**Jessica Ortiz Ospina**

**Que aprendí en cada modulo y clase de Introducción a la informática?**

**Clase 1 - 18/03/2021**

En esta clase aprendí:

La historia de la informática, donde todo comenzó en el año 1122 a.c. en China con la invención del ábaco un aparato para realizar operaciones aritméticas, luego en 1642 d.C en Francia Blaise Pascal invento la primera calculadora mecánica la cual llamo Pascalina, luego en 1801 dc Joseph Jacquard creo las tarjetas perforadas en la cual podía guardar información, aprendí tambien q los primeros intentos en crear la primera computadora fueron con Charles Babbage y que un año después de su muerte fúe creada la primera computadora.

Aprendía un dato muy interesante que Ada Lovelace uná mujer matemática británica creo el primer algoritmo procesado por una máquina por lo es considerada como la primera persona en programar ordenadores, entre los años 1833 y 1842.

Un siglo después en 1936 fue creada la máquina de Turing q logra ejecutar programas almacenados creada por Alan Turing con el propósito de descifrar los mensajes alemanes en la segunda guerra mundial.

Siete años después fue creada la computadora Eniac en 1943 operada por las chicas del refrigerador y junto con Ada fueron las primeras programadoras de la historia.

Luego con IBM en 2011 ingresamos a la era cognitiva donde encontramos tecnologías como las computadoras cuánticas q puede ejecutar en 200segundos cálculos complejos donde una computadora convencional le puede tomar unos 10.000 años. Y ahora estamos apuntando a la inteligencia artificial.

**18 de Marzo Clase 2**

Aprendí que es una terminal : Es un programa q está presente en todos los sistemas operativos. Donde podemos ejecutar casi cualquier cosa en nuestro computador sin la necesidad de un mouse logrando esto a través como por ejemplo crear una carpeta o archivos, mover archivos eliminarlos etc.. de este modo podemos darle ordenes al sistema. Aprendí como abrir la terminal de Windows presionando las teclas win +R y escribiendo en la ventana “cmd.exe”, otra forma seria a través del buscador y escribiendo cmd.

Aprendí el uso de los comandos: **ls** (muestra los archivos de la carpeta)

**dir** ((muestra los archivos de la carpeta en la que estamos ubicados).

**cd . .** (retrocedemos una carpeta).

**rm nombreArchivo.extension** (Eliminar archivo).

Tambien aprendí el uso de los comandos **cd nombreCarpeta** , **mkdir nombreCaprenta** y el comando **touch nombreArchivo.extension** que es para crear un archivo según la extensión q coloquemos por ejemplo un .txt q es de texto.

Aprendí también el uso de la consola en VS code, aquí aprendí el uso de los comandos:

Pwd y los comandos q anteriormente cite.

**23 de Marzo Clase 3**

En esta clase aprendí todo lo relacionado con Git. Aprendí que Git es un sistema de control de versiones y es muy útil para trabajar junto con otras personas y tener varias versiones de un mismo proyecto y poder guardarlos en la nube y tener un control de quien y cuando realizo un determinado cambio en el proyecto, git es una excelente herramienta para trabajar de forma colaborativa.

Aprendí **como crear** mi repositorio **local** desde la terminal de bash me ubico en esa carpeta q quiero hacer el repositorio y escribo el comando **git init** (para inicializar un repositorio). Con esto se genero un repositorio local en la carpeta en q tenes los archivos.

Aprendí q los commits son como pequeños paquetes donde se irán almacenando los archivos, estos paquetes permitirán hacer un seguimiento de los cambios q vas realizando dado q cada uno de ellos tiene una fecha de creación y un autor. Los commits van hacer nuestro historial de cambios, es por eso q antes de cualquier cosa necesitamos decirle al repositorio quien sos voz, pq es esta información q le permite a Git hacer un completo seguimiento de los cambios realizados.

Aprendí como agregar mi identidad al repositorio con los comandos:

**git config user.name** “mi-usuario”

**git config user.email** [miCorreo@email.com](mailto:miCorreo@email.com)

Para corroborar q se agrego correctamente el email: git config user.email

Para no configurar todo esto todo el tiempo escribo lo siguiente:

**git config –global user.name “mi usuario”**

**git config –global user.emal** “[miCorreo@email.com](mailto:miCorreo@email.com)”

**Aprendí como agregar archivos al repositorio**

Los commits de git son los q nos permite hacer un control del seguimiento y control sobre las versiones de nuestro archivo.

**git add nombreDelArchivo.js : l**e tengo q indicar el o los archivos q quiero agregar.

**git status** :Me dirá el estatus de mis archivos y el estado de los mismos respecto al repositorio.

**git add . :El punto indica agregar todos los archivos presentes en el repositorio, archivos nuevos y archivos q hayan sufrido cambios.**

Si luego escribís git status podras ver todos los archivos q fueron agregados al repositorio y todos van a tener seguimiento

Nota: Siempre q se modifique un archivo debemos volver hacer **git add nombreDelArchivo.js**

## CONFIRMANDO ARCHIVOS

Los commit generan puntos cronológicos en la linea de tiempo del proyecto que nos permite identificar el estado del mismo hasta ese momento en específico.

El **git commit -m "Aqui va el mensaje"** Aqui genere una linea de tiempo de mi proyecto el cual contiene los archivos q se agregaron al repocitorio.

En VSC podemos visualizar q estos tres archivos estan untracked es decir sin seguimiento por eso la U. Tenemos q agregarlos al repositorio para luego cerrar un commit o cerrar un punto en nuestra linea de tiempo.

En VSC podemos visualizar q estos tres archivos estan untracked es decir sin seguimiento por eso la U. Tenemos q agregarlos al repositorio para luego cerrar un commit o cerrar un punto en nuestra linea de tiempo.

Agregamos los archivos recordar **con git add** . ( no olvidar espacio y luego el punto). Podemos verificar con git status, q debe aparecer una A de Agregado.

En caso de que un archivo ya existe dentro del repositorio pero este fue modificado, para subirlo al repositorio local digitamos en la terminal **git add nombreArchivo.js.**

**Y luego git commit -m “mensaje” //commitea los cambios hechos**

Después de ejecutar este comando dejara de aparecer en el estatus de VSC la letra M (Modified).

**25 de Marzo Clase 4**

**GitHub**

Repositorio: Donde se irán almacenando los archivos de nuestro proyecto a través del cual podremos hacerle seguimiento de los mismos. Y debemos tener en cuanta q a un proyecto le corresponde un repositorio.

En esta clase aprendimos como crear un nuevo repositorio en GitHub ubicándonos en nuestro repositorio remoto de GitHub dándole clic en la parte superior derecha en donde **esta el + y luego** selecciono New repository . Luego le doy el nombre al repositorio y coloco la descripción.

**Conectando nuestro repositorio local a GitHub**

* **Debemos inicializar un repositorio. Para esto, ejecutemos git init en la carpeta que queramos conectar el repositorio.**
* Luego tenemos que indicar al repositorio nuestros usuario ejecutando dos comandos:

1. git config [user.name](http://user.name) “mi usuario” (escribimos nuestro nombre de usuario).
2. git config user.email “miCorreo@email.com” (escribimos nuestra dirección de correo).

Siguiendo los pasos para crear nuestro repositorio

paso 3 Copiamos y pegamos el link q nos da github en gitbush.

**git remote add origin https://github.com.......**

Con el paso q sigue sincronizaremos nuestro repositorio local con el repositorio remoto el link q copiamos lo pegamos en bash y enter.

Para verificar escribimo en bash: **git remote -v**

**Subiendo Archivos en GitHub:**

**Usamos los comandos:**

Tener en cuenta q desde octubre del 2020 se cambio el nombre de la rama master para main. Dependiendo de q versión se tenga se ecribe uno de los siguientes comandos:

**git push origin master**

**git push origin main**

Una rama dentro de un repositorio es una copia alternativa hasta el momento como una línea paralela a través de la cual se van agregando nuevas funcionalidades sin tener q modificar la línea original del tiempo.

La rama es como una versión 2 de tu proyecto en la cual podrás probar cosas nuevas y q luego si quieres y funcionan la podes fusionar con la rama principal.

## Bajando archivos

## git clone : Permite crear una copia exacta en la computadora de todos los archivos existentes en un repositorio remoto. Recordar q este comando solo se ejecuta una sola vez y cuando no estén presentes en la computadora en la estoy trabajando.

## git pull origin master o git pull origin main: para bajar y actualizar aquellos archivos q halan sufrido algún tipo de modificación así como un archivo nuevo existente.

## Resolviendo conflictos

## Aprendí que una buena manera de evitar conflictos es mantener los commits relativamente pequeños y subir al servidor frecuentemente de esta manera tenemos menos chances de q ocurran conflictos y q si ocurren sean pequeños.

## Otra forma de evitar conflictos es el uso de ramas para trabajar en paralelo a la versión principal del código. Y estar en constante comunicación con el equipo para evitar esos conflictos.

Que es una interfaz nativa : Las interfaces nativas son las que vienen ya hechas para facilitarte un poco el trabajo a la hora de crear la aplicación. Tienen un aspecto ya definido en cuanto a color, medida o tipografía.

Que es una interfaz de lineas de comando :