

Колледж космического машиностроения и технологий

КУРСОВОЙ ПРОЕКТ

По МДК.01.02 «Прикладное программирование»

Тема: «Разработка приложения Веб-скрапер»

Выполнил студент

Денисов М.В.

Группа П1-18

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Принял преподаватель

Гусятинер Л.Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

Королёв 2021 г.

**Содержание**

[**Введение.** 3](#_Toc75178644)

[**1.Теоретическая часть.** 4](#_Toc75178645)

[**1.1. Описание предметной области.** 4](#_Toc75178646)

[**1.2. Описание существующих разработок.** 5](#_Toc75178647)

[1.2.1. Screaming Frog SEO Spider. 6](#_Toc75178648)

[1.2.2. Easy Web Extract. 6](#_Toc75178649)

[1.2.3. FMiner. 7](#_Toc75178650)

[**2. Проектная часть.** 9](#_Toc75178651)

[**2.1. Диаграмма прецедентов.** 9](#_Toc75178652)

[**2.2. Выбор Инструментов.** 9](#_Toc75178653)

[**2.3.Проектирование сценария.** 11](#_Toc75178654)

[**2.4.Диаграмма типа.** 12](#_Toc75178655)

[**2.5. Описание главной функции.** 13](#_Toc75178656)

[**2.6. Описание спецификаций к функциям.** 15](#_Toc75178657)

[**2.7.Описание функций.** 16](#_Toc75178658)

[**3. Эксплуатационная часть** 19](#_Toc75178659)

[**3.1.Руководство оператора** 19](#_Toc75178660)

[3.1.1. Назначение программы 19](#_Toc75178661)

[3.1.2. Условия выполнения программы 19](#_Toc75178662)

[**Заключение.** 21](#_Toc75178663)

[**Список литературы и интернет-источников** 22](#_Toc75178664)

# **Введение.**

Целью данного курсового проекта создание Веб-скрапера (парсера). Web разработки являются неотъемлемой частью в программирование. Данная программа позволит получить из HTML страницы данные, которые будут переведены в Excel.

В 1 части будет рассмотрена предметная область данной темы.

Во 2 части будут рассмотрены классы и методы, которые были разработаны, а также структура программной части.

В 3 части будет рассказано, как пользоваться данной программой, также показана инструкция по эксплуатации.

# **1.Теоретическая часть.**

## **1.1. Описание предметной области.**

Парсер — это программа, сервис или скрипт, который собирает данные с указанных веб-ресурсов, анализирует их и выдает в нужном формате.

С помощью парсера можно делать различные полезные задачи:

**•Метаданные**: Специалисты могут извлекать данные из тегов title и description и другие метаданные;

**•Анализы страниц в интернете:** с помощью парса страницы можно быстро найти сайты с ошибками по типу (404,500,403,503), также неработающие ссылки.

**•Товары:** характеристики, название, дата, описание всё это можно спарсить для поставщика если у него есть сайт с каталогом, но нет информации.

Парсинг схож работой с анализаторами, рассмотрим как работает «Синтаксический анализатор» видит Лессинг, который сделан по шаблону (по всем нормам, у него должно быть имя токена, а также необязательный аргумент), после этого идёт вызов «Лексического анализатора» который в свою очередь, проверяет на наличие ошибок в тексте, такие как отсутствие кавычек, неправильная запись слов. При таком раскладе происходит вызов следующего токена.

**Существует 2 вида парсинга, белый и серый.** Белый отвечающий за правое использование информации, в своих целях, без сбора контактных данных с сервисов по типу Яндекс. Карты и 2Гис.Серый парсинг отвечает за скачивание данных у конкурента или целого сайта целиком, из-за него у вас будут проблемы.

**Microsoft Excel** (также иногда называется **Microsoft Office Excel**) — программа для работы с электронными таблицами, созданная корпорацией Microsoft для Microsoft Windows, Windows NT и Mac OS, а также Android, iOS и Windows Phone. Она предоставляет возможности экономико-статистических расчетов, графические инструменты и, за исключением Excel 2008 под Mac OS X, язык макропрограммирования VBA (*Visual Basic for Application*). Microsoft Excel входит в состав Microsoft Office.

## **1.2. Описание существующих разработок.**

Есть несколько вариантов:

1. Оптимальный — если в штате есть программист (а еще лучше — несколько программистов). Поставьте задачу, опишите требования и получите готовый инструмент, заточенный конкретно под ваши задачи. Инструмент можно будет донастраивать и улучшать при необходимости.
2. Воспользоваться готовыми облачными парсерами (есть как бесплатные, так и платные сервисы).
3. Десктопные парсеры — как правило, программы с мощным функционалом и возможностью гибкой настройки. Но почти все — платные.
4. Заказать разработку парсера «под себя» у компаний, специализирующихся на разработке (этот вариант явно не для желающих сэкономить).

Первый вариант подойдет далеко не всем, а последний вариант может оказаться слишком дорогим.

Что касается готовых решений, их достаточно много, и если вы раньше не сталкивались с парсингом, может быть сложно выбрать.

### 1.2.1. Screaming Frog SEO Spider.

[**Screaming Frog SEO Spider**](https://www.screamingfrog.co.uk/seo-spider/) – популярная программа, специализирующаяся на работе с SEO данными, имеет широчайший функционал для аудита сайтов, полное перечисление возможностей программы займет не одну страницу. При первом знакомстве интерфейс программы может показаться громоздким из-за множества вкладок и окон, но поработав некоторое время становится понятно, что он удобен, вкладки позволяют получить быстрый доступ к отчетам, окна удобно структурируют результаты парсинга.

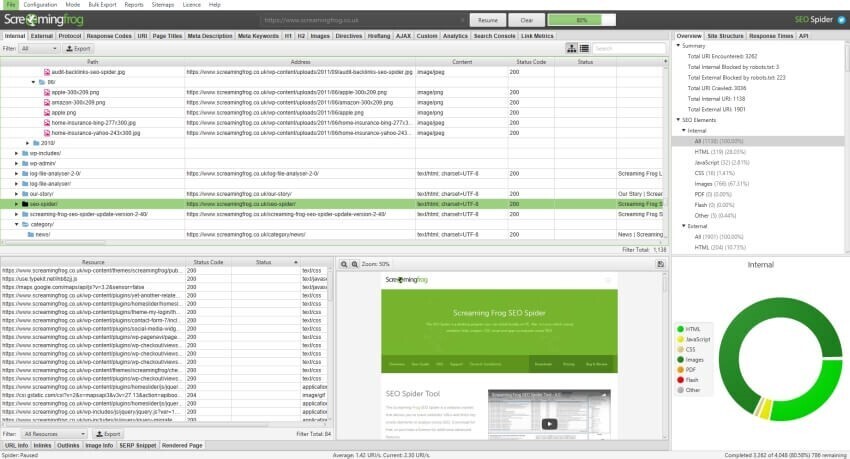


Рисунок 1. Работа с Screaming Frog SEO Spider.

### 1.2.2. Easy Web Extract.

**Easy Web Extract** дает массу возможностей, позволяющих собирать данные как с простых, так и со сложных сайтов. Программа не требует углубленных знаний программирования для настройки сбора данных, специальный мастер проведет вас по шагам в настройке шаблона парсинга, а для того, чтобы быстро разобраться с настройкой есть видео уроки. Одна из особенностей - вы можете запрограммировать автоматический поиск для определенных товаров и производить сбор только нужных данных. Еще одна особенность программы – сбор в несколько потоков, до 24 различных веб-страниц, это позволит сэкономить ваше время парсинга. Обратная сторона быстрого парсинга – блокировка вашего ip со стороны сайта из-за подозрительной активности, будьте аккуратны.

Некоторые сайты используют методы динамической загрузки данных на стороне клиента для создания асинхронных запросов. Такие данные проблема для простых парсеров, поскольку веб-контент не встроен в исходный HTML код. Easy Web Extract заявляет о возможности сбора таких данных, при тестировании программа справилась не со всеми сайтами, вам нужно проверить эту фичу на нужных вам сайтах.

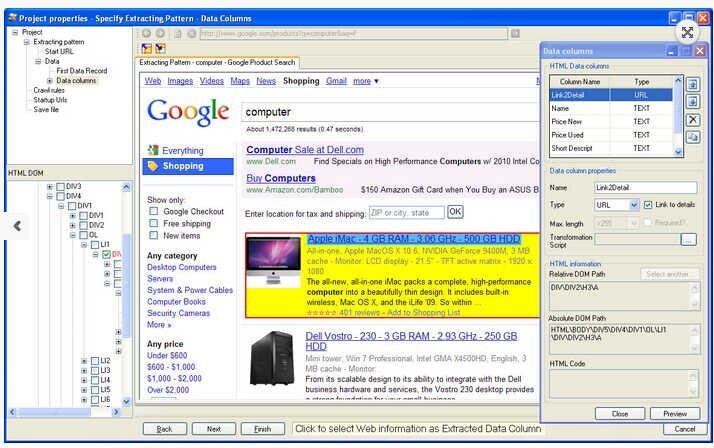


Рисунок 2. Работа с Easy Web Extract.

### 1.2.3. FMiner.

**FMiner** – инструмент для парсинга сайтов, работа которого построена на записи ваших действий и последующем воспроизведении записанных сценариев. Созданные таким образом последовательности действий (макросы), можно редактировать в визуальном формате, что позволяет использовать инструмент без знания языков программирования.

Программа работает с динамически подгружаемыми данными (AJAX), поддерживает работу с несколькими потоками, позволяет работать с результатами поиска и несколько выходных форматов. Программа имеет видеоуроки для быстрого старта, но страницы с мануалами не работают и последние обновления на сайте датированы 2015 годом, что говорит о том, что разработчик не следит за продуктом, но установочные файлы доступны и можно загрузить билд для бесплатного использования полнофункциональной версии программы в течении двух недель.

На сайте есть раздел с документацией, рассказывающий в краткой форме возможности и основы работы с программой.

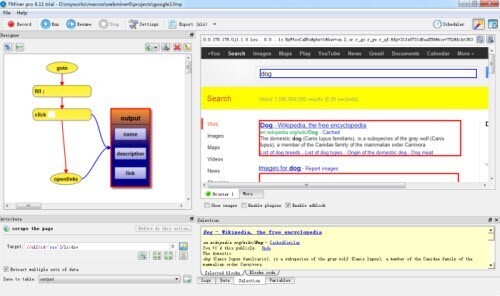


Рисунок 3. Работа с FMiner.

# **2. Проектная часть.**

## **2.1. Диаграмма прецедентов.**

В данном разделе показана диаграмма прецедентов, на ней изображено все возможные функциональные отношения.

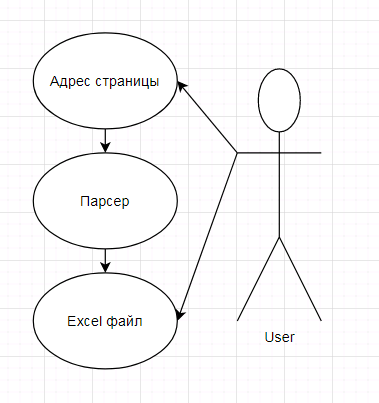


Рисунок 4. Диаграмма прецедентов.

## **2.2. Выбор Инструментов.**

Важность инструмента определялась по нескольким критериям. Выбирая инструмент выбраны критерии: низкая, ниже среднего, средняя, ниже высокой, высокая.

Таблица 1.

Важность критериев выбора.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерий | Участие в корпоративном проекте | Простота сопровождения | Наличие библиотек | Наличие документации на русском языке | Скорость разработки |
| Важность критерия | Ниже средней | Средняя | Высокая | Низкая | Ниже высокой |

Исходя из этих критериев, были выведены несколько языков программирования, всё оценивалось по 10-ти бальной системе за критерий.

Таблица 2.

Выбор языка по критериям.

|  |  |  |  |
| --- | --- | --- | --- |
|  | C++ | Golang | Java |
| Участие в корпоративном проекте | 1 | 8 | 1 |
| Простота сопровождения | 1 | 7 | 1 |
| Наличие библиотек | 1 | 10 | 1 |
| Наличие документации на русском языке | 1 | 2 | 1 |
| Скорость разработки | 1 | 8 | 1 |
| Итого баллов | 5 | 35 | 5 |

По результатам сравнения был выбран язык программирования Golang.

## **2.3.Проектирование сценария.**

В данном разделе приведён сценарий использования программы пользователем.

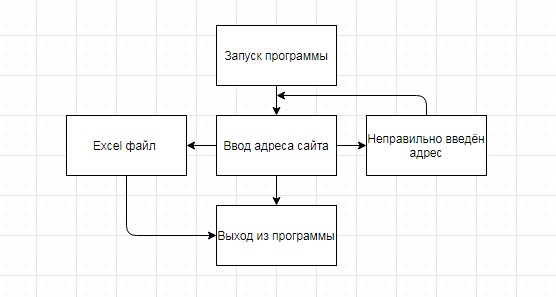


Рисунок 5. Проектирования сценария

Пользователь после запуска программы вводит адрес сайта с которого надо получить таблицы или может выйти из программы.

При выборе выхода программа заканчивает свою работу, при вводе адреса начинается поиск таблиц на сайте, после этого идёт создание Excel файла

## **2.4.Диаграмма типа.**

Go не похож на классические языки программирования. В нем нет классов и объектов, как в других популярных языках программирования вроде Python, Ruby или Java. Он также не использует наследование. Однако Go по-прежнему предоставляет инструменты для внедрения идей **объектно-ориентированного программирования.**

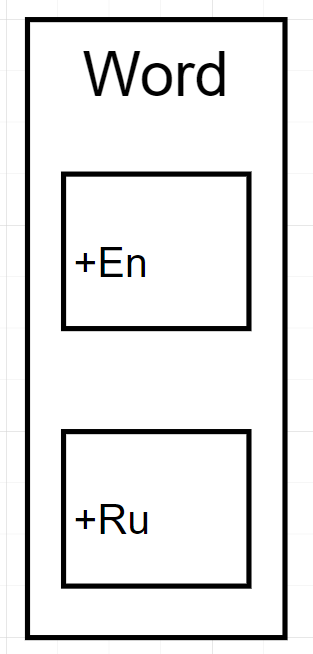


Рисунок 6. Показ Типа Word.

В данном случаем мы с нуля пишем новый тип Word и задаём, чтобы он был структурой из нескольких стрингов, они будут использованы, чтобы оставлять данные именно того языка, в моём случае парсинг сайта с 1000 слов на английском.

## **2.5. Описание главной функции.**

В главной функции, мы записываем адрес сайта, с основной проверкой на правильность и вывод данных. Данный метод заходит на адрес, и проверяет, можно ли осуществить действие или нет.

**Листинг 1. Главная Функция.**

func main(){

scrapPage("https://www.en365.ru/top1000.htm") //НУ здесь вписать сайт

scrapPage("https://www.en365.ru/top1000a.htm")//Если страниц несколько, то пишите ещё

scrapPage("https://www.en365.ru/top1000b.htm")

fmt.Printf("%s \n",wordCollection)// Для проверки вывода, выписываем в терминал, что получили

fmt.Printf("cnt %v \n",cnt)// Количество слов

if cnt == 0 {

fmt.Println("Ошибка") // Если ничего не заберёт, выведет ошибку

}writeResultXls()// Запись

}

**Блок-схема главной функции**.



Рисунок 7.Блок-схема главной функции.

## **2.6. Описание спецификаций к функциям.**

Разработаны 3 функции :

* main
* scrappage
* writeResultsXls

Структура работы функций изображена на Рисунке 8.

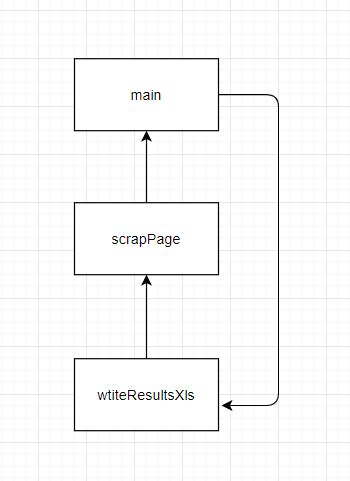


Рисунок 8. Структура работы функций.

## **2.7.Описание функций.**

**Функция 2. Поиск.**

Первая функция уже была описана выше. Поэтому начнём с 2-ой.

В данной функции реализован поиск элементов(слов), которые в свою очередь мы хотим переместить в Excel.

Листинг 1.Поиск элементов по дереву HTML.

func scrapPage(url string) {

c := colly.NewCollector()

c.OnHTML("div > table tr", func(e \*colly.HTMLElement) { //Сюда писать элемент (Только бог поможет понять хтмл код другого)

enWord := e.DOM.Find("td:nth-child(2)").Text() //Это если надо именно 2 или другой элемент в в дереве

ruWord := e.DOM.Find("td:nth-child(3)").Text() //Это если больше надо пройтись по строке ещё дальше

if !strings.Contains(enWord, "Английское") {

wordCollection = append(wordCollection, Word{enWord, ruWord})

cnt++

}

})

c.Visit(url)

}

**Функция 3. Excel.**

В данной функции происходит создание Excel файла и перенос данных из буфера в Excel файл.

Листинг 2.Создание и запись слов.

func writeResultXls() {

xlsx := excelize.NewFile()

xlsx.NewSheet("Запись")

for i, word := range wordCollection {

xlsx.SetCellValue("Запись", fmt.Sprintf("A%v", i+1), word.En)

xlsx.SetCellValue("Запись", fmt.Sprintf("B%v", i+1), word.Ru)

}

err := xlsx.SaveAs("./Slova.xlsx")

if err != nil {

fmt.Println(err)

}

}

**2.8. Описание тестовых наборов модулей.**

Каждый раз когда мы вводим адрес сайта. Идёт проверка на существование элементов в дереве HTML. В Golang есть защита от ошибок которая не даёт пользователю допустить запуск программы с ней. Поэтому всё ограничивается на написание неправильной программы.

Тест 1. Неправильная запись программы.

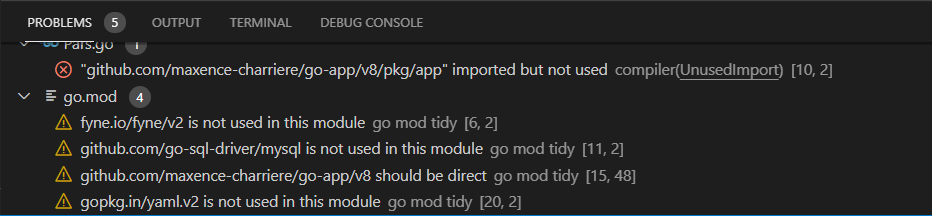


Рисунок 9. Ошибка при неправильной записи кода.

# **3. Эксплуатационная часть**

## **3.1.Руководство оператора**

### 3.1.1. Назначение программы

Разработанное приложение производит парсинг HTML кода и выводит его в виде Excel файла. Благодаря данной программе отдлы документации в разных сферах могут перекидывать данные с сайта с помощью одного нажатия. В будущем этот парсер можно использовать в более крупных разработках.

### 3.1.2. Условия выполнения программы

Операционная система: Windows 7/8/10.

Процессор: Intel Pentium G4600 или аналогичный

Оперативная память: 2гб

Свободное место на жёстком диске: ~150 Мб.

**3.1.3. Выполнение программы.**

При запуске программы в консоли будет написанно сколько слов нашёл парсер, после небольшого времени появиться Excel файл.



Рисунок 10. Название файла.

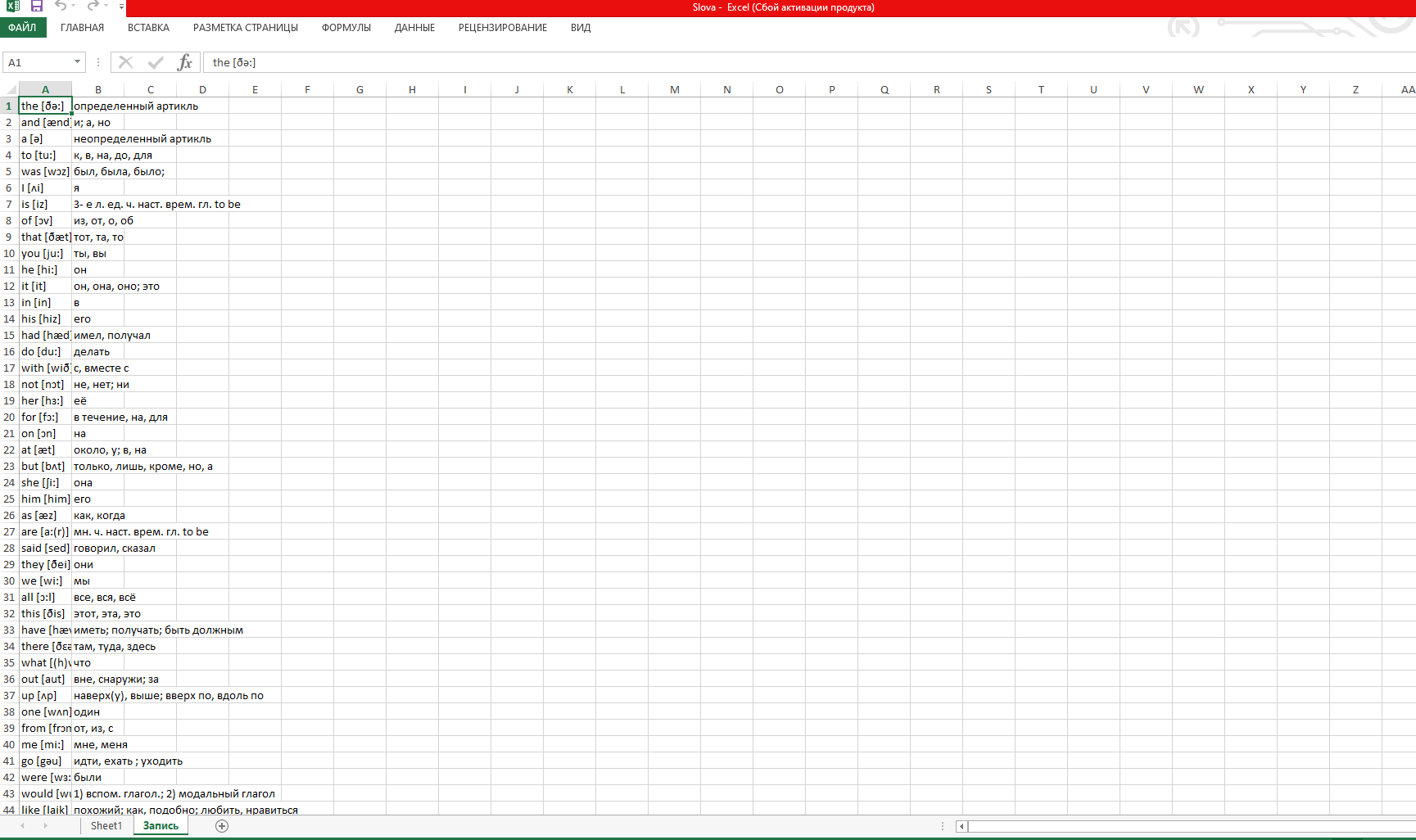


Рисунок 11. Excel файл.

3.2. To-Do лист.

* Создать интерактивную консольную программу для работы с HTML и модифицированием его

# **Заключение.**

В результате выполнения курсового проекта был разработан парсер HTML. Данная программа имеет потенциал для будущих разработок и ее можно будет использовать для более крупных проектов. Разработанная программа выполнена в соответствии с требованиями технического задания.

# **Список литературы и интернет-источников**

1. Классический учебник по теории построения компиляторов под авторством Альфреда В. Ахо, Рави Сетхи и Джеффри Д. Ульмана, известный также как «Книга дракона».
2. Парсер сайтов [1.2.1]

<https://parserok.ru/>

1. 30+ парсеров для сбора данных с любого cайта [1]

<https://habr.com/ru/company/click/blog/494020/#:~:text=%D0%9F%D0%B0%D1%80%D1%81%D0%B5%D1%80%20%E2%80%94%20%D1%8D%D1%82%D0%BE%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%2C%20%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81%20%D0%B8%D0%BB%D0%B8,%D0%A6%D0%B5%D0%BD%D1%8B.>

Приложение 1. Код.

package main

import (

"fmt"

"strings"

"github.com/360EntSecGroup-Skylar/excelize"

"github.com/gocolly/colly"

"github.com/maxence-charriere/go-app/v8/pkg/app"

)

type Word struct {

En string `excel:"en"`

Ru string `excel:"ru"`

}

var wordCollection = []Word{}

var cnt int

func main() {

scrapPage("https://www.en365.ru/top1000.htm") //НУ здесь вписать сайт

scrapPage("https://www.en365.ru/top1000a.htm")

scrapPage("https://www.en365.ru/top1000b.htm")

//fmt.Printf("%s \n", wordCollection)

fmt.Printf("cnt %v \n", cnt)

writeResultXls()

}

func scrapPage(url string) {

c := colly.NewCollector()

c.OnHTML("div > table tr", func(e \*colly.HTMLElement) { //Сюда писать элемент (Только бог поможет понять хтмл код другого)

enWord := e.DOM.Find("td:nth-child(2)").Text() //Это если надо именно 2 или другой элемент в в дереве

ruWord := e.DOM.Find("td:nth-child(3)").Text() //Это если больше надо пройтись по строке ещё дальше

if !strings.Contains(enWord, "Английское") {

wordCollection = append(wordCollection, Word{enWord, ruWord})

cnt++

}

})

c.Visit(url)

}

func writeResultXls() {

xlsx := excelize.NewFile()

xlsx.NewSheet("Запись")

for i, word := range wordCollection {

xlsx.SetCellValue("Запись", fmt.Sprintf("A%v", i+1), word.En)

xlsx.SetCellValue("Запись", fmt.Sprintf("B%v", i+1), word.Ru)

}

err := xlsx.SaveAs("./Slova.xlsx")

if err != nil {

fmt.Println(err)

}

}

//В разработке

// type hello struct {

// app.Compo

// name string

// }

// func (h \*hello) Render() app.UI {

// return app.Div().Body(

// app.H1().Body(

// app.Text("Приложение "),

// app.If(h.name != "",

// app.Text(h.name),

// ).Else(

// app.Text("Тест"),

// ),

// ),

// app.P().Body(

// app.Input().

// Type("text").

// Value(h.name).

// Placeholder("Введите адрес").

// AutoFocus(true).

// OnChange(h.ValueTo(&h.name)),

// ),

// )

// }