

1.3. Publicación de la plantilla de planificación del SLDC

1.3.1 ¿La plantilla de planificación que mencionaste en Markdown podría ser colocada dentro de una página HTML para posteriormente ser colocada en github?

Sí; la plantilla en Markdown se puede incrustar dentro de una página HTML y subir a GitHub de dos formas habituales:

1. GitHub renderiza **automáticamente** cualquier archivo README .md (o .md) como HTML; basta con subir el Markdown y ya se ve formateado sin que hagas nada más.

2. Si quieres una **página web propia** (que se sirva como HTML real, no como vista previa de Markdown) puedes:

a) Activar **GitHub Pages** en tu repositorio.

b) Crear un fichero `index.html` que incluya el contenido de la plantilla.

● Lo más rápido es usar un conversor on-line o `marked.js` para transformar el Markdown a HTML.

● También puedes escribir directamente la tabla en HTML.

Ejemplo mínimo de `index.html` (con la tabla convertida a HTML y algo de estilo):

HTML

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Definición de Alcance - Proyecto X</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 2rem; }
    h1, h2 { color: #0366d6; }
    table { border-collapse: collapse; width: 100%; margin-top: 1rem; }
    th, td { border: 1px solid #ddd; padding: .5rem; text-align: left; }
    th { background: #f6f8fa; }
  </style>
</head>
<body>
  <h1>Proyecto: Sistema de Gestión de Inventario - Tienda X</h1>

  <h2>1. Objetivo general (SMART)</h2>
  <p>Reducir el desfase de inventario del 8 % actual a ≤ 2 % anual.</p>

  <h2>2. Objetivos específicos</h2>
  <ul>
    <li>OE1: «Tiempos de respuesta ≤ 1 s» (indicador: avg_latency, meta: 1 s,
fecha: 30/09/2025)</li>
  </ul>

  <h2>3. Alcance</h2>
  <h3>3.1 In-Scope</h3>
  <ul>
    <li>✅ CRUD de productos</li>
    <li>✅ Alertas de stock mínimo</li>
  </ul>
```

```

<h3>3.2 Out-of-Scope</h3>
<ul>
  <li>✗ Facturación electrónica</li>
  <li>✗ App móvil nativa</li>
</ul>

<h3>3.3 Criterios de aceptación</h3>
<ul>
  <li>CA1: Tiempo de respuesta ≤ 1 s</li>
  <li>CA2: Disponibilidad 99 % en horario comercial</li>
</ul>

<h2>4. Restricciones</h2>
<table>
  <tr><th>Parámetro</th><th>Valor</th></tr>
  <tr><td>Presupuesto</td><td>30 000 USD</td></tr>
  <tr><td>Duración</td><td>6 meses</td></tr>
  <tr><td>Stack</td><td>PostgreSQL, Node.js, React</td></tr>
  <tr><td>Normativas</td><td>Protección de datos local</td></tr>
</table>
</body>
</html>

```

Pasos para tenerlo on-line:

Haz un repositorio nuevo en GitHub.

1. Sube `index.html` (y el `.md` si quieres mantener ambos).
2. En **Settings** → **Pages** elige «Deploy from branch» → `main` → `/` (root).
3. Guarda y GitHub te dará una URL tipo `https://tu-usuario.github.io/tu-repo/`.

Ventajas:

- El HTML se sirve estáticamente sin coste.
- Puedes enlazar la página desde el `README.md` para que quien entre al repo vea la versión bonita con un clic.

1.3.2 ¿Qué es git y para qué sirve? ¿Cuál es la dirección para obtener git?

Git es un **sistema de control de versiones distribuido** (VCS) que registra los cambios realizados en archivos a lo largo del tiempo para que puedas recuperar versiones anteriores y coordinar el trabajo entre varias personas de forma eficiente.

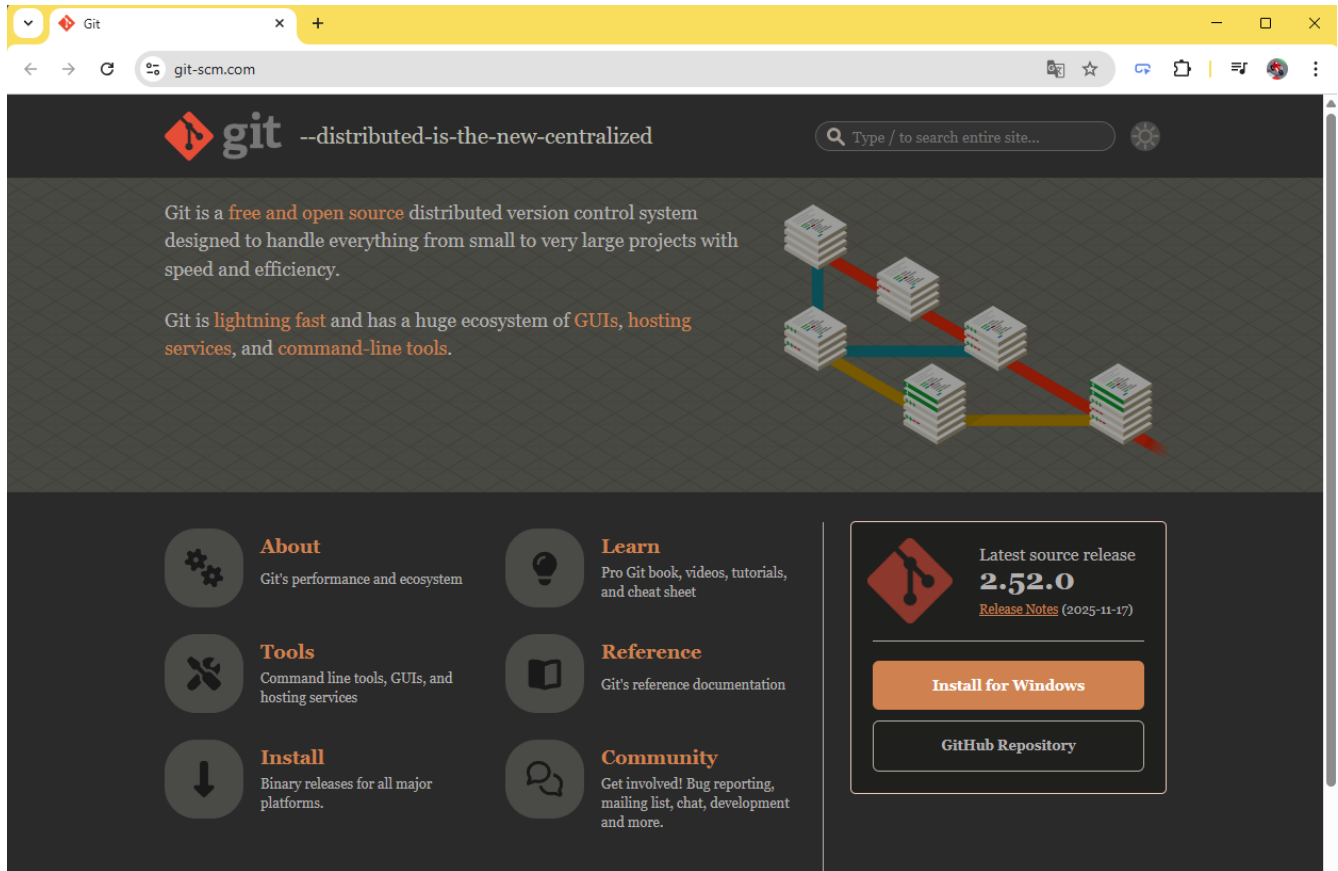
¿Para qué sirve?

- Llevar un historial completo de modificaciones (código, documentos, imágenes, etc.).
- Trabajar en paralelo sin pisar los cambios de otros (ramas, *merge*).
- Volver a cualquier punto del pasado con un solo comando (`git checkout`, `git reset`).
- Facilitar la colaboración remota (GitHub, GitLab, Bitbucket).
- Desplegar automáticamente mediante *hooks* o CI/CD.

Dirección oficial para obtener Git

👉 <https://git-scm.com>

Desde ahí puedes descargar el instalador para Windows, macOS o Linux y acceder a la documentación completa.



1.3.3 ¿Qué es Github y cómo se relaciona con git?

GitHub es una **plataforma en línea** (servicio web) que **aloja repositorios Git** y añade herramientas colaborativas sobre ellos.

Relación con Git:

Table

Git

Software de control de versiones **local** (está en tu PC)

Te permite commit, branch, merge, etc.

Funciona sin conexión

Puede usarse solo

GitHub

Servicio **en la nube** que guarda copias de tus repos Git

Añade *pull*

requests, issues, wikis, actions (CI/CD), reviewers, forks, etc.

Requiere conexión para sincronizar (*clone, push, pull*)

Se usa **junto con Git**: Git maneja el historial, GitHub lo expone y enriquece

Esquema típico:

1. Instalas Git en tu máquina.
2. Creas un repositorio local (`git init`).
3. Registras tu repo en GitHub (botón “New repository”).
4. Conectas ambos (`git remote add origin https://github.com/tu-usuario/tu-repo.git`).
5. Trabajas localmente; cuando quieres compartir: `git push origin main`.

En resumen:

Git = motor de control de versiones.

GitHub = garaje social y herramientas extra para ese motor.

1.3.4 ¿Porqué es importante usar Panel de Control → Administración de Credenciales → Credenciales de Windows → git:<https://github.com> → Editar para usar GIT?

Porque así **evitas que Git te pida usuario y contraseña/token cada vez que haces push o pull**. Windows guarda el **Personal Access Token (o contraseña)** en el **Administrador de credenciales**; si no está ahí, la autenticación falla o se desautentica al cambiar tokens. Editar esa entrada (o crearla) garantiza que Git use siempre el **token actual** sin exponerlo en scripts ni almacenarlo en texto plano.

1.3.5 ¿Qué tipos de error marca por no administrar credenciales de git?

Si la entrada **git:<https://github.com>** no existe, está desactualizada o contiene un token revocado, Git lanzará uno de estos mensajes (el texto varía ligeramente según la versión):

1.Authentication failed

Copy

```
remote: Support for password authentication was removed on August 13, 2021.  
remote: Please see https://docs.github.com/en/get-started/token  
fatal: Authentication failed for 'https://github.com/...'
```

2.Invalid username or password

Copy

```
Logon failed, use ctrl+c to cancel basic credential prompt.  
fatal: Invalid username or password
```

3.403 Forbidden / rate-limit

Copy

```
remote: 403 Forbidden  
remote: You must use a personal access token
```

```
fatal: unable to access 'https://github.com/...' The requested URL returned error: 403
```

4. Credential helper infinite loop

Ventana de credenciales que reaparece tras escribir la contraseña; Git nunca llega a autenticar.

5. No credential provider (en CI o scripts sin GUI)

```
fatal: could not read Username for 'https://github.com': terminal prompts disabled
```

En todos los casos el **origen común** es que Windows (o el credential-helper) **no encuentra un token válido** guardado; crear/actualizar la entrada en *Panel de control* → *Credenciales de Windows* resuelve el problema.

1.3.6 ¿Si tienes dos cuentas diferentes de Google y las usas en Github, que errores te puede marcar si no administras las credenciales?

Con dos cuentas de Google (p. ej. personal y laboral) ligadas a sendas cuentas de GitHub, el **Administrador de credenciales de Windows** solo puede guardar **UNA entrada** para `git:https://github.com`. Si no la editas manualmente:

1. **Autenticación cruzada**: al hacer push Git usa el token guardado de la cuenta A, pero la URL remota apunta al repo de la cuenta B →

Copy

```
remote: Permission to UsuarioB/repo.git denied to UsuarioA
fatal: unable to access 'https://github.com/...' The requested URL returned error: 403
```

2. **Infinite credential prompt**: Windows intenta el token A, GitHub lo rechaza, el helper lo borra y vuelve a pedir credenciales sin parar.

3. **Commits firmados con el e-mail “equivocado”**: tu `user.name`/`user.email` local pueden corresponder a la cuenta B, pero el token es de A; aparece el aviso
`Commit author does not match authenticated user.`

4. **Doble 2FA / SSO loop**: si una cuenta exige SSO corporativo y la otra no, el token caduca o redirige, causando

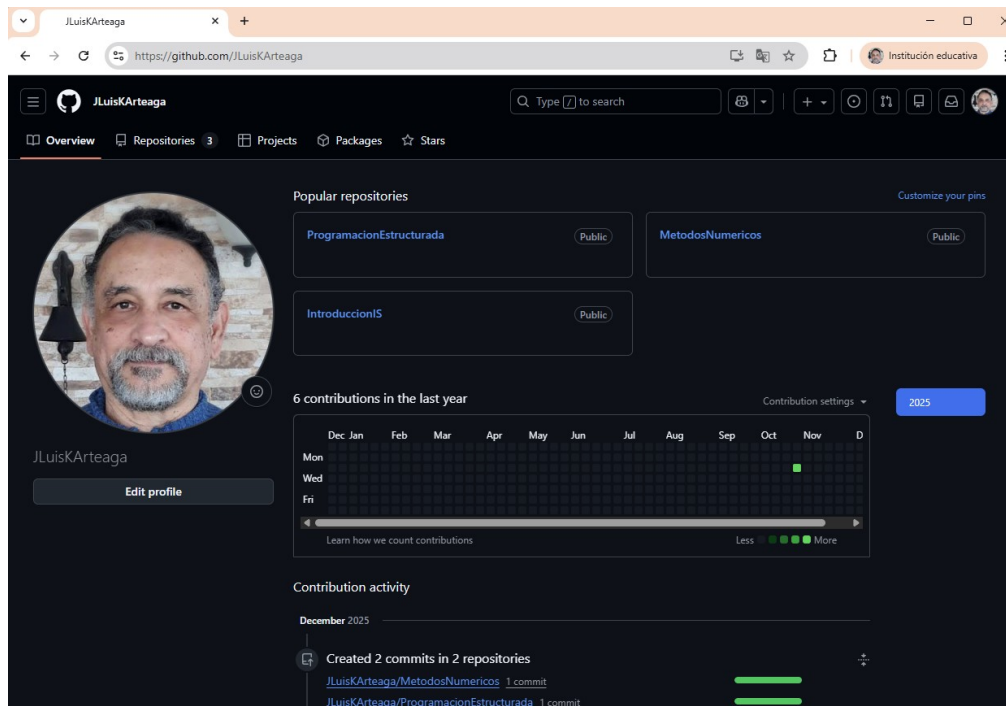
Copy

```
remote: 401 Unauthorized
fatal: Authentication failed for 'https://github.com/...'
```

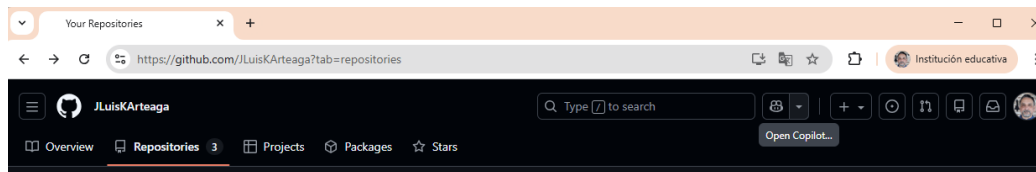
Solución:

- Borra la entrada antigua en *Credenciales de Windows*.
- Crea otra con el **token PAT** correspondiente al usuario que vas a usar (o usa `credential.<url>.username` + distintos hosts `github.com-corp` + SSH).

1.3.7 Caratula inicial de Github



1.3.8 ¿Copilot en Github es una IA?



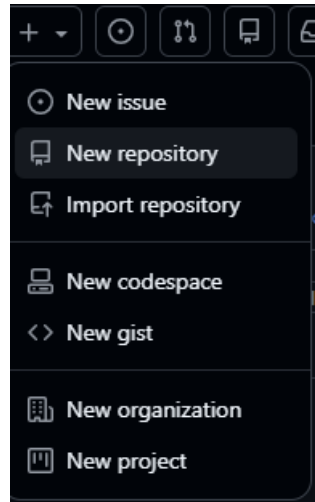
Sí, **GitHub Copilot** es una **IA generativa** (modelo de lenguaje desarrollado por OpenAI y optimizado por GitHub/Microsoft) que **sugiere código completo o fragmentos en tiempo real** mientras escribes.

1.3.9 Pasos para cambiar el password en Github

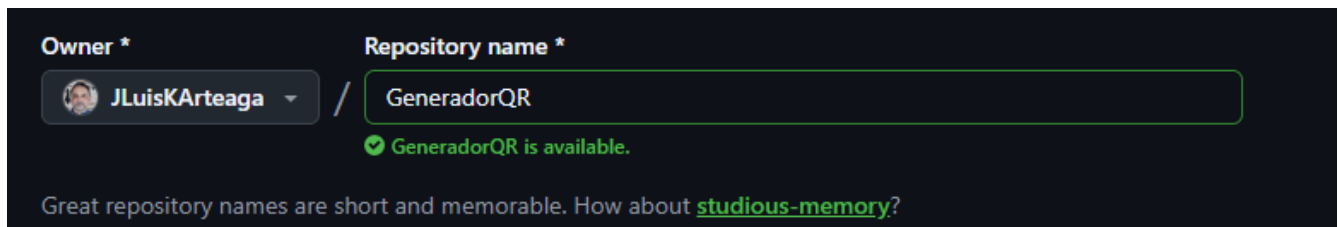
1. Inicia sesión en GitHub.
2. Haz clic en tu foto → **Settings** → **Password** (menú lateral).
3. Introduce la contraseña actual y la nueva dos veces.
4. Guarda; GitHub cerrará sesión en el resto de dispositivos como medida de seguridad.

1.3.10 Crear un nuevo repositorio Github

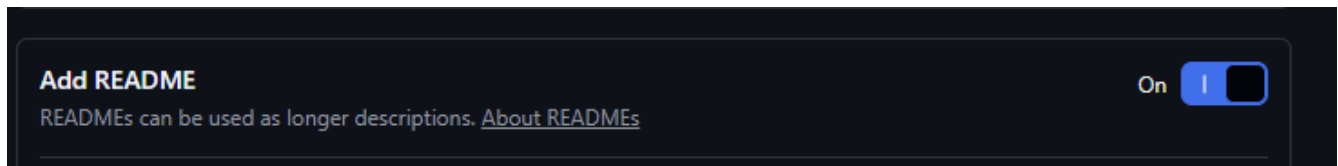
En el botón +



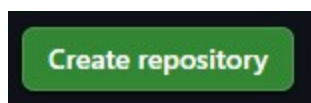
Escribir el nombre GeneradorQR

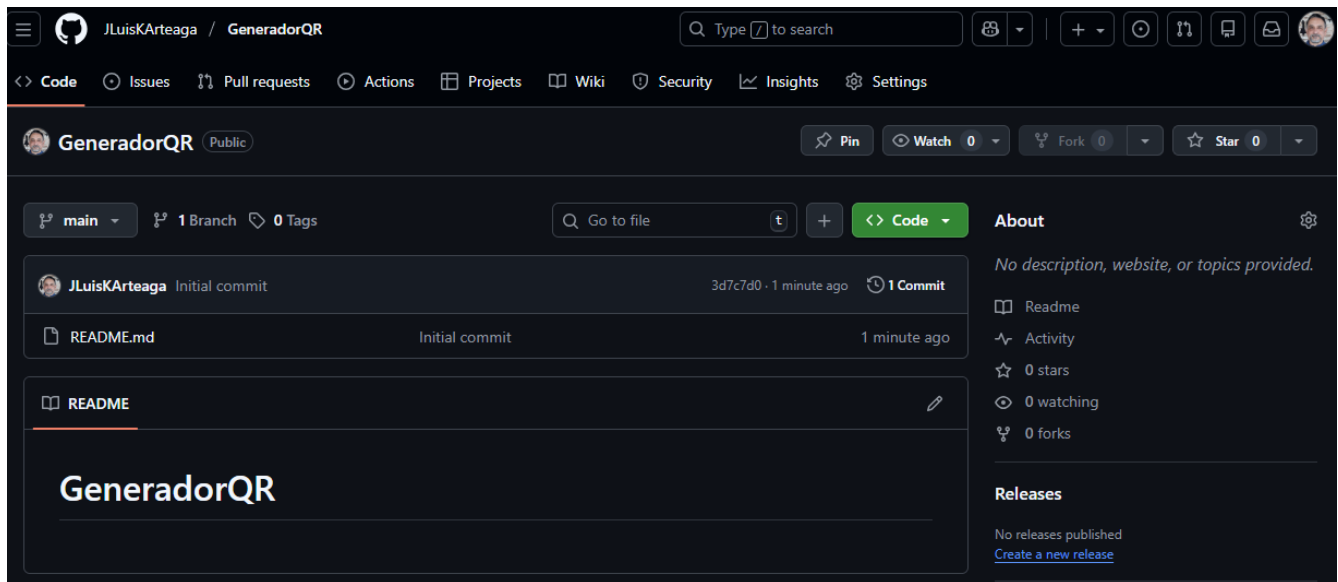
A screenshot of the GitHub repository creation form. It shows the 'Owner' field with the user 'JLuisKArteaga' and the 'Repository name' field with the text 'GeneradorQR'. Below the name field, there is a green checkmark and the text 'GeneradorQR is available.' At the bottom, there is a message: 'Great repository names are short and memorable. How about [studious-memory?](#)'

Agregar README

A screenshot of the 'Add README' toggle switch. The text 'Add README' is on the left, followed by a description: 'READMEs can be used as longer descriptions. [About READMEs](#)'. On the right, there is a toggle switch labeled 'On' which is currently turned on (blue).

Presionar botón “Create Repository”





De esta forma se crea el repositorio con el nombre “GeneradorQR”

Crear el archivo “app.js” con el siguiente código:

```
function generarQR() {  
  const texto = document.getElementById('texto').value.trim();  
  if (!texto) {  
    alert("Por favor ingresa texto o URL");  
    return;  
  }  
  
  // Limpia QR previo  
  document.getElementById('qrcode').innerHTML = "";  
  
  // Genera nuevo QR  
  new QRCode(document.getElementById('qrcode'), {  
    text: texto,  
    width: 180,  
    height: 180,  
    colorDark: "#000000",  
    colorLight: "#ffffff",  
    correctLevel: QRCode.CorrectLevel.H  
  });  
}
```

Crear el archivo index.html con el siguiente código:

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
  <meta charset="UTF-8" />
```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Generador QR - Kimi</title>
<link rel="stylesheet" href="style.css" />
<!-- Librería QR sin dependencias -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/qrcodejs/1.0.0/qrcode.min.js"></script>
</head>
<body>
  <div class="container">
    <h1>Generador de Código QR</h1>
    <input id="texto" type="text" placeholder="Texto o URL" />
    <button onclick="generarQR()">Generar QR</button>
    <div id="qrcode" class="qr-box"></div>
  </div>
  <script src="app.js"></script>
</body>
</html>

```

Crear el archivo stile.css y colocar el código:

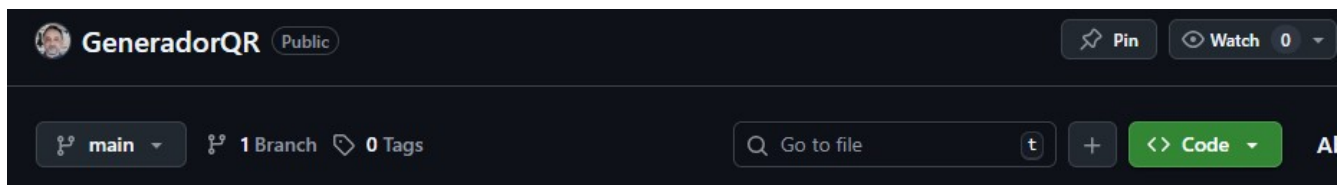
```

body {
  font-family: Arial, sans-serif;
  background: #f4f4f4;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
}
.container {
  background: #fff;
  padding: 30px;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
  text-align: center;
  width: 320px;
}
h1 {
  margin-bottom: 20px;
  font-size: 1.5em;
}
input {
  width: 90%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 4px;
}
button {
  padding: 10px 20px;

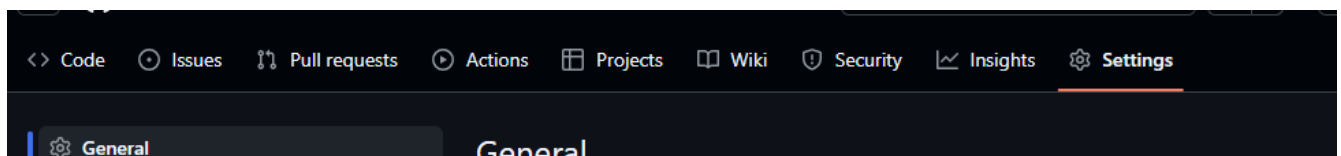
```

```
background: #007bff;
color: #fff;
border: none;
border-radius: 4px;
cursor: pointer;
}
button:hover {
  background: #0056b3;
}
.qr-box {
  margin-top: 20px;
  display: flex;
  justify-content: center;
}
```

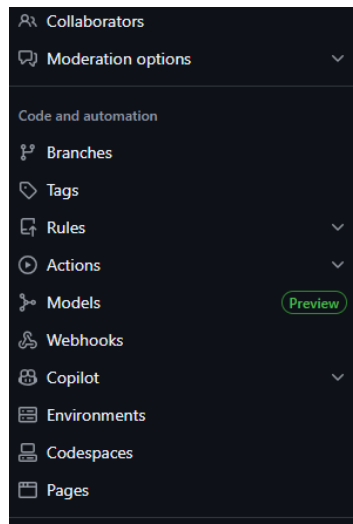
Estado en el repositorio GeneradorQR presionar el botón + y cargar los tres archivos anteriores con Upload Files



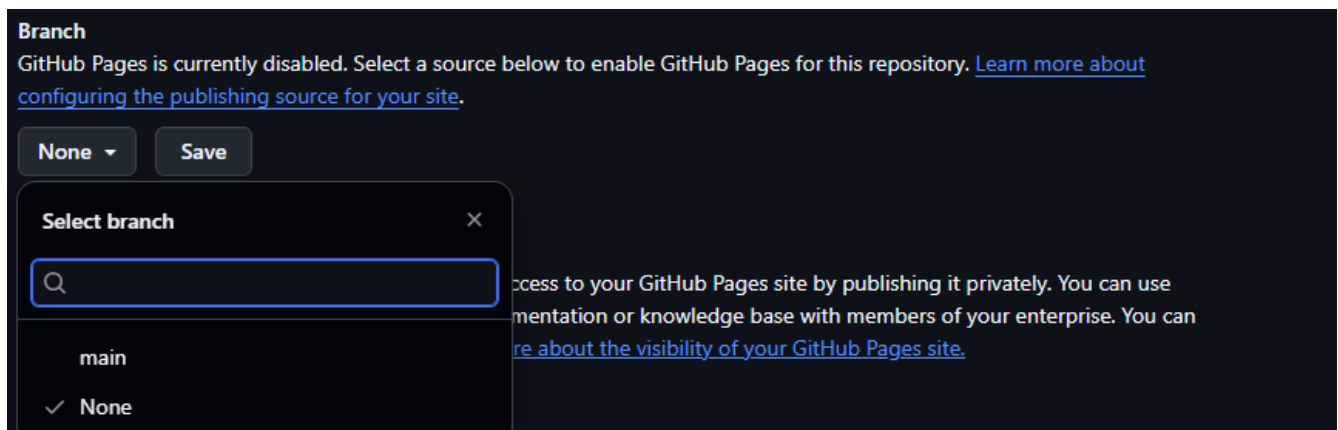
En Settings



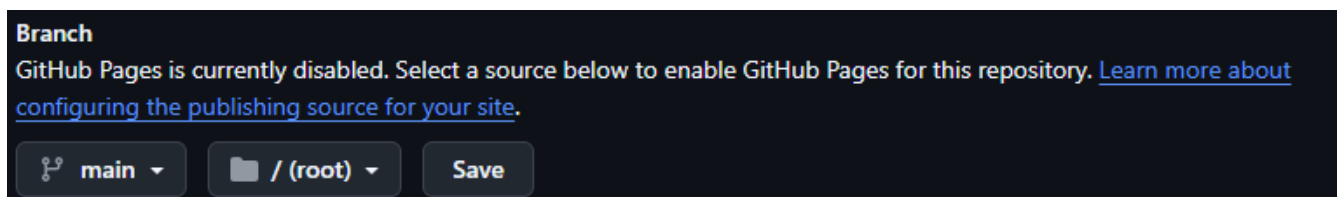
En Pages (lado izquierdo)



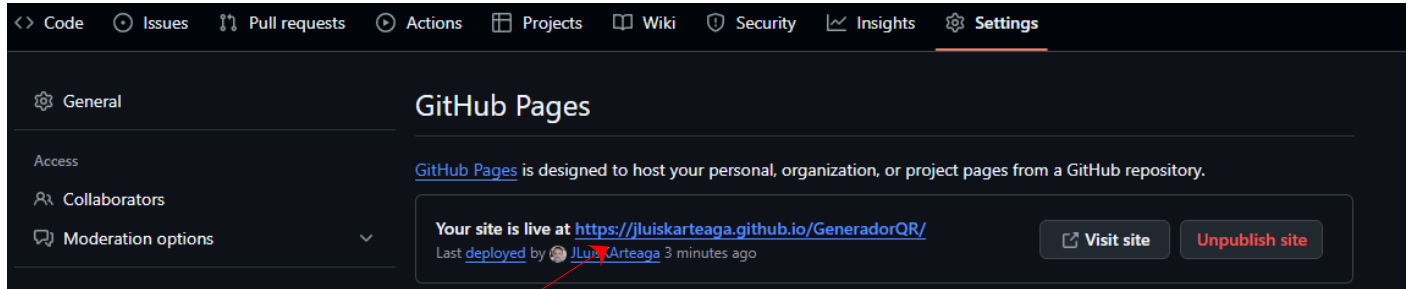
En Branch seleccionar main



Y presionar Save



Esperar a que se cargue la página ...Y regresando a Settings



Revisar la dirección de carga *Ya está activo y puedes usarlo cuando quieras*

Si presionas esa dirección podrás ver tu propio generador QR funcionando. Coloca cualquier dirección para generar su código QR



Y siempre estará activo para cuando lo necesitas utilizar.

Regresando a la plantilla de planificación del SLDC usando el código en el punto 1.3.1, construir el repositorio PlanificacionSLDC01 que pueda ser utilizada desde github. Generar el QR del repositorio para la plantilla.

