Programación Avanzada

– Curso 2009 / 2010 –

JavaTrack :: SISTEMA DE SEGUIMIENTO ERRORES SOFTWARE

COMENTARIOS

- La práctica es obligatoria e individual.
- La práctica se valorará sobre 10 puntos. Para superar la práctica, será necesario obtener una calificación mayor o igual a 5 puntos.
- Las prácticas *dudosas* deberán ser defendidas ante el profesor. Se habilitará un aula y un horario para la defensa.
- Se deberá entregar documentación impresa que incluirá:
  - **Portada**: indicando el nombre, apellidos, DNI y grupo de prácticas al que se pertenece.
  - **Índice**: especificando el número de página de cada apartado.
  - **Manual de usuario**: comentando cómo debe ejecutarse la aplicación, cuáles son sus opciones y un ejemplo de uso de cada una de ellas. SE DEBERÁN DOCUMENTAR EN UN PUNTO SEPARADO LAS AMPLIACIONES REALIZADAS.
  - **Manual del programador** (*comandos básicos*): comentando aspectos destacados de la aplicación como clases y métodos de especial relevancia, cuestiones técnicas, etc.
  - **Manual del programador** (*ampliación de comandos*): comentando cómo se ha llevado a cabo la implementación de las ampliaciones realizadas.
  - **Código fuente**: de las aplicaciones desarrolladas.
- Se deberá acompañar la documentación impresa junto con el programa en CD (incluyendo código fuente) organizado según la siguiente estructura de directorios:
  D:\----
  - **alumno.txt** (nombre, apellidos, DNI, grupo de prácticas, repetidor: SÍ/NO).
  - **notas.txt** (cualquier aclaración sobre la práctica que se desee hacer constar para su correcta ejecución: versión del JDK, etc.).
  - **proa0910.pdf** (fichero de documentación).
  - <DIR> **cJavaTrack** (ficheros del cliente de JavaTrack).
  - <DIR> **sJavaTrack** (ficheros del servidor de JavaTrack).
  - <DIR> **sBD** (ficheros del servidor de base de datos).
  - <DIR> **dTEC** (documentación técnica: ficheros generados con *javadoc*).

OBJETIVO

Un sistema de gestión de errores software tiene como objetivo coordinar a un equipo de programadores ante la aparición de errores durante el desarrollo de uno o varios proyectos software. La presente práctica consiste en programar *JavaTrack*, un pequeño sistema para este fin, que debe ser distribuido, hacer uso de bases de datos y permitir atender a varios usuarios simultáneamente.

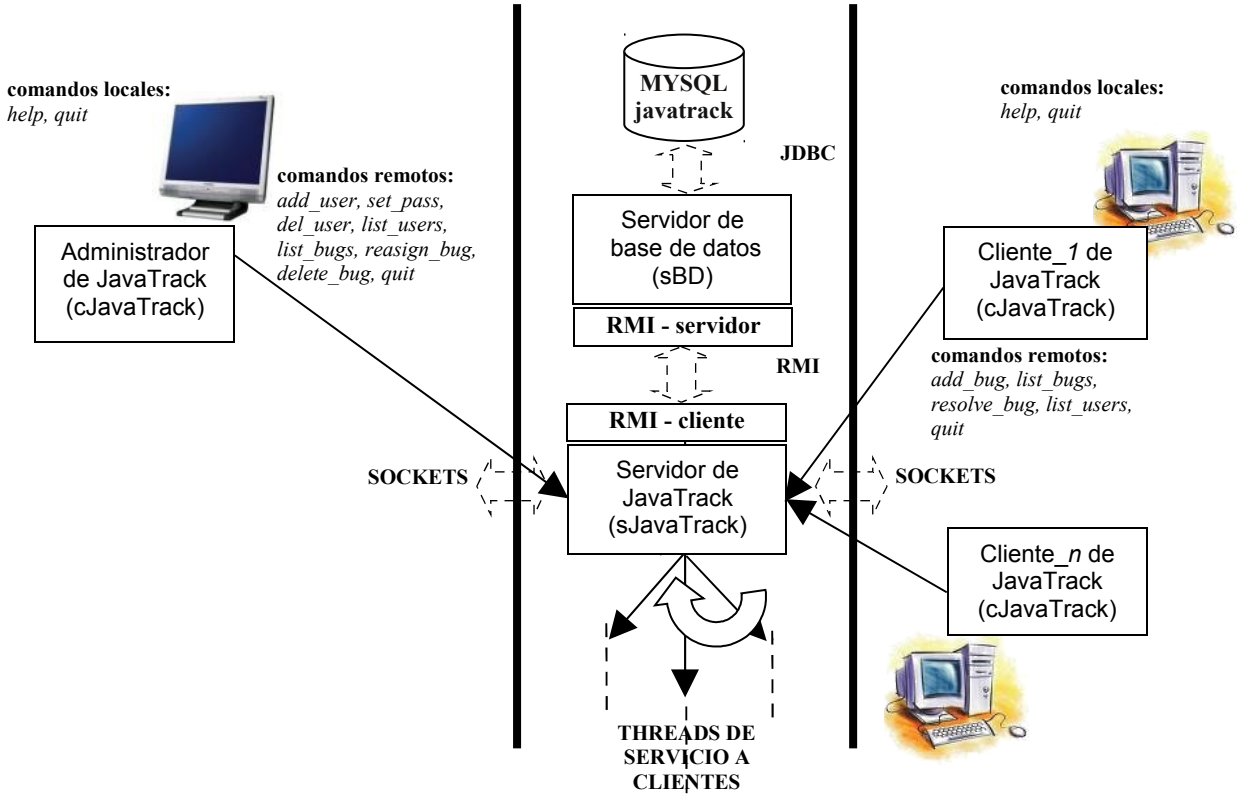
Cuando se detecta un error (incidencia o bug) en algún programa por parte de un programador (usuario del sistema a desarrollar), éste se deberá conectar al sistema *JavaTrack* para comunicar dicha incidencia, que será registrada con el estado inicial de "*PENDIENTE*". Para ello, el usuario deberá indicar el nombre de la aplicación donde se ha detectado el error, una breve descripción del problema encontrado y el nombre del programador que deberá solucionar ese error. Los programadores se conectarán regularmente al sistema para ver qué errores tienen asignados y, una vez que consideren que lo han solucionado, deberán cambiar el estado a "*RESUELTO*".

Para implementar la funcionalidad anteriormente descrita, se deberá desarrollar un conjunto de tres aplicaciones (*i*) **cliente de JavaTrack** (*ii*) **servidor de JavaTrack** y (*iii*) **servidor de BD** para posibilitar el envío, asignación y gestión de errores en el sistema.

La ejecución de los programas en modo consola será como sigue:

java sBD
java sJavaTrack <*puerto_del_servidor*>
java cJavaTrack <*direccion_ip_del_servidor*> <*puerto_del_servidor*>

ARQUITECTURA DEL SERVIDOR DE JAVATRACK



REQUISITOS MÍNIMOS

El servidor **sJavaTrack** debe ser no bloqueante, se implementará utilizando programación multihilo y deberá dar servicio a dos tipos de usuarios: (*i*) **clientes de JavaTrack**, que podrán enviar y visualizar errores registrados en el sistema y (*ii*) **administradores**, encargados de la gestión de usuarios y el mantenimiento del sistema. En un momento determinado podrán estar conectados al mismo tiempo múltiples clientes y/o administradores. Ambos tipos de usuarios utilizarán la misma aplicación para interaccionar con el sistema: **cJavaTrack**.

COMANDOS DE **cJavaTrack** PARA LOS CLIENTES DE JAVATRACK:

La aplicación permitirá a los usuarios registrados acceder al servidor de JavaTrack (**sJavaTrack**) y ejecutar los siguientes comandos:

add_bug <*aplicacion*> <*descripcion*> <*login_programador*>
Inserta en la base de datos **javatrack** el bug proporcionado que deberá ser atendido por el programador especificado como tercer parámetro. Además, para cada incidencia se almacenará en la base de datos un código de la incidencia (generado automáticamente por el servidor), el programador que ha identificado el error (usuario conectado que envía el error) y la fecha/hora en la que fue añadida dicha incidencia.

list_bugs [<*login_programador*>]
Produce un listado en el cliente de todas las incidencias registradas. Si se especifica el parámetro <*login_programador*>, el listado sólo muestra las incidencias asignadas al programdor cuyo login sea <*login_programador*>. El listado deberá incluir todos los campos almacenados para cada incidencia en la base de datos. En cualquier caso, las incidencias más antiguas serán listadas primero.

resolve_bug <*codigo_bug*> <*mensaje*>
Cambia el estado de la incidencia <*codigo_bug*> a "*RESUELTO*" añadiendo un <*mensaje*> explicativo de dicha resolución. Si la incidencia no estaba asignada al usuario que solicita esta operación, se le denegará.

list_users
Produce un listado en el cliente de todos los usuarios registrados en sistema que no son administradores.

help [<*comando*>]
Si no se especifica el parámetro <*comando*>, devuelve un listado de comandos disponibles en **sJavaTrack**. Si se especifica el parámetro <*comando*>, proporciona ayuda ampliada sobre la utilización del comando especificado.

COMANDOS DE **cJavaTrack** PARA LOS ADMINISTRADORES:

La aplicación permitirá a los administradores del sistema la realización de las siguientes funcionalidades:

add_user <*login_usuario*> <*password_usuario*> <*¿es_administrador?*>
Inserta en la base de datos **javatrack** los datos de autentificación de un nuevo usuario. Si el tercer parámetro toma el valor "1" el usuario es un administrador, en caso contrario (su valor es "0") se trata de un cliente de JavaTrack.

set_pass <*login_usuario*> <*new_password_usuario* >
Cambia en la base de datos **javatrack** la password de acceso de un usuario existente en el sistema.

del_user <*login_usuario*>
Elimina de la base de datos **javatrack** el usuario especificado. Si existen incidencias cuyo autor es este usuario también deberán ser eliminados de la base de datos. Si existen incidencias asignadas al programador que se va a eliminar, deberán quedar sin asignar a nadie hasta que el administrador la reasigne.

list_users
Produce un listado en el cliente de todos los usuarios registrados en sistema.

list_bugs < *login_programador* >
Produce un listado en el cliente de todas las incidencias registradas. Si se especifica el parámetro <*login_programador*>, el listado sólo muestra las incidencias que ese usuario tiene asignadas. El listado deberá incluir todos los campos almacenados para cada incidencia en la base de datos. En cualquier caso, las incidencias más antiguas serán listadas primero.

reasign_bug <*codigo_bug*> <*login_programador*>
Cambia el responsable de atender a una incidencia. La incidencia <*codigo_bug*> deberá quedar asignada al programador cuyo login sea <*login_programador*>.

delete_bug <*codigo_bug*>
Elimina la incidencia <*codigo_bug*> de la base de datos.

help [<*comando*>]
Si no se especifica el parámetro <*comando*>, devuelve un listado de comandos disponibles en **sJavaTrack** para el administrador. Si se especifica el parámetro <*comando*>, proporciona ayuda ampliada sobre la utilización del comando especificado.

Los clientes y los administradores del sistema deberán autentificarse utilizando **cJavaTrack** antes de poder ejecutar cualquiera de los comandos que requieran una interacción con el servidor. En concreto, el comando a ejecutar por ambos una vez arrancada la aplicación **cJavaTrack** es el siguiente:

login <*login_usuario*> <*password_usuario*>
Comprueba si en la base de datos **javatrack** existe una entrada en la tabla **usuarios** con el <*login_usuario*> y la <*password_usuario*> suministrados.

Con el fin de poder finalizar la interacción con el servidor **sJavaTrack** de forma ordenada, se deberá implementar el siguiente comando para ambos usuarios:

quit
Termina la sesión con el usuario actual. No detiene el servidor **sJavaTrack**.

El servidor **sJavaTrack** se comunicará mediante RMI con el servidor de base de datos (**sBD**) que mantendrá una base de datos utilizando JDBC y mySQL. El nombre de la base de datos será **javatrack** y tendrá la siguiente estructura:

```
-- MySQL dump 8.22
-- Host: localhost    Database: javatrack


CREATE TABLE  bugs (
  codigo integer(10),
  aplicacion char(255),
  descripcion char(255),
  autor char(10),
  programador char(10),
  estado char(10),
  descripcion_resolucion char(255),
  fecha char(17)
);

CREATE TABLE  usuarios (
  login char(10),
  password char(10),
  administrador char(1)
);
```

El campo **fecha** de la tabla **bugs** almacenará la fecha y la hora en la que el usuario registró la incidencia. El formato a utilizar será el siguiente: **aa/mm/dd hh:mm:ss**.

FICHEROS DE SOPORTE

Para facilitar la realización de la práctica, se han desarrollado un conjunto de ficheros base que codifican el esqueleto de un servidor multihilo no bloqueante. Estos ficheros son los siguientes:

- **cJavaTrack.java**: cliente socket.
- **sJavaTrack.java**: servidor socket multihilo no bloqueante.

CALIFICACIÓN

La calificación de la práctica que implemente los requisitos mínimos será de 6 puntos. Cada ampliación de las comentadas en el punto siguiente suma 1 punto más.

POSIBLES AMPLIACIONES

Se valorará positivamente (incremento de la nota base) cualquier ampliación a los requisitos mínimos especificados. Las posibles ampliaciones a realizar pasan por la implementación de nuevos comandos para los administradores del sistema:

(+1 punto)

enable_user <*login_usuario*> <*¿habilitado?*>

Habilita/deshabilita la posibilidad de acceso al sistema para el usuario especificado. Si el segundo parámetro toma el valor "1" el usuario está habilitado para poder hacer login en el sistema, en caso contrario (su valor es "0") el usuario no podrá acceder al sistema.

Para realizar esta ampliación se deberá hacer uso del campo **habilitado char(1)** existente en la tabla **usuarios** con el fin de almacenar el estado de cada usuario. Por defecto, cuando se crea un nuevo usuario su valor debe ser "1" (el usuario está habilitado).

(+1 punto)

save_bugs [<*login_programador*>]

Crea el fichero de texto "bugs.txt" en el cliente. Este fichero contendrá todos los bugs existentes en la base de datos. Si se especifica el parámetro <*login_programador*>, el fichero de texto generado en el cliente sólo contendrá los bugs asignados al programador cuyo login sea <*login_programador*>.

(+1 punto)

reasign_bugs <*programador_origen*> <*programador_destino*>

Reasigna todas las incidencias del programador cuyo login es <*programador_origen*> al programador cuyo login sea <*programador_destino*>.

(+1 punto)

list_bugs <*fecha_inicio*> <*fecha_fin*>

Produce un listado en el cliente de todas las incidencias registradas entre las dos fechas proporcionadas (en formato aa/mm/dd).

CONSIDERACIONES ADICIONALES

- No es necesario entregar la base de datos (**javatrack**) ni el controlador JDBC para mySQL, puesto que estarán correctamente instalados y configurados en el ordenador destinado a la evaluación de las prácticas.
- Las prácticas que no respeten la estructura de directorios del disquete NO SERÁN EVALUADAS.
- Las prácticas deben ser totalmente funcionales. Aquellas prácticas que no puedan ser ejecutadas siguiendo las instrucciones proporcionadas en este documento NO SERÁN EVALUADAS.
- La fecha tope para la entrega de prácticas vendrá establecida por el día y la hora del examen teórico de la asignatura en cada convocatoria.