Checklist de revisión de alta disponibilidad

Marca solo lo que puedas demostrar (implementado, probado y observable), no lo que "debería" estar.

01 Redundancia y topología

- [ ] Ningún componente crítico es un punto único de fallo (cómputo, red, datos, DNS, secretos).
- [ ] Réplicas repartidas en zonas de disponibilidad distintas; el sistema tolera perder una zona sin violar el SLO.
- [ ] El balanceo retira instancias no sanas automáticamente y no reenvía a un destino aún en recuperación.
- [ ] La ruta crítica no depende de un recurso compartido sin réplica (una base, una cola, un caché).

02 Health checks y failover

- [ ] Liveness y readiness diferenciados (reiniciar ≠ estar listo para recibir tráfico).
- [ ] El failover se probó con un corte real, no solo configurado; se mide el tiempo de detección + conmutación.
- [ ] Se evita el flapping (histéresis en umbrales; una recuperación parcial no reabre todo el tráfico de golpe).
- [ ] Una señal de salud degradada de forma parcial no arrastra el proceso entero a DOWN.

03 Capacidad y control de carga

- [ ] Hay headroom para el peor caso: perder una zona coincidiendo con un pico de tráfico.
- [ ] Pools, threads, conexiones y colas tienen límites explícitos y dimensionados.
- [ ] El autoscaling tiene límites superior e inferior; no puede amplificar una caída.
- [ ] El sistema se probó cerca de su límite de carga, no solo en condiciones normales.

04 Resiliencia de dependencias

- [ ] Cada llamada remota tiene timeout por intento y la operación un deadline end-to-end propagado.
- [ ] Los reintentos usan backoff con jitter y un presupuesto acotado (se conoce el peor caso de intentos totales).
- [ ] Los circuit breakers observan la señal correcta (tasa de fallos y de llamadas lentas) con muestra mínima.
- [ ] Existe aislamiento de recursos (bulkhead/pool dedicado) para que una dependencia lenta no agote al resto.
- [ ] Hay una degradación funcional definida para cuando una dependencia no responde.

05 SLO y presupuesto de error

- [ ] Hay SLO por operación crítica, no un único número global.
- [ ] El presupuesto de error está definido y se monitorea su consumo.
- [ ] Las alertas se basan en síntomas y SLO, no en métricas de recurso sin impacto (CPU sola, por ejemplo).
- [ ] Existe una política de qué se congela o prioriza cuando el presupuesto se agota.

06 Observabilidad

- [ ] Métricas RED por servicio (rate, errors, duration) y USE por recurso (utilización, saturación, errores).
- [ ] Trazas distribuidas con propagación de contexto; se puede seguir una petición entre servicios.
- [ ] Logs estructurados correlacionables por identificador de traza.
- [ ] Los dashboards responden preguntas de guardia (qué falla, dónde, desde cuándo), no solo muestran gráficas.
- [ ] La telemetría no desaparece durante la caída (retención y muestreo suficientes).

07 Respuesta a incidentes

- [ ] Los runbooks son ejecutables y están actualizados, no una wiki muerta.
- [ ] Hay rotación de guardia y un camino de escalamiento claro.
- [ ] Las mitigaciones son reversibles (feature flags, rollback, rate limit, reducción de tráfico).
- [ ] Cada incidente cierra con un postmortem sin culpa cuyas acciones correctivas realmente se ejecutan.

08 Datos, despliegue y validación continua

- [ ] Los backups se restauran periódicamente de verdad; RPO y RTO están definidos y medidos.
- [ ] Las escrituras críticas son idempotentes o tienen deduplicación (una respuesta perdida no duplica el efecto).
- [ ] Los despliegues son graduales (canary o azul-verde), con health-gate y rollback en un paso.
- [ ] Los cambios de configuración se tratan con el mismo cuidado que un despliegue de código.
- [ ] Se ejecutan simulacros de recuperación (game days / DR drills) y se verifica que las alertas realmente disparan.