

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)

Королев 2021 г.

**Содержание**

[**Введение.** 1](#_Toc75707379)

[**1.Теоретическая часть.** 2](#_Toc75707380)

[**1.1. Описание предметной области.** 2](#_Toc75707381)

[**1.2. Описание существующих разработок.** 5](#_Toc75707382)

[1.2.1. Screaming Frog SEO Spider. 6](#_Toc75707383)

[1.2.2. Easy Web Extract. 7](#_Toc75707384)

[1.2.3. FMiner. 8](#_Toc75707385)

[**2. Проектная часть.** 12](#_Toc75707386)

[**2.1. Диаграмма прецедентов.** 12](#_Toc75707387)

[**2.2. Выбор Инструментов.** 12](#_Toc75707388)

[2.2.2. Среда разработки. 15](#_Toc75707389)

[**2.3.Проектирование сценария.** 15](#_Toc75707390)

[**2.4.Диаграмма типа.** 16](#_Toc75707391)

[**2.5. Описание главной функции.** 17](#_Toc75707392)

[**2.6. Описание спецификаций к функциям.** 20](#_Toc75707393)

[**2.7.Описание функций.** 21](#_Toc75707394)

[**3. Эксплуатационная часть** 24](#_Toc75707395)

[**3.1.Руководство оператора** 24](#_Toc75707396)

[3.1.1. Назначение программы 24](#_Toc75707397)

[3.1.2. Условия выполнения программы 24](#_Toc75707398)

[**Заключение.** 26](#_Toc75707399)

[**Список литературы и интернет-источников** 27](#_Toc75707400)

[**Приложение Код** 28](#_Toc75707401)

# **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-ой части будет рассмотрена предметная область данной темы.

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

В 3-ей 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.

Существует 2 вида анализаторов:

**Синтаксический анализатор**, — часть программы, преобразующей входные данные (как правило, текст) в некий структурированный формат, нужный для задач последующего их (данных) анализа и использования. Технически, парсер выполняет синтаксический анализ данных (например, текста).

**Лексический анализатор**  — это программа или часть программы, выполняющая лексический анализ. Лексический анализатор обычно работает в две стадии: сканирование и оценка.

**•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.

В данном случае я использую парсинг сайта, так что же такое прасинг сайта? Простыми словами парсинг – это автоматизированный сбор информации с любого сайта, ее анализ, преобразование и выдача в структурированном виде, чаще всего в виде таблицы с набором данных. Парсер сайта — это любая программа или сервис, которая осуществляет автоматический сбор информации с заданного ресурса.

Процесс парсинга — это автоматическое извлечение большого массива данных с веб-ресурсов, которое выполняется с помощью специальных скриптов.

Если кратко, то парсер ходит по ссылкам указанного сайта и сканирует код каждой страницы, собирая информацию о ней в Excel-файл либо куда-то еще. Совокупность информации со всех страниц сайта и будет итогом парсинга сайта.

Парсинг работает на основе XPath-запросов, это язык, который обращается к определенному участку кода страницы и извлекает из него заданную критерием информацию.

**Алгоритм стандартного парсинга сайта.**

1. Поиск необходимых данных в исходном виде.
2. Извлечение данных с отделением от программного кода.
3. Формирование отчета согласно требованиям, которые были заданы.

Парсинг сайта – это рутинная и трудоемкая работа. Если вручную извлекать информацию из сайта, в котором всего 10 страниц, не такая сложная задача, то анализ сайта, у которого 50 страниц и больше, уже не покажется такой легкой.

Кроме того нельзя исключать человеческий фактор. Человек может что-то не заметить или не придать значения. В случае с парсером это исключено, главное его правильно настроить.

Если кратко, то парсер позволяет быстро, качественно и структурировано получить необходимую информацию. Далее я продемонстрирую схема алгоритма парсинга.

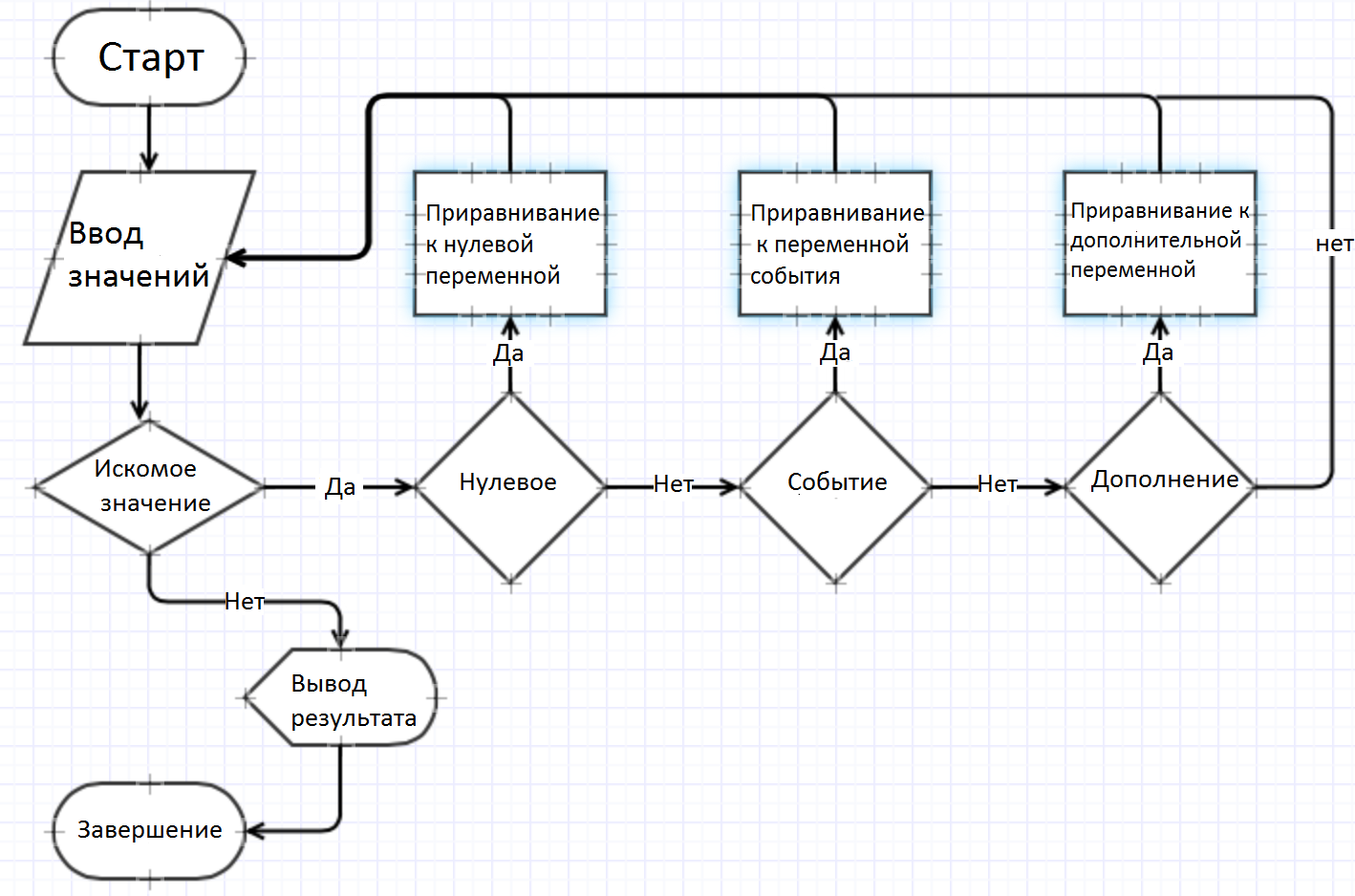


Рисунок 1. Схема парсинга сайта.

## **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.

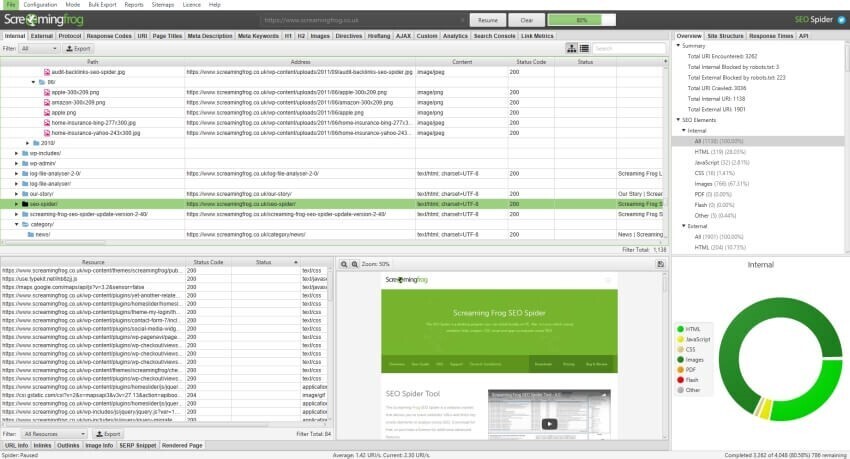


Рисунок 2. Работа с Screaming Frog SEO Spider.

### 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.

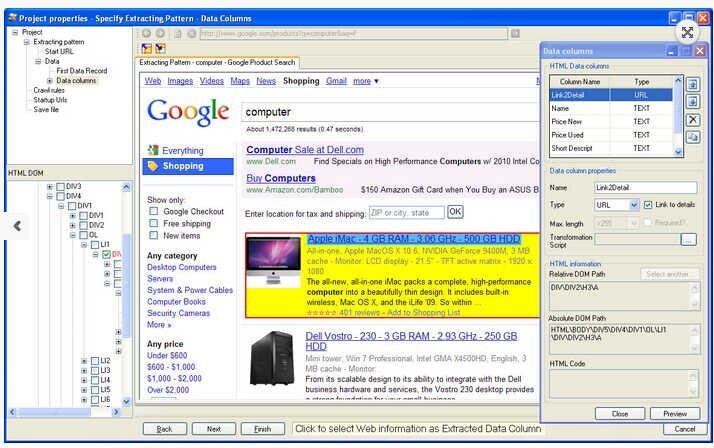


Рисунок 3. Работа с Easy Web Extract.

### 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.

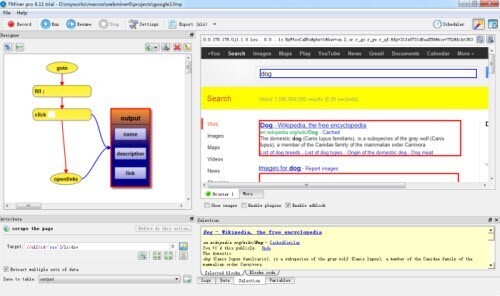


Рисунок 4. Работа с FMiner.

HTML (HyperText Markup Language) – дословно переводится как “язык гипертекстовой разметки”, где с помощью специальных слов и символов (тегов) осуществляется форматирование текста, изображений и видео на экранах устройств.

HTML называют языком, но не нужно путать его с языком программирования – это язык разметки, который ничего не делает с данными, не производит никаких вычислений, преобразований и прочих изменений, он только расставляет информацию по разным частям монитора и показывает её в удобном для восприятия виде.

HTML не может сложить 2+2, зато может раскрасить эти цифры в разные цвета, поменять размер, сдвинуть вправо, влево, вверх или вниз, подложить фон и т.д.

HTML теги тоже очень важная составляющая часть гипертекстовой разметки, скорее даже значение из-за которого придаются свойства.

HTML тег – это группа слов и символов, которые придают объекту, в отношении которого применяются, особые свойства.

Каждый тег имеет свое назначение и четко прописанный синтаксис (правила использования).

Все HTML теги имеют по краям скобки “<” “>”, обозначающие, соответственно, начало и конец тега.

Каждая страница в Интернете – это файл, содержащий HTML код. Подобный файл может находиться не только в сети, но и на домашнем компьютере или ноутбуке. Место его размещения не влияет на свойства.

Подобные файлы имеют следующие варианты расширений:

.html;

.htm.

Они ничем не отличаются.

Помимо стандартных для разметки страниц форматов, HTML код может содержаться в файлах с расширением .php и некоторых других языков программирования. Для открытия файлов с HTML кодом используются специальные программы – браузеры (Яндекс Браузер, Opera, Google Chrome, Mozilla Firefox, Internet Explorer и другие). Они преобразуют исходный код страницы в удобный восприятию вид. Когда мы посещаем Интернет сайты, то не видим никаких тегов, но благодаря браузерам видим результаты их работы.

В идеале каждый браузер должен действовать одинаково, каждый тег и свойство обрабатывать по стандарту и картинка для пользователя отличаться не должна, но на практике бывают сбои. Возможно вы встречались со страницами где текст съехал или картинка наложилась на другую, но если открыть страницу другим браузером всё отображается нормально. Чаще всего это связано с тем, что язык HTML разметки развивается, появляются новые теги и свойства, а браузеры за этими изменениями не успевают, либо не могут договориться об едином стандарте использования.

# **2. Проектная часть.**

## **2.1. Диаграмма прецедентов.**

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

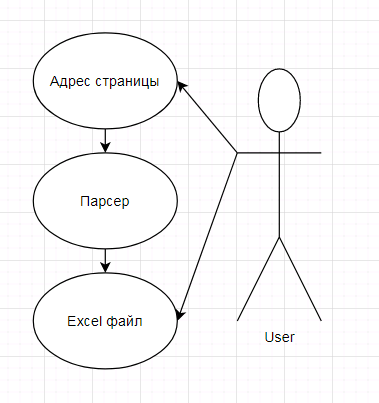


Рисунок 5. Диаграмма прецедентов.

## **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 | 2 |
| Простота сопровождения | 1 | 7 | 5 |
| Наличие библиотек | 1 | 10 | 3 |
| Наличие документации на русском языке | 1 | 2 | 1 |
| Скорость разработки | 1 | 8 | 1 |
| Итого баллов | 5 | 35 | 5 |

Язык Go разрабатывался как язык программирования для создания высокоэффективных программ, работающих на современных распределённых системах и многоядерных процессорах. Он может рассматриваться как попытка создать замену языкам Си и C++ с учётом изменившихся компьютерных технологий и накопленного опыта разработки крупных систе. По словам Роба Пайка, «Go был разработан для решения реальных проблем, возникающих при разработке программного обеспечения в Google». В качестве основных таких проблем он называет:

• медленную сборку программ;

• неконтролируемые зависимости;

• использование разными программистами разных подмножеств языка;

• затруднения с пониманием программ, вызванные неудобочитаемостью кода, плохим документированием и так далее;

• дублирование разработок;

• высокую стоимость обновлений;

• несинхронные обновления при дублировании кода;

• сложность разработки инструментария;

• проблемы межъязыкового взаимодействия.

Основными требованиями к языку стали:

• Ортогональность. Язык должен предоставлять небольшое число средств, не повторяющих функциональность друг друга.

• Простая и регулярная грамматика. Минимум ключевых слов, простая, легко разбираемая грамматическая структура, легко читаемый код.

• Простая работа с типами. Типизация должна обеспечивать безопасность, но не превращаться в бюрократию, лишь увеличивающую код. Отказ от иерархии типов, но с сохранением объектно-ориентированных возможностей.

• Отсутствие неявных преобразований.

• Сборка мусора.

• Встроенные средства распараллеливания, простые и эффективные.

• Поддержка строк, ассоциативных массивов и коммуникационных каналов.

• Чёткое разделение интерфейса и реализации.

• Эффективная система пакетов с явным указанием зависимостей, обеспечивающая быструю сборку.

По результатам сравнения был выбран язык программирования Golang.

### 2.2.2. Среда разработки.

**Visual Studio Code [8]** – это легкий, но мощный редактор исходного кода, который работает на рабочем столе и доступен для Windows, macOS и Linux. Он поставляется со встроенной поддержкой JavaScript, TypeScript и Node.js и имеет богатую экосистему расширений для других языков (таких как C++, C#, Java, Python, PHP, Go).

Visual Studio Code имеет огромное количество плагинов, поэтому написание кода ускоряется в разы.

## **2.3.Проектирование сценария.**

В данном разделе приведён сценарий использования программы пользователем.

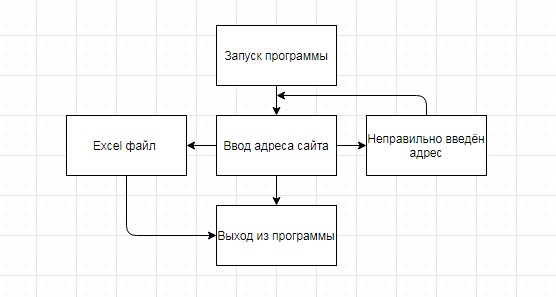


Рисунок 6. Проектирования сценария

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** (часто также **golang**) — компилируемый многопоточный язык программирования, разработанный внутри компании Google. Разработка Go началась в сентябре 2007 года, его непосредственным проектированием занимались Роберт Гризмер, Роб Пайк и Кен Томпсон, занимавшиеся до этого проектом разработки операционной системы Inferno. Официально язык был представлен в ноябре 2009 года. На данный момент поддержка официального компилятора, разрабатываемого создателями языка, осуществляется для операционных систем FreeBSD, OpenBSD, Linux, macOS, Windows, DragonFly BSD, Plan 9, Solaris, Android, AIX. Также Go поддерживается набором компиляторов gcc, существует несколько независимых реализаций. Ведётся разработка второй версии языка. 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.**

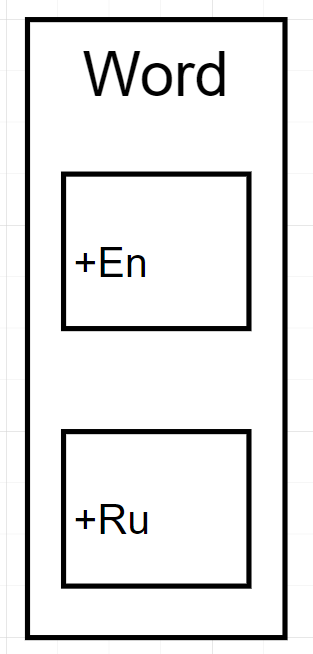


Рисунок 7. Показ Типа Word.

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.

**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**.

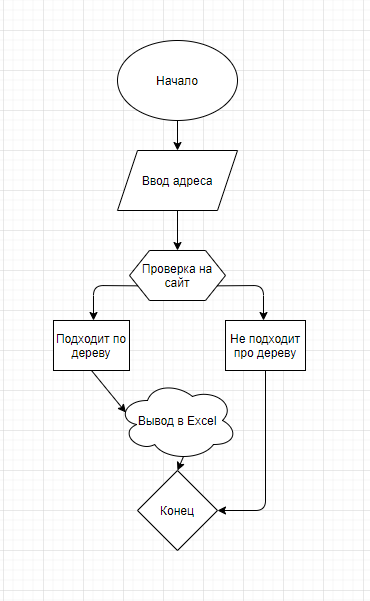


Рисунок 8.Блок-схема главной функции.

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

Разработаны 3 функции :

* main
* scrappage
* writeResultsXls

Структура работы функций изображена на Рисунке 8.

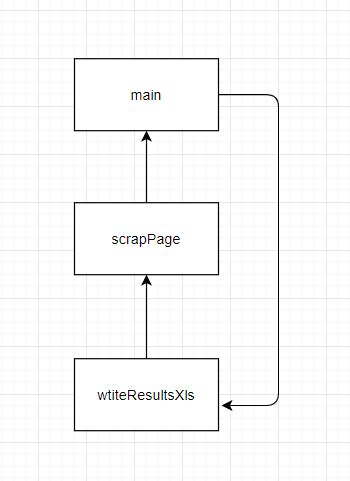


Рисунок 9. Структура работы функций

## **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.

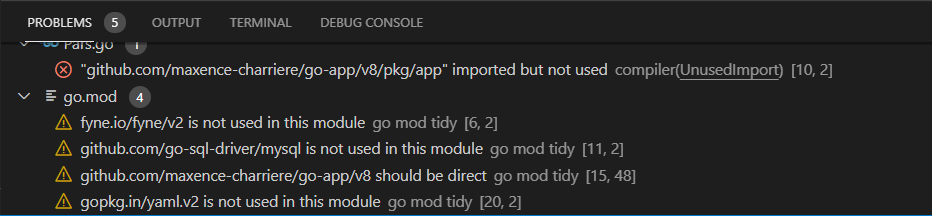


Рисунок 10. Ошибка при неправильной записи кода.

# **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 11. Название файла.

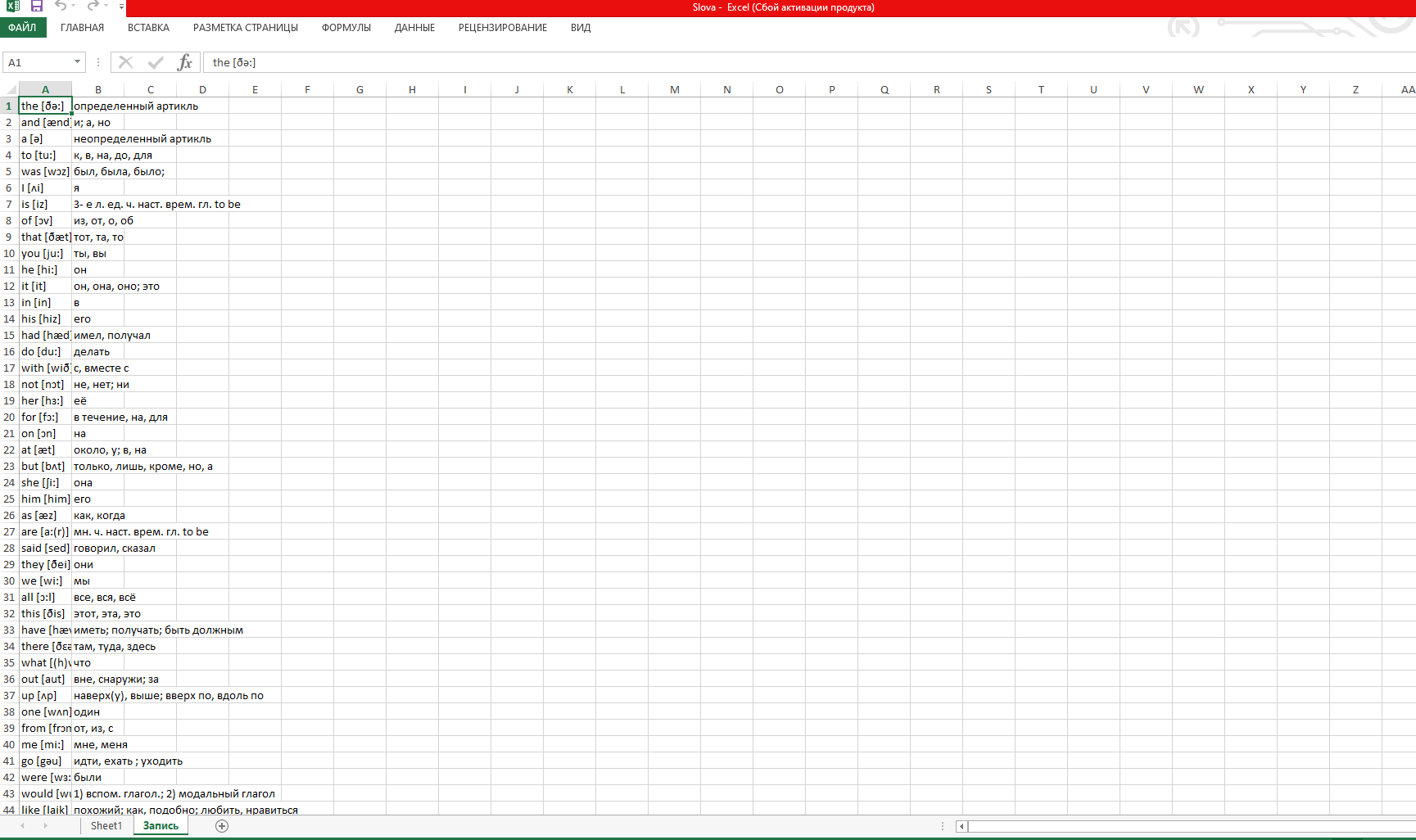


Рисунок 12. Excel файл.

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 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)),

// ),

// )

// }