urls.py:

```python
from django.urls import path, include
from django.contrib.auth import views as auth_views

from . import views

urlpatterns = [
    # Páginas principales (HTML)
    path('', views.home, name='home'),
    path('login/', views.login_view, name='login'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
    path('dashboard/', views.dashboard, name='dashboard'),
    path('documentos/', views.vista_documentos, name='documentos'),
    path('documentos/cargar/', views.cargar_documento,
name='cargar_documento'),
    path('puertos/', views.vista_puertos, name='puertos'),
    path('usuarios/', views.listar_usuarios, name='usuarios'),
    path('validaciones/', views.ver_validaciones, name='validaciones'),
    #path('rutas/', views.lista_rutas_html, name='rutas'),  # Vista HTML
para rutas

    # API Puertos
    path('api/puertos/', views.puertos_api, name='puertos_api'),
    path('api/puertos/<int:id>/', views.puerto_detalle,
name='puerto_detalle'),

    # API Documentos
    path('api/documentos/subir/', views.subir_documento_api,
name='subir_documento_api'),

    # API Rutas (funciones)
    path('api/rutas/', views.rutas_api, name='rutas_api'),
    path('api/rutas/<int:id>/', views.detalle_ruta, name='detalle_ruta'),
    path('rutas/', views.rutas_view, name='rutas'),
]
```

views.py

```python
from django.shortcuts import render, redirect
from django.http import JsonResponse, HttpResponseForbidden
from django.contrib.auth import authenticate, login
from django.contrib.auth.decorators import login_required, user_passes_test
from django.contrib import messages
```

```python
from django.views.decorators.csrf import csrf_exempt
import json
from django.db import transaction
from django.shortcuts import render, redirect, get_object_or_404

from .models import DocumentoCarga, Puerto, Validacion, Ruta, PuertoRuta
from django.contrib.auth.models import User


# Decorador para controlar acceso por grupos
def grupo_requerido(*grupos):
    def en_grupo(user):
        return user.is_authenticated and
user.groups.filter(name__in=grupos).exists()
    return user_passes_test(en_grupo, login_url='login')


# ---------- Dashboard ----------
@login_required
def dashboard(request):
    user = request.user

    context = {
        'total_documentos': DocumentoCarga.objects.count(),
        'usuarios_activos': User.objects.filter(is_active=True).count(),
        'total_puertos': Puerto.objects.count(),
        'validaciones_completadas':
Validacion.objects.filter(estado='VALIDO').count()
    }

    if user.groups.filter(name="Administradores").exists():
        template = 'core/dashboard_admin.html'
    elif user.groups.filter(name="Usuario").exists():
        template = 'core/dashboard_usuario.html'
    elif user.groups.filter(name="Encargados").exists():
        template = 'core/dashboard_encargado.html'
    else:
        return HttpResponseForbidden("No tienes un rol asignado.")

    return render(request, template, context)


# ---------- API Puertos ----------
@csrf_exempt
@login_required
@grupo_requerido('Administradores')
def puertos_api(request):
    if request.method == 'GET':
```

```
        puertos = list(Puerto.objects.values())
        return JsonResponse(puertos, safe=False)

    elif request.method == 'POST':
        try:
            data = json.loads(request.body)
        except json.JSONDecodeError:
            return JsonResponse({'error': 'JSON inválido'}, status=400)

        nombre = data.get('nombre')
        pais = data.get('pais')
        direccion = data.get('direccion')
        estado = data.get('estado', True)

        if not all([nombre, pais, direccion]):
            return JsonResponse({'error': 'Datos incompletos'}, status=400)

        puerto = Puerto.objects.create(
            nombre=nombre,
            pais=pais,
            direccion=direccion,
            estado=estado
        )
        return JsonResponse({'id': puerto.id, 'mensaje': 'Puerto creado'})

    else:
        return JsonResponse({'error': 'Método no permitido'}, status=405)


@csrf_exempt
@login_required
@grupo_requerido('Administradores')
def puerto_detalle(request, id):
    try:
        puerto = Puerto.objects.get(id=id)
    except Puerto.DoesNotExist:
        return JsonResponse({'error': 'Puerto no encontrado'}, status=404)

    if request.method == 'GET':
        data = {
            'id': puerto.id,
            'nombre': puerto.nombre,
            'pais': puerto.pais,
            'direccion': puerto.direccion,
            'estado': puerto.estado,
        }
        return JsonResponse(data)
```

```
    elif request.method == 'PUT':
        try:
            data = json.loads(request.body)
        except json.JSONDecodeError:
            return JsonResponse({'error': 'JSON inválido'}, status=400)

        puerto.nombre = data.get('nombre', puerto.nombre)
        puerto.pais = data.get('pais', puerto.pais)
        puerto.direccion = data.get('direccion', puerto.direccion)
        puerto.estado = data.get('estado', puerto.estado)
        puerto.save()
        return JsonResponse({'mensaje': 'Puerto actualizado'})

    elif request.method == 'DELETE':
        puerto.delete()
        return JsonResponse({'mensaje': 'Puerto eliminado'})

    else:
        return JsonResponse({'error': 'Método no permitido'}, status=405)


# ---------- API Rutas ----------
@csrf_exempt
@login_required
@grupo_requerido('Administradores')
def rutas_api(request):
    if request.method == 'GET':
        rutas = list(Ruta.objects.values())
        return JsonResponse(rutas, safe=False)

    elif request.method == 'POST':
        try:
            data = json.loads(request.body)
        except json.JSONDecodeError:
            return JsonResponse({'error': 'JSON inválido'}, status=400)

        nombre = data.get('nombre')
        if not nombre:
            return JsonResponse({'error': 'El nombre es obligatorio'},
status=400)

        descripcion = data.get('descripcion', '')
        ruta = Ruta.objects.create(nombre=nombre, descripcion=descripcion)
        return JsonResponse({'id': ruta.id, 'mensaje': 'Ruta creada'})

    else:
        return JsonResponse({'error': 'Método no permitido'}, status=405)
```

```python
@csrf_exempt
@login_required
@grupo_requerido('Administradores')
def detalle_ruta(request, id):
    try:
        ruta = Ruta.objects.get(id=id)
    except Ruta.DoesNotExist:
        return JsonResponse({'error': 'Ruta no encontrada'}, status=404)

    if request.method == 'GET':
        puertos_ruta =
PuertoRuta.objects.filter(ruta=ruta).order_by('orden')
        data_puertos = [{'id': pr.puerto.id, 'nombre': pr.puerto.nombre,
'orden': pr.orden} for pr in puertos_ruta]

        data = {
            'id': ruta.id,
            'nombre': ruta.nombre,
            'descripcion': ruta.descripcion,
            'estado': ruta.estado,
            'puertos': data_puertos
        }
        return JsonResponse(data)

    elif request.method == 'PUT':
        try:
            data = json.loads(request.body)
        except json.JSONDecodeError:
            return JsonResponse({'error': 'JSON inválido'}, status=400)

        ruta.nombre = data.get('nombre', ruta.nombre)
        ruta.descripcion = data.get('descripcion', ruta.descripcion)
        ruta.estado = data.get('estado', ruta.estado)
        ruta.save()
        return JsonResponse({'mensaje': 'Ruta actualizada'})

    elif request.method == 'DELETE':
        ruta.delete()
        return JsonResponse({'mensaje': 'Ruta eliminada'})

    else:
        return JsonResponse({'error': 'Método no permitido'}, status=405)


# ---------- Vista para listar rutas con puertos ----------
@login_required
@grupo_requerido('Administradores')
```

```python
def lista_rutas(request):
    rutas = Ruta.objects.all()
    rutas_con_puertos = []
    for ruta in rutas:
        puertos = PuertoRuta.objects.filter(ruta=ruta).order_by('orden')
        rutas_con_puertos.append({
            'ruta': ruta,
            'puertos': [pr.puerto for pr in puertos],
        })
    return render(request, 'core/lista_rutas.html', {'rutas_con_puertos':
rutas_con_puertos})


# ---------- Subir documento ----------
@login_required
@csrf_exempt
@grupo_requerido('Administradores', 'Usuario')
def subir_documento_api(request):
    if request.method == 'POST':
        tipo = request.POST.get('tipo')
        puerto_id = request.POST.get('puerto')
        fecha = request.POST.get('fecha')
        observaciones = request.POST.get('observaciones', '')
        archivo = request.FILES.get('archivo_pdf')

        if not all([tipo, puerto_id, fecha, archivo]):
            return JsonResponse({'error': 'Datos incompletos.'}, status=400)

        try:
            puerto = Puerto.objects.get(id=puerto_id)
        except Puerto.DoesNotExist:
            return JsonResponse({'error': 'Puerto no válido.'}, status=404)

        DocumentoCarga.objects.create(
            tipo=tipo,
            puerto=puerto,
            fecha=fecha,
            archivo_pdf=archivo,
            observaciones=observaciones,
            creado_por=request.user
        )
        return JsonResponse({'mensaje': 'Documento guardado exitosamente'})

    return JsonResponse({'error': 'Método no permitido'}, status=405)


# ---------- Vista para renderizar rutas y puertos para frontend ----------
@login_required
```

```python
@grupo_requerido('Administradores')
def rutas_view(request):
    puertos = Puerto.objects.filter(estado=True)
    rutas = Ruta.objects.all().prefetch_related('puertoruta_set__puerto')

    if request.method == 'POST':
        nombre = request.POST.get('nombre')
        codigo = request.POST.get('codigo')
        descripcion = request.POST.get('descripcion')
        estado = request.POST.get('estado') == 'active'
        puertos_ids = request.POST.getlist('puertos[]')

        if not nombre or not codigo:
            messages.error(request, "Nombre y código son obligatorios.")
            return redirect('rutas')

        with transaction.atomic():
            ruta = Ruta.objects.create(
                nombre=nombre,
                codigo=codigo,
                descripcion=descripcion,
                estado=estado
            )
            for orden, pid in enumerate(puertos_ids, start=1):
                puerto = get_object_or_404(Puerto, id=pid)
                PuertoRuta.objects.create(ruta=ruta, puerto=puerto,
orden=orden)

        messages.success(request, f"Ruta '{ruta.nombre}' creada
correctamente.")
        return redirect('rutas')

    context = {
        'rutas': rutas,
        'puertos': puertos,
    }
    return render(request, 'core/rutas.html', context)


# ---------- Vistas HTML protegidas ----------
@login_required
@grupo_requerido('Administradores')
def vista_puertos(request):
    return render(request, 'core/puertos.html')


@login_required
@grupo_requerido('Administradores')
```

```python
def listar_usuarios(request):
    return render(request, 'core/usuarios.html')


@login_required
@grupo_requerido('Administradores', 'Encargados')
def ver_validaciones(request):
    return render(request, 'core/validaciones.html')


@login_required
@grupo_requerido('Administradores', 'Usuario')
def cargar_documento(request):
    return render(request, 'core/cargar_documento.html')


@login_required
@grupo_requerido('Administradores', 'Usuario', 'Encargados')
def vista_documentos(request):
    return render(request, 'core/documentos.html')


@login_required
@grupo_requerido('Administradores')
def lista_rutas_html(request):
    return render(request, 'core/rutas.html')


def home(request):
    return render(request, 'core/home.html')


def login_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')

        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('dashboard')
        else:
            messages.error(request, "Usuario o contraseña incorrectos.")

    return render(request, 'core/login.html')
```

models.py

Python

```python
from django.db import models
from django.contrib.auth.models import User

# Modelo para los puertos donde se desembarcan
class Puerto(models.Model):
    pais = models.CharField(max_length=100)
    nombre = models.CharField(max_length=100)
    direccion = models.CharField(max_length=255)
    estado = models.BooleanField(default=True)  # True = activo, False = inactivo

    def __str__(self):
        return f"{self.nombre}, {self.pais} ({'Activo' if self.estado else 'Inactivo'})"

# Modelo para rutas de navegación
class Ruta(models.Model):
    nombre = models.CharField(max_length=100, unique=True)
    codigo = models.CharField(max_length=11, unique=True)
    descripcion = models.TextField(blank=True)
    estado = models.BooleanField(default=True)  # True = Activa, False = Inactiva


# Modelo intermedio para relacionar puertos con rutas y mantener el orden
class PuertoRuta(models.Model):
    ruta = models.ForeignKey(Ruta, on_delete=models.CASCADE)
    puerto = models.ForeignKey(Puerto, on_delete=models.CASCADE)
    orden = models.PositiveIntegerField()  # Indica el orden del puerto en la ruta

    class Meta:
        unique_together = ('ruta', 'puerto')
        ordering = ['orden']

    def __str__(self):
        return f"{self.ruta.nombre} - {self.puerto.nombre} (Orden {self.orden})"

# Perfil extendido del usuario con rol explícito
class PerfilUsuario(models.Model):
    ROLES = [
        ('ADMIN', 'Administrador'),
        ('GESTOR', 'Gestor de Documentos'),
        ('ENCARGADO', 'Encargado de Puerto'),
    ]
```

```
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    puerto_asignado = models.ForeignKey(Puerto, null=True, blank=True,
on_delete=models.SET_NULL)
    cargo = models.CharField(max_length=100, blank=True)
    rol = models.CharField(max_length=10, choices=ROLES,
default='ENCARGADO')

    def __str__(self):
        return f"{self.user.username} ({self.get_rol_display()})"

# Modelo para los documentos de carga
class DocumentoCarga(models.Model):
    TIPO_CHOICES = [
        ('FACTURA', 'Factura de Desembarco'),
        ('MANIFIESTO', 'Manifiesto'),
        ('CERTIFICADO', 'Certificado de Origen'),
        ('GUIA', 'Guía de Carga'),
        ('INSPECCION', 'Informe de Inspección'),
        ('PERMISO', 'Permiso o Licencia Especial'),
    ]

    tipo = models.CharField(max_length=20, choices=TIPO_CHOICES)
    puerto = models.ForeignKey(Puerto, on_delete=models.CASCADE)
    ruta = models.ForeignKey(Ruta, on_delete=models.CASCADE, null=True,
blank=True)
    fecha = models.DateField()
    archivo_pdf = models.FileField(upload_to='documentos/')
    observaciones = models.TextField(blank=True)
    creado_por = models.ForeignKey(User, on_delete=models.SET_NULL,
null=True)

    def __str__(self):
        return f"{self.tipo} - {self.puerto.nombre} - {self.fecha}"

# Validación de los documentos por los encargados del puerto
class Validacion(models.Model):
    ESTADO_CHOICES = [
        ('VALIDO', 'Válido'),
        ('RECHAZADO', 'Rechazado'),
        ('PENDIENTE', 'Pendiente'),
    ]

    documento = models.ForeignKey(DocumentoCarga, on_delete=models.CASCADE)
    usuario = models.ForeignKey(User, on_delete=models.CASCADE)
    estado = models.CharField(max_length=10, choices=ESTADO_CHOICES)
    comentario = models.TextField(blank=True)
    fecha_validacion = models.DateTimeField(auto_now_add=True)
```

```
    class Meta:
        unique_together = ('documento', 'usuario')

    def __str__(self):
        return f"{self.documento} - {self.estado} por
{self.usuario.username}"
```

rutas.html

HTML

```
{% extends "layout.html" %}
{% block title %}Dashboard Admin{% endblock %}
{% block content %}

<!-- Header -->
<header class="bg-white shadow-sm border-b">
  <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
    <div class="flex justify-between items-center py-6">
      <div class="flex items-center space-x-3">
        <i data-lucide="route" class="h-8 w-8 text-primary"></i>
        <div>
          <h1 class="text-2xl font-bold text-gray-900">Gestión de Rutas</h1>
          <p class="text-sm text-gray-600">Administración de rutas navieras
del sistema</p>
        </div>
      </div>
      <button id="btnOpenRutaModal" class="inline-flex items-center px-4
py-2 border border-transparent rounded-md shadow-sm text-sm font-medium
text-white bg-primary hover:bg-primary/90 transition-colors duration-300">
        <i data-lucide="plus" class="w-4 h-4 mr-2"></i>
        Nueva Ruta
      </button>
    </div>
  </div>
</header>

<!-- Main Content -->
<main class="max-w-7xl mx-auto py-6 px-4 sm:px-6 lg:px-8">

  <!-- Rutas Table -->
  <div class="bg-white shadow overflow-hidden sm:rounded-lg">
<table class="min-w-full divide-y divide-gray-200">
```

```
  <thead class="bg-gray-50">
    <tr>
      <th scope="col" class="px-6 py-3 text-left text-xs font-medium
text-gray-500 uppercase tracking-wider">
        Nombre de Ruta
      </th>
      <th scope="col" class="px-6 py-3 text-left text-xs font-medium
text-gray-500 uppercase tracking-wider">
        Descripción
      </th>
      <th scope="col" class="px-6 py-3 text-left text-xs font-medium
text-gray-500 uppercase tracking-wider">
        Puertos Asignados
      </th>
      <th scope="col" class="px-6 py-3 text-left text-xs font-medium
text-gray-500 uppercase tracking-wider">
        Estado
      </th>
      <th scope="col" class="px-6 py-3 text-right text-xs font-medium
text-gray-500 uppercase tracking-wider">
        Acciones
      </th>
    </tr>
  </thead>
  <tbody class="bg-white divide-y divide-gray-200" id="rutasTableBody">
    {% for ruta in rutas %}
    <tr class="hover:bg-gray-50">
      <td class="px-6 py-4 whitespace-nowrap">
        <div class="text-sm font-medium text-gray-900">{{ ruta.nombre
}}</div>
      </td>
      <td class="px-6 py-4">
        <div class="text-sm text-gray-900 max-w-xs truncate">{{
ruta.descripcion }}</div>
      </td>
      <td class="px-6 py-4 whitespace-nowrap">
        <div class="text-sm text-gray-900">
          {% with puertos=ruta.puertoruta_set.all|dictsort:"orden" %}
            {% for pr in puertos %}
              {{ pr.puerto.nombre }}{% if not forloop.last %}, {% endif %}
            {% endfor %}
          {% endwith %}
        </div>
      </td>
      <td class="px-6 py-4 whitespace-nowrap">
        {% if ruta.estado %}
        <span class="inline-flex items-center px-2.5 py-0.5 rounded-full
text-xs font-medium bg-green-100 text-green-800">Activa</span>
```

```
        {% else %}
        <span class="inline-flex items-center px-2.5 py-0.5 rounded-full
text-xs font-medium bg-gray-100 text-gray-800">Inactiva</span>
        {% endif %}
      </td>
      <td class="px-6 py-4 whitespace-nowrap text-right text-sm
font-medium">
        <button onclick="openRutaModal('edit', '{{ ruta.id }}')"
class="text-indigo-600 hover:text-indigo-900 mr-3" title="Editar">
          <i data-lucide="edit" class="h-5 w-5"></i>
        </button>
        <button onclick="deleteRuta('{{ ruta.id }}')" class="text-red-600
hover:text-red-900" title="Eliminar">
          <i data-lucide="trash-2" class="h-5 w-5"></i>
        </button>
      </td>
    </tr>
    {% empty %}
    <tr>
      <td colspan="5" class="text-center py-4 text-gray-500">No hay rutas
registradas.</td>
    </tr>
    {% endfor %}
  </tbody>
</table>

  </div>
</main>

<!-- Create/Edit Ruta Modal -->
<div id="rutaModal" class="hidden fixed inset-0 bg-gray-500 bg-opacity-75
flex items-center justify-center z-50">
  <div class="bg-white rounded-lg px-4 pt-5 pb-4 overflow-hidden shadow-xl
transform transition-all sm:max-w-4xl sm:w-full max-h-screen
overflow-y-auto">
    <div class="sm:flex sm:items-start">
      <div class="mt-3 text-center sm:mt-0 sm:text-left w-full">
        <h3 class="text-lg leading-6 font-medium text-gray-900 mb-4"
id="rutaModalTitle">Nueva Ruta</h3>

<form id="rutaForm" class="space-y-6" novalidate>
  {% csrf_token %}
  <div id="puertosInputs"></div>

  <!-- Basic Information -->
  <div class="grid grid-cols-1 md:grid-cols-2 gap-6">
    <div>
```

```
      <label for="rutaNombre" class="block text-sm font-medium
text-gray-700">Nombre de Ruta *</label>
      <input
        type="text"
        name="nombre"
        id="rutaNombre"
        required
        class="mt-1 block w-full rounded-md border-gray-300 shadow-sm
focus:border-primary focus:ring-primary sm:text-sm"
        placeholder="Ej: Ruta Mediterráneo Occidental"
      />
      <div id="nombreError" class="text-red-600 text-sm mt-1 hidden"></div>
    </div>

  </div>

  <div>
    <label for="rutaDescripcion" class="block text-sm font-medium
text-gray-700">Descripción</label>
    <textarea
      name="descripcion"
      id="rutaDescripcion"
      rows="3"
      class="mt-1 block w-full rounded-md border-gray-300 shadow-sm
focus:border-primary focus:ring-primary sm:text-sm"
      placeholder="Descripción breve de la ruta"
    ></textarea>
  </div>

  <div>
    <label for="rutaEstado" class="block text-sm font-medium
text-gray-700">Estado</label>
    <select
      name="estado"
      id="rutaEstado"
      required
      class="mt-1 block w-full rounded-md border-gray-300 shadow-sm
focus:border-primary focus:ring-primary sm:text-sm"
    >
      <option value="active">Activa</option>
      <option value="inactive">Inactiva</option>
    </select>
  </div>

  <!-- Puerto Management Section -->
  <div class="border-t pt-6">
    <h4 class="text-lg font-medium text-gray-900 mb-4 flex items-center">
      <i data-lucide="map-pin" class="h-5 w-5 mr-2"></i>
```

```
      Gestión de Puertos
    </h4>

    <!-- Add Puerto Section -->
    <div class="bg-gray-50 rounded-lg p-4 mb-4">
      <label for="puertoSelect" class="block text-sm font-medium
text-gray-700 mb-2">
        Agregar Puerto a la Ruta
      </label>
      <div class="flex space-x-3">
        <select
          id="puertoSelect"
          class="flex-1 rounded-md border-gray-300 shadow-sm
focus:border-primary focus:ring-primary sm:text-sm"
        >
          <option value="">Seleccionar puerto</option>
          {% for puerto in puertos %}
            <option value="{{ puerto.id }}">{{ puerto.nombre }}</option>
          {% endfor %}
        </select>
        <button
          type="button"
          id="btnAddPuerto"
          class="inline-flex items-center px-4 py-2 border
border-transparent rounded-md shadow-sm text-sm font-medium text-white
bg-primary hover:bg-primary/90"
        >
          <i data-lucide="plus" class="w-4 h-4 mr-2"></i>
          Agregar
        </button>
      </div>
    </div>

    <!-- Assigned Puertos List -->
    <div>
      <h5 class="text-sm font-medium text-gray-700 mb-3">Puertos
Asignados</h5>
      <div id="assignedPuertosList" class="space-y-2">
        <!-- Asignados dinámicamente -->
      </div>
      <div id="emptyPuertosState" class="text-center py-8 text-gray-500">
        <i data-lucide="map-pin" class="h-12 w-12 mx-auto mb-4
text-gray-300"></i>
        <p>No hay puertos asignados a esta ruta</p>
        <p class="text-sm">Selecciona puertos del menú desplegable para
agregarlos</p>
      </div>
      <div id="puertosError" class="text-red-600 text-sm mt-2 hidden"></div>
```

```
    </div>
  </div>

  <div id="formError" class="text-red-600 text-sm mt-4 hidden"></div>

  <div class="mt-5 sm:mt-4 sm:flex sm:flex-row-reverse">
    <button type="button" onclick="saveRuta()" class="w-full inline-flex
justify-center rounded-md border border-transparent shadow-sm px-4 py-2
bg-primary text-base font-medium text-white hover:bg-primary/90
focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-primary
sm:ml-3 sm:w-auto sm:text-sm">
      <i data-lucide="save" class="w-4 h-4 mr-2"></i>
      Guardar
    </button>
    <button id="btnCancelModal" type="button" class="mt-3 w-full inline-flex
justify-center rounded-md border border-gray-300 shadow-sm px-4 py-2
bg-white text-base font-medium text-gray-700 hover:bg-gray-50
focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-primary
sm:mt-0 sm:w-auto sm:text-sm">
      <i data-lucide="x" class="w-4 h-4 mr-2"></i>
      Cancelar
    </button>
  </div>
</form>

      </div>
    </div>
  </div>
</div>

<script>
document.addEventListener('DOMContentLoaded', () => {
  const btnOpenRutaModal = document.getElementById('btnOpenRutaModal');
  const rutaModal = document.getElementById('rutaModal');
  const btnCancelModal = document.getElementById('btnCancelModal');
  const puertoSelect = document.getElementById('puertoSelect');
  const btnAddPuerto = document.getElementById('btnAddPuerto');
  const assignedPuertosList =
document.getElementById('assignedPuertosList');
  const puertosInputs = document.getElementById('puertosInputs');
  const rutaForm = document.getElementById('rutaForm');

  let assignedPuertos = [];
  let currentRutaId = null;
  let rutas = [];

  // Mostrar modal para nueva ruta
  btnOpenRutaModal.addEventListener('click', () => {
```

```
    resetForm();
    rutaModal.classList.remove('hidden');
    document.getElementById('rutaModalTitle').textContent = 'Nueva Ruta';
  });

  // Cancelar modal
  btnCancelModal.addEventListener('click', () => {
    closeModal();
  });

  // Agregar puerto a la lista asignada
  btnAddPuerto.addEventListener('click', () => {
    const id = puertoSelect.value;
    const text = puertoSelect.options[puertoSelect.selectedIndex]?.text ||
'';

    if (!id) {
      alert('Seleccione un puerto');
      return;
    }
    if (assignedPuertos.includes(id)) {
      alert('Puerto ya asignado');
      return;
    }

    assignedPuertos.push(id);
    addPuertoToList(id, text);
    renderInputs();
  });

  // Función para añadir puerto a la lista visual
  function addPuertoToList(id, nombre) {
    const li = document.createElement('li');
    li.textContent = nombre + ' ';
    const btnRemove = document.createElement('button');
    btnRemove.type = 'button';
    btnRemove.textContent = 'Quitar';
    btnRemove.classList.add('ml-2', 'text-red-600', 'hover:text-red-900');
    btnRemove.onclick = () => {
      assignedPuertos = assignedPuertos.filter(pid => pid !== id);
      li.remove();
      renderInputs();
    };
    li.appendChild(btnRemove);
    assignedPuertosList.appendChild(li);
  }

  // Crear inputs ocultos para el formulario
```

```
function renderInputs() {
  puertosInputs.innerHTML = '';
  assignedPuertos.forEach(id => {
    const input = document.createElement('input');
    input.type = 'hidden';
    input.name = 'puertos[]';
    input.value = id;
    puertosInputs.appendChild(input);
  });
}

// Limpiar formulario y estado
function resetForm() {
  currentRutaId = null;
  assignedPuertos = [];
  rutaForm.reset();
  assignedPuertosList.innerHTML = '';
  puertosInputs.innerHTML = '';
}

// Cerrar modal y limpiar formulario
function closeModal() {
  rutaModal.classList.add('hidden');
  resetForm();
}

// Función para eliminar ruta
window.deleteRuta = function(rutaId) {
  if (!confirm('¿Estás seguro de que deseas eliminar esta ruta?')) return;

  fetch(`/api/rutas/${rutaId}/`, {
    method: 'DELETE',
    headers: {
      'X-CSRFToken': getCSRFToken()
    }
  })
  .then(res => {
    if (!res.ok) throw new Error('Error eliminando la ruta');
    alert('Ruta eliminada correctamente');
    fetchRutas();
  })
  .catch(() => alert('No se pudo eliminar la ruta'));
};

// Función para abrir modal en modo nuevo o edición
window.openRutaModal = function(mode, rutaId = null) {
  resetForm();
  if (mode === 'edit' && rutaId) {
```

```
      currentRutaId = rutaId;
      rutaModal.classList.remove('hidden');
      document.getElementById('rutaModalTitle').textContent = 'Editar Ruta';

      // Obtener datos de la ruta via API para rellenar formulario
      fetch(`/api/rutas/${rutaId}/`)
        .then(res => {
          if (!res.ok) throw new Error('Error al cargar ruta');
          return res.json();
        })
        .then(data => {
          document.getElementById('rutaNombre').value = data.nombre || '';
          document.getElementById('rutaDescripcion').value =
data.descripcion || '';
          document.getElementById('rutaEstado').value = data.estado ||
'Inactiva';

          assignedPuertos = [];
          assignedPuertosList.innerHTML = '';
          puertosInputs.innerHTML = '';

          if (data.puertos && data.puertos.length) {
            data.puertos.forEach(puerto => {
              assignedPuertos.push(String(puerto.id));
              addPuertoToList(String(puerto.id), puerto.nombre);
            });
            renderInputs();
          }
        })
        .catch(err => alert(err.message));
    } else {
      currentRutaId = null;
      rutaModal.classList.remove('hidden');
      document.getElementById('rutaModalTitle').textContent = 'Nueva Ruta';
    }
  };

  // Guardar ruta: crea o actualiza
  window.saveRuta = function() {
    const formData = {
      nombre: document.getElementById('rutaNombre').value,
      descripcion: document.getElementById('rutaDescripcion').value,
      estado: document.getElementById('rutaEstado').value,
      puertos: assignedPuertos.map(id => Number(id))
    };

    const method = currentRutaId === null ? 'POST' : 'PUT';
```

```
    const url = currentRutaId === null ? '/api/rutas/' :
`/api/rutas/${currentRutaId}/`;

    fetch(url, {
      method: method,
      headers: {
        'Content-Type': 'application/json',
        'X-CSRFToken': getCSRFToken()
      },
      body: JSON.stringify(formData)
    })
    .then(res => {
      if (!res.ok) throw new Error('Error al guardar la ruta');
      return res.json();
    })
    .then(data => {
      fetchRutas();
      closeModal();
    })
    .catch(err => alert(err.message));
  };

  // Cargar y mostrar rutas
  function fetchRutas() {
    fetch('/api/rutas/')
      .then(res => res.json())
      .then(data => {
        rutas = data;
        renderRutas();
      });
  }

  // Renderizar rutas en la tabla
  function renderRutas() {
  const tbody = document.getElementById('rutasTableBody');
  tbody.innerHTML = rutas.map(ruta => `
    <tr class="hover:bg-gray-50">
      <td class="px-6 py-4 whitespace-nowrap">
        <div class="text-sm font-medium text-gray-900">${ruta.nombre}</div>
      </td>
      <td class="px-6 py-4">
        <div class="text-sm text-gray-900 max-w-xs
truncate">${ruta.descripcion}</div>
      </td>
      <td class="px-6 py-4 whitespace-nowrap">
        <div class="text-sm text-gray-900">
          ${ruta.puertos.map(p => p.nombre).join(', ')}
        </div>
```

```
      </td>
      <td class="px-6 py-4 whitespace-nowrap">
        <span class="inline-flex items-center px-2.5 py-0.5 rounded-full
text-xs font-medium ${
          ruta.estado ? 'bg-green-100 text-green-800' : 'bg-gray-100
text-gray-800'
        }">
          ${ruta.estado ? 'Activa' : 'Inactiva'}
        </span>
      </td>
      <td class="px-6 py-4 whitespace-nowrap text-right text-sm
font-medium">
        <button onclick="openRutaModal('edit', ${ruta.id})"
class="text-indigo-600 hover:text-indigo-900 mr-3" title="Editar">
          <i data-lucide="edit" class="h-5 w-5"></i>
        </button>
        <button onclick="deleteRuta(${ruta.id})" class="text-red-600
hover:text-red-900" title="Eliminar">
          <i data-lucide="trash-2" class="h-5 w-5"></i>
        </button>
      </td>
    </tr>
  `).join('');
  lucide.createIcons();
}

// Obtener token CSRF
  function getCSRFToken() {
    const cookieValue =
document.cookie.match('(^|;)\\s*csrftoken\\s*=\\s*([^;]+)');
    return cookieValue ? cookieValue.pop() : '';
  }

  // Cargar rutas al iniciar
  fetchRutas();
});




</script>

{% endblock %}
```