Prueba 2, Sistemas Distribuidos - Práctica 2025-01

- Debe ser realizado por 2 o 3 personas.
- Los grupos pueden escoger entre “Problemas A” y “Problemas B”
- En el caso que el conjunto de problemas sea desarrollado por 2 personas, deben realizar (A.1 y A2) o (B.1 y B.2)
- Fecha de entrega domingo 2025-06-11, esa fecha debe entregar el github y no debe realizar modificaciones (agreguenme para poder ver el sistema luis.veasc@gmail.com o dejenlo público). Luego concretamos una reunión de 30min. para la revisión, debe estar el grupo completo.
- Ambos conjuntos de sistemas deben contener sus respectivos archivos de configuración (sin variables en duro) y README de instalación.
- Este trabajo es obligatorio y equivale al 22% de la Prueba 2.
- Recuerde, todos son sistemas diferentes que establecen protocolos que les permiten comunicarse.

Problemas A

Problema 1 (HTTP), biblioteca digital distribuida por tipo de documento, este sistema debe ser implementado en python

Implemente mediante un esquema maestro-esclavo un buscador distribuido, el cual permite distribuir la carga de trabajo desde un nodo maestro a diferentes nodos esclavos los cuales deben implementar las búsquedas sobre una base de datos, la distribución de la base de datos debe ser realizada por tipo de documento digital presente en la biblioteca y por lo menos debe presentar 4 tipos de documentos, es decir, el esquema debe presentar por lo menos 4 esclavos. Las operaciones a implementar son:

(a) Buscar por título del documento digital, en el siguiente ejemplo estoy buscando “ecuaciones” y “diferenciales”. Aquí debe implementar una función tipo broadcast, es decir, el maestro recibe la consulta, realiza la consulta sobre todos los esclavos, luego recibe la respuesta desde los esclavos y entrega el resultado en un arreglo con los datos encontrados
http://localhost:xxx/query?titulo=ecuaciones+diferenciales

(b) búsqueda por tipo de documento, como el maestro conoce la distribución de la carga por tipo de documento, la consulta debe ir dirigida sólo a los esclavos que contienen la información, luego debe reunir las respuestas desde los esclavos y entregar el resultado en un arreglo con los datos encontrados
http://localhost:xxx/query?tipo_doc=tesis+libro+video

requisitos:

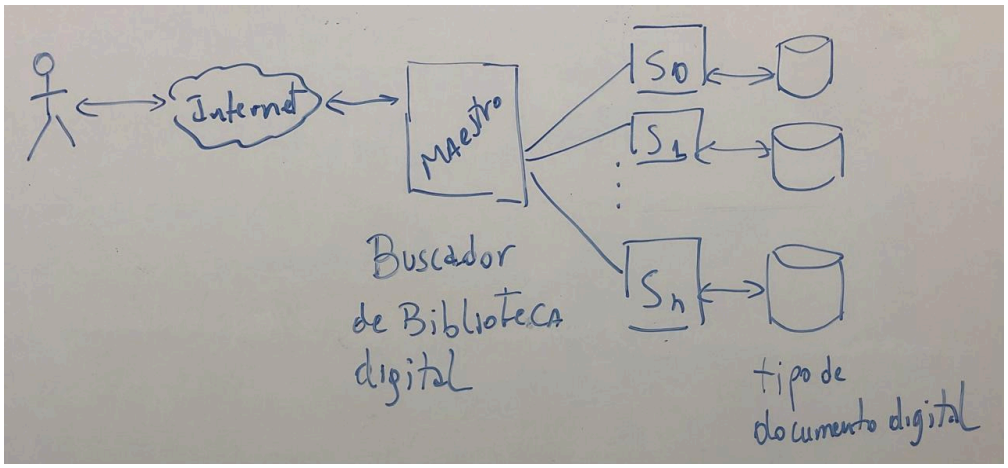
- El maestro conoce la distribución por tipo de documento alojadas en los sistemas esclavos.

- Esclavo tiene una copia de la base de datos con la porción de datos que le corresponda según la distribución por tipo de documento.

- En ambos mecanismos de búsqueda debe establecer una función que otorga un ranking, mientras más términos de la consulta encuentre más puntaje tiene. El resultado de la búsqueda debe ser presentado ordenado por el ranking.

- Debe establecer 3 tipos de usuario por rango etario y el resultado debe incorporar esta variable en el ranking, ejemplo: si el usuario tiene entre 10-15, debe dar mayor puntaje a los libros de tipo ciencia ficción.

- Debe presentar un archivo README.md con la especificación de lenguaje, frameworks usados y la forma de despliegue. Si usa archivo de configuración o variables de entorno debe explicar su contenido.



Problema 2, Crear log centralizado mediante RMI basada el buscador del problema A.1

- Debe crear una estructura de log que permita identificar tiempo de inicio y fin de cada operación interna ejecutada por cada nodo de la solución (maestro y esclavos).
- Debe Crear el README de instalación y despliegue para la prueba.

Ejemplo:

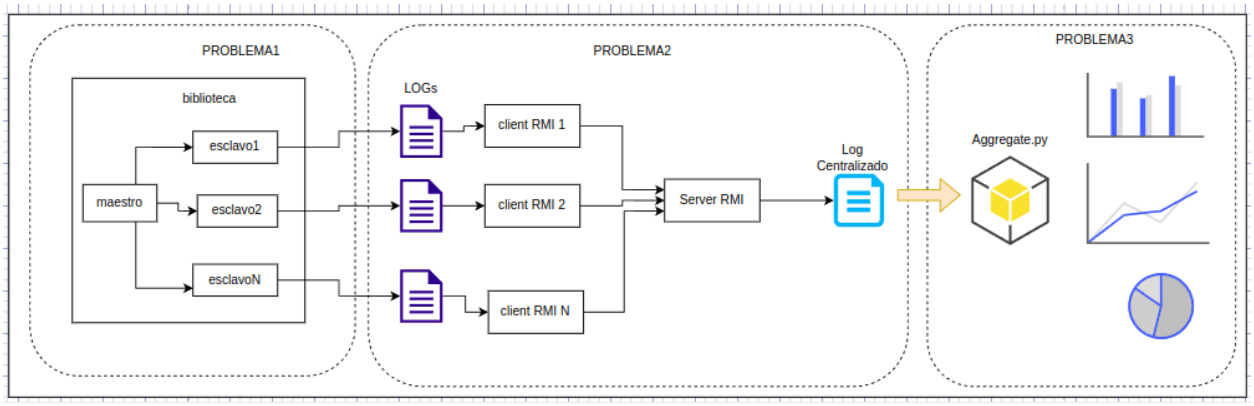
timestamp ini,timestamp fin, máquina, tipo de máquina, query búsqueda, tiempo fin, score obtenido, rango etario del cliente

cada log debe ser leído desde un cliente RMI y debe ser enviado a un Servidor RMI, el cual se encarga de crear un gran log centralizado donde se detalle la información de todas las búsquedas en todos los nodos de la solución

Problema 3, Crear gráficos estadísticos basados en el log centralizado del Problema A.2, utilizando Python

1. Gráfico de torta con porcentaje de consulta por rango etario.
2. Curvas de los promedios de score a través del tiempo, utilizar tamaños de ventana variable.
3. Gráfico de cajas destacando los tiempos promedio, min, max por esclavo.
4. Latencia de red entre el maestro y los esclavos
5. Tamaño en MB de las respuestas por hora a través del día, indicando el día.

Diagrama de contexto



PROBLEMAS B

Problema 1, juego basado en decisión por consenso del tipo p2p

El juego consiste en que dos o más equipos compiten por llegar a la meta en un tablero tipo fila de 100 posiciones, para lograrlo:

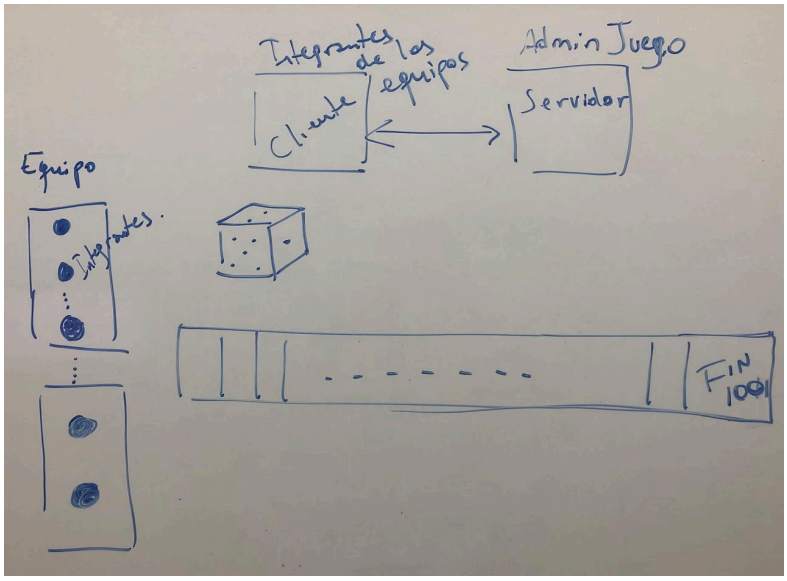
1. Cada integrante del equipo lanza un dado, el cual entrega una puntuación aleatoria de 1-6
2. Cada equipo tiene un turno de manera rotatoria

Requisitos:

- Debe presentar un archivo README.md con la especificación de lenguaje, frameworks usados y la forma de despliegue. Si usa archivo de configuración o variables de entorno debe explicar su contenido.
- La cantidad de equipo máxima es variable, según sea definida por el juego
- La cantidad de integrantes máxima por equipo es variable, según sea definida por el juego
- La cantidad máxima de posiciones es variables, es decir, pueden ser más o menos de 100 posiciones
- La cantidad mínima y máxima del dado es variable, según se defina en el juego
- Gana el primer equipo que llega o supera 100 (o el máximos de posiciones, ya que es variable)
- La puntuación de avance depende de la suma de los puntajes obtenidas por cada miembro del equipo al arrojar el dado
- Los jugadores del equipo deciden si aceptan o no a otro integrante en el equipo, es decir, todos los integrantes del equipo se conocen

Condiciones de la implementación, debe implementar el sistema bajo un esquema cliente-servidor, donde:

- El servidor tiene 2 equipos por defecto y permite que se inscriban más equipos.
- El servidor inicia el juego después de que ambos equipos solicitan el inicio del juego, es decir, el juego es asíncrono
- El servidor de forma aleatoria asigna el orden de juego de los equipos.
- El servidor finaliza el juego y notifica al ganador a todos los integrantes de los equipos, esto lo hace cuando el equipo en su conjunto superó las 100 posiciones del tablero (o máxima cantidad de posiciones, ya que es variable)
- El servidor recibe la petición de un participante de unirse a un equipo en el juego, informa al equipo en cuestión que hay un interesado en unirse. Los miembros del equipo deciden si aceptan o no, esto puede pasar en cualquier instante del juego.
- El juego parte cuando hay por lo menos un participante por equipo.



Problema 2, Crear log centralizado mediante RMI basada el el juego de tablero creado en la Probema B.1

Debe crear una estructura de log que permita identificar tiempo de inicio y fin de cada operación realizada sobre el juego

Ejemplo
timestamp(0), ini, juego1, inicio-juego
timestamp(i), ini, juego1, crea-jugador, equipo1, pepito
timestamp(i+1), fin, juego1, crea-jugador,equipo1, pepito
timestamp(i+2), ini, juego1, lanza-dado, equipo1, pepito, 5
timestamp(i+3), fin, juego1, lanza-dado, equipo1, pepito, 5

cada log debe ser leído desde un cliente RMI y debe ser enviado a un Servidor RMI, el cual se encarga de crear un gran log centralizado donde se detalle la información de todas las partidas realizadas en el juego

Problema 3, Crear gráficos estadísticos basados en el log centralizado del Problema B.2, utilizando Python

1. Jugadores creados por equipo en un juego.
2. Jugadas realizadas por jugador en un juego.
3. Curvas de puntuación por equipo a través del tiempo. El tamaño de la ventana debe ser una variable de entorno.
4. Equipos creados por ventanas de tiempo. El tamaño de la ventana debe ser una variable de entorno.

5. Jugadores creados por ventanas de tiempo. El tamaño de la ventana debe ser una variable de entorno.

Diagrama de contexto