Convenciones usadas en Git:

Commits:
-Cada commit debe realizar un cambio específico y aislado.
-El título debe ser conciso (menos de 50 caracteres) y en tiempo presente.
-La descripción puede proporcionar más detalles sobre el cambio.
-Títulos recomendados para usar en commits:

- **feat**: se usa para introducir una nueva funcionalidad.
- **fix**: se usa para corregir un error o bug.
- **docs**: se usa para cambios en la documentación.
- **style**: se usa para cambios que no afectan la lógica del código (formato, espacios, etc.).
- **refactor**: se usa para cambios en el código que no corrigen errores ni añaden funcionalidades, pero mejoran la estructura.
- **test**: se usa para añadir o corregir pruebas.
- **chore**: se usa para tareas de mantenimiento o cambios en la configuración.
- **revert**: se usa si se necesita revertir a un commit anterior, o se usa dentro del commit para mostrar dónde se revierte un cambio.

Branches:
-Usar nombres cortos y descriptivos.
-El nombre descriptivo de la rama debe ser imperativo.
-Los identificadores de problemas, bugs, Historias de Usuario o tickets podrán ser referenciadas en el nombre de las ramas.
-Nombrar las ramas en minúscula, a excepción de identificadores externos.
-Usar guiones para separar las palabras.
-Eliminar la branch del repositorio después de haber sido integrada a
a menos que haya una razón específica de no hacerlo.
-Títulos a usar en las branches:

- **master/main**: siempre debe reflejar el código en producción.
- **develop**: para integrar nuevas funcionalidades antes de pasarlas a producción.
- **feature/branch-name**: para desarrollar una nueva funcionalidad.
- **hotfix/branch-name**: para corregir errores urgentes en producción.

Merges:
-Antes de hacer *merge* de una rama, realizar un *rebase* para mantener un historial de commits más limpio y lineal.
-Cuando sea necesario usar un *merge commit*, asegurarse de que el mensaje sea descriptivo.
-Si una Branch tiene conflictos, resolver todos los conflictos localmente y verificar que el código funciona antes de realizar el *merge*.

-Si la rama de destino no ha cambiado, usar un *fast-forward merge* para evitar crear un nuevo *merge commit*.
-Siempre revisar los cambios que se van a fusionar, preferiblemente mediante una *pull request* o *merge request*.
-Cuando se realice un *merge* de una *feature branch* que ha sido desarrollada de manera significativa, usar la opción **--no-ff** para crear un *merge commit*.

Simulación de 5 commits:

1.

Título:

feat: se añade módulo de autenticación de usuario.

Descripción:

- Se implementa autenticación basada en JWT.
- Se crea la función de Login y Registro.
- Se implementa la administración de roles de usuario.

2.

Título:
fix: se resuelve el problema de tiempo de espera de la API

Descripción:

- Se configura un mayor tiempo de espera para la llamada de la API externa.
- Se optimiza el manejo de solicitudes para reducir el tiempo de procesamiento.

3.

Título:
docs: se actualiza el README con instrucciones para el configurado

Descripción:

- Se añaden instrucciones detalladas para la configuración del proyecto de manera local.
- Se incluyen tips para la resolución de problemas comunes.

4.

Título:
style: se mejora el formato del código

Descripción:

- Se aplica un formato consistente para el código a través de todo el proyecto.
- Se adhiere a las sugerencias de estilos recomendadas.

5.

Título:
refactor: se simplifica la lógica de la validación para la entrada del usuario

Descripción:

- Se refactorizan las funciones de validación para reducir código duplicado.
- Se optimiza el manejo de errores referente a los ingresos inválidos.