ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

UN MEDIDOR DE RENDIMIENTO DE SERVIDORES DE BASES DE DATOS RELACIONALES

Realizado por

JOSE ANTONIO JAMILENA DAZA

Dirigido por

ANTONIO CÉSAR GÓMEZ LORA

Departamento

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA

MÁLAGA, (SEPTIEMBRE 2009)

UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Reunido el tribunal examinador en el día de la fecha, constituido por:

Presidente/a Dº/Dª. _____

Secretario/a Dº/Dª._____

Vocal Dº/Dª. _____

para juzgar el proyecto Fin de Carrera titulado:

UN MEDIDOR DE RENDIMIENTO DE SERVIDORES DE BASES DE DATOS RELACIONALES.

del alumno Dº. Jose Antonio Jamilena Daza

dirigido por Dº. Antonio César Gómez Lora

ACORDÓ POR _____

OTORGAR LA CALIFICACIÓN DE _____

Y PARA QUE CONSTE, SE EXTIENDE FIRMADA POR LOS COMPARECIENTES DEL TRIBUNAL, LA PRESENTE DILIGENCIA.

Málaga, a _____ de _____ del 200___

El/La Presidente/a	El/La Secretario/a	El/La Vocal
Fdo:	Fdo:	Fdo:

Un medidor de rendimiento de servidores de bases de datos relacionales

Realizado por Jose Antonio Jamilena Daza
Dirigido por Antonio César Gómez Lora

CONTENIDO

TABLA DE ILUSTRACIONES

1 INTRODUCCIÓN

La mayoría de los sistemas gestores de bases de datos actuales disponen de herramientas propias o de terceros destinadas a medir el estado de salud y el rendimiento del propio sistema. Estos sistemas con capaces de proporcionar un extenso y valioso conjunto de información para administradores y auditores. Adicionalmente existen plataformas software destinadas a construir medidores de calidad de servicio, aunque, hoy por hoy, con una orientación muy específica hacia redes y servicios web.

Con este rico abanico de herramientas disponibles, y la necesidad de medir la calidad de los servicios prestados, es cada vez más común encontrar servidores dentro de una empresa u organización destinados a analizar la disponibilidad y rendimiento de los servicios ofertados (entendiendo el acceso a una base de datos como uno de estos servicios). Desgraciadamente estos servidores suelen estar situados estratégicamente en zonas clave de las intranets o de las infraestructuras de comunicación. Cuando los servicios están abiertos al exterior sólo se analiza el entorno próximo del servidor y nunca el entorno del cliente.

Para un usuario simple que accede desde su equipo a un servidor de bases de datos el rendimiento viene marcado por el rendimiento de todos y cada uno de los equipos, dispositivos y enlaces involucrados en el procesamiento y el transporte de la información. En una sesión común esto puede significar la participación de decenas, cientos e incluso miles de elementos hardware y software participantes, la mayoría de los cuales son transparentes al usuario. Y en caso de ser visible alguno de estos participantes generalmente no se tiene la capacidad ni los permisos suficientes para analizar su rendimiento individual.

Se nos plantea pues la necesidad de que un usuario pudiera ejecutar un sencillo programa que le permitiera medir el rendimiento de determinadas tareas en un servidor de bases de datos. Este sistema analizará el rendimiento de estas tareas de forma periódica siguiendo una planificación dada para cumplir su objetivo principal, que es detectar la aparición de posibles problemas. Aunque el diagnóstico del problema queda

fuera de los objetivos de esta aplicación, sí es cierto que se añadirá la capacidad de clasificar las diferentes tareas según características particulares para agrupar sus resultados y así asistir al usuario en la interpretación de los datos y estadísticos para que se pueda realizar un diagnóstico. Este programa sencillo tiene un doble destino: primero el ser utilizado por usuarios aislados; y segundo el poder ser instalado en múltiples equipos y permitir recolectar posteriormente sus datos de forma sencilla (preferiblemente en forma de un archivo simple).

1.1 OBJETIVOS

Los objetivos que este proyecto plantea son los que siguen:

- Elaborar un software que ejecute sobre sistemas gestores de bases de datos relacionales, bajo una planificación dada, unos scripts de prueba, para a partir de los tiempos de ejecución generar indicativos sobre su comportamiento.
- Construir una interfaz gráfica que sea capaz de recoger los datos de la herramienta descrita anteriormente y generar unas estadísticas que sean mostradas de forma visual para que el usuario las interprete. Dichas estadísticas se agruparan según los diferentes tipos de scripts lanzados.
- Generar un conjunto base de scripts que puedan ser clasificados en conjuntos no disjuntos de categorías o propiedades. Con esto se consiguen estadísticas según las posibles características evaluar.

1.2 METODOLOGÍAS Y FASES DE TRABAJO

En el desarrollo del proyecto se empleará el paradigma de programación orientado a objetos y para modelar el software el Lenguaje Unificado de Modelado o UML, iniciales que corresponden al acrónimo inglés *Unified Modeling Language*.

El modelo de desarrollo será el Desarrollo Iterativo Incremental [1].

En cuanto a las fases en las que se dividirá el desarrollo software éstas serán:

- Una prospección inicial del desarrollo en el que barajaremos posibles lenguajes para desarrollar el proyecto, tecnologías a emplear, etc. Así también como la obtención de los objetivos que ha de cumplir el desarrollo software que formará el proyecto. Tras la cual habremos obtenido el análisis de requisitos.
- Se realizará la especificación de los distintos componentes que formarán parte del desarrollo software.
- Una vez que tenemos los componentes que debemos implementar pasaremos a una fase de diseño de la arquitectura de la solución. En esta fase del desarrollo se realizará el diseño de los distintos componentes del sistema software, así como el diseño de la jerarquía de clases de los distintos componentes del desarrollo software.
- Posteriormente pasaremos a la implementación del diseño. En esta fase se puede llegar a tener que hacer cambios en el diseño si la naturaleza del desarrollo así lo exige.
- Finalmente se realizarán las pruebas oportunas para verificar el correcto comportamiento del sistema.

Durante todo el proceso se seguirá el Desarrollo Iterativo Incremental [1], se irán obteniendo prototipos de desarrollo sobre los que se irán haciendo pruebas de funcionamiento y de completitud de requisitos.

Durante todas estas fases se llevará un amplio proceso de documentación, tanto para la escritura de la memoria final como para guardar el conocimiento adquirido durante el desarrollo.

1.3 HERRAMIENTAS DE DESARROLLO

El desarrollo se hará sobre el lenguaje de programación Java, concretamente haciendo uso de JDK 1.6.0_14-b08 de Sun Microsystems, la última versión de este leguaje de programación orientado a objetos [2].

El interfaz elegido para el desarrollo será NetBeans 6.5, un entorno integrado de desarrollo especializado en Java [3].

Todo el proyecto se guardará con el sistema de control de versiones Subversion. Este se encarga de mantener históricos de ficheros durante los desarrollos software [4]. Subversión se instala como un software servidor al que se accede mediante un cliente. El cliente elegido es TortoiseSVN, un cliente que se integra tanto en la shell del sistema operativo como en el propio NetBeans [5].

Se realizará el diseño de la aplicación sobre Enterprise Architect y sobre un complemento específico de NetBeans para estos menesteres. Enterprise Architect es una herramienta CASE, Computer Aided Software Engineering o Ingeniería de Software Asistida por Ordenador, y en el diseño de este proyecto en cuestión se guardarán los datos de diseño en un sistema de bases de datos MySQL [6][7]. En cuanto al componente de NetBeans lo almacenará con estructura de archivos de proyecto de NetBeans.

El proyecto tendrá integrada su propia base de datos de estadísticas. Para dicho cometido se empleará SQLite, un sistema gestor de bases de datos empotrado, rápido y de poco peso en memoria [8]. Para comunicarse con Java se empleará la librería SQLiteJDBC [9]. Para facilitar el diseño en dicho sistema gestor de bases de datos se empleará la herramienta SQLite Management Studio, una herramienta para gestión gráfica de ficheros SQLite [10].

Para velar por la integridad y conocer inequívocamente los fallos que pueden acaecer, tanto en la ejecución normal como en las sesiones de pruebas, se empleará Apache Log4J. Apache Log4J es un sistema de diario de ejecución para aplicaciones Java [11].

Dicho sistema nos da la capacidad de generar ficheros de diario de ejecución con las trazas de la misma e incluso la capacidad de que llegado el caso, si el sistema al que estamos evaluando se llegara a colapsar, enviar un email advirtiendo este hecho.

Para el componente encargado de mostrar las estadísticas de las pruebas se empleara un interfaz gráfico Java implementado con Swing. Como sistema visualizador de datos y estadísticas emplearemos la biblioteca JFreeChart. JFreeChart es un sistema de generación de graficas para el entorno grafico Swing de Java [12].

Las conexiones entre el proyecto y los diferentes gestores de bases de datos se harán mediante la tecnología JDBC (Java Data Base Connectivity). JDBC es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Toda la planificación temporal del proyecto será llevada acabo con la ayuda de OpenProj [13]. Y la documentación se generará a partir de Enterprise Architect y se editará el texto con OpenOffice [14] y con Microsoft Visio [15] los diagramas que fueran necesarios.

1.4 CALIDAD

La calidad debería ser un principio fundamental en el desarrollo software, y en especial si el desarrollo lo estamos haciendo en un lenguaje como Java que dispone de multitud de estándares y herramientas de auditoría que nos permiten garantizar que el código implementado cumple todos los requisitos como para considerarlo estable y seguro.

Para la completitud de los propósitos de calidad el desarrollo se ha empleado herramientas de verificación de código estático y de chequeo de código.

Para la verificación de código estático se ha empleado PMD. PMD analiza el código fuente de Java y busca posibles problemas como:

- Posibles errores: sentencias try / catch / finally / switch vacías.
- Código de muertos: variables locales, parámetros y métodos privador sin usar.
- Optimiza código: controla el despilfarro de uso de String / StringBuffer.
- Detecta expresiones excesivamente complejas, declaraciones innecesaria o bucles sin condiciones de salida.
- Detecta código duplicado. El copiar / pegar en el código significa copiar / pegar los errores.

Para garantizar que la escritura de código es estándar y portable se emplea Checkstyle. Checkstyle es una herramienta de desarrollo para ayudar a los programadores escribir código Java que se adapte a una norma de codificación.

Automatiza el proceso de verificación y control de código Java ya que para los seres humanos de esto, a pesar de ser una tarea importante, resulta muy aburrido.

Esto hace que sea ideal para los proyectos que quieren hacer cumplir una norma de codificación.

Checkstyle es altamente configurable y se puede hacer para apoyar casi cualquier norma de codificación. Por defecto este proyecto final de carrera emplea los convenios de Sun para el desarrollo de código Java.

1.5 CAPÍTULOS

La memoria consta de x capítulos en los que se abordan los siguientes temas en cada uno de ellos.

1. En el primer capítulo nos encontramos la introducción.

2. En el segundo capítulo analizamos el mercado de este tipo de soluciones y comparamos con la llevada a cabo en este proyecto final de carrera.
3. En el tercer capítulo, nos dedicamos a abordar los distintos tipos de soluciones por las que podía haber abordado el problema, plantearemos sus ventajas e inconvenientes, y nos decantaremos por una que es la que llevaremos a su culminación.
4. En el cuarto capítulo nos embarcamos en las distintas herramientas o bibliotecas para el desarrollo y la gestión del proyecto. Indagaremos sobre cuales vamos a emplear y el porqué.
5. En el quinto capítulo empezamos con el diseño de la solución.
6. El sexto se tratará algo de calidad de software.
7. Conclusiones.

2 PROSPECCIÓN EN EL MERCADO DE PRODUCTOS QUE ABORDEN EL PROBLEMA TRATADO

El análisis y la métrica de rendimiento en los sistemas gestores de bases de datos relacionales es una práctica muy extendida. Actualmente todos los sistemas gestores de bases de datos, de una forma u otra, disponen de herramientas que ayudan tanto a su configuración como a la monitorización y supervisión de los mismos. Entonces, ¿por qué diseña otro medidor de rendimiento?

La explicación es bien sencilla, y la existencia de sistemas de métrica y supervisión de sistemas gestores de bases de datos de terceros nos dan la razón. Normalmente, las herramientas que se incluyen en con el propio sistema gestor de bases de datos están orientadas a analizar parámetros propios de rendimiento, de estado del servicio dentro del sistema operativo, de la apertura o no de los canales de comunicación por los que el sistema gestor de bases de datos se comunica con los clientes ya sean usuarios o software de terceros. Estas métricas son concienzudas y han sido realizadas por los propios creadores del gestor de bases de datos, y nadie mejor que ellos para analizar en modo teórico todo lo relacionado con dicho gestor.

Pero, ¿qué ocurre cuando por ejemplo, aquí en la Escuela, en un examen de la asignatura de bases de datos trescientos alumnos se enfrentan a un examen en el que todos de modo concurrente tienen que lanzar consultas contra un determinado gestor de bases de datos y todo va mal? Sí se consultan las herramientas de supervisión y se encuentra algún problema, es fácil de solucionar. Pero ¿y si esto no ocurre? ¿De dónde viene el fallo? ¿Es fallo del sistema gestor de bases de datos y las herramientas de supervisión no lo reflejan o por lo menos de un modo claro? ¿Puede ser causado por la red? ¿Puede ser del sistema operativo del servidor que está actualizando algún componente, verbigracia el parseador de XML y el planificador de sistema operativo lo considera de más prioridad que el servicio del sistema gestor de bases de datos? ¿Puede

ser un alumno rezagado que no ha estudiado las sentencias JOIN y ha creado sin saberlo un producto cartesiano inmenso que desborda la caché y los tiempos de CPU del gestor de bases de datos?

Con este proyecto final de carrera no se pretende hacer una herramienta que prevenga cualquier tipo de fallos y los solucione. Lo que se pretende desarrollar es un sistema muy simple para medir rendimientos, que se pueda ejecutar de una forma eficiente y casi imperceptible para el usuario y que nos dé una idea de que algo anormal está ocurriendo.

En el marcado ya existe herramientas con estas capacidades, otras que desarrollan funciones similares.

2.1 TRANSACTION PROCESSING PERFORMANCE COUNCIL (TPC)

TCP es una organización sin fines de lucro fundada en 1988 para definir los proceso de transacciones y herramientas de métrica de base de datos además de difundir los resultado de la información de forma objetiva y verificable para la industria.

Los sistemas de métrica de TPC son ampliamente utilizados hoy en la evaluación del rendimiento de los sistemas, los resultados se publican en el sitio web de TPC.

Estos sistemas de métrica son tipo benchmark y están más orientado a la comparación de distintos sistemas gestores de bases de datos.

No los podemos incluir en el mismo saco que este proyecto, ya que los objetivos pretendidos son distintos, pero fue la primera toma de contacto con este tipo de soluciones. Además de disponer de múltiple documentación de cómo hacer pruebas de rendimiento para poder encontrar posibles fallos de rendimiento o de degradación en el

comportamiento normal del sistema. También emplean metodologías de desarrollo de sus benchmark de forma altamente eficiente.

2.2 NETIQ APPMANAGER FOR ORACLE DATABASE RDBMS SERVER

Dispone de un sistema de administración y supervisión de rendimiento. Proporciona herramientas de gestión de bases de datos Oracle optimizando su rendimiento y disponibilidad. Disponible en tiempo real de sistemas de diagnóstico y se almacenan en un repositorio de históricos.

Proporciona un sistema de notificación proactiva y un sistema automático que aplica medidas correctivas. Además de un sistema de aviso de todas las ocurrencias que se producen.

Aunque orientado para gestores de bases de datos Oracle. A pesar de todo, también se pueden adquirir componentes para hacerlo trabajar con Microsoft SQL Server.

Es un software propietario. Y a diferencia de los anteriores no dispone de ningún conjunto de bibliotecas para el desarrollo de complementos.

Sólo se puede implantar en entornos Microsoft Windows.

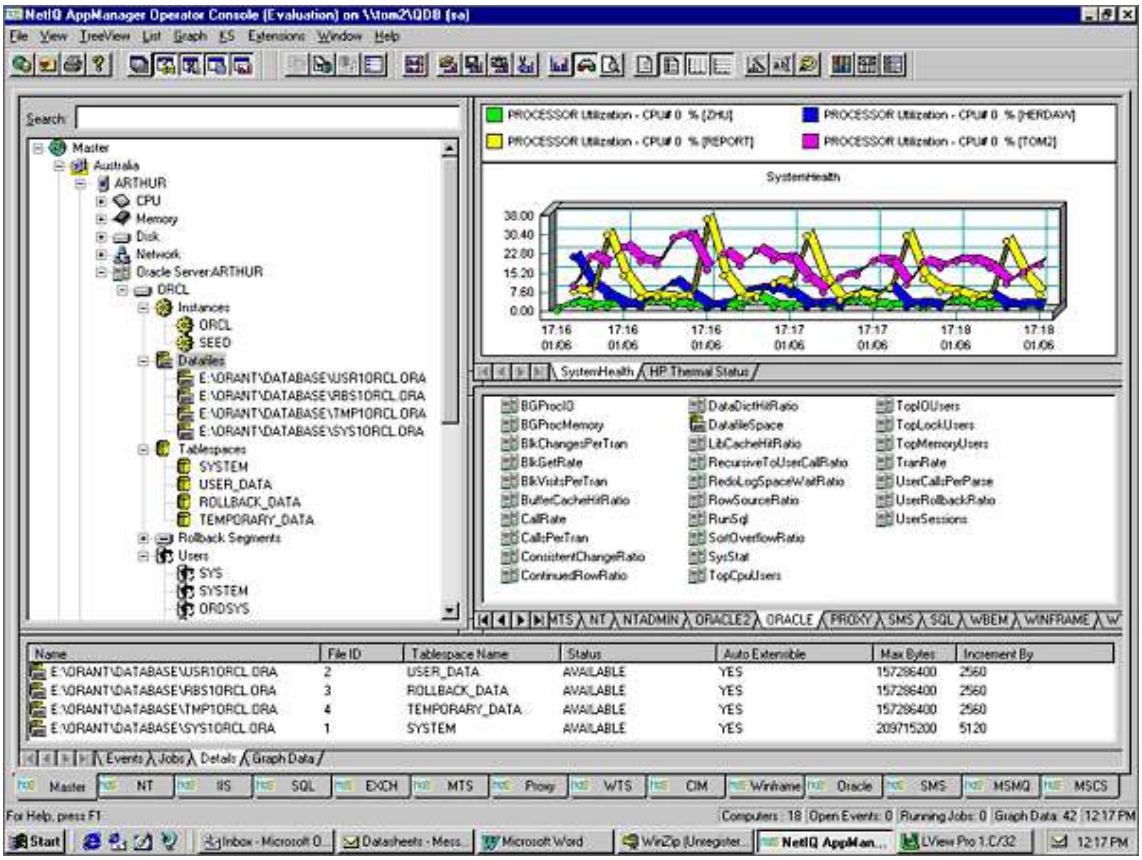

Ilustración 1

2.3 NAGIOS

Nagios es un sistema de código abierto para la supervisión de redes. Nagios está muy extendido. Nagios se emplea para la supervisión de equipos y servicios.

Nagios se distribuye bajo licencia GNU General Public License Version 2.

Es un software que proporciona una gran variedad de parámetros a consultar y supervisar en el sistema y genera alertas según pautas preestablecidas.

Entre sus virtudes figura la capacidad de supervisar protocolos de red tales como SMTP, POP3, HTTP y SNMP entre otros, la supervisión del hardware de los equipos conectados en la red y dispone de un conjunto de bibliotecas para el desarrollo de complementos para el sistema.

El diseño de dichos componentes o plugins se puede hacer mediante Bash, C++, Perl, Ruby, Python, PHP, C#,...

En cuanto a su implantación, como aplicación es bastante pesada, por lo cual normalmente necesita un host dedicado a este fin.

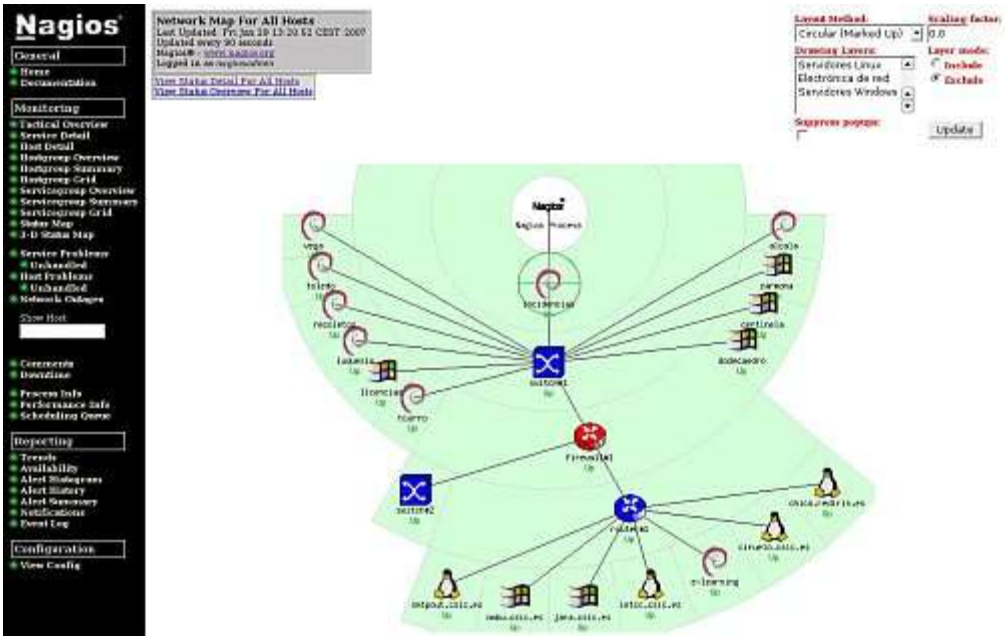

Ilustración 2

2.4 WEBMIN

Webmin un sistema de administración remota mediante HTTP para sistemas UNIX. Webmin permite configurar aspectos del sistema desde un navegador.

Webmin se distribuye bajo Licencia BSD.

Webmin está escrito en Perl, versión 5, y tiene su propio servidor HTTP ligero y con poco consumo de recursos.

Webmin es un sistema modular, y el usuario puede decidir que quiere y que no quiere cargar en la administración. Dispone a sí mismo un conjunto de bibliotecas para el desarrollo de estos módulos.

No está orientado al desarrollo de módulos no orientados a administración de sistemas.

Se implanta en sistemas UNIX, aunque existen proyecto para portarlo a Windows estas no son oficiales.

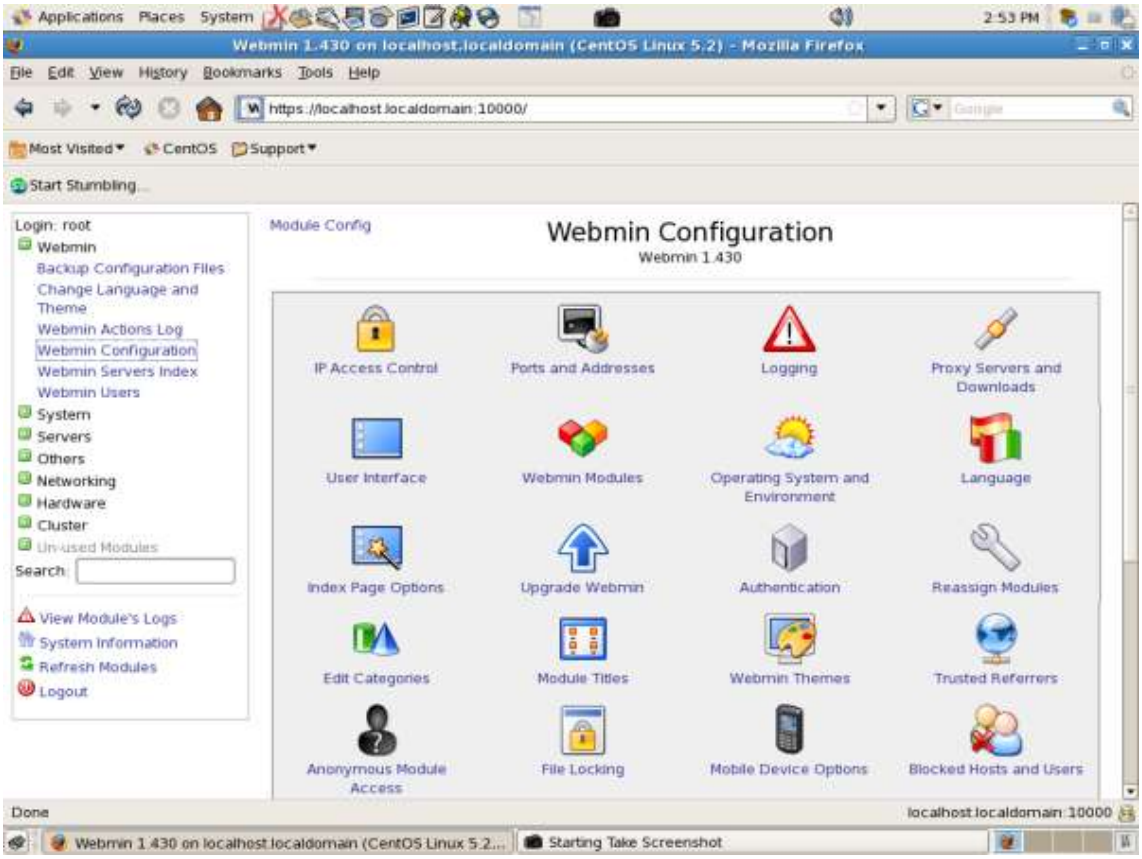

Ilustración 3

2.5 VENTAJAS E INCONVENIENTES DE ESTE PROYECTO FINAL DE CARRERA CON RESPECTO A LOS ANTERIORES

Ahora pasamos a mostrar una tabla comparativa de los sistemas anteriormente descritos y la comparación de sus capacidades con las del proyecto final de carrera al que pertenece estas memorias.

La tabla comparativa no incluye los sistemas de supervisión de los distintos sistemas gestores de bases de datos, ya que estos pueden ser muy diferentes entre sí, y enumerarlas todas es una tarea ardua y como hemos comentado con anterioridad estas herramientas tiene capacidades superiores a las analizadas, ya que su ámbito de actuación es mucho más amplio y complejo.

	Nagios	**Webmin**	**NetIQ AppManager for Oracle Database RDBMS Server**	**PFC**
Conjunto de bibliotecas para desarrollo	SI	SI	NO	NO
Multiplataforma	NO (Host dedicado)	NO (Sistemas UNIX)	NO (Sistemas Windows)	SI (JVM)
Funcionalidad de propósito especifico	NO	NO	SI	SI
Implementado	NO	NO	SI	SI

Licencia	GPL v.2	BSD	Propietaria	Apache 2.0
Distribución	Gratuita	Gratuita	De pago	-
Generación de gráficas	-	-	SI	SI (Analizador)
Supervisión en tiempo real	-	-	SI	SI (Chartserver)
Servicio de sistema	NO (Host dedicado)	SI	SI	SI (Servicio)
Lenguaje nativo	C y otros	Perl	-	Java 1.6.0_14
Múltiples SGBD	-	-	NO (Oracle)	SI (JDBC)

Ilustración 4

3 DISTINTOS TIPOS DE SOLUCIONES PARA EL MISMO PROBLEMA

Para afrontar el desarrollo del proyecto final de carrera, lo primero fue intentar enfocar el problema desde diferentes perspectivas, analizando siempre sus ventajas e inconvenientes. En dichas comparaciones se ha tenido en cuenta las ventajas que podríamos obtener en el desarrollo del proyecto final de carrera, echando mano de distintos marcos de trabajo ya consolidados para crear la solución. Frente a los inconvenientes de crear una infraestructura completa para llevar a la consecución el proyecto.

También hemos de tener en cuenta las desventajas acaecidas del hecho de emplear dichos marcos de trabajo y los distintos problemas que puedan producirse por este hecho. A veces, si nos centramos mucho en intentar acatar una disciplina de trabajo producida por el uso de marco de trabajo determinado podemos perder el control del desarrollo y esto en un proyecto que tiene como baluarte la sencillez y eficiencia en la obtención de resultados puede pesar negativamente.

En los distintos enfoques para el problema se partió de la idea de no ligarse en exceso ni de una plataforma de ejecución, ni de ningún marco de trabajo determinado y cuyos cambios puedan influir negativamente en nuestro trabajo.

3.1 COMPONENTE PARA NAGIOS

Una de los planteamientos más estudiado fue esta. Nagios es un sistema estable y bastante extendido para las labores de métrica y de monitorización tanto de equipos como de servicios de red.

Nagios durante el tiempo en que se realizo la prospección del mismo no tenía ningún componente cuya finalidad fuera la misma que la que se pretende con este proyecto final de carrera, luego si se hubiera optado por este planteamiento, hubiera sido una propuesta nueva para este sistema.

Nagios además de la multitud de componentes que incluye, dispone de un potente conjunto de bibliotecas de desarrollo y que permiten la creación de componentes de cualquier tipo.

Tras el estudio que se llevo a cabo nos encontramos con inconvenientes que hacían no abordable este mecanismo para llevarlo a cabo. Dichos inconvenientes eran principalmente que Nagios necesita unos requerimientos hardware que prácticamente lo condicionan a ser desplegado en un host para uso exclusivo de dicha aplicación, y sobre todo si tenemos por ejemplo como escenario al que puede ir dirigido uno de los laboratorios del centro.

Aun así, pasemos a ver la arquitectura de la solución. El desarrollo de la solución sería un sistema cliente/servidor, siendo el servidor una maquina de uso exclusivo para Nagios. Los clientes un servicio instalado en las máquinas que deseen probar un determinado gestor de bases de datos. Obviamente, el propio servidor sería capaz de hacer pruebas sin necesidad de clientes, los clientes solo darían distintos puntos de vista del mismo escenario.

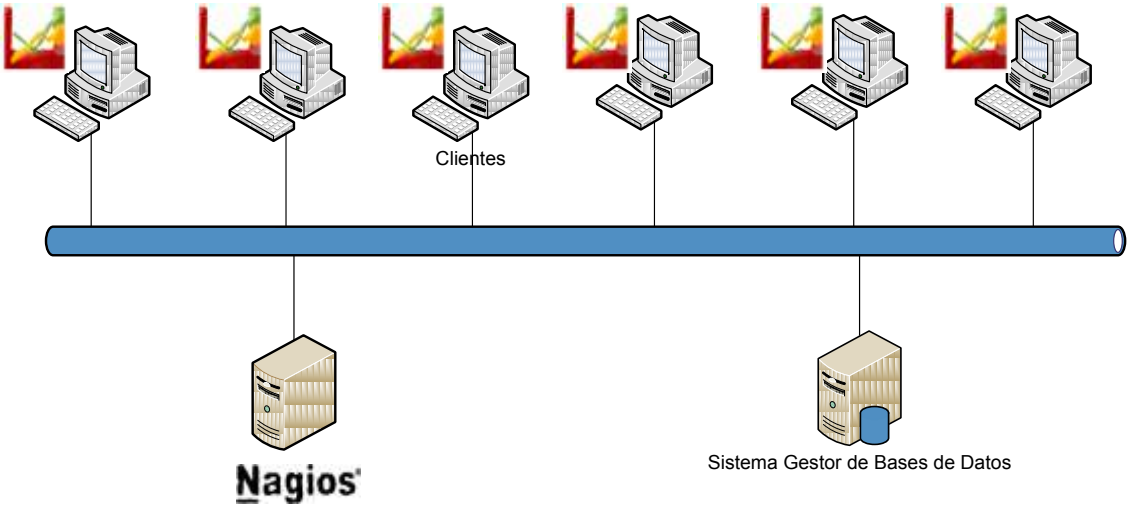

Ilustración 5

La problemática de la inclusión a en nuestra infraestructura de red de un host dedicado a Nagios, suponiendo esto un gasto extra si nuestra red no dispone de ninguno, fue el factor principal de desecharlo.

3.2 COMPONENTE PARA WEBMIN

Siguiendo la prospección de distintos sistemas de monitorización y supervisión que se encuentran muy extendidos dentro de la administración de sistemas, abordamos el intento de desarrollo de otro componente, en este caso para Webmin.

Webmin es una plataforma de administración muy extendida. La opción de diseñar un complemento para dicho sistema que nos sirviera de solución para los propósitos perseguidos era bastante atrayente y más cuando comprobamos que no existía ninguno hecho que hiciera eso mismo.

Esta solución haría que cada equipo que tuviera el componente implantado, efectuaría peticiones sobre el sistema gestor de bases de datos y mostraría los resultados obtenidos. Lo que haría necesario que también se pudiera configurar el componente de uno de los equipos como primario, y fuera el que recibiera toda la información de la granja de equipos que efectúan las pruebas.

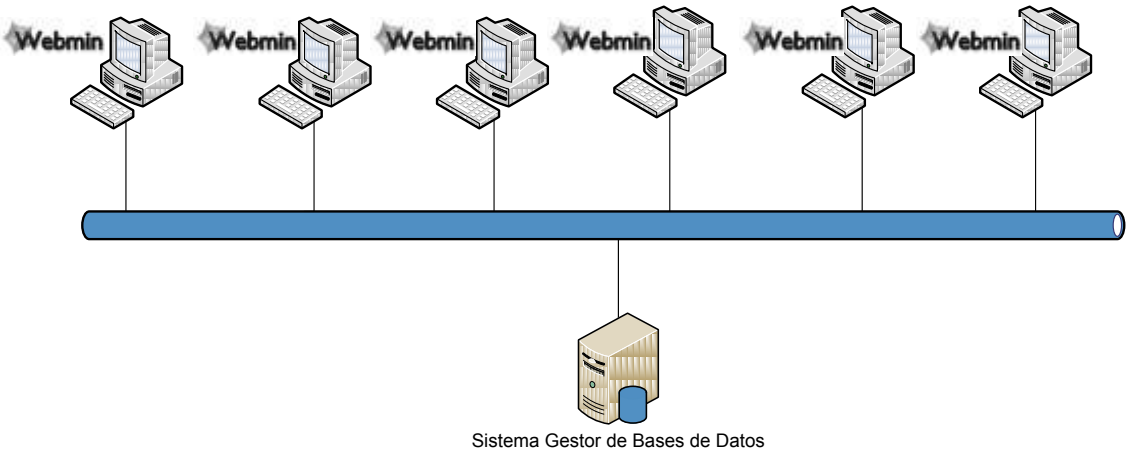

Ilustración 6

Esta opción fue descartada por dos grandes motivos. Para empezar Webmin está orientado a correr sobre sistemas operativos tipo UNIX, lo que no nos garantiza la multiplataforma que era algo que se pretendía desde un principio, en realidad existen portaciones de Webmin para entornos Windows pero no son oficiales, ni tampoco se nos garantiza la fiabilidad y la compatibilidad de los mismos. El otro de los motivos es que las bibliotecas no tienen funciones que nos ayuden en nuestros propósitos, dichas bibliotecas están hechas en Perl lo que tampoco hacen demasiado atractivo el desarrollo que partiría prácticamente de cero y nos forzaría a hacerlas compatibles con todo los ya hecho, añadiendo un requisito extra externo al proyecto final de carrera.

Ambas condiciones mostraban un escenario hostil y del que no obtendríamos ninguna ventaja.

3.3 CLIENTE/SERVIDOR DESPLEGADO EN UN SERVIDOR DE APLICACIONES

La arquitectura constaría de un servidor de aplicaciones, como Glassfish, que desplegaría un Webservice que facilitaría la comunicación con los clientes que hagan las métricas y junto a una aplicación web que se encargaría de publicar la información estadística por web.

Otro posible planteamiento para dicha solución es cambiar el servidor de aplicaciones por un motor de integración[1], que aunque no esté muy relacionado con el problema, yo personalmente tengo mucho bagaje en estas herramientas y todas disponen de herramientas de desarrollo que nos ayudarían tanto a hacer las pruebas como en obtener los datos y la información derivada de los mismo.

[1] Un motor de integración es una herramienta de software diseñado para simplificar la creación y gestión de interfaces entre aplicaciones separadas y sistemas en una organización. Los motores de integración intercambian mensajes entre sistemas y permiten la gestión, mapeo, traducción y modificación de datos entre sistemas de información para asegurar el intercambio efectivo de datos en la organización.

El cliente lanzaría las consultas a tratar contra el sistema gestor de bases de datos y tomaría los tiempos obtenidos. Dichos tiempos se enviarían al Webservice que despliega el servidor. Éste se encargaría de procesar los datos recibidos y los publicaría sobre la aplicación web.

Dicho planteamiento tiene los mismos inconvenientes que el desarrollo de un componente para Nagios, requiere un host dedicado en este caso para el servidor de aplicaciones, por tanto tiene todos los problemas consecuentes de esto.

Por otro lado esta arquitectura, a nivel académico serviría para investigar un campo de conocimientos amplio como son los Webservice, el desarrollo de aplicaciones web, la comunicación entre plataformas y sistemas de diferente índole,…, pero como el objetivo de este proyecto final de carrera es conseguir algo eminentemente práctico y usable en un entorno real como pueda ser un laboratorio de la Escuela, se termino descartando esta solución.

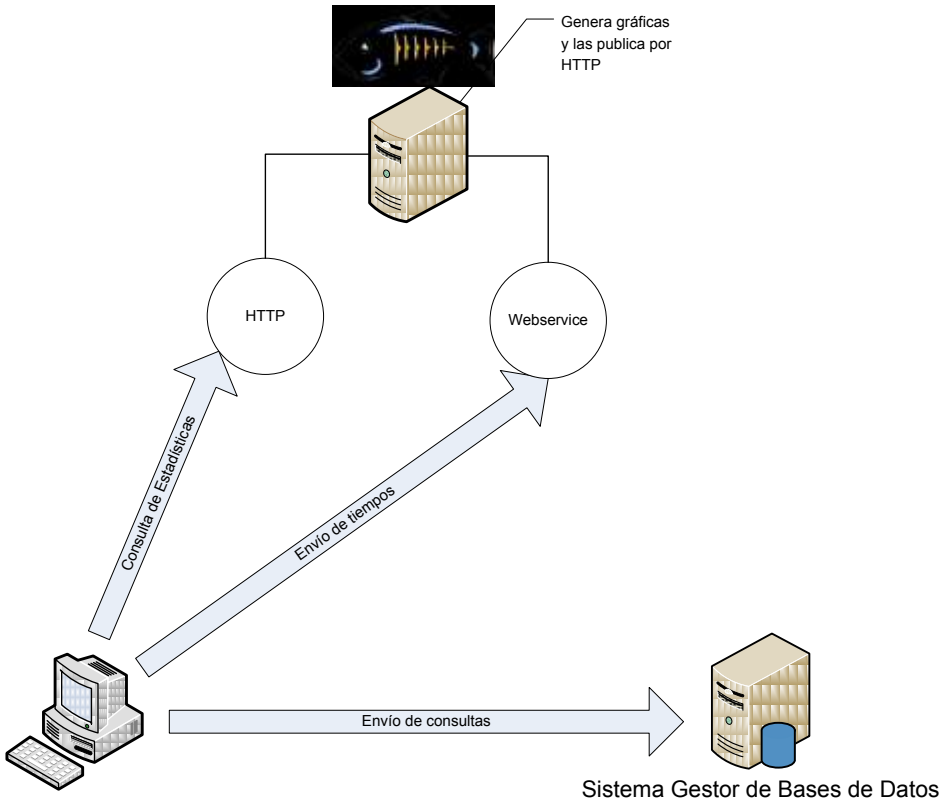

Ilustración 7

3.4 SERVICIO/ANALIZADOR DE DATOS

Esta solución, que puedo adelantar que, ha sido la elegida para la consecución de este proyecto final de carrera.

En esta arquitectura se diferencian dos sistemas bien diferenciados. Uno un servicio, llamado curiosamente Servicio, que es el encargado de lanzar consultas y hacer las pertinentes métricas de tiempo. Y otro al que se le denomina Analizador y es el que no facilita la visión y análisis de los datos que proporciona el servicio mencionado anteriormente.

El servicio es un pequeño demonio de sistema integrable en el sistema operativo, con independencia de que este sea el que sea, y que efectuaría las peticiones al sistema gestor de bases de datos y guardaría los resultados es un fichero. En la práctica, el fichero en cuestión es una base de datos SQLite, aunque se puede configurar como un acceso a otro sistema gestor de bases de datos, preferentemente no al que se le van a efectuar las pruebas aunque esto es factible.

El otro miembro de la dualidad sería el Analizador de los datos que se hayan guardado en el fichero de almacenamiento anteriormente indicado. Es un entrono gráfico con los pertinentes asistentes de interpretación.

Está arquitectura tiene como ventaja la simplicidad de desarrollo, no siendo esto por falta de trabajo o de calidad sino por su simpleza de conceptos y su buena diferenciación de campos de trabajo de cada elemento, ya que los dos componentes que la forman son independientes y su nivel de requisitos de ejecución es prácticamente despreciables.

Por otra parte, tiene la virtud de que un solo Analizador es capaz de interactuar con múltiples Servicios.

Por tanto, por ejemplo para medir la salud de un determinado sistema gestor de bases de datos desde un laboratorio de la Facultad, se puede lanzar un Servicio desde cada

equipo de los alumnos del laboratorio, mientras el profesor puede consultar los resultados desde otro equipo usando una sola instancia del Analizador.

Dado el grado de sencillez del concepto de la solución y de su versatilidad, el desarrollo proyecto final de carrera se decanto por esta solución.

Para garantizar el concepto de multiplataforma, el desarrollo se hace sobre Java SE.

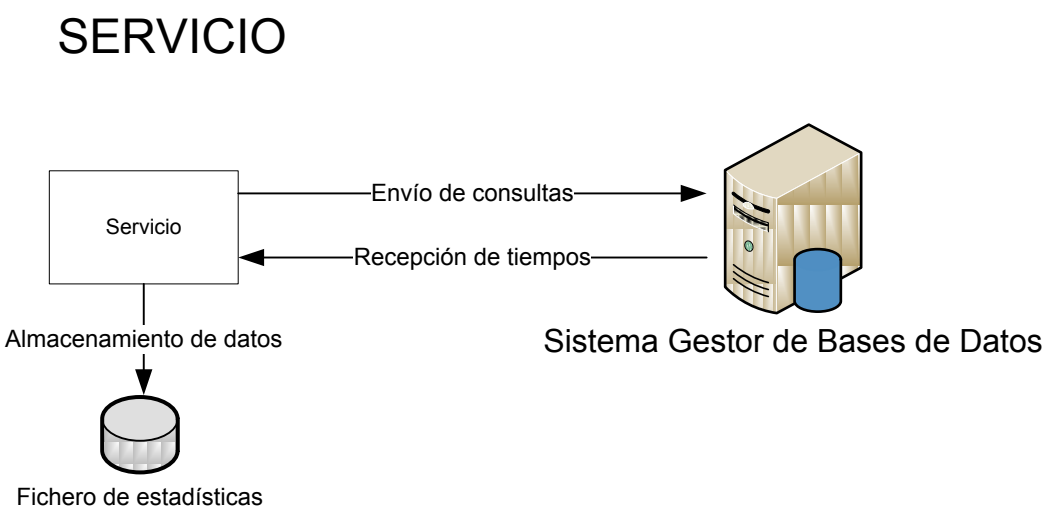

Ilustración 8

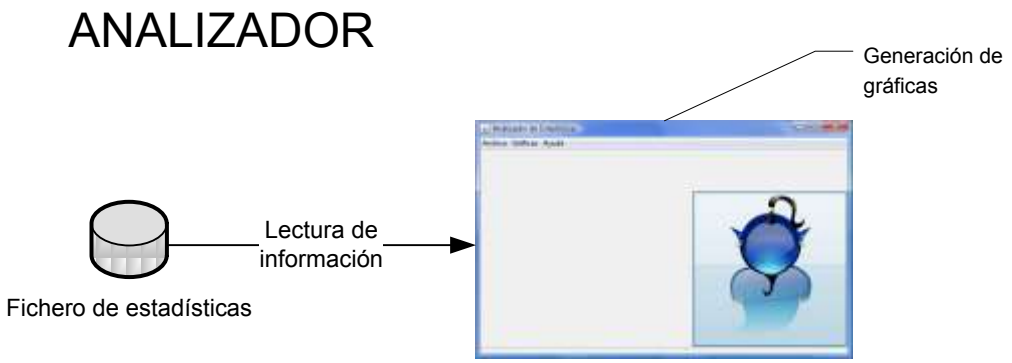

Ilustración 9

4 ARQUITECTURA Y HERRAMIENTAS EMPLEADAS PARA EL DESARROLLO Y LA GESTIÓN DEL PROYECTO.

En este capítulo comenzaremos la enumeración de los distintos elementos que ha participado en el desarrollo de este proyecto final de carrera.

4.1 LENGUAJE DE PROGRAMACIÓN: JAVA

Bueno para empezar hemos de tratar el asunto del lenguaje de programación. El lenguaje de programación elegido para el desarrollo ha sido Java. Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.

El lenguaje se escribe con una sintaxis similar a C y C++, al que se le ha simplificado el modelo de objetos y elimina las sentencias de bajo nivel, ya que estas son las causantes de muchos errores, como la manipulación directa de punteros o memoria y que no suelen poder detectarse en tiempo de compilación. Java además dispone de sentencias de control de errores mediante el mecanismo de excepciones, minimizando las incidencias de los mismos.

Java se ejecuta sobre una máquina virtual (en inglés Java Virtual Machine, JVM) que es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en bytecode, una especie de ensamblador, el cual es generado por el compilador del lenguaje Java. La generación de los bytecode que son los mismos para cualquier implantación de la máquina virtual nos garantiza el funcionamiento de nuestro código, siempre que la JVM cumpla correcta y

completamente las especificaciones, tanto en un equipo de escritorio domestico como en un grid-computer. También es interesante el recolector de basura que incorpora que se encarga de limpiar de la memoria las estructuras o variables que no se vayan a volver a usar durante la ejecución actual, evitando así el deterioro de la aplicación.

4.2 JDBC

Para conectar a Java con cualquier sistema gestor de bases de datos empleamos JDBC. JDBC es un marco de programación para los desarrolladores de Java necesitan tener un modelo de acceso a la información guardada en bases de datos, hojas de cálculo y archivos de texto planos. JDBC se utiliza comúnmente para conectar una aplicación de usuario con un sistema gestor de base de datos, sin importar qué gestor software de base de datos se utilice para almacenar la información. Empleando un dialecto del lenguaje de manipulación de datos SQL como medio de consulta o modificación de los mismos.

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar.

Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión, para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar con cualquier tipo de tareas con la base de datos a las que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

4.3 LOG4J

Como herramienta de auditoría del funcionamiento de la aplicación, usamos Log4J. Log4j es una biblioteca de código abierto desarrollada en Java por la Apache Software Foundation y tiene como función el permitir a los desarrolladores elegir la salida y el nivel de prioridad de los mensajes a tiempo de ejecución y no a tiempo de compilación como es comúnmente realizado.

Es muy útil ya que nos permite durante la ejecución configurar la salida de errores desde a ficheros de diario de ejecución, como el envío a una determinada cuenta de correo electrónico, el almacenamiento en una base de datos y otros tantos destinos más.

Todo configurable sin tocar nada de código. Se eligió este sistema de auditoría frente a otros por ser un estándar de facto.

4.4 SQLITE

Las estadísticas que se obtienen y procesan durante todo el flujo de información, por defecto se almacenan en SQLite. SQLite es un sistema gestor de bases de datos relacional que cumple las especificaciones ACID[2]. A diferencia de los sistemas gestores de base de datos cliente-servidor, SQLite es un fichero de datos relacional que al que se accede mediante una biblioteca que ha de usarse desde el programa cliente de dicha información guardada en la base de datos, y no en un proceso externo a la aplicación. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base

[2] Un sistema de gestión de bases de datos cumple ACID quiere decir que el mismo cuenta con las funcionalidades necesarias para que sus transacciones tengan las características de atomicidad, consistencia, aislamiento, durabilidad.

de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

En este proyecto final de carrera hemos empleado SQLiteJDBC que implementa la versión 3 de SQLite y tiene una capacidad de hasta 2 terabytes de tamaño la inclusión de campos tipo BLOB. Obviamente no vamos a producir 2 terabytes de datos, es más si se va a producir un elevado valor de datos, por ejemplo de más de un centenar de megabytes es recomendable cambiar la configuración del Servicio para cambiar la línea que configura el driver de conexión para guardar las estadísticas, por otro sistema gestor de bases de datos más consistente.

4.5 CSV (COMMA SEPARATED VALUE)

El formato de intercambio de información elegido para que los datos recolectados puedan ser exportados hacia otra aplicación ha sido CSV.

El formato de fichero CSV (Comma Separated Value) de Valores Separados por Comas, se usa habitualmente para el intercambio de datos entre aplicaciones diferentes Los ficheros CSV son un tipo de ficheros estructurado en un formato sencillo para representar datos en forma de tabla, en las que los atributos de cada uno de los atributos de los registros se separan por comas y los propios registros se separan por saltos de línea. Este formato también es utilizado en Microsoft Excel, y se ha convertido en un formato estándar utilizado por muchas aplicaciones incluso en plataformas GNU/Linux.

El formato CSV tuvo una gran importancia y popularidad como formato de intercambio de datos antes de la aparición del estándar XML, de tal modo que llegó a convertirse en un formato estándar de facto. Aunque en la actualidad está siendo desplazado por el estándar XML. Existen multitud de variantes del formato CSV, e incluso aplicaciones capaces de transformar ficheros de datos CSV en ficheros XML.

4.6 VISIONADO DE GRÁFICAS: JFREECHART VS GOOGLE CHART

Para la tarea de generación de los gráficos, se barajaron dos opciones. Google Chart y JFreeChart. Al final, se optaron por ambas soluciones. Google Chart se emplea en ChartServer que se ejecuta conjuntamente con la aplicación Servicio, si se encuentra implantada, y JFreeChart se emplea en la interfaz gráfica de la aplicación Analizador.

JFreeChart nos garantiza independencia del exterior. Google Chart nos garantiza mucha velocidad, pero tiene el inconveniente que sus invocaciones efectúan peticiones a URL de Google que es el que nos devuelve un gráfico en formato de imagen PNG. Google Chart, sorprendentemente y a pesar de que para su ejecución se necesita hacer una llamada a una URL remota, resulta más rápido que JFreeChart que es una biblioteca de nuestro proyecto.

Google Chart permite más tipos de gráfica, más fáciles de generar y más simples de invocar. Pero como ya he dicho, el invocar la URL remota, nos implica el enviar información a una entidad externa y la necesidad de conexión a internet para poder mostrar la gráfica, hecho que quizás no siempre nos sea posible o nos interese.

En resumidas cuentas, ambas opciones se han llegado a implementar, pero, por defecto la aplicación Analizador emplea JFreeChart, ChartServer en cambio es un complemento para la aplicación Servicio.

JFreeChart es una librería para gráficos desarrollada completamente en Java. JFreeChart facilita mostrar gráficos en nuestras aplicaciones, ya sean web o de escritorio. Entre las principales virtudes de esta biblioteca tenemos un API consistente y bien documentado con soporte para un amplio rango de tipos de gráficas, también da soporte para varios tipos de salida de archivos de imagen como PNG y JPEG, y formatos gráficos vectoriales tales como PDF, EPS y SVG. JFreeChart es una biblioteca que está distribuida bajo la licencia LGPL, luego gratuita y que permite el uso en aplicaciones propietarias.

Google Chart es uno de los últimos *juguetes* de Google. Su nombre completo es Google Chart Imagen Generator, una herramienta por la cual se nos permite crear, vía petición http:// a la API, unas gráficas que podrán ser lineales, de barras, de tarta, de Venn y de dispersión, y que podemos insertar en nuestra web. En el caso de este proyecto, ChartServer genera a partir de los datos de estadísticas las distintas URL que necesita para mostrar los datos, efectúa las peticiones, maqueta los resultados y los publica en modo de página HTML, para que el usuario de ChartServer se conecte a este por medio de un navegador web y visualice los resultados. Volviendo a Google Chart, para obtener las gráficas tan sólo deberemos de crear una URL pasando una serie de parámetros en la misma y se nos devolverá en forma de gráfica en formato PNG. Tan solo tiene como restricción que si vas a emplear más de 250000 peticiones diarias, se lo comunique para que no te tomen como una maniobra de ataque. Para más información se puede consultar http://code.google.com/intl/es/apis/chart/ .

4.7 ENTORNO GRÁFICO: SWING + JDESKTOP

El entorno gráfico Swing de la aplicación Analizado está construido con JDesktop. JDesktop es un framework al estilo de los framework web tipo Struts o Spring. Las interfaces gráficas son clases Java que se apoyan en un fichero XML donde se guardan las características visuales que se mostraran y otro fichero de propiedades que guardará metadatos de los elementos visuales. Como los framework citados anteriormente se basan en el paradigma modelo-vista-controlador.

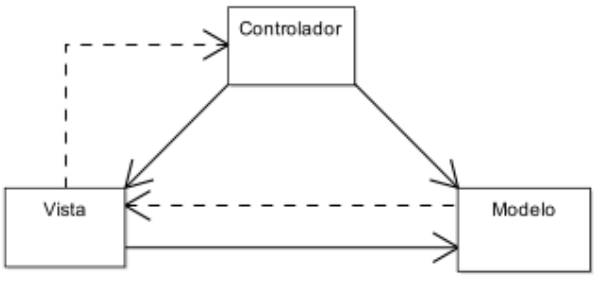

Ilustración 10

Modelo: Es la representación específica de la información con la cual el sistema opera. Esta función la hace la clase Java cargando las características del fichero de propiedades a petición.

Vista: Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Se genera a partir del XML, que es procesado por la clase Java.

Controlador: Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. La clase Java se encarga de esta función, siendo esta la que responde a los eventos. Las bibliotecas del framework simplifican estas operaciones.

Por la función vital que ocupa la clase Java, tiene más parecido de entre todos los framework web con los que he trabajado, con Spring.

4.8 IDE: NETBEANS

Y una pequeña nota ahora al orquestador de todo el desarrollo, NetBeans. NetBeans simplemente se puede definir como la mejor IDE, entorno de desarrollo integrado, de la actualidad. NetBeans es multiplataforma y corre bajo Java. NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos. NetBeans comenzó como un proyecto estudiantil en Republica Checa (originalmente llamado Xelfi), en 1996 bajo la tutoría de la Facultad de Matemáticas y Física en la Universidad de Charles en Praga. La meta era escribir un entorno de desarrollo integrado para Java parecida a la de Delphi. Xelfi fue el primer entorno de desarrollo integrado escrito en Java. Con soporte para Java SE, aplicaciones Web bajo Java EE, Java ME para móviles, Java FX, desarrollo SOA con OpenESB, Ruby, C/C++, PHP, Groovy, Javascript, Python, y la integración con GlassFish, OpenESB y Apache Tomcat, NetBeans se puede considerar el entorno de desarrollo integrado más completo existente para Java, además de otros cuantos lenguajes más.

5 DISEÑO DE LA SOLUCIÓN

A lo largo de este capítulo pasaremos a describir de forma más detallada cada una de los componentes que forman este proyecto final de carrera. Realizando un amplio análisis de su diseño y de sus funciones.

5.1 ENUMERACIÓN DE COMPONENTES

Para empezar, hemos de enunciar los distintos componentes que forman parte del proyecto, así como una breve introducción a ellos.

El proyecto consta de cinco componentes. Estos son tres aplicaciones, una biblioteca de funciones de red y un complemento para una de las aplicaciones. Sus nombres son Servicio, Analizador, Cliente de Estadisticas, ChartServer y TCP.

5.1.1 SERVICIO

Vamos a empezar a hablar del componente Servicio. Servicio es una pequeña aplicación que se lanza como servicio de sistema y que efectúa un conjunto de sentencias SQL o PL/SQL sobre un sistema gestor de bases de datos.

Servicio, mediante un sistema de planificación de tareas, de formato similar al comando cron de Linux, carga tareas según una planificación dada y son lanzados cuando les corresponde. Para cada tarea lanzada, anota el instante de la petición y de la respuesta, calcula el tiempo que se ha tardado en efectuar dicha petición y los almacena en una tabla, llamada Tabla de Estadísticas. Normalmente dicha tabla se almacena en un

fichero SQLite, aunque puede ser configurado para que se guarde en cualquier sistema gestor de bases de datos o cualquier sistema de almacenamiento accesible con JDBC.

Servicio además almacena diarios de sus ejecuciones gracias a Log4J, siendo este configurable para el envío de avisos por correo electrónico si fuera necesario.

Decir también que en los primeros prototipos empleaba iBATIS es un framework basado en capas desarrollado por Apache Software Foundation, que se ocupa de la capa de Persistencia y se sitúa entre la lógica de Negocio y la capa de la Base de Datos, pero siguiendo desarrollos propios, la versión actual emplea clases propias de este proyecto final de carrera para estos menesteres ya que con la clases integradas en el propio proyecto hemos conseguido mejores rendimientos que con el framework anteriormente mencionado.

5.1.2 CHARTSERVER

Como complemento a Servicio, se le puede incluir en su despliegue el que llamamos ChartServer.

ChartServer es un servidor HTTP que publica unas gráficas de estadísticas básicas, en tiempo de ejecución. Cuando un usuario, se conecta al puerto HTTP de ChartServer, el hilo de ejecución correspondiente a esa sesión recolecta datos del fichero de estadísticas actual, genera una URL que es enviada a la API de Google Chart, esta nos devuelve un gráfico PNG por cada una de las URL solicitadas, el hilo de ejecución genera un fichero HTML con la inclusión de los correspondientes gráficos recibido y los envía en el GET al navegador del usuario, y todo esto en un tiempo ínfimo.

5.1.3 JOSEJAMILENA.PFC.SERVIDOR.TCP

Pasamos ahora a comentar la biblioteca josejamilena.pfc.servidor.tcp también llamada TCP. Esta biblioteca contiene funciones tales como:

- Suma de verificación generando el polinomio de comprobación de redundancia cíclica de 32 bits (CRC32) y se suele utilizar para validad la integridad de los datos transmitidos.
- Comunicación por TPC/IP, de ahí su nombre.
- Utilidades de copia rápida de ficheros mediante channles. Los channels representan conexiones a entidades que son capaces de realizar operaciones I/O, tales como ficheros, sockets,... ofreciendo unos rendimientos muy elevados en la comunicación.

5.1.4 CLIENTE ESTADÍSTICAS

Esta aplicación emplea la librería TCP, para conectarse al componente Servicio, y obtiene una copia local de los datos publicados. Funciona desde línea de comandos.

5.1.5 ANALIZADOR

El Analizador es la interfaz gráfica de análisis de los datos almacenados en el fichero de estadísticas que es generado por Servicio.

Analizador abre[3] un fichero de estadísticas, o las recibe por red. Una vez obtenido el origen de los datos, Analizador se conecta a ellos mediante JDBC. En este paso, hay que indicar que el formato nativo de los datos que puede abrir el componente Analizador es SQLite versión 3 generado por SQLiteJDBC. Los ficheros que puede

3 Cuando nos referimos a abrir, es si las estadísticas vienen almacenadas en un fichero SQLite. Si los datos se han guardado en un sistema de bases de datos relacional como puede ser Oracle, para obtener los datos a analizar, hemos de hacerlo haciendo la llamada a Servicio para que el nos envíe los datos y Analizador los almacene en el formato nativo de este, es decir, SQLite.

abrir son ficheros relacionales en este formato. Si los datos que genera Servicio se han guardado en un sistema de bases de datos relacional como puede ser Oracle, para obtener los datos a analizar, hemos de hacerlo haciendo la llamada a Servicio para que el nos envíe los datos y Analizador los almacene en el formato nativo de este, es decir, SQLite. Una vez en este formato se puede trabajar con ello, exportarlos como .CSV,…

Una vez que se ha conectado a los datos al Analizador, ya se puede tratar la información.

Analizador puede generar gráficas por Sistema de Gestor de Bases de Datos, por Script y por Host Cliente. En dichas gráficas se muestran la evolución de la latencia de las peticiones realizadas por Servicio al sistema gestor de bases de datos a analizar a lo largo del tiempo.

Analizador tiene también la capacidad de exportar los datos a formato .CSV, lo que facilita el poder analizarlos en Microsoft Excel o en SPSS.

5.2 DISEÑO DE SERVICIO

Vamos ahora a mostrar los distintos diagramas del diseño del componente Servicio.

5.2.1 DIAGRAMA DE CLASES

Vamos a ver un diagrama principal, y luego lo desglosaremos.

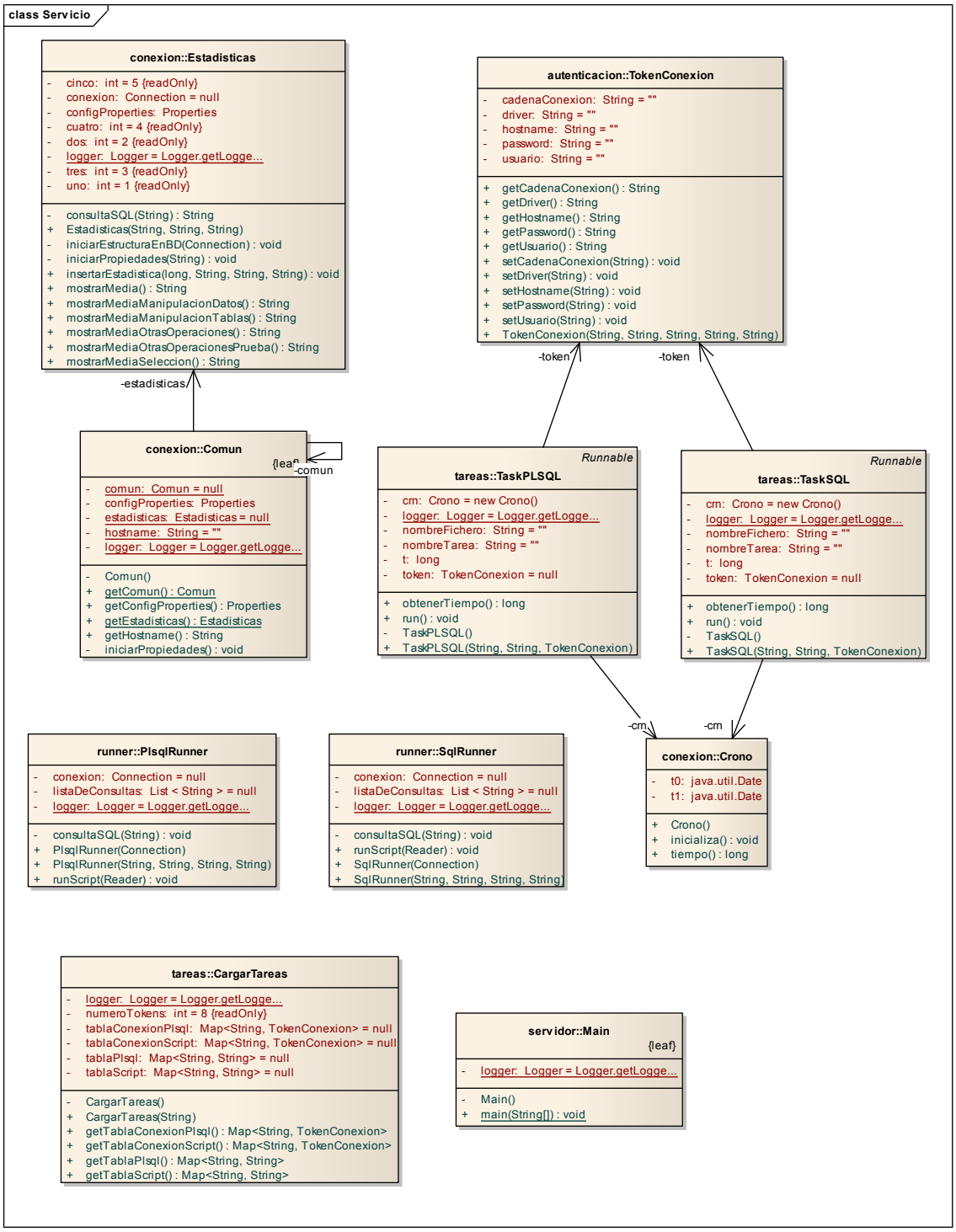

Ilustración 11

5.2.1.1 Paquete josejamilena::pfc:servidor

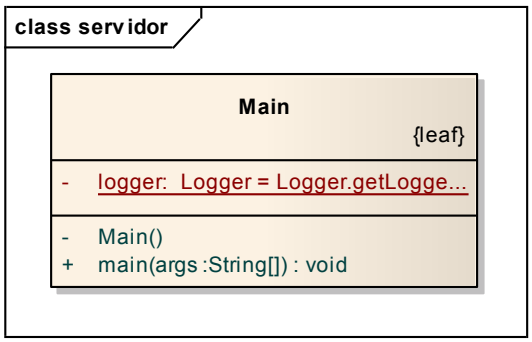

Ilustración 12

5.2.1.2 Paquete josejamilena::pfc:servidor::conexión

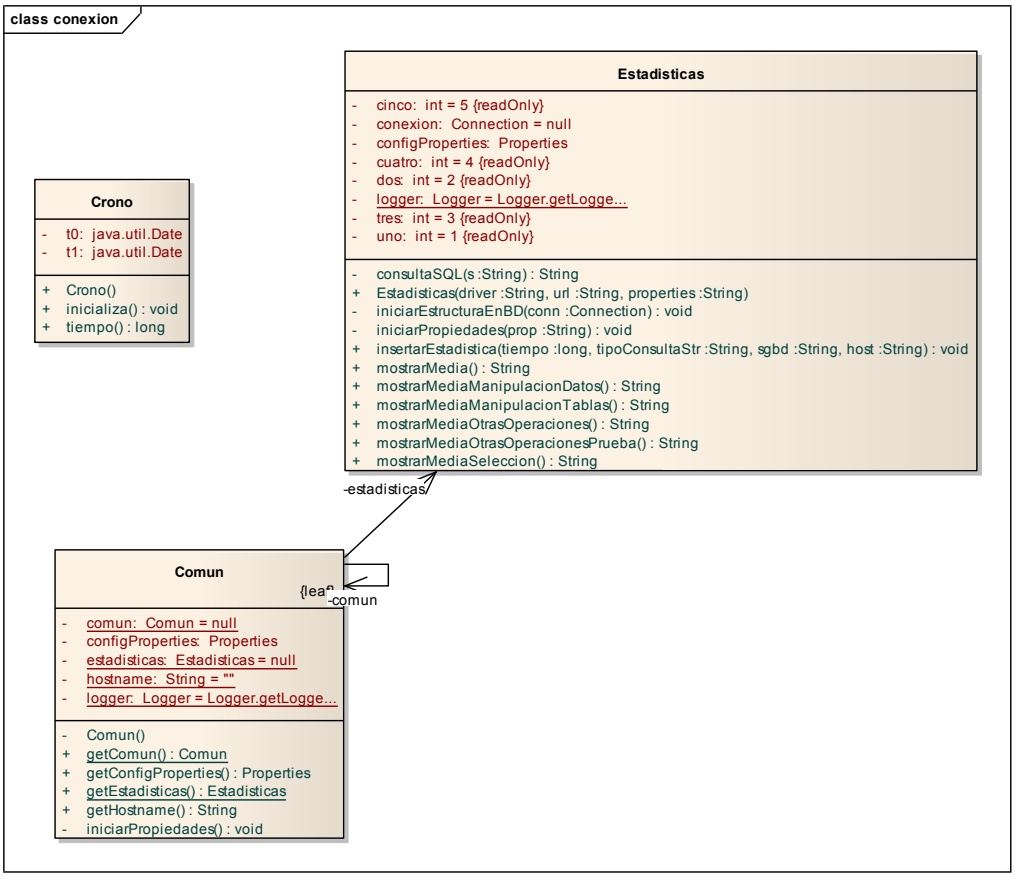

Ilustración 13

5.2.1.3 Paquete josejamilena::pfc:servidor::tareas

Ilustración 14

5.2.1.4 Paquete josejamilena::pfc:servidor::tareas::autenticacion

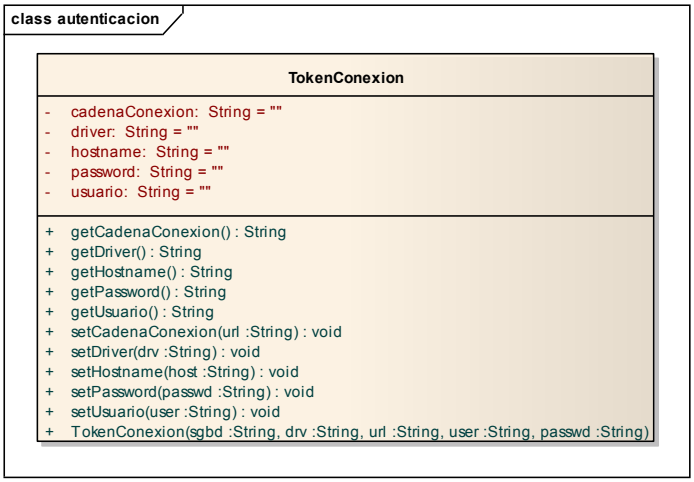

Ilustración 15

5.2.1.5 Paquete josejamilena::pfc:servidor::tareas::runner

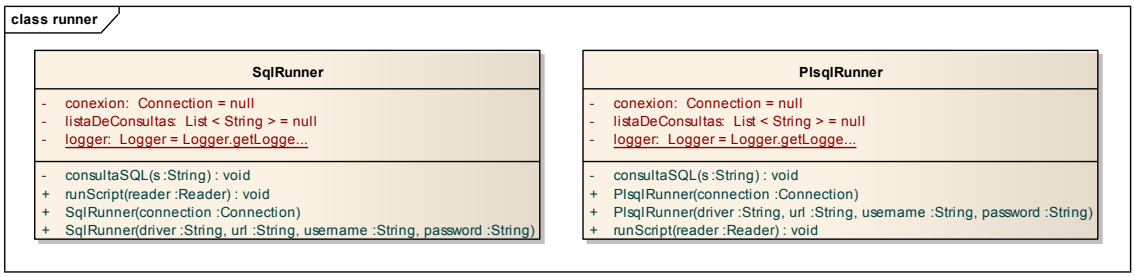

Ilustración 16

5.2.2 DIAGRAMAS DE CASOS DE USO

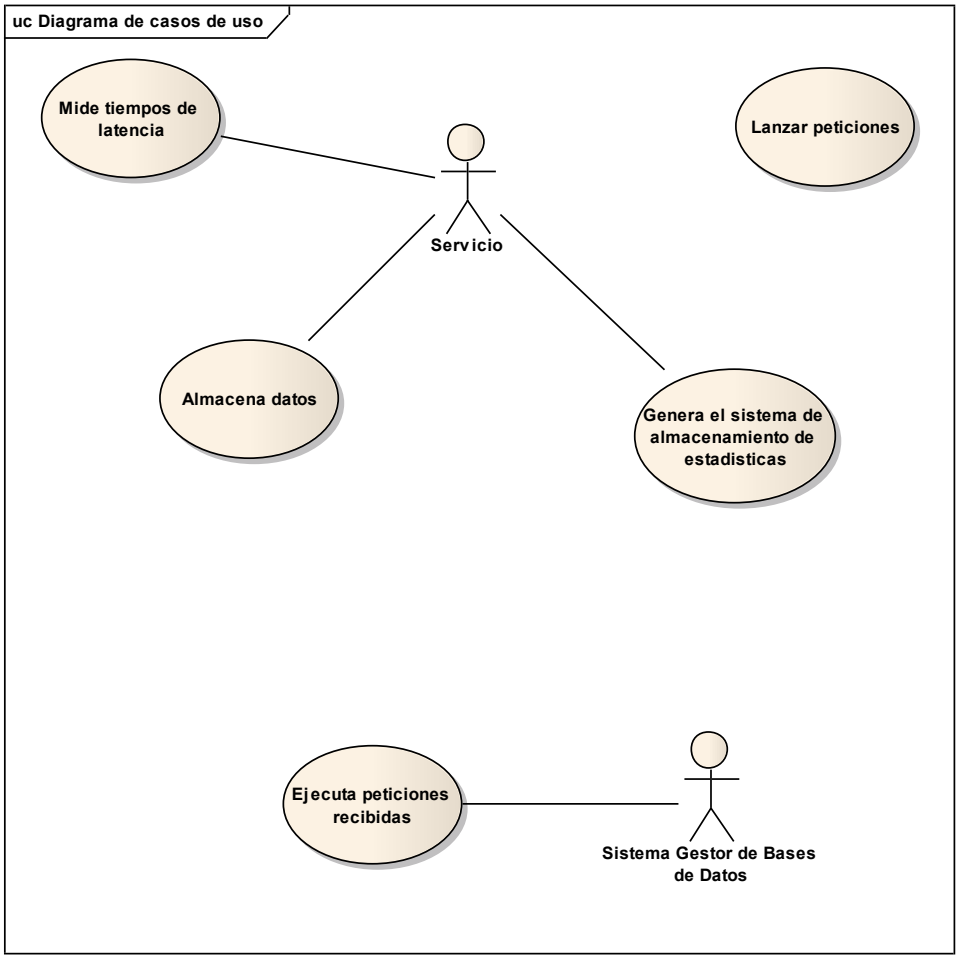

Ilustración 17

5.2.3 *DIAGRAMAS DE COMPONENTES*

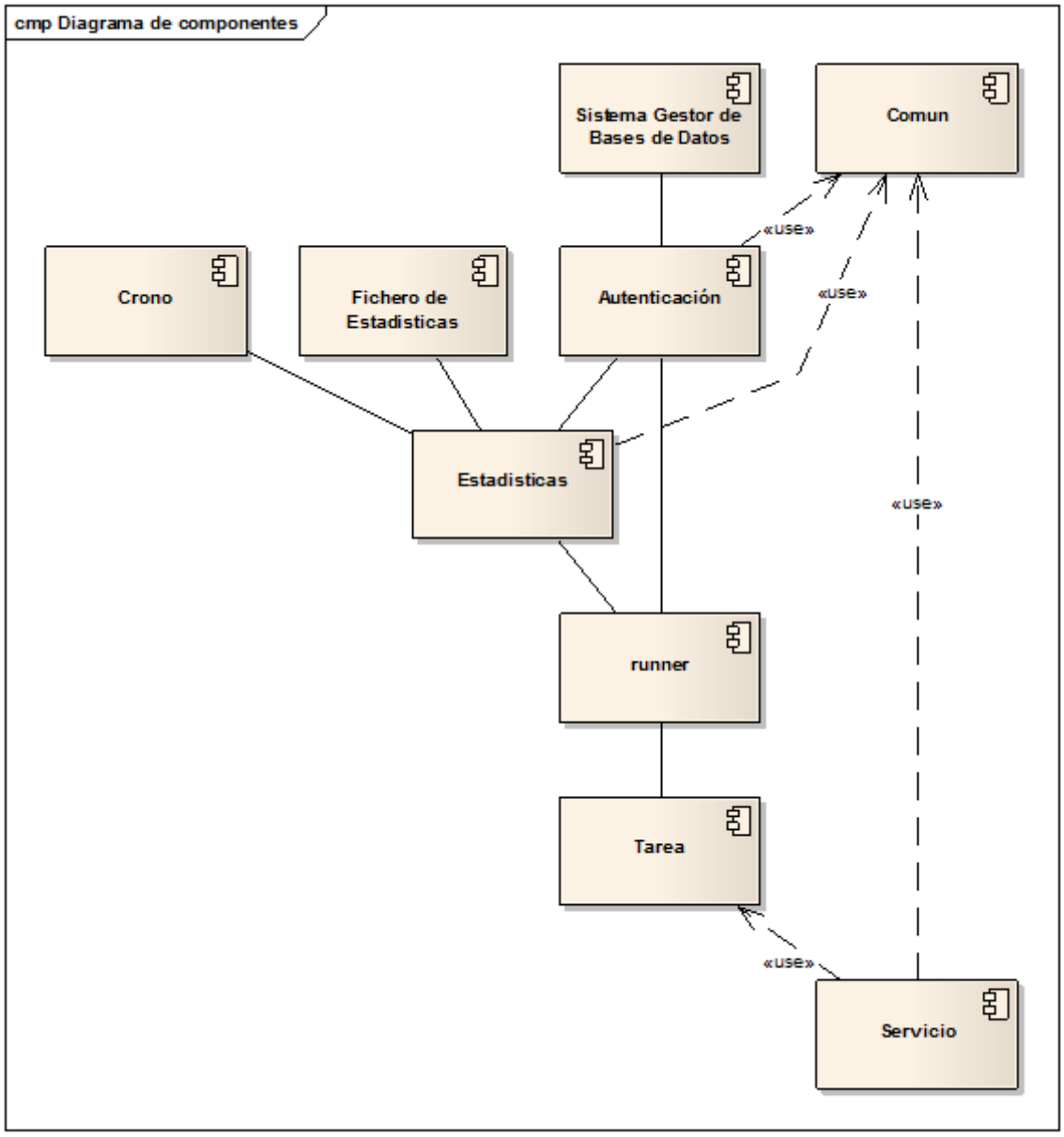

Ilustración 18

5.2.4 *DIAGRAMAS DE COMUNICACIONES*

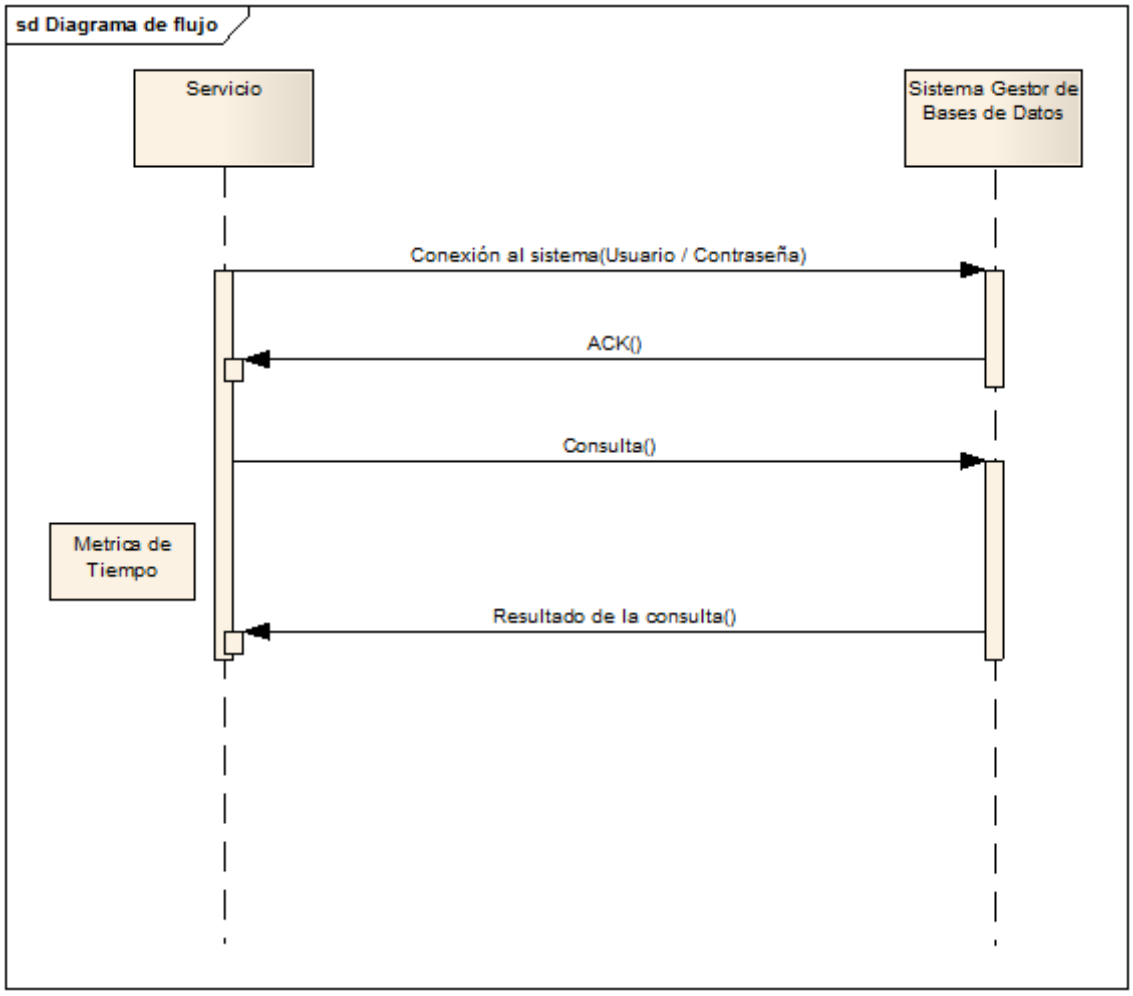

Ilustración 19

5.2.5 *DIAGRAMAS DE ACTIVIDADES*

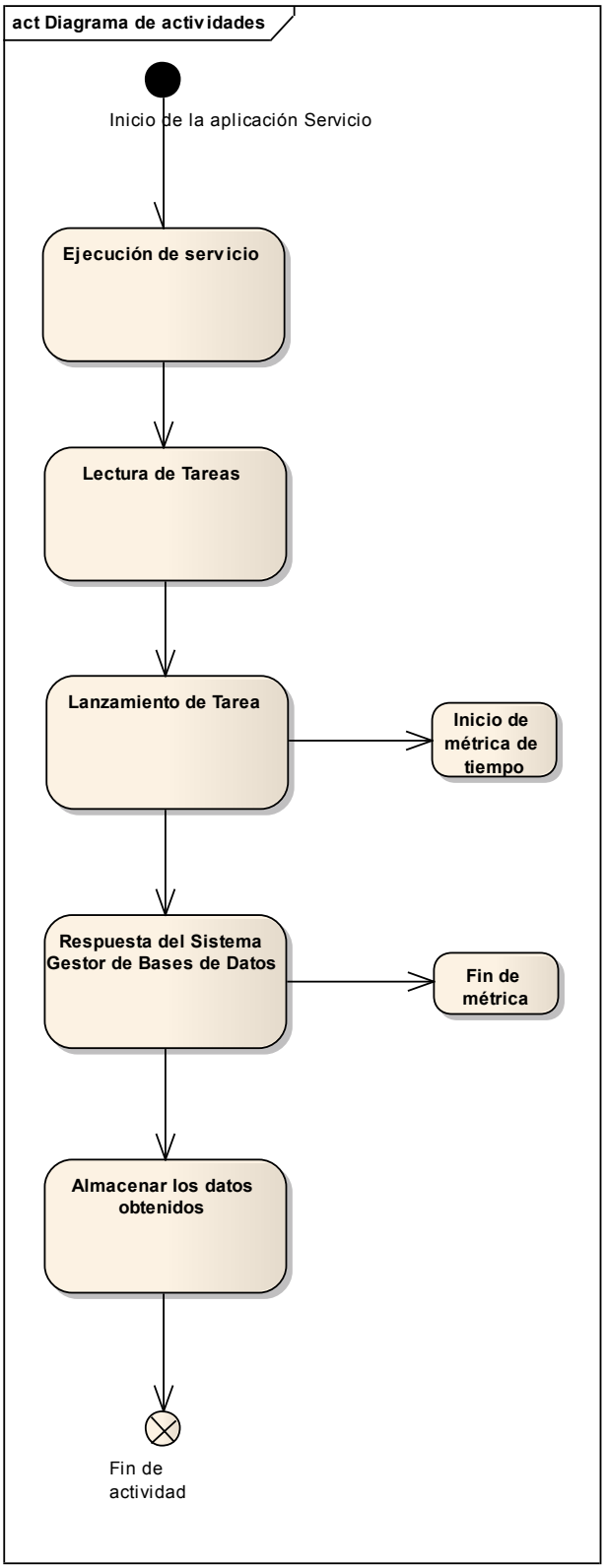

Ilustración 20

5.3 DISEÑO DE CHARTSERVER

5.3.1 *DIAGRAMAS DE CLASES*

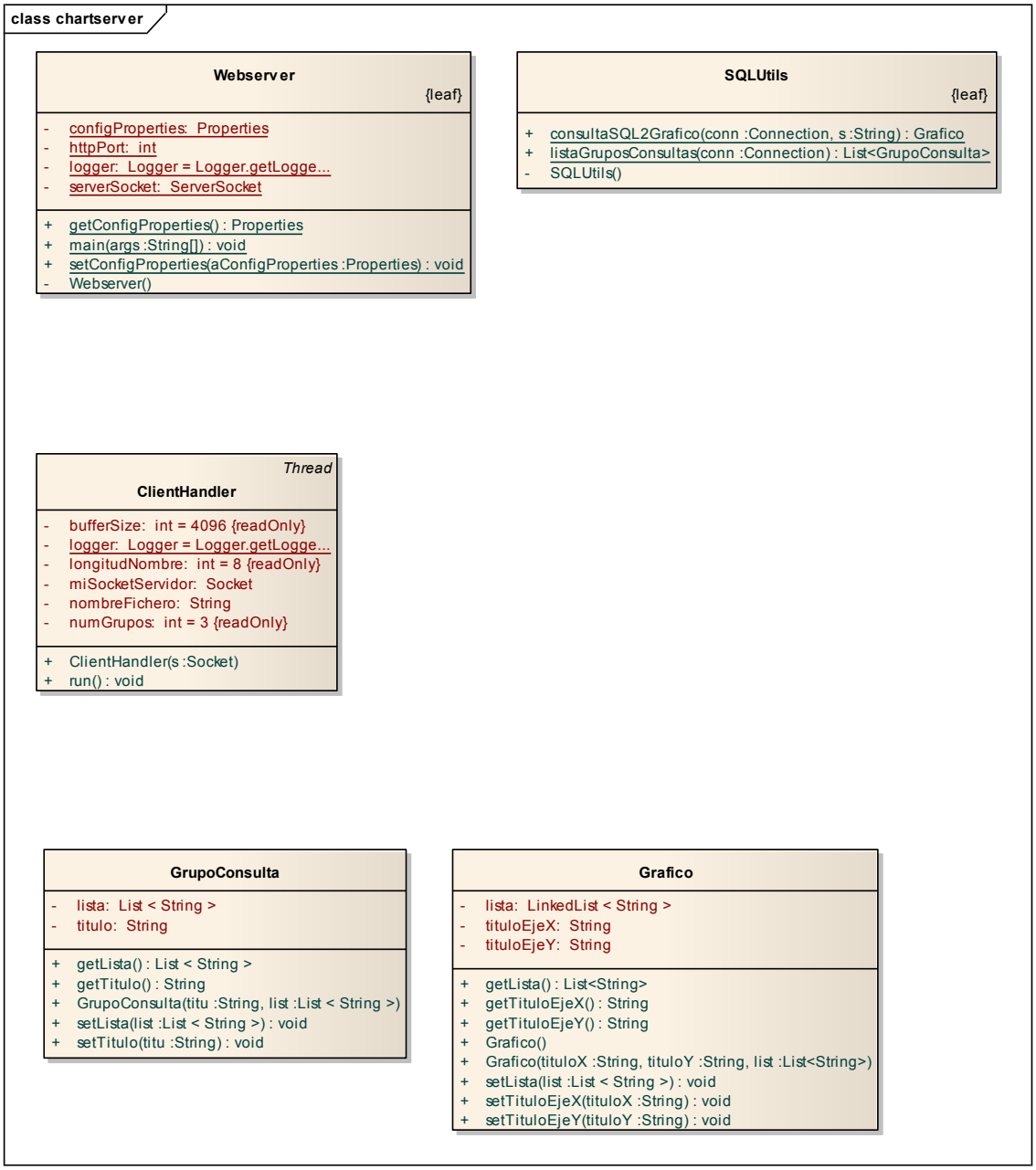

Ilustración 21

5.3.2 *DIAGRAMAS DE COMPONENTES*

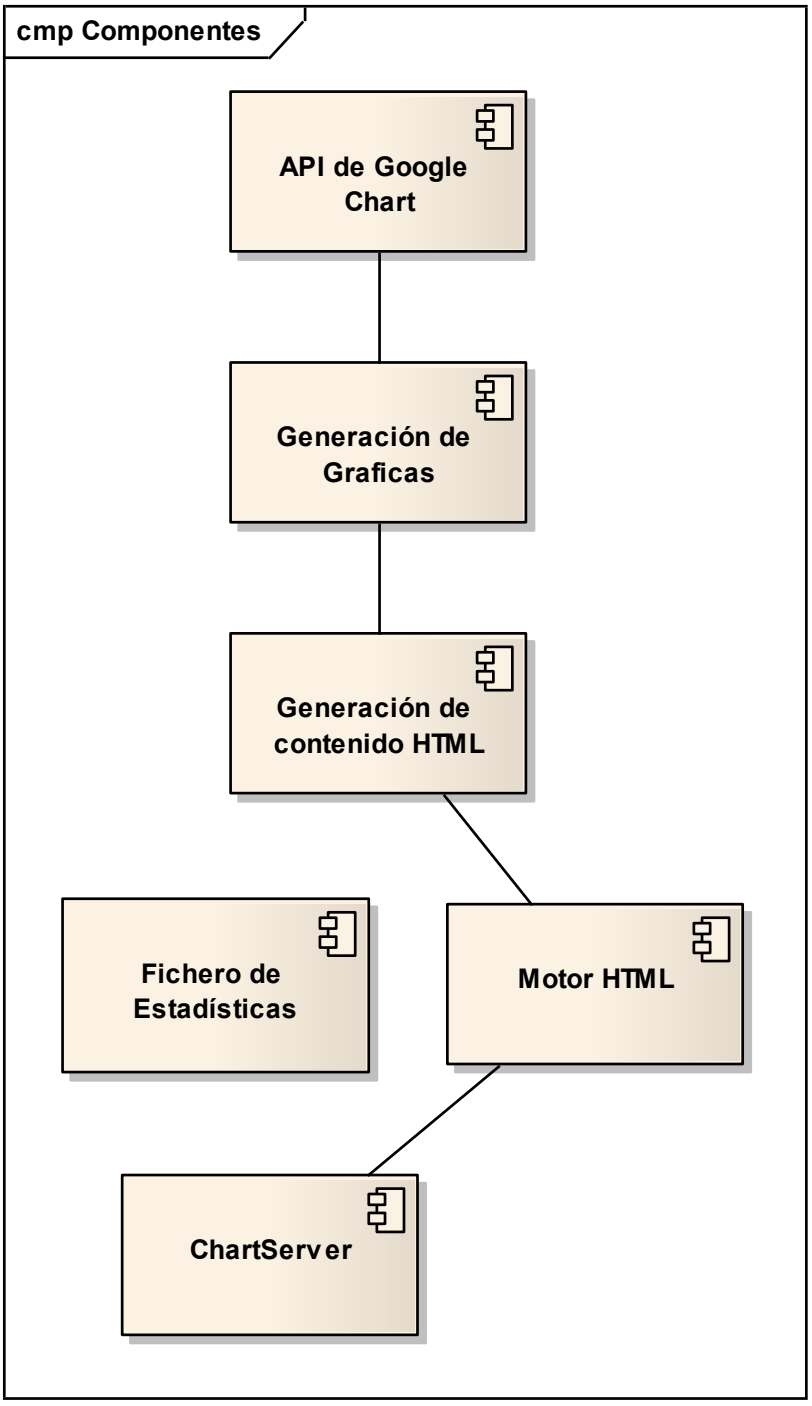

Ilustración 22

5.3.3 *DIAGRAMAS DE COMUNICACIONES*

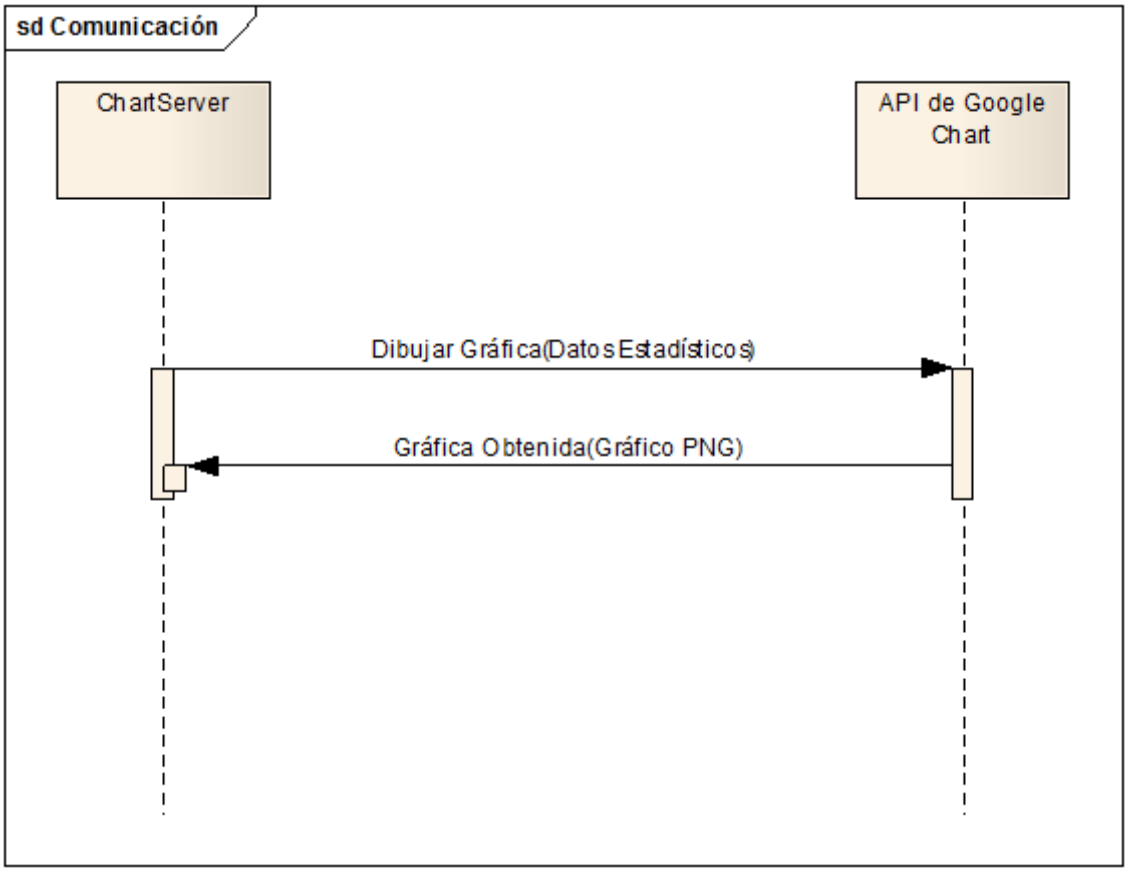

Ilustración 23

5.3.4 *DIAGRAMAS DE ACTIVIDAD*

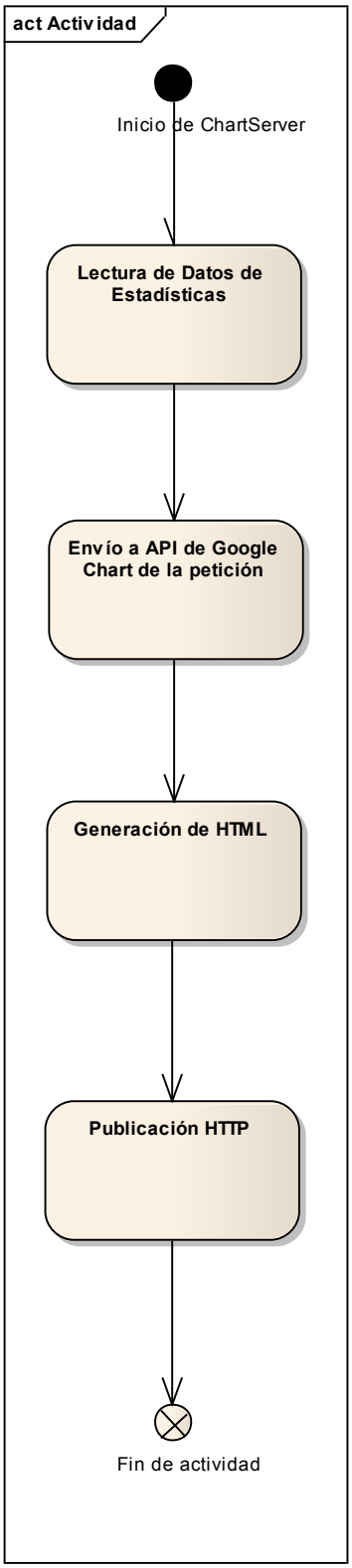

Ilustración 24

5.4 DISEÑO DE LA BIBLIOTECA DE FUNCIONES TCP

5.4.1 DIAGRAMAS DE CLASES

5.4.1.1 josejamilena::pfc::servidor::crypto::easy::checksum

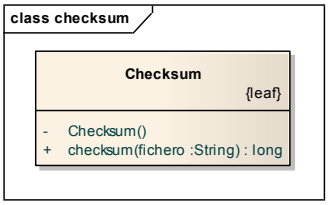

Ilustración 25

5.4.1.2 josejamilena::pfc::servidor::tcp

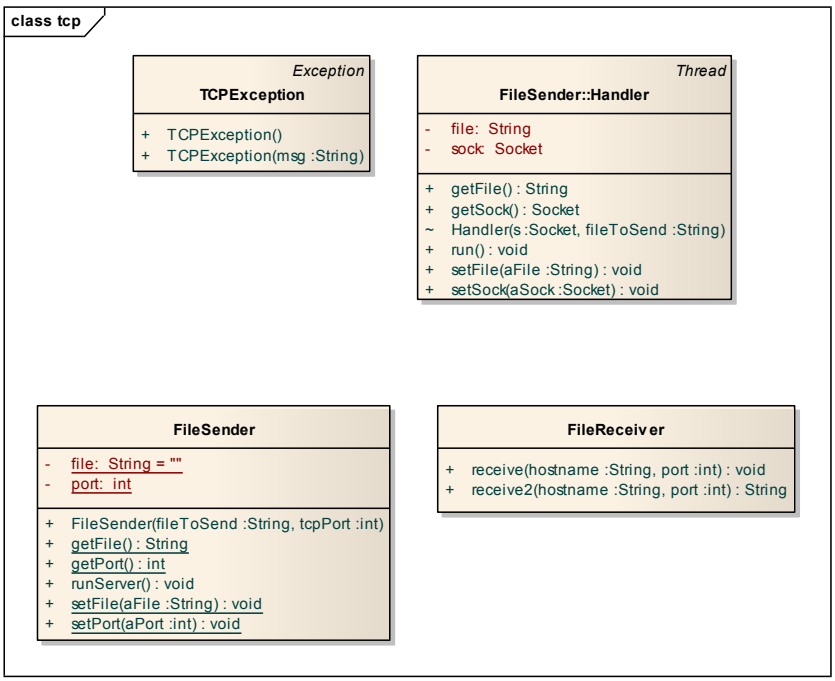

Ilustración 26

5.4.1.3 josejamilena::pfc::servidor::util

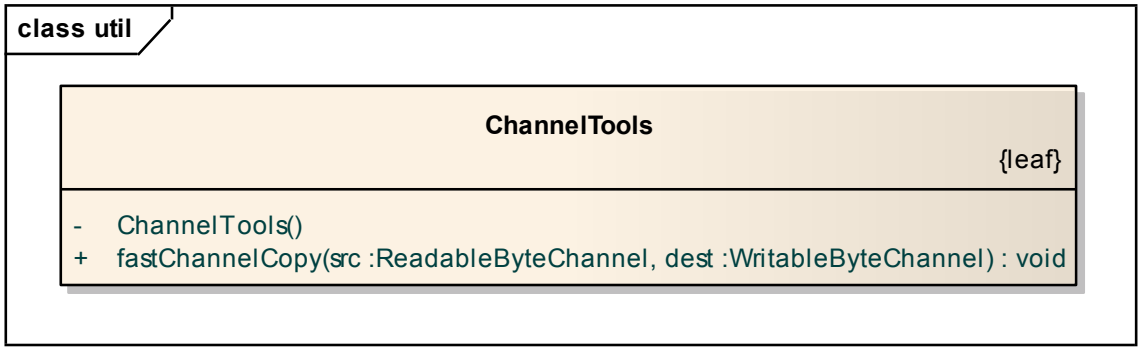

Ilustración 27

5.5 DISEÑO DEL CLIENTE DE ESTADÍSTICAS

5.5.1 DIAGRAMAS DE CLASES

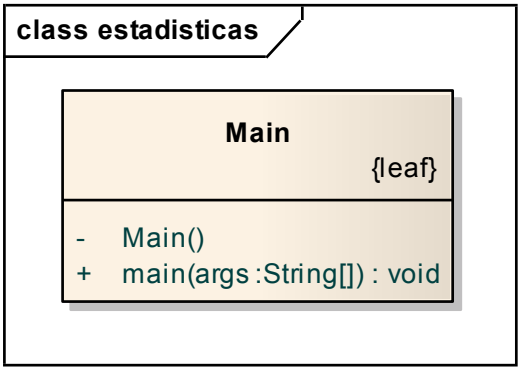

Ilustración 28

5.5.2 *DIAGRAMAS DE COMUNICACIONES*

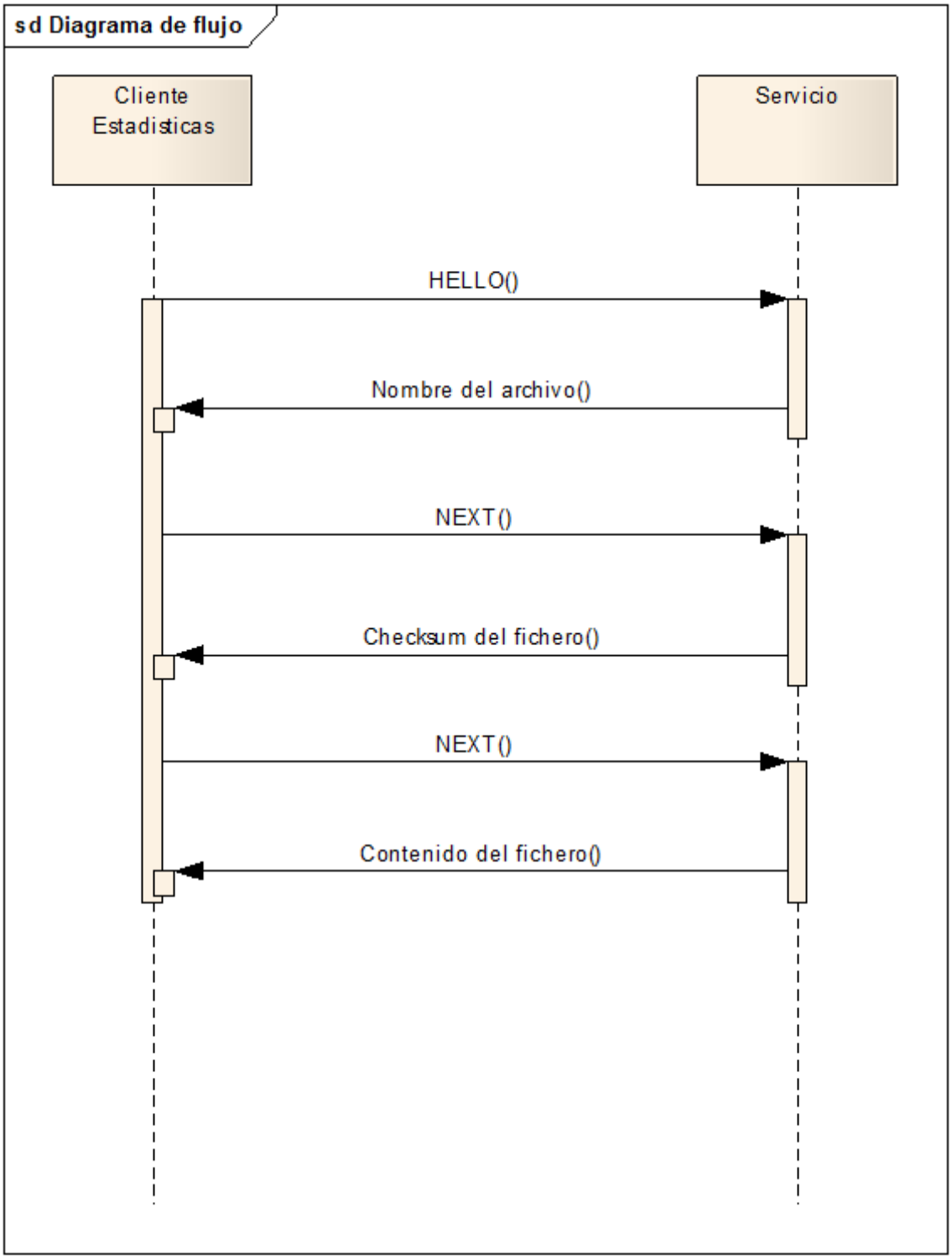

Ilustración 29

5.6 DISEÑO DE ANALIZADOR

5.6.1 DIAGRAMAS DE CLASES

5.6.1.1 josejamilena::pfc::analizador

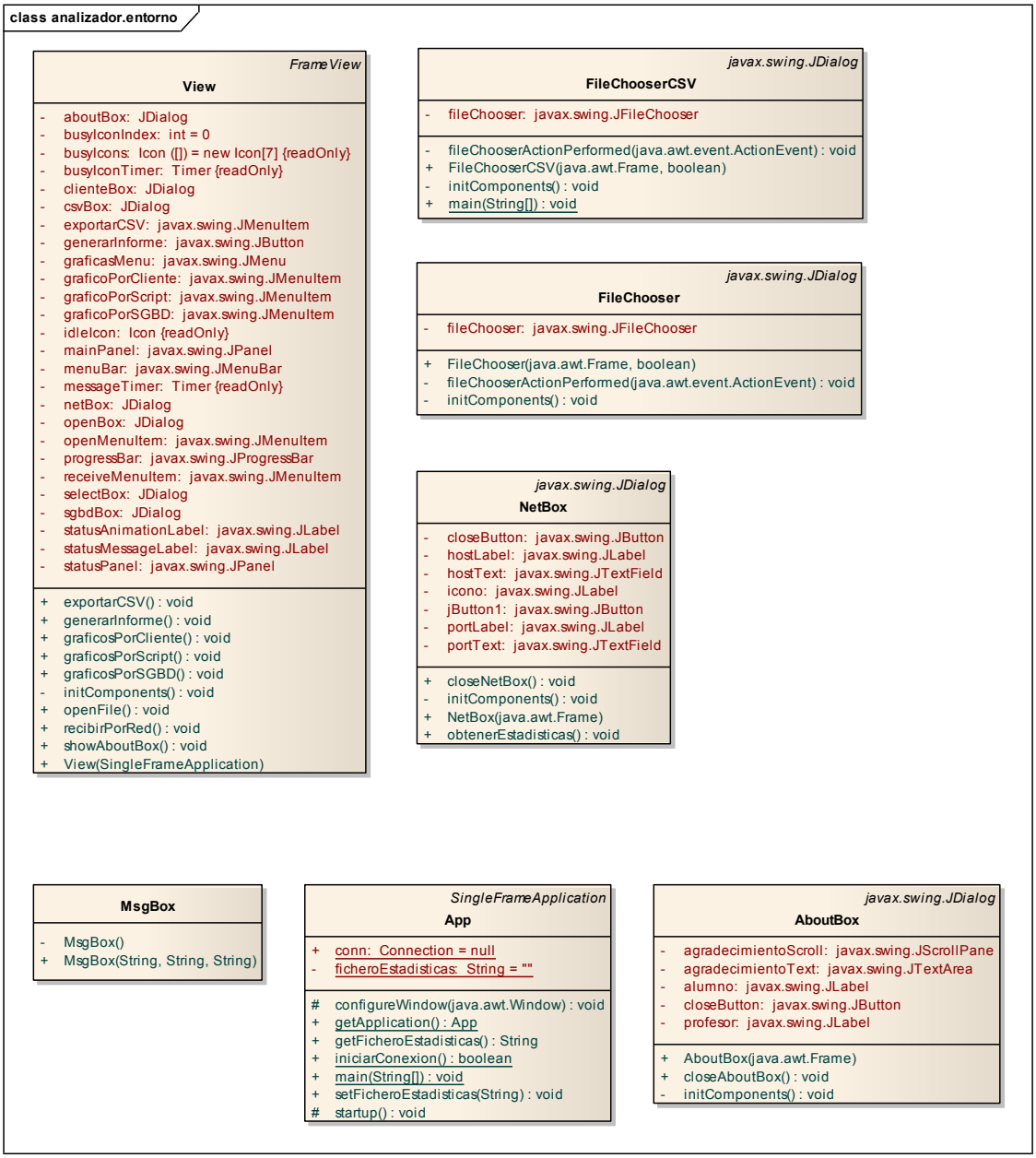

Ilustración 30

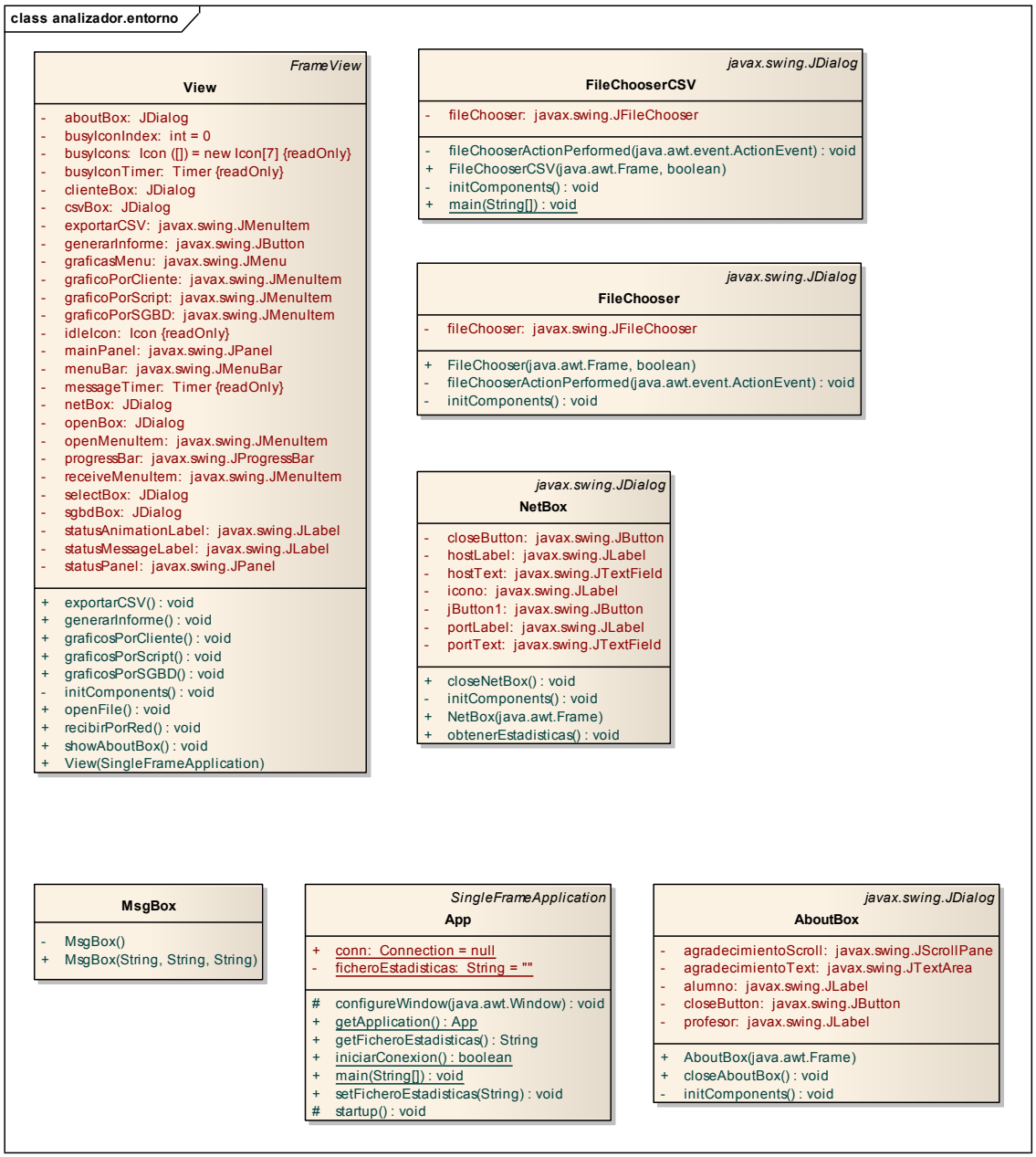

Ilustración 31

5.6.2 *JOSEJAMILENA::PFC::ANALIZADOR::SQL*

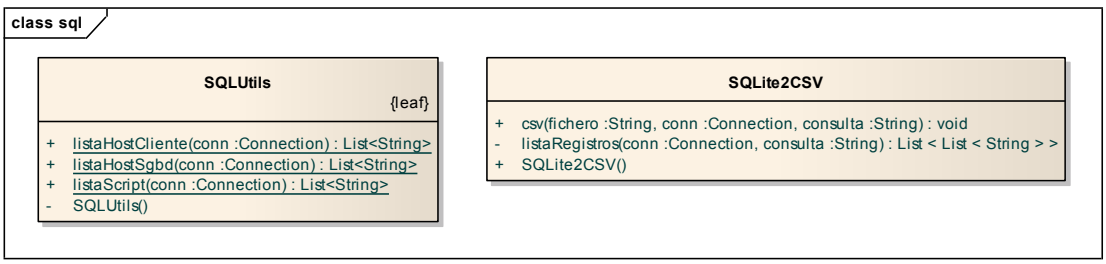

Ilustración 32

5.6.3 *DIAGRAMAS DE CASOS DE USO*

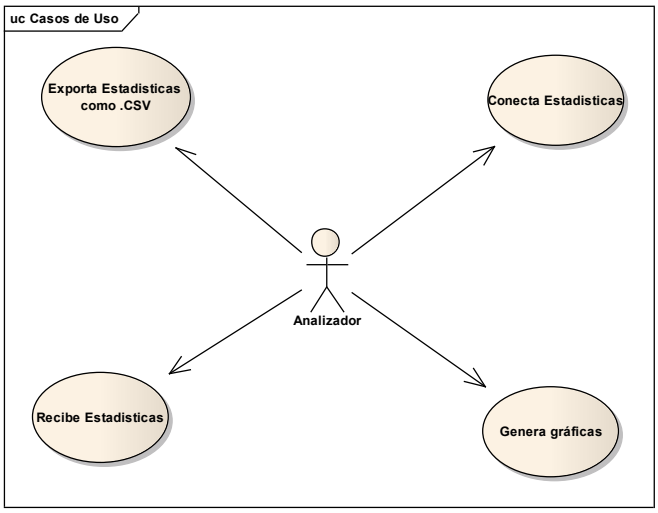

Ilustración 33

5.6.4 *DIAGRAMAS DE COMUNICACIONES*

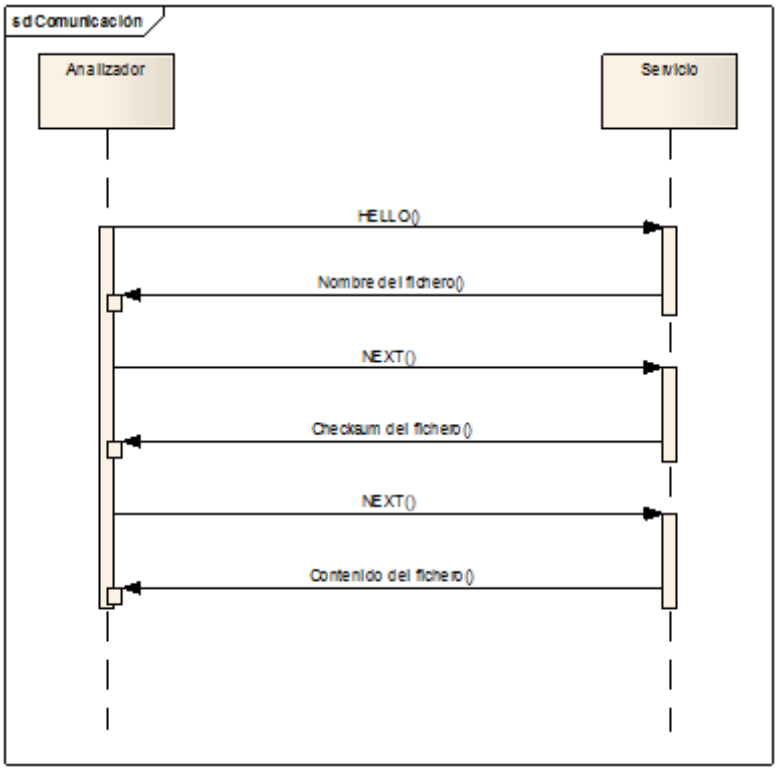

Ilustración 34

5.6.5 DIAGRAMAS DE ACTIVIDADES

5.6.5.1 Generar una gráfica por SGBD

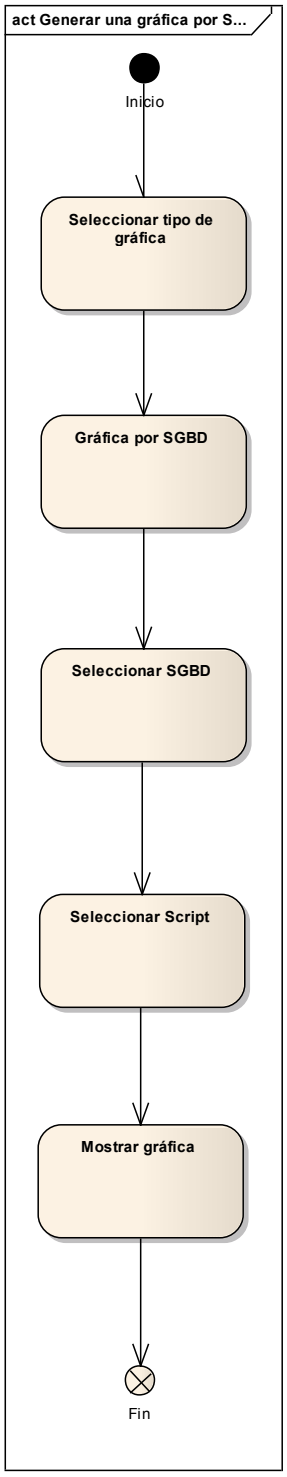

Ilustración 35

5.6.5.2 Generar una gráfica por Script

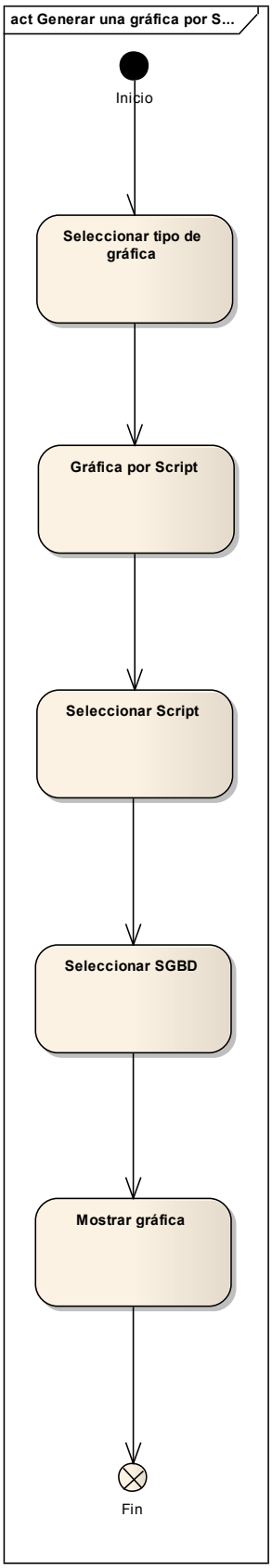

Ilustración 36

5.6.5.3 Generar una gráfica por Host Cliente

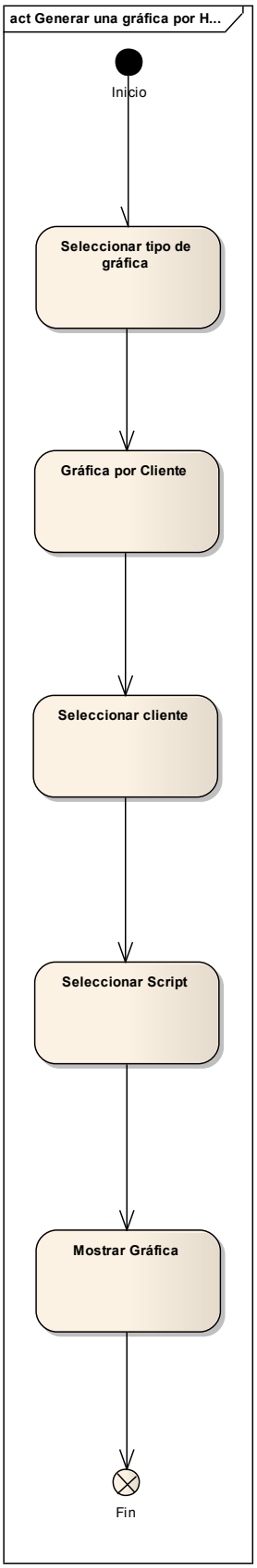

Ilustración 37

5.6.5.4 Exportar datos a archivo .CSV

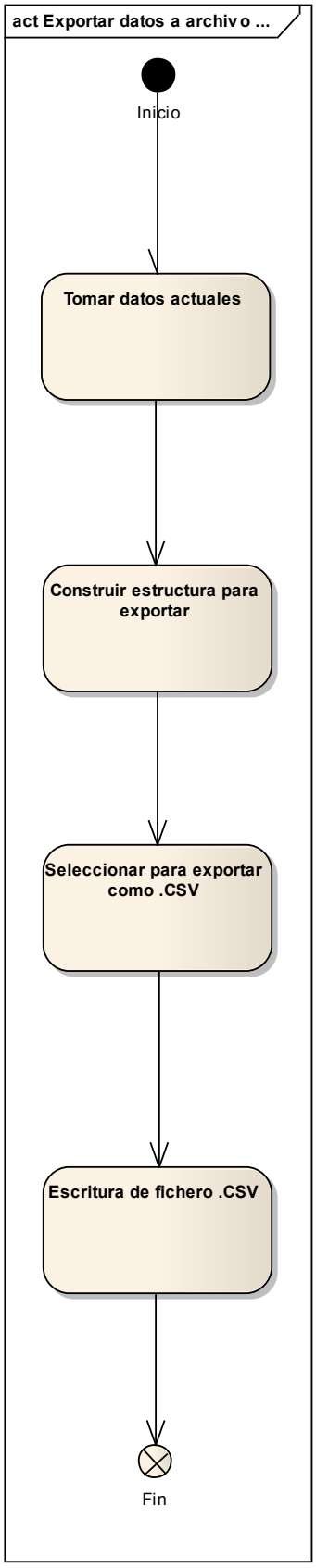

Ilustración 38

5.6.5.5 Abrir archivo de Estadísticas

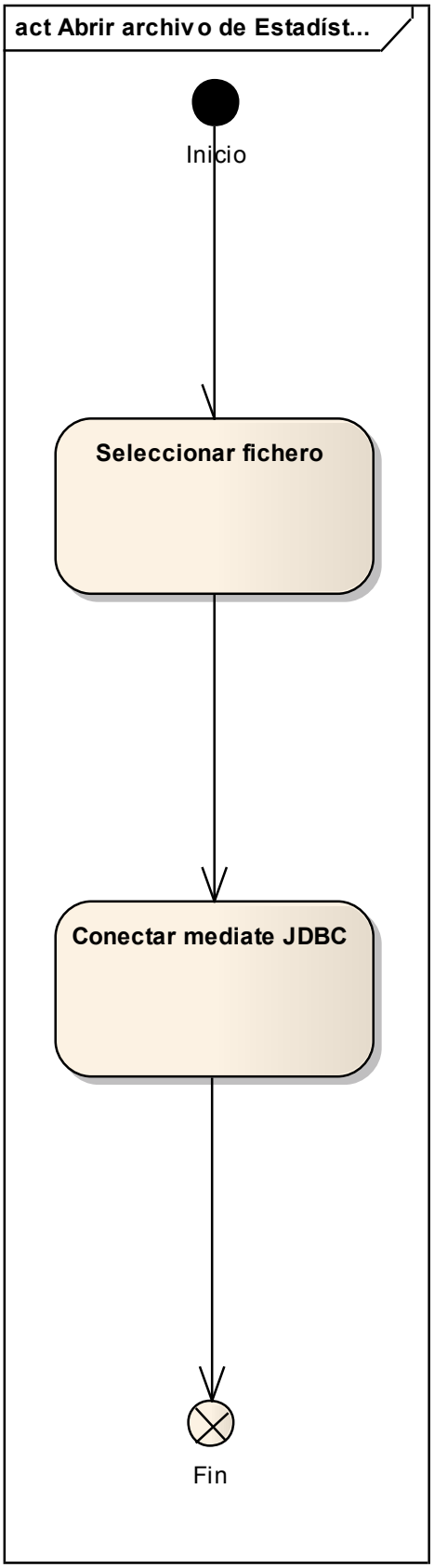

Ilustración 39

5.6.5.6 Cargar archivo de Estadísticas por Red

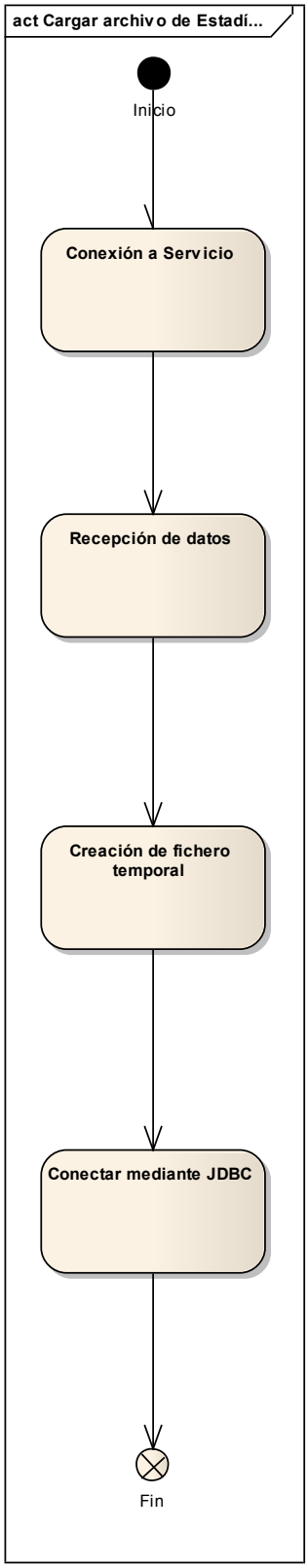

Ilustración 40

5.7 TABLA DE ESTADÍSTICAS

Ilustración 41

La base de datos de estadísticas tan solo contiene una tabla. En dicha tabla se almacenan los datos obtenidos por las métricas de tiempos llevadas a cabo por Servicio.

En la tabla se almacena:

- Tiempo. Es el tiempo de latencia de la petición
- Fecha. Fecha y hora de la muestra.
- Tipo. Tipo o Script del que se han tomado las métricas.
- Host_sgbd. Nombre del host que despliega el sistema gestor de bases de datos.
- Host_cliente. Nombre del host cliente que ha lanzado la petición.

5.8 ACLARACIONES SOBRE EL DESARROLLO.

5.8.1 EN REFERENCIA A LA FORMA DE ALMACENAMIENTO DE ESTADÍSTICAS.

Cuando nos referimos a abrir un fichero de estadísticas dentro de la aplicación Analizador, nos referimos a estadísticas almacenadas en un fichero SQLite, ya que este es el formato por defecto para dichos datos.

Si los datos se han guardado en un sistema de bases de datos relacional como puede ser Oracle, SQL Server,…, para obtener los datos a analizar hemos de hacerlo con la opción de recibir fichero por red.

La opción de recibir fichero por red hace una llamada a Servicio para que el nos envíe los datos con los que está trabajando y Analizador se encarga de almacenarlos en el formato nativo de este, es decir, SQLite.

5.8.2 EN REFERENCIA A LOG4J Y SU CAPACIDAD DE ALMACENAR EL DIARIO DE EJECUCIÓN EN UNA TABLA DE UNA BASE DE DATOS MEDIANTE JDBC

Log4j tiene mecanismos para guardar los diarios de ejecución que va redactando en distintos formatos. Por defecto este proyecto viene configurado para almacenar los diarios de ejecución en un fichero de texto. El objetivo de esta nota era aclarar porque no se emplea un fichero SQLite para almacenar los diarios de ejecución. La respuesta es simple. Un requisito de todas las aplicaciones que forman este proyecto es el bajo consumo de recursos. Emplear la conexión JDBC con Log4J aumenta el consumo de recursos, además de que puede ser un foco de errores innecesarios. Por eso se opto por un fichero de texto. Hay que decir que todo esto es configurable sin que afecte al código, luego si se quiere se pude hacer.

6 CALIDAD DEL SOFTWARE

Todas las metodologías y herramientas tienen un único fin producir software de gran calidad.

¿Cómo podemos definir la calidad del software?

"Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente" R. S. Pressman (1992).

"El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas" ISO 8402 (UNE 66-001-92).

Luego resumiendo, los requisitos del software son la base de las medidas de calidad, la falta de concordancia con los requisitos es una falta de calidad.

Los estándares o metodologías definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la ingeniería del software. Si no se sigue ninguna metodología siempre habrá falta de calidad

Existen algunos requisitos implícitos que a menudo no se mencionan, que también pueden implicar una falta de calidad. Estos requisitos son tan importantes como planificar un mantenimiento, generar código estándar que facilite el mantenimiento. La verificación formal del mismo, o por lo menos la verificación formal estática.

Cómo intentar asegurar la calidad del software

La garantía de calidad del software es el conjunto de actividades planificadas y sistemáticas necesarias para aportar los mecanismos de confianza en que el producto software satisfará los requisitos dados de calidad.

Los planteamientos de calidad del software es un proceso previo y propio de cada aplicación antes de comenzar a desarrollarla y no después.

La garantía de calidad del software está presente en:

- Los métodos y herramientas de análisis, diseño, programación y prueba. En el caso de este proyecto en cuestión, se ha empleado UML y Desarrollo Iterativo Incremental.
- Inspecciones técnicas formales en todos los pasos del proceso de desarrollo del software. Para esto se ha empleado PMD y Checkstyle.
- Estrategias de prueba parcial y total. JUnit y pruebas parciales de prototios.
- Control de la documentación del software y de los cambios realizados. Durante el desarrollo se ha empleado Subversion.
- Procedimientos para ajustarse a los estándares y dejar claro cuando se está fuera de ellos. Esto no es aplicable a este proyecto, ya que es plenamente estándar.
- Mecanismos de medida o métricas.
- Registro de auditorías y realización de informes.

Actividades para asegurar la garantía de calidad del software:

- Métricas de software para el control del proyecto.
- Verificación y validación del software a lo largo del ciclo de vida.

Incluye las pruebas y los procesos de revisión e inspección y la gestión de la configuración del software.

Gestión de la calidad del software, estándares internacionales

Aquí damos un vistazo a los estándares internacionales que están orientados a prácticas empresariales. No es aplicable a este proyecto, pero tiene una utilidad didáctica.

Gestión de la calidad (ISO 9000). Conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades y se implanta por medios tales como la planificación de la calidad, el control de la calidad, el garantía de la calidad y la mejora de la calidad, en el marco del sistema de calidad.

Política de calidad (ISO 9000). Directrices y objetivos generales de una organización, directrices y objetivos a la calidad, tal como se expresan formalmente por la alta dirección.

La gestión de la calidad se aplica normalmente a nivel de empresa.

También puede haber una gestión de calidad dentro de la gestión de cada proyecto.

Control de la calidad del software

Son las técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad, centradas en dos objetivos fundamentales:

- Mantener bajo control un proceso.
- Eliminar las causas de los defectos en las diferentes fases del ciclo de vida

En general son las actividades para evaluar la calidad de los productos desarrollados.

Sistema de calidad

Los sistema de calidad están formados por una estructura organizativa, unos procedimientos para garantizar la calidad y un conjunto de procesos y recursos necesarios para implantar la gestión de calidad.

El sistema de calidad se debe adecuar a los objetivos de calidad de la empresa. Son políticas globales.

La dirección de la empresa es la responsable de fijar la política de calidad y las decisiones relativas a iniciar, desarrollar, implantar y actualizar el sistema de calidad.

Un sistema de calidad consta de varias partes:

- Documentación.

- Manual de calidad. Es el documento principal para establecer e implantar un sistema de calidad. Puede haber manuales a nivel de empresa, departamento, producto, específicos como para compras, proyectos,…
- Parte física: locales, herramientas ordenadores,…
- Aspectos humanos:
    - Formación de personal.
    - Creación y coordinación de equipos de trabajo.
- Normativas para garantizar la calidad:
    - ISO 9000: Gestión y aseguramiento de calidad, son conceptos y directrices generales.
    - Recomendaciones externas para aseguramiento de la calidad (ISO 9001, ISO 9002, ISO 9003). Son extensiones de la anterior.
    - Recomendaciones internas para aseguramiento de la calidad (ISO 9004). Un extra.

Factores que determinan la calidad del software

Se clasifican en tres grupos:

- Operaciones del producto: características operativas.
    - Corrección. ¿Hace lo que se le pide? El grado en que una aplicación satisface sus especificaciones y consigue los objetivos encomendados por el cliente.
    - Fiabilidad. ¿Lo hace de forma fiable todo el tiempo? El grado que se puede esperar de una aplicación lleve a cabo las operaciones especificadas y con la precisión requerida.
    - Eficiencia. ¿Qué recursos hardware y software necesito? La cantidad de recursos hardware y software que necesita una aplicación para realizar las operaciones con los tiempos de respuesta adecuados.
    - Integridad. ¿Puedo controlar su uso? El grado con que puede controlarse el acceso al software o a los datos a personal no autorizado.
    - Facilidad de uso. ¿Es fácil y cómodo de manejar? El esfuerzo requerido para aprender el manejo de una aplicación, trabajar con ella, introducir datos y conseguir resultados.

- Revisión del producto. Capacidad para soportar cambios. Su extensibilidad.
    - Facilidad de mantenimiento. ¿Puedo localizar los fallos? El esfuerzo requerido para localizar y reparar errores.
    - Flexibilidad. ¿Puedo añadir nuevas opciones? El esfuerzo requerido para modificar una aplicación en funcionamiento.
    - Facilidad de prueba. ¿Puedo probar todas las opciones? El esfuerzo requerido para probar una aplicación de forma que cumpla con lo especificado en los requisitos.
- Transición del producto: adaptabilidad a nuevos entornos
    - Portabilidad. ¿Podré usarlo en otra máquina? El esfuerzo requerido para transferir la aplicación a otro hardware o sistema operativo.
    - Reusabilidad. ¿Podré utilizar alguna parte del software en otra aplicación? Grado en que partes de una aplicación pueden utilizarse en otras aplicaciones.
    - Interoperabilidad. (¿Podrá comunicarse con otras aplicaciones o sistemas informáticos? El esfuerzo necesario para comunicar la aplicación con otras aplicaciones o sistemas informáticos.

Métricas de la calidad del sofware

Es difícil, y en algunos casos imposibles, desarrollar medidas directas de los factores de calidad del software. Cada factor de calidad Fc se puede obtener como combinación de una o varias métricas:

$$Fc = \sum_{i=1}^{n} c_i * m_i$$

Siendo c_i factor de ponderación de la métrica i, que dependerá de cada aplicación específica y m_i métrica i. Habitualmente se puntúan de 0 a 10 en las métricas y en los factores de calidad.

Métricas para determinar los factores de calidad:

- Facilidad de auditoria.
- Exactitud.

- Normalización de las comunicaciones.
- Completitud.
- Concisión.
- Consistencia.
- Estandarización de los datos.
- Tolerancia de errores.
- Eficiencia de la ejecución.
- Facilidad de expansión.
- Generalidad.
- Independencia del hardware.
- Instrumentación.
- Modularidad.
- Facilidad de operación.
- Seguridad.
- Autodocuemntación.
- Simplicidad.
- Independencia del sistema software.
- Facilidad de traza.
- Formación.

Autoevaluación

Los pilares básicos de la certificación de calidad del software son:

- Una metodología adecuada.
- Un medio de valoración de la metodología.
- Un reconocimiento de la industria de la metodología utilizada y del medio de valorar la metodología.
- Todas las afirmaciones anteriores son correctas.
- Ninguna respuesta anterior es correcta.

La calidad del software implica:

- La concordancia entre el software diseñado y los requisitos.

- Seguir un estándar o metodología en el proceso de desarrollo de software.
- Tener en cuenta los requisitos implícitos, no expresados por los usuarios.
- Todas las afirmaciones anteriores son correctas.
- Ninguna respuesta anterior es correcta.

7 CONCLUSIONES

8 UN MEDIDOR DE RENDIMIENTO DE SERVIDORES DE BASES DE DATOS RELACIONALES

8.1 JOSEJAMILENA::PFC::ANALIZADOR

8.1.1 ANALIZADOR::ABOUTBOX

Tipo: Clase Pública

Implementa: javax.swing.JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:10. Modificado el 16/07/2009 11:32:10.

Autor: Jose Antonio Jamilena Daza

Esta clase muestra la ventana de "Acerca de…" de la interfaz gráfica.

8.1.1.1 Atributos de analizador::AboutBox

Atributos	Tipo	Notas
agradecimientoScroll	privado : javax.swing.JScro	

	llPane	
agradecimientoText	privado : javax.swing.JTextArea	
alumno	privado : javax.swing.JLabel	
closeButton	privado : javax.swing.JButton	
profesor	privado : javax.swing.JLabel	

8.1.1.2 Métodos de analizador::AboutBox

Método	Tipo	Notas
AboutBox (java.awt.Frame)	público:	
closeAboutBox ()	público: void	
initComponents ()	privado: void	

8.1.2 ANALIZADOR::APP

Tipo: Clase Pública

Implementa: SingleFrameApplication.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:10. Modificado el 16/07/2009 11:32:10.

Autor: Jose Antonio Jamilena Daza

La clase principal de la aplicación.

8.1.2.1 ATRIBUTOS DE ANALIZADOR::APP

Atributos	Tipo	Notas
conn	público y de clase: Connection	Conexión JDBC.
ficheroEstadisticas	privado static : String	Fichero de BD de estadísticas.

8.1.2.2 MÉTODOS DE ANALIZADOR::APP

Método	Tipo	Notas

startup ()	protegido: void	
configureWindow (java.awt.Window)	protegido: void	
getApplication ()	público y de clase: App	
getFicheroEstadisticas ()	público: String	
setFicheroEstadisticas (String)	público: void	
iniciarConexion ()	público y de clase: boolean	
main (String[])	público y de clase: void	

8.1.3 ANALIZADOR::FILECHOOSER

Tipo: Clase Pública

Implementa: javax.swing.JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:10. Modificado el 16/07/2009 11:32:10.

Autor: Jose Antonio Jamilena Daza

Está clase implementa el explorador de ficheros para apertura de un fichero de estadísticas.

8.1.3.1 ATRIBUTOS DE ANALIZADOR::FILECHOOSER

Atributos	Tipo	Notas
fileChooser	privado : javax.swing.JFileChooser	

8.1.3.2 MÉTODOS DE ANALIZADOR::FILECHOOSER

Método	Tipo	Notas
FileChooser (java.awt.Frame, boolean)	público:	
initComponents ()	privado: void	
fileChooserActionPerformed (java.awt.event.ActionEvent)	privado: void	

8.1.4 ANALIZADOR::FILECHOOSERCSV

Tipo: Clase Pública

Implementa: javax.swing.JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:10. Modificado el 16/07/2009 11:32:10.

Autor: Jose Antonio Jamilena Daza

Está clase implementa el explorador de ficheros para la exportación como fichero de valores separado por comas del fichero de estadísticas actual.

8.1.4.1 ATRIBUTOS DE ANALIZADOR::FILECHOOSERCSV

Atributos	Tipo	Notas
fileChooser	privado : javax.swing.JFileChooser	

8.1.4.2 MÉTODOS DE ANALIZADOR::FILECHOOSERCSV

Método	Tipo	Notas

FileChooserCSV (java.awt.Frame, boolean)	público:	
initComponents ()	privado: void	
fileChooserActionPerformed (java.awt.event.ActionEvent)	privado: void	
main (String[])	público y de clase: void	

8.1.5 ANALIZADOR::GRAFICOPORCLIENTE

Tipo: Clase Pública

Implementa: JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:11. Modificado el 16/07/2009 11:32:11.

Autor: Jose Antonio Jamilena Daza

Genera gráficas por cliente.

8.1.5.1 MÉTODOS DE ANALIZADOR::GRAFICOPORCLIENTE

Método	Tipo	Notas
GraficoPorCliente (String, String)	público:	

8.1.6 ANALIZADOR::GRAFICOPORSCRIPT

Tipo: Clase Pública

Implementa: JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:11. Modificado el 16/07/2009 11:32:11.

Autor: Jose Antonio Jamilena Daza

Genera gráficas por script.

8.1.6.1 MÉTODOS DE ANALIZADOR::GRAFICOPORSCRIPT

Método	Tipo	Notas
GraficoPorScript (String,	público:	

String)		

8.1.7 ANALIZADOR::GRAFICOPORSGBD

Tipo: Clase Pública

Implementa: JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:11. Modificado el 16/07/2009 11:32:11.

Autor: Jose Antonio Jamilena Daza

Genera gráficas por sistema gestor de bases de datos.

8.1.7.1 MÉTODOS DE ANALIZADOR::GRAFICOPORSGBD

Método	Tipo	Notas
GraficoPorSGBD (String, String)	público:	

8.1.8 ANALIZADOR::MSGBOX

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:11. Modificado el 16/07/2009 11:32:11.

Autor: Jose Antonio Jamilena Daza

Genera mensajes emergentes.

8.1.8.1 MÉTODOS DE ANALIZADOR::MSGBOX

Método	Tipo	Notas
MsgBox ()	privado:	
MsgBox (String, String, String)	público:	

8.1.9 ANALIZADOR::NETBOX

Tipo: Clase Pública

Implementa: javax.swing.JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:11. Modificado el 16/07/2009 11:32:11.

Autor: Jose Antonio Jamilena Daza

Abre la ventana para pedir los datos de conexión a servicio por red, para poder obtener el fichero de estadísticas de dicho servicio.

8.1.9.1 Atributos de analizador::NetBox

Atributos	Tipo	Notas
closeButton	privado : javax.swing.JButton	
hostLabel	privado : javax.swing.JLabel	
hostText	privado : javax.swing.JTextField	
icono	privado : javax.swing.JLabel	
jButton1	privado : javax.swing.JButton	
portLabel	privado : javax.swing.JLa	

	bel	
portText	privado : javax.swing.JTextField	

8.1.9.2 Métodos de analizador::NetBox

Método	Tipo	Notas
NetBox (java.awt.Frame)	público:	
initComponents ()	privado: void	
closeNetBox ()	público: void	
obtenerEstadisticas ()	público: void	

8.1.10 ANALIZADOR::SELECCIONARCLIENTE

Tipo: Clase Pública

Implementa: javax.swing.JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:12. Modificado el 16/07/2009 11:32:12.

Autor: Jose Antonio Jamilena Daza

Selecciona un host cliente.

8.1.10.1 Atributos de analizador::SeleccionarCliente

Atributos	Tipo	Notas
lista	privado : List<String>	
icono	privado : javax.swing.JLabel	
listaScript	privado : javax.swing.JComboBox	
siguienteButton	privado : javax.swing.JButton	

8.1.10.2 Métodos de analizador::SeleccionarCliente

Método	Tipo	Notas
SeleccionarCliente (java.awt.Frame, List<String>)	público:	
siguienteSelecionarScriptBox ()	público: void	

initComponents ()	privado: void	

8.1.11 ANALIZADOR::SELECCIONARCLIENTESCRIPT

Tipo: Clase Pública

Implementa: javax.swing.JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:12. Modificado el 16/07/2009 11:32:12.

Autor: Jose Antonio Jamilena Daza

Selecciona un script, habiendo sido previamente seleccionado un cliente

8.1.11.1 Atributos de analizador::SeleccionarClienteScript

Atributos	Tipo	Notas
lista	privado : List<String>	
hostCliente	privado : String	

cargarGrafica	privado : javax.swing.JButton	
closeButton	privado : javax.swing.JButton	
icono	privado : javax.swing.JLabel	
listaScript	privado : javax.swing.JComboBox	

8.1.11.2 Métodos de analizador::SeleccionarClienteScript

Método	Tipo	Notas
SeleccionarClienteScript (java.awt.Frame, List<String>, String)	público:	
closeScriptBox ()	público: void	
initComponents ()	privado: void	
cargarGrafica ()	público: void	

8.1.12 ANALIZADOR::SELECCIONARSCRIPT

Tipo: Clase Pública

Implementa: javax.swing.JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:12. Modificado el 16/07/2009 11:32:12.

Autor: Jose Antonio Jamilena Daza

8.1.12.1 Atributos de analizador::SeleccionarScript

Atributos	Tipo	Notas
lista	privado : List<String>	
icono	privado : javax.swing.JLabel	
listaScript	privado : javax.swing.JComboBox	
siguienteButton	privado : javax.swing.JButton	

8.1.12.2 Métodos de analizador::SeleccionarScript

Método	Tipo	Notas
SeleccionarScript (java.awt.Frame, List<String>)	público:	
siguienteSelecionarScript Box ()	público: void	
initComponents ()	privado: void	

8.1.13 ANALIZADOR::SELECCIONARSCRIPTSGBD

Tipo: Clase Pública

Implementa: javax.swing.JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:12. Modificado el 16/07/2009 11:32:12.

Autor: Jose Antonio Jamilena Daza

Selecciona un sistema gestor de bases de datos, previamente seleccionando un script.

8.1.13.1 Atributos de analizador::SeleccionarScriptSGBD

Atributos	Tipo	Notas
lista	privado : List<String>	
sscript	privado : String	
cargarGrafica	privado : javax.swing.JButton	
closeButton	privado : javax.swing.JButton	
icono	privado : javax.swing.JLabel	
listaScript	privado : javax.swing.JComboBox	

8.1.13.2 Métodos de analizador::SeleccionarScriptSGBD

Método	Tipo	Notas
SeleccionarScriptSGBD (java.awt.Frame, List<String>, String)	público:	

closeScriptBox ()	público: void	
initComponents ()	privado: void	
cargarGrafica ()	público: void	

8.1.14 ANALIZADOR::SELECCIONARSGBD

Tipo: Clase Pública

Implementa: javax.swing.JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:12. Modificado el 16/07/2009 11:32:12.

Autor: Jose Antonio Jamilena Daza

Selecciona un sistema gestor de bases de datos.

8.1.14.1 Atributos de analizador::SeleccionarSGBD

Atributos	Tipo	Notas
lista	privado : List<String>	
icono	privado : javax.swing.JLabel	

listaScript	privado : javax.swing.JComboBox	
siguienteButton	privado : javax.swing.JButton	

8.1.14.2 Métodos de analizador::SeleccionarSGBD

Método	Tipo	Notas
SeleccionarSGBD (java.awt.Frame, List<String>)	público:	
siguienteSelecionarScriptBox ()	público: void	
initComponents ()	privado: void	

8.1.15 ANALIZADOR::SELECCIONARSGBDSCRIPT

Tipo: Clase Pública

Implementa: javax.swing.JDialog.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:13. Modificado el 16/07/2009 11:32:13.

Autor: Jose Antonio Jamilena Daza

Selecciona un script, previamente habiendo seleccionado un sistema gestor de bases de datos.

8.1.15.1 Atributos de analizador::SeleccionarSGBDScript

Atributos	Tipo	Notas
lista	privado : List<String>	
hostSGBD	privado : String	
cargarGrafica	privado : javax.swing.JButton	
closeButton	privado : javax.swing.JButton	
icono	privado : javax.swing.JLabel	
listaScript	privado : javax.swing.JComboBox	

8.1.15.2 Métodos de analizador::SeleccionarSGBDScript

Método	Tipo	Notas
SeleccionarSGBDScript (java.awt.Frame, List<String>, String)	público:	
closeSelecionarScriptBox ()	público: void	
initComponents ()	privado: void	
cargarGrafica ()	público: void	

8.1.16 ANALIZADOR::VIEW

Tipo: Clase Pública

Implementa: FrameView.

Estado: Versión 1.0. Fase 1.0.

Paquete: analizador

Detalles: Creado el 16/07/2009 11:32:13. Modificado el 16/07/2009 11:32:13.

Autor: Jose Antonio Jamilena Daza

Pantalla principal de la aplicación.

8.1.16.1 Atributos de analizador::View

Atributos	Tipo	Notas
exportarCSV	privado : javax.swing.JMenuItem	
generarInforme	privado : javax.swing.JButton	
graficasMenu	privado : javax.swing.JMenu	
graficoPorCliente	privado : javax.swing.JMenuItem	
graficoPorSGBD	privado : javax.swing.JMenuItem	
graficoPorScript	privado : javax.swing.JMenuItem	
mainPanel	privado : javax.swing.JPanel	
menuBar	privado : javax.swing.JMe	

	nuBar	
openMenuItem	privado : javax.swing.JMenuItem	
progressBar	privado : javax.swing.JProgressBar	
receiveMenuItem	privado : javax.swing.JMenuItem	
EstadoAnimationLabel	privado : javax.swing.JLabel	
EstadoMessageLabel	privado : javax.swing.JLabel	
EstadoPanel	privado : javax.swing.JPanel	
messageTimer	privado no modificable : Timer	
busyIconTimer	privado no modificable :	

	Timer	
idleIcon	privado no modificable : Icon	
busyIcons	privado no modificable : Icon	
busyIconIndex	privado : int	
aboutBox	privado : JDialog	
openBox	privado : JDialog	
selectBox	privado : JDialog	
sgbdBox	privado : JDialog	
clienteBox	privado : JDialog	
netBox	privado : JDialog	
csvBox	privado :	

	JDialog	

8.1.16.2 Métodos de analizador::View

Método	Tipo	Notas
View (SingleFrameApplication)	público:	
openFile ()	público: void	
showAboutBox ()	público: void	
initComponents ()	privado: void	
graficosPorScript ()	público: void	
graficosPorSGBD ()	público: void	
graficosPorCliente ()	público: void	
generarInforme ()	público: void	
recibirPorRed ()	público: void	
exportarCSV ()	público: void	

8.2 JOSEJAMILENA::PFC::ANALIZADOR::SQL

8.2.1 SQL::SQLITE2CSV

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: sql

Detalles: Creado el 16/07/2009 11:32:09. Modificado el 16/07/2009 11:32:09.

Autor: Jose Antonio Jamilena Daza

Convierte datos SQLite a fichero CSV.

8.2.1.1 Métodos de sql::SQLite2CSV

Método	Tipo	Notas
SQLite2CSV ()	público:	
listaRegistros (Connection, String)	privado: List < List < String > >	
csv (String, Connection, String)	público no modificable: void	

8.2.2 SQL::SQLUTILS

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: sql

Detalles: Creado el 16/07/2009 11:32:09. Modificado el 16/07/2009 11:32:09.

Autor: Jose Antonio Jamilena Daza

Utilidades SQL. No se permiten instancias.

8.2.2.1 Métodos de sql::SQLUtils

Método	Tipo	Notas
SQLUtils ()	privado:	
listaScript (Connection)	público y de clase: List<String>	
listaHostSgbd (Connection)	público y de clase: List<String>	
listaHostCliente (Connection)	público y de clase: List<String>	

8.3 JOSEJAMILENA::PFC::SERVIDOR::CHARTSERVER

8.3.1 CHARTSERVER::CLIENTHANDLER

Tipo: Clase de paquete

Implementa: Thread.

Estado: Versión 1.0. Fase 1.0.

Paquete: chartserver

Detalles: Creado el 16/07/2009 11:33:53. Modificado el 16/07/2009 11:33:53.

Autor: Jose Antonio Jamilena Daza

Hilo de ejecución del servidor.

8.3.1.1 ATRIBUTOS DE CHARTSERVER::CLIENTHANDLER

Atributo	Tipo	Notas
logger	privado y de clase: Logger	Logger. Valor inicial: Logger.getLogger(ClientHandler.class);
miSocketServidor	privado : Socket	Socket.
nombreFichero	privado : String	Nombre de fichero index para este hilo.
bufferSize	privado no modificable:	Tamaño del buffer de lectura.

	int	Valor inicial: 4096;
numGrupos	privado no modificable : int	Número de grupos de estadísticas. Valor inicial: 3;
longitudNombre	privado no modificable : int	Longitud del nombre de fichero. Valor inicial: 8;

8.3.1.2 Métodos de chartserver::ClientHandler

Método	Tipo	Notas
ClientHandler (Socket)	público:	Parámetro: s [Socket - in] socket Hilo de ejecución del navegado cliente. Genera una pá¡gina web para el explorador cliente segúnn los datos disponibles.
run ()	público: void	Método que ejecuta el Thread. annotations = '@Override'

8.3.2 CHARTSERVER::GRAFICO

Tipo: Clase de paquete

Estado: Versión 1.0. Fase 1.0.

Paquete: chartserver

Detalles: Creado el 16/07/2009 11:33:53. Modificado el 16/07/2009 11:33:53.

Autor: Jose Antonio Jamilena Daza

Sus instancias son objetos que se emplean para definir un Grafico.

8.3.2.1 Atributos de chartserver::Grafico

Atributo	Tipo	Notas
tituloEjeX	privado : String	Titulo para el eje X.
tituloEjeY	privado : String	Titulo para el eje Y.
lista	privado : LinkedList < String >	lista de consultas.

8.3.2.2 Métodos de chartserver::Grafico

Método	Tipo	Notas
Grafico ()	público:	Constructor.

Grafico (String, String, List<String>)	público:	Parámetro: tituloX [String - in] nombre eje X Parámetro: tituloY [String - in] nombre eje Y Parámetro: list [List<String> - in] lista Constructor.
getLista ()	público: List<String>	Obtiene la lista. Devuelve: lista
setLista (List < String >)	público: void	Parámetro: list [List < String > - in] lista Establece la lista.
getTituloEjeX ()	público: String	Obtiene nombre del eje X. Devuelve: nombre
setTituloEjeX (String)	público: void	Parámetro: tituloX [String - in] nombre Define el titulo del eje Y.
getTituloEjeY ()	público: String	Obtiene nombre del eje Y. Devuelve: nombre
setTituloEjeY (String)	público: void	Parámetro: tituloY [String - in] nombre

		Define el titulo del eje Y.

8.3.3 CHARTSERVER::GRUPOCONSULTA

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: chartserver

Detalles: Creado el 16/07/2009 11:33:53. Modificado el 16/07/2009 11:33:53.

Autor: Jose Antonio Jamilena Daza

Las instancias de esta clase son objetos formados por una lista de consultas, y un nombre para dicho grupo.

8.3.3.1 Atributos de chartserver::GrupoConsulta

Atributo	Tipo	Notas
titulo	privado : String	Nombre del grupo.
lista	privado : List < String >	Lista de consultas.

8.3.3.2 Métodos de chartserver::GrupoConsulta

Método	Tipo	Notas

GrupoConsulta (String, List < String >)	público:	Parámetro: titu [String - in] nombre del grupo. Parámetro: list [List < String > - in] lista de consultas. Constructor.
getTitulo ()	público no modificable: String	Devuelve: el titulo
setTitulo (String)	público no modificable: void	Parámetro: titu [String - in] el titulo to set
getLista ()	público no modificable: List < String >	Devuelve: the lista
setLista (List < String >)	público no modificable: void	Parámetro: list [List < String > - in] the lista to set

8.3.4 CHARTSERVER::SQLUTILS

Tipo: Clase de paquete

Estado: Versión 1.0. Fase 1.0.

Paquete: chartserver

Detalles: Creado el 16/07/2009 11:33:53. Modificado el 16/07/2009 11:33:53.

Autor: Jose Antonio Jamilena Daza

Utilidades SQL.

8.3.4.1 MÉTODOS DE CHARTSERVER::SQLUTILS

Método	Tipo	Notas
SQLUtils ()	privado:	No se permiten instancias.
consultaSQL2Grafico (Connection, String)	público static: Grafico	Parámetro: conn [Connection - in] conexion JDBC Parámetro: s [String - in] sentencia SQL Obtiene datos desde SQL y los convierte en Grafico. Devuelve: Grafico
listaGruposConsultas (Connection)	público static: List<GrupoConsulta>	Parámetro: conn [Connection - in] conexion JDBC Obtiene una lista de grupos de consulta.

		Devuelve: lista de grupos de consulta

8.3.5 CHARTSERVER::WEBSERVER

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: chartserver

Detalles: Creado el 16/07/2009 11:33:54. Modificado el 16/07/2009 11:33:54.

Autor: Jose Antonio Jamilena Daza

Servidor web para mostrar gráficas con Google Charts.

8.3.5.1 ATRIBUTOS DE CHARTSERVER::WEBSERVER

Atributo	Tipo	Notas
configProperties	privado y de clase: Properties	propiedades de configuración.
logger	privado y de clase: Logger	Logger. Valor inicial: Logger.getLogger(Webserver.class);
serverSocket	privado y de	Socket servidor.

	clase: ServerSocket	
httpPort	privado y de clase: int	Puerto HTTP empleado.

8.3.5.2 Métodos de chartserver::Webserver

Método	Tipo	Notas
Webserver ()	privado:	No se permiten instancias de la clase.
main (String[])	público static: void	Parámetro: args [String[] - in] puerto HTTP que abrirá, si no se indica, usa el 80. Método principal. throws = 'IOException,ClassNotFoundException' - @exception ClassNotFoundException no encuentra el driver JDBC @exception java.io.IOException Error.
getConfigProperties ()	público static: Properties	Devuelve: the configProperties
setConfigProperties (Properties)	público static: void	Parámetro: aConfigProperties [Properties - in] the configProperties to set

8.4 JOSEJAMILENA::PFC::SERVIDOR

8.4.1 SERVIDOR::MAIN

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: servidor

Detalles: Creado el 16/07/2009 11:35:20. Modificado el 16/07/2009 11:35:20.

Autor: Jose Antonio Jamilena Daza.

Cargador inicial.

8.4.1.1 SERVIDOR::MAIN ATTRIBUTES

Atributo	Tipo	Notas
logger	privado y de clase: Logger	Logger. Valor inicial: Logger.getLogger(Main.class);

8.4.1.2 Métodos de servidor::Main

Método	Tipo	Notas
Main ()	privado:	
main (String[])	público y de	Parámetro: args [String[] - in]

	clase: void	argumentos. main.

8.5 JOSEJAMILENA::PFC::SERVIDOR::CONEXION

8.5.1 CONEXION::COMUN

Tipo: Clase Pública

Estado:Versión 1.0. Fase 1.0.

Paquete: conexion

Detalles: Creado el 16/07/2009 11:35:17. Modificado el 16/07/2009 11:35:17.

Autor: Jose Antonio Jamilena Daza.

Métodos compartidos y conexiones. Diseñada con un patrón singleton.

8.5.1.1 Atributos de conexion::Comun

Atributo	Tipo	Notas
logger	privado y de clase: Logger	Logger. Valor inicial: Logger.getLogger(Comun.class);
comun	privado y de clase:	Instancia Singleton.

	Comun	Valor inicial: null;
estadisticas	privado y de clase: Estadisticas	Instancia de Estadisticas. Valor inicial: null;
hostname	privado y de clase: String	Hostname. Valor inicial: "";
configProperties	privado : Properties	properties de configuración.

8.5.1.2 Métodos de conexion::Comun

Método	Tipo	Notas
Comun ()	privado:	Constructor.
getComun ()	público y de clase: Comun	Obtiene instancia de Comun. Devuelve: instancia de Comun.
iniciarPropiedades ()	privado: void	Iniciar properties.
getConfigProperties ()	público: Properties	Devuelve: the ConfigProperties
getEstadisticas ()	público y de clase: Estadisticas	Devuelve: the estadisticas
getHostname ()	público: String	Devuelve: the hostname

8.5.2 CONEXION::CRONO

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: conexion

Detalles: Creado el 16/07/2009 11:35:18. Modificado el 16/07/2009 11:35:18.

Autor: Jose Antonio Jamilena Daza.

Crono se encarga de medir el tiempo.

8.5.2.1 Atributos de conexion::Crono

Atributo	Tipo	Notas
t0	privado : java.util.Date	Tiempo inicial.
t1	privado : java.util.Date	Tiempo final.

8.5.2.2 Métodos de conexion::Crono

Método	Tipo	Notas
Crono ()	público:	Constructor.

inicializa ()	público no modificable: void	Inicia la cuenta.
tiempo ()	público no modificable: long	Devuelve el tiempo desde el inicio de cuenta en segundos. Devuelve: tiempo desde el inicio.

8.5.3 CONEXION::ESTADISTICAS

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: conexion

Detalles: Creado el 16/07/2009 11:35:18. Modificado el 16/07/2009 11:35:18.

Autor: Jose Antonio Jamilena Daza.

Estadísticas.

8.5.3.1 Atributos de conexion::Estadisticas

Atributo	Tipo	Notas
logger	privado y de clase: Logger	Logger. Valor inicial: Logger.getLogger(Estadisticas.class);

uno	privado no modificable : int	1. Valor inicial: 1;
dos	privado no modificable : int	2. Valor inicial: 2;
tres	privado no modificable : int	3. Valor inicial: 3;
cuatro	privado no modificable : int	4. Valor inicial: 4;
cinco	privado no modificable : int	5. Valor inicial: 5;
conexion	privado : Connection	conexion. Valor inicial: null;
configProperties	privado : Properties	properties.

8.5.3.2 Métodos de conexion::Estadisticas

Método	Tipo	Notas

Estadisticas (String, String, String)	público:	Parámetro: driver [String - in] driver JDBC. Parámetro: url [String - in] enlace. Parámetro: properties [String - in] fichero de properties para Estadísticas. Constructor.
iniciarPropiedades (String)	privado: void	Parámetro: prop [String - in] nombre del fichero. Iniciar propiedades.
insertarEstadistica (long, String, String, String)	público no modificable: void	Parámetro: tiempo [long - in] tiempo. Parámetro: tipoConsultaStr [String - in] tipo de consulta como cadena de texto. Parámetro: sgbd [String - in] Parámetro: host [String - in] Insertar estadisticas en la base de datos.
iniciarEstructuraEnBD (Connection)	privado: void	Parámetro: conn [Connection - in] conexión a la base de datos. Inicializa la tabla en la que se guardan

		las estadísticas.
consultaSQL (String)	privado: String	Parámetro: s [String - in] consulta SQL. Lanza la consulta SQL. Devuelve: resultado.
mostrarMedia ()	público no modificable: String	Media de tiempos del fichero. Devuelve: tiempo de la media en segundos. throws = 'SQLException' - @exception java.sql.SQLException excepcion SQL.
mostrarMediaSeleccion ()	público no modificable: String	Media de tiempos en consultas de selección (SELECT). Devuelve: tiempo de la media en segundos. throws = 'SQLException' - @exception java.sql.SQLException excepcion SQL.
mostrarMediaManipulacionDatos ()	público no modificable: String	Media de tiempos en manipulación de datos (INSERT / UPDATE /DELETE). Devuelve: tiempo de la media en segundos. throws = 'SQLException' - @exception java.sql.SQLException excepcion

		SQL.
mostrarMediaManipulacionTablas ()	público no modificable: String	Media de tiempos en manipulación de tablas (DROP / CREATE). Devuelve: tiempo de la media en segundos. throws = 'SQLException' - @exception java.sql.SQLException excepcion SQL.
mostrarMediaOtrasOperaciones ()	público no modificable: String	Media de tiempos de otras operaciones (?). Devuelve: tiempo de la media en segundos. throws = 'SQLException' - @exception java.sql.SQLException excepcion SQL.
mostrarMediaOtrasOperacionesPrueba ()	público no modificable: String	Media de tiempos de otras operaciones de la última prueba (?). Devuelve: tiempo de la media en segundos. throws = 'SQLException' - @exception java.sql.SQLException excepcion SQL.

8.6 JOSEJAMILENA::PFC::SERVIDOR::TAREAS

8.6.1 TAREAS::CARGARTAREAS

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: tareas

Detalles: Creado el 16/07/2009 11:35:19. Modificado el 16/07/2009 11:35:19.

Autor: Jose Antonio Jamilena Daza.

Cargador de tareas.

8.6.1.1 Atributos de tareas::CargarTareas

Atributo	Tipo	Notas
logger	privado y de clase: Logger	Logger. Valor inicial: Logger.getLogger(CargarTareas.class);
tablaScript	privado : Map<String, String>	Tareas SQL. Valor inicial: null;

tablaConexionScript	privado : Map<String, TokenConexion>	Conexión SQL. Valor inicial: null;
tablaPlsql	privado : Map<String, String>	Tareas PLSQL. Valor inicial: null;
tablaConexionPlsql	privado : Map<String, TokenConexion>	Conexión PLSQL. Valor inicial: null;
numeroTokens	privado no modificable : int	Número de tokens que espera de cada línea del fichero de tareas. Valor inicial: 8;

8.6.1.2 Métodos de tareas::CargarTareas

Método	Tipo	Notas
CargarTareas ()	privado:	Constructor.
CargarTareas (String)	público:	Parámetro: nombreFichero [String - in] fichero. Carga las tareas del fichero. throws = 'IOException' - @exception java.io.IOException error de lectura.

getTablaScript ()	público no modificable: Map<String, String>	Devuelve: the tablaScript
getTablaPlsql ()	público no modificable: Map<String, String>	Devuelve: the tablaPlsql
getTablaConexionScript ()	público: Map<String, TokenConexion >	Devuelve: the tablaConexionScript
getTablaConexionPlsql ()	público: Map<String, TokenConexion >	Devuelve: the tablaConexionPlsql

8.6.2 TAREAS::TASKPLSQL

Tipo: Clase Pública

Implementa: Runnable.

Estado: Versión 1.0. Fase 1.0.

Paquete: tareas

Detalles: Creado el 16/07/2009 11:35:20. Modificado el 16/07/2009 11:35:20.

Autor: Jose Antonio Jamilena Daza.

Define una tarea PL/SQL.

8.6.2.1 Atributos tareas::TaskPLSQL

Atributo	Tipo	Notas
logger	privado y de clase: Logger	Logger. Valor inicial: Logger.getLogger(TaskPLSQL.class);
nombreFichero	privado : String	Nombre del fichero. Valor inicial: "";
nombreTarea	privado : String	Nombre de la tarea. Valor inicial: "";
token	privado : TokenConexion	Autenticación. Valor inicial: null;
crn	privado : Crono	Crono. Valor inicial: new Crono();
t	privado : long	Tiempo.

8.6.2.2 Métodos de tareas::TaskPLSQL

Método	Tipo	Notas
TaskPLSQL ()	privado:	No se permite el constructor sin parámetros.
TaskPLSQL (String, String, TokenConexion)	público:	Parámetro: nombre [String - in] nombre de la tarea. Parámetro: ficheroPLSQL [String - in] fichero de la tarea. Parámetro: tc [TokenConexion - in] autenticacion Constructor.
run ()	público no modificable: void	Sobrecarga el método.
obtenerTiempo ()	público no modificable: long	Obtiene tiempo. Devuelve: tiempo en segundos.

8.6.3 TAREAS::TASKSQL

Tipo: Clase Pública

Implementa: Runnable.

Estado: Versión 1.0. Fase 1.0.

Paquete: tareas

Detalles: Creado el 16/07/2009 11:35:20. Modificado el 16/07/2009 11:35:20.

Autor: Jose Antonio Jamilena Daza.

Define una tarea SQL.

8.6.3.1 ATRIBUTOS DE TAREAS::TASKSQL

Atributo	Tipo	Notas
logger	privado y de clase: Logger	Logger. Valor inicial: Logger.getLogger(TaskSQL.class);
nombreFichero	privado : String	Nombre del fichero. Valor inicial: "";
nombreTarea	privado : String	Nombre de la tarea. Valor inicial: "";
token	privado : TokenConexion	Autenticación. Valor inicial: null;
crn	privado : Crono	Crono. Valor inicial: new Crono();

t	privado : long	Tiempo.

8.6.3.2 Métodos de tareas::TaskSQL

Método	Tipo	Notas
TaskSQL ()	privado:	No se permite constructor sin parámetros.
TaskSQL (String, String, TokenConexion)	público:	Parámetro: nombre [String - in] nombre de la tarea. Parámetro: ficheroSQL [String - in] fichero scripts. Parámetro: tc [TokenConexion - in] autenticación Constructor.
run ()	público no modificable: void	Método que sobrescribe.
obtenerTiempo ()	público no modificable: long	Obtiene tiempo. Devuelve: tiempo en segundos.

8.7 JOSEJAMILENA::PFC::SERVIDOR::AUTENTICACION

8.7.1 AUTENTICACION::TOKENCONEXION

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: autenticacion

Detalles: Creado el 16/07/2009 11:35:18. Modificado el 16/07/2009 11:35:18.

Autor: Jose Antonio Jamilena Daza

Token de Autenticación.

8.7.1.1 Atributos de autenticacion::TokenConexion

Atributo	Tipo	Notas
hostname	privado : String	Hostname. Valor inicial: "";
driver	privado : String	Driver. Valor inicial: "";
cadenaConexion	privado : String	Cadena de conexión. Valor inicial: "";
usuario	privado : String	Usuario. Valor inicial: "";

password	privado : String	Contraseña. Valor inicial: "";

8.7.1.2 Métodos de autenticacion::TokenConexion

Método	Tipo	Notas
TokenConexion (String, String, String, String, String)	públ ico:	Parámetro: sgbd [String - in] sgbd Parámetro: drv [String - in] driver. Parámetro: url [String - in] cadena de conexión. Parámetro: user [String - in] usuario. Parámetro: passwd [String - in] contraseña. Constructor.
getCadenaConexion ()	público no modificable: String	Devuelve: the cadenaConexion
setCadenaConexion (String)	público no modificable: void	Parámetro: url [String - in] the cadenaConexion to set
getUsuario ()	público no modificable:	Devuelve: the usuario

	String	
setUsuario (String)	público no modificable: void	Parámetro: user [String - in] the usuario to set
getPassword ()	público no modificable: String	Devuelve: the password
setPassword (String)	público no modificable: void	Parámetro: passwd [String - in] the password to set
getDriver ()	público no modificable: String	Devuelve: the driver
setDriver (String)	público no modificable: void	Parámetro: drv [String - in] the driver to set
getHostname ()	público no modificable: String	Devuelve: the hostname
setHostname (String)	público no modificable: void	Parámetro: host [String - in] hostname the hostname to set

8.8 JOSEJAMILENA::PFC::SERVIDOR::RUNNER

8.8.1 RUNNER::PLSQLRUNNER

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: runner

Detalles: Creado el 16/07/2009 11:35:19. Modificado el 16/07/2009 11:35:19.

Autor: Jose Antonio Jamilena Daza.

Lanzador de script con procesos PL/SQL.

8.8.1.1 ATRIBUTOS DE RUNNER::PLSQLRUNNER

Atributo	Tipo	Notas
logger	privado y de clase: Logger	Logger. Valor inicial: Logger.getLogger(PlsqlRunner.class);
conexion	privado : Connection	conexion. Valor inicial: null;
listaDeConsultas	privado :	lista de consultas.

	List < String >	Valor inicial: null;

8.8.1.2 Métodos de runner::PlsqlRunner

Método	Tipo	Notas
PlsqlRunner (Connection)	público:	Parámetro: connection [Connection - in] conexión JDBC. Lanzador SQL.
PlsqlRunner (String, String, String, String)	público:	Parámetro: driver [String - in] driver. Parámetro: url [String - in] cadena de cconexión. Parámetro: username [String - in] usuario. Parámetro: password [String - in] contraseña. Lanzador SQL. throws = 'IOException,SQLException,ClassNotFoundException' - @exception java.lang.ClassNotFoundException driver no encontrado. @exception java.sql.SQLException fallo en JDBC. @exception java.io.IOException fallo de lectura.

consultaSQL (String)	privado: void	Parámetro: s [String - in] consulta SQL. Lanza la consulta indicada. throws = 'SQLException' - @exception java.sql.SQLException fallo.
runScript (Reader)	público no modificable: void	Parámetro: reader [Reader - in] reader. Leer script SQL. throws = 'IOException,SQLException' - @exception java.sql.SQLException fallo JDBC. @exception java.io.IOException error de lectura.

8.8.2 RUNNER::SQLRUNNER

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: runner

Detalles: Creado el 16/07/2009 11:35:19. Modificado el 16/07/2009 11:35:19.

Autor: Jose Antonio Jamilena Daza.

Lanzador de script con procesos SQL.

8.8.2.1 Atributos de runner::SqlRunner

Atributo	Tipo	Notas
logger	privado y de clase: Logger	Logger. Valor inicial: Logger.getLogger(SqlRunner.class);
conexion	privado : Connection	Conexión. Valor inicial: null;
listaDeConsultas	privado : List < String >	Lista de consultas. Valor inicial: null;

8.8.2.2 Métodos de runner::SqlRunner

Método	Tipo	Notas
SqlRunner (Connection)	público:	Parámetro: connection [Connection - in] conexión JDBC. Lanzador SQL.
SqlRunner (String, String, String, String)	público:	Parámetro: driver [String - in] driver. Parámetro: url [String - in] cadena de conexión. Parámetro: username [String - in]

		usuario. Parámetro: password [String - in] contraseña. Lanzador SQL. throws = 'IOException,SQLException,ClassNotFoundException' - @exception java.lang.ClassNotFoundException driver no encontrado. @exception java.sql.SQLException fallo en JDBC. @exception java.io.IOException fallo de lectura.
consultaSQL (String)	privado: void	Parámetro: s [String - in] consulta SQL. Lanza la consulta indicada. throws = 'SQLException' - @exception java.sql.SQLException fallo.
runScript (Reader)	público no modificable: void	Parámetro: reader [Reader - in] reader. Leer script SQL. throws = 'SQLException,IOException' - @exception java.sql.SQLException fallo JDBC. @exception java.io.IOException error

		de lectura.

8.9 JOSEJAMILENA::PFC::SERVIDOR::CRYPTO::EASY::CHECKSUM

8.9.1 CHECKSUM::CHECKSUM

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: checksum

Detalles: Creado el 16/07/2009 11:34:33. Modificado el 16/07/2009 11:34:33.

Autor: Jose Antonio Jamilena Daza

Comprobador de suma.

8.9.1.1 Métodos de checksum::Checksum

Método	Tipo	Notas
Checksum ()	privado:	No se podrá instanciar.
checksum (String)	público y de clase: long	Parámetro: fichero [String - in] fichero Obtiene la suma de comprobación del

		fichero dado. Devuelve: suma throws = 'IOException' - @exception java.io.IOException error

8.10 JOSEJAMILENA::PFC::SERVIDOR::TCP

8.10.1 TCP::FILERECEIVER

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: tcp

Detalles: Creado el 16/07/2009 11:34:34. Modificado el 16/07/2009 11:34:34.

Autor: Jose Antonio Jamilena Daza

Receptor de ficheros.

8.10.1.1 Métodos de tcp::FileReceiver

Método	Tipo	Notas
receive (String, int)	público y no modificable: void	Parámetro: hostname [String - in] host

		Parámetro: port [int - in] puerto Recibir.
receive2 (String, int)	público y no modificable: String	Parámetro: hostname [String - in] host Parámetro: port [int - in] puerto Recibir. Devuelve: nombre fichero throws = 'TCPException' - @exception TCPException error

8.10.2 TCP::FILESENDER

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: tcp

Detalles: Creado el 16/07/2009 11:34:34. Modificado el 16/07/2009 11:34:34.

Autor: Jose Antonio Jamilena Daza.

Servicio que envía los ficheros de BBDD de estadísticas por TCP.

8.10.2.1 Atributos de tcp::FileSender

Atributo	Tipo	Notas
port	privado y de clase: int	Puerto TCP.
file	privado y de clase: String	Nombre del fichero. Valor inicial: "";

8.10.2.2 Métodos de tcp::FileSender

Método	Tipo	Notas
getPort ()	público y de clase: int	Devuelve: the port
setPort (int)	público y de clase: void	Parámetro: aPort [int - in] the port to set
getFile ()	público y de clase: String	Devuelve: the file
setFile (String)	público y de clase: void	Parámetro: aFile [String - in] the file to set
FileSender (String, int)	público:	Parámetro: fileToSend [String - in] fichero

		Parámetro: tcpPort [int - in] puerto Constructor.
runServer ()	público y no modificable: void	Ejecutor.

8.10.3 TCP::FILESENDER::HANDLER

Tipo: Clase de paquete

Implementa: Thread.

Estado: Versión 1.0. Fase 1.0.

Paquete: tcp

Detalles: Creado el 16/07/2009 11:34:34. Modificado el 16/07/2009 11:34:34.

Autor: Jose Antonio Jamilena Daza

Hilo de ejecución.

8.10.3.1 Atributos de tcp::FileSender::Handler

Atributo	Tipo	Notas
sock	privado : Socket	Socket.

file	privado : String	Fichero.

8.10.3.2 Método de tcp::FileSender::Handler

Método	Tipo	Notas
Handler (Socket, String)	Paquete:	Parámetro: s [Socket - in] socket Parámetro: fileToSend [String - in] fichero Constructor.
run ()	público: void	Método que ejecuta. annotations = '@Override'
getSock ()	público: Socket	Devuelve: the sock
setSock (Socket)	público: void	Parámetro: aSock [Socket - in] the sock to set
getFile ()	público: String	Devuelve: the file
setFile (String)	público: void	Parámetro: aFile [String - in] the file to set

8.10.4 TCP::TCPEXCEPTION

Tipo: Clase Pública

Implementa: Exception.

Estado: Versión 1.0. Fase 1.0.

Paquete: tcp

Detalles: Creado el 16/07/2009 11:34:34. Modificado el 16/07/2009 11:34:34.

Autor: Jose Antonio Jamilena Daza

Excepción.

8.10.4.1 Métodos de tcp::TCPException

Método	Tipo	Notas
TCPException ()	público:	Creates a new instance of <code>TCPException</code> without detail message.
TCPException (String)	público:	Parámetro: msg [String - in] the detail message. Constructs an instance of <code>TCPException</code> with the specified detail message.

8.11 JOSEJAMILENA::PFC::SERVIDOR::UTIL

8.11.1 UTIL::CHANNELTOOLS

Tipo: Clase Pública

Estado: Versión 1.0. Fase 1.0.

Paquete: util

Detalles: Creado el 16/07/2009 11:34:35. Modificado el 16/07/2009 11:34:35.

Autor: Jose Antonio Jamilena Daza

Utilidades para trabajar con Channels.

8.11.1.1 Métodos de util::ChannelTools

Método	Tipo	Notas
ChannelTools ()	privado:	No instanciable.
fastChannelCopy (ReadableByteChannel, WritableByteChannel)	público y de clase: void	Parámetro: src [ReadableByteChannel - in] oringen Parámetro: dest [WritableByteChannel - in] destino Copia rápida por Channels. throws = 'IOException' - @exception java.io.IOException error

9 BIBLIOGRAFÍA

[1] Wikipedia. Iterative and incremental development. [En línea] http://en.wikipedia.org/wiki/Iterative_development.

[2] Sun Microsystems. Java™ Platform, Standard Edition 6. [En línea] http://java.sun.com/javase/6/docs/api/.

[3] Sun Microsystem. NetBeans. [En línea] http://www.netbeans.org/.

[4] Visualsvn. Visualsvn. [En línea] http://www.visualsvn.com/server/.

[5] TortoiseSVN. TortoiseSVN The coolest Interface to (Sub)Version Control. [En línea] http://tortoisesvn.net/.

[6] Sparx Systems. Enterprise Architect - UML for Business, Software and Systems. [En línea] http://www.sparxsystems.com.au/.

[7] Sun Microsystems. MySQL - The world's most popular open source database. [En línea] http://www.mysql.com/.

[8] SQLite. SQLite - Small. Fast. Reliable. [En línea] http://www.sqlite.org/.

[9] SQLiteJDBC. SQLiteJDBC. [En línea] www.zentus.com/sqlitejdbc/.

[10] Osenxpsuite. SQLite2009 Pro Enterprise Manager. [En línea] http://link.osenxpsuite.net/?uid=homepage&id=sqlite2009pro.zip.

[11] Apache Software Foundation. Logging Services. [En línea] 1999-2007. http://logging.apache.org/log4j/1.2/index.html.

[12] JFree.org. JFreeChart. [En línea] http://www.jfree.org/jfreechart/.

[13] Serena. OpenProj is a free, open source project management solution. [En línea] http://openproj.org/openproj.

[14] Sun Microsystems. OpenOffice - the free and open productivity suite. [En línea] http://www.openoffice.org/.

[15] Microsoft Corporation. Microsoft Office Visio 2007. [En línea] http://office.microsoft.com/es-es/visio/FX100487863082.aspx.

[16] Dr. Bert Scalzo, Claudia Fernandez, Donald K. Burleson, Mike Ault, Kevin Kline. Database Benchmarking: Practical Methods for Oracle & SQL Server. s.l. : Rampant TechPress, 2007. ISBN 0977671534, 9780977671533.

[17] Darwin, Ian F. Java cookbook. s.l. : O'Reilly, 2001. ISBN 0596001703, 9780596001704.

NOMBRE JAMILENA DAZA JOSE ANTONIO - NIF 26043481Y

Firmado digitalmente por NOMBRE JAMILENA DAZA JOSE ANTONIO - NIF 26043481Y
Nombre de reconocimiento (DN): c=ES, o=FNMT, ou=FNMT Clase 2 CA, ou=500742697, cn=NOMBRE JAMILENA DAZA JOSE ANTONIO - NIF 26043481Y
Fecha: 2009.07.19 13:34:05 +02'00'