DRILLING M7 S12 SR YURI URZUA LEBUY

EJERCICIO:

Partiendo del ejercicio creado en los Rebound y Drilling del CUE anterior, realice las siguientes pruebas de aplicación a los modelos de Fábrica y Producto:

1. Realice la prueba unitaria llamada `test_model_content_fabrica`, de verificación de datos para el modelo de Fábrica que se crean en una función `setUpTestData`.
2. Realice la prueba unitaria llamada `test_model_content_producto`, de verificación de datos para el modelo de Producto que se crean en una función `setUpTestData`.
3. Crear una prueba unitaria que verifique el código de respuesta HTTP 200 de la URL `/producto/`.
4. Realice una prueba unitaria de verificación de contenido al llamar el nombre de la UR `mostrar-pro`, verificando la respuesta HTTP 200, verificación de plantilla correcta, y conformación de contenido HTML esperado que sea coincidente con la plantilla. Por ejemplo: **"Información de Productos"**

Lo solicitado

```
(practica_orm_django) D:\0001 BOOT CAMP\BOOTCAMP\M7\SESION12\practica_orm>python manage.py test productos
Found 4 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
....
----------------------------------------------------------------------
Ran 4 tests in 0.054s

OK
Destroying test database for alias 'default'...

(practica_orm_django) D:\0001 BOOT CAMP\BOOTCAMP\M7\SESION12\practica_orm>
```

views.py | tests.py M | producto_detail.html | settings.py M | urls.py M

SESION12 > practica_orm > productos > tests.py > ...

```python
from django.test import TestCase, Client
from django.urls import reverse
from .models import Fabricante, Producto
from datetime import date

class ProductosTests(TestCase):
    @classmethod
    def setUpTestData(cls):
        # Crear datos de prueba para Fabricante
        cls.fabricante = Fabricante.objects.create(
            nombre='P&G',
            pais='Estados Unidos'
        )

        # Crear datos de prueba para Producto
        cls.producto = Producto.objects.create(
            nombre='Protex Aloe',
            descripcion='Jabón antibacterial',
            precio=2500.00,
            fabricante=cls.fabricante,
            f_vencimiento=date(2025, 12, 31)
        )

    def test_model_content_fabrica(self):
        """Test 1: Verificar datos del modelo Fábrica"""
        fabricante = Fabricante.objects.get(id=self.fabricante.id)
        self.assertEqual(fabricante.nombre, 'P&G')
        self.assertEqual(fabricante.pais, 'Estados Unidos')

    def test_model_content_producto(self):
        """Test 2: Verificar datos del modelo Producto"""
        producto = Producto.objects.get(id=self.producto.id)
        self.assertEqual(producto.nombre, 'Protex Aloe')
        self.assertEqual(producto.fabricante.nombre, 'P&G')

    def test_url_producto(self):
        """Test 3: Verificar respuesta HTTP 200 de /producto/"""
        response = self.client.get(reverse('producto-list'))
        self.assertEqual(response.status_code, 200)

    def test_url_mostrar_pro(self):
        """Test 4: Verificar contenido de mostrar-pro"""
        # Primero crear un producto de prueba
        producto = self.producto  # Usar el producto creado en setUpTestData
        response = self.client.get(reverse('mostrar-pro', kwargs={'pk': producto.pk}))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'productos/producto_detail.html')
        self.assertContains(response, 'Información de Producto')
```