



CÁTEDRA:

SISTEMAS DISTRIBUIDOS

ENUNCIADO DE TRABAJOS DE LABORATORIO

Año 2020

Versión 3.0

Docentes

Profesor Adjunto:	Lic. Cristian Javier Parise
Auxiliar de primera:	Lic. Pedro Konstantinoff

INDICE

TRABAJOS DE LABORATORIO	3
Objetivo general de los Trabajos Prácticos de Laboratorio.	3
Características	3
Forma de aprobación	3
Lista de Trabajos de Laboratorio	3
Trabajo de Laboratorio 1	4
Cliente / Servidor. Sockets. gRPC. Threads. Concurrencia. RFS.	4
Trabajo de Laboratorio 2	6
Java RMI. Concurrencia. Sincronización. Transacciones. Timestamping	6
Trabajo de Laboratorio 3	8
HTTP. HTML. CGI. AJAX. DFS. DNS	8
Trabajo de Laboratorio 4	10
Código Móvil. Comunicación Indirecta. MQTT. IoT	10

TRABAJOS DE LABORATORIO

Objetivo general de los Trabajos Prácticos de Laboratorio.

- Conocimiento y realización de trabajos sobre arquitectura Cliente / Servidor con diferentes tecnologías.
- Conocimiento de la problemática de Sincronización.
- Manejo de la generación de páginas WEB dinámicas.
- Manejo de Código móvil y Sistemas distribuidos de archivo.


Características

El Informe deberá reunir las siguientes características:

1. Cada Grupo presentará su informe a efectos de su calificado por el profesor. Los trabajos que no reúnan los requisitos mínimos serán devueltos para su corrección.
2. Deberá ser presentado a la cátedra confeccionado en grupos **no mayores de dos alumnos**, con discusión individual por alumno.
3. Para su preparación e impresión, el trabajo práctico deberá ser entregado de la siguiente forma:
 - En formato HTML o PDF, con un índice que refleje su estructura. Se incluirá una portada que deberá identificar a los integrantes del grupo y contener la firma de los mismos.
 - Toda la bibliografía utilizada deberá ser referenciada indicando título y autor, en una sección dedicada a tal efecto.
 - El programa de aplicación que implementa la solución.
 - El código fuente debe estar debidamente comentado. La solución debe ser desarrollada utilizando el lenguaje de programación indicado. También se debe incluir el makefile correspondiente o instrucciones o script para su correcta compilación, además del propio batch de prueba de ser necesario.

Forma de aprobación

Se tendrá en cuenta para la aprobación del trabajo práctico y los integrantes del grupo:

- Funcionamiento de la aplicación desarrollada. Se evaluará si la funcionalidad cumple con lo solicitado. En caso de que así no sea, el trabajo práctico se considerará desaprobado.
- Estructura general de la presentación, su legibilidad y facilidad de lectura y comprensión. 
- Contenido del informe y el uso de la información técnica para elaborarlo.
- Evaluación del grupo como un todo y a cada uno de sus integrantes.

Lista de Trabajos de Laboratorio

PRÁCTICA N° 1 – Cliente / Servidor. Sockets. gRPC. Threads. Concurrencia.

PRÁCTICA N° 2 – Java RMI. Concurrencia. Sincronización. Transacciones. Timestamping

PRÁCTICA N° 3 – HTTP. HTML. CGI. AJAX.

PRÁCTICA N° 4 – Código Móvil. DFS.

Trabajo de Laboratorio 2

Java RMI. Concurrencia. Sincronización. Transacciones. Timestamping

1. Basándose en el proyecto java rmisum, implemente dos objetos remotos con Java RMI, cada uno de ellos implementando funciones distintas (uno de ellos suma y resta y el otro multiplicación y división). El cliente debe tomar de la línea de comando tanto el indicador de operación como los operandos a utilizar. Valiéndose de un debugger y/o un analizador de protocolos, informar:
 - a) En cuáles puertos atiende cada uno de los objetos remotos de los servidores en una determinada ejecución? Cambian si se vuelve a lanzar?.
 - b)Cuál es el puerto en el que atiende RMIRegistry?.
 - c) Identificar el/los mensajes que transfiere el cliente con RMIRegistry.
 - d) Los mensajes hacia RMI Registry se valen de TCP o UDP.?
2. Responda las siguientes cuestiones:
 - a) Cuáles son los problemas que tiene un servidor concurrente?
 - b) Qué mecanismos se tienen disponibles con Java RMI?
 - c) Podría considerar que RPC y RMI tienen el mismo nivel de abstracción? Justifique.
 - d) Identifique similitudes y diferencias entre RPC y Java RMI.
 - e) Determine si el servidor concurrente con Java RMI tiene un thread por requerimiento, por conexión o por recurso (en términos del cap. 6 de Coulouris).
3. Adaptando el proyecto del ejercicio 1 elabore escenarios donde pueda verificarse:
 - a) Si RMI otorga un servicio concurrente en forma automática para el acceso a objetos remotos (puede agregarse un delay a alguna operación)
 - b) Que puede establecerse un timeout para el objeto que hace un RMI?. (forzar la demora con el delay hecho en punto a)
 - c) Si las variables de instancia de un determinado objeto remoto, no se “contaminan” con sus equivalentes de otras instancias del mismo objeto a las que se está accediendo en forma concurrente (acceso sincronizado desde dos clientes).
 - d) Que cuando se pasan objetos remotos como parámetros el pasaje es por referencia, contrariamente de cuando se pasan objetos locales (por valor) .
4. Implemente un servidor de hora empleando Java RMI o Python. Desarrollar un cliente que emule un reloj (utilizar hilos para la actualización del mismo) y que se sincronice dicho reloj con el servidor de hora implementado, utilizando el algoritmo de Cristian. Al utilizar la hora patrón, tome un grupo de valores (por ejemplo 5), eligiendo el que haya demandado menor tiempo de tránsito de los mensajes.
5. Desarrolle una aplicación que se comuniquen con un servidor NTP de internet y que extraiga 8 muestras del retardo con que correspondería ajustar la hora obtenida. Presentar en pantalla una tabla con cada muestra del tiempo obtenido del servidor en formato YYYY/MM/DD HH:MM:SS.mmm en relación con cada retardo. Utilizar la librería NTPLib de Python (<https://pypi.org/project/ntplib/>) o las clases Java NTPClient y NTPMessage provistas en la carpeta Material U4
6. Transacciones Distribuidas:Explicar cómo el protocolo de compromiso de dos fases para transacciones anidadas, se asegura que si la transacción de nivel superior se compromete, todas las descendientes se comprometen o se abortan.
7. Timestamping: Obtener un sello de tiempo de un archivo de texto con contenido conocido,verificarlo y por último alterarlo e intentar volver a verificarlo. Documentar el proceso utilizando la GUI de dos infraestructuras:

- Nacional: bfa.ar - Para el procedimiento deberán obtener un recibo digital temporario *provisorio* (extensión rd.temp) del hash del archivo, luego esperar a que el archivo sea ingresado en un bloque Blockchain y por último obtener el recibo digital temporario *definitivo* (extensión rd)

- Externa: opentimestamps.org - Para el procedimiento deberán obtener un recibo (extensión ots) y luego verificarlo pero se debe esperar a que el hash del archivo forma parte de un bloque Blockchain (mientras dará el mensaje Pending attestation)

Intentar (y documentar) un sellado también por linea de comando

\$ pip3 install opentimestamps-client

\$ ots stamp my-file