Conecta Plus

**Juego multijugador**

*Eulogio Quemada Torres, Antonio Cañete Baena*

Ingeniería del Software – Software para Sistemas Empotrados y Dispositivos Móviles

Universidad de Málaga – Curso 2024/2025

# **Introducción**

En este trabajo, hemos desarrollado el clásico juego de conecta-4, pero de longitud variable. Lo hemos llamado Conecta Plus.

En el juego, tenemos un modo *SinglePlayer* en el que se puede jugar contra una IA, con los modos fácil, medio y difícil. También tenemos un modo *MultiPlayer* en el que se juega con otro jugador humano.

También, contamos con un historial de partidas jugadas (tanto offline como online) que usa una base de datos SQLite, una pestaña de acerca de la aplicación y un menú de opciones en el que se puede cambiar el idioma de la aplicación manualmente.

El diseño de la aplicación es eficiente y hemos tratado de seguir las buenas prácticas del diseño de software.

# **Explicaciones**

Aunque el proyecto es de Android, al haber realizado un modo multijugador, hemos tenido que implementar un servidor y un protocolo. El protocolo se explica en el siguiente apartado, y el código del servidor lo hemos incluido en esta entrega. Si se quiere probar a jugar con el servidor ejecutándose en un servidor local, se puede cambiar en la clase *WebSocketSingleton* la variable *WS\_URL* por la URL del servidor que sea, pero será necesario incluir esa dirección en el archivo *network\_security\_config.xml* para permitir las conexiones.

Ahora mismo, tenemos un servidor propio ejecutando el servicio y por tanto se debería poder jugar sin problema, aunque lógicamente hacen falta dos personas para probar el modo multijugador, si no solo podrá crear salas, pero nadie se unirá.

La idea que hemos seguido para el modo multijugador es que un jugador crea una sala, y el servidor le devuelve un código (un número de 4 dígitos). Otro jugador se unirá a esa sala con ese número, y a partir de ahí estarán conectados y el servidor hará de intermediario entre las comunicaciones.

Los jugadores podrán tirar las fichas en su turno y también tendrán un chat por el que podrán hablar.

Nos hubiera gustado ampliar la funcionalidad y meter más cosas como puntos que obtienes por ganar y con ellos puedes cambiarte el color de tus fichas, o que los usuarios se puedan poner un *username* para el modo multijugador, hacer también variable el modo multijugador (ya que solo se puede cambiar el tamaño del tablero y de más en el modo jugar con la IA) pero entre los exámenes finales, las vacaciones y otros trabajos no hemos tenido tanto tiempo, aunque había muchas ideas más para hacer.

En el modo jugar con la IA, esta está implementada con el algoritmo de la poda alfa beta. En el modo fácil la IA simplemente elige aleatoriamente. En el medio, es la poda alfa beta con una profundidad del número de fichas para gana, y en el modo difícil, es una profundidad del número de fichas para ganar por 2. Este último modo puede tardar en cargar si se pone un tablero demasiado grande.

Como hemos comentado antes, hemos intentado seguir las buenas prácticas de la programación y hacer el código lo más *clean* posible.

# **Protocolo Conecta Plus**

Este es el protocolo de comunicación que hemos desarrollado para la implementación del modo multijugador.

## **Descripción General del Protocolo**

El protocolo **Conecta Plus v1** está basado en un sistema de comunicación cliente-servidor utilizando WebSockets para garantizar una interacción en tiempo real. Este protocolo cubre todas las operaciones esenciales de un juego multijugador como la creación y unión a partidas, el movimiento de las piezas, el chat entre jugadores y la desconexión o finalización de la partida.

A continuación, se detallan los mensajes específicos que componen la comunicación entre el **cliente** (jugador) y el **servidor**.

## **Mensajes del Cliente**

El cliente (jugador) puede enviar los siguientes tipos de mensajes al servidor:

* + 1. **CHAT msg**  
       El jugador envía un mensaje de chat (msg), que será reenviado por el servidor al jugador con el que está jugando.  
       **Formato**: CHAT <msg>
    2. **CREATE**  
       El jugador crea una nueva sala de juego. El servidor asignará un código único para la sala creada.  
       **Formato**: CREATE
    3. **JOIN code**  
       El jugador intenta unirse a una sala existente, identificada por su código único (code).  
       **Formato**: JOIN <code>
    4. **MOVE x**  
       El jugador realiza un movimiento, colocando su ficha en la casilla identificada por x.  
       **Formato**: MOVE <x>
    5. **DISCONNECT**  
       El jugador indica al servidor que desea desconectarse manualmente de la partida.  
       **Formato**: DISCONNECT
    6. **END**  
       El jugador indica al servidor que la partida ha finalizado y que desea terminarla.  
       **Formato**: END

## **Mensajes del Servidor**

El servidor responderá a las acciones de los jugadores con los siguientes mensajes:

* + 1. **CHAT msg**  
       El servidor envía un mensaje de chat a uno de los jugadores con el que está interactuando.  
       **Formato**: CHAT <msg>
    2. **CREATE code**  
       El servidor confirma la creación de una nueva sala enviando el código de la sala al jugador que la ha creado.  
       **Formato**: CREATE <code>
    3. **START**  
       El servidor indica a ambos jugadores que la partida ha comenzado.  
       **Formato**: START
    4. **DENY**  
       El servidor rechaza la solicitud de un jugador para unirse a una sala. Esto ocurre cuando la sala no existe, ya está llena o el jugador intenta unirse a una partida en la que ya está.

**Formato**: DENY

* + 1. **MOVE x**  
       El servidor envía el movimiento realizado por un jugador al otro jugador en la partida.  
       **Formato**: MOVE <x>
    2. **DISCONNECT**  
       El servidor informa al jugador que el otro jugador se ha desconectado de la partida.  
       **Formato**: DISCONNECT

## **Flujo de Ejemplo**

El siguiente ejemplo describe una secuencia típica de comunicación entre los jugadores (J1 y J2) y el servidor.

* + 1. **J1 crea una sala**  
       J1 envía un mensaje al servidor para crear una nueva sala:  
       J1 --- CREATE ---> SERVER  
       El servidor responde con el código de la sala:  
       SERVER --- CREATE <code> ---> J1
    2. **J2 se une a la sala**  
       J2 envía el código de la sala para unirse a ella:  
       J2 --- JOIN <code> ---> SERVER  
       Si la sala está disponible, el servidor envía un mensaje de inicio a ambos jugadores:  
       SERVER --- START ---> J1, J2
    3. **Movimiento de los jugadores**  
       Los jugadores realizan sus movimientos en la partida, y el servidor se encarga de reenviar los movimientos entre ellos:  
       J1 --- MOVE <x> ---> SERVER  
       SERVER --- MOVE <x> ---> J2  
       J2 --- MOVE <x> ---> SERVER  
       SERVER --- MOVE <x> ---> J1
    4. **Interacción de Chat**  
       Los jugadores pueden intercambiar mensajes de chat durante la partida:  
       J1 --- CHAT <msg> ---> SERVER  
       SERVER --- CHAT <msg> ---> J2
    5. **Finalización de la partida**  
       Cuando uno de los jugadores decide terminar la partida, envía el mensaje de finalización:  
       J1 o J2 --- END ---> SERVER

## **Implementación del Servidor**

El servidor está diseñado utilizando **Python** y la biblioteca **Websockets** para la comunicación en tiempo real. A continuación, se destacan las características clave de la implementación del servidor:

* **Manejo de conexiones**: El servidor puede gestionar múltiples conexiones simultáneas utilizando **asyncio** y **websockets**.
* **Gestión de partidas**: Las partidas son gestionadas en un diccionario (matches), donde cada sala se identifica por un código único.
* **Seguridad y Consistencia**: El protocolo garantiza que un jugador no pueda crear múltiples partidas simultáneamente ni unirse a varias partidas a la vez. Además, si un jugador se desconecta, su partida es eliminada y el otro jugador es notificado.

Este protocolo está diseñado para ser fácilmente extensible, permitiendo futuras actualizaciones y mejoras.