ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

*Факультет информатики, математики и компьютерных наук*

Терехина Полина Сергеевна

**разработка приложения для автоматического создания конспектов лекций**

Курсовая работа

образовательная программа

«Компьютерные науки и технологии»

|  |  |
| --- | --- |
| Нижний Новгород, 2025 | Руководитель  к-т/д-р \_\_\_\_\_ наук, доцент/проф.  М.М. Железин  И.О. Фамилия |

**Оглавление**

[**Пояснительная записка** 3](#_Toc198489107)

[**Техническое задание** 9](#_Toc198489108)

[**Анализ проблематики** 10](#_Toc198489109)

[**Анализ существующих решений, технологий и аналогов** 11](#_Toc198489110)

[**Полученные результаты, методика испытаний и итоги анализа** 12](#_Toc198489111)

[**Список использованных источников** 13](#_Toc198489112)

[**Приложение 1: Название приложения** 14](#_Toc198489113)

**Пояснительная записка**

#### **Актуальность проекта**

В образовательном процессе студенты сталкиваются с необходимостью обработки больших объемов информации. Согласно исследованиям, около 78% учащихся испытывают трудности с эффективным конспектированием лекций, что негативно сказывается успеваемости и усвоении материала.

Разрабатываемое приложение призвано решить следующие проблемы:

* **Высокая когнитивная нагрузка** при одновременном восприятии и записи лекций.
* **Потеря ключевой информации** из-за ручного конспектирования.
* **Неэффективная структура** самостоятельно созданных конспектов.

Интеграция **GigaChat API** позволяет автоматизировать процесс создания структурированных конспектов с выделением основных тезисов, что особенно актуально для студентов технических специальностей, где важна точность передачи информации.

**Цель работы:** разработка веб-приложения для автоматического создания конспектов лекций с использованием современных технологий обработки естественного языка (NLP).

**Задачи:**

1. Реализация **backend-части** на Adonis для обработки запросов и интеграции с GigaChat API.
2. Разработка **frontend-интерфейса** на Next.js и React с использованием Ant Design для удобного взаимодействия пользователя с системой.
3. Настройка взаимодействия между frontend и backend через REST API.
4. Обеспечение функционала для загрузки текстовых/аудиофайлов, генерации и редактирования конспектов.

**Применяемые методы и подходы:**

* **Методология RESTful API** для взаимодействия между клиентом и сервером.
* **Промпт-инжиниринг** GigaChat для улучшения качества выходных данных.
* **Компонентный подход** в React для построения интерфейса.

**Ограничения:**

* Зависимость от GigaChat API (ограничения на количество запросов, скорость обработки).
* Поддержка только текстовых форматов на текущем этапе (DOCX, PDF, TXT).

**1.1. Алгоритм работы системы**  
Приложение реализует следующий workflow:

1. **Прием входных данных:**
   * Пользователь загружает файл (TXT, DOCX, PDF) или вводит текст вручную через интерфейс.
   * Frontend (Next.js) отправляет данные на backend через REST API.
2. **Обработка на backend (Adonis):**
   * Валидация формата и размера файла.
   * Извлечение текста (для DOCX/PDF используется библиотека pdf-parse/mammoth).
   * Препроцессинг: удаление стоп-слов, нормализация текста.
3. **Интеграция с GigaChat API:**

- См. Приложение 1

* Отправка промпта через официальный SDK GigaChat.
* Получение и парсинг ответа (JSON).

1. **Возврат результата:**
   * Форматирование конспекта (добавление HTML-разметки).
   * Кэширование в MongoDB (для повторных запросов).

**Обоснование выбора алгоритма:**

* **REST API** выбран для простоты интеграции между Next.js и Adonis.
* **Промпт-инжиниринг** оптимизирован под учебные материалы (академический стиль).
* **Кэширование** снижает нагрузку на GigaChat API.

**1.2. Взаимодействие подпрограмм**

| **Модуль** | **Взаимодействие** | **Протокол** |
| --- | --- | --- |
| Frontend | Отправка запросов → Backend | HTTP/HTTPS |
| Backend | Запросы → GigaChat API | gRPC (через SDK) |
| Database | Хранение истории запросов | MongoDB Query |

#### **2. Выбор технических и программных средств**

**2.1. Программный стек**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Компонент** | **Технология** | **Обоснование выбора** | | --- | --- | --- | | **Frontend** | Next.js 14 + React 18 | SSR для SEO, оптимизация загрузки | | **UI-библиотека** | Ant Design 5.x | Готовые компоненты для админ-панелей | | **Backend** | Adonis 6.x | MVC-архитектура, встроенная ORM | | **База данных** | MongoDB 7.0 | Гибкость для хранения неструктурированных данных | | **NLP-сервис** | GigaChat API | Лучшая поддержка русского языка vs GPT | |

**2.2. Аппаратные требования**

| **Ресурс** | **Минимальные требования** | **Рекомендуемые** |
| --- | --- | --- |
| CPU | 2 ядра (x86-64) | 4 ядра |
| RAM | 2 ГБ | 8 ГБ |
| Диск | 10 ГБ (SSD) | 50 ГБ |

**Расчет нагрузки (на 100 пользователей/час):**

* **Backend:** ~500 MB RAM (Adonis + Node.js)
* **GigaChat API:** ~3 сек/запрос (лимит: 100 запросов/мин на бесплатном тарифе)
* **Сеть:** 1 Мбит/с входящего трафика

**2.3. Сравнение с альтернативами**

| **Критерий** | **GigaChat** | **GPT-4** | **YandexGPT** |
| --- | --- | --- | --- |
| Стоимость | Бесплатно (до лимита) | $0.06/1K токенов | $0.05/1K токенов |
| RU-поддержка | 10/10 | 8/10 | 9/10 |
| Скорость | Средняя | Высокая | Низкая |

**Вывод:** GigaChat выбран как оптимальное решение для учебных задач на русском языке.

**Назначение:**

* Автоматизация конспектирования для студентов и преподавателей.
* Повышение эффективности работы с учебными материалами.

**Область применения:**

* Высшие учебные заведения.
* Онлайн-курсы и образовательные платформы.
* Личное использование для структурирования заметок.

**Техническое задание**

В данном разделе приводятся требования к реализуемой подпрограмме, а также ее связь с другими частями общего проекта

**Анализ проблематики**

В данном разделе приводится детальный анализ проблематики, на решение которой направлена реализация подпрограммы, в т.ч. результаты анализа литературы, результаты анализа бизнес-процессов предприятия с использованием общепринятых инструментов и средств и т.п.

**Анализ существующих решений, технологий и аналогов**

В данном разделе необходимо провести анализ по существующим решениям и подходам (фундаментальным/научным и прикладным), используемым для решения аналогичных и/или близких задач, а также технологического стека, который может быть использован для решения поставленных задач (с обоснованием итогового выбора).

В заключение раздела приводится итоговый стек подходов и технологий, используемых в дальнейшем в проекте и обоснование их выбора.

**Полученные результаты, методика испытаний и итоги анализа**

В данном разделе необходимо привести результаты, которые были получены в рамках проекта (продемонстрировать полученные результаты работы реализованной подпрограммы, ее поведение в различных ситуациях).

При этом обязательно приводятся расчеты метрик качества в соответствии с существующими стандартами и их интерпретация.

**Список использованных источников**

Приводится список источников (в соответствии с ГОСТ), использованных в ходе работы над проектом, а также в документе.

**Приложение 1: Название приложения**

В Приложениях можно разместить большие (более 1 страницы) таблицы, схемы, плановую, учетную, отчетную и др. документации, а также иные пояснительные материалы, позволяющие оценить результаты проекта.  
  
**Интеграция GigaChat в Adonis:**

*// app/Controllers/Http/AiController.js*

**const** GigaChat = require('gigachat-api');

**class** AiController {

async generateSummary({ request, response }) {

**const** { text } = request.all();

**const** client = **new** GigaChat(process.env.GIGACHAT\_API\_KEY);

**try** {

**const** summary = await client.createCompletion({

prompt: `Создай конспект: ${text}`,

maxTokens: 1000

});

**return** response.json(summary);

} **catch** (error) {

**return** response.status(500).json({ error });

}

}

}