Programa Cliente-Servidor con sockets

El juego del tres en Raya

Proyecto desarrollado para la asignatura Programación de Servicios y Procesos
Segunda Evaluación, Segundo de DAM Dual

Cristina Ramos
Javier Reyes
Santiago Sáenz de Santa María

cpi'fp
Los Enlaces

Introducción y concepto

Dos jugadores alternan para colocar sus marcas, generalmente X y O, en un tablero de 3x3,

El objetivo del juego es formar una línea recta horizontal, vertical o diagonal antes que el oponente.

Se implementa este lógica de juego a un programa Cliente - Servidor

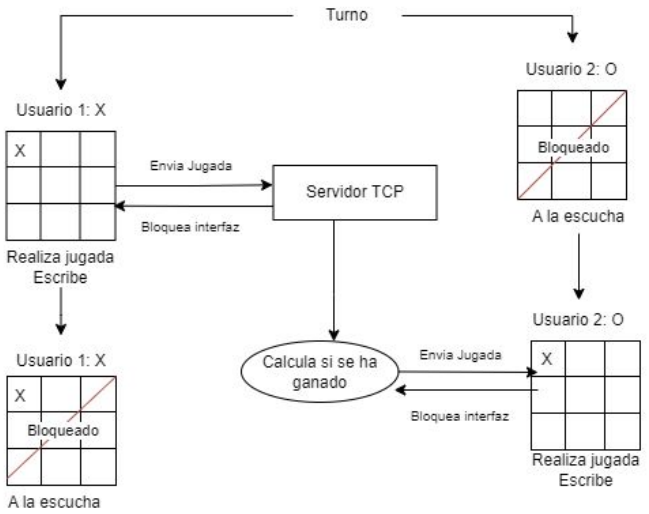


Diagrama de estados del juego.

Objetivos

01.

Crear un protocolo de comunicación que permita a dos usuarios jugar al tres en raya en una misma red local.

02.

Los usuarios compartirán el mismo tablero, deben sincronizarse entre un cliente y un servidor y enviar sus jugadas.

03.

Los usuarios realizan jugadas avisando al oponente y declarándose ganador si uno de ellos cumple la condición de partida ganada.

04.

Guardar cifradas las partidas que se han realizado y mostrarlas descifradas al final de una partida ganada.

Conceptos a conocer

Arquitectura Cliente Servidor

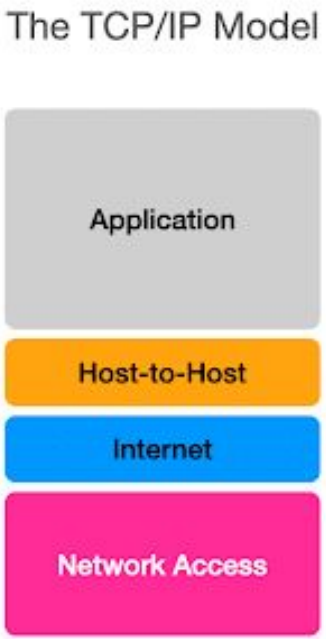
Un servidor provee una serie de servicios o recursos que serán consumidos por un cliente.

Debe existir un mecanismo que permita conectarse, en este caso TCP/IP

TCP /IP

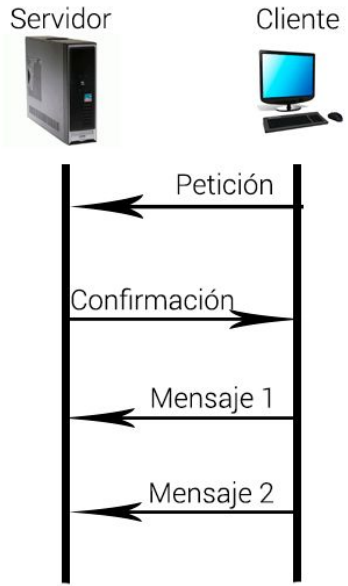
Protocolo orientado a la conexión; durante la transmisión de datos hay una conexión fija lógica entre cliente y servidor.

La IP es una dirección de identificación única de un dispositivo en una red informática.



Modelo TCP/IP.

Conceptos a conocer



Representación Servidor cliente.

Socket Servidor

La clase ServerSocket permite manipular la conexión desde el lado del servidor.

Gestiona la conexión con los clientes

Socket Cliente

La clase Socket de java se usa para implementar la conexión desde el lado del cliente.

Envía los movimientos del usuario al servidor

Primeros pasos y errores

Se comenzó con la idea de instanciar el tablero por medio un ***array de objetos Botones*** que se envían del cliente al servidor.

Problemas de este planteamiento

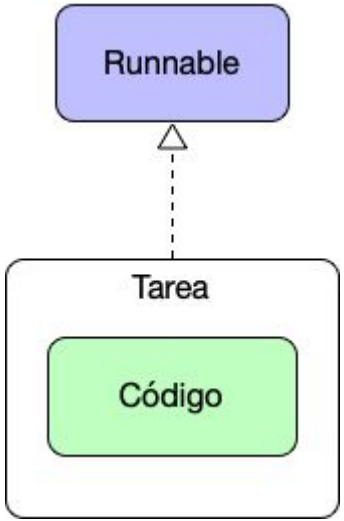
Serialización y deserialización de los objetos. Consume muchos recursos y es ineficiente.

Además, tanto el emisor como el receptor deben tener las mismas versiones de las clases.

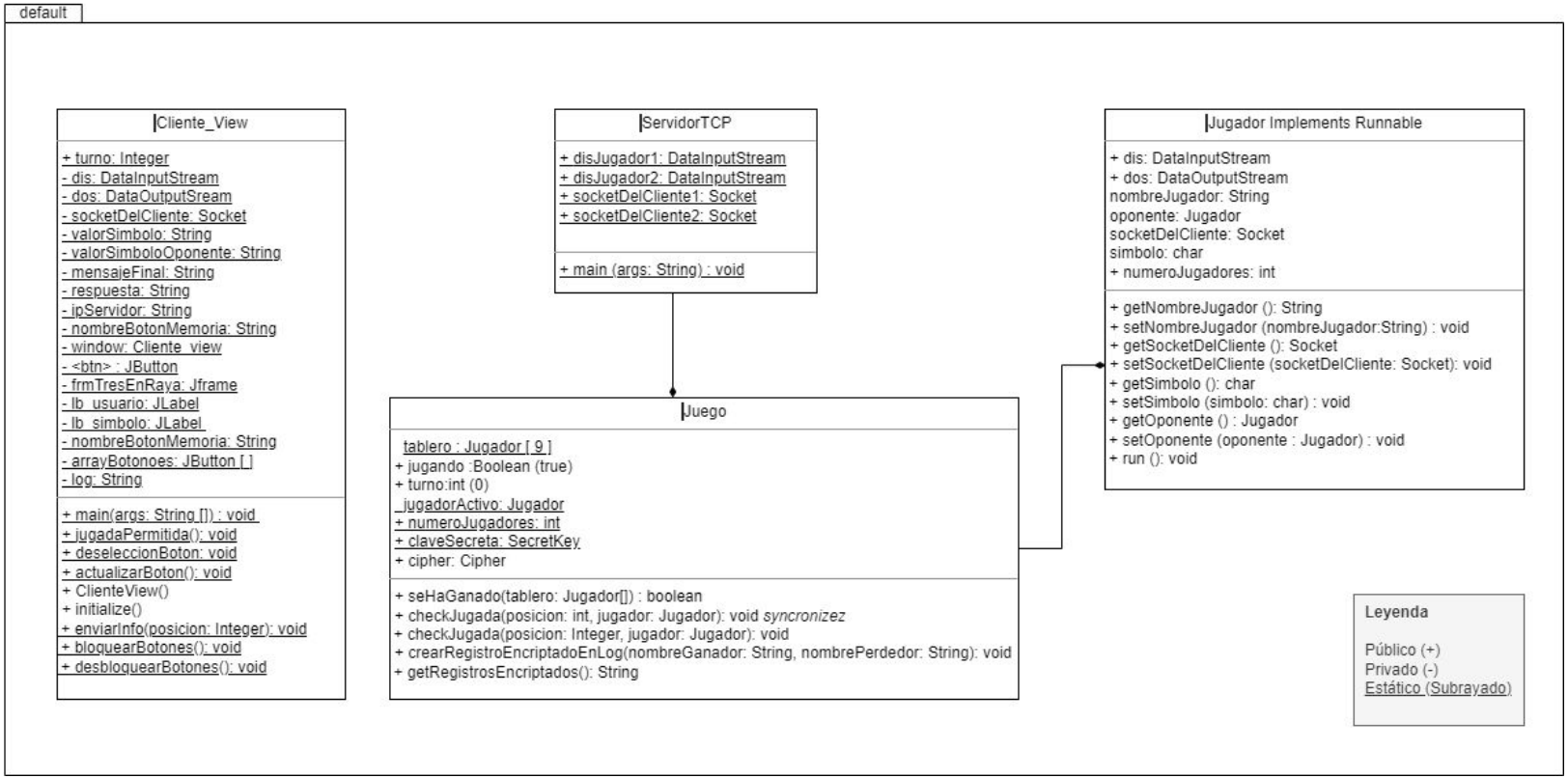
Concurrencia. El cliente se bloqueaba tras la interacción del segundo jugador pero el servidor sigue funcionando dado que no hay nada que lo interrumpa.

Comienzo de la idea

- Se vió necesario la implementación de ***Hilos*** para manejar múltiples conexiones de clientes simultáneamente con la interfaz ***Executor***.
- Esto permite que el servidor pueda manejar varias soluciones de clientes al mismo tiempo sin que se bloquee. Se arregla el problema de la concurrencia
- Además, se pueden realizar múltiples conexiones gracias a la clase ***Runnable***.
- Si se crean más Sockets, se pueden conectar más jugadores



Interfaz runnable



Explicación de las clases

Clase Juego

Esta clase contiene la lógica principal del juego del tres en raya.

Controla el tablero del juego, verifica si alguien ha ganado y maneja el registro de las partidas.

Cada instancia de Juego tiene un método run que escucha las jugadas de un jugador específico.

Interfaz gráfica del juego.

Explicación de las clases

Son dos jugadores pero la misma clase cliente..

Clase Jugador

Implementa la interfaz Runnable para ejecutar en un hilo separado.

Cada jugador tiene su propio flujo de entrada y salida de datos.

Escucha las jugadas del jugador y las envía al otro jugador a través del servidor usando sus flujos propios de entrada y salida.

En esta clase se manejan las comunicaciones con los clientes a través de sockets

Funcionamiento

1. Se lanza el servidor y espera a que se conecten dos clientes.
2. El cliente debe de indicar la dirección ip a la que debe conectarse y un nombre identificativo.
3. Una vez conectados les asigna un símbolo de jugador, “X” o “O”.
4. El jugador “O” se queda a la espera de que el jugador “X” pulse en una casilla.
5. Tras ser pulsada, se envía el servidor y se evalúa la jugada indicando si hay un ganador o no.
6. Si el tablero está lleno sin ganador se envía un mensaje de empate.
7. Además, se controla si un jugador ha abandonado la partida.

Diagrama de conexión

Al iniciarse el servidor se crea un Juego y a los Jugadores con sus sockets.

Los clientes se conectan y crea un jugador para gestionar las conexiones con el cliente.

La función void run establece las comunicaciones y define las acciones que llevará a cabo cada objeto de la clase Jugador.

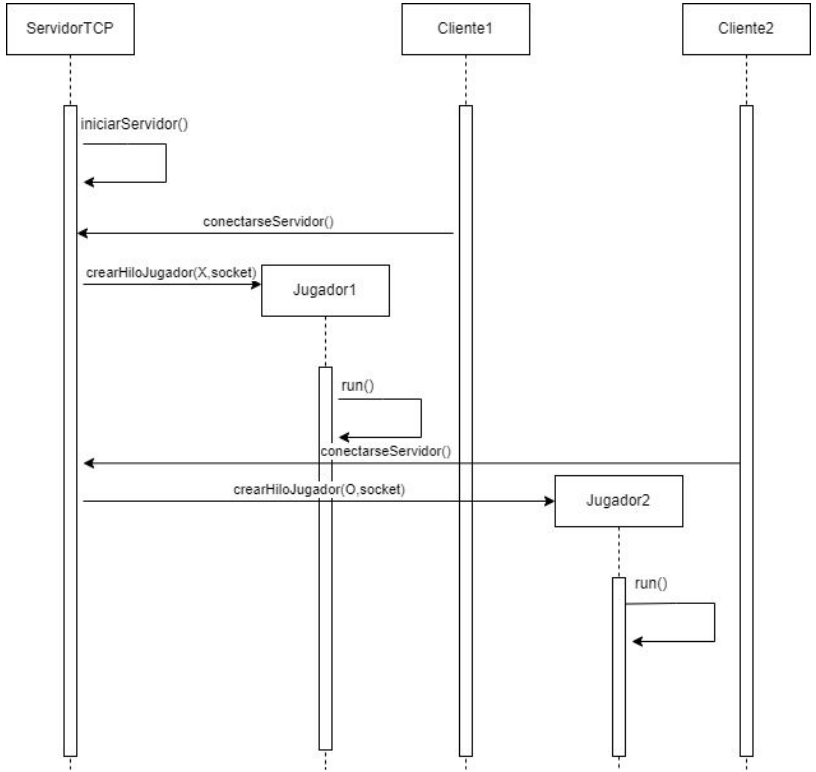


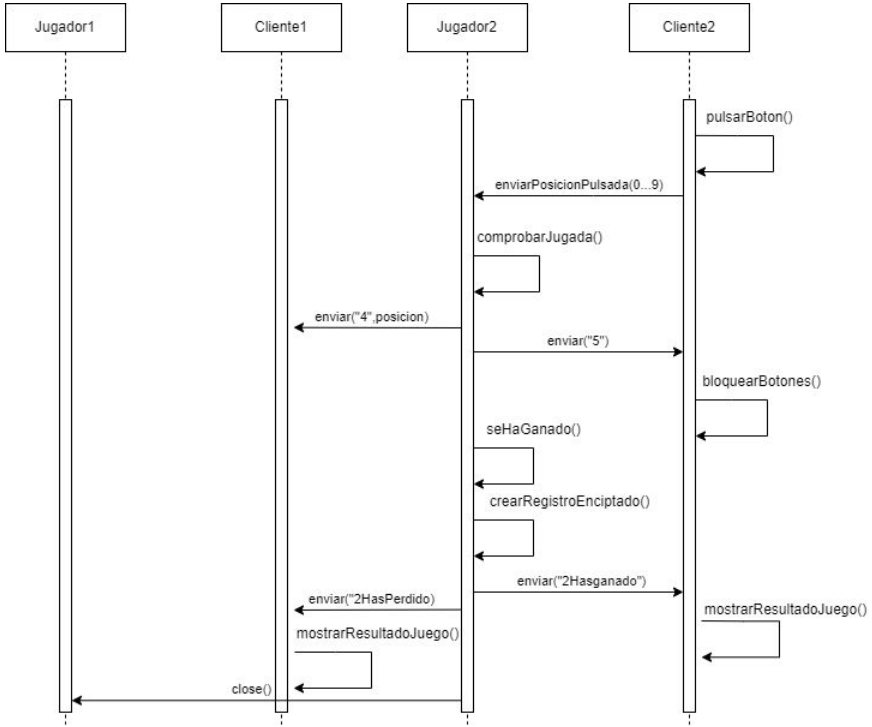
Diagrama de partida ganada

A través de los flujos de comunicación los jugadores se envían mensajes para evaluar las jugadas.

En el diagrama se muestra un caso de que se ***gane una partida***. El código “4” envía la posición y el “5” actualiza el tablero. Con el código “2” se indica que un jugador ha ganado.

Casos no mostrados en el diagrama

En caso de empate se envía el código “7” y un mensaje a ambos jugadores. Se controla si un jugador abandona antes de acabar el juego enviando “6”.



Arquitectura de software empleada

El código realizado para el tres en raya implementa una arquitectura de software sencilla en directa.

Este tipo de arquitectura se conoce como **monolítica** ya que está en una sola capa.

Todas las funcionalidades y componentes del sistema se implementan y despliegan como una sola unidad.

En este caso existe cierto grado de escalabilidad al tener el cliente y servidor separado permitiendo ejecutar el servidor desde un ordenador y el cliente en dos computadoras accediendo a través del Ip.

https://keepcoding.io/blog/cliente-servidor-con-arquitectura-monolitica/

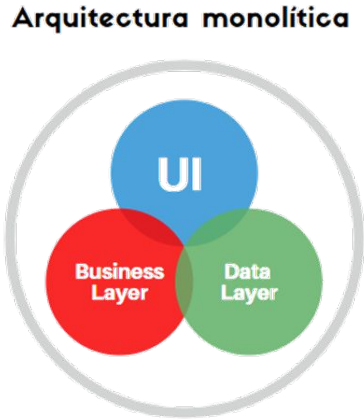


Diagrama de la arquitectura

Encriptado y desencriptado

01.

Programación Segura

implica escribir código que minimiza las vulnerabilidades y riesgos de seguridad.

02.

Encriptación

es el proceso de proteger datos mediante algoritmos con el fin de codificarlos de manera que solo las partes que tengan la clave podrá acceder a el mensaje.

03.

Codificación Base 64

representa los datos binarios mediante una cadena ASCII, traduciéndose en una representación radix-6 diseñada para tratar con datos textuales.

04.

Justificación

Dado que se guardan archivos de texto, por la sencillez de implementación y los datos no son sensibles es lo suficientemente segura.


¡Gracias!
Ahora vamos a echar unas partidas.
Que gane el mejor.
Presentación realizada por Cristina Ramos, Javier Reyes y Santiago Saenz de Santa María