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 сайта для организации, занимающейся продажей товаров народного потребления»

МДК.08.01 «Проектирование и разработка интерфейсов пользователя»

**Выполнил:**

Галанин Денис Сергеевич

Студент 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

**Оглавление**

[Введение 2](#_Toc162870301)

[1. Разработка системного проекта 3](#_Toc162870302)

[1.1 Назначение разработки 3](#_Toc162870303)

[1.2 Требования к программному продукту 4](#_Toc162870304)

[1.2.1 Требования к функциональным характеристикам. 4](#_Toc162870305)

[1.2.2 Требования к надежности и безопасности 4](#_Toc162870306)

[1.2.3 Требования к технологиям и инструментам для вёрстки 4](#_Toc162870307)

[1.2.4 Требования к составу и параметрам технических средств 5](#_Toc162870308)

[2. Разработка технического проекта 5](#_Toc162870309)

[2.1 Обоснование выбора CASE – средств 5](#_Toc162870310)

[2.2 Проектирование модели данных 6](#_Toc162870311)

[2.2.1 Диаграмма прецедентов 6](#_Toc162870312)

[2.2.2 Моделирование бизнес-процессов 6](#_Toc162870313)

[2.2.3 Словесный алгоритм бизнес-процессов 7](#_Toc162870314)

[2.2.4 Построение диаграммы действий 7](#_Toc162870315)

[3. Реализация 9](#_Toc162870316)

[3.1. Обоснование выбора средств разработки 9](#_Toc162870317)

[3.2 . Руководство программиста 9](#_Toc162870318)

[3.2.1 Построение моделей и структура БД 10](#_Toc162870319)

[3.2.2 Представления 11](#_Toc162870320)

[3.2.3 Шаблоны 11](#_Toc162870321)

[3.2.4 Формы 13](#_Toc162870322)

[3.3. Руководство пользователя 13](#_Toc162870323)

[4. Тестирование программного продукта 19](#_Toc162870324)

[4.1 Тестовые сценарии 19](#_Toc162870325)

[4.2 Юнит-тесты в Django 21](#_Toc162870326)

[5. Размещение проекта на хостинге 31](#_Toc162870327)

[Заключение 34](#_Toc162870328)

[Приложение 36](#_Toc162870329)

[Список литературы 37](#_Toc162870330)

# Введение

Товары народного потребления - это товары и продукты, которые широко востребованы и потребляются массовым населением. Они обычно относятся к базовым или повседневным потребностям людей и включают широкий спектр товаров и услуг.

Такие товары включают продукты питания, одежду, обувь, бытовую технику, мебель, предметы гигиены, товары для дома, электронику, товары для развлечений и многие другие. Они представляют собой необходимые товары и услуги, которые люди регулярно покупают для удовлетворения своих потребностей и жизненного комфорта.

Товары народного потребления часто имеют доступные цены и они производятся массово, чтобы удовлетворить спрос большого количества потребителей. Они также могут быть доступны в различных вариантах и моделях, чтобы соответствовать разным потребностям и предпочтениям потребителей.

Товары народного потребления являются важным сегментом рынка для производителей и продавцов, так как они имеют постоянный и стабильный спрос со стороны широкой аудитории потребителей.

Целью курсовой работы является изучение предметной области в сфере товаров народного потребления, а также разработка автоматизированного приложения для облегчения работы сотрудникам на предприятии.

# 1. Разработка системного проекта

## **1.1 Назначение разработки**

Сайт «ТНП» предназначен для содержания подробной информации о продуктах, их характеристиках, ценах, наличии и других релевантных сведениях. Посетители могут легко ознакомиться с ассортиментом товаров и сделать информированный выбор.

Удобное онлайн-покупка: Разработка сайта с функцией онлайн-магазина позволяет пользователям совершать покупки прямо через интернет. Это удобно и экономит время потребителей, а также расширяет географию потенциальных клиентов.

Улучшение потребительского опыта: Сайт может предоставить дополнительные функциональности, такие как удобный поиск, фильтры, отзывы клиентов, что помогает пользователям найти нужный продукт и сделать осознанный выбор.

## **1.2 Требования к программному продукту**

## **1.2.1 Требования к функциональным характеристикам.**

Сайт «ТНП» должен обеспечивать выполнение функций:

* Удобная навигация
* Каталог товаров
* Онлайн-магазин
* Аутентификация пользователя
* Отзывы и рейтинги

**1.2.2 Требования к надежности и безопасности**

Сайт должен быть в достаточной степени надёжна от сбоев.

Предусматривается: защита персональных данных, защита от вредоносных атак, резервное копирование данных, сильные пароли и аутентификация.

**1.2.3 Требования к технологиям и инструментам для вёрстки**

HTML и CSS: HTML (HyperText Markup Language) используется для структурирования содержимого веб-страницы, а CSS (Cascading Style Sheets) определяет внешний вид и стилизацию элементов страницы. Они являются основными языками для веб-верстки.

JavaScript: JavaScript является клиентским языком программирования и обеспечивает интерактивность и динамическое поведение на веб-страницах. Он может использоваться для создания веб-приложений, анимаций, валидации форм и других функциональностей.

**1.2.4 Требования к составу и параметрам технических средств**

Таблица №1 «Системные требования»

|  |  |
| --- | --- |
| Процессор | Intel Core i3-3470 Ivy Bridge LGA1155 |
| Оперативная Память | 12 ГБ |
| Разрешение экрана | 1920x1080 |
| Размер экрана | От 19 дюймов |
| Устройства ввода | Мышь и клавиатура |
| Дисковое пространство | 4 Гб |
| Операционная система | Windows 7,10,XP |

**Требования к хосту**

Таблица №2 «Системные требования для хостинга»

|  |  |
| --- | --- |
| Процессор | Intel Core i3-3470 Ivy Bridge LGA1155 |
| Оперативная Память | 8 ГБ |
| Разрешение экрана | 1920x1080 |
| Размер экрана | От 17 дюймов |
| Устройства ввода | Мышь и клавиатура |
| Дисковое пространство | 8 Гб |
| Операционная система | Windows 7,8,10,XP |

# Разработка технического проекта

CASE – средства (Computer – Aided Software Engineering) – это методы и технологии, которые позволяют проектировать различные информационные системы и автоматизировать их создание.

CASE – технологии – программная основа CASE-средств, применяемая для разработки и поддержки процессов жизненных циклов ПО, используемых в моделировании данных и генерации схем баз данных.

## **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 - 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**

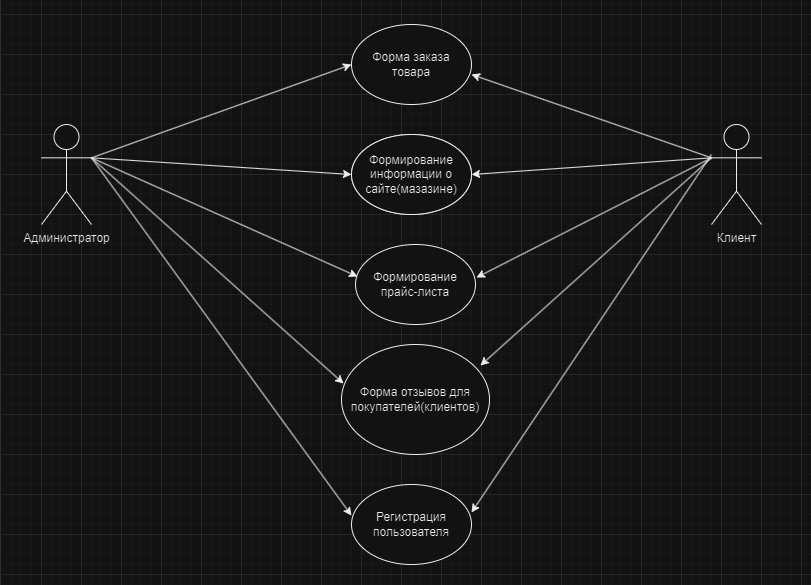
Диаграмма вариантов использования или диаграмма прецедентов (use case diagram) - это тип графического представления, который иллюстрирует отношения между актерами и прецедентами.

Рис.1 «диаграмма прецедентов»

**2.2.2 Моделирование бизнес-процессов**

Для того чтобы выделить бизнес-процессы, необходимо выделить действия, которые совершает организация и клиент (потенциальный покупатель, заказчик). В рассматриваемом случае демонстрируется регистрация, где клиент вводит свои данные, а компания проверяет их на достоверность и правильность. Также можно рассмотреть формирование заказа, его изготовление и проверку по ГОСТу.

Пример заполнения таблицы бизнес-процессов.

Таблица №2. «Бизнес-процессы»

|  |  |
| --- | --- |
| **Номер бизнес-процесса** | **Название бизнес-процесса** |
| 1.Форм\_заказ\_товар | Форма заказа товара |
| 2.Форм\_инфо\_о\_сайт | Формирование информации о сайте(интернет-магазине) |
| 3.Форм\_прайс\_лист | Формирование прайс-листа |
| 4.Форм\_отзыв | Формирование отзывов для покупателей |
| 5.Рег\_польз | Регистрация пользователя |

**2.2.3 Словесный алгоритм бизнес-процессов**

1. Клиент проходит базовую регистрацию(email, имя и пароль с подтверждением пароля).

2. Система принимает запрос на регистрацию и проверяет на правильность и достоверность данные, оставленные клиентом для регистрации.

3. Если данные достоверные и правильно составленные, то у клиента появляется свой личный кабинет(аккаунт)

4. Если система жалуется на правильность или достоверность, то клиента перебрасывает на форму регистрации, где он должен ввести свои данные корректно.

**2.2.4 Построение диаграммы действий**

На основании общего описания бизнес-процесса «Регистрация пользователя (клиента)» составляется диаграмма действий, которая показывает участников процесса, выполняемые каждым участником операции и взаимосвязь между ними. Операции на диаграмме следуют в хронологическом порядке, который определен в приведенном словесном описании бизнес-процесса.

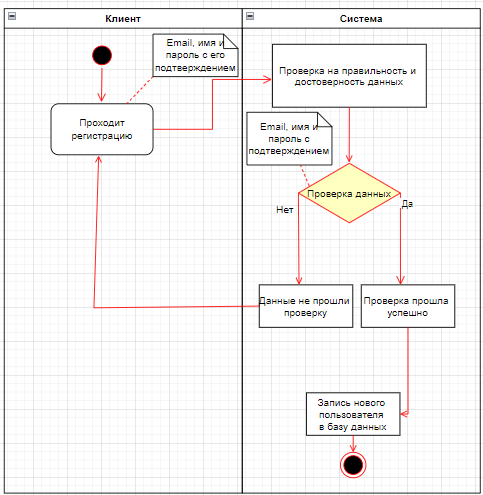


Рис.2 «диаграмма действий»

# Реализация

**3.1. Обоснование выбора средств разработки**

Среда для разработки выбрана Visual Studio Code, т.к. она является бесплатной интегрированной средой разработки и подходит для работы с языками программирования такими как HTML, JavaScript. Она предоставляет широкий спектр функций для разработчиков, например подсветка синтаксиса, автозавершение кода, отладка.

* 1. **. Руководство программиста**

Структура приложения в обозревателе решений:

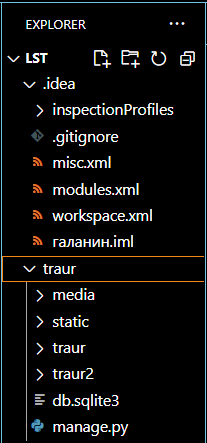


Рис.3 «Структура»

Таблица №3 «Описание папок

в обозреватели решений»

|  |  |
| --- | --- |
| **Папка** | **Вид** |
| В папке static-img хранятся все используемы изображения, а также шрифты и иконка приложения. |  |
| В папке traur2 хранятся настройки. |  |
| В папке templates хранятся шаблоны страниц и их привязки. |  |
| В папке migrations хранятся все миграции, производимые над проектом |  |
| В папке css хранятся стили сайта. |  |

* + 1. **Построение моделей и структура БД**

Таблица №4 «Структура БД»

|  |  |
| --- | --- |
| **Файл** | **Вид** |
| В файле admin.py находится привязка к панели администратора |  |
| В файле forms.py находятся формы проекта |  |
| В файле models.py находятся модели сайта |  |
| В файле urls.py находятся привязки к страницам |  |
| В файле tests.py находятся тесты сайта |  |

* + 1. **Представления**

Views.py играет ключевую роль в обработке запросов и рендеринге страничного контента. Он содержит функции (или классы в виде методов), называемые представлениями (views), которые определяют, как сервер должен обрабатывать конкретные HTTP-запросы и генерировать соответствующий ответ.

from django.shortcuts import render, redirect, get\_object\_or\_404

from django.views import View

from .models import Task,Product

from django.views.generic.detail import DetailView

from .forms import TaskForm

def main(request):

return render(request, 'traur2/main.html')

def reg(request):

return render(request, 'traur2/reg.html')

def catalog(request):

pr = Product.objects.all()

return render(request, 'traur2/catalog.html', {"pr":pr})

**3.2.3 Шаблоны**

В шаблонах находятся 5 страниц:

“base” — это общая страница, которая предоставляет пункты бокового меню, которые будут находится на каждой странице.

“catalog” — это страница, на которой расположен блок с фильтрацией товаров и сами товары с возможностью его удаления и редактирования. Это страница, на которой находиться полный прайс-лист услуг.

“product\_detail”— это страница, на которой находится форма для добавления товара.

“kontact” — это страница, на которой находиться форма отзывов и самих отзывов покупателей

“main” — это страница, но которой находится информация о компании.

Пример шаблона base.html

{% load static %}

<html lang="en">

<head>

  <meta charset="UTF-8" />

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

  <title>Document</title>

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

</head>

<body>

  <header>

    <div class="header\_\_container"><img class="logo" src="{% static 'img/logo\_logo.png'%}" alt="" />

      <ul>

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

        <li> <a href="{% url 'home'%}">О компании</a></li>

        <li> <a href="{% url 'kontact'%}">Отзывы</a></li>

        <li>

          {% if user.is\_authenticated%}

          <span>{{user.username}}</span>

          <a href="{% url 'account\_logout'%}">

            Выход

          </a>

          {% else %}

          <a href="{% url 'account\_login'%}">

            Вход

          </a>

          <a href="{% url 'account\_signup'%}">

            Регистрация

          </a>

          {% endif %}

        </li>

      </ul>

    </div>

  </header>

  {%block content%}{%endblock%}

  <footer>

    <div class="container\_\_footer">

      <div class="about">

        <ul>

          <li><a href="{% url 'kontact'%}">Отзывы</a></li>

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

          <li><a href="{% url 'home'%}">О компании</a></li>

        </ul>

      </div>

      <div class="information">

        <ul>

          <li>TNP@gmail.com</li>

          <li>© 2004—2020 ТД «Товары народного потребления» / BEGET Политика безопасности</li>

        </ul>

      </div>

      <div class="telephone">

        <ul>

          <li>Обратная связь</li>

          <img src="{% static 'img/icon\_vk.png'%}" alt="">

          <img src="{% static 'img/icon\_telegram.png'%}" alt="">

          <img src="{% static 'img/icon\_instagram.png'%}" alt="">

        </ul>

      </div>

    </div>

  </footer>

</body>

</html>

**3.2.4 Формы**

Была добавлена форма с отзывами:

from django.db import models

from django.contrib.auth.models import User

from django.utils import timezone

class Task(models.Model):

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

    date = models.DateField(default=timezone.now)

    text = models.TextField('Комментарий')

    class Meta:

        verbose\_name = 'Отзыв'

        verbose\_name\_plural = 'Отзывы'

**3.3. Руководство пользователя**

Для перехода на «Главная» нужно перейти по ссылке [Главная страница сайта](http://127.0.0.1:8000/), но для пользователя, который находится на сайте достаточно просто нажать на кнопку на боковой панели.

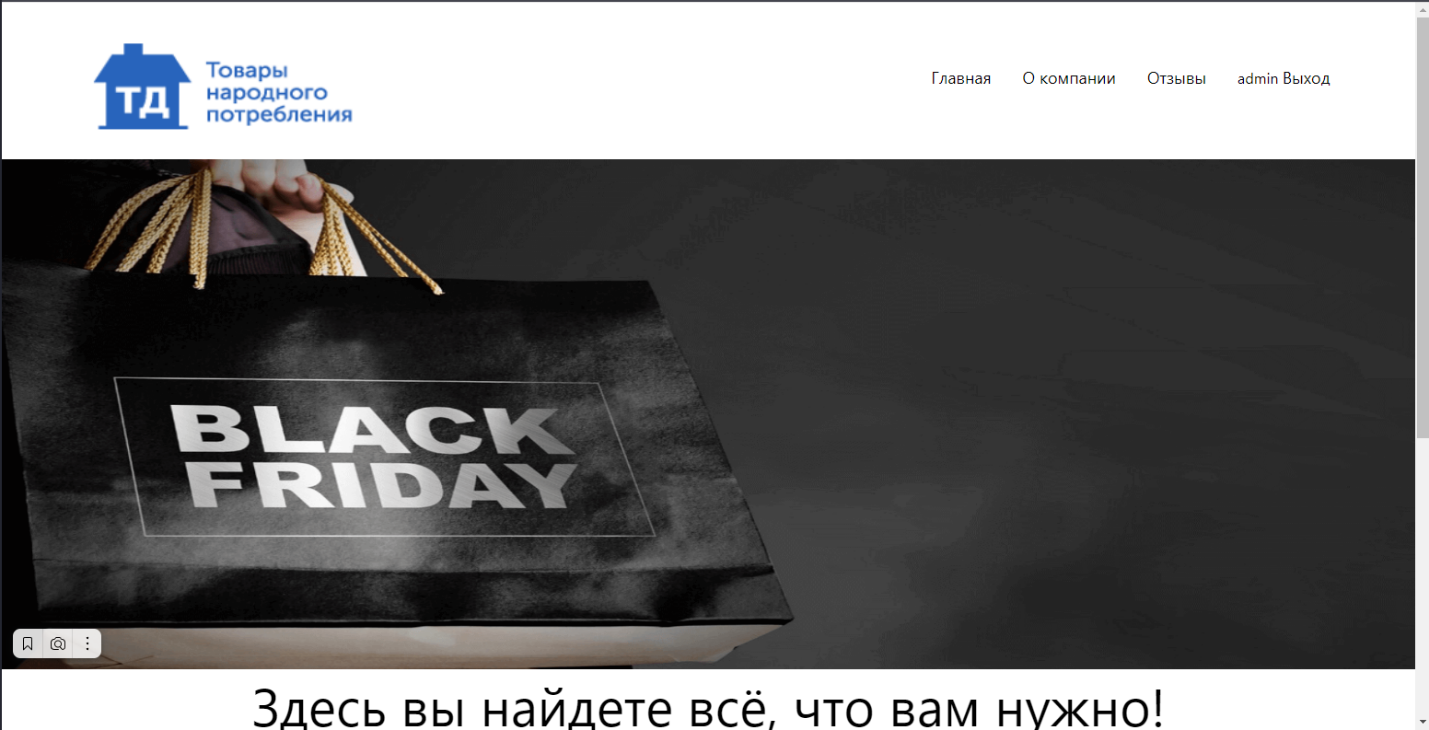
На странице «Главная» пользователю будет предоставлена информация о товарах.

Рис.4.1 «Главная»



Рис.4.2 «Главная»

Для перехода на «О компании» нужно перейти по ссылке [Главная страница сайта](http://127.0.0.1:8000/), но для пользователя, который находится на сайте достаточно просто нажать на кнопку на боковой панели.

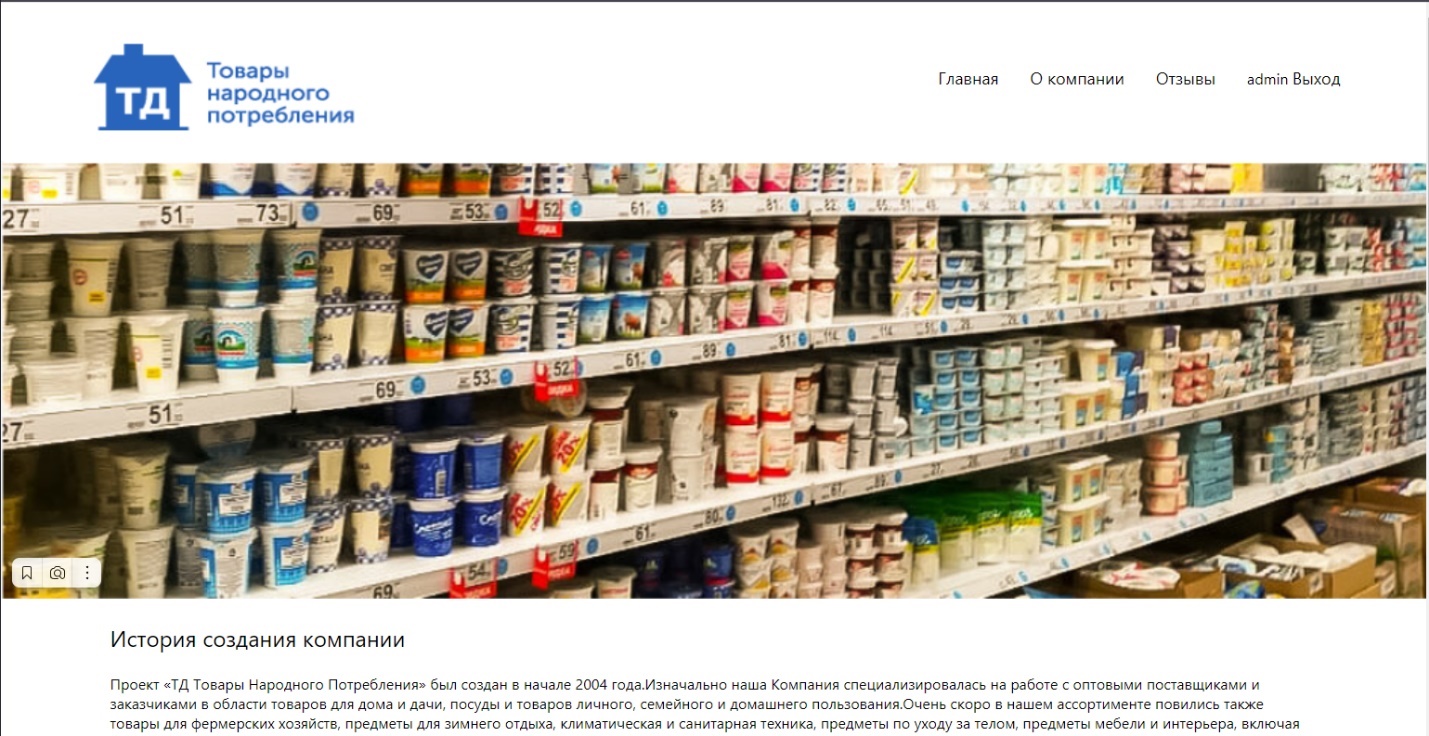
На странице «О компании» пользователю будет предоставлена информация о истории компании.

Рис. 5.1 «О компании»

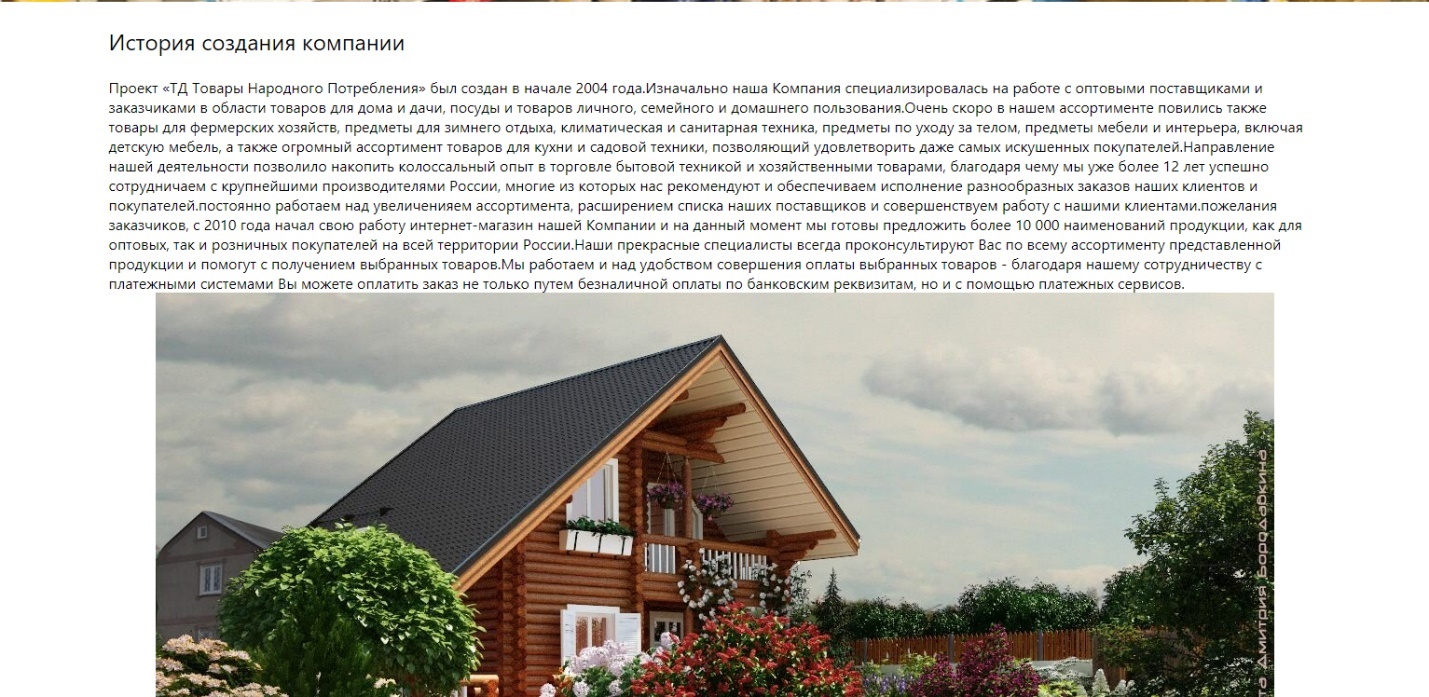


Рис. 5.2 «О компании»

Рис. 5.3 «О компании»

Для перехода на «Отзывы» нужно перейти по ссылке [Главная страница сайта](http://127.0.0.1:8000/), но для пользователя, который находится на сайте достаточно просто нажать на кнопку на боковой панели.

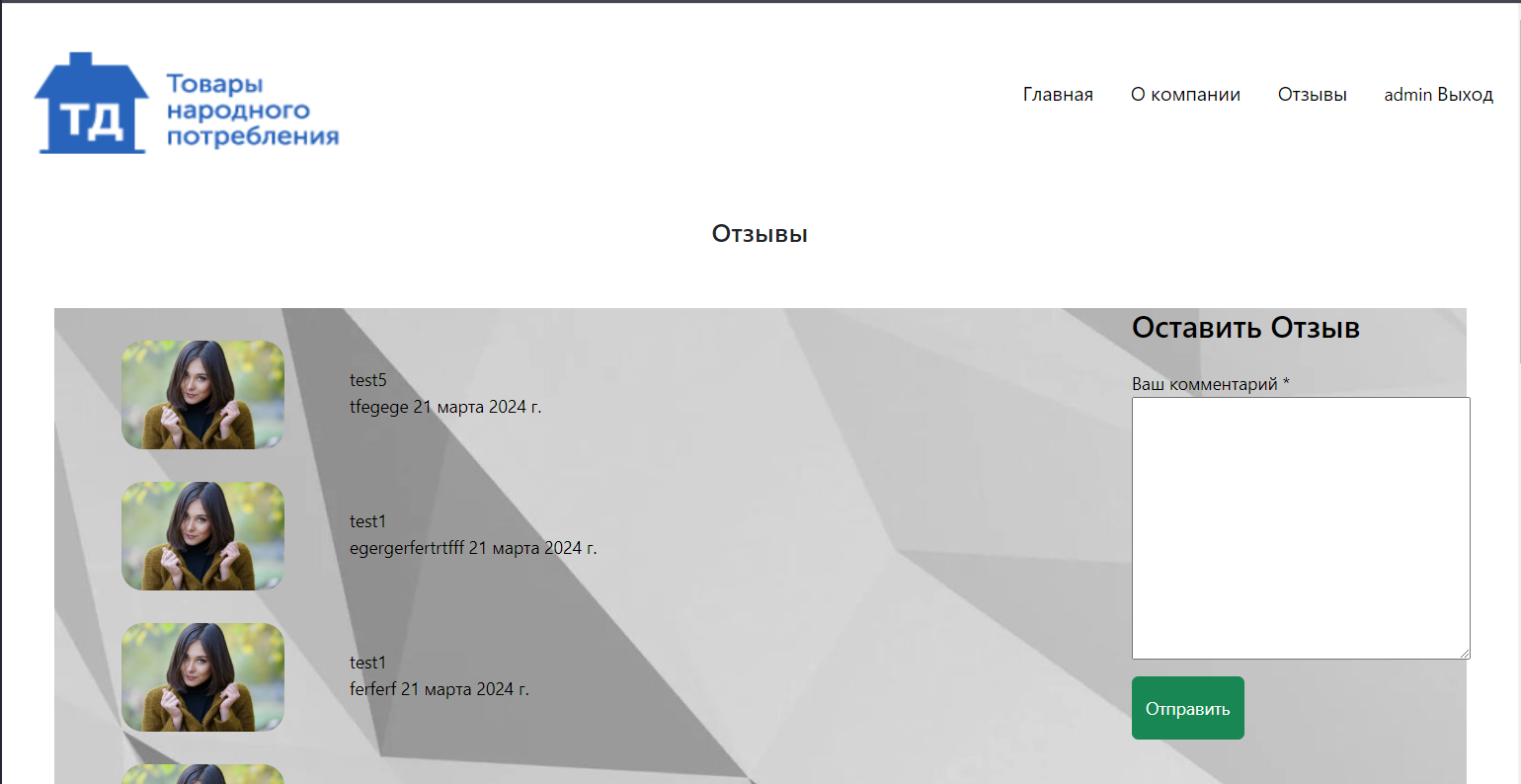
На странице «Отзывы» пользователю будет предоставлена информация о отзывах, где он также сможет его оставить.

Рис. 6.1 «Отзывы»

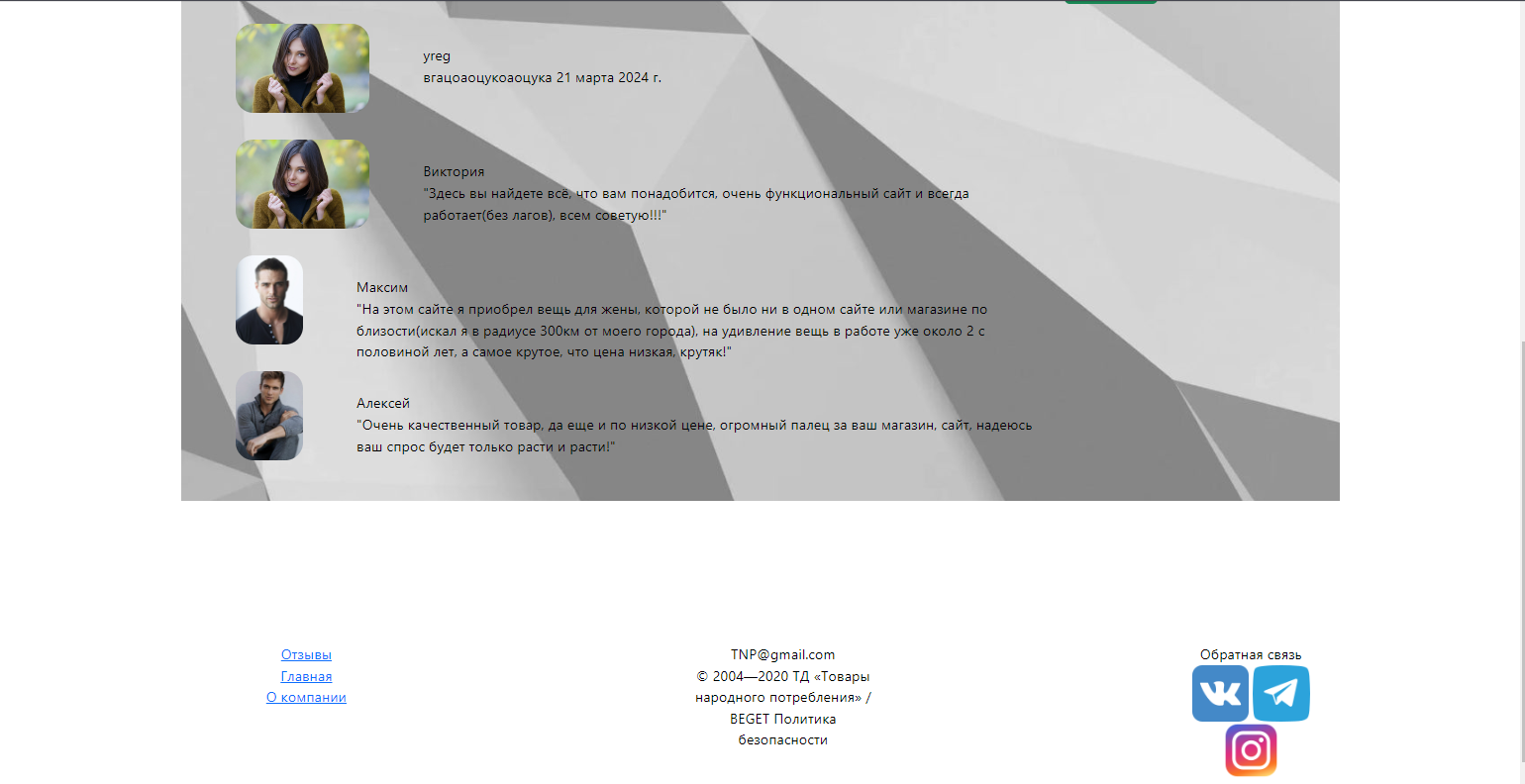


Рис. 6.2 «Отзывы»

Для перехода на «Вход и Регистрация» нужно перейти по ссылке [Главная страница сайта](http://127.0.0.1:8000/), но для пользователя, который находится на сайте достаточно просто нажать на кнопку на боковой панели.

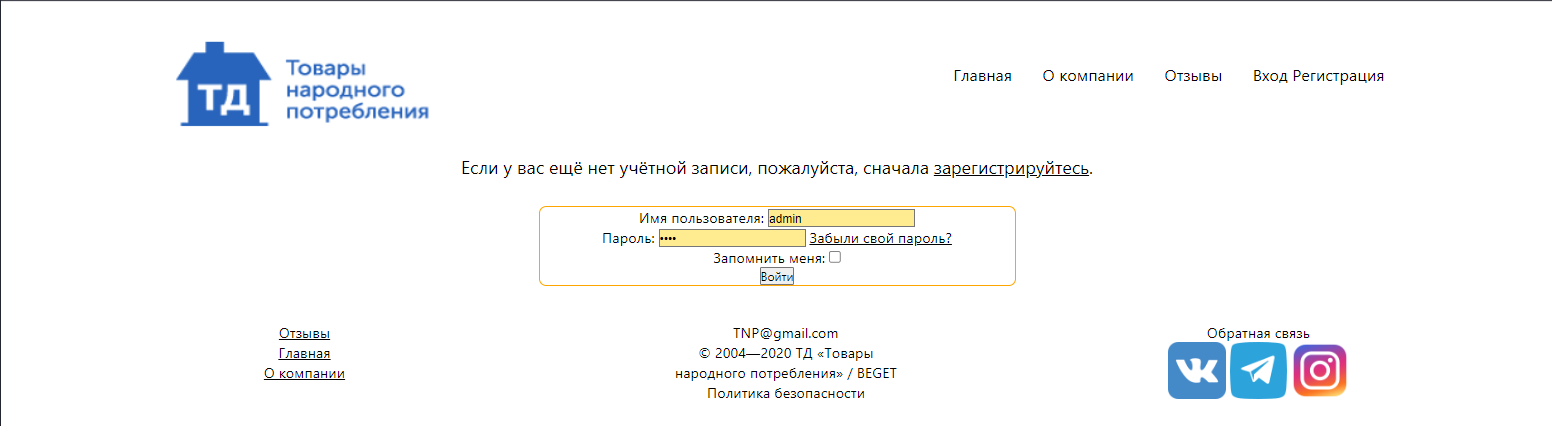
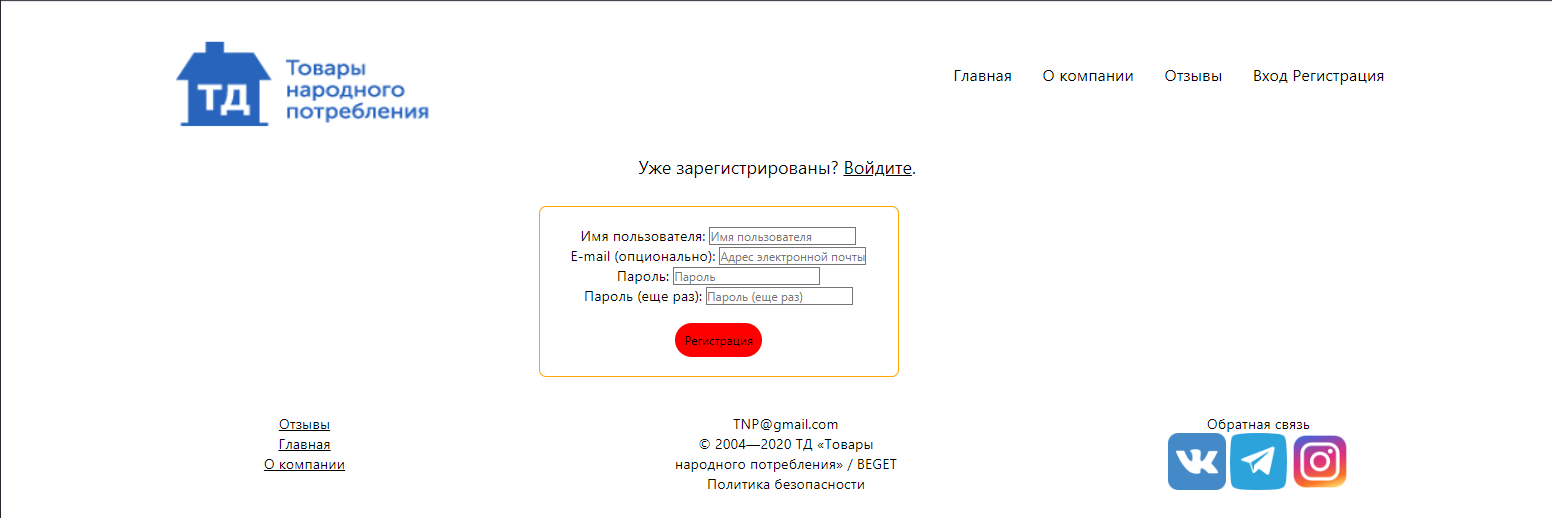
На странице «Вход и Регистрация» пользователю дают возможность создать аккаунт или войти в него.

Рис. 7.1 «Вход»

Рис. 7.2 «Регистрация»

# Тестирование программного продукта

## **4.1 Тестовые сценарии**

Таблица №5 «Тестовые сценарии»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Идентификатор** | **Описание** | **Шаги** | **Входные  данные** | **Ожидаемые  результаты** | **Фактические**  **результаты** | **Статус** |
| TUO1 | Проверка входа пользователя  с существующим логином и паролем | Откройте сайт  <http://localhost:8000>  ->  Нажмите «Войти»  ->  Введите логин  ->  Введите пароль  ->  Нажмите «Войти» | Логин= admin  Пароль =  1234 | Пользователь должен войти и попасть на главную страницу | Как ожидали | Пройден успешно |
| TUO2 | Проверка входа пользователя  с несуществующим логином и паролем | Откройте сайт  <http://localhost:8000>  ->  Нажмите «Войти»  ->  Введите логин  ->  Введите пароль  ->  Нажмите «Войти» | Логин= admin1  Пароль =  12345 | Пользователя должно перекинуть на эту же страницу и выдать ошибку и предложить зарегистрировать аккаунт | Как ожидали | Пройден успешно |
| TUO3 | Проверка регистрации пользователя, с почтой и подтверждением пароля | Откройте сайт  <http://localhost:8000>  ->  Нажмите «Зарегистрироваться»  ->  Введите логин  ->  Введите пароль  ->  Повторите пароль  ->  Нажмите «Зарегистрироваться» | Логин= admin  Пароль =  1234  П.Пароля =  1234  Email= TNP12\_@gmail.com | Пользователя  Должно успешно зарегистрировать и перекинуть на главную страницу | Как ожидали | Пройден успешно |

## **4.2 Юнит-тесты в Django**

Таблица №6 «Юнит-тесты»

|  |  |  |
| --- | --- | --- |
| **Тесты** | **Описание** | **Работа тестов** |
| from django.test import TestCase, Client  from django.urls import reverse  class TestCatalogViews(TestCase):      def setUp(self):          self.client = Client()          self.catalog\_url = reverse('catalog')      def test\_catalog\_GET(self):          response = self.client.get(self.catalog\_url)          self.assertEquals(response.status\_code, 200)          self.assertTemplateUsed(response, 'traur2/catalog.html') | Тест(каталога)  Данный тест описывает проверку GET-запроса к представлению "catalog". Тест использует класс TestCase из модуля django.test и клиентский класс Client для создания HTTP-запросов.  В методе setUp() устанавливаются начальные условия для теста. Создается экземпляр клиента Client() и определяется URL для представления "catalog" с помощью функции reverse(), которая возвращает URL по имени представления.  Метод test\_catalog\_GET() проверяет GET-запрос к представлению "catalog". Выполняется запрос с помощью self.client.get(), где передается URL представления. Затем проверяется код состояния ответа (response.status\_code) с ожидаемым значением 200 (успешный код ответа).  Далее с помощью метода assertTemplateUsed() проверяется, что при запросе был использован ожидаемый шаблон 'traur2/catalog.html'. Это гарантирует, что при GET-запросе к представлению "catalog" был использован правильный шаблон для отображения ответа.  Если код состояния и использованный шаблон соответствуют ожидаемым значениям, тест считается успешным. В противном случае, если хотя бы одно из ожиданий не выполнилось, тестовый прогон будет считаться неудачным.  Этот тест помогает убедиться, что представление "catalog" возвращает правильный код ответа и использует правильный шаблон при успешном GET-запросе. |  |
| from django.test import TestCase, Client  from django.urls import reverse  from django.contrib.auth.models import User  class TestKontactViews(TestCase):      def setUp(self):          self.client = Client()          self.kontact\_url = reverse('kontact')          self.user = User.objects.create\_user(username='testuser', password='12345')      def test\_kontact\_GET(self):          self.client.login(username='testuser', password='12345')          response = self.client.get(self.kontact\_url)          self.assertEquals(response.status\_code, 200)          self.assertTemplateUsed(response, 'traur2/kontact.html') | Тест(Kontact)  Данный тест проверяет GET-запрос к представлению "kontact" с аутентификацией пользователя. Он также использует класс TestCase из модуля django.test и класс Client для создания HTTP-запросов.  В методе setUp() устанавливаются начальные условия для теста. Создается экземпляр клиента Client() и определяется URL для представления "kontact" с помощью функции reverse(). Кроме того, с использованием User.objects.create\_user() создается тестовый пользователь с именем пользователя "testuser" и паролем "12345".  Метод test\_kontact\_GET() проверяет GET-запрос к представлению "kontact" с аутентификацией пользователя. Сначала выполняется вход пользователя с помощью self.client.login(), где передаются имя пользователя и пароль. Затем выполняется GET-запрос к URL представления с помощью self.client.get().  Далее проверяется код состояния ответа (response.status\_code) с ожидаемым значением 200 (успешный код ответа). Затем с помощью метода assertTemplateUsed() проверяется, что при запросе был использован ожидаемый шаблон 'traur2/kontact.html'.  Если код состояния и использованный шаблон соответствуют ожиданиям, тест считается успешным. Тест также проверяет, что аутентификация пользователя работает, поскольку представление "kontact" требует, чтобы пользователь был аутентифицирован.  Этот тест помогает убедиться, что представление "kontact" возвращает правильный код ответа и использует правильный шаблон при успешном GET-запросе с аутентификацией пользователя. |  |
| from django.test import TestCase, Client  from django.urls import reverse  class TestMainViews(TestCase):      def setUp(self):          self.client = Client()          self.main\_url = reverse('main')      def test\_main\_GET(self):          response = self.client.get(self.main\_url)          self.assertEquals(response.status\_code, 200)          self.assertTemplateUsed(response, 'traur2/main.html') | Тест(main)  Данный тест проверяет GET-запрос к представлению "main". Он использует класс TestCase из модуля django.test и класс Client для создания HTTP-запросов.  В методе setUp() устанавливаются начальные условия для теста. Создается экземпляр клиента Client() и определяется URL для представления "main" с помощью функции reverse().  Метод test\_main\_GET() проверяет GET-запрос к представлению "main". Запрос выполняется с помощью self.client.get(), где передается URL представления. Затем проверяется код состояния ответа (response.status\_code) с ожидаемым значением 200 (успешный код ответа).  Далее с помощью метода assertTemplateUsed() проверяется, что при запросе был использован ожидаемый шаблон 'traur2/main.html'. Это гарантирует, что при GET-запросе к представлению "main" был использован правильный шаблон для отображения ответа.  Если код состояния и использованный шаблон соответствуют ожидаемым значениям, тест считается успешным. В противном случае, если хотя бы одно из ожиданий не выполнено, тест будет считаться неудачным.  Этот тест помогает убедиться, что представление "main" возвращает правильный код состояния и использует правильный шаблон при успешном GET-запросе. |  |
| from django.test import TestCase  from django.contrib.auth.models import User  from django.utils import timezone  from . import Task  class TaskModelTest(TestCase):      @classmethod      def setUpTestData(cls):          test\_user = User.objects.create\_user(username='testuser', password='12345')          Task.objects.create(name=test\_user, date=timezone.now(), text='Test task')      def test\_name\_label(self):          task = Task.objects.get(id=1)          field\_label = task.\_meta.get\_field('name').verbose\_name          self.assertEquals(field\_label, 'name')      def test\_text\_max\_length(self):          task = Task.objects.get(id=1)          max\_length = task.\_meta.get\_field('text').max\_length          self.assertEquals(max\_length, 200) | Тест(Task)  Данный тест является модульным тестом модели Task. Он использует класс TestCase и проверяет два аспекта модели: метку поля имени и максимальную длину поля текст.  В методе setUpTestData() создается тестовый пользователь с именем пользователя "testuser" и паролем "12345" с помощью User.objects.create\_user(). Затем создается экземпляр модели Task с указанными атрибутами: имя пользователя, текущая дата и текст "Test task".  Метод test\_name\_label() проверяет метку поля имени модели Task. Получается экземпляр задачи с помощью Task.objects.get(id=1), а затем получается метка поля имени с использованием verbose\_name для поля 'name' из метаданных модели (task.\_meta.get\_field('name')). Далее с помощью self.assertEquals() проверяется совпадение метки поля имени с ожидаемым значением "name".  Метод test\_text\_max\_length() проверяет максимальную длину поля текст модели Task. Аналогично предыдущему тесту, получается экземпляр задачи и затем получается максимальное значение длины поля текст с использованием max\_length для поля 'text' из метаданных модели (task.\_meta.get\_field('text')). Затем с помощью self.assertEquals() проверяется совпадение максимальной длины поля текст с ожидаемым значением 200.  Если проверки в обоих тестах совпадают с ожидаемыми значениями, тесты считаются успешными. При любом несовпадении проверки тест будет считаться неудачным.  Этот тест помогает убедиться в правильном определении меток полей и максимальной длины поля текст модели Task. |  |
| from django.test import TestCase  from . import Product  class ProductModelTest(TestCase):      @classmethod      def setUpTestData(cls):          Product.objects.create(name='Test Product', price=100, cat='category1')      def test\_name\_label(self):          product = Product.objects.get(id=1)          field\_label = product.\_meta.get\_field('name').verbose\_name          self.assertEquals(field\_label, 'Наименование продукта')      def test\_price\_positive(self):          product = Product.objects.get(id=1)          self.assertTrue(product.price >= 0) | Тест(Product)  Данный тест является модульным тестом модели Product. Он использует класс TestCase и проверяет два аспекта модели: метку поля имени и значение цены, которое должно быть неотрицательным.  В методе setUpTestData() создается экземпляр модели Product с указанными атрибутами: наименование "Test Product", цена 100 и категория "category1".  Метод test\_name\_label() проверяет метку поля имени модели Product. Получается экземпляр продукта с помощью Product.objects.get(id=1), а затем получается метка поля имени с использованием verbose\_name для поля 'name' из метаданных модели (product.\_meta.get\_field('name')). Далее с помощью self.assertEquals() проверяется совпадение метки поля имени с ожидаемым значением "Наименование продукта".  Метод test\_price\_positive() проверяет значение цены, чтобы оно было неотрицательным. Получается экземпляр продукта и затем с помощью self.assertTrue() проверяется, что цена продукта (product.price) больше или равна 0, что гарантирует, что цена является неотрицательным числом.  Если проверки в обоих тестах совпадают с ожидаемыми значениями, тесты считаются успешными. При любом несовпадении проверки тест будет считаться неудачным.  Этот тест помогает убедиться в правильном определении метки поля имени и в том, что значение цены является неотрицательным для модели Product. |  |

# Размещение проекта на хостинге

Таблица №7 «Размещение на хостинге»

|  |  |
| --- | --- |
| **Скриншот** | **Описание** |
|  | Регистрируем аккаунт на timeweb |
|  | Проверяем наш бесплатный домен |
|  | Проверяем наш сайт |
|  | Перейдите на главную страницу и включите доступ по SSH: |
|  | Скачиваем Putty |
|  | Вставляем команды: wget https://bootstrap.pypa.io/virtualenv/3.6/virtualenv.pyz - установка виртуального окружения.  python3 virtualenv.pyz venv - включение виртуального окружения.  source venv/bin/activate - активируем виртуальное окружение  pip install django==3.2  cd public\_html |
|  | Найдите там созданный .htaccess |
|  | Прописываем в settings.py (ALLOWED\_HOSTS = [“\*”] |
|  | Теперь откройте файл wsgi.py  Измените содержание на:  # -\*- coding: utf-8 -\*-  import os  import sys  import platform  # путь к проекту  sys.path.insert(0, '/home/c/cl36524/public\_html')  # путь к фреймворку  sys.path.insert(0, '/home/c/cl36524/public\_html/productprj')  # путь к виртуальному окружению  python\_version = ".".join(platform.python\_version\_tuple()[:2])  sys.path.insert(0, '/home/c/cl36524/venv//lib/python{0}/site-packages'.format(python\_version))  os.environ["DJANGO\_SETTINGS\_MODULE"] = "productprj.settings"  from django.core.wsgi import get\_wsgi\_application  application = get\_wsgi\_application() |
|  | Переходим на ваш домен. Должна появиться стартовая страница Django: |
| INSTALLED\_APPS = [  'django.contrib.admin',  'django.contrib.auth',  'django.contrib.contenttypes',  'django.contrib.sessions',  'django.contrib.messages',  'django.contrib.staticfiles',  'traur',  ] | Ещё дописываю необходимые приложения и строки в settings.py, например у меня это приложение traur: |
|  | Если у вас есть в проекте папка статик, то её нужно пересобрать в одну единую, для этого в вашем проекте выполните команду: python manage.py collectstatic  Создастся папка static. Её нужно перекинуть на хостинг по адресу, который мы указали в settings.py :  STATIC\_ROOT = '/home/c/cl36524/public\_html/static'  И наконец выполните миграции для проекта: |
|  | Готово, сайт добавлен в «Мои сайты» |

# Заключение

В данной курсовой работе создан сайт «Товары народного потребления». Проект создан в Visual Studio code. 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 технологии Bootstrap, и фреймворк Django.

На данный момент сайт «ТНП» требует усовершенствования. Поэтому я вижу следующие решения, которые помогут улучшить будущий сайт:

1. Добавление новейшего интерфейса

2. Более детальна проработка форм обратных связей

3. Углубление в детальность кнопок и их путей

При написании документации для программного продукта, в которой описывается способ использования. Также сформированы требования к адаптивности, к технологиям и инструментам для вёрстки, к составу и параметрам технических средств. Документирование позволило рассмотреть сайт с точки зрения программиста, пользователя и тестировщика.

Полностью код свёрстанного сайта можно посмотреть в репозитории на GitHab: https://github.com/DenisGalanin/Kyrsovoi

# Список литературы

1. Чернышев, С. А.  Основы программирования на Python : учебное пособие для среднего профессионального образования / С. А. Чернышев. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 349 с. — (Профессиональное образование). — ISBN 978-5-534-17056-6.
2. Федоров, Д. Ю.  Программирование на языке высокого уровня Python : учебное пособие для среднего профессионального образования / Д. Ю. Федоров. — 5-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 227 с. — (Профессиональное образование). — ISBN 978-5-534-17319-2.
3. Гниденко, И. Г.  Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 248 с. — (Профессиональное образование). — ISBN 978-5-534-18131-9.
4. Тузовский, А. Ф.  Проектирование и разработка web-приложений : учебное пособие для среднего профессионального образования / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2024. — 219 с. — (Профессиональное образование). — ISBN 978-5-534-16767-2.
5. Полуэктова, Н. Р.  Разработка веб-приложений : учебное пособие для среднего профессионального образования / Н. Р. Полуэктова. — 2-е изд. — Москва : Издательство Юрайт, 2024. — 204 с. — (Профессиональное образование). — ISBN 978-5-534-18644-4.
6. Каменнова, М. С.  Моделирование бизнес-процессов : учебник и практикум для среднего профессионального образования / М. С. Каменнова, В. В. Крохин, И. В. Машков. — Москва : Издательство Юрайт, 2024. — 533 с. — (Профессиональное образование). — ISBN 978-5-534-16845-7.
7. Лаврищева, Е. М.  Программная инженерия. Парадигмы, технологии и CASE-средства : учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. — Москва : Издательство Юрайт, 2024. — 280 с. — (Высшее образование). — ISBN 978-5-534-01056-5.

Интернет ресурсы:

1. HTML с нуля Лучший HTML учебник для начинающих и чайников. <https://html5css.ru/html/default.php?ysclid=lucadeayj2209252290>
2. Руководство по HTML5 и CSS3. <https://metanit.com/web/html5/?ysclid=lucadgeqih655506976>
3. Django тестирование. <https://www.yourtodo.ru/posts/django-testirovanie/>
4. “Библия QA” - обновляемая база знаний объемом 560+ страниц: https://vladislaveremeev.gitbook.io/qa\_bible