

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«Национальный исследовательский ядерный университет «МИФИ»  
**Обнинский институт атомной энергетики –**  
филиал федерального государственного автономного образовательного учреждения  
высшего образования «Национальный исследовательский ядерный университет «МИФИ»  
**(ИАТЭ НИЯУ МИФИ)**  
Отделение интеллектуальных и кибернетических систем

**Отчет**  
Практика по получению профессиональных умений и опыта профессиональной  
деятельности

Выполнил:  
студент гр. ИС-Б16з

\_\_\_\_\_  
(подпись, дата)

Галиханов А.Ф.

Проверил,  
доцент ОИКС, к.ф. -м.н

\_\_\_\_\_  
(подпись, дата)

Качанов Б. В.

Обнинск, 2020 г.

## СОДЕРЖАНИЕ

|  |    |
|--|----|
| ВВЕДЕНИЕ.....  | 3  |
| ГЛАВА 1. АНАЛИЗ ВЫБРАННОГО РЕСУРСА.....  | 4  |
| 1.1 Исследование выбранного ресурса.....   | 4  |
| 1.2 Разработка алгоритма извлечения данных.....                                      | 7  |
| 1.3 Формат данных.....   | 8  |
| ГЛАВА 2. РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ<br>ИЗВЛЕЧЕНИЯ И ОЧИСТКИ ДАННЫХ..... | 10 |
| 2.1 Извлечение ссылок первого уровня.....  | 10 |
| 2.2 Извлечение ссылок второго уровня.....  | 12 |
| 2.3 Извлечение и преобразование требуемых данных.....                                | 13 |
| 2.4 Запись данных в MongoDB.....   | 15 |
| ЗАКЛЮЧЕНИЕ.....  | 17 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....  | 18 |
| ПРИЛОЖЕНИЯ.....  | 19 |

## ВВЕДЕНИЕ

Цель практической работы:

Подготовка данных необходимая для работы приложения, которая планируется реализовать в будущем, в рамках выпускной квалификационной работы.

Задачи решаемые в ходе выполнения практической работы:

- 1) Поиск ресурсов для извлечения необходимых данных
- 2) Исследование и анализ найденных ресурсов
- 3) Разработка алгоритма извлечения данных на основе проведенного анализа
- 4) Извлечение и очистка данных в соответствии с разработанным алгоритмом
- 5) Выбор СУБД удовлетворяющий требованиям
- 6) Преобразование полученных данных к необходимому виду(формату) и запись в БД

# ГЛАВА 1. АНАЛИЗ ВЫБРАННОГО РЕСУРСА

## 1.1 Исследование выбранного ресурса

В качестве источника данных был выбран сайт <http://www.goodsmatrix.ru/> Интернет-Каталог товаров: GoodsMatrix. На рисунке 1 представлена главная страница этого сайта.

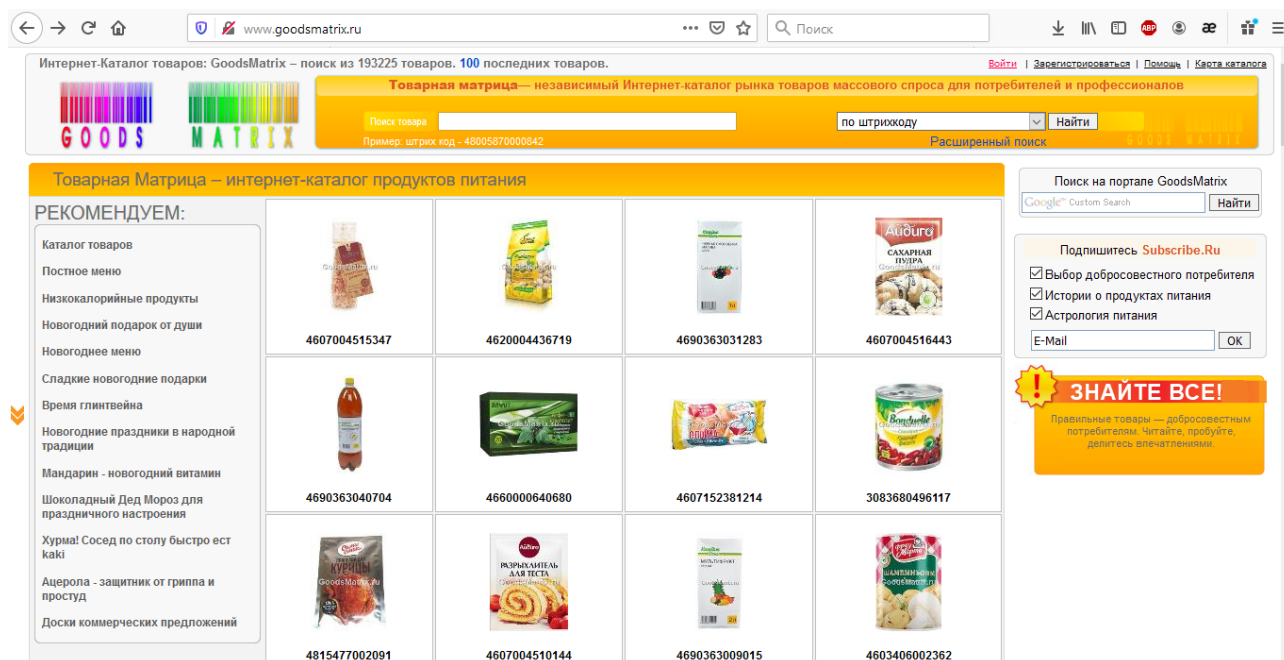


Рисунок 1 – базовый адрес источника данных

Необходимые нам данные находятся по адресу:

<http://www.goodsmatrix.ru/goods/h/> + штрих - код + [.html](#)

Пример: <http://www.goodsmatrix.ru/goods/h/4600384006634.html>.

Часть адреса [ <http://www.goodsmatrix.ru/goods/h/> ] одинаковый для всех товаров, своего рода шаблон. Числа в конце адреса [ [4600384006634](#) ] – это штрих-код продукта являющийся уникальным для каждого товара. На рисунке 2 продемонстрирован пример страницы с необходимыми данными.

Получить ссылки на все продукты сайта можно через раздел каталог товаров. Однако страница данного раздела, реализован в виде сложной древовидной структуры с большим количеством подкатегории, образуя многоуровневую систему вложенности. На каждом уровне большое количество категорий со своим уникальным идентификатором для которых необходимо

реализовывать свой путь. Демонстрация данного раздела представлена на рисунке 3. Таким образом алгоритм получения необходимых страниц, через каталог товаров был исключен, так как требует большой объем программного кода и времени для реализации.

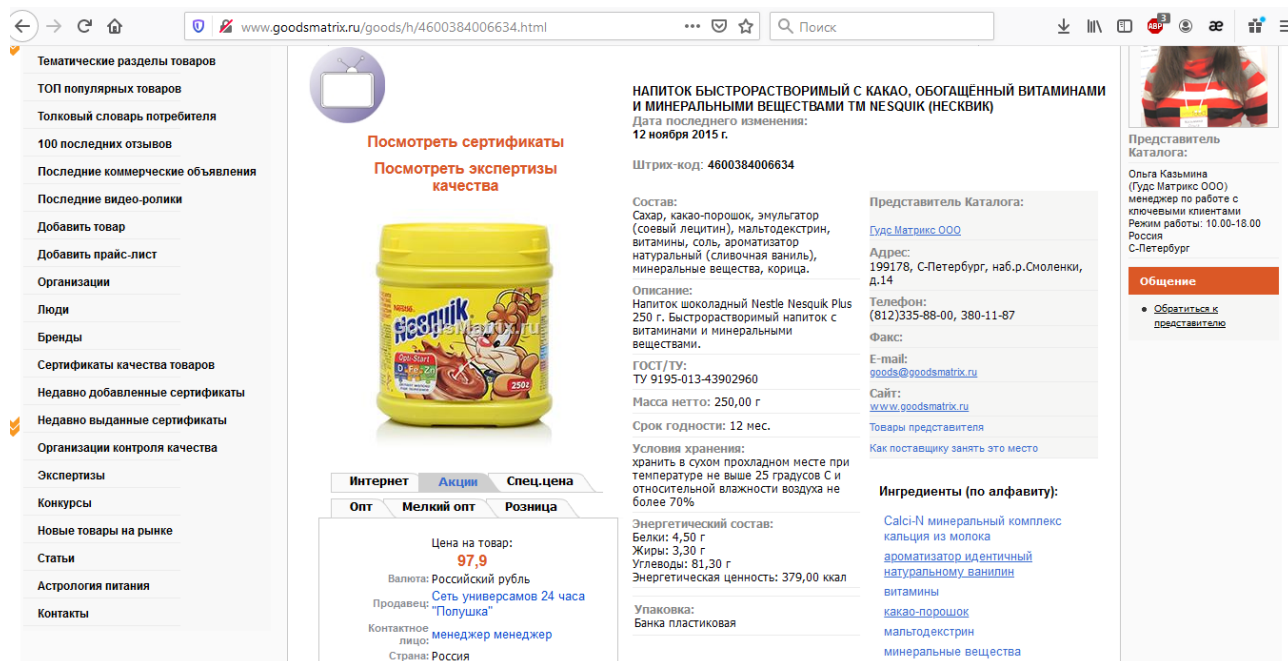


Рисунок 2 – пример страницы с необходимыми данными

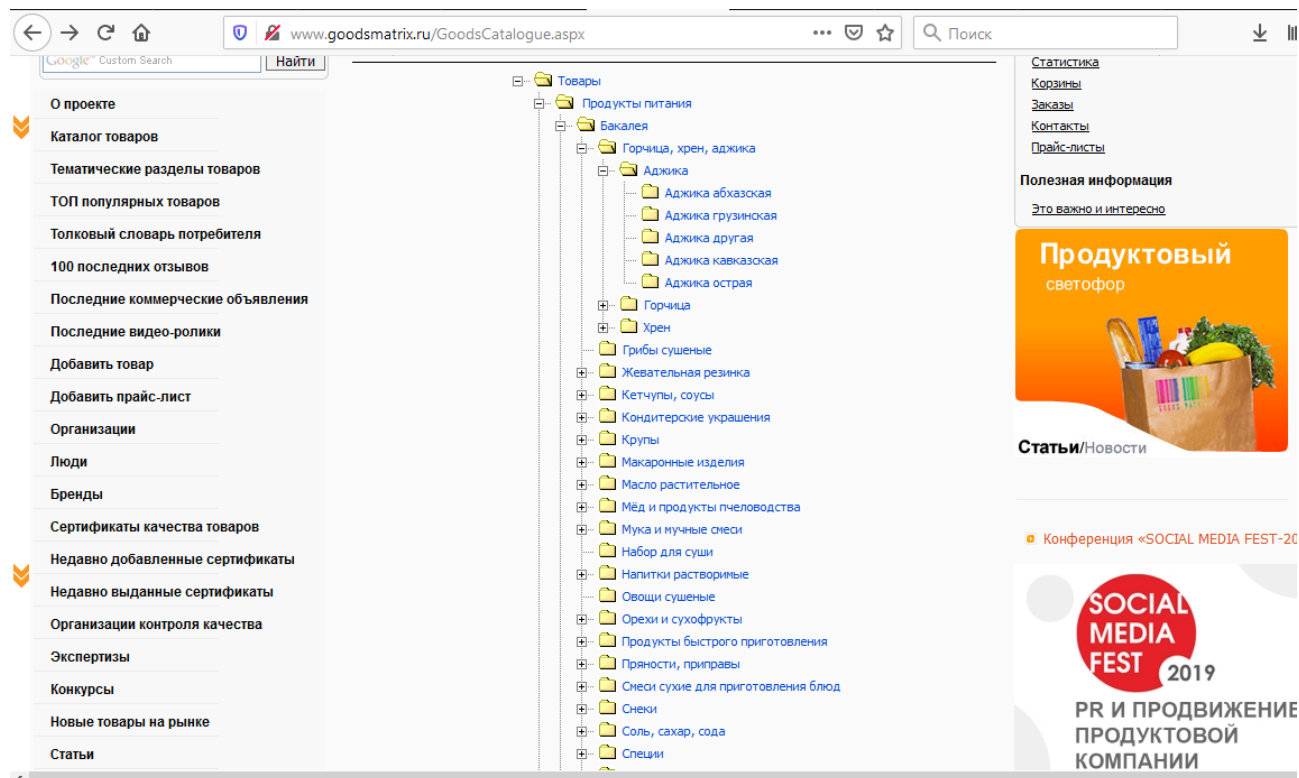


Рисунок 3 – каталог товаров

В процессе дальнейшего исследования ресурса, в правом верхнем углу шапки сайта, обнаружена ссылка обозначенная как «карта каталога». Данная ссылка переводит на страницу, расположенную по адресу <http://www.goodsmatrix.ru/GMMap.aspx>, где в виде списка представлены все категории(ссылки) последнего уровня. Данная страница представлена на рисунке 4. Каждая категория представляет собой ссылку переводящую на страницу с товарами этой категории. Товары в категориях представлены в виде таблицы с тремя столбцами(штрих-код, наименование и производитель). На рисунке 5 можем наблюдать пример таблицы с товарами определенной категории.

|                                   |   |                   |              |               |
|-----------------------------------|---|-------------------|--------------|---------------|
| Тематические разделы товаров      | Группа товаров                                | Штрих-код         | Наименование | Производитель |
| ТОП популярных товаров            | Абразивный, шлифовальный, алмазный инструмент | 1                 |              |               |
| Толковый словарь потребителя      | Абрикосы консервированные                     | Поисковые запросы |              |               |
| 100 последних отзывов             | Абсент  | Поисковый запрос  |              |               |
| Последние коммерческие объявления | Аджика абхазская                              |                   |              |               |
| Последние видеоролики             | Аджика грузинская                             |                   |              |               |
| Добавить товар                    | Аджика другая                                 |                   |              |               |
| Добавить прайс-лист               | Аджика кавказская                             |                   |              |               |
| Организации                       | Аджика острая                                 |                   |              |               |
| Люди                              | Аджика сухая                                  |                   |              |               |
| Бренды                            | Азу готовое замороженное                      |                   |              |               |
| Сертификаты качества товаров      | Айран   |                   |              |               |
| Недавно                           | Аксессуары                                    |                   |              |               |
|                                   | Аксессуары для кальяна                        |                   |              |               |
|                                   | Аксессуары для трубок                         |                   |              |               |
|                                   | Аксессуары сигарные                           |                   |              |               |
|                                   | Ананас сушеный                                |                   |              |               |
|                                   | Ананасы консервированные                      |                   |              |               |
|                                   | Анис  |                   |              |               |
|                                   | Антисептики, пропитки, теплоизоляторы         |                   |              |               |
|                                   | Антистатик                                    |                   |              |               |
|                                   | Анчоус снеки                                  |                   |              |               |
|                                   | Арахис в скорлупе неочищенный                 |                   |              |               |

Рисунок 4 – список категории

|   |   |  |  |  |
|---|---|--|--|--|
| Интернет-Каталог товаров: GoodsMatrix – поиск из 193225 товаров. 100 последних товаров.                         |   |  |  |  |
| Товарная матрица— независимый Интернет-каталог рынка товаров массового спроса для потребителей и профессионалов |   |  |  |  |
| <div> <div>Поиск товара</div> <div>по штрихкоду</div> <div>Найти</div> </div>                                   |   |  |  |  |
| Пример: штрих код - 48005870000842  |   |  |  |  |
| Расширенный поиск   |   |  |  |  |
| О проекте   | Товары / Продукты питания / Фрукты, овощи, грибы / Консервы фруктово-ягодные / Фрукты консервированные / Ананасы консервированные / |  |  |  |
| Каталог товаров   |   |  |  |  |
| Тематические разделы товаров  | Штрих-код   | Наименование   | Производитель  |  |
| ТОП популярных товаров  | 4607080610196   | Ананасы в сиропе колечками тм "Фрау Марта", 565 г                                    | ООО "Эрконпродукт"   |  |
| Толковый словарь потребителя  | 4607080610202   | Ананасы в сиропе кусочками тм "Фрау Марта", 565 г                                    | ООО "Эрконпродукт"   |  |
| 100 последних отзывов   | 8856049002619   | АНАНАСЫ КОЛЬЦАМИ ТМ "КРАСНАЯ ЦЕНА", 580 мл.  | ООО "КП Импорт"  |  |
| Последние коммерческие объявления   | 4811180005824   | Ананасы консервированные ТМ "ЭКОлайн" Кусочки в легком сиропе, стерилизованные, 567г | Изготовитель: Siam Agro-Food Industry Public Co., Ltd. По заказу: A.IBERANDALUS, S.L. С Импорт в РФ: ОДО "Эколайн" групп                           |  |
| Последние видеоролики   | 4690363007035   | Ананас в сиропе, Кусочки ТМ "Каждый день", 580 мл                                    | ООО "СЕМИТЕК"  |  |
| Добавить товар  | 4333465983585   | Кусочки ананаса в сиропе ТМ "Fine Life" (Файн Лайф), 567 г                           | "PT. Great Giant Pineapple"; Импортёр (организация по принятию претензий на территории РФ): ООО "Метро Кэш энд Керри"                              |  |
| Добавить прайс-лист   | 4337182002468   | Кольца ананаса в легком сиропе ТМ "Aro" (Аро), 560 г                                 | "Prime Products Industry Co., Ltd"; Импортёр (организация по принятию претензий на территории РФ)/Изготовлено по заказу: ООО "Метро Кэш энд Керри" |  |
|   | 4337182002420   | Кольца ананаса в легком сиропе ТМ "Aro" (Аро), 820 г                                 | "Prime Products Industry Co., Ltd"; Импортёр (организация по принятию претензий на территории РФ)/Изготовлено по заказу: ООО "Метро Кэш энд Керри" |  |
|   | 4333465848136   | Ломтики ананаса в сиропе ТМ "Select Horeca" (Селект                                  | "PT. Great Giant Pineapple"; Импортёр (организация по принятию претензий на территории РФ): ООО "Метро Кэш энд Керри"                              |  |

Рисунок 5 – таблица с товарами определенной категории

## 1.2 Разработка алгоритма извлечения данных

Опираясь на проведенное исследование в предыдущем параграфе можно сформировать определенный алгоритм извлечения данных:

1. Извлекаем все ссылки(категории товаров) на странице «карта каталога» (группа товаров см. рис. 4).
2. Переходим по первой извлеченной ссылке на страницу с товарами соответствующей категории (см. рис. 5). На этой странице из первого столбца таблицы извлекаем штрих-коды всех товаров этой категории. Повторяем этот шаг для всех извлеченных в первом пункте ссылок (категорий).
3. Составляем ссылку подставляя штрих-код в подготовленный шаблон и получаем адрес страницы с желаемыми данными. Переходим по ссылке и извлекаем необходимые данные со страницы. Повторяем этот шаг для всех извлеченных штрих-кодов во втором пункте.
4. Очищаем полученные данные, преобразуем в json-формат и записываем в файл.

Графическая иллюстрация алгоритма извлечения данных продемонстрирована на рисунке 6.

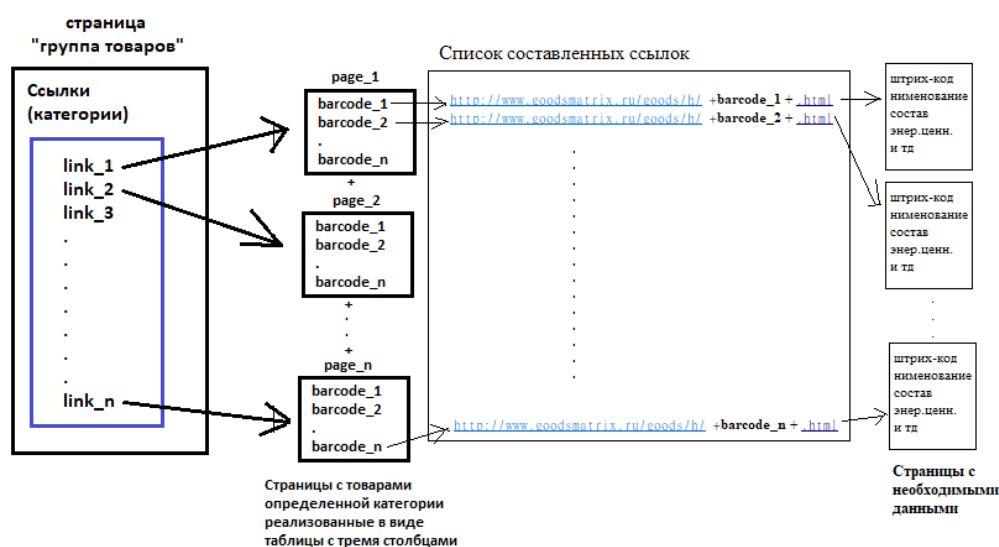


Рисунок 6 – графическая иллюстрация алгоритма извлечения данных

### 1.3 Формат данных

Все извлеченные данные будут иметь строковый тип данных, которые надо преобразовать согласно структуре приведенной ниже (структура объекта Python). В целях демонстрации в скобках указаны также соответствующие типы JSON. Для сериализации объекта Python в JSON будем использовать готовый модуль «json».

```
product = {
    "name": str (string),
    "barcode": str (string),
    "composition": [list<string> (array)],
    "comment": str (string),
    "gost": str (string),
    "net_mass": str (string),
    "keeping_time": str (string),
    "storage_conditions": str (string),
    "esl": {
        "protein": float (number),
        "fats": float (number),
        "carbohydrates": float (number),
        "calorie": float (number),
    }
    "packing_type": str (string)
}
```

В рамках данной работы планируется очистить и преобразовать только строки «состав(composition)» и «энергетическая ценность(esl)», а остальные данные пока оставить без изменений в виде полученных строк.

Состав(composition) получаем в виде такой строки (пример полученный в ходе тестового извлечения данных):

«Мука пшеничная первого сорта, сахар-песок, жир кондитерский, молоко сухое обезжиренное, крахмал, масло растительное, эмульгатор (лецитин), кислота лимонная, соль йодированная, вода питьевая, разрыхлитель (сода пищевая), ароматизатор идентичный натуральному»

Алгоритм для преобразования к необходимому виду строки (composition):



1. Разбиваем полученную строку по символу запятой – ",".
2. Очищаем каждую разбитую строку от пробелов с начала и с конца.
3. Добавляем все полученные строки в массив.

Энергетическая ценность(esl) получаем в виде такой строки (пример полученный в ходе тестового извлечения данных):

«Белки: 6,00 гЖиры: 27,90 гУглеводы: 61,70 гЭнергетическая ценность: 446,40 ккал». Из этой строки нам необходимо получить только числовые данные.

Алгоритм для преобразования к необходимому виду строки (esl):

1. Извлекаем числа из строки используя регулярные выражения. К примеру можно использовать шаблон такой структуры: `r'\d*\.\d+|\d+'`.
2. Преобразуем во float.

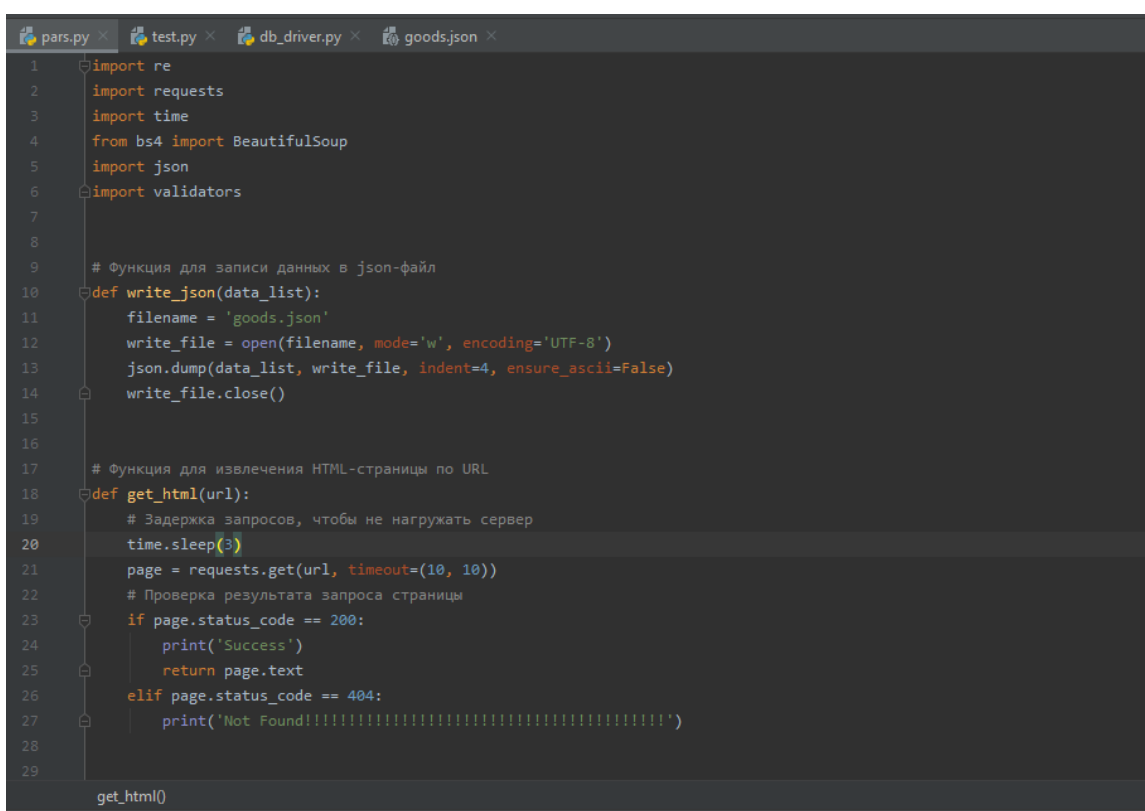
Итоговый пример ожидаемого результата:

```
{
  "name": "ВАФЛИ \"ВТА. ВКУС ЛИМОНА\" ВИТЬБА, 143 Г.",
  "barcode": "4810128003502",
  "composition": [
    "Мука пшеничная первого сорта",
    "сахар-песок",
    "жир кондитерский",
    "молоко сухое обезжиренное",
    "крахмал",
    "масло растительное",
    "эмульгатор (лецитин)",
    "кислота лимонная",
    "соль йодированная",
    "вода питьевая",
    "разрыхлитель (сода пищевая)",
    "ароматизатор идентичный натуральному"
  ],
  "comment": " ",
  "gost": "ТУ РБ 00966671.486-95",
  "net_mass": "143,00 г",
  "keeping_time": " ",
  "storage_conditions": " ",
  "esl": {
    "protein": 6,00,
    "fats": 27,90,
    "carbohydrates": 61,70,
    "calorie": 446,40
  }
  "packing_type": "Флоупак"
}
```

## ГЛАВА 2. РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ИЗВЛЕЧЕНИЯ И ОЧИСТКИ ДАННЫХ

### 2.1 Извлечение ссылок первого уровня

Для начала работы необходимо реализовать функцию для записи данных в JSON-файл(использован модуль json) и функцию для получения HTML страницы по переданному в него URL (использована библиотека requests). Задержка запросов: раз в 3 секунды. Реализованные функции представлены на рисунке 7.



```
1  import re
2  import requests
3  import time
4  from bs4 import BeautifulSoup
5  import json
6  import validators
7
8
9  # Функция для записи данных в json-файл
10 def write_json(data_list):
11     filename = 'goods.json'
12     write_file = open(filename, mode='w', encoding='UTF-8')
13     json.dump(data_list, write_file, indent=4, ensure_ascii=False)
14     write_file.close()
15
16
17 # Функция для извлечения HTML-страницы по URL
18 def get_html(url):
19     # Задержка запросов, чтобы не нагружать сервер
20     time.sleep(3)
21     page = requests.get(url, timeout=(10, 10))
22     # Проверка результата запроса страницы
23     if page.status_code == 200:
24         print('Success')
25         return page.text
26     elif page.status_code == 404:
27         print('Not Found!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!')
28
29 get_html()
```

Рисунок 7 – функции записи данных и получения страницы.

Далее приступаем к алгоритму извлечения данных. Нам необходимо получить ссылки(категории) представленные на рисунке 6, которые находятся на странице по адресу: <http://www.goodsmatrix.ru/GMMap.aspx>. Для удобства обозначим их как ссылки первого уровня(links\_level\_1). Все ссылки на странице расположены в таблице с id="ctl00\_ContentPH\_GroupsDG". На рисунке 8 представлен HTML-код этой страницы. Так как в таблице нет лишних и

ненужных ссылок, мы извлечем все ссылки в этой таблице. Для проверки ссылок используем готовый валидатор(Regular Expression for URL validation). Реализованная функция представлена на рисунке 9. Для работы с полученной HTML – страницей используем библиотеку BeautifulSoup.

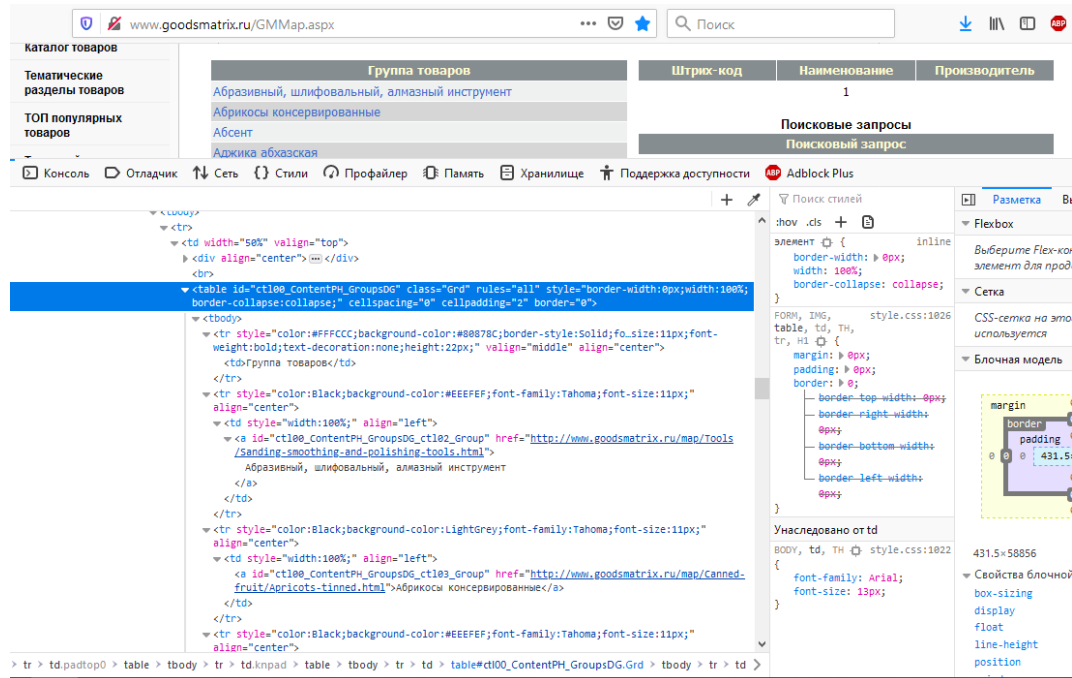


Рисунок 8 – расположение ссылок первого уровня

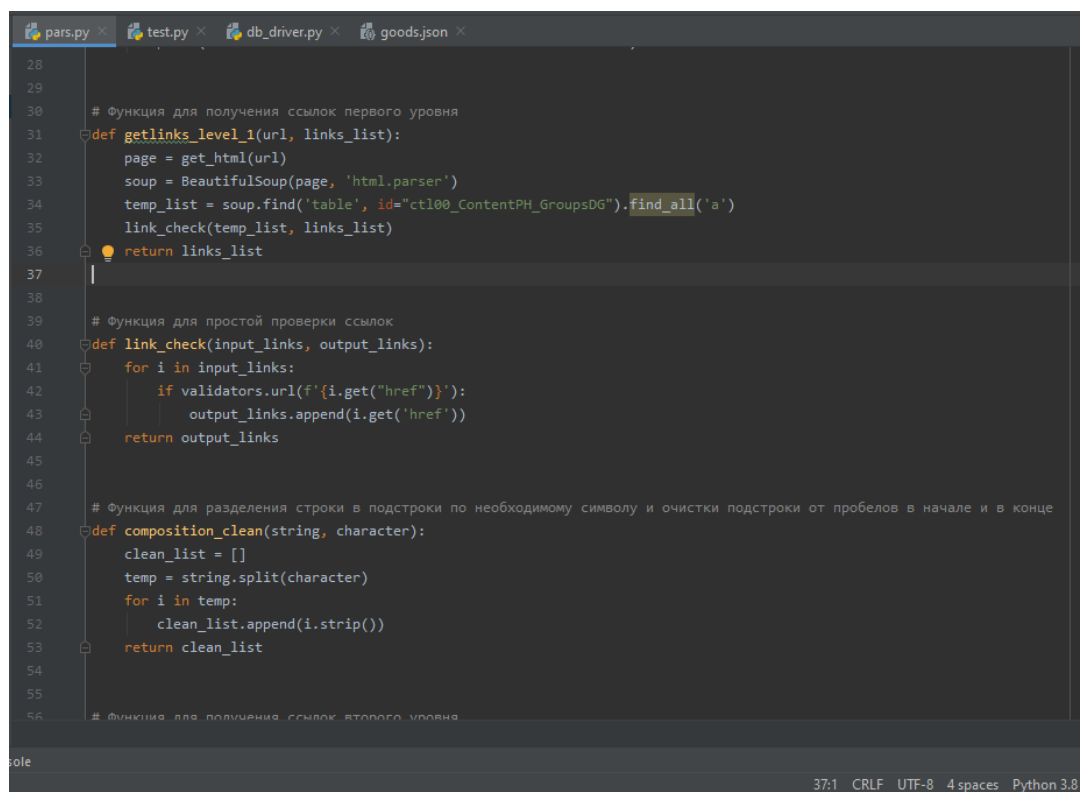


Рисунок 9 – функции извлечения и проверки ссылок

## 2.2 Извлечение ссылок второго уровня

Следующим шагом нам необходимо получить штрих-код всех товаров во всех категориях(все ссылки(категории) получили в предыдущем параграфе). Необходимые нам штрих-коды лежат в таблице с `id="ctl00_ContentPH_GoodsDG"` внутри тега «а». На рисунке 10 выделены желтым цветом, `id` тега «а» для каждого товара уникальный. Штрих-код является ссылкой (обозначим их ссылками второго уровня), но к сожалению по этой ссылке мы не можем попасть к странице с необходимыми нам данными. Поэтому на следующем шаге, мы извлечем только содержание тега «а», для составления необходимой ссылки. На этом этапе, мы пока получим из указанной таблицы все имеющиеся в нем ссылки(тег «а»). Так как таблица состоит из трех столбцов(штрих-код, наименование и производитель), в полученный нами список с каждого товара добавится два лишних элемента. На рисунке 10 обведены красным цветом. Проходим по полученному списку и извлекаем каждый 3 элемент. Таким образом у нас будет список содержащий только необходимые нам ссылки.

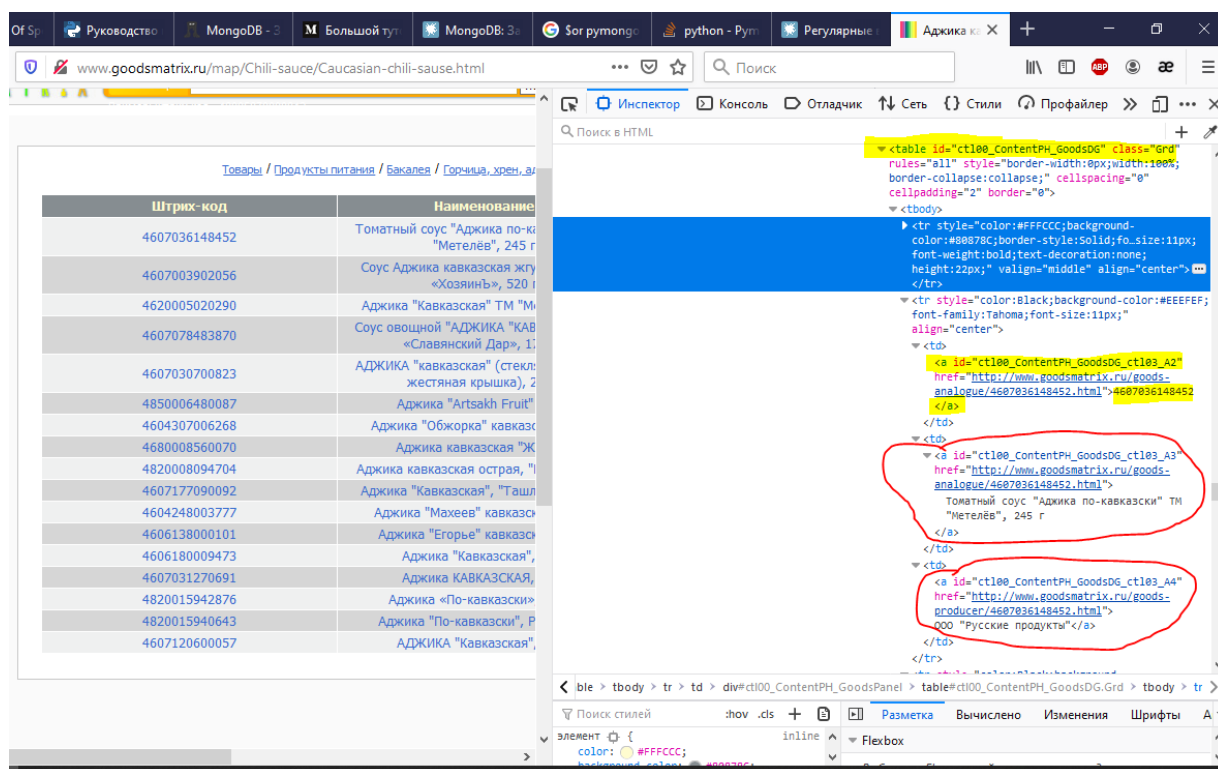
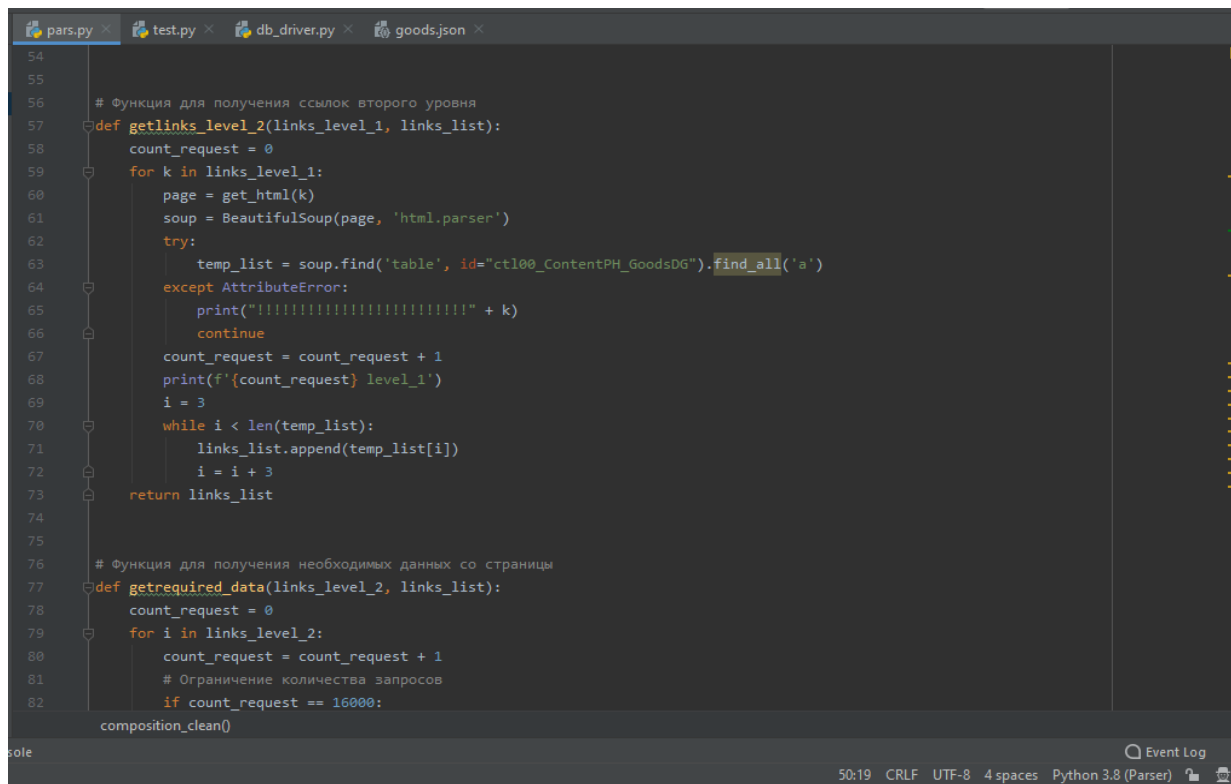


Рисунок 10 – расположение ссылок второго уровня

Реализованная нами функция для извлечения ссылок второго уровня представлена на рисунке 11.



```
54
55
56 # Функция для получения ссылок второго уровня
57 def getlinks_level_2(links_level_1, links_list):
58     count_request = 0
59     for k in links_level_1:
60         page = get_html(k)
61         soup = BeautifulSoup(page, 'html.parser')
62         try:
63             temp_list = soup.find('table', id="ctl00_ContentPH_GoodsDG").find_all('a')
64         except AttributeError:
65             print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!" + k)
66             continue
67         count_request = count_request + 1
68         print(f'{count_request} level_1')
69         i = 3
70         while i < len(temp_list):
71             links_list.append(temp_list[i])
72             i = i + 3
73     return links_list
74
75
76 # Функция для получения необходимых данных со страницы
77 def getrequired_data(links_level_2, links_list):
78     count_request = 0
79     for i in links_level_2:
80         count_request = count_request + 1
81         # Ограничение количества запросов
82         if count_request == 10000:
83             composition_clean()
```

Рисунок 11 – функция для получения ссылок второго уровня

## 2.3 Извлечение и преобразование требуемых данных

При помощи функции реализованной в предыдущем параграфе мы получили список ссылок второго уровня. Из полученного списка извлекаем содержимое определенного тега «a»(представляющий собой штрих-код определенного продукта). Подставляем, в подготовленный шаблон ссылки, штрих-код и переходим по этой ссылке. Извлекаем все необходимые данные из полученной страницы. Повторяем процедуру для все штрих-кодов из полученного списка. Часть реализованной функция для извлечения данных продемонстрирована на рисунках 12,13. Далее преобразуем полученные данные к необходимому виду и записываем в JSON-файл. Функция записи данных в файл представлена на рисунке 7. Пример записанных в JSON-файл данных представлена на рисунке 14.

```
pars.py x test.py x db_driver.py x goods.json x
74
75
76 # Функция для получения необходимых данных со страницы
77 def getrequired_data(links_level_2, links_list):
78     count_request = 0
79     for i in links_level_2:
80         count_request = count_request + 1
81         # Ограничение количества запросов
82         if count_request == 16000:
83             break
84         print(f'{count_request} level_2')
85         url = ('http://www.goodsmatrix.ru/goods/' + i.text.strip() + '.html')
86         if validators.url(f'{url}'):
87             soup = BeautifulSoup(get_html(url), 'html.parser')
88             try:
89                 name = soup.find('span', id="ctl00_ContentPH_GoodsName").text
90             except AttributeError:
91                 name = ''
92             try:
93                 barcode = soup.find('span', id="ctl00_ContentPH_BarCode").text
94             except AttributeError:
95                 barcode = ''
96             try:
97                 composition = soup.find('span', id="ctl00_ContentPH_Composition").text
98             except AttributeError:
99                 composition = ''
100             try:
101                 comment = soup.find('span', id="ctl00_ContentPH_Comment").text
102             except AttributeError:
103                 comment = ''
104         composition_clean()
```

Рисунок 12 – функция извлечения и очистки данных со страницы

```
pars.py x test.py x db_driver.py x goods.json x
135
136     try:
137         fats = clean_esl[1]
138     except IndexError:
139         fats = 0
140     try:
141         carbohydrates = clean_esl[2]
142     except IndexError:
143         carbohydrates = 0
144     try:
145         calorie = clean_esl[3]
146     except IndexError:
147         calorie = 0
148
149     # объект Python который мы передаем в JSON. Представляет собой ассоциативный массив.
150     product = {
151         'name': name,
152         'barcode': barcode,
153         'composition': composition_clean(composition, ","),
154         'comment': comment,
155         'gost': gost,
156         'net_mass': net_mass,
157         'keeping_time': keeping_time,
158         'storage_conditions': storage_conditions,
159         'esl': {
160             'protein': float(protein),
161             'fats': float(fats),
162             'carbohydrates': float(carbohydrates),
163             'calorie': float(calorie)
164         }
165     },
166     composition_clean()
```

Рисунок 13 – функция извлечения и очистки данных со страницы

```
83 {
84     "name": "АБРИКОСЫ ПОЛОВИНКИ ТМ FINE LIFE (ФАЙН ЛАЙФ) С КОЖИЦЕЙ В СИРОПЕ",
85     "barcode": "4607067556592",
86     "composition": [
87         "Абрикосы",
88         "вода",
89         "сахар",
90         "антиокислитель - аскорбиновая кислота."
91     ],
92     "comment": "Абрикосы половинки в сиропе. Стерилизованный продукт.",
93     "eac": "EAC",
94     "net_mass": "1650,00 г",
95     "keeping_time": "36 мес.",
96     "storage_conditions": "Хранить при температуре от 0 до 25 градусов С и относительной влажности воздуха не более 75%",
97     "esl": {
98         "protein": 0.0,
99         "fats": 53.0,
100        "carbohydrates": 0.0,
101        "calorie": 8.0
102    },
103    "packing_type": "Банка стеклянная"
104 },
105 {
106     "name": "АБРИКОСЫ ОЧИЩ. ПОЛОВИНКИ В ЛЕГКОМ СИРОПЕ Ж/Б 850 МЛ 1/12",
107     "barcode": "4008638120011",
108     "composition": [
109         "абрикосы",
110         "вода",
111         "сахар",
112         "регулятор кислотности: лимонная кислота"
113     ]
114 }
```

Рисунок 14 – пример записанных в JSON-файл данных

Все полученные данные были преобразованы и записаны в формате требуемой для JSON-файлов. На рисунках продемонстрированы только важные моменты разработанной программы. Полный листинг реализованного программного обеспечения для извлечения требуемых данных представлен в приложении А.

## 2.4 Запись данных в MongoDB

MongoDB это кросс-платформенная, документоориентированная система управления базами данных, не требующая описания схемы таблиц. Считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных. Каждая БД имеет свой собственный набор файлов в файловой системе. Обычно, один MongoDB сервер имеет несколько БД. Коллекция – это группа документов MongoDB. Является эквивалентом простой таблицы в реляционной базе данных. Коллекция помещена внутри одной БД. Документ в коллекции может иметь различные поля. Документ – это

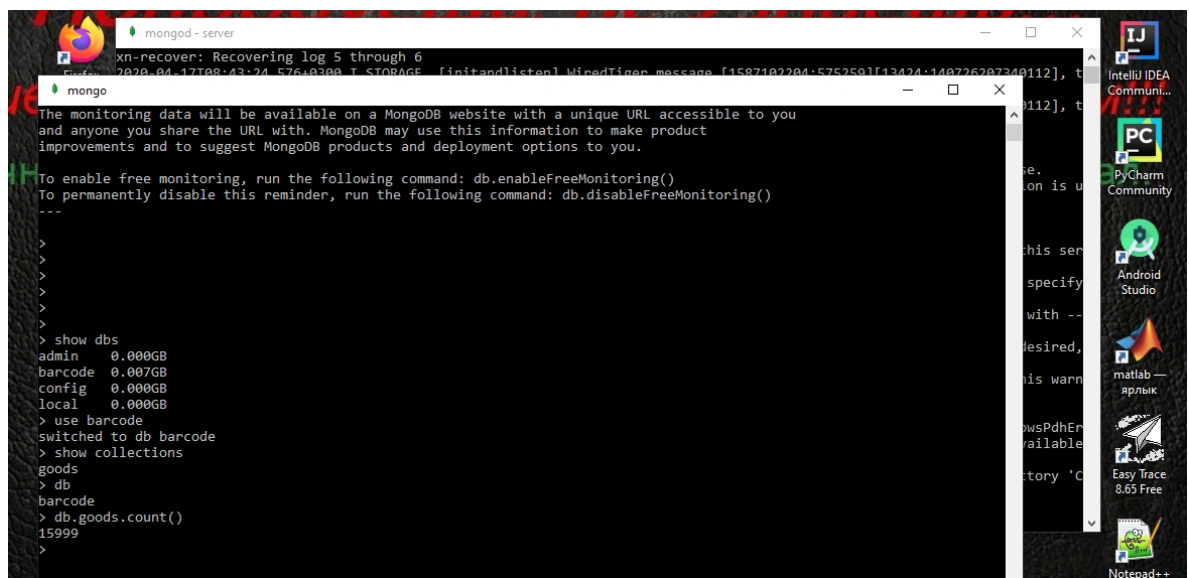
набор пар “ключ – значение”. Документ имеет динамическую схему. Это означает, что документ в одной и той же коллекции не обязан иметь один одинаковый набор полей или структуру, а общие поля в коллекции могут иметь различные типы данных. На рисунке 15 показана разработанная программа для чтения данных из JSON-файла, полученных в предыдущих этапах и добавления их в БД.



```
1
2
3 from pymongo import MongoClient
4 import json
5
6 # Чтение данных из файла
7 filename = 'goods.json'
8 rite_file = open(filename, mode='r', encoding='UTF-8')
9 list_goods = json.load(rite_file)
10 rite_file.close()
11
12 # Создание бд и добавление данных в бд
13 client = MongoClient('localhost', 27017)
14 db = client.barcode
15 col = db.goods
16 result = col.insert_many(list_goods)
17
18 print(result)
```

Рисунок 15 – разработанная программа

На рисунке 16 продемонстрировано количество записей в нашей БД.



```
mongo> show dbs
admin      0.000GB
barcode    0.007GB
config     0.000GB
local      0.000GB
> use barcode
switched to db barcode
> show collections
goods
> db
barcode
> db.goods.count()
15999
>
```

Рисунок 16 – демонстрация количества записей в бд

В результате работы программы в нашу БД(barcode) было сделано 15999 записей. Объем полученных и добавленных в БД данных составил 7 МВ.



## ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы были достигнуты поставленные цели. А именно подготовили данные, требуемые для реализации будущей выпускной квалификационной работы. Все задачи требуемые для успешного выполнения практической работы были решены. При выполнении работы были изучены и применены на практике методы анализа, подготовки и очистки данных. Приобретены навыки работы с не реляционными базами данных, регулярными выражениями, библиотекой BeautifulSoup для извлечения HTML-страниц, библиотекой requests и навыки работы с JSON форматом.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Кайл Бэнкер MongoDB в действии. / Пер. с англ. Слинкина А. А. – М.: ДМК Пресс, 2012. – 394с.
- 2 Сайт «MongoDB Documentation» [электронный ресурс] Режим доступа: <https://docs.mongodb.com/guides/>
- 3 Сайт «Beautiful Soup Documentation» [электронный ресурс] Режим доступа: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
- 4 Сайт «Introducing JSON» [электронный ресурс] Режим доступа: <https://www.json.org/json-en.html>
- 5 Грас Джоэл Data Science. Наука о данных с нуля. / Пер. с англ. Логунов А. В. – М.: БВХ-Петербург, 2020. – 416с.

## ПРИЛОЖЕНИЯ

## Приложение А

## Листинг реализованного программного обеспечения на языке python

1\_parsing\_data.py

```
import re
import requests
import time
from bs4 import BeautifulSoup
import json
import validators
```

### # Функция для записи данных в json-файл

```
def write_json(data_list):
    filename = 'goods.json'
    write_file = open(filename, mode='w', encoding='UTF-8')
    json.dump(data_list, write_file, indent=4, ensure_ascii=False)
    write_file.close()
```

## # Функция для извлечения HTML-страницы по URL

```
def get_html(url):  
    # Задержка запросов, чтобы не нагружать сервер  
    # time.sleep(3)  
    page = requests.get(url, timeout=(10, 10))  
    # Проверка результата запроса страницы  
    if page.status_code == 200:  
        print('Success')  
        return page.text  
    elif page.status_code == 404:  
        print('Not Found!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!')
```

## # Функция для получения ссылок первого уровня

```
def getlinks_level_1(url, links_list):
    page = get_html(url)
    soup = BeautifulSoup(page, 'html.parser')
    temp_list = soup.find('table', id="ctl00_ContentPH_GroupsDG").find_all('a')
    link_check(temp_list, links_list)
    return links_list
```

## # Функция для простой проверки ссылок

```
def link_check(input_links, output_links):
    for i in input_links:
        if validators.url(f'{i.get("href")}'):
            output_links.append(i.get('href'))
```

```

return output_links

# Функция для разделения строки в подстроки по необходимому символу и очистки подстроки
от пробелов в начале и в конце
def composition_clean(string, character):
    clean_list = []
    temp = string.split(character)
    for i in temp:
        clean_list.append(i.strip())
    return clean_list

# Функция для получения ссылок второго уровня
def getlinks_level_2(links_level_1, links_list):
    count_request = 0
    b = 0
    for k in links_level_1:
        page = get_html(k)
        soup = BeautifulSoup(page, 'html.parser')
        try:
            temp_list = soup.find('table', id="ctl00_ContentPH_GoodsDG").find_all('a')
        except AttributeError:
            print("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!" + k)
            continue
        count_request = count_request + 1
        print(f'{count_request} level_1')
        b = b + 1
        if b == 500:
            break
        i = 3
        while i < len(temp_list):
            links_list.append(temp_list[i])
            i = i + 3
    return links_list

# Функция для получения необходимых данных со страницы
def getrequired_data(links_level_2, links_list):
    count_request = 0
    for i in links_level_2:
        count_request = count_request + 1
        # Ограничение количества запросов
        if count_request == 3000:
            break
        print(f'{count_request} level_2')
        url = ('http://www.goodsmatrix.ru/goods/' + i.text.strip() + '.html')
        if validators.url(f'{url}'):
            soup = BeautifulSoup(get_html(url), 'html.parser')

```

```

try:
    name = soup.find('span', id="ctl00_ContentPH_GoodsName").text
except AttributeError:
    name = ''
try:
    barcode = soup.find('span', id="ctl00_ContentPH_BarCodeL").text
except AttributeError:
    barcode = ''
try:
    composition = soup.find('span', id="ctl00_ContentPH_Composition").text
except AttributeError:
    composition = ''
try:
    comment = soup.find('span', id="ctl00_ContentPH_Comment").text
except AttributeError:
    comment = ''
try:
    gost = soup.find('span', id="ctl00_ContentPH_Gost").text
except AttributeError:
    gost = ''
try:
    net_mass = soup.find('span', id="ctl00_ContentPH_Net").text
except AttributeError:
    net_mass = ''
try:
    keeping_time = soup.find('span', id="ctl00_ContentPH_KeepingTime").text
except AttributeError:
    keeping_time = ''
try:
    storage_conditions = soup.find('span',
id="ctl00_ContentPH_StoreCond").text
except AttributeError:
    storage_conditions = ''
try:
    esl = soup.find('span', id="ctl00_ContentPH_ESL").text
except AttributeError:
    esl = ''
try:
    packing_type = soup.find('span', id="ctl00_ContentPH_PackingType").text
except AttributeError:
    packing_type = ''

# Очистка строки энергетическая ценность(esl) и обработка исключения
IndexError
clean_esl = re.findall(r'\d*\.\d+|\d+', esl) #!!!! РЕГУЛЯРНОЕ ВЫРАЖЕНИЕ
НЕПРАВИЛЬНОЕ ГЛАВНОЕ НЕ ЗАБЫТЬ ИСПРАВИТЬ.
try:
    protein = clean_esl[0]
except IndexError:
    protein = 0
try:

```

```

        fats = clean_esl[1]
    except IndexError:
        fats = 0
    try:
        carbohydrates = clean_esl[2]
    except IndexError:
        carbohydrates = 0
    try:
        calorie = clean_esl[3]
    except IndexError:
        calorie = 0

    # объект Python который мы передаем в JSON. Представляет собой
    ассоциативный массив.
    product = {
        'name': name,
        'barcode': barcode,
        'composition': composition_clean(composition, ","),
        'comment': comment,
        'gost': gost,
        'net_mass': net_mass,
        'keeping_time': keeping_time,
        'storage_conditions': storage_conditions,
        'esl': {
            'protein': float(protein),
            'fats': float(fats),
            'carbohydrates': float(carbohydrates),
            'calorie': float(calorie)
        },
        'packing_type': packing_type
    }
    links_list.append(product)
else:
    continue
return links_list

def main():
    base_url = "http://www.goodsmatrix.ru/GMMap.aspx"
    links_level_1 = []
    getlinks_level_1(base_url, links_level_1)
    links_level_2 = []
    getlinks_level_2(links_level_1, links_level_2)
    required_data = []
    getrequired_data(links_level_2, required_data)
    write_json(required_data)

if __name__ == '__main__':
    main()

```

## 2\_add\_data\_bd.py

```
from pymongo import MongoClient
import json

# Чтение данных из файла
filename = 'goods.json'
rite_file = open(filename, mode='r', encoding='UTF-8')
list_goods = json.load(rite_file)
rite_file.close()

# Создание бд и добавление данных в бд
client = MongoClient('localhost', 27017)
db = client.barcode
col = db.goods
result = col.insert_many(list_goods)

print(result)
```