Sistema Distribuido de Préstamo de libros

Gabriel Jaramillo Cuberos

Roberth Méndez Rivera

Mariana Osorio Vásquez

Juan Esteban Vera Garzón

Facultad de Ingeniería

Introducción a los Sistemas Distribuidos

Docente: John Jairo Corredor Franco

Octubre 2025

Resumen

El proyecto desarrolla un sistema distribuido para la gestión de préstamos, devoluciones y renovaciones de libros en una red de bibliotecas distribuidas en dos sedes. El sistema implementa una arquitectura basada en mensajería asincrónica con ZeroMQ, replicación de datos y tolerancia a fallos mediante almacenamiento redundante. Cada sede cuenta con un Gestor de Carga (GC), Actores especializados y un Gestor de Almacenamiento (GA) sincronizado. El diseño busca garantizar alta disponibilidad, balanceo de carga y consistencia eventual, lo que busca una baja latencia en las respuestas y la continuidad del servicio ante algún fallo.

Introducción

En los sistemas de información actuales, las empresas u organizaciones distribuyen sus servicios y recursos en diferentes partes para aumentar la disponibilidad, confiabilidad y eficiencia del procesamiento de los datos de sus clientes. En el contexto de una red de bibliotecas, es necesaria la coordinación entre diferentes nodos para mantener la coherencia del inventario al momento de gestionar préstamos, renovaciones y devoluciones. Su objetivo es ofrecer respuestas rápidas y sin fallos a los usuarios.
En este proyecto, se busca aplicar los conceptos fundamentales de comunicación entre procesos, concurrencia y tolerancia a fallos para construir un sistema distribuido de préstamo de libros que funcione en entornos reales de red, con múltiples sedes y clientes simultáneos. El trabajo integra tanto el diseño arquitectónico (componentes, comunicación, despliegue y fallos) como la implementación de un protocolo de pruebas que garantice la validez y desempeño del sistema bajo diferentes cargas.

1. Planteamiento del problema

En las bibliotecas modernas que operan en múltiples sedes o que cuentan con varios puntos de préstamo, uno de los mayores desafíos es mantener la coherencia, disponibilidad y confiabilidad de los datos en todos los nodos del sistema. Los procesos de préstamo, devolución y renovación de libros requieren coordinación constante para asegurar que el inventario esté actualizado en tiempo real y que los usuarios reciban información precisa sobre la disponibilidad de todo el material ofrecido por la biblioteca.

Sin embargo, en muchos casos, las bibliotecas siguen utilizando sistemas centralizados donde toda la información se almacena en un único servidor o base de datos. Esto hace que la gestión de los datos dependa de la conectividad y del rendimiento de ese servidor principal. Si ocurre algún fallo en la red, una caída del servidor o una interrupción eléctrica, el servicio completo puede verse afectado, dejando inactivas todas las sedes conectadas. Además, los tiempos de

respuesta se incrementan conforme crece la cantidad de usuarios concurrentes, lo que reduce la eficiencia general del sistema.

Además, el problema se escala cuando se necesita coordinar operaciones concurrentes sobre los mismos recursos. Por ejemplo, si dos usuarios en diferentes sedes solicitan el mismo libro simultáneamente, el sistema debe garantizar que solo uno de ellos obtenga el préstamo, evitando inconsistencias o duplicación de registros. Estas situaciones requieren mecanismos de sincronización y replicación confiables, así como un sistema capaz de tolerar fallos y mantener una consistencia sin afectar el rendimiento.

2.1 Problema central

Se busca resolver el siguiente problema:

Diseñar e implementar un sistema distribuido de préstamo de libros que garantice la disponibilidad del servicio, la coherencia de los datos entre sedes y un tiempo de respuesta eficiente ante solicitudes concurrentes.

Este problema combina aspectos de comunicación entre procesos, sincronización de datos, tolerancia a fallos y desempeño.

2.2 Requisitos específicos

A partir del análisis del problema, se establecen los siguientes requisitos que el sistema debe cumplir:

Requisitos funcionales

1. Permitir operaciones básicas de préstamo, devolución y renovación de libros desde cualquier sede.
2. Garantizar que las solicitudes concurrentes sean procesadas correctamente y sin duplicar registros.
3. Implementar un mecanismo de replicación asíncrona entre las bases de datos de las sedes para mantener la consistencia de la información.
4. Registrar cada operación en logs distribuidos que permitan rastrear la actividad del sistema.
5. Incorporar un mecanismo de recuperación automática en caso de fallos en alguno de los nodos.

Requisitos no funcionales

1. El sistema debe ser escalable y que sea capaz de soportar el incremento del número de procesos solicitantes sin afectar significativamente el rendimiento.
2. Debe garantizar tolerancia a fallos, permitiendo que la red de bibliotecas siga operando incluso si una sede se desconecta temporalmente.

3. La arquitectura debe ser modular, facilitando el mantenimiento y la extensión futura de funcionalidades.
4. Los tiempos de respuesta promedio deben mantenerse por debajo de 200 ms en escenarios de carga media.
5. La comunicación entre procesos debe ser segura y confiable, minimizando la pérdida o duplicación de mensajes.

2.3 Problemas técnicos

- Sincronización de datos entre nodos:
  Dado que cada sede tiene su propia base de datos, es necesario establecer un mecanismo de replicación que asegure la coherencia entre los registros. Este proceso debe ser eficiente, sin esperas que afecten el rendimiento.

- Manejo de concurrencia:
  Cuando múltiples procesos solicitantes (PS) realizan operaciones simultáneamente, pueden generarse conflictos sobre los mismos recursos. Por lo tanto, es necesario implementar un control de concurrencia que garantice la integridad de los datos sin frenar la ejecución.

- Tolerancia a fallos:
  El sistema debe ser capaz de detectar la caída de un nodo, redirigir las operaciones hacia otro disponible y sincronizar los datos una vez que el nodo fallido se recupere.

- Comunicación asincrónica eficiente:
  El uso de mensajería basada en ZeroMQ introduce ventajas de velocidad, pero también exige diseñar correctamente los canales PUB/SUB y REQ/REP para evitar pérdida de mensajes, duplicación o bloqueos por dependencias cíclicas.

- Balanceo de carga y desempeño:
  A medida que se incrementa el número de procesos solicitantes, el sistema debe distribuir equitativamente la carga de trabajo entre los actores y los gestores de carga, evitando cuellos de botella para tener una latencia relativamente baja.

2. Propuesta de solución

La solución planteada consiste en un sistema distribuido de préstamo de libros basado en una arquitectura modular y escalable que integra distintos componentes interconectados mediante la librería ZeroMQ, la cual permite implementar patrones de comunicación entre procesos.

Cada sede de la red cuenta con los siguientes componentes:

- Procesos Solicitantes (PS): generan y envían solicitudes de préstamo, devolución o renovación a través de archivos o scripts automatizados.
- Gestor de Carga (GC): recibe las solicitudes, valida su formato y las distribuye a los actores correspondientes mediante el patrón PUB/SUB.
- Actores especializados (Devolución, Renovación y Préstamo): suscritos a tópicos específicos, procesan las solicitudes de acuerdo con la lógica de negocio y se comunican con el Gestor de Almacenamiento (GA).
- Gestor de Almacenamiento (GA): mantiene la base de datos local de libros y usuarios, gestionando la persistencia y la replicación asíncrona con el GA de la otra sede.

El sistema emplea comunicación síncrona (REQ/REP) para operaciones que requieren confirmación inmediata (como el préstamo) y asíncrona (PUB/SUB) para las que pueden procesarse en segundo plano.

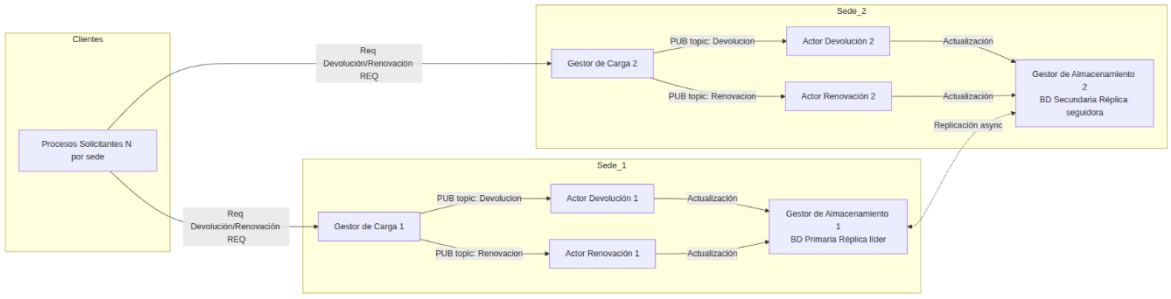
Para validar la solución, se diseña un plan de pruebas integral que incluye:

1. Pruebas funcionales: verifican el correcto comportamiento de cada componente y su comunicación.
2. Pruebas de replicación: aseguran la sincronización de datos entre bases distribuidas.
3. Pruebas de tolerancia a fallos: simulan caídas de nodos o interrupciones de red.
4. Pruebas de desempeño: miden tiempos de respuesta y tasa de rendimiento bajo distintas cargas de PS (4, 6 y 10 por sede).

3. Modelos del sistema

Modelos del sistema (Arquitectónico, interacción, fallos y seguridad). Cómo se aplican los conceptos de estos modelos al proyecto.

Arquitectónico



El diagrama arquitectónico muestra la estructura global del sistema distribuido de préstamo de libros y la relación entre sus principales componentes desplegados en dos sedes.

Cada sede cuenta con:

- Un Gestor de Carga (GC) que recibe las solicitudes de los clientes y las publica hacia los actores.
- Dos Actores especializados: uno para renovaciones y otro para devoluciones, que consumen los mensajes del GC mediante el patrón PUB/SUB.
- Un Gestor de Almacenamiento (GA) responsable de mantener la base de datos local y sincronizar los cambios con su réplica en la otra sede.

Los Procesos Solicitantes (PS), ubicados en la capa de clientes, pueden conectarse a cualquiera de los GC disponibles para enviar solicitudes de renovación o devolución.

La comunicación entre los GA de ambas sedes se realiza de forma asíncrona mediante replicación, garantizando consistencia eventual. Este diseño distribuye la carga de procesamiento y asegura tolerancia a fallos mediante redundancia de sedes.

Interacción

Los diagramas de interacción describen el flujo dinámico de mensajes entre los procesos distribuidos para las operaciones principales: devolución y renovación.

En ambos casos, la secuencia sigue el patrón asíncrono de confirmación inmediata al cliente y procesamiento en segundo plano:

- El Proceso Solicitante (PS) envía la solicitud al Gestor de Carga (GC).
- El GC responde con un estado 202 Accepted para liberar al PS rápidamente y luego publica el evento en el canal ZeroMQ correspondiente (topic: renovación o devolución).
- El Actor suscrito al tópico recibe el mensaje, ejecuta la lógica de negocio (verifica disponibilidad o número de renovaciones) y actualiza el estado del libro en el Gestor de Almacenamiento (GA).
- El GA confirma la operación (OK o error) y registra el cambio en el archivo de persistencia local.

Esta arquitectura basada en mensajería desacoplada permite alta concurrencia, resiliencia ante fallos y tiempos de respuesta bajos para el cliente.

Diagrama de devolución:

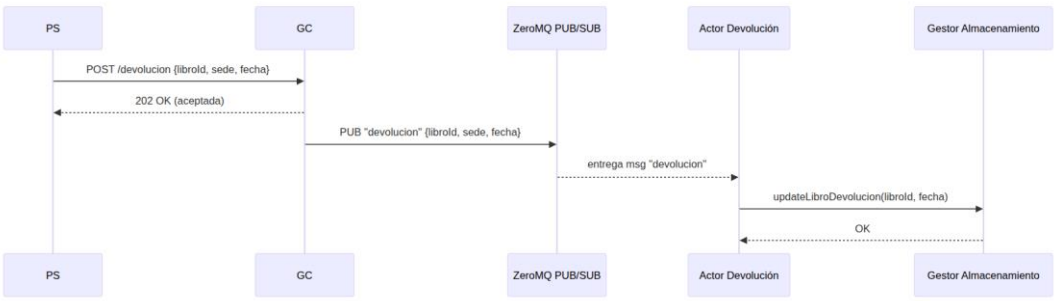
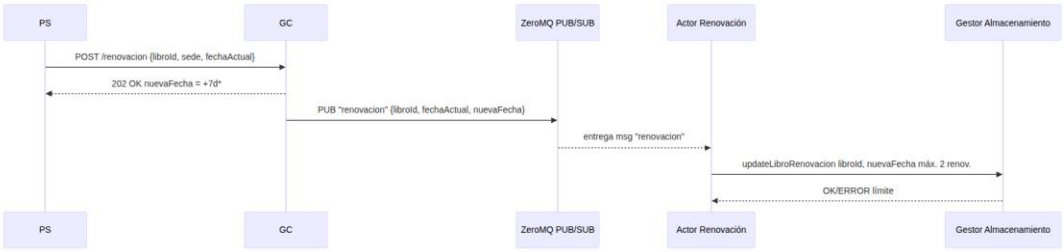
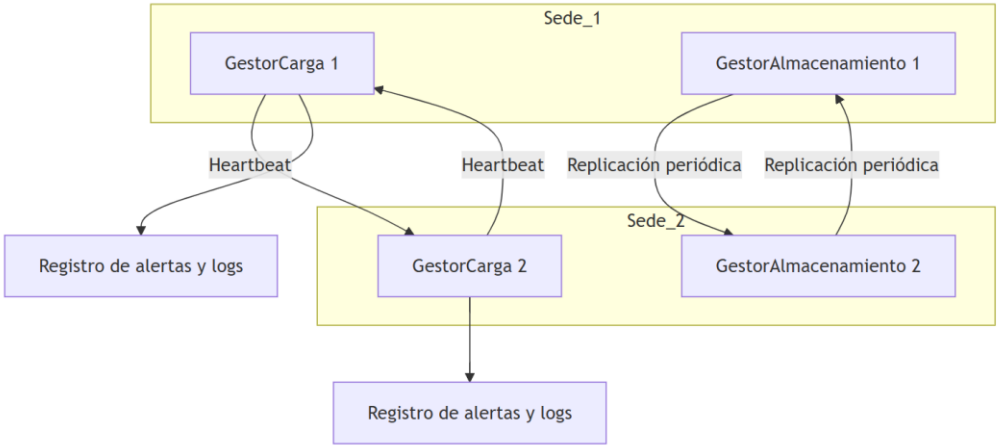


Diagrama de renovación:



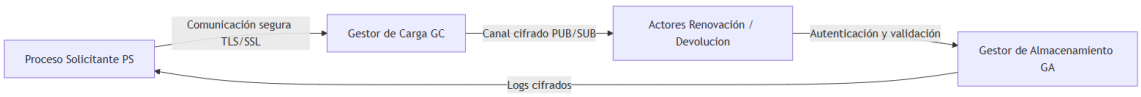
Fallos



Este diagrama de fallos muestra los mecanismos de tolerancia implementados:

- Los Gestores de Carga (GC1 y GC2) intercambian heartbeats periódicos para detectar caídas de nodo.
- Los Gestores de Almacenamiento (GA1 y GA2) sincronizan su estado por replicación periódica asíncrona.
- Si uno de los GA falla, el otro mantiene los datos hasta restablecer la conexión.
- Cada GC registra sus eventos de error en un módulo de logs y alertas locales, que luego puede revisarse para diagnóstico.

Seguridad



El Diagrama de Seguridad muestra las capas de protección aplicadas a la comunicación y persistencia del sistema distribuido:

- Cifrado de extremo a extremo (TLS/SSL): protege el intercambio de mensajes entre los procesos Solicitantes (PS) y el Gestor de Carga (GC).

- Cifrado en el canal PUB/SUB: evita la interceptación de mensajes entre GC y los Actores.
- Autenticación local de Actores y GA: cada Actor valida sus credenciales antes de actualizar datos en el Gestor de Almacenamiento.
- Logs cifrados: las transacciones y fallos se registran localmente con cifrado simétrico, garantizando integridad y confidencialidad.
- Control de acceso por IP: cada nodo acepta conexiones solo desde direcciones IP confiables configuradas en el entorno.

4. Diseño del sistema

Diagrama de despliegue

El diagrama de despliegue representa la distribución física de los componentes del sistema sobre diferentes máquinas de la red.

- Máquina A (Sede 1): ejecuta el GC1, el Actor de Renovación 1, el Actor de Devolución 1 y el GA1 (que contiene la base de datos primaria o réplica líder).
- Máquina B (Sede 2): ejecuta el GC2, el Actor de Renovación 2, el Actor de Devolución 2 y el GA2 (réplica seguidora).
- Máquina C (Clientes): aloja varios Procesos Solicitantes (PS) que generan carga de solicitudes hacia las sedes.

La comunicación entre PS y GC utiliza el patrón REQ/REP, mientras que la comunicación entre GC y Actores usa PUB/SUB.

Las GA de ambas sedes intercambian actualizaciones mediante replicación periódica y pueden continuar funcionando en modo degradado si una sede falla.

Este despliegue garantiza disponibilidad, balanceo de carga y redundancia geográfica, cumpliendo los principios básicos de los sistemas distribuidos.

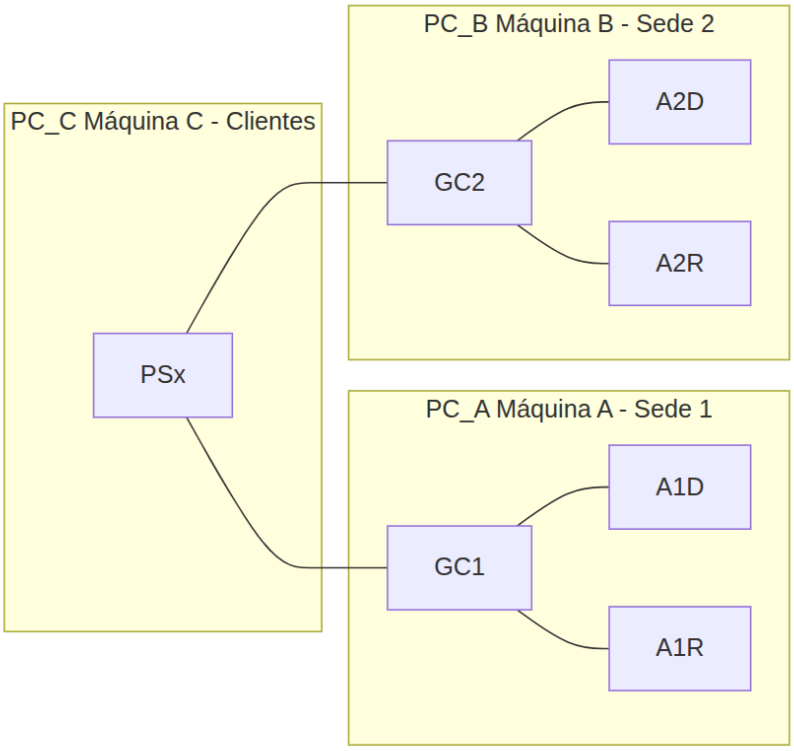


Diagrama de componentes

Representa los módulos físicos de software desplegados en cada máquina. Cada sede replica la misma estructura lógica (GC + Actores + GA).

La sincronización entre GA1 y GA2 se realiza de manera asíncrona, garantizando consistencia eventual.

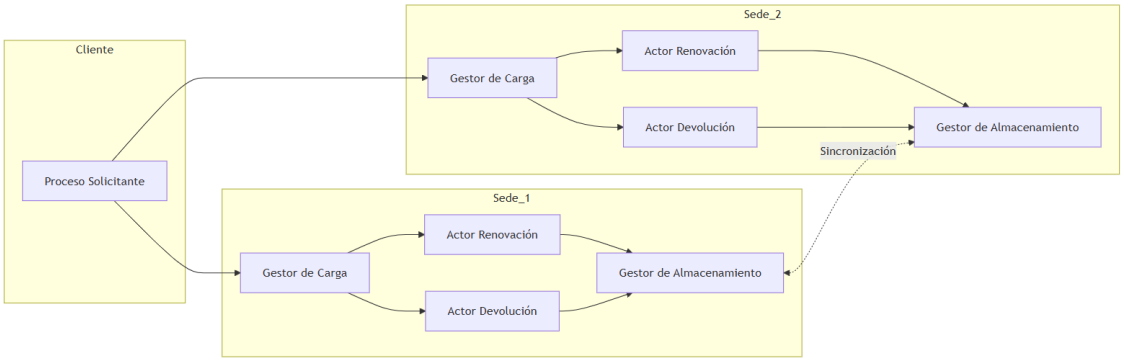


Diagrama de clases

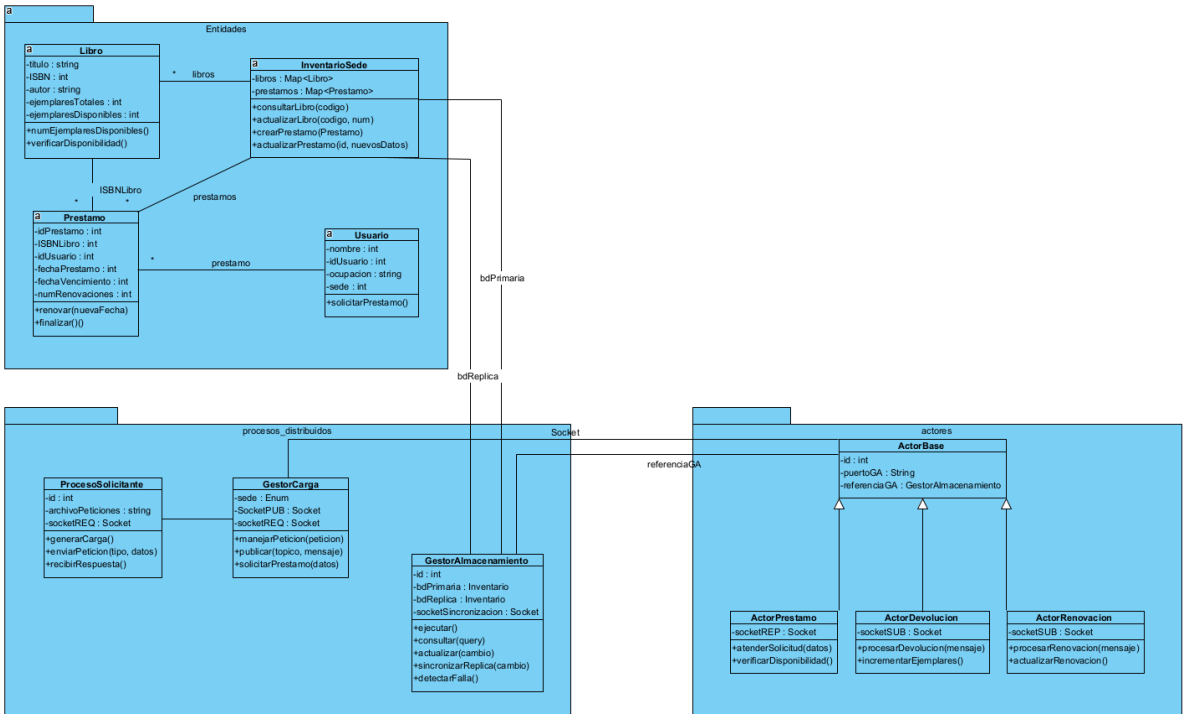
El Diagrama de Clases modela la estructura lógica del software, separando las responsabilidades de persistencia, comunicación distribuida y lógica de negocio en tres paquetes principales.

1. Paquete: entidades

Este paquete define las Entidades del Negocio, que encapsulan la información que será persistida y manejada por el GestorAlmacenamiento (GA).

2. Paquete: procesos_distribuidos
   Este paquete define los componentes principales que se ejecutan como procesos independientes. Contiene la lógica central de enrutamiento, generación de carga y persistencia.
3. Paquete: actores

   Este paquete implementa la lógica de negocio específica para cada tipo de solicitud. La lógica se centraliza a través del GestorAlmacenamiento (GA).



El diseño muestra una clara separación de capas mediante el siguiente flujo de dependencias:

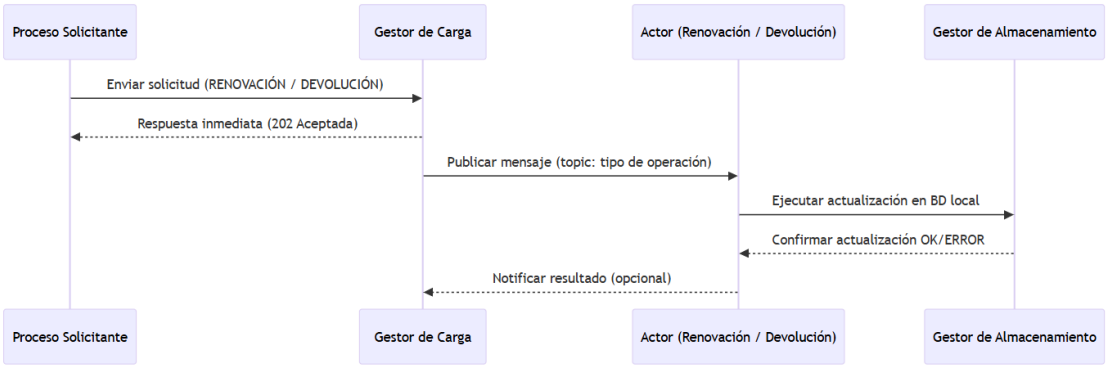
PS → GC: El PS depende del GC para enviar peticiones.

GC → Actores: El GC depende de los Actores para procesar las operaciones de negocio (síncronas o asíncronas).

Actores → GA: Los Actores dependen del GA para todas las operaciones de persistencia.

GA → Inventario: El GA depende de la clase InventarioSede para el manejo de los datos de la BD.

Diagrama de secuencia



El Diagrama de Secuencia representa el flujo completo de interacción entre los componentes del sistema distribuido durante la ejecución de una operación (ya sea renovación o devolución).

- PS (Proceso Solicitante) envía la solicitud de operación al GC (Gestor de Carga) usando el patrón REQ/REP de ZeroMQ.
- GC procesa la solicitud, responde inmediatamente al PS con un código de aceptación (202 Accepted) para no bloquear el cliente, y publica el evento en el canal correspondiente mediante PUB/SUB.
- El Actor suscrito al tópico (según sea devolución o renovación) recibe el mensaje, interpreta la operación y llama al Gestor de Almacenamiento (GA).
- GA actualiza la base de datos local (CSV o memoria) verificando las reglas de negocio (por ejemplo, máximo 2 renovaciones).
- Una vez finalizada la actualización, el GA confirma la operación al Actor, y este puede opcionalmente notificar al GC el resultado final.

Este diseño garantiza baja latencia percibida por el cliente, procesamiento asíncrono y consistencia eventual entre las réplicas de almacenamiento.

5. Protocolo de pruebas

El protocolo de pruebas que utilizará para la entrega final (considere todos los tipos de prueba que deben realizarse a un sistema), haciendo énfasis en las pruebas de desempeño.

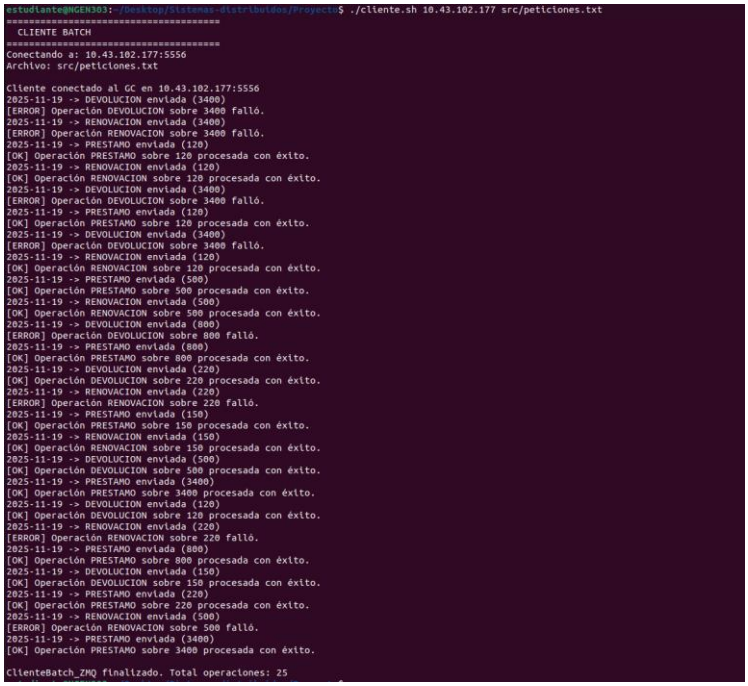
ID	Componente	Descripción	Entrada	Resultado esperado	Tipo de prueba
PS-01	Procesos Solicitantes	Validar el envío correcto de peticiones desde un archivo local hacia el gestor de carga.	Archivo ps_sede1.txt con ≥20 solicitudes.	Cada solicitud es procesada; el PS muestra confirmación de envío y recibe ACK del GC.	Funcional

PS-02	Procesos Solicitantes	Validar manejo de errores cuando el archivo no existe.	Archivo inexistente o nombre incorrecto.	El PS informa “Error: archivo no encontrado” y no envía solicitudes.	Negativa
DEV-01	Actor Devolución	Probar procesamiento correcto de una devolución válida.	DEVO LB101 (libro actualmente prestado).	GC responde OK, Actor Devolución actualiza BD primaria y programa actualización en réplica.	Funcional
DEV-02	Actor Devolución	Probar devolución de libro no prestado.	DEVO LB202 (libro ya disponible).	Actor registra intento; GC responde “sin cambios”; no genera error crítico.	Negativa
REN-01	Actor Renovación	Verificar que un préstamo puede renovarse una vez.	RENO LB301 con préstamo vigente.	Se actualiza fecha de vencimiento; Actor registra modificación en BD.	Funcional
REN-02	Actor Renovación	Verificar rechazo en tercera renovación.	RENO LB301 ya renovado dos veces.	GC responde “Renovación no permitida”; sin modificación de BD.	Negativa
PRE-01	Actor Préstamo	Validar préstamo exitoso cuando hay ejemplares disponibles.	PRES LB401 con stock ≥1.	GC responde “Préstamo aprobado”; BD reduce ejemplares en 1.	Funcional
PRE-02	Actor Préstamo	Validar respuesta ante falta de ejemplares.	PRES LB999 con stock = 0.	GC responde “Sin ejemplares disponibles”; operación no ejecutada.	Negativa
PRE-03	Actor Préstamo	Simular colisión entre devolución y préstamo simultáneos.	PRES LB777 mientras se procesa DEVO LB777.	Resultado variable (depende del retardo asíncrono); se documenta en logs.	Concurrencia
PER-01	Gestor de Almacenamiento	Verificar igualdad inicial entre BD primaria y réplica.	BD inicial cargada en ambas sedes.	Dump de datos idéntico; sincronización inicial correcta.	Integración
PER-02	Gestor de Almacenamiento	Validar replicación asíncrona de cambios.	Serie de devoluciones	Cambios inmediatos en primaria; réplica	Integración

			y renovaciones.	actualiza tras breve retardo.	
FAIL-01	Tolerancia a fallas	Probar caída del GA/BD primaria durante la operación.	Interrupción manual del proceso GA.	Sistema conmuta a réplica; operaciones continúan sin interrupción; reconexión automática al volver.	Resiliencia
FAIL-02	Tolerancia a fallas	Probar falla de un Actor de Devolución.	Detener Actor mientras GC publica.	Otro Actor suscrito (si existe) procesa el mensaje; no hay pérdida de datos.	Resiliencia
DES-01	Desempeño	Medir tiempo de respuesta promedio (Préstamo) con 4 PS por sede.	4 PS simultáneos, cada uno con 20 solicitudes.	Promedio ≤ 200 ms; sin pérdidas.	Rendimiento
DES-02	Desempeño	Medir tasa de rendimiento total con 10 PS por sede.	10 PS simultáneos (200 solicitudes).	Se mantiene tasa de rendimiento > 90% del caso base; escalabilidad aceptable.	Rendimiento
DES-03	Desempeño	Comparar GC serial vs multihilos.	Escenario base repetido con ambas configuraciones.	Versión multihilo reduce tiempo de respuesta	Comparativa
DES-04	Desempeño	Comparar asincronía completa (Devolución/Renovación/Préstamo) vs síncrona.	Activar/desactivar asincronía.	Disminución notable del tiempo promedio de respuesta	Comparativa

Algunas de estas pruebas se realizan manual, para evidenciar en pantalla su ejecución:

PS-01



Las fallas le aparecen a la sede:

```
==> /tmp/tmp.16ZMNG2UpY <==
Actor: operación DEVOLUCION -> ÉXITO
Actor reportó: RESULT|d83eb6d3-ad43-42e1-91e2-e3a48b77d4f3|SUCCESS|DEVOLUCION

==> /tmp/tmp.YCJswowS51 <==
GC registró resultado DEVOLUCION [d83eb6d3-ad43-42e1-91e2-e3a48b77d4f3]: SUCCESS
GC recibió solicitud: PRESTAMO|220|user23
GC publicó PRESTAMO: PRESTAMO|d5119807-9e2b-4f42-b74f-91e8d54d138a|220|user23|20
25-11-19|null
GC respondió préstamo: en proceso

==> /tmp/tmp.z0DTGHesmf <==
2025-11-19T17:10:41.564250008 - ActorPrestamo recibió: PRESTAMO|d5119807-9e2b-4f
42-b74f-91e8d54d138a|220|user23|2025-11-19|null

==> /tmp/tmp.8UPeFgcBTS <==
GA recibió: PRESTAMO|220|user23
GA respondió: OK|Préstamo otorgado

==> /tmp/tmp.z0DTGHesmf <==
[OK] ActorPrestamo: PRÉSTAMO OTORGADO para libro 220

==> /tmp/tmp.YCJswowS51 <==
GC recibió solicitud: RENOVACION|500|user24

==> /tmp/tmp.8UPeFgcBTS <==
GA recibió: VALIDAR_PRESTAMO|500|system
GA respondió: OK|false

==> /tmp/tmp.YCJswowS51 <==
GC rechazó RENOVACION: libro no prestado
GC recibió solicitud: PRESTAMO|3400|user25
GC publicó PRESTAMO: PRESTAMO|e8c1e8a5-efe3-435c-9ea7-0c13c6a53906|3400|user25|2
025-11-19|null
GC respondió préstamo: en proceso

==> /tmp/tmp.z0DTGHesmf <==
2025-11-19T17:10:41.973874780 - ActorPrestamo recibió: PRESTAMO|e8c1e8a5-efe3-43
5c-9ea7-0c13c6a53906|3400|user25|2025-11-19|null

==> /tmp/tmp.8UPeFgcBTS <==
GA recibió: PRESTAMO|3400|user25
GA respondió: FAILED|No disponible
```

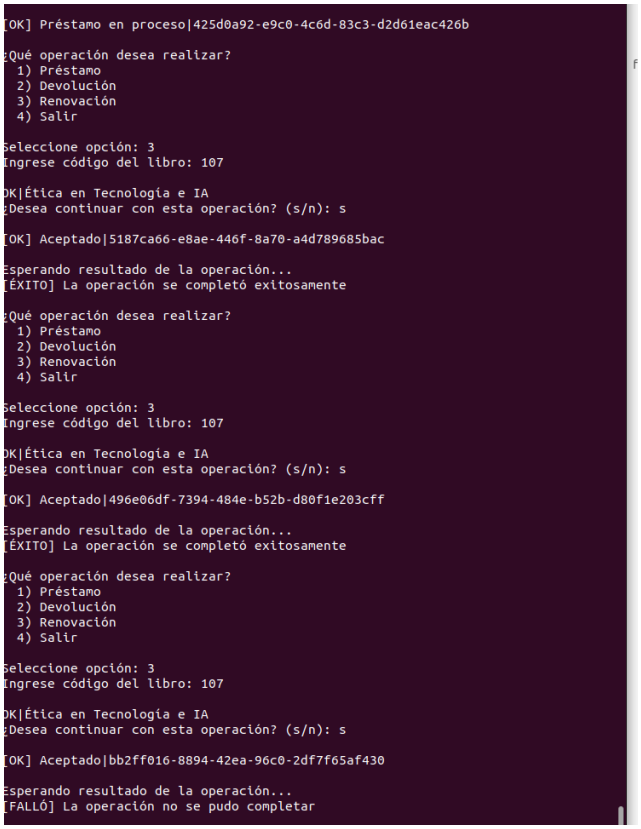
PS-02

```
estudiante@NGEN363:~/Desktop/Sistemas-distribuidos/Proyecto$ ./cliente.sh 10.43.
103.177 src/peticionesNoExiste.txt
========================================
  CLIENTE BATCH
========================================
Conectando a: 10.43.103.177:5556
Archivo: src/peticionesNoExiste.txt

[ERROR] Archivo 'src/peticionesNoExiste.txt' no encontrado
estudiante@NGEN363:~/Desktop/Sistemas-distribuidos/Proyecto$
```

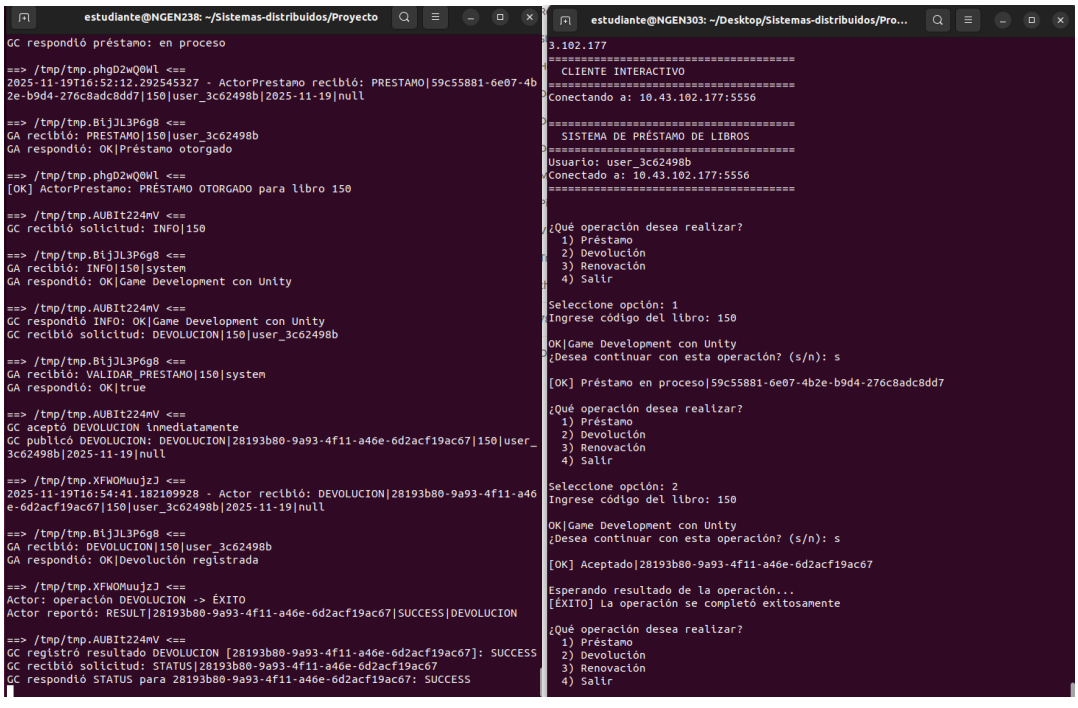
REN-01

```
./cliente.sh 10.43.102.177
========================================
  CLIENTE INTERACTIVO
========================================
Conectando a: 10.43.102.177:5556

========================================
  SISTEMA DE PRÉSTAMO DE LIBROS
========================================
Usuario: user_53e3b66f
Conectado a: 10.43.102.177:5556
========================================


¿Qué operación desea realizar?
  1) Préstamo
  2) Devolución
  3) Renovación
  4) Salir

Seleccione opción: 1
Ingrese código del libro: 107

OK|Ética en Tecnología e IA
¿Desea continuar con esta operación? (s/n): s

[OK] Préstamo en proceso|425d0a92-e9c0-4c6d-83c3-d2d61eac426b

¿Qué operación desea realizar?
  1) Préstamo
  2) Devolución
  3) Renovación
  4) Salir

Seleccione opción: 3
Ingrese código del libro: 107

OK|Ética en Tecnología e IA
¿Desea continuar con esta operación? (s/n): s

[OK] Aceptado|5187ca66-e8ae-446f-8a70-a4d789685bac

Esperando resultado de la operación...
[ÉXITO] La operación se completó exitosamente
```

REN-02



DEV-01



6. Métricas de desempeño, experimentos y resultados

Para la evaluación del desempeño del sistema distribuido de préstamo de libros, las métricas se obtendrán generando archivos con la extensión (.dat). Estos archivos guardaran los tiempos

de respuesta y ejecución de las solicitudes. Para esto, se probará el rendimiento del programa usando de 1 a 4 hilos para dividir las tareas en subtareas. El objetivo de esto es realizar 30 repeticiones por cada hilo en cada solicitud con el objetivo de obtener varias gráficas que permitan visualizar los tiempos de respuesta y su variación con respecto a la cantidad de hilos (o trabajadores) usados.

Durante la ejecución, el programa recopilará automáticamente los tiempos de respuesta, calculando el promedio en milisegundos.

Finalmente, los resultados se analizarán usando gráficas de dispersión que permitan ver la evolución del rendimiento del sistema a lo largo de la prueba. A partir de estas mediciones se podrá determinar el comportamiento del sistema bajo distintas condiciones de carga.

6.1. Especificaciones de Hardware y Software

Para garantizar la replicación de los experimentos de desempeño, todas las pruebas se ejecutaron en un entorno homogéneo. Se usaron las máquinas virtuales asignadas por la universidad. Cada máquina cuenta con el mismo hardware. Las características de los sistemas de cómputo usados son las siguientes:

Característica	VM1	VM2	VM3	VM4
L1d / L1i	192 KiB / 128 KiB	192 KiB / 128 KiB	192 KiB / 128 KiB	192 KiB / 128 KiB
L2	5 MiB	5 MiB	5 MiB	5 MiB
L3	42 MiB	42 MiB	42 MiB	42 MiB
Memoria total	12 GB	12 GB	12 GB	12 GB
Núcleos	4	4	4	4
Hilos	4	4	4	4
Procesador	Intel® Xeon® Gold 6348	Intel® Xeon® Gold 6348	Intel® Xeon® Gold 6348	Intel® Xeon® Gold 6348
Velocidad	2.6 GHz	2.6 GHz	2.6 GHz	2.6 GHz

Con respecto al software, el sistema distribuido se ejecutó en lo siguiente:

- Ubuntu 22.04 LTS: sistema operativo base de las máquinas.

- OpenJDK 17: usado para compilar y ejecutar los procesos del sistema distribuido.
- ZeroMQ: middleware empleado para la comunicación para implementar los patrones REQ/REP y PUB/SUB.
- Excel: visualización de gráficas comparativas.

Juntando todo lo mencionado anteriormente se logró realizar pruebas controladas eliminando discrepancias o variaciones a causa de hardware o software heterogéneo.

6.2. Herramientas de medición usadas

Como se mencionó previamente, para registrar y visualizar el comportamiento del sistema friente a las pruebas definidas en el plan de pruebas, se desarrolló un módulo de medición que geneta archivos .dat por cada conjunto de experimentos. En estos archivos se almacenan los tiempos de respuesta individual por solicitud.

El sistema fue probado de 1 a 4 hilos. Por cada hilo se repitió el experimento 30 veces por la distribución normal. Esto se hizo con el objetivo de tener valores estables y comparables, con promedios lo mas cercanos a los valores exactos necesarios para una comparación libre de imperfectos. Con estos datos se realizaron gráficas comparativas de desempeño, las cuales se encuentran en el archivo Excel adjunto al repositorio. Es importante tener en cuenta que, para mejorar la precisión de estos experimentos, es necesario realizarlos en sistemas de cómputo que cuenten con más hilos de procesamiento.

6.3. Resultados obtenidos

Para ver los resultados obtenidos, ver tabla de resultados en el repositorio. Estas tablas buscan visualizar lo siguiente:

- Tiempo de respuesta promedio vs. Número de hilos
- Variación del rendimiento con cargas crecientes.
- Comparación entre Gestor de Carga secuencial y multihilos.
- Comparación entre procesamiento sincrónico y asincrónico.

7. Análisis de resultados

7.1. Comportamiento del sistema con el aumento de hilos

Los resultados obtenidos muestran que el tiempo de respuesta promedio del sistema aumenta conforme se incrementa el número de hilos usados para procesar las solicitudes. En un sistema ideal se esperaría que más hilos conduzcan a más paralelismo y, por lo tanto, se reduzcan los

tiempos de ejecución. Sin embargo, en nuestros experimentos ocurre lo contrario, pues el rendimiento empeora progresivamente al pasar de 1 a 2, 3 y 4 hilos.

Este comportamiento indica que el sistema no escala de manera eficiente con el incremento del paralelismo interno. Esto es consistente con arquitecturas donde se comparten recursos, los cuales se ven saturados generan cuellos de botella cuando varios hilos compiten por ellos simultáneamente (condiciones de carrera).

En todos los experimentos se mantuvo este patrón, lo que demuestra que no es una variación puntual sino de un límite de la misma estructura del diseño del proyecto.

7.2. Razones por las cuales puede empeorar el tiempo:

Realizando un análisis del sistema distribuido y su arquitectura, se identificaron las siguientes razones por las cuales se puede explicar el aumento del tiempo de respuesta:

7.2.1. Contención por recursos compartidos

El gestor de almacenamiento es uno de los recursos mas accedidos ya que todos los hilos dependen de él para leer el estado de los libros, actualizar solicitudes o sincronizar cambios. Existe un gestor de carga por sede, por lo que si varios hilos intentan acceder a él de forma simultánea, se pueden generar bloqueos mutuos, condiciones de espera, retrasos acumulados y secciones críticas mas extensas. El archivo de persistencia es un recurso centralizado y no puede ser accedido por múltiples procesos, razón por la cual se convierte en un cuello de botella.

7.2.2. Sobrecarga por cambio de contexto

En cursos de semestres anteriores se investigó el “Context Switching”. Cada vez que se agrega un hilo trabajador al sistema, el sistema operativo de la máquina debe gestionar mas ejeciciones, y el organizador o planificador del sistema operativo debe generar mas colas para garantizar el acceso de todos los hilos a los recursos. Esto nuevamente coincide con la idea de que, por el hecho de haber un solo gestor de carga, no existe un sistema en paralelo que permita el acceso de varios hilos al recurso de forma simultánea.

7.2.3. Limitaciones por los entornos virtuales

Las pruebas no se realizaron en sistemas de cómputo físicos. Se realizaron en máquinas virtuales homogéneas en software y hardware. Estas máquinas tienen recursos compartidos, por lo que el acceso a recursos de una máquina compite con el acceso a recursos de las demás máquinas provenientes del mismo sitio. Además, es importante que para acceder a estas máquinas, se debe establecer una conexión con una VPN para el acceso remoto. El sistema

inevitablemente está condicionado por la latencia en las redes locales y las de la universidad e incluso congestiones en la VPN si se está usando en varias zonas.

7.2.4. Sincronización de ZeroMQ y retrasos en sockets

ZeroMQ es un sistema eficiente. Sin embargo, si aumenta la cantidad de hilos trabajadores que emiten mensajes, puede aumentar la presión sobre los sockets del sistema. Esto hace que se generen colas internas más grades, además de aumentar la espera en los patrones REQ/REP y PUB/SUB.

7.2.5. Acceso serializado a la base de datos de la biblioteca

La base de datos del sistema no es una base de datos transaccional. Es un archivo plano que contiene toda la información. Esto genera posibles bloqueos y dificultad en paralelizar operaciones de persistencia ya que el acceso a la base de datos es un acceso secuencial.

7.3. Análisis general del sistema

Los resultados de los experimentos muestran con evidencia que el sistema funciona correctamente. Es tolerante a cargas altas, mantiene su integridad y en general evita fallos a gran escala que comprometan el funcionamiento del sistema. Sin embargo, no existe una mejora en el rendimiento al aumentar la cantidad de hilos a causa de las limitaciones mencionadas anteriormente.

Conclusiones

1. La implementación del sistema demuestra que los principios de los sistemas distribuidos permiten construir soluciones robustas, escalables y tolerantes a fallos, aplicables a escenarios o situaciones reales.

2. El uso de ZeroMQ como middleware de comunicación facilita la integración de patrones síncronos y asíncronos, lo que permite construir una plataforma ligera y eficiente para el intercambio de mensajes entre procesos distribuidos.

3. El proyecto logra integrar de forma práctica los siguientes conceptos teóricos: comunicación entre procesos, tolerancia a fallos, replicación, asincronía y escalabilidad.

4. El proyecto está diseñado de tal forma que se pueda continuar desarrollando con el objetivo de implementar nuevas funcionalidades en el futuro.

5. Incrementar hilos no siempre aumenta el rendimiento en sistemas distribuidos, sobre todo cuando hay acceso a recursos compartidos o la arquitectura del sistema no esta paralelizada.