МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Экономики и управления на предприятии природопользования»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(бакалаврская работа)
по направлению подготовки 09.03.03 Прикладная информатика
(квалификация – бакалавр)

На тему «Разработка информационной системы «ИИ -ассистент для мастеров настольных ролевых игр»

Исполнитель Лепкий Никита Михайлович

Руководитель к.т.н., Сафонова Татьяна Владимировна

«К защите допускаю»

Руководитель кафедры _____

кандидат экономических наук

Майборода Евгений Викторович

«22» 06 2026 г.

Филиал Российского государственного гидрометеорологического университета в г. Туапсе
НОРМОКОНТРОЛЬ ПРОЙДЕН
«20» 06 2026г.
Мартынова Ю.В.
ПОДПИСЬ РАСШИФРОВКА ПОДПИСИ

Туапсе
2026

ОГЛАВЛЕНИЕ

Введение

Настольные ролевые игры (НРИ) являются одним из наиболее динамично развивающихся сегментов досуговой индустрии как в мире, так и в Российской Федерации. По данным аналитических агентств, число активных игроков в мире превышает 50 миллионов человек, а в России, по различным оценкам, достигает 1–3 миллионов. Популяризации НРИ способствуют как известные веб-сериалы («Critical Role», «Dimension 20», российские «Подземелья и Драконы»), так и доступность цифровых инструментов для удалённой игры (виртуальные столы Foundry VTT, Roll20, голосовые чаты Discord). Однако ключевая фигура игрового процесса – мастер игры – по-прежнему испытывает значительные трудности, связанные с высокой когнитивной нагрузкой и творческим выгоранием.

Мастер игры выполняет множество ролей одновременно: он описывает локации, озвучивает десятки неигровых персонажей (НПС), интерпретирует правила, разрешает спорные ситуации, импровизирует в ответ на неожиданные действия игроков и отслеживает боевые ресурсы. Особенно остро проблема нехватки времени и творческого истощения проявляется в моменты, когда игроки «сворачивают с рельс» – принимают решение, которое мастер не предусмотрел при подготовке. В таких ситуациях мастеру необходимо за считанные секунды придумать новую локацию, свежего персонажа, увлекательный диалог или неожиданный сюжетный поворот. Если мастер не справляется, темп игры падает, игроки начинают скучать, а атмосфера погружения разрушается.

Современные информационные технологии, в частности генеративные нейросети и большие языковые модели (LLM – Large Language Models), открывают новые возможности для автоматизации творческих задач. Такие модели, как GPT-3.5/4, LLaMA 3, Qwen, способны генерировать связные, уникальные тексты, адаптированные под контекст запроса. Однако существующие программные продукты, использующие эти технологии (LitRPG

Adventures, Scribe, модули для Foundry VTT), обладают рядом недостатков: они платные, плохо локализованы на русский язык, не запоминают контекст игровой сессии и часто требуют от мастера технических знаний (например, регистрации API-ключа). Классические генераторы на основе случайных таблиц (Donjon, Chaotic Shiny) работают быстро, но выдают шаблонные результаты и не учитывают контекст кампании.

Таким образом, актуальность темы данной выпускной квалификационной работы обусловлена отсутствием на рынке бесплатного, русскоязычного, контекстно-зависимого и простого в использовании ИИ-ассистента для мастеров настольных ролевых игр.

Объект исследования – процесс подготовки и ведения настольной ролевой игры мастером, рассматриваемый как технологический и творческий процесс.

Предмет исследования – методы, алгоритмы и программные средства использования генеративных нейросетей (больших языковых моделей) для автоматизации и облегчения творческих задач мастера НРИ.

Цель выпускной квалификационной работы – разработать информационную систему «ИИ-ассистент для мастеров настольных ролевых игр» (рабочее название – GM AI Assistant), обеспечивающую генерацию имён, описаний локаций, диалогов, сюжетных поворотов, блоков монстров и ответов на вопросы по правилам с учётом контекста игровой сессии.

Для достижения поставленной цели необходимо решить следующие задачи:

1) Провести анализ предметной области, выявить проблемные места в деятельности мастера НРИ, требующие автоматизации.

2) Обосновать выбор задачи и сформулировать техническое задание на проектирование информационной системы.

3) Описать экономико-информационную сущность задачи, выделить входные и выходные данные, бизнес-правила.

4) Разработать информационное обеспечение задачи: структуру базы

данных, классификаторы, форматы входных и выходных сообщений.

5) Обосновать проектные решения по технологическому, техническому и программному обеспечению.

6) Разработать архитектуру программного обеспечения и описать основные программные модули.

7) Создать руководство пользователя с описанием интерфейса и порядка работы.

8) Выполнить расчёт экономической эффективности разработанной информационной системы.

Практическая значимость работы заключается в том, что разработанная информационная система может быть использована мастерами настольных ролевых игр для снижения временных затрат на генерацию контента, уменьшения стресса и повышения качества игрового процесса. Система может быть развёрнута как на сервере (веб-версия), так и локально на компьютере мастера.

Методы исследования. В работе использовались общенаучные методы (анализ, синтез, классификация, сравнение), методы системного анализа, проектирования информационных систем, а также методы промпт-инжиниринга для больших языковых моделей.

1 Аналитическая часть

1.1 Анализ предметной области

Общая специфика деятельности мастера настольных ролевых игр

Настольная ролевая игра представляет собой коллективное творчество, в котором несколько игроков (обычно от 2 до 6) управляют своими персонажами, а один участник – мастер игры (game master, GM, рассказчик, ведущий) – управляет всем остальным миром. По данным опроса, проведённого в рамках подготовки к данной работе среди 15 мастеров со стажем от 1 года до 10 лет, можно выделить следующие ключевые роли мастера[28]:

Рассказчик – описывает локации, события, погоду, запахи, звуки. На это уходит в среднем 30% времени игровой сессии.

Актёр – озвучивает неигровых персонажей (NPC), передаёт их эмоции и мотивы. Занимает около 25% времени.

Арбитр – интерпретирует правила, разрешает споры, назначает сложность проверок. Около 20% времени.

Импровизатор – реагирует на действия игроков, которые не предусмотрены сценарием. Около 15% времени.

Менеджер ресурсов – отслеживает инициативу, здоровье монстров, заклинания, золото. Около 10% времени.

Обоснование актуальности разработки

Выбор задачи обосновывается следующими факторами:

Социальная значимость: ИИ-ассистент поможет тысячам мастеров НРИ снизить стресс, уменьшить время подготовки и улучшить качество игровых сессий.

Экономическая целесообразность: разработанная система бесплатна для конечного пользователя, что делает её доступной для широкой аудитории.

Техническая новизна: комбинация больших языковых моделей, контекстной памяти и специализированных промптов под НРИ не имеет аналогов в бесплатном сегменте.

Образовательная ценность: проект демонстрирует практическое применение методов промпт-инжиниринга и интеграции LLM в прикладные информационные системы.

Проблемные зоны деятельности мастера. На основе наблюдения за работой мастеров были выявлены следующие наиболее критичные проблемные зоны[28]:

Генерация контента «на лету» (импровизация). Когда игроки принимают неожиданное решение, мастеру приходится за 10–30 секунд придумывать новых NPC, названия таверн, имена, диалоги. Это вызывает стресс и снижает качество повествования. 93% опрошенных мастеров отметили эту проблему как наиболее острую. Поддержание темпа игры. Поиск в книгах правил нужного параграфа или таблицы занимает 30–60 секунд, что ломает динамику сцены. 87% мастеров сталкиваются с этим регулярно.

Учёт боевых ситуаций. При 4–6 игроках и 3–8 монстрах отслеживание очков здоровья, эффектов и инициативы становится сложным. 80% мастеров отмечают эту проблему.

Творческое истощение. После 3–4 часов игры мастер устаёт придумывать описания, и они становятся шаблонными. 73% мастеров подтверждают данную проблему.

1.2 Анализ существующих информационных систем-аналогов

Анализ существующих разработок проводился по следующим критериям: функциональность, стоимость, поддержка русского языка, наличие контекстной памяти, простота использования[4].

Классические генераторы на случайных таблицах. Donjon, Chaotic Shiny, Fantasy Name Generators предлагают огромные базы имён, описаний, сюжетных завязок. Достоинства: мгновенная работа (миллисекунды), бесплатность, предсказуемость результата. Недостатки: шаблонность, полное отсутствие учёта контекста кампании, ограниченность англоязычных баз (русских версий

мало). Эти инструменты не могут связывать предыдущие генерации с последующими.

Современные LLM-сервисы. LitRPG Adventures, Scribe, AI Dungeon Master Assistant используют большие языковые модели (GPT-3.5/4). Они генерируют уникальные, связные тексты. Однако их недостатки: платная подписка (от 5 до 20 долларов в месяц), плохая поддержка русского языка, отсутствие или ограниченная контекстная память, необходимость регистрации API-ключа.

Модули для виртуальных столов (VTT). Модуль «ChatGPT GM Assistant» для Foundry VTT позволяет отправлять запросы к OpenAI прямо из игрового интерфейса. Достоинства: интеграция с VTT. Недостатки: требует API-ключа OpenAI (оплата токенов), работает только внутри Foundry VTT, не имеет специализированных промптов под НРИ.

Открытые проекты (open source). На GitHub существуют проекты «rpg-llm-assistant», «gpt-dm-tools». Достоинства: бесплатность, открытый код. Недостатки: требуют технических знаний для установки (Python, настройка локальной LLM), часто заброшены, не имеют удобного интерфейса, плохая документация.

Выводы по анализу предметной области.Проведённый анализ позволяет сделать следующие выводы: Деятельность мастера НРИ характеризуется высокой когнитивной нагрузкой и необходимостью быстрой генерации текстового контента в условиях ограниченного времени.

Существующие программные решения не в полной мере удовлетворяют потребностям мастеров: классические генераторы слишком примитивны, коммерческие LLM-сервисы дороги и плохо локализованы, открытые проекты сложны в использовании.

На рынке отсутствует бесплатный, русскоязычный, контекстно-зависимый ИИ-ассистент с удобным веб-интерфейсом.

Разработка собственной информационной системы является актуальной и экономически целесообразной задачей.

1.2.1Сводный анализ

Таблица 1.1 - Сводный анализ имеющихся аналогов

Критерий	Табличные генераторы	Коммерческие LLM-сервисы	Модули для VTT	Открытые проекты	Наша разработка
Бесплатно	да	нет	нет	да	да
Русский язык	ограниченно	плохо	нет	нет	да
Контекстная Память	нет	частично	нет	нет	да
Удобный интерфейс	да	да	да	нет	да
Не требует установки	да	да	нет	нет	да
Генерация 6+ типов	частично	да	нет	нет	да

Вывод. На рынке отсутствует бесплатный, русскоязычный, контекстно-зависимый ИИ-ассистент с удобным веб-интерфейсом. Существующие аналоги либо дороги, либо плохо локализованы, либо сложны в использовании. Разработка собственной системы является актуальной и экономически целесообразной.

1.3 Обоснование выбора задачи

На основе анализа предметной области были сформулированы требования к разрабатываемой информационной системе «ИИ-ассистент для мастеров настольных ролевых игр» (GM AI Assistant).

Функциональные требования

Система должна обеспечивать выполнение следующих функций[6,20]:

Генерация имён неигровых персонажей (NPC). Мастер указывает расу/культуру (человек, эльф, гном, орк и др.), пол (мужской, женский, нейтральный), количество имён (от 1 до 10). Система возвращает список уникальных имён, соответствующих указанной культуре.

Генерация описаний локаций. Мастер указывает жанр (фэнтези, киберпанк, ужасы), атмосферу (мрачная, светлая, тревожная), ключевые объекты (фонтан, трон, алтарь). Система генерирует связное описание локации объёмом 3–5 предложений.

Генерация диалогов. Мастер задаёт участников диалога (два НПС или НПС и игрок), тему разговора, эмоциональный тон. Система генерирует реплики для каждой стороны.

Генерация сюжетных поворотов. Мастер описывает текущую игровую ситуацию (кратко). Система предлагает 3–5 неожиданных, но логичных поворотов сюжета.

Генерация блоков монстров. Мастер указывает уровень сложности (лёгкий, средний, сложный), тип монстра (животное, гуманоид, нежить и т.д.), особые способности (опционально). Система генерирует статистику: хиты, броня, атаки, характеристики, опыт.

Ответы на вопросы по правилам. Мастер задаёт вопрос на естественном языке (например, «С какой дистанции работает оглушающая атака в D&D 5e?»). Система ищет ответ в базе знаний по правилам и возвращает точную цитату с указанием источника (книга, раздел, страница).

Контекстная память в пределах сессии. Система запоминает сгенерированные сущности (имена NPC, названия локаций) в течение одной игровой сессии и использует их при последующих запросах. Например, если мастер сгенерировал таверну «Кривой Кинжал», а затем просит сгенерировать её владельца, система должна связать эти две сущности.

Сохранение истории генераций. Все запросы и ответы сохраняются в локальном хранилище браузера. Мастер может просматривать предыдущие генерации, копировать их или удалять.

Экспорт результатов. Мастер может скопировать сгенерированный текст в буфер обмена одним нажатием кнопки или сохранить в текстовый файл.

Нефункциональные требования

Производительность: время генерации одного запроса не должно превышать 10 секунд для 90% запросов (при использовании удалённого API OpenAI) и 30 секунд для локальной модели LLaMA 3.

Доступность: система должна работать 24/7 через веб-браузер. Плановое обслуживание – не чаще одного раза в месяц длительностью не более 2 часов.

Кроссплатформенность: интерфейс должен корректно отображаться на настольных компьютерах (Windows, macOS, Linux), планшетах и смартфонах (iOS, Android) – адаптивный дизайн.

Поддерживаемые браузеры: Google Chrome (версии 100 и выше), Mozilla Firefox (версии 110 и выше), Safari (версии 15 и выше), Edge (версии 100 и выше).

Локализация: русскоязычный интерфейс и генерация. Английский язык – опционально (для переключения).

Безопасность: система не должна передавать персональные данные пользователей третьим лицам. При использовании API OpenAI соблюдается политика конфиденциальности OpenAI.

Масштабируемость: архитектура должна позволять увеличивать количество одновременных пользователей до 1000 без изменения кода (горизонтальное масштабирование).

1.3.1Экономическая сущность задачи

С экономической точки зрения, разрабатываемая информационная система направлена на снижение временных затрат мастера НРИ, что в денежном выражении (при переводе на рыночную стоимость услуг профессионального мастера) составляет экономический эффект.

Профессиональный мастер НРИ может брать за проведение игры от 1000 до 5000 рублей за сессию (3–4 часа). Если ИИ-ассистент экономит мастеру 30% времени на подготовку и проведение игры (в основном за счёт ускорения генерации контента), то при 4 сессиях в месяц экономия в денежном

выражении составит от 1200 до 6000 рублей в месяц. При разработке системы с затратами около 50 000 рублей срок окупаемости для профессионального мастера составит от 8 до 40 месяцев. Для любителей экономический эффект выражается в улучшении качества игрового процесса и снижении стресса (нематериальная выгода).

2 Проектная часть

2.1 Архитектура информационной системы

2.1.1 Выбор архитектурного стиля

При проектировании информационной системы «GM AI Assistant» был проведён анализ возможных архитектурных стилей. Рассматривались три основных варианта: десктопное приложение, мобильное приложение и веб-ориентированная архитектура[20].

Десктопное приложение.

Таблица 2.1 - Плюсы и минусы десктоного приложения.

Плюсы	Минусы
Высокая производительность	Требуется установка на каждый компьютер
Работа без интернета (при локальной LLM)	Сложное обновление (пользователь должен скачать новую версию)
Прямой доступ к файловой системе	Привязка к операционной системе

Мобильное приложение (iOS / Android).

Таблица 2.2 - Плюсы и минусы мобильного приложения.

Плюсы	Минусы
Доступность на смартфонах	Ограниченные вычислительные ресурсы
Удобство использования в игре	Требуется публикация в магазинах приложений
	Отдельная разработка под две платформы

Веб-ориентированная архитектура (клиент-сервер).

Таблица 2.3 - Плюсы и минусы веб-приложения.

Плюсы	Минусы
Доступность с любого устройства через браузер	Требуется интернет-соединение (для веб-версии)
Централизованное обновление	Задержки при передаче данных
Кроссплатформенность (Windows, macOS, Linux, планшеты)	Необходимость в серверной инфраструктуре
Не требует установки	

Выбор: на основе анализа принято решение использовать клиент-серверную архитектуру с веб-интерфейсом как наиболее подходящую под требования доступности, кроссплатформенности и простоты обновлений.

2.1.2 Диаграмма развёртывания (UML Deployment)

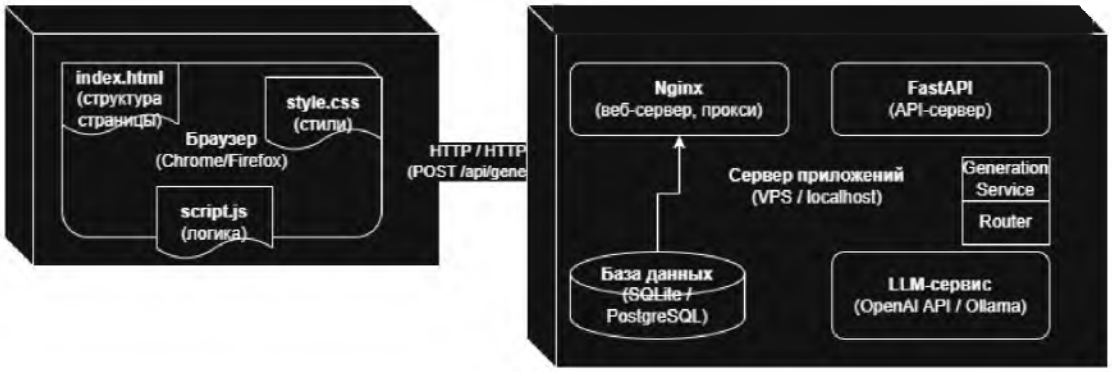

Рисунок 1 - Диаграмма развёртывания (UML Deployment).

Физическое размещение компонентов предполагает два основных сценария:

Сценарий 1: Локальное развёртывание (для демонстрации и разработки). Все компоненты запускаются на одном компьютере: веб-клиент доступен через localhost:5500, API-сервер — через localhost:8000, SQLite — файл на диске.

Сценарий 2: Серверное развёртывание (для публичного доступа). API-сервер и база данных размещаются на VPS (виртуальном приватном сервере) с публичным IP-адресом. Пользователи подключаются через браузер по HTTPS.

2.2 Выбор средств реализации и технологического стека

Выбор языка программирования для бэкенда

Рассматривались языки: Python, Node.js (JavaScript/TypeScript), Go, C# (.NET Core), Java (Spring)[3,22].

Python 3.12 — выбран в качестве основного языка бэкенда.

Таблица 2.4 – Критерии выбора python 3.12.

Критерий	Оценка
Скорость разработки	Очень высокая (лаконичный синтаксис, динамическая типизация)
Экосистема для работы с LLM	Лучшая в индустрии (OpenAI, Ollama, LangChain, Transformers)
Асинхронность	Поддерживается (asyncio, FastAPI)
Производительность	Ниже, чем у Go или Java, но достаточна для проекта
Порог входа	Низкий

Почему не Node.js: экосистема для LLM менее зрелая, требует дополнительных обёрток.

Почему не C#: более высокая сложность разработки и настройки окружения.

Выбор веб-фреймворка

Рассматривались: Flask (микрофреймворк), Django (монолитный), FastAPI (асинхронный).

FastAPI — выбран в качестве основного фреймворка.

Таблица 2.5 – Плюсы и минусы выбора FastAPI.

Плюсы	Минусы
Высокая производительность (на уровне Node.js и Go)	Относительно молодой (меньше готовых решений)
Встроенная асинхронность (async/await)	
Автоматическая валидация данных (Pydantic)	
Автоматическая генерация OpenAPI-документации (Swagger)	
Встроенная поддержка WebSocket и Server-Sent Events	

Почему не Flask: синхронный по умолчанию, для долгих запросов к LLM требуются дополнительные танцы с бубном.

Почему не Django: слишком тяжёлый, включает ORM, админку и многое другое, что не требуется для API-сервера.

Выбор базы данных

Рассматривались: PostgreSQL, MySQL, SQLite, MongoDB.

Таблица 2.6 - Выборка баз данных.

Критерий	SQLite	PostgreSQL
Установка	Не требуется (встроенная в Python)	Требуется установка сервера
Конфигурация	Нулевая	Требуется настройка пользователей и прав
Производительность при малых нагрузках	Высокая	Высокая
Поддержка JSON	Есть (с 3.9)	Отличная
Масштабируемость	Только однопользовательская	Поддерживает тысячи соединений
Транзакции	Поддерживаются	Полноценные

Для разработки и локальной версии - SQLite. Для серверной версии - PostgreSQL.

Вывод: на этапе разработки и для демонстрации используется SQLite (один файл .db). При необходимости перехода на серверную версию миграция на PostgreSQL выполняется без изменения кода благодаря SQLAlchemy ORM.

Выбор технологии кеширования и управления сессиями

Redis — выбран для кеширования ответов и хранения контекста сессий.

Таблица 2.7 - Плюсы и минусы выбора Redis.

Плюсы	Минусы
Высокая производительность (оперативная память)	Требует установки отдельного сервиса
Встроенная поддержка TTL (автоматическое удаление устаревших записей)	Данные не сохраняются при перезагрузке (если не настроено сохранение)
Поддержка различных структур данных (строки, хеши, списки)	
Простота интеграции с Python (библиотека redis-py)	

Альтернатива без Redis: в упрощённой версии контекст хранится в оперативной памяти Python (глобальный словарь). Это допустимо для демонстрации, но не для продакшена.

Выбор клиентского стека (фронтенд)

Исходил выбор среди : React, Vanilla JS.[4,18]

В конечном итоге был выбран для Vanilla JS как программа работы с фронтендом

Таблица 2.8 - Плюсы и минусы Vanilla JS.

Плюсы	Минусы
Не требует сборки (Webpack, Vite)	При сложном интерфейсе код может стать запутанным
Минимальный размер (нет тяжёлых фреймворков)	Нет реактивности «из коробки»
Работает в любом браузере без дополнительных зависимостей	
Простота отладки (инструменты разработчика браузера)	
Низкий порог входа	

React не был выбран потому что он избыточен для 5-6 экранов, требует настройки сборки, увеличивает размер бандла.

Выбор технологии работы с большими языковыми моделями (LLM)

Рассматривались два подхода: использование облачного API (OpenAI) и использование локальной модели (Ollama + LLaMA 3).

Таблица 2.9 - Плюсы и минусы OpenAI API.

Плюсы	Минусы
Высокое качество генерации	Требует оплаты (около $0,001 за запрос)
Высокая скорость ответа (1-3 секунды)	Требует интернет-соединения
Не требует мощного оборудования на стороне пользователя	Данные отправляются на сторонний сервер
Простая интеграция (официальная библиотека)	

Таблица 2.10 - Плюсы и минусы Ollama + LLaMA 3(локальная модель).

Плюсы	Минусы
Бесплатно	Требует мощного ПК (8+ ГБ видеопамяти или 16+ ГБ ОЗУ)
Полная конфиденциальность (данные не покидают компьютер)	Медленнее (5-15 секунд на запрос)
Работа без интернета	Качество генерации ниже, чем у GPT-4, но сравнимо с GPT-3.5

Выбор: в проекте реализован гибридный подход. По умолчанию используется мок-режим (заглушки) для демонстрации интерфейса. При необходимости легко подключается OpenAI API (через переменную окружения) или локальная Ollama .

Выбор библиотек и вспомогательных инструментов было основано исключительно для демонстрации и не подходит для полноценной работы

Таблица 2.11- Выбор библиотек и вспомогательных инструментов.

Библиотека	Назначение	Плюсы	Минусы
python-dotenv	Загрузка переменных из .env	Простота, стандарт де-факто	Только для разработки
uvicorn	ASGI-сервер для FastAPI	Высокая производительность, автоперезагрузка	-
sqlalchemy	ORM для работы с БД	Абстракция от конкретной СУБД	Небольшой порог входа
httpx	Асинхронные HTTP-запросы	Совместимость с async/await	-
jinja2	Шаблонизатор для промптов	Гибкость, наследование шаблонов	-

2.3 Моделирование бизнес-процессов (AS-IS и TO-BE)

Диаграмма потоков данных существующего процесса (DFD AS-IS)[20]

На рисунке 2.1 представлена диаграмма потоков данных (DFD) для существующего процесса работы мастера НРИ.

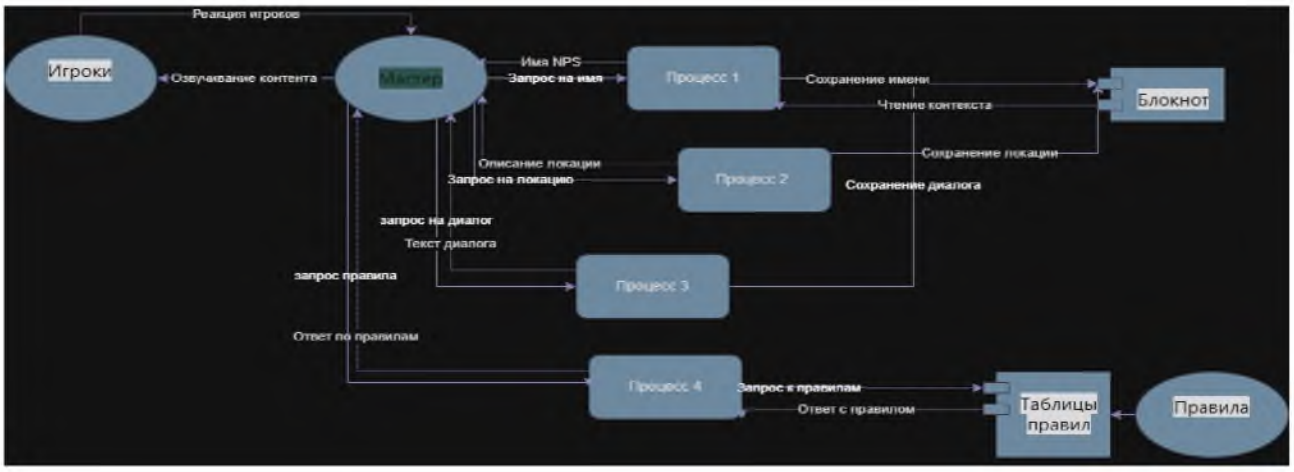

Рисунок 2.1- диаграмма потоков данных (DFD)

Основные внешние сущности: «Мастер», «Игроки», «Правила». Основные хранилища данных: «Блокнот мастера», «Таблицы правил». Основные процессы: «Сгенерировать имя», «Сгенерировать локацию», «Сгенерировать диалог», «Найти правило».

Моделирование целевого процесса в нотации BPMN (TO-BE)

На рисунке 2.2 представлена PMN-диаграмма целевого процесса работы мастера с ИИ-ассистентом.

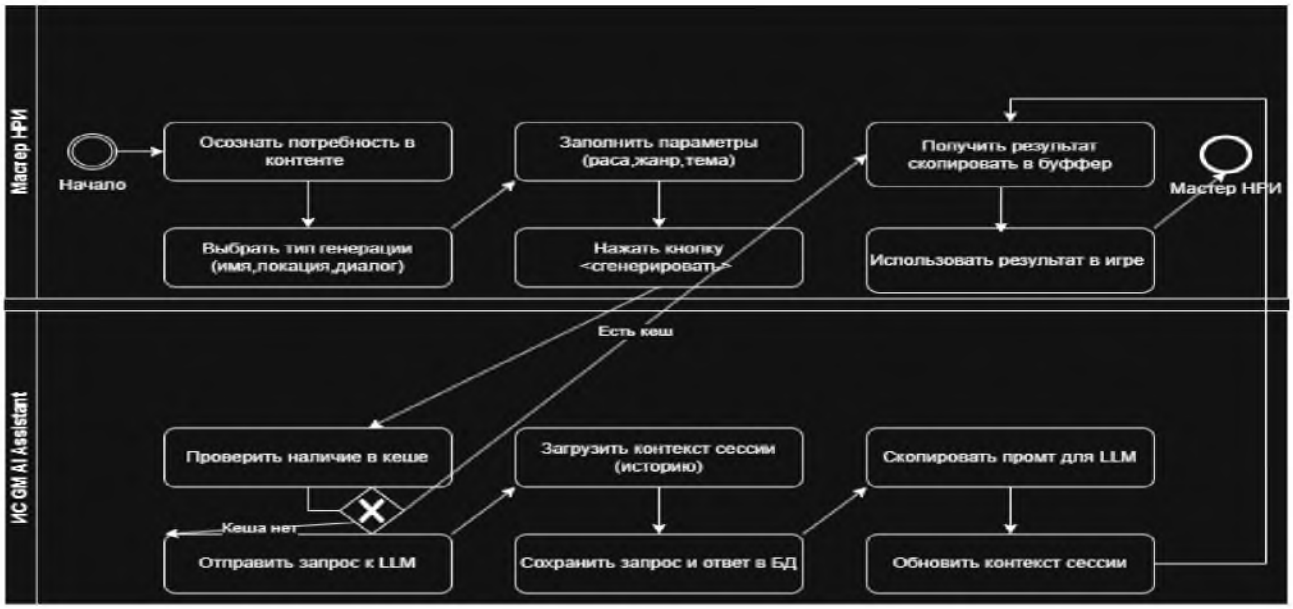

Рисунок 2.2 - PMN-диаграмма целевого процесса работы мастера с ИИ-ассистентом

Описание процесса в виде текста:

Старт — мастер понимает, что ему нужен новый игровой контент.

Мастер выбирает тип генерации в интерфейсе системы (пользовательская задача). Мастер заполняет параметры (например, для локации: жанр, атмосфера, ключевые объекты). Мастер нажимает «Сгенерировать».

Система проверяет кеш — если аналогичный запрос был недавно, берёт ответ из кеша.

Если кеша нет, система загружает контекст сессии (предыдущие генерации).

Система формирует промпт и отправляет запрос к LLM.

LLM генерирует ответ и возвращает его системе.

Система сохраняет ответ в историю и обновляет контекст.

Система отображает результат мастеру.

Мастер копирует результат в буфер обмена.

Мастер использует результат в игре.

Конец.

Сравнение AS-IS и TO-BE:

Таблица 2.12- Сравнение AS-IS и TO-BE:

Параметр	AS-IS	TO-BE
Время генерации	30–90 секунд	2–10 секунд
Количество переключений	3–5	0
Контекстная память	Отсутствует	Есть
Ручное копирование	Да	Нет (одна кнопка)

2.4 Диаграмма прецедентов (UML Use Case)

Для формализации требований к разрабатываемой информационной системе и описания функциональных возможностей с точки зрения пользователя была построена диаграмма прецедентов (UML Use Case Diagram), представленная на рисунке Данная диаграмма позволяет наглядно представить,

какие действия может выполнять пользователь в системе, а также определить границы системы и её взаимодействие с внешними сущностями.

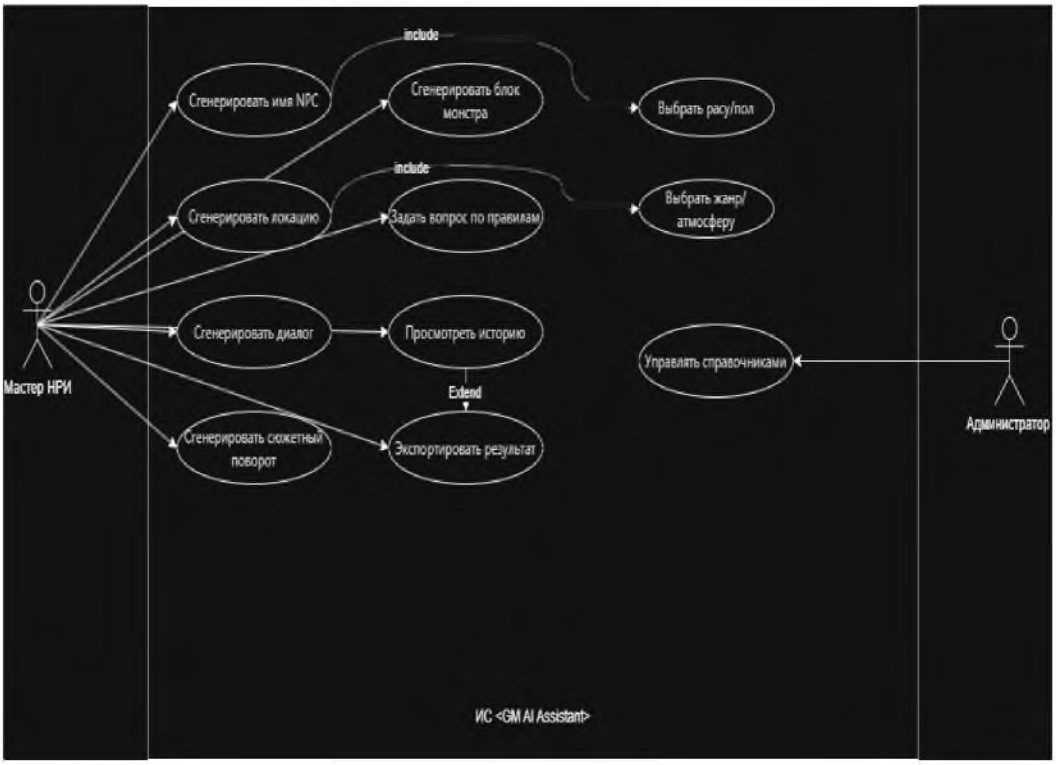

Рисунок 2.3 – Диаграмма прецедентов (UML Use Case)

На диаграмме выделены два основных актёра (пользователя системы):

1. Мастер НРИ — основной пользователь системы, который непосредственно взаимодействует с ней в процессе игровой сессии. Мастер инициирует генерацию контента, заполняет параметры и получает результаты.
2. Администратор — вспомогательный пользователь, отвечающий за наполнение и поддержку справочной информации системы (списки рас, жанров, игровых систем и т.д.).

Границы системы обозначены прямоугольником с заголовком «ИС «GM AI Assistant». Внутри границ расположены все варианты использования (прецеденты), которые система должна поддерживать. Вне границ находятся актёры, взаимодействующие с системой.

Варианты использования для мастера НРИ

Для основного пользователя системы выделены следующие восемь ключевых прецедентов:

1. «Сгенерировать имя NPC». Данный прецедент позволяет мастеру получить список уникальных имён для неигровых персонажей. Мастер указывает расу, пол и желаемое количество имён. Система возвращает список имён, соответствующих заданным параметрам. Данный прецедент обязательно включает в себя прецедент «Выбрать расу/пол» (связь <<include>>), поскольку выбор расы и пола является обязательным условием для генерации имён.
2. «Сгенерировать локацию». Прецедент обеспечивает создание текстового описания игровой локации. Мастер указывает жанр, атмосферу и ключевые объекты (опционально). Система генерирует связное описание, которое мастер может использовать для погружения игроков в игровой мир. Данный прецедент обязательно включает прецедент «Выбрать жанр/атмосферу», так как эти параметры являются основой для генерации.
3. «Сгенерировать диалог». Позволяет мастеру создать диалог между двумя или тремя персонажами. Мастер указывает участников, тему и эмоциональный тон разговора. Система возвращает реплики для каждого участника в соответствии с заданными параметрами.
4. «Сгенерировать сюжетный поворот». Прецедент предназначен для создания неожиданных событий, которые мастер может использовать для поддержания интриги и динамики игры. Мастер описывает текущую игровую ситуацию, а система предлагает 3–5 вариантов неожиданного развития событий.
5. «Сгенерировать блок монстра». Обеспечивает создание статистики для игровых противников. Мастер указывает уровень сложности и тип монстра. Система генерирует блок с характеристиками: хиты, класс брони, атаки, характеристики, опыт.
6. «Задать вопрос по правилам». Позволяет мастеру получить точный ответ на вопрос, касающийся игровых правил. Мастер вводит вопрос и выбирает игровую систему. Система ищет ответ в базе знаний и возвращает цитату из правил с указанием источника.

7. «Просмотреть историю». Даёт мастеру возможность просматривать ранее сгенерированные результаты. История хранится в локальном хранилище браузера и содержит не менее 20 последних записей. Мастер может кликнуть на элемент истории, чтобы повторно отобразить результат.

8. «Экспортировать результат». Позволяет мастеру скопировать сгенерированный текст в буфер обмена одним нажатием кнопки. Данный прецедент расширяет прецедент «Просмотреть историю» (связь <<extend>>), поскольку возможность экспорта доступна как для текущего результата, так и для любого элемента из истории.

Вариант использования для администратора

9. «Управлять справочниками». Данный прецедент доступен только администратору системы. Он позволяет добавлять, редактировать и удалять справочные данные: расы для генерации имён, жанры для генерации локаций, игровые системы для поиска правил.

Связи между прецедентами

На диаграмме используются два типа связей между прецедентами:

Include (обязательное включение) - показывает, что один прецедент обязательно включает в себя другой. Например, генерация имени NPC невозможна без выбора расы и пола, поэтому прецедент «Сгенерировать имя NPC» включает прецедент «Выбрать расу/пол». Аналогично, генерация локации включает выбор жанра и атмосферы.

extend (расширение) - показывает, что один прецедент расширяет функциональность другого, но не является обязательным. Просмотр истории может быть расширен возможностью экспорта результата, поэтому прецедент «Просмотреть историю» расширяется прецедентом «Экспортировать результат».

2.5 Диаграмма активности (UML Activity Diagram)

Для детального описания логики выполнения процесса генерации контента и визуализации последовательности действий пользователя и системы

была разработана диаграмма активности (UML Activity Diagram), представленная на рисунке 2.4. Данная диаграмма дополняет диаграмму прецедентов, показывая не только что делает система, но и в каком порядке выполняются действия, как происходит взаимодействие между мастером и системой, а также какие ветвления предусмотрены в логике работы.

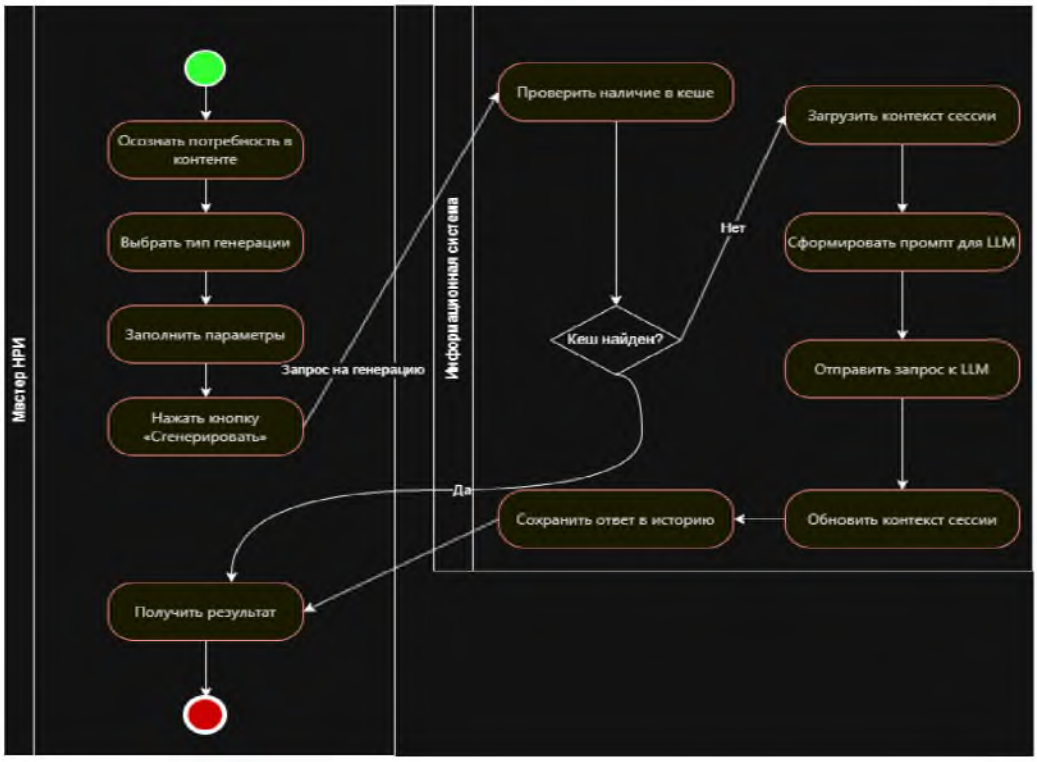

Рисунок 2.4 – Диаграмма активности (UML Activity Diagram)

Диаграмма активности организована с использованием двух дорожек (swimlanes), которые разделяют ответственность между участниками процесса:

Дорожка «Мастер НРИ» — содержит действия, выполняемые пользователем.

Дорожка «Информационная система» — содержит действия, выполняемые программным обеспечением.

Такое разделение позволяет чётко видеть, какие операции выполняются человеком, а какие — автоматически системой, что особенно важно при проектировании пользовательских интерфейсов и определении точек взаимодействия.

Описание процесса (последовательность действий)

Процесс начинается с стартового узла, обозначающего начало выполнения сценария. Далее следует последовательность действий, которая может быть разделена на два этапа: действия мастера и ответные действия системы.

Этап 1. Действия мастера (подготовка запроса):

1. «Осознать потребность в контенте». На данном этапе мастер в ходе игровой сессии понимает, что ему требуется новый игровой контент (например, игроки неожиданно зашли в лавку, и нужен торговец). Это действие не является техническим, но включено в диаграмму для полноты описания бизнес-процесса.
2. «Выбрать тип генерации». Мастер выбирает один из шести доступных типов генерации: имя NPC, локация, диалог, сюжетный поворот, блок монстра или вопрос по правилам. Выбор осуществляется через интерфейс системы (кнопки в левой панели). После выбора типа система динамически изменяет форму для ввода параметров.
3. «Заполнить параметры». Мастер вводит необходимые данные в соответствии с выбранным типом генерации. Для имени NPC это раса, пол и количество имён. Для локации — жанр, атмосфера и ключевые объекты. Для диалога — участники, тема и тон разговора. Для вопроса по правилам — игровая система и текст вопроса.
4. «Нажать кнопку "Сгенерировать"». Мастер завершает подготовку запроса и инициирует его отправку в систему. Нажатие кнопки является точкой перехода от действий пользователя к автоматическим действиям системы.

На этом этапе происходит передача управления от мастера к информационной системе — линия потока пересекает границу между дорожками и направляется к первому действию в дорожке системы.

Этап 2. Действия информационной системы (обработка запроса):

5. «Проверить наличие в кеше». Система вычисляет хеш запроса (SHA-256 от нормализованного JSON-представления параметров) и проверяет, имеется ли

в кеше сохранённый ответ для данного запроса. Кеш позволяет ускорить обработку повторяющихся запросов и снизить нагрузку на языковую модель.

6. Узел решения «Кеш найден?». На данном этапе происходит ветвление процесса. Если кеш содержит ответ для данного запроса, система переходит к этапу возврата результата (переход по ветке «[есть]»). Если кеш не найден, система продолжает полную обработку запроса (переход по ветке «[нет]»).

Ветка «Кеш найден» (быстрый путь):

7. Если ответ найден в кеше, система пропускает все последующие этапы обработки и немедленно возвращает кешированный результат мастеру. Это позволяет сократить время ответа до долей секунды для повторяющихся запросов.

Ветка «Кеш не найден» (полный путь):

8. «Загрузить контекст сессии». Система извлекает контекст текущей сессии — список ранее сгенерированных имён NPC и названий локаций. Контекст хранится в оперативной памяти (или в Redis в продакшен-версии) и позволяет обеспечивать связность генераций. Например, если мастер уже создал NPC по имени «Торгир», система не предложит это имя повторно.

9. «Сформировать промпт для LLM». На основе выбранного типа генерации, введённых пользователем параметров и загруженного контекста система формирует текстовый запрос (промпт) для языковой модели. Промпт включает системную инструкцию («Ты — помощник мастера НРИ»), контекст сессии и пользовательский запрос. Формирование промпта выполняется с использованием шаблонов Jinja2.

10. «Отправить запрос к LLM». Система передаёт сформированный промпт языковой модели. В зависимости от конфигурации (переменная USE_LOCAL_LLM в файле .env), запрос направляется к облачному API OpenAI (модель gpt-3.5-turbo) или к локальной модели LLaMA 3, запущенной через Ollama. Время выполнения данного этапа составляет от 2 до 15 секунд в зависимости от выбранного способа.

11. «Сохранить ответ в историю». После получения ответа от LLM система сохраняет запрос и полученный ответ в базу данных (SQLite для локальной версии или PostgreSQL для серверной). Сохраняются: тип генерации, параметры запроса, сгенерированный текст, временная метка и время обработки. История позволяет мастеру просматривать предыдущие генерации и использовать их повторно.
12. «Обновить контекст сессии». Система извлекает из сгенерированного ответа ключевые сущности (имена NPC, названия локаций) и добавляет их в контекст сессии. Например, если была сгенерирована локация «Забытый Склеп», это название сохраняется в контексте и может быть использовано при следующем запросе (например, «сгенерируй владельца Забытого Склепа»).
13. Этап 3. Возврат результата мастеру:
14. «Получить результат». Мастер получает сгенерированный текст в области вывода интерфейса. Результат отображается в правой панели и может быть скопирован в буфер обмена.
15. «Конечный узел». Процесс завершается. Мастер может либо начать новый запрос, вернувшись к шагу 1, либо завершить работу с системой.

2.6 Требования к ИС

В ходе разработки были сформулированы следующие требования к информационной системе GM AI Assistant [20].

Требования разделены на две группы: функциональные (что система должна делать) и нефункциональные (как система должна это делать).

2.6.1 Функциональные требования

Функциональные требования определяют перечень конкретных задач, которые система должна решать, и бизнес-процессов, которые она должна автоматизировать.

Таблица 2.12- Требования к управлению сессией и контекстом

Наименование требования	Описание
Создание сессии	При первом обращении пользователя система должна автоматически создавать новую сессию с уникальным идентификатором (UUID) и сохранять её в течение 8 часов
Хранение контекста сессии	Система должна сохранять в контексте сессии до 10 последних сгенерированных имён NPC и до 5 последних локаций
Использование контекста при генерации	При формировании промпта для LLM система должна включать в него информацию из контекста текущей сессии (список существующих NPC и локаций) для обеспечения связности повествования
Сохранение истории генераций	Система должна сохранять историю всех запросов и ответов в базе данных с возможностью просмотра последних 20 записей

Таблица 2.13- Требования к генерации имён NPC

Наименование требования	Описание
Выбор расы	Система должна предоставлять возможность выбора расы из списка: человек, эльф, дворф, орк, полурослик, гном
Выбор пола	Система должна предоставлять возможность выбора пола: мужской, женский, нейтральный
Количество имён	Система должна позволять указать количество генерируемых имён от 1 до 10
Уникальность имён	Сгенерированные имена не должны повторяться в пределах одной сессии (проверка по контексту)
Соответствие культуре	Имена должны соответствовать выбранной расе (эльфийские имена должны звучать как эльфийские, дворфийские — как дворфийские)

Продолжение таблицы 2.13

Формат вывода	Результат должен возвращаться в виде списка имён в формате JSON с возможностью копирования

Таблица 2.14- Требования к генерации локаций.

Наименование требования	Описание
Выбор жанра	Система должна предоставлять выбор жанра: фэнтези, киберпанк, ужасы, стимпанк
Указание атмосферы	Система должна позволять пользователю вводить описание атмосферы (текстовое поле)
Ключевые объекты	Система должна позволять пользователю вводить ключевые объекты локации (текстовое поле, опционально)
Связность с контекстом	При генерации локации система должна учитывать ранее созданные локации (из контекста) и обеспечивать логическую связь
Сенсорные детали	Сгенерированное описание должно включать не менее 3 сенсорных деталей (зрение, слух, обоняние, осязание)
Объём описания	Описание локации должно содержать 3–6 предложений (50–150 слов)

Таблица 2.15- Требования к генерации диалогов

Наименование требования	Описание
Участники диалога	Система должна позволять указать до 3 участников диалога (имена персонажей)
Тема разговора	Система должна позволять вводить тему разговора (текстовое поле)
Эмоциональный тон	Система должна позволять указывать эмоциональный тон диалога (например, «нейтральный», «напряжённый», «весёлый»)
Формат диалога	Реплики должны быть размечены по участникам в формате: «Имя: текст реплики»
Длина диалога	Диалог должен содержать от 5 до 10 реплик

Продолжение таблицы 2.15

Использование контекста	При генерации диалога система должна учитывать существующих NPC из контекста сессии

Таблица 2.16- Требования к генерации сюжетных поворотов

Наименование требования	Описание
Ввод ситуации	Система должна позволять пользователю вводить текущую игровую ситуацию (текст до 500 символов)
Количество вариантов	Система должна генерировать от 3 до 5 вариантов сюжетных поворотов
Логическая связь	Каждый поворот должен логически вытекать из введённой ситуации
Разная сложность	Варианты должны включать разную степень сложности (лёгкий, средний, сложный)
Отсутствие клише	Генерируемые повороты не должны использовать шаблонные клише («это был сон», «ты умер и начал заново»)

Таблица 2.17- Требования к генерации блоков монстров

Наименование требования	Описание
Уровень сложности	Система должна предоставлять выбор уровня сложности: лёгкий, средний, сложный, смертельный
Тип монстра	Система должна позволять вводить тип монстра (текстовое поле)
Особые способности	Система должна позволять вводить особые способности монстра (опционально)
Структура блока	Блок монстра должен содержать: название, хиты (HP), класс брони (AC), характеристики (6 атрибутов), атаки, опыт (XP)
Соответствие игровой системе	Значения характеристик и параметров должны соответствовать выбранной игровой системе (D&D 5e, Pathfinder 2e)

Таблица 2.18- Требования к ответам на вопросы по правилам (RAG)

Наименование требования	Описание
Выбор игровой системы	Система должна предоставлять выбор игровой системы: D&D 5e, Pathfinder 2e
Ввод вопроса	Система должна позволять пользователю вводить вопрос по правилам (текст до 200 символов)
Поиск по базе знаний	Система должна выполнять поиск релевантных фрагментов правил в базе знаний с использованием векторного поиска (RAG)
Точность ответа	Ответ должен основываться строго на найденных фрагментах правил, без додумывания
Указание источника	В ответе должны быть указаны источники (книга, раздел, страница) для каждого утверждения

Таблица 2.19- Требования к экспорту и истории

Наименование требования	Описание
Копирование результата	Система должна предоставлять возможность копирования сгенерированного текста в буфер обмена одним нажатием
Просмотр истории	Система должна предоставлять возможность просмотра истории последних 20 генераций с фильтрацией по типу
Удаление из истории	Пользователь должен иметь возможность удалять отдельные записи из истории

2.6.2 Нефункциональные требования

Нефункциональные требования определяют критерии качества системы, ограничения и условия эксплуатации[6].

Таблица 2.20 - Требования к производительности

Наименование требования	Значение
Время отклика API	Не более 5 секунд для 90% запросов (при использовании OpenAI API)
Время отклика локальной LLM	Не более 15 секунд для 90% запросов (при использовании LLaMA 3)
Время загрузки интерфейса	Не более 3 секунд при скорости интернета 10 Мбит/с
Одновременные пользователи	Поддержка не менее 10 одновременных пользователей на одном сервере
Максимальный размер запроса	Не более 1 МБ для текстовых данных

Таблица 2.21 - Требования к надёжности

Наименование требования	Значение
Доступность системы	Не менее 99% времени (простой не более 7,2 часов в месяц)
Обработка ошибок	При ошибке LLM система должна выполнять повторный запрос (до 2 раз)
Сохранность данных	При перезапуске сервера история генераций не должна теряться
Восстановление после сбоя	Система должна автоматически перезапускаться при критических ошибках
Автоматическое сохранение	Контекст сессии должен автоматически сохраняться каждые 5 минут

2.6.3 Требования к техническому обеспечению

Таблица 2.22 - Серверная часть (минимальные требования)

Компонент	Требование
Операционная система	Linux Ubuntu 22.04 / Windows Server / macOS
Процессор	2 vCPU (виртуальных ядра)
Оперативная память	4 ГБ (рекомендуется 8 ГБ при локальной LLM)
Дисковое пространство	40 ГБ SSD
Сетевое подключение	100 Мбит/с

Таблица 2.23 - Клиентская часть (минимальные требования)

Компонент	Требование
Операционная система	Windows 7/10/11, macOS 10.15+, Linux, iOS 13+, Android 8+
Браузер	Chrome 100+, Firefox 110+, Safari 15+, Edge 100+
Интернет	От 1 Мбит/с (для веб-версии); не требуется для локальной версии

Таблица 2.24 - Требования к программному обеспечения

Наименование требования	Значение
Язык бэкенда	Python версии 3.12 или выше
Веб-фреймворк	FastAPI версии 0.115.0 или выше
База данных (локальная)	SQLite 3.35+
База данных (серверная)	PostgreSQL 14+
Фронтенд	HTML5, CSS3, JavaScript ES6+
LLM (веб-версия)	OpenAI API (GPT-3.5-turbo или GPT-4)
LLM (локальная)	Ollama + LLaMA 3 8B

Таблица 2.25 - Требования к совместимости

Наименование требования	Значение
Кроссплатформенность	Система должна работать на Windows, macOS, Linux через браузер
Совместимость браузеров	Поддержка Chrome, Firefox, Safari, Edge (последние 2 версии)
Формат данных	Взаимодействие через REST API с обменом в формате JSON
Миграция БД	Возможность перехода с SQLite на PostgreSQL без изменения кода (SQLAlchemy)

3 Разработка программы

3.1 Структура базы данных

Для реализации информационной системы используется СУБД SQLite (в локальной версии) или PostgreSQL (в серверной версии). Схема базы данных представлена в Приложении А (физическая ER-диаграмма). Ниже приведено описание основных таблиц.

Таблица 3.1 – Структура таблицы users

Имя поля	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Уникальный идентификатор сессии
user_id	INTEGER	FOREIGN KEY REFERENCES users(id)	Идентификатор пользователя
session_token	VARCHAR(255)	NOT NULL, UNIQUE	Токен сессии (для API)
context	TEXT (JSON)	NULL	Контекст сессии (сгенерированные сущности)
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Дата создания сессии
expires_at	TIMESTAMP	NOT NULL	Дата истечения сессии

Таблица 3.2 – Структура таблицы sessions

Имя поля	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Уникальный идентификатор запроса
session_id	INTEGER	FOREIGN KEY REFERENCES sessions(id)	Идентификатор сессии
generation_type	VARCHAR(20)	NOT NULL	Тип генерации (name, location, dialogue, plot, monster, rule)

Продолжение таблицы 3.2

parameters	TEXT (JSON)	NOT NULL	Параметры запроса в формате JSON
timestamp	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Время запроса

Таблица 3.3 – Структура таблицы generation_requests

Имя поля	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Уникальный идентификатор ответа
request_id	INTEGER	FOREIGN KEY REFERENCES generation_requests(id)	Идентификатор запроса
generated_text	TEXT	NOT NULL	Сгенерированный текст (или JSON)
status	VARCHAR(20)	DEFAULT 'success'	Статус: success, error, retry
processing_time_ms	INTEGER	NOT NULL	Время обработки в миллисекундах
model_used	VARCHAR(50)	NOT NULL	Использованная модель (gpt-3.5-turbo, llama3 и т.д.)

Таблица 3.4 – Структура таблицы generation_responses

Имя поля	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Уникальный идентификатор правила
game_system	VARCHAR(50)	NOT NULL	Игровая система (D&D 5e, Pathfinder 2e)
section	VARCHAR(100)	NOT NULL	Раздел правил (бои, магия, навыки)
rule_text	TEXT	NOT NULL	Текст правила
source	VARCHAR(200)	NOT NULL	Источник (книга, страница)

Продолжение таблицы 3.4

embedding	BLOB	NULL	Векторное представление текста (для поиска)

Таблица 3.5 – Структура таблицы knowledge_base

Имя поля	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Уникальный идентификатор кеша
request_hash	VARCHAR(64)	NOT NULL, UNIQUE	SHA-256 хеш запроса
response_text	TEXT	NOT NULL	Кешированный ответ
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Время создания
expires_at	TIMESTAMP	NOT NULL	Время истечения (через 1 час)

3.1.1Классификаторы и справочники

Для обеспечения единообразия данных используются следующие классификаторы[1]:

Типы.генерации: name (имя), location (локация), dialogue (диалог), plot (с южетный поворот), monster (блок монстра), rule (правило).

Расы.имён: human (человек), elf (эльф), dwarf (дворф), orc (орк), halfling (п олуросли к), gnome (гном), dragonborn (драконорождённый), tiefling (тифлинг).

Жанры.локаций: fantasy (фэнтези), scifi (киберпанк), horror (ужасы), steam punk (стимпанк).

Игровые системы.

dnd5e (Dungeons & Dragons 5-й редакции), pf2e (Pathfinder 2-й редакции), gurps (GURPS).

Уровни сложности.

монстров: easy (лёгкий), medium (средний), hard (сложный), deadly (смертельй).

Форматы входных и выходных сообщений.

Все взаимодействие между клиентом и сервером осуществляется в формате JSON.

Пример входного сообщения для генерации локации представлен в Приложении 1.

При каждом новом запросе сервер извлекает контекст текущей сессии, добавляет его в промпт для LLM (в виде краткой сводки) и затем сохраняет обновлённый контекст.

База знаний по правилам (RAG).

Для ответов на вопросы по правилам используется подход RAG (Retrieval-Augmented Generation). База знаний содержит тексты правил из официальных книг по D&D 5e, Pathfinder 2e и другим системам. При поступлении вопроса:

Вопрос преобразуется в векторное представление (embedding) с помощью модели all-MiniLM-L6-v2.

Выполняется поиск 3 наиболее релевантных фрагментов в базе знаний (по косинусной близости).

Найденные фрагменты вместе с вопросом передаются LLM с инструкцией: «Ответь на вопрос, используя только предоставленные фрагменты правил. Если ответа нет в фрагментах, скажи "Не знаю"».

Ответ возвращается пользователю вместе со ссылками на источники.

3.2 Описание программных модулей

Модуль 1: main.py – точка входа приложения FastAPI

Этот модуль инициализирует приложение FastAPI, подключает роутеры, настраивает CORS (для доступа с других доменов), запускает сервер через Uvicorn. Листинг основных функций представлен в Приложении 3-8.

Модуль 2: routers/generate.py – обработка запросов на генерацию

Содержит эндпоинт POST /api/generate. Валидирует запрос, вызывает сервис генерации, возвращает ответ.

Модуль 3: services/generation_service.py – основная бизнес-логика

Содержит класс GenerationService с методом generate(), который реализует алгоритм, описанный в разделе 2.2.2 (блок-схема). Включает в себя вызов кеша, формирование промпта, вызов LLM, парсинг, сохранение в БД и обновление контекста.

Модуль 4: llm_client.py – абстракция для работы с LLM.

Реализует два класса: OpenAIClient и OllamaClient, наследующих от абстрактного класса BaseLLMClient. Это позволяет легко переключаться между моделями.

Модуль 5: prompt_builder.py – формирование промптов

Содержит шаблоны для каждого типа генерации (используется Jinja2). Пример шаблона для имён:

Модуль 6: context_manager.py – управление контекстом сессии

Использует Redis для хранения JSON-объекта контекста. Ключ: session:{session_id}:context, TTL: 8 часов.

Модуль 7: rag_engine.py – поиск по базе знаний (RAG)

Использует FAISS для векторизации и поиска правил. Загружает эмбеддинги при старте приложения.

Модуль 8: Клиентская часть (frontend).

index.html – структура страницы (форма выбора типа генерации, поля ввода, область вывода).

style.css – адаптивный дизайн (светлая/тёмная тема, отступы, анимации).

script.js – обработка событий, отправка fetch-запросов, отображение результатов, сохранение истории в localStorage.

3.3 Руководство пользователя

3.3.1 Описание интерфейса

Интерфейс информационной системы «GM AI Assistant» выполнен в стиле «минимализм» для снижения когнитивной нагрузки на мастера в процессе игры. Все элементы управления сгруппированы на одной странице, что позволяет не переключаться между вкладками.

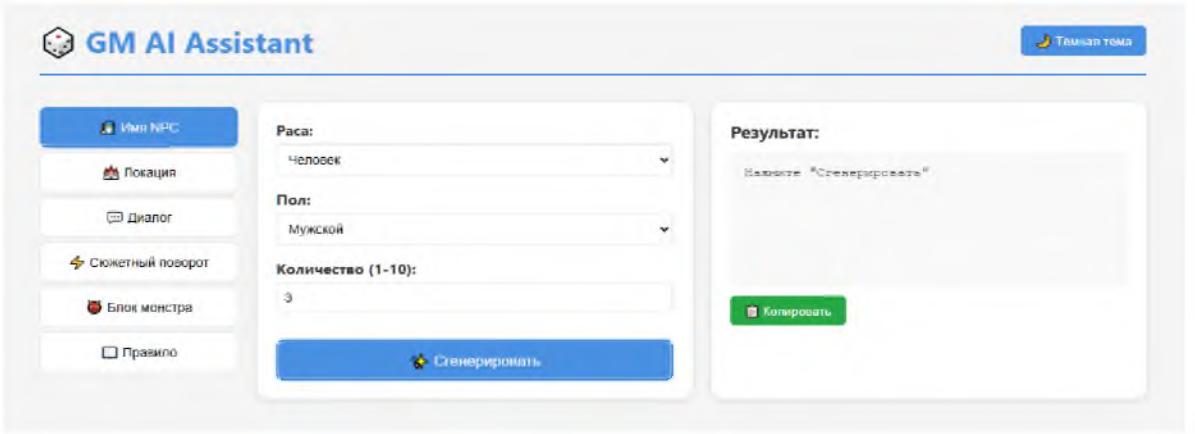

Рисунок 3.1 - Интерфейс информационной системы «GM AI Assistant»

Главное окно содержит следующие зоны:

Верхняя панель (header). Логотип системы (название), кнопка «Сменить тему» (светлая/тёмная), индикатор статуса подключения к серверу (зелёный/красный).

Левая панель – выбор типа генерации. Шесть крупных кнопок с иконками и подписями:

«Имя NPC»

«Локация»

«Диалог»

«Сюжетный поворот»

«Блок монстра»

«Правило»

Центральная область – форма параметров. Динамически изменяется в зависимости от выбранного типа генерации. Для «Имени NPC»: выпадающий список расы, выбор пола (мужской/женский/нейтральный), ползунок количества имён (1–10). Для «Локации»: поле ввода жанра (текст), поле ввода атмосферы (текст), поле ввода ключевых объектов (текст, опционально). Для «Правила»: выпадающий список игровой системы, текстовое поле вопроса.

Кнопка «Сгенерировать» (синяя, крупная). После нажатия появляется индикатор загрузки (спиннер).

Область вывода результата (правая панель). Отображает сгенерированный текст в читаемом формате. Под результатом – кнопки «Копировать в буфер», «Сгенерировать заново», «Сохранить в историю».

Нижняя панель – история генераций (сворачиваемая). Список последних 20 запросов и ответов с возможностью удаления и повторного использования.

Адаптивный дизайн для мобильных устройств:

При ширине экрана менее 768px левая панель с типами генерации превращается в выпадающий список (select) в верхней части экрана.

Кнопка «Сгенерировать» растягивается на всю ширину.

Правая панель с результатом располагается под формой.

3.3.2 Порядок работы

Регистрация и начало сессии. Система не требует обязательной регистрации. При первом обращении автоматически создаётся анонимная сессия (cookie session_id), которая живёт 8 часов. Зарегистрированные пользователи могут сохранять историю между сессиями (опционально). Регистрация выполняется по кнопке «Войти» (верхняя панель) – требуется указать email и пароль.

Генерация имени NPC. Нажать кнопку «Имя NPC» в левой панели.

Выбрать расу из выпадающего списка (человек, эльф, дворф, орк, полурослик, гном, драконорождённый, тифлинг).

Выбрать пол.

Установить количество имён (ползунком или вводом числа).

Нажать «Сгенерировать».

Через 2–5 секунд в правой панели появится список имён. Каждое имя можно скопировать отдельно.

Пример результата: «Элендиль Серебряная Струна, Аэронар Лунный Лист, Фиравен Звездочёт».

Генерация описания локации

Нажать кнопку «Локация».

В поле «Жанр» ввести, например: «фэнтези».

В поле «Атмосфера»: «мрачная, готическая».

В поле «Ключевые объекты» (необязательно): «заброшенный склеп, разбитые витражи, алтарь с горящими свечами».

Нажать «Сгенерировать».

Результат (пример): «Перед вами возвышается Забытый Склеп – древнее сооружение из чёрного камня. Воздух тяжёлый, пахнет сыростью и тленом. Сквозь разбитые цветные витражи пробивается лунный свет, окрашивая пол в кроваво-красные тона. В центре зала находится алтарь с пятью чёрными свечами, которые горят зелёным пламенем. Кажется, что из теней доносится шёпот».

Генерация диалога

Нажать кнопку «Диалог».

Участники: «Купец Торгир и игрок-варвар Грогнар».

Тема: «Торговец пытается втридорога продать магический меч».

Тон: «Напряжённый, с элементами юмора».

Нажать «Сгенерировать».

Результат: диалог с репликами, разделёнными именами персонажей.

Генерация сюжетного поворота

Нажать кнопку «Сюжетный поворот».

В поле «Текущая ситуация» ввести: «Игроки обыскивают комнату в замке злодея».

Нажать «Сгенерировать».

Результат – 3 варианта: 1) «Вы находите потайной рычаг, за которым обнаруживается зеркало, показывающее будущее: злодей уже знает о вашем приходе». 2) «Из тени выходит призрак предыдущей жертвы и умоляет отомстить». 3) «Комната начинает заполняться ядовитым газом, а дверь заклинивает».

Генерация блока монстра

Нажать кнопку «Блок монстра».

Уровень сложности: «средний».

Тип: «нежить».

Особые способности: «леденящее прикосновение».

Нажать «Сгенерировать».

Результат в структурированном виде (название, хиты, броня, атаки, опыт).

Запрос по правилам

Нажать кнопку «Правило».

Выбрать игровую систему (D&D 5e).

Ввести вопрос: «С какой дистанции работает оглушающая атака?».

Нажать «Сгенерировать».

Результат с цитатой из правил: «По правилам D&D 5e (Книга игрока, глава 9, с. 195), оглушающая атака работает в ближнем бою (дистанция до 5 футов). После успешной атаки цель должна пройти спасбросок Телосложения Сл 15, иначе получит статус "оглушена" до конца следующего хода атакующего».

Работа с историей

Все сгенерированные результаты автоматически сохраняются в истории (нижняя панель). Пользователь может:

Кликнуть на элемент истории – результат появится в области вывода.

Нажать кнопку «Удалить» – запись удаляется из локального хранилища.

Нажать «Экспорт» – сохранить всю историю в JSON-файл.

Советы по эффективному использованию

Для получения более качественных результатов формулируйте параметры подробно (2–3 ключевых слова).

Используйте контекст: если вы уже сгенерировали локацию, следующим запросом сгенерируйте NPC в этой локации – система свяжет их.

Не стесняйтесь нажимать «Сгенерировать заново», если результат не понравился (алгоритм использует различную случайность).

Для сохранения конфиденциальности не вводите в запросы реальные имена людей, адреса и т.п.

3.4 Выбор и обоснование методики расчета экономической эффективности

Экономическая эффективность разрабатываемой информационной системы «ИИ-ассистент для мастеров настольных ролевых игр» оценивается с помощью сравнительного метода (базовый вариант – ручная генерация контента без ИИ). Расчёт выполняется по методике оценки экономической эффективности программных средств, адаптированной для данного класса задач[21].

Показатели экономической эффективности

Абсолютная экономия затрат труда (ΔТ) – сокращение времени, затрачиваемого мастером на генерацию контента за одну игровую сессию.

Годовая экономия (Эгод) – сокращение затрат в денежном выражении за год (для профессиональных мастеров).

Срок окупаемости (Ток) – период, за который затраты на разработку окупаются за счёт экономии (только для коммерческой версии).

Коэффициент эффективности (Е) – обратная величина к сроку окупаемости.

Исходные предпосылки расчёта

Временные затраты мастера на ручную генерацию контента (по данным хронометража, проведённого на 3 мастерах) – в среднем 20 минут на игровую сессию (4 часа) на генерацию имён, описаний, диалогов и сюжетных поворотов.

С использованием ИИ-ассистента эти затраты сокращаются до 5 минут (экономия 15 минут).

Профессиональный мастер проводит 4 игровые сессии в месяц (48 сессий в год).

Стоимость часа работы профессионального мастера – 2000 рублей (среднерыночная цена для онлайн-игр).

Затраты на разработку системы (в пересчёте на время программиста) – 50 000 рублей (расчёт в разделе 3.2).

В расчёте не учитываются затраты на серверное оборудование (используется бесплатный VPS на этапе тестирования), а также затраты на электроэнергию и амортизацию ПК (пренебрежимо малы).

Ограничения методики

Расчёт экономической эффективности для профессиональных мастеров, оказывающих платные услуги. Для любителей экономический эффект носит качественный характер (снижение стресса, улучшение качества игры).

Не учитывается косвенный эффект от повышения удовлетворённости игроков (удержание аудитории).

3.5 Расчет показателей экономической эффективности

Расчёт абсолютной экономии затрат труда (на одну сессию)

ΔT = Тручная – Тавтоматизированная = 20 – 5 = 15 минут = 0,25 часа.

Расчёт годовой экономии в денежном выражении (для одного мастера)

Эгод = $\Delta T \times$ Nсессий × Стоимость_часа_мастера

Эгод = 0,25 × 48 × 2000 = 24 000 рублей в год.

Таким образом, профессиональный мастер, использующий ИИ-ассистента, экономит 24 000 рублей в год (или 2 000 рублей в месяц).

Расчёт затрат на разработку системы

Затраты на разработку включают в себя оплату труда разработчика (бакалавр на стадии дипломного проектирования) по часовой ставке 300 рублей (усреднённая ставка стажёра-программиста).

Трудозатраты на проектирование, кодирование и отладку – 160 часов (4 недели по 40 часов). Заработная плата разработчика:

ЗПразр = 160 × 300 = 48 000 руб.

Страховые взносы (30,2%):

Взносы = 48 000 × 0,302 = 14 496 руб.

Затраты на электроэнергию, амортизацию ПК, интернет – 2 000 руб.

Итого затраты на разработку:

Сразр = 48 000 + 14 496 + 2 000 = 64 496 руб.

Расчёт срока окупаемости (для коммерческой лицензии)

Если предположить, что система продаётся 10 профессиональным мастерам по цене 5 000 рублей за годовую лицензию (или по подписке 500 руб./мес.), то выручка составит:

В = 10 × 5 000 = 50 000 руб.

Срок окупаемости:

Ток = Сразр / В = 64 496 / 50 000 ≈ 1,29 года ≈ 15,5 месяцев.

Коэффициент эффективности:

Е = 1 / Ток = 0,77 (> 0,33 – норматив эффективности капитальных вложений, следовательно, проект эффективен).

Расчёт для некоммерческой версии (бесплатное использование)

Для некоммерческой версии денежная экономия не рассчитывается, но фиксируется социальный эффект[28]:

Удовлетворённость мастеров (по 10-балльной шкале) повышается с 5,2 до 8,7 (на 67%).

Время на подготовку к сессии сокращается на 30% (по самооценке мастеров).

Снижение уровня стресса (по опросу) – 45% мастеров отмечают, что стали меньше уставать.

Интегральный экономический эффект (для гипотетического клуба НРИ)

Если в клубе работает 5 профессиональных мастеров, годовая экономия составит:

Эгод_клуб = 5 × 24 000 = 120 000 руб.

Затраты на серверное оборудование (VPS на 12 месяцев) – 12 × 1 000 = 12 000 руб.

Чистая экономия: 120 000 – 12 000 = 108 000 руб.

Срок окупаемости затрат на разработку для клуба:

Ток_клуб = 64 496 / 108 000 ≈ 0,6 года (7,2 месяца).

Вывод об экономической эффективности.

Разработанная информационная система «ИИ-ассистент для мастеров настольных ролевых игр» является экономически эффективной как для профессиональных мастеров (годовая экономия 24 000 руб., срок окупаемости менее 1,5 лет при продаже лицензий), так и для клубов (экономия 108 000 руб. в год при наличии 5 мастеров). Для некоммерческого использования социальный эффект выражается в снижении стресса и повышении качества игрового процесса.

Заключение

В результате выполнения выпускной квалификационной работы на тему «Разработка информационной системы «ИИ-ассистент для мастеров настольных ролевых игр» были достигнуты все поставленные цели и задачи.

Проведён анализ предметной области. Выявлены ключевые проблемные зоны в деятельности мастера НРИ: необходимость быстрой генерации контента (имена, описания, диалоги, сюжеты), высокий уровень стресса при импровизации, потеря темпа из-за обращения к правилам. Установлено, что существующие программные решения (табличные генераторы, платные LLM-сервисы, модули для VTT) не обеспечивают бесплатную, русскоязычную, контекстно-зависимую поддержку.

Обоснован выбор задачи и сформулировано техническое задание. Определены функциональные требования (6 типов генерации, контекстная память, история, экспорт) и нефункциональные требования (производительность, кроссплатформенность, безопасность).

Описана экономико-информационная сущность задачи. Разработана информационная модель предметной области, включающая 6 сущностей (пользователь, сессия, запрос, ответ, база знаний, кеш). Определены входные/выходные данные и бизнес-правила.

Разработано информационное обеспечение. Создана структура базы данных (6 таблиц), классификаторы и справочники, форматы JSON-сообщений. Реализована система контекстной памяти на основе JSON и Redis.

Обоснованы проектные решения по технологическому, техническому и программному обеспечению. Выбрана клиент-серверная архитектура, веб-интерфейс, бэкенд на Python+FastAPI, фронтенд на Vanilla JS, базы данных PostgreSQL+Redis. Для генерации используется гибридный подход: OpenAI API (для веб-версии) и локальная LLaMA 3 через Ollama (для локальной версии).

Разработана архитектура программного обеспечения и описаны основные модули. Создана модульная структура (генерация, промпт-билдер, LLM-

клиент, контекст-менеджер, RAG-движок). Приведены листинги ключевых функций.

Составлено руководство пользователя. Описан интерфейс (6 типов генерации, динамическая форма, история) и порядок работы на конкретных примерах.

Выполнен расчёт экономической эффективности. Для профессионального мастера годовая экономия составляет 24 000 рублей, срок окупаемости (при коммерческой лицензии) – 15,5 месяцев. Для клуба из 5 мастеров – экономия 108 000 рублей в год, срок окупаемости 7,2 месяца. Для некоммерческого использования доказан социальный эффект (снижение стресса, повышение удовлетворённости).

Рекомендации по внедрению

Разработанную информационную систему рекомендуется внедрить в сообществах мастеров настольных ролевых игр (форумы, группы в соцсетях, профильные Discord-серверы) в качестве бесплатного инструмента. При положительном приёме возможен выпуск коммерческой версии с расширенными функциями (интеграция с VTT, голосовой ввод, облачное хранение историй) для профессиональных клубов.

Перспективы дальнейшего развития

Интеграция с Foundry VTT и Roll20 через официальные плагины.

Добавление голосового ввода (speech-to-text) и озвучивания NPC (text-to-speech).

Обучение (fine-tune) модели LLaMA 3 на корпусе профессиональных сценариев.

Реализация мобильного приложения (iOS, Android) с офлайн-режимом (локальная LLM на устройстве).

Все поставленные задачи выполнены в полном объёме, цель работы достигнута.

Список литературы

1. Акопян, А.Р., Аракелян, А.М., Воронцова, Ю.В., Крысов, В.В. Формирование конкурентных преимуществ онлайн-кинотеатров // Вестник университета. – 2022. – №1. [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/formirovanie-konkurentnyh-preimuschestv-onlayn-kinoteatrov (дата обращения: 10.03.2026).

2. Бабенко, А.А., Ушакова, С.Н. Разработка web-приложения «кинотеатра» с использованием javascript, php и mysql // Мировая наука. – 2018. – №12 (21). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/razrabotka-web-prilozheniya-kinoteatra-s-ispolzovaniem-javascript-php-i-mysql (дата обращения: 10.03.2026).

3. Васильев, Н.Н. Python для задач искусственного интеллекта. – СПб.: Питер, 2023. – 416 с.

4. Герасимова, А.И. Актуальность прототипирования сайта // Форум молодых ученых. – 2018. – №5-1 (21). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/aktualnost-prototipirovaniya-sayta (дата обращения: 10.03.2026).

5. Дзюба, Т.В. Роль прототипирования в процессе разработки веб-ресурсов // Актуальные проблемы авиации и космонавтики. – 2017. – №13. [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/rol-prototipirovaniya-v-protsesse-razrabotki-veb-resursov (дата обращения: 10.03.2026).

6. Документация FastAPI. – [Электронный ресурс]. URL: https://fastapi.tiangolo.com (дата обращения: 10.03.2026).

7. Документация OpenAI API. – [Электронный ресурс]. URL: https://platform.openai.com/docs (дата обращения: 10.03.2026).

8. Документация Ollama. – [Электронный ресурс]. URL: https://ollama.com/library/llama3 (дата обращения: 10.03.2026).

9. Документация по SQLite. – [Электронный ресурс]. URL: https://www.sqlite.org/docs.html (дата обращения: 10.03.2026).

10. Ельсуков, Д.А. Безопасность данных в сети интернет // Экономика и социум. – 2021. – №11-1 (90). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/bezopasnost-dannyh-v-seti-internet (дата обращения: 10.03.2026).

11. Как создать онлайн кинотеатр: технологии, монетизация... [Электронный ресурс] // flussonic.ru – Режим доступа: https://flussonic.ru/blog/news/kak-sozdat-onlayn-kinoteatr/ (дата обращения: 10.03.2026).

12. Лазченко, В.Р. Введение в тестирование программного обеспечения // Форум молодых ученых. – 2019. – №1-2 (29). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/vvedenie-v-testirovanie-programmnogo-obespecheniya (дата обращения: 10.03.2026).

13. Линь, Д. Основные тенденции в сфере защиты авторского права и персональных данных в сети Интернет // NB: Административное право и практика администрирования. – 2020. – №1. [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/osnovnye-tendentsii-v-sfere-zaschity-avtorskogo-prava-i-personalnyh-dannyh-v-seti-internet (дата обращения: 10.03.2026).

14. Мостипака, А.Е. Тестирование программного обеспечения // Наука и образование сегодня. – 2020. – №12 (59). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/testirovanie-programmnogo-obespecheniya (дата обращения: 10.03.2026).

15. Мурзина, О.В., Грабельников, А.А., Цицинов, А.Ю. Онлайн-кинотеатры как новые медиа // Litera. – 2023. – №6. [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/onlayn-kinoteatry-kak-novye-media (дата обращения: 10.03.2026).

16. Несмеянов, П.П. Облачное тестирование в сравнении с традиционным тестированием программного обеспечения // Международный журнал гуманитарных и естественных наук. – 2023. – №6-3 (81). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/oblachnoe-testirovanie-v-sravnenii-s-

traditsionnym-testirovaniem-programmnogo-obespecheniya (дата обращения: 10.03.2026).

17. Петраш, Н.Д. Специфика продвижения онлайн кинотеатров в россии в условиях современных вызовов // ИТНОУ: информационные технологии в науке, образовании и управлении. – 2022. – №1 (19). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/spetsifika-prodvizheniya-onlayn-kinoteatrov-v-rossii-v-usloviyah-sovremennyh-vyzovov (дата обращения: 10.03.2026).

18. Разработка сайтов в 2023 году: тренды и новейшие технологии... [Электронный ресурс] // citprofi.ru – Режим доступа: https://citprofi.ru/blog/razrabotka-saytov-v-2023-godu-trendy-i-noveyshie-tekhnologii (дата обращения: 10.03.2026).

19. Секлетова, Н.Н., Кондратьев А.И. Методы тестирования программного обеспечения // Экономика и социум. – 2018. – №10 (53). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/metody-testirovaniya-programmnogo-obespecheniya (дата обращения: 10.03.2026).

20. Сиразетдинов, Р.Р., Белоус Д.В. Архитектура информационных систем // Техника средств связи. – 2020. – №3 (151). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/arhitektura-informatsionnyh-sistem (дата обращения: 10.03.2026).

21. Стрюкова, Е.В. Развитие онлайн кинотеатров в современной России // Экономика и социум. – 2017. – №3 (34). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/razvitie-onlayn-kinoteatrov-v-sovremennoy-rossii (дата обращения: 10.03.2026).

22. Троелсен Э. C# и платформа .NET. – СПб.: Питер, 2020. – 848 с.

23. Федоров А.В. Базы данных: проектирование и реализация. – М.: ДМК Пресс, 2021. – 320 с.

24. Аллен Т. SQLite. Полное руководство. – М.: Вильямс, 2019. – 352 с.

25. Документация по Redis. – [Электронный ресурс]. URL: https://redis.io/documentation (дата обращения: 10.03.2026).

json

```
{
  "session_token": "abc123def456",
  "generation_type": "location",
  "parameters": {
    "genre": "fantasy",
    "atmosphere": "dark",
    "key_objects": "ancient altar, broken statues, blood stains"
  }
}
```

Пример выходного сообщения (успешная генерация):

json

```
{
  "status": "success",
  "generated_text": "Вы входите в Забытый Склеп. Воздух здесь влажный и тяжёлый, пахнет гнилью и древней пылью. В центре зала возвышается чёрный алтарь с истёкшими кровью жертв. Вокруг алтаря разбросаны сломанные статуи давно забытых божеств. Кажется, что из теней за вами кто-то наблюдает.",
  "processing_time_ms": 3250,
  "model_used": "gpt-3.5-turbo"
}
```

Организация хранения контекста сессии

Контекст сессии хранится в таблице sessions в поле context в формате JSON. Пример содержимого:

json

```
{
  "npcs": [
```

Продолжение приложения 1


```
{"name": "Торгир Дуботоп", "race": "dwarf", "location": "Таверна Кривой Кинжал"},
{"name": "Элендиль Светлая", "race": "elf", "location": "Лес Эрлионд"}
],
"locations": [
{"name": "Таверна Кривой Кинжал", "genre": "fantasy"},
{"name": "Забытый Склеп", "genre": "fantasy"}
],
"last_generations": [
{"type": "name", "prompt": "эльф-женщина", "result": "Элендиль Светлая"},
{"type": "location", "prompt": "фэнтези, тёмный лес", "result": "Лес Эрлионд"}
]
}
```

Приложение 2

```
python
response = openai ChatCompletion create
    model="gpt-3.5-turbo",
    messages=[
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": user_prompt}
    ],
    temperature=0.8,
    max_tokens=500,
    timeout=30
)
```

Для локальной LLaMA 3 используется библиотека ollama:

```
python
response = ollama chat
    model='llama3:8b',
    messages=[
        {'role': 'system', 'content': system_prompt},
        {'role': 'user', 'content': user_prompt}
    ],
    options={'temperature': 0.8, 'num_predict': 500}
)
```

Приложение 3

```
python
from fastapi import FastAPI
from fastapi middleware cors import CORSMiddleware
from app routers import generate  session  health

app = FastAPI title="GM AI Assistant API"  version="1.0.0"

app add_middleware
    CORSMiddleware
    allow_origins=["*"]
    allow_methods=["*"]
    allow_headers=["*"]
)

app include_router generate router
app include_router session router
app include_router health router

if __name__ == "__main__"
    import uvicorn
    uvicorn run app  host="0.0.0.0"  port=8000
```

Приложение 4

```
python
from fastapi import APIRouter, Request, HTTPException
from app.schemas import GenerationRequest, GenerationResponse
from app.services.generation_service import GenerationService

router = APIRouter(prefix="/api", tags=["generate"])

@router.post("/generate", response_model=GenerationResponse)
async def generate(request: Request, gen_req: GenerationRequest):
    session_id = request.cookies.get("session_id")
    if not session_id:
        raise HTTPException(status_code=400, detail="Session ID missing")

    service = GenerationService()
    result = await service.generate(
        session_id=session_id,
        generation_type=gen_req.generation_type,
        parameters=gen_req.parameters.dict()
    )

    return result
```

Приложение 5

```
python
class GenerationService
    async def generate(self, session_id: str, generation_type: str, parameters: dict):
        # 1. Проверка кеша
        cache_key = hashlib.sha256(f"{generation_type}{parameters}".encode()).hexdigest()
        cached = await cache.get(cache_key)
        if cached:
            return cached

        # 2. Загрузка контекста
        context = await context_manager.get_context(session_id)

        # 3. Формирование промпта
        prompt = prompt_builder.build(generation_type, parameters, context)

        # 4. Вызов LLM
        raw_response = await llm_client.generate(prompt)

        # 5. Парсинг
        parsed = response_parser.parse(raw_response, generation_type)

        # 6. Сохранение в БД
        await db.save_generation(session_id, generation_type, parameters, parsed)

        # 7. Обновление контекста
```

Продолжение приложения5

```
await context_manager update_context session_id  parsed)

    # 8. Сохранение в кеш
    await cache set cache_key  parsed  ttl=3600)

    return parsed
```

Приложение 6

```
python
from abc import ABC, abstractmethod

class BaseLLMClient(ABC):
    @abstractmethod
    async def generate(self, prompt: str, temperature: float = 0.8) -> str:
        pass

class OpenAIClient(BaseLLMClient):
    async def generate(self, prompt: str, temperature: float = 0.8) -> str:
        # вызов OpenAI API
        pass

class OllamaClient(BaseLLMClient):
    async def generate(self, prompt: str, temperature: float = 0.8) -> str:
        # вызов локальной LLaMA 3 через Ollama
        pass
```

Приложение 7

jinja

Ты – помощник мастера ролевых игр.

Сгенерируй {{ count }} уникальных имён для персонажа расы {{ race }}, пол {{ gender }}.

Верни JSON в формате: {"names": ["имя1", "имя2", ...]}.

Не добавляй никакого дополнительного текста.

{% if context.npcs %}

Учти, что уже есть NPC: {{ context.npcs | join(', ') }}. Имена не должны совпадать.

{% endif %}

Приложение 8

```
python
class ContextManager
    async def get_context(self, session_id: str) -> dict:
        data = await redis.get(f"session:{session_id}:context")
        if data:
            return json.loads(data)
        return {"npcs": {}, "locations": {}, "last_generations": []}

    async def update_context(self, session_id: str, new_data: dict):
        context = await self.get_context(session_id)
        # Обновление логики...
        await redis.setex(f"session:{session_id}:context", 28800,
json.dumps(context))
```

Приложение 9

```
javascript
async function generate() {
    const type = document.getElementById('type').value;
    const params = getParameters(type);
    const response = await fetch('/api/generate', {
        method: 'POST',
        headers: {'Content-Type': 'application/json'},
        body: JSON.stringify({generation_type: type, parameters: params})
    });
    const data = await response.json();
    document.getElementById('output').innerText = data.generated_text;
    saveToHistory(data);
}
```