Universidad ORT

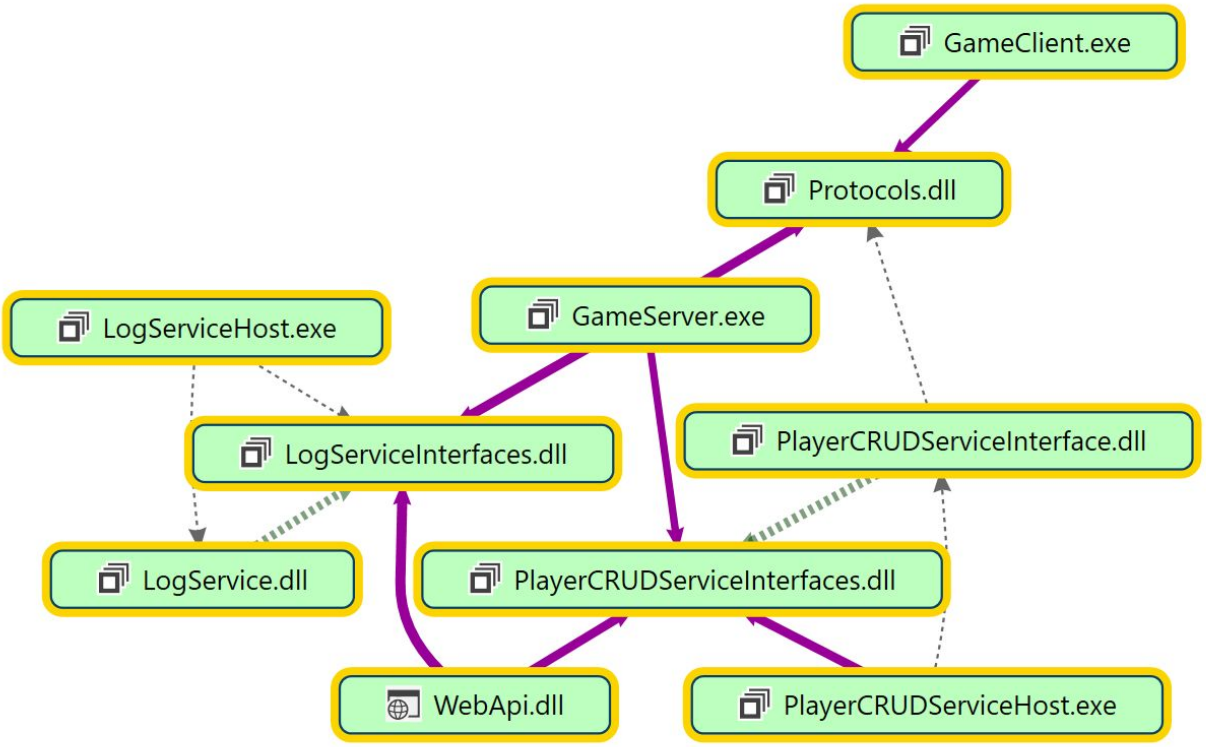
Obligatorio 2

Programación de Redes

Juan Andrés Vico (nro. 200135)
Sebastián Caraballo (nro. 200562)
15-11-2018

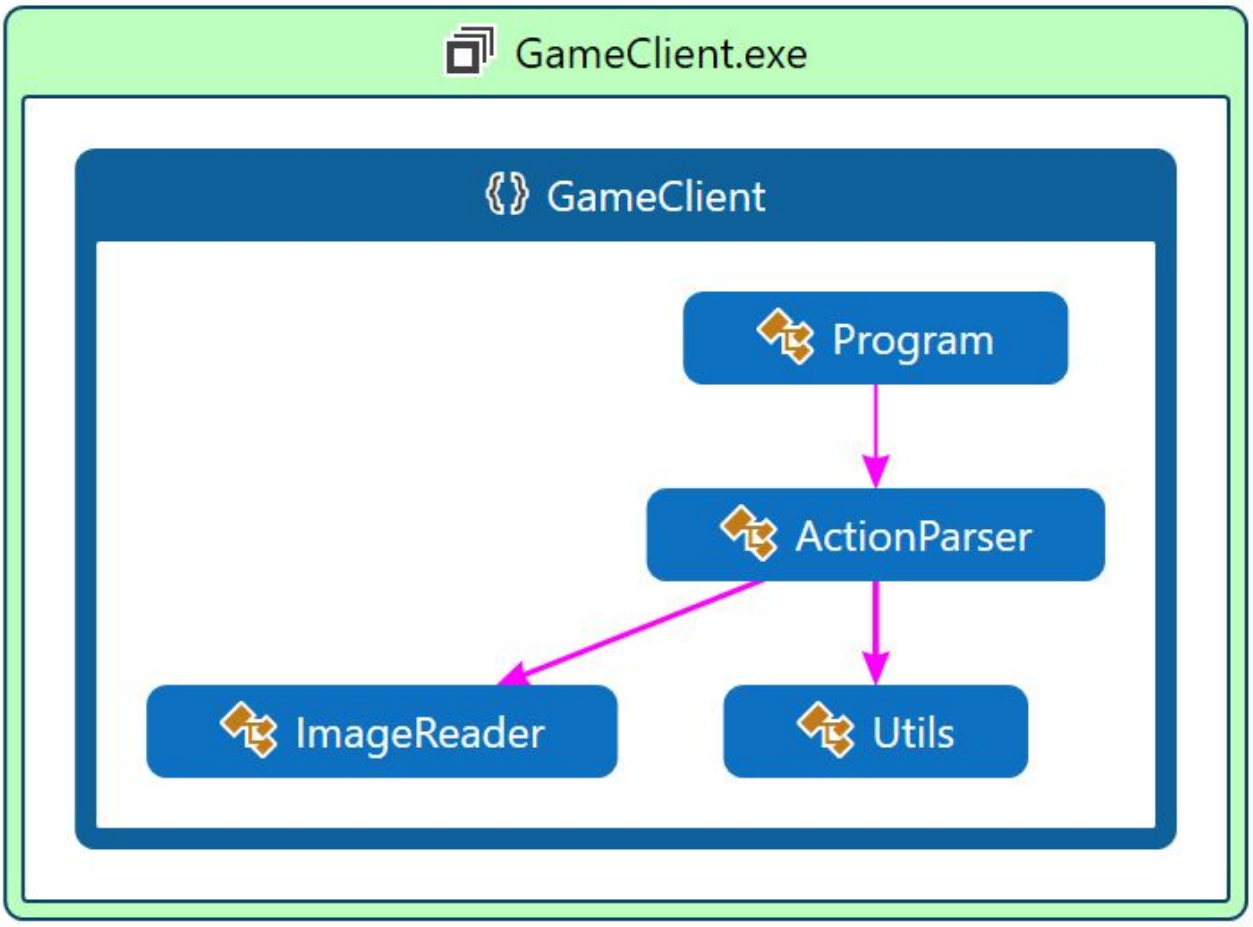
Contenido

Descripción del Diseño del Sistema



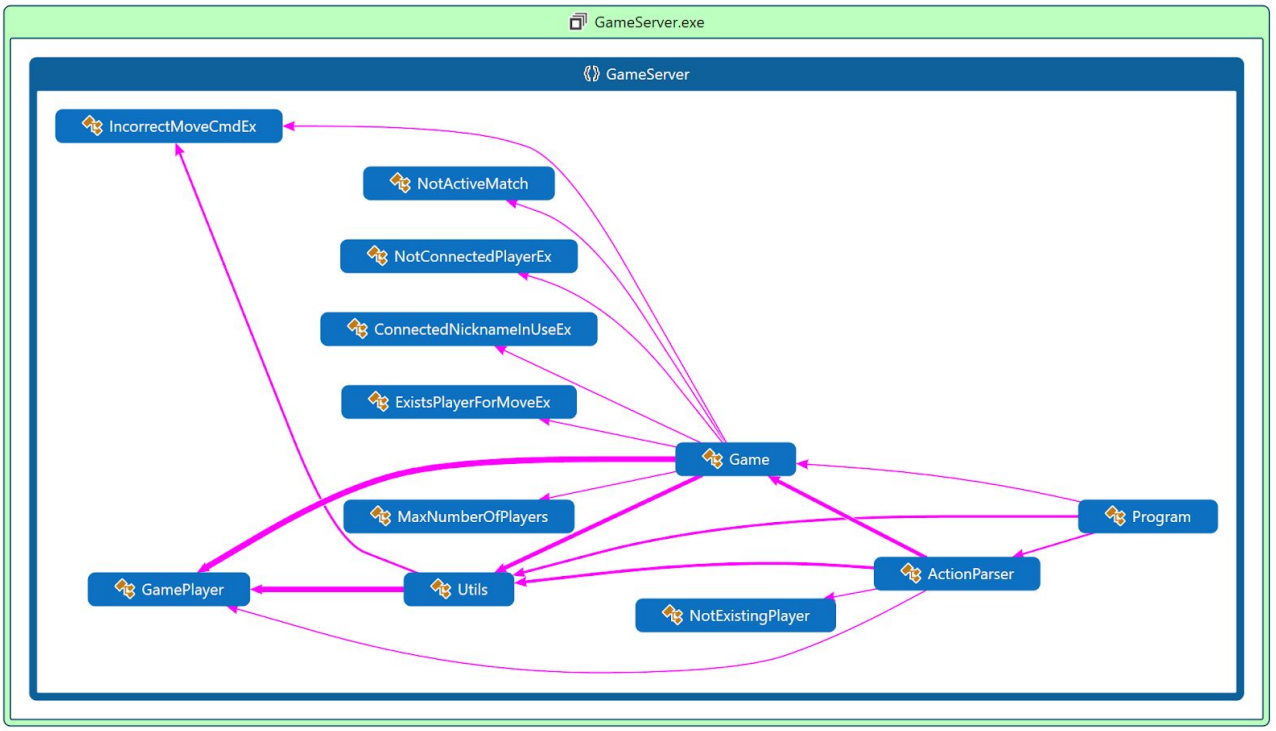
La solución quedó conformada por un total de diez proyectos. Para cumplir con los requerimientos de esta entrega es que se reestructuraron los proyectos para poder así mantener el código extensible. Para realizar el ABM de los jugadores, se utilizó Remoting para mantener en un único lugar el repositorio de los jugadores con las acciones CRUD (Create, Read, Update and Delete). Para realizar las funcionalidades de log y estadísticas se creó otro servidor, también utilizando Remoting, en donde se guarda la información de las partidas jugadas y los logs correspondientes. La principal ventaja de haber separado estas funcionalidades en dos servidores es la facilidad de conectarse a estos mediante un cliente. Notar como tanto la `WebApi`, como el `GameServer` ya existente consultan los datos a estos servidores. A continuación se incluyen los diagramas con las clases presentes en cada uno de los proyectos.

GameClient



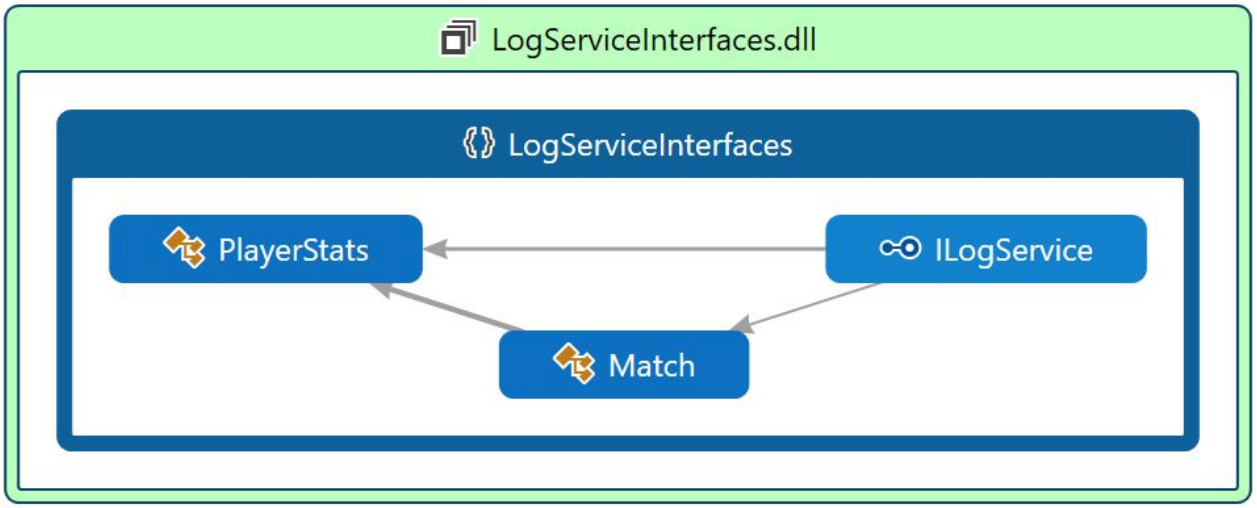
`GameClient` se conforma por cuatro clases: `Program`, `ActionParser`, `Utils` e `ImageReader`. En definitiva, `ActionParser` se encarga de interpretar los comandos del cliente y de realizar las llamadas correspondientes al servidor del juego (`GameServer`). En la clase `Utils` hay diversos métodos relacionados a mensajes que se muestran al jugador, entre otros. La clase `ImageReader` se encarga de validar que el archivo de imagen que el usuario elige exista en definitiva para luego ser codificado en Base64 y ser enviado al servidor.

GameServer



`GameServer` es el encargado de mantener la lógica del juego Slasher (contenida en la clase `Game`), además de mantener informados a los jugadores del estado del juego. A su vez, es posible realizar el registro de jugadores, por lo que `GameServer` actúa como cliente del `PlayerCRUDService` (mediante Remoting).

LogServiceInterfaces

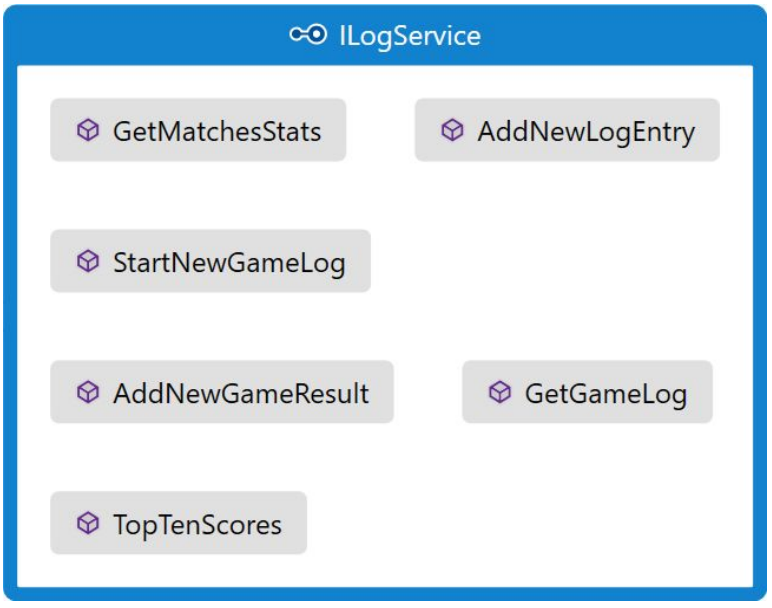


`LogServiceInterfaces` contiene a la interfaz `ILogService`, en donde se especifica los métodos necesarios para cubrir tres de las cuatros nuevas funcionalides requeridas para este obligatorio: Ver un log de la última partida, Ver una lista con los diez puntajes más altos y Ver estadísticas de los jugadores.

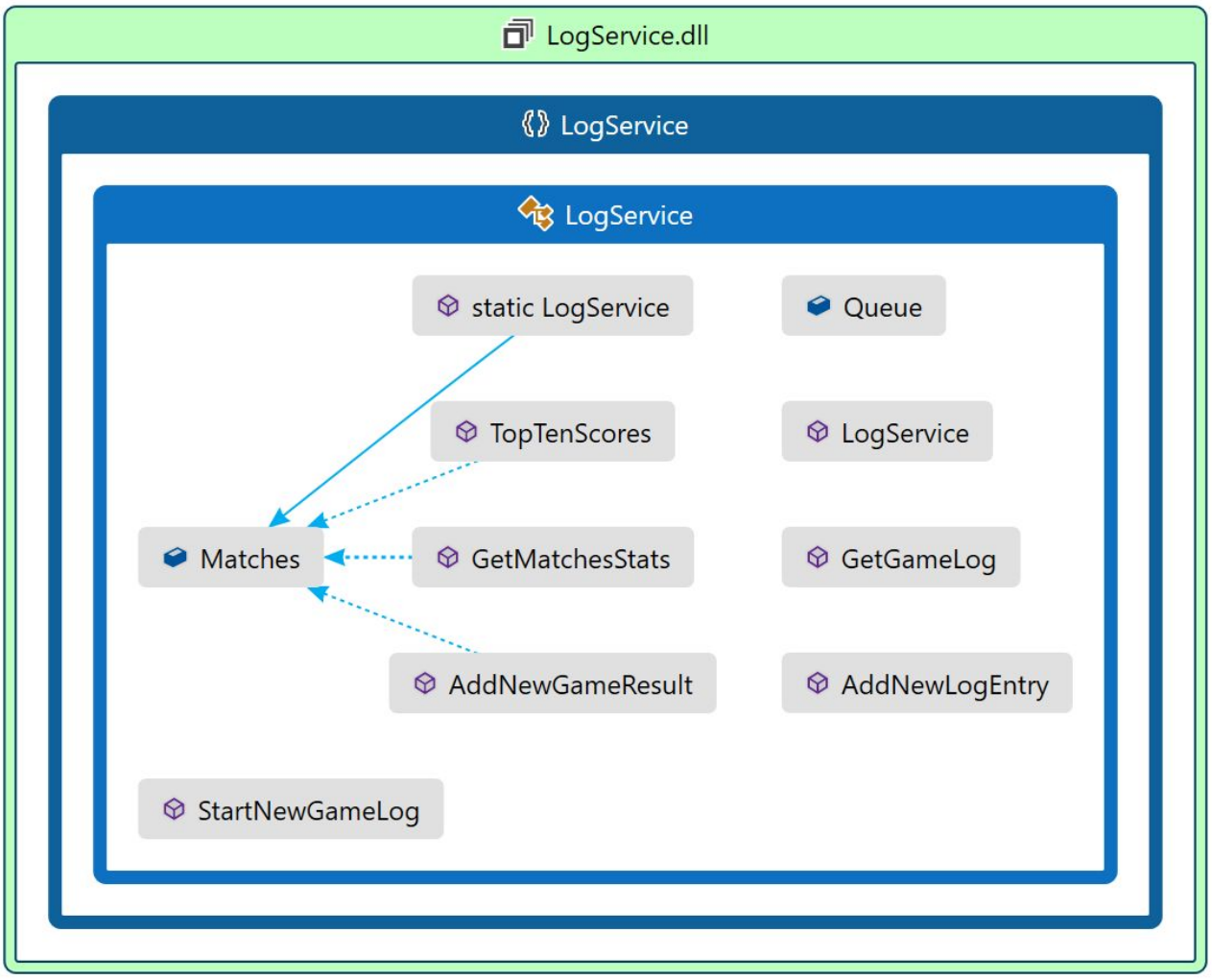
Para la funcionalidad Ver un log de la última partida, es necesario mantener guardados todos los logs para una partida reciente, por lo que se utilizan los métodos `StartNewGameLog`, `AddNewLogEntry` y `GetGameLog`. La clase `Game` que es la que maneja los comandos durante la partida (attack, move, etc) será quién llame a estos métodos para mantener el registro correspondiente.

Para la funcionalidad Ver estadísticas de los jugadores, se utiliza el método `AddNewGameResult`, para guardar toda la información de una partida y luego desde el cliente de este servicio (la `WebApi` en este caso), se utiliza el método `GetMatchesStats`.

Para la funcionalidad Ver una lista con los diez puntajes más altos, se utiliza la información ya agregada por `AddNewGameResult`, se recorren los jugadores de todas las partidas jugadas y se muestran aquellos con los puntajes más altos. Para realizar un sistema de puntos sencillo, se optó en asignarle 10 puntos a los jugadores que eliminen a otro y 100 puntos a los jugadores que ganen la partida.



LogService

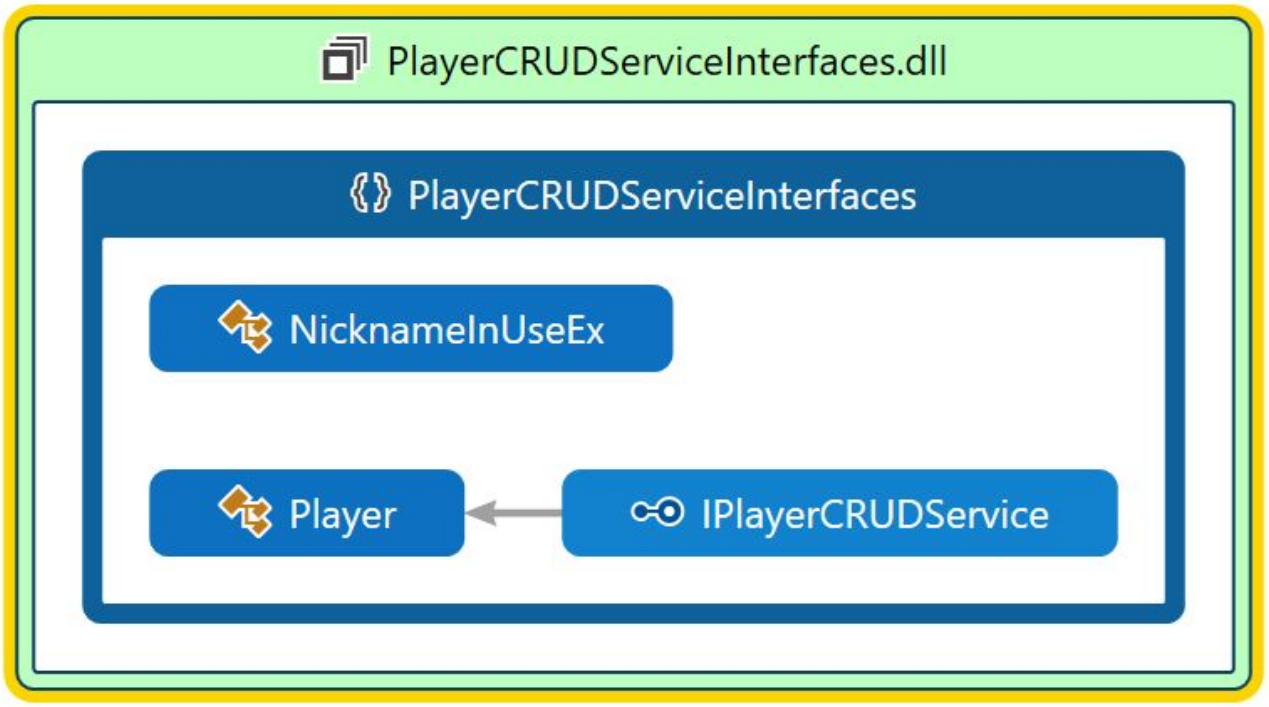


`LogService` contiene la implementación de `ILogService`. Contiene un repositorio de `Match` (para guardar la información de las partidas) y una cola de `MessageQueue` para mantener las entradas del log de la última partida jugada. Se implementan los métodos descritos en la sección anterior.

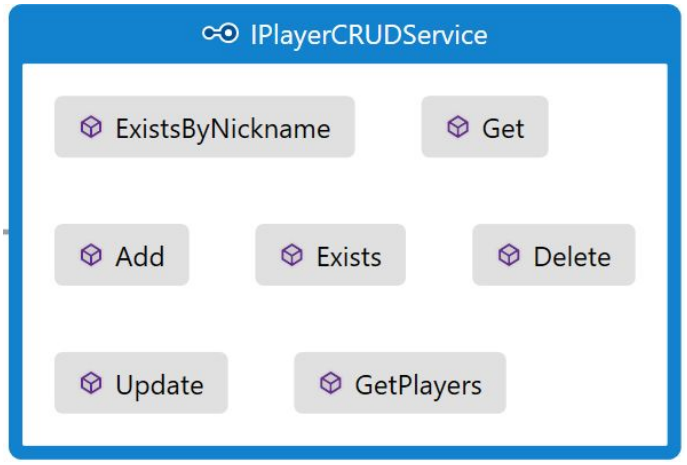
LogServiceHost

`LogServiceHost` se encarga de iniciar el servicio para ser utilizado por Remoting.

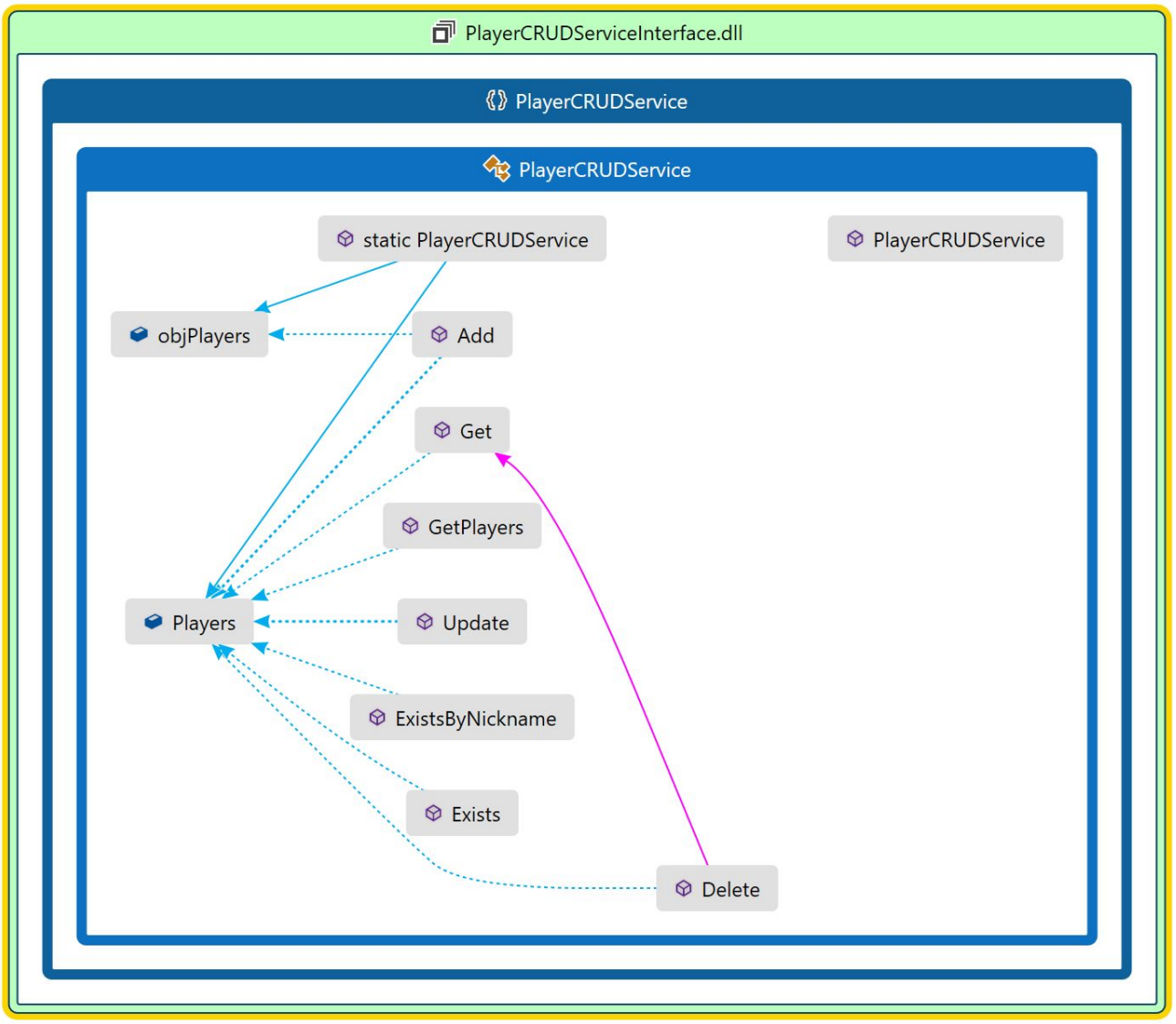
PlayerCRUDServiceInterfaces



`PlayerCRUDServiceInterfaces` contiene la interfaz en donde se especifican las operaciones del ABM de jugadores. Se tienen las operaciones `Add`, `Get`, `GetPlayers`, `Exists`, `ExistsByNickname`, `Update` y `Delete`.

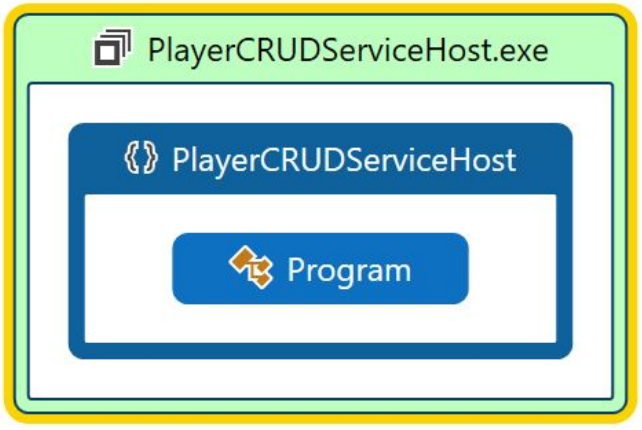


PlayerCRUDService



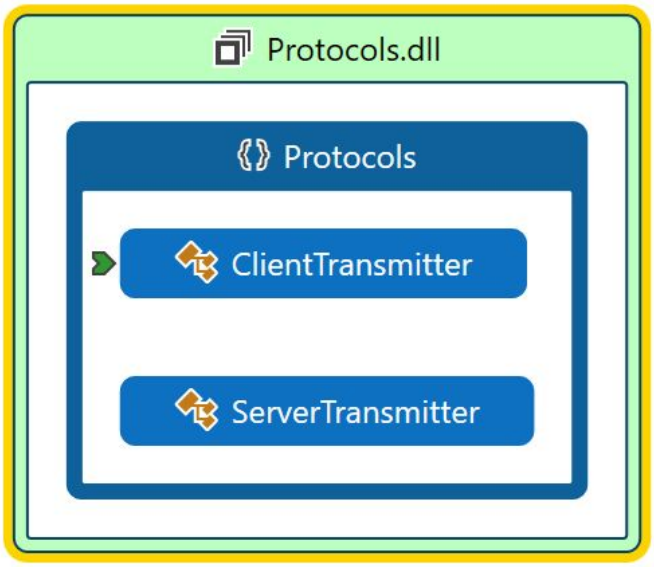
`PlayerCRUDService` contiene la implementación de `IPlayerCRUDService`. Mantiene un repositorio para guardar los jugadores y un objeto `objPlayers` para manejar la concurrencia de las operaciones.

PlayerCRUDServiceHost



PlayerCRUDServiceHost se encarga de iniciar el servicio para ser utilizado por Remoting.

Protocols



`Protocols` contiene clases con los protocolos utilizadas por `GameServer` y `GameClient` para comunicarse entre sí.

WebApi

`WebApi` funciona como cliente de `PlayerCRUDServiceHost` y `LogServiceHost`. Se implementaron controladores para poder realizar las consultas correspondientes a los dos servicios mencionados.

Configuración

En los archivos de configuración de los proyectos de los servicios de Remoting y de aquellos que consumen de estos servicios, es necesario especificar las mismas IP, puertos y nombres de los servicios correspondientes.

GameServer

App.config

```
<add key="PlayerCRUDServiceHostIP" value="127.0.0.1"/>
<add key="PlayerCRUDServiceHostPort" value="7000"/>
<add key="PlayerCRUDServiceHostName" value="PlayerCRUDService"/>

<add key="LogServiceHostIP" value="127.0.0.1"/>
<add key="LogServiceHostPort" value="8000"/>
<add key="LogServiceHostName" value="LogService"/>
```

LogServiceHost

App.config

```
<add key="LogServiceHostIP" value="127.0.0.1"/>
<add key="LogServiceHostPort" value="8000"/>
<add key="LogServiceHostName" value="LogService"/>
```

PlayerCRUDServiceHost

App.config

```
<add key="PlayerCRUDServiceHostIP" value="127.0.0.1"/>
<add key="PlayerCRUDServiceHostPort" value="7000"/>
<add key="PlayerCRUDServiceHostName" value="PlayerCRUDService"/>
```

WebApi

Web.config

```
<add key="PlayerCRUDServiceHostIP" value="127.0.0.1"/>
<add key="PlayerCRUDServiceHostPort" value="7000"/>
<add key="PlayerCRUDServiceHostName" value="PlayerCRUDService"/>

<add key="LogServiceHostIP" value="127.0.0.1"/>
<add key="LogServiceHostPort" value="8000"/>
<add key="LogServiceHostName" value="LogService"/>
```

En la carpeta del Obligatorio se encuentra la colección y el ambiente para probar las requests hacia la Web API con Postman.