**1. Estructura de carpetas**

* **Descripción**: El proyecto debe estar bien organizado, con carpetas separadas para cada tipo de archivo (por ejemplo, /css, /img, /js, etc.). El archivo HTML debe estar en la raíz del proyecto.
* **Evidencia**: Captura de pantalla del proyecto en VS Code mostrando el árbol de carpetas.

**2. Estructura básica HTML**

* **Descripción**: El archivo HTML debe contener la estructura básica con las etiquetas estándar: <!DOCTYPE>, <html>, <head>, <body>, etc. La estructura debe estar limpia y organizada.
* **Evidencia**: Captura de pantalla del código fuente del archivo HTML, mostrando la estructura básica.

**3. Estilos CSS**

* **Descripción**: Debe haber un archivo CSS correctamente vinculado al HTML. Este debe incluir al menos 10 reglas de estilo que definan la apariencia de la página (colores, fuentes, márgenes, etc.).
* **Evidencia**: Captura del archivo .css con al menos 10 reglas de estilo.

**4. Uso de Git y GitHub (Repositorio creado)**

* **Descripción**: El proyecto debe estar alojado en un repositorio de GitHub. Se debe haber hecho un commit inicial y al menos dos commits más para reflejar avances.
* **Evidencia**: Captura del repositorio en GitHub, mostrando la URL del repo y el historial de commits realizados.

**5. Commits realizados**

* **Descripción**: Se debe haber realizado un mínimo de tres commits significativos, con mensajes descriptivos para cada uno.
* **Evidencia**: Captura del historial de commits en GitHub, mostrando los mensajes de cada uno.

**6. GitHub Pages desplegado**

* **Descripción**: El proyecto debe estar desplegado utilizando GitHub Pages para que sea accesible públicamente.
* **Evidencia**: Captura de la configuración de GitHub Pages en el repositorio, junto con la URL pública de la página.

**7. Visualización de la página**

* **Descripción**: La página debe mostrarse correctamente con los estilos aplicados (mínimo: estructura básica, colores, tipografía, etc.). La funcionalidad básica de la página debe estar funcionando.
* **Evidencia**: Captura de pantalla de la página final desplegada en GitHub Pages.

**8. Formulario simple (con inputs)**

* **Descripción**: Debe incluir un formulario básico con al menos dos campos de entrada, como nombre y correo, y un botón de envío.
* **Evidencia**: Captura del formulario HTML dentro de la página, mostrando los campos de entrada y el botón.

**9. README.md en el repositorio**

* **Descripción**: El repositorio debe contener un archivo README.md donde se explique brevemente el propósito del proyecto, cómo ejecutarlo (si es necesario), y cualquier otra información relevante.
* **Evidencia**: Captura del archivo README.md en el repositorio de GitHub.

**10. Captura del Navbar y navegación entre páginas**

* **Descripción**: Si el proyecto incluye varias páginas HTML, debe haber un sistema de navegación (como un Navbar) que permita moverse entre ellas.
* **Evidencia**: Captura de la sección de código que muestra el Navbar en el HTML, y una captura de la navegación funcionando en el navegador.

**11. Código para favicon**

* **Descripción**: El proyecto debe incluir un favicon, es decir, un ícono que aparece en la pestaña del navegador.
* **Evidencia**: Línea de código utilizada para incluir el favicon. Ejemplo de código en el <head>:
* <link rel="icon" href="img/favicon.ico" type="image/x-icon">

**12. Enlace de ejemplo**

* **Descripción**: El HTML debe incluir al menos un enlace (<a>) que redirija a otra página o recurso.
* **Evidencia**: Línea de código de un enlace en HTML. Ejemplo:
* <a href="https://www.ejemplo.com">Ir a ejemplo.com</a>

**13. Captura del <head> del HTML**

* **Descripción**: El código HTML debe contener el <head> correctamente estructurado, con enlaces a archivos CSS, favicon, y cualquier otra etiqueta de metadatos. Se debe mostrar cómo se vinculan los recursos externos.
* **Evidencia**: Captura de pantalla del código en la sección <head>, mostrando las etiquetas <link> para CSS y el <meta> para el favicon.

**14. 5 Etiquetas que más les hayan gustado o quisieran resaltar**

* **Descripción**: Los estudiantes deben elegir **5 etiquetas HTML** que les parezcan más interesantes o útiles, y explicar brevemente por qué las eligieron. Por ejemplo: <img>, <a>, <table>, <div>, etc.
* **Evidencia**: Una lista en el archivo README.md o en un comentario dentro del código, mencionando las etiquetas elegidas y un breve comentario sobre cada una. Ejemplo:
* <img src="imagen.jpg" alt="Descripción de la imagen"> <!-- Esta etiqueta me gusta porque es clave para agregar imágenes -->

**15. Código de una tabla HTML**

* **Descripción**: El proyecto debe incluir una tabla simple en HTML que esté correctamente estructurada con las etiquetas <table>, <tr>, <th>, <td>, y con algún estilo básico.
* **Evidencia**: Línea de código de una tabla en HTML. Ejemplo:
* <table>
* <tr>
* <th>Nombre</th>
* <th>Edad</th>
* </tr>
* <tr>
* <td>Ana</td>
* <td>25</td>
* </tr>
* <tr>
* <td>Juan</td>
* <td>30</td>
* </tr>

</table>

**16. Paso a paso básico con Git**

* **Descripción**: Los estudiantes deben demostrar cómo han utilizado Git a lo largo de su proyecto, incluyendo los comandos básicos para gestionar el flujo de trabajo. Esto incluye el uso de git status, git add, git commit, y git push para gestionar los cambios en su repositorio.
  + **Flujo básico**:
    1. **git status**: Ver el estado de los archivos en el repositorio.
    2. **git add <archivo>**: Preparar archivos específicos para el commit.
    3. **git commit -m "mensaje"**: Realizar el commit con un mensaje descriptivo.
    4. **git push**: Subir los cambios al repositorio remoto en GitHub.
* **Evidencia**: Captura de pantalla del terminal o línea de comandos mostrando el uso de estos comandos, con ejemplos de cómo los archivos van de "modificados" a "listos para commit", y cómo se suben al repositorio remoto.

**17. Descripción de algunos comandos avanzados de Git**

* **Descripción**: Los estudiantes deben investigar y explicar brevemente algunos comandos de Git que son esenciales para una correcta configuración y gestión del repositorio. Los comandos a explicar pueden incluir:
  1. **git config**: Se utiliza para configurar las opciones globales de Git, como el nombre de usuario y correo.
  2. **git remote**: Permite gestionar las conexiones a repositorios remotos, como GitHub, para sincronizar el código local con el remoto.
  3. **git pull**: Permite traer cambios del repositorio remoto al local.
  4. **git branch**: Muestra las ramas del repositorio o permite crear nuevas ramas.
* **Evidencia**: Descripción de cada comando en el archivo README.md o como comentario dentro del código. Ejemplo:
* git config --global user.name "Tu Nombre"
* git config --global user.email "tu@correo.com"

**18. Ejemplo de trabajo con ramas en Git**

* **Descripción**: Como punto adicional, se puede pedir que los estudiantes trabajen con ramas, realizando un cambio o ajuste en una nueva rama y luego haciendo merge de esa rama con la principal (main o master).
* **Evidencia**: Explicar cómo crearon y trabajaron con ramas, con el uso de comandos como git branch, git checkout, y git merge. También, capturas de pantalla mostrando el proceso de crear una rama y hacer el merge.

### ****Resumen de los puntos nuevos****

1. **Paso a paso básico con Git**: Uso de los comandos git status, git add, git commit y git push.
2. **Descripción de comandos avanzados de Git**: Explicación de comandos como git config, git remote, git pull.
3. **Trabajo con ramas**: Creación y fusión de ramas (git branch, git checkout, git merge).