Pull/Merge Request bajo Modelo Git Flow.

El *Pull* o *Merge Request (PR/MR) "solicitud de extracción"* es un mecanismo en sistemas de control de versiones *(Git, GitHub, GitLab, Bitbucket)* que permite a un desarrollador proponer cambios en un repositorio. El *PR* notifica a otros colaboradores sobre cambios realizados listos para ser revisados, discutidos y eventualmente fusionados con *git merge* con la rama principal *(ej: main o master)*.
Dentro de los usos m{as comunes se encuentran:

1. ***Revisión de código**:* Permite a otros miembros del equipo comentar, sugerir mejoras o aprobar cambios.
2. ***Integración controlada**:* Evita fusiones directas a la rama principal sin consenso.
3. ***Documentación**:* Ayuda a registrar el historial de cambios.
4. ***Pruebas automatizadas**:* Hay proyectos que ejecutan pipelines de CI/CD *(GitHub Actions)* al crear un PR para validar cambios.

Elementos dentro del Modelo Git Flow.

Los repositorios se suelen dividir en 3 repositorios principales, cada uno independiente del anterior.

- *producción*
- *QA*
- *desarrollo*

Cada desarrollador debe crear su propia rama tomando como base a *"desarrollo"*, debe existir un *"Maintainer"* o *administrador* por proyecto y su papel será revisar y aprobar cada *Merge Request (MR)* entre ramas base (Integración).

Convención de Nombres en Branches.

Para el nombre de cada *rama/branch* se debe usar la siguiente estructura: ***[token]/[nombre descriptivo]***
Los tokens bajo convención son:

- ***chore:*** Mejoras administrativas/mantenimiento del proyecto. *Ej.: Actualización de dependencias.*
- ***docs:*** Creación/actualización de documentación. *Ej.: Guía de configuración del proyecto.*
- ***feature:*** Nueva funcionalidad dentro del proyecto. *Ej.: Un nuevo módulo o nueva función.*
- ***fix/hotfix/patch:*** Corrección de un bug esperado o inesperado. *Ej. Corrección de Links rotos.*
- ***refactor:*** Mejoras/refactorización de *features* existentes. *Ej. Creación y actualización de servicios).*
- ***test:*** Agrega *tests* a un *feature* existente. *Ej. JUnit testing del módulo de Aclaraciones.*

El nombre descriptivo asignado a la rama debe ser con palabras cortas que describan la tarea ***(sin usar punto final):***

- ***chore/****Actualización dependencias*
- ***docs/****Configuración*
- ***feature/****LDAP Login*
- ***fix/****Generación PDF*
- ***refactor/****Creación de Menú*
- ***test/****Pago TCB*

Convención de Nombres en Commits.

Los mensajes dentro de un *commit* pueden llevar la siguiente estructura: ***[token](<optional scope>): [description]***
El segmento *token* será similar a los usados en los nombres de *branches* (*chore*, *docs*, *feature*, *fix*/*hotfix*/*patch*, *refactor*, *test*).
El *scope* (Opcional) hace referencia a la parte del proyecto o aplicativo donde se pretenden insertar cambios.
La descripción debe cumplir con las siguientes reglas:

- Idioma según estándar interno, modo imperativo y tiempo presente *(cambio ✓; cambiado ✗)*;
- Primera letra mayúscula (Por estándar de Git al hacer *merge* o *rebase*);
- Máximo de 50 caracteres;
- No usar punto al final.

Los mensajes *commit* no deben describir acciones trabajadas, sino acciones que el proyecto recibe presente y tercera persona:

✗ *"Se agregó el campo Github en la configuración del perfil."*
✗ *"Se agrega el campo Github en la configuración del perfil"*
✓ *feature (configuración): Agrega campo Github a la configuración del perfil*
✓ *feature: Agrega notificaciones por correo electrónico sobre nuevos mensajes directos*
✓ *fix: Agrega parámetro faltante en llamada a servicio*

Pasos Sugeridos para un Ejecutar un Pull/Merge Request.

1. Preparar el entorno local:

- Obtener URL de Repositorio remoto por clonar.
- ***cd [ruta donde se almacenará el repositorio localmente]*** Abrir dirección local desde *Terminal/CMD*.
- ***git clone [https://github.com/usuario/repositorio.git]*** Clonar el repositorio.
- ***cd [repositorio clonado]*** Abrir dirección del repositorio clonado.
- ***git checkout -b [nombre-de-rama]*** Crear rama nueva (nunca sobre *main*).

2. Realizar los o asignación en el código;

3. Confirmar los cambios (commit):

- ***git status*** Muestra archivos modificados.
- ***git diff*** Muestra diferencias específicas.
- ***git add .*** Añade todos los cambios.
- ***git commit -m "mensaje"*** Mensaje claro y descriptivo.

4. Subir los cambios al repositorio remoto

- ***git push -u origin [nombre-de-rama-paso1]*** Primer *push* que vincula rama local y remota.

5. Crear el Pull/Merge Request en GitHub/GitLab:

- **Ir a repositorio desde manejador** (en GitHub, buscar botón *"Compare & Pull Request")*.
- **Completar campos**:
  - **Título**: Descriptivo (ej: "Fix: Error de autenticación en el login").
  - **Descripción**: Explica el **qué**, **por qué** y **cómo** de los cambios. Usa formato Markdown.
    Markdown
    Cambios realizados:
    - Se corrigió la validación del campo de email en el formulario de login.
    - Se añadieron pruebas unitarias para el caso de error.
    Issue relacionado
    Fixes -123 - Vincula PR con un issue.
  - **Selecciona la rama base** *(ej: main)* y la rama de cambios por añadir.
  - **Asigna revisores** (team members).

6. Revisión y aprobación:

- **Los revisores**:
  - Comentan en líneas específicas del código.
  - Aprueban (Approve) o solicitan cambios (Request changes).
- **Autores del PR**:
  - Hacen nuevos commits para abordar los comentarios.
  - Los cambios se reflejan automáticamente en el PR.

7. Merge (Fusión):

- **Opciones de merge**:
  - **Merge commit**: Crea un nuevo commit de fusión (recomendado para historial claro).
  - **Squash and merge**: Combina todos los commits del PR en uno solo (útil para limpieza).
  - **Rebase and merge**: Reaplica los commits sobre la rama base (evita commits de merge).
- **Eliminar la rama**: Opcionalmente, se puede borrar la rama después del merge.