**Primera clase, 27 de octubre.**

**Tema: Versionadores de system y de hosting.**

Se inicia clase con la definición de CLI que son las siglas de Comando en Línea que se refiere a la interfaz de texto que se usa para hacer algo; este es un versionado system donde se trabaja de manera local. Tiene dos vertientes. La primera es el uso de Comando Símbolo o de Terminal (en Win 11 aparece como PowerShell) para el sistema de Windows en general y, para el caso de Linux y de MacOs se recurre a Unix.

Se usa este medio de versionado de tipo system, por la facilidad, rapidez y uso mínimo de recursos. Es usado más que todo por DevOs (Desarrolladores de la Nube) y Back-End por las características mencionadas anteriormente. En contraparte o a manera complemento del CLI existe GUI que hace referencia a la Interfaz Gráfica de Interacción del usuario. Es decir, que esta última es la parte gráfica de lo que se usa con el CLI en la Terminal de un PC.

Mientras que para CLI se requiere línea a línea para hacer algo; en GUI solo se requiere usar el mouse para ir y seleccionar, es decir, se usa el Explorar de Archivos (Win) o el Finder (MacOs) para acceder a un archivo o directorio. Aunque los dos métodos funcionan igual para buscar y crear archivos o directorios (esta palabra, directorios, es igual que decir carpeta) no es suficiente para generar interacción con la plataforma de Versionado Hosting.

El versionado Hosting, se trabaja de manera virtual usando la nube como principal almacenador de información y sirve para tener información casi que en tiempo real para ver modificaciones (llámese: agregar, quitar, cambiar, revisar) el archivo que se está trabajando. Puede verse todo lo que se haga en él. Hosting de mayor uso son: GitHub, GitLab y GitBucket. Con el GitHub, se busca crear Repositorios donde se guarden las versiones nuevas y se almacenen los cambios (versiones viejas) de los proyectos que se estén ejecutando o que ya están ejecutados.

Brinda el paso a paso de la creación de un Repositorio en GH:

1. Crear desde GUI (página web)
2. Owner: colocar nombre que se diferente a programas
3. Agregar una descripción de lo qué es, para qué es, qué se usará, etc..
4. Establecer si es Público o Privado
5. Es importante dar en: Add README.md para que se cree un archivo de texto. El cual contiene:
   * + 1. Título
       2. Descripción (se pueden crear listas y agregar imagen, por ej.)
       3. Autor

Adicional a ello, en el caso de la construccion de contenido en el archivo README.md se puede jerarquizar la información hasta máximo 6 niveles los cuales se identifican por el número de # que se usen. También, menciona, se usa el [] para escribir nombre (en este caso del autor) y al lado se usan () para insertar un link de una página activa. Con la combinación de [aa, pp.] (www.) aparecerá en el archivo como un nombre con hipervínculo.

Recalca la importancia del autor puesto será quien aparezca en los Commit que harán referencia a las personas que hacen modificaciones y que puede verlo otras personas. Es una huella que se captura y deja en el historial. Explicar que los Commit son comentarios breves, más que frases, explicativas de lo que se hizo.

Para este caso, se realiza la instalación de Git Pro que es un paquete de aplicaciones donde solo se usa el Git Bash; todo este paquete está creado y basado en Linux y se maneja bajo los comandos del mismo sistema. Se resalta los siguientes comando:

| **Comando** | **Que hace** | **Ejemplo** |
| --- | --- | --- |
| **cd** | Entrar a disco, carpeta, archivo | cd Documents |
| **pwd** | Ubicación del usuario | c/User/Cristian/Documents |
| **ls** | Lista de archivos o carpetas | También se usa **ls -la** para ver todo en orden y ubicar rápido lo que se busca lo que está oculto |
| **mkdir +nombre** | Para crear una carpeta | mkdir hola |
| **comando - - help** | Ayuda a ubicar un comando o bandera | Las banderas son los guiones y una palabra. Ej. **-la** |
| **mv 1dir miGitHub** | Con este se realiza el cambio de nombre | Primero va el comando **mv** seguido el nombre del archivo **1dir** y por último el nuevo nombre **miGitHub** |
| **git config user.** | Se usa para que el programa tenga conocimientos del usuario que modifica | .name “(va nombre de usuario entre comillas)”  .email xxx@xxx.xxx |
| **git config list** | Se usa para verificar que quedó registro el usuario y el correo |  |
| **git clone +link** | Para clonar el repositorio en carpeta local del PC | git clone https://www.github.com/hola.12/prueba |
| **git branch // git brach .** | Se usa para ver el main (raiz/rama) del Git |  |
| **code .** | Abrir VSC para modificar el archivo README.md |  |

Por último, para que lo modificado desde VSC el cual fue clonado desde Git Bash, se debe de hacer un proceso desde la Terminal o desde VSC para que estos cambios ejecutados en el programa local y guardados en PC, queden registrados en el sistema de GitHub se debe de realizar:



**Segunda y tercera clase: 28 y 31 de octubre, 2022**

**Tema: HTML. Estructuras básica, semánticas, tipográficas y de contenido**

Estas dos clases se enfocan en conocer la sintaxis de HTML el cual no es un lenguaje de programación como tal, sino que, sirve para la maquetación y la construcción de hipertextos en una página, sitio o aplicativo web. Se compone de etiquetas y atributos (éstos últimos por lo general están acompañados de valores específicos con los cuales se puede nominar, identificar, etc. Son del tipo: id, name, placeholder, class, etc.) como elementos principales para darle sentido y significado a una página; dentro de las cuales existe una jerarquía entre etiquetas y condiciones de uso (que aunque es libre de usar en cualquier parte, no siempre son buenas prácticas).

Con HTML se crea un esqueleto sin forma ni función o dinámica, únicamente sirve para ver reflejado el contenido que será leído por el navegador e identificado por el motor de búsqueda (identificado por el uso de metadatos que son un elemento que contiene una etiqueta <meta></meta> o el uso de lang=”idom” usado dentro de html). En HTML existen cuatro tipo de estructuras, las cuales son:

1. **Estructura básica**

Es darle a conocer al navegador que es lo que va abrir/leer. Se inicia con un editor de código y la primera línea corresponde a: <!DOCTYPE html> el cual define el tipo de documento que se construye sobre un tipo de sintaxis específica, en este caso html. Luego de esto, sigue lo que es el <html: 5> xxxxx </html> quien coloca la estructura base de una documento HTML, es decir, coloca las etiquetas de manera automática de: head y body. Antes de continuar con la descripción de esto, es importante mencionar que dentro de <html: 5> va el atributo **lang,** encargado de informar a la máquina que el documento está escrito en un determinado idioma.

Retomando la estructura que brinda el <html: 5>; da por sentado el head y el body. Donde en el primero es más para que el navegador ubique lo que se hace, por lo tanto coloca metadatos preestablecidos que ayudan con el diseño responsivo y lectura del documento para el navegador en sí mismo; en este caso agrega:

* <meta charset="UTF-8" />, sirve para que se lean los signo de puntuación y caracteres especiales como la virgulilla o la diéresis;
* <meta http-equiv="X-UA-Compatible" content="IE=edge" />, cumple la función de que se adapte al navegador de Microsoft Edge (no es necesario dejarlo); y,
* <meta name="viewport" content="width=device-width, initial-scale=1.0" />,, el cual tiene la función de adaptación a pantallas y encargado 100 % del proceso responsivo
* <title>Segunda clase de HTML</title> este es el nombre de la pestaña

Adicional a estos elementos o etiquetas predeterminadas, se explican:

* <meta name="description" content="" />, éste aparece como resumen cuando se realiza la búsqueda en el motor de búsqueda propiamente; se aconseja no sea más de 150 carácteres.
* <meta name="keywords" content="" />, sirve para palabras clave de búsqueda y posicionamiento; se aconseja no usar más de 10 palabras y no usar signos de puntuación.
* **Favicon:** que pueden ser usado de dos maneras, este es el icono que va en la pestaña:
  + <link rel="icon" type="image/x-icon" href="descarga.jpg" />

Por otro lado, en el caso del body es lo contrario del head pues aquí todo lo que va tiene como finalidad que el usuario lo vea y de alguna manera pueda usar lo que contiene el documento. Las características básicas de este apartado del HTML, son de tipo fila y bloque; algunas etiquetas son:

1. <div></div>, son un tipo de caja que ocupa el espacio de manera horizontal y también a lo largo; tipo bloque
2. <hr>, que hace una línea horizontal; tipo línea
3. <span></span>, el cual es un contenedor genérico; tipo bloque

Sin embargo, estas etiquetas no son usadas de manera recurrente a menos de que exista la necesidad de uso estricto. Para ello, existen las **Estructuras semánticas.**

1. **Estructurales**

Dirigidas para la organización del *body,* de la siguiente manera:

<body><!--acontinuación se presentan las etiquetas -->

<header><!--Son los encabezados o parte superior de la pag-->

<nav></nav><!--Son los link para navegar dentro del html-->

</header>

<main><!--Es la parte principal o contenido principal solo uno-->

<section><!--seccion de productos, servivios, etc. Es una seccion que tiene coherencia y cohesion que si se mueve una parte puede ser dificultoso enternderlo→ </section>

<article> <!--van ariticulos, se recomienda usarlos y moverlos debido a que no tiene carga o enlace significado con el contenido→ </article>

</main>

<aside><!--contenido secundario--></aside>

<footer><!--pie de pagina--></footer>

</body>

1. **Estructuras semánticas**

En este caso las E.S son etiquetas que contienen información ya sea para describir o definir una parte específica del *body*. Algunas tienen jerarquías, se cierran solas, solo se pueden usar una vez o tiene relación familiar (padre-hijos/hermanos). Algunas son:

* Los títulos que se identifican por el uso de **<h#>**. Estos títulos son jerarquizados del más importante al menos utilizando números y negrita, donde se organizan de manera descendente iniciando por el 1 hasta el 6. Es decir: **<h1></h1> … <h6></h6>**
* La etiqueta anchor o ancla que se representa por el uso de <a></a> y tiene atributos como es el caso de href=””. La función principal es para colocar enlaces externos (links o páginas fuera de los archivos guardados y localizados en el pc) o internos (que son los que están dentro de los archivos almacenados en el pc). Es decir, **<a *href=”***[**www.xxxxxx.xxx**](http://www.xxxxxx.xxx)***”>(Aqui se escribe algo alusivo al link)</a>.*** Esta se usa como ruta relativa=interna ***//* <a *href=”***[**www.xxxxxx.xxx**](http://www.xxxxxx.xxx)***” + target=”blank” rel="noopener noreferrer">(Aquí se escribe algo alusivo al link)</a>.*** Esta se usa como ruta absoluta=externa. Adicional a esto, también se pueden generar enlaces muertos asi: **<a href=”#”></a>**
* Para la creación de listas hay tres opciones:
  + Listas desordenadas que se usa **<ul></ul>**
  + Listas ordenadas donde se usa **<ol></ol>**

En estas dos listas se usa la etiqueta **<li></li>,** quienes son los encargados de contener la información y de mostrarla según sea el caso. Para <ul> mostrará viñetas y para <ol> mostrará numeración

* + Un tipo de lista donde se estable término y definición, se usa así:

**<dl>**

**<dt>Término</dt>**

**<dd>Definición</dd>**

**<dl>**

* También existen los formularios. Siempre van un botón y un checkbox; la estructura es así:

<form>

<label for="nombre">Nombre</label><!--el for esta enlazado al id del input-->

<input type="text" placeholder="Ingrese Nombre" id="nombre"/>

<br />

<!--Dos formas de presionar nombre/apellido para que se diriga sobre el recueadro de texto para escribir opc 1 inpunt fuera de label con for e id igual; opc 2 inpun dentro de label sin for ni id-->

<label>Apellido

<input type="text" placeholder="Ingrese Apellido" />

</label>

<br />

<label

>Dirección

<input type="text" placeholder="Ingrese Dirección" />

</label>

<label for="">Origen

<select name="lista">

<!--el name va ligado al value que se usa después en JS-->

<option value="A">A</option>

<option value="B">B</option>

<option value="C">C</option>

<option value="D">D</option>

</select>

</label>

<br />

<label for="">Acepta

<input type="checkbox" name="" id=""/>

</label>

<button>Enviar</button>

</form>

* Por último, se menciona el uso de las tablas. Donde se puede hace un cruce de etiquetas donde se inicia con **table**; la **<tr>** es la fila que significa y la **<td>** es la columna que significa define la celda y se muestra lo escrito y donde interactúa el usuario; así es la estructura como tal:

<table> <!--tabla normal abre con table, sigue: tr y el contenido se escribe en td→

<tr>

<td>#</td>

<td>Nombre</td>

<td>Edad</td>

</tr>

<tr>

<td>01</td>

<td>Camilo</td>

<td>24</td>

</tr>

<tr>

<td>02</td>

<td>Sebastian</td>

<td>26</td>

</tr>

<tr>

<td>03</td>

<td>Julio</td>

<td>25</td>

</tr>

</table>

1. **Estructuras tipográficas**
2. **Estructuras de contenido**