Universidad de las Ciencias Informáticas

Facultad 3

Migración de la capa de acceso a datos del Sistema Integral de Seguridad Acaxia a Doctrine 2.0

Autor(es): Carlos Javier Martínez León

Tutor(es): Ing. René R. Bauta Camejo

Ing. Inoelkis Velázquez Osorio

Ing. Dausbel Torreblanca Pavó

Mayo 2014

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los ____ días del mes de _Junio_ del año __2014__.

Carlos Javier Martínez León

_____
Firma del Autor

Ing. René Bauta Camejo

_____
Firma del Tutor 1

Ing. Inoelkis Velázquez Osorio

_____
Firma del Tutor 2

Ing. Dausbel Torreblanca Pavó

_____
Firma del Tutor 3

Datos de Contacto

Tutor: Ing. René Bauta Camejo

Edad: 29 años

Ciudadanía: Cubano

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. Cinco años de graduado.

Tutor: Ing. Inoelkis Velázquez Osorio

Edad: 24 años

Ciudadanía: Cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

Categoría docente: Recién graduado

E-mail: inoelkis@uci.cu

Ingeniero en Ciencias Informáticas, graduado en 2013 en la Universidad de las Ciencias Informáticas. Recién graduado. Un año de experiencia en el desarrollo de software. Un año de graduado.

Tutor: Ing. Dausbel Torreblanca Pavó

Edad: 25 años

Ciudadanía: Cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

Categoría docente: Ninguna

E-mail: dtorreblanca@uci.cu

Ingeniero en Ciencias Informáticas, graduado en 2012 en la Universidad de las Ciencias Informáticas. Dos años de experiencia en el desarrollo de software. Dos años de graduado.

Dedicatoria

A mis **padres** que siempre han estado presente, en especial a mi **madre** que lo ha dado todo por mí sin dudarlo ni un segundo.

A mi **hermana** que la quiero mucho.

Mis **sobrinos** que los quiero mucho y que algún día lleguen a donde estoy.

Agradecimientos

A mis **padres** que siempre me apoyaron en todo en especial a mi **madre**.

A mi **hermana** y mis **sobrinos** que los quiero.

A mi **novia** que la quiero mucho y me ayudó y apoyó en todo momento.

A **Mayra** la viejuca y **María Esther** mis segundas madres.

A **Omar** mi segundo padre.

A **Damián** el hermano que nunca tuve.

A mis **tutores** por aguantarme tanto tiempo en especial a **Inoelkis** que siempre estuvo cuando lo necesité.

A todos mis **colegas** de 5 años de estudio. Que se han convertido en mi gran familia UCI.

A todos los **profesores** que me ayudaron a llegar hasta el final.

Resumen

En la actualidad existe una gran tendencia a la explotación de las tecnologías para la persistencia de datos y paralelamente las tecnologías relacionadas con el acceso a datos, haciéndose cada vez más necesario que el acceso se haga con mayor seguridad y eficiencia. Existen herramientas capaces de generar ficheros de mapeo, cada una con sus características, fortalezas y vulnerabilidades, sin llegar a ser alguna, la alternativa dominante.

El presente trabajo propone una solución para aumentar el rendimiento de capa de acceso a datos en el Sistema Integral de Seguridad Acaxia, en el cual actualmente se trabaja con la versión 1.2.2 del marco de trabajo de persistencia de datos Doctrine. Planteándose como objetivo migrar a la versión 2 la capa de acceso a datos de Acaxia. Para lograrlo se desarrolla un marco teórico donde se estudian los principales conceptos, herramientas y tecnologías a utilizar en las posteriores etapas de desarrollo, análisis e implementación.

Palabras claves: Doctrine, mapeo, Marco de Trabajo.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Índice de Anexos

Introducción:

En la actualidad se ha incrementado el uso de las herramientas que facilitan la creación y el desarrollo de software, ejemplo de estas herramientas son los marcos de trabajo (MT). El objetivo fundamental de estas herramientas es optimizar el tiempo de desarrollo, incrementar la profesionalidad y calidad de las aplicaciones, automatizando la creación de estructuras de acuerdo con arquitecturas de software determinadas. Mayormente estas están dirigidas a la arquitectura Modelo Vista Controlador (MVC) que ha demostrado ser una de las más utilizadas en el desarrollo de aplicaciones web incrementando la reutilización y la flexibilidad de las aplicaciones (Mestras, 2009).

En Cuba se ha hecho extensivo el uso de estas herramientas en diversas instituciones dedicadas al desarrollo de software, la Universidad de las Ciencias Informáticas (UCI) es una de ellas, en la cual existen varios centros de desarrollo con experiencia en la utilización de los MT. Estas herramientas usan para la persistencia de datos diversos Mapeadores de Objetos Relacionales (ORM por sus siglas en inglés), de los cuales hay una gran variedad a escoger de acuerdo con la compatibilidad con el lenguaje, el gestor de bases de datos, la lógica de negocios y los software con los que interactúa.

En la Facultad 3 de la UCI se encuentra el Centro de Informatización de la Gestión de Entidades (CEIGE). Este centro cuenta con varios proyectos productivos, entre los cuales se encuentra el Sistema Integral de Seguridad Acaxia, este proyecto emplea en su desarrollo el MT Sauxe. Este integra el MT Zend para el desarrollo de aplicaciones y servicios web, utiliza en la capa de acceso a datos el Lenguaje de Consultas de Datos que implementa Doctrine[1] 1 (DQL por sus siglas en inglés) y para la capa de presentación se utilizan las bibliotecas ExtJS[2] por la variedad de sus componentes para hacer la interfaz de usuario más amigable.

[1] Marco de persistencia de datos compatible con el lenguaje de programación PHP.
[2] Conjunto de bibliotecas JavaScript para el desarrollo de aplicaciones web interactivas y de interfaces de usuario completas, usa tecnologías AJAX, DHTNL y DOM.

Acaxia actualmente utiliza en su capa de acceso a datos la versión 1.2.2 del marco de persistencia de datos Doctrine, lo cual provoca bajo rendimiento y un mayor acoplamiento entre sus clases por lo que se pierde independencia y sería más costoso a la hora del cambio hacia otra tecnología (Ercoli, 2008).

Pruebas realizadas sobre el Sistema de Planificación de Recursos Empresariales Xedro-ERP, el cual fue desarrollado empleando el MT Sauxe, demostraron que se obtenía un rendimiento reducido de la aplicación y que los tiempos de ejecución se elevaban al hacer uso del marco de persistencia de datos Doctrine. A partir de los resultados obtenidos se evidenció que el rendimiento era mayor cuando no se hacía uso del marco de persistencia de datos Doctrine (Días, 2012).

Otro inconveniente que presenta el uso de esta versión de Doctrine es que a partir del 1 de junio del 2011 se dejó de dar soporte, trayendo consigo que no se corrijan posibles errores y vulnerabilidades que atentan contra su estabilidad y rendimiento (Doctrine-Team, 2011).

A partir de la problemática antes planteada se define como **problema a resolver:** ¿Cómo aumentar el rendimiento de la capa de acceso a datos del Sistema Integral de Seguridad Acaxia?

Con el fin de resolver el problema planteado se traza como **objetivo general:** obtener la versión 2 de la capa de acceso a datos del Sistema Integral de Seguridad Acaxia para aumentar su rendimiento.

Se define como **objeto estudio:** marco de trabajo para la persistencia de datos.

Y más específicamente en el **campo de acción:** capa de acceso a datos del Sistema Integral de Seguridad Acaxia.

Para dar cumplimiento al objetivo trazado este se desglosó en los siguientes **objetivos específicos:**

- Construir el Marco Teórico de la investigación relacionada con los MT para la persistencia de datos existentes para identificar buenas prácticas y posibles puntos de reutilización.

- Desarrollar la migración de la capa de acceso a datos del Sistema Integral de Seguridad Acaxia.
- Validar mediante el uso del JMeter el aumento del rendimiento de la capa de acceso a datos del Sistema Integral de Seguridad Acaxia.

Para esta investigación se parte de la siguiente **idea a defender**:
Si se obtiene una versión 2 de la capa de acceso a datos del Sistema Integral de Seguridad Acaxia se aumentará el rendimiento de la misma.

Para una mejor comprensión de la investigación la misma se estructuró de la siguiente manera:
Capítulo 1: se elabora la **Fundamentación Teórica** del trabajo, donde se realiza un estudio de los marcos de persistencias de datos más utilizados y las herramientas a usar durante la migración.
Capítulo 2: se desarrolla la **Migración de la Capa Acceso a Datos**, donde se realiza una descripción detallada de la migración a realizar.
Capítulo 3: Validación de la Migración, se exponen los resultados obtenidos, y se realizan pruebas de rendimiento.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se formulará todo el marco teórico conceptual alrededor de los marcos de persistencias de datos, tomando como criterios: los más usados, ventajas y desventajas de su uso de acuerdo con la migración que se propone. Se hace un estudio de las herramientas y la metodología a usar.

1.2 Persistencia

Persistencia es el hecho de guardar permanentemente la información generada por una aplicación en un sistema de almacenado, con el objetivo de que se pueda usar más adelante, ya sea para consultar, modificar, combinar o eliminar (Ruíz, 2008).

Una definición sería: “Persistencia es la capacidad de un lenguaje de programación o entorno de desarrollo de programación para almacenar y recuperar el estado de los objetos de forma que sobrevivan a los procesos que los manipulan”. Esta definición indica de que el programador no debería preocuparse por el mecanismo interno del proceso de persistencia, y que esta sea una capacidad del lenguaje o del entorno, entiéndase por entorno: bibliotecas, MT, o compiladores (Ruíz, 2008).

1.3 Mapeo

Mecanismo que se basa en la traducción bidireccional entre los datos encapsulados en los objetos de la lógica de un sistema orientado a objetos y una fuente de datos que maneja un paradigma distinto. Un mapeador a de lidiar con los diferentes problemas que se presenten al mapear los datos entre paradigmas (Ruíz, 2008).

- **Mapeadores Objeto-Relacional:** la intención de este proceso radica en contar con un mecanismo para mapear los objetos de la lógica de un Sistema orientado a objetos a una base de datos relacional y viceversa (Miranda et al., 2006).
- **Mapeadores Objeto-XML [3] :** es la intención de este mecanismo permitir el almacenamiento de los datos contenidos en los objetos de la lógica en forma de documentos XML (Miranda et al., 2006).

[3] Por sus siglas en ingles de Extensible Markup Language (lenguaje de marcas extensibles).

El mapeo Objeto-XML provee un medio por el cual los datos de negocio pueden ser visualizados en su forma persistida, al mismo tiempo que se facilita el intercambio de dichos datos con otros sistemas.

1.4 Tecnologías de persistencia

Hoy en día uno de los asuntos más debatidos y discutidos en la industria del software es: ¿Cuál de las tecnologías de persistencias o MT para el mapeo Objeto-Relacional, merece ser la alternativa dominante? Esta discusión ha llevado a la comunidad a una división de criterios, por lo que mientras esto continúe los programadores deberán elegir la tecnología que mejor se adapte a sus necesidades.

1.4.1 Hibernate

Hibernate es una ORM de libre distribución bajo la licencia GNU/LGPL[4], funciona asociando a cada tabla de la base de datos un Plain Old Java Object (POJO[5]). Hibernate es totalmente transparente con el uso de las base de datos, pudiendo cambiar de base de datos sin necesidad de cambiar una línea de código de la aplicación, simplemente cambiando los ficheros de configuración (Martín, 2004).

1.4.1.1 Rendimiento

Para el manejo del rendimiento Hibernate propone una cache de primer nivel y una de segundo nivel de configuraciones y estrategias de recuperación que permiten a Hibernate, ya sea dinámica o estáticamente administrar el uso de la cache. Está disponible un rango completo de números de las operaciones internas que realiza Hibernate mediante métodos especializados (King et al., 2013).

1.4.1.2 Compatibilidad con gestores de bases de datos

En Hibernate el parámetro de dialecto en los ficheros de configuración indica el nombre de la clase que se encargará de comunicarse con la base de datos en el SQL[6] que la entienda,

[4] Licencia Pública General Reducida de GNU, o más conocida por su nombre en inglés GNU Lesser General Public License.

[5] Enfatiza el uso de clases simples independientes de un MT en especial, con propiedades accesibles mediante métodos setter y getter.

[6] Por sus siglas en inglés Structured Query Lenguage (Lenguaje de Consulta Estructurado).

este parámetro debe ser siempre especificado. El valor sería una subclase que herede de net.hibernate.dialect.Dialect (Martín, 2004).

1.4.2 Django

Django es un MT para el desarrollo de aplicaciones web, escrito en Python[7], que sigue el patrón de diseño de MVC. Django pone énfasis en la reutilización de código, la conectividad y extensibilidad de componentes y el desarrollo rápido. El núcleo de Django 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 (Molina et al., 2011).

1.4.2.1 Rendimiento

El ORM de Django es muy potente, porque permite sacar provecho de las ventajas del lenguaje para el cual está diseñado. Se pueden incluir consultas SQL para optimizar ciertos aspectos de las aplicaciones (o interacciones con la base de datos). Permite diseñar/definir tipos personalizados, los cuales pueden ser utilizados dentro de diferentes proyectos que estén dentro de Django (Molina et al., 2011).

1.4.3 Doctrine 1

Doctrine 1 es una implementación de ActiveRecord (registro activo), que propone hacer la persistencia en el mismo objeto de negocio (**Figura *1***), lo que trae que pierda independencia por acoplar los objetos de negocios de la capa de negocios con la persistencia de datos, usa métodos mágicos y estáticos que hace más difícil de probarlos. Está dividido en dos capas DBAL[8] y ORM (Doctrine-Team, 2010b).

[7] Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible.
[8] Por sus siglas en inglés Database Abstraction Layer (Capa de abstracción de base datos)

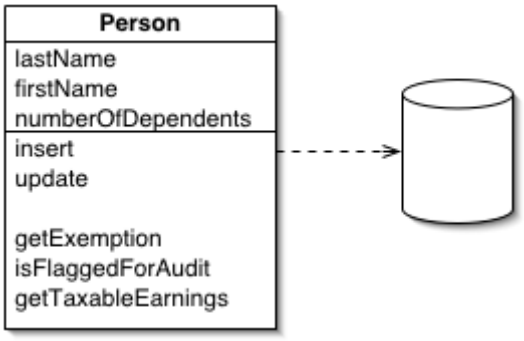


Figura 1: Modelo negocio Doctrine1 (Doctrine-Team, 2011).

1.4.3.1 Compatibilidad con lenguaje

Usa como lenguaje para la comunicación con las bases de datos el DQL, dándoles a los programadores una poderosa alternativa del SQL que mantiene la flexibilidad sin la necesidad de duplicación de código. No es compatible con la versión 5.3+ de PHP[9] (Doctrine-Team, 2010b).

1.4.3.2 Compatibilidad con gestores de bases de datos

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 la aplicación (Doctrine-Team, 2010b).

1.4.3.3 Soporte

A partir del 1 de junio del 2011 se dejó de dar soporte a esta versión de Doctrine, trayendo consigo que no se corrijan posibles errores y vulnerabilidades que atentan contra su estabilidad y rendimiento (Doctrine-Team, 2011).

1.4.4 Doctrine 2

Doctrine 2 es una implementación del patrón DataMapper (mapa de datos), no fuerza a las clases a extender de Doctrine Record, ni contiene detalles sobre el modelo relacional, como son las claves externas **Figura *2***.

[9] PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

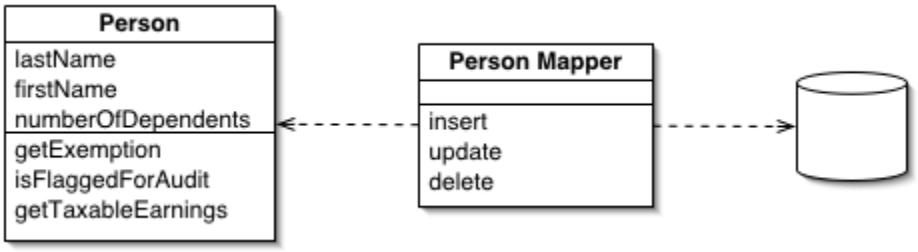


Figura 2: Modelo negocio Doctrine2 (Doctrine-Team, 2011).

Está dividido en tres paquetes (cada paquete contiene el anterior) (Sironi, 2010):

- Doctrine \ Common contiene código reutilizable como el analizador de anotaciones y el sistema de eventos.
- Doctrine \ DBAL abstrae no solo el controlador particular de base de datos, al igual que ya lo hace de forma nativa PDO[10], sino también los dialectos SQL de diferentes Sistemas de Gestión de Bases de Datos.
- Mapas de Doctrine \ ORM gráficos de objetos en base de datos y viceversa.

Doctrine 2 cuenta con una arquitectura más limpia y promueve la inyección de dependencia en lugar de acceder de la forma Doctrine_ * a las clases, ya sea estáticamente o mediante un producto único como Doctrine_Manager (Sironi, 2010).

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. Si se hace, da lugar a variantes de esquemas, es decir, el esquema de objeto y el esquema relacional no coinciden (Sironi, 2010).

1.4.4.1 Rendimiento

Doctrine 2 es superior en cuanto a rendimiento se refiere, las siguientes pruebas realizadas apoyan esta afirmación (Doctrine-Team, 2010a):

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

- Escenario 1: crear un nuevo objeto de modelo, definir sus columnas, y guardarlo. Velocidad del objeto de ensayos con modelos, y la generación de SQL INSERT.
- Escenario 2: consultar un registro por su clave primaria. Pruebas de consulta básica y de hidratación.
- Escenario 3: consultar un registro mediante una consulta compleja. Probar la velocidad de las consultas de objeto.
- Escenario 4: operaciones de búsqueda 5 registros en un criterio simple. Pruebas velocidad de hidratación.
- Escenario 5: consultar un registro e hidratar junto con su registro relacionado en otra tabla. Las pruebas unen la velocidad de hidratación.

Resultados:

La siguiente **Tabla *I*** muestra cómo se comporta el rendimiento de Doctrine 2 en comparación con su antecesor Doctrine1(Doctrine-Team, 2010a).

Tabla I: Resultados de pruebas de rendimiento (Doctrine-Team, 2010a).

	Insert	FindPK	Complex	Hydrate	With
PDOTestSuite	132	149	112	107	109
Doctrine1.2TestSuite	1673	2661	449	1710	1832
Doctrine1.2WithCacheTestSuite	1903	1179	550	957	722
Doctrine2TestSuite	165	426	412	1048	1042
Doctrine2WithCacheTestSuite	176	423	148	606	383

1.4.4.2 Compatibilidad con lenguaje

La diferencia más significativa con respecto a su antecesor es en el uso de la versión 5.3+ de php. Con respecto a compatibilidad con gestores de base de datos se mantuvieron las mismas características del lenguaje DQL de la antigua versión solo con cambios a la hora de definir consultas en el lenguaje(Doctrine-Team, 2010c).

1.4.4.3 Compatibilidad con gestores de bases de datos

La compatibilidad con los diferentes gestores de base de datos se mantuvieron de la misma manera que era realizada en la versión anterior(Doctrine-Team, 2010c).

1.4.4.4 Compatibilidad con el MT Sauxe

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

1.4.4.5 Soporte

Doctrine 2 es un marco de persistencia de datos que cuenta con un gran prestigio a nivel mundial, gracias a esto cuenta con una comunidad cada vez mayor, que le brinda facilidades de soporte y actualización a nuevas versiones.

1.4.5 Análisis del estudio del estado del arte

De todas las tecnologías antes analizadas Doctrine en su versión 2 presenta un mayor rendimiento en comparación con otras tecnologías de persistencia de datos, también las potencialidades de otros mapeadores se ven reflejadas en esta versión, se escogió esta nueva versión del marco de persistencia porque mejora la estructura, el diseño y el rendimiento de la actual versión que usa Acaxia. Un elemento a tener en cuenta es el hecho de realizar las menores transformaciones posibles en el cambio de tecnología ya que el equipo de trabajo de Acaxia tiene experiencia trabajando con el marco de persistencia Doctrine, y un cambio muy brusco traería como consecuencia impartir cursos de capacitación para la nueva tecnología. Además de retrasos en la entrega de componentes y tareas y una baja productividad del equipo de desarrollo en los inicios del uso de la nueva tecnología.

1.5 Metodología para el desarrollo

Se tomó como metodología a seguir para el desarrollo el procedimiento propuesto por Inoelkis Velázquez Osorio, Lisandra Cordero Estrada y René Bauta en la investigación titulada “PROCEDIMIENTO PARA ACTUALIZACIÓN DE LA CAPA DE ACCESO A DATOS DEL MARCO DE TRABAJO SAUXE DE DOCTRINE 1.2.2 A DOCTRINE 2.2”. Esta metodología propone tres fases (inicial, implementación y prueba), por cada una de estas etapas se definen pasos a seguir para llegar a realizar la migración con éxito.

1.5.1 Etapa Inicial

Inicialmente se lleva a cabo el estudio de la arquitectura del MT Sauxe, donde se valora el contenido de cada una de las capas que contiene. Se realiza la integración de Doctrine 2

con el MT Sauxe y como consecuencia se hace necesaria la creación de directorios donde se añaden las clases y funcionalidades. Además se adicionan los parámetros de configuración y las nuevas librerías (Osorio et al., 2013).

1.5.2 Etapa de Implementación

En esta etapa se realiza la redefinición de la capa de acceso a datos en la cual se mapean las tablas y se definen las relaciones de la base de datos para la versión 2 de Doctrine. Además se re-implementan los métodos y consultas de esta capa, ejecutándose la restructuración de la capa del negocio (Osorio et al., 2013).

1.5.3 Etapa de Pruebas

En esta etapa se hacen pruebas a cada una de las funcionalidades migradas para corregir incoherencias o errores en el código (Osorio et al., 2013).

1.6 Herramientas para el desarrollo

Se brinda una descripción de las herramientas a utilizar. El Departamento de Tecnología propone en el documento "*Vista de Entorno de Desarrollo Tecnológico del proyecto Sauxe_v2.2*" cuáles son estas herramientas.

1.6.1 PostgreSQL v9.1

PostgreSQL es un sistema gestor de base de datos relacional (SGBDR) orientado a objetos y de código abierto, publicado bajo la licencia BSD[11]. Es uno de los gestores de bases de datos más potentes del mercado compitiendo incluso con los gestores propietarios, siendo considerado una fuerte competencia para ellos incluso para el potente Oracle (Group, 2014).

A diferencia de otros gestores de bases de datos Postgres es compatible con todos los sistemas operativos incluyendo Linux, UNIX y Windows. Tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios idiomas). Incluye gran cantidad de tipos de datos, incluyendo INTEGER, numéricos, booleanos, CHAR, VARCHAR, DATE, INTERVAL, y TIMESTAMP. También es compatible con el almacenamiento de grandes objetos binarios, como imágenes, sonidos o videos.

[11] Por sus siglas en inglés Berkeley Software Distribution (Distribuidora de software Berkeley)

Cuenta con interfaces de programación nativas para C / C + +, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre otros (Group, 2014).

1.6.2 RapidSVN v0.12.0

RapidSVN es la interfaz gráfica de usuario para el Sistema de Control de Versiones Subversion escrito en C++. Su objetivo es ser simple para los principiantes, pero lo suficiente flexible como para aumentar la productividad de los usuarios experimentados de Subversion (RapidSVN, 2012).

1.6.3 NetBeans v7.4

La mayoría de los desarrolladores reconocen el NetBeans[12] IDE como la idea original libre de Java IDE. El NetBeans IDE proporciona soporte para otros lenguajes como PHP, JavaFX, C / C + +, JavaScript (Corporation, 2013).

NetBeans es un proyecto de código abierto desde el 2000 por el patrocinador Sun Microsystems hasta 2010 que se convirtió en subsidiaria de Oracle, NetBeans está dedicado a proveer productos de desarrollo de software que permiten el desarrollo de forma rápida, eficaz y sencilla mediante el aprovechamiento de los puntos fuertes de la plataforma Java (Corporation, 2013).

1.6.4 Apache v2.2

El proyecto Apache Server es un esfuerzo por desarrollar y mantener un servidor HTTP[13] de código abierto para sistemas operativos modernos incluyendo UNIX y Windows NT. El objetivo de este proyecto es proporcionar un seguro, eficiente y extensible servidor que proporcione servicios HTTP en sincronización con los estándares HTTP actuales (Foundation., 2014).

[12] NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.

[13] Por sus siglas en inglés Hypertext Transfer Protocol (protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web.

El servidor HTTP Apache implementa los protocolos más recientes, incluyendo HTTP/1.1 (RFC2616[14]), es altamente configurable y extensible con módulos de terceros que pueden ser personalizados. API[15] proporciona el código fuente completo y viene con una licencia sin restricciones. Se ejecuta en Windows 2000, NetWare 5.x y anteriormente OS/2, y la mayoría de las versiones de Unix. Alienta comentarios de los usuarios mediante nuevas ideas, implementa informes de errores, parches y con frecuencia muchas características solicitadas, incluyendo: bases de datos DBM[16] así como bases de datos relacionales y LDAP[17] para la autenticación (Foundation., 2014).

Permite configurar fácilmente las páginas protegidas con contraseña con un enorme número de usuarios autorizados, sin atascar el servidor. Permite la personalización de respuestas de los errores y problemas. Se facilita la configuración de los archivos, o incluso scripts CGI, que son devueltos por el servidor en respuesta a los errores y problemas, por ejemplo configurar una secuencia de comandos para interceptar 500 errores de servidor y realizar diagnósticos sobre la marcha (Foundation., 2014).

1.6.5 MT Zend v1.11

El principal patrocinador de MT Zend es Zend Technologies, pero muchas compañías han aportado componentes o características importantes para el MT, algunas de las compañías son Microsoft, Google y Strikelron. El MT Zend es de código abierto para el desarrollo de aplicaciones y servicios web usando PHP 5.3+. Zend usa 100% código orientado a objetos y utiliza la mayor parte de las nuevas características de PHP 5.3, utiliza los espacio de nombres (en inglés namespaces), finales de enlace estático, funciones lambda y cierres (Zend-Technologies-Ltd, 2014).

Con el espacio de nombres de servicio Zend, implementan bibliotecas de cliente para simplemente acceder a los servicios web más populares disponibles. Cualesquiera que sean sus necesidades de aplicación, es muy probable que encuentre un componente de

[14] RFC 2616 que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse.
[15] Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
[16] Por sus siglas en inglés Data Bases Manager (Gestor de bases datos).
[17] Por sus siglas en inglés Lightweight Directory Access Protocol (en español Protocolo Ligero de Acceso a Directorios).

MT Zend que se puede utilizar para reducir drásticamente el tiempo de desarrollo con una base completamente probada (Zend-Technologies-Ltd, 2014).

1.6.6 MT Sauxe v2.2

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 capa de acceso a datos 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 (Baryolo, 2001).

1.6.7 Subversion

Subversion 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 el historial de como cambiaron los datos. En este sentido, muchos especialistas piensan en un sistema de control de versiones como una especie de "máquina del tiempo" (Collins-Sussman et al., 2012).

Subversion puede acceder al repositorio a través de redes, lo que permite que pueda ser utilizado por personas en diferentes equipos. A cierto nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto a través del cual se producen todas las modificaciones. Y debido a que el trabajo se versiona, la calidad es la compensación por la pérdida de ese conducto. Si algún cambio realizado en los datos es incorrecto, simplemente se deshace ese cambio (Collins-Sussman et al., 2012).

1.6.8 JMeter

La aplicación de escritorio Apache JMeter™ es un software de código abierto, una aplicación 100% Java puro diseñada para cargar el comportamiento funcional de prueba y

medir el desempeño. Originalmente fue diseñado para aplicaciones de pruebas Web, pero desde entonces se ha expandido a otras funciones de prueba (Foundation, 2013).

Apache JMeter incluye (Foundation, 2013):

1. La posibilidad de cargar y hacer pruebas de rendimiento a diferentes tipos de servidor / protocolo:
    - Web - HTTP, HTTPS[18]
    - SOAP[19]
    - FTP
    - Bases de datos vía JDBC[20]
    - LDAP
    - Message-oriented middleware (MOM) vía JMS
    - Correo- SMTP, POP3 y IMAP
    - MongoDB[21](NoSQL[22])
    - Comandos nativos
    - TCP
2. Marco multihilos completos que permite el muestreo simultáneo de muchos hilos y de las diferentes funciones de los grupos de hilos separados.
3. Portabilidad completa y 100% de pureza Java.
4. Diseño cuidadoso de las interfaces gráficas de usuario permite la construcción de planes de pruebas y depuración más rápida.
5. El almacenamiento en cache y el análisis fuera de línea / Reproducción de los resultados de las pruebas.
6. Núcleo altamente extensible:
    - Probadores conectables permiten capacidades de prueba ilimitada.
    - Varias estadísticas de carga pueden ser elegidos con temporizadores enchufables.

[18] Por sus siglas en inglés Hypertext Transfer Protocol Secure (Protocolo seguro de transferencia de hipertexto).
[19] Por sus siglas en inglés Simple Object Access Protocol (Protocolo de Acceso de Objeto simple). Es un protocolo estándar que define como dos objetos en diferentes procesos pueden comunicarse.
[20] Por sus siglas en inglés Java Database Connectivity (conexión a bases de datos desde java).
[21] MongoDB es un sistema de base de datos NoSQL orientado a documentos.
[22] **NoSQL** (a veces llamado "**no solo SQL**") es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico.

- Análisis de datos y plugins de visualización permiten una gran extensibilidad y personalización.
- Las funciones pueden ser utilizados para proporcionar la entrada dinámica a una prueba o proporcionar la manipulación de datos.

1.7 Lenguajes para el desarrollo de la solución

Se dará una breve explicación de cada uno de los lenguajes utilizados en la realización de la solución propuesta.

1.7.1 Lenguaje PHP

Es un lenguaje que permite la generación dinámica de contenidos en un servidor web. El significado de sus siglas es HyperText Preprocessor. El código **PHP** puede incluirse dentro del código HTML de la página, delimitado por ***<?php*** para abrir la etiqueta y ***?>*** para cerrarla. Es un lenguaje de código abierto interpretado, de alto nivel. 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. Su rapidez en la ejecución y los bajos requerimientos de consumo en los sistemas donde es desplegado lo hacen uno de los preferidos por los desarrolladores. Se integra perfectamente a la mayoría de los Sistemas Gestores de Bases de Datos (Gutiérrez, 2004).

1.7.2 Lenguaje de Consulta de Doctrine (DQL)

DQL es el lenguaje de consulta de *Doctrine* y es un objeto derivado del lenguaje de consulta que es muy similar al lenguaje de consulta Hibernate (HQL) o al lenguaje de consulta persistente de Java (JPQL). Es insensible a mayúsculas y minúsculas, salvo en nombres de clase, campos y espacios de nombres, en los cuales sí distingue entre mayúsculas y minúsculas (Doctrine-Team, 2011).

DQL como un lenguaje de consulta tiene construcciones SELECT, UPDATE y DELETE que asignan a sus correspondientes tipos de declaraciones SQL. Las instrucciones INSERT no se permiten, porque las entidades y sus relaciones se tienen que introducir en el contexto de la persistencia a través de ***EntityManager#persist()*** para garantizar la coherencia de los objetos del modelo (Doctrine-Team, 2011).

Las instrucciones SELECT de DQL son una forma muy poderosa de recuperar partes del modelo de dominio que no son accesibles a través de asociaciones. Además permite recuperar entidades y sus asociaciones en una sola consulta SQL en la cual se puede hacer una gran diferencia en el rendimiento en contraste al uso de varias consultas (Doctrine-Team, 2011).

Las instrucciones UPDATE y DELETE de DQL ofrecen una manera de ejecutar cambios masivos en las entidades del modelo de dominio. Esto, a menudo es necesario cuando no se puede cargar en memoria todas las entidades afectadas por una actualización masiva (Doctrine-Team).

1.7.3 Lenguaje de Consulta Estructurado (SQL)

El SQL es el lenguaje estándar ANSI/ISO[23] de definición, manipulación y control de bases de datos relacionales. Es un lenguaje declarativo: solo hay que indicar que se quiere hacer. En cambio, en los lenguajes procedimentales es necesario especificar como hay que hacer cualquier acción sobre la base de datos. El SQL es un lenguaje muy parecido al lenguaje natural; concretamente, se parece al inglés, y es muy expresivo. Por estas razones, y como lenguaje estándar, el SQL es un lenguaje con el que se puede acceder a todos los sistemas relacionales comerciales (Paré et al., 2007).

1.8 Conclusiones Parciales

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 usada en la capa de acceso a datos del Sistema Integral de Seguridad Acaxia, 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 mejor favorecido en cuanto a rendimiento, prestaciones y también mejora la estructura y el diseño.

Se hizo un estudio de las principales características y las fases de la metodología a utilizar en la migración. Creando las bases para definir las tareas a realizar por cada una de las fases.

[23] definición del estándar del leguaje de consulta de base de datos. Este permite hacer consulta sobre una base de datos.

Capítulo 2: Migración de la capa acceso a datos

2.1 Introducción

En este capítulo se explicará detalladamente cómo se realizó la migración de la capa de acceso a datos del Sistema Integral de Seguridad Acaxia.

2.2 Etapa Inicial

En esta etapa se realiza la integración de las diferentes tecnologías presentes en el desarrollo de la migración. Llevándose a cabo para esto la inclusión de las bibliotecas necesarias para doctrine 2 en el directorio **lib\Doctrine**, también es necesario la creación y configuración del fichero **Doctrine.php** en el directorio **lib/ZendExt/App/** este fichero es utilizado para la creación de las instancias del **EntityManager**. Otros ficheros necesarios son **apps\config.php** y **ZendExt\Aspect\TransactionManager,** en este último es preciso modificar el método **openConections()** para hacer uso de los parámetros definidos en el fichero **config.php**.

2.2.1 Flujo de datos en la capa acceso a datos

Todo comienza cuando el controlador hace una llamada a una funcionalidad de la clase model, esta invoca el método **openConecction()** para obtener la instancia del **EntityManager**. Seguidamente se hace una llamada a la clase Entity mediante **getRepository()** pasándole el espacio de nombre para obtener el repositorio correspondiente y así poder acceder a los métodos de las clases Repository, a la vez que se devuelve el resultado de la consulta **DQL** hacia el model, esta a su vez le envía los resultados a la clase controladora que los solicitó. Este proceso se ve claramente en la **Figura *3***.

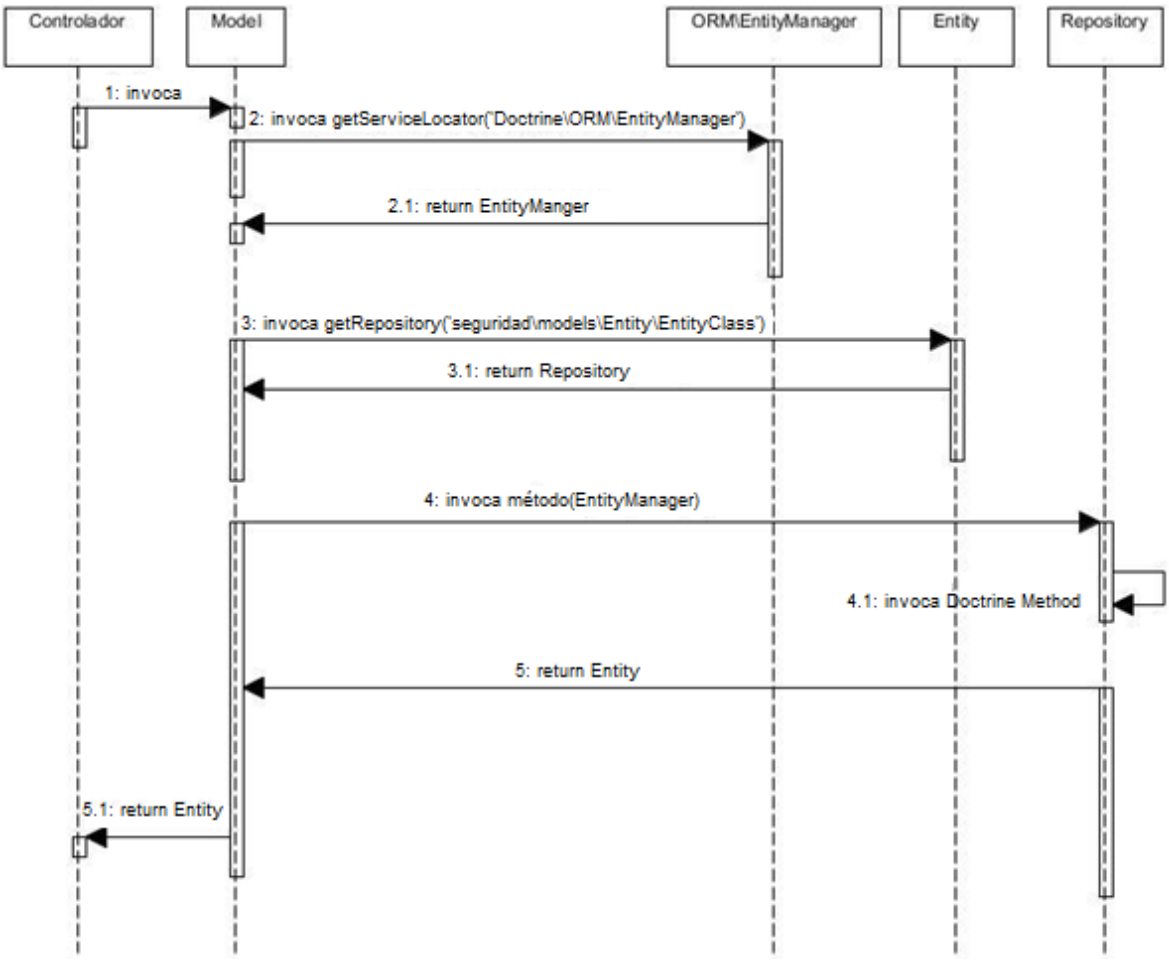


Figura 3: Flujo de datos en la capa acceso a datos

2.2.2 Estructura de carpetas

En la nueva versión Doctrine2 se define una nueva estructura de carpetas, se crean dos nuevas carpetas (***Proxy*** y ***Repository***) y por decisión interna se decidió introducir una tercera que sería ***Entity*** con el fin de una mejor organización de los ficheros de mapeo **Figura *4***.

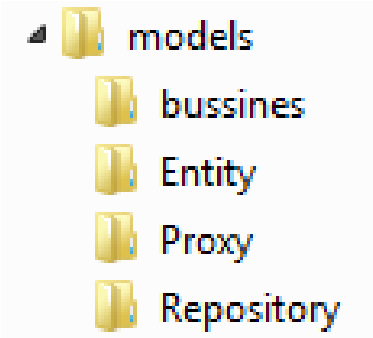


Figura 4: Nueva estructura de carpetas.

En la **Figura *5*** se evidencian las relaciones entre la estructura de carpetas propuestas por las versiones 1.x y 2.x de Doctrine.

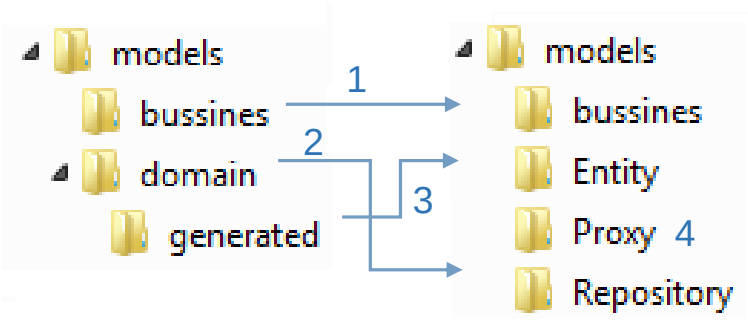


Figura 5: Relación entre las estructuras de carpetas.

1. Los ficheros localizados en la carpeta bussines que son los encargados de manejar el negocio de la capa de acceso a datos se mantienen en este directorio (Osorio and Estrada, 2013).
2. Las funcionalidades con las consultas que se encuentran en los ficheros localizados en la carpeta *domain* la deben realizar los ficheros ubicados dentro del directorio Repository, estas se deben modificar de acuerdo a la nueva estructura de Doctrine 2 (Osorio and Estrada, 2013). En esta carpeta se encuentran los ficheros Repository generados como resultado del mapeo. Por cada tabla en la base de datos se genera un fichero Repository, su nomenclatura está compuesta por el nombre del fichero de mapeo de la tabla y Repository al final. Estos ficheros son clases que extienden de la clase Doctrine\ORM\EntityRepository (Osorio and Estrada, 2013).

3. Los ficheros que representan las tablas mapeadas ubicados en generated son generados con su nueva estructura 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 no son más que objetos que se ponen en el lugar del objeto real, se utilizan para realizar varias funciones, pero 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 (Osorio and Estrada, 2013).

2.3 Etapa Implementación

En esta etapa se realizó la redefinición de la capa de acceso a datos en la cual se mapearon las tablas y se definieron las relaciones de la base de datos para la versión 2.2.0 de Doctrine. Además se re-implementaron los métodos y consultas de esta capa.

2.3.1 Redefinir capa de acceso a datos

Debido a las diferencias existentes entre las versiones 1 y 2 de Doctrine se hizo necesario realizar cambios en las capas. En la capa de acceso a datos se generó una nueva estructura de carpetas. Se hizo necesario realizar el mapeo de las tablas adaptándolas a la nueva estructura de Doctrine 2 y la migración manual de cada una de las clases ya mapeadas con Doctrine 1.

2.3.2 Redefinir capa de negocios

En esta actividad no se redefinió en su totalidad la capa de negocio. Se realizó la re-implementación de las funcionalidades adaptándolas a la nueva estructura de la capa de acceso a datos y la creación o definición de directorios y ficheros.

Debido a que todo el negocio se debe encontrar en las clases model se redefinió la forma en que se llaman las funcionalidades. En Doctrine 1 las funcionalidades se encontraban en los ficheros de la carpeta domain es por eso que las llamadas a estos desde el negocio varían respecto a la versión actual.

Doctrine 1

La **Figura *6*** muestra como se hacen las llamadas a las funcionalidades (estáticas o no) que se encontraban en los ficheros de mapeo generados por Doctrine 1:

```
public function authenticateUser($user, $password) {
    $observ = SegUsuarioDatSerautenticacion::verificarpass($user);
```

Figura 6: Llamada a las funcionalidades Doctrine 1

Doctrine 2

La **Figura *7*** muestra como se hacen las llamadas a las funcionalidades (estáticas o no) que se encuentran en las clases model dentro del directorio bussines. En este caso las llamadas a las funcionalidades se pueden realizar de la misma manera que con la versión anterior agregándole model al final del nombre de la clase o se puede crear una instancia de la clase a la que se hace referencia y luego se llama a la funcionalidad:

```
public function authenticateUser($user, $password) {
    $userinstance = new SegUsuarioModel();
    $verificar = $userinstance->verificarpass($user);
```

Figura 7: Llamada a las funcionalidades Doctrine 2.

2.3.3 Formas de mapeo

Entre Doctrine 1 y Doctrine 2 existen diferencias a la hora del mapeo de las tablas de la base de datos. Doctrine 1 genera las entidades extendiendo de una genérica llamada Doctrine_Record, ya que en esta contiene funciones predefinidas dentro del marco de persistencia de datos Doctrine. A diferencia Doctrine 2 define su propio conjunto de anotaciones en el código (docblock[24]) para mapear la información y suministrar asignación de metadatos objeto relacional.

[24] es un tipo especial de comentario en el código que puede proporcionar información detallada acerca de un elemento.

2.3.3.1 Definir tablas

Para definir el mapeo de una tabla en Doctrine 1.x se genera una clase que extiende de una clase base llamada Doctrine_Record y una funcionalidad llamada ***setTableDefinition***. Por su parte Doctrine 2 define el mapeo de una tabla a través de la anotación *@Table* y se debe indicar además que esta es una entidad *@Entity* **Figura *8* y Figura *9***.

Doctrine 1

```
abstract class BaseDatAccion extends Doctrine_Record
{
  public function setTableDefinition()
  {
    $this->setTableName('mod_seguridad.dat_accion');
```

Figura 8: Definir clase en Doctrine 1.

Doctrine 2

```
namespace seguridad\models\Entities;

/**
 * DatAccion
 *
 * @Table(name="mod_seguridad.dat_accion")
 * @Entity
 */
class DatAccion
```

Figura 9: Definir clase en Doctrine 2.

2.3.3.2 Definir columnas

Para definir columnas Doctrine 1 hace una llamada a la función *hasColumn* y en ella define todos los atributos del campo como nombre, tipo de datos, si es llave primaria y la no nulidad del atributo. Doctrine 2 por su parte lo hace mediante varias anotaciones como son *@Column, @Id* si el atributo fuera llave primaria **Figura *10*** y **Figura *11***.

Doctrine 1

```
    $this->hasColumn('icono', 'string', 250, array (
  'ntype' => 'varchar',
  'alltypes' =>
  array (
    0 => 'string',
  ),
  'fixed' => false,
  'notnull' => false,
  'primary' => false,
));
```

Figura 10: Definir columna en Doctrine 1.

Doctrine 2

```
/**
 * @var float
 *
 * @Column(name="idaccion", type="decimal", nullable=false)
 * @Id
 * @GeneratedValue(strategy="SEQUENCE")
 * @SequenceGenerator(sequenceName="dat_accion_idaccion_seq",
 * allocationSize=1, initialValue=1)
 *

 */

private $idaccion;
```

Figura 11: Definir columna en Doctrine 2.

2.3.3.3 Funcionalidades en las clases Repository

En la versión 1 de Doctrine estas consultas quedaban en la carpeta domain, quedando ahora en la nueva versión en la carpeta Repository. Las funcionalidades quedaban de la manera que se muestra en las **Figura *12* y Figura *13*.**

Las principales diferencias son que Doctrine 2 recibe en todas las funcionalidades el parámetro **$_em** que es la variable que contiene todo lo referente con la conexión, también

en esta versión se usa el método **setParameter()** para definir los parámetros necesarios para la sentencia DQL.

Doctrine 1

```
static public function obtenerRolesBuscado($rolbuscado,$limit,$start,$idusuario) {
    $query = Doctrine_Query::create();
    $roles = $query
    ->select('DISTINCT (r.idrol),r.denominacion,
    r.abreviatura, r.descripcion, sr.idusuario')
    ->from('SegRol r')
    ->leftJoin('r.DatEntidadSegUsuarioSegRol sr ON r.idrol = sr.idrol
    AND sr.idusuario = ?', $idusuario)
    ->where("r.denominacion like '$rolbuscado%'")
    ->orderby('r.idrol')
    ->limit($limit)
    ->offset($start)
    ->execute();
    return $roles;
}
```

Figura 12: Definir funcionalidad en Doctrine 1.

Doctrine 2

```
public function obtenerrolBuscado($filtroDominio, $denominacion, $limit, $start, $_em)
{

    $result = $_em
        ->createQuery("select DISTINCT (r.idrol),r.denominacion, r.abreviatura,
        r.descripcion from seguridad\models\Entity\SegRolNomDominio rd
        join rd.idrol r
        where r.denominacion
        LIKE '%$denominacion%'
        and rd.iddominio =?1
        order by r.idrol")
        ->setParameter(1, $filtroDominio)
        ->setMaxResults($limit)
        ->setFirstResult($start)
        ->getResult();


    return $result;
}
```

Figura 13: Definir funcionalidad en Doctrine 2.

2.3.3.4 Formatos del resultado de una consulta

Otro método usado en esta versión es el **getResult()** que este hace la función del **execute()** en Doctrine 1. También define el formato en el que se devuelve el resultado de una consulta. DQL SELECT puede estar influenciado por el así llamado modo de hidratación. Un modo de hidratación especifica de manera particular en qué se transforma un conjunto de resultados SQL. Cada modo tiene su propio método de hidratación dedicado en la clase Query:

- **Query#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).
- **Query#getSingleResult():** recupera un único objeto. Si el resultado contiene más de un objeto, lanza una excepción. La distinción puro/mixto no aplica.
- **Query#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.
- **Query#getScalarResult():** recupera un resultado plano/rectangular del conjunto de valores escalares que puede contener datos duplicados. La distinción puro/mixto no aplica.
- **Query#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.

2.3.3.5 Funcionalidades clases model

Las clases model son las encargadas de manejar el negocio, de hacer las llamadas a la capa de acceso a datos en Doctrine 2 y transformar los datos que esta devuelve para enviarlos a las clases controladoras.

Doctrine 1

Contiene métodos mágicos predefinidos ejemplo **delete()** y **save()** como se muestra en la **Figura 14**.

```
function insertarparametro($parametro)
{
 $parametro->save();
}
```

Figura 14: Definir funcionalidad clase model Doctrine 1.

En Doctrine 2 se hace una instancia de la clase ***TransactionManager*** para obtener una instancia de ***EntityManager*** y obtener los parámetros de la conexión, una vez realizado esto se hace una llamada al método ***getRepository()*** pasándole como parámetro el espacio de nombre de la clase de la cual se quiere obtener el Repository y finalmente la llamada a la funcionalidad como muestra la **Figura 15**.

Doctrine 2

```
public function existecertificado($idusuario)
{
        $mg = ZendExt_Aspect_TransactionManager::getInstance();
        $_em = $mg->getConnection('seguridad');
        $result = $_em  ->getRepository('seguridad\models\Entity\SegCertificado')
                        ->existecertificado($idusuario, $_em);
        return $result;
}
```

Figura 15: Definir funcionalidad clase model Doctrine 2.

En este ejemplo se llama a la funcionalidad ***existecertificado($iddusuario, $_em)*** que se encuentra en la clase de namespace ***seguridad\models\Entity\SegCertificado.***

2.3.4 Relaciones entre tablas

Entre las tablas de la base de datos existen relaciones, las cuales se representan en los ficheros de mapeo dependiendo de la cardinalidad de la relación, estas se ven claramente en el **Anexo *1***.

2.3.4.1 Lado propietario y lado inverso

Cuando se asignan asociaciones bidireccionales, es importante entender el concepto del "lado propietario" y el "lado inverso". Se aplican las siguientes reglas generales (Doctrine-Team, 2011):

- Las relaciones pueden ser bidireccionales o unidireccionales.
- Una relación bidireccional tiene tanto un lado propietario como un lado inverso.
- Una relación unidireccional solo tiene un lado propietario.
- El lado propietario de una relación determina las actualizaciones a la relación en la base de datos.

Las siguientes reglas se aplican a las asociaciones *bidireccionales*:

- El lado inverso de una relación bidireccional se debe referir a su lado propietario usando el atributo **mappedBy** de las declaraciones de asignación **OneToOne**, **OneToMany** o **ManyToMany**. El atributo **mappedBy** designa el campo en la entidad, que es el propietario de la relación.
- El lado propietario de una relación bidireccional se debe referir a su lado inverso usando el atributo **inversedBy** de la declaración de asignación **OneToOne**, **ManyToOne** o **ManyToMany**. El atributo **inversedBy** designa el campo en la entidad, que es el lado inverso de la relación.
- El lado muchos en las relaciones bidireccionales **OneToMany**/**ManyToOne** debe ser la parte propietaria, por lo tanto el elemento **mappedBy** no se puede especificar en el lado **ManyToOne**.
- Para relaciones bidireccionales **OneToOne**, la parte propietaria concuerda a la parte que contiene la clave externa correspondiente (@**JoinColumn**(s)).
- Para relaciones bidireccionales **ManyToMany** cualquiera puede ser el lado propietario (la parte que define el @**JoinTable** y/o no usa el atributo **mappedBy**, utilizando así una línea predeterminada de unión de tabla).

El "lado propietario" y el "lado inverso" son conceptos técnicos de la tecnología ORM, no conceptos del modelo de dominio. Lo que se considera como el lado propietario en el modelo del dominio puede ser diferente al lado propietario de Doctrine. Estos no están relacionados.

2.3.4.2 Relaciones utilizadas

@OneToOne **unidireccional**

Atributos requeridos:

targetEntity: define cuál será la clase con la que se relacionará.

Atributos opcionales:

Cascade: opciones de cascada.

Fetch: *Lazy* o *Eager.*

inversedBy: especifica cuál atributo de la entidad será el inverso en la relación.

Un ejemplo es la relación entre las clases ***SegUsuarioDatSerautenticacion*** y ***DatServidor,*** que se muestra en la **Figura 16**.

```
class SegUsuarioDatSerautenticacion {

    /**
     * @var \DatServidor $idservidor
     *
     *
     * @Id
     * @OneToOne(targetEntity="DatServidor")
     * @JoinColumns({
     *   @JoinColumn(name="idservidor", referencedColumnName="idservidor")
     * })
     */
    private $idservidor;
```

Figura 16: Relación **@OneToOne** en la clase **SegUsuarioDatSerautenticacion.**

En la clase ***DatServidor*** no se hace referencia al nuevo servidor autenticado, por lo tanto es unidireccional.

@ManyToMany **unidireccional**

Atributos requeridos:

targetEntity: define cuál será la clase con la que se relacionará.

Atributos opcionales:

mappedBy: esta opción especifica que el nombre de la propiedad del *targetEntity* es el lado propietario de esta relación. Es un atributo requerido para el lado inverso de una relación.
inversedBy: especifica cuál atributo de la entidad será el inverso en la relación.
cascade: opciones de cascada.
fetch: *Lazy* o *Eager.*
indexBy: define un índice para la colección por un campo en la entidad.
Un ejemplo es la relación entre ***SegUsuario*** donde se establece la misma y ***SegRol*** donde no se hace ninguna alusión a ella **(Figura *17*)**.

```
class SegUsuario {
    /**
     * @ManyToMany(targetEntity="SegRol ", mappedBy="roles")
     */
    private $roles;

    public function __construct() {
        $this->roles = new \Doctrine\Common\Collections\ArrayCollection();
    }
```

Figura 17: Relación **@ManyToMany** en la clase **SegUsuario.**

@ManyToOne **unidireccional**

Esta relación trabaja casi completamente como un ***@OneToOne*** con la diferencia de que en este no es obligatoria la opción adicional ***@JoinColunm*** mientras que en el ***@ManyToOne*** sí lo es.

En la **Figura *18*** se muestra un ejemplo de este tipo de relación entre las clases ***SegCertificado*** donde se establece y ***SegUsuario*** donde no se hace ninguna alusión a ella.

```
class SegCertificado
{
    /**
     * @var \SegUsuario
     *
     * @ManyToOne(targetEntity="SegUsuario")
     * @JoinColumns({
     *   @JoinColumn(name="idusuario", referencedColumnName="idusuario")
     * })
     */
    private $idusuario;
```

Figura 18: Relación **@ManyToOne** en la clase **SegCertificado.**

@ManyToMany **bidireccional**

Ejemplo de este tipo de relación es la existente entre ***DatSistema*** y ***SegRol*** que se establece como propietaria de la relación, evidenciándose en la **Figura *19***.

```
class SegRol {
    /**
     * @var \Doctrine\Common\Collections\Collection
     *
     * @ManyToMany(targetEntity="DatSistema", inversedBy="idrol")
     * @JoinTable(name="mod_seguridad.dat_sistema_seg_rol",
     *   joinColumns={
     *     @JoinColumn(name="idrol", referencedColumnName="idrol")
     *   },
     *   inverseJoinColumns={
     *     @JoinColumn(name="idsistema", referencedColumnName="idsistema")
     *   }
     * )
     */
    private $idsistema;

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

Figura 19: Relación **@ManyToMany** en la clase **SegRol.**

En la no propietaria (***DatSistema***) se hace referencia a la que lo es (***SegRol***) **Figura 20.**

```
class DatSistema {
    /**
     * @var \Doctrine\Common\Collections\Collection
     *
     * @ManyToMany(targetEntity="SegRol", mappedBy="idsistema")
     */
    private $idrol;

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

Figura 20: Relación **@ManyToMany** en la clase **DatSistema.**

2.4 Etapa de pruebas

En esta etapa se hicieron pruebas a cada una de las funcionalidades migradas para corregir cualquier incoherencia o error en el código.

2.4.1 Pruebas realizadas

Se realizaron pruebas de caja negra mediante la técnica de particiones equivalentes. Se montó la aplicación en una estación de trabajo y de otras en la misma subred se conectaron a la aplicación para realizar las pruebas. Definiendo un grupo de especialistas para detectar no conformidades en la aplicación.

Tabla II: Iteraciones pruebas funcionales.

	1ra Iteración	*2da Iteración*
No conformidades	21	0

Se tomaron los casos de pruebas ya antes definidos para la liberación de la anterior versión de Acaxia definidos en el *expediente de proyecto*. En una primera iteración se detectaron un total de 21 no conformidades significativas en la aplicación quedando resueltas para la segunda iteración encontrándose un total de cero no conformidades. A la conclusión de

esta actividad se emitió un acta de aceptación por parte del el departamento de tecnología como constancia del cumplimiento de todos los requerimientos por parte de la aplicación.

2.4.2 Errores más comunes

Uno de los errores más comunes que se detectaron fue que las funcionalidades en el model cuando hacían la llamada al repositorio no se devolvían los datos en el formato correcto o no llegaba a hacer la llamada, ya que el **EntityManager** no devolvía el repositorio correspondiente.

Otros de los errores más comunes se detectaron en las consultas DQL, que no devolvían los datos requeridos o en el formato requerido por una mala re-implementación de la consulta.

2.5 Conclusiones parciales

En este capítulo se realizó la migración de la capa de acceso a datos del Sistema Integral de Seguridad Acaxia migrando un total de 224 ficheros de mapeo divididos en las 4 carpetas generadas por Doctrine 2; esta actividad fue la que más tiempo consumió, no por la complejidad, sino por el monto de ficheros de mapeo que se realizaron de forma manual. Se migró este módulo teniendo en cuenta las tres fases definidas (Inicial, Implementación y Pruebas) en la guía seleccionada.

Capítulo 3: Validación de la Migración

3.1 Introducción

La validación de la migración se realizará mediante pruebas de rendimiento al MT Sauxe específicamente al Sistema Integral de Seguridad en la versión anterior a la migración y en la versión posterior a la migración, según lo propone la guía seleccionada, tomando los resultados y comparándolos para verificar el aumento del rendimiento de la capa de acceso a datos implementada en Doctrine 2.

3.2 Pruebas de rendimiento

Las pruebas de rendimiento se hacen con el objetivo de verificar cuán rápido se realiza una tarea planificada, y hasta cuanto podrá soportar el sistema cuando un grupo de usuarios se conectan concurrentemente y comienzan a realizar peticiones al servidor, todas estas pruebas se hacen en ambientes simulados por herramientas como Jmeter.

Entre las pruebas de rendimiento se encuentran varios tipos de pruebas cada una con un objetivo diferente (Martínez, 2009):

- **Pruebas de carga:** intentarán validar que se alcanzan los objetivos de prestaciones a los que se verá sometido el sistema en un entorno productivo.
- **Pruebas de capacidad:** Su objetivo es encontrar los límites de funcionamiento del sistema y detectar el cuello de botella o elemento limitante para poder actuar en caso de ampliación del servicio.
- **Pruebas de estrés:** someten al sistema a una carga por encima de los límites requeridos de funcionamiento.
- **Pruebas de estabilidad:** comprueban que no existe degradación del servicio por un uso prolongado del sistema.
- **Pruebas de aislamiento:** provocan concurrencia sobre componentes aislados del sistema para tratar de detectar posibles errores en ellos.
- **Pruebas de regresión de rendimiento:** su objetivo es comprobar si se mantienen los niveles de rendimiento tras un cambio en el sistema, comparando el nivel de rendimiento con el que ofrecía con anterioridad.

3.3 Descripción de las pruebas

Según el autor Jakob Nielsen, en el libro “Usability Engineering” existen tres límites importantes en el tiempo de respuesta (Nielsen, 1994):

• 0,1 segundo: es el límite en el cual el usuario siente que está “manipulando” los objetos desde la interfaz de usuario.

• 1 segundo: es el límite en el cual el usuario siente que está navegando libremente sin esperar demasiado una respuesta del servidor.

• 10 segundos: es el límite en el cual se pierde la atención del usuario, si la respuesta tarda más de 10 segundos se deberá indicar algún mecanismo por el cual el usuario pueda interrumpir la operación.

El tiempo de respuesta está condicionado a los siguientes puntos (Díaz et al., 2008):

• El servidor testeado se encuentra en la misma red en la cual se realizaron las pruebas.

• Velocidad de conexión del servidor.

• Velocidad de conexión del cliente.

• Tiempo en el cual el navegador web tarda para dibujar la página (tiempo muy pequeño).

• Rendimiento de la red en el momento de la prueba.

Para poder evaluar los resultados se utilizaron tres componentes provistos por la herramienta Jmeter (Martínez, 2009):

- Árbol de resultados: este punto permite visualizar con detalle cada petición HTTP realizada analizando las cabeceras de la petición y de la respuesta obtenida. De este modo, se puede monitorizar que sucede a la hora de procesar cada petición
- Informe agregado: permite visualizar los resultados del test realizado, en una tabla. La tabla muestra los siguientes datos:
  - ✓ **URL :** etiqueta de la muestra
  - ✓ **#Muestras:** cantidad de hilos utilizados para la URL.
  - ✓ **Media:** tiempo promedio en milisegundos de la prueba.
  - ✓ **Mediana:** valor en tiempo del percentil 50.

- ✓ **Línea de 90%:** máximo tiempo utilizado por el 90% de la muestra, al resto de la misma le llevó más tiempo.
- ✓ **Min:** tiempo mínimo de la muestra de una determinada URL.
- ✓ **Max:** tiempo máximo de la muestra de una determinada URL.
- ✓ **%Error:** porcentaje de requerimientos con errores.
- ✓ **Rendimiento:** rendimiento medido en los requerimiento por segundo/minuto/hora.
- ✓ **KB/seg:** rendimiento medido en Kbytes por segundo.

- Aserción de respuesta: se utilizan para hacer comprobaciones adicionales sobre los samplers[25] verificando que las respuestas coinciden con un determinado patrón.

Para evaluar cada uno de los casos de prueba elaborados se definió por un equipo de especialistas que se realizarían con un total de 50 hilos simulando 50 usuarios conectados simultáneamente al sistema haciendo peticiones al servidor ya que este sería el peor escenario en el cual se encontraría el sistema. También se definieron los enlaces y las llamadas a servidor que estos 50 usuarios realizarían por cada uno de los casos de prueba.

Para calcular el tiempo total en que los usuarios estuvieron accediendo al sistema se utilizó la siguiente fórmula (Díaz et al., 2008):

$$\text{Tiempo Total} = \#Muestras * Media\ [ms]$$

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 (Díaz et al., 2008):

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

El resultado obtenido se expresa en minutos.

[25] Peticiones que realizarán los hilos al servidor.

3.4 Preparación de los casos de pruebas

Los casos de pruebas propuestos fueron elaborados agrupando varias vistas por cada uno de los casos prueba, quedando confeccionado un total de cuatro casos de prueba de la siguiente manera:

- Caso de prueba Configurar nomencladores:

  Este agrupa la acción de cargar, adicionar, modificar y eliminar de los nomencladores *Gestores de base de datos*, *Idiomas*, *Expresiones*.

- Caso de prueba Configurar servidores:

  Este agrupa la acción de modificar, adicionar, eliminar y cargar de las vistas *Servidores*, *Gestores de base de datos* y *Gestionar roles de base de datos*.

- Caso de prueba Configurar sistemas:

  Este agrupa la acción de adicionar, modificar, eliminar y cargar de las vistas *Sistemas*, *Funcionalidades*, *Acciones*.

- Caso de prueba Configurar usuarios:

  Este agrupa la acción de adicionar, eliminar, modificar y cargar de las vistas *Roles*, *Usuarios*.

3.5 Preparación del entorno para pruebas

Se trabajó en una estación de trabajo siendo la misma el servidor donde se montaron las diferentes versiones del marco de trabajo Sauxe también trabajando con bases de datos montadas localmente. La estación de trabajo cuenta con las siguientes características:

- Capacidad de memoria RAM: 4096MB (4GB)
- Capacidad de almacenamiento: 1024 GB (1TB)
- Microprocesador: core i3 a 3.3GHz
- Se trabajará sobre el sistema operativo Linux, distribución debian wheezy en su versión 7.

Para la realización de las pruebas se configuró el proxy que provee Jmeter para construir el camino de navegación que utilizarán los hilos para simular las peticiones **Figura 21**, se utilizó como navegador web el Firefox en su versión 29.0.1, se hizo necesario configurar el proxy del navegador para establecer la comunicación con el antes mencionado de la herramienta **Figura 22**.

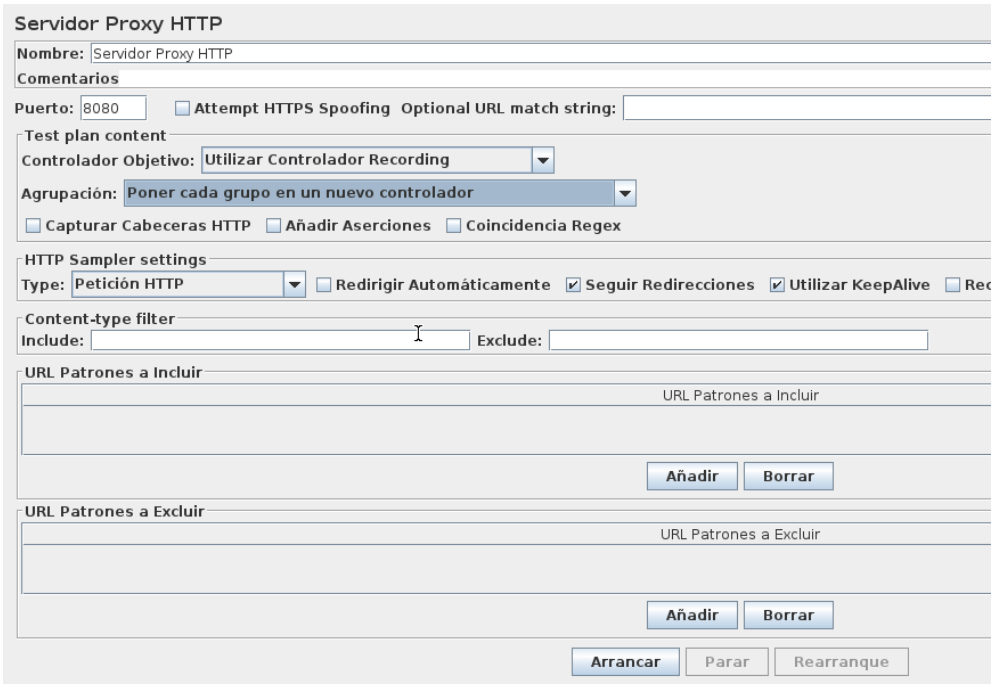


Figura 21: Configuración del proxy de la herramienta Jmeter.

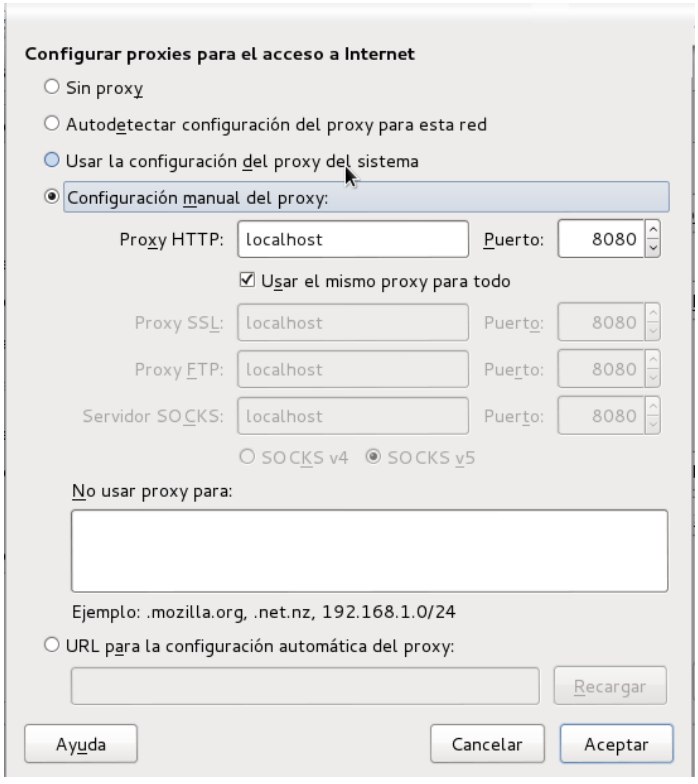


Figura 22: Configuración del proxy del navegador web Firefox.

3.6 Análisis de los Resultados de los casos de pruebas

3.6.1 Caso de prueba Configurar nomencladores:

- Acción cargar nomencladores tuvo como resultado:

Tabla III: Comparación doctrine 1 y doctrine 2 acción cargar nomenclador.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	413700	299100
Tiempo promedio de ejecución por hilo(min)	0.1379	0.0997

- Acción adicionar nomenclador tuvo como resultado:

Tabla IV: Comparación doctrine 1 y doctrine 2 acción adicionar nomenclador.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	1051200	762600
Tiempo promedio de ejecución por hilo(min)	0.3504	0.2542

- Acción modificar nomenclador tuvo como resultado:

Tabla V: Comparación doctrine 1 y doctrine 2 acción modificar nomenclador.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	417500	279300
Tiempo promedio de ejecución por hilo(min)	0.1392	0.0931

- Acción eliminar nomenclador tuvo como resultado:

Tabla VI: Comparación doctrine 1 y doctrine 2 acción eliminar nomenclador.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	1301300	644600
Tiempo promedio de ejecución por hilo(min)	0.4338	0.2149

En la **Figura 23** se comparan los tiempos de ejecución de cada escenario del caso de pruebas:

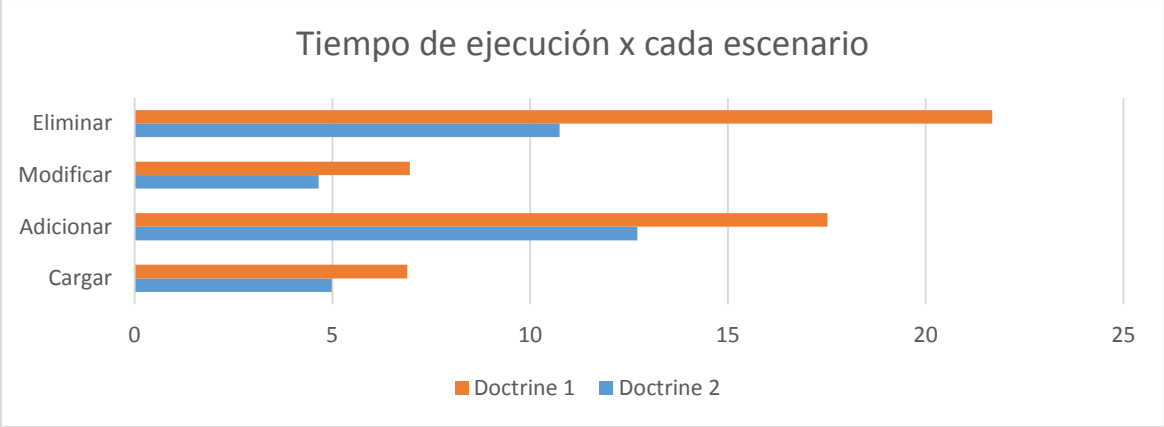


Figura 23: Comparación tiempo de ejecución caso de prueba configurar nomencladores.

Este caso de prueba concluyó con un % de errores de 0.

3.6.2 Caso de prueba Configurar Servidores

- Acción cargar servidor tuvo como resultado:

Tabla VII: Comparación doctrine 1 y doctrine 2 acción cargar servidor.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	551600	299700
Tiempo promedio de ejecución por hilo(min)	0.1839	0.0997

- Acción adicionar servidor tuvo como resultado:

Tabla VIII: Comparación doctrine 1 y doctrine 2 acción adicionar servidor.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	1226400	1143900
Tiempo promedio de ejecución por hilo(min)	0.4088	0.3813

- Acción modificar servidor tuvo como resultado:

Tabla IX: Comparación doctrine 1 y doctrine 2 acción modificar servidor.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	417500	279300
Tiempo promedio de ejecución por hilo(min)	0.1392	0.0931

- Acción eliminar servidor tuvo como resultado:

Tabla X: Comparación doctrine 1 y doctrine 2 acción eliminar servidor.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	1301300	644600
Tiempo promedio de ejecución por hilo(min)	0.4338	0.2149

En la **Figura 24** se comparan los tiempos de ejecución de cada escenario del caso de pruebas:

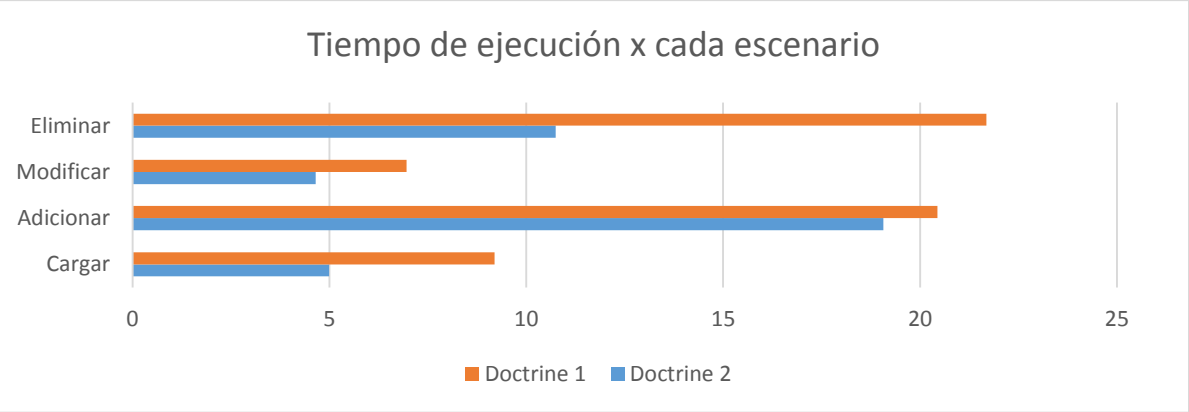


Figura 24: Comparación tiempo de ejecución caso de prueba configurar servidor.

Este caso de prueba concluyó con un % de errores de 0.

3.6.3 Caso de prueba Configurar Sistemas

- Acción cargar sistemas tuvo como resultado:

Tabla XI: Comparación doctrine 1 y doctrine 2 acción cargar sistema.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	3026000	780000
Tiempo promedio de ejecución por hilo(min)	1.0087	0.2600

- Acción adicionar sistemas tuvo como resultado:

Tabla XII: Comparación doctrine 1 y doctrine 2 acción adicionar sistema.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	1016400	750400
Tiempo promedio de ejecución por hilo(min)	0.3388	0.2501

- Acción modificar sistemas tuvo como resultado:

Tabla XIII: Comparación doctrine 1 y doctrine 2 acción modificar sistema.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	2211300	1306000
Tiempo promedio de ejecución por hilo(min)	0.7371	0.4353

- Acción eliminar sistemas tuvo como resultado:

Tabla XIV: Comparación doctrine 1 y doctrine 2 acción eliminar sistema.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	1949600	1042400
Tiempo promedio de ejecución por hilo(min)	0.6499	0.3475

En la **Figura 25** se comparan los tiempos de ejecución de cada escenario del caso de pruebas:

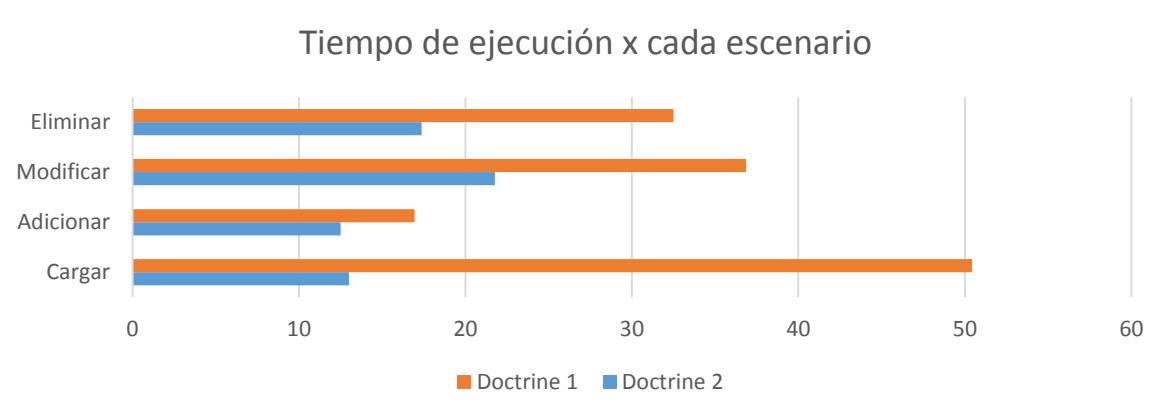


Figura 25: Comparación tiempo de ejecución caso de prueba configurar sistema.

Este caso de prueba concluyó con un % de errores de 0.

3.6.4 Caso de prueba Configurar Usuario

- Acción cargar usuario tuvo como resultado:

Tabla XV: Comparación doctrine 1 y doctrine 2 acción cargar usuario.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	162300	162000
Tiempo promedio de ejecución por hilo(min)	0.0541	0.0540

- Acción adicionar usuario tuvo como resultado:

Tabla XVI: Comparación doctrine 1 y doctrine 2 acción adicionar usuario.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	1191150	420000
Tiempo promedio de ejecución por hilo(min)	0.3970	0.1400

- Acción modificar usuario tuvo como resultado:

Tabla XVII: Comparación doctrine 1 y doctrine 2 acción modificar usuario.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	633300	350800
Tiempo promedio de ejecución por hilo(min)	0.2111	0.1169

- Acción eliminar usuario tuvo como resultado:

Tabla XVIII: Comparación doctrine 1 y doctrine 2 acción eliminar usuario.

	Doctrine 1	*Doctrine 2*
Tiempo total de ejecución(ms)	742000	269400
Tiempo promedio de ejecución por hilo(min)	0.2473	0.0898

En la **Figura 26** se comparan los tiempos de ejecución de cada escenario del caso de pruebas:

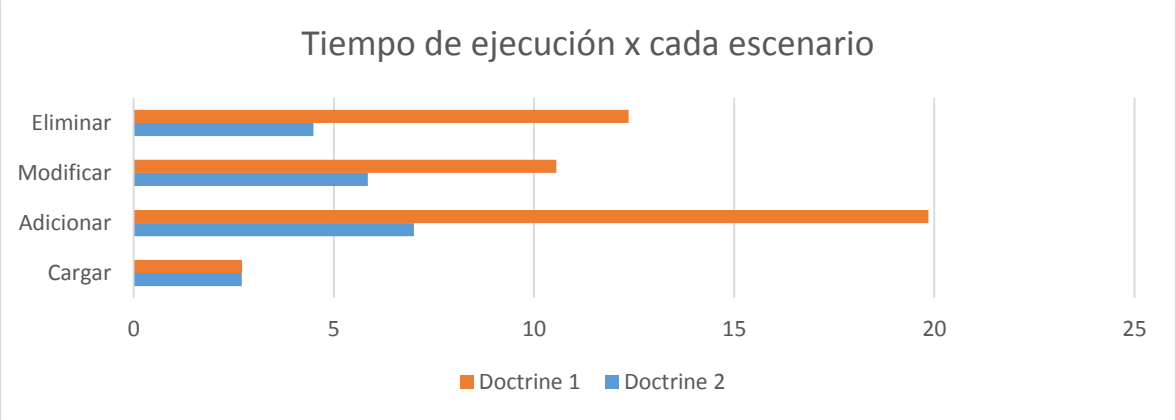


Figura 26: Comparación tiempo de ejecución caso de prueba configurar usuario.

Este caso de prueba concluyó con un % de errores de 0.

3.7 Resumen

En la **Figura 27** se recopiló la información de todas las pruebas para mejor visualización, tomándose como datos la media entre cada una de las acciones de los casos de pruebas.

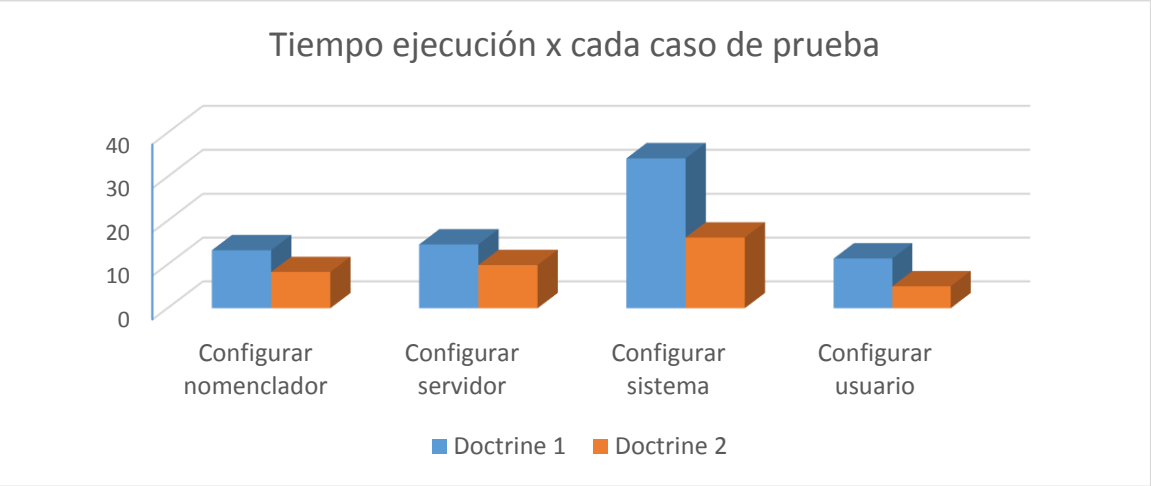


Figura 27: Resumen de resultados de los casos de prueba.

Los resultados obtenidos arrojaron que se obtuvo una disminución de los tiempos de respuesta por cada uno de los casos de pruebas definidos. En la **Tabla XIX** se muestra en por ciento la disminución de los tiempos de respuesta. Para el cálculo se estableció la fórmula:

$$\% \text{ disminuido} = 100 - \left[d2 * \frac{100}{d1}\right]$$

Tabla XIX: Disminución de los tiempos de respuesta de los casos de prueba.

Casos de pruebas	***% disminuido***
Configurar nomenclador	38 %
Configurar servidor	32 %
Configurar sistema	53 %
Configurar usuario	56 %
Promedio	45 %

3.8 Conclusiones parciales

Mediante las pruebas realizadas sobre el sistema utilizando la herramienta Apache Jmeter para realizar pruebas de rendimiento, se obtuvo como resultado un mejoramiento de un 45% en los tiempos de respuesta de la capa de acceso a datos del Sistema Integral de Seguridad Acaxia, quedando así validada la solución propuesta.

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 el Sistema Integral de Seguridad Acaxia, 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 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 Sistema Integral de Seguridad Acaxia, migrando un total de 224 ficheros de mapeo divididos en las 4 carpetas generadas por Doctrine 2. Se migró este sistema teniendo en cuenta las tres fases definidas (Inicial, Implementación y Pruebas) en la metodología seleccionada, y realizando las tareas propuestas por cada una de las fases.

Las pruebas realizadas sobre el sistema utilizando la herramienta Apache Jmeter para realizar pruebas de rendimiento arrojaron como resultado un mejoramiento en los tiempos de respuesta de la capa de acceso a datos del Sistema Integral de Seguridad Acaxia.

Recomendaciones:

1. Migrar las funcionalidades en desarrollo del Sistema Integral de Seguridad Acaxia.

Referencias bibliográficas:

BARYOLO, O. G. 2001. *SOLUCIÓN INFORMÁTICA DE AUTORIZACIÓN EN ENTORNOS MULTIENTIDAD Y MULTISISTEMA*.

COLLINS-SUSSMAN, B., B.W.F. & C.M.P. 2012. Control de Versiones con Subversion.

CORPORATION, O. 2013. *An Introduction to NetBeans* [Online]. Available: https://netbeans.org/about/.

DÍAS, D. 2012. Pruebas de rendimiento sobre Cedrux. *Universidad de las Ciencias Informáticas***,** 2-5.

DÍAZ, J., BANCHOFF TZANCOFF, C. M., RODRÍGUEZ, A. S. & SORIA, V. Usando Jmeter para pruebas de rendimiento. XIV Congreso Argentino de Ciencias de la Computación, 2008.

DOCTRINE-TEAM. 2010a. *Doctrine Performance Revisited* [Online]. Available: http://www.doctrine-project.org/blog/doctrine-performance-revisited.html.

DOCTRINE-TEAM 2010b. Guide to Doctrine1.2 for PHP. *SENSIO LABS***,** 13-16.

DOCTRINE-TEAM 2010c. Guide to Doctrine 2.0 for PHP. *SENSIO LABS***,** 13-16.

DOCTRINE-TEAM. 2011. *Doctrine 2 ORM v2.1 documentation* [Online]. Available: http://sf2-es.net16.net/doctrine/orm-documentation/reference/dql-doctrine-query-language.html.

ERCOLI, J. 2008. *Arquitectura de Sistemas Informáticos* [Online]. 2008. Available: http://metodologiasdesistemas.blogspot.com/ 2013].

FOUNDATION, A. S. 2013. *Apache JMeterTM* [Online]. Available: http://jmeter.apache.org/.

FOUNDATION., A. S. 2014. *About the Apache HTTP Server Project* [Online]. Available: http://httpd.apache.org/ABOUT_APACHE.html.

GROUP, P. G. D. 2014. *About* [Online]. Available: http://www.postgresql.org/about/.

GUTIÉRREZ, J. D. 2004. *Desarrollo Web con PHP 5 y MySQL*.

KING, G., BAUER, C., ANDERSEN, M. R., BERNARD, E., EBERSOLE, S. & FERENTSCHIK, H. 2013. Documentación de referencia de Hibernate. 329-350.

MARTÍN, C. C. 2004. *Introducción a Hibernate* [Online]. Available: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hibernate.

MARTÍNEZ, A. 2009. Manual Usuario Jmeter *Sociedad Informática del Gobierno Vasco*.

MESTRAS, J. P. 2009. Estructura de las Aplicaciones Orientadas a Objetos El patrón Modelo-Vista-Controlador (MVC). *Facultad de Informática UCM***,** 2.

MIRANDA, P., PRUDENZA, J., SEGUROLA, A., CALEGARI, D. & CORRAL, J. 2006. Arquitectura de Acceso a Datos en Sistemas Orientados a Objetos: Clasificación y Descripción de Mecanismos de Persistencia.

MOLINA, L. C. A., SEQUÉN, B. O. M. & COLMENARES, M. L. H. G. 2011. Django ORM. *Facultad de Ingeniería Universidad del Valle de Guatemala* 4.

NIELSEN, J. 1994. *Usability Engineering*, San Francisco, Morgan Kaufmann.

OSORIO, I. V. & ESTRADA, L. C. 2013. *MIGRACIÓN DE LA CAPA DE ACCESO A DATOS DEL MARCO DE TRABAJO SAUXE.* Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas, UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS.

OSORIO, I. V., ESTRADA, L. C. & BAUTA, R. R. 2013. PROCEDIMIENTO PARA ACTUALIZACIÓN DE LA CAPA DE ACCESO A DATOS DEL MARCO DE TRABAJO SAUXE DE DOCTRINE 1.2.2 A DOCTRINE 2.2. *Fordes*.

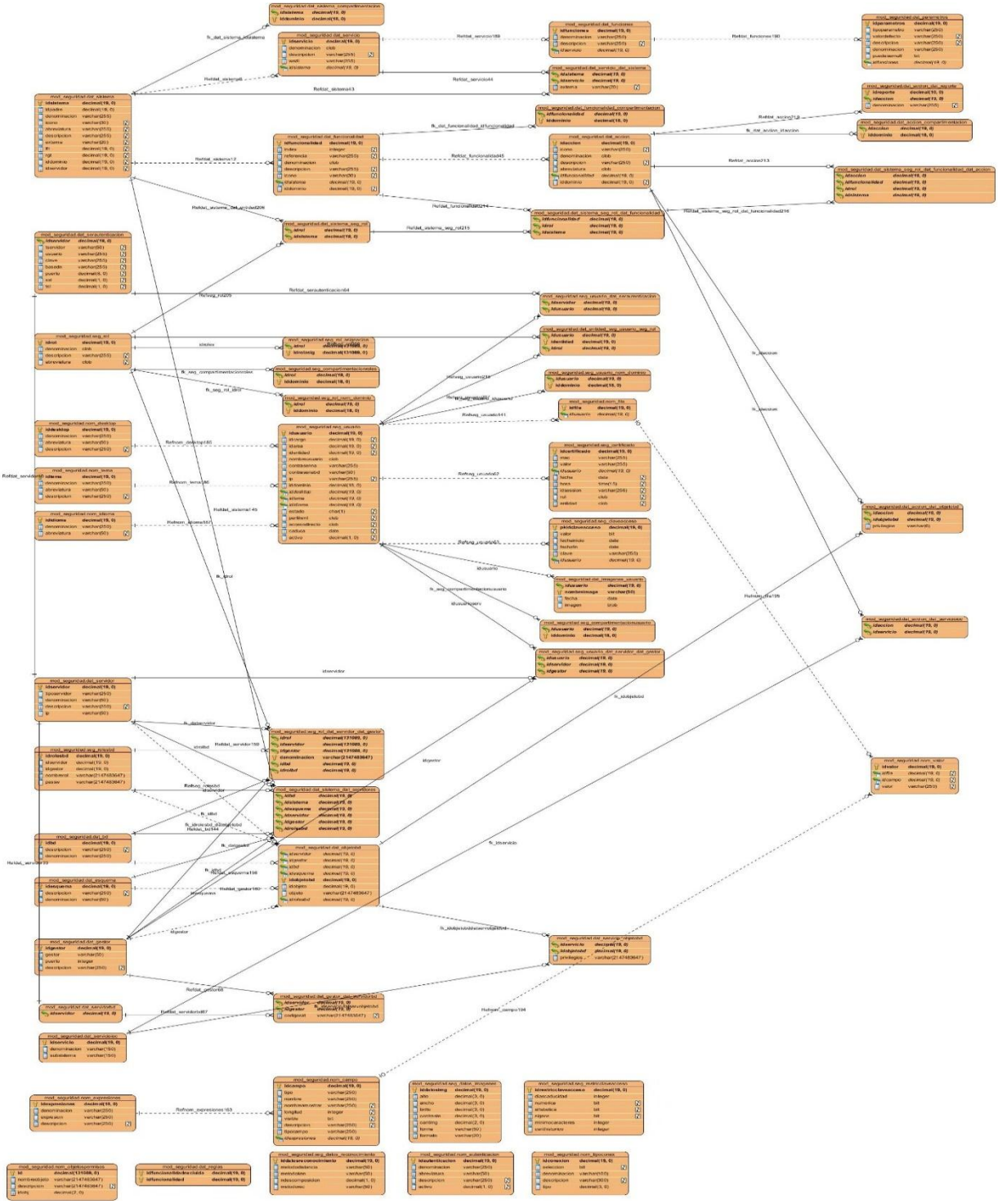
PARÉ, R. C., SANTILLÁN, L. A. C., COSTA, D. C., GINESTÀ, M. G., ESCOFET, C. M. & MORA, Ó. P. 2007. *Databases*.

RAPIDSVN. 2012. *RapidSVN* [Online]. Available: http://www.rapidsvn.org/.

RUÍZ, R. G. 2008. Diseño e implementación de un framework de persistencia. *PFC:J2EE[UOC]***,** 1-175.

SIRONI, G. 2010. *From Doctrine 1 to Doctrine 2 | Web Builder Zone* [Online]. Available: http://css.dzone.com/articles/doctrine-1-doctrine-2.
ZEND-TECHNOLOGIES-LTD. 2014. *About* [Online]. Available: http://framework.zend.com/about/.

Anexo 1: Diagrama entidad-relación del esquema seguridad de la base de datos del marco de trabajo Sauxe.



Anexo 2: Casos de prueba configurar usuario.

ID del escenario	Escenarios de la sección	Carga de trabajo	Descripción	Resultado esperado	Resultado obtenido
1: Cargar usuarios.	1.1-Cargar roles. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar usuario- Roles.	50	Esta interfaz muestra todos los roles existentes.	El sistema debe responder con una velocidad menor a: 162300 ms	162000 ms
	1.2-Cargar usuarios. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar usuario-Usuarios.	50	Esta interfaz muestra todos los usuarios existentes.	El sistema debe responder con una velocidad menor a: 162300 ms	162000 ms
2: Eliminar usuarios.	2.1-Eliminar roles. Para ello se selecciona el rol a eliminar y se presiona el botón eliminar.	50	Una vez realizada la operación el sistema avisa el resultado y recarga el grid.	El sistema debe responder con una velocidad menor a: 742000 ms	269400 ms
	2.2-Eliminar usuarios. Para ello se selecciona el usuario a eliminar y se presiona el botón eliminar.	50	Una vez realizada la operación el sistema avisa el resultado y recarga el grid.	El sistema debe responder con una velocidad menor a: 742000 ms	269400 ms
3: Adicionar usuarios.	3.1-Adicionar roles. Para ello se presiona el botón adicionar, se llenan los campos	50	Se carga una interfaz con los campos necesarios para realizar la adicción del rol.	El sistema debe responder con una velocidad menor a:	420000 ms

	correspondientes y se presiona el botón aceptar		Una vez terminada la operación se informa del resultado y se recarga el grid.	1191150 ms	
	3.2-Adicionar usuarios. Para ello se presiona el botón adicionar, se llenan los campos correspondientes y se presiona el botón aceptar	50	Se carga una interfaz con los campos necesarios para realizar la adicción del usuario. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 1191150 ms	420000 ms
4: Modificar usuarios.	4.1-Modificar roles. Para ello se selecciona el rol y se presiona el botón modificar, se carga una interfaz con los datos del rol seleccionado, se procede a modificar y luego se presiona el botón aceptar.	50	Se carga una interfaz con los campos necesarios para realizar la modificación del rol. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 633300 ms	350800 ms
	4.2-Modificar usuarios. Para ello se selecciona el usuario y se presiona el botón modificar, se carga una interfaz con los datos del usuario seleccionado, se procede a modificar y luego se presiona el botón aceptar.	50	Se carga una interfaz con los campos necesarios para realizar la modificación del usuario. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 633300 ms	350800 ms

Anexo 3: Caso de prueba configurar sistema.

ID del escenario	Escenarios de la sección	Carga de trabajo	Descripción	Resultado esperado	Resultado obtenido
1: cargar sistemas.	1.1-Cargar sistemas. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar sistemas-Sistemas.	50	Esta interfaz muestra todos los sistemas existentes.	El sistema debe responder con una velocidad menor a: 3026000 ms	780000 ms
	1.2-Cargar funcionalidades. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar sistemas-Funcionalidades.	50	Esta interfaz muestra todas las funcionalidades por cada sistema existente.	El sistema debe responder con una velocidad menor a: 3026000 ms	780000 ms
	1.3-Cargar acciones. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar sistemas-Acciones.	50	Esta interfaz muestra todas las acciones que se puedes realizar por cada una de las funcionalidades de los sistemas.	El sistema debe responder con una velocidad menor a: 3026000 ms	780000 ms
2: Eliminar sistemas.	2.1-Eliminar sistema. Para ello se selecciona el sistema a eliminar y se presiona el botón eliminar.	50	Una vez realizada la operación el sistema avisa el resultado y recarga el grid.	El sistema debe responder con una velocidad menor a: 1949600 ms	1042400 ms
	2.2-Eliminar funcionalidades. Para ello se selecciona el sistema y luego la funcionalidad que se desea	50	Una vez realizada la operación el sistema avisa el resultado y recarga el grid.	El sistema debe responder con una velocidad menor a: 1949600 ms	1042400 ms

	eliminar y se procede a presionar el botón eliminar.				
	2.3-Eliminar acciones. Para ello se selecciona el sistema, la funcionalidad y la acción que se desea eliminar y se procede a presionar el botón eliminar.	50	Una vez realizada la operación el sistema avisa el resultado y recarga el grid.	El sistema debe responder con una velocidad menor a: 1949600 ms	1042400 ms
3: Adicionar servidores.	3.1-Adicionar sistemas. Para ello se presiona el botón adicionar, se llenan los campos correspondientes y se presiona el botón aceptar	50	Se carga una interfaz con los campos necesarios para realizar la adicción del sistema. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 1016400 ms	750400 ms
	3.2-Adicionar funcionalidad. Para ello se selecciona el sistema al cual se le desea adicionar la funcionalidad y se procede a presionar el botón adicionar, se llenan los campos correspondientes y se presiona el botón aceptar.	50	Se carga una interfaz con los campos necesarios para realizar la adicción de la funcionalidad. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 1016400 ms	750400 ms
	3.3-Adicionar acciones. Para ello se selecciona el sistema y la funcionalidad a la cual se le desea adicionar la acción, y se procede a presionar el botón adicionar. Se llenan los campos correspondientes y	50	Se carga una interfaz con los campos necesarios para realizar la adicción de la acción. Una vez terminada	El sistema debe responder con una velocidad menor a: 1016400 ms	750400 ms

	se presiona el botón aceptar.		la operación se informa del resultado y se recarga el grid.		
4: Modificar sistemas.	4.1-Modificar sistemas. Para ello se selecciona el sistema y se presiona el botón modificar, se carga una interfaz con los datos del sistema seleccionado, se procede a modificar y luego se presiona el botón aceptar.	50	Se carga una interfaz con los campos necesarios para realizar la modificación del sistema. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 2211300 ms	1306000 ms
	4.2-Modificar funcionalidad. Para ello se selecciona el sistema y la funcionalidad se presiona el botón modificar, se carga una interfaz con los datos de la funcionalidad seleccionada, se procede a modificar y luego se presiona el botón aceptar.	50	Se carga una interfaz con los campos necesarios para realizar la modificación de la funcionalidad. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 2211300 ms	1306000 ms
	4.3-Modificar acciones. Para ello se selecciona el sistema, la funcionalidad y la acción y se presiona el botón modificar. Se carga una interfaz con los datos de la acción seleccionada. Se procede a modificar y luego se presiona el botón aceptar.	50	Se carga una interfaz con los campos necesarios para realizar la modificación de la acción. Una vez terminada la operación se informa del	El sistema debe responder con una velocidad menor a: 2211300 ms	1306000 ms

			resultado y se recarga el grid.		

Anexo 4: Caso de prueba configurar servidor.

ID del escenario	Escenarios de la sección	Carga de trabajo	Descripción	Resultado esperado	Resultado obtenido
1: cargar servidores.	1.1-Cargar servidores. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar servidores-Servidores.	50	Esta interfaz muestra todos los servidores existentes.	El sistema debe responder con una velocidad menor a: 551600 ms	299700 ms
	1.2-Cargar gestores de base de datos. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar servidores-Gestores de base de datos.	50	Esta interfaz muestra todos los gestores de base de datos existentes.	El sistema debe responder con una velocidad menor a: 551600 ms	299700 ms
	1.3-Cargar gestionar roles de base de datos. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar servidores-Gestionar roles de base de datos.	50	Esta interfaz muestra todos los servidores de base de datos con sus respectivos roles existentes.	El sistema debe responder con una velocidad menor a: 551600 ms	299700 ms
2: Eliminar servidores.	2.1-Eliminar servidores. Para ello se selecciona el objeto a eliminar y se presiona el botón eliminar.	50	Una vez realizada la operación el sistema avisa el resultado y recarga el grid.	El sistema debe responder con una velocidad menor a: 1301300 ms	644600 ms
	2.2-Eliminar gestores de base de datos.	50	Una vez realizada la operación el	El sistema debe responder con	644600 ms

	Para ello se selecciona el servidor al cual se le desea eliminar el gestor, y se procede a presionar el botón eliminar una vez seleccionado el gestor.		sistema avisa el resultado y recarga el grid.	una velocidad menor a: 1301300 ms	
	2.3-Eliminar roles de base de datos. Para ello se selecciona el servidor, el gestor y la base de datos, y se procede a seleccionar el rola a eliminar y presionar el botón eliminar.	50	Una vez realizada la operación el sistema avisa el resultado y recarga el grid.	El sistema debe responder con una velocidad menor a: 1301300 ms	644600 ms
3: Adicionar servidores.	3.1-Adicionar servidores. Para ello se presiona el botón adicionar, se llenan los campos correspondientes y se presiona el botón aceptar	50	Se carga una interfaz con los campos necesarios para realizar la adicción del servidor. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 1226400 ms	1143900 ms
	3.2-Adicionar gestores de base de datos. Para ello se selecciona el servidor al cual se le desea adicionar el gestor, y se procede a presionar el botón adicionar, se llenan los campos correspondientes y se presiona el botón aceptar.	50	Se carga una interfaz con los campos necesarios para realizar la adicción del gestor. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 1226400 ms	1143900 ms

	3.3-Adicionar roles de base de datos. Para ello se selecciona el servidor, el gestor y la base de datos, y se procede a presionar el botón adicionar, se llenan los campos correspondientes y se presiona el botón aceptar.	50	Se carga una interfaz con los campos necesarios para realizar la adicción del gestor. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 1226400 ms	1143900 ms
4: Modificar servidores.	4.1-Modificar servidores. Para ello se selecciona el servidor y se presiona el botón modificar, se carga una interfaz con los datos del servidor seleccionado, se procede a modificar y luego se presiona el botón aceptar.	50	Se carga una interfaz con los campos necesarios para realizar la modificación del servidor. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 417500 ms	279300 ms
	4.2-Modificar roles de base de datos. Para ello se selecciona el servidor, el gestor y la base de datos, y se selecciona el rol y se presiona el botón modificar se carga una interfaz con los datos del rol seleccionado, se procede a modificar y luego se presiona el botón aceptar.	50	Se carga una interfaz con los campos necesarios para realizar la modificación del gestor. Una vez terminada la operación se informa del resultado y se recarga el grid.	El sistema debe responder con una velocidad menor a: 417500 ms	279300 ms

Anexo 5: Caso de prueba configurar nomenclador.

ID del escenario	Escenarios de la sección	Carga de trabajo	Descripción	Resultado esperado	Resultado obtenido
1: Adicionar nomencladores.	1.1-Adicionar nomencladores Gestores de base de datos. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar nomencladores-Gestores de base de datos. Clic en el botón adicionar.	50	Carga una interfaz en la cual se deben llenar los campos correspondientes y presionar el botón insertar.	El sistema debe responder con una velocidad menor a: 1051200 ms	762600 ms
	1.2-Adicionar nomencladores Idiomas. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar nomencladores-Idiomas. Clic en el botón adicionar.	50	Carga una interfaz en la cual se deben llenar los campos correspondientes y presionar el botón insertar.	El sistema debe responder con una velocidad menor a: 1051200 ms	762600 ms
	1.3-Adicionar nomencladores Expresiones. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar nomencladores-Expresiones. Clic en el botón adicionar.	50	Carga una interfaz en la cual se deben llenar los campos correspondientes y presionar el botón insertar.	El sistema debe responder con una velocidad menor a: 1051200 ms	762600 ms
2: Cargar nomencladores.	2.1-Cargar nomencladores Gestores de base de datos.	50	Esta interfaz muestra todos los nomencladores Gestores de	El sistema debe responder con una	299100 ms

	Para ello presionar clic en la ruta Inicio-Seguridad-Configurar nomencladores-Gestores de base de datos.		base de datos existentes.	velocidad menor a: 413700 ms	
	2.2-Cargar nomencladores Idiomas. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar nomencladores-Idiomas.	50	Esta interfaz muestra todos los nomencladores Idiomas existentes.	El sistema debe responder con una velocidad menor a: 413700 ms	299100 ms
	2.3-Cargar nomencladores Expresiones. Para ello presionar clic en la ruta Inicio-Seguridad-Configurar nomencladores-Expresiones.	50	Esta interfaz muestra todos los nomencladores Expresiones existentes.	El sistema debe responder con una velocidad menor a: 413700 ms	299100 ms
3: Eliminar nomencladores.	3.1-Eliminar nomencladores Gestores de base de datos. Para ello selecciona el objeta a eliminar de los presente en el grid y se presiona el botón eliminar.	50	La interfaz informa que ha sido eliminado y recarga el grid.	El sistema debe responder con una velocidad menor a: 1301300 ms	644600 ms
	3.2-Eliminar nomencladores Idiomas. Para ello selecciona el objeta a eliminar de los presente en el grid y se presiona el botón eliminar.	50	La interfaz informa que ha sido eliminado y recarga el grid.	El sistema debe responder con una velocidad menor a: 1301300 ms	644600 ms

	3.3-Eliminar nomencladores Expresiones. Para ello selecciona el objeta a eliminar de los presente en el grid y se presiona el botón eliminar.	50	La interfaz informa que ha sido eliminado y recarga el grid.	El sistema debe responder con una velocidad menor a: 1301300 ms	644600 ms
4: Modificar nomencladores.	4.1-Modificar nomencladores Gestores de base de datos. Para ello selecciona el objeta a modificar de los presente en el grid.	50	Esta interfaz carga con los datos del objeto seleccionado del grid, se modifican y se presiona el botón aceptar.	El sistema debe responder con una velocidad menor a: 417500 ms	279300 ms
	4.2-Modificar nomencladores Idiomas. Para ello selecciona el objeta a modificar de los presente en el grid.	50	Esta interfaz carga con los datos del objeto seleccionado del grid, se modifican y se presiona el botón aceptar.	El sistema debe responder con una velocidad menor a: 417500 ms	279300 ms
	4.3-Modificar nomencladores Expresiones. Para ello selecciona el objeta a modificar de los presente en el grid.	50	Esta interfaz carga con los datos del objeto seleccionado del grid, se modifican y se presiona el botón aceptar.	El sistema debe responder con una velocidad menor a: 417500 ms	279300 ms