PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DESARROLLO DE UN SIMULADOR DE AMBIENTE DE TERMINALES DE PUNTOS DE VENTA (POS) PARA REALIZAR PRUEBAS DE ESFUERZO DE MOTORES TRANSACCIONALES

Tesis para optar por el **Título de Ingeniero Informático**, que presentan los bachilleres:

Cecilia Roxana León Polo
Juan Alberto Cisneros Tipe

ASESOR: Corrado Daly Scaletti

Lima, Febrero del 2016

TABLA DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN DE TESIS

La presente tesis tiene como objetivo desarrollar una aplicación que simule un ambiente de terminales de punto de venta (POS) que permita simular pruebas de esfuerzo sobre motores transaccionales para evitar que los defectos de rendimiento impacten la operativa normal de un entorno de producción. Es imprescindible garantizar la estabilidad del motor enrutador transaccional en situaciones en donde aumente la demanda de transacciones a raíz de una proyección en la red de terminales de puntos de venta dado que a través del tiempo se aperturan nuevos canales de venta y se expanden los establecimientos.

Al presentarse algún defecto (indisponibilidad, degradación) en el motor enrutador transaccional se puede comprometer de forma completa toda la red de terminales, imposibilitando la realización de transacciones. Esto a su vez puede generar caos en los establecimientos donde por ejemplo una cadena de tiendas no pueda realizar ventas y por consiguiente presentar pérdidas económicas en todos sus establecimientos. Estos defectos se deben a que los procesos de validación de software en entornos de prueba no cubren escenarios de alta concurrencia sobre el motor enrutador transaccional y como consecuencia no se mitiga el riesgo de incidencias en producción.

Según lo expuesto estamos frente a un escenario que genera problemas operativos en los establecimientos de venta, problemas que comprometen la relación entre proveedor tecnológico y cliente comercial, y en consecuencia afectan al cliente final quien no puede realizar una compra o pago de servicio. Ante este escenario, surge la necesidad de implementar una estrategia basada en una aplicación tecnológica que permita llevar a cabo los procesos de validación de software enfocado en pruebas de esfuerzo para mitigar el riesgo operativo.

Finalmente, el proyecto de Tesis propone una solución basada en una herramienta informática para la generación de escenarios de pruebas de esfuerzo en donde exija un alto rendimiento al motor enrutador transaccional de tal forma que las pruebas de robustez y alta disponibilidad estén cubiertas en los procesos de control de calidad de software.

1 Problemática

Las operaciones transaccionales se encuentran presentes en diversos sectores de exigente demanda tales como Banca, Retail, Servicios; en donde es imprescindible la alta disponibilidad de la plataforma tecnológica para que brinde el soporte a las operaciones transaccionales desde cada uno de los canales y en todos los establecimientos disponibles. Uno de los componentes más importantes de la plataforma tecnológica es el motor enrutador transaccional dado que concentra todas las transacciones que se generan en la red de terminales y contiene la lógica para recepcionar, interpretar y enrutar la transacción al autorizador correspondiente para luego devolver la respuesta al terminal origen.

La existencia de algún defecto de rendimiento en el motor enrutador transaccional puede comprometer de forma completa toda la red de terminales imposibilitando la realización de una transacción. Esto a su vez se traduce en un caos en los establecimientos donde por ejemplo una cadena de tiendas no pueda realizar ventas trayendo como consecuencia pérdidas económicas en todos sus establecimientos.

Los defectos de rendimiento se deben a que los procesos de validación de software no cubren pruebas de esfuerzo altamente efectivas sobre el motor enrutador transaccional y como consecuencia se incurre en riesgos de incidentes potenciales que puedan presentarse en producción. El motor enrutador transaccional constantemente recibe actualizaciones que implementan nuevas funcionalidades y que no deben impactar el rendimiento de la aplicación. Según la operativa, en fechas festivas u horarios específicos se presenta mayor demanda transaccional, en tal sentido, si estos escenarios pudieran ser simulados entonces los defectos podrían ser identificados.

A través de métodos manuales, realizar pruebas de esfuerzo implica disponer de un grupo considerable (al menos cien) de personal capacitado (Testers) y equipos POS, de tal forma que en un tiempo considerable y de manera coordinada, simultánea se ejecute transacciones.

En esta situación se identifica inconvenientes que pueden catalogar como no factible la realización de la prueba dado que disponer de personal capacitado y en un número considerable no es tarea fácil porque implica disponer de costos para la capacitación, la

labor de tester, la adquisición de equipos POS y toda la infraestructura que se requiere para llevar a cabo las pruebas de esfuerzo de forma iterativa (hasta lograr los resultados satisfactorios del plan de pruebas). Sin embargo, esta forma de prueba no garantiza la concurrencia que realmente se requiere comprobar dado que al no ser automatizado, puede presentarse descoordinaciones que afecten el objetivo. Además, es posible que se requiera probar la estabilidad del motor enrutador transaccional durante varios días y en este escenario sería poco viable optar por la forma manual.

Como parte del mantenimiento del motor enrutador transaccional, se implementan optimizaciones para mejorar el rendimiento que deben ser validadas antes de actualizarse en producción. En ese sentido la validación implica someter el motor enrutador transaccional a pruebas de esfuerzo con una alta demanda de transacciones y por un tiempo sostenible, todo ello con el fin de conocer a priori los resultados de la optimización.

Existen una serie de transacciones no financieras que se generan en los terminales y que en su conjunto pueden impactar el rendimiento del motor enrutador transaccional. Estas transacciones comprenden lo siguiente:

- “Echos” que envía los terminales cada cierto tiempo configurable para reportar que se encuentran operativos y disponibles a pesar de que no pudieran estar realizando transacciones financieras.
- Alarmas como por ejemplo la notificación de reposición de rollo de contómetro (voucher); esta alarma es reportada a través de una transacción no financiera.
- Estadístico el cual es enviado en dos momentos distintos: uno inmediatamente posterior a una transacción financiera y otro al final del día como un consolidado de las operaciones en el terminal.
- Actualizaciones remotas de la aplicación que reside en los terminales. Estas actualizaciones se realizan a través de transacciones de tipo “descarga” que por lo general conllevan a ejecutar varias transacciones (dependiendo del volumen de archivos) para completar la actualización de un solo terminal. Entonces la actualización de toda una red de terminales implica una cantidad considerable de transacciones y sumada a las operaciones financieras genera una gran carga al motor enrutador transaccional.

Según lo expuesto, se presenta un escenario que genera problemas operativos en los establecimientos de venta, problemas que comprometen la relación entre proveedor

tecnológico y cliente comercial, y en consecuencia afectan al cliente final quien no puede realizar una compra o pago de servicio. Ante este escenario, surge la necesidad de implementar una estrategia basada en una aplicación tecnológica que permita llevar a cabo los procesos de validación de software enfocado en pruebas de esfuerzo para mitigar el riesgo operativo.

Adicionalmente se requiere conocer la estabilidad del motor enrutador transaccional en situaciones en donde aumente la demanda de transacciones como resultado de un estudio de proyección de la red de terminales que a través del tiempo apertura nuevos canales de transacciones y/o incremento de terminales.

Por otro lado, como un aspecto comercial, al contar con una aplicación de simulación de terminales podría realizarse demostraciones a futuros clientes sobre la capacidad del motor enrutador transaccional simulando con los parámetros que manejaría el cliente todo ello como parte de una estrategia de venta.

En tal sentido a lo expuesto, el proyecto de Tesis propone una solución basada en una herramienta informática para la generación de escenarios de esfuerzo en donde exija un alto rendimiento al motor enrutador transaccional de tal forma que las pruebas de robustez y alta disponibilidad estén cubiertas en los procesos de control de calidad de software. Y en base a los resultados de ejecución de dichas pruebas se tomen las medidas necesarias para cumplir con los requerimientos de la aplicación.
Dicha solución informática debe presentar las siguientes características:

- Peticiones vía TCP/IP.
- Lógica desde el lado cliente.
- Concurrencia.
- Reporte de ejecución.
- Reporte de tiempos de respuesta.

En las secciones posteriores del documento, se extenderá a detalle los aspectos teóricos, conceptuales y la forma en que se abordará el proyecto de Tesis.

2 Marco teórico

Los siguientes conceptos son los que se referencian a lo largo del presente documento, haciendo énfasis en los más importantes para brindar un panorama claro sobre la información del Proyecto Tesis

2.1 Marco conceptual

Pruebas de rendimiento

Son pruebas que permiten analizar el comportamiento del sistema es decir, velocidad, escalabilidad y/o estabilidad cuando se ve sometido a una carga de manera concurrente. [MICROSOFT, 2007]

Los resultados de las pruebas de rendimiento dependen en buena medida del entorno de prueba siendo fundamental que este último sea una réplica del ambiente de producción. La configuración del entorno de pruebas de rendimiento es una labor muy difícil ya que los resultados pueden verse afectados significativamente si existieran diferencias mínimas entre la prueba y el entorno de producción. [MICROSOFT, 2007]

Pruebas de esfuerzo (stress)

Este tipo de prueba se realiza para determinar o validar el comportamiento de la aplicación en los momentos de carga superior a las condiciones normales que ayudan a los administradores a determinar si la aplicación conserva su rendimiento en caso de que la carga real supere a la carga esperada. [MICROSOFT, 2007]

Entorno de Producción

Es también llamado el entorno de la aplicación en vivo. Este es un conjunto de recursos que ofrece servicios en vivo para los usuarios de aplicaciones reales. En resumen, el entorno en el que los usuarios interactúan en vivo se llama producción / entorno real. [MICROSOFT, 2007]

Motor Enrutador Transaccional

Es el encargado de gestionar y enrutar transacciones que se transmiten entre entidades y un conjunto de dispositivos electrónicos que funcionan como canal de atención o interacción con los clientes.

Terminal

Dispositivo tecnológico que cuenta una interfaz de usuario para ejecutar una transacción de negocio afiliado a un comercio. [GONZALES, 2006]

Transacción

Es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica**.** [MC DEMID, 1992]

Anulación de una Transacción

Es el proceso por el cual se cancela la transacción, y se empieza a deshacer las órdenes ejecutadas hasta dejar todo en su estado inicial, como si la orden de la transacción nunca se hubiese realizado. [MC DEMID, 1992]

Requerimiento no funcional de rendimiento de software

Un software implementa un conjunto de requerimientos funcionales según la necesidad de negocio para lo que se disponga. Sin embargo, existen requerimientos de carácter no funcional que dependiendo el contexto en donde se aplique es necesario evidenciarlo para el éxito completo del software en producción; dicho requerimiento corresponde a los niveles de rendimiento que debe presentar frente a escenarios de mayor demanda de tal forma que el desempeño del software no resulte degradado y no genere pérdidas económicas. [SOMMERVILLE, 2006]

2.1.1 Conceptos relacionados a la propuesta de solución

Programación MultiHilos

El concepto de MultiHilos consiste en una serie de programas cada una con un camino de ejecución independiente y simultánea, utilizando al máximo el CPU y reduciendo al mínimo el tiempo en desuso. [OAKS,2004]

Este concepto se aplica en el desarrollo de la aplicación de ambiente de terminales, donde el comportamiento de un terminal se representa por la ejecución de un hilo y cada terminal forma parte de un contexto de comercio con horarios de ejecución de mayor exigencia respecto a otros comercios, en ese sentido, un comercio se representa por la ejecución de un grupo de terminales, es decir, un grupo de hilos que se implementa bajo el concepto MultiHilos.

Concurrencia

Este concepto representa un diseño de programas con la habilidad de acceso a datos de forma simultánea entre dos o más hilos. El diseño considera aspectos de sincronización de datos para evitar problemas de concurrencia. [OAKS,2004]

En el desarrollo de la aplicación simulador de ambiente de terminales, el concepto de concurrencia se aplica por ejemplo: en los recursos compartidos como el "pool" de conexiones con el motor enrutador transaccional, contadores de ejecución de transacciones por tipo y aplicación.

La concurrencia es un término bastante amplio, incluye la habilidad de desarrollar varias tareas al mismo tiempo. Por lo general, esta habilidad se conoce como paralelismo.

La programación multihilos es algo más que el paralelismo: se refiere también al diseño de programas más sencillos aprovechando ciertas características de la implementación. [OAKS,2004]

Multitarea basada en hilos

Con la multitarea basada en hilos, el hilo es la unidad de código más pequeña. Un programa puede realizar dos o más tareas de manera simultánea, como guardar un archivo mientras se está editando.

La multitarea basada en hilos genera una sobrecarga menor que la basada en procesos. Los procesos son tareas que requieren una mayor cantidad de recursos del CPU, además que la comunicación entre ellos suele ser limitada. [OAKS,2004]

La multitarea basada en hilos permite escribir programas muy eficientes que hacen una utilización óptima del procesador, minimizando el tiempo libre que éste tiene. En la Figura 1, se muestra el diagrama del proceso de simulación de terminales para el proyecto de tesis.

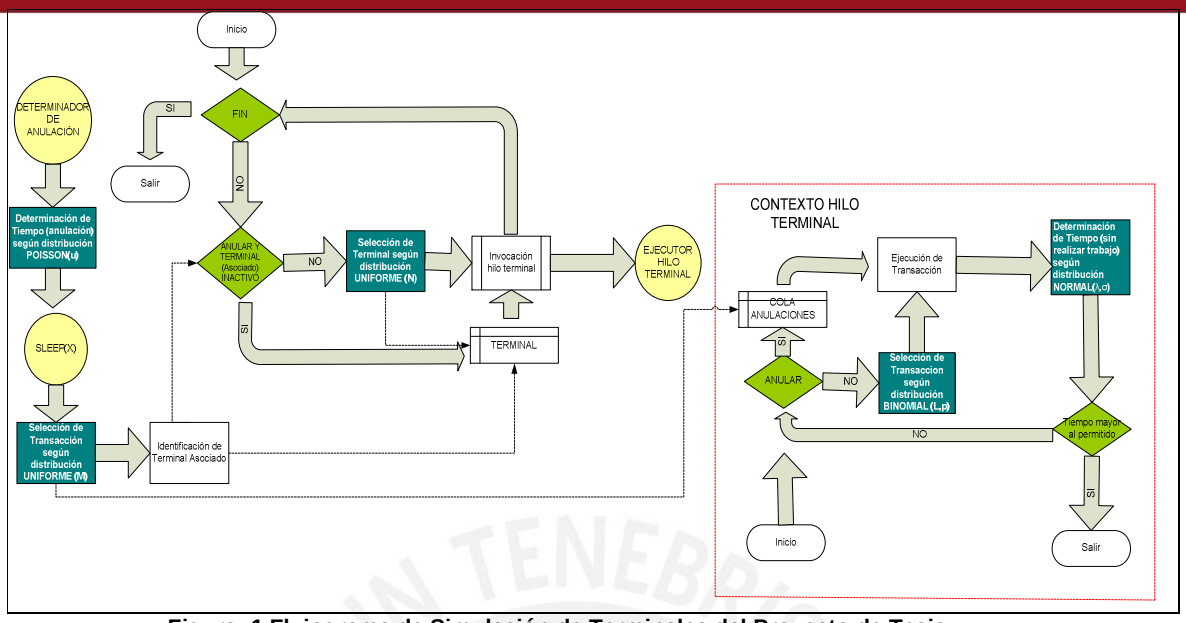


Figura: 1 Flujograma de Simulación de Terminales del Proyecto de Tesis

Modelos de distribución probabilístico

Modelo para simular los eventos transaccionales y con ello aproximar las operaciones que se generan en producción.

Distribución de Poisson

Es una distribución con variable discreta sus principales aplicaciones hacen referencia a la modelización de situaciones en las que interesa determinar el número de hechos de cierto tipo que se pueden producir en un intervalo de tiempo o de espacio, bajo presupuestos de aleatoriedad y ciertas circunstancias restrictivas. [HERNANDEZ-RAMOS-VELEZ, 2011]

El parámetro de la distribución es, en principio, el factor de proporcionalidad para la probabilidad de un hecho en un intervalo infinitésimo.

Se trata de un modelo discreto donde el campo de variación de la variable será el conjunto de los número naturales, incluido el cero: x € [0,1,2,3,4….]

Para la solución propuesta se utilizará la distribución de Poisson para determinar cada cuánto tiempo va a realizarse una transacción de tipo anulación.

Distribución Normal

Se llama distribución normal, distribución de Gauss o distribución Gaussiana, a una de las distribuciones de probabilidad de variable continua que con más frecuencia aparece aproximada en fenómenos reales.

Esta distribución permite modelar numerosos fenómenos naturales, sociales y psicológicos. Mientras que los mecanismos que subyacen a gran parte de este tipo de fenómenos son desconocidos, por la enorme cantidad de variables incontrolables que en ellos intervienen, el uso del modelo normal puede justificarse asumiendo que cada observación se obtiene como la suma de unas pocas causas independientes. [HERNANDEZ-RAMOS-VELEZ, 2011]

En la solución propuesta se utilizará la distribución normal para determinar la continuidad de la actividad de los terminales.

Distribución Uniforme (Discreta)

Es una distribución de probabilidad que asume un número finito de valores con la misma probabilidad donde todos los elementos de un conjunto finito son equiprobables. Esta distribución se utilizará para elegir aleatoriamente el tipo de transacción que se va a anular. [HERNANDEZ-RAMOS-VELEZ, 2011]

Distribución Binomial

Es una distribución de probabilidad discreta que mide el número de éxitos en una secuencia de “n” ensayos de Bernoulli independientes entre sí, con una probabilidad fija “p” de ocurrencia del éxito entre los ensayos. Un experimento de Bernoulli se caracteriza por ser dicotómico, esto es, sólo son posibles dos resultados. A uno de estos se denomina éxito y tiene una probabilidad de ocurrencia p y al otro, fracaso, con una probabilidad q = 1 - p. En la distribución binomial el anterior experimento se repite n veces, de forma independiente, y se trata de calcular la probabilidad de un determinado número de éxitos. Para n = 1, la binomial se convierte, de hecho, en una distribución de Bernoulli. [HERNANDEZ-RAMOS-VELEZ, 2011]

La distribución Binomial se utilizará para simular el tipo de transacción que se ejecutará según sus probabilidades asignadas inicialmente en base a información histórica.

3 Estado del arte

La tendencia actual es que las compras/pagos de productos y servicios se realicen sobre plataformas transaccionales, donde el motor es el componente elemental y debe presentar arquitecturas que garanticen la alta disponibilidad frente a volúmenes altos de transacción. En ese sentido existen mecanismos que permiten efectuar pruebas de rendimiento sobre aplicaciones con el objetivo de reducir el riesgo operativo; sin embargo estos mecanismos deben ser claramente analizados si cumplen con todas las características que se requiere para someter a pruebas de esfuerzo al motor enrutador transaccional.

Uno de los análisis que suelen integrar cualquier plan de control de calidad son las **pruebas de stress**. Esta evaluación pone a prueba la robustez y la confiabilidad del software sometiéndolo a condiciones de uso extremas. Entre estas condiciones se incluyen el envío excesivo de peticiones y la ejecución en condiciones de hardware limitadas. El objetivo es saturar el programa hasta un punto de quiebre donde aparezcan bugs (defectos) potencialmente peligrosos. [MICROSOFT, 2007]

Los efectos de la saturación pueden ser la pérdida o adulteración de datos, el uso excesivo de recursos incluso una vez finalizada la situación de stress, un mal funcionamiento de componentes de la aplicación o la aparición de errores inesperados.

Un buen plan de pruebas de stress debe contemplar el desarrollo de no uno, sino varios casos de stress. Cada caso diferirá en el volumen del estímulo a aplicar sobre la aplicación (cantidad de peticiones), el tiempo que durará cada estímulo y la duración total del experimento, entre otras variables. Además, deberá contar con una serie de resultados esperados. Todos los casos deben ponerse en práctica, registrándose al término de cada uno estadísticas sobre el uso de CPU, memoria, conexión y otros recursos. Al finalizar, se comparan los resultados obtenidos con los esperados y se obtienen conclusiones sobre el rendimiento de la aplicación. Si se encontraron problemas, es necesario revisar el diseño o el código de la aplicación para descubrir el origen del conflicto.

La importancia en la identificación de defectos de rendimiento radica en la mitigación del riesgo operativo, es decir, anteponerse a escenarios de alta carga de transacciones para asegurar que la calidad del rendimiento del motor transaccional no se degrade y cumpla

niveles de respuesta aceptables. De esta manera se mitiga el riesgo de impactos por defectos de rendimiento en la puesta a producción y se brinda un software de mayor calidad al cliente. De esta manera se le entrega al cliente un software que puede no ser el definitivo, pero sí goza de la robustez adecuada para su uso diario.

3.1 Formas aproximadas de resolver el problema

- Utilizar un software comercial y ejecutar pruebas de esfuerzo con las limitaciones de flujos transaccionales y configuración de carga.
- Optar por el método manual organizando un equipo de cien o más usuarios para realizar transacciones de forma coordinada, simultánea en un tiempo de veinticuatro (24) horas continuas.

3.2 Productos comerciales para resolver el problema

HP LOAD RUNNER: Herramienta para pruebas de rendimiento de software el cual se basa en la generación de diversos usuarios virtuales con cargas repetibles. Identifica los cuellos de botella del software. Se compone de cuatro herramientas; el Virtual UserGenerator (Vugen), captura los procesos de negocio de los usuarios finales y crea un script automatizado. El Controller, organiza, conduce, administra y supervisa la prueba de carga. Los Load Generators (Generadores de Carga), generan la carga ejecutando los Vusers. Y por último la herramienta de Analysis, que ayuda a ver, analizar y comparar los resultados. [LOADRUNNER, 2013]

NEOLOAD: Herramienta desarrollada por la compañía francesa Netosys escrito en JAVA, permite realizar pruebas de carga y rendimiento de software. Esta herramienta es compatible con sistemas operativos como Windows, Linux y Solaris.
Mediante una simulación real de la actividad del usuario, permite analizar el comportamiento del servidor, llegar a conocer la capacidad de la aplicación y la cantidad de usuarios que puede manejar al mismo tiempo así como también la localización de los cuellos de botella.
Soporta pruebas de rendimiento en cualquier aplicación de Internet utilizando AJAX, HTML5, SPDY, JSON, Flex, GWT, Oracle Forms, Silverlight, RTMP, tecnologías push, serialización de Java, .NET, J2EE, SOAP, SAP, Siebel entre otras. Adicionalmente brinda soporte en pruebas de rendimiento completo de aplicaciones móviles. [NEOLOAD, 2015]

WEBLOAD: Permite realizar pruebas de carga y de estrés en cualquier aplicación de Internet utilizando Ajax, Adobe Flex, .NET, Oracle Forms, HTML5 y muchas más tecnologías. Puede generar la carga de las máquinas en la nube y en las instalaciones. Los puntos fuertes de WebLOAD son su facilidad de uso con características como la grabación / reproducción basada en DOM, correlación automática y uso de JavaScript. La herramienta es compatible con las pruebas de rendimiento a gran escala con carga de usuarios pesados y complejos escenarios, y proporciona un análisis claro sobre la funcionalidad y el rendimiento de la aplicación web. [WEBLOAD, 2015]

3.3 Productos no comerciales (de investigación) para resolver el problema

JMETER: Es una herramienta para realizar pruebas de esfuerzo en donde genera múltiples peticiones que son enviadas a una aplicación destino. JMeter es un proyecto de Apache que puede ser usado como una herramienta de pruebas unitarias para conexiones de bases de datos con JDBC, FTP, LDAP, Servicios web, JMS, HTTP y conexiones TCP genéricas. JMeter puede también ser configurado como un monitor, aunque es comúnmente considerado una solución ad-hoc respecto de soluciones avanzadas de monitoreo. Soporta aserciones para asegurarse que los datos recibidos son correctos, por cookies de hilos, configuración de variables y una variedad de reportes. [JMETER, 2013]

LOADUI: Herramienta de código abierto utilizada para pruebas de carga y para medir el rendimiento de las aplicaciones web, es flexible e interactiva permitiendo crear, configurar y actualizar el set de pruebas mientras la aplicación se está probando.

El análisis avanzado y generación de informes le permite examinar el rendimiento real mediante el bombeo de nuevos datos incluso mientras la aplicación se está probando. No es necesario reiniciar el LoadUI cada vez que se modifique o cambie la aplicación, la interfaz se actualiza automáticamente. [LOADUI, 2015]

A continuación se muestra un cuadro (Tabla 1) comparativo de las soluciones que existen en el mercado haciendo énfasis en las características más importantes que se requiere para alcanzar los objetivos indicados en el punto 4.

Característica	JMETER	LOADUI	HP LOAD RUNNER	WEBLOAD	NEOLOAD
Peticiones vía TCP/IP.	✓	✓	✗	✓	✓
Lógica desde el lado cliente.	✗	✗	✗	✗	✗
Concurrencia.	✓	✓	✓	✓	✓
Reporte de ejecución.	✓	✓	✓	✓	✓
Reporte personalizado TPS.	✗	✗	✗	✗	✗

Tabla 1: Evaluación de soluciones existentes en el mercado

3.4 Conclusiones sobre el estado del arte

Las herramientas existentes en el mercado que permiten realizar pruebas de esfuerzo no permiten simular dependencias entre transacciones que ayudan a simular la lógica del negocio; por consiguiente se justifica desarrollar una nueva herramienta que maneje las características principales y personalizadas acorde al motor enrutador transaccional para garantizar con éxito el proceso de validación de software con el objetivo principal de mitigar el riesgo operativo.

4 Objetivo general

El objetivo del proyecto es desarrollar una aplicación que simule un ambiente de terminales de punto de venta (POS) que permita simular pruebas de esfuerzo sobre motores enrutadores transaccionales para evitar que los defectos de rendimiento impacten la operativa normal de un entorno de producción.

5 Objetivos específicos

- Implementar un prototipo de aplicación simulador que permita comprobar que las transacciones no financieras (echo, alarma, estadístico, download) presenten un tiempo de respuesta óptimo.

- Implantar aplicación simulador que permita realizar pruebas de esfuerzo en motores transaccionales en un entorno de producción.
- Simular las ejecuciones del motor enrutador transaccional a través de una prueba de esfuerzo, a fin de conocer a priori si las peticiones presentan tiempos de respuesta óptimos en horarios de mayor demanda.
- Calibrar los niveles de confiabilidad de un motor enrutador transaccional a través de las pruebas de esfuerzo ejecutadas en el simulador.

6 Resultados esperados

Objetivo	Resultado evidencia
Implementar un prototipo de aplicación simulador que permita comprobar que las transacciones no financieras (echo, alarma, estadístico, descargas) presenten un tiempo de respuesta óptimo.	Prototipo aplicación simulador con generador de reporte de tiempos de respuesta por tipo de transacción implementado.
Implantar aplicación simulador que permita realizar pruebas de esfuerzo en motores transaccionales en un entorno de producción.	Prototipo aplicación simulador implantado en un entorno de producción.
Simular las ejecuciones del motor enrutador transaccional a través de una prueba de esfuerzo, a fin de conocer a priori si las peticiones presentan tiempos de respuesta óptimos en horarios de mayor demanda.	Reportes de tiempos de respuesta por intervalo de horas.
Calibrar los niveles de confiabilidad de un motor enrutador transaccional a través de las pruebas de esfuerzo ejecutadas en el simulador.	Reportes de tiempos de respuesta con diferentes niveles de confiabilidad.

Tabla 2: Objetivos versus Resultado de la Evidencia

7 Herramientas, métodos y procedimientos

En esta sección se describe las herramientas y métodos seleccionados para el desarrollo del proyecto Tesis:

- Agile Unified Process (AUP)
- API STJ.Mgmt.
- Redmine
- Inspección a través de reportes, archivos traces, consultas a base de datos.

7.1 Mapeo

A continuación se muestra un cuadro (Tabla 3) que indica el uso de herramientas y métodos con sus respectivos resultados que se espera obtener a través del desarrollo del proyecto de Tesis.

Resultados esperado	Herramientas a usarse
RE1: Documentación del Simulador de Terminales. Aplicación Simulador de Terminales.	- Enfoque AUP para todas las etapas del proyecto. - API STJ.Mgmt
RE2: Reporte de tiempos de respuesta.	- Inspección a través de reportes, archivos traces, consultas a base de datos. - Redmine.
RE3: Reporte de ejecución de transacciones.	- Inspección a través de reportes, archivos traces, consultas a base de datos. - Redmine.

Tabla 3. Resultados esperados versus Herramientas y Métodos

A continuación se presentan las herramientas principales que se requieren.

7.2 SJT.Mgmt

Es una librería muy fácil de utilizar para la administración de hilos en las aplicaciones Java. La gestión de hilos permite trabajar con múltiples hilos sin el costo de funcionamiento de la memoria y el esfuerzo de CPU. Los parámetros incluyen, tamaño máximo inicial, y el límite de carga de trabajo. [SIMPLE THREAD, 2013]

Esta herramienta ayudará a la gestión de los hilos que son necesarios manejar en la solución con la finalidad de que cada hilo simule la función de un terminal. En la Figura 2 se muestra el diagrama de gestión de hilos que maneja el API STJ para el proyecto de tesis.

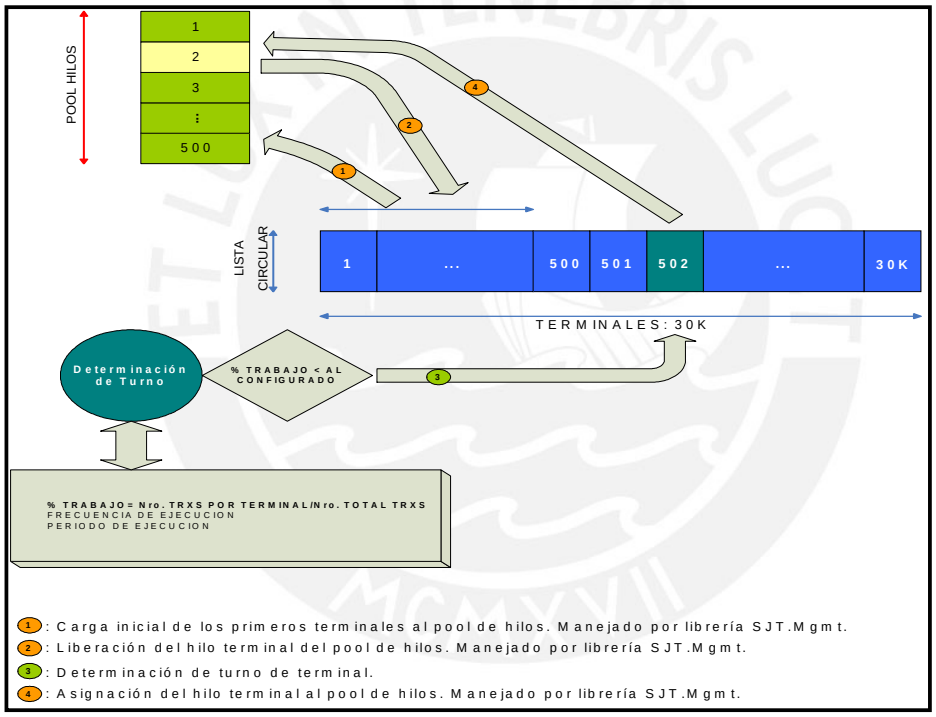


Figura: 2

Diagrama de Gestión de Hilos utilizando API Sjtm para el Proyecto de Tesis

7.3 Redmine

Es un gestor de proyecto que permitirá administrar avances del proyecto, casos de prueba, errores, wiki, calendario, etc. Adicionalmente permite tener integración con diversos sistemas de control de versiones como SVN, CVS, Git, entre otros.

Una de las ventajas más importantes de la utilización de un gestor de proyectos es la organización, que permite administrar el proyecto de forma más eficiente. Entre las tareas más importantes realizadas con Redmine, se encuentran crear la lista de funcionalidades que se tiene que implementar en el proyecto, llevar un control continuo de los errores que surgen, asignándole prioridad y recursos. Adicionalmente maneja un calendario, que permite crear diagramas de Gantt, un repositorio con acceso a la documentación y la trazabilidad de casos de uso versus casos de prueba. [READMINE, 2014]

7.4 Metodología

Es de vital importancia contar con un enfoque de desarrollo que permita cumplir con los objetivos planteados y de esta manera al finalizar el proyecto se tenga un producto de excelente calidad; cumpliendo con los requerimientos, planificación y presupuesto establecido.

A continuación se presentan las dos metodologías que fueron evaluadas para ser utilizadas en el presente proyecto, y como conclusión se sustenta la elección de una de ellas.

7.4.1 Rational Unified Process (RUP)

RUP es una metodología de desarrollo de software basada en un enfoque iterativo e incremental centrado en la arquitectura y guiado por los casos de uso, con una correcta gestión de requerimientos incorporando al diseño de software el lenguaje UML, definido como un sistema de modelamiento visual para la representación gráfica de casos de uso, clases de análisis, componentes de software entre otros. [KRUCHTEN, 2003]

Este enfoque trae como beneficios la atenuación de riesgos desde ciclos tempranos del proceso alineando las necesidades de los usuarios a las funcionalidades del producto. A su vez promueve una correcta administración del cambio y la configuración.

Esta metodología engloba una serie de entregables o artefactos del ciclo de desarrollo del producto, constituyéndose así como el activo más importante después

del producto final, pues en éstos se documentan los alcances técnicos y funcionales definitivos del producto desarrollado.

Pese a sus prestaciones, RUP enfrenta críticas debido a que prioriza el avance documentario y la elaboración de entregables como prioritarios para el software (en ciertos casos extensos y complejos en su administración) desplazando otros factores tales como la modalidad de trabajo durante la codificación del producto donde las metodologías agiles si hacen hincapié.

7.4.2 AUP (Agile Unified Process)

AUP es una metodología de desarrollo ágil heredera de otros paradigmas como la programación extrema (XP) y RUP. Esta metodología consta de principios y prácticas influyentes en la construcción del software en armonía con la documentación esencial de entregables específicos para el entendimiento de la solución. Entre sus objetivos destaca la reducción del costo del cambio en el proyecto en base a procedimientos iterativos (característica propia de RUP) donde la codificación y pruebas del software se llevan a cabo paralelamente (según XP). [AUP, 2015]

Además de la estructura metodológica fijada por RUP (como el desarrollo de producto por iteraciones y presentación de prototipos en modo incremental), AUP introduce propuestas como la gestión de requerimientos por niveles de prioridad, independencia entre herramientas para la concepción del producto y la modificación del código del programa sin alterar su comportamiento original mejorando en su estructura, performance y diseño. Asimismo propone el desarrollo dirigido por pruebas a partir de un concepto denominado unidad de prueba (sincronizando tanto la construcción como las pruebas en el prototipo) de carácter reutilizable.

7.4.3 Metodología Seleccionada

El enfoque tradicional de desarrollo de aplicaciones heredado de otras ramas de la ingeniería, está basado en un flujo secuencial, que inicia con un estudio exhaustivo del problema, que se convierte en el input para establecer un plan, para luego llevar a cabo la construcción, esta metodología es conocida como desarrollo en cascada, con las fases consecutivas de Análisis, Diseño, Codificación, Pruebas e Implementación.

Pero este enfoque no siempre es óptimo para el desarrollo de software, la programación no es exactamente comparable a otras ramas de la ingeniería como las construcciones de ingeniería civil. El desarrollo de software, es un proceso creativo, análogo a lo que en la construcción de una vivienda sería la elaboración de los planos.

A diferencia de la ingeniería civil, el desarrollo de software no tiene un proceso de construcción en el que se sigan unos planos de modo mecánico previamente establecidos por un arquitecto o ingeniero. En cada fase de desarrollo se deben tomar decisiones para dar cumplimiento a los requerimientos que se van dando en marcha.

En tal sentido para este proyecto se opta por aprovechar el enfoque AUP (Agile Unified Proccess) que describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.

A continuación se señalan las principales razones por las cuales se está optando por un enfoque AUP:

- En este proyecto tendremos que tratar con cierta incertidumbre tecnológica al tener que evaluar durante la etapa inicial de desarrollo si la simulación del ambiente de terminales puede ser implementada solo con software por ello un enfoque AUP brinda las herramientas necesarias para gestionar de manera ágil las solicitudes de control de cambio.

- Teniendo en cuenta que AUP se apoya en RUP que a su vez está basado un lenguaje para representar modelos y sistemas llamado UML (UnifiedModelingLanguage – Lenguaje para modelado unificado), el que brinda estándares de diagramas entendibles por cualquier desarrollador de software, especialmente los que tienen tendencia a la programación orientada a objetos.

Los diagramas de UML serán acoplados y se usarán solo algunos de estos que serán definidos en las fases de Concepción y Elaboración de tal manera que solo se lleven a cabo las tareas y se obtengan los artefactos (productos intermedios) necesarios para el desarrollo del proyecto.

- Otra de las razones por los cuales se opta por un enfoque AUP es debido a que es de suma importancia ver resultados que aporten valor en corto y mediano plazo para validar que se esté cumpliendo con el alcance de la solución propuesta.

- Finalmente debido a que RUP prioriza a un grado mayor la documentación se opta por un paradigma de trabajo con entregables esenciales y específicos para el entendimiento de la solución final que nos brinda el enfoque AUP.

A continuación se describen las fases de desarrollo que se utilizaron para el sistema propuesto basado en un enfoque AUP. Cada fase se concluye con un hito claramente definido, un punto en el tiempo en el cual se debe tomar una decisión crítica y, por ende, se debe haber alcanzado un objetivo clave

Concepción: Se delimitará el alcance del sistema; se elaborará el modelo del negocio comprendiendo lo que se desea elaborar; para ello se identifican todas las entidades de esta iteración a alto nivel. Esto involucra la identificación de los requerimientos para luego definir los casos de uso y realizar la especificación de los más significativos.
Al final de la fase de Concepción el primer hito: Se obtiene la Lista de Requerimientos y especificación de los casos de uso más importantes.

Los artefactos a realizarse en esta fase serán los siguientes:

- Lista de Requerimientos.
- Especificación de los Casos de Uso más importantes.

Elaboración: En esta fase se obtendrá una comprensión más detallada de los requerimientos que se vieron en la etapa de Concepción. Se valida el diseño y la aplicación de la arquitectura base. Se tienen que tomar decisiones críticas como la selección de herramientas a utilizar, la identificación de los requerimientos más críticos del proceso. Al finalizar esta fase en forma exitosa se asegura que la arquitectura y requerimientos sean lo suficientemente estables.
Los artefactos a realizarse en esta etapa son:

- Especificación de Casos de Uso

- Vista Lógica.
- Vista Física.

Construcción: En esta fase se construye todos los componentes y toda la funcionalidad del sistema a partir de las pautas definidas en los documentos de análisis y diseño.
Esta fase se dividirá en 3 iteraciones donde al finalizar cada una se validará el alcance planificado el cual puede ser ajustado durante toda esta fase.
Los artefactos a realizarse en esta etapa son:

- Prototipo completo y operativo, con todos los módulos interactuando entre sí, según el alcance de cada iteración.
- Catálogo de Pruebas que se realizarán en la siguiente etapa.

Transición: Esta etapa se refiere a la instalación final del producto dentro de su campo de aplicación real, sin embargo, en este proyecto no se llegará a este nivel real ya que para ello se necesitaría tener clientes reales los cuales publicarían su información lo que si se obtendrá es un sistema estable que soporta todos los requerimientos que en el siguiente punto se mencionan, se desarrollará en su defecto pruebas y demostraciones del prototipo construido.
Los Artefactos a obtener en esta etapa son:

- Una versión estable del producto presentado en la fase anterior.

8 Alcance

Con la finalidad de cumplir con las expectativas de los usuario del área de control de calidad y los programadores se ve la necesidad de tener una aplicación con una propuesta innovadora que permita simular lo más parecido a un ambiente real de producción y de esta manera lograr una mejor calidad de pruebas de esfuerzo (stress) en tiempo real. Lo cual permite realizar las configuraciones necesarias en base a data estadística y finalmente obtener resultados de tiempo de respuesta mediante reportes que permitan detectar problemas críticos que se puedan presentar y que conlleve a tener una mala relación entre las cadenas de comercio y el cliente.

De esta manera se genera valor agregado a los usuarios que podrán ejecutar distintos escenarios de stress en tiempos muy cortos, realizando pruebas de esfuerzo más robusta que conllevarán a un aumento en la calidad de todos los desarrollos que se realizan y en un futuro disminuyendo los costos por solución de incidencias en producción.

Siendo entre sus principales beneficios que las configuraciones permiten ingresar valores de carga (según data estadística en producción) que ayudan a tener un flujo de ejecución más cercano a la realidad , adicionalmente permitirá que exista una lógica de negocio por parte de cliente, eliminará la dependencia de HW , características que nos permitirán simular lo más parecido posible a un ambiente de producción en tiempo real obteniendo finalmente reportes de tiempo de respuesta que ayudaran a las evaluaciones pertinentes.

El Alcance que abarcará esta aplicación y el presente proyecto es el siguiente:

- Simular una red con capacidad de cien mil terminales distribuidas en cadenas comerciales.
- Permitir configurar comportamientos propios del negocio (alarmas, echos, descargas) y a través del cual enviar transacciones a un servidor Motor Enrutador Transaccional.
- Configurar las probabilidades de ocurrencia por tipo de transacción.
- Generar reportes de ejecución y tiempo de respuesta.

8.1 Acotaciones

- La aplicación manejará solo un tipo de interfaz de comunicación TCP/IP y no los otros tipos tales como: MQ, DIAL, GPRS.
- La aplicación genera transacciones hacia un solo motor enrutador transaccional.
- La aplicación tiene por objetivo generar cargas de esfuerzo para saturar al Software motor enrutador transaccional en una Red LAN. Esto aplica dado que el simulador se ejecuta en un entorno de pruebas que solo cuenta con una Red LAN.

8.2 Riesgos

Riesgo identificado	**Impacto en el proyecto**	**Medidas correctivas para mitigar**
Alta complejidad en la programación "multihilos" incremente el tiempo de construcción de la aplicación.	ALTO	Identificar un API de gestor de hilos para utilizarlo como librería en la aplicación.
La aplicación requiera una PC de características (*) costosas para simular grandes cantidades de terminales.	MEDIO	Instalar la aplicación en más de una PC de bajos recursos para cubrir la cantidad de terminales que se requiere simular.
Durante el tiempo de ejecución del proyecto de tesis, surjan herramientas en el mercado que superen las ventajas de la aplicación simulador de ambiente de terminales.	MEDIO	Definir un diseño altamente escalable que permita incorporar mejoras con el menor esfuerzo posible.
Cambios en el tema de tesis.	ALTO	Acelerar la afinación y conciliación del tema de tesis con diferentes asesores.

Tabla 4: Riesgos Identificados

(*) Procesador Intel X5450 QuadCore de 3.0 GHz, 16 GB RAM.

9 Justificación y viabilidad

9.1 Justificativa del proyecto de tesis

El objetivo del proyecto es implementar un simulador de ambiente de terminales que genere resultados de pruebas de esfuerzo y en base a los resultados obtenidos tomar las decisiones correspondientes que permitan obtener los requerimientos de rendimiento del software desarrollado.

El poder analizar los resultados de las pruebas de esfuerzo en una etapa previa a producción permite poder reducir los costos incurridos por los incidentes de operatividad que se presentan en producción en donde impacta en la realización de transacciones en todos los establecimientos afiliados a la red de terminales.
En ese sentido, a continuación, se describe la justificación del proyecto:

- La necesidad de contar con una herramienta que simule un ambiente de terminales de puntos de venta (POS) que permita realizar pruebas de esfuerzo para identificar problemas de rendimiento antes de la implementación.
- La necesidad de información que permita tomar las acciones correspondientes antes de colocar la aplicación en un ambiente productivo.
- Contar con una herramienta de fácil configuración donde se ingrese los parámetros de pruebas de esfuerzo y el tiempo de ejecución sin necesidad de un monitoreo presencial.
- Automatizar las pruebas de esfuerzo en lugar de realizarlo de forma manual dado que esto último requiere la disponibilidad de muchas personas y diversos equipos por un tiempo considerable.

9.2 Análisis de viabilidad del proyecto de tesis

En esta sección se presenta un análisis de las necesidades principales, las cuales serán cubiertas por la solución a desarrollar en el presente proyecto. Luego se presenta un análisis de la viabilidad del proyecto basado en un análisis técnico y económico del mismo.

9.2.1 Análisis de la Necesidad

Necesidad	Solución
Validar las actualizaciones de la aplicación transaccional en un ambiente de similar carga al de producción.	Monitoreo en línea de transacciones de la web operativa del motor enrutador transaccional.
Comprobar el TPS (transacciones por segundo) que resuelve el motor enrutador transaccional.	Reporte de tiempos de respuesta. Querys a la base datos operativa del motor enrutador transaccional.
Prescindir el uso de terminales físicos y usuarios disponibles para realizar pruebas de esfuerzo.	La aplicación debe simular el comportamiento de miles de terminales.
Conocer a priori el comportamiento de los servidores transaccionales, sobre todo en fechas y horarios de mayor demanda.	Reporte de tiempos de respuesta.

- **Tabla 5: Necesidad versus Solución**

Para determinar si el proyecto propuesto o alguna alternativa (entre las indicadas en la sección 3) es la solución a la problemática planteada, se consideran distintos factores como la viabilidad económica y técnica, esto último contempla funciones del producto, características de escalabilidad, rendimiento y restricciones.

9.2.2 Análisis Económico

Para la solución propuesta, los costos identificados presentan ahorros debido a la utilización de herramientas de desarrollo libres que representan costo cero. Luego a partir de los esfuerzos estimados en horas por cada fase se obtiene el costo resultante del desarrollo del proyecto. En la siguiente Tabla 6.0 se muestran los costos a considerar en el proyecto.

Presupuesto Costo x Hora (S/.40)		
Fases	**Horas**	**Soles**
ANÁLISIS	160	S/. 6,400.00
DISEÑO	120	S/. 4,800.00
CONSTRUCCIÓN	600	S/. 24,000.00
PRUEBAS	160	S/. 6,400.00
CAPACITACIÓN	40	S/. 1,600.00
TOTAL	1080	S/. 43,200.00

Tabla 6 : Etapas de Solución Propuesta Costeadas

Por otro lado, se tienen los costos operativos por incidentes generados en producción que repercute en costos de equipos POS, atención de incidencias y validación de software como parte del control de calidad.

COSTO OPERATIVO MENSUAL POR INCIDENCIAS		
Item	**Horas**	**Soles**
POS (5)		S/. 500.00
ATENCION DE INCIDENCIAS	200	S/. 10,000.00
VALIDACIÓN	320	S/. 16,000.00
TOTAL		S/. 26,500.00

Tabla 7: Costos Operativos

Teniendo como referencia los costos anteriormente mencionados, se evalúa la rentabilidad de la solución propuesta y se concluye que la inversión en este proyecto será recuperada en un aproximado de 3 meses. En ese sentido luego de la recuperación de la inversión, se tiene un ahorro anual de aproximadamente S/. 127,200.00 y con ello el sustento de la rentabilidad del proyecto.

9.2.3 Análisis Técnico

En relación a la viabilidad técnica para la solución propuesta, se indica los requerimientos de software y hardware en ámbito de desarrollo e implementación del proyecto en un entorno QA. Los requerimientos técnicos se comentan a continuación:

a) Disponibilidad de un equipo de cómputo para la ejecución del simulador de ambiente de terminales.
b) Software IDE para el desarrollo de la aplicación simulador ambiente de terminales.

c) Software JAVA como plataforma de ejecución de aplicaciones.
d) Disponibilidad de un entorno QA del motor enrutador transaccional.

El proyecto propuesto se declara técnicamente viable dado que se dispone de todos los requerimientos de Software y Hardware. La disposición de cada requerimiento se lleva a cabo de la siguiente forma:

- El requerimiento a) consiste de un equipo personal con características de Procesador Core i7 y Memora RAM 8 GB.
- El requerimiento b) consiste de un software libre como lo es el IDE Netbeans 11.0.
- El requerimiento c) consiste de un software libre JAVA 1.7 o superior.
- El requerimiento d) consiste de un entorno QA que para fines de demostración se implementa con simuladores o se adopta el mismo entorno QA de una empresa con área de desarrollo de software.

10 Plan de actividades

En la figura se muestra el cuadro de actividades para la elaboración y desarrollo del proyecto Tesis.

Nombre de tarea	Trabajo	Predecesoras
Proyecto Tesis	**848 horas**	
INCEPCIÓN	**80 horas**	
Levantamiento de Información	**40 horas**	
Lista de Requerimientos	40 horas	
Identificación de principales Casos de Uso	**40 horas**	
Especificación de Principales Casos de Uso	40 horas	4
ELABORACIÓN	**152 horas**	
Elaboración de Casos de Uso	**80 horas**	
Documento de Especificación de Casos de Uso	80 horas	6
Arquitectura de la Aplicación	**48 horas**	
Vista Física	24 horas	9
Vista Lógica	24 horas	11
Implementación de XML	**24 horas**	
Desarrollo de XMLs que utilizará el simulador	24 horas	12
CONSTRUCCIÓN	**480 horas**	
Sprint1	**160 horas**	
Desarrollar transacciones financieras.	120 horas	14
Realizar Pruebas	40 horas	17
Sprint2	**160 horas**	
Desarrollar transacciones no financieras.	120 horas	18
Realizar Pruebas	40 horas	20
Sprint3	**160 horas**	
Desarrollar Reportes	120 horas	21
Realizar Pruebas	40 horas	23
TRANSICIÓN	**80 horas**	
Documentación Actualizada	80 horas	24
GESTIÓN DEL PROYECTO	**56 horas**	
Planificación	**16 horas**	
Propuesta de Trabajo	8 horas	4
Plan de Proyecto	8 horas	29
Riesgos	**40 horas**	
Identificación dependencias técnicas	24 horas	6
Analisis de Riesgos	16 horas	32

Figura: 3 Cronograma de Proyecto de Tesis

En la figura se muestra el diagrama de Gantt resaltando la ruta crítica correspondiente al desarrollo del proyecto Tesis.

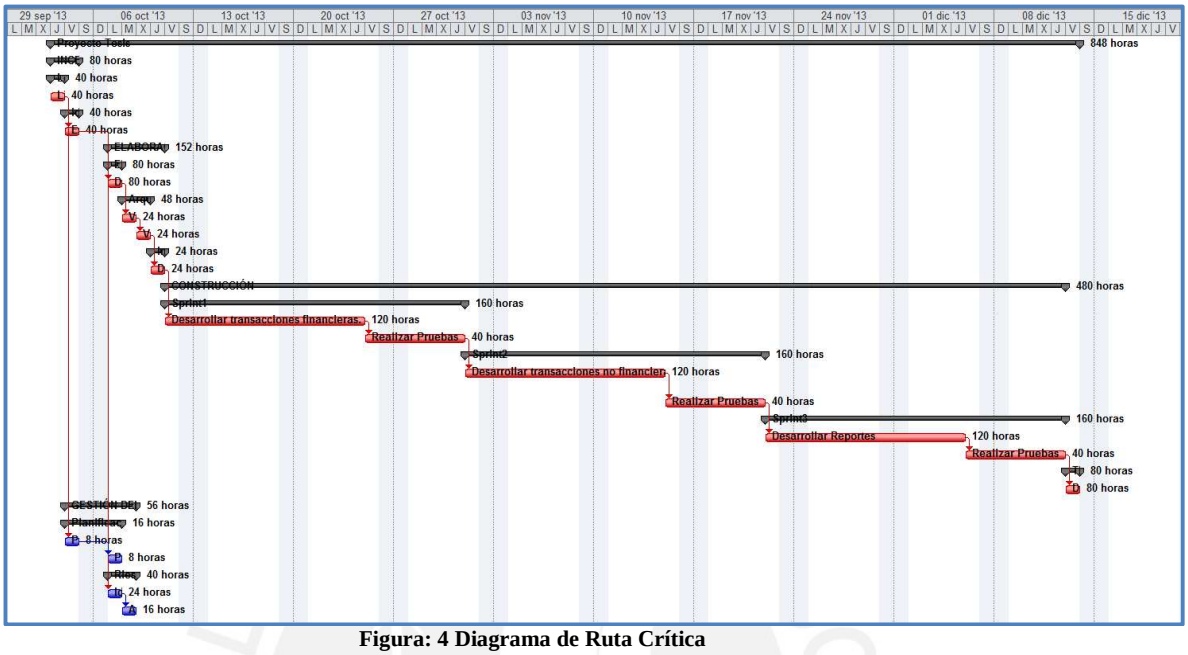


Figura: 4 Diagrama de Ruta Crítica

11 Ingeniería de Software

11.1 Identificación de Requerimientos

La gran cantidad de proyectos de software que no llegan a cumplir sus objetivos es una problemática que se vive a nivel mundial y por ello muchas veces se opta por comprar sistemas, para luego "personalizarlos" supuestamente a la medida de las empresas.

Estudios realizados muestran que una de las principales razones por la que los proyectos de software fracasan es por no realizar un estudio previo de requisitos. Otros factores como falta de participación del usuario, requerimientos incompletos y el cambio a los requerimientos, también ocupan sitiales altos en los motivos de fracasos. [STANDISH GROUP, 2013]

La identificación de requerimientos, cumple un papel primordial en el proceso de producción de software, ya que enfoca un área fundamental: la definición de lo que se desea producir, teniendo como objetivo la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema; de esta manera, se pretende minimizar los problemas relacionados al desarrollo de sistemas.

A continuación se muestra los requerimientos funcionales y no funcionales, que debe contemplar el sistema, los mismos que están organizados por módulos. Estos requerimientos serán cubiertos en diferentes diagramas y especificaciones de casos de uso.

11.1.1 Requerimientos Funcionales

REQ-1.0 Configuración de parámetros generales: Disponer de un medio de configuración de parámetros generales tales como parámetros de conexión al servidor Motor Enrutador Transaccional.

REQ-2.0 Manejo de cantidades de terminales disponibles con un máximo de cien mil: Realizar ejecuciones con cantidades reales de terminales que operan en producción. La aplicación garantiza un máximo de cien mil terminales.

REQ-3.0 Programación Fecha y Hora de Ejecución: La ejecución de transacciones se realiza en un periodo de tiempo determinado tanto en horas como en días

REQ-4.0 Generación de Grupos de Terminales: La agrupación de terminales se realiza para asignar características de ejecución a cada grupo.

REQ-5.0 Configuración de Probabilidades de Ocurrencia de Tipos de Transacciones: Asignar una probabilidad de ocurrencia de ejecución por tipo de transacción con la finalidad de simular ejecuciones donde algunos tipos de transacción presentan mayor ocurrencia respecto a otras.

REQ-6.0 Activación de Tipos de Transacciones para Realización de Pruebas: Los tipos de transacción se habilitan en Base de Datos para ser reconocidas por el Motor Enrutador Transaccional.

REQ-7.0 Ejecución de Transacciones según planificación realizada: La ejecución sigue un comportamiento definido en la planificación.

REQ-8.0 Registro de planificación en XML: Reutilizar una planificación realizada para ejecuciones posteriores.

REQ-9.0 Envíos de INICIO: El envío de INICIO es programado para realizarse todos los días. Es una de las características más importantes de un terminal dado que un envío de INICIO podría determinar la actualización de un terminal.

REQ-10.0 Manejo de Alarma: Una de las características de un terminal es la emisión de alarmas (falta de papel).

REQ-11.0 Gestión de Perfiles de Ejecución: Organizar y agrupar las características de ejecución que puede adoptar un grupo de terminal a través de un perfil de ejecución.

REQ-12.0 Manejo de descargas (download): Simular la descarga de archivos que realizan los terminales a través de los envíos de INICIO programado o identificación de una solicitud de actualización forzada.

REQ-13.0 Generación de Reporte de Ejecución de Tipos de Transacciones: Obtener un informe resumido de la ejecución para contrastar con la planificación realizada.

REQ-14.0 Generación de Reporte de Tiempos de Respuesta: Obtener un informe resumido de la ejecución para determinar el nivel de los tiempos de respuesta.

11.1.2 Requerimientos No Funcionales

Los requerimientos no funcionales han sido identificados con el objetivo de que la aplicación cumpla con las características de calidad, rendimiento y las buenas prácticas de desarrollo de software. A continuación se presentan los requerimientos no funcionales:

REQN-1 Escalabilidad: El sistema debe ser construido sobre la base de un desarrollo evolutivo e incremental, de manera tal que nuevas funcionalidades y requerimientos relacionados puedan ser incorporados afectando el código existente de la menor manera posible.

REQN-2 Flexibilidad: El sistema debe ser desarrollado con los mayores niveles de flexibilidad respecto a la parametrización de los tipos de datos.

REQN-3 Mantenibilidad: El sistema debe permitir su fácil mantenimiento con respecto a posibles modificaciones que se requieran realizar para lograr un mejor rendimiento.

REQN-4 MultiPlataforma: La aplicación debe poder instalarse en Linux o Windows, con la finalidad de que no se encuentre sujeto a un solo sistema operativo. Por lo general, el motor enrutador transaccional se encuentra instalado en un servidor Linux y la ejecución del Simulador de pruebas de esfuerzo se realiza desde un terminal Windows.

12 Arquitectura de la Solución

12.1 Arquitectura de software

Para el desarrollo de la aplicación propuesta se deben tomar en cuenta los siguientes requerimientos y necesidades fundamentales que influyen sobre la arquitectura:

- Se debe establecer un diseño que permita implementar con menor esfuerzo los cambios que se presenten.
- En vista que la información de mayor relevancia que corresponde a los resultados de las ejecuciones de transacciones no representa un volumen alto de almacenamiento, se descarta el uso de un motor de base de datos

12.1.1 Diseño de la Solución

Teniendo en cuenta las necesidades anteriormente mencionadas por cubrir, se propone como solución la organización en 3 capas, tal como se muestra en la figura 3.1, además que proporciona una base estandarizada y ordenada de la solución de software para la aplicación.

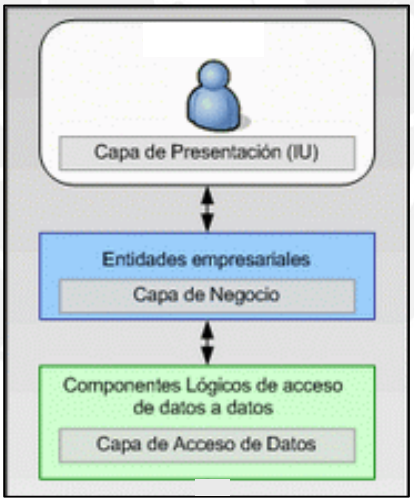


Figura: 5 Diagrama de 3 Capas

1.- Capa de presentación (GUI *GraphicalUnit Interface* - Interfaz Gráfica de Usuario): Es la que ve el usuario, también se la denomina "capa de usuario", presenta el sistema al usuario, le comunica la información y captura la información del usuario (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario.

2.- Capa de Negocio (Entidades del negocio y Entidades lógicas): Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.

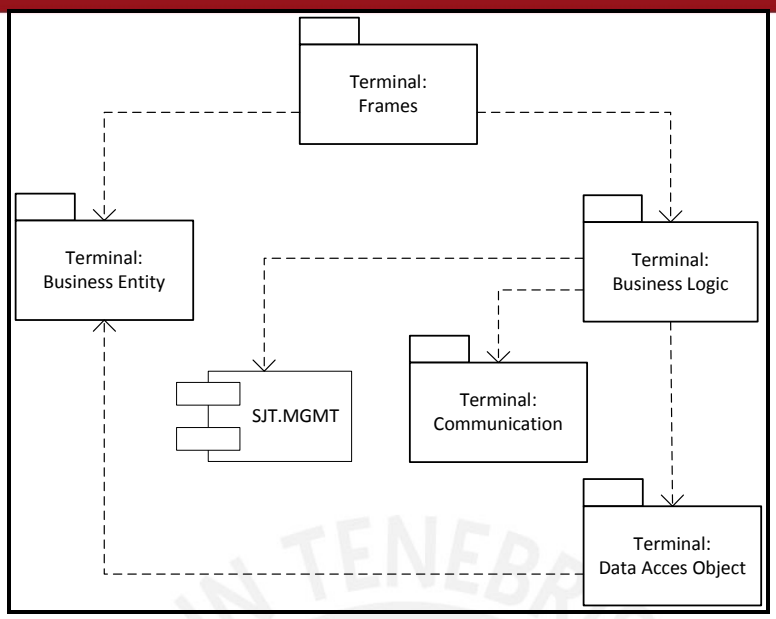
3.- Capa de datos (DAO Objeto de Acceso a Datos): Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

La arquitectura de 3 capas que se puede representar como se muestra en la figura 3.1 se propone para el proyecto pues da soporte a los requerimientos de calidad del sistema como rendimiento, portabilidad, escalabilidad, flexibilidad, y validación de la información. El resultado de una arquitectura 3 capas se indican a continuación:

- La separación de roles en tres capas, hace más fácil reemplazar o modificar una capa sin afectar a los módulos restante.
- Se tiene abstracción total acerca del origen de datos
- Bajo costo de desarrollo y mantenimiento del sistema.
- Seguridad en la información.

12.1.2 Vista Lógica

En la vista lógica se identifica los paquetes y componentes que conforman la aplicación de simulación de ambiente de terminales, de tal forma que su organización permite facilitar la inclusión de cambios posteriores.

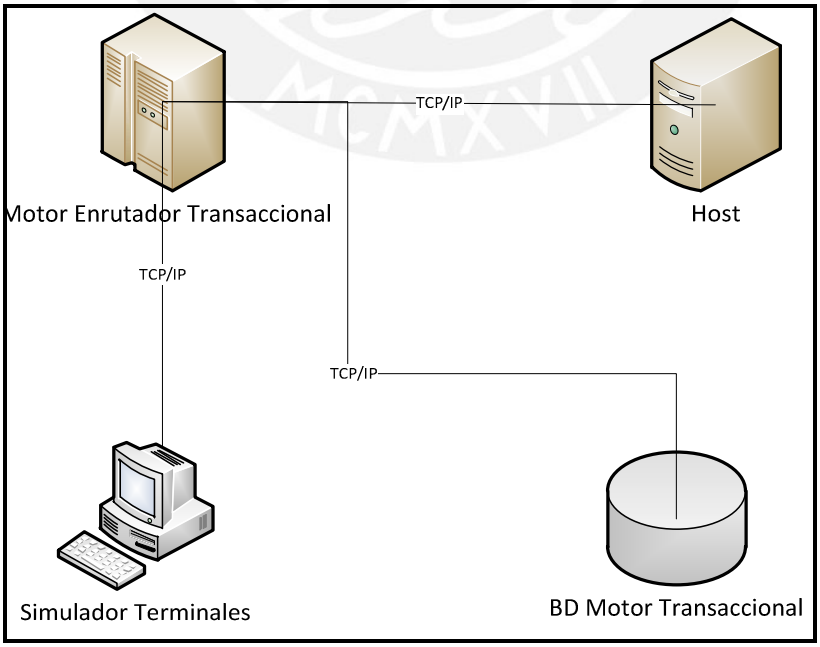


- FRAMES. Interfaz gráfica de usuario para la configuración de las pruebas y visualización de los reportes de ejecución de transacciones. Este paquete agrupa los componentes java.swing, css, html y jfreechart.
- BUSINESS LOGIC. Paquete mediadora que controla la interacción entre los frames y las capas inferiores de acceso a datos y comunicación. En este paquete se define la lógica de elaboración de cada trama que corresponde a un determinado tipo de transacción.
- BUSINESS ENTITY: Agrupa las clases entidades de negocio a disposición de los paquetes que requieran su uso. Entre las clases principales tenemos: Transacción, Terminal, Perfil de carga.
- ACCESO DATOS. Paquete en donde se captura y almacena la información de la ejecución de transacciones.
- SJT.MGMT. Componente que gestiona la creación y la liberación de los multihilos.
- COMMUNICATION. Paquete en donde se establece el enlace a través de TCP/IP con el motor enrutador transaccional. En este paquete se establece la apertura de puertos para la recepción de transacciones y la comunicación con los terminales para el retorno de las respuestas.

12.1.3 Vista de Despliegue

En la vista de despliegue se muestra los elementos que intervienen y la forma de interacción.

- Simulador de Terminales: Aplicación cliente que genera las transacciones en cargas configurables para las pruebas de esfuerzo sobre el motor enrutador transaccional. Este simulador representa el producto que genera el presente proyecto de Tesis.
- Motor Enrutador Transaccional: Es la aplicación core transaccional que recibe las peticiones, lo interpreta, traduce, enruta al host autorizador para luego obtener la respuesta y devolver al terminal cliente. Este motor enrutador es el que se somete a las pruebas de esfuerzo a través del simulador de ambiente de terminales.
- BD Motor Enrutador Transaccional: Es la base de datos del motor enrutador transaccional que almacena información de todas las operaciones que se realizan con fines de control, auditoría y conciliación con los host autorizadores. Los reportes de mediciones de tiempos de respuesta se generan a partir de la información que reside en esta base de datos.
- Host: Es el autorizador que resuelve la transacción aplicando toda la regla de negocio que compete a cada tipo de transacción.



12.1.4 Base de Datos

A continuación se muestra los XML definidos para la aplicación:

Tag Contenedor	Campo	Descripción	Ejemplo
XML Configuración General			
Motor	IP	Dirección IP del motor transaccional.	192.168.1.40
Motor	Puerto	Puerto habilitado para recepción de transacciones.	4061
Motor	Timeout	Tiempo máximo de espera de respuesta expresado en segundos.	10
XML Plan Transacción			
Aplicación	Id	Identificador de la aplicación.	CMAREQUIPA
Aplicación	Nombre	Nombre de la aplicación.	Aplicación Cajero Arequipa
Aplicación	Activo	Con este campo se habilita o deshabilita el uso de una aplicación.	1: activo, 0: inactivo.
Transacción	Id	Identificador de tipo de transacción.	1
Transacción	Nombre	Nombre del tipo de transacción.	Transacción Retiro
Transacción	Trabajo	Porcentaje de volumen de transacciones a realizar.	20
Transacción	Activo	Con este campo se habilita o deshabilita el uso de una transacción.	1: activo, 0: inactivo.

Tag Contenedor	Campo	Descripción	Ejemplo
XML Plan Terminal			
Grupo	Id	Identificador de grupo de terminales.	1
Grupo	Nombre	Nombre del grupo de terminales.	CADENAS SUPERMERCADOS ANGRY BIRDS
Grupo	Cantidad	Cantidad de terminales del grupo.	1000
Grupo	Inicio	Inicio de actividad de los terminales.	20131130100000
Grupo	Fin	Fin de actividad de los terminales.	20131130120000
Grupo	Download	Tipo de descarga.	TOTAL, PARCIAL, NINGUNO.
Grupo	Activo	Con este campo se habilita o deshabilita el uso de un grupo.	1: activo, 0: inactivo.
XML Intervalo Horario			
Intervalo	Inicio	Hora inicio intervalo.	180000
Intervalo	Fin	Hora fin intervalo.	215959
Intervalo	Transacción	Id de la transacción a configurar.	1
Intervalo	Trabajo	Porcentaje de volumen de transacciones a realizar.	60
XML Perfil de Carga			
PERFIL	ID	ID del perfil.	1
PERFIL	Descarga	Se especifica si la descarga es parcial, total o ninguno.	PARCIAL
PERFIL	Echo	Frecuencia de envío de echos expresado en segundos.	300
PERFIL	Alerta	Frecuencia de envío de alertas expresado en segundos.	18000

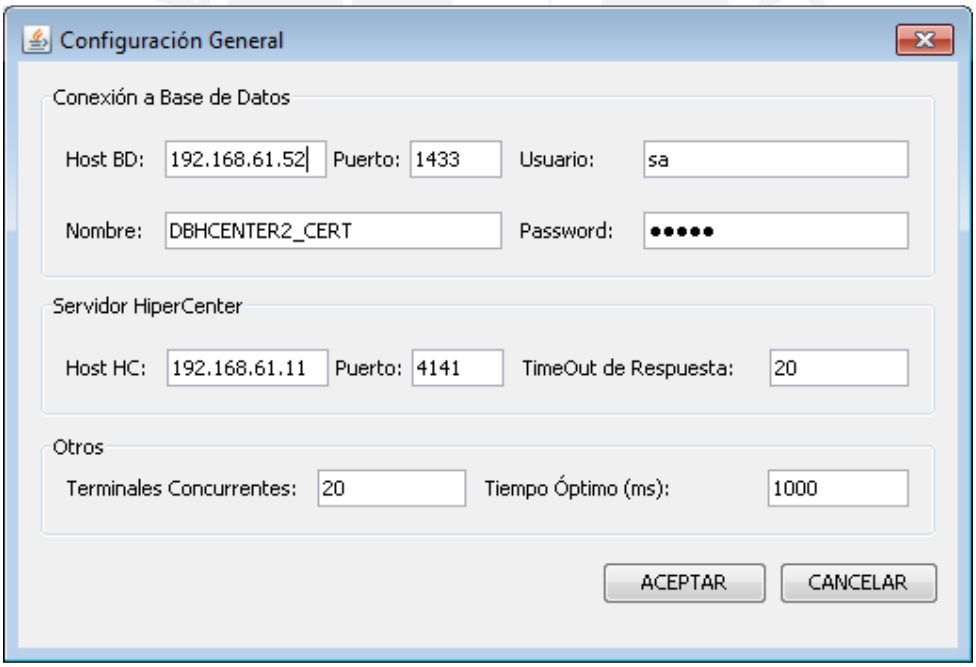
12.2 Diseño de la Interfaz Gráfica

Actualmente la labor de desarrollar un adecuado diseño de interfaz gráfica de usuario cumple un papel importante en el desarrollo de la aplicación, la interfaz va a marcar una pauta visual que ayude al usuario a la configuración dinámica de las distintas unidades necesarias para obtener reportes finales que agreguen valor.

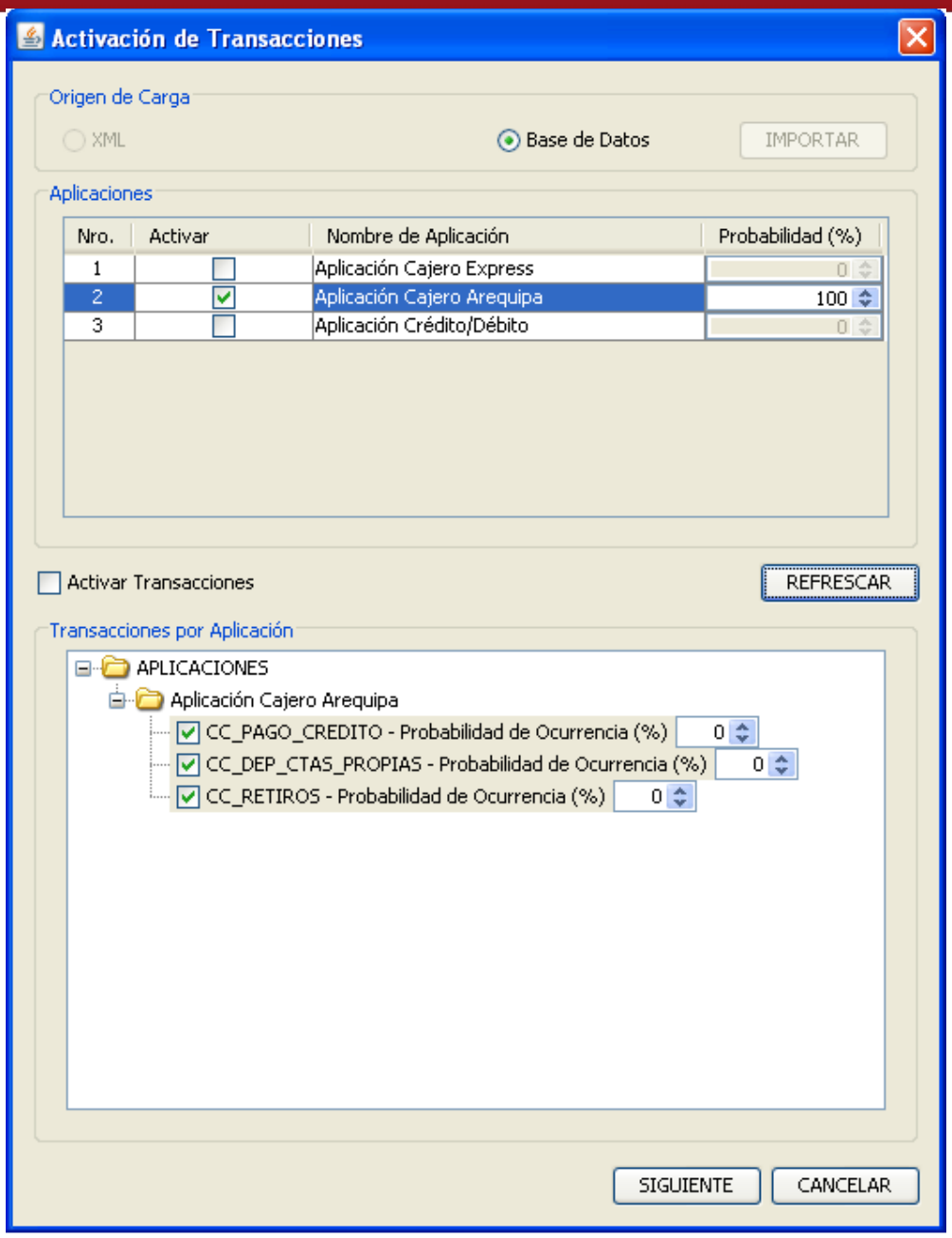
Por tal motivo la solución propuesta no solo debe poseer una interfaz amigable al usuario, sino que también debe permitir mostrar en forma ordenada elementos que cubran la necesidad definida en el documento.

12.2.1 Interfaces de la aplicación

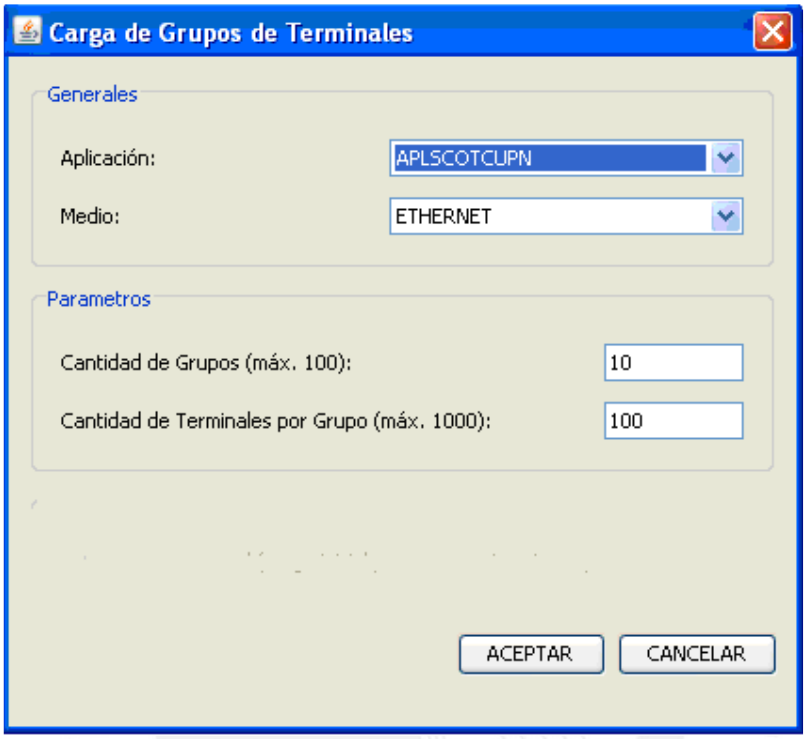
PROT01 Configuración General: Interfaz que nos permitirá configurar parámetros generales de la herramienta como datos del servidor Motor Enrutador Transaccional, conexión a Base de Datos.



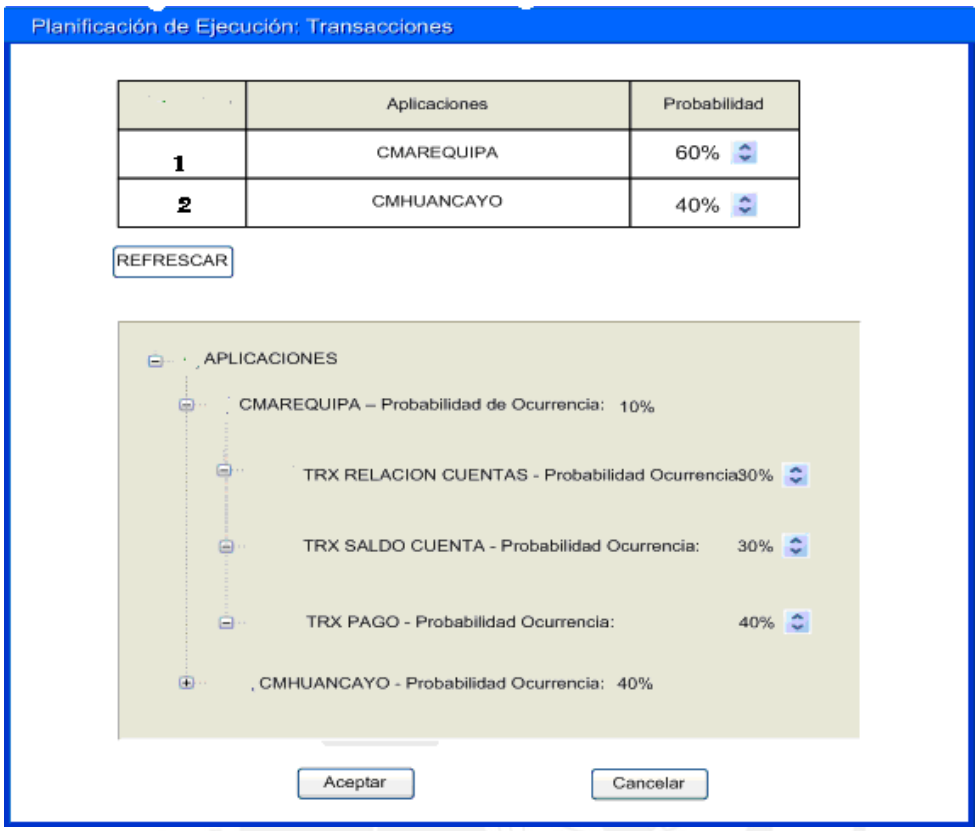
PROT02 Planificación de Ejecución: Interfaz que permite seleccionar inicialmente los tipos de aplicación que se ejecutarán y adicionalmente los tipos de transacciones y sus respectivas probabilidades de ocurrencia.



PROT03 Generación de Grupo de Terminales: Interfaz que permite crear grupos de terminales (cadenas comerciales).



PROT04 Configuración de Probabilidades Por Intervalo de Tiempo: Debido a la existencia de horarios en el día en donde algunos tipos de transacción (correctamente identificados) presentan mayor grado de ocurrencia esta interfaz permite configurar dichos intervalos.



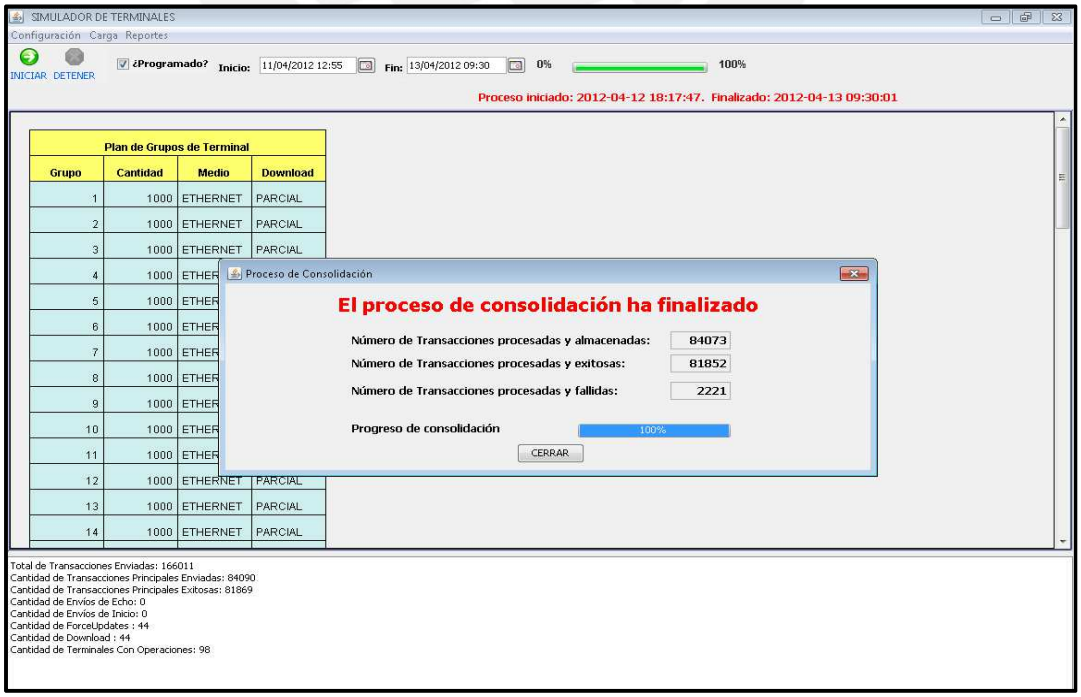
PROT05 Reporte de Ejecución: Permite obtener un informe de la ejecución que incluya transacciones realizadas por tipo de transacción, aplicación, exitosas y fallidas.

REPORTE DE EJECUCIÓN DE TRANSACCIONES FINANCIERAS POR APLICACIÓN						
TRANSACCIÓN	PROB. ASIGNADA	PROB. REALIZADA	CANT. TRXS.	EXITOSAS	FALLIDAS	DETALLE FALLIDAS
Aplicación Cajero Express	**1.0**	**0.27**	**65**	**65**	**0**	**0**
CC_DEP_CTAS_PROPIAS	0.5	0.12	28	28	0	**0**
CC_RETIROS	0.5	0.15	37	37	0	**0**
Aplicación Crédito/Débito	**1.0**	**0.46**	**113**	**113**	**0**	**0**
COMPRA	0.3	0.14	34	34	0	**0**
EFECTIVO	0.3	0.13	32	32	0	**0**
PAGO_SERVICIOS	0.2	0.12	29	29	0	**0**
GASOLINA	0.2	0.07	18	18	0	**0**
TOTALES	**1.0**	0.73	178	178	0	

PROT07 Reporte de Tiempos de Respuesta: Permite evaluar el porcentaje de transacciones con tiempo de respuesta óptimo.

TIEMPOS DE RESPUESTA												
Aplicaciones	Transacciones	0 - 500 ms		501 - 1000 ms		1001 - 5000 ms		5001 - 10000 ms		10001 - + ms		Total por Transacción
		Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%	
POS	PAGO_SERVICIOS	3	100.00	0	0.00	0	0.00	0	0.00	0	0.00	3
	COMPRA	1	100.00	0	0.00	0	0.00	0	0.00	0	0.00	1
	EFECTIVO	1	100.00	0	0.00	0	0.00	0	0.00	0	0.00	1
	GASOLINA	1	100.00	0	0.00	0	0.00	0	0.00	0	0.00	1
	Subtotal POS:	6	100.00	0	0.00	0	0.00	0	0.00	0	0.00	6
CAJERO	APLRETIRO	1	100.00	0	0.00	0	0.00	0	0.00	0	0.00	1
	DEP_CTAS_PROPIA	2	100.00	0	0.00	0	0.00	0	0.00	0	0.00	2
	Subtotal CAJERO:	3	100.00	0	0.00	0	0.00	0	0.00	0	0.00	3
	Total	22	100%	0	0.0	0	0.0	0	0.0	0	0.0	22

PROT08 Pantalla principal de ejecución de Transacciones. En la parte inferior se muestra el progreso de la ejecución y al finalizar se muestra el proceso de consolidación entre las transacciones generadas y los casos de éxito/fallido.



12.3 Construcción

12.3.1 Herramientas de Implementación

- **Java**. Su elección se debe principalmente por sus características de portabilidad e independencia de la plataforma y por ser de uso libre.
- **XML**. Es un lenguaje de marcas utilizado para almacenar datos en forma legible. Da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información

12.3.2 Selección de Otras Herramientas

- **SJT.Mgmt**. Es una librería para la administración de hilos en las aplicaciones Java. La gestión de hilos permite trabajar con múltiples hilos sin el costo de funcionamiento de la memoria y el esfuerzo de CPU. Los parámetros incluyen, tamaño máximo inicial, y el límite de carga de trabajo.
- **Log4j**. Es una herramienta robusta, confiable, completamente configurable, fácilmente extensible, y fácil de utilizar para mostrar y/o almacenar registros de los mensajes que se deseen generar en la aplicación.
- **JDOM**. Es una librería de código abierto para manipulaciones de datos XML optimizados para Java. JDOM se creó específicamente para usarse con Java y por lo tanto beneficiarse de las características de Java, incluyendo sobrecarga de métodos, colecciones.
- **JFreeChart**. Es un marco de software open source para el lenguaje de programación Java, el cual permite la creación de gráficos complejos de forma simple.
- **Colt**. Es una librería de código abierto para el uso de distribuciones probabilísticas y funciones estadísticas.

13 Ingeniería de la Solución

La aplicación consta de un proceso principal al cual se hace referencia como Gestor de Terminales y un proceso alterno llamado Gestor de Anulación de Transacciones, adicional a ello estará el contexto propio de la generación de hilos por cada terminal, en la Figura 6 se puede apreciar todo el flujo de la solución propuesta.

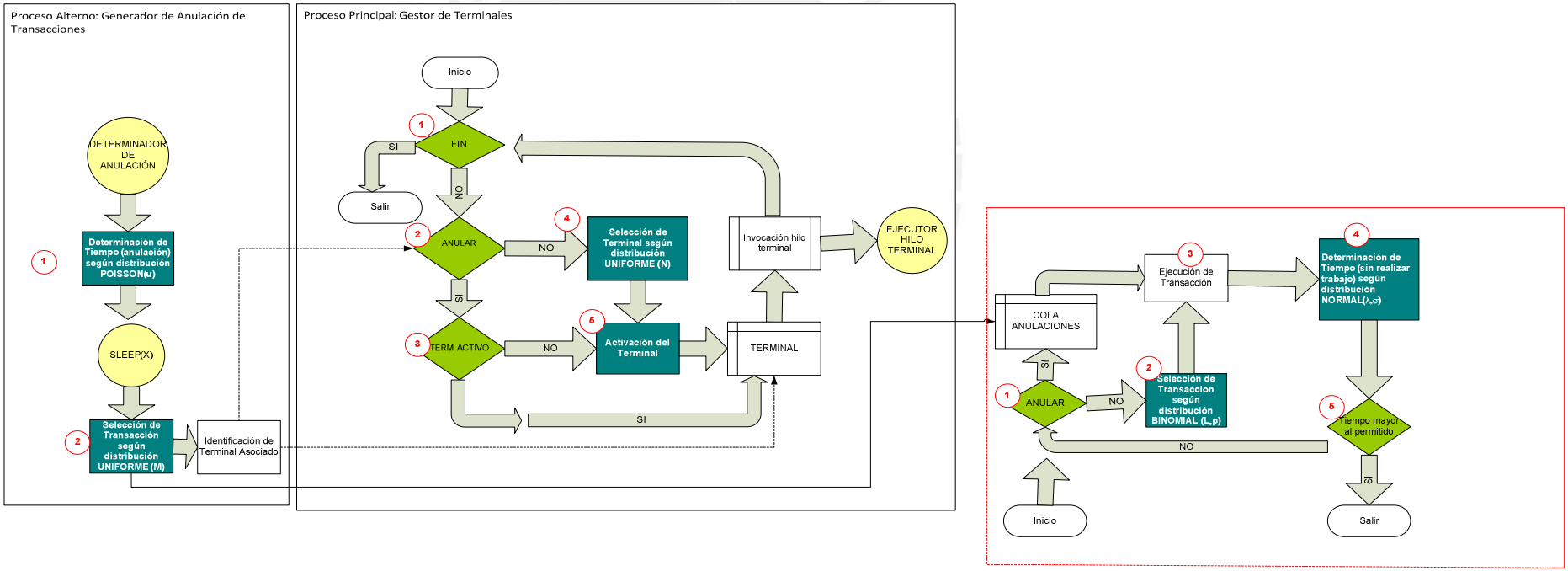


Figura: 6 Flujo Global de la Aplicación

A continuación se explica paso a paso todo el flujo de la solución propuesta que se divide en un proceso principal y otro alterno:

Proceso Principal- Gestor de Terminales

Proceso que evalúa que la ejecución se realice dentro del rango de tiempo establecido, administra la operatividad de los terminales y a la vez interactúa con el proceso alterno de identificación de anulaciones de transacción.

- Paso 1: Se evalúa la duración de la ejecución del programa; si aún se encuentra en el periodo de ejecución establecido procede al paso 2, caso contrario concluye el programa.
- Paso 2: En la interacción con el proceso alterno se valida si existe una transacción por anular; en caso sea afirmativo procede con el Paso 3; caso contrario con el Paso 4.
- Paso 3: Se valida si el terminal donde se ejecutó la transacción a anularse encuentra activo, en caso sea afirmativo procede al Paso 4; caso contrario, con el número de terminal de la transacción a anular, procede al Paso 5.
- Paso 4: Se seleccionará un terminal según distribución Uniforme para que realice transacciones.
- Paso 5: Se ejecuta el hilo terminal.

Proceso Alterno- Gestor de Anulación de Transacciones:

Proceso que identifica el momento en el cual existe una transacción pendiente de anular enviando al proceso principal el identificador de la transacción por anular y el terminal asociado. Realizar la anulación de una transacción solo es posible en el mismo terminal donde se genera la transacción inicial.

- Paso 1: Aleatoriamente según distribución de Poisson se determinará en que instante de tiempo se generará una transacción de anulación.
- Paso 2: Según distribución Uniforme se seleccionará la transacción que se anulará. En este punto se enviará al proceso principal el identificador de la transacción y el terminal donde se realizó.

Proceso de Ejecución de Hilo Terminal:

Durante el proceso principal se crea un hilo por cada terminal activado y realiza los siguientes pasos:

- Paso 1: Se valida si es una transacción de anulación, si es afirmativo procede al Paso 3; caso contrario continua con el Paso 2.
- Paso 2: De manera aleatoria según distribución Poisson se designa el tipo de transacción financiera a ejecutar.
- Paso 3: Se procede a ejecutar la transacción. En caso exista anulación pendiente, se prioriza la ejecución de la anulación.
- Paso 4: Según distribución normal se determinará el tiempo durante el cual el terminal se encontrará sin realizar ningún trabajo hasta la llegada de una nueva transacción.
- Paso 5: Se valida si el tiempo aleatorio que se obtuvo en el Paso 5 es mayor a X minutos (configurable) en caso sea afirmativo el terminal se desactivará caso contrario volverá al paso 1.

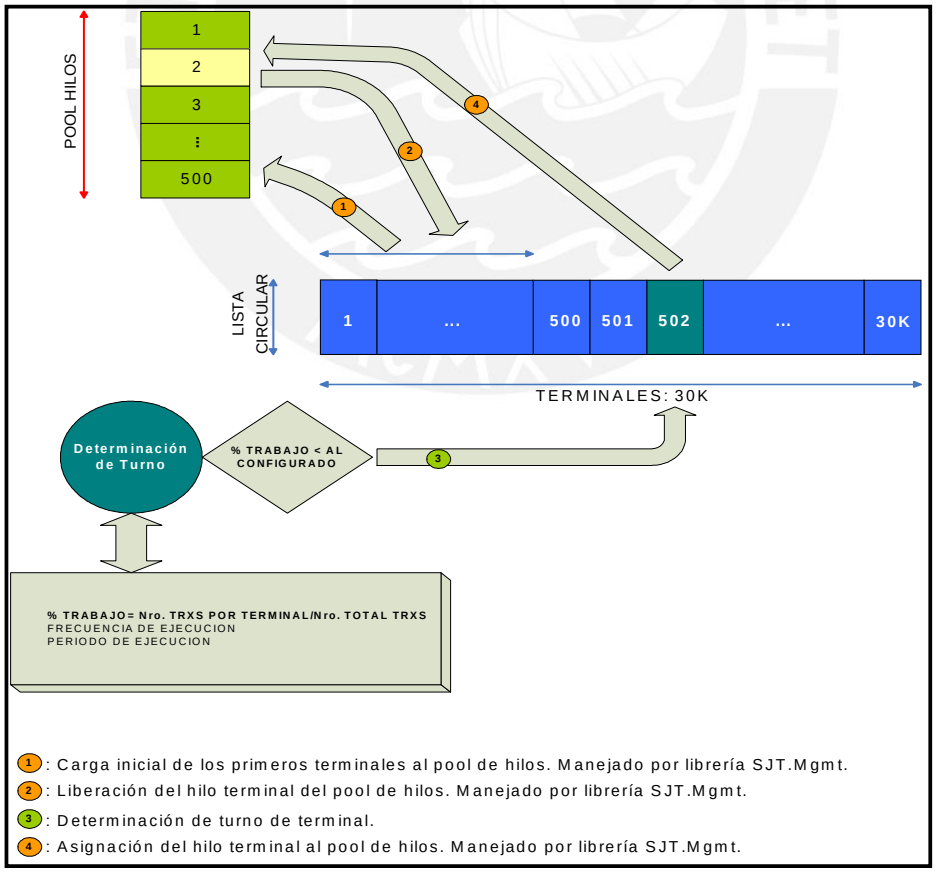


Figura: 7 Gestión de Terminales mediante Hilos

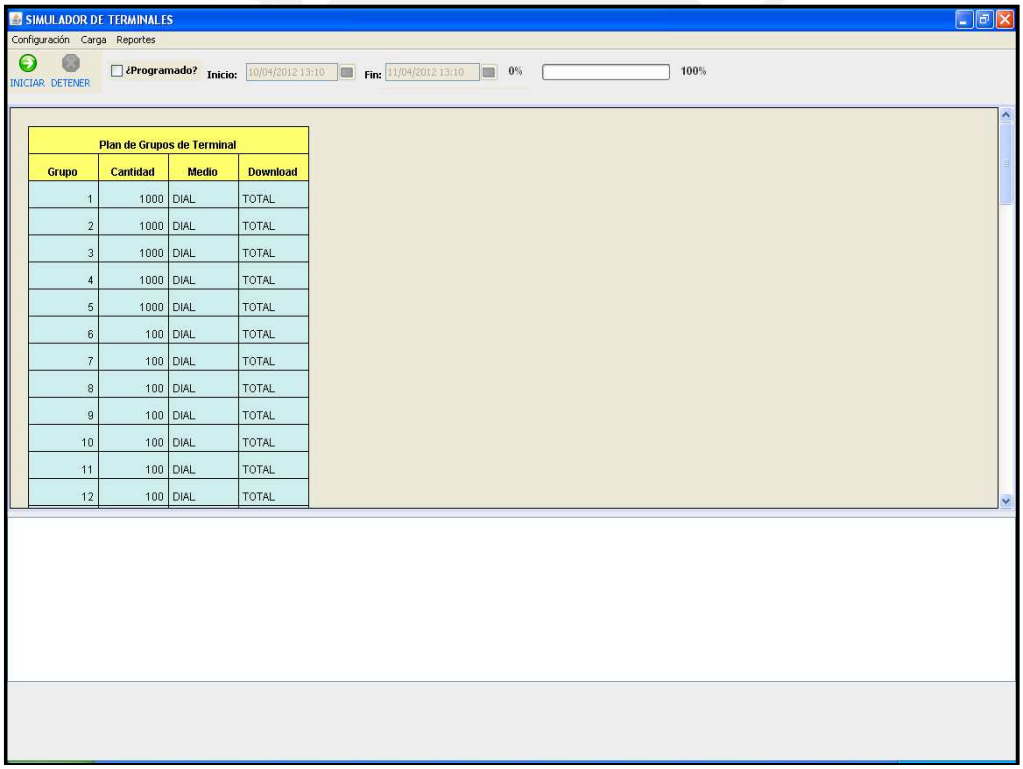
14 Procedimiento de la Ejecución de la Solución Propuesta

La aplicación propuesta como solución es una herramienta que va a permitir medir el desempeño del motor enrutador transaccional, mediantes la simulación de carga de trabajo de una red de terminales que generan transacciones según las aplicaciones disponibles.

Para ello, se contará con una serie de configuraciones que permitirán realizar dicha labor, y obtener diversas estadísticas que ayudarán a obtener una serie de reportes, con los cuales se podrá observar el desempeño del motor enrutador transaccional ante diversos escenarios de esfuerzo.

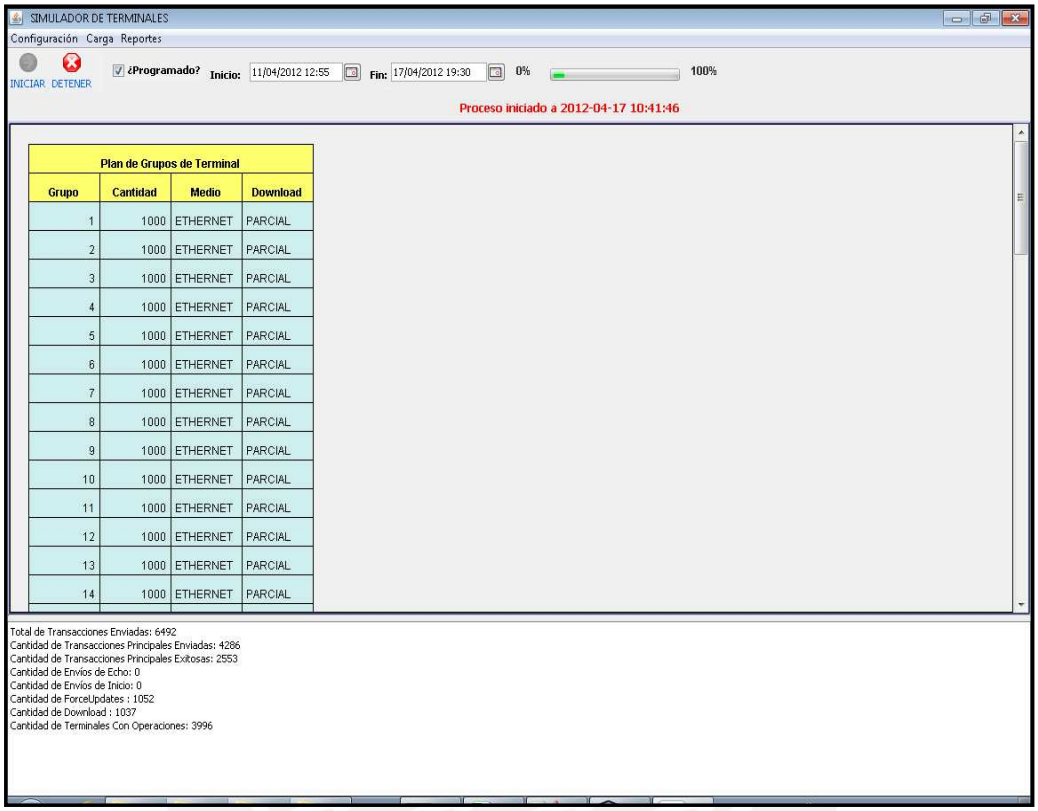
14.1 Ingreso a la Aplicación

Al momento de ingresar a la aplicación se muestra un formulario como el siguiente:



1. Lo primero que se debe hacer es configurar correctamente todo lo necesario para el ambiente de ejecución, para ello revisar puntos **3,4 y5**. Si se requiere conservar la configuración, entonces se procede a ingresar la programación de fecha y hora de ejecución directamente en la pantalla principal haciendo clic en "**¿programado?"** e ingresando **"Inicio"** y **"Fin" (Si el ingreso es a través del teclado PULSAR ENTER).**

2. Luego podemos dar clic en **INICIAR** y se iniciará el envío de transacciones al motor enrutador transaccional, según configuración establecida.

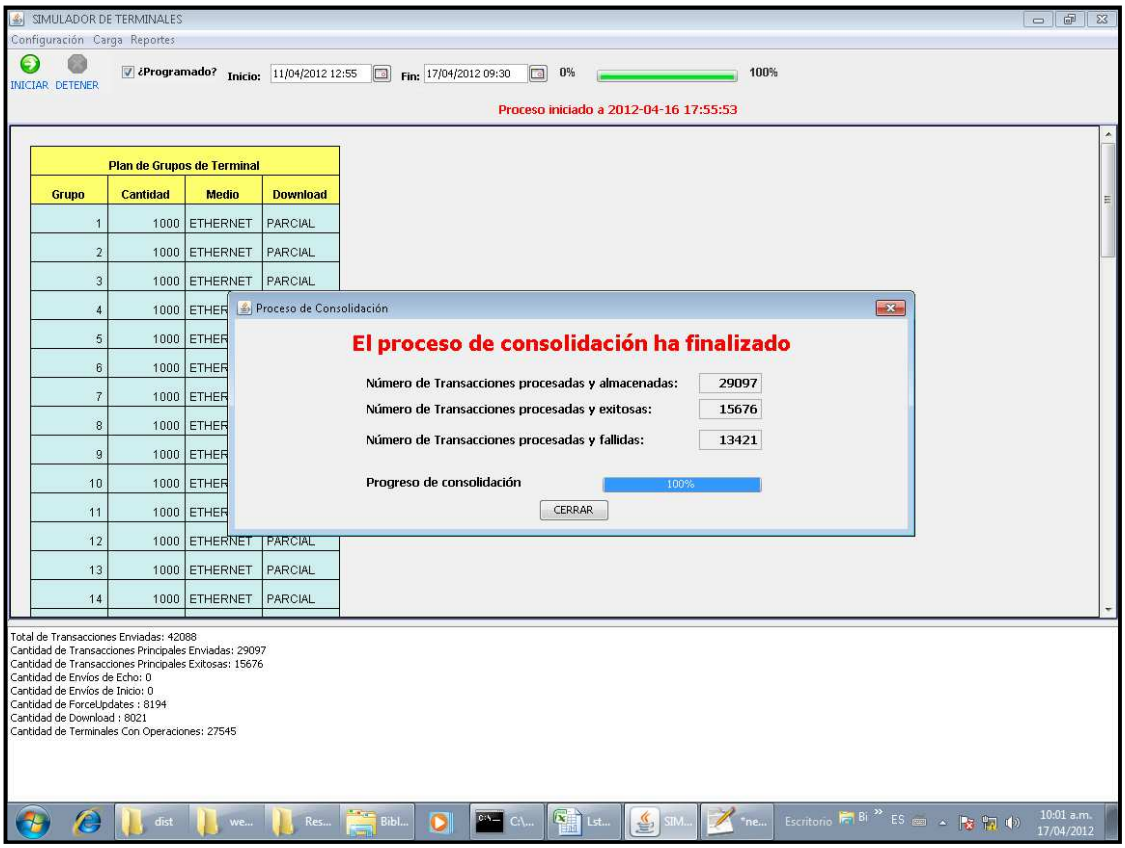


Durante la ejecución se muestra en la interfaz de usuario el progreso de la ejecución (parte superior derecha de la pantalla) en función al tiempo transcurrido. En el panel inferior de la pantalla se muestra los contadores de la ejecución que proveen la siguiente información:

- Total de transacciones enviadas: expresa el número de transacciones (entre intermedias y principales) que se están generando.
- Cantidad de transacciones principales enviadas: expresa el número de transacciones principales (no intermedias) que se están generando.
- Cantidad de transacciones principales exitosas: expresa el número de transacciones principales (no intermedias) con respuestas exitosas.
- Cantidad de envíos de ECHO: expresa la cantidad de envíos de ECHO al motor enrutador transaccional.
- Cantidad de envíos de INICIO: expresa la cantidad de envíos de INICIO al motor enrutador transaccional.

- Cantidad de “ForceUpdate”: expresa la cantidad de actualizaciones forzadas iniciadas luego de notificarse en alguna respuesta de una transacción.
- Cantidad de “Download”: expresa la cantidad de descargas realizadas con éxito entre envíos de INICIO (programado) y actualizaciones forzadas.
- Cantidad de terminales con operaciones: expresa el número de terminales en actividad, es decir, realizan transacciones.

En cualquier momento se puede presionar el botón **DETENER** y finalizará la ejecución de transacciones**.**



Al finalizar la ejecución se realiza de forma automática el proceso de consolidación de reportes y se muestra un cuadro de dialogo con información resumen de dicho proceso que comprende:

- Número de transacciones procesadas y almacenadas: corresponde a la cantidad de transacciones principales enviadas (generadas) por la red de terminales.
- Número de transacciones procesadas y exitosas: corresponde a la cantidad de transacciones principales exitosas reportadas por la aplicación.

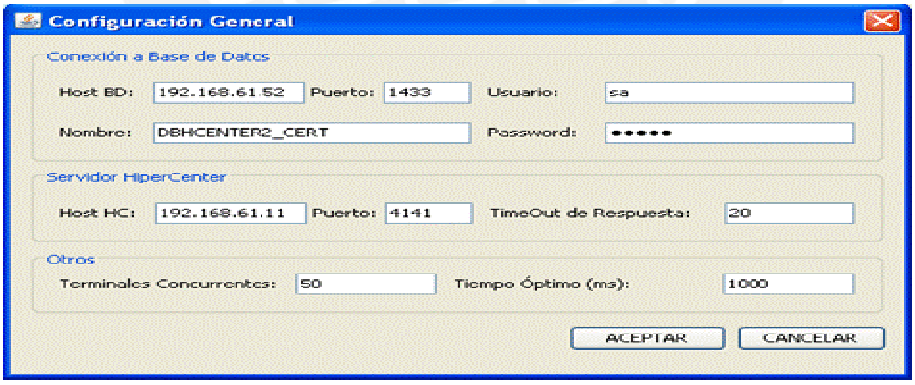
- Número de transacciones procesadas y fallidas: expresa la cantidad de transacciones reportadas como fallidos.

14.2 Configuración

Con este menú se tendrá acceso a todas las configuraciones necesarias para el proceso de planificación y preparación del ambiente antes de iniciar la ejecución. Se presenta las siguientes configuraciones: **Configuración General y Ejecución de Transacciones**.

14.2.1 Configuración General

1. En donde se podrá configurar todos los datos necesarios para la comunicación con Base de Datos y motor enrutador transaccional, así como la cantidad de número de terminales concurrentes. El uso de una Base de Datos se justifica por el almacenamiento de la información que genera cada transacción desde la aplicación simulador de terminales y finalmente con dicha información se generan los reportes de rendimiento.
2. Seleccionar del menú **Configuración** la opción **General**.
3. El sistema muestra el formulario **"Configuración General"**.



4. Ingresar Host BD, IP de host donde se encuentra la base de datos.
5. Ingresar Puerto, puerto de base de datos (default 1433).
6. Ingresar Nombre, nombre de la base de datos.
7. Ingresar Usuario, de acceso a la base de datos especificada.
8. Ingresar Password, clave de acceso del usuario a la base de datos especificada.

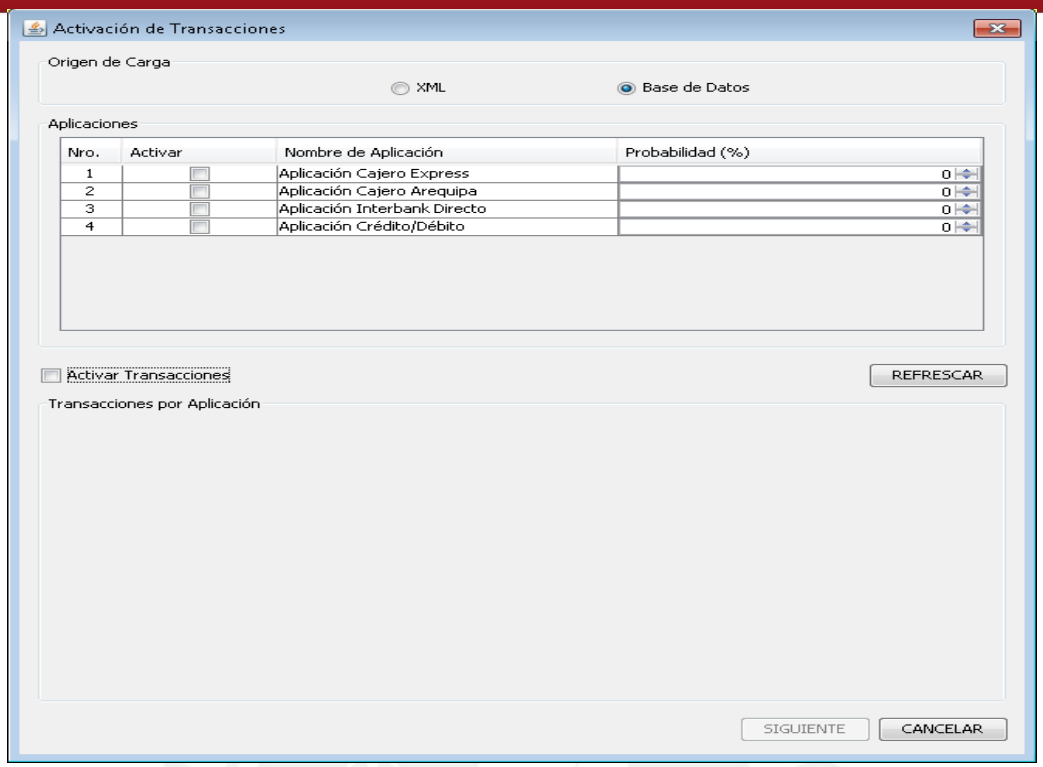
9. Ingresar Host HC, IP de host donde se encuentra el motor enrutador transaccional.
10. Ingresar Puerto, puerto del motor enrutador transaccional.
11. Ingresar TimeOut de Respuesta, tiempo en segundos que la aplicación espera las respuestas del motor enrutador transaccional.
12. Ingresar la cantidad de Terminales Concurrentes, número de terminales que procesarán transacciones en un mismo intervalo de tiempo.
13. Ingresar Tiempo Óptimo (ms), en milisegundos.
14. Después de ingresar correctamente los datos necesarios presionar el botón ACEPTAR.
15. El sistema muestre el siguiente mensaje:

14.2.2 Configuración de Ejecución de Transacciones

Se podrá configurar las diversas aplicaciones que se desean programar con sus respectivas asignaciones de carga de trabajo (probabilidad), así como las transacciones a ejecutar para dichas aplicaciones indicando también las probabilidades que se requieran manejar.

Transacciones

1. Seleccionar del menú Configuración, el submenú Ejecución y luego la opción Transacciones.
2. El sistema muestra el formulario “Activación de Transacciones”.



3. En el panel **Aplicaciones** se muestra una tabla en donde se cargan todas las aplicaciones de la BD que se encuentren en el archivo de configuración **"Transacciones.xml"**, ubicado en la ruta: **config\definicion.**
4. Seleccionar de la columna **Activar** una o más aplicaciones que se deseen configurar, indicando para cada aplicación seleccionada su correspondiente valor de **Probabilidad(%),** teniendo en cuenta lo siguiente:
   - Una aplicación seleccionada no puede tener probabilidad de 0%.
   - La suma de todas las probabilidades de las aplicaciones seleccionadas debe ser 100%.

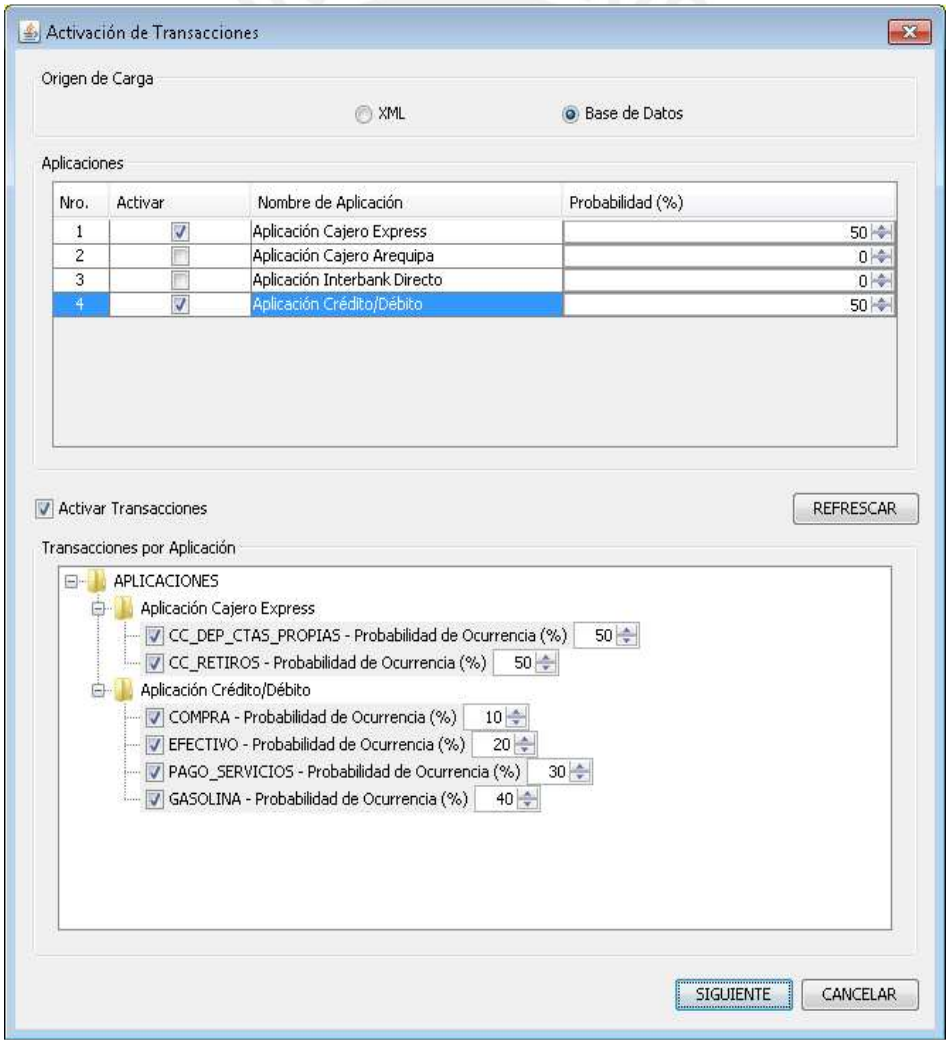
Observación:

- Al momento de ingresar las probabilidades, **sólo** se podrán ingresar valores en aquellas casillas cuya aplicación esté **Activada,** se puede hacer uso del Spinner para aumentar o disminuir el valor.

- Si se hace uso del teclado para ingresar un valor, después de ingresar el valor se debe presionar la tecla **ENTER**, de este modo se asigna el valor ingresado en dicha casilla, de lo contrario se asignará el valor en la siguiente casilla seleccionada.

- En caso se requiera agregar una o más aplicaciones, editar el archivo "**Transacciones.xml**" que se encuentra en la ruta: "**config\definicion**".

5. Después de configurar correctamente las aplicaciones presionar el botón **REFRESCAR.**
6. Se carga dentro del panel **Transacciones por Aplicación,** una estructura en forma de árbol que muestra, para todas las aplicaciones seleccionadas, sus respectivas **transacciones financieras habilitadas** que puede procesar, dichas transacciones se pueden corroborar en el archivo de configuración **"Transacciones.xml"**, ubicado en la ruta: **config\definicion.**



7. Seleccionar de la estructura del árbol, para cada aplicación, **al menos una** Transacción, y programar su correspondiente valor de **probabilidad(%),** teniendo en cuenta lo siguiente:

   - Una transacción seleccionada no puede tener probabilidad de 0%.
   - La suma de todas las transacciones seleccionadas **por cada aplicación** debe ser 100%.

Observación:

- Al momento de ingresar las probabilidades, **sólo** se podrán ingresar valores en aquellas casillas cuya transacción esté **Activada,** se puede hacer uso del Spinner para aumentar o disminuir el valor.

- Si se hace uso del teclado para ingresar un valor, después de ingresar el valor se debe presionar la tecla **ENTER**, de este modo se asigna el valor ingresado en dicha casilla, de lo contrario se asignará el valor en la siguiente casilla seleccionada.

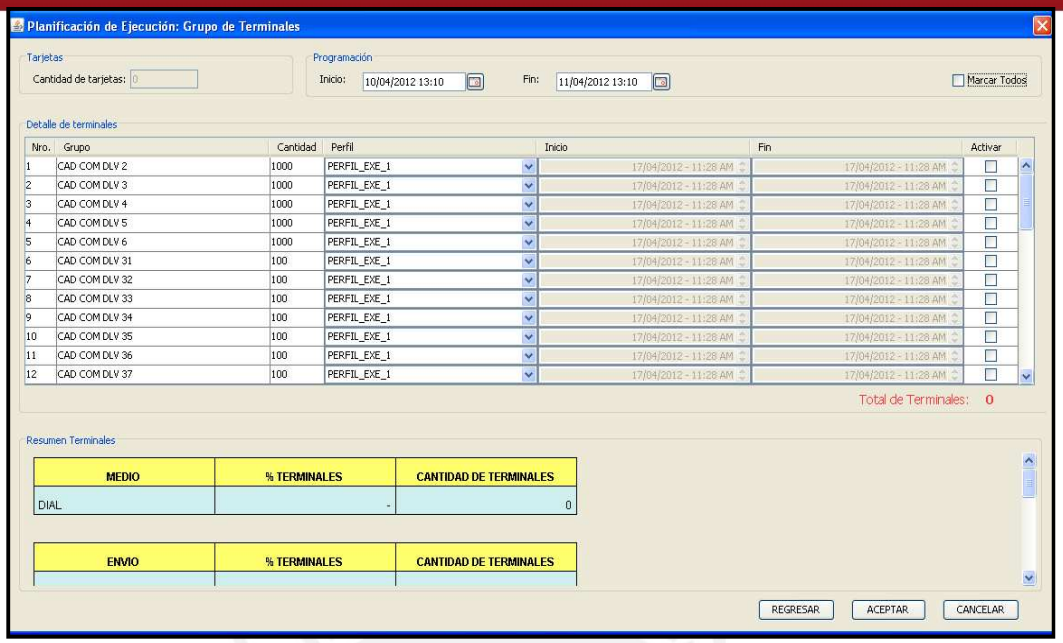
- En caso se requiera agregar uno o más tipos de transacción, editar el archivo "**Transacciones.xml**" que se encuentra en la ruta: "**config\definicion**".

8. Después de configurar correctamente las transacciones por aplicación, presionar el botón **SIGUIENTE.**

Planificación de la Ejecución grupo de Terminales

En donde se podrán configurar, mediante la columna **Activar** qué Grupos de Terminales serán procesados, y en un rango de Tiempo establecido (Hora Inicio y Fin) con la finalidad de obtener las pruebas necesarias de Performance del motor enrutador transaccional.

1. Una vez realizada correctamente la configuración del formulario **"Activación de Transacciones",** después de seleccionar SIGUIENTE.
2. El sistema muestra el formulario **"Planificación de Ejecución: Grupo de Terminales"**.



3. Ingresar **INICIO**, se programa Fecha y hora de inicio de ejecución del envío de transacciones.
4. Ingresar **FIN**, se programa Fecha y hora de fin de ejecución del envío de transacciones.
5. Opcionalmente se puede seleccionar la opción **Marcar Todos,** lo que activa/desactiva todos los grupos de terminales.
6. Dentro de la tabla se tiene:

   - Grupo: Contiene el nombre de todos los grupos de terminales registrados en la BD. En caso se requiera crear grupos de terminales, ir a la sección **5.1 GRUPO DE TERMINALES**.
   - Cantidad: Número de terminales que hay por cada Grupo.
   - Activar: Identifica los grupos de terminales activos para el envío de transacciones.
   - Perfil: se asocia un perfil de ejecución que reúne ciertas características de ejecución tales como: medio, activar envíos de INICIO, forma de descarga. En caso se requiera asociar un nuevo perfil, entonces ir a la sección **5.2 GESTIÓN DE PERFILES DE EJECUCIÓN**.

7. Después de seleccionar e ingresar correctamente los datos necesarios, presionar el botón **ACEPTAR.**
8. Esperar un momento (según la cantidad total de terminales seleccionados), hasta que el sistema muestre el siguiente mensaje:

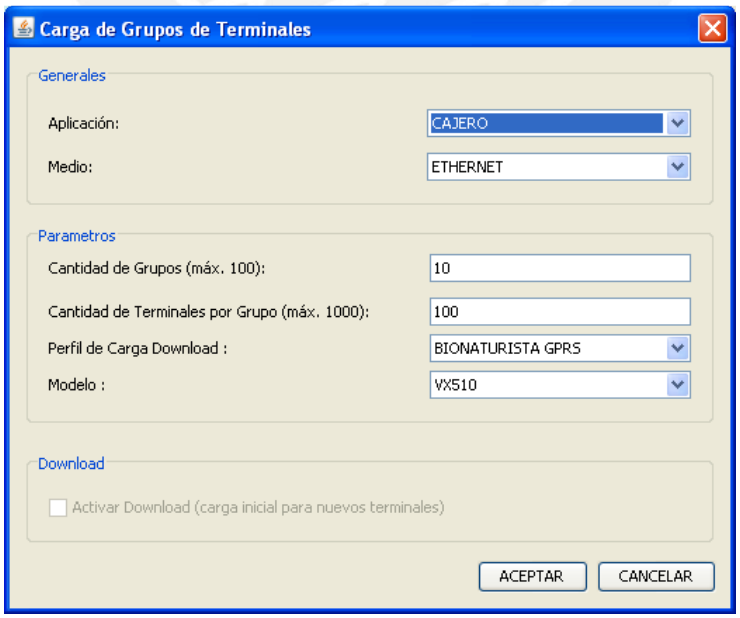
14.3 CARGA

Con este menú se tendrá acceso a poder realizar una carga de grupos de terminales directamente en BD, cuando es necesario aumentar la cantidad de terminales registrados y así poder realizar la **Simulación,** se pueden generar hasta cien mil Terminales.

14.3.1 Grupo de Terminales

En donde se podrán crear nuevos terminales según las características que se indiquen como: Aplicación, Medio, Cantidad de Grupo y Cantidad de Terminales por Grupo.

1. Seleccionar del menú **Carga** la opción **Grupo de Terminales**.
2. El sistema muestra el formulario **"Carga de Grupos de Terminales"**.



3. Se debe corroborar que en el combo box **Aplicación** se muestren sólo las aplicaciones de la BD que se encuentren en el archivo **"Transacciones.xml".**
4. Seleccionar la **Aplicación** a la cual se van a asociar los nuevos terminales.
5. Seleccionar el **Medio** que manejarán los nuevos terminales.
6. Ingresar la **Cantidad de Grupos** a crear (Cadenas de Comercio).
7. Ingresar la **Cantidad de Terminales por Grupo** (Nro. de terminales que se crearán para cada Grupo).
8. Seleccionar un **Perfil de carga Download**.
9. Seleccionar un **modelo** de terminal.
10. Después de seleccionar e ingresar correctamente los datos necesarios, presionar el botón **ACEPTAR.**
11. Esperar unos segundos (según las cantidades ingresadas), hasta que el sistema muestre el siguiente mensaje:

14.3.2 Gestión de Perfiles de Ejecución

La gestión de perfiles de ejecución tiene como finalidad reunir características de ejecución (medio, forma de DOWNLOAD, envío de ECHO, envío de INICIO, envío de alarma) y adoptarlas bajo un perfil el cual es asignado posteriormente a un grupo de terminales.

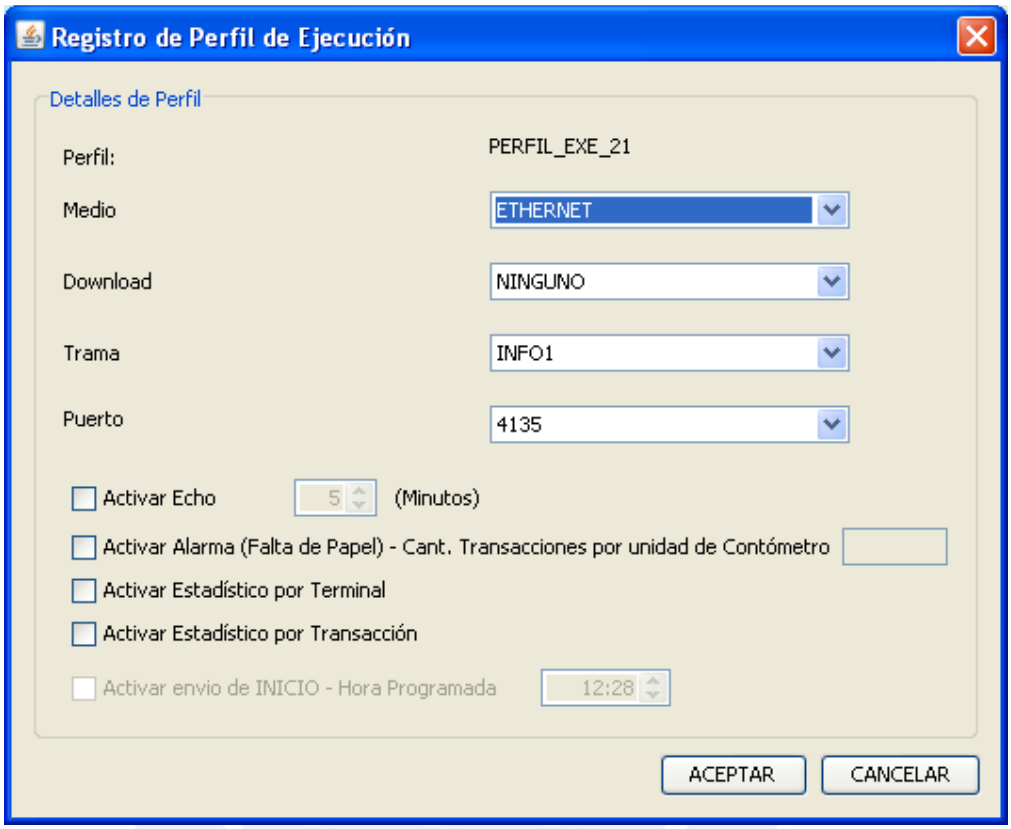
1 Seleccionar del menú **Carga** la opción **Perfil de Ejecución**.

2 Se presenta la interfaz maestra de perfiles de ejecución donde se muestra un listado de perfiles activos. Para crear un nuevo perfil, hacer clic en el botón "NUEVO".

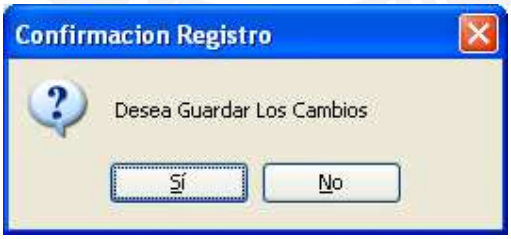
3 Se presenta la interfaz de registro de un nuevo perfil de ejecución con un nombre autogenerado manteniendo su correspondiente correlativo. Se ingresa los siguientes campos:

- Medio: es el medio de comunicación que podría ser ETHERNET, DIAL entre otros.
- Download: es la forma de descarga de actualización de versión de una aplicación. Si se elige NINGUNO entonces no se realiza descargas; si se elige PARCIAL entonces la descarga consiste en actualización de archivos determinados de forma aleatoria; y si se elige TOTAL entonces la descarga es completa, es decir, todos los archivos de la versión.
- Trama: se indica el formato de trama en INFO1 o INFO2.
- Puerto: se indica el puerto de comunicación con el motor enrutador transaccional.
- Activar “Echo”: se habilita el envío de echo indicando un periodo de tiempo expresado en minutos. El mínimo es cinco minutos.
- Activar Alarma: se habilita el envío de alarma indicando un valor estadístico el cual es el promedio de transacciones que consume una unidad de contómetro. Esto expresa que cada vez que se realiza un consumo entonces se envía una alarma al motor enrutador transaccional.
- Activar estadístico por terminal: se habilita la generación de estadísticas por terminal y el envío al motor enrutador transaccional de dicha estadística.

- Activar estadístico por transacción: se habilita la generación de estadísticas por transacción y el envío al motor enrutador transaccional de dicha estadística.

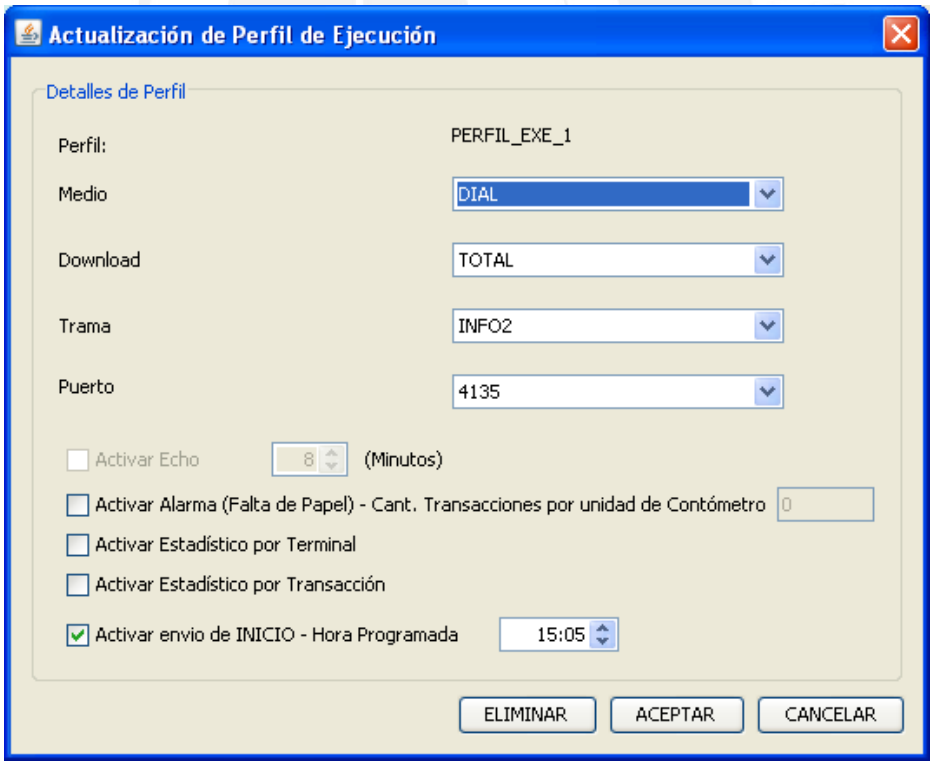


4 Se hace clic en el botón “ACEPTAR” y se confirma el registro.



5 Si se requiere modificar o eliminar un registro de perfil de ejecución, entonces ubicar un registro en la interfaz de gestión de perfiles de ejecución y hacer clic en “ACTUALIZAR”.

6 Ingresar los campos que requieran modificación y aceptar los cambios o hacer clic en “ELIMINAR” en caso se requiera eliminar el registro.



14.3.3 Reportes de Ejecución

La herramienta de Simulación de Terminales genera reportes de ejecución a partir de información de transacciones generadas por cada aplicación. La información más relevante son los tiempos de respuesta de cada transacción, con ello se determina el porcentaje de cumplimiento de tiempo óptimo de respuesta.

Reporte de Transacciones Financieras por Aplicación

1. Seleccionar el menú “Reportes”.
2. Seleccionar el reporte “Transacciones Financieras por Aplicación”.

REPORTE DE EJECUCIÓN DE TRANSACCIONES FINANCIERAS POR APLICACIÓN								
TRANSACCIÓN	PROB. ASIGNADA	PROB. REALIZADA	CANT. TRXS.	EXITOSAS	FALLIDAS	DETALLE FALLIDAS		
						DESCONOCIDO	SOCKET IOException	TIME OUT
Aplicación Cajero Express	**100.0**	**100.0**	**71478**	**47810**	**23668**	**23322**	**48**	**298**
CC_DEP_CTAS_PROPIAS	50.0	49.81	35639	23815	11824	11629	24	171
CC_RETIROS	50.0	50.19	35839	23995	11844	11693	24	127
TOTALES	100.0	100.0	71478	47810	23668			

Reporte de Tiempos de Respuestas de Transacciones

1. Seleccionar el menú “Reportes”.
2. Seleccionar el reporte “Tiempos de Respuestas de Transacciones”.

TIEMPOS DE RESPUESTA												
Aplicaciones	Transacciones	0 - 500 ms		501 - 1000 ms		1001 - 5000 ms		5001 - 10000 ms		10001 - + ms		Total por Transacción
		Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%	
CAJERO	APLRETIRO	0	0.00	5	0.33	1443	93.88	89	5.79	0	0.00	1537
	DEP_CTAS_PROPIA	0	0.00	5	0.34	1386	94.41	77	5.25	0	0.00	1468
	Subtotal CAJERO:	0	0.00	10	0.33	2829	94.14	166	5.52	0	0.00	3005
POS	PAGO_SERVICIOS	1	0.58	12	6.94	158	91.33	2	1.16	0	0.00	173
	COMPRA	0	0.00	3	7.14	39	92.86	0	0.00	0	0.00	42
	GASOLINA	1	0.37	11	4.07	255	94.44	3	1.11	0	0.00	270
	EFECTIVO	0	0.00	6	4.88	117	95.12	0	0.00	0	0.00	123
	Subtotal POS:	2	0.33	32	5.26	569	93.59	5	0.82	0	0.00	608
	Total	2	0.06	42	1.16	3398	94.05	171	4.73	0	0.00	3613

15 Conclusiones y Recomendaciones

15.1 Conclusiones

Como resultado del desarrollo del proyecto Tesis se concluye lo siguiente:

- Se logró desarrollar una aplicación que permite simular un ambiente de terminales de punto de venta (POS) para la ejecución de pruebas de esfuerzo sobre motores enrutadores transaccionales.

- Se logró simular escenarios configurando horarios con mayor volumen de transacciones para identificar los tiempos de respuesta.

- Uno de los factores claves para el éxito de las pruebas de esfuerzo radica en el diseño del software, es decir, la construcción de componentes desacoplables e independientes de tal forma que facilita la prueba de esfuerzo aislada con el objetivo de optimizar cada componente que conforma el software final. En el caso del motor enrutador transaccional, es importante que disponga de un componente independiente encargado de indicar disponibilidad del servicio y esto se implementa con las transacciones tipo “echo”, precisamente con el cual se recomienda iniciar la prueba de esfuerzo.

- Se ratificó que los tiempos de respuestas están sujetos a una serie de variables, una de las más importantes es el volumen de transacciones con frecuencia de ejecución constante donde se evidencia niveles de degradación hasta obtener un tiempo de ejecución estable. La idea es identificar el volumen y el tiempo a partir del cual una aplicación empieza a degradarse y a partir de ello tomar acciones de optimización.

- La utilización de APIS de libre distribución permite mitigar los riesgos de redundancia de código y economizan el tiempo de implementación de una solución de software, tal es el caso del API de manejador de hilos que posibilitó la implementación del componente medular de la aplicación de simulador de ambiente de terminales.

- Finalmente, en relación a la metodología, la aplicación de los conceptos de RUP para las etapas de concepción y elaboración permitieron la refinación de los

requerimientos funcionales y no funcionales de la solución del proyecto Tesis, luego la ejecución de las etapas de construcción y transición a través de los conceptos ágiles permitieron que el producto evolucione favorablemente por las constantes retroalimentaciones en los entornos de pruebas. Ambos conceptos forman parte de la metodología AUP que ha sido utilizada para el desarrollo de esta tesis.

15.2 Recomendaciones

Como complemento a las optimizaciones a nivel de software en busca de la mejora de las plataformas informáticas transaccionales se recomienda incorporar los siguientes aspectos:

- Topología alto desempeño: Redes de topología con tolerancia a fallos.
- Balanceadores de carga: Balanceadores que permitan distribuir la atención de una transacción según la carga de los servidores transaccionales disponibles.
- Servidores espejos en cluster. Los servidores en cluster permiten que se tenga una opción de atención en caso una no se encuentre disponible.
- Procesos automatizados de monitoreo de transacciones. Estos procesos permiten identificar transacciones encoladas o no atendidas para que se tome una acción de reintento o reversión.
- Notificadores automáticos en situaciones de fallo: Los notificadores permiten conocer las situaciones de fallo para luego desencadenar una acción manual o automática de reparación.

16 Referencias bibliográficas

I. [ERINLE,2013]
Erinle, Bayo
2013 "Performance Testing with JMeter 2.9", Primera Edición, Packt Publishing Ltd., UK.

II. [FOWLER,2002]
Fowler Martin /Beck Kent / John Brant/William Opdyke, y Don Roberts
1999 "Refactoring: Improving the Design of Existing Code", Amazon Digital Services, Inc., Primera Edición, USA.

III. [FOWLER,BECK, 2000]
Martin Fowler / Kent Beck
2000 "Planning Extreme Programming", Addison-Wesley Professional, Primera Edición, USA.

IV. [GONZALES,2006]
Gonzales Rocío, Bastos Ana Isabel
2006 "Operativa de caja-Terminal punto de venta", Ideaspropias Editorial, Primera Edición, España.

V. [HERNANDEZ-RAMOS-VELEZ,2011]
Hernández, Víctor / Ramos, Eduardo / Vélez, Ricardo
Modelos Probabilísticos y Optimización, Ediciones Académicas, Primera Edición, España.

VI. [KLEIMAN,1996]
Kleiman,Steve
1996 "Programming With Threads", Prentice Hall, Primera Edición.

VII. [KRUCHTEN,2003]
Kruchten, Philippe
2003 "The Rational Unified Process: An Introduction", Addison-Wesley Professional, Tercera Edición, USA.

VIII. [OAKS,2004]
Oaks Scott, Wong Henry

2004 “Java Threads”, O'Reilly Media, Tercera Edición, USA.

IX. [PRESSMAN,2002]
Pressman, Roger
2002 “Ingeniería del software: un enfoque práctico”, McGraw-Hill.

X. [SOMMERVILLE,2006]
Sommerville, Ian
2006 “Ingeniería del software”, Pearson Educación S.A, Sétima Edición, España.

XI. [MC DEMID, 1992]
McDemid John
“Software Engineer's Reference Book”, Butterworth-Heinemann Ltd.

XII. [MEDINA,2014]
Medina, Javier
2014 “Pruebas de Rendimiento TIC”, Segunda Edición, SG6 C.B., Murcia-España.

XIII. [MICROSOFT, 2007]
Microsoft Corporation
“Performance Testing Guidance for Web Applications”, Microsoft Press, Primera Edición, USA.

XIV. [MOLYNEAUX,2009]
Molyneaux, Bayo
2009 “The art of application Performance Testing”, O'Reilly Media, Primera Edición, USA.

XV. [AUP, 2015]
Ambysoft Inc.
Consulta: Agosto 2015
http://www.ambysoft.com/unifiedprocess/agileUP.html

XVI. [JMETER,2013]
Apache Software Foundation.org
Consulta: Noviembre 2013

http://jmeter.apache.org

XVII. [LOADRUNNER,2013]
Hewlett-Packard Development Company,L.P.
Consulta: Setiembre 2013
http://www8.hp.com/pe/es/software-solutions/loadrunner-load-testing/index.html

XVIII. [LOADUI,2015]
LoadUI.org
Consulta: Agosto 2015
https://www.loadui.org/

XIX. [NEOLOAD,2015]
RadView Software Ltd
Consulta: Agosto 2015
http://www.neotys.com/product/overview-neoload.html

XX. [READMINE,2013]
ReadMine.Org
Consulta: Agosto 2013
http://www.redmine.org/

XXI. [SIMPLE THREAD,2013]
Simple Java Thread Management (SJT.MGMT)
http://simplethread.sourceforge.net/
Consulta: Diciembre 2013

XXII. [STANDISH GROUP, 2013]
Standish Group
Consulta: Enero 2014
www.standishgroup.com

XXIII. [WEBLOAD,2015]
Neotys USA Inc.
Consulta: Agosto 2015
http://www.neotys.com/product/overview-neoload.html