Universidad ORT Uruguay

Facultad de Ingeniería

Programación de Redes
Obligatorio 1

Entrega como requisito de la materia

Programación de Redes

Julieta Percivale 245079 - Hernán Reyes 235861

Tutor: Roberto Assandri - Delia Malvina Alvarez

30 de Setiembre de 2021

Índice

Descripción General del trabajo/sistema

Alcance del sistema

Principales decisiones
- Funcionalidades
- Persistencia
- Concurrencia
- Manejo de excepciones

Descripción y justificación de diseño
- Diagrama de paquetes
- Diagrama de clases
  - Cliente
  - Dominio
  - Protocolo
  - Servidor

Protocolo
- Encabezado
- Comandos - Acciones
- Constantes del Protocolo
- Mapper

Configuración de la Conexión
- Servidor
- Cliente

Descripción General del trabajo/sistema

Se desarrolló una aplicación de consola. La cual permite a los usuarios la gestión de juegos. Dentro de la gestión de juegos se permite publicar, dar de baja, modificar, buscar, adquirir, mostrar los adquiridos, calificar y ver el detalle de los mismos.

Dado que es una aplicación de múltiples usuarios se manejo la gestión de usuarios. En nuestro caso solo se permite el inicio de sesión de un usuario y el alta de los mismos.

Para poder gestionar los múltiples usuarios en diferentes conexiones se implementó un servidor para la conexión al mismo de los usuarios. En la misma se realizan todas las operaciones que cuenta el cliente y maneja los datos que se deben persistir pero este no realiza dichas operaciones por consola.
Asumimos que las acciones que cuenta el servidor a través de la consola son solo para desconectar el mismo y ver el listado de juegos que se encuentran dados de alta por los usuarios.

Alcance del sistema

En esta entrega pudimos cumplir con todos los requerimientos funcionales y no funcionales que se especificaba en la letra.

Dentro de los requerimientos funcionales se tomaron algunas decisiones donde se especifica en detalle en la sección de “Principales decisiones - Funcionalidades”.

- Mejoras a tener en cuenta

No se pudo implementar el algoritmo MD5. Lo vemos necesario en la transferencia de archivos para aportar seguridad en la transferencia y poder validar si los archivos fueron alterados o no.
La misma función no se pudo implementar por temas de tiempos dado que se implementó el envío de archivos sobre la fecha de entrega. Pero la función ya fue pensada y analizada, será implementada para nuestra segunda entrega.

Principales decisiones

Funcionalidades

- Se decidió que el servidor solo cuente con las operaciones de cerrar el servidor y ver lista de juegos. El resto de las funcionalidades pedidas quedan del lado del cliente siendo procesadas por el servidor.

- Existen usuarios, los cuales solo sirven para identificar quién realizó una calificación. No cuentan con contraseña y solo puede existir un único nombre de usuario en el sistema. Si se intenta iniciar sesión con un usuario que no existe se da de alta automáticamente y se inicia sesión. En caso de existir el usuario solo iniciamos sesión con el mismo.

- Buscar por calificación. En una calificación se ingresa cantidad de estrellas (entero del 1 al 5). Al buscar por calificación se está buscando por el promedio de esas estrellas, es decir, la suma de cada estrella por calificación dividido por la cantidad de calificaciones.
  Ej. Si cuento con un juego con una calificación de 5 estrellas y otra calificación para el mismo juego de 3 estrellas el promedio va a ser 4. Por tanto al buscar por calificación 4 nos devolverá dicho juego.

- El título de un juego debe ser único.

- Si se elimina la carátula del lado del servidor (eliminar el archivo de la carpeta) el cliente ya no podrá ver el detalle de dicho juego. Dado que el archivo (carátula) fue eliminado.

Persistencia

Fue implementada a través de la clase “Persistencia” la cual implementa el patrón Singleton para solo instanciarla una única vez.
La misma contiene una lista de usuarios y una lista de juegos.

Concurrencia

Para manejar la mutua exclusión en datos compartidos utilizamos Locks de .Net.
Creímos necesario utilizarlos en cualquier acción que intente dar de alta, modificar, eliminar o mostrar datos.
Esto lo creímos excluyente en las operaciones de alta (para no crear objetos repetidos), modificación (para no acceder al mismo objeto a la misma vez), eliminación (para no acceder a eliminar el objeto a la misma vez).
Por otro lado, creímos conveniente las operaciones de mostrar datos (acceder a la persistencia de dicho objeto) dado que, por ejemplo, podemos estar intentando acceder a buscar un juego y a la misma vez poder estar eliminando el mismo.

Manejo de excepciones

- Recepción de datos: Cuando estamos recibiendo datos y se recibe 0 tiramos una excepción ya que no se está recibiendo más datos y no se logró recibir todos los datos que estábamos esperando.
- Pérdida de conexión cliente: Cuando el cliente se desconecta se maneja la excepción del lado del servidor para no matarlo.
- Pérdida de conexión servidor: Cuando el servidor se desconecta se maneja la excepción del lado del cliente para cerrar la aplicación de una manera amigable avisando el error ocurrido.

Descripción y justificación de diseño

En términos de diseño nuestro sistema fue desarrollado en capas, intentando de esta manera desacoplar tanto al cliente del servidor, como al dominio y al protocolo.
Esto nos beneficia dado que si queremos realizar cambios a futuro sólo se verá afectado el código a nivel de la capa requerido.
Además de poder reutilizar el dominio en distintos sistemas, como también el protocolo si el mismo es utilizado con el dominio el cual fue implementado.

Diagrama de paquetes

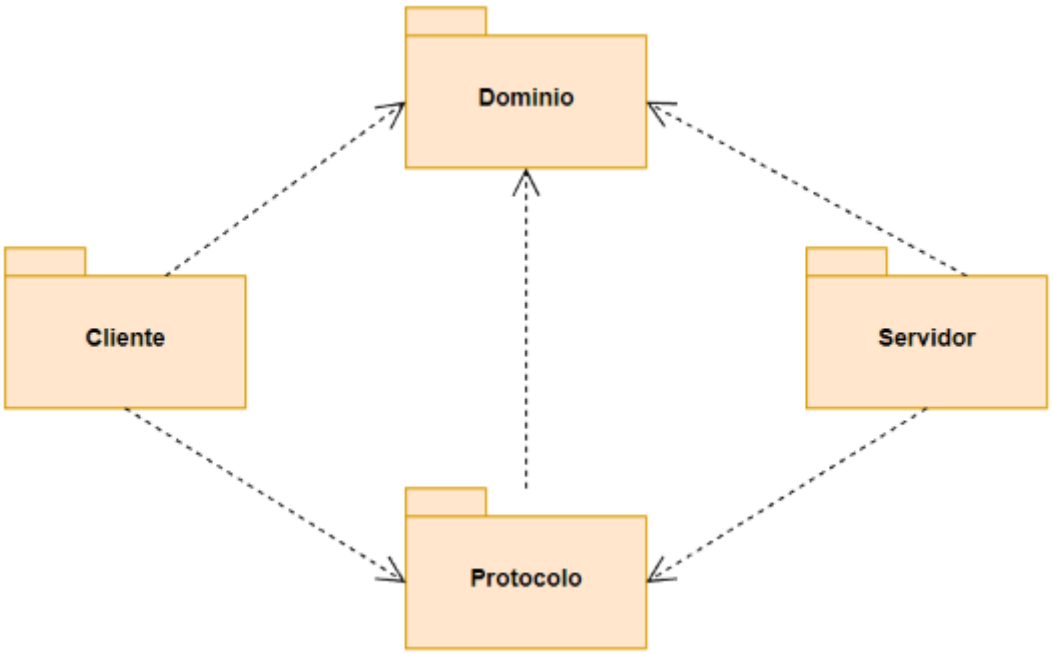


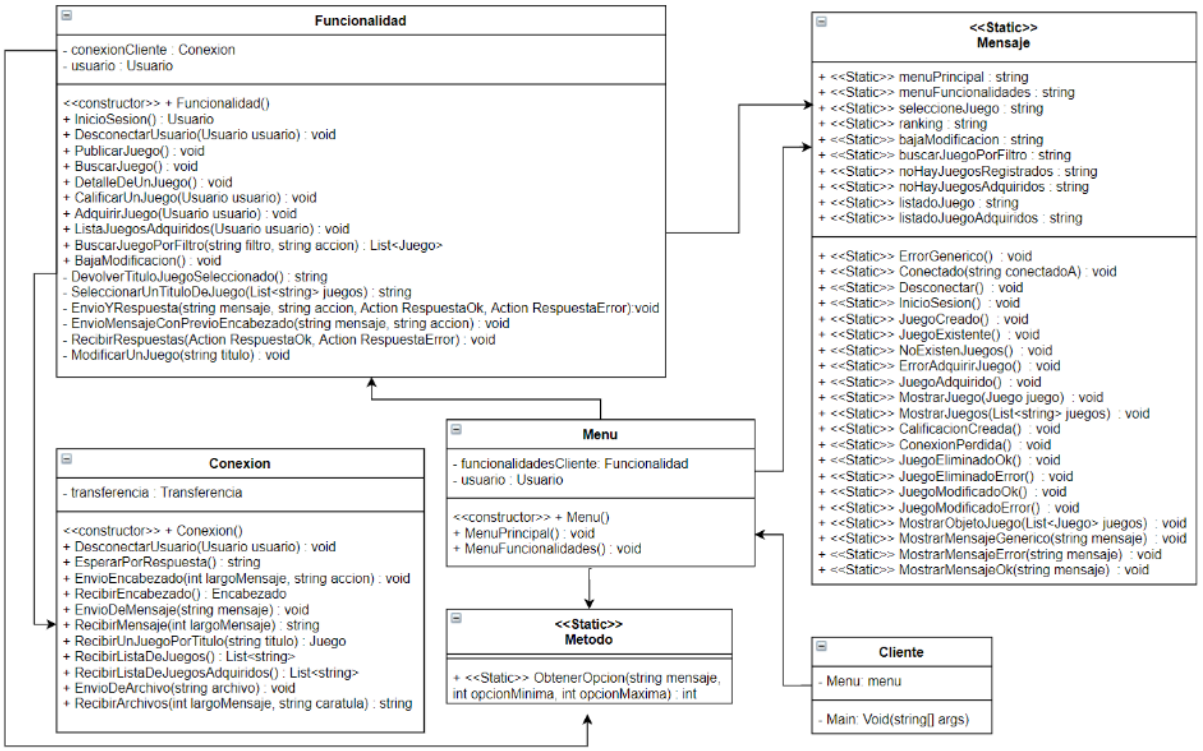
Diagrama de clases

- Cliente

En dicho paquete se encuentran todas las clases correspondientes con el cliente.

Describiendo brevemente cada responsabilidad

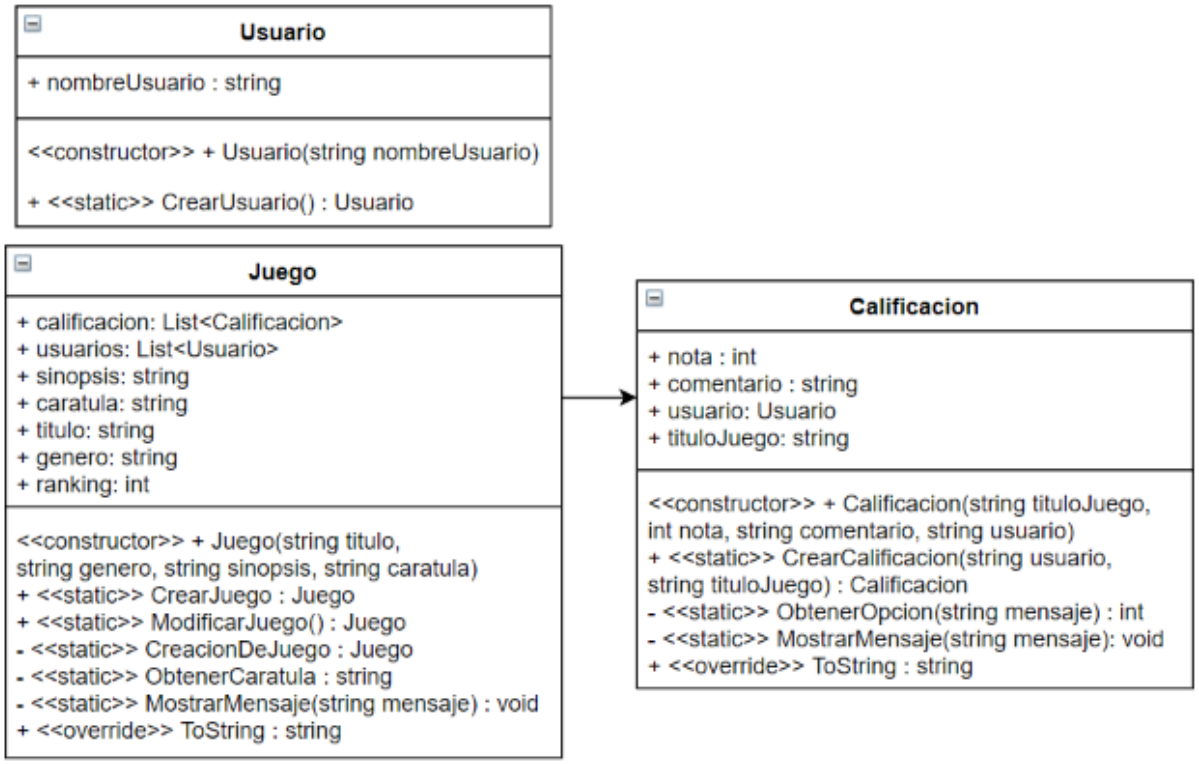
- Clase Cliente: Es la responsable de iniciar la instancia del cliente (nuestro main).
- Clase Menú: Es instanciada por nuestra clase “Cliente” y es la encargada de mostrar los menús con las distintas funcionalidades al usuario.
- Clase Funcionalidad: Es instanciada por el “Menú” y es responsable de ejecutar las distintas funcionalidades con las que cuenta el cliente.
- Clase Conexion: Es instanciada por la clase “Funcionalidad” para llevar a cabo las distintas acciones. Es responsable de comunicarse con el protocolo para realizar las distintas transferencias de datos con el servidor.
- Clase Mensaje: Es utilizada principalmente por las clases “Funcionalidad” y “Menú” y su única responsabilidad es mostrar distintos mensajes por consola.
- Clase Método: Es utilizada principalmente por las clases “Funcionalidad” y “Menú”. Cuenta con métodos que son reutilizados en distintas partes del código de nuestro paquete.



- Dominio

En dicho paquete se encuentran todas las clases correspondientes al dominio del negocio.

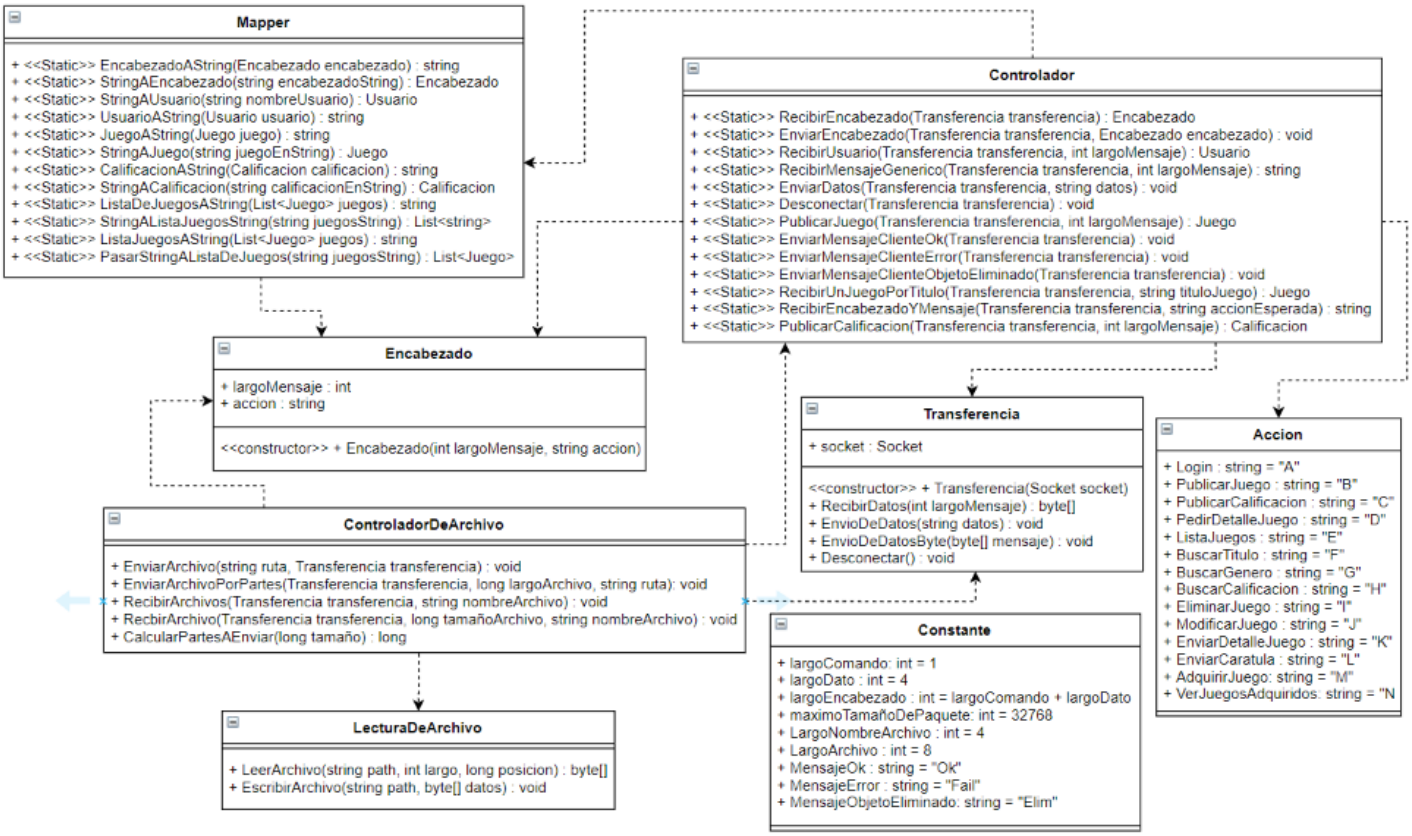
- Clase Usuario: Clase experto. Responsable de pedir y validar los datos para el alta de usuarios.
- Clase Juego: Clase experto. Responsable de pedir y validar los datos para el alta de Juegos.
- Clase Calificación: Clase experto. Responsable de pedir y validar los datos para el alta de calificaciones de un juego.



- Protocolo

En dicho paquete se encuentran todas las clases correspondientes con la transferencia de datos y archivos.

- Clase Mapper: Es la responsable de hacer las distintas conversiones de las entidades a string o viceversa.
- Clase Encabezado: Es la clase cual es utilizada para informar en una comunicación cliente/servidor que acción se va a realizar y el largo del mensaje que se debe recibir.
- Clase Controlador: Controla los envíos y recepciones de datos y delega a la transferencia para que se puedan recibir los datos correctamente.
- Clase Transferencia: Es la encargada del envío y recepción de datos.
- Clase Acción: En esta clase se encuentran definidas todas las acciones que puede realizar el cliente y el servidor sobre de nuestro sistemas.
- Clase Constante: En esta clase definimos el protocolo que usamos en todo el obligatorio para la transferencia de datos.
- Clase ControladorDeArchivo: Esta clase es la que se encarga de realizar todas las operaciones para que luego la clase “LecturaDeArchivo” envíe y reciba los archivos correctamente.
- Clase LecturaDeArchivo: En esta clase se manejan los archivos ya sea para leer el mismo o para escribirlo.

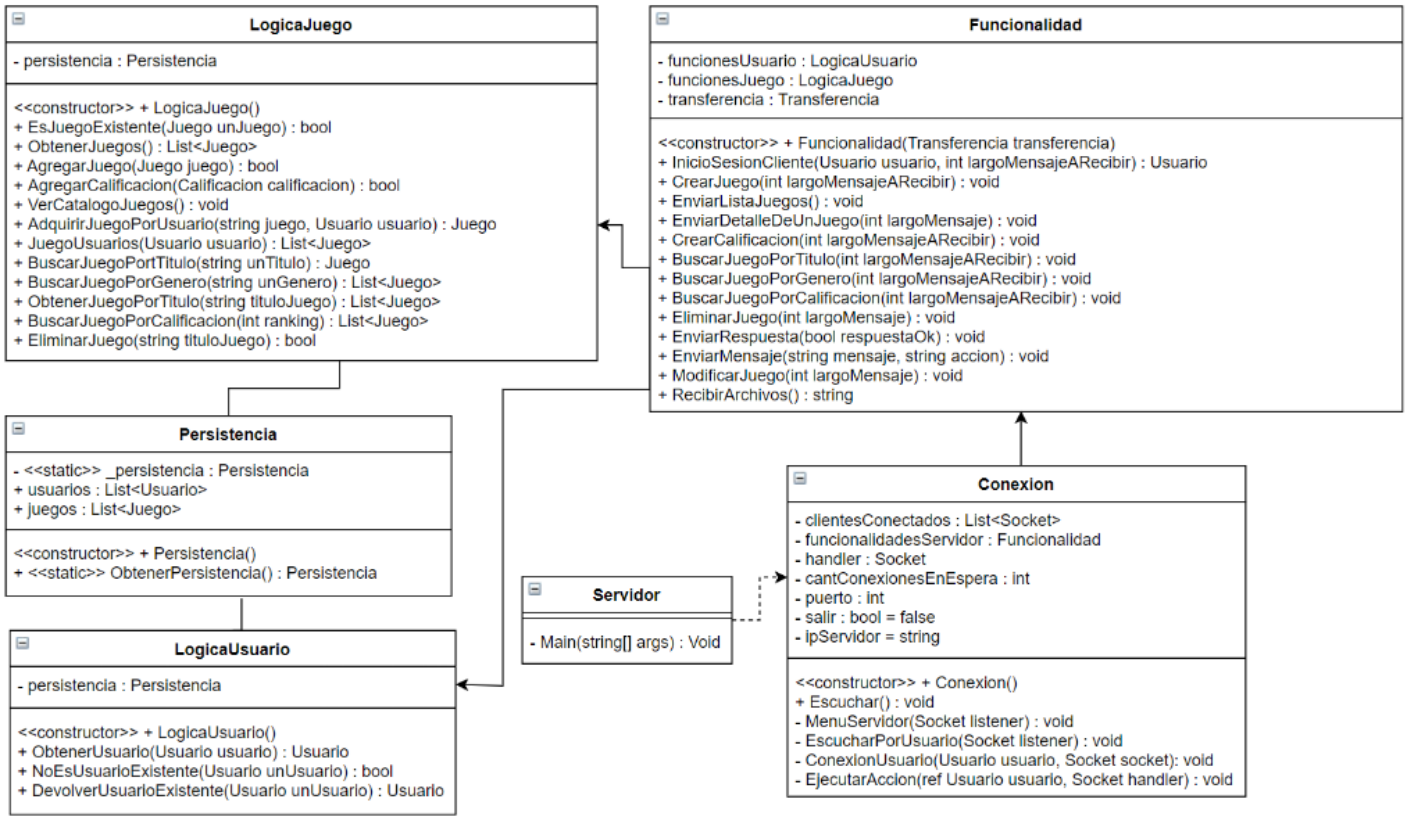


- Servidor

En dicho paquete se encuentran todas las clases correspondientes con el servidor.

Describiendo brevemente cada responsabilidad.

- Clase Funcionalidad: Es responsable de ejecutar las distintas funcionalidades con las que cuenta el servidor.
- Clase LogicaJuegos: Esta clase es instanciada por la clase Funcionalidad para realizar distintas funciones sobre la entidad juego. Además utiliza la instancia de la clase Persistencia para poder guardar los cambios correspondientes.
- Clase LogicaUsuarios: Esta clase es instanciada por la clase Funcionalidad para realizar distintas funciones sobre la entidad usuario. Además utiliza la instancia de la clase Persistencia para poder guardar los cambios correspondientes.
- Clase Servidor: Es la responsable de iniciar la instancia del servidor (nuestro main).
- Clase Conexion: Es responsable de comunicarse con el protocolo para realizar las distintas transferencias de datos con el cliente.
- Clase Persistencia: En esta clase es donde tenemos las listas de juegos y usuarios del sistema.



Protocolo

En el protocolo, el cual es utilizado por el cliente y el servidor, definimos varias reglas que se deben cumplir para el envío de datos entre las terminales.

El mismo fue diseñado únicamente para poder comunicar terminales, en este caso clientes a servidor y viceversa.
Para poder lograr esta comunicación el interesado del envío de datos, el cliente en nuestro caso, deberá enviar un encabezado donde el servidor, que siempre está escuchando por el cliente, lo recibe tomando la acción que recibió y el largo del próximo mensaje que debe de recibir.
Dependiendo de la acción que se le envía al servidor se le debe brindar información al cliente o solo dar una respuesta de si la acción se pudo realizar correctamente o no.
En caso de necesitar enviar datos (no una respuesta Ok o Fail) se realiza el mismo procedimiento detallado anteriormente pero en este caso el interesado en envío de los datos va a ser el servidor. Por tanto se intercambian los roles, el servidor pasa a mandar el encabezado y luego los datos y el cliente a recibir el encabezado y posteriormente la recepción de los datos.

Encabezado

Para llevar a cabo lo mencionado anteriormente, se implementó la clase Encabezado la cual está compuesta por una acción y el largo del próximo mensaje a recibir. Dado esto contamos con una clase “Acción” la cual tiene todas las acciones que pueden realizar tanto el cliente como el servidor, las cuales se especificarán en la sección “Comandos - Acciones”.
Además de la clase Constante para obtener el largo de un encabezado y otro tipo de información la cual se especifica en la sección “Constantes del Protocolo”.

Comandos - Acciones

Las acciones utilizadas en en dicho protocolo están representadas con una letra del abecedario, en este caso van de la letra “A” hasta la “N”.

Cada letra la podemos mapear con una funcionalidad/requerimiento del sistema.

Funcionalidad	Representación protocolo
Login	A
Publicar Juego	B
Publicar Calificación	C
Pedir Detalle De Juego	D
Pedir Lista De Juegos	E
Buscar Por Título	F
Buscar Por Género	G
Buscar Por Calificación	H
Eliminar Un Juego	I
Modificar Un Juego	J
Enviar Detalle De Un Juego	K
Enviar Caratula De Un Juego	L
AdquirirJuego	M
VerJuegosAdquiridos	N

Constantes del Protocolo

Se manejan distintas constantes definidas para la utilización del protocolo.

Nombre de la constante	Valor	Descripción
largoComando	1	Largo de la acción, es 1 dado que nuestras acciones se representan con una letra.
largoDato	4	Largo del dato que se envía en el encabezado. Dado que el dato del encabezado que se envía es el largo del próximo mensaje a enviar lo definimos en 4 el cual podemos representar enteros entre 0 a 9999.
largoEncabezado	largoComando + largoDato + 1	Largo del encabezado, compuesto por el largo del comando + el largo del dato + 1 ya que es el separador entre la acción y el largo del próximo mensaje.
maximoTamañoDePaquete	32768	Máximo tamaño que se puede enviar. El valor no tiene relevancia.
LargoNombreArchivo	4	El largo que puede tener el nombre de un archivo. Definido en 4 el cual podemos representar enteros entre 0 a 9999.
LargoArchivo	8	El largo que puede tener el largo del archivo. Definido en 4 el cual podemos representar el largo en enteros entre 0 a 99999999.
MensajeOk	"Ok"	Mensaje que se envía si la acción fue realizada correctamente.
MensajeError	"FAIL"	Mensaje que se envía si la acción no fue realizada correctamente.
MensajeObjetoEliminado	"Elim"	Mensaje que se envía si se quiere realizar una acción sobre un objeto eliminado previamente.

Mapper

Dado que se requiere (por letra) el envío de datos mediante un string fue necesario implementar un mapper.
El mismo se encarga de mapear una entidad del dominio a string.

Separadores

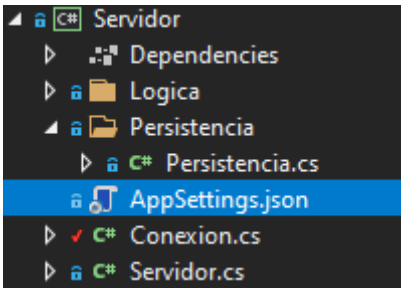
Entidad	Separador	Descripción
Encabezado	“#”	Separa los atributos de la entidad
Juego	“#”	Separa los atributos de la entidad
Calificación	“@”	Separa los atributos de la entidad
Lista de calificaciones	“/”	Separa a cada calificación en la lista
Lista de juegos	“;”	Separa cada juego en la lista

Configuración de la Conexión

Para la configuración se cuenta con archivos “AppSettings.json”.

Servidor

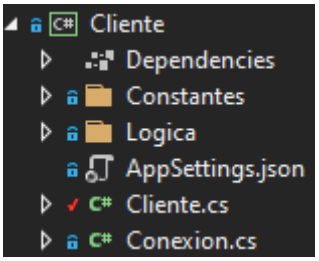
El mismo se puede encontrar dentro de package/carpeta “Servidor”.



Es ahí donde se puede configurar el puerto por el que va a escuchar el servidor, la ip del servidor y el backlog para definir por cuántas conexiones puede responder el servidor.

Cliente

Este otro se puede encontrar dentro de package/carpeta “Servidor”.



Es ahí donde se puede configurar el puerto por el que va a escuchar el cliente, la ip del servidor, el puerto del servidor y la ip del cliente.