Universidad de las Ciencias Informáticas

Facultad 3

Migración de la capa de presentación del componente Configuración del marco de trabajo Sauxe utilizando la arquitectura MVC que implementa Ext JS en su versión 4.2.1

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

Autor:

Luis Orlando Proenza Vázquez

Tutores:

Ing. Claudia Bravo Batista

Ing. Alain Fernández Deronceré

La Habana, Junio 2015

"Año 57 de la Revolución"

"Con el genio se inician las grandes obras, pero sólo con el trabajo se les acaban".

A Papi y Mami por ser un ejemplo a seguir, por estar ahí siempre cuando los necesité. Por ayudarme a lograr todos los sueños y metas que me he propuesto. A mi hermana por siempre estar atenta a mí. A toda mi familia que siempre me apoyó en todo, estoy muy feliz por tenerlos.

A todos los amigos que me han acompañado en este viaje hermoso durante estos cinco años, a mi hermano Choy, el Yasi, Dacho, los Abelitos, el Samper, Ale, Andreis, Elizabeth, Dayana, Yusle, Calvo José, Luis Ángel, Chandi, Adrian, Leo, Rayko, el man y todos los compañeros que en un momento u otro compartimos en esta universidad. A mis amigas Iviannis, Yané, Leanne y Yunaldi. Los quiero y les extrañaré muchísimo a todos.

Agradecer a todo el barrio allá en la calle más de caliente de Guisa. A mis amigos de la infancia y de toda la vida Raúl, Carlos, Enilber, Sol y Leander.

Agradecer a la mejor tutora del mundo, mi querida loquita y despeinada Claudia, para ti corazón un millón de gracias por ser la mejor. A todos los compañeros del laboratorio, a la amable Katia, al Bestial, Inoelkis, a la preciosa Odalys y al flojo rubio que hicimos un muy buen equipo de tesis.

Y bueno uno muy especial a mi novia Ana Ilén, la vecina del frente jeje, por siempre darme el ánimo, el amor y la confianza que siempre necesité de ella.

A Papi, Mami y Oli por quererme y ayudarme en todo lo que les pedí alguna vez. A toda mi familia, a mis bellas tías, a mis tíos, a mis primas y a mis abuelos; los que están y los que no: los quiero con el alma y me siento muy orgulloso por tener una familia tan preciosa y unida.

DECLARACIÓN DE AUTORÍA

Por este medio declaro que Luis Orlando Proenza Vázquez es el único autor de este trabajo y se autoriza a la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ___ días del mes de _____ del año 2015**.**

Firma del autor

Luis Orlando Proenza Vázquez

Firma del tutor

Ing. Alain Fernández Deronceré

Firma del tutor

Ing. Claudia Bravo Batista

RESUMEN

El marco de trabajo Sauxe contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Posee una arquitectura en capas, delimitada por la capa de presentación, capa de control, la capa de acceso a datos y la capa de datos, que a su vez presenta en su capa superior el patrón Modelo-Vista-Controlador (MVC).

Entre los componentes de Sauxe se encuentra el componente Configuración, el cual permite gestionar la configuración que se establece en una entidad; brinda facilidades de configuración y adaptación que guiarán a las entidades en el manejo de sus procesos contables, financieros, logísticos, entre otros procesos y funciones administrativas, permitiendo adaptar el sistema a sus peculiaridades y entorno de trabajo.

Actualmente la capa de presentación del componente 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 una versión superior 4.2.1 de Ext JS, con el objetivo de disminuir el consumo de recursos y el tiempo de carga de las interfaces del componente Configuración. Durante el desarrollo de la migración se definieron tres etapas: inicial, implementación y prueba, quedando establecida la capa de presentación con la arquitectura Modelo Vista Controlador que se define en la versión 4.2.1 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*:* Ext JS, marco de trabajo, migración, Configuración, arquitectura MVC

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad se utilizan herramientas informáticas para agilizar la producción de aplicaciones web y brindarles a los programadores y diseñadores una mejor organización y estructura del código fuente, por ejemplo los marcos de trabajo. Los cuales se definen como una arquitectura genérica que proporciona una plantilla ampliable para su aplicación dentro de un dominio [1]. Estos, con el avance de las tecnologías y las exigencias de los usuarios sufren cambios que conducen a la actualización y mejora de su base tecnológica [2].

El Centro de Informatización de Entidades de la Universidad de las Ciencias Informáticas ha desarrollado el marco de trabajo Sauxe, este se encuentra estructurado por niveles o capas, delimitando la capa de presentación, la capa de control, la capa de acceso a datos y la capa de datos, dentro de la capa de presentación es utilizada la librería de JavaScript Ext JS en la versión 2.2.

Sauxe posee una arquitectura orientada a componentes, los cuales presentan características específicas que realizan determinadas funciones, entre ellos se encuentra el componente Configuración. El mismo permite gestionar la configuración que se establece en una entidad; brinda facilidades de configuración y adaptación que guiarán a las entidades en el manejo de sus procesos contables, financieros, logísticos, entre otros procesos y funciones administrativas, permitiendo de este modo, adaptar el sistema a sus peculiaridades y entorno de trabajo [3].

La capa de presentación del componente Configuración se encuentra desarrollada utilizando el marco de trabajo Ext JS, en su versión 2.2. El código fuente JavaScript de la capa de presentación, se encuentra en 13 ficheros de extensión js, con 13 727 líneas de código, distribuido por los componentes: Formato, Ejercicio, Banco y sucursal, Documento, Desglose de moneda, Tasa, Moneda contable, Moneda entidad, Cliente y proveedor, Cliente y proveedor por entidad, País, Unidad de medida y Operación para un total de 13 componentes, el mismo se ha estructurado de diferentes formas sin un estándar que permita organizarlo.

Los componentes que implementa Ext JS se encuentran en el fichero principal ext-all.js, lo que implica que a medida que se cargan las interfaces del componente Configuración, 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 Configuración.

Se realizó una prueba para calcular el tiempo de carga de las interfaces que se encuentran en la capa de presentación del componente Configuración, con la utilización de la herramienta Firefox en su versión 38.0, obteniendo como resultado:

Tabla 1. Tiempo de carga de las interfaces y consumo de recursos con Ext JS v2.2

Componente	Recursos (KB)	Tiempo (s)
Banco y sucursal	38,99	2,95
Documentos primarios	42,70	5,70
Operaciones	24,39	2,58
Ejercicios	68,88	3,13
Formatos	56,90	3,28
País	24,26	2,62
Clientes y proveedores por entidad	72,47	3,23
Clientes y proveedores	43,79	3,07
Unidad de medida	50,71	2,72
Desglose de moneda	27,54	2,71
Tasa	52,59	2,57
Moneda contable	42,96	3,03
Moneda entidad	43,45	3,11
Total	589,63	40,7

Este tiempo es considerado alto pues según el autor Jakob Nielsen, existen tres límites importantes en el tiempo de respuesta [4]:

- 0,1 segundo: es el límite en el cual el usuario siente que está manipulando los objetos desde la interfaz de usuario.
- un segundo: es el límite en el cual el usuario siente que está navegando libremente sin esperar demasiado una respuesta del servidor.
- 10 segundos: es el límite en el cual se pierde la atención del usuario, si la respuesta tarda más de 10 segundos se deberá indicar algún mecanismo por el cual el usuario pueda interrumpir la operación.

A partir de la situación descrita con anterioridad se puede plantear el siguiente **problema a resolver:** ¿Cómo disminuir el consumo de recursos y el tiempo de carga de las interfaces de la capa de presentación del componente Configuración del marco de trabajo Sauxe?

Para resolver el problema planteado se ha propuesto como **objeto de estudio**: el proceso de construcción de interfaces web con la arquitectura Modelo Vista Controlador (MVC) que proporciona Ext JS en su versión 4.2.1.

La investigación se centra en el **campo de acción**: migración del componente Configuración del marco de trabajo Sauxe con la arquitectura MVC que proporciona Ext JS en su versión 4.2.1.

Se propone para darle solución al problema planteado como **objetivo general**: desarrollar la migración de la capa de presentación del componente Configuración del marco de trabajo Sauxe, utilizando la arquitectura MVC que implementa Ext JS en su versión 4.2.1 para disminuir el consumo de recursos y el tiempo de carga de las interfaces.

En la investigación se trazan los siguientes **objetivos específicos**:

1. Elaborar el Marco Teórico de la investigación a partir de la búsqueda y revisión bibliográfica sobre el proceso de construcción de interfaces web con la arquitectura MVC que proporciona Ext JS en su versión 4.2.1.
2. Migrar la capa de presentación del componente Configuración con la nueva arquitectura MVC que propone Ext JS en su versión 4.2.1.
3. Validar la propuesta de solución realizando pruebas de caja negra.
4. Validar la investigación realizando pruebas de rendimiento de carga y estrés.

Se tiene como **idea a defender** de la presente investigación**:** si se desarrolla la migración de la capa de presentación del componente Configuración del marco de trabajo Sauxe, utilizando la arquitectura MVC que implementa Ext JS en su versión 4.2.1 se logrará disminuir el consumo de recursos y el tiempo de carga de las interfaces.

En el desarrollo de la investigación se utilizaron los métodos científicos teóricos y empíricos que se proponen a continuación:

Métodos Teóricos

Son factibles en el estudio de las características poco observables del objeto de investigación. Dentro de este grupo se utilizan:

Histórico – Lógico: se aplicó con el objetivo de realizar un análisis de la información de las versiones de Ext JS, para establecer comparaciones que permitan determinar los cambios producidos entre las mismas.

Analítico – Sintético: se aplicó con el objetivo de realizar un análisis de las aplicaciones informáticas que se dedicaban a la construcción de interfaces web, para seleccionar la adecuada en dar respuesta a la situación problemática propuesta. Además se realizó una descripción de las migraciones de tecnologías informáticas, para la definición de los procedimientos de la migración de la capa de presentación del componente Configuración.

Método Empírico

Observación: se aplicó con el objetivo de observar el tiempo de carga de las interfaces y consumo de recursos del componente Configuración.

La presente investigación se encuentra estructurada en tres capítulos:

En el capítulo 1. Fundamentación teórica: se realiza un análisis de los principales conceptos asociados a la solución propuesta. Se realiza un análisis de las aplicaciones informáticas que se utilizan en la construcción de interfaces web a nivel internacional. Se describen las migraciones de tecnologías realizadas en la UCI y se identifican las herramientas y lenguajes de programación para el desarrollo de la solución propuesta.

En el capítulo 2. Desarrollo de la migración: se realiza una descripción detallada de la migración de la capa de presentación del componente Configuración del marco de trabajo Sauxe, identificando las etapas y las actividades a realizar, siguiendo el procedimiento propuesto por el departamento de Desarrollo de Componentes.

En el capítulo 3. Validación de la migración: se describen los resultados obtenidos en la validación de la propuesta de solución mediante las pruebas de carga y estrés.

Capítulo 1. Fundamentación teórica

En este capítulo se muestran los temas que sustentan la investigación. Se abordan conceptos asociados al marco de trabajo Sauxe, desde la óptica de la arquitectura MVC que proporciona Ext JS en su versión 4.2.1. Debido a que la migración a desarrollarse está delimitada a las soluciones desarrolladas sobre el marco de trabajo Sauxe, se seleccionan las tecnologías y herramientas informáticas a utilizar.

1.1 Conceptos asociados al dominio del problema

1.1.1 Migración

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. En ella se realizan actividades de análisis, diseño o rediseño, modelamiento, creación, configuración, almacenamiento, recuperación, contingencia, desarrollo, prueba, implantación, y puesta en producción. Es actualizar el software a una versión superior, reemplazando una versión instalada de un producto por una versión más reciente del mismo producto.

Mantener los sistemas actualizados es muy importante debido al creciente desarrollo alcanzado por las redes telemáticas e Internet, 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, en ningún caso se implantan mejoras de funcionalidad, como las herramientas nuevas [5].

Los proveedores de software definen diferentes nombres y calificativos a las actualizaciones según las distintas versiones. Estos nombres pueden ser: actualizaciones importantes, de seguridad, de alta prioridad o recomendadas [5].

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

Ventajas:

- ✓ Es posible que ocurra la incorporación de nuevas y últimas tecnologías.
- ✓ Eliminación de errores presentes en las versiones anteriores.
- ✓ Desarrollo de nuevas funcionalidades que enriquecen el manejo de las herramientas.
- ✓ El soporte técnico actualizado sobre la versión actualizada es brindado por los proveedores de la herramienta.

Desventajas:

- ✓ Algo muy importante que se tiene que tener en cuenta a la hora de actualizar es 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.

1.1.2 Arquitectura MVC en Ext JS v4.2.1

El marco de trabajo ExtJS en su versión 4.2.1 introduce el patrón arquitectonico Modelo Vista Controlador que permite organizar el código fuente y la cantidad de código que se debe escribir, a continuación se muestra cómo se define la arquitectura:

Arquitectura MVC [6]**:**

Modelo: es una colección de campos y datos (por ejemplo, un modelo de usuario con nombre de usuario y las contraseñas). Estos persisten a través del paquete de datos, y puede estar vinculado a otros modelos a través de asociaciones.

Vista: es cualquier tipo de componente ventana, árbol y panel definidos por Ext JS.

Controladores: se escribe el código que hace a la aplicación funcionar, ya sea el renderizado[1] de vistas, creación de instancias de modelos, o cualquier otra lógica de aplicación.

1.1.3 Tiempo de carga de las interfaces

Tiempo que demora la aplicación en mostrar las interfaces al usuario, para que este pueda trabajar cómodamente y realice su tarea lo más rápido posible.

Carga dinámica: las clases se pueden cargar dinámicamente y de forma asíncrona en tiempo de ejecución basándose simplemente en el árbol de dependencias que Ext JS gestiona internamente. Esto significa que usted puede dejar de incluir etiquetas en las páginas HTML[2] y simplemente dejar que el cargador de clases busque lo que su página tiene que ejecutar. Esto es muy útil en el entorno de desarrollo, cuando la dependencia y la flexibilidad son más importantes que la velocidad de carga de las interfaces [7].

1.1.4 La interfaz gráfica de usuario (GUI[3])

En términos de informática ha sido definida como un tipo de entorno que representa en la pantalla programas como archivos y opciones por medio de íconos como menús y cuadros de diálogos. El usuario puede seleccionar y activar estas opciones apuntando y haciendo clic con el ratón o, frecuentemente, con el teclado. Un elemento particular (tal como una barra de desplazamiento)

[1] *Render* en Inglés. Es un término usado en jerga informática para referirse al proceso de generar una imagen o vídeo mediante el cálculo de iluminación GI partiendo de un modelo en 3D.

[2] Siglas de *HyperText Markup Language*.

[3] *GUI: Por sus siglas en inglés Graphics User Interface*.

trabaja de igual forma para el usuario en todas las aplicaciones, pues la interfaz gráfica de usuario proporciona rutinas de software estándar; las aplicaciones invocan a estas rutinas con los parámetros específicos en lugar de intentar reproducirlas [8].

1.1.5 Consumo de recursos

Es la acción de utilizar cualquier elemento de un sistema informático o una red, como una unidad de disco, una impresora o la memoria, que puede ser asignada a un programa o a un proceso durante su ejecución. También puede ser el elemento de código o dato que puede ser usado por más de un programa o en más de una parte de un programa, como un cuadro de diálogo, un efecto de sonido, o una fuente en un entorno de ventanas. Los recursos pueden también ser copiados o pegados de un programa a otro, normalmente por un programa de utilidad especializado denominado editor de recursos [9].

1.2 Tecnologías para el desarrollo de interfaces de usuarios

1.2.1 Ext JS

Ext JS es una librería JavaScript que permite construir aplicaciones complejas en Internet [10]. Es desarrollado por Sencha, es orientado a objeto, además utilizan patrones de diseño y lo más importantes la versatilidad de heredar de sus componentes básicos para crear componentes personalizados [11].

Esta librería incluye:

- ✓ Modelo de componentes extensibles.
- ✓ Un API fácil de usar.
- ✓ Licencias código abierto, comercial y revendedor.

Otro punto a tener en cuenta son las licencias; tiene tres tipos [12]:

- ✓ Licencia comercial: esta licencia se debe comprar cuando necesitas 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 GNU GPL V3.
- ✓ Licencia revendedor: este tipo de licencia es necesaria adquirirla cuando se desea realizar una librería basada sobre Ext JS.

Tabla 2. Versiones de Ext JS [13]

Versión	Fecha de lanzamiento	Descripción
3.0	10-09-2009	Mejoras en consistencia y manejo de memoria.
3.1	17-12-2009	Mejoras en desempeño en Internet Explorer. Optimización de Layouts. Nuevos componentes como el TreeGrid
3.2	07-05-2010	Nuevos elementos como SliderTip, SliderField. Mejora de desempeño en Box Layouts, AnchorLayout y ColumnLayout. Ordenamiento y filtrado múltiple en elementos Stores.

		Transiciones animadas para elementos DataView.
3.3	11-10-2010	Agregó los elementos PivotGrid, ActionColumn y nuevos componentes para el manejo de calendarios.
4.0	26-05-2011	Incluye una refactorización de todo el macro de trabajo entre lo que cabe destacar una nueva estructura de clases y carga dinámica de objetos, paquete de datos, nuevos gráficos y temas. Se introdujo soporte para el patrón Modelo Vista Controlador.
4.2.0	13-02-2013	Incorporación de mejoras considerables en módulos tales como Charts, Cmd, Direct, Grid, Theme, aunque principalmente se centran en el trabajo con los temas y en particular Neptune, además se trabajó en la corrección de 72 errores detectados de forma general.
4.2.1	16-05-2013	Se libera de conjunto con Sencha Cmd 3.1.2, incorporan mejoras al core permitiendo el cálculo en gigabytes a través de Ext.util.Format.fileSize, así como mejoras a módulos tales como Layouts, Locale y Misc. Además de la corrección aproximada de 133 errores detectados.
5.0	3-04-2014	Se introduce una nueva arquitectura MMVC (Modelo – Vista - ViewModel), permitiendo escribir el código fuente de forma que se pueda conectar el modelo con la vista, es decir que se actualiza el modelo cuando se modifica la vista.

1.2.2 Librería Prototype

En el 2005, con la popularización de la tecnología basada en el objeto XMLHttpRequest, surgieron nuevas ideas para interfaces. Con él, surge la librería Prototype que ayuda sensiblemente a la implementación de AJAX en los sitios. Prototype resultó ser una librería muy efectiva y fácil de usar, además de integrarse independientemente del proyecto *Ruby on Rails*, por lo que se hizo muy popular. Prototype es una librería que marcó un paso adelante en lo que respecta al desarrollo de páginas ágiles e interactivas [14].

El cambio en el mundo de la programación web, hace uso de las bondades 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 *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 [15].

La parte más grande de la librería Prototype son sus extensiones de DOM. Prototype agrega muchos métodos de conveniencia. Todos los elementos de DOM tienen los métodos de extensión de Prototype incorporado.

1.2.3 Librería JQuery

Es una librería de JavaScript, rápida y concisa que simplifica el trabajo con documentos HTML. 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, como son añadirle funcionalidad, tanto al código como al resto de los elementos [16].

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.

Normalmente, 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.

También se le añaden *plugins* que le da versatilidad. Tiene un núcleo muy pequeño de solo 15 KB, que es muy estable. JQuery y virtualmente todos sus *plugins* están encogidos dentro del *namespace* del JQuery. Como una regla general, se guardan los objetos globales también dentro del *namespace*, para lograr una mejor integración entre JQuery y cualquier otra biblioteca (Prototype, MooTool, o Ext JS) [17].

1.2.4 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 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 partiera de cero [18].

Las ventajas de utilizar este interesante marco de trabajo son [19]:

- ✓ 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.

Sus desventajas son:

- ✓ 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.
- ✓ Aprender a trabajar con el marco de trabajo resulta complicado porque los ejemplos que existen del mismo son complicados.

1.3 Análisis de las tecnologías

En el análisis y resumen de las herramientas o marco de trabajo que permiten construir interfaces web, se tiene en cuenta las características y funcionalidades de las mismas, 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 4.2.1 en la capa de presentación.

Además mencionar que Ext JS brinda una arquitectura MVC que permite organizar y reducir el código fuente de la aplicación y tiene un mejor control de los componentes de la librería. La mayoría de los especialistas que desarrollan en Sauxe, tienen conocimientos sobre ExtJS y existen componentes ya construidos en la versión 4.2.1 con la arquitectura MVC en Sauxe.

El ingeniero en informática Andrés Nieto Porras realizó una prueba para evaluar el tiempo en que se demora cargar las interfaces entre los marcos de trabajo: Prototype, JQuery, Mootools y Ext JS, resultando Ext JS el más rápido con un tiempo de 44.499 milisegundos, como se muestra en la figura 1 [20].

Es destacable mencionar que los marco de trabajo Prototype, JQuery y Mootools también presentan características comparables como la reusabilidad, la simplicidad del código y la gran variedad de componentes útiles que poseen. Por lo antes mencionado se selecciona Ext JS en la versión 4.2.1 como marco de trabajo, para dar solución a la situación problemática planeada.

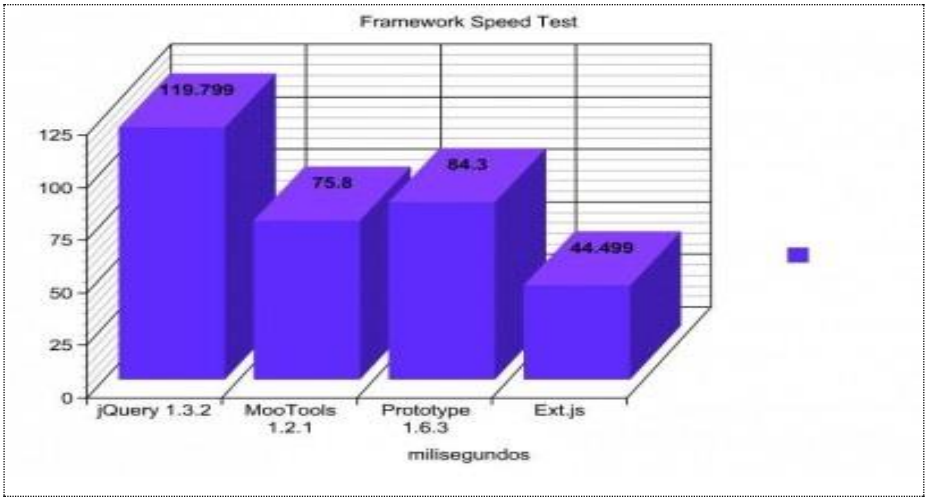


Figura 1. Prueba de velocidad de marco de trabajos

1.4 Metodologías para realizar migración de tecnologías

1.4.1 Pilares para ejecutar el proceso de migración

Una migración debe apoyarse en tres pilares básicos, a saber [21]:

1) Una metodología.
2) Un conjunto de herramientas.
3) 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 obtener 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 ser necesarias 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.

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

Como parte de todo proceso de desarrollo de software es necesario seguir etapas por las cuales se transita hasta llegar al producto final. La investigación, se guiará por los procedimientos utilizados desde el 2013 para la migración de la capa de acceso a datos, los cuales están 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, este cuenta con tres etapas las cuales se describen a continuación [22]:

- ✓ 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 carpeta 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: su objetivo es supervisar que la implementación se realizó satisfactoriamente probando todas las funcionalidades migradas.

1.4.3 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 [23].

- ✓ 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, de mayor 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.

1.4.4 Guía de implementación con Ext JS 4.2.1 aplicando la arquitectura MVC en la capa de presentación del marco de trabajo Sauxe

Se describe la etapa de implementación donde se definen los procedimientos que se realizan para migrar la capa de presentación del marco de trabajo Sauxe con Ext JS. Se realiza el estudio del patrón MVC y de la estructura del marco de trabajo Sauxe.

También se realiza la instalación y configuración del marco de trabajo Ext JS, después de conocer los requisitos de instalación del mismo. Además se realiza la configuración del fichero phtml,

realizar una instancia de la clase Application, definir un controlador, definir una vista, crear la vista panelprincipal.js, definir un almacén y definir un modelo. De todos ellos pueden ser uno o muchos los creados; es decir, se pueden definir una o varias vistas según lo requiera el diseño, al igual pasa con los controladores los modelos y los almacenes.

1.5 Análisis de las guías de migración consultadas

Se realizó 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 4.2.1 de Ext JS, como se muestra en la tabla 3.

Tabla 3. Análisis de la metodología de la migración

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 va a realizar los cambios de tecnología. ✓ Implementar el código fuente de la capa definida. ✓ Evaluar que de 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 Ext JS 4.2.1 aplicando la arquitectura MVC en la capa de presentación del marco de trabajo Sauxe.	Implementación	✓ Importar los ficheros principales de la versión 4.2.1 de Ext JS. ✓ Definir la nueva estructura de directorio para introducir la arquitectura MVC definida por Ext JS en su versión 4.2.1. ✓ Configurar los ficheros de extensión phtml para importar los ficheros principales de la librería de Ext JS. ✓ Realizar la instancia a la clase Application 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 Tecnologías y herramientas para el desarrollo

1.6.1 Marco de trabajo Sauxe

El marco de trabajo Sauxe da solución a escenarios o aspectos arquitectónicos como: gestión y configuración dinámica de caché, integración de componentes de forma distribuida o no distribuida, acceso a bases de datos a través de una capa de abstracción, también gestiona la concurrencia de recursos, administración centralizada de transacciones, gestión dinámica de las trazas generadas por los sistemas, implementación de mecanismos de autenticación y autorización, implementación de mecanismos de mensajería y control de excepciones, gestión y configuración dinámica de precondiciones, pos condiciones y validaciones de variables [24].

Cuenta con una arquitectura en capa que a su vez presenta en su capa superior una arquitectura Modelo Vista Controlador. Sauxe contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Siguiendo el paradigma de independencia tecnológica por el cual apuesta el país, reutiliza las siguientes tecnologías libres [24]:

1.6.2 Servidor web

El servidor web Apache 2.4.7 está desarrollado bajo el criterio *Open Source*. Funciona en múltiples sistemas operativos, haciéndolo prácticamente universal. Es un servidor altamente configurable de diseño modular. Entre sus principales características se encuentran [25]:

- ✓ Es multiplataforma ya que es compatible con muchos Sistemas Operativos.
- ✓ Presenta una alta configuración en la creación y gestión de *logs*[4]. Apache permite la creación de ficheros de *log* a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor.
- ✓ Extensible: gracias a que es modular se han desarrollado diversas extensiones entre las que destaca PHP[5].

1.6.3 Gestores de bases de datos

PostgreSQL 9.1 es un sistema de gestión de bases de datos. Utiliza un modelo cliente/servidor. Incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos,

[4] Es un registro oficial de eventos durante un rango de tiempo en particular.

[5] Siglas de *HyperText Pre-processor.*

funciones, restricciones, disparadores, reglas e integridad transaccional. Otras características que posee [26]:

- ✓ Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, entre otros.
- ✓ Soporte de protocolo de comunicación encriptado por SSL[6].
- ✓ PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL, similar al de Oracle, PL/SQL.

PgAdmin III es una herramienta de código abierto para la administración de bases de datos PostgreSQL y derivados. Incluye una interfaz administrativa gráfica y una herramienta de consulta SQL. Permite desde la ejecución de consultas SQL simples, hasta la elaboración de bases de datos complejas, con el apoyo de las últimas características de PostgreSQL. No requiere ningún controlador adicional para comunicarse con el servidor de base de datos. La aplicación se encuentra desarrollada por una comunidad de especialistas en base de datos de todo el mundo y se encuentra disponible en más de 30 idiomas [26].

1.6.4 Control de versiones

RapidSVN 1.6.6 es un cliente gráfico para Subversion. Es fácil de usar, tanto para los que conocen Subversion como para los que empiezan, pudiendo acceder a direcciones SVN, subir y descargar contenido y sincronizarlo con el servidor original, comprobar su estado, crear y fusionar direcciones [27].

1.6.5 IDE de desarrollo

IDE[7]NetBeans 8.0 es un entorno de desarrollo integrado de código abierto para desarrolladores de software. Cuenta con todas las herramientas necesarias para crear aplicaciones profesionales de escritorio, empresariales, web y aplicaciones móviles con la plataforma Java, así como con C / C + +, PHP, JavaScript y Groovy. NetBeans recibe el soporte integrado para lenguajes dinámicos, todo en una herramienta de gran alcance. El editor de PHP de NetBeans ofrece plantillas de código y generación (*getters* y *setters*), la refactorización, información sobre herramientas de parámetros, consejos y soluciones rápidas, y la finalización de código. El IDE también ofrece un completo editor de HTML, JavaScript y CSS, con la ventaja de proveer un resaltado de sintaxis completo, completamiento de código inteligente y la comprobación de errores para HTML, CSS y JavaScript, incluyendo HTML 5, JavaScript 1.7 [28].

[6] Siglas de “Capa de Conexión Segura” por su significado en español.

[7] Por sus siglas en inglés *Intregrated Development Environment*. Traducción: Entorno de desarrollo integrado.

1.7 Componente Configuración

El componente Configuración permite gestionar toda la configuración que se establece en una entidad; brinda facilidades de configuración y adaptación que guiarán a las entidades en el manejo de sus procesos contables, financieros, logísticos, entre otros procesos y funciones administrativas, permitiendo de este modo, adaptar el sistema a sus peculiaridades y entorno de trabajo [3].

El componente Multimoneda se ocupa de las operaciones que se realizan referentes a temas monetarios y al uso de la dualidad monetaria. Permite que las entidades definan y empleen diversas monedas, según lo requieran, en la ejecución de transacciones u operaciones económicas y su registro. Igualmente posibilita la gestión de las tasas de cambios y la reevaluación de cuentas, entre otras funcionalidades, que son esenciales para llevar el control de la gestión económica y del estado financiero en cualquier entidad [3].

Cada entidad define las monedas que utilizará en sus operaciones, seleccionándolas de un nomenclador que contiene todas las monedas reconocidas nacional e internacionalmente, y especifica los tipos de billetes y monedas en que se puede desglosar físicamente una moneda. Las tasas de conversión o cambio, son contratadas por los clientes o proveedores, estas rigen la reevaluación de cuentas y se aplican a la moneda, en relación con la moneda contable y/o alternativa que se defina, de las monedas ya establecidas. El registro contable de las operaciones, emite el valor de la mayoría de las transacciones que se ejecuten en la entidad [3].

Conclusiones del capítulo

En el presente capítulo se abordaron los principales conceptos asociados al dominio del problema permitiendo analizar y resumir los términos de gran envergadura en la investigación, brindando un mayor conocimiento de las características y repercusión que tiene la migración de una tecnología menor a mayor. Es significativo mencionar los elementos de la arquitectura MVC en Ext JS en su versión 4.2.1 que se identificaron, permitiendo tener una mejor visión de cómo estructurar el código fuente que se implementará durante construcción de la solución. Además se realiza un análisis donde quedó reflejado que la opción más adecuada para darle solución a la problemática planteada es la utilización del *marco de trabajo* Ext JS en su versión 4.2.1, brindó un importante conocimiento en cuanto a las ventajas que presentan los otros *marco de trabajos* abordados. Por último, quedo explícito las tecnologías que se utilizaron en la construcción del componente Configuración, haciendo especial hincapié en la librería Ext JS utilizada en la construcción de las interfaces de usuarios del marco de trabajo Sauxe.

Capítulo 2. Desarrollo de la migración

Introducción

En este capítulo se describe la migración de la capa de presentación del componente Configuración del marco de trabajo Sauxe, utilizando la arquitectura Modelo Vista Controlador que implementa Ext JS en su versión 4.2.1.

2.1 Desarrollo de la migración de la capa de presentación

La migración de la capa de presentación del componente Configuración del marco de trabajo Sauxe, para la versión 4.2.1 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 4.2 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 debe 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 [29].

Por último, 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 del componente Configuración.

2.2 Etapas de la migración

La migración de la capa de presentación del componente Configuración del marco de trabajo Sauxe, se desarrolla en tres etapas: inicial, implementación y prueba.

Etapa inicial**:** se realiza un análisis de la arquitectura del marco de trabajo Sauxe y de la estructura del componente 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 4.2.1, se realizan los cambios pertinentes en la estructura de la capa de presentación del componente de Configuración siguiendo la arquitectura MVC que define Ext JS en su versión 4.2.1.

Etapa de implementación**:** se describen los procedimientos para realizar la migración de la capa de presentación del componente Configuración para la versión superior 4.2.1 de Ext JS.

Etapa de prueba**:** se ejecutan las pruebas de caja negra con el objetivo detectar errores en las interfaces de usuarios construidas utilizando la versión 4.2.1 de Ext JS.

2.3 Etapa inicial

En la etapa inicial se definen los siguientes procedimientos que permitirán establecer la arquitectura MVC en la capa de presentación del componente Configuración, definida en la versión 4.2.1 de Ext JS

Procedimientos definidos en la etapa inicial

- ✓ Analizar la arquitectura del marco de trabajo Sauxe para delimitar la capa de presentación donde se ejecuta la migración.
- ✓ Construir el diagrama de componentes que permite representar el componente Configuración y la relación con los componentes que lo forman.
- ✓ Identificar los requisitos de instalación del marco de trabajo Ext JS.
- ✓ Importar los ficheros principales del marco de trabajo Ext JS en su versión 4.2.1, en Sauxe.
- ✓ Definir la nueva estructura de directorio para introducir la arquitectura MVC definida por Ext JS en su versión 4.2.1 en la capa de presentación del componente Configuración.

2.3.1 Análisis de la arquitectura de Sauxe

La arquitectura de Sauxe está basada en capas, aunque en una de sus capas implementa el patrón MVC. Está compuesta básicamente por cinco niveles o capas [30]:

1. **Capa de presentación:** en esta capa se emplea las facilidades que brinda el marco de trabajo Ext JS para la construcción de interfaces amigables a la vista de los usuarios. Ext JS permite crear interfaces de usuario bastante funcionales, fáciles de usar y atractivas, similar a una aplicación de escritorio, lo que permite a los desarrolladores web concentrarse en la funcionalidad en lugar de las advertencias técnicas y las incompatibilidades respecto a los navegadores.
2. **Capa de control o negocio:** en esta capa se emplea el patrón arquitectónico Modelo Vista Controlador. De forma vertical al modelo descrito hasta este momento, estarán los aspectos fundamentales del sistema así como el encargado del tratamiento de la seguridad a nivel de aplicación web.
3. **Capa de acceso a datos:** en esta capa estará presente el marco de persistencia de datos de Doctrine, para la comunicación con el servidor de datos mediante el protocolo PDO[8], también estará un Persistidor de Configuración que es el encargado de comunicarse vía XML con los ficheros de configuración del sistema denominado FastResponse.
4. **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.
5. **Capa de servicio:** en esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí.

[8] **PDO**: acrónimo del inglés *PHP Data Objects*, es una extensión que provee una capa de abstracción de acceso a datos para PHP 5.

2.3.2 Diagrama de componentes

El componente Configuración se encuentra estructurado de manera que facilite la reutilización de los diferentes componentes y el entendimiento de la lógica de negocio para todos los programadores que desarrollen sobre él. A continuación en la figura 2, se construye el diagrama de componentes que permite representar el componente Configuración y la relación existente entre los componentes que lo forman.

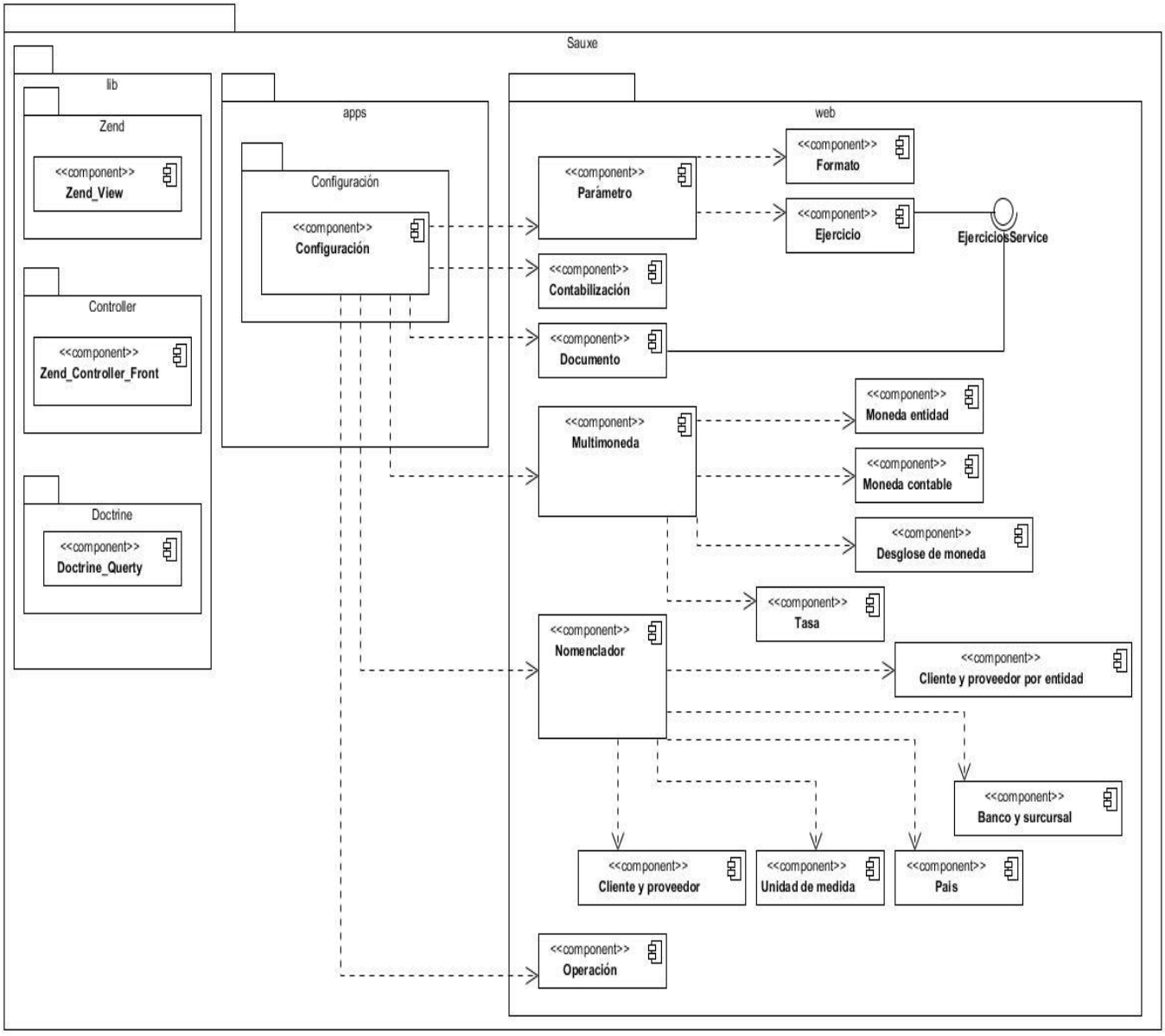


Figura 2 Diagrama de componentes

El componente Configuración está relacionado con los componentes Parámetro, Contabilización, Documento, Multimoneda, Nomenclador y Operación, y se encuentra dentro del paquete Configuración. El componente Parámetro está relacionado por los componentes Formato y Ejercicio,

este último, brinda servicios al componente Documento. El componente Multimoneda está relacionado por los componentes Moneda entidad, Moneda contable, Desglose de moneda y Tasa. El componente Nomenclador se encuentra relacionado con los componentes Cliente y proveedor por entidad, Cliente y proveedor, Banco y sucursal, País y Unidad de medida. El componente Documento tiene como propósito establecer los documentos que se consideran primarios para cada entidad.

2.3.3 Requisitos de instalación de la librería Ext JS v4.2.1

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 [31]:

- Google Chrome 10+
- Apple Safari 5+
- Mozilla Firefox 4+

Se recomienda que se utilice como servidor web la herramienta Apache HTTP Server.

2.3.4 Pasos para importar el marco de trabajo Ext JS v4.2.1, en Sauxe

A continuación se describen los pasos para importar Ext JS v4.2.1, en Sauxe.

1. Incluir el archivo ext-all.js en la dirección web/lib/ExtJS/es/js/ext-all-4.2.1, pero como se desea mantener ambas versiones del marco de trabajo, se modifica el nombre del fichero para que no se sobrescriba el mismo, en este caso se nombra ext-all-4.2.1.js como se muestra en la figura 3.
2. Incluir el archivo ext-all.css en la dirección web/lib/ExtJS/es/js/ext-all-4.2.1, como se muestra en la figura 4.

Figura 3. Importar fichero ext-all.js

Figura 4. Importar fichero ext-all.css

2.1.1 Estructura de directorio para introducir la arquitectura Modelo Vista Controlador definida por Ext JS en su versión 4.2.1, en la capa de presentación del componente Configuración

El primer cambio que se realiza en la capa de presentación del componente Configuración es la estructura de directorio, con el objetivo de introducir la arquitectura Modelo Vista Controlador definida por Ext JS. Como se muestra en la figura 5, en la versión 2.2 de Ext JS se crea un solo fichero.js, en este caso gestionarbancos.js, donde se localiza el código JavaScript de Ext JS para la construcción de la interfaces de usuarios; mientras que en la versión 4.2.1 se sigue una estructura de directorio unificado y se realiza el diseño Modelo Vista Controlador, todas las clases se colocan en el directorio app, que contiene a su vez los directorios modelos, vistas, controles y almacenes, para generar el espacio de nombres de los modelos (*model*), las vistas (*view*), los controladores (*controller*) y los almacenes (*stores*).

Además se crea el fichero app.js, en el cual se encuentra en el mismo nivel que el directorio app y se ejecuta una instancia de la clase Application.

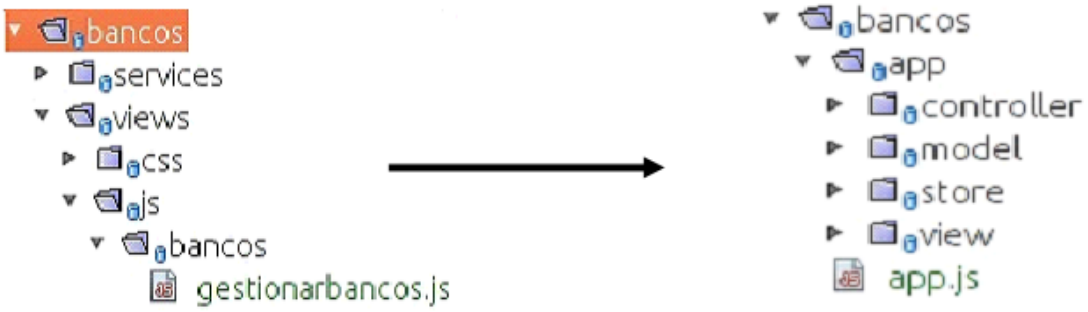


Figura 5. Estructura de directorio

Como resultado de la presente investigación se construyen los diagramas de clases del diseño usando estereotipos web, representando solamente las clases y las relaciones entre las mismas, del código JavaScript. Los diagramas se construyeron por cada componente definido en el diagrama de componentes propuesto en la figura 6. A continuación se muestra el diagrama de clases del diseño del componente Banco y sucursal, ver en los Anexos (3-14) los restantes diagramas.

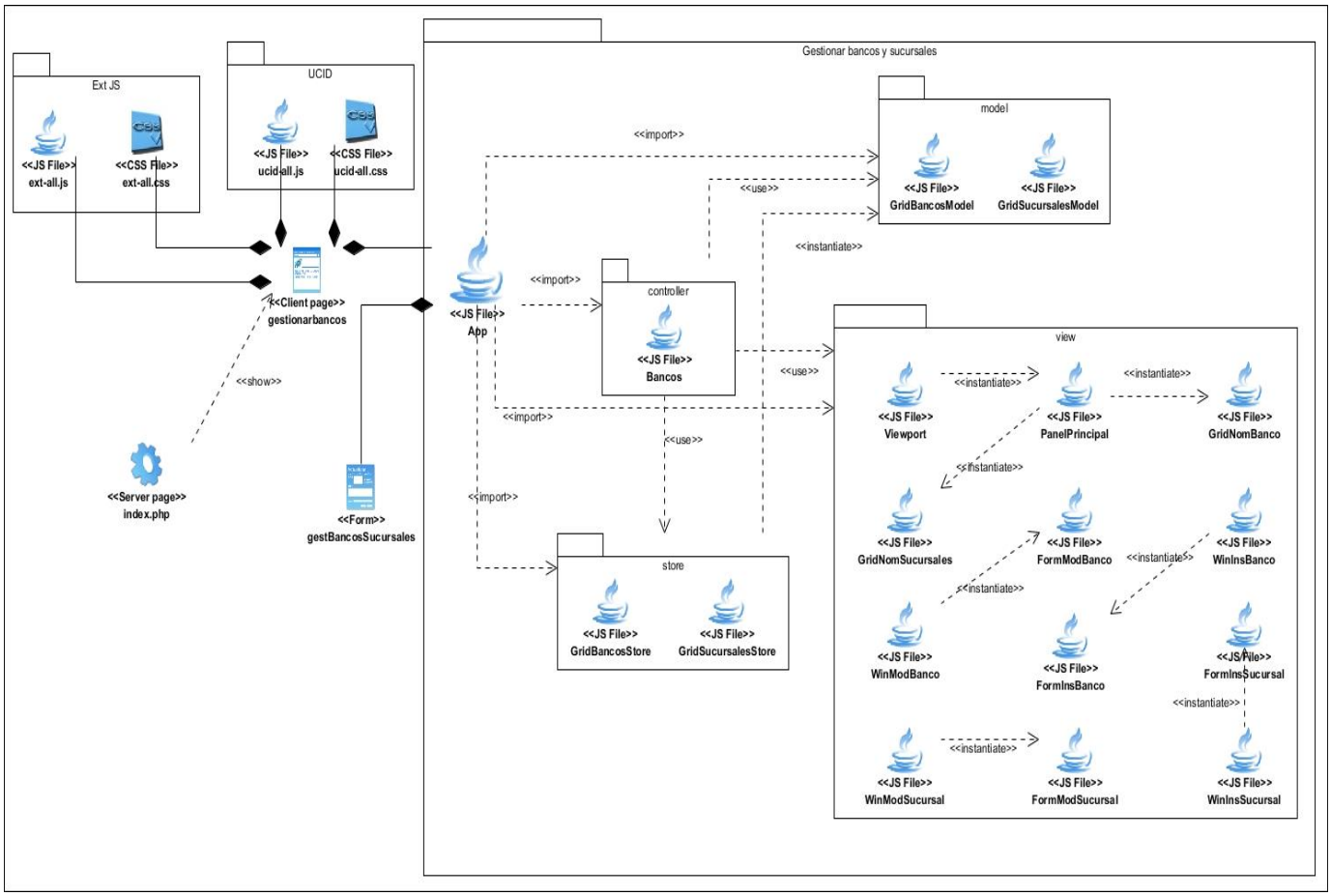


Figura 6. Diagrama de clases del diseño del componente Banco y sucursal

2.4 Etapa de implementación

En la etapa de implementación se describen los siguientes procedimientos que permitirán implementar la migración de la capa de presentación del componente Configuración para la versión superior 4.2.1 de Ext JS.

Procedimientos definidos en la etapa de implementación

- ✓ Configurar los ficheros de extensión phtml para importar los ficheros principales de la librería de Ext JS, para cada uno de los componentes definidos en el diagrama de componentes propuesto en la figura 6.
- ✓ Realizar la instancia a la clase Application que contiene la configuración global, para cada uno de los componentes definidos en el diagrama de componentes propuesto en la figura 6.
- ✓ Definir los controladores que permiten gestionar los eventos de las vistas, para cada uno de los componentes definidos en el diagrama de componentes propuesto en la figura 6.
- ✓ Definir las vistas que representan los componentes definidos por Ext JS, para cada uno de los componentes definidos en el diagrama de componentes propuesto en la figura 6.
- ✓ Definir los almacenes que hacen referencia a todos los modelos del componente, para cada uno de los componentes definidos en el diagrama de componentes propuesto en la figura 6.

2.4.1 Configuración del fichero phtml

Para importar los archivos principales del marco de trabajo Ext JS, se configura el fichero de extensión phtml, a través del <link href /> se importa el fichero ext-all.css y a través de <script src></script> el fichero ext-all-4.2.1.js, en la figura 7 se muestra un ejemplo del código fuente del archivo gestionarbancos.phtml del componente Banco.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Gestionar Bancos</title>
<script type="text/javascript" src="<?php echo $this->dir_ucid;?>js/imporcss.js"></script>
<link href="<?php echo $this->dir_ext_ccs; ?>../default/css/ext-all.css" rel="stylesheet" type="text/css" media="screen"/>
</head>
<body>
<script type="text/javascript" src="<?php echo $this->dir_extjs;?>js/ext-base.js"></script>
<script type="text/javascript" src="<?php echo $this->dir_extjs;?>js/ext-all.js"></script>
<script type="text/javascript" src="<?php echo $this->dir_ucid;?>js/ucid-all.js"></script>
<script type="text/javascript" src="../../views/js/bancos/gestionarbancos.js"></script>
</body>
</html>
```

Figura 7. Archivo de extensión phtml del componente Banco

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

1. gestionarformatos.phtml del componente Formato

2. gestionarejercicios.phtml del componente Ejercicio
3. gestionarbancos.phtml del componente Banco y sucursal
4. documentos.phtml del componente Documento
5. desglose.phtml del componente Desglose de moneda
6. gestionartasa.phtml del componente Tasa
7. gestmonedacontable.phtml del componente Moneda contable
8. gestionarmultimoneda.phtml del componente Moneda entidad
9. gestclientent.phtml del componente Cliente y proveedores por entidad
10. gestproveedclient.phtml del componente Cliente y proveedores
11. paisesentidad.phtml del componente País
12. gestionarum.phtml del componente Unidad de medida
13. operaciones.phtml del componente Operación

2.4.2 Crear la instancia de la clase Application

En la capa de presentación de cada uno de los componentes definidos en el diagrama de componentes propuesto en la figura 6, se crea una 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 y controladores utilizados por este. En la figura 8, se muestra la configuración del archivo app.js del componente Banco, en el cual se crea la instancia de la Application con el espacio de nombres global 'GestBancos'.

```
Interface = {
    init: function() {
        Ext.Loader.setPath({
            'GestBancos.controller': '../../views/js/bancos/app/controller',
            'GestBancos.view': '../../views/js/bancos/app/view',
            'GestBancos.store': '../../views/js/bancos/app/store',
            'GestBancos.model': '../../views/js/bancos/app/model'
        });

        Ext.application({
            name: 'GestBancos',
            autoCreateViewport: true,
            models: ['GridBancosModel', 'GridSucursalesModel'],
            stores: ['GridBancosStore', 'GridSucursalesStore'],
            controllers: ['Bancos']
        });
    }
    },

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

Figura 8. Configuración del archivo app.js del componente Banco

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

1. 'GestFormatos' del componente Formato
2. 'GestEjercicios' del componente Ejercicio
3. 'GestBancos' del componente Banco y sucursal
4. 'GestDocumentos' del componente Documento
5. 'GestDesglose' del componente Desglose de moneda
6. 'GestTasa' del componente Tasa
7. 'GestMonedaCont' del componente Moneda contable
8. 'GestMonedaEnt' del componente Moneda entidad
9. 'GestClientesPE' del componente Cliente y proveedor por entidad
10. 'GestProveedoresC' del componente Cliente y proveedor
11. 'GestPaises' del componente País
12. 'GestGestionarUM' del componente Unidad de medida
13. 'GestOperaciones' del componente Operación

2.4.3 Definir un controlador

Los controladores son los conectores que unen todas las partes de la aplicación entre sí. Todo lo que realmente hacen es escuchar los eventos (por lo general de vistas) y realizar algunas acciones cuando lleguen. Para crear un controlador siguiendo la arquitectura Modelo Vista Controlador que propone Ext JS v4.2.1, se crea un fichero de extensión js y se define que sea de tipo controller, extendiendo de ‘Ext.app.Controller’, ver la figura 9 el ejemplo del código fuente de la definición del controlador del componente Banco. Durante el desarrollo de este procedimiento se definieron los siguientes controladores:

1. Controlador Formatos para el componente Formato
2. Controlador Ejercicios para el componente Ejercicio
3. Controlador Bancos para el componente Banco y sucursal
4. Controlador Documentos para el componente Documento
5. Controlador Desglose para el componente Desglose de moneda
6. Controlador Tasa para el componente Tasa
7. Controlador MonedaCont para el componente Moneda contable
8. Controlador MonedaEnt para el componente Moneda entidad
9. Controlador ClientesPE para el componente Cliente y proveedor por entidad
10. Controlador ProveedoresC para el componente Cliente y proveedor
11. Controlador Países para el componente País

12. Controlador GestionarUM para el componente Unidad de medida
13. Controlador Operaciones para el componente Operación

```
Ext.define('GestBancos.controller.Bancos', {
    extend: 'Ext.app.Controller',
```

Figura 9. Código fuente de la definición del controlador del componente Banco

2.4.4 Definir una vista

Una vista es un componente, por lo general se define como una subclase de un componente de Ext JS. Para crear una vista siguiendo la arquitectura Modelo Vista Controlador que propone Ext JS v4.2.1, 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 Window sería Ext.window.Window, en la figura 10 se muestra un ejemplo del código fuente de la vista Viewport que contiene las restantes vistas del componente Banco, y en la figura 11 se muestra la interfaz de usuario.

```
Ext.define('GestBancos.view.Viewport', {
    extend: 'Ext.container.Viewport',
    layout: 'fit',
    idmio: 'view',
    requires: [
        'GestBancos.view.PanelPrincipal',
        'GestBancos.view.GridNomBanco',
        'GestBancos.view.GridNomSucursales',
        'GestBancos.view.WinInsBanco',
        'GestBancos.view.FormInsBanco',
        'GestBancos.view.WinInsSucursal',
        'GestBancos.view.FormInsSucursal',
        'GestBancos.view.FormModBanco',
        'GestBancos.view.FormModSucursal',
        'GestBancos.view.WinModBanco',
        'GestBancos.view.WinModSucursal'
    ],
    initComponent: function() {
        this.items = {
            xtype: 'panelprincipal'
        };
        this.callParent();
    }
});
```

Figura 10. Código fuente de la vista Viewport

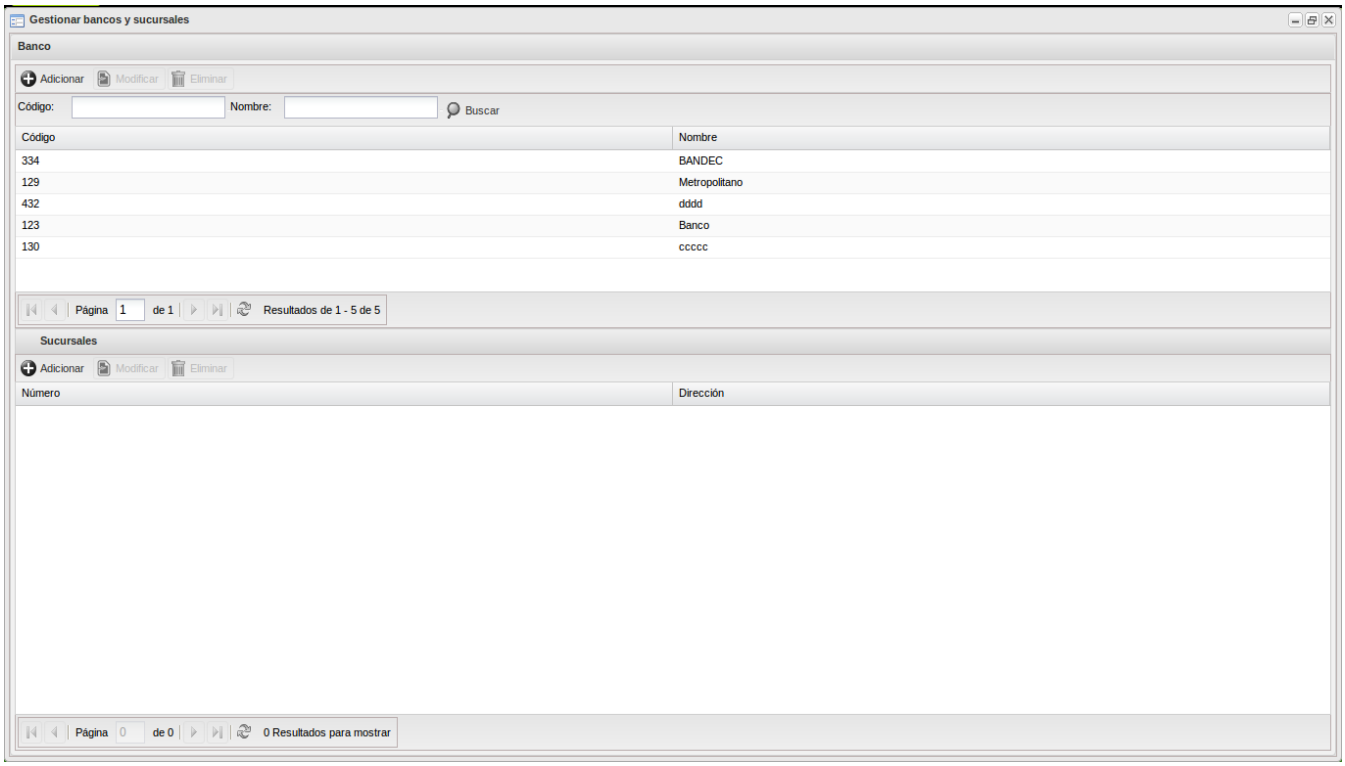


Figura 11. Interfaz de usuario del componente Banco

Durante el desarrollo de este procedimiento se definieron las siguientes vistas:

Tabla 4. Vistas de la capa de presentación

Componente	Vista	Total
Banco y sucursal	Viewport, Panel Principal, FormInsBanco, FormInsSucursal, FormModBanco, FormModSucursal, GridNomBanco, GridNomSucursales, WinInsBanco, WinInsSucursal, WinModBanco, WinModSucursal	12
Documento	FormInsDocumentos, FormModDocumentos, GridNomAddDocumentos, GridNomDocumentos, PanelPrincipal, Viewport, WinInsDocumentos, WinModDocumentos	8
Operación	FormInsOperaciones, FormModOperaciones, GridNomOperaciones, PanelPrincipal, Viewport, WinInsOperaciones, WinModOperaciones	7

Ejercicio	FormInsEjercicios, FormInsPeriodo, FormModEjercicios, FormModPeriodo, GridNomEjercicios, GridNomPeriodo, PanelPrincipal, Viewport, WinInsEjercicios, WinInsPeriodo, WinModEjercicios, WinModPeriodo	12
Formato	FormInsFormatos, FormInsNivel, FormModFormatos, FormModNivel, GridNomFormatos, GridNomNivel, PanelPrincipal, Viewport, WinInsFormatos, WinInsNivel, WinModFormatos, WinModNivel	12
País	EditorFormPais, FormInsPais, GridNomPaises, PanelPrincipal, Viewport, WinInsPais	7
Cliente y proveedor por entidad	FormInsCliente, FormModCliente, GridFormClientes, GridNomClientesPE, PanelPrincipal, Viewport, WinInsCliente, WinModCliente	9
Cliente y proveedor	FormInsProveedor, FormModProveedor, GridNomProveedoresC, PanelPrincipal, Viewport, WinInsProveedor, WinModProveedor	7
Unidad de medida	FormInsUnidad, FormModUnidad, GridNomGestionarUM, PanelPrincipal, Viewport, WinInsUnidad, WinModUnidad	7
Desglose de moneda	FormInsDesglose, GridFormDesglose, GridNomDesglose, PanelPrincipal, Viewport, WinInsDesglose	6
Tasa	FormInsTasa, FormModTasa, GridNomTasa, PanelPrincipal, Viewport, WinInsTasa, WinModTasa	7
Moneda contable	FormInsMoneda, FormModMoneda, GridNomMonedaCont, PanelPrincipal, Viewport, WinInsMoneda, WinModMoneda	7
Moneda entidad	FormInsEntidad, FormModEntidad, GridFormEntidad, GridNomMonedaEnt, PanelPrincipal, Viewport, WinInsEntidad, WinModEntidad	8

2.4.5 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

v4.2.1, 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, en la figura 12 se muestra un ejemplo del código fuente de un almacén del grid del componente Banco.

```
]Ext.define('GestBancos.store.GridBancosStore', {
    extend: 'Ext.data.Store',
    storeId: 'storebanco',
    requires: 'GestBancos.model.GridBancosModel',
    model: 'GestBancos.model.GridBancosModel',
    autoLoad:true
});
```

Figura 12. Código fuente de un almacén del grid del componente Banco

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

Tabla 5. Almacenes

Componente	Almacenes	Total
Banco y sucursal	GridBancosStore, GridSucursalesStore	2
Documento	ComboDocumentosStore, GridAddDocumentosStore, GridDocumentosStore	3
Operación	GridOperacionesStore	1
Ejercicio	GridEjerciciosStore, GridPeriodoStore	2
Formato	GridFormatosStore, GridNivelStore	2
País	EditorPaisStore, GridPaisesStore	2
Cliente y proveedor por entidad	GridClientesPEStore, GridFormClientesStore	2
Cliente y proveedor	GridProveedoresCStore	1
Unidad de medida	ComboUnidadStore, GridGestionarUMStore	2
Desglose de moneda	ComboDesgloseStore, GridDesgloseStore, GridFormDesgloseStore	3
Tasa	ComboClienteStore, ComboContableStore, ComboMonedaStore, ComboTipoTStore, GridTasaStore	5
Moneda contable	ComboMonedaStore, ComboPaisStore,	3

	GridMonedaContStore	
Moneda entidad	ComboMonedaEStore, GridFormEntidadStore, GridMonedaEntStore	3

2.4.6 Definir un modelo

El modelo es una colección de campos y sus datos del store. Para crear un modelo siguiendo la arquitectura Modelo Vista Controlador que propone Ext JS v4.2.1, 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, se configura el proxy que son la forma de cargar y guardar datos de un almacén (Store) o un Modelo en Ext JS v4.2.1. Hay Proxys de AJAX, JSON-P y HTML5 localStorage, entre otros. Aquí se utilizó un simple Proxy de AJAX, y cargará los datos desde la url: 'cargar', 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 successProperty, en la figura 13 se muestra un ejemplo del código fuente de un modelo del grid del componente Banco.

```
Ext.define('GestBancos.model.GridBancosModel', {
    extend: 'Ext.data.Model',
    fields: [
        {name: 'idbanco'},
        {name: 'codigo'},
        {name: 'nombre'}
    ],
    proxy: {
        type: 'ajax',
        url: 'cargar',
        actionMethods: {read: 'POST'},
        reader: {
            totalProperty: "cant",
            root: "datos",
            id: "id"

        }

    }
});
```

Figura 13. Definición de un model

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

Tabla 6. Modelos

Componente	Modelos	Total
Banco y sucursal	GridBancosModel, GridSucursalesModel	2
Documento	GridAddDocumentosModel, GridDocumentosModel	2

Operación	GridOperacionesModel	1
Ejercicio	GridEjerciciosModel, GridPeriodoModel	2
Formato	GridFormatosModel, GridNivelModel	2
País	EditorPaisModel, GridPaisesModel	2
Cliente y proveedor por entidad	GridClientesPEModel, GridFormClientesModel	2
Cliente y proveedor	GridProveedoresCModel	1
Unidad de medida	GridGestionarUMModel	1
Desglose de moneda	GridDesgloseModel, GridFormDesgloseModel	2
Tasa	GridTasaModel	1
Moneda contable	GridMonedaContModel	1
Moneda entidad	GridFormEntidadModel, GridMonedaEntModel	2

2.5 Etapa de prueba

2.5.1 Pruebas de caja negra

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene [32].

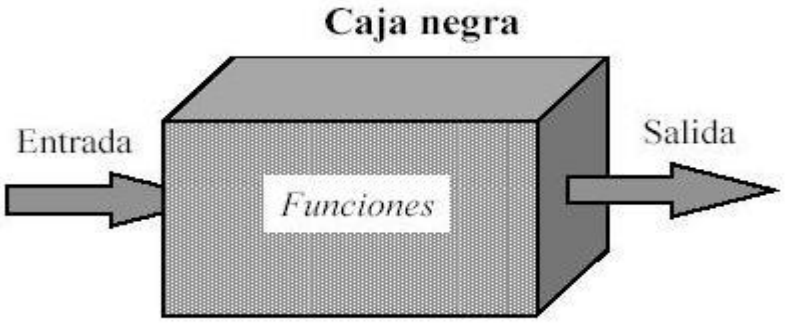


Figura 14. Enfoque de diseño de pruebas de caja negra [32]

Para Pressman, las pruebas de caja negra se centran en los requisitos funcionales del software. Es decir, permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Se trata de un enfoque que intenta descubrir diferentes tipos de errores que no se encuentran con los métodos de caja blanca.

La prueba de caja negra para Pressman [32], intenta encontrar errores de las siguientes categorías:

- ✓ Funciones incorrectas o faltantes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a bases de datos externas.
- ✓ Errores de comportamiento o desempeño.
- ✓ Errores de inicialización y término.

En las pruebas de validación se utilizará el método partición equivalente.

2.5.2 Partición equivalente

Pressman presenta la partición equivalente como un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

El objetivo de la partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases.

Las clases de equivalencia se definen de acuerdo con las siguientes directrices [32]:

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

Para la aplicación de este tipo de pruebas se diseñaron 18 escenarios, con el objetivo de demostrar que las interfaces del componente Configuración, después de realizada la migración de Ext JS a una versión superior, se encontraban sin errores y que las funciones del componente son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto. A continuación en las tablas 2, 3 y 4 se muestran el caso de prueba del requisito “Adicionar banco”, los restantes casos de

prueba se muestran en el expediente del proyecto Sauxe, específicamente en la carpeta Diseños Casos de Prueba del componente Configuración.

Tabla 7. Escenario de prueba del requisito Adicionar banco

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Adicionar banco	Permite registrar un nuevo banco en el sistema.	EP 1.1: Interfaz Principal	– Dar clic en el botón **Adicionar**. – El sistema muestra la Interfaz **Adicionar banco.**
		EP 1.2: Interfaz Adicionar banco	– Llenar los campos Código y Nombre. – Dar clic en el botón **Aceptar**. – Dar clic en el botón **Aplicar,** si se desea adicionar un nuevo banco en el sistema.

Tabla 8. Descripción de variables del escenario de prueba del requisito Adicionar banco

No	Nombre de campo	Tipo	Válido	Inválido
1	Código	*textbox*	Número y Letras	Caracteres especiales
2	Nombre	*textbox*	Letras	Números y caracteres especiales

Tabla 9. Juegos de datos a probar en escenario de prueba del requisito Adicionar banco

Id del escenario	Escenario	Variable 1 Código	Variable 2 Nombre	Respuesta del sistema	Resultado de la prueba
EP 1.2	Adicionar banco	123	Banco Popular de Ahorro	Es posible adicionar un banco en el sistema	Satisfactorio

EP 1.2	Adicionar banco	Código@SS	Banco Popular de Ahorro	El sistema muestra un mensaje de error “Debe insertar valores correctos”	Satisfactorio

2.5.3 Resultados generales de las pruebas funcionales

Los resultados generales de la aplicación de las pruebas de caja negra quedaron definidos que para una primera iteración se detectaron 15 no conformidades, ocho significativas y siete no significativas. Ya en una segunda iteración no se obtuvieron no conformidades. Quedando reflejado en la siguiente gráfica.

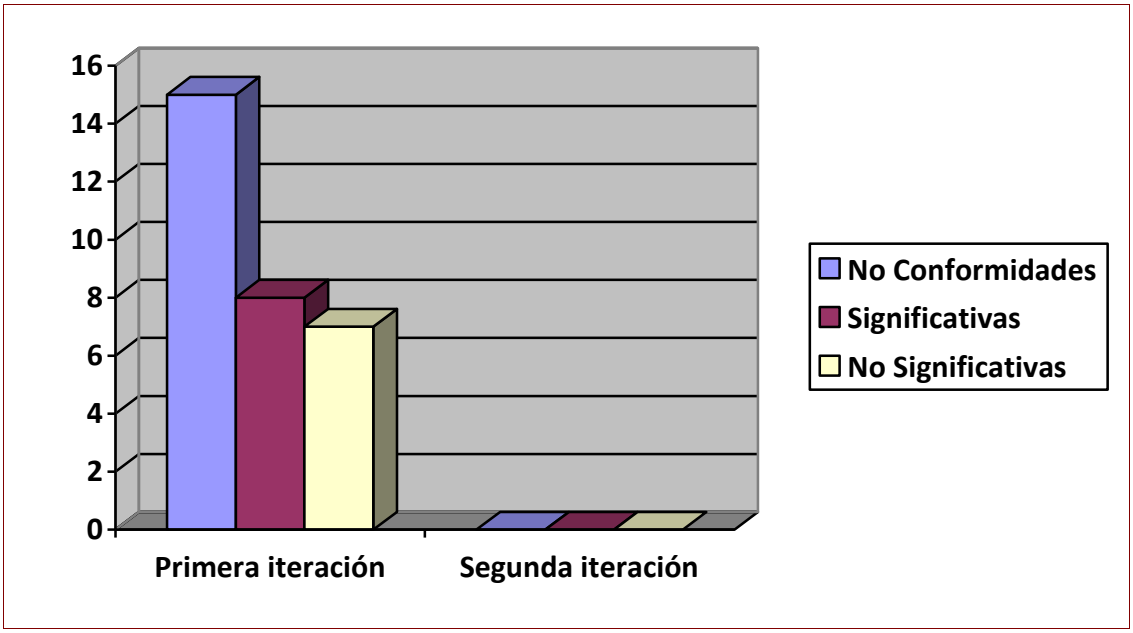


Figura 15. Representación de los resultados de las pruebas funcionales

Conclusiones del capítulo

En el presente capítulo se describieron los procedimientos que se tuvieron en cuenta para realizar la migración de la capa de presentación del componente Configuración, permitiendo definir a partir de la arquitectura Modelo Vista Controlador que propone Ext JS una nueva estructura jerárquica de presentación.

La migración se dividió en tres etapas: inicial, implementación y prueba. En la etapa inicial se realizó un análisis de la arquitectura en capas que tiene el marco de trabajo Sauxe, permitiendo identificar las características de la capa de presentación donde se desarrolló la migración. Se construyó el diagrama de componentes que permitió representar el componente de Configuración integrado en el marco de trabajo Sauxe, y las relaciones que existen entre los componentes que lo forman. Se importaron los ficheros principales de Ext JS en su versión 4.2.1, en Sauxe y quedó establecida la estructura Modelo Vista Controlador en la capa de presentación del componente Configuración permitiendo organizar el código fuente del mismo.

En la etapa de implementación se configuraron 13 archivos phtml, se realizaron 13 instancias de la clase Application, se definieron 13 controladores, 109 vistas, 31 almacenes y 21 modelos, para los componentes Formato, Ejercicio, Banco y sucursal, Documento, Desglose de moneda, Tasa, Moneda contable, Moneda entidad, Cliente y proveedor, Cliente y proveedor por entidad, País, Unidad de medida y Operación para un total de 12 componentes, permitiendo organizar y reducir el código fuente JavaScript por cada uno de los ficheros de extensión js definidos.

Por último, en la etapa de prueba se realiza pruebas unitarias de caja negra, aplicando el método de partición equivalente que permite elaborar casos de pruebas para corroborar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto y que la integridad de la información se mantiene.

Capítulo 3. Validación de la migración

Introducción

En la validación de la investigación se realizan pruebas de rendimiento de carga y estrés, con el objetivo de determinar la disminución del tiempo de carga de las interfaces, realizada la migración de Ext JS a la versión superior 4.2.1 del componente Configuración en el marco de trabajo Sauxe. 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. Los tipos de prueba de rendimiento aplicados son:

Pruebas de carga: se ejecuta para comprender el comportamiento de una aplicación ante una carga determinada. Esta carga puede ser el número de usuarios esperado ejecutando un número de transacciones durante un tiempo determinado. El resultado de esta prueba dará el tiempo de respuesta de todas las transacciones críticas. Si la base de datos, servidor de aplicación también se monitorizan, entonces esta prueba puede mostrar potenciales problemas de botella en la aplicación.

Pruebas de estrés: son utilizadas normalmente para someter a la aplicación al límite de su funcionamiento, mediante la ejecución de un número de usuarios muy superior al esperado. Esta prueba tiene como finalidad determinar la robustez de una aplicación cuando la carga es extrema y ayuda a administradores a determinar si la aplicación se comportará correctamente en dichas situaciones. Otro posible objetivo de este tipo de pruebas es determinar el límite real de la aplicación en cuanto a número de usuarios concurrentes y el número de transacciones por segundo.

3.1 Entorno de pruebas

En esta actividad se debe preparar el sistema para realizarle las pruebas que se estimen convenientes. En la misma se crean los casos de pruebas y se pueden realizar en diferentes condiciones en dependencia de la importancia del sistema. Para realizar dicha prueba se utiliza una computadora con las siguientes características:

Hardware: Intel Core i5-3337U a 1.80 GHz, 4 GB de RAM, Sistema Operativo Ubuntu 14.04, tipo de sistema 64-bit en una partición de 80 GB.

Software: gestor de base de datos: PostgreSQL v9.1, servidor de aplicaciones: Apache 2.4.7, máquina virtual de Java: openjdk-8, navegador web: Mozilla Firefox.

LAN: tarjeta de red 100 Mbps de velocidad.

Una vez establecido el entorno de prueba se carga el componente Banco y sucursal en la versión 2.2 de Ext JS y se da paso a realizarle las pruebas con el JMeter. En otro paso se carga el mismo componente pero en la versión 4.2.1 de Ext JS y se le realizan las pruebas. Por otra parte, se

confeccionan los casos de prueba con el objetivo de guiar las pruebas a las funcionalidades (Anexos 1-2).

3.2 Realizar pruebas de rendimiento

En esta actividad, inicialmente se selecciona la herramienta para las pruebas, teniendo en cuenta que la herramienta seleccionada debe obtener como resultado si se logra o no mejorar el consumo de recursos y el tiempo de carga de las interfaces del componente Configuración al realizar la migración de las vistas.

Para realizar las pruebas en esta investigación se seleccionó la herramienta Apache JMeter en su versión 2.8.2 la cual es una herramienta de carga para llevar a acabo simulaciones sobre cualquier recurso de software. Inicialmente diseñada para pruebas de estrés en aplicaciones web, hoy en día, su arquitectura ha evolucionado no sólo para llevar a cabo pruebas en componentes habilitados en Internet HTTP, sino además en bases de datos, programas en Perl y prácticamente cualquier otro medio [22].

Además, posee la capacidad de realizar desde una solicitud sencilla hasta secuencias de peticiones que permiten diagnosticar el comportamiento de una aplicación en condiciones de producción. En este sentido, simula todas las funcionalidades de un navegador, o de cualquier otro cliente, siendo capaz de manipular resultados en determinada petición y reutilizarlos para ser empleados en una nueva secuencia. El componente principal de JMeter es denominado Plan de Prueba, en él se definen todos los aspectos relacionados con una prueba de carga, como: parámetros empleados por petición, tipo de reportes a generarse con los resultados obtenidos, la posible reutilización de requisiciones compuestas por usuarios, entre otros aspectos [22].

Primeramente 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 16, se inicia la cabecera de la prueba en la que se define la cantidad de usuarios con los que se simula.

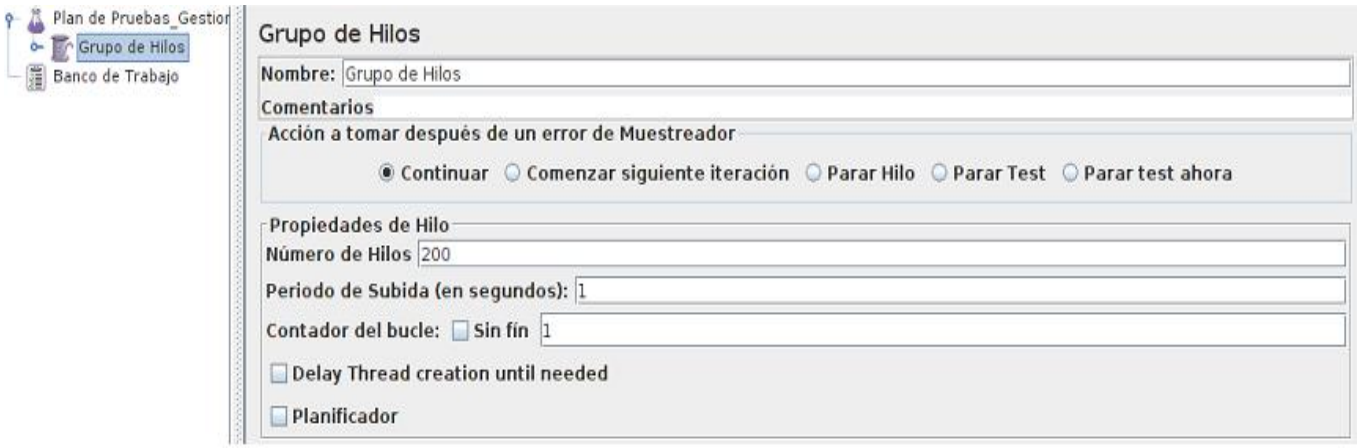


Figura 16. Elemento Grupo de Hilos

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).

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

Figura 17. 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.

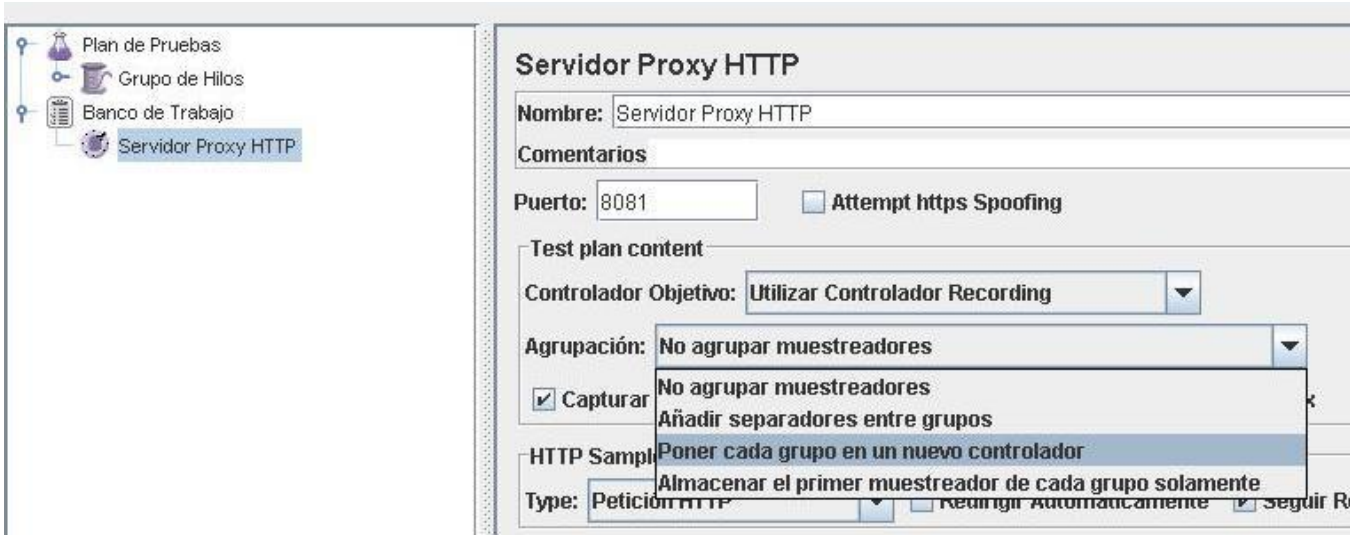


Figura 18. Valores para el Servidor Proxy HTTP

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.

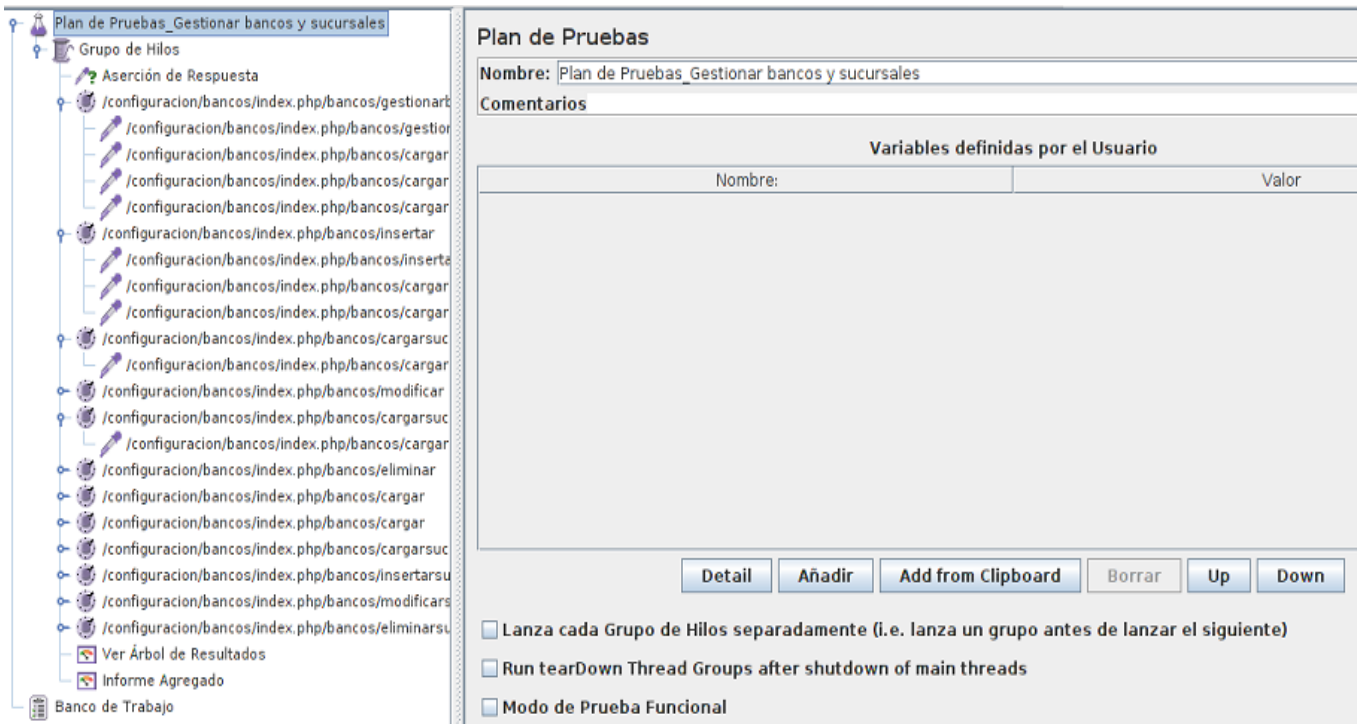


Figura 19. Grabación de navegación de la web

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 [32]; además los datos insertados en el sistema se muestran en los parámetros enviados con la petición.

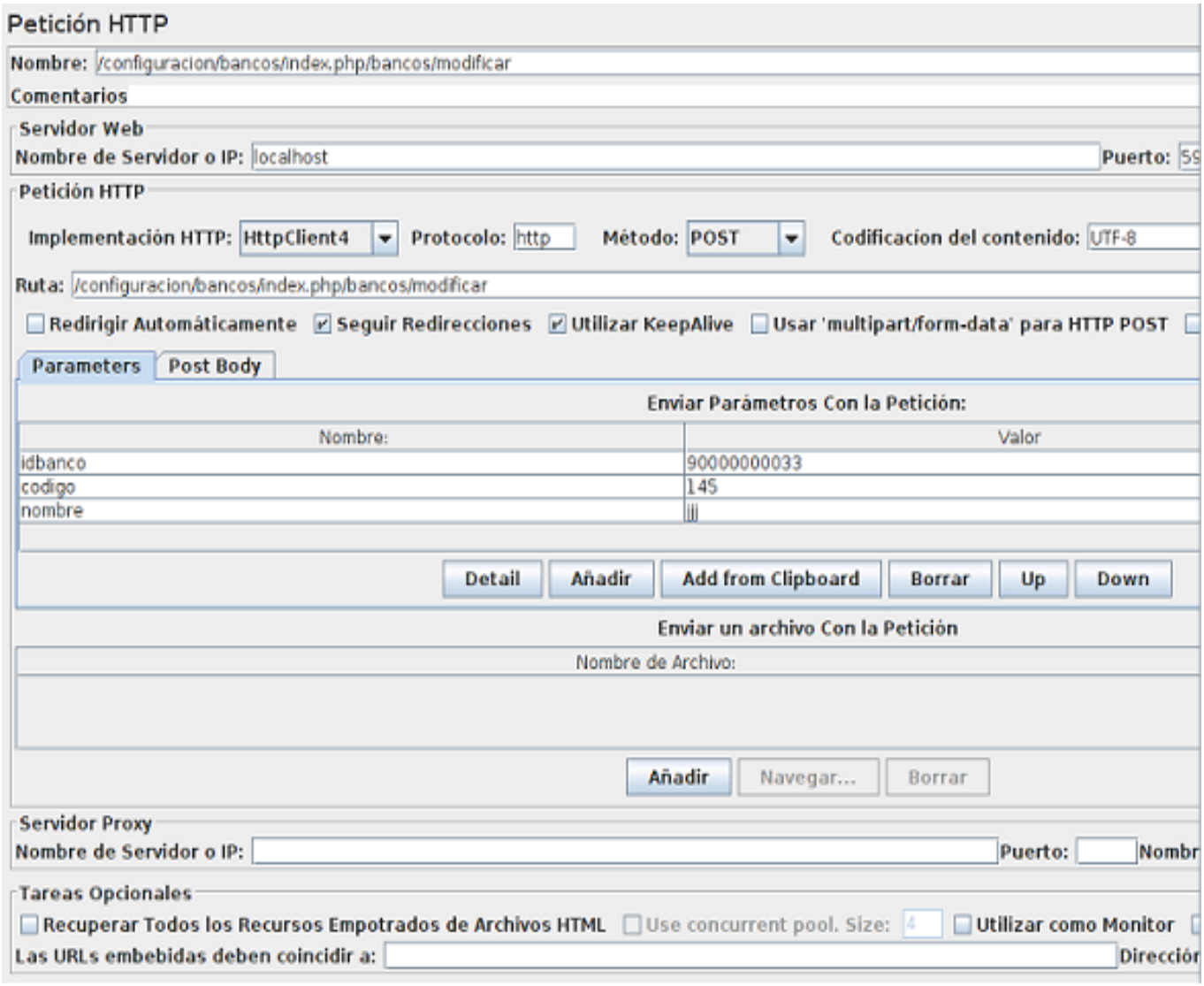


Figura 20. Ejemplo de petición HTTP de la grabación

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 21. 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 22. 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.

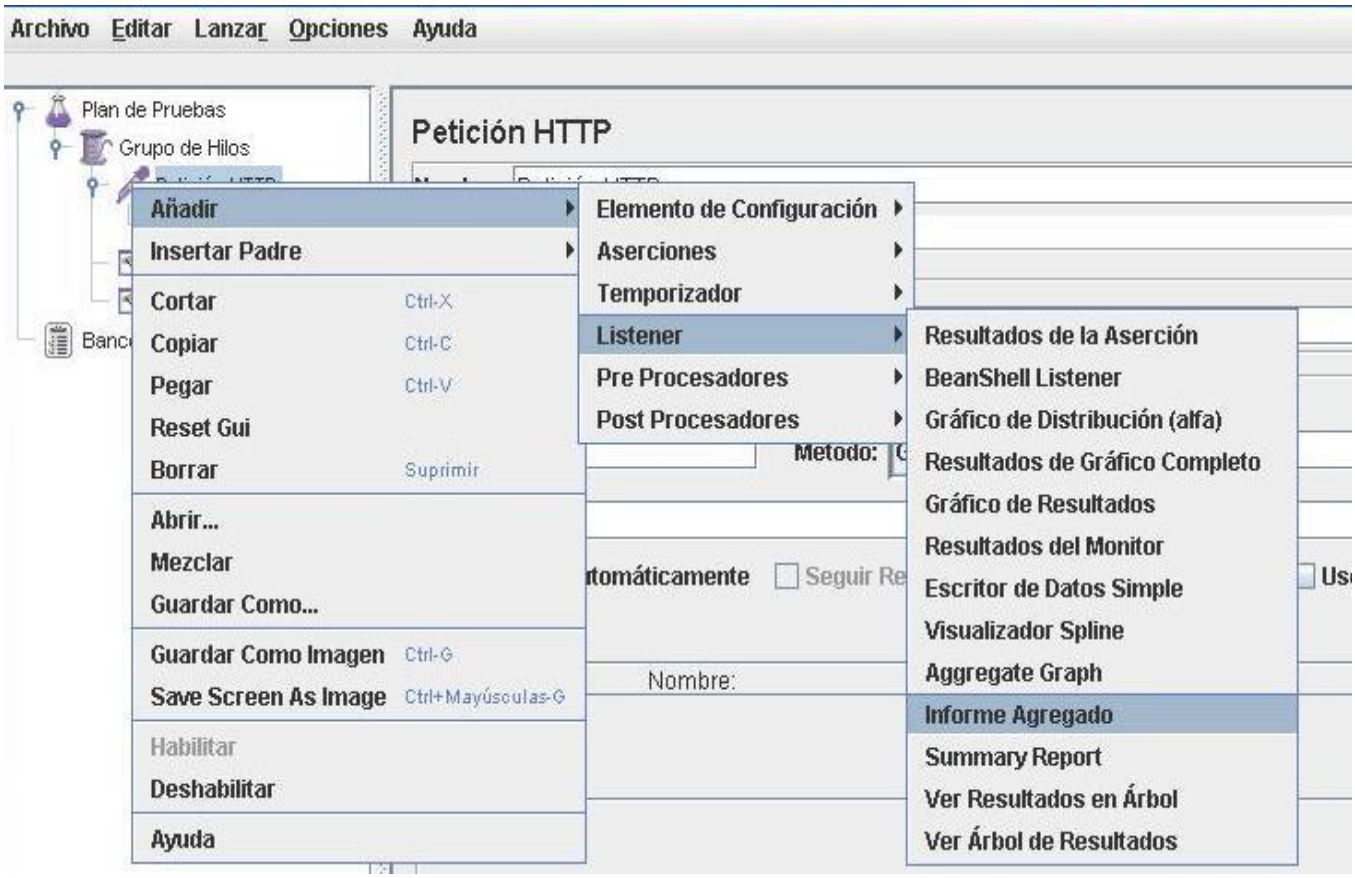


Figura 23. Selección de Informe Agregado

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.

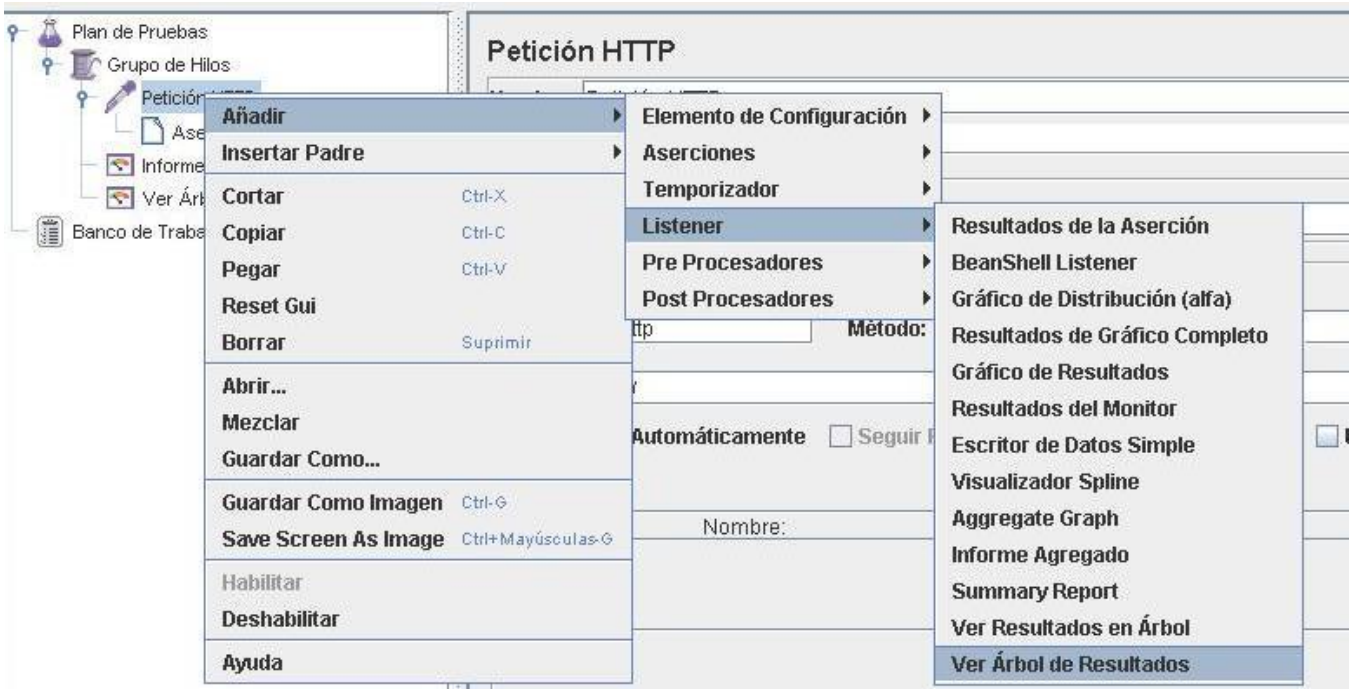


Figura 24. Selección de Ver Árbol de Resultados

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

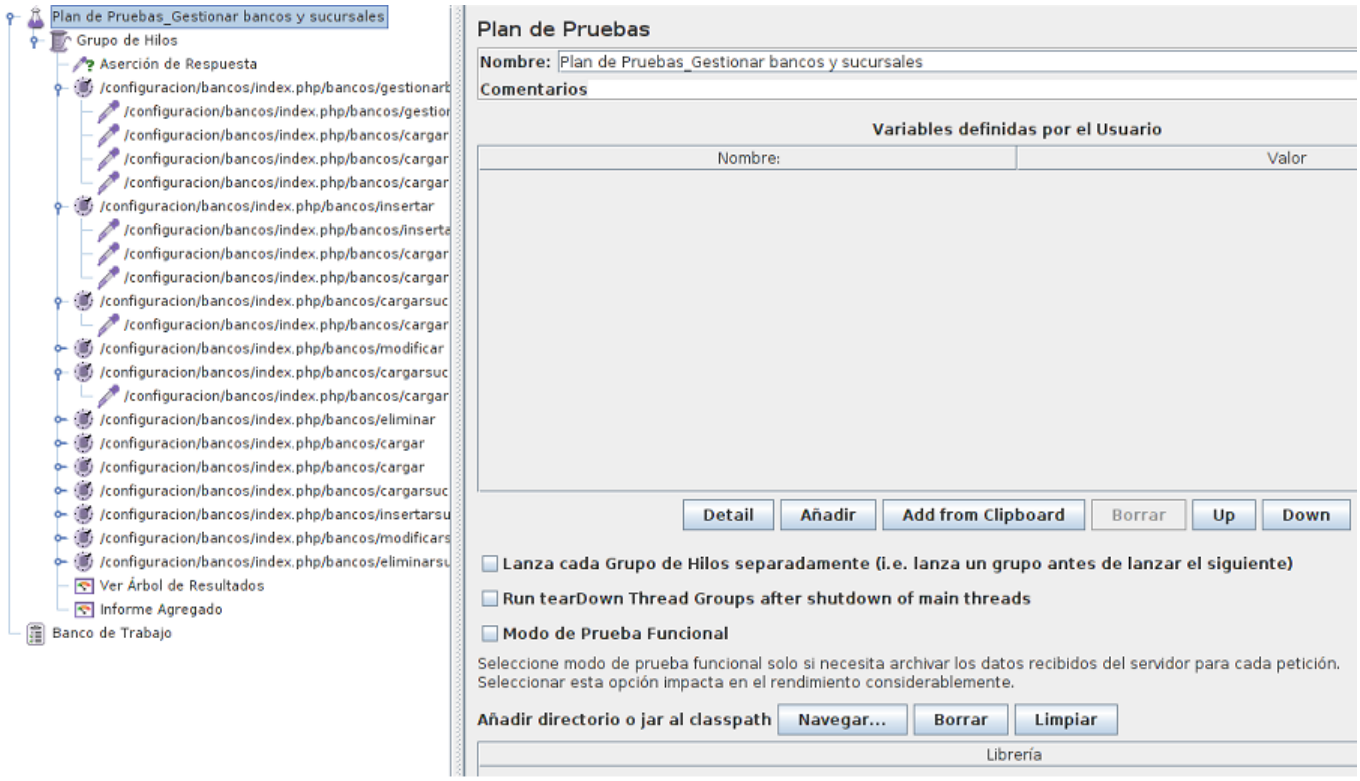
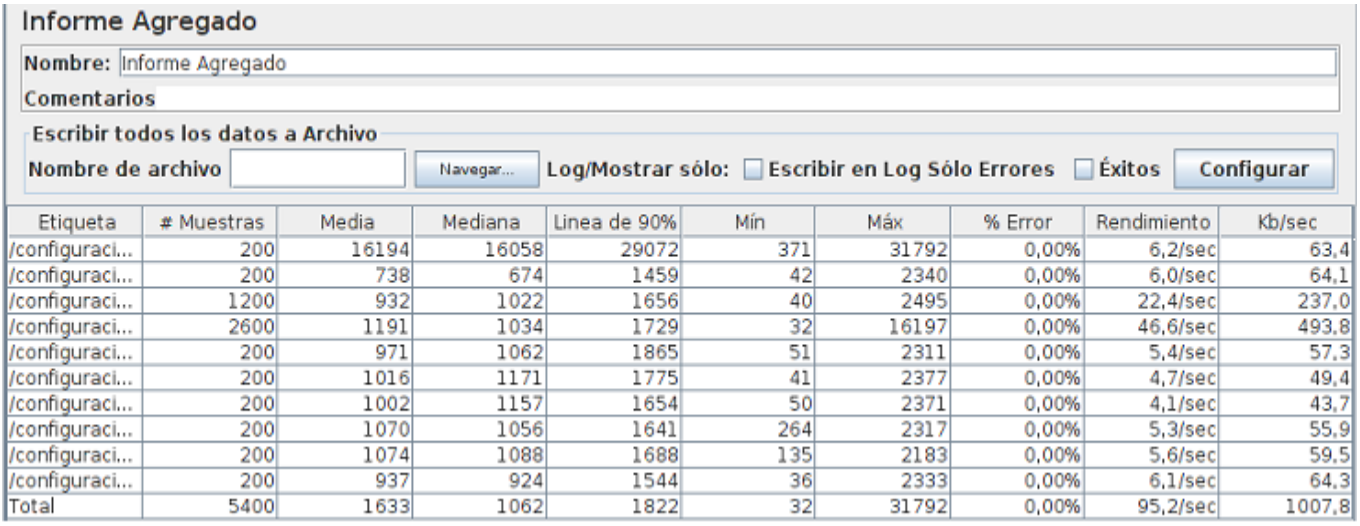


Figura 25. Confección de un Plan de Pruebas

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:



Informe Agregado

Nombre: Informe Agregado

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Mostrar sólo: Escribir en Log Sólo Errores | Éxitos | Configurar

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/configuraci...	200	16194	16058	29072	371	31792	0,00%	6,2/sec	63,4
/configuraci...	200	738	674	1459	42	2340	0,00%	6,0/sec	64,1
/configuraci...	1200	932	1022	1656	40	2495	0,00%	22,4/sec	237,0
/configuraci...	2600	1191	1034	1729	32	16197	0,00%	46,6/sec	493,8
/configuraci...	200	971	1062	1865	51	2311	0,00%	5,4/sec	57,3
/configuraci...	200	1016	1171	1775	41	2377	0,00%	4,7/sec	49,4
/configuraci...	200	1002	1157	1654	50	2371	0,00%	4,1/sec	43,7
/configuraci...	200	1070	1056	1641	264	2317	0,00%	5,3/sec	55,9
/configuraci...	200	1074	1088	1688	135	2183	0,00%	5,6/sec	59,5
/configuraci...	200	937	924	1544	36	2333	0,00%	6,1/sec	64,3
Total	5400	1633	1062	1822	32	31792	0,00%	95,2/sec	1007,8

Figura 26. 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 figura muestra un ejemplo de este componente.

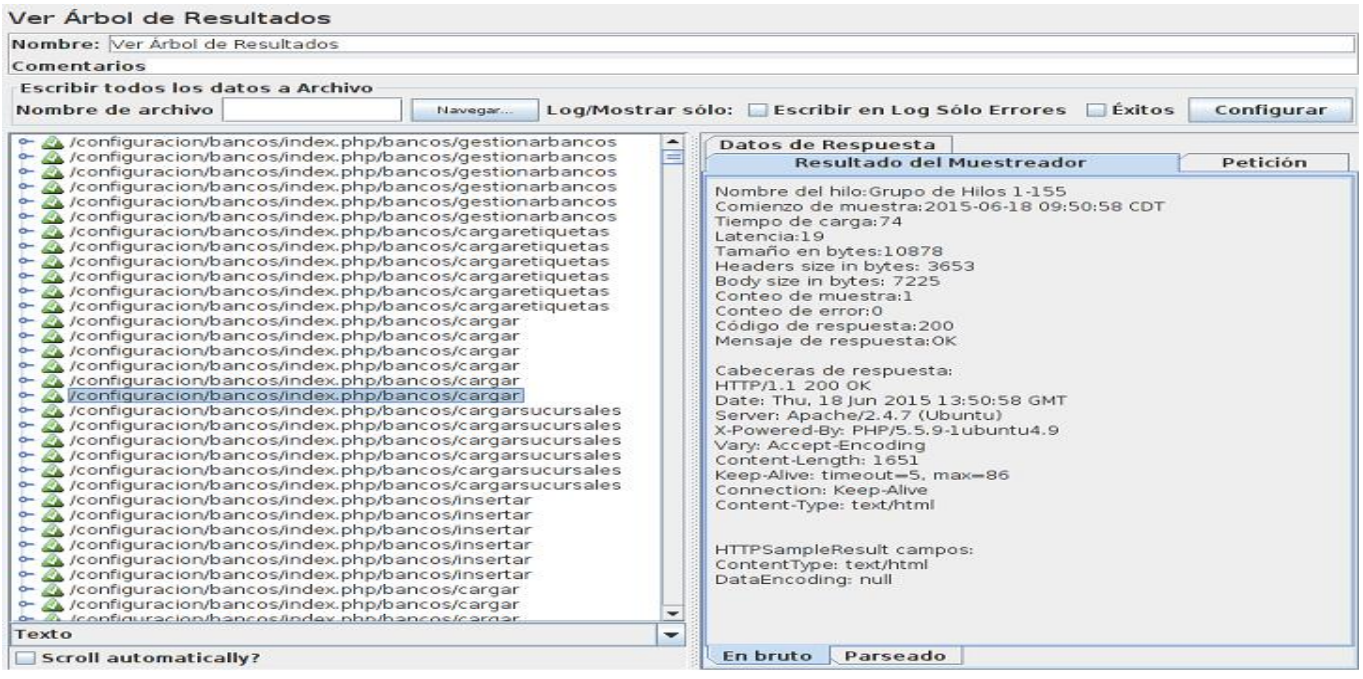


Figura 27. Ver Árbol de Resultados

3.3 Resultado de las pruebas de rendimiento

Las pruebas realizadas consistieron en definir 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 esa cantidad de usuarios conectados concurrentemente al sistema realizando peticiones al servidor, siendo estos los escenarios en los cuales se encontraría el sistema en un momento determinado.

Para calcular el tiempo total en que los usuarios estuvieron accediendo al sistema se utilizó la siguiente fórmula [33]:

- 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 [33]:

- Tiempo promedio por hilo = Tiempo 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. Buscando una mejor visualización de los resultados se procede a evaluar un componente Banco y sucursal haciendo uso de las dos versiones de Ext JS Los valores obtenidos en el Informe Agregado correspondientes a cada versión se muestran a continuación.

Ext JS 2.2

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	5400	1633	1062	1822	32	31792	0,00%	95,2/sec	1007,8

Figura 28. Informe Agregado del Ext JS 2.2

Como puede verse, el tiempo promedio para acceder a una página es 1,633 segundos, realizándose un total de 5400 requerimientos al servidor. El tiempo total utilizado para los 200 hilos se puede calcular con la siguiente fórmula:

Tiempo total = #Muestras * Media = 5400 * 1633 = 8818200 milisegundos.

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

((Tiempo total/1000)/60)/cantidad de hilos = ((8818200/1000)/60)/200 = 0,73485 minutos.

Ext JS 4.2.1

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	13600	273	12	687	1	13162	0,00%	309,6/sec	1495,5

Figura 29. Informe Agregado del Ext JS 4.2.1

Como puede verse, el tiempo promedio para acceder a una página es 0,273 segundos, realizándose un total de 13600 requerimientos al servidor.

Tiempo total = 13600 * 273 = 3712800 milisegundos.

Tiempo promedio total = ((3712800/1000)/60)/200 = 0,3094 minutos.

Tabla 10. Análisis de resultados

	50		100		200	
	Ext JS 2.2	Ext JS 4.2.1	Ext JS 2.2	Ext JS 4.2.1	Ext JS 2.2	Ext JS 4.2.1
Número Muestras	1350	3400	2700	6800	5400	13600
Media	369	85	802	122	1633	273
Mediana	257	7	682	5	1062	12
Línea de 90%	460	215	1131	253	1822	687
Min	38	0	33	0	32	1
Max	4821	2980	7407	5486	31792	13162
Porcentaje de Error	0.0	0.0	0.0	0.0	0.0	0.0
Tiempo total (ms)	498150	289000	2165400	829600	8818200	3712800
Tiempo promedio (min)	0,16605	0,0963	0,3609	0,1383	0,73485	0,3094

De los resultados anteriores se puede concluir que el tiempo total utilizado para la cantidad de hilos definida, cambia notablemente en la versión 4.2.1 de Ext JS respecto a la versión 2.2. Realizando una comparación se tiene como tiempo general de la prueba: 11 481 750 milisegundos para la versión 2.2 y 4 831 400 milisegundos para la 4.2.1. Estos resultados demuestran una reducción de 6 650 350 milisegundos en la ejecución del plan de pruebas en cuestión, lo que representa una reducción del 57.9% de este indicador.

Por su parte, en el tiempo promedio requerido para cada hilo se puede apreciar que hubo una mejora de la versión 4.2.1 respecto a la 2.2. El tiempo promedio general que se empleó para cada hilo fue de 1,2618 min en la versión 2.2 y 0,544 min en la 4.2.1. Estos resultados mostraron una reducción de 0,7178 min para la interacción con el sistema por las cantidades de usuarios simulados al utilizar ambas versiones, esto representa una disminución del 95,7% de este indicador.

3.4 Consumo de recursos

El consumo de recursos también disminuyó con el desarrollo de la migración. Con la ayuda de la herramienta Firefox, se observó que el consumo de recursos en KB del componente Configuración con la versión 4.2.1 de Ext JS es de 391.8 KB menor en comparación con la versión 2.2 que es de 589, 63 KB ver Tabla 1.

Tabla 11. Tiempo de carga de las interfaces y consumo de recursos con Ext JS v4.2.1

Componente	Recursos (KB)	Tiempo (s)
Banco y sucursal	23,40	0,71
Documentos primarios	16,98	0,83
Operaciones	4,11	0,53
Ejercicios	64,65	0,81
Formatos	43,35	0,76
País	12,48	0,63
Clientes y proveedores por entidad	26,58	1,06
Clientes y proveedores	38,81	0,56
Unidad de medida	37,24	0,62
Desglose de moneda	13,50	0,83
Tasa	47,01	0,72
Moneda contable	31,96	0,65
Moneda entidad	31,73	0,71
Total	391,8	18,84

3.5 Resolver las no conformidades

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.

Conclusiones del capítulo

Se realizaron pruebas de carga y estrés haciendo uso de la herramienta JMeter, obteniendo como resultado que el tiempo total y el tiempo promedio de carga de las interfaces para un escenario de 50 hilos disminuyó un 42%, para 100 hilos disminuyó un 61.7% y con 200 hilos disminuyó un 58%, luego de desarrollada la migración de la capa de presentación del componente Configuración en la versión 4.2.1 de Ext JS.

Se demostró que con el desarrollo de la migración de la capa de presentación del componente Configuración del marco de trabajo Sauxe utilizando la arquitectura Modelo Vista Controlador que implementa Ext JS en su versión 4.2.1, se disminuye el tiempo de carga de las interfaces y el consumo de recursos, mediante las pruebas de rendimiento de carga y estrés realizadas al componente Configuración.

CONCLUSIONES GENERALES

En la construcción del marco teórico de la investigación se realizó un análisis de los conceptos asociados al dominio del problema, permitiendo conocer las características principales de la arquitectura Modelo Vista Controlador de Ext JS en su versión 4.2.1. Se identificaron los pilares del proceso de migración permitiendo definir los elementos básicos sobre los cuales se sustenta la ejecución de la migración. Se realizó una descripción de la evolución de cada una de las versiones de Ext JS, teniendo en cuenta los criterios: cantidad de errores, nuevos componentes funcionales agregados y las mejoras en consistencia y memoria, permitiendo realizar una valoración de las versiones de Ext JS; conduciendo a identificar la versión 4.2.1 como la versión más estable y adecuada para dar solución al problema a resolver. Se realizó un análisis de las características y estructura de las tecnologías a utilizar para el desarrollo de la migración.

Se logró desarrollar la migración de Ext JS de la versión 2.2 para una versión superior 4.2.1, donde se ejecutaron las actividades y tareas que se definen en la Guía de migración de la capa de presentación definida en el Departamento de Desarrollo de Componentes.

Se elaboraron y ejecutaron las pruebas de caja negra sobre la interfaces del software permitiendo demostrar que las funciones del software son operativas y que la integridad de la información externa se mantiene.

Se demostró que con el desarrollo de la migración de la capa de presentación del componente Configuración del marco de trabajo Sauxe utilizando la arquitectura Modelo Vista Controlador que implementa Ext JS en su versión 4.2.1, disminuye el tiempo de carga de las interfaces y el consumo de recursos, mediante las pruebas de rendimiento de carga y estrés realizadas al componente Configuración.

RECOMENDACIONES

Integrar el esquema de datos donde se almacena la información relacionada con el componente Configuración con la base de datos central del marco de trabajo Sauxe.

BIBLIOGRAFÍA

1. Minetto, E.L., *Frameworks para Desenvolvimiento em PHP*. 2007, Livraria Martin Fontes: Sao Paulo.
2. Booch, G., James Rumbaugh and Ivar Jacobson, *El Lenguaje Unificado de Modelado*. 1999, Addison Wesley Iberoamericana.
3. Rivas, J.O.G., *Vistas de arquitectura de sistema y de datos de las soluciones Estructura y Composición, Configuración y Multimoneda del proyecto ERP Cuba*. 2010, Universidad de Ciencias Informáticas: Ciudad de la Habana.
4. Nielsen, J. *Response Times: The 3 Important Limits*. 1993.
5. Muñoz Tamayo, F.y.R.A.M., Félix ,y otros, *Herramienta para la migración de datos de Microsoft Access a PostgreSQL.* Universidad de las Ciencias Informáticas, 2009.
6. inteligentes, M.P.S. *Arquitectura MVC con ExtJS 4*. 13 de mayo del 2015]; Available from: http://www.marioperez.com.mx/.
7. Sencha. *API ExtJS 4.2.1*. 2013 [cited 2015 29 de enero]; Available from: http://docs.sencha.com/extjs/4.2.1/.
8. Moreno, L. *Componentes de una interfaz*. [cited 2015 28 de enero]; Available from: http:// www.desarrolloweb.com/articulos/2171.php.
9. *Diccionario de términos informáticos*. 2012.
10. Shea Frederick, C.R., Steve ´Cutter` Blades, *Learning ExtJS*.
11. spin.java, *ExtJs vs Jquery* in *Sencha ExtJS*. 2011.
12. desarrolloweb. 2015; Available from: desarrolloweb.com/manuales/manual-angularjs.html.
13. XETID *Propuesta I+D*. 2014.
14. Framework, D.P.J. *Prototype*. 2015; Available from: http://prototypejs.org/doc/latest/.
15. Yoya. *Javascript Frameworks*. 2011; Available from: https://underc0de.org/foro/programacion-web-247/javascript-frameworks/.
16. Library, D.J.J. *jQuery API*. 2015; Available from: https://api.jquery.com/.
17. Media, S. *Útiles Deslizadores o Sliders de Imagen jQuery Responsive* 2013; Available from: http://eduardoarea.blogspot.com/2013/09/60-utiles-deslizadores-o-sliders-de.html.
18. Mootools. *Mootools Documentation*. 2015; Available from: http://mootools.net.
19. Álvarez, M.A. *Mootools*. 2008; Available from: http://www.desarrolloweb.com/articulos/mootools.html.
20. anieto2k. *ExtJS es el framework Javascript más rápido*. 2010; Available from: http://www.anieto2k.com/2009/03/31/dojo-es-el-framework-javascript-mas-rapido/.
21. Ciolli, C.C.M.E., *Testing de migración de aplicaciones distribuidas a entornos Web*. 2007, Universidad Nacional de la Plata: La Plata.
22. Inoelkis Velázquez Osorio, L.C.E., *Migración de la capa de acceso a datos del marco de trabajo Sauxe*, in *Facultad 3*. 2013, Universidad de las Ciencias Informáticas: Facultad 3.

23. Velázquez, Y.O., *Guía del proceso de migración del Portal Octavitos*, in *Facultad 4*. 2011, Universidad de las Ciencias Informáticas.
24. Baryolo, I.O.G., *Solución informática de autorización en entornos multientidad y multisistema*. 2010, Universidad de las Ciencias Informáticas: Ciudad de la Habana.
25. Apache, T. *The Apache HTTP server Project*. 2013; Available from: http://httpd.apache.org/.
26. Software., P. *PgAdmin. Excelente gestor PostGreSQL*. 2011; Available from: http://www.purosoftware.com/programacion-bases-de-datos/11-pg-admin-3.htm.
27. López, J.M., *Cliente para Subversion para principiantes y expertos*. 2010.
28. Netbeans. 2015; Available from: https://netbeans.org/.
29. Tapia, J.R., *Migración de la arquitectura de presentación del subsistema Contabilidad de CEDRUX*. 2013, Universidad de las Ciencias Informáticas: La Habana, Cuba.
30. Atencio, A.Á., *Panel de administración para el marco de trabajo Sauxe*. 2014, Universidad de las Ciencias Informáticas: Ciudad de la Habana.
31. Batista, C.B. *Migración del framework ExtJS versión 2.2 a la versión 4.0 en el marco de trabajo Sauxe*. 2012.
32. Pressman_6ta_edicion, in *Pressman*. 2015.
33. F. Javier Diaz, C.M.T.B., Anahí S. Rodríguez, Valeria Soria, *Usando Jmeter para pruebas de rendimiento*. 2013, La Plata, Argentina.

GLOSARIO DE TÉRMINOS

Cuentas controladas: se le denominan al grupo de cuentas contables que son características de un subsistema o actividad, únicos con total acceso a manipular sus saldos.

Debe: es el lado izquierdo de la cuenta. En este lado se anotan las entradas o aumentos si la cuenta es de Activo, las cancelaciones o disminuciones de las obligaciones si es de Pasivo y los gastos y pérdidas si es de Resultado.

Documento: documentos utilizados en las entidades para realizar las operaciones.

Ejercicio contable: año económico; período de tiempo comprendido entre dos balances anuales sucesivos.

Ejercicio contable basado en año natural: año económico que coincide con los meses del año actual.

Gestión: son las diferentes funcionalidades que permite realizar un componente, estas son: adicionar, modificar, eliminar, y buscar.

Haber: es el lado derecho de una cuenta. Se anotan en esta sección las salidas o disminuciones si la cuenta es de Activo, la creación o aumentos si es de Pasivo, y los beneficios o ganancias si es de Resultado.

Moneda contable: moneda definida en el nomenclador de monedas que se utiliza para el registro contable de las operaciones, según la legislación vigente.

Moneda original: moneda definida en el nomenclador de monedas en el que se realizan las diferentes transacciones.

Moneda alternativa: moneda definida en el nomenclador de monedas que se utiliza para operaciones funcionales con igual poder de compra que la moneda contable. Es característico del tema de dualidad monetaria.

Moneda contable: moneda definida en el nomenclador de monedas que se utiliza para el registro contable de las operaciones, según la legislación vigente.

Objetos: concepto fundamental sobre el cual gira el negocio del subsistema, por ejemplo, las personas en capital humano, los activos en activo fijo.

Operación: es la información que ampara, justifica o respalda la existencia de un documento, o sea representa los datos primarios, contrapartes de un documento. La contrapartida se expresa a través de conceptos que especifican y detallan con qué fin fue creado el documento. El término contrapartida surge a partir del principio de contabilización basado en las partidas dobles, donde se debe especificar para cada asiento, la(s) cuenta(s) e importes que afecten el crédito y el débito respectivamente, en un asiento contable.

Período contable: se le denomina al intervalo de tiempo en que serán registrados los hechos económicos acaecidos en una entidad. El período contable puede ser un día, una semana, un mes o un año.

Período contable: agrupación de días que se establecen dentro del ejercicio contable, si el ejercicio coincide con el año actual, los períodos son los meses del año.

Párrafo fiscal: clasificador de recursos financieros del Presupuesto del Estado.

Parámetros: elementos necesarios a configurar para el correcto funcionamiento del sistema.

Reglas contables: son reglas compuestas por objetos, documentos y operaciones, las cuales tienen como objetivo establecer la forma de contabilización de cada subsistema.

Tasa de cambio: es el precio en moneda doméstica de una unidad de moneda extranjera o convertible.

ANEXOS

Anexo 1. Casos de Prueba de Carga para el requisito “Gestionar bancos y sucursales”

Escenario	Descripción	Código	Nombre	Respuesta del sistema	Flujo central
EC 1.1 Adicionar banco	Se registra un banco en el sistema.	V 23	V BPA	Satisfactorio	1. El usuario accede a la dirección Sauxe/Configuración/Nomenclad ores generales Gestionar bancos y sucursales. 2. El usuario da clic izquierdo sobre el botón Adicionar. 3. El sistema muestra la interfaz Adicionar banco. 4. El usuario introduce los valores código y nombre. 5. El usuario da clic izquierdo sobre el botón Aplicar si desea adicionar un nuevo banco si no da clic izquierdo sobre el botón Aceptar. 7. El sistema registra el banco.
EC 1.2 Modificar banco	Se modifica un banco registrado previamente en el sistema.	V 24	V BM	Satisfactorio	1. El usuario accede a la dirección Sauxe/Configuración/Nomenclad ores generales/ Gestionar bancos y sucursales. 2. El usuario selecciona un banco de los que se muestran en el sistema. 3. El usuario da clic izquierdo sobre el botón Modificar. 4. El sistema muestra la interfaz

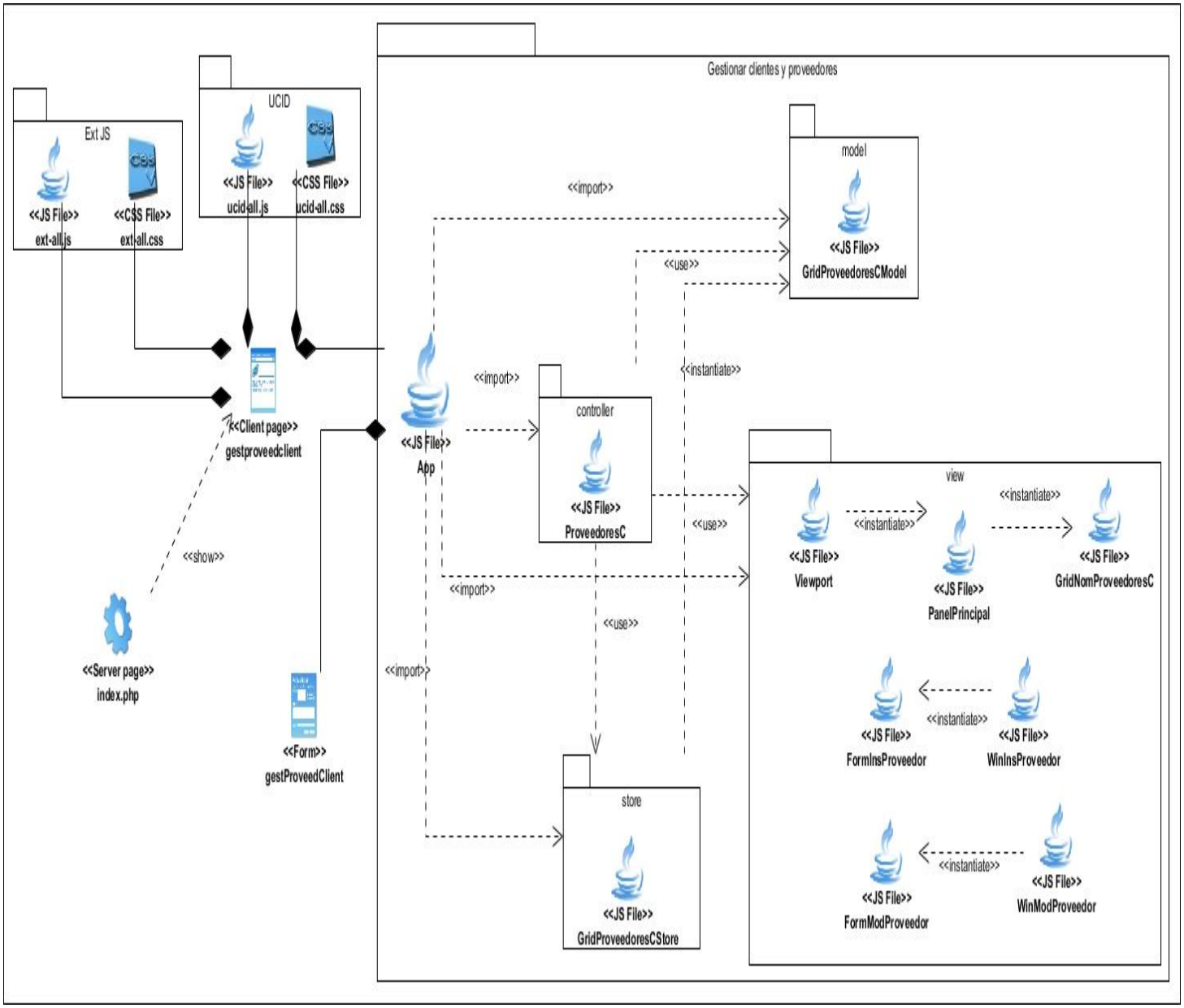
					Modificar banco. 5. El usuario modifica los valores código y nombre. 6. El usuario da clic izquierdo sobre el botón Aceptar. 7. El sistema modifica el banco.
EC 1.3 Eliminar banco	Se elimina un banco registrado previamente en el sistema.			Satisfactorio	1. El usuario accede a la dirección Sauxe/Configuración/Nomenclad ores generales/ Gestionar bancos y sucursales. 2. El usuario selecciona un banco de los que se muestran en el sistema. 3. El usuario da clic izquierdo sobre el botón Eliminar. 4. El sistema elimina el banco.
EC 1.4 Buscar banco	Se muestra un banco registrado previamente en el sistema.	23	BPA	Satisfactorio	1. El usuario accede a la dirección Sauxe/Configuración/Nomenclad ores generales/ Gestionar bancos y sucursales. 2. El usuario introduce un criterio de búsqueda que puede ser por el código y el nombre. 3. El usuario da clic izquierdo sobre el botón Buscar. 4. El sistema muestra el banco o los bancos que se corresponden al criterio de búsqueda introducido.

Anexo 2. Casos de Prueba de Carga para el requisito "Gestionar bancos y sucursales"

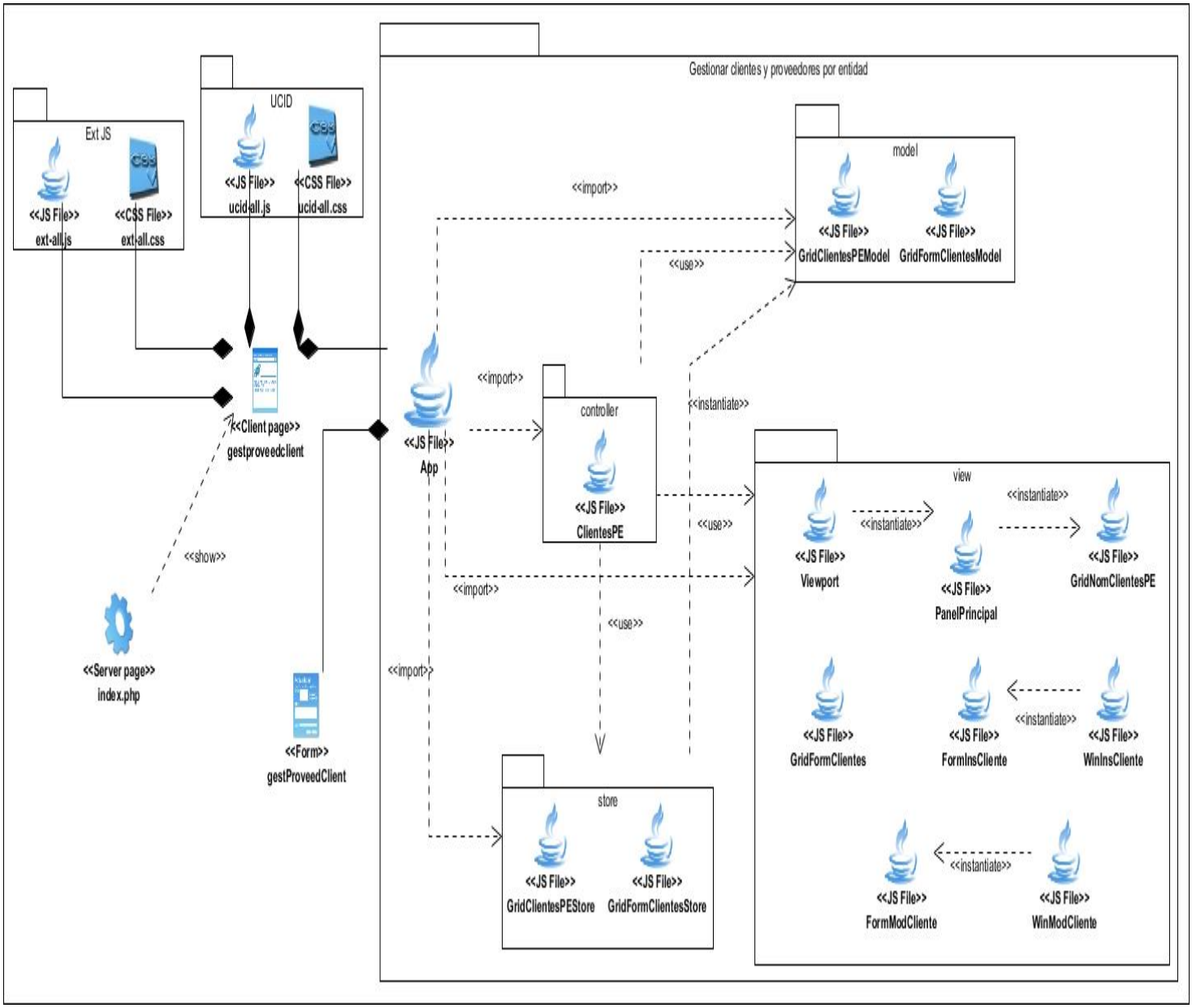
Escenario	Descripción	Nombre	Dirección	Respuesta del sistema	Flujo central
EC 1.1 Adicionar sucursal	Se registra una sucursal en el sistema.	V prueba	V calle Bolívar No.34	Satisfactorio	1. El usuario accede a la dirección Sauxe/Configuración/Nomencladores generales/Gestionar bancos y sucursales. 2. El usuario da clic izquierdo sobre el botón Adicionar. 3. El sistema muestra la interfaz Adicionar sucursal. 4. El usuario introduce los valores Nombre y Dirección. 5. El usuario da clic izquierdo sobre el botón Aplicar si desea adicionar una nueva sucursal si no da clic izquierdo sobre el botón Aceptar. 7. El sistema registra la sucursal.
EC 1.2 Modificar sucursal	Se modifica una sucursal registrada previamente en el sistema.	V prueba modificar	V calle Martí No.56	Satisfactorio	1. El usuario accede a la dirección Sauxe/Configuración/Nomencladores generales Gestionar bancos y sucursales. 2. El usuario selecciona una sucursal de las que se muestran en el sistema. 3. El usuario da clic izquierdo sobre el botón Modificar. 4. El sistema muestra la interfaz Modificar sucursal. 5. El usuario modifica los valores Nombre y Dirección. 6. El usuario da clic izquierdo sobre el botón Aceptar.

					7. El sistema modifica la sucursal.
EC 1.3 Eliminar sucursal	Se elimina una sucursal registrada previamente en el sistema.			Satisfactorio	1. El usuario accede a la dirección Sauxe/Configuración/Nomencladores generales/ Gestionar bancos y sucursales. 2. El usuario selecciona una sucursal de las que se muestran en el sistema. 3. El usuario da clic izquierdo sobre el botón Eliminar. 4. El sistema elimina la sucursal.

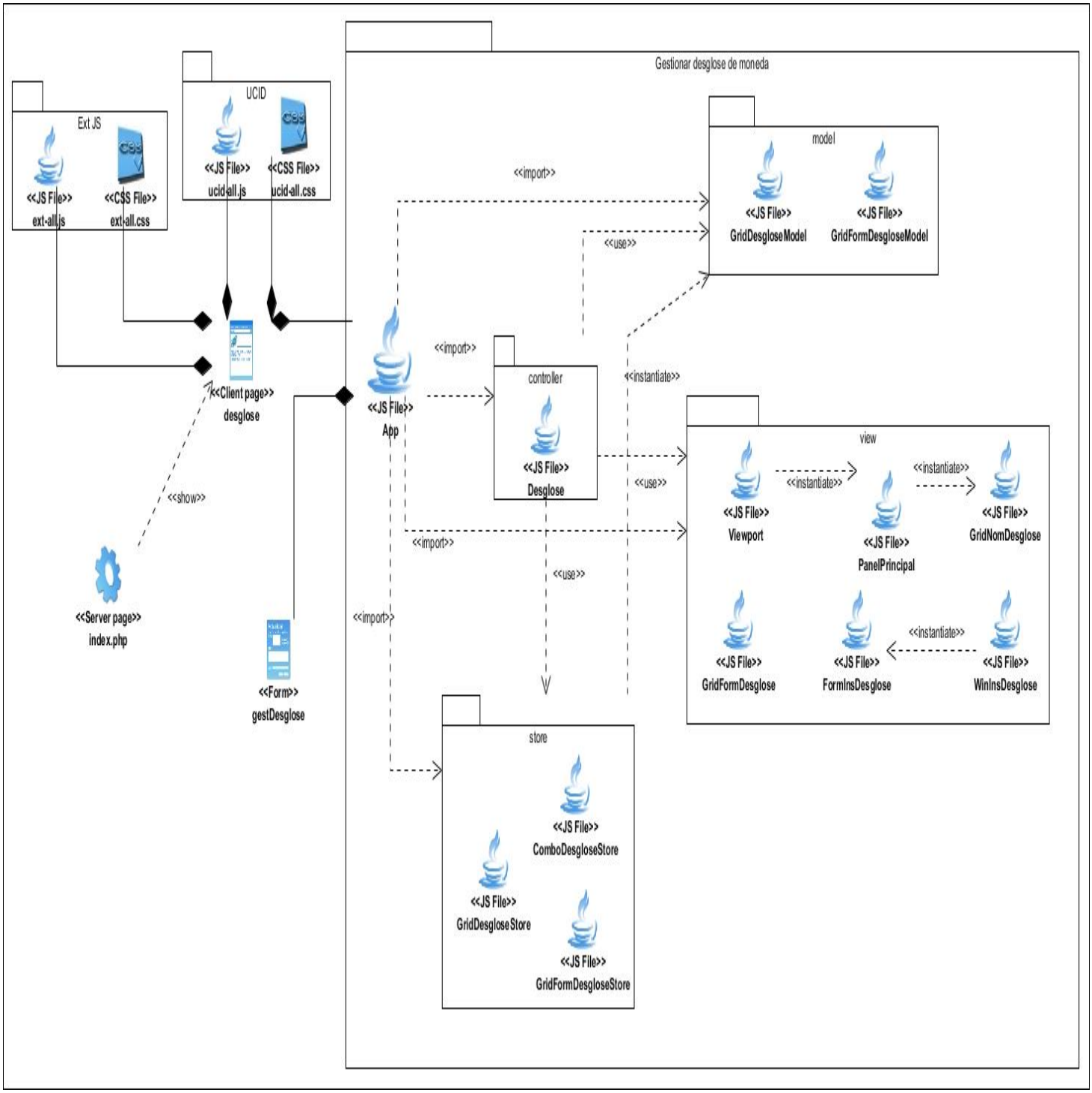
Anexo 3. Diagrama de clase del diseño Cliente y proveedor



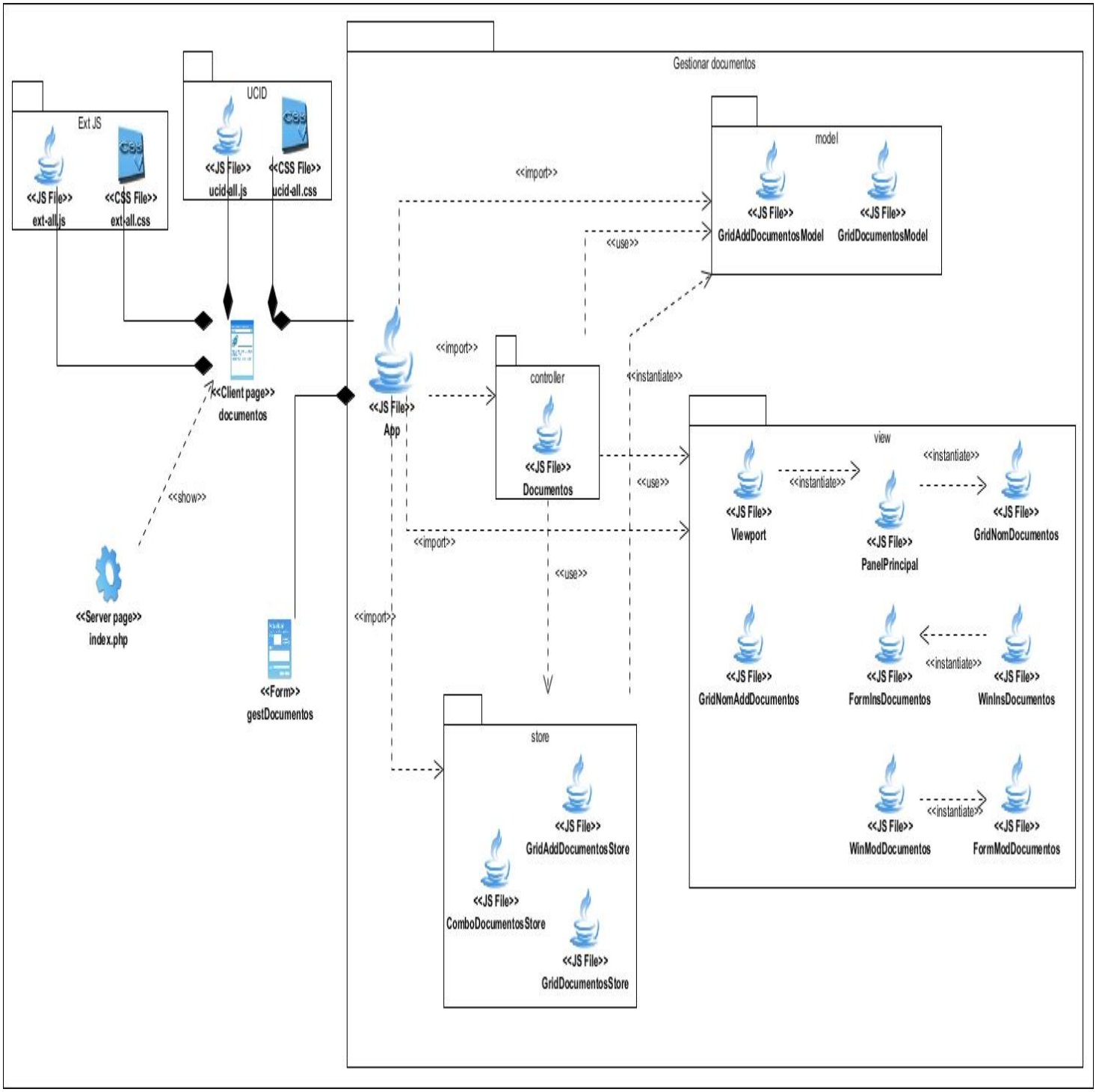
Anexo 4. Diagrama de clase del diseño Cliente y proveedor por entidad



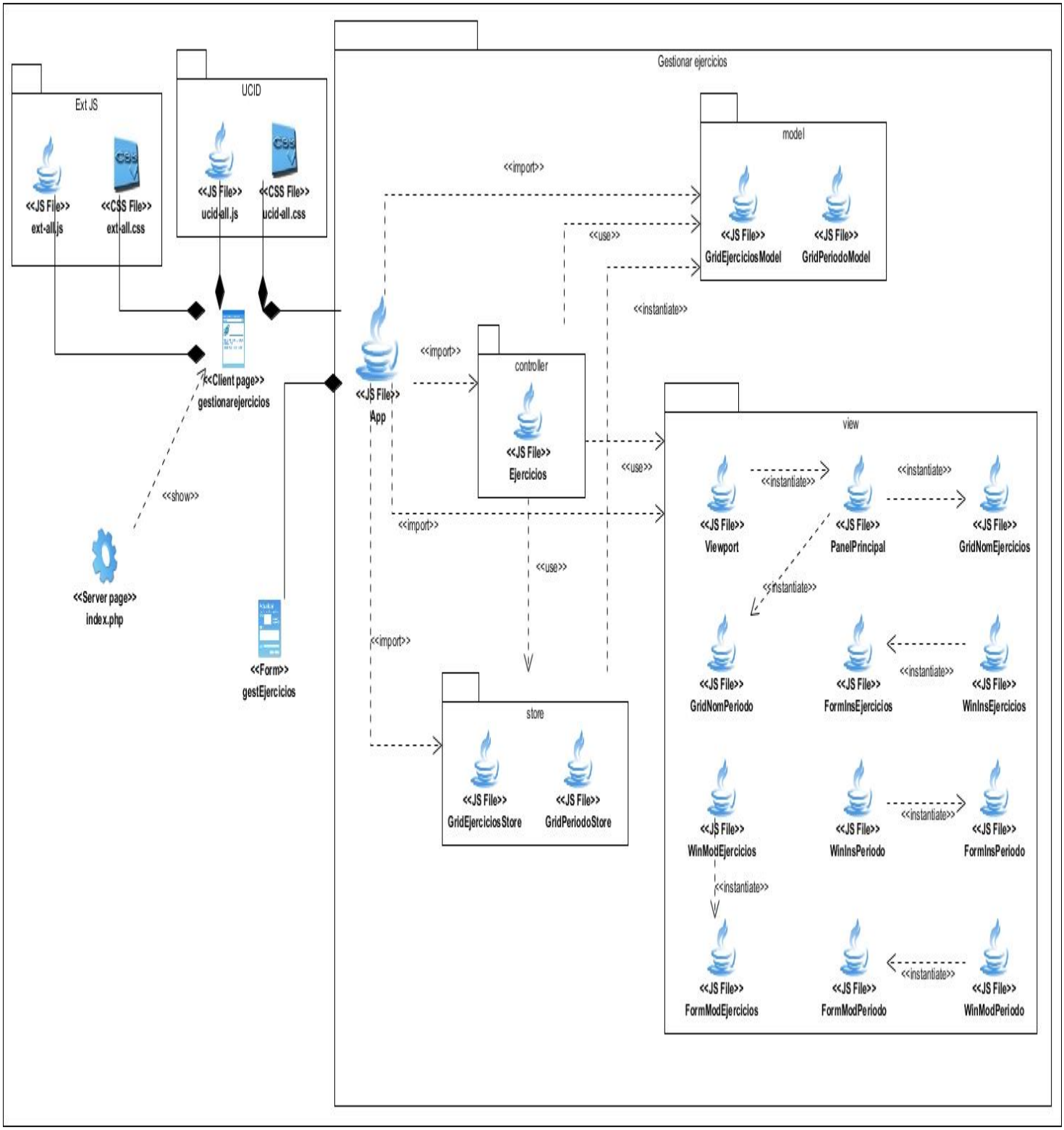
Anexo 5. Diagrama de clase del diseño del componente Desglose de moneda



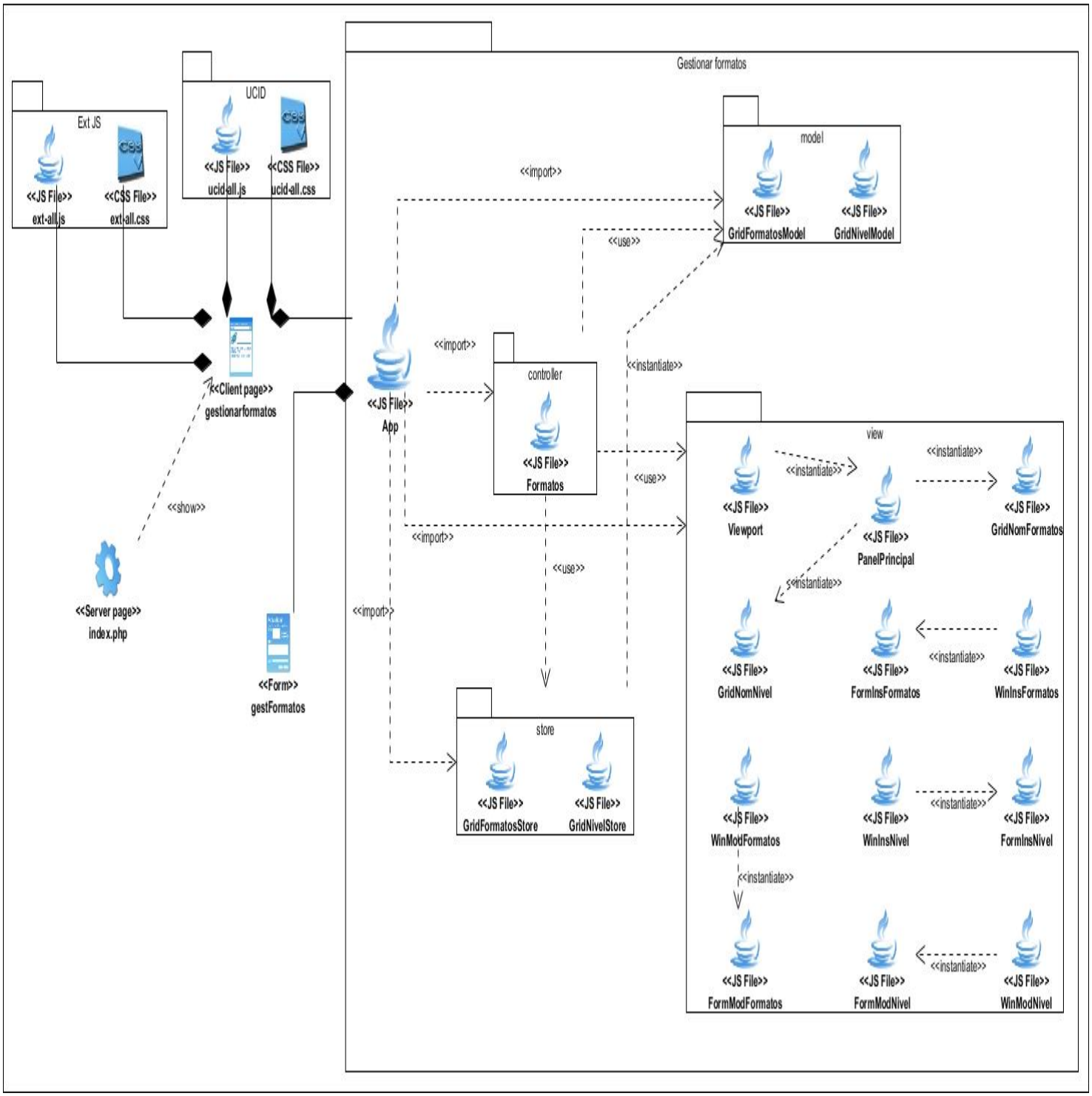
Anexo 6. Diagrama de clase del diseño del componente Documento



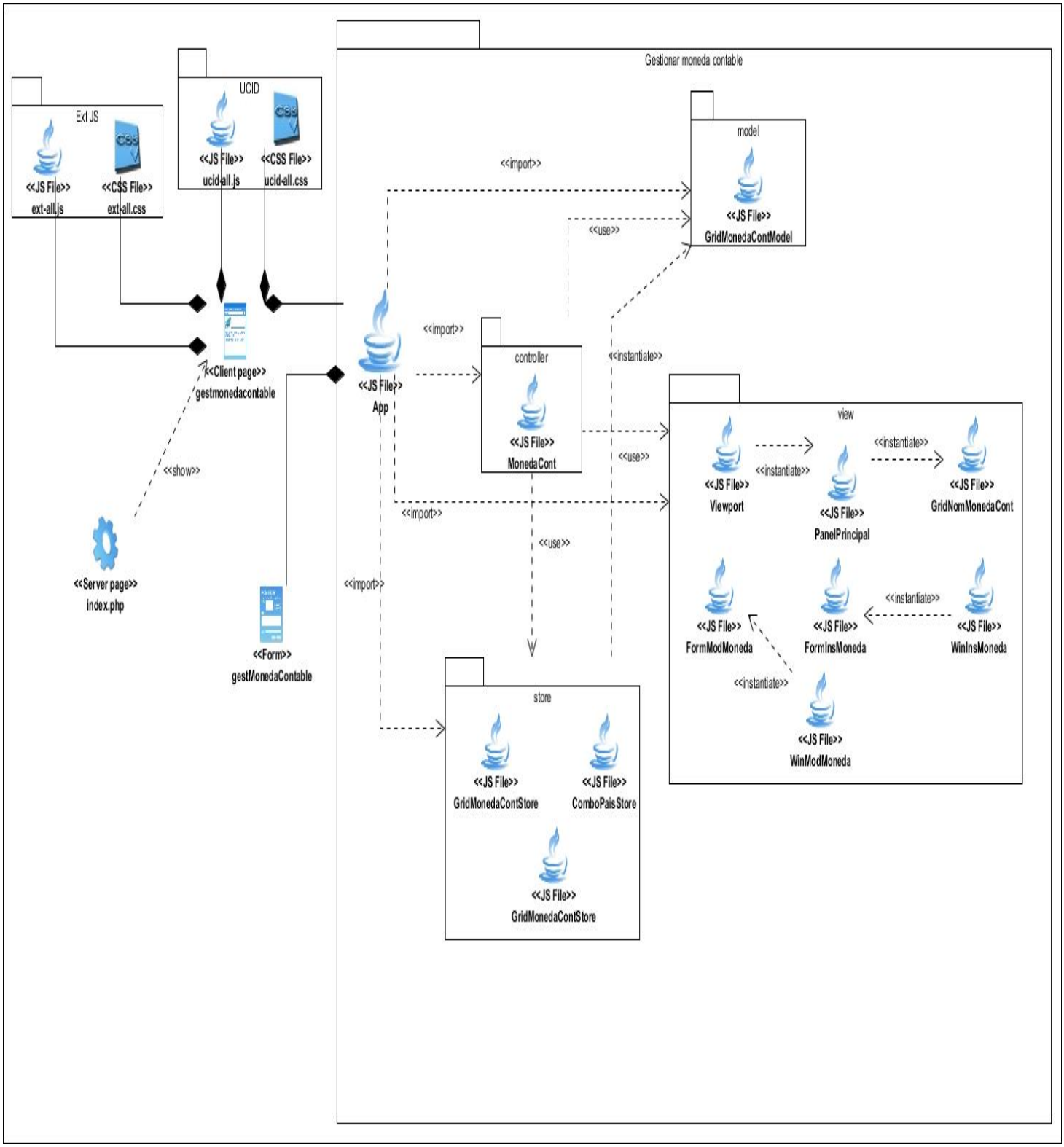
Anexo 7. Diagrama de clase del diseño del componente Ejercicio



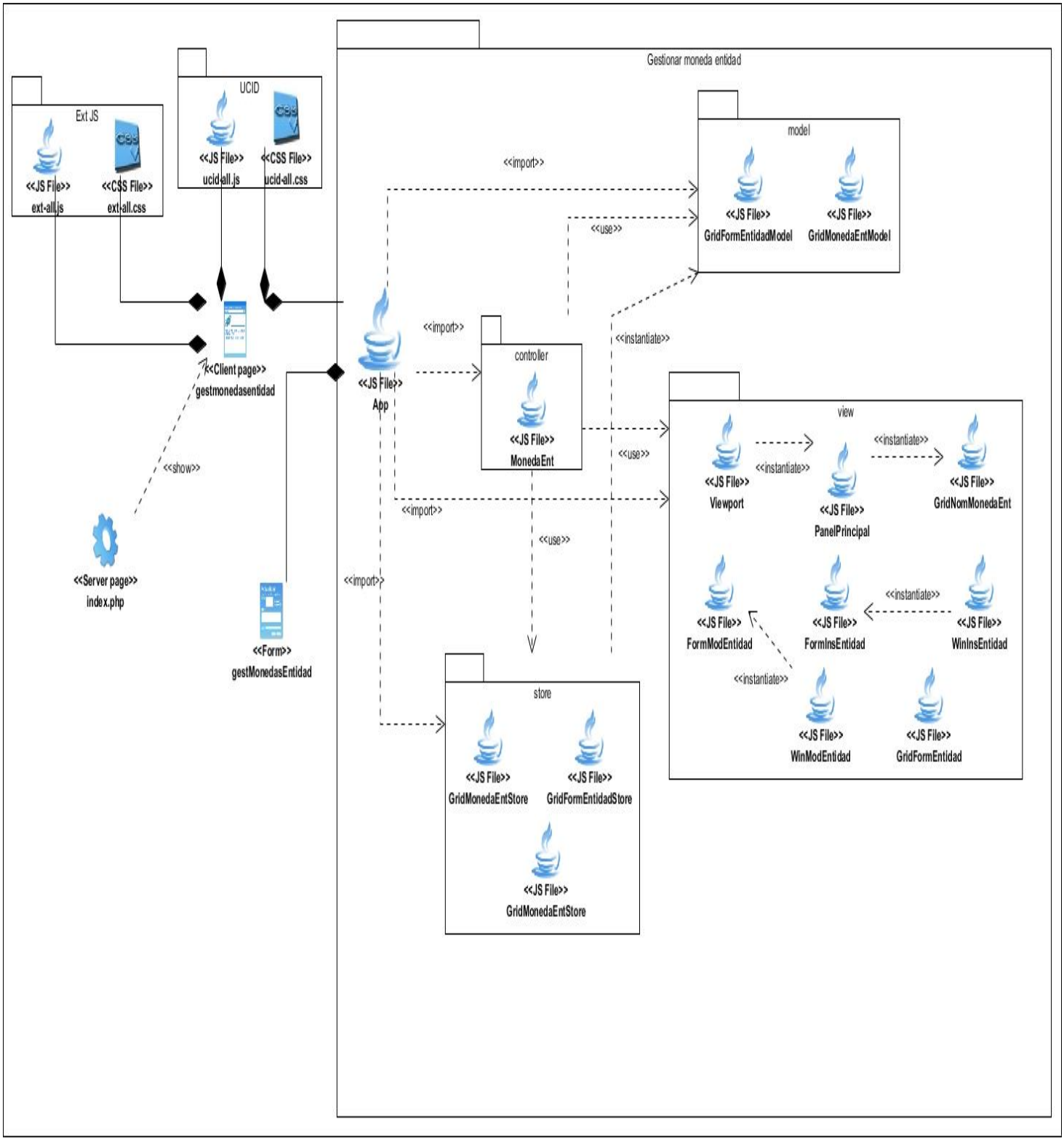
Anexo 8. Diagrama de clase del diseño del componente Formato



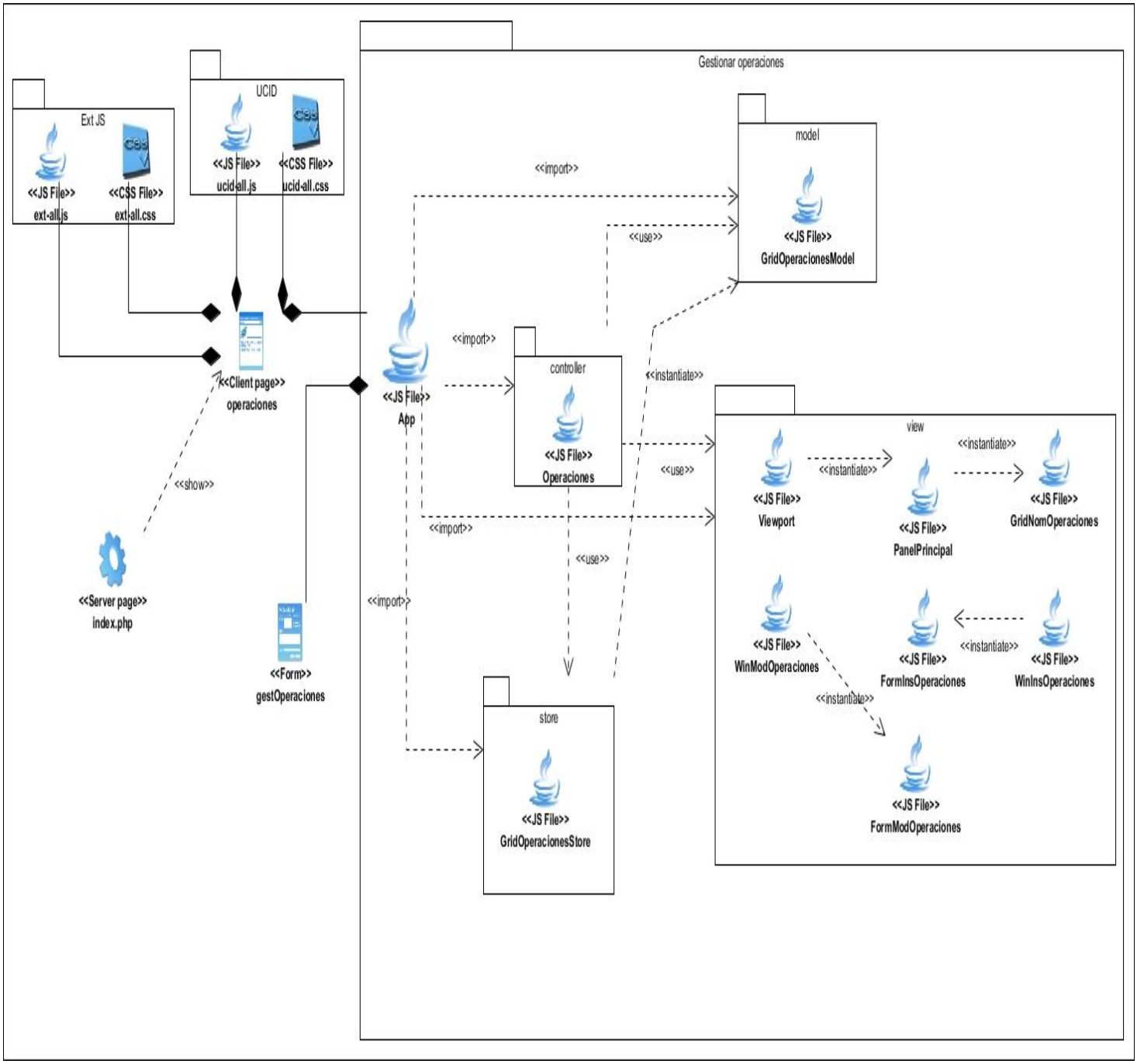
Anexo 9. Diagrama de clase del diseño del componente Moneda contable



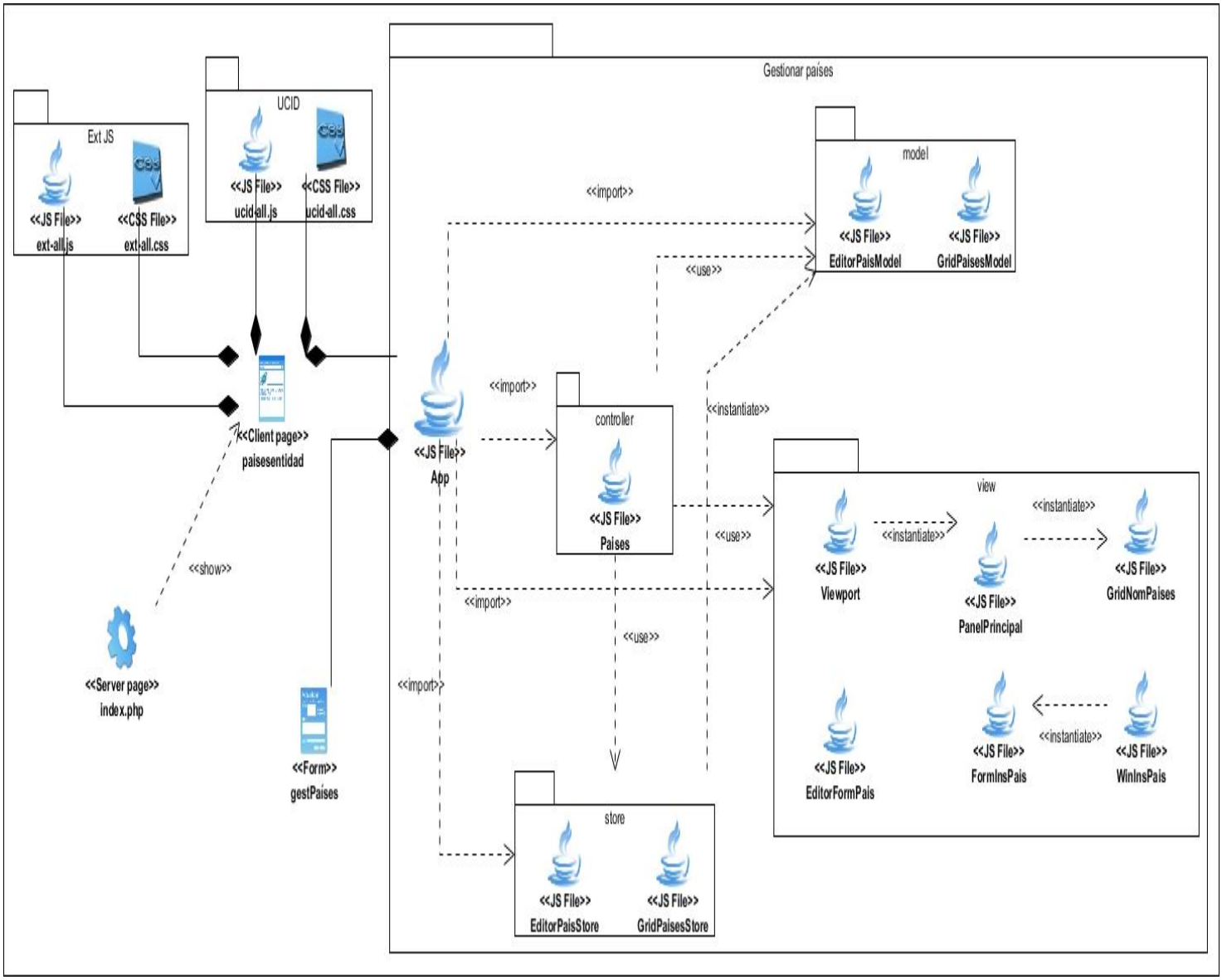
Anexo 10. Diagrama de clase del diseño del componente Moneda entidad



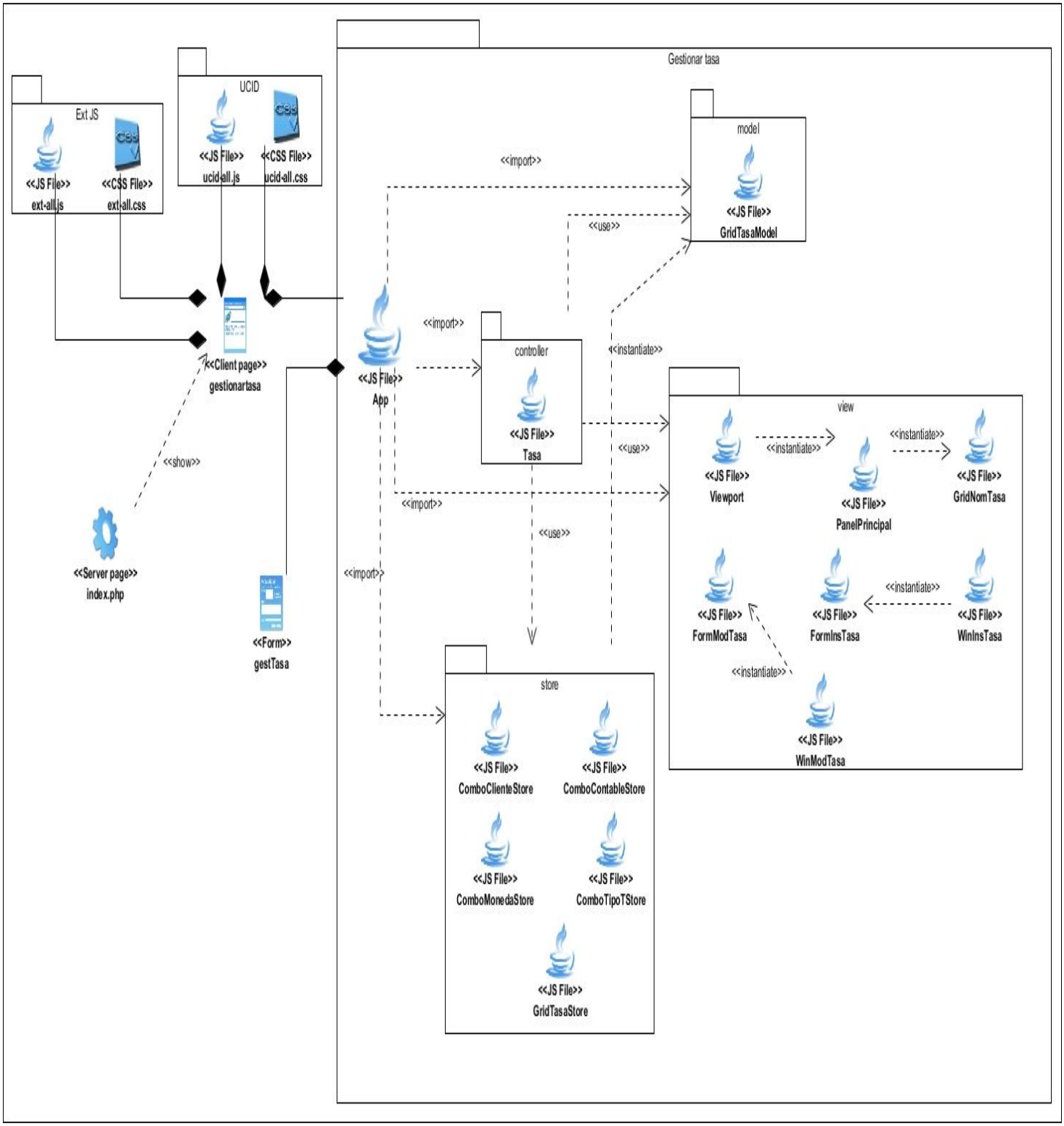
Anexo 11. Diagrama de clase del diseño del componente Operación



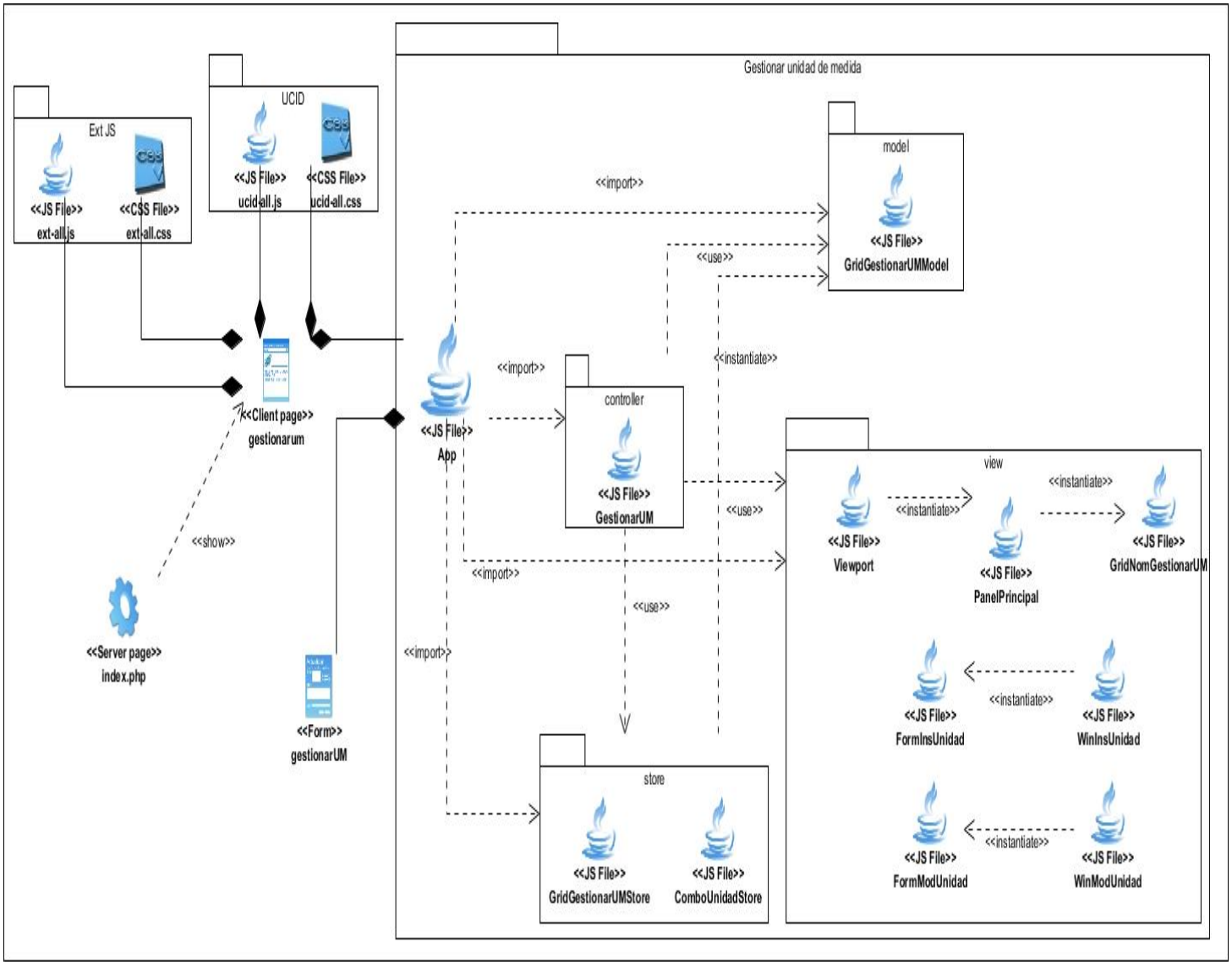
Anexo 32. Diagrama de clase del diseño del componente País



Anexo 43. Diagrama de clase del diseño del componente Tasa



Anexo 54. Diagrama de clase del diseño del componente Unidad de media



Anexo 65. Acta de Aceptación del Departamento Desarrollo de Componentes

:ENTRO DE INFORMATIZACIÓN DE ENTIDADES
)EPARTAMENTO DE DESARROLLO DE COMPONENTES

UCI
Universidad de las Ciencias Informáticas

22 de junio de 2015

A quien pueda interesar

Por este medio se hace constar que la solución ***Migración de la capa de presentación del componente de Configuración del marco de trabajo Sauxe utilizando la arquitectura MVC que implementa ExtJS en su versión 4.2.1*** del autor ***Luis Orlando Proenza Vázquez*** fue sometida a una revisión técnica en la cual se detectaron 15 no conformidades que fueron resueltas quedando esta solución estable y lista para su posterior uso.

Como parte del desarrollo de la solución se elaboraron y entregaron los siguientes artefactos:

1. Diseños de casos de prueba (15).
2. Código fuente (https://celge-repo.uci.cu/sauxe/Sauxe/Raiz/Branches/Sauxe_v2.3_Tools/)

Para que así conste firman a continuación los miembros del equipo que realizó la revisión, el autor y el tutor del trabajo.

Dado a los 22 días del mes de junio de 2015.

Nombre y apellidos	Firma
Revisores: Ing. Katia Saria Preval	
Autor (es): Luis Orlando Proenza Vázquez	
Tutor: Ing. Claudia Bravo Rabista	

J' Departamento de Desarrollo de Componentes
René R. Bauta Camejo