

# Esquema Didáctico

**Módulo: Programación de Servicios y Procesos**

**2.º DAM – Actividad: “3 en raya multijugador”**

## ◆ 2.1 ¿Qué supuesto queremos trabajar?

🎯 Objetivo general:

- Diseñar y programar **servicios concurrentes**, seguros y eficientes
- Usar **procesos, hilos, comunicación y sincronización**

📌 Simulación práctica basada en arquitectura cliente-servidor

## 2.2 Contextualización del alumnado

- Ciclo: **CFGs – Desarrollo de Aplicaciones Multiplataforma**
- Curso: **2.º DAM**
- Alumnado familiarizado con programación Java, sockets y control de flujo
- Nivel medio-alto en estructuras, concurrencia básica y depuración

## 2.3 Conocimientos previos requeridos

- Programación orientada a objetos
- Threads y procesos en Java o Python
- Sockets TCP
- Gestión básica de recursos compartidos
- Entrada/salida y estructuras de datos simples

## 🎓 2.4 Objetivos de aprendizaje

- ✓ Implementar un servidor concurrente y multicliente
- ✓ Coordinar procesos e hilos para tareas simultáneas
- ✓ Aplicar mecanismos de **IPC (sockets)**
- ✓ Sincronizar turnos y recursos compartidos
- ✓ Desarrollar software robusto y planificado

## 2.5 Metodología

- ◆ Aprendizaje basado en proyectos (ABP)
- ◆ Trabajo por parejas o grupos reducidos
- ◆ Desarrollo progresivo por hitos
- ◆ Evaluación continua + final
- ◆ Apoyo técnico individualizado

## 2.6 Material didáctico (DUA)

 Material adaptado a diferentes ritmos:

- Código base comentado
- Diagrama de arquitectura cliente-servidor
- Tests para validar partidas
- Guía técnica de sockets + planificación de procesos
- Opción de desarrollo en Java, Python o pseudocódigo



## 2.7 Secuencia de acciones formativas





1. Introducción a servidores concurrentes y sockets
2. Explicación de procesos e hilos
3. Desarrollo del servidor principal (gestor de partidas)
4. Implementación de la lógica de turnos y comunicación
5. Pruebas en red local / entorno virtualizado
6. Presentación del proyecto + defensa técnica



## 2.8 Actividad principal:

“3 en raya multijugador (conurrencia en red)”

 El alumnado desarrolla:

-  Un **servidor concurrente TCP** que gestiona múltiples partidas
-  Cada **mesa de juego** como un proceso/hilo independiente
-  El **servidor asigna procesos dinámicamente** según disponibilidad
-  Control de turnos, estado del tablero y sincronización entre jugadores

**Tecnologías:**

- Java (Sockets + Threads)
- Alternativa: Python (Sockets + threading / multiprocessing)



## 2.9 Evaluación: Instrumentos y criterios



### Criterios:

- Correcta gestión de conexiones concurrentes
- Sincronización efectiva de turnos
- Comunicación robusta entre cliente y servidor
- Código modular y comentado
- Presentación funcional y comprensible



### Instrumentos:

- Rúbrica detallada
- Pruebas funcionales en clase
- Observación directa del proceso
- Defensa y demo final

## 2.10 Inclusión y atención a la diversidad

- Código base disponible
- Tareas divididas en entregas semanales
- Feedback individualizado
- Opción de entregar versión simplificada sin sockets reales (modo local)
- Apoyo visual con esquemas

## 2.11 Actividades de ampliación

 Para quienes completen la actividad base:

- Interfaz gráfica (Swing, JavaFX, Tkinter...)
- Registro de partidas en fichero o BBDD
- Implementación por consola + GUI opcional
- Gestión web vía WebSocket o REST
- Sistema de emparejamiento automático entre clientes

## Cierre

 Esta actividad combina:

- Programación concurrente
  - Trabajo en red
  - Gestión de procesos e hilos
  - Control de recursos compartidos
- ✓ Simulación realista y aplicable a entornos profesionales