Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Гайнетдинова Вероника Андреевна

РАЗРАБОТКА СИСТЕМЫ АНАЛИЗА СТИЛИСТИКИ СИНТАКСИЧЕСКИХ КОНСТРУКЦИЙ НАУЧНЫХ ТЕКСТОВ НА АНГЛИЙСКОМ ЯЗЫКЕ

Выпускная квалификационная работа

по направлению подготовки *38.03.05 Бизнес-информатика*

образовательная программа «Бизнес-информатика»

Рецензент
к.ф.-м.н., заведующая кафедрой
математического обеспечения
вычислительных систем ПГНИУ

_____
С.И. Чуприна

Руководитель
старший преподаватель
кафедры информационных
технологий в бизнесе НИУ
ВШЭ – Пермь

_____
В.В. Ланин

Пермь, 2018 год

Аннотация

Работа посвящена разработке системы анализа стилистики синтаксических конструкций научных текстов на английском языке. Автором работы является Гайнетдинова Вероника Андреевна.

Выпускная квалификационная работа состоит из трех глав, а также введения и заключения. Первая глава содержит результаты исследования методов анализа текстов, обоснован выбор подходящих средств. Кроме того, проведен анализ существующих решений (систем и средств), позволяющих проводить интеллектуальный анализ текстов. Отдельно представлено описание корпусов документов, которые используются в данном исследовании. Вторая глава посвящена описанию алгоритмов для реализации системы, а также проектированию необходимой базы данных и архитектуры системы. Отдельно отражены алгоритмы для сбора метрик по тексту. В третьей главе отражен процесс программной реализации заявленной системы для анализа научных текстов на английском языке. Также описан процесс тестирования разработанной системы.

В данном документе 65 страниц и содержится 10 таблиц и 11 рисунков.

Оглавление

Введение 4
Глава 1. Задача автоматизированного анализа стиля текстов 7
1.1. Описание предметной области и ключевых терминов 7
1.2. Постановка задачи в целом и требования к системе 8
1.3. Обзор систем автоматизированного анализа текста 8
1.4. Описание алгоритмов и библиотек для автоматизированного анализа текстов на английском языке 11
1.5. Описание наборов эталонных документов (корпусов) для анализа 13
1.6. Экономическое обоснование разработки системы 14
1.7. Выводы по главе 17
Глава 2. Проектирование разрабатываемой системы 18
2.1. Спецификация требований к разрабатываемой системе 18
2.2. Синтаксические конструкции английского языка, значимые при оценке стиля текста и алгоритмы их поиска 19
2.3. Архитектура разрабатываемой системы 23
2.4. Проектирование базы данных, необходимой для реализации системы 25
2.5. Проектирование API разрабатываемой системы 26
2.6. Проектирование графического интерфейса 26
Глава 3. Реализация системы для анализа синтаксического стиля научных текстов на английском языке 28
3.1. Разработка серверной части приложения 28
3.2. Сбор статистики и проверка гипотез 31
3.3. Разработка клиентской части приложения 36
3.4. Тестирование разработанной системы 38
Заключение 40
Библиографический список 41
Приложения 42
Приложение А. Код серверной части приложения 42
Приложение Б. Код клиентской части приложения 55
Приложение В. Тесты серверной части приложения 64
Приложение Г. Тесты клиентской части приложения 65

Введение

Во многих сферах жизни (от чтения новостей до проведения научной работы) человек сталкивается с обработкой больших объемов информации с целью получения из нее полезных сведений. Проблема быстрой и качественной обработки информации остается актуальной в различных областях деятельности, так как до сих пор не существует ее универсального решения. Отсутствие единого инструмента для всестороннего анализа текста обусловлено тем, что есть совершенно разные стороны для анализа: синтаксис языка, семантика текста, стилевое оформление (как со стороны синтаксиса, так и со стороны лексики) и т.д. На данный момент реализованы ряд систем, позволяющих проводить интеллектуальный анализ текстов, например, для поиска по тексту или определения эмоционального окраса [1-3]. Интеллектуальный анализ текстов (text mining) – это процесс выделения полезной информации из исходного текста с использованием средств искусственного интеллекта [4]. В том числе такие средства могут применяться для анализа стиля текста.

В лингвистике выделяют понятие функционального стиля речи. Функциональный стиль – это система речевых средств языка, которые используются в той или иной сфере общения [5]. Например, выделяют научный стиль, который применяется в науке и научных журналах. Основная функция такого стиля – сообщить информацию, а также доказать ее истинность. Анализ стилистики синтаксических конструкций научных текстов может найти свое применение, например, при проверке научных статей на соответствие правилам издательства или для определения «уровня» статьи с точки зрения используемого синтаксиса. Учитывая большие объемы научных текстов, которые должны регулярно проверяться и отсеиваться (в частности, издательствами), проблема быстрого и качественного анализа стиля синтаксических конструкций является актуальной. Таким образом, вытекает противоречие между требованиями к синтаксическому стилю научных работ с одной стороны, и отсутствием четко сформулированных критериев и автоматизированной системы проверки стилистики синтаксических конструкций текстов, с другой стороны.

Исходя их вышесказанного, проблема исследования заключается в отсутствии инструмента для автоматизированного анализа стилистики синтаксических конструкций научных текстов на английском языке. Объектом же данной работы являются синтаксические конструкции документов. Предметом исследования являются средства компьютерной лингвистики для проверки стиля синтаксических конструкций текстов.

Целью исследования является разработка системы анализа стилистики синтаксических конструкций научных текстов на английском языке на основе сбора данных по настроенным метрикам (определенным экспертами).

Для достижения поставленной цели необходимо решить ряд задач, а именно:

1. Провести аналитическое исследование существующих методов анализа текстов, выбрать подходящие методы для стилистического анализа синтаксических конструкций текстов на английском языке.
2. Подготовить набор эталонных документов на английском языке для сравнительного анализа с проверяемым текстом и формирования набора показателей (маркеров).
3. Подготовить набор тестовых документов на английском языке для тестирования работы системы.
4. Выполнить проектирование модели хранения и высокоуровневого представления системы, в том числе используемых алгоритмов, с помощью современных языков моделирования.
5. Сформировать показатели (осмысленные, которые будут понятными пользователю) и алгоритмы расчета метрик по ним на основе анализа эталонных документов.
6. Разработать систему анализа стилистики синтаксических конструкций научных текстов на английском языке.
7. Протестировать систему на тестовых документах.

На данный момент уже были совершены подходы к решению проблемы анализа стиля синтаксических конструкций научных текстов на английском языке. Так, в 1992 году в одном из университетов Канады была разработана система для обучения студентов научному письменному стилю [1]. Однако, эта система обладала рядом ограничений, например, позволяла обрабатывать тексты только по одному предложению.

На этапе проведения исследования будут использованы следующие методы: анализ классификация, сравнение, моделирование, а также методы объектно-ориентированного программирования.

На защиту выносятся:

1. Набор критериев, определяющих стиль синтаксических конструкций научного текста на английском языке.
2. Разработанные алгоритмы анализа стилистических особенностей синтаксических конструкций.
3. Автоматизированное средство (система) для выполнения анализа синтаксического стиля научных текстов.

В ходе работы был выявлен и описан набор критериев, имеющих значительный вес при анализе синтаксического стиля научных документов. Разработаны алгоритмы расчета метрик для анализа стиля синтаксических конструкций научных текстов на английском языке. Реализована система, автоматизирующая проверку стилистики синтаксиса в научных текстах.

Глава 1. Задача автоматизированного анализа стиля текстов

В главе описывается предметная область проводимого исследования; постановка задачи анализа стилистики синтаксических конструкций в общем виде; сравнительный анализ существующих программных средств; описание возможностей выбранных средств, которые будут использованы при разработке; назначение системы, а также детализация требований к системе в целом.

1.1. Описание предметной области и ключевых терминов

Прежде всего необходимо привести определения ключевых понятий и терминов.

Синтаксический анализ (парсинг) – понятие, используемое в компьютерной лингвистике для обозначения процесса сопоставления последовательности лексем (токенов, слов) формального языка с его формальной грамматикой. Программа, выполняющая синтаксический анализ, называется *парсером (или синтаксическим анализатором)* [4].

Обычно результатом синтаксического анализа является построенное синтаксическое дерево (дерево разбора). *Синтаксическое дерево (дерево разбора)* – ориентированное дерево, имеющее корневую вершину, которое является представлением синтаксической структуры некоторой строки в соответствии с контекстно-независимой грамматикой [4].

Под *маркером* в работе будет подразумеваться некая известная специфическая последовательность синтаксических средств, отличающая хороший синтаксический стиль научного текста на английском языке. *Метрикой* же будем называть конкретное значение, рассчитанное по рассматриваемой выборке документов, для заданного маркера. Например, маркером может быть наличие пассивного залога в работе, а метрикой для этого маркера будет конкретное значение – 6-8%.

Во время разработки системы будет необходимо выявить маркеры и рассчитать метрики, присущие хорошему синтаксическому стилю. Для этой цели необходимо подготовить две выборки данных: обучающую и тестовую. *Обучающая выборка* будет использоваться во время формирования (выявления) маркеров и подсчета метрик во время разработки системы, а *тестовая выборка* необходима для валидирования работы системы (тестирование выдаваемых результатов).

1.2. Постановка задачи в целом и требования к системе

Основной задачей данной работы является разработка системы, позволяющей производить автоматизированный анализ стиля синтаксических конструкций английских научных текстов и выдавать рекомендации по улучшению стиля анализируемой работы.

Поставленная задача логично разбивается на следующие подзадачи:

1. Выделение синтаксических конструкций предложения (определение основных характеристик входящего предложения).
2. Сбор маркеров и конкретных значений по ним (метрик), являющиеся индикаторами синтаксического стиля. Автоматизация расчета метрик.
3. Выдача рекомендаций по стилю текста для улучшения качества проанализированной работы с точки зрения синтаксиса.
4. Расчет пользовательских метрик, а также проведение статического синтаксического анализа по заданным правилам.
5. Интерфейсная часть приложения – пользовательский интерфейс для работы с системой (веб-приложения).

Таким образом, описанные подзадачи позволяют выделить основные модули разрабатываемой системы. Система автоматизированного анализа стилистики синтаксических конструкций научных текстов на английском языке будет состоять из пяти частей: «Синтаксический анализ», «Сбор метрик», «Стилевые рекомендации», «Пользовательские правила» и «Веб-приложение». Функциональные требования, предъявляемые к каждому из модулей, будут описаны на этапе проектирования системы.

1.3. Обзор систем автоматизированного анализа текста

В качестве аналогов разрабатываемой системы будут рассмотрены приложения, предназначенные для анализа текстов на английском языке. Для анализа выбраны системы, которые позволяют производить разносторонний автоматизированной анализ текстов. Будут проанализированы следующие проекты:

1. DocBridge Delta (Compart) [2] – программное решение, позволяющее производить проверку качества документа, основываясь на сравнении двух файлов (текущей и предыдущей версии) для выявления появившихся проблем качества. Система позволяет производить проверку текста на соответствие заданному набору правил, а также запускать автоматизированные тесты.

2. PaperRater [6] – автоматизированная система, позволяющая производить интеллектуальный анализ текста, включающий проверку грамматики, подсчет процента заимствований, выдачу инструкций по улучшению.
3. After the Deadline [3] – система с открытым исходным кодом, использующая методы искусственного интеллекта для проведения анализа текстов, написанных на естественном языке. Приложение находит ошибки в текстах (написание слов и грамматика), а также генерирует предложения по улучшению стиля (с точки зрения разговорности, то есть упрощает текст с точки зрения используемых выражений).
4. OnlineCorrection.com [7] – это сервис, проверяющий тексты на английском языке на наличие ошибок написания слов, базовой грамматики, а также стилевых ошибок. Программа также умеет генерировать предложения по улучшению текста.
5. LanguageTool [8] – это приложение с открытым исходным кодом, позволяющее производить проверку наличия ошибок в текстах на различных языках.
6. Scribens [9] – система, предоставляющая инструмент для проверки грамматических и стилевых ошибок в текстах. Также предоставляется статистические данные для проверяемого текста, такие как количество слов, предложений, параграфов и другие.

Для более детального сравнения описанных выше систем необходимо разработать ряд критериев. Выбранные критерии представлены ниже:

1. Выдача рекомендаций по улучшению стиля.
2. Проверка синтаксического научного стиля.
3. Оценка «уровня» текста с точки зрения научного стиля.
4. Установка пользовательских правил.
5. Приложение доступно без установки дополнительного программного обеспечения (web-версия).
6. Бесплатная версия.
7. Обработка текста целиком.
8. Работа с документами в формате pdf и doc.
9. Предоставление статистики по тексту.

Визуализация сравнения систем на основе выделенных критериев представлена в таблице 1.1.

Таблица 1.1. Сравнение систем для автоматического анализа текстов

Критерий	DocBridge Delta	PaperRater	After the Deadline	Online Correction	Language Tool	Scribens
Выдача рекомендаций по улучшению стиля	-	+	+	+	-	+
Проверка синтаксического научного стиля	-	-	-	-	-	-
Оценка «уровня» текста с точки зрения научного стиля	-	-	-	-	-	-
Установка пользовательских правил	+	-	+	-	-	-
Приложение доступно без установки дополнительного программного обеспечения (web-версия)	+	+	+	+	+	+
Бесплатная версия	-	+	+	+	+/-	+
Обработка текста целиком	+	+	+	+	+	+
Работа с документами в формате pdf и doc	+	-	-	-	-	-
Предоставление статистики по тексту	-	+	-	-	-	-

Изучив полученные результаты сравнения, можно сделать вывод, что среди описанных выше систем нет приложения, которое бы полностью соответствовало заявленным критериям. Таким образом, систему для анализа стилистики синтаксических конструкций текстов на английском языке необходимо разработать.

1.4. Описание алгоритмов и библиотек для автоматизированного анализа текстов на английском языке

Для разработки системы, позволяющей производить анализ синтаксического стиля текстов, потребуется ряд средств, предназначенных для автоматизированного анализа текста. В частности, это библиотеки и алгоритмы для парсинга документов, построения синтаксических деревьев и выявления частей речи в предложениях.

Наиболее известный парсер в области обработки текстов на естественных языках это разработка Стэндфордского университета – The Stanford Parser, который является статистическим парсером. Данный пакет программ написан на языке программирования Java и является реализацией вероятностного парсера для английского языка. Данный парсер при разборе предложения строит синтаксическое дерево (дерево зависимостей), а также помечает все слова соответствующими тегами.

Стэндфордский парсер позволяет добавлять свои теги для разметки предложений. Однако для хороших результатов следует убедиться, что предоставлен правильно размеченные примеры и использованы верные имена для тегов. Важно отметить, что описанная библиотека не подходит для проверки грамматики в приложениях. Проверить работу парсера можно с помощью веб-интерфейса, предоставляемого официальном сайтом разработчика библиотеки (рисунок 1.1). Для проверки будем использовать предложение, взятое из научного текста по политологии: «Then we fit a statistical model and discuss results and implications».

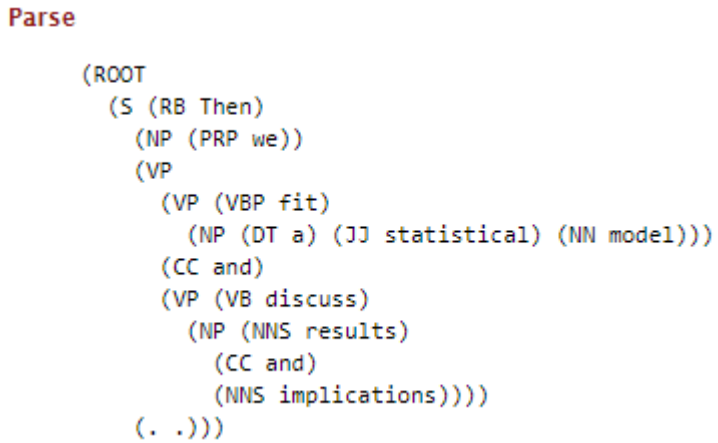

Рисунок 1.1. Результат работы стенфордского парсера

Другой полезной библиотекой для синтаксического парсинга текстов является SyntaxNet, разработанная в Google. SyntaxNet – это фреймворк основанная на технологии нейронных сетей. На вход функции-обработчику подается предложение, написанное на естественном языке. Каждое слово во входном предложении маркируется тегом, обозначающим часть речи, который описывает синтаксическую функцию слова. А также устанавливаются связи между словами в предложении, которые представляются в виде направленного дерева разбора. Пример синтаксического разбора предложения представлен на рисунке 1.2 (для наглядности опущены некоторые параметры, описывающие токены). Для проверки также воспользуемся предложением из научной работы по политологии: «Then we fit a statistical model and discuss results and implications».


```
[{"pos_tag": "VBP",
  "word": "fit",
  "dep": "root",
  "contains": [{"pos_tag": "RB",
                "word": "Then"},
               {"pos_tag": "PRP",
                "word": "we"},
               {"pos_tag": "NN",
                "word": "model",
                "contains": [{"pos_tag": "DT",
                              "word": "a"},
                             {"pos_tag": "JJ",
                              "word": "statistical"},
                             {"pos_tag": "NNS",
                              "word": "results",
                              "contains": [{"pos_tag": "CC",
                                            "word": "and"},
                                           {"pos_tag": "NN",
                                            "word": "discuss"}],},
                             {"pos_tag": "NNS",
                              "word": "implications",
                              "contains": [{"pos_tag": "CC",
                                            "word": "and"}]}
                            ]
                },
                {"pos_tag": ".",
                 "word": "."}
               ]
}]
```


Рисунок 1.2. Синтаксический разбор предложения SyntaxNet

Описанный выше фреймворк для синтаксического разбора текстов SyntaxNet будет использован в процессе разработки системы анализа стилистики синтаксических конструкций научных текстов на английском языке. Выбор обусловлен тем, что SyntaxNet осуществляет синтаксический разбор предложения значительно быстрее Стэндфордского парсера, а также предоставляет более подробную информацию о разобранном предложении

1.5. Описание наборов эталонных документов (корпусов) для анализа

Перед этапом проработки архитектуры системы и непосредственно разработки необходимо подготовить набор эталонных документов на английском языке для их анализа с целью получения целевых показателей и метрик. Полученные характеристики будут использовать для сравнительного анализа эталонных документов с проверяемым текстом. Важно, что под набором эталонным документов в данном контексте понимаются не только «хорошие», с точки зрения синтаксического стиля, статьи, но и также «плохие». В частности, имеется в виду, что будут использованы стилистически верные работы (образцы) – работы профессионалов, компетентных авторов научных текстов, а также тексты, стилистический стиль которых необходимо сравнить в образцами – работы студентов НИУ ВШЭ-Пермь разных направлений подготовки.

Научно-учебной группой НИУ ВШЭ в Перми «Разработка программного обеспечения для проведения корпусных исследований английского языка» был собран ряд корпусов студенческих работ [10]. Корпусы разделены по направлениям подготовки студентов, а именно: менеджмент, экономика, политология, история, право, бизнес-информатика и программная инженерия. Описанные шесть корпусов содержат работы Research Proposal, написанные студентами НИУ ВШЭ в рамках дисциплины «Академическое письмо на английском языке». Документы представлены в формате txt.

Корпусы профессиональных текстов представлены в виде набора статей, написанных компетентными авторами (в основном носителями языка). Данные статьи выпущены под именами известных университетов (например, Оксфорд). Корпусы профессиональных работ будут использованы во время статистического анализа для выявления синтаксических маркеров хорошего научного стиля текстов на английском языке. Для валидирования найденных маркетов будут проводиться сравнения с работами студентов, изучающих английский.

Описанные корпусы статей будут разделены на два набора: обучающая выборка и тестовая. Обучающая выборка будет использовать для получения показателей и метрик, являющихся индикаторами «хорошего» синтаксического стиля научного текста. Тестовый набор будет использован для оценки того, насколько правильные результаты выдает система при проверке работы.

1.6. Экономическое обоснование разработки системы

В рамках представленной выпускной квалификационной планируется разработка системы для анализа стилистики синтаксических конструкций научных текстов на английском языке. Данная система разрабатывается для кафедры английского языка НИУ ВШЭ-Пермь и может быть использована для проверки студенческих работ перед дальнейшей публикацией в научных журналах.

Разработка системы включается в себя ряд этапов:

1. Анализ предметной области.
2. Поиск и подготовка корпусов научных текстов, которые будут взяты за основу при разработке системы.
3. Статистический анализ найденных корпусов, формирование правил хорошего стиля.
4. Программная реализация модуля для автоматизированной проверки стилистики синтаксических конструкций научных текстов на английском языке.
5. Создание веб-оболочки пользовательского интерфейса.
6. Тестирование приложения и исправление ошибок.

Далее следует определить ресурсные затраты на разработку проекта с точки зрения участников. Список участников и соответствующие им роли представлены в таблице 1.2.

Таблица 1.2. Роли участников проекта

Участник проекта	***Роль***
Гайнетдинова Вероника Андреевна	Автор выпускной квалификационной работы, аналитик, а также разработчик и тестировщик системы
Ланин Вячеслав Владимирович	Научный руководитель выпускной квалификационной работы
Стринюк Светлана Александровна	Эксперт-консультант в области академического английского

Распределение ролей по этапам разработки, а также временные затраты представлены в таблице 1.3. В последней колонке представлен итоговый подсчет времени, необходимого на каждый из этапов.

Таблица 1.3. Распределение ролей по этапам разработки

Название этапа	***Роль***	***Временные затраты***	
1. Анализ предметной области.	Аналитик	2 дня	2 дня
2. Поиск и подготовка корпусов научных текстов, которые будут взяты за основу при разработке системы.	Аналитик	3 дня	3 дня
	Научный руководитель	1 день	

Название этапа	***Роль***	***Временные затраты***	
3. Статистический анализ найденных корпусов, формирование правил хорошего стиля.	Аналитик	3 дня	5 дней
	Разработчик	2 дня	
	Эксперт-консультант	1 день	
	Научный руководитель	1 день	
4. Программная реализация модуля для автоматизированной проверки стилистики синтаксических конструкций научных текстов на английском языке.	Разработчик	28 дней	28 дней
5. Создание веб-оболочки пользовательского интерфейса.	Разработчик	7 дней	7 дней
6. Тестирование приложения и исправление ошибок.	Разработчик	7 дней	10 дней
	Тестировщик	3 дня	

Таким образом, разработка всего приложения займет 55 дней, что приблизительно равняется 2 месяцам. Так как проект разрабатывается в рамках выпускной квалификационной работы и делается по заказу кафедры иностранных языков, на него не будет выделен отдельный бюджет. Поэтому все временные затраты участников проекта не будут оплачены материально. Однако возможно подсчитать недополученную прибыть участников, и соответственно расходы на выполнение проекта. Расходы представлены в таблице 1.4.

Таблица 1.4. Затраты на выполнение проекта

Роль	***Ставка***	***Время работы***		***Затраты (руб)***
Аналитик	180 руб/ч	8 дней	64 часа	11 520
Разработчик	340 руб/ч	44 дня	352 часа	119 680
Тестировщик	250 руб/ч	3 дня	24 часа	6 000
Научный руководитель	350 руб/ч	2 день	16 часов	5 600
Эксперт-консультант	375 руб/ч	1 день	8 часов	3 000

Таким образом, получаем итоговые приблизительные затраты на разработку описанной системы в 145,8 тысяч рублей. В дальнейшем разработанная система может быть монетизирована. Некоторые из возможных путей монетизации представлены ниже:

1. Платные подписки для сервиса – запуск нескольких планов работы с системой, например, базовый, премиум и студенческий (предполагается предоставлять студентам бесплатный доступ к полной версии сервиса).
2. Получение пожертвований (благодарностей за разработку сервиса, выраженных в денежной форме).
3. Показ рекламы на сайте сервиса.
4. Продать сервис другой компании.

5. Добавить элементы биржи в бизнес-модель сервиса – создать площадку, где знатоки английского смогут за некоторую плату в ручную проверять предложенные документы. Монетизация – взимание комиссии с транзакций, совершенных на сайте.

Наиболее рабочим методом монетизации для разрабатываемого сервиса может стать первый способ. Запуск трех видов подписок потенциально может покрыть издержки на разработку. В таблице 1.5 представлены примерные параметры выбранного способа монетизации. Предположим, что сервисов пользуется около 2,5 тыс. человек, из которых около 125 человек приобрели премиум подписку, а 350 человек пользуются студенческой подпиской сервиса, за которую заплатили 5 вузов.

Таблица 1.5. Монетизация за счет подписок

Вид подписки	***Стоимость в месяц (руб)***	***Количество купивших (чел)***	***Ожидаемая прибыль (руб)***
Базовый пакет	0	1 925	0
Премиум пакет	700	125	85 700
Студенческий пакет	0 (2000 руб. для вуза)	450 (5 вузов)	0 (10 000)

Таким образом, предположительная прибыль составит около 95,7 тысяч рублей в месяц. При этом необходимо учесть также переменные расходы на содержание сайта, а именно оплату хостинга (в среднем 5 000 руб/месяц) и администрирование (своими силами). Выбранный способ монетизации предположительно окупит издержки уже через два месяца существования (при условии, что около 5 вузов и 125 человек будут оплачивать расширенную подписку). Так как такое количество пользователей появится не в первый день, необходимо также закладывать издержки на рекламу сервиса.

1.7. Выводы по главе

Аналитическое исследование позволило выявить необходимость разработки системы для автоматизированного анализа стилистики синтаксических конструкций научных текстов на английском языке. Необходимость обусловлена отсутствием специализированного инструмента для анализа синтаксического стиля английских научных текстов. Также были описаны современные программные пакеты (библиотеки), которые позволяют производить синтаксический разбор предложений. Для реализации системы был выбран фреймворк SyntaxNet, потому что он позволяет произвести быстрый (по сравнению с другими программными решениями) синтаксический разбор предложений и предоставляет очень подробную информацию о проанализированном предложении.

Кроме того, выполнено экономическое обоснование разработки системы. Чтобы окупить расходы на разработку системы, которые составили примерно 145 тысяч рублей, в дальнейшем после разработки основной функциональности системы могут быть добавлены элементы для монетизации, а именно: платные подписки на использование системы.

Глава 2. Проектирование разрабатываемой системы

В главе содержатся требования к разрабатываемой системе для анализа стилистики синтаксических конструкций научных текстов на английском языке, а также описаны алгоритмы, которые будут использованы при реализации системы и представлен список значимых при оценке синтаксического стиля текста синтаксических конструкций.

2.1. Спецификация требований к разрабатываемой системе

На этапе анализа исследуемой проблемы анализа стиля синтаксических конструкций текстов на английском языке были выделены пять модулей. Детализация требований к каждому из модулей представлена ниже:

1. Модуль «Синтаксический анализ»:
    1.1. Осуществление разбора предложения согласно описанной формальной грамматике английского языка: то есть выделение синтаксических единиц заданного предложения.
    1.2. Осуществление разбора текста согласно описанной формальной грамматике, определение контекстных синтаксических особенностей предложений и текста в целом.
2. Модуль «Сбор метрик»:
    2.1. Создание набора показателей (и формул для их расчета), расчет которых необходим для проведения анализа стилистики синтаксических конструкций текстов на английском языке.
    2.2. Автоматизация расчета метрик по выявленным показателям.
    2.3. Сбор статистической информации (расчет метрик) по обучающей выборке.
3. Модуль «Стилевые рекомендации»:
    3.1. Выдача ответа о качестве текста в бинарном виде (хороший/плохой).
    3.2. Выдача ответа о качестве текста в виде процента «похожести» на эталонные работы.
    3.3. Выдача предположений рекомендательного характера о необходимых изменениях текста для приближения проверяемой работы к эталонному состоянию.

3.4. Объяснение полученных результатов (описание собранных метрик, расшифровка приведенных показателей).

4. Модуль «Пользовательские правила»:
   4.1. Предоставление возможности пользователю добавить свои показатели и определить правила расчета значений по ним.
   4.2. Расчет метрик и выдача результатов по заданным пользователем показателям.
   4.3. Проведение статического синтаксического анализа текстов по заданным пользователем правилам.
   4.4. Предоставление возможности пользователю определять свои правила для выдачи сообщений после автоматического анализа текста.
   4.5. Выдача заданных пользователем рекомендаций в соответствии с описанными им правилами.
5. Модуль «Веб-приложение»:
   5.1. Визуальное представление работы системы: от загрузки пользовательского документа до выдачи рекомендаций.
   5.2. Корректная работа в современных браузерах (Google Chrome, Yandex Browser).

2.2. Синтаксические конструкции английского языка, значимые при оценке стиля текста и алгоритмы их поиска

Анализ синтаксического стиля научных текстов на английском языке будет производится путем статистического подсчета различным метрик, отражающих синтаксические особенности предложений и теста в целом. Экспертами были определены ряд правил английского языка для написания письменных работ в научном стиле [11-12]. В качестве гипотез выдвигается значимость следующих синтаксических конструкций при оценивании синтаксического стиля текста:

1. Количество подлежащих в предложении – среднее значение, максимальное и минимальное значения, медиана, мода.
2. Использование «relative connectors» (who, in which, at which, when, into which, who, that, where, whose, whom) – частота использования, рассчитанная на предложение/абзац.

3. Использование деепричастных оборотов (и дополнений) – отдельный подсчет по типу оборота (описывающие время, место и т.д.) – частота использования.
4. Использование conjunctions (co-coordinate and subordinate) – частота использования.
5. Использование пассивного залога – частота на единицу текста, местоположение вхождений (введение, основная часть, заключение).
6. Наличие базовой структуры научного текста – введение, основная часть, заключение.
7. Использование вопросительных предложений – частота использования.
8. Использование условных предложений – частота использования.
9. Использование связок\ссылок на предыдущие предложения (this, that, these, those) – частота использования.

Для каждой из выборок можно рассчитать среднеквадратичное отклонение от среднего эталонного значения (посчитанного для выборки работ, написанных профессиональными авторами). Кроме того, можно учитывать глубину синтаксического дерева каждого из предложений и собирать общую статистику по проверяемому тексту – среднее значение, минимальное и максимальное значения, медиану, моду.

Далее важно определить алгоритмы расчета метрик по каждому показателю, описанному выше. Приведем статистические формулы, необходимые для расчетов [13].

Среднее значение рассчитывается как среднее арифметическое:

$$\bar{\alpha} = \frac{1}{n}\sum_{i=1}^{n} \alpha_i,$$

где α_i – элемент выборки, n – размер выборки.

Среднеквадратичное отклонение – показывает абсолютное отклонение измеренных значений от среднего арифметического, рассчитывается как:

$$\delta = \sqrt{\frac{\sum_{i=1}^{n}(\alpha_i - \bar{\alpha})^2}{n-1}},$$

где α_i – элемент выборки, $\bar{\alpha}$ – выборочное среднее, n – размер выборки.

Полезным может стать также и коэффициент вариации, который характеризует меру отклонения измеренных значений от среднего. Чем меньше значение коэффициента, тем относительно меньший разброс и большая выровненность исследуемых значений. Коэффициент вариации рассчитывается по формуле:

$$V = \frac{\delta}{\bar{\alpha}} * 100\%,$$

где $\bar{\alpha}$ *– выборочное среднее,* δ *– среднеквадратичное отклонение.*

Медиана – это величина изучаемого признака, которая расположена в середине ряда, состоящего из упорядоченных числовых значений этого признака.

Мода – это наиболее часто встречаемое значение признака вариационного ряда (то есть значение признака, обладающее наибольшей частотой).

Для определения интервалов допустимых значений можно воспользоваться квантилем. Квантиль для эмпирической интервальной выборки рассчитывается по формуле:

$$Q_j = x_{\text{н}Q_j} + i_{Q_j} \frac{j * \sum f_i - S_{Q_j - 1}}{f_{Q_j}},$$

где j *– уровень квантиля,* $x_{\text{н}Q_j}$ *– нижняя граница интервала, содержащего квантиль (интервал определяется по накопленной частоте интервалов),* i_{Q_j} *– ширина интервала, содержащего квантиль,* $S_{Q_j - 1}$ *– накопленная частота интервала, предшествующего интервалу, содержащему квантиль,* f_{Q_j} *– частота интервала, содержащего квантиль.*

Помимо статистических формул, необходимо также понимать, каким образом определять наличие той или иной синтаксической конструкции в предложении. Алгоритмы поиска описанных выше синтаксических конструкций представлены в таблице 2.1. В SyntaxNet используются стандартные аббревиатуры (тэги) для описания назначения токена в предложении [14].

Таблица 2.1. Определение синтаксических конструкций в тексте

Синтаксический маркер	***Алгоритм поиска***
Подлежащие в предложении	Определяется во время построения синтаксического дерева, маркируется парсером как Subj. Необходимо подсчитать количество слов, отвеченных тэгом Subj в предложении.
Использование «relative connectors»	Поиск заданных вхождений в строку. Подстроки для поиска: who, in which, at which, when, into which, who, that, where, whose, whom.

Синтаксический маркер	***Алгоритм поиска***
Использование деепричастных оборотов (и дополнений)	В качестве маркеры используются тэги, полученные после синтаксического разбора предложения. tmod – модификатор времени; тэг указывает на то, что данное слово служит для указания времени. agent – дополнение к пассивному глаголу, следующее после связки «by». num – числовой модификатор. purpcl – модификатор указывающий на цель (например, in order to).
Использование conjunctions	Общий тэг – «conj». Co-ordinate (используется тэг «cc»): …and (then)…; not only…, but also; …but…; …(and) yet…; …(and) so…; …(and) hence…; …or (else)… Subordinate: , who…; , which…; , where…; , when…; although…; whereas…; while…; in spite of the fact that…; despite the fact that…; so (that)…; because…; due to the fact that…; if…; as/so long as…; before…; after…; , after which…; when…; now that…
Использование пассивного залога	Поиск синтаксических структур: Subj + finite form of *to be* + Past participle.
Наличие базовой структуры научного текста	Определение введения по ключевым словам: начинается с заголовка Introduction, заканчивается следующим заголовком. Заключение – начинается после ключевого слова Conclusion.
Использование вопросительных предложений	Два подхода: 1. Поиск вопросительных знаков в конце каждого из предложений (проверить вхождение символа «?»). 2. Поиск синтаксических конструкций присущих вопросительным предложениям: инвертированный порядок подлежащего и сказуемого, использование вопросительных слов.
Использование условных предложений	Поиск синтаксических конструкций присущих условным предложениям в английском языке: 1. Нулевой тип – использование настоящего времени в обоих частях предложения, наличие ключевых слов: if, unless, when. 2. Первый тип – использование простого настоящего времени в части с условием и использование простого будущего времени во второй части; наличие ключевых слов: if, unless. 3. Второй тип – использование простого прошедшего времени в части с условием (if-clause), would/might + inf во второй части. 4. Третий тип – использование совершенного прошедшего времени в части с условием (if-clause) и would/might + have + past participle во второй.
Использование связок\ссылок на предыдущие предложения	Поиск вхождения ключевых слов: this, that, these, those.

2.3. Архитектура разрабатываемой системы

Архитектура разрабатываемой системы должна быть клиент-серверной с тонким клиентом. Под тонким клиентов подразумевается программа-клиент, которая переносит большую часть задач на сторону серверного приложения.

Архитектура разрабатываемой системы должна включать:

1. Клиентское приложение (далее – клиент).
2. Серверное приложение (далее – сервер).
3. База данных.

Взаимодействие между компонентами в виде диаграммы компонент (UML) представлено на рисунке 2.1. Стрелками обозначено инициирование взаимодействия.

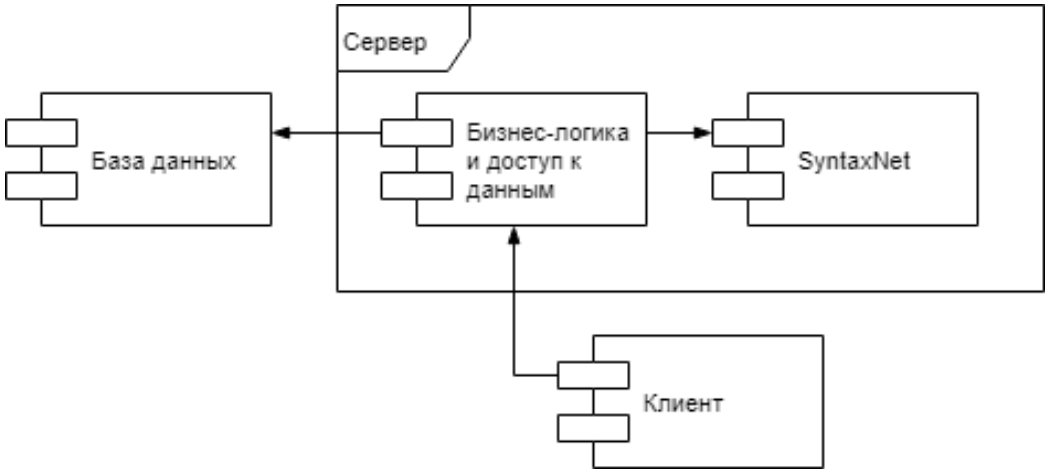

Рисунок 2.1. Архитектура разрабатываемой системы

Алгоритм работы разрабатываемой системы проиллюстрирован диаграммой последовательности, выполненной на языке моделирования UML (рисунок 2.2).

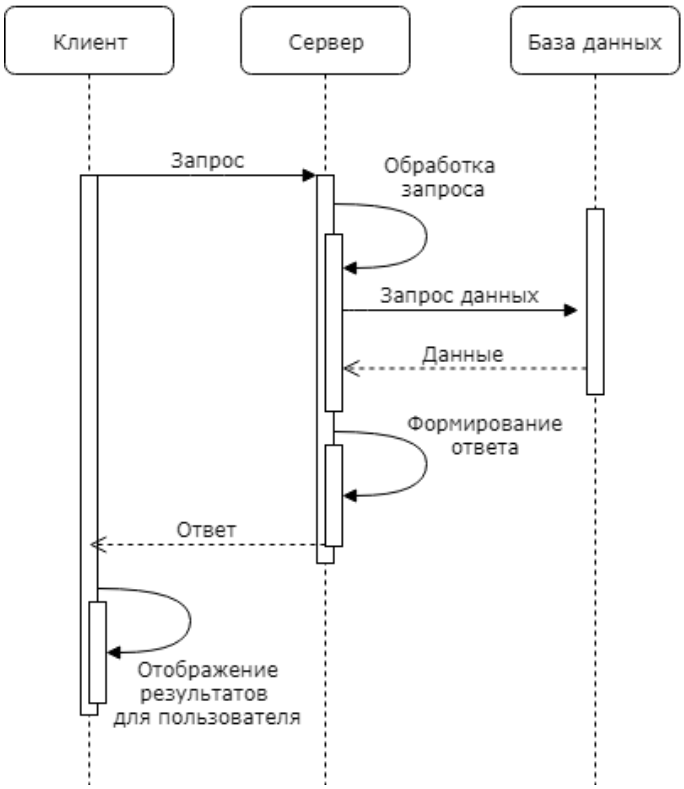

Рисунок 2.2. Диаграмма последовательности

Далее представлено описание каждого из логических компонентов системы:

1. Клиентское приложение – тонкий клиент, реализующий интерфейс пользователя; позволяет вводить пользовательские данные и отображать результаты работы в виде, понятном для пользователя. Обеспечивает пользовательское взаимодействие с системой.
2. Серверное приложение – приложение, реализующее большую часть бизнес-логики системы; реализует взаимодействие с базой данных (запись и чтение данных); производит синтаксический анализ пользовательских документов и возвращает результат клиенту; состоит из двух модулей: SyntaxNet – осуществляет синтаксический разбор заданного предложения и модуль, осуществляющий бизнес-логику и предоставляющий ресурсы пользователю.
3. База данных – хранилище данных системы; взаимодействие с сервером осуществляется посредством запросов.

Разрабатываемый веб-сервис должен соответствовать архитектурному стилю взаимодействия компонентов REST. Таким образом, вызов удаленной процедуры будет представлять собой REST-запрос (обычно GET и POST), а все необходимые данные будут передаваться в теле или параметрах запроса. В данном случае это позволит упростить архитектуру и ускорить этап разработки.

2.4. Проектирование базы данных, необходимой для реализации системы

Описание сущностей базы данных, необходимых для реализации системы представлено в таблице 2.2.

Таблица 2.2. Описание сущностей базы данных

Сущность	***Атрибуты***	***Тип данных***	***Описание***
Пользователь (User)	Login	String	Служит для разграничения рабочих областей пользователей и привязки истории к конкретному пользователю
Научная работа (Paper)	Id	String	UUID объекта
	User	String	Логин пользователя, который загрузил данную работу
	MetricsId	String	UUID объекта с рассчитанными метриками по данной работе
Узел синтаксического дерева ParsedWord	posTag	String	Часть речи
	word	String	Анализируемое слово
	dep	String	Обозначение роли в предложении
	number	String	Число: единственное или множественное (относится к существительным)
	mood	String	Настроение, характеризующее слово
	degree	String	Степень сравнения (присуще прилагательным)
	tense	String	Время (присуще глаголам)
	contains	[ParsedWord]	Дочерние вершины (зависимые слова)
Результаты проверки (PaperResult)	Id	String	UUID объекта
	PaperId	String	UUID проверенной работы
	Percent	Double	Процент близости работы к образцу
	Checks	[Check]	Значение по каждой из метрик
Проверка (Check)	Id	String	UUID объекта
	totalCount	Integer	Общее количество подсчитываемого значения на весь текст
	average	Double	Среднее количество подсчитываемого значения на предложение в тексте
Правила для проверки CheckConfig	Id	String	UUID объекта
	paperLabel	String	Метка, обозначающая тематику работы
	minValue	Double	Минимальное значение по заданной метрике
	maxValue	Double	Максимальное значение по заданной метрике
	checkIden	String	Метка, обозначающая название метрики

Объект CheckConfig используется для хранения и задания правил синтаксического стиля, выявленных по корпусам эталонных документов.

2.5. Проектирование API разрабатываемой системы

В соответствии с решаемой в данной работе задаче по разработке системы для анализа стиля синтаксических конструкций научных тестов на английском языке, необходимо спроектировать минимальное API для полноценной работы системы. Ниже в формате списка представлен перечень необходимых ресурсов для решения поставленной задачи:

1. /login – POST-запрос на авторизацию пользователя в системе по уникальному логину.
2. /papers – GET-запрос на получение списка научных работ, загруженных пользователем.
3. /papers/{id} – GET-запрос на получение заданной научный работы, загруженной пользователем.
4. /upload-paper – POST-запрос на загрузку файла с научной работой в систему.
5. /papers/{id}/checks-results – GET-запрос на получение результатов анализа стиля определенной работы авторизованного пользователя.
6. /delete-paper – POST-запрос на удаление заданного файла с научной работой авторизированного пользователя.
7. /charts/frequency – GET-запрос на получение данных для построения графиков для визуализации частоты использования заданного параметра в работах по заданной тематике.
8. /prof-paper-results – POST-запрос, запускающий проверку эталонных научных работ по заданной тематике.
9. /checks-config – POST-запрос на добавление новых стилевых правил.

2.6. Проектирование графического интерфейса

В рамках разработки системы для анализа стиля синтаксических конструкций научных текстов на английском языке необходимо и достаточно разработать интерфейс, содержащий элементы:

1. Форма логина – текстовое поле для ввода логина и кнопка с функциональностью «Войти в систему».

2. Кнопка для выхода из системы и смена логина.
3. Элемент (например, кнопка) для открытия файловой системы для выбора загружаемого файла.
4. Список загруженных пользователем работ с информацией: о названии файла, проценте близости к образцу и дате проверки.
5. Элемент (кнопка) для запуска проверки загруженного документа.
6. Элемент с подробной информацией о проверке документа.

Макет спроектированного интерфейса главной страницы приложения представлено на рисунке 2.3.

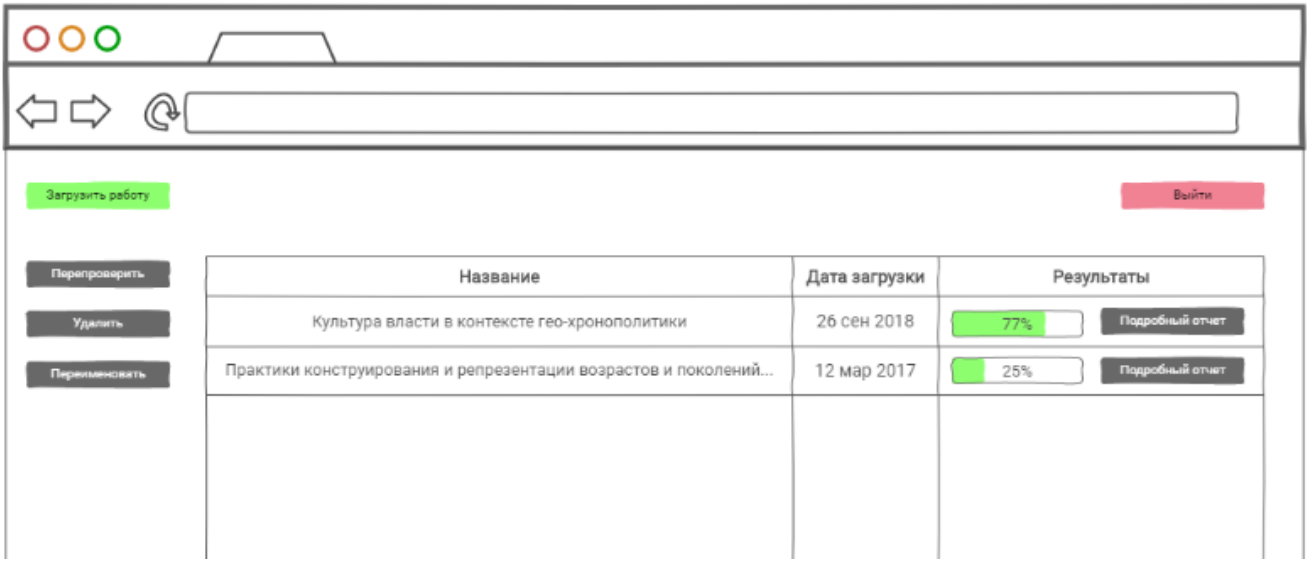

Рисунок 2.3. Макет главной страницы

Глава 3. Реализация системы для анализа синтаксического стиля научных текстов на английском языке

В главе описаны процесс программной реализации системы для анализа стиля синтаксических конструкций научных текстов на английском языке. Отдельно отражены разработка клиентской и серверной частей приложения, а также тестирование разработанной системы.

3.1. Разработка серверной части приложения

Серверная часть приложения представлена несколькими компонентами, а именно:

1. Docker-контейнер с приложением для взаимодействия с SyntaxNet фреймворком.
2. Spring Boot приложение, предоставляющее API.
3. Приложение базы данных.

Для развертывания приложения, обеспечивающего взаимодействие с SyntaxNet, используется система контейнеризации Docker – открытая платформа для разработчиков и системных администраторов для сборки, отправки и развертывания распределенных приложений [15]. Приложение, взаимодействующее с фреймворком SyntaxNet, написано на языке Python. Данное приложение предоставляет REST сервис для осуществления синтаксического разбора предложений. В качестве результата синтаксического разбора заданного предложения выдается ответ в одном из форматов: в виде синтаксического дерева или в виде массива. Формат JSON для отправки POST запроса на осуществление синтаксического разбора представлен ниже:

```
{
  “strings”: [”string”],
  “tree”: Boolean
}
```

Spring Boot приложение написано на языке программирования Java. Данный компонент реализует всю основную бизнес-логику приложения (проверку синтаксического стиля документов, авторизацию пользователей), взаимодействует с базой данных и с приложением SyntaxNet, предоставляет REST API для клиентского приложения.

Описание разработанных ресурсов представлено в таблице 3.1.

Таблица 3.1. Описание ресурсов

URL	***HTTP-метод***	***Описание***
api/v1/upload-paper	POST	Позволяет передать файл на сервер. Параметры: file – передаваемый файл. В ответ сервер отдает UUID загруженного файла.
api/v1/papers	GET	Позволяет получить полный список загруженных файлов для заданного пользователя. Параметры: userId – login пользователя.
api/v1/papers /{id}	GET	Позволяет получить заданный файл для заданного пользователя. Параметры: id – UUID файла, userId – login пользователя.
api/v1/papers /{id}/checks-results	GET	Позволяет получить результаты проверки для заданного файла заданного пользователя. Параметры: id – UUID файла, userId – login пользователя.
api/v1/login	POST	Авторизует пользователя в системе. Параметры: login – уникальное имя пользователя в системе.
api/v1/me	GET	Позволяет получить данные текущего авторизованного пользователя. Параметры: текущая сессия.
api/v1/delete-paper	POST	Удаляет загруженный файл пользователя. Параметры: fileId – UUID удаляемого файла.
api/v1/charts /frequency	GET	Позволяет получить данные для построения статистических графиков Параметры: label – строковая константа, обозначающая предметную область анализируемых работ. Пример: label=politology.
api/v1/checks-config	GET	Позволяет получить набор эталонных правил стиля синтаксических конструкций текстов.
api/v1/checks-config	POST	Позволяет задать набор эталонных правил стиля синтаксических конструкций текстов. Параметры: JSON, описывающий правила.
api/v1/upload-prof-paper	POST	Позволяет загрузить эталонный документ по заданной тематике. Параметры: file – загружаемый файл, label - строковая константа, обозначающая предметную область анализируемых работ.
api/v1/prof-papers	GET	Позволяет получить список всех работ по заданной тематике. Параметры: label - строковая константа, обозначающая предметную область анализируемых работ.
api/v1/prof-paper-results	POST	Позволяет запустить проверку эталонных работ по заданной тематике и получить результаты. Параметры: label - строковая константа, обозначающая предметную область анализируемых работ.

Реализовано также и разграничение доступа к ресурсам. В системе реализовано два уровня доступа: пользовательский и администраторский. Уровень администратора позволяет использовать ресурсы, взаимодействующие с настройками работы системы, а именно:

1. Настройка конфигурации правил для проверки синтаксического стиля.
2. Загрузка эталонных работ.
3. Запуск сбора метрик по корпусам эталонных работ.
4. Запрос на получение данных проверки эталонных работ.

Проверка стиля синтаксических конструкций загруженных текстов по определенной тематике производится следующим образом:

1. С помощью SyntaxNet строится синтаксическое дерево каждого из предложений проверяемого текста.
2. Параллельно производится подсчет статистики использования заданных синтаксических конструкций.
3. Полученные значения по каждому маркеру сравниваются с эталонными диапазонами, рассчитанными по корпусу профессиональных работ (по каждой тематике отдельно).
4. Формируется текстовый отчет о соответствии работы эталонным значениям.

В качестве базы данных выбрано нереляционное хранилище типа «ключ-значение» Redis (remote dictionary server). Выбор обусловлен простотой работы с базой данных, высокой скоростью обработки запросов на доступ к данным, а также наличием возможности относительно простого масштабирования и высокой отказоустойчивости.

Программный код серверной части приложения представлен в Приложении А.

3.2. Сбор статистики и проверка гипотез

В ходе разработки был произведен подсчет частотности употребления различных синтаксических конструкций. Возможные варианты конструкций формировались с помощью дерева: считался узел и его предок. Производился подсчет синтаксических конструкций, состоящих из одного, двух и более синтаксических единиц текста. Примеры таких маркеров представлены в таблице 3.2. Всего была подсчитана частота употребления более чем 900 различных синтаксических конструкций. Также к этим метрикам были добавлены следующие: подсчет количества вопросительных предложений и определение синтаксической сложности предложений на основе подсчета глубины (высоты) синтаксического дерева.

Таблица 3.2. Описание маркеров

Маркер	***Описание маркера***
advcl:advmod	advcl (adverbial clause modifier) – условие, которое изменяет глагол или другой предикат (прилагательное и т.д.). Выступает в роли модификатора. Указывает на придаточные времени, следствия, цели, условные предложения. advmod (adverbial modifier) – наречие или наречная фраза, которая служит для изменения предиката или слова-модификатора.
parataxis:compound	parataxis (parataxis) – отношение между словом (часто основным предикатом предложения) и другими элементами предложения таким как: скобка или двоеточие/точка с запятой, размещенными рядом без какой-либо явной связи с корневым словом. compound (compound) – выражает «комбинированное» отношения. Может выступать для связки существительных или для связки последовательных глаголов.
ccomp:xcomp	ccomp (clausal complement) – указывает на зависимое предложение, которое является основным аргументом. То есть он функционирует как объект глагола или прилагательное. xcomp (open clausal complement) – открытое дополнение глагола или прилагательного, то есть дополнение без его собственного субъекта.
conj:obl	conj (conjunct) – координирующая связь между двумя элементами, выраженная, например, с помощью «and», «or». obl (oblique nominal) – используется для выражения неосновного аргумента или дополнения, выраженного через существительное или местоимение.
conj:obj	conj (conjunct) – координирующая связь между двумя элементами, выраженная, например, с помощью «and», «or». obj (object) – объект; является вторым по важности аргументом глагола после подлежащего. Как правило, это существительное, обозначающее сущность, на которую оказывают воздействие.
ccomp	ccomp (clausal complement) – указывает на зависимое предложение, которое является основным аргументом. То есть он функционирует как объект глагола или прилагательное.

Маркер	***Описание маркера***
conj:amod	conj (conjunct) – координирующая связь между двумя элементами, выраженная, например, с помощью «and», «or». amod (adjectival modifier) – любая прилагательная фраза, которая служит для изменения значения существительного.

Кроме описанных выше маркеров, производился подсчет статистики по некоторым маркерам, выделенным профессиональными лингвистами. В частности, производился поиск следующих конструкций:

1. Длина предложений (выступает как индикатор синтаксической сложности предложений).
2. Порядок слов в предложении:
   2.1. Использование прилагательных перед существительными, которых они описывают.
   2.2. Главные субъект в начале предложения.
3. Th-конструкции (this, that, these those).
4. Wh-конструкции (who, where, what, why. when).
5. Вопросительные предложения: прямые и косвенные.

Как было сказано выше в процессе разработки было поставлено множество гипотез об использовании различных синтаксических конструкций в научных текстах на английском языке. Очевидно, что не все из поставленных гипотез были бы подтверждены. Для первоначального отсеивания неподходящих гипотез использовался коэффициент вариации. Коэффициент вариации показывает насколько выборка выровнена или по-другому: насколько велик разброс значений случайной величины. И таким образом, несмотря на то, что в процессе исследования ставилось много гипотез, данный коэффициент позволял отсеивать самые неудачные из них.

После расчета коэффициента вариации для каждой из эмпирически собранных выборки данных, были определены только самые удачные гипотезы. Важно отметить, что для каждой из анализируемых тематик был определен свой набор значимых метрик. Как оказалось в ходе разработки, даже работы, принадлежащие к одной области научных знаний (например, к социальным наукам), но к разным дисциплинам (например, экономика и политология), имеют разные не только значения эталонных диапазонов по собираемым метрикам, но и сам набор метрик отличается.

Для определения наиболее вероятных значений параметров по каждой из выбранных статистических характеристик, можно воспользоваться методом квантилей. С помощью двустороннего квантиля уровня α можно задать интервал, в который анализируемая случайная величина попадает с заданной вероятностью. Для примера рассмотрим статистику, собранную для случайной величины «синтаксическая сложность предложения». Данная случайная величина показывает среднюю сложность предложений, используемую в тексте. Синтаксическая сложность предложения является числовым выражением глубины (высоты) синтаксического дерева, полученного после разбора предложения. Вариационный ряд для непрерывной случайной величины «средняя синтаксическая сложность» представлен в таблице 3.3. Ширина интервала была рассчитана следующим образом.

Формула для нахождения количества интервалов (N – размер выборки):

$$M = \sqrt{N} = \sqrt{134} \approx 12$$

Формула для нахождения ширины интервала (y_{min} и y_{max} – минимальное и максимальное значения случайной величины):

$$\Delta y = \frac{(y_{max} - y_{min})}{M} = \frac{(3{,}17 - 2{,}24)}{12} = 0{,}078$$

Таблица 3.3. Интервальный вариационный ряд

№	*Границы интервала*	*Середина интервала*	*Частота наблюдения*	*Накопленная частота*	*Относительная частота*
1	[2,240; 2,318)	2,279	1	1	0,007463
2	[2,318; 2,396)	2,357	5	6	0,037313
3	[2,396; 2,474)	2,435	13	19	0,097015
4	[2,474; 2,552)	2,513	45	64	0,335821
5	[2,552; 2,630)	2,591	33	97	0,246269
6	[2,630; 2,708)	2,669	19	116	0,141791
7	[2,708; 2,786)	2,747	13	129	0,097015
8	[2,786; 2,864)	2,825	1	130	0,007463
9	[2,864; 2,942)	2,903	2	132	0,014925
10	[2,942; 3,020)	2,981	1	133	0,007463
11	[3,020; 3,098)	3,059	0	133	0
12	[3,098; 3,17]	3,134	1	134	0,007463
	Сумма		***134***		***1***

Визуализация относительных частот попадания случайной величины в заданный интервал в виде гистограммы выборки представлено на рисунке 3.1.

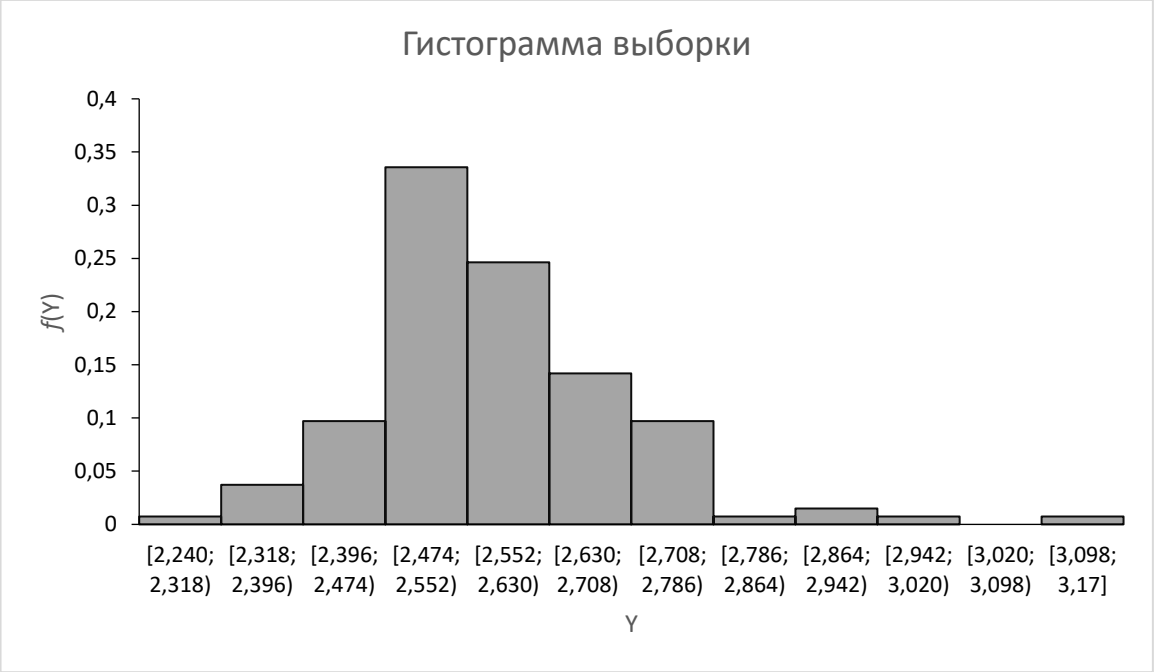

Рисунок 3.1. Гистограмма выборки

Далее необходимо построить функцию распределения. Диаграмма накопленных частот является аналогом (эмпирическим) интегральной функции распределения. Построенная диаграмма представлена на рисунке 3.2.

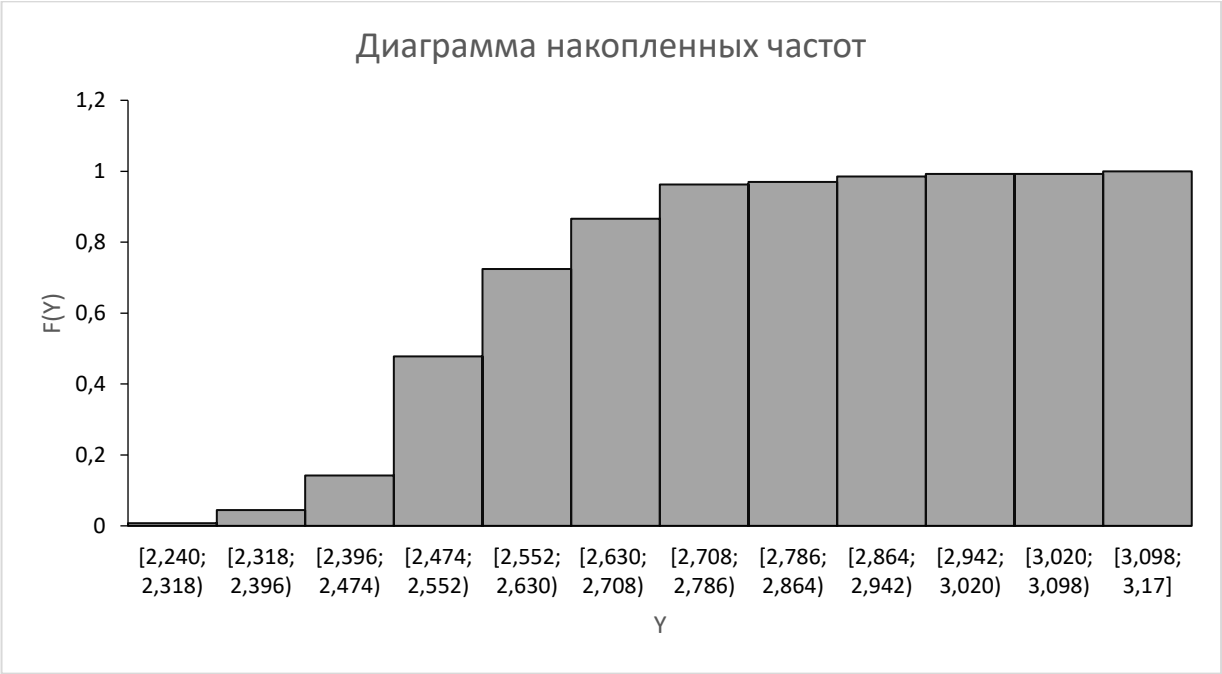

Рисунок 3.2. Диаграмма накопленных частот

Рассчитаем квантили $Q_{1/20}$ и $Q_{19/20}$. Интерпретацию полученных данных приведем после расчетов. Квантили для интервального ряда будут рассчитываться следующим образом:

$$Q_{1/20} = 2{,}396 + 0{,}078\frac{0{,}05 * 134 - 6}{13} = 2{,}4002,$$

$$Q_{19/20} = 2{,}708 + 0{,}078\frac{0{,}95 * 134 - 116}{13} = 2{,}7758.$$

Полученные результаты означают, что 5% выборки имеют среднюю синтаксическую сложность предложений менее 2,4002 и 5% выборки со значением средней сложности более 2,7758.

Посчитанные результаты можно отобразить на гистограмме распределения случайной величины (рисунок 3.3).

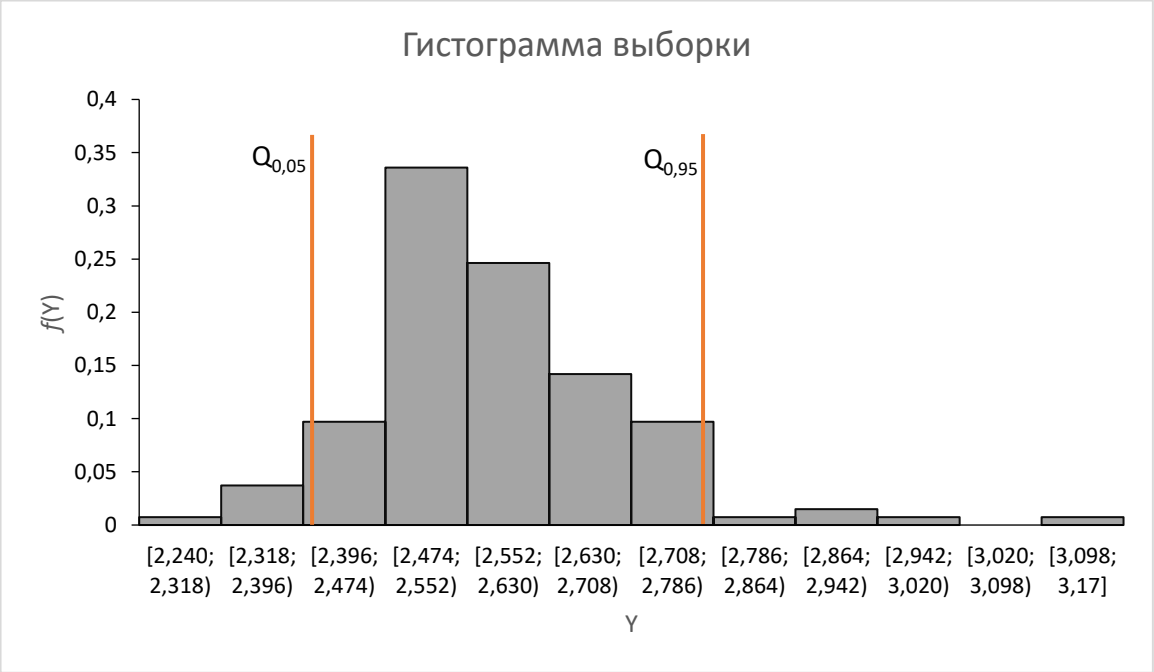

Рисунок 3.3. Квантили на гистограмме

Таким образом, на основе собранных эмпирических данных были определены допустимые диапазоны по каждому маркеру.

3.3. Разработка клиентской части приложения

Клиентское приложение выполнено в соответствии со спроектированной архитектурой и является тонким клиентом, который представлен браузером. Клиентская часть приложения, запускаемая в браузере, разработана с использованием Vue.js – JavaScript фрейморка с открытым кодом, который используется для построения пользовательского интерфейса. За счет низкого порога вхождения Vue позволяет быстро создавать универсальные и производительные приложения, которые достаточно просто поддерживать и тестировать. С помощью фреймворка Vue можно разделить веб-страницу на переиспользуемые компоненты (что ускоряет разработку), каждый из которых будет иметь свой собственный HTML код, CCS и JavaScript, необходимый для рендеринга этого компонента.

Архитектура клиентского приложения выполнена в соответствии с архитектурным паттерном MVVM (Model, View, ViewModel). Модель (Model) содержит данные и бизнес-логику, Представление (View) отвечает за пользовательский интерфейс, а Модель Представления обеспечивает связь между моделью и представлением через привязку данных (data binding).

Разработанный пользовательский интерфейс представлен на рисунке 3.4.

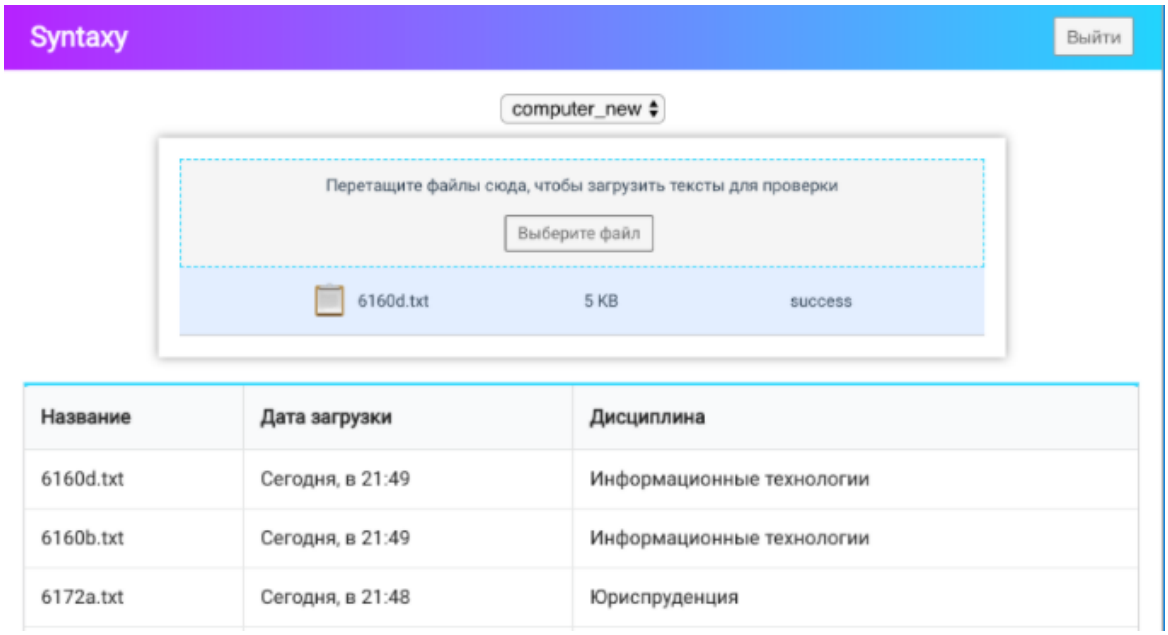

Рисунок 3.4. Главное окно личного кабинета

В процессе реализации клиентской части приложения были разработаны интерфейсы для рядового пользователя и для администратора системы. Администратор имеет следующие дополнительные возможности:

1. Загрузка новых эталонных документов по заданной тематике.
2. Запуск проверки всего корпуса эталонных документов по заданной тематике.
3. Визуализация в виде графиков собранной статистики для работ по заданной тематике.
4. Просмотр списка правил, отражающих эталонный синтаксических стиль работ по заданной тематике.
5. Редактирование эталонных диапазонов, используемых в правилах при проверке синтаксического стиля работ, добавление новых правил.

Интерфейс для загрузки эталонных работ по заданной тематике представлен на рисунке 3.5.

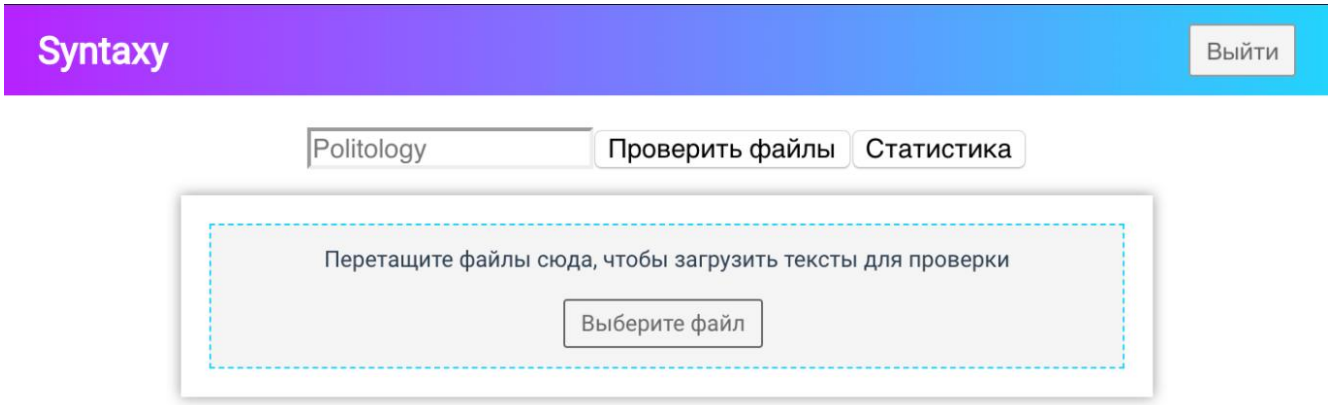

Рисунок 3.5. Загрузка эталонных работ

Редактирование стилевых правил в кабинете администратора представлено на рисунке 3.6.

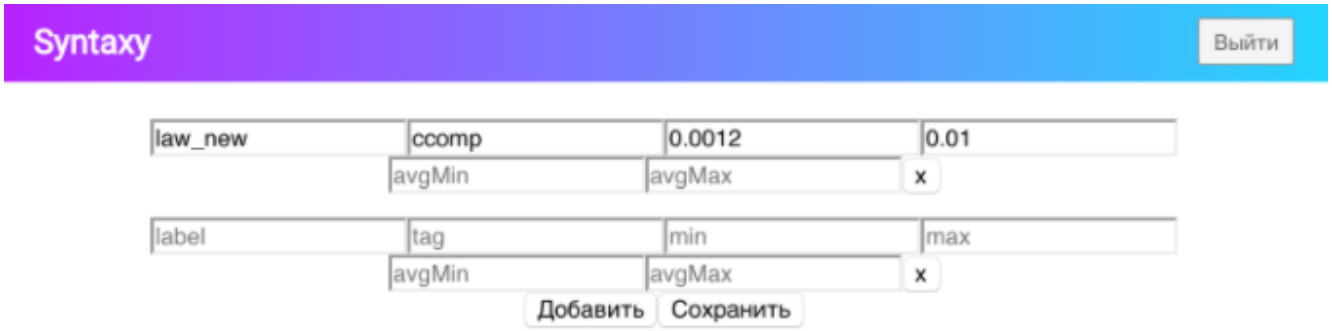

Рисунок 3.6. Кабинет администратора (редактирование правил).

Программный код клиентской части приложения представлен в Приложении Б.

3.4. Тестирование разработанной системы

После разработки клиентской и серверной частей приложения для каждой из частей были написаны автоматизированные тесты.

Для тестирования серверного приложения были написаны Unit тесты. Были использованы следующие библиотеки для тестирования: JUnit, Spring Test. Одним из ключевых преимуществ использования фремворка Spring является внедрение зависимостей (dependency injection), что значительно упрощает тестирование. Это происходит за счет инверсии зависимостей и отказа от явного создания объектов, реализующих бизнес-логику.

Пример теста, проверяющего работу конечной точки (ресурса) представлен ниже (импорты опущены для увеличения читаемости). Данный тест проверяет работоспособность ресурса, отвечающего за авторизацию пользователей. Полный набор Unit-тестов для проверки работы серверной части представлен в Приложении В.

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
@AutoConfigureWebTestClient
public class LoginResourceTests {

        @Autowired
        private WebTestClient webClient;

        @Test
        public void loginTest() {
            webClient.post()
                     .uri("/api/login")
                     .body(BodyInserts.fromObject(new LoginDto()
                                                  .setUserName("User-test")))
                     .exchange()
                     .expectStatus().isOk()
                     .expectBody().json("{\"status\": \"ok\"}");
        }
}
```

Для тестирования клиентского приложения были написаны Unit тесты с использованием фреймворка Jasmine. Ниже приведен пример unit-теста, написанного для клиентского приложения. Представленный тест проверяет наличие необходимых функций в компоненте, а также валидирует установку поля login после примонтирования экземпляра. Полный набор тестов представлен в Приложении Г.

```
import Vue from 'vue'
import SyntaxyMain from 'SyntaxyMain.vue'



describe('SyntaxyMain', () => {
  it('has an upload file function', () => {
    expect(typeof SyntaxyMain.uploadFile).toBe('function')
  })

  it('sets the correct default user login', () => {
    expect(typeof SyntaxyMain.data).toBe('function')
    const defaultData = SyntaxyMain.data()
    expect(defaultData.login).toBe('User1')
  })

  it('correctly sets the user login when created', () => {
    const vm = new Vue(SyntaxyMain).$mount()
    expect(vm.login).toBe('User-test')
  })
})
```

Заключение

В ходе данной работы была разработана система для автоматизированной проверки стиля синтаксических конструкций научных текстов на английском языке. В процессе разработки были решены следующие задачи:

1. Проведен аналитический обзор существующих программных решений для автоматизированного анализа стиля английских текстов, выявлена необходимость разработки системы для анализа стиля синтаксических конструкций научных текстов на английском языке.
2. Описаны существующие средства (фреймворки) для осуществления синтаксического разбора предложения, выбран наиболее подходящий вариант для реализации системы (клиент-серверное веб-приложение).
3. Поставлены и проверены гипотезы о значимости некоторых синтаксических конструкций при анализе синтаксического стиля текста.
4. Произведено моделирование разрабатываемой системы.
5. Разработано необходимое API для работы системы.
6. Разработана серверная и клиентская части веб-приложения, позволяющего производить анализ стиля научных текстов на английском языке с точки зрения используемых синтаксических конструкций.
7. Разработана система автоматизированных unit-тестов и произведено тестирование разработанной системы.

Таким образом, была разработана система, которая позволяет производить анализ стиля синтаксических конструкций научных текстов на английском языке. Планируется использовать данную систему для обучения студентов хорошему синтаксическому стилю научных текстов на английском языке. Также планируется произвести работы по расширению функциональности разработанного веб-приложения. В частности, планируется добавить визуализацию результатов проверки работы в графическом виде для всех пользователей (сейчас такая функциональность доступна только администратору), а также улучшить систему стилевых рекомендаций путем.

Библиографический список

1. Payette J., Hirst G. An Intelligent Computer-Assistant for Stylistic Instruction. Springer. 1992. pp. 87-102.
2. DocBridge Delta // Visual Document Comparison & Quality Control. URL: https://www.compart.com/en/docbridge-delta/ (дата обращения: 07.05.2018).
3. After the Deadline. Spell, Style, and Grammar Checker. URL: www.afterthedeadline.com/ (дата обращения: 07.05.2018).
4. Jurafsky D., Martin J. H. Speech and language processing. 2017. p. 499.
5. Ellis, J.M. Linguistics, Literature, and the Concept of Style. Word, Vol. 26. 1970. pp. 65-78.
6. Paper Rater. URL: https://www.paperrater.com/ (дата обращения: 07.05.2018).
7. Online Text Correction. URL: www.onlinecorrection.com/ (дата обращения: 07.05.2018).
8. LanguageTool. Style and Grammar Checker. URL: https://languagetool.org/ (дата обращения: 07.05.2018).
9. Scribens. Free, Powerful English Grammar Checker. URL: https://www.scribens.com/ (дата обращения: 07.05.2018).
10. Корпусы – научно-исследовательская группа «Разработка программного обеспечения для проведения корпусных исследований английского языка» // НИУ ВШЭ – Пермь. URL: https://perm.hse.ru/bi/sfcr/corpora (дата обращения: 07.05.2018).
11. Lynch T., Anderson K. Grammar for Academic Writing. English Language Teaching Centre University of Edinburgh. 2013. 90 с.
12. Siepmann D., Gallagher J.D., Hannay M., Mackenzie J.L. Writing in English: A Guide for Advanced Learners. 2011. 479 с.
13. Ивченко Г., Медведев Ю. Математическая статистика. Либроком. 2014. 353 с.
14. Marneffe M-C., Manning C.D. Stanford typed dependencies manual. 2008. 20 с.
15. Docker - Build, Ship, and Run Any App, Anywhere. URL: https://www.docker.com/ (дата обращения: 07.05.2018).

Приложения

Приложение А. Код серверной части приложения

```
package com.syntaxy.syntaxen;

import com.syntaxy.syntaxen.domain.SentenceParsingArrayResult;
import com.syntaxy.syntaxen.domain.ParseWordTreeNode;
import com.syntaxy.syntaxen.service.SyntaxNetService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.Arrays;
import java.util.List;

@SpringBootApplication
public class SyntaxEnApplication {

	@Autowired
	SyntaxNetService syntaxNetService;

	public static void main(String[] args) {
		SpringApplication.run(SyntaxEnApplication.class, args);
	}

	public void run(String... args) throws Exception {
		List<SentenceParsingArrayResult> parsingArrayFrom = 
syntaxNetService.getParsingArrayFrom(Arrays.asList(
				"The connections between Irish politicians and the 
private sector are the subject of constant speculation, particularly in the post 
Celtic Tiger period.",
				"The connections between Irish politicians and the 
private sector are the subject of constant speculation, particularly in the post 
Celtic Tiger period.")
		);
```

```
                List<ParseWordTreeNode>             parsingTreeFrom             =
syntaxNetService.getParsingTreeFrom(Arrays.asList(
                            "The connections between Irish politicians and the
private sector are the subject of constant speculation, particularly in the post
Celtic Tiger period.",
                            "The connections between Irish politicians and the
private sector are the subject of constant speculation, particularly in the post
Celtic Tiger period.")
                );
                int a = 0;
            }
        }

        package com.syntaxy.syntaxen.web.rest;

        import org.springframework.web.bind.annotation.RequestMapping;
        import org.springframework.web.bind.annotation.RequestMethod;
        import org.springframework.web.bind.annotation.RequestParam;
        import org.springframework.web.bind.annotation.RestController;
        import org.springframework.web.multipart.MultipartFile;

        @RestController
        public class FileResource {
            @RequestMapping(value = "/upload-file", method = RequestMethod.POST)
            public void handleFileUpload(@RequestParam("file") MultipartFile file) {
                int a = 0;
            }
        }
        package com.syntaxy.syntaxen.service;

        import com.syntaxy.syntaxen.domain.SentenceParsingArrayResult;
        import com.syntaxy.syntaxen.domain.ParseWordTreeNode;
        import java.util.List;

        public interface SyntaxNetService {
```

```
        List<SentenceParsingArrayResult>        getParsingArrayFrom(List<String>
sentences);
        List<ParseWordTreeNode> getParsingTreeFrom(List<String> sentences);
    }
    package com.syntaxy.syntaxen.service.impl;

    import com.syntaxy.syntaxen.domain.SentenceParsingArrayResult;
    import com.syntaxy.syntaxen.domain.ParseWordTreeNode;
    import com.syntaxy.syntaxen.service.SyntaxNetService;
    import org.springframework.core.ParameterizedTypeReference;
    import org.springframework.http.HttpEntity;
    import org.springframework.http.HttpMethod;
    import org.springframework.http.ResponseEntity;
    import org.springframework.stereotype.Service;
    import org.springframework.web.client.RestTemplate;
    import java.util.List;

    @Service
    public class SyntaxNetServiceImpl implements SyntaxNetService {
        @Override
        public List<SentenceParsingArrayResult> getParsingArrayFrom(List<String>
sentences) {
            RestTemplate template = new RestTemplate();
            HttpEntity requestEntity = new HttpEntity<>(
                    new RequestPayload().setStrings(sentences).setTree(false));

            ResponseEntity<List<SentenceParsingArrayResult>> parsingResponse =
                    template.exchange("http://localhost:9000/api/v1/query",
HttpMethod.POST,
                            requestEntity,                                    new
ParameterizedTypeReference<List<SentenceParsingArrayResult>>() {
            });
            List<SentenceParsingArrayResult> result = parsingResponse.getBody();
            return result;
        }

        @Override
```

```
        public List<ParseWordTreeNode> getParsingTreeFrom(List<String>
sentences) {
            RestTemplate template = new RestTemplate();
            HttpEntity requestEntity = new HttpEntity<>(
                    new RequestPayload().setStrings(sentences).setTree(true));

            ResponseEntity<List<ParseWordTreeNode>> parsingResponse =
                    template.exchange("http://localhost:9000/api/v1/query",
HttpMethod.POST,
                            requestEntity,                                  new
ParameterizedTypeReference<List<ParseWordTreeNode>>() {
                            });
            List<ParseWordTreeNode> result = parsingResponse.getBody();
            return result;
        }

        static class RequestPayload {
            private List<String> strings;
            private boolean tree;

            public List<String> getStrings() {
                return strings;
            }

            public RequestPayload setStrings(List<String> strings) {
                this.strings = strings;
                return this;
            }
            public boolean getTree() {
                return tree;
            }
            public RequestPayload setTree(boolean tree) {
                this.tree = tree;
                return this;
            }
        }
    }
```

```
import java.io.IOException;
import java.text.MessageFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class TagsFrequencyCheck {
    public static final String SUBJECT_TAG = "nsubj";
    public static final String CONJ_TAG = "conj";
    public static final String COMPOUND_TAG = "compound"; // phone<-book //
put->up
    public static final String ACL_TAG = "acl";
    public static final String ADVCL_TAG = "advcl";
    public static final String ADVMOD_TAG = "advmod";
    public static final String AMOD_TAG = "amod";
    public static final String APPOS_TAG = "appos";
    public static final String AUX_TAG = "aux";
    public static final String CASE_TAG = "case";
    public static final String CC_TAG = "cc";
    public static final String CCOMP_TAG = "ccomp";
    public static final String CLF_TAG = "clf";
    public static final String COP_TAG = "cop";
    public static final String CSUBJ_TAG = "csubj";
    public static final String DET_TAG = "det";
    public static final String DISCOURSE_TAG = "discourse";
    public static final String DISLOCATED_TAG = "dislocated";
    public static final String EXPL_TAG = "expl";
    public static final String FIXED_TAG = "fixed";
    public static final String FLAT_TAG = "flat";
    public static final String GOESWITH_TAG = "goeswith";
    public static final String IOBJ_TAG = "iobj";
    public static final String LIST_TAG = "list";
    public static final String MARK_TAG = "mark";
    public static final String NMOD_TAG = "nmod";
    public static final String NUMMOD_TAG = "nummod";
```

```
        public static final String OBJ_TAG = "obj";
        public static final String OBL_TAG = "obl";
        public static final String ORPHAN_TAG = "orphan";
        public static final String PARARAXIS_TAG = "parataxis";
        public static final String PUNCT_TAG = "punct";
        public static final String REPARANDUM_TAG = "reparandum";
        public static final String VOCATIVE_TAG = "vocative";
        public static final String XCOMP_TAG = "xcomp";

        private List<ParseWordTreeNode> text;
        private String paperId;
        private int totalWordsCount;

        private Map<String, Integer> statistics = new HashMap<>();
        private List<Integer> maxSentenceDepth = new ArrayList<>();

        public TagsFrequencyCheck(List<ParseWordTreeNode> text, String paperId)
{
            this.text = text;
            this.paperId = paperId;

            for (ParseWordTreeNode sentenceRoot : text) {
                TraverseContext context = new TraverseContext();
                context.currentDepth = 1;
                context.parent = sentenceRoot;
                countSubjects(sentenceRoot, context);
                maxSentenceDepth.add(context.maxDepth);
            }
        }

        public List<TagFrequencyCheck> makeAllAvailableTags() {
            List<TagFrequencyCheck> result = new ArrayList<>();
            statistics.forEach((tag, stats) -> {
                TagFrequencyCheck c = makeTagCheck(tag);
                result.add(c);
            });
            return result;
```

```
        }

        public TagFrequencyCheck makeTagCheck(String tag) {
            TagFrequencyCheck check = new TagFrequencyCheck();
            Integer totalCount = statistics.getOrDefault(tag, 0);
            check.setPaperId(paperId);
            check.setTotalCount(totalCount);
            check.setAverage((double) totalCount / totalWordsCount);
            check.setTagName(tag);
            return check;
        }

        private void countSubjects(ParseWordTreeNode node, TraverseContext
context) {
            totalWordsCount += 1;
            countTag(node.getDep());
            countTag(node.getPosTag());
            String parentDep;
            if (context.parent == null) {
                parentDep = "-";
            } else {
                parentDep = context.parent.getDep();
            }
            String parentPlusCurrentDep = parentDep + ":" + node.getDep();
            countTag(parentPlusCurrentDep);

            switch (node.getWord().toLowerCase()) {
                case "that":
                case "these":
                case "those":
                case "this":
                    countTag("ThatThose");
                    break;
            }

            if (node.getContains() == null) {
```

```
                    context.maxDepth         =         Math.max(context.currentDepth,
context.maxDepth);
                    context.currentDepth -= 1;
                    return;
                }

                ParseWordTreeNode oldParent = context.parent;

                context.currentDepth += 1;
                context.parent = node;
                for (ParseWordTreeNode child : node.getContains()) {
                    countSubjects(child, context);
                }

                context.parent = oldParent;
            }

            private void countTag(String dep) {
                if (dep != null) {
                    statistics.put(dep, statistics.getOrDefault(dep, 0) + 1);
                }
            }

            public List<ParseWordTreeNode> getText() {
                return text;
            }

            public double getAvgMaxDepth() {
                return             maxSentenceDepth.stream().mapToInt(x             ->
x).average().orElse(0);
            }

            public Result performStudentPaperAnalysis(String configJson) {
                Config config = Config.fromJson(configJson);
                List<Message> messages = new ArrayList<>();
                for (IdealValue idealValue : config.idealValues) {
                    Integer stats = statistics.get(idealValue.tag);
```

```
        if (stats == null) {
            continue;
        }
        if (idealValue.max != null && stats > idealValue.max) {
            messages.add(new Message(
                    idealValue.tag,
                    MessageFormat.format(
                            "max value is {0}, your value is {1}",
                            idealValue.max, stats)
            ));
        }
        if (idealValue.min != null && stats < idealValue.min) {
            messages.add(new Message(
                    idealValue.tag,
                    MessageFormat.format(
                            "min value is {0}, your value is {1}",
                            idealValue.min, stats)
            ));
        }
        double avg = stats.doubleValue() / totalWordsCount;
        if (idealValue.avgMax != null && avg > idealValue.avgMax) {
            messages.add(new Message(
                    idealValue.tag,
                    MessageFormat.format(
                            "avg max value is {0}, your value is {1}",
                            idealValue.avgMax, avg)
            ));
        }
        if (idealValue.avgMin != null && avg < idealValue.avgMin) {
            messages.add(new Message(
                    idealValue.tag,
                    MessageFormat.format(
                            "avg min value is {0}, your value is {1}",
                            idealValue.avgMin, avg)
            ));
        }
    }
```

```
        return new Result(messages);
    }

    private static class TraverseContext {
        private int currentDepth;
        private int maxDepth;
        private ParseWordTreeNode parent;
    }

    public static class Result {
        private final List<Message> messages;

        public Result(List<Message> messages) {
            this.messages = messages;
        }

        public List<Message> getMessages() {
            return messages;
        }
    }

    public static class Message {
        private final String tag;
        private final String message;

        public Message(String tag, String message) {
            this.tag = tag;
            this.message = message;
        }

        public String getTag() {
            return tag;
        }

        public String getMessage() {
            return message;
```

```
    }
}

public static class Config {
    private List<IdealValue> idealValues = new ArrayList<>();

    public static Config fromJson(String configJson) {
        if (configJson == null) {
            configJson = "{}";
        }
        ObjectMapper mapper = new ObjectMapper();
        Config config;
        try {
            config = mapper.readValue(configJson, Config.class);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        return config;
    }

    public String toJson() {
        ObjectMapper mapper = new ObjectMapper();
        try {
            return mapper.writeValueAsString(this);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    public List<IdealValue> getIdealValues() {
        return idealValues;
    }

    public Config setIdealValues(List<IdealValue> idealValues) {
        this.idealValues = idealValues;
        return this;
    }
```

```
}

private static class IdealValue {
    private String label;
    private String tag;
    private Double min;
    private Double max;
    private Double avgMin;
    private Double avgMax;

    public String getLabel() {
        return label;
    }

    public IdealValue setLabel(String label) {
        this.label = label;
        return this;
    }

    public String getTag() {
        return tag;
    }

    public IdealValue setTag(String tag) {
        this.tag = tag;
        return this;
    }

    public Double getMin() {
        return min;
    }

    public IdealValue setMin(Double min) {
        this.min = min;
        return this;
    }
```

```
        public Double getMax() {
            return max;
        }

        public IdealValue setMax(Double max) {
            this.max = max;
            return this;
        }

        public Double getAvgMin() {
            return avgMin;
        }

        public IdealValue setAvgMin(Double avgMin) {
            this.avgMin = avgMin;
            return this;
        }

        public Double getAvgMax() {
            return avgMax;
        }

        public IdealValue setAvgMax(Double avgMax) {
            this.avgMax = avgMax;
            return this;
        }
    }
}
```

```
<template>
    <div id="hello">
        <div id="toolbar">
            <div id="logo">
                <p id="logo-p">
                    <router-link to="SyntaxyMain">Syntaxy</router-link>
                </p>
            </div>
            <div id="top-menu">
                <button class="top-btn" v-on:click="logout">Выйти</button>
            </div>
        </div>
        <div id="admin-upload" v-if="admin">
            <p>
                <input placeholder="Politology" type="text" v-
model="labelText">
                <button v-on:click="adminCheck">Проверить файлы</button>
                <button v-on:click="showStats">Статистика</button>
            </p>
            <uploader ref="uploaderprof"
                      :options="optionsprof" class="uploader">
                <uploader-unsupport/>
                <uploader-drop>
                    <p>Перетащите файлы сюда, чтобы загрузить тексты для
проверки</p>
                    <uploader-btn>Выберите файл</uploader-btn>
                </uploader-drop>
                <uploader-list/>
            </uploader>
        </div>
        <div id="paper-uploader">
            <uploader ref="uploader"
                      :options="options" class="uploader">
                <uploader-unsupport/>
                <uploader-drop>
```

```
                        <p>Перетащите файлы сюда, чтобы загрузить тексты для
проверки</p>
                        <uploader-btn>Выберите файл</uploader-btn>
                    </uploader-drop>
                    <uploader-list/>
                </uploader>
            </div>
            <div>
                <vuetable ref="vuetable"
                          api-url="api/v1/papers"
                          :fields=fields
                          data-path="content"
                          initialPage=0
                          pagination-path="pagination"
                          @vuetable:row-clicked="rowClicked"
                          @vuetable:pagination-data="onPaginationData"

                />
                <vuetable-pagination ref="pagination"/>
            </div>
        </div>
    </template>

    <script>

        import Vuetable from 'vuetable-2/src/components/Vuetable'
        import           VuetablePagination           from           'vuetable-
2/src/components/VuetablePagination'
        import axios from 'axios';
        import moment from 'moment';

        export default {
            name: 'app',
            props: {
                msg: String
            },
            components: {
```

```
                Vuetable,
                VuetablePagination
            },
            data() {
                return {
                    currentUser: null,
                    optionsprof: {
                        //                          https://github.com/simple-
uploader/Uploader/tree/develop/samples/Node.js
                        target: '/api/v1/upload-prof-paper',
                        testChunks: false,
                        query: (file, chunck, isTest) => {
                            return {'label': this.labelText};
                        }
                    },
                    options: {
                        //                          https://github.com/simple-
uploader/Uploader/tree/develop/samples/Node.js
                        target: '/api/v1/upload-paper',
                        testChunks: false
                    },
                    attrs: {},
                    fields: [
                        {
                            name: 'name',
                            title: 'Название'
                        },
                        {
                            name: 'uploadDate',
                            title: 'Дата загрузки',
                            callback: 'formatDate'
                        }
                    ],
                    paperLabels: {
                        politology: 'politology',

                    },
```

```
                admin: false,
                labelText: "",
                adminFiles: null,
            }
        },
        watch: {
            labelText: function (val) {
                if (val) {
                    this.optionsprof.target      =      'api/v1/upload-prof-
paper?label=' + val;
                }
            },
        },
        methods: {
            transform: function(data) {
                const transformed = {};

                transformed.pagination = {
                    total: data.totalElements,
                    per_page: data.pageable.pageSize,
                    current_page: data.number + 1,
                    last_page: data.totalPages,
                    next_page_url: '/api/v1/papers?page=' + data.number + 1,
                    prev_page_url: '/api/v1/papers?page=' + data.number - 1,
                    from: data.from,
                    to: data.to
                };

                transformed.content = data.content;
                return transformed
            },
            onPaginationData (paginationData) {
                console.log(paginationData);
                this.$refs.pagination.setPaginationData(paginationData);
            },
            rowClicked: function(dataitem, event) {
```

```
                    this.$router.push({name:    'PaperDetail',    params:    {id:
dataitem.id}})
                },
                logout: function (event) {
                    event.preventDefault();
                    axios.post(`/api/v1/auth/logout`)
                        .then(response => {
                            this.$router.push({name: 'Login'});
                        });
                },
                getPaperResults: function (paperId) {
                    axios.post(`/api/v1/paper-results`, {
                        'paperId': paperId
                    }).then(response => {
                        console.log(response);
                    });
                },
                formatDate: function (value) {
                    moment.locale('ru');
                    return moment(value).calendar();
                },
                onFileChange(e) {
                    const files = e.target.files || e.dataTransfer.files;
                    if (!files.length) {
                        this.adminFiles = null;
                        return;
                    }
                    this.adminFiles = e.target.files;
                },
                sendAdminFile: function () {
                    console.log(this.adminFiles);
                    console.log(this.labelText);
                    if (this.adminFiles && this.labelText) {
                        const that = this;
                        for (const file of this.adminFiles) {
                            axios.post(`/api/v1/upload-prof-paper`, {
                                file: file,
```

```
                            label: that.labelText
                        }).then(response => {
                            console.log(response);
                        });
                    }
                }
            },
            adminCheck: function () {
                if (this.labelText) {
                    const that = this;
                    axios.post(`/api/v1/prof-paper-results`, {
                        label: that.labelText,
                        tag: 'nsubj'
                    }).then(response => {
                        console.log(response);
                    });
                }
            },
            showStats: function () {
                this.$router.push({name:    'Chart',    params:    {label:
this.labelText}});
            },
        },
        mounted: function () {
            axios.get('/api/v1/users/me')
                .then(response => {
                    this.currentUser = response.data;
                    this.admin = this.currentUser.login === "admin";
                })
                .catch(e => {
                    const response = e.response;
                    if  (response  &&  response.data  &&  response.data  &&
response.data.key === 'user.not-authenticated') {
                        this.$router.push({name: 'Login'});
                    } else {
                        this.$notice({
                            type: 'error',
```

```
                            text: e,
                        });
                        this.$router.push({name: 'Login'});
                    }
                });
            this.$refs.vuetable.loadData();

            const uploader = this.$refs.uploader.uploader;
            const that = this;

            uploader.on('fileSuccess', function (rootFile, file, message) {
                that.$refs.vuetable.refresh();
                console.log(message);
                const paperId = JSON.parse(message)["id"]
                that.getPaperResults(paperId);
            });
        }
    }
</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->
<style scoped>
    .uploader {
        /*width: 880px;*/
        padding: 15px;
        font-size: 12px;
        box-shadow: 0 0 10px rgba(0, 0, 0, .4);
    }

    .uploader .uploader-btn {
        margin-right: 4px;
    }

    .uploader .uploader-list {
        max-height: 440px;
        overflow: auto;
        overflow-x: hidden;
```

```
    overflow-y: auto;
}

.uploader-drop {
    border: 1px dashed #21d4fd;
}

.ui.blue.table {
    border-top: .2em solid #21d4fd;
}

#paper-uploader {
    margin: 65px 12% 18px 12%;
}

#top-menu {
    margin: 10px 20px 10px 10px;
    display: flex;
    flex-direction: row;
    justify-content: center;
}

.top-btn {
    display: inline-block;
    position: relative;
    padding: 4px 8px;
    font-size: 90%;
    line-height: 1.4;
    color: #666;
    border: 1px solid #9a9a9a;
    cursor: pointer;
    border-radius: 2px;
    background: #f5f5f5;
    outline: none;
}

#toolbar {
```

```
        z-index: 100;
        /*background-color: #2962FF;*/
        /*background-color: #25292f;*/
        background: linear-gradient(to left, #21d4fd, #b721ff);
        display: flex;
        flex-direction: row;
        position: fixed;
        left: 0;
        top: 0;
        margin-bottom: 4px;
        width: 100%;
        justify-content: space-between;
    }

    #logo {
        margin: 10px 10px 10px 20px;
        display: flex;
        flex-direction: row;
        justify-content: center;
    }

    #logo-p {
        font-size: 20px;
        font-weight: bold;
        color: #fff;
    }

    #admin-upload {
        margin: 65px 12% 18px 12%;
    }

    #admin-btn {
        margin-top: 10px;
    }
</style>
```

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
@AutoConfigureWebTestClient
public class LoginResourceTests {

        @Autowired
        private WebTestClient webClient;

        @Test
        public void loginTest() {
            webClient.post()
                     .uri("/api/login")
                     .body(BodyInserts.fromObject(new LoginDto()
                                                  .setUserName("User-test")))
                     .exchange()
                     .expectStatus().isOk()
                     .expectBody().json("{\"status\": \"ok\"}");
        }
}
```

```
import Vue from 'vue'
import SyntaxyMain from 'SyntaxyMain.vue'



describe('SyntaxyMain', () => {
  it('has an upload file function', () => {
    expect(typeof SyntaxyMain.uploadFile).toBe('function')
  })

  it('sets the correct default user login', () => {
    expect(typeof SyntaxyMain.data).toBe('function')
    const defaultData = SyntaxyMain.data()
    expect(defaultData.login).toBe('User1')
  })

  it('correctly sets the user login when created', () => {
    const vm = new Vue(SyntaxyMain).$mount()
    expect(vm.login).toBe('User-test')
  })
})
```