Práctica 02: INTERACCIÓN ENTRE 2 SERVIDORES
(Será realizado en la Sala de Computo)
Grupo A

La División de Gestión de la Recreación y el Deporte de la Universidad del Cauca es una dependencia adscrita a la Vicerrectoría de Cultura y Bienestar, definida como el componente operativo del Sistema Institucional de Deporte y Recreación Universitarios. Esta dependencia cuenta con 3 programas: Formación deportiva, Estilos de vida saludable y Unicauca activa de corazón. En este caso se va a centrar en el programa de Estilos de vida saludable. El programa Estilos de vida saludable se subdivide en 3 sub-programas: Programa Hora saludable, Acondicionamiento físico para estudiantes, Gimnasio para los administrativos.
Como soporte del programa de Hora saludable se cuenta con el siguiente personal: una secretaria y profesional en acondicionamiento físico. El registro de usuarios (docentes y administrativos) está a cargo de la secretaria, el profesional en acondicionamiento físico encargado de realizar valoraciones físicas elaboración de los programas físicos de los usuarios y registro de asistencia.
La División de Gestión de la Recreación y el Deporte de la Universidad del Cauca requiere que los usuarios tales como: secretaria y profesional de acondicionamiento físico puedan ser registrados en el servidor de gestión de personal y una vez registrados que este personal pueda abrir sesión en la aplicación.

Para el registro y consulta del personal los datos a registrar corresponden a: tipo de identificación (cc, ti, pp), número de identificación, nombre completo, ocupación (secretaria, paf), usuario, clave. El sistema debe gestionar la información utilizando el modelo de RMI e implementándolo por medio de Java RMI.
Al iniciar el programa se debe desplegar el siguiente menú:

```
 ==== Menú Inicio ====
* 1-Abrir sesión      *
* 2-Salir             *
 ======================
```

Figura 1. Menú de ingreso

La primera vez que se ejecuta la aplicación se debe abrir sesión para el administrador, ingresando las credenciales del administrador del programa (usuario, clave) y el número de identificación. El Menú del administrador del programa debe contar con 3 opciones: registrar personal, consultar personal y salir. Estas opciones serán controladas mediante un Menú, tal como muestra la Figura 2:

```
====== Menú Admin ======
* 1-Registrar personal *
* 2-Consultar personal *
* 3-Salir              *
 ======================
```

Figura 2. Menú del administrador

La primera opción permite registrar todos los datos de un usuario. En el lado del servidor los datos de los usuarios serán almacenados en un arreglo. La máxima cantidad de usuarios a registrar será de 5 usuarios.
Si el registro de usuario es exitoso se debe emitir un mensaje: ** Usuario registrado exitosamente **.
Si el registro el registro no es exitoso se debe emitir el mensaje: ** Usuario NO registrado, se alcanzó la cantidad máxima de usuarios a registrar **.
En la segunda opción si la consulta es exitosa se debe desplegar la siguiente información del usuario: Tipo de identificación, Identificación, Nombre completo, Role, Usuario.
Si la consulta no es exitosa se debe desplegar el mensaje: **Usuario NO encontrado**.

Para abrir sesión de un personal con ocupación secretaria o paf se deben ingresar las credenciales del personal (usuario, clave) y el número de identificación. Si el usuario que ingresa es una secretaria o el paf y ya están registrados se debe desplegar el menú que se muestra en la Figura 1.
Si las credenciales son correctas y la ocupación es una secretaria se debe desplegar el siguiente menú (Figura 3):

```
=== Menú Secretaria ===
* 1-Registrar usuario *
* 2-Consultar usuario *
* 3-Salir             *
======================
```

Figura 3. Menú de la secretaria

Si las credenciales son correctas y la ocupación es un paf se debe desplegar el siguiente menú (Figura 4):

```
====== Menú PAF =========
* 1-Valorar PAF          *
* 2-Registrar asistencia *
* 3-Salir                *
=========================
```

Figura 4. Menú del PAF

En el servidor gestión de usuarios, se debe verificar si la ocupación es un paf, si es así se debe enviar una notificación al servidor seguimiento usuarios, donde se debe desplegar el mensaje:
El personal [nombre completo] y ocupación [ocupación] está autorizado para ingresar al sistema.

Si las credenciales no son correctas, en la pantalla del nodo cliente se le mostrará el siguiente mensaje:

** El personal [usuario] no está autorizado para ingresar al sistema. **
** Verificar que el usuario y la clave sean las correctas. **

Desarrollo de la aplicación

La aplicación debe implementarse usando Java RMI, mediante el lenguaje de programación Java. En la Figura 5 se observa el diagrama de contexto que muestra las

operaciones que puede realizar el cliente. Las operaciones deben ser implementadas en 2 servidores denominados servidor gestión usuarios y servidor seguimiento usuarios, cada vez que se realice una petición en el cliente y se reciba una petición en los servidores, **se debe crear un eco por medio de un mensaje en pantalla** en el cual se describe cual función se está pidiendo o ejecutando.

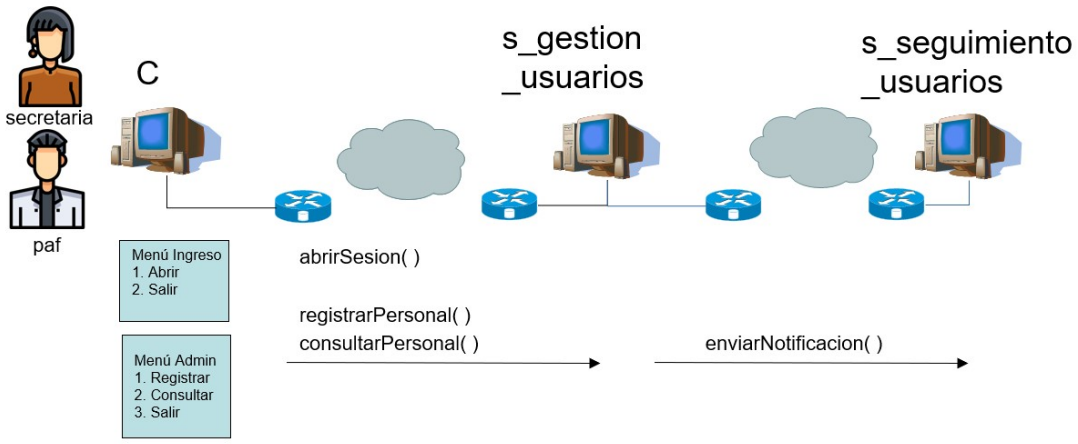

Figura 5. Sistema de registro

Tener en cuenta: La solución de este ejercicio debe ser comprimida en formato rar y enviada a la plataforma univirtual. El nombre del archivo comprimido debe seguir el siguiente formato lsdA_corba02_gx.rar. Donde gx corresponde al nombre de grupo asignado en este curso. En la carpeta se debe incluier el archivo role.txt, donde se especifica el role de cada integrante del grupo de trabajo.

Organizar los archivos de acuerdo a la estructura de directorios que se muestra en la Figura 6

```
lsdA-corba02-gx/
├── bin
└── src
    ├── cliente
    │   └── utilidades
    ├── s_gestion_usuarios
    │   ├── servidor
    │   └── sop_corba
    │       └── GestionUsuariosPackage
    └── s_seguimiento_usuarios
        ├── servidor
        └── sop_corba
            └── GestionNotificacionesPackage
```

Figura 6. Estructura del directorio de trabajo.

1. Guía de compilación y ejecución.

El primer paso para crear aplicaciones basadas en CORBA es especificar las operaciones remotas utilizando el lenguaje de definición de interfaces (IDL), el cual puede ser mapeado a una variedad de lenguajes de programación.

La interface IDL (ver Figura 6)para el servidor de gestión de usuarios es la siguiente:

```
module sop_corba{
 interface GestionPersonal{
        struct personalDTO{
                string tipo_id;
                long id;
                string nombreCompleto;
                string ocupacion;
                string usuario;
                string clave;
        };

        struct credencialDTO{
                string usuario;
                string clave;
        };

   void registrarPersonal(in personalDTO objPersonal,out boolean resultado);
   boolean consultarPersonal(in long id, out personalDTO objPersonal);
   boolean abrirSesion(in credencialDTO objCredencial);
};
 };
```

Figura 6. Contenido del archivo gusuarios.idl

La interface IDL (ver Figura 7) para el servidor de seguimiento de usuarios es la siguiente:

```
module sop_corba{
 interface GestionNotificaciones{
        struct notificacionDTO{
                string nombreCompleto;
                string ocupacion;
        };
  void enviarNotificacion(in notificacionDTO objNotificacion);
 };
};
```

Figura 7. Contenido del archivo gnotificaciones.idl

Compilar la interface idl de los servidores de gestión de usuarios y seguimiento de usuarios utilizando los siguientes comandos:
Servidor de gestión de usuarios:

```
idlj -fallTIE -pkgPrefix sop_corba s_gestion_usuarios gusuarios.idl
```

Servidor de seguimiento de usuarios:

```
idlj -fallTIE -pkgPrefix sop_corba s_seguimiento_usuarios gnotificaciones.idl
```

El compilador jidl de java permite convertir una interface definida en lenguaje IDL a correspondientes interfaces java, clase y métodos, los cuales pueden ser utilizados para implementar el código del cliente y servidor.

Este comando generará varios archivos. Revisar el contenido del directorio de trabajo desde donde ejecuto este comando. Un nuevo subdirectorio será creado, debido a que el módulo 'sop_corba' será mapeado como un paquete.

La opción **-fallTIE** genera los archivos de soporte que son utilizados para implementar el Modo TIE. De esta manera una clase GestionPersonalPOATie.java es generada. El constructor de esta clase toma un delegado. Se debe proporcionar la implementación de una clase delegada. Además, genera el paquete **GestionPersonalPackage, el cual contiene clases que proveen operaciones para leer y escribir sobre argumentos de las operaciones remotas.**

2. Utilizar y completar las plantillas del cliente y el servidor

Utilizar las plantillas que se encuentran almacenadas en el sitio destinado al curso. Crear los archivos fuente para el lado cliente ClienteDeObjetos.java, para el lado del servidor de gestión de usuarios ServidorDeObjetos.java y GestionPersonalImpl.java, y para el lado servidor de seguimiento de usuarios ServidorDeObjetos.java y GestionNotificacionesImpl.java, teniendo en cuenta que:

ClienteDeObjetos.java: Implementa la lógica de negocio del cliente, consulta una referencia del servant y llama las operaciones del objeto remoto a través de un menú.

GestionPersonalImpl.java: Se implementa el objeto corba en Modo Delegación, implementa la interfaz GestionPersonalOperations.java. Implementa la lógica de las operaciones definidas en la interface IDL.

GestionNotificacionesImpl.java: Se implementa el objeto corba en Modo Delegación, implementa la interfaz GestionNotificacionesOperations.java. Implementa la lógica de las operaciones definidas en la interface IDL.

ServidorDeObjetos.java: Implementa la lógica para crear el servant y registrarlo en el ns.

Distribuir estos archivos fuente en los subdirectorios pertinentes de src/. Para registrar el servant del lado servidor de gestión de usuarios y consultarlo se utilizará como nombre "**objRemotoPersonal**".
Para registrar el servant del lado servidor de seguimiento de usuarios y consultarlo se utilizará como nombre "**objRemotoNotificaciones**".

3. Compilar los códigos fuente

Compilar los códigos fuente del ítem 2 y los stubs y skeleton que fueron generados por el precompilador idlj en el paso 1, por medio de los siguientes comandos:

Para compilar los soportes:
Servidor de gestión de usuarios:
```
javac -d ../bin s_gestion_usuarios/sop_corba/*.java
```
Servidor de seguimiento de usuarios:
```
javac -d ../bin s_seguimiento_usuarios/sop_corba/*.java
```

Para compilar los archivos fuente del servidor:
Servidor de gestión de usuarios:
```
javac -d ../bin s_gestion_usuarios/servidor/*.java
```

Servidor de seguimiento de usuarios:
```
javac -d ../bin s_seguimiento_usuarios/servidor/*.java
```

Para compilar los fuentes del lado cliente:

```
javac -d ../bin cliente/*.java
```

4. Lanzar el Object Request Broker Daemon (ORBD)

Antes de correr el cliente y servidor, se debe correr el ORBD, el cual es un servicio de nombrado. Un servicio de nombres de CORBA es un servicio que permite a una referencia de un objeto CORBA asociarla con un nombre. El nombre del enlace se puede almacenar en el servicio de nombres, y un cliente puede proporcionar el nombre para obtener la referencia al objeto deseado.

a. Lanzar el orbd mediante el comando:

```
orbd -ORBInitialHost dir_IP -ORBInitialPort N_Puerto
```

b. Ubicarse en el directorio bin/. Lanzar el Servidor de seguimiento de usuarios mediante el comando:

```
java s_seguimiento_usuarios.ServidorDeObjetos -ORBInitialHost
dir_IP -ORBInitialPort N_Puerto
```

c. Ubicarse en el directorio bin/. Lanzar el Servidor de gestión de usuarios mediante el comando:

```
java s_gestion_usuarios.ServidorDeObjetos -ORBInitialHost dir_IP -
ORBInitialPort N_Puerto
```

dir_IP y N_Puerto son la dirección ip y puerto donde se encuentra escuchando el servicio orbd.

d. Ubicarse en el directorio bin/. Lanzar el Cliente mediante el comando:

```
java cliente.Cliente -ORBInitialHost dir_IP -ORBInitialPort
N_Puerto
```

dir_IP y N_Puerto son la dirección ip y puerto donde se encuentra escuchando el servicio orbd.