Universitat Autònoma
de Barcelona

Simulador de Rendimiento Stress Tester

Memoria del proyecto

de Ingeniería Técnica en

Informática de Sistemas

realizado por

Albert Moreso Ventura

y dirigido por

Jordi Pons Aróztegui

Escola Universitària d'Informàtica

Sabadell, Junio de 2009

El abajo firmante, Jordi Pons Aróztegui,
profesor de l'Escola Universitària d'Informàtica de la UAB,

CERTIFICA:

Que el trabajo al que corresponde la presente
memoria ha sido realizado bajo su supervisión
por Albert Moreso Ventura

I para que conste firma la presente.
Sabadell, Junio de 2009

Firmado: Jordi Pons Aróztegui

El abajo firmante, Joan Gemio Treserras,
de CCSAgresso,

CERTIFICA:

Que el trabajo al que corresponde la presente memoria
ha sido realizado bajo su supervisión
por Albert Moreso Ventura

I para que conste firma la presente.
Sabadell, Junio de 2009

Firmado: Joan Gemio Treserras

Índice

Índice de figuras

Resumen

El presente documento corresponde a la memoria del Proyecto de Final de Carrera en Ingeniería Técnica de Informática de Sistemas, en la Escola Universitària de Informática de Sabadell.

El proyecto ha sido realizado en una empresa dedicada al desarrollo de software, mediante el convenio de prácticas existente entre esta empresa, CCSAgresso (anteriormente conocida como Centro de Cálculo de Sabadell) y la Universitat Autónoma de Barcelona.

Este convenio permite al estudiante, y responsable del proyecto, incorporarse a la empresa para realizar el proyecto desarrollando una aplicación real y útil para la empresa.

La aplicación llevada a cabo tiene como tema principal la evaluación del rendimiento en una máquina, mediante los datos aportados por pruebas de simulación.

Cada prueba de simulación está contenida en un fichero de texto, de diferentes formatos según cuál sea el programa con el cual se haya realizado la simulación.

Los dos formatos más importantes son los ficheros de texto procedentes de simulaciones Karat y de simulaciones del Performance Monitor de Windows. Además de estos dos formatos, se ha conseguido que la aplicación también reconozca ficheros de texto procedentes de un sistema operativo Linux.

La lectura de estos ficheros de texto y su almacenamiento en la base de datos son la base de la aplicación. También son el proceso más complicado, ya que se realiza mediante código Java.

La aplicación ha sido implementada mediante Karat 8.0, una herramienta software para desarrollar aplicaciones que tienen como origen de valores una base de datos, y como destino la interacción total con el usuario, gestionando y visualizando esos valores de origen.

Los objetivos son poder introducir, actualizar, gestionar y mostrar esos valores de la base de datos, con una aplicación sencilla, intuitiva y que realice todas las operaciones que necesita el usuario.

1. Introducción

El presente documento es la memoria correspondiente a la aplicación "Simulador de Rendimiento Stress Tester", proyecto final de carrera de Ingeniería Técnica en Informática de Sistemas en la Escola Universitària d´Informàtica de Sabadell , perteneciente a la Universitat Autónoma de Barcelona.

Se trata de un proyecto desarrollado dentro del marco del convenio de colaboración entre la UAB y la empresa CCSAgresso (antiguamente conocida como Centro de Cálculo de Sabadell), dentro del departamento de Fábrica, concretamente en el Área de Testing.

Testing es un área dedicada a la simulación y análisis de resultados, especialmente de aplicaciones implementadas mediante Karat y otras herramientas de desarrollo software propias de la empresa CCSAgresso.

El principal objetivo es el de proporcionar beneficios a las dos partes: el estudiante tiene la posibilidad de comprobar el funcionamiento, trabajar y llevar a cabo un proyecto que, tras pasar las correspondientes validaciones y pruebas, pueda ser una aplicación eficiente y útil para la empresa.

El proyecto tiene una duración estimada de 500 horas, con la posibilidad de ampliarlo a un máximo de 550 horas.

1.1 Presentación

El proyecto consiste en una aplicación con el principal objetivo de procesar, gestionar y visualizar los datos numéricos de una simulación sobre una máquina.

Está desarrollada mediante el entorno Karat, un entorno de desarrollo software propio de la empresa CCSAgresso, principalmente enfocado al diseño de aplicaciones que trabajan en interacción con una base de datos.

Además, la nueva versión Karat 8.0, está basada en tecnología Java, y la codificación se llevará a cabo dentro de la plataforma Eclipse JDK, de libre distribución.

Tiene que permitir recoger y tratar los datos de las simulaciones, así como hacer el análisis y presentación de los resultados de una máquina, de manera que puedan ser almacenados en una base de datos.

La aplicación Stress Tester será la interfaz que permita acceder a esos valores guardados en la base de datos y permita la interacción del usuario con todos los valores numéricos de una prueba de simulación.

Las funcionalidades que ha de tener esta aplicación de simulación son las siguientes:

- Guardar y gestionar los datos de rendimiento de una maquina
- Analizar los resultados de rendimiento, mostrando gráficos de rendimiento
- Mostrar los resultados numéricos de las simulaciones, los resultados generales y también los resultados específicos de las pruebas de simulación.

Para poder llevarlo a cabo, se necesita que la aplicación pueda extraer los datos de simulación de la actividad del número de usuarios deseado, de tal manera que se puedan obtener y gestionar los parámetros de rendimiento en una maquina.

1.2 Ámbito de trabajo

La empresa Centro de Cálculo de Sabadell, actualmente conocida como CCSAgresso, es una compañía que posee una larga historia que se remonta al año 1963. Se dedica a la fabricación y distribución de productos y soluciones de Tecnologías de la información.

Algunos ejemplos son los ERPs (planificación de recursos empresariales, en inglés Enterprise resource planning) y las soluciones a medida, así como los servicios asociados que las empresas requieren (mantenimiento, interacción comercial, soporte, consultoría,...)

La plantilla está formada por aproximadamente 500 profesionales, y la empresa cuenta con oficinas centrales en Barberá del Vallés (Barcelona), además de delegaciones en Madrid, Valencia, Zaragoza y San Sebastián.

El departamento donde se ha llevado a cabo el proyecto es el de Desarrollo de Software, también llamado Fábrica dentro de la empresa.

Es el departamento desde el cual se controla todo el proceso de desarrollo, mantenimiento y documentación de las diferentes aplicaciones software dentro del catálogo de la empresa, especialmente las Ottools, herramientas para la creación de aplicaciones que pueden abarcar muchos ámbitos.

Además desde Fábrica también se mantiene una importante interacción con el departamento de soporte a clientes.

1.3 Planteamiento inicial

Los datos de la prueba de simulación pueden tener su origen en tres fuentes distintas:

- **Ficheros de texto con simulaciones de rendimiento llevadas a cabo con el Monitor de Rendimiento del sistema de Windows (Performance monitor).**

  Para utilizar el Performance monitor de Windows, basta con ejecutar el comando <perfmon> desde la ventana de Inicio →Ejecutar.

  El fichero de texto resultado tiene un formato muy determinado, y especialmente pensado para ser almacenado en una base de datos.

  Además existe una librería estándar del lenguaje Java, llamada CsvReader, que tiene funciones especificas para la lectura de este tipo de fichero.

  Los datos están organizados en columnas, y cada columna está delimitada por un carácter.

  El delimitador puede estar entre dos caracteres, dando lugar a dos tipos de fichero según cuál sea su delimitador de columnas:

- El formato más habitual es el que tiene la coma , el carácter "," , como delimitador (extensión *.csv)

- También puede tener como delimitador el tabulador (extensión *.tsv)

- **Ficheros de texto con simulaciones Karat, procedentes del programa Karat Simulator.**

  El fichero de texto obtenido del Karat Simulator tiene un formato diferente respecto del resultante del Performance Monitor de Windows.

  Los datos también están organizados en columnas, pero en este caso no existe ningún delimitador entre las diferentes columnas de datos.

  El programa Karat Simulator es el que permite especificar todas las opciones deseadas y establecer la Simulación de las actividades y eventos (llevadas a cabo por el cliente que se quiere simular).

- **Ficheros de texto con simulaciones procedentes del comando <sar> de Unix**

  Este tipo de fichero no estaba en la planificación inicial, fue una propuesta de ampliación de funcionalidades en caso de poderla llevar a cabo.
  Se planteó como una posible mejora, en caso de disponer de suficiente tiempo.

  La propuesta era la de realizar también la obtención de los datos de un fichero de texto procedente de algún programa de monitorización de rendimiento de un sistema operativo de libre distribución, preferentemente Linux, y con intérprete de comandos Unix.

  Para ello se optó por la utilización del comando <sar> (System Activity Report), ya que de los comandos analizados, era el único que permitía guardar los resultados de la monitorización en un fichero de texto.

  La opción más preferible, a la hora de almacenar los datos contenidos en el fichero de texto, era la de intentar aprovechar alguna de las tablas ya creadas anteriormente.

  Observando los datos del fichero de texto, resultaba factible utilizar la tabla st_lineas_pm, que tiene como función guardar los datos de los ficheros del Performance monitor de Windows.

  Los campos que se tienen que guardar son exactamente los mismos, a excepción del nombre del equipo, ya que en el fichero procedente del comando <sar> no se especifica el nombre de la máquina.

  En este caso, el formulario inicial de selección de fichero de texto <st_datos_sar>, tiene que incluir un campo donde se pregunte al usuario por el nombre del equipo al cual pertenece la simulación contenida en el fichero de texto. Este campo es obligatorio.

La máquina en cuestión puede ser de tipo:

- **Cliente**: Parámetros de rendimiento de un PC a nivel local, conectado o no en red. Medirá el rendimiento de las diferentes acciones llevadas a cabo por clientes a nivel de aplicaciones Karat (interacción con aplicaciones diseñadas en el entorno Karat: gestión de tablas, consultas, objetos de negocio, formularios, listados,...).

  Además también se podrá observar los valores de rendimiento específicos de la máquina, como pueden ser el tiempo de procesador, la memoria utilizada y el tráfico de red.

- **Servidor**: Parámetros de rendimiento de una máquina Servidor y administrador de red, simulando el rendimiento de una maquina de tipo Servidor o de tipo Cliente que estén conectados a una red informática.

  En este caso, se medirán parámetros de rendimiento que pueden ser dirigidos a monitorizar la actividad en red de una maquina.

En la simulación del Servidor, los parámetros a monitorizar estarán enfocados a evaluar la respuesta del Servidor según la variación de la carga de trabajo y el número de usuarios.

El hecho de poder medir esos parámetros permitirá establecer unos valores de funcionamiento óptimo del Servidor en situaciones críticas.

Además aportará una valiosa información respecto a cuál es el número máximo de usuarios soportado con una determinada carga de trabajo y dimensionar con exactitud la capacidad de funcionamiento de un Servidor en situaciones críticas

1.4 Objetivos del proyecto

El objetivo principal es llevar a cabo una mejora en la gestión, presentación y automatización de los datos de salida, como resultado de una simulación realizada desde el Monitor del Sistema de Windows o con el comando <sar> de Unix, así como si se trata de una simulación Karat .

Una vez procesados los datos de la simulación, que son el contenido de un fichero de texto, el siguiente paso es el de posibilitar que estos datos sean almacenados en una base de datos.

Los datos resultantes estarán disponibles para ser gestionados y mostrados por pantalla, en diferentes modalidades y formatos, según las opciones escogidas por el usuario.

El objetivo más importante es el de ofrecer la posibilidad de realizar diferentes pruebas de carga sobre una máquina, para poder observar la evolución de los diferentes parámetros de rendimiento.

En una prueba de carga de trabajo a nivel local, el resultado que se deberá analizar estará mucho más enfocado en la parte Cliente.

En este caso, los parámetros de rendimiento más importantes serán:

- Actividad de la CPU.
- Actividad de la Memoria RAM (Random Acces Memory, es la memoria temporal del ordenador).
- Tráfico de Red que afecta a ese cliente en concreto (Bytes enviados y recibidos por segundo por parte del cliente).

En el caso de las simulaciones Karat son llevadas a cabo mediante el programa Karat Simulator (descrito en el apartado 3.1 de este documento). Los parámetros de rendimiento que se monitorizan en este caso hacen referencia a los diferentes eventos que pueden tener lugar en la ejecución de una aplicación Karat.

Algunos de estos parámetros son:

- Tiempo en Abrir Formulario (en milisegundos)
- Tiempo de Conexión de usuario (en milisegundos)
- Tiempo de Duración de un evento (en milisegundos)

En una prueba de carga de trabajo sobre un servidor, resulta muy importante la monitorización de todas las actividades en la red, tanto por parte del Servidor como por parte de los Clientes.

Además del tráfico de red también se podrán observar los parámetros de rendimiento previamente detallados, en una máquina Servidor, en caso de que se desee monitorizar el rendimiento específico del Servidor.

La finalidad es delimitar y comprender la causa de que tengan un determinado rendimiento en cada momento de la realización de las pruebas, y cuáles son las variaciones en los parámetros de rendimiento.

1.5 Estructura de la memoria y del anexo

El presente documento ha sido estructurado de tal manera que los capítulos pudiesen reflejar la misma evolución cronológica que se ha seguido para llevar a cabo el proyecto.

La única excepción es el Capítulo 2, donde se describen las herramientas software empleadas.

En circunstancias normales, el capítulo del Estudio de Viabilidad es la fase correcta para realizar una búsqueda e investigación de cuáles son las herramientas software más adecuadas, según las necesidades de la aplicación.

En este caso, no hizo falta buscar esas herramientas, debido a que ya estaban disponibles en la empresa y fueron instaladas ya el primer día en que se empezó el proyecto.

El anexo contiene todo el conjunto de documentación que se ha generado, así como otros materiales adicionales:

Carpeta Documentación:

- Documento de Especificación de Requerimientos.
- Documento del Estudio de Viabilidad.
- Documento de Análisis y Diseño.
- Documento de Pruebas unitarias y ampliación de funcionalidades.
- Memoria del Proyecto Simulador de Rendimiento Stress Tester.

Otros materiales:

- Archivo StressTester.jar. Archivo empaquetado con todos los componentes java.
- Archivos *.java. Archivos con el código fuente, en lenguaje de programación java, que son utilizados por la aplicación.
  En la carpeta **código fuente.**
- Documentación en formato html generada mediante la utilidad Javadoc del programa Eclipse JDK.
  En la carpeta **Javadoc**.
- Archivos de texto con pruebas de simulación, utilizados para las pruebas unitarias sobre la aplicación.
  En la carpeta **karat_SIM**: Ficheros de texto de simulaciones Karat.
  En la carpeta **Perflog_SIM**: Ficheros de texto de simulaciones del Performance Monitor de Windows.
  En la carpeta **SAR_SIM**: Ficheros de texto de simulaciones del comando <sar> de Unix.

- Videos con las demostraciones de funcionamiento de la aplicación Simulador de Rendimiento Stress Tester.
  En la carpeta **Videos.**

2. Herramientas software

2.1 Introducción

En un proyecto informático cualquiera existe una labor de investigación y estudio, alrededor de cuál es la mejor alternativa en software. La opción ideal es aquella que permita afrontar el problema que se quiere solucionar de la mejor manera posible, y desarrollar una aplicación con la mayor eficiencia y el mínimo coste.

El hecho de que este proyecto se haya realizado en empresa ha supuesto que, ya desde el primer día de la realización de la aplicación, todos los programas y diferentes utilidades software necesarias estuviesen totalmente disponibles para poder ser instalados.

La razón es sencilla: son programas que tienen licencia dentro de la empresa, y que están totalmente disponibles para ser utilizados legalmente, dentro de la empresa.

2.2 Java

- Java Development Kit (JDK)
- Java Runtime Environment (JRE)
- Maquina Virtual Java (JVM)

JRE (o Entorno en Tiempo de Ejecución de Java) es el software necesario para ejecutar cualquier aplicación desarrollada para la plataforma Java. El usuario final usa el JRE como parte de paquetes software o plug-ins (o conectores) en un navegador Web.

La compañía Sun ofrece también el SDK de Java 2, o JDK (Java Development Kit) en cuyo seno reside el JRE, e incluye herramientas como el compilador de Java, Javadoc para generar documentación o el depurador de errores (Debugger).

Puede también obtenerse como un paquete independiente, y puede considerarse como el entorno necesario para ejecutar una aplicación Java, mientras que un desarrollador debe además contar con otras facilidades que ofrece el JDK.

2.3 Eclipse JDK

Eclipse es un entorno de desarrollo de software, basado en lenguaje Java, de libre distribución y con una potencia similar a la que pueden ofrecer entornos como los de Microsoft Visual C o Microsoft Visual Basic.

Es un Entorno Integrado de Desarrollo (IDE) abierto y extensible, está escrito en su mayor parte en Java (salvo el núcleo), se ejecuta sobre una máquina virtual de ésta y su uso más popular es como un IDE para Java. A pesar de este hecho, eclipse es neutral y adaptable a cualquier tipo de lenguaje.

La característica clave de eclipse es la extensibilidad. Eclipse es una gran estructura formada por un núcleo y muchos plug-ins que van conformando la funcionalidad final.

El IDE Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la plataforma de cliente, a diferencia de otros entornos monolíticos donde la funcionalidad esta toda incluida, la necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software.

2.4 Apache Tomcat

Programa de gestión de un Servidor Web, está implementado en Java, y por ello necesita la máquina Virtual de Java (JVM) para ejecutarse, funciona como un contenedor de servlets.

Un servlet es un programa que se ejecuta en un servidor y extiende su funcionalidad (similar a un applet de Java, pero para servidores). Su aplicación más común es generar páginas web de forma dinámica con los parámetros de la petición enviada por el navegador web

2.5 Karat Studio y Karat Escritorio

Ottools (Open Technology Tools) es un conjunto de herramientas desarrolladas por CCS que permiten la creación de aplicaciones multiidioma y multipais.

Se trata de un conjunto de componentes que proporcionan un entorno de desarrollo y ejecución de aplicaciones de gestión.

Dispone de herramientas de productividad para el usuario final (asistente de listados, asistente de estadísticas, gestión documental,...) y herramientas para la instalación, localización, personalización y administración.

Además, proporciona un conjunto de servicios como workflow, seguridad, auditorías, etc...

El sistema en tiempo de ejecución es un conjunto de componentes pre construidos que proporcionan al desarrollador o al usuario final la funcionalidad básica del sistema.

Todos los componentes son objetos DCOM implementados en el lenguaje más adecuado en cada caso (C++, Java, VB, etc...) que los desarrolladores utilizan para crear sus aplicaciones.

Son herramientas destinadas a funcionar en un entorno Cliente/Servidor. Este tipo de entorno implica una arquitectura basada en una parte servidora instalada en un equipo central y un conjunto de pc's donde se deben instalar los componentes clientes de Ottools.

El usuario final dispone de dos herramientas karat®Escritorio y karat®Studio que le permiten interactuar con la aplicación:

- Karat Studio, herramienta de Desarrollo
  Karat 7.5, última versión en funcionamiento, basada en lenguaje Visual Basic

- Karat Escritorio, herramienta de Interacción y Ejecución de Aplicaciones
  Karat 8, versión en proceso de desarrollo, basada en lenguaje Java

Se trata del único entorno disponible en Ottools que permite la administración y/o el desarrollo de aplicaciones Karat. Requiere un cierto grado de configuración y administración en cada uno de los equipos cliente.

Karat es un entorno basado en entorno web pero enfocado enteramente para la Maquina Virtual Java de Sun. Este entorno permite ser utilizado a nivel cliente en plataformas Windows, Linux y Macintosh.

Este tipo de entorno ofrece otra filosofía de trabajo a través de lo que se conoce como Dashboard (Cuadro de mandos) que permite la definición de miniaplicaciones (Widgets). Requiere la instalación, configuración y administración de un Microsoft Internet Information Server.

2.6 Microsoft Sql Server 2005

Programa de gestión de sistema de bases de datos, tanto para máquinas Servidor como para máquinas Cliente, es la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, Sybase ASE, PostgreSQL, Interbase, Firebird o MySQL.

Sus características más importantes son:

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se ubican en el servidor y las terminales o clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos.

Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños, que en sus versiones 2005 y 2008 pasa a ser el SQL Express Edition, que se distribuye de forma gratuita.

Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas .NET, pero el servidor sólo está disponible para Sistemas Operativos Windows.

3. Requerimientos

3.1 Sistema actual

En la actualidad el Karat Simulator ofrece la realización de simulaciones sobre una maquina, con diferentes opciones que se han de definir previamente para establecer los diferentes parámetros de la simulación:

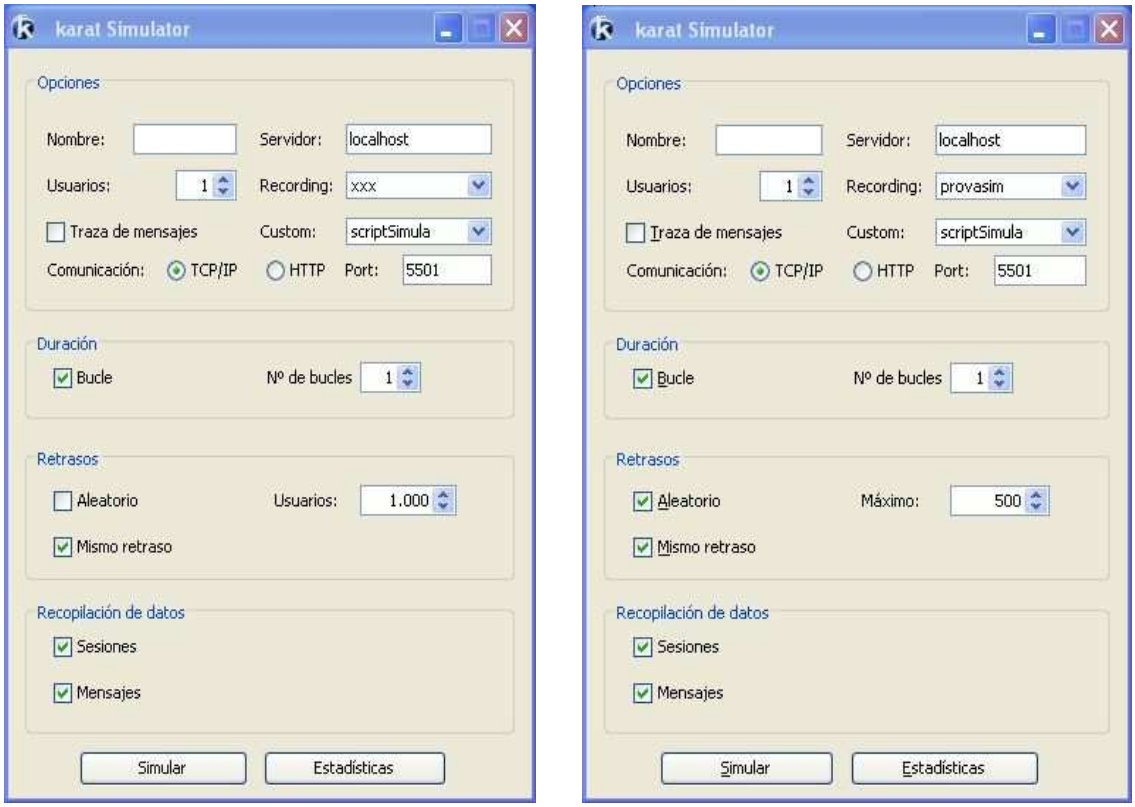

Figura 1: Capturas de pantalla del programa Karat Simulator

OPCIONES

Nombre: Nombre que se le asigna al archivo de simulación

Servidor: Servidor de red en el cual se quiere hacer la simulación, por defecto tiene el valor del servidor introducido en la última simulación.

El nombre del servidor ha de ser el nombre válido de un servidor existente y en ejecución, para especificar la maquina local se debe introducir <localhost>.

Usuarios: Número de usuarios que formarán parte de la simulación

Recording: Es la grabación de las sucesivas acciones realizadas por un usuario que han

de ser simuladas.

Una vez se le da un nombre nuevo a la grabación, se abre la aplicación Karat sobre la cual se quiere simular, de tal manera que todas las acciones del usuario quedan registradas en la simulación.

Custom: Es el nombre del script, implementado en lenguaje Java, que permite realizar cualquier acción sobre la simulación que necesite ser definida por código.

Mediante código Java se pueden definir nuevos métodos dentro del script para llevar a cabo cualquier funcionalidad deseada dentro de la simulación.

Traza de mensajes: Esta opción habilita, a la derecha de la ventana de resultados, de todos los mensajes intermedios referentes a la evolución en el tiempo de la simulación y los usuarios implicados en ella.

Comunicación: Esta opción permite elegir entre los dos protocolos mediante los cuales se comunicarán los resultados, estos son TCP/IP o HTTP, y el puerto por el cual serán comunicados.

El puerto está asignado por defecto, el 5501 para TCP/IP y el 8080 para el HTTP.

DURACION

Bucle: Esta opción permite la repetición, en ciclo cerrado, de la simulación definida.

Para indicar el número de iteraciones de la repetición se ha de indicar en la opción Nº de bucles.

RETRASOS

Aleatorio: Si se activa esta opción, se aplica un retardo aleatorio a cada acción realizada por el usuario.

El retardo tendrá como valor máximo de retraso el que se indique en la opción adyacente, Máximo.

Mismo retraso: Esta opción sirve para indicar que la simulación seguirá los mismos tiempos que el usuario ha realizado durante la simulación.

La opción adyacente Usuarios es el retraso entre usuarios en milisegundos (ms)

RECOPILACION DE DATOS

Sesiones: Graba todos los datos de la simulación ofreciendo la posibilidad de visualizarlos separados por usuarios (cada sesión corresponde a un usuario diferente).

Mensajes: Grabación de los mensajes de la traza de mensajes

Para obtener más información de una Simulación, o de un Recording (grabación de las acciones llevadas a cabo y que se desean simular) se realiza desde la pantalla de Estadísticas:

Pantalla estadísticas

Pestaña recordings

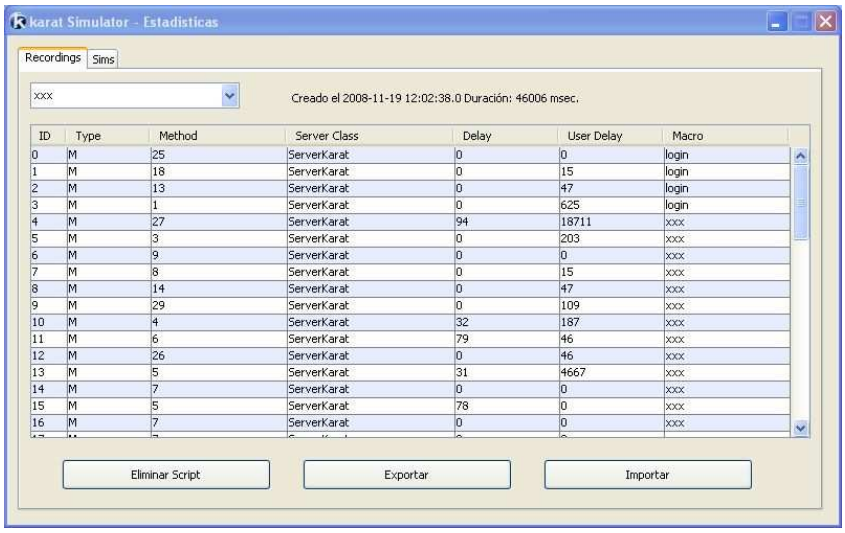

Figura 2: Captura de pantalla de la opción Recordings, del programa Karat Simulator

Pestaña Sims

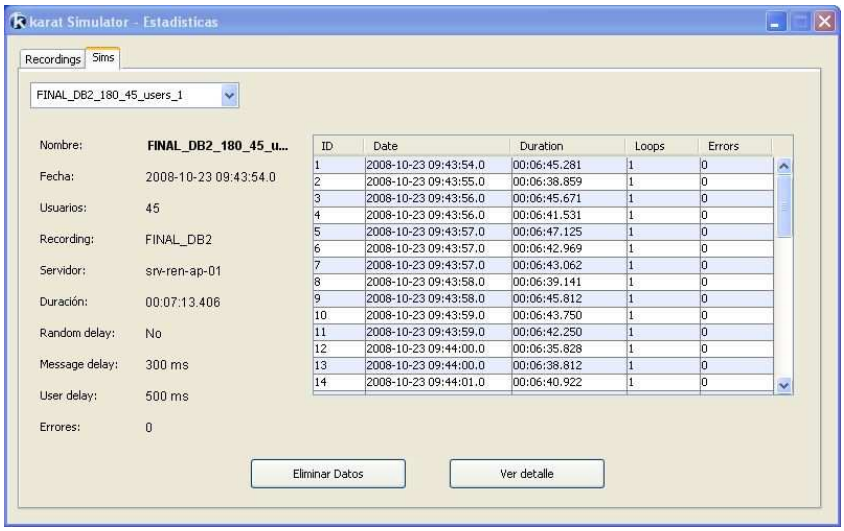

Figura 3: Captura de pantalla de la opción Sims, del programa Karat Simulator

El programa Karat Simulator es el que permite especificar todas las opciones deseadas y establecer la Simulación de las actividades y eventos (llevadas a cabo por el Cliente que se quiere Simular).

El Karat Simulator es un programa que ya posee las funcionalidades necesarias para realizar las simulaciones Karat, pero no incorpora la automatización de los datos de salida obtenidos en la simulación.

Todo ese proceso de automatización es el que tiene que llevar a cabo el Simulador de Rendimiento Stress Tester que se implementará en este proyecto.

Las dos grandes novedades que se han incorporado en el Karat Simulator 8.0 son:

- Grabación de simulaciones complejas. Permite generar una simulación juntando partes de otras simulaciones, mediante la herramienta karat Script Composer.

- Código Java en un script. Al ejecutar una simulación, también se puede ejecutar código java. Con esto se permite modificar el comportamiento de una simulación, por ejemplo grabar un registro con un valor distinto en cada repetición de la simulación.

Existe una pantalla de estadísticas, pero para poder obtener los datos la única manera es con sentencias SQL que permitan extraer los datos de las tablas almacenadas, mediante el Karat Simulator, en las tablas lógicas del repositorio (creado con las herramientas Karat).

Actualmente para obtener los datos de una de simulación, se incorporan los contadores del parámetro (Memoria, Procesador, Trafico de red,…) y la característica concreta que se desea medir (memoria disponible, tiempo de cpu, tiempo inactividad, bytes enviados, bytes recibidos,...) al programa Monitor de Rendimiento del Sistema de Windows (se accede desde Inicio-Ejecutar- perfmon.exe).

Una vez los contadores están agregados, se guardan los datos de salida en el formato de archivo separado por tabuladores (*.tsv), ya que es un formato reconocido como hoja de cálculo, y los datos son guardados en un archivo de Microsoft Excel.

En resumen, el proceso actual implica intervención manual sobre los datos obtenidos en la simulación, circunstancia que puede inducir a errores y que aumenta notablemente el tiempo necesario para ejecutar y obtener los resultados de una simulación.

Una vez establecida la Simulación deseada en el Karat Simulator, se debe ejecutar al mismo tiempo la Simulación desde el Monitor del Sistema de Windows, introduciendo los parámetros a monitorizar y configurando el Registro de Rendimiento:

- Maquina que se quiere simular

- Contadores que se quieren monitorizar (son los parámetros de rendimiento que se quieren medir)

- Ubicación deseada para el archivo de Salida (por defecto es la carpeta C:\Perflogs, en el directorio raíz de la unidad C:).

  Además existe la opción de finalizar el nombre de archivo con un sufijo de números

secuenciales o una fecha, para permitir el mantenimiento de varios archivos de registro. Esta funcionalidad resulta especialmente útil en archivos de registro generados automáticamente con el mismo nombre de archivo.

- Tipo de archivo de salida deseado para los registros de contador (donde están definidos los parámetros de archivo de los registros de seguimiento).

El tipo de archivo predeterminado para los registros de contador es el Archivo binario (con la extensión .BLG), y el tipo de archivo predeterminado para los registros de seguimiento es el Archivo de seguimiento secuencial (con la extensión .ETL).

Existen diferentes opciones para el formato del archivo de salida:

- Archivo de texto (CSV-Comma Separated Values) y Archivo de texto (TSV- Tab separated Values): Estos dos formatos se usan para exportar datos a un programa de hoja de cálculo. Los datos se almacenan como un archivo de registro delimitado por comas (con la extensión de archivo *.csv) o por tabuladores (con extensión de archivo *.tsv).

- Archivo binario: Este formato se usa para instancias se simulaciones intermitentes (instancias que se paran y se arrancan después de que el registro haya iniciado la simulación). Los datos se almacenan como un archivo de registro secuencial con formato binario con extensión de archivo *.blg.

- Archivo cíclico binario: Este formato se utiliza para guardar datos continuamente al mismo archivo de registro, donde las nuevas grabaciones se sobrescriben sobre las anteriores. Los datos se guardan como un archivo cíclico con formato binario de extensión *.blg.

- Seleccionar el tamaño de archivo máximo : Esta opción permite limitar el tamaño de archivo de registro a un tamaño determinado, en kbytes, o permitir que pueda ser tan grande como la cuota de disco o el sistema operativo tenga establecido.

  Se puede utilizar conjuntamente con la opción “Cuando el archivo este lleno”, de la pestaña Programación, que permite así ejecutar un comando en caso de que una acción concreta ocurra si el archivo de registro se llena.

El proceso que se necesita implementar es el que tiene que permitir obtener los datos de una simulación, procedentes de un fichero de texto, y gestionar esos datos desde la aplicación para que puedan ser visualizados y almacenados en una base de datos.

Existe una pantalla de estadísticas, pero para poder obtener los datos la única manera es con sentencias SQL que permitan extraer los datos de las tablas almacenadas mediante

el Karat Simulator en las tablas lógicas del repositorio creado con las herramientas Karat.

Actualmente para obtener los datos de una de simulación, se incorporan los contadores del parámetro (Memoria, Procesador, Trafico de red,…) y la característica concreta que se desea medir (memoria disponible, tiempo de CPU, tiempo inactividad, bytes enviados, bytes recibidos,...) al monitor del Sistema de Windows (se accede desde Inicio-Ejecutar-perfmon.exe).

Una vez los contadores están agregados, se guardan los datos de salida en el formato de archivo separado por tabuladores (*.tsv), ya que es un formato reconocido como hoja de cálculo, y los datos son guardados en un archivo de Microsoft Excel.

En resumen, el proceso actual implica intervención manual sobre los datos obtenidos en la simulación, circunstancia que puede inducir a errores y que aumenta notablemente el tiempo necesario para ejecutar y obtener los resultados de una simulación.

3.2 Sistema propuesto

3.2.1 Objetivos

El proyecto se basa en la realización de un programa, en lenguaje Java y mediante el entorno Eclipse JDK, que permita recoger los datos obtenidos de una simulación (realizada desde el Monitor del Sistema de Windows, con el comando <sar> de Unix o desde el programa Karat Simulator), gestionar y mostrar los datos resultantes de la simulación.

Los datos obtenidos en la simulación serán almacenados en una base de datos y, una vez guardados, podrán ser accedidos y gestionados por el programa mediante las herramientas de Karat 8.0.

La presentación de estos resultados podrá ser realizada por el programa de diferentes maneras y en diferentes formatos:

- Resultados Gráficos
- Resultados Numéricos

3.2.2 Reglas de gestión

Para implementar las funcionalidades del Simulador de Rendimiento se usarán algunos de los objetos Karat, los que mejor se adapten a cada situación.

Según se necesite introducir, gestionar o mostrar los datos, y según cuál sea el formato deseado, se utilizarán los siguientes objetos Karat:

- Interfaz para la entrada de datos : Formulario base
- Interfaz para la gestión de datos : Formularios de Gestión
- Interfaz para la presentación de resultados:
    - Formularios de Resultados
    - Listados de presentación de resultados (en formato de impresión)

	Reglas de gestión
1	Se debe tener en cuenta la configuración hardware para examinar factores de capacidad de ejecución del software realizado.
2	Todos los datos referentes a la base de datos lógica y física tienen que estar almacenados en un repositorio
3	La ejecución del monitor del Sistema de Windows, que permite la obtención del archivo con los datos de rendimiento de los parámetros deseados, está limitada a un usuario que posea permisos de creación/modificación/ejecución de los registros de rendimiento. El usuario Administrador dispone de estos permisos y puede concederlos a otros usuarios mediante la Directiva de Grupo
4	Los resultados de una simulación podrán ser gestionados por el usuario para poder ser presentados por la aplicación, según la configuración seleccionada. Además, podrán mostrarse en diferentes formatos, según la elección del usuario: Gráficos, Estadísticos y Numéricos.

3.2.3 Funcionalidades

- Selección de archivos de simulación (archivo log, en formato texto) donde leer los datos de los parámetros de rendimiento monitorizados.

- Almacenamiento de los datos de simulación en una base de datos, con un programa implementado en lenguaje Java, usando el entorno y la tecnología Karat 8.0 para conectar y operar con la base de datos.

- Gestión y selección de los resultados obtenidos en la simulación, mediante objetos Karat (tablas, consultas, objetos de negocio, formularios).

- Presentación de los resultados, según formato escogido por el usuario, hay tres formatos posibles y pueden ser mostrados de forma independiente, o en combinación entre ellos: gráficos y numéricos.

3.2.3.1 Perfil de los usuarios

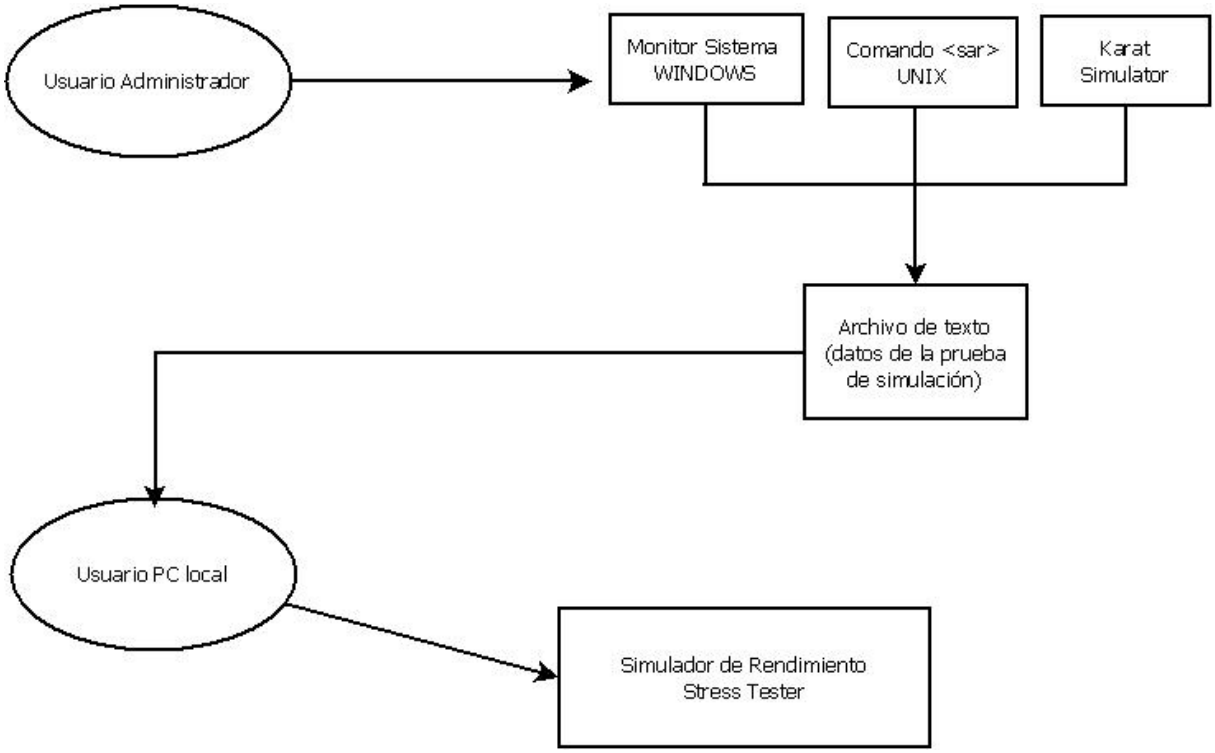

Figura 4: Diagrama de interacción con la aplicación de los diferentes perfiles de usuario

3.2.3.2 Funcionamiento de la aplicación

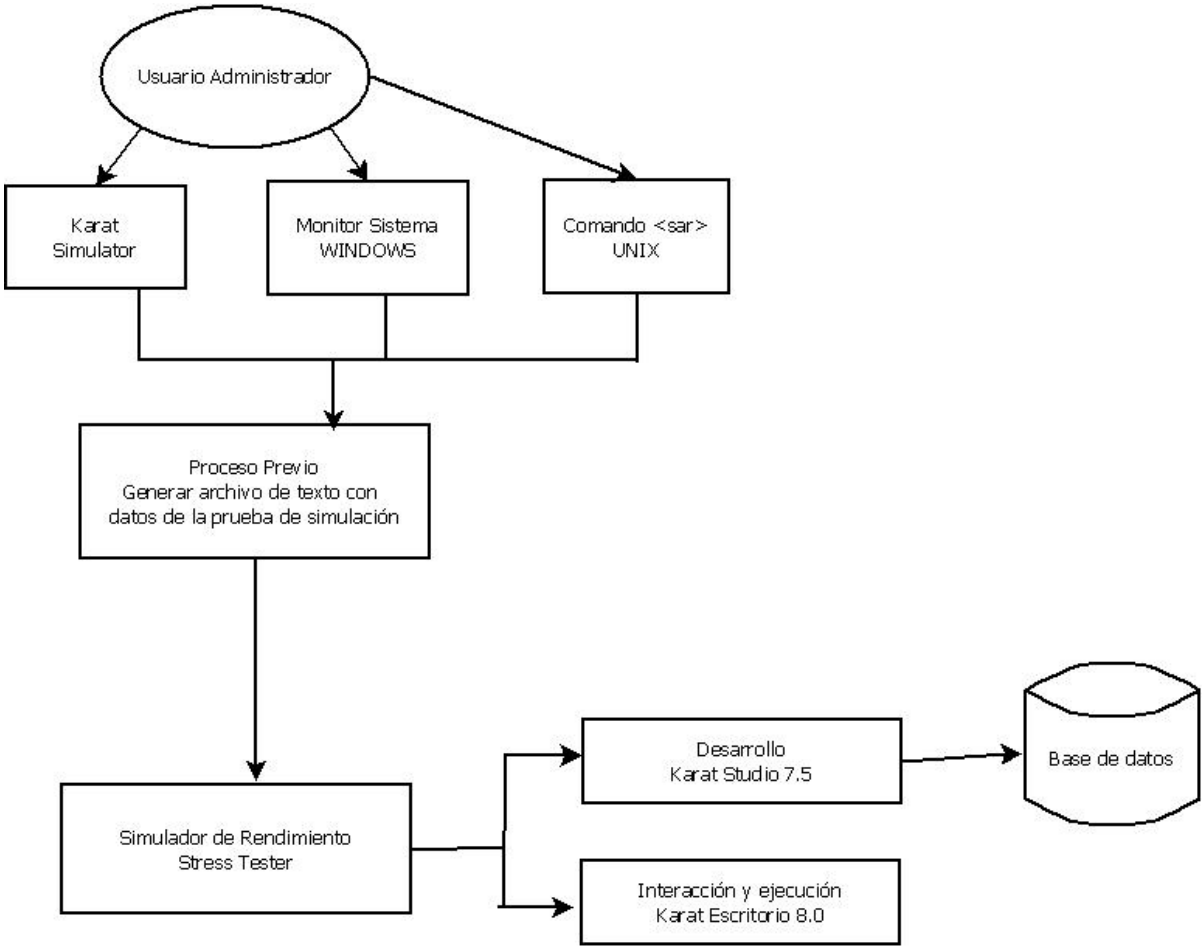

Figura 5: Diagrama de flujo de funcionamiento del sistema

MODULO PRINCIPAL: Simulador de Rendimiento

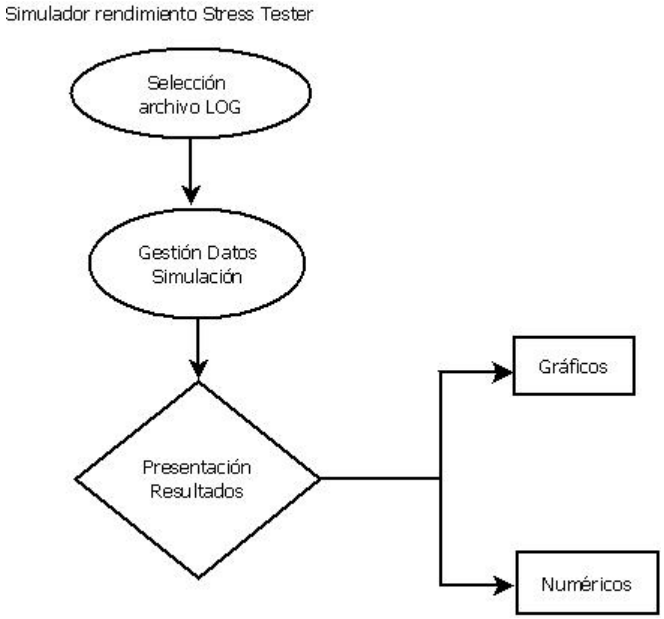

Figura 6: Diagrama de funcionamiento del módulo principal, la aplicación Simulador de Rendimiento Stress Tester

3.2.4 Plan de validación

- Revisión, corrección y validación del documento de definición de requerimientos por el tutor.
- Revisión, corrección y validación del análisis y del diseño por el tutor.
- Validación técnica de la aplicación Simulador de Rendimiento
- Test de casos de uso y realización de juegos de pruebas de la aplicación
- Depuración de errores de compilación y/o de ejecución y validación técnica final del programa
- Testeo de integración del programa sobre su entorno, pruebas sobre el sistema donde ha de funcionar y corrección de cualquier posible error o mal funcionamiento
- Validación final del Simulador de Rendimiento Stress Tester por parte del tutor.

4. Estudio de Viabilidad

4.1 Descripción general de la aplicación Stress Tester

El objetivo más importante es el de realizar diferentes pruebas de carga sobre una máquina, para poder observar la evolución de los diferentes parámetros de rendimiento.

En una prueba de carga de trabajo sobre un servidor, resulta muy importante la monitorización de todas las actividades en la red, tanto por parte del Servidor como por parte de los Clientes.

La finalidad es delimitar y comprender la causa de que tengan un determinado rendimiento en cada momento de la realización de las pruebas, y cuáles son las variaciones en los parámetros de rendimiento.

VENTAJAS

- Verificar y comprobar la respuesta del Servidor en situaciones de máxima carga de trabajo, máximo número de usuarios y diferentes valores de funcionamiento, para establecer los valores límites de funcionamiento óptimo.
- Poder medir y gestionar diferentes parámetros de rendimiento de la memoria, procesador y tráfico de red (estos serán los valores de rendimiento básicos, pero cualquier parámetro de rendimiento podrá ser introducido) , tanto de equipos locales con o sin conexión a red (Clientes) así como de equipos administradores de red a un nivel superior (Servidores)
- Compatibilidad con todos los tipos de base de datos con los cuales es compatible el entorno Karat (Posgres SQL, IBM db2, MySql , Oracle, Intercache Systems)

INCONVENIENTES

- Debido a ser un proyecto desarrollado en empresa , y con perspectiva básicamente de utilización a nivel interno, puede tener como inconveniente su poca aplicación comercial a gran escala :
    - Está enfocado a la gestión interna de Clientes o Servidores, a nivel de usuario Administrador.
    - Está desarrollado en el entorno Karat 8.0, propio y exclusivo de la empresa CCSAgresso.

4.2 Recursos necesarios

4.2.1 Recursos Humanos

Se establecerán dos perfiles diferentes de colaboradores en el Proyecto, aunque se debe tener en cuenta que en este caso es únicamente una consideración teórica, ya que todas las tareas serán llevadas a cabo por una sola persona, el alumno responsable de llevar a cabo el Simulador de Rendimiento Stress Tester.

- **Jefe de Proyecto**.

  Será responsable de la parte inicial del Proyecto:

  - Entrevistas con el cliente
  - Establecimiento de Requisitos y de Especificaciones
  - Documentación previa(Documentación de Requisitos y Especificaciones, Documentación del Estudio de Viabilidad)

- **Analista- Programador.**

  Será responsable de la segunda parte del proyecto, tomando como referencia toda la documentación generada en la primera parte del proyecto. Sus tareas más importantes son:

  - Realizar el análisis y diseño de la aplicación
  - Codificar y probar el funcionamiento del programa
  - Realizar las Pruebas y Test, en el que será el entorno real de ejecución de la aplicación

Finalmente, la fase de Validación Final, consiste en la revisión global del producto. El objetivo es revisar todo el proceso realizado, así como la evolución de todas las pruebas realizadas y el resultado final.

La validación final óptima servirá para comprobar y certificar que el resultado final cumple todo lo especificado y exigido inicialmente al Simulador de Rendimiento Stress Tester en las especificaciones y planificaciones de la empresa.

4.2.2 Recursos Materiales

Los recursos software utilizados ya han sido detallados previamente, pueden ser consultados en el Capítulo 2 de este mismo documento.

En lo que concierne a los Recursos Hardware, se detallan a continuación:

MAQUINA DE TRABAJO (Lenovo ThinkCentre, Serie M)

Proporcionada por la UAB dentro del convenio de prácticas con la empresa CSAGRESSO.

- Procesador Intel Core Duo a 2,66 GHz
- RAM 2,99 GB, con Extensión de Dirección Física (característica de los procesadores x86 que permite a los sistemas de 32-bit utilizar hasta 64 de memoria física, suponiendo que el sistema operativo proporcione el adecuado soporte
- DISCO DURO de 300 GB
- DVD+R/-RW de 16x
- 4 puertos USB
- Adaptador de Red Gigabit Ethernet PCI-E
- Windows XP Professional, Versión 2002, Service Pack 2

MAQUINA CLIENTE (HP Compaq dc5750 MicroTower)

Máquinas propiedad de la empresa CCSAGRESSO, utilizadas para simular el comportamiento de un Cliente que se conecta al Servidor.

- Procesador AMD Athlon 64 X2 - 4400+ 2x512kB (2,3GHz)
- RAM 1GB DDR2 PC5300
- DISCO DURO de 250 GB
- DVD+R/-RW de 16x
- Adaptador de Red Gigabit Ethernet Broadcom, NetXtreme integrado para HP
- Windows Vista Business

MAQUINA SERVIDOR (IBM System x3500)

Recursos mínimos del Servidor:

- Memoria RAM

  Memoria DIMM de búfer completo de 1 GB y tipo DDR II SDRAM (ampliable hasta 48 GB) a 667 MHz a través de 12 ranuras DIMM
  (DIMM significa Dual In-line Memory Module – Módulo de memoria lineal doble-, pequeño circuito impreso que contiene chips de memoria conectados directamente a la placa base, propios de la tecnología Intel)

- Procesador

  Procesador Intel Xeon 5160 con doble core, velocidad máxima de hasta 3,0 GHz (ampliable a dos procesadores)
  Formato: Torre/5U, compatible

- Controlador almacenamiento primario :

  Tecnología RAID 5 integrada (Serial ATA-150 / SAS)
  RAID=Conjunto de discos independientes
  Permite conexión de:
  - Cuatro Hot-Swap Serial Attached SCSI (SAS) o
  - Dos unidades de disco duro (HDDs) Simple-Swap Serial ATA (SATA)

- Controlador almacenamiento secundario :

  - IDE (ATA-66)
  - (Almacenamiento óptico – DVD-ROM)

- Ranuras de expansión

  3 PCI-Express, 2 PCI-X y 1 PCI

- Bahías de almacenamiento de Servidor

  Hot-swap 3,5''

- Capacidad máxima almacenamiento

  Serial Attached SCSI (SAS) 'hot-swap' de 2,4 TB

- Interfaz de red

  Ethernet Gigabit (GbE) dual integrado

4.3 Análisis de Coste/Beneficio

El coste económico queda detallado en el presupuesto presentado en el siguiente punto de este documento. Para que los datos sean lo más aproximados posibles a la realidad, se ha aplicado el salario real que se recibirá de la empresa, así como las horas reales previstas para finalizar el proyecto.

Los beneficios serán todas las funcionalidades que se podrán llevar a cabo con el Simulador de Rendimiento.

En la siguiente tabla de datos se expone la lista de todas las tareas, con su correspondiente carga de trabajo en horas efectivas, así como su coste derivado.

	Nombre de tarea	Acumulación de costos fijos	Comienzo	Fin	Trabajo	Costo total
1	**Proyecto**	**Prorrateo**	**lun 10/11/08**	**vie 05/06/09**	**520 horas**	**2.184,00 €**
2	**Diagnostico inicial**	**Prorrateo**	**lun 10/11/08**	**mié 12/11/08**	**12 horas**	**50,40 €**
3	Presentacion	Prorrateo	lun 10/11/08	mar 11/11/08	8 horas	33,60 €
4	Identificacion/ Instalación recursos necesarios	Prorrateo	mié 12/11/08	mié 12/11/08	4 horas	16,80 €
5	**Definicion de requisitos**	**Prorrateo**	**jue 13/11/08**	**lun 15/12/08**	**88 horas**	**369,60 €**
6	Documentacion especificacion problema	Prorrateo	jue 13/11/08	vie 21/11/08	28 horas	117,60 €
7	Documentacion de la definicion de requisitos	Prorrateo	lun 24/11/08	lun 15/12/08	60 horas	252,00 €
8	**Estudio de Viabilidad**	**Prorrateo**	**mar 16/12/08**	**lun 02/02/09**	**96 horas**	**403,20 €**
9	Documentacion estudio de viabilidad	Prorrateo	mar 16/12/08	lun 02/02/09	96 horas	403,20 €
10	**Analisis y diseño**	**Prorrateo**	**vie 06/02/09**	**vie 27/03/09**	**144 horas**	**604,80 €**
11	Diseño de la aplicación	Prorrateo	vie 06/02/09	vie 27/03/09	144 horas	604,80 €
12	**Realizacion del proyecto**	**Prorrateo**	**lun 30/03/09**	**mié 27/05/09**	**148 horas**	**621,60 €**
13	Codificación	Prorrateo	lun 30/03/09	lun 11/05/09	100 horas	420,00 €
14	Pruebas ,Casos de uso y Test	Prorrateo	mar 12/05/09	mié 27/05/09	48 horas	201,60 €
15	Validacion Final	Prorrateo	mar 26/05/09	vie 05/06/09	32 horas	134,40 €

Figura 7: Tabla de duración y coste de las diferentes fases del proyecto

4.3.1 Alternativas y posibles mejoras

En el caso del sistema operativo Windows, como ya ha sido descrito anteriormente en este documento, las herramientas de prestaciones están conectadas con el Monitor de Rendimiento del Sistema Operativo (aplicación perfmon.exe).

Los resultados son proporcionados por estas herramientas de forma grafica o analítica, según indique el usuario, y son los mismos datos que se muestran a través de la utilidad gráfica de monitorización (Interfaz gráfica del Monitor del Sistema de Windows).

En el caso del sistema operativo UNIX, las herramientas de prestaciones tienen una menor capacidad cuando se trata de la monitorización remota de una máquina.

Generalmente son comandos que adquieren datos estadísticos con sentencias poco elaboradas y que recogen directamente los datos publicados por el sistema.

La mejor opción es la de recoger medidas y parámetros de rendimiento directamente a través del sistema operativo, utilizando comandos de monitorización con las opciones de configuración adecuadas en cada caso.

Existen algunos comandos de monitorización que tiene su origen mayoritariamente en plataformas UNIX o LINUX, pero que se han generalizado y actualmente se encuentran integrados en un gran número de las plataformas y sistemas operativos actuales.

Los comandos más interesantes son los que pueden obtener medidas de los parámetros de rendimiento de las diferentes partes de una máquina, también llamados de Monitorización Polivalente. Los más importantes son:

- Vmstat (Virtual memory statistic).

  Su nombre solo especifica la memoria como objeto de la medición, pero puede retornar datos de los procesos del sistema operativo, utilización de CPU, utilización de memoria, tráfico en los discos, número de interrupciones, llamadas a sistema y cambios de contexto de la CPU.

  Sintaxis del comando <vstat>:

  vmstat <opciones> <intervalo de medición> <nº de muestras>

  - <opciones> permite especificar el tipo de información que se desea mostrar.

    Si no se indica ninguna opción se muestra toda la información: procesos, memoria, páginas de memoria, disco, interrupciones y CPU

  - <intervalo de medición>: periodo de tiempo entre dos mediciones, en segundos.

  - <nº de muestras>: numero de muestras a tomar en el intervalo de medición.

- Sar (System Activity Reporter).

  También permite obtener datos de rendimiento sobre diferentes partes de la máquina: utilización de memoria, de los discos y de la CPU.

  Sintaxis del comando <sar>:

  sar <opciones> <intervalo de medición> <nº de mediciones>

  Si no se especifican opciones ni intervalo, muestra por pantalla un resumen de las últimas 24 horas.

Las opciones más interesantes son:

- -b Actividad de la caché
- -c Llamadas al sistema
- -g Información de pageout.
- -p Información de pagein.
- -q Estadísticas sobre la cola de procesos en espera de CPU
- -w Estadísticas de actividad de swapping

Este es el comando más indicado para poder obtener los datos de rendimiento de las pruebas de simulación en un archivo de texto.

La sintaxis de la instrucción para obtener la salida en un fichero es la siguiente:

sar –A –o <fichero_binario> <intervalo de medición> <nº de mediciones>

Con la opción –A se indica que el fichero binario tiene que contener todos los parámetros de medición. Una vez se ha obtenido el fichero binario se debe convertir a formato texto, redireccionando la salida en UNIX a un fichero de texto, con la siguiente instrucción:

Con esta instrucción se sobrescribe el contenido de < fichero_texto>:

sar –A –f <fichero_binario> > < fichero_texto>

Con esta instrucción se agrega el contenido de < fichero_texto>, inmediatamente después del contenido ya existente en el fichero:

sar –A –f <fichero_binario> >> < fichero_texto>

Es importante señalar que todas las opciones que se hayan guardado en el fichero binario son las mismas que se podrán redireccionar al fichero de texto.

- Top

Es el comando más extendido en los sistemas UNIX, gracias a ser de libre distribución, así como a sus diferentes funcionalidades y sencillez de uso. Además de sistemas UNIX/Linux, también está disponible para plataformas Macintosh.

Muestra la situación en tiempo real de los procesos en ejecución, la ordenación por

defecto es según el porcentaje de CPU usado por cada proceso.

Informaciones mostradas:

- Número de procesos ejecutando y cuántos de ellos están activos, durmiendo, en proceso de terminar (estado zombie) y cuantos finalizados
- Utilización de CPU
- Cantidad de memoria física total, cantidad usada y cantidad libre
- Cantidad de memoria swap total, cantidad usada y cantidad libre
- Numero de PID (process ID) y usuario que lo está ejecutando (USERNAME)
- Prioridad del proceso (PRI)
- Valor nice (NI) .Siempre entre -20 y 20. Cuanto más bajo mayor prioridad tendrá el proceso.
- Tamaño del proceso (SIZE)
- Tamaño total del proceso sumado a los datos que usa (RSS)
- Estado del proceso (STATE)
- Tamaño de las librerías del proceso (LIB)
- Porcentaje de CPU (%CPU) y de memoria (%MEM)
- Tiempo de ejecución (TIME)
- Nombre del proceso (COMMAND)

La tecla “h” muestra una pantalla de ayuda, que activa el modo interactivo que permite realizar las siguientes acciones:

- Finalizar un proceso enviándole una señal de kill.
- Mostrar los procesos en estado ‘idle’
- Cambiar la ordenación de los procesos, o el numero de procesos visualizados

En la búsqueda de alternativas al programa Performance Monitor de Windows, lo indispensable es que los resultados de rendimiento puedan ser almacenados en un fichero de salida. Para extraer los datos en un fichero de texto se utilizó el sistema operativo Suse Linux 10, implementado sobre el emulador virtual Microsoft Virtual PC.

Respecto a lo que se necesita para la aplicación Stress Tester, <sar> es el comando más indicado, ya que permite obtener los datos de rendimiento de las pruebas de simulación en un archivo binario de salida.

Mediante una sencilla redirección de salida en Unix, es posible pasar del fichero binario a un fichero de texto, y obtener así el archivo de simulación necesario.

Para el caso particular de las pruebas de simulación que se quieren obtener, únicamente se necesitan los datos de CPU, Memoria y Tráfico de Red. Las opciones que tienen que ser especificadas son las siguientes:

- -n DEV : Datos de los dispositivos de Red
- -u : Datos de utilización de la CPU
- -r : Datos de utilización de la Memoria

sar –n DEV –u –r –o <fichero binario> <intervalo de tiempo> <numero de muestras>
sar –n DEV –u –r –f <fichero binario> > <fichero texto>

Es importante señalar que todas las opciones que se hayan guardado en el fichero binario son las mismas que se podrán redireccionar al fichero de texto.

4.3.2 Evaluación de riesgos

El único posible riesgo es el hecho de que la versión actual de Karat (entorno propio y exclusivo de la empresa CCSAGRESSO) está actualmente en un proceso de migración desde la antigua versión Karat 7.5, basada en Visual Basic, a la nueva versión Karat 8.0, basada en Java.

Debido a eso, el programa de desarrollo Karat Studio está en la versiona antigua 7.5, y el programa de Interfaz con el usuario y de ejecución de las aplicaciones , Karat Escritorio, está en la versión nueva , la 8.0 .

Este hecho podría provocar algún problema de compatibilidad o de otro tipo, únicamente debido a que la migración de versiones está en marcha actualmente y puede haber casos de uso que hayan quedado sin revisar, o errores que surjan y aún no hayan sido tratados.

4.4 Planificación

4.4.1 Planificación inicial

Seguidamente se expone una estimación de las fases de desarrollo del proyecto y su duración detallada en días laborables y horas de trabajo efectivas

1. **Diagnostico inicial (3 días, 12 horas)**

    1.1 Presentación (2 días, 8 horas)
    1.2 Identificación recursos necesarios, instalación software (1 día, 4 horas)

2. **Definición de requisitos (22 días, 88 horas)**

    2.1 Documentación de las Especificaciones del problema (7 días, 28 horas)
    2.2 Documentación de la Definición de requisitos (15 días, 60 horas)

3. **Estudio de Viabilidad (24 días, 96 horas)**

    3.1 Documentación del Estudio de Viabilidad

4. **Análisis y diseño (30 días , 80 horas)**

    4.1 Diseño de la aplicación (30 días, 80 horas)

5. **Realización del proyecto (37 días , 148 horas)**

5.1 Codificación (25 días, 100 horas)
5.2 Pruebas, Casos de uso y Test (12 días, 48 horas)

6. Documentación Manual de Usuario y Validación Final (12 días, 36 horas)

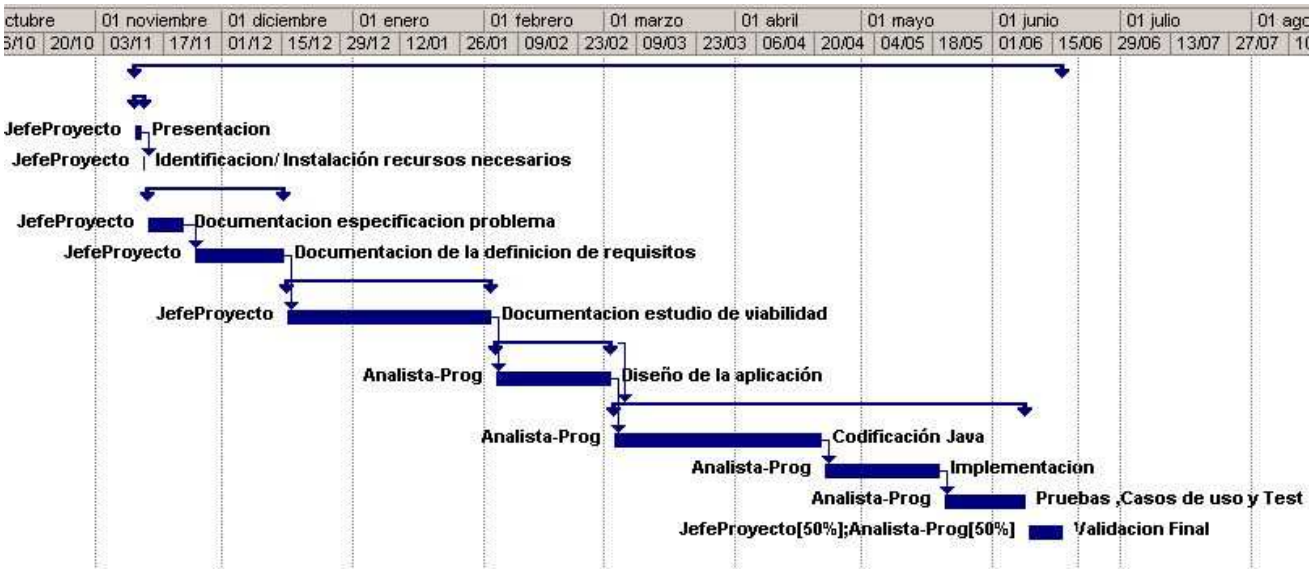

Figura 8: Diagrama de Gantt de la planificación inicial del proyecto

USO DE LOS RECURSOS HUMANOS DEL PROYECTO

Nombre del recurso	Tasa estándar	Trabajo	Costo
JefeProyecto	4,20 €/hora	214 horas	898,80 €
Analista-Prog	4,20 €/hora	306 horas	1.285,20 €

Figura 9: Tabla de los recursos humanos teóricos del proyecto

Nombre del recurso	Trabajo	Costo	Comienzo	Fin
⊟ JefeProyecto	214 horas	898,80 €	lun 10/11/08	vie 05/06/09
Presentacion	*8 horas*	*33,60 €*	*lun 10/11/08*	*mar 11/11/08*
Identificacion/ Instalación recursos necesarios	*4 horas*	*16,80 €*	*mié 12/11/08*	*mié 12/11/08*
Documentacion especificacion problema	*28 horas*	*117,60 €*	*jue 13/11/08*	*vie 21/11/08*
Documentacion de la definicion de requisitos	*60 horas*	*252,00 €*	*lun 24/11/08*	*lun 15/12/08*
Documentacion estudio de viabilidad	*96 horas*	*403,20 €*	*mar 16/12/08*	*lun 02/02/09*
Validacion Final	*18 horas*	*75,60 €*	*mar 26/05/09*	*vie 05/06/09*
⊟ Analista-Prog	306 horas	1.285,20 €	vie 06/02/09	vie 05/06/09
Diseño de la aplicación	*144 horas*	*604,80 €*	*vie 06/02/09*	*vie 27/03/09*
Codificación	*100 horas*	*420,00 €*	*lun 30/03/09*	*lun 11/05/09*
Pruebas ,Casos de uso y Test	*48 horas*	*201,60 €*	*mar 12/05/09*	*mié 27/05/09*
Validacion Final	*14 horas*	*58,80 €*	*jue 28/05/09*	*vie 05/06/09*

Figura 10: Tabla de la distribución en horas de los recursos humanos teóricos del proyecto

4.4.2 Desviaciones respecto a la planificación inicial

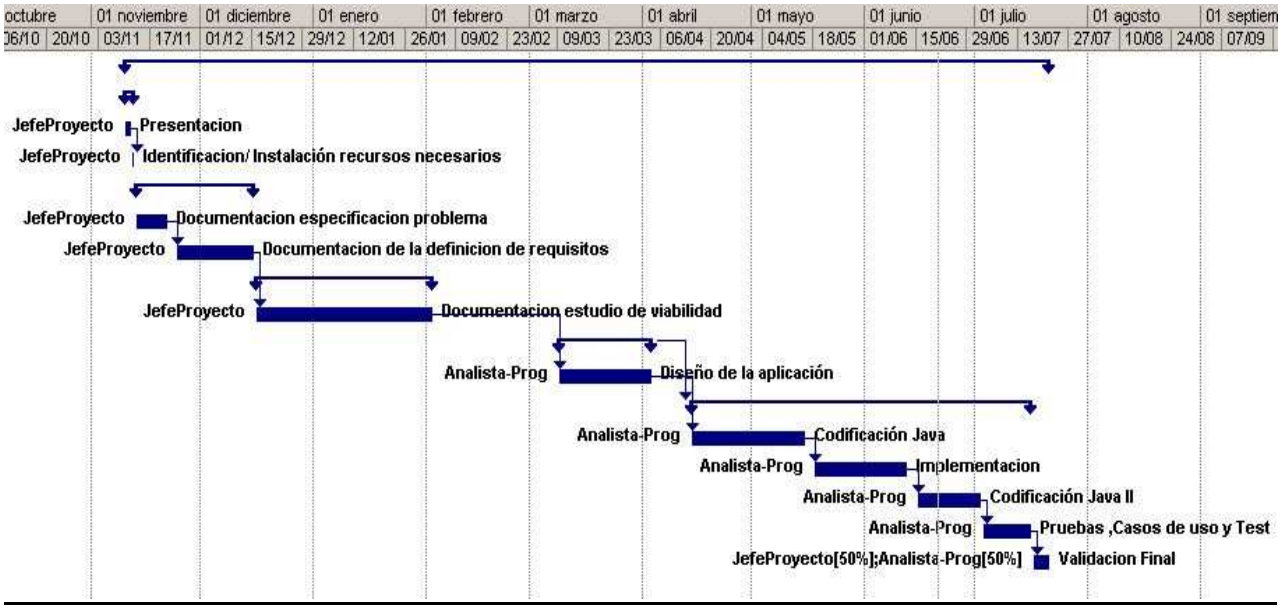

Figura 11: Diagrama de Gantt de la planificación final del proyecto

Únicamente se produjo una desviación respecto a la planificación inicial del proyecto, causada por la inclusión del tercer tipo de fichero de texto procesable, los ficheros de simulación procedentes del comando <sar> de Unix.

Este hecho provocó que la fase de Codificación Java fuese dividida en dos partes, debido a que primero la aplicación debía estar terminada para los dos tipos de fichero de texto inicialmente establecidos para la aplicación.

Debido a disponer del suficiente tiempo, una vez implementada y comprobado el óptimo funcionamiento de la aplicación, se retomó la fase de codificación para permitir a la aplicación procesar también los ficheros de simulaciones Unix.

4.5 Conclusiones

Este Estudio de Viabilidad está dedicado a analizar la viabilidad técnica, económica y legal del Simulador de Rendimiento Stress Tester, una aplicación desarrollada como Proyecto de Fin de Carrera de Ingeniería Técnica en Informática de Sistemas.

El Proyecto esta englobado en un convenio de prácticas establecido entre la empresa CCSAGRESSO y la Universidad Autónoma de Barcelona, tiene una duración estimada de 500 horas efectivas de trabajo, con la posibilidad de ampliarlo hasta un máximo de 550 horas.

En el momento de analizar su viabilidad económica, resulta muy importante tener en cuenta que se trata de un Producto destinado a ser de uso interno dentro del Área de Testing, en el departamento de Fábrica, desde el cual se hará el desarrollo y la realización del Proyecto.

Por esta razón la vocación comercial de la aplicación, como mínimo en primera instancia, quedara reducida al uso interno de la empresa CCSAGRESSO.

Por tanto, el Simulador de Rendimiento tendrá una vocación más enfocada a la Administración Interna: realización de test y pruebas, evaluación de capacidades de una determinada máquina y monitorización de parámetros de rendimiento.

Esta afirmación quedará reforzada por el hecho de que el Simulador de Rendimiento será realizado en el entorno de desarrollo Karat 8.0, la nueva versión y que está actualmente en proceso de desarrollo e implantación.

Los costes del proyecto son conocidos a priori, por un lado el coste fijado por el convenio como ayuda al estudiante, y el coste de los recursos necesarios para su realización. Estos recursos no suponen ningún coste efectivo ya que son propiedad de la empresa y ya están amortizados.

En resumen, la viabilidad del proyecto queda asegurada por su bajo coste económico y los beneficios tecnológicos que supondrá la nueva aplicación.

Respecto a la viabilidad legal del proyecto, la única ley por la que se ve afectado es la Ley de Propiedad Intelectual, y más concretamente por la normativa de la UAB respecto a la propiedad intelectual en proyectos de final de carrera.

Esta normativa establece que la propiedad intelectual es repartida al cincuenta por ciento, siendo el/los profesor/es que han dirigido el trabajo coautores de la obra final, juntamente con el estudiante.

Respecto a los derechos de explotación de la obra, pertenecen a la UAB, y en caso de que exista beneficio económico derivado del Proyecto, el autor o conjunto de autores tendrán derecho a una participación del cincuenta por ciento de los beneficios netos obtenidos.

5. Diseño y desarrollo de la aplicación Stress Tester

5.1 Visión general

El objetivo del presente capítulo es el de detallar todo el proceso de diseño y las propuestas de implementación del modelo físico y todas las funcionalidades de la aplicación Simulador de Rendimiento Stress Tester.

La aplicación utilizada es el entorno de desarrollo de software Karat 8.0, propio de la empresa CCSAgresso, y utilizando como entorno de desarrollo el programa de libre distribución Eclipse JDK (Java Development Kit).

Está desarrollado con el programa Karat Studio, versión 7.5.3.0 Build 358, e implementado para funcionar bajo la nueva versión de Karat Escritorio 8.0, basada en Java.

La aplicación Simulador de Rendimiento (Stress Tester) tiene como objetivo poder seleccionar y gestionar los datos de pruebas de simulación, procedentes de ficheros de texto.

Pueden estar en tres formatos diferentes dependiendo del programa de simulación desde donde sean generados:

- Archivo de texto con delimitador de columnas, procedente del programa Performance Monitor de Windows.

  El delimitador puede ser el carácter "," (extensión *.csv) o el tabulador (extensión *.tsv).

- Archivo de texto, procedente de Simulaciones Karat.

- Archivo de texto, procedente del comando <sar> (System Activity Report) de Unix.

Una vez realizada la selección del archivo con los datos de la simulación y el nombre de la prueba de simulación asociada, el siguiente paso es el de almacenar todos los valores del archivo de texto en la base de datos.

Finalmente, una vez la base de datos ha sido actualizada, se trata de diseñar los componentes Karat necesarios para realizar las consulta, inserción, actualización y eliminación sobre la base de datos.

Para ello serán necesarios los siguientes objetos Karat:

Tablas

Una tabla es la unidad de almacenamiento donde se crea el conjunto de datos de nuestra base de datos. Estos valores estarán ordenados en columnas verticales, que se denominan campos.

Los campos son los distintos tipos de datos que componen la tabla, por ejemplo: nombre, apellido, domicilio. La definición de un campo requiere: el nombre del campo, el tipo de campo, el ancho del campo, etc.

Las tablas son un medio de representar la información de una forma más compacta y posibilitan el acceso a información contenida en dos o más tablas.

El principal aspecto a tener en cuenta, durante el diseño de una tabla, es determinar claramente los campos necesarios y definirlos de forma adecuada con un nombre explicativo de su función, especificando su tipo y su longitud.

Consultas Base

Una consulta a una base de datos se puede abstraer como una pregunta que se formula a la base de datos con el fin de extraer y presentar la información resultante de diferentes formas (pantalla, impresora...).

Las consultas base son necesarias para realizar consultas de determinados campos que forman parte de una tabla de la base de datos, añadiendo la posibilidad de que estos campos sean seleccionados según un cierto criterio (DISTINCT, GROUP BY, HAVING) y mostrados en cierto orden (ORDER BY).

Generalmente, la consulta base está definida por el desarrollador, y contiene información sobre la estructura de la consulta: tablas, campos proyectados, relaciones entre tablas, agrupaciones, criterios de selección y la secuencia de ordenación de campos.

La consulta hereda toda la definición de la consulta base, pero puede ser modificada por el usuario de la aplicación, de manera que puede cambiar el criterio de selección y la ordenación de los campos.

Objetos de Negocio

El objeto de negocio karat es el objeto que permite gestionar como una entidad única varios niveles de información de la capa de datos.

Este objeto debe contener toda la lógica de negocio de la entidad que implementa, así como las reglas de validación de esta. Para cada entidad de la aplicación es necesaria la definición de un objeto de negocio puesto que éste será el que nos permitirá hacer el mantenimiento de los datos asociados a la entidad que representa.

El objeto de negocio es el objeto clave dentro del entorno de desarrollo karat puesto que representa el núcleo sobre el que se basarán el resto de objetos como:

- Formularios
- Listados
- Procesos de transferencia de datos

- Web services

Un web service es un conjunto de protocolos y estándares (XML, SOAP, HTTP, WSDL)

que sirven para intercambiar datos entre aplicaciones, desarrolladas en lenguajes de programaciones diferentes y ejecutadas sobre cualquier plataforma.

Este componente, por lo tanto, permite el intercambio de los datos de un objeto de negocio o de una aplicación basada en karat usando la estructura de datos XML (Extended Markup Language) permitiendo la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Un web service se compone básicamente de:

- Un formato que describa sus propiedades y métodos basado en XML (el interfaz del componente). Por lo general este formato es el WSDL (Web Service Description Language).

- Un protocolo de basado en mensajes que permita llamadas al web service. Por lo general este protocolo es SOAP (Simple Object Access Protocol).

- Un protocolo de transporte que se encargue de transportar los mensajes por internet. Por lo general este protocolo de transporte es HTTP (Hiper-Text Transport Protocol), el mismo que usamos para navegar por la Web.

Formularios

Un formulario karat es el objeto que implementa la presentación visual de una pantalla.

Existen varios tipos de formularios en entorno karat, como pueden ser:

- Mantenimientos. Son pantallas que representan gráficamente un objeto de negocio. Normalmente se utilizan para realizar el mantenimiento de datos del objeto de negocio (alta, baja, modificación).

- Diálogos. Son pantallas que no se vinculan a una fuente de datos de la aplicación, sino que se utilizan para realizar la captura de información al usuario para desencadenar cierto proceso determinado. karat proporciona una serie de diálogos propios como por ejemplo el visor de registros, el diálogo de impresión, etc.
  Es posible diseñar diálogos fácilmente por parte de la aplicación.

- Diálogos estándar. Son una serie de diálogos estándar que permiten seleccionar elementos concretos. Por ejemplo:

    - Selección de ficheros
    - Selección de fuente
    - Selección de color

Mensajes

Los mensajes son objetos Karat destinados a facilitar la interacción entre el usuario y la aplicación.

Generalmente pueden ser de dos tipos: para avisar de un error o para informar al usuario de algún evento que se haya producido durante la ejecución de la aplicación.

Listados

Un listado karat es el objeto que implementa la impresión de un objeto de negocio y permite mostrar su información en distintos formatos.

Karat proporciona un conjunto de herramientas que permiten diseñar, generar y configurar listados independientemente de su grado de complejidad.

Está formado entre otros por los siguientes componentes de karat: Diseñador de listados, Configurador de listados, Diálogo de impresión y Motor de impresión.

- El Diseñador de listados es una herramienta que facilita el diseño gráfico de listados e informes a partir de las definiciones de objetos de negocio almacenados en el repositorio.

  Permite transformar los datos de su empresa, introducidos mediante los productos, para convertirlos en información útil y obtenerlos por el dispositivo de salida que crea oportuno: impresora, fichero (en diversos formatos) o mensaje de correo electrónico.

- El Configurador de listados es una herramienta gráfica que permite personalizar los listados ya diseñados gracias a la definición de criterios de ordenación y rangos de selección en todos los tipos de impresos (informes, listados, formularios, etc.).

- El Diálogo de impresión es una herramienta gráfica que permite al usuario final la personalización en la ejecución de los listados. Integra la ejecución de los listados con karat Escritorio y el Diseñador de listados.

El diseño, configuración y parámetros de ejecución del listado karat determinan qué información y cómo se muestra esa información al imprimir.

Clases Java

En el caso de eventos Karat que sean gestionados por código, el formulario que gestione esos eventos tendrá asociado un archivo *.java que será el encargado de llevar a cabo la funcionalidad que se necesita para ese evento.

Para esta aplicación, básicamente hay dos funcionalidades que se realizan por código:

- Entrada de Datos: Procesar un fichero de texto, que contiene una prueba de simulación, para que sus valores puedan ser almacenados en la base de datos.

- Actualización de los datos de Memoria Total: opción necesaria para los ficheros de texto procedentes del Performance Monitor de Windows, debido a que se proporciona el dato de Memoria Disponible, y lo que se necesita almacenar es la Memoria Utilizada.

  Para solucionarlo, mediante esta opción el usuario debe informar de cuál es la memoria total del equipo simulado. De esta manera, se obtiene el dato de la memoria utilizada y se almacena en la base de datos.

Clases del explorador

Las clases del explorador son la herramienta que permite agrupar diferentes acciones bajo una aplicación Karat (un mismo punto de menú en la interfaz gráfica de la aplicación).

Las acciones son el resultado de la correspondencia entre un formulario, una consulta base (si se trata de un formulario de conjunto de registros, basado en una consulta base) y las opciones de trabajo con el formulario (modos como podrían ser: Consultar, Nuevo registro, Navegación,...).

Son accesibles directamente desde cada aplicación Karat: pulsando el botón derecho del ratón se muestran las diferentes acciones disponibles para la aplicación Karat en cuestión.

Aplicaciones Karat

Las aplicaciones Karat son los objetos que permiten definir lo que serán los componentes principales de la interfaz gráfica de la aplicación, las entradas del menú principal.

Para facilitar la separación, entre aplicaciones Karat que sean de ámbitos y funcionalidades diferentes, se utilizan las carpetas.

5.1.1 Entrada de datos

Es el primer paso de todo el proceso que debe llevar a cabo nuestra aplicación.

La información inicial, y la más importante, es la ubicación y el nombre del fichero de texto, ya sea procedente del Performance Monitor de Windows, del Comando <sar> de Unix o de una Simulación Karat.

Contiene los datos de las Pruebas de Simulación, y existen dos opciones:

- Los datos de la simulación del fichero de texto corresponden a una prueba de simulación ya existente en la base de datos.

  En este caso el usuario deberá seleccionar la prueba de simulación en cuestión.

- Los datos de la simulación del fichero de texto hacen referencia a una prueba de simulación nueva.

  En este caso el usuario deberá informar de los valores de los diferentes campos de esa prueba de simulación nueva.

Además, otros datos de los cuales el usuario debe informar son:

- Para Simulaciones del Performance Monitor de Windows y del Comando <sar> de Unix: Número total de usuarios.

- Para Simulaciones Karat: Número total de usuarios y de usuarios por cada simulador.

También existe la posibilidad de introducir una Prueba de Simulación nueva de forma independiente, sin que esté asociada a ningún fichero de texto que contenga los datos numéricos de una Simulación Karat, del Performance Monitor de Windows o del comando <sar> de Unix.

5.1.2 Resultados Gráficos de las simulaciones

Esta parte de la aplicación es la encargada de mostrar por pantalla los datos de una determinada prueba de simulación en formato gráfico, y también visualizar los correspondientes datos numéricos a aquello que muestra en el Gráfico de Resultados.

El usuario puede seleccionar los diferentes valores que identifican una prueba de simulación:

- Para Simulaciones del Performance Monitor de Windows y del Comando <sar> de Unix:
  - Nombre de la Máquina Simulada
  - Nombre de la Prueba de Simulación
  - Parámetro de rendimiento del performance monitor o del comando <sar> de Unix
  - Tipo de parámetro de rendimiento del performance monitor o del comando <sar> de Unix

  Con el objetivo de mostrar:
  - Máximo entre los valores numérico del parámetro de rendimiento
  - Media entre los valores numérico del parámetro de rendimiento
  - Mínimo entre los valores numérico del parámetro de rendimiento
  - Número total de usuarios

- Para Simulaciones Karat :
  - Nombre del script de simulación
  - Nombre de la Prueba de Simulación
  - Nombre de la Simulación Karat
  - Parámetro de rendimiento de la Simulación Karat
  - Tipo de parámetro de rendimiento de Simulación Karat

  Con el objetivo de mostrar:
  - Máximo entre los valores numérico del parámetro de rendimiento
  - Media entre los valores numérico del parámetro de rendimiento
  - Mínimo entre los valores numérico del parámetro de rendimiento

- Número total de usuarios

5.1.3 Resultados Numéricos de las simulaciones

En este punto el objetivo es poder visualizar los valores referentes a una Prueba de Simulación almacenados en la base de datos, directamente tal como han sido guardados en los campos de las tablas. En este caso no se muestra ninguna representación gráfica.

5.1.3.1 Resultados generales

Se muestra todo el contenido almacenado en la base de datos, mediante sus respectivas consultas base que proyectan todos los campos de cada tabla.

5.1.3.1 Resultados detallados según el número de usuarios de la simulación

En este caso, solamente se pueden mostrar los resultados detallados para Pruebas de Simulación del Performance Monitor de Windows, del Comando <sar> de Unix o de Simulaciones Karat.

Es necesario porque pueden existir Pruebas de Simulación con las mismas características, y que únicamente difieran en el número de usuarios.

De esta manera se ofrece la posibilidad de ver los detalles específicos de una Prueba de Simulación que se ha llevado a cabo para diferentes números de usuario.

5.2 Base de datos

5.2.1 Diagrama de tablas

Diagrama de las nuevas tablas que serán necesarias para el funcionamiento de la aplicación, con la especificación de sus claves primarias y sus claves foráneas.

Las claves primarias están indicadas con el símbolo de una llave en color amarillo, las claves foráneas son las siguientes:

FK_st_header_pm = El campo <xnomprueba> debe tener un valor que esté en las tablas **st_lineas_pm** y **st_header**

FK_st_header_sim = El campo <xnomprueba> debe tener un valor que esté en las tablas **st_lineas_sim** y **st_header**

FK_st_maquina = El campo <xmaquina> debe tener un valor que esté en las tablas

st_lineas_pm y **st_maquina**

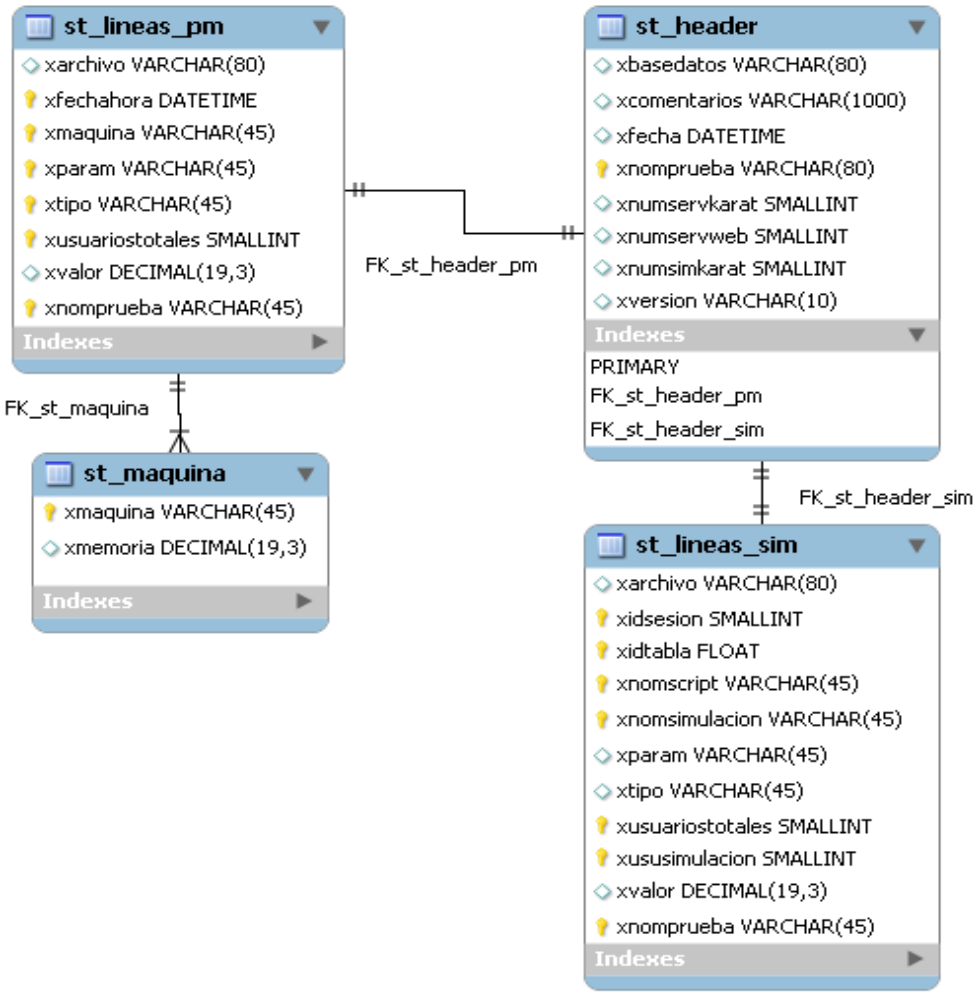

Figura 12: Diagrama de nuevas tablas

5.2.2 Descripción de las tablas

5.2.2.1 Tabla st_header

Tabla que debe almacenar los datos referentes a las pruebas de simulación.

5.2.2.1.1 Campos

xnomprueba

Descripción:	Nombre de la Prueba de Simulación		
Tipo:	TEXT	**Obligatorio:**	si
Longitud:	45	**Longitud fija:**	no
Etiqueta de entrada:	Nombre Prueba		

Descripción:	Nombre de la Prueba de Simulación		
Etiqueta de salida:	Nombre Prueba		
Mensaje:	Nombre de la Prueba de Simulación		

xbasedatos

Descripción:	Nombre del gestor de base de datos		
Tipo:	TEXT	**Obligatorio:**	no
Longitud:	45	**Longitud fija:**	no
Etiqueta de entrada:	Nombre base de datos		
Etiqueta de salida:	Nombre base de datos		
Mensaje:	Nombre de la base de datos		

xcomentarios

Descripción:	Comentarios de la Prueba de Simulación		
Tipo:	TEXT	**Obligatorio:**	no
Longitud:	500	**Longitud fija:**	no
Etiqueta de entrada:	Comentarios Prueba		
Etiqueta de salida:	Comentarios Prueba		
Mensaje:	Comentarios de la Prueba de Simulación		

xdescripcion

Descripción:	Descripción de la Prueba de Simulación		
Tipo:	TEXT	**Obligatorio:**	no
Longitud:	180	**Longitud fija:**	no
Etiqueta de entrada:	Descripción Prueba		
Etiqueta de salida:	Descripción Prueba.		
Mensaje:	Descripción de la Prueba de Simulación		

xfecha

Descripción:	Fecha de la Prueba de Simulación		
Tipo:	DATE	**Obligatorio:**	no
Longitud:		**Longitud fija:**	no
Etiqueta de entrada:	Fecha Prueba		
Etiqueta de salida:	Fecha Prueba		
Mensaje:	Fecha de la Prueba de Simulación		

xnumservKarat

Descripción:	Número de Servidores Karat		
Tipo:	INTEGER	**Obligatorio:**	no
Etiqueta de entrada:	Num. Serv. Karat		
Etiqueta de salida:	Num. Serv. Karat		
Mensaje:	Número de Servidores Karat		

xnumservweb

Descripción:	Número de Servidores Web		
Tipo:	INTEGER	**Obligatorio:**	no
Etiqueta de entrada:	Num. Serv. Web		
Etiqueta de salida:	Num. Serv. Web		
Mensaje:	Número de Servidores Web		

xnumsimKarat

Descripción:	Número de Simuladores Karat		
Tipo:	INTEGER	**Obligatorio:**	no
Etiqueta de entrada:	Num. Sim. Karat		
Etiqueta de salida:	Num. Sim. Karat		
Mensaje:	Número de Simuladores Karat		

xversion

Descripción:	Versión de Karat		
Tipo:	TEXT	**Obligatorio:**	no
Longitud:	10	**Longitud fija:**	no
Etiqueta de entrada:	Versión Karat		
Etiqueta de salida:	Versión Karat		
Mensaje:	Versión de Karat		

5.2.2.1.2 Claves únicas

Clave primaria: pk_st_header
Campos: xnomprueba

(Para ver el resto de Tablas, consultar el apartado 3.2.1 del documento de Análisis y Diseño en el Anexo)

5.2.3 Descripción de las Consultas Base

5.2.3.1 st_calculados

Obtener los campos:
<xnomprueba>,<xmaquina>, <xparam>, <xtipo>, <xusuariostotales>
de la tabla st_lineas_pm, ordenados con la misma configuración en la que se muestran los campos en el SELECT .

Se aplican las cláusulas DISTINCT y GROUP BY , además de tres campos calculados mediante las expresiones :
<xmin>= MIN(xvalor)
<xmedia>=AVG(xvalor)
<xmax>=MAX(xvalor)

Código SQL de la consulta:

```
SELECT DISTINCT st_lineas_pm.xnomprueba AS xnomprueba, st_lineas_pm.xmaquina AS xmaquina, st_lineas_pm.xparam AS xparam, st_lineas_pm.xtipo AS xtipo, st_lineas_pm.xusuariostotales AS xusuariostotales, MIN(xvalor) AS xmin, AVG(xvalor) AS xmedia, MAX(xvalor) AS xmax FROM imp.st_lineas_pm st_lineas_pm GROUP BY st_lineas_pm.xnomprueba , st_lineas_pm.xparam , st_lineas_pm.xtipo , st_lineas_pm.xmaquina , st_lineas_pm.xusuariostotales ORDER BY 1 ASC, 2 ASC, 3 ASC, 4 ASC, 5 ASC, 6 ASC, 7 ASC, 8 ASC
```

(Para ver el resto de Consultas Base, consultar el apartado 3.2.2 del documento de Análisis y Diseño en el Anexo)

5.3 Aplicación Stress Tester

5.3 .1 Descripción de las clases Java

5.3.1.1 Clase <lectura.java>

Es el archivo Java asociado al formulario st_datos_pm , su función es la de procesar los ficheros de texto, procedentes del Performance Monitor de Windows, que contienen los datos de una prueba de simulación.

El fichero de texto tiene una ventaja: sus columnas de valores están delimitadas por un carácter especial, el más habitual es la coma (carácter “,”).

Esta particularidad permite la utilización de la librería CsvReader.java, que es de libre distribución, y permite el tratamiento de ficheros con texto en columnas delimitadas por un carácter.

Una vez realizada la separación de los datos, se deben hacer las comprobaciones previas para saber a qué campo corresponde cada columna (cada columna es reconocida y etiquetada según cuál sea el título de encabezamiento).

Finalmente se almacenan todos los valores en la base de datos, concretamente en la tabla **st_lineas_pm**.

5.3.1.2 Clase <CsvReader.java>

Es una librería basada en streams para la lectura y escritura de archivos CSV (comma separated values) y otros tipos de datos con delimitación de columnas.

Esta librería es de tipo software libre, por ello puede ser utilizada sin necesidad de ningún permiso o licencia ya que es de libre distribución.

Para facilitar el acceso a la ayuda, se ha traducido la explicación, de cada una de las funciones utilizadas. Se encuentra en las páginas de ayuda en formato html generadas mediante la utilidad Javadoc (disponibles en la carpeta llamada Javadoc, dentro del anexo).

5.3.1.3 Clase <lectura_sar.java>

Es el archivo Java asociado al formulario st_datos_pm , su función es la de procesar los ficheros de texto, procedentes del comando <sar> de Unix, que contienen los datos de una prueba de simulación.

El fichero de texto también contiene los datos organizados en columnas de texto, pero en este caso los datos no están en columnas únicas.

Cada tipo de parámetro que se desea medir (CPU, Memoria y Tráfico de Red) se encuentra en un bloque de datos, debido a ello se debe hace la lectura de cada bloque de datos por separado.

Una vez realizada la separación de los datos, se deben hacer las comprobaciones previas para saber a qué campo corresponde cada columna (cada columna es reconocida y etiquetada según cuál sea el título de encabezamiento).

Finalmente se almacenan todos los valores en la base de datos, concretamente en la tabla st_lineas_pm. Los campos que se deben guardar son los mismos que en el caso de los ficheros del Performance Monitor de Windows, con la única diferencia del campo “xmaquina”.

En el caso de los ficheros de texto del comando <sar>de Unix, no se incluye el nombre del equipo donde tiene lugar la simulación, razón por la cual se introduce un campo en el formulario para que el usuario informe del nombre del equipo.

5.3.1.4 Clase <NombresMaquina.java>

Es el archivo Java asociado al formulario st_maquina , su función es la de actualizar los valores de memoria total de los equipos correspondientes a simulaciones del Performance Monitor de Windows.

En el caso de estos ficheros de texto, para el tipo de parámetro de Memoria, contiene la memoria disponible (Available MBytes), y el dato que se desea obtener es la memoria utilizada por la máquina (Used MBytes).

Para poder solucionarlo, se necesita que el usuario informe de la memoria total de cada equipo que esté en el fichero de texto del Performance Monitor de Windows.

La memoria utilizada se obtiene entonces de forma sencilla, restando de la memoria total la memoria disponible.

5.3.1.5 Clase <lectura_sql.java>

Es el archivo Java asociado al formulario st_datos_sim , su función es la de procesar los ficheros de texto, procedentes de Simulaciones Karat, que contienen los datos de una prueba de simulación.

El fichero de texto está organizado en columnas de valores, que no están delimitadas, pero que están en un bloque único de datos.

Además, la última línea del fichero de texto proporciona la información de las filas efectivas totales de datos que contiene el fichero.

Resulta un dato importante para saber el número de filas de datos que se deben leer, así como para tener una marca fiable de finalización del fichero.

Una vez realizada la separación de los datos, se deben hacer las comprobaciones previas para saber a qué campo corresponde cada columna (cada columna es reconocida y etiquetada según cuál sea el título de encabezamiento).

Finalmente se almacenan todos los valores en la base de datos, concretamente en la tabla **st_lineas_sim.**

5.3.2 Descripción de los mensajes

5.3.2.1 Módulo <st_error>

El módulo de mensajes <st_error> será usado para mostrar los mensajes en caso de error en alguna de las tareas llevadas a cabo mediante código JAVA.

5.3.2.1.1 Error debido a una excepción dentro del código JAVA

Mensaje:	001
Descripción:	Errores causados por excepciones JAVA
Icono:	
Texto:	Error: #1 Mostrará el texto que se guarde en el StackTrace de la excepción (IOException, ParseException, OTException, DAException o ,si es general, simplemente Exception) con el comando de JAVA getMessage()
Conj. botones:	Aceptar
Botón por defecto:	Aceptar

Mensaje: 001

Botón por defecto: Aceptar

(Para ver el resto de mensajes del módulo <st_error>, consultar el apartado 3.2.3.1 del documento de Análisis y Diseño en el Anexo)

5.3.2.2 Módulo <st_varios>

El módulo de mensajes <st_varios> será usado para mostrar los mensajes en diferentes situaciones, ya sea en caso de error o para informar al usuario, en alguna de las tareas llevadas a cabo mediante código JAVA.

5.3.2.2.1 Mensaje de finalización correcta del proceso de almacenamiento en la base de datos

Mensaje: 101

Descripción: Mensaje proceso finalizado ok

Icono:

Texto: Proceso finalizado correctamente.

Se han incorporado #1 registros.

Donde #1 será el número de registros guardados en la tabla **st_lineas_sim**

Conj. botones: Aceptar

Botón por defecto: Aceptar

(Para ver el resto de mensajes del módulo <st_error>, consultar el apartado 3.2.3.1 del documento de Análisis y Diseño en el Anexo)

5.3.3 Descripción de los objetos de negocio

5.3.3.1 Objeto de negocio <st_datos_pm>

El objeto de negocio <st_datos_pm> será el primer componente del proceso de adquisición de datos. Estará vinculado al formulario encargado de recopilar los datos de las pruebas de simulación procedentes del Performance Monitor de Windows.

Desde el formulario el usuario debe introducir los datos básicos para seleccionar el nombre de la prueba de simulación, el número de usuarios y el archivo de texto

generado por el Performance Monitor de Windows (con extensión *.csv o extensión *.tsv) con todos los datos de la prueba de simulación.

El objeto de negocio es de un único nivel, por tanto solo tiene una sección (o Panel tal como se denomina en Karat), y los campos proyectados de la tabla que se muestran en el objeto de negocio se denominan control.

Un control puede ser un campo obtenido directamente de una tabla o, si no está asociado a ninguna consulta, es equivalente a una variable donde se puede almacenar un cierto valor cuando sea necesario.

5.3.3.1.1 Paneles

Los paneles son las secciones en las que quedan divididos los grupos de controles que comparten una misma Consulta Base dentro del objeto de negocio.

El objeto de negocio <st_datos_pm> tiene un solo panel, en ese caso el panel principal en Karat siempre recibe el nombre de <HEADER>.

En el caso de que no haya una Consulta Base, no es necesario aplicar restricciones de campos de consulta igual a controles procedentes de otro panel.

En este panel no se necesita ninguna Consulta Base asociada, ya que los controles existentes hacen la función de variables contenedor, donde se almacenarán los datos que el usuario debe informar.

<HEADER>

Es el panel principal del objeto de negocio, y el que siempre existe por defecto cuando se crea un objeto de negocio nuevo. Identifica la cabecera del proceso principal y, generalmente, suele contener la mayor parte de los campos del formulario base.

xarchivo

Nombre:	xarchivo
Campo consulta:	<ninguno>
Tipo:	TEXT
Descripción (general):	Nombre del archivo *.csv o *.tsv
Mensaje:	Seleccione el fichero en formato texto y con extensión *.csv o *.tsv, con los datos de simulación
Etiqueta de entrada:	Nombre archivo
Etiqueta de salida:	Nombre archivo
Longitud máxima:	80
Obligatorio:	si

xusuariostotales

Nombre:	xusuariostotales
Campo consulta:	<ninguno>
Tipo:	INTEGER
Descripción (general):	Número de usuarios total
Mensaje:	Introduzca el número de usuarios total
Etiqueta de entrada:	Num. usu. total
Etiqueta de salida:	Num. usu. total
Obligatorio:	si

xnomprueba

Nombre:	xnomprueba
Campo consulta:	<ninguno>
Tipo:	TEXT
Descripción (general):	Nombre prueba simulación
Mensaje:	Introduzca el nombre de la prueba de simulación
Etiqueta de entrada:	Nombre prueba simulación
Etiqueta de salida:	Nombre prueba simulación
LookUp :	**(pestaña Miscelánea, botón Look up)** **Nombre**: pruebas_existentes **MDQO**: st_header **Título**: Pruebas existentes **Recipientes**: Control = Campo consulta <xnomprueba> =<xnomprueba> Lookup por defecto No abrir si vacio No abrir si una línea
Formularios dependientes:	**(pestaña Miscelánea, botón Obj.externos)** **Nombre**: st_prueba_nueva **DEFO**: st_prueba_nueva **Consulta inicial:** st_header **Título**: Prueba nueva **Modo**: Mantenimiento, Por defecto, Restricción **Mostrar acceso**: Ambos
Longitud máxima:	45
Obligatorio:	Si

5.3.3.1.2 Consultas dependientes

Las consultas dependientes son consultas internas del objeto de negocio que utiliza para

acceder a determinados datos almacenados en tablas del repositorio.

Una consulta dependiente necesita una consulta base a la tabla que deseamos consultar, posteriormente se pueden especificar las restricciones según los parámetros que condicionarán la búsqueda.

En la mayoría de casos el objetivo es relacionar un control del objeto de negocio (también puede ser una variable de entorno o una constante) con alguno de los campos obtenidos de la consulta dependiente. De esta manera, la consulta aplicará estas restricciones y obtendrá los resultados según el valor que le hemos introducido.

Las consultas dependientes son especialmente usadas como MDQOs en LOOKUPs o campos calculados, asociados a campos del objeto de negocio.

Nuestro objeto de negocio usará la siguiente consulta dependiente:

Consulta dependiente st_header

Consulta:	st_header
Restricciones:	Tipo de datos : control (Campo consulta) xnomprueba = (Valor) xnomprueba

5.3.3.1.3 Formularios dependientes

Los formularios dependientes los crearemos cuando deseemos acceder a otro formulario externo desde el formulario de nuestro proceso, cargándolo con la relación de datos que deseemos.

Para crear un formulario dependiente externo necesitaremos vincular un formulario creado y especificar unas dependencias de campos, control interno (campo de nuestro formulario) asociado a un control externo (campo del formulario que vamos a abrir).

Una vez creado un formulario dependiente, se podrá vincular como objeto externo a un campo (un control del objeto de negocio), de manera que cuando seleccionemos ese campo, podamos ejecutar el formulario dependiente si lo deseamos.

Usaremos el siguiente formulario dependiente:

Formulario dependiente st_prueba_nueva

Nombre:	st_prueba_nueva
Descripción:	Introducción de datos de prueba de simulación nueva
Formulario externo:	st_header_nuevo
Dependencias:	(Control interno) = (Control externo) xnomprueba = xnomprueba

(Para ver el resto de Objetos de Negocio, consultar el apartado 3.2.4 del documento de Análisis y Diseño en el Anexo)

5.3.4 Descripción de los formularios

5.3.4.1 Formulario <st_datos_pm>

El formulario <st_datos_pm> será la interfaz de entrada de nuevos datos en la aplicación, mediante un archivo de texto procedente de una Simulación del Performance Monitor de Windows. Son necesarios los siguientes datos:

- Nombre de la Prueba de Simulación :

  Para este campo el usuario tiene dos posibilidades de selección:

  1) Introducir una nueva prueba de simulación, con todos sus datos (implementado desde su correspondiente objeto de negocio, con el formulario dependiente <st_header_nuevo>).

  2) Seleccionar una prueba de simulación ya existente (implementado desde su correspondiente objeto de negocio con un LookUp), de manera que se muestren los valores de la tabla st_header y el usuario pueda seleccionar una de las pruebas de simulación.

- Número de usuarios

- Ruta y nombre del fichero de texto, con los datos de la Simulación del Performance Monitor de Windows (mediante un botón adyacente para seleccionar la ubicación del archivo).

Correspondencia con Aplicaciones en Karat Escritorio

- **Carpeta : Entrada de datos**
- **Objeto : Procesar Fichero de Simulación**
- **Clase del explorador : Selección**
  - **Acción : Seleccion_PM**

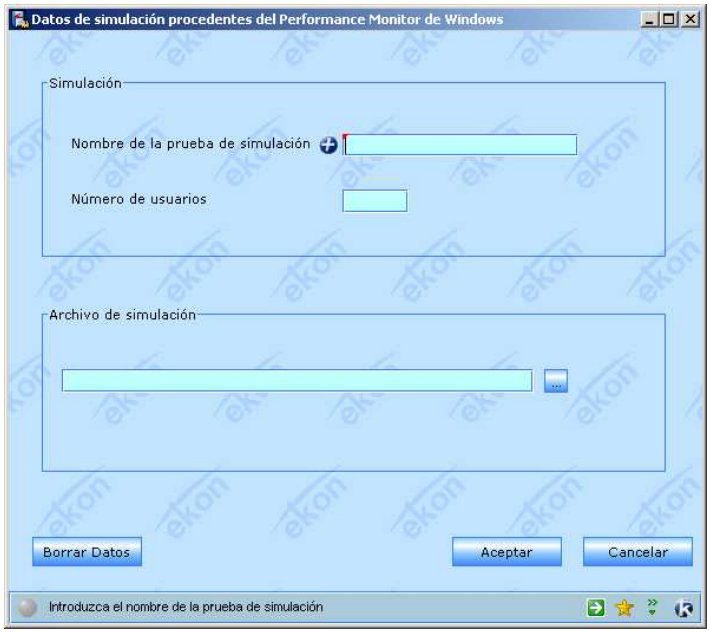

Figura 13: Formulario st_datos_pm

5.3.4.2 Formulario <st_graf_res_pm>

El formulario <st_graf_res_pm> muestra los resultados gráficos de una prueba de simulación, procedente de una simulación del Performance Monitor de Windows o del comando <sar> de Unix, que tiene que ser previamente seleccionada por el usuario.

Para seleccionarla el usuario dispone de cuatro desplegables que le permiten seleccionar todas las características que identifican a la prueba de simulación determinada.

Además de los resultados gráficos mostrados, también se visualizan (en un grid del formulario) los correspondientes valores numéricos que están representados gráficamente.

Correspondencia con Aplicaciones en Karat Escritorio

- **Carpeta : Resultados**
- **Objeto : Consultar**
- **Clase del explorador : Consultar**
  - **Acción : Resultados PerfMonitor y SAR**

Figura 14: Formulario st_graf_res_pm

(Para ver el resto de Formularios, consultar el apartado 3.2.5 del documento de Análisis y Diseño en el Anexo)

5.3.5 Descripción de las clases del explorador de Karat

5.3.5.1 Selección

Nombre: Selección

Descripción: Selección de Simulaciones Karat, Perfmonitor o SAR
Clase: Clase normal

Acciones:

Seleccion_SimKarat

- **Nombre** = Seleccion_SimKarat
- **Etiqueta** = Seleccionar Simulación Karat
- **Descripción**: Seleccionar Simulación Karat
- **Tipo de acción** = Acción GEN
  - **Servidor**: Otgen
  - **Formulario**: st_datos_sim
  - **Registro inicial** = None
  - **Consulta inicial** : None
  - **Modo** =
    - No menú
    - SDI

Seleccion_pm

- **Nombre** = Seleccion_PM
- **Etiqueta** = Seleccionar Simulación Performance Monitor
- **Descripción=** Selección Simulación Performance Monitor
- **Tipo de acción** = Acción GEN
  - **Servidor**: Otgen
  - **Formulario**: st_datos_pm
  - **Consulta inicial** : None
  - **Registro inicial** = None
  - **Modo** =
    - No menú
    - SDI

Seleccion_SAR

- **Nombre** = Seleccion_SAR
- **Etiqueta** = Seleccionar Simulación comando <sar> Unix
- **Descripción**: Selección Datos de simulación de Fichero procedente del comando <sar> de Unix
- **Tipo de acción** = Acción GEN
  - **Servidor**: Otgen
  - **Formulario**: st_datos_sar
  - **Registro inicial** = None
  - **Consulta inicial** : None
  - **Modo** =
    - No menú
    - SDI

(Para ver el resto de objetos de las Clases del Explorador de Aplicaciones, consultar el apartado 5.1.1 del documento de Análisis y Diseño en el Anexo)

5.3.6 Directrices para la construcción en Karat

Para desarrollar nuestra aplicación se han seguido los siguientes pasos:

- Crear los objetos de SDIC, del repositorio de Karat, necesarios para el desarrollo inicial: tablas y consultas base.

  Sincronización de las tablas lógicas para obtener sus correspondientes tablas físicas, para que existan en la base de datos y sean accesibles desde el SQL Server Management Studio del SQL Server 2005.

- Crear los objetos de negocio que servirán para gestionar los campos de las tablas anteriormente creadas. Tendrán como función configurar y definir los controles necesarios para las acciones y funcionalidades que se deben implementar.

- Crear los formularios, a partir de los respectivos objetos de negocio y vinculados a cada uno de ellos.

  Su objetivo es el de ser la interfaz gráfica entre el usuario y la aplicación desarrollada.

- Crear un archivo Java *.jar, englobando todos los archivos *.java que se hayan programado para realizar alguna de las funcionalidades, asociada a alguno de los formularios, mediante código Java.

  Estará vinculado a los formularios que tengan especificado como servidor primario ese archivo *.java.

6. Pruebas unitarias

6.1 Plan de pruebas

6.1.1 Pruebas de Entrada de Datos

En este primer apartado el objetivo es realizar pruebas de funcionamiento sobre la interacción con la base de datos.

Para el almacenamiento en base de datos se utilizan dos métodos:

- Por código, mediante un archivo Java asociado al formulario correspondiente.

  Se incluye, para cada tipo de fichero de texto que se puede procesar, una explicación de la implementación del código Java.

- Desde la aplicación, usando las herramientas Karat (consultas base, objetos de negocio, formularios y clases del explorador Karat).

6.1.1.1 Ficheros de texto del Performance Monitor de Windows

6.1.1.1.1 Pruebas realizadas

OBJETO DE NEGOCIO <st_datos_pm>

FORMULARIO < st_datos_pm >

Archivo java asociado al formulario : lectura.java

- Control <xnomprueba>
  - Funcionamiento LookUP (consulta dependiente <st_header>) :Pruebas existentes
  - Funcionamiento DEFO (abrir objeto externo → Formulario dependiente <st_header_nuevo>) :Prueba nueva
- Control <xarchivo>
  - Validación del funcionamiento de la introducción de la ruta del fichero directamente, sin usar el botón Examinar
  - Validación funcionamiento botón Examinar (método selectFile() de JAVA)
- Control <xusuariostotales>
  - Validación numérica, el valor introducido ha de ser de tipo numérico.

6.1.1.1.2 Análisis de resultados

Validación de almacenamiento correcto en la base de datos, mediante el formulario

<st_datos_pm> y su archivo Java asociado lectura.java, en la tabla **st_lineas_pm**.

Prueba 1

Fichero de texto de entrada: karat_8_20_u_all.csv

```
"(PDH-CSV 4.0) (Hora estándar romance)(-60)","\\ALMACEN\Memory\Available
MBytes","\\ALMACEN\Network Interface(Broadcom NetXtreme Gigabit Ethernet - Packet
Scheduler Miniport)\Bytes Received/sec","\\ALMACEN\Network Interface(Broadcom NetXtreme
Gigabit Ethernet - Packet Scheduler Miniport)\Bytes
Sent/sec","\\ALMACEN\Processor(_Total)\% Processor Time","\\srv-ren-ap-
01\Memory\Available MBytes","\\srv-ren-ap-01\Processor(_Total)\% Processor Time","\\srv-
ren-db-01\Memory\Available MBytes","\\srv-ren-db-01\Network Interface(Broadcom NetXtreme
Gigabit Ethernet)\Bytes Received/sec","\\srv-ren-db-01\Processor(_Total)\% Processor
Time","\\srv-ren-db-01\Network Interface(Broadcom NetXtreme Gigabit Ethernet)\Bytes
Sent/sec"
"10/22/2008 12:14:04.437","2670"," "," "," ","3443"," ","7403"," "," "," "
"10/22/2008
12:14:14.390","2606","19150.937734504671","10846.340879550482","17.734375000000004","341
4","5.4075235109717905","7376","2147.7983125643823","2.0408163265306145","2752.763846429
6233"
"10/22/2008
12:14:24.406","2604","58991.538158294301","22679.274914291975","7.9687500000000018","336
4","23.593750000000004","7288","116728.26557575045","10.374414976599066","255702.3107326
3066"
"10/22/2008
12:14:34.484","2601","142770.90204054269","47617.752042101551","8.7500000000000018","329
7","26.356589147286826","7263","158460.40501564997","6.4341085271317873","645456.4814611
8585"
"10/22/2008
12:14:44.406","2599","523164.65947284125","59701.001593721252","5.5468749999999956","328
9","30.738993710691819","7237","186534.71439902231","9.21259842519685","546610.994257191
2"
"10/22/2008
12:14:54.406","2596","619110.22045563487","61076.839336236219","6.5625000000000044","324
1","34.741784037558688","7233","202727.63031696531","5.9374999999999956","1253617.641162
8609"
-------------------------------------------
```

(no se muestra todo el texto del archivo debido a su longitud, para ver el archivo **karat_8_20_u_all.csv** completo, consultarlo en el anexo, en la carpeta **Perflog_SIMI**)

Resultado de almacenamiento en la base de datos

Nombre de prueba de simulación: prova_1
Número de usuarios totales: 34
Consulta SQL introducida en el programa SQL Server 2005
Desde el panel SQL del SQL Server 2005 ejecutar:

```
SELECT xarchivo, xfechahora, xmaquina, xnomprueba, xparam, xtipo,
xusuariostotales, xvalor
FROM st_lineas_pm
WHERE (xnomprueba = 'prova_1') AND (xusuariostotales = 34) AND (xmaquina =
'BBDD' OR xmaquina = 'Server LOCAL' OR xmaquina = 'SKOT001' OR
xmaquina = 'Server WEB')
```

Resultado de la consulta: 393 registros
Mensaje Karat obtenido desde la aplicación: Se han incorporado 393 registros

Prueba 2

Fichero de texto de entrada: Local_UAB06_tab.tsv

```
"(PDH-TSV 4.0) (Hora estándar romance)(-60)""\\UAB06\Procesador(_Total)\% de tiempo de
DPC"
"11/12/2008 12:26:41.716"    "1.6479682077219372e-007"
"11/12/2008 12:26:53.716"    "0.065104166666666657"
"11/12/2008 12:27:05.731"    "0.065019505851755532"
"11/12/2008 12:27:17.731"    "0.13020833333333331"
"11/12/2008 12:27:29.731"    "0"
"11/12/2008 12:27:41.731"    "0.13020833333333331"
"11/12/2008 12:27:53.731"    "0.065104166666666657"
"11/12/2008 12:28:05.731"    "0.065104166666666657"
"11/12/2008 12:28:17.731"    "0"
"11/12/2008 12:28:29.731"    "0"
"11/12/2008 12:28:41.731"    "0"
"11/12/2008 12:28:53.731"    "0"
"11/12/2008 12:29:05.731"    "0"
"11/12/2008 12:29:17.731"    "0"
"11/12/2008 12:29:29.731"    "0"
"11/12/2008 12:29:41.731"    "0.13020833333333331"
"11/12/2008 12:29:53.731"    "0.065104166666666657"
"11/12/2008 12:30:05.731"    "0.065104166666666657"
"11/12/2008 12:30:17.731"    "0.065104166666666657"
"11/12/2008 12:30:29.731"    "0.26041666666666663"
"11/12/2008 12:30:41.731"    "0.13020833333333331"
"11/12/2008 12:30:53.731"    "0.065104166666666657"
"11/12/2008 12:31:05.731"    "0.065104166666666657"
"11/12/2008 12:31:17.731"    "0.32552083333333337"
"11/12/2008 12:31:29.731"    "0.065104166666666657"
"11/12/2008 12:31:41.731"    "0"
-----------------------------------------------
```

(no se muestra todo el texto del archivo debido a su longitud, para ver el archivo **Local_UAB06_tab.tsv** completo, consultarlo en el anexo, en la carpeta **Perflog_SIMI**)

Resultado de almacenamiento en la base de datos

Nombre de prueba de simulación: prova_2
Número de usuarios totales: 37
Consulta SQL introducida en el programa SQL Server 2005
Desde el panel SQL del SQL Server 2005 ejecutar:

```
SELECT          xarchivo, xfechahora, xmaquina, xnomprueba, xparam, xtipo,
xusuariostotales, xvalor
FROM         st_lineas_pm
WHERE     (xnomprueba = 'prova_2') AND (xusuariostotales = 37) AND (xmaquina =
'UAB06')
```

Resultado de la consulta : 253 registros
Mensaje Karat obtenido desde la aplicación: Se han incorporado 253 registros

6.1.1.2 Ficheros de texto de Simulaciones Karat

6.1.1.2.1 Pruebas realizadas

OBJETO DE NEGOCIO <st_datos_sim>

FORMULARIO < st_datos_sim >

Archivo java asociado lectura_sql.java

- Control <xnomprueba>
  - Funcionamiento LookUP (consulta dependiente <st_header>) :Pruebas existentes
  - Funcionamiento DEFO (abrir objeto externo → Formulario dependiente <st_header_nuevo>) :Prueba nueva
- Control <xarchivo>
  - Validación del funcionamiento de la introducción de la ruta del fichero directamente por parte del usuario, sin usar el botón Examinar.
  - Validación del funcionamiento del botón Examinar (método selectFile() de JAVA)
- Control <xusuariostotales>
  - Validación numérica, el valor introducido ha de ser de tipo numérico
- Control <xususimulacion>
  - Validación numérica, el valor introducido ha de ser de tipo numérico

6.1.1.2.2 Análisis de resultados del archivo lectura_sql.java

Validación de almacenamiento correcto en la base de datos, mediante el formulario <st_datos_pm> y su archivo Java asociado lectura.java, en la tabla **st_lineas_sim.**

Prueba 1

Fichero de texto de entrada: FINAL_SQL_260_openForm_ANSI.rpt

```
sim_name                  script_name  session_id   xTipoMedida  xMedida             msg_delay
------------------------ ------------- ----------- ----------- -------------------- ---------
FINAL_SQL2_260_65_users_1        FINAL       1      Tiempo      Abrir formulario (ms) 0
FINAL_SQL2_260_65_users_2        FINAL       1      Tiempo      Abrir formulario (ms) 0
FINAL_SQL2_260_65_users_3        FINAL       1      Tiempo      Abrir formulario (ms) 0
FINAL_SQL2_260_65_users_4        FINAL       1      Tiempo      Abrir formulario (ms) 0
FINAL_SQL2_260_65_users_1        FINAL       1      Tiempo      Abrir formulario (ms) 15
FINAL_SQL2_260_65_users_2        FINAL       1      Tiempo      Abrir formulario (ms) 15
FINAL_SQL2_260_65_users_3        FINAL       1      Tiempo      Abrir formulario (ms) 0
FINAL_SQL2_260_65_users_4        FINAL       1      Tiempo      Abrir formulario (ms) 0
FINAL_SQL2_260_65_users_1        FINAL       1      Tiempo      Abrir formulario (ms) 31
FINAL_SQL2_260_65_users_2        FINAL       1      Tiempo      Abrir formulario (ms) 15
FINAL_SQL2_260_65_users_3        FINAL       1      Tiempo      Abrir formulario (ms) 15
.....................................

(16120 row(s) affected)
```

(no se muestra todo el texto del archivo debido a su longitud, para ver el archivo **FINAL_SQL_260_openForm_ANSI.rpt** completo, consultarlo en el anexo, en la carpeta **sql_SIM_ANSI**)

Resultado de almacenamiento en la base de datos

Nombre de prueba de simulación: prova_1
Número de usuarios por simulador: 7
Número de usuarios totales: 21

Consulta SQL, introducida en el programa SQL Server 2005
Desde el panel SQL del SQL Server 2005 ejecutar:

```
SELECT            xarchivo,   xidsesion,   xidtabla,   xnomprueba,   xnomscript,
xnomsimulacion, xparam, xtipo, xusuariostotales, xususimulacion, xvalor
FROM         st_lineas_sim
WHERE          (xnomprueba  =  'prova_1')  AND  (xusuariostotales  =  21)  AND
(xususimulacion = 7)
```

Resultado de la consulta: 16120 registros
Mensaje Karat obtenido desde la aplicación: Se han incorporado 16120 registros

Prueba 2

Fichero de texto de entrada: SIM-Duracion_ANSI.txt

```
sim_name       script_name              session_id     xTipoMedida  xMedida     xDurSeconds
-------------- ------------------------ ---------- ------------ ------------ --------------
ZVENTA                 VENTA_20081127          1        Tiempo      Duración (s) 45.323
ZVENTA                 VENTA_20081127          2        Tiempo      Duración (s) 46.153
ZVENTA                 VENTA_20081127          3        Tiempo      Duración (s) 43.743
ZVENTA                 VENTA_20081127          4        Tiempo      Duración (s) 44.517
ZVENTA                 VENTA_20081127          5        Tiempo      Duración (s) 44.890
ZVENTA                 VENTA_20081127          6        Tiempo      Duración (s) 44.317
ZVENTA                 VENTA_20081127          7        Tiempo      Duración (s) 43.907
ZVENTA                 VENTA_20081127          8        Tiempo      Duración (s) 43.737
ZVENTA                 VENTA_20081127          9        Tiempo      Duración (s) 43.540
ZVENTA                 VENTA_20081127          10       Tiempo      Duración (s) 43.927

(10 row(s) affected)
```

Resultado de almacenamiento en la base de datos

Nombre de prueba de simulación: prova_2
Número de usuarios por simulador: 5
Número de usuarios totales: 18

Consulta SQL, introducida en el programa SQL Server 2005.
Desde el panel SQL del SQL Server 2005 ejecutar:

```
SELECT            xarchivo,   xidsesion,   xidtabla,   xnomprueba,   xnomscript,
xnomsimulacion, xparam, xtipo, xusuariostotales, xususimulacion, xvalor
FROM         st_lineas_sim
WHERE          (xnomprueba  =  'prova_2')  AND  (xusuariostotales  =  18)  AND
(xususimulacion = 5)
```

Resultado de la consulta: 10 registros
Mensaje Karat obtenido desde la aplicación: Se han incorporado 10 registros

6.1.1.3 Ficheros de texto de Simulaciones del comando <sar> de Unix

6.1.1.3.1 Pruebas realizadas

OBJETO DE NEGOCIO <st_datos_sar>

FORMULARIO < st_datos_sar >

Archivo java asociado : lectura_sar.java

- Control <xnomprueba>
  - Funcionamiento LookUP (consulta dependiente <st_header>) :Pruebas existentes
  - Funcionamiento DEFO (abrir objeto externo → Formulario dependiente <st_header_nuevo>) :Prueba nueva
- Control <xarchivo>
  - Validación del funcionamiento de la introducción de la ruta del fichero directamente, sin usar el botón Examinar
  - Validación funcionamiento botón Examinar (método selectFile() de JAVA)
- Control <xusuariostotales>
  - Validación numérica, el valor introducido ha de ser de tipo numérico

6.1.1.3.2 Análisis de resultados del archivo lectura_sar.java

Validación de almacenamiento correcto en la base de datos, mediante el formulario <st_datos_sar> y su archivo Java asociado lectura_sar.java, en la tabla **st_lineas_pm.**

Prueba 1

Fichero de texto de entrada: FINAL_SQL_260_openForm_ANSI.rpt

Fichero de texto de entrada : arxiu_1_150.txt

Obtención del fichero, mediante el comando <sar> de Unix :

sar –n DEV –u –r –o fichero_ok 1 150

sar –n DEV –u –r –f fichero_ok > arxiu_1_150.txt

```
Linux 2.6.13-15-default (linux)      14/05/09

11:55:11          CPU     %user     %nice   %system   %iowait     %idle
11:55:15          all      0,25      0,00      0,50      0,50     98,75
11:55:19          all      0,00      0,00      0,00      0,00    100,00
11:55:23          all      0,00      0,00      0,25      0,00     99,75
11:55:27          all      0,00      0,00      0,25      0,00     99,75
11:55:31          all      0,00      0,00      0,25      0,00     99,75
11:55:35          all      0,00      0,00      0,25      0,00     99,75
11:55:39          all      0,00      0,00      0,25      0,00     99,75
11:55:43          all      0,00      0,00      0,00      0,00    100,00
11:55:47          all      0,00      0,00      0,25      0,00     99,75
```

```
.................................................
```

(no se muestra todo el texto del archivo debido a su longitud, para ver el archivo **arxiu_1_150.txt** completo, consultarlo en el anexo, en la carpeta **SAR_SIM**)

Resultado de almacenamiento en la base de datos

Nombre de prueba de simulación: prova_1
Nombre de la maquina: UAB06
Número de usuarios totales: 27

Consulta SQL
Desde el panel SQL del SQL Server 2005 ejecutar:

```
SELECT          xarchivo, xfechahora, xmaquina, xnomprueba, xparam, xtipo,
xusuariostotales, xvalor
FROM         st_lineas_pm
WHERE           (xnomprueba = 'prova_1') AND (xmaquina = 'UAB06') AND
(xusuariostotales = 27)
```

Resultado de la consulta: 1296 registros
Mensaje Karat obtenido desde la aplicación: Se han guardado 1296 registros

Prueba 2

Fichero de texto de entrada: arxiu_2_120.txt

```
Linux 2.6.13-15-default (linux)      14/05/09

11:55:11          CPU     %user     %nice   %system   %iowait     %idle
11:55:15          all      0,25      0,00      0,50      0,50     98,75
11:55:19          all      0,00      0,00      0,00      0,00    100,00
11:55:23          all      0,00      0,00      0,25      0,00     99,75
11:55:27          all      0,00      0,00      0,25      0,00     99,75
11:55:31          all      0,00      0,00      0,25      0,00     99,75
11:55:35          all      0,00      0,00      0,25      0,00     99,75
11:55:39          all      0,00      0,00      0,25      0,00     99,75
11:55:43          all      0,00      0,00      0,00      0,00    100,00
11:55:47          all      0,00      0,00      0,25      0,00     99,75
11:55:51          all      0,00      0,00      0,51      0,25     99,24
.................................................
```

(no se muestra todo el texto del archivo debido a su longitud, para ver el archivo **arxiu_2_120.txt** completo, consultarlo en el anexo, en la carpeta **SAR_SIM**)

Resultado de almacenamiento en la base de datos

Nombre de prueba de simulación: prova_2
Nombre de la maquina: UAB06
Número de usuarios totales: 38

Consulta SQL
Desde el panel SQL del SQL Server 2005 ejecutar:

```
SELECT          xarchivo, xfechahora, xmaquina, xnomprueba, xparam, xtipo,
xusuariostotales, xvalor
FROM         st_lineas_pm
WHERE           (xnomprueba = 'prova_2') AND (xmaquina = 'UAB06') AND
(xusuariostotales = 38)
```

Resultado de la consulta: 804 registros
Mensaje Karat obtenido desde la aplicación: Se han guardado 804 registros

6.1.2 Pruebas con las Pruebas de Simulación

OBJETO DE NEGOCIO <st_header_nuevo>

FORMULARIO < st_header_nuevo >

Validación de almacenamiento correcto en la base de datos, mediante la consulta base <st_header>, en la tabla **st_header**.

6.1.3 Pruebas con la Actualización de memoria

OBJETO DE NEGOCIO <st_maquina>

FORMULARIO < st_maquina>

- Control <xarchivo>
  - Validación del funcionamiento de la introducción de la ruta del fichero directamente, sin usar el botón Examinar
  - Validación de funcionamiento del botón Procesar archivo (archivo NombresMaquina.java).

    Se muestran, mediante un mensaje Karat, aquellos equipos que están dentro del fichero de texto pero no se encuentran en la tabla **st_lineas_pm**.

6.1.4 Pruebas con los resultados gráficos

OBJETO DE NEGOCIO <st_graf_res_pm>

FORMULARIO < st_graf_res_pm >

Para los siguientes controles del panel <HEADER>:

- xmaquina - Nombre de la Maquina Simulada en cada registro
- xnomprueba - Nombre de la Prueba de Simulación
- xparam - Parámetro de rendimiento del performance monitor o comando <sar>
- xtipo- Tipo de parámetro de rendimiento del performance monitor o comando <sar>

Validación de visualización correcta de los desplegables definidos en el objeto de negocio (mediante la opción Miscelánea, pestaña Valores) de cada control.

En el panel <subpanel2> Validación de visualización correcta de los controles (procedentes de la consulta base <st_calculados>) en el Gráfico y también en el grid.

Son campos proyectados de la tabla **st_lineas_pm**, con las restricciones de los controles introducidos en el panel <HEADER>.

OBJETO DE NEGOCIO <st_graf_res_sim>

FORMULARIO < st_graf_res_sim >

Para los siguientes controles del panel <HEADER>:

- xnomscript - Nombre del script de simulación
- xnomprueba - Nombre de la Prueba de Simulación
- xnomsimulacion - Nombre de la simulación Karat
- xparam - Nombre del parámetro de rendimiento
- xtipo- Tipo de parámetro de rendimiento del performance monitor

Validación de visualización correcta de los desplegables definidos en el objeto de negocio (mediante la opción Miscelánea, pestaña Valores) de cada control.

En el panel <subpanel_3> validar que se visualizan correctamente los controles (procedentes de la consulta base < st_calculados_sim>), en el Gráfico y también en el Grid.

Son campos proyectados de la tabla **st_lineas_pm**, con las restricciones de los controles introducidos en el panel <HEADER>.

6.1.5 Pruebas con los resultados numéricos

6.1.5.1 Resultados Generales

OBJETO DE NEGOCIO <st_datos_pm_listado>

FORMULARIO < st_datos_pm_listado >

Validación de visualización correcta: los datos de la consulta base <st_lineas_pm> son mostrados correctamente, proyectando los campos de la tabla **st_lineas_pm**.

OBJETO DE NEGOCIO <st_datos_sim_listado>

FORMULARIO < st_datos_sim_listado >

Validación de visualización correcta: los datos de la consulta base <st_lineas_pm> son mostrados correctamente, proyectando los campos de la tabla **st_lineas_pm**.

OBJETO DE NEGOCIO <st_header_nuevo>

FORMULARIO <st_header_abrir>

Validación de visualización correcta de los campos de la de la tabla st_header, mediante la consulta base <st_header>.

6.1.5.2 Resultados detallados según el número de usuarios

OBJETO DE NEGOCIO <st_detalle_pm>

FORMULARIO < st_detalle_pm >

Para los siguientes controles del panel <HEADER>:

- xmaquina - Nombre de la Maquina Simulada en cada registro
- xnomprueba - Nombre de la Prueba de Simulación
- xparam - Parámetro de rendimiento del performance monitor o del comando <sar>
- xtipo- Tipo de parámetro de rendimiento del performance monitor o del comando <sar>.
- xusuariostotales - Número total de usuarios simulados

Validar que se muestran correctamente los desplegables definidos en el objeto de negocio (mediante la opción Miscelánea, pestaña Valores) de cada control.

En el panel <subpanel2> comprobación que son mostrados correctamente los controles (procedentes de la consulta base <st_lineas_pm>), con las restricciones de los controles introducidos en el panel <HEADER>.

OBJETO DE NEGOCIO <st_detalle_sim>

FORMULARIO < st_detalle_sim >

Para los siguientes controles del panel <HEADER>:

- xnomscript - Nombre del script de simulación
- xnomprueba - Nombre de la Prueba de Simulación
- xnomsimulacion - Nombre de la simulación Karat
- xparam - Nombre del parámetro de rendimiento
- xtipo- Tipo de parámetro de rendimiento
- xusuariostotales - Número total de usuarios simulados

Validar que se muestran correctamente los desplegables definidos en el objeto de negocio (mediante la opción Miscelánea, pestaña Valores) de cada control.

En el panel <subpanel_3> validar que son mostrados correctamente los controles (procedentes de la consulta base <st_lineas_sim>), con las restricciones de los controles introducidos en el panel <HEADER>.

6.2 Problemas encontrados y soluciones aplicadas

6.2.1 Problemas Generales

- **Formato Unicode.**

  Los archivos de texto no pueden estar en formato Unicode, ya que utiliza 2 bytes (16 bits) para codificar cada carácter.

  Al hacer la lectura como fichero de texto, en Java lo que se retorna es el carácter ASCII, que utiliza 1 byte (8 bits) para codificar cada carácter.

  Java solamente soporta el estándar de Unicode UTF-8 cuando se escriben o leen strings con las clases InputStreamReader u OutputStreamReader.

  Estas son clases que tienen como función la lectura de ficheros binarios, basados en streams de entrada o salida formada por bytes, y que son decodificados a tipo carácter usando un determinado charset.

  En nuestro caso, son archivos de texto plano, sin caracteres especiales que necesiten del formato Unicode para poder ser codificados.

- **La función matches, de la clase String, sirve solo para expresiones regulares de Java.**

  Por tanto, no puede contener paréntesis (así como tampoco otros caracteres), y no funciona para valores como pueden ser “Processor(_Total)”.

  Como alternativa se puede usar la función String.equals, que sirve para comprobar si el string es igual al objeto Java que se le pasa como parámetro.

  El resultado es también óptimo porque, finalmente, un String no deja de ser también un objeto Java.

- **El carácter “\” en java se escribe ”\\” .**

  El motivo es porque el carácter barra invertida es un carácter reservado dentro de un string, utilizado para indicar el final de un String.

6.2.2 Ficheros de texto de Simulaciones del Performance Monitor de Windows

- **Debido a que la coma puede ser carácter delimitador de columnas, el separador decimal tiene que ser el punto “.”**

  En los ficheros del Performance Monitor de Windows puede ser que el separador decimal sea la coma o el punto.

En java, el separador decimal es el punto, por ello se debe cambiar en el String cada aparición de ",", por un ".", cuando se lee el valor de cada columna.

- **Los nombres de los equipos no se corresponden con los nombres que almacena el Performance Monitor de Windows en sus datos de monitorización de rendimiento guardados en su fichero de texto.**

  Los datos correctos que han de ser almacenados son los siguientes:

  srv-ren-ap-01 → SKOT001
  srv-ren-bd-01 → BBDD
  ALMACEN → Server Local (Memory, CPU)
  ALMACEN → Server Web (Network)

- **Server Web**

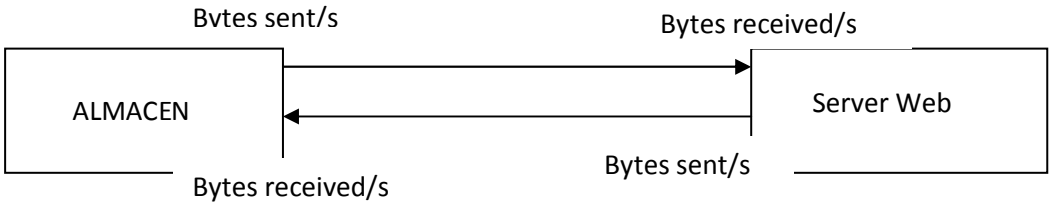

Figura 15: Funcionamiento de la medición del Tráfico de Red en la máquina Server Web (Servidor Web)

Se implementa así para tener el dato de tráfico de Red Total, debido a que el Servidor posee un procesador de más de un núcleo, y los datos pueden ser engañosos dependiendo del balanceo de carga de trabajo que se produzca.

Se han de invertir los parámetros de Bytes sent/s (Bytes enviados/s) tienen que pasar a ser los Bytes received/s (Bytes recibidos/s) y igualmente al revés, Bytes received/s (Bytes recibidos/s) tienen que pasar a ser los Bytes sent/s (Bytes enviados/s)

7. Conclusiones

Una vez finalizado el Proyecto final de carrera, una de las primeras cosas que creo se debe mencionar son las expectativas iniciales, por las cuales elegí realizarlo en empresa:

- Conocer el funcionamiento de una empresa de desarrollo software desde dentro
- Conocer el proceso real de planificación, desarrollo e implementación de un proyecto informático. Además, que sea un proyecto que trate una aplicación existente y real dentro de la empresa, enfocada hacia una posible distribución comercial, ya sea a nivel interno o externo.
- Encontrar un punto medio entre la posibilidad de que el alumno desarrolle y pueda intervenir o hacer modificaciones, si es el caso, en el proyecto, y la aplicación que la empresa necesita.

Respecto a la aplicación desarrollada, creo que se puede afirmar que se han cumplido todos los objetivos planteados en un principio.

Lo que se pretendía obtener era una aplicación capaz de leer todos los datos de un fichero de texto, que contenía datos de texto y también datos numéricos, correspondiente a una prueba de simulación.

Una vez completado este paso, la aplicación debía ser capaz de gestionar y mostrar todos esos datos, tanto gráficos como numéricos.

El resultado final ha sido un programa capaz de procesar hasta tres tipos de fichero de texto diferentes, operación llevada a cabo mediante código Java, hecho que ha comportado un considerable proceso de pruebas y depuración de errores sobre este código.

Una vez los ficheros ya fueron procesados correctamente, el resultado del almacenamiento en la base de datos ya fue óptimo, quedó únicamente la última fase: la implementación de la aplicación.

La fase de implementación fue la más sencilla, ya que todo el diseño de la aplicación ya estaba realizado, y únicamente quedaba pulir el aspecto exterior de la aplicación.

El proyecto ha sido completado con todos los requerimientos iniciales y objetivos que había planteado la empresa.

Aún así, existe la posibilidad de posibles ampliaciones, sobre todo en lo que concierne a aumentar el tipo de ficheros de texto que puede procesar la aplicación.

Esa es la metodología que se siguió para procesar los ficheros procedentes del comando <sar> de Unix.

Primeramente se intentó aprovechar la misma base de datos que se utilizaba para los ficheros del Performance Monitor de Windows, y fue posible porque los campos eran los mismos (a excepción del nombre del equipo, información que se obtiene del usuario).

El segundo punto importante es el formato del fichero de texto: como más parecido sea el formato al de los ficheros de texto aceptados, más fácil de aprovechar será el código fuente Java ya implementado.

En el caso de los ficheros del comando <sar> de Unix tiene un formato parecido al de los ficheros de simulaciones Karat, y eso ayudó a que parte de su código Java fuera muy similar.

En resumen, para facilitar que un fichero de texto sea procesable habría que tener en cuenta estos dos puntos, pero podría hacerse con cualquier fichero de texto con una estructura bien definida de la información y como contenido valores de una prueba de simulación.

La única pequeña desviación respecto a la planificación inicial se produjo al añadir como posibilidad poder procesar un tercer tipo de fichero de texto.

Se propuso como mejora o ampliación también poder procesar ficheros de texto procedentes de un sistema operativo Linux, si existía el tiempo suficiente una vez la aplicación estuviese funcionando para los dos tipos de fichero de texto prioritarios.

En el principio del proyecto no era seguro que existiese el tiempo suficiente para incorporar ese tercer tipo de fichero de texto, debido a eso se prefirió asegurar que la aplicación estaba completada para los dos tipos de ficheros de texto de simulaciones Karat y del Performance Monitor de Windows.

Una vez visto el resultado, realmente resulta claro que desde un principio se podía haber establecido que se podían procesar el tercer tipo de ficheros de texto, procedentes del sistema operativo Linux, que finalmente se ha implementado ya hacia la fase final del proyecto.

Aún así, el resultado final ha sido satisfactorio, porque lo único que ha supuesto es un pequeño cambio en la planificación inicial, y finalmente se ha conseguido que la aplicación pueda procesar también los ficheros de texto procedentes de Linux.

El trabajo realizado en la empresa me ha servido especialmente para conocer el funcionamiento desde dentro de una empresa dedicada al desarrollo informático, concretamente al desarrollo software.

Es algo que realmente valoro como muy positivo, ya que hasta ahora sí había tenido experiencias laborales, pero nunca en una empresa directamente relacionada con mis estudios de Ingeniería Técnica en Informática de Sistemas.

También me ha servido para conocer las particularidades y el proceso de desarrollo de un proyecto informático, algo que hasta ahora sólo había estudiado de manera teórica.

Poder aplicarlo en la práctica realmente ayuda para saber cómo resolver los problemas que van surgiendo, y como se deben realizar todos los diferentes procedimientos y fases del proyecto.

Espero que esta experiencia pueda servirme en un futuro próximo para enfocar totalmente mi vida laboral hacia el sector de la Informática y las Tecnologías de la Información.

Bibliografía

Libros:

- Java 2, guía esencial

García-Bermejo Giner, José Rafael
Editorial Prentice Hall, Madrid 1999

Documentos de ayuda internos de karat:

- ¿karat 8.0 & eclipse? ... ¡Pero si es muy fácil!

Tutorial de ayuda e iniciación al funcionamiento de Eclipse JDK.
Ubicación local: C:\Archivos de programa\karat\doc\kdev\kdev0001.html

- Karat data acces

Documentación de la clase Data Acces de Karat, clase responsable de proporcionar el acceso a datos, además de ofrecer funciones específicas de karat, como por ejemplo el manejo de consultas, consultas base y tablas
Ubicación local: C:\Archivos de programa\karat\doc\guide\da\index.html

- API de Karat

Especificación API para la plataforma Karat, versión 8.0 build 650 (API: Application Programming Interface)
Ubicación local: C:\Archivos de programa\karat\doc\api\index.html

- Karat Form starter kit

Tutorial de ayuda y ejemplos del objeto Karat Formulario (Form)
Uso, definición y programación
Ubicación local: C:\Archivos de programa\karat\doc\ksk\fm\index.html

- Karat Business Object starter kit

Tutorial de ayuda y ejemplos del Objeto de Negocio (BO)
Uso, definición y programación
Ubicación local: C:\Archivos de programa\karat\doc\guide\bo\index.html

Enlaces a fuentes electrónicas sobre lenguaje de programación JAVA:

- Programación en Java

De Wikilibros, colección de libros de texto de contenido libre.
http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Java

- The Java tutorials

Tutoriales Java de Sun Microsystems
http://java.sun.com/docs/books/tutorial/

- Aprenda java como si estuviera en primero

Javier García de Jalón, José Ignacio Rodríguez, Iñigo Mingo,…et altri

Escuela Superior de Ingenieros Industriales de San Sebastián, Universidad de Navarra
San Sebastián, Febrero de 2000
http://mat21.etsii.upm.es/ayudainf/aprendainf/Java/Java2.pdf

Enlaces a fuentes electrónicas sobre el Monitor del sistema de Windows:

- Administración de Windows xp : Practica de monitorización del sistema
  Universidad de Málaga
  http://www.lcc.uma.es/~valverde/Practica2.pdf

- Cómo crear un registro mediante el Monitor del sistema de Windows
  http://support.microsoft.com/kb/248345

- Trabajar con ficheros CSV : Tutorial
  http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=CSV
  Documento en formato *.pdf
  http://www.adictosaltrabajo.com/tutoriales/pdfs/CSV.pdf

- Librería java, de libre distribución, para trabajar con ficheros CSV (Comma-Separated Value)
  http://www.csvreader.com/

Enlaces a fuentes electrónicas sobre el comando <sar> de Unix:

- Utilización del comando <sar> de Unix
  http://www.computerhope.com/unix/usar.htm

- Documentación, manual de usuario y tutoriales del comando <sar> de Unix
  http://pagesperso-orange.fr/sebastien.godard/tutorial.html

- Visualización gráfica de los resultados del comando <sar> de Unix
  http://www.linux.com/archive/articles/114224

Otros enlaces:

- Enciclopedia libre y cooperativa entre usuarios de internet
  http://es.wikipedia.org/

- Características técnicas del servidor IBM System x3500
  http://www.ibm.com/mx/systems/x/tower/x3500/features.phtml

Glosario de Términos

Glosario de términos empleados en el documento.

LookUp

Los LookUp se asocian a campos de los objetos de negocio que estén mostrados en el formulario, se crean a partir de una consulta dependiente y su función es permitir al usuario abrir un desplegable con las opciones filtradas por la consulta base, permitiendo seleccionar el valor del campo.

Consulta dependiente

Están basadas en una consulta base, sobre la cual se hace un segundo filtrado de información dentro del objeto de negocio, de esta forma podemos acceder a datos específicos de la consulta base.

Formulario dependiente

Los formularios dependientes ofrecen la posibilidad de acceder a un formulario externo desde el formulario de nuestro proceso, cargándolo con la relación de datos que deseemos.

Una vez creado un formulario dependiente, se podrá vincular como objeto externo a un campo (un control del objeto de negocio), de manera que cuando seleccionemos ese campo, podamos ejecutar el formulario dependiente si lo deseamos.

Grid

Es una tabla de datos numéricos, utilizada en los formularios Karat para visualizar valores numéricos en un formato similar al de una hoja de cálculo.

Control, en un objeto de negocio

Un control en un objeto de negocio es el equivalente a un campo, con la particularidad de que puede tomar el valor de muchas fuentes distintas: un valor retornado por una consulta a la base de datos, una constante, un valor calculado, una lista de valores predefinida o procedente de una consulta,….

Firmado: Albert Moreso Ventura

Sabadell, Junio de 2009