**Министерство образования Московской области**

**ГОУ ВО МО «Государственный гуманитарно-технологический университет» (ГГТУ) Ликино-Дулевский политехнический колледж- филиал ГГТУ**

Наименование колледжа/техникума

**О Т Ч Ё Т**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

ПМ.05 Проектирование и разработка информационных систем

(код и наименование ПМ)

Обучающегося\_\_Аксеновой Софьи Алексеевны \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О.

Курс \_\_\_\_3\_\_\_\_\_ группа \_\_\_ИСП.21.2А\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность 09.02.07 Информационные системы и программирование \_\_\_

Место практики Ликино-Дулевский политехнический колледж – филиал ГГТУ

Период практики 24.11.2023 г. - 07.12.2023 г., 06.04.2024 г. - 19.04.2024 г.

Руководители практики

от колледжа \_\_\_\_\_\_\_\_Пронина Алла Юрьевна \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гжегожевский Сергей Владимирович\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Селиверстова Ольга Михайловна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Лысенко Вячеслав Витальевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Ликино-Дулево

2024 г.

**Содержание**

[1.Разработка web-приложения 4](#_Toc164342710)

[1.1. Общие сведения. 4](#_Toc164342711)

[1.1.1. Что такое Django 4](#_Toc164342712)

[1.1.2. Установка и настройка. 4](#_Toc164342713)

[1.1.3. Основные команды Django 4](#_Toc164342714)

[1.2. Руководство программиста 4](#_Toc164342715)

[1.2.1. Модели 4](#_Toc164342716)

[1.2.2. Представления 5](#_Toc164342717)

[1.2.3. Шаблоны 6](#_Toc164342718)

[1.2.4. Работа с формами 8](#_Toc164342719)

[1.3. Руководство пользователя 8](#_Toc164342720)

[2.Роутинг в CMS Bitrix 10](#_Toc164342721)

[2.1. Установить CMS Bitrix 10](#_Toc164342722)

[2.2. Настройка роутинга 11](#_Toc164342723)

[2.3. Практическая часть 11](#_Toc164342724)

[2.3.1. Получить все активные товары определенного раздела; 11](#_Toc164342725)

[2.3.2. Получить все товары по фильтру; 12](#_Toc164342726)

[2.3.3. Получить все активные товары с ценой; 13](#_Toc164342727)

[3. Быстрый старт разработчика CMS Bitrix 14](#_Toc164342728)

[3.1. Установка системы 14](#_Toc164342729)

[3.2 Интеграция верстки 20](#_Toc164342730)

[3.3 Редактируемые области 22](#_Toc164342731)

[3.4 Базовая настройка системы 22](#_Toc164342732)

[3.5 Компоненты и структура сайта 23](#_Toc164342733)

[4. Разработка API для бэкэнда 26](#_Toc164342734)

[4.1. Разработка сервера 26](#_Toc164342735)

[4.2. Выполнение запросов к серверу 34](#_Toc164342736)

[5. Проектирование предметной области 42](#_Toc164342737)

[6. Разработка технической документации 44](#_Toc164342738)

[6.1. Техническое задание 44](#_Toc164342739)

[6.2. Руководство пользователя 47](#_Toc164342740)

[6.3. Руководство программиста 54](#_Toc164342741)

[6.4. Методика тестирования и испытания программы 60](#_Toc164342742)

[6.5. Тест-план программы 68](#_Toc164342743)

[7. Обеспечения качества программного обеспечения 76](#_Toc164342744)

[8. Расчет надежности работы сайта 95](#_Toc164342745)

[Заключение 102](#_Toc164342746)

[Список литературы 104](#_Toc164342747)

# 1.Разработка web-приложения

## 1.1. Общие сведения.

### 1.1.1. Что такое Django

Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation.

### 1.1.2. Установка и настройка.

Установка Django

1 вариант через настройки File-> Setting-> Python Interpreter-> “+” Django

2 вариант через терминал pip install django

### 1.1.3. Основные команды Django

Таблица №1 «Основные команды Django»

|  |  |
| --- | --- |
| **Команда** | **Пояснение** |
| django-admin startproject myproject | создать новый проект |
| python manage.py runserver | запустить тестовый сервер |
| django-admin startapp myapp | создать приложение в текущем проекте |
| python manage.py createsuperuser | создать пользователя-администратора |
| Python manage.py [migrate](https://djangodoc.ru/3.1/ref/django-admin/#django-admin-migrate) | отвечает за выполнение и отмену миграции. |
| python manage.py makemigrations | отвечает за создание новых миграций на основе изменений, внесенных в модели. |

## 1.2. Руководство программиста

### 1.2.1. Модели

Была создана всего 1 модель “Product” в которой находятся номер продукции, наименование, цену, количество, производителя и фото.

class Product(models.Model):  
 number = models.IntegerField('Номер', max\_length=50)  
 name = models.CharField('Наименование', max\_length=50)  
 price = models.IntegerField('Цена', max\_length=50)  
 quantity = models.CharField('Количество', max\_length=50)  
 manufacturer = models.CharField('Производитель', max\_length=50)  
 photo = models.ImageField(upload\_to="photos", default=None, blank=True, null=True, verbose\_name="Фото")  
 objects = models.Manager()  
  
 @property  
 def photo\_url(self):  
 if self.photo and hasattr(self.photo, 'url'):  
 return self.photo.url  
  
 def \_\_int\_\_(self):  
 return self.number  
  
 class Meta:  
 verbose\_name = 'Товар'  
 verbose\_name\_plural = 'Товары'

### 1.2.2. Представления

Это страница «Главная»

def index(request):  
 tasks\_all = Product.objects.order\_by('id')  
 tasks = PostFilter(request.GET, queryset=tasks\_all)  
 return render(request, 'main/index.html', {'title': 'Главная страница сайта',  
 'tasks': tasks})  
  
Это страница «Доставка»  
def delivery(request):  
 return render(request, 'main/delivery.html')  
  
Это страница «Уход за мебелью»  
def care(request):  
 return render(request, 'main/care.html')  
  
Это форма добавления  
def add(request):  
 error = ""  
 if request.method == 'POST':  
 form = ProductForm(request.POST, request.FILES)  
 if form.is\_valid():  
 form.save()  
 return redirect('home')  
 else:  
 error = 'Форма была неверной'  
 form = ProductForm()  
 context = {'form': form, 'error': error}  
 return render(request, 'main/add.html', context)  
  
Это кнопка удаления товара  
class ProductDeleteView(DeleteView):  
 model = Product  
 success\_url = '/'  
 template\_name = 'main/task-delete.html'  
  
Это кнопка редактирования товара  
class ProductUpdateView(UpdateView):  
 model = Product  
 template\_name = 'main/create.html'  
 form\_class = ProductForm  
 success\_url = '/'

### 1.2.3. Шаблоны

Есть шаблон «index.html» на котором расположен блок фильтрации товаров и сами товары с возможностью его удаления и редактирования.

{% endblock %}  
  
{% block content %}  
<div class="zalogo"><h1>Список товаров</h1></div>  
<div class="container">  
 <form method="get" class="form-filter">  
 <div>  
 {{ tasks.form.as\_p }}  
 <button type="submit" class="btn btn-primary">Фильтровать</button>  
 </div>  
 </form>  
</div>  
{% if tasks %}  
<table cellspacing="0">  
 <tr>  
 <th>№</th>  
 <th>Наименование</th>  
 <th>Фото</th>  
 <th>Цена</th>  
 <th>Количество</th>  
 <th>Производитель</th>  
 <th>Действия</th>  
 </tr>  
  
 {% for elem in tasks.qs %}  
 <tr>  
 <td>{{ elem.number }}</td>  
 <td>{{ elem.name }}</td>  
 <td><img src="{{ elem.photo\_url }}" alt="Фото" width="150"></td>  
 <td>{{ elem.price }}</td>  
 <td>{{ elem.quantity }}</td>  
 <td>{{ elem.manufacturer }}</td>  
 <td><a href="{% url 'task-delete' elem.id %}" class="btn btn-warning">Удалить</a>  
 <a href="{% url 'task-update' elem.id %}" class="btn btn-info">Редактировать</a></td>  
 </tr>  
 {% endfor %}  
</table>  
{% else %}  
<p>У нас нет товаров</p>  
{% endif %}  
{% endblock %}

Есть шаблон «add.html» на котором расположена форма добавления товара с полями: номер товара, цена, производитель, наименование, количество и загрузка фотографии товара.

{% extends 'main/base.html' %}  
{% load static %}  
  
{% block title %}  
Добавить товар  
{% endblock %}  
  
{% block content %}  
<div class="zalogo"><h1>Добавить товар</h1></div>  
<form method="post" enctype="multipart/form-data">  
 <h4>Новый товар</h4>  
 <div class="form">  
 {% csrf\_token %}  
 {{ form.number }}<br>  
 {{ form.name }}<br>  
 {{ form.price }}<br>  
 {{ form.quantity }}<br>  
 {{ form.manufacturer }}<br>  
 {{ form.photo }} <br></div>  
 <button type="submit" class="btn btn-success" style="background-color: #FFF06A;">Сохранить</button>  
 <span>{{ error }}</span>  
</form>  
{% endblock %}

### 1.2.4. Работа с формами

Была создана одна форма, в которой находятся: номер товара, наименование, цена, количество, производителя и загрузка фотографии на карточку товара.

from .models import Product  
from django.forms import ModelForm, TextInput, FileField, ClearableFileInput  
from django.forms import FileInput  
  
  
class ProductForm(ModelForm):  
 class Meta:  
 model = Product  
 fields = ["number", "name", "price", "quantity", "manufacturer", "photo"]  
 widgets = {  
 "number": TextInput(attrs={  
 'class': 'form-control',  
 'placeholder': 'Введите номер'  
 }),  
 "name": TextInput(attrs={  
 'class': 'form-control',  
 'placeholder': 'Введите наименование'  
 }),  
 "price": TextInput(attrs={  
 'class': 'form-control',  
 'placeholder': 'Введите цену'  
 }),  
  
 "quantity": TextInput(attrs={  
 'class': 'form-control',  
 'placeholder': 'Введите количество'  
 }),  
 "manufacturer": TextInput(attrs={  
 'class': 'form-control',  
 'placeholder': 'Введите производителя'  
 }),  
 "photo": FileInput(attrs={  
 'class': 'p\_files',  
 'placeholder': 'Файлы'  
 }),  
 }

## 1.3. Руководство пользователя

На странице «index» у нас есть товары и форма фильтрации, товары можно редактировать и удалять.

На странице «add» можно добавить товар.

На странице «delivery» находится информация о доставке и ее условия.

На странице «care» находятся рекомендации по уходу за мебелью.

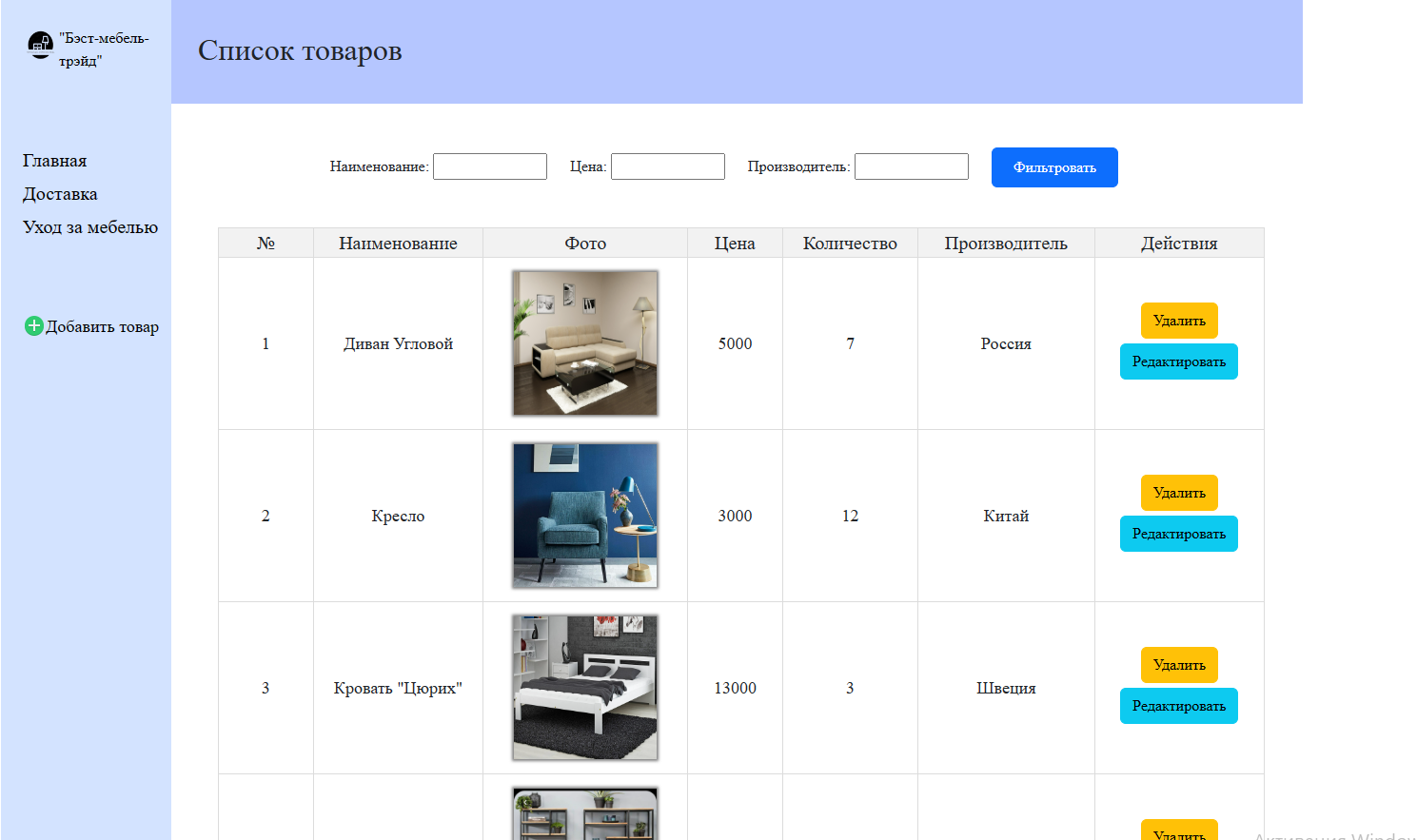


Рисунок 1 «Главная страница»

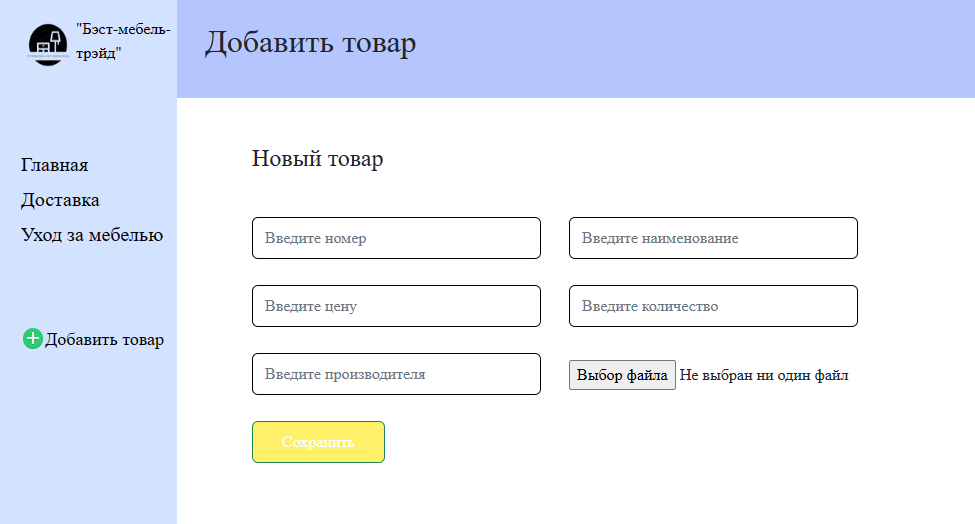


Рисунок 2 «Добавления товара»

# 2.Роутинг в CMS Bitrix

## 2.1. Установить CMS Bitrix

Таблица №2 «Установка системы Bitrix»

|  |  |
| --- | --- |
| Выбираем нужную версию Bitrix |  |
| Ждём загрузки дистрибутива |  |
| Установка Бизнес-процессов |  |
| Установка решения |  |
| Установка завершена |  |
| Настройка хоста |  |

## 2.2. Настройка роутинга

Находится в папке routes/web.php

use Bitrix\Main\HttpRequest;

use Bitrix\Main\Routing\Route;

use Bitrix\Main\Routing\RoutingConfigurator;

## 2.3. Практическая часть

### 2.3.1. Получить все активные товары определенного раздела;

Находится в папке routes/web.php

$routes->any('/ARI/v1/getActiveProductFilter', function () {

\Bitrix\Main\Loader::includeModule('iblock');

if(!$\_REQUEST['section\_id']){

return json\_encode([

'status' => 'error',

'error' => 'Нет ключа section\_id'

]);

}

$elements = \Bitrix\Iblock\Elements\ElementCatalogTable::getList([

'select' => ['ID', 'NAME', 'SORT'],

'filter' => [

'=ACTIVE' => 'Y',

'=IBLOCK\_SECTION\_ID' => $\_REQUEST['section\_id'],

],

])->fetch();

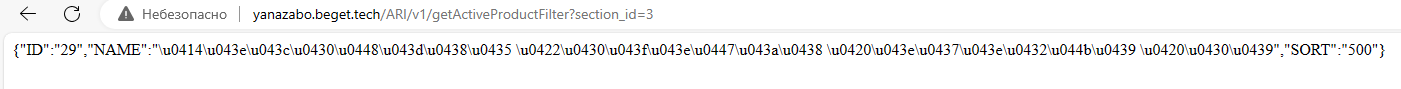


Рисунок 3 «Вывод активных товаров определённого раздела»

### 2.3.2. Получить все товары по фильтру;

Находится в папке routes/web.php

$routes->any('/ARI/v1/getActiveProductSection', function () {

\Bitrix\Main\Loader::includeModule('iblock');

if(!$\_REQUEST['section\_id']){

return json\_encode([

'status' => 'error',

'error' => 'Нет ключа section\_id'

]);

}

$elements = \Bitrix\Iblock\Elements\ElementCatalogTable::getList([

'select' => ['ID', 'NAME'],

'filter' => [

'=ACTIVE' => 'Y',

'=IBLOCK\_SECTION\_ID' => $\_REQUEST['section\_id'],

],

])->fetchAll();

if(!$elements){

return json\_encode([

'status' => 'error',

'error' => 'В данном разделе нет товаров!'

]);

}

return json\_encode($elements);

});

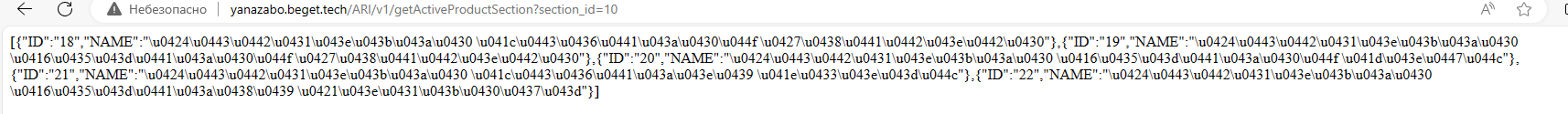


Рисунок 4 «Вывод всех товаров по фильтру»

### 2.3.3. Получить все активные товары с ценой;

Находится в папке routes/web.php

$routes->any('/ARI/v1/getActiveProductPrice', function () {

\Bitrix\Main\Loader::includeModule('iblock');

\Bitrix\Main\Loader::includeModule('sale');

$element = \Bitrix\Iblock\Elements\ElementCatalogTable::getList([

'select' => ['ID', 'NAME'],

'filter' => [

'=ACTIVE' => 'Y',

'=ID' => 4

],

])->fetch();

$price = \Bitrix\Catalog\PriceTable::getList([

'select' => ['PRICE', 'CURRENCY', 'ID', 'PRODUCT\_ID'],

'filter' => ['=PRODUCT\_ID' => $element['ID']]

])->fetch();

array\_push($element, $price);

return json\_encode($element);

d($element);

});

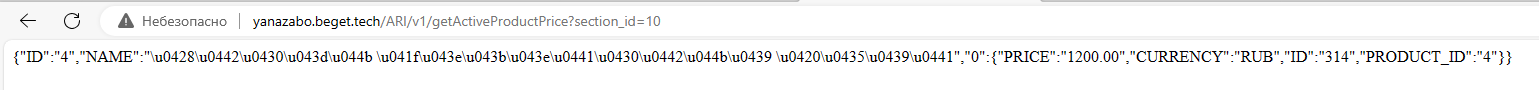


Рисунок 5 «Вывод всех активных товаров с ценой»

# 3. Быстрый старт разработчика CMS Bitrix

## 3.1. Установка системы

Таблица №3 «Установка системы Bitrix»

|  |  |
| --- | --- |
| Скачиваем bitrixsetup.php на хостинг |  |
| Загружаем bitrixsetup.php в файловый менеджер |  |
| Выбираем версию «Старт» и нажимаем «Загрузить» |  |
| Ждём загрузку дистрибутива |  |
| Устанавливаем сам продукт и нажимаем «Далее» |  |
| Принимаем лицензионное соглашение |  |
| Регистрируем продукт и вводим данные |  |
| Проверяем чтобы все пункты были включены и горят зелёным, если какие-то отключены- включаем |  |
| Создаём базу данных |  |
| Ждём установки дистрибутива |  |
| Создаём администратора сайта для дальнейшего редактирования |  |
| Выбираем решение для установки «Загрузить из Marketplace» |  |
| Выбираем решение «Чистая установка «1C-Битрикс» |  |
| Загружаем решение |  |
| Настраиваем продукт |  |
| Настраиваем решение |  |
| Выбираем нужные нам модули |  |
| Ждём установки модулей |  |
| Установка завершена |  |

## 3.2 Интеграция верстки

Пишем код для header и footer и интегрируем это на сайт

Header

<?

if(!defined('B\_PROLOG\_INCLUDED') || B\_PROLOG\_INCLUDED !== true)

die();

$bIsMainPage = $APPLICATION->GetCurPage(false) == SITE\_DIR;

?>

<!DOCTYPE html>

<!--[if lt IE 8]> <html class="no-js lt-ie9 lt-ie8"><![endif]-->

<!--[if IE 8]> <html class="no-js lt-ie9"><![endif]-->

<!--[if gt IE 8]> <html class="no-js"><![endif]-->

<head>

<!--[if IE]>

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"><![endif]-->

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

<title><?$APPLICATION->ShowTitle();?></title>

<?$APPLICATION->ShowHead();?>

<?$APPLICATION->SetAdditionalCSS(SITE\_TEMPLATE\_PATH.'/css/common-styles.css');?>

<link rel="icon" href="<?=SITE\_TEMPLATE\_PATH?>ico/favicon\_bx.png">

<!--[if lt IE 9]>

<script src="<?=CUtil::GetAdditionalFileURL(SITE\_TEMPLATE\_PATH.'/js/vendor/modernizr-html5shiv-respond.min.js')?>"></script>

<![endif]-->

<!--[if gte IE 9]><!-->

<script src="<?=CUtil::GetAdditionalFileURL(SITE\_TEMPLATE\_PATH.'/js/vendor/modernizr.min.js')?>"></script>

<!--<![endif]-->

</head>

<body>

<!--[if lt IE 8]>

<p class="chromeframe">Вы используете <strong>устаревший </strong> браузер. Пожалуйста <a

href="http://browsehappy.com/">

обновите свой браузер</a> или <a href="http://www.google.com/chromeframe/?redirect=true">установите Google Chrome

Frame</a> чтобы улучшить взаимодействие с сайтом.</p>

<![endif]-->

<?$APPLICATION->ShowPanel();?>

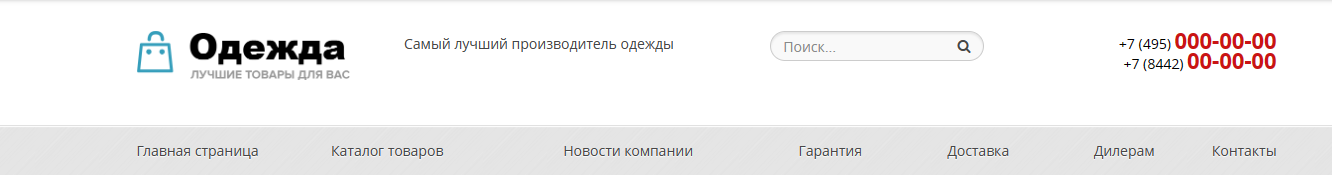


Рисунок 6 «header»

Footer

<?

if(!defined('B\_PROLOG\_INCLUDED') || B\_PROLOG\_INCLUDED !== true)

die();

?>

<div class="sticky-push"></div>

</div>

<footer>

<div class="sticky-footer">

<div class="container">

<div class="row">

<div class="col-md-4">

<address>

<?$APPLICATION->IncludeFile(

SITE\_DIR."include/address.php",

array(),

array(

"MODE" => "text"

)

);?>

</address>

</div>

<div class="col-md-4 col-md-push-4">

<div class="copyright">

Данный шаблон предоставлен компанией<br/>© <a href="http://www.intervolga.ru">ИНТЕРВОЛГА</a> для

Академии 1С-Битрикс

</div>

</div>

<div class="col-md-4 col-md-pull-4 hidden-print">

<?$APPLICATION->IncludeFile(

SITE\_DIR."include/social-links.php",

array(),

array(

"MODE" => "text"

)

);?>

</div>

</div>

</div>

</div>

</footer>



Рисунок 7 «footer»

## 3.3 Редактируемые области

Даём возможность администратору сайта менять текст в шаблоне сайта, без знаний HTML верстки и программирования.

<div class="col-sm-6 col-xs-12">

<div class="row">

<div class="col-lg-7 col-xs-12 hidden-xs">

<ul class="btn-list-inline">

<?$APPLICATION->IncludeFile(

SITE\_DIR."include/slogan.php",

array(),

array(

"MODE" => "text"

)

);?>

</ul>

</div>

Создаём папку «include» в корневой папке и закидываем туда slogan.php и вставляем туда «Самый лучший производитель одежды».

Также делаем с social-links.php, phone1.php, phone2.php, address.php.

## 3.4 Базовая настройка системы

Таблица №4 «Базовая настройка системы»

|  |  |
| --- | --- |
| Переходим в «Администрирование»- Настройки – Главный модуль  Даем название сайта |  |
| Убираем все галочки в «Оптимизация CSS» |  |
| Вставляем ключ сервиса и все сохраняем |  |
| Переходим в «Информационные блоки» Ставим галочка у 1,3 и 4 пунктов. Сохранить |  |

## 3.5 Компоненты и структура сайта

Таблица №5 «Компоненты и структура сайта»

|  |  |
| --- | --- |
| Переходим в «Администрирование»- Контент – Файлы и папки – Добавить – Добавить папку. Заполняем пустые ячейки и проставляем галочки. Сохранить |  |
| Редактирование страницы. Вставляем нужный текст  Делаем это еще для двух страниц |  |
| Добавляем пункты меню на главную страницу |  |
| Добавляем папку Components |  |
| Изменили header |  |
| В папке Components – menu – menu – template.php  Вставляем код | <nav>  <div class="navbar navbar-intervolga">  <div class="container">  <div class="navbar-header">  <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#top-nav">  <span class="sr-only">Переключить навигацию</span>  <span class="icon-bar"></span>  <span class="icon-bar"></span>  <span class="icon-bar"></span>  </button>  <a class="navbar-brand" href="/">InterVolga.ru</a>  </div>  <div class="collapse navbar-collapse" id="top-nav">  <ul class="nav navbar-nav"> |

# Разработка API для бэкэнда

## Разработка сервера

API (программный интерфейс приложения, интерфейс) (application programming interface, API [эй-пи-ай]) — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

Api может быть разным. Мы будем делать REST api.

REST (Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры.

Самое главное, что тут сказано - это архитектурный стиль. Т.е. это такой свод правил или рекомендаций. Это не отдельный протокол. REST работает на основе протокола http.

Начнем с виртуального окружения. Давайте создадим в директории Home папку с нашим проектом и назовем ее my\_drf\_blog. Затем перейдем в нее и внутри создадим виртуальное окружение myvenv с версией python3.8.

mkdir my\_drf\_blog

cd my\_drf\_blog

python3 -m venv myvenv

source myvenv/bin/activate

Устанавливаем Django:

pip install Django

Теперь создадим проект. Так и назовем его api.

django-admin startproject api

Теперь перейдем в папку с проектом и создадим наше приложение core

cd api

python manage.py startapp core

Добавляем в файл api/settings.py строки

import os #<- импортируй

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'core' #<- добавьте это

]

#<- добавь русскую локализацию

LANGUAGE\_CODE = 'ru'

STATIC\_URL = '/static/'

MEDIA\_URL = '/media/' #<- добавьте путь к папке с медифайлами

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media') # и путь до нее

Теперь создадим миграции и суперпользователя

python manage.py migrate

python manage.py createsuperuser

А теперь запустим веб-сервер

python manage.py runserver

В файле models.py создаем модель «Product»

**class** Product(models.Model):  
 objects = **None** title = models.CharField(**'Наименование'**, max\_length=50)  
 description = models.TextField(**'Описание'**)  
 slug = models.SlugField()  
 price = models.IntegerField(**'Цена'**)  
 photo = models.ImageField()  
  
 @property  
 **def** photo\_url(self):  
 **if** self.photo **and** hasattr(self.photo, **'url'**):  
 **return** self.photo.url  
  
 **def** \_\_str\_\_(self):  
 **return** self.title  
  
 **class** Meta:  
 verbose\_name = **'Товар'** verbose\_name\_plural = **'Товары'**

Теперь идем в файл urls.py нашего проекта и добавляем:

urlpatterns = [

path('admin/', admin.site.urls),

path("api/", include('core.urls')),

path("api/token/", TokenObtainPairView.as\_view(), name="token"), path("api/refresh\_token/", TokenRefreshView.as\_view(), name="refresh\_token"),

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

Токен API, часто называемый токеном доступа, аутентификации или авторизации, представляет собой уникальный идентификатор, который предоставляет ограниченный доступ к определенным ресурсам и службам, предоставляемым через API приложения. Эти токены играют решающую роль в обеспечении безопасности и сохранении конфиденциальности пользователей в области веб-, мобильных и серверных приложений, особенно при доступе к данным или выполнении действий внутри приложения.

Не забудем установить библиотеку Pillow для работы с изображениями:

pip install pillow

Теперь создадим миграцию и применим ее:

python manage.py makemigrations

python manage.py migrate

**Сериализатор (Serializer)** — это компонент Django REST Framework, который выполняет следующие задачи:

1. Конвертирует произвольные объекты языка Python в формат JSON (в том числе модели фреймворка Django и наборы QuerySet).
2. Преобразует JSON-данные обратно в соответствующие объекты Python.

Давайте внутри нашего приложения создадим файл core/serializers.py.

**class** PostSerializer(serializers.ModelSerializer):  
  
 **class** Meta:  
 model = Product  
 fields = (**"title"**, **"description"**, **"price"**, **"photo"**,**"slug"**)  
 lookup\_field = **'slug'** extra\_kwargs = {  
 **'url'**: {**'lookup\_field'**: **'slug'**}  
 }

Итак, откроем файл views.py и добавим в него:

class PostViewSet(viewsets.ModelViewSet):

serializer\_class = PostSerializer

queryset = Post.objects.all()

lookup\_field = 'slug'

В папке core создаём файл urls.py и пишем в него

router = DefaultRouter()

router.register('posts', PostViewSet, basename='posts')

urlpatterns = [

path("", include(router.urls))

]

Тестируем API

Прежде чем приступить к тестированию, добавьте записей в базу данных.

**core/admin.py**

class PostAdmin(admin.ModelAdmin):

pass

admin.site.register(Post, PostAdmin)

Далее запускаем сервер и заходим в панель администратора. Перейдите в «Товары» и добавьте 15 записей.

Тестирование с помощью Postman

Postman — это сервис для создания, тестирования, документирования, публикации и обслуживания API. Он позволяет создавать коллекции запросов к любому API, применять к ним разные окружения, настраивать мок-серверы, писать автотесты на JavaScript, анализировать и визуализировать результаты запросов.

После установки вам необходимо нажать кнопку + New

Затем в появившемся окне нажмите кнопку Collection

Назовите ее как вам захочется и нажмите кнопку Create

Далее в левой колонке выберите Collections и найдите свою только что созданную коллекцию.

Затем нажмите на три точечки в правом нижнем углу и выберите Add request.

Затем откроется окно добавления запроса. Укажите в нем название token и кнопку Save to test my blog.

После добавления запроса, он появится у нас в нашей коллекции. Щелкните на нем, чтобы открыть окно запроса.

Давайте выберем метод нашего запроса POST

В строку адреса запроса введем адрес http://localhost:8000/api/token/. Это тот самый адрес, который у нас в api/urls.py нашего проекта. Просто для наглядности приведу содержимое этого файла.

Нажмите справа кнопку Save

Сервер должен быть запущен при тестировании

И отправим наш запрос нажав кнопку Send

В ответ мы получим status code 400 и вот такой JSON:

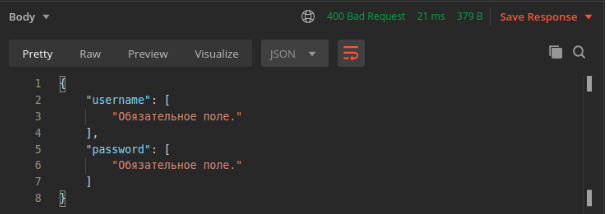


Рисунок 8. «JSON и status code»

Для того чтобы получить токен, нам необходимо авторизоваться.

Создадим структуру, которую будет отправлять на наш веб-сервер. Перейдём на вкладку Body

Выбираем формат «raw», затем рядом появится раскрывающийся список, выберите из него JSON

А теперь в тело запроса поместите вот такой JSON. Логин и пароль суперпользователя, которого мы создавали при установке django и drf:

{

"username": "admin",

"password": "1234"

}

Нажмите Send

Теперь status code 200 и в ответ мы получили access токен и refresh токен

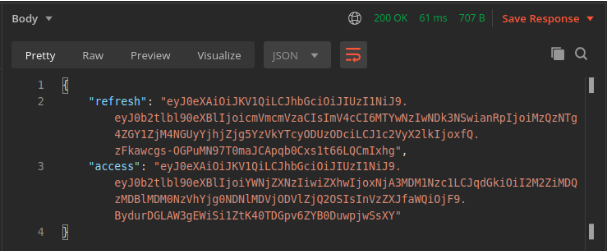


Рисунок 9. «Получение токена»

Добавим еще один запрос под названием refresh

Далее сменим метод на POST и укажем адрес http://localhost:8000/api/refresh\_token/. Этот адрес так же находится в файле api/urls.py. Данный метод нужен для того, чтобы обновлять access токен.

Если мы сейчас отправим запрос, то в ответ получим status code 400 и вот такой JSON

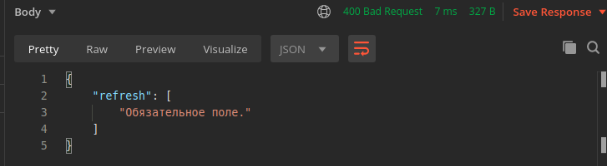


Рисунок 10. «JSON и status code»

Нам нужно на вкладку Body добавить JSON с refresh токеном, который мы получили из запроса <http://localhost:8000/api/refresh_token/>.

{

"refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTYwNzIwNDk3NSwianRpIjoiMzQzNTg4ZGY1ZjM4NGUyYjhjZjg5YzVkYTcyODUzODciLCJ1c2VyX2lkIjoxfQ.zFkawcgs-OGPuMN97T0maJCApqb0Cxs1t66LQCmIxhg"

}

Каждый раз, когда мы отправим запрос api/token нам надо ручками копировать токен и вставлять в этот запрос. Давайте лучше это автоматизируем, тем более что Postman дает нам такую возможность. Для начала нам нужно создать окружение. В правом верхнем углу найдите плашку с надписью No Enviroment

Нажмите рядом на "глазик" и выбире в появившемся окне в правом верхнем углу Add.

В появившемся окне укажите название окружения и добавьте в него 2 переменных access и refresh

Нажмите кнопку Добавить.

Затем в этой плашке выберите только что созданное окружение

Теперь идем в запрос **http://localhost:8000/api/token/**и переходим на вкладку Tests

В эту вкладку мы можем вставлять код, который будет что-то делать с полученными данными.

Вводим в тело refresh { "refresh": "{{refresh}}" }

Отправляем запрос

Отлично, мы получили status code 200 и обновленный access токен в ответе.

Добавим еще один запрос для получения всех постов.

Он получается работает по адресу api/posts

Метод запроса на этот раз GET, а адрес <http://localhost:8000/api/posts/>

Если мы отправим его, то получим в ответ status code 401 и сообщение, что "Учетные данные не были предоставлены."

Это значит, что мы не передали в заголовке наш access токен. Для того чтобы подключить заголовок Authorization нам надо перейти на вкладку Auth и в раскрывающемся списке выбрать Bearer token

В поле token укажите ссылку на нашу переменную access

Сохраните запрос нажав на кнопку Save

Отлично, теперь отправим наш запрос и в ответ получим status code 200 и список всех наших постов.

## Выполнение запросов к серверу

Было выполнено несколько запросов

**Последние 5 записей**

Давайте сделаем api aside, которая просто возвращает нам последние 5 постов.

class AsideView(generics.ListAPIView):

queryset = Post.objects.all().order\_by('-id')[:5]

serializer\_class = PostSerializer

permission\_classes = [permissions.AllowAny]

Добавляем адрес для нашего api в файл urls

path("aside/", AsideView.as\_view()),

Протестируем наш api

Как видите, запрос типа **GET**по адресу

**http://localhost:8000/api/aside/**возвращает нам последние 5 постов из базы**.**

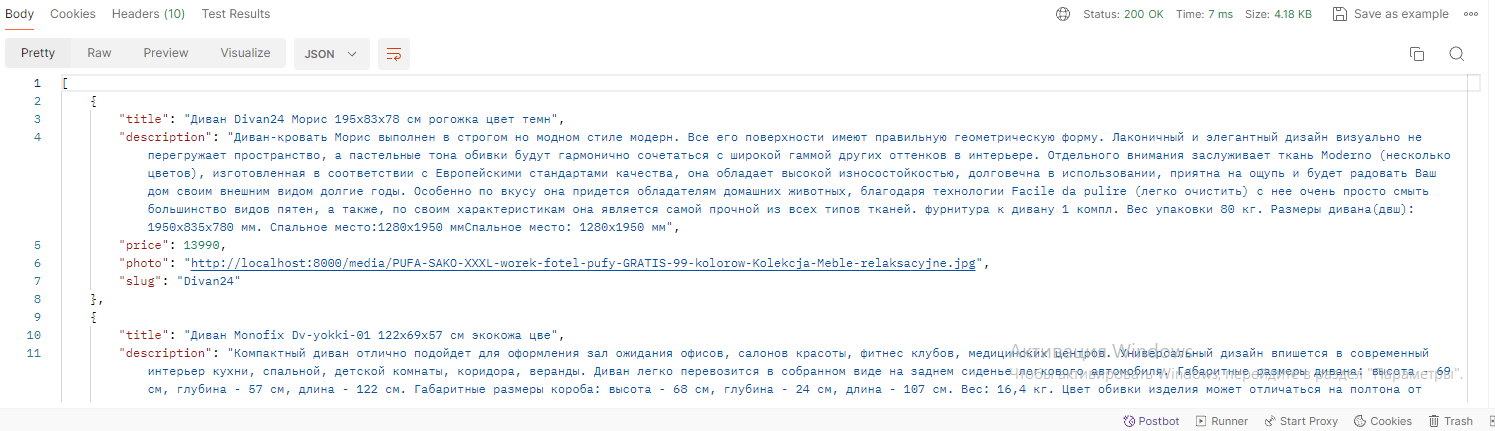


Рисунок 11. «Вывод последних 5 записей»

**Поиск**

Итак, у нас есть PostViewSet. Давайте в нее просто добавим пару строчек.

search\_fields = ['content', 'h1']

filter\_backends = (filters.SearchFilter,)

Выполняем запрос по поиску

http://localhost:8000/api/posts/?q=Кровать Топчанъ Astra 90.1 90x200 см массив сосны



Рисунок 12. «Поиск товара»

**Регистрация**

Для того чтобы реализовать регистрацию нам необходимо создать сериалайзер, view для регистрации и url.

**core/serializers.py**

from django.contrib.auth.models import User

class RegisterSerializer(serializers.ModelSerializer):

password2 = serializers.CharField(write\_only=True)

class Meta:

model = User

fields = [

"username",

"password",

"password2",

]

extra\_kwargs = {"password": {"write\_only": True}}

def create(self, validated\_data):

username = validated\_data["username"]

password = validated\_data["password"]

password2 = validated\_data["password2"]

if password != password2:

raise serializers.ValidationError({"password": "Пароли не совпадают"})

user = User(username=username)

user.set\_password(password)

user.save()

return user

Когда мы будем отправлять запрос на регистрацию, это будет выглядеть так:

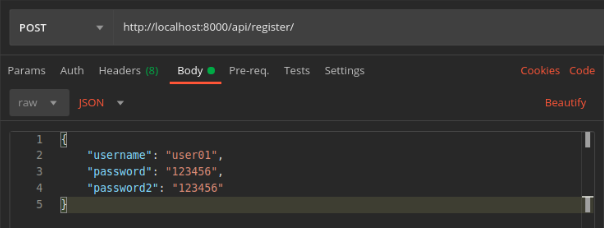


Рисунок 13. «Тело запроса»

Создаем view для регистрации

class RegisterView(generics.GenericAPIView):

permission\_classes = [permissions.AllowAny]

serializer\_class = RegisterSerializer

def post(self, request, \*args, \*\*kwargs):

serializer = self.get\_serializer(data=request.data)

serializer.is\_valid(raise\_exception=True)

user = serializer.save()

return Response({

"user": UserSerializer(user, context=self.get\_serializer\_context()).data,

"message": "Пользователь успешно создан",

})

Создаем url

**core/urls.py**

router = DefaultRouter()

router.register('posts', PostViewSet, basename='posts')

urlpatterns = [

path("", include(router.urls)),

path("tags/", TagView.as\_view()),

path("tags/<slug:tag\_slug>/", TagDetailView.as\_view()),

path("aside/", AsideView.as\_view()),

path('register/', RegisterView.as\_view()),

path('profile/', ProfileView.as\_view()),

]

Протестируем наше api

Давайте добавим в Postman запрос под название **registration.** Тип запроса будет **POST.** Эндпойнт **/api/register/.**

Структура JSON:

{

"username": "user07569",

"password": "123456",

"password2": "123456"

}



Рисунок 14. «Регистрация прошла успешно»

В ответ, мы получили сообщение, что пользователь успешно создан, а так же данные по всем полям стандартной модели User.

**Комментарии**

Начать стоит с того, что нам нужно добавить модель комментариев в файл **models.py.**

**core/models.py**

class Comment(models.Model):

post = models.ForeignKey(Post, on\_delete=models.CASCADE, related\_name='comments')

username = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='user\_name')

text = models.TextField()

created\_date = models.DateTimeField(default=timezone.now)

class Meta:

ordering = ['-created\_date']

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

return self.text

**core/serializers.py**

class CommentSerializer(serializers.ModelSerializer):

username = serializers.SlugRelatedField(slug\_field="username", queryset=User.objects.all())

post = serializers.SlugRelatedField(slug\_field="slug", queryset=Post.objects.all())

class Meta:

model = Comment

fields = ("id", "post", "username", "text", "created\_date")

lookup\_field = 'id'

extra\_kwargs = {

'url': {'lookup\_field': 'id'}

}

**core/views.py**

class CommentView(generics.ListCreateAPIView):

queryset = Comment.objects.all()

serializer\_class = CommentSerializer

permission\_classes = [permissions.IsAuthenticated]

def get\_queryset(self):

post\_slug = self.kwargs['post\_slug'].lower()

post = Post.objects.get(slug=post\_slug)

return Comment.objects.filter(post=post)

**core/urls.py**

urlpatterns = [

path("", include(router.urls)),

path("tags/", TagView.as\_view()),

path("tags/<slug:tag\_slug>/", TagDetailView.as\_view()),

path("aside/", AsideView.as\_view()),

path('register/', RegisterView.as\_view()),

path('profile/', ProfileView.as\_view()),

path("comments/", CommentView.as\_view()),

path("comments/<slug:post\_slug>/", CommentView.as\_view()),

]

Тестирование

Первый для создания комментов. Назовем его add\_comment. Это будет POST запрос по адресу http://localhost:8000/api/comments/

Структура для создания коммента должна быть такой:

{

"post": " Dv-yokki",

"username": "admin",

"text": " Всё хорошо",

"Created date": ""

}

Получаем ответ

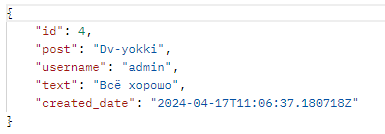


Рисунок 15. «Оставление комментария к товару»

**Заказы**

Начать стоит с того, что нам нужно добавить модель комментариев в файл **models.py**

**core/models.py**

**class** Orders(models.Model):  
 created = models.DateTimeField(verbose\_name=**'Дата'**,  
 null=**True**)  
 author = models.ForeignKey(**'auth.User'**, verbose\_name=**'Автор'**, related\_name=**'orders\_user'**,  
 on\_delete=models.CASCADE,  
 null=**True**)  
 prod = models.ForeignKey(**'Product'**, verbose\_name=**'Товары'**, related\_name=**'orders\_product'**, on\_delete=models.CASCADE,  
 null=**True**)  
 description = models.TextField(verbose\_name=**'Описание'**,  
 null=**True**)  
  
 **def** \_\_str\_\_(self):  
 **return** str(self.created)  
  
 **class** Meta:  
 ordering = [**'created'**]  
 verbose\_name = **'Заявка'** verbose\_name\_plural = **'Заявки'**

**core/serializers.py**

**class** OrdersSerializer(serializers.ModelSerializer):  
  
 author = serializers.SlugRelatedField(slug\_field=**"username"**, queryset=User.objects.all())  
 prod = serializers.SlugRelatedField(slug\_field=**"slug"**, queryset=Product.objects.all())  
  
 **class** Meta:  
 model = Orders  
 fields = (**"created"**, **"author"**, **"prod"**, **"description"**)  
 lookup\_field = **'author'** extra\_kwargs = {  
 **'url'**: {**'lookup\_field'**: **'author'**}  
 }

**core/views.py**

**class** OrdersView(generics.ListCreateAPIView):  
 queryset = Orders.objects.all()  
 serializer\_class = OrdersSerializer  
 permission\_classes = [permissions.IsAuthenticated]  
  
 **def** get\_queryset(self):  
 prod\_slug = self.kwargs[**'post\_slug'**].lower()  
 prod = Product.objects.get(slug=prod\_slug)  
 **return** Orders.objects.filter(prod=prod)

**core/urls.py**

urlpatterns = [  
 path(**""**, include(router.urls)),  
 path(**"aside/"**, AsideView.as\_view()),  
 path(**'register/'**, RegisterView.as\_view()),  
 path(**'profile/'**, ProfileView.as\_view()),  
 path(**"comments/"**, CommentView.as\_view()),  
 path(**"comments/<slug:post\_slug>/"**, CommentView.as\_view()),  
 path(**"orders/"**, OrdersView.as\_view()),  
 path(**"orders/<slug:post\_slug>/"**, OrdersView.as\_view()),  
  
]

Тестирование

Первый для создания заказа. Назовем его add\_orders. Это будет POST запрос по адресу http://localhost:8000/api/orders/

Структура для создания заказа должна быть такой:

{

    "prod": "Dv-yokki",

    "author": "admin",

    "description": "Очень жду",

    "Created date": ""

}

Получаем ответ

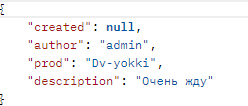


Рисунок 16. «Оформление заказа на товар»

# Проектирование предметной области

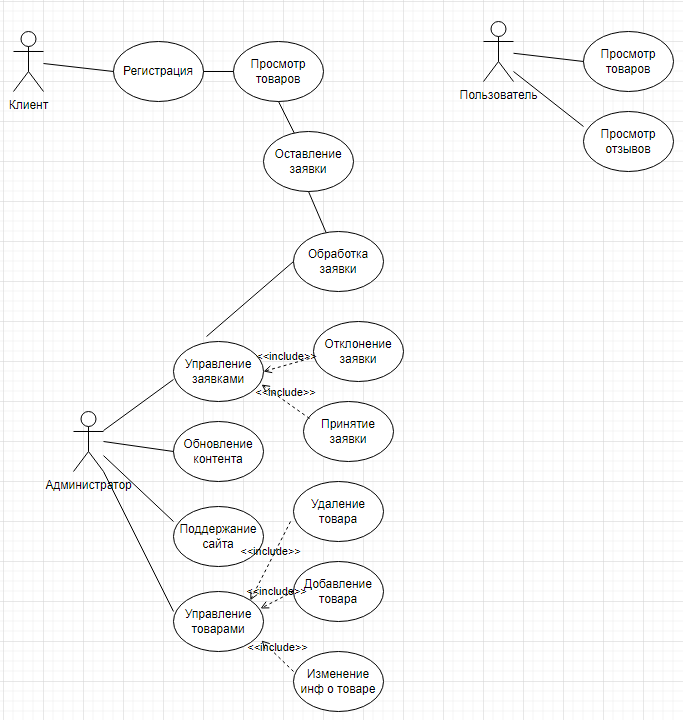


Рисунок 17. «Диаграмма прецедентов»

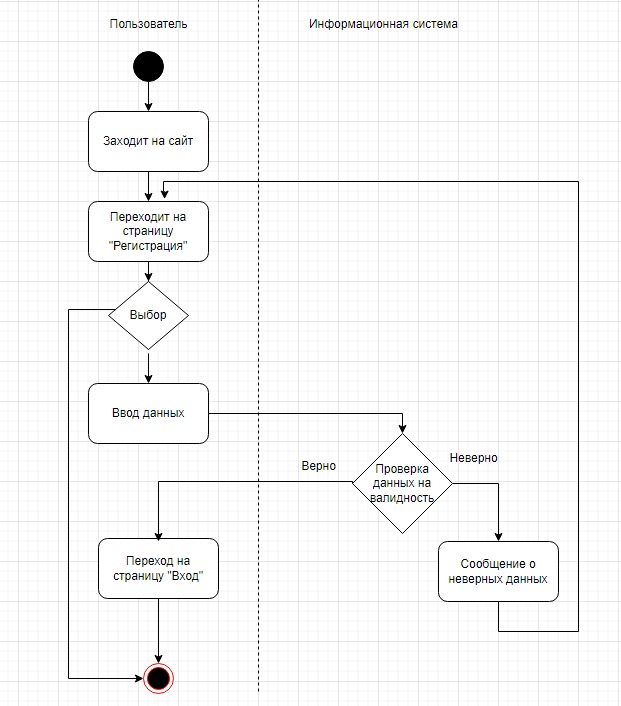


Рисунок 18. «Диаграмма действий»

# Разработка технической документации

## Техническое задание

**Введение**

Мебельные компании -это предприятия, специализирующиеся на производстве и продаже мебели. Они занимаются созданием широкого ассортимента изделий - от кухонных гарнитуров и спален до офисной мебели, и предметов интерьера.

Основная цель — предоставить покупателям возможность оставить заявку на нужную мебель прямо на сайте и при этом показать всю необходимую информацию о будущей покупке.

Для достижения данной цели были поставлены следующие задачи:

1) Изучить структуру мебельного магазина и бизнес-процессы, происходящие в нем.

2) Спроектировать архитектуру сайта, разработать алгоритмы работы с данными.

3) Протестировать работоспособность сайта, внести корректировки при необходимости.

На сайте есть такой функционал как: оставление заявки на какой-либо товар, регистрация и авторизация, просмотр подробной информации о товаре, фильтрация, оставление отзывов, просмотр архивных заявок, просмотр отзывов.

**Основания для разработки**

Разработка проводится на основании Договора Между ЛДПК и «Торговый дом «Бэст-мебель-трейд» № 3 от 16 апреля 2024г.

Наименование работы: «Торговый дом «Бэст-мебель-трейд»

**Назначение разработки**

Сайт предназначен для ознакомления посетителей с компанией,

возможностью предоставить покупателям возможность оставить заявку на нужную мебель прямо на сайте и при этом показать всю необходимую информацию о будущей покупке.

**Требования к программе или программному изделию**

**Требования к функциональным характеристикам**

**Требования к составу выполняемых функций:**

* Добавление, удаление и редактирование товаров
* Фильтрация товара по критериям: «Наименование», «Цена»
* Организация страницы с товарами и фильтрацией
* Организация страницы с отзывами
* Организация страницы «Об услуге» с подробным описанием услуги
* Организация страницы «Заявки» с информацией об актуальных и архивных заявках
* Организация страницы «Выбрать» с возможностью оставить заявку на нужный товар

**Требования к организации входных данных:**

Входные данные: имя пользователя, email, пароль, наименование, цена, фото, описание, отзывы, заявка.

**Требования к организации выходных данных:**

На странице «Главная» можно увидеть карточки с товарами. Если пользователь авторизован, то у него будет оставить заявку на товар. Если пользователь не авторизован, то он сможет просмотреть товары, но не сможет оставить заявку на товар до тех пор, пока не выполнит вход или регистрацию на сайте.

На странице «Заявки» представлена актуальная информация и действующих заявках и об архивных

На странице «Вход» предоставляются поля для ввода логина и пароля

На странице «Регистрация» предоставляются поля для ввода данных, таких как: Имя пользователя, Email, пароль, подтверждение пароля.

**Требования к надежности.**

Сайт доложен отвечать следующим требованиям:

* Хеширование пароля — процесс преобразования исходного пароля в непредсказуемую строку фиксированной длины. Хеш-функции могут быть однонаправленными, а это значит, что невозможно восстановить исходный пароль из его хеша.
* Доступность сайта – позволяет пользователям в любое время посмотреть сайт и его содержимое.
* Целостность данных при частичном выходе из строя программно-аппаратных.
* Поддержка и обслуживание сайта- после создания сайта и его загрузки на хостинг забота о безопасности и надежности сайта не заканчивается. Необходимо соблюдать определенные правила, чтобы сайт не пострадал от угроз.

Значения показателей надёжности:

* режим работы в целом – 7 дней в неделю 24 часа в сутки.

**Условия эксплуатации**

Программа не требует специального обслуживания. Для ознакомления с полным функционалом пользователь должен прочесть Руководство пользователя. Для работы с программой требуются хотя бы навыки работы с приложениями с похожим интерфейсом, содержащими информацию в виде таблиц БД.

**Климатические условия эксплуатации:**

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

**Требования к составу и параметрам технических средств**

Таблица №6. «Состав технических средств и их характеристики»

|  |  |
| --- | --- |
| **Технические средства** | **Требования** |
| Браузеры | Последняя версия Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera и др. |
| Устройства вывода | Устройства с разрешением экрана не менее 1280x720 пикселей |
| Устройства ввода | Клавиатура и мышь |
| Операционные системы | Windows 7, 8, 10, macOS 10.12 или новее, iOS 11 или новее, Android 5 или новее и другие ОС |

**Требования к хостингу**

• Обработка .htaccess.

• Поддержка SQLlite

• Наличие интерпретатора python: любой настроенный и стабильно работающий акселератор (eAccelerator, APC, Zend Server CE или другой).

• Поддержка SSL в PHP (возможность работы с SSL сокетами).

**Программные средства, используемые во время испытаний:**

Для корректной работы программы необходимо: ОС Windows 10 и выше, Chrome, Edge, Яндекс, Opera.

**Специальные требования**

Сайт должен обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса.

**Требования к программной документации**

В ходе разработки сайта должны быть подготовлены следующие программные документы: программа и методика испытаний, руководство пользователя, руководство программиста.

## Руководство пользователя

**1) Назначение программы:**

Сайт предназначен для ознакомления посетителей с компанией,

возможностью предоставить покупателям возможность оставить заявку на нужную мебель прямо на сайте и при этом показать всю необходимую информацию о будущей покупке.

**2) Условия выполнения программы:**

Таблица №7. «Состав технических средств и их характеристики»

|  |  |
| --- | --- |
| **Технические средства** | **Требования** |
| Браузеры | Последняя версия Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera и др. |
| Устройства вывода | Устройства с разрешением экрана не менее 1280x720 пикселей |
| Устройства ввода | Клавиатура и мышь |
| Операционные системы | Windows 7, 8, 10, macOS 10.12 или новее, iOS 11 или новее, Android 5 или новее и другие ОС |

**Требования к хостингу**

• Обработка .htaccess.

• Поддержка SQLlite

• Наличие интерпретатора python: любой настроенный и стабильно работающий акселератор (eAccelerator, APC, Zend Server CE или другой).

• Поддержка SSL в PHP (возможность работы с SSL сокетами).

**Программные средства, используемые во время испытаний:**

Для корректной работы программы необходимо: ОС Windows 10 и выше, Chrome, Edge, Яндекс, Opera.

**3) Выполнение программы:**

На «Главной» странице находится блок фильтрации, с помощью которого можно найти конкретный товар, имеется каталог товаров с возможностью посмотреть подробную информацию и, если пользователь авторизован, у него появляется кнопка «Выбрать» и он может оставить заявку на какую-то мебель.

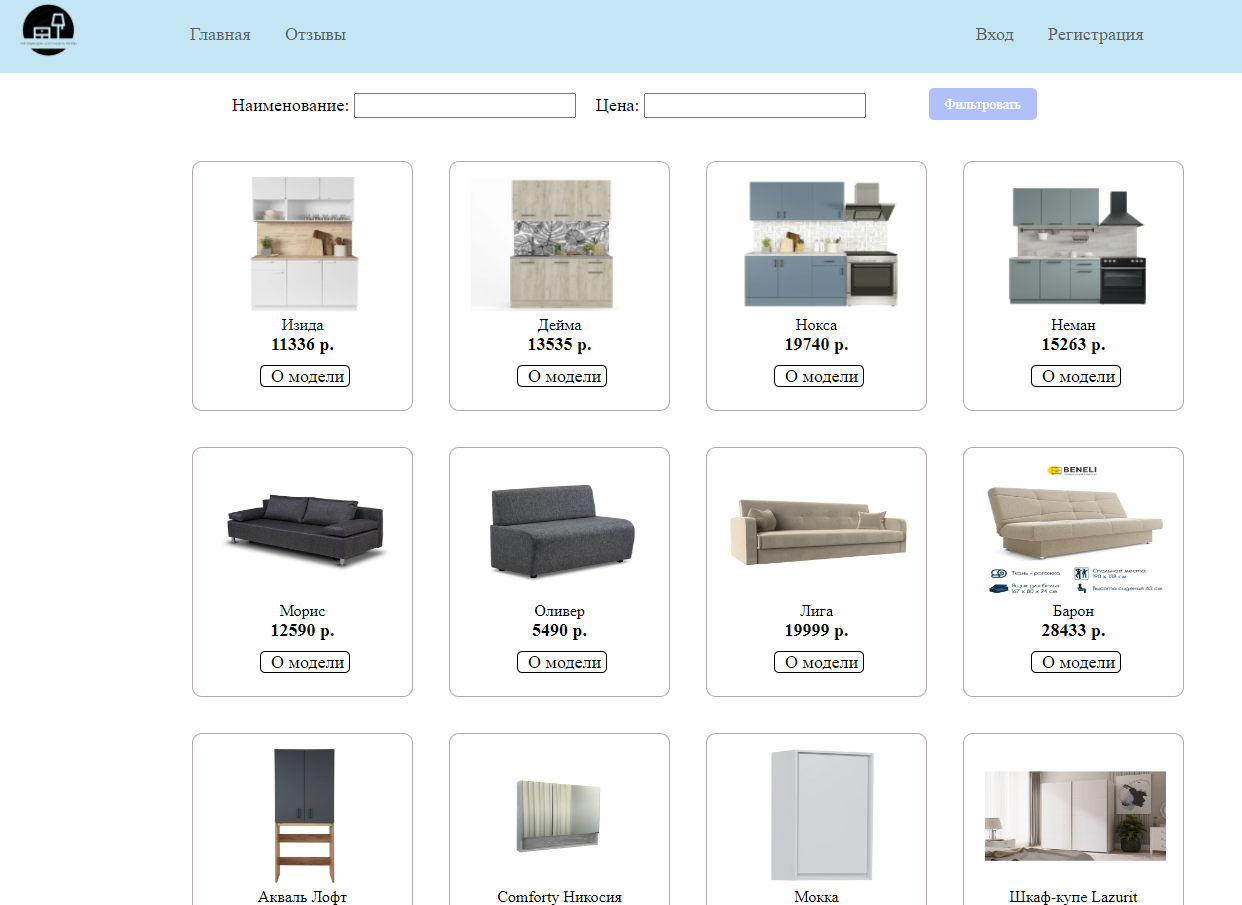


Рис 19 «Главная страница»

На странице «Отзывы» есть блок отзывов клиентов и, если пользователь авторизован имеется возможность написать отзыв с помощью кнопки. После нажатия на данную кнопку пользователя перекидывает на страницу написания отзывов.

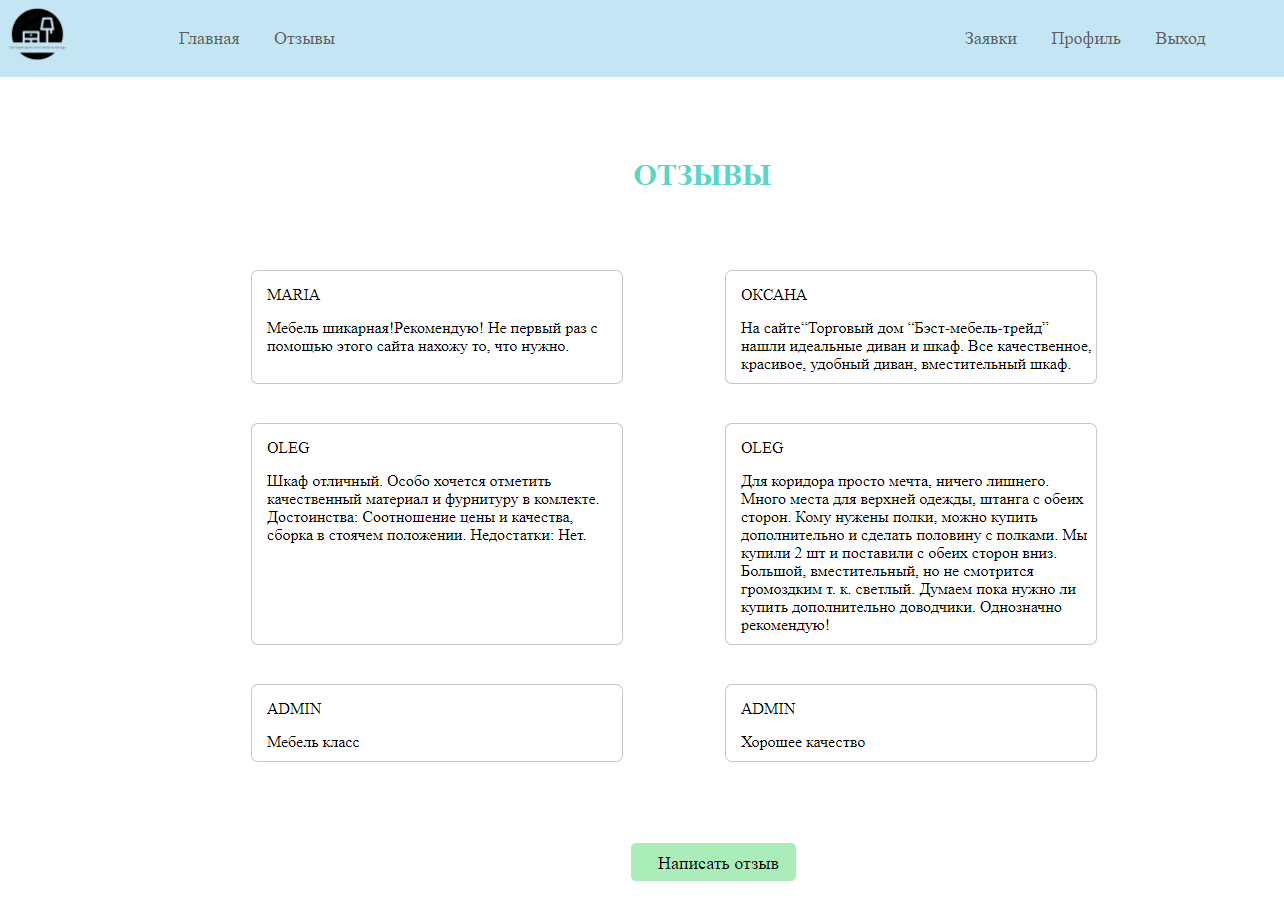


Рис 20 «Отзывы»

На странице «Заявки» отображаются архивные и активные заявки, которые можно отменить и просмотреть все свои заявки.

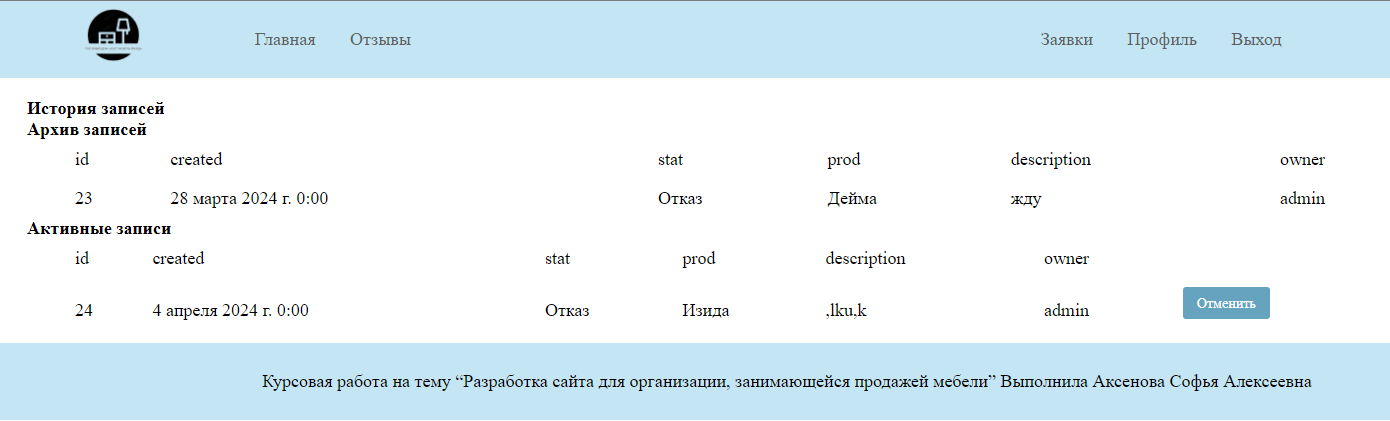


Рис 21 «Заявки»

На странице «Профиль» отображается актуальная информация о профиле, такая как: почта, имя пользователя и фото. Данные можно изменить на этой странице нажав на кнопку обновить.

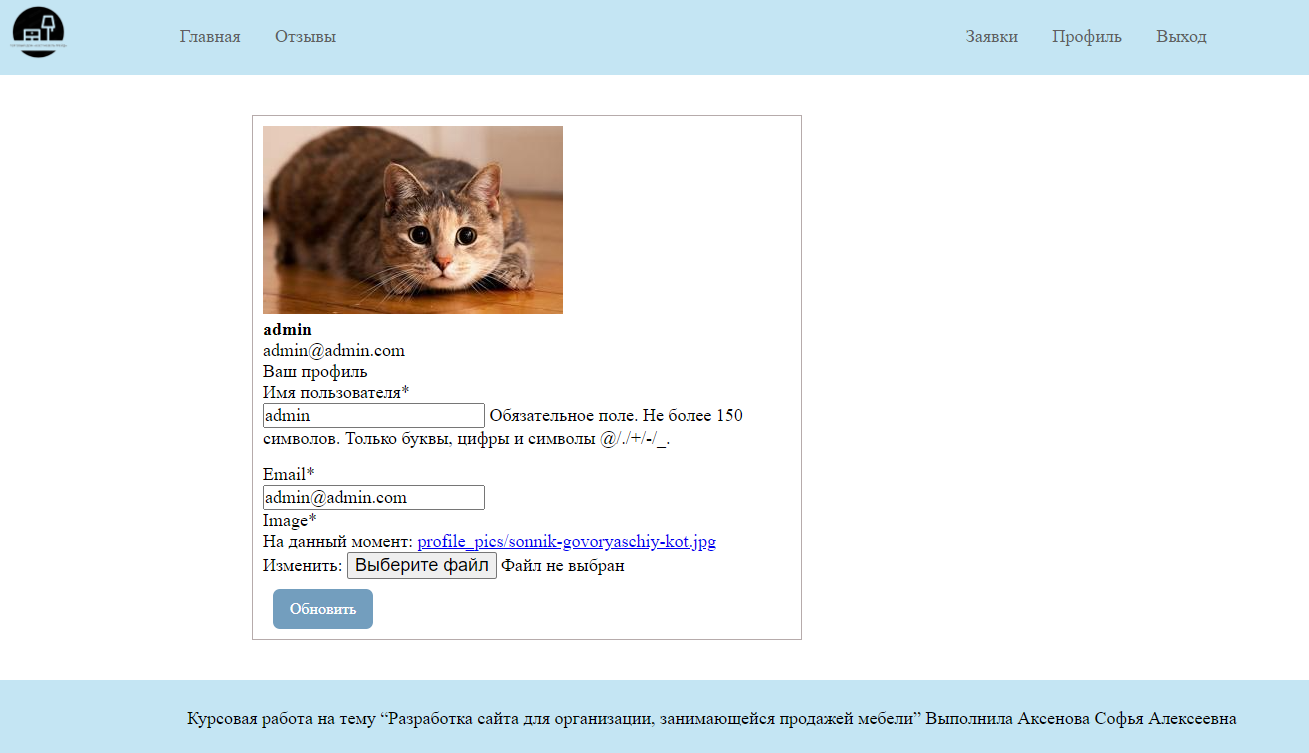


Рис 22 «Профиль»

На странице «Выход» имеется кнопка входа, нажав на которую можно заново войти в аккаунт.

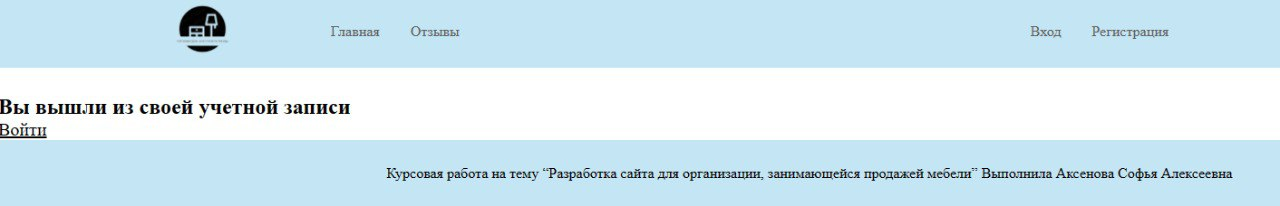


Рис 23 «Выход»

На странице «Вход» имеются поля ввода имени пользователя, пароля, кнопка входа в аккаунт и ссылка на страницу регистрации, если еще нет аккаунта.

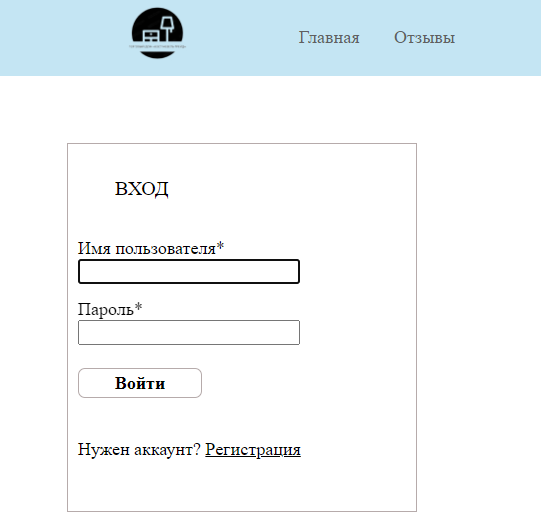


Рис 24 «Вход»

На странице «Регистрация» имеются поля ввода имени пользователя, email, пароля, кнопка регистрации и ссылка на страницу входа, если есть аккаунт.

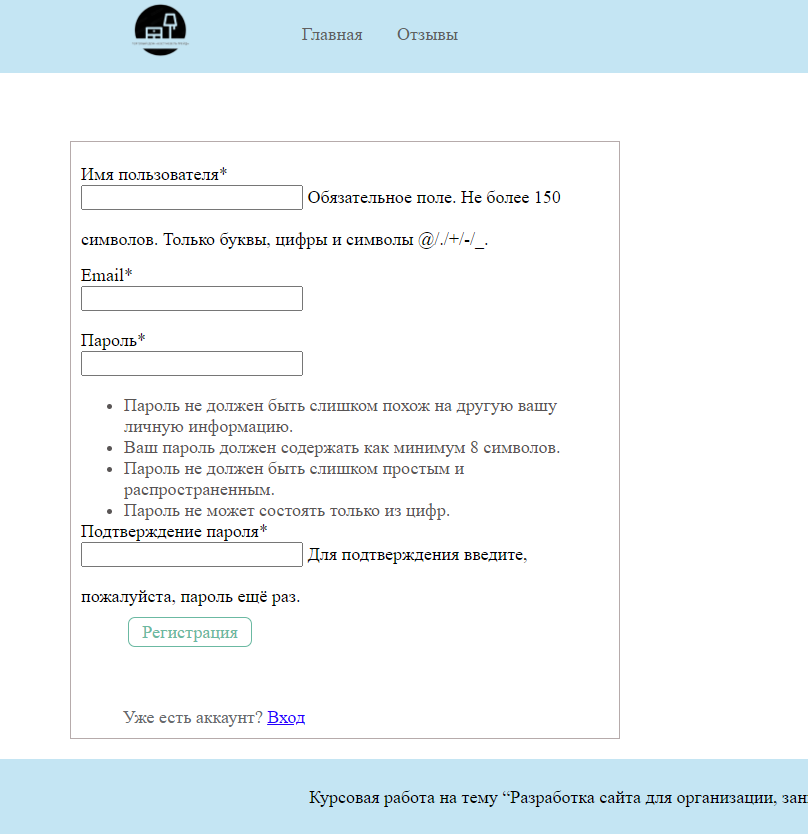


Рис 25 «Регистрация»

На странице «Подробнее о модели» находится подробная информация о конкретном товаре такая как: фотография, название, описание, цена.



Рис 26 «Подробнее о моделе»

На странице «Оставить заявку» находятся поля: дата создания, наименование товара, описание и кнопка отправить.

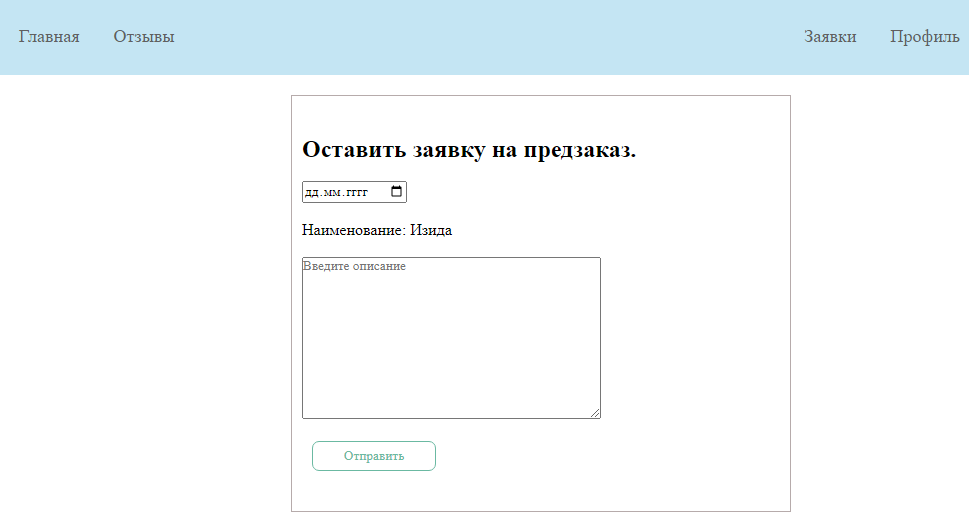


Рис 27 «Оставить заявку»

**4) Сообщение оператору:**

* Попытка регистрации с неуказанными данными



Рис.28 «Попытка регистрации с неуказанными данными»

* Ввод неверных данных при входе

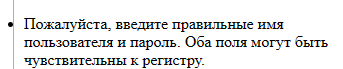


Рис.29 «Ввод неверных данных при входе»

* Ввод почты без специальных символов



Рис.30 «Ввод почты без специальных символов»

## Руководство программиста

**1) Назначение и условия применения программы:**

Сайт предназначен для ознакомления посетителей с компанией,

возможностью предоставить покупателям возможность оставить заявку на нужную мебель прямо на сайте и при этом показать всю необходимую информацию о будущей покупке.

Таблица №8. «Состав технических средств и их характеристики»

|  |  |
| --- | --- |
| **Технические средства** | **Требования** |
| Браузеры | Последняя версия Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera и др. |
| Устройства вывода | Устройства с разрешением экрана не менее 1280x720 пикселей |
| Устройства ввода | Клавиатура и мышь |
| Операционные системы | Windows 7, 8, 10, macOS 10.12 или новее, iOS 11 или новее, Android 5 или новее и другие ОС |

**Требования к хостингу**

• Обработка .htaccess.

• Поддержка SQLlite

• Наличие интерпретатора python: любой настроенный и стабильно работающий акселератор (eAccelerator, APC, Zend Server CE или другой).

• Поддержка SSL в PHP (возможность работы с SSL сокетами).

**Программные средства, используемые во время испытаний:**

Для корректной работы программы необходимо: ОС Windows 10 и выше, Chrome, Edge, Яндекс, Opera.

**2) Характеристика программы:**

Программа должна обеспечивать следующие функции:

* Добавление, удаление и редактирование товаров
* Фильтрация товара по критериям: «Наименование», «Цена»
* Организация страницы с товарами и фильтрацией
* Организация страницы с отзывами
* Организация страницы «Об услуге» с подробным описанием услуги
* Организация страницы «Заявки» с информацией об актуальных и архивных заявках
* Организация страницы «Выбрать» с возможностью оставить заявку на нужный товар

**3) Обращение к программе:**

Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации.

PyCharm – это интегрированная среда разработки на языке Python, созданная компанией JetBrains. Среди питонистов эта IDE вторая по популярности после редактора кода Visual Studio Code: как основную её используют 31% разработчиков.

Код страницы «Главная»:

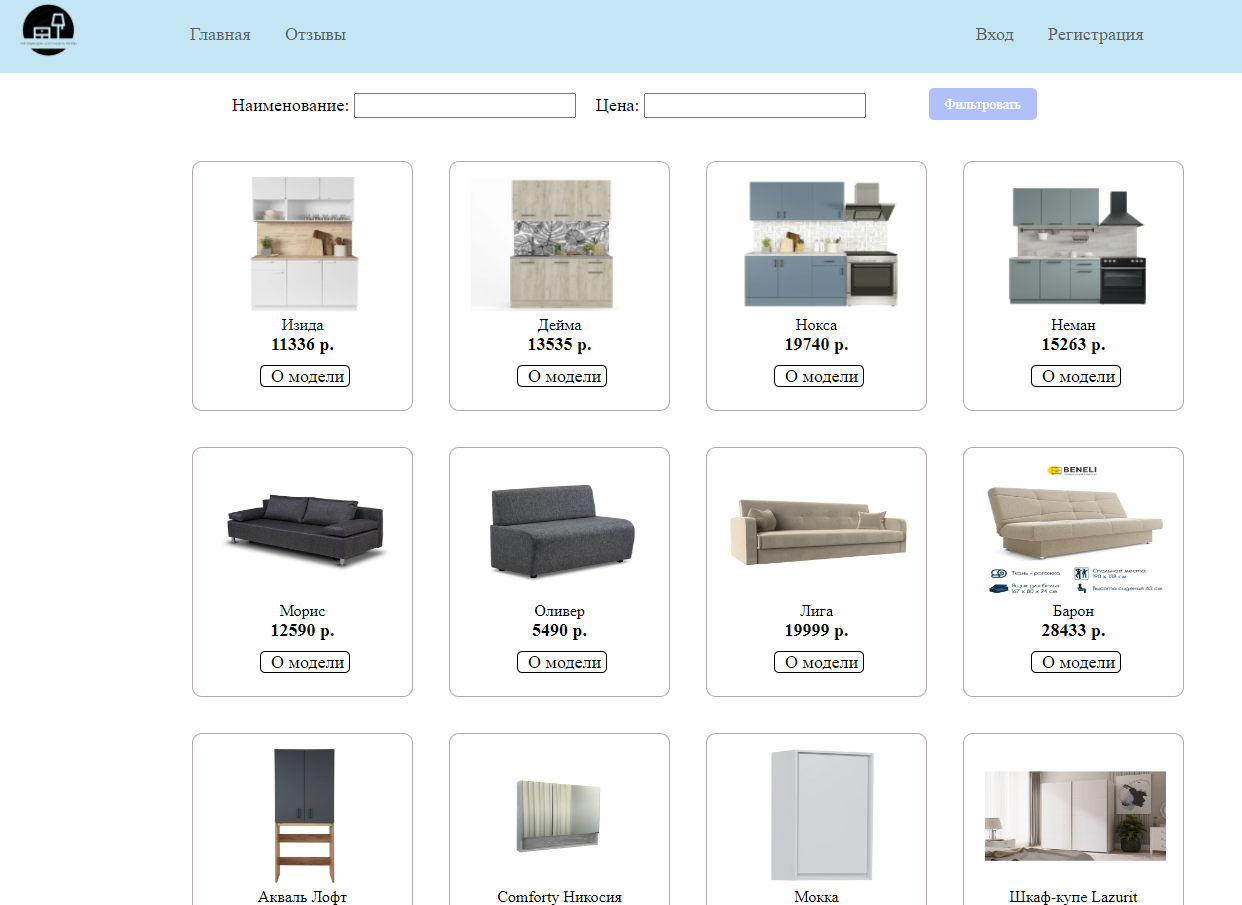


Рис.31 «Страница главная»

{% extends "blog/base.html" %}  
{% block content %}  
  
{% if posts %}  
<**div class="products"**>  
 <**div class="container"**>  
 <**form method="get" class="form-filter"**>  
 <**div**>  
 {{ posts.form.as\_p }}  
 <**button type="submit" class="btn btn-primary"**>Фильтровать</**button**>  
 </**div**>  
 </**form**>  
 </**div**>  
</**div**>  
  
<**div class="products-items"**>  
 {% for post in posts.qs %}  
 <**div class="product-item"**>  
 <**img width="181" height="134" src="media/{{ post.photo }}"**>  
 <**p class="title"**> {{ post.title }}</**p**>  
 <**p class="price"**><**b**>{{ post.price }} р.</**b**></**p**>  
 <**div class="knopki"**>  
 {% if user.is\_authenticated %}  
 <**input type="button" value="Выбрать" onclick="location.href = '{% url 'orders-create' post.id %}'"**>  
 {% endif %}  
 <**input type="button" class="model" value="О модели"  
 onclick="location.href = '{% url 'product-detail' post.id %}'"**>  
 </**div**>  
 </**div**>  
 {% endfor %}  
</**div**>  
{% else %}  
<**p**>Нет записей</**p**>  
{% endif %}  
{% endblock content %}

Код страницы «Отзывы»:

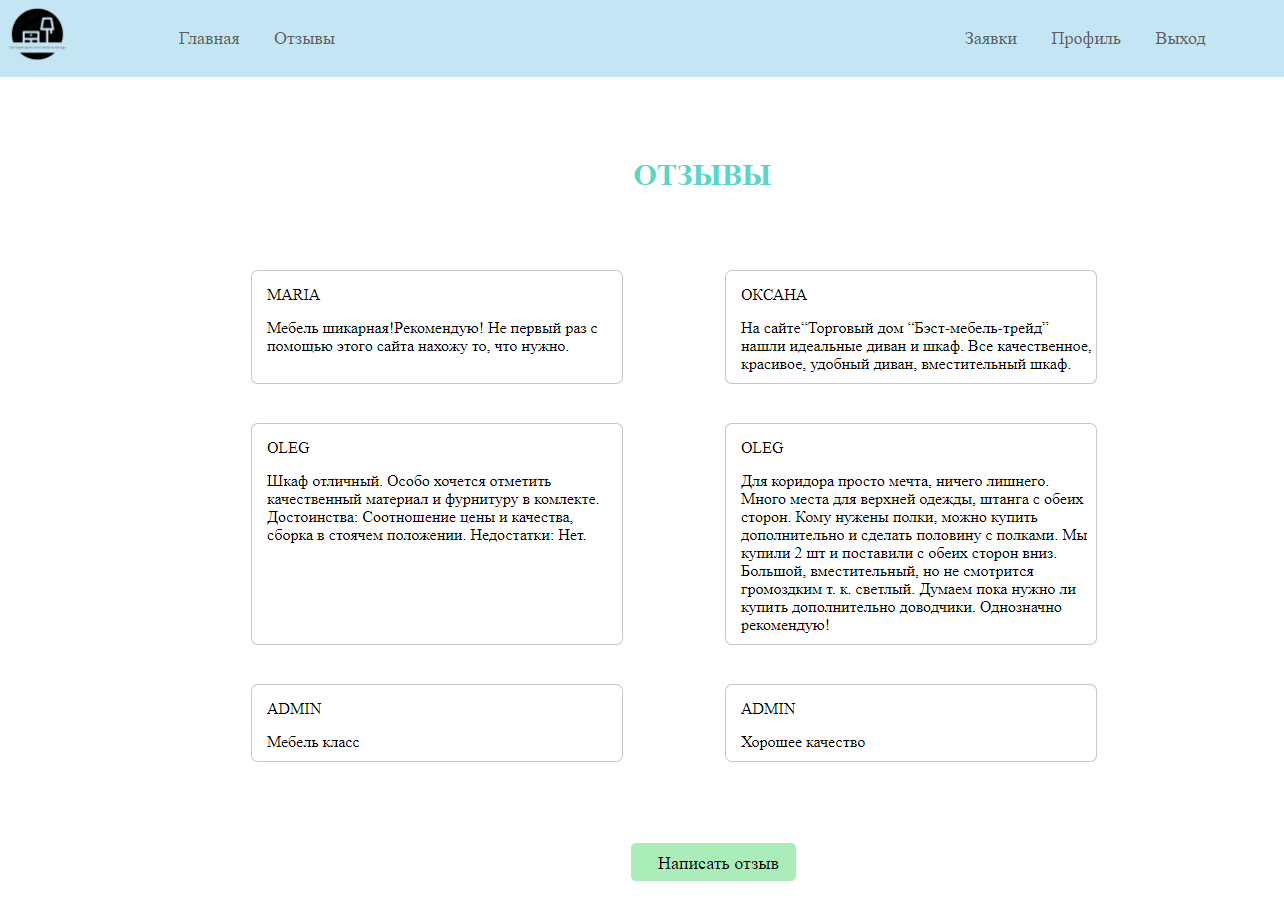


Рис.32 «Страница физических лиц»

{% extends "blog/base.html" %}  
{% block content %}  
  
<**h1 class="h1"**>ОТЗЫВЫ</**h1**>  
{% if reviews %}  
<**div class="rewievs"**>  
{% for review in reviews %}  
 <**article class="rewievs-item"**>  
 <**p**> {{ review.author }}</**p**>  
 <**p**> {{ review.content }}</**p**>  
 </**article**>  
{% endfor %}  
</**div**>  
{% else %}  
 <**p**>Нет записей</**p**>  
{% endif %}  
 {% if user.is\_authenticated %}  
 <**div class="aaa"**> <**a class="aa" href="{% url 'post-create' %}"**>Написать отзыв</**a**></**div**>  
 {% endif %}  
{% endblock content %}

**4) Входные и выходные данные:**

Таблица №9 «Входные данные в БД»

|  |  |
| --- | --- |
| **Поля** | **Тип данных** |
| Имя пользователя | varchar (150) |
| Email | varchar (254) |
| Пароль | varchar (128) |
| Наименование | varchar (50) |
| Цена | integer |
| Фото | varchar (100) |
| Описание | text |

Таблица №10 «Входные данные на сайте»

|  |  |
| --- | --- |
| **Поля** | **Тип данных** |
| Имя пользователя | TextField |
| Email | EmailField |
| Пароль | CharField |
| Наименование | CharField |
| Цена | IntegerField |
| Фото | ImageField |
| Описание | TextField |

**5) Выходная информация:**

На странице «Главная» можно увидеть карточки с товарами. Если пользователь авторизован, то у него будет оставить заявку на товар. Если пользователь не авторизован, то он сможет просмотреть товары, но не сможет оставить заявку на товар до тех пор, пока не выполнит вход или регистрацию на сайте.

На странице «Заявки» представлена актуальная информация и действующих заявках и об архивных

На странице «Вход» предоставляются поля для ввода логина и пароля

На странице «Регистрация» предоставляются поля для ввода данных, таких как: Имя пользователя, Email, пароль, подтверждение пароля.

**6) Сообщения об ошибках:**

Ошибка, возникающая при запуске сайта, не переходя в нужную директорию.

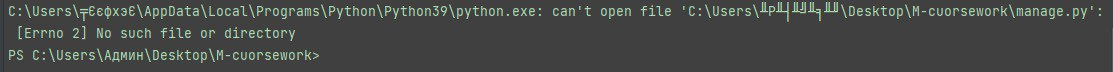


Рис.33 «Ошибка запуска сайта»

Ошибка, возникающая, когда программа не видит класс, служащий для написания отзывов, либо класс вообще отсутствует в структуре кода программы.

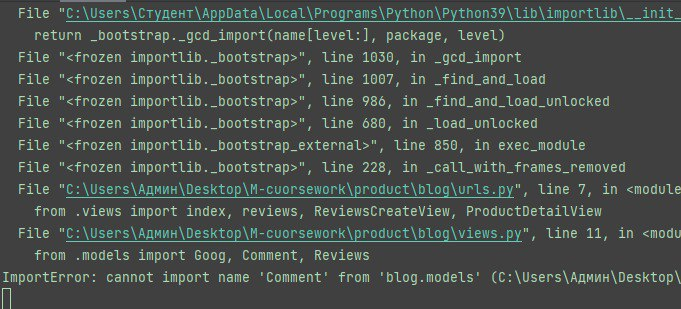


Рис.34 «Отсутствие класса»

Ошибка, возникающая, когда программа не видит модель, служащая для отображения продукции.

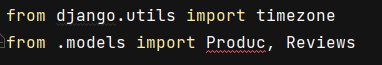


Рис.35 «Отсутствие класса»

## Методика тестирования и испытания программы

**1. Объект испытаний**

**Наименование**

«Торговый дом «Бэст-мебель-трейд»

**Область применения**

Сайт предназначен для ознакомления посетителей с компанией,

возможностью предоставить покупателям возможность оставить заявку на нужную мебель прямо на сайте и при этом показать всю необходимую информацию о будущей покупке.

**Обозначение программы**

«Торговый дом «Бэст-мебель-трейд»

**2. Цель испытаний**

Цель проведения испытаний - проверка соответствия характеристик разработанной программы (программного продукта) функциональным и отдельным иным видам требований, изложенным в документе Техническое задание.

**3. Общие положения**

**Основания для проведения испытаний**

Разработка проводятся на основании Договора Между Аксенова С.А и «Торговый дом «Бэст-мебель-трейд» № 1 от 12 апреля 2024 г.

**Место и продолжительность испытаний**

Приемосдаточные испытания должны проводиться на объекте заказчика в сроки с 12.05.2024 по 12.06.2024

Приемосдаточные испытания программы должны проводиться согласно разработанной исполнителем и согласованной с заказчиком Программы, и методики испытаний.

Ход проведения приемо-сдаточных испытаний заказчик и исполнитель документируют в Протоколе испытаний.

**Перечень документов, предъявляемых на испытания**

Состав программной документации должен включать в себя:

1. [Техническое задание](javascript:void(11959));
2. [Программу и методику испытаний](javascript:void(11958));
3. [Руководство системного программиста](javascript:void(12012));
4. [Руководство пользователя](javascript:void(12017));
5. Руководство программиста;
6. Текст программы;

**4) Объем испытаний**

**Перечень этапов испытаний**

1. **Ознакомительный –** проверка комплектности программной документации, техническое задание, программа и методика испытаний, руководство программиста, ведомость эксплуатационных документов.
2. **Испытания -** проверка комплектности программной документации, техническое задание, программа и методика испытаний, руководство программиста, руководство оператора, руководство системного программиста, ведомость эксплуатационных документов.

**Количественные характеристики, подлежащие оценке**

В ходе проведения приемосдаточных испытаний оценке подлежат количественные характеристики, такие как:

1. комплектность программной документации;
2. комплектность состава технических и программных средств.

**Качественные характеристики, подлежащие оценке**

В ходе проведения приемосдаточных испытаний оценке подлежат качественные (функциональные) характеристики программы. Проверке подлежит возможность выполнения программой перечисленных ниже функций:

* Добавление, удаление и редактирование товаров
* Фильтрация товара по критериям: «Наименование», «Цена»
* Организация страницы с товарами и фильтрацией
* Организация страницы с отзывами
* Организация страницы «Об услуге» с подробным описанием услуги
* Организация страницы «Заявки» с информацией об актуальных и архивных заявках
* Организация страницы «Выбрать» с возможностью оставить заявку на нужный товар

**Последовательность проведения и режимы испытаний**

Испытания проводятся в последовательности, указанной в п. «Перечень этапов испытаний».

**Перечень работ, проводимых после завершения испытаний**

В случае успешного проведения испытаний в полном объеме исполнитель совместно с заказчиком на основании Протокола испытаний утверждают Акт приемки-сдачи работ.

Исполнитель передает заказчику программное изделие, программную (эксплуатационную) документацию и т.д.

В случае выявления несоответствия разработанной программы отдельным требованиям технического задания исполнитель проводит корректировку программы и программной документации по результатам испытаний в сроки, согласованные с заказчиком.

По завершении корректировки программы и программной документации исполнитель и заказчик проводят повторные испытания согласно настоящей программы и методик в объеме, требуемом для проверки проведения корректировок.

**Порядок проведения испытаний**

Испытания должны проводиться поэтапно согласно п. «Перечень этапов испытаний» настоящего документа.

**5. Средства и порядок испытаний**

**Требования к составу и параметрам технических средств**

Таблица №11. «Состав технических средств и их характеристики»

|  |  |
| --- | --- |
| **Технические средства** | **Требования** |
| Браузеры | Последняя версия Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera и др. |
| Устройства вывода | Устройства с разрешением экрана не менее 1280x720 пикселей |
| Устройства ввода | Клавиатура и мышь |
| Операционные системы | Windows 7, 8, 10, macOS 10.12 или новее, iOS 11 или новее, Android 5 или новее и другие ОС |

**Требования к хостингу**

• Обработка .htaccess.

• Поддержка SQLlite

• Наличие интерпретатора python: любой настроенный и стабильно работающий акселератор (eAccelerator, APC, Zend Server CE или другой).

• Поддержка SSL в PHP (возможность работы с SSL сокетами).

**Программные средства, используемые во время испытаний:**

Для корректной работы программы необходимо: ОС Windows 10 и выше, Chrome, Edge, Яндекс, Opera.

**Условия проведения испытаний**

Испытания должны проводиться в нормальных климатических условиях по ГОСТ 22261-94. Условия проведения испытаний приведены ниже:

* температура окружающего воздуха, °С - 20 ± 5;
* относительная влажность, % - от 30 до 80;
* атмосферное давление, кПа - от 84 до 106;
* частота питающей электросети, Гц - 50 ± 0,5;
* напряжение питающей сети переменного тока, В - 220 ± 4,4.

**Условия начала и завершения отдельных этапов испытаний**

Необходимым и достаточным условием завершения 1 этапа испытаний и начала 2 этапа испытаний является успешное завершение проверок, проводимых на 1 этапе (см. п. «Перечень проверок, проводимых на 1 этапе испытаний»).

Условием завершения 2 этапа испытаний является успешное завершение проверок, проводимых на 2 этапе испытаний.

**Ограничения в условиях проведения испытаний**

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

**Требования к техническому обслуживанию**

Требования к техническому обслуживанию не предъявляются.

**Меры, обеспечивающие безопасность и безаварийность проведения испытаний**

При проведении испытаний заказчик должен обеспечить соблюдение требований безопасности.

**Порядок взаимодействия организаций, участвующих в испытаниях**

Исполнитель письменно извещает заказчика о готовности к проведению приемо-сдаточных испытаний не позднее чем за 14 дней до намеченного срока проведения испытаний.

Заказчик Договором назначает срок проведения испытаний и приемочную комиссии, которая должна включать в свой состав представителей заказчика и исполнителя.

Заказчик письменно извещает сторонние организации, которые должны принять участие в приемо-сдаточных испытаниях.

Заказчик совместно с исполнителем проводят все подготовительные мероприятия для проведения испытаний на объекте заказчика, а так же проводят испытания в соответствии с настоящей программой и методиками.

Заказчик осуществляет контроль проведения испытаний, а также документирует ход проведения проверок в Протоколе проведения испытаний.

**Требования к персоналу, проводящему испытания**

Персонал, проводящий испытания, должен иметь квалификацию, соответствующую профессиональной деятельности.

**Приложения**

Таблица №12 «Тестовый сценарий 1»

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_ПИ\_1. Добавление товара |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Добавление |
| **Краткое изложение теста** | Должно производится добавление услуги |
| **Этапы теста** | Запуск сайта  Переход в admin  Нажатие на кнопку «Товары»  Ввод данных в текстовые поля и выбор фотографии  Для занесения данных нажать на кнопку «Сохранить» |
| **Тестовые данные** | Наименование услуги – «Изида», фото услуги, цена услуги – 11336, описание товара – «Разнообразие тонов фасада не оставляет равнодушным поклонников богатого, но сдержанного стиля. Удивительно законченной кухню делают гармонично сочетающиеся стилистические элементы: актуальные оттенки, тонкие строгие линии фасада, разнообразие ящиков и полок, стеклянные дверцы, так напоминающие окна, в которые хочется заглянуть. Столешница в комплект не входит и приобретается отдельно. Входит в комплект: ножки, цоколь, петли, крепление для навесных шкафов, ручки. Приобретается дополнительно: столешница, стеновая панель, бытовая техника.» |
| **Ожидаемый результат** | Информация успешно сохранена и информация добавлена |
| **Фактический результат** |  |
| **Статус** | Зачёт |
| **Предварительное условие** | Перед проверкой теста нужно запустить программу и нажать на кнопку «Добавление» |
| **Постусловие** | После выполнения теста нажать на кнопку Сохранить. |
| **Примечания/комментарии** | Данный тест пройден успешно. |

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_ПИ\_2. Добавление комментария на сайте |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Добавление комментария |
| **Краткое изложение теста** | Должно производится добавление комментария |
| **Этапы теста** | Запуск сайта  Переход на страницу «Отзывы»  Нажатие на кнопку «Написать отзыв»  Ввод данных в текстовое поле  Для занесения данных нажать на кнопку «Сохранить» |
| **Тестовые данные** | Написание отзыва |
| **Ожидаемый результат** | Информация успешно сохранена и информация добавлена |
| **Фактический результат** |  |
| **Статус** | Зачёт |
| **Предварительное условие** | Перед проверкой теста нужно запустить программу и нажать на кнопку «Добавление» |
| **Постусловие** | После выполнения теста нажать на кнопку Сохранить. |
| **Примечания/комментарии** | Данный тест пройден успешно. |

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_ПИ\_3. Регистрация на сайте |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Регистрация |
| **Краткое изложение теста** | Пользователь должен попасть на страницу входа |
| **Этапы теста** | Запуск сайта  Переход на страницу «Регистрация»  Ввод данных в текстовое поле  Для занесения данных нажать на кнопку «Сохранить» |
| **Тестовые данные** | Имя пользователя – «admin», Email – «admin@admin.com», Пароль – «1234», Подтверждение пароля – «1234». |
| **Ожидаемый результат** | Пользователь зарегистрирован и попадает на страницу входа |
| **Фактический результат** |  |
| **Статус** | Зачёт |
| **Предварительное условие** | Перед проверкой теста нужно запустить программу и нажать на кнопку «Добавление» |
| **Постусловие** | После выполнения теста нажать на кнопку Сохранить. |
| **Примечания/комментарии** | Данный тест пройден успешно. |

## Тест-план программы

**1.Введение**

* 1. **Цель**

Целью составления данного Тест Плана является описание процесса тестирования сайта «Торговый дом «Бэст-мебель-трейд». Документ позволяет получить представление о плановых работах по тестированию проекта.

* 1. **Исходные данные**

«Торговый дом «Бэст-мебель-трейд» - сайт, позволяющий пользователю предназначен для ознакомления посетителей с компанией, возможностью предоставить покупателям возможность оставить заявку на нужную мебель прямо на сайте и при этом показать всю необходимую информацию о будущей покупке.

* 1. **Цели тестирования**

Целью тестирования сайта «Торговый дом «Бэст-мебель-трейд» является проверка корректной работы всех его функциональных возможностей на различных версиях браузеров с типовыми сценариями его использования. Часть времени (примерно 20%) будет использована для тестирования нетиповых/потенциально вызывающих ошибки работы сценариев использования.

Итогом процесса тестирования будут следующие материалы:

* заключение команды тестирования относительно общего состояния, дающее разработчикам и менеджерам данного продукта картину относительно корректности работы сайта в различных браузерах;
* отчет о результатах тестирования текущего покрытия типовые сценарии использования/браузеры;
* задокументированные баги в багтрекере заказчика.

Тестирование будет производиться вручную, методом «неформального» тестирования (ad-hoc testing) с позиции конечного пользователя приложения.

1. **Условия для тестирования**

Веб-сайт должен удовлетворять потребность пользователя в активностях, связанных с просмотром сайта, узнавать новости о скидках, пользоваться обратной связью.

1. **Стратегия процесса тестирования**

Приведенный ниже план тестирования является формальным, так как для построения развернутого плана необходимо понимание текущего состояния проекта. В результате первого прогона функциональных тестов в тест-план будут внесены изменения и улучшения. Первый прогон функциональных тестов даст нам четкое представление об уровне стабильности системы и будет четко определен набор тестов, которые будут выполнены в каждой конфигурации.

Такой подход даст возможность получить развернутый отчет по тестируемому продукту и сосредоточить максимальное внимание на узких местах.

Заказчику будут предоставляться ежедневные отчеты о ходе тестирования, найденных дефектах, предложениях по улучшению роботы продукта и его дизайна. Все обнаруженные дефекты будут занесены в виде отдельных тикетов для последующего исправления в багтрекер заказчика.

В процессе тестирования сайта «Торговый дом «Бэст-мебель-трейд» будет применено ad-hoc тестирование ввиду отсутствия строгой спецификации, а также ввиду ограниченности ресурсов на формализацию тестов.

Планируется пять этапов проведения процесса тестирования:

* первый этап заключается в анализе ТЗ, составлении тест плана, а также частичного прогона функциональных тестов;
* второй этап будет посвящен детальному прогону функциональных тестов с выявлением и описанием дефектов;
* на третьем этапе будет произведено тестирование кроссбраузерности с описанием найденных дефектов;
* четвертым этапом является проверка решенных разработчиками багов и проведение регрессионного тестирования;
* пятый этап заключается в тестировании дизайна продукта с описанием найденных дефектов.

Таким образом, достигается максимальная детализация глубины тестирования, что, в свою очередь, позволяет более точно определить затрачиваемые ресурсы, а также позволяет разработчикам проекта исправлять дефекты на самых ранних этапах.

ОС, утвержденные к проверке:

* Windows 10 и выше

Браузеры, утвержденные к проверке:

* Internet Explorer 8.0 и выше
* Firefox 3.5 и выше
* Opera 9.5 и выше
* Safari 3.2.1 и выше
* Chrome 2 и выше

Тестирование безопасности и стресс-тестирование не проводится по причине недостатка времени на тестирование.

1. Типы тестирования
2. **Функциональное тестирование**

Цель: Выявление функциональных ошибок, несоответствий ТЗ и ожиданиям пользователя путем реализации стандартных, а также нетривиальных тестовых сценариев.

Описание процесса:

Главная

* Просмотр товаров
* Фильтрация товаров
* Подробная информация о товаре

Регистрация/Авторизация

* Регистрация пользователя
* Авторизация пользователя
* Восстановление пароля

***Личный кабинет***

* Редактирование анкеты
* Выход пользователя из личного кабинета
* Добавление фотографии

Комментарии

* Корректное отображение поля комментария
* Проверка на валидность

Оставление заявки

* Ввод данных для заявки

Заявки

* Корректное отображение архивных и актуальных заявок

Написание отзыва

* Ввод текста к отзыву

1. **Тестирование кроссбраузерности**

Цель:Проверить корректную работу и дизайн проекта в различных браузерах

Браузеры

* Google Chrome 70.0.3538.102
* Opera 56.0.3051.104
* Mozilla Firefox 63.0.3

**3.1.3 Регрессионное тестирование и проверка решенных дефектов**

Цель: Проверка изменений, сделанных на сайте для того, чтобы убедиться, что новая версия программы не содержит ошибок в уже протестированных участках сайта.

В ходе регрессионного тестирования будут проведены такие виды тестов:

* Верификационные тесты
* Тестирование версии
* Тестирование смежного функционала

**3.1.4 Тестирование дизайна**

Цель: Проверка соответствия дизайна продукта макетам спецификации

Описание процесса:

* Регистрационная форма
* Личный кабинет
* Главная
* Заявки
* Отзывы
* Подробная информация о товаре
* Оставление заявки
* Написание отзыва

**3.План работ**

Таблица №13 «План работ»

|  |  |  |  |
| --- | --- | --- | --- |
| Задача | Объем работы | Дата начала | Дата  окончания |
| Составление тест плана | 12 часов | 12.04.2024 | 12.04.2024 |
| Выполнение тестирования | 35 часов | 13.04.2024 | 16.04.2024 |
| Анализ тестирования | 7 часов | 17.04.2024 | 17.04.2024 |
| Подведение итогов | 5 часов | 18.04.2024 | 18.04.2024 |

**4.Конечные результаты**

Конечным итогом проведения тестирования должен стать оформленный конечный результат процесса тестирования с описанными дефектами, а также рекомендациями по улучшению продукта с точки зрения конечного пользователя.

**Результаты тестирования**

Таблица №14 «Результаты тестирования»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Идентификатор | Описание | Шаги | Входные данные | Ожидаемые результаты | Фактические результаты | | Статус | |
| Avtor1 | Проверка входа с существующим логином и паролем | Откройте сайт-> Зайдите на страницу «Вход» ->Введите логин -> Введите пароль -> Нажмите кнопку «Войти» | Логин=admin  Пароль=1234 | Пользователь должен попасть на главную страницу | | Как ожидали | | Пройден успешно |
| Avtor2 | Проверка входа с несуществующим логином и паролем | Откройте сайт-> Зайдите на страницу «Вход» ->Введите логин -> Введите пароль -> Нажмите кнопку «Войти» | Логин=admin  Пароль=12345 | Пользователь останется на странице входа. Появится сообщение  «Пожалуйста, введите правильные имя пользователя и пароль. Оба поля могут быть чувствительны к регистру.» | | Как ожидали | | Пройден успешно |
| REG1 | Попытка регистрации с правильно введёнными данными | Откройте сайт-> Зайдите на страницу «Регистрация»  ->Введите имя пользователя -> Введите email ->  Введите пароль -> Нажмите кнопку «Регистрация» | Имя пользователя=Aksen  Почта=  [asken@yandex.com](mailto:asken@yandex.com)  Пароль=  QAZWSX2005 | Пользователь должен попасть на страницу входа | | Как ожидали | | Пройден успешно |
| REG2 | Попытка регистрации с неправильно введёнными данными | Откройте сайт-> Зайдите на страницу «Регистрация» ->Введите имя пользователя -> Введите email ->  Введите пароль -> Нажмите кнопку «Регистрация» | Имя пользователя=Aksen  Почта=  [asken@yandex.com](mailto:asken@yandex.com)  Пароль=  Aksen 2005 | Пользователь останется на странице регистрации и ему выведется сообщение «Введённый пароль слишком похож на имя пользователя.  Для подтверждения введите, пожалуйста, пароль ещё раз.» | | Как ожидали | | Пройден успешно |
| ZAV1 | Оставление заявки со всеми заполненными данными | Откройте сайт-> Зайдите на страницу «Вход» ->Введите логин -> Введите пароль -> Нажмите кнопку «Войти» ->Перейдите на товар, который вам понравился нажав на кнопку «Выбрать» ->Введите дату и описание->  Нажмите на кнопку «Отправить» | Дата=30.03.2024  Описание= Жду! | Пользователь был возвращён на главную страницу | | Как ожидали | | Пройден успешно |
| ZAV2 |  | Откройте сайт-> Зайдите на страницу «Вход» ->Введите логин -> Введите пароль -> Нажмите кнопку «Войти» ->Перейдите на товар, который вам понравился нажав на кнопку «Выбрать» ->Введите дату и описание->  Нажмите на кнопку «Отправить» | Дата=30.03.2024  Описание= «» | Пользователь остаётся на этой странице и ему выводится сообщение «Заполните это поле» | | Как ожидали | | Пройден успешно |

**Рекомендации для корректировки тестируемой программы.**

1. Реализовать регистрацию и авторизацию пользователей для повышения безопасности сайта.
2. Добавить профиль и редактор профиля
3. Добавить корзину и возможность купить товар прям на сайте

# Обеспечения качества программного обеспечения

**Основные теоретические сведения**

Все программы по характеру использования и категориям пользователей можно разделить на два класса - *утилитарные программы* и *программные продукты* *(изделия).* *Утилитарные программы* («программы для себя»)предназначены для удовлетворения нужд их разработчиков. Чаще всего утилитарные программы выполняют роль сервиса в технологии обработки данных либо являются программами решения функциональных задач, не предназначенных для широкого распространения.

*Программные продукты (изделия)* предназначены для удовлетворения потребностейпользователей, широкого распространения и продажи.

Существуют и другие варианты легального распространения программных продуктов:

* freeware – бесплатные программы, свободно распространяемые, поддерживаются самим пользователем, который правомочен вносить в них необходимые изменения;
* shareware – некоммерческие (условно-бесплатные) программы, которые могут использоваться, как правило, бесплатно.

Ряд производителей использует ОЕМ - программы (Original Equipment Manufacturer), т.е. встроенные программы, устанавливаемые на компьютеры или поставляемые вместе с вычислительной техникой.

Программные продукты (ПП) могут создаваться как:

* индивидуальная разработка под заказ;
* разработка для массового распространения среди пользователей.

Основными характеристиками программ являются:

* алгоритмическая сложность (логика алгоритмов обработки информации);
* состав и глубина проработки реализованных функций обработки;
* полнота и системность функций обработки;
* объём файлов программ;
* требования к операционной системе и техническим средствам обработки со стороны программного средства;
* объём дисковой памяти;
* размер оперативной памяти для запуска программ;
* тип процессора;
* версия операционной системы;
* наличие вычислительной сети и др.

Программные продукты имеют многообразие показателей качества, которые отражают различные аспекты.

Основная характеристика программного продукта – это его общая полезность, которая включает в себя мобильность, исходную полезность и удобство эксплуатации.

Мобильность ПП означает их независимость от технического комплекса системы обработки данных, операционной среды, сетевой технологии обработки данных, специфики предметной области и т.п. Мобильный (многоплатфорный) программный продукт может быть установлен на различных моделях компьютеров и операционных систем, без ограничений на его эксплуатацию в условиях вычислительной сети. Функции обработки такого программного продукта для массового использования без каких-либо изменений.

Исходная полезность характеризуется следующими показателями:

* надежность;
* эффективность;
* учет человеческого фактора;

*Надежность работы ПП* определяется бессбойностью и устойчивостью в работепрограмм, точностью выполнения предписанных функций обработки, возможностью диагностики возникающих в процессе работы программ ошибок.

*Эффективность ПП* оценивается как с позиций прямого его назначения–требований пользователя, так и сточки зрения расхода вычислительных ресурсов, необходимых для его эксплуатации. Расход вычислительных ресурсов оценивается через объем внешней памяти для размещения программ и объём оперативной памяти для запуска программ.

Учёт человеческого фактора означает обеспечение дружественного интерфейса для работы конечного пользователя, наличие котекстно- зависимой подсказки или обучающей системы в составе программного средства, хорошей документации для освоения и использования, заложенных в программном средстве функциональных возможностей, анализ и диагностику возникших ошибок и др.

Удобство эксплуатации включает следующие показатели качества:

* модифицируемость;
* коммуникативность.

*Модифицируемость ПП* означает способность к внесению изменений,напримеррасширение функций обработки, переход на другую техническую базу обработки и т.п.

*Коммуникативность ПП* основана на максимально возможной их интеграции сдругими программами, обеспечении обмене данными в общих форматах представления (экспорт/импорт баз данных, внедрение или связывание объектов обработки и др.).

Естественно, что в условиях существования рынка программных продуктов важными характеристиками являются: стоимость; количество продаж; длительность продаж (время нахождения на рынке); известность фирмы-разработчика и программы; наличие программных продуктов аналогического назначения.

Для оценки качества программного средства (ПС) используются различные способы получения информации о нём:

* измерительный – основан на получении информации о свойствах и характеристиках ПС с использованием инструментальных средств (например, объём ПС);
* регистрационный – получение информации во время испытаний или функционирования ПС, когда регистрируется и подсчитываются определённые события (число сбоев и отказов и др.);
* органолептический – анализ восприятия органов чувств, служащий для определения таких показателей как удобство применения ПС, его эффективность и др.;
* расчётный – на ранних этапах разработки применятся теоретические и эмпирические зависимости, статистические данные, накапливаемые при испытаниях, эксплуатации и сопровождении ПС;
* экспертный - рекомендован к применению при определении показателей наглядности, полноты и доступности программной документации, легкости освоения, структурности;
* социологические – основаны на обработке специальных анкет-вопросников.

Качество ПС определяется путём сравнения полученных расчётных значений показателей

* соответствующими базовыми значениями показателей существующего аналога или расчётного ПС, принимаемого за эталонный образец.

Значения базовых показателей ПС должны соответствовать значениям показателей, отражающих современный уровень качества и прогнозируемый мирровый уровень. В качестве аналогов выбираются реально существующие ПС того же функционального значения, что и сравниваемое, с такими же основными параметрами, подобной структуры и применяемые в тех же условиях эксплуатации.

Для оценки характеристик качества должны быть установлены метрики показателей. Для простоты расчета комплексных показателей и интегральной оценки качества, все метрики должны иметь одну и ту же область значений, соответствующую выбранной шкале: [0,1÷1] или [0,1÷10].

В метриках применяют различные методы определения значений показателей: измерительный, регистрационный, органолептический, расчетный, экспертный, социологический, а также их сочетания по установленным правилам. При определении метрик следует руководствоваться принципами реализуемости, объективности и точности оценки метрик.

**Измерительный** метод основан на получении информации с использованием инструментальных средств.

**Регистрационный** метод основан на получении информации во время испытаний или функционирования программного обеспечения, когда регистрируют или подсчитывают определенные события (время и число сбоев или отказов, время передачи управления другим модулям, время начала и окончания работы).

**Органолептический** метод основан на использовании информации, полученной в результате анализа восприятия органов чувств (зрения, слуха) для определения показателей удобства применения.

**Расчетный** метод основан на использовании теоретических и эмпирических зависимостей (на ранних стадиях разработки), статистических данных, накапливаемых при испытаниях, эксплуатации и сопровождении программного обеспечения. При помощи расчетного метода определяют длительность вычислений, время реакции, показатели надежности, необходимые ресурсы.

**Экспертный** метод основан на определении значений показателей качества ПО экспертами, компетентными в решении данной задачи, на базе их опыта и интуиции.

Экспертный метод применяют в тех случаях, когда задача не может быть решена никаким другим из существующих способов или другие методы являются значительно более трудоемкими.

**Социологические** методы основаны на обработке специальных анкет -вопросников.

В таблице 2 представлены общие контрольные вопросы для испытаний и оценки качества программного обеспечения.

***ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И ФОРМА ОТЧЕТНОСТИ:***

**Задание.** Сравнить программный продукт, разработанный студентами в соответствии с показателями качества. Сравнение проводить по следующим оценочным элементам: надежность ПС, сопровождаемость, корректность. Критерии оценки (0 до 1). Все сравнение занести в следующую таблицу.

Таблица №15 «Оценка качества»

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Метод оценки** | **Оценка программного продукта** |
| ***Оценочные элементы фактора «Надежность ПС»*** | | |
| Наличие требований к программе по устойчивости функционирования при наличии ошибок во входных данных | Экспертный | 1 |
| Возможность обработки ошибочных ситуаций | Регистрационный | 1 |
| Полнота обработки ошибочных данных | Экспертный | 1 |
| Наличие тестов для проверки допустимых значений входных данных | Экспертный | 1 |
| Наличие системы контроля полноты входных данных | Экспертный | 1 |
| Наличие средств контроля корректности входных данных | Экспертный | 1 |
| Наличие требований к программе по восстановлению процесса выполнения в случае сбоя ОС, внешних устройств, процессора | Экспертный | 0 |
| Наличие требований к программе по восстановлению результатов при отказах ОС, внешних устройств, процессора | Экспертный | 0 |
| Наличие средств восстановления при сбоях оборудования | Экспертный | 0 |
| Наличие возможности повторного старта с точки прерывания | Экспертный | 0 |
| Наличие обработки неопределенностей | Экспертный | 0 |
| Наличие возможности автоматически обходить ошибочные ситуации в процессе вычисления | Экспертный | 0 |
|  | Всего: | 6 |
| ***Оценочные элементы фактора «Сопровождаемость»*** | | |
| Наличие комментариев в точках входа и выхода в программу | Экспертный | 0 |
| Осуществляется ли передача результатов работы модуля через вызывающий его модуль | Экспертный | 0 |
| Оценка программы по числу циклов Используется ли язык высокого уровня | Экспертный | 0 |
| Наличие проверки корректности передаваемых данных | Экспертный | 1 |
| Использование при построении программ метода структурного программирования | Экспертный | 1 |
| Соблюдение принципа процесса разработки программы сверху вниз | Экспертный | 1 |
| Наличие ограничений на размеры модуля |  | 0 |
| Наличие модульной схемы программы |  | 0 |
|  | Всего: | 3 |
| ***Оценочные элементы фактора «Корректность»*** | | |
| Наличие всех необходимых документов для понимания и использования ПС | Экспертный | 1 |
| Наличие описание схемы иерархии модулей программы | Экспертный | 0 |
| Наличие описаний основных функций | Экспертный | 1 |
| Наличие описаний частных функций | Экспертный | 0 |
| Наличие описания данных | Экспертный | 1 |
| Наличие описания алгоритмов | Экспертный | 1 |
| Наличие описания интерфейсов между модулями | Экспертный | 1 |
| Наличие описания всех параметров | Экспертный | 1 |
| Наличие описание методов настройки системы | Экспертный | 0 |
| Наличие описание способов проверки работоспособности программы | Экспертный | 1 |
| Реанимация всех модулей системы | Экспертный | 0 |
| Реанимация всех основных функций | Экспертный | 1 |
| Реанимация всех алгоритмов | Экспертный | 0 |
| Наличие определений всех данных: переменные, индексы, массивы и пр. | Экспертный | 0 |
| Наличие интерфейсов с пользователем | Экспертный | 1 |
| Отсутствие противоречий в выполнении основных функций | Экспертный | 1 |
| Отсутствие противоречий в выполнении частных функций | Экспертный | 0 |
| Отсутствие противоречий в выполнении алгоритмов | Экспертный | 0 |
| Правильность взаимосвязей | Экспертный | 0 |
| Правильность реализаций интерфейса с пользователем | Экспертный | 1 |
| Отсутствие противоречий в настройке системы | Экспертный | 1 |
| Комплектность документации в соответствии со стандартами | Экспертный | 1 |
|  | Всего: | 13 |

**Задание.** Методика оценки качественных показателей ПС основана на составлении метрики ПС.

В практической работе необходимо выполнить следующее:

1. Выбрать показатели качества и сформулировать их сущность. Каждый показатель должен быть существенным, т. е. должны быть ясны потенциальные выгоды его использования. Показатели представить в виде таблицы (таблица 1). Все расчеты и построение гистограммы выполнить в MS Excel. И вставить данные в отчет.

Таблица №16 «Расчёт качества»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Показатели качества** | **Сущность показателя** | **Экспертная оценка (вес) wi** | **Оценка, установленная экспериментом, ri** | **Качество показателя** |
| 1. Надежность | определяется бессбойностью и устойчивостью в работе программ, точностью выполнения предписанных функций обработки, возможностью диагностики возникающих в процессе работы программ ошибок. | 1 |  | 0,6 |
| 1.1. Последовательность проектирования |  | 0 | 0,5 | 0 |
| 1.2.  Правильность |  | 0,5 | 0,7 | 0,35 |
| 1.3. Устойчивость к ошибкам |  | 0,5 | 0,5 | 0,25 |
| 2. Корректность | свойство безошибочной реализации требуемого алгоритма при отсутствии таких мешающих факторов, как ошибки входных данных, ошибки операторов ЭВМ (людей), сбои и отказы ЭВМ. | 1 |  | 0,65 |
| 2.1. Прослеживаемость |  | 0 | 0 | 0 |
| 2.2. Функциональная полнота |  | 0,5 | 0,7 | 0,35 |
| 2.3. Последовательность проектирования |  | 0,5 | 0,6 | 0,3 |
| 3. Эффективность | способность ПО обеспечивать требуемый уровень производительности в соответствии с выделенными ресурсами, временем и другими условиями. | 1 | 1 | 1 |
| 4. Гибкость | устойчивость к его изменениям. Это означает, что добавление новой или изменение старой функциональность в гибкой системе требует меньше усилий, чем в негибкой. | 1 |  | 0,75 |
| 4.1. Расширяемость |  | 0,5 | 0,5 | 0,25 |
| 4.2. Широта использования |  | 0,5 | 1 | 0,5 |
| 5. Функциональность | способность ПО решать задачи, которые соответсвуют потребностям пользователя при заданных условиях оспользования ПО | 1 |  | 0,85 |
| 5.1. Защищённость |  | 0 | 0 | 0 |
| 5.2. Согласованность |  | 0,1 | 0,5 | 0,05 |
| 5.3. Соответствие стандартам |  | 0,5 | 1 | 0,5 |
| 5.4. Функциональная пригодность |  | 0,3 | 0,7 | 0,21 |
| 5.5. Способность к взаимодействию |  | 0,1 | 0,9 | 0,09 |
| 6. Эргономичность проектирования | основана на максимально возможной их интеграции с другими программами, обеспечении обмене данными в общих форматах представления (экспорт/импорт баз данных, внедрение или связывание объектов обработки и др.). | 1 |  | 0,85 |
| 6.1. Ошибкоустойчивость/Целостность |  | 0,3 | 0,5 | 0,15 |
| 6.2. Доступность |  | 0,5 | 1 | 0,5 |
| 6.3. Коммуникативность |  | 0,2 | 1 | 0,2 |
| 7. Целостность | Состояние программного обеспечения и данных, характеризующееся отсутствием изменений преднамеренного или случайного характера. | 1 |  | 0,65 |
| 7.1. Управление доступом | 0,5 | 0,5 | 0,25 |
| 7.2. Контроль за доступом | 0,5 | 0,8 | 0,4 |
| 8. Функциональная совместимость | способность продукта или системы, интерфейсы которых полностью открыты, взаимодействовать и функционировать с другими продуктами или системами без каких-либо ограничений доступа и реализации. | 0,9 |  | 0,48 |
| 8.1. Модульность |  | 0 | 0 | 0 |
| 8.2. Унификация интерфейсов |  | 0,6 | 0,5 | 0,3 |
| 8.3. Унификация данных |  | 0,3 | 0,6 | 0,18 |
| 9.Сопровождаемость | характеристики программного продукта, позволяющие минимизировать усилия по внесению в него изменений: | 1 |  | 0,84 |
| 9.1. Простота работы |  | 0,4 | 1 | 0,4 |
| 9.2. Краткость |  | 0,3 | 1 | 0,3 |
| 9.3. Информативность |  | 0,2 | 0,7 | 0,14 |
| 9.4. Модульность |  | 0,1 | 0 | 0 |
| 10. Модифицируемость | факторы качества, важные для разработки новой версии ПО; | 0,5 |  | 0,75 |
| 10.1. Структурированность |  | 0,5 | 1 | 0,5 |
| 10.2. Дополняемость |  | 0 | 0,5 | 0 |
| 11. Мобильность | перенос По с одного окружения в другое. | 1 |  | 0,75 |
| 11.1. Простота работы |  | 0,5 | 1 | 0,5 |
| 11.2. Независимость программной платформы |  | 0 | 0 | 0 |
| 11.3. Независимость от аппаратной платформы |  | 0 | 0 | 0 |
| 11.4. Унификация данных |  | 0,5 | 0,5 | 0,25 |
| 12. Возможность многократного использования | Программное средство, разработанное для конкретного применения, но с возможностью другого применения, или разработанное специально для многократного использования в различных проектах или для многофункционального использования в одном проекте. | 1 |  | 1 |
| 12.1. Простота работы |  | 1 | 1 | 1 |
| 12.2. Широта использования |  | 0 | 1 | 0 |
| 12.3. Независимость программной платформы |  | 0 | 0 | 0 |
| 12.4. Независимость от аппаратной платформы |  | 0 | 0 | 0 |
| 13. Устойчивость | это способность программной системы должным образом реагировать на исключительные ситуации. Обработка исключительных ситуаций - процесс, направленный на достижение устойчивости. | 1 | 0,8 | 0,8 |
| 14. Тестируемость | это степень, в которой программный артефакт (т. е. программная система, программный модуль, требования или проектный документ) поддерживает тестирование в данном контексте тестирования. | 1 |  | 0,85 |
| 14.1. Простота работы |  | 0,3 | 1 | 0,3 |
| 14.2. Полнота протоколирования |  | 0,2 | 0,5 | 0,1 |
| 14.3. Информативность |  | 0,3 | 1 | 0,3 |
| 14.4. Расширяемость |  | 0,1 | 0,5 | 0,05 |
| 14.5. Модульность |  | 0,1 | 1 | 0,1 |
| 15. Понятность | это свойство, характеризующее, насколько легко (или трудно) специалисту, сопровождающему программный продукт, понять его работу. | 1 |  | 0,7 |
| 15.1. Последовательность проектирования |  | 0 | 0 | 0 |
| 15.2. Информативность |  | 0,2 | 1 | 0,2 |
| 15.3. Структурированность |  | 0,1 | 0,7 | 0,07 |
| 15.4. Краткость |  | 0,5 | 0,5 | 0,25 |
| 15.5. Удобочитаемость |  | 0,2 | 0,9 | 0,18 |
| 16. Практичность | способность программного продукта быть понятным, изученным, использованным и привлекательным для пользователя при применении в заданных условиях | 1 |  | 1 |
| 16.1. Удобство работы |  | 0,3 | 1 | 0,3 |
| 16.2. Удобство обучения |  | 0,5 | 1 | 0,5 |
| 16.3. Способность к взаимодействию |  | 0,2 | 1 | 0,2 |
| 17. Производительность | это мера того, насколько эффективно приложение использует ресурсы системы для выполнения действий, которые вы создали для этого. | 1 | 1 | 1 |
| 18. Зрелость процесса | это инструмент, помогающий организации в оценке и определении степени зрелости ее *процессов внедрения*. | 1 | 0,5 | 0,5 |
| 19.Масштабируемость | это способность системы адаптироваться к расширению предъявляемых требований и возрастанию объемов решаемых задач. | 1 | 0,5 | 1,5 |
| 20.Безопасность | защита приложений, которую их создатели обеспечивают уже на этапе разработки и до того, как программа окажется в открытом доступе. | 0 | 0,7 | 0 |
| 21.Эксплуатационная пригодность | Способность сооружения, его отдельного узла, элемента, изделия или конструкции выполнять эксплуатационные функции, для которых объект проектировался и использовался. | 1 | 1 | 1 |
|  | Обобщённый показатель качества: |  |  | 74% |
|  |  |  |  |  |
| Вывод: в целом программа считается качественной, 74% качества. Для повышения качества необходимо повысить: безопасность - добавить авторизацию и регистрацию. | | | | | |
|
|

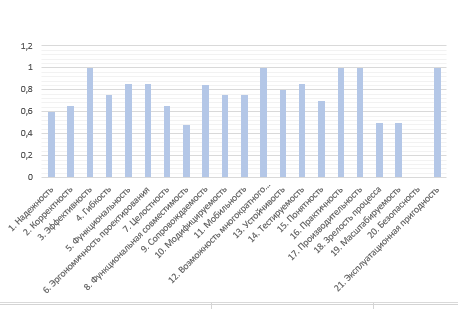


Рис.36 «Диаграмма показателей качества»



2. Установить веса показателей

3. Для каждого показателя установить конкретную численную оценку от 0 до 1, исходя из следующего: **ri**

0- свойство в ПС присутствует, но качество его неприемлемо;

0.5 - 1 - свойство в ПС присутствует и обладает приемлемым качеством;

1- свойство в ПС присутствует и обладает очень высоким качеством.

Возможно присвоение промежуточных значений в соответствии с мнением оценивающего лица относительно полезности того или иного свойства ПС.

4. Определить качество ПС как иерархическую взвешенную сумму весов отдельных показателей.

***Качество показателя =* *wi\** *ri***

5. Определить среднее значение оценки качества ПС по формуле (1).

6. Представить выходные данные:

- перечень всех показателей с оценкой 0 с указанием причин такой оценки;

- вывод по Обобщённому показателю качества;

- гистограмму, показывающую распределение показателей по интервалам оценок.

# Расчет надежности работы сайта

**Надежность** – свойство программного средства сохранять работоспособность в течение определенного периода времени, в определенных условиях эксплуатации с учетом последствий для пользователя каждого отказа.

**Работоспособным** называется такое состояние программного средства, при котором оно способно выполнять заданные функции с параметрами, установленными требованиями технического задания. С переходом в неработоспособное состояние связано событие отказа.

**Причиной отказа программного средства** является невозможность его полной проверки в процессе тестирования и испытаний. При эксплуатации программного средства в реальных условиях может возникнуть такая комбинация входных данных, которая вызовет отказ, следовательно, работоспособность программного средства зависит от входных данных, и чем меньше эта зависимость, тем выше уровень надежности.

Для оценки надежности используются три группы показателей: качественные, порядковые и количественные.

К основным количественным показателям надежности программного средства относятся:

* Вероятность безотказной работы– это вероятность того, что в пределах заданной наработки отказ системы не возникает. Наработка – продолжительность или объем работ.
* Вероятность отказа – вероятность того, что в пределах заданной наработки отказ системы возникает. Этот показатель, обратный предыдущему.
* Интенсивность отказов системы – это условная плотность вероятности возникновения отказа программного средства в определенный момент времени при условии, что до этого времени отказ не возник.
* Средняя наработка до отказа – математическое ожидание времени работы программного средства до очередного отказа.
* Среднее время восстановления – математическое ожидание времени восстановления.
* Коэффициент готовности – вероятность того, что программное средство ожидается в работоспособном состоянии в произвольный момент времени его использования по назначению.

**Причиной отказа программного средства** являются ошибки, которые могут быть вызваны: внутренним свойством программного средства, реакцией программного средства на изменение внешней среды функционирования. Это значит, что при самом тщательном тестировании, если предположить, что удалось избавится от всех внутренних ошибок, нельзя с полной уверенность утверждать, что в процессе эксплуатации программного средства не возникнет отказ.

Основным средством определения количественных показателей надежности являются **модели надежности**, под которыми понимают математическую модель, построенную для оценки зависимости надежности от заранее известных или оцененных в ходе создания программного средства параметров. В связи с этим определение надежности показателей принято рассматривать в единстве трех процессов – предсказание, измерение, оценивание.

**Предсказание** – это определение количественных показателей надежности исходя из характеристик будущего программного средства.

**Измерение** – это определение количественных показателей надежности, основанное на анализе данных об интервалах между отказами, полученных при выполнении программ в условиях тестовых испытаний.

**Оценивание** - это определение количественных показателей надежности, основанное на данных об интервалах между отказами, полученными при испытании программного средства в реальных условиях функционирования.

Все модели надежности можно классифицировать по тому, какой из перечисленных процессов они поддерживают (предсказывающие, прогнозные, оценивающие, измеряющие) Нужно отметить, что модели надежности, которые в качестве исходной информации используют данные об интервалах между отказами, можно отнести к измеряющим, и к оценивающим в равной степени. Некоторые модели, основанные на информации, полученной в ходе тестирования программного средства дают возможность делать прогнозы поведения программного средства в процессе эксплуатации.

Аналитические модели дают возможность рассчитать количественные показатели надежности, основываясь на данных о поведении программы в процессе тестирования (измеряющие и оценивающие модели). Эмпирические модели базируются на анализе структурных особенностей программ. Они рассматривают зависимость показателей надежности от числа межмодульных связей, количества циклов в модулях, отношения количества прямолинейных участков к количеству точек ветвления и тому подобное. Нужно отметить, что часто эмпирические модели не дают конечных результатов показателей надежности.

Аналитическое моделирование надежности программного средства включает четыре шага:

􀂾 определение предложений, связанных с процедурой тестирования программного средства;

􀂾 разработка или выбор аналитической модели, базирующейся на предположениях о процедуре тестирования;

􀂾 выбор параметров моделей с использование полученных данных;

􀂾 применение модели – расчет количественных показателей надежности по модели.

Аналитические модели представлены двумя группами: динамические и статические модели. В динамических моделях надежности программного средства поведение программы (появление отказов) рассматривается во времени. В статических моделях появление отказов не связывают со временем, а учитывают только зависимость количества ошибок от числа тестовых прогонов (по области ошибок) или зависимость количества ошибок от характеристики входных данных (по области данных). Для использования динамических моделей необходимо иметь данные о появлении отказов во времени. Статические модели принципиально отличаются от динамических тем, что в них не учитывается время появления ошибок в процессе тестирования и не используется никаких предположений о поведении функции риска. Эти модели строятся на твердом статистическом фундаменте.

***ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И ФОРМА ОТЧЕТНОСТИ:***

**Задание. Выполните оценку показателей надежности информационной системы, используя средства электронной таблицы MS Excel.**

1. **Расчет надежности с помощью модели Коркорэна**

В этой модели не используются параметры времени тестирования и учитывается только результат N испытаний, в которых выявлено Ni ошибок i-го типа. Модель использует изменяющиеся вероятности отказов для различных типов ошибок.

**Условие:** При тестировании программы было проведено 100 испытаний, при которых было выявлено следующее число ошибок по типам. Из 100 испытаний 30 было проведено безуспешно. Необходимо определить надёжность программы.

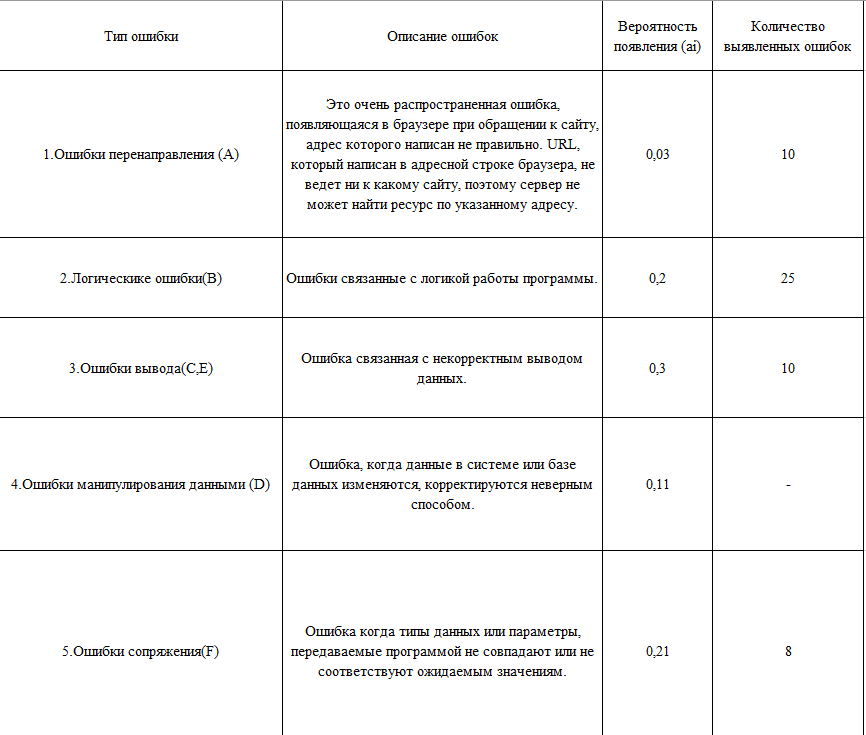




Рис.38«Таблица надёжности»



Рис.39«Вывод надёжности»



Рис.40«Оценка надежности»

1. **Расчет надежности с помощью Простой интуитивной модели**

Предполагается проводить тестирование двумя независимыми группами лиц, использующими независимые тесты. Программа некоторое время тестируется параллельно, затем результаты сравниваются.

- ошибки, обнаруженные первой группой

- ошибки, обнаруженные второй группой

- ошибки обнаруженные обеими группами

- предполагаемое количество ошибок в программе

**Условие:** При проведении тестирования первой группой было обнаружено 17 ошибок, второй – 20 ошибок. Общих ошибок - 7. Определить надежность по простой интуитивной модели.

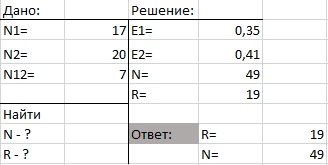


Рис.41«Задача по расчёту надёжности»

Надежность по простой интуитивной модели 49%

# Заключение

Учебную практику по ПМ.05 «Проектирование и разработка информационных систем» проходил в учебном учреждении Ликино-Дулевский политехнический колледж – филиал ГГТУ.

Во время учебной практики выполнялись следующие виды работ:

1. Разработка web-приложения (сайта);
2. Проектирование предметной области;
3. Роутинг в CMS Bitrix;
4. Разработка компонентов и структуры сайта в CMS Bitrix;
5. Разработка API для бэкэнда
6. Проведение тестирования и отладки web-приложения (сайта);
7. Разработка технической документации;
8. Оценка качественных показателей программного средства с помощью метрик;
9. Оценка надежности программного средства.

В соответствии с постановкой задачи был разработан сайт со следующими возможностями:

* Добавление, удаление и редактирование товаров
* Фильтрация товара по критериям: «Наименование», «Цена»
* Организация страницы с товарами и фильтрацией
* Организация страницы с отзывами
* Организация страницы «Об услуге» с подробным описанием услуги
* Организация страницы «Заявки» с информацией об актуальных и архивных заявках
* Организация страницы «Выбрать» с возможностью оставить заявку на нужный товар

На этапе проектирования разработана диаграмма прецедентов, диаграмма действий. В соответствии с постановкой задачи разработан дизайн, произведена верстка и разработка компонентов сайта.

Произведен расчет качественных показателей и надежности работы сайта. В соответствие с ГОСТами разработана техническая документация: техническое задание, руководство пользователя, руководство программиста, программа и методика испытаний.

Для дизайна и разработки сайта использовались программные средства:

ОС Windows 7,10,XP и выше, любой браузер, лучше FireFox или Yandex, JavaScript: JavaScript, HTML и CSS: HTML

# Список литературы

1. Дремина Е.Е. Разработка информационного контента: учебник для студентов СПО /Е.Е. Дремина. - Москва: Академия, 2020. - 249 с.- (Профессиональное образование).

2. Перлова О.Н. Проектирование и разработка информационных систем: учебник для студентов СПО /О.Н. Перлова, О.П. Ляпина, А.В. Гусева. - 3-е изд., испр. - Москва: Акакдемия, 2020. - 252 с. - (Профессиональное образование).

3. Черпаков, И. В. Основы программирования : учебник и практикум для среднего профессионального образования / И. В. Черпаков. — Москва : Издательство Юрайт, 2023. — 219 с. — (Профессиональное образование). — ISBN 978-5-9916-9984-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/513113

4. Меженин А.В. Проектирование, разработка и оптимизация веб-приложений: учебник для студентов СПО / А.В. Меженин, Д.А. Меженин. - 2-е изд.,стер. - Москва Академия,2022. - 270 с. -(Профессиональное образование).

5. Колошкина, И. Е. Компьютерная графика : учебник и практикум для среднего профессионального образования / И. Е. Колошкина, В. А. Селезнев, С. А. Дмитроченко. — 3-е изд., испр. и доп. — Москва : Издательство Юрайт, 2023. — 233 с. — (Профессиональное образование). — ISBN 978-5-534-15862-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/510043

6. Основы дизайна и композиции: современные концепции : учебное пособие для среднего профессионального образования / Е. Э. Павловская [и др.] ; ответственный редактор Е. Э. Павловская. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 119 с. — (Профессиональное образование). — ISBN 978-5-534-11671-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/517147

7. Полуэктова, Н. Р. Разработка веб-приложений : учебное пособие для среднего профессионального образования / Н. Р. Полуэктова. — Москва : Издательство Юрайт, 2023. — 204 с. — (Профессиональное образование). — ISBN 978-5-534-14744-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/519727>

8. Тузовский, А. Ф. Проектирование и разработка web-приложений: учебное пособие для среднего профессионального образования / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2023. — 219 с. — (Профессиональное образование). — ISBN 978-5-534-16767-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/531669

Интернет-ресурсы

1. ЭБС «Университетская библиотека online» - http://biblioclub.ru/

2. ЭБС «ЮРАЙТ» - https://urait.ru/