**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\_\_\_\_Степанова Дмитрия Данииловича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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 3](#_Toc164778186)

[1.1. Общие сведения. 3](#_Toc164778187)

[1.1.1. Что такое Django 3](#_Toc164778188)

[1.1.2. Установка и настройка. 3](#_Toc164778189)

[1.1.3. Основные команды Django 3](#_Toc164778190)

[1.2. Разработка компонентов 3](#_Toc164778191)

[1.2.1. Модели 3](#_Toc164778192)

[1.2.2. Представления 4](#_Toc164778193)

[1.2.3. Шаблоны 5](#_Toc164778194)

[1.2.4. Работа с формами 7](#_Toc164778195)

[1.3. Пользовательский интерфейс 8](#_Toc164778196)

[2. Роутинг в CMS Bitrix 10](#_Toc164778197)

[2.1. Установить CMS Bitrix 10](#_Toc164778198)

[2.2. Настройка роутинга 12](#_Toc164778199)

[2.3. Практическая часть 12](#_Toc164778200)

[2.3.1. Получить все активные товары определенного раздела; 12](#_Toc164778201)

[2.3.2. Получить все товары по фильтру; 13](#_Toc164778202)

[2.3.3. Получить все активные товары с ценой; 14](#_Toc164778203)

[3. Быстрый старт разработчика CMS Bitrix 15](#_Toc164778204)

[3.1. Установка системы 15](#_Toc164778205)

[3.2 Интеграция верстки 21](#_Toc164778206)

[3.3 Редактируемые области 23](#_Toc164778207)

[3.4 Базовая настройка системы 23](#_Toc164778208)

[3.5 Компоненты и структура сайта 26](#_Toc164778209)

[4. Разработка API для бэкэнда 34](#_Toc164778210)

[4.1. Разработка сервера 34](#_Toc164778211)

[4.2. Выполнение запросов к серверу 40](#_Toc164778212)

[4.2.1. Последние 5 записей 40](#_Toc164778213)

[4.2.2. Поиск 41](#_Toc164778214)

[4.2.3. Регистрация 42](#_Toc164778215)

[4.2.4. Комментарии 45](#_Toc164778216)

[4.2.5. Заказ товара 47](#_Toc164778217)

[5. Проектирование предметной области 49](#_Toc164778218)

[6. Разработка технической документации 51](#_Toc164778219)

[6.1. Техническое задание 51](#_Toc164778220)

[6.2. Руководство пользователя 53](#_Toc164778221)

[6.3. Руководство программиста 59](#_Toc164778222)

[6.4. Методика тестирования и испытания программы 62](#_Toc164778223)

[6.5. Тест-план программы 71](#_Toc164778224)

[7. Тестирование кроссбраузерности 75](#_Toc164778225)

[8. Расчет надежности работы сайта 94](#_Toc164778226)

[Заключение 100](#_Toc164778227)

[Список литературы 102](#_Toc164778228)

# 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. Сайт на Django строится из одного или нескольких приложений.

### 1.1.2. Установка и настройка.

Установка Django

Установка через Настройки: File-> Setting-> Python Interpreter-> “+” Django

Установка через pip: pip install django

### 1.1.3. Основные команды Django

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. Разработка компонентов

### 1.2.1. Модели

Создана 1 модель «Position» в ней находится имя, производитель и цена.

from django.db import models  
  
  
class Position(models.Model):  
 name = models.CharField('Название', max\_length=30)  
 type = models.CharField('Производитель', max\_length=40)  
 price = models.IntegerField('Цена')  
  
 def \_\_str\_\_(self):  
 return self.name  
 class Meta:  
 verbose\_name="Товар"  
 verbose\_name\_plural="Товар магазина"

### 1.2.2. Представления

from django.shortcuts import render, redirect  
from .models import Position  
from .forms import PosForm  
from django.views.generic import DeleteView, UpdateView  
from .filters import PostFilter

**Главная страница**

def index(request):  
 positions\_all = Position.objects.all()  
 positions = PostFilter(request.GET, queryset=positions\_all)  
 return render(request, 'main/index.html', {'title': 'Главная страница сайта', 'positions': positions})

**Страница О Нас**

def about(request):  
 return render(request, 'main/about.html')

**Страница Обслуживания**

def service(request):  
 return render(request, 'main/service.html')

**Страница Добавления**

def add(request):  
 error = ''  
 if request.method == 'POST':  
 form = PosForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('home')  
 else:  
 error = 'Форма заполнена неправильно или не до конца'  
 form = PosForm()  
 context = {  
 'form': form  
 }  
 return render(request, 'main/add.html', context)

**Кнопка Удаления**

class Delete(DeleteView):  
 model = Position  
 success\_url = '/'  
 template\_name = 'main/delete.html'

**Кнопка Редактирования**

class Update(UpdateView):  
 model = Position  
 template\_name = 'main/add.html'  
 form\_class = PosForm  
 success\_url = '/'

### 1.2.3. Шаблоны

Шаблон «add.html» там находится форма добавления товара с полями:

Название, Производитель и Цена.

{% extends 'main/base.html' %}  
  
{% block title %}Добавить товар{% endblock %}  
{% block tab-name %}Добавить товар{% endblock %}  
{% block content %}  
 <h1>Добавить товар</h1>  
 <form method="post" class="add-form">  
 {% csrf\_token%}  
 {{form.name}} <br>  
 {{form.type}} <br>  
 {{form.price}} <br>  
 <button type="submit">Добавить</button>  
 <span>{{error}}</span>  
 </form>  
{% endblock %}

Шаблон «Index.html» там находится блок фильтрации услуг и возможность их удалить или редактировать.

{% extends 'main/base.html' %}  
  
{% block title %}  
{{ title }}  
{% endblock %}  
{% block tab-name %}Главная{% endblock %}  
{% block content %}  
<h1>Каталог</h1>  
{% if positions %}  
 <form method="get" class="form-filter">  
 <div>  
 Название: {{ positions.form.name }}  
 Производитель: {{ positions.form.type }}  
 Цена: {{ positions.form.price }}  
 <button type="submit" class="btn btn-primary">Фильтровать</button>  
 </div>  
 </form>  
<table>  
 <tr>  
 <th>Название</th>  
 <th>Производитель</th>  
 <th>Цена</th>  
 <th>Управление записями</th>  
 </tr>  
 {% for el in positions.qs %}  
 <tr>  
 <td>{{el.name}}</td>  
 <td>{{el.type}}</td>  
 <td>{{el.price}} р.</td>  
 <td>  
 <div class="produtcs\_\_btns">  
 <a href="{% url 'delete' el.id %}">  
 <button class="btn btn-danger">Удалить</button>  
 </a>  
 <a href="{% url 'update' el.id %}">  
 <button class="btn btn-success">Редактировать</button>  
 </a>  
 </div>  
 </td>  
 </tr>  
 {% endfor %}  
</table>  
{% else %}  
<p>У нас нет позиций в каталоге</p>  
{%endif%}  
{% endblock %}

### 1.2.4. Работа с формами

Создана форма в которой находятся: Название, Производитель и цена.

from .models import Position  
from django.forms import ModelForm, TextInput, Textarea  
  
  
class PosForm(ModelForm):  
 class Meta:  
 model = Position  
 fields = ["name", "type", "price"]  
 widgets = {  
 "name": TextInput(attrs={'placeholder': 'Название', 'class': 'form-input'}),  
 "type": TextInput(attrs={'placeholder': 'Производитль', 'class': 'form-input'}),  
 "price": TextInput(attrs={'placeholder': 'Цена в руб.', 'class': 'form-input'}),  
 }

## 1.3. Пользовательский интерфейс

На странице «Главная» есть фильтрация и таблица с велосипедами, велосипеды можно редактировать и удалять.

На странице «О нас» располагается информация о фирме.

На странице «Обслуживание» находятся информация о том, как произвести ремонт в магазине компании.

На странице «Добавить услугу» находится форма с полями, где можно добавить услугу.

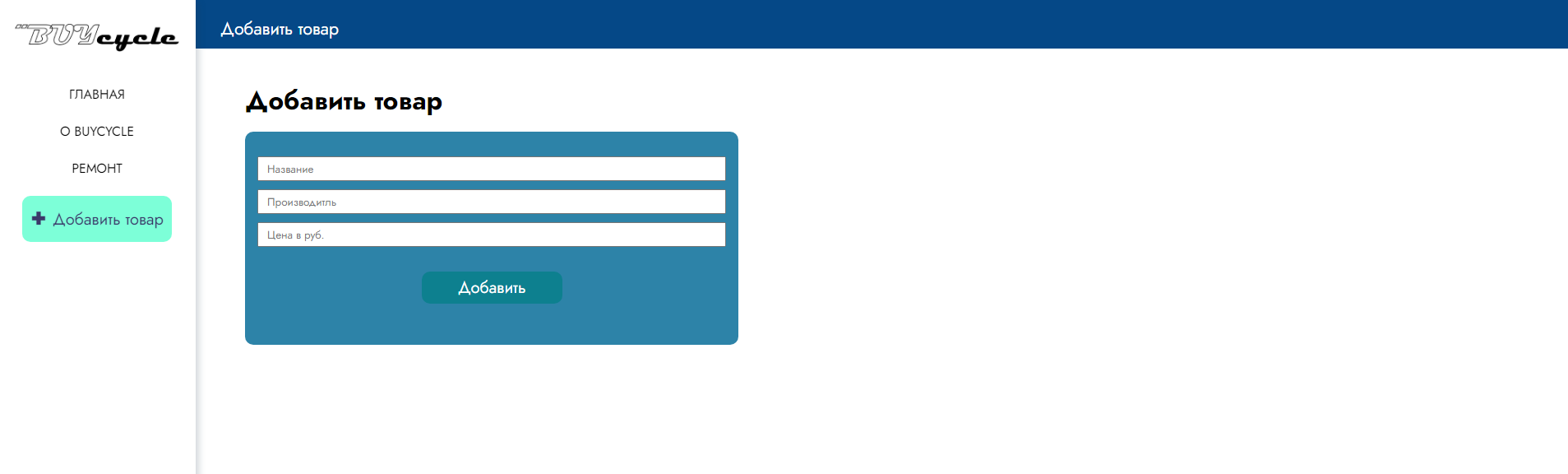


Рис. 1. «Добавление услуги»

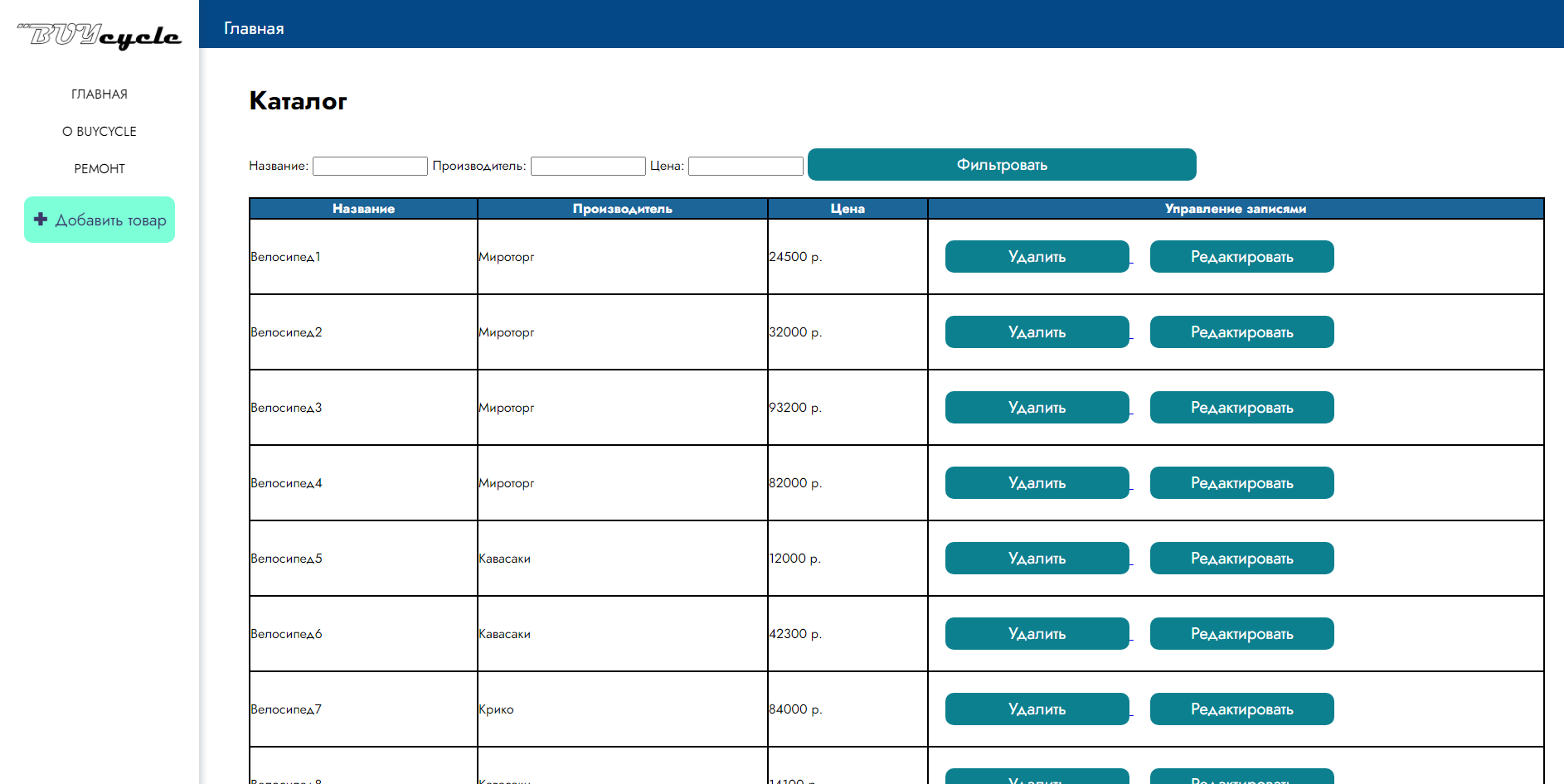


Рис. 2. «Главная страница»

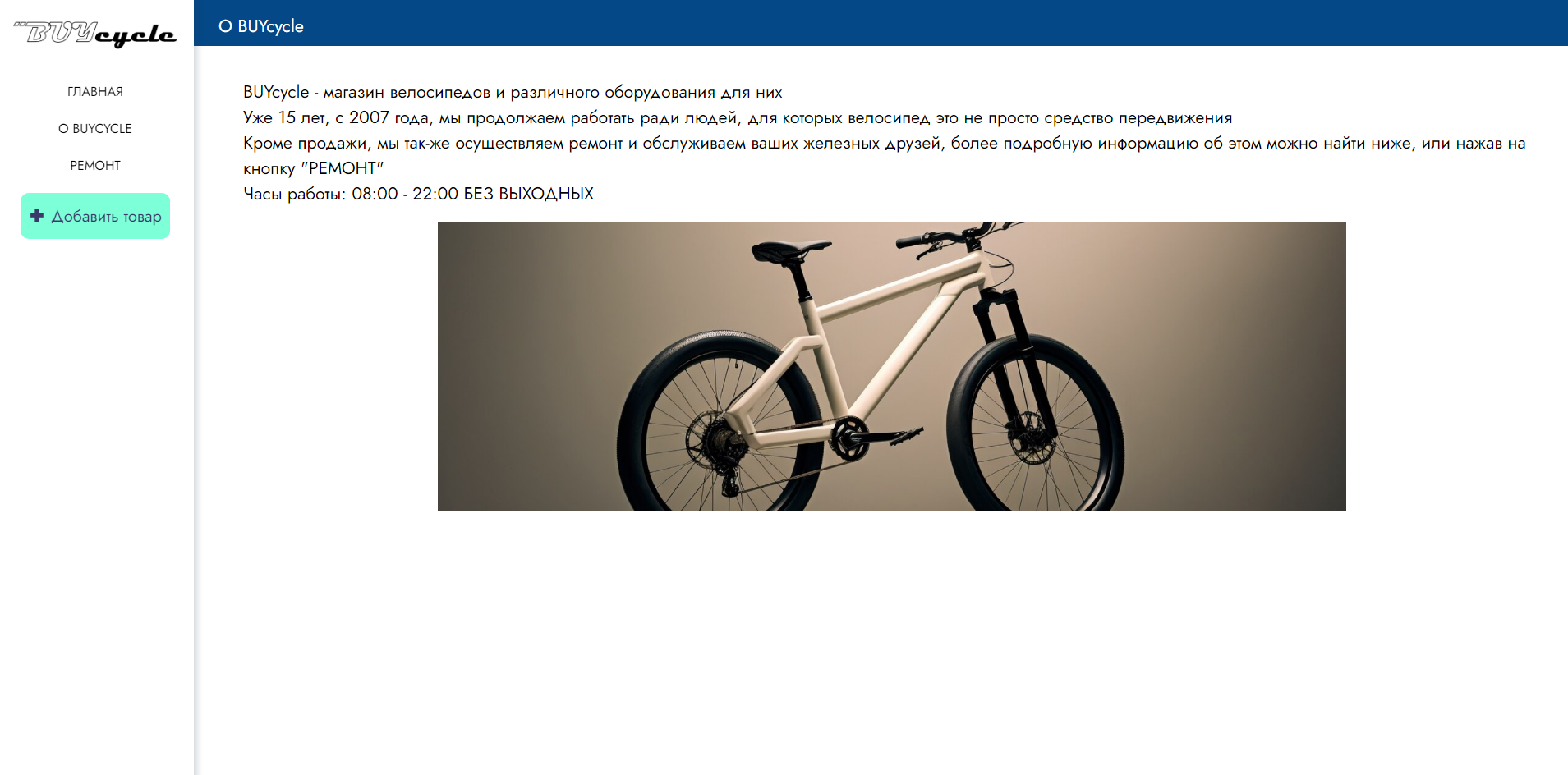


Рис. 3. «Страница О нас».

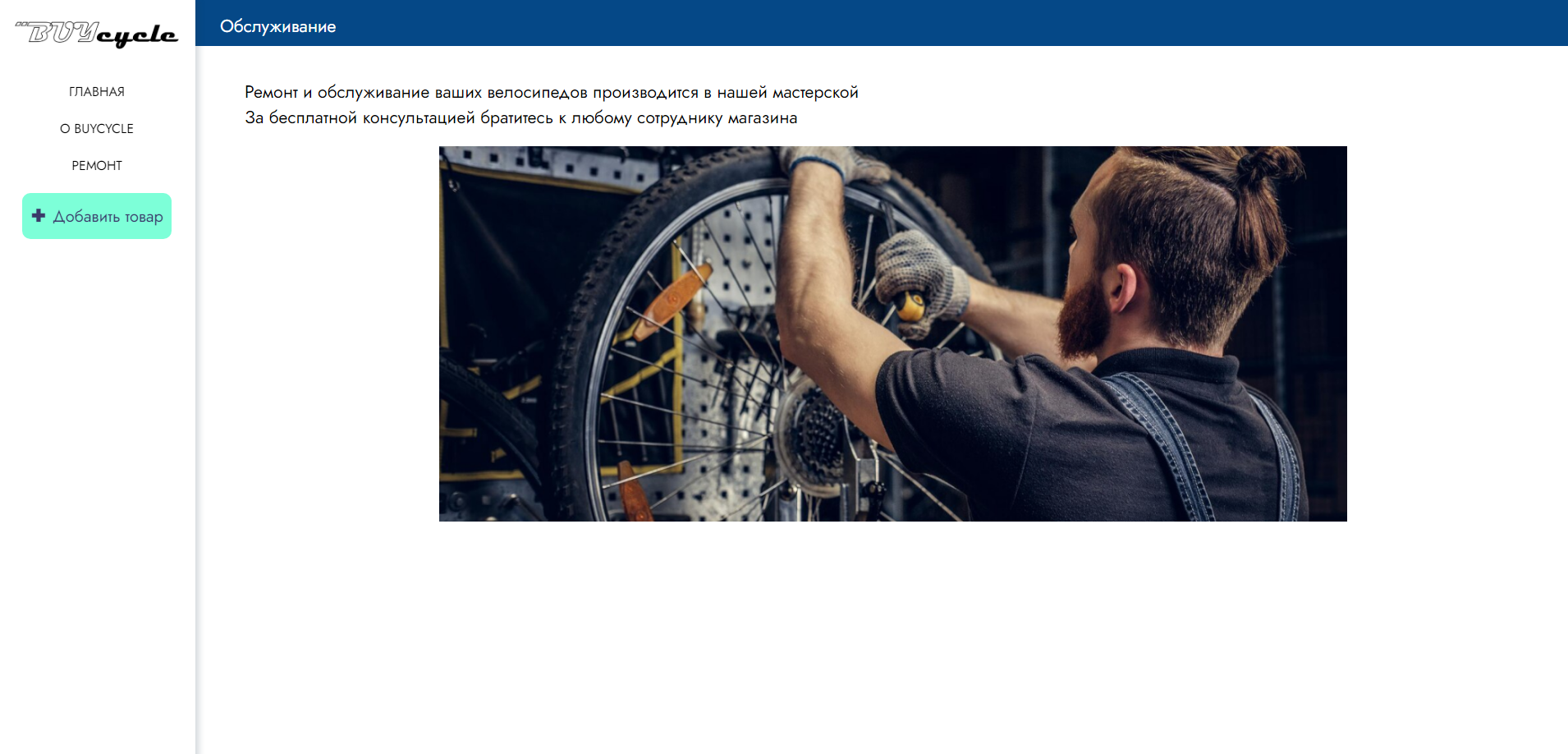


Рис. 4. «Страница обслуживания».

# 2. Роутинг в CMS Bitrix

## 2.1. Установить CMS Bitrix

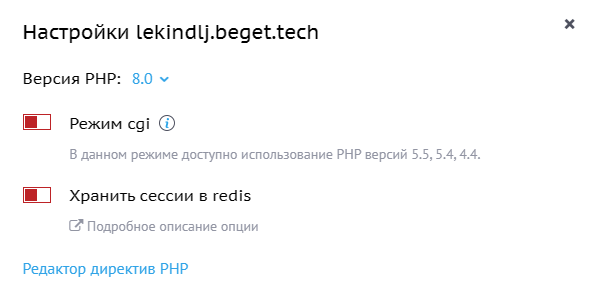


Рис. 5. «Настройка хоста».

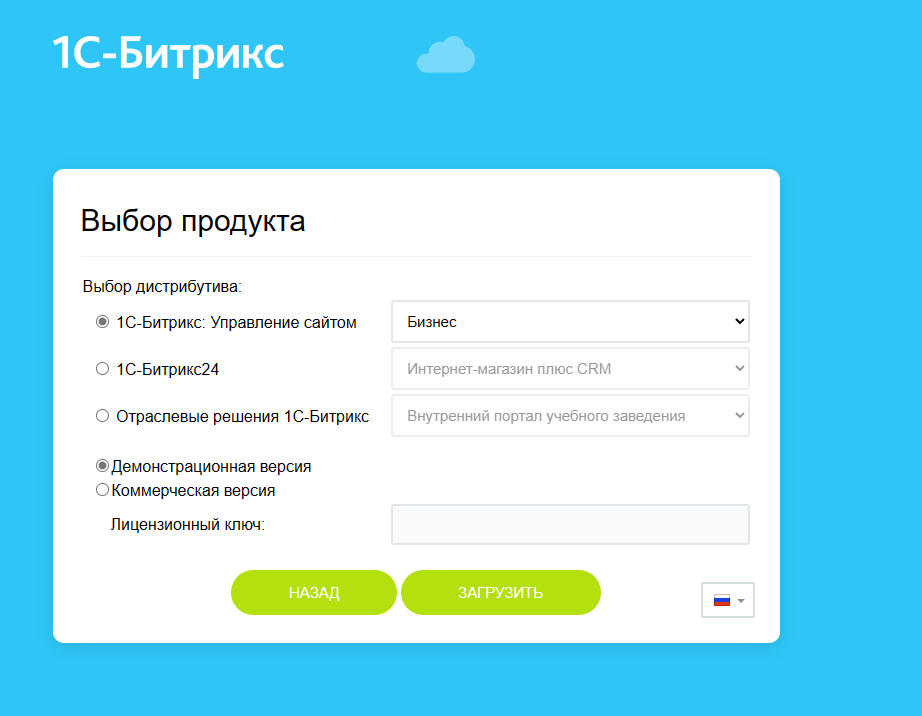


Рис. 6. «Выбор направления сайта».

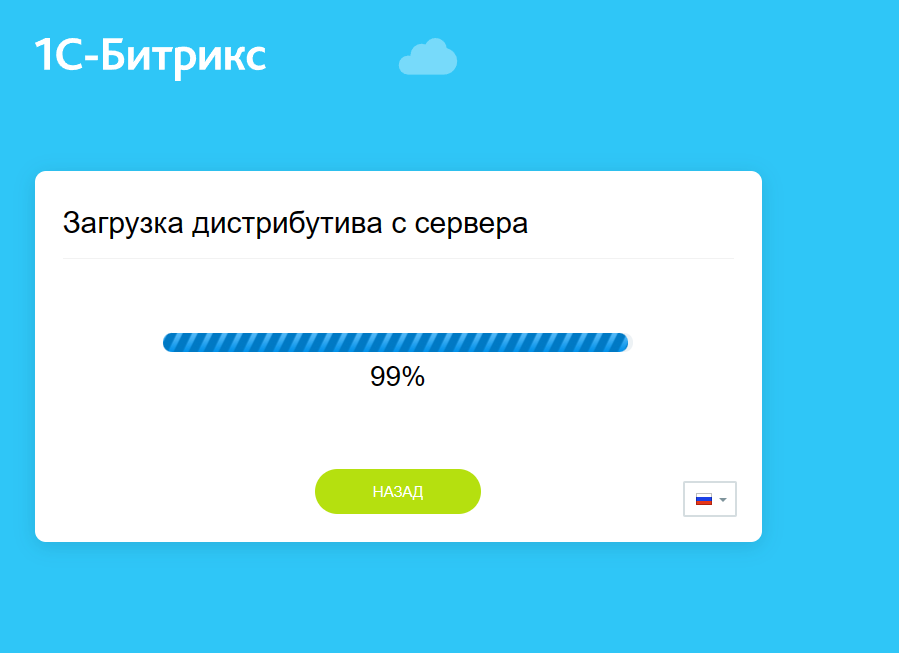
****

Рис. 7. «Загрузка дистрибутива сайта».

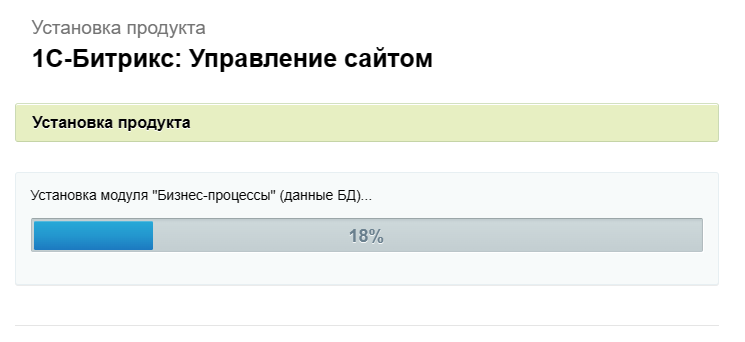
****

Рис. 8. «Загрузка продукта».

****

Рис. 9. «Настройка продукта».

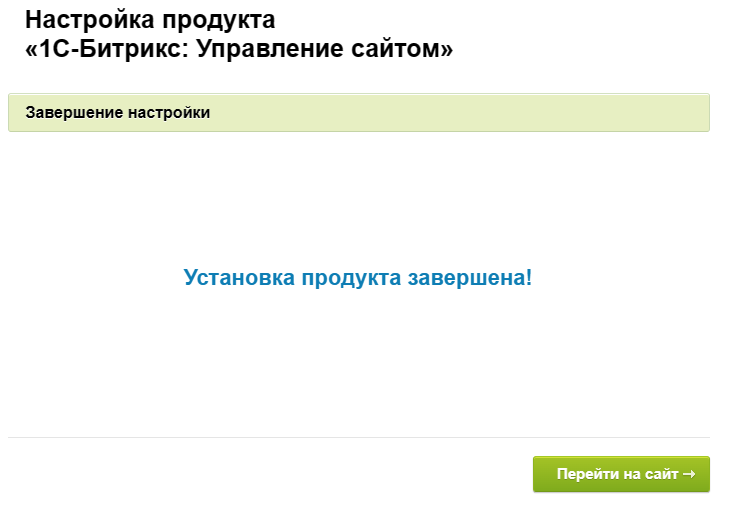
****

Рис. 10. «Настройка и загрузка завершены».

## 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

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

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

'filter' => [

'=ACTIVE' => 'Y',

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

],

])->fetch();

if(!$elements){

return json\_encode([

'status' => 'error',

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

]);

}

return json\_encode($elements);

});

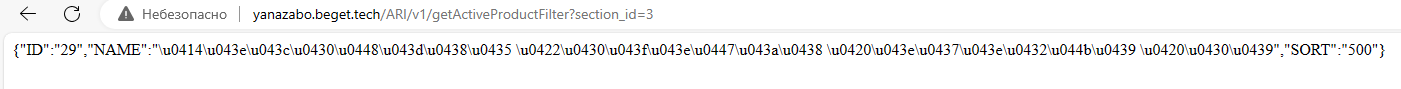


Рис. 11. «Вывод всех активных товаров определенного раздела».

### 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);

});

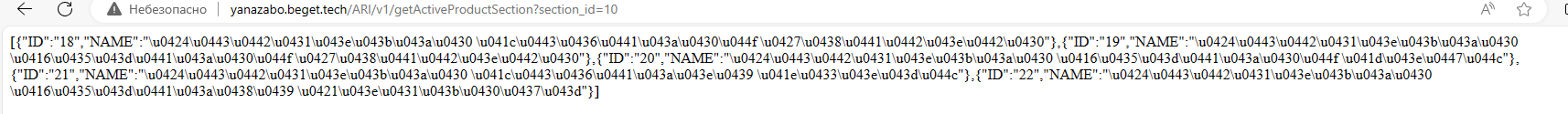


Рис. 12. «Вывод всех товаров по фильтру».

### 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);

});

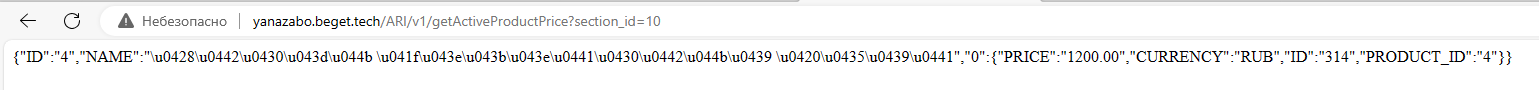


Рисунок 13 «Вывод всех активных товаров с ценой».

# 3. Быстрый старт разработчика CMS Bitrix

## 3.1. Установка системы

Таблица №1 «Процесс установки Bitrix»

|  |  |
| --- | --- |
| Шаг 1  Установка BitrixSetup |  |
| Шаг 2  Загружаем файл BitrixSetup в файловый менеджер |  |
| Шаг 3  Выбираем направление сайта |  |
| Шаг 4  Ждем окончания загрузки |  |
| Шаг 5  Начало установки нажимаем «Далее» |  |
| Шаг 6  Принимаем лицензионное соглашение и нажимаем «Далее» |  |
| Шаг 7  Вводим свои данные, выбираем установить UTF-8 и нажимаем «Далее» |  |
| Шаг 8  Пролистываем в самый конец и нажимаем «Далее» |  |
| Шаг 9  Вводим логин и пароль от базы данных и нажимаем «Далее» |  |
| Шаг 10  Ждем окончания загрузки продукта и нажимаем «Далее» |  |
| Шаг 11  Создаем администратора и нажимаем «Далее» |  |
| Шаг 12  Выбираем решение для установки «Загрузить из Marketplace» и нажимаем «Далее» |  |
| Шаг 13  Выбираем решение «Чистая установка» и нажимаем «Далее» |  |
| Шаг 14  Ждем окончания загрузки решения |  |
| Шаг 15  Даем название шаблону и нажимаем «Настройка решения» |  |
| Шаг 16  Даем название сайту и нажимаем «Настройка решения» |  |
| Шаг 17  Выбираем все кроме «Перевод» и «Социальные сервисы» и нажимаем «Настройка решения» |  |
| Шаг 18  Ждем окончания установки |  |
| Шаг 19  Конец установки |  |

## 3.2 Интеграция верстки

Интегрируем изначальный код header и footer

Header

$blsMainPage = $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>

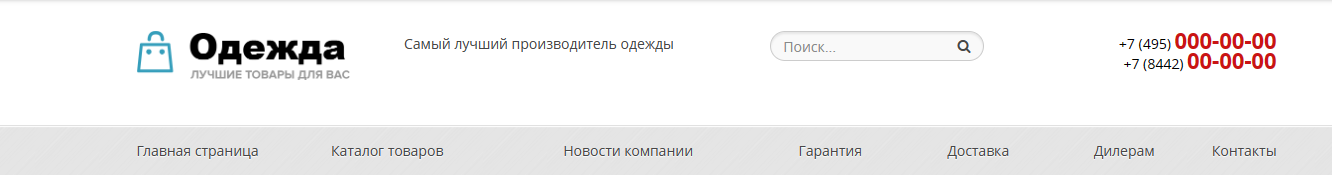


Рис. 14. «Header».

Footer

<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" => "html"

)

);?>

</address>

</div>



Рис. 15. «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, phone2, address.

## 3.4 Базовая настройка системы

Таблица №2 «Базовая настройка системы»

|  |  |
| --- | --- |
| Заходим в админку, следом в настройки и в Главный модуль, меняем название сайта под своё, переходим по ссылке «формы подключения AIP-ключа». |  |
| Нужно авторизоваться в Яндексе, чтобы продолжить. |  |
| Введите описание ключа |  |
| Вставьте полученный ключ и нажимаем «Применить» |  |
| Теперь заходим в Информационные блоки. Отмечаем первую и третью галочку и нажимаем «Сохранить» |  |

## 3.5 Компоненты и структура сайта

Таблица №3 «Компоненты и структура сайта»

|  |  |
| --- | --- |
| В админке заходим в контент и переходим в файлы и папки. |  |
| Создаем новую папку и вписываем данные, после чего нажимаем «Сохранить» |  |
| Меняем название и вставляем нужный текст, после чего нажимаем «Сохранить» |  |
| Создаем меню и нажимаем «Сохранить» |  |
| Меняем название и вставляем нужный текст, после чего нажимаем «Сохранить» |  |
| Заходим на страницу test.php, нажимаем на редактирование страницы. Открываем меню справа, выбираем навигацию и заходим в меню. После два раза подтверждаем сохранение. |  |
| Включаем режим правки |  |
| Заходим в настройки |  |
| Выбираем тип для первого уровня «Главное меню» и нажимаем сохранить. |  |
| Вставили пункты главного меню |  |
| Заходим в настройки, меню и копировать шаблон компонента. |  |
| Меняем данные и нажимаем «Сохранить» |  |
| Нажимаем на изменить страницу и заходим в «В режиме PHP-кода» |  |
| Копируем эту часть кода |  |
| Вставляем в наш код в header.php |  |
| Вот что мы получаем |  |
| Нажимаем на создать раздел |  |
| Задаем заголовок, имя папки, подтверждаем вторую стрелочку и нажимаем «Далее». |  |
| Выбираем тип меню «Главное меню», вставить перед пунктом «Гарантия» и нажимаем «Далее». |  |
| Нажимаем «Готово» |  |
| Добавляем так же остальные пункты меню. |  |
| Создаем еще один новый раздел. Называем «Поиск» и даем имя нужной папки. Убираем все галочки если они есть и нажимаем «Готово». |  |
| Вот как это должно выглядеть. |  |
| Возвращаемся на страницу test.php, но теперь выбираем не меню, а навигационная цепочка. |  |
| Открывшееся окно просто сохраняем |  |
| Вот что получается |  |
| Заходим в настройки, навигационная цепочка и выбираем «Копировать шаблон компонента» |  |
| В название шаблона вставляем имя, что написано сверху в ссылке, выбираем копировать в текущий, применить новый шаблон компонента и нажимаем «Сохранить». |  |
| Вот что у нас получается в итоге. |  |

**Итог главной страницы**

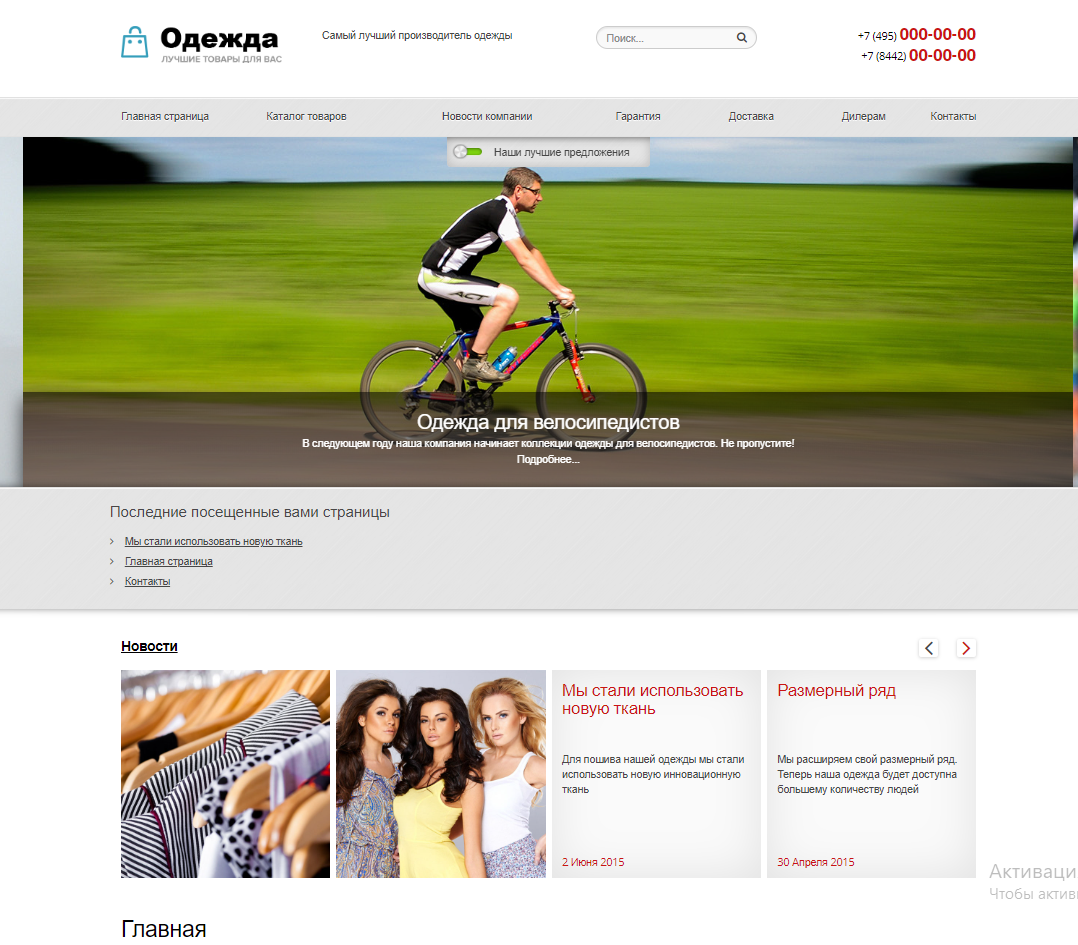


Рис. 16. « Итоговая Главная страница».

# 4. Разработка API для бэкэнда

## 4.1. 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 «Position»

class Position(models.Model):

    slug = models.SlugField()

    name = models.CharField('Название', max\_length=30)

    gramm = models.CharField('Граммовка порции', max\_length=10)

    sostav = models.TextField('Описание')

    added\_at = models.DateField(default=timezone.now)

    price = models.IntegerField('Цена')

    tags = TaggableManager()

    def \_\_str\_\_(self):

        return self.name

    class Meta:

        verbose\_name = «Позиция»

        verbose\_name\_plural = «Позиции меню»

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.

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 PositionSerializer(TaggitSerializer, serializers.ModelSerializer):

    tags = TagListSerializerField()

    class Meta:

        model = Position

        fields = ('slug', 'name', 'gramm', 'sostav', 'added\_at', 'price', 'tags')

        lookup\_field = 'slug'

        extra\_kwargs = {

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

        }

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

class PositionViewSet(viewsets.ModelViewSet):

    search\_fields = ['$name']

    filter\_backends = (filters.SearchFilter,)

    serializer\_class = PositionSerializer

    queryset = Position.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 PositionAdmin(admin.ModelAdmin):

pass

admin.site.register(Position, PositionAdmin)

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:

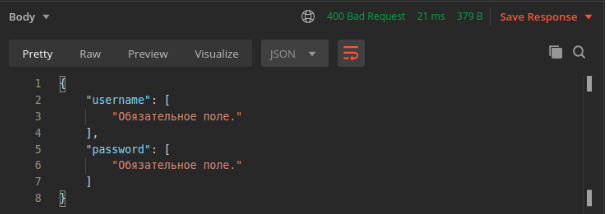


Рис. 17 «JSON и status code»

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

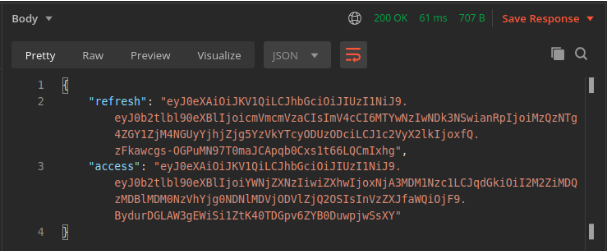


Рис. 18. «Получение токена»

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

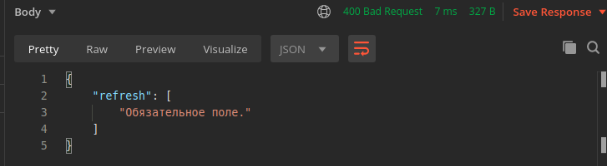


Рис. 19. «JSON и status code»

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.

## 4.2. Выполнение запросов к серверу

Было выполнено 5 запросов :

### 4.2.1. Последние 5 записей

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

Табл. №4. «Запрос на последние записи»

|  |  |
| --- | --- |
| **Описание запроса** | **api aside, которая возвращает нам последние 5 постов.** |
| Код серилизатора или view | class AsideView(generics.ListAPIView):  queryset = Position.objects.all().order\_by('-id')[:3]  serializer\_class = PositionSerializer  permission\_classes = [permissions.AllowAny] |
| Urls.py | path(«aside/», AsideView.as\_view()), |
| URL запроса | **http://localhost:8000/api/aside/** |
| Тип запроса | **GET** |
| Body запроса | **---** |
| Вывод: |  |

### 4.2.2. Поиск

Табл. №5. «Запрос на поиск»

|  |  |
| --- | --- |
| **Описание запроса** | **Api поиска по ключевому слову** |
| Код серилизатора или view | lass PositionViewSet(viewsets.ModelViewSet):  search\_fields = ['$name']  filter\_backends = (filters.SearchFilter,)  serializer\_class = PositionSerializer  queryset = Position.objects.all()  lookup\_field = 'slug'  permission\_classes = [permissions.AllowAny]  pagination\_class = PageNumberSetPagination |
| URL запроса | http://localhost:8000/api/positions/?q=Обычный |
| Urls.py | --- |
| Тип запроса | **GET** |
| Body запроса | **---** |
| Вывод: |  |

### 4.2.3. Регистрация

Табл. №6. «Запрос на регистрацию»

|  |  |
| --- | --- |
| **Описание запроса** | **Регистрация пользователя** |
| Код серилизатора и view | 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  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»,  }) |
| Urls.py | path('register/', RegisterView.as\_view()), |
| URL запроса | http://localhost:8000/api/register |
| Тип запроса | **POST** |
| Body запроса | {      »username»: »user2»,      »password»: »aboba123»,      »password2»: »aboba123»  } |
| Вывод: |  |

### 4.2.4. Комментарии

Для начала была создана модель Comments

from django.urls import path, include  
from rest\_framework.routers import DefaultRouter  
from .views import PostViewSet, TagDetailView, TagView, AsideView, RegisterView, ProfileView, CommentView  
  
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()),  
 path("comments/", CommentView.as\_view()),  
 path("comments/<slug:post\_slug>/", CommentView.as\_view()),  
]

После всех этих настроек и добавления структуры на вкладке Body, мы можем наконец создать комментарий:

Табл. №7. «Запрос комментариев»

|  |  |
| --- | --- |
| **Описание запроса** | **Вывод комментариев к определенной позиции меню** |
| Код серилизатора и view | class CommentView(generics.ListCreateAPIView):  queryset = Comment.objects.all()  serializer\_class = CommentSerializer  permission\_classes = [permissions.IsAuthenticated]  def get\_queryset(self):  position\_slug = self.kwargs['position\_slug'].lower()  position = Position.objects.get(slug=position\_slug)  return Comment.objects.filter(position=position)  # Комменты  class CommentSerializer(serializers.ModelSerializer):  username = serializers.SlugRelatedField(slug\_field=«username», queryset=User.objects.all())  position = serializers.SlugRelatedField(slug\_field=«slug», queryset=Position.objects.all())  class Meta:  model = Comment  fields = («id», «position», «username», «text», «created\_date»)  lookup\_field = 'id'  extra\_kwargs = {  'url': {'lookup\_field': 'id'}  } |
| Urls.py | http://localhost:8000/api/comments/tiramisu-cake/ |
| URL запроса | path(«comments/», CommentView.as\_view()),  path(«comments/<slug:position\_slug>/», CommentView.as\_view()), |
| Тип запроса | **GET** |
| Body запроса | **---** |
| Вывод: |  |

### 4.2.5. Заказ товара

Сначала была создана модель для заказа – Order:

class Order(models.Model) :  
 position = models.ForeignKey(Position, on\_delete=models.CASCADE, related\_name='orders')  
 owner = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='owner\_name')  
 quantity = models.IntegerField()  
 created\_date = models.DateTimeField(default=timezone.now)  
  
 class Meta :  
 ordering = ['-created\_date']  
 verbose\_name = «Заказ»  
 verbose\_name\_plural = «Заказы»  
  
 def \_\_str\_\_(self) :  
 return f»{self.position} - {self.quantity}»

Табл. №8. «Запрос заказов»

|  |  |
| --- | --- |
| **Описание запроса** | **Запрос добавляет заказ на определенную позицию** |
| Код серилизатора и view | class OrderSerializer(serializers.ModelSerializer):  owner = serializers.SlugRelatedField(slug\_field=«username», queryset=User.objects.all())  position = serializers.SlugRelatedField(slug\_field=«slug», queryset=Position.objects.all())   class Meta:  model = Order  fields = («id», «position», «owner», «quantity», «created\_date»)  lookup\_field = 'id'  extra\_kwargs = {  'url': {'lookup\_field': 'id'}  }  class OrderView(generics.ListCreateAPIView):  queryset = Order.objects.all()  serializer\_class = OrderSerializer  permission\_classes = [permissions.IsAuthenticated]   def get\_queryset(self):  position\_slug = self.kwargs['position\_slug'].lower()  position = Position.objects.get(slug=position\_slug)  return Order.objects.filter(position=position) |
| Urls.py | path(«orders/», OrderView.as\_view()),  path(«orders/<slug:position\_slug>/», OrderView.as\_view()), |
| URL запроса | http://localhost:8000/api/orders/ |
| Тип запроса | **POST** |
| Body запроса | {      »position»: »tiramisu-cake»,      »owner»: »admin»,      »quantity»: 2,      »Created date»: »«  } |
| Вывод: |  |

# 5. Проектирование предметной области

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

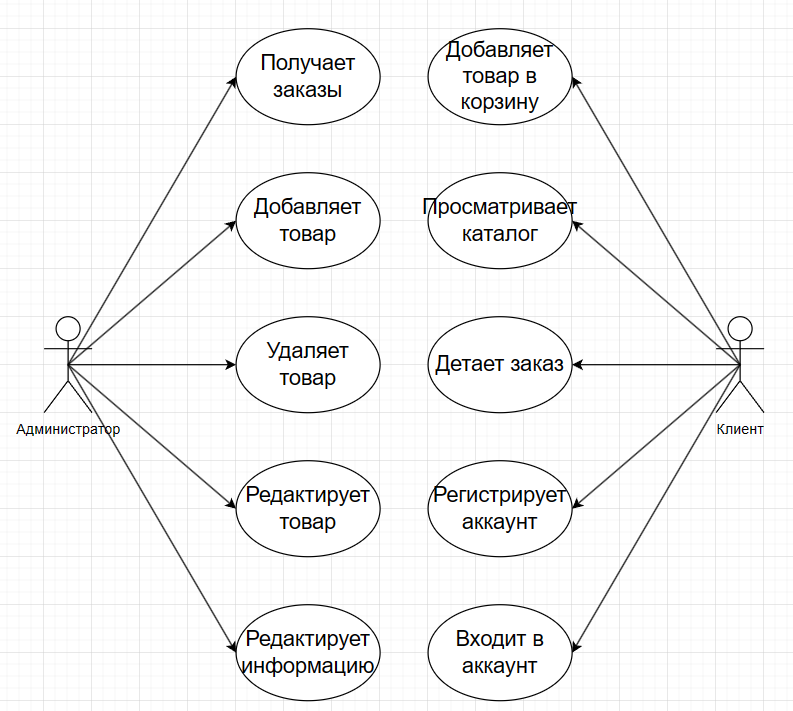


Рис. 0. «Диаграмма прецедентов»

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



Рис. 1. «Диаграмма действий»

# 6. Разработка технической документации

## 6.1. Техническое задание

**Введение**

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

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

Улучшение потребительского опыта: Сайт может предоставить дополнительные функциональности, такие как удобный поиск и интуитивно понятный интерфейс, что помогает пользователям найти нужный продукт.

**Основания для разработки**

Разработка проводятся на основании Договора Между ЛДПК и «CycleStore» № 1 от 11 января 2024 г.

**Назначение разработки**

Сайт предназначен для содержания подробной информации о продуктах, их характеристиках, ценах и других сведениях. Посетители могут легко ознакомиться с ассортиментом товаров.

**Требования к программе или программному изделию**

**Требования к функциональным характеристикам**

**Требования к составу выполняемых функций:**

* Добавление, удаление и редактирование товаров
* Фильтрация товаров по их названию
* Организация страницы с описанием деятельности компании
* Организация страницы с информацией о ремонте велосипедов
* Регистрация и вход пользователя в аккаунт

**Требования к организации входных данных:**

Входные данные программы организованы в БД в приложении со следующими полями: Физическое лицо – Логин, Email, Пароль.

**Требования к организации выходных данных:**

На странице «Главная» можно увидеть каталог, а также поисковую строку. С Главной страницы можно перейти на страницу с интересующим товаром.

На странице «О компании» представлена актуальная информация о компании.

На странице «Товар» представлена информация о конкретном товаре.

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

На странице «Регистрация» предоставляются поля для ввода данных, таких как: Имя пользователя, EMail (Не обязательно), пароль и подтверждение пароля.

**Требования к надежности.**

Сайт должен быть в достаточной степени надёжен от сбоев.

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

**Условия эксплуатации**

Программа не требует специального обслуживания. Для ознакомления с полным функционалом пользователь должен прочесть Руководство пользователя.

**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**

Табл. №9. «Технические параметры».

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

**Требования к информационной и программной совместимости**

Для корректной работы программы необходимо: ОС Windows 7,10,XP и выше, любой браузер, лучше Microsoft Edge или Google, JavaScript: JavaScript, HTML и CSS: HTML.

**Специальные требования**

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса.

**Требования к программной документации**

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

## 6.2. Руководство пользователя

**Назначение программы:**

Сайт предназначен для содержания подробной информации о продуктах, их характеристиках, ценах и других сведениях. Посетители могут легко ознакомиться с ассортиментом товаров.

**Условия выполнения программы:**

Табл. №10. «Технические параметры».

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

**Требования к программным средствам:**

Для корректной работы программы необходимо: ОС Windows 7,10,XP и выше, любой браузер, лучше Microsoft Edge или Google, JavaScript: JavaScript, HTML и CSS: HTML.

**Выполнение программы:**

При открытии программы (исполняемый «.exe» файл) появится главная страница, после по кнопке входа происходит переход к странице «Вход» с возможностью перехода на страницы.

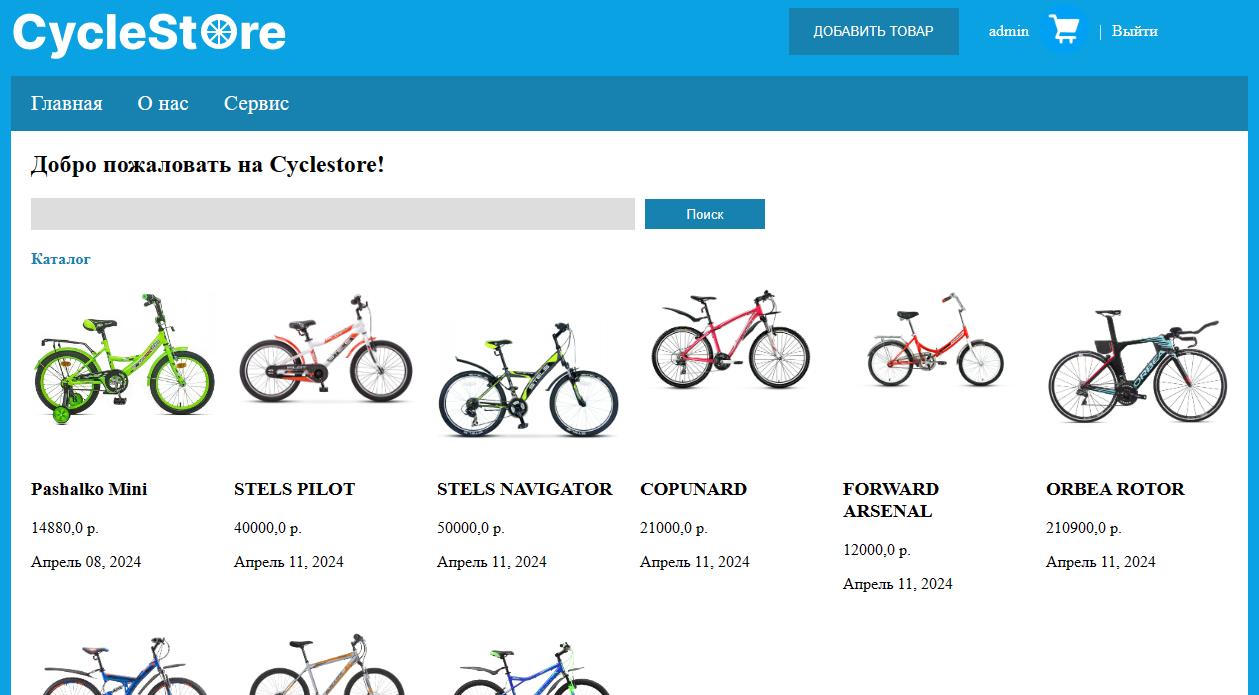


Рис. 22. «Главная страница»



Рис. 23. «Страница о компании»

**Добавление фильтрации/товара**

Для поиска необходимо нажать ЛКМ по кнопке с надписью «Введите название товара» или найти товар через синюю кнопку

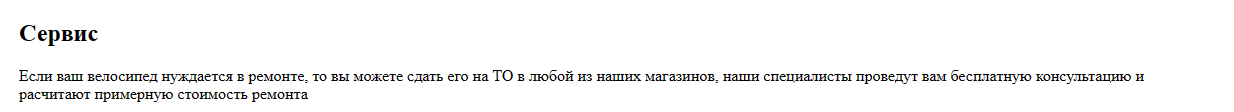
**

Рис. 24. «Страница сервис»

**Страница Входа пользователя**

Данная страница предназначена для аутентификации пользователя

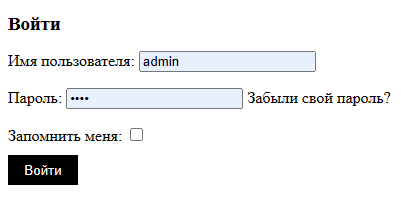
**

Рис.25. «Страница входа в аккаунт»

**Страница регистрации пользователя**

Данная страница предназначена для регистрации пользователей

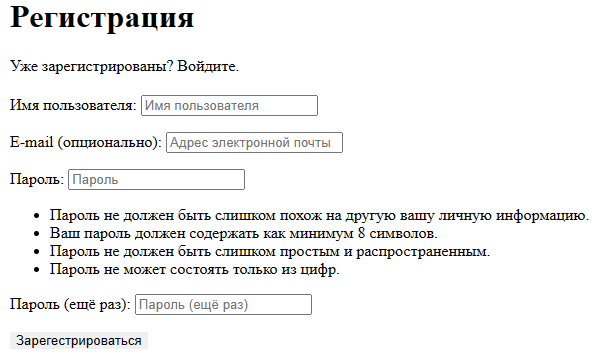


Рис. 26. «Регистрация»

**Добавление товара в каталог**

Товар добавляет только Администратор. Для добавления необходимо нажать по кнопке с надписью «Добавить товар» в шапке сайта и заполнить следующие поля: Заголовок товара, цена, выбор категории, способы доставки, описание товара, фото товара. Так-же товар можно добавить и через администрирование Django.

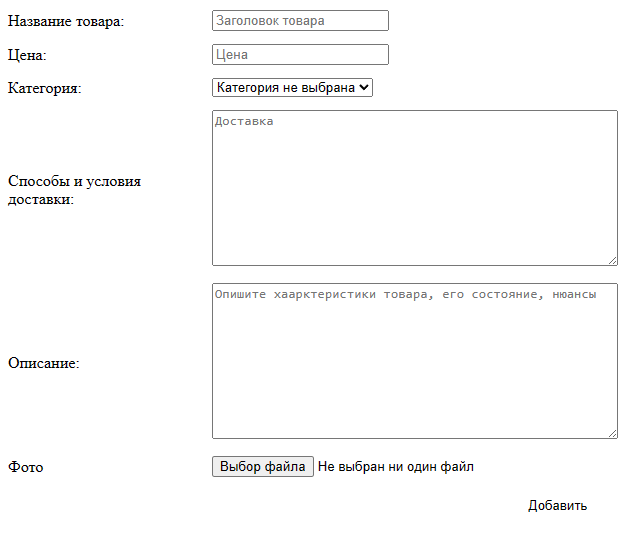


Рис. 7. «Форма добавления товара»

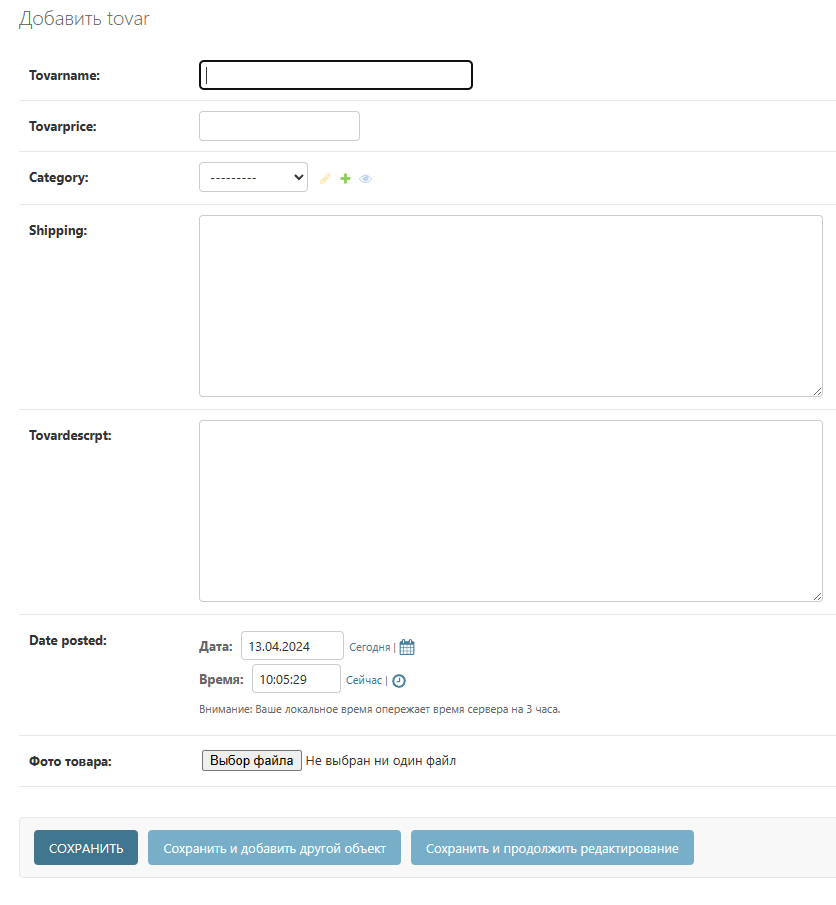
**

Рис. 28. «Администрирование Django»

**Страница товара и добавление в корзину**

Выберите товар в каталоге и перейдите на его страницу. Нажмите красную кнопку «Добавить в корзину».

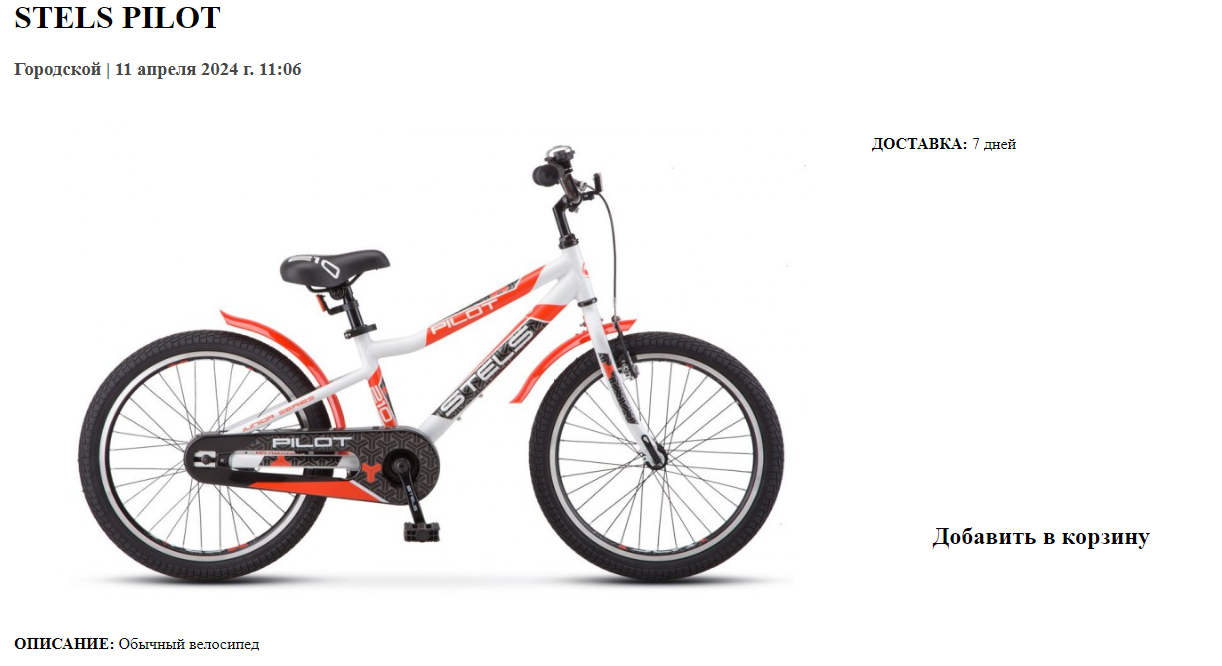


Рис. 29. «Страница товара»

**Просмотр товаров в корзине**

В корзину можно перейти, нажав на иконку Корзины в шапке сайта, после авторизации. В данном списке приведены все товары, добавленные пользователем в корзину.

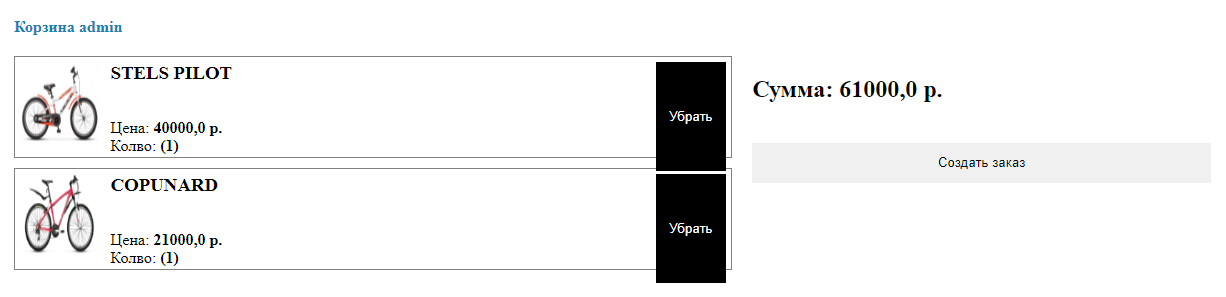


Рис. 30. «Корзина пользователя»

**Удаление товара из корзины**

Для удаления товара из корзины нужно нажать на кнопку «Удалить» в строке с товаром.



Рис. 31. «Товар в корзине»

**Фильтры по названию**

Фильтры находятся на главной странице. Для фильтрации по названию – впишите ключевое слово в поле ввода, а затем нажмите кнопку «Поиск» или Enter на клавиатуре

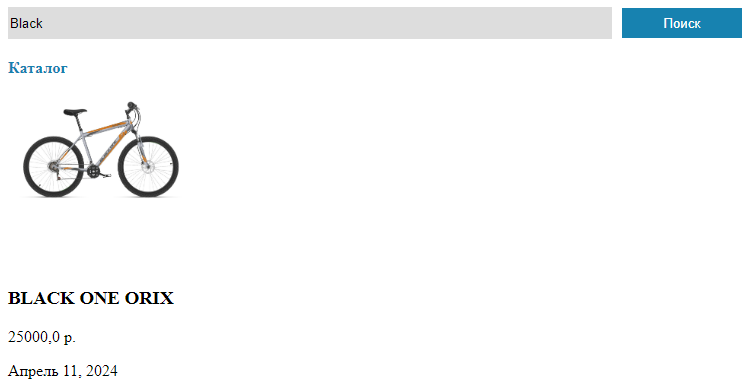


Рис. 32. «Результаты фильтрации по названию»

**Сообщение оператору:**

**Попытка входа пользователя без регистрации**



Рис. 33. «Попытка добавления договора»

**Сброс пароля**

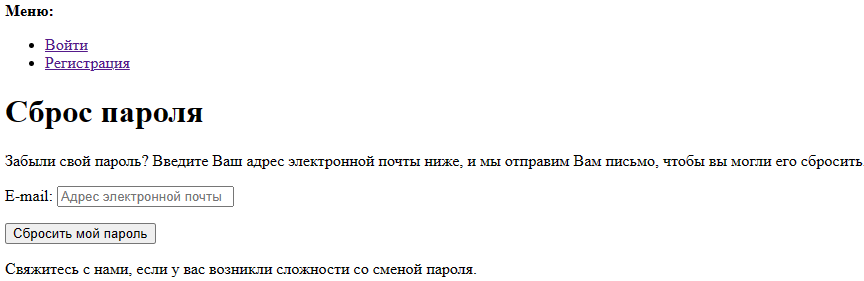
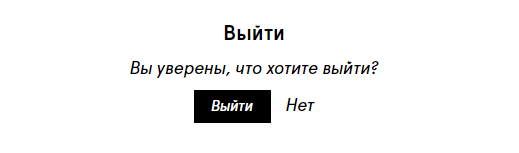


Рис. 34. «Сброс пароля»

**Сообщение при выходе из аккаунта**



Риc. 35. «Сообщение при выходе из аккаунта».

## 6.3. Руководство программиста

**Назначение и условия применения программы:**

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

Табл. №11. «Технические параметры».

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

**Требования к программным средствам:**

Для корректной работы программы необходимо: ОС Windows 7, 10, XP и выше, любой браузер, лучше Microsoft Edge или Google, JavaScript: JavaScript, HTML и CSS: HTML.

**Характеристика программы:**

Программа должна обеспечивать следующие функции:

* Добавление, удаление товаров
* Поиск физических лиц по ФИО
* Фильтрация товаров по их названию и классу
* Организация страницы с отзывами пользователей
* Регистрация и вход пользователя в аккаунт

**Обращение к программе:**

PyCharm — это среда программирования для языка Python, или IDE. Средами называют программы, в которых можно писать, запускать и отлаживать код, устанавливать новые расширения и дополнительные модули.

Код страницы «Главная»

{% extends 'base.html' %}  
  
{% block title %}Главная{% endblock %}  
  
{% block content%}  
<h2>Добро пожаловать на Cyclestore! </h2>  
  
  
<form method="get" class="form-filter">  
 <div class="form-filter\_\_inputs">  
 {{ tovars.form.tovarname }}  
 <button type="submit" class="btn btn-primary">Поиск</button>  
 </div>  
</form>  
  
<section class="tovars">  
 <h2 class="smallhead">Каталог</h2>  
 <div class="tovars\_\_line">  
 {% for tovar in tovars.qs %}  
 <div class="tovar-block">  
 <div class="tovar-block\_\_image"><img src="{{ tovar.tovarimage.url}}" alt="connect" class="tovar-block\_\_image-img" style="max-height:185px;max-width: 185px;"></div>  
 <h3><a href="{% url 'tovar' tovar.id %}">{{ tovar.tovarname }}</a></h3>  
 <p>{{ tovar.tovarprice }} р.</p>  
 <p>{{ tovar.date\_posted|date:"F d, Y" }}</p>  
 </div>  
 {% endfor %}  
 </div>  
</section>  
{% endblock %}

Код страницы «О нас»

{% extends 'base.html' %}  
{% load static %}  
  
{% block title %}О компании{% endblock %}  
  
{% block content%}  
<center><img src="{% static 'img/logo.png' %}" alt="logo"></center>  
<p>Компания CycleStore уже более 15 лет на рынке велосипедов.</p>  
{% endblock %}

Код страницы «Сервис»

{% extends 'base.html' %}  
  
{% block title %}Обслуживание{% endblock %}  
  
{% block content%}  
<h2>Сервис</h2>  
<p>Если ваш велосипед нуждается в ремонте, то вы можете сдать его на ТО в любой из наших магазинов,  
 наши специалисты проведут вам бесплатную консультацию и расчитают примерную стоимость ремонта</p>  
  
{% endblock %}

**Входные и выходные данные:**

Табл. №12. «Входные и выходные данные».

|  |  |
| --- | --- |
| **Поле** | **Тип данных** |
| Название товара | CharField |
| Цена | FloatField |
| Категория | ForeignKey |
| Доставка | TextField |
| Описание товара | TextField |

**Выходная информация:**

Выходные данные организованы в виде списка карточек товаров.

**Сообщения об ошибках:**

Ошибка, возникающая если попытаться добавить в каталог товар не выбрав категорию.

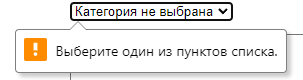


Рис. 36. «Ошибка добавления товара – категория»

Ошибка, возникающая если попытаться добавить в каталог товар не выбрав фотографию.

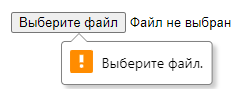


Рис. 37. «Ошибка добавления товара – фото»

Ошибка сайта при попытке добавить товар в корзину не авторизовавшись.

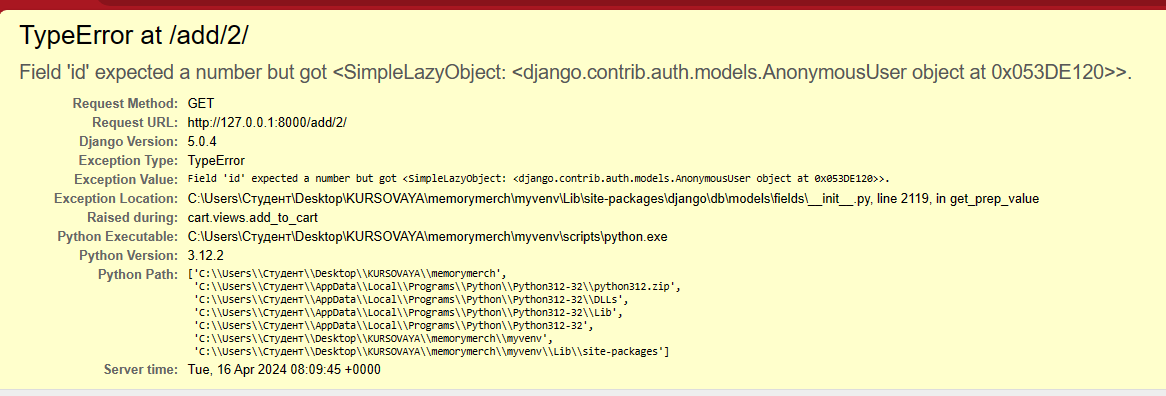


Рис. 38. «Ошибка добавления товара в корзину»

## 6.4. Методика тестирования и испытания программы

**Наименование**

«CycleStore»

**Область применения**

Сайт предназначен для содержания подробной информации о продуктах, их характеристиках, ценах, наличии и других сведениях. Посетители могут легко ознакомиться с ассортиментом товаров.

**Обозначение программы**

«Интернет-магазин по продаже велосипедов»

**Цель испытаний**

Цель проведения испытаний - проверка соответствия характеристик разработанного сайта функциональным и отдельным иным видам требований, изложенным в документе Техническое задание.

**Общие положения**

**Основания для проведения испытаний**

Разработка проводятся на основании Договора Между ЛДПК и «CycleStore» № 3 от 12 апреля 2024 г.

**Место и продолжительность испытаний**

Приемосдаточные испытания должны проводиться на объекте заказчика в сроки с 12.04.2024 по 21.04.2024

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;

**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»
* Организация страницы с товарами и фильтрацией
* Организация страницы с описанием компании
* Организация страницы «Сервис» с описанием ремонта товаров
* Организация страницы «Заявки» с информацией об актуальных и архивных заявках
* Организация страницы «Товар» с возможностью посмотреть информацию о товаре

**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.

**Средства и порядок испытаний**

**Технические средства, использованные во время испытаний:**

Системные требования для работы программного продукта должны быть следующими:

Табл. №13. «Технические средства».

|  |  |
| --- | --- |
| Технические средства | Требования |
| Браузеры | Последняя версия Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera и др. |
| Устройства | 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 |
| Хост | Новая база данных, поддержка Python, виртуальный выделенный сервер, объем дискового пространства, предоставляемого для размещения сайта. |

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

- Веб-браузер: любой браузер последней версии

- Операционная система Windows 10 и 11;

- Устройства с разрешением экрана от 1280x720 px

- Клавиатура и мышь

- Среда работы с терминалом python (PyCharm)

**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**

Табл. №14. «Тест ТС\_ПИ\_1».

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_ПИ\_1. Добавление товара через страницу добавления |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Добавление товара |
| **Краткое изложение теста** | Должно производится добавление товара через страницу добавления без ошибок |
| **Этапы теста** | Вход на сайт  Вход в админку  На главной странице нажать на кнопку «Добавление»  Ввод данных в текстовые поля и выборка данных из списка  Для сохранения данных нажать на кнопку «Сохранить» |
| **Тестовые данные** | Название товара  Цена  Категория (выбор из списка)  Условия доставки  Описание товара  Фото |
| **Ожидаемый результат** | Информация успешно сохранена и товар добавлен в каталог |
| **Фактический результат** |  |
| **Статус** | Зачёт |
| **Предварительное условие** | Перед проверкой теста нужно зайти на сайт и нажать на кнопку «Добавление» |
| **Постусловие** | После выполнения теста нажать на кнопку «Сохранить». |
| **Примечания/комментарии** | Данный тест пройден успешно. |

Табл. №15. «Тест ТС\_ПИ\_2».

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_ПИ\_2. Авторизация пользователя |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Авторизация |
| **Краткое изложение теста** | Пользователь должен перенестись на Главную страницу после авторизации |
| **Этапы теста** | Вход на сайт  Нажать на кнопку авторизация  Ввести логин пользователя  Ввести пароль от аккаунта  Нажать на кнопку «Авторизация» |
| **Тестовые данные** | Логин - admin  Пароль - 1234 |
| **Ожидаемый результат** | Пользователь успешно авторизовался и был перенесен на главную |
| **Фактический результат** |  |
| **Статус** | Зачёт |
| **Предварительное условие** | Перед проверкой теста нужно зайти на сайт и нажать на кнопку «Авторизация» |
| **Постусловие** | После выполнения теста нажать на кнопку «Авторизация». |
| **Примечания/комментарии** | Данный тест пройден успешно. |

Табл. №16. «Тест ТС\_ПИ\_3».

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_ПИ\_3. Регистрация пользователя |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Регистрация |
| **Краткое изложение теста** | Пользователь должен перенестись на Главную страницу после регистрации |
| **Этапы теста** | Вход на сайт  Нажать на кнопку регистрация  Ввести имя пользователя  Ввести почту (по желанию)  Ввести пароль и повторить его  Нажать на кнопку «Регистрация» |
| **Тестовые данные** | Логин - admin2  Пароль – kazah123 |
| **Ожидаемый результат** | Пользователь успешно зарегистрирован и перенесен на Главную |
| **Фактический результат** |  |
| **Статус** | Зачёт |
| **Предварительное условие** | Перед проверкой теста нужно зайти на сайт и нажать на кнопку «Регистрация» |
| **Постусловие** | После выполнения теста нажать на кнопку «Регистрация». |
| **Примечания/комментарии** | Данный тест пройден успешно. |

## 6.5. Тест-план программы

**Цель**

Целью составления данного Тест Плана является описание процесса тестирования сайта CycleStore. Документ позволяет получить представление о плановых работах по тестированию проекта.

**Исходные данные**

CycleStore - сайт, позволяющий пользователю узнать больше о товарах, просматривать, добавлять в корзину и заказывать их.

**Цели тестирования**

Целью тестирования сайта CycleStore 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.

Условия для тестирования

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

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 работы продукта и его дизайна. Все обнаруженные дефекты будут занесены в виде отдельных тикетов для последующего исправления в багтрекер заказчика.

В процессе тестирования сайта CycleStore 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 так же позволяет разработчикам проекта исправлять дефекты на самых ранних этапах.

ОС, утвержденные к проверке:

* Windows 7,10, XP

Браузеры, утвержденные к проверке:

* Google Chrome 70.0.3538.102
* Opera 56.0.3051.104
* Mozilla Firefox 63.0.3

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

Типы тестирования

**Функциональное тестирование**

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

Описание процесса:

Регистрация/Авторизация

* Регистрация пользователя
* Авторизация пользователя
* Выход из системы

***Обратная связь***

* Управление картой
* Контактная информация о компании

Товары

* Информация о товаре и цена к нему

Поиск/Фильтрация

* Поиск по названиям товаров, разделам
* Фильтрация товаров

Баннеры

* Корректное отображение баннеров со скидками

Отзывы

* Корректное отображение поля комментария
* Проверка на валидность

# 7. Тестирование кроссбраузерности

Цель: Проверить корректную работу и дизайн проекта в различных браузерах

Браузеры

* Google Chrome 70.0.3538.102
* Opera 56.0.3051.104
* Yandex 55.4.4
* Mozilla Firefox 63.0.3

**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

**Тестирование дизайна**

Цель: Проверка соответствия дизайна продукта макетам спецификации

Описание процесса:

* Регистрационная форма
* Письмо пользователю
* Личный кабинет
* Страницы сайта

План работ

Табл. №17. «План работ».

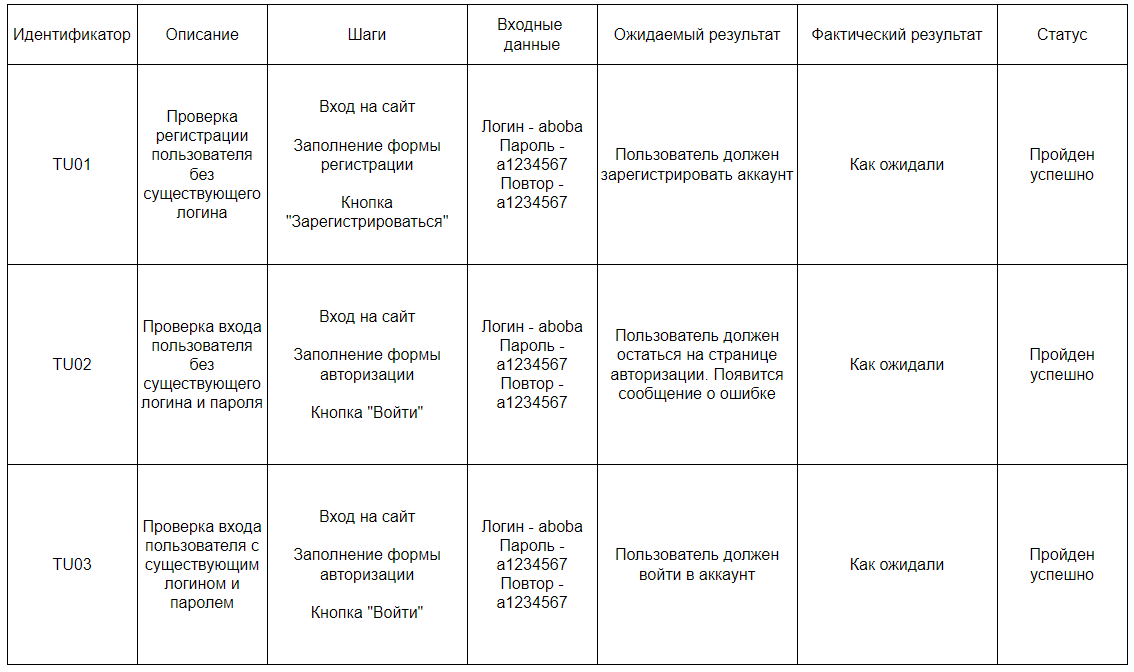
|  |  |  |  |
| --- | --- | --- | --- |
| **Задача** | **Объем работы** | **Дата начала** | **Дата завершения** |
| Составление тест плана | 5 часов | 27.03.2024 | 27.03.2024 |
| Выполнение тестирования | 4 часа | 28.03.2024 | 28.03.2024 |
| Анализ тестирования | 8 часов | 29.03.2024 | 30.03.2024 |
| Подведение итогов | 2 часа | 31.03.2024 | 31.03.2024 |

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**

Табл. №18. «Результаты тестирования».



**Рекомендации для корректировки тестируемой программы.**

* Возможность добавления отзывов под любым товаром.
* Добавить профиль пользователя, отдельную форму с его покупками, именем, фотографией и т.д.
* Обеспечения качества программного обеспечения

Краткая теория и методические рекомендации:

**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.

Табл. №19. «Тесты».

|  |  |  |
| --- | --- | --- |
| **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 | Экспертный | 0 |
| Полнота обработки ошибочных данных | Экспертный | 1 |
| Наличие тестов для проверки допустимых значений входных данных | Экспертный | 1 |
| Наличие системы контроля полноты входных данных | Экспертный | 0 |
| Наличие средств контроля корректности входных данных | Экспертный | 0 |
| Наличие требований к программе по восстановлению процесса выполнения в случае сбоя ОС, внешних устройств, процессора | Экспертный | 1 |
| Наличие требований к программе по восстановлению результатов при отказах ОС, внешних устройств, процессора | Экспертный | 1 |
| Наличие средств восстановления при сбоях оборудования | Экспертный | 1 |
| Наличие возможности повторного старта с точки прерывания | Экспертный | 0 |
| Наличие обработки неопределенностей | Экспертный | 1 |
| Наличие возможности автоматически обходить ошибочные ситуации в процессе вычисления | Экспертный | 1 |
|  | Всего: | 8 |
| ***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 | 1 |
| 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 | 0 |
| Наличие ограничений на размеры модуля | Экспертный | 1 |
| Наличие модульной схемы программы | Экспертный | 1 |
|  | Всего: | 5 |
| ***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 | 0 |
| Наличие описаний частных функций | Экспертный | 0 |
| Наличие описания данных | Экспертный | 0 |
| Наличие описания алгоритмов | Экспертный | 0 |
| 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 | 1 |
| Реанимация всех основных функций | Экспертный | 1 |
| Реанимация всех алгоритмов | Экспертный | 1 |
| Наличие определений всех данных: переменные, индексы, массивы и пр. | Экспертный | 1 |
| Наличие интерфейсов с пользователем | Экспертный | 1 |
| Отсутствие противоречий в выполнении основных функций | Экспертный | 0 |
| Отсутствие противоречий в выполнении частных функций | Экспертный | 1 |
| Отсутствие противоречий в выполнении алгоритмов | Экспертный | 1 |
| Правильность взаимосвязей | Экспертный | 1 |
| Правильность реализаций интерфейса с пользователем | Экспертный | 0 |
| Отсутствие противоречий в настройке системы | Экспертный | 1 |
| Комплектность документации в соответствии со стандартами | Экспертный | 0 |
|  | 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.

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.

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.

Табл. №20. «Качества показателей».

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Показатели качества** | **Сущность показателя** | **Экспертная оценка (вес) wi** | **Оценка, установленная экспериментом, ri** | **Качество показателя** |
| 1. Надежность | определяется бесперебойной устойчивостью в работе программ, точностью выполнения предписанных функций обработки, возможностью диагностики возникающих в процессе работы программ ошибок. | 1 |  | 0,37 |
| 1.1.Последовательность проектирования |  | 0,6 | 0,4 | 0,24 |
| 1.2.Правильность |  | 0,1 | 1 | 0,1 |
| 1.3Устойчивость к ошибкам |  | 0,3 | 0,1 | 0,03 |
| 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 |  | 0,22 |
| 2.1.Прослеживаемость |  | 0,3 | 0,5 | 0,15 |
| 2.2.Функциональная полнота |  | 0,6 | 0,1 | 0,06 |
| 2.3.Последовательность проектирования |  | 0,1 | 0,1 | 0,01 |
| 3.Эффективность | способность ПО обеспечивать требуемый уровень производительности в соответствии с выделенными ресурсами, временем и другими условиями. | 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 |  | 0,6 |
| 4.1.Расширяемость |  | 0,5 | 0,5 | 0,25 |
| 4.2.Широта использования |  | 0,5 | 0,7 | 0,35 |
| 5.Функциональность | способность ПО решать задачи, которые соответсвуют потребностям пользователя при заданных условиях оспользования ПО | 1 |  | 0,29 |
| 5.1. Защищённость |  | 0,2 | 0,1 | 0,02 |
| 5.2. Согласованность |  | 0,4 | 0,3 | 0,12 |
| 5.3. Соответствие стандартам |  | 0,1 | 0,2 | 0,02 |
| 5.4. Функциональная пригодность |  | 0,1 | 0,9 | 0,09 |
| 5.5. Способность к взаимодействию |  | 0,2 | 0,2 | 0,04 |
| 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 |  | 0,6 |
| 6.1. Ошибкоустойчивость/Целостность |  | 0,6 | 0,4 | 0,24 |
| 6.2. Доступность |  | 0,2 | 1 | 0,2 |
| 6.3. Коммуникативность |  | 0,2 | 0,8 | 0,16 |
| 7. Целостность | Состояние программного обеспечения и данных, характеризующееся отсутствием изменений преднамеренного или случайного характера. | 1 |  | 0,84 |
| 7.1. Управление доступом | 0,4 | 0,6 | 0,24 |
| 7.2. Контроль за доступом | 0,6 | 1 | 0,6 |
| 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. | 1 |  | 0,64 |
| 8.1. Модульность |  | 0,2 | 0 | 0 |
| 8.2. Унификация интерфейсов |  | 0,4 | 1 | 0,4 |
| 8.3. Унификация данных |  | 0,4 | 0,6 | 0,24 |
| 9. Сопровождаемость | характеристики программного продукта, позволяющие минимизировать усилия по внесению в него изменений: | 1 |  | 0,42 |
| 9.1. Простота работы |  | 0,6 | 0,2 | 0,12 |
| 9.2. Краткость |  | 0,1 | 0,9 | 0,09 |
| 9.3. Информативность |  | 0,2 | 1 | 0,2 |
| 9.4. Модульность |  | 0,1 | 0,1 | 0,01 |
| 10. Модифицируемость | факторы качества, важные для разработки новой версии ПО; | 1 |  | 0,52 |
| 10.1. Структурированность |  | 0,8 | 0,4 | 0,32 |
| 10.2. Дополняемость |  | 0,2 | 1 | 0,2 |
| 11. Мобильность | перенос По с одного окружения в другое. | 1 |  | 0,77 |
| 11.1. Простота работы |  | 0,2 | 0,5 | 0,1 |
| 11.2. Независимость программной платформы |  | 0,6 | 1 | 0,6 |
| 11.3. Независимость от аппаратной платформы |  | 0,1 | 0,2 | 0,02 |
| 11.4. Унификация данных |  | 0,1 | 0,5 | 0,05 |
| 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 |  | 0,52 |
| 12.1. Простота работы |  | 0,2 | 0,9 | 0,18 |
| 12.2. Широта использования |  | 0,3 | 0,6 | 0,18 |
| 12.3. Независимость программной платформы |  | 0,3 | 0 | 0 |
| 12.4. Независимость от аппаратной платформы |  | 0,2 | 0,8 | 0,16 |
| 13. Устойчивость | это способность программной системы должным образом реагировать на исключительные ситуации. Обработка исключительных ситуаций - процесс, направленный на достижение устойчивости. | 1 |  | 1 |
| 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 |  | 0,48 |
| 14.1. Простота работы |  | 0,2 | 0,4 | 0,08 |
| 14.2. Полнота протоколирования |  | 0,2 | 0,1 | 0,02 |
| 14.3. Информативность |  | 0,2 | 0,6 | 0,12 |
| 14.4. Расширяемость |  | 0,2 | 1 | 0,2 |
| 14.5. Модульность |  | 0,2 | 0,3 | 0,06 |
| 15. Понятность | это свойство, характеризующее, насколько легко (или трудно) специалисту, сопровождающему программный продукт, понять его работу. | 1 |  | 0,79 |
| 15.1. Последовательность проектирования |  | 0,1 | 0,9 | 0,09 |
| 15.2. Информативность |  | 0,3 | 0,7 | 0,21 |
| 15.3. Структурированность |  | 0,4 | 1 | 0,4 |
| 15.4. Краткость |  | 0,1 | 0,3 | 0,03 |
| 15.5. Удобочитаемость |  | 0,1 | 0,6 | 0,06 |
| 16. Практичность | способность программного продукта быть понятным, изученным, использованным и привлекательным для пользователя при применении в заданных условиях | 1 |  | 0,73 |
| 16.1. Удобство работы |  | 0,1 | 0,8 | 0,08 |
| 16.2. Удобство обучения |  | 0,4 | 1 | 0,4 |
| 16.3. Способность к взаимодействию |  | 0,5 | 0,5 | 0,25 |
| 17. Производительность | это мера того, насколько эффективно приложение использует ресурсы системы для выполнения действий, которые вы создали для этого. | 1 |  | 1 |
| 18. Зрелость процесса | это инструмент, помогающий организации в оценке и определении степени зрелости ее *процессов внедрения*. | 1 |  | 1 |
| 19. Масштабируемость | это способность системы адаптироваться к расширению предъявляемых требований и возрастанию объемов решаемых задач. | 1 |  | 1 |
| 20. Безопасность | защита приложений, которую их создатели обеспечивают уже на этапе разработки и до того, как программа окажется в открытом доступе. | 1 |  | 1 |
| 21. Эксплуатационная пригодность | Способность сооружения, его отдельного узла, элемента, изделия или конструкции выполнять эксплуатационные функции, для которых объект проектировался и использовался. | 1 |  | 1 |
|  | Обобщённый показатель качества: |  |  | 70% |
|  |  |  |  |  |
| Вывод: программа считается качественной, 70% качества. Для повышения качества необходимо повысить: Корректность, Надежность, Функциональность, Мобильность. Не реализовано: 8.1. Модульность, 12.3. Независимость программной платформы. | | | | |
|

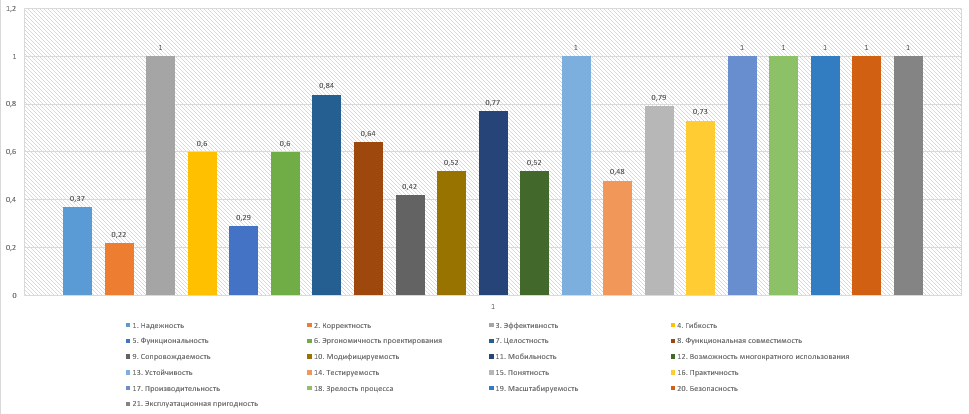


Рис. 39. «Результаты фильтрации по названию»

# 8. Расчет надежности работы сайта

**Краткая теория и методические рекомендации:**

**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.**

**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 50 испытаний, при которых было выявлено следующее число ошибок по типам. Из 50 испытаний 13 было проведено безуспешно. Необходимо определить надёжность программы.

Табл. №21. «Расчет надежности работы сайта».



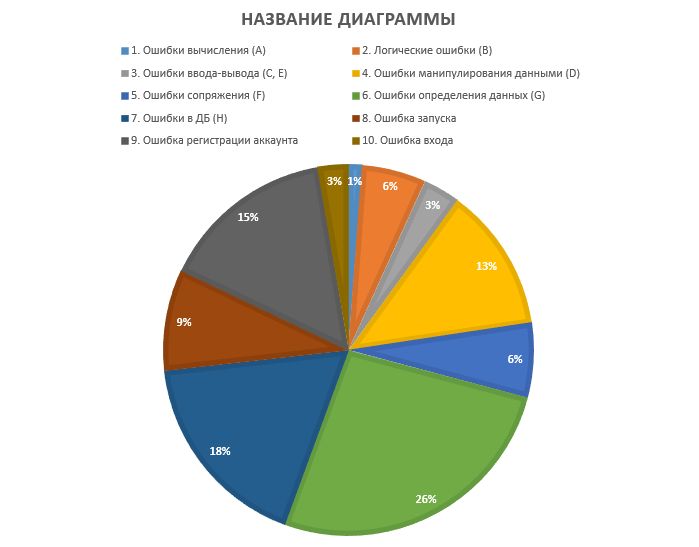


Рис. 40. «Результаты фильтрации по названию»

**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

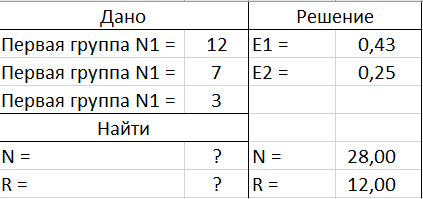
- ошибки, обнаруженные второй группой

- ошибки обнаруженные обеими группами

- предполагаемое количество ошибок в программе

**Условие:** При проведении тестирования первой группой было обнаружено 12 ошибок, второй – 7 ошибок. Общих ошибок - 3. Определить надежность по простой интуитивной модели.

Табл. №22. «Расчет надежности».





**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 50 испытаний, при которых было выявлено следующее число ошибок по типам. Из 50 испытаний 13 было проведено безуспешно. Необходимо определить надёжность программы.

# Заключение

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

Во время учебной практики выполнялись следующие виды работ:

* Разработка web-приложения (сайта);
* Проектирование предметной области;
* Роутинг в CMS Bitrix;
* Разработка компонентов и структуры сайта в CMS Bitrix;
* Разработка API для бэкэнда
* Проведение тестирования и отладки web-приложения (сайта);
* Разработка технической документации;

Оценка качественных показателей программного средства с помощью метрик;

Оценка надежности программного средства.

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

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

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 10, любой браузер, HTML, CSS и Python.

# 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/