Universidad de las Ciencias Informáticas

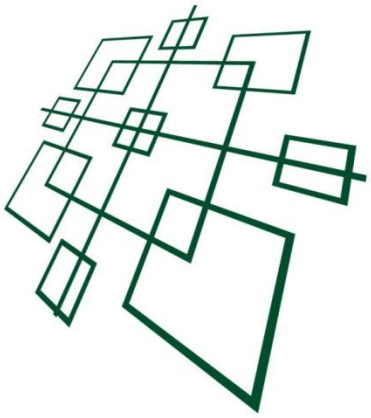
Dirección de Informatización

Akademos
Gestión Académica

ARQUITECTURA PARA AKADEMOS 2.0

Trabajo de diploma para optar por el título de:

Ingeniero en Ciencias Informáticas



Autor: Andry Suárez Hernández

Tutores: Ing. Darien Cepero Rojas

Lic. Rosmel Álvarez Monzón

Junio - Año 50 de la Revolución

"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte."

Che

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Dirección de Informatización de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente del mismo.

Para que así conste firmo la presente a los ___ días del mes de junio del año 2008.

Andry Suárez Hernández

Ing. Darien Cepero Rojas

Lic. Rosmel Álvarez Monzón

A:

mi Mamá, mi Hermana,

mi Papá y mi Familia.

Andry

Agradezco a:

Mi mamá, porque este sueño también es suyo, por todo el sacrificio, dedicación e infinito amor.

Adry, mi hermanita, mi eterno amor. Recuerda (very optimist).

Lys, mi novia, por regalarme su Abril, no sería lo mismo sin ti.

Mi abuela Lilia, mi tía Lilo, mis tíos Berna y Ricardo, mis primas Yany y Lisdy… a toda mi familia.

Mis tutores, Pepe y Rosmel, por su ayuda, confianza y amistad.

Emil por su ejemplo, seriedad y profesionalismo.

Clary y Ernesto, a quienes les debo una parte esencial de mi formación profesional.

Lili por su paciencia y gran ayuda ante el diluvio de mi español.

Roydel, no te defraudo mi herma, vuelvo ingeniero.

Mis otros dos hermanos, Liso y Juanma, por todo lo vivido y lo que aún falta.

Rebeca por su cariño, entrega y amor.

Camejo, (my Friend), por su dedicación y seguridad.

Joel por su amistad y apoyo en toda mi carrera.

Erneston, mi socio, por haberme enseñado F10 y F11.

Los miembros del equipo de desarrollo de Akademos: Grisel, Norges, el Loco (Dianly), Roberto (ent_Robert), Orlando (Prusu), Frank, Lago, Molina (el Moli), Yanirys (la Tata), Yisel (Amorcito), Catherine, Inalvys, Alexey (el Titi), Damián, Olivia (Olili), Yuneisy, los Netys (Luis Ernesto y Javier) y los Prettys (el Ferna y el Jose).

Todo aquel que de una forma u otra hizo posible la realización este anhelo.

Resumen

Este trabajo presenta una caracterización del Sistema Automatizado para la Gestión Académica, Akademos, desde el punto de vista arquitectónico. Su objetivo principal es definir una arquitectura para el desarrollo en plataforma libre de una nueva versión del sistema, teniendo en cuenta estilos y patrones arquitectónicos. La nueva solución está regida por el aumento de los requerimientos funcionales y no funcionales, al incorporar un número significativo de nuevas prestaciones, y la necesidad de que la misma se integre como un producto y se ajuste a las políticas de Software Libre emprendidas por nuestro país.

Esta investigación se sustenta en el análisis de algunas de las variantes tecnológicas más representativas en plataforma libre para el desarrollo de sistemas Web, como: Java Enterprise Edition (JEE), la plataforma Mono y el framework Symfony para PHP. En función de las características con las que debe cumplir el sistema y las condiciones del equipo de desarrollo, se determina, en dependencia de las ventajas de cada una de estas variantes, la más eficiente para su explotación.

Una vez definida la propuesta tecnológica, se realiza un estudio de su estructura y los posibles patrones a aplicar sobre la misma, con particular énfasis en el modelo del negocio y el acceso a datos, para guiar el desarrollo de la nueva solución del sistema a través de buenas prácticas.

Este estudio es de suma importancia ya que el mismo constituye una alternativa viable de orden arquitectónico para un nuevo desarrollo del sistema Akademos, contribuyendo a una mejor estructuración modular y lógica de su funcionamiento.

Índice

Índice de figuras

Introducción

La Universidad de las Ciencias Informáticas (UCI) consta en la actualidad con más de diez mil estudiantes y aproximadamente más de mil profesores, así como directivos y personal de secretaría, sujetos activos que intercambian constantemente en una de las principales actividades de la universidad: el control del proceso docente. Con el objetivo de agilizar este proceso y brindar un mayor soporte a la labor del personal de secretaría, surge el Sistema Automatizado para la Gestión Académica (Akademos).

Akademos ha sido, por más de cuatro años, el sistema que ha dado respuesta ante los cambios y necesidades de los actores de la gestión académica. Ha brindado, además, un conjunto de servicios de manera activa a la comunidad universitaria en general. Actualmente el sistema, debido a nuevas condiciones y concepciones de desarrollo, sufre una serie de cambios estructurales, que se extienden desde la base hasta los niveles superiores. Debido a ello es necesaria una redefinición total del mismo con un marcado interés en su arquitectura. La investigación, el estudio y la definición de una alternativa para una nueva arquitectura constituyen la razón fundamental de este trabajo.

Tema

Arquitectura para Akademos 2.0

Situación problémica

Akademos, en su actual versión brinda una serie de importantes servicios a la comunidad universitaria, que implican a estudiantes, profesores, personal de secretaría y directivos de la universidad. Los requerimientos tanto funcionales como no funcionales del sistema se han incrementado en amplitud. Incorporan no sólo el control de los docentes en sus cursos de pregrado, sino el tratamiento de trabajos de diplomas y cursos de postgrado, entre otros. Otro elemento relevante es que en estos momentos el sistema no constituye un producto, aspecto que impide su correspondiente comercialización. Por estas razones, la Dirección de Informatización de la UCI ha decidido realizar una nueva versión del sistema, sobre la base de un producto propio que corrija las problemáticas representadas y, a la vez, se ajuste a las políticas de

desarrollo en Software Libre llevadas a cabo por nuestro país. La arquitectura del sistema no se adapta a las nuevas necesidades y requerimientos y esto imposibilita el desarrollo del nuevo producto sobre la base arquitectónica vigente, lo que hace necesario su redefinición.

Problema científico

¿Cuál será la nueva arquitectura del Sistema Automatizado para la Gestión Académica de la Universidad de las Ciencias Informáticas – “Akademos” con vista a su nuevo desarrollo en plataforma libre?

Objeto de estudio

Sistema Automatizado para la Gestión Académica de la Universidad de las Ciencias Informáticas – “Akademos”.

Campo de acción

Arquitectura del Sistema Automatizado para la Gestión Académica de la Universidad de las Ciencias Informáticas – “Akademos”.

Objetivo general

Definir la arquitectura con vista al desarrollo en plataforma libre del Sistema Automatizado para la Gestión Académica de la Universidad de las Ciencias Informáticas – “Akademos”, teniendo en cuenta estilos y patrones arquitectónicos aplicables a la nueva solución, así como los riesgos de diseño e implementación de la misma.

Objetivos específicos

- Analizar y estudiar la actual arquitectura del sistema Akademos.
- Analizar y comparar distintas tecnologías y tendencias arquitectónicas en plataforma libre sobre aplicaciones Web.
- Definir una nueva arquitectura para el sistema Akademos en función de los nuevos requerimientos.

Pregunta científica

¿Cuál es la arquitectura en plataforma libre a utilizar en el desarrollo del Sistema Automatizado para la Gestión Académica de la Universidad de las Ciencias Informáticas – “Akademos”?

Tareas a cumplir

- Realizar un estudio de la arquitectura propuesta por Microsoft para aplicaciones Web.
- Analizar cómo se aplican los elementos propuestos por Microsoft en la actual arquitectura de Akademos.
- Estudiar las principales tendencias del desarrollo de sistemas Web en plataforma libre.
- Establecer una comparación entre las principales tendencias del desarrollo de sistemas Web en plataforma libre con el fin de seleccionar una o un híbrido entre ellas.
- Definir la nueva arquitectura del sistema Akademos basado en el estudio anterior.
- Arrojar conclusiones de la investigación.
- Elaborar un documento que recoja el resultado de la investigación.

Métodos científicos

Para el desarrollo de este trabajo se aplicaron métodos tanto teóricos como empíricos para garantizar la autenticidad del resultado de la investigación.

Métodos teóricos utilizados:

- Históricos
- Lógicos
- Sistémicos

Método empírico utilizado:

- Experimental

Capítulo I: Marco conceptual. Análisis de la actual arquitectura de Akademos 1.0

1.1 Introducción

En este capítulo se abordan definiciones claves dentro de la Arquitectura de Software, como los patrones y los estilos arquitectónicos y las bases conceptuales de la gestión académica. Estas definiciones son utilizadas como punto de partida para el análisis posterior de la arquitectura propuesta por Microsoft y su aplicación en la actual solución de Akademos.

1.2 La gestión académica

La gestión académica en un centro de estudios es el proceso mediante el cual se controlan, organizan y dirigen todas las entidades y actores que intervienen en la labor docente del mismo y que engloba, además, otros subprocesos como la definición de los planes de estudio, el control de la matrícula administrativa, el expediente académico y personal de los estudiantes, así como el control de la asistencia y las evaluaciones de los mismos. En esta tarea participan directivos, personal de secretaría y profesores.

1.3 Akademos

Akademos es un sistema informático cuya principal misión es el control de todos los procesos que intervienen en la gestión académica de un centro de estudios universitarios. Surge como respuesta a la necesidad de sustentar y dar soporte en la Universidad de las Ciencias Informáticas (UCI) a toda la labor del personal de secretaría, con la visión de obtener un producto genérico, capaz de ser aplicable y adaptable en cualquier centro que implemente el control docente universitario. Fue diseñado y desarrollado por un equipo de estudiantes del Plan CUJAE y la UCI, rectorados por la Dirección de Informatización de la UCI y bajo la tutoría del ingeniero Emil Lima Valdés. Los primeros módulos liberados fueron: Matrícula y Plan de Estudio, a pocos meses de haber comenzado su desarrollo y, posteriormente, se fueron incorporando los demás módulos que actualmente conforman el sistema: Expediente, Profesor, Estudiante, Control Docente y Reportes.

1.4 Módulos y principales características de Akademos

Akademos fue desarrollado tomando en cuenta una serie de principios básicos que determinan su estructura y funcionamiento:

- El dinamismo del proceso de gestión académica constituye la principal fuente de riesgos para un sistema que intente automatizarlo.
- Un sistema que automatice la gestión académica debe lograr que todos los involucrados (directivos, personal de secretaría, profesores y estudiantes) tengan un papel activo en el proceso.
- El plan de estudio es la entidad fundamental del proceso de gestión académica y rige todos sus subprocesos (matrícula, control, planificación, etc.).

Estos principios están sustentados en los siete módulos que componen el sistema y que interactúan coordinadamente entre sí para dar cumplimento a las tareas del negocio, estos módulos se describen a continuación:

Plan de Estudio

Constituye la entidad fundamental a partir de la cual se definen los diferentes perfiles y disciplinas por las cuales transitará el estudiante, así como las asignaturas, evaluaciones y sus formas de calificación asociadas.

Matrícula

Es el encargado del control de los estudiantes. Permite no sólo la gestión de la matrícula administrativa, sino también la de los movimientos a que son sometidos los estudiantes en su paso por la universidad, y la definición de los estados y las transiciones entre estos. Logra la descentralización del proceso de matrícula, aspecto fundamental en centros como la UCI que reciben cada año a miles de educandos.

Expediente

Gestiona toda la información que constituye el grueso de la historia de la universidad. Constituye un repositorio virtual de los documentos asociados a un estudiante, los cuales pueden estar basados en plantillas, ser generados por el sistema o tener libre formato. Los documentos de libre formato pueden ser adjuntados al expediente por parte de los usuarios, en este caso, personal de secretaría. Cada vez que se crea un documento, este es almacenado junto a otros datos como la fecha de creación y el autor, para facilitar el trabajo con el mismo y su posterior localización.

Profesor

El objetivo de este módulo es permitir la asignación de los profesores a las diferentes estructuras administrativas. Es controlado por el personal de secretaría, que asocia profesores y estructuras a través de diferentes niveles de acceso, de manera que estos puedan mantener actualizado el registro académico de sus estudiantes. Además, permite establecer un balance de la carga docente de los profesores y un mejor control de estos por facultad y grupo.

Estudiante

Este módulo está diseñado específicamente para que el estudiante pueda consultar sus evaluaciones en las distintas asignaturas vencidas, así como su desempeño en las disímiles materias del período docente en curso. Permite el acceso a su expediente y con esto a documentos como la Hoja de Matrícula, la Hoja de Prematrícula, entre otros. Por otra parte, brinda la posibilidad de consultar la trayectoria docente del resto de los estudiantes del centro de estudios.

Reportes

La generación de reportes es quizás la tarea fundamental de todo sistema de gestión de datos. Este módulo provee a los usuarios de un conjunto de herramientas, que permiten el diseño y publicación de reportes. Con esta funcionalidad el usuario puede diseñar reportes de todos los datos de los estudiantes, desde las notas hasta la proveniencia social, convirtiéndose en una poderosa herramienta para el estudio de la comunidad estudiantil.

Control Docente

Su funcionalidad básica es controlar el registro docente de los estudiantes. Uno de los principios en que se basó el diseño de Akademos persigue que la información sea introducida al sistema por los mismos que la originan, previendo de esta manera la ocurrencia de errores. Son los profesores los que aplican las diferentes formas de control del avance de los estudiantes, de ahí que Akademos les permita directamente introducir los resultados de todas las evaluaciones que apliquen, así como mantener actualizado el registro de asistencia de todos los estudiantes en las distintas asignaturas.

1.5 Seguridad de Akademos

Akademos tiene tres niveles de seguridad:

Seguridad de los datos

Para el almacenamiento de los datos se utiliza SQLServer 2000, una herramienta capaz de garantizar la seguridad tanto lógica como de integridad de los datos almacenados en la Base de Datos.

Seguridad de los recursos lógicos

El sistema operativo, Windows en este caso, tiene la responsabilidad de restringir el acceso a cada uno de los recursos lógicos de la aplicación como: páginas Web, Servicios Web, archivos XML, entre otros.

Seguridad de los elementos específicos del negocio

Akademos implementa varios niveles de acceso que restringen las acciones que puede realizar un usuario, las cuales, además, pueden ser monitoreadas en tiempo real utilizando el “Monitor de incidencias”, que es una aplicación de escritorio implementada para Akademos en la plataforma dotNET.

1.6 Arquitectura de software

En la medida en que las aplicaciones o sistemas informáticos fueron evolucionando, creciendo en funcionalidades y ganando en robustez, se hizo necesaria una mayor organización en el proceso de desarrollo. En consecuencia, se crearon nuevas metodologías y nació una nueva rama dentro de la Ingeniería de Software (IS): la Arquitectura de Software (AS).

A fines de la década de los 60’ Edsger Dijkstra, destacado científico y Doctor en Ciencias de la Computación, enuncia un número de reglas sobre la estructura que todo sistema informático debe tener, de las cuales parten los principios de la organización básica de los sistemas de software, sus ideas fueron tomadas por P. I. Sharp. En una conferencia donde se realiza un análisis sobre las mismas, Sharp expresó: “*Pienso que tenemos algo, aparte de la ingeniería de software: algo de lo que hemos hablado muy poco pero que deberíamos poner sobre el tapete y concentrar la atención en ello. Es la cuestión de la arquitectura de software”* (Reynoso, 2004).

Más adelante, en los 70’, David Parnas aborda el tema de la descomposición de los sistemas informáticos en módulos, y menciona los principios básicos de ocultación que más tarde abrirían el camino al encapsulamiento. Desarrolla también tópicos vinculados a las estructuras de software y las familias de programas, con énfasis en la calidad, evolución y mantenimiento del software, basando la técnica de modulación en decisiones de diseño.

Hacia 1992, Dewayne Perry y Alexander Wolf, publican un estudio en el cual se tratan enteramente cuestiones de AS. En este proyecto, gestado desde 1989, se establece una analogía entre la AS y la arquitectura de edificaciones y se abordan ideas sobre las propiedades

y relaciones entre los componentes de un sistema de software, a saber: elementos, forma y razón.

Roy Fielding, en el año 2000, enuncia en su tesis: "*Architectural styles and the design of network-based software architectures*", el modelo REST, donde expone los principios de la Arquitectura Orientada a Servicios (SOA). Comienza así una nueva etapa dentro de la AS y abre un camino a las nuevas tendencias que tienen su epicentro en el Instituto de Ingeniería de Software (SEI) de la Universidad de Carnegie & Mellon, en California, Estados Unidos.

Teniendo en cuenta estos elementos, es posible definir la AS como la rama dentro de la IS que engloba la organización fundamental de un sistema, reflejada en sus componentes, las relaciones de cada uno de esos componentes con el resto y con el entorno. La AS instituye los principios que guían el diseño y evolución de un sistema informático; presenta niveles más altos de abstracción y conocimiento, y se construye previo al proceso de diseño y estructuración del código.

Una perspectiva más formal, propuesta por el estándar IEEE 1471-2000, plantea que: "*La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución*" (Reynoso, 2004).

1.6.1 Estilos arquitectónicos

Cuando se trabaja con sistemas de software, particularmente en su parte conceptual, se parte de ciertos modelos con el objetivo de determinar su estructura. Estos modelos se reconocen como Estilos Arquitectónicos (EA). Los EA son caracterizaciones tipológicas que adquieren un significado distintivo. Surgen a la par de la AS y en analogía a los estilos arquitectónicos de edificaciones. Su mayor utilidad radica en la reutilización que ofrecen, al definir una solución general ante similares demandas de los clientes. La correcta implementación de un EA conlleva a una mejor estructuración del sistema. Entre los EA más recurridos se encuentran:

- Tuberías y Filtros.
- Arquitectura de Pizarra.
- Modelo Vista-Controlador.
- Arquitectura en Capas.
- Arquitectura basada en Componentes.
- Arquitectura basada en Objetos.
- Arquitectura orientada a Servicios.

Algunos de estos estilos son abordados más adelante debido a sus relaciones con la estrategia arquitectónica de Microsoft y su implementación en Akademos, así como con la nueva arquitectura.

1.6.2 Patrones arquitectónicos

Los Patrones Arquitectónicos (PA) constituyen soluciones reutilizables a problemas específicos en un marco dado. Se implementan a través de un conocimiento específico acumulado por la experiencia en un dominio. Surgen como una manera organizativa de brindar una respuesta unificada ante situaciones problemáticas similares; sin embargo, su empleo no será nunca igual, pues cada desarrollador interpreta y ejecuta cada patrón según sus requerimientos.

Los PA proporcionan una arquitectura uniforme que permite una fácil expansión, mantenimiento y modificación de la aplicación.

Un patrón se describe esencialmente de la siguiente manera:

- Descripción del problema: hace referencia al fenómeno que promueve la necesidad de su aplicación.
- Consideraciones: aspectos a considerar dentro de la solución propuesta.

- Solución general: constituye una descripción básica de la solución en sí.
- Consecuencias: pros y contras al utilizar la solución propuesta.
- Patrones relacionados: patrones con similar funcionalidad que deben ser estimados como alternativa.

Existen diferentes tipos de patrones en dependencia del nivel conceptual donde se apliquen, ejemplo de ellos son los patrones de diseño, de implementación y de arquitectura.

Dependiendo del propósito funcional de los patrones, estos se distinguen en los siguientes tipos:

- Fundamental: construye bloques de otros patrones.
- Presentación: estandariza la visualización de los datos.
- Creación: concepción condicional de objetos.
- Integración: comunicación con sistemas y recursos externos.
- Particionamiento: organización y separación de lógica compleja, conceptos y actores en múltiples clases.
- Estructurales: separación de la presentación, las estructuras de datos, la lógica de negocio y procesamiento de eventos en bloques funcionales.
- Comportamiento: coordina y organiza el estado de los objetos.
- Concurrencia: maneja el acceso concurrente a los distintos recursos.

1.7 Sistemas Web

Los sistemas informáticos desarrollados para su uso en la Web emergen desde los propios inicios de Internet. Las bondades que brinda el entorno Web en cuanto a la comunicación y trasferencia de datos, han permitido la descentralización del accionar de los usuarios en un marco altamente distribuido y, por tanto, altamente comercial; esto, unido a una tecnología que

facilita la distribución de nuevas versiones, han convertido a estos sistemas en una opción viable en el manejo y transacción de información a gran escala ante un número elevado de clientes.

1.7.1 Modelo Cliente-Servidor

La utilización y aplicación del modelo Cliente-Servidor está fuertemente ligada al desarrollo de sistemas Web. Su principal característica es que un servidor puede gestionar las peticiones de varios clientes concurrentemente (ver figura 1.1), los clientes accederían a través de protocolos determinados con anterioridad y, por ende, con una codificación y sincronización de las transacciones previstas. Las peticiones de los clientes pueden ser a una Base de Datos, a una serie de recursos compartidos, a la información de archivos o a los archivos en sí.

Figura 1.1: Modelo Cliente-Servidor

1.8 Frameworks y ambientes de desarrollo

Un framework denota un conjunto de objetos que definen un diseño abstracto para solucionar un conjunto de problemas relacionados. Puede incluir programas, bibliotecas de objetos o lenguaje interpretado, por lo que su uso facilita la elaboración de sistemas informáticos. En este sentido, los desarrolladores se centran más en la abstracción de alto nivel que en los elementos de bajo nivel, factor que repercute directamente en el incremento de la productividad y la eficiencia en un proyecto determinado.

Algunos ejemplos de frameworks son: el framework dotNET, en sus versiones 1.1, 2.0 y 3.0 para aplicaciones dotNET; el framework Symfony para sistemas Web en PHP: Kumbia, CakePHP, Codeigniter y el framework Zend, también para el trabajo con PHP; el framework Hibernate, para gestionar el acceso a datos en Java y Nhibernate para gestionar el acceso a datos en dotNET, entre otros.

Los Ambientes o Entornos de Desarrollo de sistemas informáticos, por su parte, se encuentran determinados principalmente, por elementos de hardware, su distribución y disposición en el área de desarrollo. Por ejemplo: estaciones de trabajo, servidores y dispositivos externos que se utilizan en la implementación del producto; se incluyen, además, software necesarios para la creación del sistema, el control de versiones, entornos de prueba requeridos por el servidor, entre otros. A partir de estos criterios, los Ambientes o Entornos de Desarrollo constituyen un conjunto de hardware y software en constante interacción entre ellos y con los desarrolladores del sistema.

1.9 Estrategia arquitectónica de Microsoft y su aplicación en Akademos

Akademos es el sistema, que actualmente soporta y ofrece numerosos servicios vinculados a los procesos docentes y educativos de la UCI. Dada la infraestructura tecnológica de nuestra sede universitaria, resultó necesario conformar una aplicación que permitiera gestionar de manera automatizada los distintos procedimientos que engloba la tramitación académica de un centro de estudios superiores.

Uno de los principios rectores que determinó esta solución fue la descentralización de la mayoría de los procesos, con miras a garantizar la rapidez y agilidad en la gestión de los mismos. Por tal motivo, fue seleccionado el entorno Web, que permite el manejo tanto de concurrencias como de una actualización del sistema sin que ello afecte en gran medida a los usuarios, consecuentemente, se adoptó el modelo Cliente-Servidor.

La tecnología empleada para su construcción fue ASP.NET, conjunto de librerías y clases para el manejo de páginas Web, con la herramienta de desarrollo Visual Studio.NET (VS) framework 1.1, y como Sistema Gestor de Base de Datos (SGBD) el SQLServer 2000; una tecnología

completamente propietaria, pero muy eficiente a la hora del desarrollo de sistemas WEB potentes en corto tiempo.

La estrategia concebida para la arquitectura base del sistema fue la propuesta por Microsoft para aplicaciones dotNET. Se debe tener presente que la arquitectura se abstrae en sus vistas de la plataforma de desarrollo y, aunque el Entorno Integrado de Desarrollo (IDE) Visual Studio es propietario, algunos de los principios de la arquitectura propuesta por Microsoft pueden ser utilizados en la realización del nuevo sistema.

Microsoft propone, para sistemas empresariales en entorno Web, un estilo de Llamada-Retorno denominado Arquitectura en Capas basada en componentes (ver figura 1.2). Este tipo de estilo permite el crecimiento escalable del sistema; su correcta concepción posibilita la reutilización de los componentes, así como un bajo acoplamiento tanto vertical como horizontal de la aplicación. Cada capa implementa su lógica particular y expone sus funcionalidades a las capas superiores abstrayéndose de la lógica de cada uno de los componentes que la conforman. Se establecen, por lo general, tres capas fundamentales, aunque en dependencia de lo que se desee realizar el número de capas puede incrementarse potencialmente. Las tres que tipifican al Modelo de Akademos son: Acceso a Datos, Negocio e Interfaz, descritas a continuación.

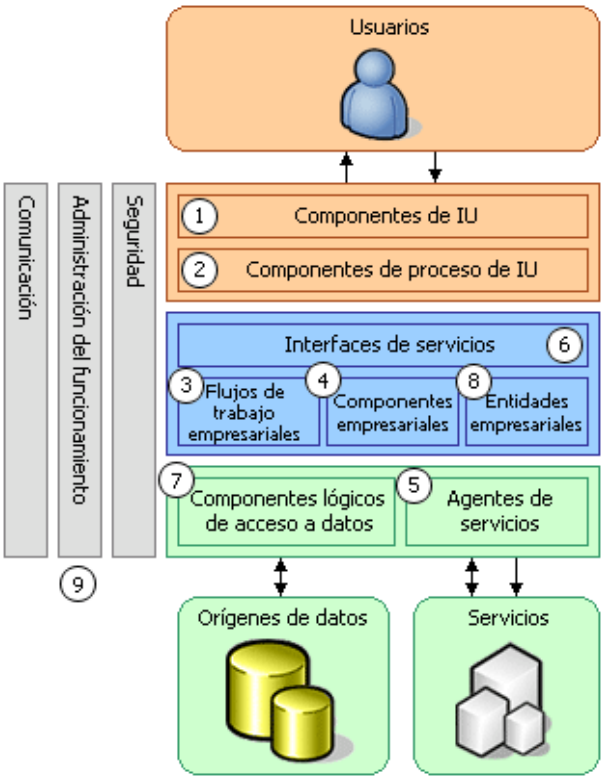


Figura 1.2: Arquitectura en Tres Capas según Microsoft

La capa de acceso a datos del sistema cuenta con un patrón de arquitectura para acceso a datos llamado *Table Data Gateway* (TblDG). Su función cardinal es actuar como un *Gateway* para cada tabla en la Base de Datos. Un *Gateway* se define como un objeto que encapsula el acceso a un sistema o a un recurso externo, en este caso a tablas de una Base de Datos. Esta estructuración del TblDG permite separar el código de acceso a datos del resto de la aplicación, lo cual asegura la adaptación del sistema ante posibles modificaciones.

Un factor medular en este proceso son los tipos de datos devueltos y sus funcionalidades durante su empleo por las capas superiores; de ahí que se recomienda un previo análisis sobre ello y la unificación de los tipos de estructura de datos devueltas, puesto que al utilizar el TblDG se puede no sólo devolver consultas simples como filas, sino también, tablas completas.

Akademos se apoya en un *Gateway* base, CEntidad, que incluye todas las operaciones básicas de acceso a datos, tanto de inclusión como de extracción, y varias sobrecargas de estas

operaciones. Esta clase es heredada por cada uno de los *Gateway* específicos de cada módulo para las tablas de la Base de Datos, y le adiciona a cada uno, además de su lógica específica, los métodos base de CEntidad, conformando entonces toda una estructura de un TblDG.

Los objetos para el trabajo con los datos son tomados del propio framework dotNET y difieren según la lógica aplicada en la capa de negocio; mas desde el acceso a datos de Akademos se devuelven sólo *DataSets*, a causa de las amplias facilidades de trabajo con los mismos en capas superiores y a la unificación de los tipos devueltos. Entre las ventajas que los *DataSets* ofrecen se pueden citar: la conversión a otros tipos de objetos de estructura de datos, y el mantenimiento en memoria de varias tablas relacionadas, entre otras. Estos objetos se enmarcan dentro de librerías del framework dotNET, como es el caso de (*System.Data*).

Otra de las propuestas de Microsoft es el uso intensivo de los Procedimientos Almacenados (*Store Procedures*), que aumenta las prestaciones del sistema y la utilización de clases controladoras de los procesos de acceso a datos en su comunicación a través de los objetos de ADO.NET. El *SQLHelper* es un ejemplo de una de estas estructuras controladoras, pero la estrategia concebida por el proyecto orientó el desarrollo de un framework de acceso a datos propio para permitir la independencia de los gestores de datos. Sin embargo, esta independencia no se alcanzó porque el acceso a la Base de Datos está acoplado a los objetos conectores que brinda ADO.NET para SQLServer 2000: (*SqlConnection*, *SqlCommand*,...); por lo tanto, si se cambiara de Gestor de Base de Datos habría que reescribir el código utilizado por la clase base de acceso a datos, *Gateway* CEntidad. Para mantener la abstracción entre la aplicación y el gestor de Base de Datos, fue necesario, entonces, trabajar con las interfaces que provee ADO y que implementan los Paquetes de *Driver* específicos para cada gestor. Este hecho frena el aprovechamiento de las facilidades que brindan los procedimientos almacenados de SQLServer 2000 y obstaculiza la independencia entre el gestor de Base de Datos y la aplicación.

La capa de negocios gestiona los procesos de negocio. Su correcta estructuración es fundamental para la confección del sistema, pues se encarga de llevar a cabo toda una lógica que podría ser demasiado compleja para la interfaz, que conllevaría a la sobrecarga de la misma.

De incluirla en el acceso a datos, limitaría los procedimientos a la hora de un cambio sustancial en la aplicación. Su puesta en práctica difiere además en dependencia de los requerimientos funcionales del sistema.

Existen tres patrones bases para la implementación de la capa de dominio o capa de negocio en aplicaciones empresariales, estos son: *Transaction Script*, *Domain Model* y *Table Model*. A continuación, se hace referencia al patrón de capa de dominio *Table Model* o Modelo de Tablas, presente en la actual capa de negocios del sistema.

El patrón *Table Model* provee un objeto para el manejo del comportamiento en una tabla de la Base de Datos, permitiendo el empaquetamiento de datos y comportamiento de la lógica de dominio. Para ello, utiliza algunas estructuras de datos, así como operaciones sobre estas. Microsoft ofrece una amplia colección de objetos para la administración de datos que permiten hacer más interactivo el trabajo con este patrón, estos objetos son los que brinda la librería del framework dotNET (*System.Data*). Un ejemplo de ellos son los *DataSets*, altamente funcionales.

Las distintas estructuras, ya sean instancias de objetos o colecciones de métodos estáticos, del patrón *Table Model*, se ejecutan en la solución primaria de Akademos, además de vincularse de manera muy efectiva con el patrón de acceso a datos TblDG. Actualmente no todas las tablas de la Base de Datos tienen asociada una clase *Table Model*; esto viene dado porque no todas las tablas participan en procesos de negocios, aunque sí se definen correctamente las fronteras de la capa de negocio y los niveles hasta los cuales extienden sus funcionalidades a las capas superiores.

La capa de Interfaz de Usuario (IU), por su parte, posee dos características fundamentales: la de actuar como presentadora de datos -toma los datos desde una fuente y son mostrados al usuario-, y como actualizadora de datos -los datos son cambiados por algún evento generado por el usuario y estos son enviados al servidor garantizando su actualización-.

Para optimizar el flujo de datos y dar una mayor y mejor utilidad a los eventos generados en esta parte del sistema, la aplicación de patrones de IU es transcendental. En Akademos se aplica el

patrón conocido como Modelo-Vista-Controlador (MVC), cuyo objetivo es mantener actualizada la IU una vez que el Modelo cambie, sin crear dependencias entre el Modelo y la Vista, delimitando para ello las fronteras entre los elementos de este patrón: el Modelo, la Vista y el Controlador. (Ver figura 1.3)

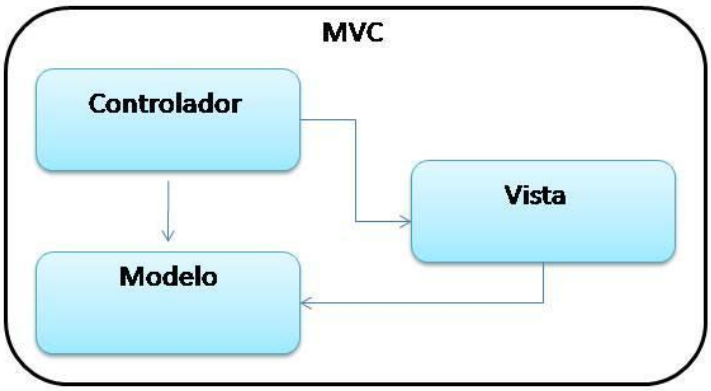


Figura 1.3: Representación clásica de MVC

El MVC separa el modelamiento del dominio de la presentación y el accionar del usuario. El Modelo es el encargado de responder ante las peticiones de información efectuadas principalmente por la interfaz, así como las peticiones del Controlador. Por su parte, la Vista maneja la información mostrada en la Interfaz; mientras que el Controlador interpreta las necesidades del usuario, ya sean acciones del mouse o entradas por teclado u otros dispositivos, mediante la información al Modelo o a través del cambio de la Vista de forma apropiada.

ASP.NET ofrece amplias potencialidades para el trabajo en la capa de presentación y en el uso del MVC, establece páginas o archivos *.aspx* que incluyen todo el código de la Vista, principalmente HTML; lo que no quiere decir que no se pueda agregar código de los demás elementos, porque al hacerlo se estarían cometiendo errores de modelamiento del patrón. ASP.NET trae consigo también la aplicación del *Code-Behind Refactoring*, funcionalidad que brinda un mecanismo para separar la Vista del Modelo y del Controlador, además de permitir que los métodos referenciados en la página se puedan implementar en clases diferentes. Al utilizar *Code-Behind Refactoring* de dotNET, es posible emplear *IntelliSense* de dotNET para mostrar la lista de los métodos de los objetos con que se trabaja en la página. Esta funcionalidad, sin

embargo, no se incluye en los *.aspx*, por lo que se utiliza otro archivo, el *.aspx.cs*, donde se manejan parte de los eventos que la página ofrece y se separa completamente la lógica de la Vista del resto de los elementos del MVC.

En Akademos la correcta implementación del MVC es evidente, pues se combinan todos estos elementos en una particularidad del MVC, que no se tiene en cuenta con la aplicación del *Code-Behind Refactoring* solamente, y es el *MVC Refactoring*. Debido a que dotNET brinda amplias potencialidades en este aspecto, la correcta implementación del *Code-Behind Refactoring* puede lograr la separación de la Vista del Modelo y el Controlador, pero no garantiza la separación y el bajo acoplamiento entre el Controlador y el Modelo. Esto se logra con el accionar de los desarrolladores del sistema al tener bien definidos el flujo de datos y de eventos entre un elemento u otro, específicamente en los datos provenientes del Modelo en algún *RecordSet*, y son interpretados por los eventos del Controlador al pasárselos a la Vista para que esta los muestre.

Los elementos descritos y analizados anteriormente, ofrecen de una manera tanto general como específica, una panorámica del actual sistema Akademos, así como los patrones aplicados en cada una de las capas del estilo con que se implementa, haciendo énfasis en cada uno de estos patrones.

1.10 Conclusiones

En este capítulo se conceptualizan, describen y analizan elementos relativos a la arquitectura de software, los estilos y patrones arquitectónicos, aspectos a tener en cuenta en el transcurso de este trabajo.

A partir de los criterios expuestos, es posible afirmar que la actual base del sistema Akademos no se ajusta a las políticas trazadas por el país, vinculadas a la incorporación creciente de los sistemas a las plataformas de Software Libre. Asimismo, en la primera versión del sistema automatizado de gestión académica de la UCI, no existe la necesaria independencia del gestor de Base de Datos.

Los patrones aplicados a esta solución en su capa de Interfaz, Negocio y Acceso a Datos respectivamente, pueden servir como base para una nueva versión del sistema en plataforma libre.

Capítulo II: Análisis de las tecnologías

2.1 Introducción

Seguidamente se abordan algunos de los frameworks y herramientas más usadas en Software Libre para el desarrollo de sistemas Web. Se hace referencia al avance obtenido por la plataforma dotNet dado su impacto sobre el proyecto Mono, pero enfocado a temas más vinculados a la estructura y funcionalidades del software empresarial y su arquitectura.

2.2 Arquitectura sobre software empresarial

Las arquitecturas para sistemas de software que gestionan transacciones y procesos empresariales, describen de manera organizada la estructura del software, así como su interacción en el entorno donde este es implantado. Su principal objetivo es dotar al sistema de un número de características fundamentales que definan su fiabilidad, escalabilidad y reusabilidad, así como un bajo acoplamiento y una alta cohesión entre los demás subsistemas que conforman la solución general, en un entorno distribuido dentro del marco empresarial donde se encuentra.

La UCI presenta un entorno altamente colaborativo entre sus sistemas de software, donde la información es manejada en las áreas de trabajo correspondientes, por tanto, la interacción entre los distintos sistemas que presenta la universidad es relativamente alta dado el flujo de información entre cada una de estas áreas. La UCI, comparable a una gran empresa, requiere de la integración de todos estos sistemas informáticos, en un marco altamente distribuido.

2.3 Análisis de las tecnologías para el desarrollo de aplicaciones empresariales sobre la Web

Con vista a definir una propuesta arquitectónica determinada por un nuevo desarrollo del sistema Akademos, se realiza un análisis de algunas de las alternativas tecnológicas más representativas para ello, sin divagar en la concepción que el sistema sea basado sobre la Web, dadas las características específicas de la universidad y las bondades ofrecidas por este entorno. De ahí que permita, de una manera más flexible, que la información sea introducida por quienes la generen, principio básico de la antigua solución y que se retoma en esta nueva definición de la

arquitectura; evitando con ello el exceso de trabajo del personal de secretaría e influir positivamente en el flujo de información entre las distintas actividades que conforman los procesos que automatiza el sistema. De manera que se logra un aumento de la eficiencia en estos al eliminar pérdidas de tiempo y de recursos.

2.3.1 Mono

Mono es un proyecto de código abierto inicializado por Ximian y actualmente impulsado por Novell. Su objetivo es crear un grupo de herramientas libres basadas en GNU\Linux y compatibles con dotNet. Mono provee el software necesario para desarrollar y ejecutar aplicaciones Cliente-Servidor en sistemas operativos como: Linux, Unix, Solaris y Windows. Es compatible, además, con varios lenguajes de programación como: C#, Java, Boo, Temerle, VisualBasic dotNET y Python. (Cepero, 2007).

Algunos de los componentes más importantes de Mono son:

- Máquina virtual de CLI, que contiene un cargador de clases, un compilador en tiempo de ejecución (JIT) y subrutinas de recolección de memoria.
- Biblioteca de clases que da soporte ante los lenguajes del CLR.
- Compilador para lenguajes como: C#, MonoBasic (versión Mono de VisualBasic dotNET), Java, entre otros.

2.3.1.1 MonoDevelop

MonoDevelop constituye un IDE libre y gratuito, diseñado en un principio para C# aunque soporta otros lenguajes contenidos en el framework dotNET como: Boo, Nermerle y Java.NET. Este IDE incluye ayuda, manejo de clases complejas, completamiento de código, ambiente de diseñado para la IU y un depurador integrado.

La arquitectura de MonoDevelop se basa en tres capas: una Capa Principal (*Core Layer*) que provee los servicios para las aplicaciones, una Capa de Proyectos (*Project Layer*) que

implementa el modelo de objetos del proyecto y una Capa Superior que constituye en general el propio IDE. (Ver figura 2.1)

Figura 2.1: Representación de MonoDevelop

2.3.1.2 Microsoft (ASP.NET)

Desde diciembre de 2002, después que Microsoft publica los documentos que especifican la arquitectura dotNET, los desarrolladores de Mono han seguido bien de cerca la evolución de dicha plataforma y llevado muchas de sus funcionalidades a Software Libre. En este epígrafe se realiza un análisis de algunas de las nuevas características concentradas por Microsoft en su plataforma dotNET y que Mono podría incorporar en un futuro a su infraestructura.

La plataforma de desarrollo dotNET, de Microsoft, es una estructura concebida para el desarrollo de aplicaciones robustas. Está diseñada especialmente para lograr la integración entre componentes programados en ella sin tener en cuenta el mismo lenguaje de programación base, pues es capaz de llevar estos lenguajes a un código intermedio MSIL y de aquí a código nativo o de máquina. En esta transacción se incorpora al CLR que permite la eficiencia en todo este proceso de compilación. Se logra con ello una transparencia entre los desarrolladores y los componentes de los sistemas de software.

Otra característica importante que brinda esta plataforma es la utilización de los XML WebServices (Servicios Web) como *middleware* entre aplicaciones o componentes que estén altamente distribuidos, o que han sido elaborados en plataformas diferentes y con sistema operativo distinto a Windows, pues se basan en el formato estándar XML.

Visual Studio.NET constituye el software base de desarrollo en la plataforma dotNET y está condicionado, tanto visual como estructuralmente, de manera que al desarrollador le sea fácil el trabajo con el mismo, lo cual permite el ahorro en tiempo y costo en la elaboración de un sistema determinado. Dentro de este entorno de trabajo se encuentra ASP.NET que constituye un marco sobre el cual se elaboran aplicaciones basadas en la Web. En ASP.NET se trabaja sobre la base del modelo Cliente-Servidor, donde un usuario hace una petición al servidor de una página determinada y este es quien se la brinda. En este sistema de colaboración al cliente sólo viaja código estándar HTML, pues el código que gestiona funcionalidades más fuertes del sistema como transacciones a Bases de Datos u operaciones específicas del negocio, está completamente del lado del servidor, y es ejecutado e interpretado por el mismo. Toda esta dinámica del trabajo con ASP.NET hace que sea más viable el manejo de patrones como: MVC, el *Front Controller* y el *Page Controller*, estos últimos toman los principios del MVC y le agregan, además, mejoras cualitativas a los procesos que gestionan.

Un aspecto importante en el trabajo con ASP.NET es la interacción y dependencia de sus componentes con los distintos frameworks de dotNET, ya que estos últimos definen el desarrollo alcanzado por la plataforma en sus versiones 1.1, 2.0 y 3.0.

Por otra parte es innegable el salto cualitativo alcanzado por dotNET desde el trabajo con ASP básico hasta la explotación del framework 1.1 con ASP.NET. A partir del framework 1.1 de dotNET se incluyen toda una serie de nuevas funcionalidades como: el *Code-Behind*, la estructuración de los Controles de Usuario, el trabajo con los *DataGrids* y los Controles de Validación.

En los repositorios de datos (*DataSources*), por su parte, el trabajo se gestiona a través de un conjunto de objetos incluidos en la librería (*System.Data*) de ADO.NET, ejemplo de ellos son los *DataSets, DataTables y DataRows*, para el trabajo con los datos recopilados del *DataSource*, así

como los de ejecución de consultas hacia el mismo, tal es el caso del *SQLQuery* y el *SQLComand*.

Hacia los framework 2.0 y 3.0 se puede observar el surgimiento de nuevas formas y estrategias de trabajo que traen consigo nuevos componentes y con ello más funcionalidades. Este es el caso de los controles de acceso a datos como los *GridView* o *DetailsView*, que permiten un *DataBinding* declarativo y bidireccional. Otro aspecto significativo son las dependencias de Caché de SQL, funcionalidad totalmente nueva que permite que la Caché no sólo dependa del tiempo en que se ejecute alguna acción, sino que puede depender, además, del contenido de la Base de Datos, de manera que cuando cambie el contenido de una cierta tabla se actualice el contenido de una Caché independiente del tiempo.

Entre las novedades de la interfaz se encuentra el trabajo con las Páginas Maestras o *Master Pages*, en inglés. Estas páginas presentan los mismos controles y formas de operación sobre ella que una página *.aspx* normal, pero con la particularidad de constituir una plantilla estructural de código y diseño. De esta plantilla heredan un conjunto determinado de páginas que comparten funcionalidades básicas, pero que pueden ser enriquecidas en dependencia de lo que se necesite en particular, lo que se logra a través de los *ASP ContentPlaceHolder*.

Una particularidad que se retoma en estas últimas versiones de los frameworks de la plataforma dotNET es lo referente a la gestión de la seguridad. En la versión 1.1 del framework dotNET se hacía particularmente difícil abordar la seguridad de una manera ágil, por lo que muchos desarrolladores dejaron de utilizar las facilidades que vienen intrínsecas en ASP.NET 1.1. En las versiones posteriores a este framework este tema es analizado nuevamente, donde se ofrecen mejores opciones ante el trabajo con la misma y se definen servicios y clases de membrecías para la administración de usuarios, roles y credenciales.

Esta administración se encuentra basada en proveedores que presentan un almacenamiento flexible, lo que posibilita que las funcionalidades se abstraigan del origen de datos. Presenta, además, controles de *Login* para la creación de nuevos usuarios y recuperación de claves o passwords, así como servicios con los que es posible la combinación de la autenticación mediante formularios y las basadas en roles.

Al analizar las características básicas con las que cuenta la plataforma dotNET, se puede expresar entonces que Microsoft brinda un entorno de trabajo muy eficiente para el desarrollo de aplicaciones Web, dotando al desarrollador de un IDE de alta calidad, con librerías de clases u objetos interrelacionados que facilitan el trabajo en este entorno. Lo anterior se evidencia en la estructura interna de los distintos framework que incorpora este IDE: framework 1.1 para VS 2003, framework 2.0 para VS 2005 y framework 1.1, 2.0 y 3.0 para VS ORCAS 2008.

Un aspecto importante a tener en cuenta cuando se habla de dotNET, es el hecho de que esta plataforma es privativa, por lo que para la utilización legal del software es necesario adquirir su licencia, es aquí donde surge un problema mayor, pues esta plataforma es propiedad de Microsoft, empresa estadounidense que está bajo las restricciones impuestas por el bloqueo a Cuba, razón por la cual no se permite el comercio o venta de sus productos hacia nuestro país.

2.3.2 Java Enterprise Edition (JEE)

2.3.2.1 Java

Java es el lenguaje de programación base de la plataforma *Java Enterprise Edition* (JEE). Surge en la década de los 90 en los laboratorios de la empresa Sun Microsystems. Es un lenguaje compilado y utiliza para ello la Máquina Virtual de Java -*Java Virtual Machine* (JVM)-, esta herramienta lleva el código Java a código intermedio o *bytecode* y de aquí a código nativo en dependencia del entorno en que este es ejecutado. Este proceso hace posible que el lenguaje Java pueda ser ejecutado en varios sistemas operativos que soporten su JVM y por tanto las aplicaciones desarrolladas bajo este lenguaje, haciendo de él un leguaje multipropósito. Java reúne, asimismo, todas las características de un lenguaje orientado a objetos, siendo a la vez sencillo, robusto, de arquitectura neutra, multihilo y portable. (Somarriba, 2007)

2.3.2.2 JEE

La plataforma JEE ha sido diseñada para el desarrollo de aplicaciones distribuidas basadas en componentes, los cuales se interrelacionan entre sí, ofreciendo excelentes perspectivas para el desarrollo de sistemas empresariales en Software Libre. Cada componente en JEE debe formar

parte de la aplicación y por tanto deben ser desplegados en un contenedor en la parte del servidor, este hecho le ofrece al componente ciertos servicios de bajo nivel, ya sea desde el punto de vista de seguridad o de manejo de concurrencia, transacciones y persistencia. Por tanto, sería erróneo pensar que JEE es un producto concreto distribuido por Sun Microsystems como es el caso de su SDK. (Somarriba, 2007)

El modelo de desarrollo bajo JEE está encaminado a la creación de aplicaciones basadas en N capas. La lógica de la aplicación se divide en componentes con diferentes funciones que componen una aplicación JEE y que están distribuidos en dependencia de la capa a la que pertenece. Aunque puede variar, típicamente una aplicación suele tener cinco capas diferentes:

- Capa cliente: representa la interfaz de usuario que maneja el cliente.
- Capa de presentación: representa el conjunto de componentes que generan la información que se mostrará en la interfaz de usuario del cliente.
- Capa de lógica de negocio: contiene los componentes de negocio reutilizables.
- Capa de integración: contiene los componentes que permiten hacer más transparente el acceso a la capa de sistemas de información. Este es el lugar idóneo para implementar la lógica de objetos de acceso a datos (DAO, *Data Access Object).* (Somarriba, 2007)
- Capa de recursos: engloba los sistemas en los cuales la información se almacena físicamente como Bases de Datos relacionales, sistemas *legacy*, Bases de Datos orientadas a objetos, bancos de ficheros de datos, etc. (Somarriba, 2007)

Toda esta organización del sistema en capas y la creación de componentes reutilizables hacen de este modelo de desarrollo una alternativa eficiente, tanto desde el punto de vista de construcción del sistema como de la arquitectura del mismo, lo que posibilita un bajo acoplamiento entre sus partes, así como una fuerte reusabilidad y escalabilidad.

Seguidamente se realiza el análisis de algunas de las herramientas y frameworks más usados para el desarrollo de aplicaciones en esta plataforma.

2.3.2.3 Eclipse (IDE)

Eclipse constituye un potente IDE para el desarrollo de aplicaciones robustas. Cuando se descarga Eclipse se obtiene un conjunto de instrumentos para el desarrollo en Java, *Java Development ToolKit* (JDT), que permite escribir y depurar programas en Java, además se obtiene un ambiente de desarrollo de *plug-ins*, *Plug-in Development Environment*, para heredar de Eclipse. (Pimentel, 2007)

Comenzó como un proyecto de IBM. Es un IDE de código abierto y actualmente lo desarrolla la Fundación Eclipse, organización sin ánimo de lucro que fomenta el Software Libre. Este IDE constituye una aplicación de "Cliente Enriquecido" opuesto de las aplicaciones de "Cliente Liviano" basadas en navegadores.

El SDK de Eclipse incluye las herramientas de desarrollo de Java, que ofrece un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. Eclipse también hace uso de un espacio de trabajo, en este caso un grupo de *metadata* en un espacio para archivos plano, lo que permite modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente.

2.3.2.4 NetBeans

NetBeans es un proyecto de código abierto fundado por Sun Microsystems especialmente diseñado para el desarrollo de aplicaciones en JAVA, pero que acepta otros lenguajes de programación. Consta de una gran base de usuarios y una comunidad en constante crecimiento, lo que le ha permitido, al igual que muchos otros sistemas libres, el progreso paulatino de sus prestaciones y la eliminación de *Bugs* que pudiesen existir.

La plataforma NetBeans permite que las aplicaciones se desarrollen a partir de un conjunto de módulos o componentes de software. Un módulo contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (*manifest file*) que lo identifica como módulo.

Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas por otros desarrolladores de software.

Este IDE a pesar de ser una excelente herramienta para el trabajo con aplicaciones JAVA, no presenta las ventajas que ofrece Eclipse en cuanto al consumo de memoria, facilidades de trabajo y rendimiento.

2.3.2.5 Framework Hibernate

El framework Hibernate constituye sin dudas una de las mejores opciones al trabajar en el entorno JEE, en cuanto al tema de acceso a datos o en el trabajo con Base de Datos Relacionales. Esta estructura establece un motor de persistencia encargado de servir como traductor entre objetos y registros, lo que permite la manipulación de datos a través de la Programación Orientada a Objetos (OOP), lo que lo convierte en un motor de persistencia objeto-relacional. (Ver figura 2.2)

Hibernate presenta un entorno altamente configurativo a través de XMLs y ficheros o archivos de propiedades, *HIBERNATE.PROPERTIES* e *HIBERNATE.CFG.XML*.

Un aspecto importante a tener en cuenta es el Dialecto de Hibernate, parámetro que indica el nombre de la clase que se comunica con la Base de Datos, donde al cambiar su configuración es posible cambiar de Gestor de Base de Datos así como de servidor o de Base de Datos en sí, sin necesidad de hacer grandes cambios en la aplicación. (Somarriba, 2007)

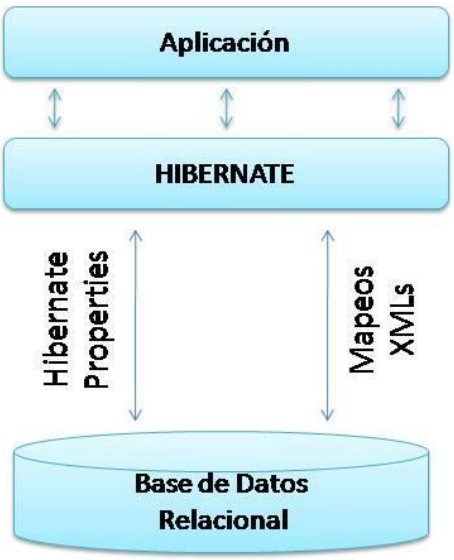


Figura 2.2: Representación del framework Hibernate

Entre las características fundamentales de Hibernate se encuentran:

- Ocultamiento de objetos: la sesión a nivel de transacción oculta los objetos persistentes.
- No actualización de objetos no modificados: no realiza viajes innecesarios al servidor de Base de Datos, manipulando sólo la colección de datos si realmente cambió.
- Implementación eficiente de la búsqueda a nivel de entidades.

2.3.2.6 Framework Struts

El framework Struts constituye una fuerte herramienta para el desarrollo de aplicaciones Web bajo la concepción del MVC para JEE. El trabajo con Struts garantiza el ahorro de tiempo de desarrollo y ayuda a la construcción y mantenimiento de aplicaciones que no sólo cumplan las especificaciones de los usuarios, sino que funcionen efectivamente sobre varios sistemas operativos.

Acorde con lo mencionado anteriormente, Struts garantiza el desarrollo de aplicaciones sobre el patrón MVC, más específicamente vinculado a la metodología del Modelo 2 propuesto por Sun Microsystems y que constituye una variación del patrón antes mencionado, razón por la cual no

pocos insisten en que este framework se acerca más a la implementación del patrón Controlador Frontal (*Front Controller*), que tiene su base en MVC, pero la documentación de Struts basa sus fundamentos en que este framework es implementado sobre MVC. En la figura 2.3 se muestra una representación de MVC según el framework Struts.

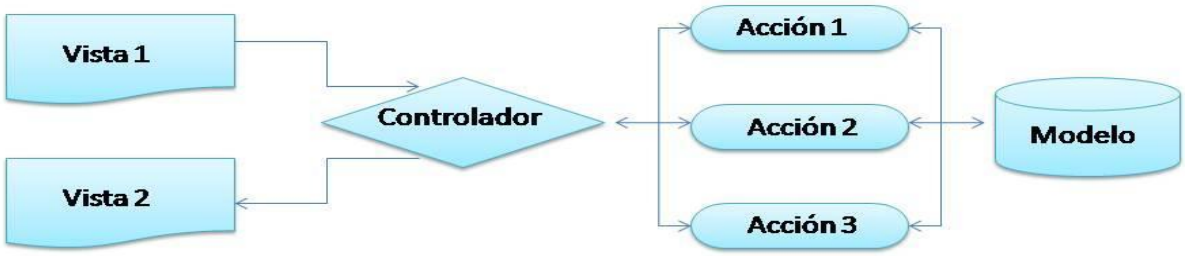


Figura 2.3: Representación de MVC según el framework Struts

Como se puede observar en la figura anterior, Struts utiliza sólo un controlador mediante el cual logra el mantenimiento de toda la lógica en un mismo lugar (*ActionServlet*). Este controlador es altamente configurable, pues evalúa las peticiones del usuario mediante un archivo de configuración (*struts-config.xml*), acción que se garantiza a través de los *tags* XML de este archivo.

Existen en la actualidad varias versiones de este framework liberadas, la 1.0 con sus subversiones y la 2.0. Se pueden observar varios avances significativos desde sus versiones anteriores hasta la 2.0, pero siempre condicionados por tres conceptos claves: construir, desplegar y mantener las soluciones desarrolladas, lo cual garantiza que ambas versiones coexistan sin que la segunda haga obsoleta a la primera.

2.3.2.7 Framework Spring

Otro de los frameworks que le dan vida a la plataforma JEE es Spring. Spring es un framework de código abierto basado en el paradigma de la Programación Orientada a Aspectos y desarrollado para la confección de aplicaciones sobre Java. Constituye un conjunto de clases y sub-frameworks que se interrelacionan haciendo posible un conjunto de funcionalidades altamente beneficiosas para los desarrolladores. Surge a través de la visión de Rod Johnson, experto en diseño y arquitectura sobre JEE, de crear un framework ligero y que reuniese un conjunto

específico de requisitos que le fueran útiles a la comunidad de desarrolladores. En el año 2002, Johnson publica en su libro "*Expert One-on-One Java EE Design and Development*" algunos aspectos sobre Spring y hacia el 2004 se libera la versión 1.0 de este framework.

Spring no obliga al desarrollador a usar un modelo de programación particular, mas puede ser considerado como sustituto del modelo *Enterprise JavaBeans*. Ofrece además mucha libertad a los desarrolladores de Java y soluciones bien documentadas y fáciles de usar en prácticas comunes dentro de la industria. Esto está dado por la facilidad que brinda de crear componentes reutilizables así como su adaptabilidad e integración con otros frameworks como: Hibernate, Struts, Ibatis, entre otros, funcionalidad que es configurable a través de XMLs específicos. (Somarriba, 2007)

Se pueden observar, en la arquitectura del framework (figura 2.4), los sub-frameworks que intervienen en su funcionamiento y como estos se interrelacionan entre sí.

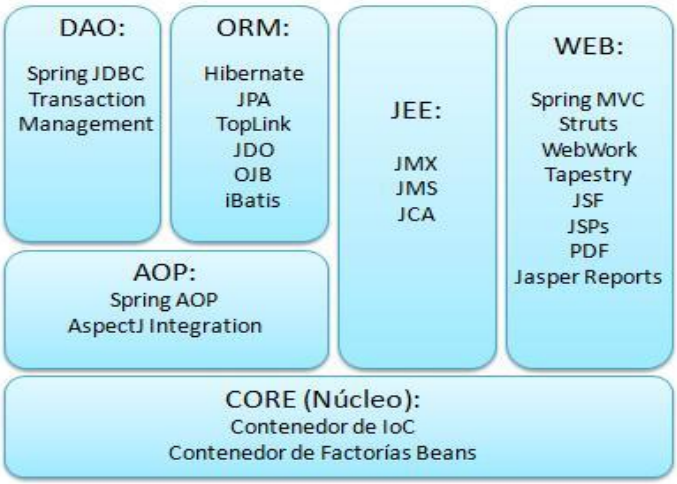


Figura 2.4: Representación de framework Spring

Algunos de los elementos más representativos de Spring son:

- Contenedor de inversión de control: la configuración de los componentes de la aplicación y el ciclo de vida es manipulado básicamente por objetos Java comunes.
- Programación orientada a aspectos: el trabajo con las funcionalidades, las cuales no pueden ser implementadas mediante la programación orientada a objetos sin hacer sacrificios de rendimiento y estabilidad.
- Framework de acceso a datos: trabaja con sistemas de administración objeto-relacionales que proveen soluciones para los desafíos técnicos reusables en multitudes de ambientes basados en Java.
- Manipulación de transacciones: armonización de varios APIs manipuladores de transacciones orquestados por objetos Java.
- Modelo-Vista-Controlador: framework basado en HTTP y *Servlets* que provee extensiones y personalizaciones.

Un aspecto interesante dentro de esta arquitectura es el sub-framework: Contenedor de Inversión de Control (IoC), también conocido como Inyección de Dependencias. Este sub-framework básicamente lo que hace es llamar al código del desarrollador y no al contrario, inyectando funcionalidades dentro de las clases y por ende de los objetos creados en dependencia de las necesidades del proyecto, utilizando para ello métodos Java. Este proceso conlleva a que las dependencias entre objetos interrelacionados sean mínimas, lo cual trae consigo que Spring pueda ser removido sin invocar grandes cambios en el código de la aplicación.

En el Núcleo Contenedor de Spring se encuentran las funcionalidades básicas de este framework. Presenta una factoría *Beans* que constituye el centro de toda aplicación basada en Spring, la cual es implementada bajo el patrón *Factory* y que aplica técnicas de IoC para separar configuración de la aplicación y las dependencias del código de la aplicación.

2.3.3 PHP

PHP, acrónimo recursivo de *PHP Hypertext Pre-Processor*, constituye un lenguaje *script* de alto nivel interpretado del lado del servidor. Este lenguaje es utilizado para la creación de páginas Web dinámicas y embebidas en páginas HTMLs.

Definido por Rasmus Lerdorf a finales de 1995 como una extensión de Perl. PHP se difunde rápidamente en la comunidad de desarrolladores y en 1997 Zeev Surasky y Andi Gutmans liberan una versión más potente al incorporar funcionalidades de C y Java, además de incluirle más características propias.

Al ser ejecutado del lado del servidor, PHP permite acceder a los recursos internos del mismo o a otros externos, como por ejemplo a una Base de Datos, siendo el resultado normalmente una página HTML con los datos y acciones enviadas al cliente. (Ver figura 2.5)

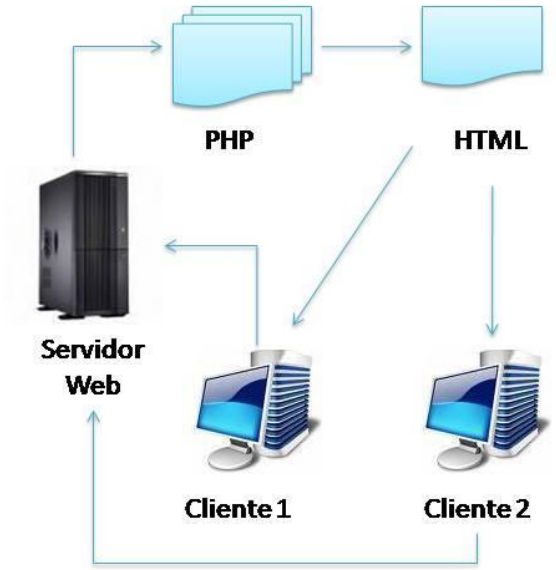


Figura 2.5: Modelo Cliente-Servidor sobre PHP

Algunas de las características básicas del lenguaje son:

- Soporte para una gran cantidad de bases de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, Sybase mSQL, Informix, entre otras.

- Integración con varias bibliotecas externas que permite, entre otras funcionalidades, generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- Ofrece una solución simple y universal para las paginaciones dinámicas la Web de fácil programación.
- Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.
- Gran comunidad de desarrolladores, como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, que garantiza que los fallos de funcionamiento se encuentren y reparen rápidamente. (Inda, 2007)
- El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.
- Con PHP se puede hacer cualquier cosa que se realice con un *script* CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de *cookies* y páginas dinámicas. (Inda, 2007)
- Permite técnicas de OOP.

Para aumentar las prestaciones y las facilidades de trabajo con este lenguaje se han fabricado un conjunto de frameworks como: Zend, Kumbia, NAJAX, Cake PHP y Symfony, los cuales constituyen ejemplos de algunos de los más usados, cada uno con sus características independientes que los hacen ser elegibles en dependencia de los requerimientos del sistema a implementar y las necesidades de los desarrolladores. Dadas las consultas realizadas a desarrolladores y expertos sobre el tema, las investigaciones efectuadas en internet y el análisis de otros trabajos, se aborda en lo adelante uno de estos frameworks en particular: Symfony.

2.3.3.1 Framework Symfony

Symfony fue desarrollado utilizando PHP 5 sobre una estructura completamente libre y licenciado bajo MIT, licencia libre del Instituto de Tecnología de Massachusetts. Está basado en nueve años de experiencia de Sensio, empresa de tecnologías de código abierto con amplia experiencia en el desarrollo Web, consultorías y auditorías. Inspirado en Ruby on Rails (framework para Ruby), Symfony se integra con proyectos ya conocidos como: Mojavi, Propel, Prado, entre otros, permitiendo un desarrollo rápido, minimizando los costos de mantenimiento y proponiendo una solución escalable. (Ver figura 2.6)

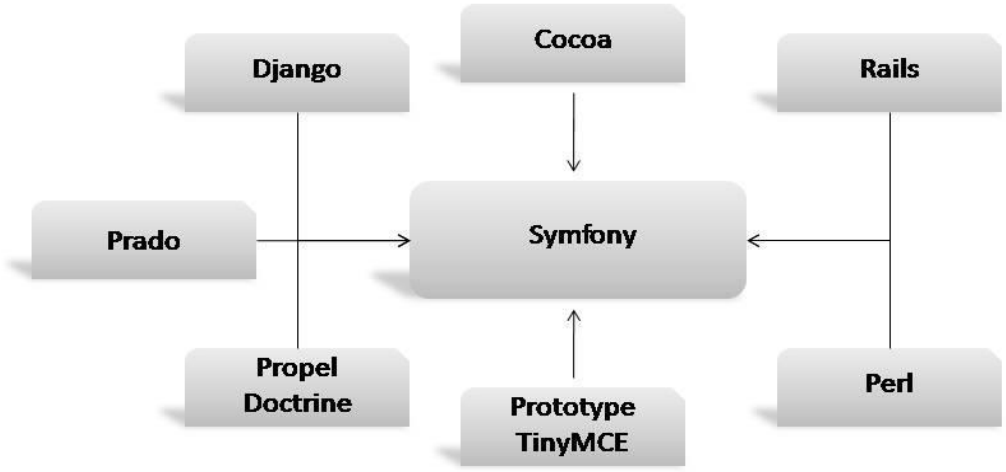


Figura 2.6: Framework Symfony

Symfony es entonces un framework completo, diseñado para el desarrollo de aplicaciones WEB dinámicas. Una de sus características más importantes es que separa la lógica de presentación de la de negocio y estas dos primeras de la de acceso a datos. Proporciona, además, un conjunto de herramientas y clases que automatizan tareas comunes, que permiten al desarrollador centrarse por completo en los aspectos específicos de su aplicación.

Algunas de sus características más destacadas son:

- Funciona con PHP 5: lenguaje orientado a objetos que garantiza un máximo de rendimiento.

- Estable: su código base es verificado por más de 4000 testeos funcionales y de unidad.
- Licenciado bajo MIT: licencia libre con la que se puede desarrollar aplicaciones libres sin restricciones significativas.
- Disponibilidad de una excelente documentación y una amplia comunidad de soporte: aseguran la calidad y la actualización sistemática de las fuentes y los recursos.
- Facilidad de programación: garantiza la aplicación de patrones y buenas prácticas de programación, lo que posibilita un desarrollo ágil al centrar al desarrollador en la lógica de la aplicación.

2.3.3.2 Eclipse (PDT)

Acorde con lo analizado anteriormente, Eclipse constituye un potentísimo IDE para el desarrollo de grandes sistemas, donde la inclusión de *plug-ins* aumenta considerablemente sus prestaciones. Uno de estos *plug-ins* es el *PHP Development Tools* (PDT) o Herramientas de Desarrollo para PHP, por su nombre en español.

Este proyecto brinda todos los componentes necesarios para el desarrollo de aplicaciones PHP dinámicas desde Eclipse. Sus principios se basan en ser intuitivo y fácil de aprender. Se rige por los estándares de Eclipse, siendo a la vez extensible y soportado continuamente por un amplio grupo de desarrolladores PHP.

2.4 Sistemas para la gestión de Base de Datos

Los sistemas que gestionan datos así como las transacciones entre estos, constituyen una potente herramienta, la cual está estrechamente vinculada a sistemas de software robustos que necesiten de la persistencia, localización y explotación de un volumen amplio de datos. El hecho de contar con un repositorio dinámico de valores hace posible que las prestaciones del software aumenten. Esto permite hacer estudios y valorar los resultados obtenidos en investigaciones propias o de otras investigaciones con lo que se almacene en el contenedor o Base de Datos. Estos sistemas sirven de interfaz entre la Base de Datos, el usuario y las aplicaciones que la

utilizan. Se compone de un lenguaje de manipulación de datos y de un lenguaje de consulta que combinados permiten las funciones anteriormente expuestas, entre otras.

Un Sistema de Gestión de Bases de Datos tiene los siguientes objetivos específicos:

- Independencia de los datos y los programas de aplicación.
- Minimización de la redundancia.
- Integración y sincronización de las bases de datos.
- Integridad de los datos.
- Seguridad y protección de los datos.
- Facilidad de manipulación de la información.
- Control centralizado.

Con el propósito de exponer una solución libre a la nueva arquitectura desde el punto de vista tecnológico, se han escogido dos de los gestores de datos que mayores prestaciones presentan, estos son: MySQL y PostgreSQL. A continuación se analizan algunas de sus características más significativas.

2.4.1 MySQL

MySQL es un sistema de gestión de Bases de Datos relacional, licenciado bajo GPL de GNU. Fue creado por la empresa sueca: MySQL AB, que mantiene el *copyright* del código fuente del servidor SQL, así como también de la marca.

Este gestor de Bases de Datos es probablemente el gestor más usado en el mundo del Software Libre, debido a su gran rapidez y facilidad de uso. Su gran aceptación está dada, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Entre sus principales características se pueden encontrar:

- Aprovechamiento de la potencia de sistemas multiprocesador, gracias a su implementación multi-hilo.
- Soporte de un número significativo de datos para las columnas.
- Posee API's para el soporte de varios lenguajes como: C, C++, Java, PHP, entre otros, lo que le permite gran portabilidad entre sistemas.
- Gestión de usuarios y contraseñas para el manteniendo de un excelente nivel de seguridad, incorporando un sistema de privilegios muy flexible que permite la verificación basada en *host*.
- Presenta un sistema de reserva de memoria muy rápido apoyado en *threads*, así como tablas *hash* en memoria, que son usadas como tablas temporales.
- Incorpora múltiples motores de almacenamiento (MyISAM, Margue, InnoDB, BDB, Memory/heap, MySQL Cluster, Federated, Archive, CSV, Blackhole y Example en 5.x), que le posibilita al usuario escoger el que sea más adecuado para cada tabla de la base de datos.
- Posee procedimientos almacenados, *triggers*, vistas actualizables y agrupación de transacciones.

Se puede decir que MySQL posee un muy buen rendimiento y velocidad a la hora de conectar con el servidor, así como a la hora de servir los datos, siendo esta una de sus principales ventajas. Está previsto, además, de varias utilidades de administración (*backup*, recuperación de errores, etc.). Hay que mencionar que este gestor en caso de tener problemas no suele corromper los datos ni perder información. Además, su escalabilidad posibilita manipular Bases de Datos enormes del orden de seis mil tablas, alrededor de cincuenta millones de registros y hasta 32 índices por tabla.

2.4.2 PostgreSQL

PostgreSQL es un sistema de gestión de Bases de Datos objeto-relacional (ORDBMS) basado en el proyecto Ingres, de la universidad de Berkeley, California. Está liberado bajo licencia BSD y utiliza el lenguaje SQL92/SQL99. Fue diseñado para ambientes de alto volumen de datos como Oracle, Sybase o Interbase, soportando transacciones. Desde su versión 7.0 soporta claves ajenas con comprobación de integridad referencial.

Pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, PostgreSQL incluye características de la orientación a objetos como pueden ser: la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional, además de otras específicas del gestor, como lo son: un mejor soporte para *sub-selects*, *triggers*, vistas y procedimientos almacenados.

Entre sus características más específicas se encuentran:

- Implementación el estándar SQL92/SQL99.
- Soporte para distintos tipos de datos, como son: datos de tipo fecha, datos sobre redes (MAC, IP…), cadenas de bits, etc., así como la creación de tipos de datos propios.
- Incorpora funciones de diversa índole como: funciones para el manejo de fechas, funciones geométricas, funciones orientadas a operaciones con redes, etc.
- Al igual que con los tipos de datos, PostgreSQL permite la declaración de funciones propias, así como la definición de dispensadores.
- Posee soporte para el uso de índices, reglas y vistas.
- Incluye herencia entre tablas, por lo que este gestor se incluye entre los gestores objeto-relacionales.
- Soporte en la mayoría de los sistemas operativos más utilizados incluyendo, Linux, varias versiones de UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, SunOS, Tru64), BeOS y Windows.

- Escritura adelantada de registros para evitar pérdidas de datos en caso de falla de energía, fallos del sistema operativo, y fallas de hardware.

Una de sus desventajas es la sobrecarga al sistema por el consumo de un número significativo de recursos.

2.5 Servidores Web

Los servidores Web constituyen una herramienta indispensable a la hora de la publicación y explotación de un sistema que esté desarrollado para su uso en la Web. Estos servidores contienen la lógica para la interpretación de las peticiones de los usuarios a través de los protocolos especificados, que hace posible la interacción dinámica entre el cliente y sus acciones sobre el sistema en que esté trabajando.

2.5.1 Apache

Acorde con el estudio realizado en el trabajo "*Estudio de alternativas para la migración del Sistema Automatizado para la Gestión Académica – Akademos*" del 2007 y al comparar los resultados obtenidos con los que brindan otras fuentes de mayor actualidad, Apache es sin dudas uno de los mejores servidores Web.

Apache ha sido, desde su salida al mercado, uno de los servidores de mayor popularidad, considerado por mucho el proyecto punta de la lanza del movimiento del Software Libre. Apache se caracteriza por ser un servidor ligero, con una alta configurabilidad y una amplia explotación, según Netcraft, empresa dedicada a la realización de encuestas a nivel global y estudios sobre el tráfico en internet, el mayor por ciento de los servidores web actuales son servidores Apache. (Ver figura 2.7)

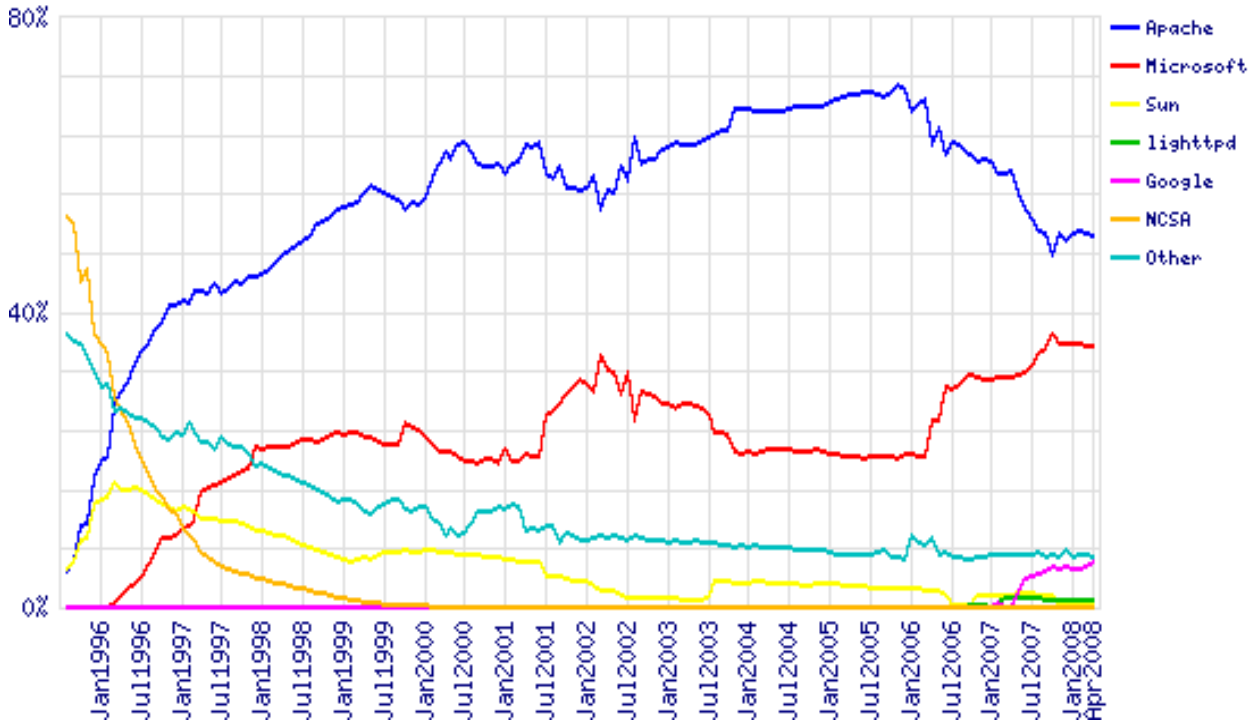


Figura 2.7: Análisis realizado por Netcraft de la explotación de Servidores Web

Entre algunas de las características más representativas de Apache se encuentran:

- Altamente configurable de diseño modular.
- Tecnología gratuita de código fuente abierto.
- Multiplataforma, funcionando tanto en Windows como en Linux o Unix.
- Soporte para CGI.
- Soporte para varios lenguajes: PHP, JAVA, Perl y librerías ASP.
- Soporte para el protocolo HTTP.
- Soporte de *host* virtuales.
- Servidor *proxy* integrado.

2.6 Propuesta tecnológica

Al analizar los resultados expuestos hasta el momento, se puede determinar una propuesta tecnológica en un entorno de desarrollo libre.

Mono podría ser una alternativa para el desarrollo del nuevo sistema, aprovechando la experiencia del equipo de desarrollo en la versión anterior del sistema con la tecnología dotNET. Esto tendría un costo mínimo y un esfuerzo casi nulo en la capacitación de este equipo sobre Mono, pero se estaría pagando un alto precio en cuanto a tiempo y cumplimiento de las tareas, pues no existe mucha documentación sobre este proyecto y no son muchos los grupos de debate y foros de discusión. Sería entonces poco práctico su uso y se podría incurrir en el atraso en la entrega de los primeros ejecutables. Por otra parte MonoDevelop, IDE para el desarrollo sobre Mono, ha liberado su versión 1.0 en marzo de este año, donde se han eliminado varios de los errores que contenían las versiones anteriores, pero estas mejoras no son significativas ante otros IDEs libres que presentan una mayor estabilidad y más facilidades de desarrollo.

Entre los lenguajes de programación, fue decisión por parte de la Dirección de Informatización y su grupo de arquitectura en conjunto con la dirección del proyecto y los resultados de esta investigación, de escoger PHP como lenguaje de programación base, y como framework para el desarrollo del sistema a Symfony.

La determinación de utilizar PHP con Symfony viabiliza que el producto o al menos las funcionalidades básicas sean liberadas en corto tiempo. Esto se sustenta en los años de experiencia y explotación del framework, así como por la amplia documentación con que cuenta. Otro factor importante que influyó en la toma de esta decisión es la curva de aprendizaje de la propuesta anterior, la cual es mucho menor que la de JEE y los frameworks que componen su plataforma, hecho que tiene un impacto positivo en el ahorro de tiempo y trabajo, pues el capital humano incorporado al proyecto, desde el punto de vista de los programadores, presenta conocimientos previos sobre PHP.

Como IDE a utilizar se elige Eclipse Europa 3.3.0 dada su vinculación con PHP a través del *plug-in* PDT y por sus características, que hacen del mismo, una herramienta potente en el trabajo con sistemas robustos.

Se debe puntualizar que el trabajo con JEE no se desdeña y puede ser considerado como propuesta a próximas versiones que se quieran hacer al sistema.

Como servidor web se elige Apache, uno de los servidores más estables y configurables en la actualidad, que soporta además un amplio número de lenguajes a ejecutar.

Tomando en cuenta las características expuestas en cada una de las ramas estudiadas, se determina escoger como gestor de Base de Datos a PostgreSQL, el cual es un potente sistema objeto-relacional que soporta: entidades referenciales, actualización en cascada, claves ajenas y un fuerte trabajo con objetos.

Otras herramientas a utilizar en la realización de este proyecto son: Visual Paradigm (VP) y DBDesigner, empleados en el modelamiento de los distintos artefactos de la metodología RUP así como de la Base de Datos respectivamente. Su estudio lo abordan los analistas y diseñadores.

Propuesta tecnológica general:

- Lenguaje de programación: PHP.
- Framework: Symfony y sub-frameworks asociados.
- IDE: Eclipse Europa 3.3.0 (PDT).
- Servidor Web: Apache 2.0.
- Gestor de Base de Datos: PostgreSQL 8.1.
- Diseño de la Base de Datos: DBDesigner.
- Diseño UML: Visual Paradigm.

La representación tecnológica se muestra en la figura 2.8, la cual se encuentra validada conjuntamente con el cliente en los requisitos no funcionales (ver anexo 9).

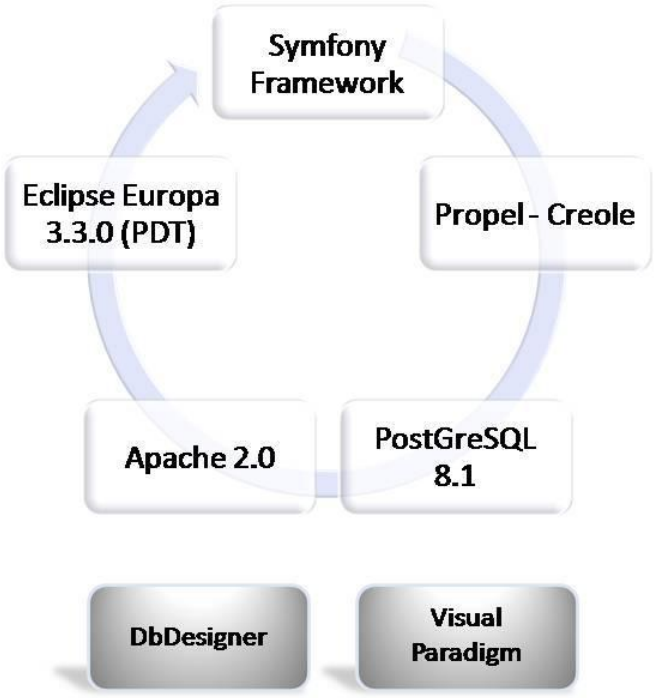


Figura 2.8: Representación tecnológica

2.7 Conclusiones

Se sientan las bases de la nueva tecnología a utilizar en el desarrollo del proyecto Akademos, analizando para ello, algunos de los frameworks y herramientas libres de mayor popularidad y eficiencia a la hora del desarrollo de aplicaciones web.

El empleo del lenguaje PHP conjunto con el framework Symfony y Eclipse Europa (PDT) como IDE de desarrollo, así como PostgreSQL como Gestor de Base de Datos en su versión 8.1 y Apache 2.0 como servidor Web, da la visión global de cómo es posible que quede confeccionado gran parte del ambiente de desarrollo.

Capítulo III: Definición de la arquitectura. Patrones y funcionalidades

3.1 Introducción

En el presente capítulo se realiza el estudio de la tecnología propuesta vinculada a la arquitectura de software. Se aborda cómo Symfony emplea en sus soluciones una serie de patrones básicos, y cómo estructura y determina las aplicaciones en dependencia de estos patrones.

3.2 Framework Symfony

Symfony es un framework para el trabajo con sistemas Web dinámicos. Permite la creación de páginas XHTML, depurar fácilmente las aplicaciones, abstraerse del gestor de Base de Datos y establecer una configuración sencilla basada en varios entornos de desarrollo.

Este framework, programado en PHP 5, simplifica la construcción de aplicaciones Web mediante la automatización de algunos patrones comunes para el desarrollo de este tipo de sistemas, por lo que obliga al desarrollador a escribir código de manera más legible y fácil de mantener.

3.2.1 Symfony sobre MVC

Symfony está basado en el patrón MVC (ver figura 3.1). Como se ha analizado, el MVC separa esencialmente la lógica del negocio de la lógica de la presentación, factor que posibilita la simplificación del trabajo y el mantenimiento de los sistemas.

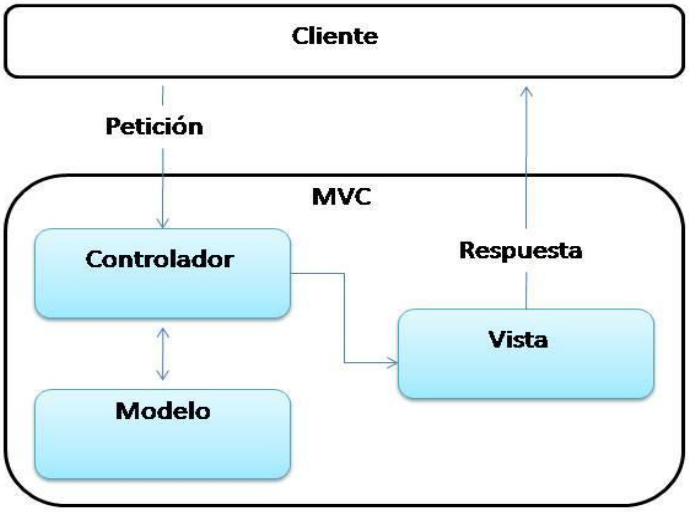


Figura 3.1: Representación MVC según el framework Symfony

La correcta implementación de este patrón dispone, como se observa en la figura anterior, de tres entidades básicas: la Vista, el Controlador y el Modelo.

La Vista se compone fundamentalmente de código HTML, y reduce al mínimo el código PHP utilizado, lo que aumenta las posibilidades de mantenimiento y abstracción.

Debido a que las páginas Web suelen contener elementos que se muestran de forma idéntica a lo largo de toda una aplicación, la Vista se separa, asimismo, en una Capa General Externa (*Layout*) y Plantillas (*Templates*).

La Capa General Externa constituye una estructura que presenta un determinado diseño y configuración. Se utiliza de manera global en todo el sistema o en grupos de páginas específicas. Por su parte, las Plantillas, sólo visualizan las variables definidas por el Controlador. Se aplican, de esta manera, los principios básicos del patrón Decorador al trabajo con las interfaces. (Ver figura 3.2)

Figura 3.2: Patrón Decorador

Otro aspecto importante de la organización del MVC en Symfony, es el Controlador. Normalmente el Controlador de una página suele tener mucho trabajo. Parte de este trabajo, como el manejo de las peticiones del usuario, el manejo de la seguridad y la carga de la configuración, es común a todos los controladores de la aplicación, por tal motivo Symfony suele dividirse en el Controlador Frontal, único en la aplicación y que agrupa todo el trabajo común, y las Acciones, que incluyen código y funciones específicas para el Controlador de cada página. (Ver figura 3.3)

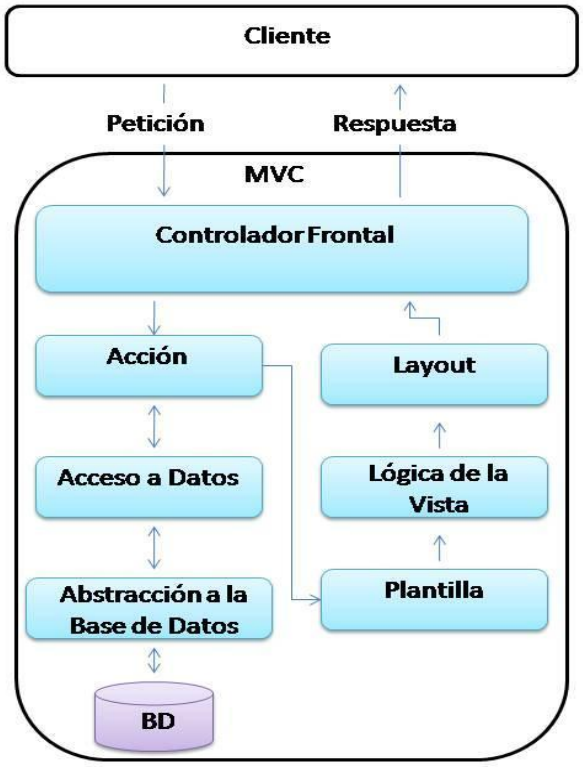


Figura 3.3: Flujo de Trabajo en Symfony

Esta disposición de los controladores en Symfony ofrece un único punto de entrada a la aplicación, lo que permite, en caso de que sea necesario, prohibir el acceso a toda la aplicación o a ciertos niveles dentro de esta, con sólo editar el archivo correspondiente y su *script* en el Controlador Frontal. Si la aplicación no dispone de un Controlador Frontal, habría que modificar cada uno de los controladores de las páginas.

Como se observa en la figura 3.4, existe una completa interacción entre las Acciones y las Plantillas, esto permite que los atributos de la clase sfAction estén completamente disponibles en la Plantilla en un ámbito global.

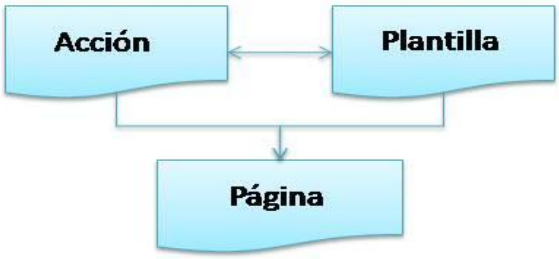


Figura 3.4: Página Symfony

Las clases que componen el Modelo se generan automáticamente en dependencia de la estructura de los datos de la aplicación, a través de Propel. Esta funcionalidad permite que el desarrollador se abstraiga de todo el trabajo tedioso que implica crear las clases y aplicar patrones, que se encuentran por defecto intrínsecos en el modelo de Symfony.

El MVC en Symfony está determinado, substancialmente, de la siguiente manera:

MVC Symfony:

- Vista:
  - Plantilla.
  - Capa General Externa.

- Controlador:
  - Acción.
  - Controlador Frontal.
- Modelo:
  - Acceso a Datos.
  - Abstracción de la Base de Datos.

La implementación que realiza Symfony del MVC incluye, al mismo tiempo, un conjunto de clases y variables para el trabajo configurativo centrado en su núcleo. Entre estas clases se encuentran:

- sfController: clase del controlador que interpreta las peticiones de los clientes y las envía a la acción correspondiente.
- sfRequest: clase que almacena todos los elementos que forman una petición (parámetros, cookies, cabeceras, entre otros).
- sfResponse: clase que contiene la cabecera de las respuestas y los contenidos. El contenido de un objeto de esta clase se transforma en HTML y se envía al usuario.
- Singleton de Contexto: se obtiene mediante la función *sfContext::getInstance()*, y almacena una referencia a todos los objetos del núcleo de Symfony.

3.2.2 Configuración

Symfony considera un proyecto como un conjunto de servicios y operaciones bajo un determinado nombre de dominio, que comparten el mismo modelo de objetos. Sobre este principio se organizan las aplicaciones dentro de un proyecto; estas, a su vez, se agrupan de forma lógica, pues normalmente una aplicación se ejecuta de forma independiente al resto.

Cada aplicación está formada por módulos y cada módulo representa una página Web o un grupo de páginas con un propósito determinado. Estos presentan acciones, las cuales incorporan cada una de las operaciones que los mismos ejecutan.

Como normalmente las aplicaciones Web comparten el mismo tipo de contenido, Symfony proporciona una estructura en forma de árbol de archivos para organizar los contenidos de forma lógica, siendo, además, consistente con el patrón MVC y configurable por el usuario en dependencia de sus necesidades, aunque se recomienda utilizar la estructura propuesta por este framework.

De este modo, Symfony define una serie de normas o conversiones que se ajustan a los requisitos habituales de las aplicaciones Web, lo que posibilita la configuración avanzada de la mayoría de los aspectos de una aplicación.

Algunas de las oportunidades que abre esta configuración son:

- Existencia de archivos de configuración a nivel de Proyecto, Aplicación y Módulo.
- Posibilidad de definir el conjunto de opciones de configuración, lote que Symfony denomina Entorno.
- Posibilidad de acceder a los archivos de configuración desde cualquier punto de la aplicación.

Los archivos de configuración de Symfony se encuentran bajo YAML, lenguaje que permite la creación de los archivos *.yml*. Este proceso no es estático, pues el desarrollador puede determinar el uso de este lenguaje o de otro como XML.

Symfony plantea un grupo de principios que identifican sus archivos de configuración:

- Potentes: máxima capacidad de gestión por parte de los archivos de configuración.
- Simples: las aplicaciones no tienen que tratar muchas características de configuración que habitualmente no usen.
- Sencillos: los archivos de configuración son fáciles de crear, leer y modificar.
- Personalizables: Los archivos están inicialmente creados en YAML, pero pueden cambiar a XML, INI, entre otros.

- Rápidos**:** la aplicación nunca procesa los archivos de configuración, de esto se encarga el sistema de configuración, lo que permite un procesamiento más rápido y eficiente.

Otro rasgo significativo de Symfony es que permite el acceso, desde cualquier punto del código, a las opciones de configuración de la aplicación, esto se logra mediante la clase sfConfig. Esta clase, básicamente, es un registro de opciones de configuración que proporciona métodos *getter/setter* que permiten el acceso a las mismas.

3.3 Entornos

Al elaborar una aplicación es común que se disponga de configuraciones diferentes, pero interrelacionadas. Para agrupar estas configuraciones se crea el concepto Entorno, entendido como un conjunto de configuraciones esenciales que presentan características similares.

Los entornos pueden ser creados por los propios desarrolladores, sin embargo, Symfony propone tres entornos base:

- **Entorno de producción:**
  - Registra los errores, pero no los alerta.
  - Ajusta las opciones de configuración para obtener el máximo de rendimiento.
- **Entorno de pruebas:**
  - Registra las alertas y los errores en el archivo *Log*.
  - Trabaja básicamente en una Base de Datos de pruebas.
  - Ejecuta líneas de comandos de pruebas.
  - No permite acceder a él mediante la URL, aunque simula el uso de *Cookies* y otros componentes PHP.
- **Entorno de desarrollo:**
  - Suele desactivar la Caché.

- Tiene activadas las opciones de *Log* y de depuración de aplicaciones, ofrece, además, mayor importancia al mantenimiento de la aplicación que a su rendimiento.

Cada Entorno define un Controlador Frontal específico, al cual se puede acceder mediante la URL; algo similar ocurre con la Base de Datos, pues la configuración de la misma puede variar entre un entorno y otro.

3.4 Propel

Propel constituye un servicio de objetos persistentes y de consulta que provee un sistema para el almacenamiento de objetos de una Base de Datos, así como de búsqueda y restauración de datos sin escribir una sola cláusula SQL. Esto garantiza que al utilizar Propel resulte más fácil el desarrollo, el despliegue y la migración de las aplicaciones a otro SGBD.

Propel puede ser descrito como un ORM, una capa ADO o una capa de objetos persistentes. Se sustenta en técnicas de acercamiento probado y ha sido optimizado para PHP.

Inicialmente, Propel implementa el patrón de entrada de datos por filas, *Row Data Gateway*, propuesto por M. Fowler, destacado orador y consultor en el diseño de software empresarial. Este patrón brinda una entrada de datos de fila que tienden a ser como el registro en su estructura, pero que puede ser accedido por los objetos y mecanismos habituales de programación.

Otro aspecto a destacar es que Propel, además, genera las clases para la implementación de otro patrón descrito también por Fowler: *Table Data Gateway*.

Propel presenta dos componentes principales:

- **Motor generador:** construye las clases y archivos SQL.
- **Ambiente de ejecución:** este componente proporciona un grupo de herramientas para crear consultas de SQL complejas y otro para el manejo de conexiones para múltiples Bases de Datos simultáneamente.

Todo este ambiente de ejecución provee de una capa de abstracción y otra de encapsulación de reglas de Bases de Datos para la lógica del negocio. (Ver figura 3.5)

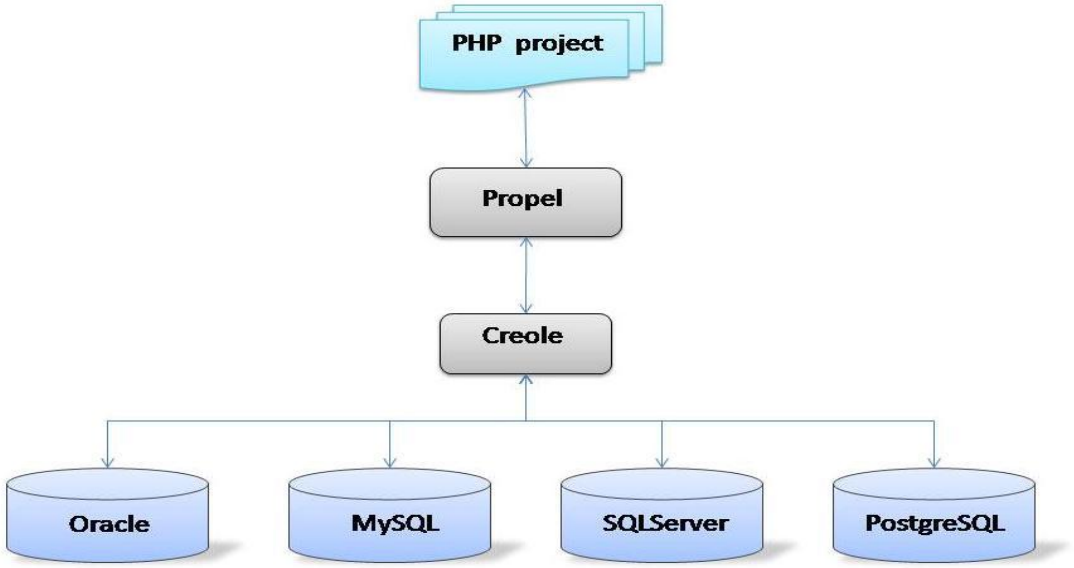


Figura 3.5: Representación ORM Propel

Propel, como se observa en la figura 3.5, usa Creole como capa de abstracción de Base de Datos, lo cual no quiere decir que siempre sea necesario, para lograr esta abstracción, la vinculación Propel-Creole; se puede lograr solamente con Creole. En este caso se encuentran los módulos que sólo requieren de consultas directas a la Base de Datos y no dependen de una lógica de negocio compleja, por lo que convertir los registros a objetos conllevaría a sobrecargar la solución y presentar deficiencias en tiempo y costo de la respuesta del sistema ante una solicitud del usuario.

3.5 MVC y su interacción con Propel

La lógica de las aplicaciones Web depende casi siempre de su modelo de datos. Para gestionar este proceso, Symfony contiene por defecto una capa de tipo ORM ejecutada mediante Propel, que permite que la conexión y la modificación de los datos se realicen mediante objetos, a través del acceso de forma implícita a la Base de Datos, lo que propicia un alto nivel de abstracción y una gran portabilidad.

Los ORM facilitan el acceso a la Base de Datos de forma efectiva desde un contexto orientado a objetos. Su interfaz hace posible la traducción de la lógica de objeto a la lógica relacional empleada por los SGBD, lo que viabiliza la reutilización de objetos y las llamadas a un objeto de datos desde distintos puntos de la aplicación, e incluso, desde diversas aplicaciones.

Para efectuar el mapeo de un modelo relacional a un modelo de objetos de datos, el ORM necesita, primeramente, la descripción del modelo relacional que se precisa en el archivo denominado esquema (*schema*). En este esquema se definen las tablas, sus relaciones y las características de sus columnas a través del formato YAML (*schema.yml*).

3.5.1 Clases del Modelo

Una de las mayores ventajas del esquema *schema.yml* es su utilidad en la generación de las clases del Modelo, lo que se concreta a través del comando *propel-buil-model*. (Ver anexo 2)

Al ejecutarlo se genera la siguiente estructura de clases en los directorios adecuados:

- BaseClase.php
- BaseClasePeer.php
- Clase.php
- ClasePeer.php

Clase constituye el nombre con que se ha definido en el esquema la denominación de la entidad correspondiente. Por cada entidad del esquema, se tienen cuatro clases en el Modelo: las del prefijo (Base) dentro del directorio *lib/model/om* y las demás dentro de *lib/model*. (Ver anexo 2)

La particularidad de tener esta estructura de clases y directorios se debe, sobre todo, a que en la medida en que avance el proyecto, si es necesaria la definición de nuevas funciones además de las ya existentes en alguna clase determinada, o si es preciso volver a generar el esquema por alguna modificación entre las entidades del mismo, se sobrescribe el modelo de clases obtenido anteriormente. Symfony, previendo estas y otras particularidades, subdivide el Modelo: las de prefijo (Base) son las que se crean nuevamente asumiendo los cambios. Es por ello que no se

recomienda incluir código personalizado en ellas, pues se perdería en caso de que ocurriese una regeneración del esquema. En cambio, las clases que se encuentran en *lib/model* no se modifican, heredan de sus clases (Base) correspondientes y adquieren, por ende, sus métodos. Es, en estas áreas, donde se incluirían métodos propios, personalizándolas.

Este mecanismo hace posible comenzar a desarrollar, sin tener en cuenta el modelo relacional definitivo de la Base de Datos, lo cual permite la evolución del mismo.

3.5.2 Patrones del Modelo

Symfony permite extender el modelo existente una vez generado, lo que a su vez propicia que se puedan redefinir en las clases personalizadas métodos de las clases (Base), agrupando así nuevas funcionalidades y reglas del negocio.

Como se aprecia, Symfony divide el modelo en clases (Base) -de acceso a datos- y clases hijas de estas clases (Base), que son personalizadas en dependencia del negocio. Dentro de estas clases se crean, además, otros dos subconjuntos: las clases Objeto y las clases Peer.

Las clases Objeto representan un registro en la Base de Datos que proporciona el acceso a las columnas de un registro y a sus registros asociados.

Clases Objeto:

- Base<u>Clase</u>.php
- <u>Clase</u>.php

Por su parte, las clases Peer contienen métodos estáticos para el trabajo con las tablas de la Base de Datos, de forma tal que facilita los medios necesarios para obtener los registros correspondientes a una tabla determinada.

Clases Peer:

- BaseClasePeer.php
- ClasePeer.php

Tal vez uno de los aspectos más relevantes en el uso del ORM de Symfony, Propel, para la generación del Modelo en la solución, constituye la aplicación de varios patrones arquitectónicos a la hora de confeccionar el mismo. Propel implementa un conjunto de patrones arquitectónicos siguiendo los principios que expresa M. Fowler en su libro "*Patterns of Enterprise Application Architectura"* (Fowler, 2002). Estos patrones convierten el Modelo generado en una estructura altamente flexible que presenta, entre otras ventajas, una mayor escalabilidad y portabilidad.

Dentro de las clases Objeto, el conjunto de clases (Base) representa el patrón *Row Data Gateway* (RDG), el cual asume el comportamiento de un objeto que actúa como puerta de enlace a una fila de una tabla en la Base de Datos; brinda objetos que representan registros de una fila en alguna tabla de la Base de Datos, pero tiene la particularidad de facilitar el acceso a dichos objetos con los mecanismos regulares de programación del lenguaje utilizado. (Ver figura 3.6)

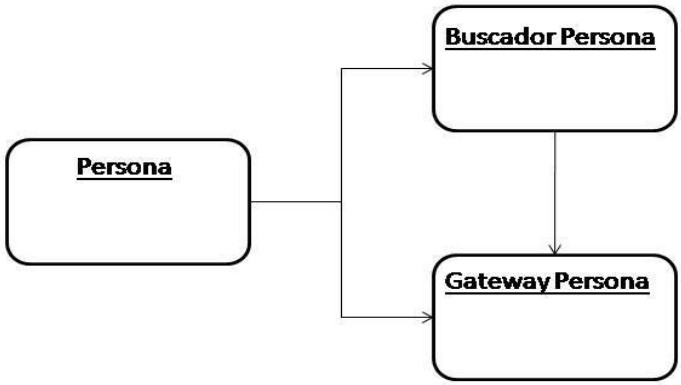


Figura 3.6: Patrón *Row Data Gateway*

Cada columna en la Base de Datos, al emplear este patrón, se convierte en un campo que actúa como interfaz por cada registro de datos. Por otra parte, es posible acudir a métodos estáticos de acceso a datos, así como a clases polimórficas. Es necesario tener presente que por cada

tabla relacional en la Base de Datos es preciso tener un *Gateway* y la clase Buscador correspondiente.

El análisis de las clases Peer arroja como conclusión que en este entorno también se implementa otro patrón de acceso a datos en el marco de sus clases (Base): el TblDG. La particularidad de este patrón, a diferencia del RDG, que define la estructura de acceso por registro en las entidades de la Base de Datos, es que especifica su acceso a nivel de tabla, proponiendo un objeto que se comporte como una puerta de enlace a cada tabla de la Base de Datos. (Ver figura 3.7)

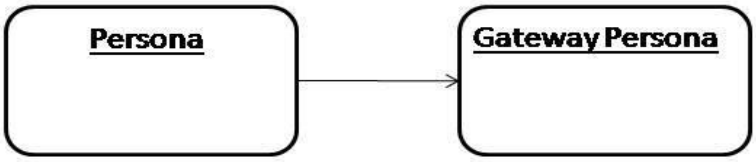


Figura 3.7: Patrón *Table Data Gateway*

Este patrón ofrece una interfaz simple, que usualmente consiste en varios métodos de acceso a datos. Cada uno de estos métodos mapea los parámetros de entrada en consultas SQL y ejecuta estos *scripts* SQL contra la Base de Datos definida previamente en la cadena de conexión. A pesar que la implementación de este patrón consiste en definir una interfaz interactiva por cada tabla de la Base de Datos, puede ser concebido, de igual forma, como un *Gateway* para casos muy sencillos en el manejo de todas las tablas de la Base de Datos.

TblDG funciona de manera eficiente en interacción con el patrón de dominio *Table Model*, ofreciendo las estructuras de registros de datos sobre los cuales este último trabaja. El patrón *Table Model* es propuesto por Symfony para ser utilizado como un conjunto de clases estáticas a través de las clases Peer, pero no (Base), sino las clases personalizables que son extensibles en el dominio o negocio del sistema.

El patrón *Table Model* provee un objeto para el manejo del comportamiento en cada una de las tablas de la Base de Datos, por tanto, por cada tabla de la Base de Datos se tendría una clase de dominio en interacción directa con el TblDG. Esto se aprecia en Symfony al heredar en el modelo las Clases Peer personalizables de las Clases Peer (Base).

Table Model se distingue de otros patrones de dominio, pues ofrece un objeto para el manejo de un conjunto de varias órdenes, permitiendo con ello el empaquetamiento de los datos y el comportamiento de los mismos de manera integrada, lo que garantiza, a su vez, explotar al máximo las potencialidades de una Base de Datos relacional.

3.6 Acceso a datos

En cuanto a la actualización de los datos, Propel detecta las relaciones entre los distintos objetos y se ocupa de la inserción y la actualización de los mismos, en dependencia de la funcionalidad deseada, todo a nivel de objeto.

A pesar que el modelo de datos es independiente de la Base de Datos utilizada, resulta imprescindible indicarle la ruta correcta donde se encuentran sus datos. La información mínima requerida por Symfony para ejecutar las operaciones con la Base de Datos exige: nombre, los datos de acceso y el SGBD que la soporta. Esta información se indica en el archivo *database.yml*.

Las opciones de conexión a una Base de Datos se establecen para cada entorno de trabajo. Se definen, para ello, diferentes opciones de conexión para los entornos de producción, de pruebas y de desarrollo, así como para cualquier otro entorno definido por el desarrollador. También, esta configuración es posible definirla para cada aplicación, estableciendo diferentes valores para las opciones de un archivo específico de la aplicación. De esta forma, se puede disponer de políticas de seguridad diferentes en cada aplicación pública y en las aplicaciones de administración del proyecto, además de especificar distintos usuarios de la Base de Datos con privilegios disímiles.

Symfony soporta los siguientes SGBD, los cuales se determinan en el atributo *phptype*:

- MySQL
- SQLServer
- PostgreSQL
- SQLite

- Oracle

El hecho de que Symfony acuda a un ORM, pudiera valorarse como un inconveniente que lo sitúa en desventaja con respecto a otros frameworks; puesto que necesitaría definir la estructura de datos dos veces: una, para la Base de Datos y otra, para el Modelo de Objetos. No obstante, Symfony presenta utilidades de línea de comandos que permiten generar un modelo con respecto a otro. Por ejemplo: el comando *propel-build-sql* crea un archivo *lib.model.schema.sql* con el esquema del modelo y se optimiza el código SQL para el SGBD establecido en el parámetro *phptype* del archivo *propel.ini*. Es posible también aprovechar el archivo *schema.sql* para construir la Base de Datos. Este código SQL generado es válido para construir la Base de Datos en otro Entorno o para cambiar de SGBD, si el archivo *propel.ini* contiene las opciones de conexión correctas a la Base de Datos. De esta manera, el comando *propel-insert-sql* se encarga de crear automáticamente las tablas.

Es preciso aclarar que este proceso puede producirse de forma retroactiva: a partir de un modelo de datos existente en una Base de Datos, se genera el esquema correspondiente, así como el modelo de objetos vinculados a dicho modelo de datos. Esto se logra a través de la Introspección, capacidad de la Base de Datos de determinar la estructura que la conforma.

3.7 Estructura arquitectónica general

Para el correspondiente completamiento del análisis sobre los patrones aplicables a la variante tecnológica escogida, resulta imprescindible determinar cómo se ajustan estas posibilidades de Symfony a la solución que constituye el objeto de esta investigación.

Aspectos como qué o cuál patrón aplicar a la solución quedan esclarecidos a partir de las funcionalidades expuestas anteriormente. Se ajusta, de esta manera, la solución a lo propuesto por Symfony en cada punto tratado, aunque de necesitarse incluir algún otro patrón o declinar el uso de algunos de los que propone el framework, se podrán hacer los cambios pertinentes con previo análisis y autorización. Se hace énfasis, por otra parte, en el Modelo y su interacción con la Base de Datos, buscando mejores prácticas de programación y extendiendo correctamente dicho Modelo.

La solución se divide en módulos que hacen referencia a cada una de las áreas de trabajo definidas previamente por la dirección del proyecto. Cada módulo se traduce, en función de Symfony, en una aplicación (ver figura 3.8). Esta división implica una mejor distribución y facilidades de trabajo, que abordar todo el grueso del sistema en una aplicación general.

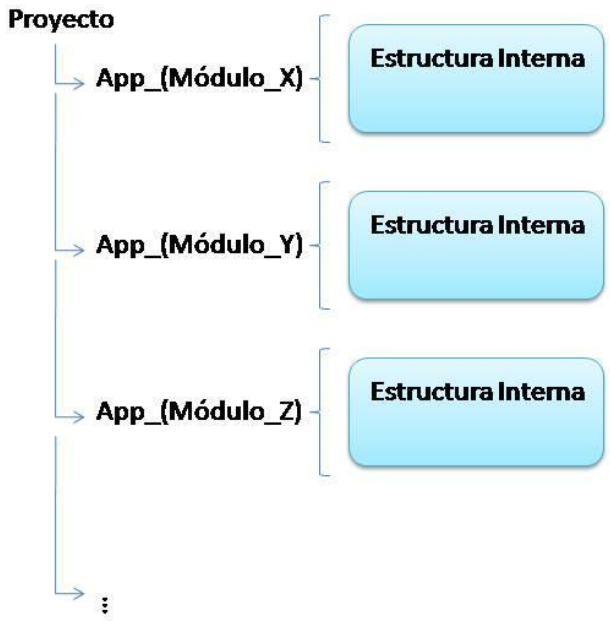


Figura 3.8: Representación de la estructura modular en Symfony

Un módulo representa una entidad independiente, pero integrada en la solución general, buscando con esto un bajo acoplamiento entre todos los módulos del sistema. Este aspecto es interesante, pues cada uno de estos módulos define su modelo de datos y por ende de objetos. Esta organización es fundamental para el nuevo sistema, porque facilita su estructuración en partes independientes, instalables e integradas.

La aplicación sigue los principios de Symfony en cuanto a la aplicación de sus patrones. Abstrayéndonos del patrón MVC que propone Symfony, se puede delimitar un híbrido entre MVC[1] y el estilo de Tres Capas, ya que en dependencia de la restructuración interna del sistema, es

[1] **MVC:** En algunas de las fuentes utilizadas el patrón Modelo-Vista-Controlador al definir la estructura de una aplicación de manera básica y modular, se define como un Estilo Arquitectónico en vez de como un Patrón Arquitectónico.

factible la combinación, en una vista diferente, de estos dos estilos de forma general. (Ver figura 3.9)

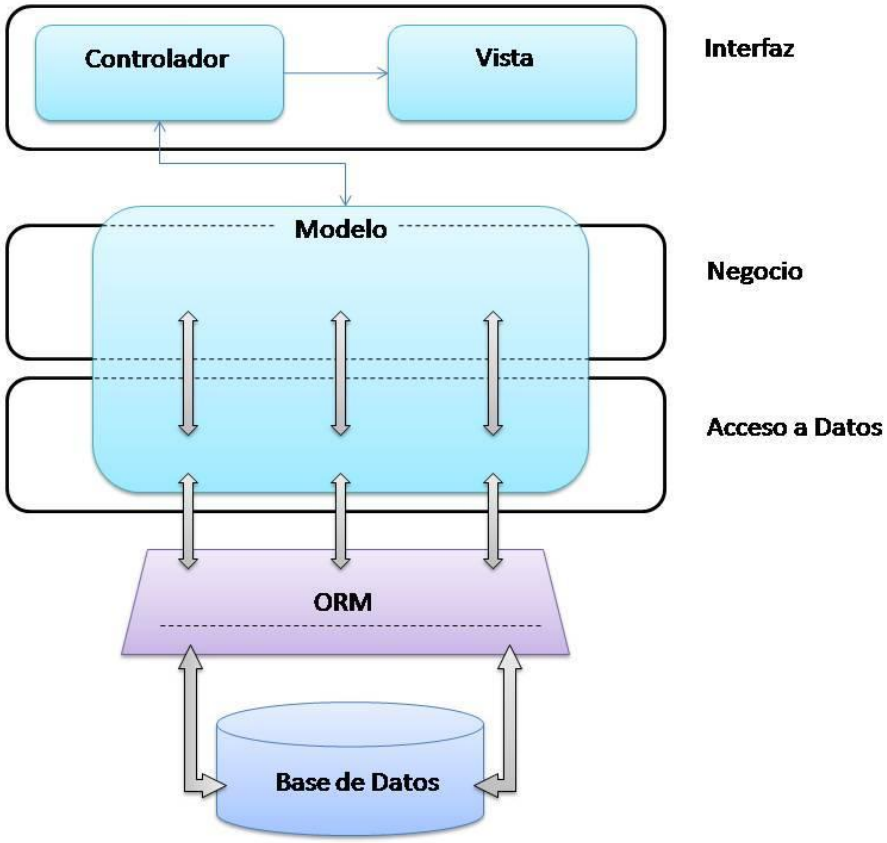


Figura 3.9: Reestructuración de MVC sobre Arquitectura en Tres Capas

Por otra parte, el sistema contará con un módulo administrativo, cuya principal tarea es centralizar el manejo de funcionalidades configurativas de cada módulo en particular. Esto revertirá en una mayor organización de los recursos administrativos en una entidad básica, funcionalidad que la actual versión del sistema no presenta. (Ver figura 3.10)

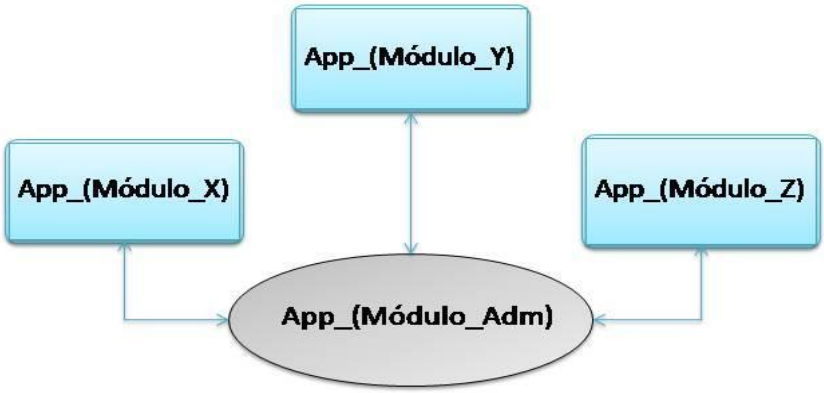


Figura 3.10: Representación de la interacción modular del sistema

Esta integración no es estática, sino dinámica, pues a la hora de instalarse un nuevo módulo sus partes administrativas y configurativas pasan al módulo de Administración.

La instalación de un módulo conlleva a la sobre-escritura y, por ende, a la actualización de las referencias y las dependencias entre el módulo que se instala y los demás existentes.

El despliegue general del sistema se encuentra definido en el anexo 1.

3.7.1 Ambiente de desarrollo

El entorno de trabajo contará con un grupo de equipos destinados al desarrollo y otro, más reducido, utilizado como servidores.

La solución estará centralizada con un servidor base, mediante el empleo del controlador de versiones Subversion. Este software permite mantener una retroalimentación entre los desarrolladores y la solución general, así como el control de cada una de las actualizaciones efectuadas sobre la misma.

En este servidor se pretende la instalación de Symfony y los tres entornos que trae por defecto para el desarrollo: entorno de pruebas, para la realización de pruebas a la solución, entorno de desarrollo y entorno de producción.

En los equipos de desarrollo además del IDE Eclipse (PDT), Apache 2.0, y el Framework Symfony, se tendrá configurado el entorno de desarrollo, que suele desactivar la Caché y tiene

activadas las opciones de *Log* y de depuración de aplicaciones, así como el entorno de producción, donde se ajustan las opciones de configuración para obtener el máximo de rendimiento. Una vez que se realice una nueva funcionalidad en el entorno de desarrollo, se recomienda pasar al de producción para su optimización.

Por otra parte, se tendrá un servidor PostgreSQL, donde se albergará la Base de Datos, y un servidor Web donde se montará el entorno de pruebas con la solución general. (Ver figura 3.11)

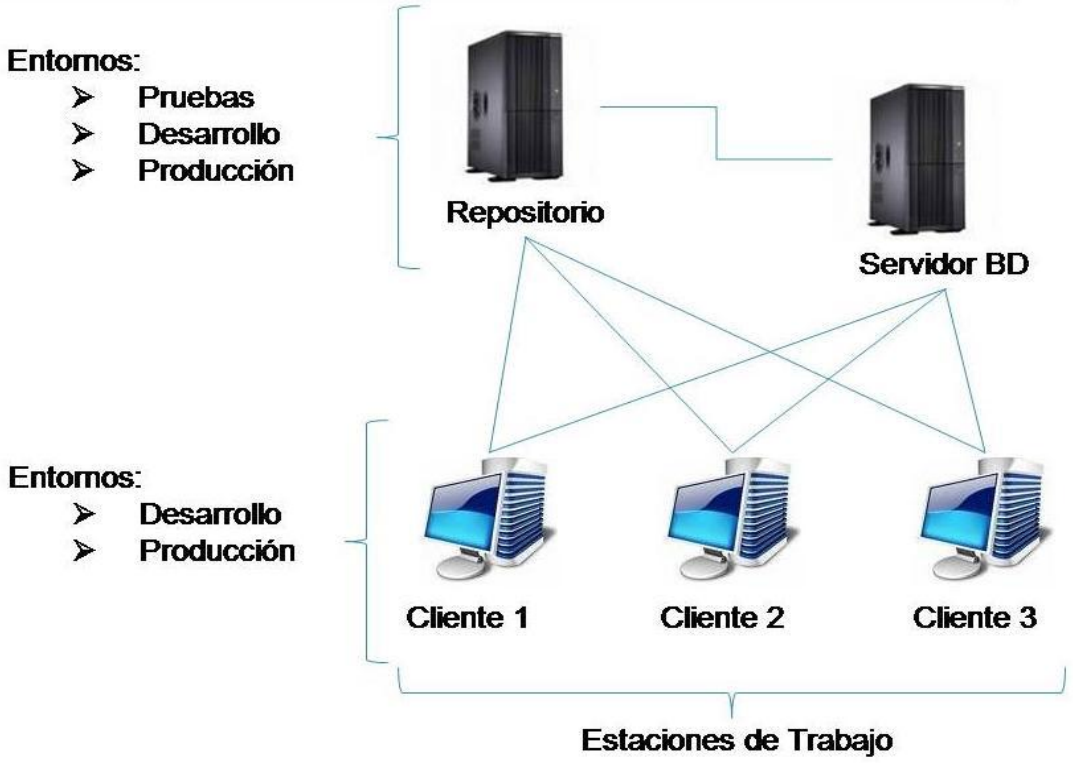


Figura 3.11: Representación del Ambiente de Desarrollo

Un aspecto importante es el uso del DBDesigner. Esta herramienta no se aborda profundamente en este estudio, debido a que constituye una herramienta para el diseño de la Base de Datos. Su análisis, por tanto, atañe directamente a los diseñadores de la misma. Su impacto en el marco de desarrollo del sistema, determinó sin embargo, la razón por la cual se elige como parte del entorno de trabajo.

Pudiera señalarse que existen dos herramientas para la realización del modelamiento de la Base de Datos, y es cierto: Visual Paradigm permite hacer el diseño de clases persistentes y de ahí, pasar al modelo Entidad-Relación. No obstante, el *script* generado no es compatible con Propel, ORM que constituye uno de los pilares fundamentales de Symfony. El problema anterior se resuelve al utilizar DBDesigner. Con esta herramienta, una vez terminado el diseño, se genera un archivo *model.xml* que contiene el modelo de la Base de Datos, el cual es transformado en un archivo *schema.xml* que puede ser interpretado por Propel. Este proceso se lleva a cabo mediante otros dos ficheros: *transform.php* y *dbd2propel.xsl*, que fueron obtenidos en el trascurso del proceso investigativo y han sido probados previamente. La transformación ocurre al ejecutar la lógica contenida en el *transform.php* que toma el modelo proveniente del *model.xml* y, siguiendo las especificaciones del archivo *dbd2propel.xsl*, genera el nuevo esquema (*schema.xml*).

Lógica del (*transform.php*):

```
<?PHP
    $xml = new DOMDocument;
    $xml->load('model.xml');
    $xsl = new DOMDocument;
    $xsl->load('dbd2propel.xsl');
    $proc = XSLProcessor();
    $proc->importStyleSheet($xsl);
    $schema_xml = $proc->transformToXML($xml);
    File_put_contents('schema.xml', $schema_xml);
?>
```

3.7.2 Módulos del sistema

El sistema cuenta con un conjunto de módulos determinados por cada una de las áreas de investigación del equipo de desarrollo, los cuales fueron definidos conjuntamente con el grupo de arquitectura. Estos son:

- Administración.
- CICE.
- Control Docente.
- Matrícula.
- Plan de Estudio.
- Profesor.
- Postgrado.
- Seguridad.
- Tesis.
- Reporte.

De este conjunto de módulos se determinó que sólo seis de ellos pasan a la primera iteración del sistema, por su importancia y prioridad dentro de la solución. Estos son: Administración (ver anexo 3), Matrícula (ver anexo 4), Profesor (ver anexo 5), Postgrado (ver anexo 6), Seguridad (ver anexo 7) y Tesis (ver anexo 8).

3.8 Conclusiones

Symfony constituye una poderosa herramienta para el desarrollo de sistemas robustos sobre la Web. Este framework cimenta su estructura en un número significativo de patrones, lo que obliga al desarrollador a seguir buenas prácticas de programación y de diseño. Patrones como MVC, TblDG o RDG, lo demuestran.

Symfony interactúa conjuntamente con otros sub-frameworks para gestionar aspectos como el acceso a datos o la generación de un grupo de clases básicas en dependencia del modelo de datos. Contiene, además, una fuerte estructura administrativa que le permite definir grupos de configuraciones aglutinadas en Entornos, según las necesidades del desarrollador o el equipo de trabajo.

La vinculación e interacción de los patrones propuestos por Symfony, así como su funcionamiento interno, confluyen finalmente en un modelado del sistema subdividido en capas, donde se definen un grupo de módulos básicos que configuran la base de la solución.

Conclusiones generales

- El sistema Akademos está actualmente basado en tecnología dotNET, la cual es propietaria y no cumple con las políticas que lleva a cabo nuestro país en la migración de sus sistemas a plataformas libres. Akademos no constituye un producto y, por tanto, no se puede comercializar como tal.

- La implementación de la actual versión del sistema Akademos está regida por la estrategia arquitectónica de Microsoft y se basa en un grupo de patrones que pueden ser reutilizados en el nuevo desarrollo.

- Para el desarrollo de la nueva versión del sistema Akademos se define utilizar Symfony, framework libre que permite la construcción de sistemas Web complejos y que presenta un número elevado de posibilidades y facilidades para el mantenimiento, actualización y soporte de los mismos.

- Se propone el lenguaje de programación PHP 5, el cual es orientado a objetos y presenta amplias ventajas en su interacción con el framework Symfony, como (IDE), Eclipse 3.3.0 en unión con el *plug-in* PDT para Eclipse, como servidor Web, Apache 2.0 y como SGBD, PostgreSQL, dadas las amplias prestaciones que brinda este sistema.

- En correspondencia con el estudio realizado se define utilizar la arquitectura que expone Symfony, la cual basa sus aplicaciones en el patrón MVC.

- La estructura de la solución estará determinada por el árbol de archivos que organiza Symfony, ubicando cada uno de los módulos del sistema como aplicaciones dentro del framework y definiendo de manera particular su modelo de datos y lógica de negocio.

- Al utilizar Symfony el desarrollo del sistema estará regido por las mejores prácticas, pues este framework incorpora un conjunto de patrones arquitectónicos que hacen de sus soluciones proyectos más robustos.

Recomendaciones

- A pesar que en esta investigación se abordan los elementos más representativos del framework Symfony, estos no son suficientes, existe otro número de elementos que influyen de manera positiva en el funcionamiento del mismo, por lo que se recomienda un estudio bibliográfico de la temática.
- Realizar un análisis sobre el trabajo con Servicios Web desde Symfony para su implementación en futuras iteraciones del software, permitiendo con ello la integración del sistema con otras aplicaciones.

Bibliografía

AUTORES, C. D. Application Architecture for .NET: Designing Applications and Services. 2002.

FOWLER, M. Patterns of Enterprise Application Architecture. 2002.

AUTORES, C. D. Enterprise Solution Patterns Using Microsoft .NET. 2003.

REYNOSO, C. y KICILLOF, N. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Marzo 2004.

REYNOSO, C. B. Introducción a la Arquitectura de Software. 2004.

FRANCIA, J. Desarrollo de una Aplicación en tres Capas con VS dotNET. 2006, Disponible en: http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art140.asp

CEPERO, D. y BENAVIDES, F. Estudio de alternativas para la migración del Sistema Automatizado para la Gestión Académica de la Universidad de las Ciencias Informáticas – "Akademos" a Software Libre. 2007.

INDA, E. Centro de Validación y Certificación de Competencias Profesionales UCI. 2007.

PIMENTEL, L. A. y PÉREZ, I. ArBaWeb: ARQUITECTURA BASE SOBRE LA WEB". 2007.

SOMARRIBA, A. y ECHEMENDIA, Y. Arquitectura J2EE aplicada en el módulo Servicio Autónomo. 2007.

AB, M. MySQL 5.0 Reference Manual. 2008, Disponible en: http://dev.mysql.com/doc/refman/5.0/en/

AUTORES, C. D. Web Server Survey. 2008, Disponible en: http://news.netcraft.com/archives/web_server_survey.html

AUTORES, C. D. Proyecto Mono. 2008, Disponible en: http://es.wikipedia.org/wiki/Proyecto_Mono

AUTORES, C. D. MonoDevelop. 2008, Disponible en: http://es.wikipedia.org/wiki/MonoDevelop

AUTORES, C. D. Java EE. 2008 Disponible en: http://es.wikipedia.org/wiki/JEE

AUTORES, C. D. Comparación de sistemas administradores de bases de datos relacionales. 2008, Disponible en: http://es.wikipedia.org/wiki/Comparaci%C3%B3n_de_sistemas_administradores_de_bases_de_datos_relacionales

POTENCIER, F. y ZANINOTTO, F. Symfony, la guía definitiva. 2008.

TEAM, M. MonoDevelop 1.0 Released. 2008, Disponible en: http://www.monodevelop.com/MonoDevelop_1.0_Released

Glosario de términos

Estructuras Administrativas: Estructuras administrativas de la institución, Eje: Facultad 1 y Facultad 2, que son estructuras de tipo facultad, y 1501 y 1403, que son de tipo grupo.

Estructuras: Estructuras de un centro de estudios. Se describen como un conjunto de estructuras de diferentes tipos, pueden ser facultades, grupos docentes, etc., que se organizan jerárquicamente para agrupar por un criterio determinado tanto a estudiantes como a los demás implicados en la labor docente, como: profesores, personal de secretaría y directivos docentes.

XML: Acrónimo de *Extensible Markup Language* (Lenguaje de Marcado Extensible).

REST: *Representational State Transfer*, modelo de arquitectura de software, generalmente aplicado a sistemas de hipermedia (Web).

IEEE: Instituto de Ingenieros Eléctricos y Electrónicos, constituye una asociación técnico-profesional internacional dedicada a la estandarización.

Gateway: Patrón Arquitectónico Base descrito por Martin Fowler. Se define como un objeto que encapsula lógica de acceso a un sistema o recurso externo.

DataSets: Plural de *DataSet*, objeto de la librería (*System.Data*), que funciona como un repositorio de datos sobre el cual se pueden realizar un número significativo de funcionalidades en el manejo de datos.

ADO.NET: *ActiveX Data Object*. Objetos para la conexión y trabajo con los repositorios de datos sobre la plataforma dotNET.

ADO: *ActiveX Data Object*. Objetos para la conexión y trabajo con los repositorios de datos.

HTML: Lenguaje Marcado de Hipertexto, diseñado para estructurar textos y presentarlos en forma de Hipertextos, que es el formato estándar de las páginas WEB.

Microsoft IntelliSense: Gama de opciones que facilitan el acceso a las referencias del lenguaje utilizado al programar con el framework dotNET, constituye una forma automatizada de autocompletamiento de código.

RecordSet: Estructura de datos cuya utilidad es la de almacenar desde una tabla de la Base de Datos.

Mono: Proyecto de código abierto impulsado por la empresa Novell, es compatible con además con algunos componentes de la plataforma dotNET.

Ximian: Empresa proveedora de Software Libre, fundada por Miguel de Icaza y Nat Friedman en 1999.

Creole: Sistema de abstracción de la Base de Datos, similar a los PDO, que proporciona una interfaz entre el código PHP y el código SQL,

Novell: Compañía estadounidense productora de Software. Compra los derechos sobre Ximian en agosto del 2003.

CLI: Acrónimo de *Common Language Infraestructure* (Leguaje Común de Infraestructura).

MSIL: Acrónimo de *Common Intermediate Language* (Lenguaje Común Intermedio), código intermedio entre código de alto nivel y la máquina. Lenguaje ensamblador orientado a objeto basado en pilas.

CLR: Acrónimo de *Common Language Runtime* (Lenguaje Común en Tiempo de Ejecución), implementación estándar del *Common Language Infraestructure* (CLI) que define un ambiente de ejecución códigos de programas.

XML WebServices: Conjunto estándar de protocolos basados en XML que permiten el intercambio de datos, desde la web, entre aplicaciones distintas, sin tener en cuenta el lenguaje de programación de la aplicación o la plataforma donde está implantada la misma.

Middleware: Software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas en ambientes heterogéneos.

ASP.NET: Acrónimo de *Active Server Pages* (Páginas Activas del Servidor). Tecnología del lado del servidor para páginas Web dinámicas. Herramienta de desarrollo para sitios Web dinámicos construida por Microsoft y que forma parte de su plataforma dotNET.

ASP: Acrónimo de *Active Server Pages* (Páginas Activas del Servidor). Tecnología del lado del servidor para páginas Web dinámicas.

DataGrid: Control de Visual Studio que permite mostrar datos de una consulta a una Base de Datos de forma sencilla, enlazando los datos que muestra a los elementos del origen de los mismos. Permite seleccionar, ordenar y editar sus elementos.

Script: Conjunto de instrucciones que permiten la automatización de tareas creando pequeñas utilidades.

DataSources: Origen de Datos.

DataRows: Objeto que representa una fila de una tabla en memoria.

DataTables: Objeto que representa una tabla de una Base de Datos en memoria.

DataBinding: Propiedad o mecanismo que presentan algunos componentes de estar pendientes de los cambios en otra variable, objeto o función.

Caché: Conjunto de datos duplicados de los datos originales, con la propiedad de que los datos originales son más costosos de acceder respecto a sus copias en la Caché.

Template: Plantilla que contiene el diseño de la página Web.

ContentPlaceHolder: Define una región editable para el contenido de una página principal (*Master Pages*) en ASP.NET.

Login: Autenticación.

SDK: Acrónimo de *Software Development Kit* (Kit para el Desarrollo de Software). Conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema en concreto, ejemplo: paquetes de software, frameworks, videoconsolas, sistemas operativos, etc.

Metadata: Constituyen datos acerca de otros datos. Se usan para facilitar el entendimiento de las características de los datos, así como la administración y el trabajo con los mismos.

Bugs: Errores o problemas en aplicaciones informáticas.

Perl: Acrónimo de Lenguaje Práctico de Extracción e Informe. Lenguaje de programación interpretado que toma características de C.

C: Lenguaje de programación.

CGI: Acrónimo recursivo de *Common Gateway Interface* (Interfaz de Entrada Común). Tecnología que permite a un cliente solicitar datos de un servidor Web.

Cookies: Fragmento de información que se almacena en el disco duro de un visitante a una página Web a través de su navegador a petición del servidor de la página.

MIT: Acrónimo de *Massachusetts Institute of Technology* (Instituto Técnico de Massachusetts).

Ruby: Lenguaje de programación reflexivo orientado a objetos.

Threads: Hilos de ejecución que permiten la realización de tareas concurrentes.

Triggers: Eventos que se ejecutan cuando se cumple una condición establecida al realizar una operación de (*Insert*), (*Update*) o (*Delete*), en una Base de Datos.

FTP: Protocolo de transferencia de ficheros entre sistemas conectados a una red TCP basado en arquitectura Cliente-Servidor.

SMTP: Protocolo red basado en texto que permite la transferencia de correos electrónicos.

NNTP: Protocolo usado para la lectura y publicación de noticias.

HTTP: Protocolo de transferencia de hipertexto. Es el protocolo usado para las transferencias de la Web.

HTTPS: Versión segura del protocolo HTTP. El sistema HTTPS utiliza un cifrado basado en *Secure Socket Layers* (SSL) para crear un canal cifrado de transferencia entre el servidor remoto y el cliente.

ORM: Acrónimo de *Object-Relational Mapping* (Mapeo Objeto-Relacional). Framework que utiliza técnicas de programación para convertir datos a objetos y viceversa, permitiendo el trabajo con datos persistentes como si formaran parte de una Base de Datos orientada a objetos.

YAML: Acrónimo de *Yet Another Markup Language* (No es otro Lenguaje de Marcado). Formato de serialización de datos inspirado en lenguajes como XML, Python y Perl, así como el formato para correos electrónicos especificado por el estándar RFC 2822.

PDO: Acrónimo de *PHP Data Objects* (Objetos de Datos para PHP). Extensión que provee una capa de abstracción de acceso a datos para PHP 5.

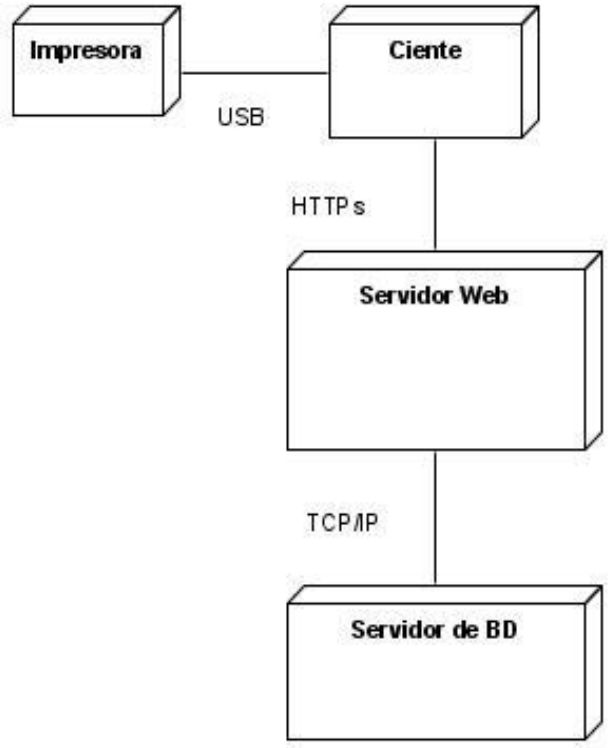
CICE: Acrónimo de Centro de Innovación y Calidad de la Educación.

Servlet: Objeto que se ejecuta en un servidor o contenedor JEE, especialmente diseñado para ofrecer contenidos dinámicos desde un servidor Web.

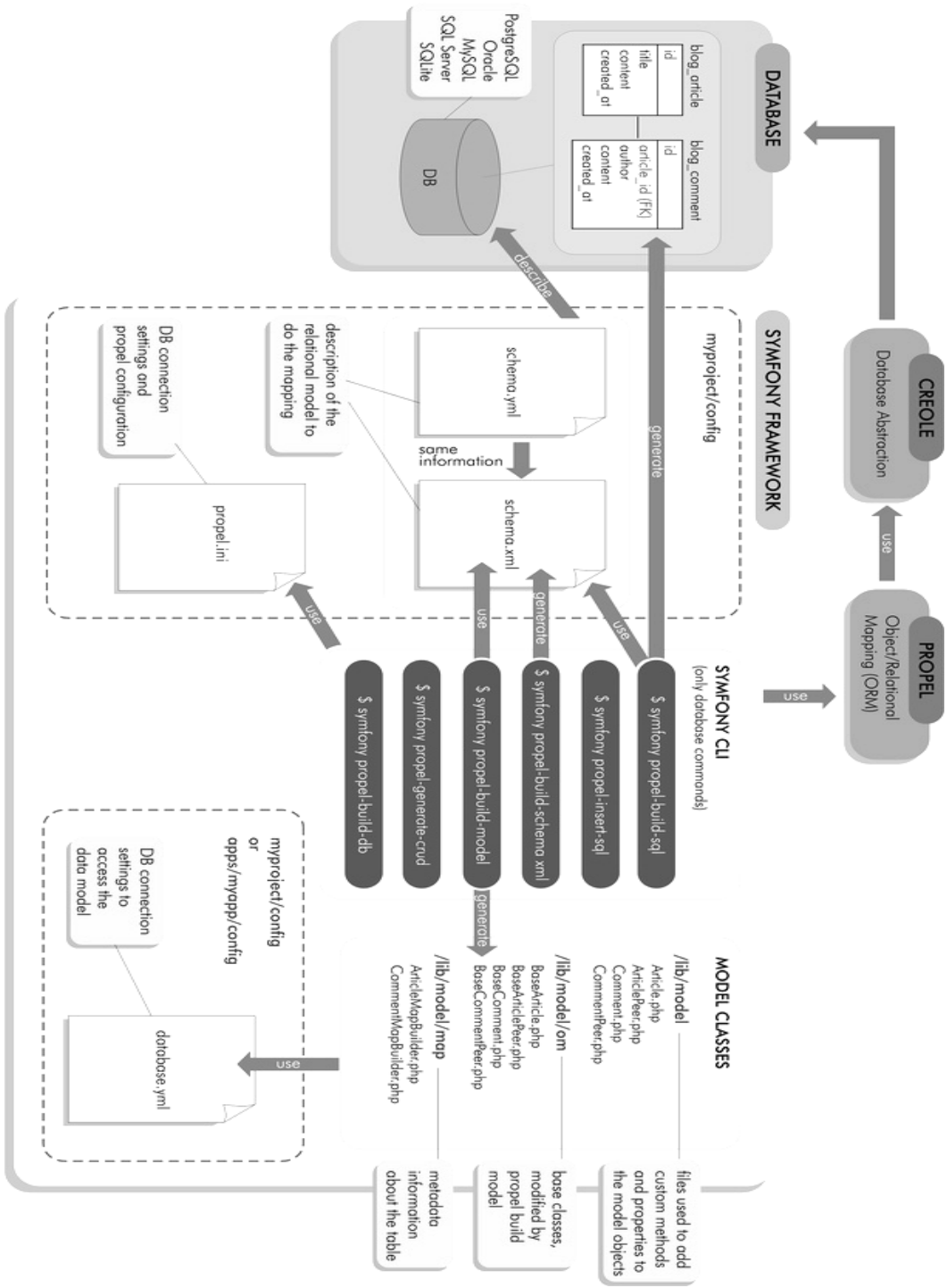
SQL: Acrónimo recursivo de *Structured Query Languaje*, lenguaje declarativo de acceso a Base de Datos relacionales.

Anexos

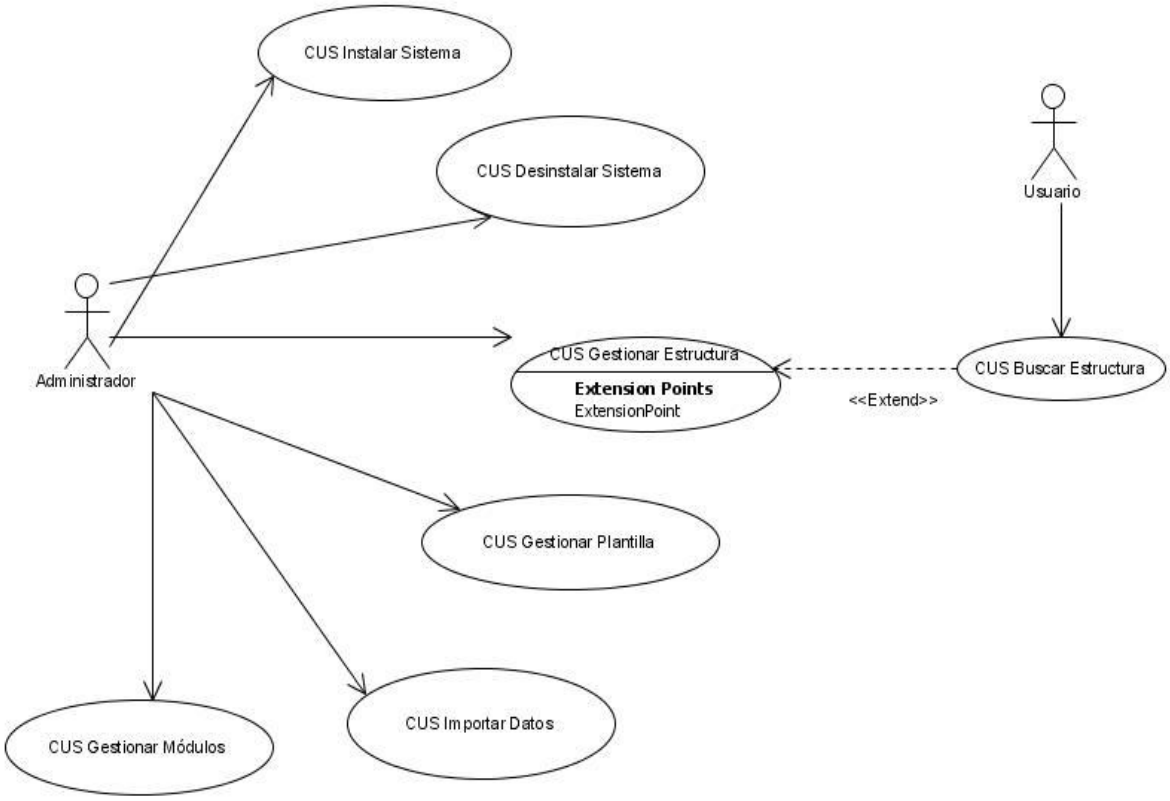
Anexo 1: Diagrama de despliegue



Anexo 2: Gestión del Modelo según Propel



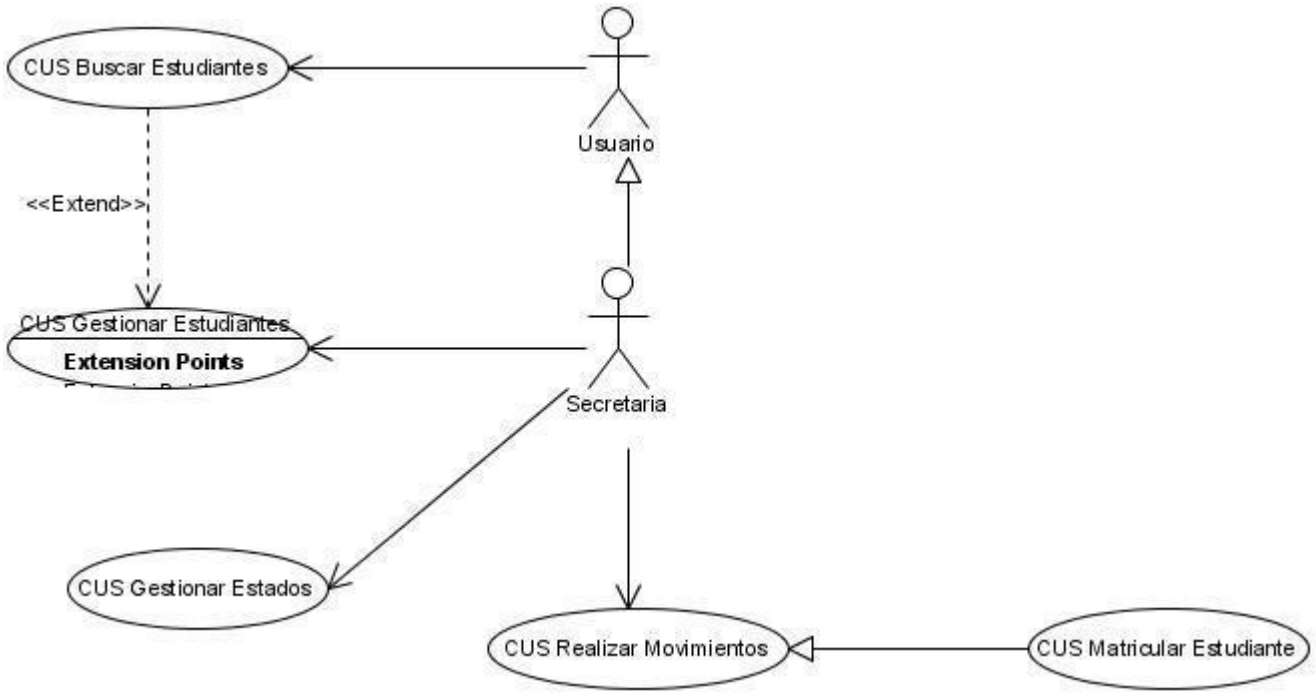
Anexo 3: Módulo Administración



Descripción Módulo Administración

Caso de Uso	Actor	Descripción	Prioridad
CUS Instalar Sistema	Usuario	Instala el sistema.	Crítico
CUS Desinstalar Sistema	Administrador	Permite desinstalar el sistema.	Crítico
CUS Gestionar Plantilla	Administrador	Crea, modifica y elimina plantillas en el sistema.	Crítico
CUS Gestionar Estructuras	Administrador	Crea, modifica y elimina estructuras en el sistema.	Crítico
CUS Importar Datos	Administrador	Importa datos de estudiantes y profesores desde contenedores de información externos al sistema.	Crítico
CUS Gestionar Módulos	Administrador	Instala, modifica y desinstala módulos en el sistema.	Crítico
CUS Buscar Estructura	Administrador	Busca una estructura en el sistema	Crítico

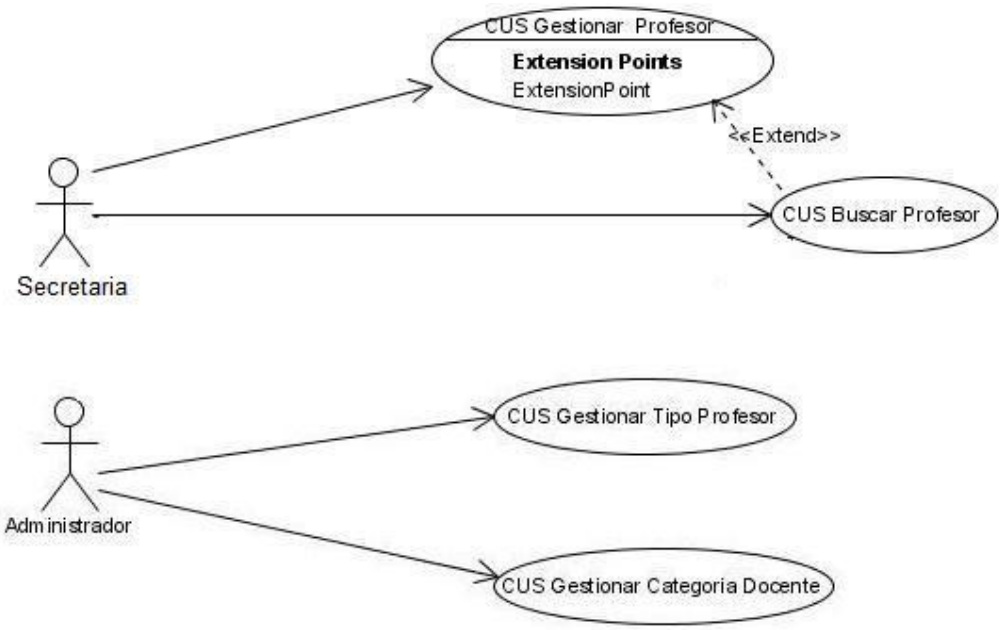
Anexo 4: Módulo Matrícula



Descripción Módulo Matrícula

Caso de Uso	Actor	Descripción	Prioridad
CUS Gestionar Estudiante	Secretaria	Adiciona y modifica un estudiante en el sistema.	Crítico
CUS Buscar Estudiante	Usuario	Permite buscar un estudiante en el sistema.	Crítico
CUS Gestionar Estado	Secretaria	Adiciona, modifica y elimina un estado de un estudiante en el sistema.	Crítico
CUS Realizar Movimientos	Secretaria	Permite realizar un movimiento sobre un estudiante según los movimientos definidos.	Crítico
CUS Matricular Estudiante	Secretaria	Permite matricular a un estudiante en el sistema.	Crítico

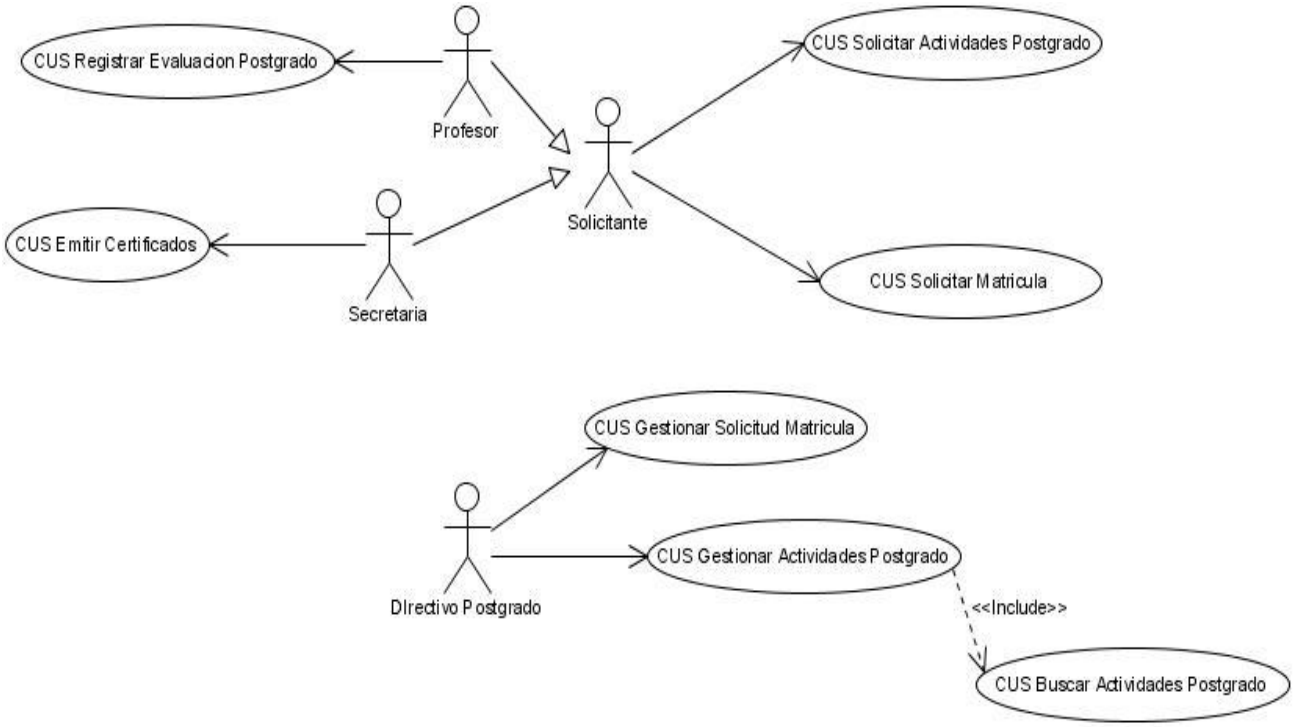
Anexo 5: Módulo Profesor



Descripción Módulo Profesor

Caso de Uso	Actor	Descripción	Prioridad
Gestionar Profesor	Secretaria	Adiciona, modifica y elimina un profesor en el sistema.	Crítico
Buscar Profesor	Secretaria	Permite buscar un profesor en el sistema.	Crítico
Gestionar Tipo Profesor	Administrador	Adiciona, modifica y elimina un tipo de profesor en el sistema.	Secundario
Gestionar Categoría Docente	Administrador	Adiciona, modifica y elimina una categoría docente en el sistema.	Secundario

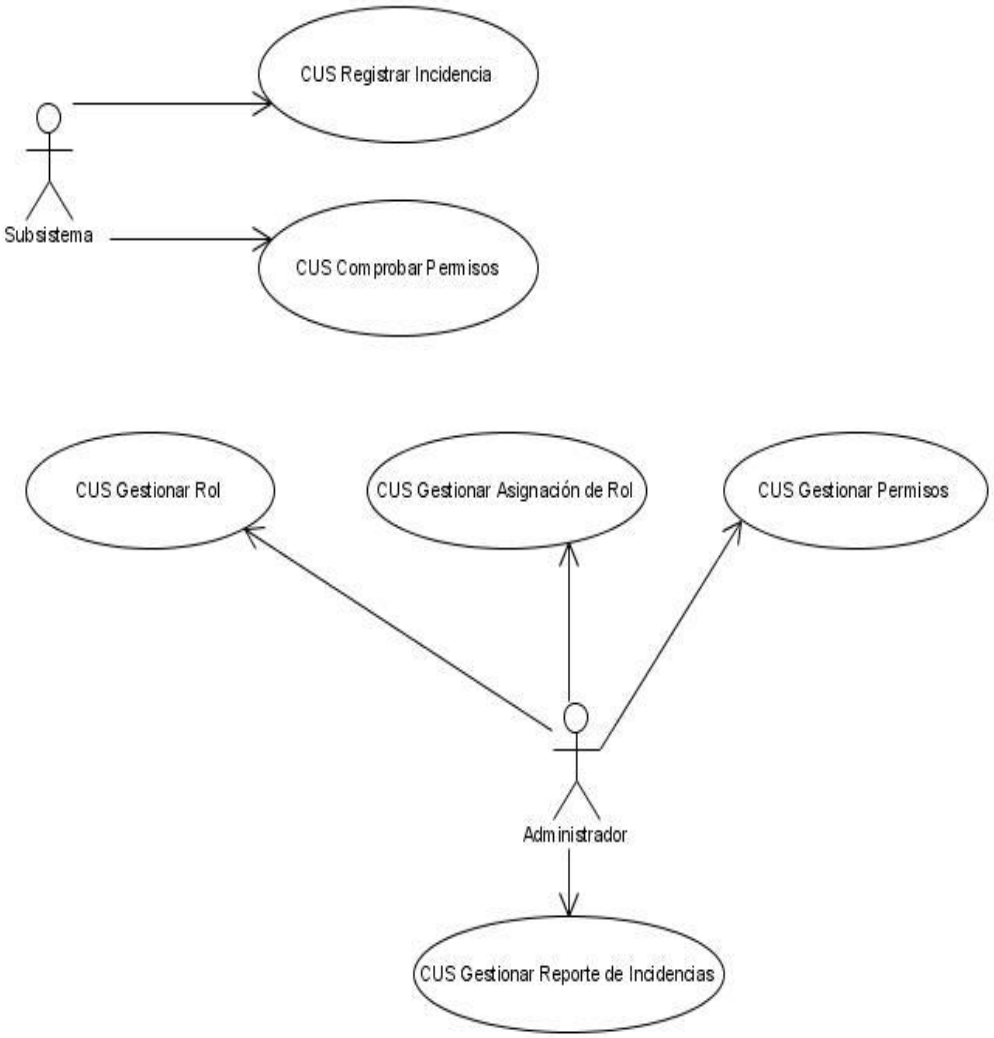
Anexo 6: Módulo Postgrado



Descripción Módulo Postgrado

Caso de Uso	Actor	Descripción	Prioridad
CUS Solicitar Matrícula	Solicitante	Permite a la secretaria o a cualquier usuario solicitar la matrícula en actividad de postgrado o superación.	Crítico
CUS Solicitar Actividad Postgrado	Solicitante	Permite a la secretaria o a cualquier usuario solicitar la realización de una actividad de postgrado o superación.	Crítico
CUS Gestionar Solicitud Matrícula	Directivo Postgrado	Permite aprobar las solicitudes de matrícula solicitadas en el sistema.	Crítico
CUS Gestionar Actividad Postgrado	Directivo Postgrado	Permite crear y modificar las actividades de postgrado.	Crítico
CUS Buscar Actividad Postgrado	Usuario	Permite buscar una actividad de postgrado en el sistema.	Secundario
CUS Emitir Certificado	Secretaria	Emite los certificados de actividades vencidas para los estudiantes y profesores.	Crítico
CUS Registrar Evaluación Postgrado	Profesor	Permite registrar las evaluaciones de los estudiantes al concluir una actividad de postgrado y adiciona los créditos.	Crítico

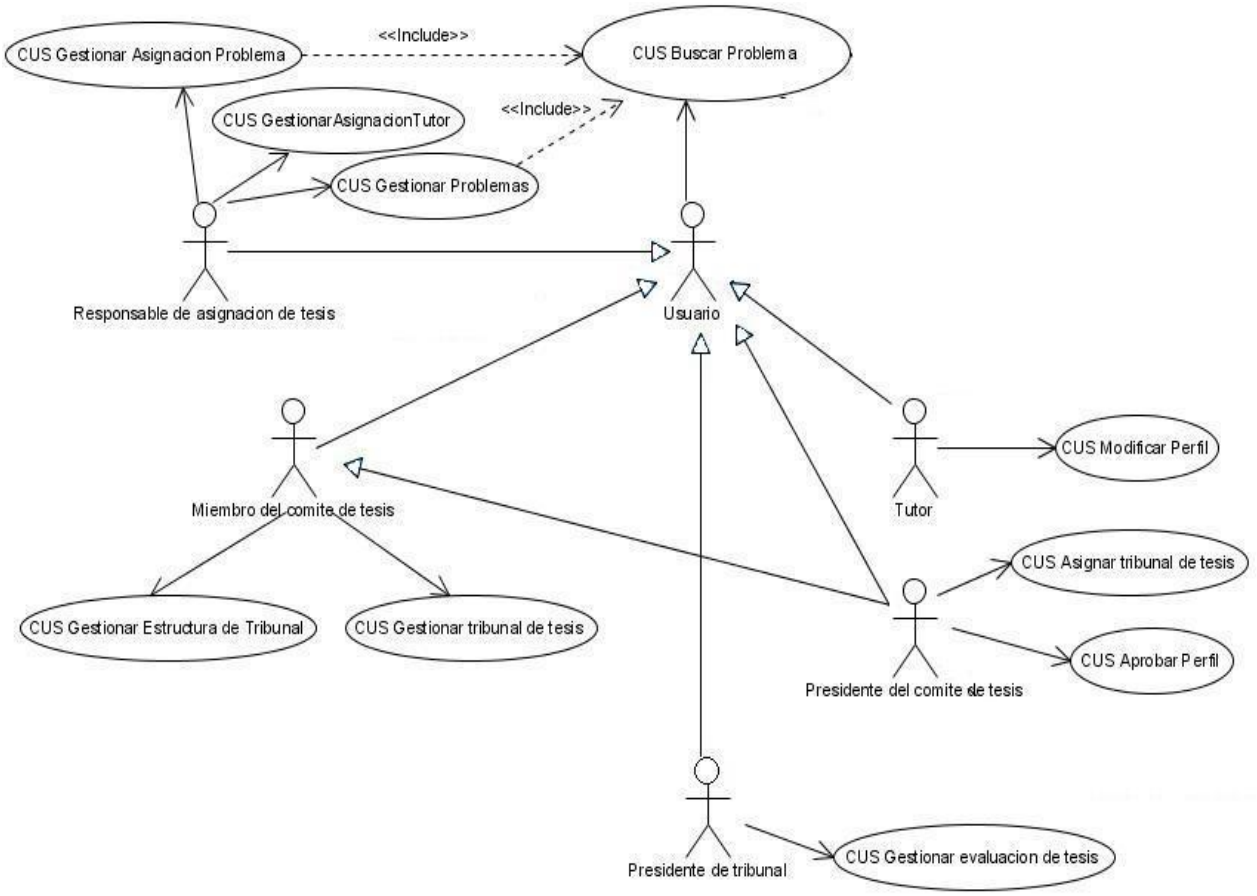
Anexo 7: Módulo Seguridad



Descripción Módulo Seguridad

Caso de Uso	Actor	Descripción	Prioridad
Gestionar Reporte de Incidencia	Administrador	Crea, modifica y elimina un reporte de incidencia en el sistema.	Crítico
Gestionar Permisos	Administrador	Crea, modifica y elimina los permisos que puede tener un rol o usuario en el sistema.	Crítico
Gestionar Rol	Administrador	Crea, modifica y elimina los permisos que pueden tener un rol o usuario en el sistema.	Crítico
Gestionar Asignación de Rol	Administrador	Asigna o elimina usuarios en los roles.	Crítico
Registrar Incidencias	Subsistema	Registra la incidencia de cualquier acción en el sistema.	Secundario
Comprobar Permisos	Subsistema	Comprueba un permiso en el sistema.	Secundario

Anexo 8: Módulo Tesis



Descripción Módulo Tesis

Caso de Uso	Actor	Descripción	Prioridad
CUS Gestionar Problema	Responsable Asignación Tesis	Adiciona, modifica y elimina un problema en el sistema.	Crítico
CUS Buscar Problema	Usuario	Permite buscar un problema en el sistema a partir de una palabra clave.	Crítico
CUS Asignar Problema	Responsable Asignación Tesis	Permite asignar un problema directamente a un estudiante.	Crítico
CUS Gestionar Tutor	Responsable Asignación Tesis	Adiciona, modifica y elimina un tutor en el sistema.	Crítico
CUS Aprobar Perfil	Presidente Comité de Tesis	Facilita que se apruebe al perfil de tesis de un estudiante.	Crítico
CUS Modificar Perfil	Tutor	Permite modificar el perfil de tesis de un estudiante.	Crítico
CUS Gestionar Estructura Tribunal	Miembro del Comité de Tesis	Permite adicionar, modificar y eliminar una estructura de tribunales de tesis.	Crítico
CUS Gestionar Tribunal de Tesis	Miembro del Comité de Tesis	Permite crear, modificar y eliminar los tribunales de tesis, según las estructuras definidas.	Crítico
CUS Asignar Tribunal de Tesis	Presidente Comité de Tesis	Asigna un tribunal a una tesis.	Crítico
CUS Gestionar Evaluación de Tesis	Presidente de Tribunal	Adiciona, modifica y elimina la evaluación del tesista.	Crítico

Anexo 9: Requerimientos no funcionales

Los requerimientos no funcionales son:

- Restricciones de diseño:
  - Lenguaje de programación será: PHP 5.0.
  - El framework base de desarrollo que se utilizará es: Symfony 1.0.11 y sub-frameworks asociados.
  - Como IDE se empleará Eclipse SDK 3.3.0 sobre JVM v_1.6.
  - Como *plug-in* para PHP se usará PDT.
  - Como servidor Web se explotará Apache 2.0.
  - El SGDB deberá ser PostgreSQL 8.1.
  - El diseño de la Base de Datos se realizará con DBDesigner 4.0.
  - El modelado UML se hará con Visual Paradigm 3.0.
  - El sistema operativo a utilizar en el entorno de desarrollo deberá ser: Windows XP SP 2 o Ubuntu 7.10.
  - El repositorio principal, el entorno de prueba y el servidor de Base de Datos estarán montados sobre Ubuntu Server 7.10.
- Hardware:

  Para el desarrollo:
  - PC con las siguientes características: Intel Pentium 4 o superior, CPU 3GHZ o superior, 512 MB RAM o superior, 160 GB HDD o superior.

Para explotación del cliente:

- PC con las siguientes características: Pentium 3 o superior, CPU 133 MHZ o superior, 128 RAM mínimo 512 RAM recomendada o superior.

Para explotación del servidor:

- PC con las siguientes características: CPU: Dual Core 2.0 GHZ o superior, RAM: 4 GB (Recomendado 6 GB), 250 GB HDD.

- Software:

  Para el cliente:

  - Sistema operativo con interfaz gráfica y conexión a red.
  - Navegador Web (Mozilla FireFox _ Recomendado).

  Para el Servidor:

  - Sistema operativo Linux del servidor: Ubuntu Server 7.10.
  - Framework Symfony 1.0.11.
  - Servidor Web: Apache 2.0.
  - Gestor de Base de Datos: PostgreSQL 8.1.
  - Software controlador de versiones: Subversion v_1.4.4 (r25188).

- Seguridad:

  - La aplicación deberá estar protegida de accesos no autorizados.
  - Se utilizará HTTPS para el envío seguro de datos.
  - Los usuarios autenticados sólo tendrán acceso a las áreas definidas para su rol.
  - Se registrarán todas las incidencias de los usuarios en el sistema.

- La información generada por el sistema será objeto de un alto nivel de protección.

- Soporte:
  - Se confeccionará un manual de usuario del sistema.
  - El sistema contará con un grupo de soporte destinado a brindar consultorías y asesoramiento técnico.

- Usabilidad:
  - Al sistema sólo tendrá acceso el personal autorizado.