

PROYECTO

Sistemas Distribuidos de Préstamo de Libros

Introducción a Sistemas Distribuidos



Arley Bernal Muñetón
Kevin Garay Gaitán
Juan Esteban Bello

Objetivos del Proyecto

El Sistema Distribuido de Préstamo de Libros de la Universidad Ada Lovelace gestiona préstamos, renovaciones y devoluciones en varias sedes mediante una arquitectura distribuida con alta disponibilidad, tolerancia a fallos y escalabilidad.

Utiliza comunicación síncrona y asíncrona implementada con ZeroMQ.

Contexto



- Gestionar préstamos, renovaciones y devoluciones en dos sedes físicas.
- Implementar replicación asíncrona de la base de datos para lograr consistencia eventual.
- Proveer mecanismos automáticos de tolerancia a fallos.
- Garantizar alto desempeño bajo distintas cargas de trabajo.
- Evaluar el impacto de diferentes estrategias de comunicación en el rendimiento.

Modelos del Sistemas

Modelo Arquitectónico

- Capa de Presentación: Procesos Solicitantes (PS) que interactúan con usuarios finales.
- Capa de Lógica de Negocio: Gestor de Carga (GC) y Actores (AS y AP) que coordinan las operaciones.
- Capa de Persistencia: Gestor de Almacenamiento (GA) con réplicas de base de datos.



Modelo de Interacción

Patrón Request-Replay (Síncrono):

- Operaciones de préstamo de libros (requieren consistencia inmediata).
- Comunicación PS → GC → Actor → GA → Respuesta.

Patrón Publicador/Suscriptor (Asíncrono):

- Operaciones de devolución y renovación (toleran consistencia eventual).
- GC publica tópicos, Actores suscritos procesan asíncronamente.

Modelos del Sistemas

Modelo de Fallos

Detección de Fallas:

- Health checks periódicos entre GC y GA mediante heartbeat.
- Timeouts configurados en comunicaciones síncronas (5 segundos).
- Monitoreo continuo del estado de los componentes.

Recuperación Automática:

- Failover automático a réplica secundaria cuando falla GA primario.
- Reconexión transparente para clientes sin interrupción del servicio.
- Resincronización de datos cuando el componente recupera operación.
- Cola de mensajes pendientes durante períodos de indisponibilidad.

Modelo de Seguridad

Control de Acceso:

- Validación de formatos de mensajes entre componentes.
- Verificación de integridad de datos en operaciones críticas.
- Autenticación básica mediante tokens en comunicaciones

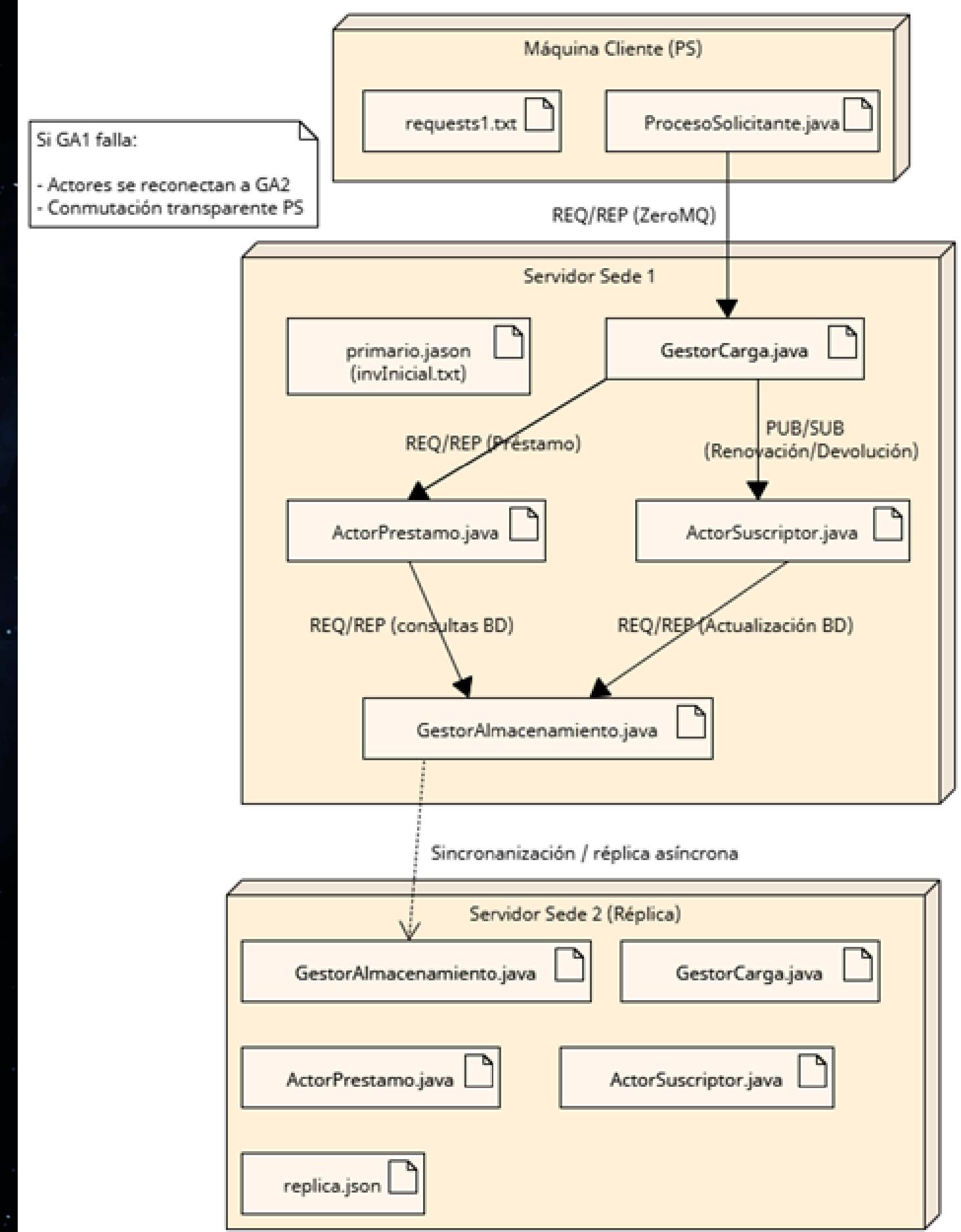
Protección de Datos:

- Validación de reglas de negocio (máximo 2 renovaciones por libro)
- Consistencia en actualizaciones de inventario
- Prevención de condiciones de carrera en operaciones concurrentes

Modelos del Sistemas

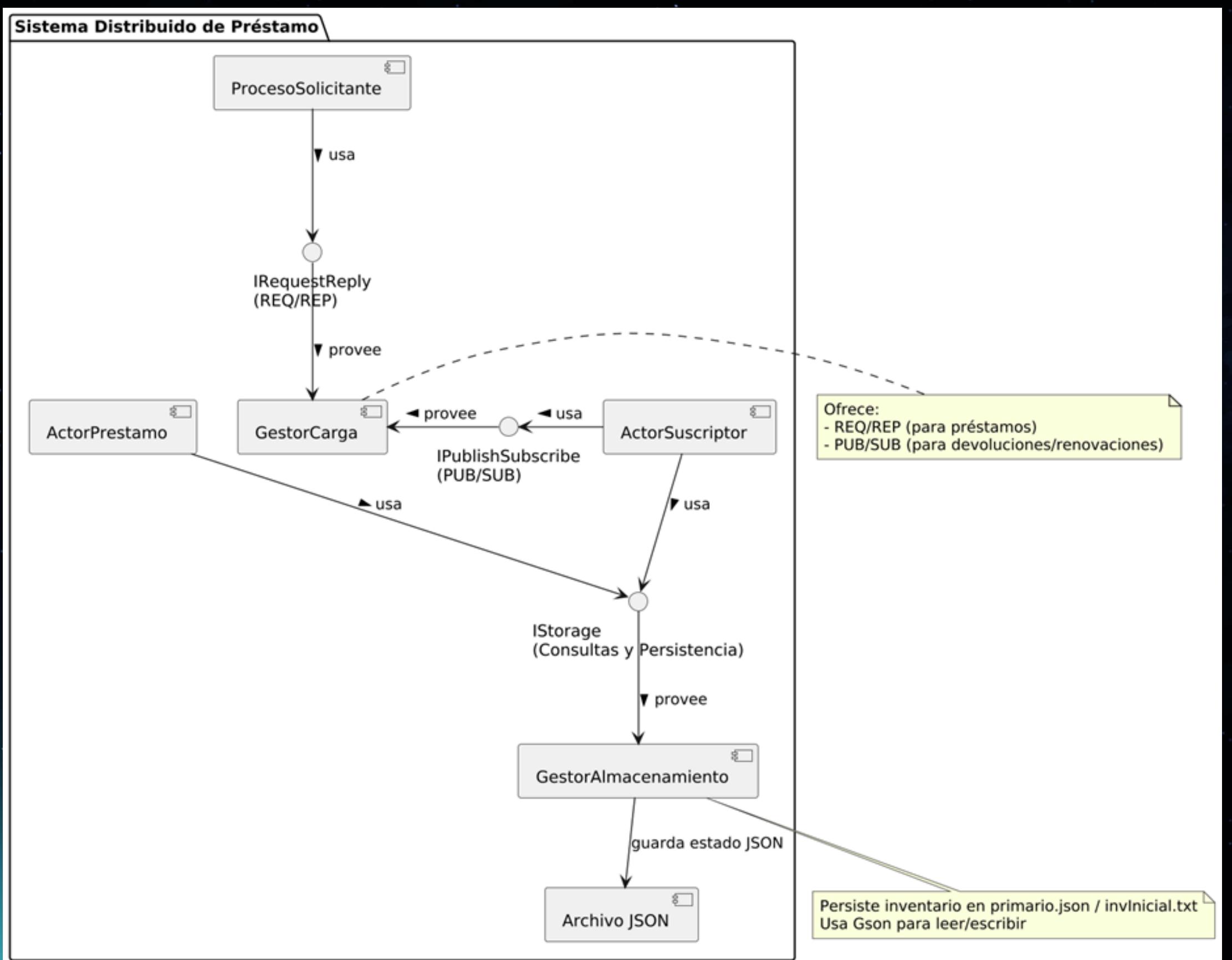
Diagrama de Despliegue

- Máquina Cliente (PS): Ejecuta ProcesoSolicitante.java con archivo requests1.txt
- Servidor Sede 1: Contiene GestorCarga.java, ActorPrestamo.java, ActorSuscriptor.java y GestorAlmacenamiento.java con primario.json
- Servidor Sede 2 (Réplica): Contiene componentes equivalentes con replica.json
- Mecanismo de Failover: Si GA1 falla, actores se recreenetan automáticamente a GA2.



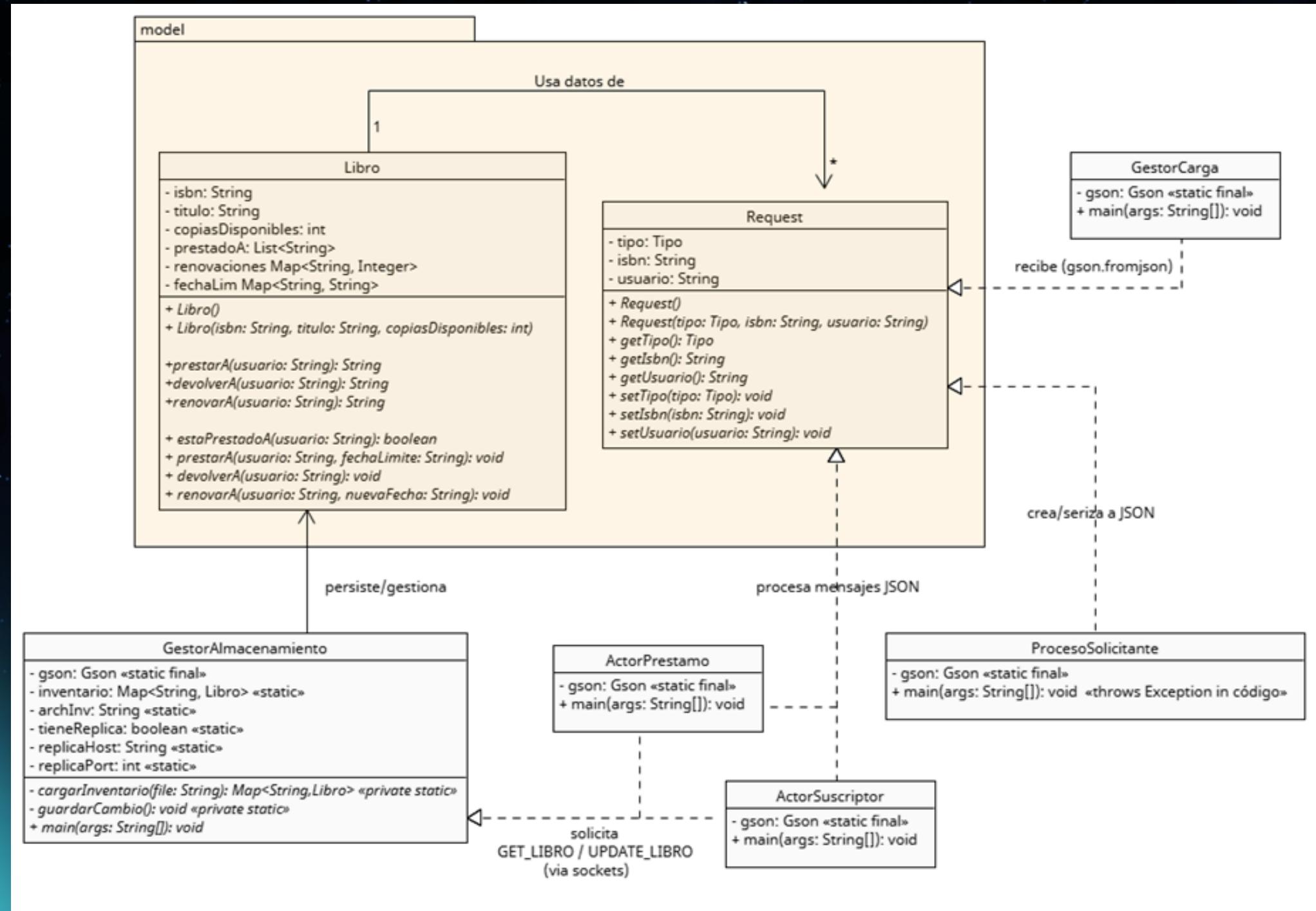
Modelos del Sistemas

Diagrama de Despliegue



Modelos del Sistemas

Diagrama de Clases



Descripción del modelo de datos:

- Clase Libro:
 - Atributos: isbn, titulo, copiasDisponibles, prestadoA, renovaciones, fechaLim.
 - Métodos: prestarA(), devolverA(), renovarA(), estaPrestadoA()
- Clase Request:
 - Atributos: tipo, isbn, usuario
 - Métodos: getters/setters para serialización JSON

Implementación de comunicaciones sincronas

Se implemento el caso B: hacer todas las operaciones del sistema manera sincrona, solo permitiendo nuevas solicitudes del PS una vez la solicitud actual haya finalizado.



Préstamo



Devolución



Renovación



Replica en BD
secundaria

```
// REQ socket para respuestas al GC
ZMQ.Socket gcReq = ctx.createSocket(SocketType.REQ);
gcReq.connect("tcp://" + gcRespHost + ":" + gcRespPort);

// Envío y recepción sincrónica
System.out.println("PS -> GC: " + json);
req.send(json);
String resp = req.recvStr(flags: 0); // 0 = sin espera
System.out.println("GC -> PS: " + resp);
```

Métricas de Desempeño

Métricas evaluadas en las pruebas del sistema con failover automático:

- **Tiempo de respuesta promedio (latencia):**

Mide cuánto tarda el sistema en procesar una solicitud.

- **Desviación estándar de la latencia:**

Evalúa la estabilidad de la respuesta bajo diferentes cargas.

- **Throughput total:**

Número de solicitudes procesadas correctamente en un intervalo de 2 minutos.

- **Promedio de solicitudes por proceso solicitante (PS):]**

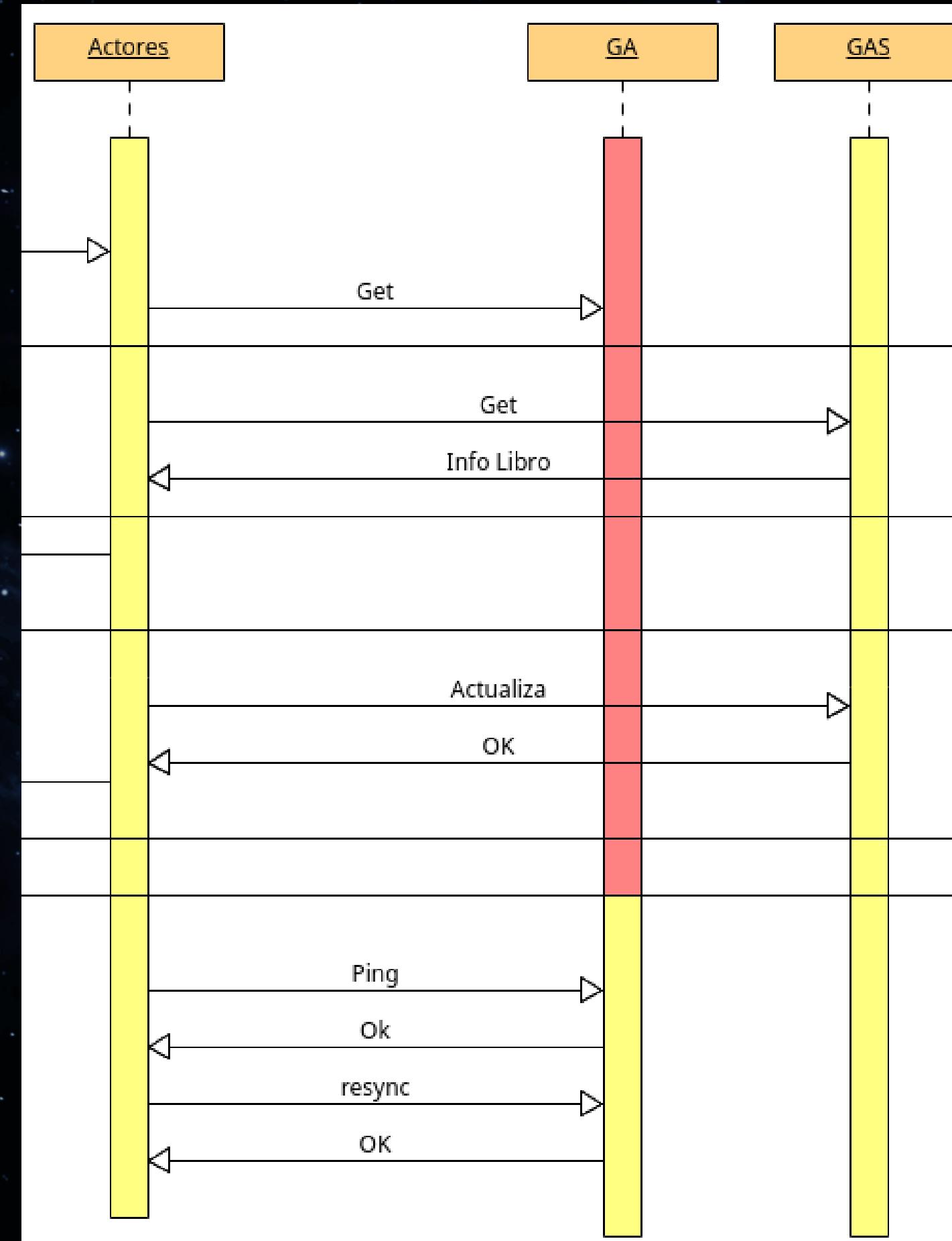
Refleja la distribución de carga por cliente.

Métricas de Desempeño

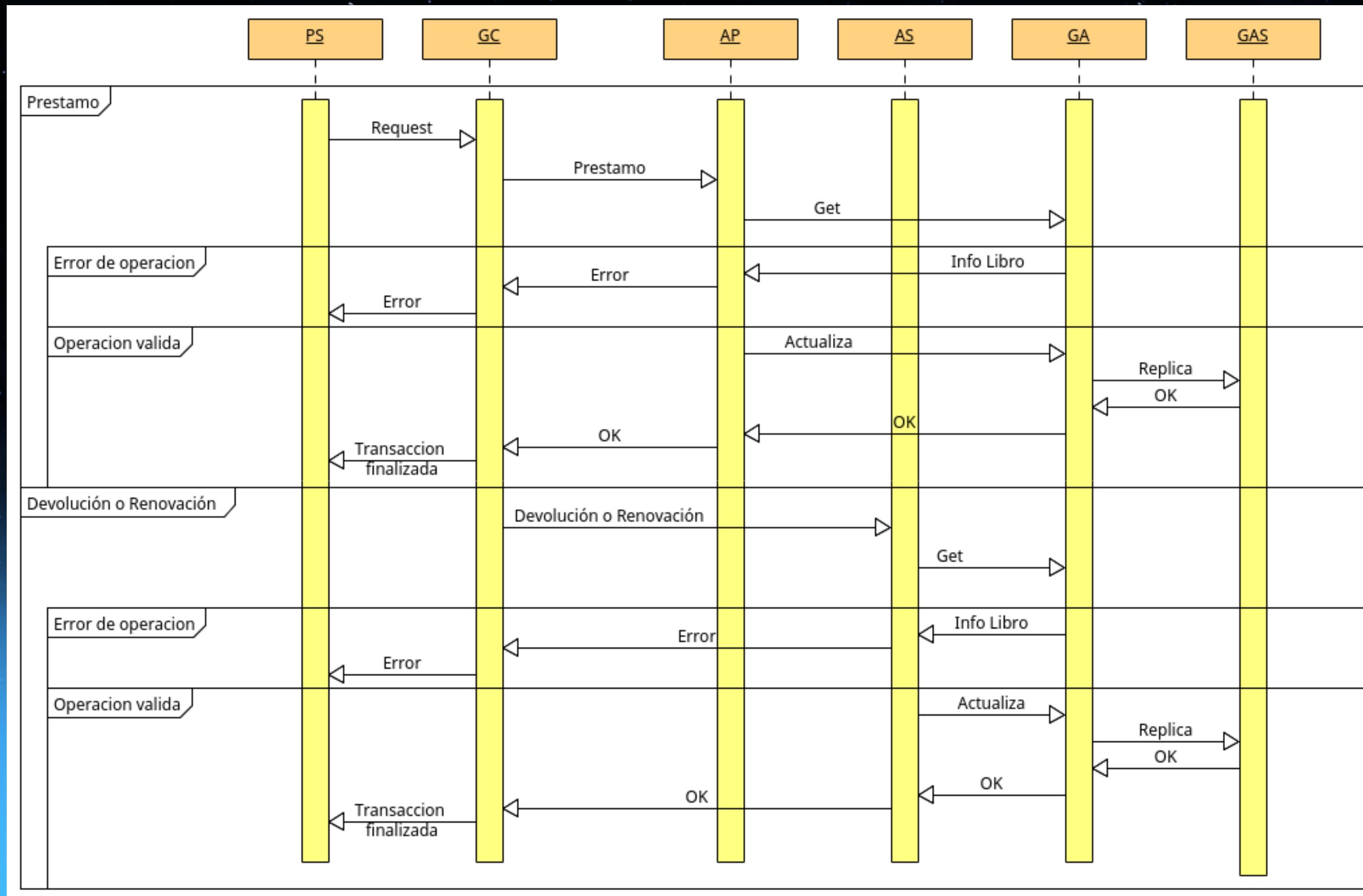
PS	Tiempo Promedio (ms)	Desv. Estandar (ms)	Solicitudes Totales	Promedio por PS
4	3.380	3.569	10,739	2,684.75
6	4.748	3.631	12,898	2,149.67
10	8.342	5.330	13,252	1,325.20

Fail-over a GAS

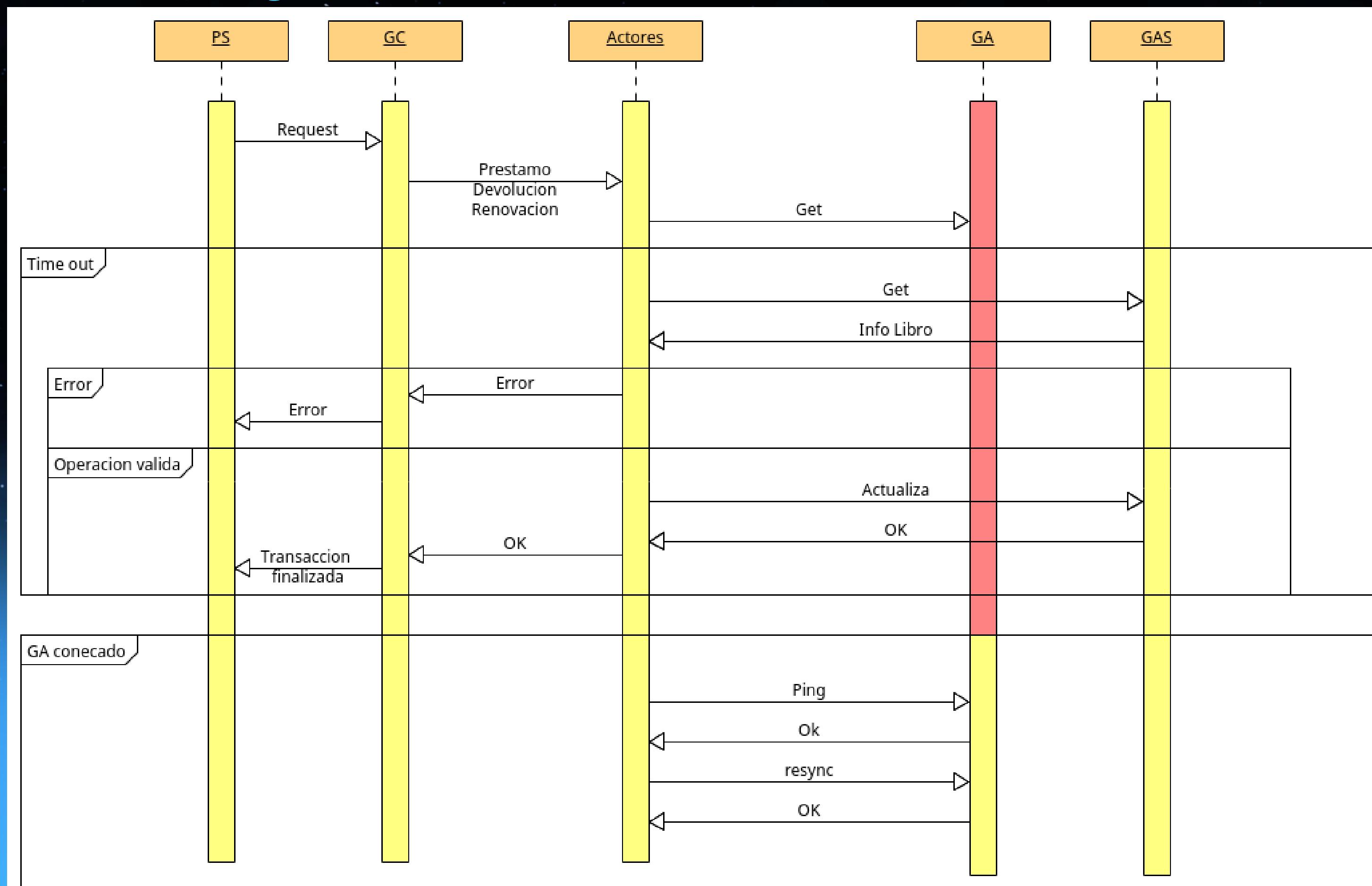
- 01** Cuando GA y GAS están activos las solicitudes son resueltas por GA y si deben hacerse cambios a la BD se replica en GAS
- 02** Cuando GA no responde, los actores direccionan las solicitudes automáticamente al GAS, si deben hacerse cambios a la BD se almacenan en una lista
- 03** Cuando GA vuelve a estar activo el GAS lo detecta y manda todas las actualizaciones pendientes para sincronizar el estado de ambas BD, los actores se vuelven a conectar automáticamente al GA



Flujo de datos caso normal



Flujo de datos caso fail-over



Repositorios

- [https://github.com/Juan-Bello/
Sis_Distribuidos/tree/main/Proyecto_SD](https://github.com/Juan-Bello/Sis_Distribuidos/tree/main/Proyecto_SD)

Muchas Gracias