

SD	Sistemas Distribuidos
22/23	Práctica no guiada: Sockets, Streaming de Eventos, Colas y modularidad.
	AGAINST ALL

Preámbulo

El objetivo de esta práctica es que los estudiantes extiendan y afiancen sus conocimientos sobre el uso de la tecnología de comunicación básica sockets, estudiada durante las sesiones de teoría.

Los sockets son la base de las comunicaciones entre computadores y suponen el punto de enlace básico entre nuestra aplicación e Internet, utilizando muy pocos recursos de red. Como contrapartida, las aplicaciones que utilizan sockets como mecanismo de comunicación deben interpretar los mensajes que intercambian entre sí, es decir, deben establecer un protocolo o acuerdo de comunicación entre ellos. Esta necesidad implica un fuerte acoplamiento entre las aplicaciones distribuidas que utilizan sockets como mecanismo de comunicación, ya que todas las partes deben conocer de antemano la estructura de los mensajes que serán intercambiados y codificarlo en su programación.

Esta práctica establece el marco inicial de un desarrollo que se ampliará durante el cuatrimestre y que permitirá al estudiante crear una aplicación similar a las que se desarrollan hoy en día en los entornos empresariales, poniendo el foco en el uso e integración de distintos paradigmas de comunicación susceptibles de ser utilizados.

Especificación

El objetivo de la práctica a desarrollar es un sistema distribuido que implemente un juego multijugador online.

Descripción funcional

Los estudiantes deberán implementar un juego distribuido multijugador online denominado "AGAINST ALL".

El juego enfrenta a un número indeterminado de jugadores en un mapa de juego en el que se enfrentan todos contra todos.

Objetivo

El objetivo de cada jugador será vencer a todos los demás antes de ser eliminado.

Gana el juego aquel jugador que queda solo en el mapa de juego.

Mapa

El mapa de juego es una matriz de 20x20 posiciones. Cada posición del mapa (elemento de la matriz) puede contener una de las siguientes posibilidades:

1. Un Jugador: Representado por un elemento que indica su identidad y nivel
2. Un NPC (Non playable character): Representado por un elemento que indica su nivel
3. Una mina: Representado por una M
4. Un alimento: Representado por una A
5. Nada: Representado por un "espacio"

Los jugadores pueden ver en todo momento el mapa completo.

El mapa representa una geometría esférica de manera que las posiciones más al este (límite derecha del mapa) están conectadas con las situadas al oeste y viceversa, de la misma forma que las situadas en el límite norte (zona superior del mapa) conectan con el lado sur y viceversa.

Adicionalmente el mapa se divide en cuatro regiones de 10x10 donde cada región representa una ciudad del mundo. Al inicio de cada partida, el mapa se constituirá con cuatro ciudades escogidas al azar mediante un mecanismo que se describe en el apartado de diseño técnico.

En cada una de esas ciudades se registra un clima el cual alterará el devenir de la acción según se comenta en los siguientes apartados.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1			M		M			M					M			M				
2	A				A	A	A	A						M				M		M
3	M		M					M						A	A	A	A	A	A	
4						A	M										M	A		
5						M	M		A		A				A		M	M		M
6	M				A	M	M					M		M				M		A
7						M						A						M		A
8		A			A		M	M				A		M	A				A	A
9	M										A			M						A
10					M							A		A			A		A	
11											M			M						M
12			M				M	M	M							A		M	A	
13	M		M	M										M						M
14	M				A		M		A			A		M			A	M		
15	M		A		A								M			A	A	A		
16	A	M		M	A			M						M			A	A	A	A
17	A		A	A	A	A	A		A	A			A							
18	A			M						A		A			A			A	A	
19		A		M	M	A											A	A	A	A
20	M	M		A			A		M	M								A	A	

Figura 1: Ejemplo de mapa de juego

Jugadores

Los jugadores disponen de los siguientes atributos:

- Posición: Coordenadas del mapa en las que se encuentra: { int [1..20], int[1..20] }
- Alias: Hasta 20 caracteres. [0..10 | a..z | A..Z]
- Nivel: int[0..n]. Todos los jugadores empiezan con nivel 1
- Efecto ante el frío: +EF. int[-10..10]
- Efecto ante el calor: +EC. int[-10..10]

El movimiento que puede hacer en cada jugada es desplazarse a una de las posiciones adyacentes produciendo los siguientes efectos según lo que se encuentre en la posición destino:

- Otro jugador o NPC: Luchará contra él. Gana el jugador o NPC que tiene más nivel. El otro muere. Caso que tengan el mismo nivel empatan y siguen en la misma casilla.
- Mina: Muere.
- Alimento: Gana 1 punto de nivel.
- Nada: Nada.
- Cambia de ciudad (región climatológica):
 - o Efecto ante el frío (puntos de nivel):
 - +EF si $T^3 \leq 10^\circ\text{C}$
 - o Efecto ante el calor (puntos de nivel):
 - +EC si $T^3 \geq 25^\circ\text{C}$

Mecánica del juego

El juego comienza mediante la funcionalidad “Nueva Partida” que el estudiante deberá implementar en la aplicación del servidor central.

Los jugadores en ese momento pueden unirse a la partida hasta que se pulsa la opción “Comenzar partida” o se alcanza el máximo número de jugadores establecido al ejecutar la aplicación.

Cada jugador podrá decidir unirse a la partida mientras esté disponible dicha opción.

El servidor mostrará entonces el tablero de inicio de 20x20 en la que el valor de cada celda se obtiene aleatoriamente de entre los mencionados: M (Mina), A (Alimento), “[blanco]” (Nada).

De la misma forma, la posición de partida de los jugadores será calculada aleatoriamente.

La partida finaliza ante la ocurrencia de uno de los siguientes eventos:

- Cuando solo queda un jugador en el mapa proclamándose vencedor.
- Cuando se agota el tiempo de partida. El vencedor será el que tenga más nivel. En caso de mismo nivel se proclamará un empate.

La partida puede verse alterada por las condiciones meteorológicas de la zona donde se encuentre el jugador. Al arrancar la partida, el mapa se divide en 4 regiones de 10x10. Cada una de ellas refleja una ciudad del mundo al azar. El tiempo que haga en esa ciudad del mundo impactará en el nivel del jugador de acuerdo a su resistencia a los factores climatológicos.




Ejemplo:

- Se inicia una partida.
- Se inscriben en ella los siguientes jugadores:

	Nivel	EF	EC	Posición
Sir_Tom_Nook	1	-1	+6	14,4
Lady_Elsa	1	+10	-8	12,15
Sir_Balrog	1	-10	+10	5,3

- Se determina al azar que el mapa se divide en Londres , Alicante, Sidney y Wisconsin. Al consultar el tiempo en esas ciudades se ve lo siguiente:
 - Londres : 5°C
 - Alicante: 26°C
 - Sidney: 32°C
 - Wisconsin: -15°C
- Efectos del clima (valores al inicio de la partida):
 - Sir Tom Nook : Alicante. Nivel resultante: $1+6=7$
 - Lady Elsa: Wisconsin. Nivel resultante: $1+10=11$
 - Sir Balrog: Londres. Nivel resultante: $1-10=0$

AGAINST ALL

Jugador	Alias	Posición	Nivel
	Elsa	12,15	11
	Balrog	5,3	0
	Tom Nook	14,4	7




	Londres: 8º C										Alicante: 26º C									
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1			M		M			M					M			M				
2	A				A	A	A	A						M				M		M
3	M		M					M						A	A	A	A	A	A	
4						A	M										M	A		
5						M	M		A		A				A		M	M		M
6	M				A	M	M					M		M				M		A
7						M						A						M		A
8		A			A		M	M				A		M	A				A	A
9	M										A			M						A
10			8		M							A		A			A		A	
11											M			M						M
12			M				M	M	M							A		M	A	
13	M		M	M										M						M
14	M				A		M		A			A		M			A	M		
15	M		A		A							M				A	A	A		
16	A	M		M	A			M					M				A	A	A	A
17	A		A	A	A	A	A		A	A			A				10			
18	A			M						A		A			A			A	A	
19		A		M	M	A											A	A	A	A
20	M	M		A			A		M	M								A	A	
	Sidney: 32º C										Wisconsin: -15º C									

Figura 2: Situación de juego al inicio de la partida

La partida se va desarrollando de forma asíncrona, es decir, sin turnos, de manera que cada jugador decide moverse en cualquier momento.

Los jugadores pueden moverse en cualquier dirección: N,S,W,E,NW,NE,SW,SE

En cada movimiento se recalcula la posición y nivel del jugador. Si un jugador muere es eliminado del mapa y de la partida.

Si un jugador intenta moverse mientras la partida no se ha iniciado, el sistema le responderá con un mensaje de error a tal propósito: Partida no iniciada.

Diseño técnico

Se propone al estudiante implementar un sistema distribuido compuesto, al menos, de los siguientes componentes software:

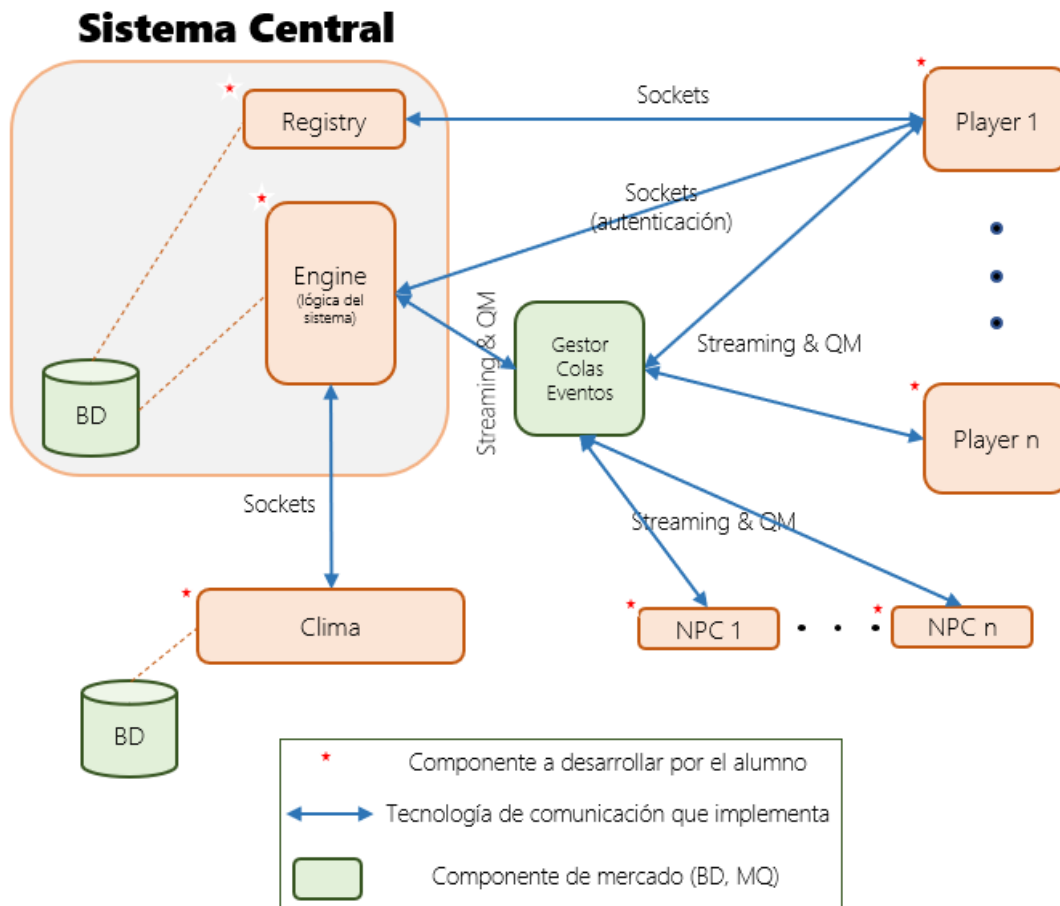


Figura 3. Esquema conceptual del Sistema software, interconexiones entre los componentes y tipos de interfaces.

Los componentes software que el estudiante ha de desarrollar son los siguientes:

- Núcleo: Módulos centrales de la solución
 - Registry: Módulo para la gestión de cuentas de usuarios
 - Engine: Módulo que implementa la lógica del juego
- Jugador: Aplicación de cliente que se conecta al núcleo
- Servidor Clima: Aplicación que devuelve la temperatura que hace en una determinada ciudad

Los componentes pueden ser desarrollados en el lenguaje de preferencia del estudiante: Java, C/C++, .NET, Python, etc. asumiendo el estudiante la responsabilidad de su conocimiento y forma de desplegarlo en el laboratorio.

A continuación, se especifica más detalladamente cada componente.

Núcleo

Contiene todos los módulos que implementan la estructura principal del juego.

Engine:

Se trata de la aplicación que implementará la lógica fundamental del juego. El nombre de la aplicación será **obligatoriamente “AA_Engine”**. Para su ejecución recibirá por la línea de parámetros los siguientes argumentos:

- Puerto de escucha
- Número máximo de jugadores
- Puerto AA_Weather

Al arrancar la aplicación quedará a la escucha en el puerto establecido.

Al iniciar una partida realizará 4 peticiones al servidor de clima para obtener la ciudad y temperatura que corresponde a cada cuadrante del mapa.

Para mayor sencillez de implementación, ante cada movimiento de cada jugador le responderá con todo el tablero.

Los estudiantes podrán decidir la forma de expresar el mapa, aunque se recomienda un array de bytes donde cada elemento de la matriz de bytes es una posición en del mapa.

Registry:

Se trata de la aplicación que implementará el registro de los jugadores en el Núcleo. El nombre de la aplicación será **obligatoriamente “AA_Registry”**. Para su ejecución recibirá por la línea de parámetros los siguientes argumentos:

- Puerto de escucha

La aplicación permanecerá a la espera hasta recibir una solicitud de registro de un jugador. Los datos que puede guardar de un jugador son:

- Alias
- Password
- Nivel
- EF
- EC

Base de Datos:

Contendrá el mapa de la partida en curso, los datos de los jugadores, sus credenciales de acceso y cualquier otra información que los estudiantes consideren necesaria para la implementación del juego.

Servirá de recurso compartido entre los distintos elementos que forman el Núcleo.

Los estudiantes podrán decidir el motor de BD a implementar, por ejemplo: SQLite, MySQL, SQLServer o MongoDB. **Igualmente será posible usar simples ficheros de texto para la implementación.**

Jugador

Aplicación que implementa la funcionalidad del jugador. El nombre de la aplicación será **obligatoriamente "AA_Player"**. Para su ejecución recibirá por la línea de parámetros los siguientes argumentos:

- IP y puerto del AA_Engine
- IP y puerto del Broker/Bootstrap-server del gestor de colas
- IP y puerto del AA_Registry

La aplicación tendrá las siguientes opciones:

- Crear perfil: Se conectará al módulo AA_Registry para crear un nuevo usuario.
- Editar perfil: Se conectará al módulo AA_Registry para editar o actualizar el perfil de un usuario existente.
- Unirse a la partida: Se solicitará el Alias y la contraseña del usuario. Una vez aceptado (autenticación por sockets en el Engine) en la partida el jugador podrá ir pulsando las teclas de movimiento que a tal propósito habrán implementado los estudiantes (comunicación por streaming). El mapa deberá ir mostrándose en pantalla conforme el jugador se mueve **con todos los detalles que muestren el estado de la partida**. Las respuestas por parte del servidor podrán ser:
 - El mapa actualizado
 - Partida no iniciada
 - Cualquier otro mensaje que el estudiante entienda necesario para implementar su práctica.

Nota: El estudiante deberá resolver cómo saber si sigue vivo un jugador.

NPC (non playable characters)

Aplicación que implementa la funcionalidad de un NPC manejado automáticamente por el sistema. El nombre de la aplicación será **obligatoriamente "AA_NPC"**. Para su ejecución recibirá por la línea de parámetros los siguientes argumentos:

- IP y puerto del Broker/Bootstrap-server del gestor de colas

El NPC podrá moverse en cualquier dirección aleatoriamente y se comporta de forma similar a un jugador. Su nivel permanece siempre estático. No le afectan las minas ni los alimentos ni el clima. No tiene front y caso que muera por enfrentamiento con otro jugador desaparece del tablero. **Se podrán instanciar tantos NPC como se deseen durante todo el desarrollo de la partida.**

Gestor de colas y streaming de eventos

Será implementado mediante la tecnología Apache Kafka.

Se implementarán los topics, productores y consumidores necesarios para resolver los requerimientos propuestos.

Servidor de clima

Aplicación que implementa la funcionalidad. El nombre de la aplicación será **obligatoriamente "AA_Weather"**. Para su ejecución recibirá por la línea de parámetros los siguientes argumentos:

- Puerto de escucha

Esta aplicación permanecerá a la escucha indefinidamente esperando a que la aplicación AA_Engine le pida una temperatura y una ciudad que esta devolverá de forma aleatoria.

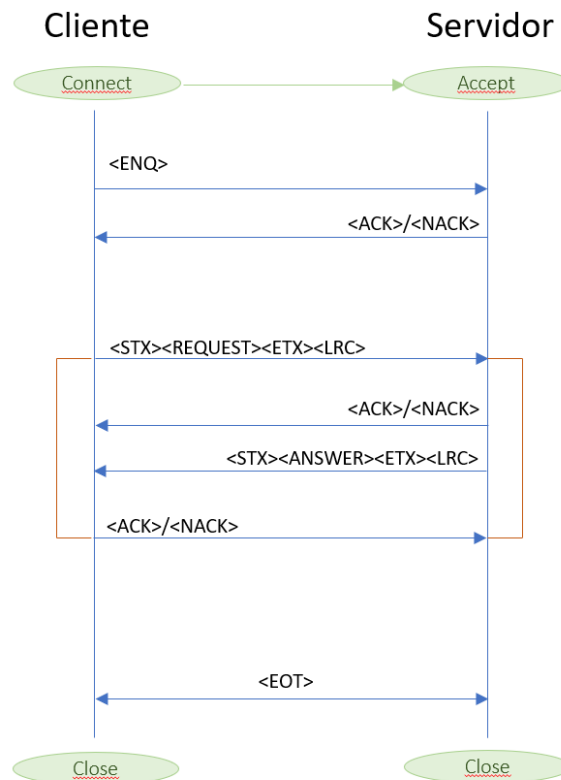
El servidor de clima generará la ciudad y la temperatura aleatoriamente de entre un conjunto que el estudiante definirá a su criterio, bien obteniéndolo de un archivo o simplemente cargándolo de una lista que el mismo defina.

La aplicación dispondrá de una BD (un fichero si así se desea) con ciudades del mundo y una temperatura indicada.

Nota: En la segunda parte de la práctica, este servidor de clima será sustituido por una consulta a un servidor real mediante las tecnologías que se indicarán en su momento.

Protocolo de comunicación (solo sockets)

Como se comenta al principio de este documento, la comunicación mediante sockets obliga a los actores involucrados a definir la mensajería y protocolo con precisión. Los estudiantes podrán definir los protocolos de comunicación entre los distintos módulos del sistema a su criterio. **Los profesores recomiendan, aunque no es obligatorio**, usar una mensajería basada en el estándar de empaquetado <STX><DATA><ETX><LRC> que corresponde al siguiente flujo:



Donde:

- <REQUEST> y <ANSWER>: Contienen el mensaje transmitido entre ambos puntos con los campos separados por un carácter determinado: Ej.: <REQUEST>: Código Operación#campo1#...#campo n
- <LRC>: Se define como el XOR(MESSAGE) byte a byte y sirve para validar que la transmisión del mensaje se ha realizado satisfactoriamente y ha sido recibida de forma correcta por el destinatario el cual responderá al mismo con un <ACK> o <NACK>

Aclaración final

Ante incongruencias funcionales o aspectos no considerados en los anteriores apartados que impidan la correcta implementación de la práctica, los estudiantes deberán implementar la alternativa que entiendan más conveniente previa consulta con el profesor para evitar un excesivo nivel de complejidad.

Guía mínima de despliegue

Para la correcta evaluación de la práctica es necesario comprobar que la aplicación distribuida solicitada es desplegada en un entorno verdaderamente distribuido. Por lo que para su revisión y prueba es necesario al menos 3 PCs distintos en los que se desplegarán los componentes solicitados. Al menos se ha de desplegar junto con el núcleo y el servidor de clima, 2 jugadores, proporcionando el siguiente escenario:

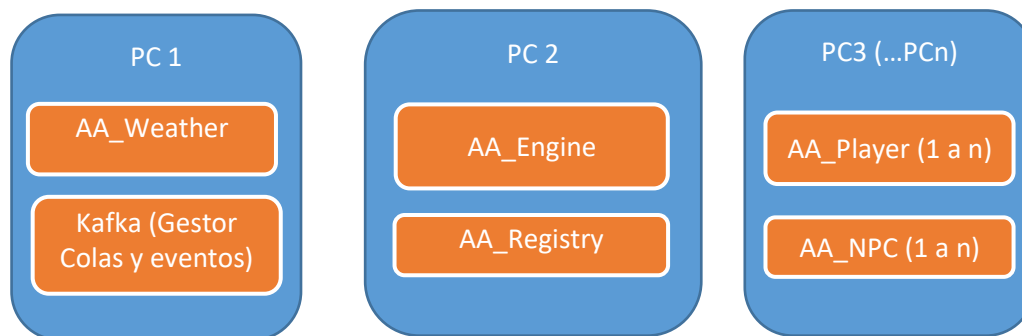


Figura 4. Escenario físico mínimo para el despliegue de la práctica.

Entregables y evaluación

La evaluación de la práctica se realizará en los laboratorios. **Se podrá realizar en grupos de hasta 2 personas sin perjuicio de que, durante el momento de la corrección, el profesor pueda preguntar a cualquier estudiante del grupo por cualquier aspecto de cualquiera de los módulos.** Los estudiantes deben desplegar por ellos mismos la práctica que resuelve el enunciado anterior. Deben desplegar un sistema completo, es decir, un núcleo, los jugadores y el servidor de clima todos ellos interconectados entre sí. **Este requisito es indispensable para poder realizar la corrección.** Además, deben poderse evaluar positiva o negativamente todos los apartados que aparecerán en la Guía de corrección que se entregará a tal propósito. Cada uno de los apartados puntúa de forma variable, por tanto, cada apartado no implementado o que no pueda comprobarse su correcto funcionamiento no podrá ser tenido en cuenta y por tanto no puntuará. Los estudiantes deberán presentar para la evaluación el documento **“Guía de corrección”** cumplimentado para que el profesor pueda validar los apartados implementados.

Los estudiantes deberán entregar, además, mediante la funcionalidad de evaluación del UACloud antes de la fecha establecida a su profesor de prácticas una **memoria de prácticas**, con el código fuente y compilados generados, así como un documento donde se detalle la siguiente información. El formato es libre, pero debe ser un documento ordenado y debidamente formateado, cuidando la redacción y ortografía.

- Portada con los nombres, apellidos y DNI de los estudiantes, año académico y el título de la práctica.
- Un informe donde se indique el nombre de los componentes software desarrollados y una descripción de cada uno de ellos, explicando y enviando además el código fuente de todos ellos.

- El detalle, paso a paso, de una guía de despliegue de la aplicación, que deberá ser la misma que utilice cuando haga la corrección de la práctica.
- Capturas de pantalla que muestren el funcionamiento de las distintas aplicaciones conectadas.

Cada profesor de prácticas podrá solicitar a los estudiantes cualquier otra evidencia que se considere adecuada para poder realizar la evaluación de forma adecuada.

La fecha oficial de entrega será la semana del 7/11/2022.