|  |  |
| --- | --- |
|  | |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  Федеральное государственное автономное образовательное учреждение высшего образования  **«Дальневосточный федеральный университет»**  (ДВФУ) | |
|  | |
| **ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ (ШКОЛА)** | |
| **Департамент математического и компьютерного моделирования** | |
| **КУРСОВОЙ ПРОЕКТ** | |
| по дисциплине «Программирование баз данных» | |
| на тему «Создание реляционной базы данных на основе парсинга веб-сайтов (магазины линолеумов)» | |
| по образовательной программе подготовки бакалавров | |
| по направлению 09.03.03 «Прикладная информатика» | |
| профиль Прикладная информатика в компьютерном дизайне | |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Выполнил студент группы  № Б9122-09.03.03пикд  \_\_\_\_\_\_\_\_\_\_\_\_\_ Сакмаркин В. А.  (подпись)  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024г. |
| Регистрационный номер \_\_\_\_\_\_\_  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024г. | Руководитель: старший преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_\_ Селезнев Т. Э.  (подпись)  «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024г. |
| г. Владивосток  2024 | |

# **Оглавление**

[Введение 3](#_Toc169540412)

[Изучение и выбор методов реализации 4](#_Toc169540413)

[1.1 Обзор существующих методов решения 4](#_Toc169540414)

[1.2 Обоснование выбранных методов и средств реализации 5](#_Toc169540415)

[1.3 Выбор подходящих веб-сайтов и их анализ 5](#_Toc169540416)

[1.4 Возможные проблемы и способы их решения 8](#_Toc169540417)

[Реализация проекта на основе выбранных методов 9](#_Toc169540418)

[2.1 Разработка алгоритмов парсинга 9](#_Toc169540419)

[2.2 Создание базы данных и внесение полученных данных 12](#_Toc169540420)

[2.3 Разработка веб-сайта 16](#_Toc169540421)

[Заключение 19](#_Toc169540422)

[Список литературы 20](#_Toc169540423)

# Введение

В современном информационном обществе инструменты автоматизации все чаще применяются для получения и обработки данных из сети Интернет. Автоматизация этих процессов позволяет рационализировать и оптимизировать работу в области поиска и анализа информации. Для получения и обработки данных из сети Интернет используется парсинг. Парсинг веб-сайтов представляет собой процесс извлечения данных, структуры и характеристик веб-страниц с использованием специализированных инструментов.

Целью курсовой работы является разработка программного решения для парсинга веб-сайтов магазинов линолеума, а также сохранение полученных данных в базу данных. Для удобного просмотра полученных данных будет разработан веб-сайт.

В общем случае, парсинг веб-сайтов применяется во многих сферах: маркетинг и исследование рынка, финансы и инвестиции, научные исследования, реклама и маркетинг. В маркетинге и исследовании рынка парсинг позволяет получить данные о конкурентах, ценах, товарах и услугах, динамике спроса и предложения на рынке, а также облегчает процесс составления прайс-листов магазинов, в случаях парсинга сайтов компаний-поставщиков.

В работе будет приведен обзор существующих методов парсинга веб-сайтов, описан выбор сайтов и анализ их структуры и содержания. Затем будет разработан алгоритм парсинга для каждого выбранного сайта, создана реляционная база данных, в которую будут внесены полученные с веб-сайтов данные. Далее будет разработан веб-сайт, который позволит удобно просматривать информацию из базы данных.

# Изучение и выбор методов реализации

## Обзор существующих методов решения

Существует ряд методов в области парсинга веб-сайтов, которые позволяют извлекать данные с веб-страниц. Ниже приведены те, которые чаще всего используются специалистами парсинга.

1. API сайтов: API – программный интерфейс, который позволяет одной программе взаимодействовать с другой. Некоторые веб-сайты предоставляют официальные API для получения данных. Как правило, такой способ парсинга наиболее легкий, однако официальные API встречаются редко.
2. Получение данных из XHR-запросов: Чаще всего, современные веб-сайты используют JavaScript для загрузки данных с сервера путем отправления GET или POST запросов после получения структуры страницы, что делает загрузку содержимого постепенной и плавной. В инструментах разработчика любого браузера можно посмотреть эти запросы, что дает возможность повторить их в программе для парсинга. Такой способ является удобным и надежным в долгосрочной перспективе, поскольку в случае изменения ответов сервера их структура остается неизменной.
3. Поиск JSON в HTML-коде страницы: Чтобы страница корректно индексировалась поисковыми системами, необходимо, чтобы в HTML-коде страницы содержалась вся нужная информация. При генерации страницы на стороне сервера в HTML-код нередко добавляется JSON (текстовый формат обмена данными, основанный на JavaScript), содержащий данные этой страницы. Этот способ используется в том случае, когда отсутствует возможность получения данных предыдущим методом, однако извлечение JSON может быть трудным.
4. Парсинг HTML-кода: Применяется тогда, когда применение всех вышеупомянутых способов невозможно. Данные извлекаются непосредственно из элементов страницы. Для этого часто используются специальные библиотеки, например, BeautifulSoup4 или Scrapy. Способ подразумевает анализ структуры веб-страницы и извлечение данных из элементов с определенными тэгами или атрибутами. Такой подход является наименее надежным в долгосрочной перспективе, поскольку структура страницы со временем может быть изменена.

## 1.2 Обоснование выбранных методов и средств реализации

Поскольку целью работы не является разработка программного решения, которое будет надежным в долгосрочной перспективе, было решено использовать парсинг HTML-кода в качестве метода извлечения данных. Для этого была выбрана библиотека BeautifulSoup4 для языка программирования Python, так как она проста в использовании и имеет понятную официальную документацию.

Для получения HTML-кода веб-страниц выбрана встроенная в язык Python библиотека Requests. В случаях, если использование этой окажется неэффективным, будет использована библиотека Selenium, которая позволит автоматически запустить браузер и получить полностью сформированную веб-страницу.

Для разработки веб-сайта выбран фреймворк Django, реализуемый на языке Python. Стоит отметить, что данный фреймворк имеет специфические методы работы с базами данных. Об этих методах будет подробно изложено в пункте «Создание базы данных и внесение полученных данных» главы «Реализация проекта на основе выбранных методов».

## 1.3 Выбор подходящих веб-сайтов и их анализ

Для выполнения работы были выбраны следующие сайты магазинов, продающих линолеум:

**Топтыгин**

Топтыгин (<https://polov.net>) – магазин напольных покрытий и сопутствующих товаров. Магазин предлагает материалы самых популярных брендов, что делает его выбор обоснованным.

Анализ структуры страниц:

* Наименование товара: Расположено в тэге <h1>. Атрибут class учитывать необязательно, поскольку по правилам верстки HTML-страниц тэг <h1> на странице может быть только один.
* Цена товара: Расположена в тэге <span>, вложенном в тэг <div> с классом “product-item-detail-unit-price”.
* Характеристики товара: Расположены в тэгах <li> с классом “product-item-detail-properties-item”, где имя характеристики находится в тэге <span> с классом “product-item-detail-properties-name”, а значение характеристики – в тэге <span> с классом “product-item-detail-properties-value”. К характеристикам товара относятся: ширина, толщина, толщина защитного слоя, класс пожарной безопасности и бренд.
* Ссылка на изображение: Расположена в атрибуте src тэга <img>, вложенном в тэг <div> с классом “product-item-detail-slider-image”.

**Залог**

Залог (<https://zalog-vostok.ru>) – магазин строительных товаров во Владивостоке. Магазин предлагает широкий ассортимент товаров, в том числе и линолеумов, что делает его хорошим вариантом выбора.

Анализ структуры страниц:

* Наименование товара: Расположено в тэге <h1>.
* Цена товара: Расположена в тэге <span> с классом “price\_value”.
* Характеристики товара: Расположены в тэгах <div> с классом “properties\_\_item”, где имя характеристики находится в тэге <div> с классом “properties\_\_title”, а значение характеристики – в тэге <div> с классом “properties\_\_value”. К характеристикам товара относятся: ширина, толщина, толщина защитного слоя, класс пожарной безопасности и бренд.
* Ссылка на изображение: Расположена в атрибуте src тэга <img>, с классом “detail-gallery-big\_\_picture”.

**ПолДома**

ПолДома (<https://vladivostok.pol-doma.com>) – магазин напольных покрытий в крупных городах России. Магазин предлагает широкий ассортимент товаров различных брендов и ценовых категорий (в частности, более 2000 видов линолеума), что делает его достаточно ценным ресурсом для извлечения данных.

Анализ структуры страниц:

* Наименование товара: Расположено в тэге <h1>.
* Цена товара: Расположена в тэге <span> с классом “price\_value”. В отличие от предыдущих сайтов, цена указана за один квадратный метр линолеума, а не за один погонный метр.
* Характеристики товара: Расположены в тэгах строк таблицы <tr>, где имя характеристики находится в тэге <td> с классом “char\_name”, а значение характеристики – в тэге <td> с классом “char\_value”. К характеристикам товара относятся: ширина, толщина, толщина защитного слоя и класс пожарной безопасности.
* Бренд: Содержится непосредственно в наименовании товара и заключен в кавычки.
* Ссылка на изображение: Расположена в атрибуте href тэга <a>, с классом “popup\_link”.

Выбор этих сайтов обусловлен их популярностью, доступностью, наличием необходимых характеристик товаров, а также разнообразием товаров.

## 1.4 Возможные проблемы и способы их решения

Извлечение данных о товарах из каталога магазина подразумевает собой обращение к серверу для получения веб-страницы каждого товара. Однако слишком частые запросы из одного источника сервер может посчитать за подозрительный трафик и заблокировать этому источнику доступ по его IP-адресу.

Решений у этой проблемы несколько:

1. Как правило, такие блокировки ограничены по времени, а значит можно дождаться ее завершения.
2. Если у используемого Интернет-провайдера не подключена услуга «Статический IP-адрес», можно дождаться смены адреса.
3. Чтобы не ждать, можно использовать несколько прокси-серверов и переключаться между ними в случае блокировок.

Наилучшим вариантом является предотвращение появления данной проблемы. Например, можно изначально использовать прокси-сервера и переключаться между ними с определенной периодичностью или же можно делать паузы между запросами на сервер длительностью в несколько секунд.

# Реализация проекта на основе выбранных методов

## 2.1 Разработка алгоритмов парсинга

На основе приведенного анализа веб-страниц можно составить следующий алгоритм:

Для каждой страницы каталога.

1. Получение HTML-кода страницы.
2. Извлечение ссылок на товары текущей страницы каталога.
3. Добавление ссылок в список всех товаров.
4. Переход к следующей странице каталога.

Для каждой страницы товара.

1. Получение HTML-кода страницы.
2. Извлечение наименования товара из соответствующего элемента.
3. Извлечение цены товара из соответствующего элемента.
4. Извлечение списка характеристик товара.
5. Поиск значений необходимых характеристик в списке характеристик товара и приведение их к требуемому типу данных (например, ширина является целым числом, а из HTML-кода ее значение извлекается в виде текста).
6. Извлечение ссылки на фотографию товара.
7. Сохранение извлеченных данных.
8. Переход к следующей странице товара.

Ниже приведен код программы для парсинга магазина ПолДома:

from bs4 import BeautifulSoup  
import requests  
import time  
from parsers import csv\_rw  
  
  
def parse\_product\_page(url, otg: bool = False, num: int = None):  
 # Получаем HTML-код  
 page = requests.get(url)  
  
 # Парсим страницу  
 soup = BeautifulSoup(page.content, "html.parser")  
  
 width = 0  
 thickness = 0  
 safe\_layer = 0  
 fire\_safety\_class = None  
 brand = None   
  
 # Ищем данные  
 name = soup.find("h1").text.strip()  
 price = int(soup.find(class\_="price\_value").text.replace(' ', ''))  
 prop\_list = soup.find(id="props").find\_all('tr')  
  
 for prop in prop\_list:  
 prop\_title = prop.find(class\_='char\_name').find('span').text.strip()  
 prop\_value = prop.find(class\_='char\_value').find('span').text.strip()  
 if prop\_title == 'Толщина,мм':  
 thickness = float(prop\_value.replace(',', '.'))  
 elif prop\_title == 'Толщина защитного слоя,мм':  
 safe\_layer = float(prop\_value.replace(',', '.'))  
 elif prop\_title == 'Ширина,м':  
 width = int(float(prop\_value.replace(',', '.')) \* 100) # Ширина в см  
 elif prop\_title == 'Класс пожаробезопасности':  
 fire\_safety\_class = prop\_value  
  
 if '"' in name:  
 brand = name.split('"')[1] # Извлекаем бренд, т.к. отдельно он нигде не прописан  
 name = name.replace('"', '') # Убираем кавычки (они не нужны)  
 price = int(price \* width / 100) # Переводим цену за м2 в цену за погонный метр  
 image\_url = f'https://vladivostok.pol-doma.com{soup.find(class\_="popup\_link").get("href")}'  
  
 data = [name, price, width, thickness, safe\_layer, fire\_safety\_class, brand, url, image\_url]  
  
 # Запись в CSV файл  
 csv\_rw.write('data.csv', data)  
  
 # Вывод для отладки  
 if otg:  
 print(num, data)  
  
  
def parse\_catalog(otg: bool = False):  
 start\_url = 'https://vladivostok.pol-doma.com/catalog/napolnye\_pokrytiya/linoleum/?PAGEN\_1='  
 count = 1  
  
 for i in range(1, 94):  
 links = []  
 page = requests.get(start\_url + str(i))  
  
 soup = BeautifulSoup(page.content, "html.parser")  
 link\_items = soup.find\_all(class\_='image\_wrapper\_block')  
 for item in link\_items:  
 link = f"https://vladivostok.pol-doma.com{item.find('a').get('href')}"  
 links.append(link)  
  
 for link in links:  
 parse\_product\_page(link, otg, count)  
 count += 1  
 time.sleep(3) # Защита от бана по IP  
  
 time.sleep(5) # Защита от бана по IP  
  
  
parse\_catalog(True)

В данной программе реализованы две функции: parse\_catalog и parse\_product\_page.

Каталог магазина ПолДома содержит 93 страницы. В цикле функции parse\_catalog у каждой страницы идет получение HTML-кода с помощью requests.get(url), где url – это ссылка на страницу каталога. Далее с помощью BeautifulSoup из кода страницы извлекаются ссылки на товары и добавляются в список links. Затем для каждой ссылки из списка вызывается функция parse\_product\_page.

В функции parse\_product\_page с помощью requests.get(url) идет получение HTML-кода страницы. Затем с помощью BeautifulSoup извлекаются наименование, цена и список характеристик. Для каждого элемента списка характеристик извлекаются имя и значение характеристики. Если имя характеристики совпадает с одним из требуемых, то значение характеристики подвергается необходимым изменениям. Например, толщина линолеума является дробным числом, в программном коде дробные числа записываются через точку (3.5, например), однако на сайте значение указано в привычном для людей формате – через запятую (3,5). Поэтому прежде чем привести значение к типу данных float, необходимо заменить запятую на точку.

Далее из наименования извлекается бренд и убираются ненужные кавычки. Цена переводится из цены за квадратный метр в цену за погонный метр путем умножения на ширину в метрах. Затем извлекается ссылка на изображение.

Извлеченные данные сохраняются в CSV-файл. Выбор этого метода сохранения обусловлен удобством работы – данные не требуют повторного парсинга, что позволяет беспрепятственно передавать их между устройствами, на которых велась работа, а именно стационарный ПК и ноутбук. Далее информация из этого файла будет внесена в базу данных.

Остальные две программы имеют такую же структуру. Исключениями являются элементы, из которых извлекаются данные, обработка значений характеристик товара, а также наличие бренда в списке характеристик. Помимо этого, в программе для парсинга сайта Топтыгин получение HTML-кода страниц выполняется с использованием Selenium – запускается браузер, открывается веб-страница, извлекается код страницы и закрывается браузер.

## 2.2 Создание базы данных и внесение полученных данных

При создании Django-проекта автоматически создается база данных, управляемая СУБД SQLite. Для создания новой таблицы необходимо создать класс, который будет являться моделью таблицы. В этом классе указываются поля таблицы, их названия, типы данных и прочие параметры.

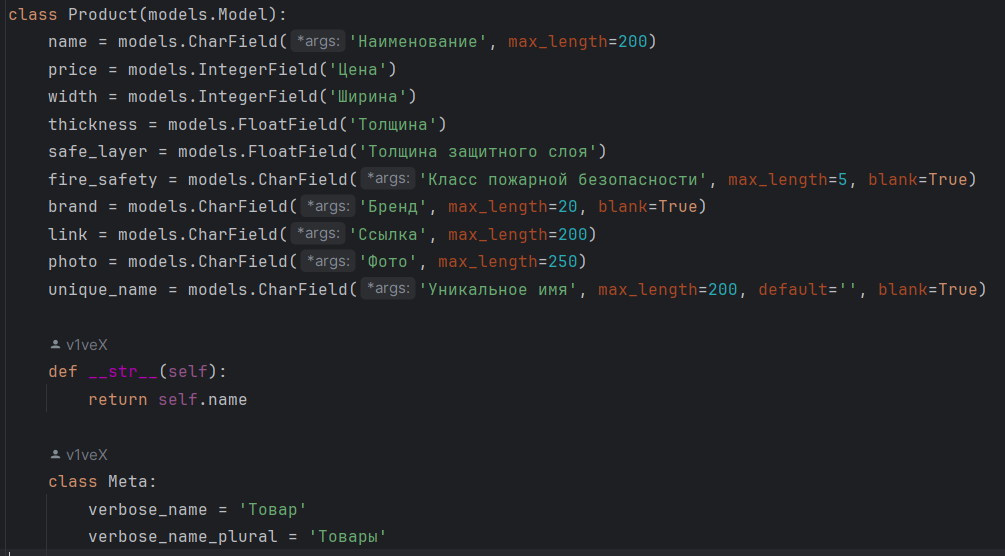


Рисунок 1 – Класс модели таблицы «Товары»

На Рис. 1 приведен класс модели таблицы «Товары». В нем указаны поля «Наименование», «Цена», «Ширина», «Толщина», «Толщина защитного слоя», «Класс пожарной безопасности», «Бренд», «Ссылка», «Фото» и «Уникальное имя». Поле «Уникальное имя» содержит исключительно служебную информацию, на основе которой будет производиться исключение из таблицы повторяющихся записей.

После создания модели, необходимо выполнить миграции. Миграции – это способ Django распространять изменения, которые вносятся в модели (добавление поля, удаление модели и т. д.) в схему базы данных. В данном случае, при выполнении миграций выполняется следующая SQL-команда:

CREATE TABLE Products (

id int PRIMARY KEY,

name varchar(200),

price int,

width int,

thickness float,

safe\_layer float,

fire\_safety varchar(5),

brand varchar(20),

link varchar(200),

photo varchar(250),

unique\_name varchar(200)

);

Итогом выполнения миграций является добавленная в схему базы данных таблица Products.

Для добавления извлеченной информации о товарах была разработана отдельная программа, которая считывает CSV-файл и добавляет новые записи в таблицу, исключая добавление повторяющихся товаров. Особое внимание стоит уделить следующим строкам:

file\_path = '../parsers/data.csv'  
  
counter = 0  
products = csv\_rw.read\_all(file\_path)  
for item in products:  
 unique\_name = reform\_name(item[0])  
 obj = Product(  
 name=item[0],  
 price=int(item[1]),  
 width=int(item[2]),  
 thickness=float(item[3]),  
 safe\_layer=float(item[4]),  
 fire\_safety=item[5],  
 brand=item[6],  
 link=item[7],  
 photo=item[8],  
 unique\_name=unique\_name  
 )  
  
 try:  
 existing\_product = Product.objects.get(unique\_name=unique\_name)  
 if existing\_product.price > obj.price:  
 existing\_product.price = obj.price  
 existing\_product.name = obj.name  
 existing\_product.link = obj.link  
 existing\_product.photo = obj.photo  
 if existing\_product.fire\_safety == '':  
 existing\_product.fire\_safety = obj.fire\_safety  
 if existing\_product.brand == '':  
 existing\_product.brand = obj.brand  
  
 counter += 1  
 existing\_product.save()  
 except Product.DoesNotExist:  
 obj.save()  
print(f'Existing products: {counter}')

Сначала создается экземпляр класса модели таблицы. Затем в блоке try-except производится попытка поиска в базе данных товара с уже существующим уникальным именем. В данном случае команда Product.objects.get(unique\_name=unique\_name) эквивалентна SQL-запросу SELECT \* FROM Products WHERE unique\_name = unique\_name, за тем лишь исключением, что данный SQL-запрос выберет все записи с заданным значением поля unique\_name, а Product.objects.get выберет только один. Если элементов окажется больше одного, то будет вызвано исключение MultipleObjectsReturned. Однако в данном случае это исключение не будет вызвано никогда, поскольку при добавлении информации в базу данных намеренно избегаются повторы.

Уникальное имя составляется следующим образом: из наименования удаляются все лишние слова и символы, в том числе пробелы, и производится замена символов русского алфавита на символы английского алфавита (транслитерация). Например, уникальное имя одного и того же товара, который на одном сайте записан как «Линолеум Tarkett Гладиатор Миллер 1 3,5м / 0,4мм», а на другом – «Линолеум бытовой Tarkett Gladiator Miller 1 (3,5м)», будет выглядеть как «tarkettgladiatormiller13,5m».

Если товар с уже существующим уникальным именем нашелся, то информация о нем будет изменена, при условии, что его цена больше, чем цена нового товара. Также если у существующего товара отсутствуют (т. е. равны пустой строке) значения полей «Класс пожарной безопасности» и «Бренд», то эти значения берутся из соответствующих полей нового товара. В данном случае команда existing\_product.save() эквивалентна SQL-запросу UPDATE Products SET name = name, price = price, link = link, photo = photo, fire\_safety = fire\_safety, brand = brand WHERE unique\_name = unique\_name.

Если же товар с уже существующим уникальным именем не найден, то новый товар добавляется в таблицу. В этом случае команда obj.save() эквивалентна SQL-запросу INSERT INTO Products (name, price, width, thickness, safe\_layer, fire\_safety, brand, link, photo, unique\_name) VALUES (name, price, width, thickness, safe\_layer, fire\_safety, brand, link, photo, unique\_name).

В результате выполнения этой программы, база данных была заполнена извлеченными с сайтов магазинов данными. При этом было обнаружено 18 одинаковых товаров.

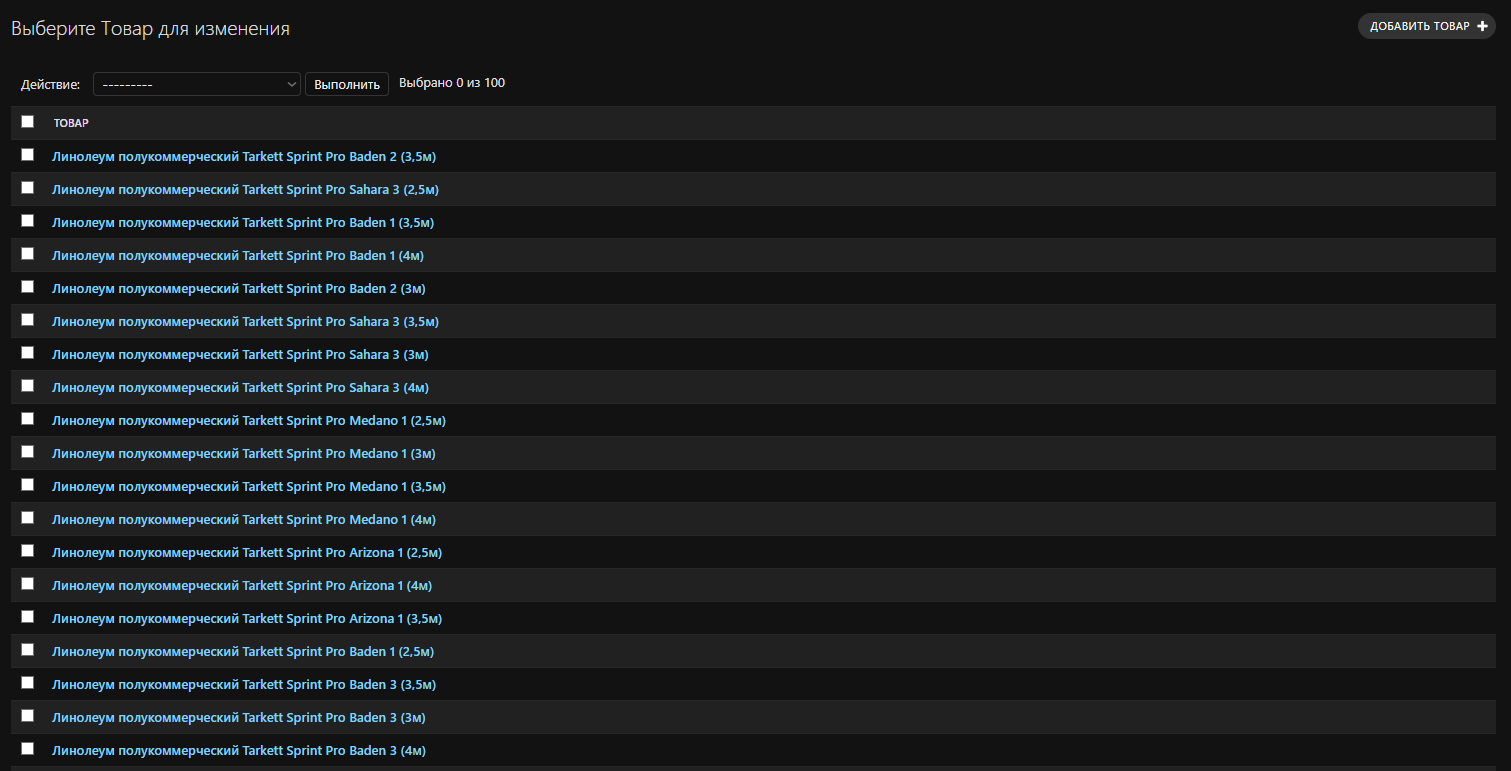


Рисунок 2 – Скриншот базы данных в панели управления сайтом

## 2.3 Разработка веб-сайта

Веб-сайт разработан таким образом, что содержимое страницы генерируется непосредственно на сервере и клиент получает полностью сформированную страницу. Наибольший интерес в данной работе представляет взаимодействие с базой данных, об этом будет изложено ниже.

При загрузке страницы клиент отправляет на сервер GET-запрос. Сервер получает этот запрос и обрабатывает его параметры. Затем из базы данных извлекаются все данные. Django позволяет обращаться к базе данных лишь один раз, а затем фильтровать и сортировать данные без дальнейшего обращения к базе данных. Данные извлекаются при помощи команды Product.objects.all(), что эквивалентно SQL-запросу SELECT \* FROM Products. Затем на основе параметров запроса данные фильтруются и сортируются при том условии, что значение параметра не равно пустой строке и не равно None (аналог null в C-подобных языках программирования).



Рисунок 3 – Код взаимодействия с базой данных

В результате на сайте реализована фильтрация товаров по бренду, ширине и толщине, а также сортировка по алфавиту и цене.

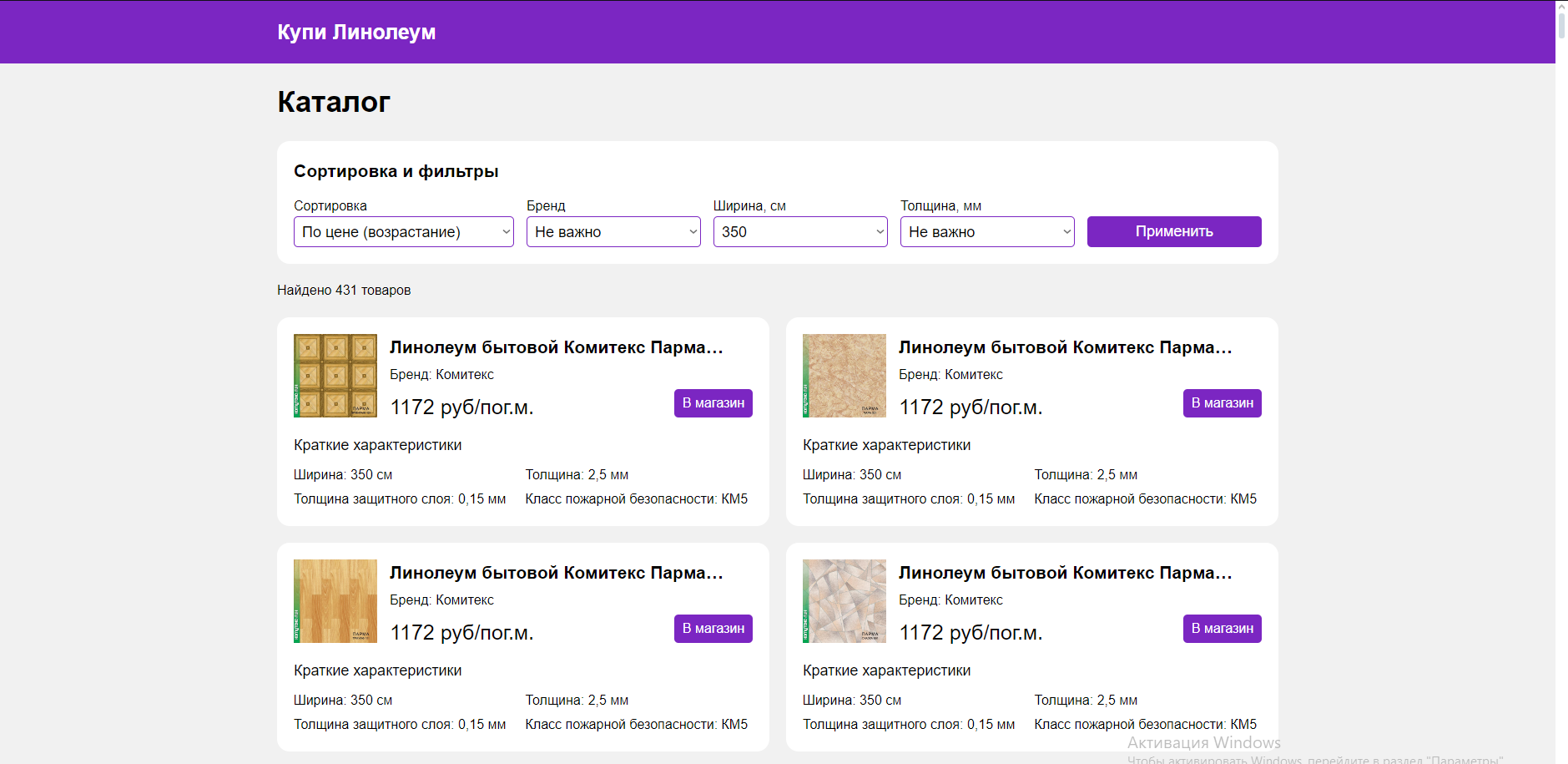


Рисунок 4 – Товары, отфильтрованные по ширине 350 см и отсортированные по цене

# Заключение

В ходе работы над курсовым проектом было разработано программное решение для парсинга трех веб-сайтов магазинов, продающих линолеум. Извлеченная информация о товарах сохранена в реляционной базе данных. Для удобного просмотра извлеченных данных был разработан веб-сайт, позволяющий фильтровать и сортировать данные.

В ходе работы были получены практические навыки работы с библиотеками BeautifulSoup4 для парсинга HTML-кода, Selenium для автоматизации браузера, а также фремворком Django для разработки веб-сайтов

# Список литературы

1. Как спарсить любой сайт? | Habr [Электронный ресурс] / URL: <https://habr.com/ru/articles/579336/>
2. Документация BeautifulSoup4 | readthedocs.io [Электронный ресурс] / URL: <https://beautiful-soup-4.readthedocs.io/en/latest/>
3. Документация Django на русском языке | Django.fun [Электронный ресурс] / URL: <https://django.fun/docs/django/5.0/>
4. Путеводитель по SQL | W3Schools [Электронный ресурс] / URL: <https://www.w3schools.com/sql/>