

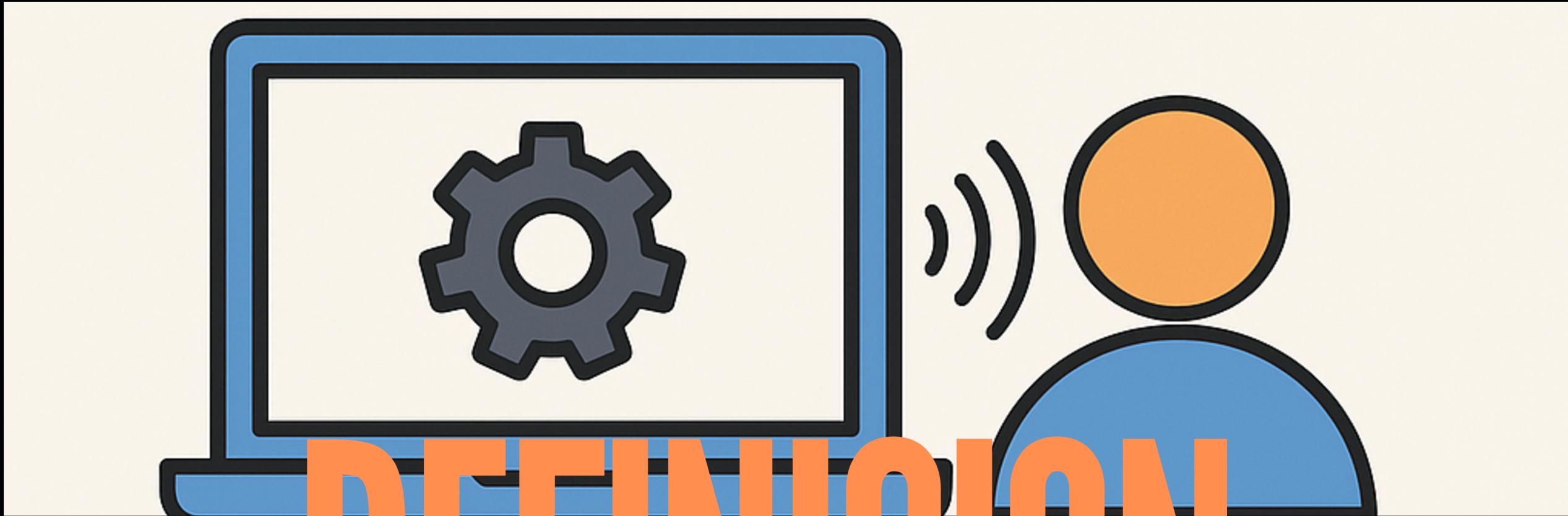


ERICK NAVA  
ERIC RAMIREZ

# Listener.

*Su concepto en relación con los eventos en los programas y sistemas operativos.*

Listener



# DEFINICION

Un listener **es un proceso que se mantiene en espera pasiva hasta que ocurre un evento específico**. Su **función es detectar ese evento y reaccionar inmediatamente**, ya sea ejecutando una acción, notificando a otro componente o iniciando un proceso.

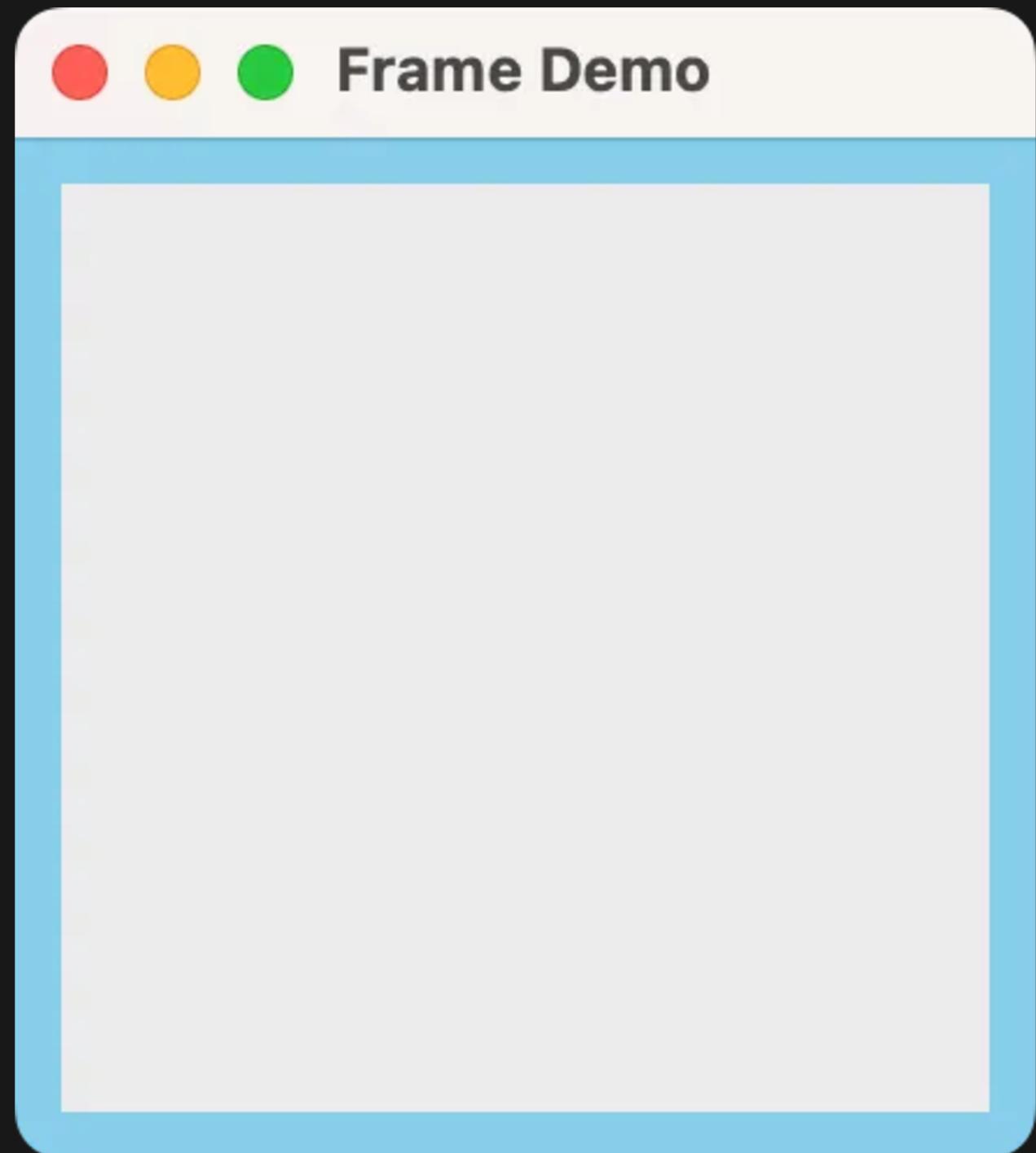
Listener

# EVENTO

Son las acciones que puede realizar el usuario que producen otros comportamiento por parte del programa

## EJEMPLO

Si el usuario pulsa un botón llamado “Salir” cerrará la aplicación.  
Validar texto que introduzcamos en un campo de texto.



Listener

# LISTENER TIME LAPSE

Primeras computadoras que trabajaban realizando una tarea a la vez.

Event loops para escuchar periféricos.

Prácticas contemporáneas: IoT, sensores, navegadores.



Inicios del internet y desarrollo de redes. Estar alerta para conexiones de red.

Auge como patrón de diseño con la POO.

# NOS PERMITE

1

## RESPUESTA AUTOMATICA A EVENTOS

Mostrar mensaje cuando el usuario haga clic

2

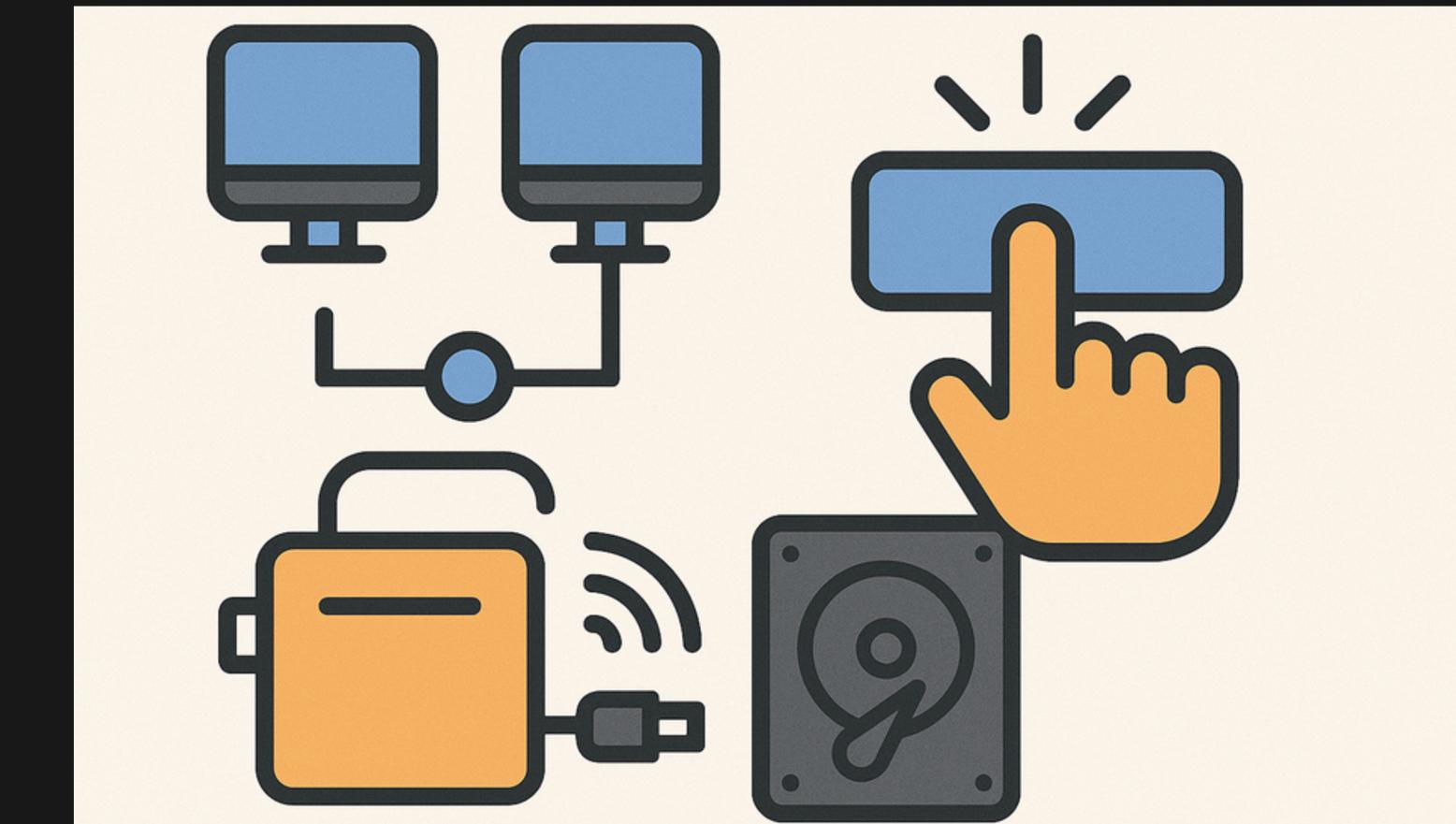
## DESACOPLAMIENTO DE LA LÓGICA DE DETECCIÓN Y RESPUESTA

Se le notifica al listener el clic, pero el botón no necesita tener integrado el mecanismo para mostrar el mensaje

3

## MODULARIDAD EN EL REGISTRO DE EVENTOS

Múltiples componentes que registran eventos



4

## FACILITA LA CONCURRENCIA

Continuar tareas mientras se espera un evento

5

## REDUCIR EL USO DE POLLING

No está revisando constantemente en busca de cambios, solo notifica cuando algo ocurre

Listener

# PATRÓN DE DISEÑO



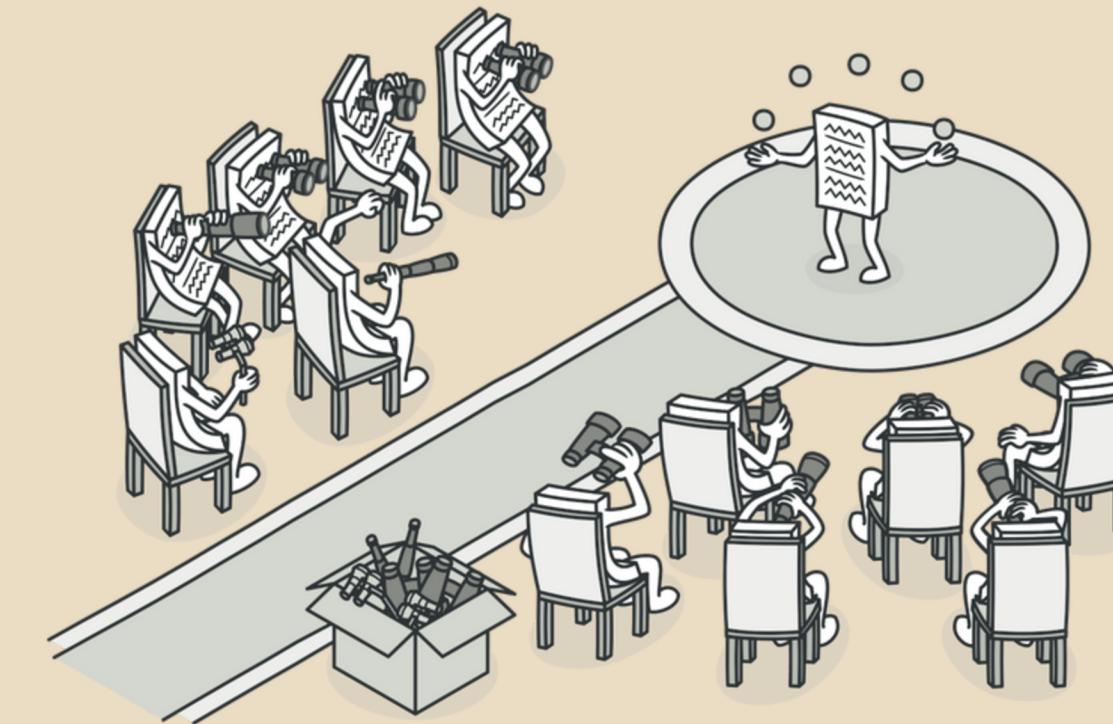
## Observer

Este permite definir un mecanismo para identificar un objeto como el observado, de manera que, cuando este cambie de estado o simplemente ocurra algún evento se le notifique a todos los otros objetos de este.



## Problema

Este patrón puede ser un problema porque el estar escuchando / observando puede ser un gasto de recursos si se tratan de muchos objetos, además estos objetos que están observando puede que nos siempre debían de estar pendientes a los eventos que le ocurrían al objeto observado.



## Solución

Una manera de evitar este tipo de estrés innecesario al sistema es crear una matriz para los objetos interesados en escuchar al objeto escuchado dentro de su interfaz, de tal forma que se tiene una lista de los suscriptores en concreto.

Observamos una clara  
adaptación del concepto con  
los kernels monolíticos

# SOFTWARE

- Llamadas al sistemas
- Notificación de estados de recursos
- Notificación de cambios en sistema de archivos

# HARDWARE

- Interrupciones de hardware
- Activación de entradas que tiene la CPU
- Decodificación de interrupciones

# INTERRUPCIONES DE HARDWARE EN LINUX

**Todo es un archivo. Todas las estructuras están en lenguaje C.**

El sistema es muy dependiente de la arquitectura.

# ESTRUCTURAS DE DATOS COMO:

- 01 irqaction: Almacena la dirección de la función que se está ejecutando (para manejar la interrupción).
- 02 irq\_action: Contiene las funciones que utiliza un controlador de interrupciones.
- 03 irq\_desc: Vector con entradas para cada una de las interrupciones que se pueden atender.
- 04 irq\_action: Apuntador a la dirección de la función que hay que llevar a cabo.



# Watcher

- Cambiamos a monitoreo activo
- Uso de polling
- Uso típico: Actualizaciones de red o memoria cuando existe un solicitante o provedor

```
while (true) {  
    if (archivo_fue_modificado()) {  
        hacer_algo();  
    }  
    sleep(1);  
}
```

# Watcher - Ejemplos

- **inotify (*node & notify*):** monitorea eventos del sistema de archivos
- **epoll (*event & poll*):** descriptores de archivos que son identificadores de recursos abiertos
- **fanotify (*file accesos notify*):** monitorea accesos a los archivos en todo el sistema

# JAVA

Se implementa como una interfaz

```
public class GameBoardPanel extends JPanel implements ActionListener {  
  
    private static final long serialVersionUID = 6802492405004738658L;  
    private static final int BoardWidth = 10;    // game board x size  
    private static final int BoardHeight = 22;   // game board y size
```

```
// keyboard listener
addKeyListener(new KeyAdapter() {
    @Override
    public void keyPressed(KeyEvent e) {
        if (!isStarted || curBlock.getShape() == Tetrominoes.NO_BLOCK) {
            return;
        }

        int keycode = e.getKeyCode(); //recibe el codigo de la tecla ingresada por teclado

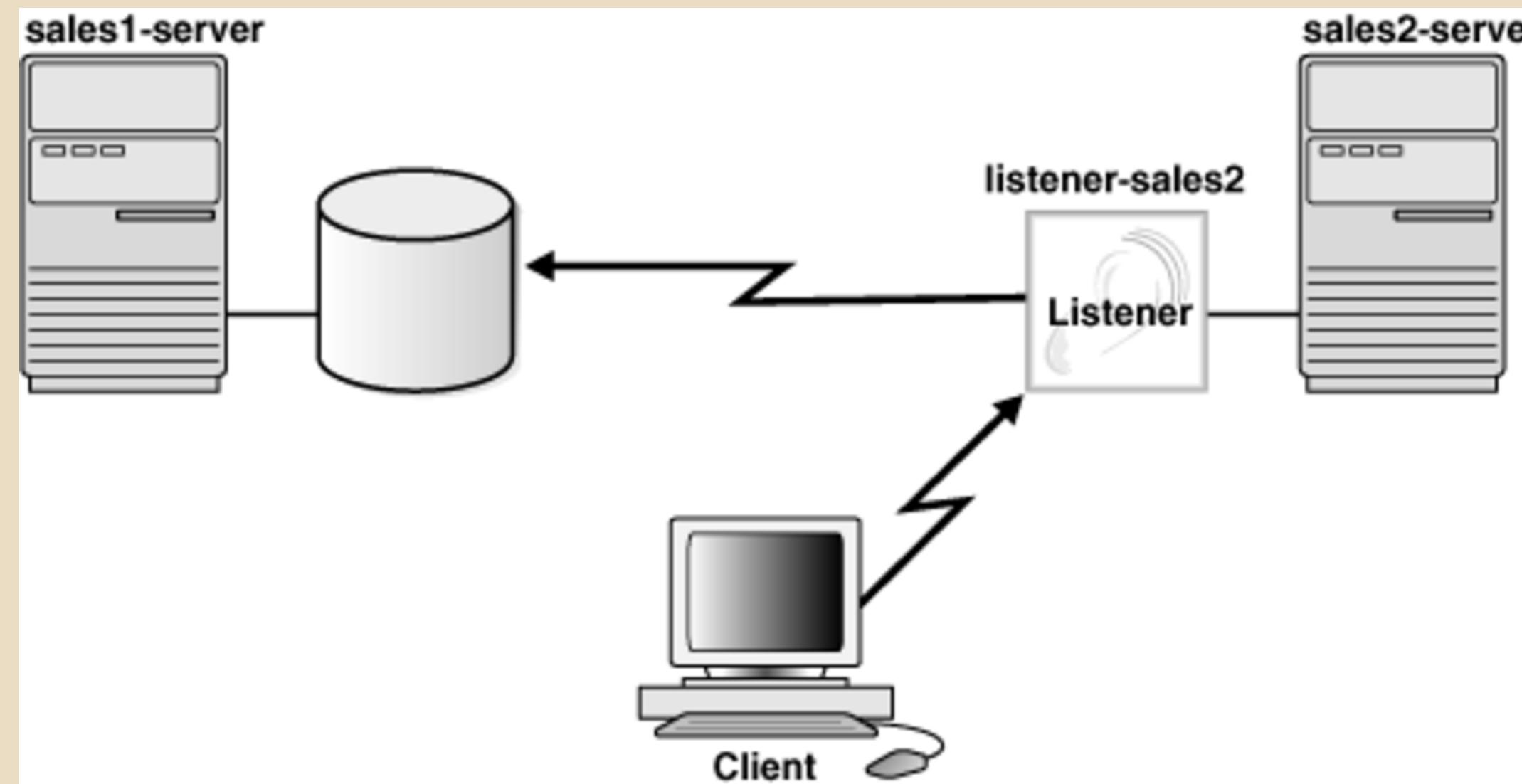
        if (keycode == 'p' || keycode == 'P') {
            pause();
            return;
        }

        if (isPaused) {
            return;
        }

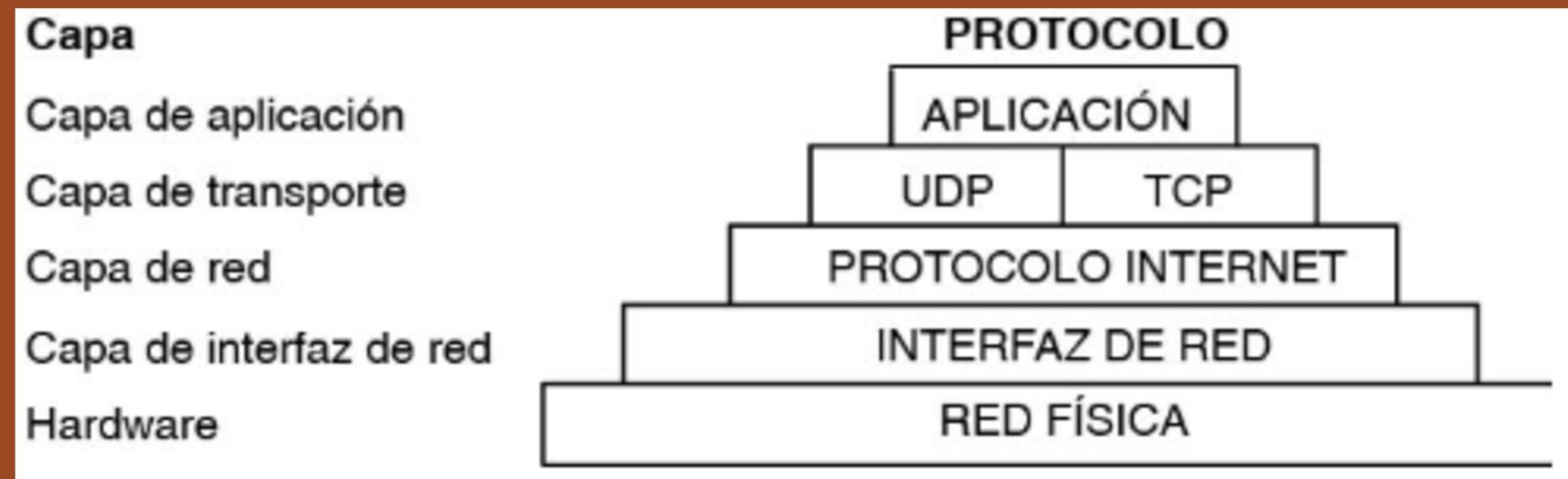
        switch (keycode) {
        case KeyEvent.VK_LEFT:
            isMovable(curBlock, curX - 1, curY);
            break;
        case KeyEvent.VK_RIGHT:
            isMovable(curBlock, curX + 1, curY);
            break;
        case KeyEvent.VK_UP:
            isMovable(curBlock.rotateRight(), curX, curY);
            break;
        case KeyEvent.VK_DOWN:
            advanceOneLine();
            break;
        case KeyEvent.VK_SPACE:
            advanceToEnd();
            break;
        case 'p':
        case 'P':
            pause();
            break;
        }
    }
});
```



# Oracle cuenta con un proceso llamado listener



- Gestiona el tráfico de solicitudes
- Gestiona las conexiones de los puertos
- Funciona con un protocolo TCP/IP

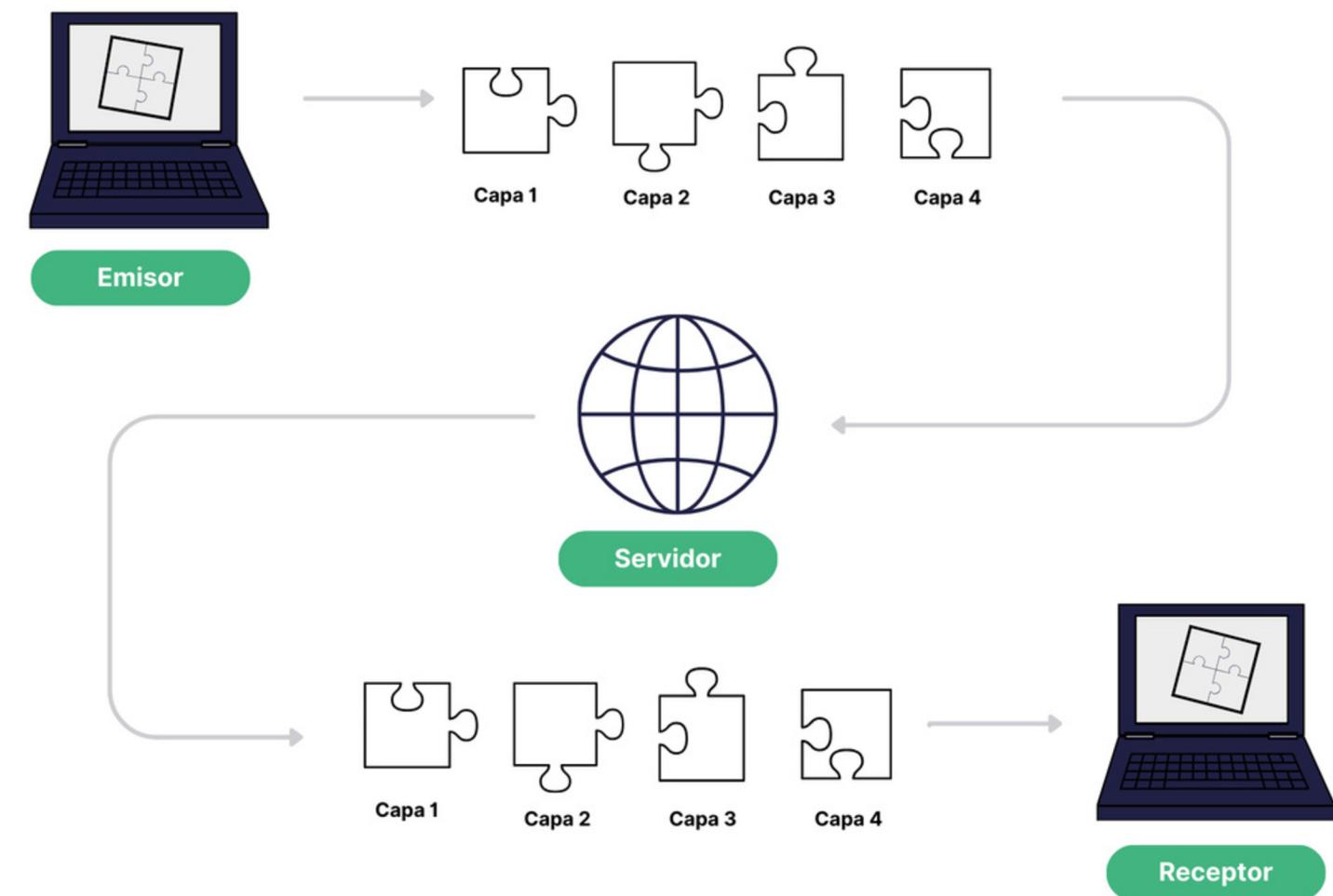


# **PROTOCOLO TCP/IP**

**Transmission Control Protocol / Internet Protocol**

Permite a programas, servicios y aplicaciones la comunicación con servicios de red

Está diseñado para la transferencia de paquetes que intercambian datos e información



**¿Es el mismo tipo de Listener?**

# NOO!

**El listener de oracle es un proceso que hace uso de un protocolo TCP/IP**

**Este cada vez que recibe una conexión generara un fork() y continuara a la espera de otra interacción**

```
1  INICIO servidor:  
2  1. Creacion socket TCP  
3  2. Enlazar socket a un puerto  
4  3. Funcion de listener  
5    a. Espera conexión  
6    b. crea un hijo(fork)  
7      I. El hijo cirra el socket  
8        Recibe datos  
9        Envie respuesta  
10       cierra conexión con cliente  
11       termina el procesos  
12  II. El padre cierra el descriptor del cliente  
13    continua a la espera de conexión de un cliente  
14
```

# REFERENCIAS

De Roer, D. D., & De Roer, D. D. (2022, Junio). Eventos y Listeners en Java. Disco Duro de Roer -. <https://www.discoduroderoer.es/eventos-y-listeners-en-java/>

El listener de Oracle | Dataprix. (2008). <https://www.dataprix.com/es/forum/oracle-database/el-listener-de-oracle>

Introduction to event listeners (The JavaTM Tutorials > Creating a GUI with Swing > Writing Event listeners). <https://docs.oracle.com/javase/tutorial/uiswing/events/intro.html>

Observer. (n.d.). <https://refactoring.guru/es/design-patterns/observer>

Grey, R. (2024, Octubre 13). ¿Qué es ARPANET? - NinjaOne. NinjaOne. <https://www.ninjaone.com/es/it-hub/it-service-management/que-es-arpnet/>

Mallón, X. (2025). Sockets: Qué son, cómo funcionan y tipos - Guía 2025. KeepCoding Bootcamps. <https://keepcoding.io/blog/que-es-un-socket/>

Universidad de Las Palmas de Gran Canaria, LECCIÓN 3: INTERRUPCIONES HARDWARE. [https://sopa.dis.ulpgc.es/ii-dso/leclinux/interrupciones/int\\_hard/LEC3\\_INT\\_HARD.pdf](https://sopa.dis.ulpgc.es/ii-dso/leclinux/interrupciones/int_hard/LEC3_INT_HARD.pdf)

Wolf, G., Ruiz, E., Bergero, F., & Meza, E. (2015). *FUNDAMENTOS DE SISTEMAS OPERATIVOS*, Universidad Nacional Autónoma de México.

*inotify(7) - Linux manual page*. <https://man7.org/linux/man-pages/man7/inotify.7.html>

*epoll(7) - Linux manual page*. <https://man7.org/linux/man-pages/man7/epoll.7.html>

*Introduction to event listeners (The JavaTM Tutorials > Creating a GUI with Swing > Writing Event listeners)*. (n.d.). <https://docs.oracle.com/javase/tutorial/uiswing/events/intro.html>

AIX 7.2. (s. f.). <https://www.ibm.com/docs/es/aix/7.2?topic=protocol-tcpip-protocols>

August2024. (2024, 24 octubre). Service data provider. Oracle Help Center.

<https://docs.oracle.com/en/cloud/paaS/integration-cloud/visual-developer/service-data-provider.html>

Database net Services reference. (s. f.).

[https://docs.oracle.com/cd/E11882\\_01/network.112/e10835/lsnrctl.htm#CHDBDHF](https://docs.oracle.com/cd/E11882_01/network.112/e10835/lsnrctl.htm#CHDBDHF)

Despliegue de una topología de DR híbrida para una Oracle Exadata local. (s. f.). Oracle Help Center. <https://docs.oracle.com/es/solutions/hybrid-dr-for-exadata/configure-listeners-and-sqlnet-ora.html#GUID-8E4A6551-2A7D-4A50-9A42-C201420012A3>

Starting the Oracle Listener (Sun Java System Mobile Enterprise Platform 1.0 Installation Guide). (s. f.). <https://docs.oracle.com/cd/E19957-01/820-3750/ggrgc/index.html>

Tivoli NetCool Performance Manager 1.4.4. (s. f.).

<https://www.ibm.com/docs/en/tnpm/1.4.4?topic=bit-configure-oracle-listener>

Williams, D., & Jashnani, P. (2024, 5 septiembre). Configuring and Administering Oracle Net Listener. Oracle Help Center.

<https://docs.oracle.com/en/database/oracle/oracle-database/19/netag/configuring-and-administering-oracle-net-listener.html#GUID-09139A19-5265-4216-9054-3237B225E09D>