Министерство образования Московской области

Государственное образовательное учреждение высшего образования Московской области

«Государственный гуманитарно-технологический университет»

**Ликино-Дулевский политехнический колледж – филиал ГГТУ**

**КУРСОВОЙ ПРОЕКТ**

Тема: «Разработка Интернет-магазина игровой продукции»

МДК.08.01 «Проектирование и разработка интерфейсов пользователя»

**Выполнила:**

Шипкова Софья Сергеевна

Студентка 3 курса группы ИСП.21.2А

09.02.07 Информационные системы и программирование)

очной формы обучения

**Руководитель:**

Гжегожевский Сергей Владимирович

Оценка

\_\_\_\_\_\_\_\_\_/\_\_\_­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

Подпись руководителя

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ликино-Дулево

2024 год

Содержание

[Введение 3](#_Toc163392302)

[1. Разработка системного проекта. 4](#_Toc163392303)

[1.1. Назначение разработки 4](#_Toc163392304)

[1.2. Требования к программному продукту 4](#_Toc163392305)

[1.2.1. Требования к функциональным характеристикам 4](#_Toc163392306)

[1.2.2. Требования к надежности и безопасности 4](#_Toc163392307)

[1.2.3. Требования к технологиям и инструментам для вёрстки 5](#_Toc163392308)

[1.2.4. Требования к составу и параметрам технических средств 5](#_Toc163392309)

[2. Разработка технического проекта 6](#_Toc163392310)

[2.1. Обоснование выбора CASE – средств 6](#_Toc163392311)

[2.2. Проектирование модели данных 6](#_Toc163392312)

[2.2.1. Диаграмма прецедентов 6](#_Toc163392313)

[2.2.2. Моделирование бизнес-процессов 7](#_Toc163392314)

[2.2.3. Словесный алгоритм бизнес-процессов 7](#_Toc163392315)

[2.2.4. Построение диаграммы действий 7](#_Toc163392316)

[3. Реализация приложения 9](#_Toc163392317)

[3.1. Обоснование выбора средств разработки 9](#_Toc163392318)

[3.2. Руководство программиста 9](#_Toc163392319)

[3.2.1. Построение моделей и структура БД. 10](#_Toc163392320)

[3.2.2. Представления 11](#_Toc163392321)

[3.2.3. Шаблоны 12](#_Toc163392322)

[3.2.4. Формы 13](#_Toc163392323)

[3.3. Руководство пользователя 13](#_Toc163392324)

[3.3.1 Страницы и разделы. 14](#_Toc163392325)

[3.3.2 Формы 15](#_Toc163392326)

[4. Тестирование программного продукта 15](#_Toc163392327)

[4.1. Тестовые сценарии 15](#_Toc163392328)

[4.2. Юнит-тесты в Django 16](#_Toc163392329)

[5. Размещение проекта на хостинге. 21](#_Toc163392330)

[Заключение 23](#_Toc163392331)

[Список литературы 24](#_Toc163392332)

[Приложения 25](#_Toc163392333)

# Введение

MemoryMerch представляет из себя интернет-магазин одноименного проекта MemoryMerch. Сайт предназначен для продажи редкой видеоигровой продукции(«мерча»). Данный сайт будет интересен в первую очередь коллекционерам, геймерам а так же просто фанатам тех или иных медиа. На сайте можно создать свой аккаунт а также приобрести выставленные в каталог товары. Товары распределяются по категориям что упрощает их поиск.

# 1. Разработка системного проекта.

## 1.1. Назначение разработки

Целью разработки является создать веб-сайт для проекта MemoryMerch, с возможностью просмотра каталога товаров, добавление товара в корзину и оформление заказа; а также ознакомления с информацией об проекте, контактами данных.

Сайт должен представлять проект «MemoryMerch» в Интернете и служить как онлайн-площадка для продажи продукции.

Цель сайта – познакомить посетителей с проектом, направлениями ее деятельности и товарами, предоставить информацию о способах приобретения товаров.

Пользователи имеют доступ только к общедоступной части сайта. Доступ к административной части имеют пользователи с правами редактора и администратора. Администратор может выполнять такие действия как: добавлять и удалять товары на сайте, добавлять, удалять и редактировать посты в блоге.

Доступ к административной части должен осуществляться с использованием уникального логина и пароля.

## 1.2. Требования к программному продукту

### 1.2.1. Требования к функциональным характеристикам

Сайт предназначен для привлечения новых клиентов на стоматологические услуги и предоставление информации об организации.

Сайт должен соответствовать функциональным требованиям:

* Пользователь должен иметь возможность войти в систему, используя свое имя пользователя и пароль;
* Система запоминает зарегистрированного пользователя;
* Пользователь должен иметь возможность переходить по страницам и оставлять добавлять товары в корзину, заказывать на сайте;
* Администратор может добавлять новые товары, состоящие из названия, фото, цены и описания о товаре.

### 1.2.2. Требования к надежности и безопасности

Чтобы добавлять товары в корзину пользователю необходимо зарегистрироваться на сайте.

Для регистрации пользователь вводит придуманное имя, email(не обязательно), пароль и подтверждает пароль. Для безопасности, к созданию пароля применяются следующие условия:

* Пароль не должен быть слишком похож на другую личную информацию.
* Пароль должен содержать как минимум 8 символов.
* Пароль не должен быть слишком простым и распространенным.
* Пароль не может состоять только из цифр.
* Попадая в базу данных, пароль пользователя хешируется.

В Django для шифрования паролей используется механизм хеширования. Хеш - это строка уникальных символов, которая получается в результате применения математической функции к исходным данным (в данном случае - к паролю). Хеш-функции обеспечивают одностороннее преобразование данных - из хеша невозможно получить исходные данные.

### 1.2.3. Требования к технологиям и инструментам для вёрстки

Для верстки данного проекта использовалась программа VS Code, так как она является наиболее удобной для работы с HTML и CSS языками: всплывающие подсказки синтаксиса, встроенный форматировщик.

### 1.2.4. Требования к составу и параметрам технических средств

Программное обеспечение клиентской части должно удовлетворять следующим требованиям:

- Веб-браузер: Internet Explorer 8.0 и выше, или Firefox 3.5 и выше, или Opera 9.5 и выше, или Safari 3.2.1 и выше, или Chrome 2 и выше;

- Операционная система Windows 10 и выше;

- Устройства с разрешением экрана не менее 1280x720 пикселей

- Клавиатура и мышь

Выбирая хостинг-провайдера, нужно обратить внимание на несколько показателей его работы. У него обязательно должны быть: лицензия на оказание услуг, служба поддержки клиентов, каналы широкополосного Интернета, продублированные системы жизнеобеспечения.

# 2. Разработка технического проекта

## 2.1. Обоснование выбора CASE – средств

CASE-средства (Computer-Aided Software Engineering) - это набор инструментов и методик, используемых для упрощения, ускорения и повышения качества процессов создания и сопровождения программного обеспечения.

draw.io (ранее известный как Lucidchart) - это веб-инструмент для построения диаграмм, который позволяет пользователям создавать различные типы диаграмм, такие как блок-схемы, UML-диаграммы, сетевые диаграммы и многое другое.

## 2.2. Проектирование модели данных

### 2.2.1. Диаграмма прецедентов

Диаграмма вариантов использования или диаграмма прецедентов (use case diagram) - это тип графического представления, который иллюстрирует отношения между актерами и прецедентами. Она помогает в определении требований к системе, анализе поведения системы и выявлении проблем и противоречий в проектировании.

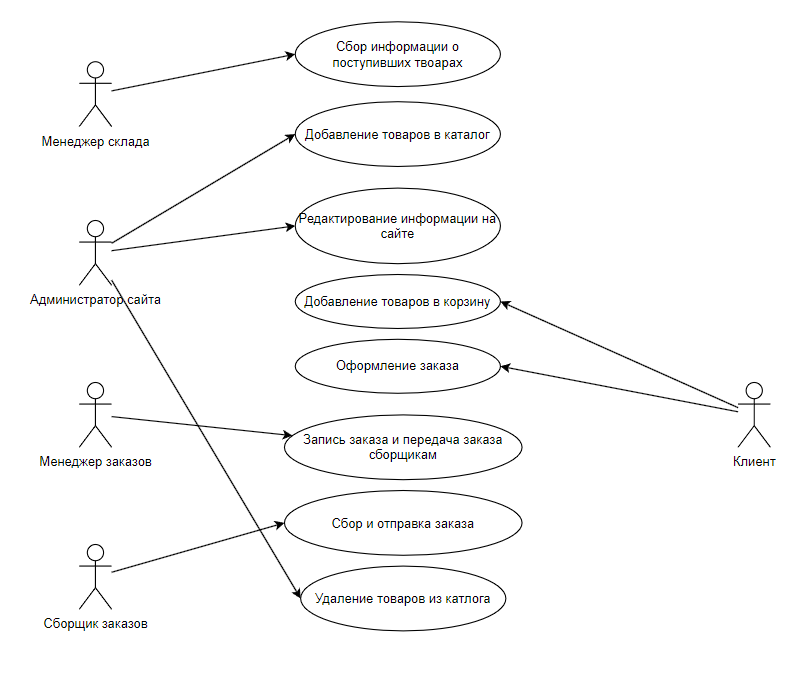


Рисунок 1 Диаграмма прецедентов

Менеджер склада собирает информацию о товаре и передает ее администратору сайта. Администратор сайта заносит информацию о товаре на сайт и добавляет товар в каталог. Клиент добавляет товар в корзину и при желании оформляет заказ. Менеджер заказа берет информацию из заказа и передает ее сборщику заказов. Сборщик собирает заказ и отправляет клиенту.

### 2.2.2. Моделирование бизнес-процессов

Для того чтобы выделить бизнес-процессы, необходимо выделить действия, которые совершает организация. В рассматриваемом случае компания формирует сайт.

Таблица 1 Бизнесс процессы

|  |  |
| --- | --- |
| **Номер бизнес-процесса** | **Название бизнес-процесса** |
| 1-Добав\_товар | Добавление товара |
| 2-Заказ\_оформ | Оформление заказа |
| 3-Заказ\_отправк | Сбор и отправка заказа |
| 4-Просм\_сайта | Просмотр сайта |

### 2.2.3. Словесный алгоритм бизнес-процессов

Процесс добавления товара на сайт.

1. Собирает информацию о товаре
2. Заносит информацию о товаре на сайт
3. Имеет возможность посмотреть и заказать товар на сайте

### 2.2.4. Построение диаграммы действий



Рисунок 2 Диаграмма действий

Все операции, участвующие в процессе «Размещение доступного товара на сайте», отражены в таблице описания опе­раций, имеющей следующий формат:

Таблица 2 Таблица операций

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Диаграмма и номер операции на диаграмме | Операция | Исполнитель | Как часто | Входящие  документы  (документы-  основания) | Исходящий  документ  (составляе­мый  документ) |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1-Добав\_товар\_1 | 1 Собирает информацию о названии и цене услуги | Менеджер склада | Еженедельно по плану | Список товаров | Нет |
| 1-Добав\_товар\_2 | 2 Заносит информацию о товаре на сайт | Администратор сайта | Еженедельно по плану | Список товаров | Нет |
| 1-Добав\_товар\_3 | 3 Имеет возможность посмотреть и заказать товар на сайте | Пользователь | По мере необходимости | Нет | Нет |

# 3. Реализация приложения

## 3.1. Обоснование выбора средств разработки

В рамках курсового проекта было принято решение разработать сайт, сайт по продаже мебели. Учитывая возможности, оборудования и программного обеспечения, необходимо создать современный программный продукт, избегая таких недостатков существующих коммерческих предложений, как высокая стоимость внедрения и сопровождения и слабая ориентированность на пользователя с разной профессиональной подготовкой. Также необходимо уделить особое внимание надежности приложения и простоте его интерфейса.

Поэтому для разработки автоматизированной системы учета продукции были выбраны:

1 Sql lite studio

2. Среда VSCode и язык программирования Python.

Для разработки изначально использовалась программа PyCharm однако большая часть позже была написана именно в VS Code. VS Code представляет удобные средства работы с HTML CSS а также с языком Python при установке бесплатного расширения из каталога. Таким образом в одной программе было удобно работать со всеми элементами проекта. Бекендом на Python и фронтендом на HTML и CSS.

## 3.2. Руководство программиста

Структура приложения в обозревателе решений (см. Приложние 2)

### 3.2.1. Построение моделей и структура БД.

Дать описать моделей и составить схему БД.

* **Приложение cart (Корзина и Заказы):**

class CartItem(models.Model):

    tovar = models.ForeignKey(Tovar, on\_delete=models.CASCADE)

    quantity = models.PositiveIntegerField(default=0)

    user = models.ForeignKey(User, on\_delete=models.CASCADE)

    date\_added = models.DateTimeField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        return f"{self.quantity} x {self.tovar.tovarname}"

class Order(models.Model):

    user = models.ForeignKey(User, on\_delete=models.CASCADE)

    date\_created = models.DateTimeField(auto\_now\_add=True)

    cart\_items = models.ManyToManyField(CartItem)

    def \_\_str\_\_(self):

        return f"Заказ для {self.user.username}"

* **Приложение main(Товары, Посты, Категории):**

class Categories(models.Model):

    catgname = models.CharField(max\_length=60, blank=False, default="")

    def \_\_str\_\_(self):

        return self.catgname

class Tovar(models.Model):

    tovarname = models.CharField(max\_length=100, blank=False, default="")

    tovarprice = models.FloatField()

    category = models.ForeignKey(

        "Categories",

        related\_name="tovars\_categories",

        on\_delete=models.CASCADE,

        null=True,

    )

    shipping = models.TextField(max\_length=300, blank=False, default="")

    tovardescrpt = models.TextField(max\_length=500, blank=False, default="")

    date\_posted = models.DateTimeField(default=timezone.now, null=True)

    tovarimage = models.ImageField(

        upload\_to="img/tovarimages",

        verbose\_name="Фото товара",

        default=None,

        blank=False,

    )

    def \_\_str\_\_(self):

        return f"{self.tovarname} {self.category}"

class Post(models.Model):

    title = models.CharField(max\_length=100, blank=False)

    postbody = models.TextField(max\_length=1000)

    def \_\_str\_\_(self):

        return self.title

### Представления

В файлах views.py находятся несколько обработчиков:

* **Представление добавления товара:**

def createtovar(request):

    error = ""

    if request.method == "POST":

        form = TovarForm(request.POST, request.FILES)

        if form.is\_valid():

            form.save()

            return redirect("index")

        else:

            error = "Форма заполнена неправильно или не до конца"

    form = TovarForm()

    context = {"form": form}

    return render(request, "main/create-tovar.html", context)

* **Представление Главной страницы**

def index(request):

    tovars\_all = (Tovar.objects.all(),)

    posts = Post.objects.all()

    filter\_tovar = TovarFilter(request.GET, queryset=Tovar.objects.all())

    return render(request, "main/index.html", {"tovars": filter\_tovar, "posts": posts})

### 3.2.3. Шаблоны

Описать какие шаблоны есть. Можно рассказать про структуру шаблона на примере одного из.

Для упрощения верстки я использовала возможности встроенного шаблонизатора jinja. Был создан «базовый» файл-шаблон верстки base.html, включающий в себя хедер и футер сайта. В данный шаблон я далее подключала различное наполнение и контент с помощью «{% block название блока %}{% endblock %}».

Код базового шаблона base.html можно посмотреть в приложении.

{% if user.is\_authenticated %} – условие если посетитель зарегистрировался

{% else %} – иначе если пользователь не зарегистрирован

{% endif %} – условие закончилось.

### 3.2.4. Формы

Код форм описан в файле forms.py. Формы создаются с помощью класса ModelForm

Форма добавления товара:

class TovarForm(ModelForm):

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

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

        self.fields['category'].empty\_label = 'Категория не выбрана'

    category = forms.ModelChoiceField(queryset=Categories.objects.all(), empty\_label="Категория", required=True)

    class Meta:

        model = Tovar

        fields = ["tovarname", 'tovarprice', 'category', 'shipping', 'tovardescrpt','tovarimage']

        widgets = {

            "tovarname": TextInput(attrs={'placeholder': 'Заголовок товара', 'class': 'form-input'}),

            "tovarprice": TextInput(attrs={'placeholder': 'Цена', 'class': 'form-input'}),

            "category": forms.Select(attrs={'placeholder': 'Категория', 'class': 'form-input'}),

            "shipping": Textarea(attrs={'placeholder': 'Доставка', 'class': 'form-input'}),

            "tovardescrpt": Textarea(attrs={'placeholder': 'Фото товара', 'class': 'form-input'}),

        }

Формы аутентификации создавалась автоматически библиотекой allauth.

## 3.3. Руководство пользователя

При загрузке страницы вы окажитесь на главной странице сайта.

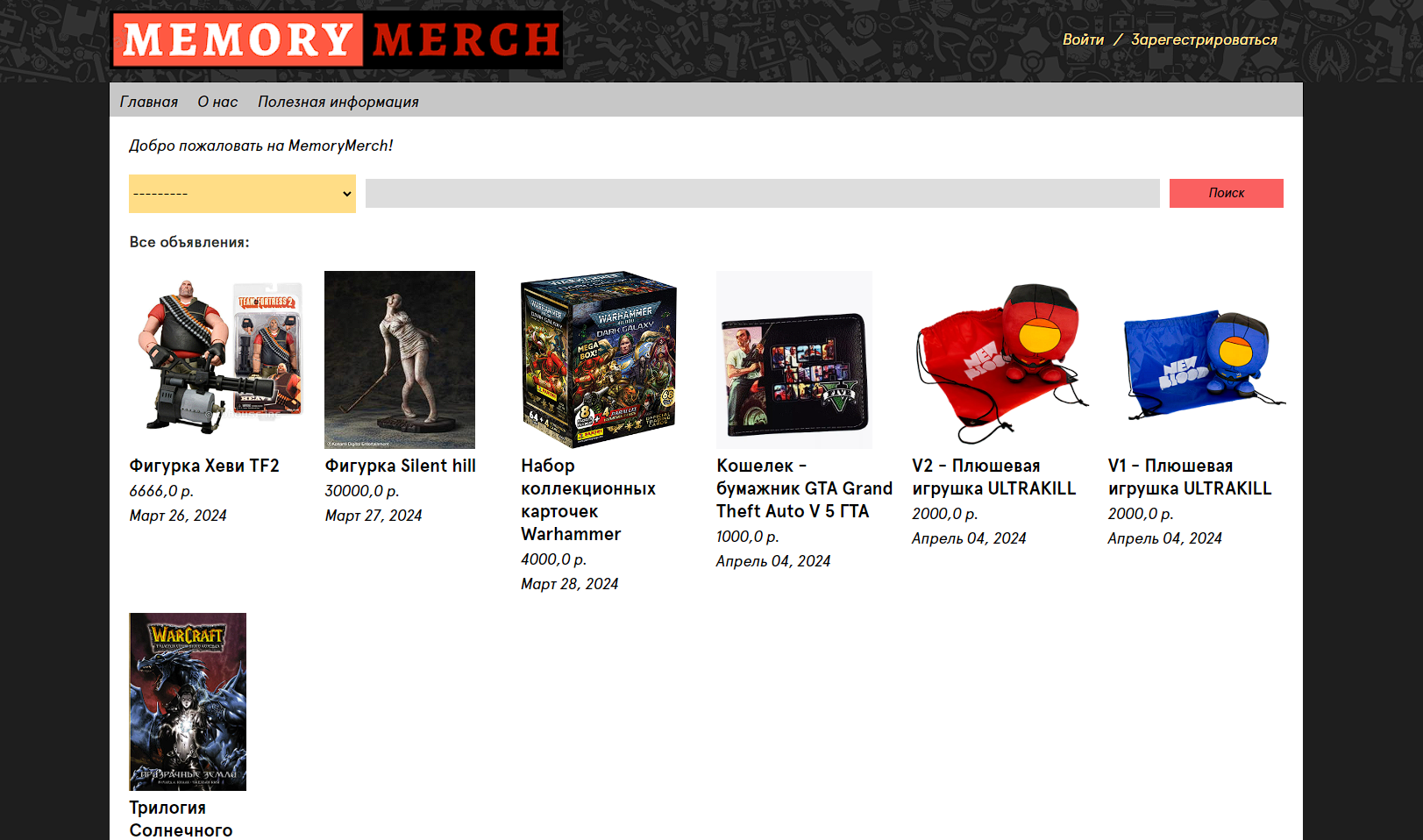


Рисунок 3. Главная страница сайта

Навигация по сайту осуществляется с помощью пунктов меню, расположенных в шапке страницы.

### 3.3.1 Страницы и разделы.

Далее в таблице представлены все разработанные страницы сайта.

|  |  |
| --- | --- |
| Главная. На Главной странице выводятся товары доступные из каталога. Также ниже выводятся новости из блога. |  |
| О нас. На странице находиться информация о проекте. |  |
| Полезная информация. На данной странице располагается различная полезная для клиентов информация и нюансы. |  |
| Страница товара. На странице товара выводится информация о товаре. Название, категория, дата добавления, фото, информация о доставке, описание и кнопка добавления в корзину. |  |

### 3.3.2 Формы

|  |  |
| --- | --- |
| Форма Регистрации. |  |
| Форма Входа. |  |
| Сообщение выхода |  |
| Форма добавления товара. |  |

# 4. Тестирование программного продукта

## 4.1. Тестовые сценарии

Таблица 3 Тестовые сценарии

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **Описание** | **Шаги** | **Входные данные** | **Ожидаемые результаты** | **Фактические результаты** | **Статусы** |
| А1 | Проверка регистрации пользователя | Откройте сайт  Нажмите на «Зарегистрироваться» в шапке  Введите данные  Нажмите кнопку «Зарегистрироваться» | Имя = Sofie  Пароль = PdE-MJ5-tDF-5Te | Пользователь должен попасть на главную страницу | Как ожидали | Успешно пройден. |
| А2 | Проверка входа пользователя с существующим логином и паролем | Откройте сайт  Нажмите на «Войти» в шапке  Введите данные  Нажмите кнопку «Войти» | Имя = Sofie  Пароль = PdE-MJ5-tDF-5Te | Пользователь должен авторизоваться и попасть на главную страницу | Как ожидали | Успешно пройден. |
| А3 | Проверка входа пользователя с не существующим логином и паролем | Откройте сайт  Нажмите на «Войти» в шапке  Введите данные  Нажмите кнопку «Войти» | Имя = Danya  Пароль = 3453453456ggrr | Пользователь должен получить сообщение о неверности логина\пароля. | Как ожидали | Успешно пройден. |
| Т1 | Проверка добавления товара в каталог | Откройте сайт  Войдите под учетной записью Администратора  В шапке сайта нажмите «Добавить товар»  Введите данные  Нажмите на кнопку «Добавить» | Заголовок товара=  Игрушка Криппер  Цена=  300  Категория=  Фигурки \ Игрушки Доставка, описание, фото = любое | Администратор должен попасть на главную страницу, товар отобразиться в каталоге | Как ожидали | Успешно пройден. |
| Т2 | Проверка добавления товара в корзину | Откройте сайт  Войдите под любой учетной записью  Перейдите на страницу товара  Нажмите на кнопку «Добавить в корзину» | - | Пользователь должен остаться на странице товара, добавленный товар отобразиться в корзине. | Как ожидали | Успешно пройден. |

## 4.2. Юнит-тесты в Django

Юнит-тест (unit test), или модульный тест, — это программа, которая проверяет работу небольшой части кода. Разработчики регулярно обновляют сайты и приложения, добавляют фичи, рефакторят код и вносят правки, а затем проверяют, как всё работает.

Тестировать систему целиком после каждого обновления — довольно муторно и неэффективно. Поэтому обновлённые или исправленные части кода прогоняют через юнит-тесты.

Таблица 4. Тесты

|  |  |
| --- | --- |
| Тест | Описание |
| class NameFunctionalTest(LiveServerTestCase):  def setUp(self):  self.selenium = webdriver.Edge()  super().setUp()  def tearDown(self):  self.selenium.quit()  super().tearDown()  def test\_product\_list\_functional(self):  # Create test data  Tovar.objects.create(  tovarname="Футболка Варкарафт",  tovartovarprice=600,  shipping="3 дня",  category="Одежда",  tovardescrpt="Описание футболки",  tovarimage="{% static 'img/fortest/test2.jpg' %}"  )  Tovar.objects.create(  tovarname="Кошелек Майнкрафт",  tovartovarprice=200,  shipping="3 дня",  category="Аксессуары",  tovardescrpt="Описание кошелька",  tovarimage="{% static 'img/fortest/test2.jpg' %}"  )  # Simulate user interactions using Selenium  self.selenium.get(self.live\_server\_url)  self.assertIn("Главная страница", self.selenium.title)  names = self.selenium.find\_elements(By.CLASS\_NAME, "media")  self.assertEqual(len(names), 2)  self.assertEqual(names[0].text.split("\n")[0], "Футболка Варкарафт")  self.assertEqual(names[1].text.split("\n")[0], "Кошелек Майнкрафт") | В этом тесте создаются два объекта Tovar (предполагается, что это модель продукта в приложении блога Django). Затем тест запускает браузер Edge и открывает веб-страницу приложения. После этого проверяется наличие заголовка страницы и количество элементов на странице, соответствующих продуктам. Также проверяется, что названия продуктов соответствуют созданным в базе данных.  Этот тест проверяет функциональность представления списка продуктов в веб-приложении. Если все условия выполняются, тест считается пройденным. |
| from unittest import skip  from django.test import TestCase, RequestFactory  from unittest.mock import Mock, patch  from main.views import index  class TovarListViewTest(TestCase):  def setUp(self):  self.factory = RequestFactory()  def test\_name\_list\_view(self):  Tovar.objects.create(title='Футболка Варкарафт', tovarprice=600)  Tovar.objects.create(title='Кошелек Майнкрафт', tovarprice=200)  request = self.factory.get('/index/')  mock\_queryset = Mock(spec=Tovar.objects.all())  mock\_queryset.return\_value = [  Mock(title='Футболка Варкарафт', tovarprice=600),  Mock(title='Кошелек Майнкрафт', tovarprice=200)  ]  with patch('main.views.Tovar.objects.all', mock\_queryset):  response = index(request)  self.assertEqual(response.status\_code, 200) | Этот код является частью теста для проверки работы представления списка товаров (index) в Django-приложении. В нем создаются два объекта Tovar (товара) в базе данных, затем генерируется запрос к представлению index. С помощью моков (unittest.mock) имитируется поведение запроса к базе данных, чтобы убедиться, что представление возвращает ожидаемый ответ.  В конце теста проверяется статус ответа представления, который должен быть равен 200 (OK). Если все условия выполнены, тест считается успешным. |
| from django.test import TestCase  from main.models import Tovar  class TovarModelTest(TestCase):    @classmethod  def setUpTestData(cls):  Tovar.objects.create(title='Футболка Варкарафт', tovarprice=600,)  def test\_first\_name\_label(self):  Tovar = Tovar.objects.get(id=1)  field\_label = Tovar.\_meta.get\_field('tovarname').verbose\_name  self.assertEqual(field\_label, 'Наименование')  def test\_phone\_label(self):  Tovar = Tovar.objects.get(id=1)  field\_label = Tovar.\_meta.get\_field('tovardescrpt').verbose\_name  self.assertEqual(field\_label, 'Описание товара')  def test\_surname\_max\_length(self):  Tovar = Tovar.objects.get(id=1)  max\_length = Tovar.\_meta.get\_field('tovartovarprice').max\_length  self.assertEqual(max\_length, 50) | Этот код представляет собой часть теста модели Goog в Django-приложении. В нем проверяются различные аспекты модели, такие как метки полей, максимальная длина поля и т.д. Каждый тест начинается с получения экземпляра модели Tovar с идентификатором 1 (который был создан в методе setUpTestData), затем проверяется соответствующее свойство модели.  Например, в тесте test\_first\_name\_label проверяется, что метка поля title соответствует ожидаемому значению Название товара. В тесте test\_phone\_label аналогично проверяется метка поля description, а в тесте test\_surname\_max\_length проверяется максимальная длина поля tovarprice. |
| from unittest import skip  from django.test import TestCase  from main.models import Tovar  class MyModelTest(TestCase):  def setUp(self):  self.object = Tovar.objects.create(title="Футболка Майнкрафт", tovarprice=600)  def test\_str\_representation(self):  self.assertEqual(str(self.object), 'Футболка Майнкрафт')  def tearDown(self):  pass | Этот код представляет собой часть теста модели Goog в Django-приложении. В тесте test\_str\_representation проверяется, что строковое представление объекта модели Tovar соответствует ожидаемому значению. Это важно, потому что в Django строковое представление объекта часто используется при выводе объектов в шаблонах, и оно должно быть логичным и понятным.  В методе setUp создается экземпляр модели Tovar с заданными параметрами, который затем используется в тесте. Метод tearDown обычно используется для очистки ресурсов после выполнения теста, но в данном случае он пустой, что может быть связано с тем, что никаких дополнительных действий по очистке не требуется. |
| from unittest import skip  from django.test import TestCase  from django.urls import reverse  class SimpleTests(TestCase):  def test\_home\_page\_status\_code(self):  url = reverse('index')  response = self.client.get(url)  self.assertEqual(response.status\_code, 200)  def test\_home\_template\_used(self):  url = reverse('index')  response = self.client.get(url)  self.assertTemplateUsed(response, 'main/index.html') | Эти тесты проверяют работу главной страницы блога в Django-приложении. Тест test\_home\_page\_status\_code проверяет, что статус ответа на запрос главной страницы равен 200 (OK). Тест test\_home\_template\_used проверяет, что для отображения главной страницы используется шаблон blog/index.html. |
| from unittest import skip  from django.test import TransactionTestCase  from main.models import Tovar  class WidgetTransactionTestCase(TransactionTestCase):  def test\_widget\_creation(self):  Tovar.objects.create(title='Футболка Майнкрафт', tovarprice=100)  Tovar.objects.create(title='Консультация', tovarprice=120)  self.assertEqual(Tovar.objects.count(), 2) | Этот тестовый класс использует TransactionTestCase из модуля django.test, что позволяет ему выполнять транзакции базы данных внутри блока with transaction.atomic(). Это означает, что если в ходе выполнения теста произойдет ошибка, все внесенные изменения будут откачены, и база данных останется в исходном состоянии.  В тесте test\_widget\_creation создаются два объекта Tovar (предположительно модели товара в приложении блога Django), и проверяется, что их количество равно двум. Это простой способ проверить, что объекты были успешно сохранены в базе данных. |

# 5. Размещение проекта на хостинге.

Регистрация на хостинге.

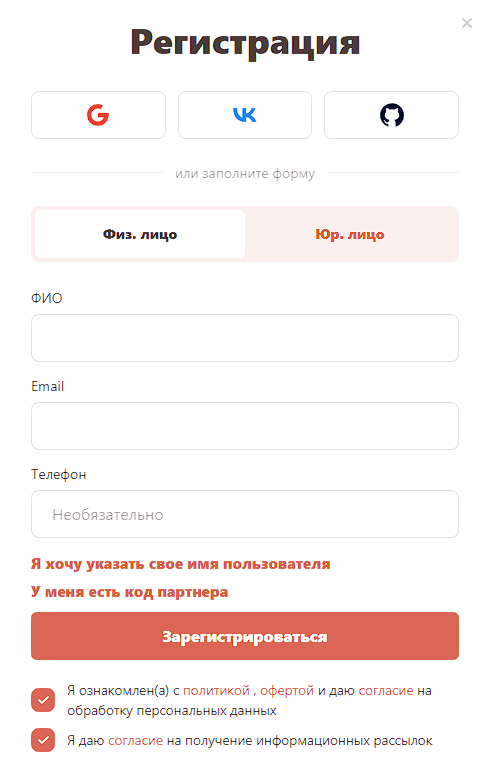


Рисунок 4 Регистрация на хостинге

В списке доменов у вас должен появиться домен

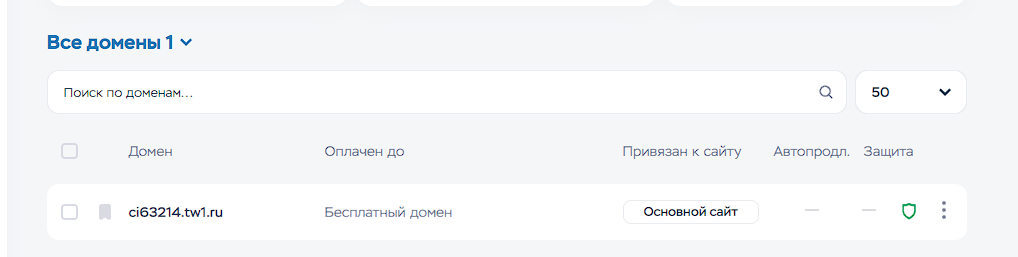


Рисунок 5 Домен

Перейдите на главную страницу и включите доступ по SSH:

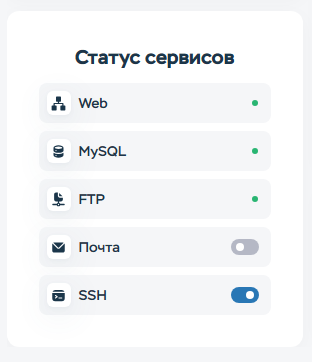


Рисунок 6 «SHH» подключние

Скачайте программу putty.

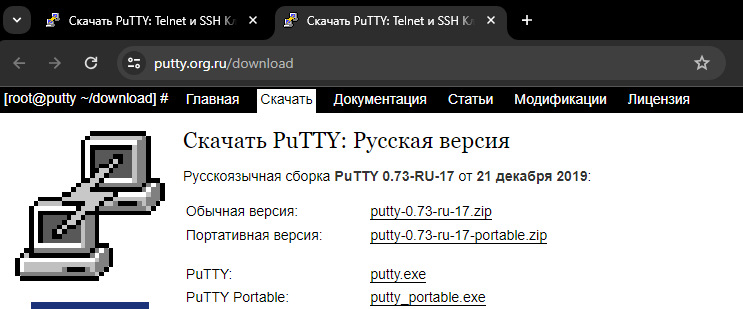


Рисунок 7 Скачивание Putty

Запустите программу

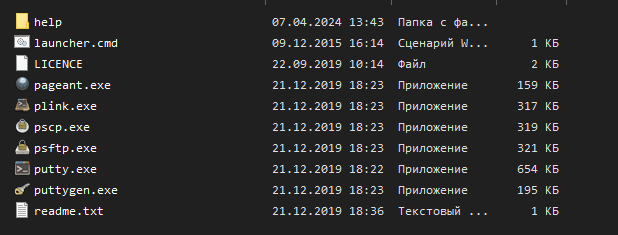


Рисунок 8 Программа в проводнике

В подключении укажите домен:

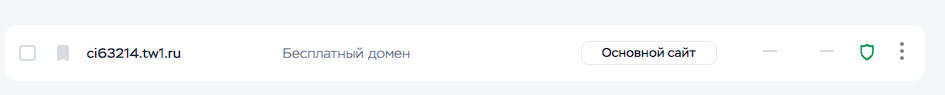


Рисунок 9 Домен

Затем введите логин и пароль в командную строку, они пришли вам на почту.

После начните установку виртуального окружения.

# Заключение

В данной курсовой работе создан сайт «MemoryMerch». Для достижения цели мы использовали свой заранее разработанный готовый дизайн. Проект создан в среде VSCode на основе фреймворка django. В результате выполнения курсовой работы по разработке сайта использованы CASE-средство для создания диаграммы прецедентов и действий, построены представления о вёрстке проектов, сформированы функциональная и каркасная архитектура сайта.

При разработке сайта освоены язык программирования Python, язык гиперразметки HTML, каскадные таблицы стилей СSS и SCSS и программа VSCode.

Чтобы обеспечить уверенность в качестве приложения выполнены тестовые сценарии. Документирование позволило рассмотреть сайта с точки зрения программиста, пользователя и тестировщика.

# Список литературы

Интернет-ресурсы:

1. Руководство по HTML5 и CSS3. https://metanit.com/web/html5/?ysclid=lucadgeqih655506976
2. Django тестирование. https://www.yourtodo.ru/posts/django-testirovanie/
3. “Библия QA” - обновляемая база знаний объемом 560+ страниц: https://vladislaveremeev.gitbook.io/qa\_bible
4. https://www.youtube.com/watch?v=BWtrVIVgA6c
5. <https://dzen.ru/a/YpM-PrHXSDUgX2_0>
6. https://pocoz.gitbooks.io/django-v-primerah/content/glava-7-sozdanie-internet-magazina/sozdanie-korzini.html

# Приложения

Приложение 1. Код шаблона base.html

{% load static %}

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="{% static 'css/styles.css'%}" type="text/css">

    <title>Memorymerch | {% block title%}{% endblock %}</title>

</head>

<body>

    <header>

        <div class="header-wrapper">

            <div class="logo"><a href="{% url 'index' %}"><img src="{% static 'img/logo.png' %}" alt="logo"></a></div>

            {% if user.is\_authenticated %}

            <form action="{% url 'create-tovar' %}" target="\_blank">

                <button id="createOB">Добавить товар</button>

            </form>

            {% else %}

            {% endif %}

            <div id="userlog">

                {% if user.is\_authenticated %}

                {{ user.username }}

                <a class="main-header\_links" href="{% url 'cart:view\_cart' %}" id="avatar"><img

                        src="{% static 'img/cart.png' %}" alt="avatar"></a> |

                <a class="main-header\_links" href="{% url 'account\_logout' %}">Выйти</a>

                {% else %}

                <a class="main-header\_links" href="{% url 'account\_login' %}">Войти</a> / <a

                    href="{% url 'account\_signup' %}" class="main-header\_links">Зарегестрироваться</a>

                {% endif %}

            </div>

        </div>

    </header>

    <div class="wrapper">

        <!-- навигация -->

        <div class="main-nav">

            <li><a href="{% url 'index' %}">Главная</a></li>

            <li><a href="{% url 'about' %}">О нас</a></li>

            <li><a href="{% url 'rules' %}">Полезная информация</a></li>

        </div>

        <!-- контент -->

        <div class="container">

            <div class="content">

                {% if messages %}

                {% for message in messages %}

                <div class="alert alert-{{ message.tags }}">

                    {{ message }}

                </div>

                {% endfor %}

                {% endif %}

                {% block content %}{% endblock %}

            </div>

        </div>

    </div>

    <footer>

        <div class="footer-wrapper">

            <div class="footer-grid">

                <div class="footer\_\_c">

                    <p>(с) MEMORYMERCH 2024 <br>

                        Все права защищены</p>

                </div>

                <div class="footer\_\_socials">

                    <a href=""><img src="{% static 'img/socials\_vk.png' %}" alt="social-1"></a>

                    <a href=""><img src="{% static 'img/socials\_tg.png' %}" alt="social-2"></a>

                    <a href=""><img src="{% static 'img/socials\_ok.png' %}" alt="social-3"></a>

                </div>

                <div class="footer\_contacts">

                    <p><span id="footer\_\_phone">+ 7 999 999 99 99</span> - 11:00 - 18:00 ПН-ПТ</p>

                    <p id="footer\_\_email">shop.memorymerch.support@mm.ru</p>

                </div>

            </div>

        </div>

    </footer>

</body>

</html>

Приложение 2 «Структура в обозревателе решений»

