

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



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

**MANUAL TÉCNICO**

**INTEGRANTES:**

CHISQUINGA LUIS

CUASCOTA EDWIN

**DOCENTE:**

DIEGO PINTO

**2023-2023**

## Contenido

<b>PREFACIO.</b>	3
<b>1. INTRODUCCIÓN.</b>	4
<b>2. OBJETIVOS.</b>	4
<b>2.1 General.</b>	4
<b>2.2 Específicos.</b>	4
<b>3. Requisitos.</b>	4
<b>4. Descarga</b>	5
<b>5. Creación de la base de datos</b>	5
.....	5
<b>6. Desarrollo de la App en Intellig.</b>	6
<b>6.1 Servicios implementados</b>	6
<b>6.2 Dependencias</b>	7
<b>6.3 Conexión a la base de datos.</b>	7
<b>7. Ejecución.</b>	8
<b>8. Conclusiones.</b>	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.**

Crear un manual para una aplicación distribuida con tolerancia a fallos es una tarea importante que busca proporcionar a los usuarios y desarrolladores una guía completa y detallada sobre cómo interactuar con el sistema y cómo garantizar su funcionamiento incluso en situaciones adversas.

### **2.2 Específicos.**

- Proporcionar una descripción detallada y clara de la arquitectura y los componentes de la aplicación distribuida, explicando cómo interactúan y se comunican entre sí.
- Proporcionar instrucciones detalladas sobre cómo configurar, instalar y desplegar la aplicación distribuida en diferentes entornos. Esto podría incluir consideraciones de hardware, software y configuración de red.
- Explicar cómo realizar pruebas de tolerancia a fallos y cómo monitorear la salud del sistema en busca de posibles problemas. Incluir herramientas y técnicas recomendadas.

## **3. Requisitos.**

Enlace al repositorio del Github

IntelliJ Idea (Cualquier versión)

Postgrest SQL

Java 17

Windows 10 home

Navegador web (Cualquier navegador)

Acceso a datos

#### 4. Descarga

Mediante el enlace colocada en la plataforma virtual ingresamos con cualquier navegador al repositorio del GitHub enlazado.

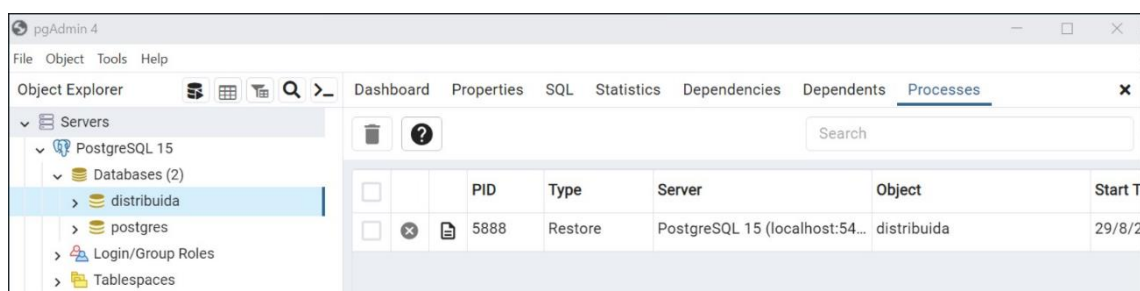
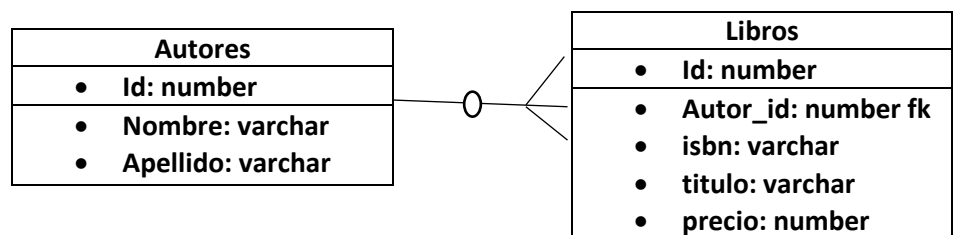
Luego procedemos a descargar todo el proyecto constatando que se encuentre las carpetas de los módulos app-authors y app-books como se muestra en la imagen.

Base de datos	evaluacion	3 hours ago
Documento	evaluacion	3 hours ago
Manual Técnico	evaluacion	3 hours ago
app-authors	evaluacion	3 hours ago
app-books	evaluacion	3 hours ago

#### 5. Creación de la base de datos

##### Opción 1

Utilizando nuestro modelo entidad relación procedemos a crear nuestra base de datos en el postgres sql.

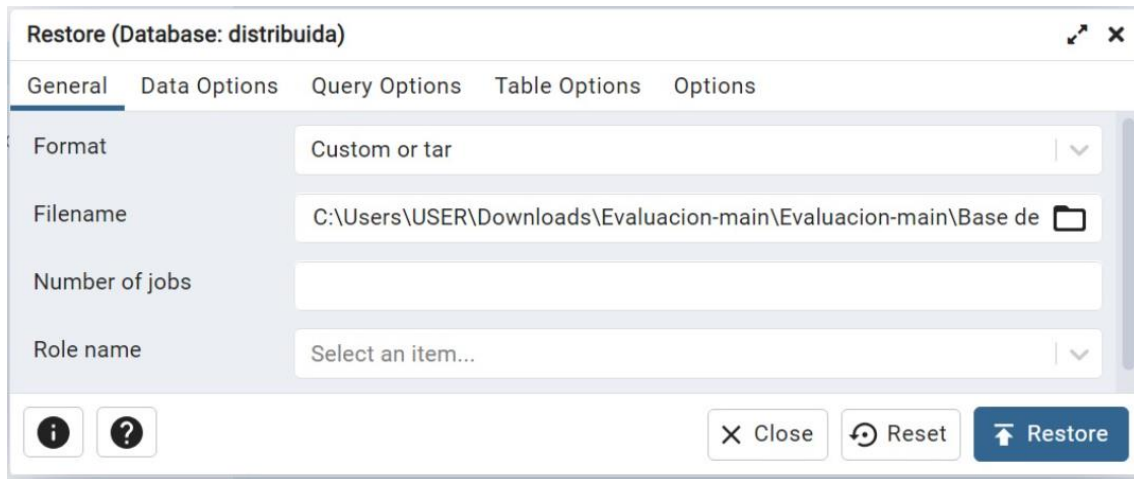


##### Opción 2.

Restaurar nuestro backup en postgres.

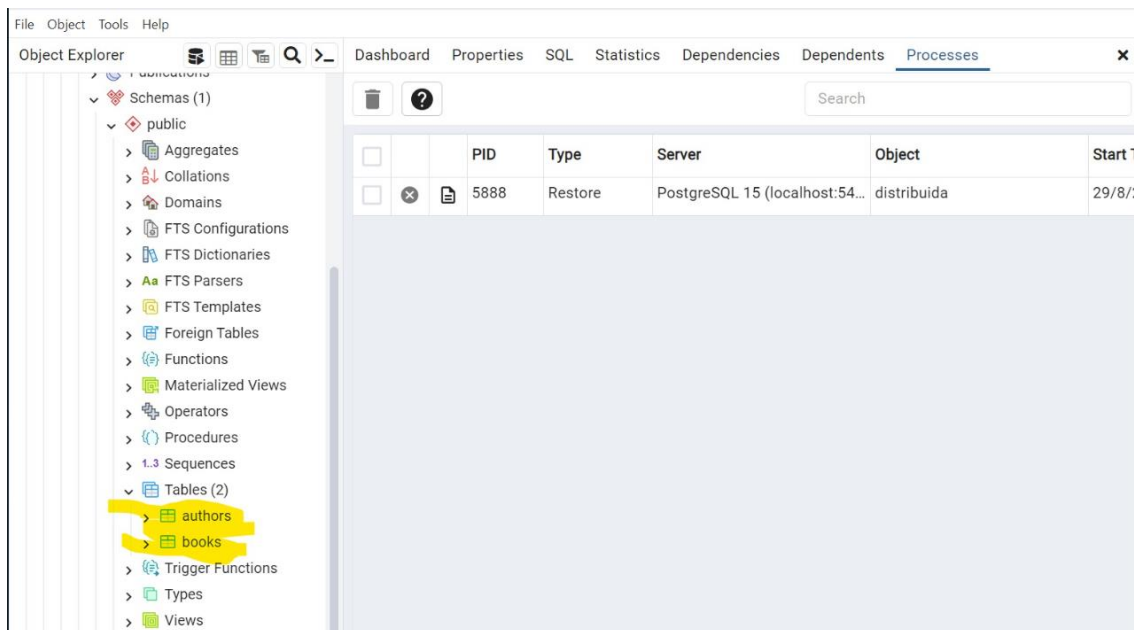
Priemramente procedemos a crear el nombre de la base de datos en mi caso es distribuida.

Luego procedemos a restaurar la base con el backup descargado del repositorio.



Exportamos la base que contenga el nombre de distribuida.sql que por lo general se encuentra en las descargas de nuestro computador.

Luego se podrá visualizar la restauración con las tablas de authors y books.

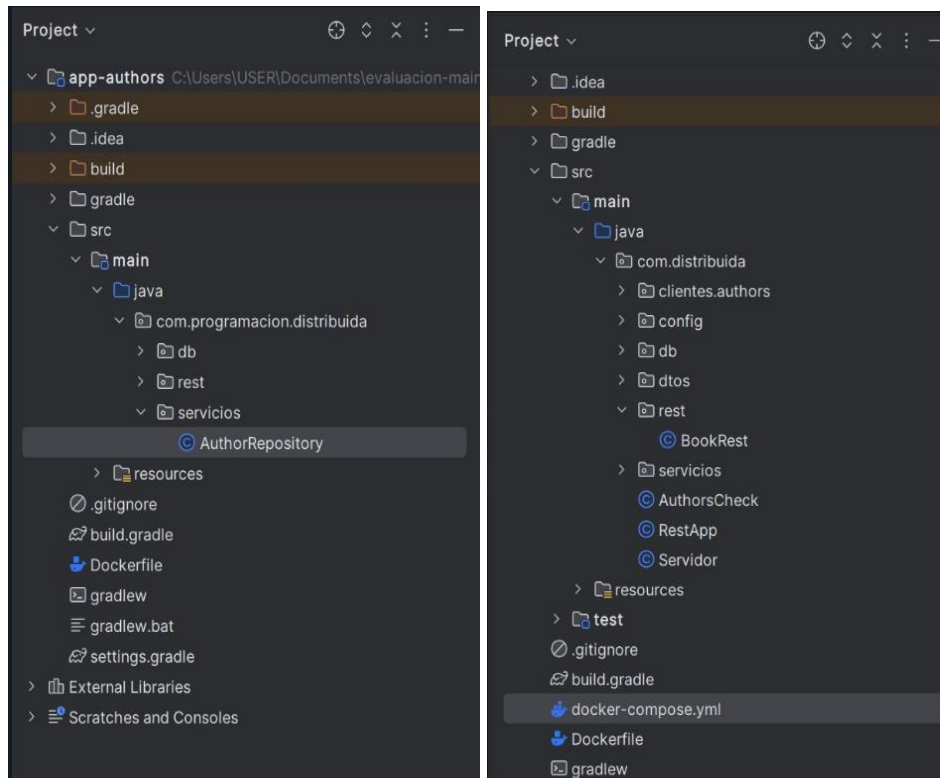


## 6. Desarrollo de la App en Intellig.

### 6.1 Servicios implementados

Primeramente inicializamos la herramienta Intellig.

Luego procedemos a abrir los srrvicios o modulos app-authors y app-books respectivamente.



## 6.2 Dependencias

Luego se procederá a descargar las dependencias que necesita nuestra app para su funcionamiento dependiendo de la velocidad de su ethernet se demora entre unos 5 a 10 minutos.

```

20 dependencies {
21     implementation platform("io.helidon:helidon-dependencies:${project.helidonVersion}")
22
23     //implementation 'io.helidon.microprofile.bundles:helidon-microprofile'
24     implementation 'io.helidon.microprofile.server:helidon-microprofile-server'
25     implementation 'io.helidon.microprofile.rest-client:helidon-microprofile-rest-client'
26
27     implementation 'io.helidon.dbclient:helidon-dbclient'
28     implementation 'io.helidon.dbclient:helidon-dbclient-jdbc'
29
30     implementation 'org.glassfish.jersey.media:jersey-media-json-binding'
31     //healthchecks
32     implementation 'io.helidon.microprofile.health:helidon-microprofile-health'
33     implementation 'io.helidon.health:helidon-health-checks'
34
35     //fault-tolerance
36     implementation 'io.helidon.microprofile:helidon-microprofile-fault-tolerance'
37     implementation 'io.helidon.microprofile.metrics:helidon-microprofile-metrics'
38
39     testImplementation 'io.helidon.microprofile.tests:helidon-microprofile-tests-junit5'
40     testImplementation 'org.junit.jupiter:junit-jupiter-api'

```

## 6.3 Conexión a la base de datos.

Para la conexión a la base de datos debemos tener configurado con el usuario y password personal, esto lo configuramos en el archivo de configuración application.properties.

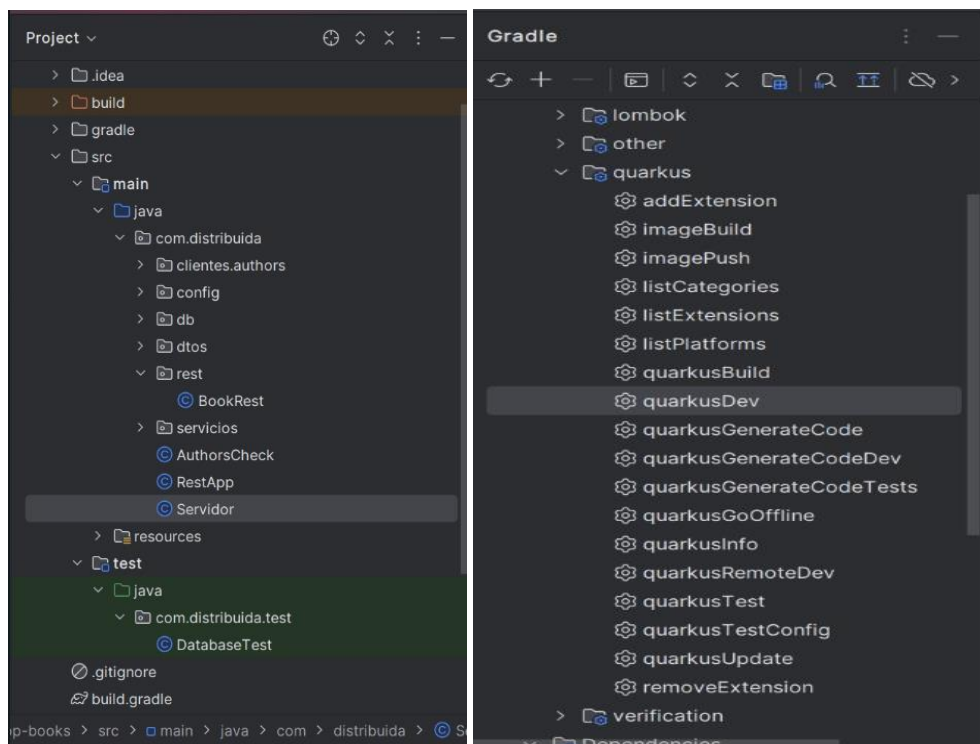


```
1 # Configuración de la base de datos
2 db.connection.url=jdbc:postgresql://127.0.0.1:5432/distribuida
3 db.connection.username=postgres
4 db.connection.password=1234
5 tracing.service=app-books
6
7 ## servicios REST a los que se conecta
8 author.url=http://localhost:8080
9 author/mp-rest/url=${author.url}
10
```

## 7. Ejecución.

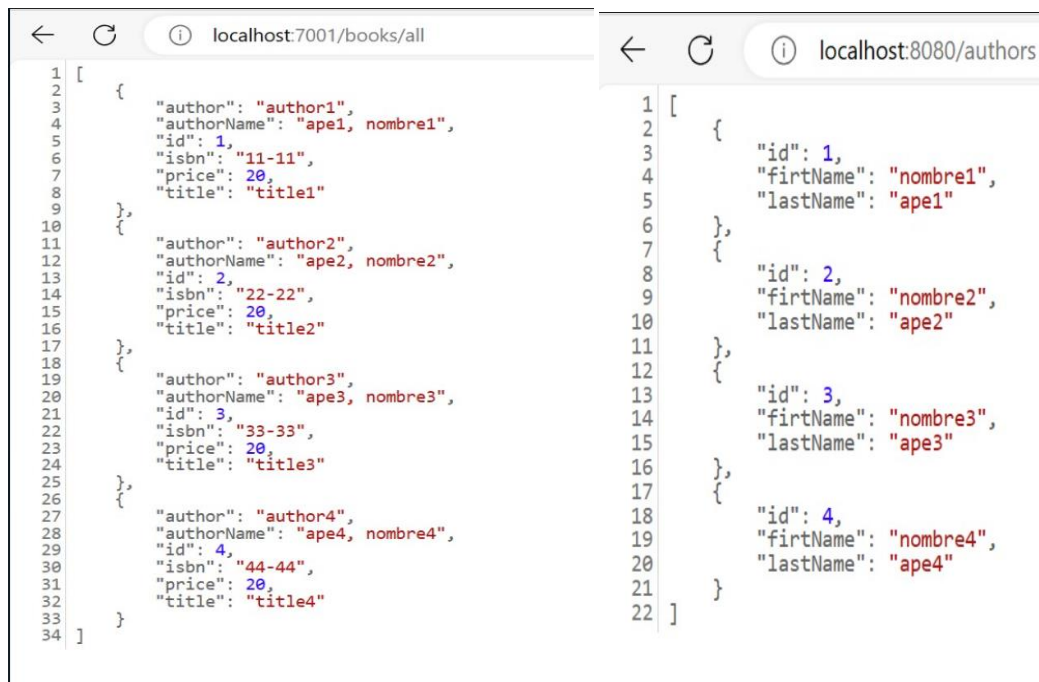
En el módulo app-books buscamos el archivo servidor y lo mandamos a ejecutar.

Mientras que en el app-authors ejecutamos me diante el gradle dando doble click en quarkus build para asegurar que todas las dependencias esten instaladas y luego en quarkus dev para ejecutar el servicio.



Una vez este corriendo las dos apps se mostrará en pantalla los datos que tenemos en la base de datos.





```
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 ]
```

***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.

**8. 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.