Приднестровский государственный университет им. Т.Г. Шевченко

Инженерно-технический институт

Инженерно-технический факультет

Кафедра информационных технологий и автоматизированного управления

производственными процессами

**КУРСОВАЯ РАБОТА**

**по дисциплине «Рекомендательные системы»**

**тема: «РАЗРАБОТКА РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ДЛЯ МАГАЗИНА ЭЛЕКТРОНИКИ»**

Направление 09.03.02 «Информационные системы и технологии»

Профиль: «Безопасность информационных систем»

Студент Цветков

группы ИТ19ДР62ИС Степан Вадимович

Научный руководитель, Попукайло

к.т.н., доцент Владимир Сергеевич

Тирасполь, 2021

Приднестровский государственный университет им. Т.Г. Шевченко

Инженерно-технический институт

Инженерно-технический факультет

Кафедра информационных технологий и автоматизированного управления

производственными процессами

Утверждаю

зав. кафедрой ИТиАУПП,

к.т.н., доцент

Ю.А. Столяренко

«\_\_\_\_» \_\_\_\_\_\_ \_\_\_\_\_ 2021 г.

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ ПО ДИСЦИПЛИНЕ**

**«РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ»**

**Студенту** Цветкову Степану Вадимовичу

**Тема работы:** «Разработка рекомендательной системы для магазина электроники»

**Исходные данные к работе:** веб-ресурс интернет-магазина, документация к библиотекам языка Python.

**Перечень подлежащих разработке вопросов:** разработка информационной системы, позволяющей выдавать рекомендации о похожих или наиболее популярных товарах.

**Перечень дополнительных вопросов:** разработка пользовательского интерфейса, обеспечение авторизации/аутентификации.

Научный руководитель, к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_/Попукайло В.С./

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_/Цветков С.В./

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 4

1 ИССЛЕДОВАНИЕ И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5

1.1 Описание поставленной задачи 5

1.2 Современное состояние исследуемой задачи 5

1.3 Обзор методов решения подобных задач 5

1.4 Постановка задачи, системные требования, требования к входным данным и выходным формам 7

2 ПРОЕКТИРОВАНИЕ СТРУКТУРЫ И АРХИТЕКТУРЫ

РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ 9

2.1 Выбор методов и средств для реализации, его обоснование (описание

выбранной БД, фреймворков, библиотек) 9

2.2 Структура, архитектура программного продукта 11

2.3 Обоснование структуры базы данных 11

2.4 Обоснование взаимодействия частей ИС 12

3 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ 14

3.1 Описание алгоритма сбора данных 14

3.2 Описание реализации алгоритма рекомендаций 20

3.3 Описание пользовательского интерфейса 22

ВЫВОД 24

СПИСОК ЛИТЕРАТУРЫ 25

ПРИЛОЖЕНИЕ А – Руководство пользователя 26

**Введение**

В ассортименте современных магазинов находится большое разнообразие товаров и услуг того или иного рода, однажды найдя интересный вам товар или услугу вы вероятно захотите посмотреть и похожие на них предложения, но ввиду бесчисленного разнообразия можете и не наткнуться на похожие товары и услуги, именно для этого и разрабатываются рекомендательные системы которые призваны решать такого рода проблемы. Цель создания рекомендательных систем это предложение пользователю наиболее популярных или же похожих вариантов просмотренного или заказанного вами контента будь то игра фильм одежда музыка или игра рекомендательная система приняв в расчёт максимально возможное количество факторов предоставит вам максимально похожий контент который вы просматривали и как было сказано ранее в условиях огромного рынка интернет магазинов и сервисов предоставления услуг необходимость рекомендации на всё большем количестве онлайн магазинов товаров и услуг становится всё более актуальной. В данной работе будет разработана рекомендательная система ля одного из таких сайтов, а именно для магазина электроники Cifrus, который не обладает своей рекомендательной системой. По окончанию работы планируется получит сайт способный подобрать товары максимально похожие на те которые выбрал пользователь.

**1 Исследование и анализ предметной области**

**1.1 Описание поставленной задачи**

В ходе выполнения курсового проекта необходимо разработать программный продукт в виде рекомендательной системы для онлайн магазина электроники Cifrus*.* Так как у данного онлайн магазина отсутствует встроенная система рекомендация похожих товаров и введу большого многообразия представленных товаров в данном магазине рекомендательная система это необходимость для более удобного пользования онлайн магазином пользователю, так как он находясь в поиске одной вещи благодаря подобной системе рекомендаций товаров может найти для себя не один товар.

**1.2 Современное состояние исследуемой задачи**

В век цифровых технологий огромное количество различных магазинов и организаций, предоставляющих те или иные услуги, перебираются на просторы интернета для привлечения большего количества клиентов. Крупные магазины и компании предоставления различных услуг делают всё возможное что бы потенциальному клиенту было как можно проще получит и заметить необходимую ему информацию, товар или услугу одним из таких способов является система рекомендаций которая показывает пользователю наиболее популярное содержание сайта-сервиса или же наиболее похожие товары, услуги и информационные материалы которые были просмотрены пользователем, но подобные системы рекомендации товаров и услуг встроены далеко не во все сайты-сервисы именно по этой причине появились интегрируемые рекомендательные системы.

**1.3 Обзор методов решения подобных задач**

Интегрируемые рекомендательные системы создаются для внедрения их в проекты которые не имеют встроенного функционала рекомендаций одной из таких систем является продукт под названием *InOne* Турецкой *IT* компании Insider продукт данной *IT* компании предоставляет своим клиентам возможность onsite персонализации, поведенческой кастомизации, товарных рекомендаций, *email* рассылок, *mobile web* персонализации, *mobile app* персонализации, умных мобильных *push*-уведомлений и *InApp* уведомлений.

Кроме того, в отличие от большинства других решений, *Insider* умеет предсказывать поведение посетителей. Компания декларирует, что с помощью новой разработки *Predictive Ad Audience* умеет выявлять клиентов, которые скорее всего совершат покупку в ближайшие 7 дней. И коммуницировать с ними во всех каналах, показывая при этом наивысший процент конверсии. Данный функционал полностью обеспечит все потребности покупателя магазина и максимально упростит его выбор товара предлагая его любым удобным пользователю способом и упростит продажу или предоставление услуг сервисам и магазином которые пользуются платформой *InOne.*

Данное решение не привязано ни к каким *CMS* и интегрируется с любыми сайтами с помощью нескольких строк *JS* кода. С любыми внешними системами интегрируется через *API*. Что позволяет с лёгкостью интегрировать данный продукт, так же он имеет пробную версию, во время которой потенциальный покупатель интеграции рекомендательной системы может решить подходит ему данная платформа или же нет.

Ещё одним подобным рекомендательным сервисом является платформа *Kameleoon* одноимённой Французской *IT* компании. Данная система изучает поведение пользователя на сайте, его действия, контекст, источники, из которых он пришел, сопоставляет информацию с данными из *CRM* и сегментирует посетителей по 40 критериям. Основываясь на полученной информации, система адаптирует содержание страниц сайта в соответствии с ожиданиями и предпочтениями посетителей*. Kameleoon* также может в реальном времени выдавать триггерные сообщения (индивидуальные предложения, напоминания и т.д.), рассылать персонализированные письма.

Алгоритмы предиктивного таргетинга работают с отдельными товарами или товарными категориями, выбирая из аудитории сайта тех посетителей, которые приобретут это товары с наибольшей вероятностью. Способ интегрирования так же с помощью *JS* кода, что обеспечивает лёгкое интегрирование на любой сайт сервис или же интернет-магазин.

Ввиду «сильной» рекомендательной способности вышеперечисленных платформ они высоко ценимы и востребованы среди крупных онлайн магазинов и сайтов сервисов численность товаров и услуг которых превышает несколько тысяч, но и для совсем небольших одностраничных сайтов продажи услуг и товаров есть свои крайне немногочисленные интегрируемые рекомендательные системы. Онлайн инструмент *Yagla* Российской *IT* компании является одним из таких.

Данный инструмент подходит для персонализации контента лэндингов или небольших сайтов, где не предусмотрены отдельные страницы для разных поисковых запросов и рекламных объявлений.

Система определяет, откуда пришел посетитель - с какой площадки, по какому поисковому запросу. На основе этой информации контент страницы максимально подстраивается под интересы пользователя. В зависимости от настроек заголовка выводится поисковая фраза, меняются изображения, текст, кнопки. Это позволяет с первых секунд максимально попасть в потребности посетителя и перевести его в разряд покупателей. Он так же интегрируется путём JS кода.

Учитывая малое количество сервисов, предоставляющих рекомендательные услуги исключительно в виде предоставления пользователю рекомендаций похожих товаров и услуг на посещаемом интернет-магазине или сайте-сервисе, и выполняется данная работа.

**1.4 Постановка задачи, системные требования, требования к входным данным и выходным формам**

В ходе выполнения работы будет разработана рекомендательная система, которая предоставляет пользователю похожие варианты товаров интернет-магазина электроники Cifrus в зависимости от того какой товар просматривает пользователь. Данная рекомендательная система будет реализована в виде одностраничного сайта, где пользователю будет необходимо ввести лишь выбранный *artikul* товара и получить схожие результаты выборки.

Список необходимых библиотек для работоспособности рекомендательной системы:asgiref==3.5.0  
beautifulsoup4==4.10.0  
certifi==2021.10.8  
charset-normalizer==2.0.12  
colorama==0.4.4  
django-extensions==3.1.5  
Django==4.0.3  
idna==3.3  
numpy==1.22.2  
pandas==1.4.1  
psycopg2-binary==2.9.3  
python-dateutil==2.8.2  
pytz==2021.3  
requests==2.27.1  
setuptools==60.9.3  
six==1.16.0  
soupsieve==2.3.1  
sqlparse==0.4.2  
tqdm==4.63.0 ,tzdata==2021.5  
urllib3==1.26.8

**2 ПРОЕКТИРОВАНИЕ СТРУКТУРЫ И АРХИТЕКТУРЫ**

**РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ**

**2.1 Выбор методов и средств для реализации, его обоснование (описание**

**выбранной БД, фреймворков, библиотек)**

В качестве базы данных для хранения списка товаров и всей необходимой информации была выбрана файловая база данных PostgreSQL*.* PostgreSQL — это популярная свободная объектно-реляционная система управления базами данных. PostgreSQL базируется на языке SQL и поддерживает многочисленные возможности. Функции в PostgreSQL являются блоками кода, исполняемыми на сервере, а не на клиенте БД. Хотя они могут писаться на чистом SQL, реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки собственно SQL и требует использования некоторых языковых расширений. Функции могут писаться с использованием различных языков программирования. PostgreSQL допускает использование функций, возвращающих набор записей, который далее можно использовать так же, как и результат выполнения обычного запроса. Функции могут выполняться как с правами их создателя, так и с правами текущего пользователя. Иногда функции отождествляются с хранимыми процедурами, однако между этими понятиями есть различие.

В разработке простых сайтов PostgreSQL используется несколько реже, чем MySQL / MariaDB, но всё же эта пара с заметным отрывом опережает по частоте использования остальные системы управления базами данных. При этом в разработке сложных сайтов и веб-приложений PostgreSQL опережает по использованию MySQL и MariaDB. Большинство фреймворков (например, Ruby on Rails, Yii, Symfony, Django) поддерживают использование PostgreSQL в разработке.

Для работы с базой данных, связи с сервером и передачей информации между backend и frontend составляющих рекомендательной системы был выбран *python* фреймворк *dgango. Django (Джанго)* — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation. Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself). Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений. Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

Для корректной работы фреймворка *Dgango* необходима *python* библиотека *django-extensions, Django. Django Extensions* — это набор пользовательских расширений для Django Framework.

До недавнего времени в *Python* отсутствовал минимальный низкоуровневый интерфейс сервера / приложения для фреймворков *asyncio*. Спецификация *OSGI* восполняет этот пробел и означает, что теперь мы можем приступить к созданию общего набора инструментов, который можно использовать во всех фреймворках *asyncio.*

*ASGI* должен помочь создать экосистему веб-фреймворков *Python*, которые будут очень конкурентоспособны по сравнению с *Node* и *Go* с точки зрения достижения высокой пропускной способности в контекстах, связанных с вводом-выводом. Он также обеспечивает поддержку *HTTP /2* и *WebSockets*, которые не могут быть обработаны *WSGI.*

В настоящее время *Uvicorn* поддерживает *HTTP/1.1* и *WebSockets*. Планируется поддержка *HTTP/2.*

Для заполнения базы данных информацией используются две python библиотеки requests и *BeautifulSoup,* предназначенные для парсинга информации.

Библиотека *requests* является стандартным инструментом для составления HTTP-запросов в *Python.* Простой и аккуратный *API* значительно облегчает трудоемкий процесс создания запросов. Таким образом, можно сосредоточиться на взаимодействии со службами и использовании данных в приложении.

*BeautifulSoup4 (bs4)* — это библиотека *Python* для извлечения данных из файлов *HTML* и *XML.* Для естественной навигации, поиска и изменения дерева *HTML*, модуль *BeautifulSoup4*, по умолчанию использует встроенный в *Python* парсер *html.parser. BS4* так же поддерживает ряд сторонних парсеров *Python*, таких как *lxml, html5lib* и *xml* (для разбора *XML*-документов).*.*

**2.2 Структура, архитектура программного продукта**

Рекомендательная система будет представлять собой сайт, написанный с помощью языка гипертекстовой разметки *HTML* и применения стилей *css.*

Программная часть, отвечающая за подбор рекомендаций для пользователя, будет выполнена на языке программирования *python*, для вывода результатов работы кода подбора рекомендаций будет использоваться набор функций язык программирования *javascript - jQuery*

Для связи данных между частью кода отвечавшую за рекомендации, и частью отвечающую за вывод полученной информации отвечает *python* фреймворк *dgango.*

**2.3 Обоснование структуры базы данных**

В данном проекте база данных хранит в себе информацию о товарах интернет-магазина, следовательно, состоит из таблиц:

*Category* внутри которой созданы два столбца: *id (integer), name(text).*

Столбец *id(integer)* предназначен для хранения нoмера товара, который необходим для определения товара в рекомендательной системе, этот параметр необходим для поиска сходственных товаров.

Столбец *name(text)* предназначен для хранения имени категории товара и отвечает за визуальную часть рекомендательной системы.

*Sub\_Category* внутри которой созданы столбцы:

Столбец *id(integer)* предназначен для хранения нoмера товара, который необходим для определения товара в рекомендательной системе, этот параметр необходим для поиска сходственных товаров.

Столбец *category(text)* предназначен для хранения имени категории товара и отвечает за визуальную часть рекомендательной системы.

Столбец *name(text)* предназначен для хранения имени подкатегории товара и отвечает за визуальную часть рекомендательной системы.

Столбец *url(text)* ссылка на товар.

*Products* внутри которой созданы столбцы:

Столбец *id(integer)* предназначен для хранения нoмера товара, который необходим для определения товара в рекомендательной системе, этот параметр необходим для поиска сходственных товаров.

Столбец *category(text)* предназначен для хранения имени категории товара и отвечает за визуальную часть рекомендательной системы.

Столбец *name(text)* предназначен для хранения имени товара и отвечает за визуальную часть рекомендательной системы.

Столбец *subcategory(text)* предназначен для хранения имени подкатегории товара и отвечает за визуальную часть рекомендательной системы.

Столбец *code(integer)* предназначен для хранения внутреннего нoмера товара.

Столбец *price(text)* предназначен для храненияцены товара.

Столбец *available(text)* предназначен для хранения информации о доступности товара.

Столбец *specifications(text)*характеристики товара.

Столбец *img* ссылка на картинку товара.

**2.4 Обоснование взаимодействия частей ИС**

Сайт, через который будет осуществляться коммуникация будет написан на язык гипертекстовой разметки *HTML* в купе с применением стилей *css.*

Язык *HTML* — это основа *web* сайтов, с его помощью создается каркас страницы, которую вы видите в браузере.

Если сравнивать страницу сайта и обычную бумажную книгу, то на сайте, как и в книге, есть абзацы и заголовки. В книге есть название всей книги (по сути самый главный заголовок), есть названия глав, параграфов в этих главах и так далее.

Язык *CSS* расширяет возможности языка *HTML*. Он позволяет менять цвета, шрифты, фон, в общем заниматься красотой сайта. А *HTML*, соответственно, отвечает за структуру сайта.

Для написания основы рекомендательной системы был выбран язык *python*. Высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности данный язык обладает огромным количеством встраиваемых библиотек различного характера что делает его невероятно гибким, функциональным и в то же время простым в использовании и позволяет экономить большое количество времени благодаря уже написанным методам и алгоритмам.

В данном случае *python* был выбран из-за наличия такого фреймворка как *fastapi* который позволяет легко передавать и получать информацию с *http* сервера, которым является разрабатываемая система.

Для согласования и вывода проанализированной информации используются функции *jQuery.* Набор функций *JavaScript*, фокусирующийся на взаимодействии *JavaScript* и *HTML*. Библиотека *jQuery* помогает легко получать доступ к любому элементу *DOM*, обращаться к атрибутам и содержимому элементов *DOM*, манипулировать ими. Также библиотека *jQuery* предоставляет удобный *API* для работы с *AJAX.*

**3 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ**

**3.1 Описание алгоритма сбора данных**

Прежде чем разрабатывать способ рекомендации товаров эти сами товары необходимо занести в базу данных. Для этого необходимо написать парсер *html* страниц.

В качестве языка программирования для данной задачи был выбран *python* и две библиотеки requests и *BeautifulSoup* методы которых позволяют без особых проблем выполнить задачу сбора информации.

Разберём код по порядку.

import random  
import time  
  
import requests  
from bs4 import BeautifulSoup  
from tqdm import tqdm  
  
from main.models import Category, Subcategory, Products  
  
BASE\_URL = r'https://www.cifrus.ru'  
  
  
def get\_random\_user\_agent():  
 fake\_user\_agents = [  
 'Opera/8.32 (Windows NT 4.0; en-US) Presto/2.12.231 Version/12.00',  
 'Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; Trident/4.0)',  
 'Mozilla/5.0 (X11; Linux i686) AppleWebKit/5331 (KHTML, like Gecko) Chrome/36.0.893.0 Mobile Safari/5331',  
 'Mozilla/5.0 (iPad; CPU OS 7\_1\_1 like Mac OS X; en-US) AppleWebKit/535.39.4 (KHTML, like Gecko) Version/3.0.5 Mobile/8B119 Safari/6535.39.4',  
 'Mozilla/5.0 (iPad; CPU OS 7\_0\_2 like Mac OS X; en-US) AppleWebKit/535.32.5 (KHTML, like Gecko) Version/4.0.5 Mobile/8B116 Safari/6535.32.5',  
 'Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/5.1)',  
 'Mozilla/5.0 (compatible; MSIE 11.0; Windows 98; Trident/3.0)',  
 'Mozilla/5.0 (X11; Linux i686; rv:6.0) Gecko/20130821 Firefox/35.0'  
 ]  
 return random.choice(fake\_user\_agents)  
  
  
def get\_soup(url: str = None):  
 try:  
 response = requests.get(url, headers={'User-Agent': get\_random\_user\_agent()})  
 soup = BeautifulSoup(response.text, features="html.parser")  
 if soup.select\_one('title').text == '404 Not Found':  
 raise requests.HTTPError  
 return soup  
 except requests.HTTPError:  
 return None  
 except Exception:  
 time.sleep(3)  
 get\_soup(url)  
  
  
def add\_categories(url: str) -> None:  
 soup = get\_soup(url)  
 categories = soup.select('ul.category-dropdown>li.dropdown-submenu')  
 for category in categories:  
 category\_name = category.select\_one('a.dropdown-toggle').text  
 category\_obj = Category(name=category\_name)  
 if category\_obj.name not in ['Услуги', 'Платный ремонт', 'Мобильные телефоны']:  
 try:  
 category\_obj.save()  
 except:  
 pass  
 for subcategory in category.select('div.dropdown-menu div.dropdown-inner ul li a'):  
 subcategory\_obj = Subcategory(category\_id=category\_obj.id,  
 name=subcategory.text,  
 url=f"{BASE\_URL}{subcategory['href']}")  
 try:  
 subcategory\_obj.save()  
 except:  
 pass  
  
  
def get\_product\_data(url: str) -> dict:  
 product\_data = {}  
 soup = get\_soup(url)  
 try:  
 product\_data['code'] = int(soup.select\_one('span.code').text)  
 except:  
 product\_data['code'] = None  
  
 try:  
 product\_data['price'] = float(soup.select\_one('div.price>div.new-price').text.split()[0])  
 except:  
 product\_data['price'] = None  
  
 try:  
 product\_data['available'] = True if soup.select\_one('span.instock').text == 'есть в наличии' else False  
 except:  
 product\_data['available'] = None  
  
 try:  
 product\_data['specifications'] = [specification.text for specification in soup.select('table.excel3 tbody td')]  
 except:  
 product\_data['specifications'] = None  
  
 try:  
 product\_data['img'] = BASE\_URL + soup.select\_one('a.highslide')['href']  
 except:  
 product\_data['img'] = None  
  
 return product\_data  
  
  
def add\_subcategory\_products(url: str, category: int, subcategory: int) -> None:  
 counter = 1  
 soup = get\_soup(f'{url}/page-{counter}')  
 while soup is not None:  
 product\_urls = soup.select('div.caption>div.name>a')  
 for product\_url in product\_urls:  
 product\_data = get\_product\_data(url=f"{BASE\_URL}{product\_url['href']}")  
 new\_product = Products(category=category,  
 subcategory=subcategory,  
 code=product\_data['code'],  
 name=product\_url.text,  
 price=product\_data['price'],  
 available=product\_data['available'],  
 specifications=product\_data['specifications'],  
 img=product\_data['img'])  
 try:  
 new\_product.save()  
 except:  
 pass  
 counter += 1  
 soup = get\_soup(f'{url}/page-{counter}')  
  
  
def run():  
 add\_categories(BASE\_URL)  
 subcategories = Subcategory.objects.all()  
 for subcategory in tqdm(subcategories):  
 add\_subcategory\_products(url=subcategory.url,  
 category=subcategory.category,  
 subcategory=subcategory)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 run()

листинг 1 – код функции *main*

Внутри данной функции происходят вызов функций, которые собирают информацию с сайта, вывод информации о состоянии процесса сбора информации, возможные ошибки и проверки соединения с интернетом.

Разберём подробней некоторые фрагменты кода.

def get\_random\_user\_agent():  
 fake\_user\_agents = [  
 'Opera/8.32 (Windows NT 4.0; en-US) Presto/2.12.231 Version/12.00',  
 'Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; Trident/4.0)',  
 'Mozilla/5.0 (X11; Linux i686) AppleWebKit/5331 (KHTML, like Gecko) Chrome/36.0.893.0 Mobile Safari/5331',  
 'Mozilla/5.0 (iPad; CPU OS 7\_1\_1 like Mac OS X; en-US) AppleWebKit/535.39.4 (KHTML, like Gecko) Version/3.0.5 Mobile/8B119 Safari/6535.39.4',  
 'Mozilla/5.0 (iPad; CPU OS 7\_0\_2 like Mac OS X; en-US) AppleWebKit/535.32.5 (KHTML, like Gecko) Version/4.0.5 Mobile/8B116 Safari/6535.32.5',  
 'Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/5.1)',  
 'Mozilla/5.0 (compatible; MSIE 11.0; Windows 98; Trident/3.0)',  
 'Mozilla/5.0 (X11; Linux i686; rv:6.0) Gecko/20130821 Firefox/35.0'  
 ]  
 return random.choice(fake\_user\_agents)

листинг 1.1 – создание ложных адресов.

Данный фрагмент предотвращает возможную блокировку процесса парсинга, сайтом, из-за слишком частых запросов.

def get\_soup(url: str = None):  
 try:  
 response = requests.get(url, headers={'User-Agent': get\_random\_user\_agent()})  
 soup = BeautifulSoup(response.text, features="html.parser")  
 if soup.select\_one('title').text == '404 Not Found':  
 raise requests.HTTPError  
 return soup  
 except requests.HTTPError:  
 return None  
 except Exception:  
 time.sleep(3)  
 get\_soup(url)

листинг 1.2 –Получение данных с сайта.

Данный фрагмент получает полный код сайта.

def add\_categories(url: str) -> None:  
 soup = get\_soup(url)  
 categories = soup.select('ul.category-dropdown>li.dropdown-submenu')  
 for category in categories:  
 category\_name = category.select\_one('a.dropdown-toggle').text  
 category\_obj = Category(name=category\_name)  
 if category\_obj.name not in ['Услуги', 'Платный ремонт', 'Мобильные телефоны']:  
 try:  
 category\_obj.save()  
 except:  
 pass  
 for subcategory in category.select('div.dropdown-menu div.dropdown-inner ul li a'):  
 subcategory\_obj = Subcategory(category\_id=category\_obj.id,  
 name=subcategory.text,  
 url=f"{BASE\_URL}{subcategory['href']}")  
 try:  
 subcategory\_obj.save()  
 except:  
 pass  
  
  
def get\_product\_data(url: str) -> dict:  
 product\_data = {}  
 soup = get\_soup(url)  
 try:  
 product\_data['code'] = int(soup.select\_one('span.code').text)  
 except:  
 product\_data['code'] = None  
  
 try:  
 product\_data['price'] = float(soup.select\_one('div.price>div.new-price').text.split()[0])  
 except:  
 product\_data['price'] = None  
  
 try:  
 product\_data['available'] = True if soup.select\_one('span.instock').text == 'есть в наличии' else False  
 except:  
 product\_data['available'] = None  
  
 try:  
 product\_data['specifications'] = [specification.text for specification in soup.select('table.excel3 tbody td')]  
 except:  
 product\_data['specifications'] = None  
  
 try:  
 product\_data['img'] = BASE\_URL + soup.select\_one('a.highslide')['href']  
 except:  
 product\_data['img'] = None  
  
 return product\_data  
  
  
def add\_subcategory\_products(url: str, category: int, subcategory: int) -> None:  
 counter = 1  
 soup = get\_soup(f'{url}/page-{counter}')  
 while soup is not None:  
 product\_urls = soup.select('div.caption>div.name>a')  
 for product\_url in product\_urls:  
 product\_data = get\_product\_data(url=f"{BASE\_URL}{product\_url['href']}")  
 new\_product = Products(category=category,  
 subcategory=subcategory,  
 code=product\_data['code'],  
 name=product\_url.text,  
 price=product\_data['price'],  
 available=product\_data['available'],  
 specifications=product\_data['specifications'],  
 img=product\_data['img'])  
 try:  
 new\_product.save()  
 except:  
 pass  
 counter += 1  
 soup = get\_soup(f'{url}/page-{counter}')

листинг 1.3 –Ищем в коде нужную информацию.

Данный фрагмент получает категории, подкатегории и товары.

def run():  
 add\_categories(BASE\_URL)  
 subcategories = Subcategory.objects.all()  
 for subcategory in tqdm(subcategories):  
 add\_subcategory\_products(url=subcategory.url,  
 category=subcategory.category,  
 subcategory=subcategory)

листинг 1.4 –Ищем в коде нужную информацию.

Данный фрагмент добавляет данные в базу.

**3.2 Описание реализации алгоритма рекомендаций**

Алгоритм реализующий подбор похожих товаров реализован на языке *python* с использованием фреймворка *dgango.*

from main.models import Category, Subcategory, Products  
  
  
def list\_to\_dict(lst: list) -> dict:  
 dictionary = {lst[i]: lst[i + 1] for i in range(0, len(lst), 2)}  
 return dictionary  
  
  
def run(product: Products) -> list[Products]:  
 product\_specs = list\_to\_dict(product.specifications)  
 products = Products.objects.filter(category=product.category).exclude(subcategory=product.subcategory)  
 recommended\_products = dict()  
 for recommended\_product in products:  
 similarities = 0  
 recommended\_product\_specs = list\_to\_dict(recommended\_product.specifications)  
 for spec in product\_specs.keys():  
 if spec in recommended\_product\_specs.keys() and product\_specs[spec] == recommended\_product\_specs[spec]:  
 similarities += 1  
 recommended\_products[recommended\_product.id] = similarities  
 recommended\_products = sorted(recommended\_products.items(), key=lambda x: int(x[1]), reverse=True)[:6]  
 return [product[0] for product in recommended\_products]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 run()

**3.3 Описание пользовательского интерфейса**

Пользовательский интерфейс представляет собой *HTML* страницу

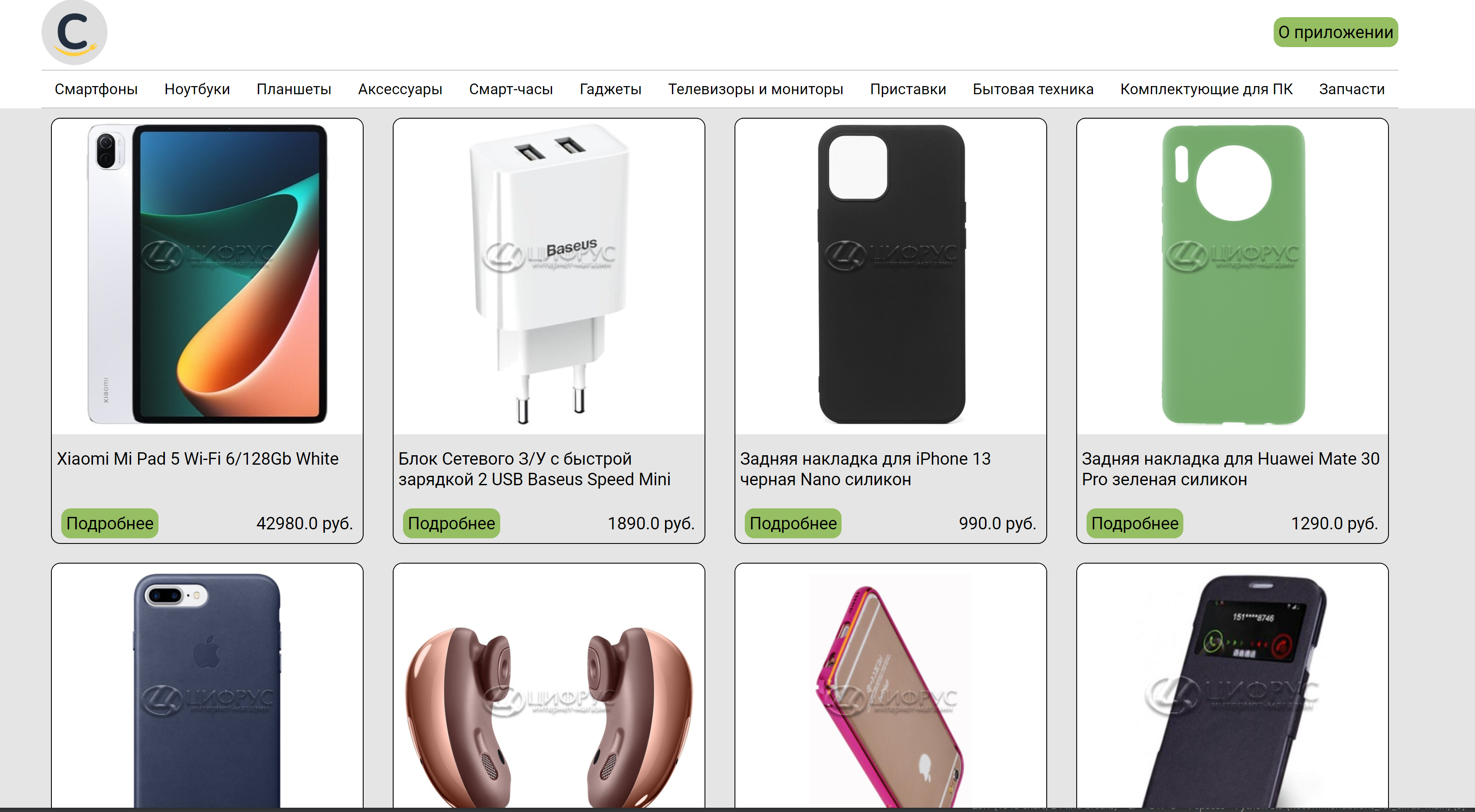


Рисунок 1 главная страница

В нижней части которой представлена ссылка на магазин *Cifrus* при нажатии на которую откроется официальная страница магазина, где можно посмотреть товары.

В верхней части можно выбрать категорию, товары из которой мы хотим увидеть.

В главной части расположены контейнеры с товарами выбранной нами категории, при нажатии на кнопку «Подробнее» мы увидим следующую страницу.

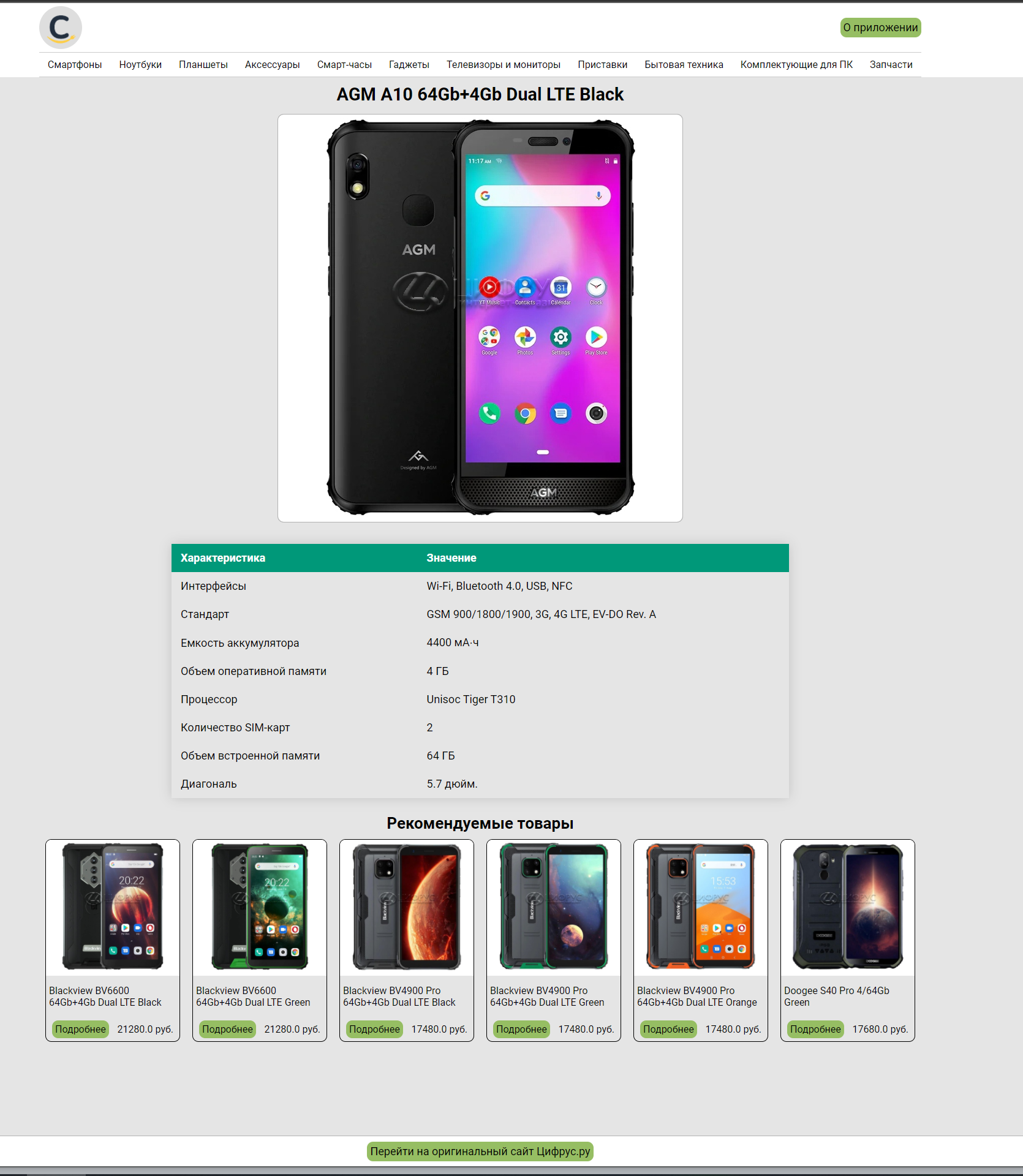


Рисунок 2 страница товара

В нижней части которой представлена ссылка на магазин *Cifrus* при нажатии на которую откроется официальная страница магазина, где можно посмотреть товары.

В верхней части можно выбрать категорию, товары из которой мы хотим увидеть.

В главной части мы видим более подробную информацию о выбранном товаре, а так же рекомендации на товары, которые могут нас заинтересовать.

**Вывод**

В ходе выполнения курсовой работы была разработана рекомендательная система интернет магазина электроники Cifrus основой которой является язык программирования *python*, на данном языке был написан алгоритм подбора похожих товаров и парсер данных который собрал всю информацию о товаре и занёс её в файловую базу данных *postgres*, пользовательский интерфейс был оформлен в виде *HTML* страницы, вывод обработанной алгоритмом информации о похожих товарах был реализован с помощью языка программирования *JavaScript*.

**Список литературы**

1. <https://fastapi.tiangolo.com/ru/>
2. <https://www.crummy.com/software/BeautifulSoup/bs4/doc.ru/bs4ru.html>
3. <https://pythonru.com/biblioteki/kratkoe-rukovodstvo-po-biblioteke-python-requests>

**Приложение**

Для начала работы сайта рекомендаций необходимо выполнить следующие действия:

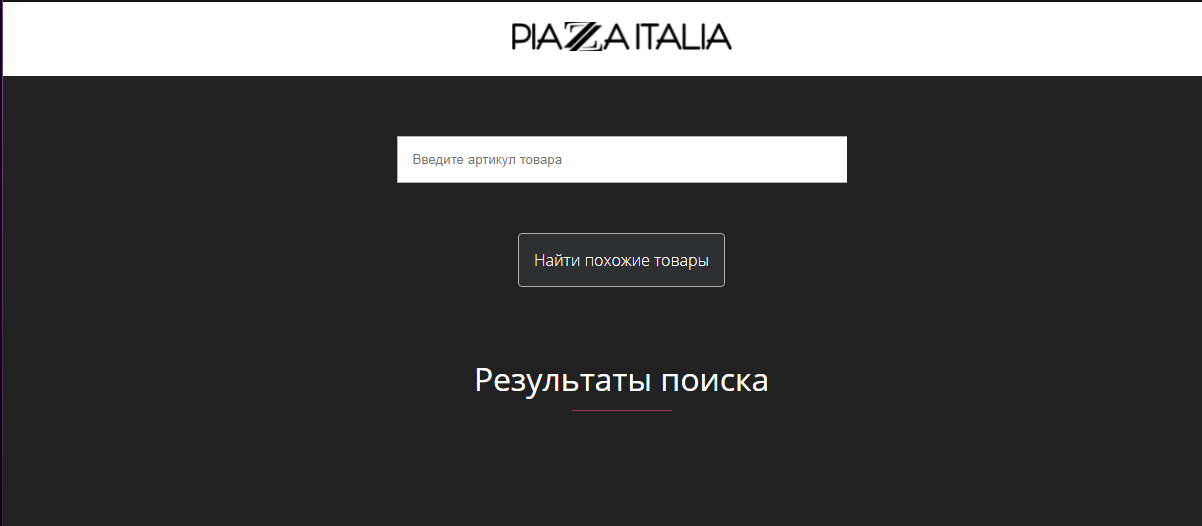
1.Создаём и подключаем виртуальное окружение

* *python -m venv your\_virtual\_environment\_name*
* *For Linux:*
  + *source your\_virtual\_environment\_name/bin/activate*
* *For Windows:*
  + *your\_virtual\_environment\_name\Scripts\activate.bat*

2. Устанавливаем зависимости: *pip install -r requirements*

3. Запускаем проект *uvicorn simple-api:app –reload*

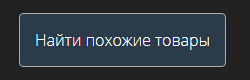
После чего открыть файл *index.html*

Перед вами откроется следующая страница 

Где в поле необходимо ввести artikul товара



И нажать, но кнопку «найти похожие товары»



После чего вы увидите товар, который вы выбрали, количество похожих товаров и их изображения ссылки

