1. ¿Qué factores clave (tamaño del equipo, frecuencia de releases, criticidad del proyecto) deberían influir en la elección de una estrategia de branching (GitFlow, Trunk-Based Development, etc.)?

Factor Equipo estrategia de branching: GitFlow/Release Branches mejor control para equipos grandes
Factor Frecuencia de releases estrategia de branching GitFlow/Release Branches: Trunk-Based Development para releases muy frecuentes
GitFlow/Release Branches para releases espaciadas
Factor Criticidad del proyecto, estrategia de branching: para alta criticidad el Trunk-Based Development presenta un riesgo e ideal para alta automatizacion
GitFlow/Release Branches ideal para alta criticidad y alta automatizacion.

2. ¿Cuáles son los pros y los contras principales de adoptar una estrategia de branching con ramas de larga duración (como develop o feature largas) frente a ramas de corta vida y integración frecuente en main?

Pros de estrategia de branching de branch de larga duracion:

Aislan cambios grandes sin afectar main.

Facilitan estabilizar versiones antes del release.

Utiles para features complejas o equipos grandes.

Contras: Integracion tardia, y mas conflictos al mergear.

Pros de estrategia de branching de branch de corta vida:
Integración continua y menos conflictos.

Feedback rapido de CI/CD y QA.
Codigo de main siempre cercano a produccion.
Aumenta la velocidad de entrega y despliegue.

Contras: Requiere alta disciplina y automatizacion (tests, pipelines).

3. ¿Cómo se debería estructurar el flujo de trabajo de branching para gestionar effectively hotfixes o parches críticos en producción de manera ágil y segura?

Mantener main siempre estable, crear ramas hotfix/* desde main, aplicar el fix, probar, merge a main y develop, y desplegar de inmediato.