СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Наименование программы

Название на русском: «Русский учебный корпус».
Название на английском: «Russian learner corpus».
Название для пользователя: «RLC».

1.2. Документы, на основании которых ведется разработка

Основанием для разработки является учебный план подготовки бакалавров по направлению 09.03.04 "Программная инженерия" и утвержденная академическим руководителем тема курсового проекта.

2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

2.1. Назначение программы

2.1.1. Функциональное назначение

Данный сайт предоставляет пользователю инструментарий для работы с текстами, их разметкой и исправлением ошибок. Помимо прочего пользователю доступен просмотр уже аннотированных текстов, что может применяться в исследовательской деятельности.

2.1.2. Эксплуатационное назначение

Сайт можно использовать для помощи пользователям, которые изучают русский как иностранный язык. Данный функционал осуществляется при помощи предварительно загруженных текстов с подробным объяснением правил русского языка.

2.2. Краткая характеристика области применения

По словам самого заказчика: “В Русском учебном корпусе содержатся образцы устной и письменной речи двух категорий нестандартных говорящих на русском языке: изучающих русский язык как иностранный и так называемых эритажных говорящих. Для первой категории русский язык не является родным, представители же второй категории начали усваивать его в детстве как первый язык, но по разным причинам (в основном, это эмиграция) в качестве основного языка общения используют другой язык.” Данное собрание текстов можно использовать в таких сферах, как исследование и образования. Поскольку на сайте есть возможность выгрузки текстов и их аннотаций, то эти данные можно использовать в машинном обучении, что является привлекательным в исследовательской деятельности. С точки зрения обучения русского языка как иностранного, RLC предлагает пользователю широкую базу размеченных текстов с тегами ошибок, чтобы лично ознакомиться с русским языком на практике и предотвратить частые ошибки в его применении.

Однако во время работы с сайтом заказчик столкнулся со множеством критических ошибок, которые необходимо решить в срочном порядке. Помимо всего так же был передан список возможных улучшений, которые хотел бы по возможности увидеть заказчик в новой версии сайта. В это и заключается цель нашего проекта.

3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

3.1. Постановка задачи на разработку программы

Проект является групповым, поэтому наша команда делегировала обязанности реализации списка проблем и улучшений от заказчика. Под мою ответственность перешла модернизация вкладки “Новости” и “Помощь”, а также помощь в проектировании базы данных и вспомогательные функции для работы с аннотациями текстов. Проблемы, которые будут решены с помощью моих введений – это нерабочая вкладка новостей, которая не помогала пользователю, поскольку инструкции, описанные в ней, не работали, ошибка аннотирования текстов, когда из-за неправильного выделения нарушалась работа аннотирования в целом (Например, если выделить аннотацию с пробелом, то ломалась индексация и вся разметка текста непредсказуемо смещалась), а также неудобство в ручном написании информации об авторах текстов, что еще сильнее сказывалось, если приходилось загружать множество текстов от одного и того же автора подряд. С учетом сказанного были выделены следующие задачи, которые позже будут рассмотрены подробней:

1) Написание моделей для отображения и хранения данных о новостях, токенах и аннотациях
2) Проектирование архитектуры базы данных проекта о новостях, токенах и аннотациях
3) Написание и добавление пользовательского руководства по использованию сайта на отдельной странице
4) Написание функционала для добавления, редактирования и удаления новостных статей
5) Автоматическое добавление и сохранение авторов текста
6) Контекстный поиск по токенам
7) Предобработка добавленных текстов для дальнейшего хранения в базе данных

3.2. Описание алгоритма и функционирования программы

3.2.1. Отображение и хранение в базе данных информации о новостях, токенах и аннотациях

В Django присутствуют встроенные модели, однако помимо них были реализованы:

1) Annotation – модель для хранения информации об аннотации текстов (ссылка на текст и на предложение, пользователь, добавивший аннотацию, сама аннотация в формате JSON)
2) Token – модель для хранения токена (ссылка на текст и предложение, позиция в предложении, часть речи, лемма и грамматические характеристики)
3) Article – модель, отвечающая за хранение информации о новостных статьях

3.2.2. Архитектура базы данных проекта о новостях, токенах и аннотациях

Для работы с аннотированием и новостями в базе данных были разработаны некоторые модели, описанные в пункте 3.2.1, однако также их следовало правильно организовать, поэтому была разработана следующая архитектура, которая изображена на рисунке 1.

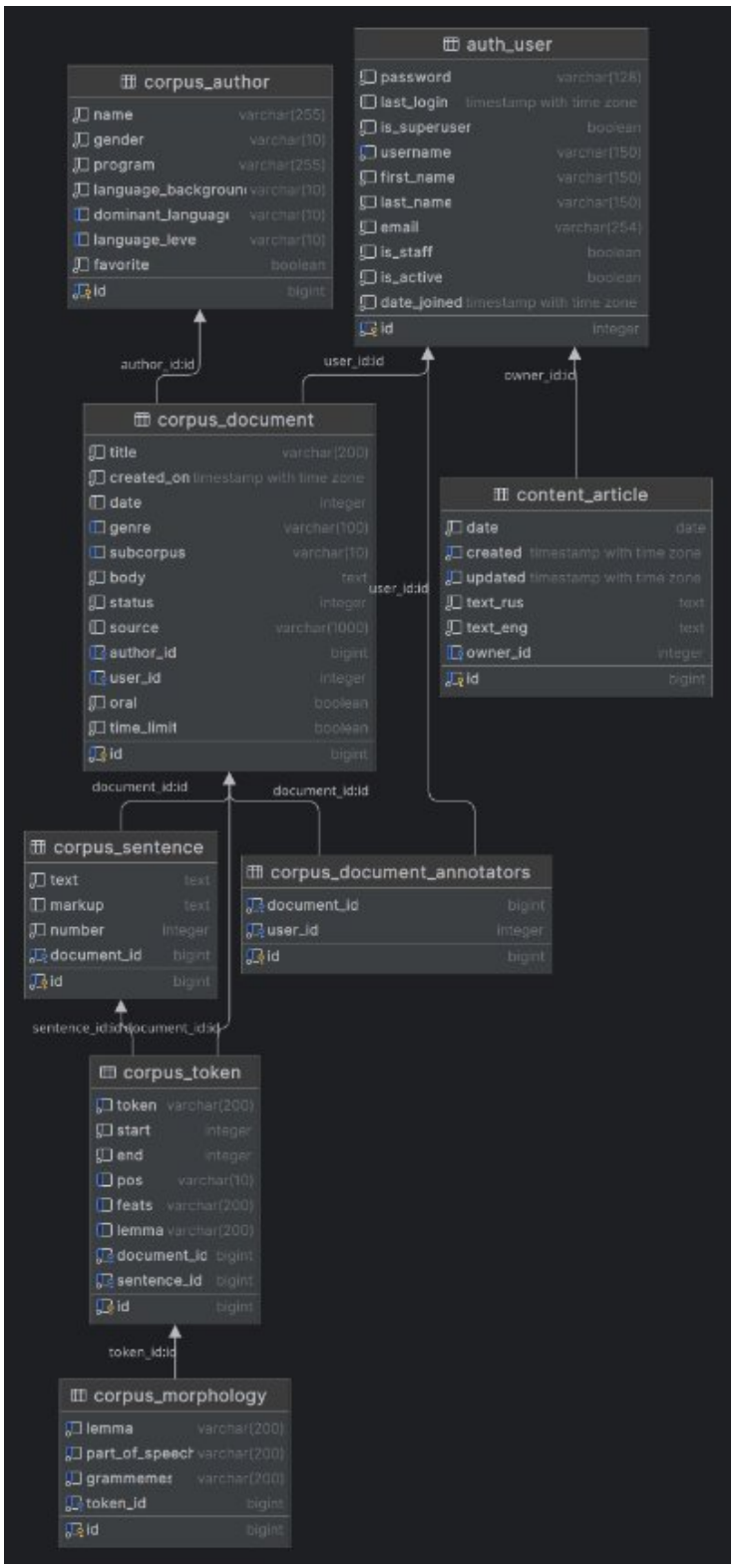

Рисунок 1. Архитектура базы данных проекта о новостях, токенах в аннотации

1) content_article – новости на сайте
2) corpus_document_annotators – информация о всех аннотаторах текущего документа
3) corpus_token – информация о токенах в документах
4) corpus_annotation – информация о всех аннотациях

5) corpus_morphology – информация о всех морфологических разборах

3.2.3. Вкладки “Помощь” и “Новости”

При детальном рассмотрении кода изначального сайта наша команда обнаружила множество серьезных ошибок, которые серьезно мешали процессу решения существующих проблем и дальнейшей модернизацией сайта в целом. Ярыми препятствиями стали сильная связность, запутанность кода, а также сомнительные реализации некоторых частей сайта (например множество SQL запросов, которые сложно тестировать и считывать, впоследствии они были переведены в ORM, подробнее об этом можно увидеть у моего коллеги по проекту Максима Пересторонина в его пояснительной записке). Поэтому было принято решение по написанию сайта с нуля. Так что все вкладки пришлось заново разрабатывать, мне же достались вкладки помощи и новостей.

3.2.3.1. Помощь пользователю

Вкладка “Помощь” – это пользовательский мануал, который инструктирует в корректном использовании RLC. В нем содержится следующая информация, доступ к которой может быстро осуществляться с помощью ссылок в левом части с соответствующими названиями статей:

1) “Вкратце о корпусе” – содержит информацию о предназначении сайта в целом, о его возможностях, партнерах и информации, содержащейся на нем.
2) “Авторизация/регистрация” – помогает пользователю разобраться в преимуществах зарегистрированного пользователя над незарегистрированным, в способе регистрации и что делать в ситуациях, когда забыл пароль.
3) “Регистрация ошибок” – разъясняет пользователю классификацию ошибок, с которой ему придется столкнуться в процессе работы с сайтом. На странице будет представлена удобная таблица с категориями ошибок, их названиями и их обозначениями.
4) “Как размечать” - содержит инструкции по правильному способу разметки текстов, если разметчик столкнётся с какими-то либо трудностями.

3.2.3.2. Управление новостными статьями

Каждому пользователю будет предоставлена возможность во вкладке “Новости” просматривать новостные статьи. Однако не зарегистрированным пользователям не будут доступны редактирование, добавление и удаление новостных статей. Интерфейс добавления/изменения новостных статей можно подробно рассмотреть на рисунке 2:

1) Два поля для написания текста статьи на двух языках - русский и английский, поскольку на сайте присутствует возможность по нажатии кнопки перевести страницу на другой язык.
2) Два поля для написания названия статьи на двух языках – русский и английский, поскольку название тоже переводимо.
3) Поле для ввода номера статьи. Номер отвечает за порядок вывода статей на сайте, от меньшего к большему.

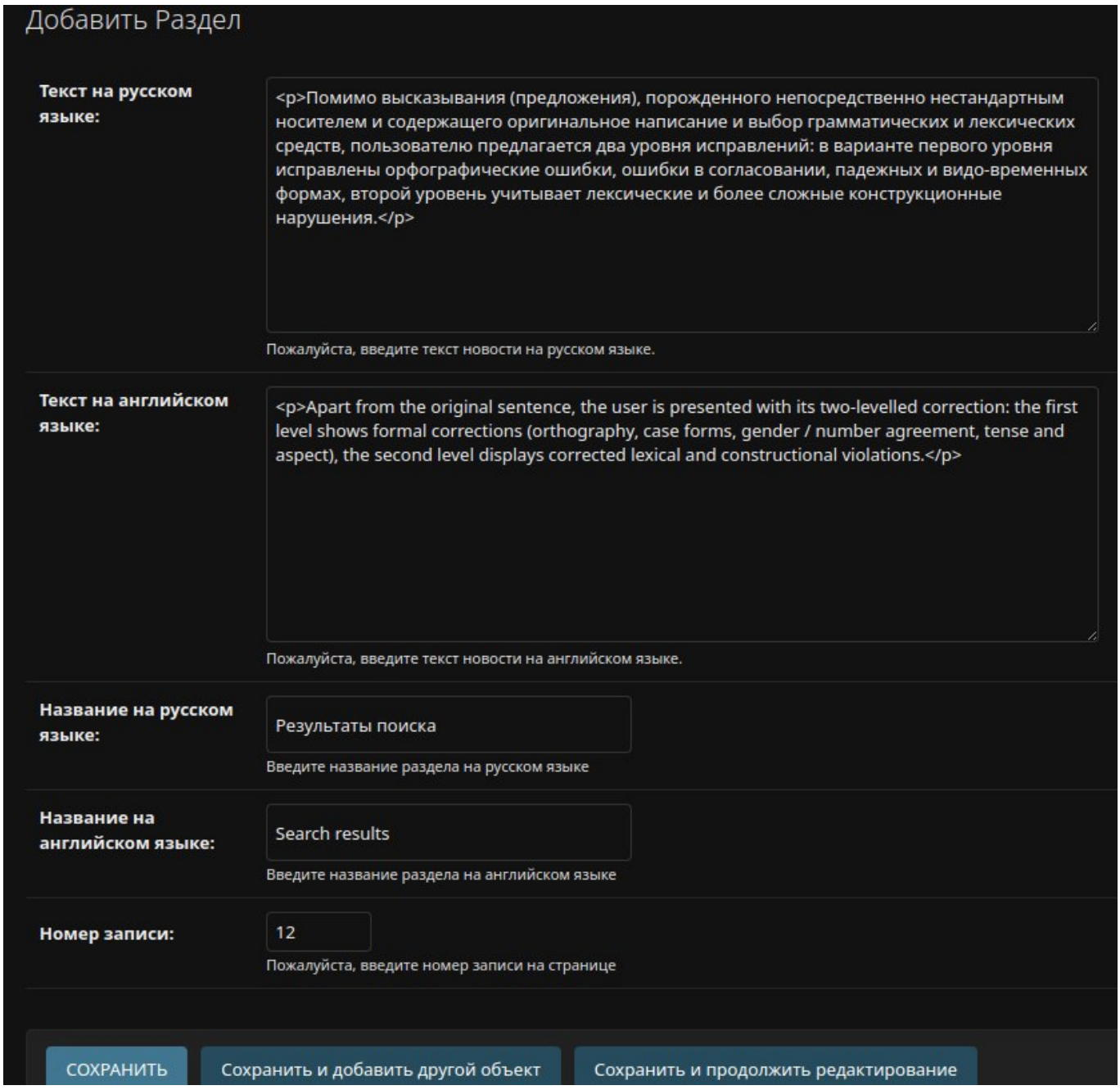

Рисунок 2. Интерфейс добавления/изменения новостной статьи

3.2.4. Автозаполнение информации об авторах текстов

Во время встречи с заказчиком было выделено такое неудобство, как ручное заполнение данных автора текста и нас попросили разрешить эту проблему. Тогда была реализована отдельная таблица базы данных авторов текста. Мне же было необходимо реализовать функционал упрощенного заполнения данных автора. Теперь при добавлении нового автора текста, его можно сохранить в базе данных, а данные всех сохраненных автором можно автоматически заполнить, если в поисковую строку указать его имя. Возможность добавления нового автора и получения данных из сохраненных авторов можно рассмотреть на рисунках 3 и 4 соответственно.

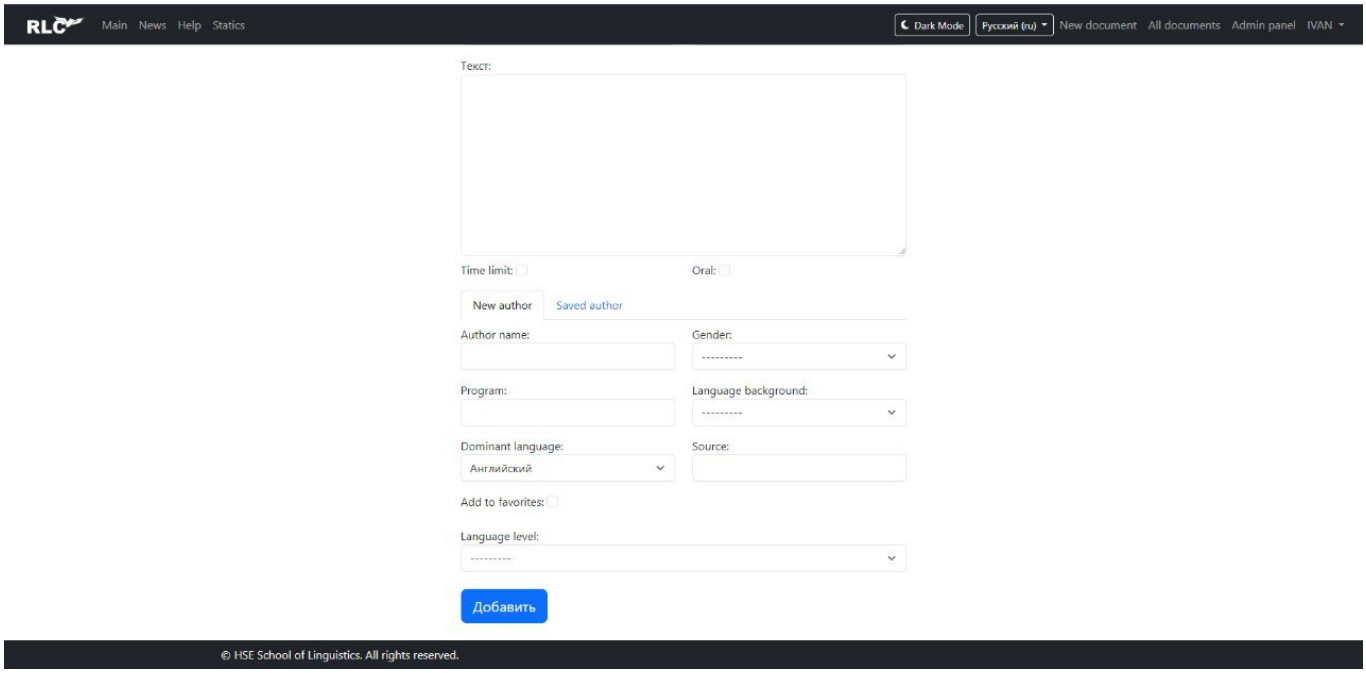

Рисунок 3. Интерфейс добавления автора текста вручную

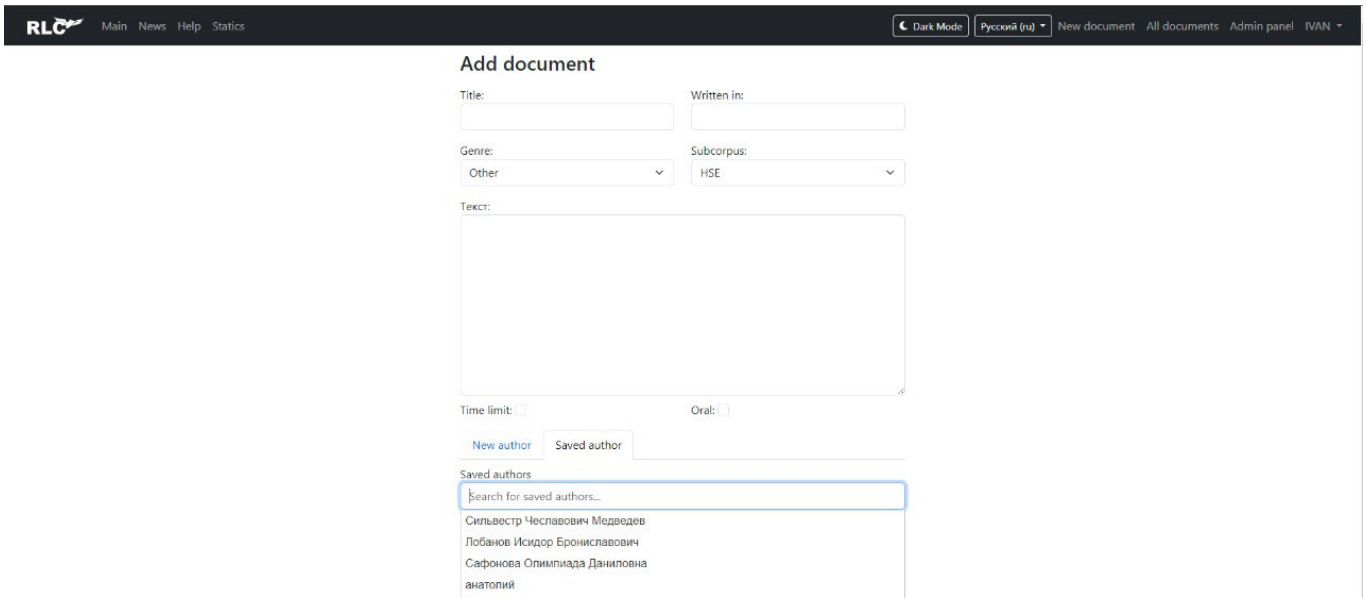

Рисунок 4. Интерфейс добавления автора текста автоматически, по его имени

3.2.5. Контекстный поиск по токенам

По задумке RLC должен представлять поиск текстов по заданным параметрам пользователя. Для удобного и эффективного поиска база данных была реализована на postgres. Для маршрутизации используется токенизация, чтобы по токенам найти список текстов. Осуществляется это с помощью следующего кода:

```
    def body_search(self, queryset, name, value):
        return queryset.filter(body__search=value)
...
    body = django_filters.CharFilter(
        method="body_search", label=_("Body search")
    )
```

3.2.6. Предобработка добавленных текстов

В изначальном проекте была серьезная ошибка с токенизацией, когда случайно аннататор мог неправильно выделить размеченную часть и вся аннотация съезжала. Данная проблема являлась по сути самой серьезной и решение ее одной уже было бы достаточно для заказчика. Решение реализовано через предобработку текстов, прежде чем добавлять их в базу данных. В основном убираются идущие подряд пробелы, это позволило разрешить большинство ошибок, связанных с некорректной аннотацией. В коде это реализовано следующим образом:

```
_RE_COMBINE_WHITESPACE = re.compile(r"\s+")
self.body = _RE_COMBINE_WHITESPACE.sub(" ", self.body).strip()
```

3.3. Описание и обоснование выбора метода организации входных и выходных данных

3.3.1. Описание метода организации входных и выходных данных

Входные данные программы – это данные, полученные при взаимодействии фронтэнда и бэкэнда посредством REST API запросов. Выходные же данные это JSON файлы, предоставляемые при выгрузке текстов и их аннотаций, а также ответы на действия пользователя.

3.3.2. Обоснования выбора метода организации входных и выходных данных

REST API – очень удобный механизм взаимодействия бэкэнд и фронтэнд разработки, который мы изучали на нашей образовательной программе. Его функциональности абсолютно достаточно, чтобы покрыть все требования разработки. JSON удобный формат хранения данных для последующей обработки, что полезно в машинном обучении.

3.4. Описание и обоснование выбора состава технических и программных средств

3.4.1. Состав технических и программных средств

Для корректной работы устройства с RLC необходимы минимальные системные требования:

1) тактовая частота процессора: не менее 1 ГГц
2) оперативная память: не менее 1 ГБ
3) свободное место на диске: не менее 1 ГБ

3.4.2. Обоснование выбора технических и программных средств

Перечисленные характеристики являются необходимыми для нормальной работы сайта.

4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

4.1. Ориентировочная экономическая эффективность

В рамках данной работы расчет экономической эффективности не предусмотрен.

4.2. Предполагаемая потребность

Данный проект находится в легкой доступности, поскольку для его использования всего лишь необходимо устройство с функционирующим браузером.

Данный сайт будет востребован иностранцами, а также исследователями в области русского языка.

4.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами

RLC – это в своем роде уникальный сайт, предоставляющий доступ к огромной базе данных аннотированных текстов, которых заинтересуют исследователи. Однако также сайт будет полезен и в образовательных целях, даже не смотря на конкурентов в интернете, позволяющих изучить русский язык. Но только у данного сайта в отличии от (образовательных роликов на ютуб, Duolingo, russianforfree, Babbel, Drops) содержится столь обширная учебная база.

5. СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1) ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2) ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3) ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4) ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5) ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7) ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8) ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9) ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

10) Django documentation [Электронный ресурс]. Режим доступа: https://docs.djangoproject.com/en/4.2/, свободный (Дата обращения 10.05.2023)
11) Bootstrap documentation [Электронный ресурс]. Режим доступа: https://getbootstrap.com/docs/5.3/getting-started/introduction/, свободный (Дата обращения 10.05.2023)
12) jQuery API Reference [Электронный ресурс]. Режим доступа: https://api.jquery.com/, свободный (Дата обращения 10.05.2023)
13) RecogitoJS API Reference [Электронный ресурс]. Режим доступа: https://github.com/recogito/recogito-js/wiki/API-Reference, свободный (Дата обращения 10.05.2023)
14) Docker documentation [Электронный ресурс]. Режим доступа: https://docs.docker.com/get-started/, свободный(Дата обращения 10.05.2023)
15) Web annotation data model [Электронный ресурс]. Режим доступа: https://www.w3.org/TR/annotation-model/, свободный (Дата обращения 10.05.2023)
16) RLC [Электронный ресурс]. Режим доступа: http://web-corpora.net/RLC, свободный (Дата обращения 10.05.2023)
17) Natasha — качественное компактное решение для извлечения именованных сущностей из новостных статей на русском языке [Электронный ресурс]. Режим доступа: https://natasha.github.io/ner/, свободный (Дата обращения 10.05.2023)

ПРИЛОЖЕНИЕ 1

ТЕРМИНОЛОГИЯ

Ниже приведен список необходимых терминов для ознакомления.

Эритажный – носитель языка, который изучал его в раннем возрасте и больше не пользуется им в повседневной жизни.

Django - бесплатный фреймворк для создания веб-приложений на языке Python

Токен – слово или знак препинания в предложении

Токенизация - разделение текста на токены

Аннотация – исправление ошибок в тексте с опциональной расстановкой тэгов и комментариями

JSON – формат обмена данными, основанный на синтаксисе объектов JavaScript, который позволяет представлять данные в виде пар ключ-значение и списков.

Фронтэнд – презентационная часть веб-сайта, его пользовательский интерфейс и связанные с ним компоненты.

Бэкэнд – серверная часть веб-сайта, отвечающая за формирование веб-страниц и логику приложения.

REST API – архитектурный стиль взаимодействия бэкенда и фронтэнда

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц в докум.)	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулированных					

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.09.11-01 81				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата