Universidad de las Ciencias Informáticas

Facultad 6

“Migración de la capa de acceso a datos del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades a Doctrine 2”.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autora: Dalinda Llapur Del Río

Tutor: Ing. René Rodrigo Bauta Camejo

La Habana, Marzo del 2015

“Año 57 de la Revolución”

Declaro ser autora de la presente tesis que tiene por título: Migración de la capa de acceso a datos del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades a Doctrine 2 y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Ing. René Rodrigo Bauta Camejo.	Dalinda Llapur Del Río.
Tutor	**Autora**

Tutor: Ing. René Rodrigo Bauta Camejo

Edad: 29 años

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

Categoría docente: Instructor

E-mail: rrbauta@uci.cu

Ingeniero en Ciencias Informáticas, graduado en 2009 en la Universidad de las Ciencias Informáticas. Instructor. Seis años de experiencia en el desarrollo de software. Seis años de graduado.

Agradezco a toda mi familia por confiar en mí, mis padres, mi abuela Malvis y mi hermana, mi motor impulsor a seguir adelante.

A mi tío Roberto, mi tía Iliana, mi prima Yuly, mi tía Mary, siempre a pesar de la distancia, estuvieron muy cerquita mío.

A mi titico un muy especial agradecimiento, mi bastón, mi consejero.

A todos mis amigos en especial a mi Amadin, Lauri y Lin, siempre en todo momento me dieron los mejores consejos de seguir y terminar la carrera.

A mi tutor y amigo Rene, por haberme brindado su ayuda, sobre todo su paciencia.

A todas las personas que aportaron su granito de arena, especialmente del centro CEIGE a Mairelys, Tere, Yoandy, Inoelkis, siempre eternamente les estaré agradecida.

A todos los que están aquí presente por haberme dedicado este tiempo.
Gracias a todos

El presente trabajo de diploma va dedicado a mis padres que siempre me hicieron ver que los sueños se cumplen, solo tienes que proponértelo. Son lo más grande que tengo los adoro, gracias por su paciencia y comprensión, para ellos es esta tesis y todo lo que haga en lo adelante.

A mi hermana a pesar de ser una beba, me dio la fuerza de seguir hasta la meta final.

A mi novio, por haber estado conmigo en las buenas y en las malas, en mis peores momentos, muchas gracias.

A mis abuelos Malvis, Zoraida, Tato por siempre estar pendientes, cuidando de mí, por entenderme. Los quiero

RESUMEN

En Cuba es sumamente importante lograr una planificación eficiente de actividades en concordancia con los recursos que se disponen. Esta se elabora siguiendo el modelo de Planificación por objetivos que surge de la unión de la Dirección por Objetivos y la Planeación Estratégica. En el Centro de Informatización de Entidades, perteneciente a la Facultad 3, es desarrollado un sistema para la planificación de actividades, basado en la “Instrucción No.1 del Presidente de los Consejos de Estado y de Ministros” para la planificación en Cuba. Mediante la misma se interrelaciona objetivos de trabajo y actividades en tiempo real; garantizando el seguimiento del desarrollo y cumplimiento de los objetivos y tareas principales en las entidades como parte de la Planeación Estratégica y Operativa. La capa de acceso a datos de este sistema quedó obsoleta, pues se basa en el mapeador objeto-relacional Doctrine en su versión 1.2.2, el cual presenta algunos problemas como la carencia de soporte, deficiencias en el manejo de los datos y problemas de rendimiento. En este trabajo se describe el proceso de migración de la capa de acceso a datos del componente Planeación del módulo Planificación a la versión 2.4.2 de Doctrine. Los resultados obtenidos en la migración fueron validados empleando la herramienta JMeter.

Palabras claves: capa de acceso a datos, Doctrine 2, Mapeador de Objeto Relacional, migración.

ABSTRACT

In Cuba is highly important to accomplish an efficient planning of activities according to the resources available. This can be achieved applying the model of planning by objectives that emerged through the union between the direction by objectives and the strategic planning. In the center for computerization of work stations that belongs to the faculty 3, it has been developed a system for the organization of activities. The Instruction No 1 from the head of the government for the planning in Cuba allows to connect goals in the work with activities in real time, this way can be guaranteed the continuation of the development and the success of the goals and main tasks in the institutions as a part of the strategic and operative planning. The platform that access the data in the system is out of date, because is based on the mapper object-relational. The version of doctrine 1.2.2 presents some problems such as: the lack of support, flaws in the data-processing and performance (functionality issues). This paper describes the migration-process from the data-access-platform to the 2.4.2 version of Doctrine. The final results were proved using the tool Jmeter.

Keywords: the data-access-platform, Doctrine 2, mapper object-relational, migration.

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA 5

1.1 INTRODUCCIÓN 5
1.2 MARCO CONCEPTUAL 5
1.2.1 Planificación 5
1.2.2 Migración 6
1.3 MARCO DE TRABAJO DE PERSISTENCIA DE OBJETOS RELACIONALES 8
1.3.1 Hibernate 8
1.3.2 Propel 9
1.3.3 Django 10
1.3.4 Subsonic 11
1.3.5 Doctrine 1.2.2 12
1.3.6 Doctrine 2.4.2 13
1.4 TECNOLOGÍAS Y HERRAMIENTAS PARA EL DESARROLLO 15
1.4.1 Subversión v1.6.6 15
1.4.2 RapidSVN v0.12.0 16
1.4.3 PostgreSQL v9.1 16
1.4.4 NetBeans v7.4 16
1.4.5 Apache v2.2 16
1.4.6 Sauxe 2.0 17
1.4.7 Marco de trabajo Zend v1.11 17
1.4.8 PHP v5.3.10 17
1.4.9 JMeter 17
1.5 CONCLUSIONES PARCIALES 18

CAPÍTULO 2: DESCRIPCIÓN DE LA MIGRACIÓN 19

2.1 FASE INICIAL 19
2.1.1 Arquitectura del marco de trabajo Sauxe 20
2.1.2 Estudio de la estructura de carpeta de la capa de acceso a datos del componente Planeación 20
2.2 FASE DE INTEGRACIÓN 21
2.2.1 Estudio de las librerías seleccionadas 21
2.2.2 Estudio de las clases seleccionadas 22
2.3 FASE DE IMPLEMENTACIÓN 26
2.3.1 Redefinir capa de acceso a datos. 26
2.3.2 Estructura de carpeta 27
2.3.3 Flujo de dato de la capa de acceso a datos 29
2.3.4 Mapeo de las tablas 29
2.3.5 Consultas y funcionalidades en los Repository 36
2.3.6 Redefinir capa del negocio 38
2.3.7 Funcionalidades de las clases Model 39
2.3.8 Actualizar estado de los elementos de configuración 40
2.4 CONCLUSIONES PARCIALES 40

CAPÍTULO 3: VALIDACIÓN DE LA MIGRACIÓN 41

3.1 INTRODUCCIÓN 41
3.2 PRUEBAS DE RENDIMIENTO 41
3.3 ENTORNO DE PRUEBA 43
3.3.1 Descripción de las pruebas de rendimiento 43

3.4 RESULTADO DE LAS PRUEBAS DE RENDIMIENTO 46
3.5 CONCLUSIONES PARCIALES 50
RECOMENDACIONES **52**
BIBLIOGRAFÍA REFERENCIADA **53**
ANEXOS **56**

ÍNDICE DE FIGURAS

Figura.1 Relación de los conceptos más asociados al dominio. (Elaboración propia) 6
Figura.2 Modelo negocio Doctrine1 (20). 13
Figura.3 Fases del procedimiento (10). 19
Figura.4 Estructura de carpeta de la capa de acceso a datos del componente Planeación para Doctrine 1. 21
Figura.5 Ubicación de las nuevas librerías de Doctrine. 22
Figura.6 Clase seleccionada para la integración. 22
Figura.7 Ubicación del fichero de configuración. 24
Figura.8 Ubicación de la clase TransactionManager. 24
Figura.9 Método OpenConections () de la clase TransactionManager. 26
Figura.10 Relación entre las estructuras de carpeta. 27
Figura.11 Flujo de la CAD. 29
Figura.12 Definición tabla y columna para Doctrine 1. 33
Figura.13 Definición tabla y columna para Doctrine 2. 33
Figura.14 Definición tabla y columna utilizando función hasColumn (). 34
Figura.15 Definición tabla y columna utilizando anotaciones. 34
Figura.16 Ejemplo de relación ManytoOne. 34
Figura.17 Ejemplo de relación en Doctrine 1. 35
Figura.18 Ejemplo nueva forma de mapeo en Doctrine 2. 35
Figura.19 Ejemplo de consulta en Doctrine 1. 36
Figura.20 Ejemplo de consulta en Doctrine 2. 37
Figura.21 Estructura de la clase Model. 39
Figura.22 Ejemplo de la Model en Doctrine 1. 39
Figura.23 Ejemplo de llamado a las funcionalidades de la clase Model. 40
Figura.24 Elemento Grupo de Hilo. 44
Figura.25 Selección del Servidor Proxy HTTP. 44
Figura.26 Selección de Informe Agregado. 45
Figura.27Gráfico de Resultados del Tiempo Total. 49
Figura.28 Gráfico de Resultados del Tiempo Promedio. 49

ÍNDICE DE TABLAS

Tabla.1 Comparación de rendimiento en las versiones de Doctrine (19). 14
Tabla.2 Clases generadas por cada tabla. 28
Tabla.3 Descripción de las anotaciones utilizadas en Doctrine 2 (33). 31
Tabla.4 Valores correspondientes a la Prueba cargar actividades para 200 hilos. 46
Tabla.5 Valores correspondientes a la Prueba Cargar Plan para 200 hilos. 47
Tabla.6 Valores correspondientes a la Prueba Obtener Reporte. 47
Tabla.7 Valores correspondientes a la Prueba Cargar Actividades para 200 hilos. 47
Tabla.8 Valores correspondientes a la Prueba Cargar Plan para 200 hilos. 48
Tabla.9 Valores correspondientes a la Prueba Obtener Reporte para 200 hilos. 48

INTRODUCCIÓN

En la actualidad el desarrollo tecnológico ha creado nuevos enfoques en las Tecnologías de la Información y las Comunicaciones (TIC), siendo estas esenciales en el proceso de desarrollo social. El estado cubano se encuentra en un proceso de informatización para la mejora del sistema de control y seguimiento de los recursos, evitando el derroche o el uso innecesario de los mismos en las empresas. Con este fin, se enfocan en los procesos de planificación, haciendo uso de métodos como el de Planificación por Objetivos (PPO), se define una meta en función de la cual se trazan planes, se definen áreas de resultados clave (áreas de concentración de metas) y se asignan actividades para cumplir dichos objetivos. A raíz del establecimiento del método de planificación por objetivo en todo el país, se comenzaron a utilizar en muchas entidades varias herramientas para la planificación, el establecimiento de actividades, así como su control, en cuyo caso se encuentran Outlook, Project y Excel. El uso de diversas herramientas trae consigo dificultades a la hora de la consolidación de la información de las empresas a distintos niveles teniendo en cuenta el nivel jerárquico de dirección y por consiguiente que no se pueda lograr un control y seguimiento de la planificación (1).

La Universidad de las Ciencias Informáticas (UCI) cuenta con varios centros de producción, entre ellos se encuentra el Centro de Informatización de Entidades (CEIGE) perteneciente a la Facultad 3. Este centro tiene como misión el desarrollo de productos y servicios asociados para la gestión integral de entidades, contribuyendo a la formación integral de profesionales que respondan a las necesidades del progreso científico-técnico y socioeconómico del país. Uno de los productos que desarrolla este centro es el Sistema para la Planificación de Actividades (SIPAC) (2).

SIPAC es un sistema desarrollado para la web, que cuenta con diversas funcionalidades para desempeñar la planificación en las entidades. Sus características funcionales están basadas 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, Organismo de la Administración Central del Estado (OACE), Entidades Nacionales y Administraciones locales del Poder Popular (4). La arquitectura base presenta 4 niveles de empaquetamiento: módulo, componente, funcionalidad y requisito.

SIPAC está integrado por los módulos Flujo de trabajo que permite definir el flujo de información de los elementos de la planificación a partir de estados de los documentos y las transiciones; el módulo Seguridad que permite gestionar la seguridad del sistema de forma centralizada; el módulo Planificación que permite el registro, seguimiento y control de la Planeación Estratégica y Operativa para todos los niveles organizacionales (ver anexo 1).

El módulo Planificación contiene los componentes Configuración que brinda las funcionalidades para crear grupos de usuarios a partir de los usuarios sobre los cuales se puede planificar, teniendo en

cuenta el dominio de acceso y la estructura definida configurados en los subsistemas Seguridad y Estructura respectivamente; el componente de Notificaciones que permite la generación de notificaciones a partir de las acciones que se realizan sobre los diferentes elementos y el componente de Planeación que permite la gestión de los Planes, Actividades, Objetivos, Factores que Influyen en Plan (FIP) y Áreas de Resultados Clave (ARC).

Teniendo en cuenta que el componente Planeación es crítico para el sistema al ser un componente horizontal y que es quien garantiza la gestión de los elementos de la planificación, se decide iniciar el proceso de migración de su capa de acceso a datos.

SIPAC está desarrollado en el marco trabajo (MT) Sauxe, este fue creado en el año 2008 como base para el desarrollo de software en el CEIGE. Su base tecnológica cuenta con una arquitectura en capas que a su vez presenta en su capa superior un Modelo Vista Controlador (MVC). Este marco de trabajo contiene un conjunto de componentes reutilizables, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo (3).

El MT Sauxe, está compuesto por Zend Framework en su versión 1.11 para el desarrollo de aplicaciones y servicios web. Es un marco de trabajo de código abierto, uno de los más completos, entendibles y fáciles de utilizar. Utiliza ExtJS en la capa de presentación, por la gran gama de componentes que se pueden reutilizar para agilizar el proceso de desarrollo y mostrarle al usuario una interfaz amigable y fácil de usar. Como marco de trabajo para la persistencia de los datos emplea Doctrine 1.2.2, es un marco de trabajo configurable, brinda una capa de abstracción de la base de datos muy completa. Doctrine utiliza la técnica de mapeo objeto-relacional a través del uso de un Mapeador de Objetos-Relacional (ORM, *Objects Relational Mapper*). Esta herramienta mapea los objetos, en el ámbito de la aplicación, a los datos que hay que persistir en una base de datos, destinado a separar el código de la base de datos relacional y el código de la aplicación aumentando así la flexibilidad de la aplicación (4).

Actualmente el SIPAC, usa la versión 1.2.2 del ORM Doctrine, el cual garantiza la persistencia de los datos. El uso de esta versión provoca un rendimiento reducido del componente Planeación, al ejecutar consultas que necesiten manejar gran cantidad de información o hacer referencia a varias tablas para obtener información. Cuando se realizan consultas sobre múltiples tablas a un sistema gestor de base de datos, la aplicación tendrá que convertir la consulta al SQL del proveedor usado, enviarla, leer los registros, limpiarlos y convertirlos a objetos (5)

Además debido a la forma en que se maneja la herencia, las consultas son excesivamente grandes y complejas, provocando demoras en la implementación de los requisitos y la obtención de la solución final (5). Otro inconveniente que presenta el uso de esta versión de Doctrine es que a partir del 1 de

junio de 2011 se le dejó de dar soporte, provocando que no se solucionen vulnerabilidades en el código que puedan atentar contra el rendimiento (6).

Las pruebas realizadas por el equipo de desarrollo de SIPAC al componente dieron como resultado tiempos de ejecución elevados al utilizar el marco de persistencia de datos. A partir de los resultados obtenidos se evidenció que el rendimiento era mayor cuando no se utilizaba esta versión del ORM.

A partir de la problemática antes descrita se identificó como **problema a resolver:**

¿Cómo mejorar el rendimiento del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades?

El **objeto de estudio** en el cual se enmarca el problema anteriormente planteado es:

La migración en el proceso de desarrollo de software**.**

Por consiguiente la presente investigación se encuentra enmarcada en el **campo de acción:**

La migración de la capa de acceso a datos del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades.

Con el propósito de darle solución al problema formulado se establece como **objetivo general:**

Realizar la migración de la capa de acceso a datos del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades haciendo uso de Doctrine 2.4.2 para mejorar el rendimiento del mismo.

Teniendo en cuenta los elementos expuestos anteriormente, se define como **idea a defender:**

Si se realiza la migración a Doctrine 2.4.2 del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades aumentará su rendimiento.

El objetivo general fue desglosado en los siguientes **objetivos específicos**:

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 del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades para mejorar el rendimiento del mismo.
3. Validar el rendimiento del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades.

Métodos Científicos:

Los métodos científicos se clasifican en teóricos y empíricos. Estos en el proceso de investigación, están relacionados (7).

Los métodos teóricos permiten estudiar las características del objeto de investigación que no son observables directamente (8). De ellos se utilizan los siguientes:

Inductivo – Deductivo: Método inductivo-deductivo se refiere al conocimiento dialéctico de la realidad, a los procesos lógicos de razonamiento que son fundamentales para la construcción de una argumentación. Aplicar este método permitió, identificar las características principales y beneficios que brindan los ORM más utilizados actualmente.

Analítico – Sintético: Estudia los hechos, partiendo de la característica del objeto de estudio en cada una de sus partes para estudiarlas en forma integral. En la investigación el uso de este método, permitió analizar las características de cada ORM, estudiarlas de manera independiente e integrarlas partiendo de lo estudiado en dicho análisis.

Métodos empíricos: describen y explican las características del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia.

Observación: Instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado. En la investigación el uso de este método permitió concentrar la información en la situación actual del SIPAC, en el MT Sauxe y los inconvenientes de la utilización de Doctrine versión 1.2.2

Medición: La medición consiste en obtener información sobre la cualidad o propiedad de un objeto, observar y registrar minuciosamente todo aquello que en el objeto de estudio seleccionado y de acuerdo con la teoría, sea relevante. En la investigación se utilizó como vía para obtener el rendimiento del sistema SIPAC, realizando pruebas mediante llamadas a funciones del componente Planeación.

Estructura del documento:

En el Capítulo 1: Se elabora la fundamentación teórica del trabajo, se realiza un estudio de las principales tendencias de los marcos de persistencia de datos, y se aborda las características de las herramientas a usar durante la migración de la capa de acceso a datos.

En el Capítulo 2: Se realiza una descripción de las fases Inicio, Integración e Implementación del procedimiento seleccionado para la migración de la capa de acceso a datos.

En el Capítulo 3: Se describen los resultados obtenidos una vez terminada la fase de Pruebas del procedimiento seleccionado para la migración de la capa de acceso a datos.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se ofrece un análisis de los temas fundamentales que sustentan la investigación y que contribuyen a esclarecer el objeto de estudio o el marco teórico. Se exponen también los principales conceptos asociados a los marcos de persistencia de datos. Por último se ofrece una visión de las tecnologías y herramientas utilizadas en la migración.

1.2 Marco conceptual

A continuación se conceptualizarán los elementos fundamentales de la Planificación por Objetivos enfatizando los procesos de Planificación Estratégica y Planificación Operativa, como concepciones bases del modelo adoptado nacionalmente. Entre los subprocesos de la Planificación Operativa se resalta el proceso de Puntualización del Plan.

1.2.1 Planificación

Actualmente existen diversas definiciones sobre el concepto de planificación. Entre ellas, se encuentra la de Jiménez C., W. 1982 que plantea que "…la planificación es un proceso de toma de decisiones para alcanzar un futuro deseado, teniendo en cuenta la situación actual y los factores internos y externos que pueden influir en el logro de los objetivos" (9). Por otra parte Terry G. y Franklin S. (1987) define que "…la planificación es el proceso de seleccionar información y hacer suposiciones respecto al futuro para formular las actividades necesarias para realizar los objetivos organizacionales" (10).

La Planificación por Objetivo se refiere a una forma particular de funcionamiento en torno a las tareas directivas. Es un proceso de planificación que tiene como principal objetivo, lograr una adecuada coordinación de personas, recursos y mecanismos de una organización para obtener resultados relevantes sobre la base de la amplia y efectiva participación. Por consiguiente se entiende por Planificación por Objetivos, como los fines hacia los cuales se orientan los planes, actividades, recursos y tareas de cualquier unidad organizativa para cumplir con el propósito de su creación. De acuerdo con la definición de Stoner, J.1996, existen dos tipos básicos de planificación: la Planificación Estratégica y la Planificación Operativa. La Planificación Operativa consiste en formular planes a corto plazo que pongan de relieve las diversas partes de la organización (11).

Por otra parte la Planificación Estratégica está diseñada para satisfacer las metas generales de la organización, mientras la Planificación Operativa muestra cómo se pueden aplicar los planes estratégicos en el quehacer diario. La Planificación Estratégica es planificación a largo plazo que enfoca a la organización como un todo (1).

El proceso de Puntualización del Plan: son correcciones sobre la planificación realizada. Son ajustes de nuevos elementos planificados durante la ejecución de un plan. El proceso abarca los subprocesos de Concepción, Aprobación y Control del plan.

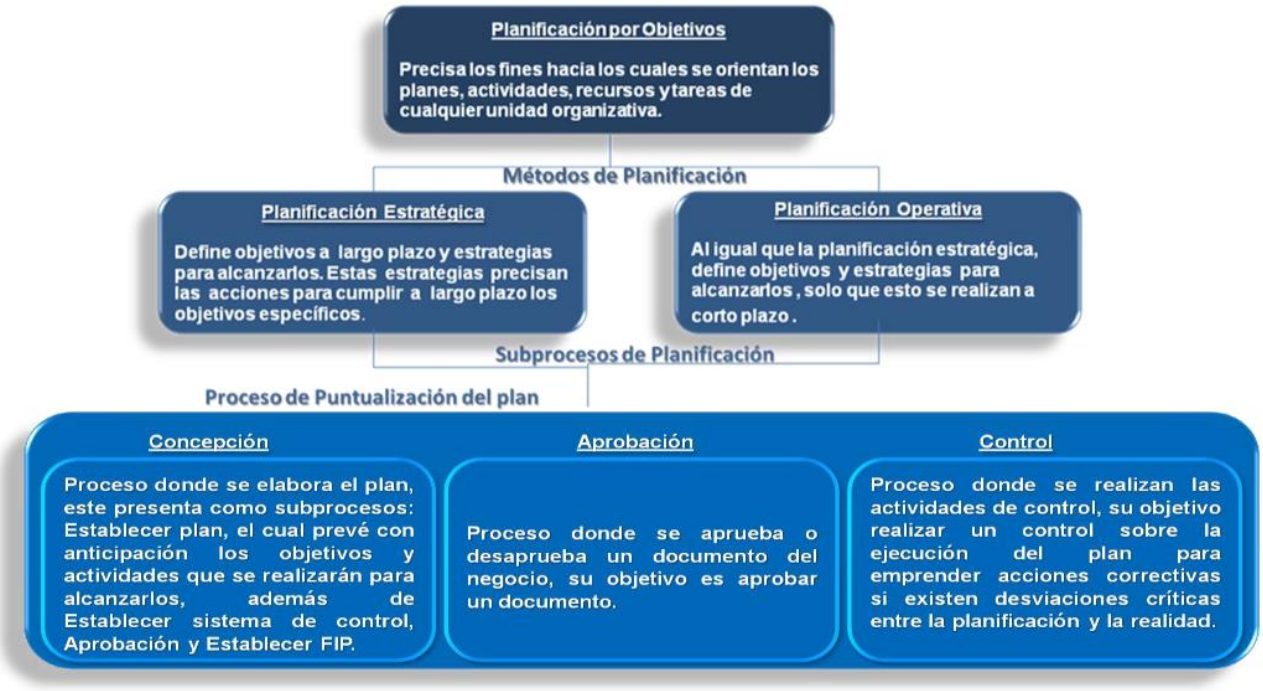


Figura.1 Relación de los conceptos más asociados al dominio. (Elaboración propia)

1.2.2 Migración

La RAE (Real Academia Española) define migración:

1. f. emigración.

2. f. Acción y efecto de pasar de un país a otro para establecerse en él. Se usa hablando de las migraciones históricas que hicieron las razas o los pueblos enteros.

3. f. Viaje periódico de las aves, peces u otros animales migratorios.

4. f. Desplazamiento geográfico de individuos o grupos, generalmente por causas económicas o sociales.

A partir de los conceptos anteriores se puede apreciar que la migración siempre implica un cambio de un estado actual a uno deseado.

En el ámbito informático la migración se refiere al traslado de una aplicación de un ordenador a otro en condiciones de compatibilidad. Es también elevar una versión de un producto software a otra de más alto nivel, o bien el movimiento de una arquitectura a otra. La migración de versiones: consiste en actualizar el software a una versión superior que reemplaza una versión instalada de un producto por una versión más reciente del mismo producto. Este proceso de actualización a una versión

superior no altera normalmente los datos ni las preferencias del usuario existentes al reemplazar el software existente por la nueva versión (12).

Normalmente las personas instalan uno o varios programas y los dejan funcionando durante un período prolongado creyendo que son perfectos. Con el paso del tiempo se detectan fallos y problemas de seguridad en los mismos. Otras veces sencillamente se busca mejorar los programas con nuevas herramientas y opciones. Por ambos motivos surgen nuevas versiones de los mismos programas y llega el momento de actualizarse (12).

Al igual que con los sistemas operativos, es necesario tener, en la medida de lo posible, actualizados los programas, como mínimo los que se utilizan con mayor frecuencia. De esta manera, siempre se podrá contar con las nuevas herramientas y utilidades que brindan estos programas (12).

Mantener los sistemas actualizados es muy importante debido al creciente desarrollo alcanzado por las redes telemáticas e Internet, los cuales han propiciado que los usuarios compartan rápidamente sus conocimientos retroalimentándose unos de otros continuamente (12).

Principales características relacionadas con el procedimiento para la actualización de la capa de acceso a datos

Con el objetivo de facilitar el proceso de migración de la capa de acceso a datos (CAD) del componente Planeación del módulo Planificación del SIPAC se realizó un estudio sobre procedimientos o metodologías que guiaran este proceso. Este estudio solo arrojó como resultado la existencia de un procedimiento elaborado y aplicado en el centro CEIGE que determina las actividades a realizar para la migración de la capa de acceso a datos de las soluciones desarrolladas sobre Sauxe. Este procedimiento propone 4 fases (Inicial, Integración, Implementación y Pruebas). Por cada una de estas fases se definen actividades a realizar para lograr la migración con éxito.

Fase inicial

En la fase inicial se estudia la arquitectura de Sauxe, el alcance que tendrá el proceso, se prepara al equipo de trabajo involucrado en el proceso de migración con la tecnología a usar. En esta etapa se genera el cronograma a usar en todo el proceso de la migración (5).

Fase integración

La fase de integración es la base para realizar la implementación de la migración. En esta fase es imprescindible para recuperar la información de la base de datos lograr la comunicación entre el ORM Doctrine 2 con el MT Zend en su versión 1.11. Además, se hace necesario la creación de directorios, definir las librerías a utilizar, clases que se modifican, crear métodos y clases necesarias y adicionar parámetros de configuración (5).

Fase implementación

En la fase de Implementación se desarrolla la redefinición de la capa de acceso a datos, se realiza el mapeo de las tablas adaptándolas a la nueva estructura de Doctrine 2, se reimplementan los métodos y las consultas. En la capa del negocio se realiza la reestructuración necesaria para la migración (5).

Fase pruebas

En esta etapa se hacen pruebas a cada una de las funcionalidades migradas para corregir incoherencias o errores en el código (5).

1.3 Marco de Trabajo de Persistencia de Objetos Relacionales

Con el objetivo de conocer sobre posibles puntos de reutilización y buenas prácticas en las tecnologías de persistencias o MT para el mapeo Objeto-Relacional se hace un estudio de los ORM**.** El mapeo objeto–relacional, es una técnica que persiste de forma transparente, los objetos de la aplicación a las tablas de una base de datos relacional. Procede como una base de datos virtual, ocultando la arquitectura de base de datos subyacente del usuario. Facilitan una funcionalidad para llevar a cabo las consultas orientadas a objetos. Los ORM también admiten el mapeo de metadatos y ayuda en la gestión de transacciones de la aplicación.(4)

La Persistencia: Es la capacidad que tiene un objeto de “perdurar” fuera del proceso que lo creó. Tiene la capacidad de guardar y almacenar un objeto, en/desde un almacenamiento persistente.(13)

1.3.1 Hibernate

Es una herramienta de mapeo objeto-relacional (ORM) para la plataforma Java, tiene que ver con la persistencia de datos, a su vez se aplica a la base de datos relacionales. No requiere interface o clases base para las clases persistentes, permite un alto rendimiento. Hibernate fue diseñado para funcionar en un clúster de servidor de aplicaciones y ofrecer una arquitectura altamente escalable.(14)

Rendimiento:

Para el manejo del rendimiento, Hibernate 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.(14)

Compatibilidad con el gestor de base de datos PostgreSQL

Hibernate es compatible con el gestor de base de datos PostgreSQL, es una herramienta para la plataforma Java y disponible también para .Net con el nombre de NHibernate.

Uso del Lenguaje PHP

Hibernate no utiliza como lenguaje de programación PHP.

Compatibilidad con el MT Sauxe

No es compatible con el MT Sauxe debido a las tecnologías en que está desarrollado.

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(14)

- ✓ 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 utilizar con frecuencia la recuperación temprana. Usar proxies, que son objetos que permiten la carga diferida, y colecciones perezosas para la mayoría de asociaciones a clases que probablemente no se encuentren en la caché de segundo nivel.

1.3.2 Propel

Propel es una herramienta de mapeo objeto-relacional de código abierto (ORM) para SQL-Bases de datos en PHP. Le permite acceder a su base de datos utilizando un conjunto de objetos,

proporcionando una API sencilla para almacenar y recuperar datos. Proporciona un generador de consultas, la migración de esquemas de bases de datos, ingeniería de base de datos existente (15).

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 significativa mejora del rendimiento (16).

Compatibilidad con el gestor de base de datos PostgreSQL

Este soporta MySQL, PostgreSQL, SQLite, MSSQL y Oracle(15)

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 prestigio elevado mundialmente por sus potencialidades.

Buenas prácticas

- ✓ A cada URL se le debe crear la ruta asociada. Y esto es obligatorio si se quitan 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.

1.3.3 Django

El ORM de Django es un marco de trabajo para el desarrollo de aplicaciones web, escrito en Python. Consiste en un mapeo objeto-relacional que media entre los modelos de datos, definidos como clases de Python, y la base de datos relacional; un sistema para procesar peticiones y un despachador de URL basado en expresiones regulares. Nos brinda una API libre para el acceso dinámico a bases de datos.(17)

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 psycopg; el uso de PostgreSQL en Django logra un delicado equilibrio entre costo, características, velocidad y estabilidad.(17)

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 MVC, esta forma de trabajar permite que sea pragmático.

1.3.4 Subsonic

SubSonic es un marco de persistencia que trabaja en la capa de acceso a datos usando Linq 3.0, es Open Source y está creado para la tecnología ASP.Net. Provee un lenguaje de consultas que permite la extracción de datos de la base de datos de manera muy simple sin menospreciar la potencia. Además de lo mencionado, ofrece en la estructura de clases generadas una separación clara que posibilita implementar el patrón MVC en la aplicación.(18)

Rendimiento

El rendimiento en Subsonic no está garantizado del todo debido a que se deja en manos del usuario la elección de donde se va a guardar el resultado en caché ya sea a nivel de aplicación o a nivel de propio Subsonic.(19)

Compatibilidad con el gestor de base de datos PostgreSQL

Subsonic está creado específicamente para el gestor de base de datos PostgreSQL por lo que su compatibilidad está garantizada.(19)

Uso del Lenguaje PHP

Esta creado sobre la tecnología ASP.Net

Compatibilidad con el MT Sauxe

La compatibilidad de SubSonic con el marco de trabajo Sauxe se ve limitada debido a que los dos utilizan lenguajes diferentes, Subsonic usa ASP.Net y Sauxe PHP.

Comunidad que respalde su soporte

SubSonic cuenta con una comunidad poco abundante y su reputación está limitada sobretodo porque es bastante joven.

Buenas prácticas

- ✓ Los objetos de negocio deben comunicarse con los objetos de acceso a datos a través de interfaces, es decir, siempre depende de abstracciones.
- ✓ Los objetos de acceso a datos deben implementar interfaces definidas por el "cliente" en la capa de lógica de negocio.

1.3.5 Doctrine 1.2.2

Doctrine 1 es una implementación de ActiveRecord (registro activo), que propone hacer la persistencia en el mismo objeto de negocio en la siguiente Figura**,** esto provoca que se pierda independencia a la hora de acoplar los objetos de negocios de la capa de negocios con la persistencia de datos. Usa métodos mágicos y estáticos que lo hace más difícil de probar. Está dividido en dos capas DBAL y ORM (20).

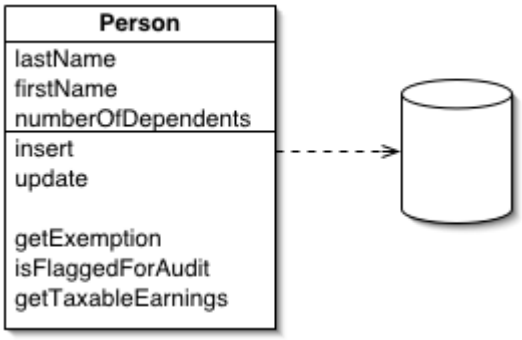


Figura.2 Modelo negocio Doctrine1 (20).

Compatibilidad con lenguaje:

Usa como lenguaje para la comunicación con las bases de datos el DQL dándoles a los programadores una alternativa del SQL que mantiene la flexibilidad sin la necesidad de duplicación de código.

Compatibilidad con el gestor de base de datos PostgreSQL:

Esta versión de Doctrine es compatible con el gestor de base de datos PostgreSQL. El paquete DBAL se encarga de la comunicación de los diferentes gestores de bases de datos con el lenguaje DQL de la capa de acceso a datos de nuestra aplicación.

Uso del Lenguaje PHP

Doctrine es una librería para PHP.

Compatibilidad con el MT Sauxe

Se garantiza la compatibilidad, la versión de Sauxe 2.2 tiene integrado Doctrine 1.2

Comunidad que respalde su soporte

Doctrine 1 no cuenta con soporte desde inicios del mes de junio del año 2011, provocando vulnerabilidades en el código (6)**.**

1.3.6 Doctrine 2.4.2

Doctrine es un ORM para PHP 5 que proporciona persistencia transparente de los objetos desde el lenguaje PHP. Utiliza el patrón *Data Mapper*, logrando separar la lógica de dominio / negocio de la persistencia en un sistema de bases de datos relacional. Una de las principales características de Doctrine es su lenguaje DQL llamado Lenguaje de Consulta de Doctrine (DQL - Doctrine *Query Language*). Soporta las operaciones de Crear, Obtener, Actualizar y Borrar (CRUD - *Create, Retrieve, Update and Delete*) habituales, desde la creación de nuevos registros a la actualización de los

antiguos. Crea manualmente y automáticamente el modelo de base de datos a implementar. Soportan varios motores de bases de datos (como *MySQL, PostgreSQL, Microsoft SQL*). Doctrine 2 cuenta con una arquitectura más limpia y promueve la inyección de dependencia en lugar de acceder a las Doctrine_ * clases estáticamente o mediante un producto único como *Doctrine_Manager*. Los objetos y las bases de datos relacionales tienen diferentes mecanismos para estructurar los datos. Muchas partes de un objeto, como colecciones y la herencia, no están presentes en las bases de datos relacionales. Cuando se construye un modelo de objeto con una gran cantidad de lógica de negocio es valioso utilizar estos mecanismos para organizar mejor los datos y el comportamiento que va con ella (21).

Rendimiento(21)

- ✓ El uso de PHP 5 brinda un gran rendimiento.
- ✓ Mejor hidratación y algoritmo optimizado.
- ✓ Nuevas implementaciones de consulta y el resultado cache.
- ✓ El código más explícito y menos mágico, como resultados: el código mejor y más rápido.

El rendimiento en Doctrine 2 es muy superior a Doctrine 1. En la siguiente tabla se muestra el resultado de un estudio realizado a un sistema haciendo uso de dos versiones diferentes de Doctrine (Doctrine 1.2.2 y Doctrine 2.4.2), 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. (20).

Resultados: En la siguiente tabla se muestra el resultado de ambos ORM en sus diferentes versiones (20) .

Tabla.1 Comparación de rendimiento en las versiones de Doctrine (19).

Versión	Registros	Tiempo	Registros	Tiempo
Doctrine 1	5000	4,3 seg.	10000	7 seg.
Doctrine 2	5000	1,4 seg.	10000	3,5 seg.

Compatibilidad con el gestor de base de datos PostgreSQL

Doctrine 2 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.

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(21)

- ✓ No utilizar 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.

Resumen del estudio de los ORM

El estudio de las herramientas de mapeo objeto relacional permitió identificar las potencialidades de Doctrine 2. El uso de esta versión aprovecha todo el potencial de PHP 5.3, brinda un alto rendimiento y reduce susceptiblemente el código necesario para llevar a cabo las operaciones de persistencia y recuperación de objetos. Teniendo en cuenta estos elementos y las definiciones del expediente de arquitectura del proyecto SIPAC respecto a su tecnología base se determina realizar la migración a la versión 2 de Doctrine.

1.4 Tecnologías y herramientas para el desarrollo

El uso de tecnologías apropiadas y herramientas es uno de los factores de mayor importancia para una solución ágil. Los lenguajes, tecnologías y herramientas que a continuación se presentan están definidos en el documento Vista de entorno de desarrollo tecnológico del proyecto (22).

1.4.1 Subversión v1.6.6

Subversión es un sistema de control de versiones libre (código abierto). Es decir, maneja ficheros y directorios a través del tiempo; 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. En este sentido, muchas personas piensan en un sistema de control de versiones como una especie de "máquina del tiempo".(23)

Subversión puede acceder al repositorio a través de redes, permite la conexión de varios usuarios que se encuentran en distintas computadoras. 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.(23)

1.4.2 RapidSVN v0.12.0

RapidSVN es la interfaz gráfica de usuario para el Sistema de Control de Versiones Subversión escrito en C++. Cliente visual que utiliza las mejores características de los clientes de otras arquitecturas de control de revisiones. Es fácil para los nuevos usuarios y hace que los usuarios experimentados sean aún más productivo (24).

1.4.3 PostgreSQL v9.1

En el desarrollo de la investigación como sistema gestor de base de datos se va a emplear la versión 9.1 de PostreSQL. Es un sistema gestor de base de datos objeto-relacional, bajo licencia BSD (del inglés *Berkeley Software Distribution*). Se ejecuta en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos y Windows. Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios. Soporte nativo para lenguajes como PHP, C, C++, Perl y Python. Altamente adaptable a las necesidades del cliente. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (25).

1.4.4 NetBeans v7.4

La presente solución se desarrolla sobre el IDE de programación multiplataforma NetBeans 7.4. La plataforma de desarrollo NetBeans es un marco de trabajo de desarrollo que puede utilizarse para aplicaciones de escritorio en JAVA y también para aplicaciones web aceptando lenguajes como PHP, JavaFX, C / C ++, JavaScript. Es un proyecto de código abierto dedicado a proveer productos de desarrollo de software, desde el 2000 por el patrocinador Sun Microsystems hasta 2010 que se convirtió en subsidiaria de Oracle. Con una arquitectura base, basada en la plataforma de Java, los productos se desarrollan de forma rápida, eficaz y sencilla (26).

1.4.5 Apache v2.2

Se utiliza como servidor web Apache 2.2 pues es una tecnología gratuita de código abierto. Apache es un servidor web gratuito, poderoso. Ofrece un servicio estable y sencillo de mantener y configurar. Apache es un software libre y el servidor web más popular, entre sus características se destaca: es multiplataforma (aunque idealmente está preparado para funcionar bajo Linux), muy sencillo de configurar, es open-source (código abierto para sistemas operativos modernos incluyendo UNIX y Windows NT).Es una plataforma donde las instituciones crean sistemas seguros, confiables. Posee diversos módulos que permiten incorporarle nuevas funcionalidades (estos son muy simples de cargar) y es capaz de utilizar lenguajes como PHP, TCL, Python, entre otros (27).

1.4.6 Sauxe 2.0

Contiene un conjunto de componentes reutilizables 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, 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 (28).

1.4.7 Marco de trabajo Zend v1.11

Zend Framework es un marco de trabajo de código abierto para desarrollar aplicaciones web y servicios web con PHP5. En la implementación usa código orientado a objeto. Cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada, permite a los desarrolladores utilizar los componentes por separado. A menudo se refiere a este tipo de diseño como “uso a voluntad” (del inglés”use at will") (29)

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de Zend conforman un potente y extensible MT de aplicaciones web al combinarse. Este ofrece un gran rendimiento y una robusta implementación del MVC, una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos (30).

1.4.8 PHP v5.3.10

PHP 5.3 es el lenguaje que se empleará para programar del lado del servidor. PHP cuyas siglas responden a un acrónimo recursivo (PHP: Hypertext Preprocessor) es el lenguaje de código abierto más utilizado para la creación de páginas Web dinámicas y que puede ser embebido en páginas HTML. La razón de esta popularidad es el equilibrio entre su potencia y facilidad de uso. Es una excelente combinación de las mejores características de los lenguajes más populares de programación siendo así sencillo de aprender y contiene una completa colección de bibliotecas de funciones. Puede ser utilizado en casi todos los sistemas operativos existentes, permitiendo migrar las aplicaciones de un sistema a otro sin necesidad de realizar cambios en el código (31).

1.4.9 JMeter

ApacheJmeter se empleará en la validación, realizando pruebas de rendimiento (carga y estrés). ApacheJMeter es un software de código abierto. Fue diseñado originalmente para pruebas de aplicaciones Web pero se ha expandido a otras funciones de prueba. JMeter puede ser utilizado para

probar el rendimiento. Se puede utilizar para simular una carga pesada en un servidor, grupo de servidores (HTTP, HTTPS, FTP, TCP, otros.), la red o el objeto para probar su resistencia o para analizar el rendimiento general bajo diferentes tipos de carga. Se puede utilizar para hacer un análisis gráfico de rendimiento o para probar su comportamiento / objeto de servidor / script bajo carga pesada concurrente (32).

1.5 Conclusiones parciales

El estudio de las tecnologías y herramientas, ayudaron a sentar las bases para cumplir con el objetivo planteado.

El estudio y análisis de los principales ORM ayudó a comprender mejor el funcionamiento interno de Doctrine 2 y cómo este favorecía el rendimiento de las soluciones.

El estudio realizado del procedimiento para la migración de la capa de acceso a datos permitió sentar las bases para el desarrollo de la migración.

CAPÍTULO 2: DESCRIPCIÓN DE LA MIGRACIÓN

En este capítulo se explicará detalladamente como se realizó la migración de la capa de acceso a datos del componente Planeación del módulo Planificación del SIPAC. Se definen las principales acciones que se deben cumplir para realizar la migración de este componente a Doctrine 2, especificando la versión del marco de trabajo que posee, es decir, siempre que sea una versión inferior a la 2.0 se realiza el proceso de migración en cuatro fases: Inicio, Integración, Implementación y Pruebas, esta última se explicará detalladamente en el próximo capítulo .En la siguiente Figura se muestra el modelado de las fases para comprender mejor la operación.

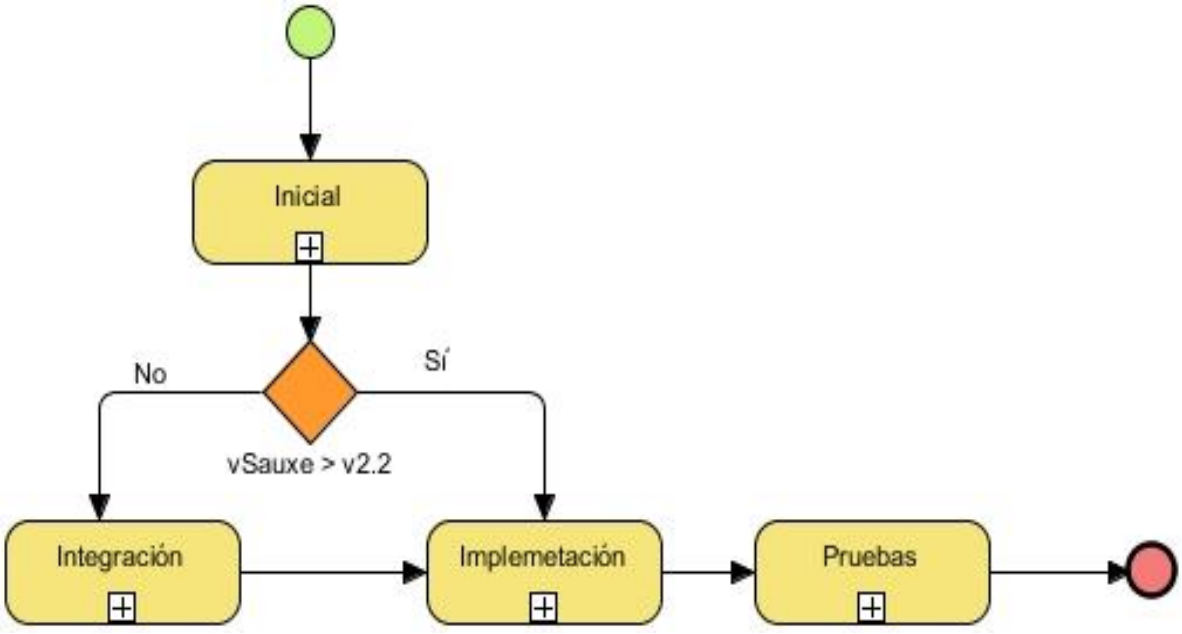


Figura.3 Fases del procedimiento (10).

2.1 Fase inicial

En la fase de inicio primeramente se define un cronograma para determinar cada una de las actividades a realizar durante todo el proceso de migración, se define el tiempo, costo y esfuerzo del procedimiento a realizar. Antes de comenzar el proceso se prepara el entorno de desarrollo, teniendo lista las herramientas y tecnologías a utilizar: marco de trabajo Sauxe en las dos versiones (1.2, 2.0), para facilitar el desarrollo se emplea el IDE Netbeans 7.4, permitiendo el almacenamiento, modificación y extracción de la información, se utiliza el sistema gestor de base de datos Postgresql 9.1, como servidor web HTTP para plataforma Linux, se utiliza el Servidor Apache 2.2, necesario para

el funcionamiento del marco de trabajo Sauxe. En la fase de inicio se procede a realizar un estudio de la arquitectura del MT Sauxe, a partir de la especificación de cada una de sus capas.

2.1.1 Arquitectura del marco de trabajo Sauxe

El MT Sauxe está desarrollado sobre una arquitectura en capas. A continuación se describen cada una de esas capas.

Capa de Presentación: En la capa de presentación Sauxe utiliza ExtJS, forma parte de una librería JavaScript. ExtJS facilita una gama de componentes que se pueden reutilizar. Centra su desarrollo en tres componentes JS-File, CSS-File y Client-page y Server-Page (28).

Capa de Control o Negocio: En esta capa se emplea el patrón de arquitectura Modelo Vista Controlador (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 (28).

Capa de Acceso a Dato: En esta capa estará presente el ORM Doctrine, como marco de trabajo de persistencia. También estará un Persistidor de Configuración que es el encargado de comunicarse vía XML con los Ficheros de Configuración del sistema denominado FastResponse (28).

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 (28).

Capa de Servicio: En esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí (28).

Esta gran estructura descrita, se comunica con varios servicios, que interactúan con el sistema proporcionándole un conjunto de funcionalidades tanto de seguridad como de negocio.

2.1.2 Estudio de la estructura de carpeta de la capa de acceso a datos del componente Planeación

La estructura de carpetas existente en la capa de datos sufre modificaciones durante la migración a Doctrine 2. En la siguiente Figura se muestra la estructura de carpeta de la capa de acceso a datos del componente Planeación para Doctrine 1.

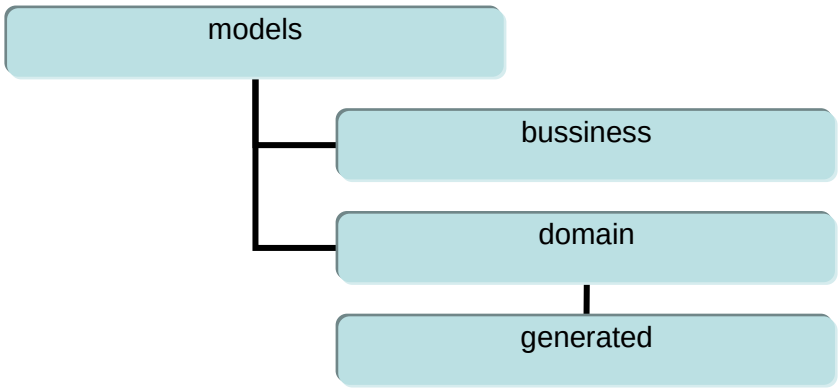


Figura.4 Estructura de carpeta de la capa de acceso a datos del componente Planeación para Doctrine 1.

2.2 Fase de integración

La actividad de integración es la base para realizar la implementación de la migración. Se definen las librerías y las clases que se modifican en la integración, para lograr correcta comunicación con el ORM Doctrine 2.4.2 y Zend Framework (5).

Para realizar la integración, inicialmente se:

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

2.2.1 Estudio de las librerías seleccionadas

Las librerías escogidas son las propuestas por Doctrine 2 (Common, DBAL y ORM); se puede apreciar su ubicación en la siguiente Figura**.** A través de un estudio realizado, se dio a conocer de cada una de las librerías:

Common: contiene código reutilizable como el analizador de anotaciones y el sistema de eventos.

DBAL: abstrae no solo el controlador particular de base de datos, al igual que ya lo hace de forma nativa PDO[1], sino también los dialectos SQL de diferentes Sistemas de Gestión de Bases de Datos.

Mapa de Doctirne\ORM: gráficos de objetos en base de datos y viceversa.

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

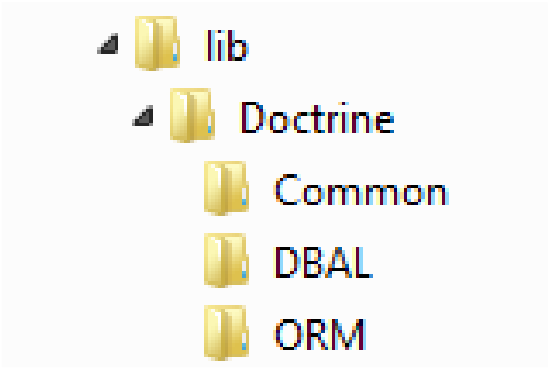


Figura.5 Ubicación de las nuevas librerías de Doctrine.

2.2.2 Estudio de las clases seleccionadas

En la integración de Doctrine 2 con el MT Sauxe es necesario la creación y configuración del fichero Doctrine.php. Para la ubicación de dicho fichero se crea en lib/ZendExt/App/ la carpeta Resource y dentro de esta se implementa la clase ZendExt_App_Resource_Doctrine como muestra la siguiente figura.

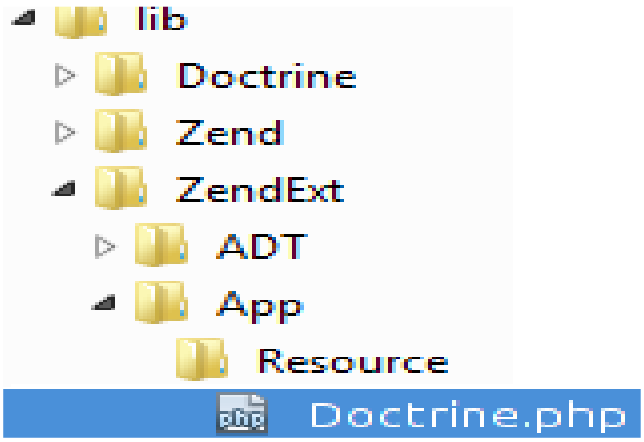


Figura.6 Clase seleccionada para la integración.

Para que la clase creada sea reutilizable en todas las aplicaciones dentro del MT Sauxe se debe crear como un recurso del MT Zend es por eso que la misma debe extender de **Zend_Application_Resource_ResourceAbstract**. La función principal de este recurso será la configuración de los parámetros necesarios para crear una instancia del **EntityManager** logrando el acceso a las funcionalidades del marco de persistencia de datos de Doctrine.

```
class ZendExt_App_Resource_Doctrine
extends Zend_Application_Resource_ResourceAbstract {

    public function init() {

        $options = Zend_Registry::get('configdoct');


        $classLoader = new \Doctrine\Common\ClassLoader('Doctrine');
        $classLoader->register();

        $classLoader = new \Doctrine\Common\ClassLoader(
                        $options['entitiesPathNamespace'],
                        $options['entitiesPath']
        );
        $classLoader->register();

        $classLoader = new \Doctrine\Common\ClassLoader(
                        $options['proxiesPathNamespace'],
                        $options['proxiesPath']
        );
        $classLoader->register();
        $classLoader = new \Doctrine\Common\ClassLoader(
                        $options['repositoriesPathNamespace'],
                        $options['repositoriesPath']
        );
        $classLoader->register();
```

```
    $cacheClass = 'Doctrine\Common\Cache\ArrayCache';

    $cache = new $cacheClass();
    $config = new \Doctrine\ORM\Configuration();
    $config->setMetadataCacheImpl($cache);
    $driverImpl = $config->newDefaultAnnotationDriver($options['module']);
    $config->setMetadataDriverImpl($driverImpl);
    $config->setQueryCacheImpl($cache);

    $config->setProxyDir($options['proxiesPath']);
    $config->setProxyNamespace($options['proxiesPathNamespace']);
    $config->setAutoGenerateProxyClasses($options['autoGenerateProxyClasses']);

    $em = \Doctrine\ORM\EntityManager::create(
                    $options['conection'], $config
    );

    Zend_Registry::set('em', $em);
    return $em;

}
```

Para la correcta configuración de Doctrine 2 será necesario ubicar en el fichero **apps/config.php** los parámetros de configuración y guardarlos para su posterior uso.

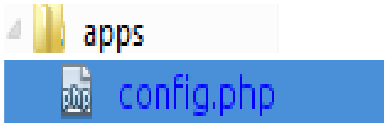


Figura.7 Ubicación del fichero de configuración.

Estos parámetros serán usados por Doctrine 2 para configurar los directorios de las entidades y los namespaces.

```
$config['configdoct']['driver'] = "pdo_pgsql";
$config['configdoct']['entitiesPath'] = $_SERVER['DOCUMENT_ROOT'] . "/../apps/";
$config['configdoct']['entitiesPathNamespace'] = "";
$config['configdoct']['proxiesPath'] = $_SERVER['DOCUMENT_ROOT'] . "/../apps/";
$config['configdoct']['proxiesPathNamespace'] = "";
$config['configdoct']['repositoriesPath'] = $_SERVER['DOCUMENT_ROOT'] . "/../apps";
$config['configdoct']['repositoriesPathNamespace'] = "";
$config['configdoct']['cacheClass'] = "Doctrine\Common\Cache\ApcCache";
$config['configdoct']['autoGenerateProxyClasses'] = true;
$config['configdoct']['Path'] = $_SERVER['DOCUMENT_ROOT'] . "/../lib/ZendExt/";
```

Una vez creado el recurso y ubicados los parámetros de configuración de Doctrine 2 será necesario modificar la funcionalidad OpenConnections () de la clase ZendExt_Aspect_TransactionManager.php, encargada de configurar y manejar las conexiones del marco de persistencia de datos de Doctrine en el MT Sauxe.

En la siguiente Figura se muestra una imagen con la ubicación de dicha clase.

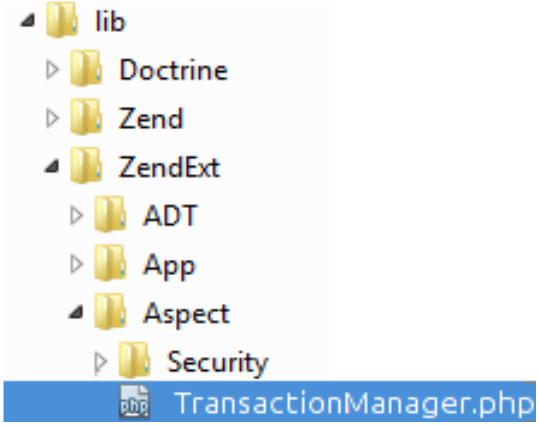


Figura.8 Ubicación de la clase TransactionManager.

Para lograr la configuración de Doctrine 2 es necesario modificar el método openConections () para hacer uso de los parámetros definidos en el fichero config.php.

A continuación se hace mención a los nuevos parámetros de configuración que se utilizarán:

- **driver**: tipo de controlador que implementan los PDO de interfaz para permitir el acceso de PHP a las bases de datos. Debido a que el MT Sauxe utiliza PostgreSQL como gestor de base de datos se debe usar *pdo_pgsql*.
- **entitiesPath:** ruta en la que se encontrarán las entidades del mapeo.
- **entitiesPathNamespace:** nombre del *namespace* que deben asociado a la ruta de las entidades del mapeo.
- **proxiesPath:** ruta en la que se encontrarán las clases *proxy* pertenecientes a las entidades del mapeo.
- **proxiesPathNamespace:** nombre del *namespace* de las clases *proxy*.
- **repositoriesPath:** ruta en las que se encontrarán las clases *repository* pertenecientes a las entidades del mapeo.
- **repositoriesPathNamespace:** nombre del *namespace* de las clases *repository*.
- **cacheClass:** tipo de cache que utilizará el ORM Doctrine, en este caso: *Doctrine\Common\Cache\ApcCache.*
- **autoGenerateProxyClasses:** parámetro de tipo boolean para indicar si se generan o no de manera automática las clases *proxy.*

De manera general es válido argumentar que es imprescindible el uso de los namespaces debido a que estos son usados por Doctrine 2 para evitar que las entidades extiendan de una clase base como en Doctrine 1.

Después de crear y configurar los parámetros necesarios se inicializa la funcionalidad de Doctrine haciendo una llamada al recurso anteriormente creado (ZendExt_App_Resource_Doctrine::init()). Seguidamente se podrá acceder a las funcionalidades del marco de persistencia de datos de Doctrine.

En la siguiente Figura se muestra la funcionalidad **OpenConections** una vez adaptada a la nueva estructura de doctrine**:**

```php
public function openConections($module = null, $current = false) {
    try {
        $config = Zend_Registry::getInstance()->config->configdoct;
        $config->entitiesPathNamespace = $module . "\models";
        $config->proxiesPath = $config->proxiesPath . $module . "/models/Proxies";
        $config->proxiesPathNamespace = $module . "\models\Proxies";
        $config->repositoriesPathNamespace = $module . "\models\Repositories";
        if ($module == "Trace") {
            $config->proxiesPath = $config->Path . $module . "/models/Proxies";
            $config->entitiesPath = $config->Path;
            $config->repositoriesPath = $config->Path;
        }
        $configdoct = $config->toArray();
        $modulesconfig = ZendExt_FastResponse::getXML('modulesconfig');
        $bdconfig = $modulesconfig->conn;
        $configdoct['conection']['driver'] = 'pdo_' . $bdconfig['gestor'];
        $configdoct['conection']['user'] = $bdconfig['usuario'];
        $RSA = new ZendExt_RSA();
        $configdoct['conection']['password'] = $RSA->decrypt($bdconfig['password'],
                                               '855506942851452308 23', '9980914335265034 1179');
        $configdoct['conection']['host'] = $bdconfig['host'];
        $configdoct['conection']['port'] = $bdconfig['port'];
        $configdoct['conection']['dbname'] = $bdconfig['bd'];
        $configdoct['module'] = $config->entitiesPath . "$module/models";
        Zend_Registry::set('configdoct', $configdoct);
        ZendExt_App_Resource_Doctrine::init();
        $this->conection = Zend_Registry::get('em');
    } catch (Exception $e) {
        throw new ZendExt_Exception('E014', $e, null);
    }
    return $this->conection;
}
```

Figura.9 Método OpenConections () de la clase TransactionManager.

2.3 Fase de implementación

En la fase de implementación se realiza el rediseño de la arquitectura base del componente Planeación del módulo Planificación del SIPAC. Se debe de establecer una nueva estructura de carpetas y archivos que requiere Doctrine 2. La CAD se debe rediseñar y elaborar nuevamente los métodos y las consultas. Igualmente, realizar el mapeo de las tablas y relaciones de la base de datos.

2.3.1 Redefinir capa de acceso a datos.

Debido a la diferencia existente entre las versiones 1 y 2 de Doctrine, y el rediseño de la CAD, se creó una nueva estructura de carpetas. También, se realizó un estudio del mapeo de las tablas y las relaciones con la base de datos para adaptarlas a la nueva versión de Doctrine 2. La migración manual de cada una de las clases ya mapeadas, es una tarea que requiere de tiempo para su

realización, más allá de su complejidad, por el monto de objetos que se deben mapear y por la cantidad de funcionalidades que se implementan nuevamente.

2.3.2 Estructura de carpeta

Con la nueva versión de Doctrine 2 surge una nueva estructura de carpetas base, muy similar a la versión de Doctrine 1.x. con la diferencia que incorpora tres carpetas nuevas (*Repositoy, Proxy, Entity*). En la siguiente Figura se evidencian las relaciones entre la estructura de carpetas propuestas por las versiones 1.x y 2.x de Doctrine.

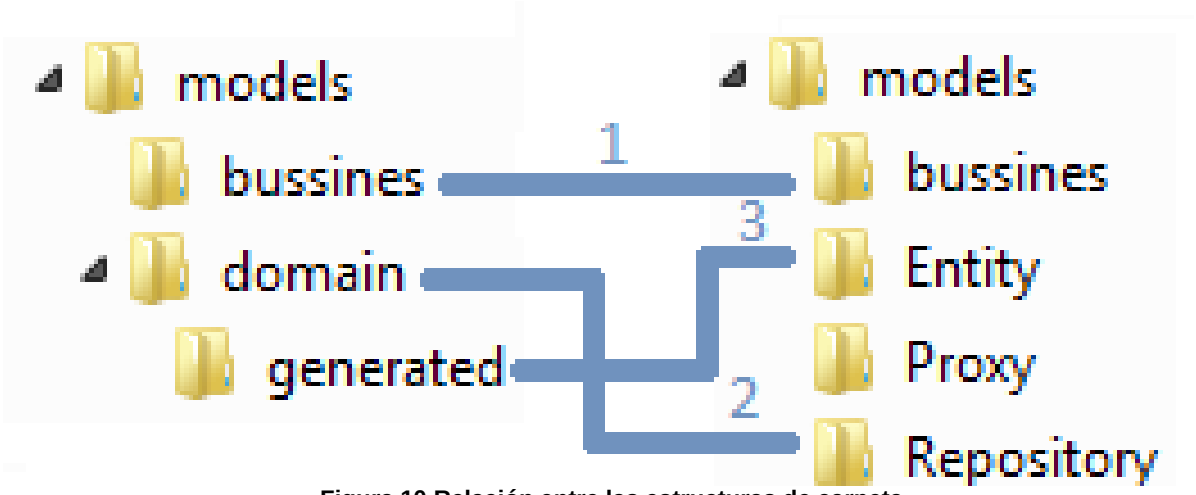


Figura.10 Relación entre las estructuras de carpeta.

1 Los ficheros localizados en la carpeta *bussines* que son los encargados de manejar el negocio de la capa de acceso a datos se mantuvieron en este directorio, aunque fueron modificados, ya que en Doctrine 1 muchas de las consultas se implementaban en las clases *model*, ahora en Doctrine 2, estas consultas se acceden a través de los repositorios, siendo el *EntityManager* la instancia que lo facilita. Por lo que hubo que reimplementar todos los métodos.

2 Con la nueva estructura de Doctrine 2 se crea un directorio llamado *Repository*, directorio en el cual se encuentran los repositorios de las entidades, se generó un repositorio por cada entidad. Todas las funcionalidades con las consultas que se encontraban localizados en la carpeta Domain en Doctrine 1, con la nueva estructura de Doctrine 2 tuvieron que ser modificadas y trasladadas para el directorio *Repository*, su nomenclatura está compuesta por el nombre del fichero de mapeo de la tabla y el sufijo *Repository*. Estos ficheros son clases que extienden de la clase Doctrine\ORM*EntityRepository***.**

3 En Doctrine 1 los ficheros que representan las tablas mapeadas en la carpeta *generated*, se tuvieron que generar con la nueva estructura de Doctrine 2 en el directorio *Entity*.

4 En la carpeta *Proxy* se ubican los ficheros generados como resultado del mapeo que contienen las clases *Proxy* usadas por Doctrine 2. 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.
La siguiente tabla muestra las funcionalidades migradas.

Tabla.2 Clases generadas por cada tabla.

Tabla BD	Directorios		
	Entity	***Repository***	***Bussines***
dat_actividades	DatActividades	DatActividadesRepository	CmpgestionaractividadesModel
dat_arc	DatArc	DatArcRepository	DatArcModel
dat_arc_dat_involucrados	DatArcDatInvolucrados	DatArcDatInvolucradosRepository	DatArcDatInvolucradosModel
dat_arc_dat_plan	DatArcDatPlan	DatArcDatPlanRepository	DatArcDatPlanModel
dat_arc_dat_plan_actividades	DatArcDatPlanActividades	DatArcDatPlanActividadesRepository	
dat_arc_dat_plan_objetivo	DatArcDatPlanObjetivos	DatArcDatPlanObjetivoRepository	DatArcDatPlanObjetivoModel
dat_elementos	DatElementos	DatElementosRepository	DatElementosModel
dat_elementos_dat_elementos	DatElementosDatElementos	DatElementosDatElementosRepository	DatElementosDatElementosModel
	DatElementosDatInvolucrados		DatElementosDatInvolucradosModel
	DatElementosNomNomenclador	DatElementosNomNomencladorRepository	DatElementosNomNomencladorModel
dat_fip	DatFip	DatFipRepository	DatFipModel
	DatInvolucrados	DatInvolucradosRepository	DatInvolucradosModel
dat_objetivo	DatObjetivo	DatObjetivoRepository	DatObjetivoModel
dat_observaciones_pdo	DatObservacionesPdo	DatObservacionesPdoRepository	DatObservacionesPdoModel
dat_plan_pdo	DatPlanPdo	DatPlanPdoRepository	DatPlanPdoModel
	NomNivelinvolucrado	NomNivelinvolucradoRepository	NomNivelinvolucradoModel
	NomNomenclador	NomNomencladorRepository	
nom_tipoelemento	NomTipoelemento	NomTipoelementoRepository	NomTipoelementoModel

2.3.3 Flujo de dato de la capa de acceso a datos

La capa de acceso a datos con la nueva versión de Doctrine adquiere otro funcionamiento interno. Este transita desde las clases controladoras que invoca a una funcionalidad en las clases *model,* en el cual se encuentran implementados los métodos que manejan el negocio del componente Planeación, hasta los ficheros *Repository*. 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 utilizaba las clases *model* para realizar peticiones directamente a la base de datos, en su lugar Doctrine 2, utiliza una instancia del *EntityManager*, donde puede acceder a los repositorios.

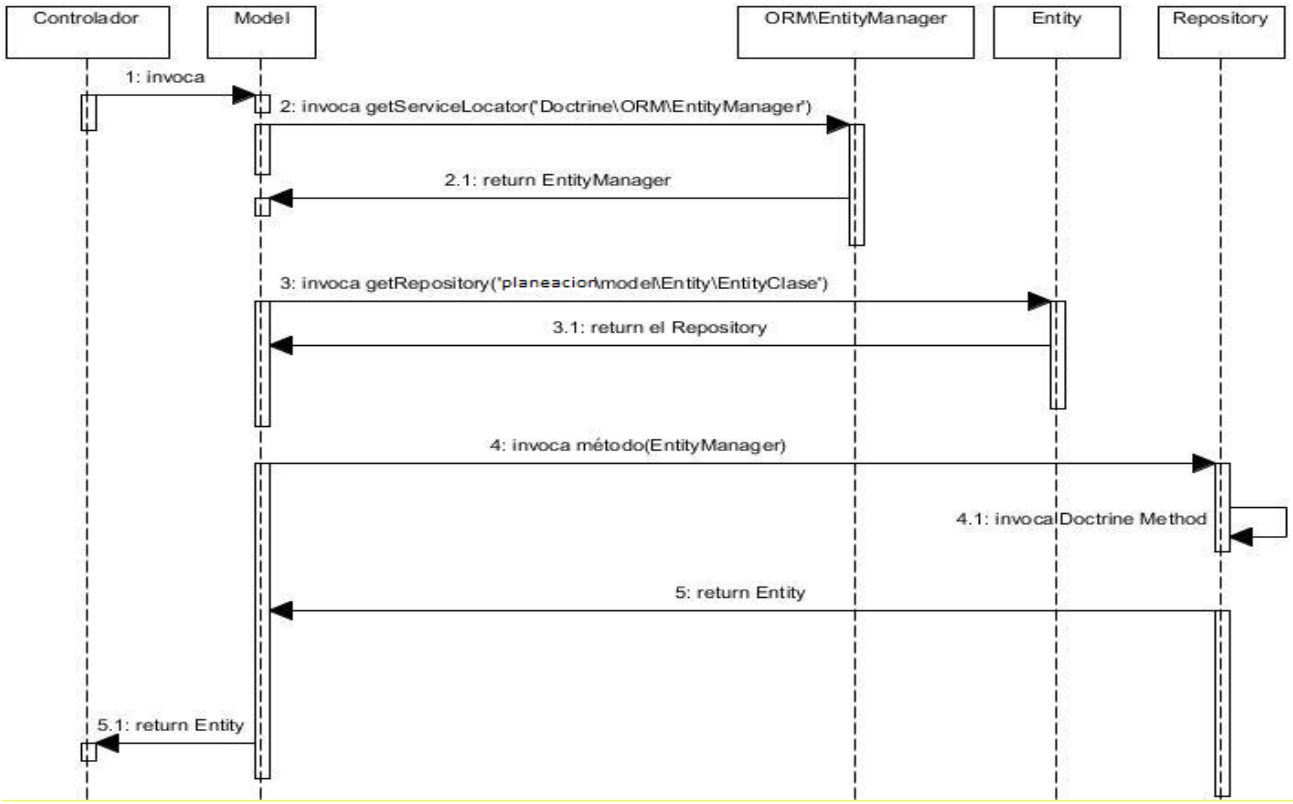


Figura.11 Flujo de la CAD.

2.3.4 Mapeo de las tablas

El uso de mapeo para la persistencia de datos mediante el marco de trabajo Doctrine 1 es muy diferente a Doctrine 2. Doctrine 1 genera entidades que extienden de una clase base, haciendo uso de tablas y columnas de funciones predefinidas. Doctrine 2 proporciona varias formas para la realización del mapeo:

- ✓ Uso de archivo XML.

- ✓ Uso de archivo YAML.
- ✓ Anotaciones Docblock.

Doctrine proporciona varias formas para realizar el mapeo objeto relacional a través de asignación básica de objetos y propiedades. La especificación de estas asignaciones de datos se realiza con el uso de archivos XML, YAML o Anotaciones Docblock. Todos ellos tienen exactamente el mismo rendimiento en cuanto a su empleo, quedando a elección del programador cuál seleccionar. Para el desarrollo de la migración se va a emplear Anotaciones Docblock por incluirse como forma de comentarios en archivos de PHP. A diferencia de los comentarios normales, las anotaciones pueden influir en la ejecución del código y propiciar una mayor flexibilidad para modificar el comportamiento de la aplicación desde los archivos en PHP. El uso de estos archivos para la generación de los metadatos.

Las anotaciones Docblock es una herramienta para incrustar metadatos dentro de la sección de documentación que luego, alguna herramienta puede procesar. Doctrine 2 generaliza el concepto de anotaciones para que se puedan utilizar en cualquier tipo de metadatos y así facilitar la definición de nuevas Anotaciones Docblock. Las anotaciones son utilizadas con el objetivo de representar la estructura de las entidades de las clases de acceso a datos para Doctrine, permitiendo la realización del mapeo objeto relacional que consiste básicamente en el trabajo con entidades o clases persistentes que representan las tablas con sus campos o columnas.

Para la representación de una entidad se utilizan anotaciones específicas que permiten la definición, identificadores, asignación de los tipos de datos, asociación o relación entre entidades con cardinalidad y direccionalidad que pueden ser de tipo unidireccional o bidireccional. Estos son algunos de los elementos necesarios para la conformación de las entidades o clases de acceso a datos (33).

Tabla.3 Descripción de las anotaciones utilizadas en Doctrine 2 (33).

Anotación	Descripción	Atributos	Descripción
@Table	Configura las características de la tabla.		
@Entity	Identifica a una clase PHP como entidad.	*repositoryClass*	Especifica la dirección en donde se encuentra la clase repositorio
@Columna	Identifica a una variable o columna como persistente.	*type*	Tipo de dato de la variable.
		name	Nombre de la columna de la base de datos
		lenght	Utilizado por el tipo de dato "*String*", para determinar su máxima longitud en la base de datos.
		precisión	Utilizado por el tipo de dato "*String*", para determinar su máxima longitud en la base de datos.
		unique	Determinar si el valor de la columna debe ser único en todas las filas de la tabla.
		nullable	Determina si el valor *NULL* es permitido para la columna donde se utilice.
@Id	Indica a una columna como identificador único de la entidad, es la clave principal en la base de datos.		
@GeneratdValue	Especifica la estrategia	*strategy*	Establece el tipo de estrategia entre los

	utilizada para la generación del identificador de una variable que es identificada con la anotación @Id.		valores validos se encuentran: AUTO: (Opción predeterminada) Ofrece una portabilidad completa al indicarle a Doctrine la estrategia preferida por la plataforma de BD utilizada. SEQUENCE: Le dice a Doctrine que utilice una secuencia de BD para la generación de id. Compatible solo con Oracle y PostgreSql. IDENTITY: Le dice a Doctrine que debe utilizar columnas de identidad especiales en la base de datos que generen un valor al insertar una fila. Compatible solo con MySQL/SQLite (AUTO_INCREMENT), MSSQL (IDENTITY) y PostgreSQL (SERIAL). *TABLE*: Esta estrategia no se encuentra implementada todavía por Doctrine. *NONE*: Es lo mismo que omitir el *@GeneratedValue.*
@SequenceGenerator	Permite especificar detalles acerca de la secuencia.	*sequenceName*	Nombre de la secuencia.
		allocationSize	Utilizado para incrementar el tamaño de la secuencia de asignación.
		initialValue	Utilizado para indicar el valor inicial de la secuencia. Por defecto se inicia con valor 1.
@OneToOne *@OneToMany* *@ManyToOne*	Indican las relaciones que son utilizadas entre dos entidades	*targetEntity*	Determina la entidad que es objeto de referencia.
		inversedBy	Designa el campo en la entidad que es el lado inverso de la relación.

@JoinColumn	Utilizada por las relaciones *@ManyToOne y @OneToOne*	*name*	Nombre de la columna que contiene el identificador de clave externa para esta relación.
		referencedColumnName	Nombre del identificador de clave principal que se utiliza para la unión de esta relación.

Definición de tabla y columna

Doctrine 1.x crea una clase que extiende de una base llamada Doctrine_Record con una funcionalidad además llamada setTableDefinition. Por su parte Doctrine 2.x, a partir del nuevo directorio, crea una carpeta llamada *Entity*, donde almacena todas las entidades a través de la anotación *@Table* y *@Entity* para indicar que es una entidad.

Doctrine 1

```
abstract class BaseDatArc extends Doctrine_Record
 {
   public function setTableDefinition()
     {
        $this->setTableName('dat_arc');

     }
 } ....
```

Figura.12 Definición tabla y columna para Doctrine 1.

Doctrine 2

```
/**
 * DatArc
 *
 * @ORM\Table(name="mod_objetivos.dat_arc", uniqueConstraints={@ORM\UniqueConstraint(name="k_arc_de
 * @ORM\Entity(repositoryClass="planificacion\pdo\plan\models\Repository\DatArcRepository")
 */
```

Figura.13 Definición tabla y columna para Doctrine 2.

Doctrine 1 permite representar las columnas con la función hasColumn (). Doctrine 2 lo hace similar pero utilizando anotaciones.

Doctrine 1

```
$this->hasColumn('idarc', 'numeric', null, array('notnull' => true, 'primary' => true/*,
'sequence' => 'sec_datarc'*/));
```

Figura.14 Definición tabla y columna utilizando función hasColumn ().

Doctrine 2

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

Figura.15 Definición tabla y columna utilizando anotaciones.

Representación de las relaciones entre tablas

La representación de las relaciones 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, se realiza mediante 2 ficheros de mapeo, el 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 2

```
class DatArcDatPlan
{
    /**
     * @var \DatPlanPdo
     * @ORM\ManyToOne(targetEntity="DatPlanPdo", mappedBy="idarc")
     * @ORM\JoinColumn(name="idplan", referencedColumnName="idplan")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="NONE")
     */
    private $datPlanPdo;
```

Figura.16 Ejemplo de relación ManytoOne.

La relación @ManyToOne señala que se tiene una única instancia del objeto mapeado por la entidad DatPlanPdo pero que un DatPlanPdo puede ser usado por muchos a la vez. Utiliza una anotación

obligatoria @JoinColumn. El atributo mappedBy se usa en el lado inverso de una relación bidireccional y designa el campo en la entidad, que es el propietario de la relación.
En el caso en que la relación es de Mucho-Mucho Doctrine 1 utiliza la siguiente manera:

```
public function setUp() {
    parent::setUp();
    $this->hasMany('DatInvolucrados', array('local' => 'idarc', 'foreign' => 'idinvolucrado', 'refClass' => 'DatArcDatInvolucrados'));

}
```

Figura.17 Ejemplo de relación en Doctrine 1.

En la Figura se muestra el mapeo de la clase DatArc, en la misma se puede apreciar la representación de la relación de esta clase con DatArcDatInvolucrados.

Doctrine 2

En este caso en Doctrine 2 cuando se realiza el mapeo, se hace referencia a la entidad de mapeo con que se posee la relación mediante la propiedad *targetEntity* de la anotación *@ORM\ManytoMany* y mediante la anotación *@JoinTable* se le dice cuál es la tabla a través de la cual se relacionan. 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* definido por Doctrine.

```
/**
 * @var \Doctrine\Common\Collections\Collection
 *
 * @ORM\ManyToMany(targetEntity="DatInvolucrados", inversedBy="idarc")
 * @ORM\JoinTable(name="mod_objetivos.dat_arc_dat_involucrados",
 *   joinColumns={
 *     @ORM\JoinColumn(name="idarc", referencedColumnName="idarc")
 *   },
 *   inverseJoinColumns={
 *     @ORM\JoinColumn(name="idinvolucrado", referencedColumnName="idinvolucrado")
 *   }
 * )
 */
```

Figura.18 Ejemplo nueva forma de mapeo en Doctrine 2.

El mapeo objeto relacional descrito anteriormente, ha representado la estrategia que adopta Doctrine en las dos versiones.

2.3.5 Consultas y funcionalidades en los Repository

Dentro de la CAD, como se mencionó anteriormente, todas las consultas y funcionalidades serán re implementadas. En la versión 1 de Doctrine estas consultas quedaban en la carpeta *domain* y utilizaban el DQL de Doctrine 1. Con la versión de Doctrine 2 una vez mapeadas las tablas de la base de datos se procede a modificar las funcionalidades y consultas; estas funcionalidades se deben ubicar en los ficheros dentro de la carpeta *Repositories*. Las funcionalidades en Doctrine 1 quedaban de la manera que se muestra.

```
public function GetActvPorUsuarioTipoNomenclador($idusuariocrea, $tiponom, $fechainicio, $fechafin) {
    $query = new Doctrine_Query ();
    $result = $query->select("*")
            ->from('DatElementos elem,DatActividades act, DatVariable var')
            ->innerJoin('elem.DatActividades act')
            ->innerJoin('act.DatVariable var')
            ->innerJoin('elem.DatElementosNomNomenclador elemnom')
            ->where("elem.idusuariocrea = $idusuariocrea")
            ->addWhere("var.fechainicio >= $fechainicio")
            ->addWhere("var.fechafin <= $fechafin")
            ->addWhere("elemnom.tiponomenclador = $tiponom")
            ->execute();
    return $result->toArray();
}
```

Figura.19 Ejemplo de consulta en Doctrine 1.

En Doctrine 2 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)*.

```
public function GetActvPorUsuarioTipoNomenclador($idusuariocrea, $tiponom, $fechainicio, $fechafin, $_em)
{
    $qb = $_em->createQueryBuilder();
    $p = $qb->select("*")
        ->from('planificacion\pdo\plan\models\Entity\DatElementos elem,planificacion\pdo\plan\models\Entity\DatActividades act,
        DatVariable var')
        ->innerJoin('elem.planificacion\pdo\plan\models\Entity\DatActividades', 'act')
        ->innerJoin('act.planificacion\pdo\plan\models\Entity\DatVariable', 'var')
        ->innerJoin('elem.planificacion\pdo\plan\models\Entity\DatElementosNomNomenclador', 'elemnom')
        ->where("elem.idusuariocrea = ?1")
        ->addWhere("var.fechainicio >= ?2")
        ->addWhere("var.fechafin <= ?3")
        ->addWhere("elemnom.tiponomenclador = ?4")
        ->setParameter(1, $idusuariocrea)
        ->setParameter(2, $fechainicio)
        ->setParameter(3, $fechafin)
        ->setParameter(4, $tiponom)
        ->getQuery();
    $result = $p->getArrayResult();
    return $result;
}
```

Figura.20 Ejemplo de consulta en Doctrine 2.

Algunas variaciones en las consultas

- ✓ *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 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), que es en gran medida intercambiable con los objetos gráficos generados por Query#getResult () de solo lectura.
- **getScalarResult *():*** Recupera un resultado plano/rectangular del conjunto de valores escalares que puede contener datos duplicados. La distinción puro/mixto no aplica.
- **getSingleScalarResult ():** Recupera un valor escalar único a partir del resultado devuelto por el SGBDR. Si el resultado contiene más de un valor escalar único, lanza una excepción. La distinción puro/mixto no aplica.

2.3.6 Redefinir capa del 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. Fueron eliminadas algunas de 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 *Models* encargados del negocio.

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, 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 GetPorLimiteNomNivelinvolucradoAction(){
    $mg = ZendExt_Aspect_TransactionManager::getInstance();
    $_em = $mg->getConnection("planificacion/pdo/plan");
    $result = $_em->getRepository("planificacion\pdo\plan\models\Entity\NomNivelinvolucrado")
                  ->GetPorLimite($limite = 10, $inicio = 0, $_em);
    print_r($result);
}
```

2.3.7 Funcionalidades de 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, 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 CmpgestionaractividadesModel extends ZendExt_Model {

    public function CmpgestionaractividadesModel() {
        parent::ZendExt_Model();
    }
```

Figura.21 Estructura de la clase Model.

Doctrine 1:

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

```
public function Insertar(DatArcDatInvolucrados $DatArcDatInvolucrados)
 {
        $DatArcDatInvolucrados->save();
 }
```

Figura.22 Ejemplo de la Model en Doctrine 1.

Doctrine 2

En Doctrine 2 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 *EntityManage*r.

Una vez hecho esto, es posible acceder al Repositor*y* de determinada entidad utilizando la funcionalidad *getRepositor*y pasándole como parámetro el *namespace* de la clase que se quiere obtener el repositorio y finalmente se solicita la funcionalidad deseada.

```
public function ActualizaEstadoCumplimiento($idelemento, $estadocumplimiento, $porciento)
{
    $mg = ZendExt_Aspect_TransactionManager::getInstance();
    $_em = $mg->getConnection("planificacion/pdo/plan");

    $estadocumplimiento = $this->actualizarEstadoCumplimiento($estadocumplimiento, $porciento);
    $result = $_em->getRepository("planificacion\pdo\plan\models\Entity\DatElementos")
        ->ActualizarEstadoCumplimiento($idelemento, $estadocumplimiento, $porciento, $_em);
    return $result;
}
```

Figura.23 Ejemplo de llamado a las funcionalidades de la clase Model.

2.3.8 Actualizar estado de los elementos de configuración

Para actualizar el estado de los elementos de la configuración es necesario realizar la actualización en el repositorio de la aplicación. Actualizar los cambios a los elementos de Configuración e integrarlos. Finalmente se deben publicar los elementos actualizados.

2.4 Conclusiones parciales

Según la guía propuesta: PROCEDIMIENTO PARA ACTUALIZACIÓN DE LA CAPA DE ACCESO A DATOS DEL MARCO DE TRABAJO SAUXE DE DOCTRINE 1.2.2 A DOCTRINE 2.2, se realizó la migración de la capa de acceso a datos del componente Planeación del módulo Planificación del SIPAC, para entender mejor las modificaciones realizadas a la capa de acceso a datos, se ejecutaron las tres primeras fases del procedimiento antes mencionado. A partir de la nueva estructura de carpetas y el rediseño de la capa de acceso a datos se eliminaron malas prácticas y se logró una mejor organización en el código.

CAPÍTULO 3: VALIDACIÓN DE LA MIGRACIÓN

3.1 Introducción

Como parte del desarrollo de la migración es de vital importancia la validación, mediante el uso de la herramienta JMeter que permite evidenciar el rendimiento de la aplicación. Para alcanzar dicho propósito 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. Con la realización de las pruebas de rendimiento se podrá conocer los diferentes criterios y las estadísticas de los procesos que se ejecutan durante su utilización.

3.2 Pruebas de rendimiento

Como parte del cumplimiento del procedimiento de la migración se realizan pruebas de rendimiento. Las cuales se centran en determinar si el sistema satisface sus requerimientos así como los problemas y defectos. Las pruebas de rendimiento implican estresar el sistema, realizando demandas que están fuera de los límites del diseño del software (34).

Técnicamente “pruebas de rendimiento” se refiere al proceso de pruebas para determinar el rendimiento de un producto software (35). Este hace referencia al estudio del comportamiento de un sistema (aplicación software, componente o una plataforma hardware) cuando se ve sometido a una carga (indicador sobre el sistema) que actúa de manera concurrente (acciones funcionales realizadas en un período de tiempo, procesos ejecutados de manera simultánea o usuarios validados en la aplicación en un momento determinado). Si el sistema no es sometido a carga su comportamiento no se puede conocer, pues las pruebas funcionales no incluyen niveles altos de concurrencia (35).

Las pruebas de rendimiento se deben de 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) (35).

Existen varios tipos de pruebas de rendimientos de los cuales los más mencionados son: (35)

Pruebas de carga:

Son la forma 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. Intentarán validar que se alcanzan los objetivos de prestaciones a los que se verá sometido el sistema en un entorno productivo

Pruebas de capacidad:
Buscan dar una estimación de hasta dónde se puede llegar cargando el sistema antes de que sea inutilizable. Su objetivo además de encontrar los límites de funcionamiento del sistema, es detectar el cuello de botella o elemento limitante para poder actuar en caso de ampliación de un servicio

Pruebas de estrés:
Se utilizan normalmente para comprender los límites superiores de la capacidad dentro del sistema. Se aplican para determinar la robustez del sistema en términos de carga extrema y ayudan a los administradores de la aplicación para determinar si el sistema funcionará suficientemente si la corriente de carga va muy por encima del máximo esperado
Las pruebas de rendimiento realizadas al componente Planeación tuvieron como propósito demostrar que el sistema cumple con distintos criterios de rendimiento. Por lo cual se compararon las funcionalidades tanto con Doctrine 1 como Doctrine 2. Utilizando herramientas como JMeter, que permiten diagnosticar que partes del software tienen un bajo rendimiento o establecer niveles de tiempo de respuesta que sean aceptables.

Otras pruebas realizadas

Además de las pruebas con la herramienta JMeter se realizaron las siguientes pruebas:
-Verificar la conectividad a la Base de Datos: En esta prueba se verifica la correcta configuración de los parámetros de conexión a la base de dato de Doctrine 2.
-Verificar la correcta configuración de los namespaces: En esta prueba se acredita la correcta configuración de los namespaces mediante la llamada a entidades con namespaces existentes.
-Verificar la correspondencia entre el modelo relacional y el modelo objetual: En esta prueba se verifica que tanto los mapeos como las clases generadas en Doctrine 2 se corresponden con la estructura existente de la base de datos.
-Verificar correcto funcionamiento de los repositorios: En estas pruebas se verifica que el código generado para la realización de las consultas en Doctrine 2 tenía total equivalencia a las consultas realizadas en Doctrine 1, por lo tanto abala el correcto funcionamiento de la migración de la capa de acceso a datos del componente Planeación. Para lo cual se realizaron 255 pruebas de funcionalidad con su respectivo modelo de dato, a las 255 funcionalidades de los repositorios.

Las pruebas realizadas al componente Planeación después de haber sido migrado, permitieron verificar el correcto funcionamiento de sus funcionalidades, así como la conectividad a la base, la

correcta configuración de los namespaces y la relación entre el modelo relacional de la base de dato y el modelo orientado a objeto de Doctrine 2.

3.3 Entorno de prueba

El componente Planeación depende de otros componentes y módulos como son Seguridad y Configuración. Este último no se encuentra migrado a Doctrine 2 motivo por el cual el módulo Planificación no se encuentra migrado en su totalidad. Esto unido a que el componente Planeación no cuenta con una interfaz única para su gestión, imposibilita las pruebas a través de una interfaz de usuario. Para realizar las pruebas de rendimiento con la herramienta JMeter y demostrar el tiempo de acceso a las consultas, se crea una clase controladora que llame directamente a cada método del repositorio. Esta clase controladora utiliza una instancia del objeto *EntityManager*, encargado de facilitar el acceso a los repositorios donde se implementan todas las consultas a la base de datos.

El sistema se prepara para realizarle las pruebas de rendimiento para ello se utiliza una PC con las siguientes características:

Hardware: Intel Core i3-2120 a 3.30 GHzx4, 4 GB de RAM, sistema operativo Ubuntu 12.04, tipo de sistema 32-bit en una partición de 50 GB.

Software: Gestor de Base de datos: PostgreSQL v9.1, Servidor de Aplicaciones: Apache 2.2, Máquina Virtual de Java: openjdk-7, Navegador Web: Mozilla Firefox.

LAN: Tarjeta de red 100 Mbps de velocidad.

Se instala un MT que utiliza la versión 1.2.6 del ORM Doctrine y otro que funciona con las librerías de Doctrine 2

3.3.1 Descripción de las 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 de carga y estrés se utiliza la herramienta Apache JMeter en su versión 2.3.1. JMeter 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.

Para iniciar las pruebas es necesario crear 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 siguiente Figura, se inicia la cabecera de la prueba en la que se define la cantidad de usuarios que se modelan.

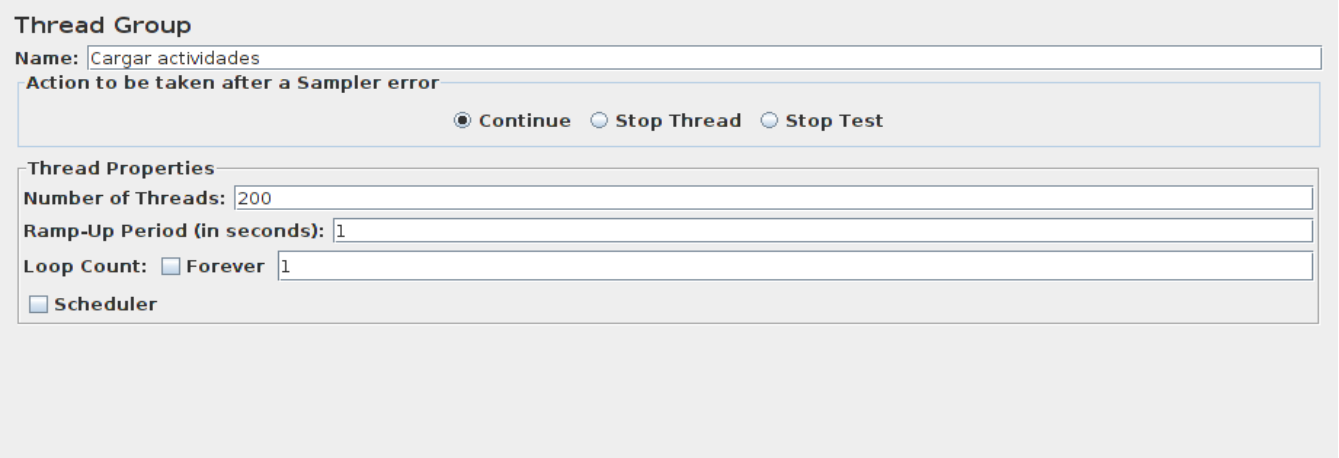


Figura.24 Elemento Grupo de Hilo.

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

- ✓ Número de hilos: número de usuarios a simular (200).
- ✓ 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:

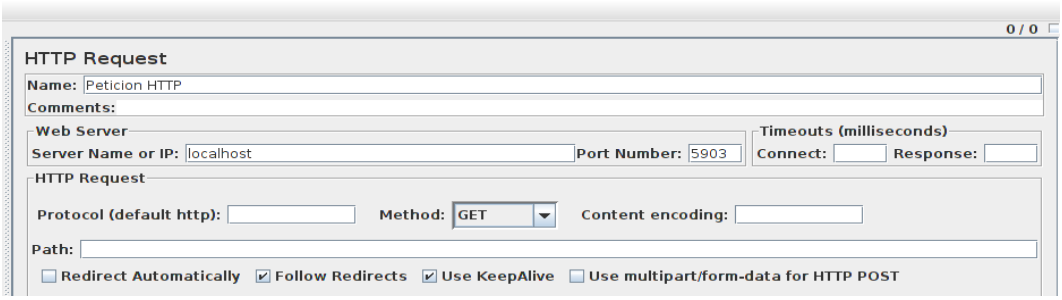


Figura.25 Selección del Servidor Proxy HTTP.

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. 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

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.

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.

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.

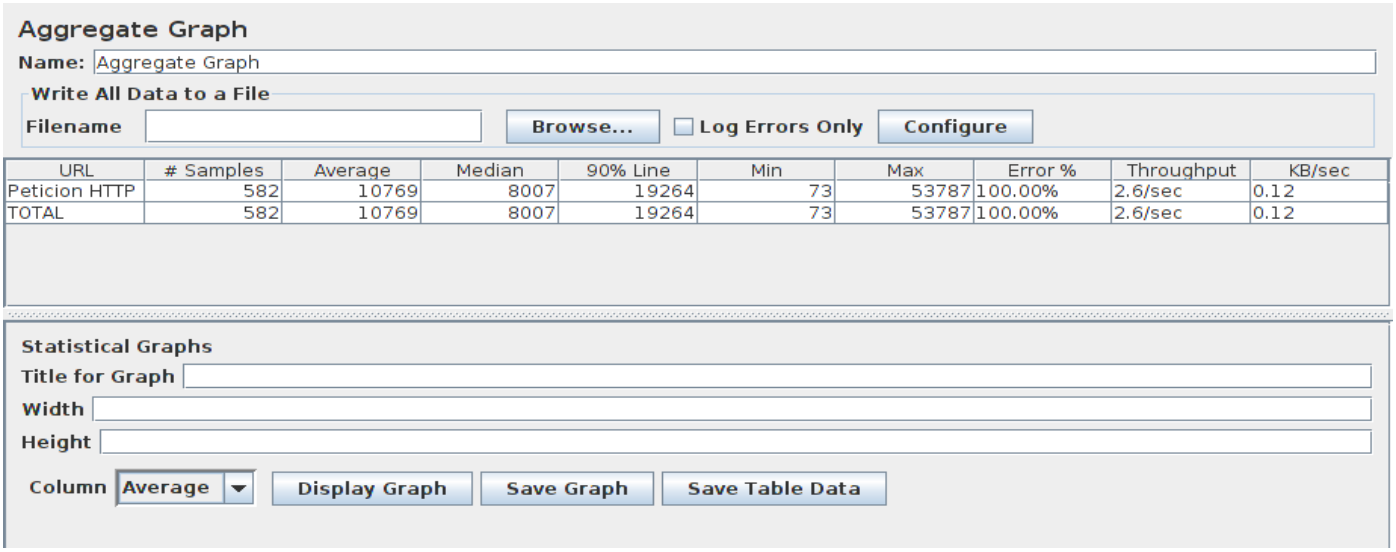


Figura.26 Selección de Informe Agregado.

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

 # 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.

% Error: porcentaje de las peticiones con errores.

El elemento Informe Agregado muestra los resultados correspondientes a cada Plan de Prueba, donde se evidencia el porciento de error para cada petición el número de muestras y el rendimiento.

3.4 Resultado de las pruebas de rendimiento

Las pruebas se realizaron con un total de 200 hilos, simulando 200 usuarios conectados de forma concurrente, siendo este el escenario más crítico.

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

Tiempo Total = $\#Muestras * Media\,[ms]$ (36)

El resultado obtenido se expresa en milisegundos.

Para calcular el tiempo promedio de cada usuario que estuvo accediendo al sistema se utilizó la siguiente fórmula:

Tiempo promedio por hilo= $Tiempo\frac{Total}{1000*60*hilos}\,[min]$ (36)

Resultados obtenidos empleando Doctrine 1

Prueba Nro. 1:

Los valores obtenidos por el elemento Informe Agregado, correspondiente a Doctrine 1 se muestran en la siguiente tabla.

Tabla.4 Valores correspondientes a la Prueba cargar actividades para 200 hilos.

	#Muestra	Media	Mediana	Línea de 90%	Min	Max	%Error
TOTAL	3400	9991	6990	23809	61	50390	0.00

Como se puede apreciar el tiempo promedio es 9.991 seg al realizar 3400 requerimientos al servidor.

El tiempo total utilizado para los 200 hilos se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media =3400 *9991= 33969400 milisegundos.

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

Tiempo Total/ (1000*60*cantidad de hilos) = 33969400/(1000*60*200) = 2,8307 minutos.

Prueba Nro. 2:

Los valores obtenidos por el elemento Informe Agregado, correspondiente a Doctrine 1 se muestran en la siguiente tabla.

Tabla.5 Valores correspondientes a la Prueba Cargar Plan para 200 hilos.

	#Muestra	Media	Mediana	Línea de 90%	Min	Max	%Error
TOTAL	4000	15510	13170	27074	1118	55030	0.00

Como se puede apreciar el tiempo promedio es 15,510 seg al realizar 4000 requerimientos al servidor. El tiempo total utilizado para los 200 hilos se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 4000*15510= 62040000 milisegundos.

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

Tiempo Total/ (1000*60*cantidad de hilos) = 62040000/(1000*60*200) = 5,17 minutos.

Prueba Nro. 3:

Los valores obtenidos por el elemento Informe Agregado, correspondiente a Doctrine 1 se muestran en la siguiente tabla.

Tabla.6 Valores correspondientes a la Prueba Obtener Reporte.

	#Muestra	Media	Mediana	Línea de 90%	Min	Max	%Error
TOTAL	3800	13079	10558	25407	460	55462	0.00

Como se puede apreciar el tiempo promedio es 13,079 seg al realizar 3800 requerimientos al servidor. El tiempo total utilizado para los 200 hilos se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 3800*13079= 49700200 milisegundos.

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

Tiempo Total/ (1000*60*cantidad de hilos) = 49700200 /(1000*60*200) = 4,14 minutos.

Resultados obtenidos empleando Doctrine 2

Prueba Nro. 4:

Los valores obtenidos por el elemento Informe Agregado, correspondiente a Doctrine 2 se muestran en la siguiente tabla.

Tabla.7 Valores correspondientes a la Prueba Cargar Actividades para 200 hilos.

	#Muestra	Media	Mediana	Línea de 90%	Min	Max	%Error
TOTAL	3400	9799	9177	17744	75	32846	0.00%

Como se puede apreciar el tiempo promedio es 9,799 seg al realizar 3400 requerimientos al servidor. El tiempo total utilizado para los 200 hilos se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 3400 *9799 = 33316600 milisegundos.

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

Tiempo Total/ (1000*60*cantidad de hilos) = 33316600 /(1000*60*200) = 2,7763 minutos.

Prueba Nro. 5:

Los valores obtenidos por el elemento Informe Agregado, correspondiente a Doctrine 2 se muestran en la siguiente tabla.

Tabla.8 Valores correspondientes a la Prueba Cargar Plan para 200 hilos.

	#Muestra	Media	Mediana	Línea de 90%	Min	Max	%Error
TOTAL	4000	5926	4951	12252	68	32740	0.00%

Como se puede apreciar el tiempo promedio es 5,926 seg al realizar 4000 requerimientos al servidor.

El tiempo total utilizado para los 200 hilos se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 4000 * 5926 = 23704000 milisegundos.

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

Tiempo Total/(1000*60*cantidad de hilos) = 23704000 /(1000*60*200) = 1,975 minutos.

Prueba Nro. 6:

Los valores obtenidos por el elemento Informe Agregado, correspondiente a Doctrine 2 se muestran en la siguiente tabla.

Tabla.9 Valores correspondientes a la Prueba Obtener Reporte para 200 hilos.

	#Muestra	Media	Mediana	Línea de 90%	Min	Max	%Error
TOTAL	3800	8278	7440	16386	71	32615	0.00%

Como se puede apreciar el tiempo promedio es 8,278 seg al realizar 3800 requerimientos al servidor.

El tiempo total utilizado para los 200 hilos se puede calcular con la siguiente fórmula:

Tiempo Total = #Muestras * Media = 3800*8278 = 31456400 milisegundos.

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

Tiempo Total/(1000*60*cantidad de hilos) = 31456400 /(1000*60*200) = 2,6213 minutos.

El tiempo total utilizado para la cantidad de hilos definida, cambia notablemente en la versión de *Doctrine 2* respecto a la anterior. En el siguiente gráfico se visualizan los resultados obtenidos.

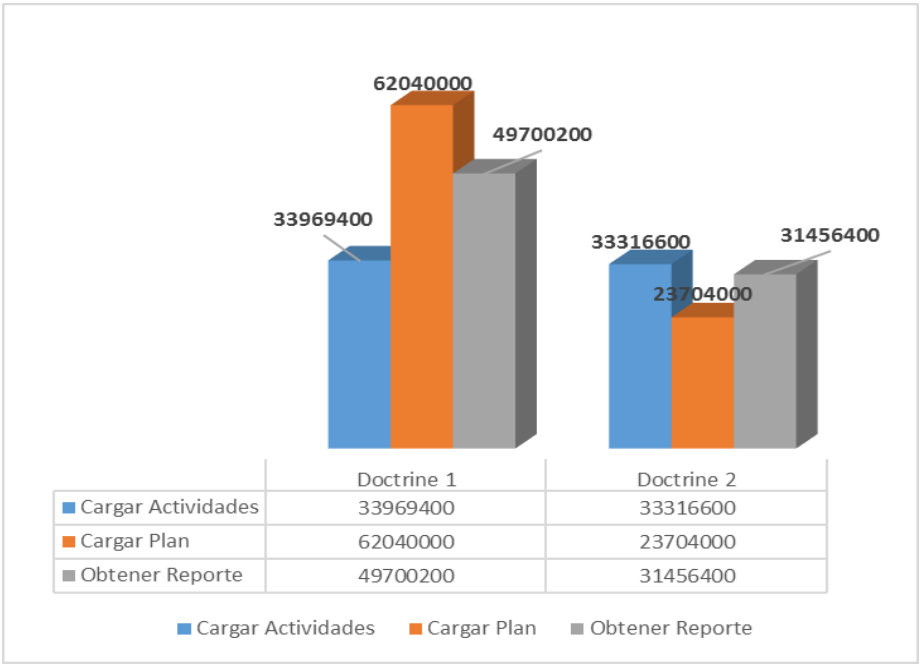


Figura.27Gráfico de Resultados del Tiempo Total.

Por su parte, el tiempo promedio requerido para cada hilo se muestra en la siguiente Figura, donde se puede apreciar que hubo una mejora también de la versión 2 respecto a la 1.

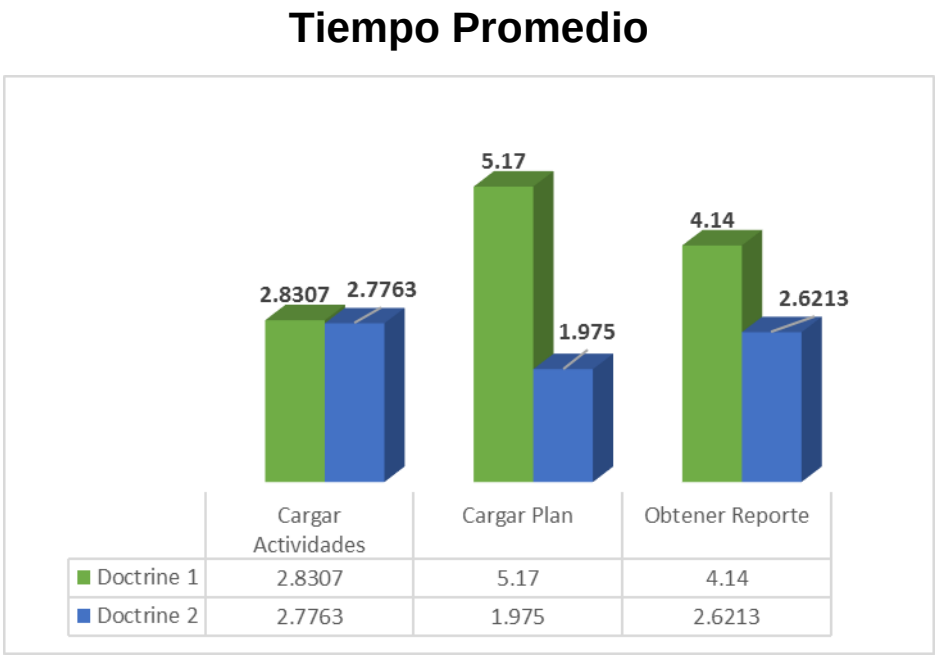


Figura.28 Gráfico de Resultados del Tiempo Promedio.

Resolver no conformidades

A partir de los resultados obtenidos con los elementos del JMeter, se puede apreciar cuál petición no se ejecutó satisfactoriamente. En el caso del Informe Agregado este indica en la **columna % Error** si existió algún problema. 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.

3.5 Conclusiones parciales

Para la realización de las pruebas de rendimiento se utilizó la herramienta JMeter, cuyo análisis demostró las facilidades que brinda su empleo en este tipo de pruebas.

Se obtuvo como resultado un mejoramiento en los tiempos de respuesta de la capa de acceso a datos del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades empleando la versión 2.4.2 de Doctrine en comparación con el mismo componente empleando la versión 1.2.2 de Doctrine.

CONCLUSIONES GENERALES

Mediante la investigación realizada se llega a la conclusión que el uso de la versión 2 de Doctrine traerá ventajas en comparación con la actual versión que se usa en la capa de acceso a datos del componente Planeación del módulo Planificación del Sistema Planificación de Actividades, ya que en él se ven reflejadas las principales potencialidades implementadas por los demás marcos de persistencia de datos, también frente a su antecesor se ve mucho mejor favorecido en cuanto a rendimiento, prestaciones y también mejora la estructura y el diseño.

Se realizó la migración de la capa de acceso a datos del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades, migrando un total de 255 ficheros de mapeo divididos en los cuatro directorios generados por Doctrine 2. Se migró este componente teniendo en cuenta las cuatro fases definidas (Inicial, Integración, Implementación y Pruebas) en el procedimiento para la migración seleccionado.

Las pruebas de rendimiento realizadas sobre el componente utilizando la herramienta Apache Jmeter arrojaron como resultado un mejoramiento en los tiempos de respuesta de la capa de acceso a datos del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades.

RECOMENDACIONES

Migrar los componentes y módulos en desarrollo del Sistema de Planificación de Actividades a la versión 2.4.2 de Doctrine.

BIBLIOGRAFÍA REFERENCIADA

1. SANDY MACHADO SCULL, NÉSTOR BERNAL VIDAL, YULIET GALÁN RAMÍREZ. Informatización de la planificación por objetivos para el contexto nacional actual. In: *Serie Científica*. 2011, Vol. 4, no. 11.

2. PROYECTO SIPAC. *Manual de Usuario del Sistema para la Planificación de Actividades (SIPAC)*. julio 2013. S.l.: s.n.

3. MSC. OINER GÓMEZ BARYOLO. *CAEM: MODELO DE CONTROL DE ACCESO PARA SISTEMAS DE INFORMACIÓN EN ENTORNOS MULTIDOMINIOS*. Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas. La Habana, 2012: s.n., 2012.

4. Marco de desarrollo de la Junta de Andalucía. In: [online]. 2014. Available from: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/180.

5. INOELKIS VELAZQUEZ OSORIO, LISANDRA CORDERO ESTRADA y RENÉ RODRIGO BAUTA CAMEJO. PROCEDIMIENTO PARA ACTUALIZACIÓN DE LA CAPA DE ACCESO A DATOS DEL MARCO DE TRABAJO SAUXE DE DOCTRINE 1.2.2 A DOCTRINE 2.2. In: . septiembre 2013,

6. Entrevista Fabien Potencier. In: [online]. enero 2011, Available from: http://desarrolla2.com/post/entrevista-con-fabien-potencier.

7. ROLANDO A. HERNANDEZ LEON. y SAYDA COELLO GONZALEZ. *El paradigma cuantitativo de la investigacion cientifica.* La Habana, 2011: Editorial Universitaria, [no date]. ISBN 978-959-16-1307-3.

8. LEÓN, R.A.H Y GONZÁLEZ, S.C. El proceso de investigación científica. In: . 2011, ISBN 978-959-16-1307-3.

9. JIMÉNEZ C., W. *Planificación*. 1982. S.l.: s.n.

10. Terry G. y Franklin S. *Principios de Administración.* México, CECSA 1987. S.l.: s.n.

11. STONER, J. *Administración. México: Prentice - Hall Interamericana*. 1996. S.l.: s.n.

12. FRANCISCO M. TAMAYO y FELIX ROBERTO ABALLI. Herramienta para la migración de datos de Microsoft Access a PostgreSQL. In: . 2009, Ciudad de la Habana:Universidad de las Ciencias Informaticas

13. BUGARIN, J.LUIS. Slideshare. In: *Slideshare* [online]. 2014. [Accessed 4 septiembre 2014]. Available from: https://www.slideshare.net/jlbugarin/persistencia-dedatoshibernatearquitecturasdesoftware#btnNext. 2014

14. Community, H. In: [online]. 2012, Available from: http://docs.jboss.org/hibernate/orm/4.2/devguide/en-US/html/ch02.html#d5e618.

15. Propel, S. Propel, Smart, easy object persistence. In: [online]. 2013, Available from: http://propelorm.org/.

16. VONDRICK, C. Symfony. ¿Cómo utilizar Propel? In: [online]. 2014, Available from: http://symfony.com/legacy/doc/cookbook/1_2/es/propel_13.

17. GARCÍA, L. Scribd, Django ORM. In: [online]. 2011. Available from: http://es.scribd.com/doc/63901225/Django-ORM.

18. SUBSONIC. SubSonic. In: [online]. 2012. Available from: http://subsonicproject.com/.

19. SUBSONIC. Subsonic. In: [online]. 2012, Available from: http://subsonicproject.com/docs/Comparisons/#Performance.

20. DOCTRINE-TEAM. Doctrine Performance Revisited. In: [online]. 2010. Available from: http://www.doctrine-project.org/blog/doctrine-performance-revisited.html.

21. Doctrine Project. In: [online]. [Accessed 8 mayo 2014]. Available from: http://www.doctrine-project.org/.

22. *Arquitectura Vista de Entorno de Desarrollo Tecnologico*. 2011. S.l.: s.n.

23. BEN COLLINS-SUSSMAN, B.W.F.Y.C.M.P.,. *Control de Versiones con Subversion*. S.l.: s.n., 2013.

24. RAPIDSVN. In: [online]. 2012. Available from: http://www.rapidsvn.org.

25. PosgreSQL-es. In: [online]. 2013 2009. [Accessed 4 marzo 2015]. Available from: https://postgresql.org.es.

26. An Introduction to NetBeans. In: *NetBeans* [online]. 2014. [Accessed 27 octubre 2014]. Available from: https://netbeans.org/.

27. Welcome to The Apache Software Foundation! In: [online]. 2014. [Accessed 30 octubre 2014]. Available from: http://www.apache.org/.

28. PUPO, Y.C. *Libro de Ayuda del Marco de Trabajo Sauxe*. S.l.: s.n., 2010.

29. Zend Framework - Guía de Referencia para el Programador - Español - zfdes.com. In: [online]. 2014. [Accessed 23 febrero 2015]. Available from: http://manual.zfdes.com/.

30. [Accessed 13 febrero 2015]. Available from: http://manual.zfdes.com/es/introduction.overview.html.

31. GUTIÉRREZ, J. D. *Desarrollo Web con PHP 5 y MySQL*. S.l.: s.n., 2004.

32. Apache JMeter - Apache JMeter™. In: *Apache JMeter* [online]. 2014. [Accessed 4 noviembre 2014]. Available from: http://jmeter.apache.org/.

33. YANET R. MORALES y MICHAEL E. MARRERO. Extension de la herramienta Visual Paradigm para la generacion de clases de acceso a datos con Doctrine 2.0. In: *5-11-2013*. 2013, Vol. Vol 6.

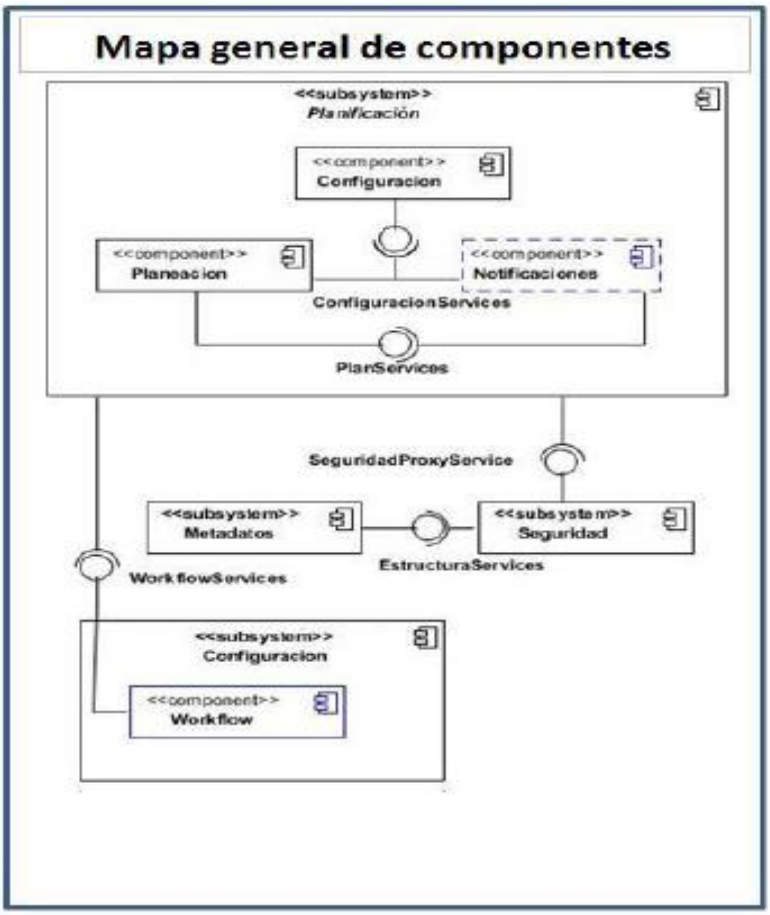
34. IAN SOMMERVILLE. *Sommerville_Parte_V_Verificacion_y_Validacion*. S.l.: s.n., 2005. ISBN 84-7829-074-5.

35. QA Técnico: Pruebas de Rendimiento: Tipos y objetivos. In: [online]. 2014. [Accessed 23 febrero 2015]. Available from: http://qatecnico.blogspot.com/2012/03/pruebas-de-rendimiento-tipos-y.html.

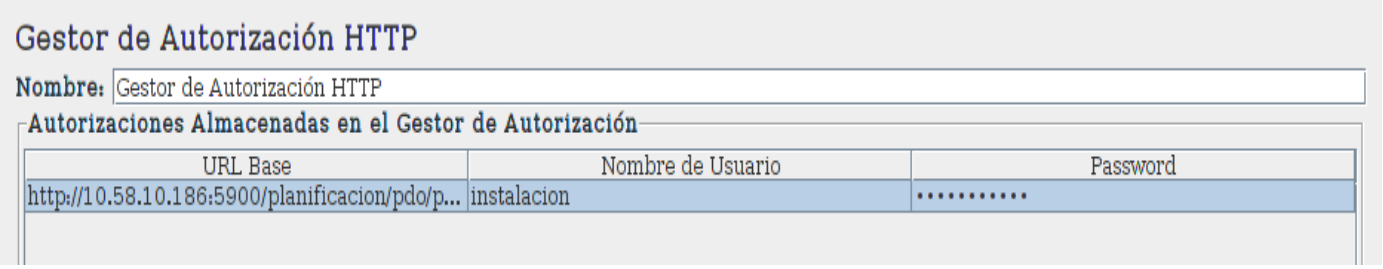
36. DÍAZ, Javier F., BANCHOFF TZANCOFF, Claudia M., RODRÍGUEZ, Anahí S. y SORIA, Valeria. Usando Jmeter para pruebas de rendimiento. In: *XIV Congreso Argentino de Ciencias de la Computación*. S.l.: s.n., 2008.

ANEXOS

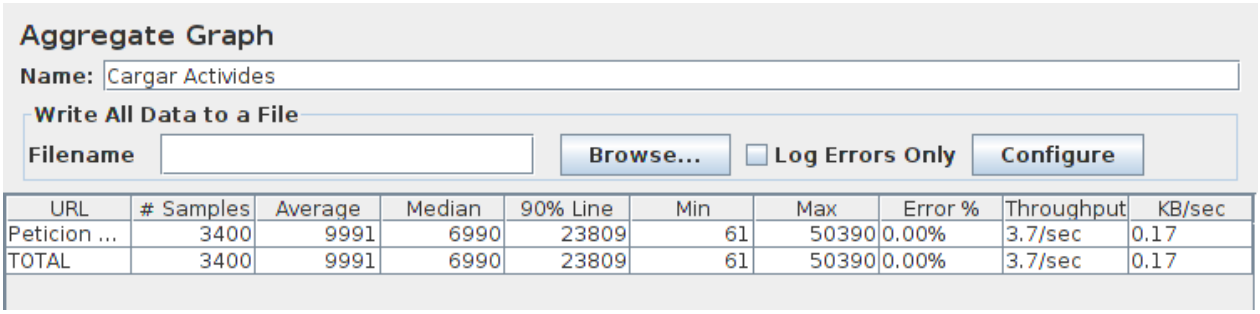
Anexo.1 Mapa general de componentes



Anexo.2 Gestor de Autorización



Anexo.3 Cargar Actividades (Doctrine 1)



Aggregate Graph

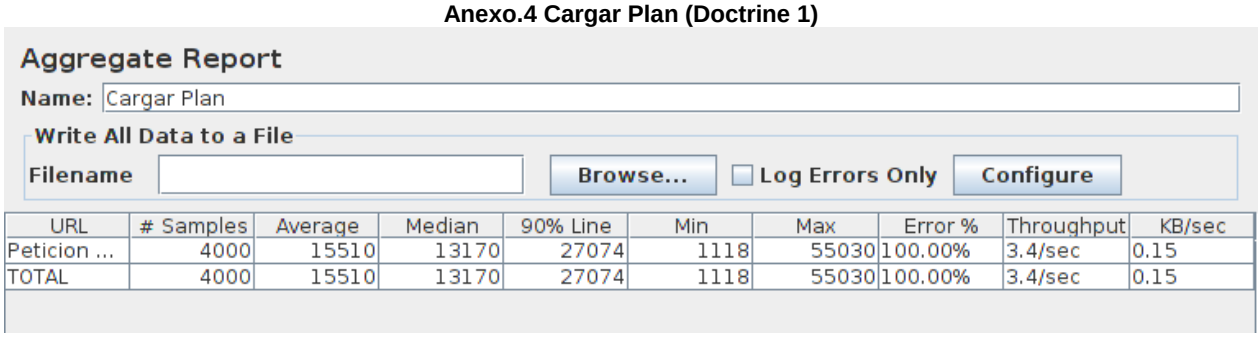
Name: Cargar Actividades

Write All Data to a File

Filename | Browse... | Log Errors Only | Configure

URL	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
Peticion ...	3400	9991	6990	23809	61	50390	0.00%	3.7/sec	0.17
TOTAL	3400	9991	6990	23809	61	50390	0.00%	3.7/sec	0.17

Anexo.4 Cargar Plan (Doctrine 1)



Aggregate Report

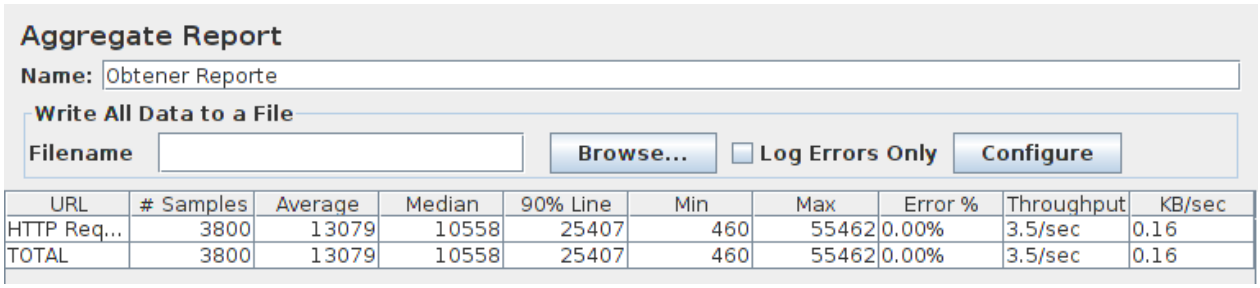
Name: Cargar Plan

Write All Data to a File

Filename | Browse... | Log Errors Only | Configure

URL	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
Peticion ...	4000	15510	13170	27074	1118	55030	100.00%	3.4/sec	0.15
TOTAL	4000	15510	13170	27074	1118	55030	100.00%	3.4/sec	0.15

Anexo.5 Obtener Reporte (Doctrine 1)



Aggregate Report

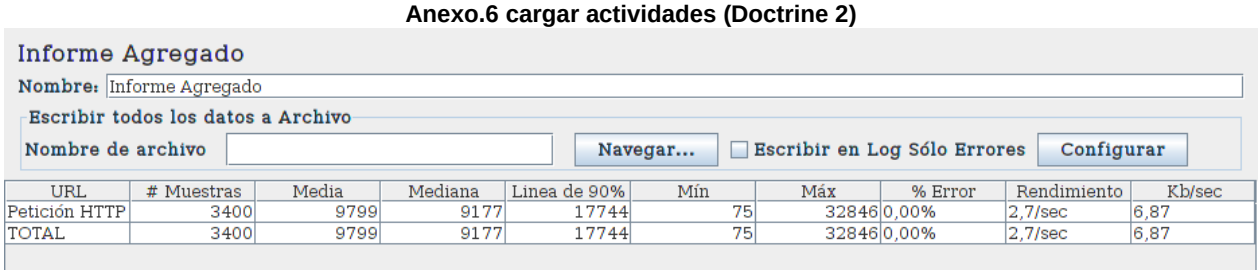
Name: Obtener Reporte

Write All Data to a File

Filename | Browse... | Log Errors Only | Configure

URL	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
HTTP Req...	3800	13079	10558	25407	460	55462	0.00%	3.5/sec	0.16
TOTAL	3800	13079	10558	25407	460	55462	0.00%	3.5/sec	0.16

Anexo.6 cargar actividades (Doctrine 2)



Informe Agregado

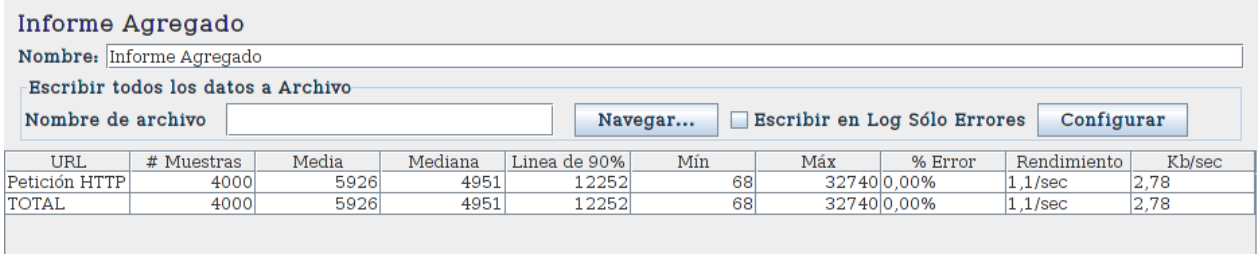
Nombre: Informe Agregado

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Escribir en Log Sólo Errores | Configurar

URL	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Petición HTTP	3400	9799	9177	17744	75	32846	0,00%	2,7/sec	6,87
TOTAL	3400	9799	9177	17744	75	32846	0,00%	2,7/sec	6,87

Anexo.7 cargar plan (Doctrine 2)



Informe Agregado

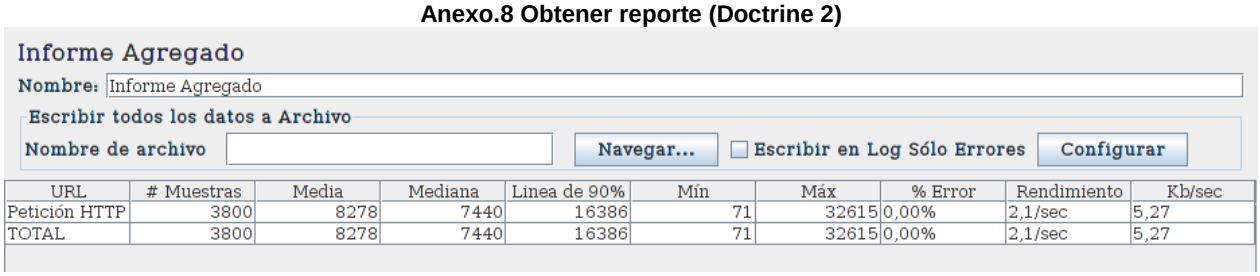
Nombre: Informe Agregado

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Escribir en Log Sólo Errores | Configurar

URL	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Petición HTTP	4000	5926	4951	12252	68	32740	0,00%	1,1/sec	2,78
TOTAL	4000	5926	4951	12252	68	32740	0,00%	1,1/sec	2,78

Anexo.8 Obtener reporte (Doctrine 2)



Informe Agregado

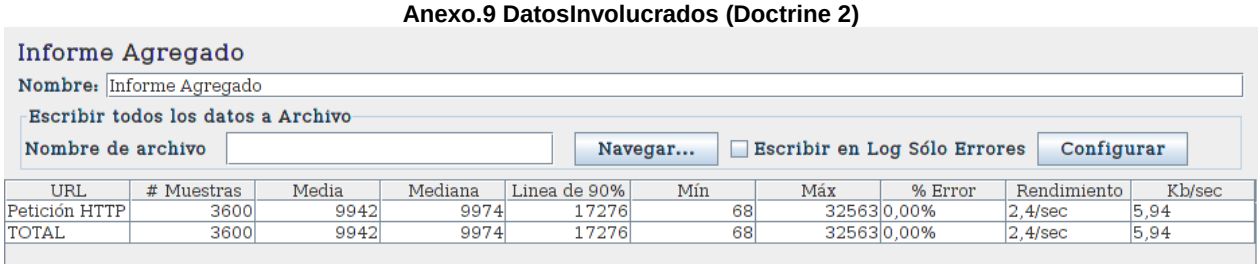
Nombre: Informe Agregado

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Escribir en Log Sólo Errores | Configurar

URL	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Petición HTTP	3800	8278	7440	16386	71	32615	0,00%	2,1/sec	5,27
TOTAL	3800	8278	7440	16386	71	32615	0,00%	2,1/sec	5,27

Anexo.9 DatosInvolucrados (Doctrine 2)



Informe Agregado

Nombre: Informe Agregado

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Escribir en Log Sólo Errores | Configurar

URL	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Petición HTTP	3600	9942	9974	17276	68	32563	0,00%	2,4/sec	5,94
TOTAL	3600	9942	9974	17276	68	32563	0,00%	2,4/sec	5,94

Anexo.10 Acta de aceptación

CENTRO DE INFORMATIZACIÓN DE ENTIDADES

Acta de aceptación

UCI Informáticas

En la Habana a los 27 días del mes de febrero 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 del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades a Doctrine 2", representado en este acto por **Dalinda Llapur Del Rio**, que a los fines y efectos derivados del presente documento se denominarán 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 del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades para mejorar el rendimiento del mismo.
3. Validar el rendimiento del componente Planeación del módulo Planificación del Sistema de Planificación de Actividades.

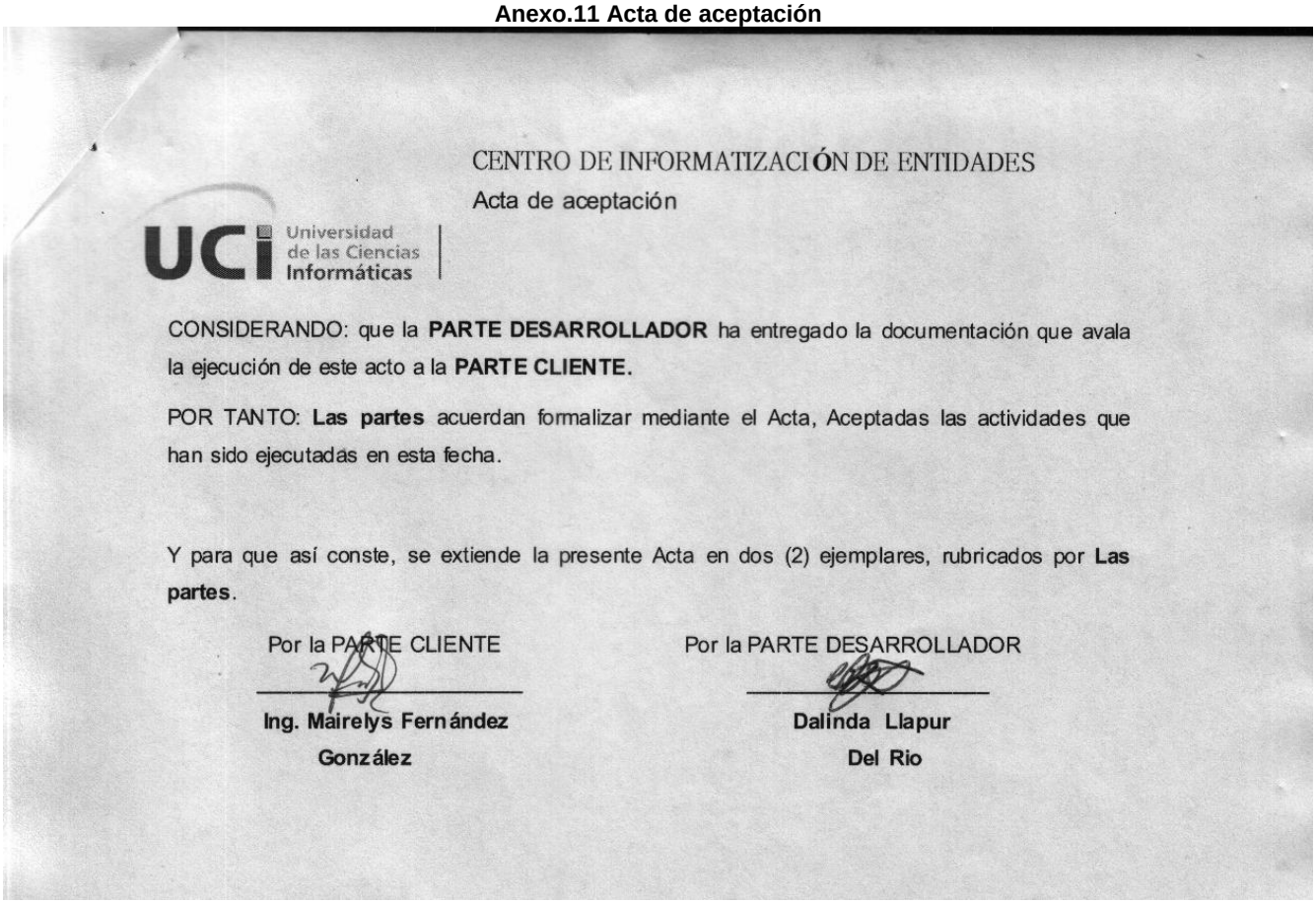
CONSIDERANDO: Que se migraron un total de 15 tablas de la capa de Acceso a datos correspondientes al componente Planeación, el cual está incluido en el módulo Planificación del sistema, de acuerdo con los niveles de empaquetamiento definidos en la arquitectura base.

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**.

Anexo.11 Acta de aceptación



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 **Las partes**.

Por la PARTE CLIENTE

Ing. Mairelys Fernández González

Por la PARTE DESARROLLADOR

Dalinda Llapur Del Rio

Anexo 12

Cronograma de actividades		
Actividades	**Responsable**	**Fecha de cumplimiento**
Estudio de la metodología e investigación	Dalinda	21/6/2014
Redacción de perfil de tesis	Dalinda, Tutor, J' Proyecto SIPAC	10/5/2014
Aprobación del perfil de tesis	Dalinda, Tutor,Tribunal de tesis	3/10/2014
Diseño teórico	Dalinda	8/10/2014
Encuentro con tutor(semanal)	Dalinda, Tutor	semanal
Taller sobre implementación usando Doctrine 2	Dalinda, Dpto Desarrollo de componente	1-31/10/2014
Marco Conceptual: Preparación sobre ORM más usados	Dalinda	15/10/2014
Marco Conceptual: Preparación del entorno desarrollo	Dalinda	17/10/2014
Marco Conceptual: Estudio del procedimiento a realizar	Dalinda	1-31/10/2014
Marco Conceptual: Preparación de	Dalinda	31/10/2014

Tecnologías a usar		
Marco Conceptual: Preparación de Herramientas a usar	Dalinda	31/10/2015
Desarrollo de la migración:		
Preparar estación de trabajo	Dalinda	2-8/11/2014
Estudio de la arquitectura	Dalinda, Arquitecto de sistema	10-14/11/2014
Estudio de la estructura de carpeta para Doctrine 1	Dalinda	16-17/11/2014
Estudio y aplicación de las nuevas librerías a utilizar	Dalinda	17-26/11/2014
Crear directorio a ubicar los ficheros a partir de la nueva estructura	Dalinda	17/11/2014
Selección de las clases a implementar nuevamente	Dalinda	18-21/11/2014
Establecer nueva estructura de carpeta	Dalinda	23-27/11/2014
Rediseñar e implementar la capa de acceso a datos	Dalinda	1-15/12/2014
Rediseñar métodos y consultas	Dalinda	1-15/12/2015
Implementar nueva forma de mapeo	Dalinda	1-15/12/2016
Re Implementar capa de negocio	Dalinda	1-15/12/2017
Investigar diferentes tipos de pruebas que existen	Dalinda	5-9/1/2015
Realizar pruebas de rendimiento	Dalinda	12-26/1/2015
Realizar pruebas a la conectividad de base de datos	Dalinda	12-26/1/2016
Verificar la correspondencia entre el modelo relacional y el modelo objetual	Dalinda	12-26/1/2017
Verificar correcto funcionamiento de los repositorios	Dalinda	12-26/1/2018
Verificar la correcta configuración de los namespaces	Dalinda	12-31/1/2019
Preparar el entorno de prueba	Dalinda	27/1/2015
Comparar resultados de ambas versiones	Dalinda	3-13/2/2015

Anexo.13 Juego de datos

Juego de datos:

$idactividad es un **$idelemento**:

9000004734 fecha de inicio: "2014-07-01" fecha fin:"2014-07-06"

9000004921 fecha de inicio: "2014-05-05" fecha fin: "2014-05-18"

9000004103 **fecha de inicio:** "2014-03-20" **fecha fin:** "2014-03-20"

$idplan 9000004942 fecha inicio:"2014-01-01" fecha fin: "2014-12-31"

$idobjetivo:

9000004937 fecha inicio:"2014-06-18" **$idautor**: 90000000046

9000004938 fecha inicio:"2014-06-18"

9000004939 fecha inicio:"2014-06-18"

9000004940 fecha inicio:"2014-06-18" **$idautor**: 90000000046

9000004941 fecha inicio:"2014-06-18"

$idarc 9000000161 **$idusuariocrea** 90000000046

$idinvolucrado: 9000000033, 9000000034,9000000035 $idarc9000000161, llaveinvolucrado900000000561, denominación: I. TRABAJO POLÍTICO IDEOLÓGICO Y DE ORGANIZACIÓN

$idtipoelemento: 0, 1, 2, 3 **$dentipoelemento:** "Plan","Objetivo","FIP","Actividades"

$idusuario: 90000000038 **$idnomenclador:** 9000000210, 9000000211

Anexo 14 Juego de datos

Juego de datos:

$idtipo_nomenclador:**2 **$dennomenclador:" investigación" **idusuariocrea**: 90000000037 fecha inicio:"2013-09-26" **$idnomenclador**: 9000000210

$asunto Nombre **$idelemento** 9000004944 **$idobservaciones:** 9000000073

$idelementopadre: 9000004942

$idelementohijo: 9000004101, 9000004102, 9000004103.....

$origen: 3

$idautor: 90000000105

$idestadoaprob 1000000088 **$idusuariocrea**: 90000000046

$denelemento: "Pleno del CC PCC. Hora: 10:00"

$idelemento: 9000004101

$denelemento:"Reunión Comisión del Buró Político para el control de la implementación de los Acuerdos del VI Congreso. Hora: 14:00."

$idelemento: 9000004102

$idarc: 9000000162

$activo: 1

$idarc: 9000000163

$activo: 1