Universidad de las Ciencias Informáticas
Facultad 3

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

Título: “Migración de la capa de acceso a datos de los componentes Flujo de Trabajo y Configuración del Sistema de Planificación de Actividades (SIPAC)”

Autor: Abel Alexis Matoses Picayo

Tutor(es):
Ing. Rodolfo Rodríguez Molinet
Ing. Dionny Cardoso Carmona

Ciudad de La Habana, Junio 2015
“Año 57 de la Revolución”

"Una tesis es como una partida de ajedrez, tiene cierto número de movimientos, pero desde el principio hay que estar capacitado para predecir los movimientos a efectuar con vistas a dar jaque mate al adversario"

Umberto Eco

DECLARACIÓN DE AUTORÍA

Declaro a Abel Alexis Matoses Picayo único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____

Abel Alexis Matoses Picayo

_____ _____

Ing. Rodolfo Rodríguez Molinet Ing. Dionny Cardoso Carmona

AGRADECIMIENTOS

…Gracias a la Universidad de las Ciencias Informáticas por permitir graduarme en ella y brindarme todos sus beneficios, además de conocimientos.

…Gracias a mis profesores que de una forma u otra influyeron en mí para lograr esta inalcanzable meta que veía al principio de la carrera.

…Gracias a mis tutores por estar ahí a la hora CERO cuando menos los esperaba.

…Gracias a mis amigos, sin ustedes no concibo mi vida.

…Gracias a mi novia, por estar ahí desde el principio y apoyarme en todo, por motivarme a ser mejor cada día, por ser TAN COMPRENSIBLE.

…Gracias a mis compañeros por nutrirme con su ejemplo y brindarme siempre su ayuda.

…Gracias a la familia tan linda que tengo y que siempre han confiado en mí en TODO MOMENTO.

…Gracias a mis hermanos por permitirme ser su ejemplo.

…Gracias a todas las personas que han pasado por mi vida dejando siempre algo positivo.

DEDICATORIA

...Dedico este gran resultado a mi razón de ser, MIS PADRES, sin ellos no hubiese logrado este resultado…

...Dedico este trabajo a mis hermanos que aunque no lo crean me han ayudado y dado su ejemplo…

…Dedico esta investigación a mi novia por saber el trabajo que pasamos juntos para lograr esta satisfacción...

…Dedico de MANERA ESPECIAL al que me motivo a ser un profesional y seguir su ejemplo y el día 13 de junio del 2015 hizo un año que lo perdí…A TI CARLOS ENRIQUE BROUGTHON OQUENDO VA DEDICADO ESTE RESULTADO.

RESUMEN

En la Universidad de las Ciencias Informáticas se desarrolló el Sistema de Planificación de Actividades, el mismo está integrado por varios componentes, dentro de los cuales se encuentran Flujo de Trabajo y Configuración. Estos componentes están sustentados por la integración de varios estilos arquitectónicos, el estilo en capas y el Modelo Vista Controlador. Las capas definidas son: Capa de Presentación, Capa de Servicios, Capa de Control o de Negocio y Capa de Acceso a Datos. La Capa de Acceso a Datos actualmente está compuesta por el Mapeador de Objeto-Relacional Doctrine en su versión 1.2.2, pero su uso presenta algunos problemas como la carencia de soporte, deficiencias en el manejo de los datos y problemas de rendimiento. Por ello surge la necesidad de migrar la Capa de Acceso a Datos de los componentes Flujo de Trabajo y Configuración para alcanzar un mejor rendimiento, un mejor manejo de los datos y solucionar los problemas existentes en la primera versión del Mapeador de Objeto-Relacional Doctrine. En este trabajo se tiene como objetivo describir todas las disciplinas por las que se transitó para la migración de los componentes Flujo de Trabajo y Configuración. Además se realizó un estudio de otros sistemas, concluyendo que era necesario migrar a la versión 2.0.6 del Mapeador de Objeto-Relacional Doctrine.

Palabras clave: Capa de Acceso a Datos, Doctrine, Migración, Rendimiento.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La sociedad actual se encuentra en un proceso de constante cambio y el país enfrenta el reto de informatizarla utilizando las nuevas Tecnologías de la Información y las Comunicaciones (TIC). El desarrollo de software ha provocado el uso de nuevas herramientas que faciliten el trabajo de los desarrolladores, destacándose la utilización de Marcos de Trabajo (MT). Los MT son un conjunto de prácticas y criterios que pretenden mejorar la rapidez, productividad y profesionalidad en cuanto a la elaboración de sistemas complejos. Además tienden a promover una estructura estándar para las aplicaciones, así como el uso de buenas prácticas de la programación [1].

La Universidad de las Ciencias Informáticas (UCI) cuenta con diversos centros de producción que utilizan MT, que apoyan y agilizan el proceso de desarrollo de software, uno de ellos es el Centro de Informatización de Entidades (CEIGE). En este centro, el Departamento de Desarrollo de Componentes desarrolló el MT Sauxe sobre el cual el Departamento de Aplicaciones de Gestión Empresarial desarrolló el Sistema de Planificación de Actividades (SIPAC). SIPAC está basado en la Instrucción no.1 del Presidente de los Consejos de Estado y de Ministros para la Planificación de los objetivos y actividades en los órganos, Organismos de la Administración Central de Estado, entidades nacionales y Administraciones locales del Poder Popular[1]. Dicho sistema está destinado a facilitar la gestión de las actividades a todos los niveles organizacionales, permite interrelacionar objetivos de trabajo y actividades en tiempo real; garantizando el seguimiento del desarrollo y cumplimiento de los objetivos y tareas principales en las entidades.

El MT Sauxe usa el modelo de desarrollo basado en componentes, integra varios estilos arquitectónicos, entre ellos el Modelo Vista Controlador (MVC) y en capas. Una de las capas definidas es la Capa de Acceso a Datos (CAD), la cual permite gestionar los datos utilizados en los procesos de negocio y abstraerse de la forma en que estos persisten o son obtenidos. La CAD maneja gran cantidad de información distribuida, por lo que necesita el manejo de objetos y la persistencia de los datos (capacidad que tiene un objeto de perdurar en el tiempo) [2]. En la actualidad, una de las soluciones más completas para tratar el tema de la persistencia de los datos, es el uso de un Mapeador de Objeto-Relacional (ORM,

[1] Instrucción No.1: Tiene como objetivo establecer el procedimiento para llevar a cabo el proceso de planificación del Gobierno, que permita dar cumplimiento a los acuerdos y resoluciones aprobadas en el VI Congreso del Partido Comunista de Cuba, las decisiones de la Asamblea Nacional del Poder Popular, el Consejo de Ministros y la actualización de los planes de la economía.

Object Relational Mapper) que permite guardar objetos en base de datos relacionales de manera lógica. Estos se encargan de crear una capa de abstracción entre la aplicación y la base de datos, separando la lógica del negocio al mapear los objetos de la aplicación a datos que persisten en una base de datos [3].
El SIPAC cuenta con varios componentes, entre ellos se encuentran: Flujo de Trabajo y Configuración; este último contiene las funcionalidades para el trabajo con las configuraciones de grupos de usuarios por rol y permisos; El componente Flujo de Trabajo define los estados y transiciones de la información, a los que serán asociados los tipos de documentos o elementos de la planificación (planes, objetivos y actividades).

Actualmente los componentes Flujo de Trabajo (FT) y Configuración (C), para garantizar la persistencia de los datos utilizan la versión 1.2.2 del ORM Doctrine*,* el cual conforma la CAD garantizando la comunicación con la base de datos. El uso de esta versión provoca que:

- Al ejecutar consultas que necesiten manejar gran cantidad de información o hacer referencia a varias tablas para obtener información, se afecte el rendimiento, por lo que cada vez que se realiza una consulta al sistema gestor de base de datos, la aplicación tendrá que convertir la consulta del lenguaje propio del ORM al lenguaje del proveedor usado, enviarla, leer los registros, limpiarlos y convertirlos a objetos [4].

- La aplicación tenga un rendimiento reducido por la forma en que se maneja la herencia entre clases, donde se debe hacer referencia a numerosas tablas para obtener los datos [5].

- No se corrijan vulnerabilidades en el código que puedan atentar contra el rendimiento debido a que en junio del 2011 se dejó de dar soporte a esta versión de ORM [5].

A partir de la problemática antes descrita se identificó como **problema a resolver**: ¿Cómo mejorar el rendimiento de los componentes FT y C del Sistema de Planificación de Actividades SIPAC? Definiéndose como **objeto de estudio:** La migración en el proceso de desarrollo de software, delimitado en el **campo de acción**: La migración de la capa de acceso a datos del Sistema de Planificación de Actividades. Para llevar a cabo esta investigación se planteó como **objetivo general**: Migrar los componentes FT y C del Sistema de Planificación de Actividades SIPAC a la versión 2.0.6 de Doctrine para mejorar el rendimiento de los mismos.

Para dar cumplimiento al objetivo general se definen como **objetivos específicos**:

- Construir el Marco Teórico de la investigación relacionada con los ORM existentes para identificar buenas prácticas y posibles puntos de reutilización.
- Realizar la migración de la capa de acceso a datos de los componentes FT y C del Sistema de Planificación de Actividades SIPAC a Doctrine 2.0.6 para aumentar su rendimiento.
- Validar mediante el uso de la herramienta JMeter el aumento del rendimiento de la capa de acceso a datos de los componentes FT y C del Sistema de Planificación de Actividades.

La investigación parte de la siguiente **idea a defender:** Si se migran los componentes FT y C del Sistema de Planificación de Actividades SIPAC haciendo uso de la versión 2.0.6 de Doctrine se posibilitará mejorar el rendimiento del mismo.

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

- Estudio del estado del arte de los ORM más utilizados en el mundo.
- Construcción del marco teórico referencial como resultado de la consulta de bibliografía nacional e internacional relacionada al proceso Migración de la CAD.
- Descripción de las técnicas, tecnologías, herramientas y los procedimientos de migración partiendo de buenas prácticas y patrones.
- Migrar la Capa de Acceso a Datos.
- Validación de dicha migración.

En el desarrollo de la investigación se han utilizado los siguientes **métodos científicos:**

Métodos teóricos:

Los métodos teóricos permiten estudiar las características del objeto de investigación que no son observables directamente [6]. De ellos, se utilizan en este trabajo los siguientes:

- **Histórico-lógico**: Para realizar el estudio del estado del arte relacionado con el proceso de migración y los elementos relacionados con los ORM.
- **Inductivo-deductivo**: Al poner en práctica este método nos permite estudiar algunos de los ORM más utilizados actualmente, con el fin de definir sus características particulares y beneficios que brinda su uso.

Métodos empíricos:

Los métodos empíricos describen y explican las características del objeto y representan un nivel de la investigación cuyo contenido procede de la experiencia [6]. De ellos se utilizan en esta investigación los siguientes:

- **Observación**: Permite centrar la atención de la situación existente en el SIPAC, en los componentes FT y C, y en los inconvenientes de la utilización de la versión 1.2.2 de Doctrine.
- **Medición**: Posibilita obtener información acerca del rendimiento de la aplicación del SIPAC al realizarse sobre este, diferentes llamadas a funciones entre varios componentes, y hacer referencia a varias tablas para obtener datos.

El trabajo de diploma se encuentra estructurado en tres capítulos. A continuación se describe el objetivo de cada uno de ellos:

En el **Capítulo 1:** Fundamentación teórica, se realiza un estudio del estado del arte de las principales tendencias mundiales de la utilización de los ORM, para identificar buenas prácticas y puntos de reutilización, se aborda además sobre algunas características de las herramientas que se utilizan en la migración de la CAD de los componentes FT y C.

En el **Capítulo 2:** Desarrollo de la migración, se realiza una descripción de la migración de la CAD de los componentes FT y C destacando los elementos más importantes y realizando los cambios necesarios para su funcionamiento con la nueva versión del ORM Doctrine.

En el **Capítulo 3:** Validación de la migración, se describen los resultados obtenidos en la validación de la propuesta de solución, mediante la aplicación de pruebas de rendimiento con la utilización de la herramienta JMeter.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el capítulo se muestra la base teórica que soporta la investigación. En él se enmarcan varios conceptos asociados al proceso de persistencia de datos, específicamente Doctrine. Se describen las características de los ORM al realizarse un estudio del arte de los mismos, principales tendencias y definiciones. Además se analizan las tecnologías y herramientas informáticas utilizadas en la migración de la CAD.

1.1. Conceptos relacionados con los ORM

A lo largo de la investigación fue necesario estudiar algunas definiciones para obtener un mejor entendimiento y poder ampliar el conocimiento sobre los ORM. Existen varios aspectos relacionados con términos tales como: persistencia y ORM que son necesarios para el proceso de migración de la capa de acceso a datos.

Por tanto se puede definir que la persistencia es la capacidad que tiene un objeto de perdurar o permanecer fuera del proceso que lo creó. El estado de un objeto puede ser almacenado en disco y recuperado en un futuro [2]. Por otra parte un ORM es una técnica que utiliza Programación Orientada a Objeto (POO) y los mapea directamente a la base de datos relacional; por lo que en cada lenguaje existen implementaciones propias las cuales se usan de igual manera independiente de su implementación [7].

Analizando los elementos distintivos de la persistencia y un ORM, cabe destacar que estos en su conjunto dan lugar a los framework de persistencia o marco de trabajo de persistencia que se sitúa entre la capa de negocio y la capa de acceso a datos, abstrayéndose uno del otro. Estos pueden ser vistos como una biblioteca de clases que facilita la tarea del programador al permitirle guardar objetos en bases de datos relacionales de manera lógica y eficiente, de otra manera se debería hacer manualmente, y este es un proceso tedioso, repetitivo y propenso a errores.

La utilización de un ORM presenta grandes ventajas a los desarrolladores, entre ellas se destacan:

- **Persistencia transparente:** Los objetos por sí solo no tienen el dominio para ver de qué forma serán persistidos en la base de datos.
- **Soporte de polimorfismo:** Se pueden cargar jerarquías de objetos de forma polimórfica.

- **Soporte completo de asociaciones:** Los ORM soportan el mapeo de todos los tipos de relaciones (unidireccionales y bidireccionales) que pueden existir en un modelo de objetos del dominio.
- **Soporte de caching:** En el contexto de una transacción permite disminuir la cantidad de veces que se encuesta a la base de datos, guardando en memoria los objetos que son accedidos varias veces.
- **Soporte de múltiples dialectos SQL:** Son independientes completamente del tipo de base de datos utilizada, simplemente cambian la configuración correspondiente [8].

1.2. ORM existentes

Las herramientas de mapeo objeto-relacional se encargan no solo de manejar e integrar las capacidades de los lenguajes de POO con las bases de datos relacionales, sino que además buscan reducir el código necesario para llevar a cabo las operaciones de persistencia y recuperación de objetos. Con el propósito de conocer posibles puntos de reutilización y buenas prácticas, se realiza un estudio de los ORM teniendo en cuenta los siguientes indicadores:

- Rendimiento.
- Compatibilidad con el gestor de bases de datos PostgreSQL.
- Uso del Lenguaje PHP.
- Compatibilidad con el Marco de Trabajo Sauxe.
- Comunidad que respalde su soporte.

Estos indicadores responden a la compatibilidad con el MT Sauxe, la facilidad de uso de software libre y al objetivo de la investigación, que se centra principalmente en aumentar el rendimiento a la hora de consultar grandes cantidades de información distribuidas en varias tablas.

Dado los indicadores antes descritos y usados como punto de partida, para realizar una comparación entre distintos ORM, se decide estudiar los siguientes ORM de forma tal que se concluya, cuál es el más apropiado para realizar la migración:

1.2.1. Hibernate

Es una herramienta ORM que facilita el mapeo de los atributos de una base de datos relacional con el modelo de datos de la aplicación. Este mapeo se hace mediante archivos XML o anotaciones en las entidades que permiten establecer estas relaciones. También ofrece la posibilidad de utilizar un lenguaje de consulta de datos HQL (del inglés Hibernate Query Language) [9].

Hibernate es un conjunto de proyectos relacionados, que permiten a los desarrolladores utilizar estilo POJO[2], que no son más que modelos de dominio en sus aplicaciones de manera que extienden mucho más allá del mapeo objeto relacional. Es software libre, de código abierto, distribuido bajo los términos de la licencia GNU/LGPL[3] [9].

Rendimiento:

Hibernate mejora el rendimiento al proporcionar instalaciones de almacenamiento en caché que ayudan a una recuperación más rápida de los datos de la base de datos [10]. Trata con la reflexión de Java y el aumento de clases en tiempo de ejecución utilizando una biblioteca de generación de código Java muy poderosa y de alto rendimiento llamada CGLIB. Este se utiliza para extender clases Java e implementar interfaces Java en tiempo de ejecución [9].

Compatibilidad con el gestor de base de datos PostgreSQL:

Este soporta Oracle, MySQL, PostgreSQL, Front Base y MSSQL.

Uso del Lenguaje PHP:

Hibernate utiliza como lenguaje de programación Java.

Compatibilidad con el MT Sauxe

La compatibilidad con el MT Sauxe se ve limitada debido a que ambos utilizan lenguajes diferentes.

[2] **POJO:** del inglés (Plain Old Java Object), utilizado por programadores que implementan con Java para enfatizar el uso de clases simples y que no dependen de un MT en especial.

[3] **GNU/LGPL:** Licencia Pública General Reducida de GNU, o más conocida por su nombre en inglés GNU Lesser General Public License.

Comunidad que respalde su soporte:

Hibernate cuenta con una amplia comunidad y con cientos de colaboradores a lo largo de todo el mundo por lo que su soporte está garantizado.

Buenas prácticas:

- Por lo general, no es aconsejable crear una conexión cada vez que se interaccione con la base de datos. Para evitar esta situación, las aplicaciones Java, suelen utilizar una piscina (del inglés pool) de conexiones. Cada subproceso de la aplicación que realiza solicitudes sobre la base de datos, solicita una conexión a la piscina, devolviéndola cuando todas las operaciones SQL han sido ejecutadas.
- Cuando ocurra una excepción, se debe deshacer la operación y cerrar la sesión.
- Mapear cada clase en su propio fichero, evitando usar un solo documento para mapear todas las clases.
- No utilice con frecuencia la recuperación temprana. Use proxies y colecciones perezosas para la mayoría de asociaciones a clases que probablemente no se encuentren en el caché de segundo nivel [11].

1.2.2. Propel

Propel es una herramienta de mapeo objeto-relacional de código abierto escrito en PHP y es además una parte integral del MT Symfony y su ORM por defecto. Permite el acceso a la base de datos utilizando un conjunto de objetos, proporcionando una API[4] simple para almacenar y recuperar datos. Este le brinda al desarrollador de aplicaciones web las herramientas para trabajar con bases de datos, de la misma manera que se trabaja con otras clases y objetos en PHP. Le da a la base de datos una API bien definida utilizando los estándares PHP5, carga automática e iteradores [12]. La función primaria de Propel es proveer un mapa entre las clases de PHP y tablas de bases de datos.

[4] **API:** Interfaz de programación de aplicaciones (IPA) o *API* (del *inglés Application Programming Interface*).

Rendimiento:

Lo más importante es que Propel utiliza PDO (del inglés PHP Data Objects) en lugar de Creole como DBAL (Capa de Abstracción de Base de Datos), ofreciendo una significante mejora de rendimiento [13].

Compatibilidad con el gestor de base de datos PostgreSQL:

Este soporta MySQL, PostgreSQL, SQLite, MSSQL y Oracle [12].

Uso del Lenguaje PHP:

Utiliza PHP5 o una versión superior.

Compatibilidad con el MT Sauxe:

Puede adaptarse perfectamente al MT Sauxe debido a que está basado en PHP como lenguaje de programación y soporta el gestor de base de datos PostgreSQL.

Comunidad que respalde su soporte:

Propel posee una amplia comunidad y con un elevado prestigio mundialmente por sus potencialidades, aunque se trabaja en el aumento de su documentación que hasta el año 2010 era escasa [14].

Buenas prácticas:

- A cada URL[5] se le debe crear la ruta asociada. Y esto es obligatorio si se eliminan las reglas de enrutamiento por defecto.
- En caso de que se elimine un método generado se debe eliminar la plantilla generada asociada a ese método.

1.2.3. Django

Es un entorno de desarrollo web escrito en Python[6] que fomenta el desarrollo rápido y el diseño limpio y pragmático. Django es un MT de código abierto que permite construir aplicaciones web más rápido y con

[5] **URL:** es un localizador uniforme de recursos, denominado URL (sigla en inglés de Uniform Resource Locator).

[6] **Python:** es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible.

menos código. Está compuesto por un mapeo objeto relacional para convertir los modelos de datos, un apoyo en configuración de URL, un sistema completo de plantillas, sistemas para realizar procesos de solicitud y administración de base de datos relacional, lo que permite un óptimo desarrollo de aplicaciones [15].

Rendimiento:

Django es muy potente, porque permite obtener provecho de las ventajas del lenguaje para el cual está diseñado: Python.

Compatibilidad con el gestor de base de datos PostgreSQL:

A pesar de que Django es compatible con el gestor de base de datos PostgreSQL se recomienda que para su uso se instale el paquete psycopg2*;* el uso de PostgreSQL en Django logra un delicado equilibrio entre costo, características, velocidad y estabilidad [15].

Uso del Lenguaje PHP:

Django está creado sobre el lenguaje Python.

Compatibilidad con el MT Sauxe:

La compatibilidad de Django con el MT Sauxe se ve limitada debido a que los dos utilizan lenguajes diferentes.

Comunidad que respalde su soporte:

Django cuenta con una comunidad en crecimiento y ya se ha extendido a varios idiomas como son: español, inglés y portugués.

Buenas prácticas:

- Sigue el principio DRY (conocido también como Una vez y sólo una).
- Utiliza una modificación de la arquitectura Modelo-Vista-Controlador (MVC) está forma de trabajar permite que sea pragmático.

1.2.4. Doctrine 1.2.2

Es un ORM escrito en PHP que proporciona una capa de persistencia para objetos PHP. Es una capa de abstracción que se sitúa justo encima de un sistema de gestión de base de datos. Además es la versión actual que utiliza el MT Sauxe para tratar la persistencia de los datos. Este ORM es analizado por los indicadores que fueron comparados los anteriores ORM, obteniendo como resultado [16]:

Compatibilidad con el gestor de base de datos PostgreSQL:

Doctrine 1.2.2 soporta múltiples gestores de base de datos (MySQL, PostgreSQL, SQLite, MSSQL y Oracle).

Uso del Lenguaje PHP:

Este está escrito en PHP.

Compatibilidad con el MT Sauxe:

Es compatible completamente con el MT Sauxe debido a que este utiliza en su CAD dicha versión de este marco de persistencia de datos.

Rendimiento:

La forma en que esta versión maneja la herencia, donde se debe hacer referencia a numerosas tablas para obtener los datos, provoca que la aplicación tenga un rendimiento reducido.

Comunidad que respalde su soporte:

Doctrine 1.2.2 no cuenta con una comunidad que respalde su soporte, lo que provoca que no se corrijan vulnerabilidades en el código que puedan atentar contra el rendimiento.

Buenas prácticas:

- Bajo nivel de configuración.
- Puede generar clases a partir de una base de datos existente y después se pueden especificar relaciones y añadir funcionalidad extra a las clases autogeneradas.
- No es necesario generar o mantener complejos esquemas XML de base de datos como en otros frameworks.

1.2.5. Doctrine 2.0.6

Es la versión superior de ORM para PHP 5.3.0+ que proporciona persistencia transparente de objetos PHP. Se sitúa en la parte superior de una poderosa capa de abstracción de base de datos [17].

Doctrine 2.0.6 implementa el diseño Mapa de Datos (del inglés Data Mapper) proponiendo que los objetos de negocio sean POCO[7] (POJO en Java), o sea, que no tengan nada de código que los acople a una tecnología de persistencia, ya que no es su función manejarla. Los objetos de negocio solo deben preocuparse por cumplir con el modelo del dominio del diagrama de clases con las asociaciones pertinentes.

Rendimiento

- El uso de PHP 5.3 brinda un buen rendimiento.
- Mejor hidratación y algoritmo optimizado.
- Nuevas implementaciones de consulta y el resultado caché.
- El código más explícito, devuelve como resultados: el código mejor y más rápido [17].

Compatibilidad con el gestor de base de datos PostgreSQL:

Doctrine 2.0.6 soporta múltiples gestores de base de datos (MySQL, PostgreSQL, SQLite, MSSQL y Oracle).

Uso del Lenguaje PHP:

Este está escrito en PHP pero solo compatible con las versiones superiores a 5.3.0.

Compatibilidad con el MT Sauxe:

La compatibilidad con el MT Sauxe está garantizada debido a que este utiliza en su CAD la versión previa de este marco de persistencia de datos.

[7] **POCO:** Componentes Portables (en inglés Portable Components).

Comunidad que respalde su soporte:

Cuenta con una amplia comunidad y colaboradores alrededor de todo el mundo, es uno de los marcos de persistencias más usados y con mayor prestigio a escala mundial.

Buenas prácticas

- No utiliza propiedades públicas en las entidades.
- Es importante limitar las relaciones tanto como sea posible.
- Se deben evitar las asociaciones bidireccionales, si es posible.
- Eliminar las asociaciones que no sean esenciales [17].

1.3. Valoración general

Los ORM Hibernate, Propel y Django no se tienen en cuenta para darle solución al problema planteado dado que no son compatibles con el MT Sauxe y no usan PHP como lenguaje de programación. Por otra parte Doctrine 1.2.2 y Doctrine 2.0.6 cumplen con los indicadores anteriormente descritos, pero aún así presentan diferencias dado que están centrados en enfoques completamente diferentes. Doctrine 1.2.2 implementa el diseño Registros Activos, se trata de una clase que se encarga de implementar todas las operaciones de consulta y modificación de una tabla concreta de la base de datos, mientras Doctrine 2.0.6 implementa el diseño Mapeo de Datos que soporta modelos de dominios complejos y enormes que transforman de forma transparente los objetos del dominio en tablas y entidades del mundo relacional de las bases de datos, esa es la diferencia fundamental. Doctrine 2.0.6 requiere PHP 5.3 o posterior y utiliza sus beneficios como espacios de nombres. Doctrine 2.0.6 se divide en conjunto de subproyectos más pequeños: Doctrine Commons, Doctrine DBAL, Mapas de Doctrine ORM, en esta versión se ven reflejadas las potencialidades de los ORM anteriormente estudiados y es la continuidad de Doctrine 1.2.2 usada en el SIPAC.

El rendimiento en Doctrine 2.0.6 es superior a Doctrine 1.2.2. En la **Tabla 1** se muestra un estudio realizado por el equipo de desarrollo del ORM Doctrine a un sistema haciendo uso de dos versiones diferentes (Doctrine 1.2.2 y Doctrine 2.0.6), la comparación se hizo teniendo en cuenta el tiempo de ejecución a la hora de insertar 5000 y 10000 registros respectivamente en una base de datos [17].

Tabla 1. Comparación de rendimiento en las versiones de Doctrine [17].

Versión	Registros	Tiempo	Registros	Tiempo
Doctrine 1.2.2	5000	4,3 seg.	10000	7 seg.
Doctrine 2.0.6	5000	1,4 seg.	10000	3,5 seg.

El estudio de los marcos de persistencia de datos permitió conocer que el más usado para PHP es Doctrine [18]. Demostró las potencialidades que representa el uso de la versión Doctrine 2.0.6 debido a que esta brinda un alto rendimiento y en ella se encuentran reflejadas las potencialidades que implementan los demás marcos de persistencia de datos [17].

1.4. Procedimientos para la migración

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 CAD, 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,* [4] este cuenta con tres etapas las cuales se describen a continuación:

- **Etapa inicial:** en esta etapa 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 CAD utilizando Doctrine 1.2.2. Por otra parte se integra el SIPAC con Doctrine 2.0.6 y se seleccionan las bibliotecas de Doctrine 2.0.6.
- **Etapa de implementación:** se garantiza la conectividad a la base de datos y los parámetros de configuración.
- **Etapa de prueba:** tiene como objetivo supervisar que la implementación se realizó satisfactoriamente probando todas las funcionalidades migradas [4].

1.5. Herramientas, tecnologías y lenguajes

Para llevar a cabo el proceso de migración, al igual que todo proceso de desarrollo de software, es necesario definir cuáles herramientas, lenguajes y tecnologías se usarán, por tanto las usadas para este proceso son las definidas en el documento Vista de Entorno de Desarrollo Tecnológico para el MT Sauxe:

1.5.1. Subversion v1.6.6

Subversion es un sistema de control de versiones libre (código abierto) que maneja ficheros y directorios a través del tiempo. Contiene un árbol de ficheros en un repositorio central y el repositorio es como un servidor de ficheros ordinario, exceptuando que guarda todos los cambios hechos a sus ficheros y directorios. Esto permite recuperar versiones antiguas de los datos, o examinar la historia de cómo cambiaron sus datos [19].

Utiliza el concepto de revisión para guardar los cambios producidos en el repositorio. Entre dos revisiones sólo guarda el conjunto de modificaciones, optimizando así al máximo el uso de espacio en disco. Permite al usuario crear, copiar y borrar carpetas con la misma flexibilidad con la que lo haría si estuviese en su disco duro local. Dada su flexibilidad, es necesaria la aplicación de buenas prácticas para llevar a cabo una correcta gestión de las versiones del software generado [20].

1.5.2. PostgreSQL v9.1

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (del inglés Berkeley Software Distribution) para bases de datos y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones ha logrado incorporar las potencialidades de otros sistemas gestores de bases de datos incluso comerciales. PostgreSQL utiliza un modelo cliente-servidor y es multiprocesos en vez de multihilos para lograr la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando [21].

Durante su desarrollo las características que más se evidencian son: estabilidad, potencia, robustez, facilidad de administración e implementación de estándares. Este presenta un rendimiento elevado con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema [21].

Soporta distintos tipos de datos: tipo fecha, monetarios, elementos gráficos, datos sobre la red, cadenas de bits y permite la creación de tipos propios. Incluye herencia entre tablas, posee múltiples métodos de autentificación, contiene documentación completa y es multiplataforma [21].

1.5.3. NetBeans v7.4

NetBeans IDE es un entorno de desarrollo integrado (IDE, del inglés Integrated Development Environment), modular, de base estándar (normalizado), escrito en el lenguaje de programación Java. El

proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general (framework) para compilar cualquier tipo de aplicación [22].

La plataforma de NetBeans es principalmente un MT de aplicación genérica para las aplicaciones de escritorio en Java, aunque también puede usarse para aplicaciones web reconociendo lenguajes como PHP, JavaScript, C y C++. El beneficio principal de esta plataforma es su arquitectura modular predefinida. Los beneficios secundarios son las soluciones reusables en combinación con las herramientas proporcionadas por su SDK[8], NetBeans IDE [22].

1.5.4. Apache v2.2

Apache es un servidor web gratuito y potente, que ofrece un servicio estable y sencillo de mantener y configurar. Es válido destacar las siguientes características: es multiplataforma (aunque idealmente está preparado para funcionar bajo Linux), muy sencillo de configurar, es open-source (código abierto), muy útil para proveedores de servicios de internet que requieran miles de sitios pequeños con páginas estáticas, amplias bibliotecas de PHP y Perl a disposición de los programadores, posee diversos módulos que permiten incorporarle nuevas funcionalidades y es capaz de utilizar lenguajes como PHP, TCL, Python, entre otros [23].

1.5.5. Sauxe v2.0

Contiene un conjunto de componentes que proveen la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor agilidad en el proceso de desarrollo y las aplicaciones de gestión. Sauxe está basado en el MT Zend [24], además utiliza en la CAD el Lenguaje de Consulta de Doctrine (DQL). Utiliza ExtJS en la capa de presentación, por la amplia gama de componentes que se pueden reutilizar y para mostrarle al usuario una interfaz más amigable. Sauxe está sustentado por la integración de estas tecnologías, además tiene un conjunto de componentes reutilizables que logran una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo [25].

[8] **SDK**: Kit de desarrollo de software o SDK (siglas en inglés de Software Development Kit).

1.5.6. PHP v5.3.10

PHP cuyas siglas responden a un acrónimo recursivo (PHP: Hypertext Preprocessor) es un lenguaje de programación interpretado, multiplataforma, originalmente diseñado para la creación de páginas web dinámicas con acceso a información almacenada en una base de datos. Sigue un estilo clásico debido a que es un lenguaje de programación con variables, sentencias condicionales, bucles y funciones. Está más cercano a JavaScript o al lenguaje C, pero a diferencia de Java o JavaScript que se ejecutan en el navegador, PHP se ejecuta en el servidor [26] [27].

1.5.7 Lenguaje de Consulta de Doctrine(DQL)

DQL es el lenguaje de consulta de Doctrine y es un objeto derivado del lenguaje de consulta que es muy similar al lenguaje de consulta Hibernate (HQL) o al lenguaje de consulta persistente de Java (JPQL). Es insensible a mayúsculas y minúsculas, salvo en nombres de clase, campos y espacios de nombres, en los cuales sí distingue entre mayúsculas y minúsculas [17]. DQL como un lenguaje de consulta tiene construcciones SELECT, UPDATE y DELETE que asignan a sus correspondientes tipos de declaraciones SQL. Las instrucciones INSERT no se permiten, porque las entidades y sus relaciones se tienen que introducir en el contexto de la persistencia a través de EntityManager->persist() para garantizar la coherencia de los objetos del modelo [17].

1.6. Conclusiones parciales

El estudio de los ORM demostró la existencia de características comunes entre ellos y que en Doctrine 2.0.6 se encuentran implementadas las principales potencialidades de los otros ORM. Además el estudio de los principales conceptos relacionados con la persistencia y el acceso a los datos, arrojó la conclusión de que uno de los principales elementos para mejorar el rendimiento de SIPAC es migrar su CAD a Doctrine 2.0.6, dado que:

- Doctrine 2.0.6 muestra un mejor rendimiento en comparación con Doctrine 1.2.2.
- Es mucho más rápido, dado que emplea el 30% menos en uso de memoria.
- Las clases no heredan de ninguna otra clase, aumentando el tiempo de respuesta al realizar peticiones sobre estas entidades.
- Tiene una nueva implementación de consultas aumentando el resultado en cache.

- Utiliza PHP5 o superior, lo que proporciona un mejor rendimiento puesto que trae implementadas funcionalidades que favorecen el rendimiento [17].

De esta forma se sentaron las bases para la migración de la CAD de los componentes FT y C.

CAPÍTULO 2: DESARROLLO DE LA MIGRACIÓN

En este capítulo se describe la implementación realizada en la migración de la CAD de los componentes FT y C a Doctrine 2.0.6. Se detallan sus elementos más importantes y se especifican las nuevas configuraciones y estructura de la CAD. Además se hace una descripción detallada de las principales acciones que se deben cumplir para realizar la migración a Doctrine 2.0.6, especificando además el procedimiento establecido con anterioridad, el cual define tres etapas: Inicial, Implementación y Pruebas. Describiendo en este capítulo la fase inicial y la de implementación.

2.1 Etapa inicial

En esta etapa se procede a realizar dos tareas fundamentales:

1. El análisis de la arquitectura de la aplicación para evaluar el contenido de cada una de las capas que contiene y el estudio de la estructura de carpetas usando Doctrine 1.2.2 de los componentes FT y C.
2. Se realiza la integración del ORM Doctrine 2 con el MT Zend al utilizar la versión 2.2 del MT Sauxe y como consecuencia se hace necesaria la creación de directorios donde se añaden las clases y funcionalidades para la integración. Además se adicionan los parámetros de configuración y las nuevas bibliotecas.

2.1.1 Arquitectura del SIPAC

El análisis de la arquitectura del SIPAC es la primera tarea a realizar en la etapa inicial, lo que demuestra que el sistema está guiado por el modelo de desarrollo basado en componentes, cuenta con una arquitectura en capas y el patrón MVC, básicamente está compuesto por cinco niveles o capas, las cuales se describen a continuación:

1. Capa de Presentación: en esta capa se emplean las facilidades que brinda el MT ExtJS, dígase la biblioteca de JavaScript para la construcción de interfaces a la vista de los usuarios. ExtJS centra su desarrollo en tres componentes fundamentales JS-File, CSS-File y Client-page y Server-Page [25].

2. Capa de Control o Negocio: en esta capa se emplea el patrón MVC. De forma vertical al modelo descrito hasta este momento, estarán los aspectos fundamentales del sistema así como el encargado del tratamiento de la seguridad a nivel de aplicación web [25].

3. Capa de Acceso a Datos: en esta capa estará presente el ORM Doctrine para la comunicación con el servidor de datos mediante el protocolo PDO[9] [25].

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 [25].

5. Capa de Servicio: en esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí [25].

Luego del estudio realizado de estas capas y conocer su funcionamiento, cabe destacar que estas se comunican mediante el mecanismo de integración IoC[10], proporcionando un conjunto de funcionalidades tanto de seguridad como de negocio [25]. Aún cuando el contenido de la migración está centrado en la CAD**,** es necesario garantizar la comunicación de la CAD con las restantes.

2.1.1 Estructura de carpeta de los componentes Flujo de Trabajo y Configuración

Como parte de la primera tarea de la etapa inicial, está además el estudio de la estructura de carpetas la cual está compuesta por un conjunto de componentes que presentan la misma estructura de carpetas en la CAD. En la **Figura 1** se muestra los componentes FT y C con la estructura de carpetas que garantiza el correcto funcionamiento del MT con la versión 1.2.2 del ORM Doctrine.

Es importante realizar un profundo análisis debido a que se realizan modificaciones en la CAD durante la migración para satisfacer las exigencias de Doctrine 2.0.6. Además, los cambios que se realicen garantizarán la comunicación con las restantes capas y de esta manera no se afecte ni el negocio ni la interacción con el MT. Siendo los principales cambios para la versión 2.0.6, la generación de tres directorios, Repository, Entity y Proxy los cuales aseguran la comunicación con las restantes capas y la interacción con el MT por la correcta ubicación de los ficheros dentro de los directorios anteriormente mencionados.

[9] **PDO:** acrónimo del inglés PHP *Data Objects*, es una extensión que provee una capa de abstracción de acceso a datos para PHP5.

[10] **IoC:** (siglas de *Inversion of Control*) es el mecanismo de integración de los componentes de Sauxe.

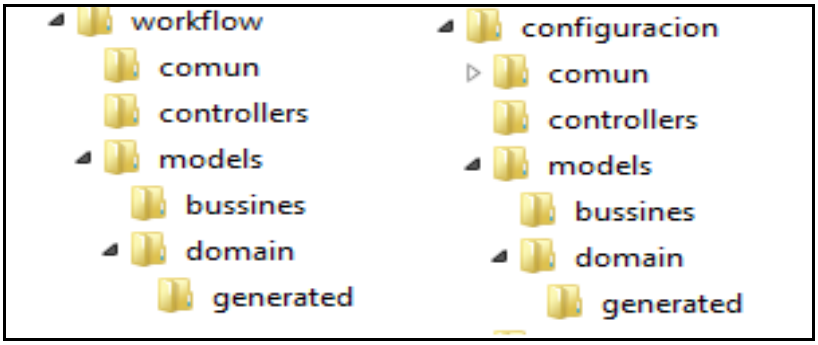


Figura 1. Estructura de carpetas de los componentes FT y C para Doctrine 1.2.2.

2.1.2 Integración de Doctrine 2.0.6 con el MT Zend

La acción de integración es la base para realizar la implementación de la migración y para ello se definen las bibliotecas que se utilizan y las clases que se modifican creando las funcionalidades necesarias, logrando que el MT Zend funcione de manera correcta con Doctrine 2.0.6 al utilizar Sauxe en su versión 2.2 o superior [4].

Para realizar la integración, primeramente se:

- Seleccionan las clases necesarias para comenzar a implementar.
- Seleccionan las nuevas bibliotecas a utilizar.
- Crean los directorios para ubicar las nuevas clases.
- Precisan los nuevos parámetros de configuración.

2.2 Etapa de implementación

En esta etapa es necesario realizar modificaciones debido a que el funcionamiento en la nueva versión es diferente a la anterior. Una de las tareas es rediseñar la arquitectura base del sistema, donde se debe establecer una nueva estructura de carpetas para los componentes FT y C; lo que debe garantizar la ubicación correcta de los nuevos ficheros y funcionalidades a utilizar con Doctrine 2.0.6. Otra de las tareas es redefinir la CAD de estos componentes, donde se mapean las tablas y se definen las relaciones de la base de datos para la versión 2.0.6 de Doctrine. También se vuelven a implementar todos los métodos y funcionalidades de esta capa utilizando el DQL de la nueva versión.

2.2.1 Nueva estructura de carpetas de los componentes Flujo de Trabajo y Configuración

Doctrine 2.0.6 propone la utilización de una nueva estructura de carpetas para las clases del modelo. Contiene nuevos directorios (Entity, Repository y Proxy) con el fin de organizar los ficheros generados por el mapeo, ya sea por su tipo o su función. En la **Figura 2** se muestra como queda conformada esta nueva forma:

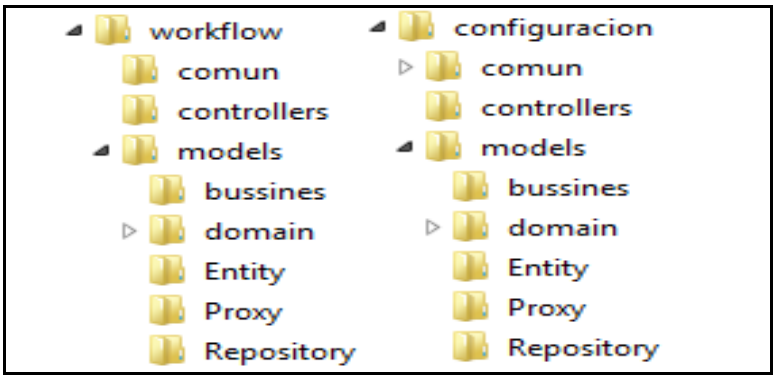


Figura 2. Nueva estructura de carpetas de los componentes FT y C para Doctrine 2.0.6.

Equivalencia entre la anterior y la nueva estructura de carpetas

La estructura anterior y la nueva, como se muestra en la **Figura 3,** son diferentes aunque se puede establecer cierta relación entre ellas, puesto que en algunos casos sólo basta con cambiar de ubicación los ficheros. La siguiente figura visualiza la nueva ubicación de los ficheros del modelo de los componentes FT y C además de la equivalencia que existe entre las versiones 1.x y las 2.x:

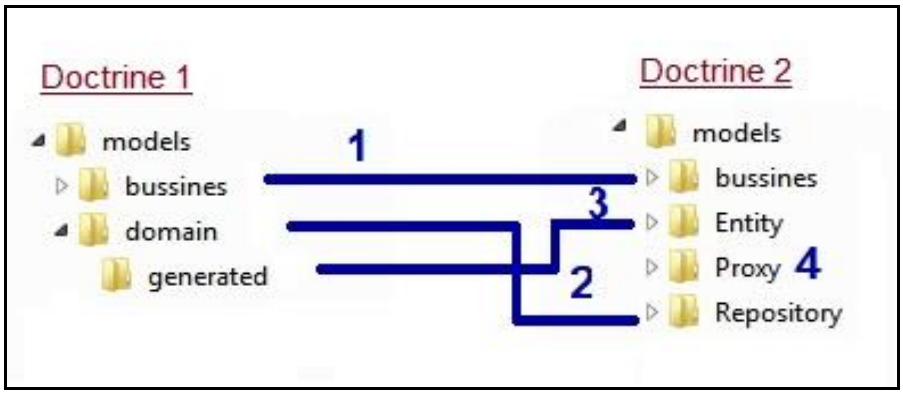


Figura 3. Equivalencia entre ambas estructuras de carpetas.

1- Los ficheros localizados en la carpeta **bussines** que son los encargados de manejar el negocio de la aplicación se mantienen igual dentro de la carpeta **models**.

2- Las funcionalidades con las consultas que se encuentran en los ficheros localizados en la carpeta **domain,** la deben realizar los ficheros ubicados dentro de la carpeta **Repository**, además se deben modificar de acuerdo a la nueva estructura de Doctrine 2.0.6, utilizando el DQL de esta versión. A continuación se detallan los principales cambios utilizando el DQL de esta versión con respecto a la versión anterior, la función execute() propia del DQL de la versión 1.2.2 es sustituida por la función getQuery() de la versión 2.0.6 mientras que la función toArray() de la versión 1.2.2 es suplantada por la función getArrayResult(), siendo sustituidas además las funciones limit() y offset() de la versión anterior por setFirstResult() y setMaxResults() de la versión 2.0.6 respectivamente.

En la carpeta **Repository** se encuentran los ficheros Repository generados como resultado del mapeo. Por cada tabla en la base de datos se genera un fichero de este tipo, su nomenclatura está compuesta por el nombre del fichero de mapeo de la tabla y la palabra Repository al final. Estos ficheros son clases que extienden de la clase ***Doctrine\ORM\EntityRepository***.

3- Los ficheros que representan las tablas mapeadas ubicados en **generated** son generados con su nueva estructura en el directorio **Entity** de la carpeta **models**.

4- En la carpeta **Proxy** se ubican los ficheros generados como resultado del mapeo que contienen las clases Proxy usadas por Doctrine 2.0.6. Estas clases son objetos que se ponen en el lugar del objeto real y se utilizan para realizar varias funciones, principalmente, para la transparencia de carga diferida. Los objetos proxy con sus instalaciones de carga diferida ayudan a mantener el subconjunto de objetos que ya están en la memoria conectada con el resto de los objetos [17].

2.2.2 Redefinición de la Capa de Acceso a Datos

Debido a la diferencia existente entre las versiones 1 y 2.0.6 de Doctrine y el rediseño de la CAD, se creó una nueva estructura de carpetas, necesaria para el Trabajo con Doctrine 2.0.6, además se procede a reimplementar todos los métodos y consultas de los componentes FT y C. Para ello es necesario mapear todas las tablas adaptándolas a la nueva estructura de la versión 2.0.6 de Doctrine y la migración manual

de cada una de las clases ya mapeadas. Esta tarea requiere un tiempo que representa un 45% del tiempo total utilizado para la migración, más allá de su complejidad, por el monto de objetos que se deben mapear y por la cantidad de funcionalidades que se implementan nuevamente.

Se mapearon un total de 20 clases, donde por cada una de ellas se generan tres ficheros el Repository, el Proxy y el Entity, lo que influyó en el cambio de la estructura de carpetas de los componentes FT y C. Se modificaron todos los ficheros Model de los componentes, los principales cambios fueron la sustitución de la función save() por la función flush() y delete() por remove() además de la instancia del EntityManager(), garantizando el acceso al Repository de cada entidad, en el cual se encuentran implementadas las consultas. Como resultado de haber vuelto a implementar las funcionalidades se cambiaron en más de 190 consultas en la CAD de los componentes FT y C, los parametros de las funciones, se optimizaron las consultas usando funciones DQL que agrupan y eliminan sentencias con bajo rendimiento como múltiples innerjoin o ciclos de numerosas transacciones.

2.2.3 Flujo de datos de la Capa de Acceso a Datos

La capa de acceso a datos con la nueva versión de Doctrine adquiere otro funcionamiento interno, donde cambia por completo su flujo de datos. Este transita desde las clases controladoras que invoca a una funcionalidad en el Model, en el cual se encuentran implementados los métodos que manejan el negocio de los componentes FT y C, hasta los ficheros Repositorys. En este fichero se implementan todas las consultas a la base de datos, se realizan las peticiones y se envían los objetos necesarios o que fueron solicitados. Doctrine 1.2.2 utilizaba la Model para realizar peticiones directamente a la base de datos, en su lugar Doctrine 2.0.6, utiliza una instancia del EntityManager, donde puede acceder a los repositorios.

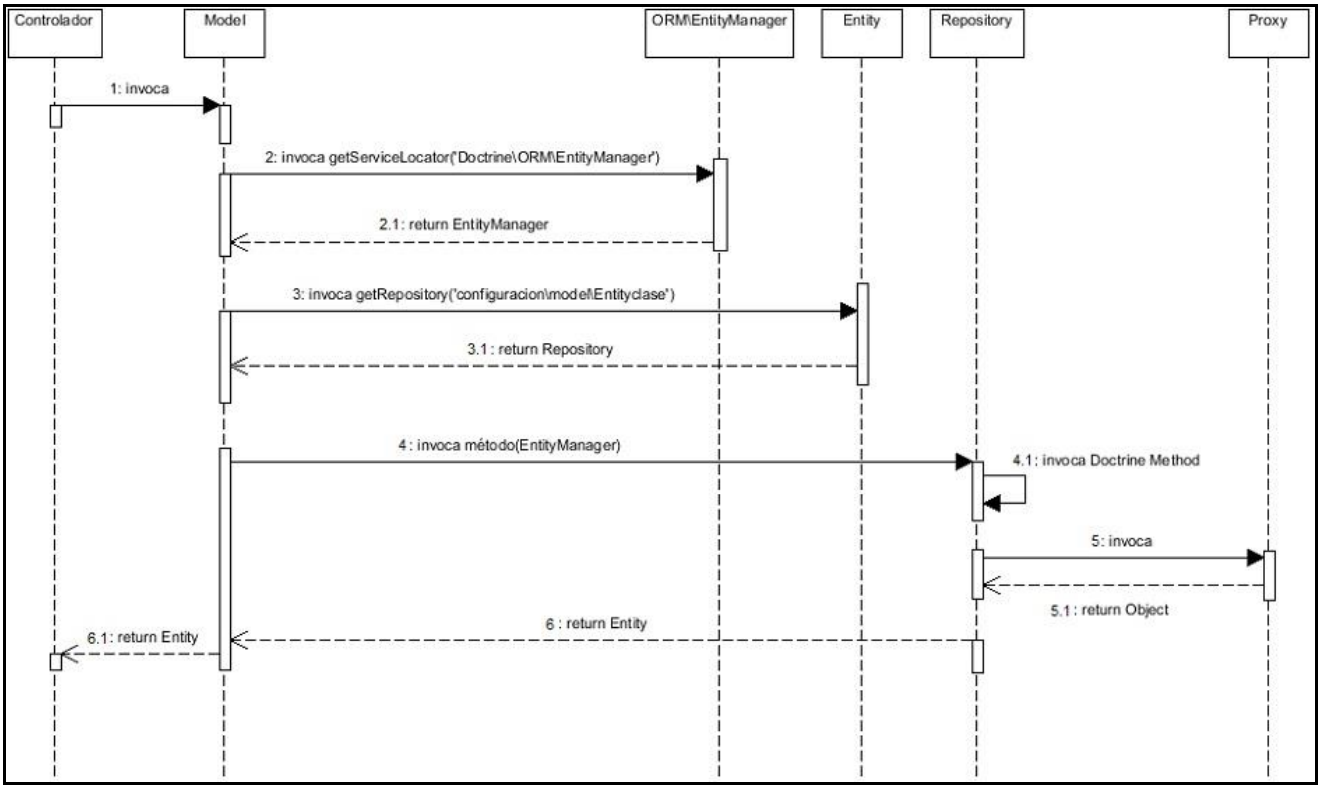


Figura 4. Secuencia desde el Controlador hasta el Repository (Flujo de la CAD de Doctrine 2.0.6)

2.2.4 Mapeo de las tablas

Después de definida esta nueva estructura de carpeta se procede a aplicar las nuevas formas de mapeo, este mapeo para la persistencia de datos mediante el framework Doctrine 1.2.2 es muy diferente a Doctrine 2.0.6, aunque ambas definen el mapeo entre un tipo de dato PHP y un tipo de dato SQL. Doctrine 1.2.2 genera entidades que extienden de una clase base, haciendo uso de funciones predeterminadas para la definición de tablas y columnas. Doctrine 2.0.6 proporciona diferentes maneras de especificar el mapeo objeto-relacional [17]:

- Por anotaciones en el código fuente (docblock).
- Desde un archivo con formato XML.
- Desde un archivo con formato YAML.

Las anotaciones docblock de Doctrine 2.0.6 cuentan con apoyo para espacios de nombre (*namespaces*) y anotaciones anidadas. Específicamente para la realización de la migración se emplea la anotación

docblock porque suministra asignaciones de metadatos objeto-relacional. Es preciso señalar, que en un mismo entorno no se pueden mezclar diferentes formas de definir los metadatos.

Doctrine permite guardar y obtener objetos enteros a partir de la información de la base de datos. Para esto mapea una clase PHP a una tabla de la base de datos y después mapea las propiedades de la clase PHP a las columnas de esta tabla; en la siguiente figura se puede apreciar esta acción:

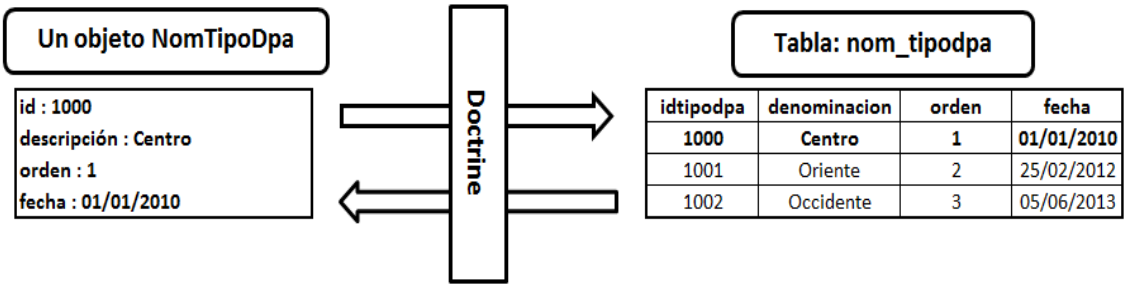


Figura 5. Ejemplo de mapeo un objeto PHP a una tabla.

Mapear la persistencia de una clase y definir una tabla

Doctrine 1.x crea una clase que extiende de una base llamada Doctrine_Record con una funcionalidad llamada setTableDefinition. Por su parte Doctrine 2.x, con el fin de marcar una clase para ser persistida en la base de datos, lo hace con la anotación *@Entity*, por defecto la entidad será persistida con el mismo nombre de la clase, pero se puede modificar utilizando la anotación *@Table*:

Doctrine 1.2.2

```
abstract class BaseConfDoc extends Doctrine_Record{
        public function setTableDefinition (){
        $this->setTableName ('conf_doc');
}
} ...
```

Doctrine 2.06

```
<?php
namespace configuracion\workflow\models\Entity;
use Doctrine\ORM\Mapping as ORM;
/**
*ModFlujotrabajo.confDoc
*@ORM\Table(name="mod_flujotrabajo.conf_doc")
*ORM\Entity(repositoryClass="configuracion\workflow\models\Repository\ConfDocRepository")
*/…
```

Con esto ahora las instancias de ConfDoc serán persistidas en una tabla llamada conf_doc en el esquema mod_flujotrabajo en la base de datos.

Mapear la persistencia de las propiedades

Las propiedades, que son representadas en forma de columnas; Doctrine 1.2.2 lo hace con la función hasColumn(), mientras que Doctrine 2.0.6 lo hace de manera similar pero utilizando las anotaciones.

Doctrine 1.2.2

```
$this->hasColumn('denominacion', 'character varying', null, array('notnull' => false, 'primary' => false));
```

Doctrine 2.0.6

```
/**
* @var string
*
* @ORM\Column(name="idtipodoc", type="decimal", precision=19, scale=0, nullable=false)
* @ORM\Id
* @ORM\GeneratedValue(strategy="SEQUENCE")
*                    @ORM\SequenceGenerator(sequenceName="mod_flujotrabajo.conf_doc_idtipodoc_seq",
allocationSize=1, initialValue=1)
*/
private $idtipodoc;
```

Cada clase Entity necesita una clave primaria que identifique unívocamente a la entidad. Se designa la propiedad que se usará como identificador con la anotación ***@Id*** y si es un valor generado automáticamente se utiliza la anotación **@GeneratedValue** y como estrategia **SEQUENCE** a través de **@SequenceGenerator** [17].

Relaciones entre tablas

La representación de las relaciones entre tablas es esencial para evitar datos redundantes, establecer vínculos o relaciones entre las tablas, principalmente realizar la representación de entidades de mapeo. El mapeo de objetos relacionales en Doctrine 1.2.2, se realiza mediante dos ficheros de mapeo, el fichero Base el cual extiende de (Doctrine Record) y otro que a su vez extiende del fichero Base. En el cual se especifican las relaciones entre clases, utilizando la función setUp ().

Doctrine 1.2.2

Se está especificando que un usuario tiene una única instancia del objeto mapeado por la entidad ConfDoc

Doctrine 2.0.6

```
class DatEstado
/**
*@var \Doctrine\Common\Collections\Collection
*@ORM\ManyToMany(targetEntity="ConfDoc", inversedBy="idestado")
*@ORM\JoinTable(name="mod_flujotrabajo.conf_doc_dat_estado", joinColumns={
*@ORM\JoinColumn(name="idestado", referencedColumnName="idestado")
*},
*inverseJoinColumns={
*@ORM\JoinColumn(name="idtipodoc", referencedColumnName="idtipodoc")
*}
*)
*/
```

En el caso particular de Doctrine 2.0.6 no se genera un fichero por la tabla en base de datos que guarda las llaves primarias de las tablas que se unen, en este caso se hace referencia a la entidad con la que se

tiene la relación y con la anotación ***@JoinTable*** se le dice cuál tabla es la relacionada. Otro aspecto que se debe tener en cuenta es que en estos casos se genera el constructor y dentro se inicializa la variable de la relación ***ManyToMany*** como un ***ArrayCollection*** () para que este no quede indefinido:

```
/**
*Constructor
*/
public function __construct(){
$this->idtipodoc = new \Doctrine\Common\Collections\ArrayCollection();
}…
```

Con lo antes descrito se evidencia que las anotaciones docblock se pueden ver como comentarios que se emplean en clases PHP, que para el caso de Doctrine 2.0.6 este las utiliza para garantizar el mapeo de las entidades. Además, se pueden representar todas las formas de mapeos que se utilizaban con la versión anterior realizando las mismas funciones.

2.2.5 Funcionalidades creadas en las clases Repository

Dentro de la CAD, como se mencionó anteriormente, todas las consultas y funcionalidades serán implementadas en los ficheros *Repository* utilizando el DQL de Doctrine 2.0.6. En la versión anterior, la implementación de estas se realizaba en el directorio *domain* y utilizaban el DQL de Doctrine 1.2.2. Esto se realizaba haciendo uso de varias funciones definidas por el propio lenguaje de consultas de Doctrine 1.2.2, las cuales realizaban muchas peticiones al navegador afectando así el rendimiento de la aplicación. La manera en que quedan implementadas se representa de la siguiente manera:

```
public function GetPorLimite($inicio,$limite) {
        $query = new Doctrine_Query ();
        $result = $query->from('ConfDoc')
        ->limit($limite)
        ->offset($inicio)
        ->setHydrationMode (Doctrine :: HYDRATE_ARRAY )
        ->execute ();
        $cant = $query->select('count(t.idtipodoc) as cant')
        ->from('ConfDoc t')
```

```
            ->setHydrationMode ( Doctrine :: HYDRATE_ARRAY )
            ->execute();
            $result = array('data' => $result, 'cant' => $cant[0]['cant']);
            return $result;
        }
```

En Doctrine 2.0.6 las funcionalidades deben tener como parámetro además de los necesarios para la ejecución de las consultas, una variable que posea una instancia de la entidad EntityManager ($_em). La implementación de la funcionalidad mostrada anteriormente, después de volver a implementarla quedaría de la siguiente manera en Doctrine 2.0.6:

```
public function GetPorLimite($inicio,$limite,$_em){
        $dql = "Select c from configuracion\workflow\ConfDoc c";
        $query = $_em->createQuery($dql);
        $query->setFirstResult($inicio);
        $query->setMaxResults($limite);
        $resul = $query->getArrayResult();
        $dql1 = "Select count(t.idtipodoc) as cant from configuracion\workflow\ConfDoc t";
        $query1 = $_em->createQuery($dql1);
        $cant = $query1->getArrayResult();
        $result = array('data' => $resul, 'cant' => $cant[0]['cant']);
         return $result;
}
```

Variaciones en las funcionalidades [17]**:**

- ✓ Utilización de funciones agregadas: DQL es compatible con la mayoría de expresiones adicionales que se emplean en SQL.
- ✓ Manejo de los parámetros: Las instrucciones en DQL utilizan parámetros posicionales, los parámetros posicionales se especifican con números, por ejemplo”?1”, ”?2”, estos son asignados con la función setParameter (1, $var) o setParameters (array ($parameters)).

- ✓ Primer elemento y máximos resultados: En la versión de Doctrine 2.0.6 se utiliza **setMaxResults** ($limit) y **setFirstResult** ($start). En su versión anterior se utilizan las funciones **offset** ($start) y **limit** ($limit)
- ✓ Formato del resultado: El formato del resultado de una consulta depende del modo de hidratación. Cada modo tiene su propio método de hidratación dedicado.
  - **getResult ():** Recupera una colección de objetos. El resultado es una colección de objetos simple (pura) o una matriz, donde los objetos están anidados en las filas del resultado (mixtos).
  - **getSingleResult ():** Recupera un único objeto. Si el resultado contiene más de un objeto, lanza una excepción. La distinción puro/mixto no aplica.
  - **getArrayResult ():** Recupera una matriz gráfica (una matriz anidada).
  - **getScalarResult ()*:*** Recupera un resultado plano/rectangular del conjunto de valores escalares que puede contener datos duplicados. La distinción puro/mixto no aplica.

2.2.6 Funcionalidades en las clases Model

Las clases model son las encargadas de manejar el negocio de la capa de acceso a datos. Con la nueva estructura de carpeta en Doctrine 2.0.6, se mantienen en el mismo directorio. Estas tienen una nomenclatura con el nombre de la tabla y la palabra Model al final, extienden todas de la clase ZendExt_Model y su estructura es la siguiente:

```
class ConfDocModel extends ZendExt_Model{
public function ConfDocModel (){
        parent :: ZendExt_Model ();
}
}
```

Doctrine 1.2.2:

Contiene métodos mágicos predefinidos ejemplo **delete()** y **save()**

```
public function Insertar(ConfDoc $ConfDoc) {
        $ConfDoc->save();
        return true;
}
```

Doctrine 2.0.6

En Doctrine 2.0.6 a la hora de hacer la llamada a las funcionalidades de las clases Repository, se hace una instancia de la clase TransactionManager, haciendo un llamado al método getConnection para obtener la instancia del EntityManager. Una vez hecho esto, se permite acceder al Repository de determinada entidad y utilizando la funcionalidad getRepository pasándole como parámetro el namespace de la clase que se quiere obtener el repositorio y finalmente se solicita la funcionalidad deseada.

```
public function Insertar(configuracion\workflow\models\Entity\ConfDoc $ConfDoc) {
        $mg = ZendExt_Aspect_TransactionManager::getInstance();
        $_em = $mg->getConnection("configuracion/workflow");
        $_em->persist($ConfDoc);
        $_em->flush();
        return true;
}
```

2.2.7 Redefinir Capa de Negocio

La capa de negocio no se redefine en su totalidad, pero los cambios que se realizaron fueron necesarios para la migración. Las funcionalidades se re-implementaron, adaptándolas a la nueva estructura de la capa de acceso a datos y la creación o definición de directorios y ficheros. Se analizaron todos los servicios de los componentes FT y C. Además fueron eliminadas las malas prácticas que se empleaban, por ejemplo:

- Llamadas directamente a las funcionalidades sin la utilización de las clases Model.
- Implementación de consultas y funcionalidades dentro de los ficheros Model encargados del negocio.
- Acceso desde subsistemas directamente a las funcionalidades de otro, sin el empleo de los servicios que se encargan precisamente de establecer una comunicación entre varios subsistemas.

Debido a que todo el negocio de la capa de acceso a datos se debe encontrar dentro de las clases Model, es necesario redefinir las funcionalidades. Las funcionalidades en Doctrine 1.2.2, se almacenaban dentro de la carpeta domain. Con la nueva versión desaparece esta carpeta. Con la versión más actual se

realizan las llamadas de las funcionalidades estáticas o no, que se encuentran en las clases Model que son las que se encargan de realizar el acceso a las funcionalidades de los Repository. Como parte de utilizar una buena práctica:

```
public function ActualizarPuntualizacion($idestado, $documento, $idusuariosession, $valor) {
        $mg = ZendExt_Aspect_TransactionManager::getInstance();
        $_em = $mg->getConnection("configuracion/workflow");
        $result = $_em->getRepository('configuracion\workflow\models\Entity\ConfDocDatEstado')
        ->ActualizarPuntualizacion($idestado, $documento, $idusuariosession, $valor, $_em);
return $result;
}
```

Anteriormente se mencionaron los cambios realizados a los ficheros de configuración. Después de realizadas las acciones hasta este punto, es necesario realizar la actualización en el repositorio de la aplicación. Además, se deben aplicar los cambios a los elementos de configuración, realizar una revisión de las actualizaciones e integrarlos. También se publican los elementos actualizados para su posterior uso en el Sistema de Planificación de Actividades.

2.3 Etapa de pruebas

En esta etapa se realizan pruebas a todas las funcionalidades migradas de los componentes FT y C para garantizar que sean operativas y que los cambios realizados sobre estas no afectan su comportamiento. Por otra parte, se garantiza que la aplicación cumple con las características planteadas por los clientes al someter la aplicación a una revisión técnica, para ello se siguen los siguientes pasos

- **Preparar el entorno de prueba:** en esta actividad se debe preparar el sistema para las pruebas planificadas. De suma importancia la creación de casos de prueba en la dependencia del sistema.

- **Realizar pruebas de rendimiento:** en este paso se utiliza la herramienta Apache JMeter para las pruebas y validar el objetivo de la investigación.

- **Resolver las no conformidades:** como último paso se procede a resolver 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 para resolver las no conformidades en un tiempo record.

2.3.1 Errores más comunes

En las pruebas realizadas los errores más comunes fueron la manera de devolver los resultados y la forma en que se capturan las excepciones, las cuales son diferentes en las dos versiones. Esto afecta el manejo de los datos recibidos en las vistas a la hora de mostrarlos; mientras que el manejo de las excepciones contribuye a que no se gestionen bien los mensajes de notificación al realizar operaciones que no se puedan ejecutar.

2.4 Conclusiones parciales

En este capítulo se realizó un estudio de la arquitectura de los componentes FT y C y de su estructura de carpetas, lo que permitió entender el conjunto de modificaciones realizadas sobre la nueva CAD para interactuar con el ORM Doctrine 2.0.6.

Con la migración de la CAD de los componentes FT y C, en la cual se estableció una nueva estructura de carpetas y se redefinió tanto la CAD como la Capa de Negocio, se eliminó el uso de malas prácticas y se logró una mejor organización en el código y en los ficheros generados del mapeo.

Fueron eliminadas las malas prácticas que se empleaban, por ejemplo:

- Llamadas directamente a las funcionalidades sin la utilización de las clases Model.
- Implementación de consultas y funcionalidades dentro de los ficheros Model encargados del negocio.
- Acceso desde subsistemas directamente a las funcionalidades de otro, sin el empleo de los servicios que se encargan precisamente de establecer una comunicación entre varios subsistemas.

Con el cumplimiento de las tareas definidas para la realización del proceso de migración a Doctrine 2.0.6, se logró la implementación de la migración de la CAD de los componentes FT y C, lo que permitirá un aumento en el rendimiento de la aplicación.

CAPÍTULO 3: VALIDACIÓN DE LA MIGRACIÓN

En este capítulo se valida la migración realizada mediante el uso de la herramienta JMeter que permite evidenciar el rendimiento de la aplicación. Se utiliza un entorno de trabajo en igualdad de condiciones para realizar una comparación del sistema tanto con la versión anterior como con la nueva. Se realizan pruebas de rendimiento que demuestren si el sistema cumple con diferentes criterios y permitan conocer las estadísticas de los procesos que se ejecutan durante su utilización.

3.1 Pruebas de rendimiento

Como parte del cumplimiento del procedimiento de la migración se realizan pruebas de rendimiento. Las cuales se centran en determinar la velocidad con la que el sistema, bajo pruebas, realiza una tarea en condiciones particulares del escenario de pruebas.

Las pruebas de rendimiento tienen que diseñarse para asegurar que el sistema pueda procesar su carga esperada. Esto normalmente implica planificar una serie de pruebas en las que el sistema se hace inaceptable. Estas pruebas implican estresar el sistema realizando demandas que están fuera de los límites del diseño del software [28].

Las pruebas de rendimiento se deben realizar en un momento y entorno determinado. Una buena estrategia es integrar las pruebas de rendimiento dentro del desarrollo del sistema. La realización de pruebas unitarias (pruebas de rendimiento de los componentes desarrollados) [29].

Entre las pruebas de rendimiento se encuentran varios tipos de pruebas, cada una con un objetivo diferente los cuales son descritos a continuación:

- **Pruebas de carga:** Es la más simple de las pruebas de rendimiento y se llevan a cabo para comprender el comportamiento del sistema bajo una carga específica esperada. Estas pruebas le dan a los tiempos de respuesta de todas las transacciones importantes criterios de negocio.
- **Pruebas de capacidad:** Su objetivo es encontrar los límites de funcionamiento del sistema y detectar el cuello de botella o elemento limitante para poder actuar en caso de ampliación del servicio.

- **Pruebas de estrés:** Se utilizan normalmente para comprender los límites superiores de la capacidad dentro del sistema.
- **Pruebas de estabilidad:** Comprueban que no existe degradación del servicio por un uso prolongado del sistema.
- **Pruebas de aislamiento:** Provocan concurrencia sobre componentes aislados del sistema para tratar de detectar posibles errores en ellos.
- **Pruebas de regresión de rendimiento:** Su objetivo es comprobar si se mantienen los niveles de rendimiento tras un cambio en el sistema, comparando el nivel de rendimiento con el que ofrecía con anterioridad [29].

El empleo de este tipo de pruebas sirve para diferentes propósitos tales como demostrar que el sistema cumple los criterios de rendimiento, comparar dos sistemas para encontrar cuál de ellos funciona mejor o medir qué partes del sistema o de carga de trabajo provocan que el conjunto rinda mal. Para su diagnóstico se utilizan herramientas que pueden ser monitorizaciones que midan qué partes de un software contribuyen más al mal rendimiento o para establecer niveles que mantenga un tiempo de respuesta aceptable [30].

La herramienta a utilizar en este caso es JMeter, que es la propuesta en el procedimiento utilizado en esta investigación. La aplicación de escritorio Apache JMeter es un software de código abierto, diseñado para probar el comportamiento funcional y medir el rendimiento. Es una herramienta que puede ser utilizada para probar el rendimiento tanto de recursos dinámicos como estáticos (archivos, lenguajes dinámicos Web, bases de datos, consultas, servidores y más). Se puede utilizar para realizar un análisis gráfico de rendimiento o para probar el comportamiento del servidor bajo cargas concurrentes [31].

3.1.1 Entorno de Prueba

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

Hardware: Intel Core i3-2348 a 3.3 GHzx4, 4 GB de RAM, sistema operativo Ubuntu 14.04, tipo de sistema 64-bit en una partición de 30 GB.
Software: Gestor de Base de datos: PostgreSQL v9.3, 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 instala un MT que utiliza la versión 1.2.2 del ORM Doctrine y otro que funciona con las bibliotecas de Doctrine 2.0.6. Por otra parte, se confeccionan los casos de prueba de carga con el objetivo de realizarle las pruebas a las funcionalidades que se emplean en el plan de prueba de la herramienta JMeter **(Anexos 1-5)**.

3.1.2 Realizar pruebas de rendimiento

Como se menciona anteriormente, utilizando una herramienta para la automatización de este proceso se logra disminuir los costos en recursos y tiempo destinados a su realización. Para la realización de las pruebas se utiliza la herramienta Apache JMeter específicamente para realizar pruebas de carga y estrés. JMeter como herramienta de prueba dispone de varios componentes que facilitan la elaboración de los escenarios de prueba con la ventaja de simular para cada uno de esos escenarios miles de usuarios [32].

Con la previa elaboración de los Casos de Pruebas de Carga se procede a la realización de las pruebas con la herramienta JMeter utilizando para este trabajo la versión 2.8.1. 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 6**, se inicia la cabecera de la prueba en la que se define la cantidad de usuarios que se modelan definidos en los Casos de Pruebas.

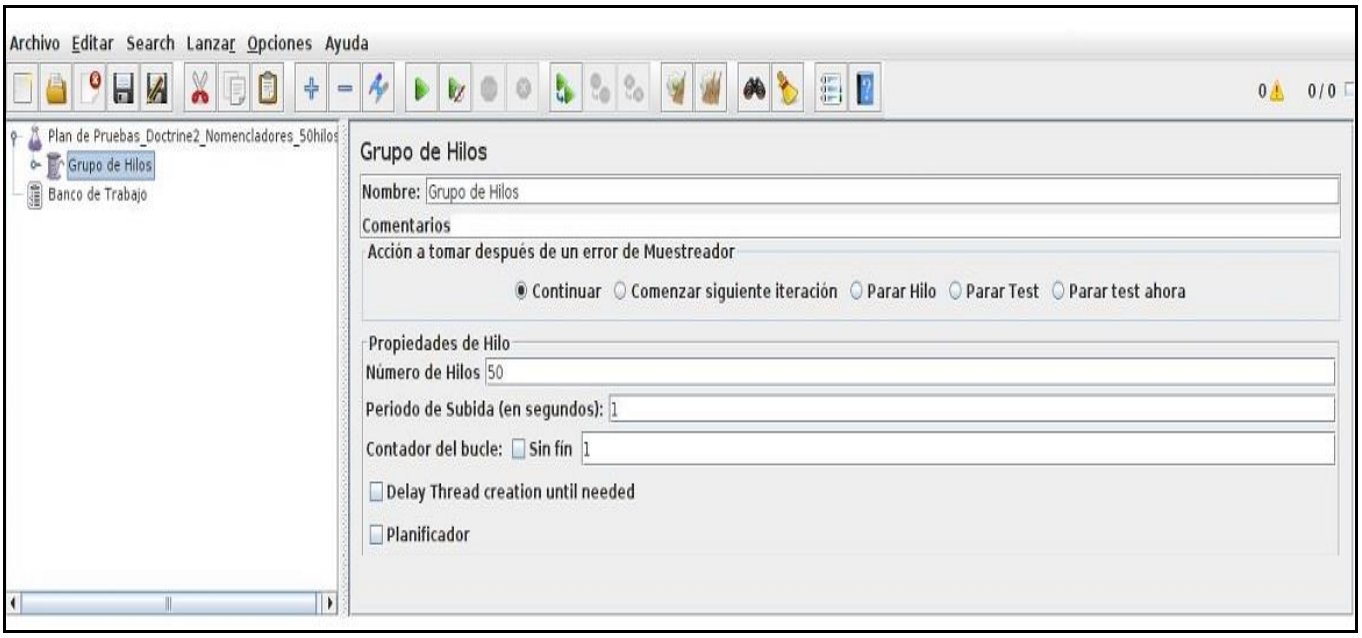


Figura 6. Elemento Grupo de Hilos.

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

- **Número de hilos:** número de usuarios a simular (100, 50 y 25 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 7. Selección del Servidor Proxy.

Este elemento permite al JMeter grabar todos los pasos realizados 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.**

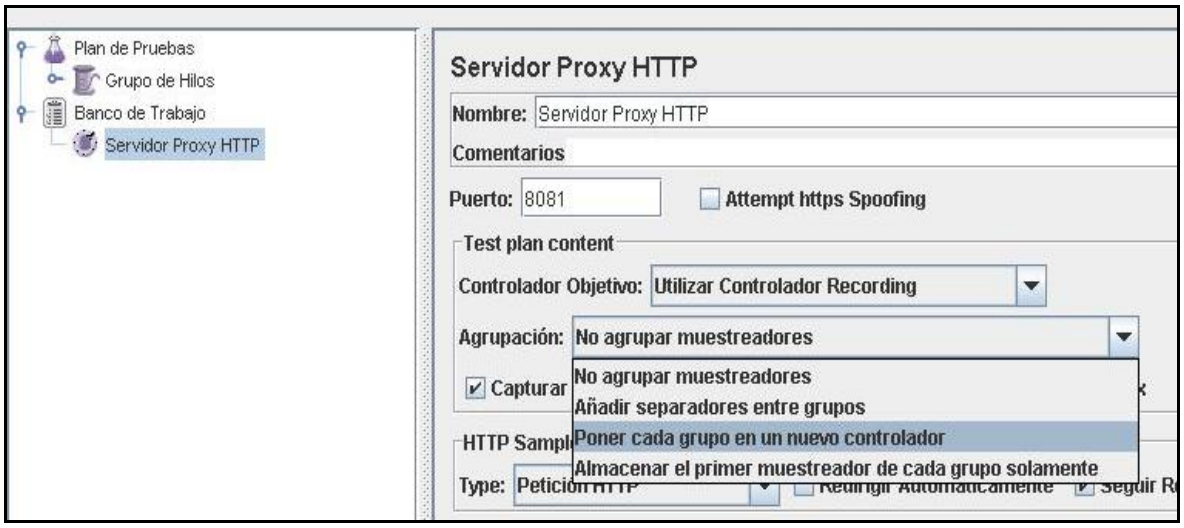


Figura 8. Valores para el Servidor Proxy HTTP.

Posteriormente en el navegador, en herramientas >>opciones >>avanzado >>red >>configuración, se le especifica la dirección y el puerto que se utiliza para la grabación de los escenarios. De esta manera se está indicando que el proxy que utilizará el navegador será el definido en el JMeter. Luego se ejecuta la aplicación y cada paso realizado en la Web será registrado en el JMeter de manera organizada.

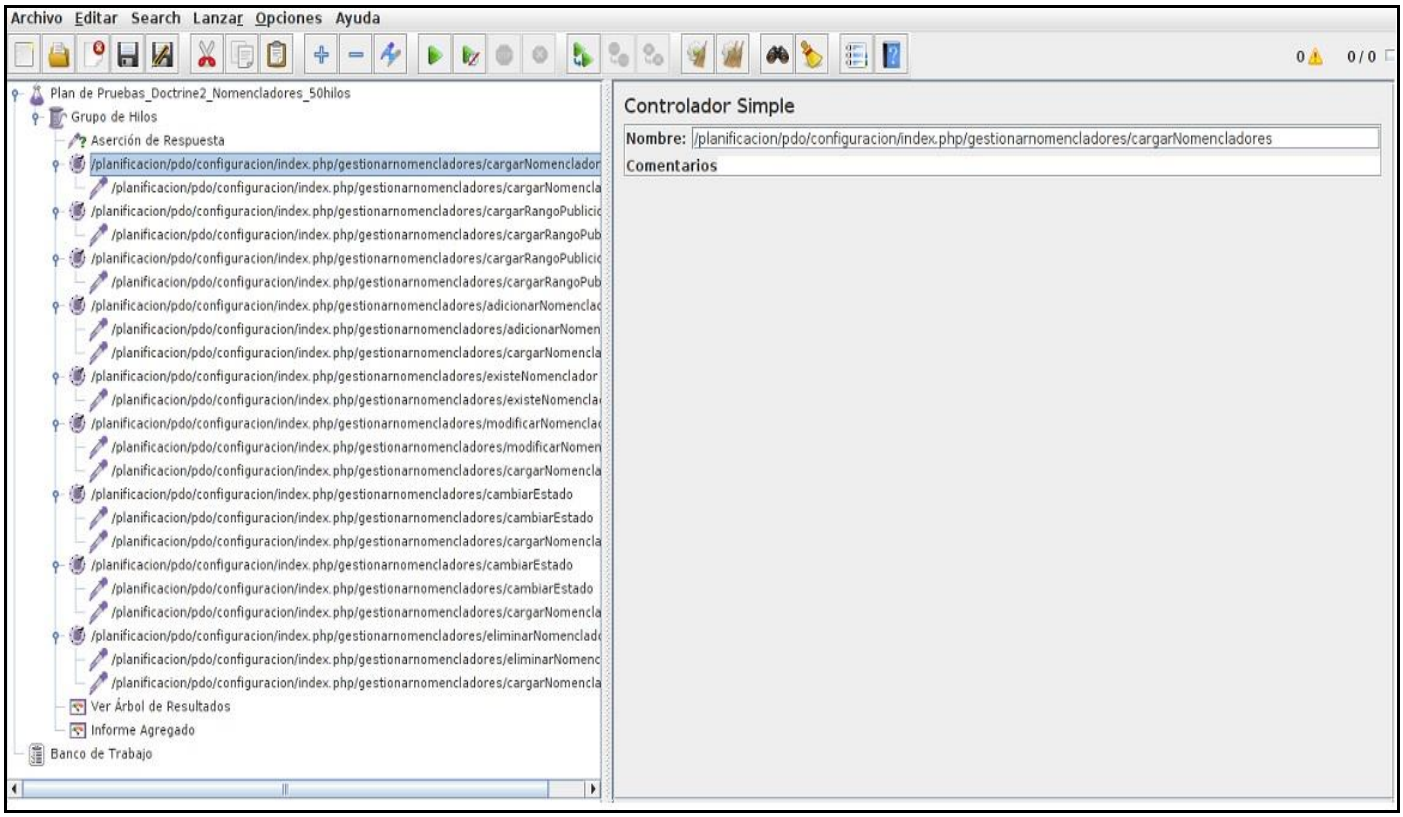


Figura 9. Grabación de Navegación de la Web.

Como se puede apreciar en la imagen anterior las peticiones http están grabadas dentro de controladores diferentes y cada una de ellas contiene toda su información. Por cada petición http se genera un gestor de cabecera http, los cuales se eliminan porque interfieren en el éxito de las respuestas del servidor Web [32]; además los datos que han sido insertados en el sistema se muestran en los parámetros enviados con la petición.

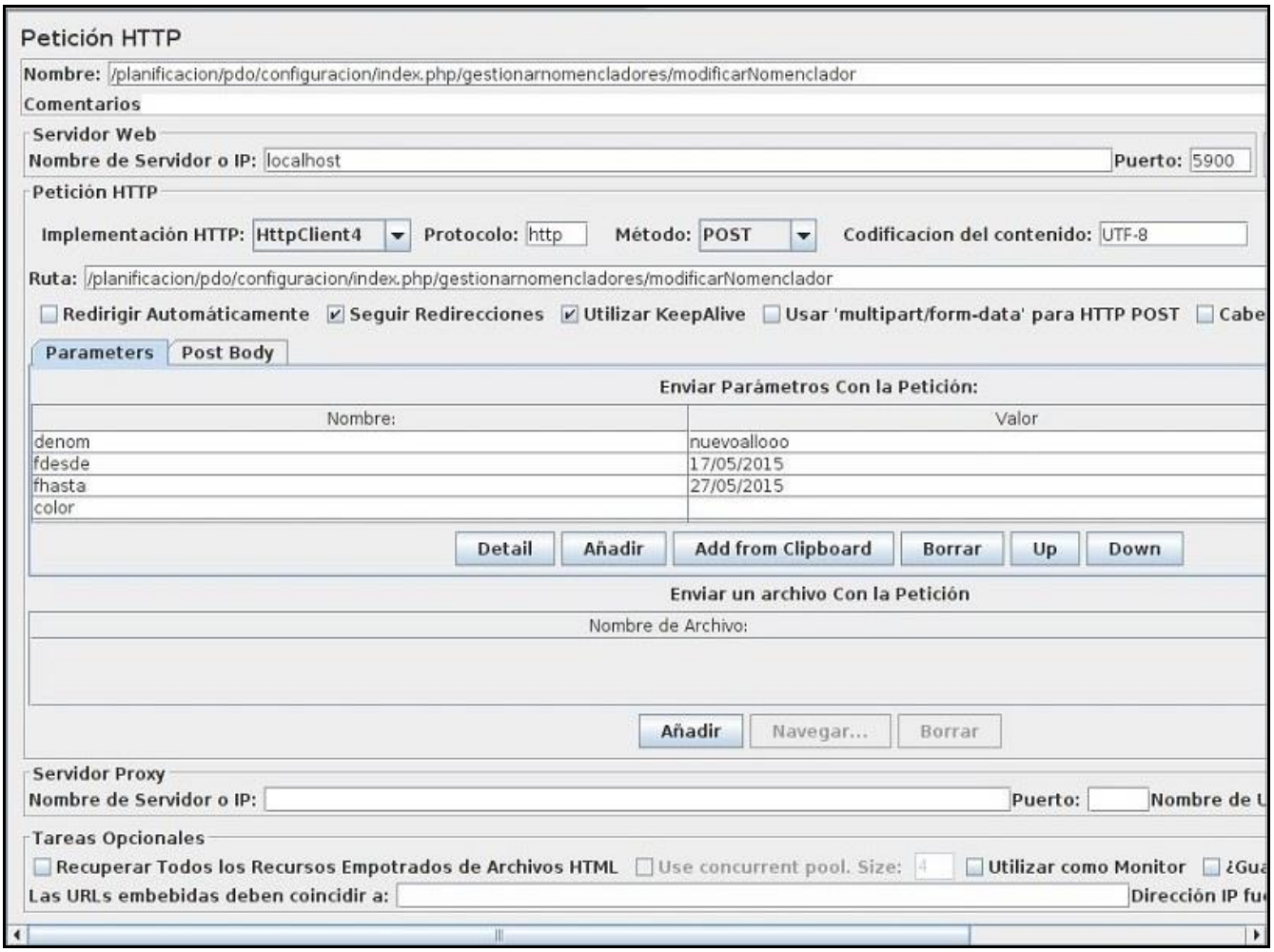


Figura 10. Ejemplo de petición HTTP de la grabación.

Una vez grabadas todas las peticiones http, es necesario agregar otros elementos en los cuales se especificarán los parámetros que se quieren comprobar en la prueba. Esto puede realizarse apoyándose en elementos como la Aserción de Respuesta.

Figura 11. Selección de Aserción de Respuestas.

En el caso de este elemento, se necesita conocer que la página a la cual se quiere acceder se ha cargado de manera satisfactoria por lo que se especifica el código 200 para una página cargada satisfactoriamente.

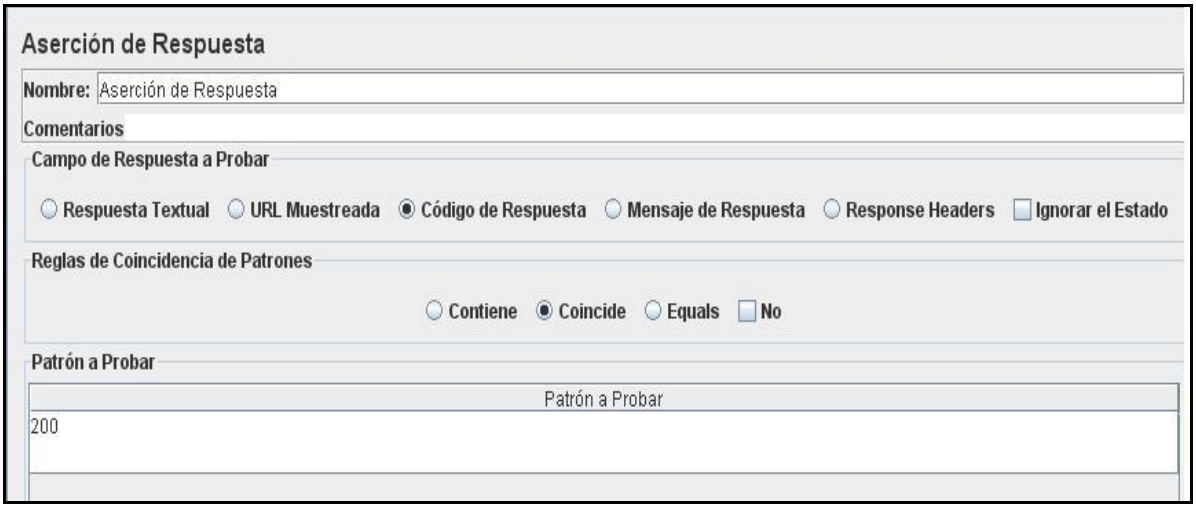


Figura 12. Datos de la Aserción de Respuesta.

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

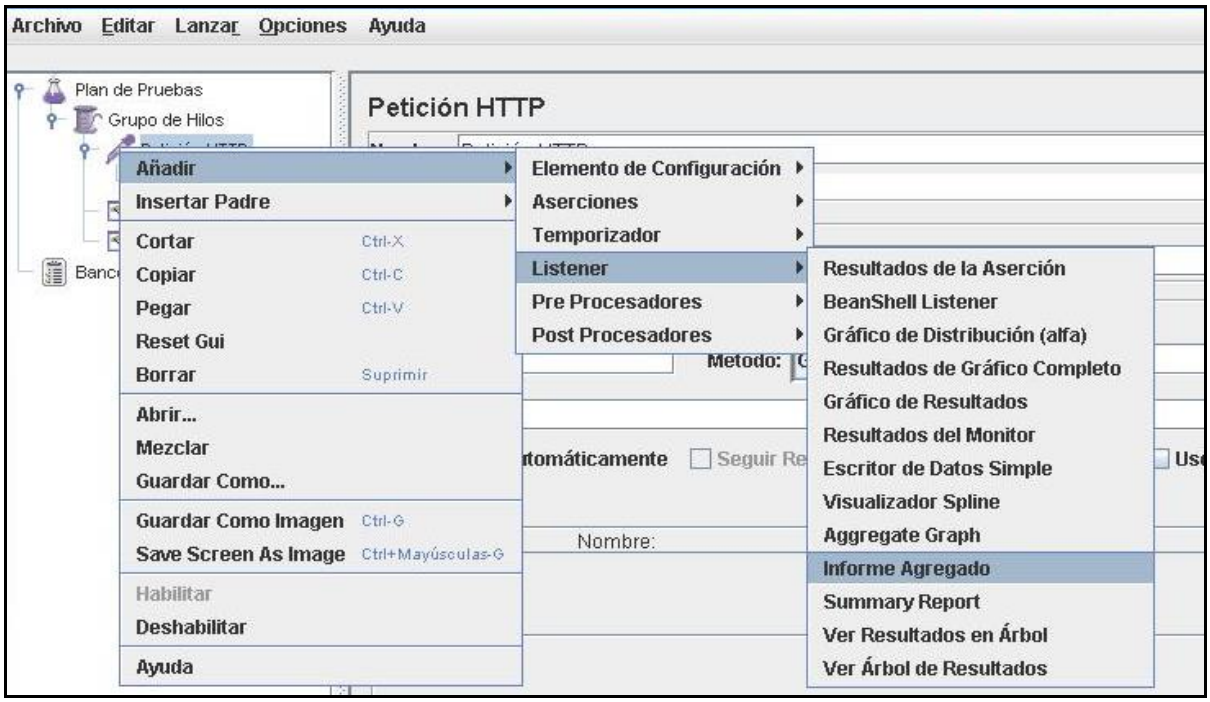


Figura 13. Selección de Informe Agregado.

En estos informes de cada una de las pruebas se recogen los siguientes indicadores:

- Número Muestras: cantidad de muestras de peticiones utilizadas para 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.

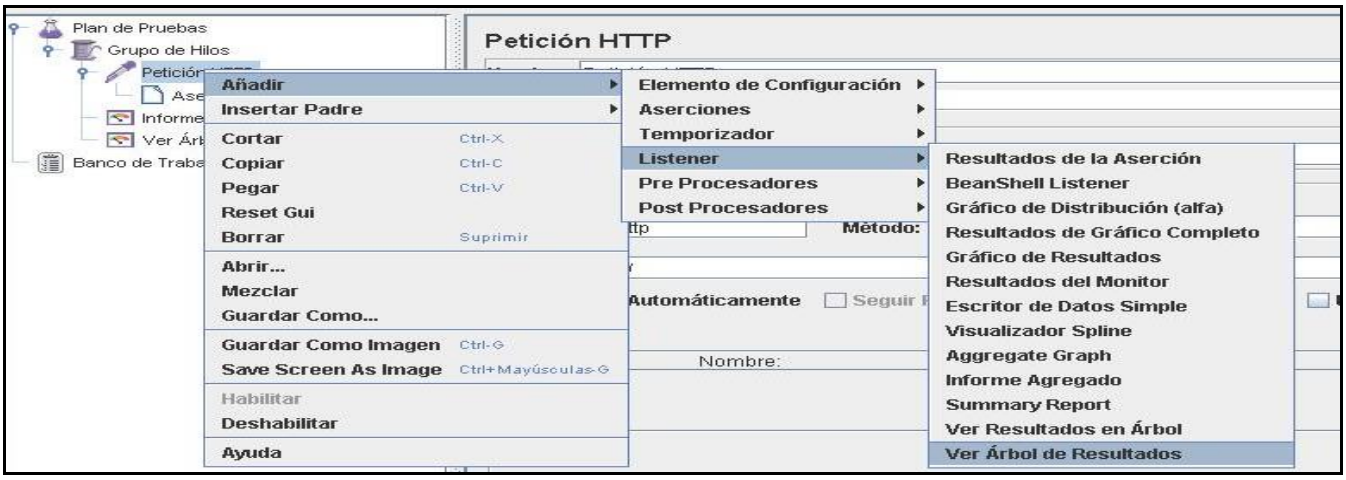


Figura 14. Selección de Ver Árbol de Resultados.

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

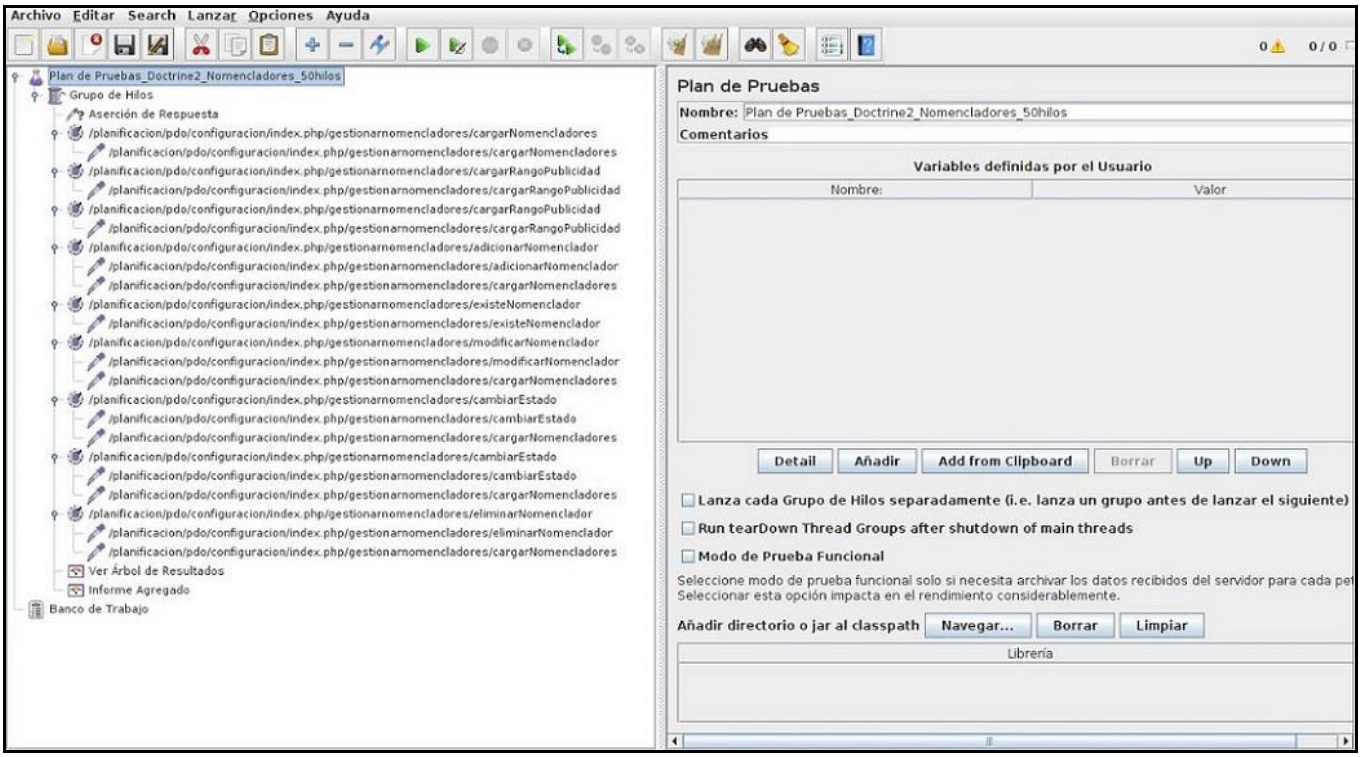


Figura 15. Confección de un Plan de Pruebas.

En la realización de las pruebas de carga y estrés a los componentes FT y C se crearon cinco planes de prueba correspondientes a los cinco Casos de Prueba de Carga: "*Gestionar Estados de la Información*", *"Gestionar Transición de la Información"*, "*Gestionar Grupo de Usuarios por Rol* ", *"Gestionar Permisos*" y "*Gestionar Nomencladores*" **(Anexos 6-10)**.

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 un ejemplo de uno de ellos:

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...	100	2531	2618	4470	29	4749	0,00%	2,3/sec	24,9
/configuraci...	500	158	133	268	35	536	0,00%	10,8/sec	114,0
/configuraci...	200	172	152	293	40	542	0,00%	4,4/sec	46,5
/configuraci...	200	164	155	269	44	496	0,00%	4,4/sec	46,8
/configuraci...	200	160	147	262	38	490	0,00%	4,4/sec	46,6
/configuraci...	200	154	137	252	43	453	0,00%	4,3/sec	46,0
/configuraci...	100	178	156	336	41	473	0,00%	2,3/sec	24,0
/configuraci...	100	186	164	311	47	610	0,00%	2,2/sec	23,8
/configuraci...	100	175	166	275	55	411	0,00%	2,2/sec	23,6
/configuraci...	100	155	137	250	41	432	0,00%	2,2/sec	23,6
Total	1800	295	150	336	29	4749	0,00%	38,7/sec	410,0

Figura 16. Informe Agregado.

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

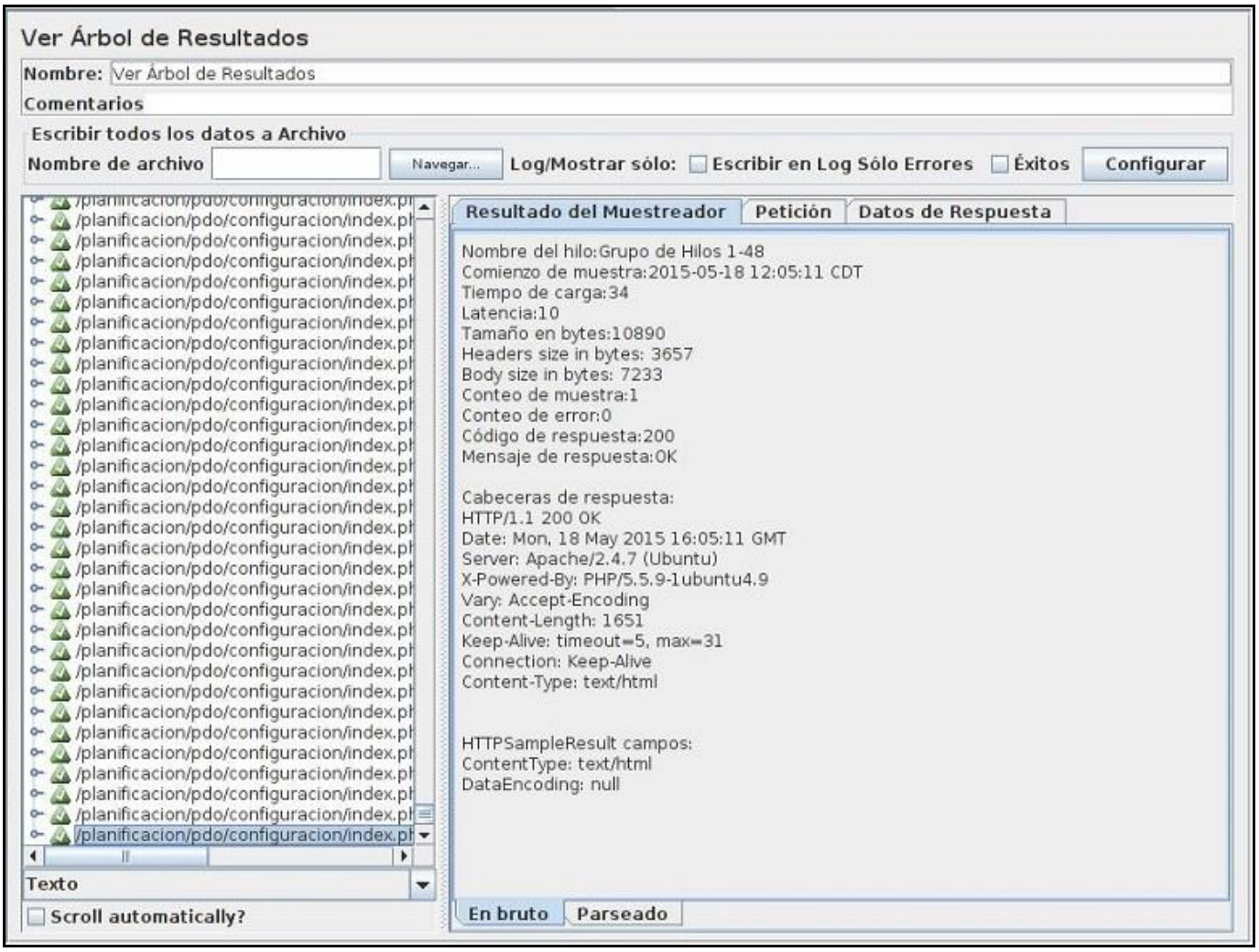


Figura 17. Ver Árbol de Resultados.

3.2 Resultado de las pruebas de rendimiento

Las pruebas realizadas consistieron en definir tres estados de 100, 50 y 25 hilos para cada caso de prueba de carga, los cuales simulan 100, 50 y 25 accesos de usuarios respectivamente; simulando esa cantidad de usuarios conectados concurrentemente al sistema haciendo peticiones al servidor, siendo estos los escenarios en los cuales se encontraría el sistema en un momento determinado.

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

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

Mientras que para calcular el Tiempo promedio [33] de cada usuario que estuvo accediendo al sistema se utilizó la siguiente fórmula:

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

Para realizar las pruebas se configuraron 50 hilos cada 1 segundo, siendo este el valor intermedio entre los tres estados definidos anteriormente. Buscando una mejor visualización de los resultados se procede a evaluar cada componente haciendo uso de las dos versiones del ORM. Los valores obtenidos por el **Informe Agregado,** correspondientes a cada componente se muestran a continuación.

Componente Flujo de Trabajo>>Estados de la Información

Doctrine 1.2.2

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	1950	776	272	2149	17	14101	0,00%	24,8/sec	13,0

Como se puede apreciar el tiempo promedio es 0.776 seg al realizar 1950 peticiones al servidor. El tiempo total utilizado para los 50 hilos se puede calcular con la siguiente fórmula:

Tiempo total = #Muestras * Media = 1950 * 776 = 1513200 milisegundos.

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

((Tiempo Total/1000)/60)/cantidad de hilos = ((1513200 /1000)/60)/50 = 0.5044 minutos.

Doctrine 2.0.6

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	1950	351	153	355	33	5722	0,00%	23,4/sec	247,7

El tiempo promedio es de 0.351 seg al realizar 1950 requerimientos al servidor.

Tiempo total = 1950 * 351 = 684450 milisegundos.

Tiempo promedio total = ((684450/1000)/60)/50 = 0.2281 minutos.

Componente Flujo de Trabajo>>Transiciones de la Información

Doctrine 1.2.2

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	2700	1095	77	1080	17	22584	0,00%	30,1/sec	15,7

El tiempo promedio es de 1.095 seg al realizar 2700 requerimientos al servidor.

Tiempo total = 2700 * 1095 = 2956500 milisegundos.

Tiempo promedio total = ((2956500/1000)/60)/50 = 0.9855 minutos.

Doctrine 2.0.6

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	2700	289	204	370	28	4161	0,00%	30,4/sec	321,9

El tiempo promedio es de 0.289 seg al realizar 2700 requerimientos al servidor.

Tiempo total = 2700 * 289 = 780300 milisegundos

Tiempo promedio total = ((780300/1000)/60)/50 = 0.2601 minutos.

Componente Configuración>>Grupo de Usuario por Rol

Doctrine 1.2.2

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	1900	1565	84	1009	17	36642	0,00%	18,9/sec	9,9

El tiempo promedio es de 1.565 seg al realizar 1900 requerimientos al servidor.

Tiempo total = 1900 * 1565 = 2973500 milisegundos.

Tiempo promedio total = ((2973500/1000)/60)/50 = 0.9911 minutos.

Doctrine 2.0.6

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	1900	278	230	397	1	4338	0,00%	44,1/sec	444,9

El tiempo promedio es de 0.278 seg al realizar 1900 requerimientos al servidor.

Tiempo total = 1900 * 278 = 528200 milisegundos.

Tiempo promedio total = ((528200/1000)/60)/50 = 0.1760 minutos.

Componente Configuración>>Permisos

Doctrine 1.2.2

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	1500	1632	188	1730	17	28899	0,00%	12,2/sec	6,4

El tiempo promedio es de 1.632 seg al realizar 1500 requerimientos al servidor.

Tiempo total = 1500 * 1632 = 2448000 milisegundos.

Tiempo promedio total = ((684450/1000)/60)/50 = 0.816 minutos.

Doctrine 2.0.6

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	1500	270	172	418	27	2917	0,00%	25,3/sec	271,2

El tiempo promedio es de 0.270 seg al realizar 1500 requerimientos al servidor.

Tiempo total = 1500 * 270 = 405000 milisegundos.

Tiempo promedio total = ((405000/1000)/60)/50 = 0.135 minutos.

Componente Configuración>>Nomencladores

Doctrine 1.2.2

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	700	1892	193	1900	17	32358	0,00%	14,7/sec	7,7

El tiempo promedio es de 1.892 seg al realizar 700 requerimientos al servidor.

Tiempo total = 700 * 1892 = 1324400 milisegundos.

Tiempo promedio total = ((1324400/1000)/60)/50 = 0.4414 minutos.

Doctrine 2.0.6

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Total	700	237	112	237	28	3819	0,00%	111,7/sec	1183,6

El tiempo promedio es de 0.237 seg al realizar 700 requerimientos al servidor.

Tiempo total = 700 * 237 = 165900 milisegundos.

Tiempo promedio total = ((165900/1000)/60)/50 = 0.0553 minutos.

Análisis de resultados

Tabla 2. Resultados generales.

	100		50		25	
	Doctrine 1	Doctrine 2	Doctrine 1	Doctrine 2	Doctrine 1	Doctrine 2
Número Muestras	9472	7400	8750	8750	2175	3950
Media	13631	3363	6967	1425	5513	754
Mediana	1725	3470	1426	1721	1014	917
Línea de 90%	8875	4313	9784	2720	5997	1013
Min	192	0	190	1	115	17
Max	136754	11237	73698	6042	39658	3214
Porcentaje de Error	0.00	0.00	0.00	0.00	0.00	0.00
Tiempo total (ms)	24474800	4957700	11215600	2563850	2553900	586400
Tiempo promedio (min)	4,079	0,8262	3,7384	0 ,8545	1,7025	0,3908

De los resultados anteriores se puede concluir que el tiempo total utilizado para la cantidad de hilos definida, cambia notablemente en la versión 2.0.6 del ORM Doctrine respecto a la anterior. Realizando una comparación se tiene como tiempo general de la prueba: 38244300 milisegundos para la versión 1.2.2 y 31177950 milisegundos para la 2.0.6 Estos resultados demuestran una disminución de 7066350 milisegundos en la ejecución del plan de pruebas en cuestión, lo que representa una reducción del 81.52% de este indicador.

Por su parte, en el tiempo promedio requerido para cada hilo se puede apreciar que hubo una mejora de la versión 2.0.6 respecto a la 1.2.2. El tiempo promedio general que se empleó para cada hilo fue de 9.5199 min en la versión 1.2.2 y 2.0715 min en la 2.0.6. Estos resultados arrojaron una reducción de 7.4484 min para la interacción con el sistema por las cantidades de usuarios simulados en cuanto a la utilización de ambas versiones, esto representa una disminución del 21.75% de este indicador.

3.3 Resolver no conformidades

A partir de los resultados obtenidos con los elementos de la herramienta JMeter: Informe Agregado y Ver Árbol de Resultados**,** se puede observar cuál petición no se ejecutó correctamente. En el caso del Informe Agregado este indica en la columna Porcentaje 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, una acción a destacar es que no se reportaron incidencias de errores mientras la ejecución de las peticiones. Esto se garantiza debido a que durante la etapa de implementación se chequeó continuamente que las funcionalidades migradas tuvieran éxito. Por su parte, se apreciaron en las vistas de la aplicación la falta de información; debido a que los resultados de las consultas de la versión 2.0.6 de Doctrine no es igual a la anterior; para ello basta con cambiar la manera de solicitarle los datos al controlador en la vista.

3.4 Conclusiones parciales

El estudio de los aspectos fundamentales para la realización de las pruebas de rendimiento que se le ejecutan a los sistemas, permitió comprender este proceso para aplicárselo a los componentes FT y C.
La validación de la migración definida, empleando la herramienta JMeter arrojó como resultado que:

- El tiempo total y tiempo promedio para las pruebas realizadas con 50 hilos disminuyó un 22.85% en la ejecución del plan de pruebas.
- El tiempo total y tiempo promedio para las pruebas realizadas con 100 hilos disminuyó un 34.70% en la ejecución del plan de pruebas.
- El tiempo total y tiempo promedio para las pruebas realizadas con 25 hilos disminuyó un 22.96% en la ejecución del plan de pruebas.

Además se implementaron las pruebas para comprobar cada uno de los problemas detectados en la anterior CAD. Se realizaron pruebas a los componentes FT y C, además de las funcionalidades implementadas.

CONCLUSIONES GENERALES

El análisis realizado sobre algunos ORM utilizados para la persistencia de datos, demostró que estos exhiben características comunes y que en Doctrine 2.0.6 se encuentran implementadas las principales potencialidades de los otros ORM. Lo antes mencionado, demuestra que las buenas prácticas de los demás ORM son reutilizables en la versión 2.0.6 de Doctrine al brindar este un alto rendimiento.

Con la actualización de la CAD de los componentes FT y C, en la cual se estableció una nueva estructura de carpetas y se redefinió tanto la CAD como la Capa de Negocio, se eliminó el uso de malas prácticas y se logró una mejor organización en el código y en los ficheros generados del mapeo, en cuanto a tipo y funcionalidad.

Las pruebas realizadas por la comisión de especialistas del proyecto sobre cada una de las funcionalidades de los componentes FT y C, permitieron demostrar que cumplen con las expectativas del cliente.

La evaluación realizada a la migración de la CAD de los componentes FT y C, mediante las pruebas de rendimiento sobre las funcionalidades, arrojó como resultados la reducción del tiempo total y del tiempo promedio utilizado para una cantidad específica de usuarios en un 81.52% y un 21.75% respectivamente, lo que representa un aumento del rendimiento de los componentes FT y C. Se demuestra así que se le ha dado cumplimiento al objetivo general planteado y a cada uno de los objetivos específicos que se definieron.

RECOMENDACIONES

El objetivo general de esta investigación fue alcanzado, pero durante el desarrollo de su migración, han surgido ideas que sería recomendable tener en cuenta para su futuro perfeccionamiento:

- Continuar la migración de los restantes componentes del sistema para actualizar completamente la capa de acceso a datos haciendo uso de Doctrine 2.0.6.
- Realizar una investigación para facilitar el mapeo.

REFERENCIAS BIBLIOGRÁFICAS

1. Minetto, E.L., *Frameworks para Desenvolvimiento en PHP*, ed. L.M.F.S. Paulo. 2007.
2. Bugarin, J.L. *Slideshare*. 07 de Julio de 2013 [cited 2014 28 de noviembre]; Available from: http://www.slideshare.net/jlbugarin/persistencia-de-datoshibernatearquitecturasdesoftware#btnNext.
3. Madeja, *Marco de Desarrollo de la Junta de Andalucía*, in *Comparación de las tecnologías de acceso a datos*.
4. I.V. Osorio, L.C.E.y.R.R.B.C., *Procedimineto para actualización de la Capa de Acceso a Datos del Marco de Trabajo Sauxe de Doctrine 1.2.2 a Doctrine 2.0*. La Habana : s.n., 2013, Universidad de las Ciencias Infomáticas.
5. McCafferty, B. *Best Practices with ASP.NET*. 1.2nd Ed. 2008 [cited 2014 11 de noviembre]; Available from: http://www.codeproject.com/Articles/13390/NHibernate-Best-Practices-with-ASP-NET-1-2nd-Ed#SUMMARY.
6. González, R.A.H.L.y.S.C., *El proceso de investigación científica*. 2011. ISBN 978-959-16-1307-3.
7. Leroy. *MyCyberAcademy*. Un poco sobre ORM y Frameworks 14 de Agosto de 2013 [cited 2014 29 de noviembre]; Available from: http://mycyberacademy.com/articulo/Un-Poco-Sobre-ORM-y-Frameworks.
8. Bisbé, R.L. *¿Qué es un ORM y por qué nos interesa?* 26 de Noviembre de 2011 [cited 2014 06 de diciembre]; Available from: http://robertoluis.wordpress.com/2011/12/13/que-es-un-orm-y-por-que-nos-interesa/.
9. Community, H. 2012 [cited 2014 26 de noviembre]; Available from: http://docs.jboss.org/hibernate/orm/4.2/devguide/en-US/html/ch02.html#d5e618.
10. Rubén. *CODECTRITICON*. Frameworks de persistencia 18 de Noviembre de 2011 [cited 2014 23 de noviembre]; Available from: http://codecriticon.com/diferencias-frameworks-persistencia-i/.
11. Community, H. 08 de Febrero de 2012 [cited 2015 09 de febrero]; Available from: http://docs.jboss.org/hibernate/orm/3.6/reference/es-ES/html/best-practices.html.
12. Propel, S. *Propel, Smart, easy object persistence*. 2013 [cited 2014 3 de diciembre]; Available from: http://propelorm.org/.
13. Vondrick, C. *Symfony*. ¿Cómo utilizar Propel? 2011; Available from: http://symfony.com/legacy/doc/cookbook/1_2/es/propel_13.

14. Hasheado. *Doctrine vs Propel*. 2010 [cited 2015 09 de febrero]; Available from: http://www.hasheado.com/doctrine-vs-propel.html.
15. García, L. *Scribd, Django ORM*. 03 de Marzo de 2011 [cited 2014 27 de noviembre]; Available from: http://es.scribd.com/doc/63901225/Django-ORM.
16. *Doctrine Migrations*. 2015; Available from: http://desarrolla2.com/post/doctrine-migrations-para-actualizar-la-base-de-datos-desde-el-fichero-schema-yml.
17. Documentation, D.O. *Doctrine 2 ORM Documentation*. Documentación del ORM Doctrine 2 04 de Diciembre de 2012 [cited 2014 04 de diciembre]; Available from: http://docs.doctrine-project.org/en/latest/.
18. Cadavid, A.N. *Sistema de investigación de la Universidad Icesi*. 2008 [cited 2014 25 de noviembre]; Available from: http://www.icesi.edu.co/investigaciones/proyecto.do?id=38.
19. Ben Collins-Sussman, B.W.F.y.C.M.P., *Control de Versiones con Subversion*. 2013.
20. Tecsisa. *Buenas Prácticas de Gestión de Versiones con Subversion*. 06 de Mayo de 2010 [cited 2015 09 de febrero]; Available from: http://blogs.tecsisa.com/articulos-tecnicos/buenas-practicas-de-gestion-de-versiones-con-subversion/.
21. Martínez, R., *Portal de Postgres en Español*. 2013.
22. NetBeans. *NetBeans quick start*. 2013.
23. Foundation, T.A.S. *The Apache Software Foundation*. 2012 [cited 2015 17 de enero]; Available from: http://www.apache.org.
24. Inc, Z.T. *Manual Zend Framework Español*. 01 de Diciembre de 2012 [cited 2015 30 de enero]; Available from: http://manual.zfdes.com/es/introduction.overview.html.
25. Pupo, Y.C., *Libro de Ayuda del Marco de Trabajo Sauxe*. Libro de Ayuda del Marco de Trabajo Sauxe. La Habana : s.n., 2010.
26. México, U.N.A.d., *Administración para la toma de decisiones*, U.N.A.d. México, Editor.
27. Php. *Manual de PHP*. [cited 2015 09 de febrero]; Available from: http://www.php.net/manual/es/index.php.
28. Técnico, Q. *Definiciones ISTQB*. [cited 2015 2 de abril]; Available from: http://qatecnico.blogspot.com/p/definiciones-istqb.html.
29. Técnico, Q. *Pruebas de rendimiento: Tipos y objetivos*. 03 de Marzo de 2012 [cited 2015 2 de abril]; Available from: http://qatecnico.blogspot.com/2012/03/pruebas-de-rendimiento-tipos-y.html.

30. Madeja. *Marco de Desarrollo de la Junta de Andalucía*. Buenas prácticas en el diseño de pruebas de rendimiento [cited 2015 04 de abril]; Available from: http://www.juntadeandalucia.es/servicios/madeja/contenido/libro-pautas/75.

31. JMeterTM, A.J.-A. *Apache JMeter* 2014 [cited 2015 13 de abril]; Available from: http://jmeter.apache.org/.

32. Almenares, L.S., *Cómo Realizar Pruebas de Carga y Estrés en JMetter*. La Habana: s.n., 2008.

33. J. Díaz, C.M.T.B., A. S. Rodríguez y V Soria, *Usando Jmeter para pruebas de rendimiento*. En XIV Congreso Argentino de Ciencias de la Computación. La Plata : s.n., 2008.

ANEXOS

Anexo 1. Casos de Prueba de Carga para “Gestionar Estados de la Información”

ID del escenario	Escenario de la sección	Carga de Trabajo	Descripción
EC1- Acceder a la interfaz “Gestionar estados de la información”.	EC1.1- Acceder a la interfaz “Gestionar estados de la información”. http://localhost:5900/portal/index.php/portal/	100,50,25	Es la interfaz que muestran los documentos, su estado, su autor y se garantiza su gestión.
EC2- Registro de estados de los documentos	EC2.1- Adicionar un documento Para ello se escribe la denominación de un documento y seleccionar la opción Adicionar	100,50,25	Esta interfaz muestra los campos necesarios para adicionar un documento.
	EC2.2- Modificar un documento. Para ello se selecciona un documento y se selecciona la opción Modificar, cambiar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para modificar un documento en la base de datos desde la aplicación.
	EC2.3 Eliminar un documento. Para ello se selecciona un documento y ejecutar el botón Eliminar y Aceptar.	100,50,25	Se muestra una alerta sobre el elemento que se desea eliminar.

Anexo 2. Casos de Prueba de Carga para “Gestionar Transición de la Información”

ID del escenario	Escenario de la sección	Carga de Trabajo	Descripción
EC1 – Acceder a la interfaz “Transición de la Información”.	EC1.1- Acceder a la interfaz “Transición de la Información”. http://localhost:5900/portal/index.php/portal/	100,50,25	Es la interfaz para gestionar las transiciones de la Información que presentan las mismas.

EC2-Registro de transiciones	EC2.1- Insertar transición. Para ello ejecutar click en el botón "Adicionar".	100,50,25	Esta interfaz muestra los campos necesarios para insertar una transición.
	EC2.2- Modificar transición. Para ello seleccionar un elemento ejecutar click en el botón Modificar, cambiar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para modificar una transición en la base de datos desde la aplicación.
	EC2.3 Eliminar transición. Para ello seleccionar una fila y ejecutar el botón Eliminar y después Aceptar. Nota: Sólo se eliminarán las transiciones que se tengan permisos.	100,50,25	Se muestra una notificación sobre la transición que se desea eliminar.

Anexo 3. Casos de Prueba de Carga para "Gestionar Grupo de Usuario por Rol"

ID del escenario	Escenario de la sección	Carga de Trabajo	Descripción
EC1-Acceder a la interfaz "Grupo de Usuario por Rol".	EC1.1- Acceder a la interfaz "Grupo de Usuario por Rol". http://localhost:5900/portal/index.php/portal/	100,50,25	Es la interfaz para gestionar los Grupo de Usuario por Rol que presenta la misma
	EC1.2 – Mostrar los grupo de usuario por rol	100,50,25	Esta interfaz muestra los grupo de usuario por rol
EC2-Gestionar Grupos	EC2.1- Insertar un grupo Para ello ejecutar click en el botón Adicionar llenar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para insertar un grupo a la base de datos desde la aplicación.
	EC2.2- Modificar un grupo. Para ello seleccionar un elemento	100,50,25	Esta interfaz muestra los campos necesarios para

	ejecutar click en el botón Modificar, cambiar los campos y ejecutar click en Aceptar.		modificar un grupo en la base de datos desde la aplicación.
	EC2.3 Eliminar un grupo. Para ello seleccionar una fila y ejecutar el botón Eliminar y Aceptar.	100,50,25	Se muestra una notificación sobre el campo que se desea eliminar.

Anexo 4. Casos de Prueba de Carga para “Gestionar Permisos”

ID del escenario	Escenario de la sección	Carga de Trabajo	Descripción
EC1 – Acceder a la interfaz “Permisos”.	EC1.1- Acceder a la interfaz “Permisos”. http://localhost:5900/portal/index.php/portal/	100,50,25	Es la interfaz para gestionar los permisos así como los campos que presentan los mismos.
EC2- Permisos otorgados por mi	EC2.1- Insertar permisos. Para ello ejecutar click en el botón “Adicionar”.	100,50,25	Esta interfaz muestra los campos necesarios para insertar un permiso.
	EC2.2- Modificar permisos. Para ello seleccionar un elemento ejecutar click en el botón Modificar, cambiar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para modificar un permiso en la base de datos desde la aplicación.
	EC2.3 Eliminar permiso. Para ello seleccionar una fila y ejecutar el botón Eliminar y después Aceptar. Nota: Sólo se eliminarán los usuarios o grupos que se tengan permisos.	100,50,25	Se muestra una notificación sobre el permiso que se desea eliminar.
EC3-Mis Permisos	EC4.1- Mostrar campos. Para ello seleccionar un permiso y	100,50,25	Esta interfaz muestra la información de los campos

	ejecutar “Actualizar”.		que contiene un determinada permiso.

Anexo 5. Casos de Prueba de Carga para “Gestionar Nomenclador”

ID del escenario	Escenario de la sección	Carga de Trabajo	Descripción
EC1-Acceder a la interfaz “Gestionar nomencladores”.	EC1.1- Acceder a la interfaz “Gestionar nomencladores”. http://localhost:5900/portal/index.php/portal/	100,50,25	Es la interfaz para gestionar los nomencladores que presenta la misma
EC2- Categoría del Plan.	EC2.1- Mostrar categoría. Para ello ejecutar click en el indicador “Categoría”.	100,50,25	Esta interfaz muestra información relacionada con las categorías registrados en la aplicación.
	EC2.3- Insertar una categoría. Para ello ejecutar click en el botón Adicionar llenar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para insertar una categoría en la base de datos desde la aplicación.
	EC2.4- Modificar una categoría. Para ello seleccionar un elemento ejecutar click en el botón Modificar, cambiar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para modificar una categoría en la base de datos desde la aplicación.
	EC2.5 Eliminar categoría. Para ello seleccionar un elemento y ejecutar el botón Eliminar y después Aceptar.	100,50,25	Se muestra una notificación sobre el categoría que se desea eliminar.

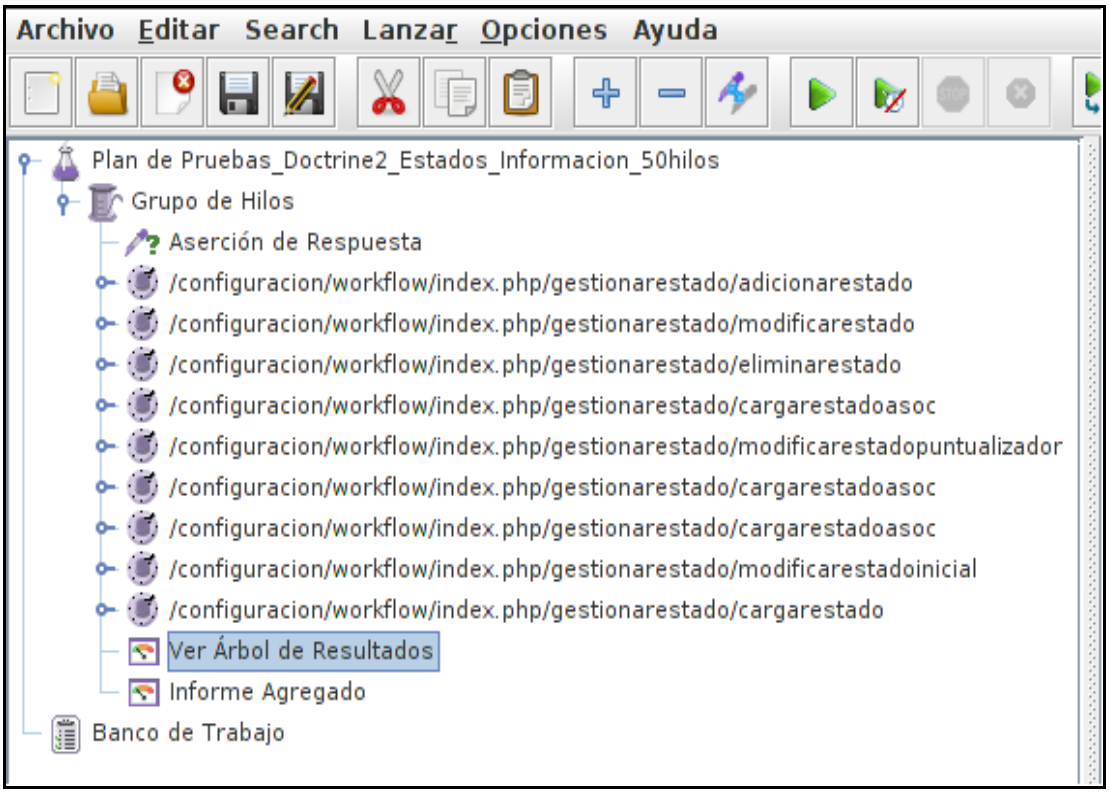
EC3- Tipo de FIP.	EC3.1- Mostrar FIP. Para ello ejecutar click en el indicador "Tipo FIP".	100,50,25	Esta interfaz muestra información relacionada con los FIP registrados en la aplicación.
	EC3.2- Buscar FIP. Para ello llenar el campo Denominación: y ejecutar click en el botón "Buscar"	100,50,25	Esta interfaz muestra información relacionada con los FIP registrados en la aplicación.
	EC3.3- Insertar un FIP. Para ello ejecutar click en el botón Adicionar llenar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para insertar un FIP en la base de datos desde la aplicación.
	EC3.4- Modificar FIP. Para ello seleccionar un elemento ejecutar click en el botón Modificar, cambiar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para modificar un FIP en la base de datos desde la aplicación.
	EC3.5 Eliminar FIP. Para ello seleccionar una fila y ejecutar el botón Eliminar y después Aceptar.	100,50,25	Se muestra una notificación sobre el FIP que se desea eliminar.
EC4- Categoría de la actividad	EC4.1- Mostrar categoría de la actividad. Para ello ejecutar click en el indicador "Categoría de la actividad".	100,50,25	Esta interfaz muestra información relacionada con la categoría de la actividad registrada en la aplicación.
	EC4.2- Buscar categoría de la actividad. Para ello llenar el campo Denominación: y ejecutar click en el botón "Buscar"	100,50,25	Esta interfaz muestra información relacionada con la categoría de la actividad registrados en la

			aplicación.
	EC4.3- Insertar categoría de la actividad Para ello ejecutar click en el botón Adicionar llenar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para insertar la categoría de la actividad en la base de datos desde la aplicación.
	EC4.4- Modificar categoría de la actividad. Para ello seleccionar un elemento ejecutar click en el botón Modificar, cambiar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para modificar la categoría de la actividad en la base de datos desde la aplicación.
	EC4.5 Eliminar categoría de la actividad. Para ello seleccionar una fila y ejecutar el botón Eliminar y después Aceptar.	100,50,25	Se muestra una notificación sobre la categoría de la actividad que se desea eliminar.
EC5- Tipo de actividad	EC5.1- Mostrar tipo de actividad. Para ello ejecutar click en el indicador “Tipo de actividad”.	100,50,25	Esta interfaz muestra información relacionada con tipo de actividad registrados en la aplicación.
	EC5.2- Buscar tipo de actividad. Para ello llenar el campo Denominación: y ejecutar click en el botón “Buscar”	100,50,25	Esta interfaz muestra información relacionada con tipo de actividad registrados en la aplicación.

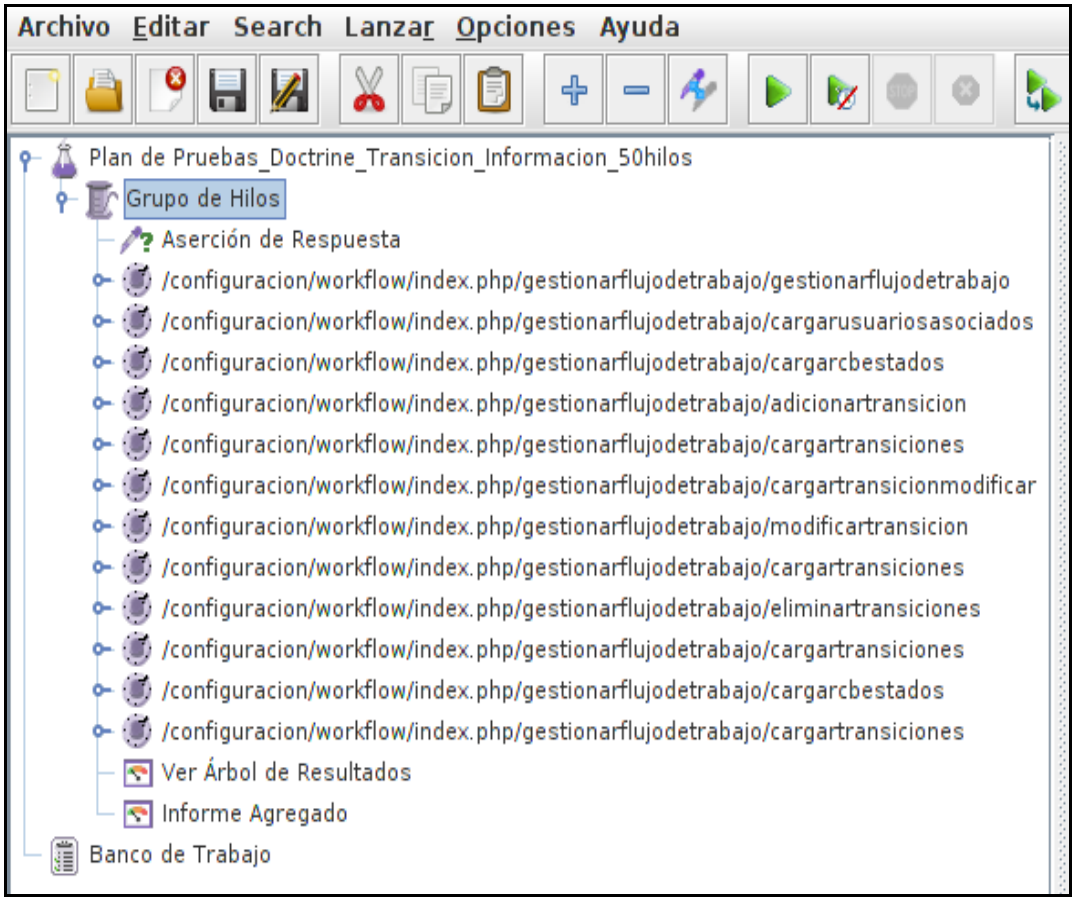
	EC5.3- Insertar tipo de actividad. Para ello ejecutar click en el botón Adicionar llenar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para insertar tipo de actividad en la base de datos desde la aplicación.
	EC5.4- Modificar tipo de actividad. Para ello seleccionar un elemento ejecutar click en el botón Modificar, cambiar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para modificar tipo de actividad en la base de datos desde la aplicación.
	EC5.5 Eliminar tipo de actividad. Para ello seleccionar una fila y ejecutar el botón Eliminar y después Aceptar.	100,50,25	Se muestra una notificación sobre tipo de actividad que se desea eliminar.
EC6- Nivel de acceso al FIP	EC6.1- Mostrar nivel de acceso al FIP. Para ello ejecutar click en el indicador “Niveles de acceso al FIP”.	100,50,25	Esta interfaz muestra información relacionada con nivel de acceso al FIP registrados en la aplicación.
	EC6.2- Buscar nivel de acceso al FIP. Para ello llenar el campo Denominación: y ejecutar click en el botón “Buscar”	100,50,25	Esta interfaz muestra información relacionada con nivel de acceso al FIP registrados en la aplicación.
	EC6.3- Insertar nivel de acceso al FIP. Para ello ejecutar click en el botón Adicionar llenar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para insertar nivel de acceso al FIP en la base de datos desde la aplicación.

	EC6.4- Modificar nivel de acceso al FIP. Para ello seleccionar un elemento ejecutar click en el botón Modificar, cambiar los campos y ejecutar click en Aceptar.	100,50,25	Esta interfaz muestra los campos necesarios para modificar nivel de acceso al FIP en la base de datos desde la aplicación.
	EC6.5 Eliminar nivel de acceso al FIP. Para ello seleccionar una fila y ejecutar el botón Eliminar y después Aceptar.	100,50,25	Se muestra una notificación sobre nivel de acceso al FIP que se desea eliminar.

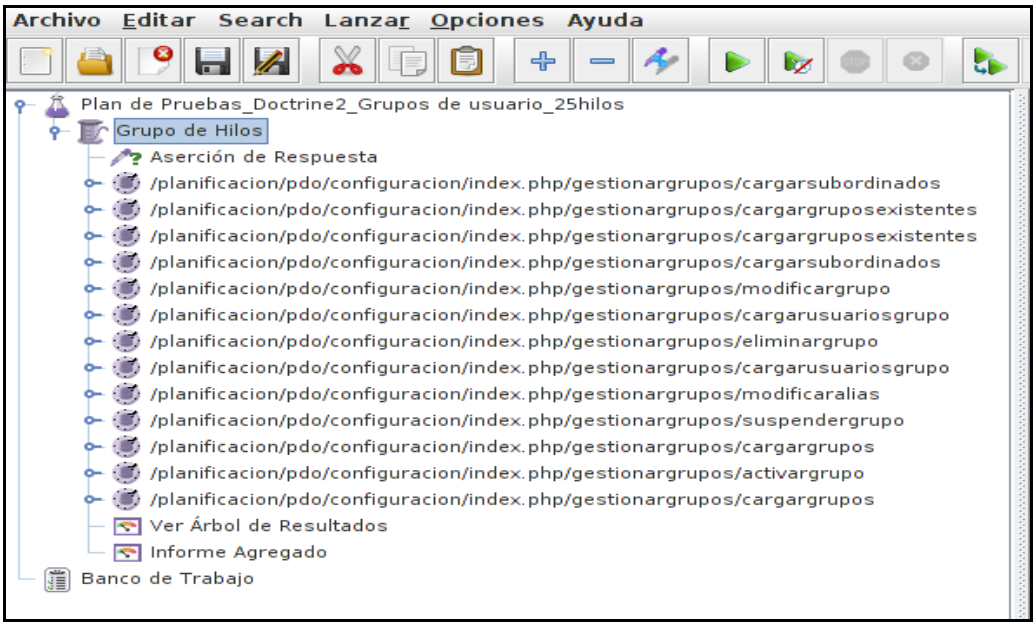
Anexo 6. Confección del Plan de Pruebas para el Caso de Prueba de Carga "Gestionar Estados de la Información"



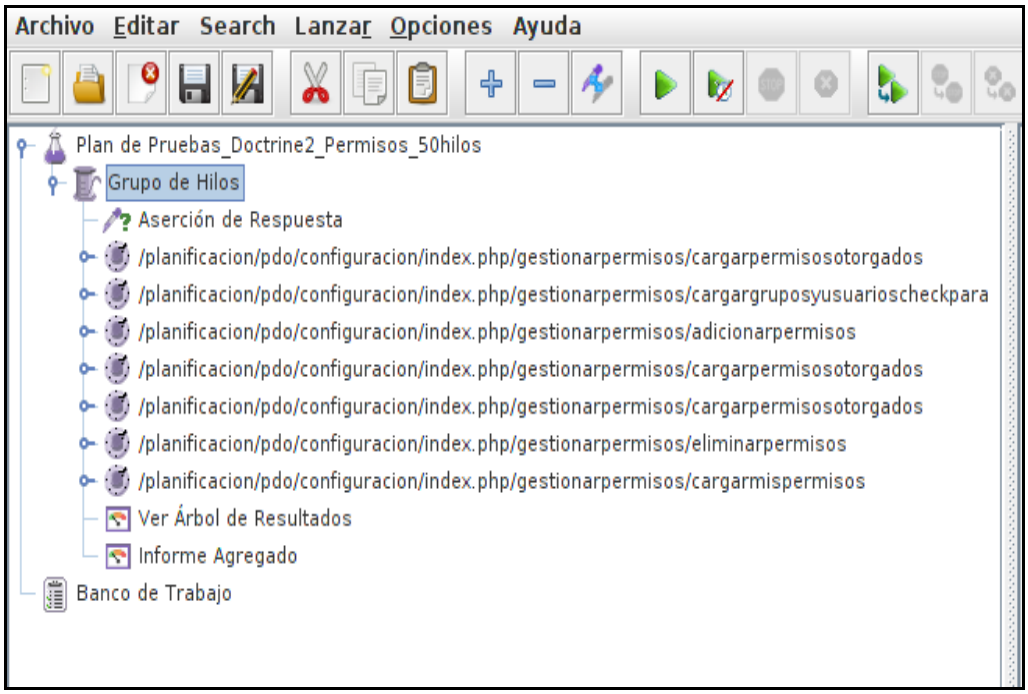
Anexo 7. Confección del Plan de Pruebas para el Caso de Prueba de Carga “Gestionar Transición de la Información”



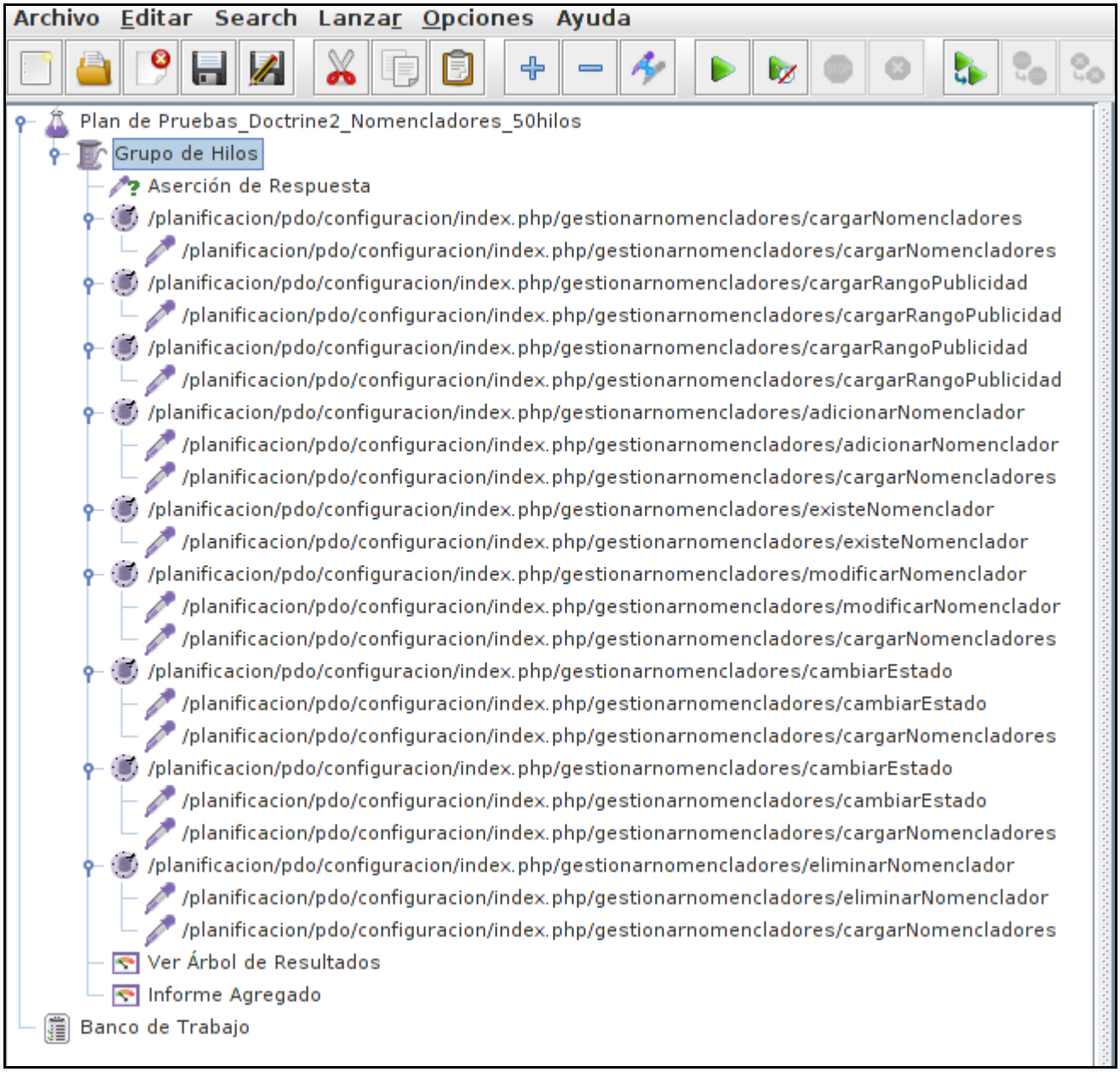
Anexo 8. Confección del Plan de Pruebas para el Caso de Prueba de Carga “Gestionar Grupo de Usuario por Rol”



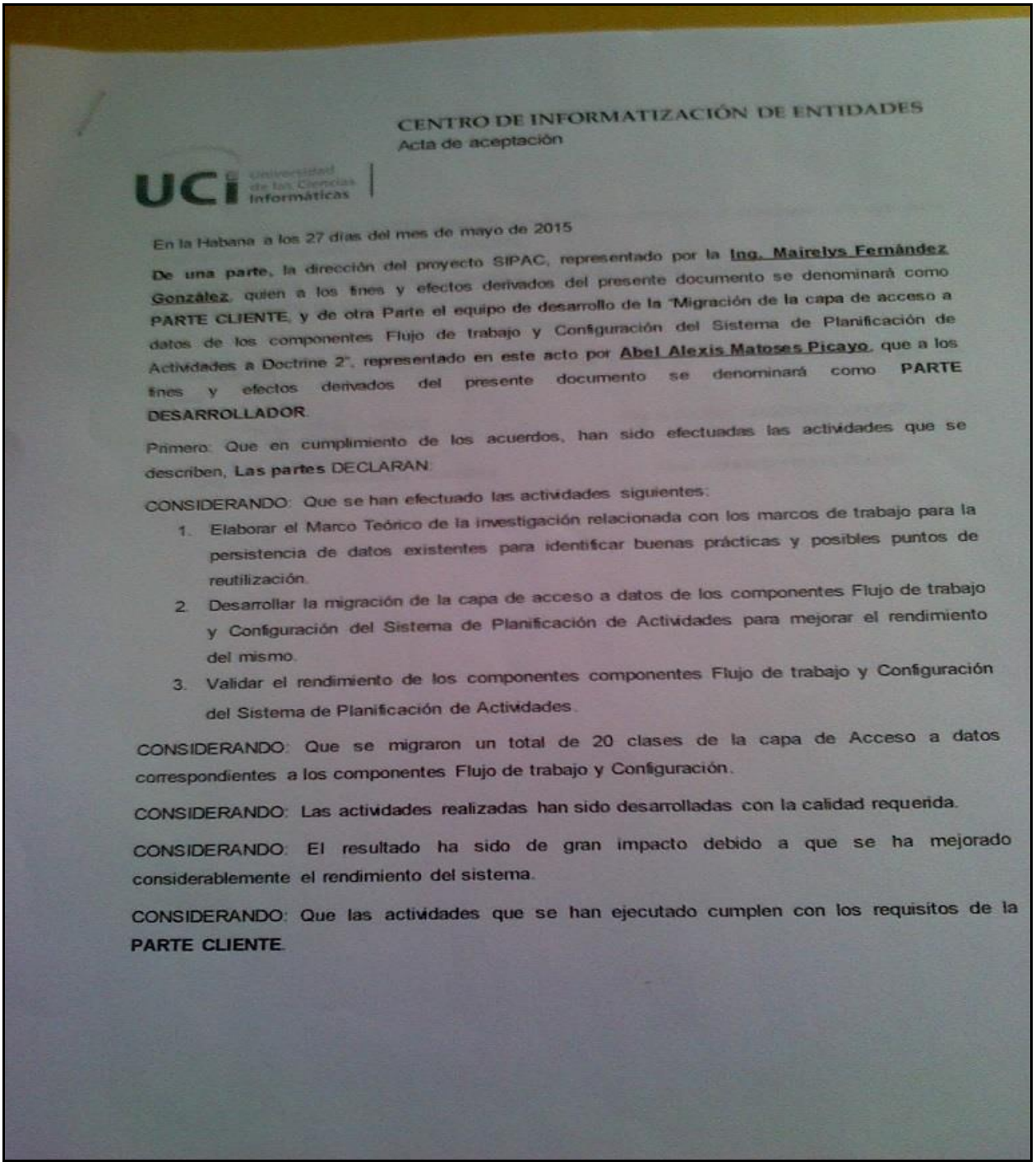
Anexo 9. Confección del Plan de Pruebas para el Caso de Prueba de Carga “Gestionar Permisos”



Anexo 10. Confección del Plan de Pruebas para el Caso de Prueba de Carga “Gestionar Nomencladores”



Anexo 11. Acta de Aceptación



CENTRO DE INFORMATIZACIÓN DE ENTIDADES
Acta de aceptación

UCI Universidad de las Ciencias Informáticas

En la Habana a los 27 días del mes de mayo de 2015

De una parte, la dirección del proyecto SIPAC, representado por la **Ing. Mairelys Fernández González**, quien a los fines y efectos derivados del presente documento se denominará como **PARTE CLIENTE**, y de otra Parte el equipo de desarrollo de la "Migración de la capa de acceso a datos de los componentes Flujo de trabajo y Configuración del Sistema de Planificación de Actividades a Doctrine 2", representado en este acto por **Abel Alexis Matoses Picayo**, que a los fines y efectos derivados del presente documento se denominará como **PARTE DESARROLLADOR**.

Primero: Que en cumplimiento de los acuerdos, han sido efectuadas las actividades que se describen, **Las partes** DECLARAN:

CONSIDERANDO: Que se han efectuado las actividades siguientes:

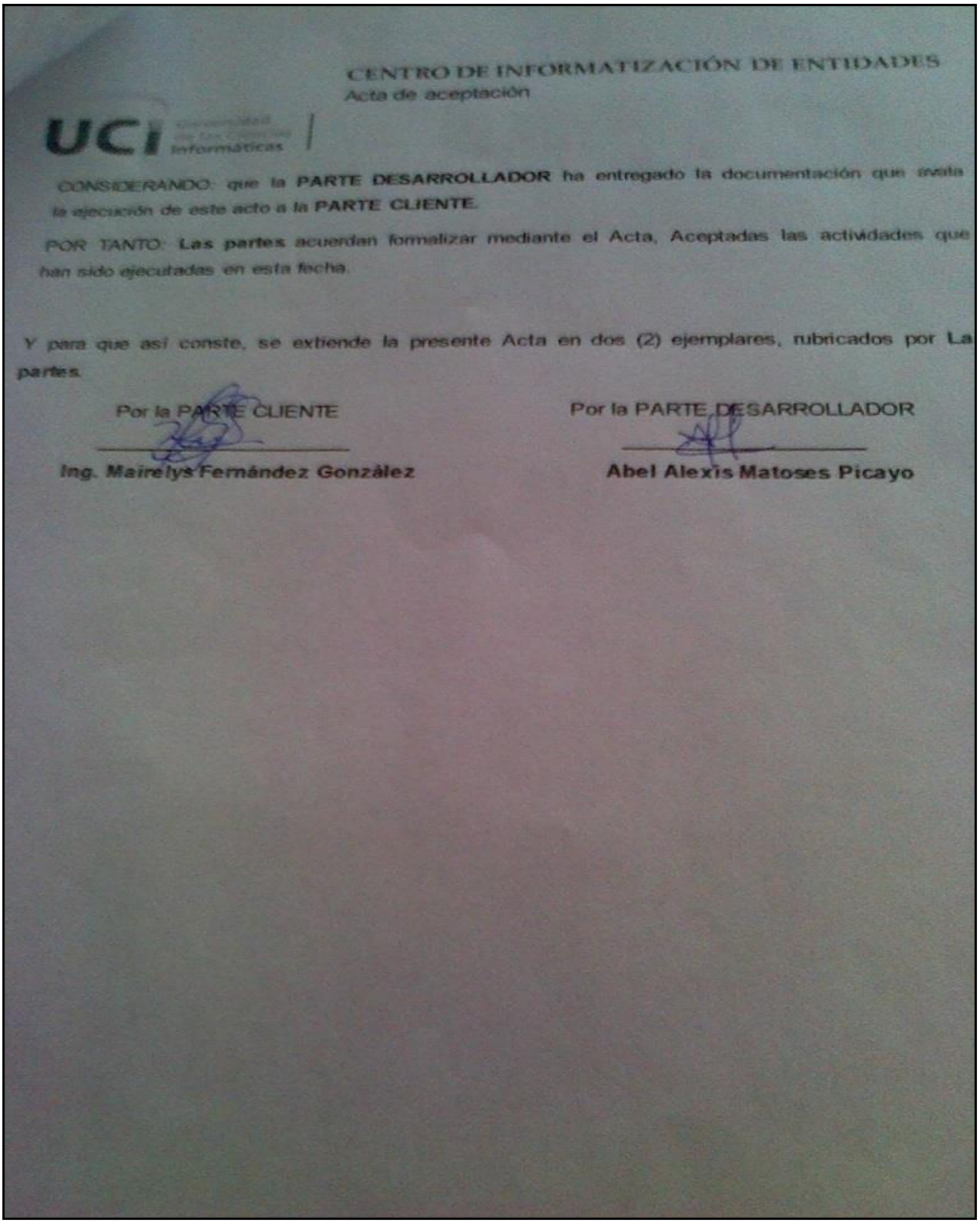
1. Elaborar el Marco Teórico de la investigación relacionada con los marcos de trabajo para la persistencia de datos existentes para identificar buenas prácticas y posibles puntos de reutilización.
2. Desarrollar la migración de la capa de acceso a datos de los componentes Flujo de trabajo y Configuración del Sistema de Planificación de Actividades para mejorar el rendimiento del mismo.
3. Validar el rendimiento de los componentes componentes Flujo de trabajo y Configuración del Sistema de Planificación de Actividades.

CONSIDERANDO: Que se migraron un total de 20 clases de la capa de Acceso a datos correspondientes a los componentes Flujo de trabajo y Configuración.

CONSIDERANDO: Las actividades realizadas han sido desarrolladas con la calidad requerida.

CONSIDERANDO: El resultado ha sido de gran impacto debido a que se ha mejorado considerablemente el rendimiento del sistema.

CONSIDERANDO: Que las actividades que se han ejecutado cumplen con los requisitos de la **PARTE CLIENTE**.



CENTRO DE INFORMATIZACIÓN DE ENTIDADES
Acta de aceptación

UCI Universidad de las Ciencias Informáticas

CONSIDERANDO: que la **PARTE DESARROLLADOR** ha entregado la documentación que avala la ejecución de este acto a la **PARTE CLIENTE**.

POR TANTO: **Las partes** acuerdan formalizar mediante el Acta, Aceptadas las actividades que han sido ejecutadas en esta fecha.

Y para que así conste, se extiende la presente Acta en dos (2) ejemplares, rubricados por **La partes**.

Por la PARTE CLIENTE

Ing. Mairelys Fernández González

Por la PARTE DESARROLLADOR

Abel Alexis Matoses Picayo