# Programación de redes – Obligatorio 1 - 254254

Descripción de la arquitectura.

La arquitectura de la aplicación consta de dos aplicaciones principales y un set de proyectos auxiliares.

Los proyectos en la solución son:

**Common**: Utilidades referentes a el manejo de la configuración de las aplicaciones principales y el uso del Network Stream.

**DomainObjects**: Repositorio donde se encuentran los objetos.

**ProtocolLibrary**: Configuraciones y utilidades relacionados a los protocolos.

**VaporClient**: Aplicación cliente.

**VaporServer**: Aplicación servidor.

Detalle de los componentes principales

Cliente VaporClient:

El componente cliente está compuesto por una clase de presentación y una handler llamado ClientHandler.

La clase Client se encarga de la interacción con el usuario del cliente en tanto que la clase ClientHandler contiene la lógica necesaria para la comunicación con el servidor.

Gran parte de las tareas relacionadas a la comunicación se delega a las clase NetworkStream que es parte del proyecto Common.

Servidor VaporServer:

El servidor está distribuido de manera similar al cliente, pero agrega una clase más llamada Logic. Esta clase se encarga de ejecutar todo lo relacionado a la interacción con los objetos del proyecto DomainObjects.

El servidor usa sockets para establecer las conexiones con los clientes, cada conexión es manejada de manera independiente en un nuevo thread que se crea al recibirlo.

Mecanismos de comunicación:

Para establecer la conexión inicial entre las partes se crea a través del protocolo Tcp usando sockets. Una vez que la conexión está establecida, las siguientes comunicaciones son a través del protocolo creado para el obligatorio.

Se usó el protocolo propuesto por la letra, descrito en la siguiente tabla:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre Del Campo | **HEADER** | **CMD** | **LARGO** | **DATOS** |
| Valores | RES/REQ | 0-99 | Entero | Variable |
| Largo | 3 | 2 | 4 | Variable |

Los datos representados en el protocolo se mueven a través de un network stream, la clase NetworkStreamHandler provee métodos que ayudan a los dos proyectos principales a llevar a cabo la comunicación.

No se llegó a implementar la transferencia de imágenes en la comunicación, pero la idea era usar los handlers de filestream que vimos en clase en combinación con lo que se implementó para llevarla a cabo.

Otras decisiones de diseño e implementación:

En la implementación se detectó que el servidor y el cliente requieren muchas veces usar las mismas funcionalidades. En particular las funcionalidades referentes al uso del Network Stream o el protocolo.

Para poder reutilizar tanto de este código como fuera posible, se decidió intentar sacar todo lo que sea puramente inherente al manejo y preparación de paquetes para ser enviados por la red fuera de las aplicaciones cliente, de manera de poder mantener las responsabilidades de cada clase acotadas.

Los proyectos que responden a esto son Common y ProtocolLibrary.

Alcance de la entrega:

Las funcionalidades implementadas fueron las siguientes:

**Servidor:**

SRF1 - Aceptar pedidos de conexión de un cliente.

SRF2 - Ver Catalogo de Juegos.

SRF4 (Sin la foto) - Publicar un juego.

SRF7 - Ver detalle de un juego

**Cliente:**

CRF1 - Conectarse y desconectarse al servidor.

CRF6 (Sin la foto) - Detalle de un Juego

Ejecución de las aplicaciones:

Se debe ejecutar primero el servidor, haciendo "dotnet run" en una consola ubicada sobre la carpeta del proyecto.

De igual manera se hace con los clientes.

En ambas aplicaciones el sistema guía los pasos a seguir.