МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИ «РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Институт компьютерных наук и технологического образования

Кафедра компьютерных технологий и электронного обучения

КУРСОВАЯ РАБОТА

СОЗДАНИЕ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ОБРАБОТКИ И АНАЛИЗА ТЕКСТА

Направление подготовки: «Информатика и вычислительная техника»

Руководитель:

Д.н.п., профессор,

Власова Е.З

Автор работы студент 2 курса 1 группы

Бережной Михаил Александрович

« ___» _____ 2022 г.

Санкт-Петербург

2017

ВВЕДЕНИЕ

В настоящее время исследовательская и аналитическая работа, связанная с обработкой текстовой информации, является достаточно не автоматизированной для обычного пользователя, не причастного к языкам программирования.

Причиной этого явления стало острая нехватка программ, приложений и прочих ресурсов для быстрой и качественной обработки текстов. Имеет место предположение о том, что данная проблема возникла в связи с развитием индустрии в несколько ином направлении. Следуя предположению автора данной курсовой работы, этим направлением стало высокоскоростное развитие голосовых помощников и сервисов, работающих только лишь с обработкой устной речи и её генерированием. В силу данных обстоятельств узкое направление IT-сферы, связанная с именно с обработкой текстов нуждается в большем внимании. Данные обстоятельства сподвигают автора на создание своего собственного веб-приложения, служащего для автоматизированной и быстрой обработке текстов.

Для воплощения данной задумки автору предстоит написать программу, нацеленную на корректную работу будущего веб-приложения. Следовательно, объектом данной курсовой работы является не сколько результат исследования, сколько процесс его достижения.

Учитывая всё вышесказанное, стоит отметить, что **целью** данной курсовой работы является создание веб-приложения для обработки текстов, польза которого была описана ранее.

Для достижения вышепоставленной цели следует выполнить следующие задачи:

1) Изучить некоторый новый материал для реализации вышеуказанного проекта;
2) Написать интуитивно понятный и удобный интерфейс;
3) Реализовать алгоритм обработки текста;
4) Реализовать оценочные данные о принятом в обработку тексте;
5) Провести тесты на некоторых текстах;
6) Описать все процессы проделанной работы.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ И МЕТОДЫ РЕАЛИЗАЦИИ

1.1. Использование «JavaScript» и «React.js»

Если целью данного проекта является создание приложения, то нужно иметь понимание о том, какими качествами, характеристиками и аспектами оно должно обладать. Так как задуманная автором работы веб-приложение создаётся для пользователя, не имеющего глубоких познаний в IT-сфере, оно должно обладать следующими характеристиками: удобство, понятность, простота в использовании, корректность получаемых результатов, универсальность.

Для создания программы с данными характеристиками нужно было принять взвешенного и аргументированное решение, связанное с выбором одного из многочисленных языков программирования. По этой причине на соответствие некоторым требованиям данного проекта были проанализированы следующие языки программирования: Python, JavaScript, Swift, PHP. Для создания веб-приложения мог бы подойти любой из этих языков, но некоторые из них являются менее удобными в использовании, так как порог вхождения для работы с ними является более высоким. Исходя из данных размышлений, автор пришёл к выводу о том, что JavaScript является наиболее удачным для реализации всех процессов, запланированных в ходе создания веб-приложения.

«JavaScript — это легковесный, интерпретируемый или JIT-компилируемый, объектно-ориентированный язык с функциями первого класса» [1]. данный факт позволяет использовать в качестве переменных объекты, которые могут хранить в себе несколько видов информации. Большое количество команд, связанных с работой объектов, сильно упрощают реализацию задуманных решений и позволяют написать «красивый», логичный и очень ёмкий код.

Нативный JavaScript хоть и универсален, но не позволяет легко реализовать нужный для пользователя интерфейс. Для того, чтобы исправить эту проблему было принято решение использовать дополнительную библиотеку или фреймворк для JavaScript. Проанализировав возможные варианты дополнительных

инструментов, автор принял решение использовать библиотеку «React.js», выбранную по причине её постоянной и регулярной поддержки со стороны разработчиков и компонентной модели написания кода. «React — это декларативная, эффективная и гибкая библиотека JavaScript для создания пользовательских интерфейсов (UI). Она позволяет вам создавать сложные UI из небольших и изолированных частей кода, называемых “компонентами”» [2]. Последнее позволяет разбить написанный код на несколько файлов, в которых может находиться логика интерфейса, что существенно упрощает его написание, так как обращение и вызов этих файлов-компонентов может быть многократным.

1.2. Методы обработки и анализ текста

Для дальнейшего понимания процесса написания кода программы стоит описать предназначение самого веб-приложения. Исходя из первоначальной задумки, пользователь данной программы с её помощью может получить на выходе текст, в котором отсутствуют так называемые филологами и лингвистами «мусорные» слова, а также получить статистику в процентном соотношении, связанную с распределением слов из текста по следующим лексическим уровням: A1-A2, B1-B2, C1-C2.

Для реализации вышеописанной задумки программа получает на вход текст, в котором считывает отдельно каждое слово. Если считываемое слово находится в базе «мусорных» слов, введённых пользователем, и соответствует некоторым условиям его положения по отношению к тексту, то оно удаляется, в противном случае программа переходит к проверке следующего слова на наличие его в базе. Таким образом, программа ещё и составляет базу уникальных слов и их количество для заданного пользователем текста. Операция по чистке текста необходима для получения более корректной статистики, так как наличие «мусорных» слов искажает статистические данные в худшую сторону.

Само распределение слов по лексическим уровням происходит определённым образом. Для того, чтобы определить лексический уровень какого-

либо слова в код программы были включены списки со словами, относящимися к определённым лексическим уровням (A1-A2, B1-B2, C1-C2). Программа считывает каждое из уникальных слов и ищет их в каком-либо из списков, после чего программа подсчитывает какое количество слов относится к каждому из этих уровней. На основе этих подсчётов выводится общий уровень текста и решение о том, для какого читателя подходит этот текст.

ГЛАВА 2. РЕАЛИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ В ПРАКТИЧЕСКОЙ РАБОТЕ

2.1. Реализация интуитивно понятного интерфейса

Для того, чтобы интерфейс был понятным и простым были выяснено, что нужно ввести следующие части интерфейса: область для ввода исходного текста, поле для ввода слов, которые нужно удалить, область, предназначенная для вывода очищенного текста, область для введения текста исключительно на анализ и четыре кнопки, запускающие соответствующий скрипт.

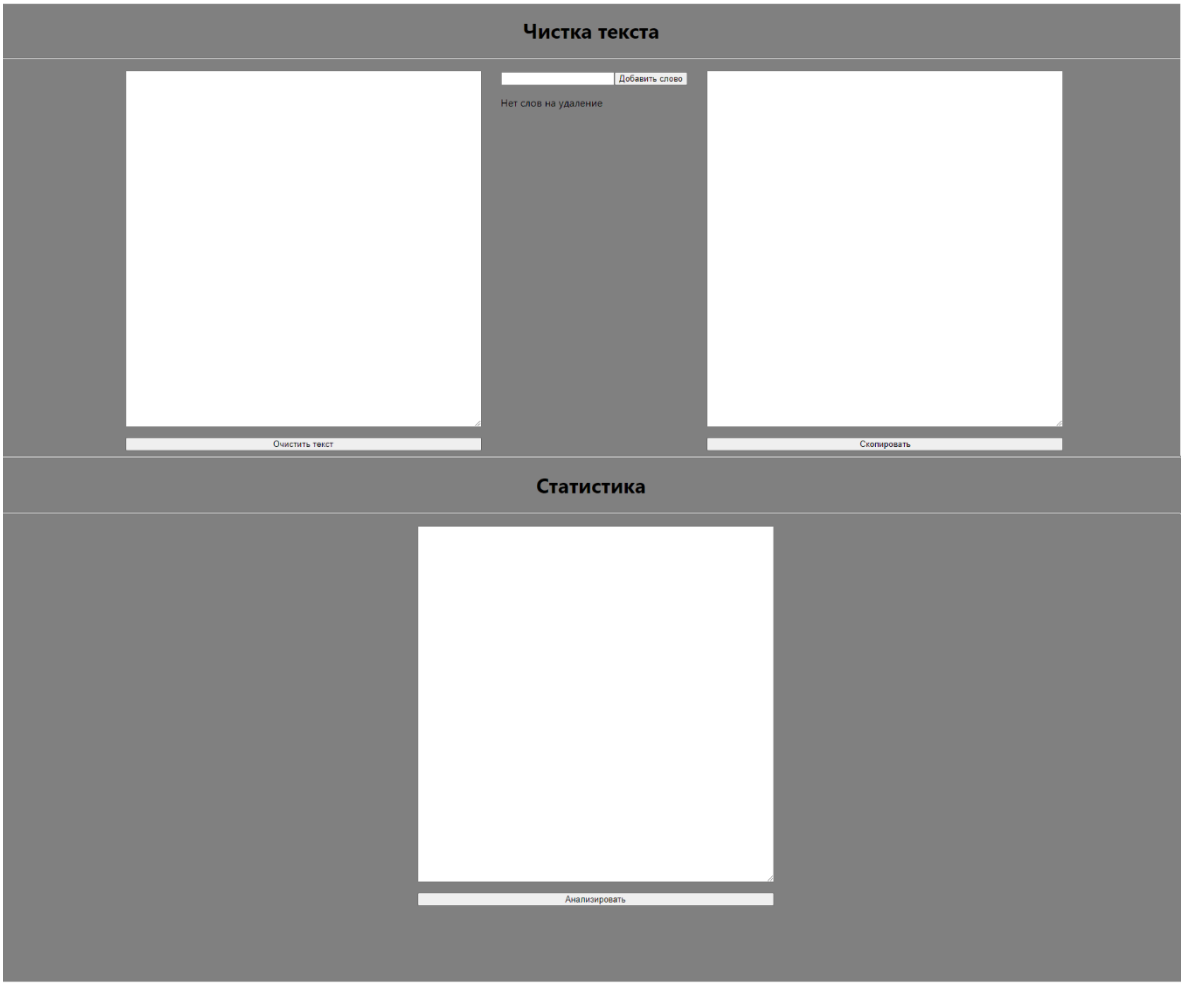

Рисунок 1

Начиная описывать элементы интерфейса, стоит начать с первой области, служащей для ввода текста, и кнопки его очистки. Область для ввода текста представляет из себя элемент «<textarea>», значения из которого переходят в компонент «ClearText», после чего при нажатии на кнопку «очистить текст»

программа получает нужные ей данные и работает с ними. «<textarea> представляет собой элемент формы для создания области, в которую можно вводить несколько строк текста. В отличие от элемента <input> в текстовом поле допустимо делать переносы строк, они сохраняются при отправке данных на сервер» [3].

```
<div style={styles.mainDiv}>
  <div style={styles.areaDiv}>
    <textarea style={styles.areaSet} value={textToClear} onChange={event => setTextToClear(event.target.value)}></textarea>
    <ClearText textToClear={textToClear} baseWords={baseWords} setNewClearText={setNewClearText} setNewBaseToAnalysis={setNewBaseToAnalysis} />
  </div>
```

Рисунок 2

```
function ClearText({textToClear, baseWords, setNewClearText, setNewBaseToAnalysis}) {

    function NeedToClear() {···
    }



    return (

        <button style={styles.button} type="submit" onClick={() => NeedToClear()}>Очистить текст</button>


    )
}
```

Рисунок 3

Следующий элемент интерфейса представляет из себя поле для ввода слов в базу «мусорных». Для реализации данной части потребовалось создать несколько отдельных компонентов кода, таких как: «DeleteWordBaseList», «DeleteWordBaseItem», «AddDeleteWord». В этих компонентах реализованы следующие функции: добавление слов в базу, возможность их удаления из неё, наглядное представление базы пользователю. Слова, входящие в эту базу, также передаются в компонент «ClearText» и именно с ними сравнивается считываемое программой слово.

```
<div>
  <AddWord onCreate={addWord}/>
  {baseWords.length ? (
  <DeleteWordBaseList baseWords={baseWords} />
  ) : (
  <p style={styles.p}>Нет слов на удаление</p>
  )}
</div>
```

Рисунок 4

```
import React, {useState} from "react";···

const styles = {···
}


function AddWord({onCreate}) {
    const [value, setValue] = useState('')

    const addDeleteWords = (input) => {
        return input.split(' ').map(item => ({
          title: item,
          id: Math.floor(Math.random() * 100000)
        }))
      }

    function submitHandler(event) {
        event.preventDefault()

        onCreate(addDeleteWords(value))
        setValue('')
    }
    return (
        <form style={styles.form} onSubmit={submitHandler}>
            <input value={value} onChange={event => setValue(event.target.value) } />
            <button type='submit'>Добавить слово</button>
        </form>
    )
    }

AddWord.propTypes = {···
}

export default AddWord
```

Рисунок 5

```
1  > import React from "react";···
4
5  > const styles = {···
15   }
16
17   function DeleteWordBaseList(props) {
18       return (
19           <ul style={styles.ul}>
20               {props.baseWords.map((baseWord, index) => {
21                   return (
22                       <DeleteWordBaseItem baseWord={baseWord} key={baseWord.id} index={index}/>
23                   )
24
25               })}
26
27           </ul>
28
29       )
30   }
31
32   DeleteWordBaseList.propTypes = {
33       baseWords: PropTypes.arrayOf(PropTypes.object).isRequired
34
35   }
36
37   export default DeleteWordBaseList
```

Рисунок 6

```
1  > import React, {useContext} from "react"···
4
5
6  > const styles = {···
17   }
18
19   function DeleteWordBaseItem({baseWord}) {
20       const {removeWord} = useContext(Context)
21       return (
22           <li style={styles.li}>
23               <span>
24                   {baseWord.title}
25               </span>
26               <button className='cr' onClick={removeWord.bind(null, baseWord.id)}>&times;</button>
27           </li>
28
29       )
30
31   }
32
33 > DeleteWordBaseItem.propTypes = {···
36   }
37
38   export default DeleteWordBaseItem
```

Рисунок 7

Далее будет описан такой элемент интерфейса как область для вывода очищенного текста и кнопка, находящаяся ниже. Область для вывода также является элементом «<textarea>», в который поступает значение, с уже очищенным текстом, из компонента «ClearText». Кнопка, находящаяся ниже, позволяет быстро

скопировать содержимое этой области. Это особенно удобно, если вводимый текст большой и не вмещается в одну страницу указанной области.

```
<div style={styles.areaDiv}>
  <textarea style={styles.areaSet} value={clearedText} onChange={event => setClearedText(event.target.value)} id='TextToCopy'></textarea>
  <Copy />
</div>
```

Рисунок 9

```
1   import React from "react";
2
3   const styles = {···
7   }
8
9   function Copy() {
10      function GetCopy() {
11          let textToCopy = document.getElementById('TextToCopy')
12          textToCopy.select()
13          document.execCommand('copy')
14      }
15
16      return (
17          <button style={styles.button} type="submit" onClick={() => GetCopy()}>Скопировать</button>
18      )
19
20
21  }
22
23  export default Copy
24
25
```

Рисунок 8

Последними элементами интерфейса является область для введения текста на анализ и кнопка, запускающая нужный скрипт. Данная область необходима чтобы пользователь мог анализировать текст без предварительной его очистки в тех случаях, когда в этом нет необходимости. Это поле является элементом «<textarea>», оно получает на вход текст, который необходимо анализировать. После нажатия кнопки значение из поля отправляется в компонент «Statistick», где слова, находящиеся в тексте, проверяются на принадлежность к какому-либо лексическому уровню. После программа подсчитывает количество слов каждого уровня, сравнивает их отношение между собой. Затем, основываясь на преобладающем лексическом уровне, выводит информацию о том, какому читателю рекомендован к прочтению данный текст.

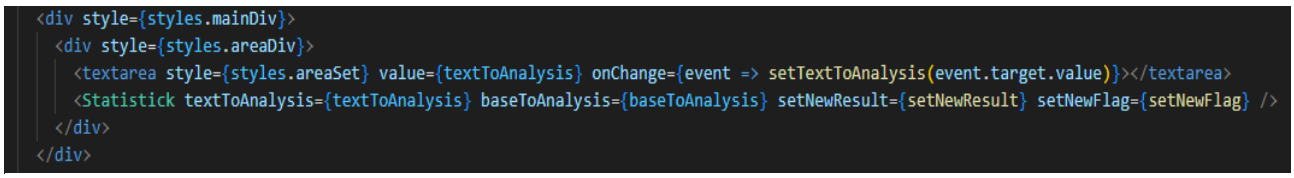

Рисунок 11

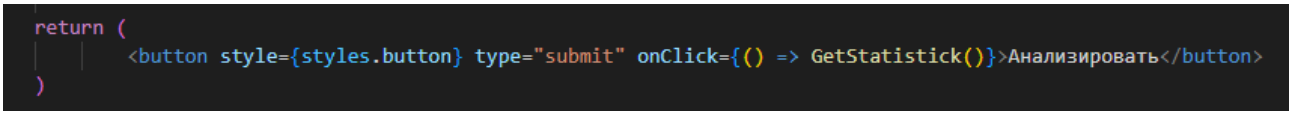

Рисунок 10

2.2. Реализация алгоритмов обработки и анализа текстовых данных

Исходя из вышеуказанной информации становится понятно, что компоненты «ClearText» и «Statistick» принимают в себя некоторые данные. Но стоит также отметить, что они с этими данными взаимодействуют.

Компонент «ClearText» принимает в себя исходный текст. Так как в JavaScript строки неизменяемы, внутри компонента создаётся новая переменная. Изменённый текст записывается в новую переменную определённым образом. Программа считывает символ из оригинального текста, если этот символ, не является буквой русского алфавита, то он записывается в новую переменную текста без изменений. Если символ относится к алфавиту русского языка, то программа записывает его в переменную «currentWord». Если переменная «currentWord» не пустая, а программа считала символ, который не является буквой русского алфавита, то содержимое переменной «currentWord» проверяется на наличие в базе «мусорных» слов. И если находится совпадение, то переменная «currentWord» приравнивается к пустой строке и не записывается в новую переменную текста. В противном случает переменная «currentWord» записывается в новую переменную текста и только после этого приравнивается к пустой строке.

В то же время, переменная «currentWord» проверяется на наличие его содержимого в базе уникальных слов. Если слова из переменной «currentWord» нет в базе уникальных слов, в переменной «unicWordBase», которая является объектом, создаётся новый ключ со значением переменной «currentWord» которому

присваивается единица. Если считываемое из переменной «currentWord» слово уже есть в базе уникальных слов, то к ключу объекта «unicWordBase» со значением переменной «currentWord» прибавляется единица. Это нужно для подсчёта повторов уникальных слов в тексте.

Перейдём к рассмотрению компонента «Statistick». в этот компонент поступает текст и база уникальных слов, если до этого производилась очистка текста. По этой причине перед тем, как проводить анализ, программа проверяет есть ли в поступившей в неё базе уникальных слов что-либо. Если база не пустая, программа работает с ней, в противном случае программа составляет базу уникальных слов таким же способом, как и в компоненте «ClearText». В базе уникальных слов проверяется наличие ключа со значением каждого слова из документов со словами разной лексической сложности. Если ключ со значением слова из документа есть в базе уникальных слов, значение самого ключа вписывается в одну из трёх переменных: «wordsCountA», «wordsCountB», «wordsCountC» в зависимости от того, из какого документа было взято слово на проверку. Когда все подсчёты закончены, в зависимости от того, каких слов в тексте больше программа выводит одно из нескольких сообщений насчёт анализируемого текста.

ЗАКЛЮЧЕНИЕ

Подводя итоги всему вышесказанному, стоить заключить, что в результате работы было создано веб-приложение, написанное на языке JavaScript. Это приложение сокращает и автоматизирует работу специалистов с большими и объёмными текстами. С помощью данного приложения пользователь может провести чистку текста и получить оценку текста по уровню лексики.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Что такое JavaScript // https://developer.mozilla.org/ru/docs/Web/JavaScript (дата обращения: 27.05.2022)

2. Что такое React.js // https://learn-reactjs.ru/tutorial (дата обращения: 27.05.2022)

3. Что такое <textarea> // https://webref.ru/html/textarea (дата обращения: 27.05.2022)