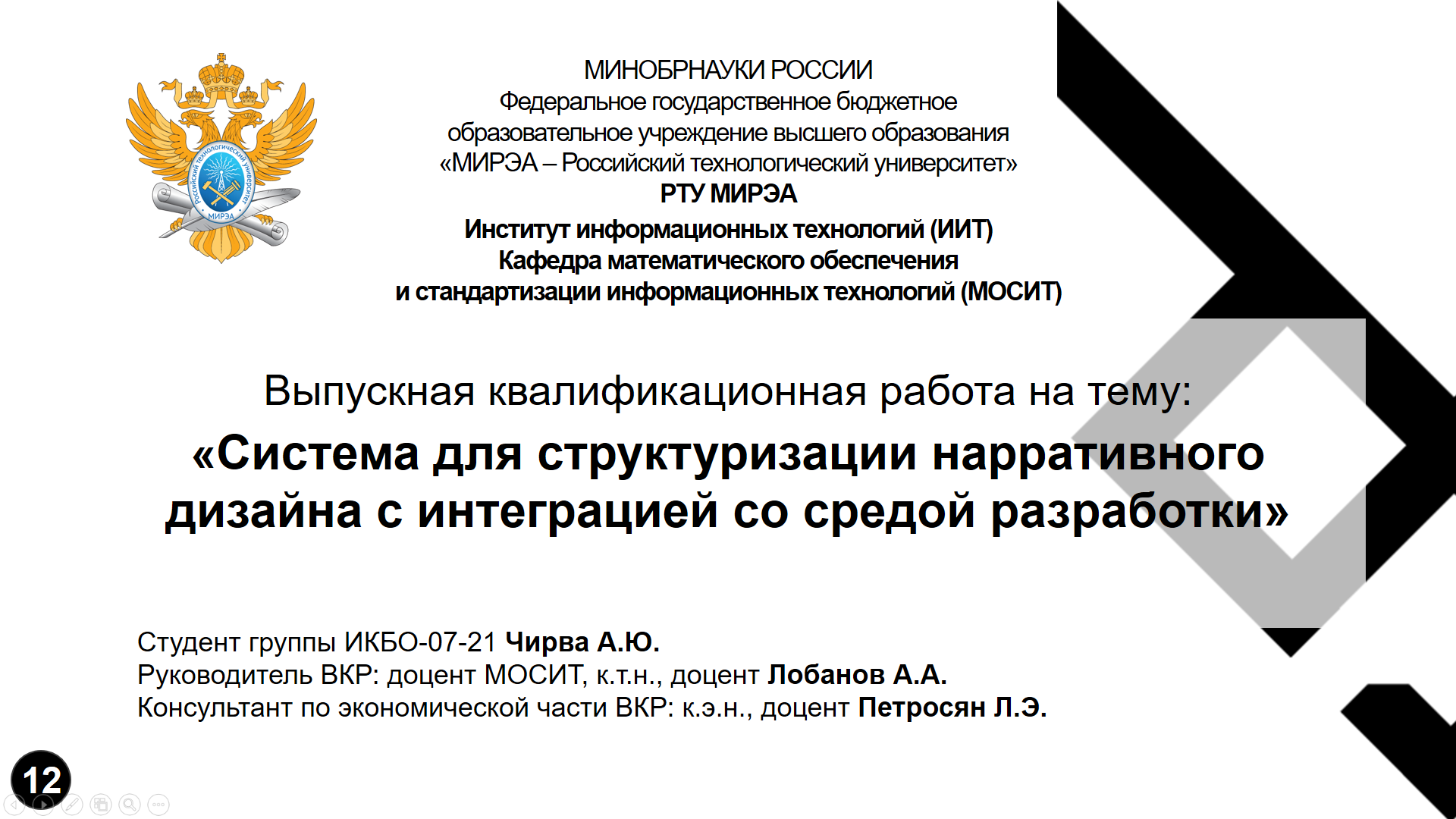


Рисунок Ж.12 – Слайд 12