UNIVERSIDAD PERUANA LOS ANDES
FACULTAD DE INGENIERÍA
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

VEGA BRAÑEZ SAMUEL MAX

Nombre del docente: Mg. Raul Bejarano Fernandez

HUANCAYO-PERÙ

2025

Guía Práctica: Subir y Publicar una Página Web en un Repositorio de GitHub

¡Hola! Si eres estudiante universitario en carreras como Ingeniería, Diseño, Informática o cualquier área donde manejes proyectos digitales, esta guía te va a ahorrar tiempo y te ayudará a construir un portafolio sólido. GitHub no solo es para código: es ideal para alojar sitios web estáticos (HTML, CSS, JS) y compartirlos con profesores, compañeros o futuros empleadores. Vamos a cubrir cómo subir archivos de una página web a un repositorio y activar GitHub Pages para que tu sitio esté online en minutos. Es un proceso eficiente, y al final tendrás un enlace público para tu CV.

Esta guía es paso a paso, asumiendo que tienes conocimientos básicos de web (si no, no hay problema: explico lo esencial). Puedes copiar este contenido a Google Docs, Word o Notion y exportarlo como PDF para tenerlo como referencia rápida. Si usas VS Code o similar, ¡mejor aún!

Introducción

Subir una página web a GitHub implica cargar archivos locales a un repositorio y, opcionalmente, desplegarla con GitHub Pages (gratuito para repos públicos). Es perfecto para prototipos de apps web, landing pages de proyectos universitarios o blogs personales. Beneficios: versionado automático, colaboración en equipo y hosting sin costo. Tiempo estimado: 5-15 minutos para la primera vez.

Requisitos Previos

- Cuenta en GitHub verificada (si no la tienes, revisa la guía anterior de creación).
- Un repositorio existente (público para Pages gratuito). Crea uno si es necesario.
- Archivos de tu página web preparados:
    - `index.html` como entrada principal (puedes generarlo con editores como VS Code, Sublime o incluso Notepad++).
    - Archivos complementarios: CSS (ej: `styles.css`), JS (ej: `app.js`), assets (imágenes, fonts en carpetas como `/assets` o `/img`).

- Ejemplo básico de `index.html` para probar:

```
html13 lines
Copy code
Download code
Click to close
<!DOCTYPE html>
<html lang="es">
...
```

- Navegador web (Chrome/Firefox recomendado) y conexión estable.
- Opcional: Git instalado (para métodos avanzados; descarga de git-scm.com).

Pasos Detallados para Subir y Publicar

1. Inicia Sesión en GitHub
    - Abre tu navegador y ve a github.com.
    - Haz clic en Sign in (arriba derecha). Ingresa tu username/email y contraseña.
    - Si tienes 2FA activado (recomendado para seguridad), ingresa el código de tu app (ej: Google Authenticator).
    - Si olvidas la contraseña, usa Forgot password? y sigue el flujo por email. Una vez dentro, estás listo para trabajar.
2. Navega a tu Repositorio
    - En la barra superior, haz clic en tu avatar (foto de perfil) > Your repositories.
    - Selecciona el repositorio objetivo (ej: "proyecto-web-universitario"). Si es nuevo:
        - Clic en el ícono + (arriba derecha) > New repository.
        - Nombre: algo descriptivo como "mi-portafolio-web".
        - Marca Public (para Pages gratuito; privado requiere GitHub Pro).
        - Opcional: Agrega un README.md inicial y un .gitignore (elige template para "HTML" o "Node").
        - Clic en Create repository. Ahora accede a él.
3. Sube los Archivos (Método Web para Rápido Inicio)
    - En la página principal del repo, haz clic en Add file > Upload files.
    - Selecciona o arrastra tus archivos/carpetas desde tu PC:
        - Prioriza `index.html` en la raíz.
        - Incluye CSS/JS en la raíz o subcarpetas (ej: `/css/styles.css`).

        - Límite: 100MB por archivo; para más grandes, usa Git LFS (avanzado).
    - En la sección Commit changes:
        - Mensaje de commit: Sé descriptivo, ej: "Inicial: Agregando estructura HTML/CSS/JS para landing page".
        - Selecciona Commit directly to the main branch (para simplicidad; usa branches para features complejas).
    - Clic en Commit changes. GitHub procesa y actualiza el repo en segundos.
    - Tip Universitario: Usa commits frecuentes con mensajes claros (ej: "feat: agregar responsive design") para rastrear progreso en proyectos grupales.
4. Verifica la Subida
    - Refresca la página del repo (F5 o Ctrl+R). Deberías ver tus archivos en la lista.
    - Clic en `index.html` para preview: GitHub renderiza HTML/CSS/JS en el navegador.
    - Si hay errores (ej: CSS no carga), revisa rutas relativas (ej: `<link href="./css/styles.css">`). Edita directamente en GitHub clicando el ícono de lápiz y commitando cambios.
5. Despliega con GitHub Pages (Hazlo Público)
    - Ve a la pestaña Settings (arriba del repo).
    - En el menú izquierdo, selecciona Pages (bajo "Code and automation").
    - En Source: Elige Deploy from a branch.
        - Branch: main (o master si es legacy).
        - Folder: / (root) para archivos en raíz, o /docs si usas esa convención.
    - Clic en Save. GitHub genera el sitio automáticamente.
    - En 1-5 minutos, verás el enlace en la misma sección: `https://tuusername.github.io/nombre-repo`.
    - Prueba abriendo el enlace en una pestaña nueva. Si no carga, verifica: repo público, `index.html` presente, y espera (puede tardar hasta 10 min en propagación DNS).
    - Personalización: Para dominios custom (ej: miuniversidad.com), agrega un CNAME en Settings > Pages (requiere DNS externo).
6. Actualizaciones y Mantenimiento
    - Para cambios: Repite paso 3 (web) o usa Git CLI para eficiencia:
        - En terminal: `git clone https://github.com/tuuser/repo.git` > `cd repo` > copia archivos > `git add .` > `git commit -m "Update: fix bug en JS"` > `git push`.
        - Pages se actualiza automáticamente al pushear a main.

    - Colaboración: Invita coautores en Settings > Collaborators para proyectos en equipo.

Consejos Prácticos para Universitarios

- Mejores Prácticas:
    - Estructura: Raíz con `index.html`, carpetas `/css`, `/js`, `/assets`. Usa responsive design (Bootstrap o CSS Grid) para móviles.
    - Versionado: Crea branches (ej: `git checkout -b feature-menu`) para experimentar sin romper main.
    - SEO y Accesibilidad: Agrega meta tags en HTML (ej: `<meta name="description" content="Mi proyecto de [curso]">`) y alt en imágenes.
- Problemas Comunes y Soluciones:
    - Archivo no sube: Verifica tamaño; comprime imágenes con tools como TinyPNG.
    - Pages no funciona: Repo debe ser público; revisa Actions en Settings para logs de build. Para JS dinámico, considera Jekyll o frameworks como React (necesita build step).
    - Errores 404: Asegura rutas correctas; usa relative paths (`./img/foto.jpg`).
    - Seguridad: Nunca subas credenciales (API keys); usa variables de entorno o .env con .gitignore.
- Eficiencia: Para flujos repetitivos, integra con VS Code (extensión GitHub Pull Requests) o GitHub Desktop app.
- Beneficios Académicos: Enlaza tu repo en LinkedIn o CV. Úsalo para demos en clases (ej: prototipo de app web para tesis).
- Recursos Recomendados:
    - Docs oficiales: docs.github.com/es/pages (en español).
    - Tutoriales: freeCodeCamp en YouTube ("GitHub Pages Tutorial") o MDN Web Docs para HTML/CSS.
    - Para avanzado: Integra con Jekyll para blogs estáticos o Netlify como alternativa si necesitas más features.

¡Listo! Con esto, tendrás tu página online y un repo profesional para impresionar en entrevistas o entregas. Si estás trabajando en un framework

-