Consigna 4

Motivación:

Esta actividad tiene la finalidad de integrar elementos de orientación a objetos propios y de terceros con comunicaciones en redes de datos, usando programas en lenguaje C++.

En la Orientación a Objetos, un ***paquete*** (package) es una forma de organizar y agrupar clases, interfaces, y otros elementos relacionados dentro de un sistema. Sirve para estructurar el código, facilitar el mantenimiento y controlar la visibilidad (acceso) de los elementos.

Un paquete es un contenedor lógico de entidades que tienen una funcionalidad relacionada.

Es un concepto similar al de una carpeta del FS, que agrupa archivos relacionados.

El concepto de comunicación entre objetos es válido dentro de un mismo programa, a su vez dentro de un mismo dispositivo.

Cuando se trata de comunicación a través de la red, los objetos de una aplicación que deseen comunicarse con objetos remotos de otra aplicación, deben hacerlo usando los mecanismos y medios estándares que ofrece la plataforma. Esto lleva a que (salvo algunos casos especiales) los objetos existentes en un extremo, deban ser transformados antes de ser trasferidos y luego reconstruidos en el otro extremo.

Si bien existen diversas formas de comunicar 2 (o más) dispositivos, el modelo de interés aquí es el denominado Cliente/Servidor. Donde uno de los nodos (el Servidor) es capaz de proveer uno o más servicios que son consumidos por otro nodo (el Cliente). Cabe aclarar que en este modelo o arquitectura de red, puede haber varios Clientes consumidores concurrentes, incluso alguno puede encontrarse en el mismo equipo donde corre el servidor.

Requerimientos:

Esta actividad tiene la finalidad de diseñar 3 paquetes, destinados a diferentes nodos de una red, a partir del código fuente C++ entregado.

Este código usa una librería externa, que permite operar usando XML y RPC combinados, para intercambiar información entre dispositivos de esa red.

Cada uno de los paquetes debe contener al menos 1 clase y un módulo de lanzamiento separado. Los paquetes previstos corresponden 1 al lado servidor, y 2 a diferentes clientes.

Restricciones al diseño e implementación:

- ✓ Las librerías de terceros usadas deben encontrarse dentro de un directorio denominado lib (o library)
- ✓ Los archivos de cabecera propios deben encontrarse en un directorio denominado inc (o include)
- ✓ Cada paquete debe tener su propio directorio de código fuente
- ✓ El diseño OO debe separar la capa de presentación/control de la de modelo.
- ✓ Cambie los controles que usan if y mensajes directos a un modelo de gestión de errores mediante clases de excepción personalizadas, en los escenarios básicos de operación previstos.
- ✓ Incorporar guardas a las clases y espacios de nombre apropiados, asociados a cada package.

Descripción complementaria

El ejemplo provisto responde a servidor RPC capaz de proveer servicios, a uno o más clientes capaces de consumir dichos servicios.

El Servidor posee las siguientes características:

- ✓ Provee de diferentes servicios principales (prueba trivial, respuesta simple y cálculo) y auxiliares (de ayuda para guía o con sintaxis esperada, ya sea nativos de la librería o extendidos a partir de ella).

- ✓ Pone los valores a disposición de los clientes mediante XML-RPC o JSON-RPC, en una dirección IP y un puerto informados.
- ✓ Aplicación básica de línea de comandos

Existen 2 (dos) aplicaciones para el lado Cliente, con las siguientes características:

- ✓ Son aplicaciones de línea de comandos para prueba del servicio,
- ✓ Una para peticiones individuales y directas. Otra con un loop de peticiones preestablecidas.
- ✓ Despliegan algún tipo de ayuda mínima al operador, en caso que se equivoque.
- ✓ Producen una salida en pantalla luego de cada petición realizada al servidor.

Procedimiento general:

1) A partir del ejemplo provisto, rediseñe bajo principios de OO, cumpliendo las siguientes restricciones mencionadas anteriormente.

2) Generar los módulos que correspondan en lenguaje C++.

3) Realizar un informe similar a los anteriores.

4) Agregar una captura que muestre la distribución de los archivos en el árbol de directorios resultante

5) Agregue un gráfico con el diagrama de clases post desarrollo (mediante análisis o ingeniería inversa) donde se observen los packages del lado cliente y el lado servidor.

6) Preparar un archivo comprimido, similar a lo pautado en entregas anteriores.

7) Realizar la entrega del informe y del archivo comprimido con la implementación.

Recursos complementarios (disponibles en el aula virtual)**:**

- Apuntes de clase.
- Código de base con librerías complementarias (base_xml_rpc)
- Referencias y ejemplos provistos con documentación sobre protocolos, comunicaciones en redes y librerías de terceros.