HACKATHON SEMANAL

Módulo 2: Control de versiones y HTML (Semana 2)

LOGRO: Crear proyecto con control de versiones y despliegue con gitHub Pages, implementar etiquetas html5 y definir estructura html del primer proyecto.

I. Es hora de demostrar lo aprendido:

Demostrarás todo lo aprendido en este reto que se basará en las clases dictadas durante la semana.

II. Insumos para resolver el Reto:

- Materiales de clase de la semana 1 y 2.
- https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5_lista_elementos
- https://www.figma.com/file/DmxA6mUgZJts6T8pJ01qww/Project-basic?node-id=0%3A1

III. Descripción del reto

El gran reto está formado por 4 pequeños retos.

a. El primer reto consiste en leer un caso, investigar y resolver ciertas preguntas.
b. El segundo reto trata de crear la estructura del proyecto inicial. Crearás tanto las carpetas como los archivos e inicializarás git, tanto de manera local como reto, y subirás el proyecto a la nube.
c. El tercer reto trata sobre escribir etiquetas html. Crearás una rama para subir los cambios específicos de esta tarea.
d. El cuarto reto es un proyecto integrador. En esta oportunidad resolveremos todo lo relacionado al HTML.

IV. Pasos a seguir para resolver los retos:

- El docente indicará si este reto se resolverá de manera individual o grupal

Primer reto:

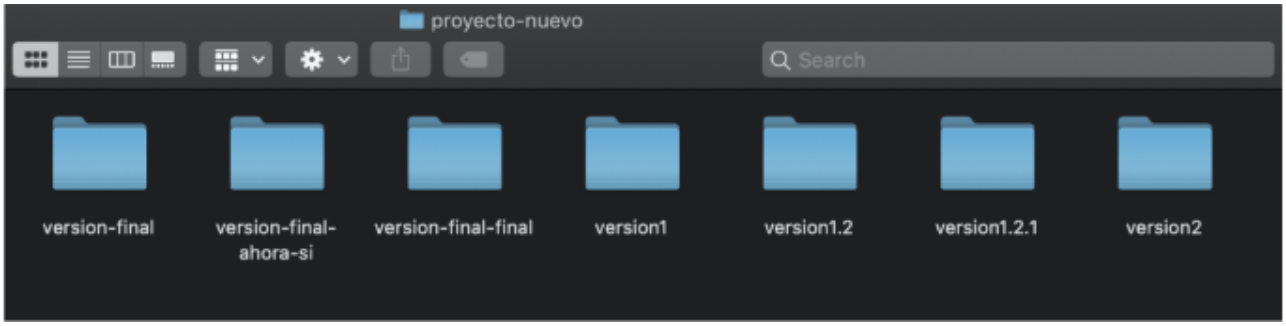
TÍTULO: control de versiones y html

¿Por qué utilizar un CVS?

Git, Github, Gitlab, Bitbucket.

PROBLEMA:

Es muy probable que en algún momento de nuestras vidas antes de conocer el control de versiones hayamos hecho esto.



Creando copias de nuestro proyecto para controlar las versiones y usando nuestra creatividad para nombrar estas carpetas para saber cuál es el proyecto final.

Si tuviéramos que compartir el proyecto, tendríamos que usar un almacenamiento externo (usb, google drive, etc) y el problema de saber quién modificó tal archivo o carpeta se multiplica por la cantidad de personas dentro del proyecto, un total caos.

Aquí es donde entra Git, permitiéndonos mantener un historial de cambios y basándose en ese historial poder regresar a cualquier versión de nuestros documentos.

BENEFICIOS:

Permite revertir los archivos seleccionados a un estado anterior, revertir todo el proyecto a un estado anterior, comparar los cambios a lo largo del tiempo, ver quién modificó por última vez algo que podría estar causando un problema, quién presentó un problema y cuándo, y más. Usar un VCS (Version Control System) también generalmente significa que, si arruinas las cosas o pierdes archivos, estos pueden recuperarse fácilmente.

Los desarrolladores de software que trabajan en equipos escriben continuamente nuevos códigos fuente y cambian el código fuente existente. El código para un proyecto, aplicación o componente de software generalmente se organiza en una estructura de carpetas o "árbol de archivos". Un desarrollador del equipo puede estar trabajando en una nueva característica, mientras que otro desarrollador corrige un error no

relacionado al cambiar el código, cada desarrollador puede hacer sus cambios en varias partes del árbol de archivos.

Los equipos de software que no usan ninguna forma de control de versiones a menudo se encuentran con problemas como no saber qué cambios se han realizado para los usuarios o la creación de cambios incompatibles entre dos piezas de trabajo no relacionadas que luego deben ser minuciosamente desenredadas y re elaboradas.

Si usted es un desarrollador que nunca ha utilizado el control de versiones, es posible que haya agregado versiones a sus archivos, tal vez con sufijos como "final" o "último" y luego tuvo que lidiar con una nueva versión final. Quizás haya comentado los bloques de código porque desea deshabilitar cierta funcionalidad sin eliminar el código, por temor a que pueda ser utilizado más adelante. El control de versiones es una forma de salir de estos problemas.

PREGUNTAS:

1. ¿Qué es un control de versiones?
2. ¿Cuáles son los problemas al no usar un control de versiones?
3. ¿Cuáles son los beneficios?
4. ¿Qué tipos de control de versiones existen?

REFERENCIAS

- https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
- Material visto en clases

Segundo reto:

- Inicializar repositorio local, configurar repositorio remoto, crear carpetas y archivos base y subir cambios
- Crear repositorio remoto en github
- En la raíz del proyecto Inicializar repositorio con git
- Agregar archivos a stage
- Crear registro (commit) y agregarle un mensaje corto descriptivo
- Configurar repositorio remoto
- Subir cambios a repositorio remoto

Tercer Reto:

- Leer la lista de etiquetas HTML5 e implementarlas para verificar su funcionalidad
- Ingresar al siguiente enlace - https://developer.mozilla.org/es/docs/HTML/HTML5_lista_elementos
- Crear rama **feature-tags-html** a partir de master
- Crear archivo **tags.html**
- Escribir etiquetas de la lista de elementos html5
- Agregar cambios al stage
- Crear versión y definir mensaje corto relacionado al registro
- Subir cambios a repositorio remoto

Cuarto Reto:

- Inicializar html del proyecto
- Ingresar al siguiente enlace - https://www.figma.com/file/DmxA6mUgZJts6T8pJ01qww/Project-basic?node-id=0%3A1
- Crear rama **feature-html** a partir de master
- Dentro del archivo **index.html** crear la estructura html del diseño
- Agregar cambios al stage
- Crear versión y definir mensaje corto relacionado al registro
- Subir cambios a repositorio remoto
- Cambiar a rama master y combinar cambios de rama feature-html

V. Solución del reto

- Para que el reto esté cumplido al 100%, se deben haber respondido las preguntas planteadas y se deben haber resuelto los ejercicios

VI. Presentación del Reto

- El documento debe ser presentado de manera individual o grupal (según se coordine con el docente)
- El tiempo de cada presentación lo definirá el docente a cargo

VII. Feedback

- El docente dará feedback a los estudiantes sobre los ejercicios realizados