

UNIVERSIDAD CENTRAL DEL ECUADOR
FACULTAD DE INGENIERIA EN CIENCIAS APLICADAS
INGENIERIA INFORMÁTICA



PROGRAMACIÓN DISTRIBUIDA
EJEMPLO DE UNA APLICACIÓN DISTRIBUIDA (TOLERANCIA A FALLOS)

TRABAJO PRÁCTICO

INTEGRANTES:

CHISIQUEQUI LUIS

CUASCOTA EDWIN

DOCENTE:

DIEGO PINTO

2023-2023

Contenido

PREFACIO.	3
1. INTRODUCCIÓN.	4
2. OBJETIVOS.	4
2.1 General.	4
2.2 Específicos.	4
3. Arquitectura.	4
4. Descripción de la Arquitectura.	5
4.1 Cliente.	5
4.2 Puerto 8080.	5
4.3 Gateway.	5
4.4 Instancias.	5
4.5 Base de datos.	5
5. Modelo entidad relación.	6
6. Módulos.	6
6.1 App-Autores.	6
6.2 App-Libros	6
7. Desarrollo de la App en Intellig.	6
7.1 Servicios implementados	7
7.2 Dependencias	7
7.3 Tolerancia a Fallos	8
8. Ejecución.	8
9. Conclusiones.	9
10. Bibliografía.	9

PREFACIO.

TÍTULO: Ejemplo de sistema distribuido de manejo de libros con tolerancia a fallos.

RESUMEN: El proyecto de implementación de un sistema distribuido de administración de libros por autores busca reducir las distintas fallas que puede ocurrir en un servidor web de consulta de libros, mediante el uso de la herramienta de IntelliJ IDEA.

AUTORES: Chisiquinga Luis
Cuascota Edwin

Responsabilidad del

desarrollo y ejecución: Estudiantes de la Facultad de Ingeniería y Ciencias Aplicadas de la carrera de Ingeniería Informática.

Responsabilidad de Ing. Diego Pinto

la revisión:

Distribución: A los estudiantes de la Universidad Central del Ecuador

1. INTRODUCCIÓN.

La creación de aplicaciones distribuidas se ha vuelto esencial para ofrecer servicios eficientes y confiables. Una de las áreas que ha experimentado un crecimiento exponencial es la consulta de libros a través de la web. La posibilidad de acceder a una vasta biblioteca digital desde cualquier lugar y en cualquier momento ha transformado la forma en que las personas buscan y obtienen información.

No obstante, con la creciente demanda de servicios en línea, surge la necesidad de desarrollar aplicaciones que sean resilientes y capaces de funcionar sin interrupciones incluso en condiciones adversas. La tolerancia a fallos se ha convertido en un factor crucial en el diseño de estas aplicaciones distribuidas, permitiendo que sigan operativas incluso cuando se presentan problemas técnicos o interrupciones inesperadas.

2. OBJETIVOS.

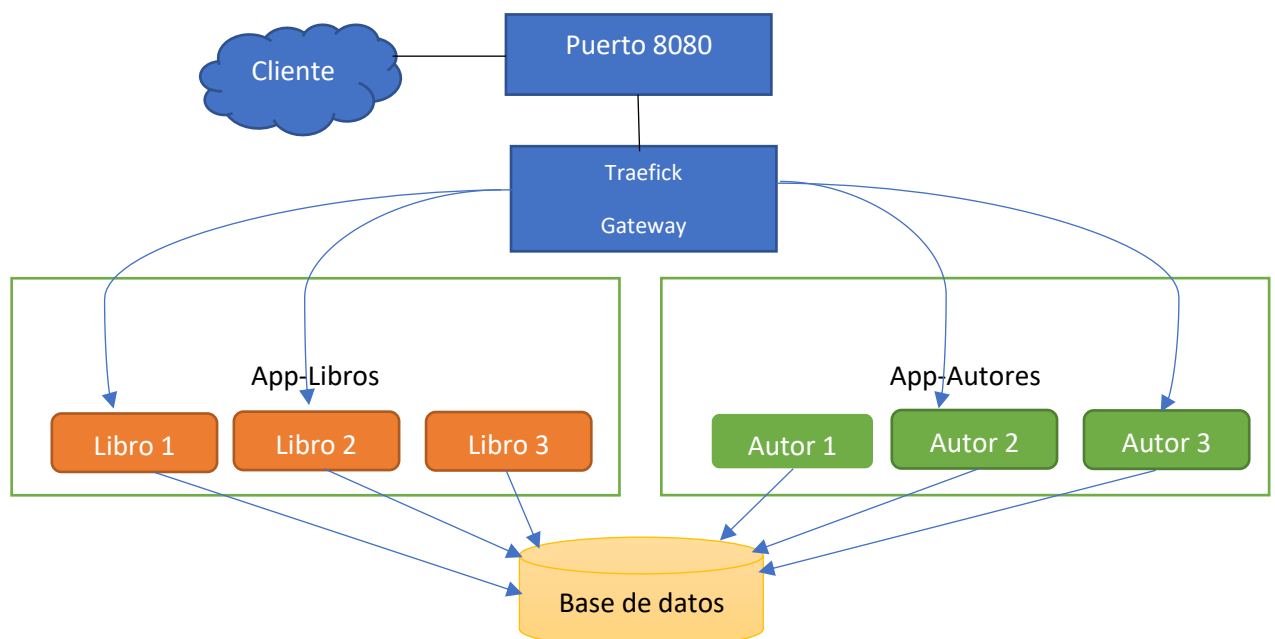
2.1 General.

Este proyecto tiene como objetivo la creación de una aplicación distribuida de consulta de libros a través de la web, enfocándose en la implementación de mecanismos de tolerancia a fallos. Esto implica diseñar un sistema que sea capaz de mantener su funcionalidad esencial incluso en situaciones donde componentes individuales puedan fallar. Para lograr esto, se explorarán tecnologías como la redundancia de servidores, la replicación de datos y la implementación de técnicas de recuperación automática.

2.2 Específicos.

- Diseñar una aplicación que pueda resistir la caída de un servidor y siga funcionando sin alterar los demás servicios o funciones.
- Utilizar distintas herramientas open source que ayude a la construcción del sistema distribuido.
- Desarrollar una arquitectura capaz de gestionar la aplicación y que incluyan la tolerancia a fallos y la escalabilidad de la misma.

3. Arquitectura.



4. Descripción de la Arquitectura.

4.1 **Cliente.** Se refiere a una entidad o componente que solicita y consume los servicios ofrecidos por otra entidad conocida como "servidor". En este modelo, la aplicación se divide en dos partes principales: el cliente y el servidor.

El cliente es la interfaz a través de la cual los usuarios interactúan con la aplicación. Puede ser una aplicación de software, una página web, una aplicación móvil u otro tipo de interfaz de usuario. El cliente envía solicitudes al servidor para obtener información, realizar operaciones o realizar funciones específicas. Por ejemplo, en una aplicación de consulta de libros en línea, el cliente podría ser la interfaz web o la aplicación móvil que los usuarios utilizan para buscar y acceder a libros.

4.2 **Puerto 8080.** se refiere a un número específico que se utiliza para identificar un punto de comunicación en un dispositivo o sistema. En términos más simples, un puerto es como una puerta virtual a través de la cual las aplicaciones pueden enviar y recibir datos en una red.

Cuando una aplicación en una red distribuida necesita establecer una conexión con otra aplicación en otro dispositivo, debe especificar tanto la dirección IP del destino como el número de puerto para dirigir la comunicación al servicio adecuado en ese dispositivo. Por ejemplo, si estás creando una aplicación web, tu servidor podría escuchar en el puerto 80 para las solicitudes HTTP.

4.3 **Gateway.** Un gateway en una aplicación distribuida actúa como un punto de entrada que facilita la comunicación entre diferentes componentes o sistemas, garantizando que los datos se transmitan de manera efectiva entre diferentes entornos y protocolos. Su función es esencial para lograr una interconexión eficiente y segura en sistemas complejos y heterogéneos.

4.4 **Instancias.** Se refiere a una ejecución individual y aislada de un componente o servicio dentro del sistema distribuido. Cada instancia representa una copia independiente de una aplicación, módulo o servicio que se ejecuta en uno de los nodos o dispositivos de la red distribuida. Estas instancias pueden ser idénticas o similares en su configuración y funcionalidad, y trabajan juntas para lograr los objetivos de la aplicación en conjunto.

Las instancias son una forma de aprovechar la escalabilidad horizontal, que es una característica esencial de las aplicaciones distribuidas. En lugar de tener una sola instancia monolítica que maneje todas las solicitudes y procesos, el sistema distribuido se compone de múltiples instancias que trabajan en paralelo para distribuir la carga y mejorar el rendimiento.

4.5 **Base de datos.** se refiere a una colección organizada y estructurada de información o datos que se almacenan en un formato digital. Las bases de datos desempeñan un papel fundamental en muchas aplicaciones distribuidas al almacenar y administrar datos de manera eficiente, permitiendo que múltiples componentes o instancias de la aplicación accedan y compartan información de manera coherente y segura. Las bases de datos juegan un papel crítico en la gestión de información, desde perfiles de usuarios y registros de actividad hasta contenido multimedia y datos de configuración. La elección de la arquitectura de base de datos depende de los

requisitos específicos de la aplicación, incluyendo la escalabilidad, la consistencia de datos, la latencia y la redundancia.

5. Modelo entidad relación.

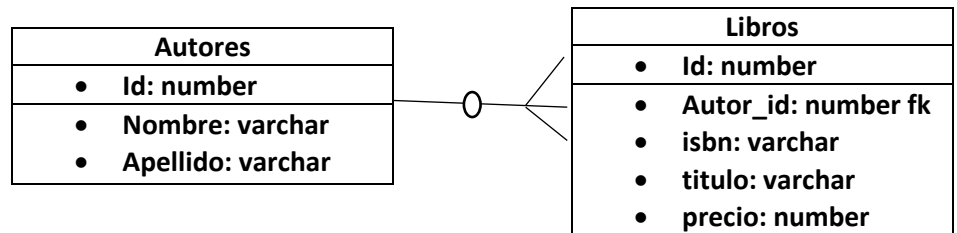


Figura 2: Modelo entidad relación de consumo de servicio autores.

6. Módulos.

6.1 App-Autores. Este módulo permite realizar operaciones CRUD sobre la tabla Books. La tecnología a utilizaren este módulo corresponde a:

Helidon 3.1

Acceso a la base de datos a través de DBClient.

6.2 App-Libros Este módulo permite realizar operaciones CRUD sobre la tabla Authors . La tecnología a utilizar en este módulo corresponde a: Qurkus 2.16

Acceso a la base de datos a través de Panache-Quarkus Data Repository

7. Desarrollo de la App en Intellig.

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) muy popular para la creación y desarrollo de aplicaciones distribuidas, entre otros tipos de aplicaciones. Hay varias razones por las cuales muchas personas eligen IntelliJ IDEA para desarrollar aplicaciones distribuidas:

Potente IDE Java: IntelliJ IDEA es conocido por ser uno de los IDE más potentes y avanzados para el desarrollo en lenguaje Java. Proporciona numerosas herramientas y funciones que facilitan la codificación, depuración y pruebas de aplicaciones Java, incluidas las aplicaciones distribuidas.

Soporte para múltiples tecnologías: Las aplicaciones distribuidas suelen involucrar una combinación de tecnologías y protocolos, como REST, SOAP, WebSocket, gRPC, etc. IntelliJ IDEA ofrece un amplio soporte para estas tecnologías, lo que facilita el desarrollo de diferentes tipos de aplicaciones distribuidas.

Integración de herramientas y complementos: IntelliJ IDEA ofrece una amplia gama de complementos y extensiones que permiten integrar herramientas externas, como sistemas de control de versiones, herramientas de construcción (como Maven y Gradle), sistemas de gestión de bases de datos y más. Esto ayuda a automatizar tareas y mejorar la eficiencia en el desarrollo de aplicaciones distribuidas.

Depuración avanzada: La depuración es esencial en el desarrollo de aplicaciones distribuidas para identificar problemas en diferentes componentes. IntelliJ IDEA proporciona un depurador avanzado que permite rastrear problemas en tiempo real, inspeccionar variables, establecer puntos de interrupción y más, lo que facilita la resolución de problemas en aplicaciones distribuidas.

Integración con frameworks y tecnologías populares: Muchos frameworks y tecnologías populares en el mundo de las aplicaciones distribuidas, como Spring Framework, Java EE y tecnologías front-end, están bien integrados en IntelliJ IDEA. Esto facilita la configuración y el uso de estas tecnologías en tu proyecto.



7.1 Servicios implementados

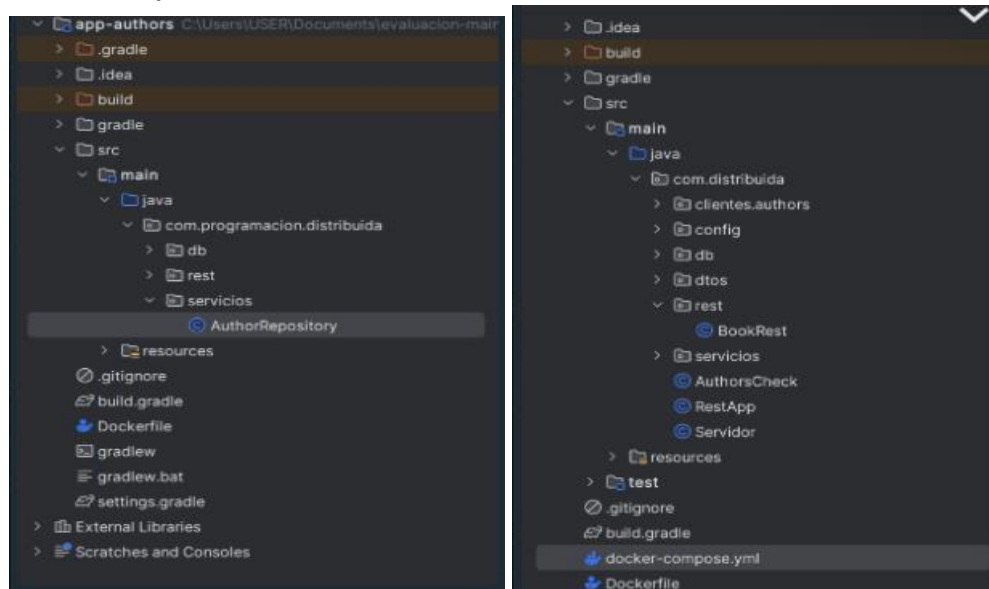


Figura 3 Implementación de los servicios autores y libros en IntelliJ Idea.

7.2 Dependencias

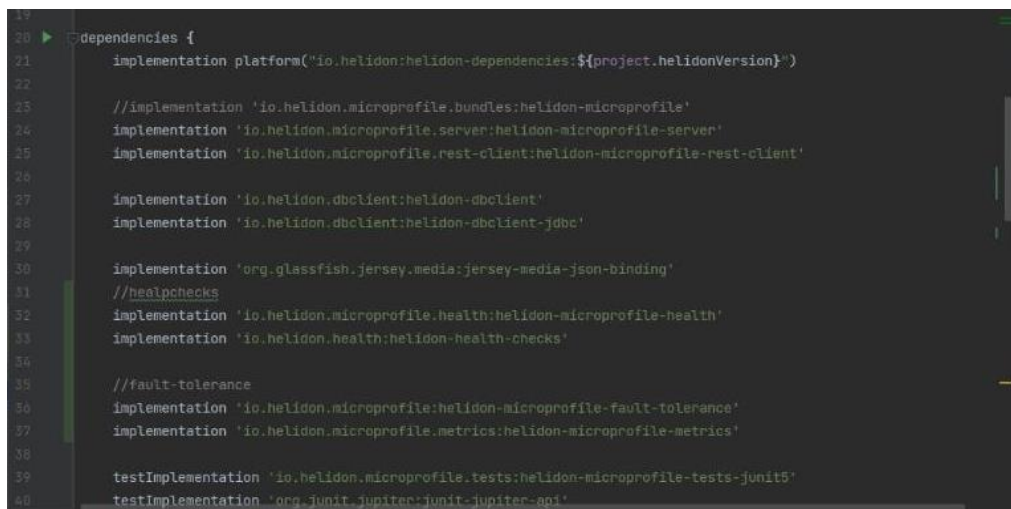


Figura 4: Dependencias de la App.

las dependencias en programación distribuida se refieren a las relaciones y requisitos entre diferentes componentes, servicios, datos y recursos que forman parte de un sistema distribuido, y son esenciales para el funcionamiento fluido y coherente del sistema en su conjunto, en la imagen anterior se muestra las dependencias necesarias para que nuestra app funcione normalmente.

7.3 Tolerancia a Fallos

```
@GET
@Path("/all")
//Método de tolerancia a fallos
@Fallback(fallbackMethod = "findAll")
public List<BookDto> findAllCompleto() throws Exception {
    var books = bookService.findAll();
}
```

Figura 5: Método implementado para la tolerancia a fallos.

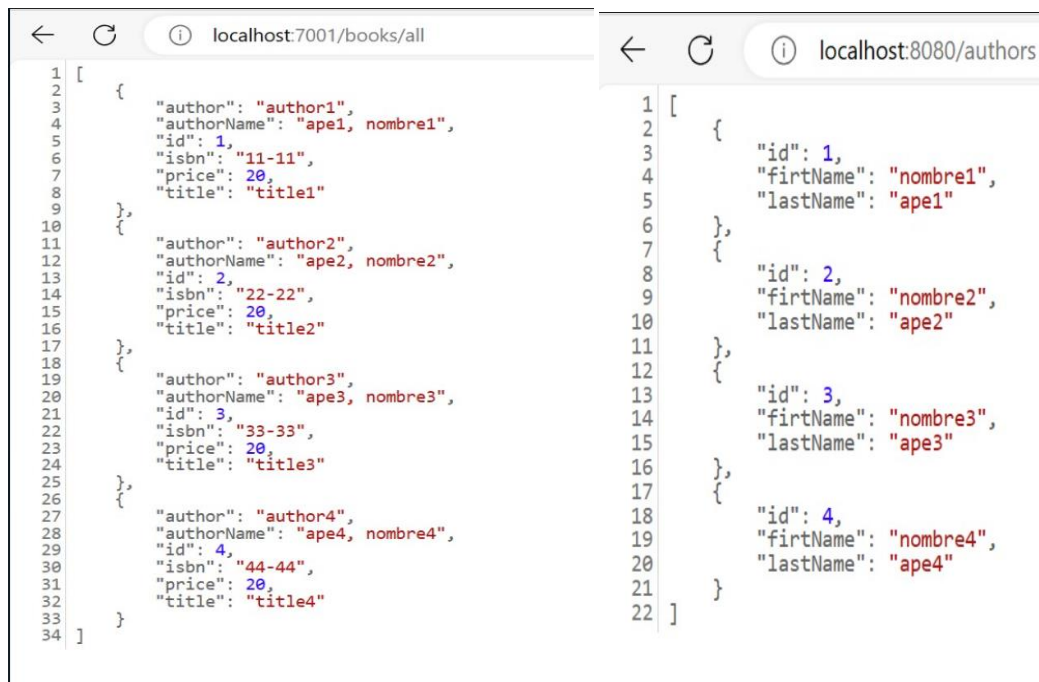
Aquí hay una descripción general de cómo funciona el método de tolerancia a fallos en una aplicación distribuida:

Detección de fallos: El primer paso es identificar que ha ocurrido una falla. Esto puede implicar la monitorización constante del estado de los componentes, la verificación de la disponibilidad de servicios, el seguimiento de los tiempos de respuesta y otras métricas relevantes. La detección temprana de fallos es esencial para iniciar las acciones correctivas lo antes posible.

Isolación de fallos: Una vez que se detecta una falla, es importante aislar su impacto para evitar que se propague a otras partes del sistema. Esto podría implicar la desconexión de un nodo defectuoso, redireccionar el tráfico lejos de componentes afectados o realizar cambios en la configuración para minimizar el impacto.

Reconfiguración dinámica: En muchos casos, el sistema debe ser capaz de reconfigurarse automáticamente para adaptarse a las fallas. Esto podría incluir la reasignación de tareas a nodos sanos, la redistribución de la carga de trabajo o incluso la reorganización de la topología de red.

8. Ejecución.

The image displays two side-by-side browser windows. The left window, with the address bar showing 'localhost:7001/books/all', contains a JSON array of four book objects. Each object has fields for 'author', 'authorName', 'id', 'isbn', 'price', and 'title'. The right window, with the address bar showing 'localhost:8080/authors', contains a JSON array of four author objects. Each object has fields for 'id', 'firstName', and 'lastName'. The JSON data is color-coded: strings are in red, numbers in blue, and punctuation in black.

```
1 [
2   {
3     "author": "author1",
4     "authorName": "ape1, nombre1",
5     "id": 1,
6     "isbn": "11-11",
7     "price": 20,
8     "title": "title1"
9   },
10  {
11    "author": "author2",
12    "authorName": "ape2, nombre2",
13    "id": 2,
14    "isbn": "22-22",
15    "price": 20,
16    "title": "title2"
17  },
18  {
19    "author": "author3",
20    "authorName": "ape3, nombre3",
21    "id": 3,
22    "isbn": "33-33",
23    "price": 20,
24    "title": "title3"
25  },
26  {
27    "author": "author4",
28    "authorName": "ape4, nombre4",
29    "id": 4,
30    "isbn": "44-44",
31    "price": 20,
32    "title": "title4"
33  }
34 ]
```

```
1 [
2   {
3     "id": 1,
4     "firstName": "nombre1",
5     "lastName": "ape1"
6   },
7   {
8     "id": 2,
9     "firstName": "nombre2",
10    "lastName": "ape2"
11  },
12  {
13    "id": 3,
14    "firstName": "nombre3",
15    "lastName": "ape3"
16  },
17  {
18    "id": 4,
19    "firstName": "nombre4",
20    "lastName": "ape4"
21  }
22 ]
```

Figura 6: Corrida de las instancias de los módulos libros y autores.

Como se puede observar tanto la App de libros y autores se ejecuta correctamente, si existiera el caso de una falla en el servidor de autores la aplicación de libros debería ejecutarse normalmente, aunque le falte los campos de nombre del autor ya que es preferible a que falle un campo y no toda la aplicación.

9. Conclusiones.

- Los sistemas distribuidos son inherentemente más propensos a las fallas debido a la complejidad de la interacción entre múltiples componentes y nodos. La detección y mitigación de fallos deben ser consideradas desde el diseño inicial.
- Las fallas en sistemas distribuidos pueden tener un impacto significativo en la experiencia del usuario. La tolerancia a fallos se centra en minimizar el impacto de estas fallas, manteniendo el sistema en funcionamiento y proporcionando un servicio aceptable incluso en situaciones de fallo.
- Implementar la tolerancia a fallos agrega una capa de complejidad al diseño y desarrollo de sistemas distribuidos. Se requiere un equilibrio entre la introducción de redundancia y la sobrecarga que esto puede generar en términos de recursos y administración.
- La tolerancia a fallos en programación distribuida busca minimizar los efectos negativos de las fallas y mantener la continuidad del servicio. A medida que los sistemas distribuidos continúan siendo esenciales en una variedad de aplicaciones, la comprensión de las estrategias de tolerancia a fallos se vuelve cada vez más crucial para los ingenieros y arquitectos de software.

10. Bibliografía.

Viñuales, S. A. (1988). Tolerancia a fallos en sistemas distribuidos mediante replicación de procesos (Doctoral dissertation, Universidad Politécnica de Madrid).

Vila, J. (2017). Replicación en sistemas distribuidos. DISCA/UPV Departamento de Informática de Sistemas y Computadoras Universidad Politécnica de Valencia.

Gallardo Gómez, A. (2013). Descubrimiento de servicios tolerante a fallos basado en hipercubos para sistemas distribuidos de gran escala.