Universidad de las Ciencias Informáticas

Facultad 3

Título: Desarrollo de una aplicación web que permita incrementar la celeridad en el procesamiento de la información generada en las secciones sindicales de la Universidad de las Ciencias Informáticas.

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

Autor: Maricet Moreta Fernández

Tutor: Ing. Elsydania López Guerra

Co-tutor: Ing. Yulia Fustiel Álvarez

Ciudad de la Habana, Cuba

Curso: 2012-2013

En Cuba nadie ha hecho tanto en tan poco tiempo.

Fidel Castro.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste se firma la presente a los ____ días del mes de junio del año 2013.

_____

Maricet Moreta Fernández

Autor

_____ _____

Ing. Elsydania López Guerra Ing. Yulia Fustiel Álvarez

Tutor Co-tutor

DATOS DE CONTACTO

Autor: Maricet Moreta Fernández
Correo Electrónico: mmoreta@estudiantes.uci.cu

Tutor: Ing. Elsydania López Guerra
Correo Electrónico: elguerra@uci.cu
Co-Tutor: Ing. Yulia Fustiel Álvarez
Correo Electrónico: yfustiel@uci.cu

AGRADECIMIENTOS

Quiero agradecer a mi familia, principalmente a mi mamá que siempre me ha apoyado en todo y me ha dado fuerzas cuando me he sentido insegura. A mi abuelita del alma que siempre cuida de mí y nunca se queja de nada. A mi tía, a todos mis tíos que son padres para mí, a mi hermana que aunque somos muy distintas en los momentos difíciles nos volvemos confidentes. A mi hermanito que lo quiero con todo mi vida aunque casi no he podido verlo crecer. A mis tutoras, por haberme apoyado durante todo el trabajo y ser comprensivas. A todos los profesores que me han dado clases. A mis compañeros de grupo con los que he compartido hermosas ocasiones durante estos 5 años. Y finalmente un agradecimiento especial para mi novio Jose que siempre me ha dado ánimos y ha soportado mis malos humores pero nunca me ha dejado de demostrar cuanto me quiere.

Maricet Moreta Fernández

DEDICATORIA

Quiero dedicar este trabajo a mi mamá que siempre ha soñado por el momento en que su hija logre un título de profesional y hoy puede ver cumplido ese sueño. A mi tía y mis tíos que siempre se preocupan por mí. A mis hermanas y mi hermanito que quiero tanto. A mi papá. A mi padrastro que ha hecho muy feliz a mi mamá. A mi abuelita que siempre me tiene presente y reza por mí. A todas mis amistades y mis compañeros de aula. A mi novio por siempre estar conmigo en las buenas y las malas.

Maricet Moreta Fernández

RESUMEN

En la Universidad de las Ciencias Informáticas (UCI), la Central de Trabajadores de Cuba (CTC) está conformada por varios sindicatos y estos a su vez están compuestos por secciones sindicales las cuales están integradas por un conjunto de trabajadores. En dichas secciones sindicales se lleva un registro manual de toda la información que se genera. Esto trae consigo que se consuma una cantidad de tiempo considerable en gestionar la información.

Para la obtención del sistema se estudiaron las principales herramientas y tecnologías propicias para la construcción de la aplicación así como las metodologías que actualmente han tomado auge en el desarrollo de software. Se obtienen los artefactos para la implementación de la solución, los cuales son validados a través de métricas y pruebas de caja negra arrojando resultados satisfactorios. Finalmente se propone un sistema que permita incrementar la celeridad en el procesamiento de la información generada en las secciones sindicales, permitiéndoles que los procesos realizados de manualmente sean simplificados por el uso de las Tecnologías de la Informática y las Comunicaciones (TIC´s).

PALABRAS CLAVES:

Historias de usuario, Metodología XP, Métrica, Netbeans, PostgreSQL, Secciones Sindicales, Symfony 2.

Índice

Introducción

La revolución informática, a pesar de haberse iniciado hace más de cuarenta años, se ha visto intensificada en las últimas décadas mediante la creación de nuevas tecnologías. Esto trae consigo que en el mundo actual muchos países se vean inmersos en un proceso de informatización.

En el caso de Cuba, aunque no es un país desarrollado y a pesar de los obstáculos que ha tenido que afrontar debido a la imposición del bloqueo económico, ha obtenido resultados satisfactorios en el campo de la informática. En este aspecto se destaca la creación de centros de enseñanza que tengan como base el estudio de esta ciencia y en la cual las universidades desempeñan un papel protagónico y decisivo.

En estas circunstancias se crea la Universidad de las Ciencias Informáticas (UCI) marcando con su surgimiento el inicio de una nueva era de la informática en Cuba. Esta es una universidad productiva, que sobresale entre otros aspectos por producir software y servicios informáticos a partir de la vinculación estudio-trabajo como modelo de formación, logrando de esta forma un gran avance en la informatización de la sociedad y un incremento en la economía cubana con la comercialización de los productos que se desarrollan.

La UCI está conformada por un gran número de estudiantes, trabajadores y profesores que se encuentran agrupados por diferentes organizaciones de masas. Entre estas se distingue el movimiento sindical representado por la Central de Trabajadores de Cuba (CTC), organización de masas integrada por todos los sindicatos y sus afiliados con el objetivo de defender sus intereses, propiciar la unidad, contribuir a la educación económica, política, ideológica y cultural de la sociedad, representar al movimiento sindical cubano en el plano internacional y afianzar la solidaridad, salvaguardando la independencia, la Revolución y el Socialismo.

Los sindicatos que integran a la CTC están constituidos por las organizaciones sindicales de base, que no son más que las secciones sindicales, siendo estas el elemento fundamental del sindicato. Las secciones sindicales están integradas por todos trabajadores afiliados a un sindicato dentro de una empresa y en la cual estas personas pueden ser elegidas para diversos cargos. Los cargos imprescindibles son el de secretario general y secretario, pero puede haber otros como financiero, activista de emulación, entre otros.

En la actualidad los financieros y activistas de la universidad deben llevar un registro manual de toda la información, como por ejemplo los meses pagados por cada trabajador, si han efectuado o no el pago de las MTT, los trabajadores destacados y vanguardias, los acuerdos incumplidos, entre otros. Producto de esta forma de procesar los datos, cuando se desea obtener reportes o determinada información de una sección sindical, se necesita emplear varios minutos para conseguirla, dependiendo de la cantidad de trabajadores afiliados que posea.

A partir de la situación descrita con anterioridad, se define el siguiente **problema a resolver:**

¿Cómo gestionar los procesos que se realizan en las secciones sindicales de la universidad de manera que se incremente la celeridad en el procesamiento de la información generada en las mismas?

Por consiguiente el **objeto de estudio** es el proceso de desarrollo de software.

Con el fin de darle solución al problema planteado se identifica el siguiente **objetivo general**: Desarrollar una aplicación web que permita incrementar la celeridad en el procesamiento de la información generada en las secciones sindicales de la universidad.

En consecuencia el **campo de acción** es:

Desarrollo de sistemas de gestión de procesos sindicales.

Para alcanzar la meta propuesta, se derivaron los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación.
- Obtener los artefactos necesarios para el desarrollo de la solución.
- Implementar las funcionalidades especificadas.
- Validar los resultados obtenidos.

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 frameworks 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-Colaboración).
12. Especificación de tareas de ingeniería.
13. Realización de tareas de ingeniería.
14. Implementación de las historias de usuarios especificadas.
15. Descripción de los patrones de diseño y arquitectura utilizados en la propuesta.
16. Validación del diseño realizado.
17. Validación de las funcionalidades implementadas.

A partir de lo expuesto con anterioridad, se propone la siguiente **idea a defender**:

El desarrollo de una aplicación web para gestionar los procesos fundamentales que se realizan en

las secciones sindicales de la universidad posibilitará el incremento de la celeridad en el procesamiento de la información generada en las mismas.

Para dar cumplimiento a los objetivos se emplearon varios **métodos científicos de la investigación**. A continuación se detallan los métodos teóricos y empíricos utilizados:

Los métodos teóricos utilizados son los siguientes:

- **Análisis histórico-lógico:** Permitió realizar un estudio de las principales aplicaciones informáticas que se han desarrollado para las secciones sindicales que existen en Cuba y en la universidad.
- **Analítico-sintético:** Permitió obtener los elementos significativos a partir del estudio realizado sobre el proceso de desarrollo de software y las actividades que se realizan en las secciones sindicales.
- **Modelación:** Permitió la creación de modelos que representan abstracciones de la realidad y de la cual se obtienen los artefactos con el objetivo de obtener una representación visual del sistema que se va a desarrollar.

Los métodos empíricos utilizados son los siguientes:

- **Entrevista:** Facilitó la obtención de información acerca de las actividades que realizan los financieros y activistas de las secciones sindicales.
- **Observación:** Posibilitó obtener una mejor perspectiva acerca del funcionamiento y organización de las secciones sindicales, así como la interacción de los financieros y activistas.
- **Medición:** Permitió realizar la medición de la calidad de los artefactos que se obtuvieron durante el desarrollo del sistema.

El presente trabajo está estructurado en tres capítulos, además contiene varios anexos con los artefactos generados durante el desarrollo. A continuación se describe el objetivo principal de cada uno de los capítulos:

Capítulo 1: Fundamentación Teórica.

En este capítulo se establecen todos los elementos teóricos de la investigación. Se realiza un estudio del estado del arte de sistemas similares al que se desea implementar, así como de marco de trabajo, patrones de diseño, lenguajes de programación, metodologías y herramientas orientadas al desarrollo web. Además se estudiaron los elementos fundamentales del proceso de desarrollo de software para dar solución al problema planteado.

Capítulo 2: Solución Propuesta.

En este capítulo se presenta la solución propuesta con todos los aspectos definidos en la fundamentación teórica. Esta solución contiene los artefactos necesarios para desarrollar el sistema a construir, entre los que se encuentran: historias de usuarios, tarjetas CRC, modelo de datos,

tareas de ingeniería, entre otros.

Capítulo 3: Validación de resultados.

En este capítulo se realizan validaciones de los resultados que se obtuvieron del sistema ya en ejecución, haciendo uso de algunos de los métodos definidos en el marco teórico de la investigación que permiten establecer comparaciones entre los resultados obtenidos. Además, se valida que el diseño realizado cumpla con la calidad requerida y que el sistema implementado satisface las necesidades de los clientes.

Capítulo 1: Fundamentación teórica.

Introducción.

En el presente capítulo se abordan los elementos esenciales relacionados con las secciones sindicales así como los sistemas existentes similares al que se desea desarrollar. Se hace referencia a las tendencias y tecnologías actuales sobre metodologías de desarrollo y herramientas para el modelado de datos. Además se realiza un estudio de algunos frameworks de desarrollo, lenguajes de programación y sistemas gestores de base de datos, con el objetivo de seleccionar los más indicados para el desarrollo de una solución de software con las características deseadas.

1.1 Secciones sindicales

Las secciones sindicales representan la base organizativa del sindicato que a la vez, junto con sus afiliados, integran la Central de Trabajadores de Cuba (CTC). Son los órganos deliberantes, consultivos y ejecutivos en el ámbito laboral, creados con el objetivo de unificar las masas trabajadores con el propósito de lograr la justicia social, libertad e independencia en el movimiento sindical cubano. Son organizaciones autónomas capaces de tomar sus propias decisiones y en la cual sus miembros aprueban sus estatutos y reglamentos. Estas secciones están conformadas por todos los trabajadores sin distinción de sexo, raza ni convicción religiosa organizados de forma voluntaria y encargados de organizar y controlar todas las actividades realizadas por los trabajadores y afiliados. (Estatutos, 2006)

Las secciones sindicales de una misma empresa, unidad presupuestada, establecimiento, etc., pertenecerán a un solo sindicato. En la universidad todas las áreas que la conforman están representadas por su respectiva sección sindical y sus afiliados. El secretariado ejecutivo de estas secciones sindicales está conformado por un secretario general y activistas para las distintas tareas que se ejecutan. Las personas que son elegidas desempeñan una determinada función como por ejemplo: gestión de acta, de funcionamiento, de finanzas, de inquietudes, entre otros. Las secciones sindicales llevan a cabo sus reuniones una vez al mes en las cuales se toman acuerdos, se plantean las inquietudes y se les da repuesta a las que fueron planteadas en reuniones anteriores, además de tratar cualquier otro tema de interés asociado al funcionamiento de las secciones sindicales.

1.2 Sistemas existentes.

Actualmente existen portales y sitios destinados a los sindicatos, desde las más sofisticadas de ámbito nacional hasta las más sencillas páginas de pequeños sindicatos. Estas aplicaciones se encargan de gestionar la información que se genera en las secciones sindicales, lo cual los hacen ser similares al producto que se desea desarrollar.

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

La Central de Trabajadores de Cuba (CTC) cuenta con una aplicación web, donde se automatiza todo el proceso de gestión de la información, así como el control de las secciones sindicales que la

conforman. El sistema se encarga del proceso de generación de la información que comienza en los municipios subordinados a las provincias y estas a cada uno de los sindicatos y a su vez estos a la CTC nacional. La aplicación realiza algunas funcionalidades como son: reporte de bajas del sindicato, gestionar los modelos A1[i], gestionar la información perteneciente a las secciones sindicales de los municipios y provincias respectivamente así como brindar los datos de los miembros del sindicato que se desee conocer.

Sistema de Gestión para la Sección Sindical Vicerrectoría Primera de la Universidad de Ciencias Informáticas.

La vicerrectoría primera de la universidad dispone de un sistema capaz de administrar y controlar toda la información que se genera en la sección sindical. La aplicación desarrolla algunas funcionalidades tales como: gestión de las actividades sindicales, reglamentos y evaluación de los afiliados además de la generación de reportes, gestión de los participantes en las actividades y de los parámetros de emulación. Sin embargo el sistema no permite dar de baja o alta a un afiliado, al igual que modificar el salario de un afiliado, de manera que no se tiene un control de las finanzas del sindicato.

Sitio web: Sección Sindical de la Facultad No.8 de la Universidad de Ciencias Informáticas.

El sitio web: Sección Sindical de la Facultad No.8 se encarga de gestionar los procesos que se llevan a cabo en el sindicato de la facultad. El sitio cumple con diversas funcionalidades como: gestionar emulación de la sección sindical, gestionar las finanzas y mostrar información, como por ejemplo: reglamentos, historia del sindicato, estructura de los comités sindicales de la facultad, las actividades que se realizarán, noticias nacionales e internacionales y reflexiones de Fidel. Sin embargo el sistema no permite gestionar los acuerdos tomados en las reuniones del sindicato.

En el ámbito nacional los sistemas informáticos desarrollados para las secciones sindicales que se analizaron se enfocan en realizar publicaciones de la información correspondiente al sindicato y gestionar algunos de los procesos que se realizan en las secciones sindicales, sin abarcar todas sus funciones. De esta forma se puede afirmar que los sistemas existentes estudiados no ofrecen una solución completa a todos los procesos que se desean gestionar en el sistema a desarrollar.

1.3 Metodologías para el desarrollo de software.

El desarrollo de software no es una tarea sencilla, prueba de ello son las investigaciones que se realizan con el fin de obtener el éxito en este campo y lograr la calidad del producto que es el principal objetivo de las organizaciones. De ahí que los desarrolladores prestaran más atención a este tema y surgieran nuevas propuestas metodológicas con diferentes resultados en el proceso de desarrollo del software. (Letelier, Canós & Penadés, 2003).

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 nuevo software. Una metodología durante su ciclo de vida indica qué es lo que hay que obtener durante

todo el proceso de desarrollo, es decir, cómo se obtienen los productos parciales y finales pero no cómo hacerlos. (García, et al., 2009-2010)

Estas metodologías se clasifican en dos grandes grupos: metodologías pesadas o tradicionales y metodologías ágiles o ligeras:

Las metodologías tradicionales están enfocadas en el control de proceso donde se establecen rigurosamente las actividades que se realizarán y los artefactos que se obtendrán. Estas metodologías han demostrado ser efectivas y necesarias en proyectos de larga duración pero han presentado problemas en proyectos donde el entorno es muy cambiante y en el cual se exige reducir el tiempo de desarrollo pero sin que afecte la calidad del producto.

Las metodologías ágiles están enfocadas en el factor humano donde se establece la colaboración con el cliente y el desarrollo incremental del software con iteraciones muy cortas. Se caracterizan por ser flexibles al cambio y por poder reducir el tiempo de desarrollo del software demostrando así que es más importante contar con un software funcional que una documentación excesiva. Estas metodologías han demostrado tener éxito en proyectos pequeños. (Figueroa, Solís, Cabrera. 2007)

A partir de lo analizado con anterioridad, teniendo en cuenta que el sistema a desarrollar no posee un gran número de funcionalidades a implementar y el equipo de desarrollo es pequeño, se decide enfocarse en el estudio de las metodologías ágiles por ser las idóneas para este tipo de proyecto.

A continuación se detallan las características esenciales de las metodologías ágiles más destacadas en la actualidad.

1.3.1 Extreme Programming (XP)

Extreme Programming (XP) es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y retroalimentación o reutilización del código desarrollado. Esta metodología se caracteriza por centrarse 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 mantener una conversación fluida, además de 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, y donde existe un alto riesgo técnico. (Letelier, Canós & Penadés, 2003).

Algunas de las características fundamentales de XP son: (Mendoza Sánchez, 2004)

- Pruebas unitarias: se basa en las pruebas realizadas a los principales procesos con el objetivo de detectar futuros errores.
- Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

- Programación en pares: consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

A continuación se detallan las principales ventajas de la metodología XP: (Figueroa, Solís, Cabrera. 2007)

- Apropiada para entornos volátiles.
- Preparada para el cambio, lo que significa reducir su coste.
- Planificación más transparente para los clientes, conocen las fechas de entrega de las funcionalidades.
- Permite definir en cada iteración cuáles son los objetivos de la siguiente.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

Entre las desventajas de la metodología XP se pueden considerar las siguientes: (Figueroa, Solís, Cabrera. 2007)

- No se tiene la definición de costo y el tiempo de desarrollo.
- El sistema va creciendo después de cada entrega al cliente y no se puede saber con certeza si el cliente no necesitará una funcionalidad más.
- Se requiere de la presencia constante del usuario, lo cual en la realidad es muy difícil de lograr.

1.3.2 Proceso Unificado Ágil (AUP).

El Proceso Unificado Ágil (AUP) es una versión simplificada del Proceso Unificado de Rational (RUP). Describe de manera fácil la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos válidos para RUP. 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 proceso establece un modelo más simple que el de RUP porque integra en una sola las disciplinas de modelado del negocio, requisitos y análisis y diseño, en el caso del resto de las disciplinas coinciden con las restantes de RUP. (Flores, 2006)

Las principales características de la metodología AUP son: (Fowler, Martin,2004)

- 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.

La metodología AUP es ágil porque está basada en los principios ejemplificados a continuación: (Martín Ramírez, 2009)

- Simplicidad: Todo se describe concisamente utilizando poca documentación.
- Agilidad: El ajuste a los valores y principios de *La Alianza Ágil*. [ii]
- Centrarse en actividades de alto valor: La atención se centra en las actividades que en realidad lo requieren, no en todo el proyecto.
- Facilidad de adaptar el producto que se obtiene de manera que satisfaga sus propias necesidades.

1.3.3 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 y las reuniones diarias que se llevan a cabo a lo largo del proyecto 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: Scrum de Scrums. (K Schwaber, 2010)

Las principales características de la metodología Scrum son las siguientes: (Kniberg, 2007)

- Permite la creación de equipos autoorganizados impulsando la co-localización de todos los miembros del equipo, y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.
- Posibilita que los clientes puedan cambiar de idea sobre lo que quieren y necesitan durante el desarrollo del proyecto.
- Maximiza la capacidad del equipo de realizar las entregas rápidamente y de respuesta a los requisitos emergentes.

1.4 Fundamentación de la metodología a utilizar

Después del análisis de cada metodología propuesta se decide utilizar la metodología XP para la realización del trabajo por ser la que más se adapta a las características del proyecto. Es una metodología que consiste en una programación rápida o extrema que se identifica por tener como parte del equipo de desarrollo al usuario final, pues es la clave para llegar al éxito en el proyecto. Además es capaz de adaptarse a los cambios que puedan surgir en cualquier punto del ciclo de vida del proyecto. XP es ante todo una metodología que busca la satisfacción del cliente para ello le ofrece la posibilidad de mantener un constante intercambio con el equipo de desarrollo, de forma tal que pueda intervenir cuando el producto que se obtenga en cada iteración no sea el deseado.

1.5 Herramientas CASE.

Las herramientas CASE (Ingeniería de Software Asistida por Computadora) son aplicaciones

informáticas dedicadas al aumento de la productividad en el desarrollo de software. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. (*Loucopoulos & Karakostas, 1995*)

Entre las principales herramientas utilizadas para el modelado de datos se encuentran Erwin, Enterprise Architect y Visual Paradigm, detalladas a continuación.

1.5.1 Visual Paradigm

Visual Paradigm 8.0 es una herramienta de modelado profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue. El software de modelado UML[iii] (Lenguaje de Modelado Unificado) ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite modelar todos los tipos de diagramas de clases así como generar código inverso, código desde diagramas y documentación. Es fácil de instalar y actualizar además de ser compatible entre ediciones.

Dentro de las principales características de la herramienta se encuentran: disponibilidad de múltiples versiones, de acuerdo a su necesidad, así como la capacidad de integrarse a los principales IDE[iv]´s (Entorno de Desarrollo Integrado), ofrece un diseño centrado en casos de uso y enfocado al negocio de manera que genera un software de mayor calidad. Otra de sus características es que permite realizar ingeniería tanto directa como inversa. Además la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto. Es capaz de generar la documentación del proyecto en diferentes formatos como web o pdf y permite el control de versiones. (Paradigm, 2011)

1.5.2 ERwin

La herramienta ERwin Data Modeling ofrece un entorno de modelado de datos de colaboración para administrar datos empresariales con una interfaz intuitiva y gráfica. Las partes técnicas y de negocio pueden compartir una vista de la información en contexto. Con una vista centralizada de definiciones de datos clave, puede comprender mejor los datos corporativos, que se administran de manera más eficiente y rentable, mediante la colaboración por diseño. Brinda productividad en diseño, generación, y mantenimiento de aplicaciones desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la base de datos diseñada. (Erwin. 2013)

1.5.3 Enterprise Architect

Enterprise Architect (EA) es una herramienta para el diseño y construcción de sistemas de software. EA soporta la especificación de UML 2.0, que describe un lenguaje visual por el cual se pueden definir mapas o modelos de un proyecto. Es una herramienta progresiva que cubre todos los aspectos del ciclo de desarrollo, proporcionando una trazabilidad completa desde la fase inicial

del diseño a través del despliegue y mantenimiento. También provee soporte para pruebas, mantenimiento y control de cambios. (Solus S.A, 2006)

Enterprise Architect sustenta todos los diagramas y modelos UML. Puede modelar procesos de negocio, sitios web, interfaces de usuario, redes, configuraciones de hardware, mensajes y más. Es capaz de estimar el tamaño de su proyecto en esfuerzo de trabajo en horas, capturar y trazar requisitos, recursos, planes de prueba, solicitudes de cambio y defectos.

1.6 Fundamentación de la herramienta CASE a utilizar

Se decide utilizar como herramienta CASE el visual paradigm para la realización del modelo de datos, debido a su alta capacidad de integración con lenguajes de programación. Es una herramienta multiplataforma, de código abierto, disponible en varios idiomas además de ser fácil de instalar y actualizar. Visual Paradigm facilita la organización de los diagramas generando la documentación del proyecto en varios formatos. Además de ser capaz de generar código, modelo de datos, entre otras funcionalidades útiles para el desarrollo de un producto.

1.7 Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. (Gamma, 1995)

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Entre los patrones de diseño se pueden mencionar los patrones de asignación de responsabilidades GRASP (patrones generales de software para asignación de responsabilidades, siglas de General Responsibility Assignment Software Patterns) y los patrones GOF (siglas de Gang of Four) que es el nombre con el que se conoce comúnmente a los autores del libro Design Patterns.

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos. Dentro de este grupo de patrones se encuentran los siguientes: experto, creador, bajo acoplamiento, alta cohesión, controlador, fabricación pura, indirección, variaciones protegidas, no hables con extraños y polimorfismo.

Los patrones de diseño GOF son 23 y están clasificados según su ámbito en objeto y de clase y según su propósito en creacionales, estructurales y de comportamiento, detallados a continuación:

- **Creacionales**: Los patrones creacionales abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados. Ejemplos de patrones de creación son: Abstract Factory[1], Factory Method[2], Prototype[3] *y* Singleton[4].

[1] Fábrica abstracta

[2] Método de fabricación

- **Estructurales**: Los patrones estructurales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades. Algunos ejemplos de patrones estructurales son: Adapter[5], Decorator[6], Fachada, *y* Proxy.
- **Comportamiento**: Los patrones de comportamiento están relacionadas con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos. Ejemplo de patrones de comportamiento son: Chain of Responsability[7], Interpreter[8], Observer [9] y Template Method[10]. (Gamma, 1995)

1.8 Arquitectura de software.

La arquitectura de software se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para la guía en el desarrollo de software dentro de un sistema informático. Así, los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado. (Casanovas, 2004)

Resulta muy complejo capturar la arquitectura software en un sólo modelo (o diagrama). Para manejar esta complejidad se representan diferentes aspectos y características de la arquitectura en múltiples vistas. Una vista es una presentación de un modelo, la cual es una descripción completa de un sistema desde una particular perspectiva.

Los patrones arquitectónicos, o patrones de arquitectura, son patrones de diseño de software que ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor.(Pressman, 2005)

Uno de los aspectos más importantes de los patrones arquitectónicos es que encarnan diferentes atributos de calidad. Por ejemplo, algunos patrones representan soluciones a problemas de rendimiento y otros pueden ser utilizados con éxito en sistemas de alta disponibilidad.

---

[3] Prototipo

[4] Instancia única

[5] Adaptador

[6] Decorador

[7] Cadena de responsabilidad

[8] Intérprete

[9] Observador

[10] Método plantilla

1.8.1 Patrón arquitectónico Modelo-Vista-Controlador (MVC).

El patrón arquitectónico MVC (Modelo Vista Controlador) permite realizar la programación multicapa, separando en tres componentes distintos los datos de una aplicación, la interfaz del usuario y la lógica de control. Este patrón se ve usualmente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes.(Larman, Craig,2000).

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

- El **modelo** representa la información con la que trabaja la aplicación, es decir, su **lógica de negocio**.
- La **vista** transforma el modelo en una página web que permite al usuario interactuar con ella.
- El **controlador** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Ejemplo de la estructura del patron arquitectornico MVC(Modelo Vista Controlador):

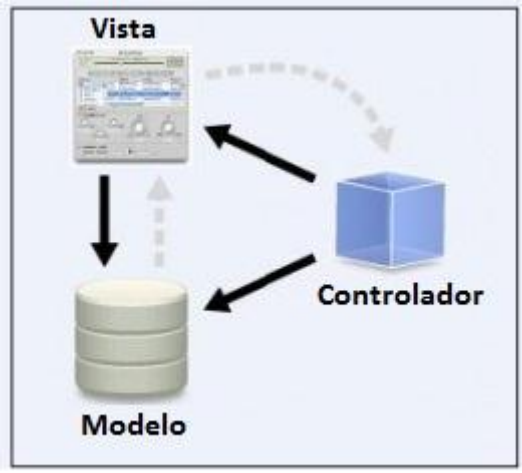

Figura 1.Patrón arquitectónico MVC(Modelo Vista Controlador)

La principal ventaja de utilizar esta arquitectura es que existe una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multilenguaje, distintos diseños de presentación, etc. sin alterar la lógica de negocio. La separación de capas es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y más fácilmente mantenibles, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos.

Al existir la separación de vistas, controladores y modelos es más sencillo realizar labores de mejora como:

• Agregar nuevas vistas.

• Agregar nuevas formas de recolectar las órdenes del usuario (interpretar sus modelos mentales).

• Modificar los objetos de negocios para migrar a otra tecnología.

• Las labores de mantenimiento también se simplifican y se reduce el tiempo necesario para ellas. Las correcciones solo se deben hacer en un solo lugar y no en varios como sucedería si se tuviera una mezcla de presentación e implementación de la lógica del negocio.

• Las vistas también son susceptibles de modificación sin necesidad de provocar que todo el sistema se paralice. Adicionalmente el patrón MVC propende a la especialización de cada rol del equipo, por tanto en cada liberación de una nueva versión se verán los resultados.

1.9 Marco de trabajo.

Un marco de trabajo (framework) en el desarrollo de software, no es más que una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Un marco de trabajo representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. (Dirk, 2000)

Algunos de los frameworks que se analizaron para el desarrollo de aplicaciones web son: Symfony 2, Zend Framework y Yii Framework los cuales se detallan a continuación.

1.9.1 Zend framework

Zend framework (ZF) es un marco de trabajo de código abierto para desarrollar aplicaciones web y servicios web con PHP 5. ZF es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de ZF es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. A menudo se refiere a este tipo de diseño como "use-at-will" (uso a voluntad). (Zend, 2006 – 2013)

1.9.2 Symfony 2

Symfony 2 es un marco de trabajo rápido, flexible y fácil de aprender que le permite a los desarrolladores construir aplicaciones webs más mantenibles. Symfony 2 ha sido desarrollado teniendo en cuenta el rendimiento como mayor prioridad, por lo que es uno de los frameworks más rápidos. Hasta 3 veces más rápido que Symfony 1.4 o Zend Framework 1.10 y consume la mitad de la memoria virtual. Se construye a base de bundles (conocidos como plugins en Symfony 1). Un bundle es un conjunto estructurado de archivos que implementan una característica única y que puede ser fácilmente compartido con otros desarrolladores. Los bundles permiten que los desarrolladores configuren y personalicen el sistema de una forma limpia. (Eguiluz, 2011)

Symfony2 es un marco de trabajo de código abierto por lo que los desarrolladores son libres de hacer lo que deseen con él, incluso en un entorno comercial. Está construido utilizando un contenedor de inyección de dependencias, inspirado en el marco de trabajo Spring de Java. En un

proyecto, el desarrollador no interactúa directamente con el contenedor. Todos los detalles de implementación están ocultos detrás de un buen sistema de configuración que permite personalizar todo a través de archivos .yml o .xml, o incluso a través de código PHP. (Eguiluz, 2011)

1.9.3 Yii framework

Yii es un marco de trabajo abierto (open source) de programación utilizado para desarrollar todo tipo de aplicaciones web. Entre sus principales características se destacan que es orientado a objetos, de software libre, posee un alto rendimiento basado en componentes, PHP y marco de trabajo de aplicaciones web. Yii se pronuncia en español como se escribe y es un acrónimo para "Yes It Is!" (En español: ¡Sí lo es!). Yii viene con una colección de documentos oficiales, tales como un tutorial para desarrollar un simple blog, una guía que recoge la descripción de cada función y una referencia de clases que ofrece todos los detalles acerca de las propiedades, métodos y eventos. Yii se puede utilizar de forma gratuita para desarrollar cualquier aplicación Web de código abierto. En general, el contenido de la documentación Yii puede ser copiado, modificado y redistribuido siempre y cuando la nueva versión de subvenciones dé las mismas libertades a los demás y reconozca a los autores del artículo de la documentación utilizada Yii. (Winesett, 2010)

1.10 Fundamentación del marco de trabajo a utilizar

Para la realización del proyecto se utiliza Symfony 2.2 por haber sido ideado para exprimir al límite todas las nuevas características de PHP en su versión 5.3 y por eso ser uno de los frameworks de PHP con mejor rendimiento. Además de que gracias a la peculiaridad que posee, de tener su arquitectura interna totalmente desacoplada permite que se reemplacen o se eliminen fácilmente aquellas partes que no encajen con el proyecto a desarrollar, lo cual le permite adaptarse con facilidad a los nuevos cambios que puedan surgir durante el proceso de desarrollo del software. También existen módulos que se han desarrollado para ser integrados a Symfony 2, para realizar operaciones tales como: generación de reportes en formato .pdf;[11] manipulación de seguridad y usuarios, gestión de envíos de correos electrónicos entre otros. Symfony 2 posee características que lo hacen único con respecto a otros frameworks, como el soporte HTTP,[12] el contenedor de inyección de dependencias, el sistema de plantillas Twig[v] y el sistema de bundles.[vi]

También brinda la facilidad de utilizar el ORM[13] Doctrine 2.0 c[vii]on el que se logra la independencia del gestor de base de datos. Además Symfony 2 hace uso de los lenguajes HTML 5 y de CCS 3 que se incluyen dentro de las twigs las cuales son las definidas por este marco de trabajo para ofrecerle al usuario un entorno vistoso y una mejor interacción con la aplicación desarrollada.

1.11 Frameworks de JavaScript

Básicamente un marco de trabajo de JavaScript es una biblioteca que contiene una serie de

[11] Formato de archivo de texto.

[12] Protocolo de transmisión del hipertexto.

[13] Mapeador de Objeto Relacional.

funciones y sentencias utilizadas para facilitar la interacción con los documentos HTML, poder manipular de manera sencilla el DOM,[14] desarrollar aplicaciones AJAX,[15] realizar animaciones y manipular eventos.

En su mayoría, los frameworks JavaScript proveen componentes para:

- **Compatibilidad**. Agregan la posibilidad de escribir código JavaScritp totalmente compatible con todos los navegadores y motores JavaScritp más utilizados. Esto aumenta la portabilidad y eliminan el problema de incompatibilidad entre navegadores y sus motores intérpretes JavaScritp.
- **Comunicación asíncrona** (Ajax). Usando este acercamiento, es fácil utilizar XMLHttpRequest para manejar y manipular los datos en los elementos de un sitio web, aumentando la interactividad y experiencia del usuario.
- **DOM**. Maximizan la capacidad de agregar, editar, cambiar, eliminar elementos de manera dinámica agregando bibliotecas que facilitan usar DOM.
- **Validación de Formularios**. Permiten de una manera relativamente fácil validar campos dentro de uno o varios formularios. Esto, desde el punto de vista del desarrollador, simplifica y reduce el código para procesar dichos formularios, ya que los datos llegan previamente validados, reduciendo los errores de tipos de datos.
- **Efectos visuales**. Utilizando la manipulación de los elementos, se pueden crear efectos visuales y animaciones. Entre los efectos se encuentran: Aparecer y Desaparecer, Redimensionamiento, entre otros.
- **Almacenamiento del lado del cliente.** En adición provee funciones para leer y escribir cookies. También proveen una abstracción de almacenamiento que permite a las aplicaciones web guardar datos de manera segura.
- **Manejo JSON**. Incrementa al máximo el manejo de datos, que pueden ser utilizados para presentar informaciones de manera dinámica y en tiempo de ejecución.
- **Manejo de Eventos**. Esta característica agregada, permite reaccionar de una manera determinada dependiendo de las acciones del usuario.
- **Recibidores de Datos**. Permiten utilizar diferentes formatos de datos como XML, HTML, Texto, JSON, ATOM, entre otros.
- "**Arrastra y Suelta**". Más conocido como Drag and Drop. Es una funcionalidad que brinda la posibilidad de arrastrar elementos dentro de una misma página que interactúe con el resto de los elementos. (Programming languages, 2000-2012)

1.11.1 JQuery

JQuery es un software libre y de código abierto, bajo la Licencia Pública General de GNU [viii]v2,

[14] Modelo de Objetos de Documento, es una forma de representar los elementos de un documento estructurado.

[15] JavaScript y XML asíncronos, es una técnica de desarrollo web para crear aplicaciones interactivas.

permitiendo su uso en proyectos libres y privativos. JQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX. Es una biblioteca de JavaScript, creada inicialmente por John Resig. (ScottGu, 2008)

Dentro de las funcionalidades de JQuery se destaca que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas Web.

Además es capaz de ofrecer una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Las principales características de jQuery son:

- Manipulación de la hoja de estilos CSS.
- Soporta diferentes extensiones.
- Ofrece varias utilidades como obtener información del navegador, operar con objetos y vectores, funciones como trim () (elimina los espacios en blanco del principio y final de una cadena de caracteres), etc.
- Compatibilidad con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 9+ y Google Chrome. (ScottGu, 2008)

1.11.2 ExtJs

Ext JS es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Fue desarrollada por Sencha. (Sencha, 2012)

Originalmente construida como una extensión de la biblioteca YUI por Jack Slocum, en la actualidad puede usarse como extensión para la biblioteca jQuery y Prototype. Desde la versión 1.1 puede ejecutarse como una aplicación independiente. (Extjs, 2012)

1.11.3 Prototype

Prototype es un marco de trabajo desarrollado en JavaScript por Sam Stephenson para el desarrollo sencillo y dinámico de páginas Web. Prototype simplifica gran parte del trabajo cuando se pretende desarrollar páginas altamente interactivas. Es una herramienta que implementa las técnicas AJAX y su potencial es aprovechado al máximo cuando se desarrolla con Ruby On Rails.

1.12 Fundamentación del marco de trabajo de JavaScript a utilizar

Se seleccionó el marco de trabajo JQuery debido a poseer bibliotecas potentes que facilitan el trabajo con el lenguaje JavaScript además de que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX, y que contiene un set de plugins,[16] componentes para interfaces de usuario y efectos visuales. También facilita la creación de código JavaScript, simplifica el trabajo con Ajax, permite añadir animación a una página, gestiona la interacción con el usuario, permite alterar contenido utilizando pocas líneas

de código, verifica estándares CSS en distintos navegadores. Además es uno de los frameworks más utilizados en la actualidad por lo que en internet se puede obtener toda la documentación suficiente además de opiniones sobre la experiencia de otros programadores sobre su aplicación.

1.13 Frameworks de CSS

1.13.1 Bootstrap

Boostrap es un marco de trabajo desarrollado para simplificar la creación de diseños web. Combina CCS y JavaScript para lograr una interfaz agradable y vistosa. La mayor ventaja es que se puede crear interfaces que se adapten a los distintos navegadores con el apoyo de un marco de trabajo potente con numerosos componentes web que ahorrarán mucho esfuerzo y tiempo.

Características principales de Bootstrap

Bootstrap ofrece una serie de plantillas CSS y ficheros JavaScript que facilitan la integración del marco de trabajo de forma sencilla y potente en los proyectos webs.

- Permite crear interfaces que se adapten a los diferentes navegadores, tanto de escritorio como tabletas y móviles a distintas escalas y resoluciones.
- Se integra perfectamente con las principales bibliotecas de JavaScript, por ejemplo JQuery.
- Ofrece un diseño sólido usando estándares como CSS3/HTML5.
- Es un marco de trabajo ligero que se integra de forma sencilla con el proyecto actual.
- Funciona con todos los navegadores, incluido Internet Explorer usando HTML para que reconozca las etiquetas HTML5.
- Dispone de distintas capas redefinidos con estructuras fijas a 940 píxeles de distintas columnas o diseños fluidos. (Borrillo, 2012)

1.13.2 Fundamentación del marco de trabajo de CCS a utilizar

Se decide ultilizar el marco de trabajo bootstrap debido a que es fácil de integrar con cualquier proyecto web y trae consigo varios componentes que facilitan que se le pueda dar una mejor cara a los sitios web que lo utilicen. Es fácil de utilizar y dentro del propio marco de trabajo se encuentra toda la documentación necesaria para su uso.

1.14 Lenguajes de programación.

Un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo.

Estos lenguajes pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación.

---

[16] Extenciones que se instala aun lenguaje determinado.

Los lenguajes de programación están clasificados en dos grandes grupos en dependencia de donde se implementan con respecto a la arquitectura Cliente/Servidor. Estos son lenguajes del lado del servidor y los lenguajes del lado del cliente. (Aaby, 1996)

Entre los lenguajes del lado del servidor se encuentran: PHP, JSP, ASP, entre otros. Los cuales permiten desarrollar la lógica del negocio dentro del servidor y posibilitan el acceso a las bases de datos y el procesamiento de la información.

Entre los lenguajes del lado del cliente se encuentran: JavaScript, CSS, HTML, entre otros. Estos lenguajes son totalmente independientes del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio.

1.14.1 PHP

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje "código abierto" interpretado de alto nivel, especialmente diseñado para el desarrollo web y el cual puede ser incrustado en páginas HTML. Es usado en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica.

La meta de este lenguaje es permitir a los desarrolladores la generación dinámica de páginas web. Sus características son que es un lenguaje multiplataforma, soporta una gran cantidad de bases de dato, no requiere un mantenimiento muy seguido y es mucho más sencillo de poner al día que otros lenguajes. (PHP, 2001-2012)

1.14.2 JavaScript

JavaScript es un lenguaje de programación que surgió por la necesidad de ampliar las posibilidades del HTML y permite la creación y manipulación de objetos sencillos, y la definición de métodos y propiedades para dichos objetos. JavaScript es un lenguaje interpretado que, al contrario de las aplicaciones normales, que son ejecutadas por el sistema operativo, es ejecutado por el navegador que utilizamos para ver las páginas. Eso hace que se pueda desarrollar aplicaciones de diversos tipos, desde generadores de HTML, comprobadores de formularios, etc., hasta programas que gestionen las capas de una página. Pueden desarrollarse incluso aplicaciones que permitan poder tener capas en una página como si fueran ventanas, y dar la sensación de estar trabajando con una aplicación con interfaz de ventanas.

Entre las acciones típicas que se pueden realizar en JavaScript existen dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, JavaScript permite ejecutar instrucciones como respuesta a las acciones del usuario (eventos), con lo que se pueden crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo. (Pilgrim, 2010).

1.15 Entorno de Desarrollo Integrado (IDE).

Un Entorno Integrado de Desarrollo (IDE) es un sistema que facilita el trabajo del desarrollador de software, integrando sólidamente la edición orientada al lenguaje de programación, la compilación o interpretación, la depuración, las medidas de rendimiento, la incorporación de los fuentes a un sistema de control de fuentes, etc., normalmente de forma modular. (Joaquín Seoane, 2003-2007)

1.15.1 Netbeans

Netbeans es un Entorno de Desarrollo Integrado (IDE) que permite crear aplicaciones de escritorio, aplicaciones web, entre otras. Es una aplicación gratuita y sin restricciones de uso. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java el cual contiene un conjunto de clases escritas para interactuar con las API´s[17] de NetBeans y un archivo especial (**manifiest file**) que se encarga de identificarlo como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. (Corporation, 2012)

Las principales características del NetBeans son:

- Entorno de desarrollo multiplataforma, multilenguaje y amigable tanto para usuarios novatos como para profesionales.
- Disponible en muchos idiomas.
- Es de código abierto.
- Desarrollado por módulos. Brinda la posibilidad de agregar nuevos módulos para aumentar su funcionalidad.
- Cuenta con una amplia documentación y una gran comunidad de usuarios.
- Admite distintos tipos de lenguaje como PHP, C++ y Java.

1.15.2 Eclipse

Eclipse es un entorno de desarrollo integrado, de código abierto y multiplataforma. Es una potente y completa plataforma de Programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. Eclipse puede considerarse actualmente como el IDE referencia en el mundo del software libre, con el inconveniente de funcionar mejor sobre una máquina virtual Java que no es libre. Eclipse es mucho más que un simple IDE, es toda una comunidad de desarrolladores de código libre, dedicados a la implementación de mejoras del entorno.

1.15.3 Zend Studio

Zend Studio es un IDE destinado a desarrolladores profesionales. Es compatible con las plataformas Linux, Mac y Windows e integrado para el lenguaje de programación PHP. Consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos

partes se instalan por separado, la del cliente 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 habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración.

1.16 Fundamentación del Entorno de Desarrollo Integrado (IDE) a utilizar

Se utiliza como entorno de desarrollo el NetBeans 7.2, debido a que es un es un proyecto de código abierto, soporta lenguajes dinámicos como PHP y Java Script, es multiplataforma, posee todas las herramientas necesarias para crear aplicaciones profesionales ya sean de escritorio, empresariales, web, móviles y otras, en diferentes lenguajes de programación además de que también tiene integración con el marco de trabajo Symfony2 del cual posee algunas bibliotecas y plugins que el programador puede agregarle a su aplicación. Además posee integración con la consola de php por lo que puede ejecutar comandos internos del mismo, los cuales son muy útiles en el trabajo con el marco de trabajo Symfony2. El NetBeans permite el completamiento de código, además de insertar funcionalidades definidas como los constructores, métodos get y set entre otros. También se pueden instalar plugins para realizar distintas operaciones internas como deshabilitar el escaneador de proyectos, bibliotecas para comandos de Symfony2 y muchos más.

1.17 Servidor de Aplicación

Un servidor de aplicaciones es un servidor que proporciona servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas.

Se trata de un dispositivo de software que le brinda servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Los beneficios principales de la aplicación de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones. (Corporation, 2012)

1.17.1 Servidor Apache

El servidor HTTP Apache es un servidor web HTTP el cual se encuentra bajo la licencia de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1.

Apache es un servidor potente y flexible. Es altamente configurable y extensible con módulos de terceros, proporciona el código fuente completo y viene con una licencia sin restricciones, se ejecuta en Windows 2000, Netware 5.x o superior, OS / 2, y la mayoría de las versiones de Unix, así como varios sistemas operativos. Es una aplicación que está en constante desarrollo y alienta los comentarios de los usuarios a través de nuevas ideas, informes de errores y parches. (Foundation, 2011)

1.17.2 Servidor JBoss AS

[17] La Interfaz de Programación de Aplicaciones, es un conjunto de funciones residentes en bibliotecas.

JBoss AS es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, JBoss AS puede ser descargado, utilizado, incrustado y distribuido sin restricciones por la licencia.

Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.
- Confiable a nivel de empresa.
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente.
- Servicios del middleware para cualquier objeto de Java.
- Ayuda profesional los 24 días de la semana por parte de la fuente.

1.18 Fundamentación del servidor de aplicación a utilizar

Se seleccionó Apache 2.0 como servidor web y de aplicaciones al mismo tiempo. Es una herramienta modular, con gran cantidad de extensiones para diversas tecnologías y cuenta con abundante documentación. Además de que implementa muchas características de uso frecuente, tales como: la configuración de las páginas protegidas con contraseña con un enorme número de usuarios autorizados, la capacidad de ofrecer respuestas personalizadas a los errores y problemas, entre otras. Es una tecnología libre cuya licencia permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

1.19 Sistemas Gestores de Bases de Datos (SGBD).

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

Algunos de los SGBD que existen en la actualidad son: Oracle, DB2, PostgreSQL, MySQL, MS SQL Server, etc.

Las características de un Sistema Gestor de Base de Datos SGBD son:

• **Abstracción de la información.** Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

• **Independencia.** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

• **Redundancia mínima.** Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. Lo ideal es lograr una redundancia nula; no obstante, en

algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

• **Consistencia.** En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

• **Seguridad.** La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

• **Integridad.** Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

• **Respaldo y recuperación.** Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

• **Control de la concurrencia.** En la mayoría de entornos, lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

1.19.1 PostgreSQL

PostgreSQL es un Sistema Gestor de Bases de Datos derivado de Postgres, desarrollado en la Universidad de California, en el Departamento de Ciencias de la Computación de Berkeley. Es un gestor de bases de datos de código abierto, brinda un control de concurrencia multiversión (MVCC por sus siglas en inglés) que permite trabajar con grandes volúmenes de datos; soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación.

PostgreSQL brinda extensiones de orientación a objetos, permitiendo definir un nuevo tipo de tabla a partir de una previamente definida, haciendo un cómodo acoplamiento con el paradigma de programación seleccionado. Posee además interfaces de programación nativas para diversos lenguajes y plataformas y es capaz de ejecutar procedimientos almacenados en muchos de estos lenguajes, y en su propio lenguaje PL/pgSQL. PostgreSQL es un SGBD multiplataforma muy conocido y usado en entornos de software libre. Es distribuido bajo licencia BSD, la que permite su libre uso, modificación y redistribución con la única restricción de mantener el copyright del software original a sus autores. Es un sistema de bases de datos objeto-relacional con una probada arquitectura que ha ganado gran reputación por su confiabilidad, estabilidad y mantenimiento de la

integridad de los datos. (Postgresql, 2013)

1.19.2 MySQL

MySQL es un sistema gestor de bases de datos relacionales rápido, sólido y flexible. Es idóneo para la creación de bases de datos con acceso desde páginas web dinámicas, así como para la creación de cualquier otra solución que implique el almacenamiento de datos, posibilitando realizar múltiples y rápidas consultas. Está desarrollado en C y C++, facilitando su integración en otras aplicaciones desarrolladas también en esos lenguajes.

Es un sistema cliente/servidor, por lo que permite trabajar como servidor multiusuario y de subprocesamiento múltiple, o sea, cada vez que se crea una conexión con el servidor, el programa servidor establece un proceso para manejar la solicitud del cliente, controlando así el acceso simultáneo de un gran número de usuarios a los datos y asegurando el acceso a usuarios autorizados solamente. Es uno de los sistemas gestores de bases de datos más utilizado en la actualidad, utilizado por grandes corporaciones como Yahoo! Finance, Google, Motorola, entre otras.

1.19.3 Oracle

El SGBD Oracle, fabricado por Oracle Corporation, utiliza la arquitectura cliente/servidor. Ha incorporado en su sistema el modelo objeto-relacional, pero al mismo tiempo garantiza la compatibilidad con el tradicional modelo relacional de datos. Así ofrece un servidor de bases de datos híbrido. Es uno de los más conocidos y ha alcanzado un buen nivel de madurez y de profesionalidad. Se destaca por su soporte de transacciones, estabilidad y escalabilidad. Los tipos objeto de Oracle son tipos de datos definidos por el usuario que permiten modelar entidades complejas del mundo real en una estructura que trata cada entidad como una unidad atómica simple en la base de datos. Su principal desventaja es que es un producto de elevado precio por lo que por lo general se utiliza solamente en empresas muy grandes y multinacionales.

1.20 Fundamentación del Sistema Gestor de Bases de Datos (SGBD) a utilizar

Como Sistema Gestor de Base de datos se seleccionó PostgreSQL en su versión 9.2 ya que es un servidor de base de datos relacional y libre. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y es una herramienta multiplataforma. Como cliente gráfico tiene el PgAdmin III que es distribuido junto a PostgreSQL, el cual también es libre.

1.21 Métricas de validación

La medición es un aspecto fundamental para cualquier disciplina de la ingeniería. Esta nos permite tener una visión de la eficacia del software. Se utilizan medidas para entender mejor los atributos de

los modelos que se crean y para valorar la calidad de los productos de ingeniería o de los sistemas que se construyen. En la Ingeniería de Software se obtiene un conjunto de medidas indirectas que dan lugar a métricas que proporcionan una indicación de la calidad de algún producto de software. Las métricas constituyen un método de evaluación de los productos y procesos de software, las cuales suelen ser aplicadas a muchas organizaciones, procesos y productos. (Edith, 2008)

Por ello se estudiaron varios métodos de validación de manera que fuesen validados todos los artefactos generados durante esta investigación.

Métricas del Modelo de Diseño

Se estudiaron las métricas orientadas a clases dada la importancia que tiene para el equipo de desarrollo conocer la complejidad del proceso de implementación y las características del diseño.

Dentro de las métricas orientadas a clases se encuentra la métrica Tamaño de clase (TC) en la que se tienen en cuenta los siguientes aspectos:

- El número total de operaciones (de instancia heredada y privada) que están encapsuladas dentro de la clase.
- El número de atributos (de instancia heredada y privada) que están encapsulados por la clase.
- Promedio general de los dos anteriores para el sistema completo. (Edith, 2008)

Estos valores se suman de acuerdo a la clase que se analiza y el resultado determina el TC de cada clase. Cuanto menos sean los valores para el TC más probables será que las clases dentro del sistema puedan reutilizarse ampliamente.

1.22 Pruebas

Las pruebas presentan una interesante anomalía para el ingeniero del software. Durante las fases anteriores de definición y de desarrollo, el ingeniero intenta construir el software partiendo de un concepto abstracto y llegando a una implementación tangible. A continuación, llegan las pruebas. El ingeniero crea una serie de casos de prueba que intentan demoler el software construido. La prueba demuestra hasta qué punto las funciones del software parecen funcionar de acuerdo con las especificaciones y parecen alcanzarse los requisitos de rendimiento. Además, los datos que se van recogiendo a medida que se llevan a cabo las pruebas proporcionan una buena indicación de la fiabilidad del software y, de alguna manera, indican la calidad del software como un todo. (Pressman, R.S, 2005)

1.22.1 Pruebas de Caja Blanca

Las pruebas de caja blanca (también conocidas como pruebas de caja de cristal o pruebas estructurales) se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. Aunque las pruebas de caja blanca son aplicables a varios niveles —unidad, integración y sistema—, habitualmente se aplican a las unidades de software. Su objetivo es comprobar los flujos de ejecución dentro de cada unidad (función, clase, módulo, etc.)

pero también pueden probar los flujos entre unidades durante la integración, e incluso entre subsistemas, durante las pruebas de sistema.

Las pruebas de caja blanca se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas (integración).Mediante los métodos de prueba de caja blanca, se pueden obtener casos de prueba que:

- Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo. Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejecuten todos los ciclos en sus límites y con sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Las principales técnicas de diseño de pruebas de caja blanca son:

- Pruebas de la estructura de control
- Pruebas de caminos básicos (Pressman, R.S, 2005)

1.22.2 Pruebas de Caja Negra

Las pruebas de caja negra, también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Las pruebas de caja negra no son una alternativa a las técnicas de pruebas de caja blanca, sino que se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca. Las pruebas de caja negra intentan encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación.

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

- Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones. (Pressman, R.S, 2005)

1.23 Conclusiones parciales

Luego de haberse realizado un estudio sobre los sistemas existentes relacionados con las secciones sindicales se concluye que ninguno cumple con las características deseadas por lo que es necesario desarrollar un nuevo sistema capaz de automatizar todos los procesos que se llevan a cabo dentro de estas. El estudio realizado sobre metodologías de desarrollo permitió arribar a la conclusión de que XP, debido a sus características, es la más apropiada para guiar el desarrollo del sistema a implementar. El estudio realizado sobre herramientas y tecnologías para el desarrollo de software, permitió seleccionar: el marco de trabajo Symfony 2 por su potencia, rendimiento,

funciones y solidez; PHP como lenguaje de programación del lado del servidor por ser el lenguaje que tiene predefinido Symfony 2 y uno de los mejores en la generación dinámica de páginas web; el sistema de bases de datos PostgreSQL y el Entorno de Desarrollo Integrado Netbeans por su facilidad de integración con el marco de trabajo Symfony 2.Teniendo en cuenta las estructuras de la Universidad de las Ciencias Informáticas en cuanto a las tecnologías que se utilizan, se puede afirmar que la selección realizada es la más óptima para proporcionar una solución eficiente al problema planteado.

Capítulo 2: Solución Propuesta.

Introducción

En el presente capítulo se realizará un análisis de las principales características del sistema que se desarrolló Se exponen los artefactos generados propios de la metodología de desarrollo utilizada para la implementación de la solución que se propone tales como: historias de usuario, tareas de ingeniería y tarjetas CRC. Además también se describe la arquitectura del sistema, los patrones de diseño que fueron utilizados y el modelo de datos que fue generado para la solución propuesta.

2.1 Exploración

En esta primera fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2.1.1 Historias de usuario

Entre los artefactos que define la metodología seleccionada se encuentran las historias de usuario que son utilizadas para especificar las funcionalidades que brindará el sistema. Cada historia de usuario es una representación de un requerimiento de software escrito en una o dos frases utilizando el lenguaje común del usuario. Representan una forma rápida de administrar los requerimientos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Además son la base para realizar las pruebas de aceptación, así como la estimación y planificación del proyecto.

Como resultado del trabajo realizado durante la fase de exploración se identificaron un total de 25 historias de usuario, a continuación se muestran algunas de ellas:

Tabla 1: Historia de Usuario #1.

Historia de usuario	
Número: 1	**Nombre:** Registrar usuario
Usuario: Administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Alto (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir que se registren los usuarios: Secretario General, Financiero y Activista. El primero será registrado por el Administrador del Sistema y debe asociarse a una sección sindical, y los demás serán registrados por el propio Secretario General.	
Observaciones: Existe un Administrador general del sistema.	

Tabla 3: Historia de Usuario #3.

Historia de usuario	
Número: 3	**Nombre:** Registrar datos de Sección sindical.
Usuario: Administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Alto (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir registrar una nueva sección sindical con los siguientes datos: Nombre, Facultad. Esta funcionalidad debe ser accesible sólo para el Administrador del Sistema.	
Observaciones: Existe un Administrador general del sistema.	

Tabla 5: Historia de Usuario #5.

Historia de usuario	
Número: 5	**Nombre:** Conformar Nuevo Potencial.
Usuario: Secretario General	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Alto (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir que el secretario general conforme el potencial para el año siguiente. Inicialmente se deberá mostrar a todos los afiliados del año precedente y las opciones Modificar Salario, Agregar Trabajador, Aceptar y Cancelar. La opción Agregar Trabajador brindará la posibilidad de agregar un nuevo trabajador al sistema, o de afiliar a alguno que no estuviese afiliado por alguna razón. La opción Aceptar indicará que se terminó la elaboración del potencial.	
Observaciones: Existe un Secretario General por cada sección sindical.	

2.2 Planificación

En esta fase los clientes establecen la prioridad de las historias de usuario en correspondencia con las necesidades más inmediatas para luego asignarlas, por el orden de relevancia, a las iteraciones planificadas. A partir de las historias se realiza la estimación del esfuerzo que se requiere por parte del equipo para su posterior implementación. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. El tiempo de implementación de una historia de usuario generalmente es de uno a tres puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

2.2.1 Estimación de esfuerzos por Historias de Usuario

Para el desarrollo satisfactorio de la solución propuesta, se realizó una estimación de esfuerzo para cada una de las historias de usuario, arrojando los siguientes resultados:

Tabla # 26: Puntos de estimación por historias de usuario

No	Historias de usuario	Puntos de estimación
1	Registrar usuario	2
2	Autenticar Usuario	2
3	Registrar datos de una sección sindical	2
4	Modificar datos de una sección sindical	2
5	Conformar Nuevo Potencial	1
6	Registrar datos del trabajador	1
7	Modificar salario del trabajador	1
8	Buscar Trabajador	2
9	Dar Alta a un trabajador	1
10	Dar Baja a un trabajador	1
11	Registrar acuerdo	2
12	Registrar cumplimiento de acuerdo	1
13	Obtener reporte de acuerdos	2
14	Registrar el mes de compromiso de pago de la MTT	2
15	Registrar el mes en que se efectuó el pago de la MTT	2
16	Registrar meses de pago de la cuota sindical	2
17	Obtener reporte de los atrasados en el pago de la cuota sindical	1
18	Obtener reporte de los atrasados en el pago de la MTT	1
19	Obtener un reporte de los afiliados que pagaron el año completo	1
20	Obtener un reporte sobre el estado actual de las finanzas	1
21	Obtener un reporte de los afiliados que han causado alta	1
22	Obtener un reporte de los afiliados que han causado baja	1
23	Obtener un reporte de los atrasados en el pago de las cuotas en el año	2
24	Registrar trabajadores destacados por trimestre	1
25	Registrar trabajadores vanguardias	1

2.2.2 Plan de Iteraciones

En el plan de liberaciones se establece cuántas iteraciones serán necesarias realizar sobre el sistema para su consecución. Sin embargo, se precisa establecer el contenido de trabajo para todas y cada una de ellas y es aquí donde se establece el plan de iteraciones, regulando la cantidad de historias de usuario a implementar dentro del rango establecido por la estimación efectuada. Tomando como referencia los aspectos antes tratados la aplicación que se pretende construir se desarrollará en 3 iteraciones, explicadas más detalladamente a continuación:

Iteración 1

La iteración tiene como finalidad implementar las historias de usuario que se consideraron más necesarias atendiendo a su relevancia e impacto para el negocio. Se da respuesta a las funcionalidades de registrar sección sindical, registrar trabajador, conformar potencial, además de modificar y eliminar los datos insertados.

Iteración 2

En esta iteración se realizan todas las historias de usuarios relacionadas con la gestión de pago de la cuota sindical y el aporte a las MTT. Además de la gestión del estado de las finanzas de las secciones sindicales.

Iteración 3

En esta iteración se realizan todas las historias de usuario relacionadas con los reportes que se generan en el sistema en cuanto al proceder de los trabajadores de las secciones sindicales. Además también se realizan las historias de usuario en las que se gestionan los procesos de selección y estimulación de los trabajadores destacados.

A modo de resumen se presenta la siguiente tabla que muestra las 3 iteraciones analizadas previamente con las historias de usuario que incluyen y su duración:

Tabla # 27: Plan de duración de las iteraciones.

Iteraciones	Historias de usuario	Duración
Iteración 1	Gestionar usuario	6 semanas
	Gestionar sección sindical	
	Conformar Nuevo Potencial	
	Gestionar trabajador	
	Gestionar acuerdo	
	Reporte de acuerdo	
Iteración 2	Registrar pago de MTT	5 semanas
	Registrar pago de la cuota sindical	
	Reporte de atrasados en la MTT	
	Reporte de atrasados en la cuota sindical	
	Reporte del estado de las finanzas	
Iteración 3	Reporte de alta/baja de trabajadores	3 semanas
	Registrar trabajadores destacados	
	Reporte de trabajadores vanguardias	

2.3 Diseño

XP establece prácticas especializadas que inciden directamente en la realización del diseño para lograr un sistema robusto y reutilizable tratando de mantener su simplicidad, es decir, crear un diseño evolutivo que se va mejorando incrementalmente y que permite hacer entregas pequeñas y

frecuentes de valor para el cliente. A la hora de darle cumplimiento a la actividad de diseñar, XP no especifica ninguna técnica de modelado. Pueden utilizarse indistintamente sencillos esquemas en una pizarra, diagramas de clases utilizando UML o tarjetas CRC (Clase, Responsabilidad y Colaboración) siempre que sean útiles, tributen a la comprensión y no requieran mucho tiempo en su creación.

Como base para el desarrollo de la aplicación propuesta se utilizó el marco de trabajo Symfony 2 que implementa, como lo hacen muchos otros, el patrón arquitectónico MVC (Modelo Vista Controlador) que obliga a organizar el código de acuerdo a sus convenciones.

Symfony 2 proporciona una estructura en forma de árbol de archivos para organizar de forma lógica todos esos contenidos, además de ser consistente con la arquitectura MVC utilizada y con la agrupación de los directorios de la forma proyecto / src / bundle. Cada vez que se crea un nuevo proyecto, aplicación o módulo, se genera de forma automática la parte correspondiente de esa estructura. Además, la estructura se puede personalizar completamente, para reorganizar los archivos y directorios o para cumplir con las exigencias de organización de un cliente.

2.3.1Tarjetas CRC

La metodología XP en lugar de utilizar diagramas para desarrollar modelos representa las clases mediante tarjetas. Las tarjetas CRC (Clase, Responsabilidad y Colaboración) ayudan al equipo a definir actividades durante el diseño del sistema. Cada tarjeta representa una clase en la programación orientada a objetos y define sus responsabilidades (lo que ha de hacer) y las colaboraciones con las otras clases (cómo se comunica con ellas).

Como resultado del trabajo realizado durante la fase de diseño se obtuvieron un total de 35 tarjetas CRC, a continuación se muestran algunas de ellas:

Tabla 28: Tarjeta CRC #1.

Clase: dAcuerdo	
Responsabilidades: Obtener la denominación del acuerdo Modificar la denominación del acuerdo Obtener la fecha de cumplimiento Modificar la fecha de cumplimiento Obtener la causa de incumplimiento Modificar la causa de incumplimiento Obtener la estado del acuerdo Modificar la estado del acuerdo Obtener el responsable del acuerdo Modificar el responsable del acuerdo	**Colaboradores:** • dTrabajador • dSeccionSindical • nEstadoAcuerdo

Tabla 29: Tarjeta CRC #2.

Clase: GestAcuerdoController	
Responsabilidades: Crear el acuerdo Listar los acuerdos Actualizar el estado del acuerdo Eliminar el acuerdo	**Colaboradores:** • dAcuerdo • DAcuerdoRepository

Tabla 30: Tarjeta CRC #3.

Clase: DAcuerdoRepository	
Responsabilidades: Obtener los acuerdos de una sección sindical	**Colaboradores:** • dAcuerdo

Tabla 31: Tarjeta CRC #4.

Clase: AcuerdosGestor	
Responsabilidades: Obtener los acuerdos de una sección sindical en un rango de fecha y el estado del acuerdo	**Colaboradores:** • dAcuerdo

2.3.2 El subdirectorio Entity

El subdirectorio Entity, que se encuentra en la raíz del Bundle SindicalBundle según la estructura que brinda Symfony2, guarda el modelo de objetos del proyecto, en otras palabras, la lógica de negocio de la aplicación. Además también contiene un directorio llamado EP donde se encuentras las entidades de presentación que se manipulan a fin de crear entidades con algunos de los atributos de las clases persistentes con el objetivo de representar campos de una tabla que será mostrada en una interfaz determinada.

Las clases persistentes se obtienen de acuerdo al dominio del problema analizado y se corresponden por tanto con lo obtenido durante las sesiones en las que se construyeron las tarjetas CRC.

2.3.3 Modelo de datos

Un modelo de datos es la representación abstracta de los datos en un sistema gestor de base de datos. Básicamente el modelo de datos está formado por tres elementos fundamentales que son:

- ✓ Objetos (entidades que existen y se manipulan).
- ✓ Atributos (Características básicas de estos objetos).
- ✓ Relaciones (forma en que se enlazan los distintos objetos entre sí).

El modelo de datos correspondiente al sistema consta de un total de 17 tablas, de ellas 7 nomencladores y las 10 restantes para el almacenamiento de la información manipulada por este. Para su construcción se tuvo en cuenta todos los datos que deben persistir en el sistema y en los nomencladores se agruparon los estándares predeterminados del negocio del proceso.

A continuación se muestra el modelo de dato que se obtuvo a partir de las clases generadas en las tarjetas CRC:

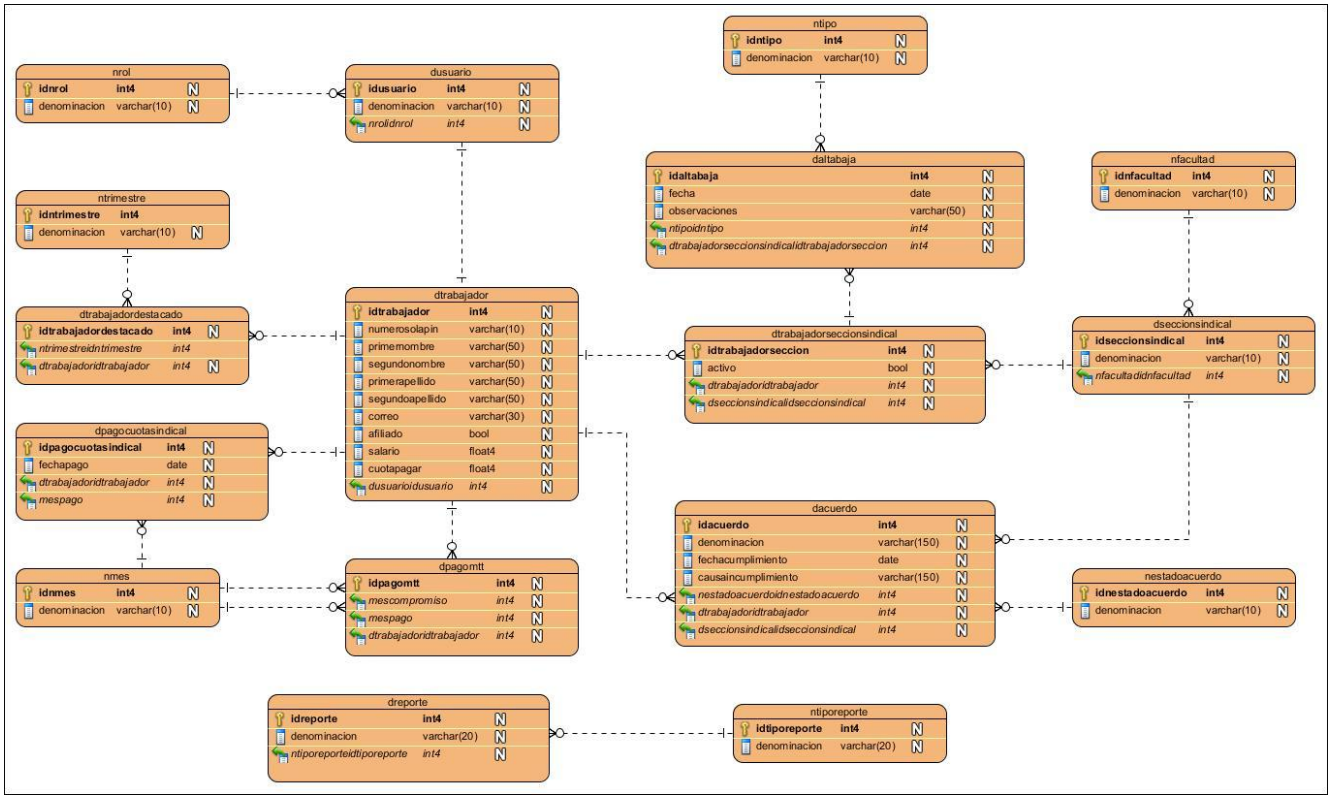

Figura 2. Representación del Modelo de datos

El modelo de datos mostrado anteriormente refleja los conceptos, las relaciones y las reglas del dominio estudiado, con el objetivo de darle respuesta al problema afrontado por el equipo.

2.4 Codificación

En esta fase las historias de usuario elegidas se descomponen en tareas de programación o ingeniería, escritas en lenguaje técnico, que a su vez son convertidas posteriormente a código. La programación en parejas, refactorización, la integración continua y la compilación de diez minutos son un subconjunto de las prácticas que sirven de guía durante esta crucial actividad. XP en este sentido no define artefactos específicos, pero se decide utilizar el modelo de datos debido a su alto grado de importancia, basado en que a partir de él se generan las clases por parte del Object-Relational Mapping (ORM), y al relativamente poco esfuerzo requerido en su construcción.

2.4.1 Tareas de Ingeniería

Una vez identificadas las historias de usuario los programadores proceden a descomponer cada una en tareas específicas, las denominadas tareas de programación que están escritas técnicamente y que darán solución a la historia correspondiente.

Como resultado del trabajo realizado durante la fase de codificación se obtuvieron un total de 25 tareas de ingeniería, a continuación se muestran algunas de ellas:

Tabla 32: Tarea de Ingeniería #1.

	Tarea de Ingeniería
Número Tarea: 1	**Historia de Usuario (Nro.1):** Registrar usuario

Nombre Tarea: Crear interfaz registrar usuario	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora)	**Puntos Estimados:** 1 semana
Fecha Inicio: 20/01/13	**Fecha Fin:** 25/01/13
Programador Responsable: Maricet Moreta Fernández	
Descripción: Esta tarea facilita la creación de una interfaz para la gestión de todos los usuarios en el sistema. Una vez que el administrador del sistema selecciona la opción Registrar usuario, aparecerá la interfaz gestionar usuario con los campos correspondientes para registrar un usuario en el sistema.	

Tabla 33: Tarea de Ingeniería #2.

	Tarea de Ingeniería
Número Tarea: 3	**Historia de Usuario (Nro.3):** Registrar datos de una sección sindical
Nombre Tarea: Crear interfaz gestionar sección sindical	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora)	**Puntos Estimados:** 1 semana
Fecha Inicio: 25/01/13	**Fecha Fin:** 30/01/13
Programador Responsable: Maricet Moreta Fernández	
Descripción: Esta tarea facilita la creación de una interfaz para la gestión de secciones sindicales. Una vez que el administrador del sistema selecciona la opción Sección Sindical, aparecerá la interfaz de Secciones Sindicales con la opción correspondiente para registrar una sección sindical.	

Tabla 34: Tarea de Ingeniería #3.

	Tarea de Ingeniería
Número Tarea: 5	**Historia de Usuario (Nro.5):** Conformar Nuevo Potencial.
Nombre Tarea: Crear interfaz gestionar trabajador	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora)	**Puntos Estimados:** 1 semana
Fecha Inicio: 1/02/13	**Fecha Fin:** 7/02/13
Programador Responsable: Maricet Moreta Fernández	
Descripción: Esta tarea facilita la creación de una interfaz para la gestión de trabajadores. Una vez que el secretario general selecciona la opción Nuevo Potencial, aparecerá la interfaz de Nuevo Potencial con la opción correspondiente para conformar el nuevo potencial de trabajadores de una sección sindical.	

2.5 Patrones de diseño utilizados

Un patrón de diseño abarca una idea completa dentro de un programa, y por lo tanto puede también abarcar las fases de análisis y diseño de alto nivel. Sin embargo, dado que un patrón a menudo tiene una implementación directa en código, podría no mostrarse hasta el diseño de bajo nivel o la implementación. (Gamma, 1995)

Lograr un diseño simple pero a la vez robusto requiere en gran medida del empleo de buenas prácticas. Se tuvieron en cuenta al modelar el sistema los conceptos de bajo acoplamiento para evitar las dependencias excesivas, y la alta cohesión tratando de que cada clase realice labores únicas y bien relacionadas, siempre con la intención de lograr un punto de equilibrio entre ambos. Además, el patrón experto para la realización de las tareas, siendo responsabilidad de la clase que cuenta con los datos involucrados, y el controlador para manejar los eventos del sistema.

A continuación se detallan los patrones utilizados durante el desarrollo del sistema:

2.5.1 Inyección de Dependencias

En informática, Inyección de Dependencias es un patrón de diseño orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase quien cree el objeto. Este patrón es implementado mediante un "contenedor DI[18]" y objetos planos o simples. El contenedor inyecta a cada objeto los objetos necesarios según las relaciones plasmadas en un fichero de configuración. (Griffin Caprio, 2005)

Típicamente este contenedor es implementado por un marco de trabajo externo a la aplicación por lo cual en la aplicación también se utilizará inversión de control al ser el contenedor (almacenado en una biblioteca) quien invoque el código de la aplicación. Esta es la razón por la que los términos de inversión de control e inyección de dependencias se confunden habitualmente entre sí. (Martin Flower, 2011**)**

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 de la anteriormente mencionada se pueda obtener una instancia del contenedor de dependencias por lo que se puede instanciar cualquier servicio existente haciendo uso de la función get ().

[18] Contenedor de inyección de dependencias

```php
namespace REGSIND\SindicalBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use REGSIND\SindicalBundle\Entity\Acuerdo;
use REGSIND\SindicalBundle\Form\AcuerdoType;

/*
 * Registrar Acuerdo controller.
 */
class RegistarAcuerdoController extends Controller
{
    private function getGestor() {
        if (!$this->container->has('sind.Acuerdo')) {
            throw new \LogicException('Este servicio no esta
     registrado en la aplicacion');
        }

        return $this->container->get('sind.Acuerdo');
    }

    public function indexAction() {

        return $this->render('SindicalBundle:Default:index.html.twig');
    }
```

Figura 3. Representación de la inyección de dependencias

2.5.2 Patrones GRASP

Controlador: Este patrón se puede evidenciar dentro del sistema desarrollado en las clases controladoras y en el controlador frontal que posee Symfony2, el cual maneja todas las peticiones que se realizan por el usuario. Dicho contenedor se encuentra ubicado en el directorio /web/app.php. Además se crean clases controladoras para cada caso de uso del sistema, ubicadas en SistemaSindical/src/REGSIND/SindicalBundel/Controller.

Experto: Experto en información consiste en que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene encapsulada, es decir, disminuye el acoplamiento. En el sistema desarrollado se aprecia este patrón en las clases **repository** las cuales se especializan en contener información propia. Además se puede apreciar en las clases **gestores** ya que se encargan de gestionar la información necesaria de las entidades.

```php
<?php

/**
 * Description of AcuerdoGtr
 *
 * @author Maricet Moreta
 */
namespace REGSIND\SindicalBundle\Negocio;


use Base\ArquitecturaBundle\Negocio\BaseGtr;


class AcuerdoGtr extends BaseGtr {

   public function registrarAcuerdo($acuerdo)
    {
        return $this->getEm()->persist($acuerdo);
    }
}

?>
```

Figura 4. Representación del gestor AcuerdoGtr

Alta cohesión: En el caso del sistema propuesto utilizando el patrón alta cohesión se le asigna a cada clase las responsabilidades que le corresponde y se establecen las condiciones para que cada clase colabore con las demás en la resolución de tareas que las implican a todas y que no son capaces de resolver por sí solas.

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte también al bajo acoplamiento lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. (Alcalá)

En el caso del sistema desarrollado este patrón fue utilizado en las clases controladoras en las cuales se crean objetos de las clases entidades para su manipulación.

Bajo acoplamiento: Las clases que implementan la lógica de negocio y de acceso a datos se encuentran en el modelo, estas clases no tienen asociaciones con las de la vista o el controlador por lo que la dependencia en este caso es baja, demostrándose así el uso de este patrón (Gamma et al., 1995).

En la solución propuesta se puede apreciar que las clases entidades son las más reutilizadas puesto que son requeridas para varias de las funcionalidades dentro de un mismo subproceso.

2.5.3 Patrones GOF

Patrón Decorador: En el sistema este patrón permite añadir dinámicamente nuevas

responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender funcionalidad. Se aplica con la intención de proporcionar una forma flexible de introducir o eliminar funcionalidad de un componente sin modificar su apariencia externa o su función.

Factory Method: El marco de trabajo Symfony2 define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos. Las clases delegan la responsabilidad en una de entre varias clases auxiliares y se quiere localizar qué subclase concreta es en la que se auxilia, y se quiere localizar qué subclase concreta es en la que se delega.

2.6 Arquitectura del sistema

El desarrollo de la presente aplicación responde a una arquitectura basada en capas la cual está compuesta por la capa de presentación, capa de negocio y capa acceso a datos. Los sistemas o arquitecturas en capas brindan entre otros beneficios un cierto aislamiento ya que en caso de ocurrir un error o de que sea necesario actualizar algún elemento solo se realizaría el cambio en la capa correspondiente y el resto se mantiene intacto. Como patrón arquitectónico se adopta el Modelo-Vista-Controlador (MVC) por ser uno de los más usados en el desarrollo de aplicaciones web, además viene integrado al Symfony2 en las capas de presentación y negocio. En la siguiente imagen se puede apreciar la estructura en capas y la ubicación de los elementos del patrón arquitectónico utilizado.

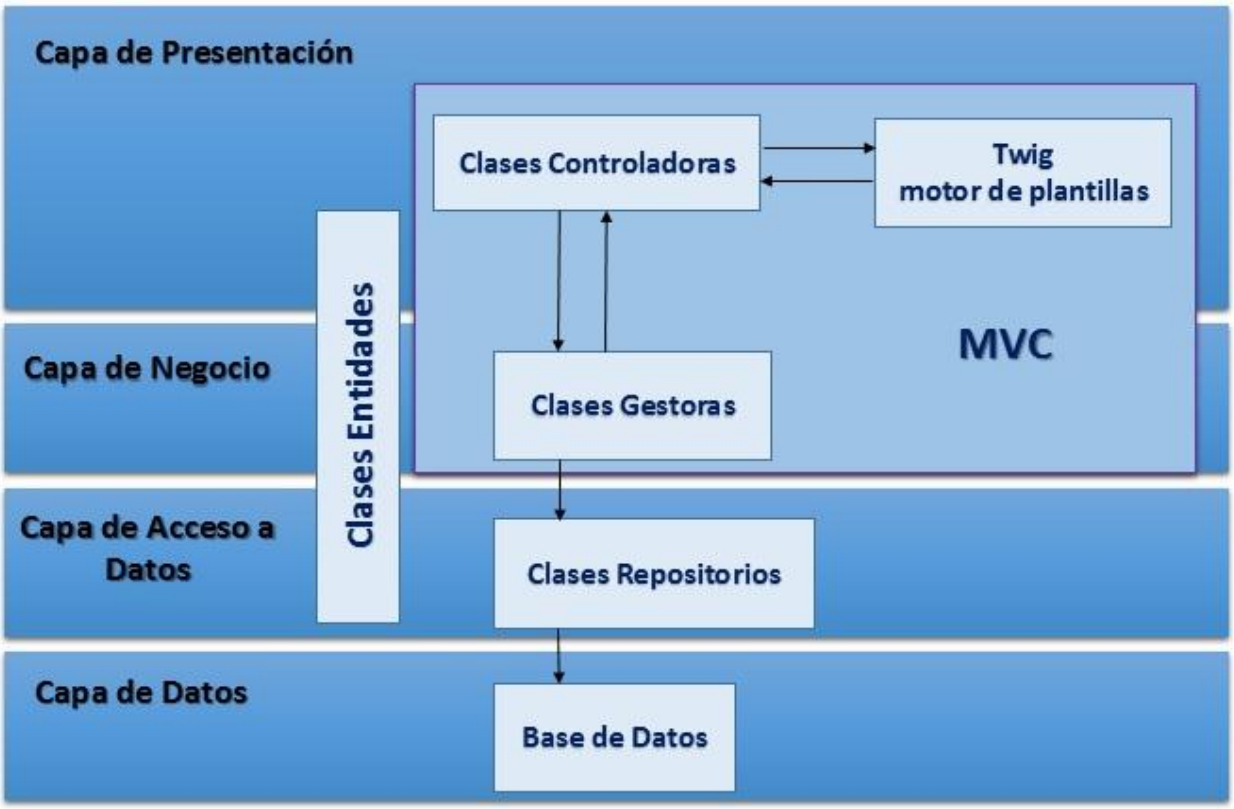

Figura 5. Esquema de la arquitectura en capas

2.7 Conclusiones parciales

En este capítulo, quedaron definidas las características para el desarrollo de la solución. Además, fueron concebidas las historias de usuario para lograr una planificación en el desarrollo de la aplicación. Las tarjetas CRC fueron elaboradas para las clases más importantes de la aplicación, con el objetivo de plasmar sus responsabilidades y colaboradores. Además, se definieron los patrones de diseño utilizados para lograr una correcta estructuración del sistema.

Capítulo 3: Validación de resultados.

Introducción

En el presente capítulo se muestran las pruebas realizadas al conjunto de artefactos que se obtuvieron del trabajo realizado en la planificación y el diseño por parte del equipo de analistas. Se aplicarán métricas específicas para la validación de estos artefactos además de pruebas unitarias y de aceptación a la aplicación desarrollada tal como lo define la metodología utilizada.

3.1 Métricas

Para comprobar la efectividad de las actividades y validar los artefactos se utilizarán las Métricas Orientadas a Clases debido a la importancia que tiene conocer el acoplamiento entre las clases, la reutilización de las mismas, así como la cantidad de pruebas y complejidad de mantenimiento.

3.1.1 Tamaño de clase (TC)

Para evaluar las métricas son necesarios los umbrales. En este caso las clases se clasifican en tres grupos según su tamaño, los que se representan en la siguiente tabla junto con los umbrales seleccionados para su clasificación. (Edith, 2008)

Tabla 35: Clasificación de los umbrales para TC.

Clasificación	Valores de los umbrales
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Nota: Cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes en el sistema se puedan reutilizar ampliamente.

Tabla 36: Tamaño de Clases.

No	Clases	No.Atributos	No.Operaciones
1	GestionarUsuarioController	0	5
2	GestionarTrabajadorController	0	12
3	GestionarAcuerdoController	0	7
4	GestionarSeccionSindicalController	0	7
5	IndexController	0	3
6	BaseController	0	17
7	DefaultController	0	2
8	EntityRepository	3	14

9	Dusuario	3	6
10	Dtrabajador	11	22
11	Dacuerdo	7	14
12	Dseccionsindical	3	6
13	Dtrabajadorseccionsindical	4	8
14	Dtrabajadordestacado	3	6
15	Daltabaja	5	10
16	Dpagomtt	4	8
17	Dpagocuotasindical	3	6
18	Nrol	2	4
19	Ntipo	2	4
20	Ntrimestre	2	4
21	Nfacultad	2	4
22	Nmes	2	4
23	Nestadoacuerdo	2	4
24	Dreporte	3	6
25	Ntiporeporte	2	4
	Total	63	187

Se implementaron total de 25 clases para un promedio de 2.52 atributos y 7.48 de operaciones.

Para evaluar las métricas son necesarios los valores de los umbrales. Existen tres umbrales para esta métrica, los cuales quedan con los datos mostrados a continuación:

Tabla 37: Cantidad de clases por clasificación.

Clasificación	Valores de los umbrales	Cantidad de clases
Pequeño	<=20	23
Medio	>20 y <=30	1
Grande	>30	1

De acuerdo a los umbrales mostrados en la tabla el 92 % de las clases son pequeñas, lo cual significa que estas clases se pueden reutilizar ampliamente. Al analizar los resultados obtenidos para

cada uno de los atributos de calidad para esta métrica, se evidencia que el 68 % de las clases poseen una responsabilidad baja, un 24 % media y el 8 % alta. Por otra parte el 68 % de las clases presentan una complejidad baja, el 24 % media y un 8 % alta. Estos atributos de calidad son directamente proporcionales. En el caso de la reutilización, se obtiene que el 68 % de las clases poseen una alta reutilización, el 24 % media y un 8 % baja. En el siguiente gráfico se puede observar los resultados de la métrica.

Figura 6. Resultados de la métrica TC.

3.1.2 Relaciones entre clases (RC)

Esta métrica evalúa la cantidad de relaciones de uso que existe entre las distintas clases que forman el diseño propuesto. Se aplica a las mismas clases en las que fue aplicada la métrica TC. Los aspectos de calidad que se miden son: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas.

Tabla 38: Relaciones entre clases.

No	Clases	Relaciones de uso
1	GestionarUsuarioController	3
2	GestionarTrabajadorController	3
3	GestionarAcuerdoController	4
4	GestionarSeccionSindicalController	2
5	IndexController	1

6	BaseController	0
7	DefaultController	1
8	EntityRepository	0
9	Dusuario	2
10	Dtrabajador	6
11	Dacuerdo	3
12	Dseccionsindical	3
13	Dtrabajadorseccionsindical	3
14	Dtrabajadordestacado	2
15	Daltabaja	2
16	Dpagomtt	3
17	Dpagocuotasindical	1
18	Nrol	1
19	Ntipo	1
20	Ntrimestre	1
21	Nfacultad	1
22	Nmes	3
23	Nestadoacuerdo	1
24	Dreporte	1
25	Ntiporeporte	1

Para medir el acoplamiento según los resultados de esta métrica, se plantean los siguientes valores.

Tabla 39: Acoplamiento.

Categoría	Relaciones de uso	Cantidad
Ninguno	0	2
Bajo	1	10
Medio	2	4
Alto	>2	9

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 1,96.

Tabla 40: Cantidad de pruebas y Complejidad de mantenimiento

Categoría	Criterio	Cantidad de clases
Baja	<=Promedio	12
Media	>Promedio<=2*Promedio	11
Alta	>2*Promedio	2

Tabla 41: Reutilización.

Categoría	Criterio	Cantidad de clases
Baja	>2*Promedio	2
Media	>Promedio<=2*Promedio	11
Alta	<=Promedio	12

Al analizar los resultados obtenidos para cada uno de los atributos de calidad para esta métrica, se evidencia que el 40 % de las clases poseen un bajo acoplamiento, un 16 % media y el 36 % alta. Por otra parte el 48 % de las clases presentan una complejidad de mantenimiento baja, el 44 % media y un 8 % alta. En el caso de los valores de cantidad de pruebas se obtiene que el 48 % de las clases poseen una cantidad de pruebas baja, el 44 % media y un 8 % alta. En cuanto a la reutilización se puede apreciar que el 48 % de las clases poseen una alta reutilización, el 44 % media y un 8 % baja. En el siguiente gráfico se puede observar los resultados de la métrica.

Figura 7. Resultados de la métrica RC.

3.2 Pruebas

La metodología ágil XP divide las pruebas en dos grupos: pruebas unitarias y pruebas de aceptación. Las pruebas unitarias son desarrolladas por los programadores y se encargan de verificar el código automáticamente y las pruebas de aceptación están destinadas a verificar que al final de cada iteración las historias de usuario cumplen con la funcionalidad asignada y satisfagan las necesidades del cliente.

3.2.1 Pruebas de caja blanca

El método de pruebas de caja blanca precisa del acceso al código del programa de modo que se pueda comprobar su lógica interna. Al sistema desarrollado se le aplicaron tanto pruebas unitarias como de aceptación, tal y como propone XP.

3.2.1.1 Prueba del camino básico

Para la aplicación de las pruebas de caja blanca se hizo uso de la técnica camino básico. El método del camino básico permite obtener una medida de la complejidad de un diseño procedimental, y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución, diseñando casos de prueba que garanticen que cada camino se ejecuta al menos una vez.

Se toma de ejemplo el método CrearAcuerdoAction, perteneciente a la clase GestAcuerdoController como base para realizar la técnica del camino básico.

El método CrearAcuerdoAction consiste en mostrar la página GestAcuerdo.html.twig enviándole varios datos gestionados para que sean procesados en dicha página. Para realizar esta operación el método recibe datos como el identificador de la sección sindical en la que se creará el acuerdo para luego con ese valor poder crear un acuerdo. Luego se pregunta si la petición fue realizada por el método POST o GET. Luego se obtienen los datos enviados desde el formulario y se crea un objeto de tipo Dacuerdo y se guarda en base de datos y se le da una respuesta a la página que envió los datos por Ajax.

Partiendo del fragmento de código tomado se obtuvo el siguiente grafo de flujo.

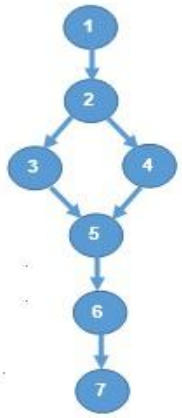

Figura 8. Representación del grafo de flujo del camino básico

Luego se calculó la complejidad ciclomática V (G), obteniendo como resultado lo siguiente:

V (G) = Aristas (A) – Nodos (N) + 2

V (G) = 7 – 7 + 2

V (G) = 2

El valor V (G) expresa la cantidad de caminos linealmente independientes de la estructura de control del programa, por lo que se definen los siguientes 2 caminos:

Camino básico 1: 1-> 2 -> 3 -> 5 -> 6 -> 7

Camino básico 2: 1-> 2 -> 4 -> 5 -> 6-> 7

Cada camino independiente es un caso de prueba a realizar, de forma que los datos señalados causen que se visiten las sentencias vinculadas a cada nodo de camino. En el caso anterior se calcularon dos caminos básicos por tanto surge la necesidad de hacer igual número de casos de prueba, para aplicar las pruebas a este método.

Tabla 42: caso de prueba de caja blanca para el camino básico 1.

Entrada	Se obtienen los valores iniciales, se necesita que la petición haya sido realizada por el método *POST,* tiene que existir al menos una sección sindical para que se agregue el nuevo acuerdo y no existan errores en los datos de acuerdo a las reglas del negocio.
Resultados esperados	Se envían satisfactoriamente todos los datos hacia la página *GestAcuerdo.html.twig* y se renderiza dicha página
Condiciones	POST' == $request->getMethod(), count($seccionSindical >= 1)

Una vez ejecutadas las pruebas de caja blanca a través de la técnica del camino básico al método CrearAcuerdoAction, en una primera iteración se detectaron un total de 2 errores en la lógica del código. Estos errores detectados fueron solucionados y se procedió a aplicar una segunda iteración donde no se detectó error alguno. De esta manera se comprobó que el flujo de trabajo de las funciones implicadas en el método es correcto y cumple con las condiciones planteadas anteriormente.

3.2.2 Pruebas de caja negra

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos de prueba en que el módulo no se atiene a su especificación. Esto se refiere a que se llevan a cabo para verificar el ajuste del sistema con los requerimientos determinados. Además se enfocan especialmente en los módulos que se relacionan con la interfaz de usuario. No requieren el conocimiento de la estructura interna del programa para su puesta en marcha. (Schenone, 2004)

3.2.2.1 Pruebas de aceptación

Cada historia de usuario está asociada a una prueba de aceptación, conocidas también como pruebas funcionales, las mismas se realizan en esta etapa del proyecto y en ellas se describen las

posibles formas de utilización del software. Las pruebas funcionales no solo validan la transformación de una entrada en una salida, sino que validan una característica completa.

En estos documentos de prueba se indican las posibles respuestas que tiene el software en la utilización de cada funcionalidad, así como los posibles mensajes de error, información o de aceptación que emite el software cuando se utiliza dicha funcionalidad.

Pasos para escribir pruebas de aceptación:

1. Identificar todas las acciones en la historia.
2. Para cada acción escribir dos pruebas.
3. Para algunos datos, remplazar las entradas que hacen que la acción ocurra y llenar en la casilla resultados esperados los resultados obtenidos.

Para otros datos, remplazar las entradas que hacen que la acción falle, y registrar los resultados. (Letelier, Canós & Penadés, 2003).

Como resultado del trabajo realizado durante la fase de pruebas a continuación se ejemplifican algunas de las pruebas de aceptación:

Tabla 45: Caso de Prueba de Aceptación # 1.

Caso de Prueba de Aceptación	
Código: HU3-PF1	**Historia de Usuario (No. y Nombre):** HU.3 Registrar datos de Sección sindical.
Nombre: Registrar datos de Sección sindical.	
Descripción: Prueba para la funcionalidad de registrar los datos de una nueva sección sindical.	
Condiciones de ejecución: El usuario debe estar registrado en el sistema. El rol del usuario debe ser Administrador.	
Entrada/Pasos de ejecución: Se intenta registrar una nueva sección sindical con sus datos correspondientes (Nombre y Facultad).	
Resultado esperado: Se registra la nueva sección sindical.	
Evaluación de la prueba: Cumplida	

Tabla 46: Caso de Prueba de Aceptación # 2.

Caso de Prueba de Aceptación	
Código: HU5-PF2	**Historia de Usuario (No. y Nombre):** HU.5 Conformar Nuevo Potencial.
Nombre: Conformar nuevo potencial	
Descripción: Prueba para la funcionalidad de conformar un nuevo potencial.	
Condiciones de ejecución: El usuario debe estar registrado en el sistema. El rol del usuario debe ser Secretario General.	
Entrada/Pasos de ejecución: Se intenta generar un nuevo potencial que muestra toda la información de los trabajadores de la sección sindical y permite registrar un nuevo trabajador y afiliar el mismo a una sección sindical sino pertenece a ninguna.	
Resultado esperado: Se genera el nuevo potencial.	

Evaluación de la prueba: Cumplida.

Tabla 47: Caso de Prueba de Aceptación # 3.

	Caso de Prueba de Aceptación
Código: HU6-PF3	**Historia de Usuario (No. y Nombre):** HU.6 Registrar datos del trabajador.
Nombre: Registrar datos del trabajador	
Descripción: Prueba para la funcionalidad de registrar un nuevo trabajador a la sección sindical.	
Condiciones de ejecución: El usuario debe estar registrado en el sistema. El rol del usuario debe ser Secretario General.	
Entrada/Pasos de ejecución: Se intenta registrar un nuevo trabajador en la sección sindical con todos sus datos correspondientes (Nombre, Apellidos, Usuario, Solapín, Correo y Salario).	
Resultado esperado: Se registra el nuevo trabajador.	
Evaluación de la prueba: Cumplida.	

Tabla 48: Caso de Prueba de Aceptación # 4.

	Caso de Prueba de Aceptación
Código: HU11-PF4	**Historia de Usuario (No. y Nombre):** HU.11 Registrar acuerdo
Nombre: Registrar acuerdo	
Descripción: Prueba para la funcionalidad de registrar un acuerdo en la sección sindical.	
Condiciones de ejecución: El usuario debe estar registrado en el sistema. El rol del usuario debe ser Activista.	
Entrada/Pasos de ejecución: Se intenta registrar un nuevo acuerdo en la sección sindical con sus datos correspondientes(Denominación, Responsable y Fecha de cumplimiento)	
Resultado esperado: Se registra el nuevo acuerdo.	
Evaluación de la prueba: Cumplida.	

Después de aplicadas las pruebas de aceptación a las funcionalidades correspondientes a las historias de usuario 3, 5, 6 y 11. Las funcionalidades registrar datos de sección sindical y registrar acuerdo resultaron satisfactorias en una primera iteración. En el caso de las funcionalidades conformar nuevo potencial y registrar datos de un trabajador los resultados no fueron satisfactorios en su primera iteración. Después de realizados los cambios pertinentes se realizó una segunda iteración de pruebas a estas funcionalidades y se obtuvo un 100% de satisfacción por parte del cliente.

3.3 Conclusiones parciales

En este capítulo mediante la aplicación de las métricas de diseño TC y RC se determinó que el diseño propuesto no presenta una alta complejidad, evidenciándose que la mayoría de las clases poseen altos niveles de reutilización y bajo acoplamiento. Además se realizaron las pruebas de

caja negra cuyos resultados permitieron obtener una aplicación a nivel de interfaz lo más funcional posible, como resultado de las iteraciones realizadas. Además mediante la utilización de las pruebas unitarias se evaluó la calidad de los artefactos obtenidos así como las funcionalidades implementadas en la aplicación. Estas pruebas permitieron corregir una serie de funcionalidades las cuales fueron subsanadas, permitiendo que el sistema posea un correcto funcionamiento.

Conclusiones Generales

Con la realización del presente trabajo de diploma se desarrolló un sistema que facilita la celeridad de la información generada en las secciones sindicales de la Universidad de las Ciencias Informáticas. Es por ello que se puede concluir afirmando que:

- El estudio realizado de los sistemas existentes relacionados con las secciones sindicales permitió comprender la necesidad de desarrollar una aplicación capaz de gestionar los procesos que se realizan en las secciones sindicales de la universidad.
- La realización de las historias de usuarios facilitó una descripción detallada de las funcionalidades del sistema.
- El diseño de la solución permitió obtener un modelo que representa las clases que conformarán al sistema y con la elaboración de las tarjetas CRC se evidencian las relaciones entre ellas, de manera que se facilita la implementación de las funcionalidades de la aplicación.
- A partir de esta implementación se obtuvo un sistema capaz de satisfacer las necesidades del cliente.
- La ejecución de métricas y pruebas a la solución permitió asegurar la calidad y el correcto funcionamiento de la aplicación, de manera que se cumplan las expectativas del cliente.

Recomendaciones

Debido a los resultados de la investigación efectuada 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:

- La implementación de la funcionalidad de enviar correo a los atrasados en el pago de las cuotas sindicales.
- Adicionar otras funcionalidades al sistema desarrollado para facilitar el control del pago por parte de los directivos del sindicato de la universidad.

Bibliografía

1. **Aeaby, A. (1996).** Introduction to Programming Languages.
2. **Análisis y Diseño de un Nodo Virtual de Procesos**. *[Libro] / aut. Edith Maylen y Ortiz, Leydis A. - Habana, UCI: [s.n.], 2008.*
3. **Borrillo, R**. **(2012).** *Diseñando tu nuevo proyecto web con Bootstrap 2.0.* http://www.genbetadev.com/desarrollo-web/disenando-tu-nuevo-proyecto-web-con-bootstrap-2-0.
4. **Casanova, J. (2004).** *DesarrolloWeb.com. Usabilidad y arquitectura del software. About.* http://www.desarrolloweb.com/articulos/1622.php.
5. **Dirk, R. (2000).** *Framework Design: A Role Modeling Approach.* http://www.riehle.org/computer-science/research/dissertation/diss-a4.pdf.
6. **Eguiluz, J. (2011).** *Desarrollo web ágil con Symfony2.*
7. **Erwin. (2013).** *About. http://erwin.com/worlwide/spanish-latin-america*
8. **Estatutos. (2006).** *Estatutos aprobados por el XIX congreso de la CTC. Septiembre de 2006*
9. **Extjs**. **(2012).** *About. http://www.extjses.com/*
10. **Figueroa Roberth, Solís Camilo, Cabrera Armando. (2007).** *Metodologías tradicionales y metodologías ágiles.*
11. **Flores, Lic. Ervin. (2006).** *Metodología agiles Proceso Unificado Ágil (AUP).*
12. **Foundation, The Apache Software. (2011).** *Documentación del Servidor HTTP Apache 2.0. [En línea] 2011. [Citado el: 16 de Enero de 2012.] http://httpd.apache.org/docs/2.0/es/.*
13. **Gamma, Erich**., *Helm, R., Johnson, R., & Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. s.l. : Addison-Wesley Professional.*
14. **García, Félix Óscar Rubio and Santos, Crescencio Bravo.** (2009-2010). *Escuela de Ingeniería Civil Informática. [Online] 2009-2010.* http://www.eici.ucm.cl/Academicos/R_Villarroel/descargas/ing_sw_1/Metodologias.pdf.
15. **Loucopulos, P, Karacostas, V. (1995).** *System Requirements engineerinuality which will perform effectively.*
16. **Kniberg, Henrik. (2007).** *Scrum y XP desde las trincheras.pdf*
17. **Joaquín Seoane Pascual, Jesús M. González Barahona y Gregorio Robles.** (2003-2007). *[Citado el 20 de enero del 2013]. http://curso-sobre.berlios.de/introsobre/2.0.1/sobre.html/sec-ide.html.*
18. **Larman, Craig. (2000).** *UML y Patrones, 2da Edición.*
19. **Letelier, Patricio, Canós, José H. & Penadés, Mª Carmen.** (2003). *Metodologías ágiles en el desarrollo de software: Extreme Programming (XP)*
20. **Mendoza Sánchez, María A. (2004).** *Metodologías del desarrollo de software. Junio 7 del*

2004.

21. **NetBeans. (2012).** *NetBeans. [En línea] Oracle Corporation and/or its affiliates, 2012. [Citado el: 26 de Enero de 2012.] http://netbeans.org/.*
22. **Pilgrim, Mark. 2010.** *HTML5 Up and Running.* O'Reilly, 2010.
23. **Postgresql. (1996-2013).** *About.* http://www.postgresql.org/about/.
24. **Pressman, Roger. (2005).** *La Ingeniería del Software, un enfoque práctico.*
25. **Programming languages** on the internet. (2000-2012). http://www.webdevelopersnotes.com/basics/languages_on_the_internet.php3.
26. **Ramírez Martín, Carlos E. y Rodríguez Donatien, Ariagna.** (junio 2009). *Sistema para la Identificación de Aguas en Pozos Petroleros (SIAPP). Ciudad de La Habana: Universidad de las Ciencias Informáticas. Facultad 9.*
27. **S.A Solus, 2006**. *Guía de Usuario de Enterprise Architect 6.5 [Citado el 25 de enero del 2013]. http://www.sparxsystems.com.ar/EAUserGuide/ea.html*
28. **Schenone, Marcelo Hernán**. (2004). *Diseño de una Metodología Ágil de Desarrollo de Software.*
29. **Schwaber, K. 1ro de Julio del 2010.** Advanced Development Methods. *SCRUM Development Process* Retrieved.
30. **ScottGu.** (2008). *Jquery and Microsoft.* http://weblogs.asp.net/scottgu/archive/2008/09/28/jquery-and-microsoft.aspx.
31. **Sencha.** (2012). About http://www.sencha.com/
32. **Visual Paradigm** . (2012). *Visual Paradigm.* http://www.visual-paradigm.com./.
33. **Winesett, Jeffrey,** (2010). Agile Web Application Development with Yii1.1 and PHP5.
34. **Zarza Samuel,** (2011). *HERRAMIENTAS DE MODELADO DE DATOS GRATUITAS (PARA POSTGRESQL). [Citado el 25 de enero del 2013]. file:///H:/Docencia/Tesis/Documentos/Paginas/Herramientas-de-Modelado-de-datos-Gratuitas.html*
35. **Zend Technologies Ltd. 2006 – 2013.** [Citado el 15 de enero del 2013]. http://framework.zend.com/about/

Anexos

Historias de Usuarios

Historia de usuario	
Número: 1	**Nombre:** Registrar usuario
Usuario: Administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Alto (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir que se registren los usuarios: Secretario General, Financiero y Activista. El primero será registrado por el Administrador del Sistema y debe asociarse a una sección sindical, y los demás serán registrados por el propio Secretario General.	
Observaciones: Existe un Administrador general del sistema.	

Tabla 1: Historia de Usuario #1.

Historia de usuario	
Número: 2	**Nombre:** Autenticar Usuario
Usuario: Administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Alto (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir que un usuario se autentique, y que acceda a las funcionalidades de su interés. Los roles disponibles para los usuarios del sistema serán: Secretario General, Financiero y Activista.	
Observaciones: Existe un Administrador general del sistema.	

Tabla 2: Historia de Usuario #2.

Historia de usuario	
Número: 3	**Nombre:** Registrar datos de Sección sindical.
Usuario: Administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Alto (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir registrar una nueva sección sindical con los siguientes datos: Nombre, Facultad. Esta funcionalidad debe ser accesible solo para el Administrador del Sistema.	
Observaciones: Existe un Administrador general del sistema.	

Tabla 3: Historia de Usuario #3.

Historia de usuario	
Número: 4	**Nombre:** Modificar datos de Sección sindical.
Usuario: Administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir que se modifiquen los datos de una sección sindical.	
Observaciones: Existe un Administrador general del sistema.	

Tabla 4: Historia de Usuario #4.

Historia de usuario	
Número: 5	**Nombre:** Conformar Nuevo Potencial.
Usuario: Secretario General	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Alto (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir que el secretario general conforme el potencial para el año siguiente. Inicialmente se deberá mostrar a todos los afiliados del año precedente y las opciones: Modificar Salario, Agregar Trabajador, Aceptar y Cancelar. La opción Agregar Trabajador brindará la posibilidad de agregar un nuevo trabajador al sistema, o de afiliar a alguno que no estuviese afiliado por alguna razón. La opción Aceptar indicará que se terminó la elaboración del potencial.	
Observaciones: Existe un Secretario General por cada sección sindical.	

Tabla 5: Historia de Usuario #5.

Historia de usuario	
Número: 6	**Nombre:** Registrar datos del trabajador.
Usuario: Secretario General	**Iteración Asignada:** 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Alto (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir registrar un trabajador con los siguientes datos: Nombres, Apellidos, Nro. de Solapín, usuario, Salario Total, Sección Sindical donde debe pertenecer, Si está afiliado o no al sindicato.	
Observaciones: Existe un Secretario General por cada sección sindical.	

Tabla 6: Historia de Usuario #6.

Historia de usuario

Número: 7	**Nombre:** Modificar salario del trabajador
Usuario: Secretario General	**Iteración Asignada:** 1
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Bajo (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir que se modifique el salario total de un trabajador.	
Observaciones: Existe un Secretario General por cada sección sindical.	

Tabla 7: Historia de Usuario #7.

Historia de usuario	
Número: 8	**Nombre:** Buscar Trabajador
Usuario: Secretario General	**Iteración Asignada:** 1
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Bajo (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir buscar un trabajador con alguno de los siguientes datos: Nombres, Apellidos, Nro. de Solapín, usuario, Salario Total, Sección Sindical donde debe pertenecer.	
Observaciones: Existe un Secretario General por cada sección sindical.	

Tabla 8: Historia de Usuario #8.

Historia de usuario	
Número: 9	**Nombre:** Dar Alta a un trabajador
Usuario: Secretario General	**Iteración Asignada:** 1
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir afiliar a un trabajador no afiliado a ninguna otra sección sindical. Si el trabajador no se encuentra en el sistema, debe ser registrado con los siguientes datos: Nombres, Apellidos, Nro. de Solapín, usuario, Salario Total, Sección Sindical. Si se encuentra registrado el trabajador, solo se afilia si no está afiliado a ninguna Sección Sindical. Se deben registrar el motivo del alta y la fecha en que se produjo.	
Observaciones: Esta funcionalidad debe ser utilizada, una vez que ya están conformados los potenciales, y se desea afiliar a la sección sindical un nuevo trabajador.	

Tabla 9: Historia de Usuario #9.

Historia de usuario	
Número: 10	**Nombre:** Dar Baja a un trabajador

Usuario: Secretario General	**Iteración Asignada:** 1
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir dar baja a un trabajador de su sección sindical. Ese trabajador no estará más afiliado a dicha sección sindical. Se deben registrar el motivo de la baja y la fecha en que se produjo.	
Observaciones: Existe un Secretario General por cada sección sindical.	

Tabla 10: Historia de Usuario #10.

Historia de usuario	
Número: 11	**Nombre:** Registrar acuerdo
Usuario: Activista	**Iteración Asignada:** 1
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir registrar acuerdos de una sección sindical con los siguientes datos: Denominación del Acuerdo, Responsable, Fecha de cumplimiento.	
Observaciones: Existe un Activista por cada sección sindical.	

Tabla 11: Historia de Usuario #11.

Historia de usuario	
Número: 12	**Nombre:** Registrar información sobre el cumplimiento de acuerdo
Usuario: Activista	**Iteración Asignada:** 1
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir registrar si un acuerdo fue cumplido, incumplido o pospuesto.	
Observaciones: Existe un Activista por cada sección sindical.	

Tabla 12: Historia de Usuario #12.

Historia de usuario	
Número: 13	**Nombre:** Obtener reporte de acuerdos
Usuario: Activista	**Iteración Asignada:** 1
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana

Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe generar un reporte de acuerdos, especificando un rango de fechas y si se desean visualizar acuerdos cumplidos, incumplidos o pospuestos. Se debe registrar además la causa del incumplimiento de un acuerdo.	
Observaciones: Existe un Activista por cada sección sindical.	

Tabla 13: Historia de Usuario #13.

Historia de usuario	
Número: 14	**Nombre:** Registrar el mes de compromiso de pago de la MTT de un afiliado.
Usuario: Financiero	**Iteración Asignada:** 2
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir registrar el mes en que cada afiliado se compromete a pagar las MTT.	
Observaciones: Existe un Financiero por cada sección sindical.	

Tabla 14: Historia de Usuario #14.

Historia de usuario	
Número: 15	**Nombre:** Registrar el mes en que un afiliado efectuó el pago de la MTT.
Usuario: Financiero	**Iteración Asignada:** 2
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir registrar el mes en cada afiliado para pagar las MTT.	
Observaciones: Existe un Financiero por cada sección sindical.	

Tabla 15: Historia de Usuario #15.

Historia de usuario	
Número: 16	**Nombre:** Registrar meses que ha pagado un afiliado de la cuota sindical.
Usuario: Financiero	**Iteración Asignada:** 2
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir registrar el o los meses que ha pagado un afiliado la cuota sindical y la fecha en que realizó el pago.	

Observaciones: Existe un Financiero por cada sección sindical.

Tabla 16: Historia de Usuario #16.

Historia de usuario	
Número: 17	**Nombre:** Obtener reporte de los afiliados que están atrasados en el pago de la cuota sindical.
Usuario: Financiero	**Iteración Asignada:** 2
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir obtener un listado de todos los afiliados que no han pagado la cuota sindical al menos en el mes anterior al actual. Se debe mostrar el nombre y los apellidos del afiliado y el mes en que efectuó el último pago.	
Observaciones: Existe un Financiero por cada sección sindical.	

Tabla 17: Historia de Usuario #17.

Historia de usuario	
Número: 18	**Nombre:** Obtener reporte de los afiliados que están atrasados en el pago de la MTT.
Usuario: Financiero	**Iteración Asignada:** 2
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir obtener un listado de todos los afiliados que no han pagado la MTT pasado al menos un mes después del mes en que hicieron el compromiso. Se debe mostrar el nombre y los apellidos del afiliado y el mes de compromiso.	
Observaciones: Existe un Financiero por cada sección sindical.	

Tabla 18: Historia de Usuario #18.

Historia de usuario	
Número: 19	**Nombre:** Obtener un reporte de los afiliados que terminaron de pagar el año completo.
Usuario: Financiero	**Iteración Asignada:** 2
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir obtener un listado de los afiliados que pagaron las 12 cuotas sindicales y la MTT. Se debe mostrar el nombre y los apellidos del afiliado.	

Observaciones: Existe un Financiero por cada sección sindical.

Tabla 19: Historia de Usuario #19.

Historia de usuario	
Número: 20	**Nombre:** Obtener un reporte sobre el estado actual de las finanzas.
Usuario: Financiero	**Iteración Asignada:** 2
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir generar un reporte que contenga: los nombres y apellidos de todos los afiliados, el Total de meses pagados, el Total de meses por pagar. Además los que estén atrasados deben estar resaltados en rojo.	
Observaciones: Existe un Financiero por cada sección sindical.	

Tabla 20: Historia de Usuario #20.

Historia de usuario	
Número: 21	**Nombre:** Obtener un reporte de los afiliados que han causado alta en una sección sindical.
Usuario: Secretario General	**Iteración Asignada:** 2
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir obtener un reporte de los afiliados que han resultado alta durante el año en curso. Se debe mostrar el nombre y los apellidos, la fecha en que causaron alta en la sección sindical y la causa.	
Observaciones: Existe un Secretario General por cada sección sindical.	

Tabla 21: Historia de Usuario #21.

Historia de usuario	
Número: 22	**Nombre:** Obtener un reporte de los afiliados que han causado baja en una sección sindical.
Usuario: Secretario General	**Iteración Asignada:** 2
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir obtener un reporte de los afiliados que han resultado baja durante el año en curso. Se debe mostrar el nombre y los apellidos, y la fecha en que causaron baja en la sección sindical y la causa.	
Observaciones: Existe un Secretario General por cada sección sindical.	

Tabla 22: Historia de Usuario #22.

Historia de usuario	
Número: 23	**Nombre:** Obtener un reporte de los afiliados que estuvieron atrasados en el pago de la cuota sindical o de la MTT en algún momento del año.
Usuario: Financiero	**Iteración Asignada:** 2
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir obtener un reporte con todos los afiliados que alguna vez durante el año estuvieron atrasados en el pago de la cuota sindical y la MTT.	
Observaciones: Existe un Financiero por cada sección sindical.	

Tabla 23: Historia de Usuario #23.

Historia de usuario	
Número: 24	**Nombre:** Registrar trabajadores destacados por trimestre
Usuario: Activista	**Iteración Asignada:** 3
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir registrar cuáles trabajadores de una sección sindical resultaron destacados en un trimestre específico (enero-feb-marzo, abril-mayo-junio, julio-agosto-sept, oct-nov-dic).	
Observaciones: Existe un Activista por cada sección sindical.	

Tabla 24: Historia de Usuario #24.

Historia de usuario	
Número: 25	**Nombre:** Obtener un reporte con los posibles candidatos a vanguardia de la sección sindical.
Usuario: Activista	**Iteración Asignada:** 3
Prioridad en Negocio: Media (Alta / Media / Baja)	**Puntos Estimados:** 1 semana
Riesgo en Desarrollo : Medio (Alto / Medio / Bajo)	**Puntos Reales:** 1 semana
Descripción: El sistema debe permitir obtener un reporte con los posibles candidatos a vanguardia, que no son más que los trabajadores que al menos resultaron destacados en un trimestre.	
Observaciones: Existe un Activista por cada sección sindical.	

Tabla 25: Historia de Usuario #25.

Interfaz de sistema

Interfaz de sistema

Glosario

[i] **Modelo A1:** Modelo imprescindible en el funcionamiento de la CTC ya que recoge la información de la base de las estructuras que la conforman.

[ii] **La Alianza Ágil:** Organización dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía ágil.

[iii] **UML:** (Unified Modeling Languague) Lenguaje de Modelado Unificado. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Es el lenguaje de modelado de sistemas software más conocido y utilizado en la actualidad.

[iv] **IDE:** Entorno de Desarrollo Integrado. Es un programa informático que contiene un conjunto de herramientas de programación.

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

[vi] **Bundle:** Estructura de directorios generado en los marco de trabajo.

[vii] **Doctrine:** Es un Mapeador de Objeto Relacional (ORM) creado para PHP.

[viii] **Licencia 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.