**Implementación de HTTPS en servidor Windows para conectar GitHub Pages con API Python - Resumen de Pasos**  
ID: efd57003...

**Objetivo del Proyecto**

Resolver el problema de GitHub Pages bloqueando peticiones HTTP desde una aplicación web a un servidor Windows en AWS, configurando HTTPS con un certificado SSL válido en el servidor para permitir comunicación segura entre ambos sistemas.

**Situación Inicial**

* Aplicación web alojada en GitHub Pages
* Servidor Windows en Amazon AWS con una API Python (Flask) ejecutándose en el puerto 5000
* GitHub Pages bloqueaba las peticiones por no ser HTTPS

**Pasos Realizados**

**1. Obtención de Certificado SSL**

* Adquirimos un dominio en GoDaddy (menuidac.com)
* Configuramos DNS en AWS Route 53 para que el dominio apunte a la IP del servidor Windows
* Instalamos Win-ACME (cliente Let's Encrypt para Windows)
* Obtuvimos un certificado SSL gratuito de Let's Encrypt para el dominio

**2. Configuración de IIS como Servidor Web**

* Configuramos un sitio web en IIS con el nombre de host menuidac.com
* Instalamos el certificado SSL en el Windows Certificate Store (WebHosting)
* Configuramos el enlace HTTPS (puerto 443) en IIS
* Configuramos la renovación automática del certificado

**3. Configuración de Proxy Inverso**

* Instalamos los módulos URL Rewrite y Application Request Routing (ARR) en IIS
* Habilitamos el proxy en la configuración del servidor
* Creamos una regla de URL Rewrite para redirigir las peticiones HTTPS a la API Python local:
  + Pattern: ^api/(.\*)
  + Action: Rewrite
  + URL destino: [http://localhost:5000/api/{R:1}](http://localhost:5000/api/%7BR:1%7D)
* Esto permite que IIS reciba peticiones HTTPS y las redirija a la API Python en el puerto 5000

**4. Modificación del Código Python**

* Actualizamos todas las URL en el código de la API Python:
  + Cambiamos "<http://44.201.81.192:5000>" a "<https://menuidac.com>"
  + Mantuvimos el puerto 5000 para la ejecución local del servidor Python
  + Actualizamos las referencias en los mensajes de inicio y URLs generadas para QR

**5. Modificación del Código Unity**

* Actualizamos las URL en el código de Unity:
  + Cambiamos la URL de la API: "<https://menuidac.com:5000/api/procesar>" a "<https://menuidac.com/api/procesar>"
  + Cambiamos la URL para abrir documentos, quitando el puerto y usando el dominio

**Flujo Final de Comunicación**

1. Cliente accede a la aplicación web en GitHub Pages
2. La aplicación hace peticiones HTTPS a <https://menuidac.com/api/>...
3. IIS recibe las peticiones en el puerto 443 (HTTPS)
4. IIS redirige internamente a la aplicación Python en el puerto 5000
5. La aplicación Python procesa la petición y devuelve una respuesta
6. IIS envía la respuesta encriptada al cliente

**Beneficios Obtenidos**

* Comunicación segura (HTTPS) entre GitHub Pages y el servidor
* GitHub Pages ya no bloquea las peticiones al servidor
* Certificado SSL válido y gratuito con renovación automática
* Implementación profesional que cumple con estándares de seguridad web actuales

**Comandos y Herramientas Clave**

* Win-ACME: Para gestionar certificados SSL
* IIS Manager: Para configurar el servidor web y proxy inverso
* URL Rewrite y Application Request Routing: Para implementar el proxy inverso

Este documento proporciona una guía resumida para implementar una solución similar en futuros proyectos que requieran conexiones seguras entre GitHub Pages (o cualquier host que requiera HTTPS) y un servidor Windows.