from .models import Product, Cart, CartItem, Order, OrderItem

from .forms import OrderForm

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

from django.views.decorators.http import require\_POST

from django.http import JsonResponse

from django.contrib import messages

from django.shortcuts import render

from django.core.mail import send\_mail

from django.conf import settings

from .forms import ContactForm

from django.contrib import messages

def home(request):

"""

Представление для главной страницы.

"""

return render(request, 'production/home.html')

def product\_list(request):

products = Product.objects.all()

return render(request, 'production/product\_list.html', {'products': products})

def product\_detail(request, product\_id):

product = get\_object\_or\_404(Product, id=product\_id)

data = {

'id': product.id,

'name': product.name,

'price': str(product.price), # Преобразуйте Decimal в строку

'description': product.description,

'image': product.image.url if product.image else None, # Обработайте отсутствие изображения

'specs': {

'Вес': '1 кг', # Замените на реальные данные из вашей модели

'Срок хранения': '12 месяцев',

'Упаковка': 'Пакет',

'Сорт': 'Высший'

}

}

return JsonResponse(data)

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

from django.views.decorators.http import require\_POST

from django.http import JsonResponse

from .models import Product, Cart, CartItem

import json

import logging

from django.shortcuts import get\_object\_or\_404

from django.http import JsonResponse

from django.views.decorators.http import require\_POST

from .models import Product, Cart, CartItem

logger = logging.getLogger(\_\_name\_\_) # Получаем логгер

@require\_POST

def cart\_add(request, product\_id):

"""

Добавляет товар в корзину или обновляет его количество.

"""

product = get\_object\_or\_404(Product, id=product\_id)

quantity = int(request.POST.get('quantity', 1))

try:

# Получаем или создаем корзину для текущей сессии

cart, created = Cart.objects.get\_or\_create(session\_id=request.session.session\_key)

# Получаем или создаем элемент корзины

cart\_item, created = CartItem.objects.get\_or\_create(cart=cart, product=product)

# Обновляем количество

cart\_item.quantity += quantity

cart\_item.save()

# Рассчитываем общее количество товаров в корзине

cart\_total\_items = sum(item.quantity for item in cart.items.all())

return JsonResponse({

'success': True,

'message': 'Товар добавлен в корзину',

'cart\_total\_items': cart\_total\_items

})

except Exception as e:

# Логируем ошибку с трассировкой

logger.exception("Error adding to cart:")

return JsonResponse({

'success': False,

'message': f'Ошибка при добавлении в корзину: {e}'

}, status=400)

def cart\_update\_quantity(request, item\_id):

"""

Обновляет количество товара в корзине.

"""

cart\_item = get\_object\_or\_404(CartItem, id=item\_id)

quantity = int(request.POST.get('quantity'))

if quantity > 0:

cart\_item.quantity = quantity

cart\_item.save()

return JsonResponse({'message': 'Количество обновлено', 'success': True})

else:

cart\_item.delete()

return JsonResponse({'message': 'Товар удален', 'success': True})

@require\_POST

def cart\_remove(request, item\_id):

"""

Удаляет товар из корзины.

"""

cart\_item = get\_object\_or\_404(CartItem, id=item\_id)

cart\_item.delete()

return JsonResponse({'message': 'Товар удален', 'success': True})

def cart\_detail(request):

"""

Отображает содержимое корзины.

"""

# Получаем корзину для текущей сессии

cart = None

try:

cart = Cart.objects.get(session\_id=request.session.session\_key)

except Cart.DoesNotExist:

pass # Корзина еще не создана для этой сессии

return render(request, 'production/cart\_detail.html', {'cart': cart})

def create\_order(request):

"""

Представление для создания заказа.

"""

cart = None

try:

cart = Cart.objects.get(session\_id=request.session.session\_key)

except Cart.DoesNotExist:

messages.error(request, "Ваша корзина пуста.")

return redirect('cart\_detail') # Перенаправляем на страницу корзины

if request.method == 'POST':

form = OrderForm(request.POST)

if form.is\_valid():

# Создаем заказ

order = Order.objects.create(

first\_name=form.cleaned\_data['first\_name'],

last\_name=form.cleaned\_data['last\_name'],

email=form.cleaned\_data['email'],

phone=form.cleaned\_data['phone'],

address=form.cleaned\_data['address'],

postal\_code=form.cleaned\_data['postal\_code'],

city=form.cleaned\_data['city']

)

# Создаем элементы заказа

for item in cart.items.all():

OrderItem.objects.create(

order=order,

product=item.product,

price=item.product.price, # Сохраняем цену на момент заказа

quantity=item.quantity

)

# Очищаем корзину

cart.items.all().delete()

cart.delete()

request.session.pop('session\_key', None) # Очищаем session\_key

# Перенаправляем на страницу подтверждения заказа

return redirect('order\_confirmation', order\_id=order.id)

else:

# Форма не валидна, отображаем ошибки

return render(request, 'production/create\_order.html', {'cart': cart, 'form': form})

else:

form = OrderForm()

return render(request, 'production/create\_order.html', {'cart': cart, 'form': form})

def order\_confirmation(request, order\_id):

order = get\_object\_or\_404(Order, id=order\_id)

return render(request, 'production/order\_confirmation.html', {'order': order})

def contact(request):

if request.method == 'POST':

form = ContactForm(request.POST)

if form.is\_valid():

name = form.cleaned\_data['name']

email = form.cleaned\_data['email']

phone = form.cleaned\_data['phone']

subject = form.cleaned\_data['subject']

message = form.cleaned\_data['message']

# Отправка email

try:

send\_mail(

f'Сообщение с сайта от {name} ({email})',

f'Телефон: {phone}\n\n{message}',

settings.DEFAULT\_FROM\_EMAIL, # Укажите в settings.py

[settings.CONTACT\_EMAIL], # Укажите в settings.py

fail\_silently=False,

)

messages.success(request, 'Ваше сообщение успешно отправлено! Мы свяжемся с вами в ближайшее время.')

return redirect('contact') # Перенаправляем на эту же страницу

except Exception as e:

messages.error(request, f'Произошла ошибка при отправке сообщения. Пожалуйста, попробуйте позже. Ошибка: {e}')

# Логирование ошибки (вместо print)

print(f"Ошибка при отправке email: {e}")

else:

form = ContactForm()

return render(request, 'production/contact.html', {'form': form})

from django.urls import path

from . import views

urlpatterns = [

path('', views.home, name='home'),

path('products/', views.product\_list, name='product\_list'),

path('cart/add/<int:product\_id>/', views.cart\_add, name='cart\_add'),

path('cart/', views.cart\_detail, name='cart\_detail'),

path('products/<int:product\_id>/', views.product\_detail, name='product\_detail'),

path('cart/update/<int:item\_id>/', views.cart\_update\_quantity, name='cart\_update\_quantity'),

path('cart/remove/<int:item\_id>/', views.cart\_remove, name='cart\_remove'),

path('order/create/', views.create\_order, name='create\_order'),

path('order/confirmation/<int:order\_id>/', views.order\_confirmation, name='order\_confirmation'),

path('contact/', views.contact, name='contact'),

]

{% extends 'production/base.html' %}

{% block title %}Продукция - kfxkobin.ru{% endblock %}

{% block content %}

<section id="product-list">

<h2>Наша Продукция</h2>

<div class="product-grid">

{% for product in products %}

<div class="product-card" data-product-id="{{ product.id }}">

<a href="{% url 'product\_detail' product.id %}" class="product-details-link">

<img src="{{ product.image.url }}" alt="{{ product.name }}">

<h3>{{ product.name }}</h3>

<p class="product-price">{{ product.price }} руб.</p>

<p class="product-description">{{ product.description|truncatechars:100 }}</p>

</a>

<button class="add-to-cart" data-product-id="{{ product.id }}">

<span class="btn-text">Добавить в корзину</span>

<span class="spinner" style="display: none;">Загрузка...</span>

</button>

</div>

{% endfor %}

</div>

</section>

<div id="product-modal" class="modal">

<div class="modal-content">

<span class="close">&times;</span>

<div id="modal-product-details">

</div>

</div>

</div>

<div id="notification">

<span id="notification-message"></span>

</div>

{% endblock %}

<form style="display:none;" method="post">

{% csrf\_token %}

</form>

{% block extra\_scripts %}

<script>

// Функция для получения данных о продукте

async function getProductData(productId) {

try {

const response = await fetch(`/products/${productId}/`);

if (!response.ok) {

throw new Error(`HTTP error! Status: ${response.status}`);

}

const data = await response.json();

return data;

} catch (error) {

console.error('Ошибка при получении данных о продукте:', error);

showNotification('Ошибка при загрузке данных о продукте', true);

return null;

}

}

// Добавление товара в корзину

const addToCartBtns = document.querySelectorAll('.add-to-cart');

addToCartBtns.forEach(button => {

button.addEventListener('click', function(e) {

e.preventDefault(); // Предотвращаем отправку формы, если она есть

const productId = this.dataset.productId;

let quantity = 1;

// Если это кнопка в модальном окне, берем количество из инпута

if (this.id === 'modalAddToCart') {

const quantityInput = document.querySelector('.quantity-input');

if (quantityInput) {

quantity = parseInt(quantityInput.value, 10); // Используем parseInt с основанием 10

}

const spinner = this.querySelector('.spinner');

const btnText = this.querySelector('.btn-text');

if (spinner) spinner.style.display = 'block';

if (btnText) btnText.style.display = 'none';

}

addToCart(productId, quantity);

});

});

// Функция добавления в корзину

async function addToCart(productId, quantity) {

try {

const response = await fetch(`/cart/add/${productId}/`, {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

'X-CSRFToken': getCookie('csrftoken')

},

body: new URLSearchParams({

'quantity': quantity

})

});

if (!response.ok) {

throw new Error(`HTTP error! Status: ${response.status}`);

}

const data = await response.json();

if (data.success) {

showNotification(data.message);

updateCartCounter(data.cart\_total\_items);

} else {

showNotification(data.message || 'Произошла ошибка', true);

}

} catch (error) {

console.error('Ошибка:', error);

showNotification('Произошла ошибка при добавлении в корзину', true);

} finally {

const modal = document.getElementById('product-modal');

if (modal && modal.classList.contains('active')) {

const spinner = document.querySelector('#modalAddToCart .spinner');

const btnText = document.querySelector('#modalAddToCart .btn-text');

if (spinner) spinner.style.display = 'none';

if (btnText) btnText.style.display = 'inline';

}

}

}

// Показать уведомление

const notification = document.getElementById('notification');

const notificationMessage = document.getElementById('notification-message');

function showNotification(message, isError = false) {

if (notificationMessage) notificationMessage.textContent = message;

if (notification) notification.style.backgroundColor = isError ? '#FFB7B2' : '#B5EAD7';

if (notification) notification.style.color = isError ? '#d63031' : '#846C5B';

if (notification) notification.classList.add('show');

setTimeout(() => {

if (notification) notification.classList.remove('show');

}, 3000);

}

// Обновить счетчик корзины

function updateCartCounter(count) {

const cartCounter = document.querySelector('.cart-counter');

if (cartCounter) {

cartCounter.textContent = count;

cartCounter.classList.add('animate-bounce');

setTimeout(() => cartCounter.classList.remove('animate-bounce'), 1000);

}

}

// Функция для получения CSRF-токена

function getCookie(name) {

let cookieValue = null;

if (document.cookie && document.cookie !== '') {

const cookies = document.cookie.split(';');

for (let i = 0; i < cookies.length; i++) {

let cookie = cookies[i].trim();

if (cookie.substring(0, name.length + 1) === (name + '=')) {

cookieValue = decodeURIComponent(cookie.substring(name.length + 1));

break;

}

}

}

return cookieValue;

}

// Анимации при скролле

const scrollElements = document.querySelectorAll('[data-scroll]');

const elementInView = (el) => {

const elementTop = el.getBoundingClientRect().top;

return (elementTop <= (window.innerHeight || document.documentElement.clientHeight) \* 0.75);

};

const displayScrollElement = (element) => {

element.classList.add('is-visible');

};

const handleScrollAnimation = () => {

scrollElements.forEach((el) => {

if (elementInView(el)) {

displayScrollElement(el);

}

});

};

window.addEventListener('scroll', handleScrollAnimation);

window.addEventListener('DOMContentLoaded', handleScrollAnimation);

// Открытие модального окна

const productDetailsLinks = document.querySelectorAll('.product-details-link');

const modal = document.getElementById('product-modal');

const modalContent = document.getElementById('modal-product-details');

const closeBtn = document.querySelector('.close');

productDetailsLinks.forEach(link => {

link.addEventListener('click', async function(e) {

e.preventDefault();

const productId = this.dataset.productId;

const productData = await getProductData(productId);

if (productData) {

populateModal(productData);

if (modal) modal.style.display = 'block';

}

});

});

// Закрытие модального окна

if (closeBtn) {

closeBtn.addEventListener('click', () => {

if (modal) modal.style.display = 'none';

});

}

if (modal) {

window.addEventListener('click', (e) => {

if (e.target == modal) {

modal.style.display = 'none';

}

});

}

// Заполнение модального окна данными о продукте

function populateModal(product) {

if (modalContent) {

modalContent.innerHTML = `

<h2>${product.name}</h2>

<img src="${product.image.url}" alt="${product.name}">

<p class="product-price">${product.price} руб.</p>

<p>${product.description}</p>

<ul>

<li>Вес: ${product.specs['Вес']}</li>

<li>Срок хранения: ${product.specs['Срок хранения']}</li>

<li>Упаковка: ${product.specs['Упаковка']}</li>

<li>Сорт: ${product.specs['Сорт']}</li>

</ul>

<input type="number" class="quantity-input" value="1" min="1">

<button id="modalAddToCart" data-product-id="${product.id}">

<span class="btn-text">Добавить в корзину</span>

<span class="spinner" style="display: none;">Загрузка...</span>

</button>

`;

// Добавляем обработчик событий для кнопки "Добавить в корзину" в модальном окне

const modalAddToCartBtn = document.getElementById('modalAddToCart');

if (modalAddToCartBtn) {

modalAddToCartBtn.addEventListener('click', function(e) {

e.preventDefault();

const productId = this.dataset.productId;

const quantity = parseInt(document.querySelector('.quantity-input').value, 10);

const spinner = this.querySelector('.spinner');

const btnText = this.querySelector('.btn-text');

if (spinner) spinner.style.display = 'block';

if (btnText) btnText.style.display = 'none';

addToCart(productId, quantity);

});

}

}

}

</script>

{% endblock %}

{% load static %}

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Каталог</title>

<script src="{% static 'js/cart.js' %}" defer></script>

</head>

<body>

<div class="product-card">

<img src="{{ product.image.url }}" alt="{{ product.name }}">

<h3>{{ product.name }}</h3>

<p>{{ product.description|truncatechars:100 }}</p>

<p class="price">Цена: {{ product.price }} руб.</p>

<div class="product-details">

{% if product.nutrition\_facts %}

<p><strong>Пищевая ценность:</strong> {{ product.nutrition\_facts }}</p>

{% endif %}

{% if product.storage\_conditions %}

<p><strong>Условия хранения:</strong> {{ product.storage\_conditions }}</p>

{% endif %}

{% if product.manufacturer\_info %}

<p><strong>Информация о производителе:</strong> {{ product.manufacturer\_info }}</p>

{% endif %}

</div>

<button class="add-to-cart" data-product-id="{{ product.id }}">Добавить в корзину</button>

</div>

</body>

</html>