TALENTO DIGITAL
INTELIGENCIA HUMANA

Talento Digital para Chile:

Módulo 3
Fundamentos de Desarrollo Web

UN PROYECTO DE:

DESARROLLADO POR:

MÓDULO 3 – FUNDAMENTOS DE DESARROLLO WEB

3.5.- Contenido 5: Gestión del código fuente con GitHub

Objetivo de la jornada

- Entender la importancia de contar con un repositorio para los datos y administrarlo correctamente
- Conocer y aplicar conceptos como “commit”, “ramas”, “uniones”, “conflictos”, entre otros.
- Aplicar GitHub para administrar los proyectos creados en la nube

3.5.1.- Fundamentos de GIT

3.5.1.1.- ¿Por qué es importante un repositorio?

Git es un sistema de control de versiones distribuido, gratuito, de código abierto y eficiente [1], y se complementa con la plataforma **GitHub** para alojar repositorios de código. Entre las ventajas [2] que posee es posible destacar:

- **Versionamiento de código:** cada cambio en un proyecto se puede almacenar en versiones diferentes, lo que permite poder ir “hacia atrás” en caso que se desee revisar casos previos de un módulo o plugin. A diferencia
- **Aprender y probar cambios:** existen muchos proyectos que son públicos en GitHub, pudiendo experimentar con cambios sobre éstos, sin afectar el código original. Este último proceso es conocido como **fork**.
- **Contribuir:** Una vez que se han realizado cambios y testeados existosamente, se pueden enviar al propietario original a modo de aporte o mejora. Este proceso se llama **pull request**, y permite al encargado del repositorio validar la contribución, proponer mejoras, rechazar el cambio, o simplemente aceptarla para fusionarla con el proyecto original; este último escenario es conocido como **“merge”**.
- **Trabajo en equipo:** permite un excelente trabajo en equipo para el desarrollo de proyectos de diversa envergadura, incluso de alto impacto.
- **Visor de código:** GitHub tiene incorporada una herramienta que permite revisar el código almacenado en el repositorio, pudiendo comparar diversas versiones y volver a versiones anteriores de ser necesario.
- **Precio:** GitHub es completamente gratis e ilimitado para proyectos públicos, esto quiere decir que el código se podrá visualizar a través de la red. Se puede tener proyectos privados, pero de no más de tres colaboradores.

3.5.1.2.- Instalación, configuración y comandos básicos

La instalación de Git es un ambiente local es sencillo. Solo debe ir a la sección de descargas del sitio oficial [3], y seleccionar el sistema operativo de su equipo. Esto último lo detecta el navegador, y por defecto se muestra la última versión disponible del instalador.

Debe considerar que lo que se instala en este punto es solo la funcionalidad que permite administrar repositorios a través de comandos en un terminal, esto independiente del sistema operativo en uso. Si se desea tener una mejor experiencia a través de un entorno más ameno, existen programas que ayudan bastante a los desarrolladores, además de la integración que se puede hacer entre Git y diversos IDEs.

Lo primero que se debe hacer posterior a la instalación es la creación de un repositorio local. Para ello, siga los siguientes pasos:

- Cree un directorio para un proyecto
- En una consola de comandos, acceda al directorio
- Escribe el comando **`git init`**; con esto se inicializará un repositorio vacío.
- Ahora ingresa el comando **`git add [ARCHIVO]`**, lo cual agregará al listado de documentos del proyecto el documento de nombre **`[ARCHIVO]`**. Esto debe hacerlo por cada documento del proyecto, o bien seleccionarlos todos.
- Finalmente, para aplicar los cambios debes ingresar el comando **`git commit`**; el programa solicitará un texto con el detalle de los cambios aplicados. De no agregarse, no se podrá hacer el proceso.

A modo de pruebas, se recomienda crear un pequeño proyecto de una página web con un archivo CSS en un directorio independiente. La idea es que se puedan ir haciendo cambios y agregando nuevos documentos al proyecto.

3.5.1.3.- Commit y restauración de archivos

Tal como se indicó previamente, una de las características interesantes de Git es la posibilidad de volver a versiones anteriores de un archivo. Para hacerlo puede hacer uso de los comandos que se muestran a continuación:

- Realiza un cambio en el sitio del proyecto, idealmente el título y agregando un elemento nuevo.
- En un terminal o consola de comandos, abre el directorio del proyecto
- Ingresa el comando **`git add [ARCHIVO]`**, con lo cual se agregará el documento de [ARCHIVO] al listado de elementos para actualizar.

- Una vez realizado lo anterior, ingresa el comando **`git commit`**; al igual que en el caso anterior, se solicitará al usuario el detalle de las modificaciones realizadas.
- Finalmente, si el proceso se ejecutó correctamente informará la cantidad de elementos actualizados, incorporados y eliminados.
- Repite este proceso para otro documento del proyecto, y ten cuidado de hacer el "commit" final para que los cambios se vean reflejados.

Para restaurar el proyecto en primer lugar se debe revisar el registro de cambios del proyecto, lo cual se puede hacer a través del comando **`git log`**. Se desplegarán todas las versiones existentes desde la más reciente a la más antigua. Por cada versión almacenada se despliega un código alfanumérico llamado "**hash**" seguido de la palabra "**commit**". Una vez conocido este identificador, se debe ingresar el comando **`git checkout [IDENTIFICADOR]`**, donde **`[IDENTIFICADOR]`** es el código obtenido del hash; ten en cuenta que no es necesario agregar el hash, con los cuatro primeros es suficiente, siempre y cuando no exista redundancia en los identificadores. Lo que hace esta acción en lo concreto es mover la cabecera o puntero HEAD a otra versión del proyecto, y los cambios que se realicen a futuro serán en base a dicho caso. No debes olvidar que las otras versiones seguirán co-existiendo en el repositorio.

3.5.1.4.- Cambiar nombre de archivo

Renombrar un archivo en un proyecto de Git puede parecer trivial si creemos que se logra cambiando directamente el nombre del archivo; en realidad no es así. Si deseas lograr este objetivo, debes realizar los siguientes pasos:

- Por medio de una consola de comandos accede a la carpeta en la que está el proyecto.
- Usa un comando para cambiar el nombre el archivo: **`git mv nombre_original nombre_final`**.
- Con el comando **`git status`** puedes comprobar que se detectó el cambio de nombre.
- Para confirmar el cambio debes realizar el comando **`git commit`**. Antes de realizar la tarea pedirá ingresar un texto de forma obligatoria.
- Con el comando **`git log`** se pueden ver los cambios realizados. Deberá aparecer el último cambio incorporado.

3.5.1.5.- Ignorar archivos

Cuando se hace una actualización sobre un repositorio Git, previo a esa acción se debe agregar cada archivo usando el comando "**`git add [ARCHIVO]`**", y después "**`git`**

commit". Si se usa la opción "**git add .**" se subirían a la nueva versión todos los archivos que han sido creados o modificados sin distinción; por lo mismo, se hace necesario contar con un método que permita filtrar archivos que no se deseen agregar al repositorio.

Lo anterior se puede lograr creando y/o editando el archivo **.gitignore**, el cual en cada línea deberá contener nombres de archivos o patrones de nombres que deberá ignorar. Generalmente se ubica en la raíz del proyecto, sin embargo pueden existir múltiples archivos de este tipo, siendo los patrones o expresiones indicadas relativas a la ubicación del archivo.

En este ejemplo se ignorarán todos los archivos con extensión .log, excepto el archivo **example.log**.

```
*.log
!example.log
```

Ilustración 1: Archivo .gitignore - Ejemplo 1

Es posible asimismo bloquear un directorio completo; considerar que, si se bloquea un directorio, no se pueden desbloquear por separado sus archivos.

```
/logs
```

Ilustración 2: Archivo .gitignore - Ejemplo 2

Además es posible agregar comentarios en un archivo de este tipo. En el ejemplo siguiente se ignorará el archivo .DS_Store, propio del sistema operativo Mac y que, para efecto de compartir proyectos, puede resultar molesto.

```
# macOS Files
.DS_Store
```

Ilustración 3: Archivo .gitignore - Ejemplo 3

Anexo 1: Referencias

[1] Sitio oficial de Git
Referencia: https://git-scm.com/

[2] Diez razones para usar Github
Referencia: https://gist.github.com/erlinis/57a55dfb0337f5cd15cd