Cómo generar aportes

Proyecto Informática II

- **Autores:**
  Facundo Zacco (Leg. 97663)
  Joaquín Salas (Leg. 98368)
  Mercadal Juan Ignacio (Leg. 95561)
  Gatica Marcos Raúl (Leg. 402006)

- **Curso:** 2R4
- **Asignatura:** Informática II.
- **Institución:** Universidad Tecnológica Nacional - Facultad Regional de Córdoba.

UTN FRC

Índice

1. Flujo de trabajo

En el siguiente documento se explica la metodología de trabajo para el proyecto "Prototipo de sistema de seguridad en cámaras frigoríficas", con el fin de organizar y separar las diferentes áreas de desarrollo, facilitando que cada integrante o aportes voluntarios puedan subir sus modificaciones o generarlos. El motivo de esta metodología surge con el fin de evitar conflictos entre cambios no relacionados y manteniendo el historial de trabajo estructurado.

1.1. main

Rama principal del proyecto. Aquí se integran los cambios aprobados y estables. Todo lo que se llega a *main* debe haber pasado por un proceso de revisión. Es importante mantener esta rama libre de errores, ya que representa la versión final y funcional del proyecto.

1.2. devel

Esta es la rama de desarrollo principal. Todo el código relacionado con el software del proyecto se trabaja aquí. El objetivo de esta rama es que cada integrante pueda hacer sus aportes y pruebas antes de integrarlo a *main*. Al trabajar en *devel*, nos permite implementar y probar nuevas funcionalidades sin afectar la estabilidad del proyecto.

1.3. hardware

Es la rama dedicada al desarrollo y documentación del hardware. Todos los cambios que afectan las especificaciones, la configuración o la implementación del hardware se gestionan en esta rama. Esto permite que el equipo de hardware pueda trabajar de forma independiente sin interferir con los desarrollos de software y documentación.

1.4. docs

Esta rama está destinada a la creación y mantenimiento de la documentación del proyecto. Los manuales, guías de instalación, y cualquier otro tipo de documentación necesaria se alojan aquí. Separar la documentación en su propia rama garantiza que los desarrolladores puedan trabajar en el código sin preocuparse por las modificaciones en los documentos, y viceversa.

2. Por qué de estas ramas

2.1. Organización

Al tener ramas dedicadas para cada parte del proyecto (software, hardware, y documentación), podemos mantener un orden claro sobre qué se está trabajando y dónde. Esto evita confusiones y reduce la probabilidad de mezclar cambios de áreas diferentes.

2.2. Flujo de trabajo eficiente

Al trabajar en ramas específicas, los colaboradores pueden implementar y probar sus cambios sin comprometer la estabilidad del proyecto en *main*. Solo cuando los cambios han sido revisados y validados, se hacen los *Pull Requests (PRs)* para integrarlos a *main*.

2.3. Control de versiones

Este esquema permite tener un registro claro de cada cambio hecho en el proyecto. Al crear PRs y hacer merges entre ramas, se mantiene un historial detallado de quién hizo cada contribución y cuándo. Esto es especialmente útil cuando hay que rastrear la fuente de un problema o realizar revisiones de código.

2.4. Colaboración y revisión

El uso de Pull Requests permite que cualquier cambio sea revisado por el equipo antes de ser integrado a la rama principal (*main*). Esto asegura que todos los aportes sean revisados y aprobados, fomentando un mejor control de calidad en el código, la documentación y el hardware.

3. Flujo típico de trabajo con estas ramas

3.1. Desarrollo separado

- Cada colaborador trabaja en su área específica en alguna de las ramas (*devel, hardware, docs*), según corresponda.
- Cada colaborador debe notificar el comienzo y fin de su trabajo.
- Los desarrolladores pueden hacer *push* directamente a sus respectivas ramas de trabajo.
- Se realizan los commits correspondientes para reflejar los avances.

3.2. Revisión e integración a *main*

- Los supervisores del proyecto revisan periódicamente los cambios en las ramas de desarrollo.
- Cuando los cambios en una rama de desarrollo (*devel, docs, hardware*) están listos y probados, un supervisor abre un PR desde la rama de desarollo, hacia la rama principal (*main*).
- El equipo principal de desarrollo revisan los cambios propuestos, los comentarios y, si es necesario, solicita ajustes.

3.3. Integración con *main*

Una vez aprobado el PR, los cambios se fusionan a *main* asegurando que el proyecto se mantenga actualizado con los aportes más recientes.

4. Trabajo con GNU/Linux

Esta sección contiene la explicación sobre cómo trabajar con GitHub usando la terminal en sistemas operativos GNU/Linux.

4.1. Instalación de Git

Asegúrese de tener **git** instalado en su sistema. Opcionalmente puede instalar GitHub CLI en su sistema para hacer los PR desde su consola:
Sistemas basados en Debian:

```
apt update
apt full-upgrade
apt install git gh
```

Sistemas basados en Arch Linux:

```
pacman -Syu git github-cli
```

Una vez instalado en su sistema, ya puede clonar el repositorio remoto del proyecto. Recomendamos usar SSH con este link:

```
git@github.com:marcosgatica2003/
    proyectoInformaticaII.git
```

Nota: va a necesitar una clave SSH en su computadora para poder clonar correctamente el repositorio y hacer sus aportes, puede generarla revisando esta documentación de GitHub:

```
https://docs.github.com/es/authentication/
      connecting-to-github-with-ssh
/generating-a-new-ssh-key-and-adding-it-to-
              the-ssh-agent
```

4.2. Configuración

Una vez clonado el repositorio, necesita generar la configuración para poder subir correctamente sus aportes. Git utiliza el nombre de usuario y el correo electrónico configurados para asociar los commits con la identidad del desarrollador. Estos valores se pueden establecer de la siguiente manera:

```
# Establecer nombre de usuario
git config --global user.name "Tu Nombre
    "

# Establecer correo
git config --global user.email "Tu
    correo"

# Establecer un editor de texto (VIM)
git config --global user.editor vim

# Puede ver los cambios con:
git config --list
```

Después de ejecutar estos comandos, Git usará el nombre y el correo electrónico proporcionados para identificar las contribuciones en todos los repositorios locales.

4.3. Subir cambios al repositorio

Antes de realizar cualquier cambio, es necesario cambiarse a la rama en la que se desea trabajar. El siguiente comando cambia a una rama existente, por ejemplo *devel*:

```
git checkout devel
```

Una vez en la rama correcta, puedes realizar cambios en los archivos del proyecto. Después de hacer las modificaciones, sigue estos pasos para agregar y confirmar los cambios:

```
# Agregar los cambios
git add .
```

A partir de aquí sus cambios están preparados para ser agregados a un commit, puede hacerlo con:

```
# Realizar un commit con un mensaje
    descriptivo
git commit -m "su mensaje breve"

# Abrir el editor VIM y escribir su
    mensaje
git commit
```

Para compartir los cambios con el resto del equipo, debes subir la rama de trabajo al repositorio remoto. Esto se hace con el siguiente comando (suponiendo que está trabajando en la rama devel):

```
git push origin devel
```

Una vez que los cambios se han subido al repositorio remoto y haya pasado la revisión, el supervisor puede generar un Pull Request y combine con la rama principal.

4.4. Generar un Pull Request - supervisores

Para generar un pull request, es necesario utilizar la herramienta **GitHub CLI**, como se explicó al principio de la sección **TRABAJO CON GNU/LINUX**:

```
# Crear un PR desde la rama actual
    hacia main
gh pr create --base main --title
"Detalles del PR" --body
"Detalles del cambio"
```

A partir de aquí, los cambios de una rama de desarrollo están aplicados a la rama *main*.

5. Manejo de Issues

5.1. Creación y asignación de Issues

Los Issues son la herramienta principal para el seguimiento de tareas y problemas en el proyecto. Pueden representar:

- Errores que necesitan ser corregidos.
- Nuevas funcionalidades por implementar.
- Mejoras en el código existente.
- Tareas de documentación.
- Implementación de módulos de hardware.

Cada Issue puede contener:

- Una descripción clara del problema o tarea.
- Los requisitos necesarios para resolverlo.
- Cualquier información técnica relevante.

5.2. Flujo de trabajo con Issues

Para trabajar en un Issue asignado, se debe seguir el siguiente procedimiento:

1. Asegúrese de estar en la rama de desarrollo correspondiente a lo que detalla el Issue:

```
#Ejemplo con devel
git checkout devel
```

2. Cree una nueva rama sobre la rama de desarrollo con el nombre “fix-issue#XX”, siendo el número “XX”, el que asigna GitHub al Issue:

```
#Ejemplo con un issue #12
git checkout -b fix-issue#12
```

3. A partir de aquí puede trabajar en su rama fix-issue#XX, resulva los detalles descriptos en el issue y haga los commits correspondientes al finalizar su trabajo.

4. Subir los cambios:

```
git add .
git commit -m "fix-issue#XX: (
    detalles opcional)"
git push origin fix-issue#XX
```

5. Una vez completado y subido su trabajo, un supervisor revisará su aporte y puede generar un pr a la rama de desarrollo correspondiente, o puede solicitar un ajuste sobre la misma interfaz de comunicación del issue en la página de GitHub.

   Nota a los supervisores: se usa el mismo comando para generar un pull request a main, solo que en este punto es a una rama de desarrollo. También se puede generar otro pr a main posteriormente a la resolución del issue.

6. Tras haber resuelto el issue y haber subido su trabajo, es necesario borrar su rama con el nombre ”fix-issue#XX” para mantener únicamente las ramas de desarrollo.