# GUÍA COMPLETA: DESPLEGAR BACKEND PYTHON EN RENDER.COM

**Proyecto:** ImageProcessor - TechResources

**Objetivo:** Conectar backend Python Flask con frontend React en Cloudflare

**Tiempo estimado:** 20-30 minutos

---

## PASO 1: PREPARAR BACKEND PARA PRODUCCIÓN

### 1.1 Crear requirements.txt

En tu carpeta `backend/`, crea el archivo `requirements.txt` con este contenido:

```
Flask==3.0.0
flask-cors==4.0.0
Pillow==10.1.0
rembg==2.0.50
gunicorn==21.2.0
werkzeug==3.0.1
```

### 1.2 Modificar app.py

Al **final** de tu archivo `app.py`, busca este código:

```python
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Y cámbialo por:

```python
if __name__ == '__main__':
    port = int(os.environ.get('PORT', 5000))
    app.run(debug=False, host='0.0.0.0', port=port)
```

**IMPORTANTE:** Asegúrate de tener `import os` al inicio del archivo.

### 1.3 Crear .gitignore

En la carpeta `backend/`, crea el archivo `.gitignore`:

```
uploads/
*.pyc
__pycache__/
venv/
.env
*.log
```

---

# PASO 2: SUBIR BACKEND A GITHUB

## 2.1 Crear repositorio en GitHub

1. Ve a **https://github.com/new**
2. Nombre del repositorio: `imageprocessor-backend`
3. **NO marques** "Add a README"
4. Click en **"Create repository"**

## 2.2 Subir tu código

Abre la terminal en tu carpeta `backend/` y ejecuta estos comandos:

```bash
git init
git add .
git commit -m "Initial backend commit"
git branch -M main
git remote add origin https://github.com/TU-USUARIO/imageprocessor-backend.git
git push -u origin main
```

**NOTA:** Reemplaza `TU-USUARIO` con tu nombre de usuario de GitHub.

---

# PASO 3: DESPLEGAR EN RENDER.COM

## 3.1 Crear cuenta en Render

1. Ve a **https://render.com**
2. Click en **"Get Started"**
3. Regístrate usando tu cuenta de **GitHub** (opción recomendada)
4. Autoriza a Render para acceder a tus repositorios

### 3.2 Crear Web Service

1. En el dashboard de Render, click en **"New +"**
2. Selecciona **"Web Service"**
3. Click en **"Connect a repository"**
4. Busca y selecciona `imageprocessor-backend`
5. Click en **"Connect"**

### 3.3 Configurar el servicio

Llena los campos con estos valores exactos:

| Campo | Valor |
|---|---|
| **Name** | `imageprocessor-backend` |
| **Region** | Oregon (US West) |
| **Branch** | `main` |
| **Root Directory** | *(dejar vacío)* |
| **Runtime** | Python 3 |
| **Build Command** | `pip install -r requirements.txt` |
| **Start Command** | `gunicorn app:app` |
| **Plan** | Free |

### 3.4 Variables de entorno

Por ahora, deja esta sección vacía. No necesitas configurar nada adicional.

### 3.5 Iniciar el despliegue

1. Scroll hasta el final de la página
2. Click en **"Create Web Service"**
3. Render comenzará a construir tu aplicación

**TIEMPO DE ESPERA:** 5-10 minutos

El proceso incluye:

- Instalación de dependencias Python
- Descarga del modelo de IA para rembg (350MB)
- Compilación de librerías

### 3.6 Obtener tu URL

Una vez que el despliegue termine, verás **"Live"** en verde.

Tu URL será similar a:

```
https://imageprocessor-backend-abc123.onrender.com
```

**PRUEBA QUE FUNCIONE:** Visita esta URL en tu navegador:

```
https://tu-url.onrender.com/api/health
```

Deberías ver una respuesta JSON como:

```json
{
  "status": "ok",
  "message": "ImageProcessor Backend funcionando",
  "rembg_available": true,
  ...
}
```

---

# PASO 4: CONECTAR FRONTEND CON BACKEND

## 4.1 Actualizar código React

En tu archivo `ImageProcessor.jsx`, busca esta línea:

```javascript
const API_BASE_URL = 'http://localhost:5000/api';
```

Y reemplázala con:

```javascript
const API_BASE_URL = import.meta.env.MODE === 'production'
  ? 'https://imageprocessor-backend-abc123.onrender.com/api'
  : 'http://localhost:5000/api';
```

**IMPORTANTE:** Reemplaza `imageprocessor-backend-abc123.onrender.com` con tu URL real de Render (sin el `/api` al final en la primera línea).

**Ejemplo real:**

```javascript
```

```
const API_BASE_URL = import.meta.env.MODE === 'production'
  ? 'https://imageprocessor-backend-xyz789.onrender.com/api'
  : 'http://localhost:5000/api';
```

### 4.2 Hacer commit y push

Desde la carpeta de tu frontend, ejecuta:

```bash
git add .
git commit -m "Update API URL for production"
git push
```

Cloudflare Pages detectará el cambio automáticamente y redespleará tu frontend (toma 2-3 minutos).

---

## PASO 5: PROBAR APLICACIÓN COMPLETA

### Prueba final paso a paso

1. Ve a: **https://frontend-5n4.pages.dev/**
2. Sube una imagen (arrastra o selecciona)
3. Activa algún switch:
   - **Eliminar Fondo** o
   - **Redimensionar** (con dimensiones)
4. Click en **"Procesar Imágenes"**
5. Espera a que termine el procesamiento
6. Click en **"Descargar"**

**SI TODO FUNCIONA:** ¡Tu aplicación está 100% en la nube!

---

## PROBLEMAS COMUNES Y SOLUCIONES

### Error: "CORS policy"

**Causa:** Backend no permite peticiones desde tu frontend.

**Solución:** Verifica que en `app.py` tengas:

```python
```

```
from flask_cors import CORS
CORS(app)
```

---

### Error: "503 Service Unavailable"

**Causa:** El plan gratuito de Render "duerme" después de 15 minutos sin uso.

**Solución:** Este es el comportamiento normal. La primera petición después de dormir tarda 30-50 segundos en "despertar" el servicio. Peticiones subsecuentes son rápidas.

---

### Error al procesar imágenes grandes

**Causa:** El plan gratuito de Render tiene límites de memoria (512MB).

**Solución:**

- Procesa menos imágenes a la vez (máximo 10-20)
- Usa imágenes más pequeñas
- Considera upgrade a plan pagado si necesitas más capacidad

---

### El backend tarda mucho la primera vez

**Causa:** La primera vez que se usa `rembg`, descarga el modelo de IA (350MB).

**Solución:** Esto es normal. Después de la primera ejecución, el modelo queda en caché y el procesamiento es mucho más rápido.

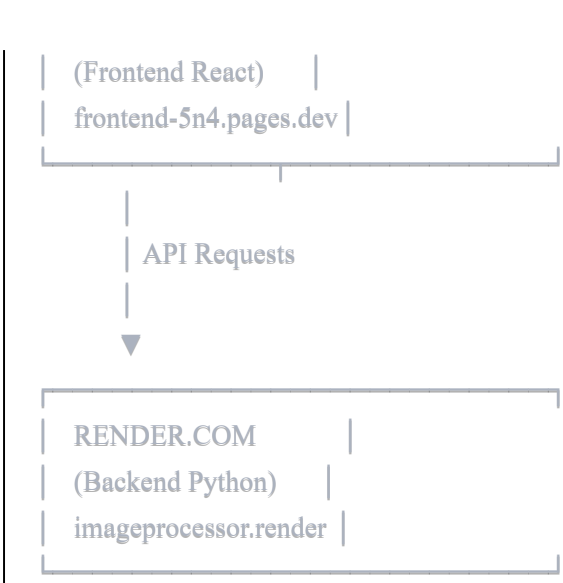
---

## RESUMEN DE URLS FINALES

| Componente | URL |
|---|---|
| **Frontend** | https://frontend-5n4.pages.dev/ |
| **Backend** | https://TU-APP.onrender.com/ |
| **API Health** | https://TU-APP.onrender.com/api/health |

---

## ARQUITECTURA FINAL




---

## NOTAS IMPORTANTES

**Plan Gratuito de Render incluye:**

- 750 horas de servicio por mes
- 512MB de RAM
- Auto-sleep después de 15 minutos de inactividad
- Almacenamiento temporal (no persistente)

**Limitaciones a considerar:**

- Primera petición después de sleep tarda ~30-50 segundos
- Procesamiento de imágenes muy grandes puede fallar
- El servicio se reinicia periódicamente (archivos en `/uploads` se borran)

---

## ¿NECESITAS AYUDA?

Si encuentras errores no cubiertos en esta guía:

1. Revisa los logs en el dashboard de Render
2. Verifica que todas las URLs estén correctas
3. Confirma que el frontend está usando el modo producción
4. Prueba el endpoint `/api/health` directamente

---

**Creado para:** ImageProcessor - TechResources
**Fecha:** Octubre 2025
**Versión:** 1.0