Universidad de las Ciencias Informáticas

Facultad 3

Título: Sistema para la gestión de información en las secciones sindicales de la Facultad 3 de la Universidad de las Ciencias Informáticas

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

Autores:

Javier Sanamé Mena

Javier Alejandro Domínguez Prado

Tutores:

Ing. Elsydania López Guerra

Ing. Frank Lemus Paz

La Habana, julio 2014

"Año 55 de la Revolución"

Declaración de autoría

Declaramos ser autores del presente trabajo de diploma y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Autor: Javier Sanamé Mena

_____

Autor: Javier Alejandro Domínguez Prado

_____

Tutora: Ing. Elsydania López Guerra

_____

Tutor: Ing. Frank Lemus Paz

Datos de contacto

Ing. Elsydania López Guerra

Email: elguerra@uci.ci

Instructor. Se desempeña como desarrollador en el Sistema de Informatización de Tribunales.

Ing. Frank Lemus Paz

Recién graduado en adiestramiento. Se desempeña como desarrollador en el Sistema de Informatización de Fiscalías.

Agradecimientos

Queremos agradecer a todos aquellos que han contribuido de una forma u otra en el desarrollo de este trabajo, aquellos sin los cuales este momento hubiese sido solo un sueño; y en especial:

A nuestras familias, por inspirarnos a recorrer este camino, acompañarnos en cada minuto de nuestras vidas y brindarnos su apoyo y amor incondicional.

A nuestros tutores, porque sin su guía el desarrollo de este trabajo hubiese sido imposible.

A nuestros compañeros de clases, que nos han acompañado durante mucho tiempo, gracias por su ayuda, entrega y el tiempo dedicado.

A todos y a cada uno de nuestros amigos, para los cuales no alcanza la memoria, para dar con justicia créditos y méritos por todo lo que han hecho por nosotros.

En fin a todos nuestro eterno agradecimiento.

Dedicatoria

Dedico este trabajo a todas las personas que de una forma u otra me brindaron su amor y apoyo incondicional, a toda mi familia en general y en especial a mi papá, a mi mamá y a mi hermano, a los cuales les debo todo lo que soy. Gracias por estar siempre a mi lado, por haberse sacrificado tanto por mí y por haber sido mi ejemplo durante toda mi vida.

Javier Sanamé Mena.

A mis padres de quienes estoy muy orgulloso, ya que el resultado de ser hoy quien soy, es debido a ellos.

A mi hermano para que sea esto un ejemplo... (te quiero José).

A mis abuelas y abuelos por ser tan especiales y siempre estar ahí para mí.

A Lucero y a Francisco por ser mis guías y protectores, por batallar a mi lado hoy, mañana y siempre

***Javier Alejandro Domínguez Prado**.*

Resumen

La Universidad de las Ciencias Informáticas (UCI) cuenta con un gran número de trabajadores, los cuales pertenecen a diferentes secciones sindicales. En cada una de ellas se registra de forma manual información relacionada con finanzas, acuerdos, inquietudes y emulación. Esta situación provoca demoras cuando se desea obtener datos actualizados de una determinada sección. El esfuerzo realizado por las personas encargadas de esta labor es considerable, lo cual aumenta cuando una sección sindical cuenta con un número elevado de afiliados.

El objetivo del presente trabajo es desarrollar un sistema que permita agilizar la gestión de la información generada en las secciones sindicales de la Facultad 3, que es una de las facultades de la UCI. Para lograrlo se realizó un estudio sobre sistemas similares al que se desea construir, y además sobre metodologías ágiles, herramientas y tecnologías en general con el objetivo de seleccionar las más adecuadas. Además se presentan los artefactos desarrollados, propios de la metodología seleccionada. Se describe la arquitectura, los patrones empleados en la solución y el diagrama de despliegue.

Mediante la aplicación de métricas y pruebas de software se verificó la calidad de los artefactos obtenidos. Todas las deficiencias detectadas fueron corregidas satisfactoriamente. Finalmente se obtuvo una aplicación informática que posibilita solucionar la problemática planteada y cumplir el objetivo trazado.

Palabras clave: facultad, gestión de información, metodología, sección sindical.

Índice

Índice de Tablas

Índice de Figuras

Introducción

En la actualidad las Tecnologías de la Información y las Comunicaciones (TICs) evolucionan a grandes pasos, adquiriendo importancia al punto de cambiar la forma en que operan las organizaciones y empresas. Las TICs agrupan los elementos y las técnicas utilizadas en el tratamiento de la información y las comunicaciones. Han sido conceptualizadas como un medio para transmitir y gestionar datos, información y conocimiento (Quiroga, 2002). Estos pueden ser uno de los tantos factores críticos para la determinación del éxito o fracaso en los diferentes negocios, posibilitando que los procesos sean más eficientes y rápidos al establecerse en sistemas de software que ayudan a la gestión de información en dichas empresas.

Cuba ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir en la práctica social las TICs y lograr una cultura digital como una de las características imprescindibles, por lo cual se lleva a cabo un proceso de informatización en casi todas las áreas del país. Este despliegue de informatización va dirigido sobre todo por la Universidad de las Ciencias Informáticas (UCI) e influye en el nuevo proceso social del país.

La UCI cuenta con varias facultades, entre las que se encuentra la número 3, donde se forman con éxito estudiantes y se desarrollan proyectos informáticos. Las labores realizadas en la facultad son acometidas actualmente por cientos de profesionales, técnicos y personal de servicio, los cuales pertenecen a diferentes organizaciones políticas y de masas, entre las que se encuentra la Central de Trabajadores de Cuba (CTC).

En dicha organización los trabajadores deben pertenecer a una determinada sección sindical, en la cual pueden, entre otros, debatir temas de interés. En cada una de las secciones sindicales se debe gestionar información relacionada con las finanzas, la emulación, inquietudes y acuerdos. El control de esta información es llevada a cabo por los directivos de la sección sindical, entre los cuales se encuentran el secretario general y activistas de finanzas, emulación y acta.

En la actualidad, el financiero y los activistas de las secciones sindicales de la Facultad 3 registran y procesan manualmente todos los datos. Esto que provoca demoras cuando alguno de los directivos de una determinada sección sindical necesita confeccionar un informe actualizado, entre los que se encuentran por ejemplo: cuáles son los afiliados atrasados en el pago de la cuota sindical o el día de la patria, cuáles se encuentran al día en el pago, obtener los afiliados que han terminado de pagar el año completo, cuáles acuerdos fueron incumplidos, cuáles inquietudes no fueron respondidas, cuáles trabajadores son candidatos a vanguardia, cuáles han resultado alta o baja de la sección, entre otros. Esta situación se agrava cuando la cantidad de afiliados en una sección sindical es considerable, como sucede con la del Centro de Gobierno Electrónico (CEGEL) y las del Centro de Informatización de la

Gestión de Entidades (CEIGE). Además para obtener resúmenes sobre el comportamiento de las secciones sindicales de la Facultad 3 en general, es necesario emplear gran cantidad de tiempo.

Por todo lo anterior, sería de gran ayuda para los directivos de las secciones sindicales de la Facultad 3 contar con alguna herramienta que permita agilizar la búsqueda de determinada información de interés.

Para solventar los problemas anteriormente mencionados se define como **problema a resolver:**

¿Cómo agilizar la gestión de la información generada en las secciones sindicales de la Facultad 3 de la Universidad de las Ciencias Informáticas?

Como **objeto de estudio** se precisa: desarrollo de software de gestión.

Para responder al problema a resolver, se define como **objetivo general**: Desarrollar un sistema de información para las secciones sindicales de la Facultad 3 de la Universidad de las Ciencias Informáticas que permita agilizar la gestión de la información generada en las mismas.

Se asume como **campo de acción**: desarrollo de aplicaciones web para la gestión de datos sindicales.

A partir del objetivo general se derivan los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación mediante un análisis de las características de los sistemas similares, así como las principales herramientas, metodologías de desarrollo y mecanismos para desarrollar sistemas web.
2. Realizar el diseño mediante la metodología de desarrollo de software establecida, para transformar los requisitos en un diseño adecuado para la construcción del sistema.
3. Implementar los requisitos identificados mediante las herramientas y tecnologías seleccionadas para la obtención del producto.
4. Validar los resultados obtenidos para verificar su calidad y comprobar el cumplimiento del objetivo trazado.

Para dar cumplimiento a estos objetivos se proponen las siguientes **tareas de investigación**:

1. Estudio de sistemas informáticos existentes para la gestión de actividades sindicales.
2. Estudio de metodologías de desarrollo de software.
3. Estudio de herramientas de modelado.
4. Estudio de patrones de diseño y arquitectura.
5. Estudio de marcos de trabajo para el desarrollo web.
6. Estudio de lenguajes de programación para el desarrollo web.
7. Estudio de servidores de aplicación para el desarrollo web.
8. Estudio de sistemas gestores de bases de datos.

9. Estudio de métricas existentes para validar el diseño y la implementación.
10. Realización del modelo de datos.
11. Especificación de tarjetas CRC (Clase Relación y Colaboración).
12. Implementación de las historias de usuarios especificadas.
13. Descripción de los patrones de diseño y arquitectura utilizados en la propuesta.
14. Validación del diseño realizado.
15. Validación de las funcionalidades implementadas.

Como **idea a defender** para esta investigación se plantea que: Si se implementan las funcionalidades especificadas se obtendrá un sistema que permitirá agilizar la gestión de la información generada en las secciones sindicales de la Facultad 3 de la Universidad de las Ciencias Informáticas

Para desarrollar las tareas que posibiliten cumplir los objetivos trazados se utilizan métodos teóricos y empíricos.

De los **Métodos Teóricos** se emplean los siguientes:

- ✓ Analítico–Sintético: Para realizar el estudio del estado del arte de tecnologías, herramientas, metodologías para desarrollar aplicaciones web, y a partir del estudio realizado obtener los elementos significativos y arribar a conclusiones para dar solución al problema.
- ✓ Histórico–Lógico: Para conocer la evolución y desarrollo del tema tratado en la investigación, y además realizar un estudio crítico de trabajos anteriores como puntos de referencia y de comparación con los resultados alcanzados.
- ✓ Modelación: Permite la creación de modelos que representan abstracciones de la realidad. Los artefactos generados constituyen una representación visual del sistema que se va a desarrollar. Entre estos pueden encontrarse: modelo de datos, etc.

De los **Métodos Empíricos** se emplean los siguientes:

- ✓ Entrevista: Facilita la obtención de información acerca de las actividades que realizan los financieros y activistas de las secciones sindicales.
- ✓ Observación: Posibilita obtener una mejor perspectiva acerca del funcionamiento y organización de las secciones sindicales.
- ✓ Medición: Permite realizar la medición de la calidad de los artefactos que se obtienen durante el desarrollo del sistema.

El presente trabajo está estructurado en tres capítulos, tal y como se describe a continuación:

Capítulo 1: Fundamentación teórica

En este capítulo se realiza un estudio del estado del arte sobre sistemas similares al que se desea implementar, así como marcos de trabajo, patrones de diseño, lenguajes de programación, metodologías y herramientas orientadas al desarrollo web. Además se estudian los elementos fundamentales del proceso de desarrollo de software para dar solución al problema planteado.

Capítulo 2: Solución Propuesta

En este capítulo se presentan las funcionalidades que debe cumplir el sistema y además la planificación de la entrega. Además se exponen los artefactos propios del diseño, así como el modelo de datos la arquitectura del sistema y el diagrama de despliegue.

Capítulo 3: Validación de Resultados

Este capítulo tiene como objetivo fundamental aplicar métricas y pruebas de software para evaluar la calidad de los artefactos obtenidos durante el desarrollo del sistema. Se evidenciar los resultados obtenidos al evaluar la calidad de los artefactos.

.

Capítulo 1: Fundamentación teórica

En este primer capítulo se realiza un estudio sobre las características de sistemas similares al que se desea construir. Además se realiza un estudio sobre metodologías de desarrollo de software, herramientas y tecnologías en general con el objetivo de seleccionar las más convenientes para el desarrollo del sistema.

1.1 Sistemas similares existentes

A continuación se presentan las características más relevantes de algunos sistemas que han sido desarrollados para gestionar información relacionada con los sindicatos y sus trabajadores.

Sistema de Gestión de la Central de Trabajadores de Cuba (SigestCTC)

La CTC cuenta con una aplicación web que se encarga de gestionar información de cuadros, visualizar información sobre centros de trabajo y cantidad de afiliados por municipios, así como obtener datos sobre los comités, burós municipales y consejos por cada una de las provincias. También posibilita realizar reportes de bajas, gestionar datos de municipios y provincias e insertar centros de trabajo y afiliados. Fue implementado utilizando el sistema de gestión de contenidos Drupal.

Sistema de gestión y control de información para la sección sindical de la facultad 1

Este sistema gestiona datos propios de la sección sindical, así como datos de sus afiliados. Además permite registrar, visualizar y eliminar los pagos de la cuota sindical y del día de la patria de un afiliado, así como obtener reportes de los que no han realizado los pagos. Otras de las funcionalidades que posee el sistema son: obtener un reporte de los afiliados evaluados de vanguardia y los evaluados de mal, gestionar sanciones, reconocimientos, evaluaciones mensuales, trimestrales, anuales, registrar inquietudes de afiliados y sus respuestas además de publicar datos de trabajos voluntarios y actividades. Fue desarrollado en el año 2009 utilizando el sistema de gestión de contenidos Drupal.

Sitio web para la sección sindical de la facultad 8

Este sistema gestiona información de la sección sindical, además permite mostrar noticias y avisos de interés. Entre las principales funcionalidades que posee se encuentran: la gestión de evaluaciones mensuales, trimestrales y anuales de los afiliados, registro de opiniones, gestión de datos de afiliados, y el registro de pagos del día de la patria y de la cuota sindical por cada afiliado. Permite emitir reportes sobre los meses que ha pagado un afiliado, así como los que le restan por pagar. Cuenta con un foro para el debate. Fue desarrollado en el año 2009 utilizando el sistema de gestión de contenidos Drupal.

En general el sistema SigestCTC, posee funcionalidades destinadas a la gestión de datos de la organización a nivel de país, por lo que no podría ser utilizado para solucionar el problema a resolver en

el presente trabajo. Por otra parte los sistemas desarrollados para las facultades 1 y 8, poseen algunas funcionalidades afines al problema que se desea resolver.

No obstante presentan inconvenientes como que están diseñados para un entorno específico. Ninguno permite por ejemplo, gestionar acuerdos, inquietudes, secciones sindicales, configurar las escalas de pago de la cuota sindical, dar altas y bajas y registrar sus causas, gestionar trabajadores destacados y vanguardias mediante un sistema de votación, gestionar documentación propia de la facultad, generar reportes de interés relacionados con las finanzas que no solo se limiten a mostrar los meses que ha pagado un afiliado o los que le restan por pagar, entre otros. Incluso dichos sistemas no se encuentran en explotación en la actualidad y a pesar de las gestiones realizadas no se puede tener acceso al código fuente de los mismos, por lo que es necesario desarrollar un nuevo sistema.

1.2 Metodología de desarrollo de software

Para los desarrolladores de software es una tarea intensa desarrollar sistemas informáticos que cumplan con los requerimientos y las exigencias cada vez mayores de los clientes. Las metodologías para el desarrollo de software consisten en un conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar un software nuevo o modificado. Una metodología de desarrollo de software tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. Estas se pueden dividir en dos grupos de acuerdo con sus características y los objetivos que persiguen: ágiles o ligeras y pesadas o tradicionales.

Las **metodologías tradicionales** se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada (Woodcock, 2003). Sin embargo, estas no se adaptan adecuadamente a los cambios, por lo que no son adecuadas cuando se trabaja en un entorno donde los requisitos no pueden predecirse o bien pueden variar. Entre las más conocidas y utilizadas se puede citar **RUP (Proceso Unificado de Desarrollo de Software)**.

Como una posible respuesta a las distintas dificultades de las metodologías tradicionales emergen las **metodologías ágiles** que, por estar principalmente orientadas para proyectos pequeños, constituyen una solución a la medida para ese entorno, aportando una elevada simplificación y flexibilidad sin renunciar a las prácticas esenciales para asegurar la calidad del producto. Por tanto, por las características del sistema que se desea construir, resulta conveniente seleccionar una metodología ágil. Entre las más destacadas se pueden nombrar:

SCRUM

La metodología Scrum es un modelo de referencia que define un conjunto de prácticas y roles que pueden tomarse como punto de partida para el proceso de desarrollo de software. Esta metodología

está indicada para proyectos con un alto grado de cambios en los requisitos. Su principal característica es que el desarrollo de software se realiza mediante iteraciones denominadas sprint, donde cada sprint representa un incremento del producto. Además se realizan reuniones diarias para la coordinación e integración de las actividades a desarrollar. Aunque Scrum se enfoca en la gestión de procesos de desarrollo de software, puede ser utilizado en equipos de mantenimiento de software, o en una aproximación de gestión de programas (Schwaber, 2010).

Proceso Unificado Ágil

Proceso Unificado Ágil (Traducido de las siglas en inglés AUP: *Agile Unified Process)* es una versión simplificada del Proceso Unificado de Desarrollo de Software (Traducido de las siglas en inglés RUP: *Rational Unified Process)*, describiendo un enfoque simple y fácil de entender. AUP se enfoca principalmente en la gestión de riesgos, haciendo la propuesta de que aquellos elementos con alto riesgo tengan prioridad en el proceso de desarrollo y sean tratados en etapas tempranas del mismo. El AUP aplica técnicas ágiles incluyendo desarrollo orientado a pruebas, modelado y gestión de cambios ágiles y refactorización de bases de datos para mejorar la productividad (Sommerville, 2005).

Las principales características de la metodología AUP son:

- ✓ Abarca siete flujos de trabajos, cuatro de ingeniería y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente.
- ✓ El modelado agrupa los tres primeros flujos de RUP (Modelado del negocio, Requerimientos y Análisis y Diseño).
- ✓ Dispone de cuatro fases igual que RUP: Creación, Elaboración, Construcción y Transición.

AUP está basada en principios de simplicidad y agilidad lo que implica una considerable ventaja sobre otras metodologías.

Programación Extrema

Programación Extrema (Traducido de las siglas en inglés XP: *Extreme Programming)* es una metodología ligera de desarrollo de software basada en la simplicidad, la comunicación y retroalimentación o reutilización del código desarrollado. Esta metodología se centra en fortalecer las relaciones interpersonales para el éxito del desarrollo de software, promueve el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen ambiente de trabajo. XP se basa en la interacción continua entre el cliente y el equipo de desarrollo caracterizándose por poseer soluciones implementadas simples y la capacidad de afrontar los cambios. Debido a estas particularidades, XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico (Letelier, Canós, José H. & Penadés, 2003).

Una de las características fundamentales de XP que favorecerá el proceso de desarrollo es la programación en pareja, requiriéndose que dos programadores participen en un esfuerzo combinado de desarrollo en un sitio de trabajo, siendo esta una ventaja de su uso. Otras de las ventajas son la cohesión de equipo, la programación organizada, la existencia de menor taza de errores, la implementación de una forma de trabajo que se adapta a las circunstancias y que permite definir en cada iteración cuáles son los objetivos de la siguiente.

Como resultado del análisis de cada una de las metodologías se decide utilizar XP por ser la que más se adapta a las características del proyecto. Esencialmente es una metodología que resulta muy útil para proyectos no tan complejos y donde el equipo de desarrollo es pequeño. Además favorece la programación en pareja y es capaz de adaptarse a los cambios que puedan surgir en cualquier punto del ciclo de vida del proyecto, demostrando su flexibilidad. Otra de las razones por las que fue seleccionada esta metodología es porque plantea que el cliente debe formar parte del equipo de desarrollo, posibilitando así la intervención del mismo cuando sea necesario y facilitando la comprensión de las funcionalidades a implementar.

1.3 Marco de trabajo

Los *frameworks* o marcos de trabajo son estructuras de software compuestas de componentes personalizables e intercambiables para el desarrollo de una aplicación. Se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta (Gutiérrez, 2006). Para elegir el marco de trabajo a utilizar se realizó un estudio sobre algunos de los más utilizados en la actualidad.

Zend Framework 2

Zend Framework (ZF) es un marco de trabajo de código abierto que está desarrollado por Zend bajo el criterio *Open Source* para PHP 5.0. Implementa el patrón Modelo Vista Controlador (MVC) y es completamente orientado a objetos. La estructura de los componentes de ZF es algo único; cada uno está construido con una baja dependencia de los otros componentes. Esta arquitectura es de acoplamiento flexible y permite a los desarrolladores utilizar cualquier componente (Benjamín González, 2009).

Zend Framework 2 está basado en un 100% PHP 5.0, utiliza PHPUnit para probar el código y Travis CI como un Servicio de Integración continua. Entre las principales características de Zend Framework 2 destacan:

- ✓ Modular: Utiliza bloques de construcción que se pueden manejar pieza por pieza con otras aplicaciones u otros marcos de trabajo.
- ✓ Seguridad: Utiliza herramientas de codificación de cifrado y seguridad que aseguran el sistema.

- ✓ Flexible: Fácil de adaptar a las necesidades.
- ✓ Alto rendimiento: Ofrece un alto rendimiento al sistema.

Empresas como Google y Microsoft se han asociado con Zend para proporcionar interfaces de servicios web y otras tecnologías para los desarrolladores de Zend Framework. Zend sin duda es uno de los mejores, pero para poder hacer uso de sus principales ventajas se hace necesario utilizar los componentes de la biblioteca estándar de zend; sin ellos el rendimiento disminuye y las características auxiliares pueden no ser completamente funcionales lo cual dificultaría su utilización en desarrollo.

Yii 1.1.10

Yii es un acrónimo para "*Yes It Is*!" (En español: ¡Sí lo es!). Entre sus principales características se destacan que es orientado a objetos y posee un alto rendimiento basado en componentes (Winesett, 2010). Es un marco de trabajo de programación web genérica que se puede utilizar para el desarrollo de prácticamente cualquier tipo de aplicación web, es ligero y equipado con sofisticados mecanismos de caché. Como la mayoría de los marcos de PHP, Yii 1.1.10 implementa el estilo arquitectónico MVC el cuál es ideal para la programación web Yii 1.1.10 (Quiang Xue and Xiang Wei, 2008-2012).

Posee características importantes como:

- ✓ Es posible usar el código de PHP o Zend Framework en una aplicación Yii.
- ✓ Avanzado control de Temas y generador de código automático, muy útil a la hora de implementar formularios o bien operaciones básicas de Insertar, Modificar, Eliminar y Mostrar (CRUD).
- ✓ Buen manejo de errores y manejo de Seguridad (inyección SQL, ataques XSS (del inglés *Cross-Site Scripting* (Cruzar Código al Sitio)) y CSRF (Cross-Site Request Forgery), etc.).
- ✓ Posee herramientas para pruebas unitarias y funcionales basadas en PHPUnit y Selenium.

Symfony 2.4.1

Symfony 2.4.1 es un completo marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo en sistemas complejos. Además, automatiza las tareas más comunes. Symfony 2.4.1 está desarrollado completamente con PHP 5. Es compatible con la mayoría de gestores de bases de datos como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft y se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Symfony utiliza un poderoso marco de trabajo de Mapeo Objeto Relacional (ORM) llamado Doctrine, hogar de varias bibliotecas del lenguaje de programación PHP, se centra principalmente en el almacenamiento de la base de datos y el mapeo de objetos, sus componentes básicos son mapeo entre objetos y relaciones y la capa de abstracción de la base de datos (DBAL). Doctrine es muy estable, con

una base de código de alta calidad, un mapeo extremadamente flexible y de gran alcance (Potencier, 2011).

Como motor de plantillas utiliza Twig cuya principal característica radica en la herencia de plantillas. Es muy rápido y seguro, permite separar el código PHP del HTML, lo que garantiza que los usuarios pueden modificar el diseño de la plantilla, sin modificar el código. Twig es también un marco de trabajo muy flexible que permite al desarrollador definir sus propias etiquetas y filtros.

Symfony 2.4.1 cuenta con ciertas ventajas sobre los demás marcos de trabajo en la implementación de un sistema.

- ✓ Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- ✓ Está preparado para sistemas empresariales y es adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- ✓ Su código es fácil de leer y documentar, además permite un mantenimiento muy sencillo.
- ✓ Permite su integración con bibliotecas desarrolladas por terceros.
- ✓ Permite muy fácilmente utilizar la arquitectura MVC.

Se considera que el marco de trabajo más apropiado para construir el sistema informático es Symfony 2.4.1, debido a las ventajas que aporta al desarrollo. Symfony 2.4.1 integra fácilmente la arquitectura MVC optimizando el desarrollo de las aplicaciones web. Ha sido probado en numerosos proyectos y utiliza los potentes marcos de trabajo Doctrine y Twig. Contiene un amplio soporte para la seguridad de la información sobre todo contra ataques comunes como inyección SQL, XSS y CSRF, además tiene una comunidad muy grande que contribuye en la implementación del marco de trabajo. Posee gran cantidad de documentación, sus módulos son fáciles de descargar y de utilizar facilitando el proceso de desarrollo.

1.4 Lenguajes de programación

Un lenguaje de programación es un mecanismo mediante el cual es posible crear programas a través de un conjunto de instrucciones, operadores y reglas que permiten establecer la comunicación entre el programador y los dispositivos, ya sean hardware o software existentes. Para el desarrollo de aplicaciones web, estos lenguajes tienden a clasificarse de dos formas: del lado del cliente y del lado del servidor. Este último interpretado y ejecutado por el propio servidor y es enviado al cliente en un formato comprensible para él. Mientras del lado del cliente son los lenguajes que basan su procesamiento en el cliente web, es decir que se ejecutan en el navegador del usuario (Programación, 2013). A continuación se exponen los lenguajes que serán utilizados durante el desarrollo.

PHP 5

Debido a que se utilizará el marco de trabajo Symfony 2.4.1, en consecuencia se utilizará la versión **5** del lenguaje PHP (acrónimo de PHP: *Hypertext Preprocessor*). Es un lenguaje interpretado de alto nivel, especialmente diseñado para el desarrollo web y la generación dinámica de páginas web. Es un lenguaje multiplataforma, fácil de entender y de leer, soporta una gran cantidad de bases de datos. Además no requiere un mantenimiento muy seguido y es mucho más sencillo de poner al día que otros lenguajes (Vaswani, 2009).

JavaScript

Como lenguaje de programación del lado del cliente se escogió JavaScript (JS) un lenguaje creado por la empresa Netscape. Es el lenguaje de programación más utilizado en Internet para añadir interactividad a las páginas web. Un programa en JavaScript se integra en una página web (entre el código HTML) y es el navegador el que lo interpreta. Es decir, JavaScript es un lenguaje interpretado, no compilado.

HTML 5

Se utilizará además del lado del cliente, el Lenguaje de Marcas de Hipertexto **(**Traducido de las siglas en inglés HTML *HyperText Markup Language*) el cual es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web, como texto, imágenes, etc. HTML 5 permite hacer páginas más ligeras que se cargan más rápidamente, favoreciendo la usabilidad y la indexación en buscadores, además ofrece mayor compatibilidad con los navegadores.

CSS 3

Finalmente para dar estética a las vistas HTML, colores, tamaños de las fuentes y tamaños de elemento, se utilizará del lado del cliente CSS 3 u Hojas de Estilo en Cascada (Traducido de las siglas en inglés CSS*: Cascading Style Sheets*). El mismo permite indicar propiedades como el color, el tamaño de la letra, el tipo de letra, dar colores de fondo de una página, tamaños de un elemento etc. Garantiza un mayor control de la presentación.

1.5 Marcos de trabajo de JavaScript

Existe una gran cantidad de herramientas para el desarrollo de aplicaciones web, entre las cuales se encuentran los marcos de trabajo JavaScript, basados en el lenguaje JS. A continuación se presentan algunos de los más conocidos actualmente.

Prototype 1.7

Es un marco de trabajo basado en JavaScript orientado a proporcionar técnicas de desarrollo web para crear aplicaciones interactivas JavaScript asíncrono (Traducido de las siglas en inglés AJAX: *Asynchronous JavaScript And XML)* y XML, listas para ser usadas, orientadas al desarrollo sencillo y

dinámico de aplicaciones web. Permite simplificar gran parte del trabajo cuando se pretende desarrollar páginas altamente interactivas.

ExtJS 4.2

Es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, HTML Dinámico (DHTML) y DOM[1]. Esta biblioteca incluye componentes de alto desempeño y personalizables, modelo de componentes extensibles, una Interfaz de Programación de Aplicaciones (API) fácil de usar y licencias *Open Source* y comerciales. ExtJS permite tener además estos beneficios:

- ✓ Existe un balance entre Cliente-Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- ✓ Comunicación asíncrona. En este tipo de aplicación el motor de renderizado puede comunicarse con el servidor sin necesidad de estar sujeta a una acción del usuario, dándole la libertad de cargar información sin que el cliente tenga conocimiento.
- ✓ Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

ExtJS no se adapta al contexto, ya que por sus características se utilizan principalmente en proyectos complejos, además al ser aplicaciones grandes, especialmente porque cargan todo al inicio, hace que el tiempo de descarga sea mayor al de una aplicación web tradicional.

JQuery 2.1

Es una biblioteca o marco de trabajo de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web. JQuery 2.1, ofrece una serie de funcionalidades basadas en JS que de otra manera requerirían de mucho más código. Consiste en un único archivo JS, que contiene las funcionalidades comunes de DOM, eventos, efectos y funciones de AJAX (JQuery, 2014). Es un software libre y de código abierto, bajo la Licencia Pública General de GNU.

Las principales características de JQuery son:

[1] **DOM** (*Modelo de Objetos del Documento*). Permite ver el mismo documento de otra manera, describiendo el contenido del documento como un conjunto de objetos que un programa JavaScript puede actuar sobre ellos.

- ✓ Selección de elementos DOM.
- ✓ Manipulación de las hojas de estilo en cascada.
- ✓ Permite agregar efectos y funcionalidades complejas como galerías de fotos dinámicas y elegantes, validación de formularios, calendarios, etc.
- ✓ Soporta diferentes extensiones.
- ✓ Compatible con los navegadores Mozilla Firefox, Internet Explorer, Safari, Opera, entre otros.

Se considera que el marco de trabajo JS más conveniente a utilizar es JQuery2.1 debido a la utilidad que brinda para el desarrollo del sistema. Es fácil de usar y aprender, contando con una gran cantidad de documentación y se le pueden agregar fácilmente *plugins* [2]que extienden su funcionalidad, lo cual permite un ahorro de tiempo y esfuerzo para el equipo de desarrollo. Sin embargo ExtJS es más difícil de entender, menos flexible y sus características no se adaptan a proyectos simples y pequeños. JQuery funciona de igual manera en todos los navegadores lo que implica que no es necesario escribir código específico para cada navegador y además es un software libre y de código abierto, no siendo así para otros marcos como Prototype.

1.6 Servidores Web

Cualquier sitio web debe ser alojado en una aplicación de servidor web. Un servidor web es un programa que procesa cualquier aplicación del lado del servidor. Es el encargado de almacenar imágenes, textos y documentos HTML, manteniéndose a la espera de peticiones HTTP llevadas a cabo por parte de un cliente. El cliente realiza una petición al servidor y este responde con el contenido solicitado. A continuación se exponen algunos de los servidores estudiados.

Internet Information Services 8.0 (IIS)

Es un potente servidor web que ofrece una infraestructura de gran fiabilidad, capacidad de manejo y escalabilidad para aplicaciones web. El IIS en español Servicio de Información de Internet, hace posible que las organizaciones aumenten la disponibilidad de sus sitios y aplicaciones web y a la vez reducir sus costes administrativos. Es un software propietario que debe ser usado bajo licencia y tiene una serie de servicios incluidos que funcionan en computadoras que posean el sistema operativo Windows (IIS, 2014).

[2]*Plugins*: Complemento. Es una aplicación que se relaciona con otra para aportarle una nueva función.

La falta de presupuesto en el proyecto dificulta el uso de este servidor y el hecho de que el mismo solo funcione en el sistema operativo Windows lo descarta completamente, ya que este no será utilizado por la misma razón.

Apache 2.4

Apache es un servidor web basado en software libre y es de código abierto. Es flexible, rápido y potente, continuamente actualizado y adaptado a los nuevos protocolos (HTTP 1.1). Es el servidor HTTP más común y usado en todo el mundo (Apache, 2014).

Entre sus principales características se encuentran:

- ✓ Fundamentalmente se puede utilizar en una multitud de plataformas y sistemas operativos.
- ✓ Ofrece tecnología libre y de código abierto.
- ✓ Es altamente configurable y de diseño modular, capaz de ampliar su funcionalidad y calidad de servicios.
- ✓ Trabaja en conjunto con gran cantidad de lenguajes de programación interpretados como PHP, Java, Servidor de Páginas Java (Traducido de sus siglas en inglés JSP: *Java Server Pages*) y otros lenguajes de script.

Se considera que el servidor web Apache 2.4 es el más conveniente, por ser rico en características que son muy útiles para los sitios web modernos y seguir los últimos protocolos web. Su arquitectura modular se puede personalizar cuando hay necesidad de crear una configuración de servidor para las necesidades del cliente. Sus archivos de configuración son extremadamente fáciles de administrar, siendo posible editar los archivos utilizando cualquier editor de texto. Todo esto funciona a favor del proceso de desarrollo haciéndolo más rápido y simple, además aventaja a IIS en que es un servidor basado en software libre, de código abierto y multiplataforma.

1.7 Entorno Integrado de Desarrollo

Los Entornos Integrados de Desarrollo (Traducido de sus siglas en inglés IDE: *Integrated Development Environment*) son sistemas informáticos que proporcionan herramientas que facilitan el trabajo del desarrollador. Están compuestos por un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Para identificar el IDE más favorable se realizó un estudio teniendo en cuenta los principales entornos en la actualidad y las características particulares de la situación.

Eclipse 4.3

Es una plataforma universal o IDE de código abierto, multiplataforma desarrollado inicialmente por International Business Machines (IBM) y ahora por la Fundación Eclipse, organización independiente

que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios (T.E.Foundation, 2013).

La plataforma Eclipse 4.3, cuando se combina con JDT(Java Development Toolkit), ofrece características como: editor con sintaxis coloreada, compilación incremental, un depurador que tiene en cuenta los hilos a nivel fuente, un navegador de clases, un controlador de ficheros/proyectos, e interfaces para control estándar de código fuente, como Sistema de Control de Versiones (CVS) y *ClearCase*. Incluye varias características únicas, como la refactorización de código, la actualización/instalación automática de código (mediante *Update* Manager), una lista de tareas, soporte para unidades de test con JUnit, e integración con la herramienta de construcción de Jakarta: Ant.

En general Eclipse 4.3 es completamente neutral a la plataforma, tiene una mezcla ecléctica de lenguajes soportados como (Java, C/C++, Cobol Python, Eiffel, PHP, Ruby, y C#) y tiene muchas ventajas para cualquier tipo de aplicación; pero su integración con el marco de trabajo seleccionado no es tan buena como la de otros IDEs como Netbeans.

Zend Studio 9.0.1

Es un completo IDE para el lenguaje de programación PHP. Zend Studio ofrece a los desarrolladores de PHP profesionales las herramientas para escribir y mantener código PHP más rápido, resolver problemas más rápidamente y mejorar la colaboración en equipo (MacIntyre, 2008). El programa entero está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones del producto para Windows, Linux y MacOS.

Zend Studio divide las funcionalidades en dos partes, la parte del cliente y la del servidor. Las dos partes se instalan por separado, la primera contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración se necesita disponer de la parte del servidor, que instala Apache y el módulo PHP (Álvarez, 2010). Entre sus desventajas se destaca que requiere Licencia de pago, no incluye editor visual HTML y es un poco complejo y lento.

NetBeans 7.4

NetBeans es un IDE que permite de forma rápida y fácil desarrollar aplicaciones de escritorio, móviles y aplicaciones web en Java, así como aplicaciones HTML5, JavaScript y CSS3. Este IDE también proporciona un gran conjunto de herramientas para PHP. Es gratuito y de código abierto y tiene una gran comunidad de usuarios y desarrolladores de todo el mundo. Las particularidades que posee NetBeans 7.4 como IDE Open Source y multiplataforma lo convierten en uno de los IDEs más potentes

actualmente. Incluye características muy importantes como la integración con uno de los marcos de trabajo de desarrollo más utilizado en la actualidad: Symfony2. Además es uno de los IDEs más ligeros y flexibles que existen (NetBeans, 2014).

Para realizar la implementación se seleccionó como entorno integrado de desarrollo NetBeans 7.4 por su gran rendimiento y facilidad de trabajo con respecto a los demás IDEs. Además, ofrece una mejora significativa del rendimiento y completamiento de código, debido a sus nuevas capacidades de análisis de código estático en el editor Java (NetBeans, 2014). La versión también incluye características notables, como la integración con el marco de trabajo Symfony 2 y soporta lenguajes dinámicos como PHP, JavaScript y HTML 5 los cuales serán utilizados en el proceso de desarrollo. Además es soportado por los sistemas operativos principales y más utilizados como son GNU/Linux y Windows.

1.8 Sistemas Gestores de Base de Datos

Los sistemas gestores de bases de datos (Traducido de las siglas en inglés SGBD: *Data Base Management System*) permiten a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos. Proporcionan un conjunto coordinado de programas, procedimientos y lenguajes que permiten a los distintos usuarios realizar sus tareas con los datos, garantizando la seguridad de los mismos. En la actualidad existen muchos SGBD muy utilizados por sus ventajas y funcionalidades. Para identificar el gestor a utilizar se llevó a cabo un estudio sobre algunos de los más sobresalientes.

MySQL 5.5

MySQL 5.5 está distribuido bajo una licencia "*Copyleft*", es una licencia restrictiva que obliga a los vendedores de software propietario a liberar su código o adquirir una licencia propietaria de una sola entidad comercial. MySQL 5.5 destaca como el SGBD más utilizado del mundo. Es un SGBD relacional escrito en C y en C++ y licenciado bajo la GPL de sus siglas en inglés (*General Public License*) de la GNU probado con un amplio rango de compiladores diferentes. Su diseño multihilos le permite soportar una gran carga de forma muy eficiente. Posee una adecuada gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos. Entre sus principales características se conoce que:

- ✓ Funciona en diferentes plataformas y es compatible con diferentes lenguajes como C, C++, Eiffel, Java, Perl, PHP, Python, Ruby.
- ✓ Uso completo de multihilos mediante hilos del kernel. Pueden usarse fácilmente múltiples microprocesadores si están disponibles.
- ✓ Proporciona sistemas de almacenamiento transaccional y no transaccional.
- ✓ Las funciones SQL usan una biblioteca altamente optimizada y son tan rápidas como es posible.

- ✓ Puede mezclar tablas de distintas bases de datos en la misma consulta.
- ✓ Un sistema de privilegios y contraseñas muy flexibles y seguro, que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está cifrado cuando se conecta con un servidor (MySQL, 2014).

Entre las principales desventajas que afectarían el trabajo con MySQL se encuentran que no soporta transacciones, ni subconsultas y no considera las claves ajenas. Además no posee una buena escalabilidad lo que causa que no sea factible con bases de datos muy grandes donde se accede continuamente, aunque la situación problemática no precisa una base de datos con muchas tablas es muy probable que sea necesario guardar una gran cantidad de datos.

Oracle 12c

Oracle es un SGBD fabricado por Oracle Corporation que utiliza la arquitectura cliente/servidor. Oracle Database 12*c* presenta una arquitectura que facilita desplegar y gestionar nubes de bases de datos. Innovaciones como Oracle Multitenant, para la consolidación rápida de múltiples bases de datos, la optimización automática de datos con mapa de riesgo, para la compresión y la organización por capas de datos con densidad más alta, maximizan la eficiencia y la flexibilidad de los recursos. Es un sistema multiplataforma, disponible en Windows, Linux y Unix. Los programadores de aplicaciones pueden acceder directamente a tipos de objetos Oracle, sin necesidad de ninguna capa adicional entre la base de datos y la capa cliente. Soporta las características del paradigma orientado a objetos. Tiene buen rendimiento y brinda soporte a la mayoría de los lenguajes de programación (Oracle, 2014). Como desventaja se puede destacar que es un producto de elevado precio.

PostgreSQL 9.2

PostgreSQL es un SGBD relacional libre, publicado bajo una licencia de software libre permisiva. Cuenta con una arquitectura probada que se ha ganado una sólida reputación por su fiabilidad e integridad con un largo historial de desarrollo. Está disponible para una amplia variedad de plataformas y es usado desde el más pequeño de los sistemas integrados hasta enormes sistemas de terabytes. Ser de código abierto es uno de los más claros beneficios de PostgreSQL 9.2. Es bien conocido por bases de datos que pueden mantenerse en línea por prolongados períodos de tiempo (Smith, 2010).

Entre sus características principales destacan las siguientes:

- ✓ Excelente cumplimiento del estándar SQL, siguiendo el último SQL: 2011.
- ✓ Arquitectura cliente-servidor con un amplio rango de drivers y clientes.
- ✓ Diseño de alta concurrencia donde los lectores y escritores no se bloquean.
- ✓ Excelente escalabilidad y rendimiento con características de ajustes extensas.

- ✓ Optimizador de consultas sofisticado, adecuado para inteligencia de negocios.
- ✓ Soporta totalmente el acceso y procedimientos de base de datos en Java, Python, Perl, PHP y muchos más.
- ✓ Altamente confiable con características extensivas para durabilidad y alta disponibilidad.
- ✓ Tipos de datos avanzados como Información Geográfica, búsqueda completa de texto y más.

Dado el prestigio con el que cuenta PostgreSQL 9.2, se selecciona como SGBD a utilizar. En general cumple con todas las características fundamentales, integridad, seguridad y fiabilidad. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que MySQL no podría. Como cliente gráfico tiene el PgAdmin III que es un software libre distribuido junto a PostgreSQL. Además permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos y tiene la capacidad de comprobar la integridad referencial, mientras Oracle queda descartado por ser un SGBD privado y con un elevado costo.

1.9 Herramientas CASE

La Ingeniería de Software Asistida por Computadora (Traducido de las siglas en inglés CASE: *Computer Aided Software Engineering)* es considerada un conjunto de programas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Estas herramientas facilitan la creación de diagramas y el diseño de sistemas informáticos. Para la selección de la herramienta CASE a utilizar se realiza un estudio de las más relevantes en la actualidad. A continuación se exponen las características más destacadas de cada una de ellas.

Embarcadero ER/Studio 8.0

ER/Studio es una herramienta de modelado de datos, se usa para el diseño y la construcción lógica y física de bases de datos. Su ambiente es de gran alcance y multinivel. Simple y fácil al usuario, ayuda a las organizaciones en la toma de decisiones, resuelve embotellamientos de datos, elimina redundancia y alcanza usos de alta calidad que entregan datos eficientes y exactos a la empresa (Bureau, 2008).

Ofrece un entorno de modelado de datos de colaboración para administrar datos empresariales con una interfaz intuitiva y gráfica. Su principal desventaja radica en que es un software privativo.

Visual Paradigm 5.0

Visual Paradigm para UML (acrónimo de *Unified Modeling Language)* es una herramienta CASE que ayuda al equipo de desarrollo de software al modelado, la documentación del software y generar código fuente. Soporta el ciclo de vida completo del desarrollo del software (análisis y diseño orientado a objeto,

construcción, prueba y despliegue) a través de sus correspondientes diagramas. Las principales características de esta herramienta son que:

- ✓ Genera los informes para la documentación.
- ✓ Presenta un editor de detalles de casos de uso incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- ✓ Distribuye automáticamente los diagramas y reorganiza las figuras y conectores de los diagramas del lenguaje unificado de modelado.
- ✓ Permite la ingeniería inversa de bases de datos (Paradigm, 2013).

Se considera que la herramienta CASE más conveniente para asistir el desarrollo del sistema es Visual Paradigm 8.0 ya que es una herramienta multiplataforma que soporta el ciclo de vida completo del software. Visual Paradigm 8.0 posee características importantes que facilitarían la resolución de la problemática dada, como la generación de código y de scripts para crear bases de datos. Es fácil de usar y aprender, contando con una gran cantidad de documentación y el equipo de desarrollo se encuentra familiarizado con la misma.

1.10 Métricas de validación del diseño

Las métricas son medidas que se utilizan para entender mejor los atributos de los modelos y valorar la calidad de los productos de ingeniería. Las métricas o medidas de diseño determinan la calidad del diseño y guían la evolución del mismo. Ayudan en la evaluación de los modelos de análisis y de diseño, donde proporcionan una indicación de la complejidad de los procedimientos, del código fuente y del diseño de pruebas (Pressman, 2005).

Existen varios tipos de métricas de diseño como por ejemplo las métricas de diseño arquitectónico, que se concentran en las características de la arquitectura del programa, las medidas del diseño orientado a objetos que describen las características principales de los objetos del sistema, las medidas orientadas a las operaciones dentro de la clase, etc. Entre las más utilizadas se encuentran las métricas orientadas a clases. A continuación se exponen brevemente sus características:

- ✓ **Acoplamiento Entre Clases de Objetos (AECO):** Está dado por el número de relaciones de una clase con otra. A medida que aumenta el AECO, es posible que disminuya la facilidad de reutilización de una clase. Los valores elevados de AECO también complican las modificaciones en las clases y las pruebas que aseguran que esas modificaciones se han hecho, por esto la necesidad de disminuir el AECO.
- ✓ **Tamaño Operacional de Clase (TOC):** consiste en el número de atributos y métodos asignados a una clase y por tanto la responsabilidad de la clase. Conforme crece el número de operaciones

en una clase, mayor será el esfuerzo requerido para implementar y probar la clase. Además el aumento del TOC limita las posibilidades de reutilización de la clase.

- ✓ **Falta de Cohesión en los Métodos (FCM):** radica en la cantidad de métodos que acceden a uno o más de los mismos atributos. Si la FCM es alta estos pueden acoplarse entre sí mediante los atributos, por tanto lo deseable es mantener alta la cohesión o sea mantener baja la FCM.
- ✓ **Árbol de Profundidad de Herencia (APH):** consiste en la importancia de reducir la longitud máxima desde el nodo principal hasta la raíz del árbol. A medida que el APH crece las clases inferiores heredan muchos métodos, lo cual complica la implementación y dificulta el poder predecir el comportamiento de una clase (Pressman, 2005).

Un aspecto importante a tener en cuenta en la fase de evaluación de la calidad del diseño mediante las métricas, es cubrir los principales atributos de calidad de software.

- ✓ **Responsabilidad**: reside en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto de la problemática propuesta.
- ✓ **Complejidad de implementación**: grado de dificultad que tiene implementar un diseño de clases determinado.
- ✓ **Reutilización**: consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- ✓ **Acoplamiento**: está dado por el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- ✓ **Complejidad del mantenimiento**: reside en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirectamente en los costos y la planificación del proyecto.
- ✓ **Cantidad de pruebas**: radica en el número o el grado de esfuerzo para realizar las pruebas de calidad del componente diseñado. (Pressman, 2005)

Las métricas concebidas para comprobar la efectividad del diseño son las métricas de TOC y AECO. La métrica APH no se aplicará ya que el diseño del sistema no contempla varios niveles de herencia. Por otra parte existen muy pocos métodos que acceden a uno o más de los mismos atributos, por lo que tampoco se aplicará la métrica FCM. En cambio las métricas seleccionadas pueden dar una medida cuantificada de varios atributos que se necesitan conocer como responsabilidad, reutilización, complejidad de implementación, cantidad de pruebas, complejidad de mantenimiento, acoplamiento sin importar el tamaño del proyecto o la complejidad del mismo.

1.11 Pruebas de software

La incorporación de las prácticas de calidad en el desarrollo de software tiene numerosos beneficios. En cuanto al cliente, se consigue una mejora de su satisfacción, se reducen los errores en explotación y se

logra el cumplimiento de los requisitos. Desde la perspectiva de una organización, se consigue verificar que se han implantado las características que indicó el usuario, asegurar que los procesos se aplican de la forma adecuada y que esos procesos se mejoran con el tiempo (Brualla, 2005).

Los principios básicos de las pruebas de software son:

1. A todas las pruebas se les debería poder hacer un seguimiento hasta los requisitos del cliente.
2. Las pruebas deberían planificarse mucho antes que empiecen.
3. Las pruebas deberían empezar por “lo pequeño” y progresar hacia “lo grande”.
4. Para ser más eficaces, las pruebas deberían ser realizadas por un equipo independiente.

Un método de prueba es un enfoque sistemático, independiente del nivel en que se enmarque la prueba, que ayuda a encontrar buenos conjuntos de casos de prueba para detectar diferentes tipos de errores. Entre los diferentes tipos de pruebas estudiados para la selección se encuentran las siguientes.

Pruebas de caja blanca

La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba. En la prueba de caja blanca se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, los bucles y condiciones, además de examinar el estado del programa en varios puntos.

Las pruebas de caja blanca intentan garantizar que:

- ✓ Se ejecuten al menos una vez todos los caminos independientes de cada módulo.
- ✓ Se utilicen las decisiones en su parte verdadera y en su parte falsa.
- ✓ Se ejecuten todos los bucles en sus límites.
- ✓ Se utilicen todas las estructuras de datos internas (Pressman, 2005).

Para el desarrollo de la prueba de caja blanca las principales técnicas son:

- ✓ **Pruebas de la estructura de control** (Basadas en el flujo de control): Los criterios basados en el flujo de control o la estructura del programa, cubren todas las sentencias o bloques de sentencias en un programa, o combinaciones especificadas de ellas.
- ✓ **Pruebas de caminos básicos:** Técnica que define un conjunto básico de caminos de ejecución y se genera un caso de prueba para cada camino de ejecución. Permite obtener una medida de la complejidad lógica de un diseño procedimental (SWEBOK, 2004).

Pruebas de caja negra

Cuando se considera el software de computadora, la prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Este tipo de prueba examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software. Los métodos de prueba de la caja negra se centran en los requisitos funcionales de software. La prueba de la caja negra permite al ingeniero del software obtener conjuntos de condicionales de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de la caja negra intenta encontrar errores de las siguientes categorías:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en acceso a bases de datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y terminación.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

- ✓ **Partición de equivalencia:** Este método de prueba de caja negra divide el campo de entrada de un programa en clases de datos, a partir de las cuales deriva los casos de prueba. Cada una de estas clases de equivalencia representa a un conjunto de estados válidos o inválidos para las condiciones de entrada.
- ✓ **Análisis del valor límite:** Los casos de prueba que exploran condiciones límites tienen una rentabilidad mayor. Las condiciones límite son situaciones en los bordes, por arriba y por debajo de las clases de equivalencia para los valores de la entrada y de la salida (Myers G, 2004).
- ✓ **Prueba de casos de uso:** Las pruebas se pueden especificar a partir de casos de uso o escenarios del negocio. Los casos de uso describen los escenarios a través de un sistema basado en su uso probable real, por lo que los casos de prueba derivados a partir de ellos son muy útiles para encontrar defectos en los escenarios durante el uso real del sistema. Son muy útiles para diseñar pruebas de aceptación con la participación del cliente y ayudan a descubrir defectos en la integración causados por la interacción de diversos componentes (ISTQB, 2005).

Pruebas de aceptación

La metodología XP plantea la realización de pruebas de aceptación. Este tipo de pruebas intenta demostrar que el software es el que desea el cliente, así como garantizar que todos los requisitos han sido cumplidos y que el sistema ha sido aceptado por el mismo. Estas pruebas se hacen a partir de las Historias de Usuario (HU) y no por un listado de requerimientos. En las iteraciones las HU se traducen

a pruebas de aceptación y se prueba cada uno de los escenarios para probar que una HU ha sido implementada satisfactoriamente Si el software se desarrolla como un producto que usarían muchos clientes, no es práctico realizar pruebas de aceptación para cada uno. Por tanto se emplean procesos llamados **prueba alfa** y **prueba beta** para descubrir errores que solo el usuario final podría detectar.

Las **pruebas alfa** las aplican los usuarios finales en el lugar de trabajo del desarrollador. El software se utiliza en un entorno natural mientras que el desarrollador supervisa la prueba y registra los errores y problemas de uso. Se realiza en un entorno controlado.

Las **pruebas beta** se aplican en el lugar de trabajo de los usuarios finales, A diferencia de las pruebas alfa, por lo general el desarrollador no está, por lo que generalmente esta es una prueba del software en un entorno no controlado. El usuario final registra los problemas y los informa luego al desarrollador (Pressman, 2005).

Pruebas de rendimiento

Este tipo de pruebas se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados. Está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. Entre los tipos de pruebas para verificar el rendimiento de un sistema se destacan:

Las **pruebas de carga** consisten en probar el funcionamiento del software bajo condiciones extremas. Estudia la especificación del software, las funciones que debe realizar, las entradas y las salidas analizando los valores límites.

Las **pruebas de estrés** están diseñadas para enfrentar al programa a condiciones anormales, donde se ejecute de manera que demande recursos en cantidad, frecuencia y volumen. (Pressman, 2005)

Finalmente se decidió aplicar las pruebas de caja negra, específicamente la técnica de partición de equivalencia. Dicha técnica constituye una de las más efectivas ya que permite examinar los valores válidos e inválidos de las entradas existentes en el software. Como prueba de caja blanca se optó por utilizar las pruebas de Camino Básico, ya que estas además de reflejar la complejidad de cada método, permiten la comprobación continua del código y el buen funcionamiento de las funcionalidades implementadas. Se realizarán además las pruebas alfa perteneciente a las pruebas de aceptación debido a la necesidad de que el cliente certifique que el sistema es válido para él. Si el resultado es satisfactorio para el cliente, el producto se considera listo para su puesta en producción. Finalmente se realizarán las pruebas de rendimiento para comprobar el rendimiento del sistema en un entorno

determinado para la cantidad de trabajadores pertenecientes a las secciones sindicales de la Facultad 3.

1.12 Conclusiones parciales

Como consideraciones finales del presente capítulo se relacionan las siguientes:

- ✓ El estudio de los sistemas similares existentes demostró que carecen de varias funcionalidades que se desean implementar para darle solución al problema planteado. Además no se pudo acceder al código fuente de los sistemas adaptables, por lo que se concluyó que era necesario construir un nuevo sistema.
- ✓ Se demostró que la metodología ágil XP, el marco de trabajo Symfony 2.4.1, el entorno de desarrollo NetBeans 7.4, la herramienta CASE Visual Paradigm, el SGBD PostgreSQL y el servidor web Apache resultaron ser las más convenientes para el desarrollo del sistema.
- ✓ Además se realizó un estudio sobre métricas y pruebas de software, y se seleccionaron las que se serán aplicadas para verificar la calidad.

Capítulo 2: Solución propuesta

En el presente capítulo se definen las historias de usuario, las cuales representan las características que debe cumplir el sistema a desarrollar. Además se presenta la planificación de la entrega, y los restantes artefactos propios de la metodología XP: tarjetas CRC y tareas de Ingeniería. Se describe la arquitectura del sistema, los patrones de diseño utilizados y se presentan otros diagramas como el modelo de datos y el diagrama de despliegue.

2.1 Fase de exploración

En esta fase los clientes seleccionan las historias de usuarios que desean que se realicen para la primera entrega. Las historias de usuario son un conjunto de fichas que indican las funciones que debe realizar el sistema, constituyendo el mecanismo base de captura de requerimientos en XP. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Esta fase se extiende desde unas pocas semanas a varios meses dependiendo de la adaptación del equipo de desarrollo.

Como resultado del trabajo realizado durante la fase de exploración se identificaron un total de 64 historias de usuario. A continuación se muestran algunas de ellas y en el Anexo 2 se pueden visualizar otras:

Tabla 1: historia de usuario 8.

Historias de Usuario	
Número: 8	**Usuario:** Administrador
Nombre: Registrar datos de Facultad	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Javier Sanamé Mena	
Descripción: El sistema debe permitir registrar los datos de una nueva facultad.	
Observaciones:	

Tabla 2: historia de usuario 11.

Historias de Usuario	
Número: 11	**Usuario:** Administrador

Nombre: Registrar datos de sección sindical	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Javier Sanamé Mena	
Descripción: El sistema debe permitir registrar una nueva sección sindical con los siguientes datos: nombre, facultad (opcional).	
Observaciones:	

Tabla 3: historia de usuario 14.

Historias de Usuario	
Número: 14	**Usuario:** Secretario General
Nombre: Registrar datos de trabajador	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Javier A. Domínguez Prado	
Descripción: El sistema debe permitir registrar un trabajador con los siguientes datos: nombres, apellidos, carne de identidad, usuario, salario total, sección sindical donde debe pertenecer.	
Observaciones:	

Tabla 4: historia de usuario 18.

Historias de Usuario	
Número: 18	**Usuario:** Secretario General
Nombre: Dar alta a un trabajador	
Prioridad del negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Javier A. Domínguez Prado	

Descripción: El sistema debe permitir dar alta en una sección a un nuevo trabajador. Debe registrarse la fecha.
Observaciones:

2.2 Fase de planificación

El objetivo de esta fase es fijar la prioridad de cada una de las historias de usuario en correspondencia a las necesidades inmediatas y establecer cuál va a ser el contenido de la primera entrega. Los programadores estiman cuánto esfuerzo requiere cada historia de usuario y a partir de ahí se establece el cronograma, basado en los puntos de estimación. También se define cuántas versiones o entregas del sistema se deben hacer, qué historias de usuario se deben desarrollar en cada versión y el tiempo que tarda en desarrollarse y publicarse cada versión del programa.

2.2.1 Estimación de esfuerzo por HU

El cliente es el encargado de establecer la prioridad de cada historia de usuario y los programadores tienen la responsabilidad de estimar el esfuerzo necesario para dar cumplimiento a cada una de ellas.

La estimación se puede realizar utilizando alguna de las siguientes unidades:

- ✓ **Días ideales**: los días necesarios para que el equipo pueda completar un objetivo, sin considerar interrupciones. Para pasar a días reales hay que aplicar un factor de corrección que puede ir del 60 % al 70 % de dedicación real al proyecto. Asimismo, habrá que tener en cuenta un margen para imprevistos como bajas por enfermedad, etc.
- ✓ **Puntos de historia de usuario**: la complejidad que tiene cada historia de usuario. Un equipo en un proyecto determinado es capaz de completar un número semi-regular de puntos de historia cada iteración (“velocidad”) (Calderón , y otros, 2007).

Como equivalente a la medida de estimación se eligió la unidad de puntos de historias de usuario llegando a los resultados que se muestran a continuación:

Tabla 5: Puntos de estimación por historias de usuario.

No	Historias de Usuario	Puntos de Estimación
1.	Registrar datos de usuario	1
2.	Modificar datos de usuario	1
3.	Eliminar los datos de usuario	1
4.	Registrar rol	1

5.	Modificar rol	1
6.	Eliminar rol	1
7.	Autenticar usuario	1
8.	Registrar datos de facultad	1
9.	Modificar datos de facultad	1
10.	Eliminar datos de facultad	1
11.	Registrar datos de sección sindical (SS)	1
12.	Modificar datos de sección sindical	1
13.	Eliminar datos de sección sindical	1
14.	Registrar datos de trabajador	1
15.	Modificar datos de trabajador	1
16.	Eliminar datos de trabajador	1
17.	Buscar trabajador	2
18.	Dar alta a un trabajador	2
19.	Dar baja a un trabajador	2
20.	Registrar datos de acuerdo	1
21.	Modificar datos de acuerdo	1
22.	Eliminar datos de acuerdo	1
23.	Obtener reporte de acuerdos	3
24.	Registrar datos de inquietud	1
25.	Modificar datos de inquietud	1
26.	Eliminar datos de inquietud	1

27.	Registrar factor de división para calcular salario a comprometer	2
28.	Modificar factor de división para calcular salario a comprometer	1
29.	Eliminar factor de división para calcular salario a comprometer	1
30.	Registrar trimestre	1
31.	Modificar trimestre	1
32.	Eliminar trimestre	1
33.	Registrar datos del pago del día de haber	1
34.	Modificar datos del pago del día de haber	1
35.	Eliminar datos del pago del día de haber	1
36.	Registrar tipo	1
37.	Modificar tipo	1
38.	Eliminar tipo	1
39.	Registrar el mes de compromiso de pago del día de haber de un afiliado	1
40.	Registrar el mes en que un afiliado efectuó el pago del día de haber	2
41.	Registrar la cantidad de meses que ha pagado un afiliado la cuota sindical	3
42.	Obtener reporte de los afiliados que están atrasados en el pago de la cuota sindical	3
43.	Obtener un reporte de los afiliados que terminaron de pagar el año completo	3
44.	Enviar correos a los atrasados en el pago del día de haber o la cuota sindical	5
45.	Obtener un reporte de los afiliados que han causado alta en una sección sindical	2

46.	Obtener un reporte de los afiliados que han causado baja en una sección sindical	2
47.	Obtener un reporte de los afiliados que estuvieron atrasados en el pago de la cuota sindical o del día del haber en algún momento del año	2
48.	Registrar trabajadores destacados por trimestre	1
49.	Obtener un reporte con los posibles vanguardias de la sección sindical	2
50.	Registrar puntos a cada candidato vanguardia según indicadores establecidos en la sección sindical	1
51.	Obtener un reporte con los vanguardias, ordenados de mayor a menor por cantidad de puntos asignados	2
52.	Registrar indicador para evaluar vanguardias	1
53.	Modificar indicador para evaluar vanguardias	1
54.	Eliminar indicador para evaluar vanguardias	1
55.	Obtener un reporte resumen del comportamiento del pago de cada una de las secciones sindicales de la facultad	3
56.	Permitir que los afiliados puedan votar por los candidatos a vanguardia	3
57.	Mostrar información general de la organización	1
58.	Generar modelos con datos de la sección en formato PDF	5
59.	Generar gráficos de pastel teniendo en cuenta los datos de reportes.	5
60.	Generar modelo de bajas	3
61.	Obtener reporte con el estado actual del pago de la cuota sindical de todos los trabajadores de una sección	3
62.	Obtener reporte con el estado actual del pago del día del haber de todos los trabajadores de una sección	3
63.	Obtener reporte con los trabajadores que han resultado destacados en una sección	3

64.	Mostrar a cada trabajador información detallada sobre los pagos efectuados por él.	3

2.2.2 Plan de iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. Las iteraciones son fases o etapas de la implementación donde se obtienen resultados en un tiempo estimado. Una vez definidas las historias de usuario y estimado el esfuerzo propuesto para la realización de cada una de ellas, se distribuye el desarrollo del sistema en cuatro iteraciones.

Cada iteración tiene como objetivo la implementación de algunas de las HU, una vez terminada cada una de las iteraciones se contará con una versión de prueba del sistema. Al concluir la tercera iteración gran parte del problema que fue planteado en el momento que surgió la necesidad de implementar el sistema, estará resuelto. Por último en la iteración número 4 las HU son integradas con el resultado de las iteraciones antes descritas y como beneficio de esta integración se obtendrá el componente de software deseado.

Luego se estima cuánto debe demorar cada iteración teniendo en cuenta los puntos de historia, que permitirán medir la velocidad que tiene el equipo completando objetivos a lo largo de las iteraciones. Generalmente las iteraciones duran de 1 a 3 semanas aproximadamente. La duración total de las iteraciones, brinda una forma de controlar que todas las tareas se realicen en el tiempo que se dispone. Es conveniente reevaluar esta medida cada 3 o 4 iteraciones. A continuación se muestra el plan de duración de las iteraciones

Tabla 6: Plan de duración de las iteraciones.

No. Iteración	**Historias de Usuario**	**Duración Total de Iteraciones**
Iteración 1	De la HU 1 a la HU 19	20 días
Iteración 2	De la HU 20 a la HU 35	21 días
Iteración 3	De la HU 36 a la HU 50	28 días
Iteración 4	De la HU 51 a la HU 64	29 días

2.2.3 Plan de entrega

Una vez definido el plan iteraciones, se elabora el plan de entrega con el cliente, que tiene como objetivo definir el orden y la cantidad de entregas intermedias a realizar antes de la entrega final. La reunión puede repetirse en el transcurso del proyecto siempre que la velocidad del mismo cambie o surjan nuevas HU. El plan de entrega asegura el interés del cliente y le permite que se plantee incorporar nuevas funcionalidades o eliminar otras. Además contribuye a mantener un proceso de comunicación fluido con el cliente. A continuación se muestra el plan de entrega:

Tabla 7: Plan de entrega.

No Iteración	Duración	Fecha Inicio	Fecha Final
Entrega 1	20 días	15/2/2014	07/3/2014
Entrega 2	21 días	08/3/2014	29/3/2014
Entrega 3	28 días	30/3/2014	27/4/2014
Entrega 4	29 días	28/4/2014	27/5/2014

2.3 Características no funcionales del sistema

Además de las especificaciones funcionales incluidas en las HU, es necesario definir las características no funcionales que deberá cumplir el sistema. A continuación se detallan diferentes tipos de ellas.

Usabilidad:

Requisito no Funcional (RnF) 1. Interfaz de usuario.

El sistema debe ofrecer una interfaz amigable, fácil de operar. Las interfaces deben ostentar un diseño sencillo, con pocas entradas, permitiendo un balance adecuado entre funcionalidad y simplicidad de tal manera que no se haga difícil para los usuarios utilizar el sistema.

Requisitos de software:

RnF 2. Servidor

- ✓ El servidor de base de datos debe ser PostgreSQL 9.2 o superior.
- ✓ El servidor de aplicación debe poseer Apache 2.0 o superior.
- ✓ El servidor debe tener instalado un intérprete de PHP en su versión 5.3 o superior.

RnF 3. PC Cliente

- ✓ Las estaciones clientes deberán tener instalado el navegador. Se recomienda como navegador Mozilla Firefox.

Requisitos de hardware

RnF 4. Servidor

Los servidores deben poseer memoria RAM de 1 GB o superior. El disco duro debe ser de 60 GB o superior. El procesador debe ser Intel Pentium 4 a 1.80GHz de velocidad de procesamiento o superior.

RnF 5. PC Cliente

Las estaciones clientes deben poseer como mínimo 128 MB de RAM, una capacidad de 60 GB de disco duro y una velocidad de procesamiento de 1.5 GHZ.

Seguridad:

RnF 6. Restringir el acceso al sistema

Se establecerán roles para acceder al sistema, garantizando que la información almacenada solo podrá ser modificada y/o visualizada por los usuarios autorizados.

RnF 7.Almacenamiento de contraseñas

Las contraseñas no se almacenarán en texto plano o en espacios lógicos que permitan el acceso o modificación por personas no autorizadas.

Rendimiento:

RnF 8. Reducir el tiempo de respuesta de la aplicación

Los tiempos de respuesta a las peticiones deben ser inferiores a los 10 segundos.

2.4 Fase de diseño

2.4.1 Tarjetas CRC

Las tarjetas CRC (Clase, Responsabilidad y Colaboración) constituyen la columna vertebral del diseño. Su principal utilidad es dejar el enfoque procedimental e ingresar en el modelo orientado a objetos ya que cada tarjeta se convierte en un objeto, sus responsabilidades se convierten en métodos públicos y sus colaboradores en llamados a otras clases. La técnica CRC propone una forma de trabajo, preferentemente grupal, para encontrar los objetos del dominio de la aplicación. Como resultado del trabajo realizado durante la fase de diseño se obtienen las tarjetas CRC. A continuación se muestran las realizadas en el desarrollo del presente trabajo, las otras se pueden apreciar en el anexo 3:

Tabla 8: Tarjeta CRC 1.

Tarjeta CRC # 1
Datos de la clase

Nombre: AcuerdoController	
Responsabilidades	**Colaboradores**
✓ Insertar acuerdo	Acuerdo, AcuerdoType, SeccionSindical, TipoEstadoAcuerdo, Trabajador
✓ Modificar acuerdo	
✓ Eliminar acuerdo	
✓ Obtener reporte de acuerdos	

Tabla 9: Tarjeta CRC 2.

Tarjeta CRC # 2	
Datos de la clase	
Nombre: InquietudController	
Responsabilidades	**Colaboradores**
✓ Insertar inquietud	Inquietud, InquietudType, SeccionSindical
✓ Modificar inquietud	
✓ Eliminar inquietud	

Tabla 10: Tarjeta CRC 3.

Tarjeta CRC # 3	
Datos de la clase	
Nombre: SeccionSindicalController	
Responsabilidades	**Colaboradores**
✓ Insertar sección sindical	SeccionSindical, SeccionSindicalType, Facultad
✓ Modificar sección sindical	
✓ Eliminar sección sindical	
✓ Generar reporte resumen de pagos realizados en las secciones	

Tabla 11: Tarjeta CRC 4.

Tarjeta CRC # 4	
Datos de la clase	
Nombre: TrabajadorController	
Responsabilidades	**Colaboradores**
✓ Registrar Trabajador	Trabajador, TrabajadorType, Usuario
✓ Modificar Trabajador	
✓ Buscar Trabajador	

2.4.2 Modelo de datos

El modelo de datos permite describir las estructuras de los datos de la base y la forma en que se relacionan. Incluye además las restricciones de integridad, lo cual permite definir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí (Chronotech, 2013). El modelo de datos correspondiente al sistema a desarrollar consta de un total de 20 tablas, de ellas 9 nomencladores y las 11 restantes para el almacenamiento de la información manipulada por este. A continuación se muestra el modelo de datos:

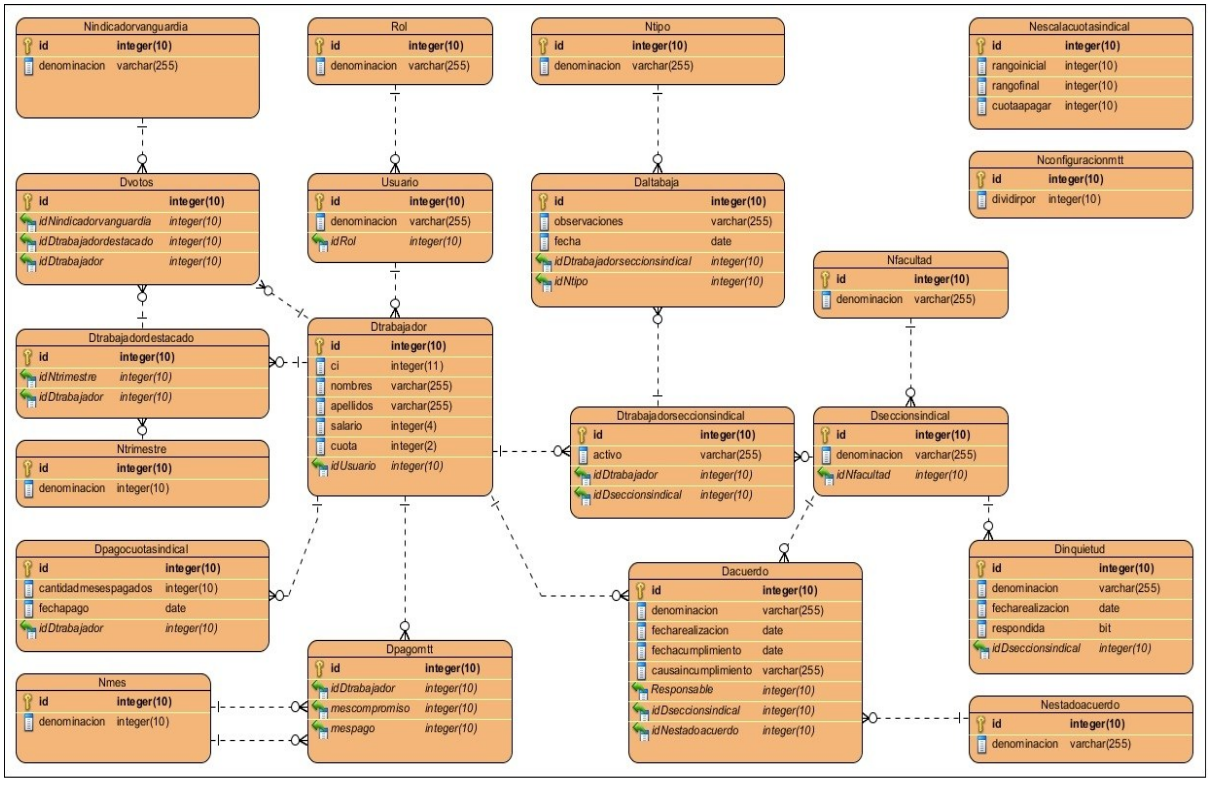

Figura 1: Representación del Modelo de Datos.

2.5 Fase de codificación

La codificación es un proceso que se realiza en forma paralela con el diseño. Es importante aplicar de forma conjunta y equilibrada determinadas prácticas que XP propone para el desarrollo de esta actividad, como son la programación en parejas, refactorización, el cliente siempre presente, seguir estándares de programación y la integración continua. En esta fase se definen las tareas de programación o ingeniería a partir de cada HU y son asignadas a los programadores.

Las tareas de ingeniería se utilizan para describir las tareas que se realizan sobre el proyecto y dan una solución a las historias de usuarios relacionadas a ellas asignándole un equipo de desarrollo o una persona para la implementación. Con ellas se pretende cumplir con las funcionalidades básicas que luego conformarán las funcionalidades generales de cada historia. Las tareas no tienen que ser entendidas necesariamente por el cliente, pueden ser escritas en lenguaje técnico, pues las mismas son usadas únicamente por los programadores.

Se obtuvieron un total de 61 tareas de ingenierías como resultado del trabajo realizado en la fase de codificación, a continuación se presentan algunas de ellas:

Tabla 12: Tarea de Ingeniería 1.

	Tarea de Ingeniería
Número Tarea: 1	**historia de usuario (Nro. 1):** Registrar datos de usuario
Nombre Tarea: Crear interfaz registrar usuario	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:** 1
Fecha Inicio: 19/2/2014	**Fecha Fin:** 20/2/2014
Programador Responsable: Javier Sanamé Mena	
Descripción: Esta tarea facilita la creación de una interfaz para la gestión de los usuarios en el sistema. Cuando el Administrador del sistema seleccione la opción Registrar usuario, aparecerá la interfaz gestionar usuario con los campos correspondientes para registrarlo en el sistema.	

Tabla 13: Tarea de Ingeniería 11.

Tarea de Ingeniería	
Número Tarea: 11	**historia de usuario (Nro. 1):** Registrar datos de sección sindical
Nombre Tarea: Crear interfaz registrar sección sindical	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:** 1
Fecha Inicio: 23/2/2014	**Fecha Fin:** 24/2/2014
Programador Responsable: Javier Sanamé Mena	
Descripción: Esta tarea facilita la creación de una interfaz para la gestión de todas las Secciones Sindicales del sistema. Cuando el Administrador del sistema seleccione la opción Registrar, aparecerá una interfaz con los campos correspondientes para registrarla en el sistema.	

Tabla 14: Tarea de Ingeniería 14.

Tarea de Ingeniería	
Número Tarea: 14	**historia de usuario (Nro. 8):** Registrar datos de Trabajador
Nombre Tarea: Crear interfaz registrar trabajador	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados:** 1
Fecha Inicio: 26/2/2014	**Fecha Fin:** 27/2/2014
Programador Responsable: Javier A. Domínguez Prado	
Descripción: Esta tarea facilita la creación de una interfaz para la gestión de los trabajadores en el sistema. Cuando el Secretario General seleccione la opción Gestionar Trabajadores, se mostrara la interfaz con los campos correspondientes para registrarlo en el sistema.	

2.6 Arquitectura de software

La arquitectura de un sistema es la estructura del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente y las relaciones entre ellos (Pressman, 2005). Representa la base para todo el funcionamiento de la aplicación; es el pilar principal del producto que se quiere construir. En su forma más simple, es la estructura u organización de los componentes del programa (módulos), la manera en que estos componentes interactúan y la estructura de datos que utilizan.

2.6.1 Modelo Vista Controlador

Los patrones arquitectónicos son patrones de diseño de software que ofrecen soluciones a problemas de arquitectura de software. Proporcionan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Expresan un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones (Pressman, 2005). A continuación se exponen algunas de las características principales del patrón arquitectónico seleccionado:

Modelo Vista Controlador es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos (Ver Figura 2).

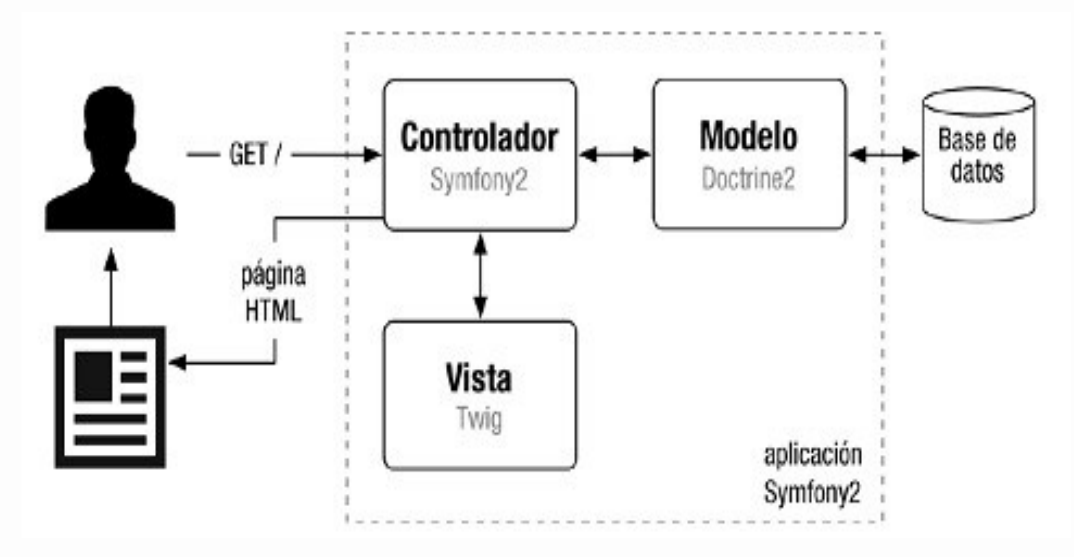

Figura 2: Patrón arquitectónico MVC.

El **Modelo** representa la parte de la aplicación que implementa la lógica de negocio. Esto significa que es responsable de la recuperación de datos, convirtiéndolo en conceptos significativos para la aplicación, así como su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos.

La **Vista** hace una presentación de los datos del modelo. Estando separada de los objetos del modelo, es responsable del uso de la información de la cual dispone para producir cualquier interfaz de presentación de cualquier petición que se presente.

El **Controlador** actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

2.6.2 Arquitectura del sistema

La arquitectura del sistema desarrollado está basada en capas. Estas se denominan: capa de presentación, capa de negocio, capa de acceso a datos y capa de datos. Es importante también destacar el uso del patrón Modelo Vista Controlador en las capas de presentación y de negocio. En la Figura 3 se puede apreciar la estructura arquitectónica en capas y la ubicación de los componentes en cada una de ellas.

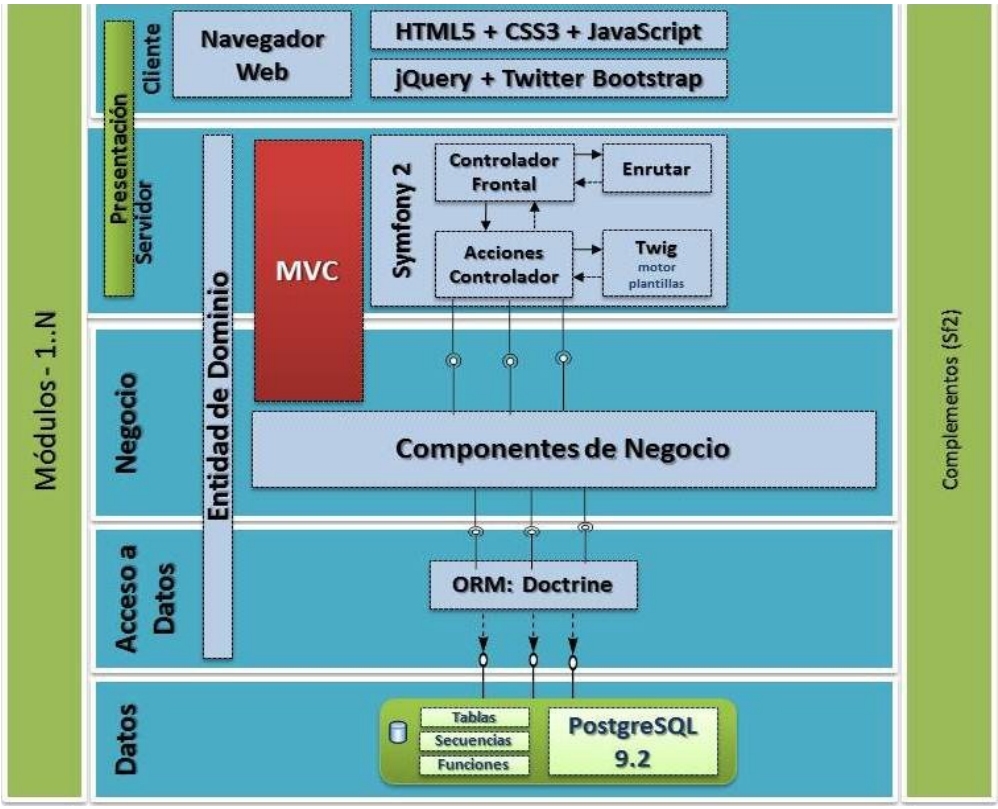

Figura 3: Arquitectura del sistema.

La arquitectura propuesta posee la característica de tener varios complementos transversales a las capas para garantizar seguridad, tratamiento de excepciones, entre otros aspectos. Las entidades del dominio son accesibles en la sub-capa servidor de la capa de presentación, la capa de negocio y la capa de acceso a datos. A continuación se detallan algunas características principales de la arquitectura definida

Capa de Presentación: Se divide en dos subcapas: cliente y servidor. En la primera se visualiza mediante el navegador web (utilizando tecnologías como HTML5, CSS3, JavaScript, Twitter- Bootstrap y JQuery). En la segunda se maneja la lógica de control así como la construcción de páginas y formularios.

Capa del Negocio: Esta capa se comunica con la de presentación para recibir las solicitudes y devolver los resultados, y con la capa de acceso a datos para solicitar la información necesaria. Contiene las clases gestoras, encargadas del manejo de la lógica de negocio.

Capa de Acceso a Datos: Esta capa contiene las clases entidad y además los repositorios donde se realizan las consultas necesarias. Gestiona las peticiones de la capa de negocio y retorna los datos que se recuperan al gestor correspondiente. Las entidades del dominio se encuentran en esta capa pero son accesibles además desde otras capas superiores.

Capa de Datos: En esta capa se localiza el gestor de base de datos PostgreSQL 9.2 y un conjunto de esquemas, tablas, vistas y procedimientos almacenados que permiten persistir la información.

Complementos: son un grupo de facilidades y mejoras que permiten lograr una arquitectura más flexible y adaptable tales como gestión de seguridad, de validaciones, de mensajería y de configuración así como tratamiento de excepciones y otras.

Bundles Permiten utilizar funcionalidades construidas por terceros o empaquetar sus propias funcionalidades para distribuirlas y reutilizarlas (Eguiluz, 2012).

2.7 Patrones de diseño utilizados

Un patrón de diseño define una posible solución general a un problema de diseño en un contexto dado. Estos utilizan un conjunto de buenas prácticas del diseño orientado a objetos para crear sistemas reutilizables y fáciles de mantener (Gamma, 1995).

Dentro de los patrones de diseño se pueden encontrar los patrones **GOF** (*Gang of Four* o Pandilla de los Cuatro) y **GRASP** (*General Responsibility Asignment Software Patterns* o Patrones Generales de Software para Asignar Responsabilidades).

Entre los patrones utilizados se encuentra el patrón GoF **Decorador**: El cual se utilizó en la generación de vistas, para responder a la necesidad de añadir dinámicamente funcionalidad a un objeto. Además, el sistema de plantillas Twig está provisto de un mecanismo de herencia que permite que la decoración de plantillas resulte de una flexibilidad y versatilidad total. El uso de este patrón se pone de manifiesto en casi todas las plantillas de la aplicación, ya que obtienen dinámicamente funcionalidades definidas en las plantillas **base.html.twig y layout.html.twig** a través de la herencia. En la Figura 4 se aprecia un ejemplo del uso del patrón decorador.

```
{% extends '::layout.html.twig' %}
{% block title %}  Trabajador {% endblock %}

{% block stylesheets %}
 {{ parent() }}
{% endblock %}

{% block body_attrs %}
{{ parent() }}
{% endblock %}

{% block body_inner %}

    <div class="row">
```

Figura 4: Patrón GoF Decorador.

Los patrones **GRASP** describen los principios fundamentales de la asignación de responsabilidades a objetos expresados, en forma de patrones (Larman, 1999). Contribuyen a resolver problemas en el diseño del sistema y encontrar la mejor solución a los mismos. A continuación se detallan los patrones GRASP utilizados y sus beneficios:

- ✓ **Experto:** Se encarga de asignar una responsabilidad a aquella clase que cuenta con la información necesaria para cumplir la responsabilidad. Los repositorios son ejemplos de clases especializadas en contener información propia. Este patrón se utiliza en múltiples clases, por ejemplo las clases entidad, repositorios y gestoras. Con la utilización de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener (Gutiérrez, 2008).
- ✓ **Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos. Este patrón se utiliza en las clases controladoras y gestoras ya que en ellas se crean objetos de las clases entidad. Brinda soporte a un bajo acoplamiento lo que supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización (Gutiérrez, 2008).
- ✓ **Alta Cohesión:** Mediante este patrón es posible asignar a cada clase las responsabilidades que le corresponden y establecer las condiciones para que colaboren con otras. Mediante este patrón mejora la claridad y la facilidad con que se entiende el diseño. Las clases generadas en el sistema poseen las responsabilidades que les corresponden.
- ✓ **Bajo Acoplamiento:** Mediante este patrón se disminuyen las dependencias entre clases. No puede verse separado de otros patrones como son el experto y la alta cohesión. Su utilización se evidencia por ejemplo en la capa de acceso a datos ya que la misma las clases tienen un alto

grado de independencia de las clases de abstracción de datos, es decir, las clases entidades no tienen asociación con la vista ni con el controlador. La poca dependencia entre esas clases permite que en la aplicación sean las entidades las de mayor reutilización. Con el uso de este patrón los componentes no se afectan por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar (Gutiérrez, 2008).

- ✓ **Controlador:** Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. En el sistema el uso de este patrón se evidencia en las clases controladoras y también en los controladores frontales de Symfony2, app.php y app_dev.php.

Inyección de dependencias

Es un patrón de diseño orientado a objetos que consiste en pasar (inyectar) a las clases todos los objetos que necesitan (dependencias) ya creados y configurados. Este patrón es implementado mediante un "contenedor". El contenedor inyecta a cada objeto los objetos necesarios según las relaciones plasmadas en un fichero de configuración. Dentro del marco de trabajo Symfony2 cada bundle proporciona un fichero de configuración llamado services.yml. Este contiene dos secciones (services y parameters). En el primero se declaran todos los parámetros de configuración y en el segundo todos los servicios que se utilizarán. Dentro del marco de trabajo Symfony2 se encuentra la clase *Symfony\Bundle\FrameworkBundle\Controller\Controller* la cual proporciona un atributo público llamado container, una instancia del contenedor de dependencias. Este permite que desde cualquier clase derivada se pueda obtener una instancia del contenedor de dependencias por lo que se puede instanciar cualquier servicio existente (Eguiluz, 2012). Como resultado, se obtiene un código más flexible, más limpio y más modular. A continuación se muestra un ejemplo donde se puede apreciar el uso de inyección de dependencias mediante el ***$this->getDoctrine()*** que constituye un atajo de ***$this->container->get('doctrine')***.

```
/**
 * Editar Acuerdo
 *
 */
public function updateAction(Request $request, $id) {
    $em = $this->getDoctrine()->getManager();
    $entity = $em->getRepository('NegocioBundle:Dacuerdo')->find($id);
    if (!$entity) {
         $this->get('session')->getFlashBag()->add('notfound',null);
          return $this->redirect($this->generateUrl('acuerdos'));
          }

    $deleteForm = $this->createDeleteForm($id);
    $editForm = $this->createEditForm($entity);
    $editForm->handleRequest($request);

   if ($editForm->isValid()) {
        $em->flush();
        $this->get('session')->getFlashBag()->add('edit',null);
        return $this->redirect($this->generateUrl('acuerdos', array('id' => $id)));
    }else
        $this->get('session')->getFlashBag()->add('error', null);
        return $this->redirect($this->generateUrl('acuerdos_edit', array('id' => $entity->getId())));

    return $this->render('NegocioBundle:Dacuerdo:edit.html.twig', array(
                'entity' => $entity,
                'edit_form' => $editForm->createView(),
                'delete_form' => $deleteForm->createView(),
    ));
}
```

Figura 5: Ejemplo de inyección de dependencias.

2.8 Modelo de despliegue

El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. Permite el mapeo de procesos dentro de los nodos, asegurando la distribución del comportamiento a través de aquellos que son representados (Pressman, 2005).

El diagrama de despliegue es un modelo de objetos que define la arquitectura física del sistema por medio de nodos interconectados. Dichos nodos son elementos de hardware sobre los cuales pueden ejecutarse los elementos de software. El diagrama de despliegue se utiliza para visualizar la distribución de los componentes de software en los nodos físicos (Wesley, 2000). A continuación se muestra una descripción de los elementos que componen el diagrama de despliegue correspondiente a la aplicación (Ver Figura 6):

- ✓ **Computadora personal (PC) cliente**: Este nodo representa la computadora utilizada por el usuario para conectarse al sistema.

- ✓ **Servidor de aplicaciones:** En este nodo se encuentra desplegado o instalado el servidor web Apache.
- ✓ **Servidor de autenticación LDAP:** Este nodo representa el servidor LDAP de la UCI utilizado para la autenticación.
- ✓ **Servidor de base de datos:** En este nodo se encuentra desplegado o instalado el servidor de base de datos con el que interactúa el sistema.

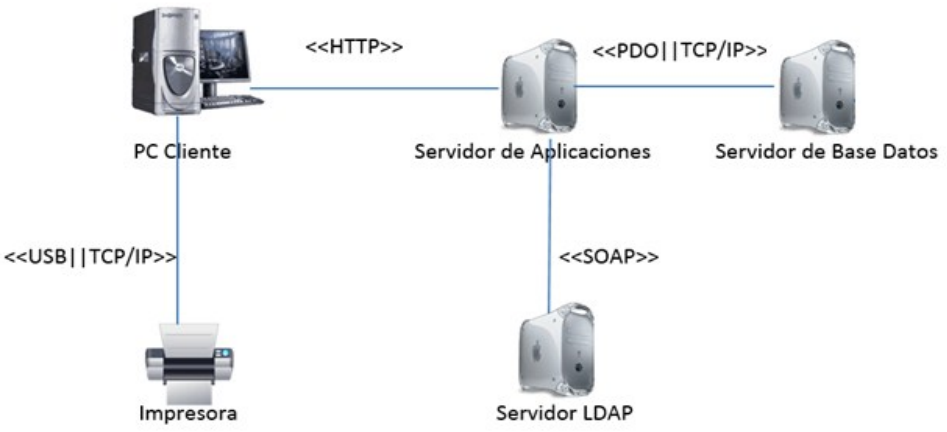

Figura 6: Diagrama de despliegue.

El proceso inicia cuando un usuario accede a la aplicación mediante una PC Cliente. Esta se conecta a su vez con el servidor web Apache a través del protocolo HTTP. Luego se verifica si el usuario y contraseña introducidos son correctos, mediante el servidor de autenticación LDAP utilizando el protocolo SOAP. Si los datos son correctos podrá acceder a la aplicación, y en caso contrario se mostrará un mensaje indicando que el usuario o contraseña son incorrectos. Con estos datos personales el servidor Apache valida que exista el usuario en el sistema. Luego se procede a atender las peticiones del mismo, comunicándose con el servidor de base de datos Este servidor se ejecuta sobre el SGBD PostgreSQL9.2, utilizando PDO y mediante el protocolo TCP/IP Este protocolo permite una comunicación fiable entre los dos servidores y mantiene la integridad de la información. Finalmente el usuario puede utilizar la impresora para imprimir algún reporte o informe que desee en formato PDF.

2.9 Conclusiones parciales

Como consideraciones finales del presente capítulo se relacionan las siguientes:

- ✓ Se generaron los artefactos propios de la metodología de desarrollo seleccionada, lo cual posibilitó realizar satisfactoriamente la implementación del sistema.
- ✓ Se realizó la planificación de las entregas y la estimación del esfuerzo por cada una de las historias de usuario a desarrollar.

- ✓ Se definió una arquitectura robusta para el sistema a desarrollar.
- ✓ Se aplicaron buenas prácticas en el diseño de la solución mediante la utilización de diferentes patrones.
- ✓ El modelo de despliegue permitió obtener una visión general de la estructura que se requiere para la posterior fase de transición cuando el producto sea implantado en el entorno real.

Capítulo 3: Validación de los resultados

En presente capítulo se utilizan las métricas de validación del diseño para evaluar su calidad y evidenciar los resultados obtenidos. También se realizan diferentes pruebas al componente de software con el objetivo de comprobar su funcionamiento y el cumplimiento de todos los requisitos.

3.1 Métrica de la calidad de la especificación

Los requerimientos sistema fueron comprobados para determinar la calidad de la especificación, a través de la métrica para la calidad de especificación de los requerimientos de software. Para comenzar a aplicar esta métrica se reunió el equipo de desarrollo junto al cliente, con el objetivo de verificar si todos interpretaban de igual forma las funcionalidades definidas. Una vez concluida esta tarea, se realizaron los cálculos necesarios para determinar el grado de ambigüedad de la especificación (Q1). Para ello se dividió el número de requerimientos para los que todos los revisores tuvieron interpretaciones idénticas (nui) entre la cantidad de requisitos de software (nr). Es válido destacar que la cantidad total de requisitos de software (nr) es igual a la suma de las cantidades de los funcionales y no funcionales.

El valor óptimo de Q1 es 1, que significa la ausencia de ambigüedad en los requisitos. Entre más cerca esté el valor de Q1 a 1 mayor será la consistencia de la especificación de los requisitos. A continuación se muestra un resumen de los resultados obtenidos:

Tabla 15: Resultados de la métrica de calidad de especificación.

Atributo de Calidad	Tipo de requerimiento	Interpretaciones	
		Iguales	Desiguales
Especificidad	Funcionales	64	0
	No Funcionales	5	3
	Total	69	3

Atendiendo a los valores antes expuestos, se realizaron las operaciones necesarias para determinar el valor de Q1. Q1 resultó ser igual a 0.958. Los requisitos ambiguos fueron reelaborados, por lo que Q1 obtuvo el valor 1, y la especificación finalmente estuvo libre de ambigüedad.

3.2 Métricas de validación del diseño

En la ingeniería de software se hace necesario comprobar si los artefactos generados en la búsqueda de una solución son factibles. Por lo tanto es importante aplicar métricas que permitan obtener resultados cuantificados que evidencien si se ha tomado el camino correcto. A continuación se exponen los

resultados obtenidos en la aplicación de la métrica Tamaño Operacional de la Clase (TOC) y Acoplamiento Entre Clases de Objetos (AECO).

El tamaño general de una clase se puede determinar empleando la cantidad de atributos de cada entidad, y la cantidad de operaciones que realiza. Luego el umbral es la suma de ambos valores y por último se realiza una estimación del tamaño de la clase a partir de los valores contenidos en la tabla 16:

Tabla 16: Relación umbral-tamaño.

Umbral	Tamaño
<=20	Pequeño
>20 y <30	Mediano
>30	Grande

Cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes en el sistema se puedan reutilizar ampliamente. Partiendo de esta relación se obtiene la valoración de cada clase. A continuación se muestra un resumen de los resultados obtenidos en la tabla 17:

Tabla 17: Valores obtenidos para cada clase en la aplicación de la métrica TOC.

No	Clases	Cantidad de atributos	Cantidad de operaciones	Umbral	Tamaño
1.	Acuerdo	7	16	23	Mediano
2.	AltaBaja	5	9	14	Pequeño
3.	Inquietud	5	10	15	Pequeño
4.	PagoCuotaSindical	4	6	10	Pequeño
5.	PagoMTT	4	5	9	Pequeño
6.	SeccionSindical	3	6	9	Pequeño
7.	Trabajador	7	19	26	Mediano
8.	TrabajadorDestacado	3	7	10	Pequeño

9.	TrabajadorSeccionSindical	4	9	13	Pequeño
10.	Votos	4	7	11	Pequeño
11.	ConfiguracionMTT	2	3	5	Pequeño
12.	EscalaCuotaSindical	4	7	11	Pequeño
13.	EstadoAcuerdo	2	7	9	Pequeño
14.	Facultad	2	7	9	Pequeño
15.	IndicadorVanguardia	2	4	6	Pequeño
16.	Mes	2	3	5	Pequeño
17.	Tipo	2	3	5	Pequeño
18.	Trimestre	2	4	6	Pequeño
19.	Rol	2	6	8	Pequeño
20.	Usuario	3	9	12	Pequeño

La gráfica refleja que el 50% de las clases contienen entre uno y diez procedimientos y solo el 10% de ellas tienen más de 20 procedimientos. Luego el Promedio del umbral para todas las clases es igual a 10.8. A partir de este se calculan la responsabilidad, complejidad y reutilización utilizando las pautas que se muestran en la tabla 18:

Tabla 18: Categorías para la métrica TOC.

	Categoría	**Criterio**
Responsabilidad y Complejidad Implementación	Baja	< =Promedio(Prom)
	Media	Entre Prom y 2* Prom
	Alta	> 2* Prom
Reutilización	Baja	> 2* Prom
	Media	Entre Prom y 2* Prom

	Alta	< =Prom

Como existen valores pequeños de TOC la mayoría de las clases posee poca responsabilidad, lo cual amplía su reutilización y facilita la implementación y la comprobación. A continuación se muestra en la gráfica los valores de reutilización obtenidos

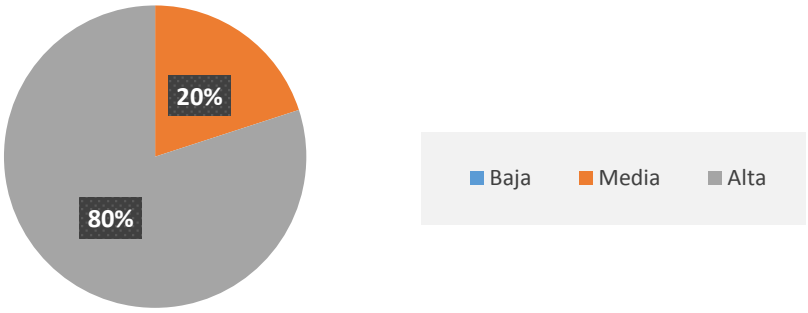

Figura 7: Porcentaje de clases agrupadas en clasificaciones según su reutilización

Se obtuvieron también valores satisfactorios asociados con la responsabilidad y complejidad. Se evidenció que el 80% de las clases presentan niveles bajos y solo un 20% medio. En la figura 8 se muestran los valoras asociados a la responsabilidad y la complejidad de implementación.

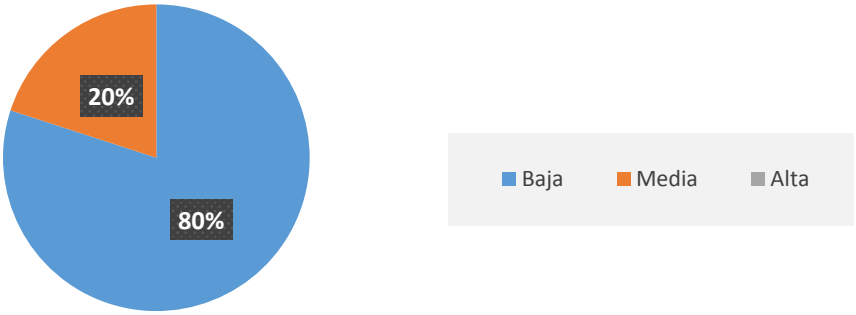

Figura 8: Porcentaje de clases agrupadas en clasificaciones según su responsabilidad y complejidad.

La métrica AECO evalúa la cantidad de relaciones de uso que existe entre las distintas clases que forman el diseño propuesto. A continuación se muestran la cantidad total de relaciones de uso de cada clase en la tabla 19:

Tabla 19: Relaciones de uso.

No	Clases	Relaciones de uso

1	Trabajador	7
2	Usuario	2
3	Acuerdo	3
4	Inquietud	1
5	Seccionsindical	4
6	Pagomtt	3
7	Pagocuotasinsical	1
8	Trabajadordestacado	4
9	Trabajadorseccionsindical	3
10	Altabaja	2
11	Votos	3
12	Rol	1
13	Trimestre	1
14	Mes	2
15	Tipo	1
16	Facultad	1
17	Estadoacuerdo	1
18	Indicadorvanguardia	1
19	Configuracionmtt	0
20	Escalacuotasindical	0

En la tabla 20 se muestran los valores implicados para medir el acoplamiento:

Tabla 20: Categorías para el acoplamiento en la métrica AECO.

Categoría	Criterio	Cantidad de clases
Ninguno	0	2

Baja	1	8
Media	2	3
Alta	> 2	7

Los resultados son positivos ya que el 50% de las clases poseen entre un bajo nivel de acoplamiento y ningún acoplamiento, mientras solo el 35% posee un alto nivel de acoplamiento en la figura 9 se puede apreciar los resultados:

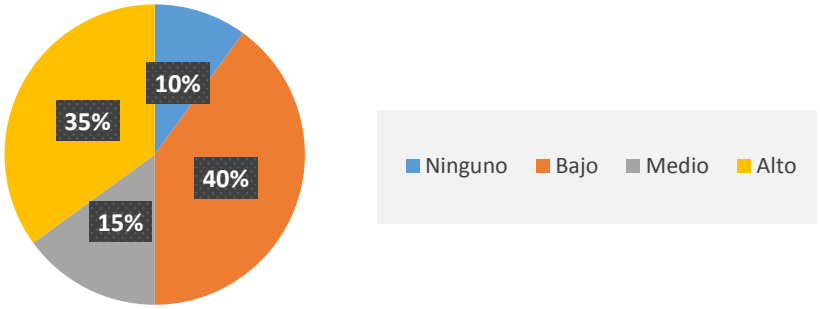

Figura 9: Porcentaje de clases agrupadas en clasificaciones según su acoplamiento.

Los otros parámetros de calidad que mide esta métrica dependen del valor promedio de las relaciones de uso de todas las clases, en este caso dicho promedio es de 2.05. Para medir la reutilización según los resultados, en la tabla 21 se plantean los siguientes valores:

Tabla 21: Categorías para la reutilización en la métrica AECO.

Categoría	Criterio	Cantidad de clases
Baja	> 2* Prom	1
Media	Entre Prom y 2* Prom	6
Alta	< =Prom	13

Utilizando las categorías antes determinadas, se obtiene que el 65% de las clases poseen un alto nivel de reutilización constituyendo datos satisfactorios en la figura 10 se muestran todos los datos obtenidos:

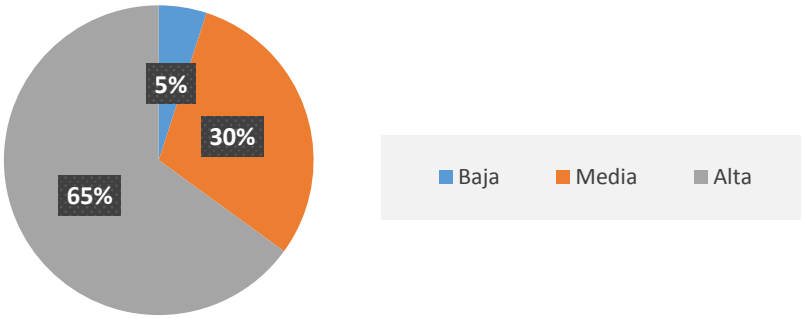

Figura 10: Porcentaje de clases agrupadas en clasificaciones según su reutilización.

La complejidad de mantenimiento y la cantidad de pruebas siguen el mismo criterio de evaluación, en la tabla 22 se exponen los datos necesarios para su evaluación:

Tabla 22: Categorías para la complejidad de mantenimiento y la cantidad de pruebas para la métrica AECO.

Categoría	Criterio	Cantidad de clases
Baja	< =Prom	13
Media	Entre Prom y 2* Prom	6
Alta	> 2* Prom	1

Se obtuvieron también valores satisfactorios asociados con los atributos complejidad de mantenimiento y cantidad de pruebas. Se evidenció que la mayoría de las clases poseen niveles bajos y medios. En la figura 11 se muestra resultados obtenidos.

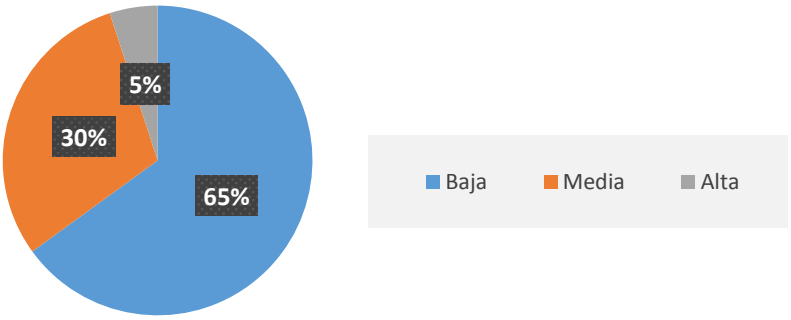

Figura 11: Porcentaje de clases agrupadas en clasificaciones según su complejidad de mantenimiento y cantidad de pruebas.

Los valores obtenidos permiten concluir que el acoplamiento es bajo, así como que existe mayor facilidad de mantenimiento de las mismas. De estos resultados se puede concluir además que, ya que la reutilización de las clases es alta o media en su mayoría, es factible el diseño realizado.

3.3 Pruebas de software

3.3.1 Pruebas del camino básico

La prueba del camino básico es una técnica de prueba de la caja blanca propuesta por Tom McCabe. Permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado, el cual describe un flujo de control lógico, donde cada círculo llamado nodo representa una o más instrucciones procedimentales. Las flechas llamadas aristas o enlaces representan el flujo de control y son análogos a las flechas de los diagramas de flujo.
2. Se calcula la complejidad ciclomática del grafo que se basa en la teoría gráfica y da un límite superior para el número de pruebas que se calcula de las siguientes maneras:
   - ✓ El número de regiones corresponde a la complejidad ciclomática V (G).
   - ✓ V (G) se define como E-N+2 donde E es el número de arista y N la cantidad de nodos de la gráfica.
   - ✓ También se define como V (G)=P+1 donde P es el número de nodos predicados incluidos en la gráfica de flujos.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.
5. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (Pressman, 2005).

A continuación se muestra un ejemplo de realización de la prueba camino básico al método **atrasadosCSAction** de la clase controladora ReportesController, el cual debe retornar los trabajadores atrasados en el pago de la cuota sindical en una sección sindical determinada en el anexo 4 se pueden ver otras de las pruebas de camino básico realizadas:

```
public function atrasadosCSAction() {

$idSS =  $this->get('security.context')->getToken()->getUser()->getSeccion();
$em = $this->getDoctrine()->getEntityManager();
$trabajadoresSS = $em->getRepository('NegocioBundle:Dtrabajador')->obtenerAfiliadosSeccion($idSS);

$SS = $em->getRepository('NegocioBundle:Dseccionsindical')->find($idSeccionSindical);
$datos['nombreSS'] = $SS->getDenominacion();
$hoy = time();
$datos['fecha'] = date("d/m/Y", $hoy);
$fechaActual = time();
$mesActual = date("m", $fechaActual);

$i = 0;
$atrasados = array();
foreach ($trabajadoresSS as $dtrabajador) {
    $idTrab = $dtrabajador->getId();
$cuotasTrabajador = $em->getRepository('NegocioBundle:Dpagocuotasindical')->findBy(array('dtrabajador' => $idTrab));
if (empty($cuotasTrabajador) && $mesActual > 1) {
        $atrasados[$i]['NombreCompleto'] = $dtrabajador->getNombreCompleto();
        $atrasados[$i]['cuotaapagar'] = $dtrabajador->getCuotaapagar();
        $atrasados[$i]['mesesPagados'] = 'Ninguno';
        $atrasados[$i]['mesesAtraso'] = '12';
} else {
       $cantMesesPagados = 0;
       foreach ($cuotasTrabajador as $cuota) {
       $cantMesesPagados = $cantMesesPagados + $cuota->getCantidadmesespagados();
}if ($mesActual > $cantMesesPagados + 1) {
            $atrasados[$i]['NombreCompleto'] = $dtrabajador->getNombreCompleto();
            $atrasados[$i]['cuotaapagar'] = $dtrabajador->getCuotaapagar();
            $atrasados[$i]['mesesPagados'] = $cantMesesPagados;
            $atrasados[$i]['mesesAtraso'] = $mesActual - $cantMesesPagados;
}}$i++;
}return $this->render('NegocioBundle:Reportes:reporteAtrasadosCS.html.twig', array('entities' => $atrasados, 'datos' => $datos));
}
```

Figura 12: Código del método AtrasadosCSAction.

El grafo de flujo resultante se presenta en la figura 13:

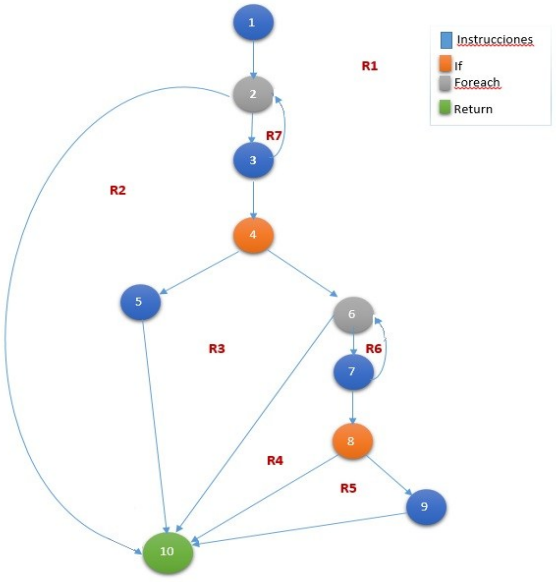

Figura 13: Grafo de flujo asociado al método AtrasadosCSAction.

Luego se calcula la complejidad ciclomática.

1. V (G)=7

2. V(G)= A–N+2= 15-10+2=7
3. V(G)=P+1=7

Luego se obtiene como conjunto básico de caminos independientes:

Camino 1: 1-2-10

Camino 2: 1-2-3-4-5-10

Camino 3: 1-2-3-4-6-10

Camino 4: 1-2-3-4-6-7-8-10

Camino 5: 1-2-3-4-6-7-8-9-10

Camino 6: 1-2-3-2-10

Camino 7: 1 -2-3-4-6-7-6-10

Para cada camino independiente se realiza un caso de prueba. En este caso se determinaron siete caminos, por tanto se debe realizar el mismo número de casos de prueba. A continuación se muestran ejemplos de los casos de prueba realizados:

Tabla 23: Caso de prueba camino 1.

Entrada	Sección sindical del financiero que realiza la petición.
Resultados esperados	Se muestra satisfactoriamente la vista la tabla con los trabajadores atrasados.
Condiciones	La sección sindical debe al menos poseer un trabajador que no tenga registrado ningún pago. Además se debe probar en un mes distinto a enero.

Tabla 24: Caso de Prueba Camino 2.

Entrada	Sección sindical del financiero que realiza la petición.
Resultados esperados	Se muestra satisfactoriamente la vista la tabla con los trabajadores atrasados.

Condiciones	La sección sindical debe al menos poseer un trabajador que tenga registrado 1 o varios pagos. La cantidad total de meses pagados más 1 debe ser inferior al mes actual.

Los casos de pruebas fueron realizados, y se detectaron deficiencias que fueron corregidas. Este tipo de prueba se realizó sobre otros métodos importantes, obteniéndose siempre los caminos y en consecuencia los casos de prueba asociados. La aplicación de estos casos de pruebas también arrojó varias no conformidades, las cuales fueron totalmente resueltas.

3.3.2 Pruebas de evaluación dinámica y pruebas de funcionalidad

Para la realización de esta prueba se realizaron los diseños de casos de prueba que se basan en la evaluación de las clases de equivalencia para las condiciones de entrada. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entradas. Por lo general una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana. Estas pruebas se llevaron a cabo en tres iteraciones, en caso de no eliminar todos los problemas del sistema se debe continuar iterando hasta que no se encuentre ninguna no conformidad. En la figura 23 se muestra un resumen de las no conformidades detectadas, y las recomendaciones realizadas en cada iteración.

Tabla 25: Resultados de las pruebas de evaluación dinámica y funcionalidad.

Iteración	No conformidades	Recomendaciones
1	48	10
2	45	6
3	6	1

En la Figura 14 se puede apreciar un desglose de las deficiencias detectas en significativas, no significativas y recomendaciones:

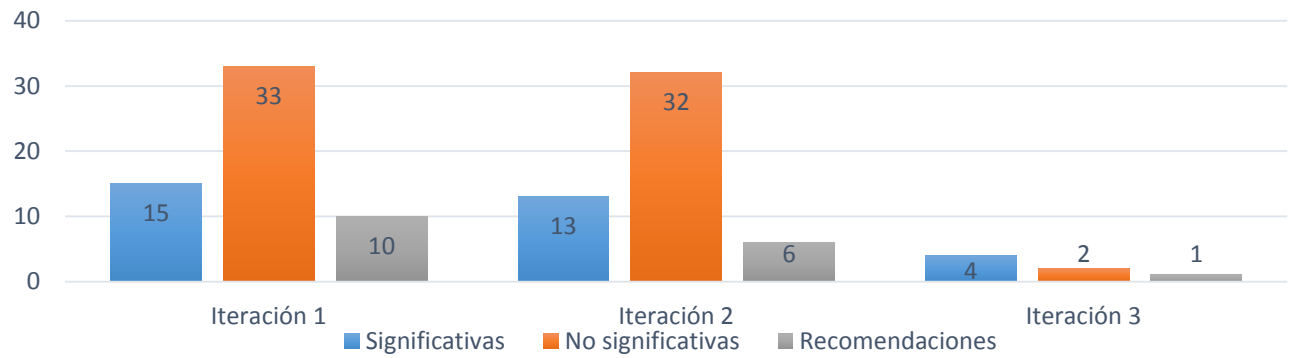

Figura 14: Resultados de las pruebas de evaluación dinámica y funcionalidad.

Las deficiencias detectadas en cada una de las iteraciones fueron corregidas satisfactoriamente. Finalmente se hizo entrega de un acta de liberación del producto (Ver anexo 1).

3.3.3 Pruebas de aceptación

Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a testear cuando una historia de usuario ha sido correctamente implementada. A las pruebas de aceptación también se las conoce con el nombre de pruebas de funcionalidad, y constituyen la garantía de que los requerimientos fijados por los usuarios han sido reflejados en el sistema (Sommerville, 2005).

Se realizaron un total de cuatro iteraciones, donde se detectaron varias deficiencias, las cuales se resumen en la tabla 24.

Tabla 26: Resultados de las pruebas de aceptación.

Iteración	No conformidades Significativas	No significativas
1	15	10
2	9	11
3	5	2
4	2	0

Las no conformidades detectadas fueron corregidas, por lo que el sistema finalmente satisface los requisitos definidos por el cliente La gráfica que a continuación se presenta, muestra los resultados anteriormente mencionados:

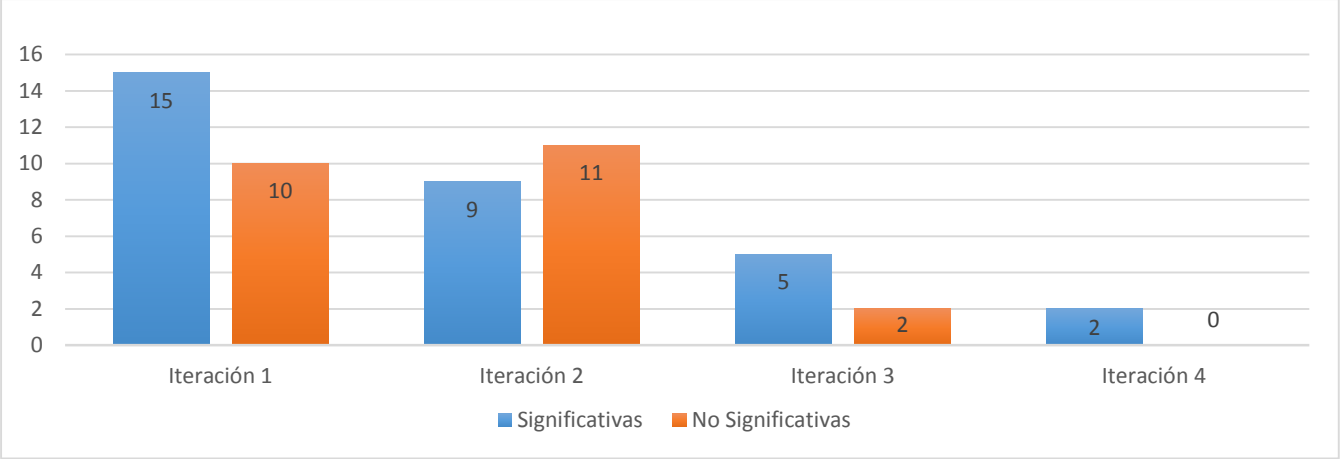

Figura 15: Resultados de las pruebas de aceptación.

A continuación se muestra un ejemplo de los Casos de Prueba de Aceptación (CPA) obtenidos, los otros se pueden ver en el anexo 5:

Tabla 27: Caso de prueba de aceptación asociado a la historia de usuario Autenticar usuario.

Caso de Prueba de Aceptación
HU5: Autenticar usuario.
Nombre: Autenticar usuario.
Descripción: El sistema debe permitir que un usuario se autentique mediante el servicio web LDAP y que acceda a las funcionalidades definidas para el rol.
Condiciones de ejecución: El usuario debe introducir su nombre y su contraseña correctamente.
Entrada/Pasos ejecución: ✓ Autenticación ✓ Autorización
Resultado Esperado: Autenticación correcta.
Evaluación de la prueba: Satisfactoria.

Tabla 28: Caso de prueba de aceptación asociado a la historia de usuario Registrar datos de sección sindical.

Caso de Prueba de Aceptación

HU9: Registrar datos de sección sindical.
Nombre: Registrar datos de sección sindical.
Descripción: El sistema debe permitir registrar una nueva sección sindical con los siguientes datos: Nombre, Facultad (opcional). Esta funcionalidad debe ser accesible solo para el Administrador del Sistema.
Condiciones de ejecución: El administrador debe introducir correctamente los datos solicitados.
Entrada/Pasos ejecución: ✓ Si los datos introducidos son incorrectos se muestra un mensaje de error ✓ Si los datos son correctos se registra una nueva sección sindical y se muestra un mensaje de confirmación.
Resultado Esperado: Se registra la sección sindical satisfactoriamente.
Evaluación de la prueba: Satisfactoria.

3.3.4 Pruebas de carga y estrés

El desarrollo de las pruebas de rendimiento se realizó utilizando una herramienta de prueba automática para simular un gran número de usuarios, carga y volumen de información. La aplicación propuesta para esto es la herramienta JMeter versión 2.9, teniendo en cuenta que se destaca dentro de las herramientas de libre distribución empleadas en la universidad para efectuar pruebas de rendimiento sobre aplicaciones web.

Para realizar las pruebas de carga y de estrés a la aplicación se utiliza un servidor con las siguientes características: Procesador (CPU): Intel Pentium 4, CPU M 430 @ 1.80GHz con memoria RAM de 1024MB y disco duro de 60 GB de capacidad. Fueron escogidos tres grupos de 100, 200 y por último 400 hilos lo cual representa la cantidad de trabajadores existentes en la Facultad 3 de la UCI, y figura la cantidad de usuarios que realizan las peticiones a la aplicación.

Tabla 29: Resultados de las pruebas de carga y estrés.

Cantidad de hilos	Cantidad de Peticiones	Rendimiento(segundos)
100	4800	6,6 sec
200	10250	13,5 sec
400	16400	17,66 sec

3.4 Validación de la solución propuesta

El problema a resolver en el presente trabajo está relacionado con la agilización del procesamiento de la información generada en las secciones sindicales.

En la actualidad el registro y obtención de datos de interés en estas es realizado de forma manual, por lo que provoca que cuando se desee obtener un determinado reporte, los resultados no se alcancen en un periodo de tiempo breve. La nueva herramienta desarrollada permitirá que la información deseada se obtenga más rápidamente. Con el propósito de comprobar lo anteriormente planteado se realizaron algunas pruebas. Para realizar dichas pruebas se tomó como muestra datos reales correspondientes a 50 afiliados pertenecientes a la sección sindical CEGEL. A continuación se presenta un resumen de los resultados obtenidos:

Tabla 30: Validación de la solución propuesta.

Tipo de información a obtener	Tiempo aproximado para obtener la información	
	Manual	Utilizando el sistema
Nombres y apellidos de los afiliados atrasados en el pago de la cuota sindical	2 minutos	5 segundos
Nombres y apellidos de los afiliados que han incumplido el compromiso de pago del día del haber	1.5 minutos	4 segundos
Nombres y apellidos de los afiliados que terminaron de pagar todo el año.	4 minutos	6 segundos
Nombres y apellidos de los afiliados que estuvieron atrasados en el pago de la cuota sindical o del día del haber en algún momento del año	15 minutos	5 segundos
Nombres y apellidos de los afiliados que han causado baja en una sección sindical.	35 segundos	3 segundos
Nombres y apellidos de los afiliados que han causado alta en una sección sindical.	30 segundos	3 segundos
Nombres y apellidos de los afiliados que han resultado destacados.	2 minutos	4 segundos

Obtener los acuerdos realizados y su estado.	10 minutos	5 segundos
Obtener las inquietudes y su estado.	10 minutos	5 segundos

El sistema posibilitará además disminuir el esfuerzo realizado por secretarios generales, financieros y activistas en la generación de informes, ya que para realizarlos no tendrán que chequear el estado de cada acuerdo, inquietud, pago del día de la patria por trabajador, pago de la cuota sindical por trabajador, entre otros. Otro valor agregado será que la información de interés de cada sección estará disponible en todo momento.

3.5 Conclusiones parciales

Como consideraciones finales del presente capítulo se relacionan las siguientes:

- ✓ La aplicación de la métrica de la calidad de la especificación permitió que cada requisito tuviera una única interpretación.
- ✓ A través de las métricas de validación de diseño, tamaño operacional de la clase y acoplamiento entre clases de objetos, se demostró que el diseño es simple y posee una buena calidad, además de un muy buen nivel de aceptación para realizar la implementación.
- ✓ Mediante diferentes pruebas, como la de camino básico, evaluación dinámica y prueba de funcionalidad y pruebas de aceptación, se evaluó la calidad y correcto funcionamiento del componente propuesto, además de certificar que el sistema es válido para el cliente.
- ✓ Los resultados que se obtuvieron demuestran la validez del sistema implementado y la solución de la problemática planteada.

Conclusiones generales

Una vez finalizado el desarrollo de este trabajo se arribó a las siguientes conclusiones:

- ✓ El estudio de sistemas similares permitió concluir que era necesario desarrollar uno nuevo.
- ✓ El estudio realizado sobre metodologías, herramientas y tecnologías permitió seleccionar las más adecuadas para el desarrollo del sistema.
- ✓ La aplicación de la métrica para la calidad de la especificación permitió que cada funcionalidad especificada estuviera libre de ambigüedad, asegurando así la comprensión de estas por el equipo de desarrollo.
- ✓ La realización del diseño permitió materializar con precisión los requerimientos del cliente. Los resultados obtenidos en la aplicación de las métricas validaron una alta reutilización de clases, además, que el acoplamiento y la complejidad de la implementación tienen niveles bajos.
- ✓ La aplicación de métricas y pruebas de software posibilitó verificar la calidad del sistema desarrollado, y que se realizaran todas las correcciones necesarias
- ✓ Se desarrolló una aplicación web que permite agilizar la gestión de la información generada en las secciones sindicales de la Facultad 3 de la Universidad de las Ciencias Informáticas. Esta cumple con los requisitos definidos por el cliente, por lo que se le dio cumplimiento a los objetivos propuestos en este trabajo.

Recomendaciones

Debido a los resultados obtenidos y a la experiencia adquirida durante la realización de este trabajo, y con el propósito de asegurar la posterior ampliación, modificación y mejora del sistema de gestión, se exponen a continuación algunas recomendaciones:

- ✓ Utilizar el sistema en otras secciones sindicales de la Universidad de las Ciencias Informáticas.
- ✓ Implementar un sistema de trazas.

Glosario de términos

UML: Lenguaje de Modelado Unificado (Traducido de las siglas en inglés UML: *Unified Modeling Language*). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

IDE: Entorno Integrado de Desarrollo (Traducido de las siglas en inglés IDE: *Integrated Development Environment*). Es un programa informático que contiene un conjunto de herramientas de programación.

TWIG: Archivo de texto que pude generar cualquier tipo de contenido (HTML, XML, etc.). Motor de plantillas para PHP.

ORM: Mapeo objeto-relacional.

GNU/GPL: En español Licencia Pública General, es una licencia creada por la *Free Software Foundation* y orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software Libre.

API: Interfaz de programación de aplicaciones (Traducido de las siglas en inglés API: *Application Programming Interface*). Es un conjunto de funciones y procedimientos que ofrece una biblioteca para ser utilizado por otro software.

HTML: Lenguaje de marcado para hipertextos (Traducido de las siglas en inglés HTML: *Hypertext Markup Language*), utilizado para la confección de páginas web.

LDAP: Protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

SQL: Lenguaje declarativo de acceso a bases de datos relacionales (Traducido de las siglas en inglés SQL: *Structured Query Language*).

TCP/IP: Protocolo de Control de Transmisión/Protocolo de Internet (Traducido de las siglas en inglés TCP/IP: *Transmission Control Protocol/Internet Protocol*). Es un sistema de protocolos que hace posible los servicios de Internet y correo, permitiendo la transferencia de datos entre redes de ordenadores.

Bibliografía

Bustos, Guillermo. 2012. *Guía de Uso de la Herramienta CASE.* 2012.

Reynoso , Carlos y Kicillof, Nicolás . 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* s.l. : UNIVERSIDAD DE BUENOS AIRES, 2004.

Rumbaugh, James y Jacobson, Ivar. 2000. *El lenguaje Unificado de Modelado, Manual de Referencia.* 2000. ISBN: 8478290370.

(IEEE), Instituto de Ingenieros Eléctricos y Electrónicos. 1990. *Standard Glossary of Software Engineering Terminology.* 1990. 610.

Álvarez, Miguel Ángel. 2010. *Desarrollo Web.* 2010.

Apache. 2014. Apache. [En línea] 2014. http://httpd.apache.org/.

Armand, Sébastien. 2014. *Extending Symfony2 Web Application Framework.* s.l. : Packt Publishing Ltd,, 2014.

Beck, Kent. 1999. *Extreme Programming Explained. Embrace Change.* s.l. : Pearson Education, 1999.

Benjamín González, Pedro Boda, Kattia Ninahuanca, Javier Martínez. 2009. *Zend Framework Manual en español.* 2009.

Billy Reynoso, Carlos . *Introducción a la Arquitectura de Software.* s.l. : UNIVERSIDAD DE BUENOS AIRES.

Blaha, M. 2010. *Patterns of Data Modeling.* 2010.

Blanco, Carlos. 2012. *Entornos Integrados de Desarrollo.* 2012.

Bonanata, M. 2008. *Programación y Algoritmos.* s.l. : MP Ediciones S.A., 2008.

Brodie, M. L., J. Mylopoulos & J. W. Schmidt. 1990. *On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases, and Programming Languages.* 1990.

Brualla, C. R. 2005. *¨Curso de introducción a la Calidad y Mejora del Proceso Software¨.* 2005.

Bureau. 2008. Bureau. [En línea] 2008. [Citado el: 21 de mayo de 2014.] Bureaudeprensa.com.

Calderón , Amaro y Valverde Rebaza, Sarah Dámaris. 2007. *Metodologías Ágiles.* Trujillo-Perú : Universidad Nacional de Trujillo, 2007.

Dolado, José Javier. 2007. *Técnicas Cuantitativas para la Gestión en la Ingeniería del Software.* 2007.

Eguiluz, Javier. 2012. *Desarrollo web ágil con Symfony2.* 2012.

F Durán, F. Gutiérrez and E. Pimentel. 2007. *Programación orientada a objetos con Java.* s.l. : Paraninfo, 2007.

Fernández, Adelaida Ramírez. 2012. *Clasificaciones de tipos de requisitos para la mejora del proceso de desarrollo del software.* Madrid : Universidad Carlos III de Madrid. Departamento de Informática, 2012.

Figueroa, Roberth F. , Solís, Camilo J. y Cabrera, Armando A. *Metodologías tradicionales vs metodlogías ágiles.* s.l. : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

GAMMA, E., HELM, R. y JOHNSON, R. y VLISSIDES, J. 2000. *Patrones de diseño.* 2000.

Gamma, Erich., Helm, R., Johnson, R., & Vlissides, J. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software. s.l: Addison-Wesley Professional.* 1995.

García, Prof.César Arturo Guerra. 2007. *Ingeniería de Requerimientos.* México : Universidad Politécnica de San Luis Potosí, 2007.

Garlan , David y Shaw, Mary. 1994. *An introduction to software architecture.* s.l. : CMU Software Engineering Institute Technical Report, 1994. CMU/SEI-94-TR-21.

Giraldo, Luis. 2007. *Herramientas de desarrollo de software para Linux.* 2007.

Gutiérrez, Javier. 2006. *Frameworks de desarrollo.* 2006.

Gutiérrez, Jorge A. Saavedra. 2008. *PATRONES GRASP Parte II.* 2008.

Hernández Sampieri, Roberto, Fernández-Collado, Carlos y Baptista Lucio, Pilar. 2006. *Metodología de la Investigación Científica.* [ed.] Ricardo A. del Bosque Alayón. Cuarta. México D. F. : The McGraw Hill, 2006. pág. 850.

Heurtel, Olivier. 2011. *PHP 5.3: Desarrollar un sitio Web dinámico e interactivo.* s.l. : ENI, 2011.

IIS. 2014. [En línea] 2014. http://www.microsoft.com/spain/windowsserver2003/technologies/webapp/iis.mspx.

ISTQB. 2005. [En línea] 2005. http://www.istqb.org/fileadmin/media/SyllabusFoundation.pdf..

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley, 2000. pág. 464.

JQuery. 2014. [En línea] 2014. http://www.jquery.com.

Krutchen, P. 2002. *The Rational Unified Process An Introduction.* s.l. : Addison Wesley, 2002.

Lancker, L.V. 2005. *CSS en DHTML: JavaScript aplicado a hojas de estilo.* s.l. : ENI, 2005.

Larman, Craig. 1999. *UML y Patrones, 2da Edición.* 1999.

Letelier, Canós, José H. & Penadés. 2003. *Metodologías ágiles en el desarrollo de software: Extreme Programming (XP).* 2003.

Lincke, R., Lundberg, J. y Löwe, W. 2008. *Comparing software metrics tools. International Symposium on Software Testing and Analysis.* Berlin : s.n., 2008.

LLOYD, I. 2008. *The Ultimate HTML Reference. United States of America:.* 2008.

MacIntyre, Peter, and Ian Morse. 2008. *Zend Studio for Eclipse Developer's Guide .* s.l. : Sams Publishing, 2008.

McClure, Carma . Abril 1989. *The CASE Experience.* s.l. : BYTE, Abril 1989.

Metsker, Steven John. 2002. *Design Patterns Java™ Workbook.* s.l. : Addison Wesley, 2002.

Myers G. 2004. *"The art of software testing".John Wiley & Sons Inc.* 2004.

MySQL. 2014. MySQL. [En línea] 2014. [Citado el: 11 de febrero de 2014.] http://dev.mysql.com/doc/refman/5.0/es/features.html.

NetBeans. 2014. NetBeans. [En línea] 2014. [Citado el: 12 de febrero de 2014.] https://netbeans.org.

Oracle. 2014. Oracle. [En línea] 2014. [Citado el: 11 de febrero de 2014.] http://www.oracle.com.

Paradigm, Visual. 2013. Visual Paradigm. [En línea] 2013. [Citado el: 28 de mayo de 2014.] http://www.visualparadigm.com/product/vpuml/editions/standard.jsp.

Piattini, Mario. 2006. *Tecnología y diseño de bases de datos.* s.l. : Ra, 2006.

Píriz Calabria, Luis y Pablo. 2003. *Metodología XP.* Uruguay : s.n., 2003.

Potencier, Fabien. 2011. *Symfony Project.* 2011.

Pressman, Roger S. 2005. *Ingeniería de Software 6th.* s.l. : Ed.McGraw, 2005.

Pressman, Roger S. 2010. *Software Engineering: A Practitioner's Approach.* s.l. : McGraw-Hill, 2010.

Programación. 2013. *Programación en Castellano.* 2013.

Quiang Xue and Xiang Wei. 2008-2012. *The Definitive Guide to Yii.* 2008-2012.

Raghavan, Sridhar. 1994. *Lecture Notes on Requirements Elicitation.* Pittstburgh (E.E.U.U) : Software Engineering Institute, 1994.

Roberto, Hernández Sampieri, Carlos, Fernández Collado y Pilar, Baptista Lucio. 2006. *Metodología de la Investigación Científica.* Iztapalapa, México : s.n., 2006. 4ta Edición.

Rodríguez Salas, Karla. 2013. *Gestión de la información en las organizaciones.* 2013.

S. Pressman, Roger. 2007. *Software Engineering. A Practitioner´s Aprproach.* Seventh. New York : Mc Graw Hill, 2007. pág. 980.

Santos González, , Manuel, Patiño Cortés, Ismael y Carrasco Vallinot, Raúl. *Fundamentos De Programación.*

Schenone, Marcelo Hernán. 2004. *Diseño de una Metodología Ágil de Desarrollo de Software.* 2004.

Schwaber, Ken. 2010. *Advanced Development Methods. SCRUM Development Process Retrieved.* 2010.

SECURED. 2013. *JavaScript(Manual FV).* 2013.

Shiflett, Chris. 2006. *Essential PHP security.* s.l. : O'Reilly Media, Inc, 2006.

Smith, Gregory. 2010. *PostgreSQL 9.0: High Performance.* s.l. : Packt Publishing Ltd, 2010.

Sommerville, Ian. 2005. *Ingeniería de Software, Séptima edición.* Madrid : PERSON EDUCATION. SA., 2005.

SWEBOK. 2004. *"Guide to the Software Engineering Body of Knowledge SWEBOK". IEEE Computer Society.* 2004.

T.E.Foundation. 2013. Eclipse. *The Eclipse Foundation open source community web site.* [En línea] 2013. [Citado el: 10 de diciembre de 2013.] www.eclipse.org.

Vaswani, Vikram. 2009. *PHP Beginner's Guide.* New York : Mc Graw Hill, 2009.

Vidal Ledo, María Josefina, and Ana Bárbara Araña Pérez. 2012. *Gestión de la información y el conocimiento.* 2012.

Wesley, Addison. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

Winesett, Jeffrey. 2010. *Agile Web Application Development with Yii1.1 and PHP5.* 2010.

Woodcock, Sanjiv Augustin y Susan. 2003. *Agile Project Management.* s.l. : ccPase, 2003.

Zend, Studio. 2014. Zend Studio. [En línea] 2014. [Citado el: 12 de febrero de 2014.] www.zend.com/en/products/studio.

Anexos

Anexo 1. Acta de liberación de calidad de CEGEL

UCI Universidad de las Ciencias Informáticas | FACULTAD # 3
CENTRO DE GOBIERNO ELECTRÓNICO

CEGEL

Acta de Liberación Interna de Productos Software

Fecha de emisión del acta: 11/06/2014

Emitida a favor de: Tesis Sistema de Gestión Información de la Secciones Sindicales.

Datos del producto

Artefacto	Versión	Estado final	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación
Sistema de Gestión Información de la Secciones Sindicales.	*1.0*	*0*	*3*	*Evaluación dinámica* *Pruebas de Funcionalidad*	*11/06/14*

Javier Saname Mena
Autor

Javier Alejandro Domínguez Prado
Autor

Ing. Raúl Velázquez Alvarez
Asesor de Calidad del Centro

UCI Universidad de las Ciencias Informáticas
Centro de Gobierno Electrónico

1