Оглавление

[ВВЕДЕНИЕ 6](#_Toc74033659)

[1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧ 7](#_Toc74033660)

[1.1 Описание предметной области 7](#_Toc74033661)

[1.2 Постановка задачи проектирования 8](#_Toc74033662)

[1.3 Анализ аналогов и прототипов 8](#_Toc74033663)

[1.4 Анализ требований к проекту 8](#_Toc74033664)

[1.5 Выбор среды и средств разработки 9](#_Toc74033665)

[1.6 Разработка технического задания 9](#_Toc74033666)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 11](#_Toc74033667)

[2.1 Описание вариантов использования 11](#_Toc74033669)

[2.2 Проектирование базы данных 11](#_Toc74033670)

[2.3 Проектирование интерфейса web-приложения 13](#_Toc74033671)

[3 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ 15](#_Toc74033672)

[2.1 Создание и настройка проекта 15](#_Toc74033673)

[2.2 Реализация главного приложения 15](#_Toc74033674)

[2.3 Реализация авторизации и регистрации 18](#_Toc74033675)

[4 ЭКОНОМИЧЕСКАЯ ЧАСТЬ 21](#_Toc74033677)

[4.1 Обоснование необходимости выведения продукта на рынок 21](#_Toc74033682)

[ЗАКЛЮЧЕНИЕ 22](#_Toc74033683)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc74033684)

[ПРИЛОЖЕНИЯ 24](#_Toc74033685)

# ВВЕДЕНИЕ

С развитием информационных технологий, а в частности, с появлением сети «Интернет», сфера коммерции вышла на новый уровень. Раньше приходилось ездить по всему городу для того, чтобы найти необходимую вещь, а теперь стало доступным покупать товары за границей и для этого даже не нужно вставать с кресла.

Электронный магазин – это веб-ресурс, на котором размещена информация о товарах для продажи. Подобно обычному магазину, электронный магазин реализует следующие основные функции: представление товаров (услуг) покупателю, обработку заказов, продажу и доставку товаров.

Интернет-магазины имеют ряд преимуществ, в отличие от обычной формы торговли. Онлайн вариант может обеспечить потребителя большим объемом информации о товарах, необходимым для принятия решения о покупке. Теперь «Юзер» может в пару «кликов» найти необходимый продукт. Кроме того, за счет использования компьютерных технологий стало возможным отслеживать интересы пользователей, что позволяет продвигать именно те товары, которые нужны потребителям.

Однако у электронной коммерции свои недостатки. В обычной торговле покупатель привык к тому, что есть возможность оценить товар визуально, определить его качество и характеристики. В электронной торговле он такой возможности лишен. Зачастую визуальной информации достаточно, но здесь вступают в действие эмоциональные и психологические факторы. Часто существуют проблемы с доставкой товаров, особенно если их цена невелика.

По данным очередного исследования E-data, проведенного «маркетплейсом» Deal.by, в 2020 году онлайн-продажи показали самые высокие темпы роста с 2018-го. Такие показатели обусловлены увеличением онлайн-покупок белорусов во время самоизоляции.

По оценке Deal.by, в сравнении с прошлым годом за восемь месяцев 2020-го товарооборот в интернете вырос на 30%. По данным Белстата, вся розница за аналогичный период прибавила 7%. Активность покупателей на некоторых площадках в период апрель-июнь выросла в два и более раза.

Целью данной дипломной работы является разработка информационной системы предприятия по продаже обуви. Данная система будет работать в рамках определенного предприятия и выполнять функции основной площадки для ведения коммерческой деятельности.

## АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧ

## Описание предметной области

После того как предприятие, занимающееся торговлей, захватывает долю на рынке, а его «Brand name» становится все более узнаваемым, единственно верным шагом становится перенос всей его инфраструктуры в онлайн пространство и автоматизация некоторых его процессов.

Конечно, выйти на готовые интернет-площадки было бы очень заманчиво. Ведь «онлайн-гипермаркеты» уже имеют готовый трафик покупателей и посетителей. Также для порога входа в таком случае будет минимальным. Есть возможность использовать инфраструктуру гипермаркетов – техподдержка, логистика. На этом можно сэкономить много времени и денег. Такой вариант отлично подходит малоизвестным брэндам. Однако есть у этого подхода свои недостатки. Главный из них – не всегда финансовые условия с «торговой площадкой» бывают приемлемыми. Так же существует риск быть незамеченным среди других более известных брэндов.

Если же есть возможность доставить товар в течение рабочего дня, подобрать индивидуальные условия для каждого клиента, или же давать кредиты под покупку, то лучшим вариантом станет создание собственной «интернет-площадки».

Интернет не позволяет пользователю просматривать товары на витринах торговых залов, к тому же рядом нету кучи консультантов готовых предоставить руку помощи, и пользователю остается полагаться только на себя и на свои компьютерные навыки.

Интернет-магазин – это сайт, который торгует товарами в интернете. Позволяет пользователям сформировать заказ на покупку, выбрать способ оплаты и доставки в сети Интернет. Информация о товарах размещается в каталогах, которые пользователи просматривают. Сделав выбор на конкретном товаре, пользователь обычно имеет выбрать способ оплаты и доставки. Совокупность выбранных товаров, способ оплаты и доставки представляют собой заказ, который оформляется на сайте путем сообщения минимальной информации о пользователе.

Интернет-магазины обычно создают с использование CMS (систем управления контентом сайтов), оснащенных необходимыми модулями. Крупные интернет-магазины работают на специально разработанных или адаптированных типовых системах управления. Средние и малые магазины обычно используют типовое коммерческое и свободное ПО.

Нужды администраторов интернет-магазина в складском, торговом, бухгалтерском и налоговом учете должны поддерживаться невидимой посетителям частью интернет-магазина — интернет-офисом. Экономически эффективной практикой создания интернет-магазинов является применение специализированных систем учета. Интернет-магазин обычно интегрирован с такими системами учета.

Как правило все предприятия, связанные с торговлей, имеют свои «хранилища данных». База данных должна иметь возможность хранить всю информацию, связанную с деятельностью предприятия – информация о товарах, персонале, пунктах выдачи и др. Современные базы данных предоставляют огромную скорость выполнения запросов, сохраняют целостность данных, а также позволяют легко вносить в них дополнения.

## Анализ аналогов и прототипов

Аналогом данного проекта на территории Беларуси, можно назвать торговую сеть «МЕГАТОП». Предприятие имеет «онлайн-магазин» и множество торговых точек по всей Беларуси.

## Анализ требований к проекту

На основании анализа предметной области были выделены следующие требования к проекту.

Необходимо спроектировать базу данных, обеспечить пользователя понятным интерфейсом, а менеджеров, администраторов и другой персонал, инструментами для ведения бизнеса.

Каталог товаров легко просматривается, фильтруется и сортируется. Товары можно добавить в корзину для дальнейшего оформления заказа или сравнения. Пользователь должен иметь возможность авторизоваться на сайте. На сайте должна быть реализована административная часть, которая позволяет производить импорт и экспорт товаров из/в JSON, Xml, CSV. Административная часть должна позволять легко изменять контент сайта, назначать товарам скидки и полностью менять информацию о товаре.

Существует четыре вида пользователей – Администратор, покупатель, менеджер, бухгалтер. Администратор может назначать пользователям права.

## Выбор среды и средств разработки

Существует огромное количество инструментов для разработки web-приложений на подобие «интернет-магазина». Основные языки программирования на которых пишут подобные программы – PHP, Python, JavaScript, Ruby.

PHP – (от англ. Hypertext Preprocessor) язык программирования высокого уровня. Подходит для разработки средних и малых сайтов. Имеет множество инструментов «из-под коробки».

JavaScript – относится к так называемым «скриптовым» языкам программирования. Наиболее широко используется в браузерах как язык сценариев. Платформа Node, которая транслирует его код в «машинный», тем самым делая JavaScript языком общего назначения. У Node.js есть множество веб-Фреймворков, наиболее популярный из них – Express.js.

Ruby – динамический, интерпретируемый, высокоуровневый язык программирования. Обладает «многопоточностью», которая не зависит от операционной системы. Специализируется на разработке веб-приложений, для этого используется «фреймворк» Ruby on Rails.

Python – это высокоуровневый язык общего назначения, интерпретируемый и строго типизированный. Он генерирует байт-код, который использует PVM (Python Virtual Machine) – виртуальная машина для запуска программ. Программы удобно запускать, ведь Python является кроссплатформенным языком программирования, и часто поставляется вместе с операционными системами, например – Linux. Python широко используется в web разработке. Он имеет ряд инструментов для написания web-приложений. Самый популярный из них – Django. «Питон» будет лучшим выбором, для написания веб-приложения.

Django – свободный Фреймворк для веб-приложений, использующий шаблон программирования MVC. Сайт на Django состоит из нескольких приложений, которые общаются друг с другом. Это одно из существенных архитектурных отличий от Ruby on Rails.

В качестве среды разработки будет использоваться Visual Studio Code. Это редактор исходного кода, разработанный корпорацией Microsoft, который поддерживает множество языков программирования, в том числе и Python. Редактор очень удобный, а также имеет множество плагинов для облегчения разработки.

## Разработка технического задания

На основании анализа предметной области были выделена следующая функциональность приложения.

Пользователь должен иметь возможность:

* авторизация в системе;
* просмотр каталога товаров;
* выбор товара;
* сравнение товаров;
* добавление товара в корзину;
* оформление заказа.

Менеджер должен иметь возможность:

* просмотр заказов;
* обработку заказов;
* связь с клиентом;
* все возможности пользователя.

Администратор имеет возможность

* добавление, изменение, удаление товаров на складе;
* импорт, экспорт товаров;
* просмотр и редактирование базы данных;
* все возможности менеджера.

Бухгалтер должен иметь возможность доступа к информации о проданных вещах.

Разработка программы осуществляется на языке программирования Python 3.9.0, «веб-фреймворк» – Django 3.1.7, база данных – PostgreSQL 13.0, Веб-сервер – Nginx, Операционная система для развертывания – Ubuntu 18.04 LTS.

1. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## Описание вариантов использования

Диаграмма вариантов использования (Use case Diagram) описывает функциональное назначение системы. Она позволяет определить общие границы проектируемой системы в контексте моделируемой предметной области.

Можно выделить четыре вида актеров в проектируемой системе:

* Пользователь.
* Администратор.
* Менеджер.
* Бухгалтер.

Пользователь имеет следующие возможности:

* авторизация в системе;
* просмотр каталога товаров;
* выбор товара;
* сравнение товаров;
* добавление товара в корзину;
* оформление заказа.

Менеджер имеет возможности:

* просмотр заказов;
* обработку заказов;
* связь с клиентом;
* все возможности пользователя.

Администратор имеет возможности:

* добавление, изменение, удаление товаров на складе;
* импорт, экспорт товаров;
* просмотр и редактирование базы данных;
* все возможности менеджера.

Бухгалтер имеет возможность просматривать, делать выборку данных из сводных таблиц по продажам.

Диаграмма вариантов использования предоставлена в приложении А.

## Проектирование базы данных

База данных интернет-магазина должна соответствовать всем нормам качества. В ней должна быть сохранена целостность данных, доступ к которым можно быстро получать.

База данных должна позволять хранить товары и их вариации. Товар может иметь несколько атрибутов. Товары каждой категории имеют свои атрибуты. Атрибуты бывают простые и вариативные. Первые не влияют ни на что, а товары с вариативными атрибутами могут иметь собственные описание и цену. Должна быть реализована возможность добавлять новые атрибуты. Товар может иметь несколько фотографий. Одна фотография может быть у нескольких вариаций товаров.

Исходя из всех требований, спроектированная база данных выглядит следующим образом.

Таблица Product имеет следующие поля:

* имя;
* брэнд;
* дата публикации;
* категория;
* пол;
* сезон;
* описание.

Таблица ProductVariation имеет следующие поля:

* идентификатор вариации;
* артикул;
* идентификатор товара;
* идентификатор значения аттрибута;
* цена;
* замер скидки;
* количество товара на складе;
* идентификатор группы товаров;
* короткое описание.

Таблица Categories имеет следующие поля:

* идентификатор категории;
* название.

Таблица Brand имеет следующие поля:

* идентификатор брэнда;
* название.

Таблица Attributes имеет следующие поля:

* идентификатор атрибута;
* заголовок;
* название на латинице;
* тип атрибута.

Таблица AttributeAndCategory имеет следующие поля:

* идентификатор категории;
* идентификатор атрибута.

Таблица Values имеет следующие поля:

* идентификатор значения;
* значение;
* идентификатор атрибута;
* идентификатор товара.

Таблица Photo имеет следующие поля:

* идентификатор фото;
* описание;
* web url фото.
* локальный путь к фото

Таблица ProductPhoto имеет следующие поля:

* идентификатор фотографии;
* идентификатор товара.

Диаграмма базы данных предоставлена в приложении Б.

## Проектирование интерфейса web-приложения

Разработка графического интерфейса пользователя является немаловажной задачей. Необходимо разработать интерфейс главной страницы для выполнения выдвинутых требований. Структура интерфейса главной страницы представлена ниже:

* Каталог.
* Корзина.
* Акции.
* Кабинет пользователя

Далее представлено описание разделов интернет-магазина. Главная страница является основной точкой входа на сайт.

Назначение главной страницы:

* идентифицировать сайт, как представительство магазина в сети интернет;
* обеспечивать доступ ко всем основным разделам сайта;
* информировать посетителей о акциях магазина.

Основные элементы, которые должна содержать главная страница - это идентификационные данные. К ним относятся данные, позволяющие отождествить сайт с магазином: логотип и название магазина;

Раздел «Каталог» включает:

* назначение раздела: предоставить клиенту информацию о товарах в магазине.
* контентный блок. Страница раздела должна содержать:
* основные подразделы, разделяющие всю продукцию по типам. В подразделах описания продукции имеют следующие атрибуты (название раздела, фото, цена, описание);
* кнопку «Добавить в корзину», которая добавляет выбранный товар в корзину.

Раздел «Корзина» включает:

* Назначение раздела: вывод информации о товарах в корзине;
* Контентный блок.

Таблица с полями:

* название товара;
* цена;
* количество;
* возможность редактирования товаров в корзине;
* возможность очистить корзину;
* возможность оформить заказ.

Раздел «Контакты» включает:

* адрес, время работы, телефон магазина;
* блок обратной связи.

Форма администратора для удобства должна быть поделена на несколько частей. Структура формы администратора:

Раздел «Каталог товаров» включает:

* «Категории»;
* «Товары»;
* «Дополнительные характеристики товаров».

Раздел «Заказы» включает список всех заказов.

Раздел «Пользователи» включает таблицы пользователей и групп.

Раздел «Настройки» включает настройки магазина.

1. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ

## Создание и настройка проекта

Разработка интернет-магазина ведется на языке программирования Python 3.9.0. с использованием «Фреймворка» Django 3.1.7. В качестве СУБД была выбрана PostgreSQL 13.0. Разработка ведется в редакторе кода Visual Studio Code.

Все программы на Django состоят из функциональных частей (приложений). Работа каждого приложения должна строго соответствовать его назначению и быть «подключаемой». Это один из важнейших принципов в разработке на Django.

Для того чтобы создать проект необходимо установить интерпретатор Python и выполнить следующие инструкции в командной строке:

1. Python –m pip install –update pip
2. Python –m pip install –upgrade pip

После чего нужно перейти в директорию проекта и выполнить следующие инструкции

1. Python –m venv env
2. Django-admin startproject shoestore .

Это создаст в директории проекта все необходимые файлы. Главным, из которых будет manage.py. Эта утилита позволяет управлять всем проектом.

«Джанго» использует ORM для работы с базой данных. Для подключения PostgreSQL к проекту необходимо создать базу данных shoestoredb и в файле shoestore/settings.py внести изменения как показано на листинге 1.

**Листинг 1** – Инструкция подключения базы данных к проекту

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql\_psycopg2',

'NAME': 'shoestoredb',

'USER': 'storeadmin',

'PASSWORD': 'storeadmin',

'HOST': '127.0.0.1',

'PORT': 5432

}

}

## Реализация главного приложения

После того как проект подготовлен, необходимо создать главное приложение. Для этого нужно ввести команду python manage.py startapp main. После чего появится новый каталог с файлами:

* Views.py
* Admin.py
* Apps.py
* Test.py
* Models.py

На листинге 2 предоставлен код из модуля models.py. Этот модуль создает все необходимые таблицы в базе данных.

Листинг 2 – Содержимое файла models.py

from django.db import models

import random

import string

class Brand(models.Model):

name = models.CharField('Brand name', max\_length=100, unique=True)

def \_\_str\_\_(self):

return "{0}".format(self.name)

class Categories(models.Model):

name = models.CharField('Имя категории', max\_length=100, unique=True, default=None)

slug = models.SlugField(max\_length=50, unique=True, default=None)

parent = models.ForeignKey('self', default=None, blank=True, verbose\_name='Категория родитель', null=True, on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return "{0}".format(self.name)

class Attributes(models.Model):

TYPE\_CHOICES = (

('simple', 'Simple'),

('variable', 'Variable'),

('variation', 'Variation')

)

title = models.CharField(max\_length=50)

slug = models.CharField(max\_length=50)

attribute\_type = models.CharField(max\_length=50, choices=TYPE\_CHOICES)

def \_\_str\_\_(self):

return "{0} ({1})".format(self.title, self.attribute\_type)

class AttributeAndCategory(models.Model):

category= models.ForeignKey(Categories, verbose\_name='Категория', on\_delete=models.CASCADE)

attribute = models.ForeignKey(Attributes, verbose\_name='Аттрибут', on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return "category.name={0}, attribute.slug={1}>".format(self.category.name, self.attribute.slug)

class Values(models.Model):

value = models.CharField(max\_length=200)

attribute = models.ForeignKey(Attributes, on\_delete=models.CASCADE)

product = models.ForeignKey('Product', on\_delete=models.CASCADE, blank=True, null=True, default=None)

def \_\_str\_\_(self):

return "{1}: {0}".format(self.value, self.attribute.title)

def save(self, \*args, \*\*kwargs):

if (self.attribute.attribute\_type != "simple"

) and (self.product is not None):

raise AttributeError("Attribute of value have type '{0}'. Most be a simple".format(self.attribute.attribute\_type))

super().save(\*args,\*\*kwargs)

class Photo(models.Model):

photoUrl = models.CharField(max\_length=255, blank=True, null=True)

photoBin = models.ImageField(upload\_to='images/', blank=True, null=True)

photoName = models.CharField(max\_length=100, default="No Image")

def \_\_str\_\_(self):

return "img={0}".format(self.photoName)

class ProductPhoto(models.Model):

product = models.ForeignKey('ProductVar', on\_delete=models.CASCADE, default=None)

photo = models.ForeignKey('Photo', on\_delete=models.CASCADE, default=None)

def \_\_str\_\_(self):

return "sku={0}, name={1}".format(self.product.sku,self.photo.photoName)

class Product(models.Model):

SEX\_CHOICES = (

('M', 'Male'),

('F', 'Female'),

('U', 'Unisex'),

)

name = models.CharField(max\_length=200)

brand = models.ForeignKey(Brand, on\_delete=models.CASCADE)

pub\_date = models.DateTimeField() # дата публикации

category = models.ForeignKey(Categories, on\_delete=models.CASCADE)

sex = models.CharField(max\_length=20, choices=SEX\_CHOICES)

season = models.CharField(max\_length=20)

description = models.TextField(blank=True, null=True)

def \_\_str\_\_(self):

return "{0}".format(self.name)

class ProductVar(models.Model):

id = models.BigAutoField(primary\_key=True)

sku = models.CharField(max\_length=12, unique=True, blank=True) # артикул

product = models.ForeignKey(Product, on\_delete=models.CASCADE)

value = models.ForeignKey(Values, on\_delete=models.CASCADE)

reg\_price = models.FloatField(default=0.0)

sale\_size = models.IntegerField(default=None, blank=True, null=True)

count = models.IntegerField(default=0) # количество товаров с таким значением атрибута

group\_id = models.PositiveIntegerField(default=None, blank=True, null=True)

short\_description = models.CharField(max\_length=200, blank=True, null=True)

def \_\_str\_\_(self):

return "id={0}, value={1}".format(self.id, self.value.value)

def save(self, \*args, \*\*kwargs):

random\_symbols = string.ascii\_uppercase + string.digits

while not self.sku:

sku = (random.choice(random\_symbols) for i in range(10))

sku = "{0}{1}{2}{3}-{4}{5}{6}{7}-{8}{9}".format(\*sku)

if not ProductVar.objects.filter(sku=sku):

self.sku = sku

super().save(\*args,\*\*kwargs)

## Реализация авторизации и регистрации

Для создания форм авторизации и регистрации в Django, необходимо создать классы, унаследованные от forms.Form. Код классов регистрации и авторизации предоставлены на листингах 3 и 4.

Листинг 3 – Класс формы регистрации.

class RegisterForm(forms.Form):

login = forms.CharField(widget=forms.TextInput(attrs={

'class': 'form-control form-account',

}))

email = forms.EmailField(widget=forms.EmailInput(attrs={

'class': 'form-control form-account',

'placeholder': 'example@email.com'

}))

password = forms.CharField(widget=forms.PasswordInput(attrs={

'class': 'form-control form-account',

}))

Листигн 4 – Класс формы Авторизации.

class LogInForm(forms.Form):

login = forms.CharField(max\_length=100, widget=forms.TextInput(attrs={

'class': 'form-control form-account',}))

password = forms.CharField(widget=forms.PasswordInput(attrs={

'class': 'form-control form-account',}))

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self.fields['login'].label = "Ваш логин"

self.fields['password'].label = "Ваш пароль"

def clean(self):

print(self.cleaned\_data)

username = self.cleaned\_data['login']

password = self.cleaned\_data['password']

user = authenticate(username=username, password=password)

if user is None:

raise forms.ValidationError("Неверный логин или пароль.")

def clean\_login(self):

login = self.cleaned\_data['login']

if login[0] in string.punctuation + string.digits + string.whitespace:

raise ValidationError("Логин должен начинаться с латинской буквы.")

if len(login) > len(login.replace(' ', '')):

raise ValidationError("Логин не должен содержать пробелов и знаков табуляции.")

if login.isnumeric():

raise ValidationError("Логин не может состоять только из цифр.")

return login

def clean\_password(self):

password = self.cleaned\_data['password']

if len(password) > len(password.replace(' ', '')):

raise ValidationError("Пароль не должен содержать пробелов и знаков табуляции.")

if len(password) < 3:

raise ValidationError("Пароль должен состоять из 3 и более символов")

return password

class Meta:

model = User

fields = ['login', 'password']



Представление авторизации в первую очередь показывает форму для заполнения данными. После того как пользователь вводит данные и нажимает кнопку отправки, на сервер приходит POST запрос, со всеми данными пользователя, после чего на стороне сервера, происходит «распаковка» данных в класс формы, проверяется их «правильность» и в случае успеха происходит вход в систему, а пользователь перенаправляется на другую страницу. Если же данные для входа не верны, пользователь получит уведомления о неправильно введенных логина и пароля. Код класса представления предоставлен на листинге 6.

Листинг 6 – Код класса LogInView.

1. def LogInView(request):
2. if request.method == "POST":
3. form = LogInForm(request.POST)
4. next\_page\_url = request.GET['next']
5. if form.is\_valid():
6. client\_data = form.cleaned\_data
7. user = authenticate(username=client\_data['login'],
8. password=client\_data['password'])
9. if user is not None:
10. login(request, user)
11. return HttpResponseRedirect(next\_page\_url)
12. return render(request, 'account/login.html', context={ # В случае если форма не верна.
13. 'user': request.user,
14. 'form': form,
15. })
16. form = LogInForm()
17. return render(request, 'account/login.html', context={ # Первичное отображение формы.
18. 'user': request.user,
19. 'form': form
20. })

Представление регистрации работает аналогично представлению авторизации. Код класса регистрации предоставлен на листинге 7.

Листинг 7 – Код класса RegisterView

1. def RegisterView(request):
2. if request.method == "POST":
3. form = RegisterForm(request.POST)
4. if form.is\_valid():
5. client\_data = form.cleaned\_data
6. user = User(
7. username=client\_data['login'],
8. email=client\_data['email'],
9. password=client\_data['password']
10. )
11. user.save()
12. return HttpResponseRedirect(reverse('login'))
13. return render(request, 'account/registration.html', context={
14. 'user': request.user,
15. 'form': form})
16. form = RegisterForm()
17. return render(request, 'account/registration.html', context={
18. 'user': request.user,
19. 'form': form})
20. ЭКОНОМИЧЕСКАЯ ЧАСТЬ

## Обоснование необходимости выведения продукта на рынок

# ЗАКЛЮЧЕНИЕ

В результате выполнения дипломного проекта

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЯ