**GUÍA PASO A PASO: Reestructuración Completa del Proyecto GUIA**

**ANTES DE EMPEZAR - BACKUP OBLIGATORIO**

# 1. Asegúrate de estar en el directorio correcto

cd C:\proyecto\Guia

# 2. Commit todo lo que tengas pendiente

git add .

git commit -m "Backup antes de reestructuración"

# 3. Crear branch de respaldo (opcional pero recomendado)

git branch backup-estructura-actual

**PASO 1: Crear la nueva estructura de directorios**

**1.1 Crear directorios principales**

# Desde C:\proyecto\Guia\

mkdir apps

mkdir templates

mkdir apps\\_\_init\_\_.py

mkdir apps\core

mkdir apps\core\\_\_init\_\_.py

**1.2 Crear archivos init.py necesarios**

# Crear archivo vacío

echo. > apps\\_\_init\_\_.py

echo. > apps\core\\_\_init\_\_.py

**PASO 2: Renombrar ERP → guia\_erp**

# En Windows (Command Prompt)

ren ERP guia\_erp

**PASO 3: Mover aplicación terceros**

# Mover toda la carpeta terceros dentro de apps

move terceros apps\terceros

**PASO 4: Crear estructura de templates**

**4.1 Mover templates base**

# Crear directorio templates en raíz

mkdir templates\includes

mkdir templates\terceros

# Mover base.html

move terceros\templates\base.html templates\base.html

# Mover templates específicos de terceros

move apps\terceros\templates\terceros\\* templates\terceros\

**4.2 Eliminar directorios vacíos**

# Eliminar directorios de templates que quedaron vacíos

rmdir /s apps\terceros\templates\terceros

rmdir /s apps\terceros\templates

**PASO 5: Reorganizar archivos estáticos**

**5.1 Crear nueva estructura de static**

mkdir static\css

mkdir static\js

mkdir static\images

mkdir static\js\components

mkdir static\js\terceros

# Mover archivos estáticos

move static\terceros\css\main.css static\css\main.css

move static\terceros\js\tercero\_form\_search.js static\js\terceros\tercero\_form\_search.js

move static\terceros\images\Logo.png static\images\logo.png

# Eliminar directorio vacío

rmdir /s static\terceros

**PASO 6: Actualizar archivos de configuración**

**6.1 Actualizar guia\_erp/settings.py**

**Buscar y reemplazar estas líneas:**

# CAMBIO 1: INSTALLED\_APPS (línea ~44)

# ANTES:

'terceros.apps.TercerosConfig',

# DESPUÉS:

'apps.terceros.apps.TercerosConfig',

# CAMBIO 2: ROOT\_URLCONF (línea ~68)

# ANTES:

ROOT\_URLCONF = 'ERP.urls'

# DESPUÉS:

ROOT\_URLCONF = 'guia\_erp.urls'

# CAMBIO 3: WSGI\_APPLICATION (línea ~86)

# ANTES:

WSGI\_APPLICATION = 'ERP.wsgi.application'

# DESPUÉS:

WSGI\_APPLICATION = 'guia\_erp.wsgi.application'

**6.2 Actualizar guia\_erp/wsgi.py**

# Buscar y reemplazar:

# ANTES:

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'ERP.settings')

# DESPUÉS:

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'guia\_erp.settings')

**6.3 Actualizar guia\_erp/asgi.py**

# Buscar y reemplazar:

# ANTES:

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'ERP.settings')

# DESPUÉS:

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'guia\_erp.settings')

**6.4 Actualizar manage.py**

# Buscar y reemplazar:

# ANTES:

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'ERP.settings')

# DESPUÉS:

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'guia\_erp.settings')

**PASO 7: Actualizar configuración de la app terceros**

**7.1 Actualizar apps/terceros/apps.py**

# Buscar y reemplazar:

# ANTES:

class TercerosConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'terceros'

# DESPUÉS:

class TercerosConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'apps.terceros'

**PASO 8: Actualizar imports en guia\_erp/urls.py**

# Buscar y reemplazar:

# ANTES:

from terceros import views as terceros\_views

urlpatterns = [

path('', terceros\_views.dashboard\_view, name='dashboard'),

path('info/', terceros\_views.landing\_page\_view, name='landing\_page'),

path('terceros/', include('terceros.urls', namespace='terceros')),

path('admin/', admin.site.urls),

]

# DESPUÉS:

from apps.terceros import views as terceros\_views

urlpatterns = [

path('', terceros\_views.dashboard\_view, name='dashboard'),

path('info/', terceros\_views.landing\_page\_view, name='landing\_page'),

path('terceros/', include('apps.terceros.urls', namespace='terceros')),

path('admin/', admin.site.urls),

]

**PASO 9: Actualizar referencias en templates**

**9.1 Actualizar templates/base.html**

<!-- Buscar y reemplazar: -->

<!-- ANTES: -->

<link rel="stylesheet" href="{% static 'terceros/css/main.css' %}?v={% now 'U' %}">

<!-- DESPUÉS: -->

<link rel="stylesheet" href="{% static 'css/main.css' %}?v={% now 'U' %}">

<!-- ANTES: -->

<img src="{% static 'terceros/images/logo.png' %}" alt="Logo GUIA" class="sidebar-logo">

<!-- DESPUÉS: -->

<img src="{% static 'images/logo.png' %}" alt="Logo GUIA" class="sidebar-logo">

**9.2 Actualizar templates/terceros/tercero\_form.html**

<!-- Buscar y reemplazar al final del archivo: -->

<!-- ANTES: -->

<script src="{% static 'terceros/js/tercero\_form\_search.js' %}?v={% now 'U' %}" defer></script>

<!-- DESPUÉS: -->

<script src="{% static 'js/terceros/tercero\_form\_search.js' %}?v={% now 'U' %}" defer></script>

**PASO 10: Crear archivos base para escalabilidad**

**10.1 Crear apps/core/models.py**

# Crear archivo: apps/core/models.py

from django.db import models

from django.utils.translation import gettext\_lazy as \_

class TimeStampedModel(models.Model):

"""

Modelo abstracto que proporciona campos de auditoría

para todas las entidades del sistema.

"""

fecha\_creacion = models.DateTimeField(

auto\_now\_add=True,

verbose\_name=\_('Fecha de Creación')

)

fecha\_modificacion = models.DateTimeField(

auto\_now=True,

verbose\_name=\_('Fecha de Modificación')

)

class Meta:

abstract = True

class SoftDeleteModel(models.Model):

"""

Modelo abstracto para implementar eliminación suave (soft delete)

"""

activo = models.BooleanField(default=True, verbose\_name=\_('Activo'))

class Meta:

abstract = True

**10.2 Crear apps/core/utils.py**

# Crear archivo: apps/core/utils.py

import logging

import requests

from typing import List, Dict, Any

from django.conf import settings

logger = logging.getLogger(\_\_name\_\_)

def consultar\_api\_externa(url: str, timeout: int = 10) -> List[Dict[str, Any]]:

"""

Función auxiliar reutilizable para consultar APIs externas.

Maneja timeouts, rate limits, y otros errores de forma robusta.

"""

try:

response = requests.get(url, timeout=timeout)

if response.status\_code == 429:

logger.warning("Límite de peticiones excedido. URL: %s", url)

return []

response.raise\_for\_status()

return response.json().get('geonames', [])

except requests.Timeout:

logger.error("Timeout al conectar con API. URL: %s", url)

except requests.RequestException as e:

logger.error("Error de red: %s. URL: %s", e, url)

return []

**10.3 Crear templates/includes/messages.html**

<!-- Crear archivo: templates/includes/messages.html -->

{% if messages %}

{% for message in messages %}

<div class="alert {% if message.tags == 'success' %}alert-success{% elif message.tags == 'error' %}alert-danger{% else %}alert-{{ message.tags }}{% endif %} alert-dismissible fade show mb-3" role="alert">

<i class="fas {% if message.tags == 'success' %}fa-check-circle{% elif message.tags == 'error' %}fa-exclamation-triangle{% else %}fa-info-circle{% endif %} me-2"></i>

{{ message }}

<button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>

</div>

{% endfor %}

{% endif %}

**PASO 11: Verificación y testing**

**11.1 Verificar que el servidor funciona**

python manage.py runserver

**11.2 Verificar URLs principales**

* http://127.0.0.1:8000/ (Dashboard)
* http://127.0.0.1:8000/terceros/crear/ (Formulario)
* http://127.0.0.1:8000/admin/ (Admin)

**11.3 Verificar static files**

* CSS se carga correctamente
* JavaScript funciona
* Logo se muestra

**PASO 12: Limpiar y commit**

**12.1 Limpiar archivos temporales**

# Limpiar cache de Python

del /s /q \_\_pycache\_\_

del /s /q \*.pyc

**12.2 Commit final**

git add .

git commit -m "Reestructuración completa: apps/, templates globales, static centralizado"

**ESTRUCTURA FINAL RESULTANTE:**

C:\proyecto\Guia\

├── .git, .venv, .idea/

├── manage.py, db.sqlite3, requirements.txt

├── .env, .gitignore

│

├── guia\_erp/ # Configuración principal

│ ├── \_\_init\_\_.py

│ ├── settings.py

│ ├── urls.py

│ ├── wsgi.py

│ └── asgi.py

│

├── apps/ # Todas las aplicaciones

│ ├── \_\_init\_\_.py

│ ├── core/ # Código compartido

│ │ ├── \_\_init\_\_.py

│ │ ├── models.py # Modelos base abstractos

│ │ └── utils.py # Utilidades reutilizables

│ └── terceros/ # App terceros reubicada

│ ├── \_\_init\_\_.py

│ ├── models.py, views.py, forms.py

│ ├── urls.py, admin.py, apps.py

│ └── migrations/

│

├── templates/ # Templates globales

│ ├── base.html # Template base

│ ├── includes/

│ │ └── messages.html # Componente de mensajes

│ └── terceros/

│ ├── tercero\_form.html

│ ├── dashboard.html

│ └── landing\_page.html

│

├── static/ # Assets centralizados

│ ├── css/

│ │ └── main.css

│ ├── js/

│ │ ├── components/ # JS reutilizable

│ │ └── terceros/

│ │ └── tercero\_form\_search.js

│ └── images/

│ └── logo.png

│

├── media/ # Archivos subidos (crear después)

└── staticfiles/ # Collectstatic (producción)

**🎉 BENEFICIOS OBTENIDOS:**

1. **✅ Estructura escalable** para múltiples módulos
2. **✅ Templates reutilizables** entre apps
3. **✅ Assets centralizados** y organizados
4. **✅ Código base compartido** en core/
5. **✅ Convenciones Django estándar**
6. **✅ Preparado para nuevos desarrolladores**

**Tiempo estimado total: 2-3 horas** **Ahorro futuro: 20+ horas en próximos módulos**

¿Algún paso no está claro o necesitas que detalle algo específico?