Оглавление

[Введение 8](#_Toc102319817)

[Определения, обобщения и сокращения 9](#_Toc102319818)

[Описание предметной области 10](#_Toc102319819)

[Описание прикладного процесса 10](#_Toc102319820)

[Формирование требований 10](#_Toc102319821)

[Проектирование 12](#_Toc102319822)

[Используемый стек технологий 12](#_Toc102319823)

[Системная архитектура 12](#_Toc102319824)

[Архитектура данных 13](#_Toc102319825)

[Программная архитектура 15](#_Toc102319826)

[Разработка 17](#_Toc102319827)

[Реализация серверного API 17](#_Toc102319828)

[Реализация пользовательского интерфейса 18](#_Toc102319829)

[Заключение 24](#_Toc102319830)

[Список использованной литературы 25](#_Toc102319831)

# Введение

Объектом разработки является веб-приложение для генерации текстов в стиле рэп-исполнителя, реализованное на языке программирования Python с использованием фреймворка FastAPI — для серверной части приложения, и JavaScript — для клиентской части, без применения сторонних фреймворков. Выбранный фреймворк отличается высокой производительностью, лаконичностью и активной поддержкой сообщества. Целью работы является разработка веб-приложения и пользовательского интерфейса, анализ требования и моделирование процессов, средств автоматизации и архитектуры информационной системы.

В ходе работы были получены следующие результаты:

* Серверная часть системы, принимающая запросы.
* Клиентская часть системы, предоставляющая интерфейс пользователя.
* База данных для хранения информации о пользователях, файлах, а также служебной информации внутри системы.

# Определения, обобщения и сокращения

*Браузер* – программное обеспечение для просмотра веб-страниц, обработки HTML, CSS и JavaScript. Используется конечным пользователем для взаимодействия с веб-приложением

*Фреймворк* – набор инструментов и библиотек, предоставляющий архитектурную основу для разработки программного обеспечения.

*Роль «администратор» внутри приложения* – просмотр всех пользователей.

*Генеративная модель* – языковая модель, способная создавать текст на основе заданного входа.

*API (Application Programming Interface)* – программный интерфейс приложения. Позволяет внешним приложениям взаимодействовать с функциональностью сервера.

*FAISS (Facebook AI Similarity Search)* — библиотека для эффективного поиска похожих векторов в больших объёмах данных. Используется для поиска похожих отрывков текста на основе эмбеддингов.

*RAG (Retrieval-Augmented Generation)* — подход генерации текста, при котором модель получает внешние текстовые фрагменты (документы) из базы знаний перед генерацией ответа, что повышает точность и релевантность результата.

# Описание предметной области

## Описание прикладного процесса

Разрабатываемое веб-приложение представляет собой интерактивную систему генерации текстов в стиле конкретного исполнителя, построенную на основе языковых моделей и RAG-системы. Приложение позволяет пользователю ввести начало текста, на основе которого система сгенерирует продолжение, стилизованное под выбранного исполнителя.

В отличие от классических систем управления контентом, где основной задачей является просмотр, редактирование и публикация информации на сайте, данное приложение ориентировано на генерацию уникального контента в диалоге с пользователем.

Формирование требований

В ходе анализа прикладного процесса был получен следующий список функциональных требований:

Модуль аутентификации пользователей должен иметь:

* Возможность регистрации новых пользователей с уникальным именем
* Возможность входа в систему с именем пользователя и пароля
* Возможность выхода из системы
* Система обновления токенов JWT

Модуль генерации текста должен иметь:

* Возможность генерации текста на основе пользовательского запроса
* Использование векторной базы данных для RAG-системы
* Интеграция с API OpenAI

Модуль работы с избранным должен иметь:

* Возможность сохранения сгенерированных текстов в раздел избранного
* Просмотр всех сохраненных текстов пользователя

Модуль веб-интерфейса должен иметь:

* Интуитивно понятный пользовательский интерфейс
* Отображение статуса процесса генерации в реальном времени
* Возможность переключения между страницами генератора и избранного

Нефункциональные требования

Разрабатываемая система не является публичной, поэтому основным требованием выступает её закрытость, путем обязательного прохождения процесса авторизации и аутентификации.

Так же должна быть возможность работы с приложением напрямую, через общие программные интерфейсы, описанные по спецификации OpenAPI версии не ниже 3.0

# Проектирование

## Используемый стек технологий

Решение создания именно веб-приложения обусловлено тем, что необходимо было обеспечить доступ к системе с любого устройства, в любое время. Веб-приложение решает этот вопрос, а также снимает вопрос обновлений на стороне клиента.

Проект использует стек стандартных технологий, характерный для большинства веб-приложений: HTML, CSS, Javascript.

В качестве веб-сервера используется FastAPI.

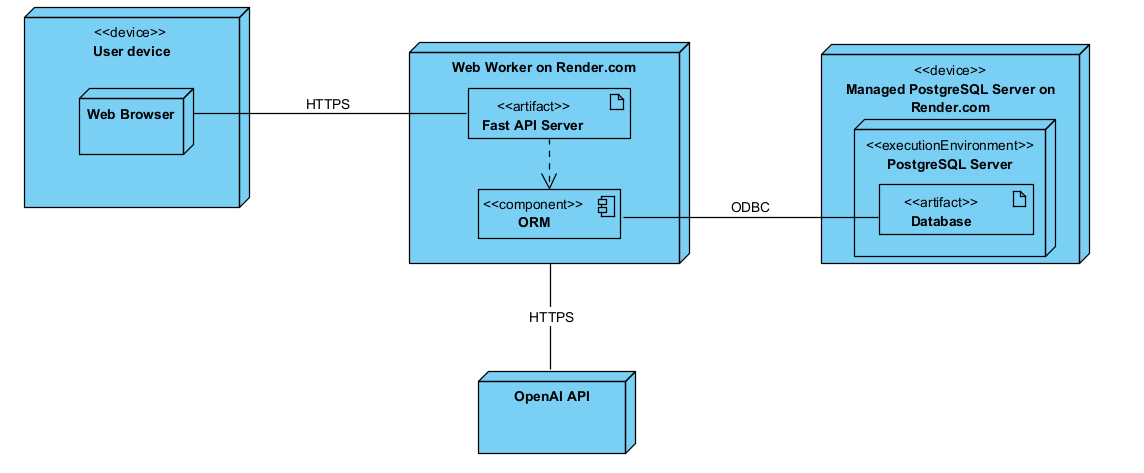
В качестве базы данных используется PostgreSQL, ввиду того, что данная СУБД является самой стабильной в данной связке. Для работы с данными была выбрана ORM SQLAlchemy.

В проекте используется свободная распределённая система управления версиями Git, хранилищем исходных кодов является крупнейший веб-сервис GitHub, а в качестве хостинга используется облачный сервис Render.com.

## Системная архитектура

Архитектура системы основана на клиент-серверной модели. Клиентская часть представлена веб-браузером, который взаимодействует с сервером через протокол HTTPS. Сервер реализован на FastAPI и обрабатывает запросы пользователей, а также осуществляет аутентификацию и авторизацию.

Сервер взаимодействует с PostgreSQL через ODBC протокол для хранения информации о пользователях и их избранных текстах. Для генерации текстов используется интеграция с OpenAI API по протоколу HTTPS.

  
Рисунок 1. Системная архитектура приложения.

## Архитектура данных

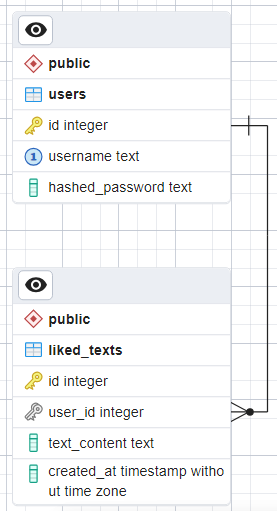


Рисунок 2. Схема таблиц базы данных

Users – Предоставляет информацию о пользователе

Liked\_texts – Предоставляет информацию о понравившихся пользователю текстах

## Программная архитектура

Таблица 1. Отношения модулей и классов

|  |  |  |
| --- | --- | --- |
| **Название модуля** | **Функциональный компонент** | **Назначение компонента** |
| Модуль аутентификации | register, login, logout | Регистрация, вход и выход пользователя в системе |
| Модуль генерации текста | generate | Генерация текста песни на основе пользовательского запроса |
| Модуль генерации текста | stream\_generation | Потоковая генерация текста с отображением прогресса |
| Модуль избранного | like\_text | Сохранение текста в избранное пользователя |
| Модуль веб-интерфейса | serve\_frontend, generator\_page,  favorites\_page | Отображение главной страницы приложения, страницы генератора и избранных текстов. |

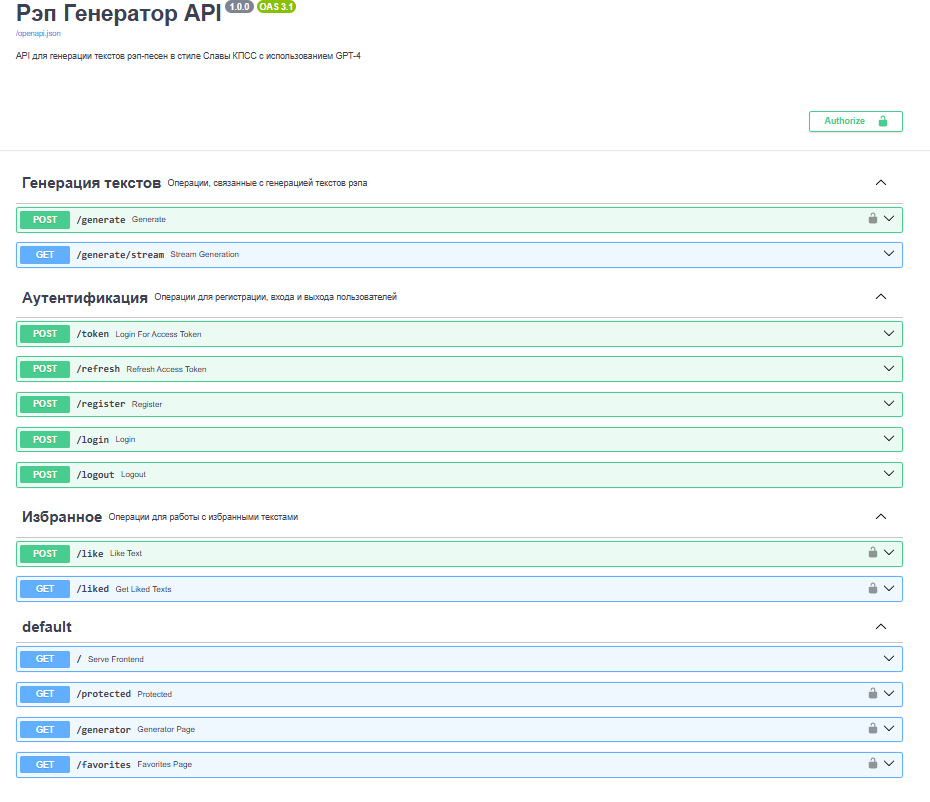
Таблица 2. Описание классов

|  |  |
| --- | --- |
| **Название компонента** | **Описание компонента** |
| register | Эндпоинт для регистрации новых пользователей с проверкой на уникальность имени и хешированием паролей. |
| login | Эндпоинт для аутентификации пользователей |
| generate | Защищенный эндпоинт для генерации текста песни с помощью RAG и GPT-4. Требует авторизации и визуализируется на странице генератора. |
| stream\_generation | Эндпоинт для пошаговой генерации текста с отображением процесса в реальном времени с помощью SSE. |
| like\_text | Эндпоинт для сохранения понравившихся текстов. Отображается в виде кнопки "Добавить в избранное" на странице генератора. |

# Разработка

## Реализация серверного API

В качестве описания программного интерфейса был выбран инструмент, поддерживающий стандарт OAS 3.0 – Swagger. Далее представлена полученная документация API полученная автоматически по директивам, указанным в декораторах различных методов и структурах данных внутри разрабатываемой информационной системы.

 Рисунок 3. Программный интерфейс серверного API.

## Реализация пользовательского интерфейса

Аутентификации пользователей:

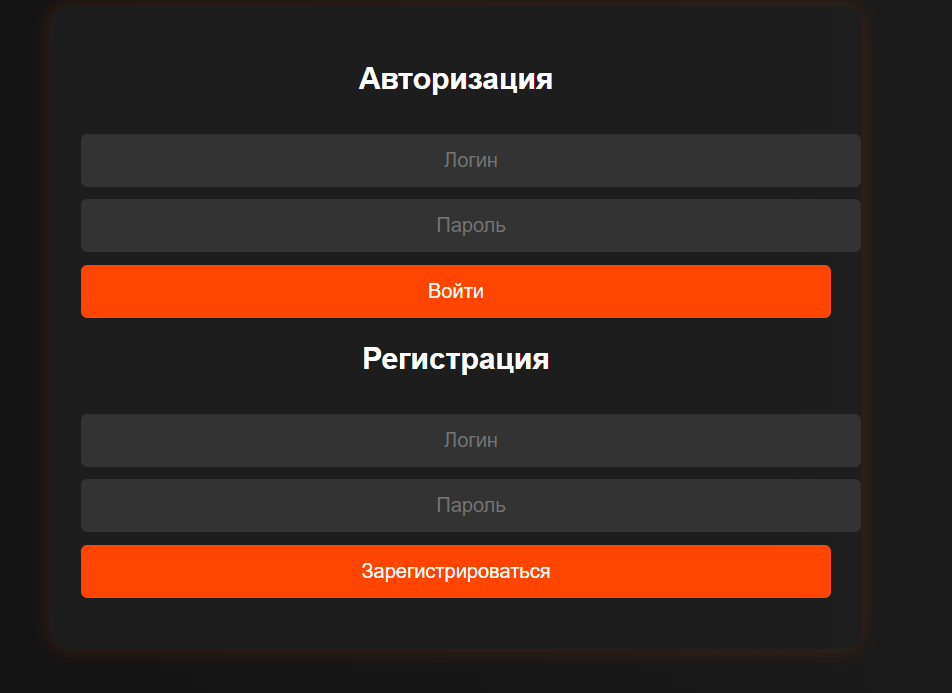


Рисунок 4

Пользователь проходит регистрацию и далее может авторизоваться:

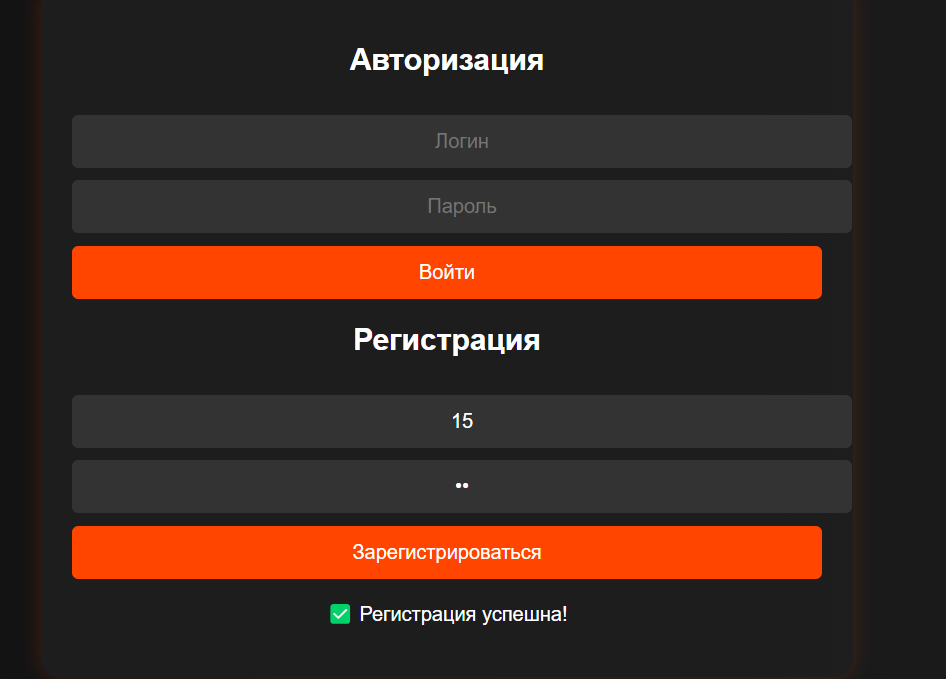


Рисунок 5

Выдается JWT токен

После авторизации пользователь попадает на страницу генерации и может ввести название песни или посмотреть ранее сохраненные тексты:

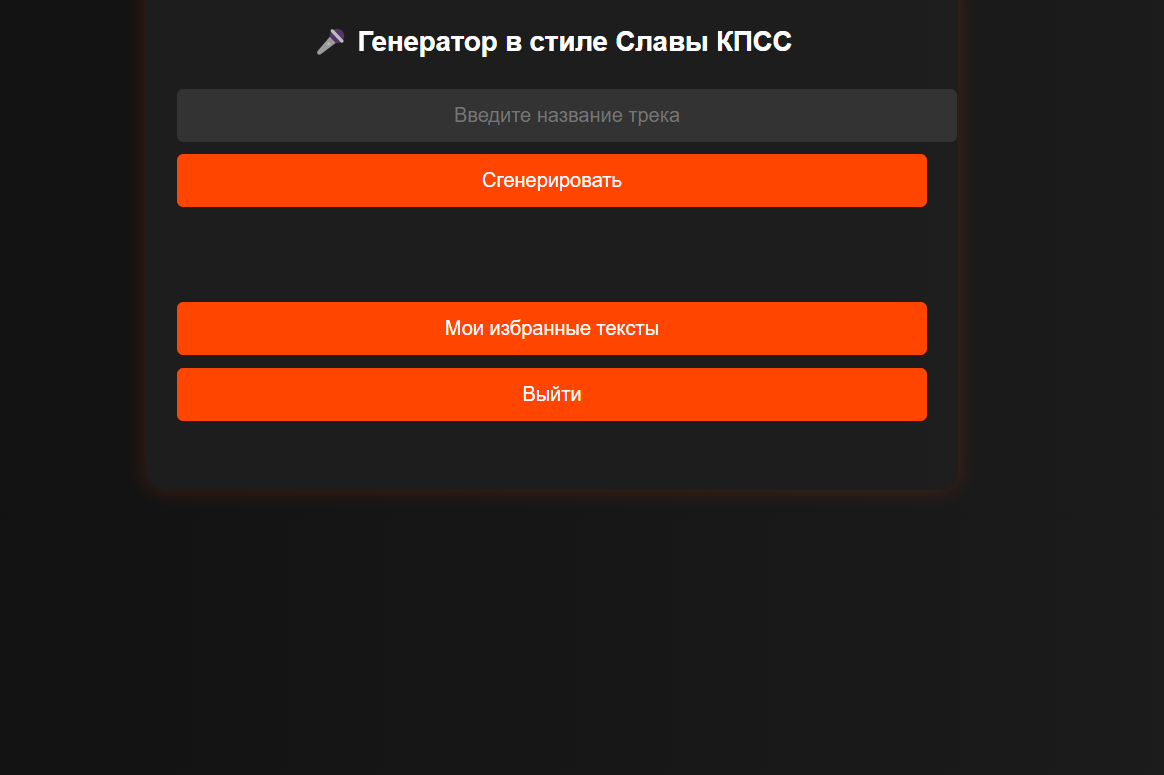


Рисунок 6

После того, как пользователь ввел строчку за который генератор должен продолжить текст, на сайте изображаются шаги генерации.

В этот момент через FAISS в векторной базе данных ищутся похожие текста из 1500 спаршенных песен исполнителя, на их основе генерируется текст, который видит пользователь

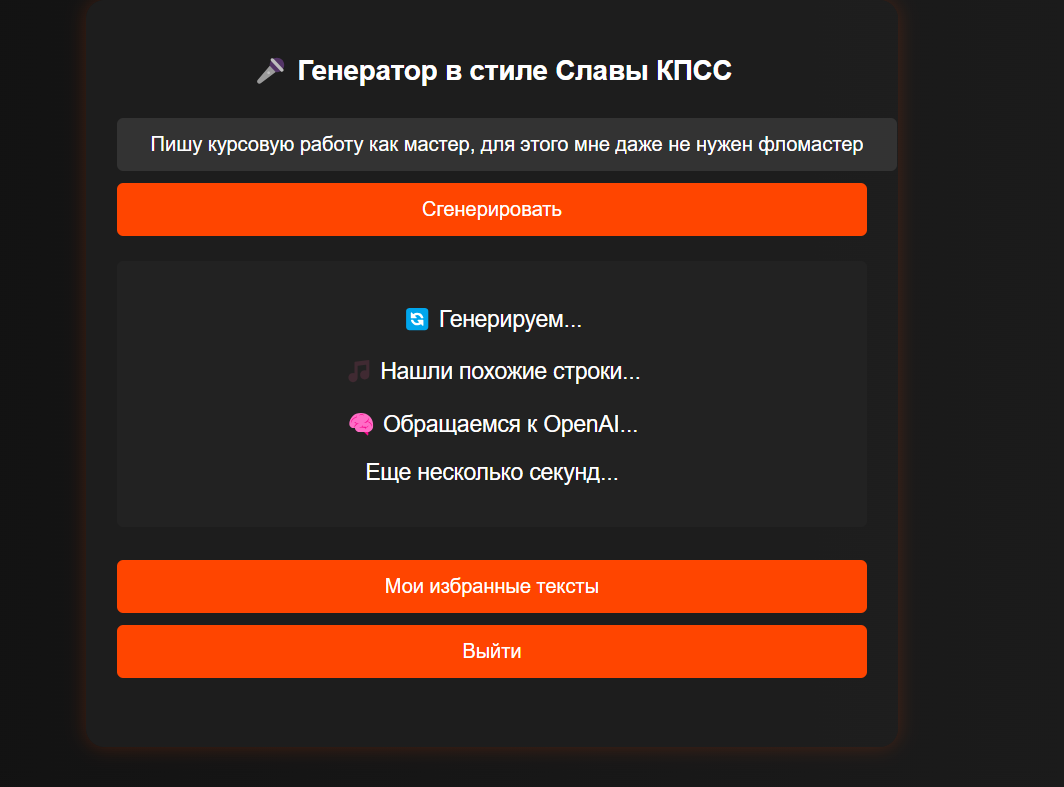


Рисунок 7

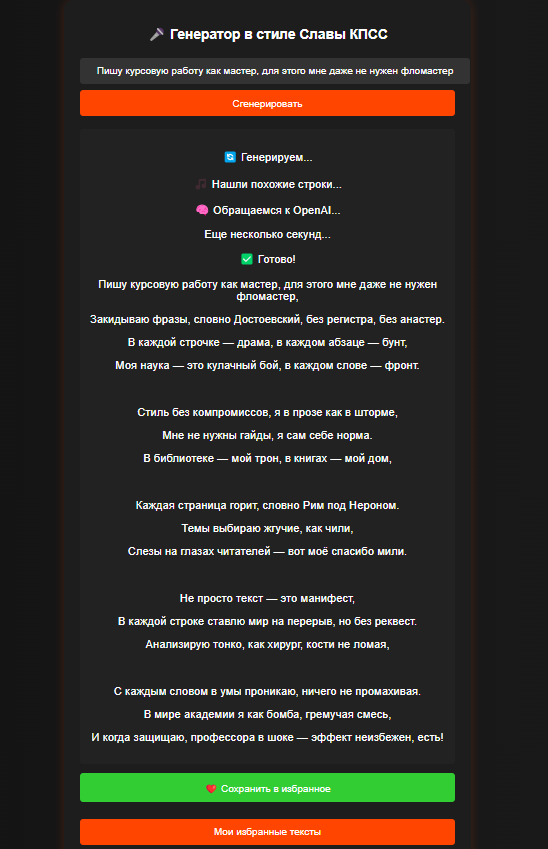


Рисунок 8

Далее пользователь видит текст и может нажать на кнопку «Сохранить в избранное»

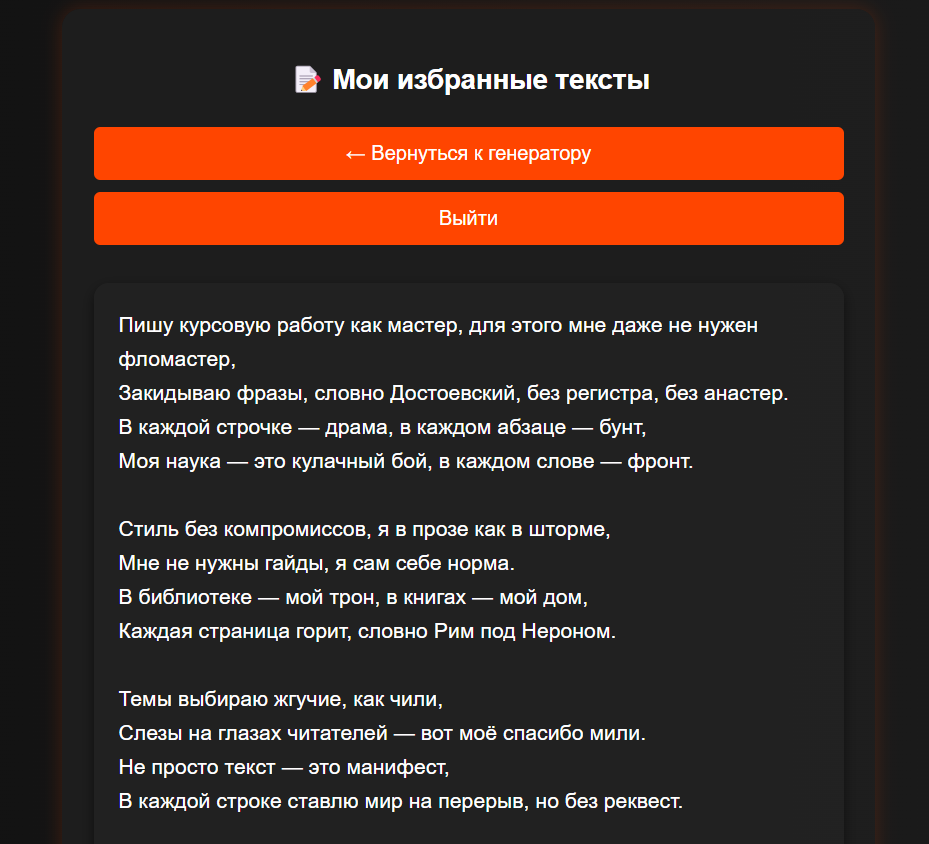


Рисунок 9

# Заключение

В ходе выполнения курсовой работы был проведён анализ работы классических систем управления сайтами, а также существующих моделей генерации текстов, исходя из которых были выявлены и сформированы требования к разрабатываемому веб-приложению.

Исходя из выбранной архитектуры и наложенных ограничений были сформированы требования к используемым технологиям внутри модулей. Была спроектирована архитектура данных, программная и системная архитектура в виде набора диаграмм в нотации UML.

Опираясь на выше изложенные требования и стек технологий было разработано веб-приложение и пользовательский интерфейс в рамках дисциплины «Web-программирование».

Таким образом, все поставленные ранее цели были выполнены.

Разработанное приложение является результатом данной курсовой работы.

# Список использованной литературы

1. Мартин Фаулер - Архитектура корпоративных программных приложений. Издательский дом "Вильяме'. 2006 г.

2. Флэнаган, Дэвид. JavaScript. Полное руководство, 7-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2021. — 720 с .

3. Янг А., Мек Б., Кантелон М. Node.js в действии. 2-е изд. — СПб.: Питер, 2018. — 432 с.

4. Браун И.Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. 2-е издание. — СПб.: Питер, 2021. — 336 с.

5. Современный учебник JavaScript [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/. – Дата доступа: 04.05.2021.