МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра ИБ

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Трекер цен на сайтах

Студенты гр. 0361 Бушуев Д.И.
Иконников С.Н.
Чаплиев С.А.

Преподаватель Спиридонов Р.Е.

Санкт-Петербург
2021

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студенты Бушуев Д.И, Иконников С.Н, Чаплиев С.А.

Группа 0361

Тема работы: Трекер цен на сайтах

Исходные данные:
Написать программу на языке программирования Python, которая будет отслеживать динамику цен товаров на сайте. Требуется защита от бана из-за большого количества запросов. Сохранять данные о ценах в таблицу CSV, XLS

Содержание пояснительной записки:
«Содержание», «Введение», «Теоретические сведения», «Описание методов и функций отображения и сохранения результата», «Примеры работы программы», «Заключение», «Список использованных источников», «Исходный код программы».

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания: 11.11.2021

Дата сдачи курсовой работы: 15.12.2021

Дата защиты курсовой работы: 21.12.2021

Студенты Бушуев Д.И.
Иконников С.Н.
Чаплиев С.А.
_____

Преподаватель _____ Спиридонов Р.Е.

АННОТАЦИЯ

В ходе работы была реализована программа на языке программирования Python с использованием фреймворка Qt, обеспечивающая динамическое отслеживание цен на сайтах. Был реализован графический пользовательский интерфейс для удобной работы с программой.

SUMMARY

In the course of work, a program was implemented in the Python programming language using the Qt framework, which provides dynamic tracking of prices on websites. A graphical user interface has been implemented to make it easy to work with the program.

СОДЕРЖАНИЕ

ОПИСАНИЕ ЗАДАЧИ

Требуется написать программу, которая будет отслеживать динамику цен товаров на сайте.

В программе должна быть возможность работать как по отдельному названию товара, сохраняя цены только для него, так и группами (по категориям, брендам), сохранять между запусками программы какие товары (категории или бренды) нужно мониторить, выводить результаты работы за прошлые дни, по кнопке выполнять запросы с получением актуальных цен, в настройках программы дать возможность выполнять автоматический сбор данных после запуска программы.

Предлагаемые группы товаров - бытовая электроника, сайты mvideo, eldorado, dns-shop, online-trade, citilink, ozon, яндекс.маркет, avito и другие.

ВВЕДЕНИЕ

Целью данной работы является создание трекера цен на языке Python с использованием фреймворка Qt для поиска товаров и записи цен в таблицу.

Для достижения поставленной цели требуется решить следующие задачи:

- Реализация работы по отдельному товару и по группам;
- Реализация вывода результатов работы за прошедшие дни;
- Реализация обхода бана и блокировки по ip у опрашиваемых сайтов;
- Реализация сохранения результатов работы в таблицу CSV.

ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Не попадать под бан и блокировку по IP у опрашиваемых сайтов от большого количества запросов.

Сохранять в таблицу CSV/XLS получаемые данные о ценах.

СТОРОННИЕ ЗАВИСИМОСТИ

В данной реализации задачи курсовой работы были использованы следующие библиотеки:

- requests (для отправки http запросов; для установки необходимо прописать в консоль pip install requests (для ОС Windows 10));
- BeautifulSoup (для поиска необходимой информации на HTML странице; для установки необходимо прописать в консоль pip install beautifulsoup4 (для ОС Windows 10));
- csv (для сохранения результата в таблицу формата csv);
- time (для некоторых внутренних функций);
- datetime (для определения даты парсинга сайта);
- PyQt5 (для создания графического интерфейса; для установки необходимо прописать в консоль pip install pyqt5 (для ОС Windows 10));
- random (для некоторых внутренних функций);
- sys (для некоторых внутренних функций).

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Мониторинг цен — это давно известный бизнес-процесс. Раньше сотрудники розничных компаний заходили в магазины конкурентов и проверяли цены на продукты на каждой полке. Пользуясь своей фотографической памятью (поскольку блокнот и ручка были бы слишком подозрительны), они запоминали цены и отчитывались о них перед своим начальством (кстати, в РФ законодательно не запрещено переписывать и фотографировать цены в магазинах).

Цель такого мониторинга состоит в том, чтобы компании (относительно) своевременно узнавали о структуре цен на рынке. Благодаря этому компании могут удостовериться, что их ценовые стратегии соответствуют рынку.

Разумеется, данные, которые собирали вручную, не обновлялись в реальном времени. К тому времени, когда цены конкурентов принимались во внимание, они уже менялись, или конкуренты организовывали новые маркетинговые акции.

Фактически, существует четыре простые причины, по которым вы должны следить за ценами ваших конкурентов:

- Цены на продукцию влияют на успех рекламы
- Они играют важную роль в решении клиента о покупке
- Цены дают вам преимущество над конкурентами
- Они помогают вам лучше управлять своими запасами и инвентарем

2. ОПИСАНИЕ МЕТОДОВ И ФУНКЦИЙ ОТОБРАЖЕНИЯ И СОХРАНЕНИЯ РЕЗУЛЬТАТА

1. **Метод on_button_1**

Вызывается при нажатии на кнопку «Выбрать сайт» в графическом интерфейсе. Отображает нужную часть графического интерфейса для данного сайта и запускает парсинг имеющихся брендов для сайта ForestHome, если таковой не запускался раньше.

2. **Метод on_button_2**

Вызывается при нажатии на кнопку «Принять» в графическом интерфейсе. Запускает парсинг для выбранного сайта с учетом выбранных фильтров (для сайта ZakaZaka) или парсинг выбранного бренда (для сайта ForestHome), результаты автоматически сохраняются в файл.

3. **Метод on_button_3**

Вызывается при нажатии на кнопку поиска (значок лупы) и запускает парсинг выбранного сайта с учетом введенного запроса и выбранных фильтров (для сайта ZakaZaka), результаты автоматически сохраняются в файл.

4. **Метод save_result**

Вызывается после окончания парсинга, если нашлись товары. Сохраняет результат работы парсера в файл csv формата.

3. ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

На рисунке 1 представлен пользовательский интерфейс программы. Окно содержит выпадающее меню с сайтами для парсинга, кнопку «Выбрать сайт», отвечающую за выбор сайта, поле ввода запроса для поиска по выбранному сайта, кнопку, запускающую поиск по сайту и кнопку «Принять», которая запускает парсинг выбранного сайта.

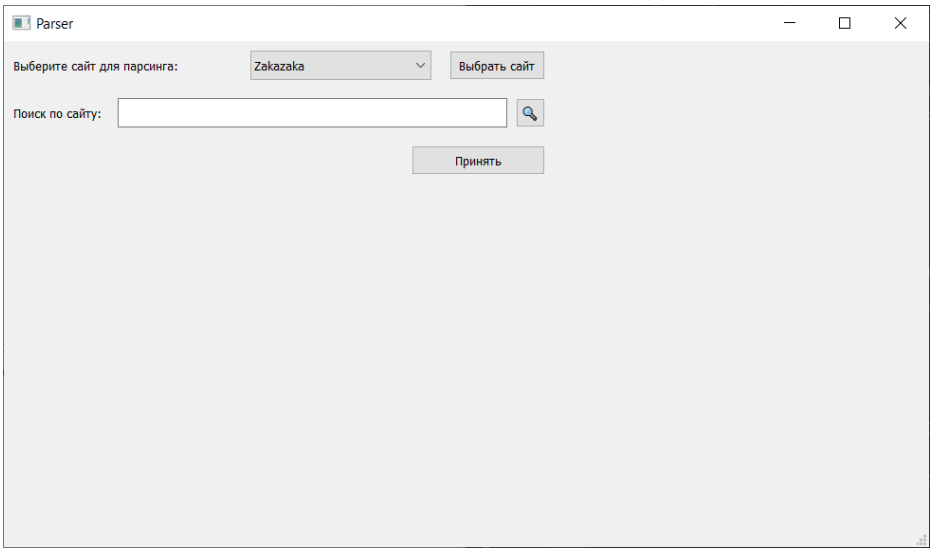

Рисунок 1 – Пользовательский интерфейс программы

На рисунке 2 показано меню с выбором фильтров для сайта Zakazaka, которое появляется после выбора этого сайта.

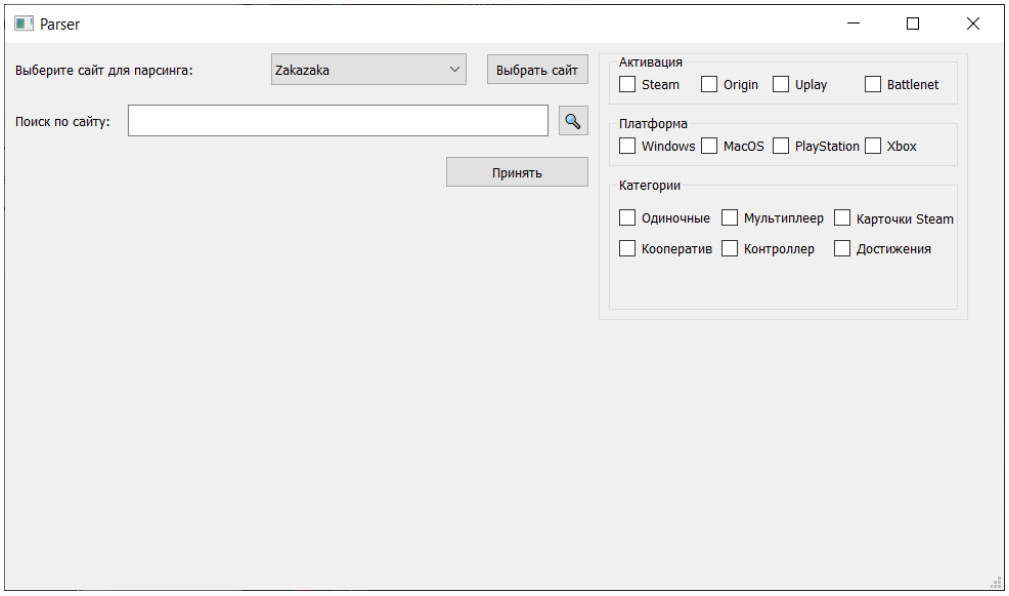

Рисунок 2 – Фильтры для парсинга сайта Zakazaka

На рисунке 3 показан выпадающий список с выбором бренда для парсинга при выборе сайта ForestHome.

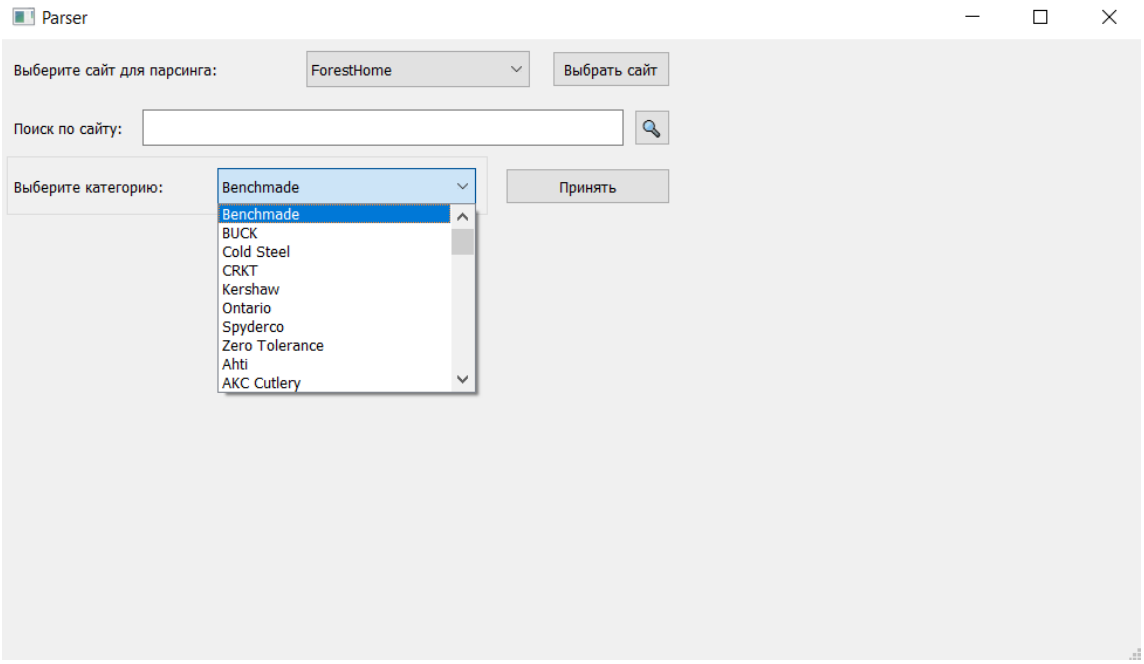

Рисунок 3 – Выбор бренда

На рисунке 4 программа в процессе парсинга сайта Zakazaka по запросу «fallout» с выбранными фильтрами «Steam», «Windows».

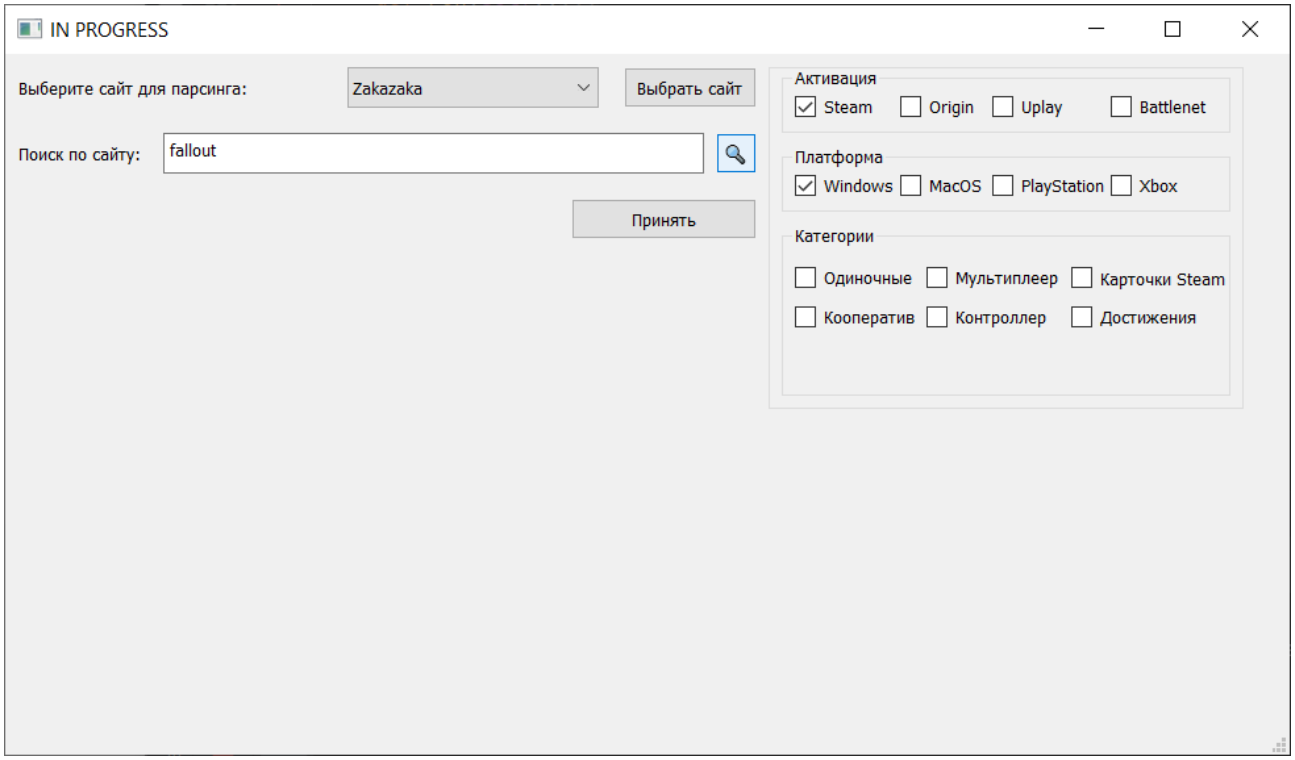

Рисунок 4 – Парсинг сайта по запросу

На рисунке 5 – файл с сохраненным результатом в формате таблицы. Программа сохраняет название продукта, цену (текущую, со скидкой, если такая есть), скиду и дату парсинга.

	A	B	C	D	E
1	Название продукта	Цена	Скидка	Дата	
2	Fallout New Vegas	79 руб	320 руб	20.12.2021	
3	Fallout 4 VR	489 руб	1510 руб	20.12.2021	
4	Fallout 4 GOTY	445 руб	1254 руб	20.12.2021	
5	Fallout Tactics: Brotherhood of Steel	79 руб	220 руб	20.12.2021	
6	Fallout: A Post Nuclear Role Playing Game	79 руб	220 руб	20.12.2021	
7	Fallout 3	89 руб	210 руб	20.12.2021	
8	Fallout 2: A Post Nuclear Role Playing Game	87 руб	212 руб	20.12.2021	
9	Fallout Classic Collection	139 руб	290 руб	20.12.2021	
10	Fallout 4	299 руб	600 руб	20.12.2021	
11	Fallout 3 – Game of the Year Edition	169 руб	330 руб	20.12.2021	
12	Fallout 76: Steel Dawn Deluxe Edition	799 руб	1500 руб	20.12.2021	
13	Fallout New Vegas Ultimate Edition	155 руб	274 руб	20.12.2021	
14	Fallout 4 Season Pass	499 руб	1000 руб	20.12.2021	
15	Fallout 4 Wasteland Workshop	179 руб	280 руб	20.12.2021	
16	Fallout 4 Nuka World	199 руб	300 руб	20.12.2021	
17	Fallout 4 Contraptions Workshop	99 руб	150 руб	20.12.2021	
18	Fallout 4 - Far Harbor	299 руб	400 руб	20.12.2021	
19	Fallout 4 Vault-Tec Workshop	139 руб	110 руб	20.12.2021	
20					

Рисунок 5 – Результат работы парсера

На рисунке 6 показана программа в процессе парсинга сайта ForestHome для бренда BUCK.

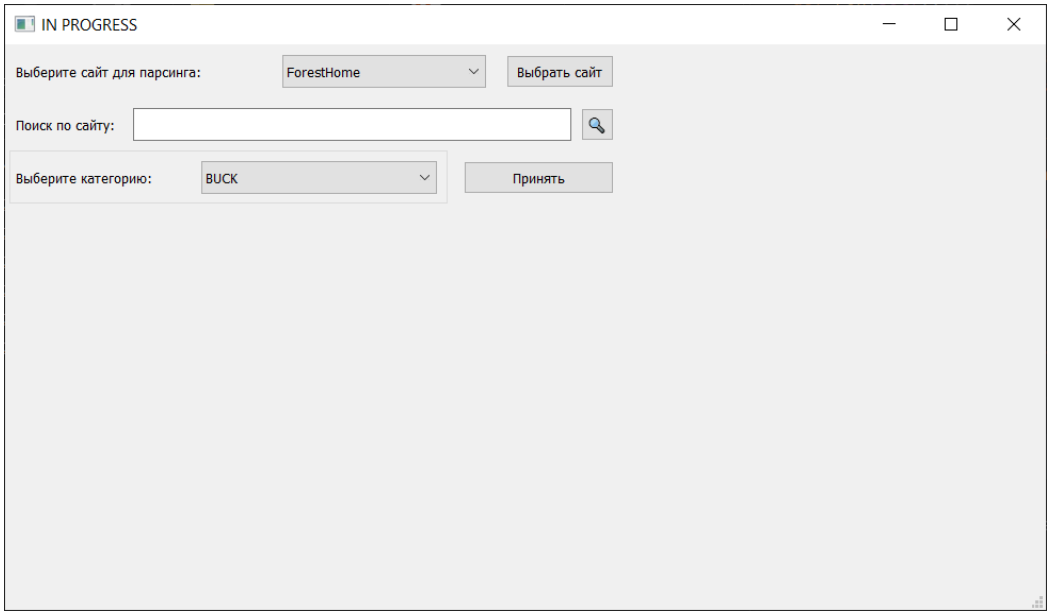

Рисунок 6 – Парсинг бренда BUCK

На рисунке 7 – файл с сохраненным результатом работы парсера для бренда BUCK на сайте ForestHome. Программа сохраняет название товара, цену, информацию о наличии в магазине и дату парсинга для данного товара.

	A	B	C	D	E
1	Название продукта	Цена	Наличие	Дата	
2	Нож BUCK Bantam BHW сталь 420HC рукоять Black GFN (0286BKS)	2 800 ₽	В наличии	20.12.2021	
3	Нож BUCK 110 Folding Hunter сталь 420HC рукоять дерево (0110BRS)	6 910 ₽	В наличии	20.12.2021	
4	Нож BUCK Silver Creek Filet складной филейный 420J2 (0220BLS)	3 950 ₽	В наличии	20.12.2021	
5	Нож BUCK Vantage Pro сталь S30V рукоять G10 (0347BKS)	8 830 ₽	В наличии	20.12.2021	
6	Нож BUCK 113 Ranger Skinner сталь 420HC рук. орех (0113BRS)	7 580 ₽	В наличии	20.12.2021	
7	Нож BUCK Nobelman Carbon сталь 440A (0327CFS)	3 590 ₽	В наличии	20.12.2021	
8	Нож BUCK Nobelman Stainless складной сталь 440A (0327SSS)	3 000 ₽	В наличии	20.12.2021	
9	Нож BUCK Vanguard сталь 420HC рук. Dymondwood (0192BRSDPO1)	10 950 ₽	В наличии	20.12.2021	
10	Нож BUCK Vantage Avid сталь 420HC рукоять DymaLux Red Wood/Nylon (0346RWS)	5 920 ₽	В наличии	20.12.2021	
11	Нож филейный BUCK Silver Creek 6 3/8" Filet сталь 420J2 рукоять GRN/Dynaflex (0223BLS)	3 630 ₽	В наличии	20.12.2021	
12	Нож BUCK Silver Creek 9 5/8" филейный 420J2 рукоять GRN/TPE (0225BLS)	4 110 ₽	В наличии	20.12.2021	
13	Нож BUCK 110 Folding Hunter Finger Grooved сталь 420HC рукоять дерево (0110BRSFG)	7 500 ₽	В наличии	20.12.2021	
14	Нож BUCK Bantam BHW сталь 420HC рукоять Kryptek Highlander Nylon (0286CMS26)	3 230 ₽	В наличии	20.12.2021	
15	Нож BUCK Selkirk +огниво сталь 420HC рукоять Micarta (0836BRS)	7 740 ₽	В наличии	20.12.2021	
16	Нож BUCK Small Selkirk сталь 420HC рукоять Micarta/Steel (0835BRS)	4 660 ₽	В наличии	20.12.2021	
17	Нож BUCK Hunter Fixed Blade сталь 420HC рук. дерево (0101BRS)	7 900 ₽	В наличии	20.12.2021	
18	Нож BUCK 112 Ranger сталь 420HC рукоять дерево (0112BRS)	6 320 ₽	В наличии	20.12.2021	
19	Нож BUCK 110 Folding Hunter LT сталь 420HC рукоять Nylon (0110BKSLT)	3 000 ₽	В наличии	20.12.2021	
20	Нож BUCK 110 Folding Hunter Pro сталь S30V рук. G10 (0110BKSNS1)	10 660 ₽	В наличии	20.12.2021	
21	Нож BUCK BuckLite Max II Small сталь 420HC рукоять Nylon (0684BKS)	2 770 ₽	В наличии	20.12.2021	
22	Нож BUCK Rival III сталь 420HC рук Nylon (0366BKS)	2 920 ₽	В наличии	20.12.2021	
23	Нож BUCK 110 Slim Select сталь 420HC рукоять Red GFN (0110RDS2)	3 470 ₽	В наличии	20.12.2021	
24	Нож BUCK 110 Slim Hunter Pro сталь S30V рукоять Black G10 (0110BKS4)	9 320 ₽	В наличии	20.12.2021	
25	Нож BUCK 110 Slim Hunter Pro сталь S30V рукоять Brown Micarta (0110BRS4)	9 320 ₽	В наличии	20.12.2021	
26	Нож BUCK 110 Slim Hunter Pro сталь S30V рукоять Green Micarta (0110ODS4)	9 320 ₽	В наличии	20.12.2021	
27	Нож BUCK 110 Slim Hunter Select сталь 420HC рукоять Black GFN (0110BKS1)	3 470 ₽	В наличии	20.12.2021	
28	Нож BUCK 110 Slim Hunter Select сталь 420HC рукоять Blue GFN (0110BLS2)	3 470 ₽	В наличии	20.12.2021	

Рисунок 7 – Результат работы парсера для бренда BUCK

ЗАКЛЮЧЕНИЕ

В ходе работы была реализована программа на языке Python с использованием фреймворка Qt для динамического отслеживания цен сайтов. Для взаимодействия с программой был реализован графический интерфейс.

Пользователю доступны для выбора отслеживания цен два сайта: Zaka-zaka и ForestHome. Для обоих сайтов доступен поиск конкретного товара с помощью поисковой строки. Также пользователю доступен поиск товаров по категориям. Данные сохраняются в таблицу формата CSV.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Платформа для предпринимателей и высококвалифицированных специалистов. [Электронный ресурс] – URL: https://vc.ru/services/98642-top-15-servisov-dlya-monitoringa-cen-vashih-konkurentov (дата обращения: 8.12.2021).

2. Документация Beautiful Soup. [Электронный ресурс] – URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc.ru/bs4ru.html (дата обращения: 10.12.2021).

3. Requests: HTTP for Humans. [Электронный ресурс] – URL: https://docs.python-requests.org/en/latest/ (дата обращения: 10.12.2021).

4. Qt for Python - Qt Documentation [Электронный ресурс] – URL: https://doc.qt.io/qtforpython/ (дата обращения: 12.12.2021).

ПРИЛОЖЕНИЕ: КОД ПРОГРАММЫ

```
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import QObject, QThread, pyqtSignal
from PyQt5.QtWidgets import QComboBox
import random
import requests
from bs4 import BeautifulSoup
import csv
import os
import time
from datetime import date

URL_1 = 'https://zaka-zaka.com/search/sort/sale.desc/'
HOST_1 = 'https://zaka-zaka.com'
FILE_1 = "zakazaka.csv"

USER_AGENT_LIST = ['Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:93.0) Gecko/20100101 Firefox/93.0',
                   'Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/96.0.4664.55 Safari/537.36 Edg/96.0.1054.41',
                   'Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/96.0.4664.45 Safari/537.36']

URL_2 = 'https://forest-home.ru/category/brendy/'
HOST_2 = 'https://forest-home.ru'
FILE_2 = "ForestHome.csv"

HEADERS = { 'accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,im
age/avif,image/webp,image/apng,*/*;q=0.8,application/sign
ed-exchange;v=b3;q=0.9',
            'user-agent': 'Mozilla/5.0 (Windows NT 10.0;
Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0'
}
```

```
LIST_BRANDS = []

class Ui_MainWindow(object):
    """
    Класс главного окна
    """

    def setupUi(self, MainWindow):
        """
        Создание главного окна программы
        :param MainWindow:
        :return:
        """
        MainWindow.setObjectName("MainWindow")
        MainWindow.setEnabled(True)
        MainWindow.resize(976, 533)
        self.centralwidget =
QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.comboBox =
QtWidgets.QComboBox(self.centralwidget)
        self.comboBox.setGeometry(QtCore.QRect(260, 10,
191, 31))
        self.comboBox.setObjectName("comboBox")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.pushButton =
QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(470, 10,
101, 31))
        self.pushButton.setObjectName("pushButton")
        self.textEdit =
QtWidgets.QTextEdit(self.centralwidget)
        self.textEdit.setGeometry(QtCore.QRect(120, 60,
411, 31))
        self.textEdit.setObjectName("textEdit")
        self.pushButton_2 =
QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2.setGeometry(QtCore.QRect(430,
110, 141, 31))
```

```
        self.pushButton_2.setObjectName("pushButton_2")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(10, 10, 221,
31))
        self.label.setObjectName("label")
        self.label_2 =
QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(10, 60,
131, 31))
        self.label_2.setObjectName("label_2")
        self.groupBox_4 =
QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_4.setGeometry(QtCore.QRect(580, 10,
361, 261))
        self.groupBox_4.setTitle("")
        self.groupBox_4.setObjectName("groupBox_4")
        self.groupBox_3 =
QtWidgets.QGroupBox(self.groupBox_4)
        self.groupBox_3.setGeometry(QtCore.QRect(10, 120,
341, 131))
        self.groupBox_3.setObjectName("groupBox_3")
        self.checkBox_9 =
QtWidgets.QCheckBox(self.groupBox_3)
        self.checkBox_9.setGeometry(QtCore.QRect(10, 30,
91, 20))
        self.checkBox_9.setObjectName("checkBox_9")
        self.checkBox_10 =
QtWidgets.QCheckBox(self.groupBox_3)
        self.checkBox_10.setGeometry(QtCore.QRect(110,
30, 101, 20))
        self.checkBox_10.setObjectName("checkBox_10")
        self.checkBox_11 =
QtWidgets.QCheckBox(self.groupBox_3)
        self.checkBox_11.setGeometry(QtCore.QRect(220,
30, 121, 21))
        self.checkBox_11.setObjectName("checkBox_11")
        self.checkBox_13 =
QtWidgets.QCheckBox(self.groupBox_3)
        self.checkBox_13.setGeometry(QtCore.QRect(10, 60,
101, 20))
```

```
        self.checkBox_13.setObjectName("checkBox_13")
        self.checkBox_14 =
QtWidgets.QCheckBox(self.groupBox_3)
        self.checkBox_14.setGeometry(QtCore.QRect(110,
60, 101, 20))
        self.checkBox_14.setObjectName("checkBox_14")
        self.checkBox_15 =
QtWidgets.QCheckBox(self.groupBox_3)
        self.checkBox_15.setGeometry(QtCore.QRect(220,
60, 101, 20))
        self.checkBox_15.setObjectName("checkBox_15")
        self.groupBox =
QtWidgets.QGroupBox(self.groupBox_4)
        self.groupBox.setGeometry(QtCore.QRect(10, 0,
341, 51))
        self.groupBox.setObjectName("groupBox")
        self.checkBox =
QtWidgets.QCheckBox(self.groupBox)
        self.checkBox.setGeometry(QtCore.QRect(10, 20,
61, 20))
        self.checkBox.setObjectName("checkBox")
        self.checkBox_2 =
QtWidgets.QCheckBox(self.groupBox)
        self.checkBox_2.setGeometry(QtCore.QRect(90, 20,
61, 20))
        self.checkBox_2.setObjectName("checkBox_2")
        self.checkBox_3 =
QtWidgets.QCheckBox(self.groupBox)
        self.checkBox_3.setGeometry(QtCore.QRect(160, 20,
61, 20))
        self.checkBox_3.setObjectName("checkBox_3")
        self.checkBox_4 =
QtWidgets.QCheckBox(self.groupBox)
        self.checkBox_4.setGeometry(QtCore.QRect(250, 20,
81, 20))
        self.checkBox_4.setObjectName("checkBox_4")
        self.groupBox_2 =
QtWidgets.QGroupBox(self.groupBox_4)
        self.groupBox_2.setGeometry(QtCore.QRect(10, 60,
341, 51))
```

```
        self.groupBox_2.setObjectName("groupBox_2")
        self.checkBox_5 =
QtWidgets.QCheckBox(self.groupBox_2)
        self.checkBox_5.setGeometry(QtCore.QRect(10, 20,
81, 20))
        self.checkBox_5.setObjectName("checkBox_5")
        self.checkBox_6 =
QtWidgets.QCheckBox(self.groupBox_2)
        self.checkBox_6.setGeometry(QtCore.QRect(90, 20,
71, 20))
        self.checkBox_6.setObjectName("checkBox_6")
        self.checkBox_7 =
QtWidgets.QCheckBox(self.groupBox_2)
        self.checkBox_7.setGeometry(QtCore.QRect(160, 20,
91, 20))
        self.checkBox_7.setObjectName("checkBox_7")
        self.checkBox_8 =
QtWidgets.QCheckBox(self.groupBox_2)
        self.checkBox_8.setGeometry(QtCore.QRect(250, 20,
61, 20))
        self.checkBox_8.setObjectName("checkBox_8")
        self.groupBox_5 =
QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_5.setGeometry(QtCore.QRect(4, 100,
411, 51))
        self.groupBox_5.setTitle("")
        self.groupBox_5.setObjectName("groupBox_5")
        self.label_3 = QtWidgets.QLabel(self.groupBox_5)
        self.label_3.setGeometry(QtCore.QRect(6, 10, 161,
31))
        self.label_3.setObjectName("label_3")
        self.comboBox_2 =
QtWidgets.QComboBox(self.groupBox_5)
        self.comboBox_2.setGeometry(QtCore.QRect(180, 10,
221, 31))
        self.comboBox_2.setObjectName("comboBox_2")
        self.pushButton_3 =
QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_3.setGeometry(QtCore.QRect(540,
60, 31, 31))
```

```
        self.pushButton_3.setObjectName("pushButton_3")
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

        self.groupBox_4.hide()
        self.groupBox_5.hide()

    def retranslateUi(self, MainWindow):
        """
        Изменение объектов главного окна
        :param MainWindow:
        :return:
        """
        _translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(_translate("MainWindow",
"Parser"))
        self.comboBox.setItemText(0,
_translate("MainWindow", "Zakazaka"))
        self.comboBox.setItemText(1,
_translate("MainWindow", "ForestHome"))
        self.pushButton.setText(_translate("MainWindow",
"Выбрать сайт"))
        self.textEdit.setHtml(_translate("MainWindow",
                                         "<!DOCTYPE HTML
PUBLIC \"-//W3C//DTD HTML 4.0//EN\"
\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"

"<html><head><meta name=\"qrichtext\" content=\"1\"
/><style type=\"text/css\">\n"
                                         "p, li { white-
space: pre-wrap; }\n"

"</style></head><body style=\" font-family:\'MS Shell Dlg
```

```
2\'; font-size:7.8pt; font-weight:400; font-
style:normal;\">\n"
                                        "<p style=\"-qt-
paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0;
text-indent:0px;\"><br /></p></body></html>"))

self.pushButton_2.setText(_translate("MainWindow",
"Принять"))
        self.label.setText(_translate("MainWindow",
"Выберите сайт для парсинга:"))
        self.label_2.setText(_translate("MainWindow",
"Поиск по сайту:"))
        self.groupBox_3.setTitle(_translate("MainWindow",
"Категории"))
        self.checkBox_9.setText(_translate("MainWindow",
"Одиночные"))
        self.checkBox_10.setText(_translate("MainWindow",
"Мультиплеер"))
        self.checkBox_11.setText(_translate("MainWindow",
"Карточки Steam"))
        self.checkBox_13.setText(_translate("MainWindow",
"Кооператив"))
        self.checkBox_14.setText(_translate("MainWindow",
"Контроллер"))
        self.checkBox_15.setText(_translate("MainWindow",
"Достижения"))
        self.groupBox.setTitle(_translate("MainWindow",
"Активация"))
        self.checkBox.setText(_translate("MainWindow",
"Steam"))
        self.checkBox_2.setText(_translate("MainWindow",
"Origin"))
        self.checkBox_3.setText(_translate("MainWindow",
"Uplay"))
        self.checkBox_4.setText(_translate("MainWindow",
"Battlenet"))
        self.groupBox_2.setTitle(_translate("MainWindow",
"Платформа"))
```

```
        self.checkBox_5.setText(_translate("MainWindow",
"Windows"))
        self.checkBox_6.setText(_translate("MainWindow",
"MacOS"))
        self.checkBox_7.setText(_translate("MainWindow",
"PlayStation"))
        self.checkBox_8.setText(_translate("MainWindow",
"Xbox"))
        self.label_3.setText(_translate("MainWindow",
"Выберите категорию:"))

self.pushButton_3.setText(_translate("MainWindow", "🔍"))

class mainWindow(QtWidgets.QMainWindow):
    """
    Класс главного окна программы
    """
    def __init__(self, parent=None):
        """
        Инициализация объекта класса главного окна
        :param parent:
        """
        QtWidgets.QMainWindow.__init__(self)
        self.gui = Ui_MainWindow()
        self.gui.setupUi(self)
        self.gui.retranslateUi(self)
        self.parsingThread = None



self.gui.pushButton.clicked.connect(self.on_button_1)

self.gui.pushButton_2.clicked.connect(self.on_button_2)

self.gui.pushButton_3.clicked.connect(self.on_button_3)

        self.site = None

    def on_button_1(self):
```

```
        """
        Обработка нажатия кнопки "Выбрать сайт"
        :return:
        """
        self.site = self.gui.comboBox.currentText()
        if self.site == 'Zakazaka':#скрыть элементы
интерфейса для сайта ForestHome, показать элементы для
Zakazaka
            self.gui.groupBox_5.hide()
            self.gui.groupBox_4.show()

        if self.site == 'ForestHome':#скрыть элементы для
Zakazaka, показать элементы для ForestHome, загрузить
бренды, если не загружены
            self.gui.groupBox_4.hide()
            if (LIST_BRANDS == []) and not
self.parsingThread:#Если бренды не были до этого
загружены с сайта, то загрузить
                self.setWindowTitle('IN PROGRESS')
                self.parsingThread =
Parser(mainwindow=self.gui, buttonPressed=1)

self.parsingThread.progressed.connect(self.on_finished)
                self.parsingThread.start()
            elif LIST_BRANDS != [] and LIST_BRANDS !=
None:
                self.gui.groupBox_4.hide()
                self.gui.groupBox_5.show()

    def on_button_2(self):
        """
        Обработка нажатия кнопки "Принять", запускающей
парсинг выбранного сайта, если таковой уже не запущен
        :return:
        """
        if (not self.parsingThread) and self.site !=
None:
            self.setWindowTitle('IN PROGRESS')
```

```
            self.parsingThread =
Parser(mainwindow=self.gui, buttonPressed=2,
site=self.site)

self.parsingThread.progressed.connect(self.on_finished)
            self.parsingThread.start()

    def on_button_3(self):
        """
        Обработка нажатия кнопки поиска по сайту
        :return:
        """
        if (not self.parsingThread) and self.site !=
None:
            self.setWindowTitle('IN PROGRESS')
            self.parsingThread =
Parser(mainwindow=self.gui, buttonPressed=3,
site=self.site)

self.parsingThread.progressed.connect(self.on_finished)
            self.parsingThread.start()

    def on_finished(self, finish):
        """
        Корректное завершение отработанного потока
        :param finish:
        :return:
        """
        if finish == 0:
            self.setWindowTitle('Parser')

self.parsingThread.progressed.disconnect(self.on_finished
)
            self.parsingThread = None
        if finish == 1:
            self.setWindowTitle('ERROR!')

self.parsingThread.progressed.disconnect(self.on_finished
)
            self.parsingThread = None
```

```
class Parser(QThread):
    """
    Класс, обрабатывающий парсинг в отдельном потоке
    """

    progressed = pyqtSignal(int) #сигнал для сообщения с
основным потоком

    def __init__(self, mainwindow, buttonPressed,
site=None):
        """
        Инициализация для объекта класса Parser
        :param mainwindow:
        :param buttonPressed:
        :param site:
        """
        super().__init__()
        self.window = mainwindow
        self.pressedButton = buttonPressed
        self.currentSite = site

    def run(self):
        """
        Обработка логики парсера
        :return:
        """
        global HOST_1
        if self.pressedButton == 1: #если нажата кнопка
"Выбрать сайт" - парсинг брендов для ForestHome
            check = self.get_brands_foresthome()
            if check != None:
                self.progressed.emit(0)
            else:
                self.progressed.emit(1)

        elif self.pressedButton == 2:#если нажата кнопка
"Принять" - парсинг выбранного сайта
            results = []
            if self.currentSite == 'Zakazaka':
```

```
                url = self.zakazaka_get_filters()
                results = self.parse_zakazaka(HOST_1 +
'/search' + url)
            if self.currentSite == 'ForestHome':
                brand =
self.window.comboBox_2.currentData()
                if brand != None:
                    results +=
self.parse_foresthome(brand)
            if results != [] and results != None:
                self.save_result(results)
                self.progressed.emit(0)
            else:
                self.progressed.emit(1)

        elif self.pressedButton == 3: #если нажата кнопка
для поиска по сайту - поиск по сайту
            results = []
            results = self.searching_site()
            if results != [] and results != None:
                self.save_result(results)
                self.progressed.emit(0)
            else:
                self.progressed.emit(1)

    def zakazaka_get_filters(self):
        """
        Сбор выбранных фильтров для парсинга ZakaZaka в
одну ссылку
        :return:
        """
        activate = ['', '', '', '']

        if self.window.checkBox.isChecked():
            activate[0] = 'steam'
        if self.window.checkBox_2.isChecked():
            activate[1] = 'origin'
        if self.window.checkBox_3.isChecked():
            activate[2] = 'uplay'
        if self.window.checkBox_4.isChecked():
```

```
            activate[3] = 'battlenet'
        platform = ['', '', '', '']
        if self.window.checkBox_5.isChecked():
            platform[0] = 'windows'
        if self.window.checkBox_6.isChecked():
            platform[1] = 'mac'
        if self.window.checkBox_7.isChecked():
            platform[2] = 'playstation'
        if self.window.checkBox_8.isChecked():
            platform[3] = 'xbox'
        category = ['', '', '', '', '', '']
        if self.window.checkBox_9.isChecked():
            category[0] = 'single'
        if self.window.checkBox_10.isChecked():
            category[1] = 'multi'
        if self.window.checkBox_11.isChecked():
            category[2] = 'cards'
        if self.window.checkBox_13.isChecked():
            category[3] = 'coop'
        if self.window.checkBox_14.isChecked():
            category[4] = 'pad'
        if self.window.checkBox_15.isChecked():
            category[5] = 'achiv'
        # https://zaka-
zaka.com/search/activation/platform/category/sort/sale.de
sc
        URL = 'https://zaka-zaka.com/search/'
        activation_before = ''
        activation_after = ''
        for i in range(0, 4):
            activation_after += activate[i]
            if (activation_after != activation_before):
                activation_after += '.'
            activation_before = activation_after
        activation_after =
activation_after[:(len(activation_after)) - 1]

        platform_before = ''
        platform_after = ''
        for j in range(0, 4):
```

```
            platform_after += platform[j]
            if (platform_after != platform_before):
                platform_after += '.'
            platform_before = platform_after
        platform_after =
platform_after[:(len(platform_after)) - 1]

        category_before = ''
        category_after = ''
        for j in range(0, 6):
            category_after += category[j]
            if (category_after != category_before):
                category_after += '.'
            category_before = category_after
        category_after =
category_after[:(len(category_after)) - 1]
        url = ''
        if activation_after != '':
            url = '/activation/' + activation_after + '/'
        if platform_after != '':
            url = url + 'platform/' + platform_after +
'/'
        if category_after != '':
            url = url + 'category/' + category_after +
'/'
        return url

    def get_brands_foresthome(self):
        """
        Добавление брендов из ForestHome в выпадающий
список
        :return:
        """
        global LIST_BRANDS
        LIST_BRANDS = self.get_knife_brands()
        if LIST_BRANDS != None and LIST_BRANDS != []:
            for brand in LIST_BRANDS:

self.window.comboBox_2.addItem(brand['name'],
brand['link'])
```

```
self.window.comboBox_2.setCurrentIndex(self.window.comboB
ox_2.count() - 1)
            self.window.groupBox_5.show()
            return 1
        else:
            return None

    def searching_site(self):
        """
        Поиск по сайту и парсинг найденых страниц
        :return:
        """
        global HOST_1
        url = ''
        currentText = self.window.textEdit.toPlainText()
        currentText = currentText.strip()
        currentSite = self.currentSite
        results = []
        if currentSite == 'Zakazaka':
            currentText = currentText.replace(' ', '%20')
            currentFilters = self.zakazaka_get_filters()
            if currentText != '':
                url = '/search/ask/' + currentText +
currentFilters + '/sort/sale.desc/'
                url = url.replace('//', '/')
                url = HOST_1 + url
                results+=self.parse_zakazaka(url)
        if currentSite == 'ForestHome':
            currentText = currentText.replace(' ', '+')
            if currentText != '':
                url = '/search/?page=1&query=' +
currentText
                url = url.replace('//', '/')
                url = HOST_2 + url
                results+=self.parse_foresthome(url)
        if results != None or results != []:
            return results
        else:
            return None
```

```
    def get_content_knives(self, html):
        """
        Парсинг сайта ForestHome
        :param html:
        :return:
        """
        knives = []
        soup = BeautifulSoup(html, 'html.parser')
        if soup.find('div', class_='product') == None:
            return knives
        items = soup.find_all('div', class_='product')
        j = 0
        for item in items:
            name = item.find('div',
class_='product_name').find('a').get('title')
            cost = item.find('span', class_='price')
            exist = item.find('div', class_='stock')
            #print(name, cost.get_text(strip=True),
exist.get_text(strip=True))
            knives.append(
                {
                    'name': name,
                    'cost': cost.get_text(strip=True),
                    'exist': exist.get_text(strip=True)
                }
            )
            j += 1
        if j != 24:
            return knives
        return knives

    def get_knife_brands(self):
        """
        Парсинг брендов из ForestHome
        :return:
        """
        global URL_2
        list_urls = []
        html = self.get_page(URL_2)
```

```
        if html != None:
            soup = BeautifulSoup(html.text,
'html.parser')
            items = soup.find_all('a', class_='btn')
            for item in items:
                temp = item['href']
                if item.get_text(strip=True) ==
'Очистить' or temp == '/category/brendy/':
                    break
                if temp != None:
                    list_urls.append(
                        {
                            'link': temp,
                            'name':
item.get_text(strip=True)
                        }
                    )
            return list_urls
        else:
            return None

    def get_page(self, url, params=None):
        """
        Получение html страницы сайта
        :param url:
        :param params:
        :return:
        """
        HEADERS['user-agent'] =
USER_AGENT_LIST[random.randint(0, 999) % 3]
        try:
            page = requests.get(url, headers=HEADERS,
params=params)
            if page.status_code != 200:
                return None
            else:
                return page

        except (requests.exceptions.Timeout,
requests.exceptions.TooManyRedirects,
```

```
                requests.exceptions.ConnectionError,
requests.exceptions.HTTPError,):
            return None

    def parse_foresthome(self, URL=URL_2):
        """
        Связка воедино функций для парсинга сайта
ForestHome
        :param URL:
        :return:
        """
        global HOST_2, URL_2
        knives = []
        if URL.find('query') == -1:
            URL = HOST_2 + URL + '?page='
        i = 1
        while True:
            results = []
            if URL.find('query') == -1:
                html = self.get_page(URL + str(i))
            if URL.find('query') != -1 and i != 1:
                URL = URL.replace(str(i - 1), str(i))
                html = self.get_page(URL)
            elif i == 1:
                html = self.get_page(URL)
            if html == None:
                return None
            results += self.get_content_knives(html.text)
            i += 1
            if results != []:
                knives += results
            else:
                break
        return knives

    def get_content_zaka(self, html):
        """
        Парсинг сайта ZakaZaka
        :param html:
        :return:
```

```
        """
        soup = BeautifulSoup(html, 'html.parser')
        items = soup.find_all('a', class_='game-block')
        games = []
        for item in items:
            name = item.find('div', class_='game-block-
name')
            cost = item.find('div', class_='game-block-
price')
            sale = item.find('div', class_='game-block-
discount-sum')
            if item == None:
                break
            if sale == None:
                sale = 'Скидки нет'
            else:
                sale = sale.text.replace('-',
'').replace('c', 'руб')
            if name == None or cost == None:
                continue
            name = name.text
            cost = cost.text.replace('c', 'руб')
            games.append(
                {
                    'name': name,
                    'cost': cost,
                    'sale': sale
                }
            )
        return games

    def save_result(self, items):
        """
        Сохранение результата работы парсера в таблицу
        :param items:
        :return:
        """
        global FILE_1, FILE_2
        number = 0
        if self.currentSite == 'Zakazaka':
```

```
            with open(FILE_1, 'a', newline='',
encoding='utf-16') as file:
                writer = csv.writer(file, delimiter='\t')
                writer.writerow(['Название продукта',
'Цена', 'Скидка', 'Дата'])
                for item in items:
                    writer.writerow([item['name'],
item['cost'], item['sale'], date.today()], )
        if self.currentSite == 'ForestHome':
            with open(FILE_2, 'a', newline='',
encoding='utf-16') as file:
                writer = csv.writer(file, delimiter='\t')
                writer.writerow(['Название продукта',
'Цена', 'Наличие', 'Дата'])
                for item in items:
                    writer.writerow([item['name'],
item['cost'], item['exist'], date.today()])

    def pagination(self, html):
        """
        Получение количества страниц сайта ZakaZaka
        :param html:
        :return:
        """
        soup = BeautifulSoup(html, 'html.parser')
        pages = soup.find('span', class_='search-
items').text
        return (int(pages) // 10)

    def parse_zakazaka(self, URL=URL_1):
        """
        Связка воедино функций для парсинга сайта
ZakaZaka
        :param URL:
        :return:
        """
        URL += 'offset/'
        games = []
        html = self.get_page(URL)
        if html == None:
```

```
            return None
        pages = self.pagination(html.text)
        for page in range(0, pages + 1):
            html = self.get_page(URL + str(page * 10))
            # print(html.text)
            games += (self.get_content_zaka(html.text))
        return games

if __name__ == '__main__':
    app = QtWidgets.QApplication([])
    window = mainWindow(Ui_MainWindow)
    window.show()
    sys.exit(app.exec())
```