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 AI Flashcard Generator для создания карточек запоминания на основе ИИ»

Исполнитель Трушин Даниил Андреевич

Руководитель к.т.н., Тарасов Елизар Саввич

«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 2026 г.
Мартынова А.В.
ПОДПИСЬ РАСШИФРОВКА ПОДПИСИ

Туапсе
2026

ОГЛАВЛЕНИЕ

Введение

Работа с учебной, профессиональной и справочной информацией всё чаще связана не с одним источником, а с большим количеством разнородных материалов. Пользователь может работать с электронными учебниками, методическими пособиями, PDF-файлами, текстовыми документами, веб-страницами, видеолекциями, презентациями, технической документацией и собственными заметками. Такие материалы содержат полезную информацию, однако не всегда удобны для повторения и самопроверки. Чтобы использовать их в дальнейшем, пользователь должен выделить основные положения, сформулировать вопросы, подготовить ответы и привести материал к более компактной форме.

Одним из распространённых способов закрепления информации являются карточки запоминания. Карточка обычно содержит вопрос, термин или краткую подсказку, а также ответ, определение или пояснение. Такой формат удобен тем, что пользователь не просто перечитывает готовый текст, а пытается самостоятельно воспроизвести нужную информацию. Карточки часто применяются совместно с интервальным повторением, при котором материал возвращается к пользователю через определённые промежутки времени. Такой подход используется в системах Anki, Quizlet, SuperMemo и других программных решениях [30], [31], [32].

Несмотря на пользу карточек, их создание часто остаётся ручным и однообразным процессом. Пользователь открывает исходный материал, выбирает важный фрагмент, формулирует вопрос, записывает ответ и переносит результат в отдельное приложение или файл. При небольшом объёме текста такой подход допустим, но при работе с крупными документами, лекциями, статьями или технической документацией подготовка карточек начинает занимать значительное время. В результате пользователь тратит усилия не только на изучение материала, но и на техническое оформление базы для повторения.

Частично эту проблему решают облачные сервисы и универсальные чатботы, использующие искусственный интеллект для генерации вопросов и отве-

тов; к таким специализированным решениям можно отнести сервисы Knowt и Monic.ai [33], [34].Они позволяют быстрее получить первичный набор карточек, однако имеют ограничения. Многие такие сервисы требуют подключения к интернету, работают по подписочной модели и предполагают загрузку пользовательских материалов на внешний сервер. Для открытых учебных материалов это не всегда критично, но при работе с личными заметками, внутренними документами, черновиками и профессиональными материалами такой способ обработки может быть нежелательным.

Дополнительная проблема связана с разрозненностью инструментов, что также снижает удобство пользовательского взаимодействия и усложняет работу с материалом [21], [22], [23].Источник может храниться в PDF-файле, заметки — в текстовом редакторе, карточки — в отдельной системе повторения, а экспорт — выполняться через ещё один сервис. При таком подходе пользователь постоянно переключается между программами, а связь между карточкой и исходным фрагментом может теряться. Поэтому возникает необходимость в информационной системе, которая объединяет основные этапы работы: добавление источника, извлечение текста, генерацию карточек, проверку результата, хранение данных, повторение и экспорт.

В рамках данной выпускной квалификационной работы рассматривается разработка информационной системы “AI FlashcardGenerator” для создания карточек запоминания на основе ИИ. Система предназначена для локальной работы с источниками информации и карточками. Пользователь добавляет материал, после чего приложение подготавливает текст, передаёт его в модуль генерации, сохраняет полученные карточки в базе данных и связывает их с исходным источником. Далее карточки можно просматривать, редактировать, повторять и экспортировать во внешние форматы.

Разрабатываемая система реализуется как локальное веб-приложение. Серверная часть запускается на компьютере пользователя, а взаимодействие с приложением выполняется через браузерный интерфейс. Для серверной части используется Python и FastAPI [35], [36], для хранения данных применяется

SQLite [38], клиентская часть реализуется средствами HTML, CSS и JavaScript [41], [42], [43]. Генерация карточек выполняется локальным модулем языковой модели, интегрированным в серверную часть приложения.

Актуальность темы определяется необходимостью сократить трудоёмкость подготовки карточек запоминания и объединить работу с источниками, генерацией, хранением, повторением и экспортом в одном приложении. Локальный характер системы также важен, поскольку пользовательские материалы могут содержать личные, учебные или профессиональные данные, которые не всегда желательно передавать внешним сервисам.

Объектом исследования является процесс подготовки информационных материалов для повторения и запоминания.

Предметом исследования являются методы, алгоритмы и программные средства автоматизации создания карточек запоминания на основе обработки источников информации и применения языковых моделей.

Цель выпускной квалификационной работы заключается в разработке информационной системы “AI FlashcardGenerator” для создания карточек запоминания на основе ИИ, обеспечивающей добавление источников, генерацию карточек, хранение результата, визуальную организацию материала, повторение и экспорт данных.

Для достижения поставленной цели необходимо решить следующие задачи:

– провести анализ предметной области, связанной с подготовкой материалов для повторения и карточками запоминания;

– рассмотреть существующие программные решения для создания и повторения карточек, определить их преимущества и ограничения;

– сформулировать требования к информационной системе “AI FlashcardGenerator”;

– спроектировать структуру данных для хранения колод, источников, карточек, тегов и статусов повторения;

– выбрать и обосновать технологический стек разработки;

– разработать серверные модули для работы с источниками, генерацией карточек, базой данных, импортом, экспортом и повторением;
– разработать клиентский интерфейс для работы с графом знаний, списком карточек, инспектором объектов, поиском и режимом повторения;
– реализовать механизм проверки результатов генерации, исключающий сохранение некорректных, дублирующихся или неполных карточек;
– рассчитать экономическую эффективность разработки и применения информационной системы.

В работе использовались методы системного анализа, сравнительного анализа программных решений, проектирования информационных систем, моделирования данных, проектирования пользовательских интерфейсов и функционального тестирования программного обеспечения [6, с. 18], [8, с. 31], [16, с. 55], [17, с. 41].Практическая значимость работы состоит в создании программного продукта, который может применяться для подготовки карточек запоминания по собственным материалам пользователя. Система позволяет сократить объём ручных операций, сохранить связь между карточкой и исходным источником, организовать материал в виде локальной базы знаний, перейти к повторению и выгрузить результат в распространённые форматы.

Выпускная квалификационная работа состоит из введения, трёх глав, заключения, списка использованных источников и приложения, что соответствует принятой структуре оформления выпускной квалификационной работы [1, с. 6].В первой главе рассматривается предметная область и обосновывается необходимость разработки системы. Во второй главе приводятся проектные решения по информационному, технологическому, техническому и программному обеспечению. В третьей главе выполняется расчёт экономической эффективности. В приложении размещается фрагмент листинга программных модулей.

1 Аналитическая часть

1.1 Анализ предметной области

Предметная область выпускной квалификационной работы связана с подготовкой информационных материалов к повторению и запоминанию и рассматривается как часть задачи проектирования прикладной информационной системы [6, с. 24], [7, с. 19].В рамках данной работы рассматривается процесс, при котором пользователь получает исходный материал, выделяет из него значимую информацию и преобразует её в форму, пригодную для самопроверки. Такая задача возникает не только в учебной деятельности, но и при изучении профессиональной документации, подготовке к проверке знаний, освоении новой темы, работе с техническими инструкциями и систематизации материалов по проекту.

Исходные материалы могут иметь разный формат и структуру. Пользователь может работать с электронными учебниками, методическими пособиями, PDF-документами, DOCX-файлами, веб-страницами, презентациями, видеолекциями, текстовыми заметками и фрагментами технической документации. Каждый из этих источников может содержать полезные сведения, однако сам по себе источник не всегда удобен для дальнейшего повторения. Длинный текст необходимо перечитать, выделить в нём основные положения, отделить важные сведения от второстепенных пояснений и привести материал к более компактной форме.

Одним из способов такой переработки является создание карточек запоминания. Карточка представляет собой небольшую информационную единицу, в которой одна часть содержит вопрос, термин или подсказку, а другая — ответ, определение или пояснение. Такой формат удобен для самопроверки, потому что пользователь не просто повторно читает готовый текст, а пытается самостоятельно воспроизвести нужную информацию. При таком подходе быстрее обнаруживаются пробелы в понимании темы, а повторение становится более активным.

Карточки запоминания часто используются вместе с интервальным повторением, поскольку распределение повторений во времени рассматривается как один из способов повышения устойчивости запоминания [25, с. 43], [26, с. 58], [28, с. 354–380]. Его смысл заключается в том, что материал предъявляется пользователю не постоянно, а через определённые промежутки времени. Если карточка вспоминается легко, интервал до следующего показа увеличивается. Если ответ вызывает затруднение, карточка возвращается раньше. Такой подход связан с исследованиями памяти, активного воспроизведения и распределённой практики, согласно которым повторение, разделённое во времени, помогает лучше удерживать информацию.

На практике карточки могут применяться для разных видов информации: терминов, определений, дат, формул, классификаций, команд программирования, свойств объектов, этапов алгоритмов и взаимосвязанных понятий. В области информационных технологий такой формат может использоваться для изучения языков программирования, библиотек, фреймворков, архитектурных подходов, команд, стандартов и технических терминов. Поэтому карточки запоминания являются универсальным инструментом работы с материалом, который требует регулярного повторения.

Несмотря на простоту идеи, создание карточек часто остаётся трудоёмким ручным процессом. Пользователь должен открыть источник, выбрать важный фрагмент, сформулировать вопрос, подготовить ответ, при необходимости добавить пояснение, а затем перенести карточку в отдельное приложение или файл. Если требуется создать несколько карточек, такая работа не вызывает серьёзных затруднений. Однако при обработке большой главы, лекции, статьи или технического документа количество однотипных действий быстро увеличивается. В результате подготовка карточек может занимать больше времени, чем само повторение.

Основная проблема предметной области заключается не в отсутствии программ для повторения, а в сложности первичной подготовки качественного материала. Существуют приложения, которые хорошо показывают карточки по

расписанию, но пользователь всё равно должен самостоятельно наполнить их содержанием. При большом объёме источников это приводит к усталости, пропуску важных сведений, повторяющимся формулировкам и ошибкам при переносе текста.

Дополнительная сложность связана с разнородностью источников. PDF-документ может содержать номера страниц, колонтитулы, таблицы и изображения. Веб-страница может включать основной текст, меню, рекламные блоки и служебные элементы. Техническая документация часто содержит фрагменты кода, параметры, списки и вложенные разделы. Поэтому перед созданием карточек материал необходимо очистить, разбить на смысловые фрагменты и привести к виду, пригодному для дальнейшей обработки.

В последние годы для подобных задач всё чаще используются языковые модели. Они могут анализировать текстовые фрагменты, выделять смысловые элементы, формулировать вопросы и ответы, сокращать материал и приводить неструктурированную информацию к более удобному виду.Для задачи создания карточек это особенно важно, поскольку языковая модель способна взять на себя наиболее однообразную часть работы: первичное выделение фактов и преобразование их в пары «вопрос — ответ».

При этом применение искусственного интеллекта не отменяет необходимости контроля результата, поскольку программная система должна сохранять проверяемость данных, устойчивость к ошибкам и возможность последующей корректировки результата пользователем [16, с. 86], [17, с. 92].Сгенерированная карточка может быть слишком общей, повторять другую карточку, содержать неполный ответ или терять связь с исходным фрагментом. Поэтому информационная система должна не только обращаться к языковой модели, но и выполнять дополнительные операции: подготавливать текст, проверять структуру ответа, исключать некорректные или неполные карточки, сохранять связь с источником и предоставлять пользователю возможность редактирования.

Для рассматриваемой предметной области важна связь карточки с исходным материалом. Если карточка создана вручную, пользователь обычно понимает, откуда она была взята.

При автоматической генерации эта связь может потеряться. Поэтому система должна хранить не только вопрос и ответ, но и сведения об источнике: название документа, тип материала, исходный фрагмент или цитату.

Это позволяет проверить корректность карточки и при необходимости вернуться к первичному тексту.

Также важна организация уже созданных карточек. При большом количестве элементов простой список становится неудобным.

Пользователю нужно понимать, какие карточки относятся к определённому источнику, какие связаны с одной темой, какие требуют повторения, а какие уже отредактированы или готовы к экспорту.

Для этого необходимы колоды, теги, статусы, поиск, фильтрация и визуальное представление связей между источниками и карточками.

В системе “AI Flashcard Generator” данная предметная область рассматривается как процесс преобразования исходных материалов в структурированную базу карточек запоминания.

Исходный материал выступает входной информацией, а карточки, теги, связи с источниками, статусы повторения и экспортируемые файлы — выходной информацией системы.

Между этими состояниями располагается программный процесс обработки: извлечение текста, очистка, разбиение на фрагменты, генерация, проверка, сохранение и отображение результата.

Таким образом, анализ предметной области показывает, что пользователю требуется не только средство повторения, но и инструмент подготовки базы карточек. Такая система должна уменьшать объём ручных действий, поддерживать работу с разными источниками, сохранять связь карточек с исходным материалом, обеспечивать контроль результата генерации и позволять использовать созданные карточки для повторения и экспорта.

1.2 Обоснование выбора задачи

Выбор задачи разработки информационной системы “AI FlashcardGenerator” связан с необходимостью автоматизировать процесс подготовки карточек запоминания по исходным информационным материалам. Для обоснования выбора задачи необходимо рассмотреть существующие решения, которые применяются для создания карточек, интервального повторения, генерации вопросов и организации материалов.

Существующие программные решения можно условно разделить на несколько групп: системы интервального повторения, онлайн-сервисы карточек, AI-сервисы генерации учебных материалов и инструменты организации знаний [30], [31], [32], [33], [34].К первой группе относятся классические системы интервального повторения, например Anki и SuperMemo. Ко второй группе относятся онлайн-сервисы карточек, такие как Quizlet. К третьей группе можно отнести сервисы с применением искусственного интеллекта, которые формируют вопросы и ответы по загруженному тексту. Отдельную группу составляют универсальные инструменты для заметок и организации знаний, например Obsidian, Notion и Logseq.

Anki является одним из наиболее известных инструментов для интервального повторения и используется для организации колод, карточек, тегов и расписания повторений [30].Программа позволяет создавать колоды, добавлять карточки разных типов, использовать теги, прикреплять изображения и настраивать режим повторения. Сильной стороной Anki является развитый механизм повторения и возможность локальной работы. Однако основное ограничение состоит в том, что пользователь обычно должен самостоятельно создавать карточки. При большом объёме исходных материалов ручное наполнение колод становится трудоёмким процессом [30].

Quizlet больше ориентирован на онлайн-обучение, быстрый обмен наборами карточек и использование готовых учебных комплектов. Пользователь может создавать термины и определения, запускать режимы тренировки и тес-

тирования, а также работать с уже опубликованными наборами. Такой сервис удобен для простых карточек вида «термин — определение», однако он зависит от интернет-соединения и внешней платформы. Кроме того, для локальной обработки личных или рабочих материалов облачный характер сервиса может быть ограничением [31].

SuperMemo связан с развитием подходов к интервальному повторению и ориентирован на планирование показов карточек с учётом результатов пользователя [32], [26, с. 61].Его преимуществом является методическая основа повторения, однако для рассматриваемой задачи этого недостаточно. Система повторения сама по себе не решает проблему первичной подготовки карточек из разнородных источников. Пользователь всё равно должен самостоятельно преобразовать документ, статью или конспект в набор вопросов и ответов [32].

Сервисы с применением искусственного интеллекта частично решают проблему ручного создания карточек. Они позволяют загрузить текст, PDF-файл или ссылку и получить первичный набор вопросов и ответов. Такой подход ближе к задаче данной выпускной квалификационной работы, так как он автоматизирует начальный этап подготовки материала. Однако у подобных сервисов есть ограничения: зависимость от интернета, подписочная модель, отсутствие полного контроля над обработкой данных и необходимость передачи пользовательских материалов на внешний сервер.

Для “AI FlashcardGenerator” особенно важен вопрос локальной обработки. Пользователь может работать не только с открытыми учебными материалами, но и с личными заметками, черновиками, внутренними документами, рабочими файлами и профессиональной документацией. В таких случаях передача материалов во внешний сервис может быть нежелательной. Поэтому разрабатываемая система должна обеспечивать обработку источников на стороне пользователя и снижать зависимость от облачных платформ.

Универсальные чат-боты также могут использоваться для создания карточек. Пользователь может вставить фрагмент текста и попросить сформировать вопросы и ответы. Такой способ гибкий, но он не является полноценной

информационной системой. Чат-бот не хранит колоды в локальной базе данных, не связывает карточки с источниками, не ведёт статусы повторения, не предоставляет визуальный граф материалов и не всегда обеспечивает удобный экспорт результата. После генерации пользователю всё равно приходится вручную переносить карточки в другое приложение.

Системы заметок и организации знаний, такие как Obsidian, Notion и Logseq, помогают структурировать материалы, использовать теги, создавать связи между страницами и вести личную базу знаний. Однако их основная задача заключается не в автоматической генерации карточек из источников. Для получения карточек обычно требуются плагины, ручная разметка или дополнительные внешние инструменты. Поэтому такие системы полезны для организации информации, но не закрывают полностью задачу автоматизированной подготовки карточек.

Проведённое сравнение показывает, что существующие решения закрывают отдельные части общей задачи. Anki и SuperMemo хорошо подходят для повторения, но требуют ручного наполнения. Quizlet удобен для простых учебных наборов, но зависит от облачной платформы. AI-сервисы автоматизируют генерацию, но обычно требуют загрузки материалов на внешний сервер. Универсальные чат-боты гибки, но не являются системой хранения, повторения и экспорта. Инструменты заметок помогают организовать знания, но не специализируются на создании карточек.

На основании анализа можно выделить основные требования к разрабатываемой системе с учётом общих подходов к формированию требований и стадий создания автоматизированных систем [4], [5], [16, с. 102].Она должна поддерживать работу с разными источниками, выполнять предварительную обработку текста, создавать карточки с использованием языковой модели, сохранять связь карточек с исходным материалом, предоставлять пользователю возможность редактирования, поддерживать теги, статусы, повторение и экспорт. При этом генерация и экспорт должны быть разделены: сначала система создаёт универсальные карточки, а затем пользователь выбирает формат выгрузки.

Важным требованием является контроль качества результата. Система не должна сохранять любые ответы языковой модели без проверки. Карточка должна содержать понятный вопрос, содержательный ответ и связь с исходным материалом. Если модель сформировала меньше карточек, чем ожидалось, система должна сохранить только валидные элементы, а не заменять недостающие карточки заранее прописанными шаблонами.

Для более наглядного обоснования выбора задачи разработки была выполнена сравнительная характеристика существующих решений, применяемых для создания карточек, интервального повторения, генерации вопросов и организации учебных материалов. Результаты сравнения представлены в таблице 1.1.

Таблица 1.1 – Сравнительная характеристика существующих решений

Anki	Интервальное повторение карточек	Локальная работа, развитый механизм повторения, поддержка колод и тегов	Основное наполнение карточек выполняется вручную
Quizlet	Онлайн-сервис карточек и тренировок	Удобство простых наборов, режимы тренировки, доступ через веб-интерфейс	Зависимость от облачной платформы и интернет-соединения
SuperMemo	Интервальное повторение и планирование повторов	Методическая основа повторения, ориентация на долговременное запоминание	Не решает задачу автоматического создания карточек из разнородных источников

Продолжение таблицы 1.1

AI-сервисыгенерации карточек	Автоматическое создание вопросов и ответов	Быстрое получение набора карточек по тексту или документу	Часто требуется загрузка материалов на внешний сервер, возможны ограничения подписки
Универсальныечат-боты	Генерация текста и ответов на запросы пользователя	Гибкость формулировок и возможность работы с разными темами	Нет единой локальной базы карточек, связи с источниками, повторения и экспорта
Системы заметок	Организация материалов и связей между заметками	Удобная структура знаний, теги, ссылки между страницами	Не являются специализированной системой генерации и повторения карточек
AI Flash-card Generator	Локальная генерация, хранение, повторение и экспорт карточек	Работа с источниками, карточками, тегами, проверкой результата, повторением и экспортом в одном приложении	Требует локального запуска и предварительной настройки программной среды

Проведённое сравнение показывает, что существующие решения закрывают отдельные этапы работы с карточками, но не объединяют полный цикл

обработки материала в одной локальной системе. Поэтому разработка “AIF-lashcardGenerator” позволяет объединить добавление источников, генерацию карточек, хранение, проверку, повторение и экспорт в рамках единого программного продукта.

Таким образом, выбор задачи разработки информационной системы “AI FlashcardGenerator” является обоснованным. На рынке существуют инструменты для повторения, генерации текста и организации знаний, однако пользователю требуется единая локальная среда, которая объединяет добавление источников, генерацию карточек, проверку результата, хранение, визуальную организацию, повторение и экспорт.

1.3 Обоснование необходимости разработки информационной системы

Проведённый анализ предметной области и существующих программных решений показывает, что задача создания карточек запоминания остаётся недостаточно автоматизированной. На практике пользователь часто вынужден использовать несколько разных инструментов: один — для просмотра исходного материала, другой — для генерации вопросов, третий — для хранения карточек, четвёртый — для повторения или экспорта. Такой подход увеличивает количество ручных действий и затрудняет регулярную работу с материалами.

Необходимость разработки информационной системы “AI FlashcardGenerator” связана с тем, что существующие решения не обеспечивают полного цикла работы с источником. Для пользователя важно не только получить отдельные вопросы и ответы, но и сохранить исходный материал, связать с ним карточки, проверить результат генерации, отредактировать неудачные элементы, организовать карточки по колодам и тегам, перейти к повторению и при необходимости выгрузить результат во внешний формат.

Первым основанием для разработки является высокая трудоёмкость ручной подготовки карточек. При традиционном подходе пользователь самостоятельно читает материал, выделяет ключевые сведения, формулирует вопросы,

записывает ответы и переносит их в систему повторения. Если материал содержит большое количество терминов, определений, классификаций или взаимосвязанных фактов, количество однотипных действий быстро возрастает. Автоматизация этого этапа позволяет сократить подготовительную работу и быстрее перейти к повторению содержания.

Вторым основанием является разнородность исходных материалов. В реальной работе пользователь может использовать PDF-документы, DOCX-файлы, веб-страницы, заметки, презентации, изображения и фрагменты технической документации. Каждый формат требует предварительной обработки. Информационная система должна приводить разные источники к единому внутреннему представлению, пригодному для дальнейшего анализа и генерации карточек.

Третьим основанием является необходимость сохранения связи между карточкой и исходным источником. При ручном копировании или использовании универсального чат-бота эта связь часто теряется. В результате пользователь может получить набор вопросов и ответов, но не всегда понимает, из какого фрагмента материала они были сформированы. В “AI FlashcardGenerator” карточка должна храниться как объект, связанный с конкретным источником, что позволяет проверить корректность ответа и при необходимости вернуться к исходному тексту.

Четвёртым основанием является потребность в локальной обработке данных. Пользовательские материалы могут содержать личные заметки, рабочие документы, черновики, внутренние инструкции и другие сведения, которые нежелательно передавать во внешние сервисы. Локальная информационная система позволяет обрабатывать такие материалы на компьютере пользователя и уменьшает зависимость от облачных платформ.

Пятым основанием является необходимость контроля качества генерации. Использование языковой модели не должно сводиться к механическому сохранению любого полученного ответа. Система должна проверять, что карточка содержит вопрос, ответ и смысловую связь с исходным материалом. Если мо-

дель сформировала меньше валидных карточек, чем было указано пользователем, сохранению должны подлежать только корректные элементы.

Подмена результата заранее прописанными шаблонными карточками снижает учебную ценность материала и не соответствует назначению системы.

На основании этого к информационной системе “AI Flashcard Generator” предъявляются следующие требования:

1) поддержка разных типов источников; извлечение и предварительная очистка текста;

2) генерация карточек с использованием языковой модели;

3) сохранение связи карточек с источниками; хранение колод, карточек, тегов и статусов повторения; возможность редактирования результата;

4) поиск и фильтрация данных; поддержка повторения;

5) экспорт готовых карточек в распространённые форматы; работа приложения в локальном режиме.

К нефункциональным требованиям относятся удобство интерфейса, устойчивость к ошибкам генерации, возможность работы с большими текстовыми фрагментами, расширяемость архитектуры, сохранение данных между запусками и минимизация зависимости от внешних сервисов [16, с. 113], [17, с. 119], [21, с. 84], [24].

Эти требования важны, поскольку система должна использоваться не как разовый генератор вопросов, а как локальная среда для подготовки и организации карточек запоминания.

Таким образом, необходимость разработки “AI Flashcard Generator” обусловлена высокой трудоёмкостью ручного создания карточек, разнородностью источников информации, ограничениями облачных сервисов, потребностью в локальной обработке данных и необходимостью контроля качества генерации.

Разрабатываемая система должна объединить добавление источников, генерацию карточек, проверку результата, хранение, визуальную организацию, повторение и экспорт в одном приложении.

1.4 Экономико-информационная сущность задачи

Экономико-информационная сущность задачи разработки информационной системы «AI FlashcardGenerator» заключается в преобразовании разнородных источников информации в структурированные карточки запоминания. На вход система получает текстовые материалы, документы, ссылки, сведения о колоде, параметры генерации и теги. На выходе формируются карточки, связанные с источниками, колодами, статусами повторения и экспортируемыми файлами.

С информационной точки зрения задача представляет собой последовательную обработку данных, что соответствует общему подходу к описанию информационных процессов в прикладных системах [6, с. 44], [13, с. 71].Сначала пользователь добавляет источник или вводит текст. Затем система извлекает содержимое, выполняет предварительную очистку, разбивает материал на фрагменты, передаёт подготовленный текст в модуль генерации и получает набор карточек. После проверки валидные карточки сохраняются в базе данных и становятся доступными для просмотра, редактирования, повторения и экспорта.

Основными информационными объектами системы являются колода, источник, карточка, тег и состояние повторения; их выделение связано с необходимостью формального описания данных и связей между сущностями информационной системы [9, с. 37], [10, с. 52], [14, с. 80].Колода объединяет материалы по теме или проекту. Источник хранит исходный материал и его метаданные. Карточка содержит вопрос, ответ, связь с источником, теги и параметры повторения. Состояние повторения определяет, когда карточка должна быть показана пользователю повторно.

Важной особенностью задачи является сохранение связи между карточкой и исходным материалом. Это позволяет пользователю проверить корректность ответа, вернуться к исходному фрагменту и понять происхождение кар-

точки. При автоматической генерации такая связь особенно важна, поскольку результат должен быть доступен для просмотра и редактирования.

Экономический смысл разработки состоит в сокращении времени на подготовку карточек. При ручном способе пользователь самостоятельно выделяет ключевые сведения, формулирует вопросы, оформляет ответы и переносит их в отдельную систему. При использовании «AI FlashcardGenerator» значительная часть этих операций выполняется автоматически, а пользователь сосредоточивается на проверке и уточнении результата.

Таким образом, система автоматизирует преобразование исходных материалов в локальную базу карточек запоминания. Это снижает объём ручной работы, уменьшает количество переключений между программами, сохраняет связь карточек с источниками и позволяет использовать созданные материалы для повторения и экспорта.

2 Проектная часть

2.1 Информационное обеспечение задачи

Информационное обеспечение информационной системы “AI FlashcardGenerator” определяет состав данных, используемых в приложении, их структуру, связи между объектами и порядок хранения [6, с. 91], [7, с. 88], [8, с. 74].Для рассматриваемой системы важно учитывать, что пользователь работает не только с готовыми карточками, но и с исходными источниками, колодами, тегами, статусами повторения и экспортируемыми результатами.

Основная задача информационного обеспечения состоит в том, чтобы представить разнородные материалы пользователя в виде структурированной базы данных. Исходный материал может поступать в систему как текст, документ, ссылка или иной информационный объект. После обработки он связывается с карточками запоминания, которые могут редактироваться, повторяться и экспортироваться во внешние форматы.

В системе можно выделить несколько групп данных. Первая группа включает исходные материалы, добавляемые пользователем. Вторая группа содержит сведения об этих материалах: название, тип источника, дату добавления, теги, цветовую маркировку и служебные параметры отображения. Третья группа связана с карточками: вопросом, ответом, цитатой из источника, статусом повторения, тегами и принадлежностью к колоде. Четвёртая группа включает данные, необходимые для повторения и экспорта.

Основными информационными объектами являются колода, источник и карточка. Колода используется для объединения материалов по теме, проекту или другому пользовательскому основанию. Источник представляет собой добавленный материал, из которого могут быть сформированы карточки. Карточка является конечным элементом, используемым для повторения, редактирования и выгрузки. Между объектами устанавливаются связи: источник относится к колоде, карточка относится к колоде и может быть связана с конкретным источником.

Такое построение данных позволяет сохранить контекст создания карточек. Если карточка хранится только как вопрос и ответ, пользователь со временем может потерять понимание, из какого материала она была получена. В “AI FlashcardGenerator” карточка дополнительно связывается с источником, что позволяет вернуться к исходному тексту и проверить корректность формулировки.

2.1.1 Характеристика входной и выходной информации

Входная информация системы включает данные, которые пользователь передаёт приложению для обработки, а также параметры, влияющие на дальнейшую работу. К входной информации относятся:

– текстовые материалы, введённые вручную;

– файлы документов, содержащие учебную, справочную или профессиональную информацию;

– ссылки на веб-страницы;

– сведения о выбранной колоде;

– параметры генерации карточек;

– теги и дополнительные пометки;

– команды интерфейса, связанные с поиском, редактированием, повторением и экспортом.

Текстовый источник является наиболее простым вариантом входной информации. Пользователь вставляет фрагмент текста в приложение, после чего система сохраняет его как источник и использует для генерации карточек. Такой вариант удобен при работе с небольшими заметками, выдержками из статьи, фрагментами конспекта или технического описания.

Файловые источники требуют предварительного извлечения текста. При работе с PDF-документами система получает текстовое содержимое страниц и удаляет лишние элементы. Для DOCX-файлов извлекается текст из структуры документа. Для веб-страниц система получает содержимое страницы и форми-

рует на его основе текстовый источник. Независимо от формата итоговый материал должен быть приведён к единому виду, пригодному для дальнейшей обработки.

Параметры генерации также относятся к входной информации. Пользователь может выбрать источник, указать желаемое количество карточек, задать дополнительные параметры и запустить генерацию. При этом указанное количество карточек рассматривается как ориентир, а не как единственный критерий успешного результата. Сохранению подлежат только карточки, прошедшие проверку качества.

Выходной информацией системы являются данные, полученные после обработки источников и действий пользователя. Основным выходным объектом является карточка запоминания. Она содержит вопрос, ответ, связь с источником, теги, статус повторения, дату создания и служебные параметры. Кроме карточек, к выходной информации относятся списки источников, данные графа знаний, результаты поиска, экспортируемые файлы и сообщения о ходе выполнения операций.

Отдельным видом выходной информации являются экспортируемые файлы. После проверки карточек пользователь может выгрузить результат в один из поддерживаемых форматов: Anki, Quizlet, CSV, PDF или JSON. Экспорт выполняется отдельно от генерации, поэтому система сначала сохраняет универсальные карточки, а затем преобразует их в выбранный формат.

2.1.2 Структура данных информационной системы

Для хранения данных в системе “AI FlashcardGenerator” используется реляционная модель. Такой подход выбран потому, что основные объекты приложения имеют устойчивую структуру и связаны между собой. Колоды содержат источники и карточки, карточки могут иметь теги, статусы повторения и связь с источником, а источники принадлежат определённой колоде. Реляцион-

ная модель позволяет явно описать такие связи и обеспечить целостность данных [9, с. 45], [10, с. 118], [11, с. 64], [15, с. 95].

В качестве системы управления базой данных используется SQLite. Она подходит для локального приложения, так как не требует отдельного серверного процесса и хранит данные в одном файле. Это соответствует назначению “AI FlashcardGenerator”: система должна запускаться на компьютере пользователя и не требовать сложного развёртывания внешней серверной инфраструктуры [38].

Основными таблицами базы данных являются таблицы колод, источников и карточек. Таблица колод хранит сведения о логических наборах материалов. Таблица источников содержит добавленные пользователем материалы и их метаданные. Таблица карточек хранит результаты генерации, ручного добавления или импорта. Между таблицами устанавливаются связи, позволяющие определить, к какой колоде относится источник и с каким источником связана карточка.

Таблица колод содержит идентификатор, название, дату создания и дополнительные параметры. Колода может соответствовать учебной теме, разделу документации, проекту или другому направлению работы пользователя. Наличие колод позволяет не смешивать материалы разных тем и упрощает навигацию по базе знаний.

Таблица источников хранит сведения о материалах, из которых создаются карточки. Для каждого источника сохраняются идентификатор, принадлежность к колоде, название, тип источника, текстовое содержимое, теги, цветовая маркировка и параметры отображения на холсте. Это позволяет использовать источник не только как текст для генерации, но и как самостоятельный объект интерфейса.

Дополнительно для источника сохраняется цветовая маркировка, которая используется при отображении объекта на холсте знаний. Цвет позволяет визуально отличать один источник от другого и быстрее ориентироваться при работе с несколькими материалами в одной колоде. Пользователь может изменить

цвет источника через окно настройки, где задаются оттенок, насыщенность, яркость и HEX-значение.

Таблица карточек является центральной таблицей системы. В ней сохраняются вопрос, ответ, цитата или фрагмент источника, теги, статус, дата повторения, принадлежность к колоде, связь с источником, сведения о модели генерации и координаты отображения.

Такая структура делает карточку универсальным объектом: она может использоваться для повторения, редактирования, поиска, отображения в графе и экспорта.

Поля, связанные с повторением, позволяют системе планировать дальнейшую работу с карточками. Для карточки могут храниться статус, дата следующего повторения, интервал, количество повторений, количество ошибок и дата последнего просмотра. Благодаря этому приложение не только хранит карточки, но и поддерживает процесс их регулярного повторения.

Для уточнения структуры данных основные сущности информационной системы представлены в таблице 2.1; такой способ описания позволяет наглядно показать состав объектов, их назначение и связи между ними [12, с. 73], [19, с. 148], [20, с. 93].

Таблица 2.1 – Основные информационные объекты системы

Объект	Назначение	Основные данные
Колода	Объединение материалов по теме или проекту	Идентификатор, название, дата создания, теги
Источник	Хранение исходного материалапользователя	Идентификатор, тип источника, название, содержимое, теги, координаты отображения
Карточка	Основной элемент повторения и экспорта	Вопрос, ответ, цитата источника, теги, статус, дата повторения

Продолжение таблицы 2.1

Тег	Дополнительная классификация материалов	Название тега, связь с карточками и источниками
Состояние повторения	Планирование показа карточек пользователю	Интервал, дата следующего повторения, количество повторений, статус
Экспортный профиль	Преобразование карточек во внешний формат	Формат экспорта, набор полей, параметры выгрузки

Использование таких объектов позволяет хранить не только сами карточки, но и контекст их создания. Благодаря этому карточка остаётся связанной с исходным материалом и может использоваться в разных режимах: на холсте знаний, в списке, в режиме повторения и при экспорте.

2.1.3 Организация хранения источников, карточек, тегов и данных повторения

Организация хранения данных в “AI FlashcardGenerator” направлена на то, чтобы один и тот же материал мог использоваться в разных режимах без дублирования. Карточки могут отображаться на холсте, в списке, в инспекторе, в режиме повторения и в окне экспорта. При этом исходные данные хранятся централизованно в базе, а интерфейс получает их через серверные API-запросы. При добавлении источника система создаёт запись в таблице источников. В ней сохраняются название, тип источника, текстовое содержимое, теги и параметры отображения. Если источник содержит большой текст, он используется для дальнейшего разбиения на фрагменты и генерации карточек. Исходный текст

не удаляется после генерации, так как он нужен для проверки результата и восстановления контекста.

Карточки создаются в результате генерации, ручного добавления или импорта. При сохранении карточки фиксируется её принадлежность к колоде и связь с источником. Кроме вопроса и ответа, сохраняются теги, статус повторения, дата создания и служебные параметры. Это позволяет отличать созданные карточки, искать их, фильтровать по признакам и использовать в разных сценариях работы.

Теги применяются для дополнительной классификации. Один тег может быть связан с разными карточками и источниками, поэтому он не заменяет колоду, а дополняет её. С помощью тегов пользователь может выделять темы, типы материалов, сложность или направление подготовки.

Данные повторения хранятся вместе с карточкой. Такой подход упрощает выбор карточек для показа пользователю. Система может определить, какие карточки находятся во входящих, какие требуют повторения сегодня, а какие уже отложены на будущие даты. После ответа пользователя параметры карточки обновляются.

Экспортные данные формируются на основе уже сохранённых карточек. Генерация и экспорт разделены: сначала карточка создаётся как универсальный объект, а затем при необходимости преобразуется в формат Anki, Quizlet, CSV, PDF или JSON. Благодаря этому один и тот же набор карточек можно использовать внутри приложения или выгружать во внешние системы.

Таким образом, информационное обеспечение “AI FlashcardGenerator” строится вокруг трёх основных объектов: колоды, источника и карточки. Колода задаёт область хранения, источник содержит исходный материал, а карточка представляет результат обработки. Теги, статусы повторения, связи с источниками и экспортные параметры расширяют эти объекты и позволяют использовать систему как локальную базу знаний для подготовки, повторения и выгрузки карточек.

2.2 Технологическое обеспечение

Технологическое обеспечение информационной системы “AI FlashcardGenerator” определяет набор технологий, программных средств и последовательность операций, с помощью которых выполняется обработка источников, генерация карточек, хранение данных и взаимодействие пользователя с приложением. Для разрабатываемой системы технологическое обеспечение должно поддерживать локальную работу, обработку разных типов материалов, обращение к языковой модели, сохранение результата и отображение данных в интерфейсе.

Система проектируется как локальное веб-приложение. Такой подход позволяет совместить удобство браузерного интерфейса с возможностью запуска серверной части на компьютере пользователя. Пользователь взаимодействует с приложением через веб-страницу, а основные операции выполняются backend-частью: обработка источников, работа с базой данных, генерация карточек, проверка результата, импорт, экспорт и повторение.

Технологический процесс в “AI FlashcardGenerator” можно представить как последовательность этапов. Сначала пользователь добавляет источник или вводит текст. Затем система сохраняет сведения об источнике, извлекает и очищает текст, разбивает его на фрагменты, передаёт подготовленные данные в модуль генерации, получает карточки, проверяет их структуру и сохраняет валидные элементы в базе данных. После этого карточки становятся доступны для просмотра, редактирования, повторения и экспорта.

2.2.1 Обоснование выбора технологического стека

Для реализации серверной части выбран язык программирования Python, документация и экосистема которого позволяют использовать язык для обработки данных, работы с файлами, веб-сервисами и прикладной логикой [35].Он подходит для разработки прикладных информационных систем, так как имеет

большое количество библиотек для обработки текста, работы с файлами, базами данных, веб-серверами и интеграцией моделей машинного обучения. В рамках “AI FlashcardGenerator” Python используется для реализации API, обработки источников, взаимодействия с базой данных, запуска генерации и выполнения служебных операций [35].

В качестве backend-фреймворка используется FastAPI, который поддерживает построение API, обработку запросов и валидацию данных в связке с современными инструментами Python-разработки [36], [37], [40].Для рассматриваемой системы это важно, поскольку клиентская часть должна получать список колод, источников, карточек, статусы генерации, результаты поиска и данные для экспорта через единый набор API-запросов.

Для хранения данных используется SQLite, а работа с таблицами и объектами приложения дополнительно упрощается за счёт применения ORM-подхода [38], [39].Это упрощает запуск системы на компьютере пользователя и соответствует требованию минимальной зависимости от внешней инфраструктуры.

Для работы с базой данных применяется SQLAlchemy. Использование ORM-слоя позволяет описывать основные сущности системы в виде программных моделей и выполнять операции создания, чтения, обновления и удаления данных без ручного написания большого количества SQL-запросов. Это упрощает сопровождение кода и делает структуру данных более понятной.

Клиентская часть реализуется средствами HTML, CSS и JavaScript, которые используются соответственно для структуры страницы, визуального оформления и интерактивного поведения интерфейса [41], [42], [43].HTML используется для разметки интерфейса, CSS — для визуального оформления, JavaScript — для обработки действий пользователя, отправки запросов к backend-части и динамического обновления данных на странице. Такой подход позволяет создать интерфейс без установки отдельного desktop-клиента и использовать браузер как основную среду взаимодействия.

Для генерации карточек используется локальный модуль языковой модели. Такой вариант выбран для снижения зависимости от облачных сервисов и сохранения возможности локальной обработки пользовательских материалов. Backend-часть подготавливает фрагмент текста, передаёт его в модуль генерации, получает структурированный результат, проверяет карточки и сохраняет валидные элементы.

Выбранный технологический стек соответствует назначению системы. Python и FastAPI обеспечивают серверную обработку и API, SQLite подходит для локального хранения данных, SQLAlchemy упрощает работу с базой, HTML, CSS и JavaScript позволяют реализовать пользовательский интерфейс, а локальный модуль языковой модели обеспечивает создание карточек на основе исходных материалов.

2.2.2 Технологический процесс обработки учебных и информационных материалов

Технологический процесс обработки материалов начинается с добавления источника. Пользователь может ввести текст вручную, выбрать файл или добавить другой информационный объект, поддерживаемый системой. После этого приложение создаёт запись об источнике, сохраняет его название, тип, принадлежность к колоде, теги и другие служебные параметры.

Следующим этапом является извлечение содержимого, в том числе из PDF-документов, для чего в системе могут использоваться средства работы с PDF-файлами [44].Если пользователь вводит текст вручную, система сразу использует его как основу для дальнейшей обработки. Если добавляется файл, приложение должно получить из него текстовую часть. Для документов это означает извлечение текста из структуры файла, для PDF — получение текстового содержимого страниц, для веб-страницы — выделение основного текста из загруженного содержимого.

После извлечения выполняется предварительная очистка материала. На этом этапе удаляются лишние переносы строк, технические символы, повторяющиеся пробелы, случайные фрагменты разметки и другие элементы, которые могут ухудшить качество генерации. Цель очистки состоит не в изменении смысла текста, а в приведении материала к более стабильному виду для дальнейшей обработки.

Далее текст разбивается на смысловые фрагменты. Это необходимо потому, что большой источник не всегда удобно передавать в модуль генерации целиком. Разбиение позволяет обрабатывать материал частями, сохранять логическую связность фрагментов и снижать риск потери важных сведений. Размер фрагмента должен быть достаточным для понимания контекста, но не слишком большим, чтобы языковая модель могла сформировать конкретные вопросы и ответы.

После подготовки фрагментов система может запускать генерацию карточек. Пользователь выбирает источник, при необходимости задаёт параметры генерации и запускает процесс. Backend-часть формирует запрос к языковой модели, передаёт подготовленный текст и ожидает структурированный ответ. Результат генерации не сохраняется автоматически без проверки.

Полученный ответ проходит проверку. Система должна убедиться, что карточка содержит вопрос и ответ, не является пустой, не повторяет уже сохранённый элемент и имеет смысловую связь с исходным материалом. Если часть карточек не проходит проверку, она не сохраняется в базу данных. Такой подход позволяет избежать накопления некорректных, дублирующихся или неполных карточек.

После проверки валидные карточки сохраняются в базе данных. Для каждой карточки фиксируются вопрос, ответ, источник, колода, теги, статус повторения и служебные параметры. С этого момента карточка становится доступной в интерфейсе: её можно просмотреть, отредактировать, переместить, отметить тегами, использовать в режиме повторения или экспортировать.

2.2.3 Технологический процесс генерации карточек запоминания

Процесс генерации карточек является центральной технологической операцией системы “AI FlashcardGenerator”. Его задача заключается в том, чтобы преобразовать подготовленный текстовый фрагмент в набор карточек запоминания, пригодных для дальнейшей проверки и использования. Общая последовательность этапов генерации карточек представлена на рисунке 2.1.

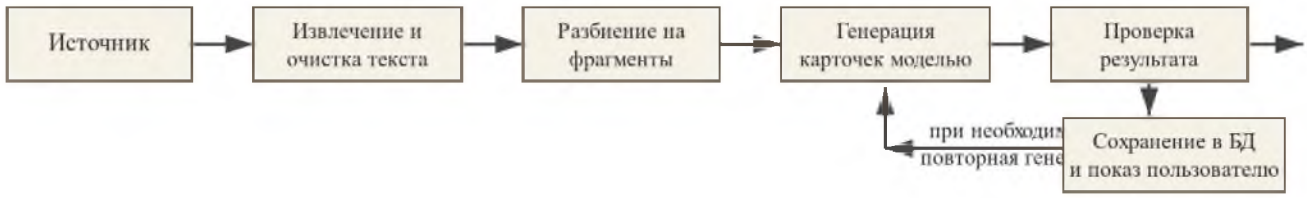


Рисунок 2.1 – Технологический процесс генерации карточек запоминания

Как видно из рисунка 2.1, генерация карточек включает несколько последовательных этапов. Сначала пользователь выбирает источник или фрагмент материала и задаёт параметры генерации. После этого backend-часть подготавливает текст, формирует запрос к локальному модулю языковой модели и получает структурированный результат.

Полученные карточки не сохраняются автоматически без проверки. Система анализирует наличие обязательных полей, исключает пустые, повторяющиеся или неполные элементы и связывает валидные карточки с исходным источником и выбранной колодой. Такой подход позволяет сохранить качество базы карточек и обеспечить возможность последующей проверки пользователем.

После сохранения карточки становятся доступны в интерфейсе приложения. Пользователь может просмотреть вопросы и ответы, отредактировать формулировки, добавить теги, перейти к повторению или выполнить экспорт во внешний формат. Таким образом, процесс генерации в системе включает не

только обращение к языковой модели, но и подготовку текста, проверку результата, сохранение данных и отображение карточек в пользовательском интерфейсе.

2.3 Техническое обеспечение

Техническое обеспечение информационной системы “AI FlashcardGenerator” включает аппаратные и системные средства, необходимые для запуска приложения, хранения данных, обработки источников и выполнения генерации карточек. Так как система проектируется как локальное веб-приложение, основные вычислительные операции выполняются на компьютере пользователя.

Локальный характер системы определяет требования к техническому обеспечению. Приложение не требует отдельного серверного оборудования, так как backend-часть запускается на том же компьютере, с которого пользователь работает с интерфейсом. Взаимодействие с системой выполняется через браузер, а хранение данных осуществляется в локальной базе SQLite. Такой подход упрощает установку и эксплуатацию системы.

2.3.1 Обоснование проектных решений по техническому обеспечению

При выборе технического обеспечения учитывались локальный режим работы системы, необходимость обработки текстовых материалов, хранение пользовательских данных и выполнение генерации карточек. Информационная система «AI FlashcardGenerator» проектируется как локальное веб-приложение, поэтому для её работы не требуется отдельный сервер или специализированная инфраструктура.

Основным техническим средством является персональный компьютер или ноутбук пользователя. На нём размещаются серверная часть приложения,

браузерный интерфейс, локальная база данных SQLite, загруженные источники и компоненты генерации карточек. Пользователь запускает backend-часть приложения и открывает интерфейс через веб-браузер, что позволяет работать с системой без установки отдельного desktop-клиента.

Для выполнения основных функций достаточно компьютера с современным многоядерным процессором, оперативной памятью, локальным накопителем и установленным веб-браузером. Операции просмотра карточек, поиска, экспорта и работы с базой данных не требуют высокопроизводительного оборудования. Дополнительная нагрузка возникает при генерации карточек с использованием локального модуля языковой модели, поэтому для комфортной работы рекомендуется достаточный объём оперативной памяти и свободного места на накопителе.

Постоянное подключение к интернету не является обязательным условием эксплуатации системы. Оно может потребоваться для первоначальной установки зависимостей или обновления компонентов, однако основная работа с сохранёнными источниками, карточками, повторением и экспортом выполняется локально. Такой подход соответствует назначению системы и снижает зависимость пользователя от внешних сервисов.

2.3.2 Комплекс технических средств

Комплекс технических средств для работы информационной системы “AI Flashcard Generator” включает следующие элементы:

– персональный компьютер или ноутбук пользователя;

– центральный процессор, обеспечивающий выполнение backend-части приложения и операций обработки текста;

– оперативную память, необходимую для работы приложения, браузера и локальной языковой модели;

– локальный накопитель для хранения файлов приложения, базы данных, источников, моделей и экспортируемых результатов;

– монитор, клавиатуру и мышь или тачпад для взаимодействия с интерфейсом;
– веб-браузер для открытия клиентской части приложения;
– сетевой адаптер для установки компонентов и, при необходимости, обновления зависимостей.

Минимальная конфигурация технических средств должна обеспечивать запуск Python, работу браузера и выполнение основных операций с базой данных. Для комфортной работы рекомендуется использовать компьютер с многоядерным процессором, объёмом оперативной памяти не менее 8 ГБ и достаточным свободным пространством на накопителе. При активном использовании локальной языковой модели желательно иметь больший объём оперативной памяти и более производительный процессор.

В процессе эксплуатации пользователь запускает серверную часть приложения на локальном компьютере. После запуска backend-часть принимает запросы от клиентского интерфейса, обращается к базе данных, обрабатывает источники, выполняет генерацию карточек и возвращает результат в браузер. Клиентская часть отображает колоды, источники, карточки, граф знаний, инспектор объектов, режим повторения и окна экспорта.

Такой состав технических средств соответствует назначению разрабатываемой системы. Он позволяет использовать “AI Flashcard Generator” без отдельного сервера, сохранять данные локально, выполнять обработку материалов на стороне пользователя и обеспечивать доступ к функциям приложения через обычный браузер.

2.4 Программное обеспечение задачи

Программное обеспечение информационной системы “AI Flashcard Generator” включает серверную часть, клиентскую часть, локальную базу данных, модули обработки источников, модуль генерации карточек, средства проверки результата, механизм интервального повторения и средства экспорта. Основная задача программного обеспечения состоит в том, чтобы обеспечить

полный цикл работы пользователя с материалом: от добавления источника до получения набора карточек, их повторения и выгрузки во внешний формат.

Система реализуется как локальное веб-приложение. Серверная часть запускается на компьютере пользователя и предоставляет API для клиентского интерфейса. Клиентская часть открывается в браузере и взаимодействует с backend-частью через HTTP-запросы. Такой подход позволяет использовать приложение без отдельной серверной инфраструктуры и при этом сохранить удобство веб-интерфейса.

2.4.1 Обоснование проектных решений по программному обеспечению

При выборе программных средств учитывались несколько требований: локальный режим работы, возможность обработки текстовых источников, хранение данных между запусками, работа с языковой моделью, наличие интерфейса для просмотра и редактирования карточек, а также возможность экспорта результата.

Для серверной части выбран язык программирования Python. Он подходит для реализации прикладной логики, обработки текста, работы с файлами, базой данных и интеграции с моделями машинного обучения. В системе “AI Flashcard Generator” Python используется для обработки API-запросов, извлечения и очистки текста, запуска генерации карточек, проверки результата, сохранения данных и формирования экспортируемых файлов [35].

FastAPI позволяет создавать маршруты API, принимать данные от клиентской части, возвращать ответы в формате JSON и разделять функции приложения на отдельные обработчики. Это удобно для системы, в которой клиентский интерфейс должен получать данные о колодах, источниках, карточках, статусах повторения, результатах поиска и процессе генерации [36].

SQLite это база данных, которая не требует отдельного серверного процесса и подходит для локального приложения. Данные хранятся в файле на компьютере пользователя, что упрощает установку и эксплуатацию системы.

Для работы с базой данных применяется SQLAlchemy, позволяющий описывать основные сущности приложения в виде моделей и выполнять операции с ними через программный код [38].

HTML определяет структуру страницы, CSS отвечает за оформление интерфейса, JavaScript обеспечивает интерактивность, отправку запросов к серверу, обновление данных без перезагрузки страницы, работу с карточками, источниками, поиском, графом знаний и режимом повторения.

Для генерации карточек используется локальный модуль языковой модели. Такой вариант выбран для снижения зависимости от облачных сервисов и сохранения возможности локальной обработки пользовательских материалов. Backend-часть подготавливает фрагмент текста, передаёт его в модуль генерации, получает структурированный результат, проверяет карточки и сохраняет валидные элементы.

Выбранные программные средства соответствуют назначению системы. Python и FastAPI обеспечивают серверную логику, SQLite и SQLAlchemy отвечают за хранение данных, HTML, CSS и JavaScript формируют пользовательский интерфейс, а локальная языковая модель используется для создания карточек на основе исходных материалов.

2.4.2 Архитектура информационной системы

Архитектура “AI FlashcardGenerator” построена по принципу разделения клиентской и серверной частей, что соответствует распространённым подходам к проектированию программных систем и разделению ответственности между компонентами [16, с. 171], [17, с. 209], [18, с. 43].Клиентская часть отвечает за отображение интерфейса и обработку действий пользователя. Серверная часть выполняет основную прикладную логику: работу с базой данных, обработку источников, генерацию карточек, проверку результата, повторение и экспорт.Общая структура взаимодействия компонентов информационной системы представлена на рисунке 2.2.

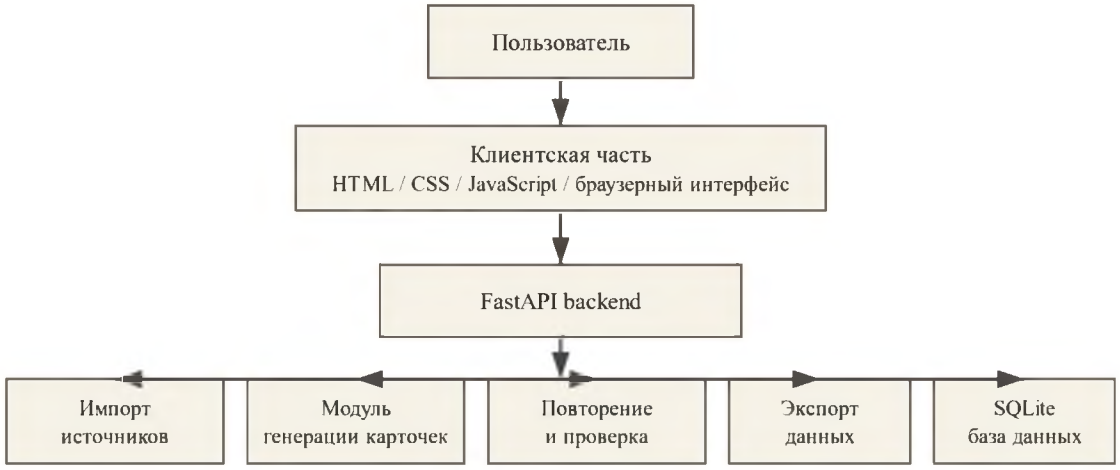


Рисунок 2.2 – Архитектура информационной системы AI FlashcardGenerator

Согласно рисунку 2.2, пользователь взаимодействует с системой через браузерный интерфейс, клиентская часть обменивается данными с FastAPI-сервером, а серверные модули выполняют обработку источников, генерацию карточек, повторение и экспорт с использованием локальной базы данных.

Взаимодействие между частями системы выполняется через локальный веб-сервер. Пользователь открывает интерфейс приложения в браузере. При выполнении действия, например добавлении источника или запуске генерации, клиентская часть отправляет запрос на backend. Сервер обрабатывает запрос, выполняет необходимые операции и возвращает ответ в формате JSON. После этого интерфейс обновляет отображаемые данные.

В архитектуре системы можно выделить несколько основных уровней; такое разбиение упрощает описание структуры приложения, распределение функций между компонентами и дальнейшее сопровождение программного продукта [18, с. 89], [19, с. 211], [20, с. 146].Первый уровень — пользовательский интерфейс. Он включает холст знаний, список карточек, панели источников, инспектор объектов, поиск, командное меню, режим повторения и окна экспорта. Второй уровень — API backend-части. Он принимает запросы от интерфейса и передаёт их соответствующим модулям. Третий уровень — прикладная логика, где выполняется обработка источников, генерация карточек,

фильтрация, проверка качества, повторение и экспорт. Четвёртый уровень — хранение данных в SQLite.

Основными информационными объектами системы являются колоды, источники и карточки. Колода объединяет материалы по теме или проекту. Источник хранит добавленный материал и его метаданные. Карточка содержит вопрос, ответ, связь с источником, теги, статус повторения и служебные параметры. Такая структура позволяет использовать карточки в разных режимах: на холсте, в списке, в режиме повторения и при экспорте.

Архитектура системы также предусматривает модульную организацию кода. Отдельные файлы и модули отвечают за запуск приложения, настройки языковой модели, генерацию карточек, обработку русского текста, проверку качества, интервальное повторение, работу с медиа, экспортные профили, миграции базы данных и форматы карточек. Такое разделение упрощает сопровождение системы и позволяет дорабатывать отдельные функции без полного изменения приложения.

2.4.3 Описание серверной части приложения

Серверная часть является центральным элементом системы “AI FlashcardGenerator”. Она принимает запросы от клиентского интерфейса, выполняет операции с базой данных, запускает обработку источников, обращается к модулю генерации и возвращает результат пользователю.

Основной файл серверной части отвечает за запуск приложения, регистрацию маршрутов API и связь между модулями. Через API выполняются операции создания и получения колод, добавления источников, создания карточек, запуска генерации, поиска, редактирования, удаления объектов, работы с повторением и экспорта данных. Такой подход позволяет клиентской части не работать напрямую с базой данных, а обращаться к серверу через контролируемые маршруты.

Модуль конфигурации языковой модели содержит параметры, необходимые для локального инференса. В нём задаются настройки подключения к модели, параметры генерации и служебные значения, влияющие на получение результата. Это позволяет отделить настройки модели от основной логики приложения и упростить дальнейшее изменение способа генерации.

Модули обработки текста выполняют очистку и подготовку исходных материалов. Они удаляют лишние символы, нормализуют пробелы и переносы строк, помогают разбивать текст на фрагменты и приводят материал к виду, пригодному для передачи в языковую модель. Для русскоязычных материалов дополнительно учитываются особенности формулировок, чтобы вопросы и ответы были пригодны для использования в карточках.

Модуль генерации карточек отвечает за подготовку запроса к языковой модели и получение ответа. Он передаёт модели фрагмент текста и инструкцию сформировать карточки в заданной структуре. После получения ответа данные передаются на разбор и проверку. Это важно, потому что результат языковой модели может содержать неполные, повторяющиеся или неправильно оформленные элементы.

Модуль проверки качества карточек используется для отбраковки слабых результатов. Он проверяет наличие вопроса и ответа, длину полей, отсутствие пустых значений, повторов и слишком шаблонных формулировок. Если карточка не проходит проверку, она не сохраняется в базе данных. Такой подход позволяет избежать ситуации, когда система формально создаёт карточки, но часть из них не имеет учебной ценности.

Отдельный модуль отвечает за интервальное повторение. Он хранит и обновляет параметры карточек, связанные со статусом повторения, датой следующего показа, числом повторений и ошибками пользователя. Благодаря этому карточки используются не только как статичный список, но и как элементы системы повторения.

Серверная часть также содержит средства импорта и экспорта. Импорт позволяет добавлять материалы или карточки из внешних источников, а экс-

порт — выгружать готовые карточки в распространённые форматы. Экспорт выполняется после проверки и сохранения карточек, поэтому пользователь выгружает уже подготовленный результат, а не необработанный ответ языковой модели.

2.4.4 Описание клиентской части приложения

Клиентская часть “AI FlashcardGenerator” предназначена для взаимодействия пользователя с системой и проектируется с учётом принципов удобства, понятности и человеко-ориентированного интерфейса [21, с. 52], [22, с. 72], [23, с. 11], [24].Она отображает данные, принимает действия пользователя, отправляет запросы к backend-части и обновляет интерфейс после получения ответа. Основная задача клиентской части — сделать работу с источниками и карточками наглядной и удобной.

Интерфейс приложения включает несколько основных областей. Центральная область используется для визуального представления материалов и карточек. В ней пользователь может видеть источники, карточки и связи между ними. Такое представление помогает воспринимать материал не как обычный список, а как структуру, в которой карточки связаны с исходными источниками.

Боковые панели используются для управления объектами и просмотра подробной информации. Левая часть интерфейса может содержать список колод, источников, фильтры или элементы навигации. Правая часть используется как инспектор объекта: в ней отображаются данные выбранной карточки или источника, поля для редактирования, теги, статус и дополнительные действия.

Для работы с большим количеством данных в интерфейсе предусмотрены поиск и фильтрация. Пользователь может быстро найти нужную карточку, источник или группу элементов по тексту, тегам, статусу или принадлежности к колоде. Это особенно важно при накоплении большой базы карточек.

Клиентская часть также обеспечивает запуск генерации карточек. Пользователь выбирает источник, задаёт параметры, запускает процесс и получает результат. После генерации карточки отображаются в интерфейсе, где их можно проверить, отредактировать, удалить, пометить тегами или отправить в повторение.

Режим повторения является отдельным пользовательским сценарием. В нём карточка показывается пользователю в формате вопроса и ответа. Пользователь оценивает результат, после чего система обновляет параметры повторения. Такой режим позволяет использовать приложение не только для генерации и хранения карточек, но и для регулярной работы с ними.

Окна экспорта позволяют выгрузить подготовленные карточки во внешний формат. Пользователь выбирает нужный профиль экспорта, после чего система формирует файл на основе сохранённых карточек. Это позволяет продолжить работу с материалом в других системах, например Anki, Quizlet или табличных редакторах.

2.4.5 Описание модуля генерации карточек

Модуль генерации карточек является ключевой частью системы, так как именно он преобразует исходный текст в набор вопросов и ответов. Его работа начинается после того, как пользователь выбрал источник и запустил генерацию.

Сначала backend-часть получает текст источника и выполняет его подготовку. Материал очищается от лишних символов, разбивается на фрагменты и приводится к виду, удобному для анализа языковой моделью. Если источник слишком большой, он обрабатывается частями. Это позволяет сохранить контекст и не передавать в модель чрезмерно длинный текст.

Затем формируется запрос к языковой модели, а ожидаемая структура ответа должна быть пригодна для последующей программной проверки и валидации данных [40].В запрос включается фрагмент исходного материала и ин-

струкция создать карточки в заданной структуре. Для системы важно, чтобы модель вернула не произвольный текст, а элементы, которые можно разобрать программно: вопрос, ответ, при необходимости тип карточки, цитату, тег или краткое пояснение.

После получения ответа выполняется разбор результата. Если модель вернула карточки в корректной структуре, они передаются на проверку качества.

Если структура нарушена, система не должна сохранять такой ответ как готовый результат.

Это защищает базу данных от некорректных записей.

Проверка качества включает несколько условий. Карточка должна содержать непустой вопрос и непустой ответ. Вопрос не должен быть слишком общим, а ответ не должен состоять из одного слова, если этого не требует смысл материала.

Также проверяется отсутствие повторов и слишком шаблонных формулировок. При наличии связи с источником карточка сохраняется вместе с указанием исходного материала.

Важной особенностью модуля является то, что желаемое количество карточек не является единственным показателем успешной генерации. Если пользователь указал определённое число карточек, система использует это значение как ориентир.

Однако сохранению подлежат только те элементы, которые прошли проверку. Недостающие карточки не должны заменяться заранее прописанными шаблонами, так как это снижает качество результата.

После проверки валидные карточки сохраняются в базе данных и отображаются в интерфейсе.

Пользователь может просмотреть результат, отредактировать формулировки, удалить лишние элементы, добавить теги и перейти к повторению или экспорту.

2.4.6 Описание модулей импорта, экспорта и интервального повторения

Модули импорта, экспорта и интервального повторения расширяют основные функции “AI Flashcard Generator” и позволяют использовать систему не только как генератор карточек, но и как инструмент дальнейшей работы с ними.

Модуль импорта предназначен для добавления материалов и карточек из внешних источников.

Пользователь может добавить текстовый материал, документ или набор карточек, после чего система преобразует данные к внутреннему виду. При импорте важно сохранить структуру объекта: название, текст, принадлежность к колоде, теги и другие параметры.

Модуль экспорта предназначен для выгрузки уже подготовленных карточек в форматы, которые могут использоваться во внешних системах повторения и учебных сервисах [30], [31].

Экспорт выполняется на основе сохранённых данных, а не на основе необработанного ответа языковой модели. Пользователь может выбрать нужный формат, после чего система формирует файл для дальнейшего использования. Поддержка разных форматов делает систему более гибкой, так как пользователь может работать с карточками внутри приложения или переносить их во внешние сервисы.

Экспортные профили позволяют учитывать требования разных форматов. Например, для Anki может потребоваться один порядок полей, для Quizlet — другой, для CSV — табличное представление, для JSON — структурированный набор данных. Наличие профилей позволяет не менять внутреннюю структуру карточек, а преобразовывать её только на этапе выгрузки.

Модуль интервального повторения отвечает за планирование показа карточек. Каждая карточка имеет статус и параметры повторения.

После ответа пользователя система обновляет состояние карточки: переносит её на ближайшее повторение, увеличивает интервал или возвращает

раньше при ошибке. Это позволяет использовать карточки не только для хранения, но и для регулярного закрепления материала.

В режиме повторения пользователь видит вопрос, пытается вспомнить ответ, затем открывает обратную сторону карточки и оценивает результат.

На основе этой оценки система изменяет параметры карточки. Такой подход помогает уделять больше внимания трудным элементам и реже показывать уже освоенные карточки.

Таким образом, программное обеспечение “AI FlashcardGenerator” включает взаимосвязанные модули, обеспечивающие полный цикл работы с карточками.

Серверная часть выполняет обработку данных и генерацию, клиентская часть обеспечивает взаимодействие пользователя с системой, база данных хранит материалы и результаты, а дополнительные модули импорта, экспорта и повторения позволяют использовать созданные карточки в разных сценариях.

2.5 Руководство пользователя

Руководство пользователя описывает порядок работы с информационной системой “AI Flashcard Generator” и основные элементы её интерфейса.

Приложение предназначено для локальной работы с источниками информации, генерации карточек запоминания, редактирования результата, повторения и экспорта готовых карточек во внешние форматы.

Перед началом работы пользователь запускает серверную часть приложения на локальном компьютере.

После запуска система становится доступна через браузер по локальному адресу.

Пользователь открывает интерфейс приложения и получает доступ к основным функциям: созданию колод, добавлению источников, генерации карточек, просмотру результата, редактированию, повторению и экспорту.

2.5.1 Описание интерфейса

Интерфейс “AI Flashcard Generator” построен таким образом, чтобы пользователь мог работать с материалами в нескольких представлениях. Центральная часть экрана используется для отображения графа знаний или рабочей области, где могут располагаться источники и карточки. Такое представление позволяет видеть связь между исходным материалом и созданными карточками.Общий вид главного экрана приложения представлен на рисунке 2.3.

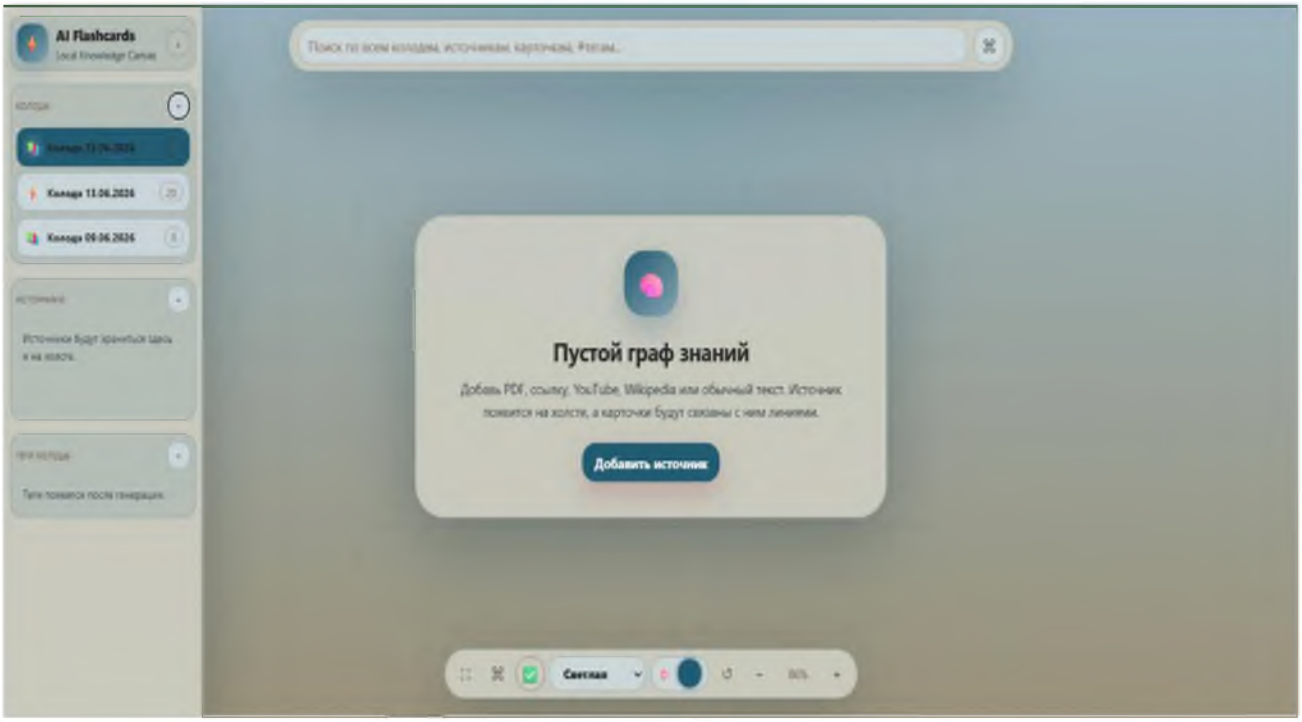


Рисунок 2.3 – Главный экран приложения AI FlashcardGenerator

В верхней части интерфейса размещаются основные элементы управления. К ним относятся строка поиска, кнопки вызова команд, элементы управления текущей колодой и действия, связанные с добавлением источников или запуском генерации. Поиск позволяет быстро находить карточки, источники и другие элементы базы знаний.

Левая панель используется для навигации по материалам. В ней могут отображаться колоды, источники, фильтры, списки объектов или дополнительные элементы управления. С помощью этой панели пользователь выбирает нужную область работы и переходит между наборами карточек.

При выборе источника в интерфейсе открывается панель управления выбранным объектом. В ней отображаются параметры генерации карточек, выбор модели, поле пользовательского пожелания, тип карточек, язык, теги, а также действия для создания карточек, повторения, экспорта и удаления источника. Панель управления источником представлена на рисунке 2.4.

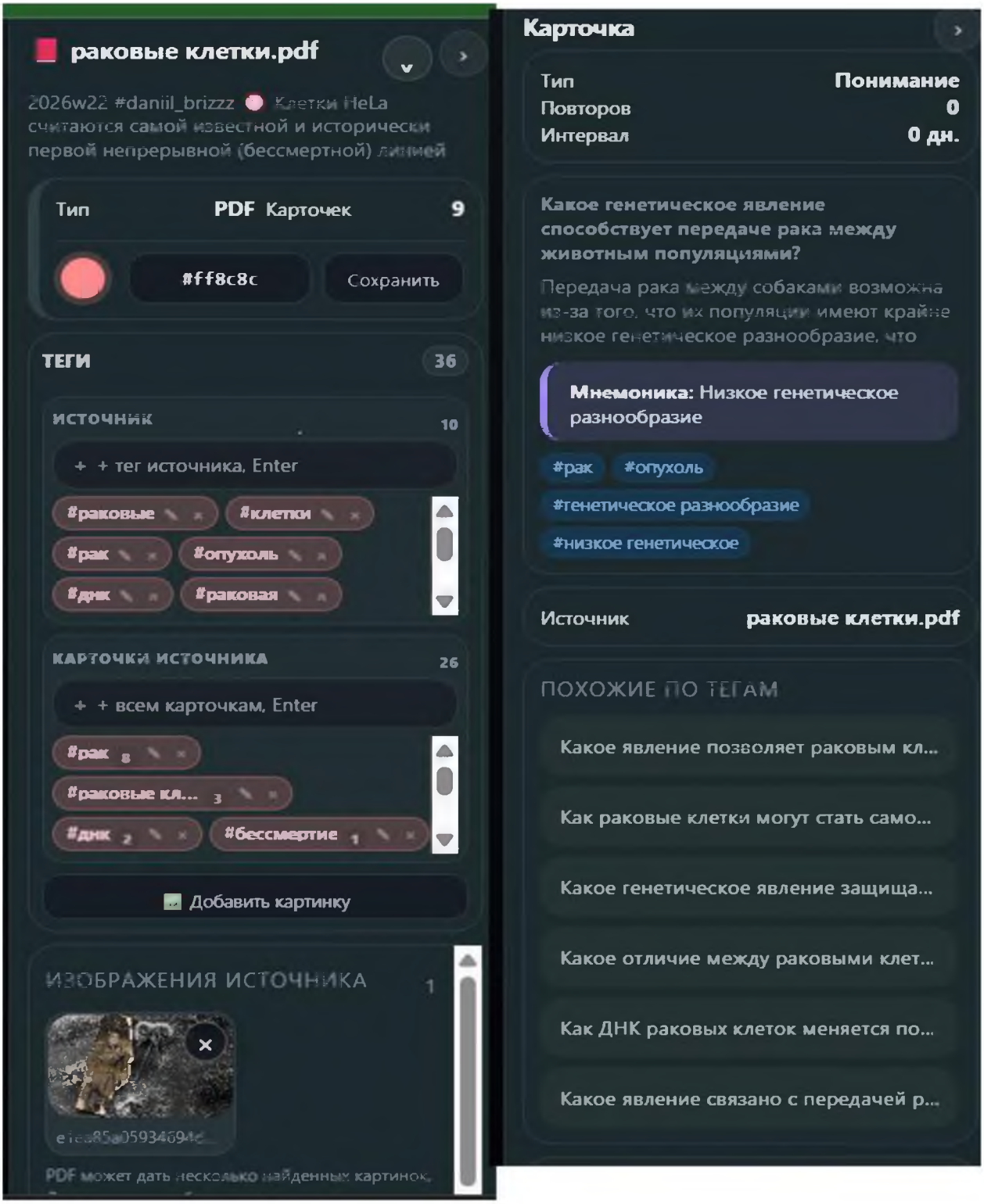


Рисунок 2.4 – Панель управления выбранным источником

Правая панель выполняет роль инспектора объекта. Если пользователь выбирает карточку или источник, в этой области отображаются подробные сведения: название, текст, вопрос, ответ, теги, статус повторения, связь с источником и доступные действия. Через инспектор можно редактировать данные, удалять лишние элементы, изменять теги и уточнять параметры объекта.

Для работы с карточками используется список карточек и режим просмотра. В списке пользователь может увидеть созданные элементы, проверить вопросы и ответы, отредактировать формулировки или удалить неудачные карточки. Такой режим удобен после генерации, когда требуется быстро оценить качество результата.

Отдельным элементом интерфейса является режим повторения. В нём карточка показывается пользователю как вопрос. После попытки вспомнить ответ пользователь открывает обратную сторону карточки и оценивает результат. На основе этой оценки система изменяет состояние карточки и определяет дальнейший порядок её повторения.

В интерфейсе также предусмотрены функции экспорта. Пользователь выбирает готовые карточки, указывает формат выгрузки и получает файл для дальнейшего использования. Экспорт может применяться для переноса карточек в другие системы повторения или для сохранения результата в универсальном формате.

2.5.2 Порядок работы с системой

Работа с системой начинается с создания или выбора колоды. Колода используется для объединения материалов по теме, дисциплине, проекту или другому признаку. После выбора колоды пользователь может добавлять источники и создавать карточки внутри выбранной области.

Следующим этапом является добавление источника. Пользователь может ввести текст вручную, добавить документ или использовать другой поддерживаемый тип материала. После добавления источник сохраняется в системе и

отображается в интерфейсе.

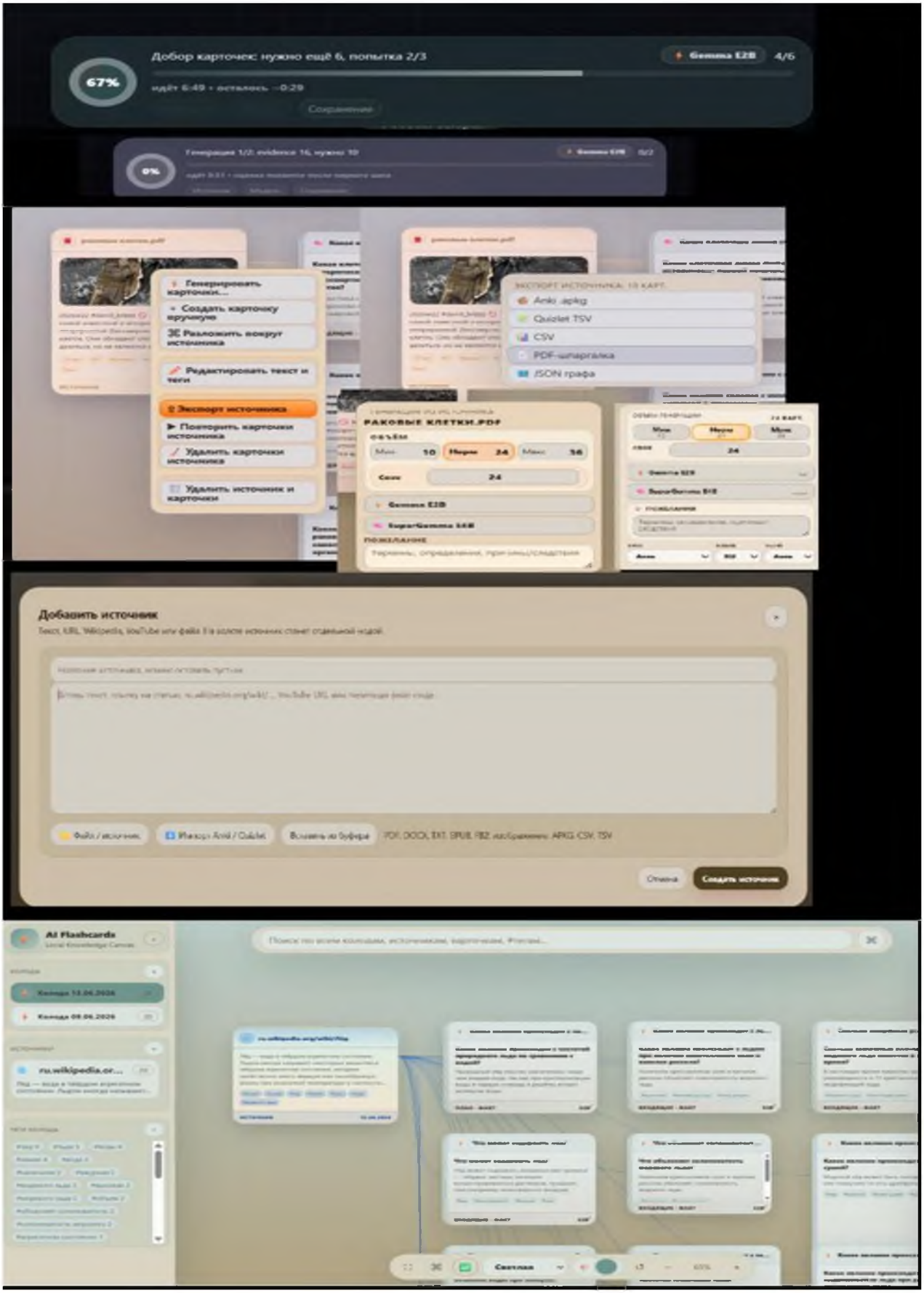


Рисунок 2.5 – Окно генерации карточек из выбранного источника

После добавления источника пользователь запускает генерацию карточек. Для этого выбирается нужный источник и задаются параметры генерации, например желаемое количество карточек. Система подготавливает текст, передаёт его в модуль генерации и получает первичный набор карточек. Полученный результат проходит проверку, после чего валидные карточки сохраняются в базе данных.Пример окна генерации карточек из источника представлен на рисунке 2.5.

На рисунке 2.5 показано, что пользователь может управлять параметрами генерации до запуска процесса. Это позволяет адаптировать создание карточек под объём источника и требуемый формат результата.

После генерации пользователь просматривает созданные карточки. На этом этапе можно проверить вопросы и ответы, отредактировать формулировки, добавить теги и убедиться, что карточки связаны с исходным источником.Такой просмотр позволяет пользователю уточнить формулировки карточек, добавить теги и подготовить материал к повторению или экспорту.

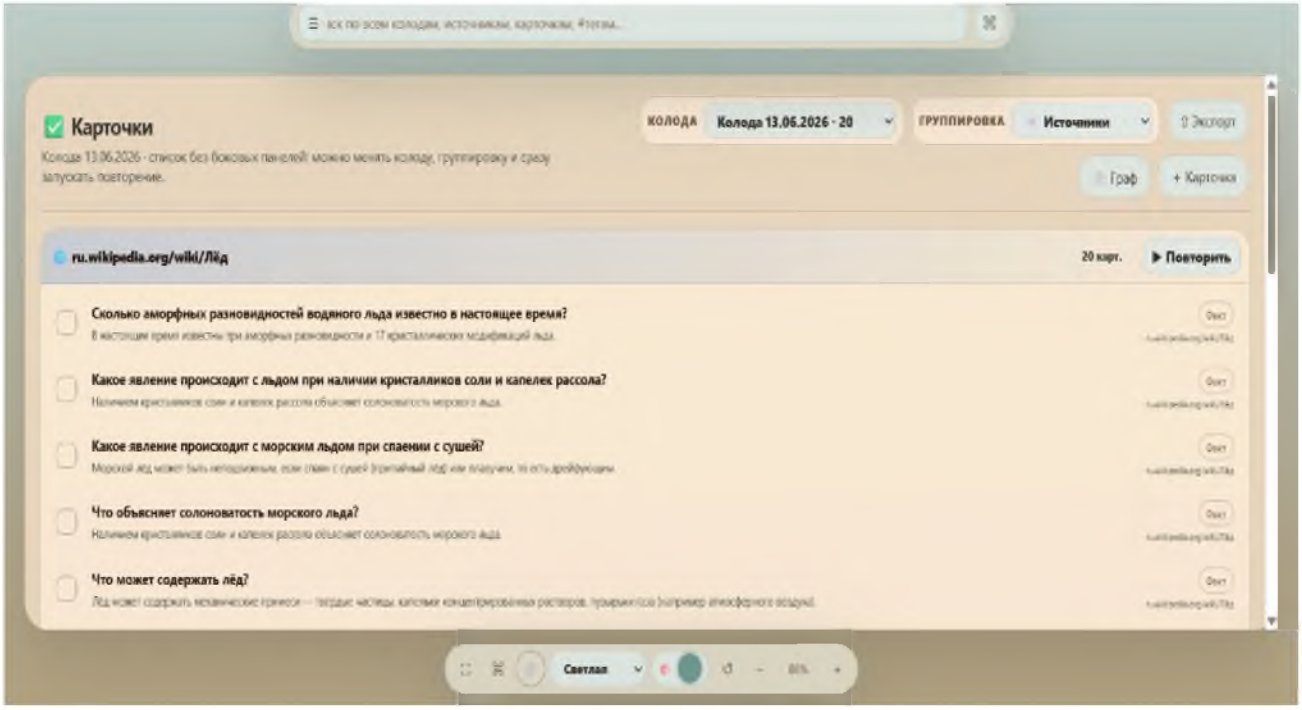


Рисунок 2.6 – Списочный режим просмотра карточек

После формирования карточек пользователь может перейти к их просмотру в списочном режиме. В этом режиме карточки отображаются в виде пе-

речня, могут группироваться по источникам, а также использоваться для повторения и экспорта. Пример списочного режима представлен на рисунке 2.6.

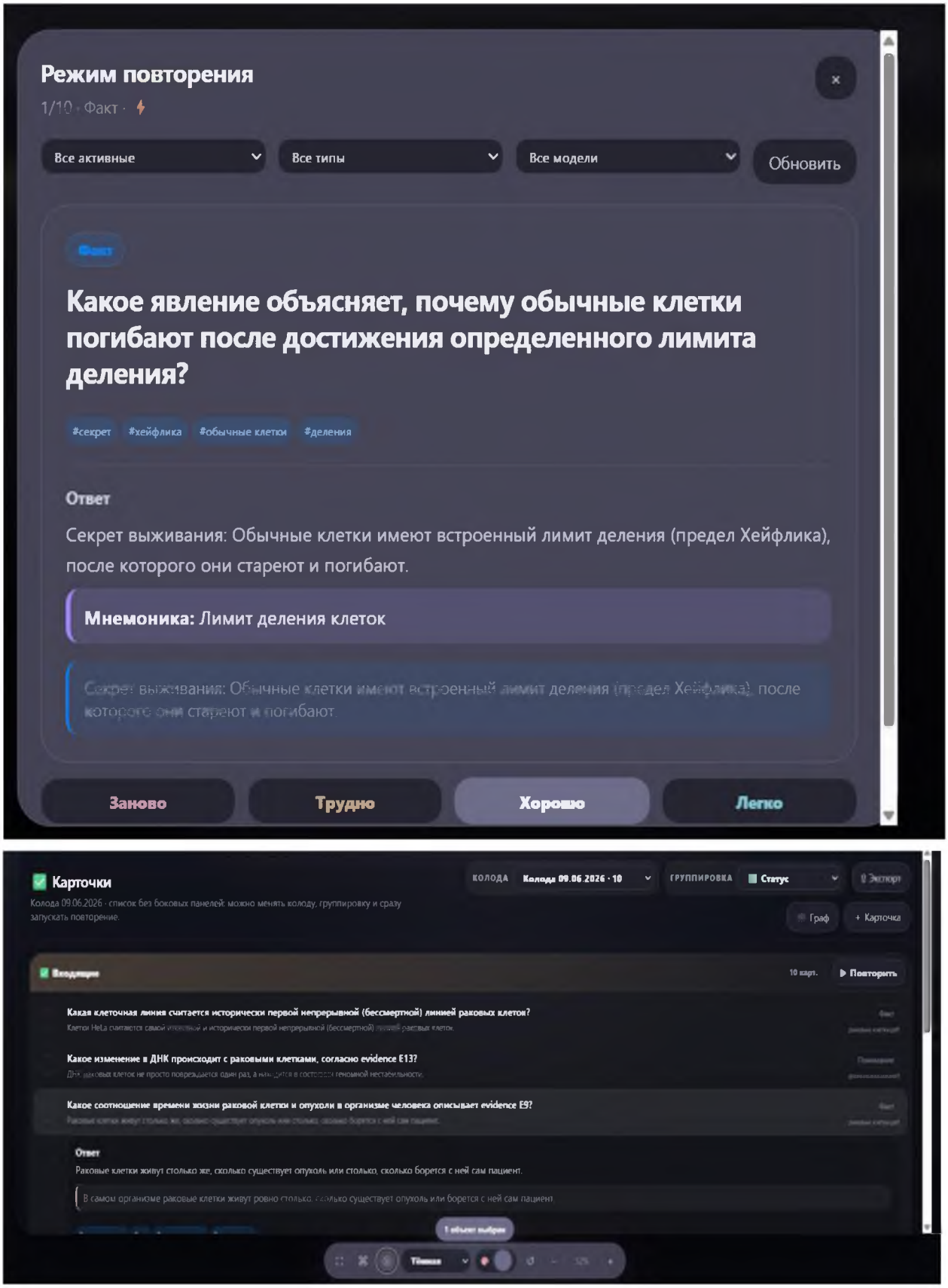


Рисунок 2.7 – Режим повторения карточек

Когда карточки проверены, пользователь может перейти к повторению. В режиме повторения система показывает вопрос, затем пользователь открывает ответ и оценивает, насколько хорошо он вспомнил материал.

После оценки система обновляет статус карточки и дату следующего повторения. Это позволяет использовать приложение не только для генерации, но и для закрепления материала. Пример режима повторения карточек представлен на рисунке 2.7.

При необходимости пользователь может выполнить поиск или фильтрацию. Поиск помогает быстро найти нужную карточку, источник или фрагмент материала.

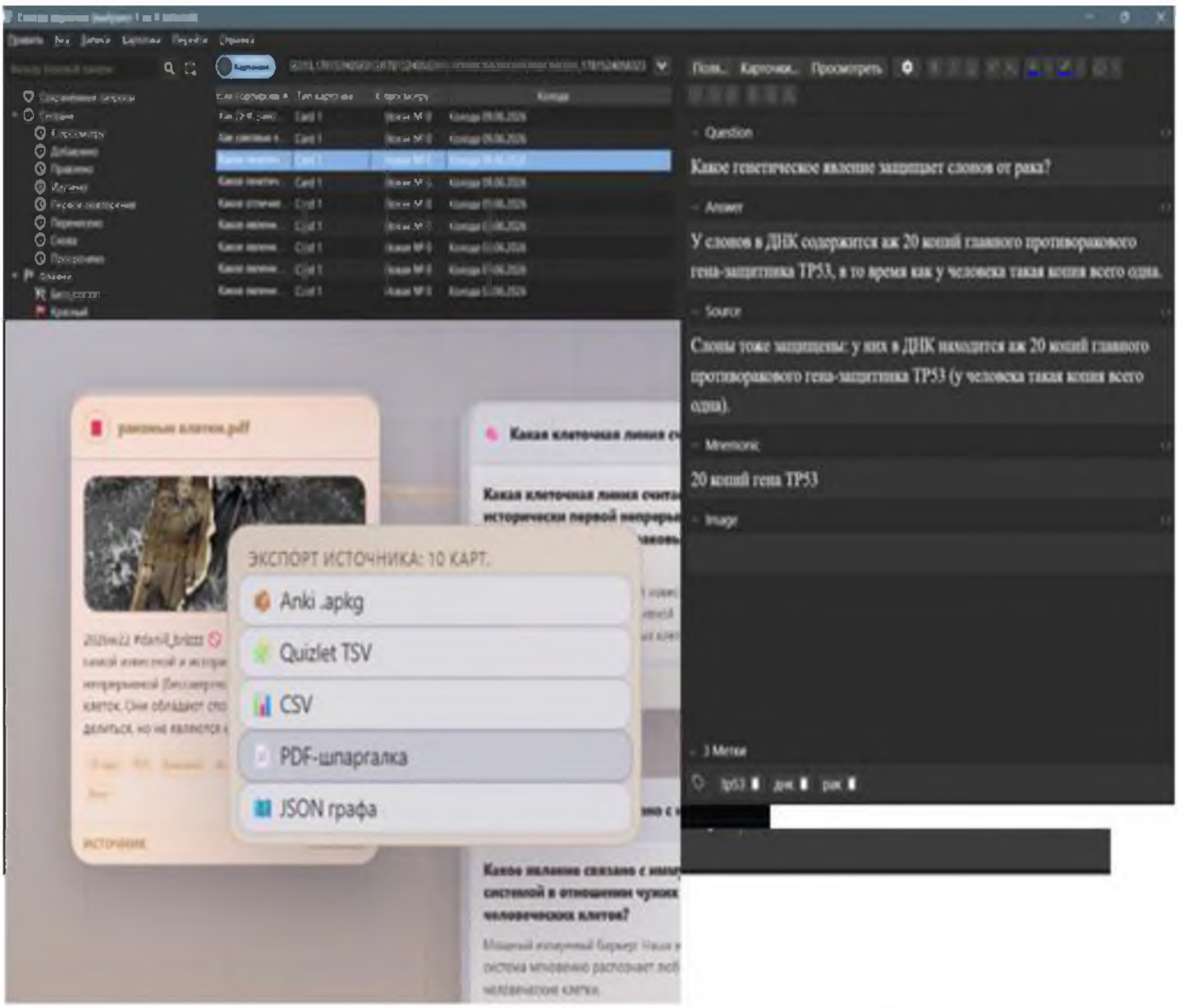
Фильтрация может использоваться по тегам, статусам, колодам или другим признакам. Это особенно важно при накоплении большого количества карточек.

Последним этапом работы может быть экспорт. Пользователь выбирает подготовленные карточки и формат выгрузки. Система формирует файл на основе сохранённых данных.

Экспорт позволяет использовать результат в других приложениях, передавать карточки между устройствами или сохранять их как отдельный набор.Окно экспорта карточек во внешние форматы представлено на рисунке 2.8.

Как видно из рисунка 2.8, пользователь может выгрузить готовые карточки во внешние форматы, включая PDF, Anki, CSV или JSON. Это позволяет использовать созданные материалы как внутри приложения, так и во внешних системах повторения.

В случае ошибки генерации или некорректного результата пользователь может повторить генерацию, изменить исходный материал, уменьшить объём текста или вручную отредактировать полученные карточки. Если языковая модель сформировала меньше карточек, чем было указано пользователем, это не считается критической ошибкой: система сохраняет только те элементы, которые прошли проверку качества.



1. Какое изменение в ДНК происходит с раковыми клетками, согласно evidence E13?
Ответ: ДНК раковых клеток не просто повреждается один раз, а находится в состоянии геномной нестабильности.

Источник: У раковых клеток ДНК не просто «сломалась» один раз — теперь она находится в состоянии геномной нестабильности.
Подсказка: Состояние ДНК раковых клеток

2. Какое соотношение времени жизни раковой клетки и опухоли в организме человека описывает evidence E9?
Ответ: Раковые клетки живут столько же, сколько существует опухоль или столько, сколько борется с ней сам пациент.

Источник: В самом организме раковые клетки живут ровно столько, сколько существует опухоль или борется с ней сам пациент.
Подсказка: Сравнение продолжительности жизни раковой клетки и опухоли

3. Какое явление связано с иммунной системой в отношении чужих человеческих клеток?
Ответ: Мощный иммунный барьер: Наша иммунная система мгновенно распознает любые чужие человеческие клетки.

Источник: Мощный иммунный барьер: Наша иммунная система мгновенно распознает любые чужие человеческие клетки.
Подсказка: Иммунный барьер

4. Какое явление объясняет, почему раковые клетки отличаются от собаки и собаки?

Рисунок 2.8 – Экспорт карточек во внешние форматы

Таким образом, порядок работы с “AI FlashcardGenerator” включает несколько основных действий: выбор колоды, добавление источника, запуск генерации, проверку карточек, редактирование, повторение и экспорт. Такая последовательность позволяет пройти полный путь от исходного материала до готовой базы карточек запоминания в одном локальном приложении.

3 Обоснование экономической эффективности результатов ВКР

3.1 Выбор и обоснование методики расчета экономической эффективности

В качестве методики расчёта используется оценка затрат на разработку информационной системы и годового экономического эффекта от её применения, что соответствует логике технико-экономического обоснования проектных решений в информационных системах [6, с. 214], [13, с. 356].

Затраты на разработку включают оплату труда разработчика, начисления на оплату труда и прочие расходы, связанные с созданием и тестированием программного продукта.

Экономический эффект определяется через экономию рабочего времени пользователя, которое ранее затрачивалось на ручную подготовку карточек.

Для расчёта применяются следующие показатели: трудоёмкость разработки, стоимость одного часа работы разработчика, начисления на оплату труда, прочие расходы, годовая экономия времени пользователя, стоимость одного часа работы пользователя, годовой экономический эффект и срок окупаемости разработки.

Затраты на оплату труда разработчика определяются по формуле:

$$Зтр = Тразр \times Сч \quad (3.1)$$

где Зтр — затраты на оплату труда разработчика, руб.;

Тразр — трудоёмкость разработки информационной системы, ч;

Сч — стоимость одного часа работы разработчика, руб.

Начисления на оплату труда рассчитываются по формуле:

$$Звзн = Зтр \times Нвзн / 100 \quad (3.2)$$

где Звзн — начисления на оплату труда, руб.;

Зтр — затраты на оплату труда разработчика, руб.;

Нвзн — ставка начислений на оплату труда, %.

Общие затраты на разработку информационной системы рассчитываются по формуле:

$$Зобщ = Зтр + Звзн + Зпр \quad (3.3)$$

где Зобщ — общие затраты на разработку информационной системы, руб.;

Зтр — затраты на оплату труда разработчика, руб.;

Звзн — начисления на оплату труда, руб.;

Зпр — прочие расходы, связанные с разработкой и тестированием, руб.

Годовой экономический эффект от применения системы определяется по формуле:

$$Эгод = Тэк \times Счп \quad (3.4)$$

где Эгод — годовой экономический эффект, руб.;

Тэк — годовая экономия времени пользователя, ч;

Счп — стоимость одного часа работы пользователя, руб.

Срок окупаемости разработки определяется по формуле:

$$Ток = Зобщ / Эгод \quad (3.5)$$

где Ток — срок окупаемости разработки, лет;

Зобщ — общие затраты на разработку информационной системы, руб.;

Эгод — годовой экономический эффект от применения системы, руб.

Коэффициент экономической эффективности рассчитывается по формуле:

$$Кэф = Эгод / Зобщ \quad (3.6)$$

где Кэф — коэффициент экономической эффективности;

Эгод — годовой экономический эффект от применения системы, руб.;

Зобщ — общие затраты на разработку информационной системы, руб.

3.2 Расчет затрат на разработку информационной системы

Расчёт затрат на разработку информационной системы начинается с определения трудоёмкости работ, поскольку трудовые затраты являются одной из основных составляющих стоимости разработки локального программного продукта [13, с. 362], [17, с. 643].

В состав работ включены анализ предметной области, проектирование структуры данных и архитектуры, разработка серверной и клиентской частей, интеграция модуля генерации карточек, реализация импорта, экспорта и повторения, тестирование и подготовка документации.

Расчёт трудоёмкости представлен в таблице 3.1.

Таблица 3.1 – Расчёт трудоёмкости разработки информационной системы

№	Видработ	Трудоёмкость, ч
1	Анализ предметной области и требований	35
2	Проектирование структуры данных и архитектуры	45
3	Разработкасерверной частиприложения	85
4	Разработкаклиентской частиприложения	95
5	Интеграциямодулягенерациикарточек	50
6	Реализация импорта, экспорта и интервального повторения	35
7	Тестирование и корректировка программного продукта	35
8	Подготовка пользовательской и проектной документации	20
	Итого	400

Стоимость одного часа работы разработчика принимается равной 450 руб. Тогда затраты на оплату труда разработчика составят:

$$Зтр = 400 \times 450 = 180\,000 \text{ руб.} \quad (3.7)$$

Начисления на оплату труда принимаются в размере 30 %. Тогда сумма начислений составит:

$$Звзн = 180\,000 \times 30 / 100 = 54\,000 \text{ руб.} \quad (3.8)$$

К прочим расходам относятся расходы, связанные с использованием оборудования, электроэнергией, тестированием, хранением файлов, эксплуатацией локальной среды разработки и подготовкой материалов. Величина прочих рас-

ходов принимается равной 43 500 руб. Общий расчёт затрат представлен в таблице 3.2.

Таблица 3.2 – Расчёт затрат на разработку информационной системы

№	Статьязатрат	Сумма, руб.
1	Оплататрударазработчика	180 000
2	Начислениянаоплатутруда	54 000
3	Прочие расходы на разработку и тестирование	43 500
	Итого	277 500

Общие затраты на разработку информационной системы составят:

$$З_{общ} = 180\,000 + 54\,000 + 43\,500 = 277\,500 \text{ руб.} \quad (3.9)$$

Таким образом, расчётная сумма затрат на разработку информационной системы “AIFlashcardGenerator” составляет 277 500 руб.

3.3 Расчет экономического эффекта от внедрения системы

Экономический эффект от внедрения системы определяется за счёт сокращения времени, которое пользователь тратит на подготовку карточек запоминания. При ручном способе работы пользователь самостоятельно выделяет ключевые фрагменты текста, формулирует вопросы, готовит ответы и переносит карточки в отдельное приложение.

При использовании “AIFlashcardGenerator” значительная часть этих операций выполняется автоматически, а пользователь в основном проверяет и редактирует результат.

Для расчёта принято, что система используется в течение года при регулярной подготовке карточек по учебным, справочным и профессиональным материалам. Сравнение трудозатрат при ручной подготовке и при использовании разрабатываемой системы представлено в таблице 3.3.

Таблица 3.3 – Сравнение годовых трудозатрат при подготовке карточек

№	Операция	Ручной способ, ч	С использованием системы, ч
1	Выделение значимых фрагментов материала	160	24
2	Формулирование вопросов	190	12
3	Подготовка ответов и пояснений	170	72
4	Перенос данных, оформление и подготовка к экспорту	80	10
	Итого	600	118

Годовая экономия времени пользователя определяется как разность между трудозатратами при ручном способе и трудозатратами при использовании информационной системы:

$$Тэк = 600 - 118 = 482 \text{ ч.} \quad (3.10)$$

Стоимость одного часа работы пользователя принимается равной 450 руб. Тогда годовой экономический эффект составит:

$$Эгод = 482 \times 450 = 216\,900 \text{ руб.} \quad (3.11)$$

Полученный результат показывает, что за счёт автоматизации подготовки карточек пользователь может сократить значительный объём ручных операций. При этом система не исключает участие пользователя полностью: проверка и редактирование карточек сохраняются, так как результат генерации должен контролироваться.

3.4 Анализ результатов расчета экономической эффективности

На основании рассчитанных показателей можно определить срок окупаемости разработки информационной системы. Общие затраты на разработку составляют 277 500 руб., а годовой экономический эффект от применения системы составляет 216 900 руб.

$$Ток = 277\ 500 / 216\ 900 = 1{,}28 \text{ года} \quad (3.12)$$

Коэффициент экономической эффективности составит:

$$Кэф = 216\ 900 / 277\ 500 = 0{,}78 \quad (3.13)$$

Итоговые показатели экономической эффективности представлены в таблице 3.4.

Таблица 3.4 – Итоговые показатели экономической эффективности

№	Показатель	Значение
1	Общие затраты на разработку информационной системы	277 500 руб.
2	Годоваяэкономиявременипользователя	482 ч
3	Годовойэкономическийэффект	216 900 руб.
4	Срококупаемостиразработки	1,28 года
5	Коэффициент экономической эффективности	0,78

Расчёт показывает, что разработка информационной системы “AIFlashcardGenerator” является экономически целесообразной. Срок окупаемости составляет 1,28 года, что является приемлемым показателем для локального программного продукта, направленного на сокращение ручной обработки материалов. Основной экономический эффект достигается за счёт уменьшения времени

на выделение фрагментов, формулирование вопросов, подготовку ответов и перенос данных в систему повторения.

Дополнительный организационный эффект связан с тем, что пользователь получает единое локальное пространство для хранения источников, карточек, тегов, статусов повторения и экспортируемых результатов. Это снижает количество переключений между программами, уменьшает риск потери данных и позволяет повторно использовать уже созданные наборы карточек.

Таким образом, внедрение информационной системы “AIFlashcard Generator” позволяет сократить трудозатраты на подготовку карточек запоминания, уменьшить зависимость от внешних сервисов и повысить удобство работы с учебными и профессиональными материалами. Полученные расчёты подтверждают экономическую эффективность разработки.

Заключение

В ходе выполнения выпускной квалификационной работы была рассмотрена задача разработки информационной системы “AI FlashcardGenerator” для создания карточек запоминания на основе ИИ. Актуальность выбранной темы связана с тем, что пользователи всё чаще работают с большим количеством учебных, справочных и профессиональных материалов, которые необходимо не только прочитать, но и преобразовать в форму, удобную для повторения и самопроверки. Ручное создание карточек требует значительных временных затрат, особенно при обработке крупных документов, лекций, статей и технической документации.

В первой главе была рассмотрена предметная область, связанная с подготовкой материалов к повторению и запоминанию. Было показано, что карточки запоминания являются удобным инструментом для активного воспроизведения информации, а интервальное повторение позволяет организовать более рациональную работу с материалом. При этом основная проблема заключается не в отсутствии программ для повторения, а в трудоёмкости подготовки самих карточек.

Также в аналитической части были рассмотрены существующие программные решения: системы интервального повторения, онлайн-сервисы карточек, AI-сервисы генерации вопросов, универсальные чат-боты и инструменты организации знаний. Анализ показал, что каждая группа решений закрывает только часть потребностей пользователя. Одни системы хорошо подходят для повторения, но требуют ручного наполнения; другие позволяют генерировать карточки, но зависят от облачной платформы; третьи помогают организовывать материалы, но не являются специализированной системой генерации карточек. На основании этого была обоснована необходимость разработки локальной информационной системы, объединяющей добавление источников, генерацию карточек, хранение, проверку результата, повторение и экспорт.

Во второй главе были разработаны проектные решения по информационному, технологическому, техническому и программному обеспечению системы. В рамках информационного обеспечения были определены основные объекты системы: колода, источник, карточка, тег и состояние повторения. Была описана структура входной и выходной информации, а также порядок хранения источников, карточек, тегов и данных повторения.

При выборе технологического стека были обоснованы решения по использованию Python, FastAPI, SQLite, SQLAlchemy, HTML, CSS и JavaScript. Такой набор технологий позволяет реализовать локальное веб-приложение, в котором серверная часть выполняет обработку данных и генерацию карточек, а клиентская часть обеспечивает удобное взаимодействие пользователя с системой. Использование SQLite соответствует локальному характеру приложения, так как база данных хранится на компьютере пользователя и не требует отдельного серверного процесса.

В рамках программного обеспечения была описана архитектура системы “AI FlashcardGenerator”. Система построена по принципу разделения клиентской и серверной частей. Серверная часть отвечает за обработку источников, работу с базой данных, генерацию карточек, проверку результата, интервальное повторение, импорт и экспорт. Клиентская часть предназначена для отображения графа знаний, списка карточек, инспектора объектов, поиска, режима повторения и окон экспорта.

Особое внимание было уделено модулю генерации карточек. В работе отмечено, что использование языковой модели не должно сводиться к механическому сохранению любого полученного ответа. Результат генерации должен проверяться: карточка должна содержать вопрос, ответ, иметь смысловую связь с исходным материалом и не быть пустой или шаблонной. Такой подход позволяет повысить качество создаваемой базы карточек и избежать сохранения элементов, которые не имеют учебной ценности.

В проектной части также было описано руководство пользователя. Были рассмотрены основные элементы интерфейса и порядок работы с системой: вы-

бор колоды, добавление источника, запуск генерации, проверка карточек, редактирование, повторение и экспорт. Такая последовательность позволяет пользователю пройти полный путь от исходного материала до готовой базы карточек запоминания в одном локальном приложении.

В третьей главе была выполнена оценка экономической эффективности разработки. Были рассчитаны затраты на разработку информационной системы, годовая экономия времени пользователя, годовой экономический эффект, срок окупаемости и коэффициент экономической эффективности. Общие затраты на разработку составили 277 500 руб., годовой экономический эффект — 216 900 руб., срок окупаемости — 1,28 года, коэффициент экономической эффективности — 0,78. Полученные показатели подтверждают целесообразность разработки системы, так как она позволяет сократить трудозатраты на подготовку карточек и уменьшить зависимость пользователя от внешних сервисов.

Практическая значимость работы заключается в создании программного продукта, который может использоваться для подготовки карточек запоминания по собственным материалам пользователя. Система позволяет добавлять источники, генерировать карточки с применением языковой модели, сохранять связь карточек с исходным материалом, редактировать результат, организовывать данные по колодам и тегам, выполнять повторение и экспортировать карточки во внешние форматы.

Таким образом, цель выпускной квалификационной работы достигнута. В работе была разработана информационная система “AI FlashcardGenerator” для создания карточек запоминания на основе ИИ. Поставленные задачи выполнены: проведён анализ предметной области, рассмотрены существующие решения, сформулированы требования к системе, разработаны проектные решения, описана архитектура и основные модули приложения, рассмотрен порядок работы пользователя и рассчитана экономическая эффективность разработки.

Список использованных источников

1. Методические рекомендации по оформлению выпускной квалификационной работы для обучающихся направления подготовки 09.03.03 «Прикладная информатика». — Туапсе : филиал ФГБОУ ВО «РГГМУ», 2025. — 31 с.
2. ГОСТ 7.32–2017. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. — Москва :Стандартинформ, 2017.
3. ГОСТ Р 7.0.5–2008. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая ссылка. Общие требования и правила составления. — Москва :Стандартинформ, 2008.
4. ГОСТ 34.601–90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. — Москва : Издательство стандартов, 1990.
5. ГОСТ 34.602–2020. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. — Москва :Стандартинформ, 2020.
6. Чистов, Д. В. Проектирование информационных систем : учебник и практикум для вузов / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук. — Москва :Юрайт, 2024. — 293 с.
7. Чистов, Д. В. Проектирование информационных систем : учебник для вузов / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук. — Москва :Юрайт, 2026. — 273 с.
8. Григорьев, М. В. Проектирование информационных систем : учебник для вузов / М. В. Григорьев, И. И. Григорьева. — Москва :Юрайт, 2025. — 278 с.
9. Нестеров, С. А. Базы данных : учебник и практикум для вузов / С. А. Нестеров. — 2-е изд., перераб. и доп. — Москва :Юрайт, 2024. — 258 с.
10. Советов, Б. Я. Базы данных : учебник для вузов / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — Москва :Юрайт, 2024. — 463 с.

11. Кузнецов, С. Д. Основы баз данных / С. Д. Кузнецов. — Москва : Интернет-Университет Информационных Технологий, 2007. — 484 с.
12. Маклаков, С. В. BPwin и ERwin. CASE-средства разработки информационных систем / С. В. Маклаков. — Москва : Диалог-МИФИ, 2007. — 304 с.
13. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем / А. М. Вендров. — Москва : Финансы и статистика, 2006. — 544 с.
14. Connolly, T. Database Systems: A Practical Approach to Design, Implementation, and Management / T. Connolly, C. Begg. — 6th ed. — Boston : Pearson, 2015. — 1440 p.
15. Silberschatz, A. Database System Concepts / A. Silberschatz, H. F. Korth, S. Sudarshan. — 7th ed. — New York : McGraw-Hill Education, 2019. — 1376 p.
16. Sommerville, I. Software Engineering / I. Sommerville. — 10th ed. — Boston : Pearson, 2016. — 816 p.
17. Pressman, R. S. Software Engineering: A Practitioner's Approach / R. S. Pressman, B. R. Maxim. — 9th ed. — New York : McGraw-Hill Education, 2020.
18. Fowler, M. Patterns of Enterprise Application Architecture / M. Fowler. — Boston : Addison-Wesley, 2002. — 560 p.
19. Ларман, К. Применение UML 2.0 и шаблонов проектирования / К. Ларман. — Москва : Вильямс, 2013. — 736 с.
20. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Якобсон. — Москва : ДМК Пресс, 2006. — 496 с.
21. Нильсен, Я. Веб-дизайн: книга Якоба Нильсена / Я. Нильсен. — Санкт-Петербург : Символ-Плюс, 2006. — 512 с.
22. Norman, D. The Design of Everyday Things / D. Norman. — Revised and expanded ed. — New York : Basic Books, 2013. — 368 p.
23. Krug, S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability / S. Krug. — 3rd ed. — Berkeley : New Riders, 2014. — 216 p.

24. ISO 9241-210:2019. Ergonomics of human-system interaction. Human-centred design for interactive systems. — Geneva : International Organization for Standardization, 2019.
25. Эббингауз, Г. Память: вклад в экспериментальную психологию / Г. Эббингауз. — Москва : АСТ, 2020. — 160 с.
26. Лейтнер, С. Как учиться быстрее и эффективнее / С. Лейтнер. — Москва : Эксмо, 2018. — 224 с.
27. Brown, P. C. Make It Stick: The Science of Successful Learning / P. C. Brown, H. L. Roediger, M. A. McDaniel. — Cambridge : Harvard University Press, 2014. — 336 p.
28. Cepeda, N. J. Distributed practice in verbal recall tasks: A review and quantitative synthesis / N. J. Cepeda, H. Pashler, E. Vul, J. T. Wixted, D. Rohrer // Psychological Bulletin. — 2006. — Vol. 132, № 3. — P. 354–380.
29. Dunlosky, J. Improving Students' Learning With Effective Learning Techniques / J. Dunlosky, K. A. Rawson, E. J. Marsh, M. J. Nathan, D. T. Willingham // Psychological Science in the Public Interest. — 2013. — Vol. 14, № 1. — P. 4–58.
30. AnkiManual : официальное руководство пользователя Anki [Электронный ресурс]. — URL: https://docs.ankiweb.net/ (дата обращения: 14.06.2026).
31. Quizlet Help Center : справочный центр сервиса Quizlet [Электронный ресурс]. — URL: https://help.quizlet.com/hc/en-us (дата обращения: 14.06.2026).
32. SuperMemoMethod : материалы по методу интервального повторения SuperMemo [Электронный ресурс]. — URL: https://www.supermemo.com/en/supermemo-method (дата обращения: 14.06.2026).
33. Knowt : сервис для создания карточек и учебных материалов [Электронный ресурс]. — URL: https://knowt.com/ (дата обращения: 14.06.2026).
34. Monic.ai FlashcardMaker : сервис генерации карточек на основе ИИ [Электронный ресурс]. — URL: https://monic.ai/flashcard-maker/ (дата обращения: 14.06.2026).

35. Python Documentation : официальная документация языка программирования Python [Электронный ресурс]. — URL: https://docs.python.org/3/ (дата обращения: 14.06.2026).
36. FastAPIDocumentation : официальная документация веб-фреймворка FastAPI [Электронный ресурс]. — URL: https://fastapi.tiangolo.com/ (дата обращения: 14.06.2026).
37. UvicornDocumentation : официальная документация ASGI-сервера Uvicorn [Электронный ресурс]. — URL: https://www.uvicorn.org/ (дата обращения: 14.06.2026).
38. SQLiteDocumentation : официальная документация системы управления базами данных SQLite [Электронный ресурс]. — URL: https://www.sqlite.org/docs.html (дата обращения: 14.06.2026).
39. SQLAlchemyDocumentation : официальная документация библиотеки SQLAlchemy [Электронный ресурс]. — URL: https://docs.sqlalchemy.org/ (дата обращения: 14.06.2026).
40. PydanticDocumentation : официальная документация библиотеки Pydantic [Электронный ресурс]. — URL: https://pydantic.dev/docs/ (дата обращения: 14.06.2026).
41. MDN Web Docs. HTML: HyperTextMarkup Language [Электронный ресурс]. — URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата обращения: 14.06.2026).
42. MDN Web Docs. CSS: Cascading Style Sheets [Электронный ресурс]. — URL: https://developer.mozilla.org/en-US/docs/Web/CSS (датаобращения: 14.06.2026).
43. MDN Web Docs. JavaScript [Электронный ресурс]. — URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата обращения: 14.06.2026).
44. PyMuPDFDocumentation : официальная документация библиотеки PyMuPDF [Электронный ресурс]. — URL: https://pymupdf.readthedocs.io/ (дата обращения: 14.06.2026).

Приложение А

Фрагмент листинга программных модулей

Листинг А.1 – Фрагмент запуска серверной части приложения

```
class NoCacheStaticFiles(StaticFiles):
    async def get_response(self, path, scope):
        response = await super().get_response(path, scope)
response.headers["Cache-Control"] = "no-store, no-cache, must-revalidate, max-age=0"
response.headers["Pragma"] = "no-cache"
        return response

app = FastAPI(title="AI Flashcards — Knowledge Graph")

STATIC_DIR = os.path.join(BASE_DIR, "static")
UPLOADS_DIR = os.path.join(BASE_DIR, "uploads")
os.makedirs(UPLOADS_DIR, exist_ok=True)

app.add_middleware(
CORSMiddleware,
allow_origins=["*"],
allow_credentials=True,
allow_methods=["*"],
allow_headers=["*"],
)

if os.path.isdir(STATIC_DIR):
app.mount("/static", NoCacheStaticFiles(directory=STATIC_DIR), name="static")

if os.path.isdir(UPLOADS_DIR):
app.mount("/uploads", StaticFiles(directory=UPLOADS_DIR), name="uploads")
```

Продолжение приложения А

```
@app.on_event("startup")
async def startup_event():
    try:
init_engine(model_name=current_model)
    except Exception as e:
        print(f"[STARTUP] Модельнезагруженапристарте: {e}")

@app.on_event("shutdown")
async def shutdown_event():
unload_engine()

@app.get("/")
async def root():
    return FileResponse(
os.path.join(BASE_DIR, "index.html"),
        headers={
            "Cache-Control": "no-store, no-cache, must-revalidate, max-age=0",
            "Pragma": "no-cache",
        },
    )

@app.get("/api/health")
async def health():
    return {"status": "ok", "llm": get_engine_status()}
```

Листинг А.2 – Фрагмент модели данных карточки

```
class Card(Base):
    __tablename__ = "cards"

    id = Column(Integer, primary_key=True, index=True)
    front = Column(String)
    back = Column(String)
source_quote = Column(String, nullable=True)
    mnemonic = Column(String, nullable=True)
    tags = Column(String, nullable=True)
    status = Column(String, default="inbox")
due_date = Column(DateTime, nullable=True)
card_type = Column(String, default="basic")
ease_factor = Column(Float, default=2.5)
interval_days = Column(Integer, default=0)
review_count = Column(Integer, default=0)
    lapses = Column(Integer, default=0)
last_reviewed_at = Column(DateTime, nullable=True)
image_path = Column(String, nullable=True)
    order = Column(Integer, default=0)
deck_id = Column(Integer, ForeignKey("decks.id"), index=True)
created_at = Column(DateTime, default=func.now())
source_node_id = Column(String, nullable=True, index=True)
    model = Column(String, nullable=True)
export_profile = Column(String, default="anki")
fields_json = Column(String, nullable=True)
    x = Column(Integer, nullable=True)
    y = Column(Integer, nullable=True)

    deck = relationship("Deck", back_populates="cards")
```

Листинг А.3 – Фрагмент создания колоды

```
@app.post("/api/decks")
async def create_deck(deck_data: dict, db: Session = Depends(get_db)):
    name = (deck_data.get("name") or "").strip()

    if not name or name.lower() in ["новаяколода", "new deck"]:
        name = datetime.now().strftime("%d.%m.%Y %H:%M")

    deck = Deck(
        name=name,
        tags=normalize_tags_string(deck_data.get("tags") or "", max_tags=24) or None,
    )

db.add(deck)
db.commit()
db.refresh(deck)

    return {
        "id": deck.id,
        "name": deck.name,
        "tags": deck.tags or "",
        "created_at": deck.created_at.isoformat(),
}
```

Листинг А.4 – Фрагмент подготовки текстовых фрагментов для генерации карточек

```
@dataclass(frozen=True)
class EvidenceUnit:
eid: str
    text: str
    score: float
    order: int
```

```
@dataclass(frozen=True)
class EvidenceBatch:
   prompt: str
   count: int
   evidence: tuple[EvidenceUnit, ...]

def build_evidence_units(
   text: str,
desired_count: int,
   language: str = "ru",
) ->list[EvidenceUnit]:
   sentences = _split_sentences(text)

   if not sentences:
      return []
   candidates: list[EvidenceUnit] = []

   for i, sent in enumerate(sentences):
      score = _sentence_score(sent, i)

      if score <= 0.15:
         continue

candidates.append(
EvidenceUnit(
eid=f"E{i + 1}",
            text=sent,
            score=score,
            order=i,
         )
      )

   ranked = sorted(candidates, key=lambda x: (-x.score, x.order))
   selected: list[EvidenceUnit] = []
```

```
max_units = max(desired_count * 3, desired_count + 8, 12)

    for unit in ranked:
        if any(sentence_similarity(unit.text, old.text) >= 0.91 for old in selected):
            continue

selected.append(unit)

        if len(selected) >= max_units:
            break

    return sorted(selected, key=lambda x: x.order)
```

Листинг А.5 – Фрагмент формирования запроса к языковой модели

```
def build_evidence_prompt(
    evidence: Sequence[EvidenceUnit],
    count: int,
    language: str = "ru",
custom_prompt: str = "",
forced_card_type: str = "auto",
avoid_fronts: Sequence[str] | None = None,
retry_mode: bool = False,
tag_hints: str = "",
output_profile: str = "anki",
) -> str:
    count = max(1, int(count))
source_payload = _source_text_payload(evidence)

lines = [
        "Создай качественные учебные карточки по SOURCE_TEXTS_JSON.",
f"Верни до {count} разных карточек, если в источнике хватает разных фактов.",
        "Ответ только валидный JSON-массив с короткими ключами:",
        '[{"t":"fact","q":"вопрос","a":"ответ","s":"точная цитата из источника","m":"короткая под-
сказка"}]',
```

```
        "Вопрос должен быть самодостаточным.",
        "Ответ должен объяснять факт своими словами.",
        "Поле s должно содержать точное предложение или фрагмент из SOURCE_TEXTS_JSON.",
        "Поле m обязательно: короткая подсказка для запоминания на 2-7 слов.",
        "Используй разные факты и разные углы вопроса.",
        "Без markdown, без пояснений, без скрытых рассуждений, без системного текста.",
    ]

    if custom_prompt:
        lines.append(
            f"Фокуспользователя: {compact_spaces(custom_prompt)[:180]}."
        )

    if avoid_fronts:
        lines.append("Неповторяйэтивопросы: " + "; ".join(avoid_fronts))

    return (
        "<bos><start_of_turn>user\n"
        + "\n".join(lines)
        + "\n\nSOURCE_TEXTS_JSON:\n"
        + json.dumps(source_payload, ensure_ascii=False, separators=(",", ":"))
        + "\n<end_of_turn>\n<start_of_turn>model\n"
    )
```

Листинг А.6 – Фрагмент проверки результата генерации

```
def validate_model_card(
    card: dict,
    evidence: Sequence[EvidenceUnit],
    language: str = "ru",
    output_profile: str = "anki",
) -> tuple[dict | None, str]:
    if not isinstance(card, dict):
        return None, "not dict"
```

```
    front = compact_spaces(str(
card.get("front")
or card.get("question")
        or card.get("q")
        or ""))
    back = compact_spaces(str(
card.get("back")
        or card.get("answer")
        or card.get("a")
        or ""
    ))
    quote = compact_spaces(str(
card.get("source_quote")
        or card.get("quote")
        or card.get("s")
        or ""
    ))

    mnemonic = compact_spaces(str(
card.get("mnemonic")
        or card.get("hint")
        or card.get("m")
        or ""
    ))

card_type = str(
card.get("card_type")
        or card.get("type")
        or card.get("t")
        or "basic"
).strip().lower().replace(" ", "_")

    if card_type not in {"basic", "definition", "fact", "concept", "cloze", "true_false", "mcq"}:
card_type = "basic"
```

```
    if not front or not back:
        return None, "empty"
    if not front.endswith("?"):
        return None, "not question"
    if back.endswith("?") or len(back.split()) < 4:
        return None, "bad answer"
    if not quote_is_supported(quote, evidence):
repaired_quote = best_evidence_quote(front, back, quote, evidence)
        if repaired_quote:
            quote = repaired_quote
        else:
            return None, "unsupported quote"

    result = {
        "card_type": card_type,
        "front": front[:240],
        "back": back[:900],
        "source_quote": quote[:900],
"mnemonic": mnemonic[:360],
    }
    return result, ""
```

ЛистингА.7 – Фрагменталгоритмаинтервальногоповторения

```
@dataclass(frozen=True)
class ReviewState:
ease_factor: float = 2.5
interval_days: int = 0
review_count: int = 0
    lapses: int = 0



@dataclass(frozen=True)
class ReviewResult:
ease_factor: float
```

```
interval_days: int
review_count: int
    lapses: int
due_date: datetime
    status: str
last_reviewed_at: datetime

def schedule_review(
    state: ReviewState,
    rating: str,
    now: datetime | None = None,
) ->ReviewResult:
    rating = (rating or "good").lower()
    if rating == "medium":
        rating = "good"

reviewed_at = now or datetime.now()
    ease = float(state.ease_factor or 2.5)
    interval = int(state.interval_days or 0)
    reps = int(state.review_count or 0) + 1
    lapses = int(state.lapses or 0)

    if rating == "again":
        lapses += 1
        interval = 0
        ease = max(1.3, ease - 0.2)
        status = "today"
elif rating == "hard":
        interval = max(1, math.ceil(max(1, interval) * 1.25))
        ease = max(1.3, ease - 0.08)
        status = "planned"
elif rating == "easy":
        interval = 4 if reps <= 1 else max(4, math.ceil(max(1, interval) * (ease + 0.35)))
        ease = min(3.2, ease + 0.15)
        status = "planned"
```

```
    else:
        if reps <= 1:
            interval = 1
elif reps == 2:
            interval = max(3, interval)
        else:
            interval = max(2, math.ceil(max(1, interval) * ease))
        status = "planned"

due_date = start_of_local_day(interval)

    return ReviewResult(
ease_factor=round(ease, 2),
interval_days=int(interval),
review_count=reps,
        lapses=lapses,
due_date=due_date,
        status=status,
last_reviewed_at=reviewed_at,
    )
```

ЛистингА.8 – Фрагментэкспортакарточек

```
def query_export_cards(
deck_id: int,
card_ids: Optional[str],
source_id: Optional[str],
db: Session,
) ->Tuple[Deck, List[Card]]:
    deck = db.query(Deck).filter(Deck.id == deck_id).first()

    if not deck:
        raise HTTPException(status_code=404, detail="Deck not found")

    query = db.query(Card).filter(Card.deck_id == deck_id)
```

```
    if card_ids:
        ids = [int(x) for x in card_ids.split(",") if x.strip().isdigit()]
        if ids:
            query = query.filter(Card.id.in_(ids))

elifsource_id:
        query = query.filter(Card.source_node_id == source_id)
    cards = query.order_by(Card.order.asc(), Card.created_at.asc()).all()

    if not cards:
        raise HTTPException(status_code=404, detail="No cards to export")
    return deck, cards

@app.get("/api/decks/{deck_id}/export/json")
async def export_deck_json(
deck_id: int,
card_ids: Optional[str] = None,
source_id: Optional[str] = None,
db: Session = Depends(get_db),
):
    deck, cards = query_export_cards(deck_id, card_ids, source_id, db)

    payload = {
        "deck": deck.name,
        "cards": [
            {
                "front": card.front,
                "back": card.back,
                "source_quote": card.source_quote,
                "mnemonic": card.mnemonic,
                "tags": card.tags,
                "card_type": card.card_type,
            }
            for card in cards
```

```
        ],
    }
    data = json.dumps(payload, ensure_ascii=False, indent=2)

    return Response(
        content=data,
media_type="application/json",
        headers={
            "Content-Disposition": f"attachment; filename*=UTF-8"{quote(deck.name)}.json"
},
    )
```