

Participantes: Francisco González Prieto, Adrián Brihuega Sánchez y Alexander Pulupa Romero

PRÁCTICA 3: Aplicación usando CORBA

Para poder realizar y ejecutar este programa necesitaremos un editor como IntelliJ, Eclipse, Visual Studio y el JDK de Java, que lo puedes descargar de [0'](#)

1. Vamos a trabajar con una Biblioteca en CORBA (1 punto)

La aplicación contendrá un archivo IDL, un archivo servidor y uno de cliente. Todas las instrucciones de la aplicación deben estar comentadas en castellano, con nuestras palabras para argumentar que se entiende.

Compilaremos primero el IDL, luego el servidor y luego el cliente usando los siguientes códigos: respectivamente:

```
$ idlj -fall Biblioteca.idl
$ javac ServidorBiblioteca.java
$ javac ClienteBiblioteca.java
```

Para ejecutar el programa necesitamos tener abiertas tres ventanas de consola. La primera iniciará el puerto, la segunda ejecutará el servidor y la tercera el cliente. El código para ejecutarla es, respectivamente:

```
$ tnameserv -ORBInitialPort 1050
```

```
(base) adrian@adrians-mbp corba $ orbd -ORBInitialPort 1050
```

```
$ java ServidorBiblioteca -ORBInitialHost localhost -ORBInitialPort 1050
```

```
(base) adrian@adrians-mbp corba $ java ServidorBiblioteca -ORBInitialPort 1050 -ORBInitialHost localhost
El servidor de la biblioteca está listo y esperando ...
```

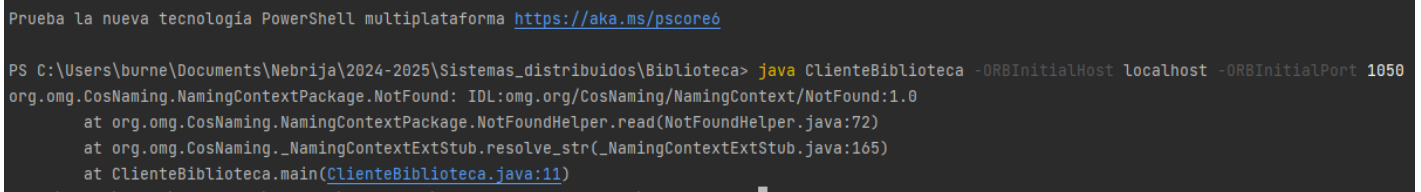
```
$ java ClienteBiblioteca -ORBInitialHost localhost -ORBInitialPort 1050
```

```
(base) adrian@adrians-mbp corba $ java ClienteBiblioteca -ORBInitialPort 1050 -ORBInitialHost localhost
Libro encontrado: El principito, Antoine de Saint-Exupéry, ISBN: 1234
Libro prestado con éxito.
```

Realiza unas capturas de que tienes en funcionamiento la aplicación

2. Preguntas sobre la Biblioteca en CORBA (puedes añadir capturas) (1 puntos):

a. ¿Qué sucede si lanzo antes el cliente que el servidor?

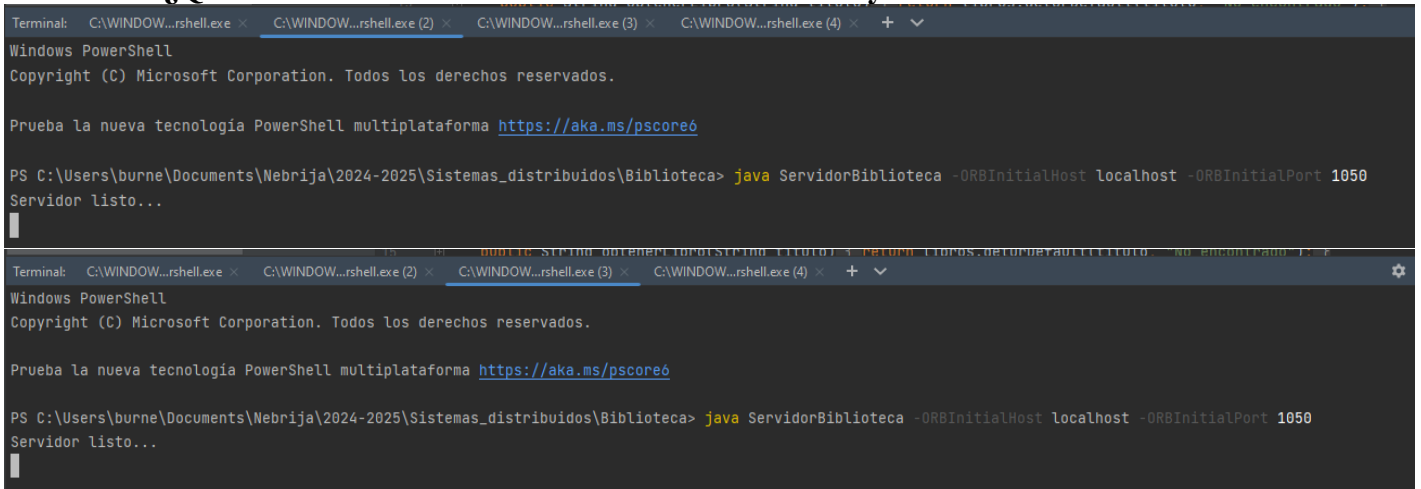


```
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\burne\Documents\Nebrija\2024-2025\Sistemas_distribuidos\Biblioteca> java ClienteBiblioteca -ORBInitialHost localhost -ORBInitialPort 1050
org.omg.CosNaming.NamingContextPackage.NotFound: IDL:omg.org/CosNaming/NamingContext/NotFound:1.0
    at org.omg.CosNaming.NamingContextPackage.NotFoundHelper.read(NotFoundHelper.java:72)
    at org.omg.CosNaming._NamingContextExtStub.resolve_str(_NamingContextExtStub.java:165)
    at ClienteBiblioteca.main(ClienteBiblioteca.java:11)
```

Si se intenta lanzar el cliente antes que el servidor se produce un error, debido a que este no puede localizarlo como consecuencia de que no se haya lanzado ni el servidor, ni, por consiguiente, el servicio de biblioteca.

b. ¿Qué sucedería si lanzase varios servidores a la vez y un solo cliente?



```
Terminal: C:\WINDOW...rshell.exe x C:\WINDOW...rshell.exe (2) x C:\WINDOW...rshell.exe (3) x C:\WINDOW...rshell.exe (4) x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\burne\Documents\Nebrija\2024-2025\Sistemas_distribuidos\Biblioteca> java ServidorBiblioteca -ORBInitialHost localhost -ORBInitialPort 1050
Servidor listo...

Terminal: C:\WINDOW...rshell.exe x C:\WINDOW...rshell.exe (2) x C:\WINDOW...rshell.exe (3) x C:\WINDOW...rshell.exe (4) x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\burne\Documents\Nebrija\2024-2025\Sistemas_distribuidos\Biblioteca> java ServidorBiblioteca -ORBInitialHost localhost -ORBInitialPort 1050
Servidor listo...
```

En mi caso no surge ningún error (se puede ver que tengo dos servidores en el mismo puerto funcionando en dos terminales distintas), esto puede deberse a que quizás al ya haber uno abierto, la segunda terminal te redirija a este o que se esté sobrescribiendo el anterior para crear el nuevo idéntico (En resumen, puede que se esté produciendo un **rebind**). Tras esto, el cliente se ejecuta con normalidad.

Si se crean dos servidores con puerto distinto (1050 y 1060) por ejemplo, el cliente localizará el que le indiques.

c. ¿Puedes conectarte al servidor de un compañero? ¿Cómo lo harías?

Ahora mismo estamos haciendo la práctica desde casa y no lo podemos comprobar, pero seguramente si los



dos estamos conectados a la misma red y el que se conecta al otro apunte a su dirección ip.

3. Actualiza el código añadiendo un par de funciones más a la gestión de la biblioteca así como una modificación en la estructura. Ejecútalo y haz capturas. Intenta que sea una aplicación/funcionalidad diferente la que modificas, diferenciate. (7 puntos)

```
public void eliminarLibro(String var1) { this.libros.remove(var1); }

public String buscarPorAutor(String var1) {
    Iterator var2 = this.libros.entrySet().iterator();

    Map.Entry var3;
    do {
        if (!var2.hasNext()) {
            return "No encontrado";
        }

        var3 = (Map.Entry)var2.next();
    } while(!((String)var3.getValue()).equals(var1));

    return (String)var3.getKey();
}
```

```
PS C:\Users\burne\Documents\Nebrija\2024-2025\Sistemas_distribuidos\Biblioteca>
ava ClienteBiblioteca -ORBInitialHost localhost -ORBInitialPort 1050
Libro encontrado: Orwell
Buscar por autor: 1984
Libro tras eliminaci³n: No encontrado
```

```
System.out.println("Buscar por autor: " + var4.buscarPorAutor("Orwell"));
var4.eliminarLibro("1984");
System.out.println("Libro tras eliminaci³n: " + var4.obtenerLibro("1984"));
```

Se puede ver que busca correctamente el título del libro según el autor (1984 de George Orwell).

También hemos implementado la función de eliminar libros, que se ejecuta también sobre el libro “1984”, y luego se intenta volver a obtener el libro esta vez por título para ver si lo encuentra. No lo hace y sale el mensaje de “No encontrado” esperado.

4. Ahora quiero que hagas un fichero README.md en markdown con información de qué versiones has utilizado, de como se ejecuta la práctica y con un diagrama explicando que es lo que has añadido. (1 punto)

Guión paso a paso para realización de la práctica:

1. Preparación del Entorno de Desarrollo

Antes de comenzar a programar, necesitas tener el entorno de desarrollo adecuado configurado en tu PC. Asegúrate de instalar los siguientes componentes:

- **Java Development Kit (JDK):**
 - Asegúrate de tener instalada la versión 8 o superior de Java. Si no lo tienes, descárgalo desde [la página oficial de Oracle](#).
- **Editor de código:**
 - Puedes usar un editor de código como IntelliJ IDEA, Eclipse o Visual Studio Code para escribir y compilar el código.
- **CORBA:**
 - Asegúrate de tener un servidor CORBA configurado, como JacORB o OpenORB, que permita la ejecución de la aplicación CORBA.

2. Crear el Archivo IDL (**Biblioteca.idl**)

El primer paso es definir las interfaces que serán utilizadas por el servidor y el cliente. Esto se hace en un archivo IDL (Interface Definition Language).

Pasos:

1. Crea un archivo llamado **Biblioteca.idl**.
2. Define las operaciones que vas a implementar en el servidor. Por ejemplo:
 - **obtenerLibro**: Para obtener un libro por su título.
 - **agregarLibro**: Para agregar un libro a la biblioteca.

El archivo IDL debe verse de la siguiente manera:

```
module Biblioteca {  
  
    interface GestionBiblioteca {  
  
        string obtenerLibro(in string titulo);  
  
        void agregarLibro(in string titulo, in string autor);  
  
    }  
}
```

};

Tarea para los alumnos:

- Añadir una nueva operación en la interfaz **GestionBiblioteca**. Puedes implementar algo como **eliminarLibro** o **buscarPorAutor**. Tendrás que decidir qué tipo de operación quieres agregar para hacer la biblioteca más funcional.

3. Compilación del Archivo IDL

Una vez tengas el archivo IDL listo, debes compilarlo para generar las clases Java que necesitarás para la comunicación entre el servidor y el cliente.

Pasos:

1. Abre una terminal o consola en la carpeta donde se encuentra tu archivo **Biblioteca.idl**.

Ejecuta el siguiente comando para compilar el archivo IDL:

```
$ idlj -fall Biblioteca.idl
```

Este comando generará varios archivos Java que contienen las interfaces necesarias para que el cliente y el servidor interactúen a través de CORBA.

Tarea para los alumnos:

- Verifica que los archivos Java se generen correctamente. Si hay algún problema, revisa el archivo IDL y asegúrate de que esté bien escrito.

4. Implementación del Servidor CORBA (**ServidorBiblioteca.java**)

El servidor es el componente que implementa la interfaz definida en el archivo IDL. El servidor debe gestionar los libros y permitir a los clientes agregar y obtener libros.

Pasos:

1. Crea un nuevo archivo Java llamado **ServidorBiblioteca.java**.
2. Implementa la interfaz **GestionBiblioteca** generada a partir del archivo IDL.
3. En esta clase, debes:

- Mantener una lista de libros (puedes usar una lista de tipo `ArrayList` de `Strings`).
- Implementar los métodos `obtenerLibro` y `agregarLibro`.

4. Registra el servidor en el naming service para que el cliente pueda encontrarlo.

Tarea para los alumnos:

- Implementar una nueva función como `eliminarLibro` o `buscarPorAutor`. Esto hará que la biblioteca sea más completa.
- Asegúrate de que el servidor esté correctamente registrado en el naming service para que el cliente pueda acceder a él.

5. Implementación del Cliente CORBA (`ClienteBiblioteca.java`)

El cliente es el que invoca los métodos del servidor. Se conecta al servidor CORBA y ejecuta las operaciones definidas en el archivo IDL.

Pasos:

1. Crea un nuevo archivo Java llamado `ClienteBiblioteca.java`.
2. Conéctate al servidor a través del naming service usando la dirección del servidor.
3. Invoca los métodos `agregarLibro` y `obtenerLibro` desde el cliente.

Tarea para los alumnos:

- Asegúrate de que el cliente pueda agregar libros y obtener información de ellos correctamente.
- Añadir la nueva operación que implementaste en el servidor al cliente para que también pueda ejecutar la nueva funcionalidad (como eliminar un libro o buscar por autor).

6. Ejecución de la Aplicación

Una vez que el servidor y el cliente estén listos, tendrás que ejecutar la aplicación. Necesitarás abrir tres consolas para iniciar el naming service, el servidor y el cliente.

Pasos:

Iniciar el Naming Service (en la primera ventana de consola):

\$ tnameserv -ORBInitialPort 1050

Ejecutar el Servidor (en la segunda ventana de consola):

\$ java ServidorBiblioteca -ORBInitialHost localhost -ORBInitialPort 1050

Ejecutar el Cliente (en la tercera ventana de consola):

\$ java ClienteBiblioteca -ORBInitialHost localhost -ORBInitialPort 1050

Tarea para los alumnos:

- **Verifica que las tres consolas se ejecuten sin errores.**
- **Realiza pruebas de comunicación entre el cliente y el servidor usando los comandos que ejecutas en la consola. Asegúrate de que todo funcione correctamente.**

7. Responder las Preguntas del Ejercicio

Una vez que tengas todo funcionando, responde las siguientes preguntas en tu informe:

- 1. ¿Qué sucede si lanzo antes el cliente que el servidor?**
 - **Explica lo que sucedería si el cliente intenta conectarse antes que el servidor.**
- 2. ¿Qué sucedería si lanzase varios servidores a la vez y un solo cliente?**
 - **Analiza qué ocurriría si se ejecutan varios servidores y un solo cliente.**
- 3. ¿Puedes conectarte al servidor de un compañero? ¿Cómo lo harías?**
 - **Explica cómo podrías conectarte al servidor de un compañero. ¿Qué cambios harías en el cliente?**
- 4. Actualizar el código añadiendo un par de funciones más:**
 - **Implementa una nueva funcionalidad en la biblioteca (como eliminar un libro o buscar por autor) y describe el cambio que realizaste. Recuerda modificar tanto el servidor como el cliente para que interactúen con esta nueva funcionalidad.**



UNIVERSIDAD
NEBRIJA