**Práctica: Desarrollo Colaborativo en GitHub con IntelliJ IDEA**

**Consigna**

**Realizar todos los pasos e ir pegando los resultados en un documento en Google drive accesible por todos los miembros del equipo.**

**Al finalizar el práctico copiar el enlace al documento y al github creado en un Notepad o Word que se subirá a la tarea.**

**Tareas**

Supongamos un equipo de desarrollo que trabaja en un proyecto llamado "MiApp" y hay que implementar una nueva función. Aquí están los pasos para un flujo de trabajo colaborativo:

**Paso 0 :** Consultar en <https://chat.openai.com/> para que sirve git compartir la consulta con sus compañeros y pegar la conclusión

**Paso 1: Creación del Repositorio**

Un miembro del equipo crea un nuevo repositorio "MiApp" en GitHub

*Invitar a Colaboradores:*

Una vez que creado el repositorio personal en GitHub al principio, invitar a otros miembros del equipo como colaboradores.

Ir a la página principal de tu repositorio personal en GitHub.

En la pestaña "Settings" (Configuración).

En la barra lateral izquierda, seleccionar "Manage access" (Gestionar acceso).

Hacer clic en el botón "Invite a collaborator" (Invitar a un colaborador).

Ingresar el nombre de usuario de GitHub o la dirección de correo electrónico de las personas a invitar como colaboradores.

Después de que el colaborador haya aceptado la invitación, verás su nombre de usuario en la sección "Collaborators" (Colaboradores).

Junto al nombre del colaborador, hay un botón desplegable que te permite configurar los permisos. Los permisos comunes incluyen:

Read (Lectura): El colaborador puede ver y clonar el repositorio, pero no puede hacer cambios.

Write (Escritura): El colaborador puede hacer cambios y realizar push, pero no puede eliminar el repositorio.

Admin (Administrador): El colaborador tiene control total sobre el repositorio, incluyendo la capacidad de configurar las opciones del repositorio, gestionar colaboradores y eliminar el repositorio. Ten precaución al otorgar permisos de administrador.

*Seleccionar el nivel de permisos:*

Hacer clic en el botón desplegable junto al nombre del colaborador y selecciona el nivel de permisos que deseas otorgar. Por ejemplo, seleccionar "Write" para permitir que el colaborador haga cambios y realice push al repositorio.

Confirmar la configuración de permisos:

Una vez que hayas seleccionado el nivel de permisos, confirmar la configuración haciendo clic en "Add [nombre del colaborador]" o en un botón similar.

El colaborador ahora tendrá los permisos y podrá realizar las acciones correspondientes en el repositorio. Se puede modificar estos permisos en cualquier momento si es necesario.

Hacer clic en "Add [nombre del colaborador]" para enviar las invitaciones.

Invitar colaboradores a tu repositorio personal les dará acceso para colaborar en el proyecto y les permitirá realizar cambios en el repositorio. Una vez que acepten la invitación, podrán clonar el repositorio, crear ramas y realizar push de sus cambios. Esto facilita la colaboración y el trabajo en equipo en tu proyecto personal en GitHub.

**Paso 2: Clonar el Repositorio**

Cada miembro del equipo clona el repositorio en su máquina local utilizando IntelliJ IDEA:

git clone <URL del repositorio>

cd MiApp

**Paso 3: Crear una Rama de Trabajo**

Cada miembro del equipo crea una rama de trabajo para la nueva función que van a implementar:

git checkout -b mi\_rama

Esta forma de crear una rama no es la única opción

***Otras formas de crear una nueva rama en Git:***

git checkout -b mi\_rama: Este comando crea una nueva rama llamada "mi\_rama" y cambia a esa rama al mismo tiempo. Es una forma abreviada de crear y cambiar a la nueva rama en un solo paso.

git branch mi\_rama seguido de git checkout mi\_rama: Puedes dividir la creación de la rama y el cambio a la rama en dos pasos separados. Primero, creas la rama con git branch mi\_rama y luego cambias a la rama recién creada con git checkout mi\_rama.

Ambas opciones son válidas y dependen de tus preferencias personales. La primera opción (git checkout -b mi\_rama) es más concisa y comúnmente utilizada cuando deseas comenzar a trabajar en una nueva característica o corrección de errores. La segunda opción (git branch seguido de git checkout) es más explícita y te permite dividir claramente la creación de la rama del cambio a la misma.

Se puedes optar por la opción que te resulte más cómoda y clara.

**Paso 4: Desarrollo en Ramas Individuales**

Cada miembro del equipo trabaja en su rama individual, implementando la nueva función en su máquina local.

**Paso 5: Hacer Commit de los Cambios Locales**

Después de implementar cambios significativos, cada miembro hace commit de sus cambios en su rama local:

git add .

git commit -m "Implementación de nueva función"

**Paso 6: Actualizar la Rama Principal (Opcional)**

Si la rama principal ha avanzado desde que se clonó el repositorio, es importante mantenerla actualizada. Esto se hace con:

git checkout main

git pull origin main

**Paso 7: Subir Cambios a GitHub**

Cada miembro sube su rama al repositorio de GitHub:

git push origin mi\_rama

**Paso 8: Creación de una Solicitud de Extracción (Pull Request)**

En GitHub, cada miembro del equipo abre una Solicitud de Extracción (Pull Request) desde su rama hacia la rama principal del repositorio "MiApp".

**Paso 9: Revisión y Comentarios**

Los otros miembros del equipo revisan la Solicitud de Extracción y proporcionan comentarios si es necesario. Pueden discutir los cambios directamente en GitHub.

**Paso 10: Fusionar Cambios**

Una vez que la Solicitud de Extracción es aprobada, el miembro que la creó puede fusionarla en la rama principal ("main") del repositorio en GitHub.

**Paso 11: Actualizar Ramas Locales (Opcional)**

Todos los miembros del equipo pueden actualizar sus ramas locales con los cambios fusionados:

git checkout mi\_rama

git pull origin main

**Paso 12: Repetir el Proceso**

El equipo puede repetir este proceso para implementar nuevas funciones o realizar correcciones de errores en el proyecto.

Este flujo de trabajo permite a tu equipo colaborar de manera efectiva en el desarrollo de software, manteniendo un registro de cambios y permitiendo revisiones y comentarios antes de fusionar los cambios en la rama principal del repositorio. Además, IntelliJ IDEA facilita la gestión de Git y GitHub en este proceso.

**Paso 13 :** Agregar al documento con los pasos las conclusiones con lo que aprendimos en esta práctica.