Guía Profesional: Cómo Crear un eCommerce con Django REST + React

Módulo 3 (Detallado): Modelos, Serializers y API de Productos

3.1 Crear el modelo Producto

En tienda/models.py:

```
from django.db import models  # Importamos el módulo para definir modelos de base de datos

class Producto(models.Model):  # Creamos una clase que representa un producto
    nombre = models.CharField(max_length=100)  # Campo de texto corto para el nombre
    descripcion = models.TextField()  # Campo de texto largo para la descripción
        precio = models.DecimalField(max_digits=10, decimal_places=2)  # Número decimal para precios
    stock = models.PositiveIntegerField()  # Número entero positivo para el stock disponible
    creado = models.DateTimeField(auto_now_add=True)  # Fecha de creación automática

    def __str__(self):
        return self.nombre  # Representación del objeto como string (útil para el admin)
```

3.2 Crear y aplicar migraciones

```
python manage.py makemigrations  # Crea archivos de migración para reflejar los modelos en la base de datos
python manage.py migrate  # Aplica las migraciones a la base de datos
```

3.3 Crear el Serializer

En tienda/serializers.py:

```
from rest_framework import serializers  # Importamos DRF serializers
from .models import Producto  # Importamos el modelo Producto

class ProductoSerializer(serializers.ModelSerializer):  # Creamos un serializer basado en el modelo
    class Meta:
        model = Producto  # Le decimos qué modelo usar
        fields = '__all__'  # Incluimos todos los campos del modelo
```

3.4 Crear la Vista (ViewSet)

En tienda/views.py:

```
from rest_framework import viewsets  # ViewSets permiten manejar CRUD automáticamente
from .models import Producto  # Modelo
from .serializers import ProductoSerializer  # Serializer

class ProductoViewSet(viewsets.ModelViewSet):  # ViewSet con operaciones CRUD
    queryset = Producto.objects.all()  # Todos los productos de la base de datos
    serializer_class = ProductoSerializer  # Qué serializer se usa
```

3.5 Configurar las rutas

En tienda/urls.py:

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter  # Importamos router para registrar vistas automáticamente
from .views import ProductoViewSet

router = DefaultRouter()  # Creamos el router
router.register(r'productos', ProductoViewSet)  # Registramos el endpoint /productos

urlpatterns = [
    path('', include(router.urls)),  # Incluimos las rutas del router
]
```

Y en backend/urls.py agregar:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),  # Ruta para acceder al panel de administración
    path('api/', include('tienda.urls')),  # Incluye las rutas definidas en la app tienda
]
```

3.6 Probar la API

Ejecutá el servidor:

```
python manage.py runserver  # Inicia el servidor en local
```

Ingresá a: http://127.0.0.1:8000/api/productos/

Deberías ver una lista vacía si no hay productos, o datos si ya agregaste alguno desde el admin.