

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования «Самарский национальный исследовательский
университет имени академика С.П. Королева»
(Самарский университет)

Институт _____ информатики и кибернетики
Кафедра _____ программных систем

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**«РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ОБУЧЕНИЯ
ПО СИСТЕМЕ ЛЕЙТНЕРА С РЕАЛИЗАЦИЕЙ
АЛГОРИТМА ИНТЕРВАЛЬНОГО ПОВТОРЕНИЯ»**

по направлению подготовки 02.03.02

Фундаментальная информатика и информационные технологии
(уровень бакалавриата)

направленность (профиль) «Информационные технологии»

Обучающийся _____ А.А. Алёнушка
(подпись, дата)

Руководитель ВКР

к.т.н., доцент _____ О.А. Гордеева
(подпись, дата)

Нормоконтролер _____ Е.В. Сопченко
(подпись, дата)

Самара 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

Кафедра _____ программных систем _____

УТВЕРЖДАЮ
Заведующий кафедрой

_____ С.В. Востокин
« ____ » _____ 2025 г.

задание на выпускную квалификационную работу (ВКР)

обучающемуся _____ Алёнушка Александру Александровичу
группы _____ 6401-020302D _____

1. Тема ВКР: Разработка веб-приложения для обучения по системе Лейтнера с реализацией алгоритма интервального повторения
утверждена приказом по университету от «24» апреля 2025 г. № 223-Т

2. Перечень вопросов, подлежащих разработке в ВКР:

- 1) Провести анализ предметной области: интервальное повторение, система Лейтнера, флеш-карточки, модель SM-2, процессы обучения и запоминания информации.
- 2) Сделать обзор систем-аналогов в области существующих решений для интервального повторения и обучения с карточками.
- 3) Разработать проект системы с использованием методологии структурного и объектно-ориентированного проектирования
- 4) Разработать и реализовать информационное и программное обеспечение
- 5) Провести тестирование и отладку разработанного веб-приложения для обучения по системе Лейтнера с реализацией алгоритма интервального повторения

3. Дата выдачи задания: «24» апреля 2025г.

4. Срок представления на кафедру законченной ВКР: « 5 »июня 2025г.

Руководитель ВКР

к.т.н., доцент, доцент кафедры программных систем _____ О.А. Гордеева
« 24 » _____ 04 _____ 2025 г.

Задание принял к исполнению

_____ А.А. Алёнушка
« 24 » _____ 04 _____ 2025 г.

РЕФЕРАТ

Пояснительная записка 57 с, 23 рисунка, 1 таблица, 22 источника, 2 приложения.

Графическая часть: 20 слайдов презентации PowerPoint.

ИНТЕРВАЛЬНОЕ ПОВТОРЕНИЕ, СИСТЕМА ЛЕЙТНЕРА, ОБУЧАЮЩИЕ КАРТОЧКИ, ВЕБ-ПРИЛОЖЕНИЕ, АЛГОРИТМ SM-2, GRAPHQL, SPRING BOOT, REACT, LIQUIBASE, POSTGRESQL.

Цель работы – разработать автоматизированную систему для обучения по методу Лейтнера с реализацией алгоритма интервального повторения, обеспечивающую эффективное запоминание информации с помощью обучающих карточек.

В процессе работы были разработаны алгоритмы и соответствующее программное обеспечение, позволяющее пользователю создавать, редактировать и повторять карточки в соответствии с принципами интервального повторения. Система автоматически рассчитывает оптимальные интервалы для повторения и отслеживает прогресс пользователя.

Система разработана на языке Java с использованием фреймворка Spring Boot, а также технологий GraphQL, Liquibase (для миграций базы данных) и React (для пользовательского интерфейса). Функционирует под управлением операционных систем семейства Windows и Linux. Доступ к данным осуществляется с помощью СУБД PostgreSQL.

СОДЕРЖАНИЕ

Введение.....	6
1 Описание и анализ предметной области	8
1.1 Основные понятия и определения.....	8
1.2 Актуальность задачи,.....	11
1.3 Описание систем-аналогов.....	12
1.3.1 Anki.....	12
1.3.2 StudyStack	14
1.3.3 Конкурентный анализ систем-аналогов	15
1.4 Описание автоматизируемого процесса	16
1.5 Постановка задачи	18
1.6 Выводы по главе.....	19
2 Проектирование системы	20
2.1 Выбор и обоснование архитектуры системы	20
2.2 Проект системы	22
2.2.1 Структурная схема системы	22
2.2.2 Диаграмма вариантов использования	24
2.2.3 Диаграмма деятельности.....	25
2.2.4 Диаграмма последовательности	27
2.2.5 Диаграмма классов, логический уровень	27
2.3 Выбор и обоснование средств реализации.....	29
2.4 Выводы по главе.....	31
3 Реализация системы	32
3.1 Описание интерфейса пользователя	32
3.1.1 Начало работы: регистрация и вход.....	32
3.1.2 Работа с коллекциями карточек	33
3.1.3 Добавление и редактирование карточек.....	34
3.1.4 Редактирование коллекций и поиск карточек.....	35
3.1.5 Повторение карточек	35
3.2 Физическая модель данных.....	36

3.3 Выводы по главе.....	38
Заключение	39
Список использованных источников	40
Приложение А. Руководство пользователя	42
Приложение Б. Код программы.....	47

ВВЕДЕНИЕ

В условиях стремительного развития цифровых технологий и увеличения объёма информации, которую необходимо усваивать, особую актуальность приобретают методы эффективного запоминания и обучения. Одним из таких методов является интервальное повторение, основанное на принципе увеличения интервалов между повторениями изучаемого материала, что способствует его долговременному запоминанию [1]. Этот подход был впервые описан немецким психологом Германом Эббингаузом в конце XIX века, который выявил закономерность забывания информации со временем и предложил способы её закрепления через повторение [2].

Среди практических реализаций метода интервального повторения выделяется система Лейтнера, предложенная немецким научным журналистом Себастьяном Лейтнером в 1970-х годах [3]. Суть метода заключается в использовании карточек с информацией, которые распределяются по группам в зависимости от уровня усвоения, и повторяются с различной частотой: чем хуже запомнена информация, тем чаще она повторяется.

Современные исследования подтверждают высокую эффективность интервального повторения в обучении. Так, в работе Керфута и соавторов была проведена серия исследований с участием студентов-медиков, в ходе которых использование интервального повторения привело к значительному улучшению результатов тестирования по сравнению с традиционными методами обучения [4]. Это свидетельствует о том, что регулярное повторение материала с увеличивающимися интервалами способствует более прочному закреплению знаний в долгосрочной памяти.

Несмотря на наличие программных решений, реализующих методы интервального повторения (например, Anki, SuperMemo), существует потребность в разработке адаптивных и ориентированных на пользователя систем, учитывающих современные требования к интерфейсу.

Цель данной работы — разработка веб-приложения, реализующего метод интервального повторения на основе системы Лейтнера. Приложение должно

обеспечивать эффективное управление процессом обучения, предоставлять инструменты для создания и редактирования карточек.

В качестве методической основы выбран подход, сочетающий принципы активного воспроизведения и интервального повторения, что соответствует современным тенденциям в области педагогики. Разработка будет осуществляться с использованием современных веб-технологий: Java (Spring Boot) для серверной части, GraphQL для организации взаимодействия между клиентом и сервером, React для клиентской части, PostgreSQL в качестве системы управления базами данных, а также Liquibase для управления версиями базы данных.

Таким образом, предлагаемая система будет представлять собой современное, адаптивное и эффективное средство для самостоятельного обучения, основанное на проверенных научных методах и современных технологических решениях.

1 Описание и анализ предметной области

1.1 Основные понятия и определения

Флэш-карточка — это эффективное средство обучения, представляющее собой двустороннюю карточку, на одной стороне которой записан вопрос, термин или фрагмент текста, а на другой — ответ, определение или пояснение. Такой формат заточен под активное воспроизведение задуманных сведений и самопроверку. [5]

Cloze, или тест с пропусками, — это упражнение, в котором из исходного текста удаляются отдельные слова или фразы, а обучающийся восстанавливает их по контексту. Данный метод развивает навыки понимания текста и активного воспоминания, требуя от пользователя анализа синтаксиса и семантики фрагмента. [6]

На рисунке 1 показан пример флэш-карточки с cloze-пропусками.

Подготовка для реализации простого аспекта

Надо реализовать логирование начала и окончания логики некоторого сценария использования.

Для использования аспектов надо [...]:

```
[...]  
[...]  
[...]
```

```
[...]  
public class CommentService {  
    private [...] = [...](...);  
    public void publishComment(Comment comment) {  
        logger.info("Publishing comment: " + comment.getText());  
    }  
}
```



Подготовка для реализации простого аспекта

Надо реализовать логирование начала и окончания логики некоторого сценария использования.

Для использования аспектов надо **добавить зависимость** в pom.xml:

```
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-aspects</artifactId>  
    <version>6.1.14</version>  
</dependency>
```

```
@Service  
public class CommentService {  
    private Logger logger = Logger.getLogger(...);  
    public void publishComment(Comment comment) {  
        logger.info("Publishing comment: " + comment.getText());  
    }  
}
```

Рисунок 1 – Пример флэш-карточки с cloze-пропусками

Классическая кривая забывания Эббингауза — график, иллюстрирующий закономерности утраты информации с течением времени после первоначального запоминания. Сразу после изучения материал удерживается практически полностью, но затем забывание происходит наиболее интенсивно, а со временем скорость утраты знаний замедляется и стремится к асимптоте.

На рисунке 2 приведено изображение этой кривой.

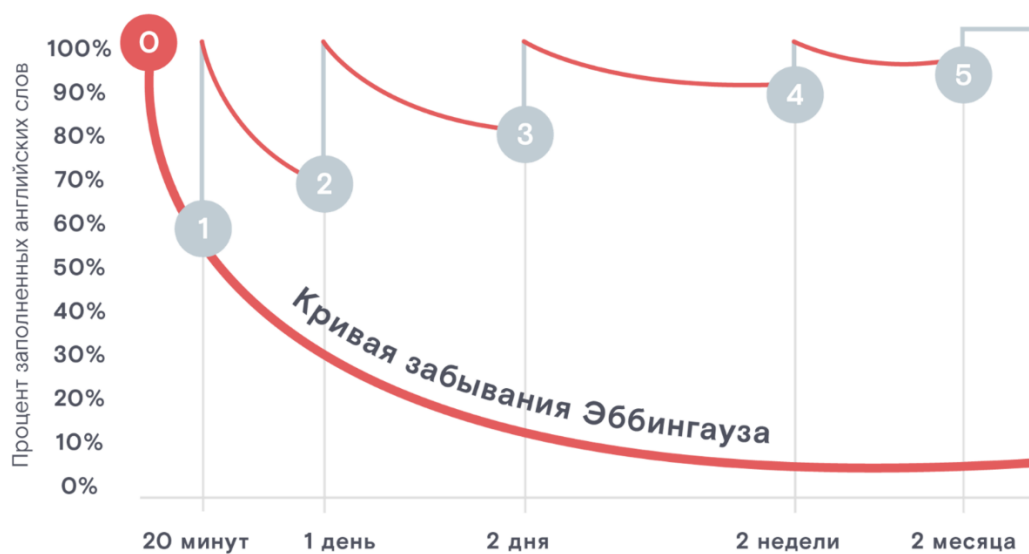


Рисунок 2 – Кривая Эббингауза

Множество методов интервального повторения опираются на кривую Эббингауза, назначая повторения в такие промежутки, когда забывание ещё не стало критическим, а частичное ослабление памяти при последующем повторении обеспечивает более прочное закрепление материала.

Система Лейтнера — один из первых практических алгоритмов интервального повторения, предложенный Себастьяном Лейтнером в 1970-х годах. Суть метода состоит в разделении набора карточек на несколько «ящиков», каждый из которых соответствует своему интервалу повторения (например, 1 день, 2 дня, 4 дня и т. д.). При успешном воспроизведении карточка перемещается в следующий ящик с увеличенным интервалом; если ответ был неверным, карточка возвращается в предыдущий ящик для более частых повторений.

На рисунке 3 показана схема работы системы Лейтнера.

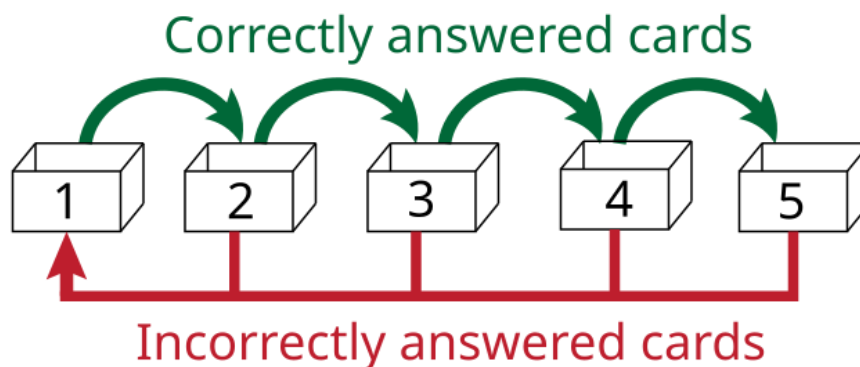


Рисунок 3 – Схема работы системы Лейтнера

Алгоритм SM-2 лежит в основе оригинальной программы SuperMemo и до сих пор остаётся открытым стандартом. Он включает следующие шаги:

- 1) разделить материал на отдельные элементы и каждому присвоить исходное значение коэффициента легкости $EF = 2.5$;
- 2) первые два интервала повторения задаются жёстко:
 - $I(1) = 1$ день
 - $I(2) = 6$ дней
- 3) для n -го повторения при $n > 2$ рассчитывать интервал $I(n) = I(n - 1) \times EF$;
- 4) сразу после ответа оценивать качество реакции q по шкале 0 – 5 (5 — идеальный ответ, 0 — полное забывание);
- 5) пересчитывать EF по формуле $EF' = EF + (0.1 - (5 - q) \times (0.08 + (5 - q) \times 0.02))$, если $EF' < 1.3$, присвоить $EF' = 1.3$;
- 6) если $q < 3$, считать, что элемент не выучен, и начать повторение заново, без изменения EF ;
- 7) в тот же день повторить все элементы с $q < 4$ до достижения оценки ≥ 4 .

Таким образом SM-2 адаптируется к индивидуальным результатам пользователя, автоматически увеличивая интервалы для хорошо запоминаемых

карточек и чаще возвращая «трудные».

1.2 Актуальность задачи,

В условиях постоянного увеличения объёмов информации, необходимой для профессиональной и образовательной деятельности, актуальность методов эффективного обучения и запоминания существенно возрастает. Одним из наиболее эффективных подходов является интервальное повторение, реализуемое в системе Лейтнера, которая основана на увеличении интервалов между повторениями изучаемого материала. Это способствует его надёжному закреплению в долговременной памяти.

Разрабатываемая автоматизированная система предназначена не только для студентов и учащихся, но также может эффективно использоваться при подготовке к профессиональным и квалификационным экзаменам. Среди таких сфер:

- 1) курсы повышения квалификации и профессиональной переподготовки сотрудников;
- 2) подготовка сотрудников предприятий и организаций к сдаче квалификационных экзаменов, например, при повышении профессиональной категории инженеров и технических специалистов;
- 3) подготовка сотрудников органов внутренних дел к прохождению профессиональных аттестаций и экзаменов, требующих запоминания значительного объёма нормативной и правовой информации;
- 4) обучение и подготовка кандидатов на получение водительских удостоверений, где требуется запоминать правила дорожного движения и другую необходимую информацию.

Помимо непосредственного запоминания информации, система также решает важную задачу управления временем пользователей (тайм-менеджмент), позволяя оптимально распределить процесс обучения во времени и минимизировать затраты на повторение материала.

Существуют решения с уже внедрённой методикой интервального повторения, такие как SuperMemo, Anki и Duolingo, которые успешно

используются в образовательных целях и для изучения языков. SuperMemo, разработанная Петром Возняком в 1982 году, предлагает алгоритмы интервального повторения, адаптирующиеся под индивидуальные особенности обучающихся [7]. Приложение Anki, созданное на основе алгоритма SM-2, предоставляет открытый и доступный всем пользователям инструмент с возможностью гибкой настройки интервалов повторения [8]. Duolingo внедряет модель интервального повторения, сочетающую машинное обучение с психолингвистическими подходами, ориентированную преимущественно на изучение языков [9].

Предлагаемое решение отличается от существующих аналогов тем, что ориентировано на универсальность применения и нацелено на поддержку различных типов обучающих материалов. Карточки в приложении создаются с помощью удобного и простого синтаксиса Markdown, в основном используя формат cloze-тестов для активного воспоминания скрытых фрагментов текста. В отличие от других систем с более сложным интерфейсом и неудобным процессом создания карточек, данное веб-приложение обеспечивает адаптивность, интуитивную простоту и лёгкость интеграции в различные образовательные и профессиональные контексты.

Таким образом, разработка предлагаемого веб-приложения является актуальной и востребованной задачей, так как решает реальные проблемы эффективного запоминания информации, способствует оптимизации учебного процесса и может успешно применяться в различных профессиональных и образовательных средах.

1.3 Описание систем-аналогов.

Для анализа были выбраны наиболее популярные системы-аналоги, реализующие подход интервального повторения или схожие функциональные возможности. В этом разделе приведены краткие характеристики каждой системы, их достоинства и недостатки, а также сравнительный анализ.

1.3.1 Anki

Anki – популярная система для интервального повторения, широко

используемая для изучения языков, терминов, медицинской информации и других образовательных материалов, требующих эффективного запоминания. Приложение построено на основе алгоритма SM-2, предложенного Петром Возняком, и предназначено для длительного сохранения знаний путем планомерного увеличения интервалов между повторениями. В Anki предусмотрены различные режимы повторения карточек, статистика прогресса, а также расширенные возможности настройки параметров интервального повторения и внешнего вида карточек. Несмотря на широкие функциональные возможности, пользователи часто отмечают необходимость наличия технических навыков для создания и редактирования карточек, что может стать препятствием для начинающих. [8]

На рисунке 7 приведена экранная форма создания карточки в системе Anki.

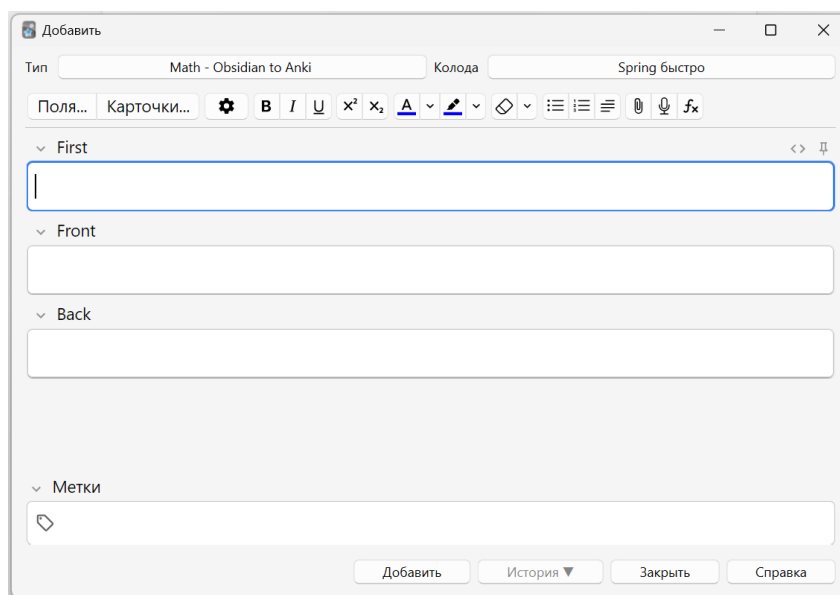


Рисунок 7 – Экранная форма создания карточки в системе Anki

На рисунке 8 приведен режим повторения карточек в Anki.

К достоинствам системы Anki относятся:

- гибкие и адаптивные алгоритмы интервального повторения;
- открытый исходный код;
- поддержка множества платформ (Windows, Linux, MacOS, Android, iOS).

К недостаткам системы относятся:

- неудобный интерфейс для новых пользователей.

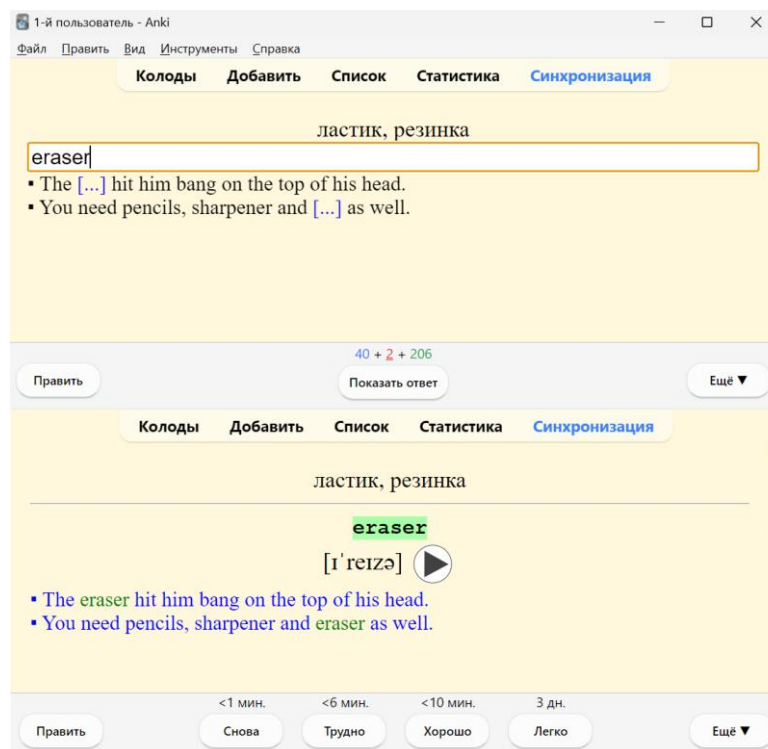


Рисунок 8 – Режим повторения карточек в Anki

– сложный процесс создания карточек с использованием шаблонов на HTML, CSS и JavaScript

1.3.2 StudyStack

StudyStack – онлайн-приложение, позволяющее создавать и использовать обучающие карточки, а также различные образовательные игры и активности на их основе. StudyStack ориентирован на активное повторение информации с помощью различных игровых механик, однако не поддерживает алгоритмы интервального повторения [10].

На рисунке 9 приведена экранная форма создания карточки в StudyStack.

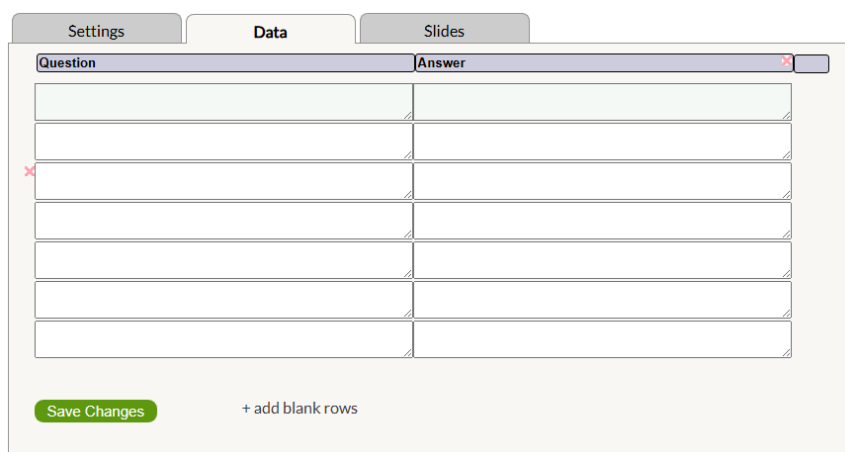


Рисунок 9 – Экран создания карточки в StudyStack

На рисунке 10 приведен режим повторения карточек в StudyStack.

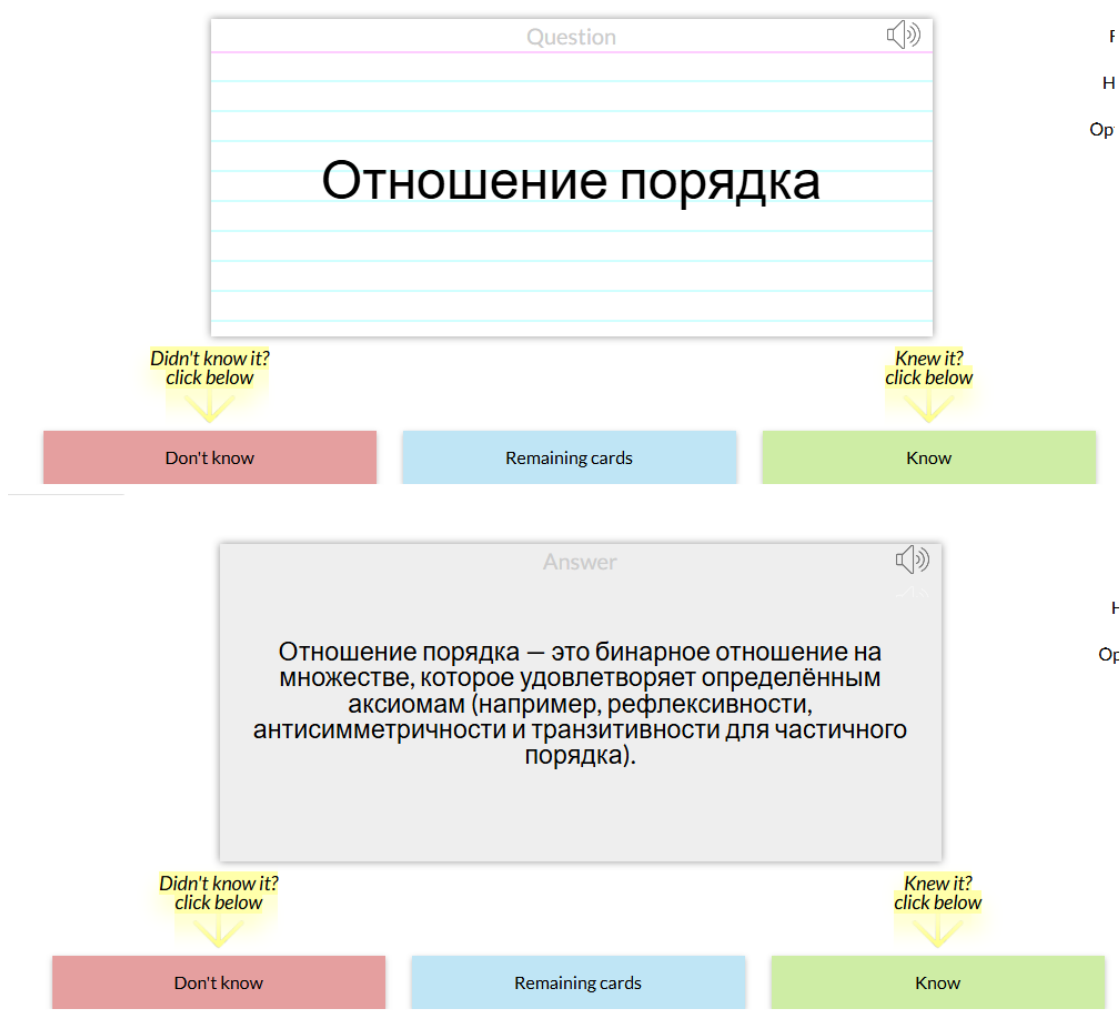


Рисунок 10 – Режим повторения карточек в StudyStack

К достоинствам системы StudyStack относятся:

- простота и доступность создания карточек;
- разнообразие игровых форм для повторения материала.

К недостаткам системы относятся:

- отсутствие алгоритмов интервального повторения;
- маленький размер карточек и неудобное их отображение при повторении;
- ограниченные возможности персонализации и настройки карточек.

1.3.3 Конкурентный анализ систем-аналогов

Для наглядного сравнения выбранных систем-аналогов представлена таблица 1.

Таблица 1 – Сравнительная характеристика систем-аналогов

Показатель	Anki	StudyStack	Разрабатываемое приложение
Интервальное повторение	+	—	+
Удобство создания карточек	—	±	+
Поддержка Markdown-синтаксиса	—	—	+
Использование cloze-тестов	±	—	+
Простота интерфейса	—	±	+
Наличие игрового режима	—	+	—
Возможность тонкой настройки карточек	+	—	-

1.4 Описание автоматизируемого процесса

Разработка веб-приложения для обучения по системе Лейтнера с реализацией алгоритма интервального повторения включает в себя несколько ключевых этапов. Каждый из них направлен на создание эффективной и удобной системы для пользователей, стремящихся к долговременному запоминанию информации.

1 Анализ предметной области и постановка задачи

На начальном этапе проводится исследование существующих методов интервального повторения, включая систему Лейтнера и алгоритм SM-2. Изучаются потребности целевой аудитории, такие как студенты, профессионалы, готовящиеся к квалификационным экзаменам, и сотрудники предприятий. Целью является выявление требований к функциональности системы, обеспечивающей эффективное запоминание и удобство использования.

2 Проектирование архитектуры системы

Система разрабатывается как клиент-серверное веб-приложение с использованием следующих технологий:

- Backend: Java с использованием фреймворка Spring Boot для обработки бизнес-логики и взаимодействия с базой данных;
- Frontend: React для создания интерактивного пользовательского интерфейса;
- API: GraphQL для эффективного обмена данными между клиентом и сервером;
- база данных: PostgreSQL для хранения информации о пользователях, карточках и расписании повторений;
- управление миграциями: Liquibase для контроля версий схемы базы данных.

Архитектура обеспечивает масштабируемость, модульность и возможность дальнейшего расширения функциональности.

3 Реализация алгоритма интервального повторения

В основе системы лежит алгоритм SM-2, адаптированный для автоматического расчета интервалов повторения карточек. Пользователь оценивает свою уверенность в ответе на карточку, и на основе этой оценки система определяет оптимальное время следующего повторения. Это позволяет индивидуализировать процесс обучения и повысить эффективность запоминания.

4 Разработка интерфейса создания и управления карточками

Особое внимание уделяется удобству создания карточек. Используется синтаксис Markdown, что позволяет пользователям легко форматировать текст и создавать cloze-тесты для активного воспоминания. Интерфейс предусматривает предпросмотр карточек и простое редактирование, что снижает порог входа для новых пользователей.

5 Тестирование и отладка системы

Проводится комплексное тестирование всех компонентов системы:

- модульное тестирование: проверка отдельных функций и методов;

- интеграционное тестирование: оценка взаимодействия между различными модулями системы;

- тестирование пользовательского интерфейса: обеспечение удобства и интуитивной понятности интерфейса.

На основе результатов тестирования вносятся необходимые корректировки для обеспечения стабильной и надежной работы приложения.

6 Внедрение

После завершения разработки и тестирования система разворачивается на сервере и становится доступной для пользователей.

Таким образом, автоматизируемый процесс охватывает полный цикл разработки веб-приложения, начиная от анализа требований и заканчивая внедрением системы. Использование современных технологий и адаптация проверенных алгоритмов интервального повторения обеспечивают высокую эффективность и удобство обучения для пользователей.

1.5 Постановка задачи

Цель работы: во время выпускной квалификационной работы необходимо разработать веб-приложение для автоматизации процесса обучения по системе Лейтнера, включающее создание, редактирование и повторение карточек с реализацией алгоритма интервального повторения. Приложение должно обеспечивать автоматический расчёт оптимальных интервалов повторений, отслеживать прогресс по колодам и поддерживать эффективное усвоение информации посредством удобного и простого интерфейса.

Задачи:

- изучить основные понятия предметной области: интервальное повторение, система Лейтнера, флеш-карточки, модель SM-2, процессы обучения и запоминания информации;

- выполнить обзор систем-аналогов в области существующих решений для интервального повторения и обучения с карточками;

- разработать проект автоматизированной системы с использованием методологии структурного и объектно-ориентированного проектирования;

- разработать информационное и программное обеспечение системы, произвести его тестирование и отладку.

Разрабатываемая автоматизированная система должна выполнять следующие функции.

Функции клиентской части:

- удобное создание и редактирование обучающих карточек с использованием markdown-синтаксиса и cloze-тестов;

- создание и удаление колод для обучающих карточек;

- отображение списка карточек и возможность их повторения;

- просмотр текущего прогресса по колодам;

Функции серверной части:

- расчёт оптимальных интервалов повторений на основе алгоритма интервального повторения SM-2;

- хранение информации о пользователях, карточках и расписании повторений;

- обеспечение взаимодействия между клиентской и серверной частями с использованием технологии GraphQL;

- обработка и сохранение данных, переданных с клиентской части;

- управление миграциями базы данных с использованием Liquibase для обеспечения целостности и актуальности схемы данных.

1.6 Выводы по главе.

В данной главе был произведен анализ предметной области: изучены и описаны основные определения в области веб-приложений для автоматизации процесса обучения по системе Лейтнера, актуальность исследования.

Произведен обзор существующих систем-аналогов, сформулирована постановка задачи ВКР и определены основные функции разрабатываемой системы.

2 Проектирование системы

Проектирование информационных систем представляет собой ключевой этап в процессе создания эффективных программных решений. Оно начинается с определения целей проекта и включает в себя разработку архитектуры, компонентов и интерфейсов системы. Цель проектирования — обеспечить требуемую функциональность, производительность, надёжность и безопасность системы, а также её адаптацию к изменяющимся условиям эксплуатации. Качественное проектирование служит основой для создания высокопроизводительных и устойчивых информационных систем.

Современные методологии проектирования информационных систем, такие как структурный анализ и объектно-ориентированное проектирование, позволяют эффективно моделировать бизнес-процессы и разрабатывать системы, соответствующие требованиям пользователей. Использование инструментов моделирования, таких как унифицированный язык моделирования (UML), способствует созданию понятных и точных моделей системы. Это обеспечивает более глубокое понимание требований и упрощает процесс реализации программного обеспечения.

2.1 Выбор и обоснование архитектуры системы

При проектировании веб-приложения для обучения по системе Лейтнера с реализацией алгоритма интервального повторения был выбран подход, сочетающий монолитную трёхуровневую архитектуру с тонким клиентом и использованием GraphQL в качестве API-интерфейса. Такой выбор обусловлен стремлением к упрощению разработки, обеспечению гибкости и эффективности взаимодействия между компонентами системы.

Существуют различные архитектурные подходы к построению веб-приложений, включая монолитную, микросервисную и сервис-ориентированную архитектуры.

Монолитная архитектура предполагает объединение всех компонентов приложения в единую кодовую базу. Это упрощает разработку, тестирование и развертывание системы, особенно на начальных этапах проекта. Монолитные

приложения легче отлаживать и сопровождать, что делает их привлекательными для небольших и средних проектов.

Микросервисная архитектура разделяет приложение на независимые сервисы, каждый из которых отвечает за определённую функциональность. Это обеспечивает гибкость и масштабируемость, но увеличивает сложность разработки и требует значительных ресурсов для координации между сервисами.

Учитывая цели и масштаб разрабатываемого приложения, была выбрана монолитная архитектура, которая позволяет быстрее приступить к разработке и упростить управление проектом на его начальной стадии.

Приложение реализовано в соответствии с трёхуровневой архитектурой, включающей:

- клиентский уровень: предоставляет пользовательский интерфейс и минимальную бизнес-логику, обеспечивая взаимодействие с сервером через API;
- сервер приложений: обрабатывает бизнес-логику, управляет сессиями и обеспечивает безопасность приложения;
- сервер баз данных: отвечает за хранение и управление данными приложения.

Использование тонкого клиента снижает требования к вычислительным ресурсам на стороне пользователя и упрощает обновление интерфейса, поскольку основная логика приложения сосредоточена на сервере.

В качестве API-интерфейса между клиентом и сервером в разрабатываемом приложении используется GraphQL, что предоставляет ряд преимуществ по сравнению с традиционной архитектурой REST. В отличие от REST, где каждый тип данных имеет свой отдельный эндпоинт, GraphQL использует единую точку входа для всех запросов, что упрощает архитектуру API и снижает количество необходимых HTTP-запросов.

Одним из ключевых преимуществ GraphQL является возможность клиенту запрашивать только те данные, которые ему необходимы. Это позволяет избежать проблемы избыточной или недостаточной выборки данных, характерной для REST, где часто приходится получать больше информации, чем

требуется, или делать несколько запросов для получения всех необходимых данных. Кроме того, GraphQL поддерживает вложенные запросы, что позволяет получать связанные данные в рамках одного запроса, сокращая количество обращений к серверу и уменьшая нагрузку на сеть. Такая гибкость особенно полезна в приложениях с динамическими и сложными структурами данных.

В заключение, выбранная архитектура — монолитное трёхуровневое приложение с тонким клиентом и использованием GraphQL — обеспечивает баланс между простотой разработки, эффективностью и возможностью масштабирования в будущем.

2.2 Проект системы

2.2.1 Структурная схема системы

Структурная схема информационной системы представляет собой графическое отображение её компонентов и взаимосвязей между ними [11]. Она служит инструментом для визуализации архитектуры системы, облегчая понимание её структуры и функционирования. Согласно определению, структурная схема — это совокупность элементарных звеньев объекта и связей между ними, один из видов графической модели.

В контексте проектирования информационных систем, структурная схема помогает разработчикам и аналитикам определить основные компоненты системы, их функции и способы взаимодействия. Это особенно важно при разработке сложных систем, где необходимо обеспечить согласованную работу различных подсистем и модулей. Использование структурных схем способствует выявлению потенциальных узких мест и оптимизации процессов внутри системы.

На рисунке 11 приведена структурная схема разрабатываемой системы, в ее состав входят клиентское и серверное приложения, которые взаимодействуют между собой с помощью протокола HTTP и технологии GraphQL.

В состав клиентского приложения входят следующие подсистемы:

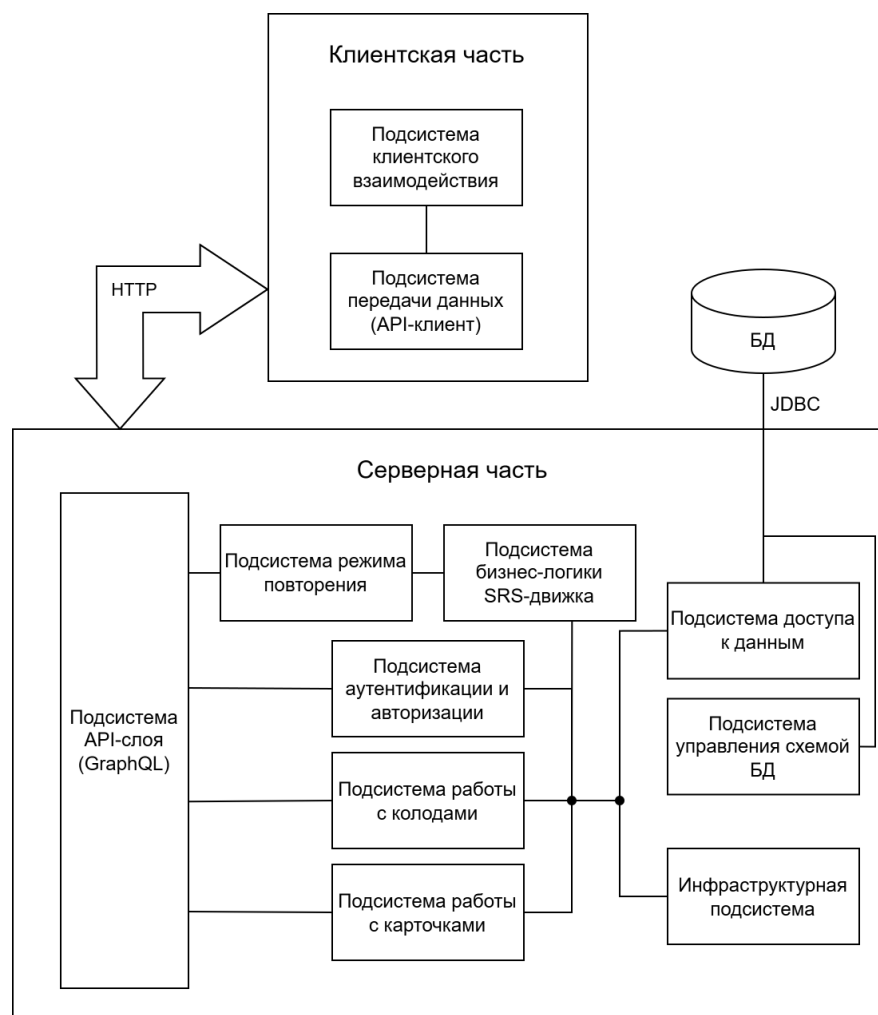


Рисунок 11 – Структурная схема системы

– подсистема клиентского взаимодействия, обеспечивающая пользователям возможность просматривать интерфейс, взаимодействовать с элементами управления и перемещаться между страницами приложения;

– подсистема передачи данных, отвечающая за коммуникацию между клиентской и серверной частями приложения;

В состав серверного приложения входят следующие подсистемы:

– подсистема аутентификации и авторизации, реализующая вход, регистрацию пользователей и управление доступом к функционалу системы;

– подсистема бизнес-логики SRS-движка, которая обеспечивает реализацию алгоритма интервального повторения и управление процессом повторения карточек;

- подсистема обработки пользовательских запросов, принимающая и обрабатывающая запросы пользователей, и предоставляющая необходимые данные клиентской части;
- подсистема режима повторения, позволяющая пользователям повторять и закреплять изученный материал в соответствии с алгоритмом интервального повторения;
- подсистема управления карточками, предоставляющая пользователям возможность создавать, редактировать и удалять обучающие карточки;
- подсистема управления колодами, позволяющая объединять карточки в тематические группы для удобного изучения и организации материала;
- подсистема хранения и обработки данных, отвечающая за сохранение и предоставление информации, используемой в приложении;
- подсистема управления схемой базы данных, обеспечивающая инициализацию и поддержку актуальности структуры базы данных;
- инфраструктурная подсистема, отвечающая за базовые настройки и конфигурацию среды выполнения приложения.

2.2.2 Диаграмма вариантов использования

Для иллюстрации функциональных требований к системе на концептуальном уровне используется диаграмма вариантов использования, позволяющая наглядно отразить ключевые сценарии взаимодействия пользователей и компонентов приложения.

Диаграмма вариантов использования (use case diagram) — это поведенческая диаграмма в языке UML, отображающая отношения между акторами (пользователями или внешними системами) и прецедентами (вариантами использования), что позволяет описать функциональные требования системы на концептуальном уровне [12]. Такая диаграмма показывает, какие сервисы предоставляет система и каким образом они используются внешними сущностями, не вдаваясь в детализацию внутренней реализации [13].

На рисунке 12 представлена диаграмма прецедентов (use case), демонстрирующая виды взаимодействия акторов с Java-приложением и определяющая основные варианты использования.

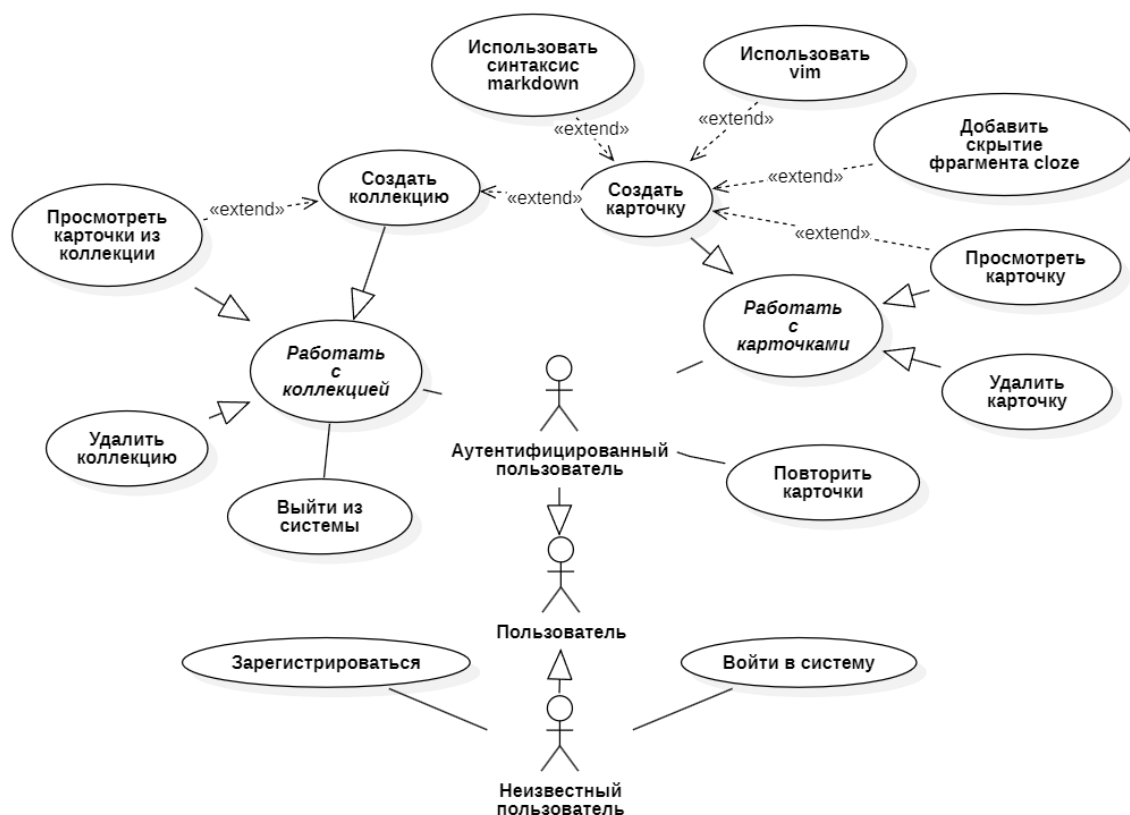


Рисунок 12 – Диаграмма вариантов использования

2.2.3 Диаграмма деятельности

Диаграмма активностей является одним из ключевых инструментов визуального моделирования поведения системы в нотации UML (Unified Modeling Language). Она используется для описания потоков управления и деятельности в системе, демонстрируя, какие действия выполняются и в каком порядке. Такие диаграммы помогают понять, как ведёт себя система в ответ на определённые события или сценарии использования, что особенно полезно на этапе проектирования программного обеспечения.

Согласно определению, диаграмма активностей — это поведенческая диаграмма, которая моделирует динамический аспект системы, отображая последовательность действий, управляющие переходы, условия ветвления и

параллельные процессы [14]. Она может использоваться для описания как бизнес-процессов, так и поведения отдельных компонентов программной системы. В UML 2.5 диаграммы активностей считаются разновидностью state-machine диаграмм, адаптированных для моделирования рабочих процессов и потоков управления в программных системах.

Использование диаграммы активностей особенно оправдано в задачах, связанных с описанием логики пользовательского взаимодействия или бизнес-логики процессов. Благодаря её наглядности и последовательному отображению шагов выполнения, такие диаграммы позволяют эффективно коммуницировать между аналитиками, разработчиками и другими участниками проекта, минимизируя риск недопонимания требований и поведения системы.

На рисунке 13 представлена диаграмма деятельности системы.

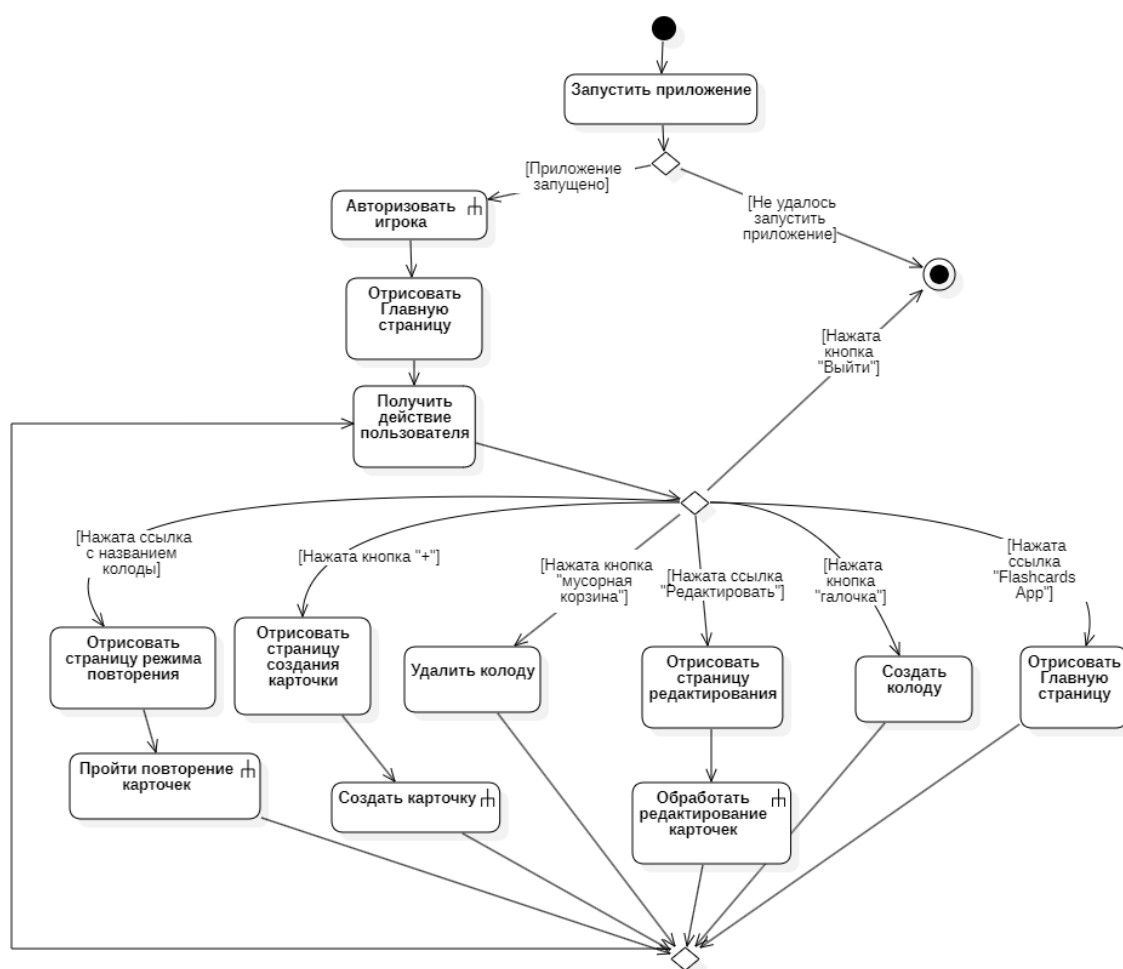


Рисунок 13 – Диаграмма деятельности

2.2.4 Диаграмма последовательности

Диаграмма последовательностей — это один из основных типов поведенческих диаграмм UML, предназначенный для отображения взаимодействия между объектами в определённой временной последовательности. Она показывает, какие сообщения (вызовы методов, сигналы и т.п.) передаются между участниками системы в рамках конкретного сценария. Такой тип диаграммы помогает понять, как реализуется логика взаимодействия компонентов и как протекает выполнение функциональности во времени.

Согласно спецификации UML, диаграмма последовательностей описывает поведение объектов, упорядоченных по горизонтали, и их взаимодействие по времени, представленному по вертикали. Основными элементами диаграммы являются объекты (участники взаимодействия), линии жизни и сообщения. Она наглядно демонстрирует последовательность вызовов, создание и удаление объектов, а также может отображать условные ветвления и циклы [14].

На рисунке 14 отображена диаграмма последовательности для варианта использования «Повторить карточки».

2.2.5 Диаграмма классов, логический уровень

Диаграмма классов — это один из важнейших инструментов проектирования логической модели данных, который наглядно отражает структуру проектируемой информационной системы. В рамках логического проектирования она позволяет визуализировать классы (сущности предметной области), их атрибуты, методы, а также отношения между ними, такие как ассоциации, агрегация и наследование. Диаграмма классов на этом этапе не привязывается к конкретным техническим средствам реализации, а предназначена для чёткого понимания структуры и взаимосвязей элементов модели.

Использование диаграммы классов в логическом проектировании помогает разработчикам и аналитикам уточнить требования, выявить

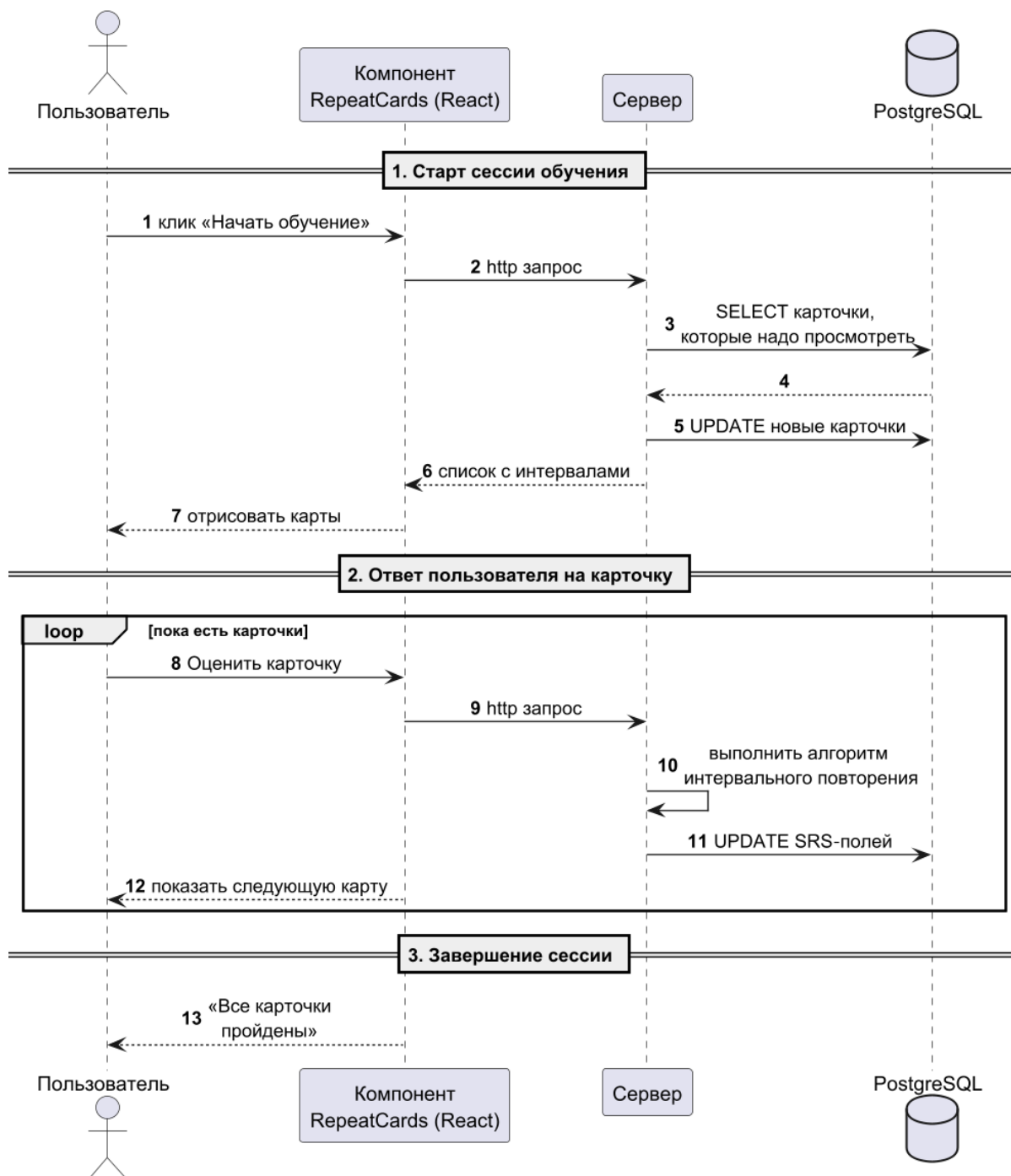


Рисунок 14 – Диаграмма последовательности для варианта использования «Повторить карточки»

зависимости и ограничения, а также обеспечить согласованность структуры будущей системы. Такая диаграмма является фундаментом для последующего проектирования и реализации базы данных, интерфейсов и программной логики приложения. Её использование способствует снижению вероятности ошибок и упрощает коммуникацию между всеми участниками разработки [14].

На рисунке 15 отображена диаграмма классов на логическом уровне.

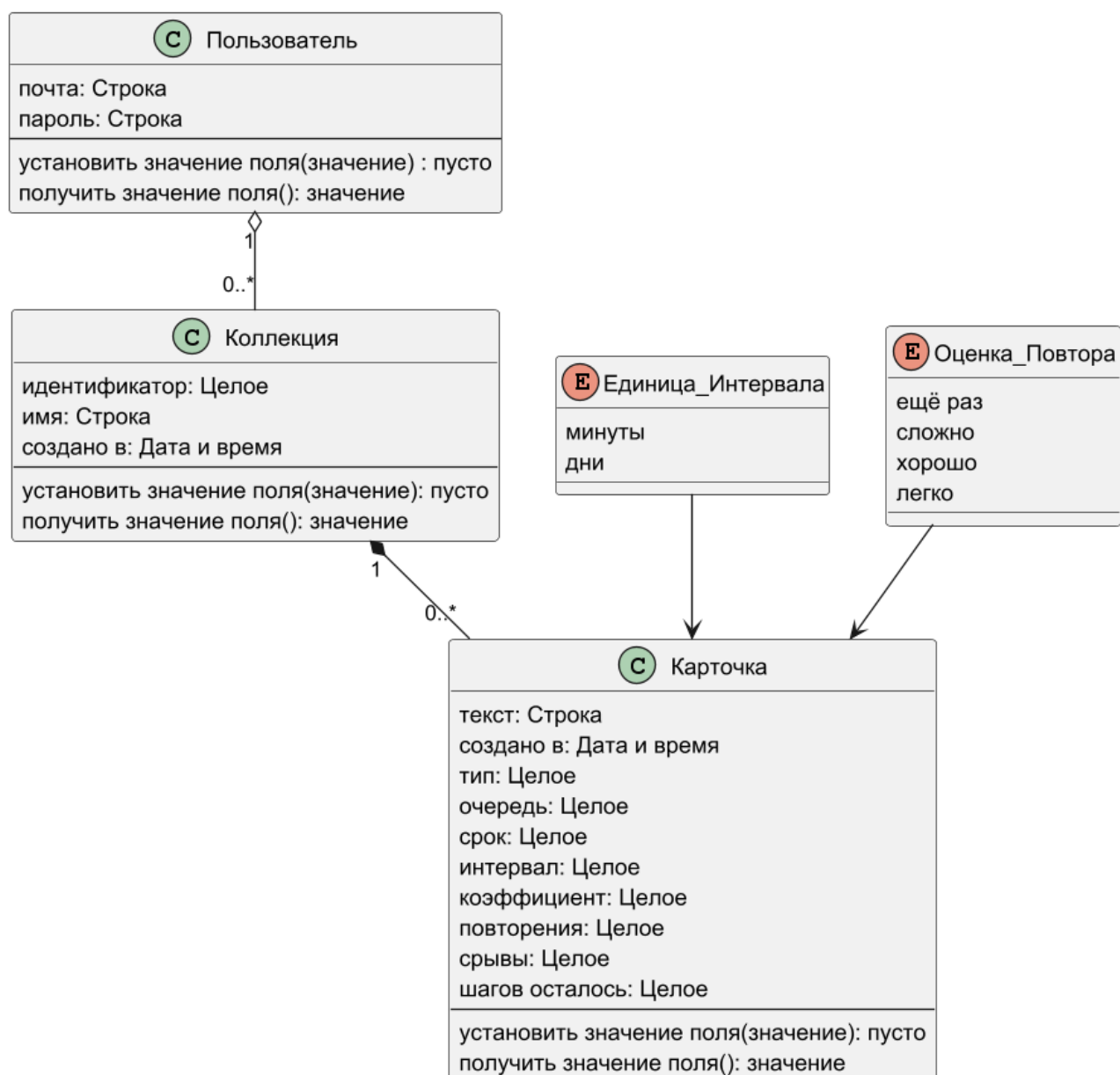


Рисунок 15 – Диаграмма классов на логическом уровне

2.3 Выбор и обоснование средств реализации

В качестве основного фреймворка для серверной части приложения выбран Spring Boot. Это обусловлено его удобством и эффективностью при разработке веб-приложений на платформе Java, встроенной поддержкой различных интеграций и зависимостей, а также минимальным временем на конфигурацию и развёртывание приложения. Spring Boot предоставляет готовые решения для работы с REST и GraphQL API, управления транзакциями, безопасностью и многими другими аспектами разработки.

Для реализации серверной логики используется язык Java версии 17. Эта версия обеспечивает высокую производительность, надёжность и удобство работы, предлагая последние улучшения языка и среды выполнения. Применение актуальной версии Java позволяет эффективно использовать новые возможности платформы, такие как улучшенная работа с типами данных, оптимизированная сборка мусора и повышенная стабильность приложений.

GraphQL был выбран в качестве основного интерфейса для взаимодействия между клиентом и сервером, так как предоставляет гибкий и мощный подход к обмену данными. GraphQL позволяет клиенту самостоятельно запрашивать необходимые данные в удобном для него формате, уменьшая объём передаваемой информации и число запросов к серверу. [15] Это существенно снижает нагрузку на сеть и упрощает управление API, обеспечивая единую точку входа для всех запросов приложения.

В качестве системы управления базами данных выбрана PostgreSQL, одна из наиболее распространённых реляционных СУБД с открытым исходным кодом. PostgreSQL предоставляет мощные возможности для хранения и обработки данных, поддерживает транзакции, сложные SQL-запросы и обладает высокой производительностью и надёжностью. Также она позволяет гибко масштабировать хранилище данных при увеличении нагрузки на приложение.

Для управления версиями исходного кода и организации совместной разработки используется GitHub. Эта платформа облегчает взаимодействие между разработчиками, позволяет эффективно отслеживать изменения, проводить ревью кода и интегрировать автоматические тесты и сборку приложений с помощью GitHub Actions. Таким образом, GitHub обеспечивает надёжность, прозрачность и удобство разработки.

Docker Compose выбран в качестве инструмента для развёртывания приложения и управления его окружением. Docker Compose позволяет быстро и удобно разворачивать все компоненты приложения в контейнерах, обеспечивая консистентность среды разработки и лёгкость деплоя на серверы. [16] Применение Docker Compose значительно упрощает процесс развёртывания,

обновления и поддержки приложения в различных средах.

Для реализации клиентской части выбран фреймворк React. Он обеспечивает создание динамического и интерактивного пользовательского интерфейса, обладает высокой производительностью и удобством разработки благодаря компонентному подходу и широкому сообществу разработчиков. React позволяет быстро реализовывать сложные пользовательские интерфейсы, обеспечивая при этом удобство и простоту поддержки кода приложения.

2.4 Выводы по главе

В данной главе было выполнено проектирование разрабатываемого веб-приложения для обучения по системе Лейтнера с реализацией алгоритма интервального повторения с использованием методологий структурного проектирования, объектно-ориентированного проектирования и языка UML. Была описана архитектура системы, приведена структурная схема, а также создан UML-проект, включающий необходимые диаграммы. Представлена логическая модель данных системы и описан комплекс программных средств для реализации.

3 Реализация системы

3.1 Описание интерфейса пользователя

3.1.1 Начало работы: регистрация и вход

Для работы с веб-приложением пользователь должен пройти процедуру регистрации, указав email, пароль и подтверждение пароля. Требования к регистрации:

- email должен соответствовать формату электронной почты (проверка через регулярное выражение: $^{\wedge}\backslash S+@ \backslash S+ \backslash . \backslash S+ \$$);
- пароль должен содержать не менее 6 символов;
- при успешной регистрации отображается сообщение: «Пользователь успешно зарегистрирован!».

На рисунке 16 приведена форма регистрации.

Flashcards App

Регистрация

Email

Пароль

Подтверждение пароля

Зарегистрироваться

Рисунок 16 – Страница регистрации

После успешной регистрации можно выполнить вход в систему указав логин и пароль. На рисунке 17 приведена форма для входа.

Вход

Email

Пароль

Войти

Рисунок 17 – Страница входа

3.1.2 Работа с коллекциями карточек

После входа пользователь попадает на главную страницу, где отображается список всех его коллекций карточек. Здесь отображаются следующие данные для каждой коллекции:


- название коллекции;
- количество всех карточек;
- количество новых;
- количество карточек в стадии «обучения»;
- количество карточек; готовых к повторению.

На рисунке 18 изображена главная страница.

Flashcards App

Выйти Редактировать

Список коллекций пользователя

Коллекция	Всего	Новые	Learning	К повтор.	Действия
Java	0	0	0	0	+ 




Рисунок 18 – Главная страница

Также на этой странице реализованы следующие функции:

- создание новой коллекции;
- удаление существующих коллекций;
- переход к добавлению карточек в коллекцию;
- запуск режима повторения для выбранной коллекции.

3.1.3 Добавление и редактирование карточек

Выбрав коллекцию, пользователь может добавить в неё новые карточки.

Редактор поддерживает:

- ввод текста карточки с использованием Markdown-разметки;
- выделение фрагментов для последующего скрытия («cloze»-режим) — позволяет создавать карточки с пропусками, которые надо вспомнить при повторении;
- предпросмотр карточки в реальном времени;
- включение Vim-режима для продвинутых пользователей.

На рисунке 19 изображена страница для создания карточек.

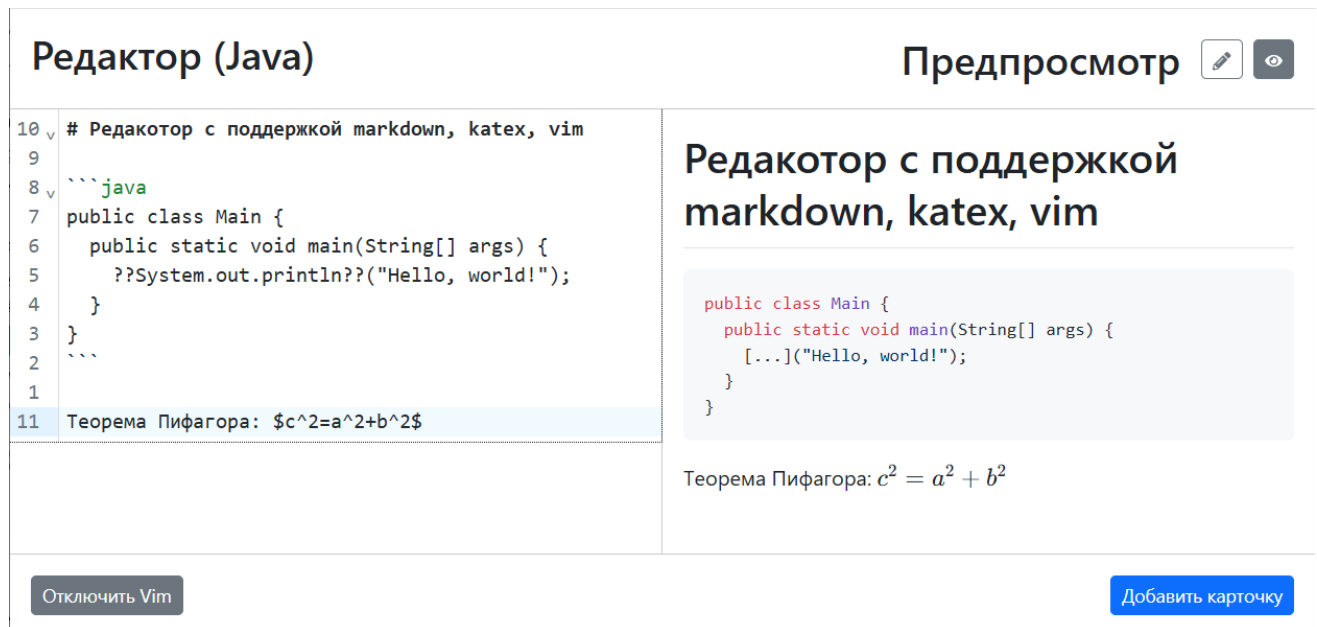


Рисунок 19 – Страница создания карточек

После ввода текста карточки пользователь сохраняет её в коллекции нажатием кнопки «Добавить карточку».

3.1.4 Редактирование коллекций и поиск карточек

В разделе редактирования коллекций пользователь может:

- просматривать и фильтровать все свои коллекции;
- просматривать список всех карточек в выбранной коллекции;
- выполнять сортировку и поиск по содержимому карточек;
- просматривать содержимое отдельной карточки в Markdown-формате.

На рисунке 20 представлена страница редактирования коллекций.

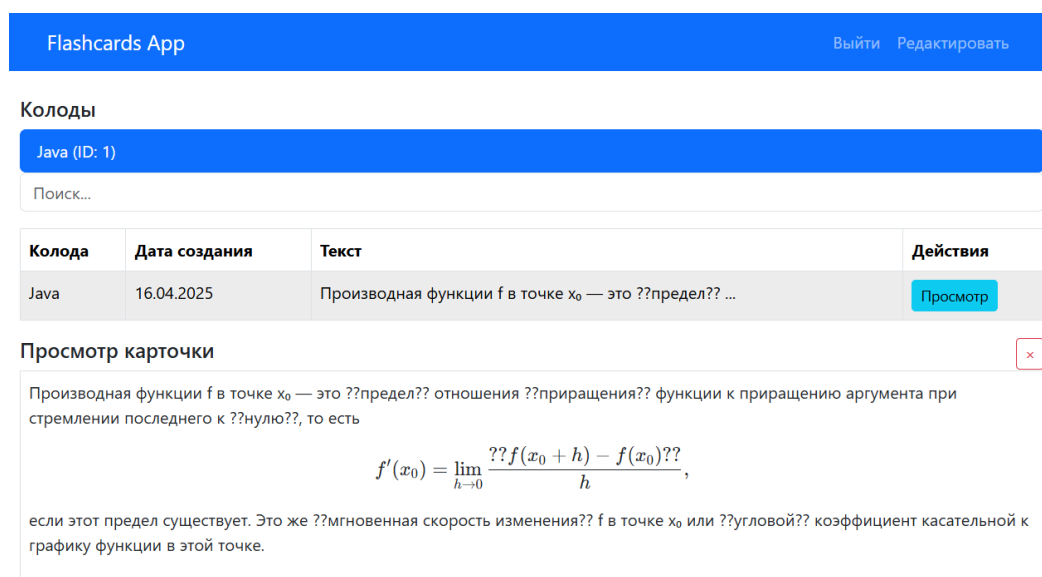


Рисунок 20 – Страница редактирования коллекций

3.1.5 Повторение карточек

Основная функция системы — проведение интервального повторения по алгоритму Лейтнера. При запуске повторения:

- пользователю последовательно показываются карточки из выбранной коллекции;
- карточки автоматически сортируются по приоритету показа: сначала новые; затем карточки, находящиеся в стадии обучения, и далее — карточки на повторение,
- для cloze-карточек часть информации скрыта и открывается по нажатию Tab.

После просмотра карточки пользователь выбирает оценку ответа (кнопки:

«Again»; «Hard», «Good», «Easy»), что влияет на дату следующего показа по алгоритму интервального повторения.

На рисунке 21 изображена страница режима повторения.

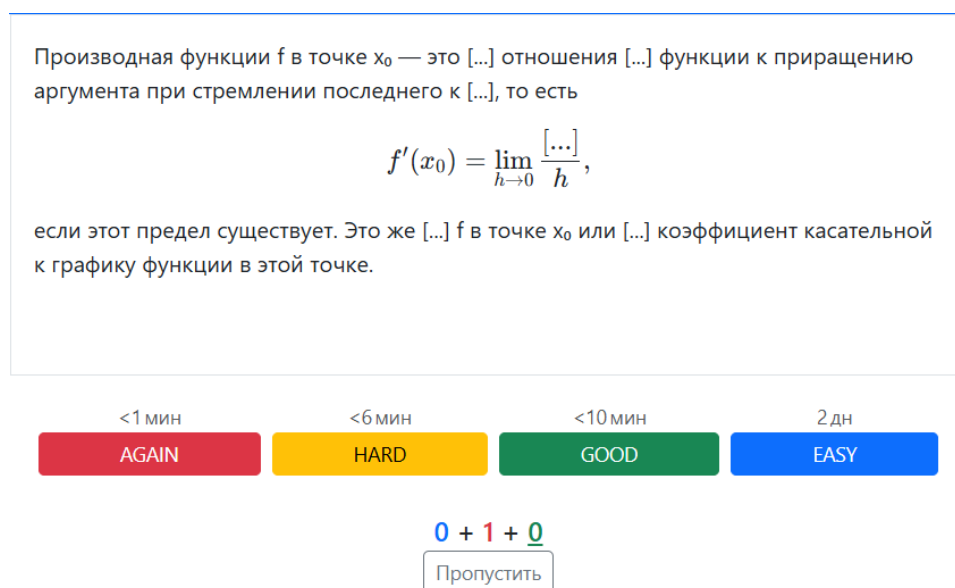


Рисунок 21 – Страница режима повторения

3.2 Физическая модель данных.

Физическая модель данных представляет собой детализированное описание структуры базы данных, которое непосредственно реализуется в системе управления базами данных (СУБД). На этом этапе определяется конкретный набор таблиц, типов данных, первичных и внешних ключей, индексов, ограничений и связей между таблицами. Для отображения физической модели данных будет использована ER-диаграмма, на которой наглядно представлены сущности и их взаимосвязи в контексте базы данных.

На рисунке 22 представлена ER-диаграмма разработанного приложения. Кроме того, для дополнительного удобства и более тесной интеграции с программным кодом приложения будет применяться диаграмма классов, которая позволит представить структуру данных в терминах объектно-ориентированного подхода. Использование диаграммы классов для отображения физической модели данных упрощает разработчикам понимание и реализацию модели, так как она более близка к структуре используемых в приложении классов и объектов, тем самым обеспечивая лучшее взаимодействие между бизнес-логикой приложения и хранилищем данных.

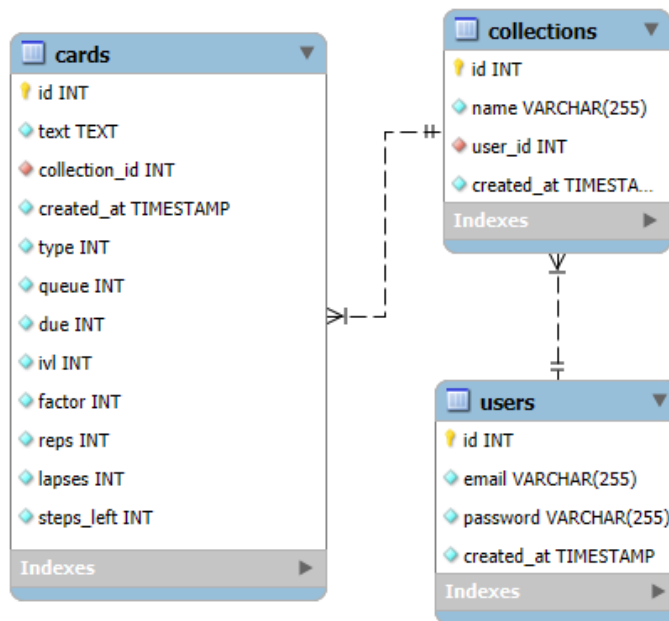


Рисунок 22 – ER диаграмма

На рисунке 23 представлена диаграмма классов разработанного приложения.

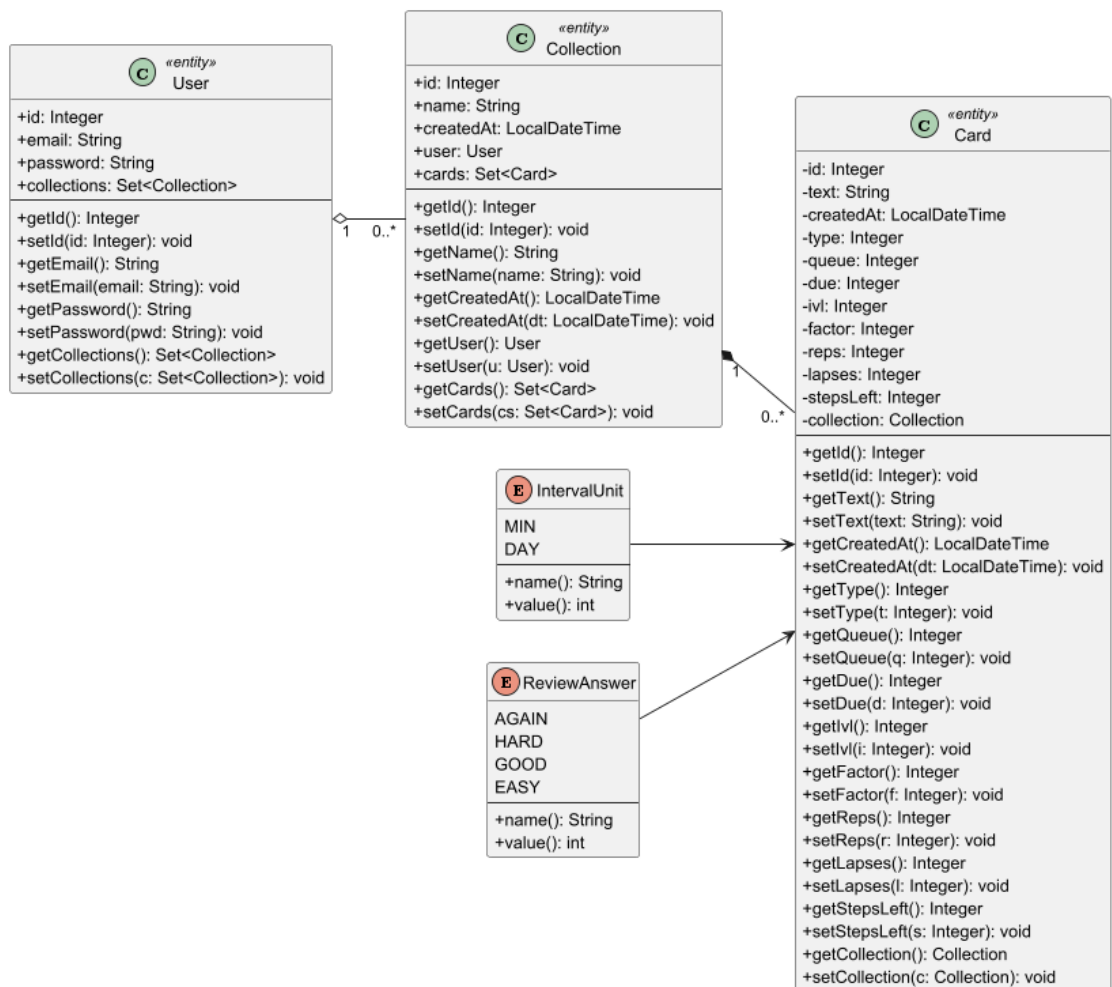


Рисунок 23 – Диаграмма классов

3.3 Выводы по главе

В данной главе была подробно рассмотрена реализация разрабатываемой информационной системы для обучения по системе Лейтнера с алгоритмом интервального повторения. Было описано взаимодействие пользователя с приложением, включая процедуры регистрации и авторизации, управление коллекциями и карточками, а также проведение сеансов повторения. Интерфейс системы был разработан с акцентом на удобство пользователя, поддержку Markdown-разметки и возможность использования cloze-тестов для эффективного запоминания материала.

Также была представлена физическая модель данных, реализованная в конкретной СУБД — PostgreSQL. Для визуализации и документирования структуры хранения данных использованы ER-диаграмма и диаграмма классов, позволяющие наглядно отразить структуру и связи между сущностями приложения.

Таким образом, реализованные в ходе данной главы решения обеспечивают простоту использования, эффективное взаимодействие пользователя с приложением и соответствуют сформулированным на этапе проектирования требованиям. Использование наглядных диаграмм облегчает понимание структуры приложения, способствуя дальнейшему сопровождению и развитию системы.

ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной работы была разработано веб-приложение для обучения по системе Лейтнера с реализацией алгоритма интервального повторения.

В первом разделе были приведены основные понятия и определения предметной области обучения по системе Лейтнера и алгоритму интервального повторения с применением флэш карточек, приведены характеристики систем-аналогов, на основании этого была сформулирована постановка задачи и основные требования к системе.

Во втором разделе была разработана структура системы, разработан проект системы на языке, логическая модель данных, а также был выбран комплекс программных средств.

В третьем разделе описан интерфейс пользователя, физическая модель данных системы, приведены примеры использования системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Leitner System: The Most Effective Way to Revise [Электронный ресурс]. URL: <https://www.lecturio.com/blog/revamp-your-revision-with-the-leitner-system> (дата обращения: 14.05.2025).

2 Метод интервального повторения иностранных слов [Электронный ресурс]. URL: <https://sportzania.ru/about/publikatsii/metod-intervalnogo-povtoreniya-inostrannykh-slov> (дата обращения: 14.05.2025).

3 Интервальные повторения [Электронный ресурс] // Википедия: свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Интервальные_повторения (дата обращения: 14.05.2025).

4 Spaced repetition [Электронный ресурс] // Wikipedia: the free encyclopedia. URL: https://en.wikipedia.org/wiki/Spaced_repetition (дата обращения: 14.05.2025).

5 Kerfoot B.P. et al. Spaced education improves the retention of clinical knowledge: a randomized controlled trial [Электронный ресурс]. URL: <https://medicine.wright.edu/sites/medicine.wright.edu/files/page/attachments/Kerfoot%20etal%20spaced%20education%20retention%20knowledge%20MedEduc2007.pdf> (дата обращения: 19.05.2025).

6 Cloze test [Электронный ресурс] // Википедия: свободная энциклопедия. URL: https://en.wikipedia.org/wiki/Cloze_test (дата обращения: 14.05.2025).

7 Официальный сайт SuperMemo [Электронный ресурс]. URL: <https://www.supermemo.com/ru/supermemo-app> (дата обращения 14.05.2025).

8 Официальный сайт Anki [Электронный ресурс]. URL: <https://apps.ankiweb.net/> (дата обращения: 14.05.2025).

9 Официальный сайт Duolingo [Электронный ресурс]. URL: <https://ru.duolingo.com/> (дата обращения 14.05.2025).

10 Официальный сайт Study Stack [Электронный ресурс]. URL: <https://www.studystack.com/> (дата обращения: 14.05.2025).

11 Структурная схема [Электронный ресурс] // Википедия: свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Структурная_схема (дата

обращения: 20.05.2025).

12 Диаграмма прецедентов [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Диаграмма_прецедентов (дата обращения: 12.12.2024).

13 Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Электронный ресурс]. URL: <https://habr.com/ru/articles/566218/> (дата обращения: 12.12.2024).

14 OMG Unified Modeling Language (OMG UML), Version 2.5.1 [Электронный ресурс]. URL: <https://www.omg.org/spec/UML/2.5.1> (дата обращения: 20.05.2025).

15 Официальный сайт GraphQL [Электронный ресурс]. URL: <https://graphql.org/> (дата обращения: 14.05.2025)

16 Официальный сайт Docker [Электронный ресурс]. URL: <https://www.docker.com/> (дата обращения: 14.05.2025)

17 Буч Г. Язык UML. Руководство пользователя: Пер. с англ. / Г. Буч, Д. Рамбо, Б. Джекобсон. М.: ДМК-Пресс, 2001. 432 с.

18 Коварцев А.Н. Автоматизация тестирования программного обеспечения учебное пособие. Самара: СГАУ, 2010. 122 с.

19 Монолитная и микросервисная архитектура [Электронный ресурс]. URL: <https://habr.com/ru/companies/haulmont/articles/758780/> (дата обращения: 14.05.2025)

20 UML: Class Diagram [Электронный ресурс]. URL: <https://shesterov.by/tpost/pgu0yucas1-uml-class-diagram> (дата обращения: 14.05.2025)

21 UML: Activity Diagram [Электронный ресурс]. URL: <https://shesterov.by/tpost/s30fpt0j11-uml-activity-diagram> (дата обращения: 14.05.2025)

22 UML: Use Case Diagram [Электронный ресурс]. URL: <https://shesterov.by/tpost/g9mg3zhpi1-uml-use-case-diagram> (дата обращения: 14.05.2025)

ПРИЛОЖЕНИЕ А

Руководство пользователя

А.1 Назначение системы

Данная система предназначена для создания флэш карточек и изучения с их помощью любой информации. Используется методика cloze-тестов, когда важная информация скрывается и обучающийся должен восстановить ее из памяти. Также применяется алгоритм интервального повторения, который определяет минимальное количество повторений и интервалы для успешного запоминания информации.

А.2 Условия работы системы

Для корректной работы системы необходимо наличие соответствующих программных и аппаратных средств.

1) Требования к техническому обеспечению:

- ЭВМ типа IBM PC;
- процессор типа x86 или x64 тактовой частоты 1400 МГц и выше;
- клавиатура или иное устройство ввода;
- мышь или иное манипулирующее ввода;
- дисплей с разрешением не менее 1280×768 пикселей;
- широкополосное подключение к сети Интернет, не менее 1

Мб/сек.

2) Требования к программному обеспечению:

- операционная система Windows 10 и выше;
- подсистема линукс для windows (WSL);
- docker for desktop.

А.3 Установка системы

Для локального запуска системы достаточно иметь исходный код проекта. Необходимо запустить терминал и перейти командой `cd` в директорию с исходным кодом. В данной директории необходимо прописать команду `docker compose up`. Дождаться запуска приложения. После данных манипуляций

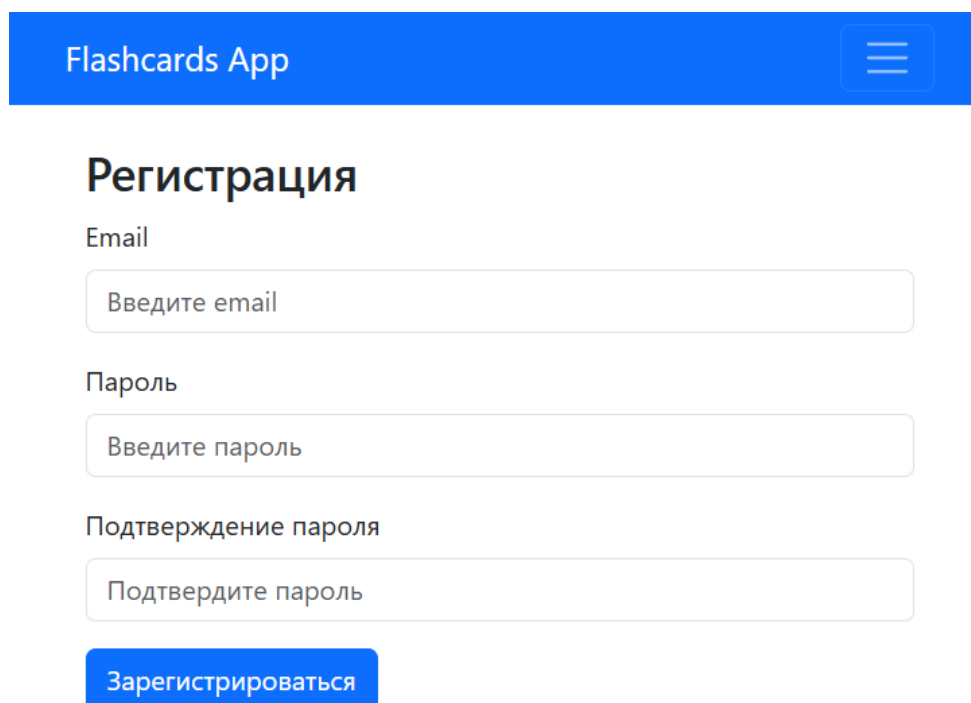
система будет доступна в браузере по ссылке <http://localhost:3000>.

А.4 Работа с системой

1. Начало работы: регистрация и вход

Для начала работы с веб-приложением вам необходимо зарегистрироваться, указав действительный адрес электронной почты, пароль и подтверждение пароля. Убедитесь, что введенный email соответствует формату электронной почты (например, `user@example.com`), а пароль содержит не менее 6 символов. При успешном завершении регистрации появится сообщение: «Пользователь успешно зарегистрирован!».

На рисунке А1 показана форма регистрации.



Flashcards App

Регистрация

Email

Пароль

Подтверждение пароля

Зарегистрироваться

Рисунок А1 – Страница регистрации

После успешной регистрации вы можете войти в систему, используя указанные при регистрации email и пароль. Форма входа представлена на рисунке А2.

2. Работа с коллекциями карточек

После входа вы попадаете на главную страницу приложения, где представлен список ваших коллекций карточек. Для каждой коллекции отображается название и статистика, включающая общее количество карточек, количество новых карточек, карточек на стадии обучения и карточек, готовых к

Вход

Email

Пароль

Войти

Рисунок А2 – Страница входа

повторению. Главная страница также предоставляет возможность создать новую коллекцию, удалить ненужные коллекции, добавить карточки в выбранную коллекцию и запустить режим повторения карточек для выбранной коллекции.

Главная страница приложения представлена на рисунке А3.

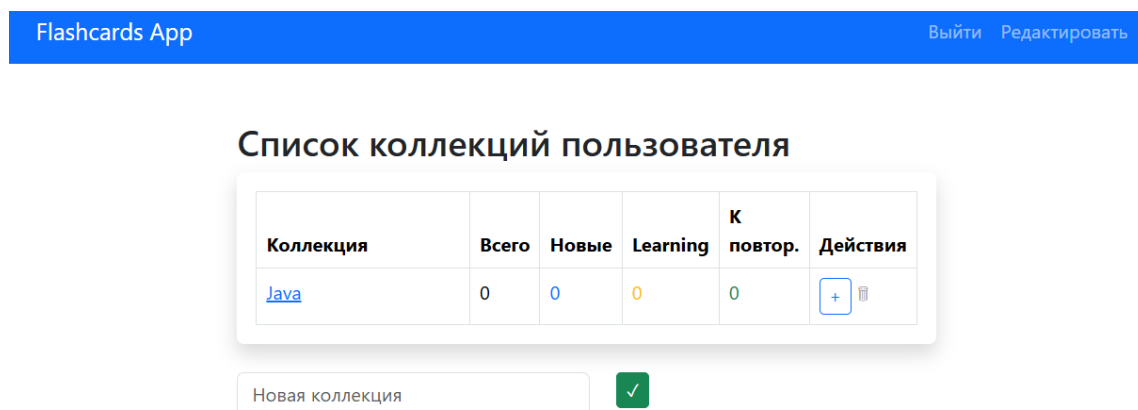


Рисунок А3 – Главная страница

3. Добавление и редактирование карточек

Для добавления новых карточек необходимо выбрать нужную коллекцию и открыть редактор карточек. Редактор поддерживает создание текста с использованием Markdown-разметки, выделение отдельных фрагментов текста для скрытия (режим cloze), что позволяет создавать карточки с пропущенными элементами для более эффективного обучения. Также доступен предпросмотр карточек в режиме реального времени и возможность активации Vim-режима для

опытных пользователей. После подготовки карточки необходимо нажать кнопку «Добавить карточку», чтобы сохранить её в коллекции.

Страница редактора карточек представлена на рисунке А4.

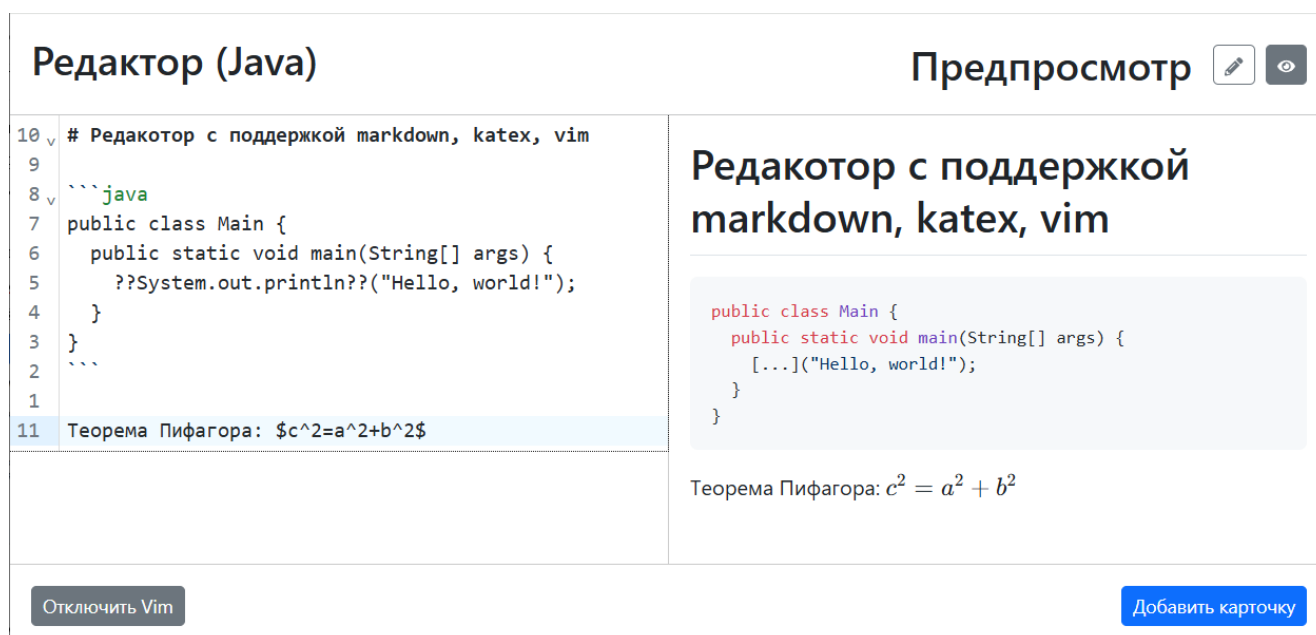


Рисунок А4 – Страница создания карточек

После ввода текста карточки пользователь сохраняет её в коллекции нажатием кнопки «Добавить карточку».

4. Редактирование коллекций и поиск карточек

В разделе редактирования коллекций вы можете просматривать список всех ваших коллекций с возможностью применения фильтров для поиска. Также доступен просмотр всех карточек в выбранной коллекции с возможностью сортировки и поиска по содержимому карточек. Кроме того, вы можете просматривать содержимое каждой отдельной карточки в формате Markdown.

Страница редактирования коллекций представлена на рисунке А5.

5. Повторение карточек

Основная задача приложения — повторение карточек по алгоритму Лейтнера, обеспечивающему эффективное запоминание информации. В режиме повторения вам последовательно показываются карточки из выбранной коллекции. Карточки автоматически распределяются по приоритету: сначала

показываются новые карточки, затем карточки на стадии обучения, а далее карточки, готовые к повторению.

Flashcards App Выйти Редактировать

Колоды

Java (ID: 1)

Поиск...

Колода	Дата создания	Текст	Действия
Java	16.04.2025	Производная функции f в точке x_0 — это ??предел?? ...	Просмотр

Просмотр карточки ✕

Производная функции f в точке x_0 — это ??предел?? отношения ??приращения?? функции к приращению аргумента при стремлении последнего к ??нулю??, то есть

$$f'(x_0) = \lim_{h \rightarrow 0} \frac{f(x_0 + h) - f(x_0)}{h},$$

если этот предел существует. Это же ??мгновенная скорость изменения?? f в точке x_0 или ??угловой?? коэффициент касательной к графику функции в этой точке.

Рисунок А5 – Страница редактирования коллекций

Для карточек, созданных в режиме cloze, скрытые части открываются нажатием клавиши Tab. После изучения каждой карточки вы оцениваете свой ответ с помощью кнопок: «Again», «Hard», «Good», «Easy». Ваша оценка влияет на планирование следующего повторения карточки согласно алгоритму интервального повторения.

Страница режима повторения представлена на рисунке А6.

На рисунке А6 изображена страница режима повторения.

Производная функции f в точке x_0 — это [...] отношения [...] функции к приращению аргумента при стремлении последнего к [...], то есть

$$f'(x_0) = \lim_{h \rightarrow 0} \frac{[...]}{h},$$

если этот предел существует. Это же [...] f в точке x_0 или [...] коэффициент касательной к графику функции в этой точке.

<1 мин <6 мин <10 мин 2 дн

AGAIN HARD GOOD EASY

0 + 1 + 0

Пропустить

Рисунок А6 – Страница режима повторения

ПРИЛОЖЕНИЕ Б

Код программы

apolloClient.js

```
import { ApolloClient, InMemoryCache, createHttpLink } from '@apollo/client';  
import { setContext } from '@apollo/client/link/context';
```

```
const httpLink = createHttpLink({  
  uri: 'http://localhost:8080/graphql',  
});
```

```
const authLink = setContext((_, { headers }) => {  
  const token = localStorage.getItem('token');  
  return {  
    headers: {  
      ...headers,  
      authorization: token ? `Bearer ${token}` : "",  
    },  
  };  
});
```

```
const client = new ApolloClient({  
  link: authLink.concat(httpLink),  
  cache: new InMemoryCache(),  
});
```

```
export default client;
```

App.css

```
.App {
```

```
text-align: center;
}
```

```
.App-logo {
  height: 40vmin;
  pointer-events: none;
}
```

```
@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}
```

```
.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}
```

```
.App-link {
  color: #61dafb;
}
```



```

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

```

App.jsx

```

import React from 'react';
import {Routes, Route, Navigate, Link, useNavigate} from 'react-router-dom';
import {Navbar, Container, Nav} from 'react-bootstrap';
import Login from './components/Login';
import Register from './components/Register';
import Main from './components/Main';
import AddCards from './components/AddCards';
import RepeatCards from './components/RepeatCards';
import EditPage from './components/EditPage';

```

// Обёртка для приватных маршрутов (требуется токен)

```

const PrivateRoute = ({children}) => {
  const token = localStorage.getItem('token');
  return token ? children : <Navigate to="/login"/>;
};

```

```

const PublicRoute = ({children}) => {
  const token = localStorage.getItem('token');
  return token ? <Navigate to="/" /> : children;
};

```

```
};
```

```
const App = () => {
```

```
  const navigate = useNavigate();
```

```
  const token = localStorage.getItem('token');
```

```
  const handleLogout = () => {
```

```
    localStorage.removeItem('token');
```

```
    navigate('/login');
```

```
  };
```

```
  return (
```

```
    <>
```

```
    <Navbar bg="primary" variant="dark" expand="lg">
```

```
      <Container>
```

```
        <Navbar.Brand as={Link} to="/">Flashcards App</Navbar.Brand>
```

```
        <Navbar.Toggle aria-controls="basic-navbar-nav"/>
```

```
        <Navbar.Collapse id="basic-navbar-nav">
```

```
          {token ? (
```

```
            <Nav className="ms-auto">
```

```
              <Nav.Link onClick={handleLogout}>Выйти</Nav.Link>
```

```
              <Nav.Link as={Link} to="/edit">Редактировать</Nav.Link>
```

```
            </Nav>
```

```
          ) : (
```

```
            <Nav className="ms-auto">
```

```
              <Nav.Link as={Link} to="/login">Вход</Nav.Link>
```

```
              <Nav.Link as={Link} to="/register">Регистрация</Nav.Link>
```

```
            </Nav>
```

```
          )}
```

```

        </Navbar.Collapse>
    </Container>
</Navbar>
<Routes>
    <Route
        path="/login"
        element={
            <Container className="mt-4">
                <PublicRoute>
                    <Login/>
                </PublicRoute>
            </Container>
        }
    />
    <Route
        path="/register"
        element={
            <Container className="mt-4">
                <PublicRoute>
                    <Register/>
                </PublicRoute>
            </Container>
        }
    />
    <Route
        path="/"
        element={
            <Container className="mt-4">
                <PrivateRoute>

```

```

        <Main/>
    </PrivateRoute>
</Container>
}
/>
<Route
    path="/collection/:collectionId"
    element={<AddCards/>}
/>
<Route
    path="/edit"
    element={
        <PrivateRoute>
            <EditPage />
        </PrivateRoute>
    }
/>
<Route
    path="/repeat/:collectionId"
    element={
        <PrivateRoute>
            <RepeatCards/>
        </PrivateRoute>
    }
/>
</Routes>

</>

);

```

```
};
```

```
export default App;
```

```
components\AddCards.jsx
```

```
// src/components/AddCards.jsx
```

```
import React, { useState, useMemo, useCallback, useRef } from 'react';
```

```
import { Container, Row, Col, Button, Spinner } from 'react-bootstrap';
```

```
import { useParams } from 'react-router-dom';
```

```
import { gql, useQuery, useMutation } from '@apollo/client';
```

```
import CodeMirror from '@uiw/react-codemirror';
```

```
import { markdown } from '@codemirror/lang-markdown';
```

```
import { githubLight } from '@uiw/codemirror-theme-github';
```

```
// Vim-режим
```

```
import { vim } from '@replit/codemirror-vim';
```

```
import { StateField } from '@codemirror/state';
```

```
import { lineNumbers } from '@codemirror/view';
```

```
// Markdown + плагины
```

```
import ReactMarkdown from 'react-markdown';
```

```
import remarkGfm from 'remark-gfm';
```

```
import remarkMath from 'remark-math';
```

```
import rehypeKatex from 'rehype-katex';
```

```
import rehypeHighlight from 'rehype-highlight';
```

```
import rehypeRaw from 'rehype-raw';
```

```

import { FaPencilAlt, FaEye } from 'react-icons/fa';

import 'katex/dist/katex.min.css';
import 'highlight.js/styles/github.css';
import 'github-markdown-css/github-markdown.css';

import { processMarkedText } from '../utils/highlightLogic';

/* ----- GraphQL ----- */

const GET_COLLECTION = gql`
  query GetCollection($id: ID!) {
    collection(id: $id) {
      id
      name
      cards { id } # чтобы refetch обновил количество
    }
  }
`;

const SAVE_CARD = gql`
  mutation SaveCard($card: CardInp!) {
    saveCard(card: $card) {
      id
      text
      createdAt
    }
  }
`;

```

```
/* ----- Вспомогательные плагины CodeMirror ----- */
```

```
const relativeLineNumbers = lineNumbers({  
  formatNumber: (n, state) => {  
    const cur = state.doc.lineAt(state.selection.main.head).number;  
    return n === cur ? String(n) : String(Math.abs(n - cur));  
  },  
});
```

```
const vimState = StateField.define({  
  create: () => ({}),  
  update(v) { return v; }  
});
```

```
/* ----- КОМПОНЕНТ ----- */
```

```
const AddCards = () => {  
  const { collectionId } = useParams();  
  const { loading, error, data } = useQuery(GET_COLLECTION, {  
    variables: { id: collectionId },  
  });
```

```
  const [cardText, setCardText] = useState("");  
  const [vimMode, setVimMode] = useState(false);  
  const [annotationMode, setAnnotation] = useState(false);  
  const [clozeMode, setClozeMode] = useState(false);  
  const hiddenContentsRef = useRef([]);
```

```

const [saveCard, { loading: saving, error: saveError }] =
useMutation(SAVE_CARD, {
  refetchQueries: [{ query: GET_COLLECTION, variables: { id: collectionId }
}],
  onCompleted: () => {
    setCardText("");
    document.querySelector('.cm-content')?.focus();
  }
});

```

```

/* --- CodeMirror плагины --- */

```

```

const baseExtensions = useMemo(() => [markdown(), relativeLineNumbers], []);
const editorExtensions = useMemo(
  () => (vimMode ? [...baseExtensions, vimState, vim()] : baseExtensions),
  [vimMode, baseExtensions]
);

```

```

/* --- Обработчик выделения в предпросмотре --- */

```

```

const handlePreviewMouseUp = useCallback(() => {
  if (!annotationMode) return;
  const sel = window.getSelection();
  const text = sel?.toString();
  if (!text) return;

  const idx = cardText.indexOf(text);
  if (idx === -1) return;

  setCardText(
    cardText.slice(0, idx) + '== ' + text + '==' + cardText.slice(idx + text.length)
  );

```



```
);  
    sel?.removeAllRanges();  
}, [annotationMode, cardText]);  
  
/* --- Cloze-обработка текста --- */  
hiddenContentsRef.current = [];  
const processedText = processMarkedText(cardText, clozeMode,  
hiddenContentsRef.current);  
  
/* --- Сохранение карточки --- */  
const handleAddCard = async () => {  
    if (!cardText.trim()) return;    // пустые строки не шлём
```