Práctica 1: PROCESO BÁSICO DE DESARROLLO CON CORBA

Práctica 1. (Será realizada en la Sala de Computo)

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) esta a cargo de las secretaria, el profesional en acondicionamiento físico encargado de realizar valoraciones físicas elaboración de los programas físicos de los usuarios y registro de asistencia.

Para la implementación del programa Hora saludable se requiere de una aplicación distribuida que permita la gestión del personal de apoyo (secretaria y profesional en acondicionamiento físico). La aplicación debe permitir el registro y consulta del personal de apoyo del programa Hora Saludable. Lo datos a registrar corresponden a: tipo de identificación(cc, ti, pp), número de identificación, nombre completo, ocupación (secretaria, profesional de acondicionamiento físico), usuario (mínimo 8 caracteres alfanuméricos), contraseña(mínimo 8 caracteres alfa numéricos). El sistema debe gestionar la información utilizando el modelo CORBA e implementándolo por medio de Java IDL de Oracle.

El administrador del programa debe contar con 2 opciones: registrar usuario y consultar usuario. El sistema debe gestionar la información utilizando el modelo de CORBA e implementándolo por medio de Java IDL . Estas opciones serán controladas mediante un Menú, tal como muestra la Figura 1:

```
======= M E N U =======
1. Registrar Personal.
2. Consultar Personal.
3. Salir.
=======================
Seleccione una opcion:
```

Figura 1. Menú del cliente

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 2.
.- Si el registro de usuarios es exitoso se debe emitir un mensaje: *** Personal registrado exitosamente ***.

.- Si el registro el registro no es exitoso se debe emitir el mensaje: *** Personal NO registrado, se alcanzó la cantidad máxima de personas a registrar ***.
Si el registro no es exitoso por el ingreso de datos erróneos debe emitir el mensaje: **Personal NO registrado, error en el formato de los datos ingresados**.

.- En la segunda opción si la consulta es exitosa se debe desplegar la siguiente información del usuario: Tipo de identificación, Identificación, Nombre completo, Ocupación, Usuario.
Si la consulta no es exitosa se debe desplegar el mensaje: *** Personal con [id] NO encontrado ***.
Todos los datos de registro son obligatorios si se omite alguno no se debe permitir el registro de usuarios.

Desarrollo de la aplicación

En la Figura 2 se observa un diagrama de contexto que muestra las operaciones que puede realizar el administrador. Las operaciones deben ser implementadas en un servidor, cada vez que se realice una petición en el nodo cliente y se reciba una petición en el nodo servidor**, se deben crear ecos por medio de un mensaje en pantalla** en el cual se describe cual función se está solicitando o ejecutando.

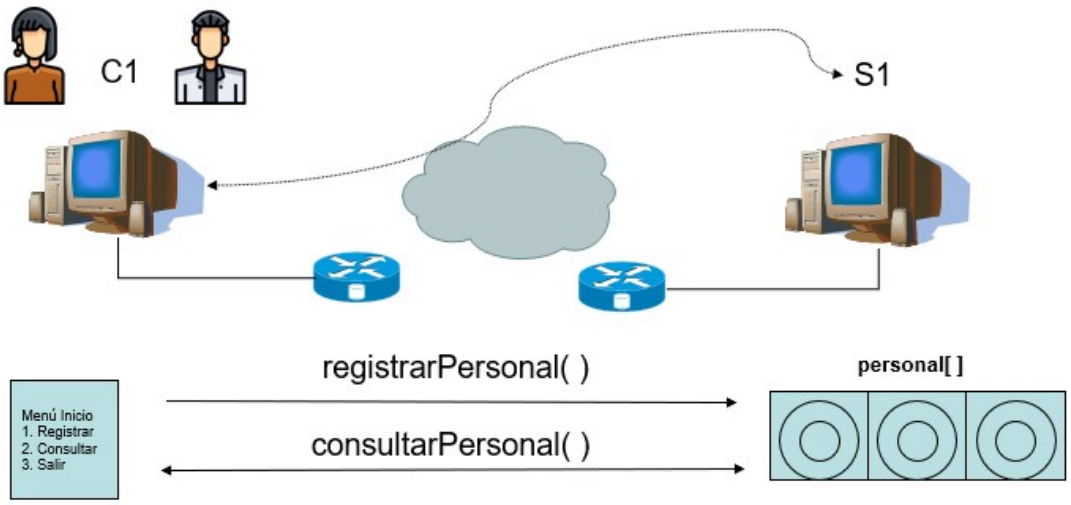

Figura 2. Diagrama de contexto del sistema

.

Tener en cuenta: La solución de este ejercicio debe ser comprimida en formato rar o zip y enviada a la plataforma de Univirtual. El nombre del archivo comprimido debe seguir el siguiente formato lsd_corba01_gx. Donde gx, corresponde al nombre del grupo asignado en este curso. En la carpeta se debe almacenar el archivo roles.txt donde se especificar el role de cada integrante.

Antes de iniciar, estructurar una carpeta de trabajo donde se ubicaran los archivos fuente (directorio 'src') y los archivos binarios (bytecode) (directorio bin). El nombre del directorio

de trabajo debe coincidir con el nombre del archivo comprimido. La estructura del directorio de trabajo es mostrada en la Figura 3.

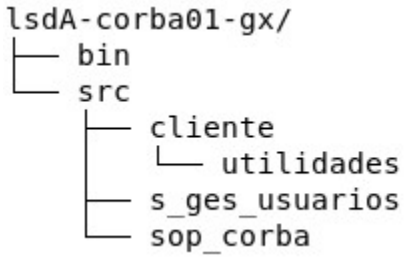

Figura 3. Estructura del directorio de trabajo

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. Editar el archivo **ges-personal.idl** que contiene la interface IDL para esta práctica (ver figura 4):

```
// ges-personal.idl
// tipos de id: cc,ti,pp
module sop_corba{
interface GestionPersonal{

        struct personalDTO{
        string tipo_id;
        long id;
        string nombreCompleto;
        string ocupacion;
        string usuario;
        string clave;
};

  void registrarPersonal(in personalDTO objPersonal, out boolean res);
  boolean consultarPersonal(in long id,out personalDTO objPersonal);
 };
};
```

Figura 4. Contenido del archivo ges-asintomaticos.idl

Ubicarse en la carpeta src/. Compilar la interface idl utilizando el siguiente comando:

idlj -fall ges-personal.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 - **fall** genera los archivos del stub y skeleton para el cliente y servidor. 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 (descargar **plantillas-corba.zip**) que se encuentran publicadas en el sitio del curso en univirtual. Crear los archivos fuente ClienteDeObjetos.java, ServidorDeObjetos.java y GestionPersonalImpl.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.

Tener en cuenta que se van a generar datos tipo XXHolder por tanto estos tipos de datos es necesario inicializarlos, por ejemplo:

```
BooleanHolder resp=new BooleanHolder();
.
.
ref.registrarPersonal(objPersonal, resp);
```

Para recuperar los datos de un dato tipo XXHolder se debe manejar el atributo value, por ejemplo:

```
boolean consulta = ref.consultarPersonal(id, objPersonal);
.
.
String nombreC = objPersonal.value.nombreCompleto;
```

GestionPersonalImpl.java: Hereda de GestionPersonalPOA. 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 y consultarlo se utilizara como nombre "***objRemotoPersonal***".

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:
javac -d ../bin sop_corba/*.java

Para compilar los archivos fuente del servidor:
javac -d ../bin servidor/*.java

Para compilar los archivos de soporte:
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 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.

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

java cliente.ClienteDeObjetos -ORBInitialHost dir_IP -ORBInitialPort N_Puerto

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