TP2-PROGRAMACION

Nombre: Matias Dagostino

Preguntas

1)

¿Qué es GitHub?

GitHub es una plataforma de alojamiento de código basada en la web que utiliza el sistema de control de versiones Git. Permite a los desarrolladores colaborar en proyectos, gestionar código, realizar seguimiento de problemas y mucho más. Es como una red social para programadores donde pueden compartir y trabajar juntos en proyectos.

¿Cómo crear un repositorio en GitHub?

1. Inicia sesión en tu cuenta de GitHub
2. Haz clic en el botón "+" en la esquina superior derecha y selecciona "New repository"
3. Dale un nombre a tu repositorio
4. Añade una descripción opcional
5. Elige si será público o privado
6. Puedes inicializarlo con un README
7. Haz clic en "Create repository"

¿Cómo crear una rama en Git?

Puedes crear una rama con el comando:

git branch {nombre}

¿Cómo cambiar a una rama en Git?

Usa el comando:

git checkout {nombre}

¿Cómo fusionar ramas en Git?

1. Primero, cámbiate a la rama que recibirá los cambios (normalmente main o master):

git checkout main

2. Luego fusiona la otra rama:

git merge {nombre}

¿Cómo crear un commit en Git?

1. Primero añade los cambios:

git add . # Para todos los archivos cambiados

2. Luego crea el commit con un mensaje:

git commit -m "Mensaje del cambio"

¿Cómo enviar un commit a GitHub?

Después de hacer commit, usa:

git push origin {nombre}

Por ejemplo, para la rama main:

git push origin main

¿Qué es un repositorio remoto?

Un repositorio remoto es una versión de tu proyecto que está alojada en Internet o en otra red. Te permite colaborar con otros desarrolladores y mantener una copia de respaldo de tu trabajo.

¿Cómo agregar un repositorio remoto a Git?

Usa el comando:

git remote add origin {url}

¿Cómo empujar cambios a un repositorio remoto?

git push origin {nombre}

¿Cómo tirar de cambios de un repositorio remoto?

git pull origin {nombre}

Esto descargará los cambios y los fusionará automáticamente con tu rama local.

¿Qué es un fork de repositorio?

Un fork es una copia personal de un repositorio de otro usuario. Te permite experimentar con cambios sin afectar el proyecto original.

¿Cómo crear un fork de un repositorio?

1. Ve al repositorio que quieres copiar en GitHub
2. Haz clic en el botón "Fork" en la esquina superior derecha
3. Espera a que GitHub cree la copia en tu cuenta

¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

1. Haz cambios en tu fork y envíalos a GitHub
2. Ve a la página de tu fork en GitHub
3. Haz clic en "Pull request"
4. Selecciona las ramas adecuadas (tus cambios vs el original)
5. Escribe una descripción clara de tus cambios
6. Haz clic en "Create pull request"

¿Cómo aceptar una solicitud de extracción?

1. Ve a la pestaña "Pull requests" del repositorio
2. Selecciona la solicitud que quieres revisar
3. Revisa los cambios y los comentarios
4. Si todo está bien, haz clic en "Merge pull request"
5. Opcionalmente, confirma la fusión

¿Qué es una etiqueta en Git?

Una etiqueta (tag) es una referencia estática a un punto específico en el historial de Git.

¿Cómo crear una etiqueta en Git?

git tag {nombre}

¿Cómo enviar una etiqueta a GitHub?

git push origin {nombre}

¿Qué es un historial de Git?

El historial de Git es el registro completo de todos los commits realizados en un repositorio, mostrando quién hizo qué cambios y cuándo.

¿Cómo ver el historial de Git?

git log

¿Cómo buscar en el historial de Git?

Puedes usar:

git log --grep="texto a buscar"

¿Cómo borrar el historial de Git?

1. Crear un nuevo repositorio

2. Copiar solo los archivos actuales
3. Hacer un commit inicial

¿Qué es un repositorio privado en GitHub?

Un repositorio privado es aquel que solo pueden ver y modificar tú y las personas que invites explícitamente. Los repositorios privados son ideales para proyectos personales o trabajo confidencial.

¿Cómo crear un repositorio privado en GitHub?

Al crear un repositorio, selecciona la opción "Private" en lugar de "Public". En cuentas gratuitas, GitHub ahora permite repositorios privados con algunas limitaciones.

¿Cómo invitar a alguien a un repositorio privado en GitHub?

1. Ve a la página del repositorio
2. Haz clic en "Settings"
3. Selecciona "Manage access"
4. Haz clic en "Invite a collaborator"
5. Introduce el nombre de usuario o email de la persona
6. Haz clic en "Add collaborator"

¿Qué es un repositorio público en GitHub?

Un repositorio público es visible para cualquier persona en Internet, aunque solo los colaboradores pueden hacer cambios directamente. Es ideal para proyectos de código abierto.

¿Cómo crear un repositorio público en GitHub?

Al crear un repositorio, selecciona la opción "Public". Esta es la configuración predeterminada.

¿Cómo compartir un repositorio público en GitHub?

Simplemente comparte la URL del repositorio. Cualquiera podrá verlo y clonarlo. Para permitir que otros contribuyan, pueden hacer fork del repositorio y enviar pull requests.

Actividades

Actividad 1: Crear y configurar un repositorio

1. **Crear un repositorio público**:
    - Ve a GitHub y haz clic en "New repository"

- Dale un nombre como "mi-primer-repo"
- Selecciona "Public"
- Marca "Initialize this repository with a README"
- Haz clic en "Create repository"

2. **Agregar un archivo**:
   - Clona el repositorio a tu máquina:

```
git clone https://github.com/tuusuario/mi-primer-repo.git
cd mi-primer-repo
```

   - Crea un archivo "mi-archivo.txt":

```
echo "Hola mundo" > mi-archivo.txt
```

   - Añade y haz commit del archivo:

```
git add mi-archivo.txt
git commit -m "Agregando mi-archivo.txt"
```

   - Sube los cambios:

```
git push origin main
```

3. **Crear una rama**:
   - Crea una nueva rama:

```
git checkout -b nueva-rama
```

   - Haz cambios o añade otro archivo
   - Haz commit de los cambios:

```
git add .
git commit -m "Cambios en nueva rama"
git push origin nueva-rama
```

Actividad 2: Resolución de conflictos

Seguí todos los pasos detallados en el práctico para crear un conflicto y resolverlo:

1. Creé el repositorio "conflict-exercise" en GitHub con un README.md
2. Lo cloné a mi máquina local
3. Creé una rama "feature-branch" y modifiqué el README.md

4. Volví a main, hice otro cambio en el mismo archivo
5. Intenté fusionar las ramas, lo que generó un conflicto
6. Resolví el conflicto editando manualmente el archivo, manteniendo ambos cambios pero en líneas diferentes
7. Completé el merge y subí los cambios a GitHub

La resolución del conflicto fue educativa porque me mostró exactamente cómo Git marca los conflictos y cómo decidir qué cambios mantener.