UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC y LSC	2003-1	5038	Programación Orientada a Objetos II

PRÁCTICA No.	LABORATORIO DE		DURACIÓN (HORA)
4	NOMBRE DE LA PRÁCTICA	Sockets de servidor	4

1 INTRODUCCIÓN

Un socket de Internet, socket de red o simplemente socket, es una herramienta para permitir la comunicación bidireccional entre dos computadoras conectadas en una red basada en IP. En este contexto, el socket es una abstracción de software en la cual se combina una dirección IP con un número de puerto en una sola entidad. Así, el socket se convierte en una interfaz entre un proceso o hilo de una aplicación y la pila de protocolos IP proporcionada por el sistema operativo, siendo el primer paso en el establecimiento del flujo de datos entre una aplicación con otro proceso o servicio.

En Java, las clases de sockets se utilizan para representar la conexión entre aplicaciones funcionando bajo una arquitectura cliente-servidor. El paquete java.net proporciona dos clases – Socket y ServerSocket – que implementan el lado del cliente y el lado del servidor en la conexión, respectivamente.

2 OBJETIVO (COMPETENCIA)

Crear una aplicación que utilice un servicio de la red para obtener información y representarla de una manera adecuada.

Formuló	Revisó	Aprobó	Autorizó
M.C. Jorge Eduardo Ibarra Esquer	M.C. Gloria Etelbina Chávez Valenzuela y LSC Mónica Lam Mora		M.C. Miguel Ángel Martínez Romero
Maestro	Coordinador de carrera	Gestión de la Calidad	Director de la Facultad

3 FUNDAMENTO

Un socket de Internet es identificado por el sistema operativo como una combinación de los siguientes elementos:

- Protocolo (TCP, UDP, etc.)
- Dirección IP local
- Número de puerto local

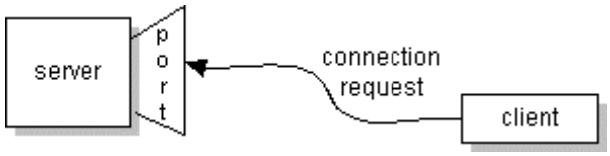
Cuando el socket ya se ha establecido, también se incluyen:

- Dirección IP remota
- Número de puerto remoto

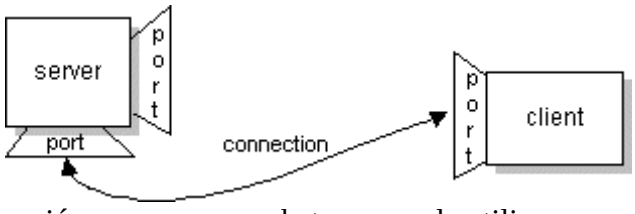
El sistema operativo se encarga de redirigir la información recibida a través del socket hacia el proceso correspondiente.

Por medio de sockets, podemos escribir aplicaciones basadas en el modelo cliente-servidor. Normalmente, un servidor se ejecuta en una computadora específica y tiene un socket vinculado a un puerto específico. El servidor simplemente espera, permitiendo que el socket "escuche" las solicitudes de conexión de los clientes.

En el lado del cliente, éste conoce la dirección de la computadora en que se ejecuta el servidor y el número del puerto al que se encuentra conectado. Para solicitar una conexión, el cliente trata de encontrarse con el servidor en su computadora y puerto.



Si todo funciona correctamente, el servidor acepta la conexión. Una vez aceptada, éste abre un nuevo socket en un puerto diferente. Se necesita un nuevo socket (y, en consecuencia, un número de puerto distinto) de manera que el socket original pueda seguir recibiendo solicitudes de conexión de los clientes.



En el cliente, si se acepta la conexión, se crea un socket que puede utilizarse para comunicarse con el servidor. A este socket se le asigna un puerto local en la computadora en la que se ejecuta el cliente, el cual no necesariamente es el mismo número de puerto que el del socket del servidor.

En este momento, el cliente y el servidor pueden comunicarse escribiendo y leyendo a través de sus sockets.

La edición estándar de Java proporciona algunas clases para trabajar con sockets. Estas clases se encuentran en el paquete java.net. La clase *java.net.ServerSocket* nos permite designar y abrir un puerto a través del cual aceptar conexiones desde otros equipos en una red. Esto se hace con la llamada a uno de los constructores de la clase. Por ejemplo, el constructor que tiene la declaración:

```
public ServerSocket(int port);
```

recibe como único argumento el número de puerto que utilizará para esperar solicitudes de conexión. Por ejemplo, si queremos abrir un *ServerSocket* para aceptar conexiones en el puerto 54321, lo hacemos de la siguiente forma:

```
ServerSocket ss=new ServerSocket(54321);
```

Una vez abierto el puerto, se pueden aceptar las solicitudes de conexión mediante el método *accept()*, el cual regresa una referencia a un *Socket*, mismo que se utilizará para la comunicación.

```
Socket s=ss.accept();
```

El método *accept()* bloquea la ejecución del programa mientras espera una solicitud de conexión. Una vez obtenida la referencia al *Socket*, se obtienen de éste sus flujos de E/S y se inicia el intercambio de información.

4 PROCEDIMIENTO (DESCRIPCIÓN)	
A EQUIPO NECESARIO	**MATERIAL DE APOYO**

Computadoras con una versión reciente del Java Development Kit JDK y acceso a Internet.

B DESARROLLO DE LA PRÁCTICA

1. Escribir una aplicación en Java que proporcione un servicio de localización de ciudades dentro de México, utilizando sockets TCP.
    a. Definir el protocolo que se utilizará para el servicio. Ver los anexos para un ejemplo.
    b. Al iniciar la ejecución del servidor, éste abrirá un ServerSocket en un puerto dado y esperará las conexiones de los clientes.
    c. Al conectarse un cliente, el servidor enviará un mensaje de bienvenida a través del flujo de salida asociado al Socket.
    d. Por cada comando que el cliente envía, deberá recibir la respuesta adecuada de parte del servidor.
    e. La conexión termina cuando el cliente envíe el comando de salida.
2. Los comandos que se utilizarán durante la sesión son:
    a. Mostrar ayuda: Se presentará una descripción breve de los comandos y su sintaxis.
    b. Buscar ciudad: Se enviará el nombre de la ciudad o población y el servidor dará como respuesta el estado donde ésta se encuentre. En caso de que se encuentren ciudades con el mismo nombre en más de un estado, se enviará primero un número indicando el total de resultados encontrados y después la lista de resultados.
    c. Agregar o modificar ciudad. Se añadirá una nueva localidad o se modificará una existente.
    d. Salir. Terminar la interacción con el servidor.
    e. Ver los anexos para un ejemplo de conexión.
3. Se programará únicamente el servidor. Como cliente se puede utilizar el comando telnet.
4. El servidor deberá aceptar conexiones simultáneas de varios clientes. Se recomienda utilizar el modelo “hilo por conexión” para manejo de concurrencia.
5. Recuerda que no es necesario incluir una interfaz gráfica para el servidor.
6. La información de ciudades y estados se almacenará en un archivo.

C	CÁLCULOS Y REPORTE

Se verificará el funcionamiento adecuado de la aplicación y se solicitará un reporte con el código de la misma.

5 RESULTADOS Y CONCLUSIONES

El alumno será capaz de elaborar aplicaciones que hagan uso de diversos servicios de red.

6 ANEXOS

Ejemplo de protocolo para la localización de ciudades.

- Comando ENCUENTRA

  Se utiliza para buscar la ubicación de una población. Su sintaxis es como sigue:

  ```
  ENCUENTRA nombreCiudad
  ```

  La respuesta puede ser alguna de las siguientes:

  ```
  NombreCiudad está en NombreEstado
  ```

  ```
  n
  NombreCiudad – NombreEstado1
  NombreCiudad – NombreEstado2
  …
  NombreCiudad – NombreEstadoN
  ```

  ```
  Ciudad no encontrada.
  ```

- Comando AGREGA

  Se utiliza para agregar información a la lista. Su sintaxis es la siguiente:

  ```
  AGREGA nombreCiudad - nombreEstado
  ```

  La respuesta para este comando puede ser:

  ```
  Información agregada con éxito.
  ```

  ```
  Error: Información duplicada
  ```

- Comando MODIFICA

  Se utiliza para modificar la información de la lista. Su sintaxis es la siguiente:

  ```
  MODIFICA nombreCiudad nombreEstado nombreCiudadNuevo nombreEstadoNuevo
  ```

La respuesta puede ser:

```
Información modificada con éxito.
```

```
Error: Combinación de ciudad/estado no encontrada.
```

- Comando SALIR

  Se utiliza para salir de la aplicación. Su sintaxis es:

  ```
  SALIR
  ```

  La respuesta a este comando es:

  ```
  Hasta luego.
  ```

Un ejemplo de uso de la aplicación, suponiendo que el servidor se ejecuta en la computadora local y en el puerto 54321, sería el siguiente:

```
telnet localhost 54321
Bienvenido al localizador de comunidades.
ENCUENTRA Mexicali
Mexicali está en Baja California.
ENCUENTRA San Felipe
Ciudad no encontrada.
AGREGA San Felipe – Baja California
Información agregada con éxito.
ENCUENTRA San Felipe
San Felipe está en Baja California.
ENCUENTRA Matamoros
2
Matamoros está en Coahuila.
Matamoros está en Tamaulipas.
SALIR
Hasta luego.
```