FACULTAD 3

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Título: Migración de la capa de presentación de los componentes Flujo de Trabajo y Configuración del Sistema de Planificación de Actividades SIPAC a Ext.js v6.0

Autor: Keylier Vega Lobaina

Tutor: Ing. Rodolfo Rodríguez Molinet

Ing. Yordi Chaveco Bustamante

La Habana, junio de 2016

"Año 58 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Firma del autor

_____

Keylier Vega Lobaina

Firma del tutor

_____

Ing. Rodolfo Rodríguez Molinet

Firma del tutor

_____

Ing. Yordi Chaveco Bustamante

DATOS DEL CONTACTO

Autor:

Keylier Vega Lobaina.

Correo electrónico: kvega@estudiantes.uci.cu

Tutores:

Ing. Rodolfo Rodríguez Molinet.

Correo electronico: rmolinet@uci.cu

Ing. Yordi Chaveco Bustamante.

Correo electronico: ybistamante@uci.cu

El presente trabajo de diploma está dedicado a mi mamá que ha sido mi todo desde que nací, mami como tú no hay dos.

A mi hermana por darme el mayor regalo, mi sobrino. A mi hermano que, aunque no lo crea lo quiero cantidad.

A mis tíos Oriol y Rafael que dios los tenga en la gloria.

.

A mi familia, por estar en los momentos buenos y malos.

A mi papá por aconsejarme, te quiero con tus virtudes y con tus defectos.

A Deysi que es como una madre para mí y me apoyo cuando mi mamá no se encontraba.

A Zurama por ser como una tía para mí.

A mi cuñado por ayudarme en este proceso.

A mis profesores Rosalina, Dariela, Ariam, Leandro y Ana Marys, por su aporte para mi formación.

A mis tutores Rodolfo y Yordi por su gran apoyo.

A Claudia Bravo por su paciencia.

A mis compañeros de aula.

A todas aquellas personas que formaron parte de mi vida en la universidad Lientz, mi primo Leansi, Grisell, Pedro, Carlos, Yotsan, Claudia Salcedo, Ulises, Jan Carlos, Yoel, Diomne, Neysi, Yoandry, Karla, Rosmery, Diego, Joenis, Daymaris, Anayancy, Angelica, Osbel, Pucho, Norberto, Kutu, y a todos mis amigos que son muchos.

Resumen

El Sistema de Planificación de Actividades permite interrelacionar objetivos de trabajo y actividades en tiempo real; garantizando el seguimiento del desarrollo y cumplimiento de los objetivos y tareas principales en las entidades. Se encarga de la planificación a largo plazo, así como la planificación a mediano y corto plazo; lo que posibilita una mayor coincidencia entre lo que aspira la dirección y lo que debe proponerse cada miembro de la organización, garantizando que todo el personal se encuentre identificado con las tareas a desempeñar y las metas a alcanzar. Dentro del Sistema de Planificación de Actividades se encuentran los componentes Flujo de Trabajo y Configuración, los cuales están encargados de generar las configuraciones necesarias que otorgan al sistema y al cliente una simulación de los procesos de organización del personal.

La capa de presentación de los componentes Flujo de Trabajo y Configuración utiliza el marco de trabajo Ext JS en su versión 2.2. En la presente investigación se describen los procedimientos y actividades que permitieron realizar la migración de la capa a la versión 6.0 de Ext JS, con el objetivo de disminuir el consumo de recursos y el tiempo de carga de las interfaces de estos componentes. Durante el desarrollo de la migración se definieron tres etapas: inicio, implementación y prueba, quedando establecida la capa de presentación con la arquitectura Modelo Vista Controlador (MVC) que se define en la versión 6.0 de Ext JS, la ejecución de los procedimientos para realizar los cambios en el código fuente JavaScript y la realización de las pruebas de caja negra.

Palabras claves: arquitectura MVC, componentes, Ext JS, migración, planificación.

ABSTRACT

The Activities Planning System allows to interrelate work objectives and activities in real time; ensuring monitoring the development and implementation of the objectives and main tasks in entities. It is responsible for long-term planning, also for short and medium term planning; which enables a better match between the goals of the direction and what each member of the organization should aim, ensuring that all personnel are identified with the tasks to be performed and goals to achieve. Within the Activities Planning System are found the components Workflow and Configuration, which are responsible for generating the necessary configurations that give the customer and the system a simulation of the processes of staff organization. The presentation layer of components Workflow and Configuration uses the framework Ext JS in version 2.2, in the present investigation the procedures and activities that allowed the migration of the layer to version 6.0 of Ext JS are described, with the aim of reducing the consumption of resources and interfaces loading time of these components. During the development of migration three stages were defined: initial, implementation and testing, being established the presentation layer with Model View Controller (MVC) architecture as defined in version 6.0 of Ext JS, the implementation of procedures for changes in JavaScript source code and conducting black box tests.

Keywords: *architecture MVC, components, Ext JS, migration, planning.*

ÍNDICE

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Introducción

El creciente progreso de las Tecnologías de la Información y las Comunicaciones (TIC) ha facilitado el avance del proceso de desarrollo de software, trayendo consigo el aumento de disímiles herramientas que favorecen el trabajo de los desarrolladores. Dentro de ellas se destacan los marcos de trabajo, que son un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto (Velázquez Osorio et al. 2015).

El marco de trabajo Sauxe usa el modelo de desarrollo basado en componentes, integra varios estilos arquitectónicos, entre ellos el Modelo Vista Controlador (MVC) y en capas. Una de las capas definidas es la capa de presentación, la cual está formada por componentes, formularios y los controles que se encuentran en los formularios que implementan la capa de presentación, mostrando información y manejando las interacciones del usuario. Es la capa con la que interactúa el usuario.

Cuba está en inmersa en un proceso de informatización de entidades para facilitar el trabajo de algunos sectores estatales, donde la Universidad de las Ciencias Informáticas (UCI) tiene un papel protagónico en este proceso. La UCI desarrolló el Sistema de Planificación de Actividades (SIPAC) utilizando Sauxe en el Centro de Informatización de Entidades.

SIPAC permite interrelacionar objetivos de trabajo y actividades en tiempo real; garantizando el seguimiento del desarrollo y cumplimiento de los objetivos y tareas principales en las entidades. Es una solución que informatiza los procesos de Concepción y Ejecución para la planificación a largo plazo (planificación estratégica), así como para la planificación a mediano y corto plazo (planificación operativa y personal), lo que posibilita una mayor coincidencia entre lo que aspira la dirección y lo que debe proponerse cada miembro de la organización, garantizando que todo el personal se encuentre identificado con las tareas a desempeñar, las metas a alcanzar y las prioridades establecidas. Cuenta con varios módulos encargados de generar las configuraciones necesarias que otorgan al sistema y al cliente una simulación de los procesos de organización del personal, así como los niveles de subordinación entre ellos, para permitir que la información planificada sea accedida por la persona autorizada, en el momento indicado (Santamaria Pérez et al. 2015).

Dicho sistema cuenta con varios componentes, entre ellos se encuentran, Flujo de Trabajo que cuenta con las siguientes funcionalidades: estados y transiciones de la información y el componente Configuración que contiene las funcionalidades: grupos de usuarios por rol, nomencladores y permisos

para el trabajo con las configuraciones. A estos componentes serán asociados los tipos de documentos o elementos de la planificación (planes, objetivos y actividades).

Los componentes que implementa Ext JS v2.2 se encuentran en el fichero principal ext-all.js, lo que implica que a medida que se cargan las interfaces de algún componente, también se carga el fichero con todos los componentes de Ext JS, aunque no sean utilizados en la construcción de las interfaces, por lo que aumenta el tiempo de carga de las interfaces y el consumo de recursos del componente cargado.

Actualmente SIPAC está desarrollado en el marco de trabajo Sauxe, por lo que utiliza para la capa de presentación la versión 2.2 del marco de trabajo Ext JS; garantizando la comunicación con la lógica del negocio. Dicha versión fue liberada en agosto del 2008 y actualmente no cuenta con soporte por parte de los desarrolladores. Se realizó una prueba para calcular el tiempo de carga de las interfaces que se encuentran en la capa de presentación de los componentes Flujo de Trabajo y Configuración, con la utilización de la herramienta Firefox en su versión 42.0, obteniendo como resultado:

Tabla 1: consumo de recursos y tiempo de carga de las interfaces

Funcionalidades	Recursos (KB)	Tiempo (s)
Estados de la información	670	6.49
Transiciones de la información	107.2	6.93
Permisos	61.8	5.27
Nomencladores	503.9	6.83
Grupo de usuarios por rol	87.1	4.4
Total	1430	22.99

Según el autor Jacob Nielsen, existen 3 tipos de límites importantes en el tiempo de respuesta, por lo que estos tiempos mostrados en la tabla son altos:

0.1 segundos es el límite para que el usuario aprecie que el sistema reacciona instantáneamente al interactuar con él. La única información necesaria es la visualización de los resultados.

1.0 segundos es el límite para que el procesamiento de la información del usuario permanezca ininterrumpido, a pesar de que se dará cuenta de la demora.

10 segundos es el límite para mantener la atención del usuario, en otras palabras es el momento de abandono (Nielsen 2014).

Esta problemática trae consigo:

- ✓ Demora del tiempo de carga de las interfaces.
- ✓ Alto consumo de recursos.
- ✓ Se guardan en caché datos que no se van a reutilizar.
- ✓ Selección incorrecta del almacenamiento para caché.
- ✓ Asumir que un dato está en caché cuando realmente no lo está.
- ✓ Fallo al no desacoplar correctamente los módulos que conforman la aplicación.
- ✓ Se provocan fallos al diseñar para diferentes resoluciones de pantalla (pueden no mostrarse controles en los diferentes escenarios en el sistema).
- ✓ No permite crear componentes en tiempo de ejecución.

De la problemática anteriormente expuesta se ha identificado como **problema a resolver:** ¿Cómo aumentar el rendimiento de los componentes Flujo de Trabajo y Configuración del Sistema de Planificación de Actividades?

Se define como **objeto de estudio:** La migración en el proceso de desarrollo de software.

Proponiéndose como **campo de acción:** La migración de la capa de presentación del Sistema de Planificación de Actividades.

Para resolver el problema planteado se trazó el siguiente **objetivo general:** Migrar la capa de presentación de los componentes Flujo de Trabajo y Configuración del Sistema de Planificación de Actividades en la versión 6.0 de Ext JS para aumentar el rendimiento del mismo.

Desglosándose los siguientes **objetivos específicos:**

- ✓ Construir el Marco Teórico de la investigación relacionada con la versión 6.0 de Ext JS para identificar buenas prácticas al migrar a la misma.
- ✓ Desarrollar la migración de la capa de presentación de los componentes Flujo de Trabajo y Configuración a Ext JS 6.0 para aumentar su rendimiento.
- ✓ Validar mediante el uso de la herramienta JMeter el aumento del rendimiento de la capa de presentación de los componentes Flujo de Trabajo y Configuración del Sistema de Planificación de Actividades.

Para resolver el problema anteriormente planteado y dar cumplimiento a los objetivos trazados se proponen las siguientes **tareas de investigación:**

✓ Estudio del estado del arte de los marcos de trabajo para capa de presentación de aplicaciones web más utilizados en el mundo.

✓ Construcción del marco teórico referencial como resultado de la consulta de bibliografía nacional e internacional relacionada al proceso migración de la capa de presentación.

✓ Descripción de las técnicas, tecnologías, herramientas y los procedimientos de migración partiendo de buenas prácticas y patrones.

✓ Migración la capa de presentación.

✓ Validación de dicha migración.

Se traza como **Idea a defender:** Si se migra la capa de presentación de los componentes Flujo de Trabajo y Configuración del Sistema de Planificación de Actividades haciendo uso de la versión 6.0 de Ext JS se mejorará el rendimiento del mismo.

Se hace necesario utilizar los siguientes **métodos científicos** para realizar una adecuada investigación:

Métodos teóricos:

✓ **Analítico-sintético:** este método facilitó la extracción de elementos fundamentales relacionados con aplicaciones web y el Sistema de Planificación de Actividades, lo que contribuyó también a la sistematización de la información sobre el tema y la selección de los aspectos esenciales para la elaboración del estado del arte.

✓ **Histórico-lógico:** a través de este método se hizo un análisis de los marcos de trabajos encontrados a nivel mundial para el desarrollo de aplicaciones web. Obteniéndose de ellos elementos necesarios, como el diseño y la estructura para la implementación de la nueva aplicación.

Métodos Empíricos:

✓ **Entrevistas:** se aplica al cliente, así como a una pequeña muestra de los especialistas que trabaja en el departamento; con el objetivo de obtener información relacionada con la aplicación y las principales deficiencias que afectan el proceso, lo que contribuye a la obtención de la información.

Capítulo 1. Fundamentación teórica

1.1 Introducción

En el presente capítulo se muestra la base teórica en la que se basa la investigación. Se abordan varios conceptos asociados al proceso de migración. Se describen los marcos de trabajo más utilizados en el mundo para la capa de presentación de aplicaciones web, realizándose un estudio del estudio arte de los mismos. Se analizan un conjunto de herramientas y tecnologías utilizadas que posibilite el desarrollo de la migración con el fin de seleccionar las adecuadas.

1.2 Marco teórico

Durante la investigación fue necesario estudiar algunas definiciones para un mejor entendimiento y poder ampliar el conocimiento sobre el desarrollo del proceso de migración. Existen varios aspectos relacionados con términos tales como: migración, componente, entre otros que son necesarios para el proceso de migración de la capa de presentación.

1.2.1 Migración

Internet requiere que se mantengan los sistemas actualizados, los cuales han propiciado que los usuarios compartan rápidamente sus conocimientos retroalimentándose unos de otros continuamente. En la actualidad, las empresas más importantes de software actualizan sus productos por Internet, mediante una búsqueda automática o manual (accediendo a la web del producto), dependiendo del fabricante y siempre bajo el consentimiento del usuario. Por lo general, estas actualizaciones corrigen los errores que se detectan una vez lanzada la aplicación (Muñoz Tamayo, Aballi Morell y Yero Tarancón 2009).

Una de las acepciones del término hace referencia a la acción de convertir los programas de un lenguaje a otro. En una interpretación más abarcadora, se hace referencia a la migración de un sistema informático cuando se traslada de una plataforma a otra, lo que puede involucrar cambios de arquitectura y/o de tecnología, y normalmente lleva implícita la necesidad de reescribir los programas en un lenguaje diferente (Ciolli 2007).

Migrar es también elevar una versión de un producto software a otra de más alto nivel, o bien el movimiento de una arquitectura a otra. Es el proceso de mejora que tiene como objetivo aprovechar tecnologías más eficientes y responder a nuevos requerimientos de usuario o de software. Busca mejoras en el funcionamiento, la interoperabilidad, la actualización de versiones, la estandarización de la tecnología, nuevos procesos de negocio o mejoras en la seguridad y el control de la información. Es actualizar el software a una versión superior, reemplazando una versión instalada de un producto por

una versión más reciente del mismo producto (lawebdelprogramador.com 2013).

A continuación, se listan las ventajas y desventajas de la migración de tecnologías.

Ventajas:

- ✓ Aprovechar el uso de nuevas tecnologías.
- ✓ Disminuir costos altos en licencias.
- ✓ Disminuir los costos de mantenimiento y actualización.
- ✓ No cargar con los altos costos y riesgos de traslado a nuevas tecnologías o productos.
- ✓ Integrar sistemas de legado o existentes con nuevos sistemas.
- ✓ Desarrollo de nuevas funcionalidades que enriquecen el manejo de las herramientas.
- ✓ El soporte técnico actualizado sobre la versión actualizada es proporcionado por los proveedores de la herramienta.

Desventajas:

- ✓ El precio que se tiene que pagar para mantener actualizado un sistema.
- ✓ No todas las versiones nuevas que aparecen son compatibles con sus versiones anteriores.
- ✓ En muchos casos se estrecha más el tiempo entre una versión y otra (Rizzo 2011).

1.2.2 Rendimiento

Len Bass, Paul Clements y Rick Kazman afirman que el rendimiento define la capacidad de respuesta del sistema, es decir, el tiempo necesario para responder a estímulos (eventos) o el número de eventos procesados en un cierto intervalo de tiempo. Además, exponen que las cualidades de rendimiento se expresan a menudo mediante el número de transacciones por unidad de tiempo que procesa el sistema o por la cantidad de tiempo que se tarda en completar una transacción (Bass, Clements y Kazman 2012).

1.2.3 Interfaz gráfica de usuario

La interfaz gráfica de usuario, conocida también como Graphical User Interface (GUI, por sus siglas en inglés), es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal objetivo, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador (Webopedia 2015).

En el contexto del proceso de interacción persona-computadora, la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

1.2.4 Componente

Existen varias definiciones del término componente, la mayoría coincide en que estos son unidades de composición fundamentales de un sistema y que son independientes. Un componente es un elemento software que se ajusta a un modelo de componentes y que puede ser desplegado y compuesto de forma independiente sin modificación según un estándar de composición (Sridhar 2015).

Es una unidad de composición de aplicaciones software que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio (Szyperski, Bosch y Weck 1999).

1.2.5 Marco de Trabajo

El concepto marco de trabajo se emplea en muchos ámbitos del desarrollo de sistemas software, no solo en el ámbito de aplicaciones Web. Se puede encontrar marcos de trabajo para el desarrollo de aplicaciones médicas, para el desarrollo de juegos, y para cualquier ámbito. En general, el término marco de trabajo, hace referencia a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Un marco de trabajo se puede considerar como una aplicación genérica incompleta y configurable a la que se puede añadir las últimas piezas para construir una aplicación concreta. Los objetivos principales que persigue un marco de trabajo son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones (Gutiérrez 2014).

1.3 Estado del arte

Se realiza un estudio del estado del arte de los marcos de trabajo para la capa de presentación de aplicaciones web más utilizados en el mundo, y para ello se tienen en cuenta las características asociadas a cada uno de ellos. Además, se expone la metodología que guiará todo el proceso de desarrollo del software, así como las herramientas, tecnologías y métodos empleados en el desarrollo de la migración de la capa de presentación de SIPAC.

1.3.1 Ext JS

Ext JS es una librería JavaScript que permite construir aplicaciones complejas en Internet. Es desarrollado por Sencha, es orientado a objeto, además utilizan patrones de diseño y lo más importante es la versatilidad de heredar de sus componentes básicos para crear componentes personalizados.

Esta librería incluye:

- ✓ Modelo de componentes extensibles.
- ✓ Un Application Programming Interface [1](API, por sus siglas en inglés) fácil de usar.
- ✓ Licencias código abierto, comercial.

Otro punto a tener en cuenta son las licencias; tiene dos tipos:

- ✓ Licencia comercial: esta licencia se debe comprar cuando se necesite desarrollar software propietario.
- ✓ Licencia de código abierto: este tipo de licencia se aplica cuando se desea desarrollar un proyecto de código abierto, esto implica liberar el proyecto con licencia General Public Licensem [2](GNU, por sus siglas en inglés) (Sencha 2012).

Esta tabla muestra los plazos oficiales de soporte para los productos Sencha. El soporte extendido para cada versión principal se vende por separado y debe adquirirse para todas las licencias en poder de la organización.

Tabla 2: Tabla de versiones

Versión	Fecha de lanzamiento	Política	Final de soporte estándar	Soporte extendido por compra	Fin del soporte extendido
5.x	6/1/2014	Soporta hasta 2 años después de la próxima versión, además de 2 años. Soporte extendido.	7/1/2017	7/1/2018	7/1/2019
4.x	4/26/2011	Soporta hasta 2,5 años después de la próxima versión principal, además de 2 años. Soporte extendido.	12/31/2016	12/31/2017	12/31/2018
3.x	7/1/2009	Soporta hasta 2 años después de la próxima versión, más de 5 años. Soporte extendido.	4/25/2013	5/31/2017	5/31/2018

[1] Application Programming Interface (API, por sus siglas en inglés): Conjunto de subrutinas, funciones y procedimiento que ofrece biblioteca para ser utilizado por otro software como una capa de abstracción.

[2] GNU (General Public License): Licencia que permite modificar, copiar y compartir un software.

1.3.2 Librería Prototype

Prototype es una librería que tiene como objetivo facilitar el desarrollo de aplicaciones web dinámicas. Ofrece un amplio soporte, construcciones de programación de orden superior, y la manipulación DOM fácil. El cambio en el mundo de la programación web, hace uso de las ventajas de la web 2.0. Con este cambio las técnicas de desarrollo de páginas web necesitaban dar un gran salto. Prototype brinda su aporte a este auge, para lograr mejores servicios a menor costo de desarrollo. Su desventaja es el consumo de memoria, pero hay opciones muy buenas para comprimirlas.

El potencial de Prototype es aprovechado al máximo si se desarrolla con el marco de trabajo Ruby on Rails, esto no quiere decir que no se puede usar desde otro lenguaje, solamente que demandará un mayor esfuerzo en el desarrollo. Prototype es una librería JavaScript que apunta al desarrollo sencillo y dinámico de aplicaciones web. Es una herramienta para el desarrollo de clases única y de fácil uso, además de ser la biblioteca más agradable de AJAX.

La parte más grande de la librería Prototype son sus extensiones de Document Object Model[3] (DOM, por sus siglas en inglés). Prototype agrega muchos métodos de conveniencia. Todos los elementos de DOM tienen los métodos de extensión de Prototype incorporado (prototypejs.org 2011).

1.3.3 Librería JQuery

Es una librería de JavaScript, rápida y concisa que simplifica el trabajo con documentos HyperText Markup Language[4] (HTML, por sus siglas en inglés). JQuery ha sido diseñado para cambiar la forma de escribir JavaScript. Utiliza un interesante concepto para hacer código corto y simple, tiene manejadores de eventos. Otro tema que JQuery resuelve con facilidad es el de los efectos, añade dinamismo visual a la presentación del sitio, funcionalidades, código y al resto de los elementos. Los métodos de JQuery se requieren para colocar automáticamente todos los elementos de DOM en el código, y aplicar el método deseado. Se elimina la iteración en el código (en la mayoría de los casos), esta es una de las ventajas prácticas, cotidianas y superiores a usar en JQuery. Dada la madurez que ha adquirido esta librería, es más fácil construir plugins a partir de una estructura ya existente, permitiendo así que se elimine prácticamente toda la iteración molesta.

Generalmente, cuando se trabaja con JavaScript el código no compila hasta tanto no se hayan cargado las imágenes, incluyendo los banners. Para solucionar este problema JQuery ha implementado un procedimiento que se puede utilizar, conociendo como ready.event, este código chequea el documento y aguarda a que esté listo para manipularlo (jquery.org 2012).

[3] DOM (Document Object Model): Interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML y XML.

[4] HTML (HyperText Markup Language): Se refiere a un lenguaje de marcado para la elaboración de páginas web.

1.3.4 Mootools

Es un conjunto de librerías, también llamado API, que proveen clases de programación orientada a objetos en JavaScript, para realizar una amplia gama de funcionalidades en páginas web, como trabajo con capas, efectos diversos, AJAX y mucho más. Con Mootools se puede programar todo tipo de scripts en el lado del cliente rápidamente y sin preocuparse de las distintas particularidades de cada navegador. Está especialmente indicado para programar scripts complejos, que costarían mucho más trabajo realizarlos si se comenzará de cero.

Ventajas:

- ✓ Ligero: el marco de trabajo no pesa demasiado en KB y el procesamiento carga poco al navegador.
- ✓ Modular: se compone de diversos módulos y el desarrollador puede seleccionar los que va a utilizar para incorporarlos en sus páginas web, dejando los otros para que no ocupen tiempo de descarga ni procesamiento.
- ✓ Libre de errores: las herramientas de Mootools funcionan perfectamente, sin emitir errores en tiempo de ejecución.
- ✓ Soportado por una amplia comunidad: existen muchos desarrolladores que lo utilizan con éxito y han creado una serie de componentes adicionales ya listos para usar, como calendarios y editores de texto.

Desventajas:

- ✓ La escasa documentación del mismo, pues la que existe es buena pero no se puede encontrar la suficiente que un desarrollador necesita.
- ✓ No se encuentran muchos ejemplos.
- ✓ Dificultad para trabajar con el marco de trabajo resulta complicado porque los ejemplos que existen del mismo son complejos (mootools.net 2014).

1.3.5 Dojo

Está compuesto por Widgets (controles) que son componentes de código en Javascript pre-empaquetados que puede ser utilizados para enriquecer sitios web con varias características que trabajan a través de la mayoría de los navegadores. Es un framework que contiene APIs para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX. También es posible inicializar paquetes adicionales dentro o al mismo nivel que el paquete dojo, permitiendo extensiones o bibliotecas de terceros.

Los paquetes de Dojo pueden contener múltiples archivos. Cualquier paquete o archivo puede depender de otro. En este caso, cuando el paquete es cargado, cualquier dependencia será también

cargada. Proporciona una gama más amplia de opciones en una sola biblioteca JavaScript y es compatible con navegadores antiguos.

Es de destacar que esta biblioteca es de código abierto y se puede descargar de forma gratuita en su página oficial. La licencia nos permite crear aplicaciones, utilizarlo en productos comerciales y modificarlo. Cuenta con una gran comunidad de desarrolladores e información que la hacen muy accesible (dojotoolkit.org 2011).

1.3.6 Análisis de las tecnologías

Se realiza un análisis de las librerías o marcos de trabajo más utilizados en el mundo para el desarrollo de aplicaciones web, para seleccionar cuál es la adecuada para dar respuesta a la solución planteada. Es necesario tener en cuenta que actualmente en el marco de trabajo Sauxe, se está redefiniendo la utilización del marco de trabajo Ext JS en su versión 6.0 en la capa de presentación. La mayoría de los especialistas que desarrollan en Sauxe, tienen conocimientos sobre Ext JS y existen componentes ya construidos en la versión 6.0 con la arquitectura MVC en Sauxe.

JQuery ha tenido gran éxito, debido a que es fácil de usar y de entender, además de que existe en internet una gran cantidad de documentación y plugins que extienden su funcionalidad. Ha sido muy utilizado por diseñadores con pocos conocimientos en programación. Prototype, se integra bastante bien con Ruby On Rails, lo cual ha beneficiado a desarrolladores independientes y empresas al momento de crear aplicaciones de vanguardia.

Dojo Toolkit, en su incorporación con Zend Framework promete mucho, pero no goza de la popularidad de MooTools, que, aunque es liviano y orientado a objetos, en algunas ocasiones resulta complicado realizar simples acciones, que utilizando Prototype suelen ser bastantes sencillas, como peticiones periódicas y acceso a diferentes elementos.

Para crear una aplicación web con estilo de aplicación web, Ext JS es la mejor opción, inclusive se podría crear un Sistema Operativo Web utilizando este marco de trabajo de forma relativamente fácil. Lo importante es tener conocimientos de las herramientas con las que se cuentan y sacarle el máximo provecho (Tavárez 2009).

Además, mencionar que Ext JS brinda una arquitectura MVC que permite organizar, reducir el código fuente de la aplicación y tener mejor control de los componentes de la librería. Es el marco de trabajo con mayores funcionalidades y ventajas para el desarrollo aplicaciones web.

Por lo antes mencionado se selecciona Ext JS en la versión 6.0 como marco de trabajo, para dar solución a la situación problemática planeada.

1.4 Herramientas y tecnologías

Para llevar a cabo el proceso de migración, al igual que todo proceso de desarrollo de software, es necesario definir cuáles herramientas, y tecnologías serán utilizadas.

1.4.1 Sauxe v2.3

Es un marco de trabajo, fusionado bajo tecnologías totalmente libres (entre ellas PHP, Postgresql, Apache) que posee el desarrollo de tecnologías propias basadas en otros marcos de trabajo como ZendFramework para el manejo de la lógica de negocio, Doctrine para el acceso a datos y Ext JS para la capa de presentación. Cuenta con una arquitectura en capas que a su vez presenta en su capa superior un MVC. Contiene un conjunto de componentes reutilizables que provee una estructura genérica y un comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo.

1.4.2 Ext JS v6.0

Introduce un marco único para la creación de aplicaciones que se ejecutan en todo tipo de dispositivos, desde teléfonos, tabletas u ordenadores de escritorio. Produce una experiencia de usuario óptima al escribir menos código. El proceso de fusión Ext JS y SenchaTouch[5] ha recorrido un largo camino. En Ext JS 5, Sencha ha reconciliado el centro de sus marcos en el paquete de "núcleo". La capa visual se mantuvo como una parte propias de Ext JS en el paquete "ext". El paso final de la fusión de los componentes visuales SenchaTouch requiere un sitio para estos distintos aspectos del marco de trabajo. Para diferenciar las familias de componentes entre sí en Ext JS 6.0, se utiliza una caja de herramientas. La misma es un paquete que contiene sólo los elementos visuales de la estructura. Los cuales incluyen componentes como paneles, botones, rejillas y entre otros. Hay dos juegos de herramientas en Ext JS 6: clásicos y modernos. Los elementos visuales de Ext JS están ahora incluidos en el kit de herramientas clásica de Ext JS 6, mientras que los elementos visuales de SenchaTouch están contenidos en la caja de herramientas modernas (docs.sencha.com 2012a).

1.4.3 PostgreSQL v9.4

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia Berkeley Software Distribution [6](BSD, por sus siglas en inglés) para bases de datos y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones ha logrado incorporar las potencialidades de otros sistemas

[5] SenchaTouch: Marco de trabajo multi-plataforma para aplicación web móvil basado en HTML5 y JavaScript para crear aplicaciones móviles universales.

[6] BSD (Berkeley Software Distribution): Sistema operativo derivado del sistema Unix desarrollado a partir de los aportes realizados a ese sistema por la Universidad de California en Berkeley.

gestores de bases de datos incluso comerciales. PostgreSQL utiliza un modelo cliente-servidor y usa multiprocesos en vez de multihilos para lograr la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (postgresql.org 2010).

Dentro de las características que más se evidencian son: estabilidad, potencia, robustez, facilidad de administración e implementación de estándares. Este presenta un rendimiento elevado con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

Soporta distintos tipos de datos: tipo fecha, monetarios, elementos gráficos, datos sobre la red, cadenas de bits y permite la creación de tipos propios. Incluye herencia entre tablas, posee múltiples métodos de autentificación, contiene documentación completa y es multiplataforma (Kasián y Reyes 2012).

1.4.4 Apache v2.4.9

Se utilizará como servidor web Apache 2.4.9 pues es una tecnología gratuita de código abierto, presenta compatibilidad con muchos Sistemas Operativos. Apache sigue siendo desarrollado por la comunidad de usuarios desarrolladores que trabaja bajo la tutela de Apache Software Foundation. Cuenta con el soporte necesario para tener páginas dinámicas. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Posibilita configurar la creación y gestión de registros de actividad. Apache permite la creación de ficheros de registro a la medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor (apache.org 2012).

1.4.5 Subversion

Subversion es un sistema de control de versiones libre (código abierto). Maneja ficheros y directorios a través del tiempo; un árbol de ficheros en un repositorio central y el repositorio realiza la función de un servidor de ficheros ordinario, exceptuando que guarda todos los cambios hechos a sus ficheros y directorios. Esto permite recuperar versiones antiguas de los datos, o examinar la historia de cómo cambiaron sus datos. Puede acceder al repositorio a través de redes, lo que permite que pueda ser utilizado por personas en diferentes equipos. A cierto nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones (subversion.apache.org 2015).

1.5 Pilares para ejecutar el proceso de migración

Una migración debe apoyarse en tres pilares básicos (Ciolli 2007):

- ✓ Una metodología.
- ✓ Un conjunto de herramientas.
- ✓ Técnicas de pruebas y personalización.

La metodología garantiza, en primer lugar, un procedimiento sistemático que asegura que el trabajo realizado sea controlable y sus resultados predecibles. En segundo lugar, que se dispone de un repositorio con toda la información necesaria para abordar la migración: cadenas de programas, programas fuente, estructura de base de datos y librerías de funciones. En tercer lugar, contempla la obtención del modelo de negocio a migrar, a partir de la información contenida en el repositorio, y considera además la realización de los planes de prueba de las aplicaciones migradas. Por último, define las reglas de generación del código migrado, conforme a los estándares establecidos, las librerías de funciones usadas y cualquier otra consideración de interés.

Las herramientas de migración permiten la obtención un modelo del negocio a migrar, que lo hace independiente de los lenguajes de las aplicaciones, con lo cual el modelo obtenido resultará válido en caso de futuras migraciones a otras tecnologías. Estas herramientas deben permitir, también, la incorporación de las reglas básicas del negocio a los efectos de obtener aplicaciones optimizadas para su funcionamiento en el entorno informático existente en una empresa.

Las técnicas de pruebas y personalización que incorporan las reglas de generación introducidas por la metodología, para obtener aplicaciones fiables, funcionales y operativas, que optimizan su funcionamiento en el entorno informático existente en la empresa. El uso de estos pilares permite asegurar el éxito del proyecto, manteniendo los plazos y costos de realización dentro de las previsiones.

1.5.1 Migración de la capa de acceso a datos del marco de trabajo Sauxe

El proceso de desarrollo de software es difícil de controlar, surge entonces la necesidad de tener una metodología o un conjunto de procedimientos que garanticen cumplir con los planes de producción del software y la satisfacción del cliente con el cumplimiento de los objetivos del proyecto. Para la realización del proceso de desarrollo se basará en los procedimientos definidos en el artículo científico Procedimientos para la actualización de la Capa de Acceso a Datos del Marco de Trabajo Sauxe de Doctrine 1.2.2 a Doctrine 2.0 (Velázquez Osorio et al. 2015), el cual cuenta con tres etapas que se describen a continuación:

✓ Etapa Inicial: se realiza un análisis de la arquitectura de la aplicación para evaluar el contenido y el funcionamiento interno del sistema. Se estudia la estructura de carpetas que contiene la capa de acceso a datos utilizando Doctrine 1.2.2.

✓ Etapa de Implementación: se garantiza la conectividad a la base de datos y los parámetros de configuración. Se implementa el código fuente del sistema.

✓ Etapa de prueba: tiene como objetivo supervisar que la implementación se realizó satisfactoriamente probando todas las funcionalidades migradas.

1.5.2 Guía del proceso de migración del Portal Octavitos

El proceso de migración del Portal Octavitos se divide en tres fases o etapas fundamentales: Preparación, Migración y Consolidación (Orosa Velázquez y Rodríguez Rodríguez 2011).

✓ Preparación o Planificación: fase o etapa previa al desarrollo del proceso de migración, cuyo contenido dependerá de la guía en estudio. Se centra en el estudio de factibilidad, la organización de los recursos humanos y la planificación de las actividades.

✓ Migración o Ejecución: fase o etapa en la que se acomete como tal el proceso de migración y cuyo contenido dependerá de la guía en estudio. Constituye el núcleo del proceso, y por tanto es la etapa de mayor complejidad, esfuerzo y organización; el tiempo y la calidad del trabajo dependen casi totalmente de la Preparación.

✓ Consolidación o Evaluación: fase o etapa post-migración cuyo contenido dependerá de la guía en estudio. Garantiza los servicios de soporte y mantenimiento, así como la evaluación del proceso.

1.5.3 Análisis de las bibliografías consultadas

Se efectuó un análisis de las migraciones de tecnologías que se proponen en el desarrollo de la investigación, las cuales definen etapas y procedimientos que se tienen en cuenta para desarrollar la migración de la capa de presentación del componente Configuración en la versión 6.0 de Ext JS, como se muestra en la tabla.

Tabla 3: Metodologías

Migración	Etapas	Procedimientos
Migración de la capa de acceso a datos del marco de trabajo Sauxe	Inicial, Implementación y Prueba	✓ Analizar la arquitectura del marco de trabajo Sauxe y la estructura del componente donde se van a realizar los cambios de tecnología. ✓ Implementar el código fuente de la capa definida. ✓ Evaluar que la implementación se realizó satisfactoriamente probando todas las funcionalidades migradas.
Guía del proceso de migración del Portal Octavitos	Consolidación o Evaluación	Garantizar que los objetivos de la migración fueron cumplidos.
Guía de implementación con	Implementación	✓ Analizar la arquitectura de SIPAC.

Ext JS 6.0 aplicando la arquitectura MVC en la capa de presentación del marco de trabajo Sauxe.		✓ Definir la nueva estructura de directorio para introducir la arquitectura MVC definida por Ext JS en su versión 6.0 ✓ Configurar los ficheros de extensión phtml para importar los ficheros principales de la librería de Ext JS. ✓ Realizar la instancia a la clase Aplicación que contiene la configuración global. ✓ Definir los controladores que permiten gestionar los eventos de las vistas. ✓ Definir las vistas que representan los componentes definidos por Ext JS. ✓ Definir los almacenes que hacen referencia a todos los modelos del componente. ✓ Aplicar pruebas de caja negra para encontrar errores en las interfaces.

1.6 Conclusiones parciales

Como resultado de la fundamentación teórica de la presente investigación, luego de realizar un análisis de los principales conceptos asociados al dominio del problema fue posible:

✓ Analizar y resumir los términos de gran envergadura referentes a la investigación, lo cual brindó un mayor conocimiento de las características y repercusión que tiene la migración de una tecnología menor a una mayor.

✓ Estructurar el código fuente que será implementado durante el desarrollo de la solución a partir de la arquitectura MVC en Ext JS.

✓ Sentar las bases para el desarrollo de la solución a partir de la selección de las principales herramientas, tecnologías y metodología de desarrollo de software.

Capítulo 2. Descripción de la propuesta

2.1 Introducción

En el presente capítulo se describe el proceso de migración de la capa de presentación de los componentes Flujo de Trabajo y Configuración del Sistema de Planificación de Actividades a la versión 6.0 de Ext JS. En el mismo, se establece la arquitectura Modelo Vista Controlador de Ext JS para los componentes de SIPAC. Además, se describen los cambios realizados entre las versiones 2.2 y 6.0 de Ext JS.

2.1. Estrategias de migración

Las estrategias de migración reconocen los dos enfoques siguientes:

- ✓ Habilitación gradual

La nueva aplicación es construida gradualmente en la plataforma de destino, haciéndose cargo en forma progresiva de las funcionalidades de la aplicación original. En este proceso ambas aplicaciones están integradas en un único sistema con una transferencia gradual de responsabilidades de una a otra. Con este enfoque la información está duplicada y es necesario un importante esfuerzo de coordinación para asegurar la integridad y consistencia de los datos.

- ✓ Habilitación súbita

La aplicación original mantiene todas sus prestaciones mientras la aplicación en la nueva plataforma es construida, implementada y probada. Las bases de datos de esta última son progresivamente actualizadas hasta el momento en que se decide la transferencia del control, momento en que la aplicación original queda desafectada y sus bases de datos quedan como referencia únicamente para consulta (Ciolli 2007).

De las estrategias antes mencionadas se utiliza la estrategia habilitación súbita.

2.2. Desarrollo de la migración de la capa de presentación

La migración de la capa de presentación de los componentes Flujo de Trabajo y Configuración de SIPAC, para la versión 6.0 de Ext JS se realizó de forma manual por los siguientes motivos: el código fuente JavaScript se encuentra en un solo fichero js, estructurado e implementado de diferentes formas sin un estándar, Ext JS en su versión 6.0 realiza cambios significativos con respecto a la versión 2.2 porque incluye una refactorización de todo el marco de trabajo Ext JS, realiza una nueva estructura de clases y carga dinámica de objetos, paquetes de datos, nuevos gráficos y temas, que se deben tener en cuenta en los cambios de las versiones lo que complejiza el proceso de migración, aunque se

destaca que se puede reutilizar el código fuente en la versión 2.2, siendo una ventaja para la realización de la migración de forma manual.

Se desea mantener el diseño de las interfaces con las mismas características y funcionalidades de la versión 2.2, solamente el cambio se desarrollará a nivel de código fuente JavaScript, que se encuentra en la capa de presentación de los componentes Flujo de Trabajo y Configuración.

2.3. Fases de la migración

La migración de la capa de presentación de los componentes Flujo de Trabajo y Configuración de SIPAC, se desarrolla en tres etapas: inicial, implementación y prueba.

- ✓ **Inicial:** se realiza un análisis de la arquitectura de SIPAC y de la estructura de los componentes Flujo de Trabajo y Configuración, se identifican los requisitos de instalación del marco de trabajo Ext JS, se importan los ficheros principales que propone Ext JS en su versión 6.0, se realizan los cambios pertinentes en la estructura de la capa de presentación de los componentes Flujo de Trabajo y Configuración siguiendo la arquitectura MVC que define Ext JS en su versión 6.0.
- ✓ **Implementación:** se describen los procedimientos para realizar la migración de la capa de presentación de los componentes Flujo de Trabajo y Configuración para la versión 6.0 de Ext JS.
- ✓ **Prueba:** se ejecutan las pruebas de caja negra con el objetivo detectar errores en las interfaces de usuarios construidas utilizando la versión 6.0 de Ext JS.

2.3.1 Arquitectura del SIPAC

El análisis de la arquitectura del SIPAC es la primera tarea a realizar en la etapa inicial. El sistema está guiado por el modelo de desarrollo basado en componentes, presenta un estilo arquitectónico en capas, compuesto por cinco niveles o capas e implementa el patrón arquitectónico Modelo Vista Controlador. La propuesta de solución se centra específicamente en la capa de presentación.

- ✓ **Capa de Presentación:** en esta capa se emplean las facilidades que brinda el marco de trabajo Ext JS, dígase la biblioteca de JavaScript para la construcción de interfaces a la vista de los usuarios. Ext JS centra su desarrollo en tres componentes fundamentales JS-File, CSS-File y Client-page y Server-Page.
- ✓ **Capa de Control o Negocio:** en esta capa se emplea el patrón MVC. De forma vertical al modelo descrito hasta este momento, estarán los aspectos fundamentales del sistema; así como el tratamiento de la seguridad a nivel de aplicación web.
- ✓ **Capa de Acceso a Datos:** en esta capa estará presente el Object-Relational Mapping [7](ORM, por sus siglas en inglés) Doctrine para la comunicación con el servidor de datos mediante el protocolo PHP

[7] ORM (Object-Relational mapping): Técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

DataObjects[8] (PDO, por sus siglas en inglés), también estará un Persistidor de Configuración que es el encargado de comunicarse vía XML con los Ficheros de Configuración del sistema denominado FastResponse.

- ✓ **Capa de Datos:** En esta capa estará ubicado como servidor de base de datos PostgreSQL y un conjunto de Ficheros de Configuración de la arquitectura tecnológica.
- ✓ **Capa de Servicio:** En esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí.

2.5. Requisitos de instalación de Ext JS 6.0

Navegadores web:

Ext JS es compatible con todos los navegadores web, desde Internet Explore 6 a la última versión de Google Chrome., sin embargo, se recomienda utilizar los siguientes navegadores para una mejor compilación del mismo:

Google Chrome 10+

Apple Safari 5+

Mozilla Firefox 4+

Servidores web:

Se recomienda que se utilice como servidor web la herramienta Apache HTTP Server (docs.sencha.com 2012b).

2.6. Estructura de directorio de Ext JS 6.0.

Teniendo en cuenta la arquitectura Modelo Vista Controlador definida por Ext JS en su versión 6.0 el primer cambio se efectuará en la estructura de directorio de los componentes Flujo de Trabajo y Configuración. Para la versión 2.2 de Ext JS en la figura 1 se muestra el fichero gestionarestado.js, en el cual se localiza todo el código JavaScript de Ext JS; mientras que en la versión 6.0 todas las clases se colocan en el directorio app, que contiene a su vez todos los directorios modelos (model), vistas (view), controlador (controller) y almacenes (stores).

[8] **PDO (**PHP DataObjects): Extensión que provee una capa de abstracción de acceso a datos para PHP5.

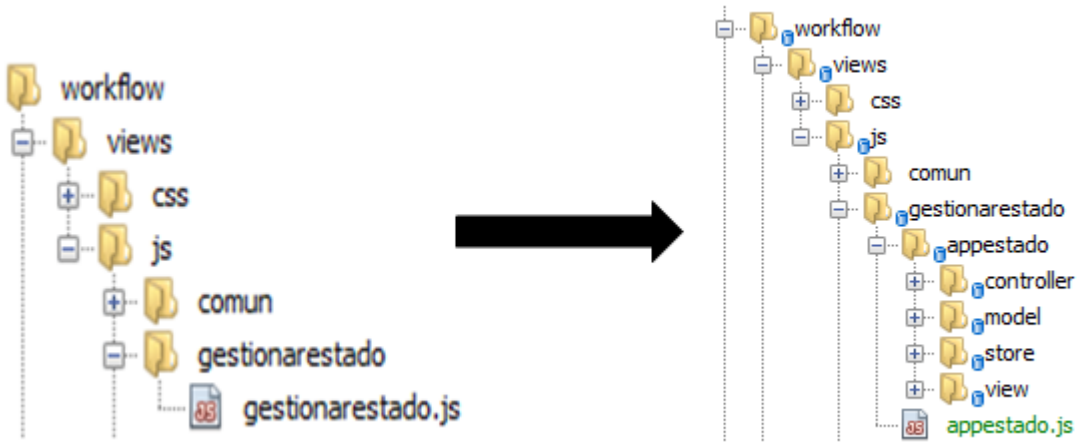


Figura 1: Estructura de directorio.

2.7. Etapa de implementación

En la presente etapa se describirán los pasos que se seguirán para llevar a cabo la migración de la capa de presentación de los componentes Flujo de Trabajo y Configuración.

2.7.1. Configuración del fichero phtml

Al importar los archivos principales del marco de trabajo Ext JS, se configura el fichero de extensión phtml, a través del<link href />se sustituye el archivo ../default/css/ext-all.css por el archivo css/ext-all.css y a través de <script src></script>se sustituyen los archivos ext-base.js y ext-all.js por el archivo ext-all-6.js, en la figura 2 se muestra un ejemplo del código fuente del archivo gestionarestado.phtml del componente Flujo de Trabajo.

```
<html>
<head>
<title>Gestionar Estado</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="<?php echo $this->dir_ext_ccs; ?>css/ext-all.css" rel="stylesheet" type="text/css" media="screen"/>
<script type="text/javascript" src="<?php echo $this->dir_ucid;?>js/imporcss.js"></script>
<link rel="stylesheet" type="text/css" href="../../views/css/gestionarflujodetrabajo/icons.css" />
</head>
<body>
<script type="text/javascript" src="<?php echo $this->dir_extjs; ?>js/ext-all-6.js"></script>
<script type="text/javascript" src="<?php echo $this->dir_ucid;?>js/ucid-all.js"></script>
<script type="text/javascript" src="../../../../../planificacion/pdo/comun/js/CmpBase.js"></script>
<script type="text/javascript" src="../../../../../planificacion/pdo/comun/js/CmpBarraProgreso.js"></script>
<script type="text/javascript" src="../../views/js/gestionarflujodetrabajo/CheckColumn.js"></script>
<script type="text/javascript" src="../../views/js/gestionarestado/appestado.js"></script>
</body>
</html>
```

Figura 2: Fichero de extensión phtml del componente Flujo de Trabajo.

Durante la ejecución de este procedimiento se configuraron los archivos de extensión phtml que se muestran a continuación:

Del componente Flujo de Trabajo:

- ✓ gestionarestado.phtml.
- ✓ gestionarflujodetrabajo.phtml.

Del componente Configuración:

- ✓ gestionargrupos.phtml.
- ✓ gestionarnomencladores.phtml.
- ✓ gestionarpermisos.phtml.

2.7.2. Crear la instancia de la clase Application

La instancia de Application contiene la configuración global del componente (por ejemplo, el nombre del componente), así como mantiene referencias a todos los modelos, vistas, controladores y almacenes creados por este. En la figura 3 se muestra la configuración del archivo app.js del componente Flujo de Trabajo, en el cual se crea la instancia de la Application con el espacio de nombres global'Gestestado'.

```
var perfil = window.parent.UCID.portal.perfil
perfil.etiquetas = Object();

Interface = {
    init: function() {
        Ext.Loader.setPath({
            'Gestestado.controller': '../../views/js/gestionarestado/appestado/controller',
            'Gestestado.view': '../../views/js/gestionarestado/appestado/view',
            'Gestestado.store': '../../views/js/gestionarestado/appestado/store',
            'Gestestado.model': '../../views/js/gestionarestado/appestado/model'

        });
        Ext.application({
            name: 'Gestestado',
            autoCreateViewport: true,
            models: ['GridEstadoModel'],
            stores: ['GridEstadoStore'],
            controllers: ['EstadosController']
        });

    }
};

UCID.portal.cargarEtiquetas('gestionarestado', Interface.init);
```

Figura 3: Configuración del archivo appestado.js

Durante el desarrollo de este procedimiento se crearon las instancias de las clases Application siguientes:

Del componente Flujo de Trabajo:

- ✓ Gestestado.
- ✓ Gestflujodetrabajo.

Del componente Configuración:

- ✓ Gestgrupos.
- ✓ Gestnomenclador.
- ✓ Gestpermisos.

2.7.3. Definir una vista

Para crear una vista siguiendo la arquitectura Modelo Vista Controlador que propone Ext JS 6.0, se crea un fichero de extensión js y se define que sea de tipo view, extendiendo de un componente de Ext JS, por ejemplo en el caso del componente Viewport, quedaría Ext.container.Viewport, en el caso del componente GridPanel sería Ext.grid.GridPanel, en la figura 4 se muestra un ejemplo del código fuente de la vista Viewport que contiene las restantes vistas del componente Configuración y en la figura 5 se muestra la interfaz de usuario.

```
Ext.define('Gestpermisos.view.Viewport', {
    extend: 'Ext.container.Viewport',
    layout: 'fit',
    idItem: 'view',
    requires: [
        'Gestpermisos.view.TabPanel',
        'Gestpermisos.view.PanelPermisos',
        'Gestpermisos.view.GridPermOtorgados',
        'Gestpermisos.view.PanelSubordinados',
        'Gestpermisos.view.PanelOtorgados',
        'Gestpermisos.view.GridPermisos',
        'Gestpermisos.view.PanelArbol'
    ],
    initComponent: function() {
        this.items = {
            xtype: 'tabpanelgeneral'
        };

        this.callParent();
    }
});
```

Figura 4:Código fuente de la vista Viewport.

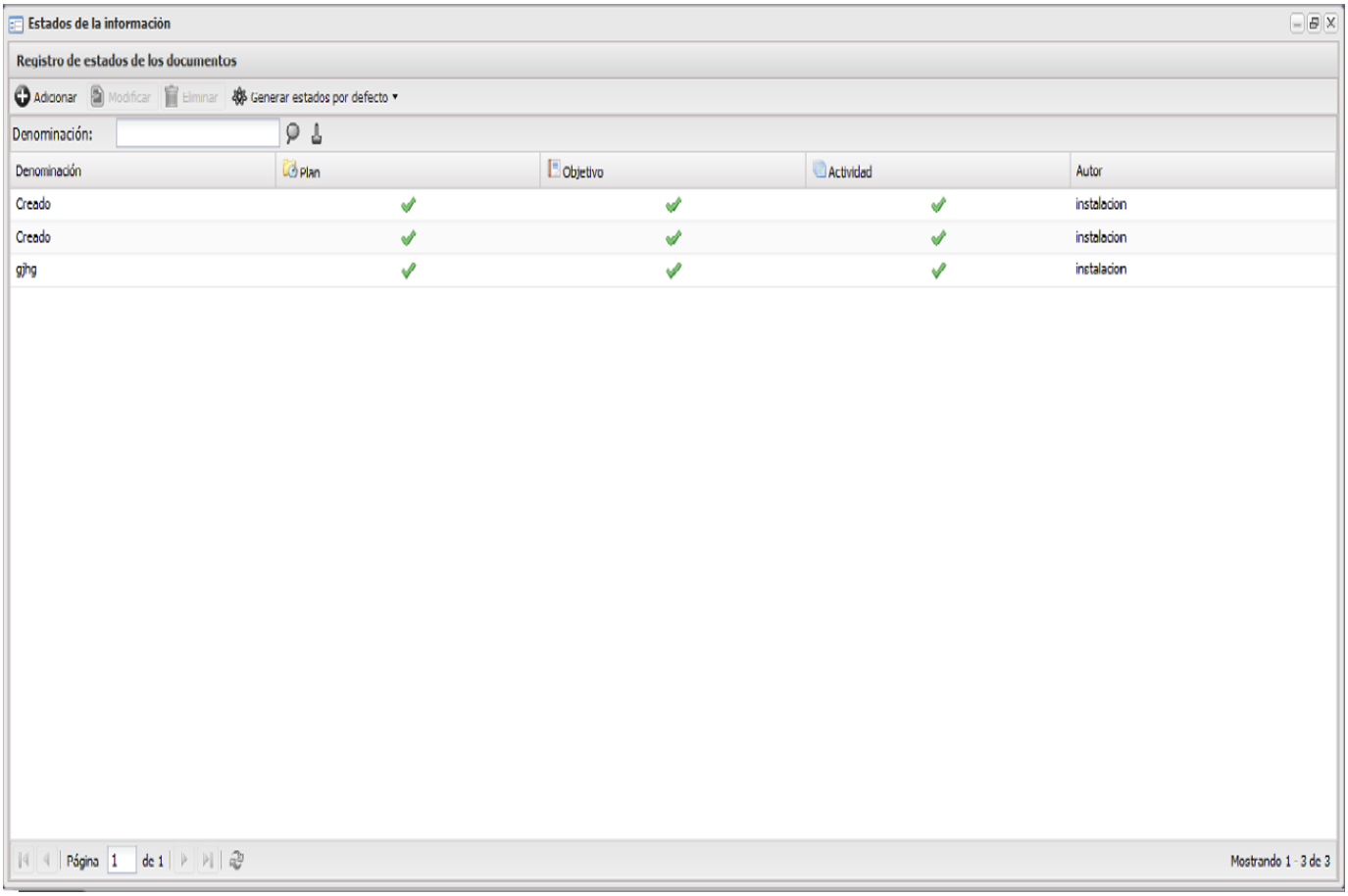


Figura 5: Interfaz de usuario del componente Flujo de Trabajo.

Se definieron las siguientes vistas durante el desarrollo de este procedimiento:

Tabla 4: Vistas

Funcionalidades	Vista	Total
Estados de la información	Viewport, PanelPrincipal, GridEstadoWindowAdd, WindowMod, FormModEstado, FormEstado	7
Transiciones de la información	Viewport, PanelPrincipal, PanelBusqueda, GridFlujodeTrabajo, Formulario, comboBoxFinal, comboBoxInicial.	7
Permisos	ArbolPermiSobre, ArbolPermisoPara, GridPermOtorgados, GridPermisos, PanelArbol, PanelOtorgados, PanelPermisos, PanelSubordinados, TabPanel, Viewport, WinDarPermiso, WinModPermiso, SmGridPermisos, SmPanelOtorgados.	14
Nomencladores	PanelArbol, Arbol, PanelCentral, Viewport,	4

Grupo de usuarios por rol	ArbolEntidades, ArbolGrupo, FormEstado, FormWinGestGrupos, GridGrupo, GridUsuarios, PanelGestGrupos, PanelPrincipal, PanelTab1, PanelTab2, TabWinGestGrupos, Viewport, WinGestGrupos, ComboBoxActivo.	14

2.7.4. Definir un almacén

El almacén hace referencia a todos los modelos que se creen dentro de un componente definido por Ext JS. Para crear un almacén siguiendo la arquitectura Modelo Vista Controlador que propone Ext JS v6.0, se debe crear un archivo de extensión js, luego se define que el fichero sea de tipo store extendiendo de Ext.data. Store para el almacén del grid y Ext.data. TreeStore para el almacén del árbol. Aquí se utilizó un simple Proxy de AJAX, y cargará los datos desde la url: 'cargarmispermisos', siendo el nombre de la funcionalidad que está definida en la controladora.php. También adjunta un lector para el proxy. El lector es responsable de la decodificación de la respuesta del servidor en un formato que el store pueda entender. Esta vez se utiliza un lector de JSON, y especificado las configuraciones de root y totalProperty, en la figura 6 se muestra un ejemplo del código fuente de un modelo del grid del componente Flujo de Trabajo.

```
Ext.define('Gestpermisos.store.GridPermisosStore', {
    extend: 'Ext.data.Store',
    requires: 'Gestpermisos.model.GridPermisosModel',
    model: 'Gestpermisos.model.GridPermisosModel',
    autoLoad: true,
    proxy: {
        type: 'ajax',
         url: 'cargarmispermisos',
        actionMethods: {read: 'POST'},
        reader: {
            type: 'json',
            totalProperty: "total",
            root: "campos",
            id: "idelemento"
        }
    }

});
```

Figura 6: Código fuente de un almacén del grid del componente Flujo de Trabajo.

Durante el desarrollo de este procedimiento se definieron los siguientes almacenes:

Tabla 5: Almacenes

Funcionalidades	Almacenes	Total
Estados de la información	GridEstadoStore	1
Transiciones de la información	comboBoxStore,GridFlujodeTrabajoStore	2
Permisos	ArbolPermParaStore, ArbolPermSobreStore, GridPermisosStore, StorePermisoPara	4
Nomencladores	ArbolStore	1
Grupo de usuarios por rol	GridGrupoStore, GridUsuarioStore, cbActivoStore, ArbolEntidadStore, ArbolGrupoStore	5

2.7.5. Definir un modelo

Se define como un conjunto de campos y cualesquiera métodos arbitrarios y las propiedades relevantes para el modelo. Para crear un modelo siguiendo la arquitectura Modelo Vista Controlador que propone Ext JS v6.0, se debe crear un archivo de extensión js. Luego se define que el fichero sea de tipo model extendiendo de Ext.data.Model, se nombran los campos que va a tener el store., en la figura 7 se muestra un ejemplo del código fuente de un modelo del grid del componente Flujo de Trabajo.

```
Ext.define('Gestpermisos.model.GridPermisosModel', {
    extend: 'Ext.data.Model',
    fields: [{
            name:'idelemento'
        },{
            name:'denominacion'
        },{
            name:'tipo'
        },{
            name:'leer'
        },{
            name:'annadir'
        },{
            name:'modificar'
        },{
            name:'eliminar'
        },{
            name:'origen'
        }]
});
```

Figura 7:Definición de un model.

Durante el desarrollo de este procedimiento se definieron los siguientes modelos:

Tabla 6:Modelos

Funcionalidades	Modelos	Total
Estados de la información	GridEstadoModel	1
Transiciones de la información	GridFlujodeTrabajoModel, comboBoxModel	2
Permisos	GridPermisosModel	1
Nomencladores	ArbolModel	1
Grupo de usuarios por rol	GridGrupoModel, GridUsuarioModel, cbActivoModel,	3

2.7.6. Definir un controlador

El controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones o eventos disparados desde la Vista. Para crear un controlador siguiendo la arquitectura Modelo Vista Controlador que propone Ext JS v6.0, se crea un fichero de extensión js y se define que sea de tipo controller, extendiendo de 'Ext.app.Controller'. Con la etiqueta ref se hace referencia a la vista y la etiqueta selector a los eventos dentro de la propia clase controladora, como se muestra una parte del código fuente en la figura del componente Configuración.

Del componente Flujo de Trabajo:

- ✓ EstadosController.
- ✓ FlujoTrabController.

Del componente Configuración:

- ✓ GruposController.
- ✓ NomenController.
- ✓ PermisosController.

```
Ext.define('Gestestado.controller.EstadosController', {
    extend: 'Ext.app.Controller',
    refs: [{
            ref: 'panelp',
            selector: '#vpanelp'
        }, {
            ref: 'btnAddEstado',
            selector: '#add'
        }, {ref: 'btnModEstado',
            selector: '#modif'
        }, {ref: 'btnDellEstado',
            selector: '#dell'},
        {ref: 'btnGenEstado',
            selector: '#gener'},
        {ref: 'gridestado',
            selector: 'gridest'}
```

Figura 8: Definición de un controller.

Durante el desarrollo de este procedimiento se definieron los siguientes controllers*:*

Tabla 7: Controller.

Funcionalidades	Almacenes	Total
Estados de la información	EstadoController	1
Transiciones de la información	FlujoTrabController	1
Permisos	PermisosController	1
Nomencladores	NomenController	1
Grupo de usuarios por rol	GruposController	1

2.8. Pruebas de software

El único instrumento adecuado para determinar el grado de calidad de un producto de software, es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el nivel en que el software cumple con los requerimientos. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante Técnicas de prueba. El proceso de pruebas, sus objetivos y los métodos y técnicas usados se describen en el plan de prueba (pruebasdesoftware 2015).

2.8.1. Pruebas de caja negra

Para Ian Sommerville, las pruebas de caja negra se derivan a partir de la especificación del sistema. El sistema se trata como una caja negra cuyo comportamiento solo puede ser determinado estudiando

sus entradas y sus salidas relacionadas. Otro nombre para esto es pruebas funcionales, debido a que el probador solo le interesa la funcionalidad y no la implementación del software.

La figura 9 ilustra el modelo de un sistema que se admite en las pruebas de caja negra. El probador presenta las entradas al componente o al sistema y examina las correspondientes salidas. Si las salidas no son esperadas entonces la prueba ha detectado un problema con el software.

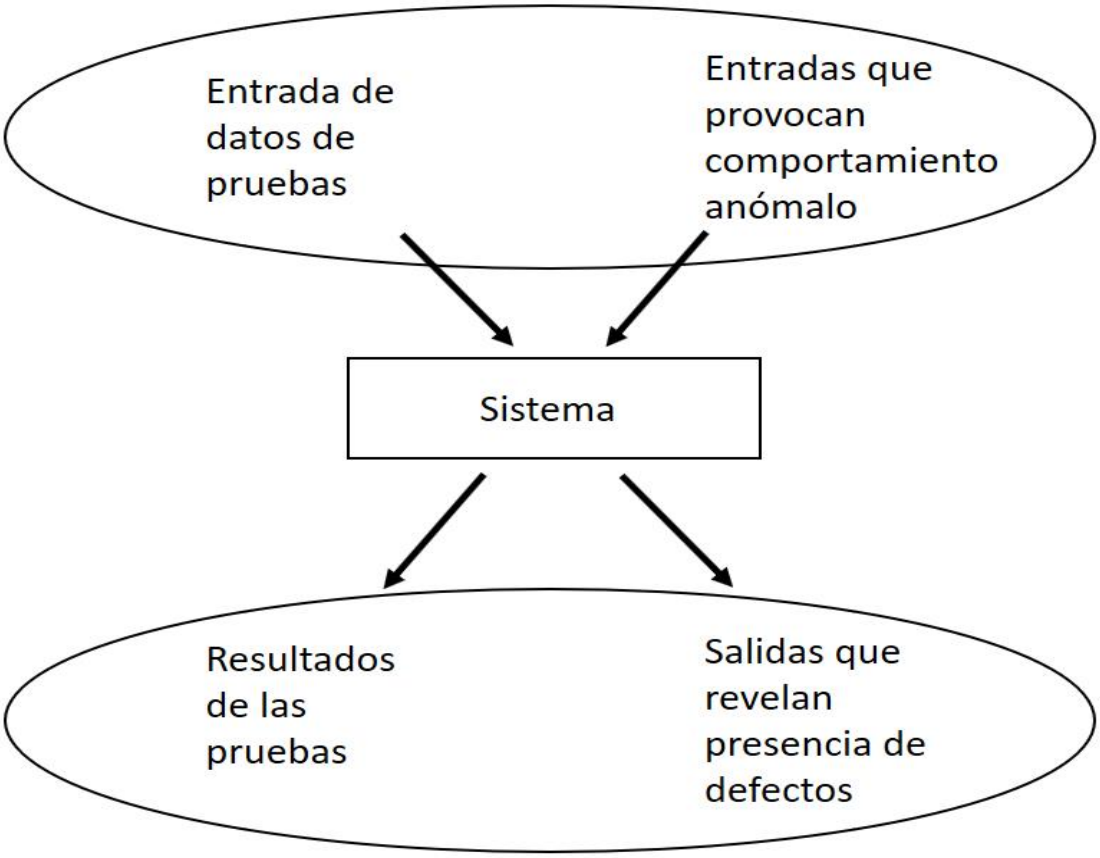


Figura 9: Enfoque de diseño de pruebas de caja negra

El objetivo debería ser seleccionar entradas que tengan una alta probabilidad de fallos de ejecución del sistema. Se utiliza la experiencia previa de las cuáles son las pruebas de defectos que probablemente tendrán éxito y las guías de pruebas ayudarán a elegir la adecuada(Sommerville y Galipienso 2005).

Algunos ejemplos de estas guías son:

- ✓ Elegir entradas que fuerzan a que el sistema genere todos los mensajes de error.
- ✓ Diseñar entradas que hacen que los búferes de entrada se desborden.
- ✓ Repetir la misma entrada o series de entradas varias veces.
- ✓ Forzar a que se generen las salidas inválidas.
- ✓ Forzar los resultados de los cálculos para que sean demasiado grandes o demasiado pequeños.

2.8.2. Partición de equivalente

Según Ian Sommerville, los datos de entrada y los resultados de salida de un programa normalmente se pueden agrupar en clases diferentes que tienen características comunes tales como números positivos, números negativos y selecciones de menús. Los programas normalmente se comportan de forma similar para todos los miembros de una clase. Es decir, si se prueba un programa que realiza algún cálculo y requiere dos números positivos, entonces se esperaría que el programa se comportase de la misma forma para todos los números positivos.

Debido a este comportamiento equivalente, estas clases se denominan a menudo partición equivalente. Una aproximación sistemática al diseño de casos de prueba se basa en identificar todas las particiones para un sistema o componente. Los casos de prueba se diseñan para las entradas o salidas pertenezcan a estas particiones. Las pruebas de particiones pueden utilizarse para diseñar casos de prueba tanto para sistemas como para componentes (Sommerville y Galipienso 2005).

Las clases de equivalencia se definen de acuerdo con las siguientes directrices:

- ✓ Si una condición de entrada especifica un rango, se definen una clase de equivalencia válida y dos no válidas.
- ✓ Si una condición de entrada requiere un valor especifico, se definen una clase de equivalencia válida y dos no válidas.
- ✓ Si una condición de entrada especifica un miembro de un conjunto, se definen una clase de equivalencia válida y otra no válida.
- ✓ Si una condición de entrada es booleana, se definen una clase de equivalencia válida y otra no válida.

De las antes mencionadas, se decide aplicar la técnica de **Partición de Equivalencia**, esta permite definir casos de prueba que declaren clases de errores, reduciendo el número de casos de prueba a desarrollar para demostrar que las funciones del software son operativas.

A continuación, se muestra un ejemplo de 5 escenarios de uno de los casos de pruebas realizados.

Condiciones de ejecución:

- ✓ Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- ✓ Se debe seleccionar el subsistema de Configuración.
- ✓ Se debe seleccionar la opción Flujo de Trabajo/Estados de la información.

Tabla 8: Escenario de prueba del requisito Adicionar estado.

Nombre del registro	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar estado de la información.	Se adiciona un estado de la información, con el atributo: Denominación.	EP 1.1: Adicionar un nuevo estado de la información introduciendo datos válidos.	– Se presiona el botón **Adicionar**. – Se insertan todos los datos. – Se presiona el botón **Aceptar**. – Se presiona el botón **Aceptar** del mensaje de información.
		EP 1.2: Adicionar un nuevo estado de la información dejando campos requeridos en blanco.	– Se presiona el botón **Adicionar**. – Se introducen los datos dejando campos requeridos en blanco. – Se presiona el botón **Aceptar**. – Se presiona el botón **Aceptar** del mensaje de información.
		EP 1.3: Adicionar un nuevo estado de la información con errores en los campos.	– Se presiona el botón **Adicionar**. – Se introducen los datos con errores. – Se presiona el botón **Aceptar**. – Se presiona el botón **Aceptar** del mensaje de información.
		EP 1.4: Adicionar un nuevo estado de la información presionando el botón Aplicar.	– Se presiona el botón **Adicionar**. – Se insertan todos los datos. – Se presiona el botón **Aplicar**. – Se retorna a la ventana **Adicionar** un nuevo estado de la información.
		EP 1.5: Cancelar	– Se presiona el botón **Adicionar**. – Se introducen o no los datos en el formulario. – Se presiona el botón **Cancelar**.

Tabla 9: : Descripción de variables del escenario de prueba del requisito Adicionar estado.

No	Nombre de campo	Tipo	Válido	Inválido	Inválido
1	Denominación.	Campo de texto.	Cadena de caracteres, números y letras.	Vacío.	Caracteres especiales: @#$%^&*()_

Tabla 10: Juegos de datos a probar en escenario de prueba del requisito Adicionar estado.

Id del escenario	Escenario	Variable 1 (Denominación)	Respuesta del sistema	Resultado de la prueba
EP 1.1	Adicionar un nuevo estado de la información introduciendo datos válidos.	V(Creado)	Se guarda la información del estado adicionado. El sistema muestra el siguiente mensaje: “El estado ha sido adicionado correctamente.”	
		V(Aprobado)		
EP 1.2	Adicionar un nuevo estado de la información dejando campos requeridos en blanco.	I(Vacío)	El sistema muestra el siguiente mensaje: “Por favor verifique que el campo está vacío.”. Se muestra marcado en color rojo el campo requerido.	
EP 1.3	Adicionar un nuevo estado de la información con errores en los campos.	I($&~)	El sistema no permite la escritura de los caracteres especificados para la variable Denominación	

EP 1.4	Adicionar un nuevo estado de la información presionando el botón Aplicar.	V(Creado)	Se guarda la información del estado adicionado. Se retorna a la ventana Adicionar un nuevo estado de la información.	
EP 1.5	Cancelar	NA	Se cancela la operación y se cierra la ventana.	

2.8.3. Resultados generales de las pruebas funcionales

Al aplicar las pruebas de caja negra, se realizaron tres iteraciones. En una primera iteración las no conformidades detectadas fueron un total de quince, de ellas nueve significativas y seis no significativas. En una segunda iteración se detectaron ocho no conformidades, de ellas tres significativas y cinco no significativas. Ya en una tercera iteración, se puedo concluir que los resultados obtenidos han sido satisfactorios desde el punto de vista funcional.

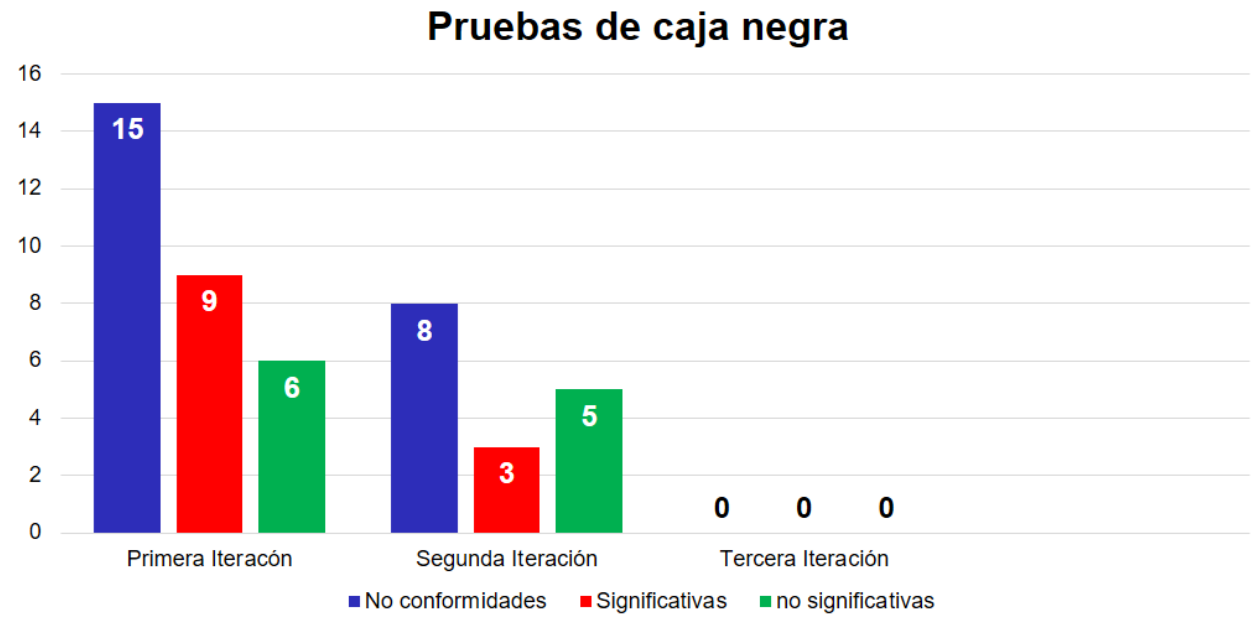


Figura 10: Pruebas de caja negra

2.9. Conclusiones parciales

Con la realización de este capítulo se concluye, que el estudio de la arquitectura SIPAC, posibilitó la definición de la migración de la capa de presentación del sistema. Los procedimientos que se tuvieron en cuenta para realizar la migración de la capa de presentación de los componentes Flujo de Trabajo y Configuración, permitieron definir a partir de la arquitectura Modelo, Vista, Controlador que propone Ext JS una nueva estructura jerárquica de presentación además se logró:

- ✓ Realizar un análisis de la arquitectura de SIPAC, lo que permitió identificar las características de la capa de presentación donde se desarrolló la migración.
- ✓ Definir la nueva estructura de directorio para introducir la arquitectura MVC definida por Ext JS en su versión 6.0.
- ✓ Configurar los ficheros de extensión phtml para importar los ficheros principales de la librería de Ext JS.
- ✓ Definir los controladores que permiten gestionar los eventos de las vistas, las vistas que representan los componentes definidos por Ext JS y los almacenes que hacen referencia a todos los modelos del componente.
- ✓ Aplicar pruebas de caja negra para comprobar el correcto funcionamiento del sistema.

Capítulo 3. Validación de la migración

3.1. Introducción

El presente capítulo está centrado en la aprobación de la investigación, validando la migración realizada, donde se ejecutan pruebas de rendimiento mediante el uso de la herramienta JMeter que permite evidenciar el rendimiento de la aplicación. Además, se utiliza un entorno de trabajo en igualdad de condiciones para realizar una comparación del sistema con la versión 2.2 y la 6.0 de Ext JS. Los resultados permitirán establecer comparaciones entre las versiones, en dependencia del tiempo de carga de las interfaces, accediendo cierta cantidad de usuarios al mismo tiempo.

3.2. Pruebas de rendimiento

La prueba de rendimiento está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. La migración se da durante todos los pasos del proceso de la prueba. Incluso al nivel de unidad, se debe asegurar el rendimiento de los módulos individuales. Sin embargo, hasta que no están completamente integrados todos los elementos del sistema no se puede asegurar realmente el rendimiento del sistema.

Las pruebas de rendimiento, a menudo, van emparejadas con las pruebas de resistencia (Stress) y, frecuentemente, requieren instrumentación tanto de software como de hardware. Es decir, muchas veces es necesario medir la utilización de recursos (por ejemplo, ciclos de procesador), de un modo exacto. La instrumentación externa puede monitorizar los intervalos de ejecución, los sucesos ocurridos (por ejemplo, interrupciones) y muestras de los estados de la máquina en un funcionamiento normal. Instrumentando un sistema, el encargado de la prueba puede descubrir situaciones que lleven a degradaciones y posibles fallos del sistema (Pressman y Maxim 2016).

3.2.1 Tipos de pruebas de rendimiento

✓ **Prueba de Carga**. Es una prueba que somete al sistema a una carga de trabajo concreta y estable durante un tiempo relativamente corto (puede ser una hora). El nivel de carga debe ser alto y continuo para verificar que el sistema soporta esa carga sin pérdidas de servicio y con un tiempo de respuesta estable y sin degradaciones. Se suelen hacer con varios niveles de carga (100%, 150% y 200%, por ejemplo), para conocer el comportamiento del sistema si la carga real llegara a esos niveles. Por ejemplo: se prevé que el sistema esté sometido a una carga de 100 peticiones por segundo, así que en las pruebas probaremos 3 escenarios: 100 peticiones por segundo, 150 peticiones por segundo y 200 peticiones por segundo. Individualmente los resultados dirán cuál es el tiempo de respuesta en cada uno de ellos y en conjunto se construirá y analizará la evolución de los tiempos de respuesta y cuáles son los niveles de carga en los que el rendimiento se empieza a deteriorar.

✓ **Prueba de Stress**. Es un tipo de prueba que va buscando el punto de ruptura del sistema, es decir, a qué nivel de carga de trabajo se pierde el servicio y qué pasa en los niveles anteriores. El objetivo de este tipo de pruebas es conocer el límite de carga de trabajo al que podemos llevar a un sistema. Este límite en sí mismo no nos dice nada, pero hace posible que se pueda tomar medidas antes de llegar a ese nivel, por ejemplo: el sistema en pruebas se satura con 100 usuarios, así que cuando en producción llegue a 75 habilitaremos una alerta para cortar el acceso a los usuarios o desviaremos peticiones a sistemas auxiliares, o activaremos un protocolo de actuación para evitar la caída del servicio.

✓ **Prueba de Volumen.** Este tipo de prueba tiene por objetivo verificar si el sistema es estable durante un largo periodo de tiempo. Básicamente es como una prueba de carga con una duración superior, por ejemplo 24 horas. La idea es encontrar errores acumulativos, es decir, errores que pasan una prueba de esfuerzo por que producen un daño muy pequeño y que a la larga van a terminar deteriorando o colapsando el rendimiento del sistema (Mingo 2013).

El empleo de este tipo de pruebas sirve para diferentes propósitos tales como demostrar que el sistema cumple los criterios de rendimiento, comparar dos sistemas para encontrar cuál de ellos funciona mejor o medir qué partes del sistema o de carga de trabajo provocan que el conjunto rinda mal. Para su diagnóstico se utilizan herramientas que pueden ser monitorizaciones que midan qué partes de un software contribuyen más al mal rendimiento o para establecer niveles que mantenga un tiempo de respuesta aceptable (juntadeandalucia.es 2014).

Luego de realizado un análisis de los diferentes tipos de pruebas de rendimiento existentes, se seleccionan para ser ejecutadas al sistema la prueba de carga, por ser la más simple entre las pruebas de rendimiento y ofrecer importantes criterios de negocio a los tiempos de respuesta de todas las transacciones y la prueba de estrés pues es esta prueba la que permite comprender los límites superiores de la capacidad dentro del sistema.

Estas pruebas de rendimiento de carga y estrés son realizadas de manera general con el objetivo de determinar la disminución del tiempo de carga de las interfaces, en la versión 6.0 de Ext JS de los componentes Flujo de Trabajo y Configuración.

3.3. Entorno de Prueba

Para realizarle las pruebas a los componentes anteriormente migrados se hace necesario preparar el entorno donde se realizarán, las pruebas, para ello se utiliza una computadora con las siguientes características:

Hardware: Intel Core i5-2348 a 2.8 GHz, 4 GB de RAM, sistema operativo Ubuntu 14.04, tipo de sistema 64-bit en una partición de 40 GB.

Software: gestor de base de datos: PostgreSQL v9.4, Servidor de Aplicaciones: Apache 2.4.9, máquina virtual de Java: openjdk-8, navegador web: Mozilla Firefox.

LAN: tarjeta de red 100 Mbps de velocidad.

3.4. Realizar pruebas de rendimiento

En esta sección se describe tanto la herramienta utilizada como las pruebas de rendimiento que se ejecutarán a los componentes Flujo de trabajo y Configuración. El objetivo de obtener una mejora en el consumo de recursos y el tiempo de carga de las interfaces definiéndose a partir de los resultados obtenidos.

Para analizar el tiempo de respuesta de la aplicación se utiliza la herramienta Jmeter en su versión 2.13. La aplicación Jmeter es un software de código abierto, una aplicación 100% Java puro diseñado para cargar la conducta funcional de prueba y medida de rendimiento. Fue diseñado originalmente para aplicaciones Web de prueba, pero desde entonces se ha expandido a otras funciones de prueba.

Jmeter puede ser utilizado para probar el rendimiento tanto en los recursos estáticos y dinámicos (Servicios Web (SOAP / REST), Web dinámicas idiomas PHP, Java, ASP.NET, archivos, objetos Java, bases de datos y consultas, servidores FTP y más). Se puede utilizar para simular una carga pesada en un servidor, grupo de servidores, la red o el objeto a probar su resistencia o para analizar el rendimiento general bajo diferentes tipos de carga. Se puede utilizar para realizar un análisis gráfico de rendimiento o para probar su comportamiento / objeto de servidor / script bajo carga pesada concurrente (jmeter.apache.org 2014).

Para empezar, se crea un plan de pruebas sobre la base de una lista ordenada de peticiones HTTP. Utilizando el elemento Grupo de Hilos que se muestra en la Figura 11, se inicia la cabecera de la prueba en la que se define la cantidad de usuarios que se modelan definidos en los Casos de Pruebas.

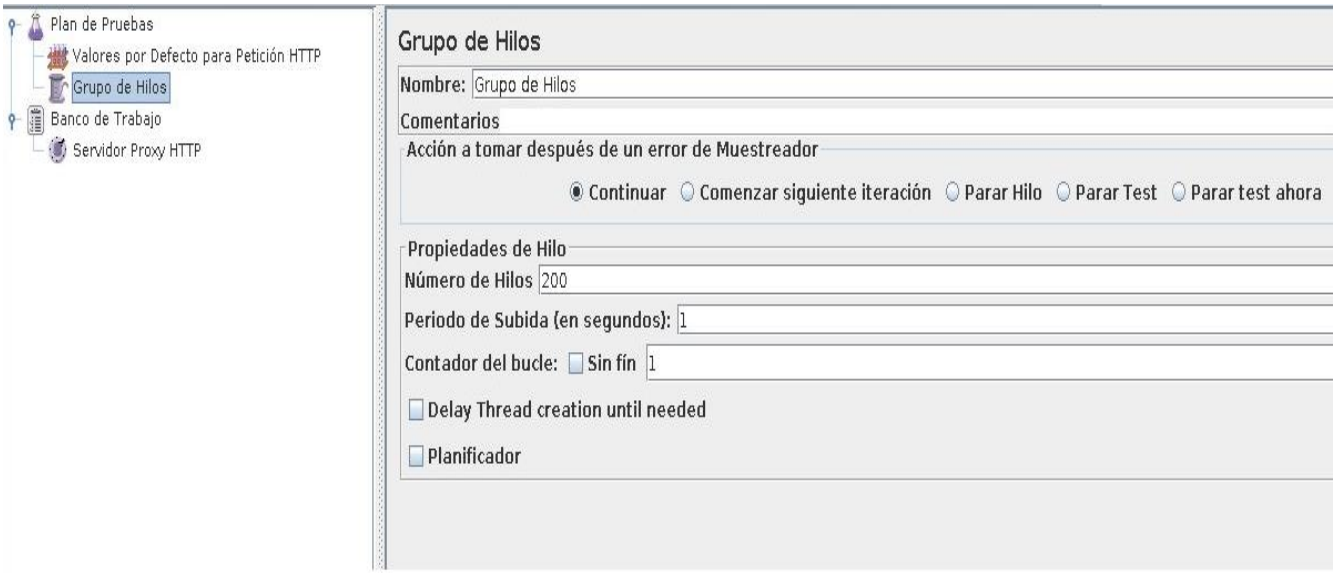


Figura 11: Elemento de grupo de hilos

Las propiedades de los hilos que se utilizan con este componente son:

- ✓ Número de hilos: número de usuarios a simular (200, 100 y 50 para cada caso de prueba utilizado).
- ✓ Período de subida: tiempo que se toma el JMeter en lanzar todos los hilos (especifica el período intermedio en segundos en los cuales va a ir iniciándose cada uno de los usuarios, para este caso se especifica un segundo).
- ✓ Contador del bucle: número de veces a realizar la prueba (para esta prueba se especifica que se simule una sola vez).

Se graban los escenarios a través del elemento Servidor Proxy HTTP para obtener una muestra de los elementos de interacción del sistema, donde se entran datos y se envían a la base de datos, como se muestra en la siguiente figura:

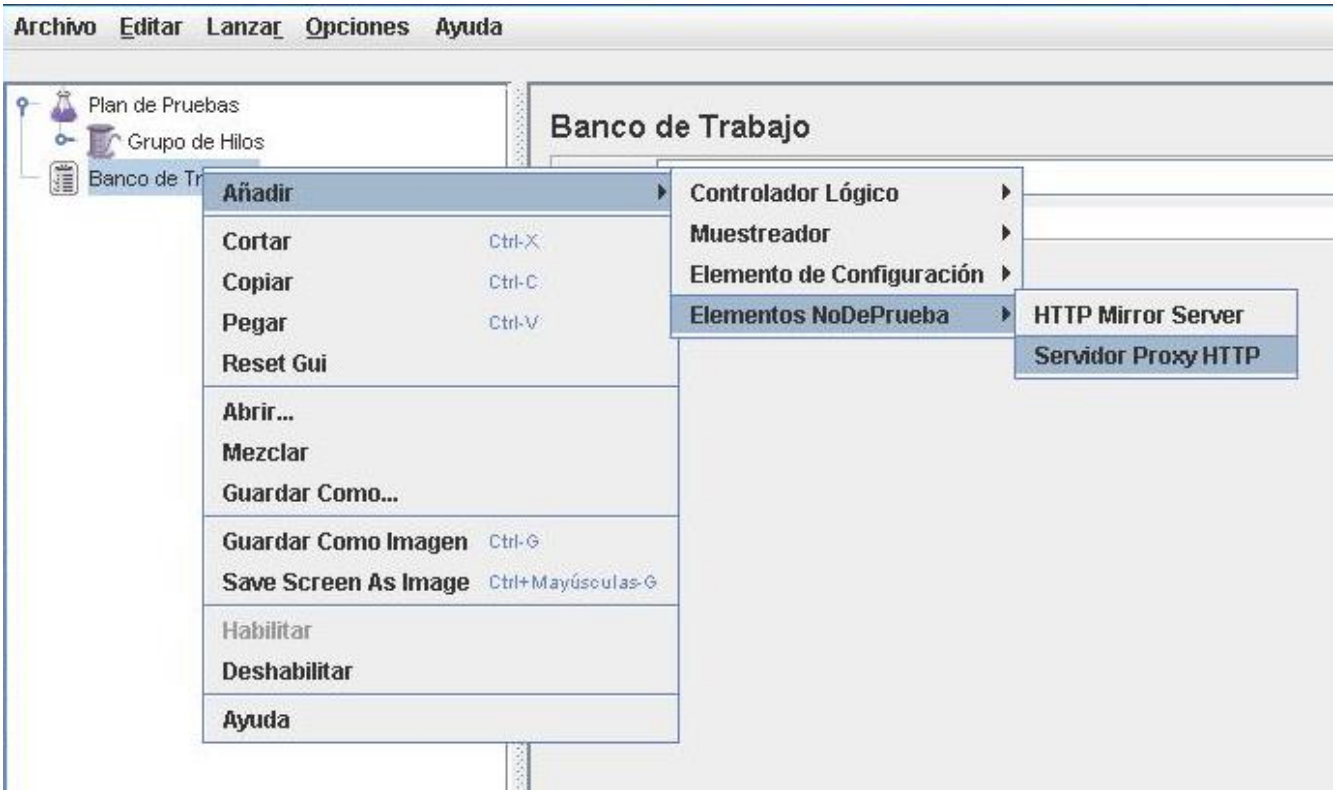


Figura 12: Selección del Servidor Proxy.

Este elemento permite grabar todas las acciones realizadas en la aplicación de manera que se generan los escenarios de prueba automáticamente. Aquí se especifica un puerto para el servidor y la manera en la que se desean organizar los resultados de la grabación y se presiona el botón Arrancar.

Figura 13: Valores para el Servidor Proxy HTTP.

Posteriormente en el navegador, se le especifica la dirección y el puerto que se utiliza para la grabación de los escenarios. De esta manera se indica que el proxy que utilizará el navegador es el definido en el JMeter. Luego se ejecuta la aplicación y cada paso realizado en esta es registrado en el JMeter de manera organizada.

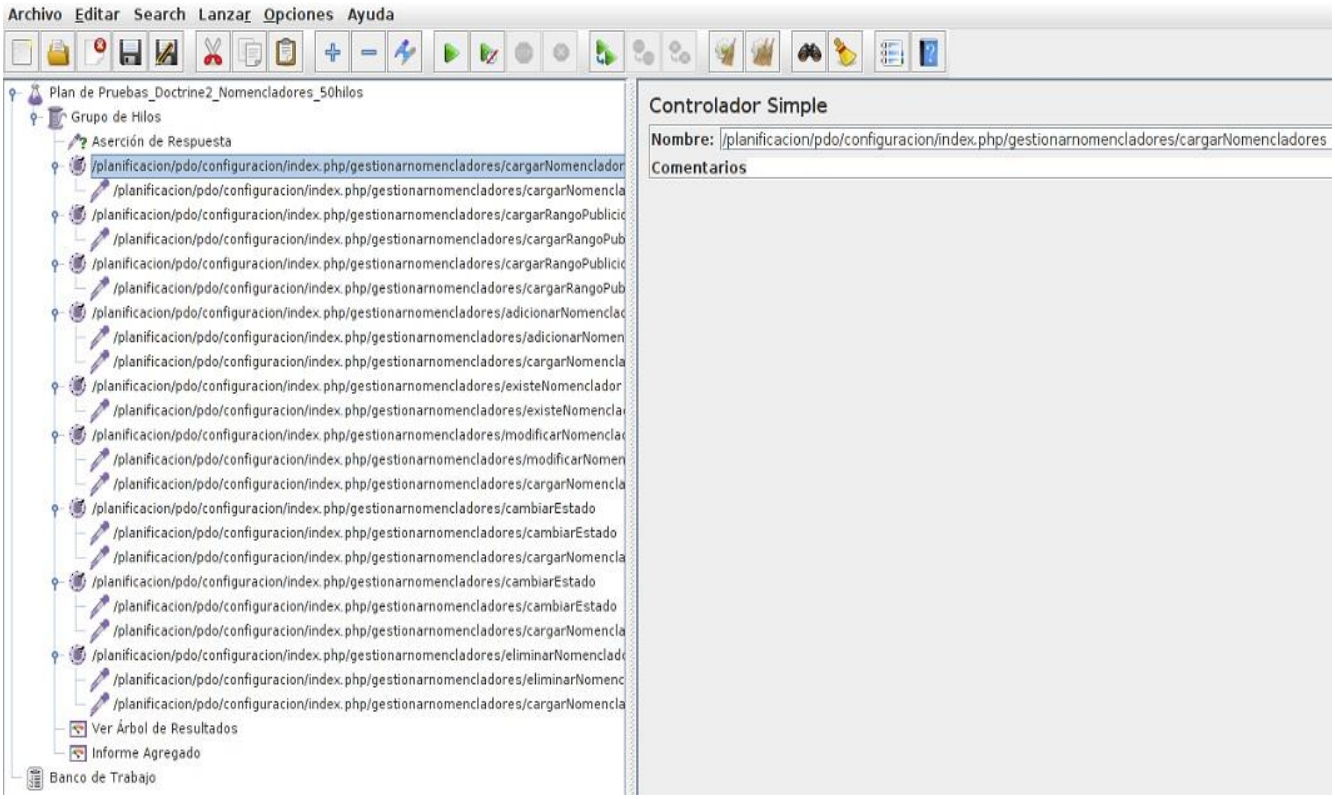


Figura 14: Grabación de navegación de la web

Como se puede apreciar en la imagen anterior las peticiones HTTP están grabadas dentro de controladores diferentes que contienen toda la información. Por cada petición HTTP se genera un gestor de cabecera HTTP, estos se eliminan porque interfieren en el éxito de las respuestas del servidor web; además los datos insertados en el sistema se muestran en los parámetros enviados con la petición.

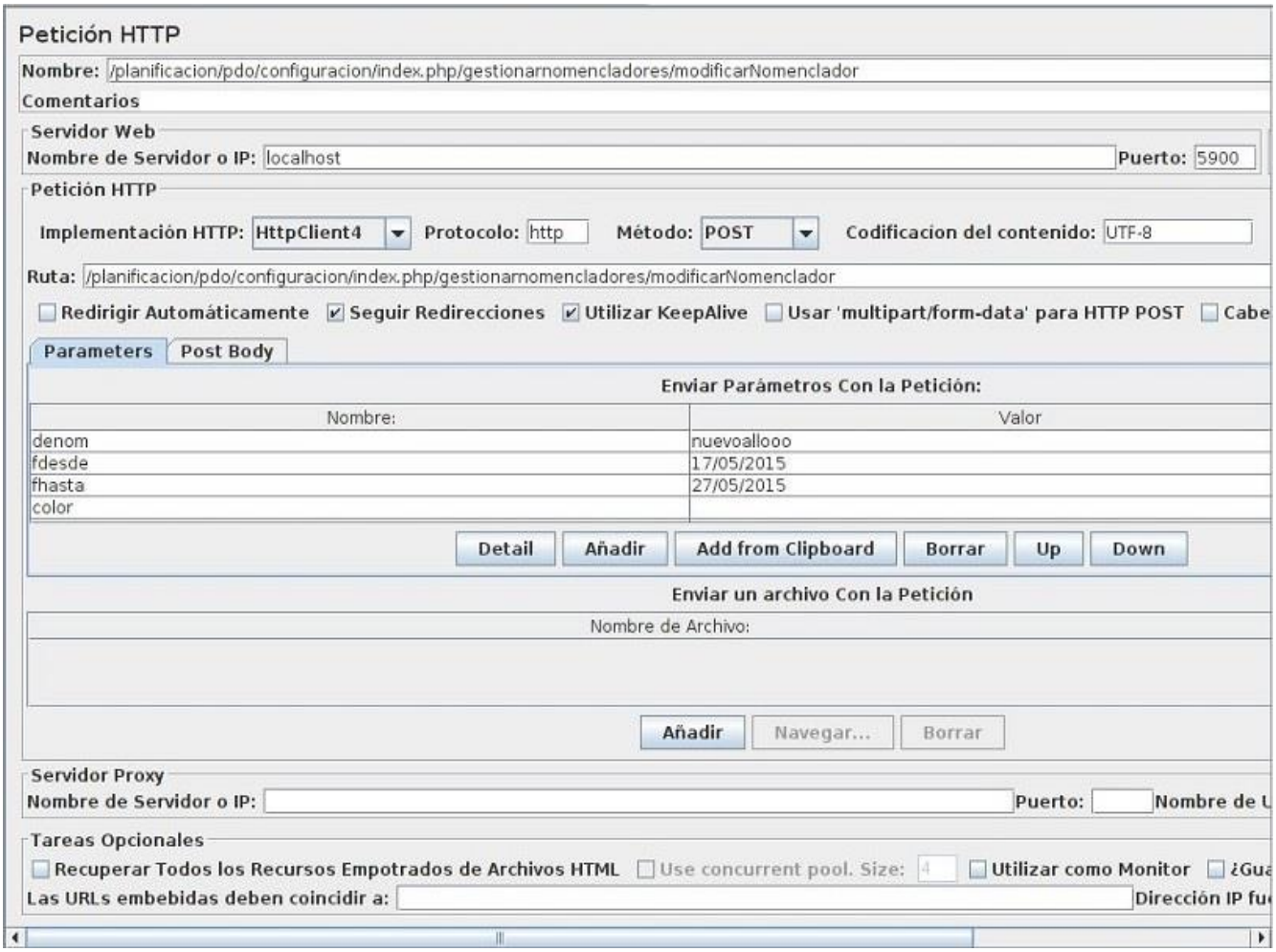


Figura 15: Ejemplo de petición HTTP de la grabación.

Una vez grabadas todas las peticiones HTTP, es necesario agregar otros elementos en los cuales se especificarán los parámetros a comprobar en la prueba. Esto puede realizarse apoyándose en el elemento Aserción de Respuesta.

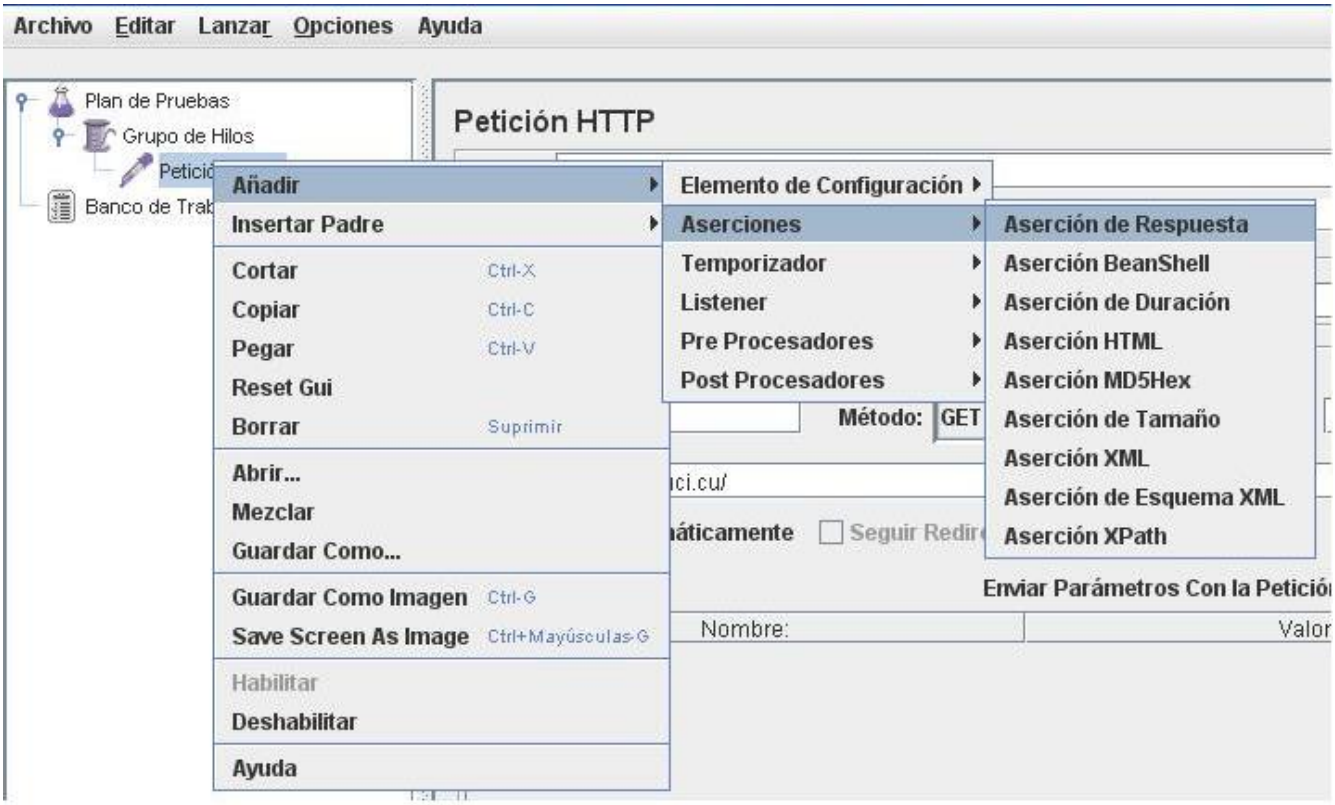


Figura 16: Selección de Aserción de Respuestas.

Este necesita conocer que la página a la cual se quiere acceder se cargó satisfactoriamente por lo que se especifica en la propiedad Patrón a Probar el código 200 para una página cargada exitosamente.

Figura 23. Datos de la Aserción de Respuesta

Como las pruebas que se realizan son de carga y estrés se necesita un informe que muestre los resultados relacionados a los datos necesarios para comprobar el comportamiento de la aplicación. Para ello se utiliza el elemento Informe Agregado.

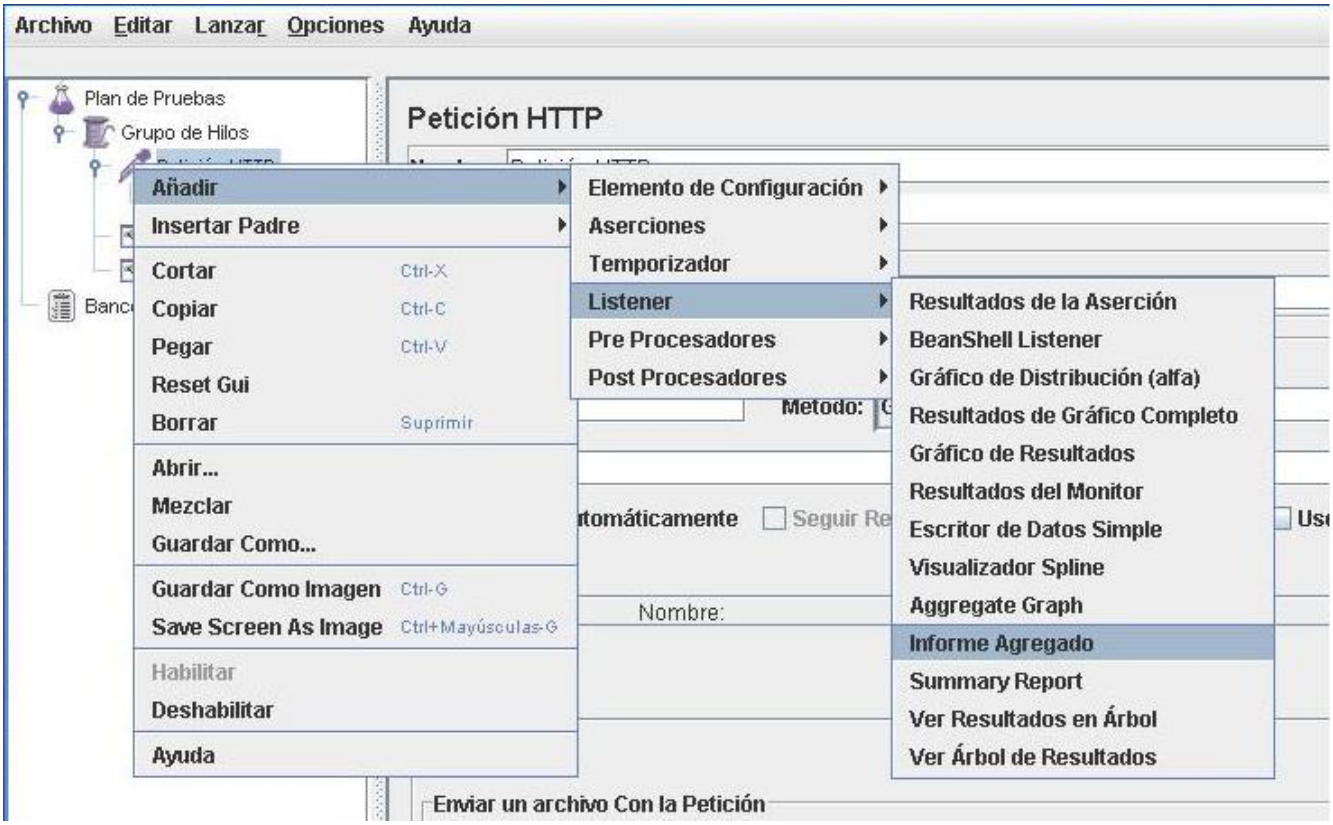


Figura 17:Selección de Informe Agregado

En estos informes se recogen los siguientes indicadores:

- ✓ Número Muestras: cantidad de peticiones realizadas a la URL.
- ✓ Media: tiempo promedio en milisegundos transcurrido para un conjunto de resultados.
- ✓ Mediana: mediana aritmética (elemento de una serie ordenada de valores crecientes de manera que divide en dos partes iguales).
- ✓ Min: tiempo mínimo transcurrido para las muestras de peticiones de una determinada URL.
- ✓ Max: tiempo máximo transcurrido para las muestras de peticiones de una determinada URL.
- ✓ Porcentaje de Error: porcentaje de las peticiones con errores.

Por otra parte, es recomendable utilizar el elemento Ver Árbol de Resultados, que muestra en detalle la información que ha sido cargada.

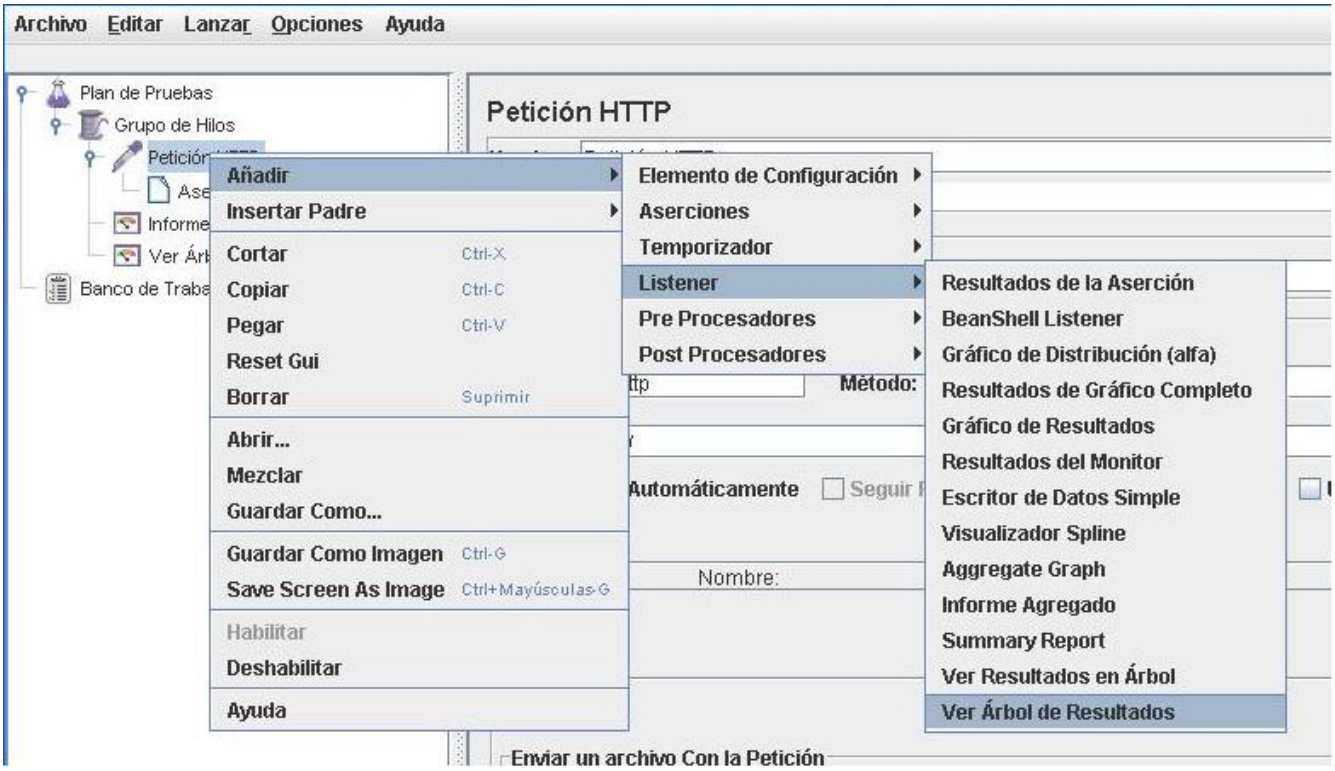


Figura 18: Selección de Ver Árbol de Resultados

Con la grabación de las peticiones el Plan de Pruebas quedaría de la siguiente manera:

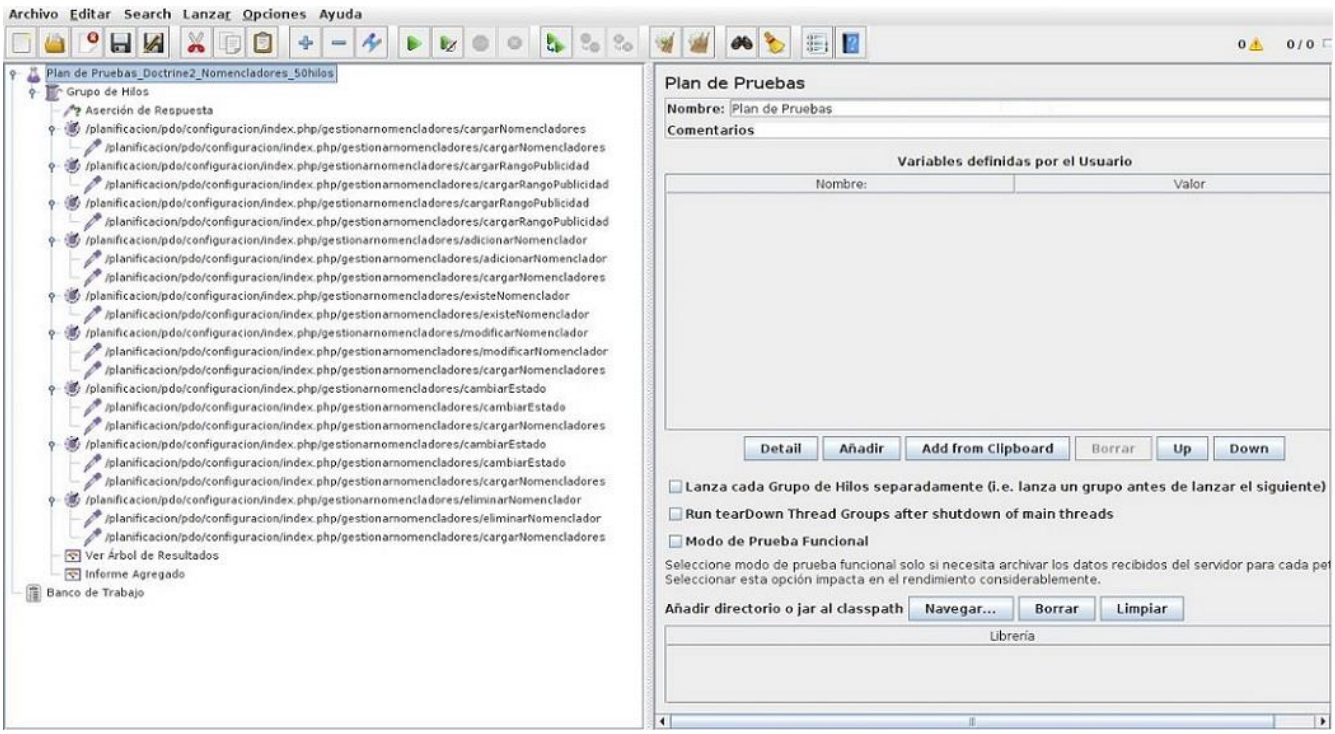
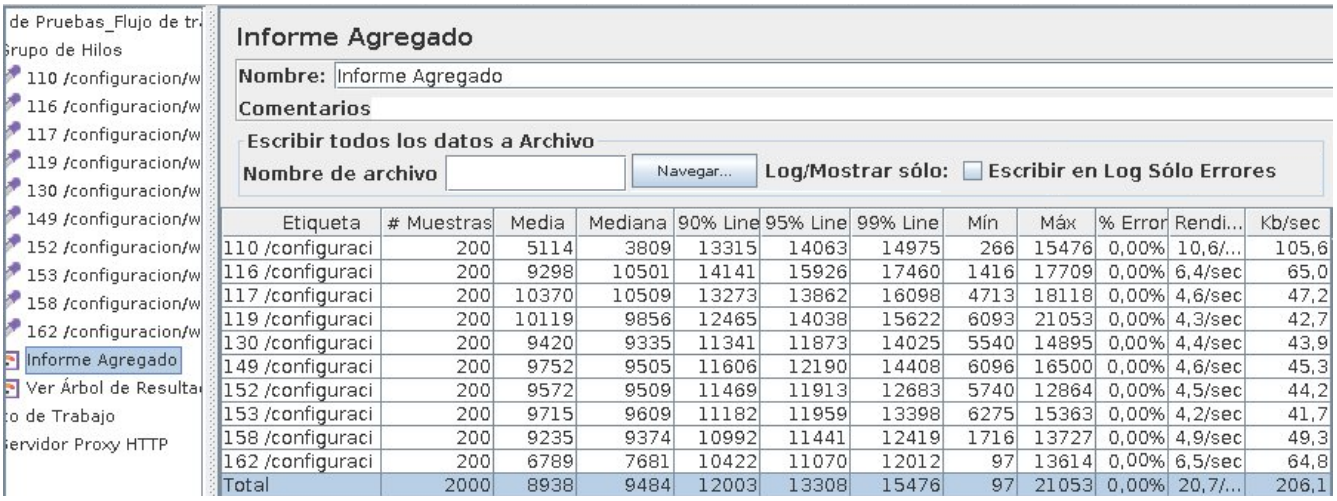


Figura 19: Confección de un Plan de Pruebas.

El elemento Informe Agregado muestra los resultados correspondientes a cada Plan de Prueba, donde se evidencia el porcentaje de error para cada petición, el número de muestras y el rendimiento en segundos o milisegundos. A continuación, se muestra un ejemplo:



Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendi...	Kb/sec
110 /configuraci	200	5114	3809	13315	14063	14975	266	15476	0,00%	10,6/...	105,6
116 /configuraci	200	9298	10501	14141	15926	17460	1416	17709	0,00%	6,4/sec	65,0
117 /configuraci	200	10370	10509	13273	13862	16098	4713	18118	0,00%	4,6/sec	47,2
119 /configuraci	200	10119	9856	12465	14038	15622	6093	21053	0,00%	4,3/sec	42,7
130 /configuraci	200	9420	9335	11341	11873	14025	5540	14895	0,00%	4,4/sec	43,9
149 /configuraci	200	9752	9505	11606	12190	14408	6096	16500	0,00%	4,6/sec	45,3
152 /configuraci	200	9572	9509	11469	11913	12683	5740	12864	0,00%	4,5/sec	44,2
153 /configuraci	200	9715	9609	11182	11959	13398	6275	15363	0,00%	4,2/sec	41,7
158 /configuraci	200	9235	9374	10992	11441	12419	1716	13727	0,00%	4,9/sec	49,3
162 /configuraci	200	6789	7681	10422	11070	12012	97	13614	0,00%	6,5/sec	64,8
Total	2000	8938	9484	12003	13308	15476	97	21053	0,00%	20,7/...	206,1

Figura 20: Informe Agregado

Mientras que el componente Ver Árbol de Resultados permite visualizar cómo fueron manejadas cada una de las peticiones. Este elemento muestra el código de respuesta, el mensaje de respuesta, la cabecera de respuesta y otras propiedades. La siguiente imagen muestra un ejemplo de este componente.

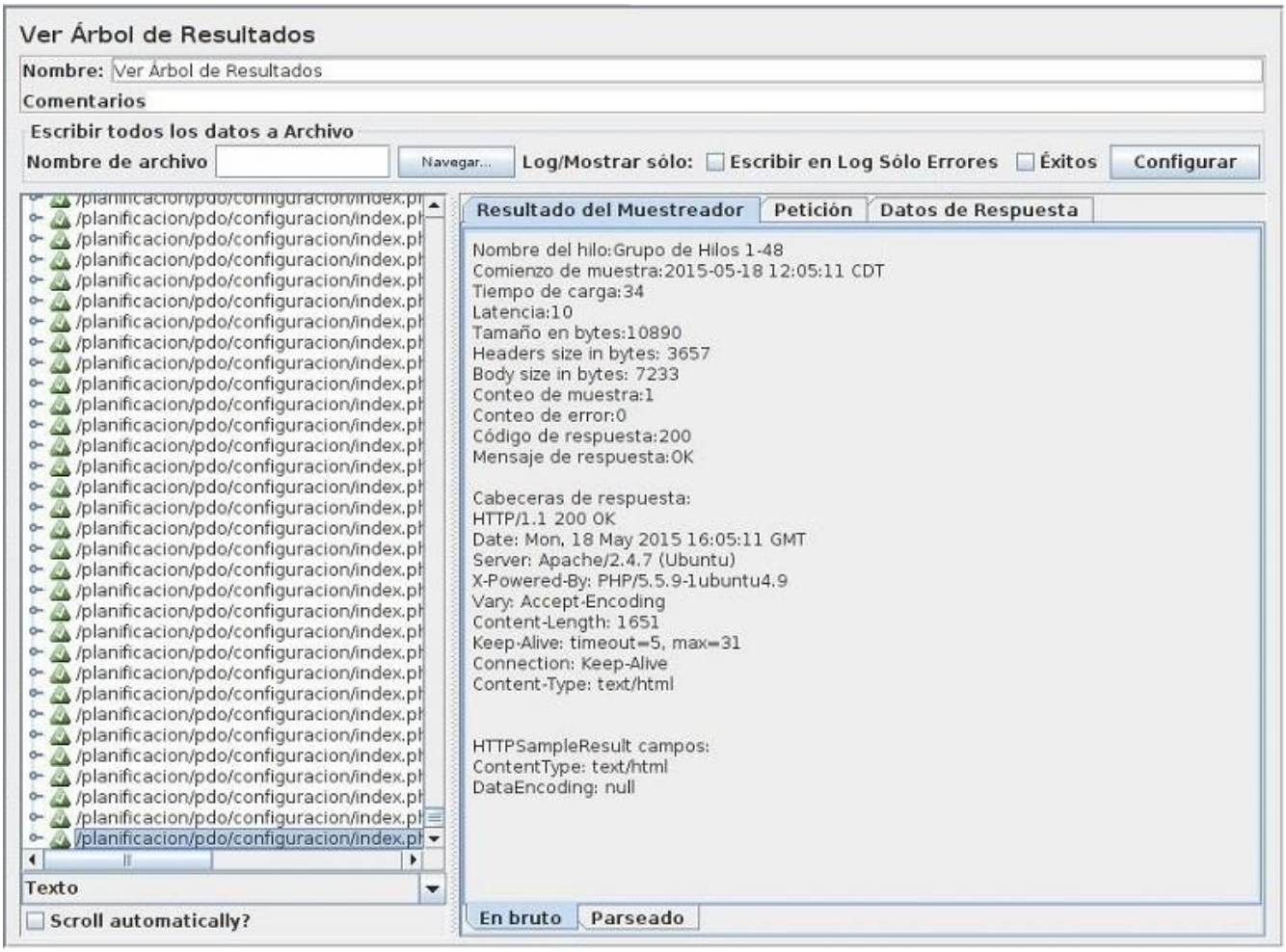


Figura 21: Ver Árbol de Resultados

3.5. Resultados de las pruebas

Para que se llevarán a cabo las pruebas se definieron tres estados de 200, 100 y 50 hilos para cada caso de prueba de carga, los cuales simulan 200, 100 y 50 accesos de usuarios respectivamente; simulando estas cantidades de usuarios conectados concurrentemente al sistema haciendo peticiones al servidor.

Para calcular el Tiempo total (Díaz et al. 2012) en que los usuarios estuvieron accediendo al sistema, se utilizó la siguiente fórmula:

- ✓ Tiempo Total = $\#Muestras * Media\,[ms]$

Mientras que para calcular el Tiempo promedio de cada usuario que estuvo accediendo al sistema se utilizó la siguiente fórmula:

- ✓ Tiempo promedio por hilo = $Tiempo\frac{Total}{1000*60*hilos}\,[min]$

Para realizar las pruebas se configuraron 200 hilos cada un segundo, siendo este el mayor valor entre los tres estados definidos anteriormente. Se procede a evaluar un componente haciendo uso de las dos versiones de Ext JS para obtener una mejor visualización de los resultados. Los valores obtenidos en el Informe Agregado correspondientes a cada versión se muestran a continuación.

El tiempo promedio para acceder a una página es 1,476 segundos, realizándose un total de 13000 requerimientos al servidor como se muestra en la figura#. El tiempo total utilizado para los 200 hilos se puede calcular con la siguiente fórmula:

Tiempo total = #Muestras * Media = 13000 * 1476 = 19188000 milisegundos.

El tiempo promedio total requerido por cada hilo, se puede calcular de la siguiente manera:

((Tiempo total/1000) /60) /cantidad de hilos = ((19188000 /1000) /60) /200 = 1,599 minutos.

Ext JS 2.2

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendi...	Kb/sec
Total	13000	1476	2584	12003	13308	15476	97	21053	0,00%	20,7/...	206,1

Como puede reflejarse en la figura#, el tiempo promedio para acceder a una página es 0,713 segundos, realizándose un total de 13000 requerimientos al servidor.

Tiempo total = 13000 * 713 = 9269000 milisegundos.

Tiempo promedio total = ((9269000 /1000) /60) /200 = 0,772 minutos.

Ext JS 6.0

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendi...	Kb/sec
Total	13000	713	694	10351	9305	15476	126	1583	0,00%	17,4/...	179,9

Tabla 11: Análisis de resultados.

	50		100		200	
	Ext JS 2.2	Ext JS 6.0	Ext JS 2.2	Ext JS 6.0	Ext JS 2.2	Ext JS 6.0
Número de Muestras	3250	3250	6500	6500	13000	13000
Media	369	148	725	336	1476	713
Mediana	461	229	1097	421	2584	694
Línea de 90%	3516	215	1131	4124	12003	10351
Min	122	0	33	134	97	126
Max	5894	361	9365	859	21053	1583
Porcentaje de Error	0.0	0.0	0.0	0.0	0.0	0.0
Tiempo total (ms)	1199250	481000	4712500	2184000	19188000	9269000
Tiempo promedio (min)	0,3998	0,1603	0,8541	0,364	1,599	0,772

A partir de los resultados anteriores se puede concluir que el tiempo total utilizado para la cantidad definida de hilos, cambia notablemente en la versión 6.0 de Ext JS respecto a la versión 2.2. Realizando una comparación se tiene como tiempo general de la prueba: 25099750 milisegundos para la versión 2.2 y 11934000 milisegundos para la 6.0. La herramienta arrojó como resultado que hubo una mejora en el rendimiento de 13165750 milisegundos, estos resultados demuestran una reducción del tiempo, lo que representa una reducción del 35,55% de este indicador.

Por su parte, en el tiempo promedio requerido para cada hilo se puede apreciar que hubo una mejora de la versión 6.0 respecto a la 2.2. El tiempo promedio general que se empleó para cada hilo fue de 2,8529 min en la versión 2.2 y 1,2963 min en la 6.0. Estos resultados mostraron una reducción de 1,5566min para la interacción con el sistema por las cantidades de usuarios simulados al utilizar ambas versiones, esto representa una disminución del 38,15% de este indicador.

3.6. Consumo de recursos

Con la ayuda de la herramienta Firefox, se observó que el consumo de recursos en KB del componente Configuración con la versión 6.0 de Ext JS es de 391.8 KB menor en comparación con la versión 2.2 que es de 589, 63 KB. El consumo de recursos también disminuyó con el desarrollo de la migración. ver Tabla 1.

Tabla 12: Pruebas de consumo y tiempo.

Funcionalidades	Recursos (KB)	Tiempo (s)
Estados de la información	227.5	2.14
Transiciones de la información	68.2	2.84
Permisos	46.1	1.29
Nomencladores	209.6	2.65
Grupo de usuarios por rol	47.5	0.8
Total	598.9	9.72

3.7. Resolver no conformidades

En la presente investigación se toma como parámetro de rendimiento a mejorar los tiempos de respuestas de las funcionalidades del sistema.

Promedio que se demoran las peticiones fue de 225 milisegundos y 447 milisegundos.

El tiempo máximo que se demora en cargar las peticiones fue de 12 008 milisegundos y 2770 milisegundos evidenciándose una diferente de casi 10 segundos.

La herramienta arrojo que hubo una mejora en el rendimiento de 2100 milisegundos.

Se observa una mejora en dos de los tres parámetros medidos en cuanto al tiempo de respuesta, por lo que se concluye que el rendimiento del módulo Vehículo mejoró. Aunque se observa una mejora en el rendimiento del módulo, es necesario esperar a la integración de todos los módulos del sistema Orbita, para comprobar si existe una mejora en el rendimiento.

En la actividad se resuelven las no conformidades encontradas durante la ejecución de las pruebas en el tiempo determinado, teniendo en cuenta que el tiempo en esta actividad es muy importante, para la

culminación de la migración, se debe tener el compromiso de cada uno de los involucrados en la migración para resolver las no conformidades en un tiempo record. A partir de los resultados obtenidos con los elementos de la herramienta JMeter: Informe Agregado y Árbol de Resultados, se concluye que las peticiones se ejecutaron correctamente. En el caso del Informe Agregado se indica en la columna porciento de error si existió algún problema, mientras que el elemento Ver Árbol de Resultados indica el código de error y en color rojo la petición con errores.

Durante las pruebas realizadas, se destaca que no se reportaron incidencias de error durante la ejecución de las peticiones. Esto se garantiza durante la etapa de implementación pues se chequeó continuamente que las funcionalidades migradas tuvieran éxito.

3.8. Conclusiones parciales

- ✓ La aplicación de técnicas de rendimiento permitió ratificar que los resultados obtenidos estaban en correspondencia con las solicitudes del cliente.
- ✓ Además, las pruebas de Carga y Stress proporcionaron una medida cuantitativa del rendimiento del software.
- ✓ Con la herramienta JMeter se pudo medir los tiempos de respuesta del sistema y se logró de forma general hacer una comparación entre los resultados obtenidos para las diferentes versiones de la migración de SIPAC.

Conclusiones generales

El estudio realizado sobre los principales marcos de trabajo para el desarrollo de aplicaciones web arrojó como resultado que estos presentaban características comunes y que Ext JS en su versión 6.0 presenta las principales potencialidades de los marcos de trabajo estudiados.

Se describieron los procedimientos que se tuvieron en cuenta para realizar la migración de la capa de presentación de los componentes Flujo de Trabajo y Configuración, permitiendo definir a partir de la arquitectura Modelo, Vista, Controlador que propone Ext JS una nueva estructura jerárquica de presentación.

Se realizó la migración de la capa de presentación de los componentes Flujo de Trabajo y Configuración, permitiendo establecer una guía para facilitar dicha migración en las soluciones desarrolladas sobre los componentes que facilitaron el trabajo, garantizando la organización de la misma.

Con la herramienta JMeter fueron medidos los tiempos de respuesta del sistema y se logró de forma general hacer una comparación entre los resultados obtenidos para las diferentes versiones de la migración de SIPAC.

Recomendaciones

El objetivo general de esta investigación fue alcanzado, pero durante el desarrollo de su migración surge la idea que sería recomendable tener en cuenta para su futuro perfeccionamiento del sistema:

- ✓ Continuar con la migración de la capa de presentación de los restantes componentes de SIPAC.

Referencias bibliográficas

APACHE.ORG, 2012. Información General sobre las Nuevas Características en Apache HTTP Server 2.4 - Servidor HTTP Apache Versión 2.5. [en línea]. [Consulta: 24 junio 2016]. Disponible en: https://httpd.apache.org/docs/trunk/es/new_features_2_4.html.

BASS, L., CLEMENTS, P. y KAZMAN, R., 2012. *Software Architecture in Practice*. 3 edition. S.l.: Addison-Wesley Professional. ISBN 978-0-321-81573-6.

CIOLLI, M.E., 2007. *Testing de migración de aplicaciones distribuidas a entornos Web* [en línea]. S.l.: Facultad de Informática. [Consulta: 19 mayo 2016]. Disponible en: http://sedici.unlp.edu.ar/handle/10915/4148.

DÍAZ, F.J., BANCHOFF TZANCOFF, C.M., RODRÍGUEZ, A.S. y SORIA, V., 2012. Usando Jmeter para pruebas de rendimiento. En: 00002, *XIV Congreso Argentino de Ciencias de la Computación*. S.l.: s.n.,

DOCS.SENCHA.COM, 2012a. Ext JS Guides | Sencha Ext JS Guides. En: 00000 [en línea]. [Consulta: 3 junio 2016]. Disponible en: https://docs.sencha.com/extjs/6.0/.

DOCS.SENCHA.COM, 2012b. Ext JS Guides | Supported Browsers. En: 00000 [en línea]. [Consulta: 26 junio 2016]. Disponible en: https://docs.sencha.com/extjs/6.0/supported_browsers.html.

DOJOTOOLKIT.ORG, 2011. Dojo Toolkit. [en línea]. [Consulta: 23 junio 2016]. Disponible en: http://dojotoolkit.org/.

GUTIÉRREZ, J.J., 2014. ?` Qué es un framework Web? En: 00011, *Available in: http://www. lsi. us. es/∼javierj/investigacion_ficheros/Framework. pdf Accessed May* [en línea], vol. 12. [Consulta: 17 mayo 2016]. Disponible en: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

JMETER.APACHE.ORG, 2014. Apache JMeter - Apache JMeter™. En: 00066 [en línea]. [Consulta: 24 junio 2016]. Disponible en: http://jmeter.apache.org/.

JQUERY.ORG, jQuery F.-, 2012. jQuery. En: 00000 [en línea]. [Consulta: 17 mayo 2016]. Disponible en: http://jquery.com/.

JUNTADEANDALUCIA.ES, 2014. Buenas prácticas en el diseño de pruebas de rendimiento | Marco de Desarrollo de la Junta de Andalucía. En: 00000 [en línea]. [Consulta: 26 junio 2016]. Disponible en: http://www.juntadeandalucia.es/servicios/madeja/contenido/libro-pautas/75.

KASIÁN, F. y REYES, N.S., 2012. Búsquedas por similitud en PostgreSQL. En: 00005, *XVIII Congreso Argentino de Ciencias de la Computación* [en línea]. S.l.: s.n., [Consulta: 24 junio 2016]. Disponible en: http://hdl.handle.net/10915/23754.

LAWEBDELPROGRAMADOR.COM, 2013. Buscar en el diccionario informático. En: 00000 [en línea]. [Consulta: 28 junio 2016]. Disponible en: http://www.lawebdelprogramador.com/diccionario/buscar.php?opc=1&charSearch=migracion.

MINGO, P.S., 2013. Para qué sirven las pruebas de rendimiento (I). Introducción. En: 00000, *PEDRO SEBASTIAN MINGO* [en línea]. [Consulta: 25 junio 2016]. Disponible en: http://pedrosebastianmingo.com/para-que-sirven-las-pruebas-de-rendimiento-i-introduccion/.

MOOTOOLS.NET, 2014. MooTools. En: 00015 [en línea]. [Consulta: 17 mayo 2016]. Disponible en: http://mootools.net/.

MUÑOZ TAMAYO, F., ABALLI MORELL, F.R. y YERO TARANCÓN, Y., 2009. Herramienta para la migracion de datos de Microsoft Access a PostgreSQL : modulo Examen Clinico. En: 00000 [en línea], [Consulta: 16 mayo 2016]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_2535_09.

NIELSEN, J., 2014. Response Time Limits: Article by Jakob Nielsen. En: 00001 [en línea]. [Consulta: 22 junio 2016]. Disponible en: https://www.nngroup.com/articles/response-times-3-important-limits/.

OROSA VELÁZQUEZ, Y. y RODRÍGUEZ RODRÍGUEZ, Y., 2011. Guía del proceso de migración del Portal Octavitos. En: 00000 [en línea], [Consulta: 5 junio 2016]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_04599_11.

POSTGRESQL.ORG, 2010. www.postgresql.org.es. En: 00000 [en línea]. [Consulta: 28 junio 2016]. Disponible en: http://www.postgresql.org.es/.

PRESSMAN, R. y MAXIM, B., 2016. *Ingeniería de Sftware - 8va Edición*. S.l.: s.n. ISBN 978-85-8055-534-9.

PROTOTYPEJS.ORG, 2011. Prototype JavaScript framework: a foundation for ambitious web applications. En: 00000 [en línea]. [Consulta: 17 mayo 2016]. Disponible en: http://prototypejs.org/.

PRUEBASDESOFTWARE, 2015. Las pruebas de software forman el único instrumento adecuado para determinar la calidad de software. En: 00000 [en línea]. [Consulta: 28 junio 2016]. Disponible en: http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

RIZZO, J., 2011. Migración de datos. En: 00014, *prezi.com* [en línea]. [Consulta: 23 junio 2016]. Disponible en: https://prezi.com/og5clzmnkboc/migracion-de-datos/.

SANTAMARIA PÉREZ, L., RODRÍGUEZ CRUZ, A.J., RENDÓN ARTOLA, A. y FERNÁNDEZ GONZÁLEZ, M., 2015. «Gestor de reglas y criterios de medidas para evaluar el cumplimiento de objetivos estratégicos en el Sistema de Planificación de Actividades SIPAC». En: 00000 [en línea], [Consulta: 16 mayo 2016]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/8956.

SENCHA, 2012. Ext JS - JavaScript MVC/MVVM framework for cross-platform web apps | Sencha. En: 00000 [en línea]. [Consulta: 23 junio 2016]. Disponible en: https://www.sencha.com/products/extjs/#overview.

SOMMERVILLE, I. y GALIPIENSO, M.I.A., 2005. *Ingeniería del software*. S.l.: Pearson Educación. ISBN 978-84-7829-074-1.

SRIDHAR, S., 2015. A Review on Reuse of Software Components for Sustainable Solutions in Development Process. En: 00002, *IJITR*, vol. 3, no. 2, pp. 1998-2001.

SUBVERSION.APACHE.ORG, (primero), 2015. Subversion Documentation. [en línea]. [Consulta: 24 junio 2016]. Disponible en: https://subversion.apache.org/docs/.

SZYPERSKI, C., BOSCH, J. y WECK, W., 1999. Component-oriented programming. En: 08053, *Object-oriented technology ecoop'99 workshop reader* [en línea]. S.l.: Springer, pp. 184–192. [Consulta: 26 mayo 2016]. Disponible en: http://link.springer.com/chapter/10.1007/3-540-46589-8_10.

TAVÁREZ, D., 2009. Comparación de Frameworks en Javascript. En: 00000, *Maestros del Web* [en línea]. [Consulta: 23 junio 2016]. Disponible en: http://www.maestrosdelweb.com/comparacion-frameworks-javascript/.

VELÁZQUEZ OSORIO, I., CORDERO ESTRADA, L., BAUTA CAMEJO, R. y PÉREZ–BORROTO VIVERO, L., 2015. Migración de la capa de acceso a datos del Marco de Trabajo Sauxe. En: 00000 [en línea], [Consulta: 15 mayo 2016]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/8709.

WEBOPEDIA, 2015. What is Graphical User Interface (GUI)? Webopedia Definition. En: 00000 [en línea]. [Consulta: 23 junio 2016]. Disponible en: http://www.webopedia.com/TERM/G/Graphical_User_Interface_GUI.html.