## SECCIÓN 3: ESTRATEGIAS AVANZADAS DE GIT Y DEVOPS

### CONTEXTO:

En Podemos Progresar estamos buscando implementar prácticas DevOps sólidas para nuestros repositorios de infraestructura y ETL. Actualmente nuestro proceso carece de un enfoque estructurado, lo que dificulta el control de cambios en nuestros procesos críticos de microcréditos.

### Escenario:

Como líder técnico, debes definir la estrategia de branching y flujo de trabajo de Git basado en tu experiencia real implementando estos flujos en organizaciones financieras.

### Tareas:

3.1. Estrategia de branching para entornos de microfinanzas:

- Comparte una estrategia avanzada de branching que hayas implementado en tu experiencia para:

**R.**En mi experiencia en Walmart los Branch se organizaban en 5 capas, rama local , rama de dev, rama QA, rama de Preprod para pruebas y validaciones finales y main Produccion

\* Gestionar configuraciones para diferentes productos financieros o tipos de préstamos

**R.**En mi experiencia, recomendaría un repositorio donde se almacenen el core, config(archivo de configuraciones, conexiones,variables),params.yamls, workflows, entornos,scripts, usar GCP Secret Manager, pruebas unitarias

\* Manejar releases coordinados entre componentes dependientes (por ejemplo, cambios en cálculo de intereses que afectan múltiples sistemas)

**R.**En Walmart utilice branching para cada cambio, desarrollo en feature Branches cada equipo trabajaba en su repositorio, publicación del Branch en QA, pruebas end-to-end, commit en preprod, merge en main

\* Balancear la necesidad de agilidad con los requisitos de control y auditabilidad propios del sector financiero

- Proporciona ejemplos específicos de:

\* Estructura de branches que implementaste en un proyecto financiero real

En Walmart, pipeline de ETL para segmentación de Clientes

├── main *# Solo releases aprobados por arquitectura, líder proyecto,etc*

├── develop *# Integración continua*

├── release/\* *# Preparación de versión*

├── feature/\* *# Desarrollo explicando el desarrollo nuevo o cambio*

├── hotfix/\* *# Correcciones urgentes (merge a main + develop)*

\* Políticas de protección que configuraste para cumplir con requerimientos de auditoría

**R.** El área de Arquitectura de Walmart implemento Branch Protection Rules Approvals obligatorios: 2 reviewers (incluyendo al menos 1 del equipo de Compliance).

Checks requeridos:

-CI/CD exitoso (tests + escaneo de seguridad).

-Validación de firma.

-Aprovals Gerente, en caso de campo urgente

- Bloqueo de force push.

\* Adaptaciones que tuviste que hacer al trabajar en un entorno regulado

**R.**En Walmart Cumplir con las mejores prácticas, un Checklist para Cumplimiento por ejemplo

Commits firmados (GPG o SSH).

Plantillas de PRs con campos obligatorios.

Branches compliance

Integración con herramientas de trazabilidad

Validación automática de documentación en CI/CD.

3.2. Resolución de conflictos en el mundo real de servicios financieros:

- Describe dos situaciones complejas de conflictos de Git que hayas enfrentado en proyectos de tecnología financiera:

\* ¿Cómo gestionaste cambios conflictivos en componentes críticos para el negocio?

**R.** En Walmart, un merge genero un conflicto en el reporte de ventas. Se realizo la detección rápida

git merge --abort # Detener el merge conflictivo

git checkout HEAD~1 – venta\_intereses.py # Revertir al estado estable

Reunion urgente con ambos equipos, usamos GitHub Conflict Editor, Priorizar la fórmula implementada por equipo de negocio (Equipo A) y aplicar el redondeo de los intereses después (Equipo B), merge manual.

En soriana se presento un conflicto en el modelo de Riesgo, los pasos a seguir fueron la recuperación de la versión estable, reconstrucción manual para comparar versiones ejecutamos el modelo con datos históricos para asegurar consistencia

\* ¿Qué procesos implementaste para evitar la recurrencia de estos problemas?

**R.**Estrategias de branching estricta por ejemplo para Walmart , Ownership de Archivos Críticos para  cualquier PR que modifique estos archivos requiere aprobación explícita de los dueños. Usamos branches por componente (feature/intereses-\*, feature/redondeo-\*) en lugar de branches genéricas. Si dos equipos van a realizar un cambio al mismo componente deben coordinarse, Git Hooks para Detectar Conflictos Potenciales en  Soriana, este hook evitó que un conflicto en un DAG de Airflow llegara a main y pruebas de integracion

\* ¿Cómo balanceaste la velocidad de desarrollo con la seguridad en el proceso?

**R.**En entornos financieros o Retail los cambios rápidos pueden generar riesgos. Algunas de las estrategias implementadas CI/CD esto para solo ejecutar cambios en componentes críticos los PR de bajo riesgo se realiza el merge rápido, también se implemento el paralelismo controlado con branches por ejemplo para Correciones UI la rama feature/ui-\*, requisitos 2 approval + tests unitarios, pero cambios que representan un riesgo la rama era feature/clientes\_riesgo-\*, requisitos 3 approvals + SAT validation, donde uno de los approvals era el encargado del área de arquitectura. El objetivo era automatizar lo repetitivo pero manteniendo controles manuales para situaciones criticas.