Laboratorio miércoles 14 de Marzo

Introducción a Github

Objetivo

El objetivo de este es que tengan un primer encuentro con Git, una herramienta que utilizarán en todos los ramos de programación.

Git es una herramienta que permite ”sacar fotos” de su proyecto en determinado momento, estas fotos se llaman commits. La idea de Git es realizar commits cada cierto tiempo de manera de mantener un historial de su proyecto, lo cual le puede permitir recuperar el trabajo en caso de cualquier error. Git se instala en el computador y las ”fotos” se guardan en el mismo, eso si existen aplicaciones web que permiten guardar las ”fotos” en la nube, una de estas es Github.

Para que entienda más facil, piense en Git como Dropbox o Drive. Git permite mantener versiones de su proyecto en el tiempo, las cuales puede revisar o recuperar en caso de necesitarlo. La unica diferencia con las herramientas mencionadas es que Git no saca fotos con cada cambio, sino que usted define manualmente cuándo y qué archivos.

Otra ventaja de trabajar con Git y Github es que el trabajo en equipo se vuelve más fácil y robusto. Sus compañeros pueden subir los cambios y usted podrá revisarlos en su computador. En caso de que exista algun conflicto, por ejemplo, usted editó la linea 20 del código y su compañero también, Git le avisará y le pedirá ayuda para definir si dejar su línea o la de su compañero o ambas. Esto permite mayor control sobre lo que está pasando en el proyecto.

La idea es que tengan una primera aproximación a esta herramienta en este laboratorio. Eso si, primero instalen Git en su computador. https://git-scm.com/.

Problema

Deben seguir el paso a paso presentado a continuación:

1. Cree una cuenta en Github con su correo de la universidad. Su usuario debe ser lo más parecido a su nombre, de manera que en las evaluaciones pueda ser reconocido.

2. Cree un nuevo repositorio en Github con el nombre lab_1_OOP_202310. Este repositorio debe ser público y no es necesario que tenga README, .gitignore o license.

3. Cree una carpeta en su computador, agregue un archivo llamado answer.txt y escriba algo en él.

4. Abra en la terminal de su computador la carpeta que contiene el archivo answer.txt. (Asegúrese que esté en la carpeta correspondiente, sino va a inicializar git en otra parte).

5. Escriba *git init* en la consola. Este comando le dirá a git que esta carpeta sera un proyecto a versionar.

6. Escriba *git add .* (El comando va con el punto!. Este va a decirle a git, que debe versionar todos los archivos dentro de la carpeta. Uno también podría hacer git add answer.txt y esto le dirá a git, solo sigue el archivo answer.txt)

7. Vamos a hacer la primera foto. Escriba *git commit -m "first commit"*. El mensaje entre comillas puede ser cualquier cosa, la idea es que sea explicativo de lo que es su commit para que usted pueda entenderlo en el caso de necesitarlo en el futuro o sus compañeros.

8. Seleccione la rama Main. Oh! un nuevo concepto, ramas, estas permiten trabajar el código en versiones paralelas, es decir, no hacer todos los cambios en la versión principal, solo agregarlos cuando estén listos. Esto se utiliza mucho en proyectos grandes y en equipos. Se verá detenidamente en clases, por ahora solo preocúpese de trabajar en la rama Main con el siguiente comando: *git branch -M main*

9. Ahora vamos a conectar git con su proyecto en github. Para esto debe decirle a git a cual proyecto de github debe subir los cambios con el siguiente comando (este se ejecuta una sola vez): *git remote add origin <link al proyecto>*. El link lo pueden sacar de la página de su proyecto recién creado.

10. A continuación vamos a subir los cambios a la nube: *git push -u origin main*

11. Ahora, haga un cambio en el archivo answer.txt, haga un nuevo commit y suba los cambios. Revise que en github estén los cambios. Además, en github puede revisar los commits anteriores y vera su archivo antiguo!. Por otro lado, en la consola puede escribir: *git log*, que le mostrará la historia y podrá ver ambos commits.

12. Agregue un archivo llamado this_will_be_ignored.txt. Vamos a hacer que git ignore este archivo y no lo suba a github.

13. Agregue un archivo llamado .gitignore y dentro de el escriba this_will_be_ignored.txt. Este archivo se encarga de decirle a git cuales archivos debe ignorar y no subirlos a la nube. En general se ignoran archivos pesados (imágenes, etc), archivos con información sensible (claves, tokens, etc) y archivos de configuración.

14. Haga un commit y suba los cambios a github, si todo está bien, no aparecerá el archivo que queríamos ignorar en github.

Entrega

Suba al buzón habilitado en canvas el link a su proyecto antes del martes 21 a las 23:59. Se revisará que hayan realizado todos los pasos anteriores.