Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Programación de Redes

26 de Octubre de 2021

[Repositorio]

Marco Fiorito N° Est.: 227548

Nicolás Praderi N° Est.: 214540

Grupo N6A

Docente: Roberto Assandri

Declaración de autoría

Nosotros, Marco Fiorito y Nicolás Praderi, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos Programación de Redes.
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad.
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra.
- En la obra, hemos acusado recibo de las ayudas recibidas.
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros.
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Alcance de la aplicación

La aplicación realizada comprende todas las funcionalidades solicitadas.
El sistema cuenta con dos aplicaciones, un servidor en el que se maneja la información relacionada a los juegos que pueden obtener los usuarios y archivos relacionados con los mismos. Además, contamos con una aplicación cliente para dicho servidor que se encargará de la interacción de los usuarios con el sistema.
En este documento se presentan todos los detalles de la implementación del sistema.

Modificaciones realizadas

En esta instancia se realizaron modificaciones sobre los Threads al modelo asincrónico de .NET. Para ello se utilizaron las Task que permiten correr tareas de manera asincrónica mediante async y await. Para los retornos de los métodos asincrónicos se modificaron aquellas que fuesen de tipo void por Task y en caso de retornar algún elemento se retornará Task<T> siendo T dicho elemento. Para todos los métodos modificados para ser asincrónicos se modificó su firma como buena práctica.

Dado que en la entrega anterior se utilizaron TCP Listeners, para esta segunda entrega se utilizaron sockets.

Para el acceso a datos y se modificó la metodología de mutua exclusión de locks por semáforos.
Fueron utilizados los comandos WaitAsync y Release del semáforo creado para realizar la mutua exclusión del código.

Por último fueron solucionados los siguientes errores:

- La condición que chequeaba el Rating ingresado por el usuario estaba mal, a último momento cometimos un typo y la condición no era correcta generando que no se pudiera ingresar un rating (esto se comentó en la defensa).
- Del lado del cliente ahora se verifica si el Server está bajo y controlamos la excepción.

Descripción de la arquitectura

A continuación se presenta una descripción de la arquitectura presentada como solución, haciendo énfasis en la infraestructura utilizada, la estructura interna de cada uno de los componentes, y los mecanismos de comunicación entre ellos.

Diseño los componentes.

Diagrama de deploy

1. **Servidor Vapor:** Corresponde al servidor donde se reciben las peticiones de los clientes.
2. **Cliente:** Corresponde al cliente el cual consume los servicios que ofrece el servidor.

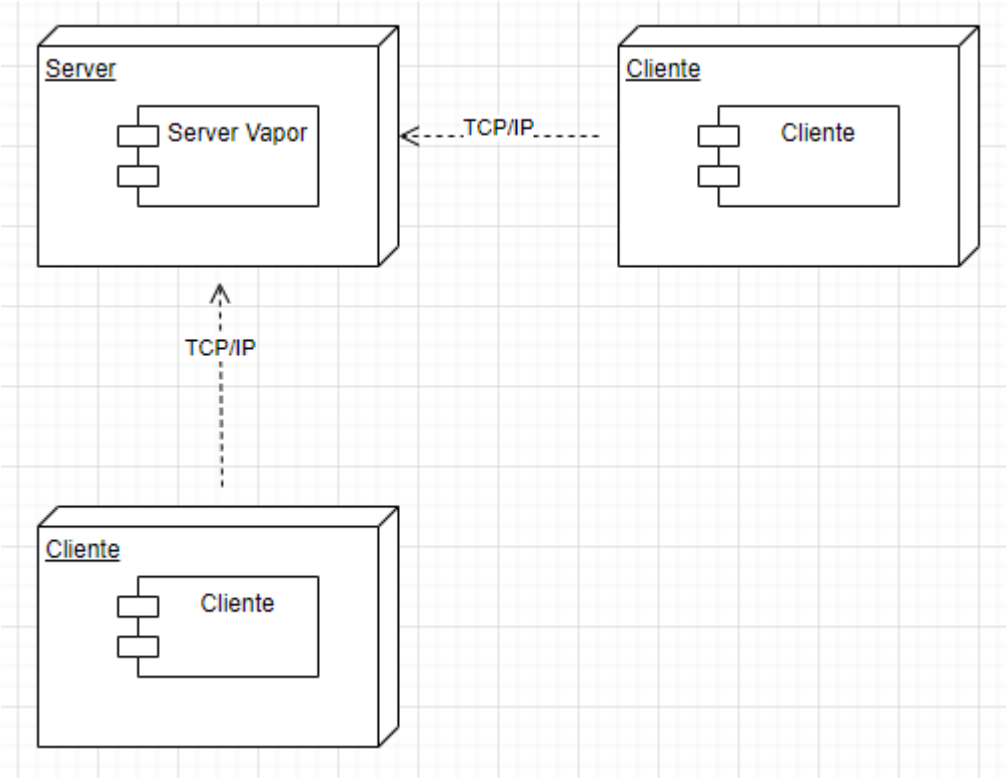
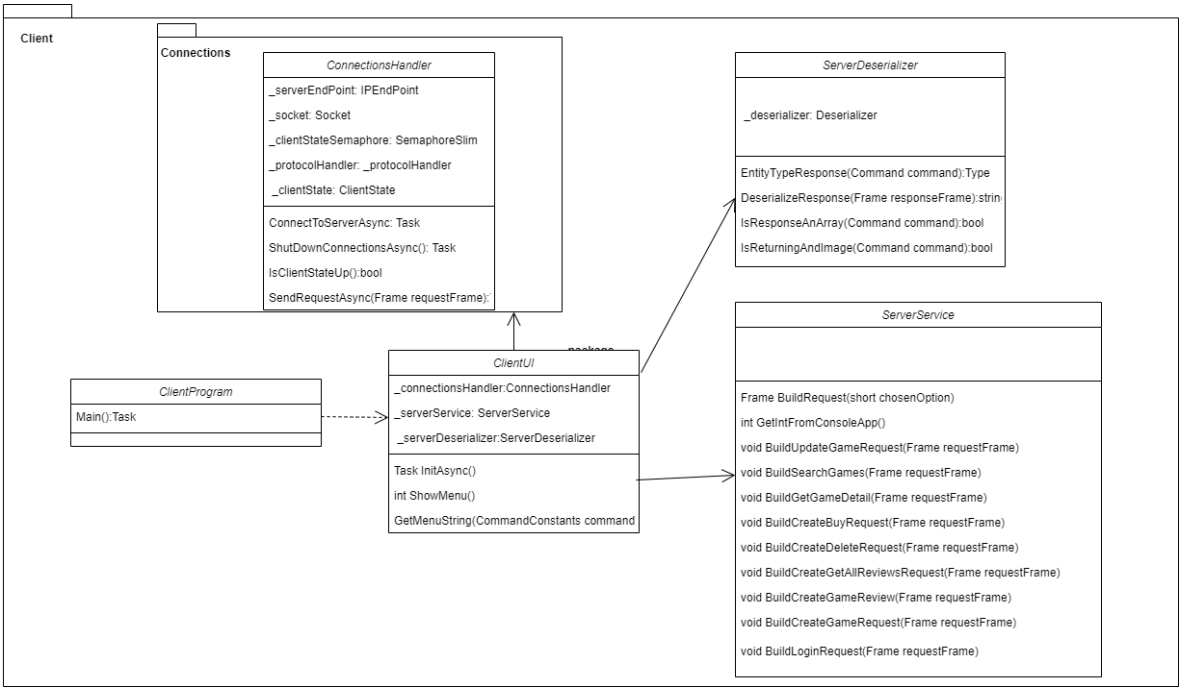
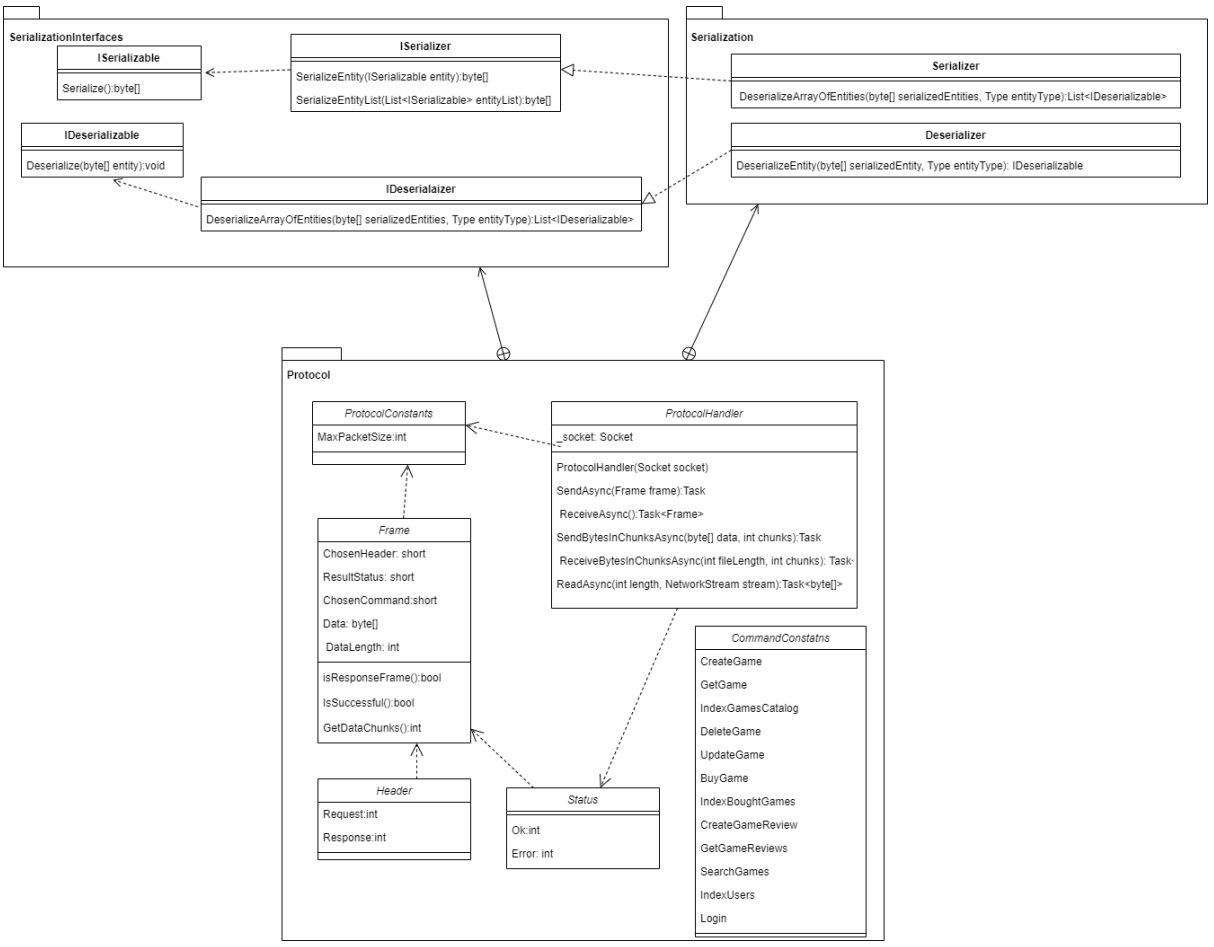


Diagrama de clases:

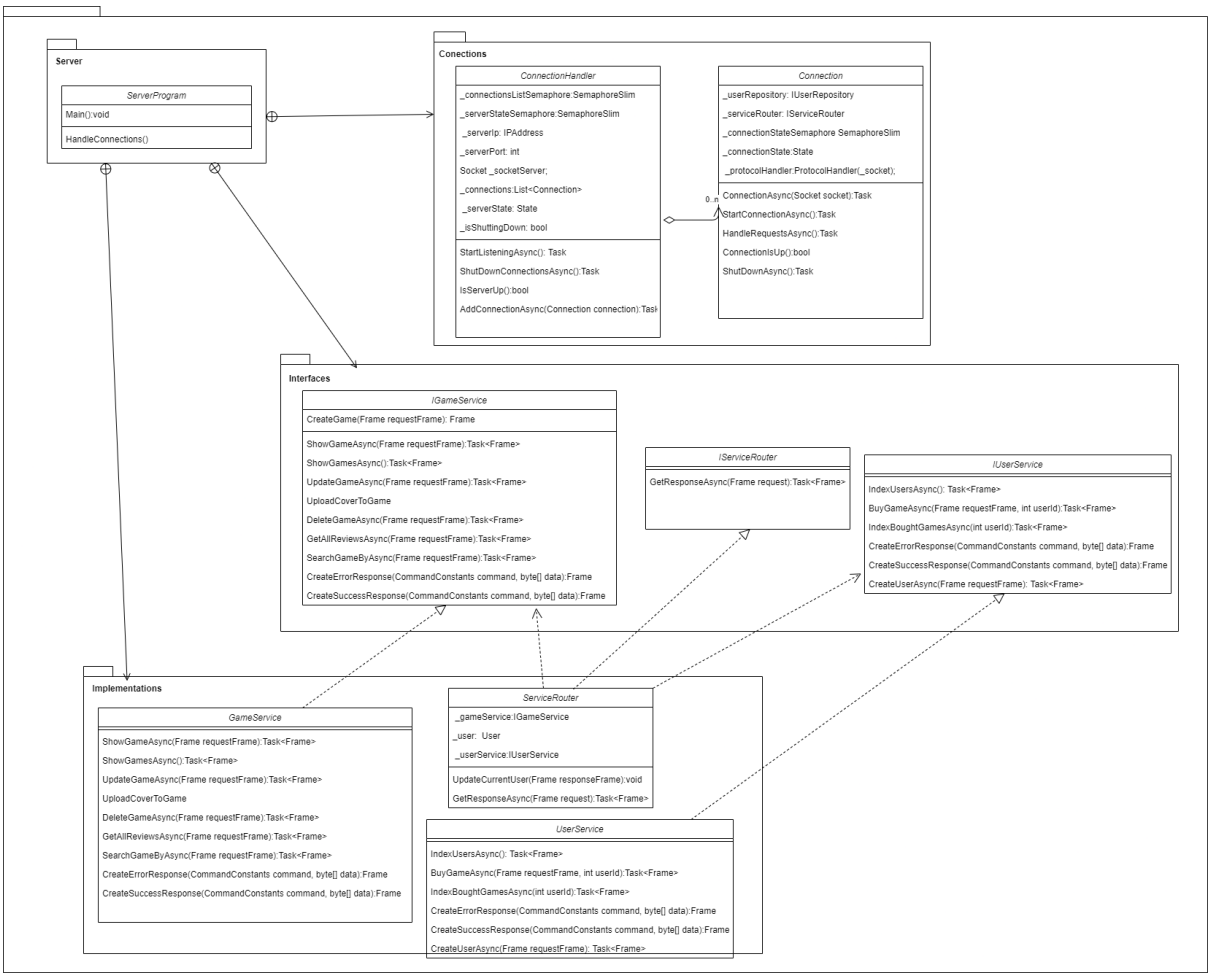
Cliente:

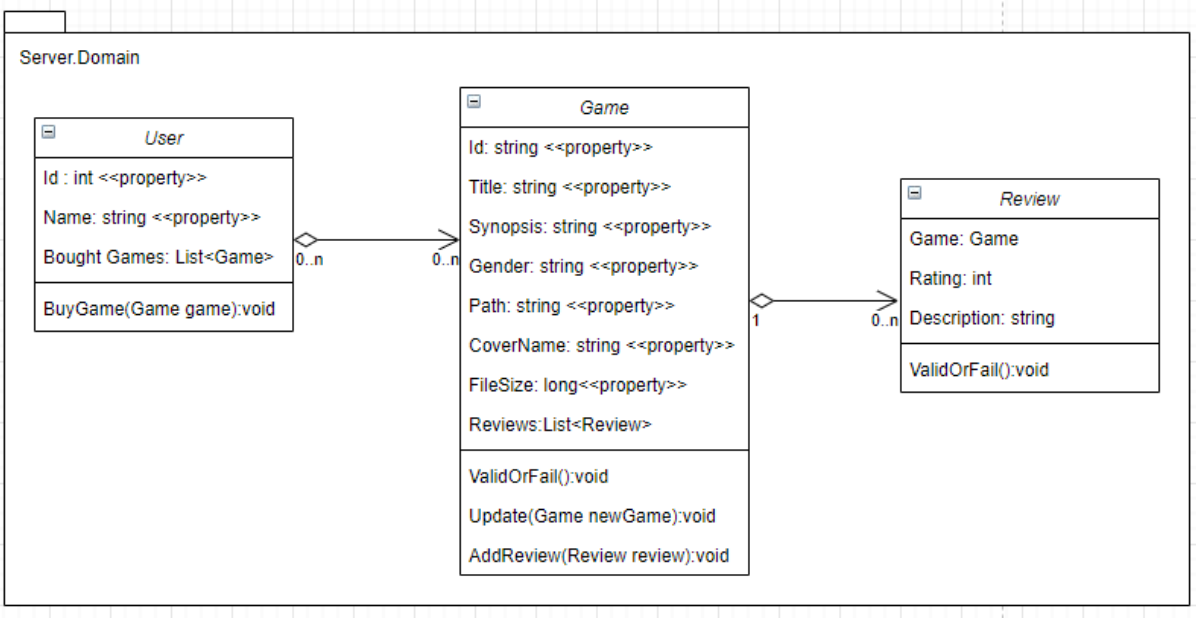


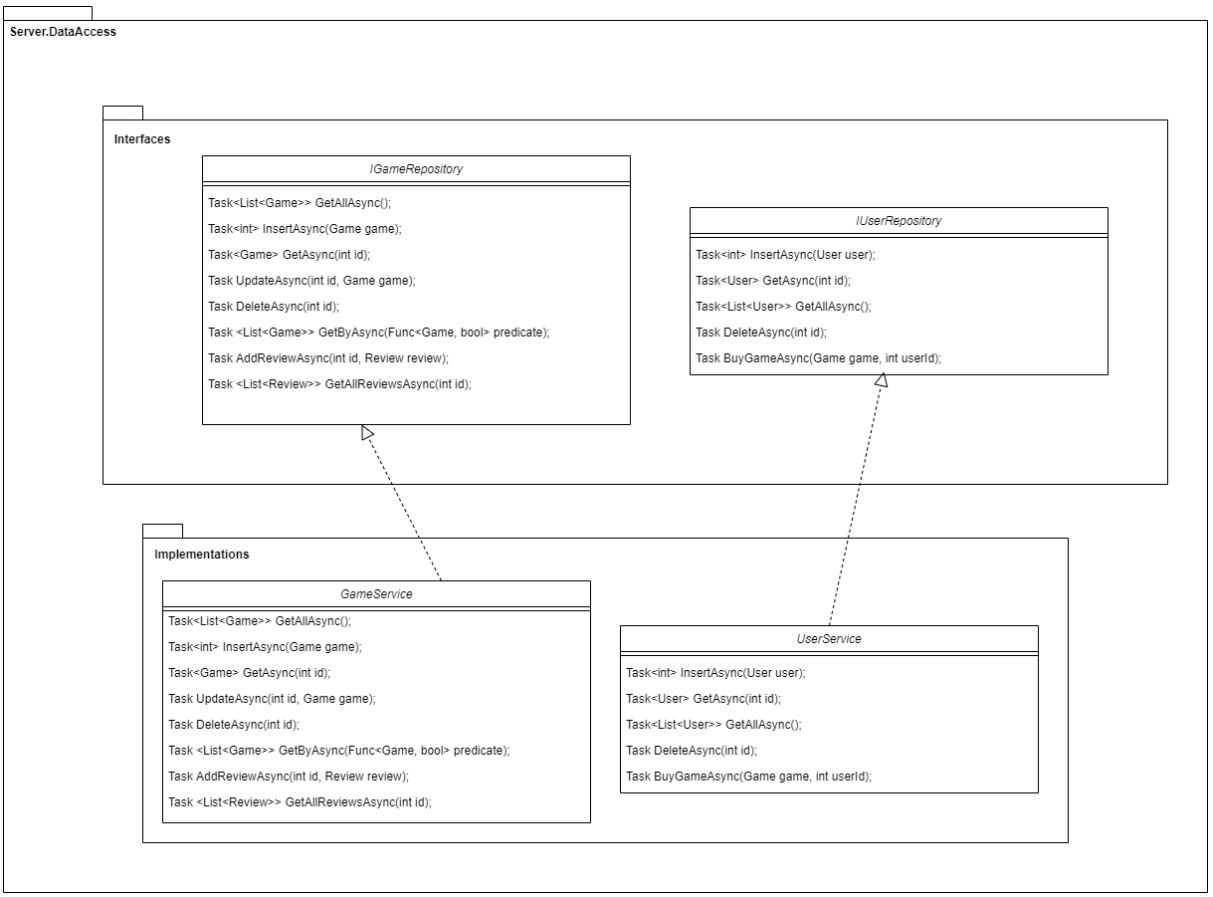
Common:


SerializationInterfaces
ISerializable
Serialize():byte[]
ISerializer
SerializeEntity(ISerializable entity):byte[]
SerializeEntityList(List<ISerializable> entityList):byte[]
IDeserializable
Deserialize(byte[] entity):void
IDeserialaizer
DeserializeArrayOfEntities(byte[] serializedEntities, Type entityType):List<IDeserializable>
Serialization
Serializer
DeserializeArrayOfEntities(byte[] serializedEntities, Type entityType):List<IDeserializable>
Deserializer
DeserializeEntity(byte[] serializedEntity, Type entityType): IDeserializable
Protocol
ProtocolConstants
MaxPacketSize:int
ProtocolHandler
socket: Socket
ProtocolHandler(Socket socket)
SendAsync(Frame frame):Task
ReceiveAsync():Task<Frame>
SendBytesInChunksAsync(byte[] data, int chunks):Task
ReceiveBytesInChunksAsync(int fileLength, int chunks): Task
ReadAsync(int length, NetworkStream stream):Task<byte[]>
Frame
ChosenHeader: short
ResultStatus: short
ChosenCommand:short
Data: byte[]
DataLength: int
isResponseFrame():bool
IsSuccessful():bool
GetDataChunks():int
CommandConstatns
CreateGame
GetGame
IndexGamesCatalog
DeleteGame
UpdateGame
BuyGame
IndexBoughtGames
CreateGameReview
GetGameReviews
SearchGames
IndexUsers
Login
Header
Request:int
Response:int
Status
Ok:int
Error: int

Server:


Server
ServerProgram
Main():void
HandleConnections()
Conections
ConnectionHandler
_connectionsListSemaphore:SemaphoreSlim
_serverStateSemaphore:SemaphoreSlim
_serverIp: IPAddress
_serverPort: int
Socket _socketServer;
_connections:List<Connection>
_serverState: State
_isShuttingDown: bool
StartListeningAsync(): Task
ShutDownConnectionsAsync():Task
IsServerUp():bool
AddConnectionAsync(Connection connection):Task
0..n
Connection
_userRepository: IUserRepository
_serviceRouter: IServiceRouter
_connectionStateSemaphore SemaphoreSlim
_connectionState:State
_protocolHandler:ProtocolHandler(_socket);
ConnectionAsync(Socket socket):Task
StartConnectionAsync():Task
HandleRequestsAsync():Task
ConnectionIsUp():bool
ShutDownAsync():Task
Interfaces
IGameService
CreateGame(Frame requestFrame): Frame
ShowGameAsync(Frame requestFrame):Task<Frame>
ShowGamesAsync():Task<Frame>
UpdateGameAsync(Frame requestFrame):Task<Frame>
UploadCoverToGame
DeleteGameAsync(Frame requestFrame):Task<Frame>
GetAllReviewsAsync(Frame requestFrame):Task<Frame>
SearchGameByAsync(Frame requestFrame):Task<Frame>
CreateErrorResponse(CommandConstants command, byte[] data):Frame
CreateSuccessResponse(CommandConstants command, byte[] data):Frame
IServiceRouter
GetResponseAsync(Frame request):Task<Frame>
IUserService
IndexUsersAsync(): Task<Frame>
BuyGameAsync(Frame requestFrame, int userId):Task<Frame>
IndexBoughtGamesAsync(int userId):Task<Frame>
CreateErrorResponse(CommandConstants command, byte[] data):Frame
CreateSuccessResponse(CommandConstants command, byte[] data):Frame
CreateUserAsync(Frame requestFrame): Task<Frame>
Implementations
GameService
ShowGameAsync(Frame requestFrame):Task<Frame>
ShowGamesAsync():Task<Frame>
UpdateGameAsync(Frame requestFrame):Task<Frame>
UploadCoverToGame
DeleteGameAsync(Frame requestFrame):Task<Frame>
GetAllReviewsAsync(Frame requestFrame):Task<Frame>
SearchGameByAsync(Frame requestFrame):Task<Frame>
CreateErrorResponse(CommandConstants command, byte[] data):Frame
CreateSuccessResponse(CommandConstants command, byte[] data):Frame
ServiceRouter
_gameService:IGameService
_user: User
_userService:IUserService
UpdateCurrentUser(Frame responseFrame):void
GetResponseAsync(Frame request):Task<Frame>
UserService
IndexUsersAsync(): Task<Frame>
BuyGameAsync(Frame requestFrame, int userId):Task<Frame>
IndexBoughtGamesAsync(int userId):Task<Frame>
CreateErrorResponse(CommandConstants command, byte[] data):Frame
CreateSuccessResponse(CommandConstants command, byte[] data):Frame
CreateUserAsync(Frame requestFrame): Task<Frame>


Server.Domain
User
Id : int <<property>>
Name: string <<property>>
Bought Games: List<Game>
BuyGame(Game game):void
0..n
0..n
Game
Id: string <<property>>
Title: string <<property>>
Synopsis: string <<property>>
Gender: string <<property>>
Path: string <<property>>
CoverName: string <<property>>
FileSize: long<<property>>
Reviews:List<Review>
ValidOrFail():void
Update(Game newGame):void
AddReview(Review review):void
1
0..n
Review
Game: Game
Rating: int
Description: string
ValidOrFail():void


Server.DataAccess
Interfaces
IGameRepository
Task<List<Game>> GetAllAsync();
Task<int> InsertAsync(Game game);
Task<Game> GetAsync(int id);
Task UpdateAsync(int id, Game game);
Task DeleteAsync(int id);
Task <List<Game>> GetByAsync(Func<Game, bool> predicate);
Task AddReviewAsync(int id, Review review);
Task <List<Review>> GetAllReviewsAsync(int id);
IUserRepository
Task<int> InsertAsync(User user);
Task<User> GetAsync(int id);
Task<List<User>> GetAllAsync();
Task DeleteAsync(int id);
Task BuyGameAsync(Game game, int userId);
Implementations
GameService
Task<List<Game>> GetAllAsync();
Task<int> InsertAsync(Game game);
Task<Game> GetAsync(int id);
Task UpdateAsync(int id, Game game);
Task DeleteAsync(int id);
Task <List<Game>> GetByAsync(Func<Game, bool> predicate);
Task AddReviewAsync(int id, Review review);
Task <List<Review>> GetAllReviewsAsync(int id);
UserService
Task<int> InsertAsync(User user);
Task<User> GetAsync(int id);
Task<List<User>> GetAllAsync();
Task DeleteAsync(int id);
Task BuyGameAsync(Game game, int userId);

Mecanismos de comunicación de los componentes

Protocolo

El protocolo utilizado para la comunicación entre el cliente y servidor es el siguiente:

Nombre del campo	Header	Command	DataLength	Data(optional)	Result
Valor	RES/REQ	0 - 11	Int	Variable	OK/ERROR
Largo	2	2	4	Variable	2

Se utilizaron los siguientes comandos para comunicar según la funcionalidad requerida:

Comando	**Valor**
Disconnect from server	0
Create a game	1
Get game information	2
Get all games	3
Delete a game	4
Update game info	5
Buy a game	6
Show my bought games	7
Give a review	8
Get game reviews	9
Search games by title, gender or rating	10
Get server users	11

En caso de que el usuario no esté autenticado los comandos son:

Comando	Valor
Disconnect from server	0

Login	1

Cliente

De parte del cliente al seleccionar uno de los comandos se le solicita al usuario en caso de que sea necesario todos los datos para construir la Request utilizando el ServerService. Este servicio utiliza los DTOs definidos bajo el proyecto DTO -> carpeta Request, los serializa y los asigna al Frame que se va a enviar mediante la Request utilizando la clase ConnectionHandler.
Luego para recibir una respuesta se utiliza el método Receive de la clase ProtocolHandler y por último se utiliza el ServerDeserializer para deserializar la respuesta, dependiendo del EntityType que sea. En el método DeserializeResponse del ServerDeserializer se chequea primero que nada si la respuesta es success (si no, se imprime el error devuelto en un ErrorDTO), luego dependiendo de si es un Array o no se se genera el string a imprimir de forma distinta. Además, en el caso puntual que se solicite un Game por si solo (en este caso se descarga la imágen también) por lo que se realiza una lógica adicional aquí.

Servidor

De parte del servidor se utiliza la clase ConnectionHandler para manejar el estado del servidor y se utiliza la clase Connection para manejar las conexiones entrantes.
Como primer punto a comentar como no se especificaba nada sobre funcionalidades de login/logout decidimos al iniciar crear un usuario utilizando el UserRepository para luego tener la funcionalidad de buy asociada a un usuario.
Luego a la hora de recibir request se utiliza la clase ProtocolHandler para recibir el Frame y mediante la utilización de una clase ServiceRouter se realiza el pasamano entre la Connection y los servicios particulares que terminan realizando las llamadas al repositorio en sí el cuál maneja toda la interacción con los datos.
Por último el método GetResponse del ServiceRoute nos devuelve el Frame response serializado el cual es enviado utilizando el método Send del ProtocolHandler.

.

Justificación de las decisiones importantes de diseño e implementación

A continuación presentamos las decisiones estructurales más importantes tomadas. También aprovechamos y explicamos un poco el funcionamiento de algunos mecanismos de nuestro obligatorio:

- En la aplicación utilizamos threads dentro de la clase ConnectionHandler del proyecto Server para la gestión de múltiples clientes.
  Esta clase se encarga de escuchar activamente por nuevas solicitudes de conexión.
  Los pasos que se siguen son:
  - Comienza el server y la clase Main lanza un hilo en el que crea el ConnectionHandler.
  - Dentro del ConnectionHandler se lanza un hilo donde escucha por nuevas conexiones.
  - Este hilo que escucha por nuevas conexiones se mantiene en bucle activo donde cada vez que se recibe una solicitud de conexión se acepta la misma, se almacena y se lanza otro hilo independiente con la clase Connection que maneja las Requests.
- La conexión se cierra de parte del cliente utilizando el comando 0 y del lazo del servidor se hace shutdown utilizando cualquier tecla.
- Se utilizó como separador el caracter # para serializar y deserializar los datos durante la comunicación.
- Se definieron DTOs de request y response, cada uno de ellos sabe cómo serializarse y deserializarse.
- En el cliente al recibir una response es utilizado el ServerDeserializaer el cual tiene un método DeserializeResponse, este funciona de la siguiente forma:
  - Recibe la respuesta
  - Dependiendo del comando obtiene cuál es el tipo de la respuesta utilizando el método EntityTypeResponse.
  - Utilizando la clase Deserializer bajo Common deserializa la lista de entidades que responde el server o una entidad sola dependiendo del caso que sea.
  - Utilizando el método IsReturningAndImage verificamos si dado el comando la responde debe devolver una imágen, si este es el caso escribo el archivo a disco.
  - El deserializar a su vez utiliza la clase Activator para dado un entityType crear una instancia de esa entidad y deserializarla, sabemos que la entidad recibida va a implementar el método deserialize dado que implementan la interfaz IDeserializable.
- Para mantener el login sencillo, el usuario se identifica por el Name.

Datos de prueba

En esta instancia se mostrará evidencia en base a una serie de datos de prueba realizados.
Al conectar la máquina cliente se despliega la pantalla de para hacer **Login**.

Login:

```
Connected to server.


Choose an option:
0 - Disconnect from server
1- Login
```

Se ingresa un dato vacio

```
Choose an option:
0 - Disconnect from server
1- Login

Invalid option, please enter a new one
```

Se logra el login con exito

```
Connected to server.


Choose an option:
0 - Disconnect from server
1- Login
1

Indicate the user name:
Marco

Id: 1, Name: Marco
```

Se realiza un segundo login con exito

```
Choose an option:
0 - Disconnect from server
1- Login
1

Indicate the user name:
Nicolas

Id: 2, Name: Nicolas
```

Del lado del servidor se visualizan los 2 clientes conectados

```
Starting...
Write any key to shutdown the server
Client accepted
Client accepted
```

Se despliega el menu principal

```
Connected to server.


Choose an option:
0 - Disconnect from server
1 - Create a game
2 - Get game information
3 - Get all games
4 - Delete a game
5 - Update game info
6 - Buy a game
7 - Show my bought games
8 - Give a review
9 - Get game reviews
10 - Search games by title, gender or rating
11 - Get server users
```

1 - Create a game

Se crea un juego correctamente:

```
Indicate the new game title:
Doom
Indicate the new game synopsis:
Shooter en primera persona
Indicate the new game gender:
Shooter
Indicate the full path of the cover:
C:\Doom.png

Id: 1, Title Doom
```

Se crea un juego incorrectamente:

```
Indicate the new game title:

Indicate the new game synopsis:

Indicate the new game gender:

Indicate the full path of the cover:

Please indicate a path that exists

Please indicate a path that exists
C:\Doom.png

Error: Game must have a title
```

Se crea un juego repetido:

```
Indicate the new game title:
Doom
Indicate the new game synopsis:

Indicate the new game gender:

Indicate the full path of the cover:

Please indicate a path that exists

Please indicate a path that exists

Please indicate a path that exists
C:\Doom.png

Error: Game name need to be unique
```

Se crea un juego sin sinopsis

```
Indicate the new game title:
God of War
Indicate the new game synopsis:

Indicate the new game gender:

Indicate the full path of the cover:

Please indicate a path that exists

Please indicate a path that exists
C:\Doom.png

Error: Game must have a Synopsis
```

Se crea un juego sin genero

```
Indicate the new game title:
God of War
Indicate the new game synopsis:
God of Sparta killing every god
Indicate the new game gender:

Indicate the full path of the cover:

Please indicate a path that exists

Please indicate a path that exists
C:\Doom.png

Error: Game must have a gender
```

Se crea correctamente segundo juego

```
Indicate the new game title:
God of War
Indicate the new game synopsis:
God of Sparta killing every god
Indicate the new game gender:
Adventure
Indicate the full path of the cover:
C:\Gow.jpg

Id: 2, Title God of War
```

2 - Get game information

Se lista el juego por Id

```
2

Indicate the game id to show:
2

Id: 2
        Title: God of War
        Synopsis: God of Sparta killing every god
        Gender: Adventure
        Download Cover Path: C:\Users\2innovateIT\Desktop\pr-fiorito-praderi-main\Obligatorio\Client\bin\Debug\netcoreapp3.1\Gow.jpg
        Average Rating: 0
```

3 - Get all games

Se listan todos juegos

```
Id: 1
        Title: Doom
        Synopsis: Shooter en primera persona
        Gender: Shooter
        Cover: C:\Users\2innovateIT\Desktop\pr-fiorito-praderi-main\Obligatorio\Server\bin\Debug\netcoreapp3.1\Doom.png
Id: 2
        Title: God of War
        Synopsis: God of Sparta killing every god
        Gender: Adventure
        Cover: C:\Users\2innovateIT\Desktop\pr-fiorito-praderi-main\Obligatorio\Server\bin\Debug\netcoreapp3.1\Gow.jpg
```

Se indica un id que no existe

```
Indicate the game id to show:
3

Error: Game not exist
```

4 - Delete game

Se inserta un id de un juego inexistente

```
4

Indicate the game id to delete:
3

Error: Game not exist
```

Se borra correctamente

```
4

Indicate the game id to delete:
2

Game deleted!
```

5 - Update game info

Se actualiza un juego mediante un id que no existe

```
Indicate the Id of the game to update
2
Indicate the new name for the game
Bob construye
Indicate the new synopsis for the game
Juego entretenido para niños
Indicate the new gender for the game
Infantil

Error: Game not exist
```

Se inserta una letra en la solicitud de Id

```
5

Indicate the Id of the game to update
Doom 2
Please enter a number:
```

Se actualiza un juego por id correcto pero sin nombre

```
5

Indicate the Id of the game to update
1
Indicate the new name for the game

Indicate the new synopsis for the game

Indicate the new gender for the game


Error: Game must have a title
```

Se actualiza un juego por id correcto pero sin synopsis

```
5

Indicate the Id of the game to update
1
Indicate the new name for the game
Doom 2
Indicate the new synopsis for the game

Indicate the new gender for the game


Error: Game must have a Synopsis
```

Se actualiza un juego por id correcto pero sin género

```
5

Indicate the Id of the game to update
1
Indicate the new name for the game
Doom 2
Indicate the new synopsis for the game
Secuela del famoso juego de shooter
Indicate the new gender for the game


Error: Game must have a gender
```

Se actualiza el juego correctamente

```
5

Indicate the Id of the game to update
1
Indicate the new name for the game
Doom 2
Indicate the new synopsis for the game
Secuela del famoso juego de shooter
Indicate the new gender for the game
Shooter

Id: 1, Title Doom 2
```

6 - Buy a game

Se deja en blanco y se inserta una letra en la solicitud

```
Indicate the game id to buy:

Please enter a number:
a
Please enter a number:
```

Se compra un juego correctamente

```
Indicate the game id to buy:

Please enter a number:
a
Please enter a number:
1

Game bought!
```

7- Show my bought games

Se muestran los juegos comprados

```
7


Id: 1
        Title: Doom 2
        Synopsis: Secuela del famoso juego de shooter
        Gender: Shooter
        Cover: C:\Users\2innovateIT\Desktop\pr-fiorito-praderi-main\Obligatorio\Server\bin\Debug\netcoreapp3.1\Doom.png
```

8 - Give a review

Se inserta un id vacio

```
8

Indicate the game id to review:

Please enter a number:
```

Se inserta un review correctamente

```
Indicate the game id to review:
1
Indicate the rating between the range 1-5:
5
Indicate a brief description about your experience
Muy copado

Review added!
```

9 - Get game reviews

Se indica un id de juego que no existe

```
9

Indicate the game id to see the reviews:
4

Error: Game not exist
```

Se indica un id correcto y se visualiza el review

```
9

Indicate the game id to see the reviews:
1

Review:
        Rating: 5
        Description: Muy copado
```

10 - Search games by title, gender or rating

Se busca un juego por titulo con exito

```
1
Indicate the title to search:
Doom

Id: 1
        Title: Doom 2
        Synopsis: Secuela del shooter mas famoso
        Gender: Shooter
        Cover: C:\Users\2innovateIT\Desktop\pr-fiorito-praderi-main\Obligatorio\Server\bin\Debug\netcoreapp3.1\Doom.png
```

Se busca un juego por genero con exito

```
Indicate the search metric you want to use:
1- Title
2- Gender
3 - Rating
2
Indicate the gender to search:
Shooter

Id: 1
        Title: Doom 2
        Synopsis: Secuela del shooter mas famoso
        Gender: Shooter
        Cover: C:\Users\2innovateIT\Desktop\pr-fiorito-praderi-main\Obligatorio\Server\bin\Debug\netcoreapp3.1\Doom.png
```

Se busca un juego por rating con exito

```
Indicate the Rating to search:
5

Id: 1
        Title: Doom 2
        Synopsis: Secuela del shooter mas famoso
        Gender: Shooter
        Cover: C:\Users\2innovateIT\Desktop\pr-fiorito-praderi-main\Obligatorio\Server\bin\Debug\netcoreapp3.1\Doom.png
```

11 - Get server users

Se muestran los multiples clientes conectados

```
11 - Get server users
11


Id: 1, Name: Marco
Id: 2, Name: Nicolas
```

0 - Disconnect from server

```
Starting...
Write any key to shutdown the server
Client accepted
Client accepted
Client has disconnected
Client has disconnected
```

Aclaraciones para ejecutar la aplicación

Para ejecutar la aplicación deberemos tener las siguientes consideraciones:

- Por defecto los App.config vienen configurados utilizando la ServerIP 127.0.0.1, ServerPort 6000 y ClientIP 127.0.0.1. En caso de querer cambiarlo deberán abrir el App.config de cada proyecto y configurarlo.
- Primero se debe ejecutar el servidor y luego cada cliente que se quiera conectar con la aplicación servidor.
- Los proyectos bajo la carpeta Common y bajo la carpeta Server no quedaron del todo bien estructurados (por un tema de tiempo no lo pudimos mejorar) por lo que en dos ocasiones nos sucedió un error de compilación dado que el compilado del proyecto DTO y Exceptions quedaba bajo el proyecto Protocol y lo mismo ocurre con el compilado de los proyectos Server.Domain y Server.DataAccess bajo el proyecto Server generando una ambigüedad en algunos imports.
  Este error simplemente se soluciona yendo al proyecto Protocol y Server para excluir las carpetas DTO, Exceptions, Server.Domain y Server.DataAccess.
  Para el siguiente obligatorio planeamos resolverlo.
- Las aplicaciones compiladas en Release fueron generadas desde Mac y por lo que vimos no generó un .exe. Para correr la aplicación se deberá hacer mediante el comando dotnet utilizando las siguientes dlls:
  - Server.dll dentro de la carpeta Server.
  - Client.dll dentro de la carpeta Client.